# Adafruit IO Home: Security

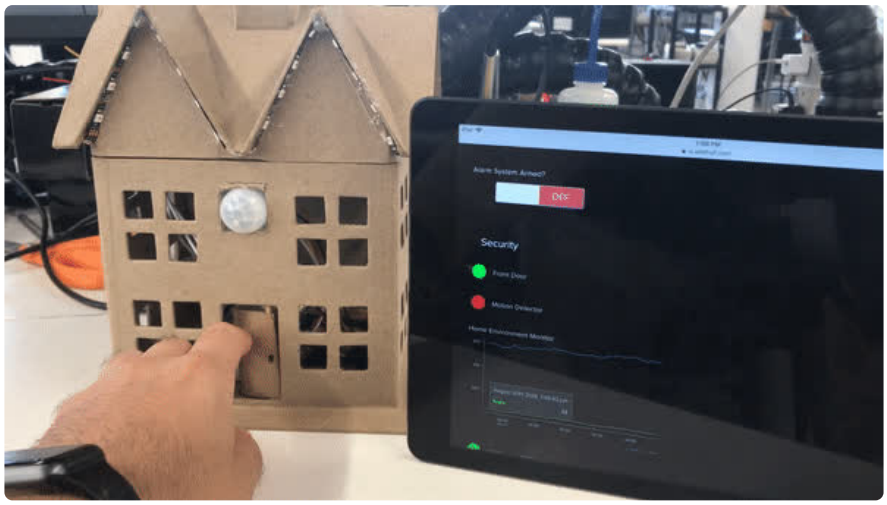
Created by Brent Rubell


https://learn.adafruit.com/adafruit-io-home-security

Last updated on 2024-03-08 03:08:35 PM EST

# Table of Contents

# Overview




## Home Security

A smart home is a safe home. In this guide, we're going to take our cardboard smart-home and equip it with sensors to protect the home against intruders, wanna-be burglars, and invisible chemicals.

### What is the IO Home?

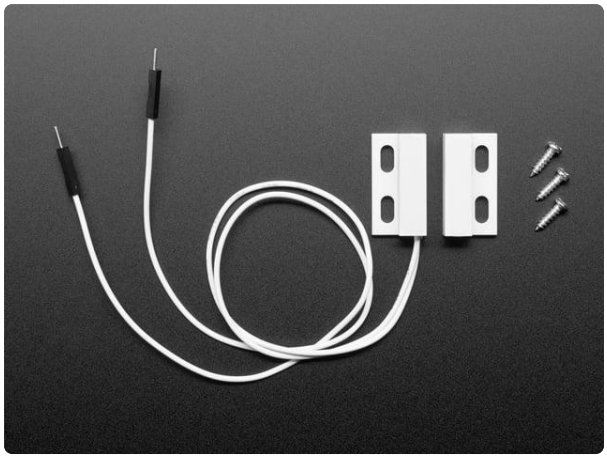
Interested in making your house a bit smarter?  Why not start small by building a **Cardboard Smart Home**! Adafruit IO Home is a series of learning guides covering all aspects of a smart house: from temperature monitoring to an intelligent home security system.

Want to scale up from the cardboard home to a real home?  We've selected real-world components, sensors, and hardware which can also be installed in your home, office or laboratory!

# Parts

### Door Sensor

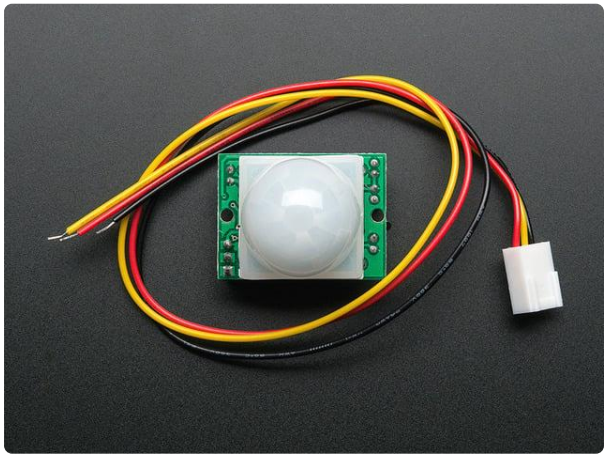
We'll be building a small front door (out of cardboard, of course) for our home. We'll use a reed switch and magnet to detect if the door is open or closed. We'll be adding a toggle-block on our Adafruit IO dashboard to arm/dis-arm the system.



Magnetic contact switch (door sensor)
This sensor is essentially a reed switch, encased in an ABS plastic shell. Normally the reed is 'open' (no connection between the two wires). The other half is a magnet. When...
https://www.adafruit.com/product/375

## Motion Sensor

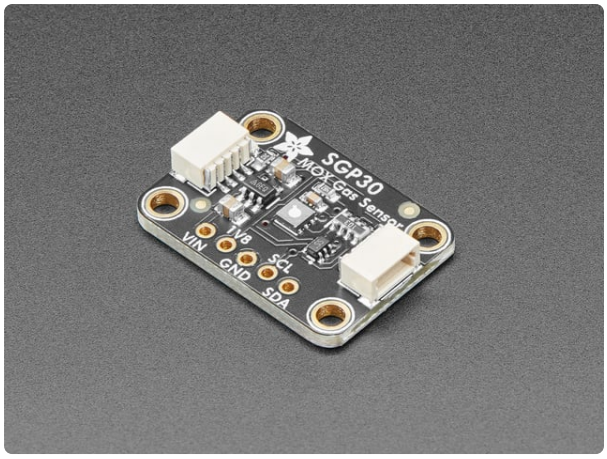
We'll also utilize a PIR sensor to detect any intruders who hang out in front of our house after dark.



PIR (motion) sensor
PIR sensors are used to detect motion from pets/humanoids from about 20 feet away (possibly works on zombies, not guaranteed). This one has an adjustable delay before firing (approx...
https://www.adafruit.com/product/189

## Air Quality Sensor

Now, what about the thing most people don't think to monitor? We're talking about invisible intruders - gasses and air particulates. While gasses and volatile organic compounds are already inside your home - you'd benefit from adding the ability to measure them. The SGP30 Air Quality Sensor is one of our favorite sensors - it can monitor both the amount of Carbon Dioxide (https://adafru.it/CpD) and Volatile Organic Compounds (https://adafru.it/CpE) in your home's air.




Adafruit SGP30 Air Quality Sensor Breakout - VOC and eCO2

Breathe easy with the SGP30 Multi-Pixel Gas Sensor, a fully integrated MOX gas sensor. This is a very fine air quality sensor from the sensor experts...

https://www.adafruit.com/product/3709

## Alarm System

A piezo buzzer and a NeoPixel Jewel are used in this guide as a combination of visual and audio alerts to notify you when your home has detected an after-dark intruder.

**1 x** Piezo Buzzer — https://www.adafruit.com/product/160

Small, but mighty, buzzer.

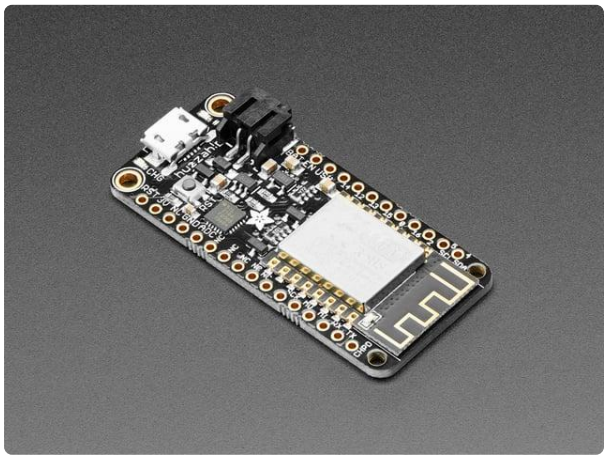
---

**1 x** NeoPixel Jewel — https://www.adafruit.com/product/2226

NeoPixel Jewel - 7 x 5050 RGB LED with Integrated Drivers

---

To use this guide, we'll need a microcontroller capable of connecting to Adafruit IO. The Feather HUZZAH with ESP8266 is used in most of our Adafruit IO Guides.



Adafruit Feather HUZZAH with ESP8266 - Loose Headers

Feather is the new development board from Adafruit, and like its namesake, it is thin, light, and lets you fly! We designed Feather to be a new standard for portable microcontroller...

https://www.adafruit.com/product/2821

## Materials

You'll need these extra materials to assemble the home security system.

Half-Size Breadboard

**1 x** Breadboard https://www.adafruit.com/product/64

Half-Size Breadboard

---

**1 x** Breadboarding Wire Bundle https://www.adafruit.com/product/153

You'll want some of these wires to hook up your sensors.

---

**1 x** Solder https://www.adafruit.com/product/145

Mini Spool, 100g of 60/40 rosin-core solder.

---

**1 x** Right Angle USB Cable https://www.adafruit.com/product/1318

USB Cable which bends at a right angle, can fit easily inside the house.

---

## Tools

Owning, or having access to, these tools will make this guide much easier to follow. If you do not have access to them, pick some up from the Adafruit Store.

**1 x** Ruler https://www.adafruit.com/product/1554

Our snazzy PCB reference ruler. You don't need THIS ruler, but you do need a ruler.

---

**1 x** Helping Hands https://www.adafruit.com/product/291

Helping Third Hand Magnifier W/Magnifying Glass Tool

---

**1 x** Soldering Station https://www.adafruit.com/product/1204

Hakko FX-888D

---

- X-Acto Knife, Box Cutter, or Sharp Knife
- Hot Glue Gun
- Double Sided Tape

---

# Putting it Together

If you've been following along with the Adafruit IO Home Series, you should have already built and wired the house's lighting system.

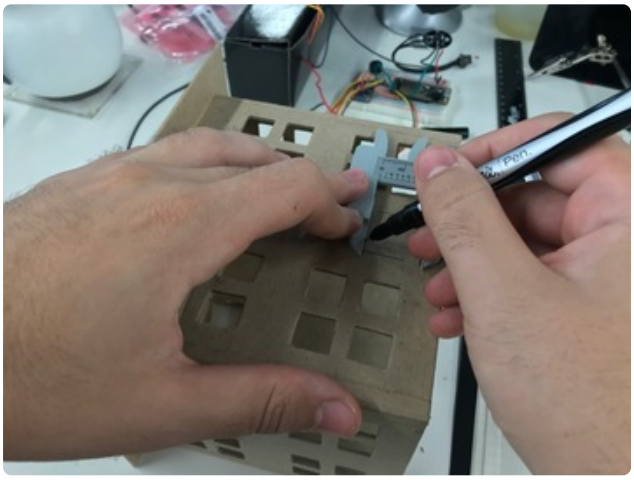
- If you need to assemble your house, click here to read the previous guide's Putting it Together page. (https://adafru.it/Co5)

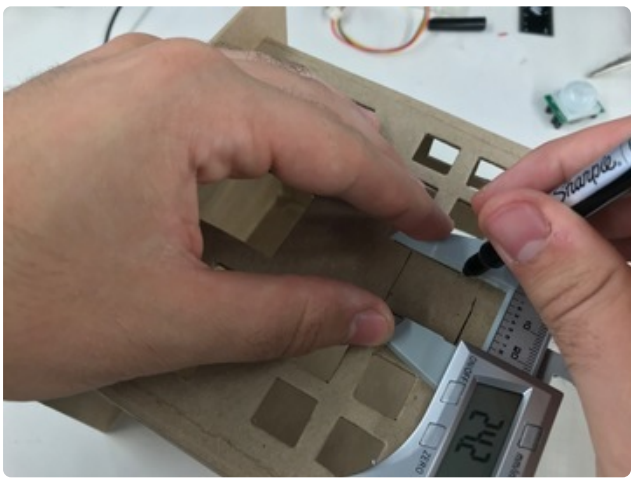
# Adding a Motion Sensor (PIR)


A PIR sensor will be used to detect motion in front of the home. PIR sensors can detect motion from about twenty feet away.

Interested in learning more about PIR Sensors? We have a full tutorial with information about these Pyroelectric InfraRed Sensors. (https://adafru.it/CeB)

We're going to mount it on the front-face of the home. The PIR sensor used for this guide has a lens with a square base of 23mm x 23mm.

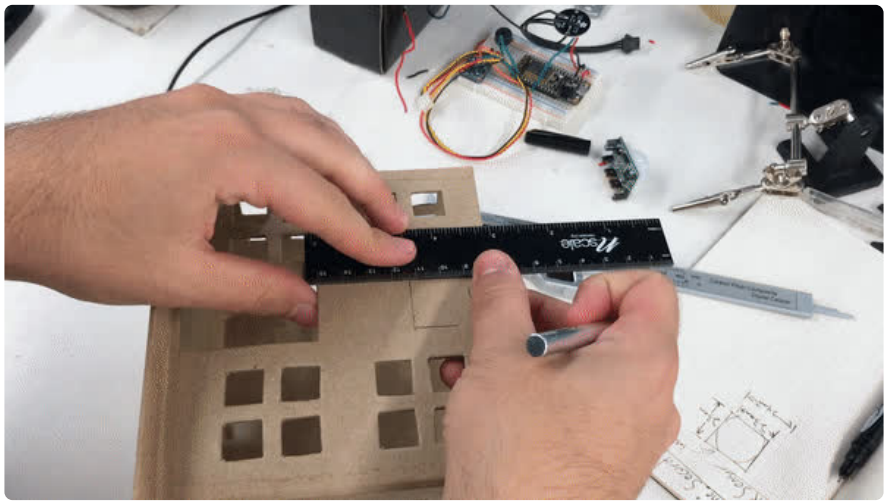


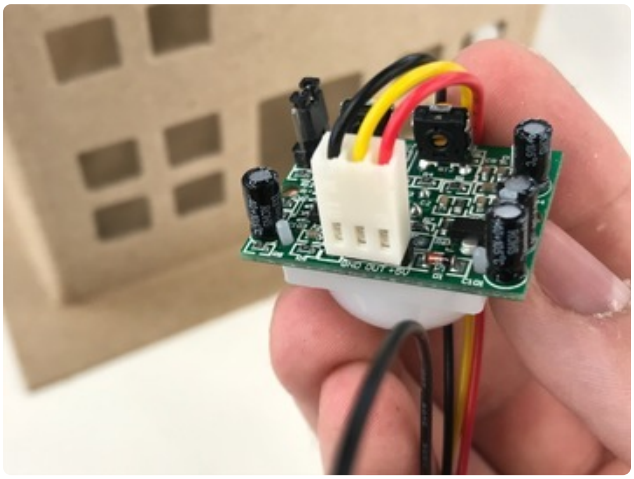


Measure and mark a 24mm x 24mm square on the front face of the home. We'll leave a little bit of room to push the sensor's lens through the home.

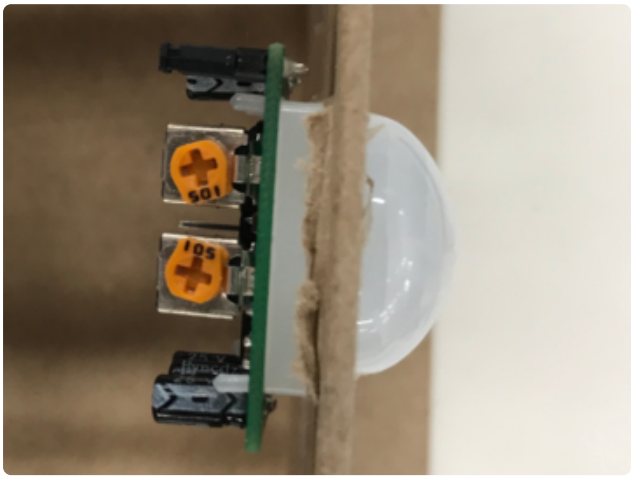
Using an X-Acto knife or a box cutter, make small scores on each side of the rectangle, one-by-one, until you have a rectangular hole.

Test-fit the sensor through the hole. The lens should poke through.





Before pushing the PIR sensor through the hole we created, ensure the 3-pin cable is properly connected to the ground, signal, and power pins.



Then, push it through the hole we cut in the cardboard.

## Adding the Door Sensor

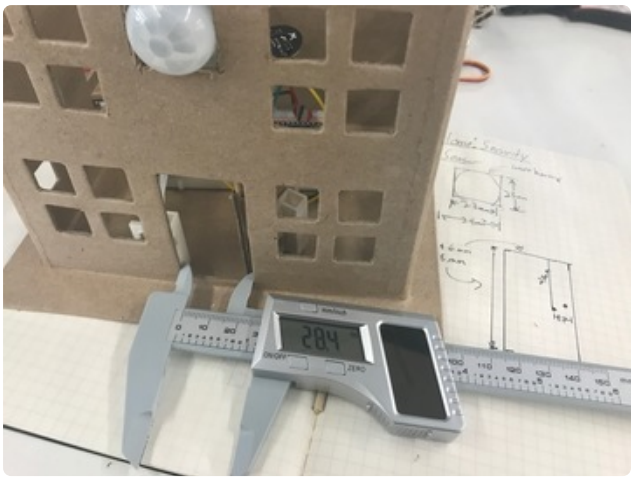
Next, we're going to add a door and a sensor to detect when the door is either open or closed. We'll be using a reed switch (http://adafru.it/375) to accomplish this. These super simple sensors operate on the principle that when a magnet is less than 0.5" away, the reed switch internally closes. We'll detect this change with the Feather HUZZAH and send the value to the Adafruit IO Dashboard's Door Status Indicator.

These switches aren't just for an IO Cardboard Home, they've been used in real-world Adafruit Learning Guides such as the IO Door Detector (https://adafru.it/jdq).

Using a hot glue gun, glue the reed switch to the side of the door-frame.

If you don't have a hot glue gun handy, you can substitute a piece of double-sided tape instead.

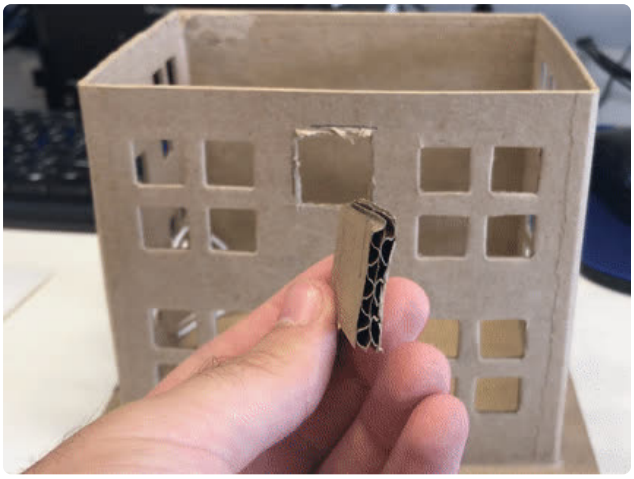
Oh no...our house does not have a door. But we can **build one out of CARDBOARD!**



First, take measurements of the doorframe using a ruler (http://adafru.it/1554) or pair of digital calipers (http://adafru.it/3720). We measured ours to be 58mm tall and 28.6mm wide, but rounded down for cutting out the door.

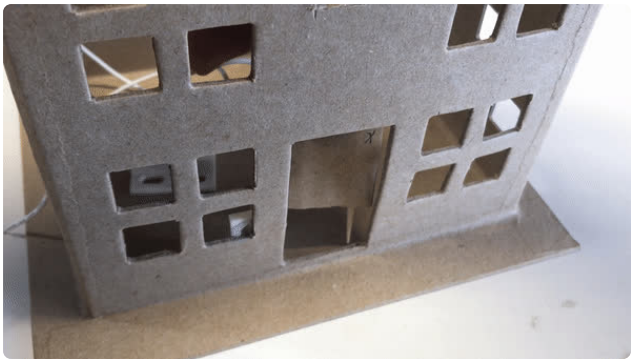
Next, grab the closest empty Adafruit/ Amazon/DigiKey box (We know you have some around, we do!).

Using a ruler, sketch a door on a this piece of cardboard. Then, cut it out using a pair of scissors or a knife.



We'll want a a way for the door to open and close. To create the door hinge, use a Bamboo skewer and poke it through the corrugated cardboard's fluting.

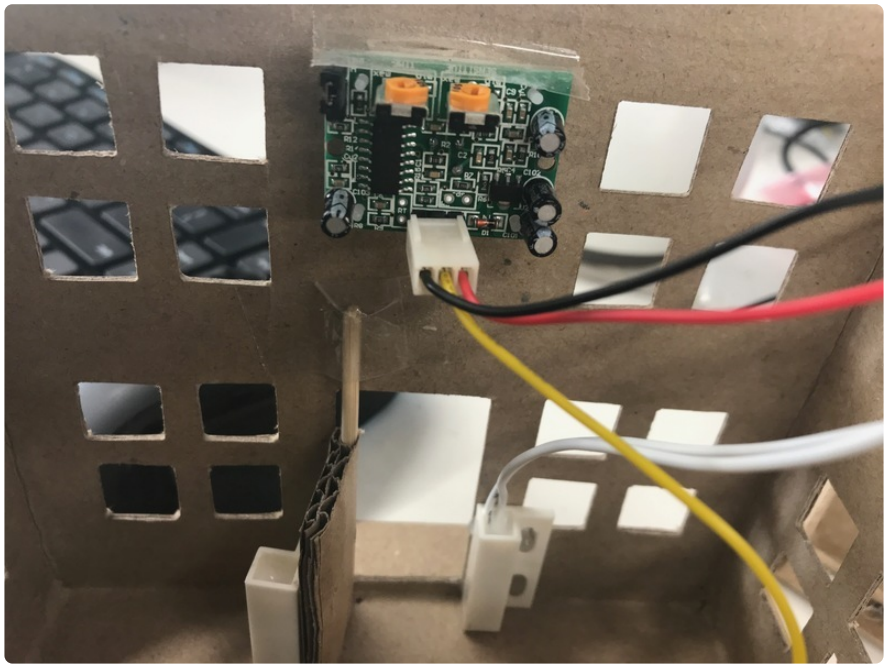
Using double sided tape or hot glue, affix the magnet onto the door.



Test-fit the hinge and make sure the door doesn't catch on the either the sensor or the frame.

Also - make sure the magnet passes the reed switch when it's opened.

To finish off the assembly, we'll use transparent tape to affix the PIR sensor and the top of the door skewer to the front face of the home.

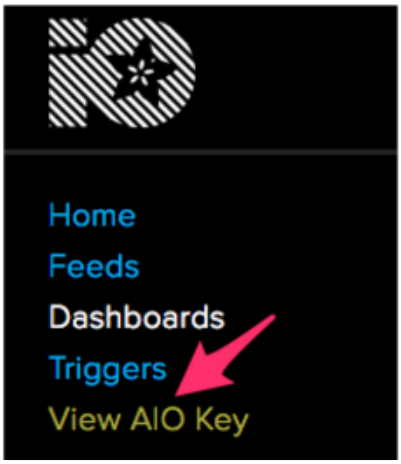


Next, we'll wire up our hardware!

---

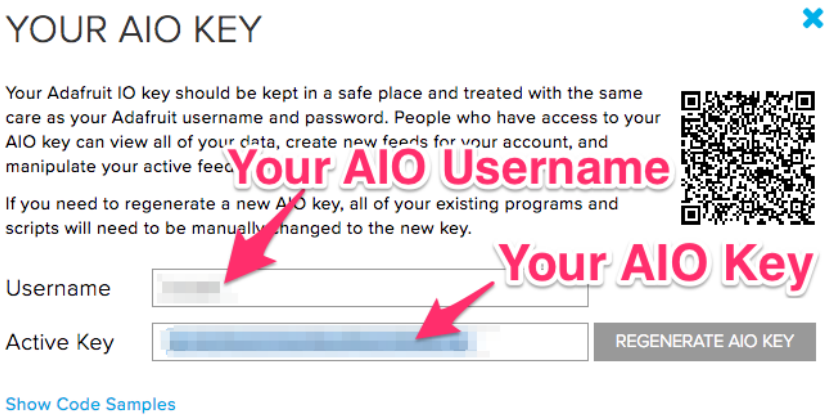
# Adafruit IO Setup

New to Adafruit IO? You can read all about it in this guide (https://adafru.it/Co4). Go ahead and get your account set up, and come back to this page when you're ready.




We'll need to obtain our Adafruit IO Key and Username. Visit your Adafruit IO Profile page (https://adafru.it/BmD) and click the **VIEW AIO KEY** button on the left-sidebar.

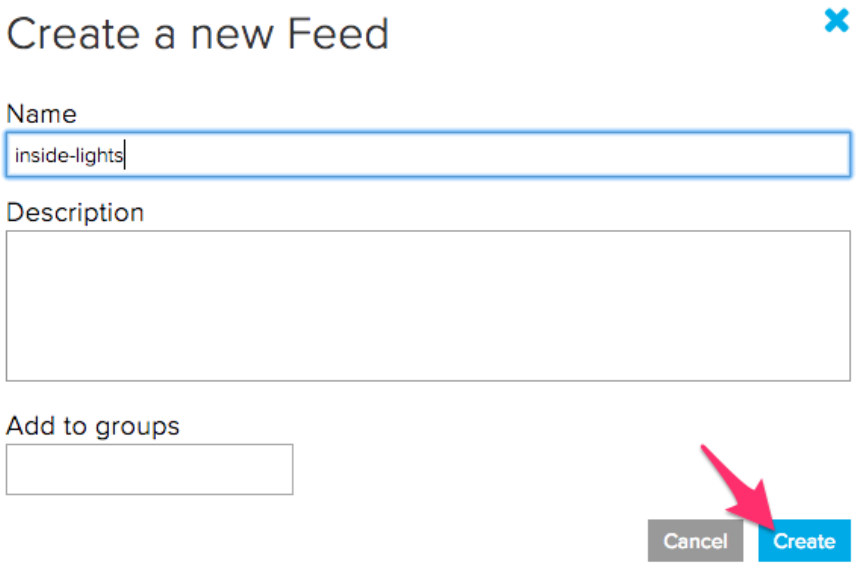
A window will pop up with your Adafruit IO key and username. Keep a copy of them in a safe place, we'll need them later.




## Setting up Adafruit IO Feeds

We are going to create a new feed per each unique source of data. We'll use feeds for a variety of data, from the home's eCO2 level to the colors of the lights inside the house.

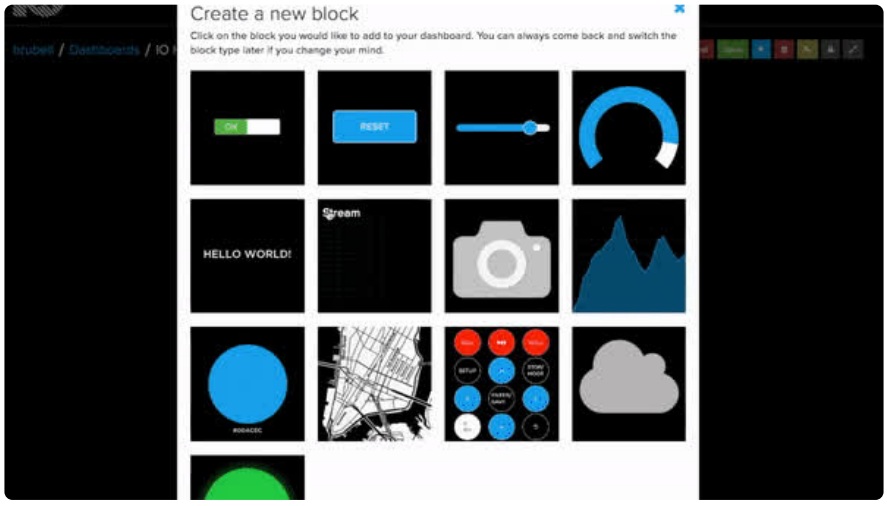
To create a feed for the lights inside the house, navigate to the Adafruit IO Feeds Page (https://adafru.it/mxC) and click **Actions->Create a New Feed.** Name the new feed inside-lights.




Next, we're going to create the other feeds which will be used by our house. Create a feed for each of the following: **outside-lights, eco2, tvoc, front-door, motion-detector, and alarm-status**

- If you do not know how to create feeds, head over to the Adafruit IO Basics: Feeds for a quick overview (https://adafru.it/ioA) of this process.

## Creating the Adafruit IO Dashboard




Next, we'll create an Adafruit IO Dashboard to display and control our feeds. Navigate to the Adafruit IO Dashboard page (https://adafru.it/elS) and click **Actions -> Create a New Dashboard**.

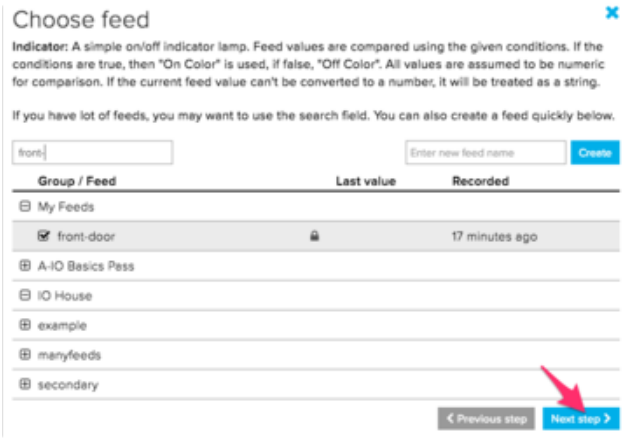
Name this dashboard **IO Home** and **click Create.** You'll be re-directed to the new Dashboard. If you have already created a dashboard using the previous IO Home learning guide, you can disregard this step.

## Adding Status Indicators

We're going to **create three Indicator Blocks** to monitor the status of our security system and environmental sensor.

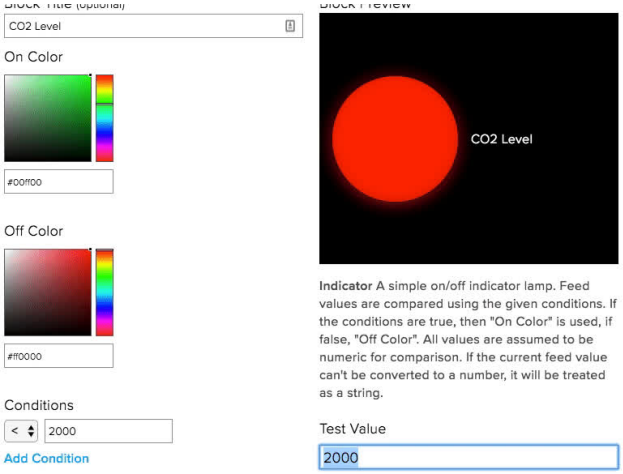
From the IO Home dashboard, click the **blue plus icon** to add a new block to the dashboard. Click the **indicator block**, select the front-door feed, configure its settings, and click **create block**.

We're going to **create three Indicator Blocks** to monitor the status of our security system and environmental sensor.




From the IO Home dashboard, **click the blue plus icon** to add a new block to the dashboard. **Click the indicator block**, **select the front-door feed**, **configure its settings**, and **click create block.**


Repeat this process for the motion-detector feed.




We'll take advantage of the Conditions feature of the Indicator Block. You can set conditions to trigger the block's off color. We're going to set a condition for "when the CO2 sensor (SGP30) feed receives a value greater than 2000 (parts per million), set the indicator to the off color".

## Adding Alarm Controls

What good is an is a security system if it can only notify you? Let's add controls for an alarm system to scare off wanna-be intruders.

**Add a toggle block** to your dashboard, and **connect it to an alarm-status feed**.

Here's an example of the dashboard we made for this guide. You can go crazy and add buttons/switches/colors, or keep it tame and add only what you need.

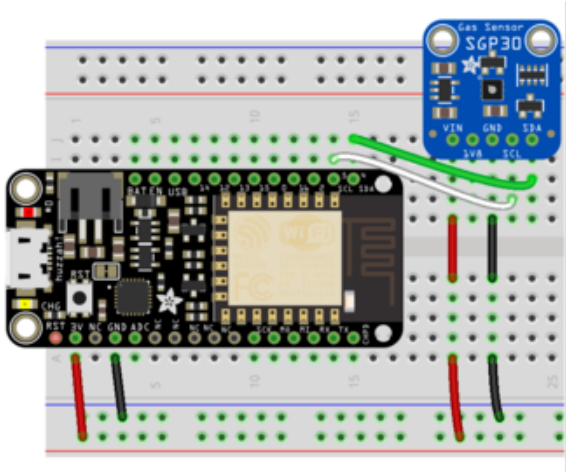
**Note:** The color picker blocks for lighting was created in the previous guide, click here to read the Adafruit IO Setup portion of the guide (https://adafru.it/CpF).

Next, we're going to set up our hardware for use with Adafruit IO.

---

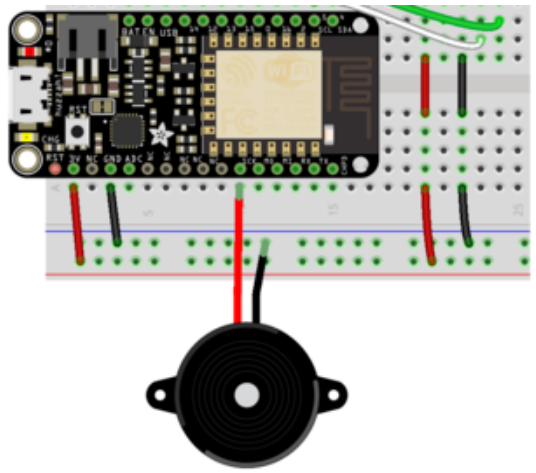
# Arduino Wiring

The Wiring for this sketch is less complicated than it looks. We'll connect each sensor/output to the Feather individually.



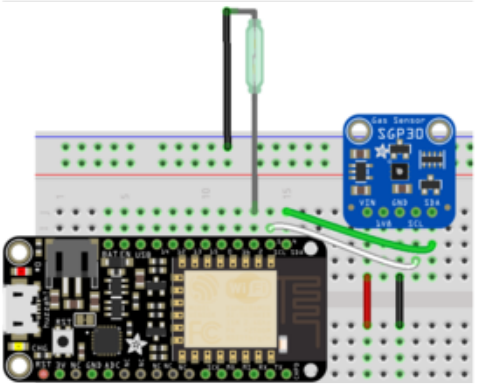

Make the following connections **between the Feather HUZZAH and the SGP30:**

**Feather 3V** to **SGP30 Vin**
**Feather GND** to **SGP30 GND**
**Feather SCL** to **SGP30 SCL**
**Feather SDA** to **SGP30 SDA**



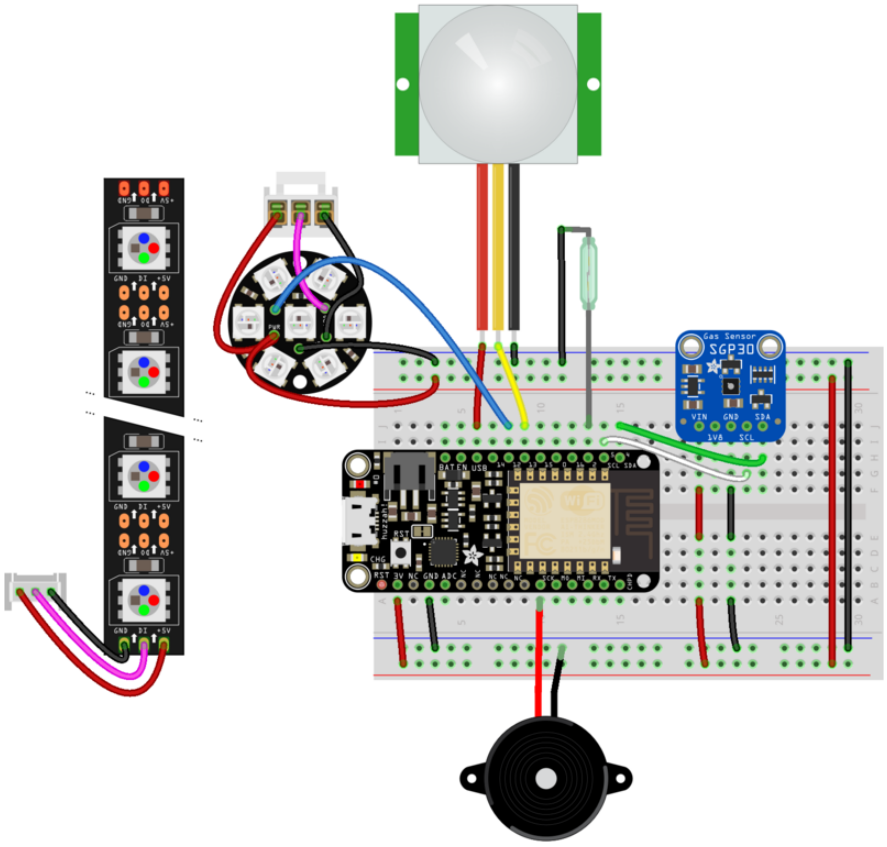
Connect one end of the Piezo to Feather SCK.

Connect the other end to GND.



Make the following connections between the **Feather Huzzah and the Door Sensor (Reed Switch):**

**Feather Pin 2** to one end of the reed switch
**Feather GND** to the other end of the reed switch



Make the following connections between the **Feather Huzzah and the NeoPixel Jewel:**

- **Feather GND** to **NeoPixel Jewel GND**
- **Feather 3V** to **NeoPixel Jewel PWR**
- **Feather Pin 12** to **NeoPixel Jewel DIN**

Make the following connections between the **Feather Huzzah and the PIR Sensor:**

- **Feather USB** to **PIR +5V**

- **Feather GND** to **PIR GND**
- **Feather 13** to **PIR Signal**

Next, let's set up our Arduino for this project.

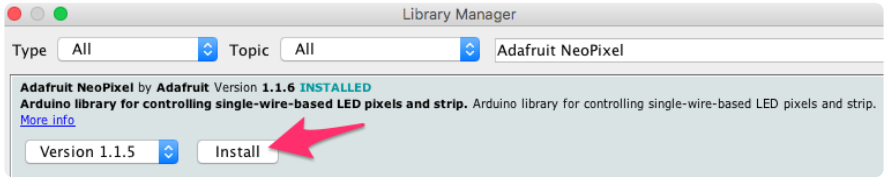
---

# Arduino Setup

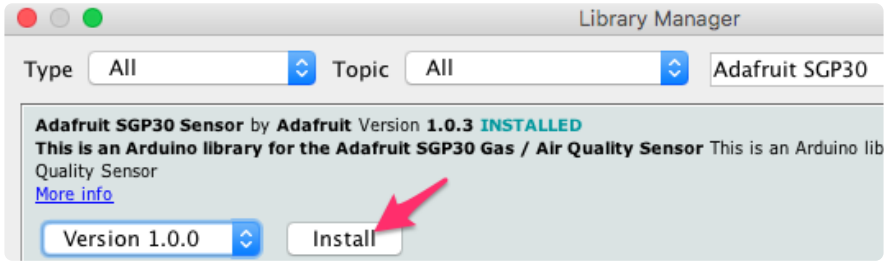
This guide assumes you've completed the setup required to get your ESP8266 up and running with Arduino IDE and Adafruit IO.

- If you haven't yet set up your ESP8266 for use with Adafruit IO and the Arduino IDE, follow along with this guide (https://adafru.it/DCI). The setup only needs to be performed once.

We'll need a library to control the NeoPixels. In the search bar, enter **Adafruit NeoPixel.** Click **Install.**

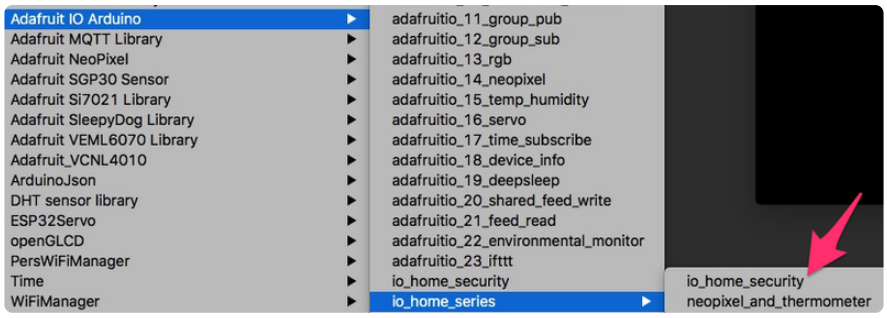



We'll need a library to read the SGP30 sensor. In the search bar, enter **Adafruit SGP30.** Click **Install.**

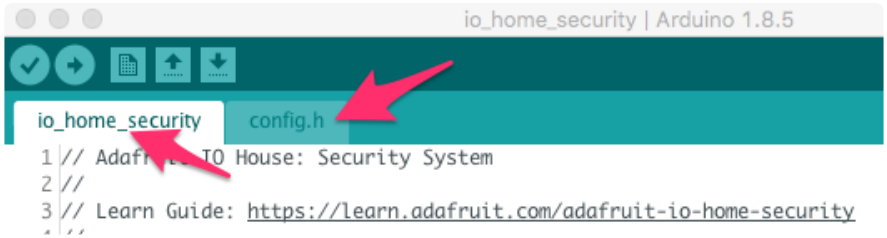



## Opening the Code

The code for this guide is stored within the latest Adafruit IO Arduino Library release (versions =>2.7.17). From the Arduino IDE, navigate to **File->Examples->Adafruit IO Arduino -> io_home_series -> io_home_security.ino**




The IDE should open the sketch (io_home_security.ino) and the configuration file (config.h):

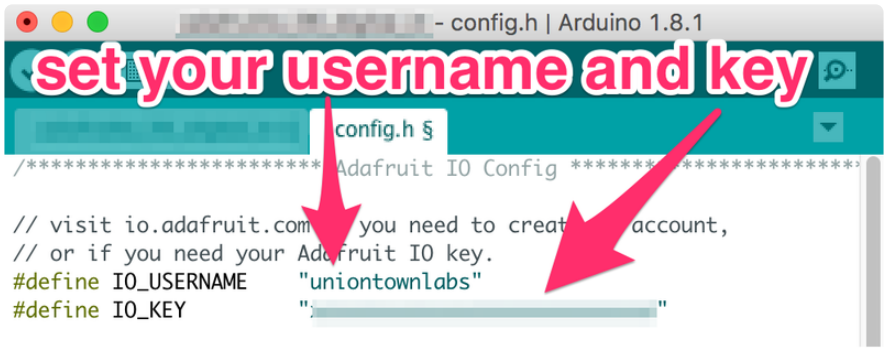



With those files open, let's configure the sketch for use with your Adafruit IO account and network.

---

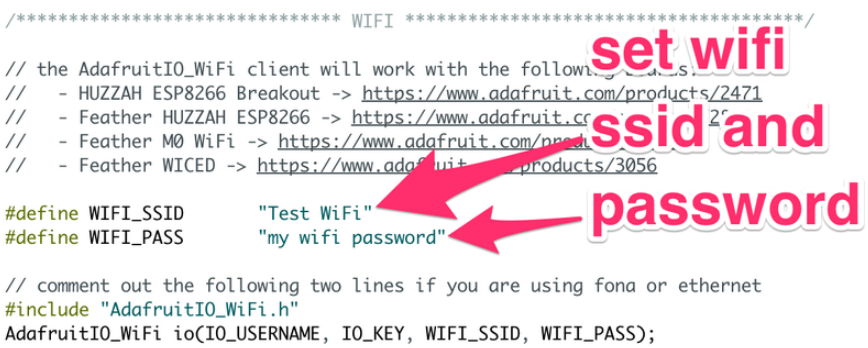
# Arduino Network Config

To configure the network settings, click on the **config.h** tab in the sketch. You will need to set your Adafruit IO username in the **IO_USERNAME** define, and your Adafruit IO key in the **IO_KEY** define.



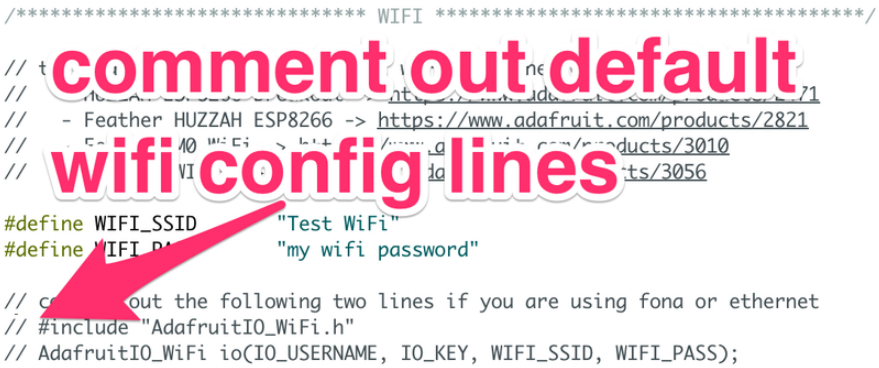

## WiFi Config

WiFi is enabled by default in **config.h** so if you are using one of the supported WiFi boards, you will only need to modify the **WIFI_SSID** and **WIFI_PASS** options in the **config.h** tab.




## FONA Config

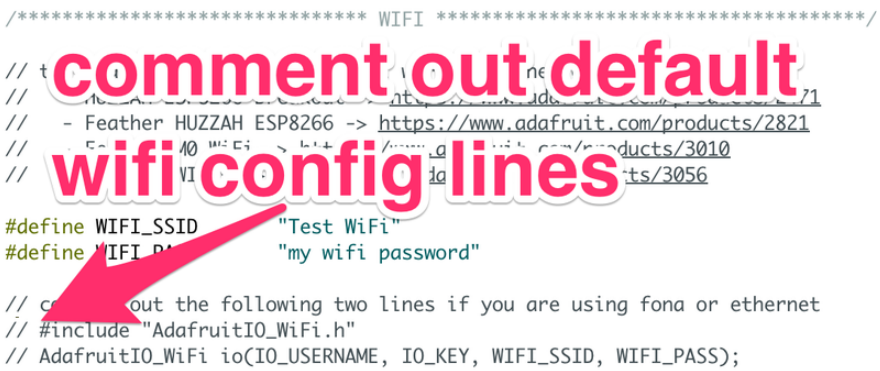
If you wish to use the FONA 32u4 Feather to connect to Adafruit IO, you will need to first comment out the WiFi support in **config.h**




Next, remove the comments from both of the FONA config lines in the FONA section of **config.h** to enable FONA support.


## Ethernet Config

If you wish to use the Ethernet Wing to connect to Adafruit IO, you will need to first comment out the WiFi support in **config.h**




Next, remove the comments from both of the Ethernet config lines in the Ethernet section of **config.h** to enable Ethernet Wing support.


Next, we will look at how the example sketch works.

---

# Arduino Code

## Setting up the Sketch

Since we don't want our motion detector to alert us of daytime activities (like a dog passing by, or a skateboarder), we'll need to give the sketch (prior to uploading) the current hour, minutes and seconds. In the sketch, set `startingHour`, `currentMinutes`, and `currentSeconds` to the current time (we're using 24-hour time for this sketch). For the sketch to work properly, you'll need to change these values on every upload.

- While we're not using one for this project, you can add a Real-Time-Clock (like the RTC FeatherWing (http://adafru.it/3028)) if you'd like more precise timekeeping.

```
/**** Time Setup ****/
// set the current hour
int startingHour = 0;
// set the current minutes
int currentMinutes = 0;
```

```
// set the current seconds
int currentSeconds = 0;
```

## Code Overview

The first chunk of the code (`setup()`) starts a MQTT connection to the Adafruit IO server. We also add three message handlers to handle changes from the indoor-lights, outdoor-lights, and home-alarm feeds from the dashboard blocks. We also set up the PIR sensor, reed switch, SGP30 sensor, and initalize the NeoPixel strip.

```
void setup() {
  // start the serial connection
  Serial.begin(115200);

  // wait for serial monitor to open
  while(! Serial);
  Serial.println("Adafruit IO Home: Security");

  Serial.println("Connecting to Adafruit IO");
  // start MQTT connection to io.adafruit.com
  io.connect();

  // attach a message handler for the `home-alarm` feed
  alarm-&gt;onMessage(handleAlarm);
  // subscribe to lighting feeds and register message handlers
  indoorLights-&gt;onMessage(indoorLightHandler);
  outdoorLights-&gt;onMessage(outdoorLightHandler);

  // wait for an MQTT connection
  // NOTE: when blending the HTTP and MQTT API, always use the mqttStatus
  // method to check on MQTT connection status specifically
  while(io.mqttStatus() &lt; AIO_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  // we are connected
  Serial.println();
  Serial.println(io.statusText());

  // declare PIR sensor as input
  pinMode(pirPin, INPUT);
  // declare reed switch as input
  pinMode(doorPin, INPUT);
  // set up the SGP30 sensor
  setupSGP30();
  // init the neopixel strip and set to `off`
  strip.begin();
  strip.show();
}
```

The next chunk of code (`loop()`) calls `io.run()` which keeps the client connected to Adafruit IO. Next, it makes a call to `getTime()` which calculates and returns the current time. Then we read the door, SGP30, and PIR sensor (`readDoorSensor`, `readPIR`, `readSGP30`) and send their values to their respective Adafruit IO feeds.

```
void loop(){
  io.run();
```

```
  getTime();
  Serial.println("* read door sensor...");
  readDoorSensor();
  Serial.println("* read motion detector");
  readPIR();
  Serial.println("* reading SGP30...");
  readSGP30();
}
```

The toggle button on your Adafruit IO dashboard is connected to the `handleAlarm` message handler. When the toggle button's value is changed, it receives the current value of the feed and checks if it's `ON` or `OFF` and sets the `isAlarm` boolean.

```
void handleAlarm(AdafruitIO_Data *data) {
// handle the alarm toggle on the Adafruit IO Dashboard
  String toggleValue = data-&gt;toString();
  Serial.print("&gt; rcv alarm: ");
  Serial.println(toggleValue);
  if(toggleValue == String("ON")) {
    Serial.println("* Alarm Set: ON");
    isAlarm = true;
  } else {
    Serial.println("* Alarm Set: OFF");
    isAlarm = false;
  }
}
```

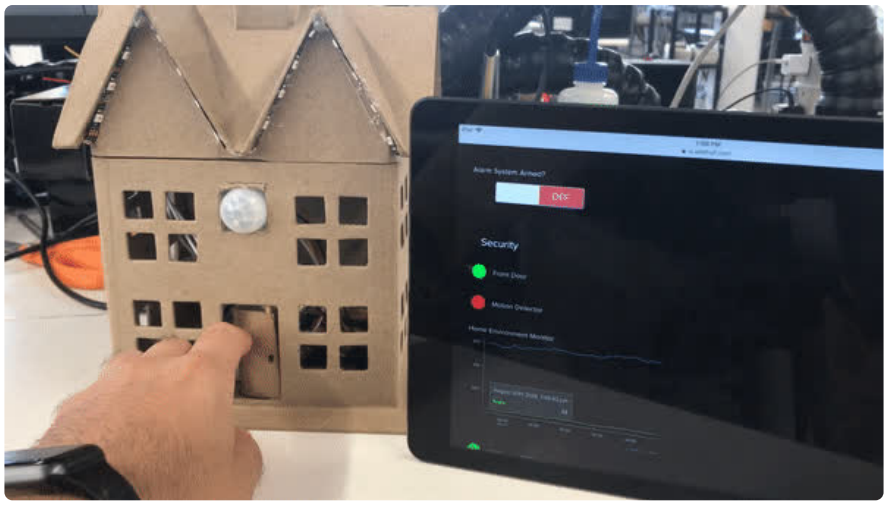
Finally, we'll check if the alarm is armed (`isAlarm==true`). If the alarm is armed, we check if the door is open (`doorState == HIGH`) or if the hour is later than `alarmHour` **and** if the motion detector is triggered, we'll make a call to the `playAlarmAnimation()` function.

```
  // check if the alarm toggle is armed from the dashboard
  if (isAlarm == true) {
    if (doorState == HIGH || (hour&gt;alarmHour &amp;&amp; pirState == HIGH)) {
      playAlarmAnimation();
   }
  }
```

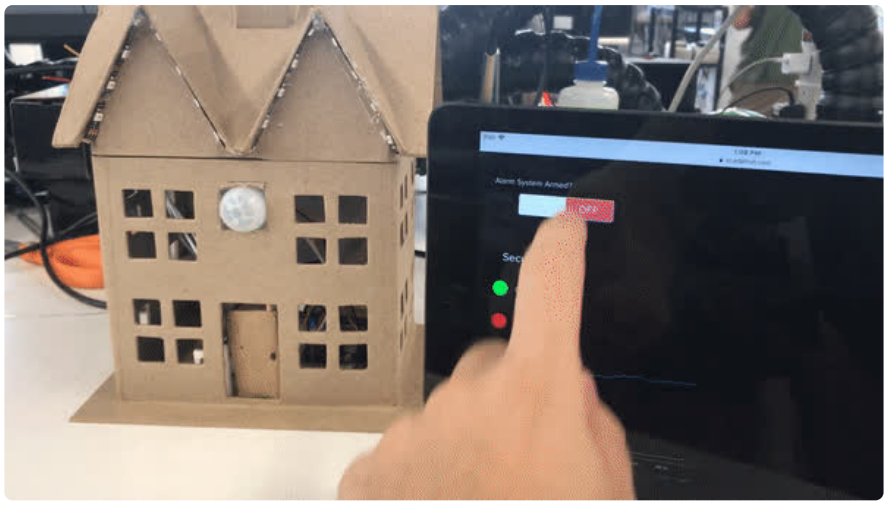
## Using the Smart Home Security System

Connect your Feather Huzzah to USB and upload the **io_home_security** sketch.

Open up the IO-Home dashboard to interact with your smart home. Push the door open. You should see the **Front Door Indicator** change from green to red.




Switch the toggle from **OFF** to **ON** to arm your house's security system. When you push the front door open (or move past the PIR sensor past the time set by `alarmHour`), the inside lights will blink red and the piezo will make noise.

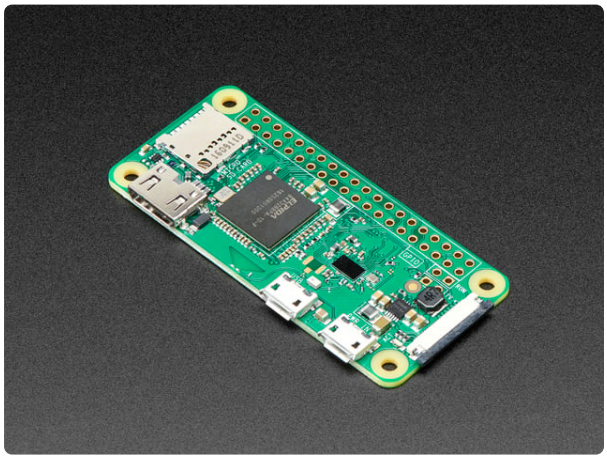


Tired of the alarm? Want some peace and quiet? Toggle the alarm **OFF**.

# Python Wiring

This guide is also compatible with the Raspberry Pi using CircuitPython. We're going to build the house's lighting system, wire up the home security circuit, and program it with CircuitPython.

The Pi Zero W has **built-in WiFi** - which is great for connecting our environmental monitor to Adafruit IO. It's also smaller than a regular Raspberry Pi 3, making it the perfect size to stick it in a small corner of your room.



Raspberry Pi Zero W
If you didn't think that the Raspberry Pi Zero could possibly get any better, then boy do we have a pleasant surprise for you! The new Raspberry Pi Zero W...
https://www.adafruit.com/product/3400

While we could use a breadboard, **we'll build our own pHAT** for our IO House.

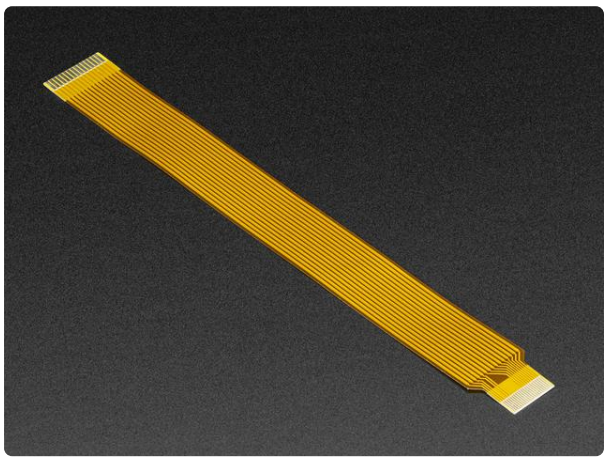
Adafruit Perma Proto Bonnet Mini Kit
Design your own Bonnet or pHAT, attach custom circuitry and otherwise dress your Pi Zero with this jaunty prototyping Bonnet kit!To add to the
https://www.adafruit.com/product/3203

## Home Surveillance

Since we're using a Raspberry Pi, we're going to use the eight-megapixel Raspberry Pi Camera v2 to capture images from outside our home.

Raspberry Pi Camera Board v2 - 8 Megapixels
Snap, snap! The Camera v2 is the new official camera board released by the Raspberry Pi Foundation!The Raspberry Pi Camera Board v2 is a high quality 8...
https://www.adafruit.com/product/3099

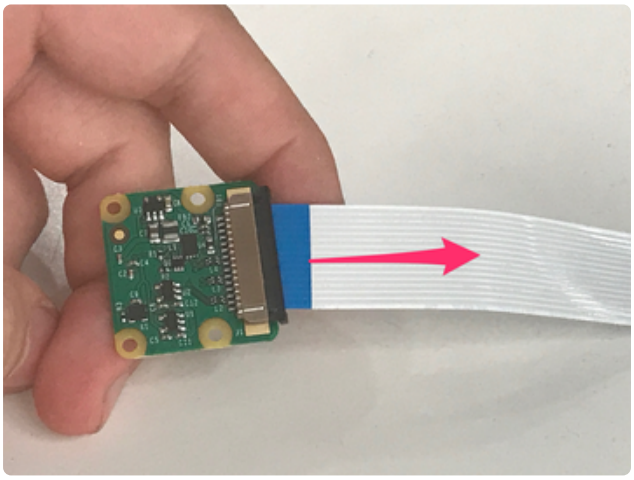
Because the Pi Camera Board is designed for the Raspberry Pi and not the Raspberry Pi Zero, we'll use a Raspberry Pi Zero camera cable.



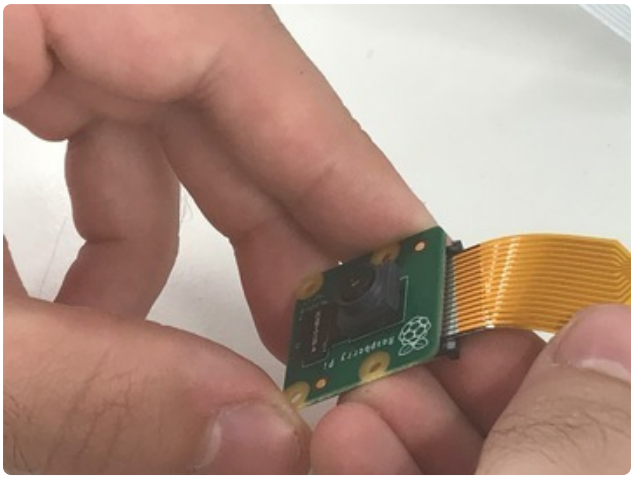
Raspberry Pi Zero v1.3 Camera Cable
This camera cable is specifically designed to work with the Raspberry Pi Zero - Version 1.3! Just plug it into your Pi...
https://www.adafruit.com/product/3157

Make sure the Pi is not connected to power during any of these steps, including the camera connection.

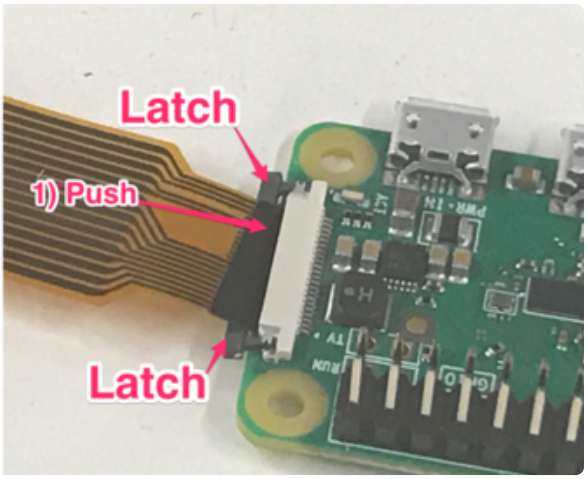
## Connect the Raspberry Pi Camera



Remove the CSI cable which comes from the Pi Camera and replace it with the Pi Zero camera cable by lifting the collar of the camera module and removing the cable.

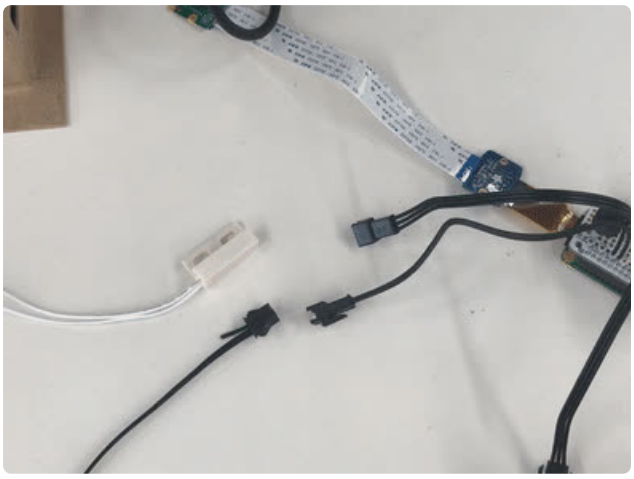


Connect the wider end of the Pi Zero camera cable into to the Pi Camera.




Open the collar on the Pi Zero W's camera connector. Push the camera cable through the connector, making sure the cable is seated firmly in place and the contacts are facing the back of the Pi Zero.

## (Optional) Adding Quick Connects

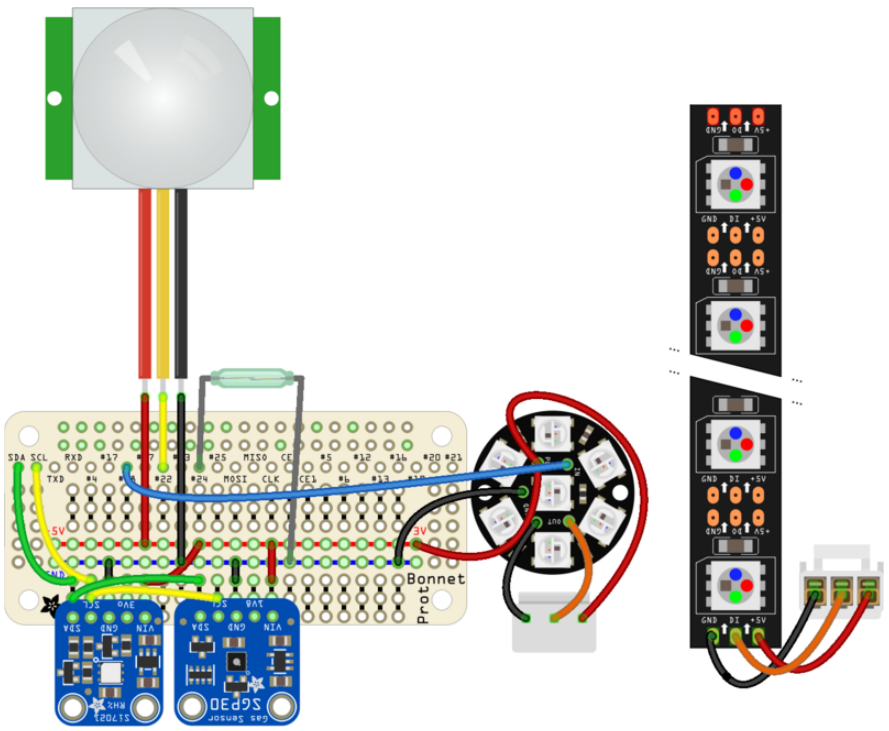


While going through this project, I found it easiest to add a JST SM receptacle to the bonnet and attach the plug to the end of the cable. It makes it easy to quick connect / disconnect everything in your home from the Pi.

I did this for both the PIR sensor (using a 3-pin JST) (http://adafru.it/1663) and the reed switch (using a 2-pin JST (http://adafru.it/2880)).

## Wiring the Pi

*pi camera not shown



**NOTE:** Since this is a series, **we'll be building off what we built in** the previous guide **(https://adafru.it/Cru)**, but adding a SGP30 (you can wire it by chaining SCL/SDA pins), the motion detector (PIR sensor), and a PiCam.

Connect the SGP30 to the Pi:

- **SGP30 SCL** to **Pi SCL**
- **SGP30 SDA** to **Pi SDA**
- **SGP30 GND** to **Pi GND**
- **SGP30 Vin** to **Pi 3V**

Connect the PIR Sensor to the Pi:

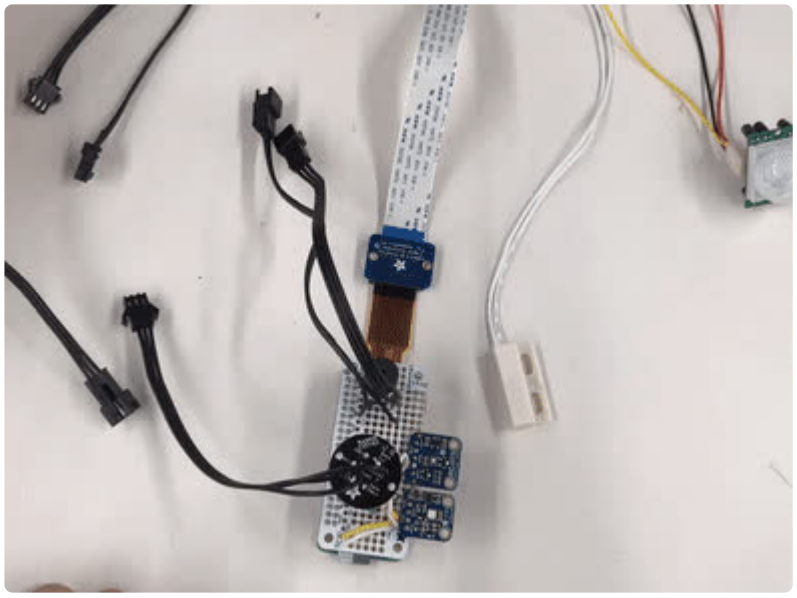
- **PIR 5V** to **Pi 5V**
- **PIR Signal** to **Pi GPIO #22**
- **PIR GND** to **Pi GND**

Connect the NeoPixel Jewel to the Pi:

- NeoPixel **Jewel GND** to **Pi GND**
- NeoPixel **Jewel PWR** to **Pi 3V**
- NeoPixel **Jewel IN** to **Pi GPIO #18**

Connect one wire of the **Reed Switch to Pi GPIO #24**, and connect the other wire to GND.

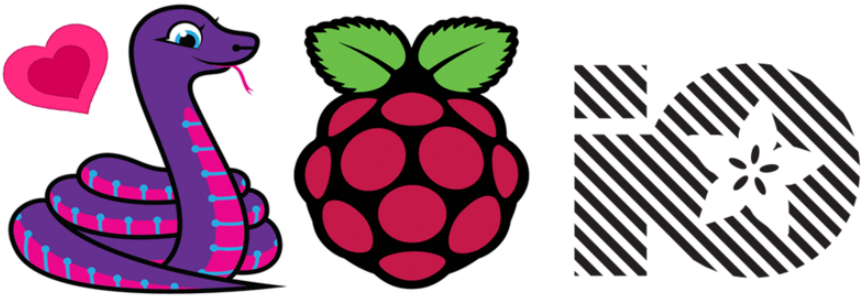
We're ready to power up - connect the plugs for the sensors and the neopixel strip/jewel to the appropriate receptacles and connect your Pi to a power supply.



Next, let's move on to setting up your Pi.

---

# Python Setup



If you're following along with a Raspberry Pi, Beaglebone or any other supported small linux computer, we'll use a special library called **adafruit_blinka** (https://adafru.it/BJS) (named after Blinka, the CircuitPython mascot (https://adafru.it/BJT)) to provide the layer that translates the CircuitPython hardware API to whatever library the Linux board provides. It's CircuitPython, on Pi!

If you haven't set up **Blinka and/or the Adafruit IO Python Library yet on your Raspberry Pi**, follow our guide before continuing with the steps on this page:

- Blinka and Adafruit IO Setup (https://adafru.it/BMB)

## Enabling I2C

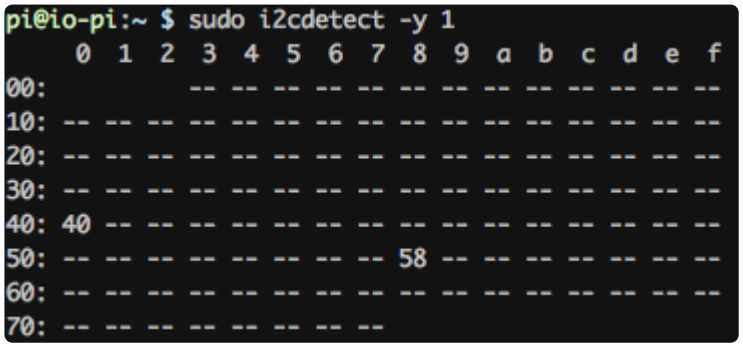
We "talk" to the Si7021 and SGP30 sensors over I2C. To do this, we'll need to set up our Pi's I2C interface. You only have to do this once per Raspberry Pi, the interface is disabled by default.

- Enabling I2C (https://adafru.it/dEO)

Once you're done with this and have rebooted, verify you have the I2C devices with the command:

```
sudo i2cdetect -y 1
```

The output from running this command should look like the following:




## Installing the Adafruit_CircuitPython_SGP30 Sensor Library

We'll need libraries to communicate with the SGP30 sensor. Since we're using Adafruit Blinka (CircuitPython), we can install CircuitPython libraries straight to our Raspberry Pi by using PyPi.

Enter the following command into your terminal to install the **Adafruit_CircuitPython_SGP30** library:

```
sudo pip3 install adafruit-circuitpython-sgp30
```

## Installing the Adafruit_CircuitPython_NeoPixel Library

To control our NeoPixels, we'll use the NeoPixel Library for CircuitPython (https://adafru.it/Cr-).

Enter the following command into your terminal to install the **Adafruit_CircuitPython_NeoPixel** library:

```
sudo pip3 install Adafruit_CircuitPython_NeoPixel
```

## Installing and Configuring picamera

Depending on what distro you are using, picamera may be installed by default. You can check this by entering the following into your terminal:

```
python3 -c "import picamera"
```

If you get an error, something like the following:

```
Traceback (most recent call last):
File "&lt;string&gt;", line 1, in &lt;module&gt;
ImportError: No module named 'picamera'
```

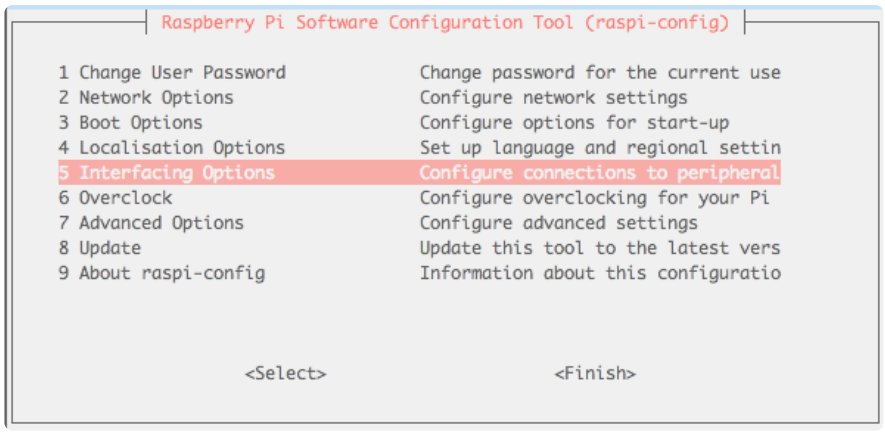
You'll need to install picamera. We're going to enter two commands into our terminal to install it:

```
sudo apt-get update
sudo apt-get install python-picamera python3-picamera
```

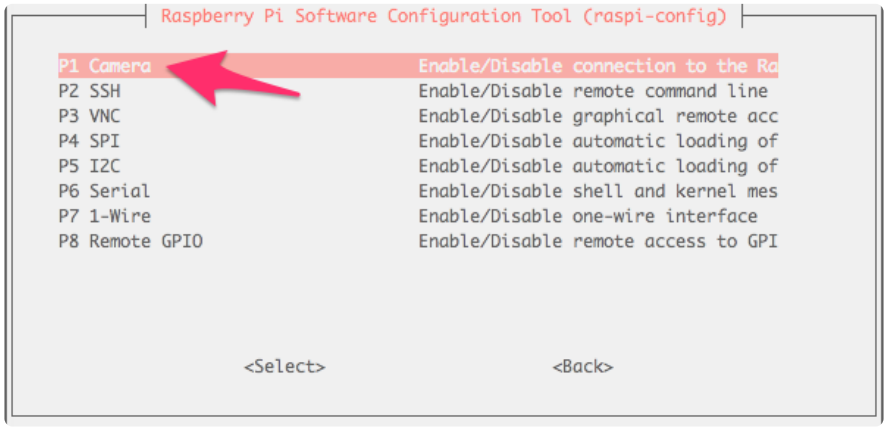
Once you've got it installed, or if you did not get an error earlier, we'll configure our Raspberry Pi Zero for use with the camera.

**Type** `raspi-config` into your terminal and **press enter.**

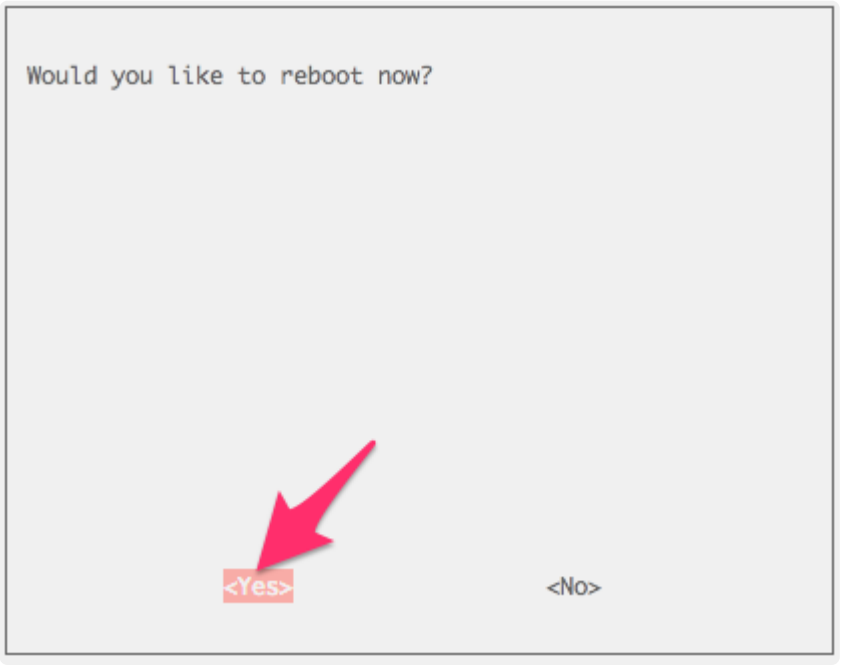
You'll be greeted with the following screen. Using your arrow keys, navigate to **Interfacing Options** and **press the enter key**.




The first option is to enable the camera, use your arrow keys to navigate to this option and **press enter.**



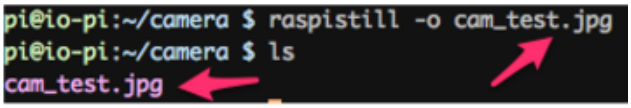

You'll be prompted to reboot the Pi. Use the arrow keys to select the **Yes** option and **press enter**.

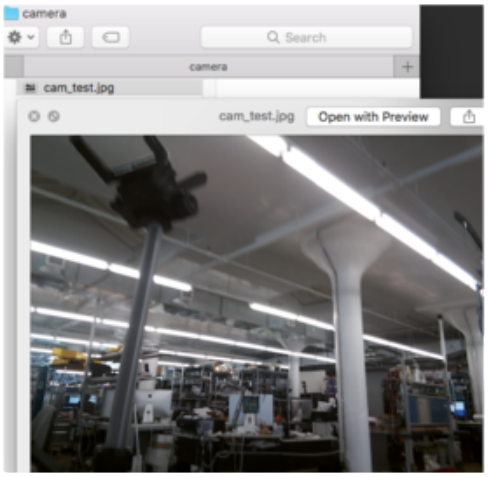


After the Pi reboots, test that the camera works by using the raspistill (https://adafru.it/jd3) command:

```
raspistill -o cam_test.jpg
```

If this command fails, go back and check that you have enabled the camera using `raspi-config` and the cable is firmly connected to both the Pi's connector and the camera board's connector.




Typing `ls` into the terminal should return the `cam_test.jpg` file. You can copy this file off of the Pi and view it on your computer, too!

The Pi Camera is set up and ready to use!

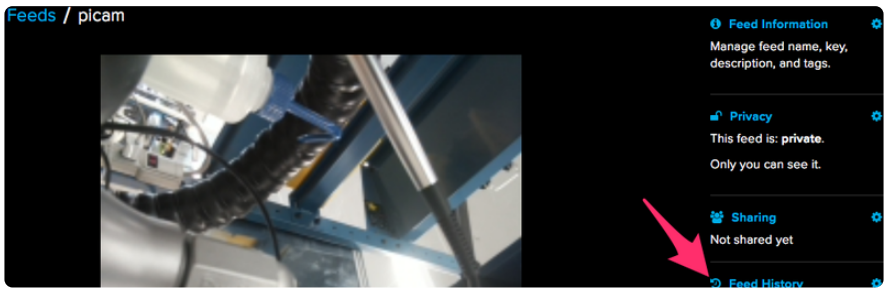
## Adafruit IO Security Camera Dashboard Setup

Since we're using a camera, we'll need to make a small modification to our dashboard to display the feed with the images captured.

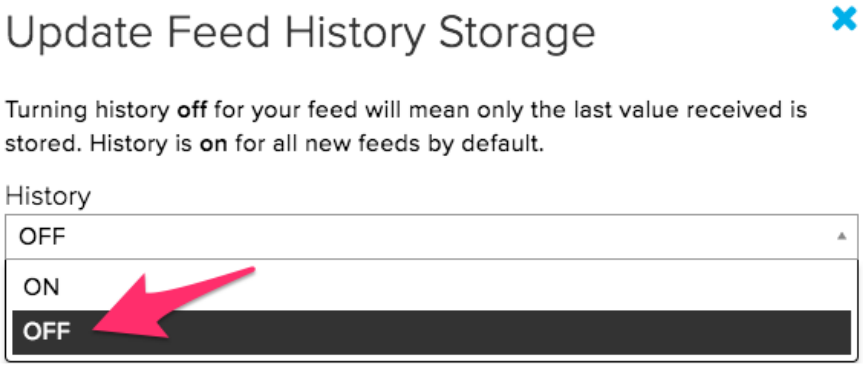
To do this, we'll **create a new feed called `picam`**.

Adafruit IO has a limit when it comes to image feeds - you can not store history for feeds containing large values (like base64-encoded images).

To avoid this, we'll turn off the feed's history. **Navigate to the feed page for the picam** and **click Feed History**.




From the popup, **disable feed history by setting it to OFF**.




## Python Code Setup

The code for this guide is located on the Adafruit Learning Guides repository on GitHub under the IO House Series (https://adafru.it/Cpj).

If you'd like to directly load this code on your Pi, **type the following in your terminal and press enter:**

```
git clone https://github.com/adafruit/
Adafruit_Learning_System_Guides.git
```

Then, navigate to the directory where the code for this tutorial is located by typing the following into your terminal:

```
cd Adafruit_Learning_System_Guides/IO_House_Series/Security
```

Let's check to make sure the Python code is within that directory. **To do this, type `ls` into your terminal and hit enter.** You should see `code.py` in this directory:

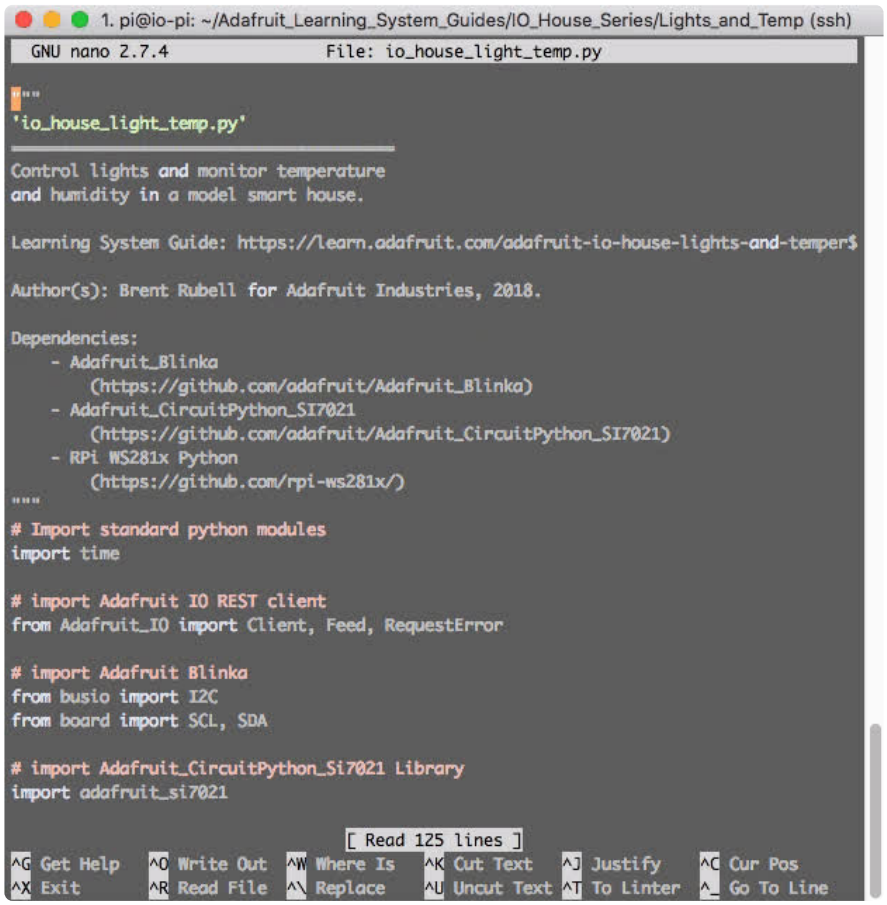
**Before running the code, we need to set our Adafruit IO Key(`ADAFRUIT_IO_KEY`) and Adafruit IO Username(`ADAFRUIT_IO_USERNAME`).**

To do this, open the code in your text editor of choice (I'm using nano for this example) by entering the following into your terminal:

```
nano code.py
```

The code should open in the nano editor. Scroll down to the `Adafruit_IO_KEY` variable and set it to the your Adafruit IO Key. Then, set the `Adafruit_IO_USERNAME` to your Adafruit IO Username.

- If you do not have these values, navigate to your Adafruit IO Profile page (https://adafru.it/BmD) and click ACTIVE IO KEY




When you're done editing the values, **save the file by pressing control + x.**

When prompted to save the modified buffer, **type Y and press enter.**

At the File Name to Write prompt, **press enter** and you should be directed back to the terminal.

Next, we're going to learn how the code works and run it from our Pi.

---

# Python Code

The first chunk of our code sets up the Adafruit IO Feeds:

```
tvoc_feed = aio.feeds('tvoc')
eco2_feed = aio.feeds('eco2')
door_feed = aio.feeds('front-door')
motion_feed = aio.feeds('motion-detector')
alarm_feed = aio.feeds('home-alarm')
outdoor_lights_feed = aio.feeds('outdoor-lights')
indoor_lights_feed = aio.feeds('indoor-lights')
picam_feed = aio.feeds('picam')
```

Then, it sets up the camera, sensors, and NeoPixel Strip:

```
# set up PiCamera
camera = picamera.PiCamera()
camera.resolution = (320, 240)

# set up door sensor
door_sensor = digitalio.DigitalInOut(D5)
door_sensor.direction = digitalio.Direction.INPUT

# set up motion sensor
pir_sensor = digitalio.DigitalInOut(D6)
pir_sensor.direction = digitalio.Direction.INPUT
prev_pir_value = pir_sensor.value
is_pir_activated = False

# set up sgp30
i2c_bus = I2C(SCL, SDA, frequency=100000)
sgp30 = adafruit_sgp30.Adafruit_SGP30(i2c_bus)

# set up the neopixel strip
pixels = neopixel.NeoPixel(D18, NUM_PIXELS_STRIP)
pixels.fill((0, 0, 0))
pixels.show()
```

In the loop (`while True`), we first take measurements for the `co2eq` and `tvoc` from the SGP30 and send those values to Adafruit IO.

```
  co2eq, tvoc = sgp30.iaq_measure()
  print("CO2eq = %d ppm \t TVOC = %d ppb" % (co2eq, tvoc))
  # send SGP30 values to Adafruit IO
  aio.send(eco2_feed.key, co2eq)
  aio.send(tvoc_feed.key, tvoc)
  time.sleep(0.5)
```

Next, the code reads the door and motion sensor. If the door is open, or if motion is detected, a value of **3** is sent the corresponding feed to change the indicator block's color. For the motion detector, we also keep track of the motion using a boolean, `is_pir_activated`.

```
# read/send door sensor
if door_sensor.value:
  print('Door Open!')
  # change indicator block to red
  aio.send(door_feed.key, 3)
else:
  print('Door Closed.')
  # reset indicator block to green
  aio.send(door_feed.key, 0)
# read/send motion sensor
if door_sensor.value:
  if not prev_pir_value:
    print('Motion detected!')
    is_pir_activated = True
    # change indicator block to red
    aio.send(motion_feed.key, 3)
else:
  if prev_pir_value:
    print('Motion ended.')
    is_pir_activated = False
    # reset indicator block to green
    aio.send(motion_feed.key, 0)
```

Next, we'll take a picture with the picam (using `camera.capture()`), convert it to a base64-encoded string (for use with the image block on an Adafruit IO Dashboard), and send it to Adafruit IO (`aio.send()`)

```
camera.capture('picam_img.jpg')
print('snap!')
with open("picam_img.jpg", "rb") as imageFile:
  str = base64.b64encode(imageFile.read())
  try:
    aio.send('picam', str)
  except:
    print('Sending camera image failed...')
```

Finally, we check if the alarm toggle block on the dashboard has been enabled. If it has been, we take a sample of the current hour (according to the Pi's internal clock. You can set this in **`raspi-config`->Internationalization Options->Change Time Zone)**. Then, if the hour is later than the ALARM_HOUR variable and if the PIR has been triggered - we call the `alarm_trigger()` method.

```
# Alarm System
is_alarm = aio.receive(alarm_feed.key)
if (is_alarm.value == "ON"):
  # sample the current hour
  cur_time = time.localtime()
  cur_hour = time.tm_hour
  if (cur_hour &gt; ALARM_HOUR and is_pir_activated == True):
    alarm_trigger()
```

The alarm toggle on your Adafruit IO dashboard is dependent on if the alarm set in the `ALARM_HOUR` variable. You can also set the delay between executing the loop (`while True`), `LOOP_INTERVAL`

```
# Set to the hour at which to arm the alarm system, 24hr time
ALARM_HOUR = 16

# Set to the interval between loop execution, in seconds
LOOP_INTERVAL = 2
```

To run the script, enter:

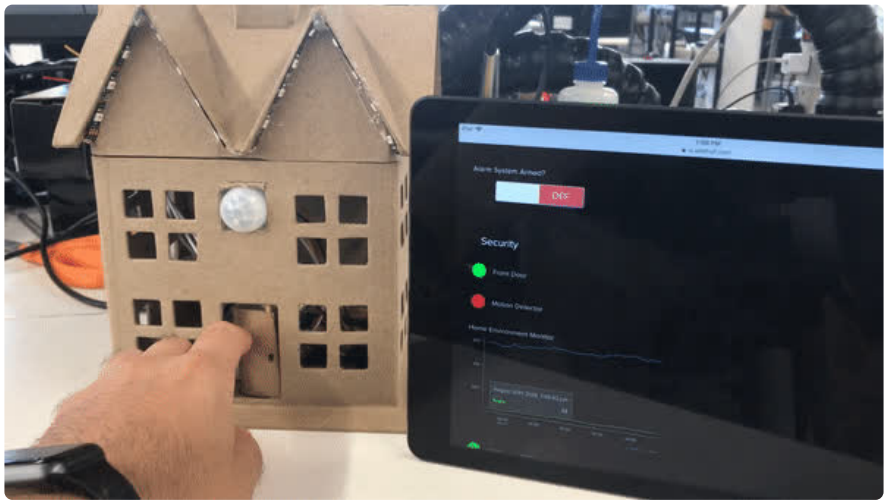
```
sudo python3 io_home_security.py
```

**Note:** You'll need to prefix the script with sudo to write to the NeoPixels.

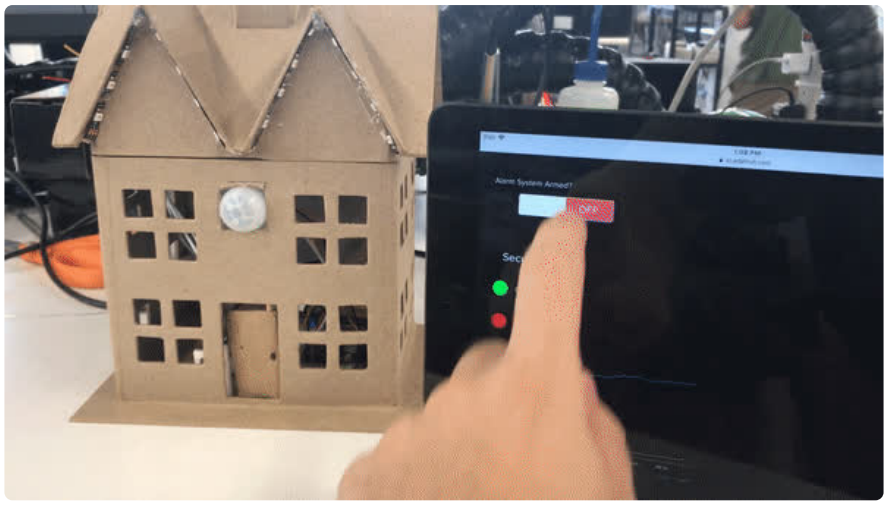
If everything went correctly, you should see the following in your terminal:

```
Adafruit IO Home: Security
CO2eq = 400 ppm  TVOC = 0 ppb
Door Closed.
Motion detected!
snap!
sent to AIO!
```

Let's interact with our home from the IO-Home Dashboard! From your Adafruit IO account, select the IO-Home dashboard and push the door open. You should see the **Front Door Indicator** change from green to red.

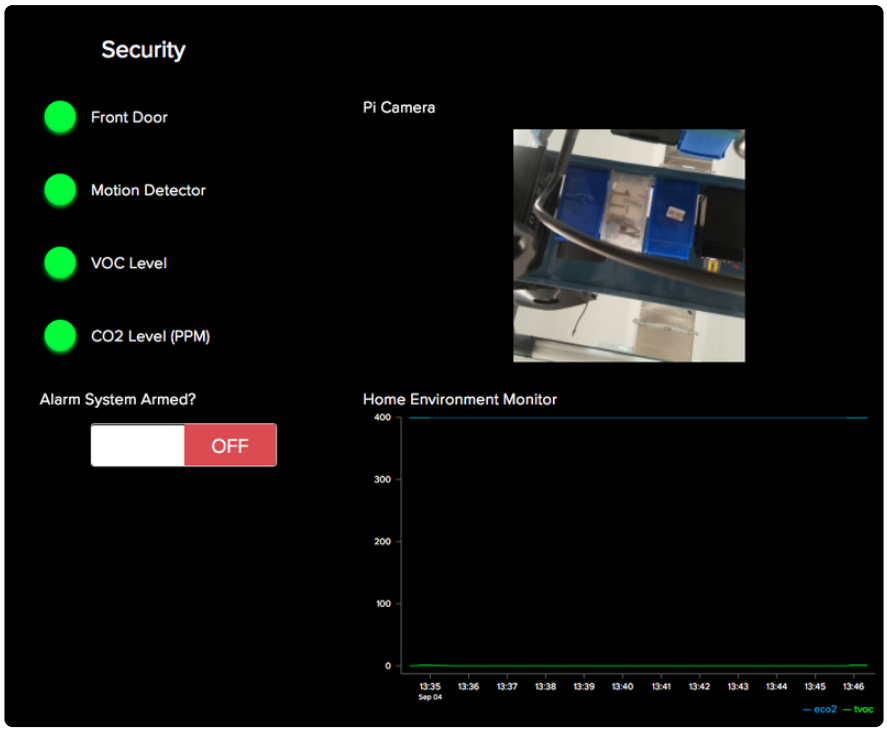



Switch the toggle from **OFF** to **ON** to arm your house's security system. When you push the front door open (or move past the PIR sensor past the time set by `alarmHour`), the inside lights will blink red.



Tired of the alarm? Want some peace and quiet? Toggle the alarm **OFF**.


If you set up the Raspberry Pi Camera, you'll also be able to see the camera feed from the Image Block element created in **Python Setup**


Security
Front Door
Motion Detector
VOC Level
CO2 Level (PPM)
Alarm System Armed?
OFF
Pi Camera
Home Environment Monitor
400
300
200
100
0
13:35
Sep 04
13:36
13:37
13:38
13:39
13:40
13:41
13:42
13:43
13:44
13:45
13:46
eco2
tvoc