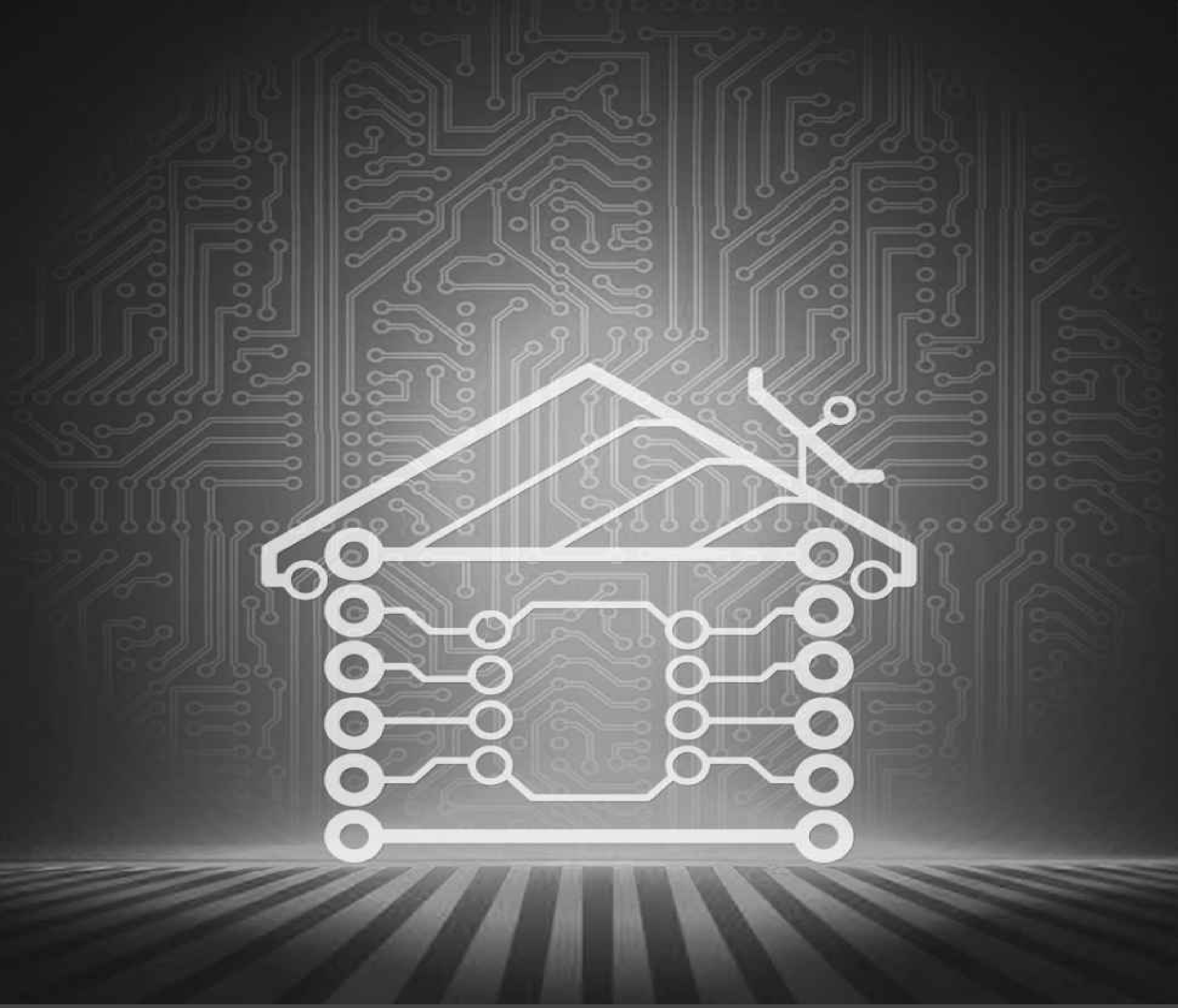
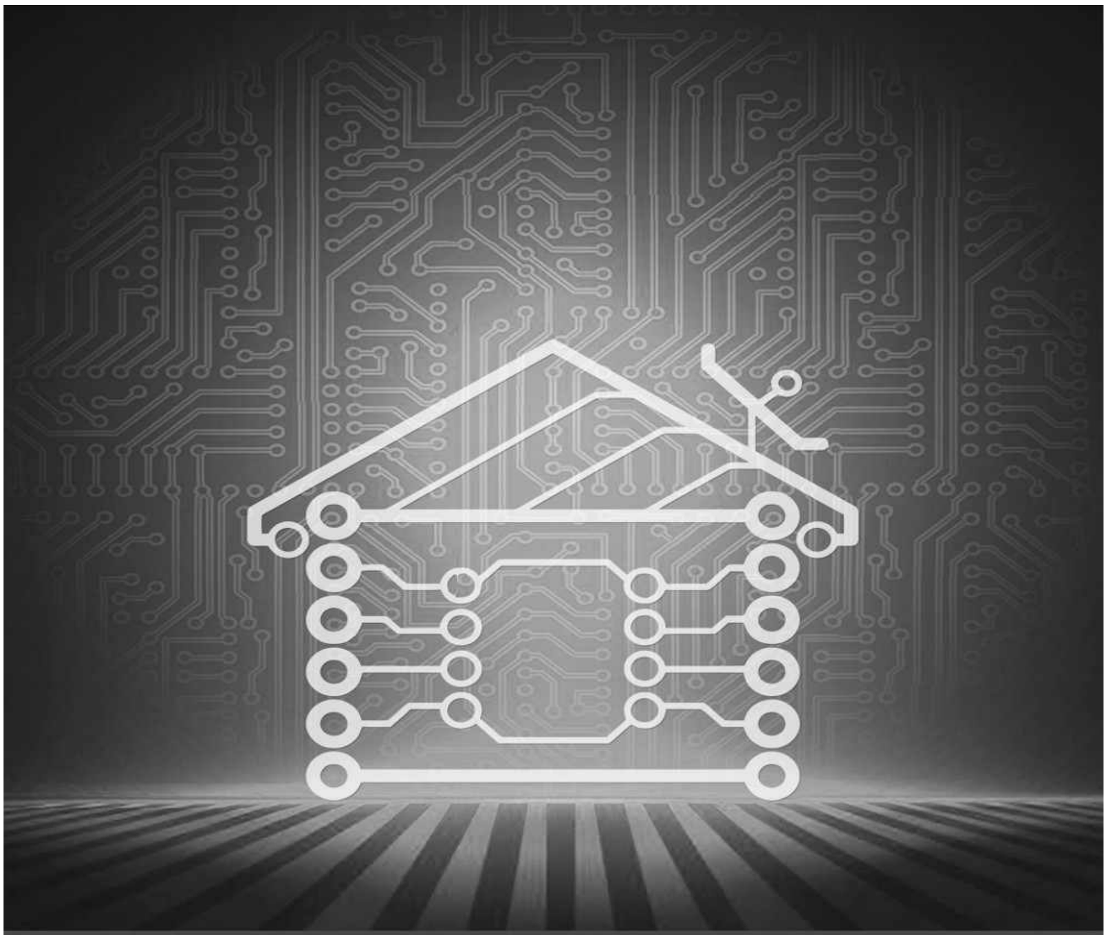


Internet of Things With Arduino

Build Internet of Things Projects With the Arduino Platform

Open Home Automation

Marco Schwartz



Internet of Things With Arduino

Build Internet of Things Projects With the Arduino Platform

Open Home Automation

Marco Schwartz

Internet of Things with Arduino

Build Internet of Things Projects With the Arduino Platform

Marco Schwartz, PhD

Contents

Chapter 4 Wireless Security Camera

Chapter 6 Resources

Legal

Introduction

1 Why the Internet of Things?

The Internet of Things (IoT) is one of the major technological trend at the moment. The idea behind the Internet of Things is to have every object from our everyday lives connected to the Internet. This will allow objects to continuously send data to the web and to be accessible from anywhere. It will also allow objects to interact directly with each other. This is called Machine To Machine communication, or M2M.

The Internet of Things is for example a major trend in the home automation space, and this is why the projects in this book are mostly in the home automation field.

Indeed, more and more home automation systems are connected with the web. Even some objects like power switches are directly connected to the Internet. This completely removes the need of a central home automation hub that coordinates everything.

In this book, I will show you how to build your own Internet of Things projects with the Arduino platform. We are going to build several web-connected projects that will illustrates all the major aspects of the Internet of Things. For example, we will build a project that continuously logs data to a web service.

2 How is the book organized?

This book is organized around four main chapters, each chapter focusing on a given Internet of Things project using the Arduino platform.

In each chapter, I will give the complete list of hardware components that are required to build the project, along with links to get the components on the web. Note that these links are only here for reference purposes to help you find out these articles. I have no commercial contract with the websites that I mention in these links. Also, in every project I also give some advice to use similar components to the one I used, in case you already have some components on your desk.

All chapters include a detailed walk through of all the code used to build the different projects of this book. In these code explanations, I detail the most important parts of the code so you can understand how each project is working. You should also always refer to the GitHub repository of this book to get the most up-to-date code for all projects of this book. You will find the link to the GitHub repository in every chapter of this book, or if you want to have a look right now, the link is:

https://github.com/openhomeautomation/iot-book

Note that all chapters were designed to be completely independent from each other, so you can start with whatever chapter you like in the book without being lost. However, especially if you are a beginner on the topic it is highly recommended to follow the chapters of this book in order.

In Chapter 1, you will make your first Internet of Things project with the Arduino platform. We are going to learn the basics of the Internet of Things: building a device that connects to the web, send data to the cloud, and monitor this data online.

After that, in Chapter 2 we are going to build a lamp that can be controlled from everywhere in the world. We will use the Arduino Ethernet Shield to connect our project to the web, and a dedicated web service so the project is accessible from wherever you are on the globe.

In Chapter 3, we will continue our journey into the Internet of Things with Arduino, and use a more complex board: the Arduino Yun. Using this powerful board, we will log data directly inside a Google Docs spreadsheet and monitor this data from your web browser.

Finally, in Chapter 4, we will connect a camera to the Arduino Yun, to create a wireless security camera. This camera will constantly be connected to the web, and upload pictures to Dropbox if motion is detected in front of the camera.

3 What Will you Learn?

Before diving into the heart of this book, and build your first Internet of Things project, I wanted to spend some time to explain what you will actually learn in this book.

Of course, you will learn about the Internet of Things. You will learn the basics of the topic, for example how to use online services like Temboo to send your data to the cloud, and monitor this data from wherever you are in the world.

But you will learn more than that. You will learn about what I consider to be one of the most important platform at the moment when it comes to do-it-yourself electronics projects: the Arduino platform. What you will learn about this platform can then be used again in many other projects.

You will also learn about electronics in general, as we will interface many sensors and actuators to Arduino. Finally, you will also learn about software development, especially about C/C++ when we will build Arduino sketches.

4 Prerequisites

First of all, you could just read this book from start to finish without actually doing any of the projects you will read about. By doing so, you would still learn a lot about electronics, programming and of course the Internet of Things. However, I really recommend spending time doing the projects yourself, you will learn so much more by doing so.

To follow the projects that you will find this book, you will need to know some basics about software development, electronics and the Arduino platform. If you want to learn more about Arduino first, you will find several resources at the end of this book to help you with that.

Chapter 1

Internet of Things with Arduino

In this first chapter of the book, you are going to build a weather measurement station that will automatically send data to an online cloud service. To do so, we will simply use an Arduino Uno board, a WiFi module and some sensors.

This first chapter of the book is here to show you how easy it is to build an Internet of Things project. We are going to learn how to store measurements online. Then, we will see how to automatically display these measurements so they can be monitored from a central place on the web. Let's dive in!

1.1 Hardware & Software Requirement

For this project, you will first need an Arduino Uno board.

For temperature and humidity measurements, you will also need a DHT11 sensor, along with a 4.7K resistor. You can also use a DHT22 sensor which is more precise, you will only have one line of code to change.

For light levels measurements, I used a photocell with a 10K Ohm resistor. This will return a signal which is proportional to the ambient light level.

Then, you need the CC3000 chip for WiFi connectivity. There are many different board for this chip on the market. What I recommend is using the Adafruit CC3000 breakout board, which is the only one I tested that worked without problem. It is nice and compact, has voltage regulators onboard (so you can connect it directly to your Arduino board), as well as an onboard antenna.

Finally, you need a breadboard and some jumper wires to make the connections between the different parts.

This is a list of all components used in this project, along with links to purchase them online:

- Arduino Uno (http://www.adafruit.com/product/50)
- DHT11 sensor with 4.7K resistor (http://www.adafruit.com/product/386)
- Photocell (http://www.adafruit.com/product/161)
- 10k Ohm resistor (https://www.sparkfun.com/products/8374)
- CC3000 breakout board (https://www.adafruit.com/products/1469)
- Breadboard (http://www.adafruit.com/product/64)
- Jumper wires (http://www.adafruit.com/product/758)

One software side, you will need the library for the DHT sensor:

https://github.com/adafruit/DHT-sensor-library

You will also need the library for the CC3000 WiFi chip:

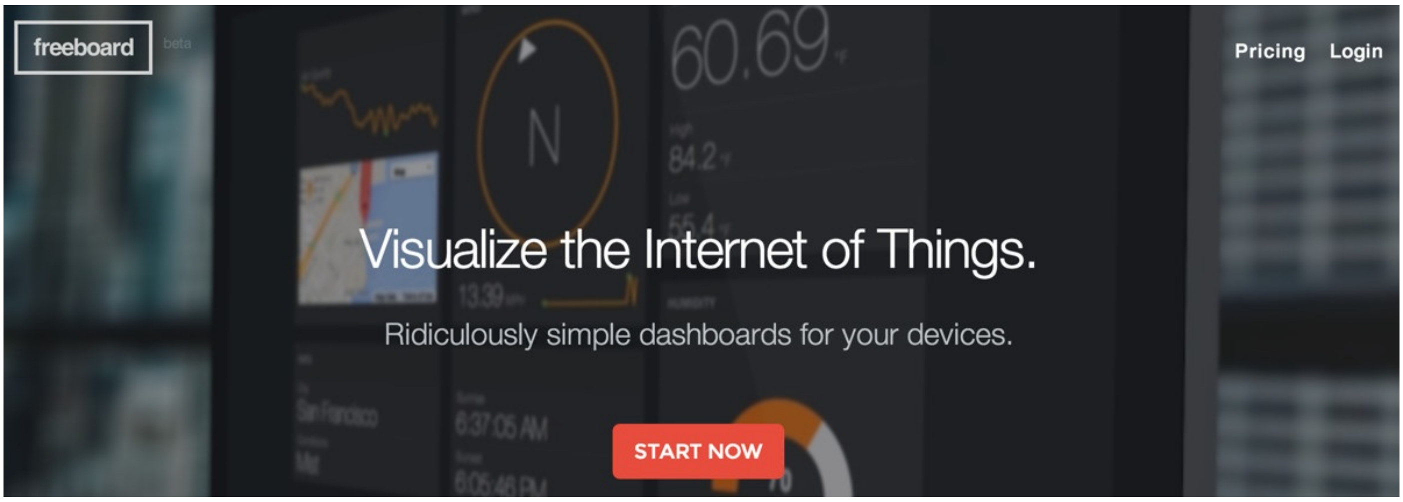
https://github.com/adafruit/Adafruit_CC3000_Library

To install a library, simply extract the library folder inside your /libraries folder of you main Arduino folder.

You will also need to create an account on Freeboard, as we will use this service to display the measured data. Go over to:

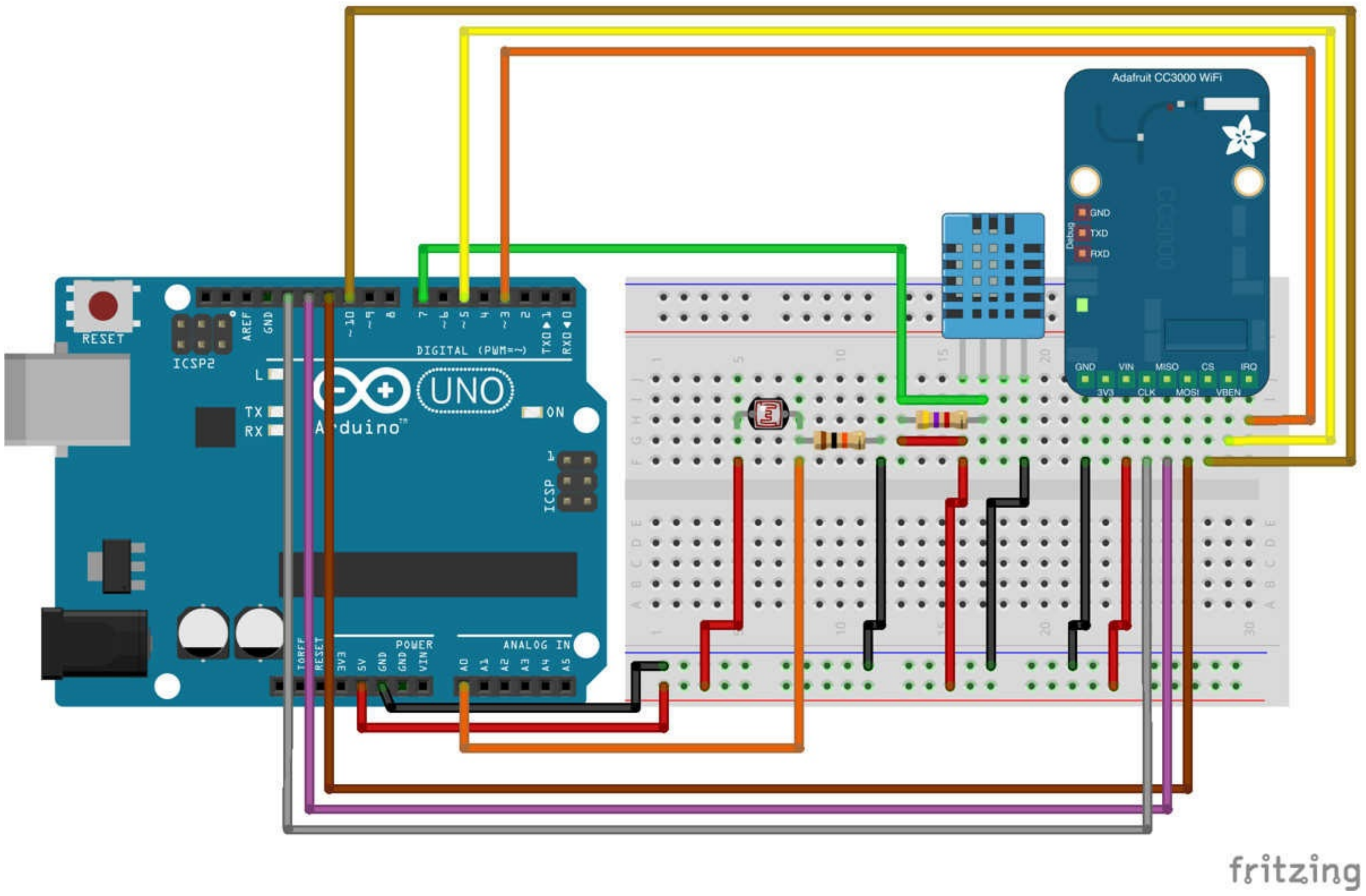
https://www.freeboard.io/

You will be taken to the welcome page of the Freeboard service, where you can create an account:


freeboard
beta
Pricing
Login
Visualize the Internet of Things.
Ridiculously simple dashboards for your devices.
START NOW

1.2 Connecting Sensors to Arduino

The hardware connections for this project are actually quite simple: we have to connect the DHT11 sensor, the part responsible for the light level measurement with the photocell, and the CC3000 WiFi chip. To help you out, the following picture summarizes the hardware connections:



This image was created with Fritzing (http://fritzing.org/).

First, connect the Arduino Uno +5V pin to the red rail on the breadboard, and the ground pin to the blue rail. Then, place the DHT sensor and the CC3000 breakout board on the breadboard.

After that, connect pin number 1 of the DHT11 sensor (see the schematic) to the red rail on the breadboard, and pin number 4 the blue rail. Also connect pin number 2 of the sensor to pin number 7 of the Arduino board. To complete the connections of the DHT11 sensor, connect the 4.7k Ohm between pin number 1 and 2 of the sensor.

For the photocell, first place the cell in series with the 10k Ohm resistor on the breadboard. Then, connect the other end of the photocell to the red rail on the breadboard, and the other end of the resistor to the ground. Finally, connect the common pin to the Arduino Uno analog pin A0.

Now, the WiFi module. First, connect the IRQ pin of the CC3000 board to pin number 3 of the Arduino board, VBAT to pin 5, and CS to pin 10. Then, you need to connect the SPI pins to the Arduino board: MOSI, MISO, and CLK go to pins 11,12, and 13, respectively. Finally, take care of the power supply: Vin goes to the Arduino 5V (red power rail), and GND to GND (blue power rail).

1.3 Testing the Sensors

Now that the hardware of the project is fully assembled, we are going to test the different sensors on the board. And to do so, we are going to write a simple Arduino sketch. We will simply read out data from the sensors, and print this data on the Serial port.

The DHT11 sensor is digital sensor, so we will use a dedicated library to get data from this sensor. For the photocell, which is a purely analog sensor, we will use the `analogRead()` function of Arduino to perform light level measurements.

This is the complete code for this section:

```
// Libraries
#include "DHT.h"

// DHT sensor
#define DHTPIN 7
#define DHTTYPE DHT11

// DHT instance
DHT dht(DHTPIN, DHTTYPE);

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

  // Init DHT
  dht.begin();
}


void loop()
{
  // Measure from DHT
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Measure light level
  float sensor_reading = analogRead(A0);
  float light = sensor_reading/1024*100;

  // Display temperature
  Serial.print("Temperature: ");
  Serial.print((int)temperature);
  Serial.println(" C");

   // Display humidity
  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.println("%");

  // Display light level
  Serial.print("Light: ");
  Serial.print(light);
  Serial.println("%");
  Serial.println("");

  // Wait 500 ms
  delay(500);

}
```

Let's now see the details. It starts by importing the library for the DHT sensor:

```
#include "DHT.h"
```

And create a DHT instance:

```
DHT dht(DHTPIN, DHTTYPE);
```

In the `setup()` function of the sketch, we have to initialize the sensor:

```
dht.begin();
```

And the Serial port:

```
Serial.begin(9600);
```

In the `loop()` function, we are going to continuously read data from the sensors, and print it on the Serial port. It starts by getting data from the temperature & humidity sensor:

```
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
```

For the photocell, we first read data from the analog pin A0, which returns a value from 0 to 1023, at the resolution of the Analog-To-Digital converter of the Arduino Uno board is 10 bits, so 1024 values. Then, we divide this reading by 1024 and multiply it by 100 to have the light level as a percentage:

```
float sensor_reading = analogRead(A0);
float light = sensor_reading/1024*100;
```

Then, we print these different measurements on the Serial port. First, the temperature:

```
Serial.print("Temperature: ");
Serial.print((int)temperature);
Serial.print((char)223);
Serial.println("C");
```

The humidity is exactly similar, just as the light level:

```
Serial.print("Light: ");
Serial.print(light);
Serial.println("%");
```

Finally, we introduce a delay of 500 ms between each new set of measurements:

```
delay(500);
```

Note that the complete code for this chapter can be found on the corresponding folder inside the GitHub repository of the book:

https://github.com/openhomeautomation/iot-book

It’s now time to test this first Arduino sketch. Upload the code to the Arduino board, open the Serial monitor inside the Arduino IDE (make sure the Serial speed we defined in the code), and this is what you should see:

```
Temperature: 25 C
Humidity: 36.00%
Light: 83.79%
```

If that works, congratulations, your sensors are working correctly! You can try for example to pass your hand in front of the photocell, and you should see that the light level is changing instantly.

In case it is not working at this point, there are several things you can check. First, make sure that you correctly downloaded and installed the required libraries for the chapter. Also make sure that you correctly connected the sensors to your Arduino board, as defined earlier in the chapter. Finally, make sure that you are using the latest version of the code from the GitHub repository of the book.

1.4 Uploading Data to the Cloud

We are now going to do the most important part of this chapter: upload data to the cloud via WiFi. To do so, we will use a service called dweet.io, which is a service that proposes to store your data online via a very simple API. You can check out their page at:

https://dweet.io/

The nice thing with dweet.io is that it doesn't require any account creation or configuration: you can upload data immediately. It is based on the principle of having objects called "things", to which you will upload new data via HTTP requests. And if you upload data to a new thing that doesn't exist yet, it will be created automatically.

Let's now build the sketch that we will use to automatically make measurements, connect to the dweet.io server, and upload the data there. This is the complete code for this section:

```
// Libraries
#include <Adafruit_CC3000.h>
#include <ccspi.h>
#include <SPI.h>
#include "DHT.h"
#include <avr/wdt.h>

// Define CC3000 chip pins
#define ADAFRUIT_CC3000_IRQ   3
#define ADAFRUIT_CC3000_VBAT  5
#define ADAFRUIT_CC3000_CS    10

// DHT sensor
#define DHTPIN 7
#define DHTTYPE DHT11

// Create CC3000 instances
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT, SPI_CLOCK_DIV2);

// DHT instance
DHT dht(DHTPIN, DHTTYPE);

// WLAN parameters
#define WLAN_SSID       "yourWiFiNetwork"
#define WLAN_PASS       "yourPassword"
#define WLAN_SECURITY   WLAN_SEC_WPA2

// Xively parameters
#define thing_name  "yourThingName"

// Variables to be sent
int temperature;
int humidity;
int light;

// IP variable
int32_t ip;

void setup(void)
{
  // Initialize
  Serial.begin(115200);
  // Start CC3000 chip
  Serial.println(F("\nInitializing…"));
  if (!cc3000.begin())
  {
    Serial.println(F("Couldn't begin()! Check your wiring?"));
    while(1);
  }
}

void loop(void)
{

  // Connect to WiFi network
  cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY);
  Serial.println(F("Connected!"));

  // Start watchdog
  wdt_enable(WDTO_8S);

  // Wait for DHCP to complete
  Serial.println(F("Request DHCP"));
  while (!cc3000.checkDHCP())
  {
    delay(100);
  }

  // Reset watchdog
  wdt_reset();

  // Get IP
  uint32_t ip = 0;
  Serial.print(F("www.dweet.io -> "));
  while  (ip  ==  0)  {
    if  (!  cc3000.getHostByName("www.dweet.io", &ip))  {
      Serial.println(F("Couldn't resolve!"));
    }
    delay(500);
  }
  cc3000.printIPdotsRev(ip);
  Serial.println(F(""));

  // Reset watchdog
  wdt_reset();

  // Measure from DHT sensor
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  temperature = (int)t;
  humidity = (int)h;

  // Measure light level
  float sensor_reading = analogRead(A0);
  light = (int)(sensor_reading/1024*100);
  Serial.println(F("Measurements done"));

  // Reset watchdog
  wdt_reset();

  // Send request to Dweet.io
  Adafruit_CC3000_Client client = cc3000.connectTCP(ip, 80);
  if (client.connected()) {
    Serial.print(F("Sending request… "));

    client.fastrprint(F("GET /dweet/for/"));
    client.print(thing_name);
    client.fastrprint(F("?temperature="));
    client.print(temperature);
    client.fastrprint(F("&humidity="));
    client.print(humidity);
    client.fastrprint(F("&light="));
    client.print(light);
    client.fastrprintln(F(" HTTP/1.1"));

    client.fastrprintln(F("Host: dweet.io"));
    client.fastrprintln(F("Connection: close"));
    client.fastrprintln(F(""));

    Serial.println(F("done."));

  } else {
    Serial.println(F("Connection failed"));
    return;
  }

  // Reset watchdog
  wdt_reset();

  // Read answer
  Serial.println(F("Reading answer…"));
  while (client.connected()) {
    while (client.available()) {
      char c = client.read();
      Serial.print(c);
    }
  }
  Serial.println(F(""));

  // Reset watchdog
  wdt_reset();

  // Close connection and disconnect
  client.close();
  Serial.println(F("Disconnecting"));
  Serial.println(F(""));
  cc3000.disconnect();

  // Reset watchdog & disable
  wdt_reset();
  wdt_disable();

  // Wait 10 seconds until next update
  delay(10000);

}
```

Let’s now go into the details of the code. It starts by importing the required libraries:

```
#include <Adafruit_CC3000.h>
#include <ccspi.h>
#include <SPI.h>
#include "DHT.h"
#include <avr/wdt.h>
```

We then define the pins on which the CC3000 module is connected to:

```
#define ADAFRUIT_CC3000_IRQ   3
#define ADAFRUIT_CC3000_VBAT  5
#define ADAFRUIT_CC3000_CS    10
```

Then, we can create an instance of the CC3000 WiFi chip:

```
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT, SPI_CLOCK_DIV2);
```

Now, you will need to modify the sketch to enter your own SSID network name, and the associated password. If your network is not using WPA2 authentication, you will also have to change this parameter:

```
#define WLAN_SSID       "yourWiFiNetwork"
#define WLAN_PASS       "yourPassword"
```

```
#define WLAN_SECURITY   WLAN_SEC_WPA2
```

You will also need to give a name to your ‘thing’. Note that all the things on dweet.io are public by default. It is not a problem here, as we just want to upload and monitor simple data that is not sensible at all. However, I recommend choosing a complicated name for the device so nobody else can find it. For example, you can use names like `weather_station_l5ir457xda`. Once you have a good name, you can enter it inside the sketch:

```
#define thing_name  "yourThingName"
```

We also need to define some variables that will contain the measurements made by the project:

```
int temperature;
int humidity;
int light;
```

In the `setup()` function of the sketch, we initialize the CC3000 chip:

```
Serial.println(F("\nInitializing…"));
if (!cc3000.begin())
{
  Serial.println(F("Couldn't begin()! Check your wiring?"));
  while(1);
}
```

In the `loop()` function, we first connect the CC3000 chip to your local WiFi network:

```
cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY);
```

We also need to start the watchdog. The watchdog is basically a circuit inside the Arduino microcontroller, independent from the code itself. It will reset the Arduino sketch automatically after a given time, unless we send it a reset signal. This will ensure that even if our Arduino sketch hangs for some reasons (for example, it cannot connect to the Dweet.io servers), the Arduino sketch will just reset itself and the project will continue to work. We first need to enable the watchdog with the maximal delay of 8 seconds:

```
wdt_enable(WDTO_8S);
```

When we are connected, we perform the temperature, humidity and light level measurements:

```
float t = dht.readTemperature();
float h = dht.readHumidity();
temperature = (int)t;
humidity = (int)h;

float sensor_reading = analogRead(A0);
```

```
light = (int)(sensor_reading/1024*100);
Serial.println(F("Measurements done"));
```

Once we got the measurements, we can upload them on the dweet.io service. To do so, we need to connect to their servers, and then send the data inside a standard HTTP GET request. This is done by the following piece of code:

```
Adafruit_CC3000_Client client = cc3000.connectTCP(ip, 80);
if (client.connected()) {
  Serial.print(F("Sending request… "));

  client.fastrprint(F("GET /dweet/for/"));
  client.print(thing_name);
  client.fastrprint(F("?temperature="));
  client.print(temperature);
  client.fastrprint(F("&humidity="));
  client.print(humidity);
  client.fastrprint(F("&light="));
  client.print(light);
  client.fastrprintln(F(" HTTP/1.1"));

  client.fastrprintln(F("Host: dweet.io"));
  client.fastrprintln(F("Connection: close"));
  client.fastrprintln(F(""));

  Serial.println(F("done."));
} else {
  Serial.println(F("Connection failed"));
  return;
}
```

We should also not forget to reset the watchdog after this long request:

```
wdt_reset();
```

After the data is sent, we read back the answer from the dweet.io server, to be sure that the data was correctly received on the other end. We also close the connection and disconnect the CC3000 chip from your WiFi network, in order to save energy:

```
Serial.println(F("Reading answer…"));
while (client.connected()) {
  while (client.available()) {
    char c = client.read();
    Serial.print(c);
  }
}
Serial.println(F(""));

// Reset watchdog
wdt_reset();

// Close connection and disconnect
client.close();
Serial.println(F("Disconnecting"));
Serial.println(F(""));
cc3000.disconnect();
```

After that, we also disable the watchdog:

```
wdt_disable();
```

Finally, we repeat this loop again after 10 seconds. Note that you can customize this delay if you want less frequent measurements:

```
delay(10000);
```

Note that the complete code for this chapter can be found on the corresponding folder inside the GitHub repository of the book:

https://github.com/openhomeautomation/iot-book

We are now going to test this code. Make sure that you entered the name of your thing inside the sketch, and that you also entered the data for your WiFi network. Then, upload the code to the Arduino board. Open the Serial monitor, and after a while you should see the answer coming back from the dweet.io server:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 174
Date: Thu, 24 Jul 2014 12:08:32 GMT
Connection: keep-alive

{"this":"succeeded","by":"dweeting","the":"dweet","with":
{"thing":"yourThingName","created":"2014-07-24T12:08:32.443Z",
"content":{"temperature":25,"humidity":35,"light":59}}}
```

You can also check online that the data was correctly recorded. Just type in a browser:

```
https://dweet.io/get/dweets/for/yourThingName
```

Of course, you have to replace the name with the name of thing you entered in the Arduino sketch. You should see that at least one measurement was inserted for this thing. For example:

```
{
  "thing": "weather_station_1a2cx3s8",
  "created": "2014-07-13T12:58:10.924Z",
  "content": {
    "temperature": 26,
    "humidity": 40,
    "light": 69
  }
}
```

In case it is not working at this point, there are several things you can check. First, make sure that you Internet connection is up and running, and that your Arduino board is actually connecting to your WiFi network. You can also check that the answer from the Dweet.io website is correct. Finally, make sure that you correctly entered your ‘Thing’ name inside the Arduino sketch.

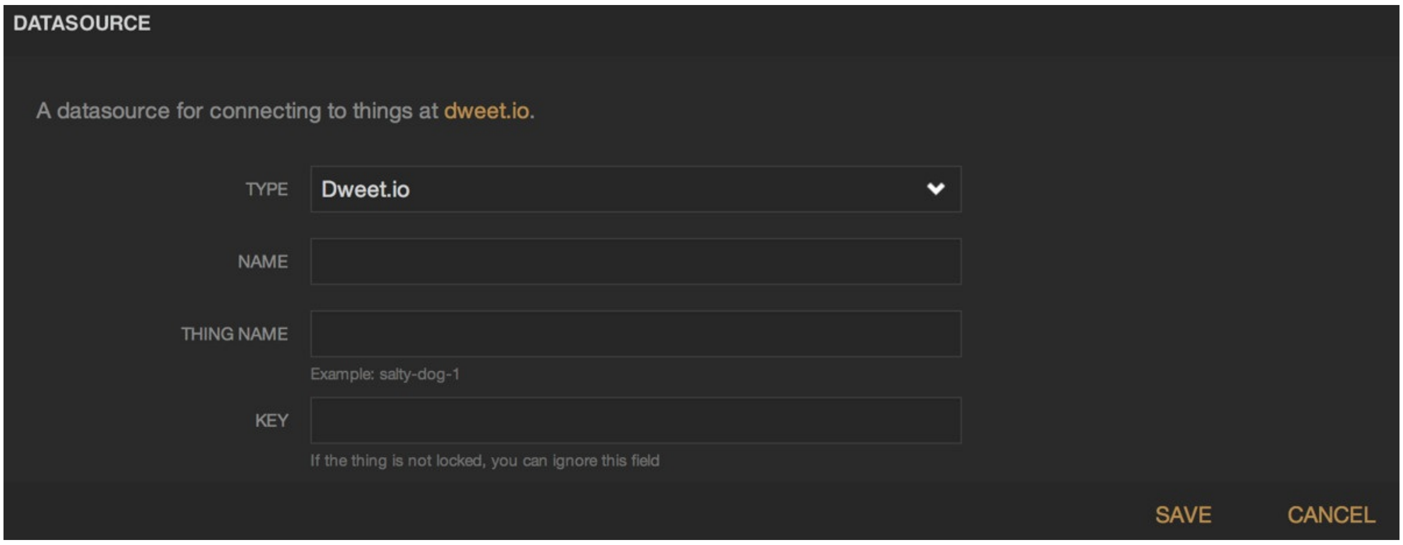
1.5 Monitoring Data in the Cloud

We are now able to store data in the cloud, using the dweet.io service. Your Arduino board is now continuously sending data over there. However, even if the data it returns is human-readable, it would be much better to visualize this data graphically. This is why we created an account on Freedboard before. Go over again to this website and log in:

https://www.freeboard.io/

Inside this interface, you will be able to create a new dashboard. This will be the main interface from which you will monitor everything. Do so by entering a name for your dashboard, and click on "Create New". Then, you have to create two things: panes and datasources.

Let's start with the datasource, which is the source of the data that will be displayed inside the dashboard. Click on "Add" to add a new data source, and select "Dweet.io" in the menu:



DATASOURCE

A datasource for connecting to things at dweet.io.

TYPE: Dweet.io

NAME:

THING NAME:

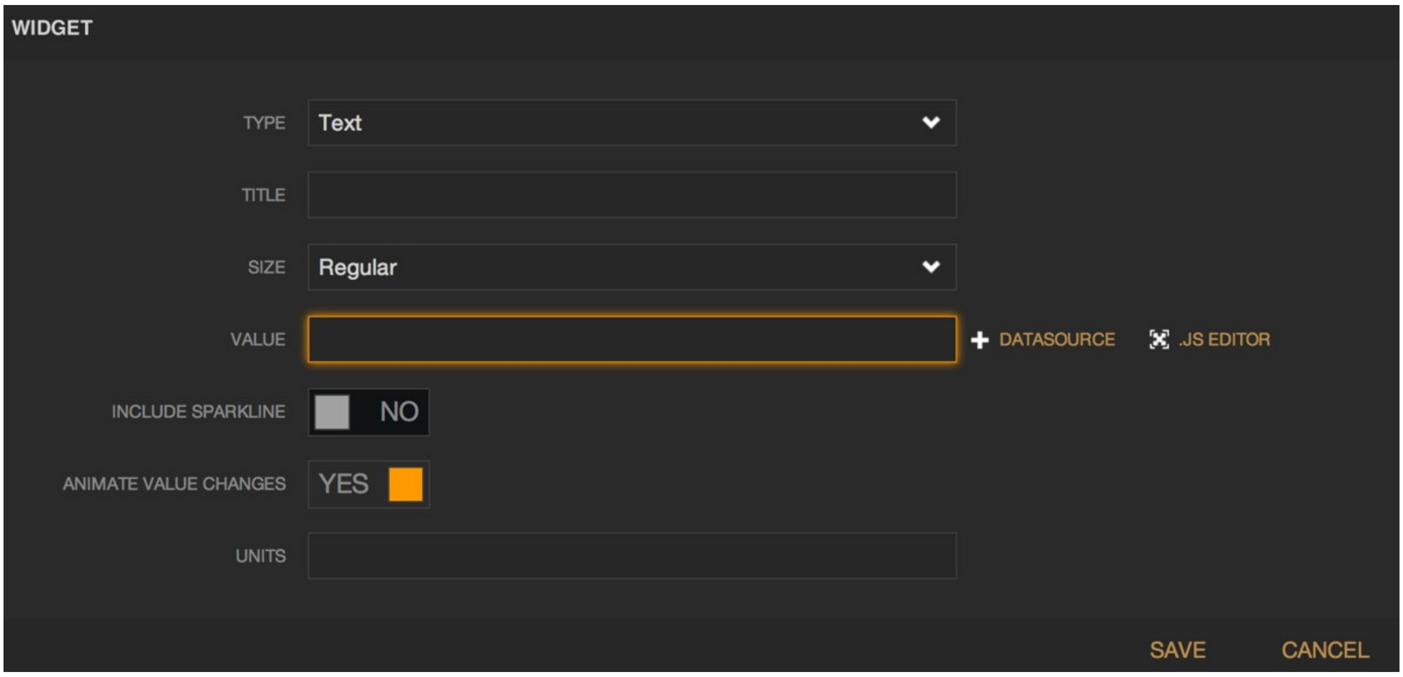
Example: salty-dog-1

KEY:

If the thing is not locked, you can ignore this field

Inside this box, you also need to enter a name, and the name of your thing that you defined in the Arduino sketch. Then, click on "Save".

Now, we need to create a new pane. A pane is basically a space on the dashboard where the data is displayed graphically. Click on the button, and then click on the pane "+" icon to add a widget. These widgets will display the data itself. I created three different panes, so I can display all the recorded data at the same time. I started with text widgets:


WIDGET

TYPE Text

TITLE

SIZE Regular

VALUE

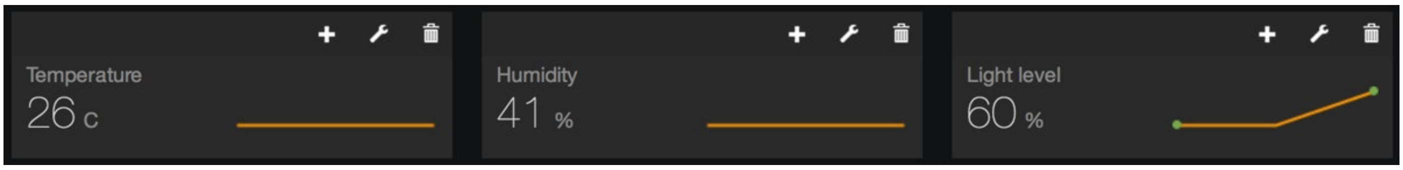
INCLUDE SPARKLINE NO

ANIMATE VALUE CHANGES YES

UNITS

The most important here is to select the "Value" field by clicking on "+ Datasource" on the right. This will link directly the widget with the data coming from dweet.io. For example, you can name your first widget temperature, and then link it to the temperature field of the datasource we defined earlier.

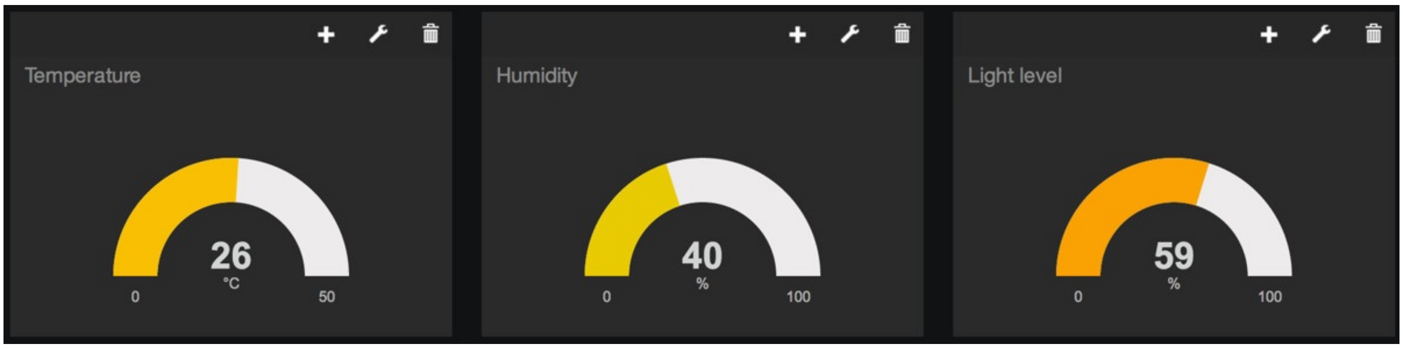
This is the result I got with three text widgets:


Temperature 26 c

Humidity 41 %

Light level 60 %

I also played with the interface, and changed these widgets for gauges widgets, but based on the same data:



You can also notice that the data is changing in live depending on the measured data. This means that you can now monitor this data from anywhere in the world, just by having the

URL of your dashboard.

In case it is not working at this point, make sure that the data is correctly received by the Dweet.io website first. Also make sure that you entered the correct ‘Thing’ name inside your Freeboard.io dashboard.

1.6 How to Go Further

Let’s summarize what you learned in this chapter. You just learned how to send measurements from your Arduino board to the cloud, so this data can be accessed from outside of your local WiFi network. We also saw how to monitor this data from a central online dashboard, so you can monitor it from anywhere.

You can of course improve this project in many ways. The first one is to add more sensors to the project, for example a wind speed sensor or a smoke sensor. You can also install many of these boards around your home, all uploading data to a different “thing” on dweet.io. Then, you can use what you learned about Freeboard to monitor all these measurements from a central place.

Chapter 2

Controlling a Lamp From Anywhere

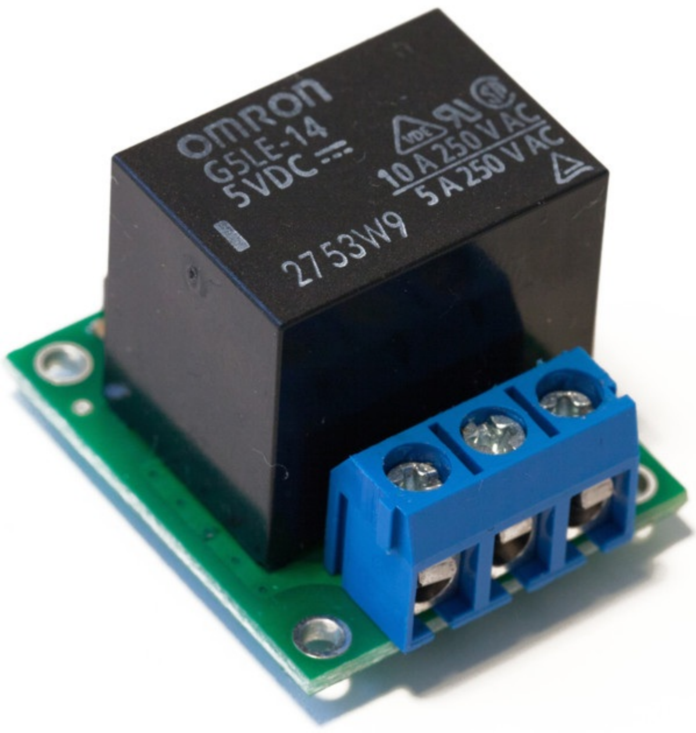
In this book so far, we built Internet of Things projects that sent data to the cloud so this data can be safely stored and monitored online. In this chapter, we are going to do something completely different. We are going to control an object, in this case a lamp, from the web.

To achieve this, we will use a service called Teleduino. We are going to use the Arduino Ethernet shield to connect your Arduino board to the web, connect a lamp to the project via a relay module, and then control this relay remotely. Because the Teleduino service works from any web browser, you will be able to control this lamp from anywhere in the world. Let’s dive in!

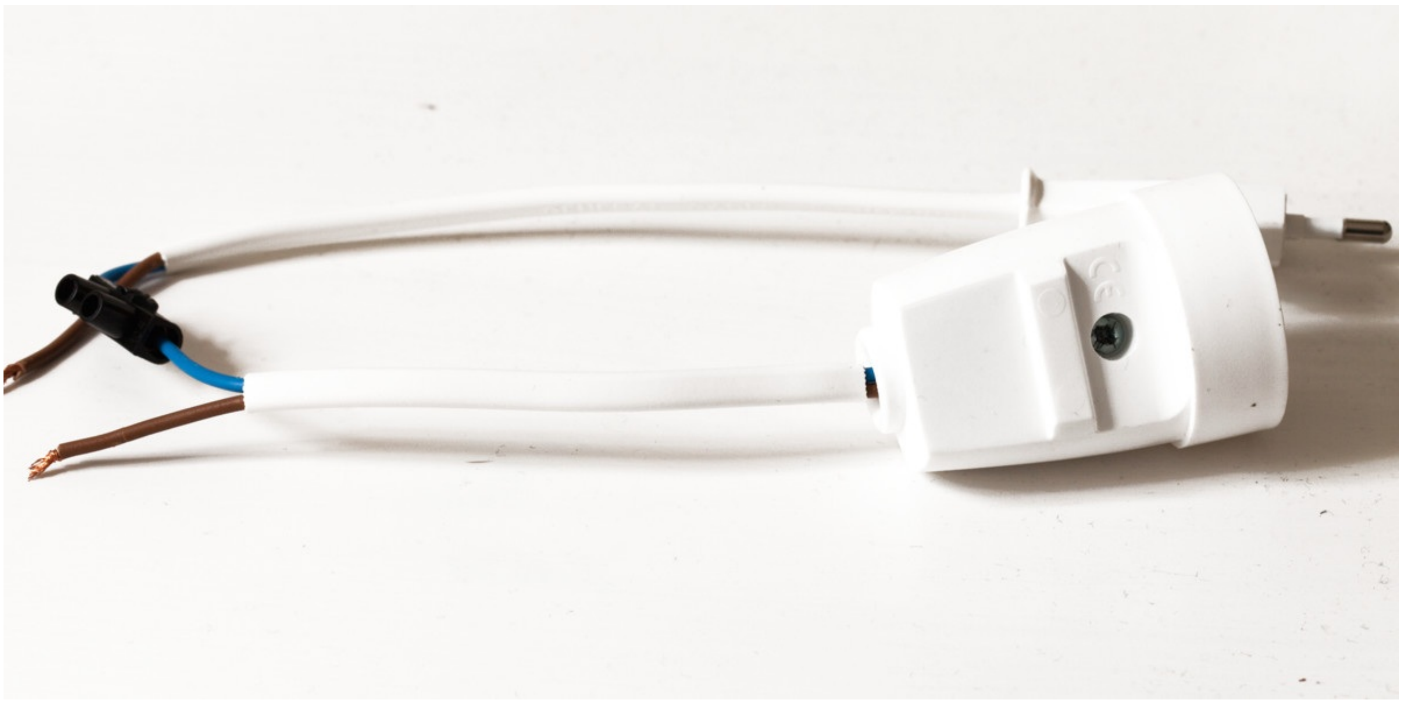
2.1 Hardware & Software Requirements

Let’s first see what we need to build this project. For the Arduino board, I simply used an Arduino Uno R3 board. For the Internet connectivity, I used the Arduino Ethernet shield, which is very convenient to connect your projects to the Internet and to build Internet of Things projects.

Then, for the relay module, I used a 5V relay module from Polulu, which nicely integrates a relay on a board, along with all the required components to control the relay from Arduino. This is a picture of the relay module I used:



To connect to the lamp to the project, I used a standard pair of power plugs with bare cables at the end, with one female socket (to plug the lamp in) and one male socket (to plug it into the power socket in the wall). This is a picture of the cables I used:



Note that you can perfectly test this project without having anything connected to the relay: you can build the Arduino part first, and then decide which device you want to connect to it later.

This a list of the components that are required for this project:

- Arduino Uno (http://www.pololu.com/product/2191)
- Arduino Ethernet Shield (https://www.adafruit.com/product/201)
- Relay module (http://www.pololu.com/product/2480)
- Female/male jumper wires (https://www.adafruit.com/products/824)

On the software side, you will of course need the usual Arduino IDE. You also need the Teleduino library, which you can download at:

https://www.teleduino.org/downloads/

To install the library, simply extract the folder in your Arduino /libraries folder (or create this folder if it doesn't exist yet).

You will also need to get an API key to use the Teleduino service. To do so, go to this link:

https://www.teleduino.org/tools/request-key

You will be taken to this page where you can request your key:

Request Key

A key is required to uniquely identify your Teleduino device.

Keys are generated and emailed to you within a few minutes.

Your privacy is protected, and your email address is not passed on to third parties.

Please fill in the below form to request a key. If you require multiple keys for multiple devices, you can complete the form multiple times using the same email address.

First Name:

Last Name:

Email Address:

Are you human? (type "yes"):

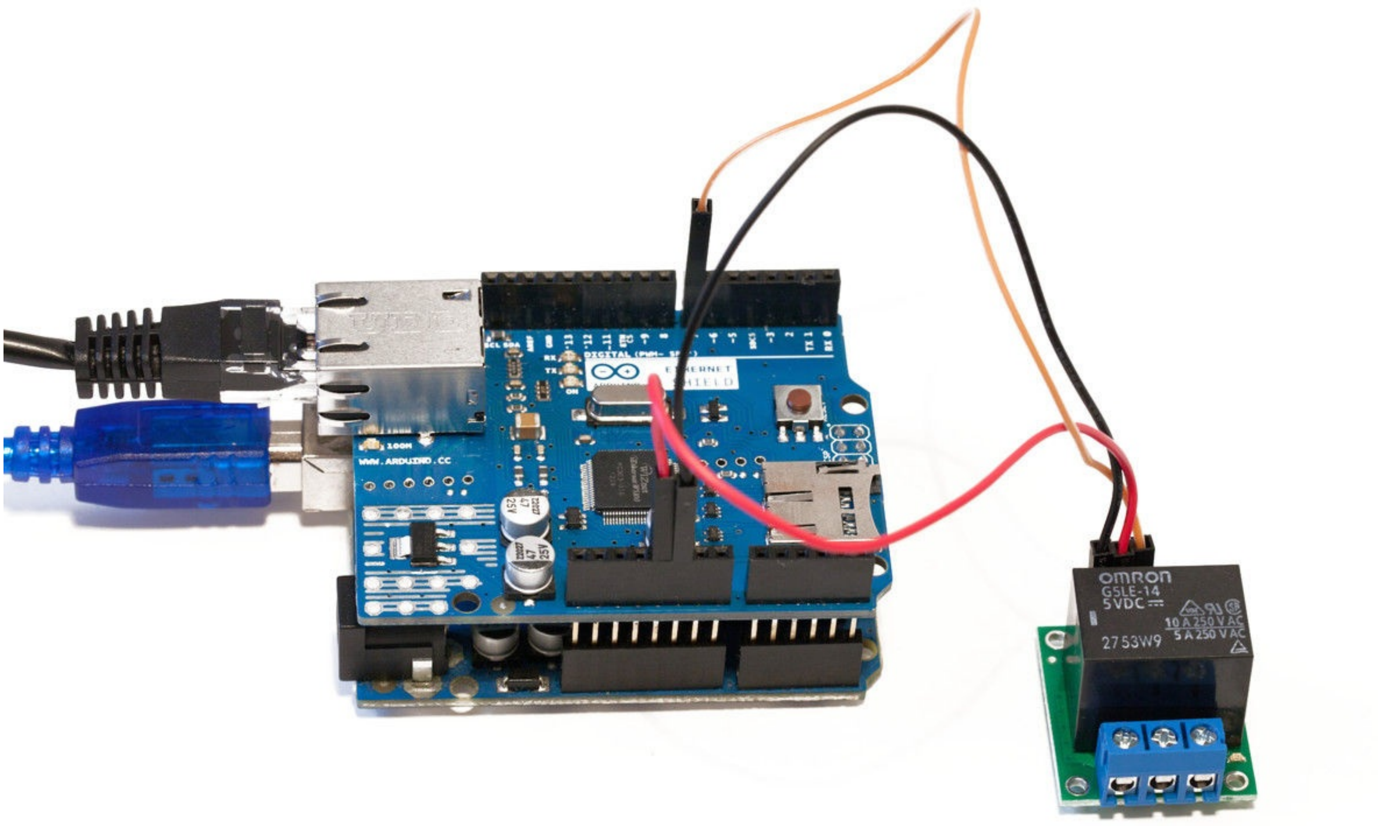
☐ I accept the Teleduino Terms and Conditions

Submit Request

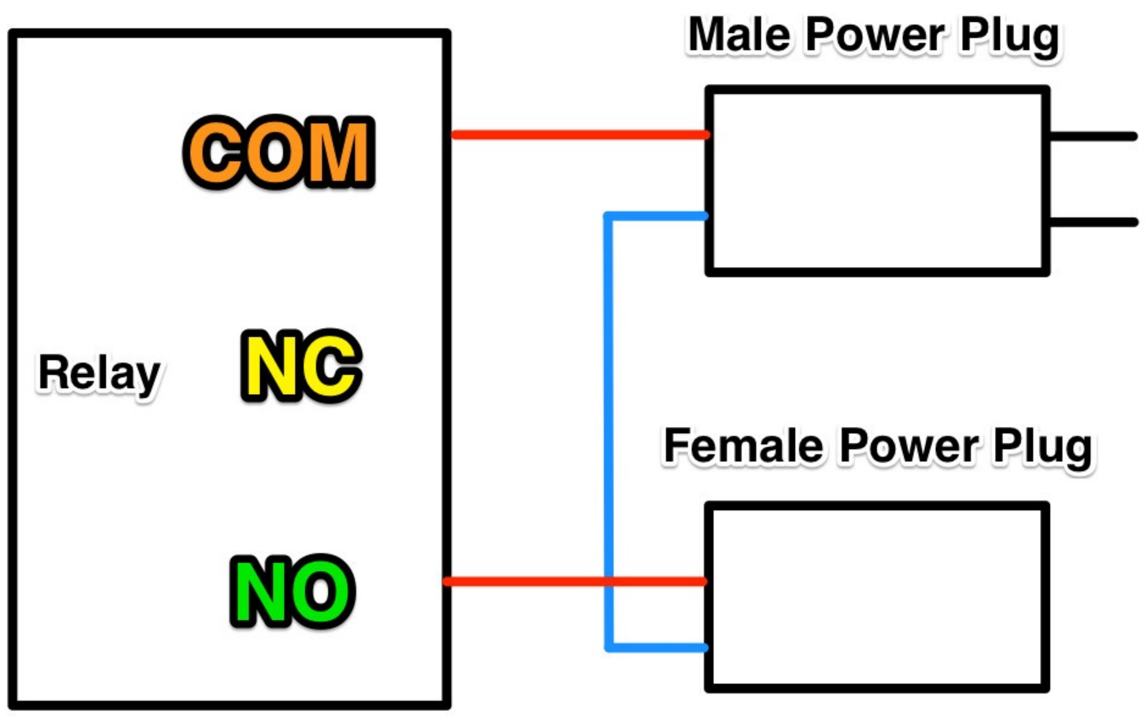
2.2 Connecting a Relay and a Lamp to Arduino

Assembling the hardware for this project is really simple. First, plug the Ethernet shield on top of your Arduino board, and connect the shield to your Internet router via an Ethernet cable.

For the relay module, there are three pins you need to connect: VCC, GND and SIG. VCC needs to go the Arduino 5V pin, and GND goes to the Arduino ground pin. Finally, connect the SIG pin to pin number 7 of the Arduino board. This is a picture of the project at this point:

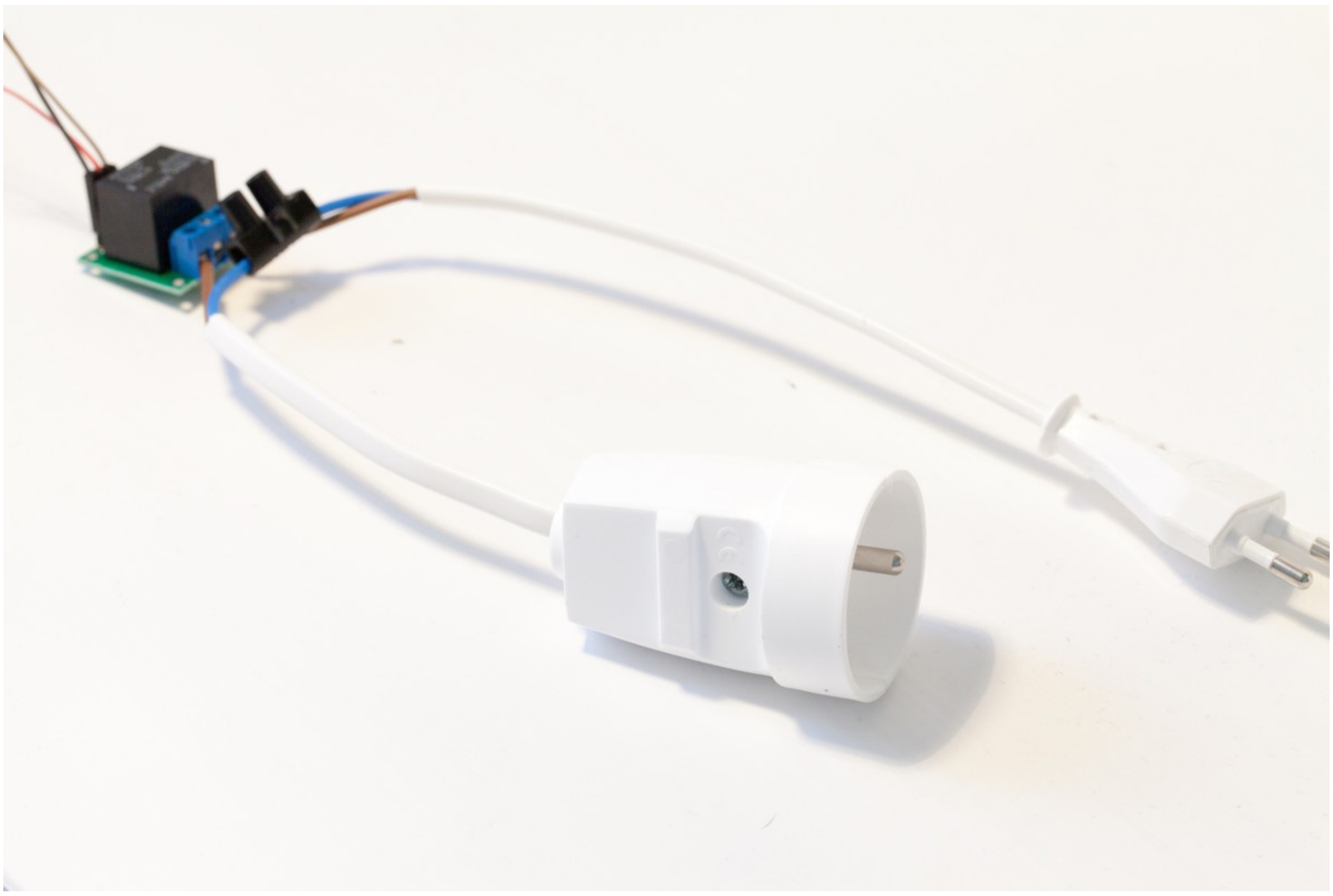


We are now going to connect the lamp to the hardware we already assembled. Basically, the idea is to have the main power supply (coming from the power socket in the wall) going to the relay on the COM pin, going out of the relay via the NO pin, and finally back to the lamp. This picture summarizes the connections between the different parts:



Note that I didn’t use any earth connections for this project, on both the male and the female plugs. Also note that this project is shown for education purposes only. It could be indeed really unsafe to use this project as it is in your home, as using the mains electricity directly is dangerous. If you want to install this project in your home, consider putting all the project in a plastic enclosure so it cannot be touched when in operation.

This is a picture of the completely assembled project:



2.3 Testing the Relay

It is now time to test the project. As the most important part of the project is the relay module, this is what we are going to test here. We are simply going to switch the relay on and off continuously every 5 seconds, just to check that the relay is working correctly. At this point, you can already plug a device (like the lamp I used as an example) into the project and connect the switch to an electrical socket, to see that all the connections are correctly made.

This is the complete code for this section:

```
// Sketch to test the relay

// Relay pin
const int relay_pin = 7;

void setup() {
  pinMode(relay_pin,OUTPUT);
}

void loop() {

  // Activate relay
  digitalWrite(relay_pin, HIGH);

  // Wait for 5 seconds
  delay(5000);

   // Deactivate relay
  digitalWrite(relay_pin, LOW);

  // Wait for 5 seconds
  delay(5000);
}
```

Note that all the files are available on the GitHub repository of the book:

https://github.com/openhomeautomation/iot-book

It is now time to test the sketch. Make sure that the lamp is correctly connected to the project, and that the male plug is plugged into the power socket in the wall. Then, upload the Arduino sketch to the board. You should see that every 5 seconds, the relay is switching, turning the device connected to it on and off.

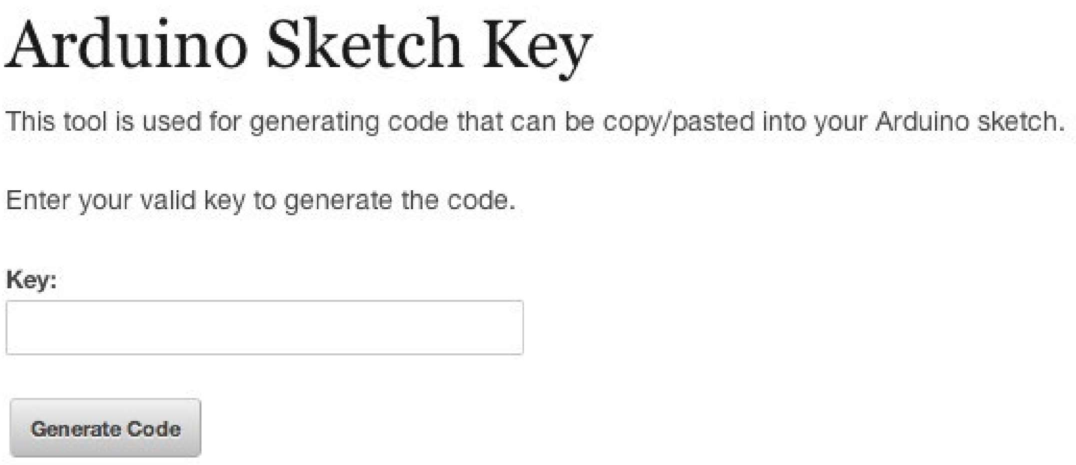
2.4 Controlling Your Project From Anywhere

Now that we are sure that the relay is working correctly, we can move on to the core of this chapter: controlling the relay from anywhere you are in the world. You will be able to control it from any web browser, just by knowing your API key that you got before.

There is one more thing you need to do before we can have a look into the Arduino sketch of this section. You need to convert the API key you got before to hexadecimal format, so it can be pasted inside the Arduino sketch. Luckily, the Teleduino website has a page to do that:

https://www.teleduino.org/tools/arduino-sketch-key

You will be taken to a page where you can paste your API key:


Arduino Sketch Key

This tool is used for generating code that can be copy/pasted into your Arduino sketch.

Enter your valid key to generate the code.

Key:

You will receive in return some code. Just copy this code somewhere else for now, or keep the web page open: we will need this hexadecimal code quite soon.

We are now going to see the most important points of the Arduino sketch of this section. The code itself is given by Teleduino and is really complex, so refer to the code inside the GitHub repository to get the complete code for this project.

The sketch starts by including the required libraries for the project:

```
#include <EEPROM.h>
#include <Servo.h>
#include <Wire.h>
#include <Teleduino328.h>
#include <SPI.h>
#include <Ethernet.h>
```

Then, you need to enter the MAC address of your Arduino Ethernet shield. You can find it below the shield itself. Just enter it in the `mac` variable:

```
byte mac[] = { 0x90, 0xA2, 0xDA, 0x0E, 0xFE, 0x40 };
```

Later in the code, you can also paste your API key in hexadecimal format:

```
byte key[] = { 0x64, 0x26, 0xFF, 0xC9,
               0x20, 0x4C, 0xF2, 0xCF,
               0xAE, 0x42, 0xD4, 0x1A,
               0xED, 0x6C, 0xB0, 0xB7 };
```

Now, we declare the pin on which you can connect a LED to monitor the status of Teleduino. I didn't use this feature for this project, but if you want to do so you can just connect a LED to this pin:

```
byte statusLedPin = 8;
```

We also declare an Ethernet client, that we will use to connect to the Teleduino service:

```
EthernetClient Client;
```

Now, in the `setup()` function of the sketch, we initialize the Ethernet shield using the `begin()` function. If this function is successful (meaning that the Ethernet shield got an IP address), we reset the Teleduino object:

```
if(!Ethernet.begin(mac))
{
  Teleduino328.setStatusLed(2, false, 10000);
  Teleduino328.reset();
}
```

In the 'loop()'' function of the sketch, the important part is to call the Teleduino instance whenever we are connected to the web:

```
if(Client.connected())
```

The rest of the code is about handling the request by the Teleduino service, and we will not get into the details of this part.

Note that all the files are available on the GitHub repository of the book:

https://github.com/openhomeautomation/iot-book

It's now time to test the project. Make sure that you grabbed the code from the GitHub repository, and that you insert the MAC address of your shield and the API key inside the code. Now, you can upload the code to the Arduino board.

Wait for a moment, and then open your favorite web browser. The first thing we have to do is to set the pin number 7 as an output. This can be done by typing the appropriate command in your web browser (make sure to replace yourKey by your own API key):

```
http://us01.proxy.teleduino.org/api/1.0/328.php?k=yourKey&
r=definePinMode&pin=7&mode=1
```

After typing this command, you should get a confirmation inside your web browser. After that, you can type the following command to turn the relay on:

```
http://us01.proxy.teleduino.org/api/1.0/328.php?k=yourKey&
r=setDigitalOutput&pin=7&output=1
```

You should see that the lamp turns on instantly. To switch it off again, simply type:

```
http://us01.proxy.teleduino.org/api/1.0/328.php?k=yourKey&
r=setDigitalOutput&pin=7&output=0
```

Congratulations, you can now control this lamp from anywhere in the world!

In case it is not working at this point, first make sure that your Internet connection is up and running. Also, make sure that you correctly enterered your Teleduino API key. Finally, make sure to always use the latest version of the code from the GitHub repository of the book.

2.5 How to Go Further

Let’s summarize what we did in this chapter. We built a project to control a lamp (or any other electrical device in your home) that can be controlled from anywhere. We used the Arduino Ethernet shield to connect the project to the web, and the Teleduino service so you can simply control your project from a web browser. Because the Teleduino service works wherever you are in the world, this lamp can now be controlled from anywhere.

There are of course many ways to improve this project. Of course, you can plug more than one relay module to the project, allowing you to control many different electrical devices at once. For example, you can connect many lamps to the project, along with other electrical appliance like water heaters or coffee machines.

You can also read data using this project. Using this function of Teleduino, you can read from sensors remotely. For example, you can imagine inserting a current sensor into the project, to be able to follow the energy consumption of your lamp remotely. To get more details on how to do so, I invite you to read the Teleduino API documentation on their website:

https://www.teleduino.org/documentation/api/328-full

Chapter 3

A Weather Station in the Cloud

In this chapter, you are going to build a weather measurement station that will automatically send data to the cloud. To do so, we will use the powerful Arduino Yun board with the Temboo service.

Introduced in 2013 by Arduino, the Arduino Yun is a powerful Arduino board with on board WiFi & that also feature a small Linux machine. It is especially really easy to interface with the web service Temboo, which is what we are going to use in this chapter. We will automatically send data to a Google Docs spreadsheet, which can be accessed from anywhere.

3.1 Hardware & software requirements

For this project, you will of course need the Arduino Yun board. You will also need a DHT11 (or DHT22) sensor, along with a 4.7K resistor, for humidity measurements.

For pressure & temperature measurements, I used a BMP085 sensor on a breakout board. You can also use the newer BMP180 sensor, which works with the same library.

For light levels measurements, I used a photocell with a 10K Ohm resistor. Finally, I used a breadboard and some male-male jumper wires.

- Arduino Yun (https://www.adafruit.com/product/1498)
- DHT11 sensor + 4.7K resistor (https://www.adafruit.com/product/386)
- Photocell (https://www.adafruit.com/product/161)
- 10K Ohm resistor (https://www.sparkfun.com/products/8374)
- BMP180 sensor (https://www.adafruit.com/product/1603)
- Some male/male jumper wires (https://www.adafruit.com/product/758)
- Breadboard (https://www.adafruit.com/product/64)

On the software side, you will need the Arduino IDE 1.5.x in the latest version (in Beta when this article was written):

http://arduino.cc/en/Main/Software#toc3

You will also need the DHT library:

https://github.com/adafruit/DHT-sensor-library

The BMP085 or BMP180 library:

https://github.com/adafruit/Adafruit_BMP085_Unified

And the Adafruit unified sensor library:

https://github.com/adafruit/Adafruit_Sensor

To install a library, simply put the folder in your /libraries/ folder of you main Arduino folder. This tutorial also assumes that your Arduino Yun is already connected to your WiFi network. If you need help with that, please follow the dedicated tutorial on the Arduino website:

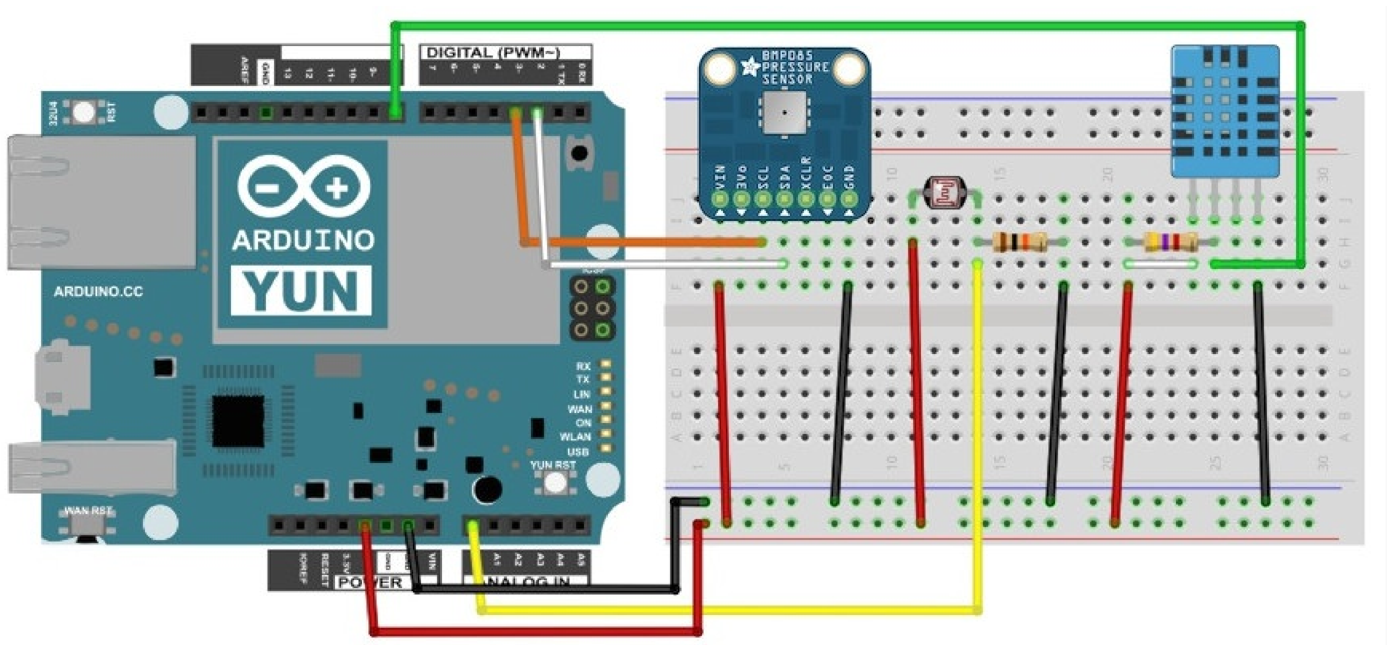
http://arduino.cc/en/Guide/ArduinoYun

You will also need a Google account for the rest of the project. If you don’t have one yet, please create one, for example by going on the Google Drive page:

https://drive.google.com/

3.2 Hardware configuration

The hardware connections for this project are actually quite simple. We have to connect the DHT11 sensor, the pressure sensor, and the part responsible for the light level measurement. The following picture summarizes the hardware connection:



This image was created with Fritzing (http://fritzing.org/).

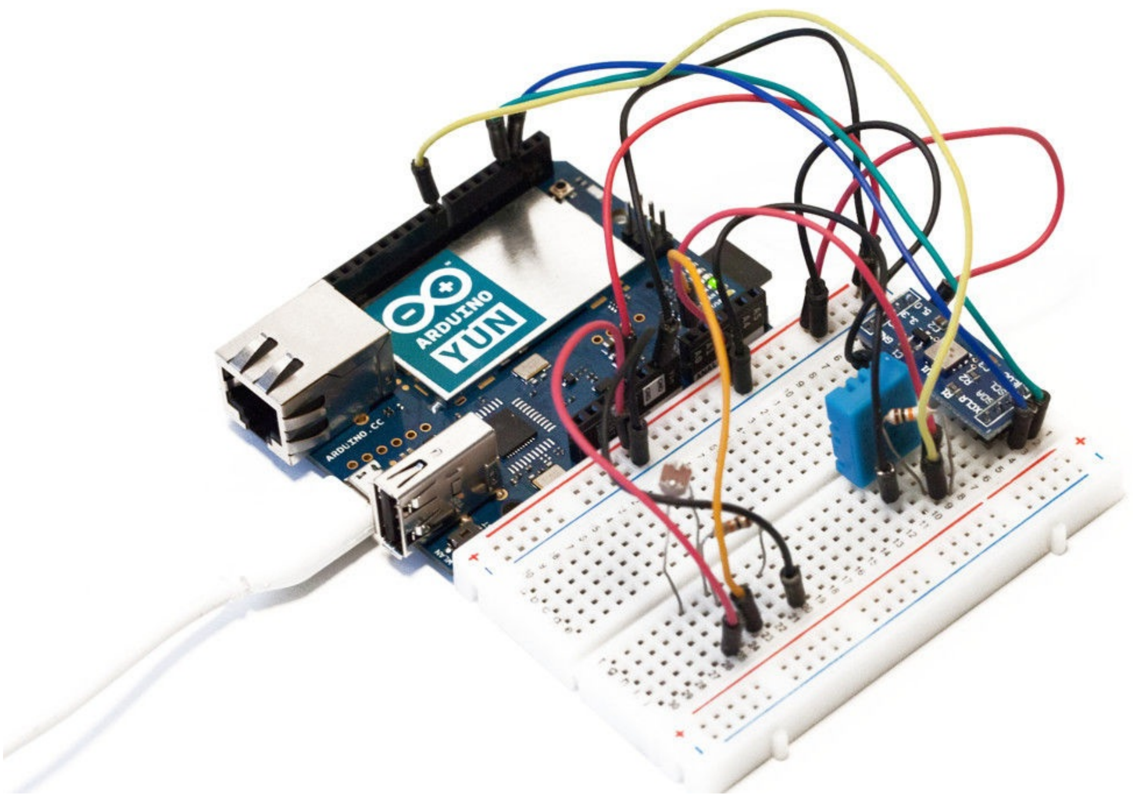
First, place the DHT11 sensor and the BMP sensor on the breadboard. After that, connect the Arduino Yun +5V pin to the red rail on the breadboard, and the ground pin to the blue rail.

Then, connect pin number 1 of the DHT11 sensor (see the schematic) to the red rail on the breadboard, and pin number 4 the blue rail. Also connect pin number 2 to pin number 8 of the Arduino Yun. To complete the connections of the DHT11 sensor, connect the 4.7k Ohm between pin number 1 and 2 of the sensor.

For the photocell, first place the cell in series with the 10k Ohm resistor on the breadboard. Then, connect the other end of the photocell to the red rail on the breadboard, and the other end of the resistor to the ground. Finally, connect the common pin to the Arduino Yun analog pin A0.

For the BMP085 or BMP180 sensor, connect the VIN pin to the +5V and GND to Ground. Then, connect the SCL pin to Arduino Yun pin number 3, and the SDA pin to Arduino Yun pin number 2.

You can see how the assembled project looks like on this picture:



3.3 Testing The Sensors

Before connecting the project to the cloud, we will test every sensors individually. To do so, we will write a simple test sketch. This is the complete code for this part:

```
// Include required libraries
#include "DHT.h"
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085_U.h>

// Variables
int lightLevel;
float humidity;
float temperature;

unsigned long time;

// DHT11 sensor pins
#define DHTPIN 8
#define DHTTYPE DHT11

// DHT & BMP instances
DHT dht(DHTPIN, DHTTYPE);
Adafruit_BMP085_Unified bmp = Adafruit_BMP085_Unified(10085);

void setup(void)
{

  // Initialize DHT sensor
  dht.begin();

  // Init serial
  Serial.begin(115200);

  // Initialise the sensor
  if(!bmp.begin())
  {
    Serial.print("Ooops, no BMP085 detected… Check your wiring or I2C ADDR!");
    while(1);
  }

}

void loop(void)
{

    // Measure the humidity
    float humidity = dht.readHumidity();

    // Measure light level
    int lightLevel = analogRead(A0);

    // Measure pressure & temperature from BMP sensor
    sensors_event_t event;
    bmp.getEvent(&event);
    float pressure = event.pressure;

    float temperature;
    bmp.getTemperature(&temperature);

    float seaLevelPressure = SENSORS_PRESSURE_SEALEVELHPA;
    float altitude;
    altitude = bmp.pressureToAltitude(seaLevelPressure,
                                      event.pressure,
                                      temperature);

    // Print measurements
    Serial.print("Humidity: ");
    Serial.println(humidity);
    Serial.print("Light level: ");
    Serial.println(lightLevel);
    Serial.print("Barometric pressure: ");
    Serial.println(pressure);
    Serial.print("Temperature: ");
    Serial.println(temperature);
    Serial.print("Altitude: ");
    Serial.println(altitude);
    Serial.println("");

    // Repeat 50 ms
    delay(50);

}
```

Let's now see the details of the code. The first step in the code is to import the correct libraries:

```
#include "DHT.h"
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085_U.h>
```

We need to define the pin & type of the DHT sensor:

```
#define DHTPIN 8
#define DHTTYPE DHT11
```

And create the instances for the DHT sensor & BMP sensor:

```
DHT dht(DHTPIN, DHTTYPE);
Adafruit_BMP085_Unified bmp = Adafruit_BMP085_Unified(10085);
```

In the `setup()` function, we can initialize the BMP sensor:

```
bmp.begin()
```

In the `loop()` part of the sketch, we make the different measurements:

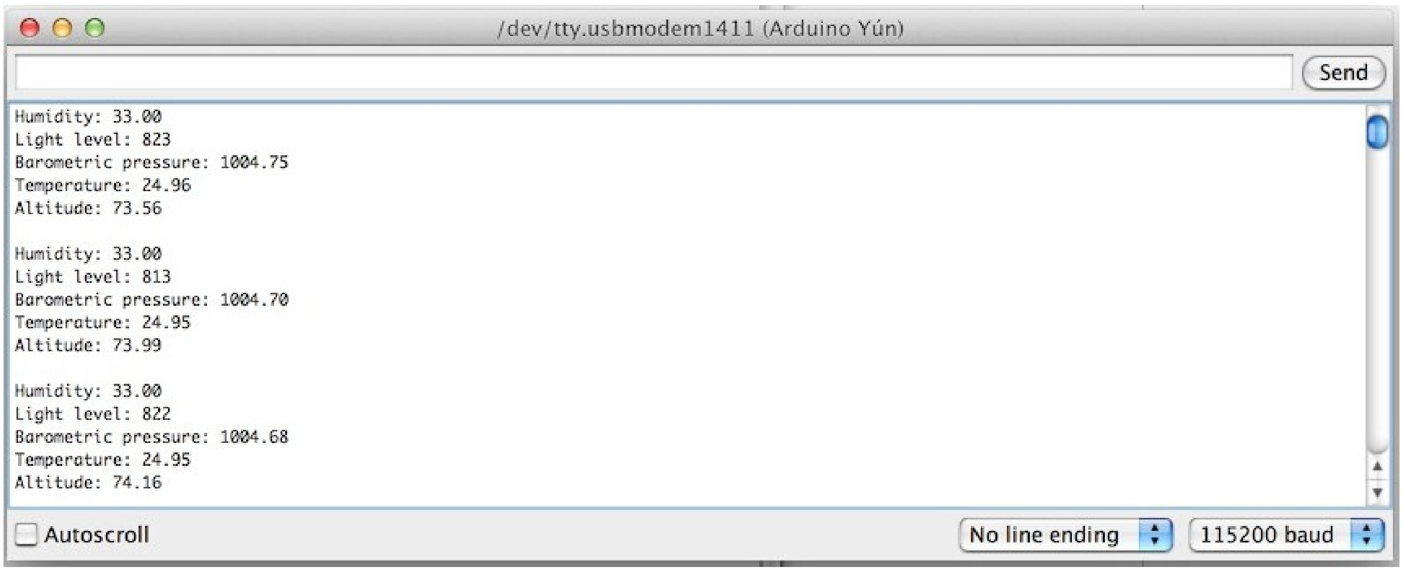
```
// Measure the humidity
float humidity = dht.readHumidity();

// Measure light level
int lightLevel = analogRead(A0);

// Measure pressure & temperature from BMP sensor
sensors_event_t event;
bmp.getEvent(&event);
float pressure = event.pressure;
```

These measurements will then be printed on the Serial monitor.

Note that all the code is available inside the GitHub repository of the book:

https://github.com/openhomeautomation/iot-book

You can now upload the sketch to the board & open the Serial monitor (making sure the Serial speed matches the one you defined in the code). This is what you should see:



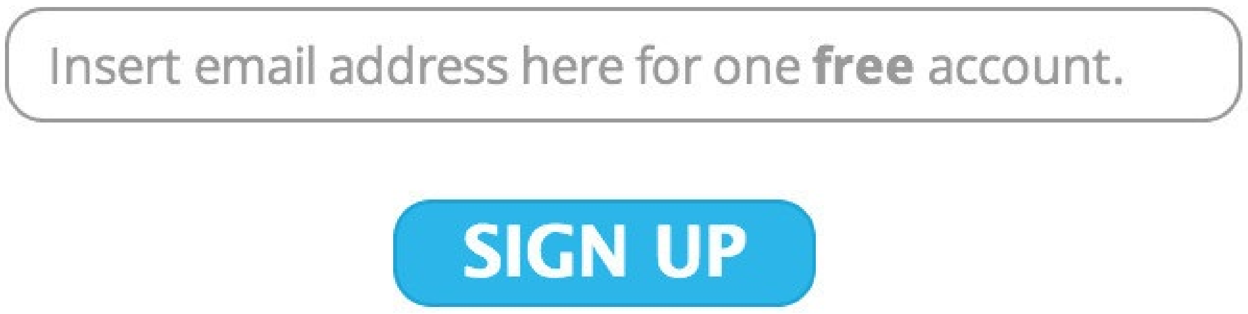
```
Humidity: 33.00
Light level: 823
Barometric pressure: 1004.75
Temperature: 24.96
Altitude: 73.56

Humidity: 33.00
Light level: 813
Barometric pressure: 1004.70
Temperature: 24.95
Altitude: 73.99

Humidity: 33.00
Light level: 822
Barometric pressure: 1004.68
Temperature: 24.95
Altitude: 74.16
```

If that works, it means all your hardware connections are correct, and you can move to the next part of the project.

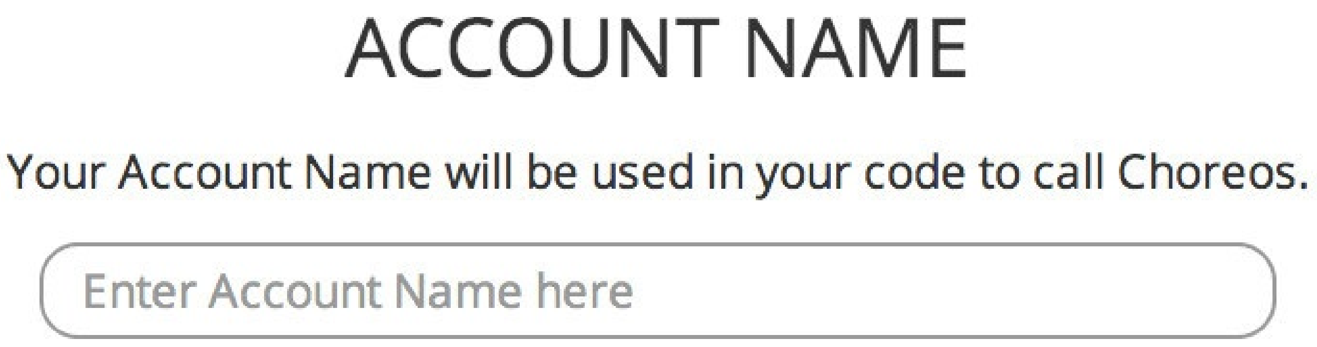
In case it is not working at this point, there are several things you can check. First, make sure you that correctly downloaded & installed all the Arduino libraries that are required for this project. Also make sure that you correctly followed the instructions to assemble the project.

3.4 Setting up your Temboo account

Before you can upload data to Google Docs, you need to have a Temboo account. Go over to the Temboo website and enter your email to start the account creation process:



You will also be prompted to enter an account name:

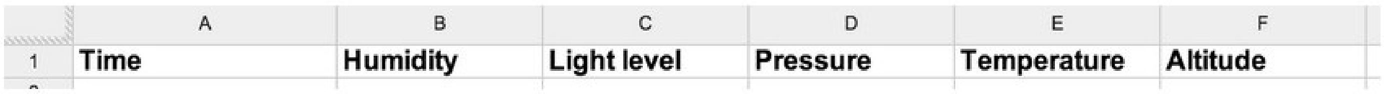

ACCOUNT NAME

Your Account Name will be used in your code to call Choreos.

After that step, you can go to the “Account” tab, and then to “Applications”. You will find your first application name (should be “myFirstApp” by default) and your API key. Please keep your account name, application name and API key at hand, you will need them soon.

3.5 Logging Data to Google Docs

It is now time to build the software part of our project. First, we will do some configuration in Google Docs. Create a new spreadsheet in Google Docs, and give it a name (I named mine “Yun”). Then, enter the title of each columns in the first row, like the picture below:



	A	B	C	D	E	F
1	Time	Humidity	Light level	Pressure	Temperature	Altitude

Now, we are going to build the Arduino sketch. This is the complete code for this part:

```
#include <Bridge.h>
#include <Temboo.h>
#include <Process.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085_U.h>

// Contains Temboo account information
#include "TembooAccount.h"

// Variables
int lightLevel;
float humidity;
float temperature;
unsigned long time;

// Process to get the measurement time
Process date;

// Your Google Docs data
const String GOOGLE_USERNAME = "yourEmailAddress";
const String GOOGLE_PASSWORD = "yourPassword";
const String SPREADSHEET_TITLE = "Yun";

// Include required libraries
#include "DHT.h"

// DHT11 sensor pins
#define DHTPIN 8
#define DHTTYPE DHT11

// DHT & BMP instances
DHT dht(DHTPIN, DHTTYPE);
Adafruit_BMP085_Unified bmp = Adafruit_BMP085_Unified(10085);

// Debug mode ?
boolean debug_mode = false;

void setup() {

  // Start Serial
  if (debug_mode == true){
    Serial.begin(115200);
    delay(4000);
    while(!Serial);
  }

  // Initialize DHT sensor
  dht.begin();
  // Start bridge
  Bridge.begin();

  // Start date process
  time = millis();
  if (!date.running())  {
    date.begin("date");
    date.addParameter("+%D-%T");
    date.run();
  }

  if (debug_mode == true){
    Serial.println("Setup complete. Waiting for sensor input…\n");
  }

  // Initialise the sensor
  if(!bmp.begin())
  {
    while(1);
  }
}

void loop() {

  // Measure the humidity & temperature
  humidity = dht.readHumidity();

  // Measure light level
  int lightLevel = analogRead(A0);

  // Measure pressure & temperature from BMP sensor
  sensors_event_t event;
  bmp.getEvent(&event);
  float pressure = event.pressure;

   bmp.getTemperature(&temperature);

   float seaLevelPressure = SENSORS_PRESSURE_SEALEVELHPA;
   float altitude;
   altitude = bmp.pressureToAltitude(seaLevelPressure,
                                        event.pressure,
                                        temperature);

  if (debug_mode == true){
    Serial.println("\nCalling the AppendRow Choreo…");
  }

  // Append data to Google Docs sheet
  runAppendRow(humidity, lightLevel, pressure, temperature, altitude);

  // Repeat every 10 minutes
  delay(600000);
}

// Function to add data to Google Docs
void runAppendRow(float humidity, int lightLevel,
  float pressure, float temperature, float altitude) {
  TembooChoreo AppendRowChoreo;

  // Invoke the Temboo client
  AppendRowChoreo.begin();

  // Set Temboo account credentials
  AppendRowChoreo.setAccountName(TEMBOO_ACCOUNT);
  AppendRowChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
  AppendRowChoreo.setAppKey(TEMBOO_APP_KEY);

  // Identify the Choreo to run
  AppendRowChoreo.setChoreo("/Library/Google/Spreadsheets/AppendRow");

  // your Google username (usually your email address)
  AppendRowChoreo.addInput("Username", GOOGLE_USERNAME);

  // your Google account password
  AppendRowChoreo.addInput("Password", GOOGLE_PASSWORD);
  // the title of the spreadsheet you want to append to
  AppendRowChoreo.addInput("SpreadsheetTitle", SPREADSHEET_TITLE);

  // Restart the date process:
  if (!date.running())  {
    date.begin("date");
    date.addParameter("+%D-%T");
    date.run();
   }

  // Get date
  String timeString = date.readString();

  // Format data
  String data = "";
  data = data + timeString + "," +
  String(humidity) + "," +
  String(lightLevel) + "," +
  String(pressure) + "," +
  String(temperature) + "," +
  String(altitude);

  // Set Choreo inputs
  AppendRowChoreo.addInput("RowData", data);

  // Run the Choreo
  unsigned int returnCode = AppendRowChoreo.run();

  // A return code of zero means everything worked
  if (returnCode == 0) {
    if (debug_mode == true){
      Serial.println("Completed execution of the AppendRow Choreo.\n");
    }
  } else {
    // A non-zero return code means there was an error
    // Read and print the error message
    while (AppendRowChoreo.available()) {
      char c = AppendRowChoreo.read();
      if (debug_mode == true){ Serial.print(c); }
    }
    if (debug_mode == true){ Serial.println(); }
  }
  AppendRowChoreo.close();
}
```

Let's now see the details of the code. You need to import all the libraries for the Arduino Yun web & bridge functionalities:

```
#include <Bridge.h>
#include <Temboo.h>
#include <Process.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085_U.h>
```

I also placed all my Temboo account information in a separate file called `Temboo.h`. Please make sure that you update your personal account information in this file. This is the content of this file:

```
#define TEMBOO_ACCOUNT "tembooAccountName"  // Your Temboo account name
#define TEMBOO_APP_KEY_NAME "myFirstApp"  // Your Temboo app key name
#define TEMBOO_APP_KEY "tembooAccountKey"  // Your Temboo app key
```

We then have to include it inside the sketch:

```
#include "TembooAccount.h"
```

You also need to set your own informations about your Google account:

```
const String GOOGLE_USERNAME = "yourEmailAddress";
const String GOOGLE_PASSWORD = "yourPassword";
const String SPREADSHEET_TITLE = "Yun";
```

Note that if you are using Google’s two-step verification, you will need to get an application-specific password for this step. You can get one here:

https://security.google.com/settings/security/apppasswords

In the `setup()` function, we start the Bridge between the Arduino part & the Linux machine with:

```
Bridge.begin();
```

We also start a date process, to automatically send the measurements date to Google Docs:

```
time = millis();
if (!date.running())  {
  date.begin("date");
  date.addParameter("+%D-%T");
  date.run();
}
```

In the `loop()` function, the most important line is:

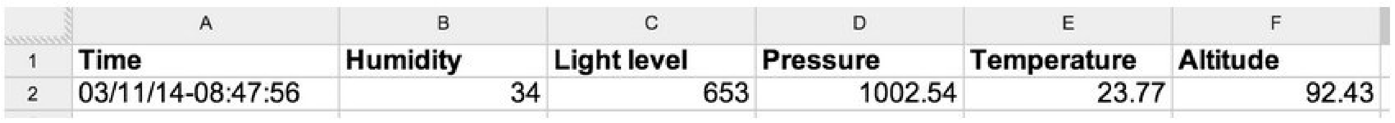
```
runAppendRow(humidity, lightLevel, pressure, temperature, altitude);
```

This function calls the function to automatically send the data to the Google Docs spreadsheet. I won’t get into the details of this function, but of course you will find all the code in the GitHub repository of this project. The `loop()` of the Arduino sketch is repeated every 10 minutes with a `delay()` function, as these variables are usually changing slowly over time.

Note that all the code is available inside the GitHub repository of the book:

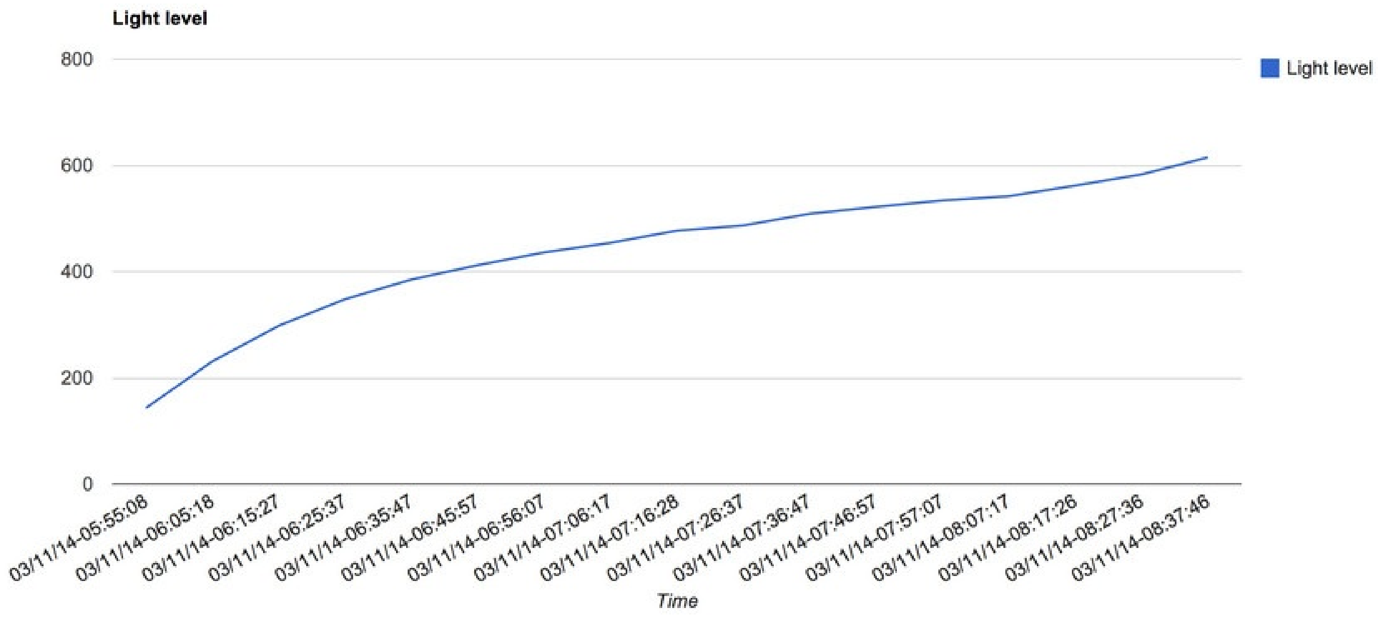
https://github.com/openhomeautomation/iot-book

Finally, you are ready to test the project. Make sure you modified the files with your own data, upload the code to the Arduino Yun board. After that, open the Google Docs spreadsheet in your browser again, and wait a moment. After a while, the first measurement should appear inside the spreadsheet:

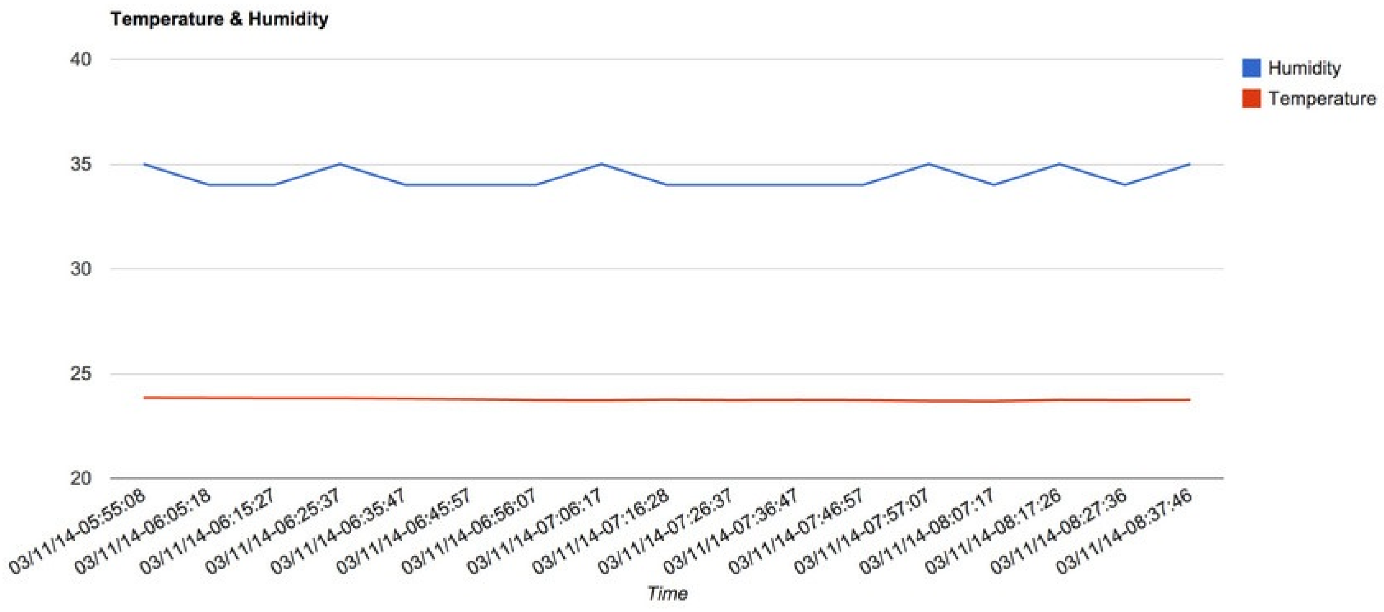


	A	B	C	D	E	F
1	Time	Humidity	Light level	Pressure	Temperature	Altitude
2	03/11/14-08:47:56	34	653	1002.54	23.77	92.43

I also used the built-in plotting functions of Google Docs to plot my data as it comes in. For example, I used the simple "Line" chart type to plot the light level. This is the result over some hours in the morning, showing that the light level is slowly rising over time:



And over the same period of time, I plotted the evolution of the temperature & humidity:



The really nice thing about this project is that it is updated automatically as data comes in. It can also be accessed from anywhere: you just need your Google user name & password!

In case it is not working at this point, there are several things you can check. First, make sure that you correctly configured your Arduino Yun board, and that it can access the Internet. Also make sure that you correctly set up your Google Docs spreadsheet. Finally, make sure that you entered your Temboo & Google account credentials inside the Arduino files after your downloaded the files from the GitHub repository of the book.

3.6 How to Go Further

Let’s summarize what you learned in this chapter. You just learned how to send measurements from your Arduino Yun board to Google Docs, so this data can be accessed from anywhere. We also saw how to plot this data, so you can monitor measurements live from your web browser.

You can of course improve this project in many ways. The first one is to add more sensors to the project, for example a wind speed sensor or a smoke sensor. You can of course use many Arduino Yun boards, for example in different parts of your home.

Chapter 4

Wireless Security Camera

Ever saw these wireless security cameras that you can buy off the shelf? These are devices that you can just put somewhere in your home or outside and connect them to your WiFi network. After that, they automatically take pictures if some motion is detected, for example. However, they are usually using the interface given by the manufacturer, which means you are quite limited with what you can do with your camera.

In this chapter, we are going to build our own DIY version of such device. The project is based on the Arduino Yun, to which we are going to connect a standard USB webcam and a PIR motion sensor to create a nice Internet of Things application.

The application will be a modern version of a standard task that you expect from a security camera: taking pictures when some motion is detected. The project will store pictures taken by the USB camera on an SD card inserted into the Yun, but that is not all. Because we are in a book about the Internet of Things, we also want these pictures to be automatically uploaded on a secure location. And that is exactly what we are going to do by uploading the pictures to Dropbox at the same time.

4.1 Hardware & Software Requirements

The first step is to make sure you have the right hardware components. First off, you will need an Arduino Yun, and a PIR motion sensor.

For the USB camera, you can choose any webcam that is compatible with the UVC protocol. Most of the recent webcams are compatible. I chose a Logitech C270 that can take pictures up to 720p resolution. You can find a list of compatible cameras here:

http://en.wikipedia.org/wiki/List_of_USB_video_class_devices

You will also need a microSD card to store the pictures locally on your Arduino Yun, and some female/male jumper wires to connect the PIR motion sensor.

This a list of the components that are required for this project:

- Arduino Yun (https://www.adafruit.com/product/1498)
- USB webcam (http://www.logitech.com/en-us/product/hd-webcam-c270)
- PIR motion sensor (https://www.adafruit.com/product/189)
- MicroSD card (https://www.adafruit.com/products/102)
- Some female/male jumper wires (https://www.adafruit.com/products/825)

Before we can build exciting applications with our hardware, we need to setup some accounts on the web services we are going to use. The first one is Temboo, where you will need to create an account (if you didn't create one yet after following the previous chapter). Temboo will basically make the interface between the Arduino Yun and Dropbox. Just go over to:

https://www.temboo.com/

You will be prompted to create an account, and then your first app. Write down the name of your account, the name of the app and your app API key, you will need them later.

There is also one more thing you need from the Temboo website: the Temboo Python SDK. We will use it later to upload pictures to Dropbox from the Arduino Yun. You can get it at:

https://www.temboo.com/python

Once downloaded, simply extract the “temboo” folder at the root of the microSD card.

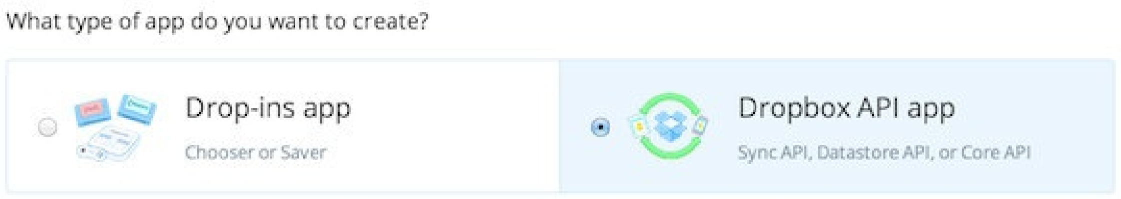
Then, you need a Dropbox account. Go over to the Dropbox website to do so:

https://www.dropbox.com/home

Once the account is created, you need to create an App so the Yun can upload pictures to your Dropbox folder. To create an app, you need to go to the developers section of Dropbox:

https://www.dropbox.com/developers/apps

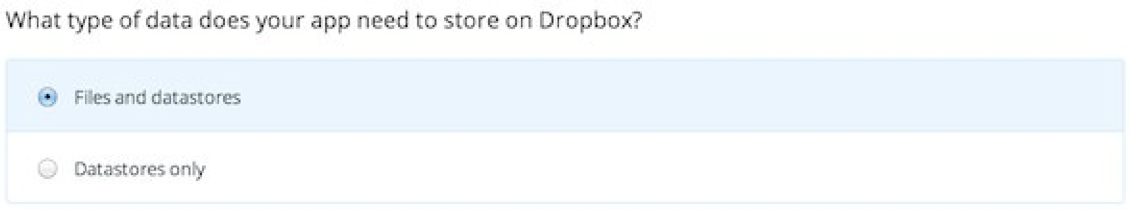
Then, click on "Create app", and choose which type of app you want to create (Dropbox API app for our project):


What type of app do you want to create?

○ Drop-ins app
Chooser or Saver

● Dropbox API app
Sync API, Datastore API, or Core API

Then, select "Files and datastores":


What type of data does your app need to store on Dropbox?

● Files and datastores

○ Datastores only

After that, you can give a name to your app and create it. Also specify that your app should only have access the files in its own folder.

What you need to get now is all the keys relative to your Dropbox app, so you can enter them later in the software part of the chapter. You will need the App Key and App Secret at this point, which are displayed on the same page as your app.

The next set of data we need to get is the Token Key & Token Secret key. To get them, the first step is to go to the InitialiseOAuth Choreo on the Temboo website:

https://temboo.com/library/Library/Dropbox/OAuth/InitializeOAuth/

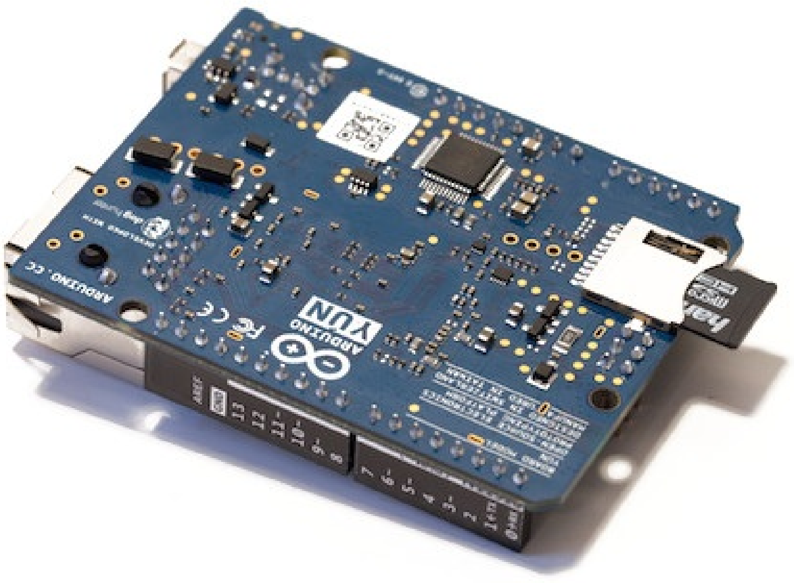
Here, you will need to enter the App Key and App Secret. That will generate some additional information, like a callback ID, and a temporary token secret. You'll also be asked to visit a link to Dropbox to confirm the authentication. Finally, go to the FinalizeOAuth page to finish the process. You will be asked to enter your App Key, App Secret, callback ID and temporary token secret:

https://temboo.com/library/Library/Dropbox/OAuth/FinalizeOAuth/

After that step, you will be given your final Token Key and Token Secret. Write them down as well, you will need them later.

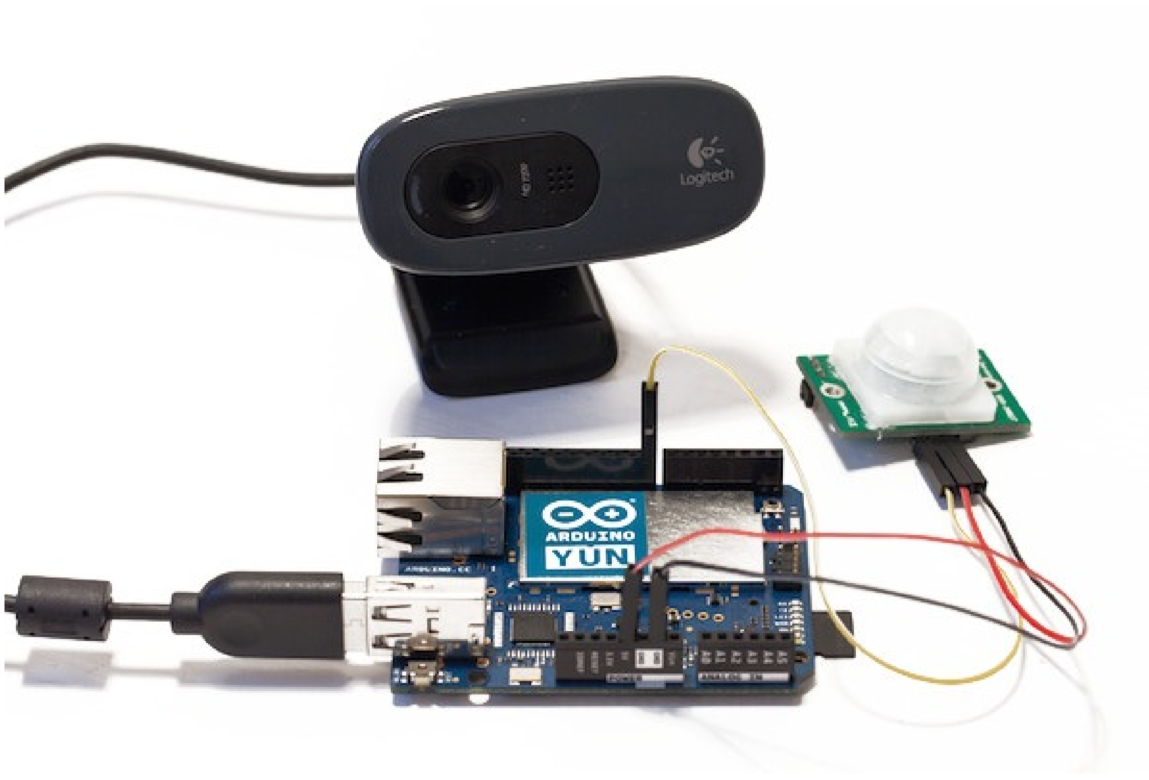
4.2 Connecting a USB Camera to the Yun

The first step is to insert the microSD card in the Arduino Yun board:



Then, connect the camera to the USB port of the Yun:

Finally, connect the motion sensor to the Yun. Simply connect the VCC pin to the Yun 5V pin, GND to GND, and the SIG pin to the Yun pin number 8:



Finally, just connect the project to your computer via the microUSB port, and you are good to go.

4.3 Testing the Camera

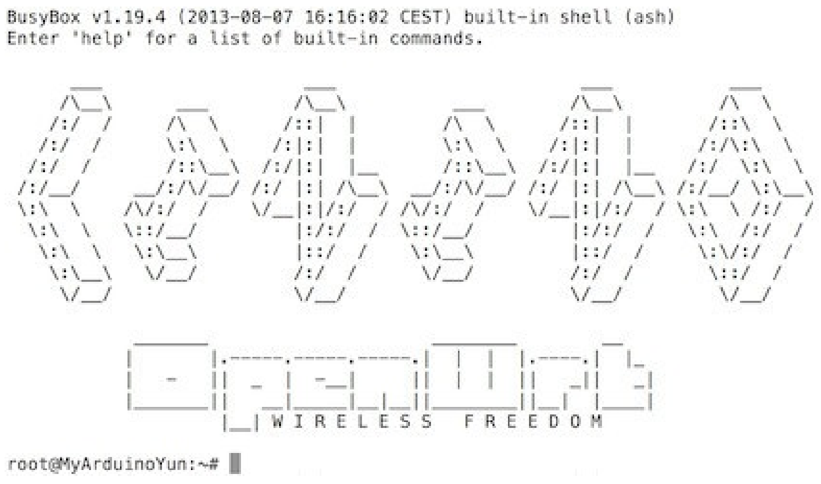
Before we can write the software for our project, we need to install some more software on your Arduino Yun. We are also going to test the USB camera to see if the drivers are working correctly. In the rest of the project, I will assume that your Yun is already setup and connected to your local WiFi network. If it is your first time using your Yun, I recommend following this guide first:

http://arduino.cc/en/Guide/ArduinoYun

First, the UVC drivers for the camera. To install them, you need to connect to your Yun via SSH. Simply open a terminal, and type:

```
ssh root@yourYunName.local
```

Where you need to insert the correct name of your Arduino Yun, that you set when you first used the Yun. You will also be prompted to enter your password. If the connection is successful, you should see some ASCII art:



```
BusyBox v1.19.4 (2013-08-07 16:16:02 CEST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M

root@MyArduinoYun:~#
```

We can now install the required packages. Start with an update of the package manager:

```
opkg update
```

Then, install the UVC drivers:

```
opkg install kmod-video-uvc
```

And the python-openssl package:

```
opkg install python-openssl
```

We also need the fswebcam utility that we will use to take pictures from the terminal:

```
opkg install fswebcam
```

We are now ready to test the webcam. Make sure that the SD card is mounted into the Yun, and go over to the SD card folder with:

```
cd /mnt/sda1
```

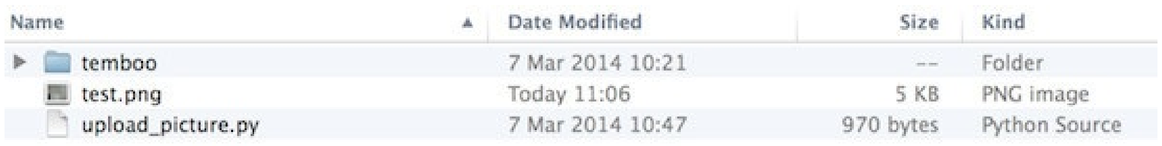
To test the camera and take a picture, it is really easy. Simply type:

```
fswebcam test.png
```

You should see some information being displayed, along with some errors, but do not worry about them. The important thing is to see these lines:

```
— Opening /dev/video0…
Trying source module v4l2…
/dev/video0 opened.
```

To check that the picture was correctly taken, remove the SD card from the Yun and read it using your computer. You should see the picture appearing at the root of the SD card:



Name	Date Modified	Size	Kind
temboo	7 Mar 2014 10:21	--	Folder
test.png	Today 11:06	5 KB	PNG image
upload_picture.py	7 Mar 2014 10:47	970 bytes	Python Source

Simply open it to make sure it was correctly taken and that it is not corrupted. If the picture looks good, you can go over to the next section and start building cool applications with the project!

In case it is not working at this point, first check that the camera is correctly connected to your Arduino Yun. Then, make sure that the camera you chose for this project is compatible with the UVC protocol. Finally, check that the SD card you are using is correctly formatted by trying to read it beforehand with your computer.

4.4 Wireless Pictures Recording

What we want to achieve in this application is to take a picture whenever some motion is detected by the PIR motion sensor. And when that happens, store this picture locally on the SD card, and upload it to Dropbox. To do so, the code will be composed of two parts.

The first one is a Python script that will connect to Dropbox, take a picture on the SD card, and then upload this picture to Dropbox. The reason to use Python for this part is that it is much easier to upload files to Dropbox using Python than directly from the Arduino sketch.

The second part of the code will be the Arduino sketch itself, which will basically call the Python script to take pictures via the Bridge library of the Yun.

Let's first code the Python script. This is the complete code for this part:

```
# Import correct libraries
import base64
import sys
from temboo.core.session import TembooSession
from temboo.Library.Dropbox.FilesAndMetadata import UploadFile

print str(sys.argv[1])

# Encode image
with open(str(sys.argv[1]), "rb") as image_file:
    encoded_string = base64.b64encode(image_file.read())

# Declare Temboo session and Choreo to upload files
session = TembooSession('yourSession', 'yourApp', 'yourKey')
uploadFileChoreo = UploadFile(session)

# Get an InputSet object for the choreo
uploadFileInputs = uploadFileChoreo.new_input_set()

# Set inputs
uploadFileInputs.set_AppSecret("yourAppSecret")
uploadFileInputs.set_AccessToken("yourAccessToken")
uploadFileInputs.set_FileName(str(sys.argv[1]))
uploadFileInputs.set_AccessTokenSecret("yourTokenSecret")
uploadFileInputs.set_AppKey("yourAppKey")
uploadFileInputs.set_FileContents(encoded_string)
uploadFileInputs.set_Root("sandbox")

# Execute choreo
uploadFileResults = uploadFileChoreo.execute_with_results(uploadFileInputs)
```

Let's now see the details of this script. It starts by including the required libraries from the Temboo Python SDK:

```
from temboo.core.session import TembooSession
from temboo.Library.Dropbox.FilesAndMetadata import UploadFile
```

The Python script will also take the name of the picture we want to upload as an argument:

```
with open(str(sys.argv[1]), "rb") as image_file:
  encoded_string = base64.b64encode(image_file.read())
```

Remember these Temboo credentials that you created earlier ? This is where you need to enter them:

```
session = TembooSession('yourTembooName', 'yourTembooApp', 'yourTembooKey')
```

We can then create the correct Dropbox library to upload files, called a “Choreo” on Temboo:

```
uploadFileChoreo = UploadFile(session)
uploadFileInputs = uploadFileChoreo.new_input_set()
```

It is now the time to enter all the informations about your Dropbox account, like your app Key, app Secret, Access Token an Access Token Secret:

```
uploadFileInputs.set_AppSecret("appSecret")
uploadFileInputs.set_AccessToken("accessToken")
uploadFileInputs.set_FileName(str(sys.argv[1]))
uploadFileInputs.set_AccessTokenSecret("accessTokenSecret")
uploadFileInputs.set_AppKey("appKey")
uploadFileInputs.set_FileContents(encoded_string)
uploadFileInputs.set_Root("sandbox")
```

And finally, upload the file on Dropbox:

```
uploadFileResults = uploadFileChoreo.execute_with_results(uploadFileInputs)
```

You can now save this code in a file named upload_picture.py. Note that all the files are available inside the GitHub repository of the book:

https://github.com/openhomeautomation/iot-book

We will now work on the Arduino sketch. This is the complete code for this part:

```
// Sketch to upload pictures to Dropbox when motion is detected
#include <Bridge.h>
#include <Process.h>

// Picture process
Process picture;

// Filename
String filename;

// Pin
int pir_pin = 8;

// Path
String path = "/mnt/sda1/";

void setup() {

  // Bridge
  Bridge.begin();

  // Set pin mode
  pinMode(pir_pin,INPUT);
}
```

```
void loop(void)
{

  if (digitalRead(pir_pin) == true) {

    // Generate filename with timestamp
    filename = "";
    picture.runShellCommand("date +%s");
    while(picture.running());

    while (picture.available()>0) {
      char c = picture.read();
      filename += c;
    }
    filename.trim();
    filename += ".png";

    // Take picture
    picture.runShellCommand("fswebcam " + path + filename +
      " -r 1280x720");
    while(picture.running());

    // Upload to Dropbox
    picture.runShellCommand("python " + path + "upload_picture.py " +
      path + filename);
    while(picture.running());
  }
}
```

The sketch starts by including the required libraries:

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

We also have to declare a process, that we are going to use to call functions on the Linux machine of the Yun (for example the fswebcam utility we used before):

```
Process picture;
```

We are also going to build a filename for each picture the project will take, that will be stored in a string:

```
String filename;
```

We also declare the pin on which the PIR motion sensor is connected:

```
int pir_pin = 8;
```

And the path of the SD card on the Yun:

```
String path = "/mnt/sda1/";
```

Because we need to call functions on the Linux machine of the Yun, we have to start the Bridge:

```
Bridge.begin();
```

Then, in the loop() part of the sketch, we check if some motion was detected by the PIR sensor:

```
if (digitalRead(pir_pin) == true) {
```

If this is the case, we build a unique filename for the picture, with the date at which the picture was taken:

```
filename = "";
picture.runShellCommand("date +%s");
while(picture.running());

while (picture.available()>0) {
  char c = picture.read();
  filename += c;
}
filename.trim();
filename += ".png";
```

We then make the first call to the Linux machine of the Yun, first to take a picture with the fswebcam utility. Note that here, we provide an extra argument with the -r command, which set the resolution. I used the maximum resolution of my camera, which is 720p:

```
picture.runShellCommand("fswebcam " + path + filename + " -r 1280x720");
while(picture.running());
```

We then make a second call to the Linux machine, this time calling the Python script with the name of the picture as an argument, which will upload the picture to Dropbox:

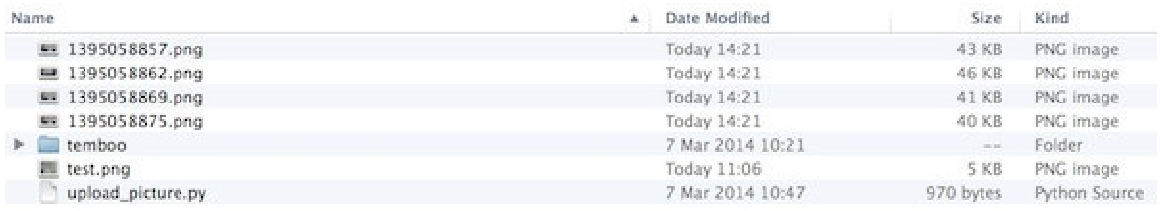
```
picture.runShellCommand("python " + path + "upload_picture.py "
+ path + filename);
while(picture.running());
```

You are now ready to test the project. Again, all the files are available inside the GitHub repository of the book:

https://github.com/openhomeautomation/iot-book

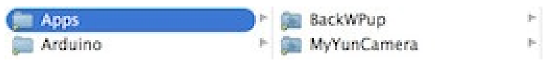
First, put the Python file at the root of the SD card, and put the SD card back into the Arduino Yun board. Then, upload the Arduino sketch to the Yun. Now, try to trigger the motion sensor, for example by waiving your hand in front of it. You should see that the webcam is being activated shortly after (for example, my webcam has a LED that turns green when it is active).

To check that the project is working correctly, after a while you can check the SD card, you should see that some pictures have been recorded:



Name	Date Modified	Size	Kind
1395058857.png	Today 14:21	43 KB	PNG image
1395058862.png	Today 14:21	46 KB	PNG image
1395058869.png	Today 14:21	41 KB	PNG image
1395058875.png	Today 14:21	40 KB	PNG image
temboo	7 Mar 2014 10:21	--	Folder
test.png	Today 11:06	5 KB	PNG image
upload_picture.py	7 Mar 2014 10:47	970 bytes	Python Source

You can also check on your Dropbox folder, where the same pictures should have been uploaded. They should be located in your Dropbox apps folder:



In case it is not working at this point, there are several things you can check. First, check that your Arduino Yun is correctly configured, and that it can access the Internet. After that, make sure that your correctly configured your app on Dropbox, and that you got the required API keys for your Dropbox account. Finally, make sure that you correctly enterer your Temboo & Dropbox credentials inside the Arduino files.

4.5 How to Go Further

In this chapter, you learned how to connect a USB camera to your Arduino Yun and build an exciting project on top of it. We built a security camera that automatically uploads pictures on Dropbox when motion is detected by a motion sensor.

Of course, there are several ways to build other cool applications using this project. You can for example drop the motion detection part, and build a camera that take snapshots at regular intervals and upload these on Dropbox. You can for example easily create time-lapse videos with this kind of project: just collect the pictures from your Dropbox account, paste them into a time-lapse software, and done! You can also extend this project by adding more Yun and camera modules, to build a complete monitoring system for your home.

Chapter 5

Conclusion

5.1 What did you Learn in This Book?

Let's first summarize what you learned in this book. With the first chapter, you learned the basics of the Internet of Things. You built projects that performed local measurements and sent these measurements to the cloud so they are stored online. We also saw how to monitor these measurements online, from anywhere in the world.

Then, we saw another major pillar of the Internet of Things: having web-connected objects that are accessible from anywhere in the world. We built a lamp that was controllable from any web browser.

After that, we took things to another level by introducing a more powerful Arduino board: the Arduino Yun. First, we connected this board to the web service Temboo, to automatically log data in a Google Docs spreadsheet.

Finally, we connected a USB camera to create our own wireless IP security camera. The camera was configured to automatically take pictures when motion was detected. When that was the case, the project automatically uploaded these pictures on Dropbox.

5.2 How to go Further

The purpose of this book was really to teach you the basics about how to build Internet of Things applications with Arduino. There are many things you can do to extend your skills on the topic. The first thing you can do is to re-do again all the examples found in this book, to really get a good knowledge about how to connect your Arduino devices to the Internet.

You can also mix the different projects that you did in this book to create more exciting applications. For example, you can use the Arduino Yun and a camera to send pictures to other services than Dropbox. You can also use a central online dashboard to display pictures took by your wireless camera. Finally, you can use all these projects together to create a whole home automation system purely based on cloud services.

Chapter 6

Resources

The following is a list of the best resources concerning the Internet of Things with the Arduino platform. I organized this chapter in different categories so you can find the all information you need in a timely manner.

6.1 General Information on Arduino

- Arduino: the reference website on the Arduino platform. Especially go over to their fantastic forums to find help on your Arduino related projects.
- Instructables: A website containing step-by-step projects. Search there for “Arduino” or “Internet of Things” and you will find a lot of exciting projects.
- Adafruit Learning System: An online learning platform with a selection of high-quality step-by-step articles on making things in general. Many projects use the Arduino platform, and some are about the Internet of Things.

6.2 Internet of Things Platforms

- Dweet.io: A online service to easily store data coming from your Arduino projects.
- Freeboard: An online dashboard to quickly visualize the current state of all your Things.
- Temboo: The Temboo website allows you to quickly interface your Arduino projects with other online services like Google Docs or Dropbox.
- The Thing System: An open-source framework for the Internet of Things.

6.3 Components

- SparkFun: A website selling many Arduino related products. All their products are open-source and you can download the source files directly from their product descriptions.
- Adafruit: A company based in New York that sells high quality products for the Arduino platform.
- SeeedStudio: A Chinese company that sells many original products for the Arduino platform. They also offer their own PCB production & assembly services.

6.4 Suggested Reading

- Internet of Things with the Arduino Yun: My other book on the topic of the Internet of Things. Published by PacktPub, the book contains for exciting projects using the Arduino Yun. Compared to the book you are currently reading, this other book explores topics like energy monitoring and robotics.
- Building Internet of Things with the Arduino: A very good introduction to the Internet of Things with the Arduino platform. The author explores all the aspect of the Internet of Things, including how to interact remotely with your Arduino projects using mobile devices.

Table of Contents