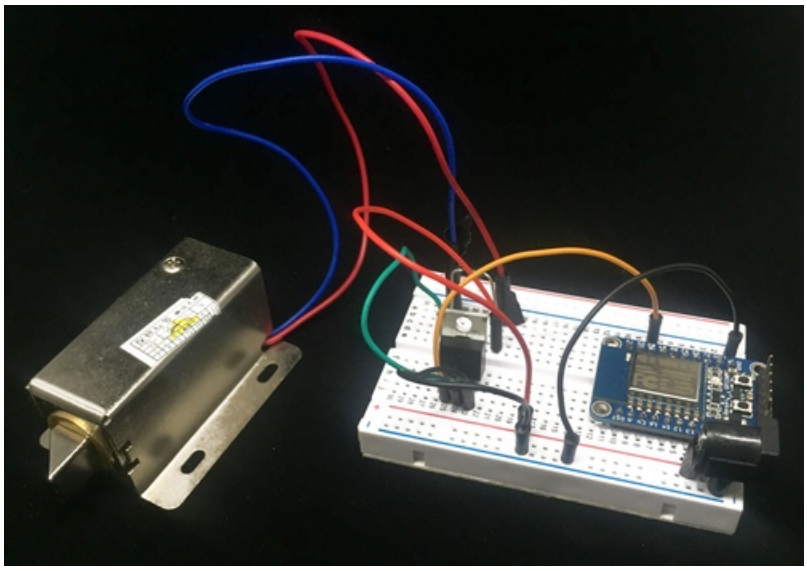
adafruit learning system

Remote controlled door lock using a fingerprint sensor & Adafruit IO

Created by Marc-Olivier Schwartz



Last updated on 2018-08-22 03:50:06 PM UTC

Guide Contents

Introduction

Smart locks are now widely available, and allow users to open & close a door by the press of a button on their smartphone. Some locks are even connected to the cloud, and allows the user to open/close it remotely, and share the access to the door with other people.

In this project, we are going to build our own DIY version of such a smart lock, that can be controlled from anywhere in the world, as we'll connect it to Adafruit IO.

However, we'll add a twist in the project: we will actually also allow the user to control the door lock by using their fingerprint. And as the fingerprint sensor will also be connected to the Adafruit IO platform, it will be really easy to make both devices talk to each other. Let's dive in!

Setting up the Fingerprint Sensor

We are first going to set up the fingerprint sensor device. As the library for the fingerprint sensor was designed for Arduino, we are going to use an Arduino Uno for this part of the project. For web connectivity, we are going to use an Adafruit CC3000 breakout board.

Let's start by connecting the power supply:

- Connect the 5V pin from the Arduino board to the red power rail
- The GND from Arduino the blue power rail on the breadboard.

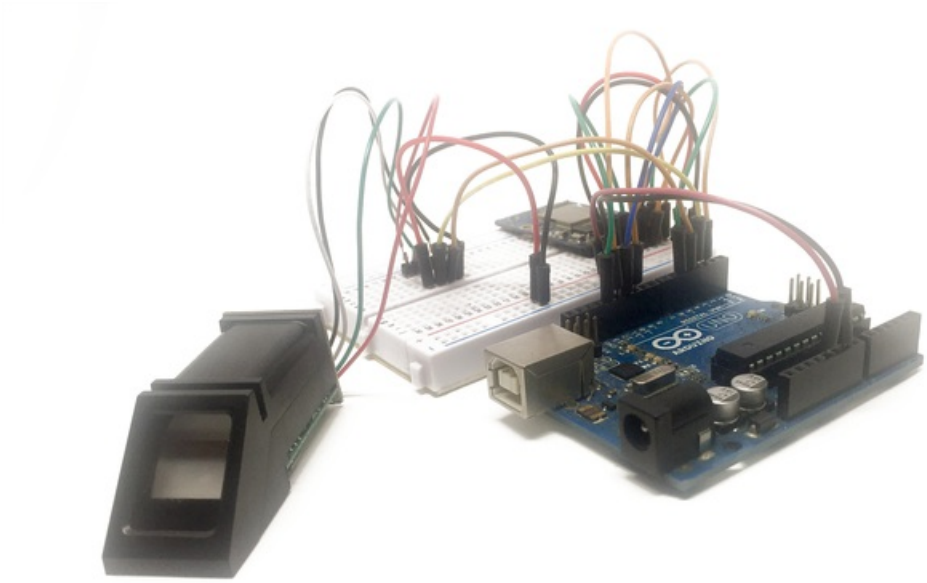
Now, let's connect the fingerprint sensor.

- First, connect the power, by connecting the cables to their respective color on the breadboard: **Red** to **+5V** rail, **Black** to **ground** rail
- Then, connect the **white wire** from the sensor to Arduino pin **4**
- & the **green wire** to pin number **3**.

Now, the CC3000 module.

- First, connect the **IRQ** pin of the CC3000 board to pin number **2** of the Arduino board
- **VBAT** to pin **5**
- **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).

This is the completely assembled project:



Before building the sketch that will actually upload data to the Adafruit IO platform, we need to go through the process of enrolling your fingerprint into the sensor itself, so it can be recognized later. I recommend doing that with the Arduino board itself, and you can find the whole procedure at:

https://learn.adafruit.com/adafruit-optical-fingerprint-sensor/enrolling-with-arduino (https://adafru.it/Cg8)

At this point, also create an account on Adafruit IO if that's not done yet:

https://io.adafruit.com (https://adafru.it/eZ8)

Once that's done, you can move on to the next step: building the sketch that will send data to Adafruit IO. As the sketch is really long, I will only highlight the most important parts here, and link later to the GitHub repository of the project.

It starts by including all the required libraries:

```
#include <Adafruit_SleepyDog.h>
#include <Adafruit_CC3000.h>
#include <SPI.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_CC3000.h"
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
```

Then, you'll need to modify the sketch by inserting your WiFi network SSID & password:

```
#define WLAN_SSID       "your_wifi_ssid"
#define WLAN_PASS       "your_wifi_password"
#define WLAN_SECURITY   WLAN_SEC_WPA2
```

After that, you also need to enter your Adafruit IO username & AIO key:

```
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "adafruit_io_username"
#define AIO_KEY         "adafruit_io_key"
```

Then, we also define a feed specific for the fingerprint sensor. It will simply contain '1' if the sensor has just been activated with a valid fingerprint:

```
const char FINGERPRINT_FEED[] PROGMEM = AIO_USERNAME "/feeds/fingerprint";
Adafruit_MQTT_Publish fingerprint = Adafruit_MQTT_Publish(&mqtt, FINGERPRINT_FEED);
```

We also need to create as SoftwareSerial instance, for the fingerprint sensor:

```
SoftwareSerial mySerial(3, 4);
```

After that, we can actually create the instance for the sensor:

```
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
```

Inside the sketch, we specify which fingerID should actually activate the lock later on. I used 0, which is the ID of the

first fingerprint I enrolled into the sensor:

```
int fingerID = 0;
```

Then, we'll define a counter & a delay for the project. Basically, we want the lock to automatically close again after it's been opened. I used 10 seconds here as an example, but you can of course modify this delay at your convenience:

```
int activationCounter = 0;
int lastActivation = 0;
int activationTime = 10 * 1000;
```

Then, in the setup() function of the sketch, we simply initialise the sensor, and also connect the CC3000 chip to your WiFi network.

In the loop() function of the sketch, we then connect to Adafruit IO using:

```
MQTT_connect();
```

Once we are sure to be connected to the Adafruit IO platform, we check for the last recognised fingerprint. If it matches, and the lock is currently not activated, we then send a '1' message to the feed on Adafruit IO:

```
if (fingerprintID == fingerID && lockState == false) {
  Serial.println(F("Access granted!"));
  lockState = true;
  state = 1;
  if (! fingerprint.publish(state)) {
    Serial.println(F("Failed"));
  } else {
    Serial.println(F("OK!"));
  }
  lastActivation = millis();
}
```

Still in the loop() function of the sketch, if the lock is currently active, and the delay we defined earlier has been reached, we send a '0' message to Adafruit IO:

```
if ((activationCounter - lastActivation > activationTime) && lockState == true) {

  lockState = false;
  state = 0;
  if (! fingerprint.publish(state)) {
    Serial.println(F("Failed"));
  } else {
    Serial.println(F("OK!"));
  }
}
```

Note that you can find the latest version of the code on the GitHub repository of the project:

https://github.com/openhomeautomation/lock-control-fingerprint (https://adafru.it/ije)

It's now time to test the project! Use the Arduino library manager to download all the required libraries of this project.

Make sure that you modified the code with your settings, and then upload it to the Arduino board. Also open the Serial monitor.

Then, once the Arduino board is connected to the WiFi network, you should see that the fingerprint sensor is blinking with red light. Place the finger that you enrolled earlier on the sensor. You should see on the Serial monitor the ID number of the recognised finger, and if it matches, you should also see an 'OK!' message meaning that data has been send to Adafruit IO.

You can then also check on Adafruit IO, in your feeds, to make sure that the feed has been updated with the correct data. Then, after the delay defined in the code, this feed should automatically return to '0'.

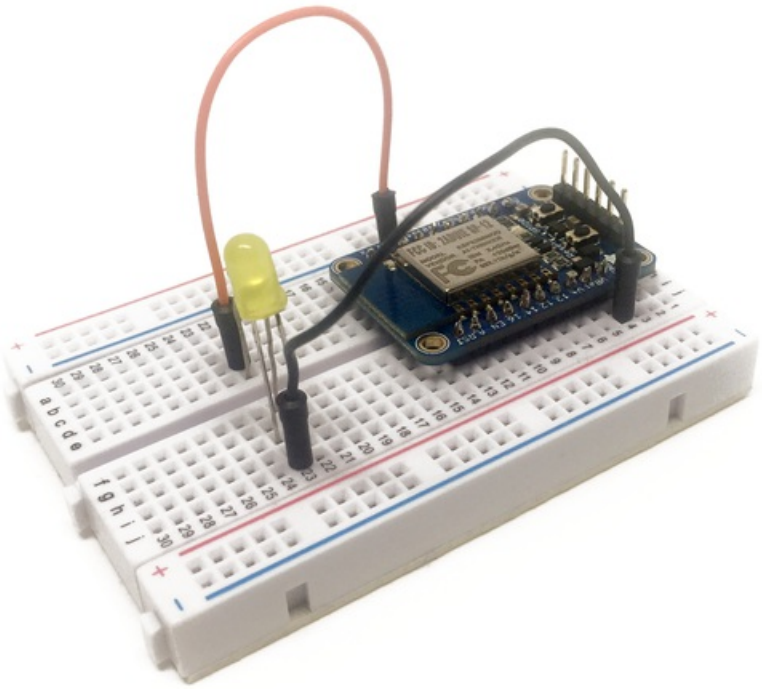
Setting Up the Dummy Door Lock

We are now going to set up the part of the project that will actually control the door lock. We'll use the Adafruit ESP8266 breakout board here, as that's just what we need to connect to the WiFi network and activate/deactivate the lock. It will also illustrate how simple it is to make two different platforms (Arduino & the ESP8266) communicate with each other using Adafruit IO.

In this part, we are actually not going to use the lock right away, as the hardware is quite complicated to assembled. Instead, we are simply going to connect an LED to the pin where we'll put the lock later. This will allow us to test the code without having to care about the complexity of the hardware.

To assemble this part of the project, it's really simple: just put the ESP8266 on the breadboard first. Then, put the LED on the breadboard, with the longest side of the LED in series with the resistor. After that, connect the other side of the resistor the pin 5 of the ESP8266 board, and the other side of the LED to the GND pin of the ESP8266 board.

This is the completely assembled project:



Let's now see the sketch that we'll use for this part of the project. Again, as the sketch is quite complex, I'll only show you the most important parts here.

It starts by including the right libraries:

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
```

Then, you need to set your WiFi settings:

```
#define WLAN_SSID       "your_wifi_ssid"
#define WLAN_PASS       "your_wifi_password"
#define WLAN_SECURITY   WLAN_SEC_WPA2
```

And also your Adafruit IO settings, just as in the previous project:

```
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "adafruit_io_username"
#define AIO_KEY         "adafruit_io_key"
```

We also define on which pin we'll connect the LED (and later the lock, or a relay for example):

```
int relayPin = 5;
```

We also define a new feed to the lock:

```
const char LOCK_FEED[] PROGMEM = AIO_USERNAME "/feeds/lock";
Adafruit_MQTT_Subscribe lock = Adafruit_MQTT_Subscribe(&mqtt, LOCK_FEED);
```

In the setup() function of the sketch, we set the pin of the LED as an output:

```
pinMode(relayPin, OUTPUT);
```

We also subscribe to the lock feed that we just defined earlier:

```
mqtt.subscribe(&lock);
```

In the loop() function of the sketch, we first check if we are connected to Adafruit IO:

```
MQTT_connect();
```

Then, we check what's coming from the lock feed we subscribed to. If the message is '1', we activate the pin we defined earlier, that is connected to the LED. If we receive a '0', we just put this pin to a low state again:

```
Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(1000))) {
  if (subscription == &lock) {
    Serial.print(F("Got: "));
    Serial.println((char *)lock.lastread);

    // Save command to String
    String command = String((char *)lock.lastread);
    command.trim();

    if (command == "0") {
      digitalWrite(relayPin, LOW);
    }
    if (command == "1") {
      digitalWrite(relayPin, HIGH);
    }
  }
}
```

Note that you can find the latest version of the code on the GitHub repository of the project:

https://github.com/openhomeautomation/lock-control-fingerprint (https://adafru.it/ije)

It's now time to test the project! Use the Arduino library manager to download all the required libraries of this project.

Also make sure that you modified the code with your settings, like your Adafruit IO settings & your WiFi network credentials.

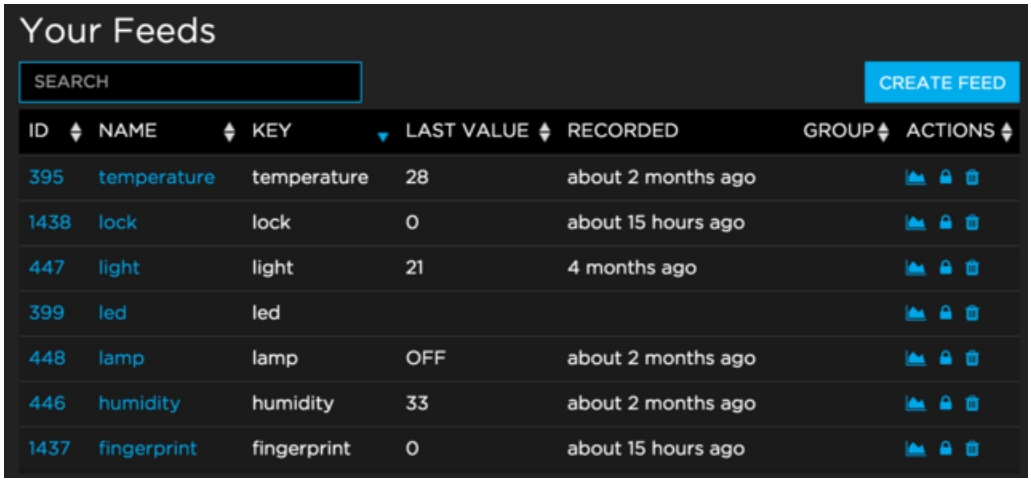
To program the ESP8266 chip, I used a simple USB-FTDI board. To learn more about how to program the Adafruit ESP8266 breakout board, I recommend this excellent guide:

https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout (https://adafru.it/irC)

Finally, upload the code to the board. Also open the Serial monitor. You basically just want to check for now that the project can indeed connect to Adafruit IO: we'll test the other functionalities in the next page.

Testing the Project

It's now time to finally test our project! First, go to your Adafruit IO dashboard, in the Feeds section. You need to check that the fingerprint & lock feeds have been created by your sketches:

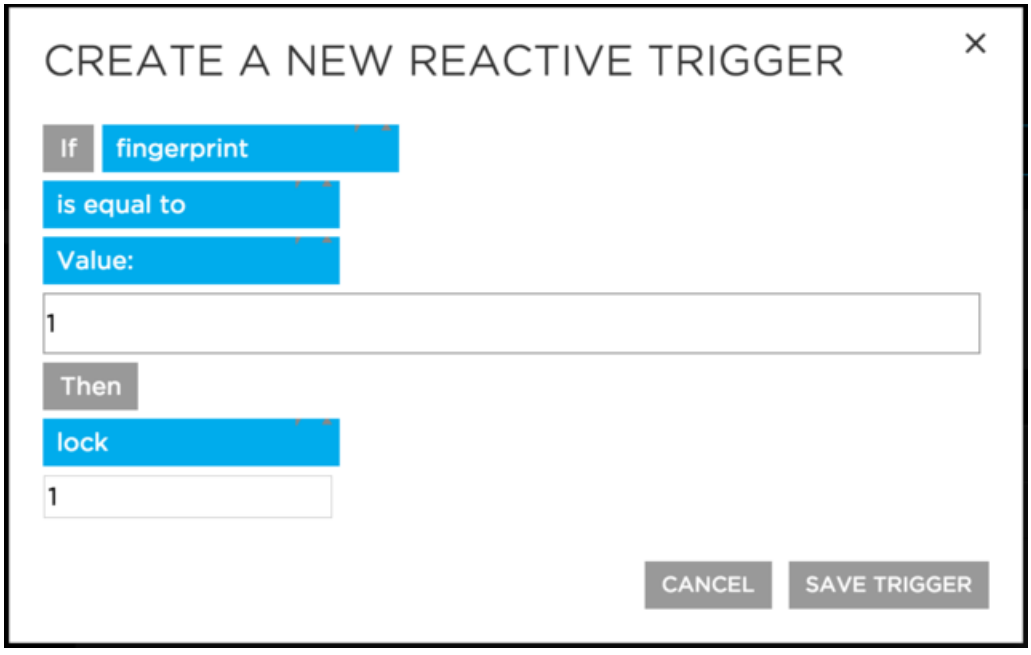


If that's not the case, you'll need to create them manually.

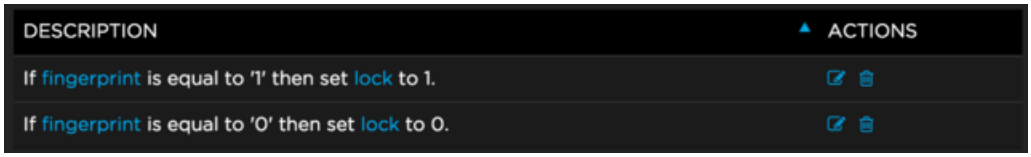
Then, we still need a way to link the fingerprint feed, and the lock feed. We want the lock feed to take the '1' value when the fingerprint feed goes to 1, and vice-versa.

For that, we'll use a very powerful feature of Adafruit IO: triggers. Triggers are basically conditions that you can set on your feeds, for example to link two feeds.

We'll define a new reactivr trigger from the Triggers section of Adafruit IO, linking our fingerprint & lock feeds:



Here is how it should look like with both triggers in the Adafruit IO dashboard:



Once that's done, you can actually test the project! Make sure that both parts of the project are working correctly. Then, put the finger that you enrolled before on the fingerprint sensor.

You should see a little light coming up on the Arduino board, meaning that data was transmitted to Adafruit IO. Then, very shortly after, you should see blinking light on the ESP8266 board, meaning data was received via MQTT. The LED should also light up immediately.

After the delay you set in the code (10 seconds by default), you should see that the LED will go off again. Congratulations, you are now able to control the LED with your fingerprint, even if both projects were on the other side of the world!

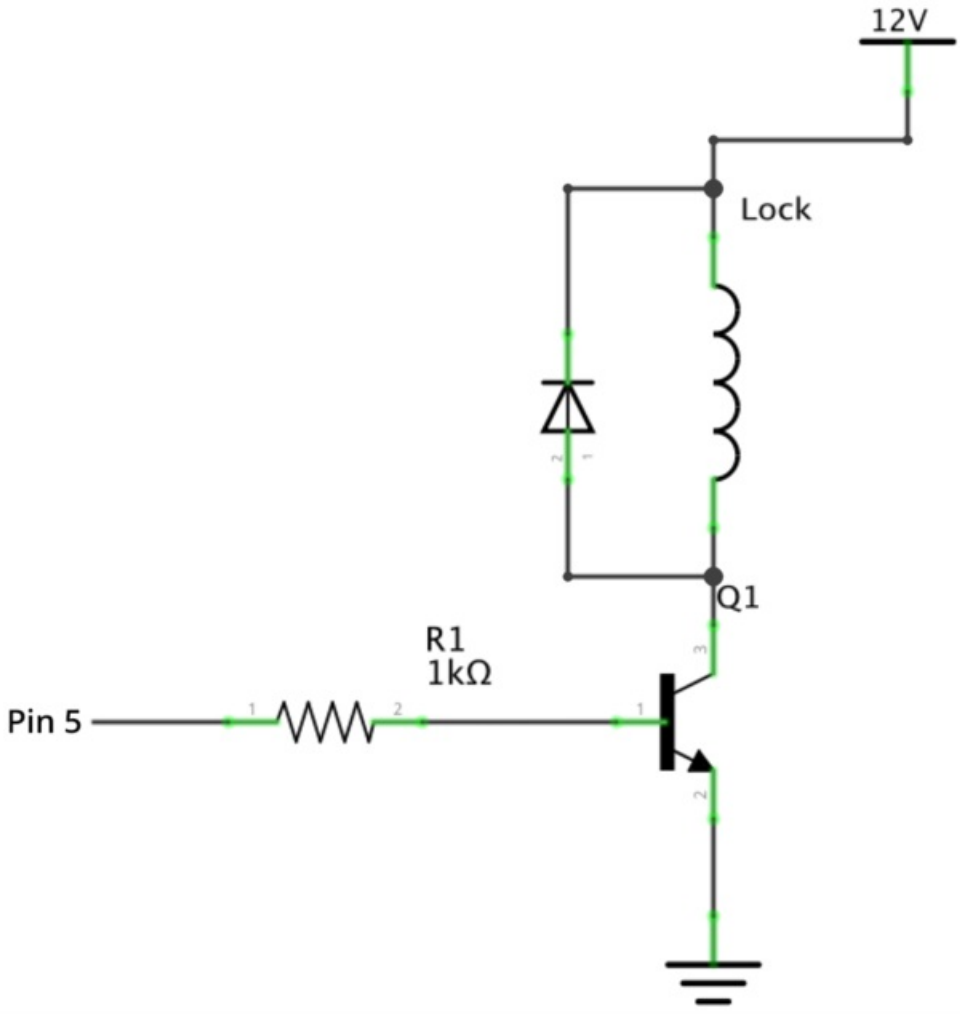
Setting Up the Electronic Door Lock

It's now time for the last part of the project: actually connecting the electronic lock. This is quite complex, and you could perfectly use the project as it is, for example by connecting a relay instead of the LED.

To actually connect the door lock, you'll need a bunch of additional components. You will first need a 12V power supply, and a breadboard-friendly DC barrel jack.

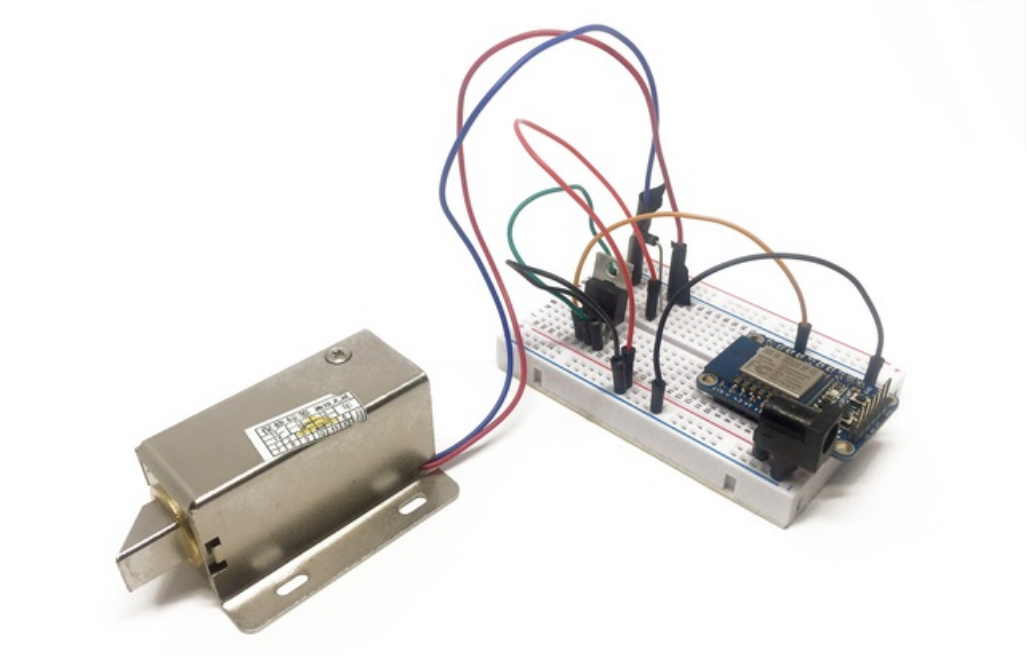
You will also need a power transistor (I used an IRLB8721 power MOSFET (https://adafru.it/ijf), but you can also use a TIP102 bipolar transistor for example.) You'll also need a 1K Ohm resistor in case you are using a bipolar transistor. Finally, you will need a rectifier diode. (https://adafru.it/ijA)

For the assembly of all these components to the ESP8266 board, it's better to look at this schematic:



Note that if you are using a MOSFET transistor, you don't need the resistor between the pin number 5 of the ESP8266 board & the transistor.

This is the completely assembled project:



When the project is assembled, power up again the ESP8266 using the FTDI module for example, and then connect the 12V DC power supply. You don't need to change anything in the code: it should work out of the box, as it's using the same pin as before.

Now, try again placing your finger on the fingerprint sensor: you should see the lock reacts nearly immediately.

The following video shows the project in action:

How to Go Further

In this project, we built a remotely controled door lock, that can be controled using your fingerprint. Thanks to Adafruit IO, it was also really easy to connect the two parts of the project wirelessly.

You can now experiment with this project, and of course modify it at your convenience. You can for example replace the door lock by anything you want: a simple relay, or a PowerSwitch Tail Kit to control an electrical appliance.

You could also imagine controlling several devices at once in your home just by placing your finger on the sensor once. For example, you could have all the doors in your home connected to such a project, and locking them all at once.