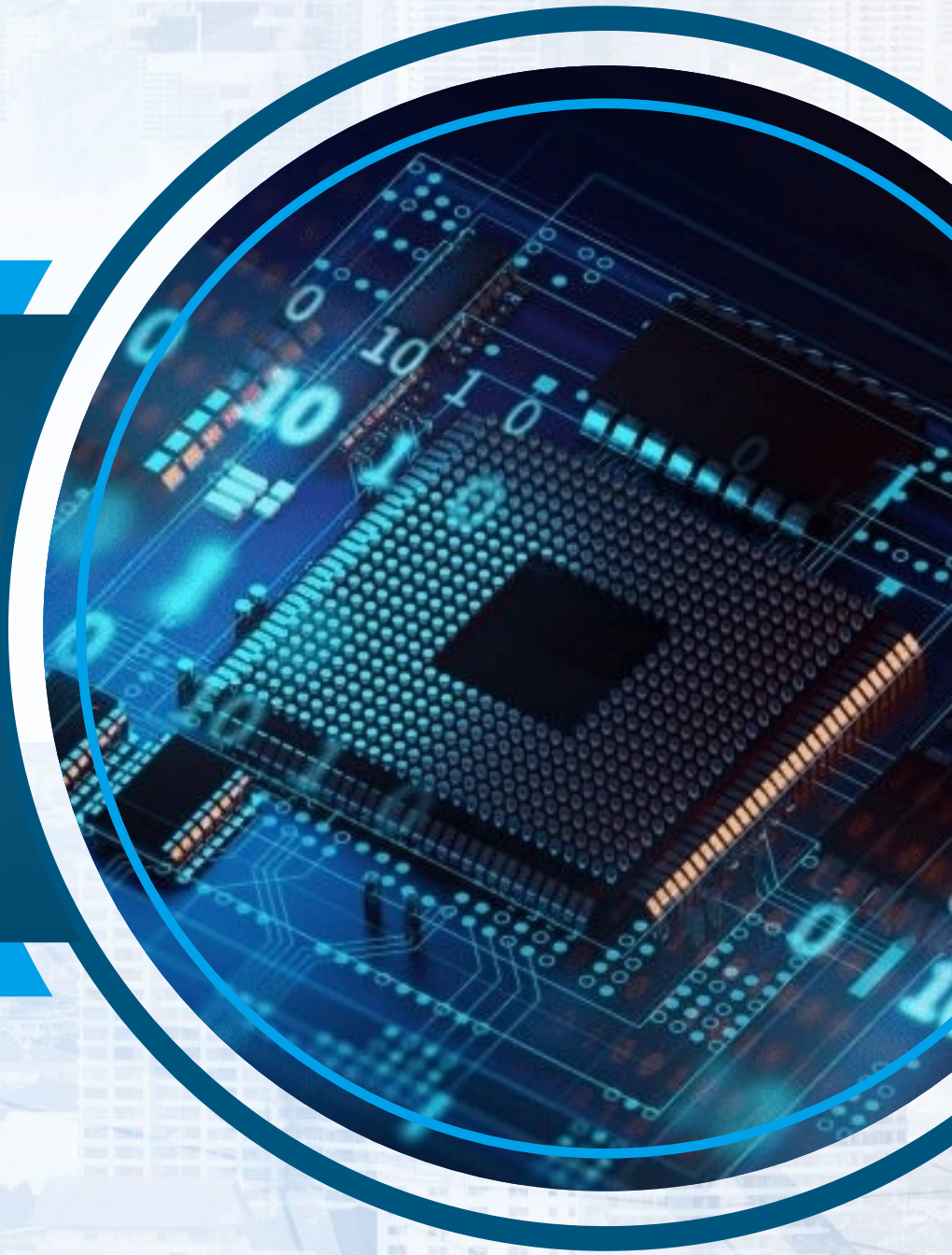
YAYASAN PRIMA AGUS TEKNIK



PROYEK PRAKTIS ARDUINO UNTUK IoT (Internet of Things)

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

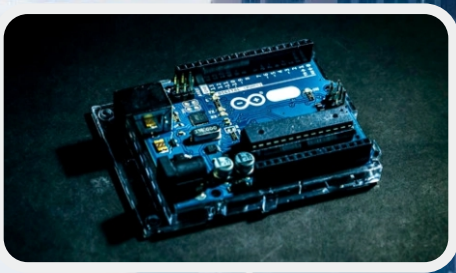
Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

PROYEK PRAKTIS ARDUINO UNTUK IoT (Internet of Things)

BIO DATA PENULIS

Penulis memiliki berbagai disiplin ilmu yang diperoleh dari Universitas Diponegoro (UNDIP) Semarang. dan dari Universitas Kristen Satya Wacana (UKSW) Salatiga. Disiplin ilmu itu antara lain teknik elektro, komputer, manajemen dan ilmu sosiologi. Penulis memiliki pengalaman kerja pada industri elektronik dan sertifikasi keahlian dalam bidang Jaringan Internet, Telekomunikasi, Artificial Intelligence, Internet Of Things (IoT), Augmented Reality (AR), Technopreneurship, Internet Marketing dan bidang pengolahan dan analisa data (komputer statistik).

Penulis adalah pendiri dari Universitas Sains dan Teknologi Komputer (Universitas STEKOM) dan juga seorang dosen yang memiliki Jabatan Fungsional Akademik Lektor Kepala (Associate Professor) yang telah menghasilkan puluhan Buku Ajar ber ISBN, HAKI dari beberapa karya cipta dan Hak Paten pada produk IPTEK. Penulis juga terlibat dalam berbagai organisasi profesi dan industri yang terkait dengan dunia usaha dan industri, khususnya dalam pengembangan sumber daya manusia yang unggul untuk memenuhi kebutuhan dunia kerja secara nyata.



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-5734-32-3 (PDF)

Dr. Ir. Agus Wibowo, M.Kom, M.Si, MM

PROYEK PRAKTIS ARDUINO UNTUK IoT (Internet of Things)

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Proyek Praktis Arduino untuk IoT (Internet of Things)

Penulis :
Dr. Ir. Agus Wibowo, M.Kom., M.Si., MM.

ISBN : 9 786235 734323

Editor :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur pada Tuhan Yang Maha Esa bahwa buku yang berjudul "Proyek Praktis Arduino untuk IoT (*Internet of Things*)". Menurut satu analisis, lebih dari 50 miliar perangkat terhubung pada tahun 2020, dan total pendapatan *Internet of Things* (IoT) akan sedikit di atas Rp 15 triliun. Angka-angkanya terlihat luar biasa, tetapi apa sebenarnya IoT itu? Apakah hanya itu yang terhubung ke internet? Mengapa yang terhubung itu penting?

IoT lebih dari sekadar hal-hal yang terhubung ke Internet. IoT adalah tentang membuat hal-hal bodoh menjadi lebih pintar dengan memberi mereka kemampuan untuk dapat merasakan, berkomunikasi, dan merespon. Kita memiliki lima indera—kita bisa melihat, mendengar, mengecap, mencium, dan menyentuh. Demikian pula jika Anda menambahkan sensor ini ke hal-hal, mereka juga dapat melakukan hal yang sama. Misalnya dengan menggunakan kamera benda dapat melihat, menggunakan alat pendeteksi suara benda dapat mendengar, dan menggunakan speaker benda dapat berbicara.

Ada begitu banyak sensor lain yang dapat digunakan untuk melakukan lebih banyak hal daripada kita. Dengan menghubungkan hal-hal ini ke Internet, mereka dapat berkomunikasi dengan kita, dengan hal-hal lain, dan perbatasan berikutnya di mana mereka dapat menggunakan kecerdasan buatan untuk berpikir juga. Ada banyak aplikasi IoT, tetapi berikut adalah beberapa contoh untuk lebih memahami bagaimana IoT digunakan untuk meningkatkan kehidupan kita.

Buku ini dibagi menjadi dua bagian. Bagian pertama menerangkan dasar-dasar dalam membangun aplikasi IoT dan bagian kedua mengikuti pendekatan berbasis proyek. Di akhir setiap bab, Anda akan memiliki prototipe aplikasi IoT yang memiliki fungsi. Bagian pertama mencakup 3 Bab awal. Bab 1 menjelaskan tentang pengetahuan dasar arduino, memperkenalkan platform prototipe Arduino, yang digunakan di seluruh buku ini. Kemudian Bab 2 menerangkan tentang Konektivitas Internet, membahas berbagai opsi yang tersedia untuk menghubungkan berbagai hal ke Internet. Bagian ke 1 akhir yaitu Bab 3 yang akan menerangkan tentang Protokol Komunikasi, mengajarkan pembaca apa itu protokol komunikasi dan apa saja yang tersedia untuk IoT.

Sedangkan Bagian ke 2 dalam buku ini berisikan tentang Prototipe yang memungkinkan dibangunnya aplikasi IoT, yaitu mencakup Bab 4 sampai dengan Bab 12 yang akan menerangkan membangun prototipe aplikasi IoT. Bab 4 memperkenalkan Node-RED, yang merupakan desainer visual yang membantu mengurangi jumlah kode yang diperlukan untuk aplikasi IoT. Pada Bab 5 berbicara tentang komponen yang diperlukan untuk membangun aplikasi IoT yang menyediakan data kepada pengguna secara real time. Bab 6, membahas komponen aplikasi IoT yang dapat mengontrol berbagai hal dari jarak jauh, dalam bab ini mencakup sistem kontrol pencahayaan. Pada Bab 7 menunjukkan kepada pembaca berbagai komponen yang terlibat dalam membangun aplikasi IoT sesuai permintaan. Pembaca akan menerangkan cara membangun *smarter parking system* di bab ini.

Pada Bab 8 mengajarkan pembaca tentang sistem pemantauan suhu berbasis web. Bab 9 membahas pentingnya lokasi perangkat atau mengembangkan sistem pelacakan ternak. Untuk Bab 10 berbicara tentang skenario di mana respons manusia diperlukan; sebagai contoh pembaca akan membangun sistem pengelolaan sampah. Bab 11 akan membahas pola IoT yang akan menjadi sangat populer seiring dengan semakin pintarnya berbagai hal. Contohnya adalah sistem konservasi energi. Sedangkan Bagian akhir adalah Bab 12 memperkenalkan pembaca ke platform IoT yang membantu mempercepat masuk ke IoT. Contoh dalam bab ini membangun sistem kontrol kelembaban tanah.

Buku ini ditujukan untuk para pemilik hobi dan profesional yang ingin memasuki dunia IoT. Materi dalam buku ini memperkenalkan pengetahuan tentang Arduino atau perangkat serupa dan pengalaman pemrograman. Penulis menggunakan komponen hardware dasar dan memberikan petunjuk langkah demi langkah untuk membangun sirkuit. Akhir kata semoga buku ini berguna bagi para pembaca.

Semarang, Januari 2022
Penulis

Dr Agus Wibowo, M.Kom, M.Si, MM

DAFTAR ISI

BAGIAN 1
BUILDING BLOCK
BAB 1
DASAR ARDUINO

Arduino adalah platform *open-source* yang terdiri dari *hardware* dan *software* yang sangat sederhana dan mudah digunakan. Singkatnya Arduino dapat membaca data sensor dan komponen kontrol seperti lampu, motor, termostat, dan pintu garasi Anda, terutama dikembangkan untuk tujuan pembuatan prototipe, jadi sangat cocok untuk buku pemula IoT ini.

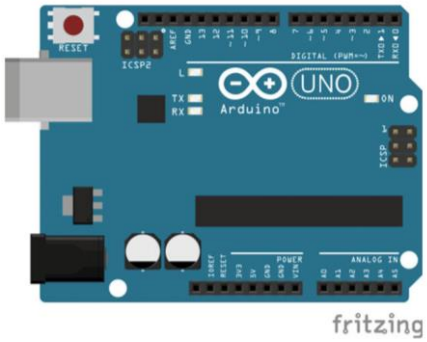
1.1 TUJUAN PEMBELAJARAN

Pada akhir bab ini, Anda akan dapat:

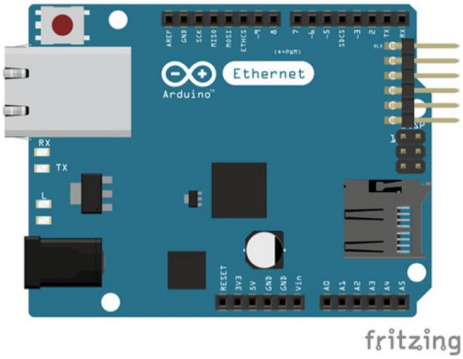
- Gunakan hardware Arduino
- Gunakan Arduino IDE
- Tulis, Upload, dan jalankan program dasar Arduino

1.2 PERSYARATAN HARDWARE

Arduino hadir dalam berbagai model (juga dikenal sebagai *board*). Setiap *board* memiliki spesifikasi yang berbeda. Jika papan Anda tidak dilengkapi dengan fitur yang Anda cari, maka Anda selalu memiliki opsi untuk menambahkan shield yang mendukung fitur yang diperlukan. Di dunia Arduino, shield sangat mirip dengan papan, tetapi hanya mendukung fungsi tertentu seperti kemampuan untuk terhubung ke jaringan WiFi atau kemampuan untuk mengontrol motor servo.



Gambar 1-1. Arduino Uno



Gambar 1-2. Ethernet shield

Shield bertindak sebagai tambahan; yaitu, secara fisik terpasang ke bagian atas Arduino board. Setelah terpasang, *Arduino board* menjadi mampu menangani fitur shield juga. Gambar 1-1 menunjukkan diagram Arduino Uno, sedangkan Gambar 1-2 menunjukkan diagram ethernet shield.

***Catatan** : Bagian-bagian yang ada disini akan bervariasi berdasarkan Arduino board yang Anda pilih.*

Listing berikut merangkum beberapa bagian penting dari papan yang telah digunakan dalam proyek-proyek di seluruh buku ini:

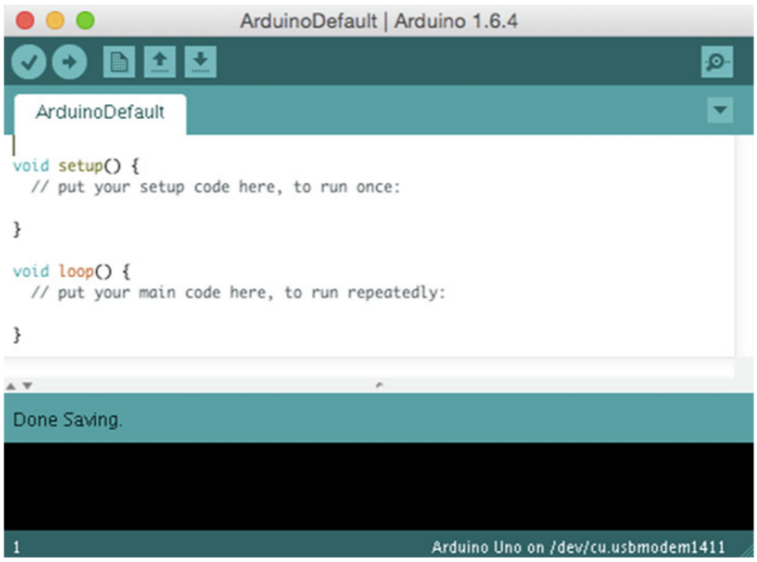
- **Digital Pin:** Total ada 14 pin digital pada Arduino Uno. Pin digital dapat berupa INPUT dan OUTPUT, tetapi statusnya hanya bisa HIGH atau LOW. HIGH berarti ada arus dan LOW berarti tidak ada arus. Contoh penggunaan pin digital adalah menyalakan atau mematikan lampu LED. Untuk menyalakannya, pin digital harus diatur ke HIGH dan untuk mematikannya pin digital harus diatur ke LOW.
- **Analog Pin:** Arduino Uno mendukung enam pin analog, A0 hingga A5. Tidak seperti pin digital, pembacaan pin analog dapat berkisar dari 0 hingga 1023. Contoh yang baik dari sensor yang menyediakan pembacaan analog adalah sensor kelembaban tanah. Kisaran membantu mengidentifikasi berapa banyak kelembaban yang tersisa di tanah.
- **USB connector**: *USB connector* memungkinkan Anda menghubungkan Arduino ke komputer, memberi daya pada board, mengUpload kode, dan menerima log pada monitor serial.
- **Daya baterai:** Aplikasi IoT yang perlu ditempatkan di lokasi terpencil akan membutuhkan sumber daya sendiri. Anda dapat menggunakan konektor daya baterai untuk memberi daya pada board.

Buku ini menggunakan Arduino Uno untuk semua proyek. Arduino Uno dikategorikan sebagai papan entry-level yang paling cocok untuk pemula. Meskipun buku menggunakan Arduino Uno, Anda tidak diharuskan menggunakannya; Anda dapat memilih salah satu Arduino board untuk menyelesaikan proyek dalam buku ini. Karena buku ini tentang Internet hal, konektivitas Internet merupakan persyaratan penting. Apa pun Arduino board yang Anda putuskan untuk digunakan, pastikan bahwa itu mendukung konektivitas Internet dalam beberapa bentuk. Arduino board harus dilengkapi dengan opsi konektivitas Internet bawaan atau Anda harus memiliki pelindung konektivitas Internet yang diperlukan.

***Catatan**: Arduino Uno tidak dilengkapi dengan dukungan konektivitas Internet built-in, jadi dalam buku ini Ethernet dan WiFi shield telah digunakan. Di sisi lain, model Arduino yang lebih canggih yang disebut Yún mendukung konektivitas Ethernet dan WiFi bawaan. Bab 2 membahas konektivitas Internet secara lebih rinci.*

1.3 PERSYARATAN SOFTWARE

Arduino menyediakan bahasa ‘C’ untuk memprogram Arduino board. Anda akan menggunakan Arduino IDE untuk menulis kode dan mengUploadnya ke Arduino board. Anda dapat menginstal Arduino IDE versi terbaru dari https://www.arduino.cc/en/Main/Software. Setelah Arduino IDE diinstal pada mesin Anda, bukalah seperti yang ditunjukkan pada Gambar 1-3, anda akan melihat Arduino mememuat dengan kode default.



Gambar 1-3. Tampilan default Arduino IDE

Ada tiga komponen Arduino IDE yang dirujuk dalam setiap bab buku ini.

Toolbar

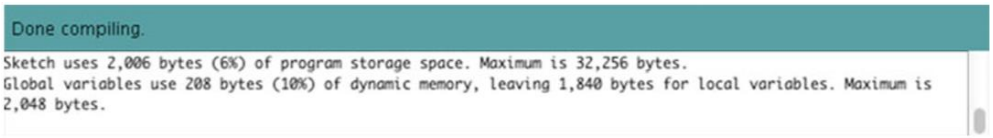
Toolbar di atas IDE, seperti yang ditunjukkan pada Gambar 1-4. Toolbar menyediakan akses mudah ke opsi yang sering digunakan.

Gambar 1-4. Toolbar Arduino IDE

- **Verify/Compile**: Ini adalah tombol pertama dari kiri (tanda centang). Tombol ini untuk memverifikasi dan mengkompilasi kode Anda untuk kebenaran. Anda dapat melihat hasilnya di jendela Status di bagian bawah.
- **Upload**: Ini adalah tombol kedua dari kiri (panah penunjuk kanan). Jika Arduino board Anda terhubung ke mesin Anda yang sedang menjalankan Arduino IDE, ini akan mengUpload kode di Arduino board. Anda dapat melihat hasil penerapan di jendela Status di bagian bawah.
- **New/Open/Save**: Tiga tombol berikutnya, seperti namanya, memungkinkan Anda membuka jendela kode baru, membuka file kode yang ada, atau menyimpan kode yang sedang dibuka. File kode Arduino memiliki ekstensi *.ino.
- **Serial/Monitor**: Tombol terakhir di sebelah kanan memungkinkan Anda membuka jendela Serial Monitor.

Status Window

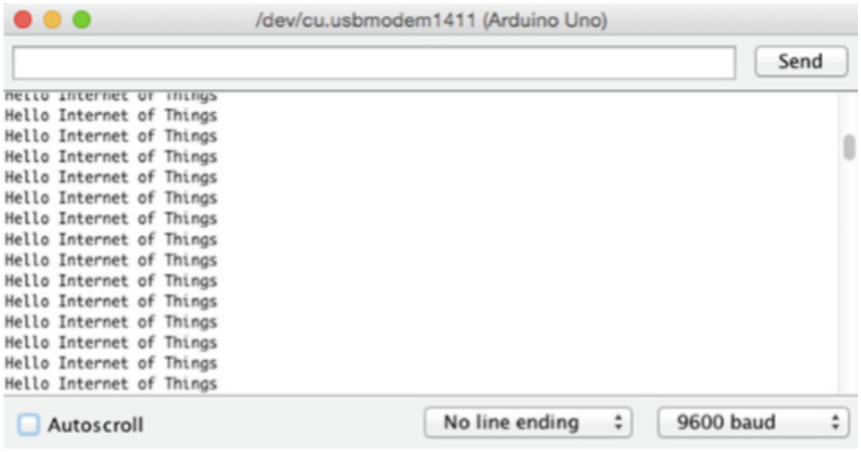
Saat Anda memverifikasi kode atau mengUploadnya ke board, Status Window yang ditunjukkan pada Gambar 1-5 mencantumkan semua hasil. Setiap kesalahan yang terjadi selama verifikasi atau pengUploadan kode akan ditampilkan di Status Window.



Gambar 1-5. Status window Arduino IDE

Serial Monitor Window

Serial Monitor Window yang ditunjukkan pada Gambar 1-6 mencetak semua pesan log yang dihasilkan oleh fungsi Serial.print() dan Serial.println() dalam kode. Untuk mencetak pesan apa pun pada jendela Serial Monitor, Anda harus terlebih dahulu menginisialisasi pesan dalam kode.



Gambar 1-6. Log pesan di jendela Serial Monitor

1.4 REFERENSI BAHASA PEMROGRAMAN ARDUINO

Bahasa pemrograman Arduino memiliki beberapa konstruksi. Namun, bab ini memberikan dasar-dasar yang telah digunakan di seluruh proyek dalam buku ini; lihat Tabel 1-1.

Tabel 1-1. Referensi Bahasa

Code Construct	Deskripsi
int	Nilai bilangan bulat, seperti 123
float	Nilai desimal, seperti 1,15
char[]	Nilai string, seperti "Arduino"
HIGH	Pin digital dengan arus
LOW	Pin digital tanpa arus
INPUT	Pin hanya bisa dibaca
OUTPUT	Pin hanya dapat disetel
A0 – A7	Konstanta untuk pin analog; bervariasi menurut papan
0 – 13	Nilai untuk pin digital; bervariasi menurut papan
analogRead()	Mengembalikan nilai pin analog (0 – 1023)
analogWrite(...)	Menyetel nilai pin analog
digitalRead()	Mengembalikan nilai pin digital (HIGH atau LOW)
digitalWrite(...)	Mengatur nilai pin digital (HIGH atau LOW)
Serial.benin()	Inisialisasi monitor serial
Serial.print()	Pesan log pada monitor serial
Serial.println()	Pesan log pada monitor serial dengan saluran baru
Delay(ms)	Menambahkan waktu tunggu dalam pemrosesan
Setup	Fungsi standar Arduino dipanggil sekali
Loop()	Fungsi Arduino standar dipanggil berulang kali
If	Memeriksa kondisi benar/salah
If ... else	Memeriksa kondisi benar/salah; jika salah pergi ke yang lain
//	Komentar satu baris

/* */	Komentar multibaris
#define	Mendefinisikan konstanta
#include	Termasuk perpustakaan eksternal

Anda dapat menjelajahi bahasa lengkapnya di https://www.arduino.cc/en/Reference. Arduino IDE menyediakan interface yang sangat sederhana dan bersih untuk menulis kode. Biasanya Anda dapat menyusun kode Anda dalam tiga bagian:

- **External libraries**: Termasuk semua library yang diperlukan. Library adalah bagian kode yang dikembangkan dan diuji sepenuhnya yang dapat Anda sertakan dan gunakan dalam kode Anda. Misalnya, jika Anda ingin berkomunikasi melalui Internet menggunakan koneksi Ethernet, alih-alih menulis semua kode itu dari awal, Anda cukup mengimpor dan menyertakan library Ethernet menggunakan #include <Ethernet.h>.
- **Konstanta dan variabel** : Mendefinisikan semua konstanta dan variabel yang akan digunakan untuk membaca dan memanipulasi data. Konstanta tidak berubah, jadi Anda dapat, misalnya, menggunakannya untuk nomor port di papan. Variabel dapat berubah, sehingga dapat digunakan untuk membaca data sensor.
- **Fungsi** : Menyediakan implementasi semua fungsi kustom dan standar. Sebuah fungsi merangkum fungsi tertentu. Disarankan untuk memasukkan kode Anda ke dalam fungsi, terutama ketika Anda ingin menggunakan kembali potongan kode itu. Fungsi membantu menghindari duplikasi kode.

Listing 1-1 memberikan contoh kode yang disusun menurut poin yang dibahas sebelumnya.

Listing 1-1. Struktur Kode yang Direkomendasikan

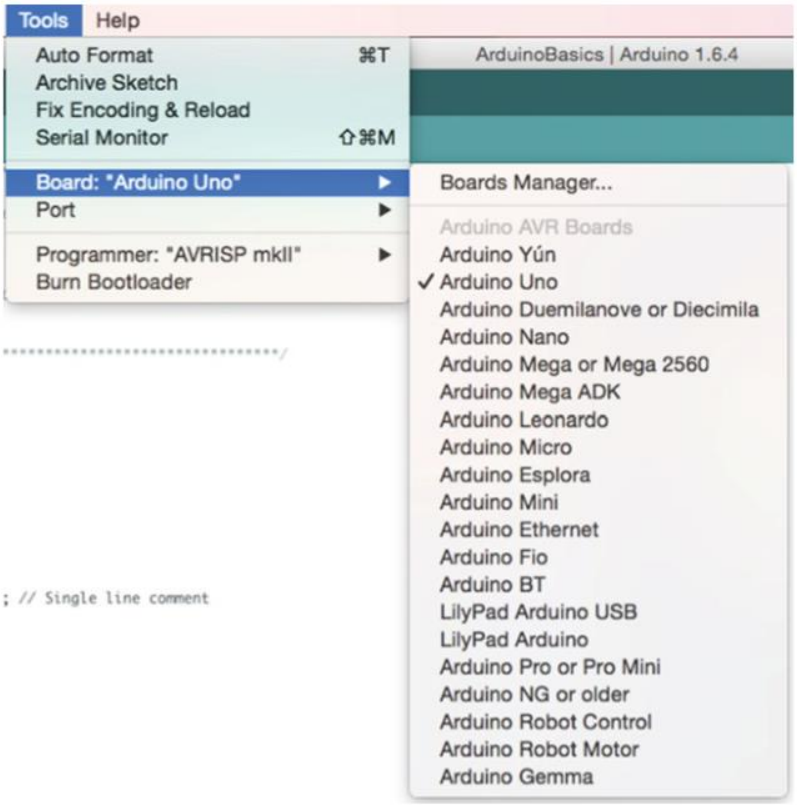
```
/*
* External Libraries
*/
#include <SPI.h>
/*
* Constants & Variables
*/
char message[] = “Hello Internet of Things”;
// Single line comment
/*
* Custom & Standard Functions
*/
void printMessage()
{
Serial.println(message);
}
void setup()
{
        // Initialize serial port
        Serial.begin(9600);
}
void loop()
{
        printMessage();
        delay(5000);
}
```

Listing 1-1 terdiri dari tiga fungsi. Ini memiliki dua fungsi Arduino standar, yang disebut setup() dan loop(), yang secara otomatis dipanggil oleh Arduino setelah kode diUpload. Oleh karena itu, mereka harus ada agar kode dapat dijalankan. Yang ketiga adalah fungsi kustom yang disebut printMessage() yang hanya mencetak pesan ke jendela Serial Monitor yang ditunjukkan pada Gambar 1-6

Fungsi setup() dipanggil hanya sekali. Inisialisasi dilakukan dalam fungsi ini termasuk inisialisasi monitor serial menggunakan kode Serial.begin(9600). Fungsi loop(), seperti namanya, berjalan dalam loop berkelanjutan. Pemrosesan pasca-inisialisasi seperti membaca data sensor dapat dilakukan dalam fungsi ini. Fungsi loop() memanggil fungsi printMessage() dan kemudian menunggu 5.000 milidetik sebelum diulang.

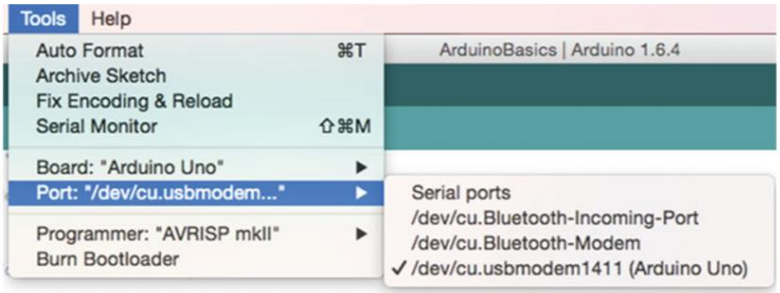
1.5 EKSEKUSI KODE ARDUINO

Mulai Arduino IDE Anda dan ketik kode yang disediakan di Listing 1-1. Klik tombol **Verify** untuk mengkompilasi dan memeriksa kode. Selanjutnya, dengan menggunakan kabel USB yang disertakan dengan Arduino Anda, hubungkan Arduino Anda ke komputer yang sedang menjalankan Arduino IDE. Setelah Arduino terhubung ke komputer Anda, seperti yang ditunjukkan pada Gambar 1-7, klik **Tools** lalu pilih **Board** dan pilih **Arduino Uno** (atau board mana saja yang Anda gunakan). Ini akan menginformasikan Arduino IDE tentang tempat kodeboard akan diUpload.



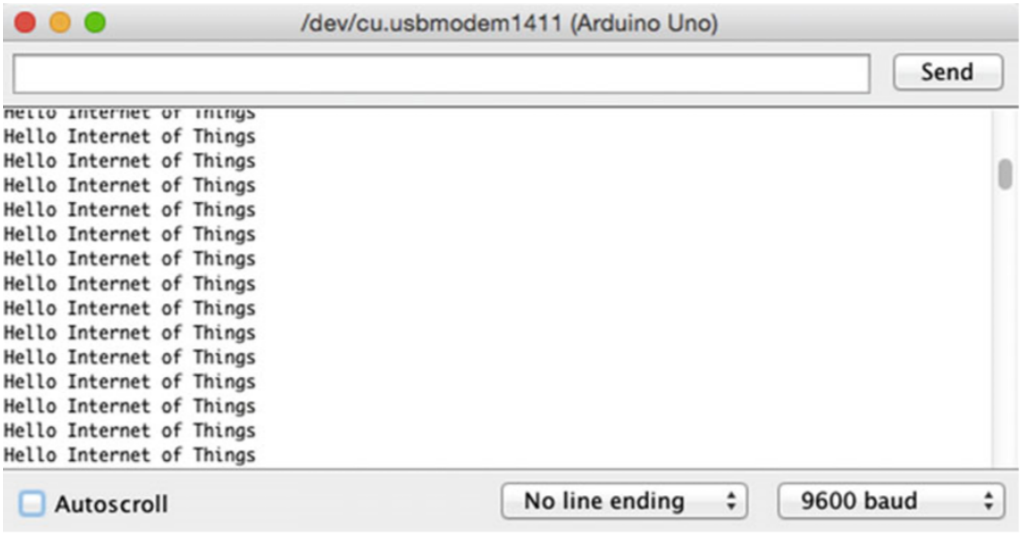
Gambar 1-7. Pilih Arduino board

Anda juga harus memilih port yang akan digunakan untuk mengUpload kode. Seperti yang ditunjukkan pada Gambar 1-8 dari **Tools** lalu klik **Port**, pilih **port USB** yang menghubungkan Arduino ke komputer Anda.



Gambar 1-8. Pilih port Arduino

Terakhir, klik tombol **Upload** dan **Open jendela Serial Monitor.** Pastikan nilai yang dipilih dalam tarik-turun Serial Monitor sama dengan nilai yang ditetapkan dalam fungsi Serial.begin(). Dalam hal ini, ini adalah 9600 dalam kode, jadi 9600 baud perlu dipilih di dropdown Serial Monitor. Jika tidak, Anda tidak akan dapat melihat pesan log. Seperti yang ditunjukkan pada Gambar 1-9, Anda akan mulai melihat pesan log di jendela Serial Monitor dengan interval 5.000 milidetik.



Gambar 1-9. Pesan log dari kode di jendela Serial Monitor

1.6 RINGKASAN

Dalam bab ini Anda telah mempelajari dasar-dasar hardware dan software Arduino. Anda juga sudah mempelajari konstruksi kode umum dari bahasa pemrograman Arduino, yang akan digunakan di seluruh buku ini. Bab bukan referensi lengkap Arduino, buku ini hanya menyediakan dasar-dasar yang diperlukan untuk menyelesaikan semua proyek yang akan dipelajari dalam buku ini. Untuk mempelajari lebih lanjut tentang Arduino, kunjungi situs web resmi di https://www.arduino.cc.

BAB 2
KONEKTIVITAS INTERNET

Semua perangkat IoT memerlukan mekanisme untuk mengirim atau menerima data. Ada banyak opsi yang tersedia untuk menghubungkan perangkat ke Internet, termasuk opsi kabel dan nirkabel, Bluetooth, jaringan seluler, dan banyak lagi lainnya. Opsi yang Anda pilih bergantung pada berbagai faktor, seperti:

- Skala dan ukuran jaringan tempat aplikasi akan dijalankan
- Jumlah data yang perlu diproses dan ditransfer
- Lokasi fisik perangkat

Tabel 2-1 mencantumkan beberapa pilihan konektivitas Internet dengan contoh Arduino yang telah digunakan.

Tabel 2-1. Opsi Konektivitas Internet untuk Perangkat IoT

Opsi	Contoh
Wired (Ethernet)	Pemantauan suhu penyimpanan makanan
Wireless (WiFi)	Sensor kelembaban tanah
Bluetooth	Key tracker
Cellular data (data seluler)	Pelacak satwa liar
RFID (Radio Frequency Identification)	Manajemen persediaan

2.1 TUJUAN PEMBELAJARAN

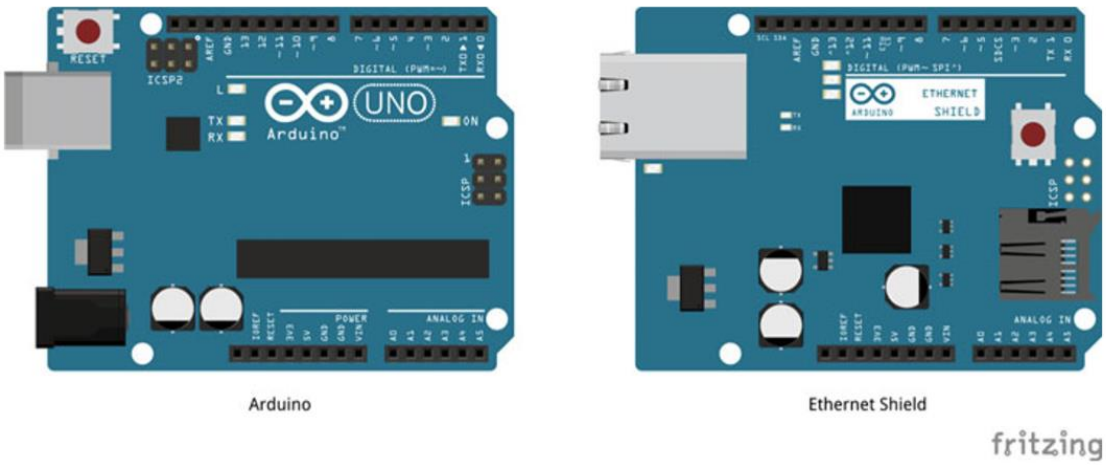
Di akhir bab ini, Anda akan dapat:

- Memasang ethernet shield ke Arduino dan menulis kode konektivitas Ethernet
- Memasang pelindung WiFi ke Arduino dan menulis kode konektivitas WiFi
- Mengatur Arduino Yún untuk terhubung ke WiFi

2.2 KONEKTIVITAS KABEL ARDUINO UNO (ETHERNET)

Pada bagian ini, Anda akan memasang ethernet shield ke Arduino Uno Anda dan menulis kode untuk menghubungkannya ke Internet menggunakan Ethernet.

***Catatan** : Jika Anda menggunakan model Arduino yang dilengkapi dengan kemampuan Ethernet bawaan seperti Arduino Yn, maka Anda tidak memerlukan ethernet shield terpisah. Pengaturan konektivitas Internet Arduino Yún akan dibahas nanti dalam bab ini*



Gambar 2-1. Hardware yang diperlukan untuk konektivitas Internet kabel

Software Yang Perlukan

Untuk menulis kode konektivitas Internet, Anda memerlukan software berikut:

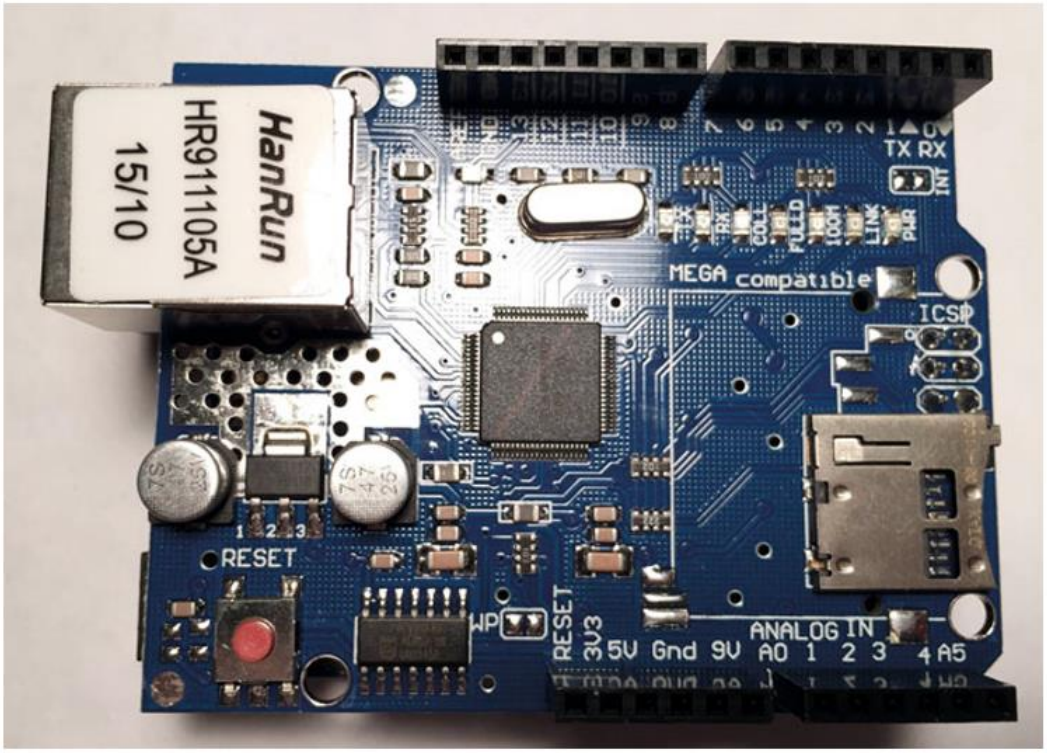
- Arduino IDE 1.6.4 atau versi yang lebih baru

Sirkuit

Di bagian ini, Anda akan membangun sirkuit yang diperlukan untuk konektivitas Internet menggunakan Ethernet.

1. Pastikan Arduino Anda tidak terhubung ke sumber listrik, seperti komputer melalui USB atau baterai.
2. Pasang ethernet shield ke bagian atas Arduino. Semua pin harus sejajar.
3. Hubungkan kabel Ethernet dari Arduino ke port LAN (Local Area Network) router Anda. Router harus sudah terhubung ke Internet.

Setelah Ethernet shield dipasang ke Arduino, itu akan terlihat seperti Gambar 2-2.



Gambar 2-2. Ethernet shield terpasang di bagian atas Arduino Uno

Kode (Arduino)

Sekarang Arduino Anda terhubung secara fisik ke Ethernet, Anda akan menulis kode yang memungkinkan Arduino Anda mengirim dan menerima data melalui Internet. Mulai Arduino IDE dan ketik kode yang disediakan di sini atau unduh dari situs buku dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut dibagi menjadi tiga bagian.

- Library eksternal
- Konektivitas internet (Ethernet)
- Fungsi standar

Library Eksternal

Bagian pertama dari kode seperti yang disediakan dalam Listing 2-1 mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Karena Anda terhubung ke Internet menggunakan Ethernet, ketergantungan utama kode ada di <Ethernet.h>. Arduino

IDE Anda seharusnya sudah menginstal library Ethernet, tetapi ijka tiba-tiba hilang, Anda dapat mengunduhnya dari:

- <Ethernet.h> : https://github.com/arduino/Arduino/tree/master/libraries/Ethernet

Listing 2-1. Kode Including External Dependencies

```
#Including <Ethernet.h>
```

2.3 KONEKTIVITAS INTERNET (ETHERNET)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Seperti yang disediakan dalam Listing 2-2, pertama-tama Anda perlu menentukan alamat MAC dalam variabel mac[]. Untuk ethernet shield yang lebih baru, alamat MAC mungkin ada pada stiker yang tertempel dibawah body macbook Anda. Anda juga perlu mengatur alamat IP statis Arduino jika gagal mendapatkan IP dinamis dari DHCP (*Dynamic Host Configuration Protocol*). Pastikan alamat IP yang Anda gunakan gratis, yaitu, saat ini tidak digunakan oleh beberapa perangkat lain di jaringan. Tentukan variabel EthernetClient yang akan digunakan untuk konektivitas.

Listing 2-2. *Konstanta dan Variabel untuk Menghubungkan ke Internet Menggunakan Ethernet*

```
byte mac[] = {  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED  };
IPAddress staticIP( 10, 0, 0, 20 );
EthernetClient client;
```

Listing 2-3 menyediakan kode untuk pengaturan konektivitas Ethernet. Fungsi connectToInternet() pertama kali mencoba terhubung ke Ethernet dengan DHCP. Jika DHCP gagal menetapkan alamat IP dinamis ke Arduino, ia akan mencoba koneksi ke Ethernet dengan IP statis yang Anda tetapkan.

Listing 2-3. Kode untuk Menghubungkan ke Internet Menggunakan Ethernet

```
void connectToInternet()
{
    // Attempt to connect to Ethernet with DHCP
    if (Ethernet.begin(mac) == 0)
    {
    Serial.print("[ERROR] Failed to Configure Ethernet using DHCP");
    // DHCP failed, attempt to connect to Ethernet withstatic IP
    Ethernet.begin(mac, staticIP);
    }
    // Delay to let Ethernet shield initialize
    delay(1000);
    // Connection successful
    Serial.println("[INFO] Connection Successful");
    Serial.print(""); printConnectionInformation();
    Serial.println("---------------------------------------");
    Serial.println("");
}
```

Setelah Arduino berhasil terhubung ke Internet, fungsi Ethernet printConnectionInformation(), yang tersedia pada Listing 2-4. Fungsi ini mencetak informasi koneksi seperti alamat IP, subnet mask, gateway, dan DNS ke jendela Serial Monitor.

Listing 2-4. Fungsi untuk Menampilkan Informasi Koneksi

```
void printConnectionInformation()
{
```

```
    // Print Connection Information
    Serial.print("[INFO] IP Address: ");
    Serial.println(Ethernet.localIP());
    Serial.print("[INFO] Subnet Mask: ");
    Serial.println(Ethernet.subnetMask());
    Serial.print("[INFO] Gateway: ");
    Serial.println(Ethernet.gatewayIP());
    Serial.print("[INFO] DNS: ");
    Serial.println(Ethernet.dnsServerIP());
  }
```

Fungsi Standar

Terakhir, kode di bagian ketiga dan terakhir ini disediakan di Listing 2-5. Ini mengimplementasikan fungsi standar Arduino setup() dan loop(). Untuk proyek ini, Anda cukup menghubungkan Arduino ke Internet tanpa pemrosesan setelahnya, sehingga fungsi loop() akan tetap kosong.

Listing 2-5. Kode untuk Fungsi Arduino Standar

```
void setup()
{
// Initialize serial port
Serial.begin(9600);

// Connect Arduino to internet
connectToInternet();
}

void loop()
{
// Do nothing
}
```

Sekarang kode Arduino anda sudah lengkap.

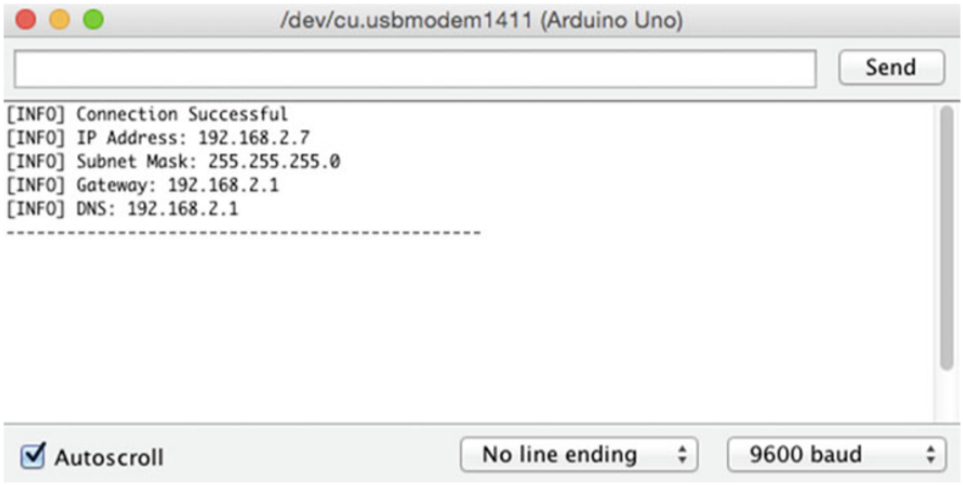
Produk Akhir

Untuk menguji aplikasi, verifikasi dan Upload kode ke Arduino seperti yang dibahas dalam Bab 1. Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log seperti yang ditunjukkan pada Gambar 2-3.

2.4 ARDUINO UNO *WIRELESS CONNECTIVITY* (WiFi)

Di bagian ini, Anda akan memasang pelindung Nirkabel ke Arduino Uno Anda dan menulis *kode untuk menghubungkannya ke Internet menggunakan WiFi.*

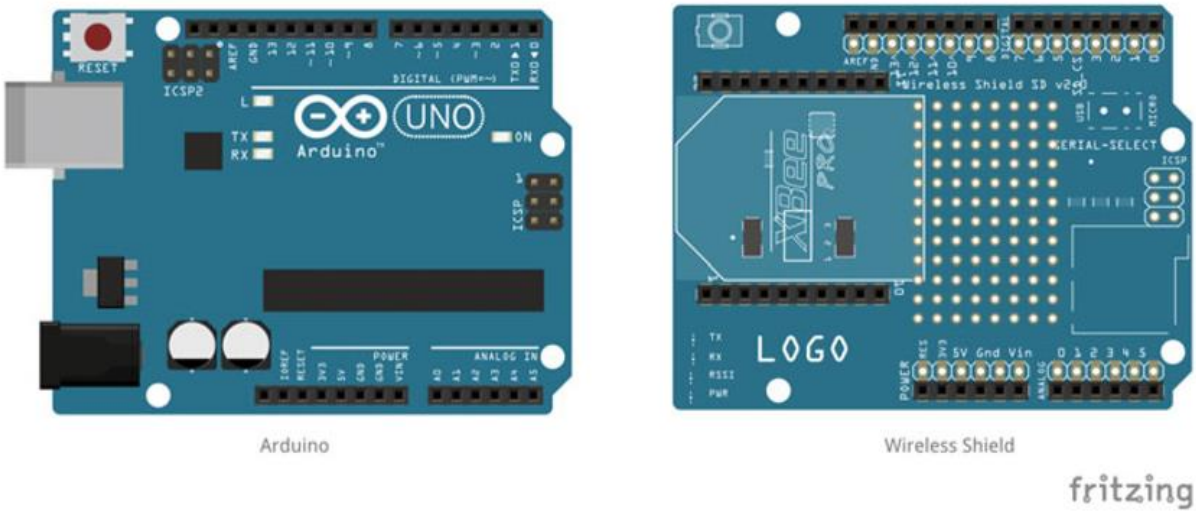
***Catatan** : Jika Anda menggunakan model Arduino yang dilengkapi dengan kemampuan nirkabel bawaan seperti Arduino Yn, maka Anda tidak memerlukan pelindung Nirkabel terpisah. Pengaturan konektivitas Internet Arduino Yún dibahas nanti dalam bab ini.*



Gambar 2-3. Pesan log dari Arduino

Hardware yang Dibutuhkan

Gambar 2-4 memberikan Listing semua komponen hardware yang diperlukan untuk menghubungkan Arduino Uno ke Internet menggunakan pelindung Nirkabel.



Gambar 2-4. Hardware yang diperlukan untuk konektivitas Internet nirkabel

Software yang Diperlukan

Untuk menulis kode konektivitas Internet, Anda memerlukan software berikut:

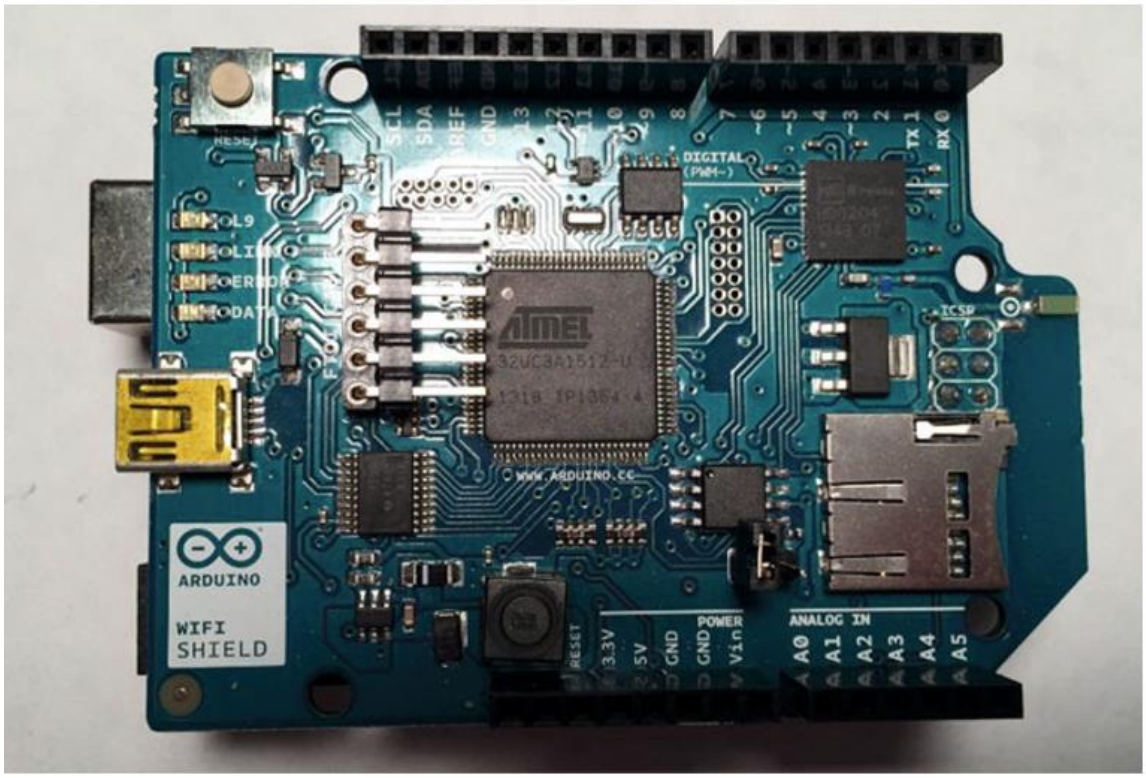
- Arduino IDE 1.6.4 atau versi yang lebih baru

Sirkuit

Di bagian ini Anda akan membangun sirkuit yang diperlukan untuk konektivitas Internet menggunakan WiFi.

1. Pastikan Arduino Anda tidak terhubung ke sumber listrik, seperti komputer melalui USB atau baterai.
2. Pasang pelindung WiFi (alias pelindung nirkabel) ke bagian atas Arduino Anda. Semua pin harus sejajar.

Setelah pelindung nirkabel dipasang ke Arduino, itu akan terlihat seperti Gambar 2-5.



Gambar 2-5. Shield WiFi terpasang di bagian atas Arduino Uno

Kode (Arduino)

Sekarang setelah Arduino Anda dapat terhubung ke jaringan nirkabel, Anda akan menulis kode yang memungkinkan Arduino Anda mengirim dan menerima data melalui Internet. Mulai Arduino IDE Anda dan ketik kode atau unduh dari situs Arduino dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi tiga bagian.

- Library eksternal
- Konektivitas Internet (nirkabel)
- Fungsi standar

Library Eksternal

Bagian pertama dari kode, seperti yang disediakan dalam Listing 2-6, mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Karena Anda terhubung ke Internet secara nirkabel, ketergantungan utama kode ada pada <WiFi.h>. Arduino IDE Anda seharusnya sudah menginstal library WiFi, tetapi karena alasan apa pun hilang, Anda dapat mengunduhnya dari:

- <WiFi.h>: https://github.com/arduino/Arduino/tree/master/libraries/WiFi

Listing 2-6. Library eksternal

```
#include <SPI.h>
#include <WiFi.h>
```

Konektivitas Internet (Wireless)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Untuk menghubungkan Arduino ke perute nirkabel Anda, atur ssid dan **password** (pass) dari jaringan nirkabel Anda, seperti yang disediakan dalam Listing 2-7. Buat juga variabel WiFiClient yang akan digunakan untuk konektivitas Internet.

Listing 2-7. Konstanta dan Variabel untuk Menghubungkan ke Internet Menggunakan WiFi

```
char ssid[] = " YOUR_SSID ";
char pass[] = " YOUR_PASSWORD ";

int keyIndex = 0;
int status = WL_IDLE_STATUS;

WiFiClient client;
```

Listing 2-8 menyediakan kode untuk pengaturan konektivitas nirkabel. Fungsi connectToInternet() pertama-tama memeriksa apakah pelindung WiFi terpasang. Selanjutnya, kode terus mencoba untuk terhubung ke jaringan nirkabel. Loop dan fungsi berakhir setelah Arduino berhasil terhubung ke jaringan nirkabel.

Listing 2-8. Kode untuk Menghubungkan ke Internet Menggunakan WiFi

```
void connectToInternet()
{
   status = WiFi.status();
   // Check for the presence of the shield
   if (status == WL_NO_SHIELD)
{
   Serial.println("[ERROR] WiFi Shield Not Present");
   // Do nothing
   while (true);
}
// Attempt to connect to WPA/WPA2 Wifi network
while ( status!= WL_CONNECTED)
{
   Serial.print("[INFO] Attempting Connection - WPA SSID: ");
   Serial.println(ssid);

   status = WiFi.begin(ssid, pass);
}
// Connection successful
Serial.print("[INFO] Connection Successful");
Serial.print("");     printConnectionInformation();
Serial.println("-----------------------------------------");
Serial.println("");
}
```

Setelah Arduino berhasil terhubung ke jaringan nirkabel, fungsi printConnectionInformation() yang disediakan di Listing 2-9 dipanggil. Ini mencetak SSID, alamat MAC router, Kekuatan Sinyal (RSSI), alamat IP Arduino, dan alamat MAC Arduino, semuanya di jendela Serial Monitor.

Listing 2-9. Fungsi untuk Menampilkan Informasi Koneksi

```
void printConnectionInformation()
{
// Print Network SSID
Serial.print("[INFO] SSID: ");
Serial.println(WiFi.SSID());
// Print Router's MAC address
```

```
byte bssid[6];    WiFi.BSSID(bssid);
Serial.print("[INFO] BSSID: ");
Serial.print(bssid[5], HEX);
Serial.print(":");
Serial.print(bssid[4], HEX);
Serial.print(":");
Serial.print(bssid[3], HEX);
Serial.print(":");
Serial.print(bssid[2], HEX);
Serial.print(":");
Serial.print(bssid[1], HEX);
Serial.print(":");
Serial.println(bssid[0], HEX);
// Print received signal strength
long rssi = WiFi.RSSI();
Serial.print("[INFO] Signal Strength (RSSI): ");
Serial.println(rssi);
// Print encryption type
byte encryption = WiFi.encryptionType();
Serial.print("[INFO] Encryption Type: ");
Serial.println(encryption, HEX);
// Print WiFi Shield's IP address    IPAddress ip = WiFi.localIP();    Serial.print("[INFO] IP
Address: ");    Serial.println(ip);
// Print MAC address
byte mac[6];
WiFi.macAddress(mac);
Serial.print("[INFO] MAC Address: ");
Serial.print(mac[5], HEX);
Serial.print(":");
Serial.print(mac[4], HEX);
Serial.print(":");
Serial.print(mac[3], HEX);
Serial.print(":");
Serial.print(mac[2], HEX);
Serial.print(":");
Serial.print(mac[1], HEX);
Serial.print(":");
Serial.println(mac[0], HEX);
```

Fungsi Standar

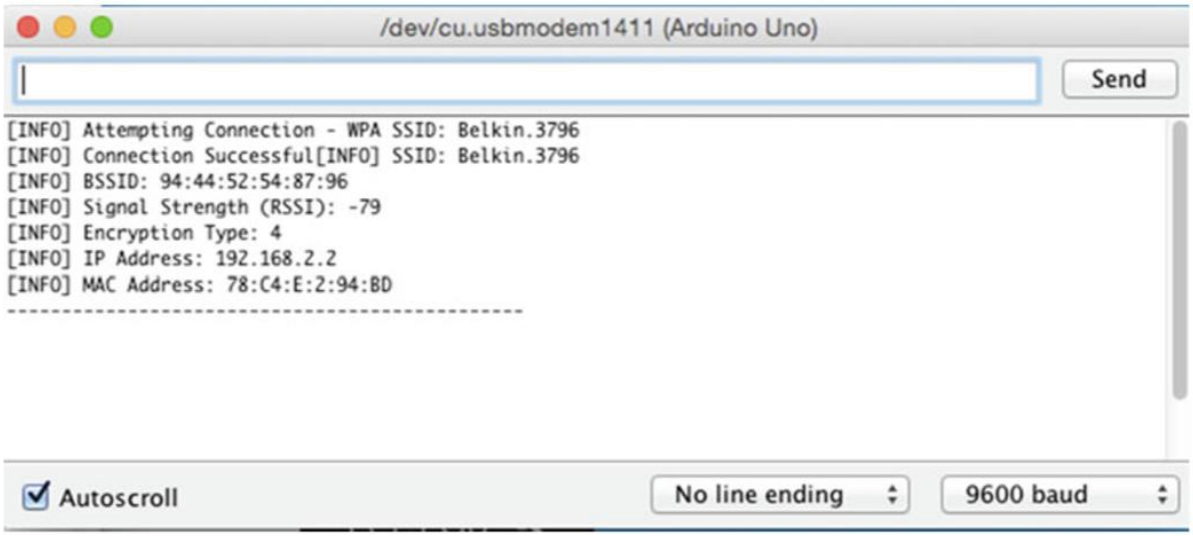
Akhirnya, kode di bagian ketiga dan terakhir, seperti yang disediakan dalam Listing 2-10, mengimplementasikan fungsi setup() dan loop() standar Arduino. Untuk proyek ini, yang Anda lakukan hanyalah menghubungkan Arduino ke Internet dan tidak ada pemrosesan setelahnya, sehingga fungsi loop() akan tetap kosong.

Listing 2-10. Kode untuk Fungsi Arduino Standar

```
void setup()
{
// Initialize serial port
Serial.begin(9600);
```

```
// Connect Arduino to Internet
connectToInternet();
}
void loop()
{
// Do nothing
}
```

Arduino Anda sekarang sudah siap.



Gambar 2-6. Pesan log dari Arduino

Produk Akhir

Untuk menguji aplikasi, verifikasi dan Upload kode ke Arduino seperti yang dibahas dalam Bab 1. Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log seperti yang ditunjukkan pada Gambar 2-6.

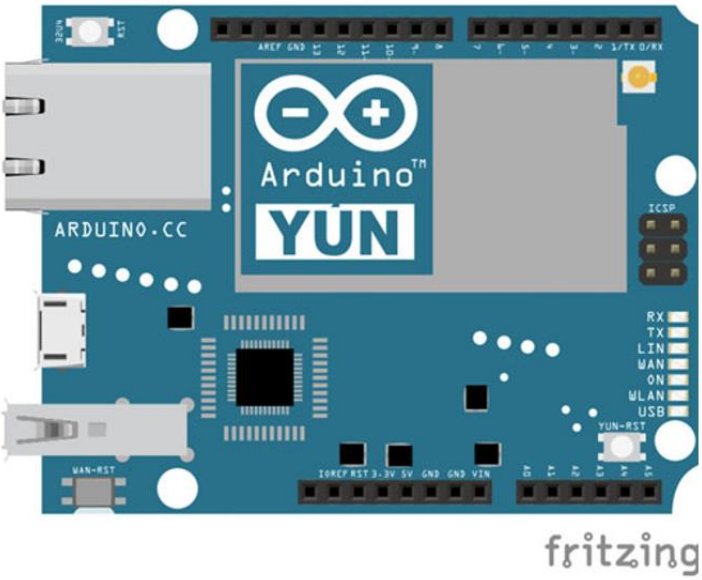
2.5 ARDUINO YÚN WIRELESS CONNECTIVITY (WiFi)

Yún adalah model Arduino yang lebih canggih yang telah dikembangkan untuk Internet of things. Untuk pemula, Arduino Yún mungkin sedikit rumit dibandingkan dengan Arduino Uno, tetapi ia dilengkapi dengan Ethernet dan kemampuan nirkabel built-in sehingga Anda tidak perlu membeli pelindung tambahan.

Seperti disebutkan dalam Bab 1, buku ini menggunakan Arduino Uno secara keseluruhan. Bagian ini hanya disediakan sebagai referensi bagi pembaca yang telah memiliki Arduino Yn dan masih ingin mengikuti prototipe kehidupan nyata yang dikembangkan dalam buku ini. Meskipun Arduino Yn tidak dirujuk di sisa buku ini, unduhan kode juga berisi kode yang kompatibel dengan Arduino Yn.

Hardware yang diperlukan

Anda tidak memerlukan hardware tambahan untuk menghubungkan Arduino Yn ke Internet, jadi Gambar 2-7 hanya menyertakan diagram Arduino Yn.



Gambar 2-7. Arduino Yun

Software yang Diperlukan

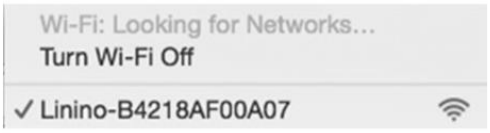
Untuk menulis kode konektivitas Internet, Anda memerlukan software berikut:

- Arduino IDE 1.6.4 atau versi yang lebih baru

Setup Wireless

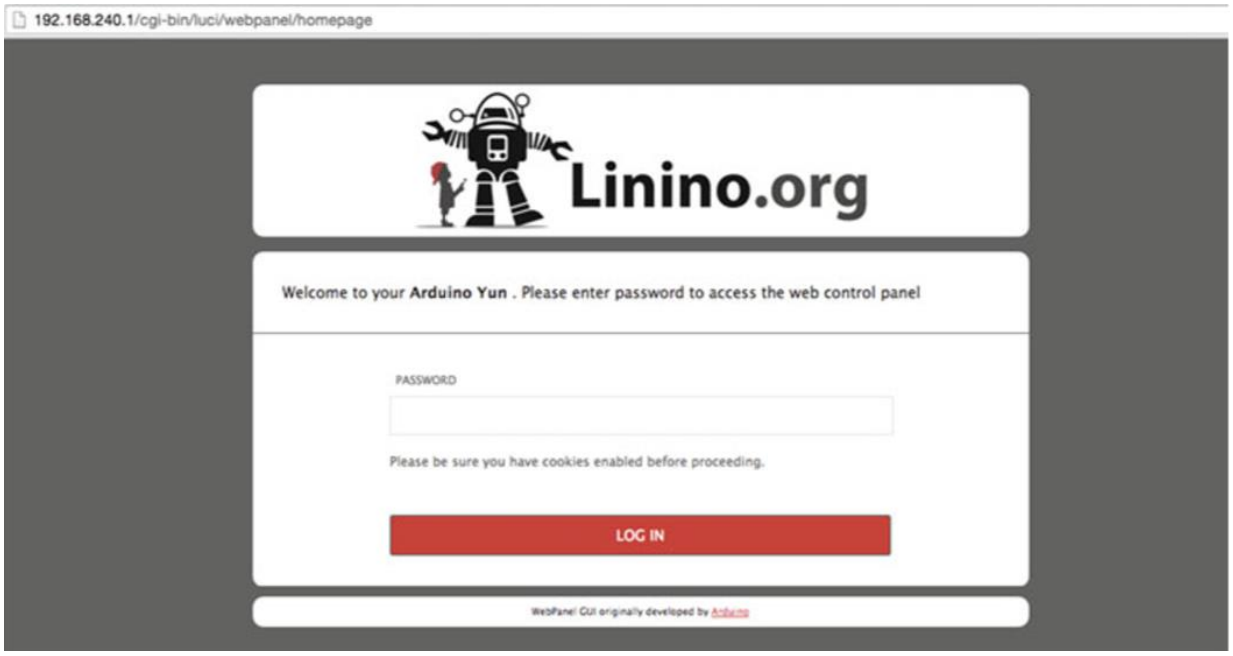
Tidak seperti Arduino Uno, ketika Anda perlu memasang pelindung nirkabel atau Ethernet, Arduino Yún hadir dengan kemampuan konektivitas Ethernet dan nirkabel built-in. Arduino Yn bertindak sebagai hotspot dengan menghubungkan langsung ke jaringan kabel atau nirkabel Anda. Jadi, Anda tidak perlu menulis kode konektivitas Internet; sebagai gantinya, Anda hanya perlu mengatur Arduino Yn Anda untuk terhubung ke jaringan Anda. Bagian ini membahas pengaturan nirkabel untuk Arduino Yn.

1. Hubungkan Arduino Yn ke komputer Anda dengan kabel Micro USB.
2. Arduino Yn juga berfungsi sebagai hotspot, jadi dari WiFi komputer Anda, cari Arduino Yn. Tergantung di mana Anda membeli Arduino Yn Anda, itu mungkin muncul sebagai ArduinoYunXXXXXXXXXXX atau LininoXXXXXXXXXXX di koneksi WiFi yang tersedia di komputer Anda. Seperti yang ditunjukkan pada Gambar 2-8, sambungkan ke Arduino Yún secara nirkabel.



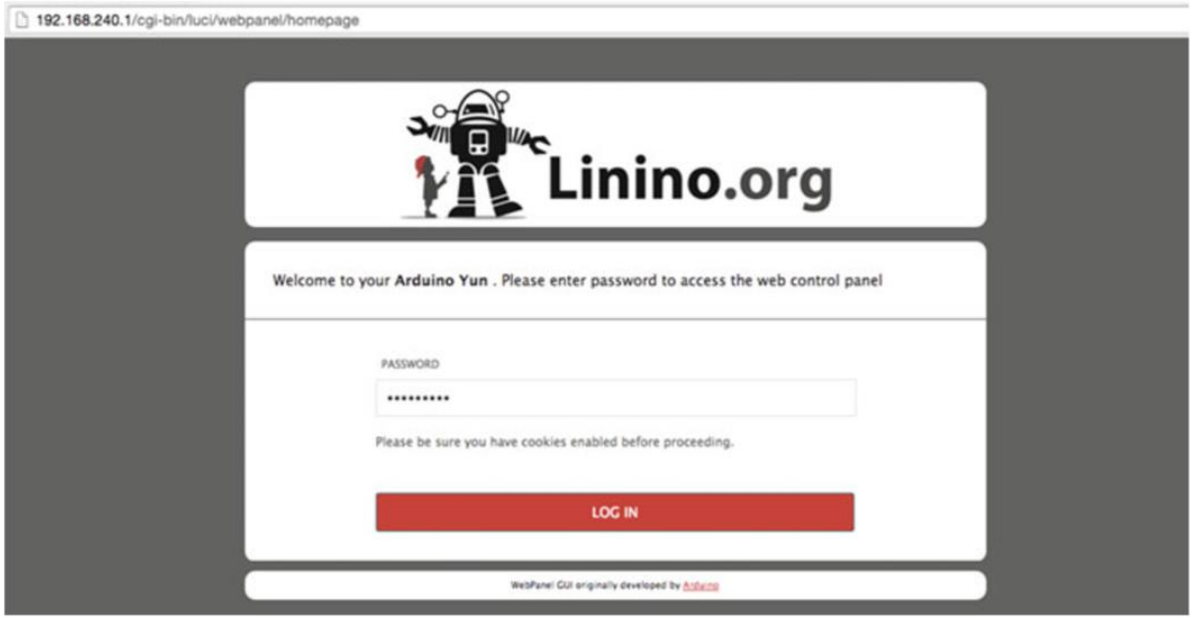
Gambar 2-8. Pilih Arduino Yún dari jaringan nirkabel

3. Setelah terhubung, buka browser web di komputer Anda dan masukkan http://arduino.local (jika ini tidak berhasil maka masukkan IP default http://192.168.240.1). Seperti yang ditunjukkan pada Gambar 2-9, layar login untuk Arduino Yún Anda akan terbuka.



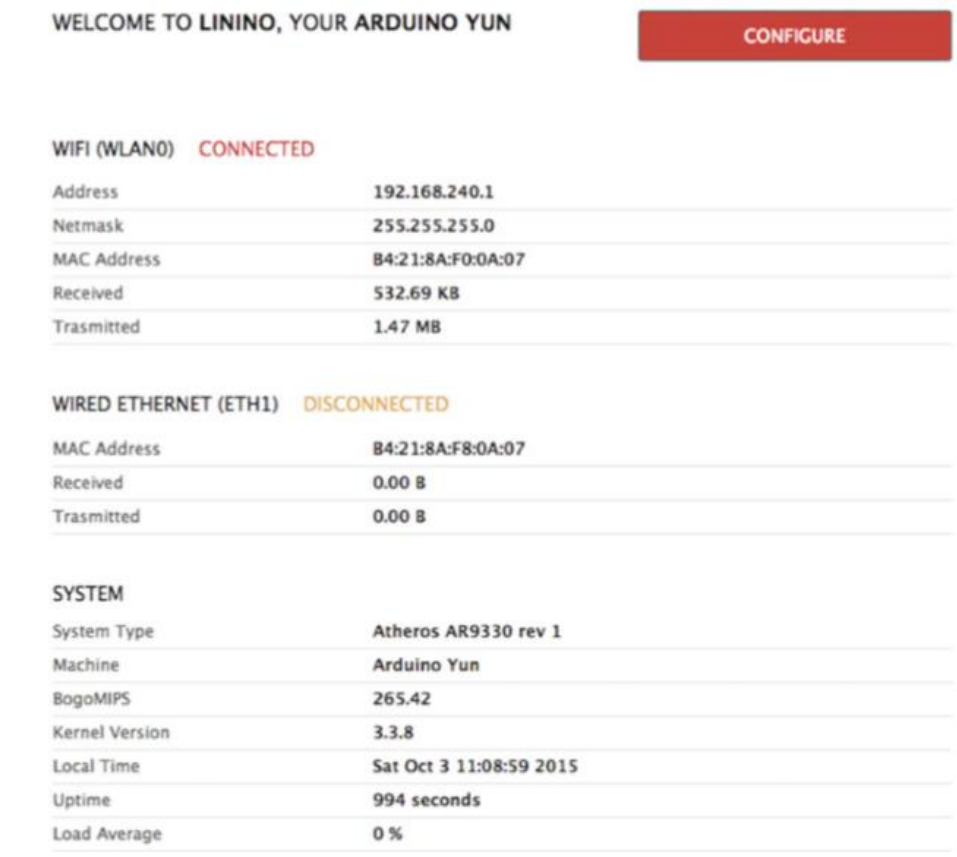
Gambar 2-9. Layar masuk Arduino Yn

4. Jika ini adalah pertama kalinya Anda mengakses Arduino Yún Anda, lalu masukkan p assword arduino default (jika ini tidak berhasil, coba doghunter ; jika tidak, periksa dokumentasi pabrikan). Klik tombol Masuk seperti yang ditunjukkan pada Gambar 2-10.

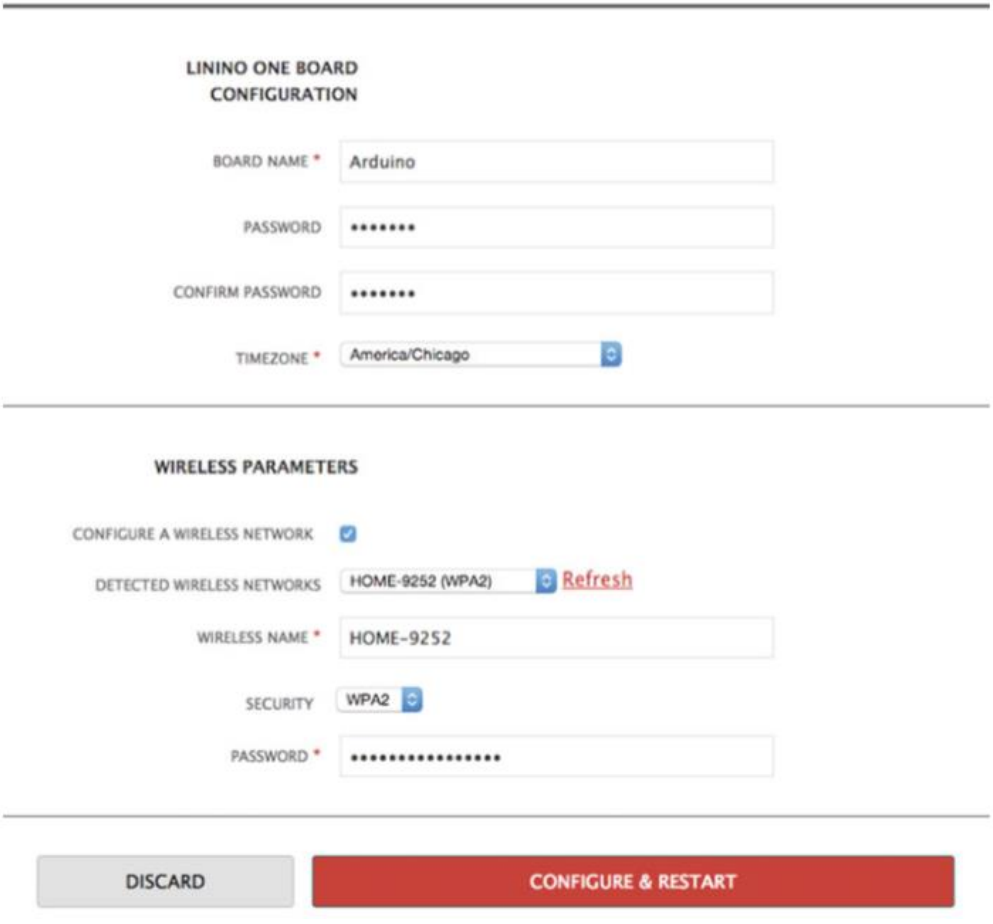


Gambar 2-10. Masukkan **password** dan masuk

5. Setelah login berhasil, Anda akan diarahkan ke halaman konfigurasi Arduino Yn Anda, seperti yang ditunjukkan pada Gambar 2-11. Klik tombol **Configurb**.



Gambar 2-11. Konfigurasi rduino Yún default c

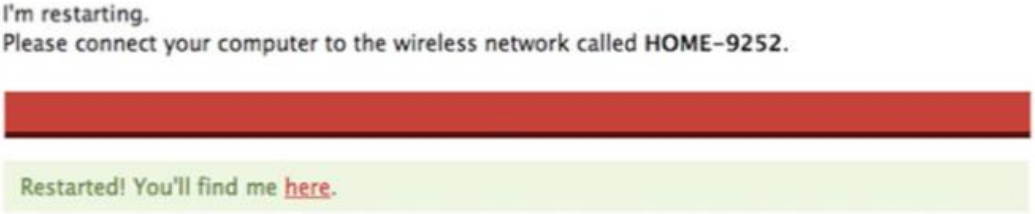


Gambar 2-12. Konfigurasi c nirkabel rduino Yún

6. Seperti yang ditunjukkan pada Gambar 2-12, Anda dapat mengubah Nama Papan, **Password**, dan Zona Waktu Arduino Yn Anda. Di bawah bagian Parameter Nirkabel, pilih jaringan nirkabel yang biasa Anda gunakan dari Listing Jaringan Nirkabel Terdeteksi. Pilih jenis keamanan dan masukkan **password** jaringan. Setelah selesai, klik tombol Configure & Restart.
7. Arduino Yún akan dimulai ulang dengan setelan yang diUpdate, seperti yang ditunjukkan pada Gambar 2-13.

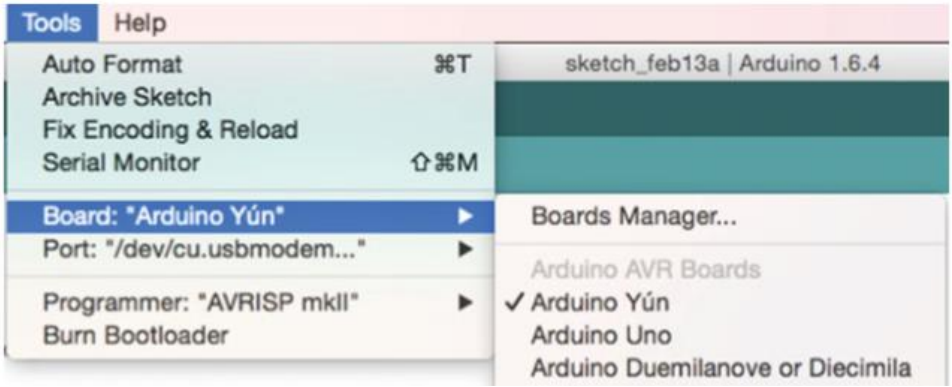
Gambar 2-13. Sebuah rduino Yún r mulai

8. Seperti yang ditunjukkan pada Gambar 2-14, selama restart Arduino Yún akan menampilkan pesan agar Anda terhubung ke jaringan nirkabel yang umum digunakan. Setelah restart, Anda akan dapat mengakses Arduino Yn Anda menggunakan IP yang ditetapkan oleh router nirkabel Anda. Jika Anda tidak dapat menemukan IP yang ditetapkan, ikuti langkah selanjutnya dan Upload kode yang disediakan di bagian selanjutnya yang mencetak informasi koneksi.



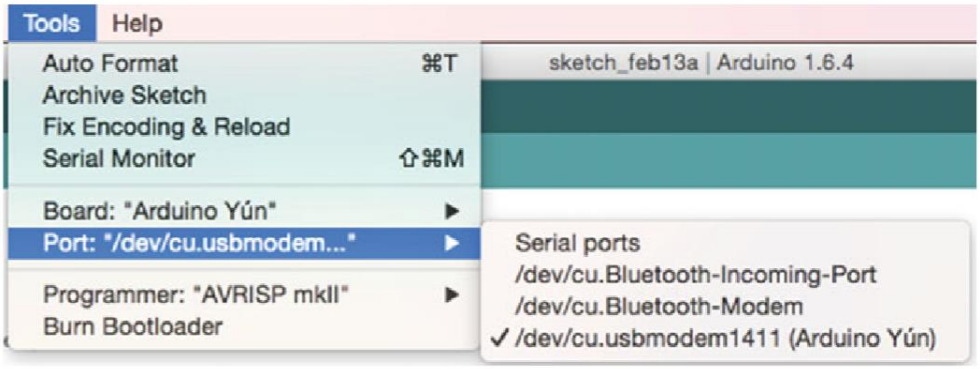
Gambar 2-14. Sebuah rduino Yún restart selesai

9. Buka Arduino IDE saat Arduino Yn masih terhubung melalui Micro USB ke komputer Anda. Seperti yang ditunjukkan pada Gambar 2-15 dari **Tools** lalu pilih **Board**, pilihlah **Arduino Yún**.



Gambar 2-15. Pilih Arduino board Yn

10. Seperti yang ditunjukkan pada Gambar 2-16, dari **tool** lalu pilih **Port**, pilih port yang bertuliskan Arduino Yn.



Gambar 2-16. Pilih port Arduino Yún

Kode (Arduino)

Sekarang Arduino Yn Anda terhubung ke jaringan nirkabel, Anda akan menulis kode yang memungkinkan Arduino Anda mengirim dan menerima data melalui Internet. Karena Arduino Yún sudah terhubung ke Internet, di sinilah kodenya akan sedikit berbeda. Alih-alih menambahkan kode untuk terhubung, Anda cukup menggunakan library <Bridge.h> untuk menggunakan koneksi nirkabel.

Mulai Arduino IDE Anda dan ketik kode berikut atau unduh dari situs kami dan buka. Semua kode masuk ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi tiga bagian.

- Library eksternal
- Konektivitas Internet (Nirkabel)
- Baca data sensor

Library Eksternal

Bagian pertama dari kode seperti yang disediakan dalam Listing 2-11 mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Untuk Arduino Yún, <Bridge.h> memungkinkan Anda mengakses koneksi Internet yang sudah dibuat. Anda juga akan menggunakan <Process.h> untuk mencetak informasi koneksi. Arduino IDE Anda telah menginstal kedua library ini.

Listing 2-11. Library eksternal

```
#include <Bridge.h>
#include <Process.h>
```

Konektivitas Internet (Wireless)

Bagian kedua dari kode, yang disediakan di Listing 2-12, mendefinisikan fungsi yang akan digunakan untuk menampilkan informasi koneksi. Karena Arduino sudah terhubung ke jaringan nirkabel, fungsi printConnectionInformation() dipanggil. Ini mencetak informasi koneksi nirkabel.

Listing 2-12. Fungsi untuk Menampilkan Informasi Koneksi

```
void printConnectionInformation()
{
   // Initialize a new process
   Process wifiCheck;
   // Run Command

   wifiCheck.runShellCommand("/usr/bin/pretty-wifi-info.lua");
   // Print Connection Information
   while wifiCheck.available() > 0)
```

```
    {
    char c = wifiCheck.read();
    Serial.print(c);
    }
    Serial.println("----------------------------------------");
    Serial.println("");
}
```

Fungsi Standar

Terakhir, kode di bagian ketiga dan terakhir, yang disediakan di Listing 2-13, mengimplementasikan fungsi setup() dan loop() standar Arduino. Untuk proyek ini, yang Anda lakukan hanyalah mencetak informasi koneksi Internet dan tidak ada pemrosesan setelahnya, sehingga fungsi loop() akan tetap kosong.

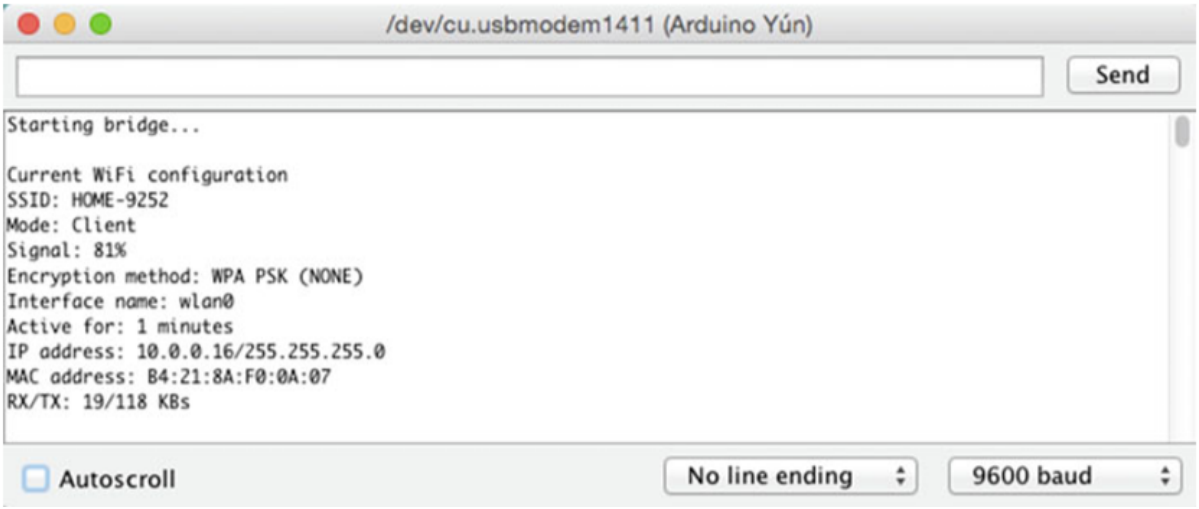
Satu perbedaan utama dalam kode ini versus kode Arduino Uno adalah Anda perlu menginisialisasi jembatan menggunakan Bridge.begin(). Ini pada dasarnya memungkinkan Anda mengakses koneksi Internet Arduino Yn.

Listing 2-13. Kode untuk Fungsi Arduino Standar

```
void setup()
{
    // Initialize serial port    Serial.begin(9600);
        // Do nothing until serial monitor is opened
    while (!Serial);
    // Contact the Linux processor
    Bridge.begin();
    // Connect Arduino to Internet
    printConnectionInformation();
    }
    void loop()
    {
    // Do nothing
}
```

Produk Akhir

Untuk menguji aplikasi, verifikasi dan Upload kode ke Arduino, seperti yang dibahas pada Bab 1. Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log seperti yang ditunjukkan pada Gambar 2-17.



Gambar 2-17. Pesan log dari Arduino

2.6 RINGKASAN

Dalam bab ini Anda mengembangkan kode untuk menghubungkan Arduino Uno ke Internet menggunakan ethernet shield dan pelindung WiFi. Anda juga melihat pengaturan nirkabel untuk Arduino Yn dan kode yang diperlukan untuk mengakses koneksi Internet. Untuk proyek masa depan Anda yang memerlukan konektivitas Internet menggunakan Ethernet atau WiFi, Anda dapat menggunakan kode yang disediakan dalam bab ini sebagai dasar dan kemudian menambahkan kode Anda sendiri ke dalamnya.

BAB 3
PROTOKOL KOMUNIKASI

Pada Bab 2, Anda sudah mempelajari cara menghubungkan Arduino ke Internet masing-masing menggunakan Ethernet dan WiFi. Bab ini membahas dua protokol yang digunakan untuk mengirim dan menerima data. Protokol adalah format terstruktur yang disepakati yang digunakan untuk komunikasi jaringan. Ini mendefinisikan apa yang harus dikirim dan diterima dan tindakan apa yang harus diambil.

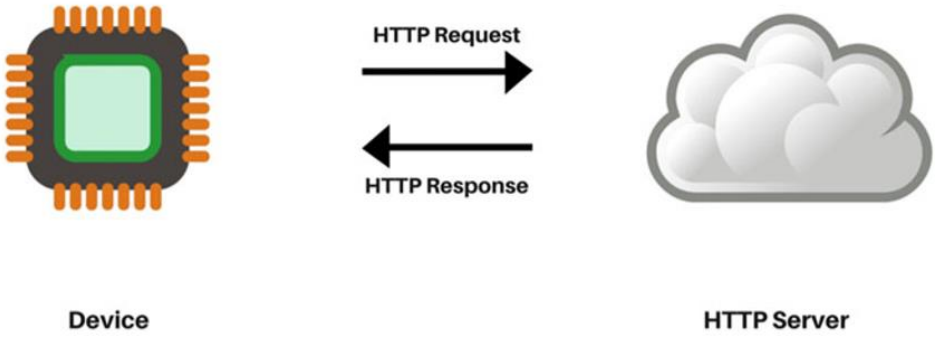
3.1 TUJUAN PEMBELAJARAN

Di akhir bab ini, Anda akan dapat:

- Memahami dasar-dasar protokol HTTP
- Mengirim permintaan HTTP ke server
- Memahami dasar-dasar protokol MQTT
- Memublikasikan dan subscribe ke broker MQTT

3.2 HTTP

Web menggunakan *Hyper Text Transfer Protocol* (HTTP) sebagai protokol dasarnya. HTTP mendukung beberapa metode transmisi data, tetapi dalam proyek ini Anda akan menulis kode untuk dua metode yang lebih populer, GET dan POST. Metode GET dan POST melakukan pekerjaan yang sama dan kodenya sangat mirip, tetapi dengan sedikit perbedaan dalam format permintaan. Dibandingkan dengan POST, GET tidak memiliki batasan seperti itu, yang membatasi jumlah data yang dapat ditransfer. POST dianggap lebih aman daripada GET. Tergantung pada kebutuhan Anda, Anda dapat memutuskan mana yang tepat untuk Anda. Gambar 3-1. menunjukkan interaksi tingkat tinggi antara perangkat dan server HTTP.



Gambar 3-1. Protokol Transfer Teks Hiper (HTTP)

***Catatan** : Untuk persyaratan hardware dan software serta instruksi sirkuit, lihat bagian "Konektivitas Nirkabel Arduino Uno (WiFi)" di Bab 2.*

Kode (Arduino)

Selanjutnya Anda akan menulis kode untuk menghubungkan Arduino ke Internet menggunakan WiFi dan mengirim data uji ke server menggunakan HTTP. Mulai Arduino IDE dan ketik kode berikut atau unduh dari situs buku dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino) dalam urutan yang sama seperti yang disediakan di sini, tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi empat bagian.

- Library Eksternal
- Konektivitas Internet (Wireless)

- Data publish (HTTP)
- Fungsi Standar

Library Eksternal

Bagian pertama dari kode mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Kode di bagian ini sama dengan Listing 2-6.

Konektivitas Internet (Nirkabel)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (Bab 2) di sini.

3.3 PUBLISH DATA

Bagian ketiga dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk mengirim data ke server menggunakan HTTP. Seperti yang disediakan dalam Listing 3-1, Anda terlebih dahulu menentukan alamat dan port server yang akan dihubungkan oleh Arduino dan mengirim data. Untuk tujuan proyek ini, Anda dapat mempublishkannya ke www.httpbin.org, yang merupakan server uji yang tersedia secara terbuka yang hanya menampilkan semua informasi permintaan bersama dengan beberapa informasi tambahan. Dalam proyek mendatang, Anda akan menggunakan server yang memproses data permintaan.

Listing 3-1. Variabel untuk Menentukan Server HTTP.

```
char server[] = {"www.httpbin.org"};
int port = 80;
```

Fungsi doHttpGet() yang disediakan di Listing 3-2 merangkum semua detail persiapan permintaan untuk metode GET, menghubungkan ke server dan mengirim permintaan. Mencoba terhubung ke server menggunakan client.connect(server, port) dalam kondisi IF. Jika koneksi berhasil, maka siapkan permintaan. Dalam permintaan yang menggunakan metode GET, data dikirim sebagai bagian dari URL dalam format pasangan nama/nilai, misalnya, http://www.httpbin.org/get?temperatureSensor=85&metric=F. Contoh menunjukkan bahwa dua parameter akan dikirim, yang pertama adalah sensor suhu dengan nilai 85 dan yang kedua adalah metrik dengan nilai F. Terakhir, kirimkan permintaan HTTP ke server menggunakan metode client.println(). Metode ini akan mengirimkan perintah ke server melalui jaringan dan kemudian menerima respons apa pun dari server.

Listing 3-2. HTTP GET Request

```
void doHttpGet()
{
// Prepare data or parameters that need to be posted to server
String requestData = "requestVar=test";
    // Check if a connection to server:port was made
    if (client.connect(server, port))
    {
    Serial.println("[INFO] Server Connected - HTTP GET Started");
    // Make HTTP GET request
    client.println("GET /get?" + requestData + " HTTP/1.1");
    client.println("Host: " + String(server));
    client.println("Connection: close");
    client.println();
    Serial.println("[INFO] HTTP GET Completed");
```

```
    }
    else
    {
    Serial.println("[ERROR] Connection Failed");
    }
    Serial.println("-----------------------------------------");
}
```

Kode ini adalah untuk mengirim permintaan HTTP GET, tetapi seperti yang disebutkan sebelumnya, ia memiliki batasan panjang, jadi jika Anda ingin menghindari batasan ini, gunakan HTTP POST sebagai gantinya. Fungsi doHttpPost() yang disediakan di Listing 3-3 merangkum semua detail persiapan permintaan untuk metode POST, menghubungkan ke server, dan mengirim permintaan. Mencoba terhubung ke server menggunakan client.connect(server, port) dalam kondisi IF. Sejauh ini, kodenya mirip dengan permintaan HTTP GET. Jika koneksi berhasil, maka siapkan permintaan.

Dalam permintaan yang menggunakan metode POST, data juga dikirim dalam format pasangan nama/nilai, tetapi merupakan bagian dari permintaan. Seperti yang Anda lihat di Listing 3-3, mengirim permintaan HTTP POST memerlukan informasi header tambahan. Terakhir, kirimkan permintaan HTTP ke server menggunakan metode client.println(). Metode ini akan mengirimkan perintah ke server melalui jaringan dan kemudian menerima respons apa pun dari server.

Listing 3-3. Permintaan HTTP POST

```
void doHttpPost()
{
// Prepare data or parameters that need to be posted to server    String requestData =
"requestData={\"requestVar:test\"}";
// Check if a connection to server:port was made
if (client.connect(server, port))
{
Serial.println("[INFO] Server Connected - HTTP POST Started");
// Make HTTP POST request
client.println("POST /post HTTP/1.1");
client.println("Host: " + String(server));
client.println("User-Agent: Arduino/1.0");
client.println("Connection: close");
client.println("Content-Type:
application/x-www-form-urlencoded;");
client.print("Content-Length: ");
client.println(requestData.length());
client.println();
client.println(requestData);
Serial.println("[INFO] HTTP POST Completed");
}
else
{
// Connection to server:port failed
Serial.println("[ERROR] Connection Failed");    }
Serial.println("-----------------------------------------");
}
```

Itu cukup untuk mempublishkan data dari Arduino Anda ke server.
Fungsi Standar Kode di bagian keempat dan terakhir mengimplementasikan fungsi setup() dan loop() standar Arduino. Seperti yang ditunjukkan Listing 3-4, fungsi setup() menginisialisasi port serial, menghubungkan ke Internet, dan kemudian membuat permintaan HTTP GET dengan memanggil doHttpGet() atau permintaan HTTP POST dengan memanggil fungsi doHttpPost().

Listing 3-4. Kode untuk Fungsi Arduino Standar—setup()

```
void setup()
{
    // Initialize serial port
    Serial.begin(9600);
    // Connect Arduino to internet
    connectToInternet();
    // Make HTTP GET request
    doHttpGet();
}
```

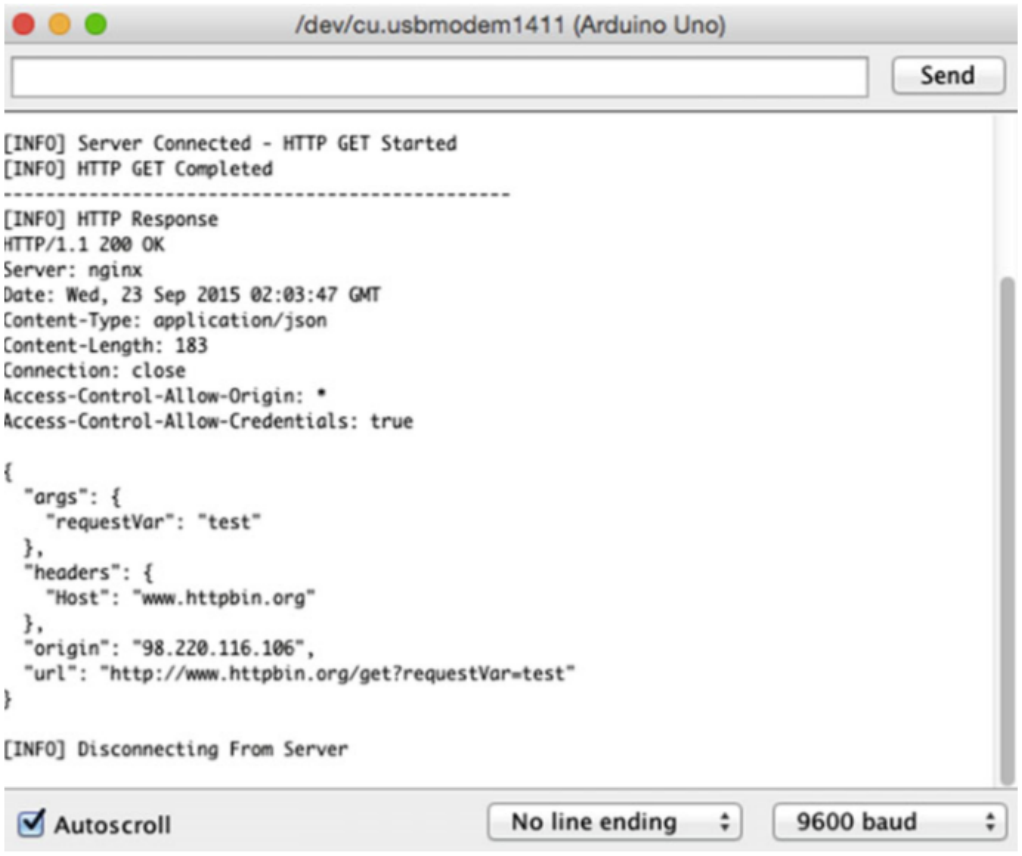
Karena dalam proyek ini Anda tidak melakukan pemrosesan sisi server dengan data yang dikirim dari sensor, Anda akan menambahkan kode untuk membaca respons dari server ke fungsi loop(). Server uji yang Anda gunakan hanya menggemakan semua informasi permintaan dalam respons, jadi Anda hanya akan membaca respons dan mencetaknya ke jendela Serial Monitor.

Seperti yang disediakan di Listing 3-5, periksa apakah ada byte yang tersedia untuk dibaca dari WiFiClient, baca semua byte yang tersedia, dan cetak ke jendela Serial Monitor. Setelah semua byte telah dibaca dan dicetak, hentikan klien.

Listing 3-5. Kode untuk Fungsi Arduino Standar—loop()

```
void loop()
{
if (client.available())
{
Serial.println("[INFO] HTTP Response");
}
// Read available incoming bytes from the server and print    while (client.available())
    {
    char c = client.read();
    Serial.write(c);
    }
    // If the server:port has disconnected, then stop the client
    if (!client.connected())
    {
    Serial.println();
    Serial.println("[INFO]
    Disconnecting From Server");
    client.stop();
    }
}
```

Kode Arduino Anda selesai.



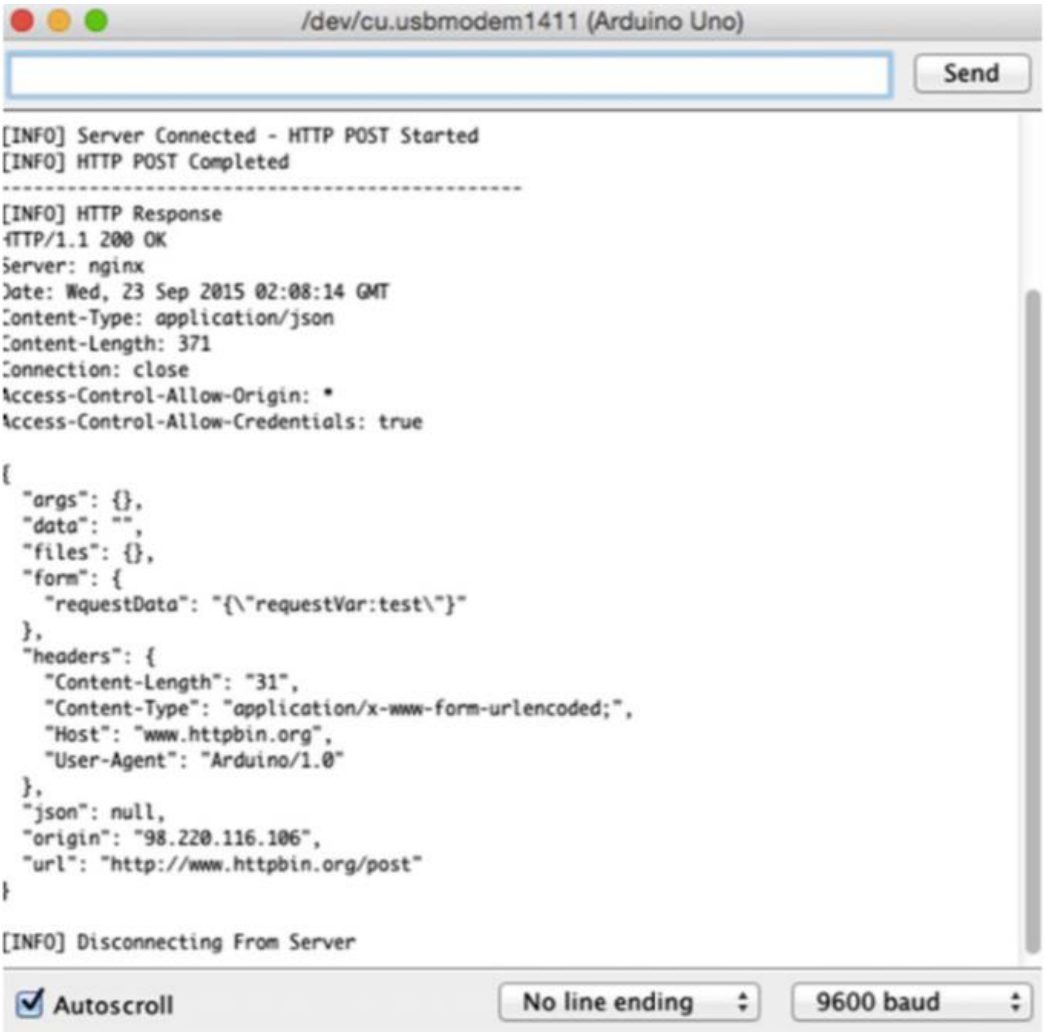
Gambar 3-2. Permintaan H TTP: Metode G ET

Produk Akhir

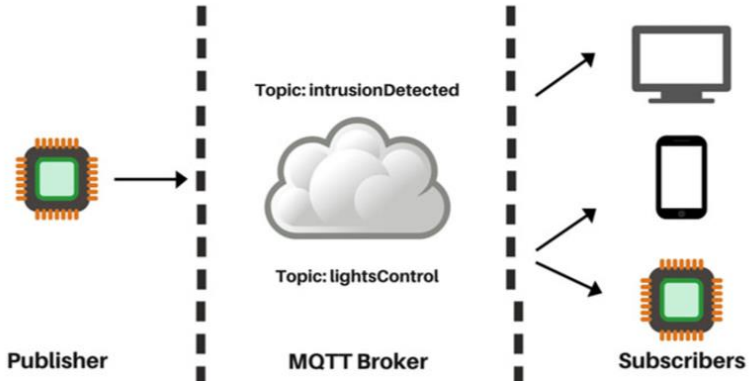
Untuk menguji aplikasi, verifikasi dan Upload kode seperti yang dibahas dalam Bab 1. Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log yang mirip dengan yang ditunjukkan pada Gambar 3-2 untuk HTTP GET dan pada Gambar 3-3 untuk HTTP POST.

3.4 MQTT

MQTT adalah protokol mesin-ke-mesin yang ringan. Ini mengikuti model penerbit-pelanggan, di mana penerbit menerbitkan data ke server (alias, broker) dan pelanggan menerima data. Penerbit dan pelanggan tidak saling mengenal; mereka terhubung ke broker, yang membuat komunikasi ini tidak sinkron.



Gambar 3-3. Permintaan HTTP: metode POST



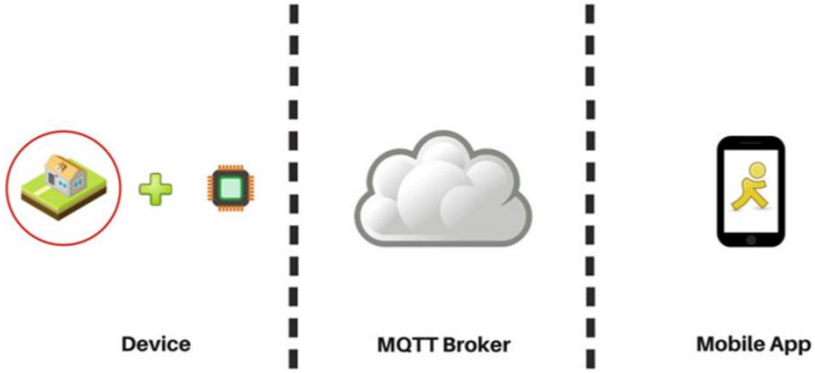
Gambar 3-4. Protokol MQTT

Broker memberi tahu semua pelanggan bahwa data yang relevan telah diterbitkan menggunakan konsep topik. Topik mirip dengan umpan berita, di mana Anda subscribe topik tertentu yang ingin Anda terima beritanya. Penerbit dan pelanggan dapat berupa sensor, mesin, dan aplikasi seluler. Gambar 3-4 memberikan ikhtisar tingkat tinggi tentang protokol MQTT.

Memahami MQTT penting untuk membangun aplikasi IoT, jadi mari kita lihat beberapa skenario yang akan membantu Anda memahami MQTT.

3.5 SISTEM PENDETEKSI INTRUSI

Versi sederhana dari sistem deteksi intrusi ditunjukkan pada Gambar 3-5. Ini akan terdiri dari tiga komponen—sensor gerak yang mendeteksi intrusi dan memublikasikan data, aplikasi seluler yang menerima data ini dan memperingatkan pengguna aplikasi, dan komponen, yang merupakan topik di broker MQTT.



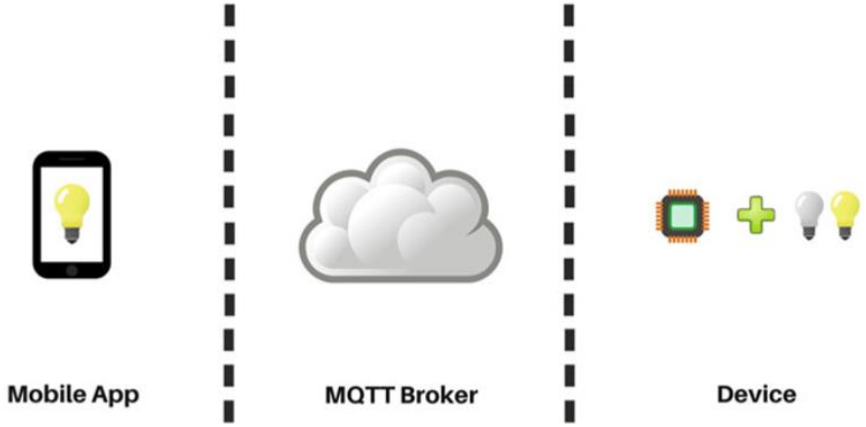
Gambar 3-5. Komponen sistem deteksi intrusi

Sensor akan bertindak sebagai penerbit dan mempublishkan pesan baru ke topik *codifythings/ intrusionDetected* pada broker MQTT segera setelah intrusi terdeteksi. Broker MQTT akan menambahkan pesan ini ke topik. Aplikasi seluler akan menjadi pelanggan topik codifythings/intrusionDetected. Setiap kali pesan baru diterbitkan ke topik, itu akan mendapat pemberitahuan. Ini akan mengakibatkan aplikasi seluler membuat pemberitahuan untuk pengguna aplikasi. Anda akan membangun sistem ini di Bab 6.

3.6 KONTROL PENCAHAYAAN JARAK JAUH

Penggunaan hebat lainnya dari MQTT adalah mengembangkan aplikasi seluler yang berfungsi sebagai kontrol jarak jauh untuk berbagai jenis perangkat, seperti aplikasi kontrol pencahayaan. Seperti yang ditunjukkan pada Gambar 3-6, aplikasi remote control juga akan terdiri dari tiga komponen, tetapi dibandingkan dengan contoh sebelumnya, urutan dua komponen pertama terbalik. Itu berarti komponen pertama adalah aplikasi seluler yang memungkinkan pengguna menyalakan atau mematikan lampu, komponen kedua adalah perangkat yang terhubung ke lampu, dan komponen ketiga adalah topik di broker MQTT.

Pengguna aplikasi seluler berinteraksi dengan aplikasi untuk menyalakan atau mematikan lampu, apa pun pilihan yang dibuat, aplikasi seluler akan menerbitkan pesan baru ke topik kodifikasi/lampuKontrol pada broker MQTT. Broker MQTT akan menambahkan pesan ini ke topik. Perangkat yang terhubung ke lampu fisik akan menjadi pelanggan topik *Codifythings/lightsControl.* Setiap kali sebuah pesan baru diterbitkan ke topik itu akan mendapat pemberitahuan; perangkat sebagai hasilnya akan menyalakan atau mematikan lampu. Anda akan membuat kendali jarak jauh ini di Bab 8.



Gambar 3-6. Komponen kontrol pencahayaan jarak jauh

Catatan : Untuk persyaratan hardware dan software serta instruksi rangkaian, lihat bagian "Konektivitas Nirkabel Arduino Uno (WiFi)" di Bab 2.

Kode (Arduino)

Selanjutnya Anda akan menulis kode untuk menghubungkan Arduino ke Internet menggunakan WiFi dan mempublishkannya ke server menggunakan MQTT. Mulai Arduino IDE Anda dan ketik kode berikut atau unduh dari situs buku dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino) dalam urutan yang sama seperti yang disediakan di sini, tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi empat bagian.

- Library Eksternal
- Konektivitas Internet (wireless)
- Data publish (MQTT)
- Dara subscribe (MQTT)

Library Eksternal

Bagian pertama dari kode disediakan dalam Listing 3-6. Ini mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Sketsa ini memiliki dua dependensi utama. Untuk konektivitas Internet, Anda harus menyertakan <WiFi.h> (dengan asumsi Anda menggunakan shield WiFi) dan, untuk komunikasi broker MQTT, Anda harus menyertakan <PubSubClient.h>. Anda dapat menginstal library <PubSubClient.h> dari:

- <PubSubClient.h> : https://github.com/knolleary/ pubsubclient/releases/tag/v2.3

Listing 3-6. Library Eksternal

```
#include <SPI.h>
#include <WiFi.h>
#include <PubSubClient.h>
```

Konektivitas Internet (Nirkabel)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan Listing 2-9 dari Bab 2 di sini.

3.7 DATA PUBLISHKAN/SUBSCRIBE MQTT

Bagian ketiga dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menerbitkan dan subscribe ke broker MQTT. Kode menerbitkan dan subscribe topik yang sama.

Tentukan alamat dan port (default adalah 1883) dari broker MQTT yang Anda ingin Arduino untuk terhubung, seperti yang ditunjukkan pada Listing 3-7. Variabel topik

menentukan topik mana pada data broker yang akan dipublishkan dan subscribe. Jika Anda tidak menginstal broker MQTT di mesin Anda, Anda dapat menggunakan broker MQTT yang tersedia secara terbuka dari Eclipse Foundation (iot.Eclipse.org) atau Mosquitto (test.mosquitto.org).

Listing 3-7. Pengaturan MQTT

```
// IP address of the MQTT broker
char server[] = { "iot.eclipse.org" };
int port =  1883
char topic[] = { "codifythings/testMessage" };
```

Seperti yang ditunjukkan pada Listing 3-8, inisialisasi klien MQTT. Fungsi callback() merangkum semua detail penerimaan payload dari broker.

Listing 3-8. Inisialisasi MQTT dan Fungsi Panggilan Balik

```
PubSubClient pubSubClient(server, 1883, callback, client);
void callback(char* topic, byte* payload, unsigned int length)
{
    // Print payload
    String payloadContent = String((char *)payload);
    Serial.println("[INFO] Payload: " + payloadContent);
}
```

Fungsi Standar

Terakhir, kode di bagian terakhir ini disediakan di Listing 3-9. Ini mengimplementasikan fungsi setup() dan loop() standar Arduino. Dalam fungsi setup(), kode menginisialisasi port serial dan terhubung ke Internet. Jika broker MQTT terhubung, itu akan subscribe topik *codifythings/testMessage*. Setelah berhasil subscribe, kode menerbitkan pesan baru ke topik codifythings/testMessage. Kode subscribe topik yang sama dengan yang diterbitkan. Oleh karena itu, segera setelah pesan dipublishkan, fungsi callback() akan dipanggil. Fungsi loop() hanya menunggu pesan baru dari broker MQTT.

Listing 3-9. Kode untuk Fungsi Arduino Standar

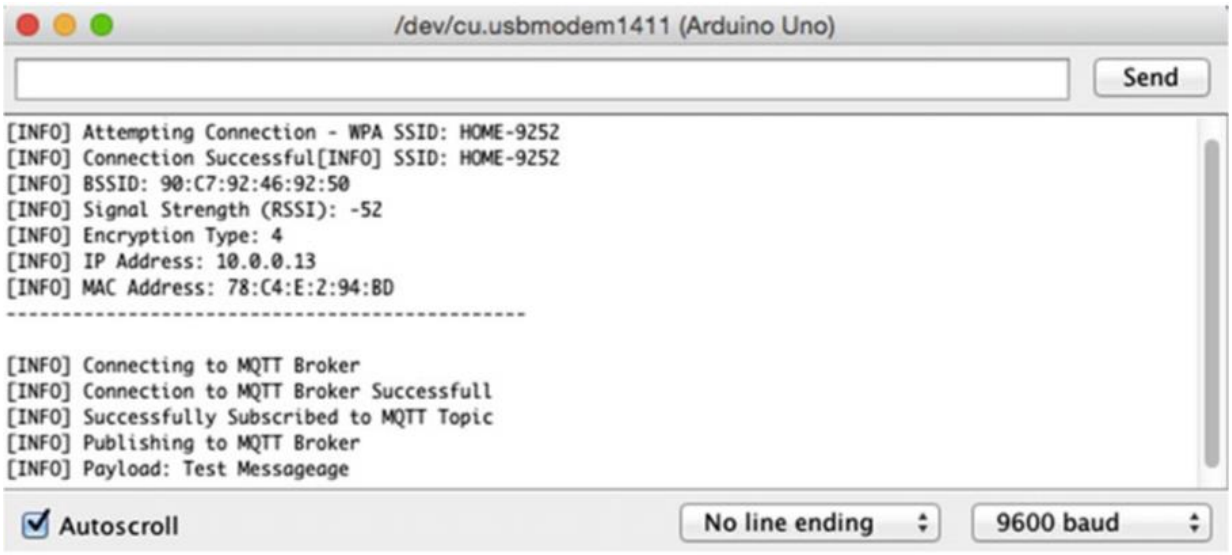
```
void setup()
{
// Initialize serial port
Serial.begin(9600);
   // Connect Arduino to internet
   connectToInternet();
   //Connect MQTT Broker
   Serial.println("[INFO] Connecting to MQTT Broker");
   if (pubSubClient.connect( "arduinoClient" ))
   {
   Serial.println("[INFO] Connection to MQTT Broker Successful");
   pubSubClient.subscribe(topic);
   Serial.println("[INFO] Successfully Subscribed to MQTT Topic ");
   Serial.println("[INFO] Publishing to MQTT Broker");
   pubSubClient.publish(topic, "Test Message");
   }
   else
   {
   Serial.println("[INFO] Connection to MQTT Broker Failed");
   }
```

```
    }
    void loop()
    {
    // Wait for messages from MQTT broker    pubSubClient.loop();
}
```

Arduino Anda sudah lengkap.

Produk Final

Untuk menguji aplikasi, klik **Verify** dan **Upload** kode seperti yang dibahas dalam Bab 1. Setelah kode diterapkan, buka jendela Serial Monitor. Anda akan mulai melihat pesan log dari Arduino seperti yang ditunjukkan pada Gambar 3-7.



Gambar 3-7. M QTT: Publishkan/subscribe pesan-pesan

3.8 RINGKASAN

Dalam bab ini, Anda mempelajari tentang HTTP dan MQTT, dua protokol komunikasi yang sangat penting, populer, dan ringan yang digunakan dalam aplikasi IoT. Protokol ini adalah perangkat agnostik, sehingga dapat digunakan untuk komunikasi dengan semua jenis perangkat atau server. Anda akan menggunakan kedua protokol ini secara ekstensif di bab-bab berikutnya.

BAGIAN 2
PROTOTIPE
BAB 4
COMPLEX FLOWS: NODE-RED

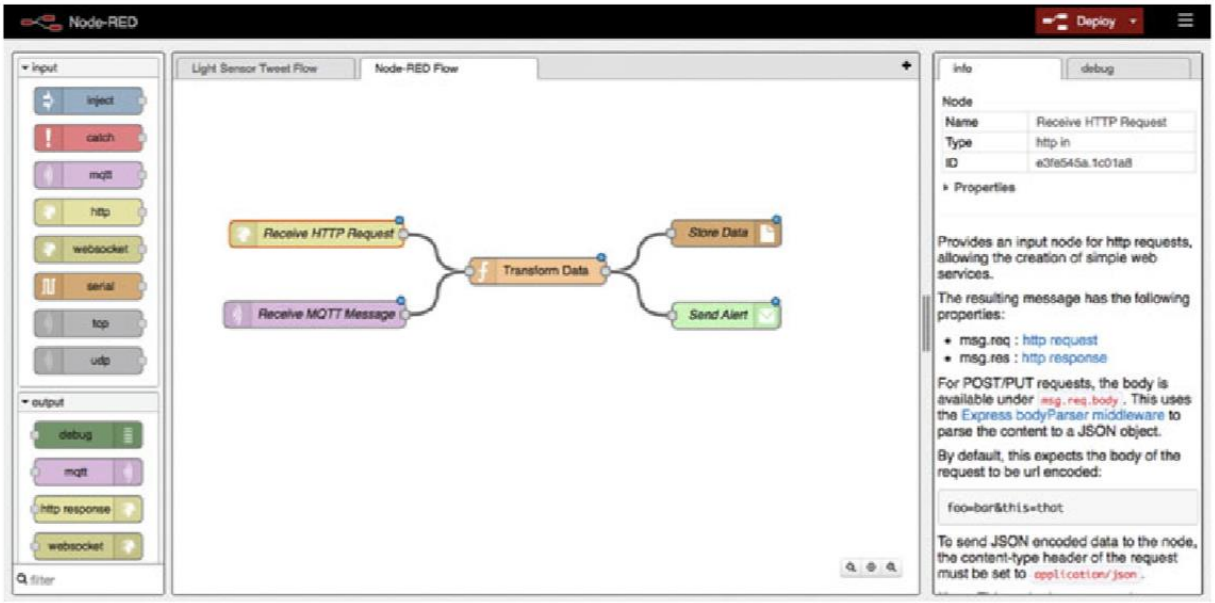
Sekarang, setelah Anda memahami dasar-dasar Arduino dan berbagai opsi konektivitas yang tersedia dan juga protokol komunikasi, Anda akan menggunakan pengetahuan itu untuk membuat prototipe aplikasi IoT.

Bab ini dimulai dengan skenario hipotetis. Bayangkan saja Anda akan bertanggung jawab untuk memantau tingkat kebisingan di sekitar suaka marga satwa. Setiap kali tingkat kebisingan melewati ambang batas tertentu, Anda diminta untuk mengirim SMS ke supervisor dan mencatat informasi kebisingan dalam database untuk analisis tren di masa mendatang. Mari kita lihat apa yang diperlukan untuk mengimplementasikan aplikasi IoT ini:

- Menghubungkan sensor suara ke Arduino
- Menulis kode yang mengirimkan permintaan HTTP ke server setiap kali tingkat kebisingan melebihi ambang batas
- Membuat layanan di server yang menerima permintaan HTTP
- Menulis layanan untuk mengirim SMS ke supervisor
- Menulis layanan untuk menyimpan data sensor dalam database

Melihat tugas-tugas tersebut, tentunya Anda tahu bahwa banyak kode yang perlu dikembangkan untuk membuat aplikasi ini. Sebagian besar aplikasi IoT memerlukan implementasi tugas-tugas seperti permintaan/tanggapan HTTP, publish/subscribe MQTT, email, SMS, tweet, dan penyimpanan/pemuatan data. Insinyur di IBM menghadapi masalah yang sama. Setiap kali mereka harus membuat prototipe baru, mereka diminta untuk mengkodekan alur dan tugas dari awal, meskipun itu berulang. Jadi, mereka mengembangkan Node-RED, yang merupakan *toolkit drag-and-drop* yang sangat baik dari kode yang dapat digunakan kembali yang melakukan tugas-tugas ini dan banyak lagi.

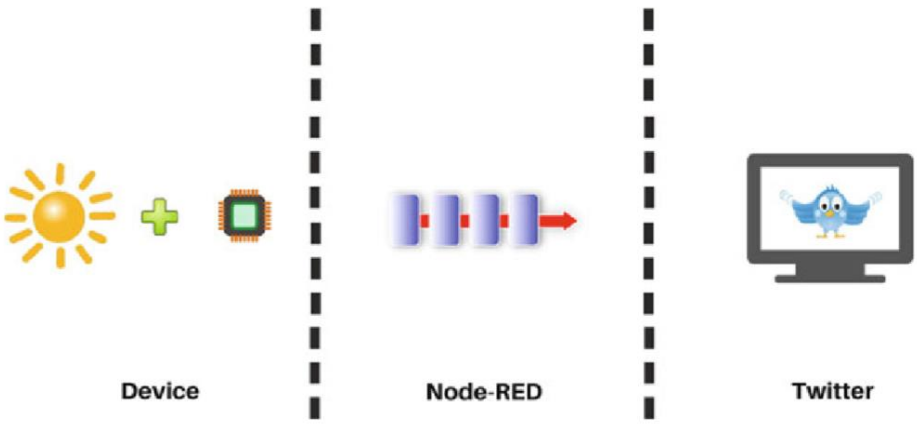
Node-RED adalah mesin pemroses peristiwa yang membantu pengembang IoT menghindari penciptaan kembali roda. Anda masih perlu menulis kode tetapi jumlah kode yang diperlukan berkurang secara signifikan. Gambar 4-1 menunjukkan lingkungan pengembangan Node-RED.



Gambar 4-1. Lingkungan pengembangan Node-RED

Seperti yang Anda lihat, aliran Node-RED terbuat dari node. Setiap node merangkum sepotong kode yang dapat digunakan kembali yang melakukan tugas tertentu. Untuk membuat aliran, Anda cukup menyeret node dari palet di sebelah kiri dan meletakkannya di desainer aliran Anda. Anda dapat menemukan banyak node yang dibuat sebelumnya dan tersedia secara terbuka untuk digunakan. Sebuah aliran dimulai setelah menerima masukan. Ada beberapa sumber input standar yang tersedia, seperti HTTP, MQTT, dan TCP. Aliran diakhiri dengan tugas keluaran seperti respons HTTP, publish MQTT, tweet, dll. Aliran tidak terbatas pada satu simpul input/output; itu bisa dimulai atau diakhiri dengan banyak node. Node di antara input dan output biasanya mengubah atau memanipulasi data, misalnya, mengubah permintaan HTTP menjadi badan email.

Anda akan membuat proyek sederhana untuk lebih mengenal NodeRED. Ide dari proyek ini adalah untuk men-tweet setiap kali di luar cerah. Gambar 4-2 menampilkan semua komponen yang akan digunakan untuk merancang sistem ini. Komponen pertama adalah perangkat Arduino dengan sensor cahaya yang terpasang padanya. Komponen kedua adalah aliran Node-RED yang dimulai oleh Arduino. Komponen terakhir adalah Twitter, karena aliran Node-RED Anda akan men-tweet pesan setiap kali ambang batas tertentu dilewati.



Gambar 4-2. Komponen sistem tweet sensor cahaya

4.1 TUJUAN PEMBELAJARAN

Di akhir bab ini, Anda akan dapat:

- Membaca data sensor cahaya dari Arduino
- Membangun aliran Node-RED yang menerima permintaan HTTP dan tweet pesan.
- Mengirim data sensor dalam permintaan HTTP untuk memulai aliran Node-RED

Hardware yang Diperlukan

Gambar 4-3 memberikan Listing semua komponen hardware yang diperlukan untuk membuat sistem tweet sensor cahaya.

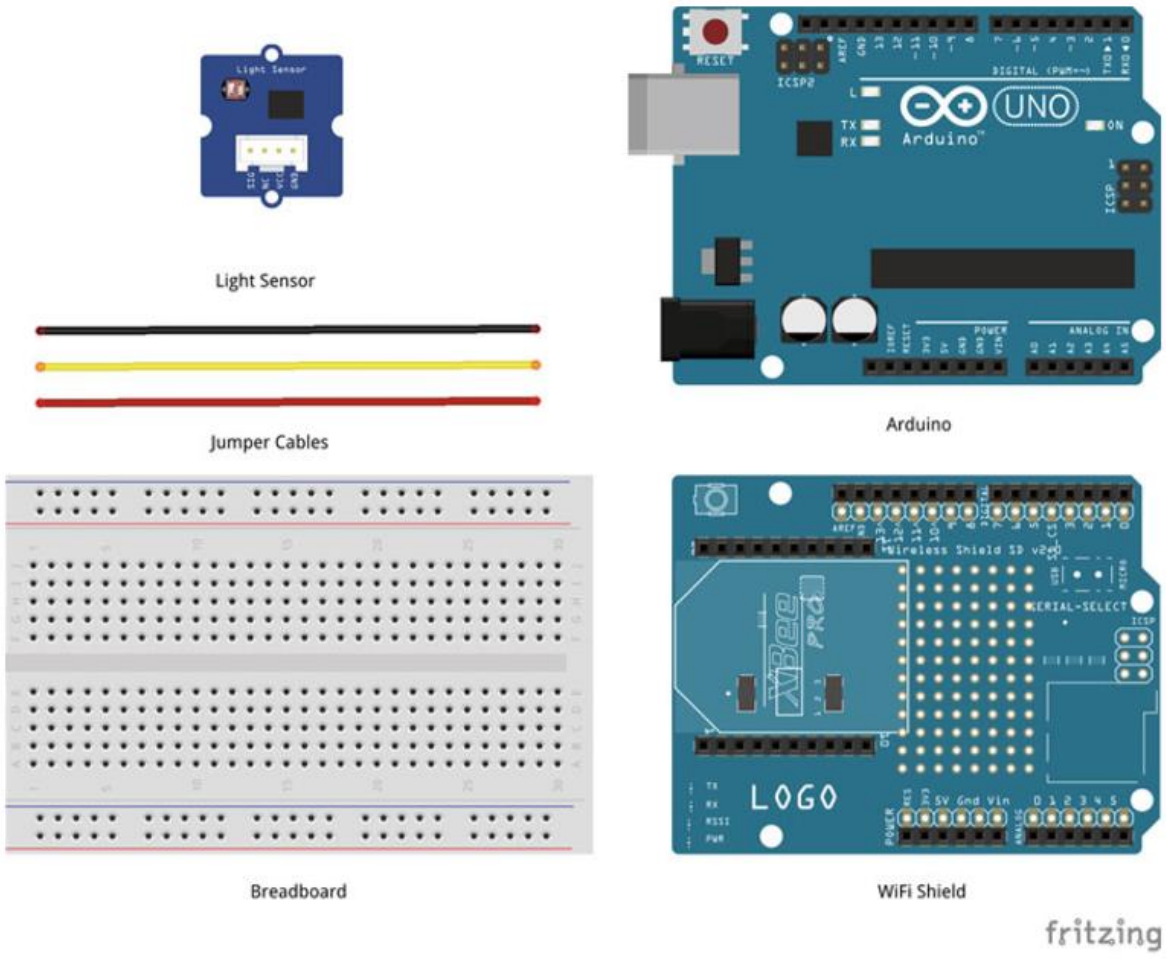
Software yang diperlukan

Untuk mengembangkan sistem tweet sensor cahaya, Anda memerlukan software berikut:

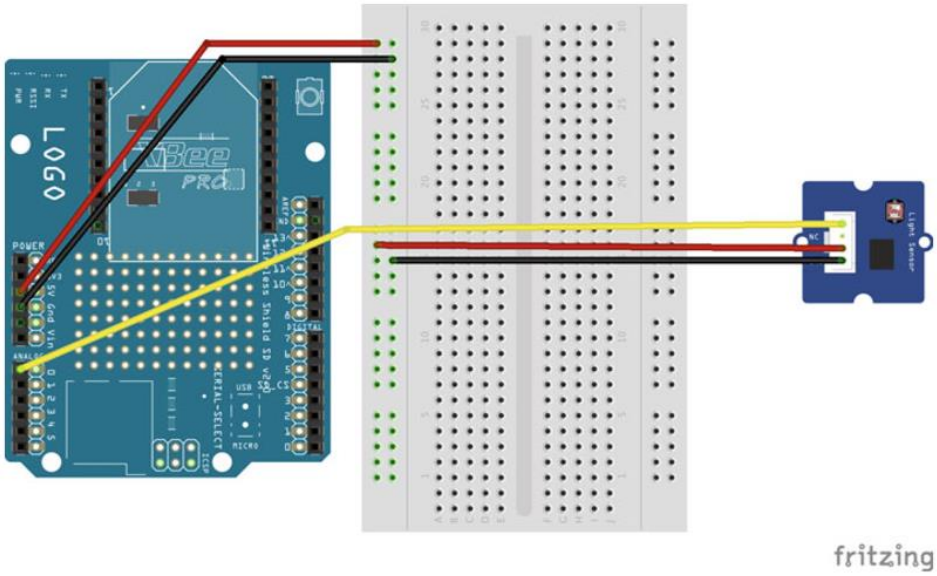
- Arduino IDE 1.6.4 atau versi yang lebih baru
- Node-RED 0.13.2 atau versi yang lebih baru

Sirkuit

Pada bagian ini Anda akan membangun sirkuit yang diperlukan untuk sistem tweet sensor cahaya. Sirkuit ini menggunakan sensor intensitas cahaya analog, yang mengembalikan nilai antara 0 dan 1023. Nilai yang lebih tinggi berarti intensitas cahaya yang lebih tinggi.



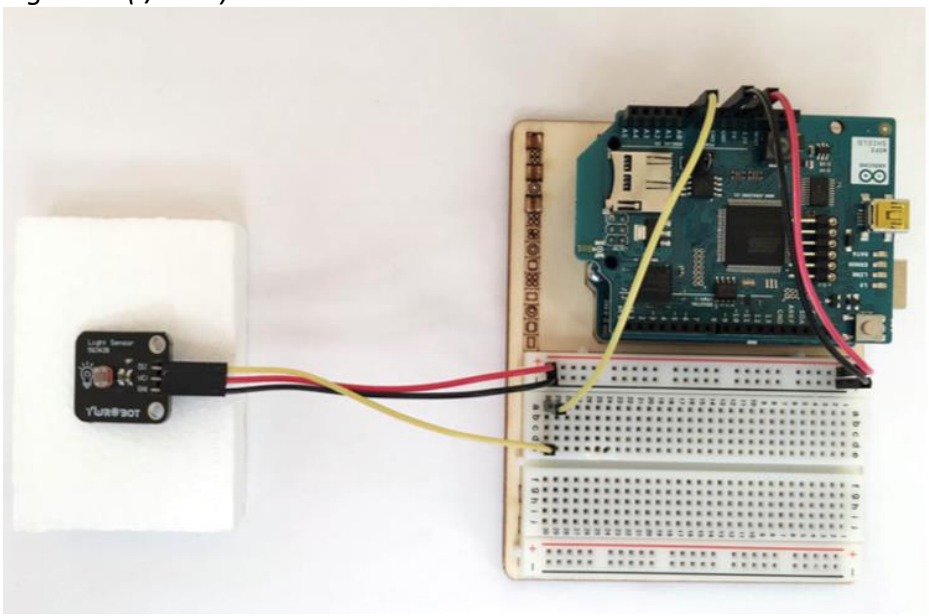
Gambar 4-3. Hardware yang diperlukan untuk sistem tweet sensor cahaya



Gambar 4-4. Diagram sirkuit sistem tweet sensor cahaya

1. Pastikan Arduino Anda tidak terhubung ke sumber listrik, seperti komputer melalui USB atau baterai.
2. Pasang pelindung WiFi ke bagian atas Arduino. Semua pin harus sejajar.
3. Menggunakan kabel jumper, sambungkan port power (5V) dan ground (GND) pada Arduino ke port power (+) dan ground (-) pada breadboard.
4. Sekarang papan breadboard Anda memiliki sumber daya, gunakan kabel jumper untuk menghubungkan port daya (+) dan arde (-) papan breadboard Anda ke port daya dan arde dari sensor cahaya.
5. Untuk membaca nilai sensor cahaya, Anda perlu menghubungkan kabel jumper dari port baca analog sensor cahaya ke port A0 (analog) Arduino Anda. Kode Anda akan menggunakan port ini untuk membaca nilai intensitas cahaya. Sirkuit Anda sekarang sudah selesai dan seharusnya terlihat seperti Gambar 4-4 dan 4-5.

***Tip** : Sebaiknya gunakan kabel jumper merah untuk daya (+/VNC/5V/3.3V) dan kabel jumper hitam untuk ground (-/GND).*



Gambar 4-5. Sirkuit sebenarnya dari sistem tweet sensor cahaya

```
Adeels-MacBook-Air:~ adeeljaved$ node-red

Welcome to Node-RED
===================

4 Oct 17:25:28 - [info] Node-RED version: v0.10.6
4 Oct 17:25:28 - [info] Node.js  version: v0.10.36
4 Oct 17:25:28 - [info] Loading palette nodes
4 Oct 17:25:29 - [warn] ---------------------------------------
4 Oct 17:25:29 - [warn] Failed to register 4 node types
4 Oct 17:25:29 - [warn] Run with -v for details
4 Oct 17:25:29 - [warn] ---------------------------------------
4 Oct 17:25:29 - [info] User Directory : /Users/adeeljaved/.node-red
4 Oct 17:25:29 - [info] Flows file     : /Users/adeeljaved/.node-red/flows_Adeel
s-MacBook-Air.local.json
4 Oct 17:25:30 - [info] Server now running at http://127.0.0.1:1880/
4 Oct 17:25:30 - [info] Starting flows
4 Oct 17:25:30 - [info] Started flows
```

Gambar 4-6. Log startup dari Node-RED

4.2 Aliran Node-RED

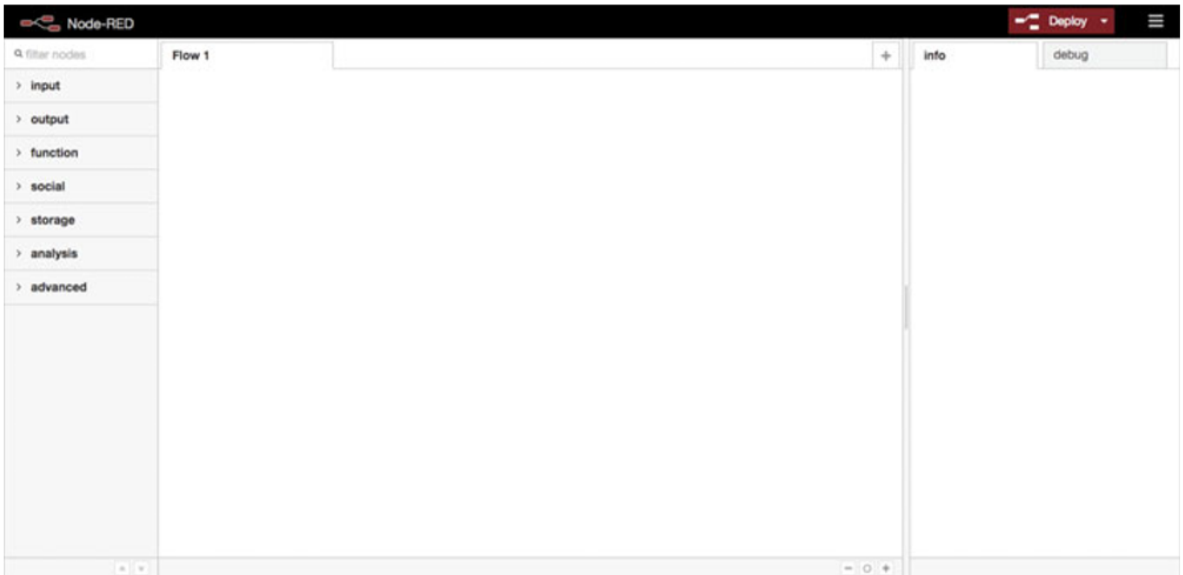
***Catatan** : Buku ini tidak mencakup instalasi server Node-RED. Anda dapat menemukan petunjuk instalasi di situs web resmi Node-RED (http://nodered.org/docs/get-started/installation.html).*

Di bagian ini Anda akan mengembangkan aliran di Node-RED yang akan melakukan berikut:

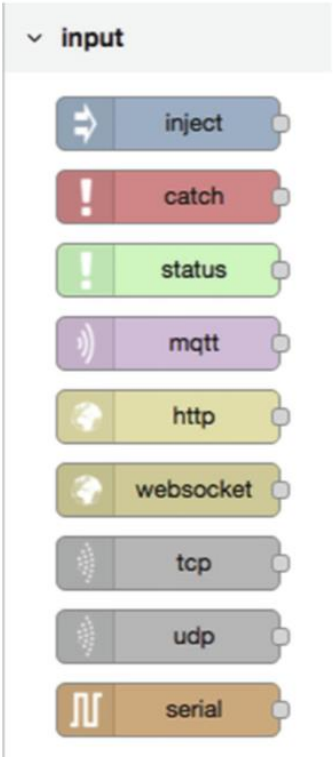
- Menerima permintaan HTTP yang dikirim dari sensor cahaya
- Menyiapkan tweet menggunakan data yang dikirim oleh sensor cahaya
- Men-Tweet pesannya
- Mengirim tanggapan HTTP

Mulai server Node-RED Anda menggunakan perintah node-red di jendela terminal. Gambar 4-6 menunjukkan pesan log yang akan Anda lihat setelah server Node-RED dimulai. Pada Gambar 4-6, Server pesan log yang sekarang berjalan di http://127.0.0.1:1880 berisi URL yang tepat dari server Node-RED.

***Catatan**: URL server Node-RED di log http://127.0.0.1:1880 adalah IP komputer lokal Anda dan tidak dapat diakses oleh Arduino. Anda harus mengganti IP lokal 127.0.0.1 dengan IP jaringan mesin Anda. IP server Node-RED yang digunakan dalam buku ini adalah 10.0.0.6, jadi URL yang akan Anda lihat adalah http://10.0.0.6:1880.*



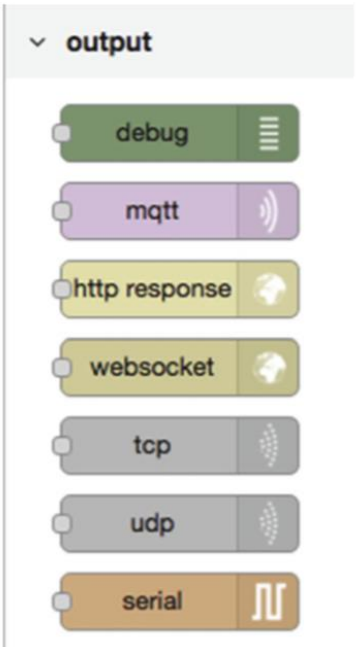
Gambar 4-7. Tampilan default dari desainer Node-RED



Gambar 4-8. Masukkan node dalam instalasi default Node-RED

Masukkan URL server Node-RED di browser untuk mengakses desainer. Desainer membuka dengan tab aliran kosong yang disebut Aliran 1. Gambar 4-7 menunjukkan tampilan default desainer Node-RED.

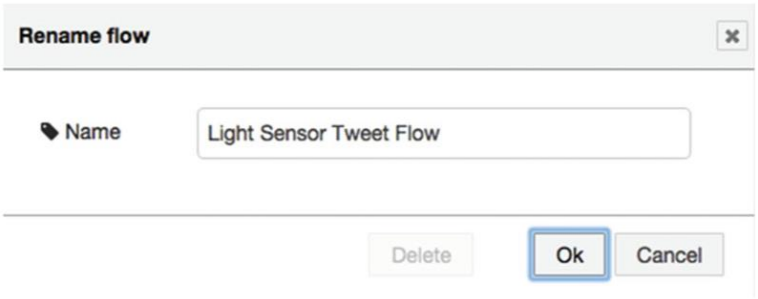
Di sisi kiri desainer, seperti yang ditunjukkan pada Gambar 4-7, adalah palet dengan semua node yang tersedia. Node dikelompokkan ke dalam berbagai kategori, seperti input, output, fungsi, dll. Gambar 4-8 menunjukkan Listing node input yang disertakan dengan instalasi default Node-RED, dan Gambar 4-9 menunjukkan Listing node output di instalasi bawaan.



Gambar 4-9. Output node dalam instalasi default Node-RED

Di sisi kanan desainer, seperti yang ditunjukkan pada Gambar 4-7, terdapat tab Info dan Debug. Tab Info menampilkan dokumentasi tentang node yang dipilih saat ini di palet Node atau tab Flow. Tab Debug menampilkan pesan log dan kesalahan yang dihasilkan dari aliran selama eksekusi. Terakhir, tombol Terapkan di kanan atas desainer, seperti yang ditunjukkan pada Gambar 4-7, memungkinkan Anda menerapkan dan mengaktifkan perubahan alur ke server.

Sekarang mari kita mulai membuat alurnya. Jika ini adalah aliran pertama Anda di Node-RED, Anda dapat menggunakan Aliran 1 untuk membuat aliran Anda. Jika Anda telah membuat beberapa aliran dan ingin membuat yang baru, klik tombol plus (+) di sisi kanan atas untuk menambahkan aliran baru. Klik dua kali nama tab aliran untuk membuka kotak dialog properti yang ditunjukkan pada Gambar 4-10. Panggil aliran baru Sensor Cahaya Tweet Flow lalu klik OK untuk menyimpan perubahan Anda.

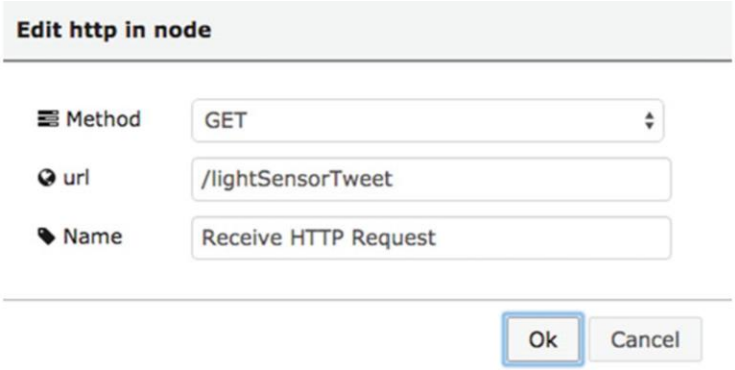


Gambar 4-10. Kotak dialog properti aliran

Drag dan jatuhkan simpul masukan permintaan http dari palet di tab flow. Alur Anda akan terlihat mirip dengan Gambar 4-11.

Gambar 4-11. permintaan node Http

Klik dua kali simpul http untuk membuka kotak dialog properti, seperti yang ditunjukkan pada Gambar 4-12. Setel metode ke **GET**, yang menetapkan bahwa permintaan HTTP akan dikirim oleh klien (dalam hal ini, sensor cahaya) menggunakan metode GET. Seperti yang dibahas dalam Bab 3, struktur permintaan bervariasi berdasarkan metode yang Anda pilih. Anda melihat kode Arduino untuk metode GET dan POST di Bab 3.



Gambar 4-12. Kotak dialog properti node permintaan H TTP

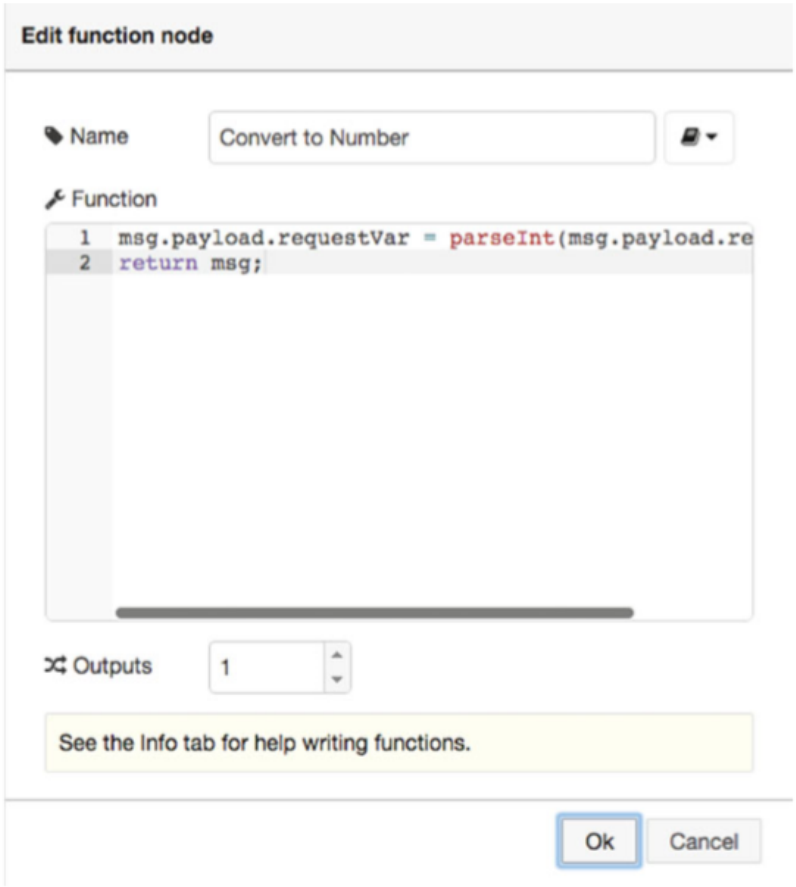
Setel properti URL ke /lightSensorTweet. URL ini akan didahului oleh server dan port Node-RED. Server Node-RED yang digunakan dalam proyek ini tersedia di 10.0.0.6:1880, sehingga Arduino akan mengirimkan data ke **10.0.0.6:1880/lightSensorTweet**. Terakhir,

setiap node dapat diberi nama khusus yang menjelaskan tugas yang dilakukannya. Panggil node ini Terima Permintaan HTTP.

Klik **OK** untuk melakukan pembaruan. Data yang berasal dari perangkat yang menggunakan HTTP dalam format string, jadi Anda perlu mengubahnya menjadi angka. Drag dan lepas node fungsi dan letakkan di **tab Flow** setelah **Receive HTTP Request node**. Sebuah node fungsi memungkinkan Anda menulis kode untuk memanipulasi payload. Alur Anda akan terlihat mirip dengan Gambar 4-13 pada saat ini.

Gambar 4-13. simpul fungsi

Klik dua kali node fungsi untuk membuka dialog properti, seperti yang ditunjukkan pada Gambar 4-14. **Rename** menjadi **Convert to Number**. **Update** kode di dalam fungsi seperti yang disediakan dalam Listing 4-1. Klik **OK** untuk menyimpan perubahan Anda. Hubungkan Anda Terima Permintaan HTTP dan Konversikan ke node Nomor.



Gambar 4-14. Kotak dialog properti node fungsi

Listing 4-1. Kode untuk Mengonversi String ke Angka

```
msg.payload.requestVar = parseInt(msg.payload.requestVar);
return msg;
```

Pada titik ini, sensor cahaya Anda akan mengirim bacaan setiap beberapa detik baik saat cerah maupun tidak. Jadi dalam aliran Node-RED, Anda perlu menambahkan aturan untuk memeriksa apakah nilai sensor telah melewati ambang batas tertentu dan hanya men-tweet ketika ambang batas tersebut telah dilewati.

Anda juga dapat menambahkan pemeriksaan ambang batas ini dalam kode Arduino, tetapi pertimbangkan implementasi kehidupan nyata dari proyek yang sama ini. Alih-alih men-tweet, Anda juga dapat menggunakan logika yang sama untuk membangun aplikasi yang menghemat energi dengan membuka tirai jendela dan mematikan lampu di dalam rumah jika di luar cerah.

Jika Anda meng-hard-code pemeriksaan tersebut dalam kode Arduino, maka pengguna individu mungkin tidak dapat mengatur preferensi cahaya mereka, karena mereka tidak dapat secara langsung **mengupdate** kode. Mengambil logika seperti itu dari sensor akan memungkinkan Anda membangun sesuatu yang dapat disesuaikan oleh pengguna individu. Drag dan lepas simpul sakelar dari kategori fungsi dan letakkan di tab Alur setelah simpul Konversi ke Nomor. Alur Anda akan terlihat mirip dengan Gambar 4-15 pada saat ini.

Gambar 4-15. Beralih simpul

Sebuah **node switch** memungkinkan Anda mengikuti jalur tertentu dalam aliran berdasarkan suatu kondisi. Klik dua kali node switch untuk membuka kotak dialog propertinya dan menyetel kondisi yang ditunjukkan pada Gambar 4-16.

Gambar 4-16. Beralih kotak dialog properti simpul

Ubah nama menjadi Periksa Ambang. Secara default, hanya akan ada satu jalur, jadi klik tombol + Aturan untuk menambahkan jalur baru. Jika nilai sensor lebih besar dari 750, maka akan mengikuti jalur 1; jika tidak, itu akan mengikuti jalur 2. Jalur 2 tidak akan memeriksa kondisi apa pun, jadi Anda dapat mengubahnya dari dropdown.

Node-RED menyimpan semua informasi input di msg.payload. Anda akan mengirimkan nilai sensor di *requestVar* dari Arduino, itulah sebabnya kondisi memeriksa msg. payload.requestVar.

Hubungkan node Convert to Number dan Check Threshold Anda. Anda akan menggunakan nilai sensor untuk membuat pesan tweet. Drag dan jatuhkan simpul fungsi ke diagram alir. Tempatkan setelah node Check Threshold, seperti yang ditunjukkan pada Gambar 4-17.

Gambar 4-17. simpul fungsi

Klik dua kali node fungsi untuk membuka kotak dialog properti, seperti yang ditunjukkan pada Gambar 4-18. Beri nama **Setup Tweet Message**. Update kode di dalam simpul fungsi, seperti yang ditunjukkan pada Listing 4-2. Klik **OK** untuk menyimpan perubahan Anda.

Gambar 4-18. Kotak dialog properti node fungsi

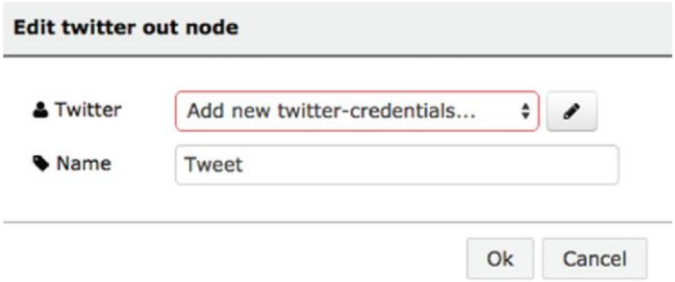
Listing 4-2. Kode untuk Membuat Tweet

```
msg.payload = "Sunny Outside! " + msg.payload.requestVar + " #IoT";  return msg;
```

Hubungkan node Set Tweet Message ke jalur pertama dari node switch Check Threshold. Koneksi ini akan memastikan bahwa setiap kali nilai sensor cahaya melewati ambang 750, aliran mengikuti jalur 1 yang men-tweet. Selanjutnya, drag dan lepas node Tweet ke diagram alur setelah node Setel Pesan Tweet, seperti yang ditunjukkan pada Gambar 4-19.

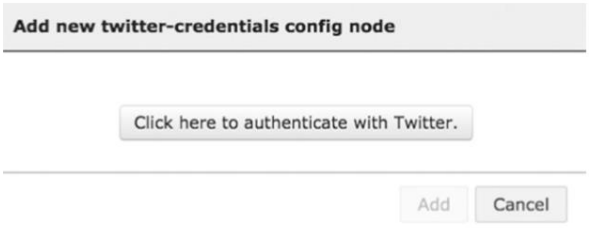
Gambar 4-19. simpul tweet

Agar Node-RED dapat menge-tweet, Anda perlu mengonfigurasi kredensial Twitter Anda. Klik dua kali simpul twitter out untuk membuka kotak dialog properti yang ditunjukkan pada Gambar 4-20. Jika Anda sudah mengonfigurasi kredensial Twitter Anda di Node-RED, pilih dari dropdown Twitter. Jika tidak, pilih opsi Add New Twitter- Credentials dari dropdown dan klik ikon Edit/Pencil untuk memulai langkah-langkah konfigurasi.

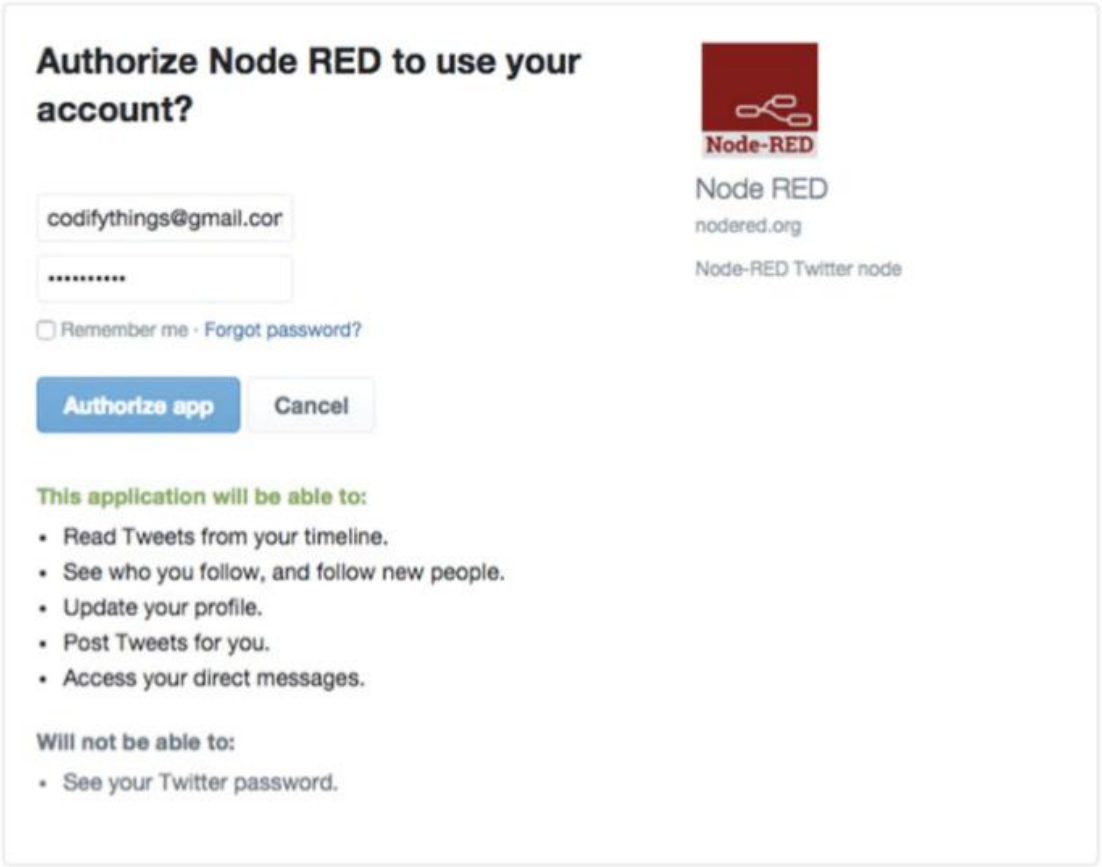


Gambar 4-20. Tambahkan kredensial Twitter baru

Gambar 4-21 menunjukkan kotak dialog berikutnya yang muncul. Klik tombol **Click here** untuk Otentikasi dengan Twitter.



Gambar 4-21. Otentikasi akun Twitter

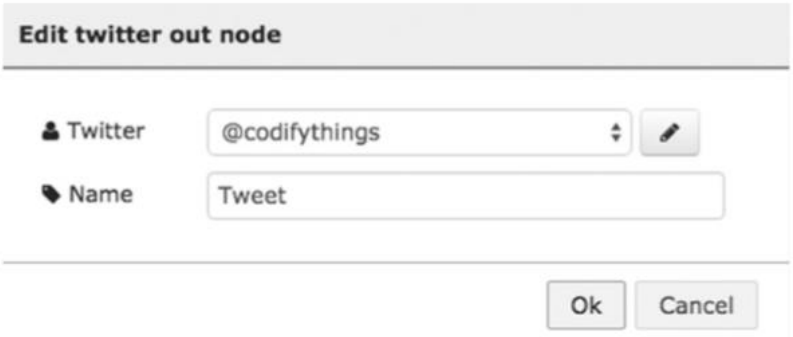


Gambar 4-22. Beri wewenang Node-RED untuk menggunakan akun Twitter Anda

Di layar berikutnya, yang ditunjukkan pada Gambar 4-22, masukkan **username** dan **password** Twitter Anda dan klik tombol Authorize App untuk memberikan akses Node-RED ke akun Twitter Anda.
Setelah proses otorisasi selesai, klik tombol Tambah di kotak dialog yang ditunjukkan pada Gambar 4-23.

Gambar 4-23. Tambahkan akun Twitter resmi ke alur

Gambar 4-24 menunjukkan kotak dialog yang akan ditampilkan berikutnya; itu adalah kotak dialog yang sama tempat Anda memulai proses konfigurasi Twitter. Klik **OK** untuk menyelesaikan konfigurasi Twitter.



Gambar 4-24. S pilih kredensial Twitter resmi

Hubungkan simpul Tweet ke simpul Setel Pesan Tweet. Terakhir, tambahkan simpul respons HTTP ke aliran Anda di bawah simpul Twitter. Alur Anda akan terlihat seperti Gambar 4-25.

Gambar 4-25. Node respons HTTP

Node respons HTTP hanya akan mengirim msg.payload kembali ke klien dalam format JSON. Ubah nama node ini menjadi **Send HTTP Response**. Hubungkan node Send HTTP Response ke jalur kedua dari node switch Check Threshold dan juga ke Set Tweet Message. Alur akhir Anda akan terlihat mirip dengan Gambar 4-26.

Kode (Arduino)

Selanjutnya, Anda akan menulis kode untuk menghubungkan Arduino ke Internet menggunakan WiFi, membaca data sensor cahaya, dan mengirimkannya ke server Node-RED

sebagai permintaan HTTP. Mulai Arduino IDE Anda dan ketik kode yang disediakan di sini atau unduh dari situs buku dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi lima bagian.

- Library Eksternal
- Konektivitas Internet (WiFi)
- Read sensor data
- HTTP (GET)
- Fungsi Standar

Library Eksternal

Bagian pertama dari kode, seperti yang disediakan dalam Listing 4-3, mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Karena Anda terhubung ke Internet secara nirkabel, ketergantungan utama kode ada pada <WiFi.h>.

Listing 4-3. Kode untuk Menyertakan Dependensi Eksternal

```
#include <SPI.h>
#include <WiFi.h>
```

Internet Konektivitas (Wireless)

Bagian kedua dari kode mendefinisikan variabel, konstanta dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (Bab 2) di sini.

4.3 BACA DATA SENSOR

Bagian ketiga dari kode ini disediakan dalam Listing 4-4. Ini mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membaca data sensor. Fungsi readSensorData() membaca data dari Pin Analog A0, dan hasilnya antara 0 dan 1023. Semakin besar nilai yang dikembalikan, semakin terang sumber cahayanya. Nilai sensor cahaya ditetapkan ke variabel lightValue.

Listing 4-4. Kode untuk Data Sensor Lampu Baca

```
int lightValue;
        void readSensorData()
        {
        //Read Light Sensor Value
        lightValue = analogRead(A0);

        Serial.print("[INFO] Light Sensor Reading: ");
        Serial.println(lightValue);
}
```

4.4 DATA PUBLISH

Bagian keempat dari kode ini disediakan di Listing 4-5. Ini mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membuat dan mengirim permintaan HTTP ke server. Kode ini adalah versi HTTP GET yang sedikit dimodifikasi yang Anda kembangkan di Bab 3.

Modifikasi utama dalam kode ini adalah kemampuannya untuk membuka dan menutup koneksi ke server secara berulang. Selain itu, pastikan untuk mengubah nilai server dan port ke nilai server Node-RED Anda. Perubahan lainnya termasuk meneruskan variabel lightValue dalam permintaan dan menjalankan /lightSensorTweet URL.

Listing 4-5. Kode untuk Memulai Aliran Node-RED Menggunakan Permintaan HTTP

```
//IP address of the HTTP server
char server[] = { "10.0.0.6" };
int port =  1880 ;
unsigned long lastConnectionTime s= 0;
const unsigned long postingInterval = 10L * 1000L;
void doHttpGet()
{
        // Read all incoming data (if any)
        while (client.available())
        {
        char c = client.read();
        Serial.write(c);
        }
                Serial.println();
                Serial.println("---------------------------------------");
                if (millis() - lastConnectionTime > postingInterval)
                {
                client.stop();
                //Read sensor data
                readSensorData();
                // Prepare data or parameters that need to be posted to server
                String requestData = "requestVar=" + String(lightValue);
                Serial.println("[INFO] Connecting to Server");
                // Check if a connection to server:port was made
                if (client.connect(server, port))
                {
                Serial.println("[INFO] Server Connected - HTTP GET Started");
                // Make HTTP GET request
                client.println("GET /lightSensorTweet?" + requestData + " HTTP/1.1");
                client.println("Host: " + String(server));
                client.println("Connection: close");
                client.println();
                lastConnectionTime = millis();
                Serial.println("[INFO] HTTP GET Completed");
                }
                Else
                {
                // Connection to server:port failed
                Serial.println("[ERROR] Connection failed");
        }
}
}
```

4.5 FUNGSI STANDAR

Terakhir, kode di bagian kelima dan terakhir disediakan di Listing 4-6. Ini mengimplementasikan fungsi setup() dan loop() standar Arduino. Fungsi setup() menginisialisasi port serial dan menghubungkan ke Internet. Saat berada dalam fungsi loop(),

ia memanggil doHttpGet() pada interval 5.000 milidetik. Fungsi doHttpGet() membaca data sensor dan mengirimkan nilai sensor ini ke Node-RED dalam permintaan HTTP.

Listing 4-6. Kode untuk Fungsi Arduino Standar

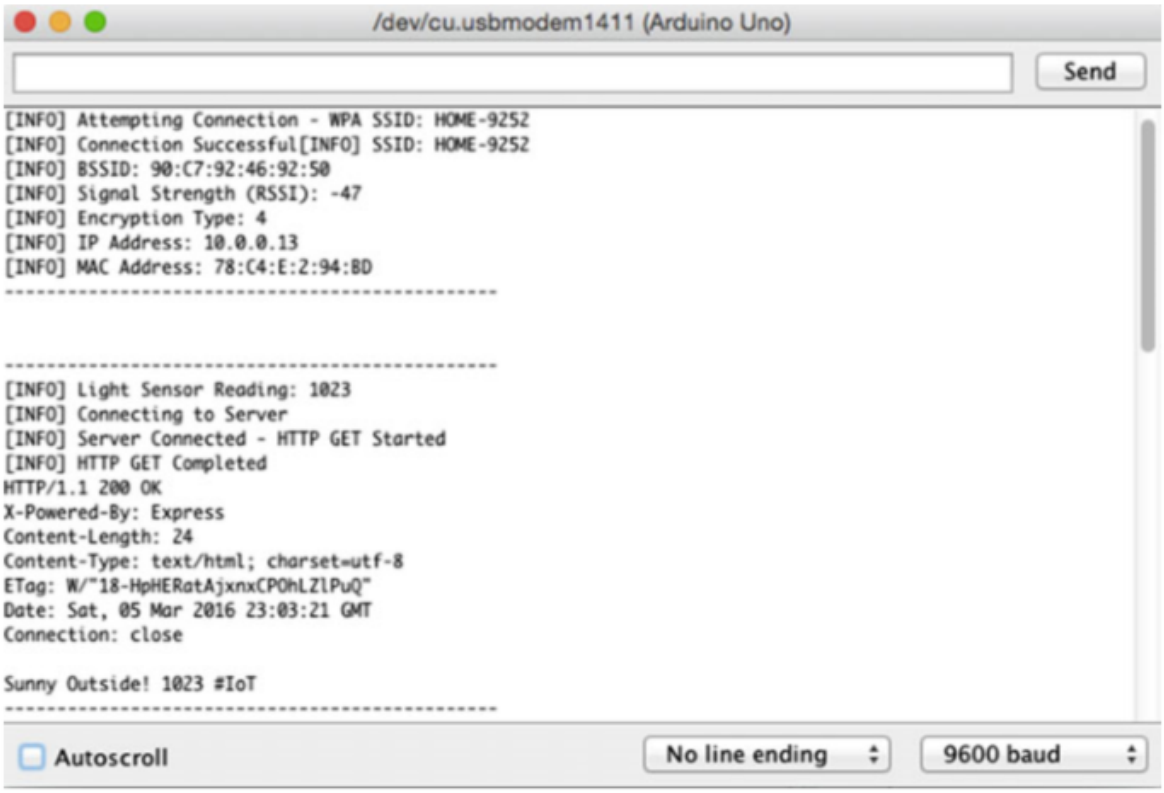
```
void setup()
{
  // Initialize serial port    Serial.begin(9600);
  // Connect Arduino to internet    connectToInternet();
  }
  void loop()
  {
  // Make HTTP GET request
  doHttpGet();
  delay(5000);
}
```

Arduino Anda lengkap

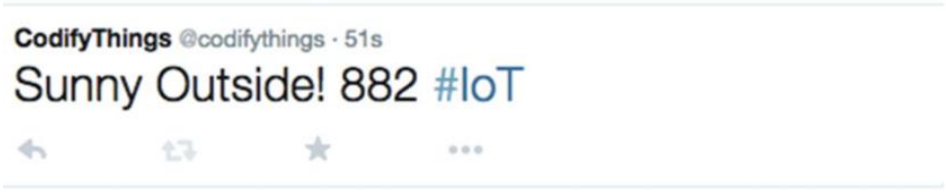
4.6 PRODUK AKHIR

Untuk menguji aplikasi, pastikan server Node-RED Anda aktif dan berjalan dengan alur yang diterapkan. Juga verifikasi dan Upload kode Arduino, seperti yang dibahas dalam Bab 1. Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log yang mirip dengan yang ditunjukkan pada Gambar 4-27.

Arduino akan terus mengirimkan data ke server, sehingga segera setelah Anda menempatkan sensor dalam cahaya terang, kondisi aliran Node-RED akan menjadi benar dan tweet akan dikirim. Hal ini ditunjukkan pada Gambar 4-28. Tidak ada syarat untuk mengirim ini sekali, jadi aplikasi akan terus mengirim tweet kecuali sensor dipindahkan dari cahaya terang atau dimatikan.



Gambar 4-27. Pesan log dari sistem tweet sensor cahaya



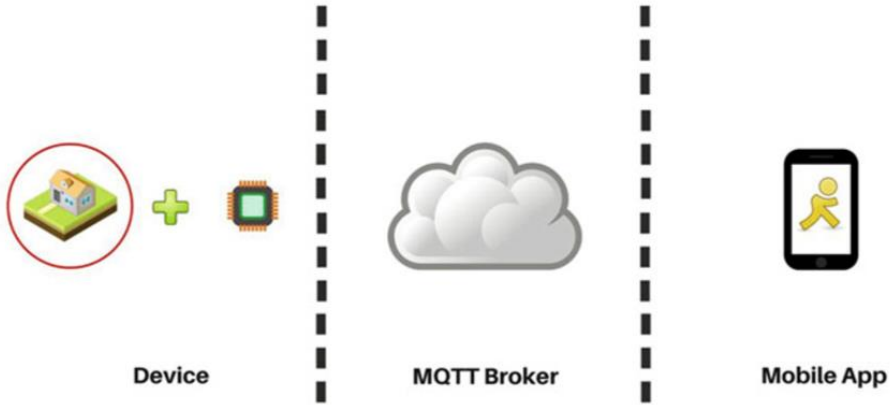
Gambar 4-28. Tweet dari sistem tweet sensor cahaya

4.7 RINGKASAN

Dalam bab ini Anda belajar tentang Node-RED dan mengembangkan aliran sederhana yang diprakarsai oleh Arduino. Aliran ini menerbitkan tweet setiap kali nilai ambang batas tertentu dilewati. Anda dapat memanfaatkan ratusan node yang tersedia di Node-RED untuk mempercepat pengembangan aplikasi IoT Anda.

BAB 5
POLA IoT : REALTIME CLIENT

Pola penting IoT adalah kemampuan untuk merasakan data dan membuatnya tersedia bagi pengguna secara realtime, seperti dengan solusi pemantauan rumah, aplikasi keamanan perimeter, dan peringatan inventaris. Dalam bab ini, Anda akan membuat contoh pola ini, sistem deteksi intrusi. Gambar 5-1 menunjukkan komponen sistem deteksi penyusupan. Komponen pertama adalah perangkat Arduino yang memiliki sensor gerak yang terpasang padanya. Komponen kedua adalah broker MQTT. Anda akan menggunakan model publish-subscribe dari MQTT untuk mengirimkan notifikasi deteksi penyusupan secara realtime (untuk detailnya, lihat Bab 3). Komponen terakhir dari aplikasi IoT Anda adalah aplikasi Android yang subscribe broker MQTT dan menunjukkan pemberitahuan peringatan kepada pengguna setiap kali Arduino mendeteksi intrusi dan menerbitkan pesan ke broker MQTT.



Gambar 5-1. Komponen sistem deteksi intrusi

5.1 TUJUAN PEMBELAJARAN

Tujuan Pembelajaran Di akhir bab ini, Anda akan dapat:

- Membaca data sensor gerak dari Arduino
- Publishkan data sensor ke broker MQTT
- Bangun aplikasi Android yang subscribe broker MQTT
- Tampilkan pemberitahuan di aplikasi setiap kali pesan baru dipublishkan ke broker MQTT

Hardware yang Diperlukan

Gambar 5-2 memberikan Listing semua komponen hardware yang diperlukan untuk membangun sistem deteksi intrusi.

Software yang Diperlukan

Untuk mengembangkan sistem deteksi intrusi, Anda memerlukan software berikut:

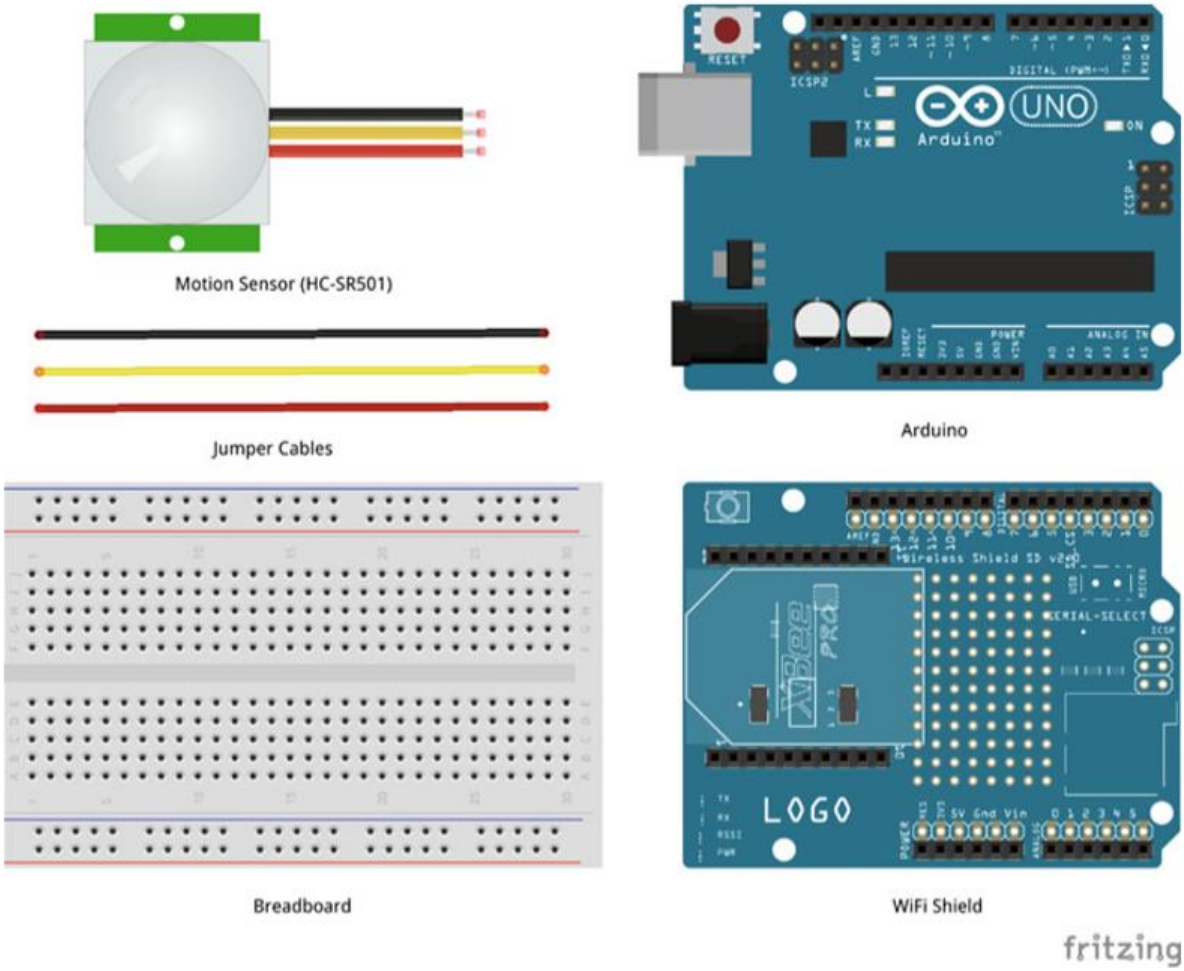
- Arduino IDE 1.6.4 atau versi yang lebih baru
- Android Studio 1.5.1 atau lebih baru

Sirkuit

Pada bagian ini, Anda akan membangun sirkuit yang diperlukan untuk sistem deteksi intrusi. Sirkuit ini menggunakan sensor gerak HC-SR501 untuk mendeteksi intrusi.

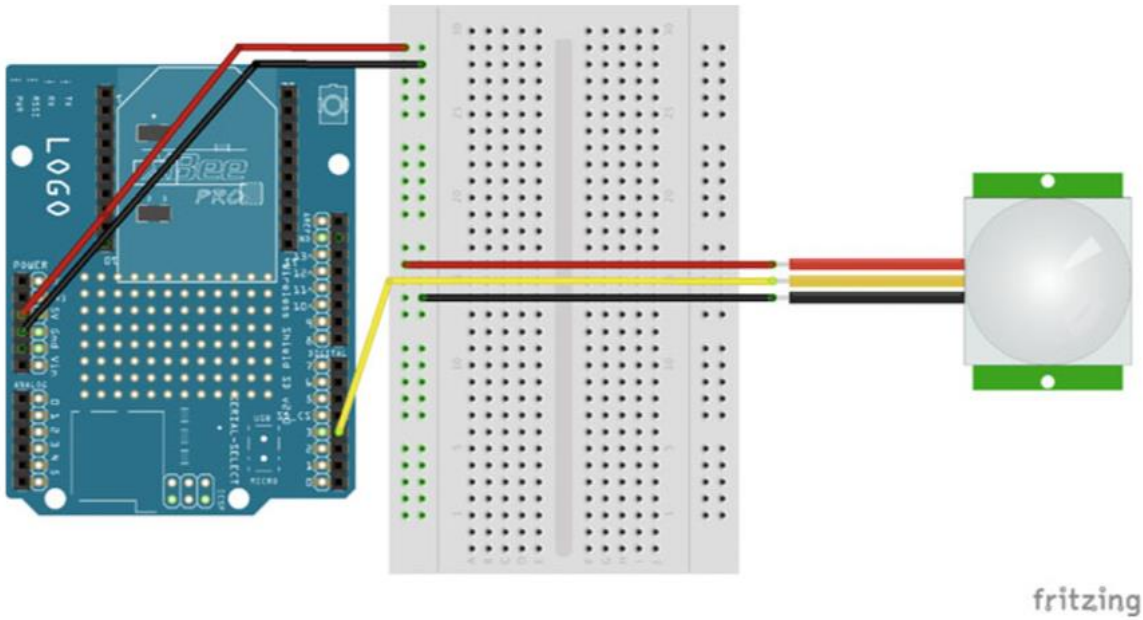
1. Pastikan Arduino Anda tidak terhubung ke sumber listrik, seperti ke komputer melalui USB atau baterai.
2. Pasang pelindung WiFi ke bagian atas Arduino. Semua pin harus sejajar.

3. Gunakan kabel jumper untuk menghubungkan port power (5V) dan ground (GND) pada Arduino ke port power (+) dan ground (-) pada breadboard.
4. Sekarang papan breadboard Anda memiliki sumber daya, gunakan kabel jumper untuk menghubungkan port daya (+) dan arde (-) papan breadboard Anda ke port daya dan arde dari sensor gerak.
5. Untuk membaca nilai sensor gerak, Anda perlu menghubungkan kabel jumper dari port sinyal sensor gerak (biasanya port tengah) ke port digital 3 Arduino Anda. Anda juga dapat menggunakan port digital lain, tetapi jika Anda melakukannya, pastikan untuk mengubah kode Arduino dengan benar.

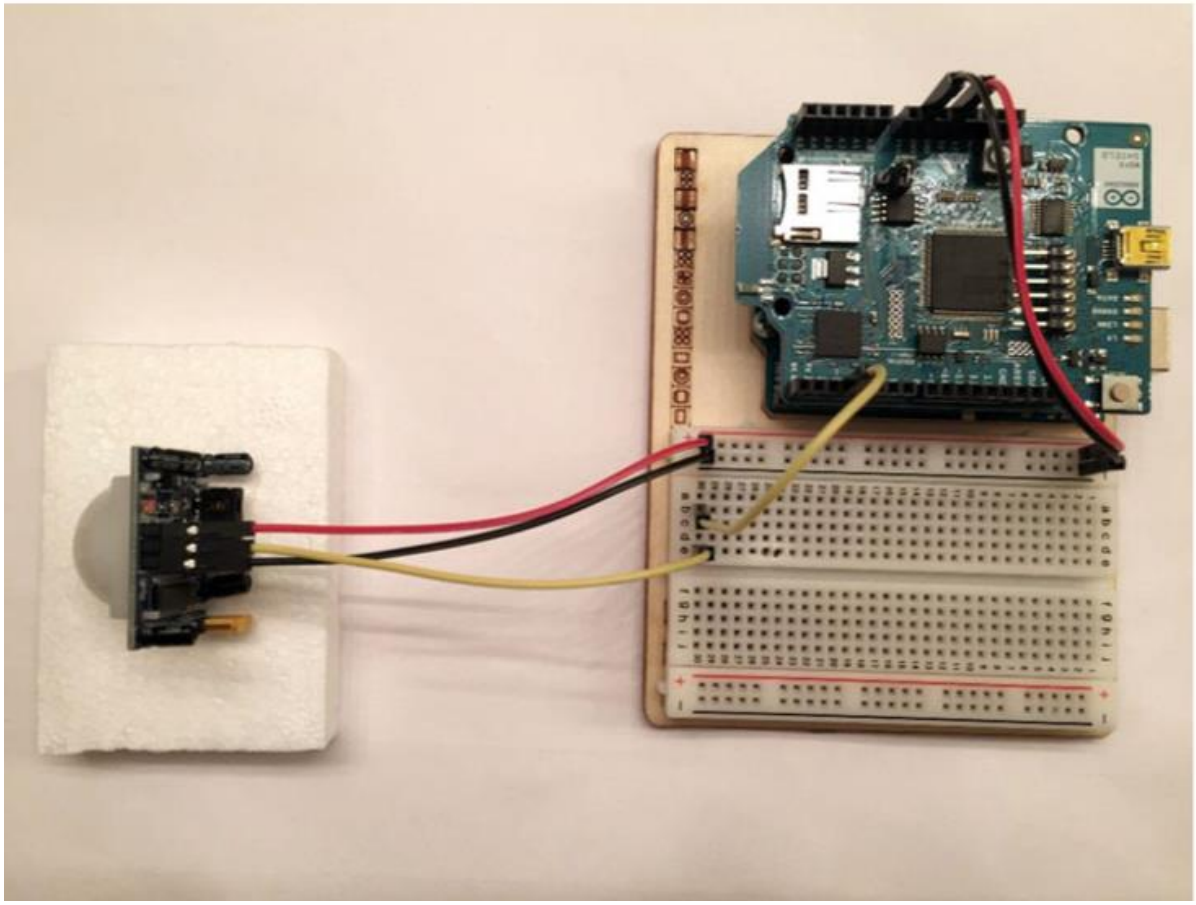


Gambar 5-2. Hardware yang diperlukan untuk sistem deteksi intrusi

Sirkuit Anda sekarang telah selesai dan seharusnya terlihat seperti Gambar 5-3 dan 5-4.



Gambar 5-3. Diagram sirkuit dari sistem deteksi intrusi



Gambar 5-4. Sirkuit sebenarnya dari sistem deteksi intrusi

Kode (Arduino)

Selanjutnya Anda akan menulis kode untuk menghubungkan Arduino ke Internet menggunakan WiFi, membaca data sensor gerak, dan memublikasikannya ke broker MQTT. Mulai Arduino IDE Anda dan ketik kode yang disediakan di sini atau unduh dari situs buku dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk

membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi lima bagian.

- Library eksternal
- Konektivitas Internet (Wi-Fi)
- Baca data sensor
- MQTT (terbitkan)
- Fungsi standar

5.2 *LIBRARY* EKSTERNAL

Bagian pertama dari kode disediakan di Listing 5-1. Ini mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Sketsa ini memiliki dua dependensi utama. Untuk konektivitas Internet, Anda harus menyertakan <WiFi.h> (dengan asumsi Anda menggunakan pelindung WiFi) dan untuk komunikasi broker MQTT, Anda harus menyertakan <PubSubClient.h>.

Listing 5-1. Kode untuk Menyertakan Dependensi Eksternal

```
#include <SPI.h>
#include <WiFi.h>
#include <PubSubClient.h>
```

Konektivitas Internet (Nirkabel)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (dari Bab 2) di sini.

Baca Data Sensor

Bagian ketiga dari kode ditunjukkan pada Listing 5-2. Ini mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membaca data sensor.

Listing 5-2. Variabel untuk Membaca Data Sensor Gerak

```
int calibrationTime = 30;
long unsigned int lowIn;
long unsigned int pause = 5000;
boolean lockLow = true;
boolean takeLowTime;
int pirPin = 3;
```

Listing 5-3 menyediakan kode untuk fungsi calibrateSensor(), yang menunggu sensor gerak untuk mengkalibrasi dengan benar. Sensor dapat memerlukan waktu antara 5 dan 15 detik untuk mengkalibrasi, sehingga kode tersebut memungkinkan sensor untuk mengkalibrasi selama 30 detik. Setelah kalibrasi selesai, sensor gerak aktif dan dapat memulai deteksi. Jika Anda tidak memberikan cukup waktu untuk mengkalibrasi, sensor gerak mungkin mengembalikan pembacaan yang salah.

Listing 5-3. Berfungsi untuk Mengkalibrasi Sensor Gerak

```
{
        pinMode(pirPin, INPUT);
        digitalWrite(pirPin, LOW);
        Serial.println("[INFO] Calibrating Sensor ");
        for(int i = 0; i < calibrationTime; i++)
        {
        Serial.print(".");
        delay(1000);
```

```
        }
        Serial.println("");
        Serial.println("[INFO] Calibration Complete");
        Serial.println("[INFO] Sensor Active");
        delay(50);
    }
```

Fungsi readSensorData() dalam Listing 5-4 membaca data dari Digital Pin 3 dan hasilnya adalah HIGH atau LOW. HIGH berarti gerakan terdeteksi dan LOW berarti tidak ada gerakan atau gerakan berhenti. Kondisi tambahan if(lockLow) ada untuk menghindari penerbitan terlalu banyak pesan ke broker MQTT untuk mosi yang sama.

Listing 5-4. Kode untuk Membaca Data Sensor Gerak

```
    void readSensorData()
    {
       if(digitalRead(pirPin) == HIGH)
       {
            if(lockLow)
            {
                lockLow = false;
                Serial.print("[INFO] Activity Detected @ ");
                Serial.print(millis()/1000);
                Serial.print(" secs");
                Serial.println("");
                // Publish sensor data to MQTT broker
                publishSensorData();

                delay(50);
            }
            takeLowTime = true;
            }
            if(digitalRead(pirPin) == LOW)
            {
            if(takeLowTime)
            {
            lowIn = millis();
            takeLowTime = false;
            }
                        if(!lockLow && millis() - lowIn > pause)
                        {
                        lockLow = true;

                        Serial.print("[INFO] Activity Ended @ ");     //output
                        Serial.print((millis() - pause)/1000);
                        Serial.print(" secs");
                        Serial.println("");
                        delay(50);
              }
    }
    }
```

Data Publish

Bagian keempat dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk mempublishkan data ke broker MQTT. Ini adalah kode yang sama yang Anda lihat di Bab 3. Anda tidak perlu membuat perubahan apa pun agar kode berfungsi, tetapi Anda disarankan untuk menyesuaikan beberapa pesan agar tidak tercampur dengan orang lain yang menggunakan nilai yang sama. Semua nilai yang dapat diubah telah disorot dalam huruf tebal pada Listing 5-5. Jika Anda menggunakan server MQTT Anda sendiri, pastikan untuk mengubah nilai server dan port. Dua perubahan yang disarankan termasuk nilai variabel topik dan nama klien yang harus Anda lewati saat terhubung ke broker MQTT.

Listing 5-5. Kode untuk Menerbitkan Pesan MQTT

```
// IP address of the MQTT broker
char server[] = { "iot.eclipse.org" };
int port =  1883 ;
char topic[] = { " codifythings/intrusiondetection" };
void callback(char* topic, byte* payload, unsigned int length)
{
//Handle message arrived
}
PubSubClient pubSubClient(server, port, 0, client);
void publishSensorData()
{
// Connect MQTT Broker
Serial.println("[INFO] Connecting to MQTT Broker");
if (pubSubClient.connect( "arduinoIoTClient" ))
{
Serial.println("[INFO] Connection to MQTT Broker Successful");
}
else
{
Serial.println("[INFO] Connection to MQTT Broker Failed");
}

// Publish to MQTT Topic
if (pubSubClient.connected())
{
    Serial.println("[INFO] Publishing to MQTT Broker");
    pubSubClient.publish(topic, "Intrusion Detected");
    Serial.println("[INFO] Publish to MQTT Broker Complete");
}
else
{
Serial.println("[ERROR] Publish to MQTT Broker Failed");
}
pubSubClient.disconnect();
}
```

Fungsi Standar

Kode di bagian kelima dan terakhir mengimplementasikan fungsi setup() dan loop() standar Arduino. Dalam fungsi setup(), Anda menginisialisasi port serial, menyambungkan ke

Internet, dan mengkalibrasi sensor untuk pembacaan yang benar, seperti yang ditunjukkan pada Listing 5-6.

Listing 5-6. Kode untuk Fungsi Arduino Standar—setup()

```
void setup()
{
// Initialize serial port
Serial.begin(9600);
// Connect Arduino to internet    connectToInternet();
// Calibrate motion sensor      calibrateSensor();
}
```

In the loop() function, you only need to call the readSensorData() function , as shown in Listing 5-7.

Listing 5-7. K ode untuk Standard Arduino Function—loop()

```
void loop()
{
//Read sensor data
readSensorData();
}
```

Kode Arduino Anda sekarang selesai.

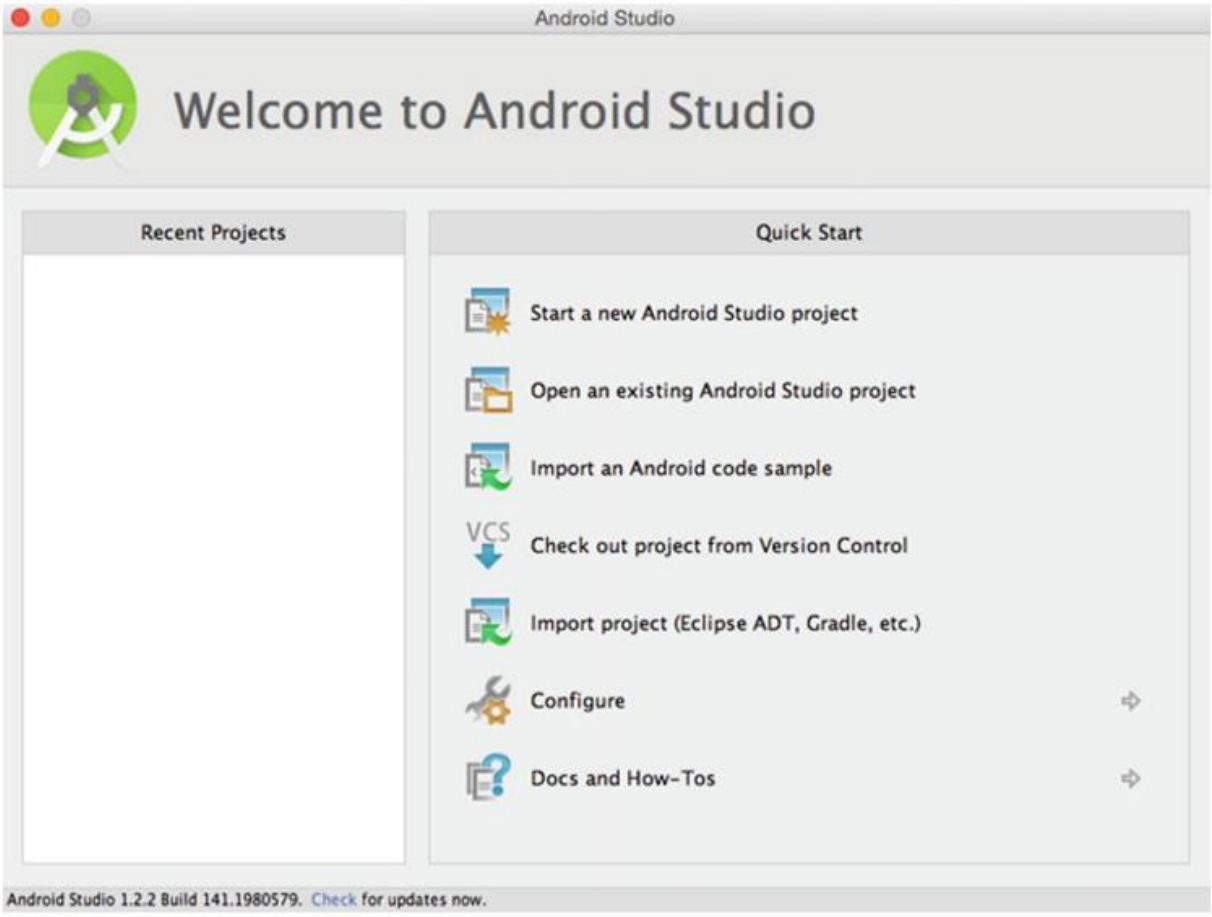
Kode (Android)

Bagian ini memberikan petunjuk untuk mengembangkan aplikasi Android yang akan memenuhi dua persyaratan berikut:

- Tampilkan pemberitahuan secara realtime setiap kali gerakan terdeteksi oleh sensor
- Buat layar sederhana tempat pengguna aplikasi dapat melihat saat gerakan terakhir terdeteksi

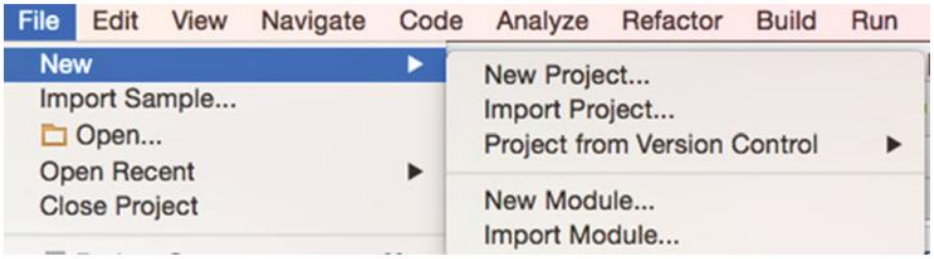
5.3 *PROJECT SETUP*

Di bagian ini, Anda akan membuat proyek baru di Android Studio untuk mengembangkan aplikasi. Android Studio adalah IDE resmi untuk pengembangan platform Android dan dapat diunduh dari http://developer.android.com/sdk/index.html. Mulai Android Studio dan buat proyek Android Studio baru. Jika Anda berada di layar **Quick Start**, seperti yang ditunjukkan pada Gambar 5-5, klik opsi **Start New Project Studio Android** untuk membuat proyek baru.



Gambar 5-5. Buat proyek baru dari layar **Quick Start**

Jika Anda sudah berada di Studio Android, seperti yang ditunjukkan pada Gambar 5-6, pilih pilih **File** lalu **New** lalu klik **New Project** untuk membuat project Android Studio baru.

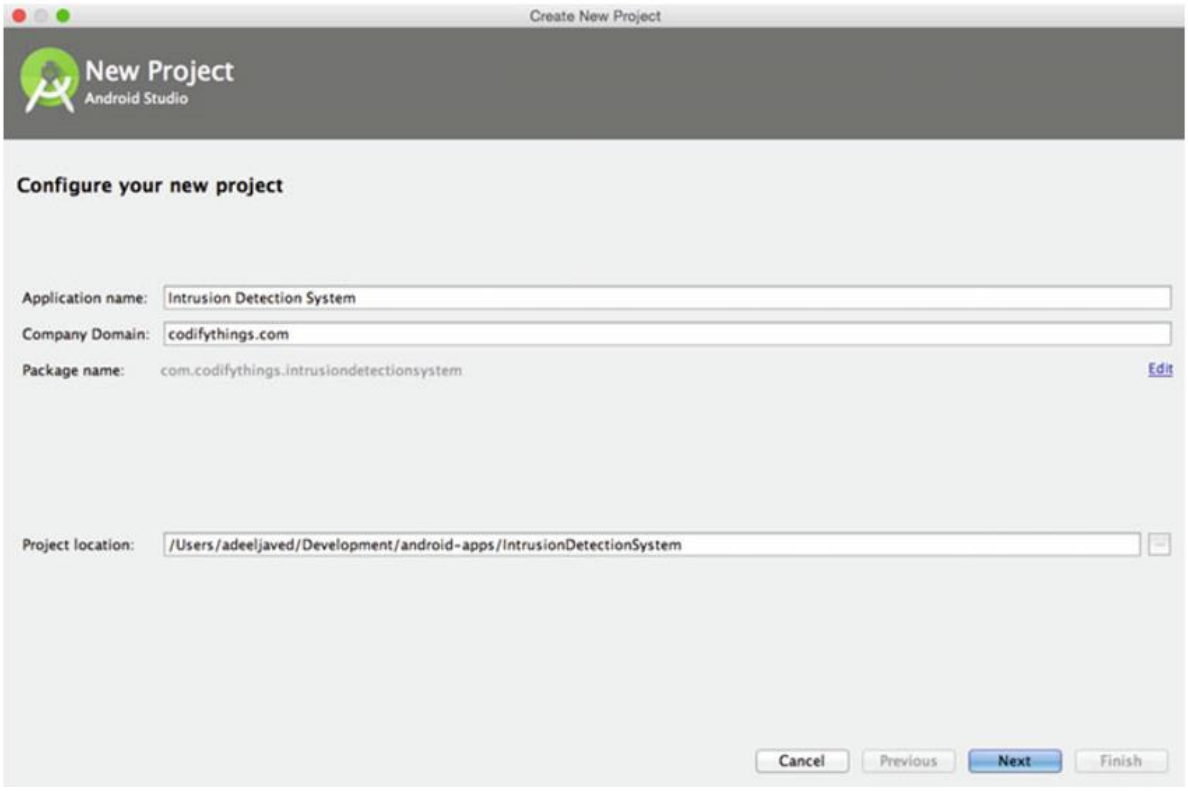


Gambar 5-6. Buat new project dari bilah menu Android Studio

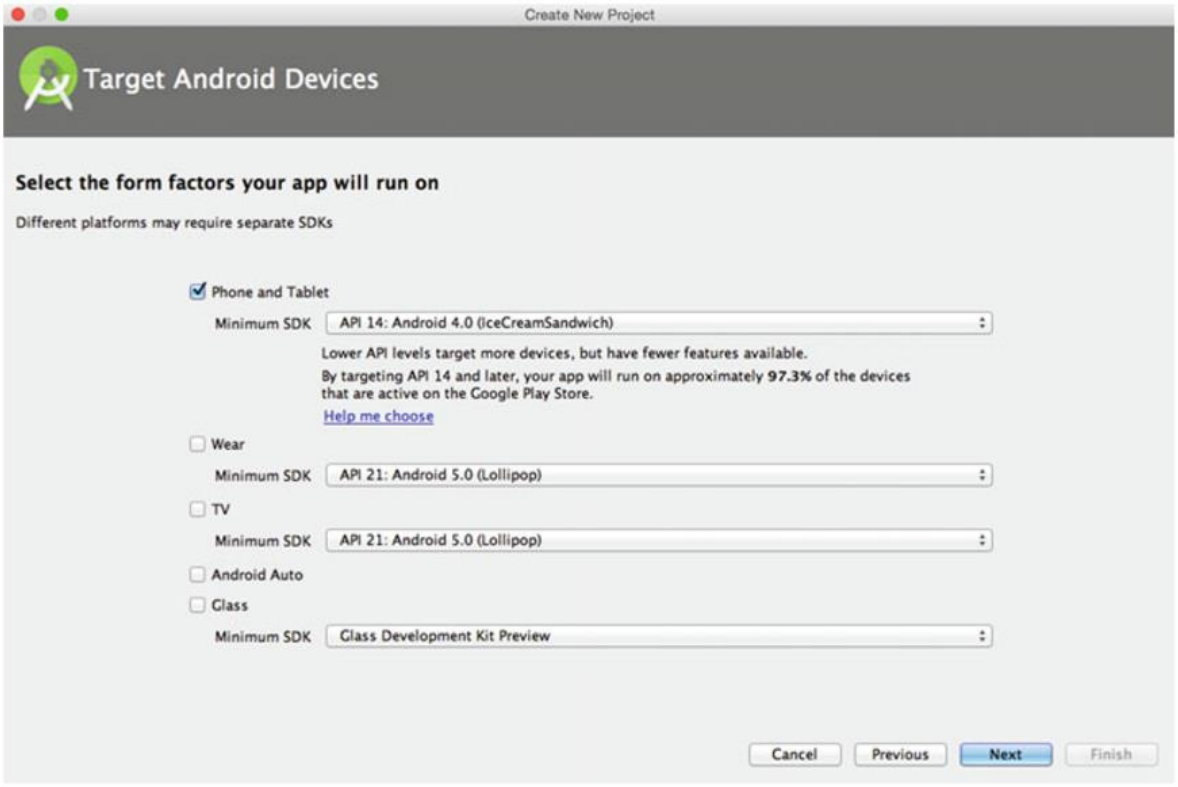
Gambar 5-7 menunjukkan layar konfigurasi proyek baru. Masukkan nama untuk proyek baru, misalnya, Intrusion Detection System. Masukkan nama domain perusahaan atau pribadi Anda. Ini akan digunakan oleh Android Studio untuk mendefinisikan hierarki paket kode Java. **Klik Next**.

***Catatan**: Sebagai norma, hierarki paket adalah nama domain yang terbalik. Oleh karena itu, codifythings.com menjadi com.codifythings.<nama paket>.*

Untuk proyek ini, Anda hanya akan menjalankan aplikasi di ponsel atau tablet Android, jadi pilih Ponsel dan Tablet untuk platform target, seperti yang ditunjukkan pada Gambar 5-8.

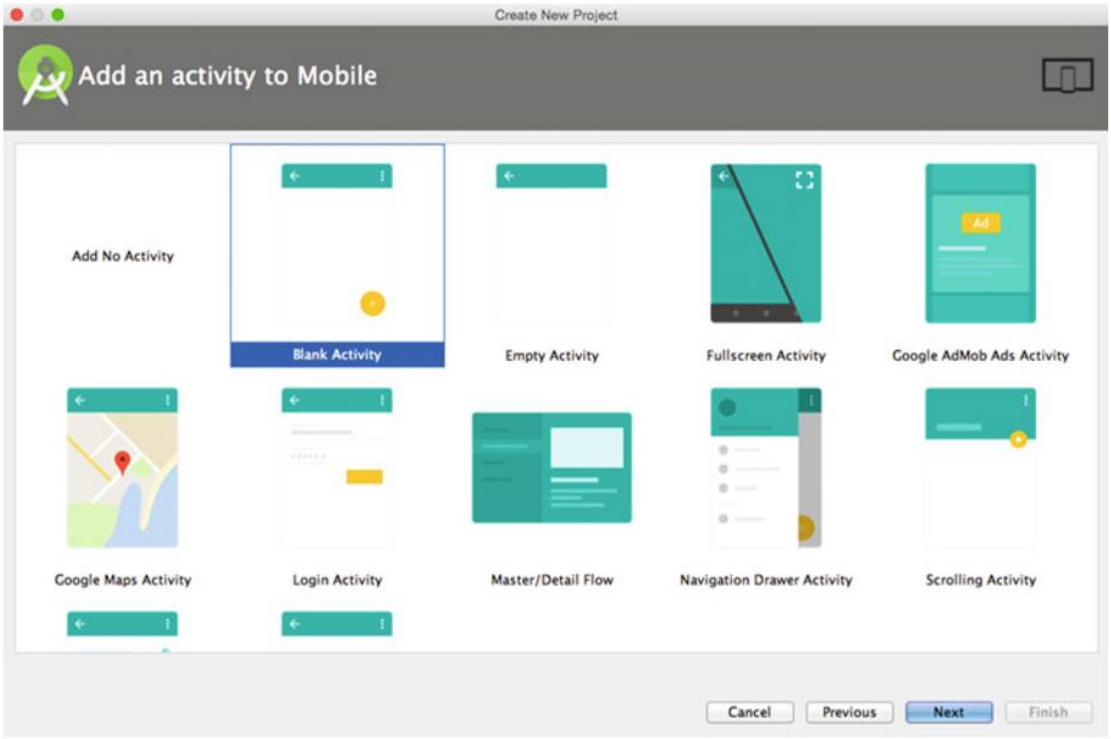


Gambar 5-7. Konfigurasi proyek c baru



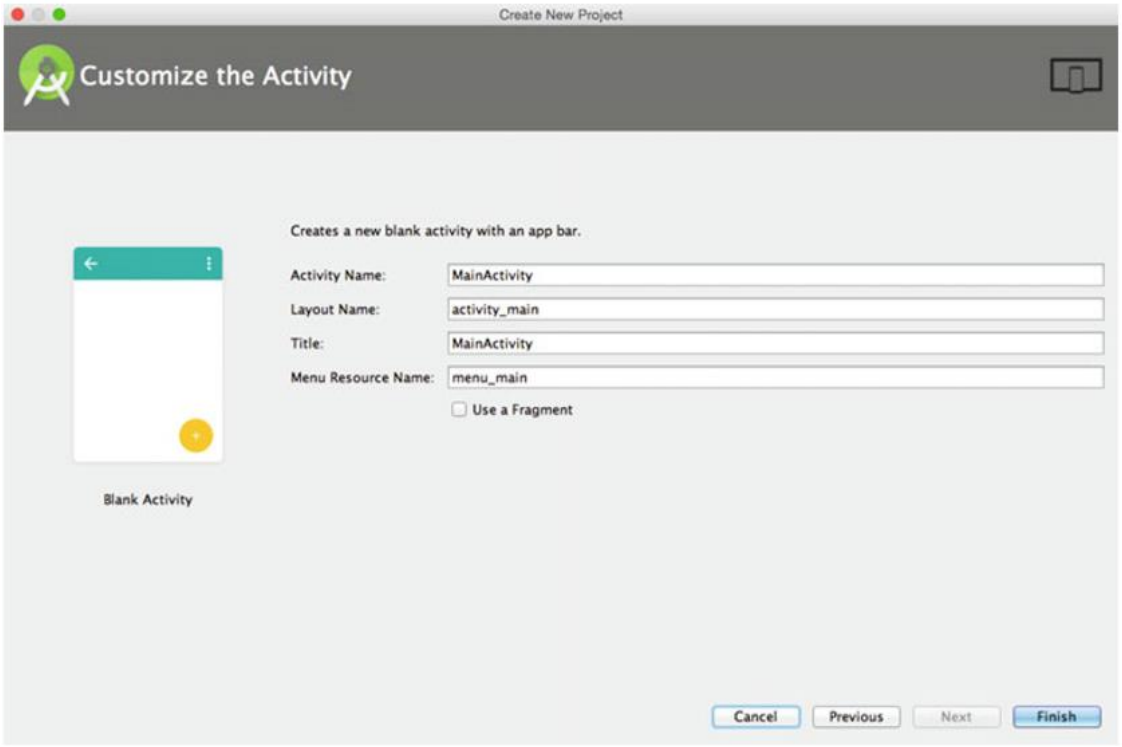
Gambar 5-8. Layar pemilihan perangkat droid

Aplikasi Anda memerlukan layar untuk menampilkan waktu saat intrusi terakhir terdeteksi. Untuk mencapai ini, Anda perlu membuat aktivitas. Dari layar pemilihan templat aktivitas, pilih **Blank Activity**; lihat Gambar 5-9. **Klik Next**.



Gambar 5-9. Layar pemilihan template aktivitas

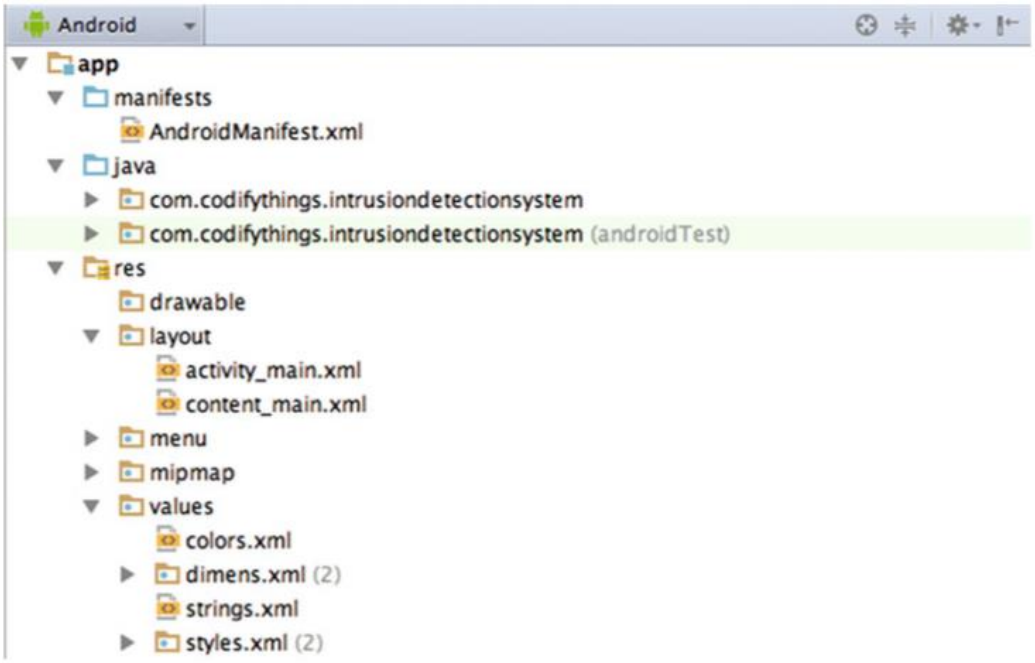
Biarkan nilai **default** untuk **Activity Name, Layout Name, Title**, dan **Menu Resource Name**, seperti yang ditunjukkan pada Gambar 5-10. Sisa bab ini akan merujuk mereka dengan nama yang sama.



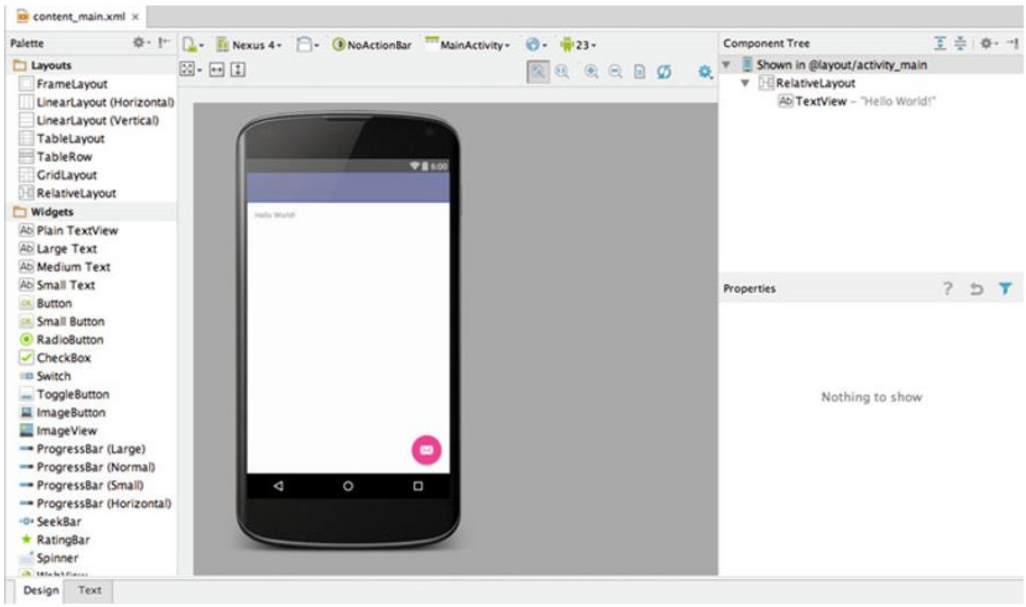
Gambar 5-10. Layar kustomisasi aktivitas

Klik Selesai. Android Studio akan membuat beberapa folder dan file, seperti yang ditunjukkan pada Gambar 5-11. Ini adalah yang paling penting:

- app > manifes > AndroidManifest.xml —File wajib yang diperlukan oleh sistem yang berisi informasi aplikasi, seperti izin yang diperlukan, layar dan layanan, dll. Sebagian besar elemen dalam file ini dibuat oleh sistem, tetapi Anda dapat **mengupdate**nya secara manual sebagai dengan baik.
- app > java > *.* - package-hierarchy —Berisi semua kode Java dan pengujian unit.
- app > res > layout > *.xml —Berisi XML layout untuk semua layar. Menentukan bagaimana setiap layar akan terlihat, font, warna, posisi, dll. Anda dapat mengakses XML tata letak apa pun di Java menggunakan kelas **R** Java yang dibuat secara otomatis, seperti **R.layout.activity_main.** Untuk mengakses elemen individual dalam XML layout, Anda dapat menggunakan sintaks **R.id.updated_field**.



Gambar 5-11. Folder default yang dihasilkan oleh Android Studio



Gambar 5-12. Tampilan pengembangan default Android Studio

Untuk mulai mendesain layout layar, klik **activity_main.xml** di App lalu pilih **Res** lalu klik **folder Layout**. Ini akan membuka layar Aktivitas Utama. Layar default dalam tampilan Desain akan terlihat seperti Gambar 5-12.

Ada dua opsi untuk menyesuaikan Layout Screen. Anda dapat menggunakan fitur drag-dan-lepas dalam tampilan Desain atau mengedit file XML secara manual dalam tampilan Teks. Kita akan langsung mengedit XML dalam tampilan Teks. Beralih ke tampilan Teks dan Anda akan dapat melihat Layout Screen dalam XML, seperti yang ditunjukkan pada Listing 5-8. File tata letak ini bertindak sebagai wadah untuk file sub-tata letak lainnya. Seperti yang Anda lihat di Listing 5-8, content_main disertakan dalam file layout **activity_main.xml.**

Listing 5-8. Tampilan Teks Default dari activity_main.xml

```
< <?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        tools:context="com.codifythings.intrusiondetectionsystem.MainActivity">
<android.support.design.widget.AppBarLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:theme="@style/AppTheme.AppBarOverlay">
                <android.support.v7.widget.Toolbar
                        android:id="@+id/toolbar"
                        android:layout_width="match_parent"
                        android:layout_height="?attr/actionBarSize"
                        android:background="?attr/colorPrimary"
                        app:popupTheme="@style/AppTheme.PopupOverlay" />
    </android.support.design.widget.AppBarLayout>
    <include layout="@layout/content_main" />
    <android.support.design.widget.FloatingActionButton
            android:id="@+id/fab"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom|end"
            android:layout_margin="@dimen/fab_margin"
            android:src="@android:drawable/ic_dialog_email" />
</android.support.design.widget.CoordinatorLayout>
```

File activity_main.xml menambahkan toolbar dan tombol aksi mengambang pada tampilan. Tak satu pun dari widget ini diperlukan dalam aplikasi ini, jadi Anda akan menghapus keduanya. Setelah menghapus toolbar dan tombol aksi mengambang, activitiy_main.xml akan terlihat seperti Listing 5-9.

Listing 5-9. activity_main.xml Tanpa Toolbar dan Tombol Tindakan Mengambang

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://
schemas.android.com/apk/res/android"
```

```
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        tools:context="com.codifythings.intrusiondetectionsystem.MainActivity">
        <include layout="@layout/content_main" />
</android.support.design.widget.CoordinatorLayout>
```

Listing 5-10. Tampilan Teks Default dari content_main.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        ndroid:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        tools:context="com.codifythings.intrusiondetectionsystem.MainActivity"
        tools:showIn="@layout/activity_main">
        <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Hello World!" />
</RelativeLayout>
```

Anda akan mulai dengan terlebih dahulu menghapus elemen **TextView** yang ada untuk **Hello World** yang ditunjukkan pada Listing 5-11.

Listing 5-11. Hapus Elemen Default dari content_main.xml

```
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
```

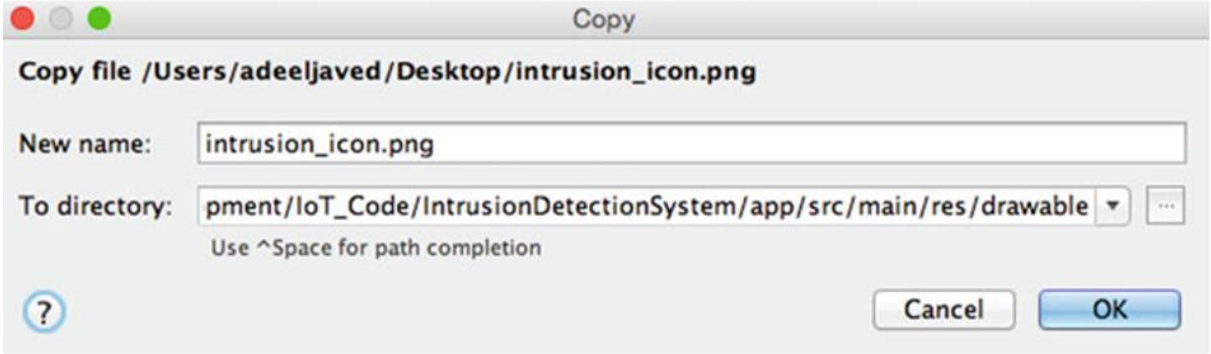
Selanjutnya, tambahkan elemen ImageView yang disediakan di Listing 5-12 ke content_main.xml. Ini akan menampilkan gambar penyusup.

Listing 5-12. Tambahkan elemen ImageView ke content_main.xml

```
<ImageView
        android:id="@+id/intrusion_icon"
        android:src="@drawable/intrusion_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        />
```

Elemen tersebut mereferensikan gambar yang disebut intrusion_icon, jadi Anda perlu menempelkan gambar bernama intrusion_icon. **png** ke App lalu pilih **Res** lalu klik Drawable

folder, seperti terlihat pada Gambar 5-13. Anda dapat mengUpload gambar Anda sendiri atau mengunduh yang ada di contoh dari https://openclipart.org/detail/212125/walking.



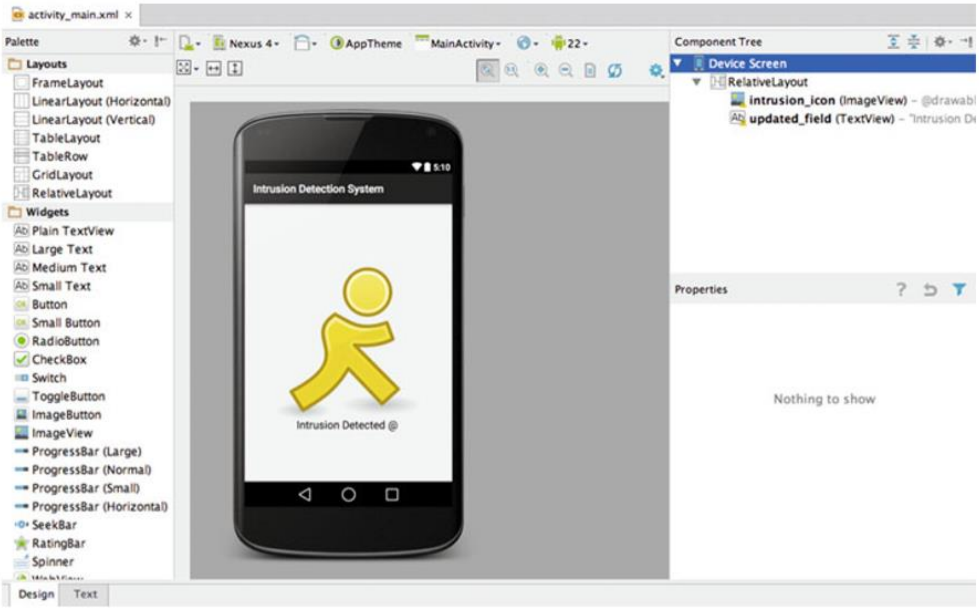
Gambar 5-13. Kotak dialog untuk menambahkan gambar ke aplikasi

Seperti yang disediakan di Listing 5-13, tambahkan elemen kedua TextView. Ini akan menampilkan waktu saat gerakan terakhir terdeteksi.

Listing 5-13. Tambahkan Elemen TextView ke content_main.xml

```
<TextView
        android:id="@+id/updated_field"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/intrusion_icon"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textSize="20sp"
        android:textColor="#000000"
        android:text="Intrusion Detected @ "
        />
```

Layout Screen aplikasi Anda sudah siap, dan akan terlihat seperti Gambar 5-14.



Gambar 5-14. Layout Screen akhir aplikasi

5.4 LOGIKA SCREEN

Selanjutnya Anda akan menambahkan logika ke layar yang akan membuatnya dinamis dan membuat pemberitahuan ketika pesan baru diterima dari sensor. Buka file **MainActivity.java** dari **App** lalu pilih **Java** lalu klik paket **com.codifythingsintrusiondetectionsystem**. Secara default, akan ada tiga metode yang dibuat secara otomatis oleh Android Studio, seperti yang ditunjukkan pada Listing 5-14.

Listing 5-14. Kode default untuk MainActivity.java

```
public class MainActivity extends AppCompatActivity {
{
        @Override
        protected void onCreate(Bundle savedInstanceState) {... }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {... }
        @Override
        public boolean onOptionsItemSelected(MenuItem item) {... }
}
```

Untuk saat ini Anda akan menambahkan dua metode baru. Metode pertama disediakan dalam Listing 5-15. Ini updateView(...) dan akan **mengupdate** layar dengan pesan baru dari sensor.

Listing 5-15. Tambahkan Metode updateView(...) ke MainActivity.java

```
public void updateView(String sensorMessage) {
try {
                SharedPreferences sharedPref = getSharedPreferences(
                "com.codifythings.motionsensorapp.PREFERENCE_FILE_KEY",
                Context.MODE_PRIVATE);
if (sensorMessage == null || sensorMessage == "") {
        sensorMessage = sharedPref.getString("lastSensorMessage",
                        "No Activity Detected");
            }
            final String tempSensorMessage = sensorMessage;
            runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            TextView updatedField = (TextView)
                               findViewById(R.id.updated_field);
                            updatedField.setText(tempSensorMessage;
                        }
                    });
              SharedPreferences.Editor editor = sharedPref.edit();
              editor.putString("lastSensorMessage", sensorMessage);
              editor.commit();
           } catch (Exception ex) {
           Log.e(TAG, ex.getMessage());
      }
}
```

Metode kedua disediakan dalam Listing 5-16. Ini adalah createNotification(...) dan akan membuat notifikasi realtime di ponsel atau tablet untuk mengingatkan pengguna.

Listing 5-16. Tambahkan Metode createNotification(...) ke MainActivity.java

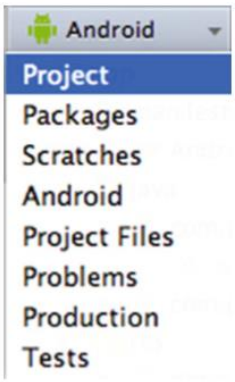
```
public void createNotification(String.notificationTitle)String (notificationMessage) {
NotificationCompat.Builder mBuilder =
        New NotificationCompat.Builder(getApplicationContext())
            .setSmallIcon(R.drawable.notification_template_icon_bg)
            .setContentTitle(notificationTitle)          .setContentText(notificationMe
    ssage);
// Creates an explicit intent for an Activity in your app
Intent resultIntent = new Intent(getApplicationContext(),
                MainActivity.class);
// The stack builder object will contain an artificial back
// stack for the started Activity. This ensures that navigating
// backward from the Activity leads out of your application to the
// Home screen.
TaskStackBuilder stackBuilder =
TaskStackBuilder.create(getApplicationContext());

// Adds the back stack for the Intent (but not the Intent itself)
stackBuilder.addParentStack(MainActivity.class);
// Adds the Intent that starts the Activity to the top of the stack
stackBuilder.addNextIntent(resultIntent);

PendingIntent resultPendingIntent =
stackBuilder.getPendingIntent(0,
        PendingIntent.FLAG_UPDATE_CURRENT);
mBuilder.setContentIntent(resultPendingIntent);
NotificationManager mNotificationManager = NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
// mId allows you to update the notification later on.
mNotificationManager.notify(100, mBuilder.build());
}
```

5.5 MQTT CLIENT

Bagian terakhir dari aplikasi Anda adalah klien MQTT. Ini akan terhubung ke server MQTT dan subscribe topik codifythings/intrusiondetection. Untuk berkomunikasi dengan broker MQTT, aplikasi Anda memerlukan library MQTT. Oleh karena itu, unduh dua library berikut:

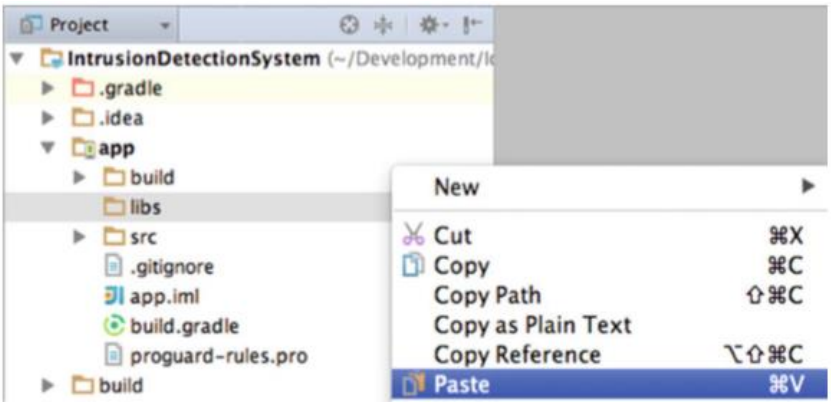
- Library klien MQTT: https://Eclipse.org/paho/clients/Java/
- Library layanan Android: https://Eclipse.org/paho/clients/android/

Setelah Anda mengunduh kedua file JAR, alihkan tampilan Android Studio dari Android ke Project, seperti yang ditunjukkan pada Gambar 5-15.



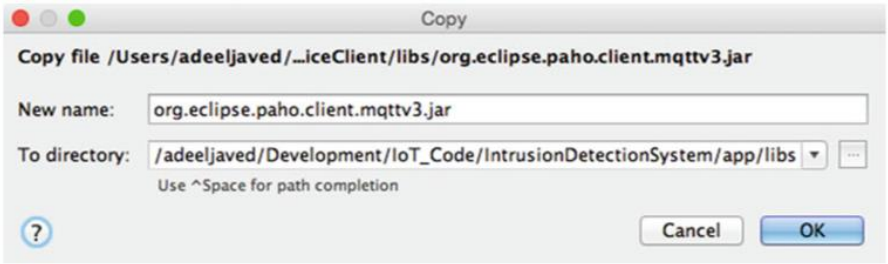
Gambar 5-15. Pindahkan perspektif dari Android ke proyek

Perluas IntrusionDetectionSystem pilih App dan rekatkan kedua library ke dalam folder libs. Gambar 5-16 menunjukkan folder libs tempat kedua library perlu ditempelkan.



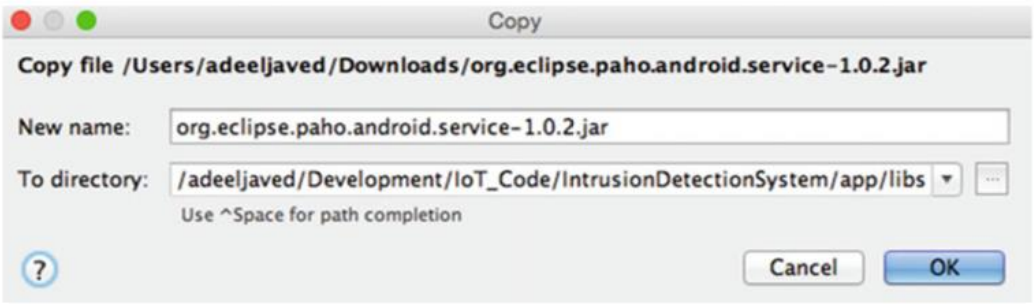
Gambar 5-16. Impor library untuk menyelesaikan dependensi

Gambar 5-17 menunjukkan kotak dialog yang akan ditampilkan saat Anda menempelkan library MQTT



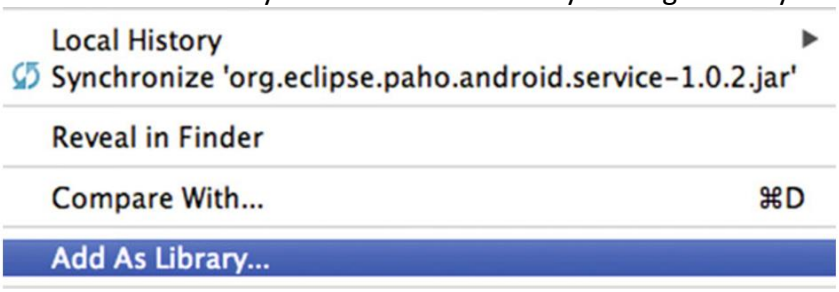
Gambar 5-17. Impor library MQTT

Gambar 5-18 menunjukkan kotak dialog yang akan ditampilkan saat Anda menempelkan library Layanan Android.



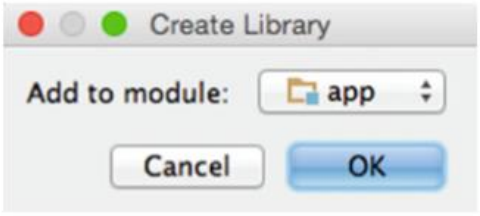
Gambar 5-18. Impor library Layanan Android

Seperti yang ditunjukkan pada Gambar 5-19, klik kanan pada library yang baru ditambahkan dan klik opsi Add As Library. Anda dapat melakukan ini untuk kedua library satu per satu atau memilih keduanya lalu menambahkannya sebagai library.



Gambar 5-19. Tambahkan file yang diimpor sebagai library

Seperti yang ditunjukkan pada Gambar 5-20, pilih Aplikasi dari opsi **Add to Module**. Klik **OK** dan beralih kembali ke tampilan Android.

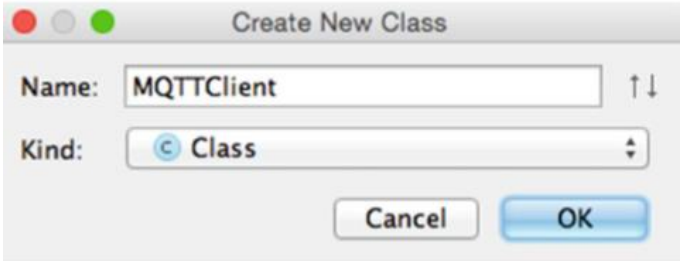


Gambar 5-20. Tambahkan library ke modul aplikasi

Selanjutnya Anda akan menulis kode untuk berkomunikasi dengan broker MQTT. Seperti yang ditunjukkan pada Gambar 5-21, klik kanan pada paket tingkat atas (dalam contoh ini adalah com.codifythings. intrusiondetectionsystem) dan pilih **New** pilih **Java Class.**

Gambar 5-21. Tambahkan kelas baru

Masukkan MQTTClient di **Name Field** dan klik **OK**, seperti yang ditunjukkan pada Gambar 5-22.



Gambar 5-22. Masukkan nama kelas baru

Android Studio akan menghasilkan file kelas kosong dengan kode default, seperti yang ditunjukkan pada Listing 5-17.

Listing 5-17. Kode Default untuk MQTTClient.java

```
public class MQTTClient
{
...
}
```

Selanjutnya Anda akan menambahkan kode ke MQTTClient yang akan terhubung dan subscribe ke broker MQTT, dan setiap kali pesan baru diterima tentang topik langganan, kode akan **mengupdate** antarmuka pengguna aplikasi. Listing 5-18 menyediakan kode lengkap untuk MQTTClient.

Listing 5-18. Kode Lengkap MQTTClient.java

```
package com.codifythings.intrusiondetectionsystem;
import android.util.Log;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;
import java.text.DateFormat;
import java.util.Date;
public class MQTTClient {
        private static final String TAG = "MQTTClient";
        private String mqttBroker = "tcp://iot.eclipse.org:1883";
        private String mqttTopic = "codifythings/intrusiondetection";
        private String deviceId = "androidClient";
        // Variables to store reference to the user interface activity.
        private MainActivity activity = null;
        public MQTTClient(MainActivity activity) {
        this.activity = activity;
        }

public void connectToMQTT() throws MqttException {
// Request clean session in the connection options.
        Log.i(TAG, "Setting Connection Options");
        MqttConnectOptions options = new MqttConnectOptions();
        options.setCleanSession(true);
        //Attempt a connection to MQTT broker using the values of
        // connection variables.
        Log.i(TAG, "Creating New Client");
        MqttClientclient=new MqttClient(mqttBroker, deviceId, new
                                                MemoryPersistence());
        client.connect(options);
        // Set callback method name that will be invoked when a new message
        // is posted to topic, MqttEventCallback class is defined later in
        // the code.
        Log.i(TAG, "Subscribing to Topic");
        client.setCallback(new MqttEventCallback());
```

```
        // Subscribe to topic "codifythings/intrusiondetection", whenever a
        // new message is published to this topic
        // MqttEventCallback.messageArrived will be called.
        client.subscribe(mqttTopic, 0);
    }
    // Implementation of the MqttCallback.messageArrived method, which is
    // invoked whenever a new message is published to the topic
    // "codifythings/intrusiondetection".
    private class MqttEventCallback implements MqttCallback {
    @ Override
    public void connectionLost(Throwable arg0) {
    // Do nothing
    }
    @Override
    public void deliveryComplete(IMqttDeliveryToken arg0) {
    // Do nothing
    }
    @Override
    public void messageArrived(String topic, final MqttMessage msg)
    throws Exception {
    Log.i(TAG, "New Message Arrived from Topic - " + topic);
    try {
    // Append the payload message "Intrusion Detected"
    // with "@ Current Time".
    DateFormat df = DateFormat.getDateTimeInstance();
    String sensorMessage = new String(msg.getPayload()) + " @ "+ df.format(new Date());
    // User is not going to be on the screen all the time,
    // so create a notification.
    activity.createNotification("Intrusion Detection System sensorMessage);
                    // Update the screen with newly received message.
                    activity.updateView(sensorMessage);
                    } catch (Exception ex) {
                    Log.e(TAG, ex.getMessage());
        }
    }
    }
```

Di Listing 5-18, variabel TAG akan digunakan saat masuk sehingga Anda dapat mengidentifikasi pesan aplikasi Anda di log. Variabel mqttBroker, mqttTopic, dan deviceId menentukan broker MQTT yang akan terhubung dengan aplikasi Anda, topik langganan aplikasi Anda, dan ID perangkat yang akan muncul di server saat aplikasi Anda berhasil terhubung. Variabel aktivitas didefinisikan untuk menyimpan referensi aktivitas antarmuka pengguna sehingga Anda dapat langsung melakukan pembaruan.

Kode untuk menghubungkan dan subscribe ke broker MQTT menggunakan metode connectToMQTT(). Inisialisasi MqttClient baru dan hubungkan ke file iot. Eclipse.org:1883 server dengan sesi bersih. Anda perlu mengeksekusi kode Anda setiap kali pesan baru dipublishkan ke antrean codifythings/intrusiondetection, jadi pertama-tama atur metode callback dengan menyediakan instance baru MqttEventCallback dan kemudian subscribe topik codifythings/intrusiondetection.

Setelah Anda subscribe topik dan menyetel metode panggilan balik, library MQTT akan selalu memanggil metode MqttCallback.messageArrived Anda. Jadi sekarang Anda perlu menyediakan implementasi yang menentukan apa yang harus dilakukan ketika pesan baru telah tiba.

Aplikasi Anda memiliki dua persyaratan. Itu perlu membuat pemberitahuan baru untuk pengguna dan **mengupdate** layar dengan waktu terakhir aktivitas terdeteksi. Anda telah mengimplementasikan dua metode ini di **MainActivity class**, jadi Anda akan menggunakan referensi aktivitas dan memanggil metode createNotification dan updateView. Layar dan klien MQTT sekarang sudah siap, tetapi Anda belum menambahkan kode di **MainActivity class** yang benar-benar memulai MQTTClient setiap kali aplikasi dibuat. Jadi Update metode onCreate() dari **MainActivity class** untuk **mengupdate** layar dengan string kosong dan mulai MQTTClient. Karena Anda menghapus Toolbar dan tombol tindakan mengambang dari activity_main.xml, Anda juga perlu menghapus referensi dalam metode onCreate. Kode terakhir untuk MainActivity disediakan di Listing 5-19, dengan perubahan pada metode onCreate() disorot.

Listing 5-19. Kode Lengkap MainActivity.java

```
package com.codifythings.intrusiondetectionsystem;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.TaskStackBuilder;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v4.app.NotificationCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
private static final String TAG = "MainActivity";
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
        updateView("");
        try
        {
        MQTTClient client = new MQTTClient(this);
        client.connectToMQTT();
        }
        catch(Exception ex)
        {
        Log.e(TAG, ex.getMessage());
        }
}
@Override
```

```
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it
    // is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
    return true;
    }
return super.onOptionsItemSelected(item);
}
//Custom function that renders view     public void
updateView(String sensorMessage) {
try {
SharedPreferences sharedPref = getSharedPreferences(
"com.codifythings.motionsensorapp.PREFERENCE_FILE_KEY",
Context.MODE_PRIVATE);
if (sensorMessage == null || sensorMessage == "") {
sensorMessage = sharedPref.getString("lastSensorMessage",
"No Activity Detected");          }
final String tempSensorMessage = sensorMessage;
runOnUiThread(new Runnable() {
@Override
public void run() {
TextView updatedField = (TextView)
findViewById(R.id.updated_field);
updatedField.setText(tempSensorMessage);
}
});
        SharedPreferences.Editor editor = sharedPref.edit();
        editor.putString("lastSensorMessage", sensorMessage);
        editor.commit();
        } catch (Exception ex) {
        Log.e(TAG, ex.getMessage());
}
}
public void createNotification(String notificationTitle,
String notificationMessage) {
NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(getApplicationContext())
```

```
.setSmallIcon(R.drawable.notification_template_icon_bg)
                    .setContentTitle(notificationTitle)
                    .setContentText(notificationMessage);
// Creates an explicit intent for an Activity in your app
Intent resultIntent = new Intent(getApplicationContext(),
MainActivity.class);
// The stack builder object will contain an artificial back
// stack for the started Activity. This ensures that navigating
// backward from the Activity leads out of your application to the
// Home screen.
TaskStackBuilder stackBuilder =
TaskStackBuilder.create(getApplicationContext());
// Adds the back stack for the Intent (but not the Intent itself)
stackBuilder.addParentStack(MainActivity.class);
 // Adds the Intent that starts the Activity to the top of the stack
      stackBuilder.addNextIntent(resultIntent);
      PendingIntent resultPendingIntent =
      stackBuilder.getPendingIntent(0,
      PendingIntent.FLAG_UPDATE_CURRENT);
      mBuilder.setContentIntent(resultPendingIntent);
      NotificationManager mNotificationManager= (NotificationManager)
      getSystemService(Context.NOTIFICATION_SERVICE);
      // mId allows you to update the notification later on.
      mNotificationManager.notify(100, mBuilder.build());
     }
}
```

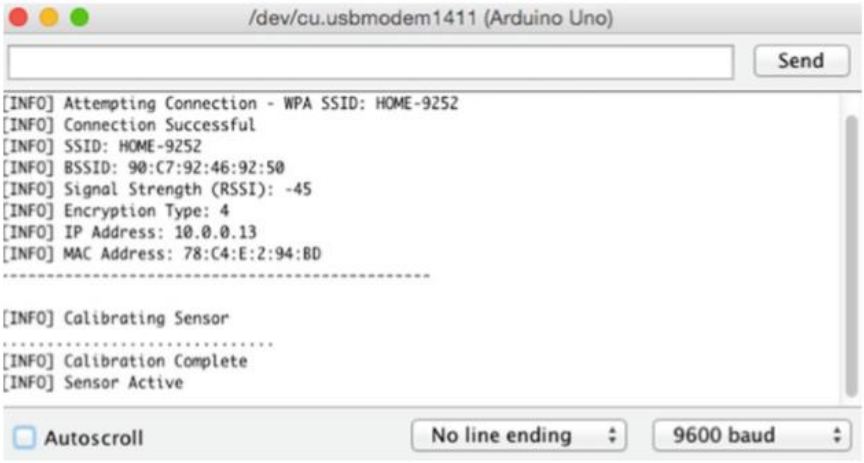
Terakhir, Anda perlu **mengupdate** AndroidManifest.xml di bawah folder App Manifests. Aplikasi Anda menggunakan MqttService di backend, jadi Anda perlu menambahkan referensi ke layanan. Aplikasi Anda juga perlu mengakses Internet untuk menghubungkan ke broker MQTT, jadi tambahkan juga izin Internet di AndroidManifest.xml. Listing 5-20 menyediakan kode yang perlu ditambahkan ke AndroidManifest.xml.

Listing 5-20. Tambahkan Izin Aplikasi di AndroidManifest.

```
<!-- MQTT Service -->
<service android:name="org.eclipse.paho.android.service.MqttService" >
</service>
<uses-permission android:name="android.permission.INTERNET" />
```

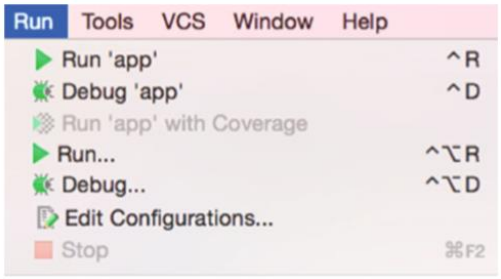
5.6 PRODUK AKHIR

Untuk menguji aplikasi, verifikasi dan Upload kode Arduino, seperti yang dibahas dalam Bab 1. Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log yang mirip dengan yang ditunjukkan pada Gambar 5-23.



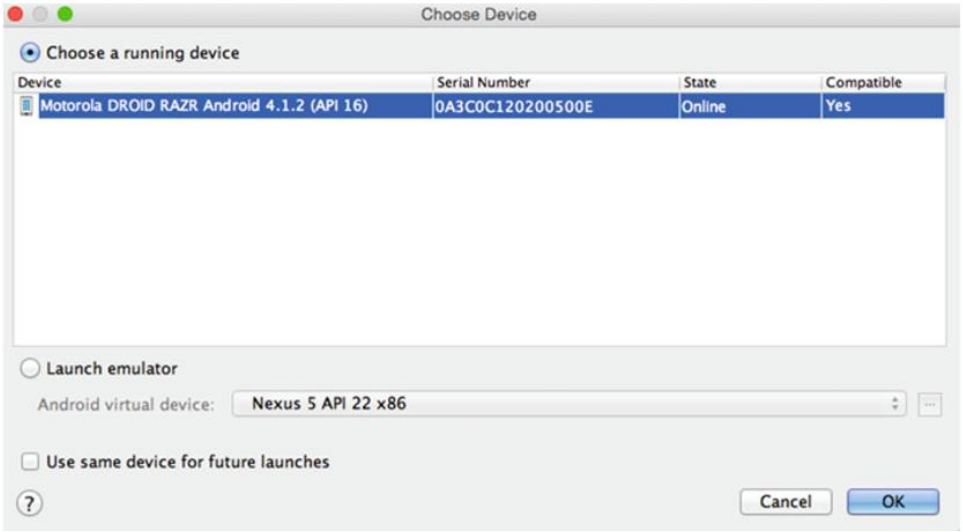
Gambar 5-23. Pesan log dari sistem deteksi intrusi

Di Android Studio, terapkan dan jalankan aplikasi di perangkat Android Anda dengan memilih Jalankan pilih Jalankan 'Aplikasi' dari bilah menu, seperti yang ditunjukkan pada Gambar 5-24.



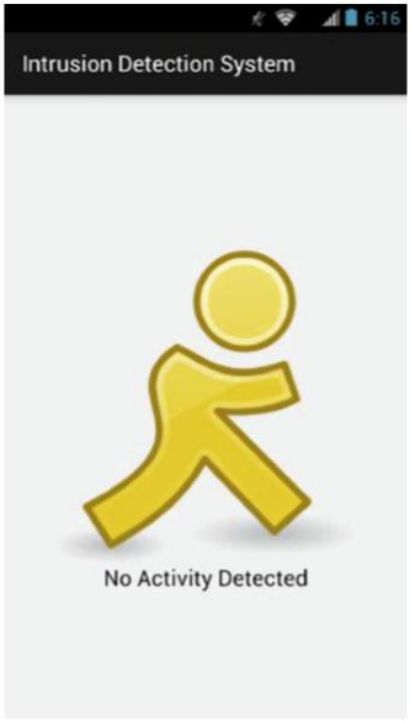
Gambar 5-24. Terapkan dan jalankan aplikasi dari Android Studio

Jika Anda memiliki perangkat Android yang terhubung ke komputer, Android Studio akan meminta Anda untuk menggunakan perangkat berjalan yang ada atau meluncurkan emulator baru untuk menjalankan aplikasi. Seperti yang ditunjukkan pada Gambar 5-25, pilih emulator atau perangkat tempat Anda ingin menguji aplikasi dan klik **OK**.



Gambar 5-25. Pilih perangkat untuk menyebarkan dan menjalankan aplikasi

Buka perangkat tempat aplikasi Anda di-deploy. Jika aplikasi Anda belum berjalan, cari aplikasi Anda di perangkat dan jalankan. Gambar 5-26 menunjukkan tampilan default aplikasi Anda.



Gambar 5-26. Tampilan default aplikasi Android

Buat beberapa gerakan di depan sensor gerak Anda. Segera setelah sensor mendeteksi gerakan, sebuah pesan akan dipublishkan ke broker MQTT dan aplikasi Anda akan menampilkan notifikasi, seperti yang ditunjukkan pada Gambar 5-27.

Gambar 5-27. Pemberitahuan intrusi dari aplikasi Android

Klik pada notifikasi untuk membuka layar aplikasi. Ini akan menampilkan terakhir kali intrusi terdeteksi, seperti yang ditunjukkan pada Gambar 5-28.

Gambar 5-28. Detail intrusi di aplikasi Android

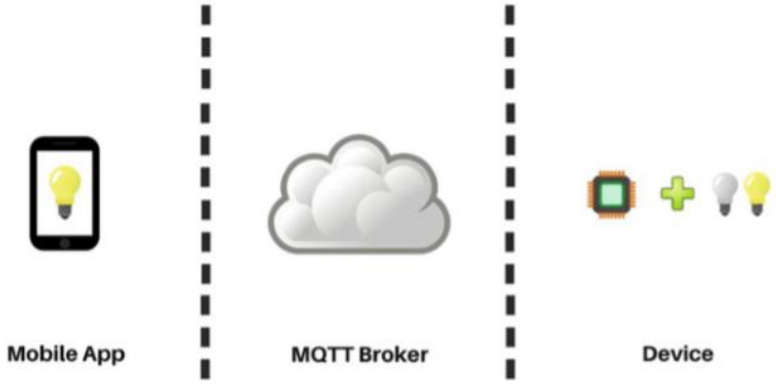
5.7 RINGKASAN

Dalam bab ini, Anda mempelajari tentang klien waktu nyata, pola aplikasi IoT yang sangat penting. Anda mengembangkan sistem deteksi intrusi dengan aplikasi Android sebagai klien untuk mengilustrasikan pola ini. Aplikasi Android hanyalah salah satu contoh dan klien dapat terdiri dari berbagai jenis, termasuk iOS, perangkat yang dapat dikenakan, aplikasi berbasis web, dll.

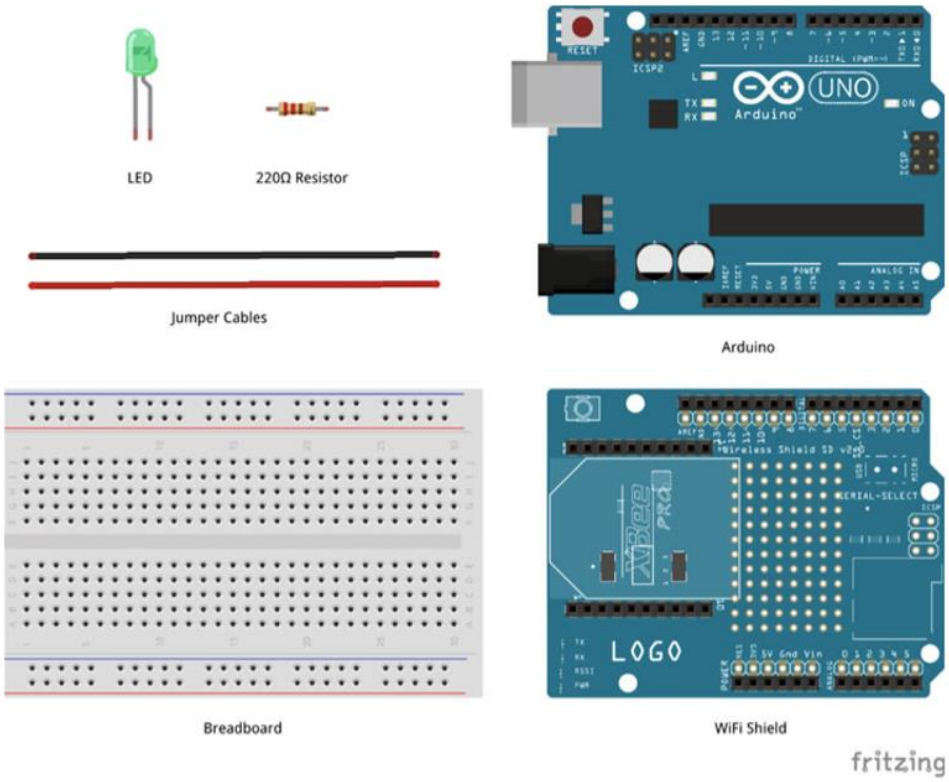
BAB 6
POLA IOT : REMOT KONTROL

Remote control saat ini adalah salah satu pola IoT paling populer. Contoh pola ini dapat ditemukan di aplikasi IoT yang bagiannya dapat Anda kontrol seperti seperti lampu, termostat, dan pintu garasi dari jarak jauh menggunakan perangkat genggam atau komputer. Ini terutama telah digunakan untuk aplikasi otomatisasi rumah sejauh ini.

Dalam bab ini, Anda akan membangun sistem kontrol pencahayaan. Gambar 6-1 menunjukkan komponen sistem kontrol pencahayaan. Komponen pertama adalah aplikasi Android yang memungkinkan pengguna mengontrol lampu. Ini menerbitkan pesan ke broker MQTT setiap kali pengguna mengetuk layar aplikasi. Komponen kedua adalah broker MQTT, dan komponen terakhir dari aplikasi IoT ini adalah perangkat Arduino yang menyalakan atau mematikan lampu berdasarkan pesan yang diterima dari broker MQTT.



Gambar 6-1. Komponen sistem kontrol pencahayaan



Gambar 6-2. Hardware diperlukan untuk sistem kontrol pencahayaan ini

6.1 TUJUAN PEMBELAJARAN

Di akhir bab ini, Anda akan dapat:

- Menulis kode untuk menghidupkan atau mematikan LED yang terhubung ke Arduino
- Subscribe Arduino ke broker MQTT
- Membangun aplikasi Android yang dipublishkan ke broker MQTT

Hardware yang Diperlukan

Gambar 6-2 memberikan Listing semua komponen hardware yang diperlukan untuk membuat sistem kontrol pencahayaan ini.

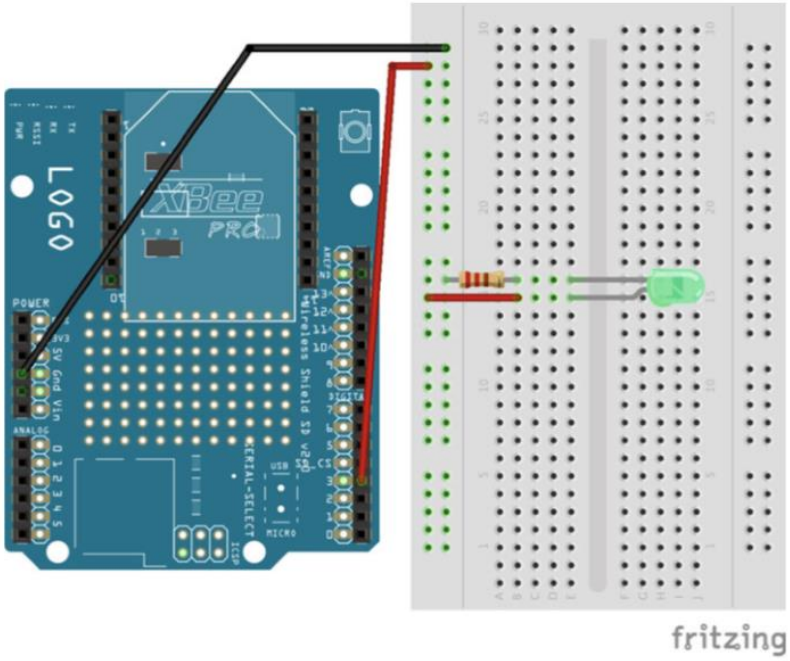
Software yang Diperlukan

Untuk mengembangkan sistem kontrol pencahayaan ini, Anda memerlukan software berikut:

- Arduino IDE 1.6.4 atau lebih baru
- Android Studio 1.5.1 atau lebih baru

Sirkuit

Di bagian ini, Anda akan membangun sirkuit yang diperlukan untuk sistem kontrol pencahayaan.

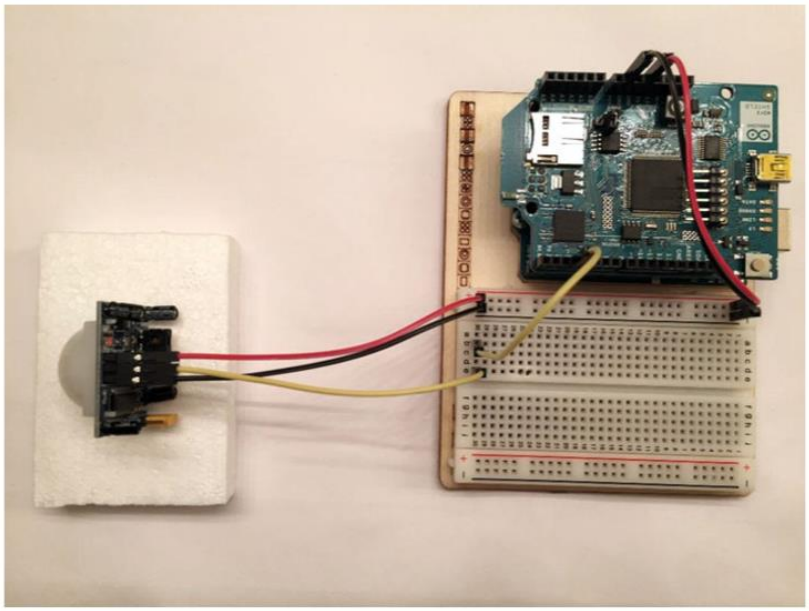


Gambar 6-3. Diagram sirkuit sistem kontrol pencahayaan

1. Pastikan Arduino Anda tidak terhubung ke sumber listrik, seperti ke komputer melalui USB atau baterai.
2. Pasang pelindung WiFi ke bagian atas Arduino. Semua pin harus sejajar.
3. Tidak seperti sirkuit sebelumnya, Anda tidak ingin memberi daya pada papan breadboard sepanjang waktu, tetapi Anda ingin mengontrolnya. Jadi gunakan kabel jumper untuk menghubungkan port digital 3 Arduino Anda ke port power (+) di papan breadboard. Anda akan menggunakan port ini untuk menyalakan dan mematikan LED.
4. Gunakan kabel jumper untuk menghubungkan port ground (GND) pada Arduino ke port ground (-) pada breadboard.
5. Pasang LED ke papan breadboard Anda.
6. Gunakan kabel jumper untuk menyambungkan port daya (+) papan breadboard ke port daya (+) LED.

7. Pasang resistor 220Ω antara port ground (-) pada papan breadboard dan port ground (-) LED.

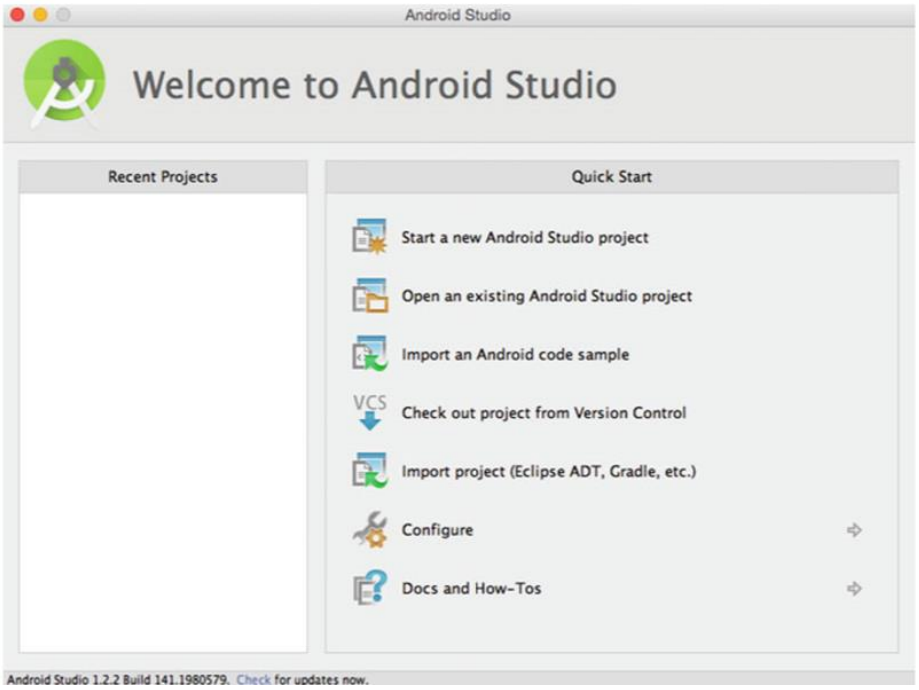
Sirkuit Anda sekarang telah selesai dan akan terlihat mirip dengan Gambar 6-3 dan 6-4.



Gambar 6-4. Sirkuit sebenarnya dari sistem kontrol pencahayaan

Kode (Android)

Bagian ini memberikan petunjuk untuk mengembangkan aplikasi Android yang memungkinkan pengguna mengetuk layar untuk menyalakan dan mematikan lampu.



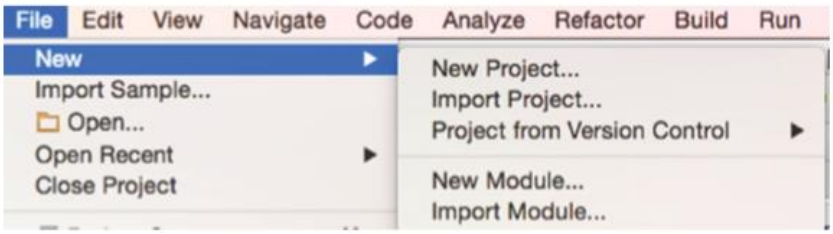
Gambar 6-5. Buat proyek baru dari layar **Quick Start**

6.2 PENGATURAN PROYEK

Di bagian ini, Anda akan membuat proyek baru di Android Studio untuk mengembangkan aplikasi. Mulai Android Studio dan buat proyek Android Studio baru. Jika

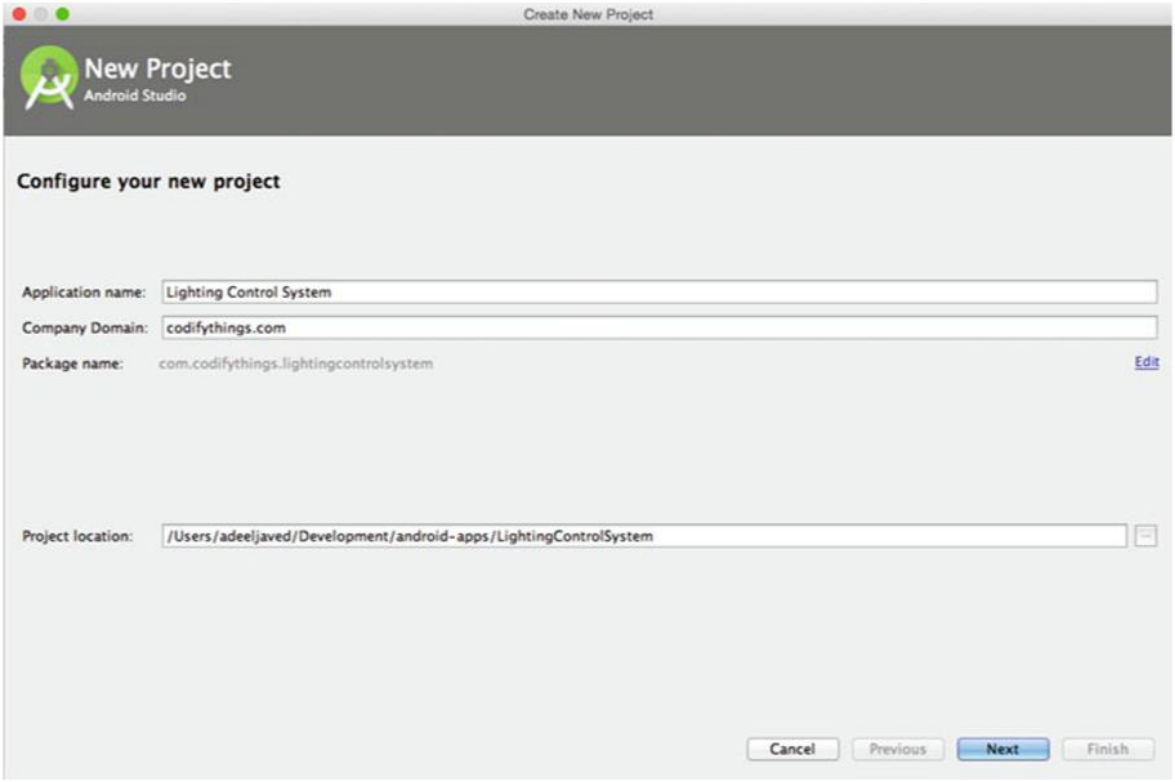
Anda berada di layar **Quick Start**, seperti yang ditunjukkan pada Gambar 6-5, lalu klik **Start New Project Studio Android** untuk membuat proyek baru.

Jika Anda sudah berada di Android Studio, seperti yang ditunjukkan pada Gambar 6-6, pilih File lalu klik New lalu pilih New Project untuk membuat proyek Android Studio baru.



Gambar 6-6. Buat proyek baru dari bilah menu Android Studio

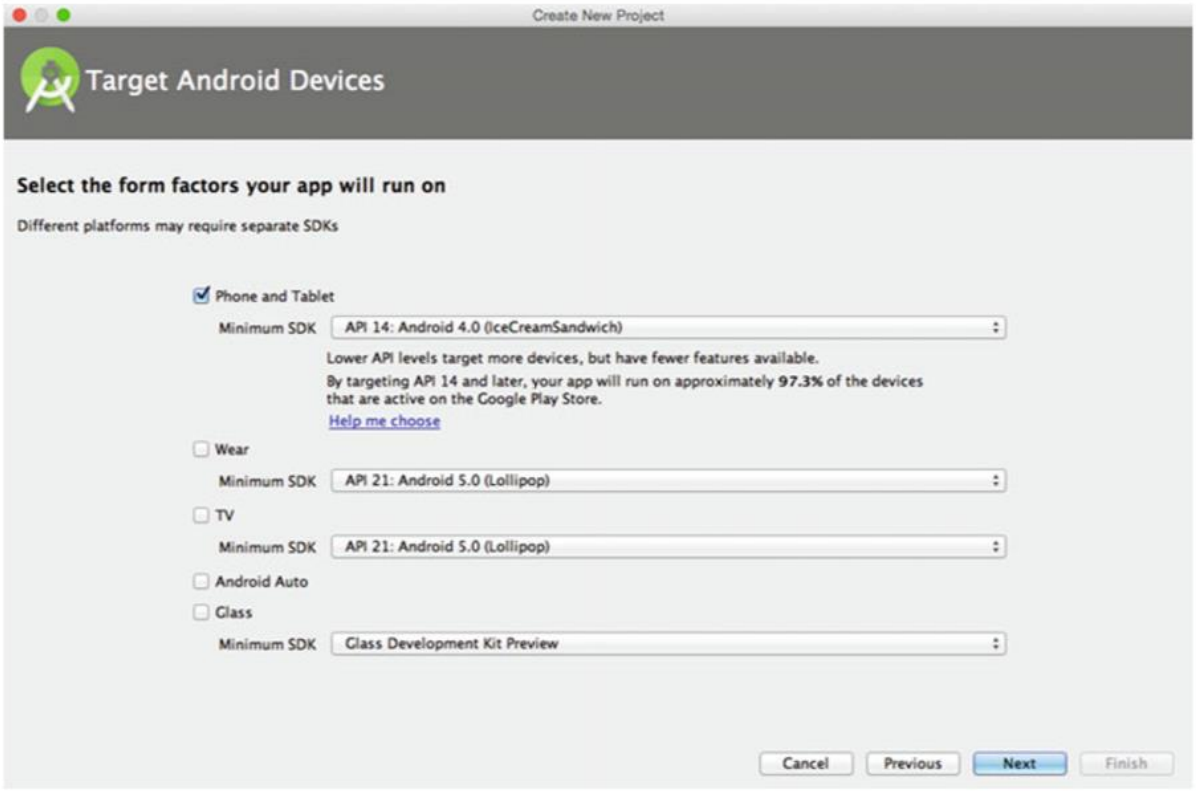
Gambar 6-7 menunjukkan layar konfigurasi proyek baru. Masukkan nama untuk proyek baru sebagai Sistem Kontrol Pencahayaan. Masukkan nama domain perusahaan atau pribadi Anda, karena ini akan digunakan oleh Android Studio untuk menentukan hierarki paket kode Java. **Klik Next**.



Gambar 6-7. Konfigurasi proyek p baru

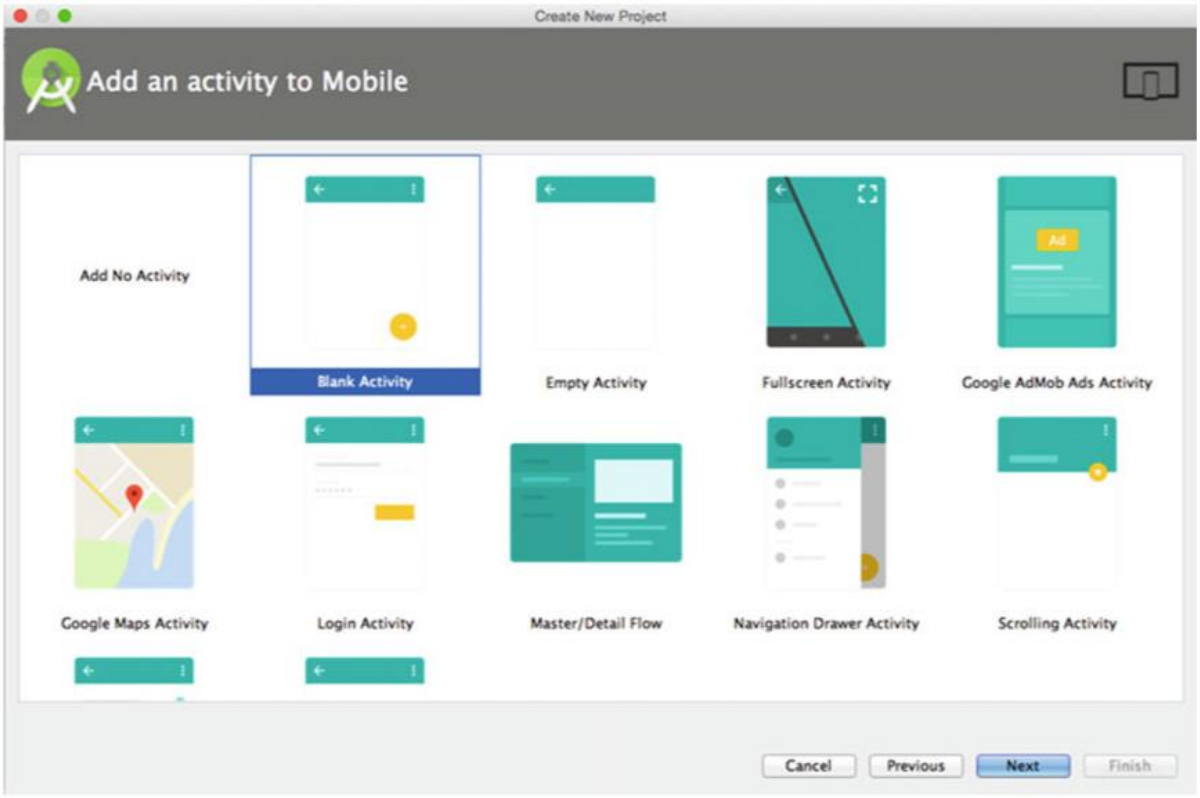
Catatan *: Sebagai norma, hierarki paket adalah nama domain yang terbalik, jadi codifythings.com menjadi com.codifythings.<packagename>.*

Untuk proyek ini, Anda hanya akan menjalankan aplikasi Anda di ponsel atau tablet Android. Seperti yang ditunjukkan pada Gambar 6-8, centang Ponsel dan Tablet sebagai platform target dan klik **Next**.



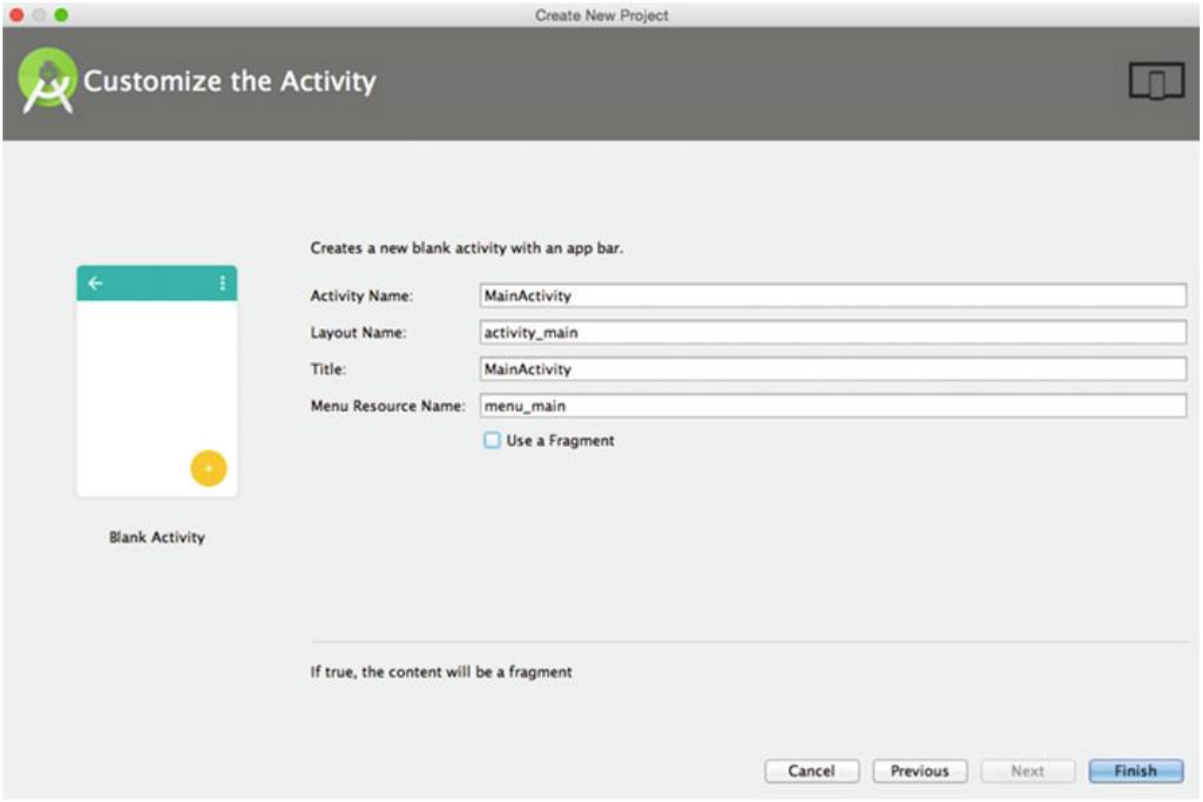
Gambar 6-8. Layar pemilihan perangkat droid

Aplikasi Anda memerlukan layar tempat pengguna dapat mengetuk untuk menyalakan atau mematikan lampu. Untuk mencapai ini, Anda perlu membuat aktivitas. Jadi, dari layar pemilihan Template Aktivitas, pilih **Blank Activity**, seperti yang ditunjukkan pada Gambar 6-9. Klik **Next**.



Gambar 6-9. Layar pemilihan contoh aktivitas

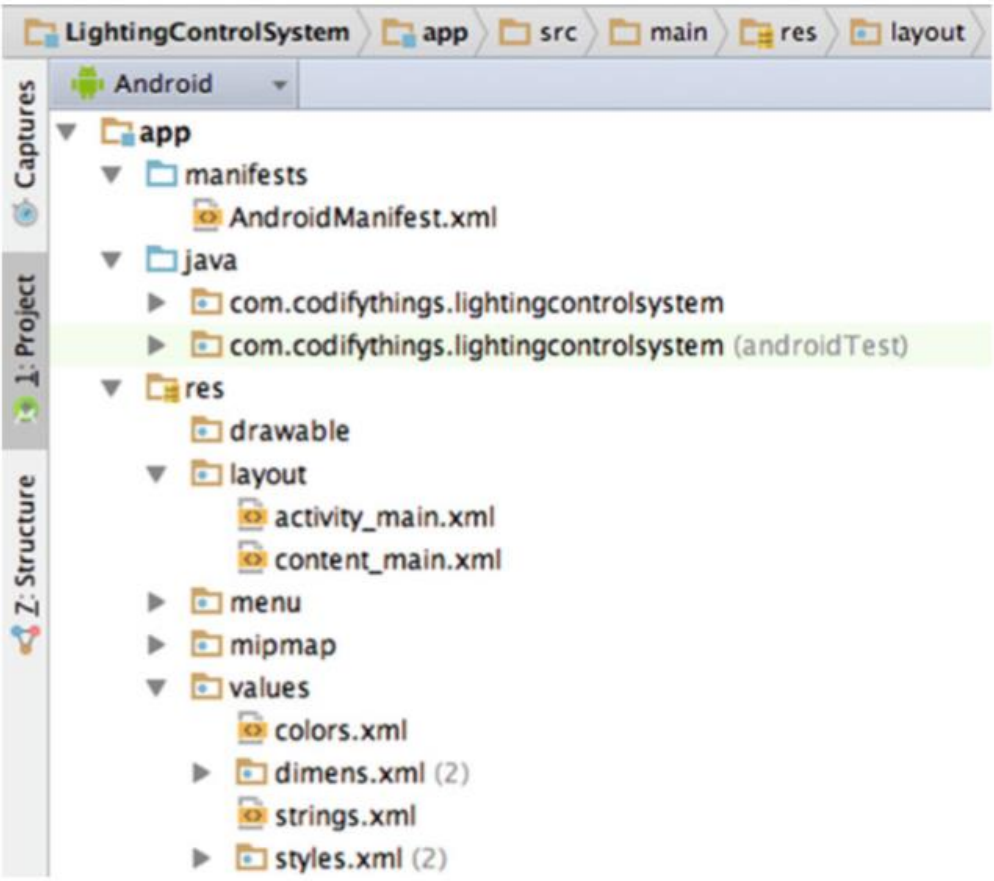
Biarkan nilai default untuk **Activity Name, Layout Name, Title**, dan **Menu Resource Name,** seperti yang ditunjukkan pada Gambar 6-10. Sisa bab ini merujuk mereka dengan nama yang sama.



Gambar 6-10. Layar kustomisasi aktivitas

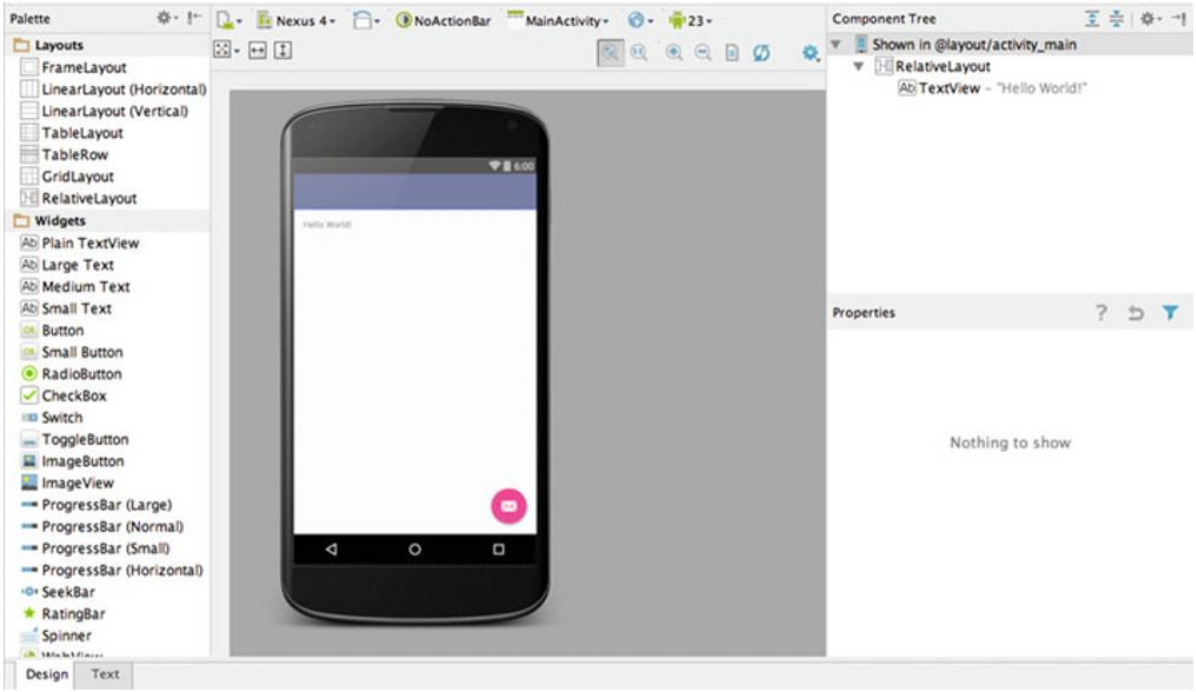
Klik **Finish**. Android Studio akan membuat beberapa folder dan file, seperti yang ditunjukkan pada Gambar 6-11. Ini adalah yang paling penting:

- app > manifests > AndroidManifest.xml : File wajib yang diperlukan oleh sistem yang berisi informasi aplikasi seperti izin yang diperlukan, layar dan layanan, dll. Sebagian besar elemen dalam file ini dibuat oleh sistem, tetapi Anda juga dapat **mengupdate**nya secara manual.
- app > java > *.* - package-hierarchy : Folder ini berisi semua kode Java dan pengujian unit.
- app > res > layout > *.xml : Folder ini berisi XML layout untuk semua layar, termasuk tampilan setiap layar, font, warna, posisi, dll. Anda dapat mengakses XML tata letak apa pun di Java menggunakan kelas Java yang dibuat secara otomatis., seperti R.layout.activity_main. Untuk mengakses elemen individual dalam XML layout, Anda dapat menggunakan sintaks R.id.updated_field.



Gambar 6-11. Folder default yang dihasilkan oleh Android Studio

Layout Screen Untuk mulai merancang Layout Screen, klik file activity_main.xml di **App** pilih folder **Res** lalu **Layout**, yang akan membuka layar Aktivitas Utama. Layar default dalam tampilan Desain akan terlihat seperti Gambar 6-12.



Gambar 6-12. Tampilan pengembangan default dari Android S tudio

Ada dua opsi untuk menyesuaikan Layout Screen—Anda dapat menggunakan fitur drag dan lepas dalam tampilan Desain atau mengedit file XML secara manual dalam tampilan Teks. Kami akan langsung mengedit XML dalam tampilan Teks. Beralih dari tampilan Desain ke tampilan Teks dan Anda akan dapat melihat Layout Screen dalam XML, seperti yang ditunjukkan pada Listing 6-1. File layout ini bertindak sebagai wadah untuk file sublayout lainnya. Seperti yang Anda lihat di Listing 6-1, content_main disertakan dalam file layout activity_main.xml.

Listing 6-1. Tampilan Teks Default dari activity_main.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://
schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        tools:context="com.codifythings.lightingcontrolsystem.MainActivity">
        <android.support.design.widget.AppBarLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:theme="@style/AppTheme.AppBarOverlay">
        <android.support.v7.widget.Toolbar
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                android:background="?attr/colorPrimary"
                app:popupTheme="@style/AppTheme.PopupOverlay" />
</android.support.design.widget.AppBarLayout>
<include layout="@layout/content_main" />
<android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@android:drawable/ic_dialog_email" />
</android.support.design.widget.CoordinatorLayout>
```

File activity_main.xml menambahkan toolbar dan tombol aksi mengambang pada tampilan. Tak satu pun dari widget ini diperlukan dalam aplikasi ini, sehingga Anda dapat menghapus keduanya. Setelah menghapus toolbar dan tombol aksi mengambang, activitiy_main.xml akan terlihat mirip dengan Listing 6-2.

Listing 6-2. activity_main.xml Tanpa Toolbar dan Tombol Tindakan Mengambang

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://
schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fitsSystemWindows="true"
        tools:context="com.codifythings.lightingcontrolsystem.MainActivity">
        <include layout="@layout/content_main" />
</android.support.design.widget.CoordinatorLayout>
```

Disarankan untuk menambahkan konten khusus di file content_main.xml. Listing 6-3 menunjukkan kode default content_main.xml.

Listing 6-3. Tampilan Teks Default dari content_main

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        tools:context="com.codifythings.lightingcontrolsystem.MainActivity"
        tools:showIn="@layout/activity_main">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello World!" />
</RelativeLayout>
```

Anda dapat memulai dengan terlebih dahulu menghapus elemen TextView yang ada untuk Hello World yang ditunjukkan pada Listing 6-4.

Listing 6-4. Hapus Elemen Default dari content_main.xml

```
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
```

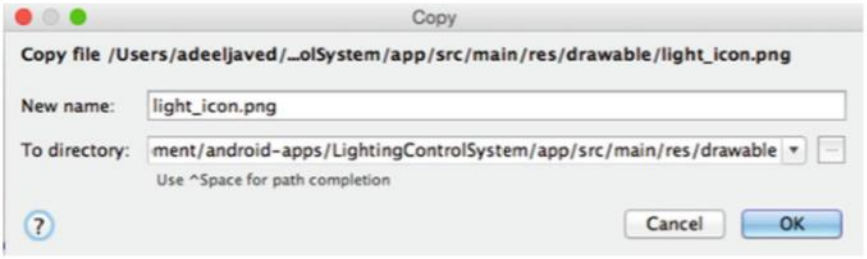
Selanjutnya, tambahkan elemen ImageView yang disediakan di Listing 6-5 ke content_main.xml ; ini akan menampilkan gambar bola lampu.

Listing 6-5. Tambahkan Elemen ImageView ke content_main.xml

```
<ImageView
        android:id="@+id/light_icon"
        android:src="@drawable/light_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        />
```

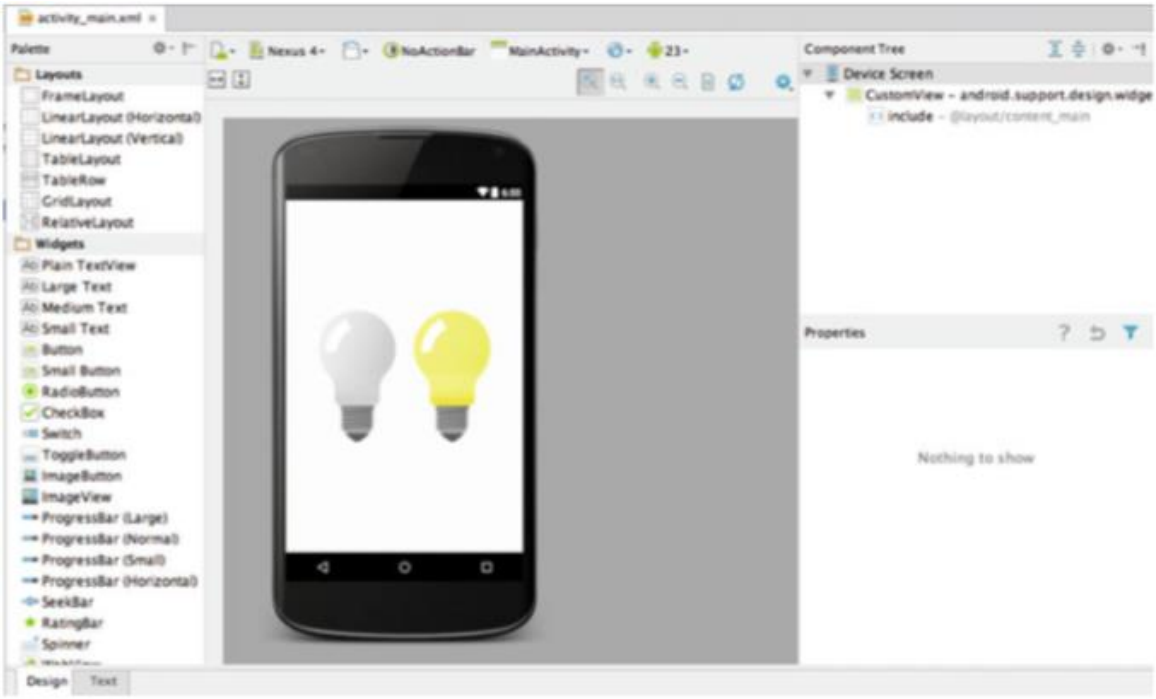
Elemen mereferensikan gambar yang disebut light_icon, jadi Anda perlu menyediakan gambar bernama light_icon. png di App lalu pilih Res pilih Drawable folder, seperti yang

ditunjukkan pada Gambar 6-13. Anda dapat mengUpload gambar Anda atau mengunduh yang sama yang telah digunakan dalam contoh dari https://openclipart.org/detail/220988/light-bulb-on-off.



Gambar 6-13. Kotak dialog untuk menambahkan gambar ke aplikasi

Layout Screen aplikasi Anda sudah siap, dan akan terlihat seperti Gambar 6-14.



Gambar 6-14. Layout Screen akhir aplikasi

6.3 LOGIKA LAYAR

Selanjutnya Anda akan membuat layar menjadi interaktif sehingga pengguna aplikasi dapat mengetuk ikon bola lampu untuk menyalakan atau mematikan lampu. Aplikasi ini tidak menampilkan jika lampu sedang hidup atau mati; sebagai gantinya, itu hanya mengubah status dari hidup ke mati dan dari mati ke hidup. Buka **file MainActivity.java** dari Aplikasi **Java com.codifythings**. paket sistem kontrol pencahayaan. Secara default, akan ada tiga metode yang dibuat secara otomatis oleh Android Studio seperti yang ditunjukkan pada Listing 6-6.

Listing 6-6. Kode Default untuk MainActivity.java

```
public class MainActivity extends ActionBarActivity  {
@Override
protected void onCreate(Bundle savedInstanceState) {... }
@Override
public boolean onCreateOptionsMenu(Menu menu) {... }
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {... }
}
```

Karena Anda menghapus toolbar dan tombol tindakan mengambang dari activity_main.xml, Anda juga perlu menghapus referensi dalam metode onCreate. Anda ingin ikon bola lampu menjadi interaktif sehingga ketika pengguna aplikasi mengetuk ikon, sebuah pesan dipublishkan ke broker MQTT. Untuk mencapai ini, Anda perlu **mengupdate** metode onCreate(), seperti yang ditunjukkan pada Listing 6-7. Anda akan menListingkan pendengar onClick() yang akan dipanggil setiap kali seseorang mengetuk ikon bola lampu. Untuk saat ini implementasi onClick() masih kosong dan akan diupdate nanti.

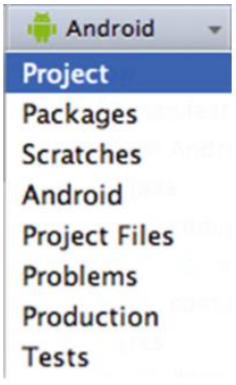
Listing 6-7. Ketuk Layar/Klik Kode Pendengar

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ImageView lightIcon = (ImageView) findViewById(R.id.light_icon);
        lightIcon.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
            //TODO - add action
            }
});
}
```

6.4 MQTT Client

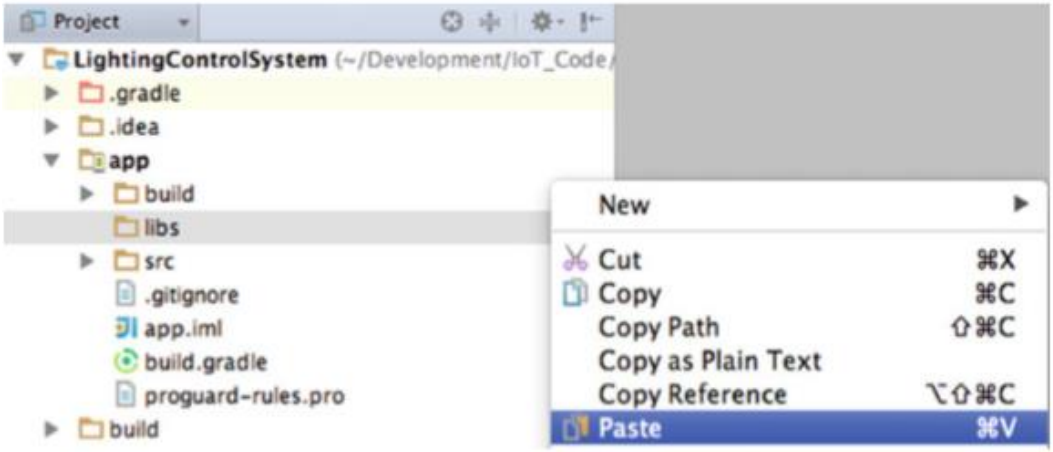
Bagian terakhir dari aplikasi ini adalah klien MQTT yang akan terhubung ke server MQTT dan mempublishkan ke topik codifythings/lightcontrol. Untuk berkomunikasi dengan broker MQTT, aplikasi Anda memerlukan library MQTT yang dapat diunduh dari https://Eclipse.org/paho/clients/java/.

Setelah Anda mendownload library, alihkan tampilan Android Studio dari Android ke Project, seperti yang ditunjukkan pada Gambar 6-15.



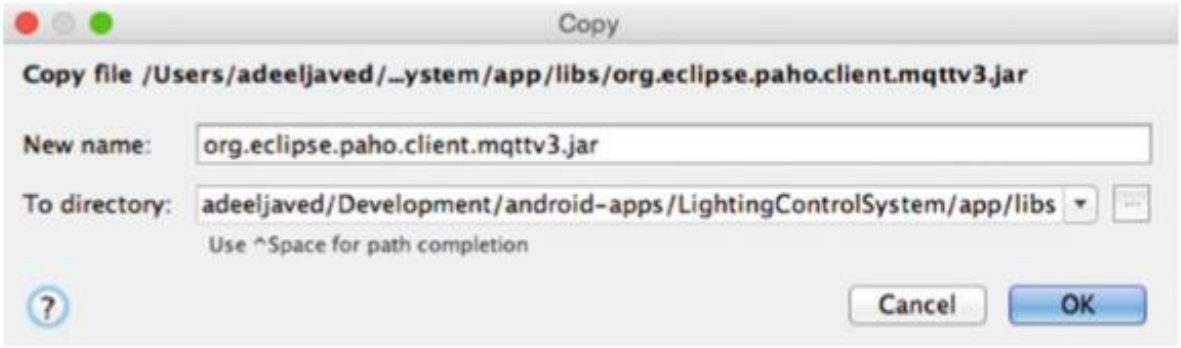
Gambar 6-15. S beralih perspektif dari Android ke Proyek

Perluas **LightingControlSystem I**alu pilih App dan rekatkan library MQTT di folder libs. Gambar 6-16 menunjukkan folder libs tempat semua library harus ditempelkan.



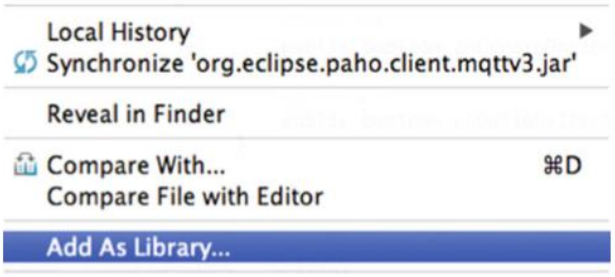
Gambar 6-16. Impor library untuk menyelesaikan dependensi

Gambar 6-17 menunjukkan kotak dialog yang akan ditampilkan saat Anda menempelkan library MQTT l.



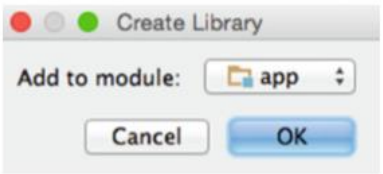
Gambar 6-17. Impor library MQTT

Seperti yang ditunjukkan pada Gambar 6-18, klik kanan pada library yang baru ditambahkan dan klik opsi **Add as Library**.



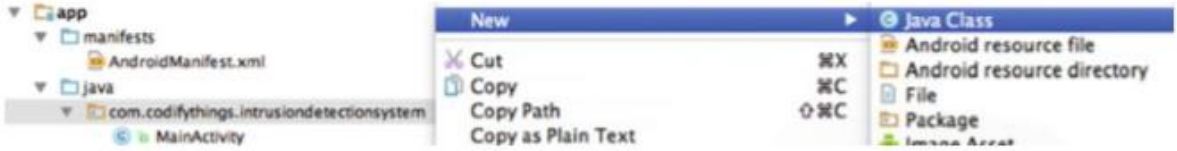
Gambar 6-18. Tambahkan file yang diimpor sebagai library

Seperti yang ditunjukkan pada Gambar 6-19, pilih Aplikasi dari opsi **Add to Module**. Klik OK dan beralih kembali ke tampilan Android.



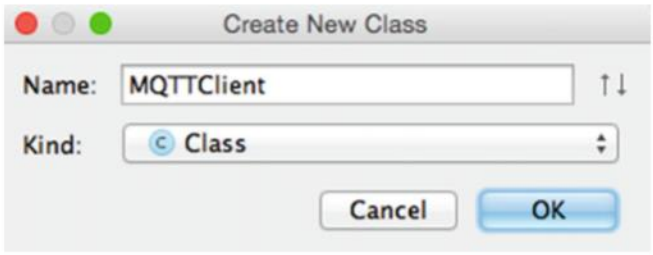
Gambar 6-19. Tambahkan library ke modul aplikasi

Selanjutnya Anda akan menulis kode untuk berkomunikasi dengan broker MQTT. Seperti yang ditunjukkan pada Gambar 6-20, klik kanan pada paket tingkat atas (dalam contoh ini adalah **com.codifythings.lightingcontrolsystem**) dan pilih **New** lalu pilih **Java Class.**



Gambar 6-20. Tambahkan kelas baru

Masukkan MQTTClient di **Name Field** dan klik OK, seperti yang ditunjukkan pada Gambar 6-21.



Gambar 6-21. Masukkan nama kelas baru

Android Studio akan menghasilkan kelas kosong dengan kode default yang ditampilkan di Listing 6-8.

Listing 6-8. Kode default untuk MQTTClient.java

```
public class MQTTClient
{
...
}
```

Selanjutnya Anda akan menambahkan kode ke MQTTClient yang akan terhubung dan dipublishkan ke broker MQTT setiap kali pengguna mengetuk layar aplikasi. Listing 6-9 menyediakan implementasi lengkap dari class MQTTClient.

Listing 6-9. Kode Lengkap MQTTClient.java

```
package com.codifythings.lightingcontrolsystem;
import android.util.Log;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
importorg.eclipse.paho.client.mqttv3.persist.MemoryPersistence;
public class MQTTClient {
    private static final String TAG = "MQTTClient";
    private String mqttBroker = "tcp://iot.eclipse.org:1883";
    private String mqttTopic = "codifythings/lightcontrol";
    private String deviceId = "androidClient";
    private String messageContent = "SWITCH";

public void publishToMQTT() throws MqttException {
    // Request clean session in the connection options.
```

```
        Log.i(TAG, "Setting Connection Options");
        MqttConnectOptions options = new MqttConnectOptions();
        options.setCleanSession(true);
        // Attempt a connection to MQTT broker using the values
        // of connection variables.
        Log.i(TAG, "Creating New Client");
        MqttClient client = new MqttClient(mqttBroker, deviceId, new
    MemoryPersistence());
        client.connect(options);
        // Publish message to topic
        Log.i(TAG, "Publishing to Topic");
        MqttMessage mqttMessage=new MqttMessage(messageContent.getBytes());
        mqttMessage.setQos(2);
        client.publish(mqttTopic, mqttMessage);
        Log.i(TAG, "Publishing Complete");
        Log.i(TAG, "Disconnecting from MQTT");
        client.disconnect();
      }
    }
```

Di Listing 6-9, variabel TAG akan digunakan saat masuk sehingga Anda dapat mengidentifikasi pesan aplikasi Anda di log. Variabel mqttBroker, mqttTopic, dan deviceId menentukan broker MQTT yang akan terhubung dengan aplikasi Anda, topik yang akan dipublishkan oleh aplikasi Anda, dan ID perangkat yang akan muncul di server saat aplikasi Anda berhasil terhubung. Jika Anda tidak menginstal broker MQTT di mesin Anda, Anda dapat menggunakan broker MQTT yang tersedia secara terbuka dari Eclipse Foundation.

Dalam proyek ini, Anda hanya mengalihkan status satu lampu, seperti dari hidup ke mati dan sebaliknya. Anda tidak mengontrol banyak lampu atau beberapa peralatan; oleh karena itu, Anda tidak perlu membuat perintah khusus untuk semua tindakan. Anda akan mempublishkan pesan berikut setiap kali pengguna mengetuk layar aplikasi.

Kode untuk menghubungkan dan memublikasikan ke broker MQTT menggunakan metode publishToMQTT(). Inisialisasi MqttClient baru dan sambungkan ke iot.Eclipse. org:1883 server dengan sesi bersih. Buat objek MqttMessage dan publishkan ke broker MQTT saat pengguna mengetuk layar aplikasi. Terakhir, putuskan sambungan aplikasi dari broker MQTT, karena Anda tidak memerlukan koneksi aktif secara keseluruhan.

Sekarang setelah konektivitas MQTT dan kode publish sudah siap, Anda akan kembali ke **MainActivity class** dan mengupdate metode onCreate(). Sebelumnya Anda telah menambahkan pendengar ke ikon cahaya yang akan dipanggil setiap kali pengguna mengetuk layar aplikasi. Anda akan memberikan implementasi pendengar yang hilang. Anda hanya akan menginisialisasi objek MQTTClient baru dan memanggil metode publishToMQTT() di dalam listener. Listing 6-10 menyediakan kode lengkap **MainActivity class** dalam metode onCreate() yang disorot.

Listing 6-10. Kode Lengkap MainActivity.java

```
package com.codifythings.lightingcontrolsystem;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
```

```
import android.view.View;
import android.widget.ImageView;
public class MainActivity extends AppCompatActivity {
 private static final String TAG = "MainActivity";
 @Override
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      ImageView lightIcon = (ImageView) findViewById(R.id.light_icon);
      lightIcon.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                  try {
                  new MQTTClient().publishToMQTT();
                  } catch (Exception ex) {
                  Log.e(TAG, ex.getMessage());
            }
        }
    });
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it
//is present.
getMenuInflater().inflate(R.menu.menu_main, menu);
return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
          // Handle action bar item clicks here. The action bar will
          // automatically handle clicks on the Home/Up button, so long
           // as you specify a parent activity in AndroidManifest.xml.
          int id = item.getItemId();
          //noinspection SimplifiableIfStatement
          if (id == R.id.action_settings) {
           return true;
          }
          return super.onOptionsItemSelected(item);
      }
}
```

Terakhir, Anda perlu **mengupdate** AndroidManifest.xml di bawah App lalu pilih folder Manifests. Aplikasi Anda perlu mengakses Internet untuk menghubungkan ke broker MQTT, jadi Anda juga perlu menambahkan izin Internet di AndroidManifest.xml. Listing 6-11 menyediakan kode yang perlu diUpdate di AndroidManifest.xml.

Listing 6-11. Tambahkan Izin Aplikasi di AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET" />
```

Kode (Arduino)

Selanjutnya, Anda akan menulis kode untuk menghubungkan Arduino ke Internet menggunakan WiFi, subscribe broker MQTT, dan mengontrol LED yang terpasang. Mulai Arduino IDE Anda dan ketik kode yang disediakan di sini atau unduh dari situs dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi lima bagian.

- Library eksternal
- Konektivitas Internet (Wi-Fi)
- MQTT (subscribe)
- Kontrol LED
- Fungsi standar

Library Eksternal

Bagian pertama kode mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Sketsa ini memiliki dua dependensi utama—untuk konektivitas Internet, Anda harus menyertakan <WiFi.h> (dengan asumsi Anda menggunakan pelindung WiFi) dan untuk komunikasi broker MQTT, Anda harus menyertakan <PubSubClient.h>. Listing 6-12 menyediakan bagian pertama dari kode dengan semua library yang diperlukan.

Listing 6-12. Kode untuk Menyertakan Dependensi Eksternal

```
#include <SPI.h>
#include <WiFi.h>
#include <PubSubClient.h>
```

Konektivitas Internet (Wereless)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (dalam Bab 2) di sini.

6.5 SUBSCRIBE DATA

Bagian ketiga dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke broker MQTT dan panggilan balik ketika pesan baru tiba (untuk detailnya, lihat Bab 3).

Ini adalah kode yang sama yang Anda lihat di Bab 3. Anda tidak perlu membuat perubahan apa pun agar kode berfungsi, tetapi Anda disarankan untuk menyesuaikan beberapa kode sehingga pesan Anda tidak tercampur dengan orang lain yang menggunakan nilai yang sama. Semua nilai yang dapat diubah telah disorot dalam huruf tebal pada Listing 6-13. Jika Anda menggunakan server MQTT Anda sendiri, pastikan untuk mengubah nilai server dan port. Dua perubahan yang disarankan termasuk nilai variabel topik dan nama klien yang harus Anda lewati saat terhubung ke broker MQTT. Setiap kali pesan baru diterima, fungsi callback() dipanggil. Ini mengekstrak muatan dan memanggil fungsi turnLightsOnOff().

Listing 6-13. Kode untuk Subscribe ke Broker MQTT

```
// IP address of the MQTT broker
char server[] = {" iot.eclipse.org "};
int port =  1883 ;
char topic[] = {" codifythings/lightcontrol "};
PubSubClient pubSubClient(server, port, callback, client);
void callback(char* topic, byte* payload, unsigned int length)  {
// Print payload
String payloadContent = String((char *)payload);
```

```
Serial.println("[INFO] Payload: " + payloadContent);
// Turn lights on/off
turnLightsOnOff();
}
```

6.6 KONTROL LAMPU

Bagian keempat dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk mengontrol LED. Kode yang diberikan dalam Listing 6-14 memeriksa apakah LED sudah hidup atau mati dan cukup mengganti status LED. Jika nilai digital port 3 HIGH, berarti LED menyala. Dalam hal ini, berubah menjadi LOW, yang mematikan LED.

Listing 6-14. Kode untuk Mengontrol Lampu LED

```
int ledPin = 3;
void turnLightsOnOff()
{
// Check if lights are currently on or off    if(digitalRead(ledPin) == LOW)
{
//Turn lights on
Serial.println("[INFO]
Turning lights on");
digitalWrite(ledPin, HIGH);
}
else
{
    // Turn lights off
    Serial.println("[INFO] Turning lights off");
    digitalWrite(ledPin, LOW);
    }
}
```

Fungsi Standar

Terakhir, kode di bagian kelima dan terakhir ditunjukkan pada Listing 6-15. Ini mengimplementasikan fungsi setup() dan loop() standar Arduino. Dalam fungsi setup(), kode menginisialisasi port serial, menghubungkan ke Internet, dan subscribe topik MQTT. Broker MQTT telah diinisialisasi dan subscribe, jadi dalam fungsi loop(), Anda hanya perlu menunggu pesan baru dari broker MQTT.

Listing 6-15. Kode untuk Fungsi Arduino Standar

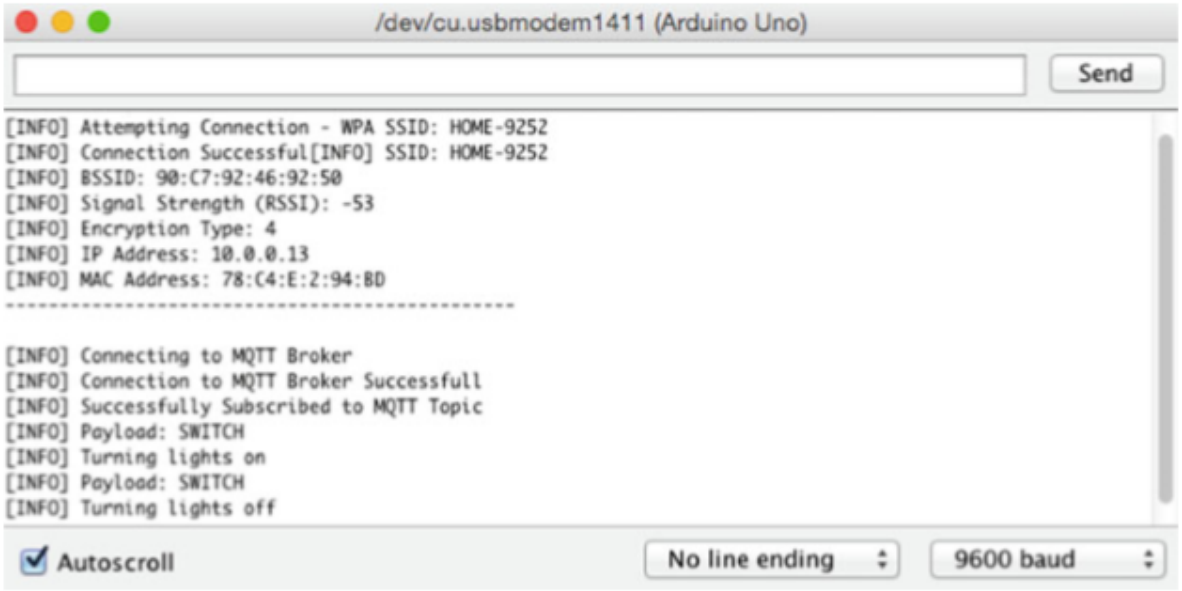
```
void setup()
{
    // Initialize serial port
    Serial.begin(9600);
    // Connect Arduino to internet
    connectToInternet();
    // Set LED pin mode
    pinMode(ledPin, OUTPUT);
    //Connect MQTT Broker
    Serial.println("[INFO] Connecting to MQTT Broker");
    if (pubSubClient.connect("arduinoClient"))
    {
    Serial.println("[INFO] Connection to MQTT Broker Successful");
```

```
    pubSubClient.subscribe(topic);
    }
    else
    {
    Serial.println("[INFO] Connection to MQTT Broker Failed");
    }
}
void loop()
{
// Wait for messages from MQTT broker
pubSubClient.loop();
```

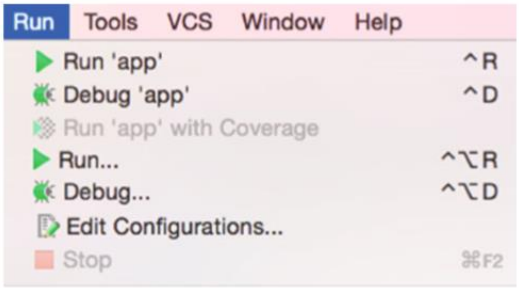
Kode Arduino Anda sekarang selesai.

Produk Akhir

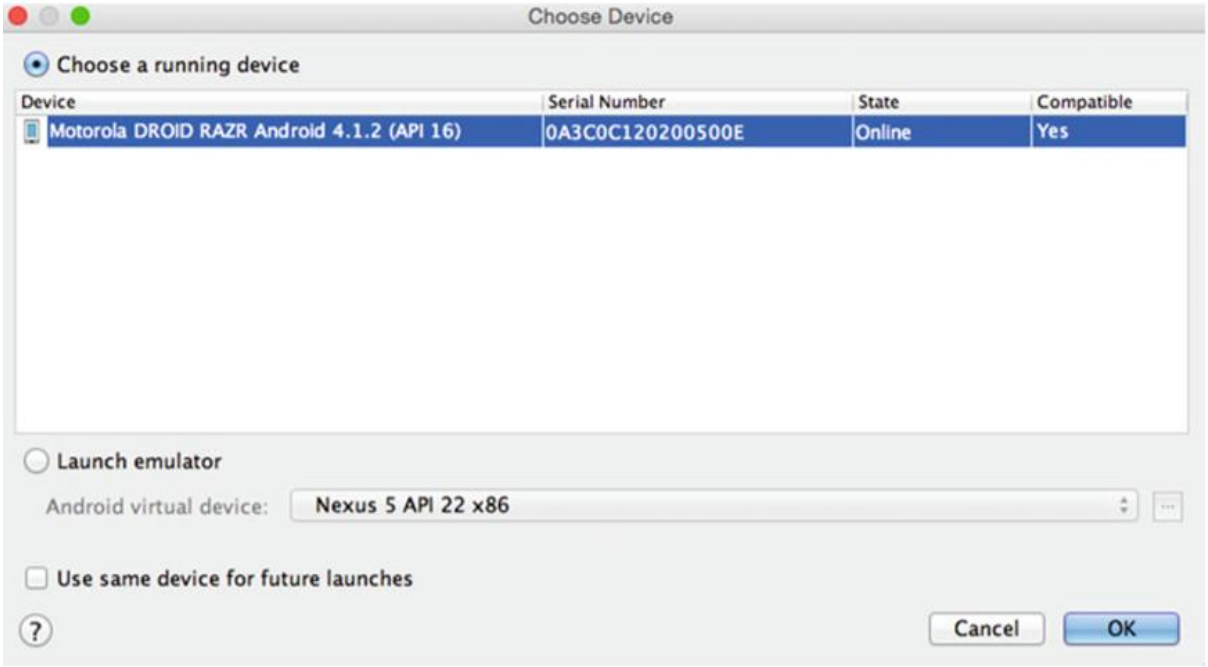
Untuk menguji aplikasi, verifikasi dan Upload kode Arduino seperti yang dibahas dalam Bab 1. Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log yang mirip dengan yang ditunjukkan pada Gambar 6-22.



Gambar 6-22. Pesan log dari Sistem Kontrol Pencahayaan



Gambar 6-23. Terapkan dan jalankan aplikasi dari Android Studio



Gambar 6-24. Pilih perangkat yang akan digunakan dan jalankan aplikasi

Di Android Studio Anda, terapkan dan jalankan aplikasi di perangkat Android Anda dengan memilih Jalankan lalu pilih Jalankan 'Aplikasi' dari bilah menu, seperti yang ditunjukkan pada Gambar 6-23.

Jika Anda memiliki perangkat Android yang terhubung ke komputer Anda, Android Studio akan meminta Anda untuk menggunakan perangkat berjalan yang ada atau meluncurkan emulator baru untuk menjalankan aplikasi. Seperti yang ditunjukkan pada Gambar 6-24, pilih emulator atau perangkat tempat Anda ingin menguji aplikasi dan klik **OK**. Buka perangkat tempat aplikasi Anda di-deploy. Jika aplikasi Anda belum berjalan, cari aplikasi Anda dan jalankan. Gambar 6-25 menunjukkan tampilan default aplikasi Anda.

Gambar 6-25. Tampilan default aplikasi Android

Ketuk layar dan periksa LED yang terpasang pada Arduino Anda. Statusnya harus berubah setiap kali Anda mengetuk.

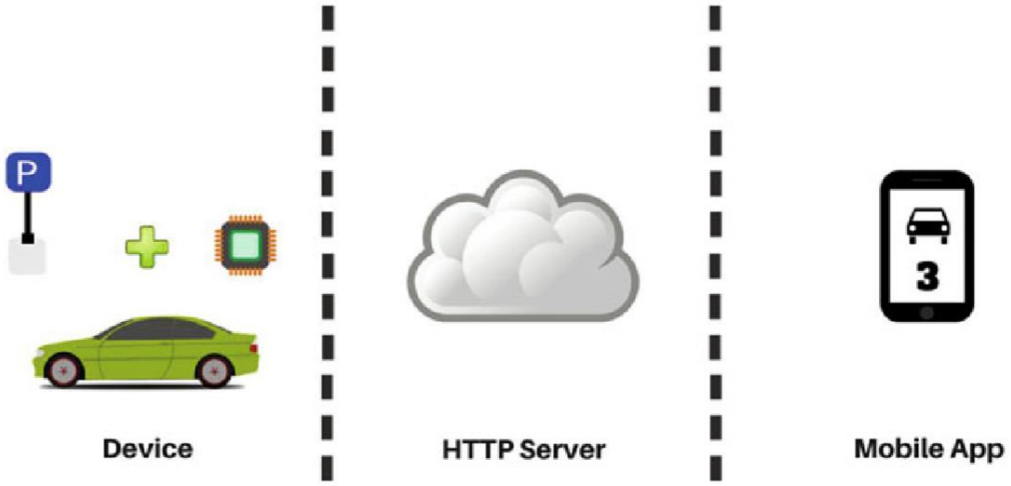
6.7 RINGKASAN

Dalam bab ini Anda telah mempelajari tentang pola kendali jarak jauh dari aplikasi IoT. Pola ini memungkinkan pengguna mengontrol perangkat mereka dari jarak jauh menggunakan antarmuka genggam atau berbasis web. Anda juga membuat aplikasi Android yang berfungsi sebagai remote control untuk perangkat Arduino Anda. Seperti yang disebutkan di Bab 5, aplikasi Android hanyalah salah satu contohnya. Kontrol jarak jauh dapat dibuat dari berbagai jenis seperti iOS, perangkat yang dapat dikenakan, dan aplikasi berbasis web.

BAB 7
POLA IoT: *ON-DEMAND CLIENTS*

Dibandingkan dengan pola IoT waktu nyata yang menyediakan data secara instan kepada pengguna akhir, pola sesuai permintaan menyediakan data kepada pengguna akhir hanya saat diminta. Aplikasi IoT yang dibangun menggunakan pola ini mendapatkan informasi dengan mengakses perangkat secara langsung atau dengan mendapatkannya dari lokasi yang telah disimpan sebelumnya. Pola sesuai permintaan berguna saat aplikasi Anda tidak secara aktif mencari data dan hanya mengaksesnya saat dibutuhkan.

Dalam bab ini, Anda akan membuat contoh pola ini, yang disebut *smarter parking system*. Gambar 7-1 menunjukkan diagram tingkat tinggi dari semua komponen yang terlibat dalam membangun sistem ini. Komponen pertama adalah perangkat Arduino yang memonitor status tempat parkir dengan sensor jarak dan mempublishkannya ke server menggunakan permintaan HTTP. Komponen kedua adalah server dengan layanan untuk menyimpan data tempat parkir dan layanan antarmuka yang menyediakan jumlah tempat parkir terbuka. Komponen terakhir adalah aplikasi iOS yang mengakses data tempat parkir terbuka dan menampilkannya kepada pengguna saat diminta.



Gambar 7-1. Komponen smarter parking system

Karena ini hanyalah sebuah contoh untuk membantu Anda lebih memahami polanya, ini sengaja dibuat sederhana. Anda akan memeriksa status hanya satu tempat parkir. Proyek ini dapat dengan mudah diskalakan untuk beberapa tempat parkir.

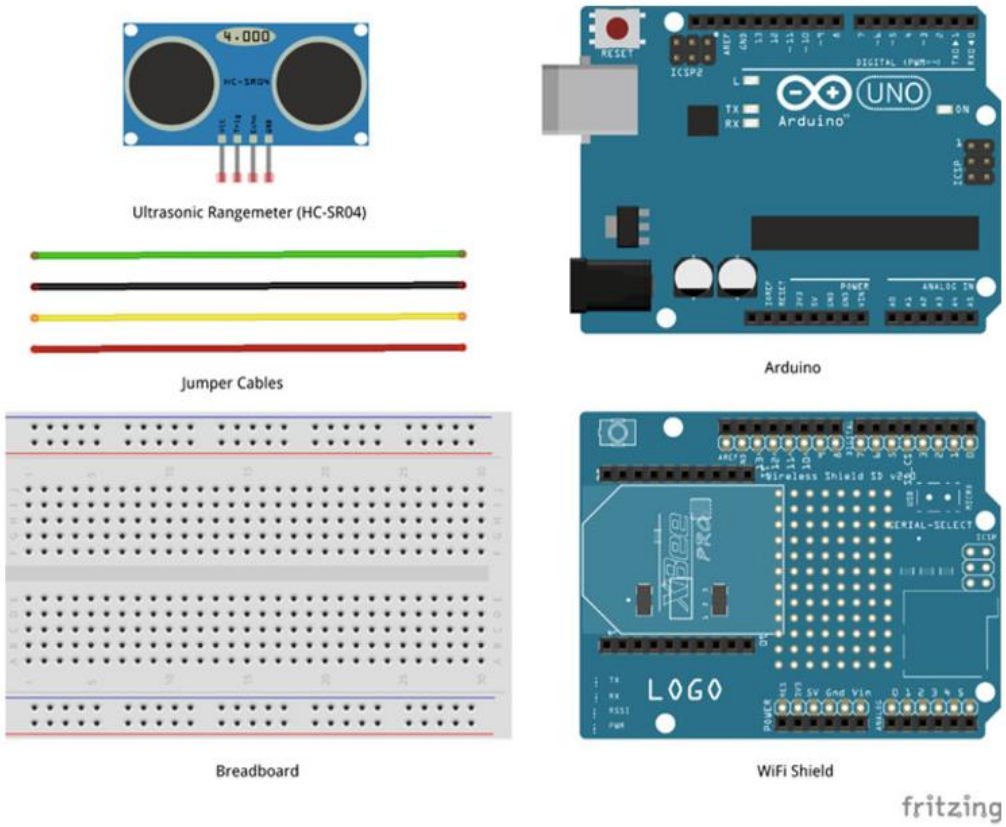
7.1 TUJUAN PEMBELAJARAN

Di akhir bab ini, Anda akan dapat:

- Membaca data dari sensor jarak
- Mengirim data sensor ke server menggunakan HTTP
- Menampilkan data sensor di aplikasi iOS menggunakan HTTP

Hardware yang Diperlukan

Gambar 7-2 memberikan daftar semua komponen hardware yang diperlukan untuk membangun smarter parking system ini.



Gambar 7-2. Hardware yang diperlukan untuk sistem smarter parking

Software yang Diperlukan

Untuk mengembangkan smarter parking system, Anda memerlukan software berikut:

- Arduino IDE 1.6.4 atau lebih baru
- Server PHP (diinstal atau di-host)
- Server MySQL (diinstal atau dihosting)
- Editor teks
- Xcode

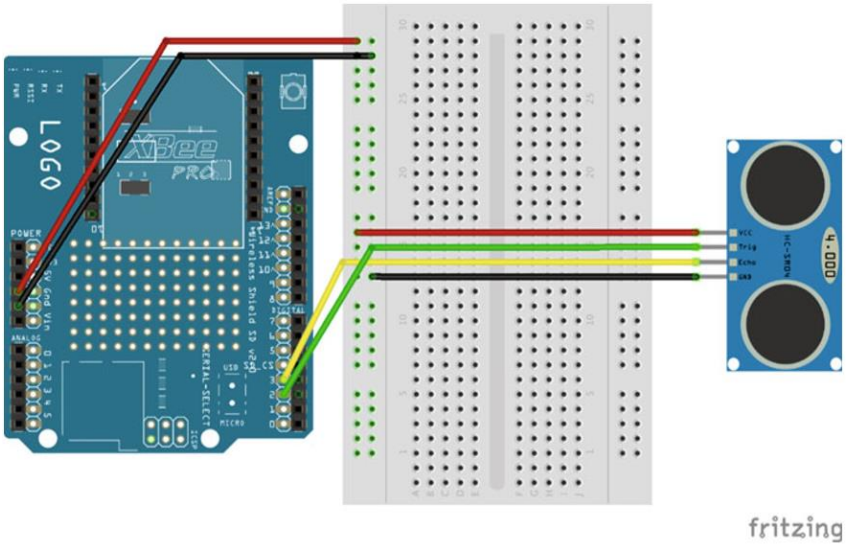
Sirkuit

Di bagian ini, Anda akan membangun sirkuit yang diperlukan untuk smarter parking system. Rangkaian ini menggunakan sensor jarak ultrasonik untuk mendeteksi objek. Sensor mengirimkan ledakan ultrasonik, yang dipantulkan dari objek di depannya. Rangkaian membaca gema yang digunakan untuk menghitung jarak ke objek terdekat.

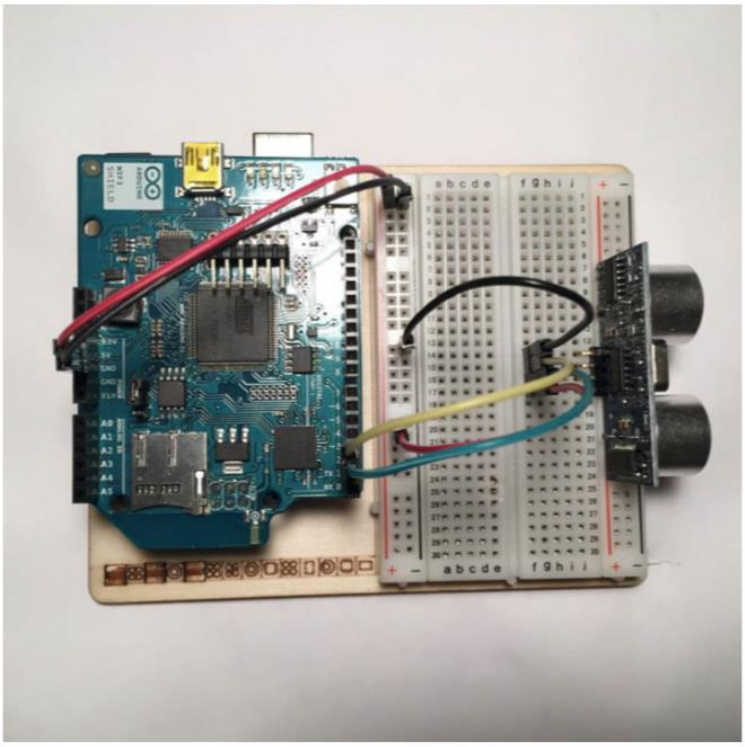
1. Pastikan Arduino tidak terhubung ke sumber listrik, seperti ke komputer melalui USB atau baterai.
2. Pasang pelindung WiFi ke bagian atas Arduino. Semua pin harus sejajar.
3. Gunakan kabel jumper untuk menghubungkan port power (5V) dan ground (GND) pada Arduino ke port power (+) dan ground (-) pada breadboard.
4. Sekarang papan breadboard Anda memiliki sumber daya, gunakan kabel jumper untuk menghubungkan port daya (+) dan arde (-) papan breadboard Anda ke port daya dan arde dari sensor jarak.
5. Untuk memicu ledakan ultrasonik, sambungkan kabel jumper dari pin TRIG sensor ke port digital 2 Arduino. Kode Anda akan mengatur nilai port ini ke LOW, HIGH, dan LOW untuk memicu burst.

6. Untuk membaca gema, sambungkan kabel jumper dari pin ECHO sensor ke port digital 3 Arduino. Kode Anda akan membaca nilai dari port ini untuk menghitung jarak objek.

Sirkuit Anda sekarang telah selesai dan akan terlihat mirip dengan Gambar 7-3 dan 7-4.



Gambar 7-3. Diagram sirkuit sistem smarter parking



Gambar 7-4. Sirkuit sebenarnya dari sistem smarter parking

7.2 TABEL DATABASE (MySQL)

Sebelum Anda dapat mengirim permintaan HTTP dari Arduino, Anda perlu membangun layanan yang akan menerima data. Data yang diterima dari Arduino perlu disimpan agar aplikasi iOS Anda dapat mengakses dan menampilkan informasi ini kepada pengguna. Persyaratan penyimpanan data untuk proyek ini relatif sederhana. Anda hanya perlu membuat tabel dua kolom yang dapat menyimpan jumlah tempat parkir terbuka dan stempel waktu untuk melacak kapan terakhir diUpdate.

Buku ini menggunakan MySQL sebagai databasenya. Buat tabel baru bernama PARKING_SPOTS_DATA menggunakan skrip SQL yang disediakan di Listing 7-1. Jalankan skrip ini di database yang ada atau buat yang baru. Kolom pertama akan berisi jumlah tempat parkir dan kolom kedua akan menjadi stempel waktu yang dibuat secara otomatis. Selain membuat tabel sql, Listing 7-1 juga berisi pernyataan insert. Pernyataan ini menginisialisasi jumlah tempat parkir, yang kemudian akan diUpdate saat data diterima dari sensor.

Listing 7-1. Buat dan Inisialisasi Tabel SQL

```
CREATE TABLE `PARKING_SPOTS_DATA` (
`PARKING_SPOTS_COUNT` int(11) NOT NULL,
`TIMESTAMP` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
)
INSERT INTO `PARKING_SPOTS_DATA`(`PARKING_SPOTS_COUNT`) VALUES (1)
```

Gambar 7-5 menunjukkan struktur tabel PARKING_SPOTS_DATA.

Gambar 7-5. Struktur tabel P ARKING_SPOTS_DATA

Kode (PHP)

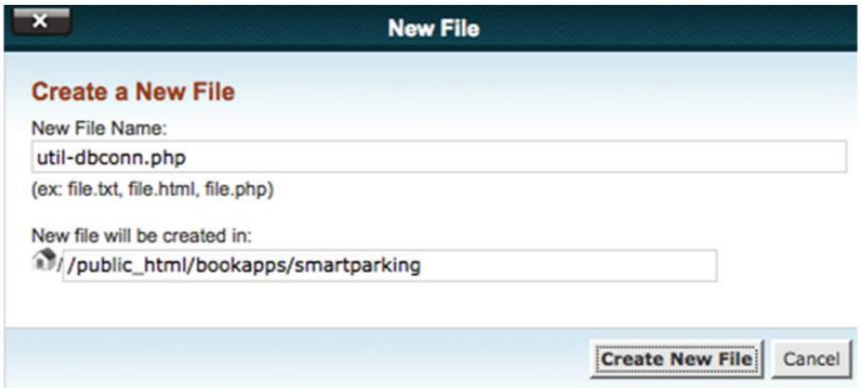
Sekarang tabel database sudah siap, Anda perlu membangun dua layanan. Layanan pertama akan menerima data sensor Arduino dalam permintaan HTTP dan dengan demikian **mengupdate** jumlah tempat parkir terbuka ke database. Layanan kedua akan bertindak sebagai antarmuka untuk aplikasi iOS—layanan ini akan mengembalikan data dalam format yang dapat diuraikan dan ditampilkan oleh aplikasi iOS.

Proyek ini menggunakan PHP untuk membangun penyimpanan data dan layanan antarmuka. PHP adalah bahasa pemrosesan sisi server yang sederhana dan open source yang dapat memproses permintaan HTTP dan mengirim tanggapan HTTP. Buat folder baru bernama smartparking di folder publik/root server PHP Anda. Semua kode sumber PHP untuk proyek ini akan masuk ke folder smartparking. Mulai editor teks pilihan Anda.

***Catatan** : Semua kode PHP dikembangkan menggunakan Brackets, yang merupakan editor teks open source. Kunjungi http://brackets.io/ untuk informasi lebih lanjut.*

7.3 KONEKSI DATABASE

Baik skrip PHP untuk penyimpanan data dan antarmuka harus terhubung ke database. Seperti yang ditunjukkan pada Gambar 7-6, buat file baru bernama **util-dbconn.php** di folder smartparking. File ini akan digunakan oleh kedua skrip alih-alih mengulangi kode.



Gambar 7-6. File konektivitas database umum yang disebut util-dbconn.php

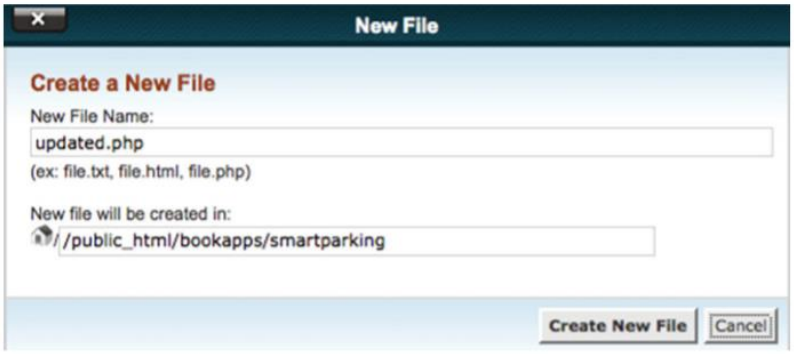
Buka file dalam editor teks dan salin atau ketik kode dari Listing 7-2. Seperti yang Anda lihat, tidak banyak kode dalam file ini. Empat variabel **$servername, $username, $password,** dan **$dbname** berisi informasi koneksi. Buat koneksi baru dengan melewatkan keempat variabel ini dan simpan referensi koneksi di variabel **$mysqli**. Kondisi IF dalam kode hanya memeriksa kesalahan selama upaya koneksi dan mencetaknya jika ada.

Listing 7-2. Kode Konektivitas Database Umum util-dbconn.php

```
<?php
        $servername =  "SERVER_NAME" ;
        $dbname =  "DB_NAME" ;
        $username =  "DB_USERNAME" ;
        $password =  "DB_PASSWORD" ;
        //Open a new connection to MySQL server
        $mysqli = new mysqli($servername, $username, $password, $dbname);
        //Output connection errors     if ($mysqli->connect_error)
        {
        die("[ERROR] Connection Failed: ". $mysqli->connect_error);
        }
?>
```

7.4 MENERIMA DAN MENYIMPAN DATA SENSOR

Seperti yang ditunjukkan pada Gambar 7-7, Anda dapat membuat file baru bernama **update.php** di dalam folder smartparking. Skrip ini akan melakukan dua tugas—pertama akan mengambil informasi dari permintaan HTTP dan kemudian akan **mengupdate** jumlah tempat parkir terbuka di database.



Gambar 7-7. File untuk menerima dan **mengupdate** data tersimpan yang disebut **update.php**

Buka file yang baru dibuat dalam editor teks dan salin atau ketik kode yang disediakan di Listing 7-3. Seperti disebutkan di langkah sebelumnya, untuk menyimpan data, koneksi database perlu dibuat, dan Anda membuat **util-dbconn.php** untuk melakukan tugas itu, jadi dalam file ini Anda perlu menyertakan **util-dbconn.php**. **Util-dbconn.php** menyediakan akses ke variabel $mysqli, yang berisi referensi koneksi dan akan digunakan untuk menjalankan kueri SQL.

Contoh dalam buku ini di-host di http://bookapps.codifythings.com/ smartparking, dan Arduino akan mengirimkan data tempat parkir terbuka ke update.php menggunakan metode HTTP GET. Seperti yang dibahas dalam Bab 2, HTTP GET menggunakan string kueri untuk mengirim data permintaan. Jadi, URL lengkap dengan string kueri yang akan digunakan Arduino menjadi

http://bookapps.codifythings.com/smartparking/update.php?parkingUpdate= BUKA. Kode PHP Anda perlu mengekstrak parkingUpdate dari string kueri menggunakan pernyataan **$_GET['parkingUpdate'].**

Karena Anda hanya memeriksa status satu tempat parkir, nilai default $currentParkingCount dalam kode ditetapkan sebagai 1, yang merupakan nilai yang sama dengan nilai awal database. Jika Anda memantau beberapa tempat parkir, Anda cukup menambah atau mengurangi hitungan berdasarkan data dari sensor jarak. Untuk proyek ini, kode hanya menetapkan nilai sebagai 1 setiap kali Arduino mengirim OPEN sebagai nilai parameter parkingUpdate dan menetapkan nilai ke 0 jika Arduino mengirimkan OCCUPIED sebagai nilai parameter.

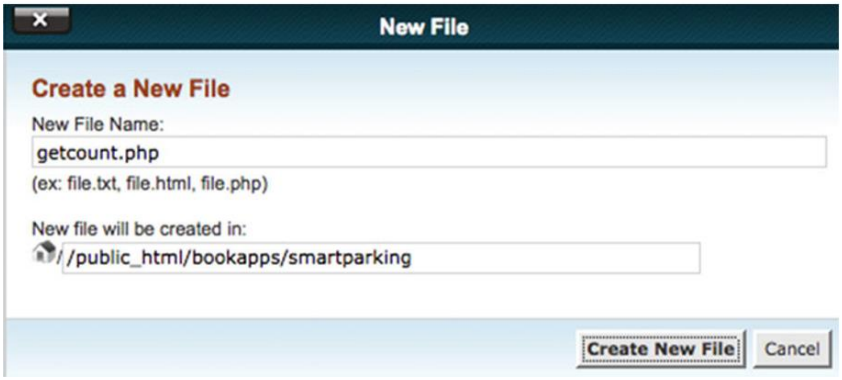
Untuk **mengupdate** hitungan ini dalam tabel database, siapkan pernyataan SQL UPDATE dalam variabel $sql. Anda hanya perlu meneruskan nilai $currentParkingCount dan nilai TIMESTAMP akan dibuat secara otomatis. Terakhir, jalankan pernyataan SQL UPDATE menggunakan $mysqli->query($sql) dan periksa variabel $result apakah berhasil atau gagal.

Listing 7-3. Kode untuk Menerima dan Mengupdate Data yang Disimpan di update.php

```
<?php
        include('util-dbconn.php');
        $parkingUpdate = $_GET['parkingUpdate'];
        echo "[DEBUG] Parking Update: ". $parkingUpdate. "\n";
        $currentParkingCount = 1;
        if($parkingUpdate == "OPEN")
        {
        $currentParkingCount = 1;
        }
        Else
        {
        $currentParkingCount = 0;
        }
        $sql = "UPDATE `PARKING_SPOTS_DATA` SET PARKING_SPOTS_COUNT =
  $currentParkingCount";
        if (!$result = $mysqli->query($sql))
        {
        echo "[Error] ". mysqli_error(). "\n";
        exit();
        }
        $mysqli->close();
        echo "[DEBUG] Updated Parking Spots Counter Successfully\n";
?>
```

7.5 DAPATKAN *PARKING SPOT COUNT*

Aplikasi Anda tidak akan langsung menanyakan database untuk jumlah tempat parkir; alih-alih Anda akan membuat layanan PHP yang mengembalikan informasi melalui HTTP. Seperti yang ditunjukkan pada Gambar 7-8, Anda harus membuat file baru bernama **getcount.php** di folder smartparking. Setelah aplikasi iOS memanggil URL http://bookapps.codifythings.com/smartparking/getcount.php, layanan PHP akan mengembalikan jumlah tempat parkir terbuka dari database dalam format JSON sebagai bagian dari respons HTTP.



Gambar 7-8. File untuk antarmuka ke database adalah getcount.php

Listing 7-4 menyediakan kode lengkap untuk getcount.ph, jadi salin atau tulis kode di getcount.php. Kode memerlukan akses ke database jadi sertakan util-dbconn.php, buat pernyataan sql SELECT baru, dan jalankan menggunakan **$mysqli->query($sql).** Periksa apakah ada hasil yang dikembalikan dan teruskan semua hasil dalam format JSON sebagai bagian dari respons HTTP.

Listing 7-4. Kode untuk Antarmuka ke Database di getcount.php

```
<?php
        include('util-dbconn.php');
        $sql = "SELECT PARKING_SPOTS_COUNT FROM
        `PARKING_SPOTS_DATA`";
        $result = $mysqli->query($sql);
        $resultCount = $result->num_rows;
        if ($resultCount > 0)
        {
        $row = $result->fetch_assoc();
        print(json_encode($row));
        }
        else
        {
        echo "0 results";
        }
        $mysqli->close();
?>
```

Kode (Arduino)

Komponen kedua dari proyek ini adalah kode Arduino. Kode ini menghubungkan Arduino ke Internet menggunakan WiFi, memeriksa apakah tempat parkir terbuka atau tidak, dan mempublishkan informasi ini ke server.

Mulai Arduino IDE Anda dan ketik kode yang disediakan atau unduh dari situs buku dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi lima bagian. :

- Library eksternal
- Konektivitas internet (WiFi)
- Membaca data sensor
- H TTP (terbitkan)
- Fungsi standar

Library Eksternal

Bagian pertama kode, seperti yang disediakan dalam Listing 7-5, mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Karena Anda terhubung ke Internet secara nirkabel, ketergantungan utama kode ini adalah pada <WiFi.h>.

Listing 7-5. Kode untuk Menyertakan Dependensi Eksternal

```
#include <SPI.h>
#include <WiFi.h>
```

Konektivitas Internet (Nirkabel)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (dalam Bab 2) di sini.

Baca Data Sensor

Bagian ketiga dari kode, seperti yang disediakan dalam Listing 7-6, mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membaca data sensor. Fungsi kalibrasiSensor() menunggu sensor jarak untuk mengkalibrasi dengan benar. Setelah kalibrasi selesai, sensor jarak aktif dan dapat memulai deteksi. Jika Anda tidak memberikan cukup waktu untuk mengkalibrasi, sensor jarak mungkin mengembalikan pembacaan yang salah.

Fungsi readSensorData() menghasilkan ledakan untuk mendeteksi jika tempat parkir kosong. Ini memicu ledakan pada Digital Pin 2 dengan mengirimkan sinyal alternatif—HIGH, HIGH, dan LOW lagi. Kemudian membaca gema dari Digital Pin 3, yang memberikan jarak objek terdekat. Akhirnya, ia memeriksa apakah nilai gema kurang dari ambang batas. Jika ya, itu berarti ada objek yang menempati tempat parkir. Karena ini hanya prototipe, nilai gema 500 telah digunakan, jadi ketika Anda menggunakan sensor ini dalam kehidupan nyata, Anda perlu menyesuaikan nilainya dengan melakukan beberapa tes. Jika tempat parkir ditempati, ia memanggil publishSensorData(...) dengan parameter OCCUPIED; jika tidak, ia akan mengirimkan OPEN dalam parameter.

Listing 7-6. Kode untuk Mendeteksi Jika Tempat Parkir Kosong

```
int calibrationTime = 30;
#define TRIGPIN 2       // Pin to send trigger pulse
#define ECHOPIN 3       // Pin to receive echo pulse
void calibrateSensor()
    {
    Give sensor some time to calibrate
    Serial.println("[INFO] Calibrating Sensor ");
    for(int i = 0;
    i < calibrationTime; i++)
    {
    Serial.print(".");
    delay(1000);
```

```
        }
        Serial.println("");
        Serial.println("[INFO]
        Calibration Complete");
        Serial.println("[INFO] Sensor Active");
        delay(50);
        }
        void readSensorData()
        {
        // Generating a burst to check for objects
        digitalWrite(TRIGPIN, LOW);
        delayMicroseconds(10);
        digitalWrite(TRIGPIN, HIGH);
        delayMicroseconds(10);
        digitalWrite(TRIGPIN, LOW);
        // Distance Calculation
        float distance = pulseIn(ECHOPIN, HIGH);
        Serial.println("[INFO] Object Distance: " + String(distance));
        if(distance < 500)
        {
        Serial.println("[INFO] Parking Spot Occupied");
        // Publish sensor data to server
        publishSensorData("OCCUPIED");
        }
        else
        {
        Serial.println("[INFO] Parking Spot Open");
        // Publish sensor data to server
        publishSensorData("OPEN");
        }
    }
```

Data Publish

Bagian keempat dari kode, seperti yang disediakan dalam Listing 7-7, mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membuat dan mengirim permintaan HTTP ke server. Kode ini adalah versi yang sedikit dimodifikasi dari HTTP GET yang dikembangkan di Bab 3.

Modifikasi utama dalam kode ini adalah kemampuannya untuk membuka dan menutup koneksi ke server secara berulang. Selain itu pastikan untuk mengubah nilai server dan port ke nilai server PHP Anda. Pastikan untuk mengubah variabel server, port, dan requestData serta nilai URL.

Listing 7-7. Kode untuk Mengirim Permintaan HTTP

```
    //IP address of the server
    char server[] = { "bookapps.codifythings.com" };
    int port =  80 ;
    unsigned long lastConnectionTime = 0;
    const unsigned long postingInterval = 10L * 1000L;
    void publishSensorData(String updateParkingSpot)
    {
```

```
        // Read all incoming data (if any)
        while (client.available())
        {
        char c = client.read();
        Serial.write(c);
        }
        if (millis() - lastConnectionTime > postingInterval)
        {
        client.stop();
        Serial.println("[INFO] Connecting to Server");
        String requestData = "parkingUpdate=" + updateParkingSpot;
        // Prepare data or parameters that need to be posted to server
        if (client.connect(server, port))
        {
        Serial.println("[INFO] Server Connected - HTTP GET Started");
        // Make HTTP request:
        client.println("GET /smartparking/update.php?" + requestData + " HTTP/1.1");
        client.println("Host: " + String(server));
        client.println("Connection: close");
        client.println();
        lastConnectionTime = millis();
        Serial.println("[INFO] HTTP GET Completed");
        }
        else
        {
        // Connection to server:port failed
        Serial.println("[ERROR] Connection Failed");
        }
    }
    Serial.println("-----------------------------------------");
    }
```

Fungsi Standar

Bagian kode kelima dan terakhir ditunjukkan pada Listing 7-8. Ini mengimplementasikan fungsi setup() dan loop() standar Arduino. Fungsi setup() menginisialisasi port serial, menyetel mode pin untuk pin trigger dan gema, menghubungkan ke Internet, dan mengkalibrasi sensor jarak. Fungsi loop() perlu memanggil readSensorData() secara berkala karena secara internal memanggil fungsi publishSensorData().

Listing 7-8. Kode untuk Fungsi Arduino Standar

```
    void setup()
    {
        // Initialize serial port
        Serial.begin(9600);
        // Set pin mode
        pinMode(ECHOPIN, INPUT);
        pinMode(TRIGPIN, OUTPUT);
        // Connect Arduino to internet
        connectToInternet();
        // Calibrate sensor
```

```
    calibrateSensor();
    }
    void loop()
    {
    // Read sensor data
    readSensorData();
    delay(5000);
}
```

Kode Arduino Anda sekarang sudah lengkap

7.6 KODE (IOS)

Komponen terakhir dari aplikasi IoT Anda adalah aplikasi iOS yang akan menunjukkan jumlah tempat parkir terbuka kepada pengguna. Aplikasi akan mengambil jumlah tempat parkir terbuka dari layanan PHP setiap kali pengguna mengetuk tombol Segarkan.

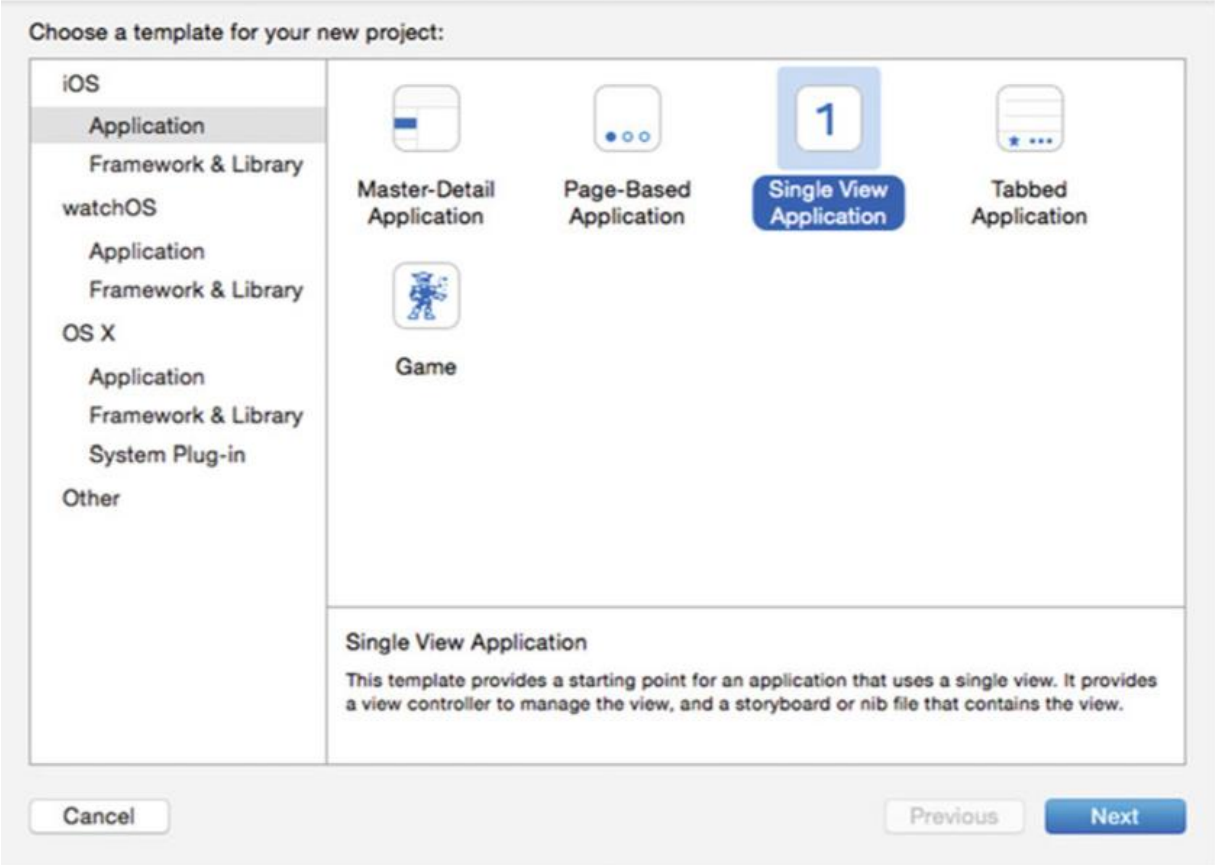
Gambar 7-9. Buat proyek Xcode baru

7.7 PENGATURAN PROYEK

Di bagian ini, Anda akan mengatur proyek Xcode Anda untuk aplikasi iOS. Anda dapat mengunduh Xcode dari https://developer.apple.com/xcode/download/. Xcode juga dapat diunduh langsung dari Mac App Store. Mengembangkan dan menguji aplikasi iOS di Xcode gratis. Anda dapat menggunakan simulator bawaan untuk menguji aplikasi Anda. Untuk menguji aplikasi Anda di perangkat iOS atau mempublishkannya ke App Store, Anda memerlukan akun pengembang berbayar (https://developer.apple.com/programs/). Bab ini menggunakan emulator bawaan untuk pengujian, jadi Anda tidak memerlukan akun berbayar untuk menyelesaikan bab ini. Mulai Xcode dari Aplikasi dan, seperti yang ditunjukkan pada Gambar 7-9, klik Buat Proyek kode X Baru.

Pilih Aplikasi Tampilan Tunggal untuk proyek tersebut, seperti yang ditunjukkan pada Gambar 7-10. Persyaratan layar untuk aplikasi ini sangat sederhana dan template Single

View Application menyelesaikannya. Jika Anda tertarik untuk membangun aplikasi yang lebih rumit, Anda dapat menggunakan salah satu template lain yang disediakan Xcode. **Klik Next**.

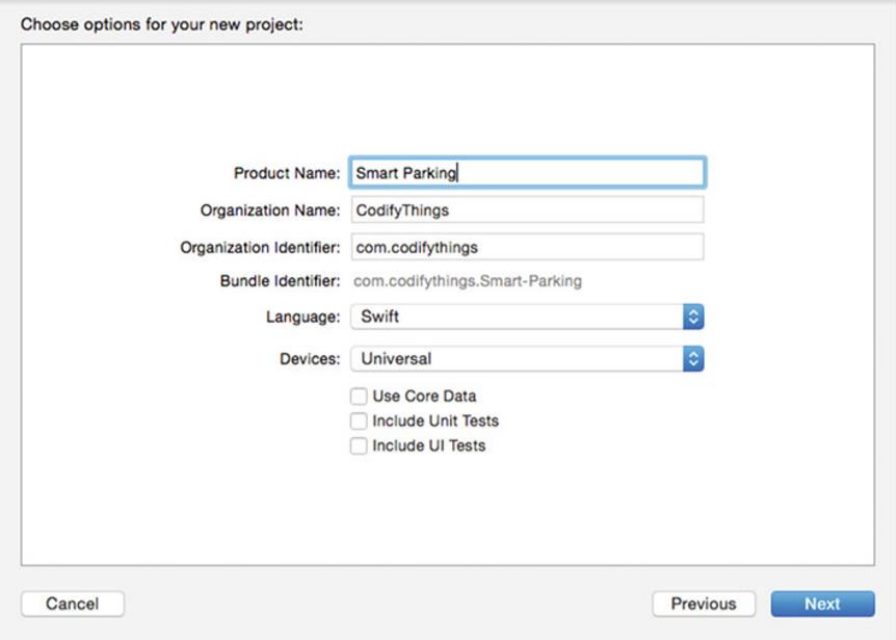


Gambar 7-10. Layar pemilihan template

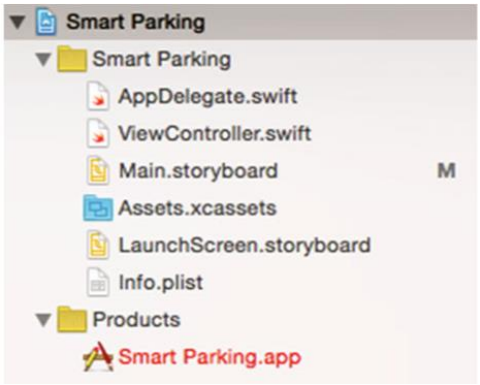
Isi detail proyek Anda seperti yang ditunjukkan pada Gambar 7-11. Anda harus memberikan Nama Produk, Nama Organisasi, dan Pengenal Organisasi. Pengidentifikasi Organisasi dapat berupa nama domain perusahaan atau pribadi Anda, yang digunakan untuk membuat Pengidentifikasi Bundel. Proyek ini akan dikembangkan menggunakan Swift, jadi pilih opsi itu dari Languange top-down. Jika Anda ingin aplikasi Anda berjalan di semua jenis perangkat iOS, lalu pilih Universal. Hapus centang pada semua opsi lain karena tidak diperlukan untuk proyek ini. Klik **next**.

Pilih lokasi di mesin Anda di mana Anda ingin menyimpan proyek dan klik Buat. Xcode akan membuat beberapa folder dan file, seperti yang ditunjukkan pada Gambar 7-12. Berikut ini adalah yang paling penting:

- Parkir Cerdas > Main.storyboard : File ini menyediakan titik masuk utama untuk aplikasi Anda dan digunakan untuk mendesain antarmuka visual.
- Parkir Cerdas > ViewController.swift : File ini berisi semua kode Swift untuk membuat aplikasi Anda dinamis dan interaktif.
- Parkir Cerdas > Assets.xcassets : File ini berisi semua aset yang akan digunakan oleh aplikasi (gambar).
- Smart Parking > Info.plist : File ini berisi informasi penting tentang runtime aplikasi.



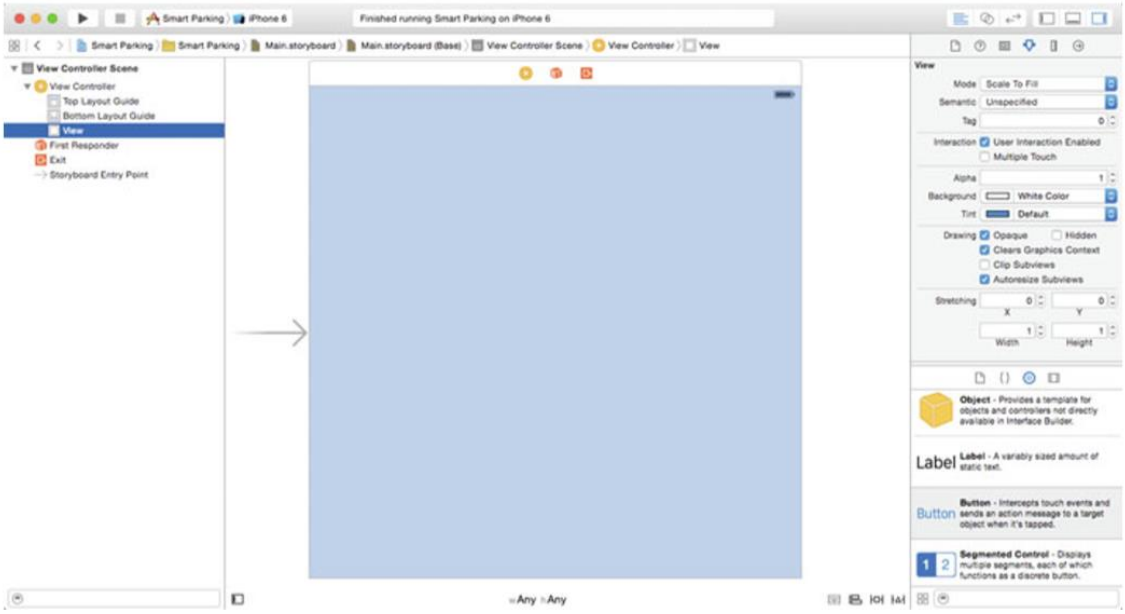
Gambar 7-11. Konfigurasi proyek c baru



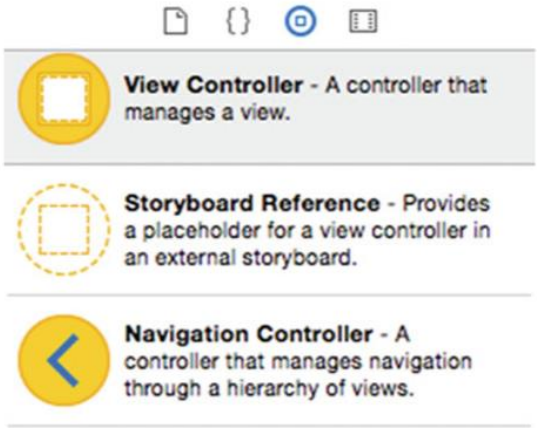
Gambar 7-12. File default yang dihasilkan oleh kode X

Layout Screen

Untuk mulai merancang *Layout Screen*, klik **Main.storyboard** di bawah folder Smart Parking utama. Ini akan membuka storyboard default, seperti yang ditunjukkan pada Gambar 7-13. Storyboard adalah tempat Anda menarik dan melepas berbagai widget untuk membuat antarmuka pengguna aplikasi Anda. Semua widget tersedia di Object Library di sisi kanan bawah storyboard. Gambar 7-14 menunjukkan Object Library tempat Anda dapat menarik dan melepas widget di storyboard.

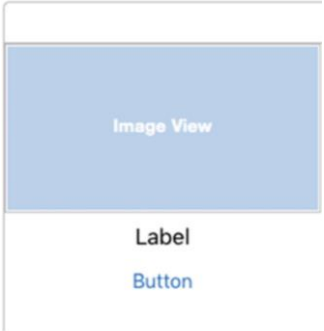


Gambar 7-13. Tampilan pengembangan default Xcode



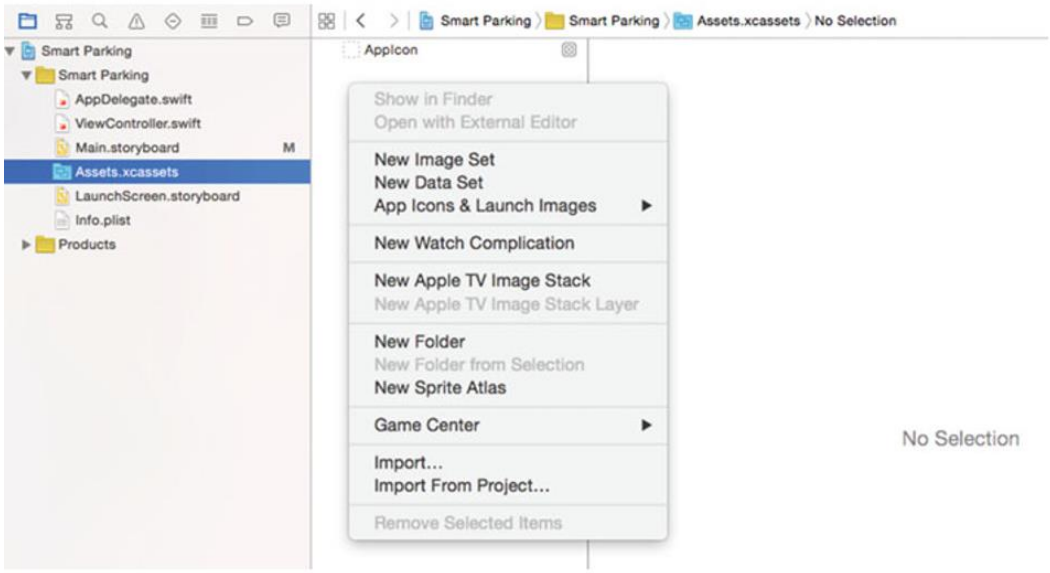
Gambar 7-14. Library objek dengan widget antarmuka pengguna

Aplikasi ini membutuhkan tiga widget di layar—yang pertama adalah ImageView opsional untuk menampilkan gambar mobil, yang kedua adalah Label untuk menampilkan tempat parkir yang terbuka, dan yang ketiga adalah tombol yang dapat diklik pengguna untuk memeriksa ulang tempat parkir yang terbuka. Drag widget ini dari *Object Library* ke *storyboard*; jangan khawatir tentang keselarasan atau ukuran widget sekarang. Layar Anda akan terlihat seperti Gambar 7-15.



Gambar 7-15. Layar S dengan widget yang diperlukan

Karena widget pertama adalah ImageView, Anda perlu mengatur gambar di properti. Anda perlu mengimpor gambar di Assets.xcassets. Seperti yang ditunjukkan pada Gambar 7-16, pilih Assets.xcassets dan klik kanan untuk melihat menu, lalu pilih Impor. Anda dapat memberikan gambar Anda sendiri atau menggunakan yang sama yang telah digunakan dalam contoh dari https://openclipart.org/detail/122965/car-pictogram. Pilih gambar dan **Klik Open**.

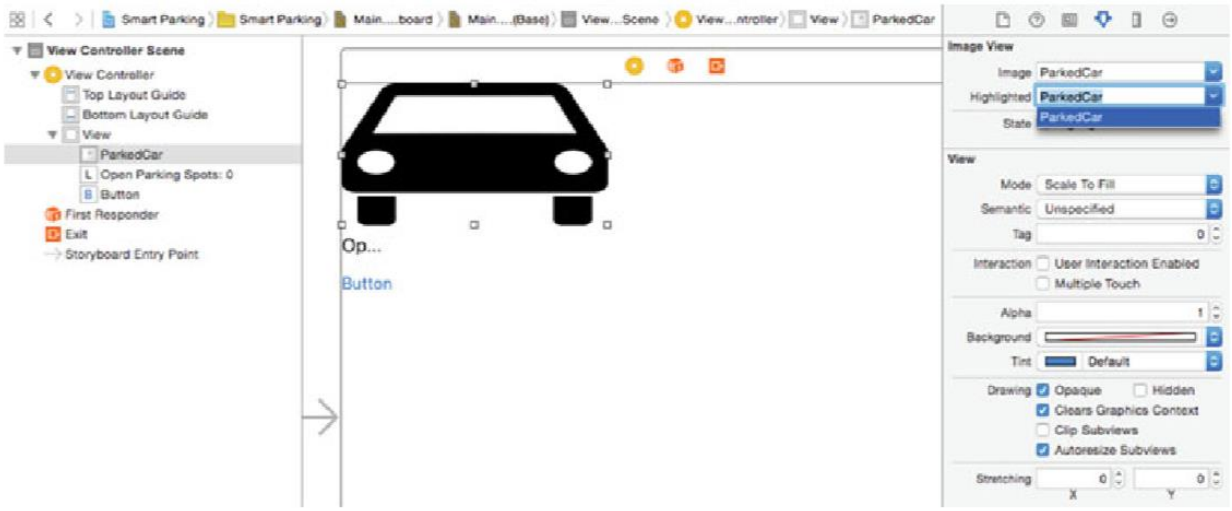


Gambar 7-16. Saya mengimpor aset gambar

Setelah diimpor, gambar akan tersedia di aplikasi, seperti yang ditunjukkan pada Gambar 7-17.

Gambar 7-17. Aset yang tersedia/diimpor

Sekarang gambar tersedia di aset, Anda perlu mengatur gambar ini di widget ImageView yang Anda tambahkan di storyboard. Beralih kembali ke **Main.storyboard** dan pilih **ImageView** dari storyboard. Seperti yang ditunjukkan pada Gambar 7-18, dari Attribute Inspector di sebelah kanan, pilih **ParkedCar** (atau nama gambar Anda) di menu tarik-turun Gambar dan Sorotan. Anda selalu dapat mengatur dua gambar berbeda saat gambar disorot versus skenario normal.

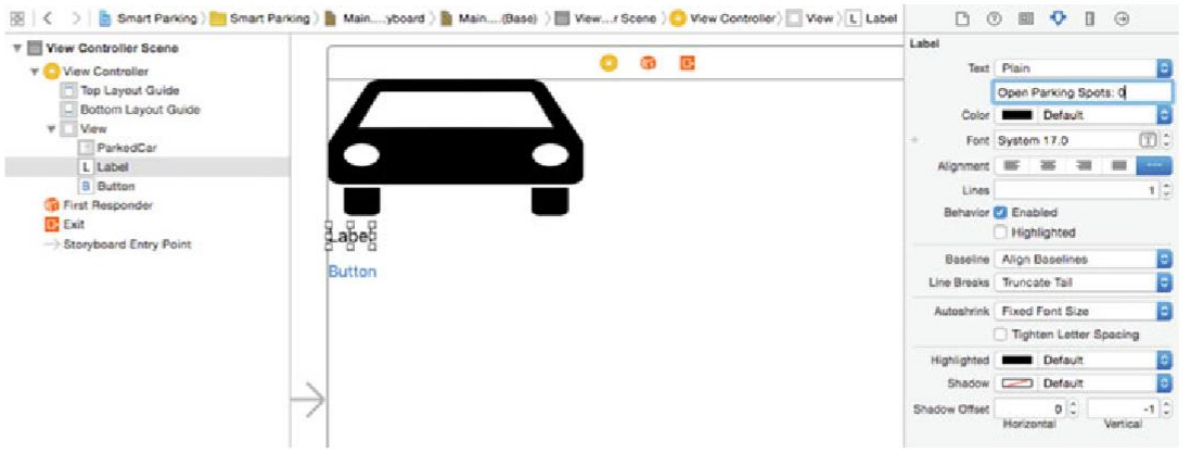


Gambar 7-18. Properti ImageView

Setelah properti ImageView disetel, gambar juga akan terlihat di storyboard. ImageView Anda akan terlihat mirip dengan Gambar 7-19 (atau gambar yang Anda pilih).

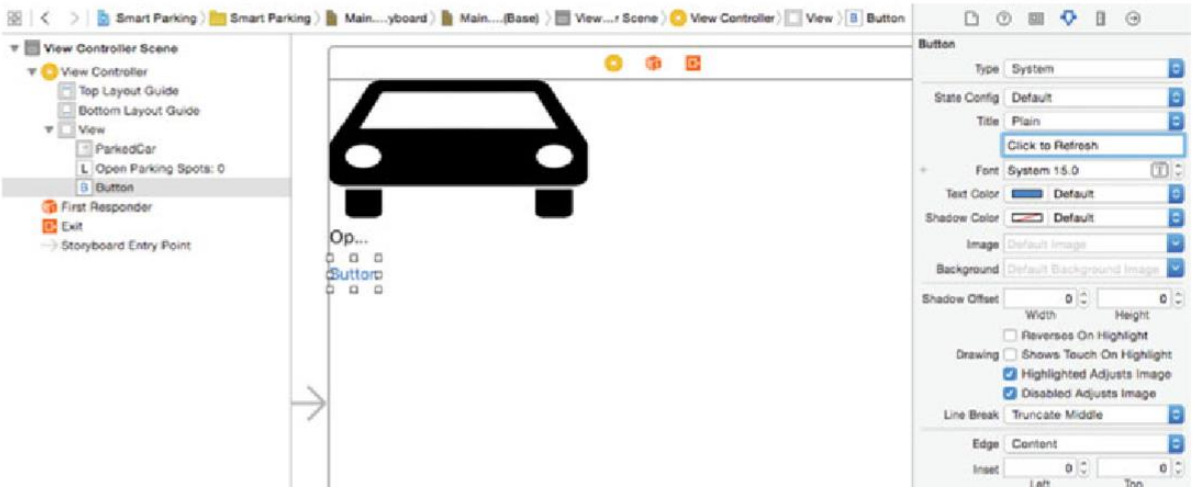
Gambar 7-19. Image View dengan gambar yang baru diimpor

Sama seperti dengan ImageView, Anda juga perlu **mengupdate** properti Label dan Tombol. Pilih **Label** dan, dari Attribute Inspector, ubah properti Text menjadi **Open Parking Spots: 0,** seperti yang ditunjukkan pada Gambar 7-20.



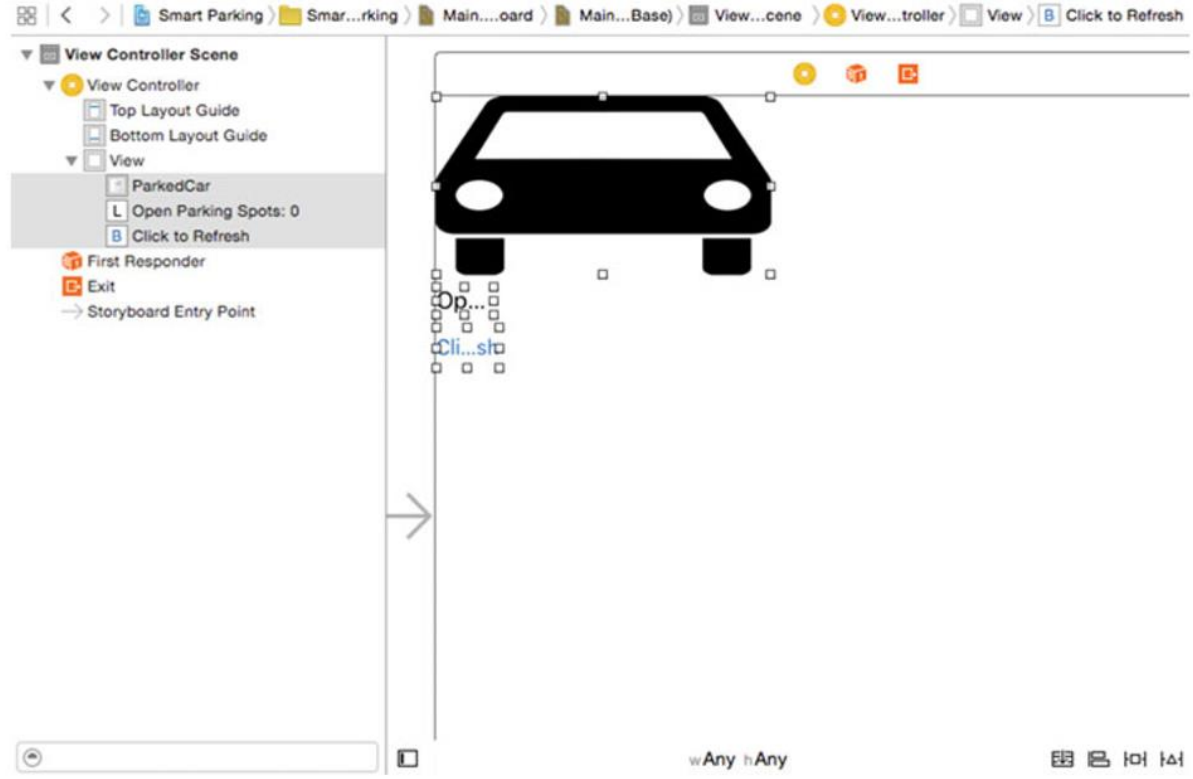
Gambar 7-20. Label properti

Selanjutnya, pilih **Button** dan, dari Attribute Inspector, ubah properti Title menjadi Click to **Refresh**, seperti yang ditunjukkan pada Gambar 7-21.



Gambar 7-21. Properti tombol

Anda hampir selesai dengan Layout Screen. Langkah terakhir melibatkan menyelaraskan widget. Seperti yang ditunjukkan pada Gambar 7-22, pilih ketiga widget.

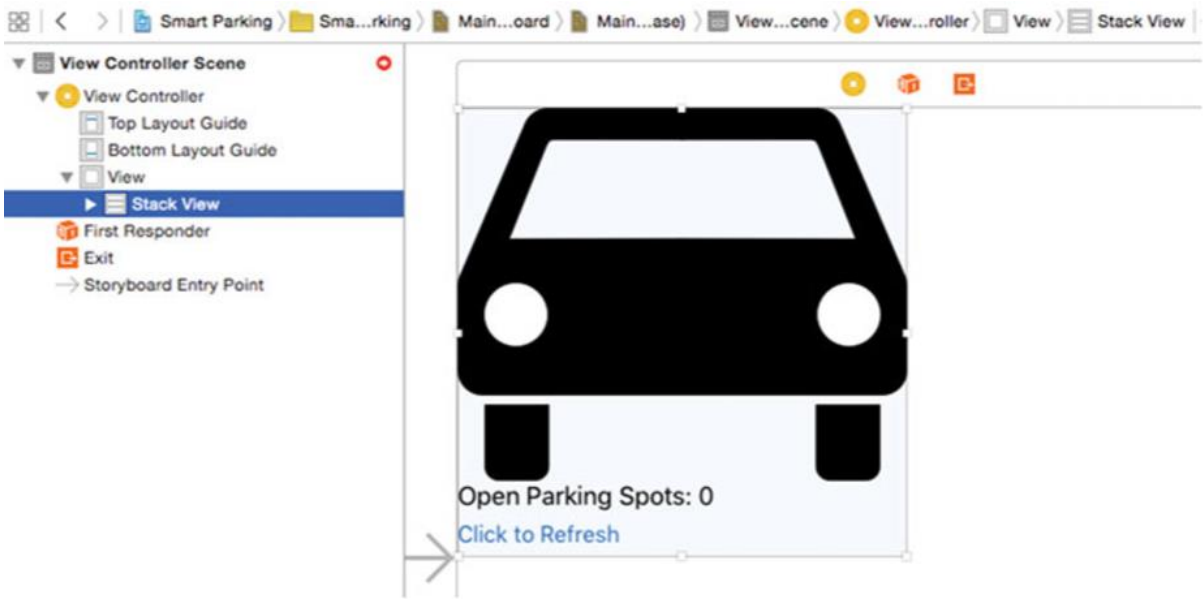


Gambar 7-22. Sejajarkan widget

Gambar 7-23 memberikan versi perataan yang diperbesar dan membatasi menu yang terlihat di sisi kanan bawah storyboard.

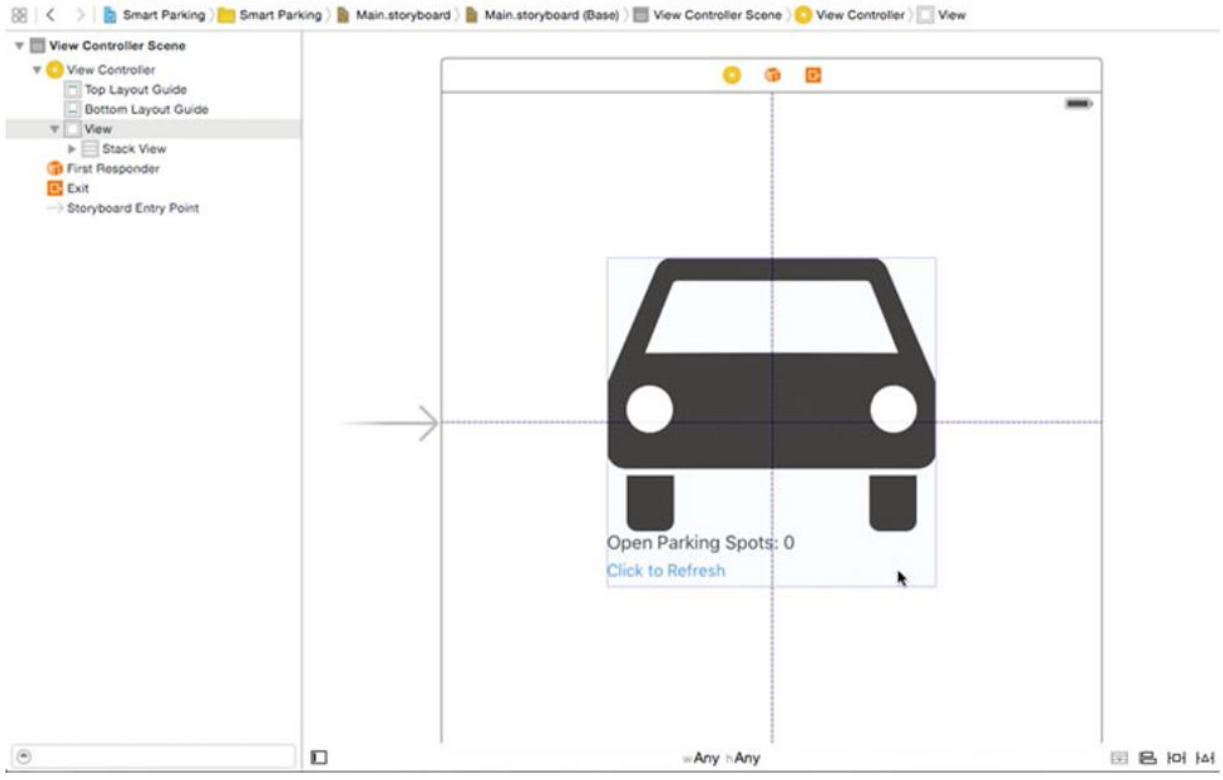
Gambar 7-23. Menu ligamen dan batasan

Saat ketiga widget dipilih, klik tombol Stack ( ). Ini akan menumpuk semua widget yang dipilih secara vertikal, seperti yang ditunjukkan pada Gambar 7-24.



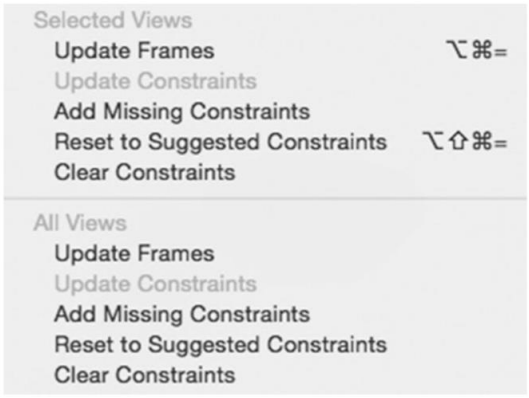
Gambar 7-24. Widget yang ditumpuk secara vertikal

Untuk menyelesaikan penyelarasan layar, Anda perlu menambahkan beberapa batasan ke widget sehingga, meskipun berjalan di perangkat yang berbeda, perilakunya konsisten. Seperti yang ditunjukkan pada Gambar 7-24, pastikan Anda memilih **Stack View** di View Controller. **Drag Stack View** ke tengah layar sehingga panduan perataan horizontal dan vertikal terlihat, seperti yang ditunjukkan pada Gambar 7-25.



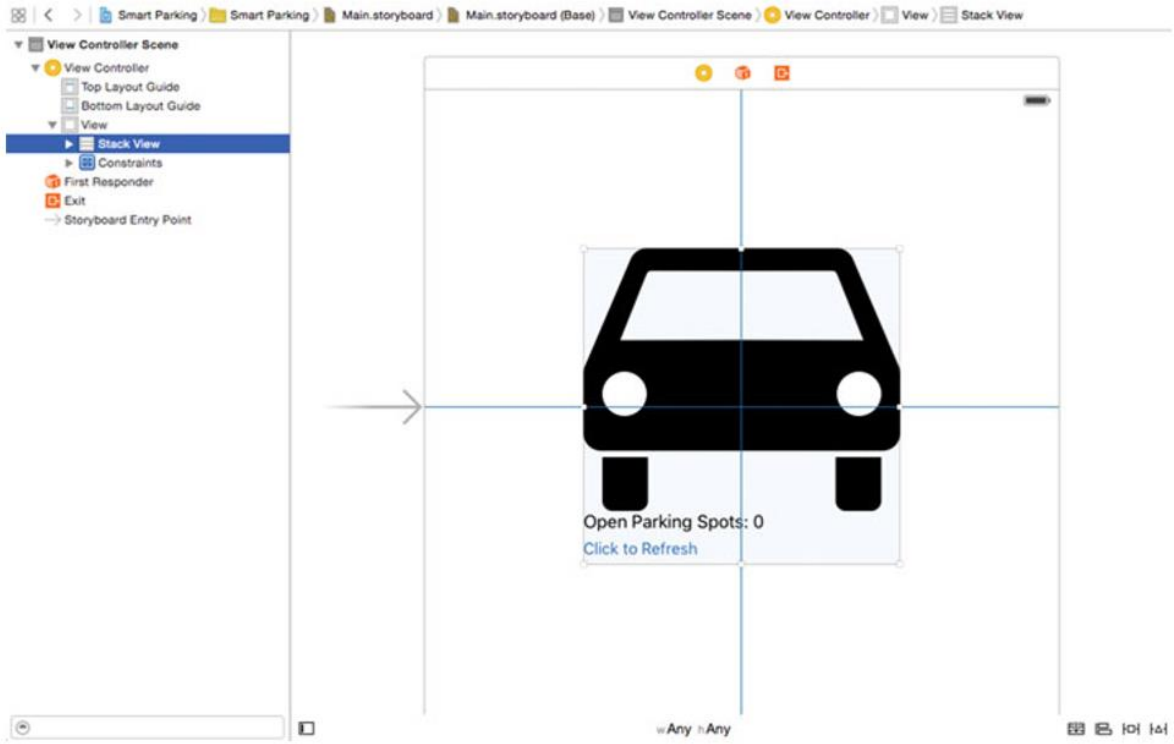
Gambar 7-25. Widget rata tengah

Saat Stack View dipilih dari menu perataan dan batasan yang ditunjukkan pada Gambar 7-24, klik tombol Resolve Auto Layout Issues ( ). Pilih Add Missing Constraints untuk Selected Views, seperti yang ditunjukkan pada Gambar 7-26.



Gambar 7-26. Add Missing Contrains

Layout Screen Anda sudah siap dan akan terlihat seperti Gambar 7-27.



Gambar 7-27. Layout Screen akhir aplikasi

Logika Layar

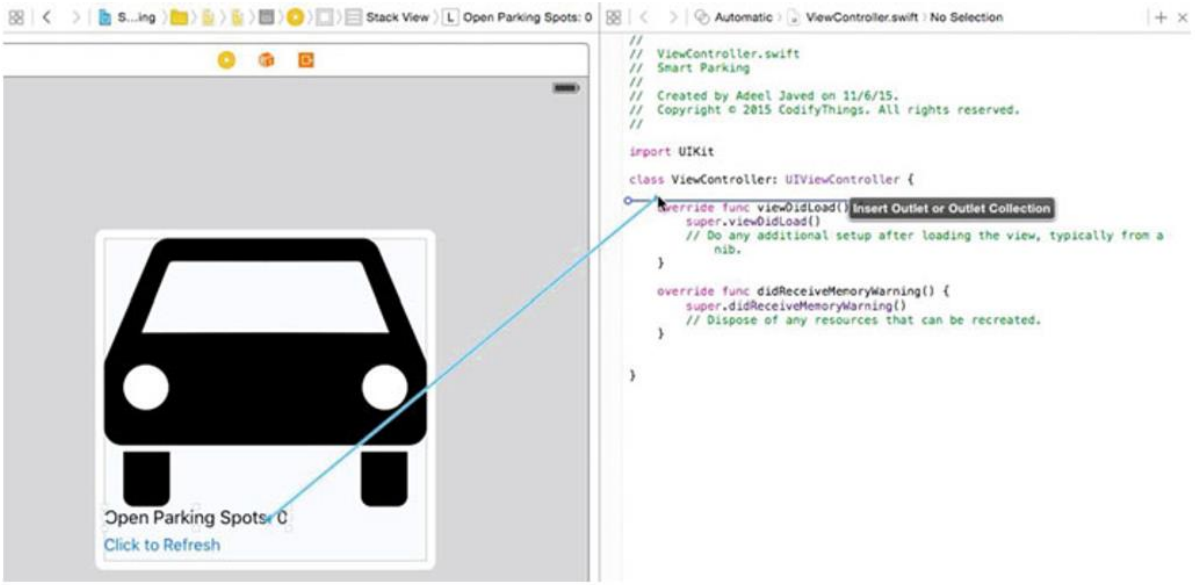
Selanjutnya Anda akan menambahkan beberapa logika yang akan membuat layar menjadi interaktif. Setiap kali pengguna mengetuk tombol Klik untuk Refresh, aplikasi akan memeriksa server untuk informasi tempat parkir terbuka. Buka file ViewController.swift berdampingan dengan storyboard. Seperti yang ditunjukkan pada Listing 7-9, secara default akan ada dua fungsi, yang disebut viewDidLoad() dan didReceiveMemoryWarning(), yang dibuat secara otomatis oleh sistem. Anda tidak akan membuat perubahan apa pun pada fungsi-fungsi ini.

Listing 7-9. Kode default untuk ViewController.swift

```
import UIKit
```

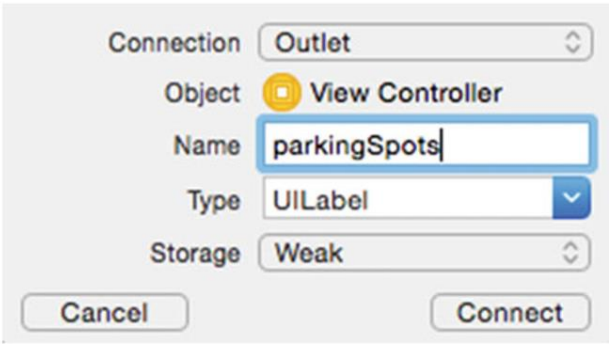
```
class ViewController: UIViewController {
    override func viewDidLoad()
    {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically
        // from a nib.
}
override func didReceiveMemoryWarning() {
super.didReceiveMemoryWarning()
// Dispose of any resources that can be recreated.
    }
}
```

Untuk mengupdate nilai tempat parkir terbuka di layar, Anda memerlukan referensi label Tempat Parkir Terbuka: 0 di file **ViewController.swift** Anda. Xcode menyediakan cara yang sangat mudah untuk melakukannya, seperti yang ditunjukkan pada Gambar 7-28. Anda cukup drag dan drop label dari storyboard pada file **ViewController.swift**. Pastikan Anda terus menekan tombol **Ctrl** pada keyboard.



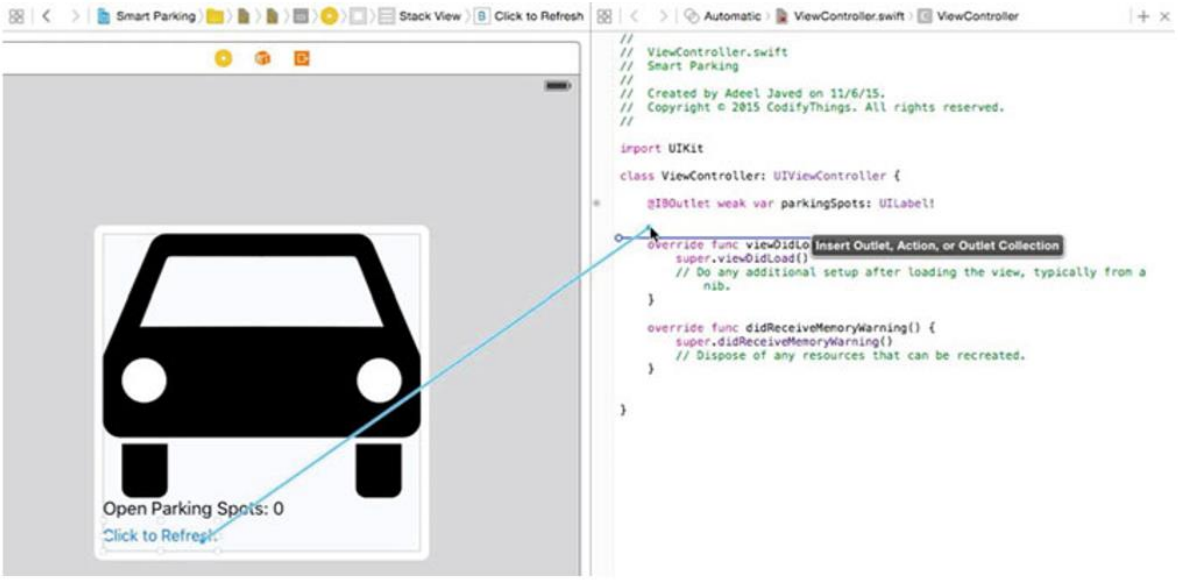
Gambar 7-28. D rag and drop label dari storyboard

Saat Anda melepaskan label pada kode, Xcode akan menampilkan popup untuk memasukkan nama properti ini. Seperti yang ditunjukkan pada Gambar 7-29, masukkan nama dan pastikan Anda membiarkan Connection diatur ke Outlet. Klik Connect untuk menambahkan properti baru ke ViewController.swift.



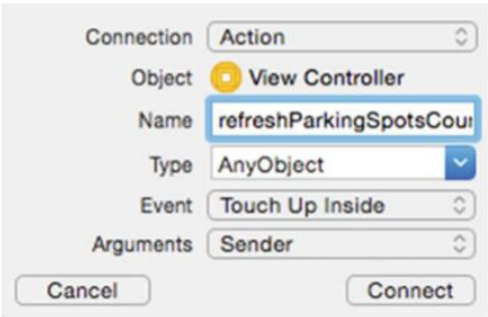
Gambar 7-29. Properti outlet

Drag dan lepas tombol **Click to Refresh** dengan cara yang sama dari storyboard pada file **ViewController.swift**, seperti yang ditunjukkan pada Gambar 7-30.



Gambar 7-30. Drag and drop button dari s toryboard

Seperti yang ditunjukkan pada Gambar 7-31, dari popup properti, pilih Tindakan dari Sambungan karena Anda perlu menambahkan kode untuk merespons setiap kali pengguna mengetuk tombol. Di bidang Nama, masukkan **refreshParkingSpotsCount** dan klik Hubungkan.



Gambar 7-31. Properti tindakan

Pada titik ini, ViewController.swift Anda akan terlihat mirip dengan Listing 7-10.

Listing 7-10. Kode Tindakan di ViewController.swift

```
import U]IKit
class ViewController: UIViewController {
    @IBOutlet weak var parkingSpots: UILabel!
    @IBAction func refreshParkingSpotsCount(sender: AnyObject) {

    }
    override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically
    // from a nib.
    }
    override func didReceiveMemoryWarning() {
```

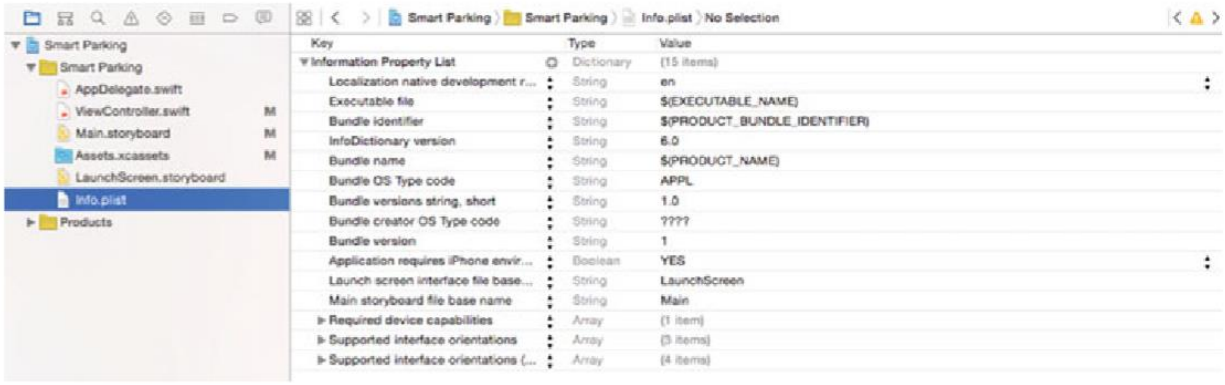
```
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
        }
    }
```

Sekarang Anda akan menambahkan kode yang perlu dijalankan sebagai respons terhadap tindakan yang diklik Tombol. Tambahkan semua kode dalam fungsi **refreshParkingSpotsCount(sender: AnyObject).** Kode yang diberikan di Listing 7-11 pertama-tama mengirimkan permintaan ke URL http://bookapps.codifythings.com/smartparking/getcount.php. Layanan respons PHP yang Anda tulis sebelumnya mengirimkan jumlah tempat parkir dalam format JSON kembali ke klien. Potongan kode berikutnya mem-parsing respons JSON ini dan mengekstrak nilai hitungan menggunakan kunci PARKING_SPOTS_COUNT. Terakhir, ia mengupdate label ParkingSpots dengan jumlah tempat parkir terbuka yang diUpdate.

Listing 7-11. Kode Lengkap untuk ViewController

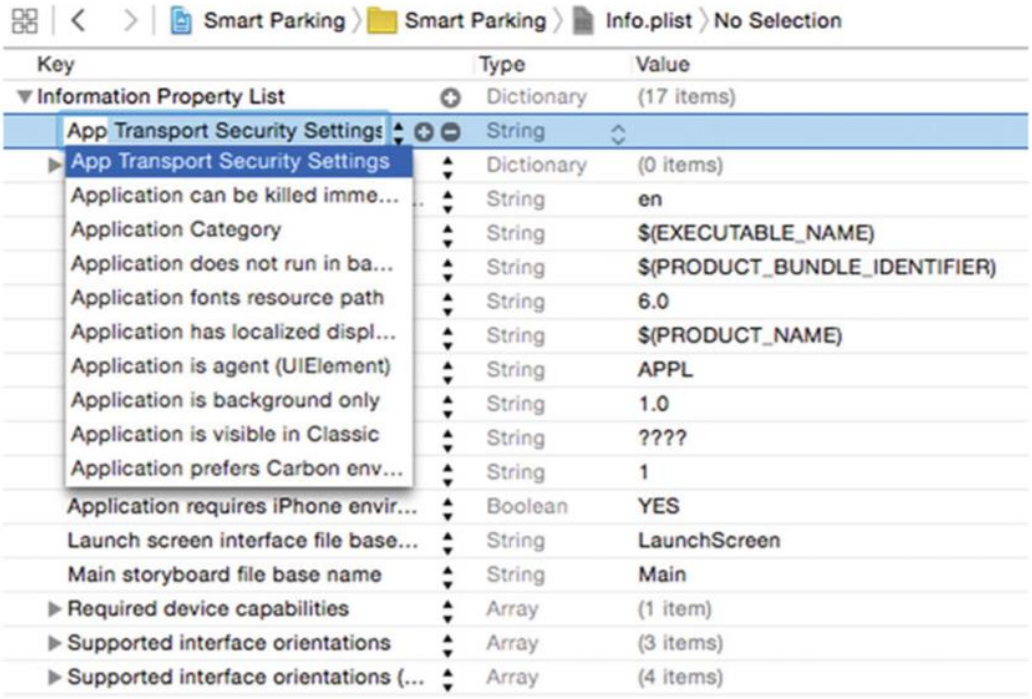
```
    import UIKit
    class ViewController: UIViewController {
        @IBOutlet weak var parkingSpots: UILabel!
        @IBAction func refreshParkingSpotsCount(sender: AnyObject) {
        let url = NSURL(string: "http://bookapps.codifythings.com/
        smartparking/getcount.php")
        let request = NSURLRequest(URL: url!)
        NSURLConnection.sendAsynchronousRequest(request, queue:
        NSOperationQueue.mainQueue()) {(response, data, error) in
        let jsonResponse: NSDictionary!=(try!
        NSJSONSerialization. JSONObjectWithData(data!, options:
        NSJSONReadingOptions.MutableContainers)) as! NSDictionary
        self.parkingSpots.text = "Open Parking Spots: " +
        String(jsonResponse["PARKING_SPOTS_COUNT"]!)
        }
        }
        override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view,
        typically from a nib.
        }
        override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
        }
    }
```

Sebelum aplikasi iOS Anda dapat melakukan panggilan Internet, Anda perlu menambahkan properti di **Info.plist.** Seperti yang ditunjukkan pada Gambar 7-32, klik + pada Listing Properti Informasi induk tingkat atas.



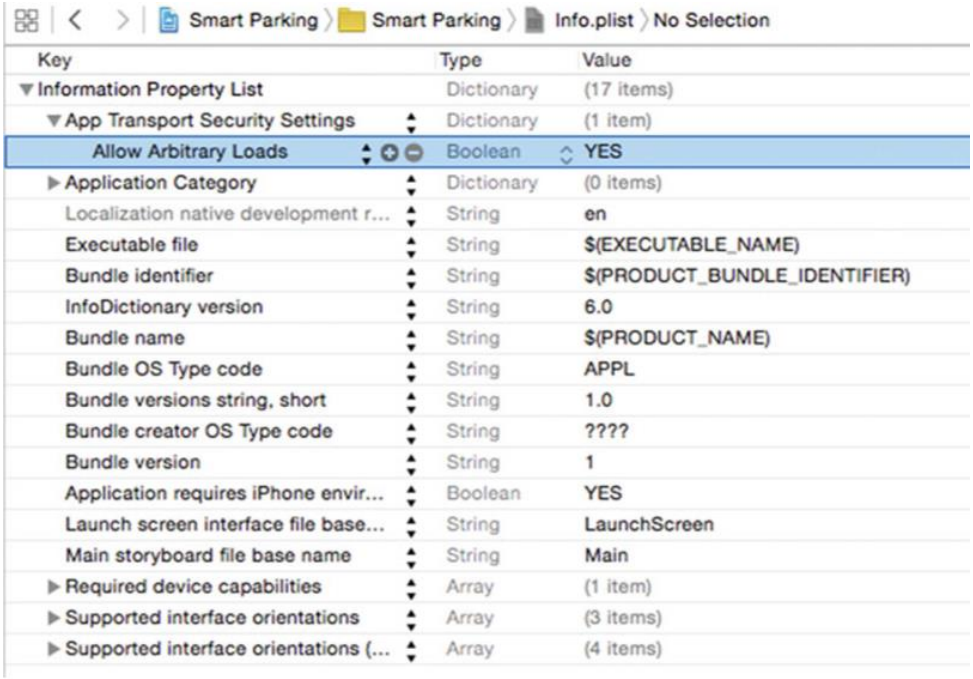
Gambar 7-32. Info.plist Listing properti

Properti baru akan ditambahkan ke Listing. Seperti yang ditunjukkan pada Gambar 7-33, pilih **App Transport Security Settings**.

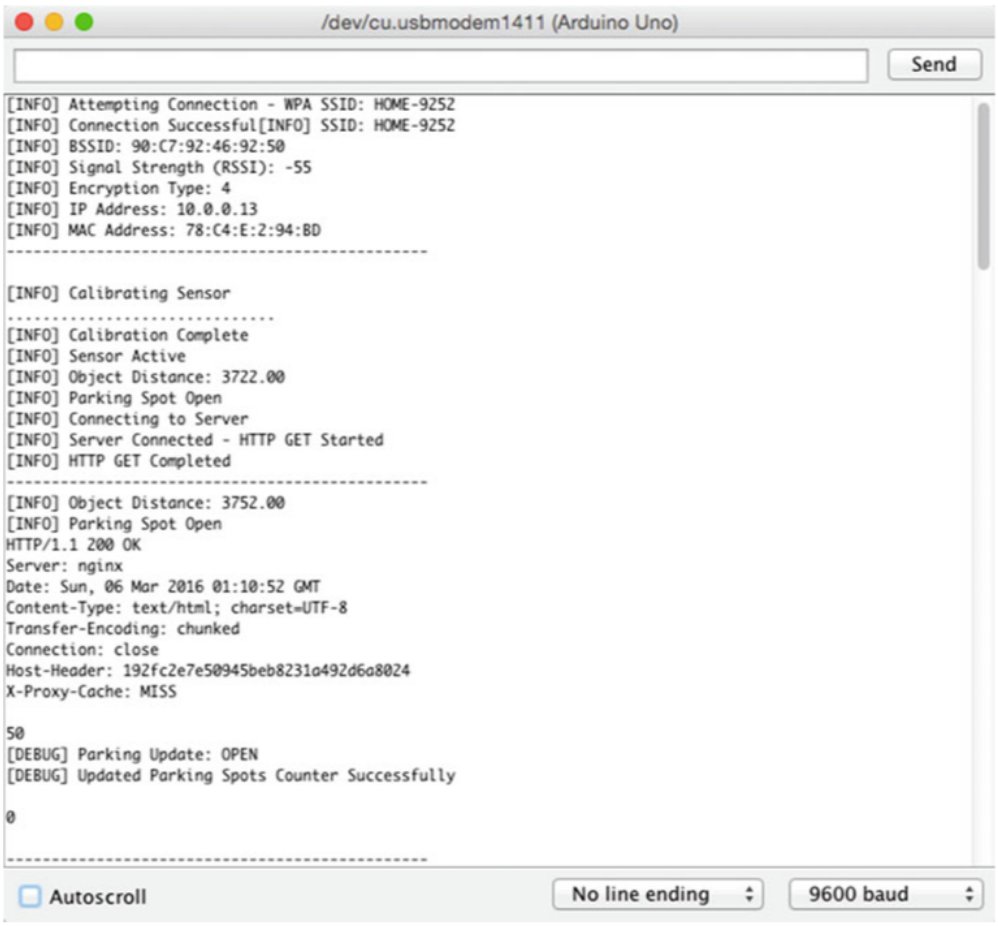


Gambar 7-33. Pilih properti Pengaturan Keamanan Transportasi Aplikasi

Klik + di properti **App Transport Security Settings** yang baru ditambahkan, seperti yang ditunjukkan pada Gambar 7-34. Ini akan menambahkan properti anak. Pilih properti **Allow Arbitrary Loads** dari Listing dan ubah nilainya dari **NO** menjadi **YES**.



Gambar 7-34. Pilih properti Allow Arbitrary Loads



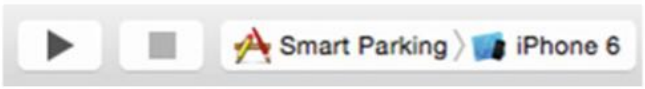
Gambar 7-35. Catat pesan dari smarter parking system

Ini menyelesaikan implementasi aplikasi iOS Anda.

7.8 PRODUK AKHIR

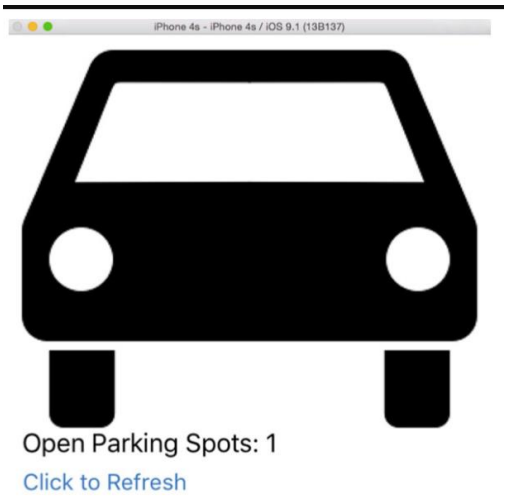
Untuk menguji aplikasi, pastikan server MySQL dan PHP Anda aktif dan berjalan dengan kode yang diterapkan. Juga verifikasi dan Upload kode Arduino seperti yang dibahas dalam Bab 1. Pastikan awalnya tidak ada objek di depan sensor jarak Anda. Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log yang serupa dengan yang ditunjukkan pada Gambar 7-35.

Selanjutnya, buka aplikasi iOS Anda di Xcode. Klik tombol Putar dari menu yang terlihat di sisi kiri atas papan cerita yang ditunjukkan pada Gambar 7-36 untuk meluncurkan aplikasi Anda dalam simulator.



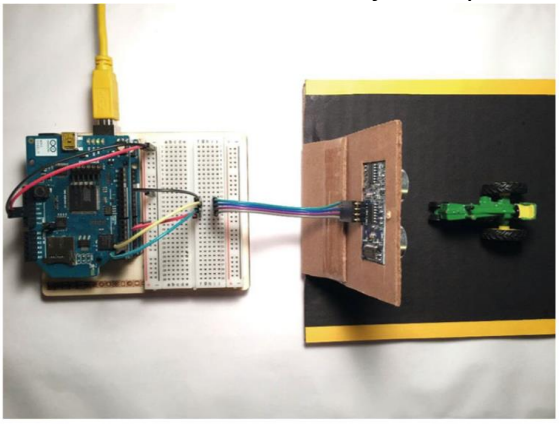
Gambar 7-36. Menu simulasi layar S

Setelah aplikasi diluncurkan, klik tombol Klik untuk Refresh untuk mengambil hitungan terbaru dari tempat parkir terbuka. Gambar 7-37 menunjukkan tampilan layar di simulator. Layar akan menampilkan 1 karena Anda belum meletakkan objek apa pun di depan sensor proximity.

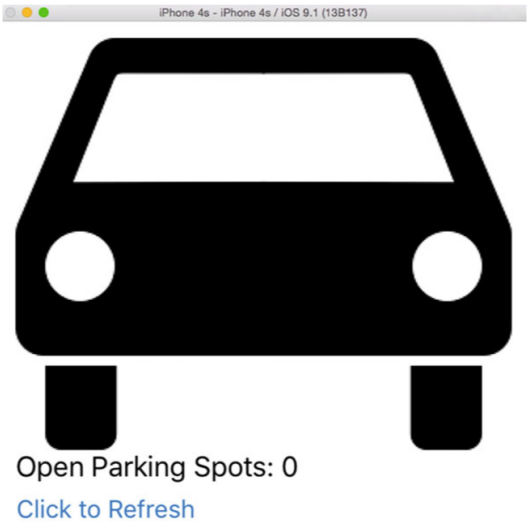


Gambar 7-37. Layar aplikasi dalam simulator dengan satu tempat terbuka

Seperti yang ditunjukkan pada Gambar 7-38, letakkan objek di depan sensor jarak Anda. Segera setelah Arduino Anda mengirimkan permintaan HTTP berikutnya ke server, hitungannya akan berubah. Klik tombol Segarkan di aplikasi iOS Anda. Seperti yang ditunjukkan pada Gambar 7-39, ini tidak akan menunjukkan pot s terbuka.



Gambar 7-38. Objek di depan sensor jarak



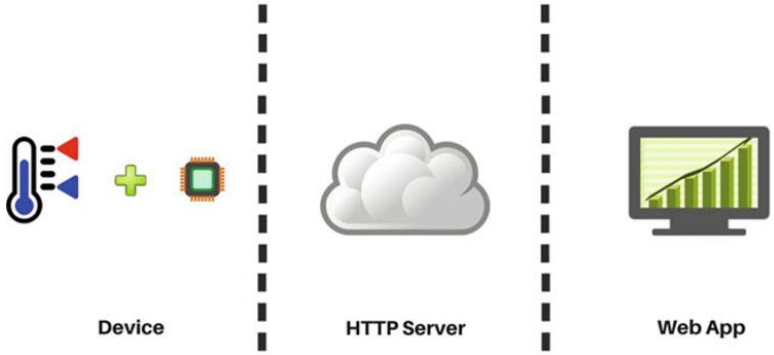
Gambar 7-39. Layar aplikasi dalam simulator tanpa pot s terbuka

7.9 RINGKASAN

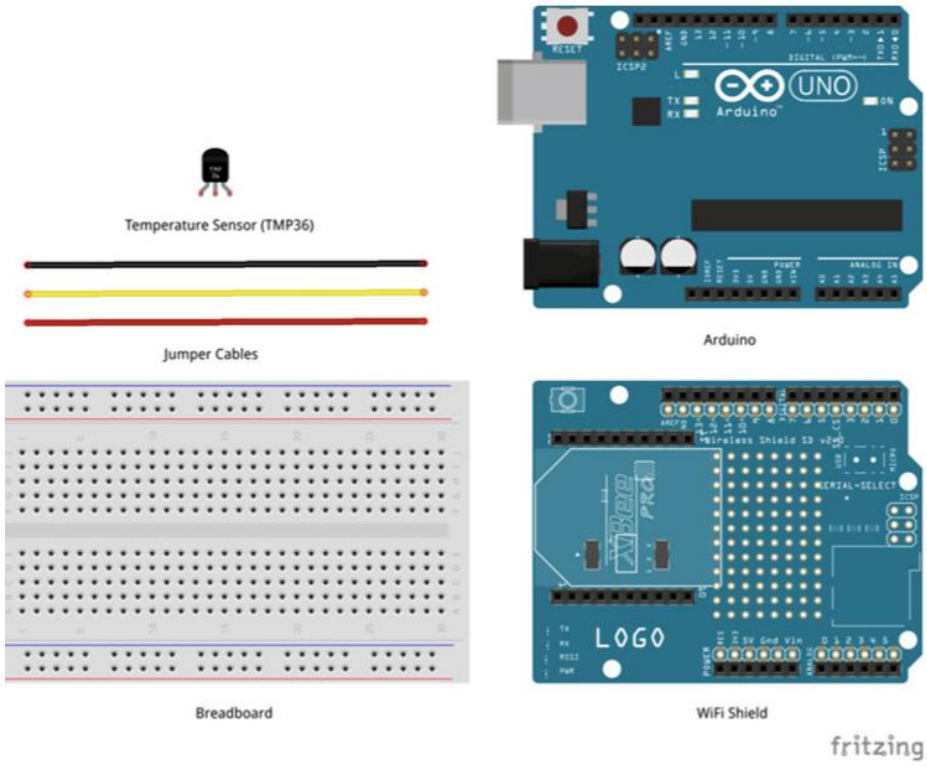
Dalam bab ini, Anda telah mempelajari tentang pola permintaan aplikasi IoT. Pola ini cocok jika pengguna Anda tidak diharuskan untuk diberi data dan sebaliknya hanya diberikan data terbaru saat mereka memintanya. Smarter parking system yang Anda bangun dalam bab ini adalah contoh yang baik dari pola ini, karena pengguna hanya memperhatikan informasi tempat parkir terbuka saat mereka mencari tempat parkir; jika tidak, mereka tidak peduli.

BAB 8
POLA IoT: APLIKASI WEB

Karena sebagian besar tugas dan interaksi kita sehari-hari berpindah ke perangkat seluler, aplikasi web akan tetap memiliki tempat. Di ruang IoT, mereka terutama akan digunakan untuk memantau dan mengendalikan implementasi skala besar. Dalam bab ini, Anda akan membangun sistem pemantauan suhu. Gambar 8-1 menunjukkan diagram tingkat tinggi dari semua komponen yang terlibat dalam sistem ini. Komponen pertama adalah perangkat Arduino yang mengumpulkan data suhu dan mempublishkannya ke server menggunakan permintaan HTTP. Komponen kedua adalah server yang menerima data suhu dan menyimpannya dalam database. Komponen terakhir mengakses data suhu dari server dan menyajikannya kepada pengguna di dasbor analitik berbasis web. Dasbor analitik berbasis web ini akan berada di server juga.



Gambar 8-1. Komponen sistem pemantauan suhu



Gambar 8-2. Hardware yang diperlukan untuk sistem pemantauan suhu

8.1 TUJUAN PEMBELAJARAN

Pada akhir bab ini, Anda akan dapat:

- Membaca data dari sensor suhu
- Publish data sensor ke server
- Menampilkan data sensor di dasbor berbasis web

Hardware yang Diperlukan

Gambar 8-2 memberikan Listing semua komponen hardware yang diperlukan untuk membangun sistem pemantauan suhu ini.

Software yang Diperlukan

Untuk mengembangkan sistem pemantauan suhu, Anda memerlukan software berikut:

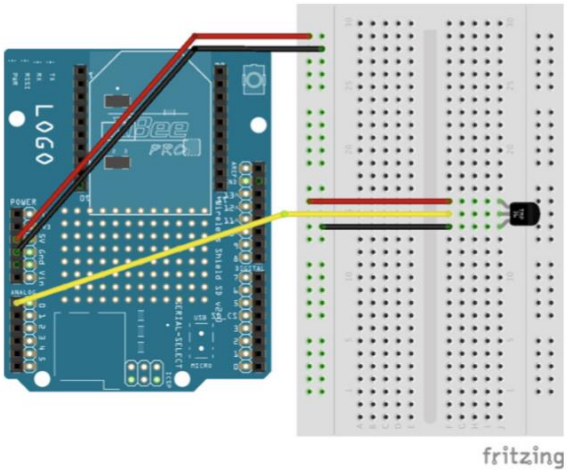
- Arduino IDE 1.6.4 atau lebih baru
- Server PHP (diinstal atau di-host)
- Server MySQL (diinstal atau dihosting)
- Editor teks

Sirkuit

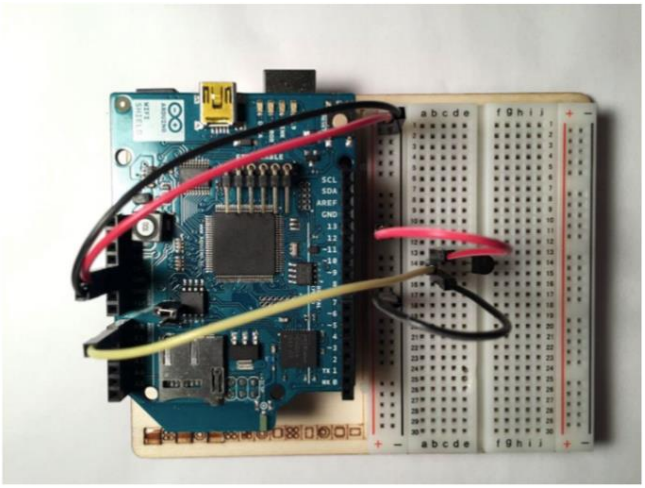
Pada bagian ini, Anda akan membangun sirkuit yang diperlukan untuk sistem pemantauan suhu. Rangkaian ini menggunakan sensor suhu TMP36 yang murah dan mudah digunakan. Sensor mengembalikan nilainya dalam tegangan, yang diubah menjadi Celcius dan Fahrenheit.

1. Pastikan Arduino tidak terhubung ke sumber listrik, seperti ke komputer melalui USB atau baterai.
2. Pasang pelindung WiFi ke bagian atas Arduino. Semua pin harus sejajar.
3. Gunakan kabel jumper untuk menghubungkan port power (5V) dan ground (GND) pada Arduino ke port power (+) dan ground (-) pada breadboard.
4. Sekarang papan breadboard Anda memiliki sumber daya, gunakan kabel jumper untuk menghubungkan port daya (+) dan arde (-) papan breadboard Anda ke port daya dan arde dari sensor suhu. Pin kiri dari sensor adalah power (+) dan pin kanan adalah ground (-).
5. Untuk membaca nilai dari sensor suhu, Anda perlu menghubungkan kabel jumper dari port tegangan analog (pin tengah) sensor suhu ke port A0 (Analog) Arduino. Kode Anda akan membaca tegangan dari port ini untuk menghitung suhu dalam Celcius dan Fahrenheit.

Sirkuit Anda sekarang telah selesai dan akan terlihat seperti Gambar 8-3 dan 8-4.



Gambar 8-3. Diagram sirkuit sistem pemantauan suhu



Gambar 8-4. Sirkuit sebenarnya dari sistem pemantauan suhu

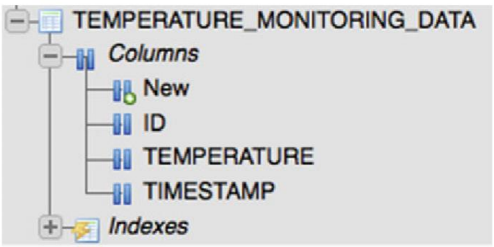
8.2 TABEL DATABASE (MySQL)

Seperti yang telah dibahas pada bab sebelumnya, sebelum Anda dapat mengirim permintaan HTTP dari Arduino, Anda perlu membangun layanan yang akan menerima data. Bab ini juga menggunakan MySQL sebagai database. Aplikasi ini membutuhkan tabel tiga kolom yang sangat sederhana. Jadi buat tabel baru bernama **TEMPERATURE_MONITORING_DATA** menggunakan skrip SQL yang disediakan di Listing 8-1. Jalankan skrip ini di database yang ada atau buat yang baru. Kolom pertama akan menjadi ID yang dibuat secara otomatis, kolom kedua akan menjadi stempel waktu yang dibuat secara otomatis, dan kolom ketiga akan digunakan untuk menyimpan pembacaan suhu.

Listing 8-1. Buat Tabel SQL

```
CREATE TABLE `TEMPERATURE_MONITORING_DATA`
    (
    `ID` int(11) NOT NULL AUTO_INCREMENT,
    `TIMESTAMP` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
    CURRENT_TIMESTAMP,
    `TEMPERATURE` double NOT NULL,
     PRIMARY KEY (`ID`)
)
```

Gambar 8-5 menunjukkan struktur tabel **TEMPERATURE_MONITORING_DATA.**



Gambar 8-5. Struktur tabel DATA PEMANTAUAN SUHU

Kode (PHP)

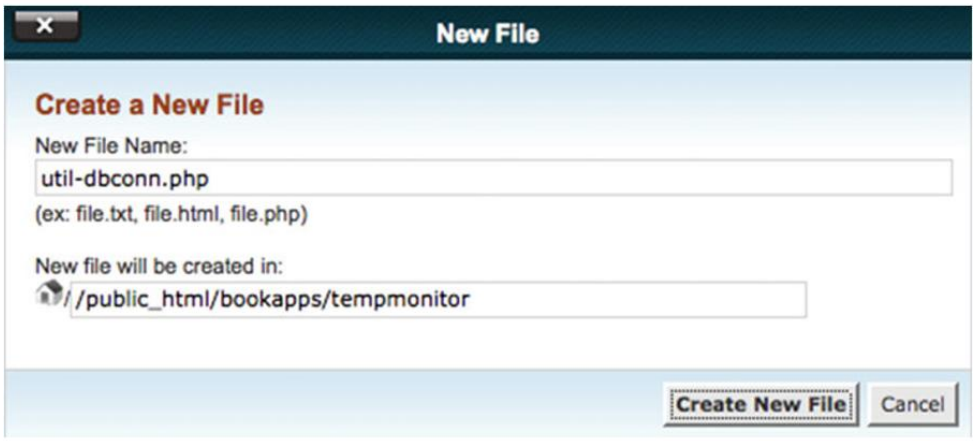
Sekarang tabel database sudah siap, Anda perlu membangun dua layanan. Layanan pertama akan menerima data sensor Arduino dan menyimpannya di tabel database yang baru dibuat. Layanan kedua akan menampilkan data sensor historis di dasbor. Proyek ini juga menggunakan PHP untuk membangun penyimpanan data dan layanan antarmuka. Buat folder baru bernama tempmonitor di folder publik/root server PHP Anda. Semua kode

sumber PHP untuk proyek ini akan masuk ke folder tempmonitor ini. Mulai editor teks pilihan Anda.

Catatan:*Semua kode PHP dikembangkan menggunakan Brackets, yang merupakan editor teks open source. Lihat http://brackets.io/ untuk informasi lebih lanjut.*

8.3 KONEKSI DATABASE

Skrip PHP untuk menyimpan dan menampilkan data harus terhubung ke database. Seperti yang ditunjukkan pada Gambar 8-6, buat file baru bernama util-dbconn.php di folder tempmonitor. File ini akan digunakan oleh kedua skrip alih-alih mengulangi kode.



Gambar 8-6. File konektivitas database umum yang disebut util-dbconn.php

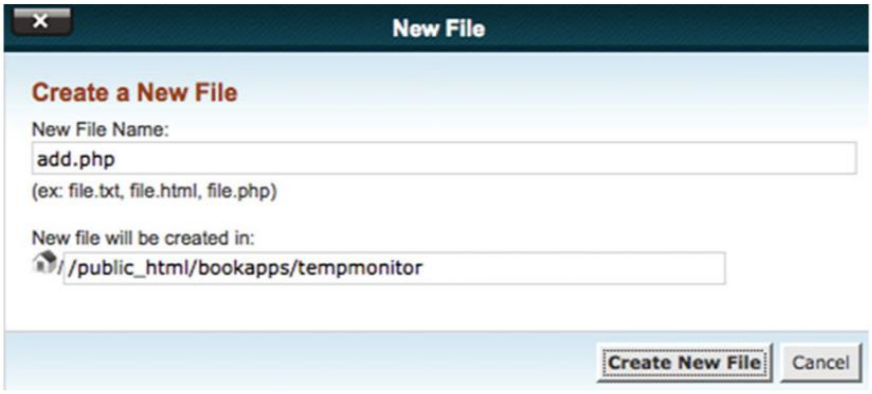
Buka file dalam editor teks dan salin atau ketik kode dari Listing 8-2. Seperti yang Anda lihat, tidak banyak kode dalam file ini. Empat variabel **$servername, $username, $password, dan $dbname** berisi informasi koneksi. Buat koneksi baru dengan melewatkan keempat variabel ini dan simpan referensi koneksi di variabel $mysqli. Kondisi IF dalam kode hanya memeriksa kesalahan selama upaya koneksi dan mencetaknya jika ada.

Listing 8-2. Kode Konektivitas Database Umum util-dbconn.php

```
<?php
    $servername = "SERVER_NAME";
    $dbname = "DB_NAME";
    $username = "DB_USERNAME";
    $password = "DB_PASSWORD";
    //Open a new connection to MySQL server
    $mysqli = new mysqli($servername, $username, $password, $dbname);
    //Output connection errors
    if ($mysqli->connect_error)
    {
    die("[ERROR] Connection Failed: ". $mysqli->connect_error);
    }
?>
```

8.4 MENERIMA DAN MENYIMPAN DATA SENSOR

Seperti yang ditunjukkan pada Gambar 8-7, buat file baru bernama **add.php** di folder tempmonitor. Skrip ini akan melakukan dua tugas—pertama akan mengambil informasi dari permintaan HTTP dan kemudian akan menyisipkan informasi ini sebagai baris baru di tabel database.



Gambar 8-7. File untuk menerima dan menyimpan data di add. php

Buka file yang baru dibuat dalam editor teks dan salin atau ketik kode yang disediakan di Listing 8-3. Seperti disebutkan pada langkah sebelumnya, untuk menyimpan data, koneksi database perlu dibuat. Anda membuat util-dbconn.php untuk melakukan tugas itu, jadi dalam file ini Anda perlu menyertakan util-dbconn.php. File util-dbconn.php menyediakan akses ke variabel $mysqli, yang berisi referensi koneksi dan akan digunakan untuk menjalankan kueri SQL.

Contoh dalam buku ini di-host di http://bookapps.codifythings.com/tempmonitor, dan Arduino akan mengirimkan data suhu ke add.php menggunakan metode HTTP GET. Seperti yang dibahas dalam Bab 2, HTTP GET menggunakan string kueri untuk mengirim data permintaan. Jadi, URL lengkap dengan string kueri yang akan digunakan Arduino menjadi http://bookapps.codifythings.com/tempmonitor/add.php?temperature=79.5. Kode PHP Anda perlu mengekstrak nilai suhu dari string kueri menggunakan pernyataan $_GET['temperature'].

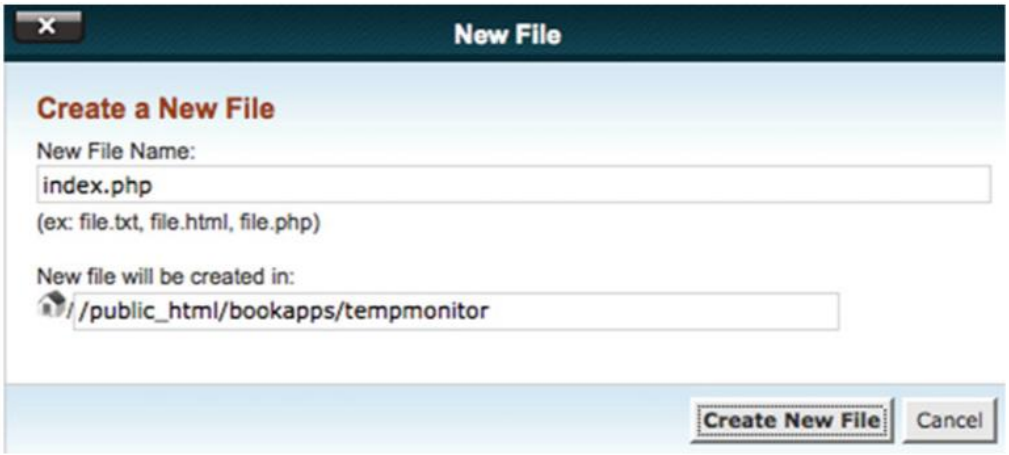
Sekarang Anda perlu menyimpan nilai suhu ini di tabel database sebagai baris baru. Siapkan pernyataan INSERT SQL dalam variabel $sql. Anda hanya perlu memberikan nilai suhu, karena ID dan TIMESTAMP keduanya dibuat secara otomatis, sehingga database akan menanganinya untuk Anda. Terakhir, jalankan pernyataan INSERT SQL menggunakan $mysqli->query($sql) dan periksa variabel $result apakah berhasil atau gagal.

Listing 8-3. Kode untuk Menerima dan Menyimpan Data di add.php

```
<?php
    include('util-dbconn.php');
    $temperature = $_GET['temperature'];
    echo "[DEBUG] Temperature Sensor Data: ". $temperature. "\n";
    $sql = "INSERT INTO `TEMPERATURE_MONITORING_DATA`(`TEMPERATURE`)
    VALUES ($temperature)";
    if (!$result = $mysqli->query($sql))
    {
    echo "[Error] ". mysqli_error(). "\n";
    exit();
    }
    $mysqli->close();
    echo "[DEBUG] New Temperature Sensor Data Added Successfully\n";
?>
```

8.5 DASBOR

Semua data yang ditangkap oleh sensor dan disimpan dalam database tidak terlihat oleh siapa pun. Jadi selanjutnya, Anda akan membangun dasbor analitik yang akan memuat 30 entri terakhir dari database dan menampilkannya dalam format diagram batang. Seperti yang ditunjukkan pada Gambar 8-8, buat file baru bernama **index.php** di folder tempmonitor.



Gambar 8-8. File untuk dashboard analytics adalah index.php

Listing 8-4 menyediakan struktur file index.php. Strukturnya adalah HTML standar, jadi dalam tag <head> Anda akan memuat data dari tabel database, memuat dependensi, dan menginisialisasi bagan. Tag <body> hanya akan menampilkan grafik.

Listing 8-4. Struktur Kode untuk Dasbor Analytics di index.php

```
<html>
        <head>
        <title>...</title>
        <?php
        ?>
        <script src="..." />
        <script>
        // chart customization code
        </script>
        </head>
        <body>
        // chart display
        </body>
</html>
```

Listing 8-5 menyediakan kode lengkap untuk index.php, jadi salin atau tulis kode di index.php. Untuk mengembangkan diagram batang, Anda akan menggunakan Dojo, yang merupakan toolkit JavaScript yang sangat populer. Anda tidak perlu mengunduh atau menginstal kode apa pun. Toolkit ini dapat diakses melalui Internet sehingga tag skrip Anda hanya perlu mengarah ke *//ajax.googleapis.com/ajax/libs/dojo/1.10.4/dojo/dojo.js* sebagai sumbernya.

Untuk mengisi bagan, Anda harus terlebih dahulu memuat data dari database dalam variabel array yang disebut chartData. Pada tag <script>, tambahkan kode PHP untuk memuat data dari tabel database. Sertakan util-dbconn.php karena koneksi database perlu

dibuat, lalu siapkan pernyataan SQL SELECT. Jalankan kueri dan siapkan array dari hasilnya. Format akhir larik harus serupa dengan var chartData = [Val1, Val2, Val3].

Untuk menggunakan sumber daya toolkit Dojo, Anda perlu memuat semua dependensi menggunakan require(). Untuk pengembangan bagan, dua dependensi yang paling penting adalah sumber daya bagan dojox/charting/Chart dan tema dojox/charting/themes/PlotKit/orange. Dependensi yang tersisa disertakan untuk menyesuaikan bagan.

Di dalam function(Chart, theme){...}, buat bagan baru, atur temanya, sesuaikan area plot dan sumbu x/y, tambahkan seri chartData ke bagan. Terakhir, render grafiknya. Tag <body> memiliki kode untuk menampilkan judul di atas halaman dan bagan yang dibuat sebelumnya.

Listing 8-5. Kode Lengkap untuk Dasbor Analytics di index.php

```
<html lang="en">
<head>
        <title>Temperature Monitoring System - Dashboard</title>
        <script src="//ajax.googleapis.com/ajax/libs/dojo/1.10.4/dojo/dojo.js"
                data-dojo-config="async: true"></script>
        <script>
        <?php
        include('util-dbconn.php');
        $sql = "SELECT * FROM (SELECT * FROM
                `TEMPERATURE_MONITORING_DATA`
                ORDER BY ID DESC LIMIT 30) sub ORDER BY id ASC";
        $result = $mysqli->query($sql);
        $resultCount = $result->num_rows;
        if ($resultCount > 0)
        {
        $currentRow = 0;
        echo "var chartData = [";
        // output data of each row
        while($row = $result->fetch_assoc())
        {
                        $currentRow = $currentRow + 1;
                echo $row["TEMPERATURE"];
                if($currentRow < $resultCount)
                {
                echo ",";
                }
                }
                echo "];";
                }
        Else
        {
        echo "0 results";
        }
        $mysqli->close();
        ?>
                require([
```

```
                "dojox/charting/Chart",
                "dojox/charting/themes/PlotKit/orange",
                "dojox/charting/plot2d/Columns",
                "dojox/charting/plot2d/Markers",
                "dojox/charting/axis2d/Default",
                "dojo/domReady!"
                ], function(Chart, theme) {
                        var chart = new Chart("chartNode");
                        chart.setTheme(theme);
                        chart.addPlot("default", {type: "Columns",
                                gap: 5, labels: true,
                                labelStyle: "outside"});
                        chart.addAxis("x", {title: "Readings (#)",
                                titleOrientation: "away"});
                        chart.addAxis("y", {title: "Temperature (F)",
                                titleOrientation: "axis", min: 0,
                                max: 270, vertical: true, fixLower: "major",
                                fixUpper: "major" });
                        chart.addSeries("TemperatureData",chartData);
                        chart.render();
                });
        </script>
</head>
<body style="background-color: #F5EEE6">
        <div style="align: center;">
        <font size="5px">
                Temperature Monitoring System – Dashboard
                </font></div>
        <div id="chartNode" style="width: 100%; height: 50%; margin-top: 50px;">
        </div>
        <script type="text/javascript">
                init();
        </script>
</body>
</html>
```

8.6 KODE (ARDUINO)

Komponen akhir dari proyek ini adalah kode Arduino untuk menghubungkan ke Internet menggunakan WiFi, membaca data dari sensor suhu, dan memublikasikannya ke server. Mulai Arduino IDE Anda dan ketik kode yang disediakan di sini atau unduh dari situs buku dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi lima bagian.

- Library eksternal
- Konektivitas Internet (Wi-Fi)
- Baca data sensor
- HTTP (terbitkan)
- Fungsi standar

Library Eksternal

Bagian pertama dari kode, seperti yang disediakan dalam Listing 8-6, mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Karena Anda terhubung ke Internet secara nirkabel, ketergantungan utama kode ada pada <WiFi.h>.

Listing 8-6. Kode untuk Menyertakan Dependensi Eksternal

```
#include <SPI.h>
#include <WiFi.h>
```

Konektivitas Internet (Nirkabel)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (dalam Bab 2) di sini.

Baca Data Sensor

Bagian ketiga dari kode, seperti yang disediakan dalam Listing 8-7, mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membaca data sensor. Fungsi readSensorData() membaca data dari Analog Pin A0 dan hasilnya antara 0 dan 1023. Semakin besar nilai yang dikembalikan, semakin tinggi suhunya. Nilai sensor tidak secara langsung memberikan suhu dalam Celcius atau Fahrenheit, jadi rumus, seperti yang disorot dalam Listing 8-7, digunakan untuk mengubah nilai sensor ke dalam format yang diperlukan.

Listing 8-7. Kode untuk Membaca Suhu

```
int TEMP_SENSOR_PIN = A0;
float temperatureC = 0.0;
float temperatureF = 0.0;
void readSensorData()
{
    //Read Temperature Sensor Value
    int temperatureSensorValue = analogRead(TEMP_SENSOR_PIN);
    float voltage = temperatureSensorValue * 5.0 / 1024;
     //Converting reading to Celsius
    temperatureC = (voltage - 0.5) * 100;
     //Converting reading to Fahrenheit
    temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
     //Log Sensor Data on Serial Monitor
    Serial.print("[INFO] Temperature Sensor Reading (F): ");
    Serial.println(temperatureF);
 }
```

Penerbitan Data Bagian keempat dari kode seperti yang disediakan dalam Listing 8-8 mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membuat dan mengirim permintaan HTTP ke server. Kode ini adalah versi HTTP GET yang sedikit dimodifikasi yang Anda kembangkan di Bab 3.

Modifikasi utama dalam kode ini adalah kemampuannya untuk membuka dan menutup koneksi ke server secara berulang. Selain itu, pastikan untuk mengubah nilai server dan port ke nilai server PHP Anda, variabel requestData, dan nilai URL.

Listing 8-8. Kode untuk Mengirim Permintaan HTTP

```
//IP address of the server
char server[] = {"bookapps.codifythings.com"};
int port = 80;
unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 10L * 1000L;
```

```
void transmitSensorData()
{
        // Read all incoming data (if any)
        while (client.available())
        {
        char c = client.read();
        Serial.write(c);
        }
        if (millis() - lastConnectionTime > postingInterval)
        {
        client.stop();
        Serial.println("[INFO] Connecting to Server");
        String requestData = "temperature=" + String(temperatureF);
        // Prepare data or parameters that need to be posted to server
        if (client.connect(server, port))
        {
        Serial.println("[INFO] Server Connected - HTTP GET Started");
        // Make a HTTP request:
        client.println("GET /tempmonitor/add.php?" + requestData +
  " HTTP/1.1");
        client.println("Host: " + String(server));
        client.println("Connection: close");
        client.println();
        lastConnectionTime = millis();
        Serial.println("[INFO] HTTP GET Completed");
        }
        Else
        {
        // Connection to server:port failed
        Serial.println("[ERROR] Connection Failed");
        }
}
Serial.println("-----------------------------------------");
}
```

Fungsi Standar

Kode di bagian terakhir disediakan di Listing 8-9. Ini mengimplementasikan fungsi setup() dan loop() standar Arduino. Fungsi setup() menginisialisasi port serial dan menghubungkan ke Internet. Fungsi loop() memanggil readSensorData() untuk membaca data suhu dan kemudian memublikasikan data ke server menggunakan HTTP dengan memanggil transmitSensorData() secara berkala.

Listing 8-9. Kode untuk Fungsi Arduino Standar

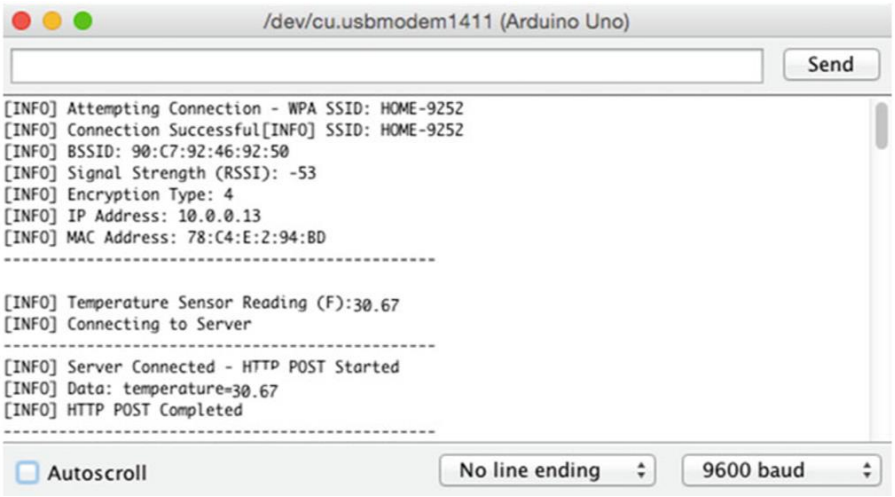
```
void setup()
{
    // Initialize serial port
    Serial.begin(9600);
    //Connect Arduino to internet
    connectToInternet();
}
```

```
void loop()
{
// Read sensor data
readSensorData();
// Transmit sensor data    transmitSensorData();
// Delay
delay(6000);
}
```

Kode Arduino Anda sekarang sudah lengkap

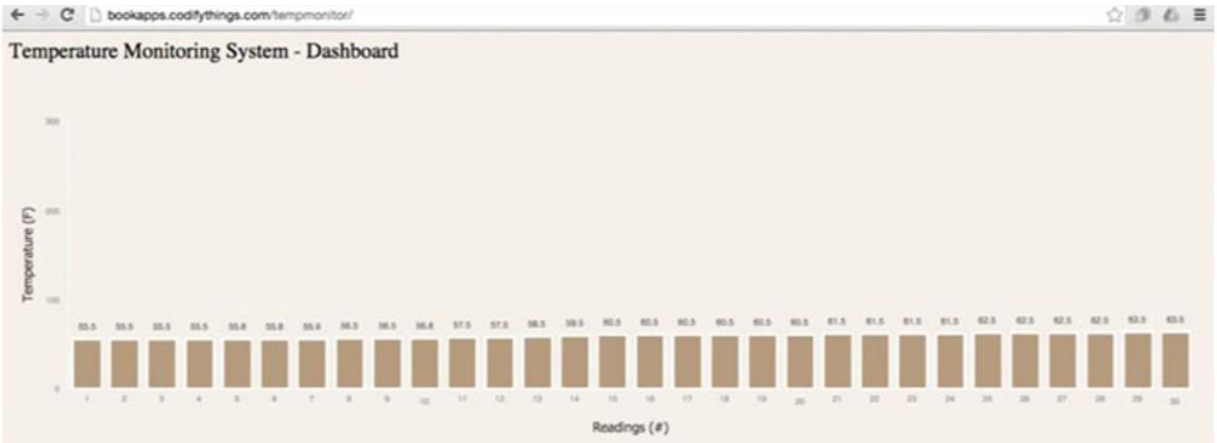
8.7 PRODUK AKHIR

Untuk menguji aplikasi, pastikan server MySQL dan PHP Anda aktif dan berjalan dengan kode yang diterapkan. Juga verifikasi dan Upload kode Arduino seperti yang dibahas dalam Bab 1. Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log yang mirip dengan yang ditunjukkan pada Gambar 8-9.



Gambar 8-9. Mencatat pesan dari sistem pemantauan suhu

Biarkan Arduino Anda berjalan selama beberapa menit sehingga cukup data yang dikirim ke server. Periksa dasbor Anda dengan mengakses URL proyek, dalam hal ini http://bookapps.codifythings.com/tempmonitor. Dasbor Anda akan terlihat seperti Gambar 8-10.



Gambar 8-10. Dashboard sistem pemantauan suhu

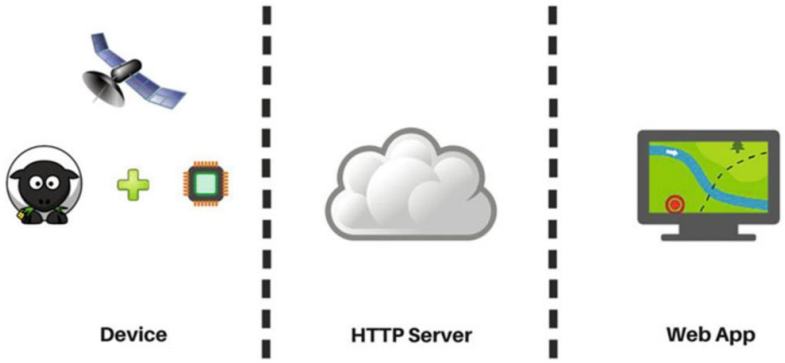
8.8 RINGKASAN

Dalam bab ini, Anda telah mempelajari tentang membuat aplikasi web kustom. Aplikasi web banyak digunakan untuk memantau aplikasi IoT dan implementasi skala besar, serta untuk membuat dasbor.

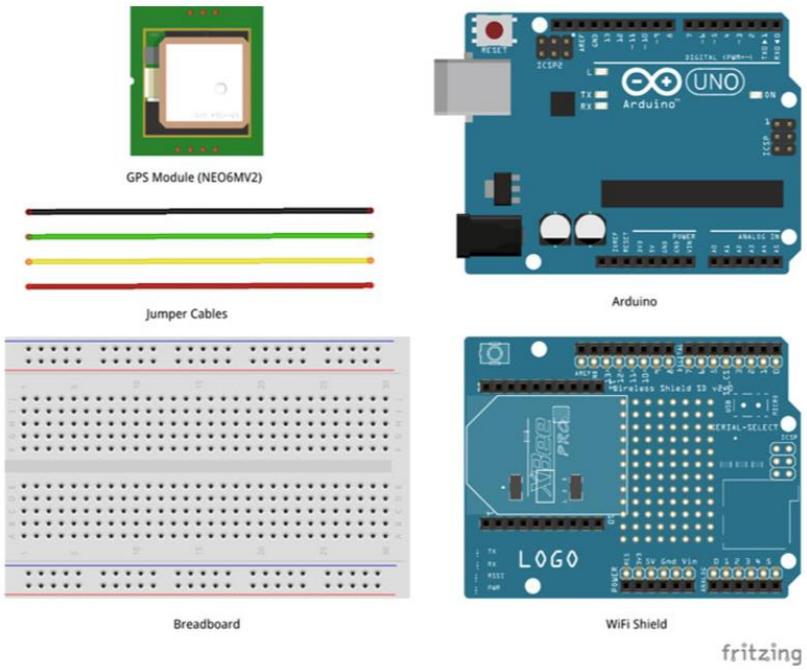
BAB 9
POLA IoT: LOCATION AWARE

Perangkat yang sadar lokasi akan menjadi salah satu kontributor penghematan terbesar dari implementasi IoT. Pola IoT terlihat dalam berbagai jenis skenario, termasuk perencanaan rute yang optimal, pelacakan satwa liar yang terancam punah, dan penentuan lokasi kecelakaan.

Dalam bab ini, Anda akan membangun sistem pelacakan ternak. Gambar 9-1 menunjukkan diagram tingkat tinggi dari semua komponen yang terlibat dalam sistem ini. Komponen pertama adalah perangkat Arduino yang menangkap koordinat saat ini dan mempublishkannya ke server menggunakan permintaan HTTP. Komponen kedua adalah server yang menerima koordinat GPS dan menyimpannya dalam database. Komponen terakhir adalah halaman web yang menunjukkan koordinat GPS yang tersimpan di peta. Halaman web ini juga berada di server.



Gambar 9-1. Komponen sistem pelacakan ternak



Gambar 9-2. Hardware yang diperlukan untuk sistem pelacakan ternak

Untuk tujuan proyek ini, Anda hanya akan melacak satu hewan.

9.1 TUJUAN PEMBELAJARAN

Di akhir bab ini, Anda akan dapat:

- Membaca koordinat GPS
- Publish koordinat GPS ke server
- Menampilkan koordinat GPS di peta

Hardware yang Diperlukan

Gambar 9-2 menyediakan Listing semua komponen hardware yang diperlukan untuk membangun sistem pelacakan ternak.

Software yang Diperlukan

Untuk mengembangkan sistem pelacakan ternak ini, Anda memerlukan software berikut:

- Arduino IDE 1.6.4 atau yang lebih baru
- Server PHP (diinstal atau dihosting)
- Server MySQL (diinstal atau dihosting)
- Editor teks

Sirkuit

Di bagian ini, Anda akan membangun sirkuit yang diperlukan untuk sistem pelacakan ternak. Rangkaian ini menggunakan modul GPS NEO6MV2 untuk mendapatkan data lintang dan bujur saat ini. Modul GPS memiliki akurasi posisi 5 meter.

1. Pastikan Arduino tidak terhubung ke sumber listrik, seperti ke komputer melalui USB atau baterai.
2. Pasang pelindung WiFi ke bagian atas Arduino. Semua pin harus sejajar.
3. Gunakan kabel jumper untuk menghubungkan port power (3.3V) dan ground (GND) pada Arduino ke port power (+) dan ground (-) pada breadboard.
4. Sekarang papan breadboard Anda memiliki sumber daya, gunakan kabel jumper untuk menghubungkan port daya (+) dan ground (-) ports breadboard Anda ke port daya dan pembumian GPS.
5. Untuk membaca data GPS, Anda perlu menghubungkan kabel jumper dari port RX (Receive) GPS ke Port Digital 3 Arduino. Kode Anda akan menggunakan data dari port ini untuk menemukan informasi garis lintang dan garis bujur.
6. Mirip dengan Langkah 5, Anda juga perlu menghubungkan kabel jumper dari port TX (Transmit) GPS ke Port Digital 2 Arduino. Kode Anda akan menggunakan data dari port ini untuk menemukan informasi garis lintang dan garis bujur.

***Catatan**: Modul GPS lain mungkin memiliki kebutuhan daya dan sirkuit yang berbeda. Periksa lembar data modul GPS Anda untuk memastikan persyaratannya.*

Sirkuit Anda sekarang telah selesai dan akan terlihat mirip dengan Gambar 9-3 dan 9-4.

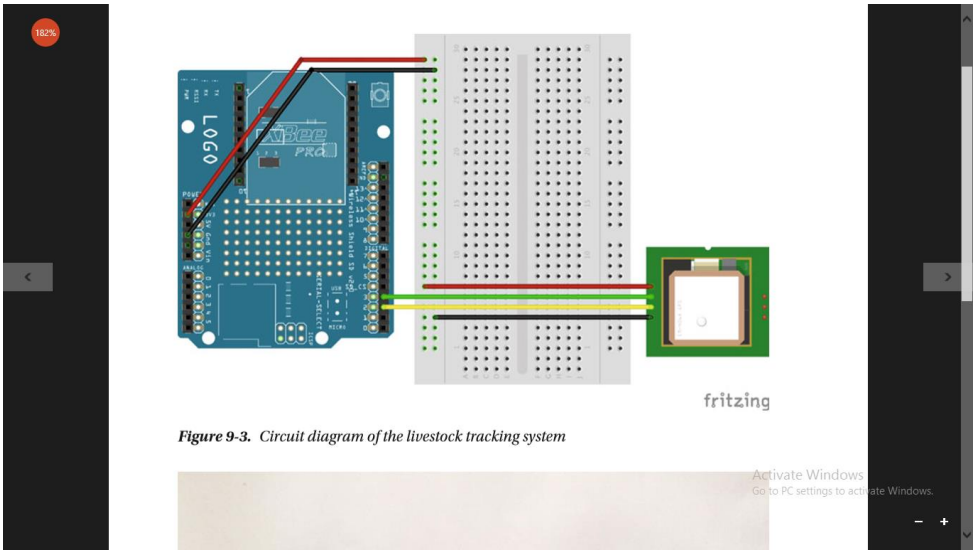
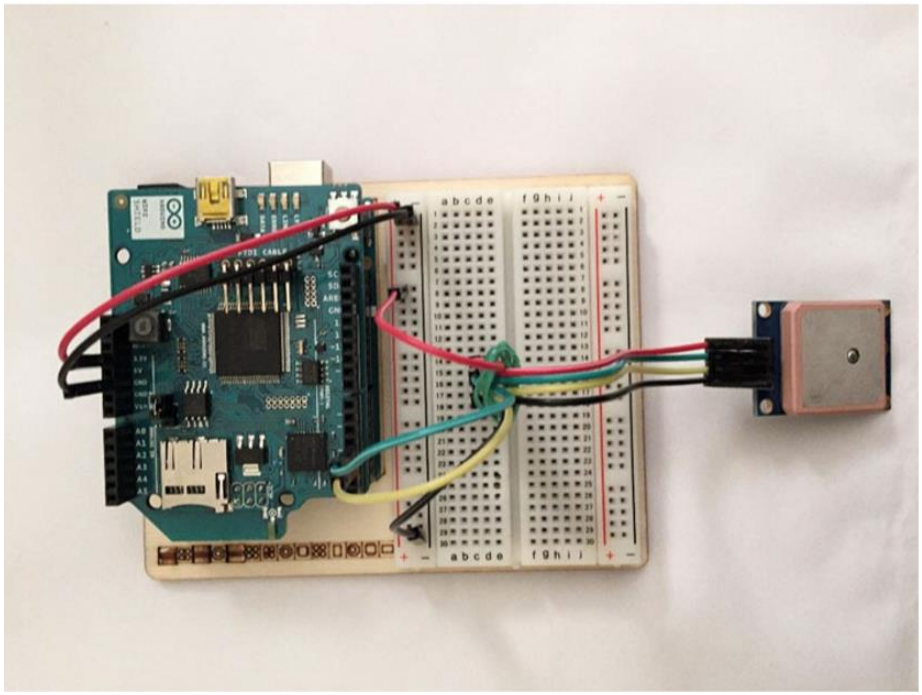


Figure 9-3. Circuit diagram of the livestock tracking system

Gambar 9-3. Diagram sirkuit dari sistem pelacakan ternak



Gambar 9-4. Sirkuit sebenarnya dari sistem pelacakan ternak

9.2 TABEL *DATABASE* (MySQL)

Seperti yang dibahas dalam dua bab sebelumnya, sebelum Anda dapat mengirim permintaan HTTP dari Arduino, Anda perlu membangun layanan yang akan menerima data. Sistem pelacakan ternak akan menampilkan koordinat GPS terbaru pada peta, sehingga Anda perlu membuat tabel database yang akan menyimpan koordinat GPS tersebut.

Bab ini juga menggunakan MySQL sebagai databasenya. Meskipun Anda hanya akan melacak satu perangkat, struktur tabel akan sama seperti jika Anda melacak beberapa perangkat. Jadi buat tabel baru bernama GPS_TRACKER_DATA menggunakan skrip SQL yang disediakan di Listing 9-1. Jalankan skrip ini di database yang ada atau buat yang baru. Kolom pertama akan menyimpan ID hewan/perangkat pengirim koordinat; kolom kedua akan menyimpan garis lintang; kolom ketiga akan menyimpan garis bujur; dan kolom keempat akan berisi stempel waktu yang dibuat secara otomatis.

Listing 9-1. Buat Tabel SQL

```
CREATE TABLE `GPS_TRACKER_DATA`
(
    `CLIENT_ID` varchar(40) NOT NULL,
    `LATITUDE` varchar(40) NOT NULL,
    `LONGITUDE` varchar(40) NOT NULL,
    `LAST_UPDATED` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
    ON UPDATE CURRENT_TIMESTAMP,
    PRIMARY KEY (`CLIENT_ID`)
)
```

Gambar 9-5 menunjukkan struktur tabel GPS_TRACKER_DATA.

Gambar 9-5. Struktur tabel GPS_TRACKER_DATA

Kode (PHP)

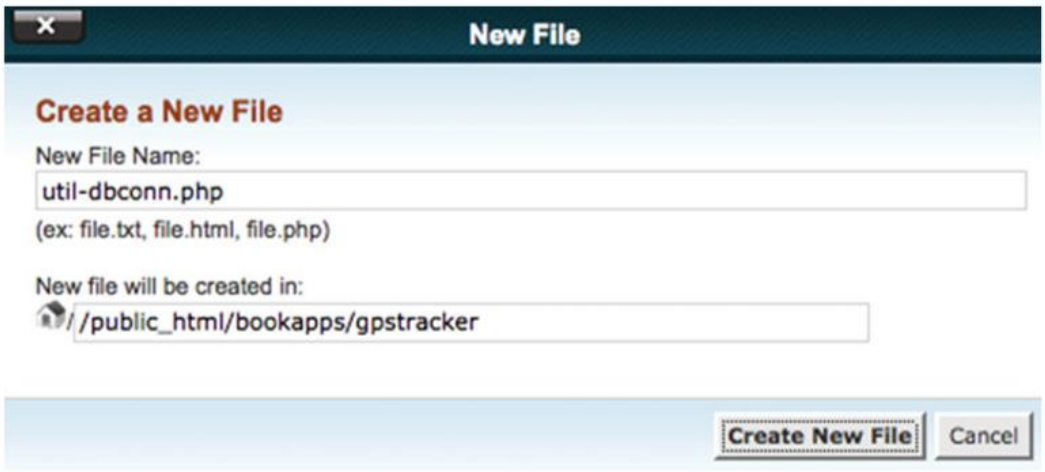
Sekarang tabel database sudah siap, Anda perlu membangun dua layanan. Layanan pertama yang akan menerima koordinat GPS dan menyimpannya di tabel database yang baru dibuat. Layanan kedua akan menampilkan koordinat GPS yang tersimpan di peta.

Proyek ini juga menggunakan PHP untuk membangun penyimpanan dan layanan antarmuka pengguna. Buat folder baru bernama gpstracker di folder publik/root server PHP Anda. Semua kode sumber PHP untuk proyek ini akan masuk ke folder gpstracker ini. Mulai editor teks pilihan Anda.

***Catatan** : Semua kode PHP dikembangkan menggunakan Brackets, yang merupakan editor teks open source. Lihat http://brackets.io/ untuk informasi lebih lanjut.*

9.3 KONEKSI DATABASE

Kedua skrip PHP untuk menyimpan dan menampilkan data harus terhubung ke database. Seperti yang ditunjukkan pada Gambar 9-6, buat file baru bernama util-dbconn.php di folder gpstracker. File ini akan digunakan oleh kedua skrip alih-alih mengulangi kode.



Gambar 9-6. File konektivitas database umum yang disebut util-dbconn.php

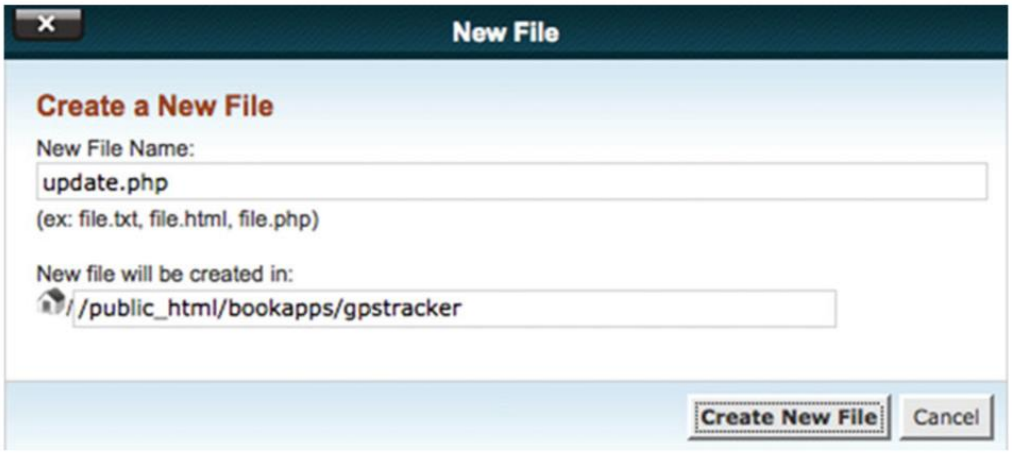
Buka file dalam editor teks dan salin atau ketik kode dari Listing 9-2. Seperti yang Anda lihat, tidak banyak kode dalam file ini. Empat variabel $servername, $username, $password, dan $dbname berisi informasi koneksi. Buat koneksi baru dengan melewatkan keempat variabel ini dan simpan referensi koneksi di variabel $mysqli. Kondisi IF dalam kode hanya memeriksa kesalahan selama upaya koneksi dan mencetaknya jika ada.

Listing 9-2. Kode Konektivitas Database Umum util-dbconn.php

```
<?php
    $servername = "SERVER_NAME";
    $dbname = "DB_NAME";
    $username = "DB_USERNAME";
    $password = "DB_PASSWORD";
    //Open a new connection to MySQL server
    $mysqli = new mysqli($servername, $username, $password, $dbname);
    //Output connection errors
    if ($mysqli->connect_error)
    {
    die("[ERROR] Connection Failed: ". $mysqli->connect_error);
    }
?>
```

9.4 MENERIMA DAN MENYIMPAN DATA SENSOR

Seperti yang ditunjukkan pada Gambar 9-7, buat file baru bernama update.php di folder gpstracker. Skrip ini akan melakukan dua tugas—pertama akan mengambil informasi dari permintaan HTTP dan kemudian akan mengupdate informasi ini di tabel database. Buka file yang baru dibuat dalam editor teks dan salin atau ketik kode yang disediakan di Listing 9-3. Seperti disebutkan pada langkah sebelumnya, untuk menyimpan data, koneksi database perlu dibuat. Anda membuat util-dbconn.php untuk melakukan tugas itu, jadi dalam file ini Anda perlu menyertakan util-dbconn.php. File util-dbconn.php menyediakan akses ke variabel $mysqli, yang berisi referensi koneksi dan akan digunakan untuk menjalankan kueri SQL.



Gambar 9-7. File untuk menerima dan menambah/**mengupdate** data di update.php

Contoh dalam buku ini di-host di http://bookapps.codifythings.com/gpstracker/, dan Arduino akan mengirimkan koordinat GPS ke update.php menggunakan metode HTTP GET. Seperti yang dibahas dalam Bab 2, HTTP GET menggunakan string kueri untuk mengirim data

permintaan. Jadi, URL lengkap dengan string kueri yang akan digunakan Arduino menjadi http://bookapps.codifythings.com/gpstracker/update.php?clientID=Sheep1&latitude=41.83&longitude=-87.68. Kode PHP Anda perlu mengekstrak ID klien dan koordinat GPS dari string kueri menggunakan pernyataan $_GET['parameterName'].

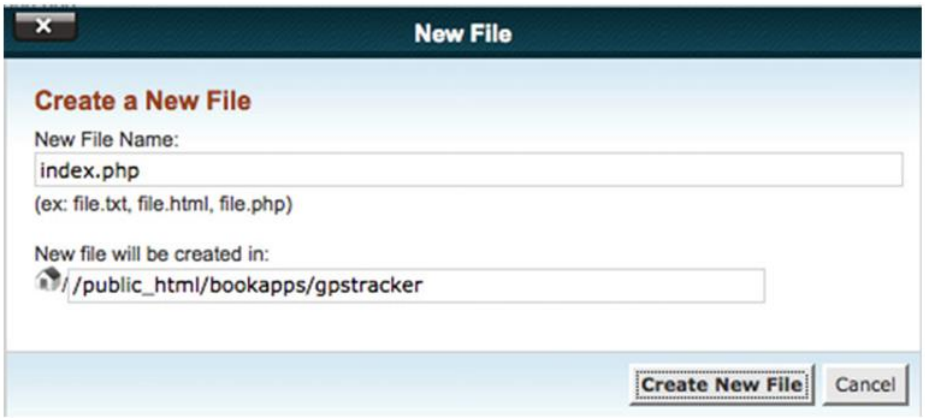
Sekarang Anda perlu menyimpan koordinat GPS ini di tabel database, baik di baris yang ada atau dengan menyisipkannya di baris baru. Siapkan pernyataan SQL INSERT OR UPDATE dalam variabel $sql. Anda harus meneruskan nilai CLIENT_ID, LATITUDE, dan LONGITUDE dalam kueri SQL sementara TIMESTAMP akan dibuat secara otomatis oleh database. Terakhir, jalankan pernyataan SQL INSERT OR UPDATE menggunakan $mysqli->query($sql) dan periksa variabel $result apakah berhasil atau gagal.

Listing 9-3. Kode untuk Menerima dan Menambah/Mengupdate Data di update.php

```
<?php
    include('util-dbconn.php');
    $clientID = $_GET['clientID'];
    $latitude = $_GET['latitude'];
    $longitude = $_GET['longitude'];
    $sql = "INSERT INTO `GPS_TRACKER_DATA` (CLIENT_ID, LATITUDE, LONGITUDE)
    VALUES('$clientID', $latitude, $longitude) ";
    $sql = $sql. "ON DUPLICATE KEY UPDATE CLIENT_ID='$clientID',
    LATITUDE=$latitude, LONGITUDE=$longitude";
    echo $sql;
    if (!$result = $mysqli->query($sql))
    {
    echo "[Error] ". mysqli_error(). "\n";
    exit();
    }
    $mysqli->close();
    echo "[DEBUG] Updated GPS Coordinates Successfully\n";
?>
```

9.5 *MAP*

Semua koordinat GPS yang disimpan dalam database belum terlihat oleh siapa pun. Selanjutnya, Anda akan membangun halaman web yang akan menampilkan semua koordinat pada peta. Seperti yang ditunjukkan pada Gambar 9-8, buat file baru bernama index.php di folder gpstracker.



Gambar 9-8. File untuk menampilkan peta adalah index.php

Listing 9-4 menyediakan kode lengkap untuk index.php, jadi salin atau tulis kode di index.php. Kode ini menggunakan Google Maps API untuk membuat peta dan menampilkan semua koordinat. Anda tidak perlu mengunduh atau memasang kode apa pun; API dapat diakses melalui Internet sehingga tag skrip Anda hanya perlu menunjuk ke http://maps.googleapis.com/maps/api/js?sensor=false sebagai sumbernya.

Untuk mengisi peta, Anda harus terlebih dahulu memuat data dari database dalam variabel array lokasi. Tambahkan kode PHP Anda untuk memuat data dari tabel database di dalam fungsi JavaScript init(). Sertakan util-dbconn.php karena koneksi database harus dibuat terlebih dahulu, lalu siapkan pernyataan SQL SELECT. Jalankan kueri dan siapkan larik lokasi dari hasilnya.

Setelah kode PHP dan di dalam fungsi init(), inisialisasi peta baru. Atur tingkat zoom, koordinat default, dan jenis petanya. Selanjutnya baca array dalam satu lingkaran dan tandai semua koordinat pada peta. Tag <body> memiliki kode untuk menampilkan judul di atas halaman dan peta yang dibuat sebelumnya.

Listing 9-4. Struktur Kode untuk Peta di index.php

```
<html lang="en">
<head>
<title>Livestock Tracking System</title>
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/
js?sensor=false"></script>
<script>
function init()
{
<?php
        include('util-dbconn.php');
        $sql = "SELECT * FROM `GPS_TRACKER_DATA`";
        $result = $mysqli->query($sql);
        $resultCount = $result->num_rows;
        $zoomLatitude = "";
        $zoomLongitude = "";
         echo "var locations = [";
        if ($resultCount > 0)
        {
                $currentRow = 0;
                while($row = $result->fetch_assoc())
                {
                        $currentRow = $currentRow + 1;
                        $clientID=$row["CLIENT_ID"];
                        $latitude=$row["LATITUDE"];
                        $longitude=$row["LONGITUDE"];
                        if($currentRow == 1)
                        {
                        $zoomLatitude = $latitude;
                        $zoomLongitude = $longitude;
                         }
                        echo "['".$clientID."',".$latitude.",".$longitude."]";
                        if($currentRow < $resultCount)
                        {
```

```
                    echo ",";
                    }
            }
        }
    echo "];";
    echo "var latitude = '$zoomLatitude';";
    echo "var longitude = '$zoomLongitude';";
    $mysqli->close();
    ?>
    map = new google.maps.Map(document.getElementById('map'),
    {
                                zoom: 10,
                                center: new google.maps.LatLng(latitude,  longitude),
                                mapTypeId: google.maps.MapTypeId.ROADMAP
                                });
    var infowindow = new google.maps.InfoWindow();
    var marker, i;
    for (i = 0; i < locations.length; i++)
    {
        marker = new google.maps.Marker({
                position: new
                                google.maps.LatLng(locations[i][1],
                                locations[i][2]),map: map});
          google.maps.event.addListener(marker, 'click',
                (function(marker, i)
                {
                        return function()
                                {
                                infowindow.setContent(locations[i][0]);
                                infowindow.open(map, marker);
                                }
                })(marker, i));
        }
}
</script>
</head>
<body style="background-color: #9bcc59">
<div style="align: center;">
<font size="5px" color="white">Livestock Tracking System</font></div>
<div id="map" style="width: 100%; height: 50%; margin-top: 50px;"></div>
<script type="text/javascript">  init();
</script>
</body>
</html>
```

9.6 KODE (ARDUINO)

Komponen terakhir dari proyek ini adalah kode Arduino untuk menghubungkan ke Internet menggunakan WiFi, mendapatkan koordinat GPS saat ini, dan mempublikasikannya

ke server. Mulai Arduino IDE Anda dan ketik kode yang disediakan di sini atau unduh dari situs dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi lima bagian.

Library eksternal

- Konektivitas Internet (Wi-Fi)
- Baca koordinat GPS
- HTTP (terbitkan)
- Fungsi standar

Bagian pertama dari kode, seperti yang disediakan dalam Listing 9-5, mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Sketsa ini memiliki banyak dependensi—untuk konektivitas Internet, Anda harus menyertakan <WiFi.h>, untuk komunikasi dengan modul GPS, Anda perlu menyertakan <SoftwareSerial.h>, dan untuk membaca koordinat GPS, Anda harus menyertakan < TinyGPS.h>. Anda dapat mengunduh <TinyGPS.h> dari https://github.com/mikalhart/TinyGPS/releases/tag/v13.

Listing 9-5. Kode untuk Menyertakan Dependensi Eksternal

```
#include <SPI.h>
#include <WiFi.h>
#include <TinyGPS.h>
#include <SoftwareSerial.h>
```

Konektivitas Internet (Nirkabel)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (Bab 2) di sini.

9.7 DAPATKAN KOORDINAT GPS

Bagian ketiga dari kode, seperti yang disediakan dalam Listing 9-6, mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membaca koordinat GPS. Setelah modul GPS terhubung ke Arduino dan dihidupkan, modul akan mencari satelit dan mulai mengirim data pada port serial D2 dan D3 ke Arduino. Data ini tidak masuk akal, jadi untuk menemukan informasi garis lintang dan garis bujur, Anda akan menggunakan library TinyGPS. Library ini mem-parsing data yang berasal dari modul GPS dan menyediakan cara mudah untuk mengambil informasi yang diperlukan. Jadi inisialisasi variabel library TinyGPS.

Fungsi getGPSCoordinates() membaca data GPS dari port serial D2 dan D3. Modul GPS mungkin memerlukan beberapa detik untuk menemukan satelit, sehingga nilai lintang dan bujur yang dikembalikan mungkin tidak valid. Jika lintang dan bujur sama dengan TinyGPS::GPS_INVALID_F_ANGLE, itu berarti koordinatnya tidak valid, jadi sampai kode menerima koordinat yang valid, itu terus mencetak Pencarian Satelit di monitor serial. Setelah koordinat yang valid diterima, fungsi transmitSensorData(latitude, longitude) dipanggil.

Listing 9-6. Kode untuk Membaca Koordinat GPS

```
TinyGPS gps;
SoftwareSerial ss(2, 3); // GPS TX = Arduino D2, GPS RX = Arduino D3
static void smartdelay(unsigned long ms)
{
unsigned long start = millis();
do
{
```

```
while (ss.available())
gps.encode(ss.read());
} while (millis() - start < ms);
}
void getGPSCoordinates()
{
        float latitude;
        float longitude;
        unsigned long age = 0;
        gps.f_get_position(&latitude, &longitude, &age);
        smartdelay(10000);
        // Transmit sensor data
        if(latitude != TinyGPS::GPS_INVALID_F_ANGLE &&
        longitude != TinyGPS::GPS_INVALID_F_ANGLE)
        {
        transmitSensorData(latitude, longitude);
        }
        Else
        {
        Serial.println("[INFO] Searching for Satellite");
    }
}
```

Publishkan Data

Bagian kode keempat, seperti yang disediakan dalam Listing 9-7, mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membuat dan mengirim permintaan HTTP ke server. Kode ini adalah versi HTTP GET yang sedikit dimodifikasi yang Anda kembangkan di Bab 3.

Modifikasi utama dalam kode ini adalah kemampuannya untuk membuka dan menutup koneksi ke server secara berulang. Selain itu, pastikan untuk mengubah nilai server dan port ke nilai server PHP Anda, variabel requestData, dan nilai URL.

Listing 9-7. Publish HTTP

```
//IP address of the server
char server[] = {"bookapps.codifythings.com"};
int port = 80;
unsigned long lastConnectionTime = 0;
const unsigned long postingInterval = 10L * 1000L;
void transmitSensorData(float latitude, float longitude)
{
// Read all incoming data (if any)
while (client.available())
{
char c = client.read();
}
if (millis() - lastConnectionTime > postingInterval)
{
client.stop();
Serial.println("[INFO] Connecting to Server");
```

```
String requestData = "clientID=Sheep1&latitude=" + String(latitude)
+ "&longitude=" + String(longitude);
Serial.println("[INFO] Query String: " + requestData);
 // Prepare data or parameters that need to be posted to server
 if (client.connect(server, port))
{
Serial.println("[INFO] Server Connected - HTTP GET Started");
  // Make a HTTP request:
client.println("GET /gpstracker/update.php?" + requestData + " HTTP/1.1");
        client.println("Host: "+ String(server));
        client.println("Connection: close");
        client.println();
        lastConnectionTime = millis();
        Serial.println("[INFO] HTTP GET Completed");
        }
        else
        {
        // Connection to server:port failed
        Serial.println("[ERROR] Connection Failed");
        }
   }
   Serial.println("----------------------------------------");
}
```

Fungsi Standar

Bagian kode terakhir disediakan dalam Listing 9-8. Ini mengimplementasikan fungsi setup() dan loop() standar Arduino. Fungsi setup() menginisialisasi port serial. Perhatikan bahwa baud rate adalah 115200, yang berbeda dari yang Anda gunakan sejauh ini. Alasan perbedaan akan jelas ketika Anda melihat baris kode berikutnya: ss.begin(9600). Pernyataan ini menginisialisasi komunikasi dengan modul GPS pada port serial D2 dan D3 (ss adalah turunan dari library SoftwareSerial yang Anda inisialisasi di Listing 9-6). Modul GPS yang digunakan dalam proyek ini berkomunikasi pada 9600 baud rate secara default, oleh karena itu 115200 digunakan untuk log monitor serial. Modul GPS yang Anda gunakan mungkin memiliki baud rate default yang berbeda, jadi pastikan untuk memeriksa lembar data pabrikan untuk menemukan yang benar. Selanjutnya, sambungkan ke Internet menggunakan WiFi. Fungsi loop() hanya perlu memanggil fungsi getGPSCoordinates(). Ia membaca koordinat GPS dan, secara berkala, memanggil fungsi transmitSensorData() untuk memublikasikan koordinat GPS ke server.

Listing 9-8. Kode untuk Fungsi Arduino Standar

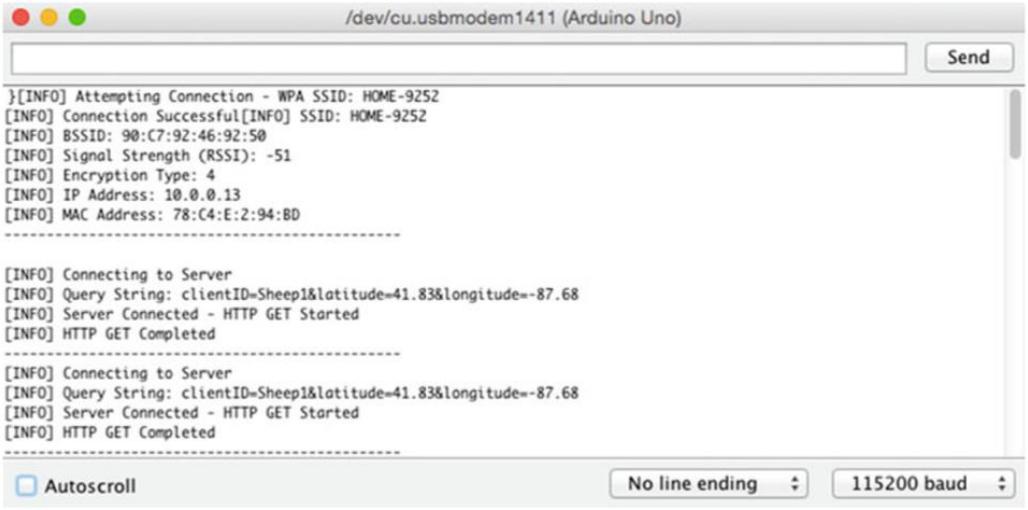
```
void setup()
{
    // Initialize serial port
    Serial.begin(115200);
    // Initialize serial port for GPS data
    ss.begin(9600);
    //Connect Arduino to internet
    connectToInternet();
}
void loop()
```

```
{
// Get GPS Coordinates
getGPSCoordinates();
```

Kode Arduino Anda sekarang sudah lengkap.

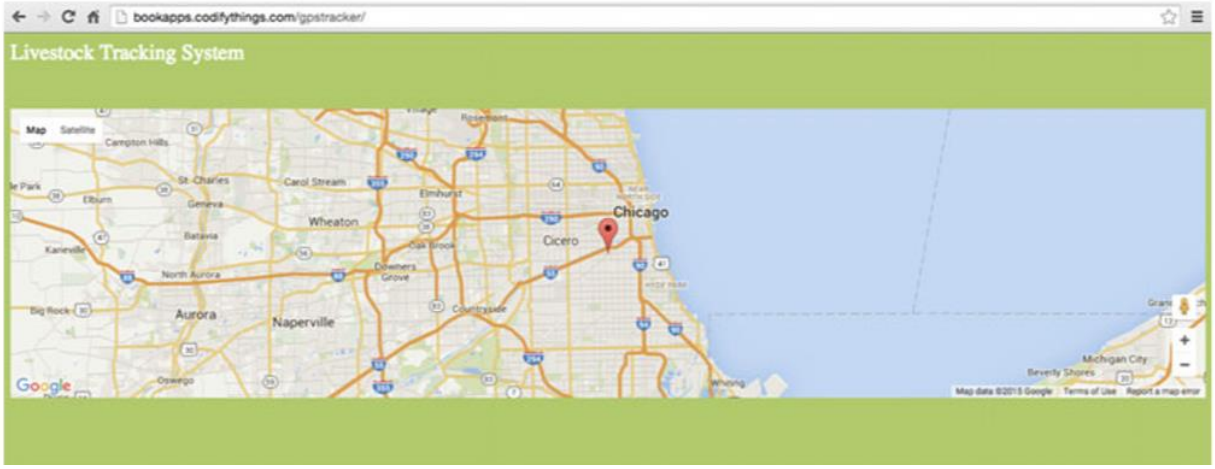
9.8 PRODUK AKHIR

Untuk menguji aplikasi, pastikan server MySQL dan PHP Anda aktif dan berjalan dengan kode yang diterapkan. Juga verifikasi dan Upload kode Arduino seperti yang dibahas pada Bab 1. Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log yang mirip dengan yang ditunjukkan pada Gambar 9-9.



Gambar 9-9. Pesan log dari sistem pelacakan ternak

Setelah GPS diinisialisasi, yang mungkin memakan waktu beberapa detik, koordinat saat ini akan dipublishkan ke server. Periksa aplikasi web Anda dengan mengakses URL proyek; dalam hal ini adalah http://bookapps.codifythings.com/gpstracker. Aplikasi web Anda akan terlihat mirip dengan Gambar 9-10.



Gambar 9-10. Versi terakhir dari sistem pelacakan ternak

9.9 RINGKASAN

Dalam bab ini Anda belajar tentang hal-hal yang sadar lokasi. Mereka memiliki banyak kegunaan hebat, dan, ketika digabungkan dengan sensor lain, mereka dapat meningkatkan banyak aspek kehidupan kita, seperti tanggap darurat, pemeliharaan dan

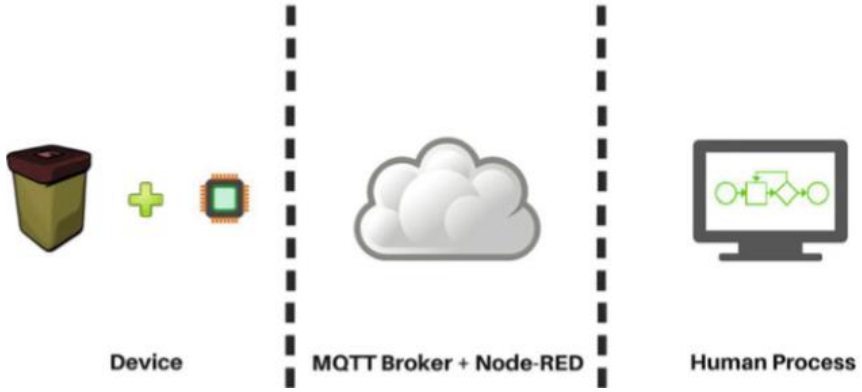
perutean yang dioptimalkan, dan banyak lagi. Anda mengembangkan aplikasi IoT yang menerbitkan data pelacakan ternak ke server tempat informasi ini ditampilkan di peta. Anda dapat meningkatkan beberapa aplikasi lain yang Anda kembangkan di bab sebelumnya dengan membuatnya sadar lokasi, termasuk:

Sistem deteksi penyusupan dari Bab 5. Ketika penyusupan terdeteksi, Anda dapat mengirim peringatan ke perusahaan keamanan dengan koordinat yang tepat sehingga mereka dapat mengirim seseorang untuk menyelidiki.

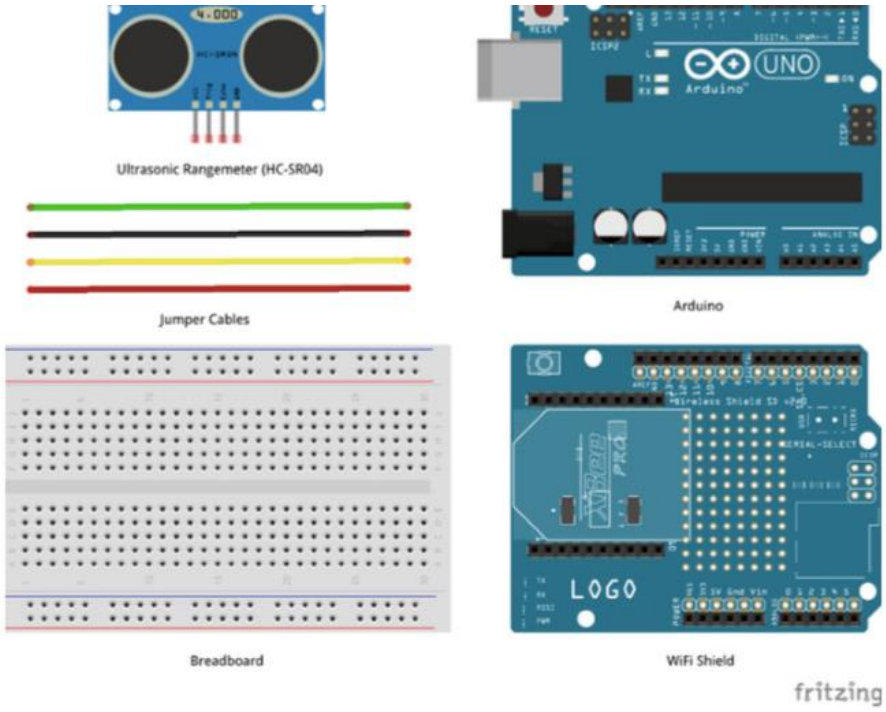
Sistem parkir pintar dari Bab 7. Anda dapat memberikan koordinat tempat parkir yang tepat sehingga pengemudi yang mencari tempat parkir dapat memasukkan koordinat di GPS mereka untuk petunjuk arah. Tidak semua skenario memerlukan modul GPS yang dibuat khusus. Ponsel cerdas juga peka terhadap lokasi dan dapat digunakan untuk membangun aplikasi IoT. Untuk skenario seperti pelacakan ternak, Anda perlu melampirkan modul GPS yang dibuat khusus, tetapi untuk skenario lain, seperti pelacak jarak tempuh mobil, Anda juga memiliki opsi untuk menggunakan ponsel cerdas.

BAB 10
POLA IoT : MACHINE to HUMAN

Karena persyaratan peraturan atau kurangnya teknologi, akan ada skenario di mana intervensi manusia diperlukan untuk menanggapi peringatan yang dihasilkan sensor. Dalam bab ini, Anda akan membangun sistem pengelolaan sampah sederhana untuk menguraikan kasus penggunaan ini. Gambar 10-1 menunjukkan diagram tingkat tinggi dari semua komponen yang terlibat dalam sistem ini. Komponen pertama adalah perangkat Arduino yang memonitor tingkat sampah dengan sensor jarak dan menerbitkan pesan ke broker MQTT. Komponen kedua adalah aliran Node-RED yang subscribe broker MQTT. Komponen terakhir adalah alur kerja yang dimulai setiap kali tingkat sampah tinggi dan pengambilan perlu dijadwalkan.



Gambar 10-1. Komponen sistem pengelolaan sampah



Gambar 10-2. Hardware yang diperlukan untuk sistem pengelolaan limbah

10.1 TUJUAN PEMBELAJARAN

Pada akhir bab ini, Anda akan dapat:

- Membaca data dari sensor jarak
- Publish pesan ke broker MQTT
- Membangun alur kerja di *Effektif* (berganti nama menjadi Signavio Workflow)
- Membuat aliran Node-RED dan mulai dari Arduino

Hardware yang Diperlukan

Gambar 10-2 memberikan Listing semua komponen hardware yang diperlukan untuk membangun sistem pengelolaan limbah.

Software yang Diperlukan

Untuk mengembangkan sistem pengelolaan sampah ini, Anda memerlukan software berikut:

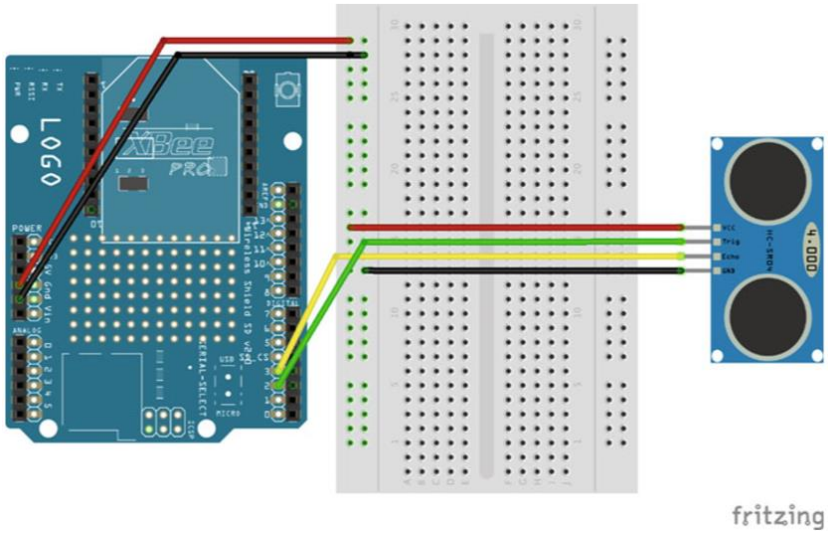
- Arduino IDE 1.6.4 atau lebih baru
- Efektif (dihosting)
- Node-RED 0.1 3.2 atau lebih baru

Sirkuit

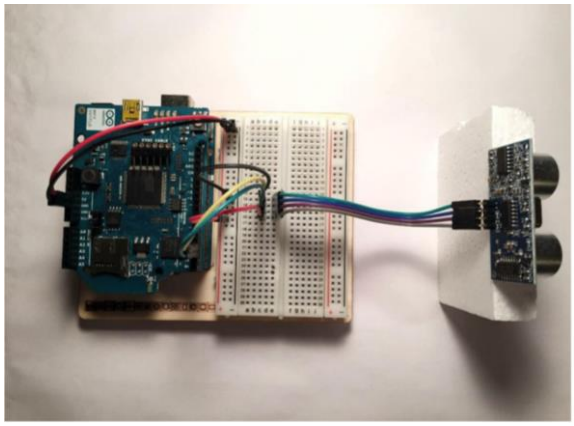
Di bagian ini, Anda akan membangun sirkuit yang diperlukan untuk sistem pengelolaan limbah. Rangkaian ini menggunakan sensor jarak ultrasonik untuk mendeteksi objek, seperti yang diilustrasikan pada BaB 7. Sensor dipasang di bagian atas tong sampah dan mengirimkan ledakan ultrasonik yang memantul dari sampah di tong sampah. Sirkuit membaca gema, yang digunakan untuk menghitung tingkat sampah.

1. Pastikan Arduino tidak terhubung ke sumber listrik, seperti ke komputer melalui USB atau baterai.
2. Pasang pelindung WiFi ke bagian atas Arduino. Semua pin harus sejajar.
3. Gunakan kabel jumper untuk menghubungkan port power (5V) dan ground (GND) pada Arduino ke port power (+) dan ground (-) pada breadboard.
4. Sekarang papan breadboard Anda memiliki sumber daya, gunakan kabel jumper untuk menghubungkan port daya (+) dan arde (-)port breadboard Anda ke port daya dan arde dari sensor jarak.
5. Untuk memicu ledakan ultrasonik, sambungkan kabel jumper dari pin TRIG sensor ke Digital Port 2 Arduino. Kode Anda akan mengatur nilai port ini ke LOW, HIGH, dan LOW untuk memicu burst.
6. Untuk membaca gema, sambungkan kabel jumper dari pin ECHO sensor ke Digital Port 3 Arduino. Kode Anda akan membaca nilai dari port ini untuk menghitung tingkat sampah di kaleng.

Sirkuit Anda sekarang telah selesai dan akan terlihat mirip dengan Gambar 10-3 dan 10-4.



Gambar 10-3. Diagram sirkuit sistem pengelolaan sampah



Gambar 10-4. Sirkuit sebenarnya dari sistem pengelolaan sampah

10.2 KODE (ARDUINO)

Selanjutnya Anda akan menulis kode untuk komponen pertama dari aplikasi ini. Kode ini akan menghubungkan Arduino ke Internet menggunakan WiFi, membaca data sensor jarak untuk mendapatkan tingkat sampah, dan mempublishkan informasi tersebut ke broker MQTT. Mulai Arduino IDE Anda dan ketik kode yang disediakan di sini atau unduh dari situs dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi lima bagian.

- Library eksternal
- Konektivitas Internet (Wi-Fi)
- Baca data sensor
- MQTT (terbitkan)
- Fungsi standar

Library Eksternal

Bagian pertama kode, seperti yang disediakan di Listing 10-1, mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Sketsa ini memiliki dua dependensi utama—untuk konektivitas Internet, Anda harus menyertakan <WiFi.h> (dengan asumsi Anda menggunakan pelindung WiFi) dan untuk komunikasi broker MQTT, Anda harus menyertakan <PubSubClient.h>.

Listing 10-1. Kode untuk Menyertakan Dependensi Eksternal

```
#include <SPI.h>
#include <WiFi.h>
#include <PubSubClient.h>
```

Konektivitas Internet (Nirkabel)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (Bab 2) di sini.

Baca Data Sensor

Bagian ketiga dari kode, seperti yang disediakan dalam Listing 10-2, mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membaca data sensor. Fungsi kalibrasiSensor() menunggu sensor jarak untuk mengkalibrasi dengan benar. Setelah kalibrasi selesai, sensor jarak aktif dan dapat mulai mendeteksi. Jika Anda tidak memberikan cukup waktu untuk mengkalibrasi, sensor jarak mungkin mengembalikan pembacaan yang salah.

Fungsi readSensorData() menghasilkan ledakan untuk mendeteksi tingkat sampah di kaleng. Ini memicu ledakan pada Digital Pin 2 dengan mengirimkan sinyal alternatif—LOW, HIGH, dan LOW lagi. Kemudian membaca gema dari Digital Pin 3, yang memberikan jarak antara sensor dan sampah. Akhirnya, ia memeriksa apakah jaraknya kurang dari ambang batas, dan jika ya, itu berarti tempat sampah hampir penuh dan penjemputan perlu dijadwalkan. Karena ini hanya prototipe, nilai gema 700 telah digunakan. Saat Anda menggunakan sensor ini dalam kehidupan nyata, Anda perlu menyesuaikan nilainya dengan melakukan beberapa tes. Jika tingkat sampah di atas ambang batas, panggil publishSensorData(...) dengan HIGH.

Listing 10-2. Kode untuk Mendeteksi Level Sampah

```
int calibrationTime = 30;
#define TRIGPIN 2       // Pin to send trigger pulse
#define ECHOPIN 3       // Pin to receive echo pulse
void calibrateSensor()
{
//Give sensor some time to calibrate
Serial.println("[INFO] Calibrating Sensor ");
for(int i = 0;
i < calibrationTime; i++)
{
Serial.print(".");
delay(1000);
}
Serial.println("");
Serial.println("[INFO] Calibration Complete");
Serial.println("[INFO] Sensor Active");
delay(50);
}
void readSensorData()
{
// Generating a burst to check for objects
digitalWrite(TRIGPIN, LOW);
delayMicroseconds(10);
digitalWrite(TRIGPIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIGPIN, LOW);
// Distance Calculation
float distance = pulseIn(ECHOPIN, HIGH);
Serial.println("[INFO] Garbage Level: " + String(distance));
if(distance < 700)
{
    Serial.println("[INFO] Garbage Level High");
    // Publish sensor data to server
    publishSensorData("HIGH");
    }
}
```

Publishkan Data

Bagian kode keempat, yang disediakan di Listing 10-3, mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk memublikasikan data ke broker MQTT. Kode ini adalah versi yang sedikit dimodifikasi dari publish MQTT yang Anda kembangkan di Bab 3. Anda tidak perlu membuat perubahan apa pun agar kode berfungsi, tetapi Anda disarankan untuk menyesuaikan beberapa pesan agar tidak tercampur dengan orang lain yang menggunakan nilai yang sama. Semua nilai yang dapat diubah telah disorot dalam huruf tebal pada Listing 10-3. Jika Anda menggunakan server MQTT Anda sendiri, pastikan untuk mengubah nilai server dan port. Dua perubahan yang disarankan termasuk nilai variabel topik dan nama klien yang harus Anda lewati saat terhubung ke broker MQTT.

Listing 10-3. Kode untuk Menerbitkan Pesan ke Broker MQTT

```
// IP address of the MQTT broker
char server[] = {"iot.eclipse.org"};
int port = 1883;
char topic[] = {"codifythings/garbagelevel"};
void callback(char* topic, byte* payload, unsigned int length)
{
 //Handle message arrived  }
PubSubClient pubSubClient(server, port, 0, client);
void publishSensorData(String garbageLevel)
{
// Connect MQTT Broker
Serial.println("[INFO] Connecting to MQTT Broker");
if (pubSubClient.connect( "arduinoIoTClient" ))
{
    Serial.println("[INFO] Connection to MQTT Broker Successful");
    }
     else
    {
    Serial.println("[INFO] Connection to MQTT Broker Failed");
    }
    // Publish to MQTT Topic    if (pubSubClient.connected())
    {
    Serial.println("[INFO] Publishing to MQTT Broker");
    pubSubClient.publish(topic, "Garbage level is HIGH, schedule pickup");
    Serial.println("[INFO] Publish to MQTT Broker Complete");
    }
    else
    {
    Serial.println("[ERROR] Publish to MQTT Broker Failed");
    }
    pubSubClient.disconnect();
}
```

Fungsi Standar

Bagian kode terakhir ditunjukkan pada Listing 10-4. Ini mengimplementasikan fungsi setup() dan loop() standar Arduino. Fungsi setup() menginisialisasi port serial, menyetel mode pin untuk pin trigger dan gema, menghubungkan ke Internet, dan mengkalibrasi sensor jarak. Fungsi loop() hanya perlu memanggil readSensorData() secara berkala.

Listing 10-4. Kode untuk Fungsi Arduino Standar

```
void setup()
{
   // Initialize serial port
   Serial.begin(9600);
   // Set pin mode
   pinMode(ECHOPIN, INPUT);
   pinMode(TRIGPIN, OUTPUT);
   // Connect Arduino to internet
   connectToInternet();
   // Calibrate sensor
   calibrateSensor();
   }
   void loop()
   {
   // Read sensor data
   readSensorData();
   // Delay
   delay(5000);
}
```

Koding Arduino Anda sekarang sudah lengkap.

***Efektif* Workflow**

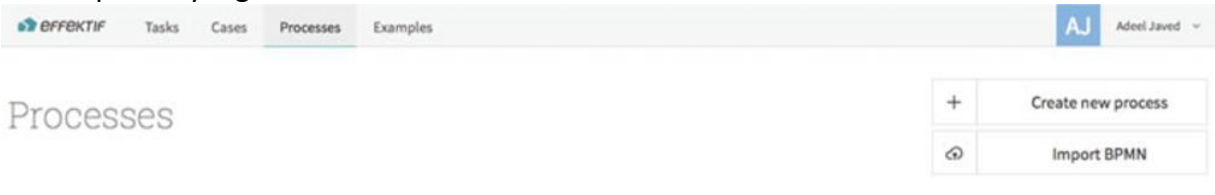
Effektif adalah platform berbasis cloud yang memungkinkan Anda mengotomatiskan alur kerja dan proses rutin ke dalam aplikasi dalam hitungan menit. Untuk tujuan proyek ini, Anda dapat menListing untuk keanggotaan uji coba 30 hari gratis mereka. Anda akan menentukan alur kerja satu langkah yang sangat sederhana yang memungkinkan seseorang untuk memasukkan jadwal pengambilan sampah. *Effektif* hanyalah salah satu contoh solusi manajemen alur kerja dan proses; Anda dapat menggunakan salah satu dari banyak solusi lain yang tersedia juga.

10.3 PROSES PEMBUATAN

Masuk menggunakan kredensial Anda di https://app.effektif.com/. Setelah Anda masuk, pilih Proses dari menu yang ditunjukkan pada Gambar 10-5.

Gambar 10-5. Menu efektif

Ini akan membawa Anda ke Listing semua proses yang ada dan memberi Anda opsi untuk membuat yang baru. Gambar 10-6 menunjukkan layar tempat Anda akan melihat semua proses yang ada.



Gambar 10-6. Listing proses yang ada

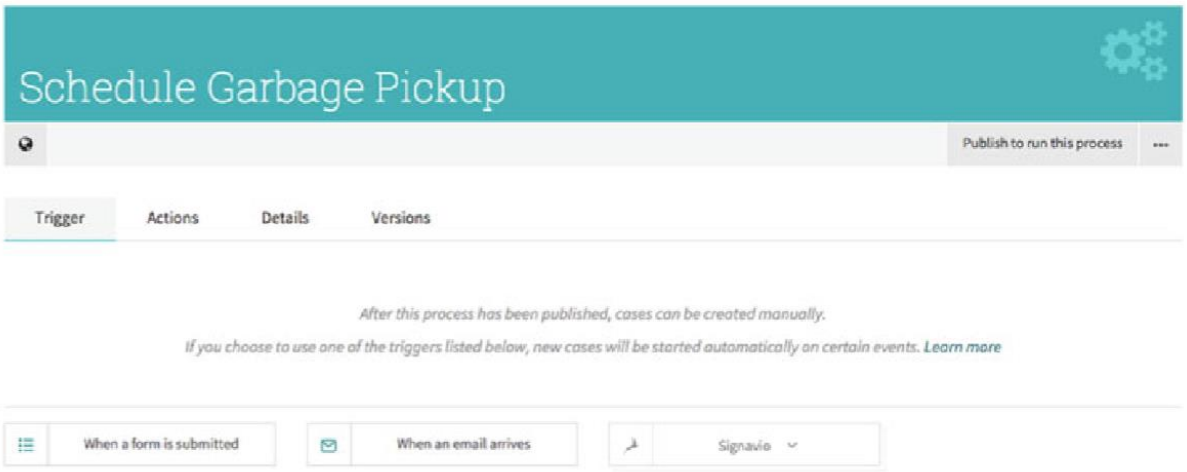
Dari tab Proses yang ditunjukkan pada Gambar 10-6, klik tombol Buat Proses Baru. Seperti yang ditunjukkan pada Gambar 10-7, masukkan Jadwal Pengambilan Sampah sebagai nama proses. Tekan Enter untuk membuat proses dan pindah ke layar berikutnya.

Gambar 10-7. Nama proses baru

10.4 KONFIGURASI PROSES

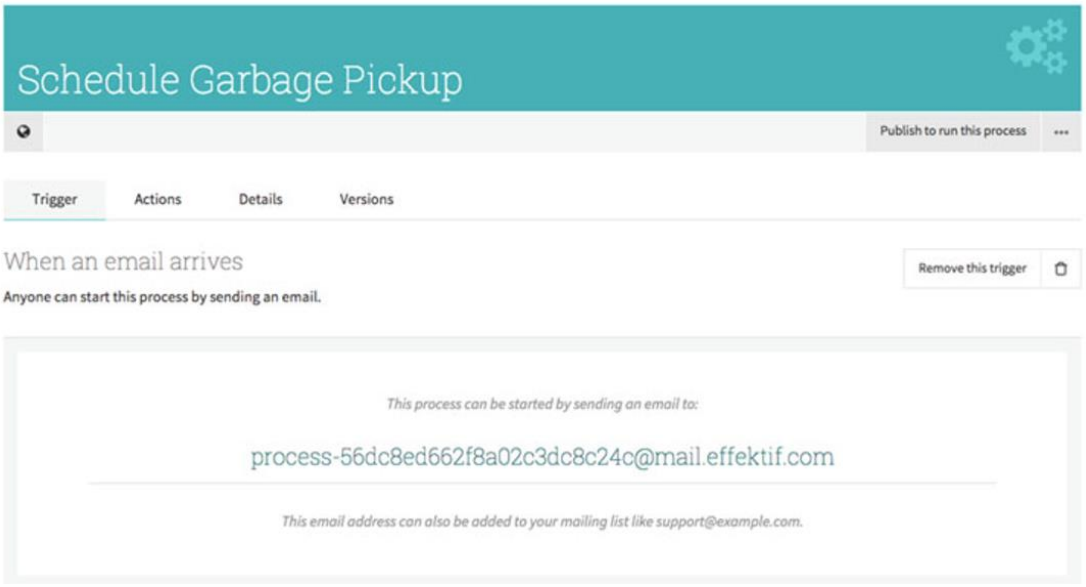
Selanjutnya Anda perlu mengkonfigurasi proses yang baru dibuat. Gambar 10-8 menunjukkan layar konfigurasi proses, di mana Anda dapat menentukan semua aspek proses Anda, termasuk:

- Trigger: Pilih bagaimana proses dapat dimulai
- Actions: Tentukan tindakan manusia dan sistem apa yang akan terjadi dalam proses dan perintahnya
- Detail: Pilih siapa yang akan terlibat dalam proses
- VeVersionrsi: Lihat Listing semua versi proses yang diterbitkan hingga saat ini



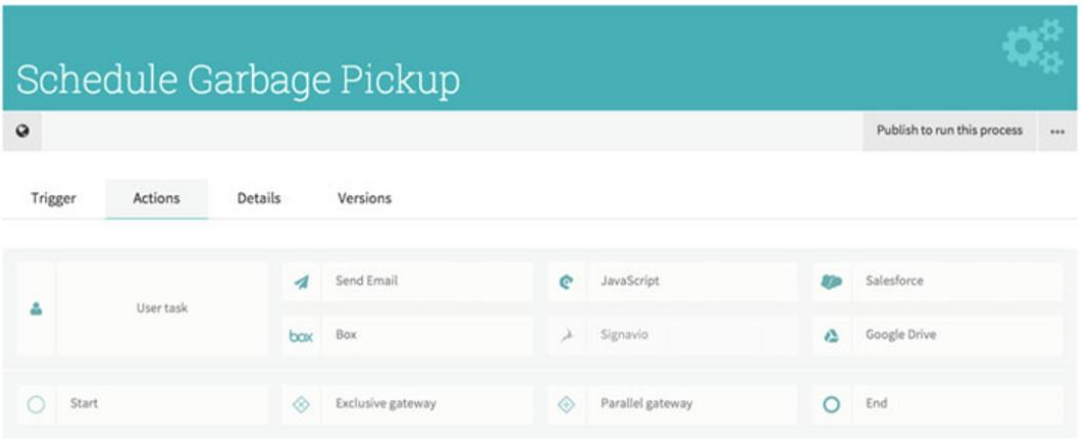
Gambar 10-8. Layar konfigurasi proses P

Pertama, Anda akan memilih trigger untuk proses Anda. Untuk proyek ini, Anda akan memilih email sebagai Trigger. Seperti yang ditunjukkan pada Gambar 10-9 di bawah tab Trigger, klik Saat Email Tiba sebagai opsi Trigger. Setelah Anda memilih trigger email, Effektif menyediakan alamat email yang dibuat secara otomatis. Setiap orang atau sistem dapat mengirim email ke alamat yang dibuat secara otomatis ini dan contoh proses baru akan dimulai di Effektif.



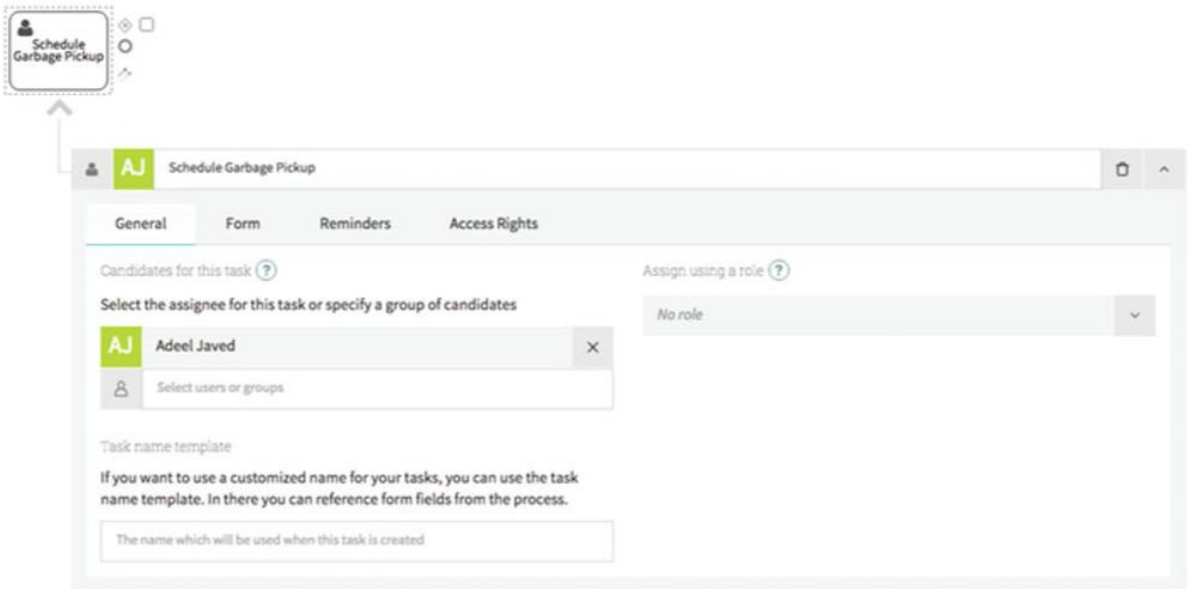
Gambar 10-9. Opsi trigger proses

Selanjutnya Anda akan membuat dan mengkonfigurasi proses penjadwalan satu langkah. Seperti yang ditunjukkan pada Gambar 10-10, tab Tindakan memungkinkan Anda memilih jenis tugas yang Anda inginkan untuk dilakukan proses.



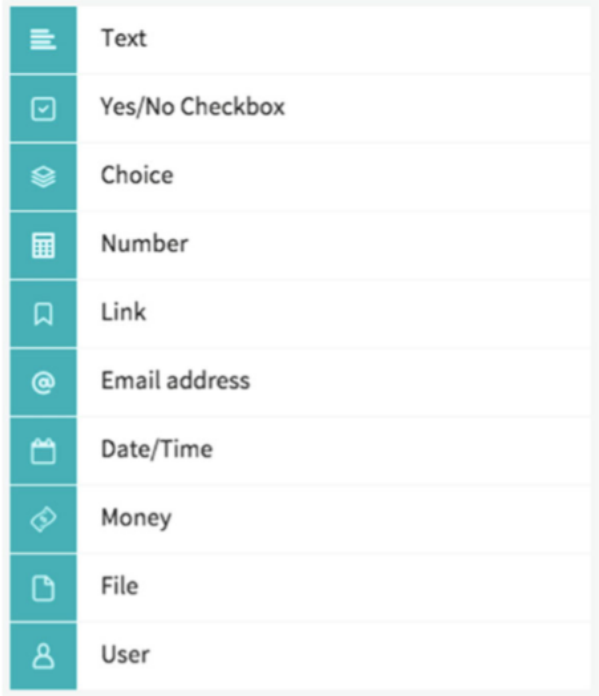
Gambar 10-10. Proses tindakan

Dari tab Tindakan yang ditunjukkan pada Gambar 10-10, klik Tugas Pengguna untuk membuat tindakan yang perlu dilakukan seseorang. Seperti yang ditunjukkan pada Gambar 10-11, masukkan judul tugas ini sebagai Jadwal Pengambilan Sampah. Anda juga perlu menentukan informasi Tugas, seperti apakah satu pengguna atau sekelompok pengguna dapat melakukan tugas ini. Kandidat ini dapat ditentukan di layar Organisasi Saya di bawah Profil. Untuk mempermudah, pilih nama yang Anda gunakan saat membuat akun Effektif.



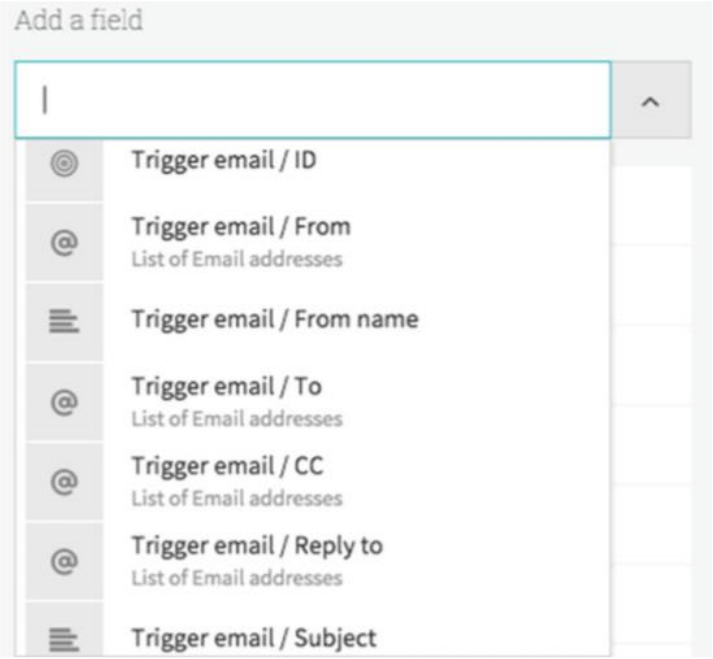
Gambar 10-11. Jenis tindakan dan tugas

Selanjutnya Anda akan mengkonfigurasi bagaimana layar akan terlihat. Ini adalah aktivitas point-and-klik sederhana. Klik pada tab Formulir. Ada dua cara untuk menambahkan bidang ke layar—Anda dapat membuat bidang baru menggunakan salah satu kontrol yang disediakan atau Anda dapat menggunakan bidang yang sudah ada (dibuat oleh sistem atau ditentukan sebelumnya). Gambar 10-12 menunjukkan Listing kontrol yang saat ini tersedia di Effektif.



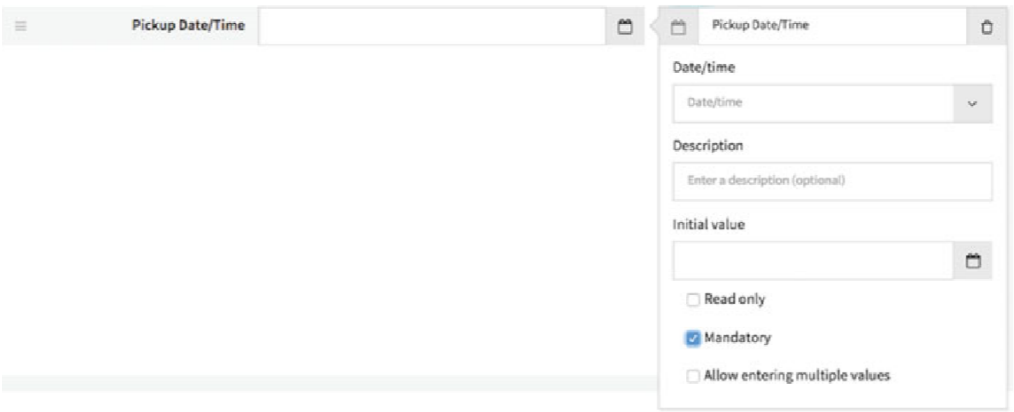
Gambar 10-12. Listing kontrol yang tersedia

Gambar 10-13 menunjukkan Listing kolom yang ada yang dapat digunakan kembali. Karena Anda memilih email sebagai opsi Trigger, bidang email trigger tersedia dalam Listing.



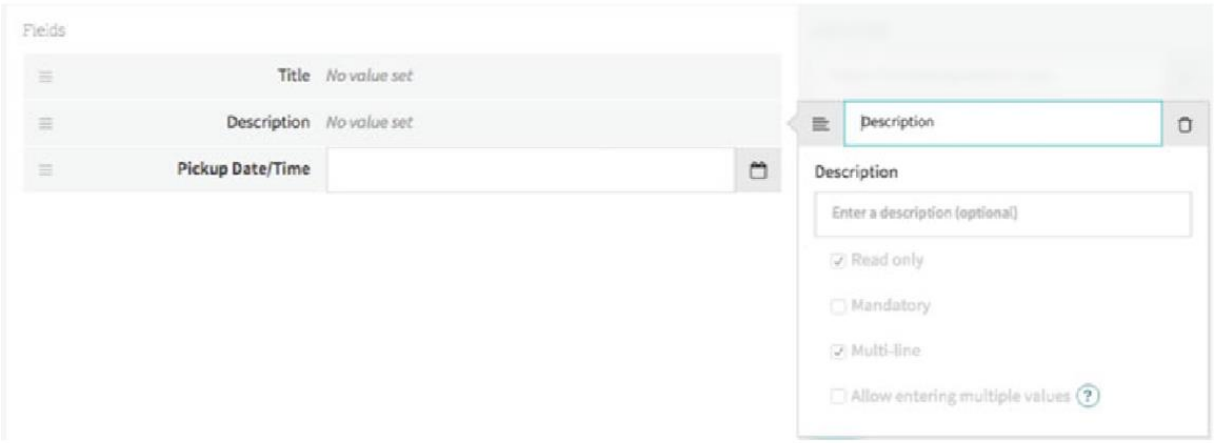
Gambar 10-13. Listing bidang yang ada

Untuk proses ini, Anda akan menggunakan bidang baru dan yang sudah ada. Karena proses dipicu oleh email, Anda perlu menampilkan beberapa informasi dari email. Pilih **Trigger Email/Subject** dan **Trigger Email/Body** dari Add a **Field list**. Ini akan ditambahkan ke formulir Anda. Ubah namanya masing-masing menjadi Judul dan Deskripsi. Anda juga perlu menambahkan bidang Tanggal/Waktu baru agar seseorang dapat memasukkan tanggal/waktu pengambilan sampah. Seperti yang ditunjukkan pada Gambar 10-14, pilih **Date/Time** dari **Add a Field list,** atur namanya menjadi **Pickup Data/Time**, dan jadikan sebagai **field wajib**.



Gambar 10-14. Tambahkan bidang baru pada formulir

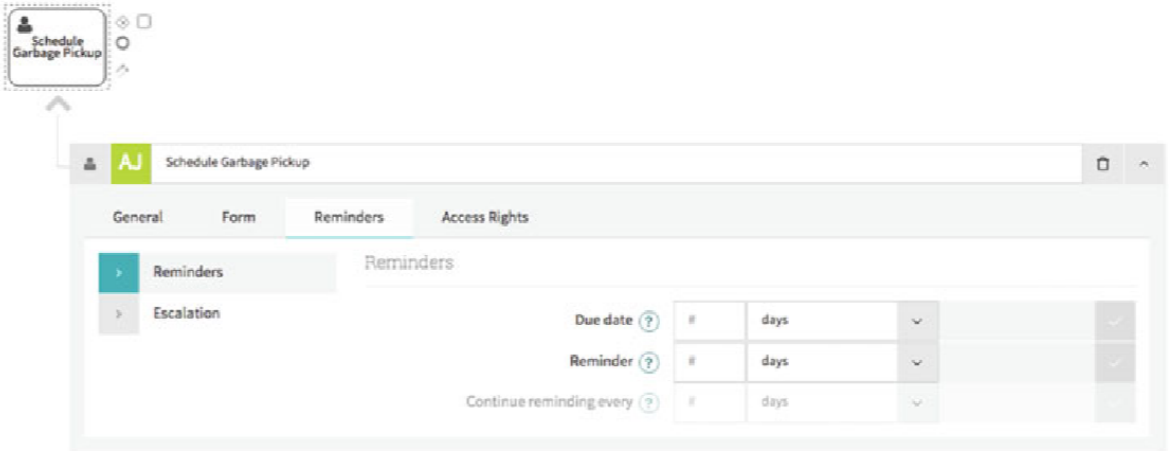
Anda dapat mengatur ulang urutan semua bidang pada formulir dan mengubah propertinya agar lebih mudah dipahami. Layout Screen akhir dari tindakan Anda akan terlihat mirip dengan Gambar 10-15.



Gambar 10-15. Tata letak formulir akhir

Selanjutnya klik pada tab Pengingat dan, seperti yang ditunjukkan pada Gambar 10-16, Anda memiliki opsi untuk menentukan berbagai jenis pengingat untuk tugas tersebut. Untuk saat ini Anda dapat membiarkannya apa adanya untuk semua jenis pengingat.

- ***Due Date***: Kapan tugas jatuh tempo?
- ***Reminder***: Kapan pengingat harus dikirim ke pengguna bahwa tugas semakin tertunda?
- ***Continue reminding every***: Sampai kapan sistem harus terus mengirimkan pengingat?
- ***Escalation***: Jika pengguna masih tidak mengambil tindakan, kepada siapa tugas harus dipindahkan atau didelegasikan?

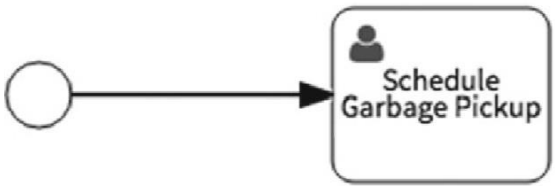


Gambar 10-16. Pengingat tugas

Tindakan **Pickup process available** telah dikonfigurasikan sepenuhnya, jadi sekarang Anda perlu menentukan alurnya. Dari tab Tindakan, pilih tindakan Mulai untuk menambahkannya ke alur tepat sebelum tindakan **Pickup process available**, seperti yang ditunjukkan pada Gambar 10-17.

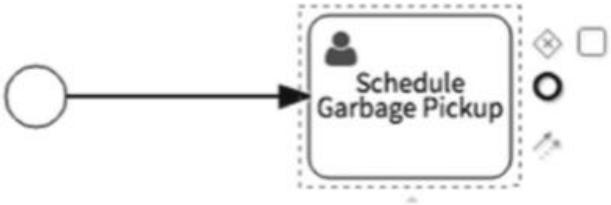
Gambar 10-17. Mulai tindakan ditambahkan ke aliran

Hubungkan tindakan Mulai ke tindakan **Pickup process available**, seperti yang ditunjukkan pada Gambar 10-18.



Gambar 10-18. Hubungkan tindakan Mulai dan **Pickup process available**

Selanjutnya, pilih tindakan Jadwalkan Penjemputan Sampah dan, dari opsi yang tersedia, klik Akhiri tindakan, seperti yang ditunjukkan pada Gambar 10-19.

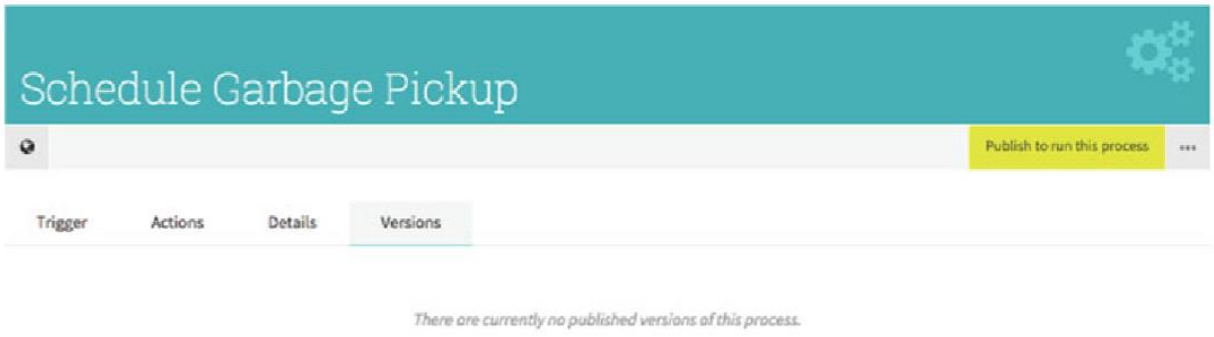


Gambar 10-19. Hubungkan Mulai dan Jadwalkan aktivitas Penjemputan

Tindakan Akhiri baru akan ditambahkan ke alur, tepat setelah tindakan Jadwalkan Pengambilan, seperti yang ditunjukkan pada Gambar 10-20. Ini akan memastikan bahwa proses berakhir setelah pengguna memasukkan tanggal/waktu pengambilan pada formulir.

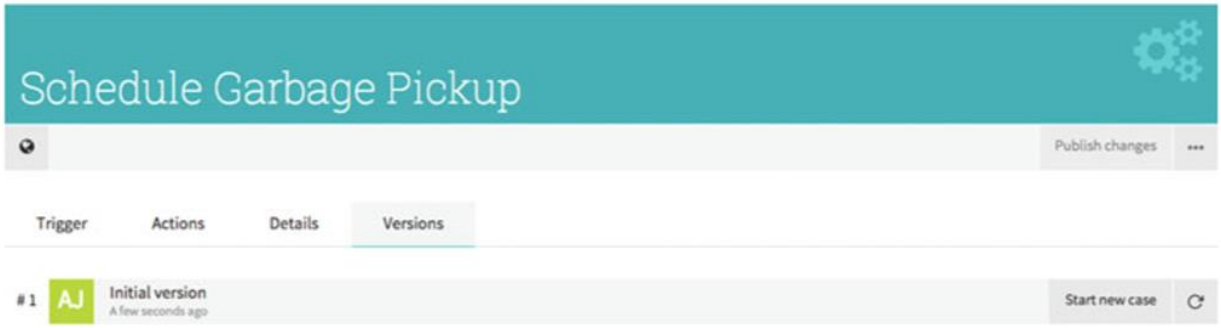
Gambar 10-20. Hubungkan Jadwal Penjemputan dan Akhiri aktivitas

Langkah terakhir adalah membuat proses tersedia. Untuk melakukannya, alihkan ke tab Versions, seperti yang ditunjukkan pada Gambar 10-21.



Gambar 10-21. Publishkan perubahan proses

Selanjutnya, klik tombol Publishkan Perubahan dan versi baru dari proses tersebut akan segera muncul di Listing Versi, seperti yang ditunjukkan pada Gambar 10-22.



Gambar 10-22. Versi proses

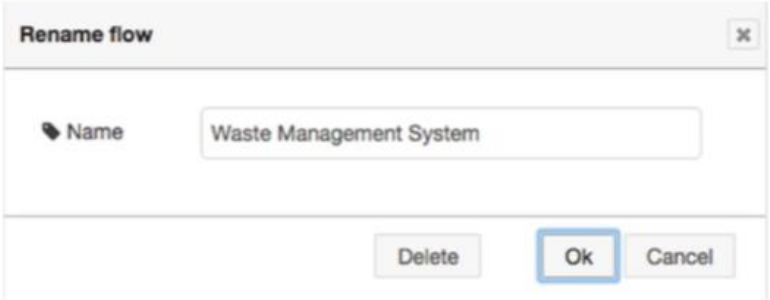
Ini melengkapi konfigurasi proses Anda.

10.5 NODE-RED FLOW

Komponen terakhir dari aplikasi IoT Anda adalah aliran Node-RED yang akan subscribe topik MQTT tempat Arduino menerbitkan pesan dan kemudian memulai proses di Effektif. Mulai server dan desainer Node-RED, seperti yang dijelaskan di Bab 4. Seperti yang ditunjukkan pada Gambar 10-23, klik **+** untuk membuat alur baru.

Gambar 10-23. Buat aliran N ode-RED baru

Klik dua kali nama tab aliran untuk membuka kotak dialog properti. Seperti yang ditunjukkan pada Gambar 10-24, ganti nama aliran **Waste Management System** dan klik **OK** untuk menyimpan perubahan Anda.

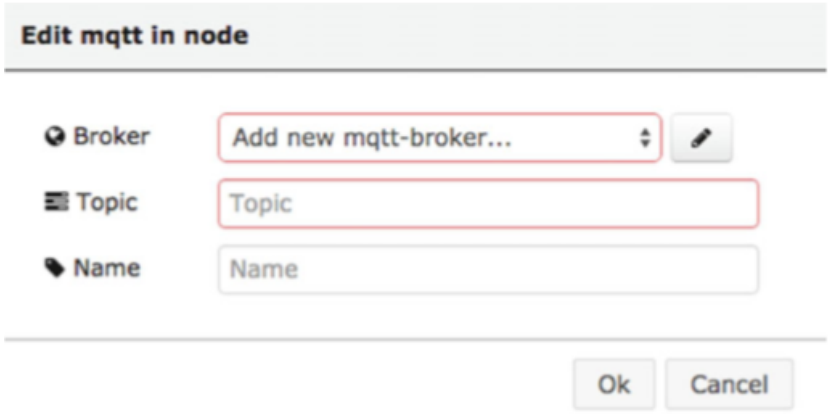


Gambar 10-24. Ganti nama lembar alur

Drag dan lepas simpul masukan mqtt dari palet di tab aliran; alur Anda akan terlihat seperti Gambar 10-25.

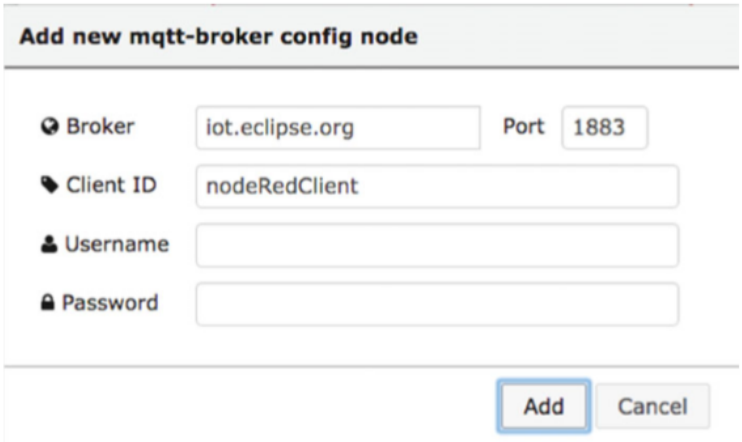
Gambar 10-25. Node subscribe MQTT

Klik dua kali node mqtt untuk membuka kotak dialog properti, seperti yang ditunjukkan pada Gambar 10-26. Anda perlu mengonfigurasi broker MQTT baru, pilih Add New mqtt-Broker… dari bidang Broker dan klik ikon Pencil.



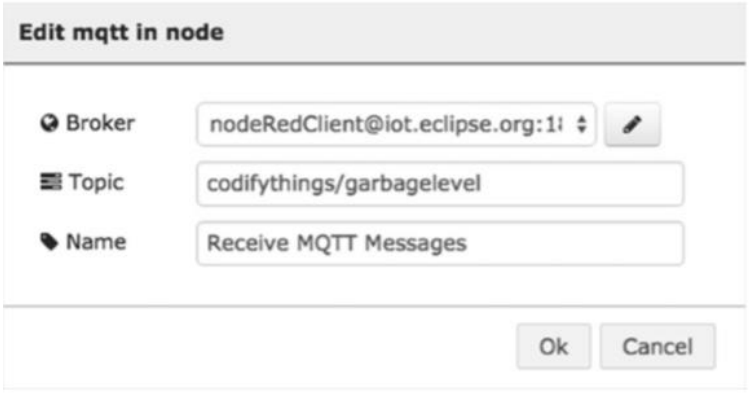
Gambar 10-26. MQTT dan properti ode

Kotak dialog konfigurasi MQTT akan terbuka, seperti yang ditunjukkan pada Gambar 10-27. Anda perlu mengonfigurasi broker MQTT yang sama tempat Arduino menerbitkan pesan, yang dalam hal ini adalah broker yang tersedia untuk umum dari Eclipse Foundation. Masukkan **iot.Eclipse.org** di bidang Broker, 1883 di bidang Port, dan **nodeRedClient** di bidang ID Klien. Klik Add untuk menambahkan broker MQTT yang baru dikonfigurasi.



Gambar 10-27. Konfigurasi broker MQTT

Sekarang setelah Anda mengonfigurasi broker MQTT, Anda harus memasukkan topik yang harus dilanggani oleh node mqtt Anda. Karena Arduino menerbitkan ke kodifikasi/tingkat sampah, Anda harus memasukkan yang sama di bidang Topik. Update nama menjadi **Receive MQTT Messages**, seperti yang ditunjukkan pada Gambar 10-28, dan klik **OK** untuk menyimpan perubahan.

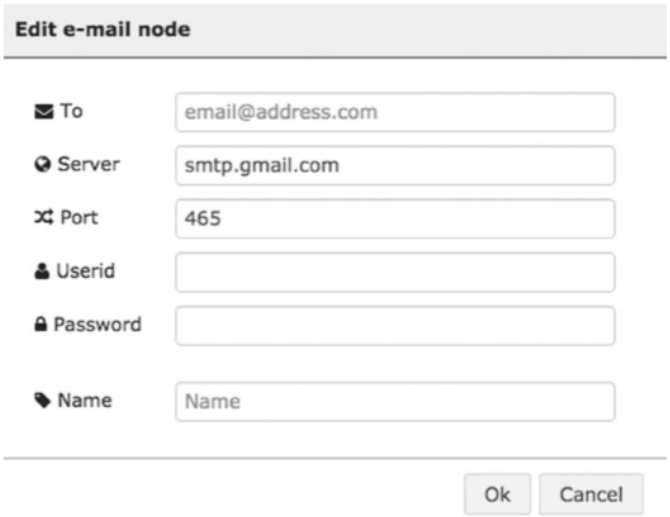


Gambar 10-28. Topik Broker MQTT

Drag dan lepas simpul email dari palet sosial dan letakkan di tab aliran setelah simpul Terima Pesan MQTT. Alur Anda akan terlihat mirip dengan Gambar 10-29 pada saat ini.

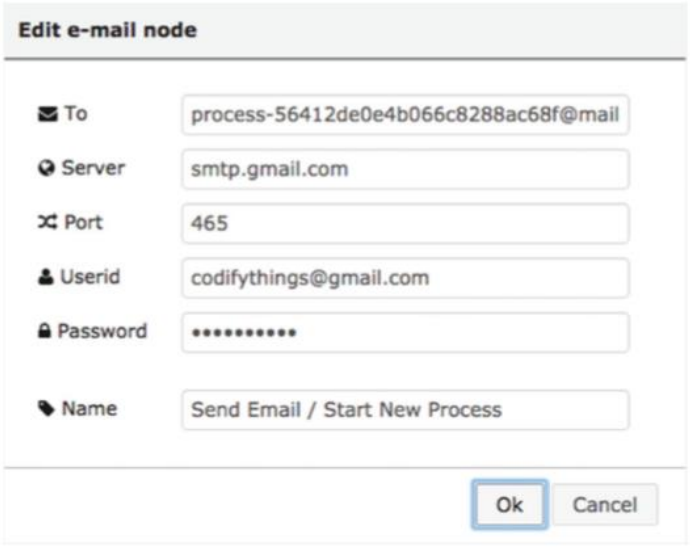
Gambar 10-29. Node email

Node email memungkinkan Anda mengirim pesan email ke alamat yang diberikan. Klik dua kali node email untuk membuka kotak dialog properti yang ditunjukkan pada Gambar 10-30.



Gambar 10-30. Properti simpul email

Update properti node email seperti yang ditunjukkan pada Gambar 10-31. Pada kolom To, masukkan alamat email yang dibuat secara otomatis oleh **Effketif BPM**. Pada kolom Server, **Port, Userid**, dan **Password**, berikan informasi tentang server SMTP yang dapat digunakan **Node-RED** untuk mengirim email ini. Secara default, node email memiliki properti Gmail. Update opsi Nama untuk **Send Email/Start New Process**. Klik **OK** untuk menyimpan perubahan Anda.



Gambar 10-31. Properti node email yang diUpdate

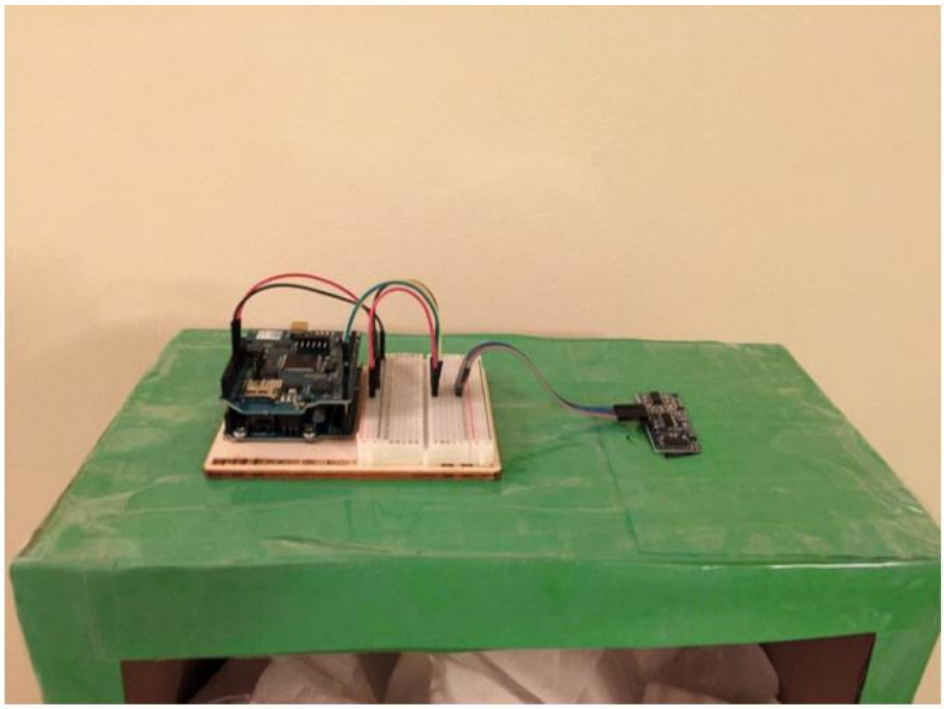
Anda telah menambahkan semua node yang diperlukan, jadi sekarang Anda dapat menghubungkan node Terima Pesan MQTT ke node **Send Email/Start New Process**. Ini melengkapi alur NodeRED Anda dan akan terlihat seperti Gambar 10-32. Klik tombol Deploy untuk membuat alur ini tersedia.

Gambar 10-32. Aliran Node-RED akhir

Gambar 10-33. Sirkuit sistem pengelolaan sampah

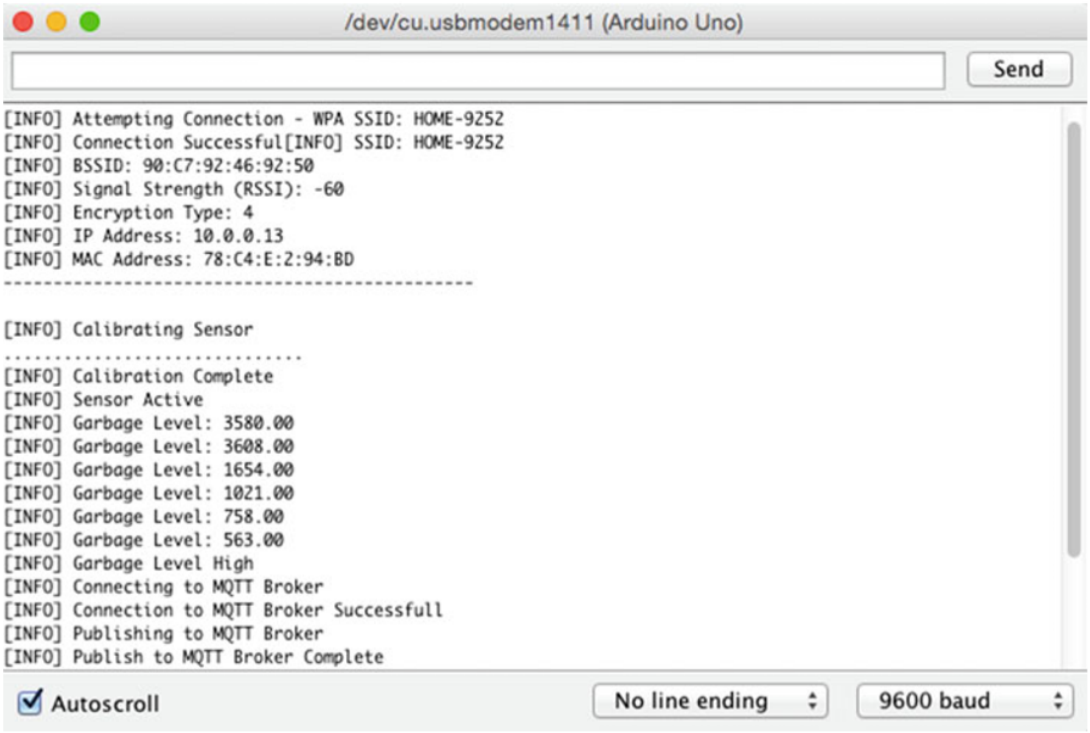
10.6 PRODUK AKHIR

Untuk menguji aplikasi, verifikasi dan Upload kode Arduino seperti yang dibahas pada Bab 1. Tempatkan sensor jarak Anda di atas tempat sampah atau kotak kardus kosong, seperti yang ditunjukkan pada Gambar 10-33 dan 10-34. Pastikan tidak ada sampah di kaleng awalnya.



Gambar 10-34. *Close-up* dari sirkuit sistem pengelolaan limbah

Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log yang mirip dengan yang ditunjukkan pada Gambar 10-35.

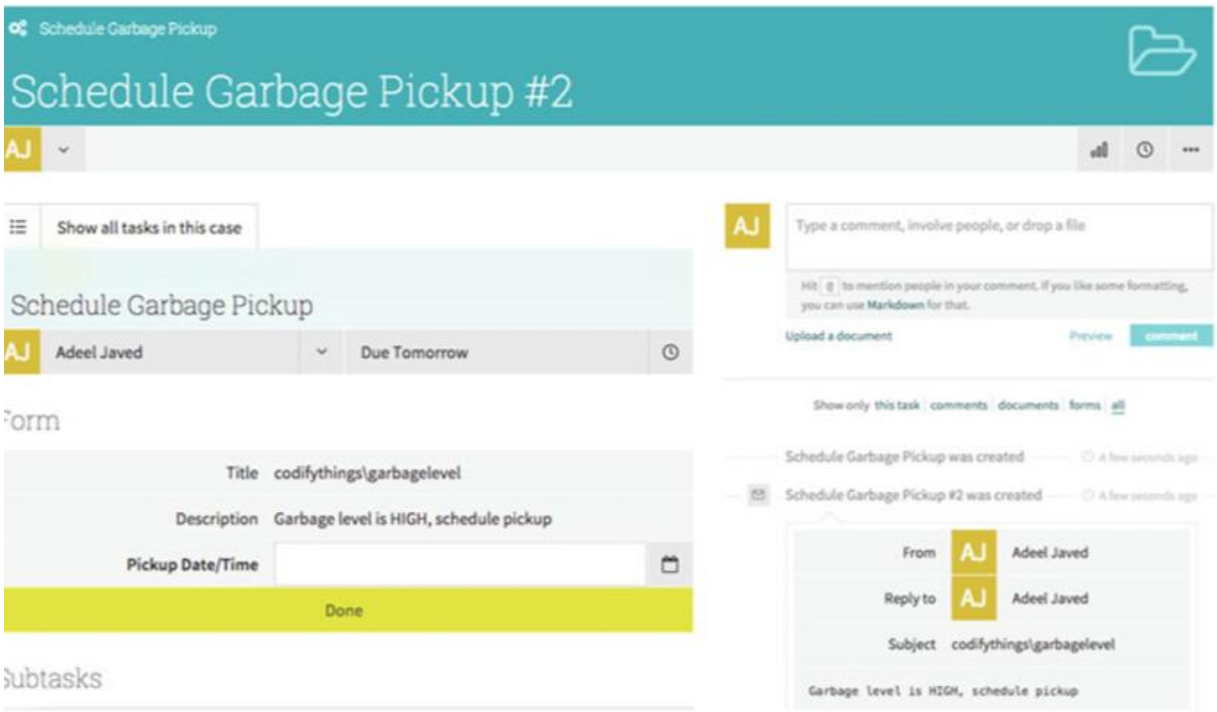


Gambar 10-35. Pesan log dari sistem pengelolaan limbah

Server Node-RED Anda harus aktif dan berjalan dan aliran sistem pengelolaan limbah harus diterapkan. Komponen terakhir adalah proses Effektif, dan Anda seharusnya sudah mempublishkannya di langkah sebelumnya. Mulailah menambahkan barang ke tempat sampah/kotak Anda, dan segera setelah mencapai level tertentu (ditetapkan pada 700 dalam kode Arduino), sebuah pesan akan dipublishkan ke broker MQTT. Alur NodeRED Anda mendengarkan broker MQTT untuk pesan baru dan, segera setelah menerima yang baru, email akan dikirim yang memulai proses Effektif. Masuk ke Effektif menggunakan kredensial Anda dan, seperti yang ditunjukkan pada Gambar 10-36, Anda akan melihat tugas baru untuk proses **Pickup process available** yang tersedia di bawah tab Tugas.

Gambar 10-36. Tugas baru tersedia di Effektif

Klik tautan tugas untuk melihat detailnya. Seperti yang Anda lihat pada Gambar 10-37, Judul berisi nama topik MQTT dan Deskripsi berisi pesan yang dikirim oleh Arduino. Masukkan tanggal/waktu pengambilan dan klik Selesai. Ini akan menyelesaikan proses dan tugas akan dihapus dari Listing Tugas Anda.



Gambar 10-37. Detail tugas di Effektif

Ini melengkapi pengujian ujung-ke-ujung proyek ini.

10.7 RINGKASAN

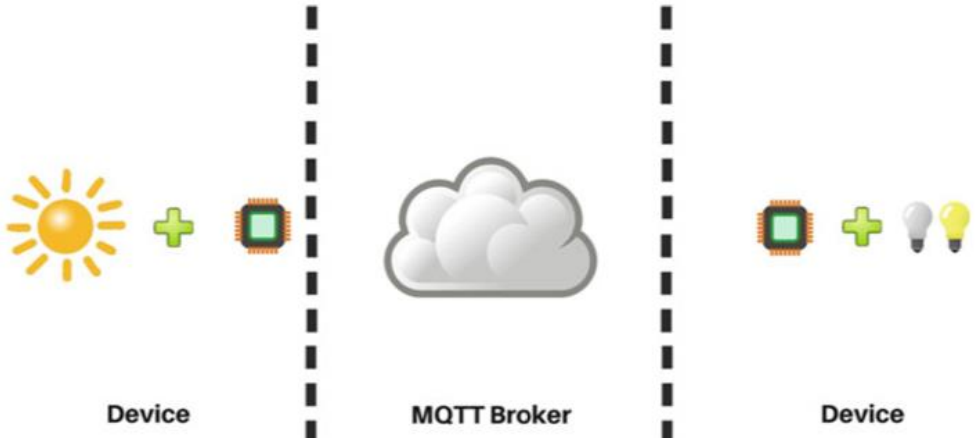
Dalam bab ini, Anda mempelajari tentang pola IoT yang akan digunakan saat respons manusia diperlukan untuk peringatan yang dihasilkan perangkat. Proyek yang Anda kembangkan adalah salah satu contoh di mana Arduino mengirimkan peringatan dan sebuah proses dimulai sebagai tanggapan terhadapnya.

Memulai suatu proses hanyalah salah satu cara untuk merespons. Proses memberi Anda respons yang lebih ramping dan terstruktur, tetapi tergantung pada persyaratannya, Anda selalu dapat menggunakan email, SMS, dll. Karena persyaratan peraturan atau kurangnya teknologi respons, intervensi manusia akan terus menjadi persyaratan untuk beberapa IoT aplikasi. Seiring kemajuan IoT, jumlah intervensi manusia juga akan berkurang.Dalam bab ini, Anda mempelajari tentang pola IoT yang akan digunakan saat respons manusia diperlukan untuk peringatan yang dihasilkan perangkat. Proyek yang Anda kembangkan adalah salah satu contoh di mana Arduino mengirimkan peringatan dan sebuah proses dimulai sebagai tanggapan terhadapnya.

Memulai suatu proses hanyalah salah satu cara untuk merespons. Proses memberi Anda respons yang lebih ramping dan terstruktur, tetapi tergantung pada persyaratannya, Anda selalu dapat menggunakan email, SMS, dll. Karena persyaratan peraturan atau kurangnya teknologi respons, intervensi manusia akan terus menjadi persyaratan untuk beberapa IoT aplikasi. Seiring kemajuan IoT, jumlah intervensi manusia juga akan berkurang.

BAB 11
POLA IoT: MACHINE to MACHINE

Ketika teknologi IoT berkembang dan mesin menjadi lebih pintar dan lebih mampu, kebutuhan akan intervensi manusia akan berkurang. Mesin akan dapat secara mandiri menanggapi peringatan yang dihasilkan oleh mesin lain. Dalam bab ini, Anda akan membangun sistem konservasi energi yang akan menunjukkan bagaimana dua mesin dapat berkomunikasi. Gambar 11-1 menunjukkan diagram tingkat tinggi dari semua komponen yang terlibat dalam sistem ini. Komponen pertama adalah perangkat Arduino yang memonitor tingkat kecerahan cahaya dan mengirimkan peringatan setiap kali tingkat rendah. Komponen kedua adalah broker MQTT yang membantu menghindari komunikasi point-to-point. Beberapa perangkat dapat berkomunikasi satu sama lain tanpa mengetahui identitas masing-masing.



Gambar 11-1. Komponen sistem konservasi energi

Komponen terakhir adalah perangkat Arduino lain yang mengontrol lampu. Jika sensor cahaya mengirimkan pesan ke broker MQTT bahwa tingkat kecerahan cahaya LOW, perangkat akan secara otomatis menyalakan lampu. Dalam kehidupan nyata, sistem ini dapat digunakan untuk menyalakan atau mematikan lampu jalan hanya ketika diperlukan, bukan melakukannya pada waktu yang dijadwalkan, terlepas dari seberapa terangnya.

11.1 TUJUAN PEMBELAJARAN

Di akhir bab ini, Anda akan dapat:

- Membaca data dari sensor cahaya
- Publishkan pesan ke broker MQTT
- Kontrol LED
- Subscribe Arduino ke broker MQTT

11.2 PERANGKAT SENSOR CAHAYA

Komponen pertama dari aplikasi IoT Anda adalah perangkat Arduino yang akan memantau tingkat kecerahan cahaya dan menerbitkan pesan saat rendah.

***Catatan** : Anda telah membuat rangkaian ini di Bab 4, jadi untuk persyaratan hardware dan software serta instruksi rangkaian, lihat kembali Bab 4. Perubahan hanya pada kode Arduino, yang dalam hal ini menerbitkan pesan ke broker MQTT alih-alih memulai Aliran node-RED.*

11.3 KODE (ARDUINO)

Selanjutnya Anda akan menulis kode untuk menghubungkan Arduino ke Internet menggunakan WiFi, membaca data sensor cahaya, dan memublikasikannya ke broker MQTT. Mulai Arduino IDE Anda dan ketik kode yang disediakan di sini atau unduh dari situs dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi lima bagian:

- Library eksternal
- Konektivitas Internet (Wi-Fi)
- Baca data sensor
- MQTT (terbitkan)
- Fungsi standar

Library Eksternal

Bagian pertama dari kode, seperti yang disediakan dalam Listing 11-1, mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Sketsa ini memiliki dua dependensi utama—untuk konektivitas Internet, Anda perlu menyertakan <WiFi.h> (dengan asumsi Anda menggunakan pelindung WiFi), dan untuk komunikasi broker MQTT, Anda harus menyertakan <PubSubClient.h>.

Listing 11-1. Kode untuk Menyertakan Dependensi Eksternal

```
#include <SPI.h>
#include <WiFi.h>
#include <PubSubClient.h>
```

Konektivitas Internet (Nirkabel)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (dalam Bab 2) di sini.

Baca Data Sensor

Bagian kode ketiga, seperti yang disediakan dalam Listing 11-2, mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membaca data sensor. Fungsi readSensorData() membaca data dari Pin Analog A0 ; hasilnya antara 0 dan 1023. Semakin besar nilai yang dikembalikan, semakin terang sumber cahayanya. Nilai sensor cahaya ditetapkan ke variabel lightValue. Berdasarkan variabel lightValue, nilai LOW atau HIGH yang sesuai diteruskan sebagai parameter ke fungsi publishSensorData().

Listing 11-2. Kode untuk Membaca Data Sensor Cahaya

```
int lightValue;
void readSensorData()
{
    //Read Light Sensor Value
    lightValue = analogRead(A0);
    Serial.print("[INFO]
    Light Sensor Reading: ");
    Serial.println(lightValue);
    if(lightValue < 500)
    {
    publishSensorData("LOW");
    }
    else
    {
    publishSensorData("HIGH");
```

```
        }
        Serial.println("-----------------------------------------");
    }
```

Publishkan Data

Bagian keempat dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk memublikasikan data ke broker MQTT (untuk detailnya, lihat Bab 3). Kode ini mirip dengan apa yang Anda lihat di Bab 3. Ada beberapa perubahan yang perlu Anda lakukan. Semua perubahan disorot dalam huruf tebal di Listing 11-3. Pastikan untuk mengubah variabel server, port, dan topik serta nama klien yang harus Anda lewati saat terhubung ke broker MQTT. Perubahan utama lainnya mencakup kondisi IF/ELSE yang memublikasikan pesan berbeda berdasarkan parameter lightLevel yang diteruskan oleh fungsi readSensorData().

Listing 11-3. Kode untuk Menerbitkan Pesan MQTT

```
// IP address of the MQTT broker
char server[] = {"iot.eclipse.org"};
int port = 1883;
char topic[] = {"codifythings/lightlevel"};
void callback(char* topic, byte* payload, unsigned int length)
{
//Handle message arrived
}
PubSubClient pubSubClient(server, port, 0, client);
void publishSensorData( String lightLevel )
{
    // Connect MQTT Broker
    Serial.println("[INFO] Connecting to MQTT Broker");
    if (pubSubClient.connect( "arduinoIoTClient"))
    {
    Serial.println("[INFO] Connection to MQTT Broker Successful");
    }
    Else
    {
    Serial.println("[INFO] Connection to MQTT Broker Failed");
}
      // Publish to MQTT Topic
      if (pubSubClient.connected())
      {
      Serial.println("[INFO] Publishing to MQTT Broker");
      if(lightLevel == "LOW")
      {
      Serial.println("[INFO] Light Level is LOW");
      pubSubClient.publish(topic, "LOW");
      }
      Else
      {
      Serial.println("[INFO] Light Level is HIGH");
      pubSubClient.publish(topic, "HIGH");
      }
```

```
        Serial.println("[INFO] Publish to MQTT Broker Complete");
        }
        Else
        {
Serial.println("[ERROR] Publish to MQTT Broker Failed");
        }
        pubSubClient.disconnect();
}
```

Fungsi Standar Bagian terakhir disediakan dalam Listing 11-4. Ini mengimplementasikan fungsi setup() dan loop() standar Arduino. Fungsi setup() menginisialisasi port serial dan menghubungkan ke Internet. Fungsi loop() memanggil readSensorData() saja, karena secara internal memanggil fungsi publishSensorData() saat tingkat cahaya rendah.

Listing 11-4. Kode untuk Fungsi Arduino Standar

```
void setup()
{
   // Initialize serial port
   Serial.begin(9600);
   // Connect Arduino to internet
   connectToInternet();
}
void loop()
{
   // Read sensor data
   readSensorData();
   // Delay
   delay(5000);
}
```

Kode Arduino Anda untuk perangkat sensor cahaya sekarang selesai.

11.4 PERANGKAT KONTROL PENCAHAYAAN

Komponen lain dari aplikasi IoT Anda adalah perangkat Arduino yang akan menyalakan atau mematikan lampu kontrol tergantung pada pesan yang diterima dari broker MQTT. Sirkuit dan kode untuk perangkat ini pada dasarnya sama dengan sirkuit dan perangkat yang Anda kembangkan di Bab 6.

***Catatan** : Anda telah membuat rangkaian ini di Bab 6, jadi untuk persyaratan hardware dan software serta instruksi rangkaian, lihat Bab 6. Perubahan hanya pada kode Arduino, yang dalam hal ini menggunakan logika yang berbeda untuk mempublishkan pesan ke broker MQTT*

Kode (Arduino)

Selanjutnya Anda akan menulis kode untuk menghubungkan Arduino ke Internet menggunakan WiFi, subscribe broker MQTT, dan mengontrol LED yang terpasang. Mulai Arduino IDE Anda dan ketik kode yang disediakan di sini atau unduh dari situs dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi lima bagian:

- Library eksternal
- Konektivitas Internet (Wi-Fi)
- MQTT (subscribe)
- Kontrol LED

- Fungsi standar

Library Eksternal

Bagian pertama kode, seperti yang disediakan dalam Listing 11-5, mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Sketsa ini memiliki dua dependensi utama—untuk konektivitas Internet, Anda perlu menyertakan <WiFi.h> (dengan asumsi Anda menggunakan pelindung WiFi) dan untuk komunikasi broker MQTT, Anda harus menyertakan <PubSubClient.h>.

Listing 11-5. Kode untuk Menyertakan Dependensi Eksternal

```
#include <SPI.h>
#include <WiFi.h>
#include <PubSubClient.h>
```

Konektivitas Internet (Nirkabel)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (dalam Bab 2) di sini.

Data Subscribe

Bagian ketiga dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke broker MQTT dan panggilan balik ketika pesan baru tiba (untuk detailnya, lihat Bab 3).

Kode ini mirip dengan apa yang Anda lihat di Bab 3. Hanya ada beberapa perubahan yang perlu Anda buat agar kode berfungsi. Semua perubahan telah disorot dalam huruf tebal di Listing 11-6. Pastikan untuk mengubah nilai variabel server, port, dan topik ke nilai server MQTT Anda.

Setiap kali pesan baru diterima, fungsi callback(...) dipanggil. Ini mengekstrak muatan dan memanggil fungsi turnLightsOnOff(). Satu tambahan untuk kode ini adalah kondisi IF/ELSE, yang memeriksa nilai payloadContent dan jika LOW, mengirimkan ON sebagai parameter ke fungsi turnLightsOnOff(...). Jika tidak, OFF dikirim sebagai parameter.

Listing 11-6. Kode untuk Subscribe ke Broker MQTT

```
// IP address of the MQTT broker
char server[] = {"iot.eclipse.org"};
int port = 1883;
char topic[] = {"codifythings/lightlevel"};
PubSubClient pubSubClient(server, port, callback, client);
void callback(char* topic, byte* payload, unsigned int length)
{
    // Print payload
    String payloadContent = String((char *)payload);
    Serial.println("[INFO] Payload: " + payloadContent);
    if(payloadContent.substring(0,3) == "LOW")
    {
    // Turn lights on/off
    turnLightsOnOff("ON");
    }
    else
    {
    // Turn lights on/off
    turnLightsOnOff("OFF");
    }   }
```

11.5 LAMPU KONTROL

Bagian keempat dari kode, seperti yang disediakan dalam Listing 11-7, mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk mengontrol LED. Kode ini mengubah status LED berdasarkan nilai parameter tindakan.

Listing 11-7. Kode untuk Mengontrol LED

```
int ledPin = 3;
void turnLightsOnOff(String action)
{
    // Check if lights are currently on or off
    if(action == "ON")
    {
    //Turn lights on
    Serial.println("[INFO] Turning lights on");
    digitalWrite(ledPin, HIGH);
    }
    else
    {
    // Turn lights off
    Serial.println("[INFO] Turning lights off");
    digitalWrite(ledPin, LOW);
    }
}
```

Fungsi Standar

Bagian kode terakhir disediakan di Listing 11-8. Ini mengimplementasikan fungsi setup() dan loop() standar Arduino. Fungsi setup() menginisialisasi port serial, menghubungkan ke internet, dan subscribe topik MQTT. Broker MQTT telah diinisialisasi dan subscribe, jadi dalam fungsi loop(), Anda hanya perlu menunggu pesan baru dari broker MQTT.

Listing 11-8. Kode untuk Fungsi Arduino Standar

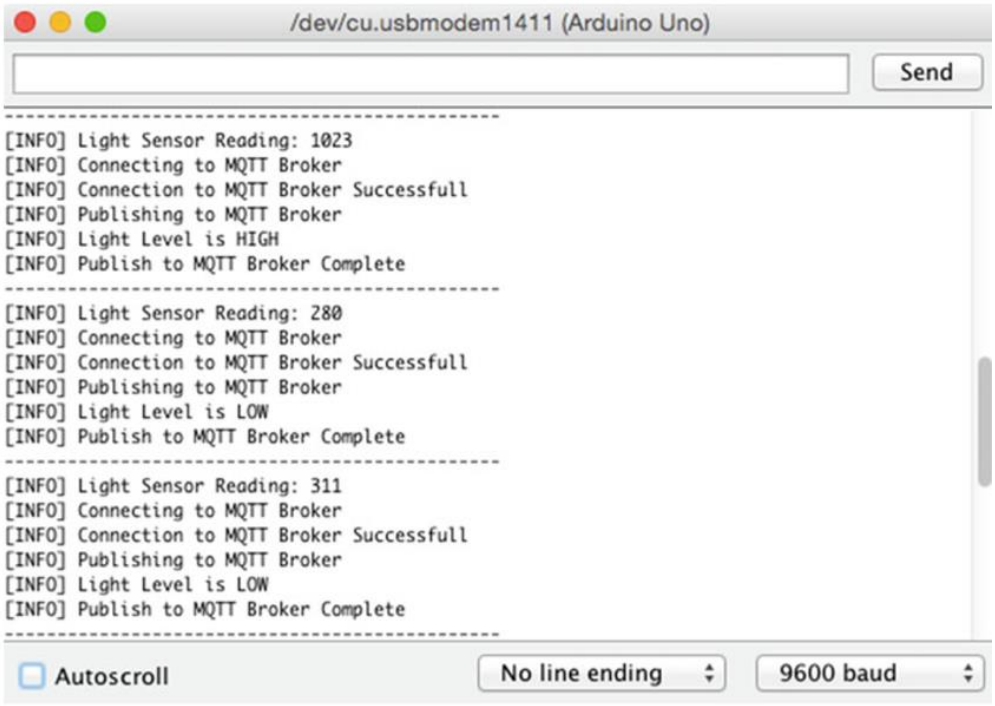
```
void setup()
{
    // Initialize serial  port
    Serial.begin(9600);
    // Connect Arduino to internet
    connectToInternet();
    // Set LED pin mode
    pinMode(ledPin, OUTPUT);
    //Connect MQTT Broker
    Serial.println("[INFO] Connecting to MQTT Broker");
    if (pubSubClient.connect("arduinoClient"))
    {
    Serial.println("[INFO] Connection to MQTT Broker Successful");
    pubSubClient.subscribe(topic);
    }
    Else
    {
    Serial.println("[INFO] Connection to MQTT Broker Failed");
    }
```

```
}
void loop()
{
// Wait for messages from MQTT broker
pubSubClient.loop();
}
```

Kode Arduino Anda untuk perangkat kontrol pencahayaan sekarang selesai.

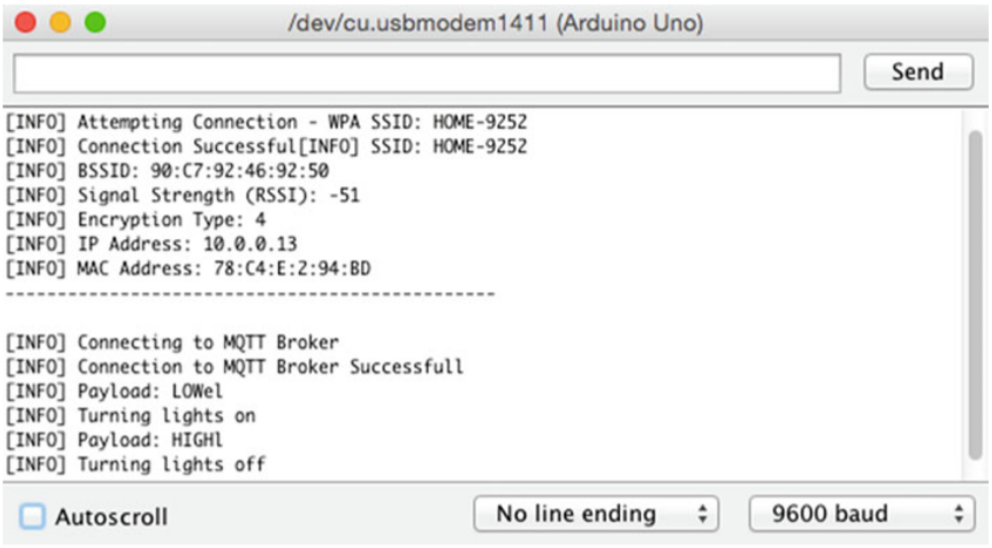
11.6 PRODUK AKHIR

Untuk menguji aplikasi, pastikan kedua perangkat Anda—perangkat sensor cahaya dan perangkat kontrol pencahayaan—dinyalakan dan kode telah diterapkan (lihat Bab 1 untuk proses penerapan). Buka jendela Serial Monitor untuk kedua perangkat Anda. Gambar 11-2 menunjukkan jendela Serial Monitor dengan pesan log yang dihasilkan dari perangkat sensor cahaya. Segera setelah Anda memindahkan perangkat ini dari cahaya terang ke area gelap, itu akan menerbitkan pesan ke broker MQTT.



Gambar 11-2. Pesan log dari perangkat sensor cahaya

Gambar 11-3 menunjukkan jendela Serial Monitor dengan pesan log yang dihasilkan dari perangkat kontrol pencahayaan. Segera setelah perangkat sensor cahaya menerbitkan pesan, perangkat kontrol pencahayaan akan menyalakan LED. Jika Anda memindahkan perangkat sensor cahaya kembali ke area yang lebih terang, perangkat kontrol pencahayaan akan mematikan LED.



Gambar 11-3. Pesan log dari perangkat kontrol pencahayaan

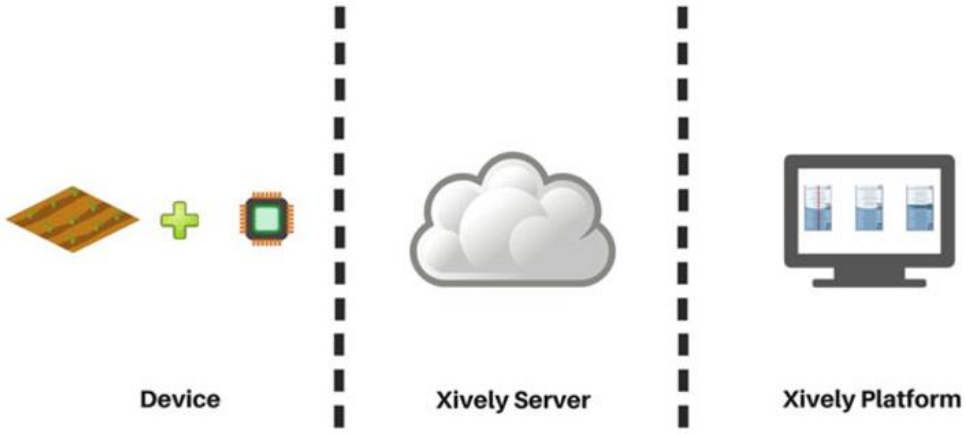
11.7 RINGKASAN

Dalam bab ini Anda telah mempelajari cara membuat beberapa perangkat berkomunikasi satu sama lain menggunakan broker MQTT. Broker seperti MQTT menghilangkan kebutuhan akan komunikasi langsung. Sebuah perangkat memublikasikan pesan yang dapat diterima oleh semua perangkat atau sistem yang tertarik dengan pesan tersebut dan merespons sesuai dengan itu. Pola mesin-mesin pasti memberikan manfaat maksimal di ruang IoT. Perbatasan berikutnya dalam bidang ini tentu saja mengembangkan perangkat AI (kecerdasan buatan) yang dapat belajar dan beradaptasi dengan lingkungan yang selalu berubah.

BAB 12
PLATFORM IoT

Platform IoT memberi pengembang kemampuan untuk mengembangkan, menyebarkan, dan mengelola aplikasi IoT mereka dari satu lokasi pusat dengan cara yang aman. Platform IoT mempercepat proses pengembangan dengan menyediakan alat yang diperlukan di lingkungan berbasis cloud, yang berarti pengembang tidak menghabiskan waktu untuk penyiapan. Platform IoT yang baik idealnya mencakup sebagian besar alat yang telah kita bahas dalam 11 bab sebelumnya, seperti broker MQTT, server HTTP, dukungan REST API, database untuk menyimpan data sensor, Node-RED untuk orkestrasi kompleks, lokasi perangkat, aman komunikasi, pelaporan, analitik, dan alat yang mudah digunakan untuk membangun aplikasi web dan seluler.

Bab ini membahas **platform IoT** populer yang disebut **Xively**. Anda akan membangun sistem kontrol kelembaban tanah yang mengirimkan peringatan email setiap kali tingkat kelembaban tanah turun di bawah ambang batas tertentu. Gambar 12-1 menunjukkan diagram tingkat tinggi dari semua komponen yang terlibat dalam sistem ini. Komponen pertama adalah perangkat Arduino yang memonitor tingkat kelembaban tanah dan mengirimkan pesan ke Xively. Komponen kedua dan ketiga berada di platform Xively. Dengan beberapa konfigurasi dasar, platform akan dapat menerima, menyimpan, dan menampilkan data yang dikirim oleh sensor.



Gambar 12-1. Komponen sistem kontrol kelembaban minyak s

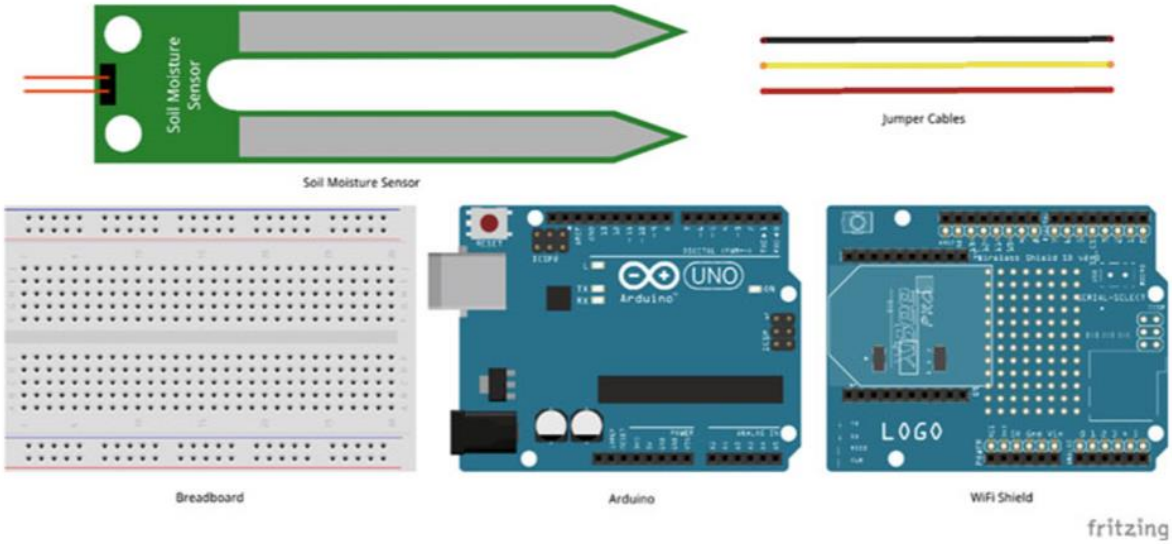
12.1 TUJUAN PEMBELAJARAN

Di akhir bab ini, Anda akan dapat:

- Baca data sensor kelembaban tanah
- Siapkan Xively untuk menerima data sensor kelembapan
- Siapkan trigger di Xively untuk mengirim email menggunakan tugas Zapier
- Tulis kode untuk membaca data sensor kelembapan dan publishkan ke Xively

Hardware yang Diperlukan

Gambar 12-2 memberikan Listing semua komponen hardware yang diperlukan untuk membangun sistem kontrol kelembaban tanah.



Gambar 12-2. Hardware yang diperlukan untuk sistem kontrol kelembaban

Software yang Diperlukan

Untuk mengembangkan sistem kontrol kelembaban tanah ini, Anda memerlukan software berikut:

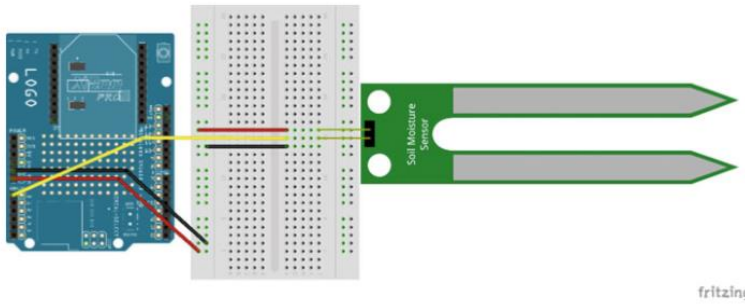
- Arduino IDE 1.6.4 atau lebih baru
- Xively (dihosting)
- Zapier (dihosting)]

Sirkuit

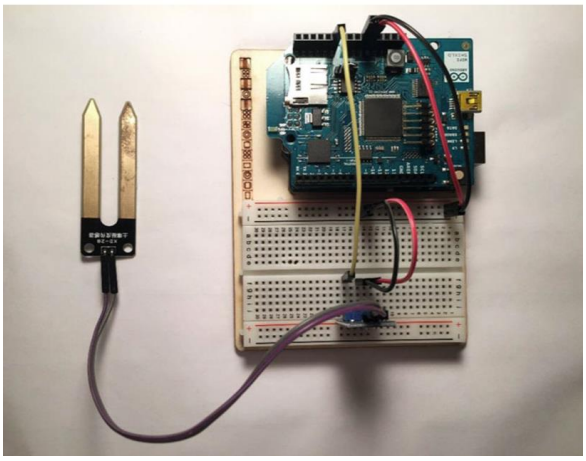
Pada bagian ini, Anda akan membangun sirkuit yang diperlukan untuk sistem kontrol kelembaban tanah. Rangkaian ini menggunakan sensor kelembaban tanah untuk mendeteksi jumlah kelembaban dalam tanah.

1. Pastikan Arduino Anda tidak terhubung ke sumber listrik, seperti ke komputer melalui USB atau baterai.
2. Pasang pelindung WiFi ke bagian atas Arduino.
3. Gunakan kabel jumper untuk menghubungkan port power (5V) dan ground (GND) pada Arduino ke port power (+) dan ground (-) pada breadboard.
4. Sekarang papan breadboard Anda memiliki sumber daya, gunakan kabel jumper untuk menghubungkan port daya (+) dan arde (-) papan breadboard Anda ke port daya dan arde dari sensor kelembaban.
5. Untuk membaca nilai sensor kelembaban, Anda perlu menghubungkan kabel jumper dari port analog sensor kelembaban ke port A0 (Analog) Arduino Anda. Kode Anda akan membaca tingkat kelembapan dari port ini. Sensor mengembalikan nilai antara 0 dan 1023. Nilai yang lebih tinggi sesuai dengan tingkat kelembaban tanah yang lebih rendah.

Sirkuit Anda sekarang telah selesai dan akan terlihat seperti Gambar 12-3 dan 12-4.



Gambar 12-3. Diagram sirkuit sistem kontrol kelembaban tanah



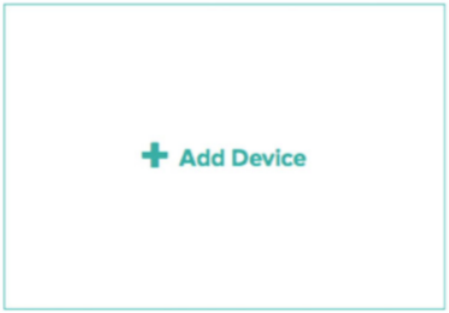
Gambar 12-4. Sirkuit sebenarnya dari sistem kontrol kelembaban tanah

12.2 PENGATURAN XIVELY

Seperti yang disebutkan sebelumnya, Xively adalah platform IoT yang populer. Untuk menggunakan Xively, Anda harus terlebih dahulu menyiapkan akun gratis di https://personal.xively.com/. Setelah akun Anda diatur, masuk ke Xively. Setelah login, Anda akan diarahkan ke dashboard akun Anda. Klik DEVELOP dari bilah menu di atas, seperti yang ditunjukkan pada Gambar 12-5.

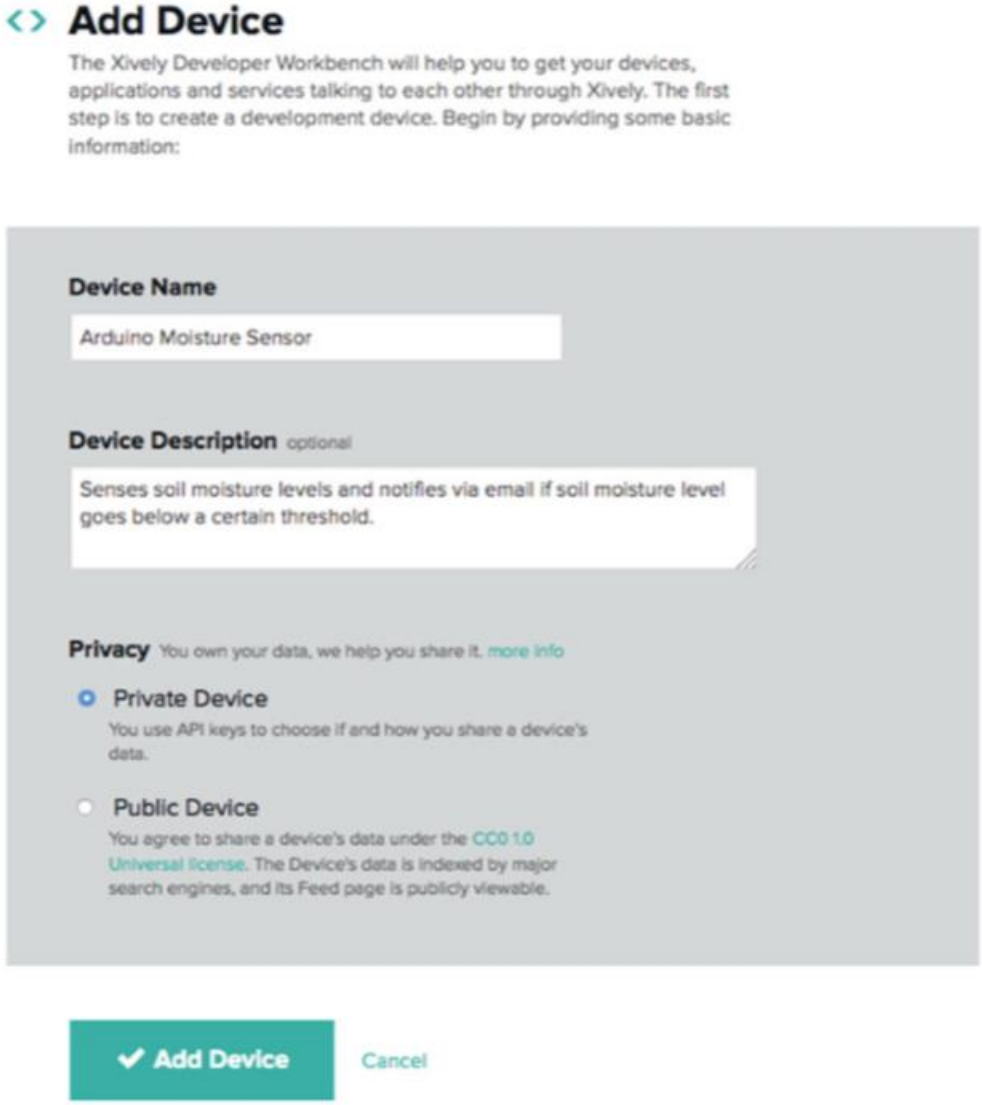
Gambar 12-5. Dasbor akun Xively

Add device pengembangan baru, seperti yang ditunjukkan pada Gambar 12-6, dengan mengeklik tautan **+ Add Device**.



Gambar 12-6. Add device pengembangan baru

Pada layar penyiapan perangkat, masukkan nama perangkat dan deskripsi perangkat, seperti yang disediakan pada Gambar 12-7. Jaga privasi perangkat Anda disetel ke Perangkat Pribadi. Klik **Add Device** untuk menyelesaikan langkah ini.



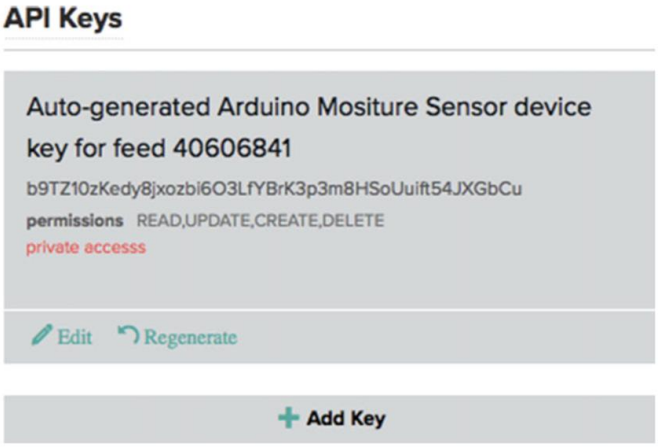
Gambar 12-7. pengaturan perangkat

Xively akan secara otomatis menghasilkan kunci API unik dan ID Umpan, dan keduanya diperlukan dalam kode Arduino. Anda dapat menemukan ID Umpan di sisi kanan atas dasbor (lihat Gambar 12-8).

Gambar 12-8. ID umpan

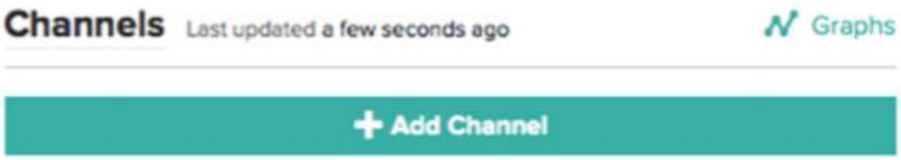
Seperti yang disebutkan sebelumnya, Xively secara otomatis menghasilkan kunci API, tetapi Anda juga memiliki opsi untuk menambahkan kunci Anda sendiri. Dalam proyek ini

Anda akan menggunakan kunci API yang dibuat secara otomatis. Anda dapat menemukan kunci API di bagian Kunci API di dasbor (lihat Gambar 12-9).



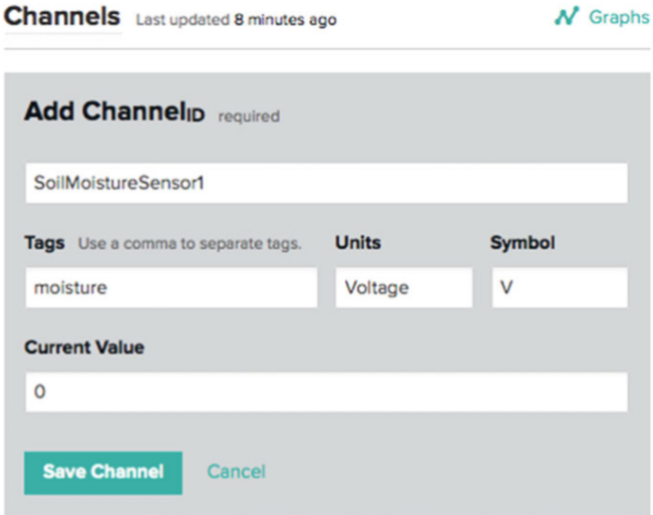
Gambar 12-9. Kunci PI (dibuat secara otomatis dan dibuat khusus)

Selanjutnya Anda akan membuat saluran. Sebuah channel akan dipetakan langsung ke sebuah sensor, yaitu data dari sebuah sensor akan diterima dan disimpan oleh sebuah channel. Seperti yang ditunjukkan pada Gambar 12-10, Klik **button Add Channel** dari bagian Saluran.



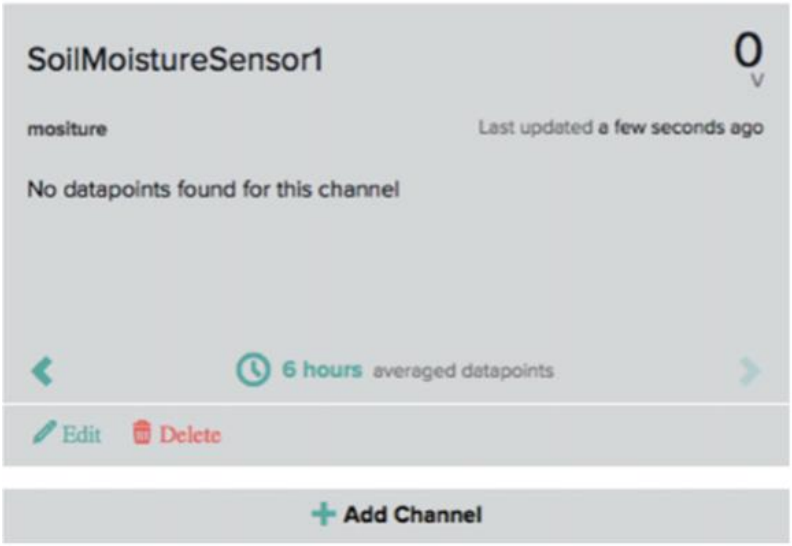
Gambar 12-10. saluran tambahan

Masukkan nilai saluran Anda, seperti yang ditunjukkan pada Gambar 12-11. ID Saluran adalah satu-satunya bidang yang diperlukan, dan seperti yang akan Anda lihat di bagian selanjutnya, ID saluran juga digunakan dalam kode Arduino untuk mengirim data sensor. Jika Anda memiliki beberapa saluran, maka Tag akan membantu Anda mencari. Bidang Unit dan Simbol akan digunakan saat menampilkan data. Nilai Saat Ini juga digunakan saat menampilkan data saat grafik Anda dimulai dari titik ini. Klik **save** Saluran untuk menyelesaikan penyiapan saluran.



Gambar 12-11. Pengaturan saluran baru

Setelah Anda menyimpan pengaturan saluran, Xively siap menerima data sensor.



Gambar 12-12 menunjukkan bagian tempat data setiap sensor akan ditampilkan.

Jika lokasi perangkat penting, Anda juga dapat menyetelnya dari bagian Lokasi, seperti yang ditunjukkan pada Gambar 12-13. Dalam proyek ini, Anda tidak akan mengubahnya dari kode, jadi ini hanya statis yang akan muncul di dasbor.

Gambar 12-13. Menambahkan lokasi

Klik **Add Location** dan masukkan nama lokasi dan alamat tempat sensor Anda secara fisik berada. Data lokasi selalu berguna untuk tujuan pemeliharaan. **Klik save**. Untuk saat ini, tidak ada lagi pengaturan yang diperlukan di Xively.

12.3 SETUP ZAPIER

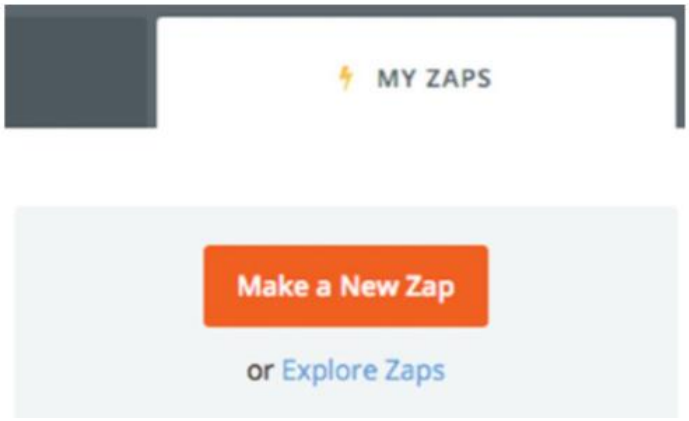
Xively mendukung memicu tugas eksternal; misalnya, jika nilai saluran melewati ambang tertentu, Anda dapat menjalankan tugas Anda sendiri. Xively menggunakan HTTP POST untuk memicu tugas eksternal. Semua data akan dikirimkan ke penerima menggunakan metode HTTP POST (lihat Bab 3 untuk detail selengkapnya tentang HTTP POST).

Data Xively tersedia melalui HTTP dan dapat digunakan untuk mengembangkan dasbor khusus dan menghasilkan peringatan. Beberapa aplikasi IoT yang Anda kembangkan di Bab 7, 8, dan 9 memiliki komponen HTTP. Anda akan kehilangan keuntungan menggunakan platform IoT jika Anda akhirnya menulis kode khusus. Untuk menghasilkan trigger di Xively, Anda dapat menghindari semua pengkodean hanya dengan menggunakan tugas Zapier. Zapier adalah alat berbasis web yang memungkinkan Anda mengotomatiskan tugas jika/maka. Anda membuat tugas (alias, Zap) yang memerlukan trigger dan tindakan yang sesuai.

***Catatan** : Anda juga dapat memicu tugas Zapier dari Arduino menggunakan metode HTTP POST yang dibahas di Bab 3.*

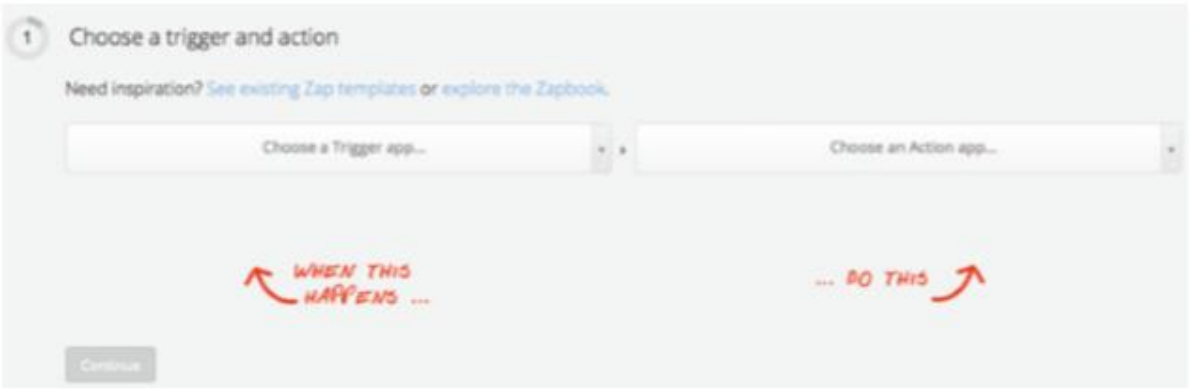
Untuk menyiapkan Zap, Anda harus terlebih dahulu menyiapkan akun Zapier gratis. Setelah Anda menyelesaikan proses penyiapan akun, masuk ke Zapier di https://zapier.com/. Setelah login, Anda akan diarahkan ke dashboard akun Anda. Seperti yang ditunjukkan pada

Gambar 12-16, di bawah tab My Zaps, klik tombol Make a New Zap untuk memulai proses pembuatan Zap



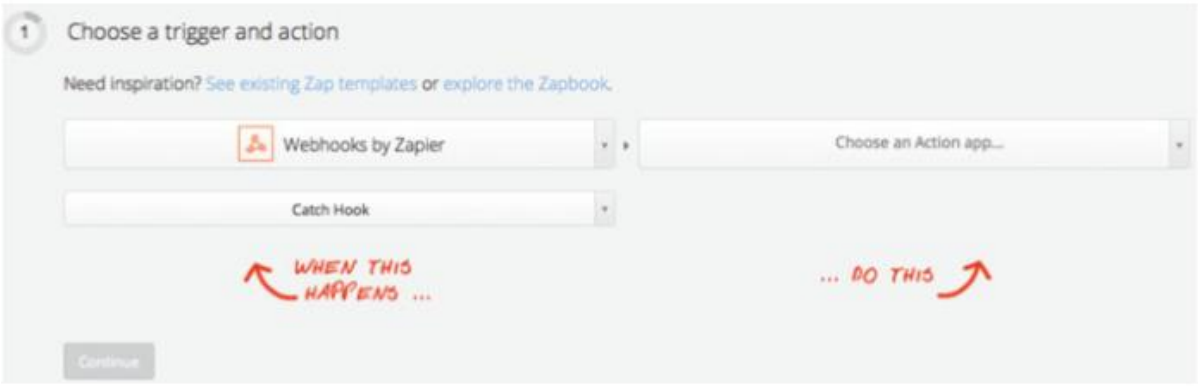
Gambar 12-16. My Zaps (Listing semua zap)

Seperti yang ditunjukkan pada Gambar 12-17, Langkah 1 mengharuskan Anda memilih aplikasi Trigger dan aplikasi Tindakan.



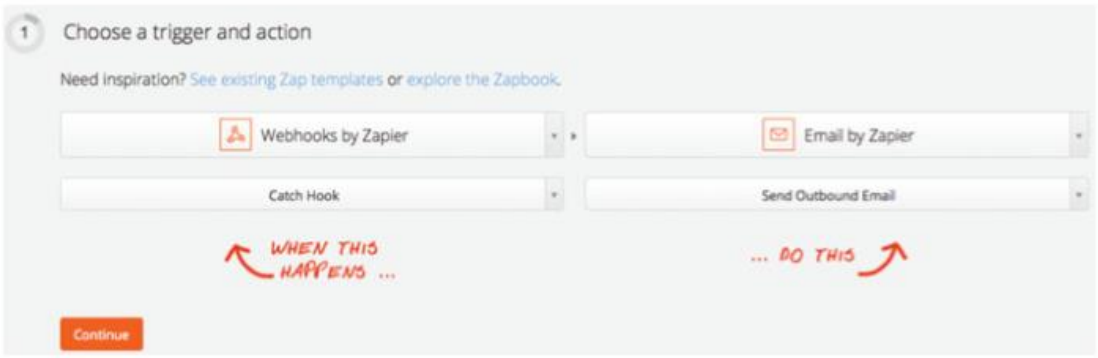
Gambar 12-17. Pilih trigger dan tindakan Zap

Pilih Webhooks by Zapier dari dropdown aplikasi Trigger dan pilih Catch Hook dari dropdown di bawahnya. Ini memberi tahu Zapier bahwa setiap kali URL tertentu dipanggil, Zap ini akan dipicu. Anda akan melihat URL yang dihasilkan nanti. Gambar 12-18 menunjukkan pemilihan Trigger.



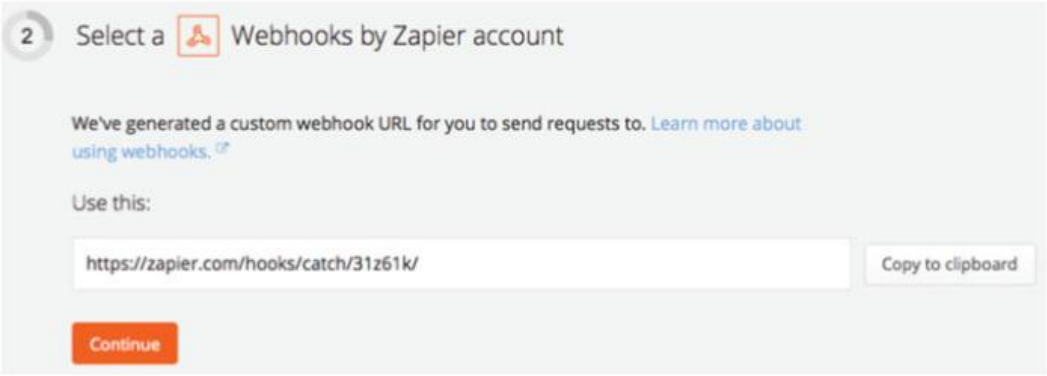
Gambar 12-18. Trigger zap

Anda harus mengirim email saat tugas ini dipanggil, jadi dari dropdown aplikasi Action, pilih Email by Zapier dan pilih Send Outbound Email dari dropdown di bawahnya, seperti yang ditunjukkan pada Gambar 12-19.



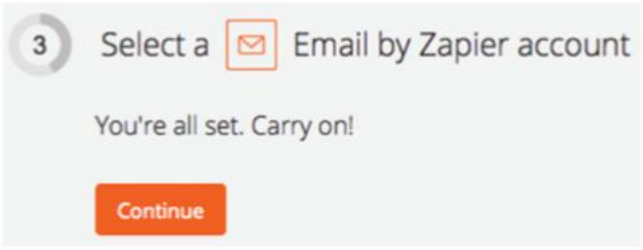
Gambar 12-19. Aksi zap

Pada Langkah 2, Zapier akan membuat URL webhook kustom yang akan dipanggil oleh Xively. Gambar 12-20 menunjukkan URL webhook khusus yang dibuat oleh Zapier. Klik Continue untuk melanjutkan ke langkah berikutnya.



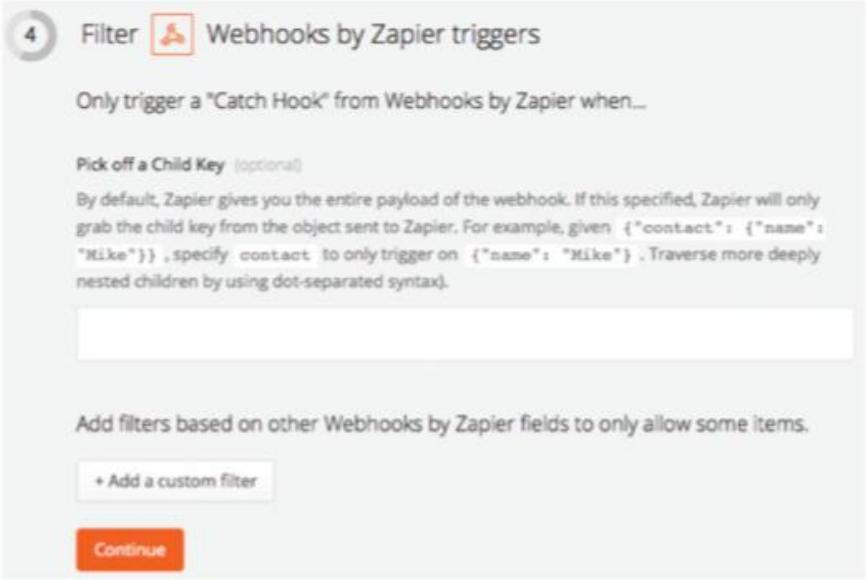
Gambar 12-20. URL webhook khusus

Karena Anda memilih Email by Zapier sebagai tindakan Anda, Langkah 3 tidak memerlukan masukan apa pun dari Anda, seperti yang ditunjukkan pada Gambar 12-21. Anda sudah siap, jadi klik Lanjutkan. Jika Anda memilih beberapa mekanisme email lain, seperti Gmail, maka Zapier akan meminta Anda untuk menyiapkan akun Gmail.



Gambar 12-21. Pengaturan akun email

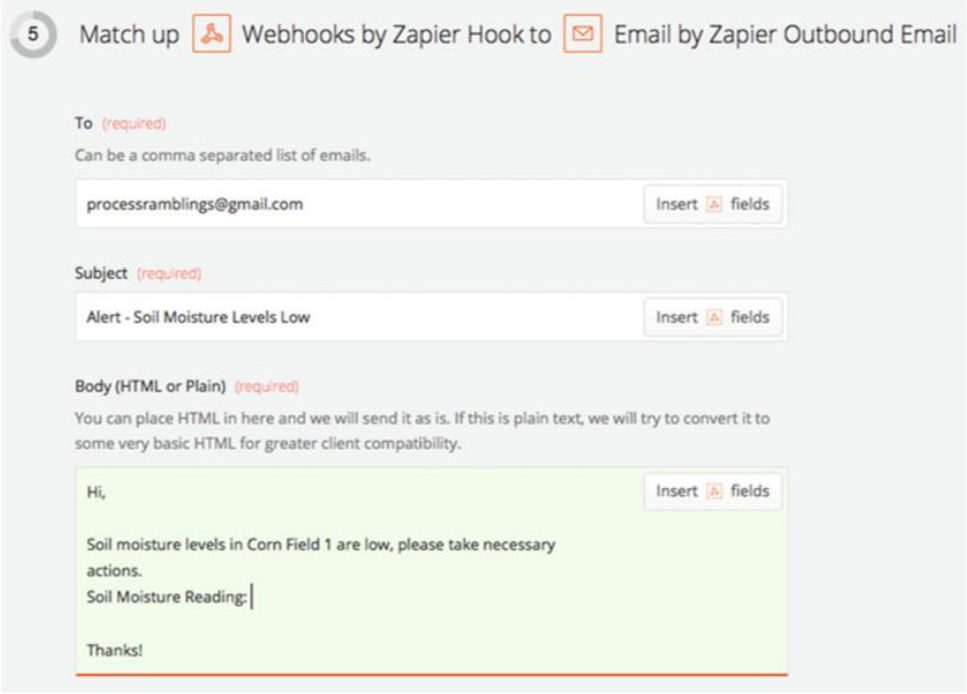
Seperti yang ditunjukkan pada Gambar 12-22, Langkah 4 memungkinkan Anda memfilter permintaan yang masuk melalui webhook. Anda dapat melewati langkah ini, karena Anda ingin semua permintaan Xively datang. Klik Lanjutkan.



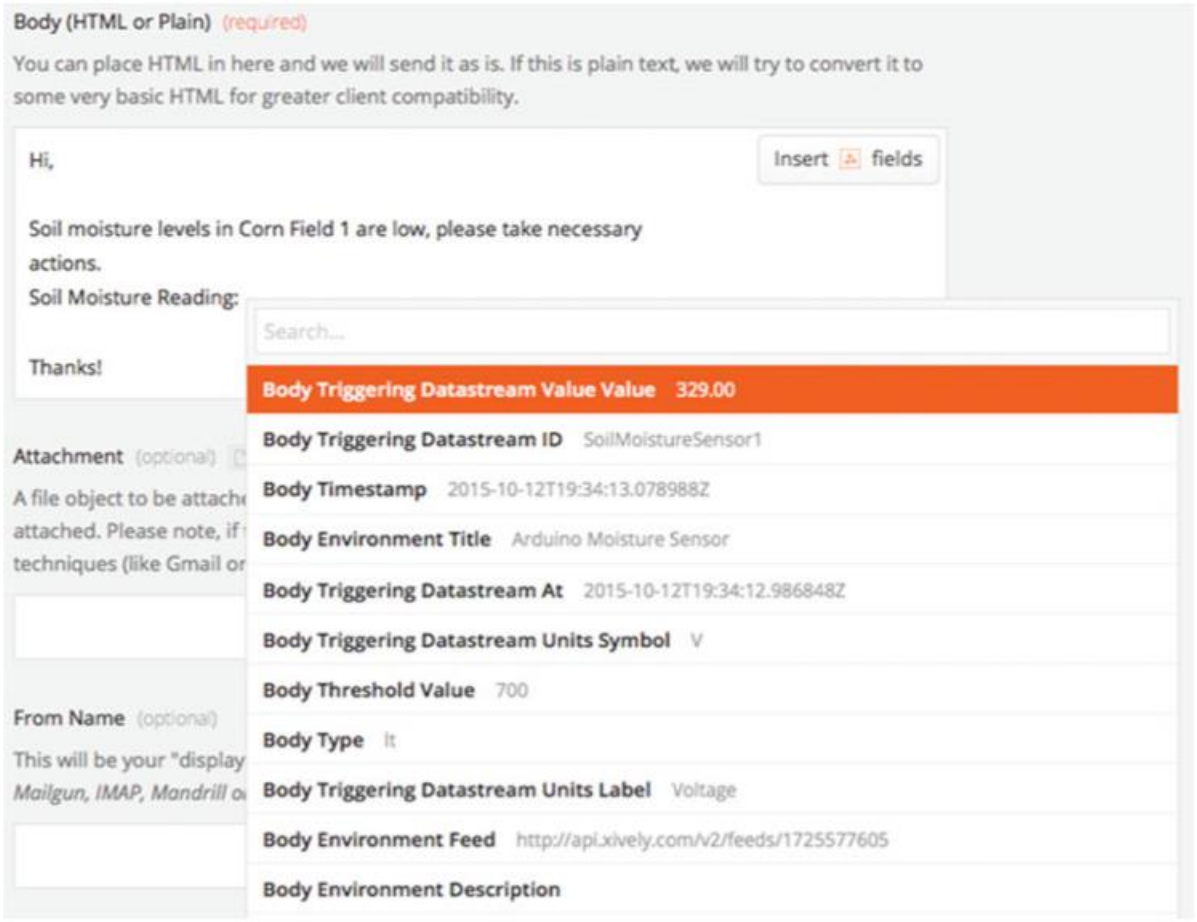
Gambar 12-22. Filter webhook

Pada Langkah 5, Anda perlu memberikan detail tentang peringatan email, seperti siapa yang harus dituju, apa yang harus menjadi subjek, dan apa yang harus menjadi teks isi. Ketika Xively memanggil URL webhook, itu akan mengirim beberapa data dalam permintaan HTTP POST juga. Anda dapat menggunakan data itu di Zapier di mana pun Anda melihat opsi Sisipkan Bidang. Gambar 12-23 menunjukkan setelan email.

Sebagai ilustrasi, proyek ini menggunakan nilai sensor kelembaban dan memasukkannya ke dalam badan email. Seperti yang ditunjukkan pada Gambar 12-24, saat Anda mengklik tombol Sisipkan Bidang, Listing semua variabel yang dapat dimasukkan akan ditampilkan. Awalnya, Anda mungkin tidak melihat data apa pun, jadi Anda dapat kembali ke langkah ini setelah **Xively Triggerly** disiapkan dan Anda telah mengirim beberapa permintaan pengujian. Zapier akan secara otomatis mulai menampilkan Listing semua variabel permintaan.

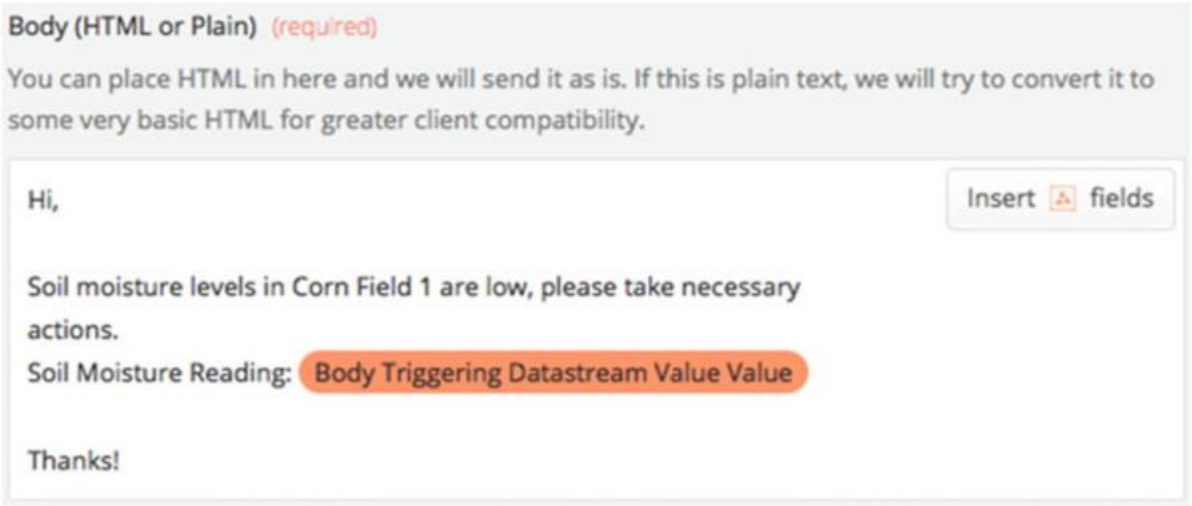


Gambar 12-23. Pengaturan email



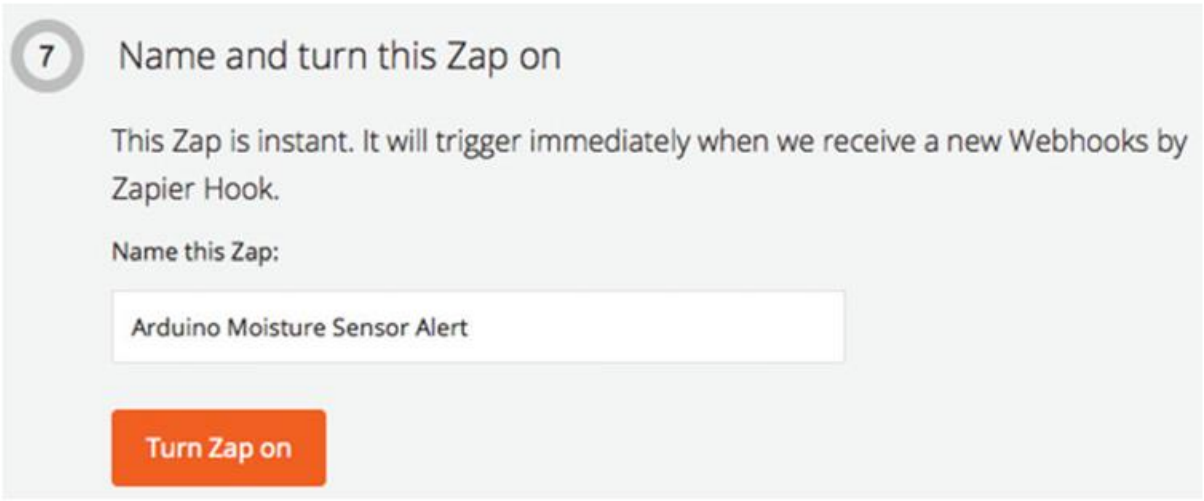
Gambar 12-24. Variabel permintaan

Gambar 12-25 menunjukkan pesan email terakhir setelah variabel permintaan dimasukkan.



Gambar 12-25. Badan email dengan variabel permintaan

Anda dapat melewati Langkah 6, dan pada Langkah 7 masukkan nama untuk Zap. Klik Turn Zap On seperti yang ditunjukkan pada Gambar 12-26.

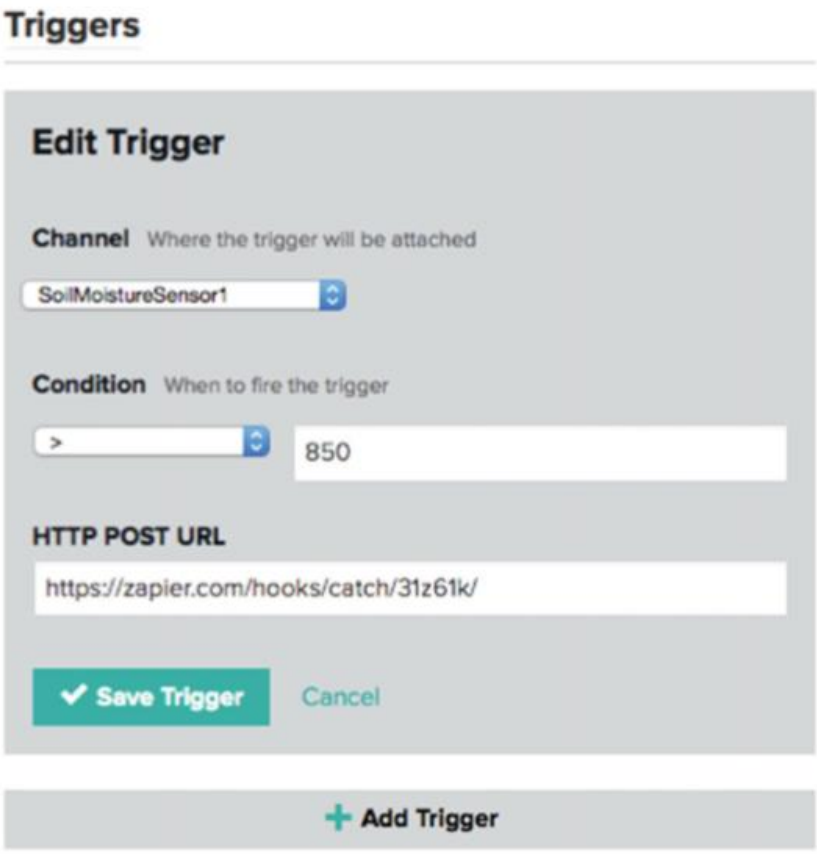


Gambar 12-26. Zap nama dan aktifkan Zap

Ini menyelesaikan pengaturan di Zapier. Sekarang Anda hanya perlu menyiapkan trigger di Xively yang akan memanggil URL webhook khusus yang dibuat oleh Zapier.

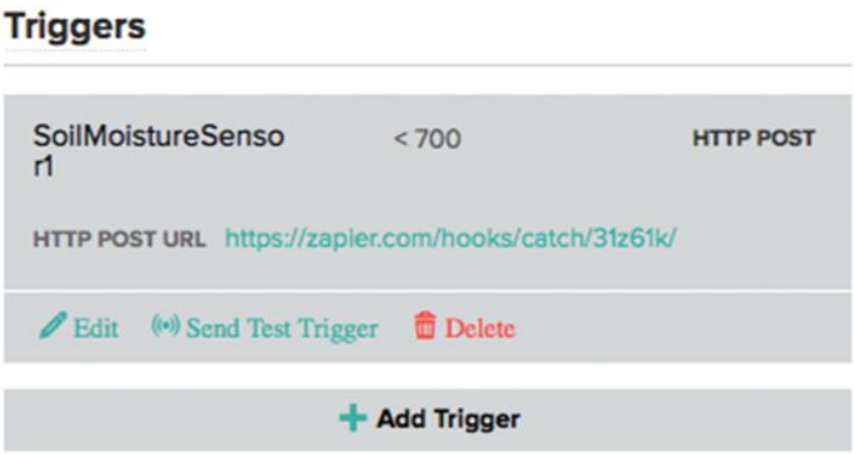
12.4 XIVELY TRIGGER

Masuk ke Xively dan buka bagian Trigger pada penyiapan perangkat. Klik tombol **Add Trigger**. Seperti yang ditunjukkan pada Gambar 12-27, pilih kondisi saat trigger harus diaktifkan; dalam hal ini IF SoilMoistureSensor1 > 850 THEN CALL HTTP POST URL. Di bidang HTTP POST URL, rekatkan URL webhook khusus yang dibuat oleh Zapier. Klik **save** Trigger untuk mengaktifkan Trigger.



Gambar 12-27. Pengaturan

Seperti yang ditunjukkan pada Gambar 12-28, Anda dapat dengan cepat menguji trigger Anda dengan mengklik **Send Test Trigger**. Ini memanggil URL webhook khusus yang Anda berikan di bidang **HTTP URL POST**.



Gambar 12-28. Trigger tes

12.5 KODE (ARDUINO)

Selanjutnya Anda akan menulis kode untuk menghubungkan Arduino ke Internet menggunakan WiFi, membaca data sensor kelembaban tanah, dan memublikasikannya ke saluran Xively. Mulai Arduino IDE Anda dan ketik kode yang disediakan di sini atau unduh dari situs dan buka. Semua kode dimasukkan ke dalam satu file sumber (*.ino), tetapi untuk membuatnya mudah dipahami dan digunakan kembali, kode tersebut telah dibagi menjadi lima bagian.

- Library eksternal
- Konektivitas Internet (Wi-Fi)
- Baca data sensor
- Xively (terbitkan)
- Fungsi standar

Library Eksternal

Bagian pertama kode, seperti yang disediakan di Listing 12-1, mencakup semua library eksternal yang diperlukan untuk menjalankan kode. Sketsa ini memiliki banyak dependensi—untuk konektivitas Internet Anda perlu menyertakan <WiFi.h> (dengan asumsi Anda menggunakan pelindung WiFi) dan untuk konektivitas Xively, Anda perlu menyertakan <HttpClient.h> dan <Xively.h>. Anda dapat mengunduh <Xively.h> dari https://github.com/xively/xively_arduino.

Listing 12-1. Kode untuk Menyertakan Dependensi Eksternal

```
#include <SPI.h>
#include <WiFi.h>
#include <HttpClient.h>;
#include <Xively.h>;
```

Konektivitas Internet (Nirkabel)

Bagian kedua dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk menghubungkan ke Internet. Gunakan kode dari Listing 2-7, 2-8, dan 2-9 (Bab 2) di sini.

Baca Data Sensor

Bagian ketiga dari kode ini disediakan dalam Listing 12-2. Ini mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk membaca data sensor. Fungsi readSensorData() membaca data dari Analog Pin A0 dan hasilnya antara 0 dan 1023. Nilai yang lebih tinggi sesuai dengan tingkat kelembaban tanah yang lebih rendah.

Listing 12-2. Kode untuk Membaca Nilai Sensor Kelembaban Tanah

```
int MOISTURE_SENSOR_PIN = A0;
float moistureSensorValue = 0.0;
void readSensorData()
{
    //Read Moisture Sensor Value
    moistureSensorValue = analogRead(MOISTURE_SENSOR_PIN);
    //Display Readings
    Serial.print("[INFO] Moisture Sensor Reading: ");
    Serial.println(moistureSensorValue);
}
```

Data Publish

Bagian keempat dari kode mendefinisikan variabel, konstanta, dan fungsi yang akan digunakan untuk memublikasikan data sensor ke saluran Xively. Untuk berkomunikasi dengan Xively, Anda perlu memberikan ID Umpan dan kunci API yang dihasilkan setelah Anda menyelesaikan penyiapan perangkat di Xively. Kedua kunci ini unik untuk Anda. Anda juga harus memberikan nama saluran yang tepat yang Anda masukkan di Xively. Jika kunci API atau ID Umpan salah, perangkat Anda tidak akan dapat terhubung dengan akun Xively Anda, dan jika nama saluran salah, data tidak akan muncul di grafik yang benar di dasbor Xively. Semua nilai ini telah disorot dalam kode (lihat Listing 12-3).

Jika Anda memiliki beberapa sensor dan ingin mengirim data ke Xively untuk semuanya, Anda cukup menyiapkan beberapa saluran di Xively. Dalam kode Arduino, Anda perlu menentukan nama saluran dengan cara yang sama seperti yang Anda definisikan moistureSensorChannel. Semua nama saluran ini harus diteruskan ke larik aliran data.

Umpan variabel XivelyFeed meneruskan data untuk semua saluran dengan nomor yang menentukan berapa banyak aliran data yang terkandung dalam umpan. Dalam hal ini, hanya ada satu aliran data, jadi nilainya akan menjadi 1. Selanjutnya Anda mendefinisikan variabel XivelyClient menggunakan WiFiClient. Ini akan digunakan untuk benar-benar membuat koneksi dan meneruskan umpan. Semua ini adalah pengaturan satu kali dan kode berulang ada di dalam fungsi transmitData(). Fungsi transmitData() menyetel kelembapanSensorValue terbaru dalam aliran data[0] dan kemudian mengirimkan umpan ke Xively. Jika kode status yang dikembalikan dari Xively di variabel ret adalah 200, itu berarti feed Anda berhasil dikirim ke Xively.

Listing 12-3. Kode untuk Memublikasikan Data ke Xively

```
// API Key - required for data upload
char xivelyKey[] = "YOUR_API_KEY";
#define xivelyFeed FEED_ID // Feed ID
char moistureSensorChannel[] = "SoilMoistureSensor1";
//Channel Name
// Datastream/Channel IDs  XivelyDatastream datastreams[] =
{
XivelyDatastream(moistureSensorChannel, strlen(moistureSensorChannel),
                                DATASTREAM_FLOAT),
```

```
};
// Create Feed  XivelyFeed feed(xivelyFeed, datastreams, 1);
                                                    // Number of Channels
                                                    // in Datastream
XivelyClient xivelyclient(client);
void transmitData()
{
   //Set Xively Datastream
   datastreams[0].setFloat(moistureSensorValue);
   //Transmit Data to Xively
   Serial.println("[INFO] Transmitting Data to Xively");
   int ret = xivelyclient.put(feed, xivelyKey);
   Serial.print("[INFO] Xively Response (xivelyclient.put): ");
   Serial.println(ret);
   Serial.println("----------------------------------------");
   }
```

Fungsi Standar

Bagian kode terakhir disediakan di Listing 12-4. Ini mengimplementasikan fungsi setup() dan loop() standar Arduino. Fungsi setup() menginisialisasi port serial dan menghubungkan ke Internet. Fungsi loop() pertama-tama membaca sensor kelembaban tanah dengan memanggil readSensorData() dan kemudian mentransmisikan nilai-nilai ini ke Xively dalam feed dengan memanggil transmitData(). Untuk setiap iterasi, Anda dapat menambahkan penundaan tergantung pada kebutuhan Anda.

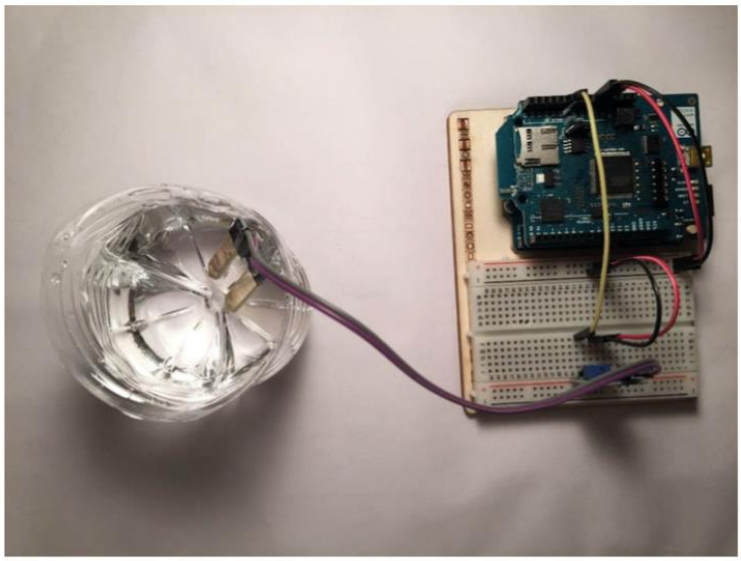
Listing 12-4. Kode untuk Fungsi Arduino Standar

```
void setup()
{
// Initialize serial port
Serial.begin(9600);
     // Connect Arduino to internet
     connectToInternet();
     }
     void loop()
     {
     readSensorData();
     transmitData();
     //Delay    delay(6000);
}
```

Kode Arduino Anda sekarang selesai.

12.6 PRODUK AKHIR

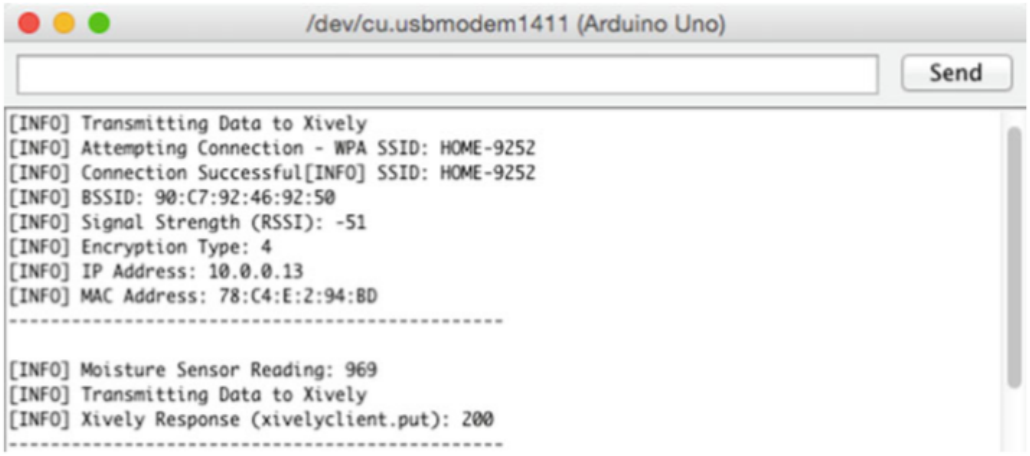
Untuk menguji aplikasi, verifikasi dan Upload kode Arduino seperti yang dibahas dalam Bab 1. Masukkan sensor kelembaban tanah Anda ke tanah kering atau cukup celupkan ke dalam air seperti yang ditunjukkan pada Gambar 12-29.



Gambar 12-29. Sirkuit terakhir dengan sensor terendam air

Catatan *: Jangan merendam sirkuit atau sensor sepenuhnya di dalam air atau tanah. Pastikan kabel tidak basah. Untuk instruksi yang tepat tentang sensor kelembaban tanah Anda, baca spesifikasi dan petunjuk produk pabrikan.*

Setelah kode diUpload, buka jendela Serial Monitor. Anda akan mulai melihat pesan log yang serupa dengan yang ditunjukkan pada Gambar 12-30.

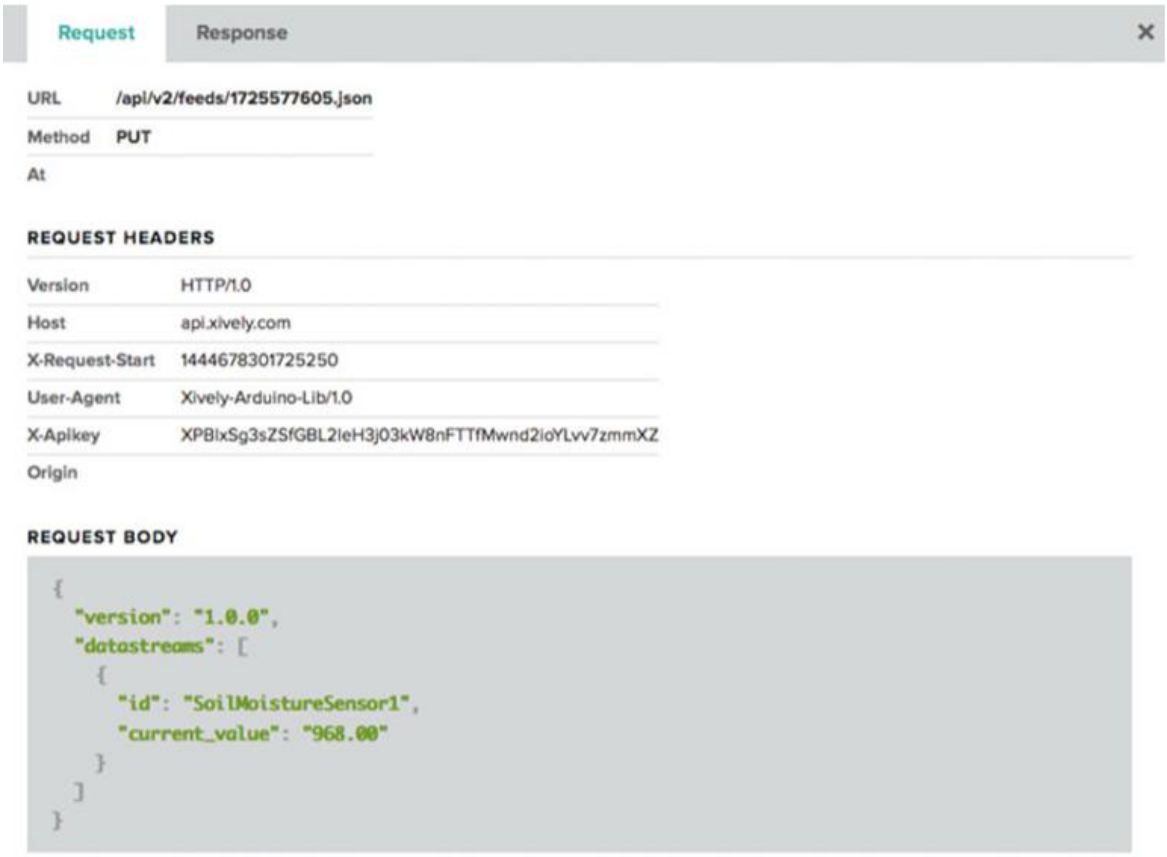


Gambar 12-30. Pesan log dari kontrol kelembaban tanah

Segera setelah Anda melihat respons Xively 200 di log serial Anda, masuk ke dasbor Xively dan lihat bagian Request Log, seperti yang ditunjukkan pada Gambar 12-31. Riwayat umpan data sensor Anda akan mulai muncul di bagian ini.

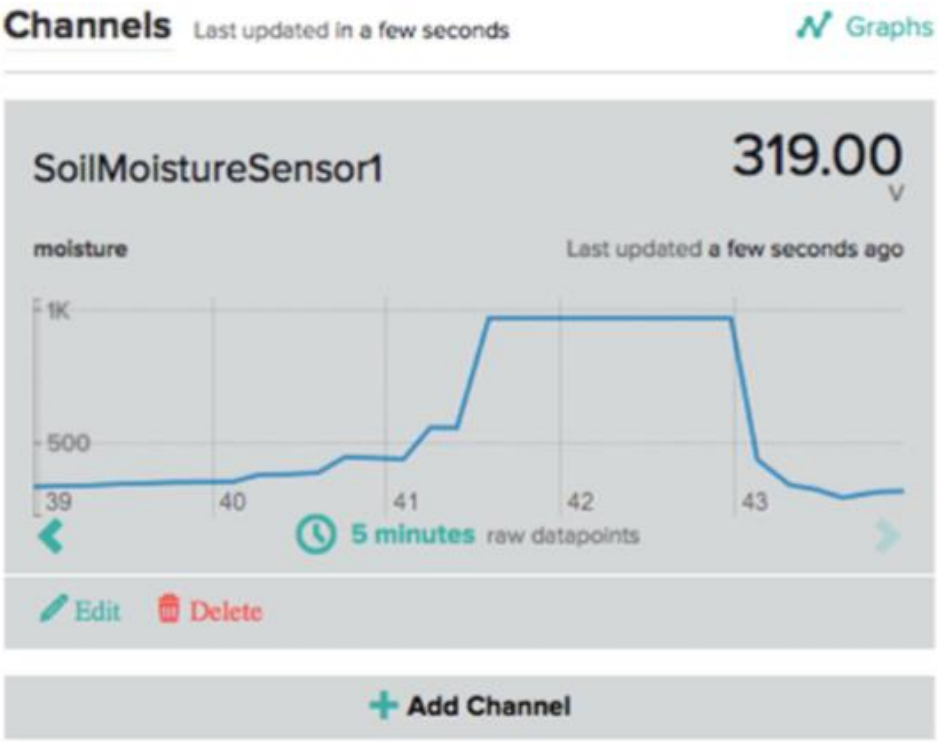
Gambar 12-31. Permintaan log dari sensor kelembaban tanah

Klik salah satu permintaan dan Anda akan dapat melihat permintaan persis yang dikirim dari sensor ke Xively (lihat Gambar 12-32).



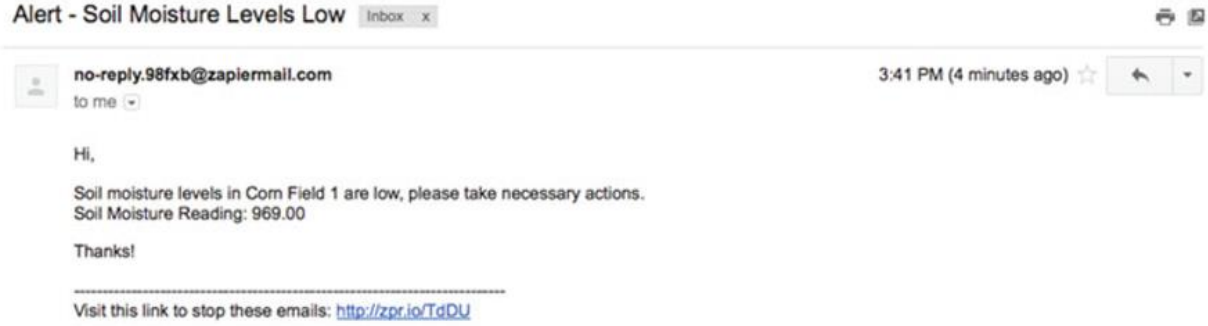
Gambar 12-32. Detail permintaan

Selanjutnya lihat grafik di bagian Channels, seperti yang ditunjukkan pada Gambar 12-33. Data sensor Anda akan mulai mengisi grafik selama jangka waktu tertentu.



Gambar 12-33. Tampilan data sensor S

Terakhir, pastikan Xively Triggerly Anda mengirimkan peringatan email:
Jika Anda menguji sensor kelembaban menggunakan air, maka keluarkan sensornya. Pembacaan harus segera naik, menunjukkan bahwa tingkat kelembaban telah turun. Xively Triggerly Anda akan menyala dan Zapier akan mengirimkan peringatan email.
Demikian pula, jika Anda menguji sensor kelembaban menggunakan tanah yang sebenarnya, keluarkan sensor Anda dari tanah basah. Ini akan menghasilkan peringatan email juga. Gambar 12-34 menunjukkan peringatan email yang dibuat oleh Xively/Zpier.



Gambar 12-34. Beritahu e-m ail

12.7 RINGKASAN

Dalam bab ini, Anda telah mempelajari tentang platform IoT dan kelebihannya. Anda mengembangkan aplikasi IoT yang memublikasikan data sensor ke Xively, yang merupakan salah satu platform IoT lebih populer yang tersedia di pasar. Ada lebih dari 100 platform IoT skala kecil, menengah, dan besar yang saat ini tersedia. Tabel 12-1 mencantumkan beberapa platform IoT utama dengan link untuk mengaksesnya. Semua platform ini menyediakan versi uji coba gratis atau pengurangan untuk penggunaan pribadi.

Tabel 12-1. Platform IoT Utama

Platform	**Contoh**
IBM Internet of Things Foundation/ IBM Bluemix	http://www.ibm.com/internet-of-things/
Intel IoT	https://software.intel.com/en-us/iot/home
Microsoft Azure IoT	https://www.azureiotsuite.com/
Amazon AWS IoT	https://aws.amazon.com/iot/
Thingworx	http://www.thingworx.com/
ively	https://xively.com/

Ada banyak bahan yang tersedia yang dapat membantu Anda menentukan mana yang terbaik untuk kebutuhan Anda. Platform IoT mempercepat masuknya begitu banyak orang ke dunia IoT. Seiring berkembangnya IoT, platform ini akan menjadi lebih canggih dan semakin menyederhanakan pengembangan aplikasi IoT.

DAFTAR PUSTAKA

Aditya, G. 2015 *Analisis Dan Perancangan Prototype Smart Home Dengan Sistem Client Server Berbasis Platform Android Melalui Komunikasi Wireless*. Teknik Telekomunikasi, Universitas Telkom.

Alamsyah, Ardi, A., & Faisal, M. N. (2015). *Perancangan dan Penerapan Sistem Kontrol Peralatan Elektronik Jarak Jauh Berbasis Web*. Jurnal Mekanikal, 6(2), 577-584.

Alper Gurek, Caner Gur, CagriGurakin, Mustafa Akdeniz, Senem Kumova Metin, Ilker Korkmaz, "*Sistem Otomasi Rumah Berbasis Android*", 2013 IEEE

Amsar, Khairuman. (2020). Perancangan Alat Pendeteksi Co2 Menggunakan Sensor Mq-2 Berbasis Internet of Thing. *Jurnal Manajemen Informatika & Komputerisasi Akuntansi,* 4 (1), 73-79.

Andriyanto, H., & Darmawan, A. (2015). *Arduino Belajar Cepat dan Pemrograman*. Bandung: Informatika.

Anushri Sadar, Sonali Vaidya, PriyankaAshture, Varsha Gaiwal, "Otomasi Rumah menggunakan Aplikasi Android dan Jaringan Cloud", *Jurnal Internasional Penelitian Teknik dan Ilmu Pengetahuan Umum Vol. 3, Edisi 3, Mei-Juni, 2015.*

Carelin Felix, I. Jacob Raglend, "*Otomatisasi Rumah Menggunakan GSM*", Prosiding Konferensi Internasional 2011 tentang Teknologi Pemrosesan Sinyal, Komunikasi, Komputasi dan Jaringan (ICSCCN 2011).

Chiu-Chiao, H. Ching Yuan, W. Shiau-Chin, L. Cheng-Min, "*Aplikasi Android Interaktif Berbasis Bluetooth untuk Smart Living,*" Dalam Inovasi dalam Komputasi dan Aplikasi Bioinspired (IBICA), 2011 *Konferensi Internasional Kedua,* 2011, hlm. 309-312

Dedy Hamdani, Elda Handayani. (2019). Rancang Bangun Alat Pendeteksi Asap Rokok Dan Nyala Api Untuk Penanggulangan Kesehatan dan Kebakaran Berbasis Arduino Uno dan Gsm Sim900a. *Jurnal Ilmu Fisika, 11 (1), 37-46.*

Despa, Dikpride, et al. , "*Smart Monitoring of Electrical Quantities Based on Single Board Computer BCM2835*" ,Int. Conference on Information Technology, Computer and Electrical Engineering ICITACEE, Oct – 2015.

Emilia Hesti, Adewasti. (2018). Aplikasi Android Sebagai Pengontrol Jarak Jauh Smarthome Dengan Koneksi Jaringan Internet. *Jurnal Surya Energy, 2(2), 157-165.*

Fathansyah. (2015). *Sistem Basis Data.* Bandung: Penerbit Informatika.

Fukuoka-shi, "Desain dan Implementasi Sistem Peningkatan Otentikasi untuk Pengguna Smartphone Android", *Konferensi Internasional IEEE tentang Lokakarya Jaringan Informasi dan Aplikasi Lanjutan, 2012.*

Giyartono, A., & Kresnha, P. E. (2015). *Aplikasi Android Pengendali Lampu Rumah Berbasis Mikrokontroler ATmega328*. Seminar Nasional Sains dan Teknologi 2015, pp. 1-9.

Ismail, R. 2017. *Pemanfaatan Sistem Android Sebagai Pengendali Lampu Ruangan Berbasis Mikrokontroler Arduino.* Teknik Elektro Undana. Kupang.

Kadir, A 2013. *Pengenalan Sistem Informasi edisi Revisi*. Yogyakarta : Andi.

Kadir, A. 2012. *Panduan Praktis Mempelajari Aplikasi Mikrokontroler Dan Pemrogramannya Menggunakan Arduino.* Penerbit Andi. Yogyakarta.

Karumbaya, A., & Satheesh, G. (2015). IoT Empowered Real Time Environment Monitoring System*. International Journal of Computer Applications, 129(5), 30-32.*

Kristomson, Rosalia. (2018). Sistem Keamanan Ruangan Berbasis Internet Of Things Dengan Menggunakan Aplikasi Android*. Jurnal Tesla Teknik Elektro Universitas Tarumanegara, 20(2), 127-134.*

Luitel, S.2013. Design and Implementation of a Smart Home System. Tesis. *Degree Programme Information Technology, Helsinki Metropolia University of Apllied Science*. Helsinki, Finlandia.

M. A. Al-Qutayri, J. S. Jeedella, "*Teknologi Nirkabel Terpadu untuk Aplikasi Rumah Pintar*," Dalam Sistem Rumah Pintar, M. A. Al-Qutayri, Ed., Ed: InTech, 2010.

Malvino, A. 1996. *Elektronika Komputer Digital Pengatar Mikrokomputer(Edisi Ke Dua).* Penerbit Erlangga. Jakarta.

Muhamad Muslihudin, Willy Renvillia. (2018). Implementasi Aplikasi Rumah Pintar Berbasis Android Dengan Arduino Microcontroller. *Jurnal Keteknikan dan Sains, 1(1), 23-31.*

Muhamad, S. 2013. *Panduan Mudah Simulasi dan Praktik Mikrokontroler.* Penerbit Andi. Yogyakarta.

Muhammad, P. 2014. *Miniatur Rumah Pintar Menggunakan Pengendali Via Android Berbasis Arduino Mega 2560*; Depok: Skripsi Fakultas Ilmu Komputer & Teknologi Informasi Universitas Gunadarma.

Nugraha, Nunu. 2017. "*Rancang Bangun Sistem Monitor Dan Kendali Ruang Laboratorium Berbasis Arduino ESP-2E NodeMCU*". Kuningan: Vol 2, No 1 , 2016. Universitas Kuningan.

Pasha, S.(2016).Thingspeak Basic Sensing and Monitoring System for IoT with Matlab Analisis.International Journal of New Technology and Research(IJNTR) .2(6).19-23.

Prachi T. Deokar, Dr. Manoj S. Nagmode, “Sistem Otomasi Rumah Berbasis Server Cloud Menggunakan Ponsel Android”, (IJIRSE) *Jurnal Internasional Penelitian Inovatif dalam Sains & Teknik*

Rajeev Piyare, "*Internet of Things*: Kontrol Rumah Berada dan Sistem Pemantauan menggunakan Ponsel Pintar berbasis Android", *Jurnal Internasional Internet of Things 2013, 2 (1): 5-11*

Rizaldy Haris, Mochtar Yahya. (2018). Prototipe Sistem Peringatan Dini Kebakaran Menggunakan Hybrid Sensor Api dan Mq-2 Berbasis IoT. *Jurnal Ilmiah Setrum, 7 (2), 228-236.*

Sri Zholehaw, Ali Basrah Pulungan. (2019). Sistem Monitoring Realtime Gas Co Pada Asap Rokok Berbasis Mikrokontroler. *Jurnal Teknik Elektro dan Vokasional, 5(1), 17-21.*

Sulistyanto, M. T., Nugraha, D. A., dkk. (2015). Implementasi IoT (*Internet of Things*) dalam pembelajaran di Universitas Kanjuruhan Malang. *SMARTICS Journal, 1(1), 20-23.*

Tje Kevin Ariefaldi Ahmad, Moh. Abdullah Anshori. (2020). Implementasi Iot Sebagai Monitoring Sistem Pembayaran Uang Kos Berbasis Android. *Jurnal Jaringan Telekomunikasi,10(1), 85-95.*

Y. Liu, "*Studi tentang Sistem Rumah Pintar Berbasis Internet of Things Technology*," Dalam Ilmu Informatika dan Manajemen IV. vol. 207, W. Du, Ed., Ed: Springer London, 2013, hlm. 73-81.

Yoyon Efendi. (2018). *Internet Of Things* (IoT) Sistem Pengendalian Lampu Menggunakan Raspberry Pi Berbasis Mobile. *Jurnal Ilmiah Ilmu Komputer, 4(1), 19-26.*

Yudhanto, Y. 2018. *Mudah Membuat dan Berbisnis Aplikasi Android dengan Android Studio.* Elex Media Komputindo. Yogyakarta.

Zanella, A., & Vangelista, L.. Internet of Things for Smart Cities. *IEEE INTERNET OF THINGS JOURNAL, 1(1), 22-32*.