# RASPBERRY PI SECURITY CAMERA

**The Particle Raspberry Pi project has been discontinued. Future new releases of the Particle Agent software are unlikely.**

## Introduction

Welcome to the official tutorial for building your very own Particle-connected security camera with Raspberry Pi! This project uses the Raspberry Pi integration with the Particle Device Cloud to control a PIR sensor, NeoPixel LED ring, and Raspberry Pi camera to watch for intruders, snap a photo, and upload it to the web via Dropbox.

All firmware and software files for the project are located at the following GitHub repository: https://github.com/particle-iot/particle-pi-camera

## Particle Pi beta program

Please note that the Raspberry Pi integration with the Particle Device Cloud is *currently in beta*. The steps in the Provision your Pi section *will not work* until you have received an email confirmation your active status in the beta program.

Upon open release of the Raspberry Pi + Particle Device Cloud integration, the provisioning instructions will work for everyone. To learn more and to join the beta program, visit our Raspberry Pi + Particle landing page by clicking the button below:

JOIN THE BETA!

## How it works

The project includes a few major sensing components:

- PIR (motion) sensor
- NeoPixel LED rings
- Raspberry Pi camera

The Pi checks the PIR sensor to check for motion within its field of view. If it detects motion, it triggers the LED rings to illuminate the scene, so you can capture intruders in both light and dark environments. The camera then snaps a picture of the intruder, and uses the Pi's connection to the Internet to upload to your personal Dropbox folder.

## What you'll need

Here's a list of the parts you'll need to build your Particle-connected security camera:

**The essentials:**

- Raspberry Pi (v3 preferred)
- Raspberry Camera V2
- PIR (motion) sensor
- NeoPixel LED rings
- Micro USB cable

- Micro SD card

**For the enclosure:**

- Acrylic, MDF or Plywood sheet (3 mm thick)
- M3x12 screws and nuts (4)
- M2.5X12 screws and nuts (4)
- M2x12 screws and nuts (4)
- Access to Laser cutter
- Soldering tools and hardware

# Setting up your Raspberry Pi

## Download and install the Raspberry Pi image

The first thing you'll need to do, if you haven't already, is to create a SD card that you can use to boot up your Raspberry Pi. If you've already set up your Pi, you can skip these steps:

- Make sure your SD card is **FAT32 formatted**
- **Install an operating system** image on the SD card. We recommend Raspberry Pi's preferred operating system, Raspbian Jessie with Pixel, which you can download here.
- **Install the operating system** onto your SD card by following the Raspberry Pi Foundation's official installation instructions, here.

## Connect your Pi to the Internet

There are two primary ways to connect your Raspberry Pi to the web.

### Connect over Ethernet

If your Raspberry Pi has an Ethernet port, connecting it to the Internet is as simple as plugging in a cable to the on-board RJ-45 jack on your Pi. The operating system should automatically work with your router to obtain an IP address and connect to the web.

**Note**: The Pi Zero does not have an on-board Ethernet port, but can be connected with a Ethernet --> USB adapter.

### Connect over Wi-Fi

If you'd like to connect your Pi over Wi-Fi, you can either use the included GUI application to select an available Wi-Fi network, or use the command line tool to manually configure your Pi onto the web.

The official Raspberry Pi Foundation tutorials for connecting to Wi-Fi are available at the links below:

- Using the GUI application
- Using the Command Line

If you're using the command line, you might modify your file to match the example below:

```
network={
    ssid="YourWiFiSSDID"
    psk="YourPassword"
    key_mgmt=WPA-PSK
}
```

You can verify that your Raspberry Pi is online by running the `ifconfig` command from your Pi's command line.

## Install the Particle Agent

Downloading and installing the Particle Pi software is a straightforward, single-step process. After your Pi is connected to the web, simply copy and paste the command below:

```
bash <( curl -sL https://particle.io/install-pi )
```

Your Pi should automatically download the necessary tooling and install the Particle Agent software package. This step may take several minutes, depending on the version of Raspberry Pi hardware you have.

## Provision your Pi

Follow the on-screen prompts to log into your Particle account, provision your Pi on the Particle Device Cloud, and give your brand new Particle Pi hardware a name (or let us generate a goofy one for you).

Once your Pi has been successfully provisioned, you should be able to see it in your device list in our Web IDE and in our Command Line Utility by typing `particle list`. Your Raspberry Pi should be running our default "Tinker" firmware, so you should see the following `Particle.function()`'s exposed through the API:

```
int digitalread(String args)
int digitalwrite(String args)
int analogread(String args)
int analogwrite(String args)
```

## Setup the Pi Camera

Setting up the Raspberry Pi Camera is relatively easy! Before you can start taking pictures, you have to connect the Pi camera to the Raspberry Pi via the camera connector and enable it via `raspi-config`.

You can find more instructions for connecting and setting up your Raspberry Pi's camera in this tutorial from ThePiHut.

The easiest way to validate that your camera is configured correctly is to try capturing an image! You can use the following `raspistill` command to confirm your camera setup:

```
raspistill -vf -hf -o /home/pi/picam/selfie.jpg
```

## Setup the Dropbox API

In order to get the automatic upload to your Dropbox account working correctly, follow this tutorial from Adafruit. You'll have to download and configure a script on your Raspberry Pi, as well as configure an App Key for your Pi to give it upload privileges to access to your Dropbox.

The best way to test the Dropbox uploader by uploading the image above, or any other file on your Raspberry Pi to Dropbox and confirming the file is viewable from your computer or mobile device. You should also make and run a shell script to capture an image and upload it to Dropbox.

You can learn more about writing and running shell scripts on the Raspberry Pi here.

Here's an example script that you can use for your project (note that your directory structure may be slightly different):

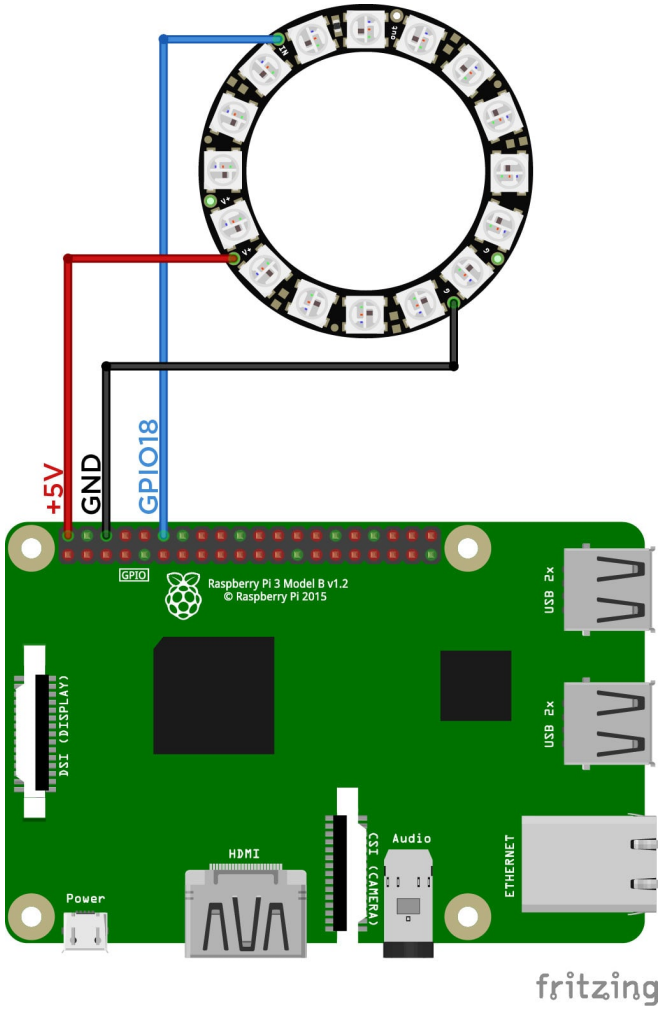
```
#!/bin/bash

echo "running shell script"
DATE=$(date +"%Y-%m-%d_%H%M")
echo "capturing image"
raspistill -vf -hf -o /home/pi/picam/$DATE.jpg
echo "uploading image"
/usr/local/bin/dropbox_uploader upload /home/pi/picam/$DATE.jpg "camera/"
```

## Connect your hardware

The next step of the process is to wire up your hardware, and to use simple example sketches on your Raspberry Pi to confirm that everything is working the way you'd like. The two major components we need to validate are the PIR sensor and the NeoPixel LEDs.

### Connect and test your NeoPixel rings

For this project, we used these awesome NeoPixel LED rings from Adafruit.




Follow these instructions for wiring up the LED rings:

- Connect the positive supply of the ring to +5V on the Pi, GND to GND and input pin of the NeoPixel ring to GPIO18 of the Pi
- Use this modified version of the NeoPixel library, labeled `ws2811`, to control the ring. Note that it is included as a library dependency of the test app below.
- Use the following app to test the ring. You can flash it to your Pi by copying and pasting it into the Web IDE and flashing it to your Raspberry Pi by clicking the star icon next to your device in the "Devices" panel.

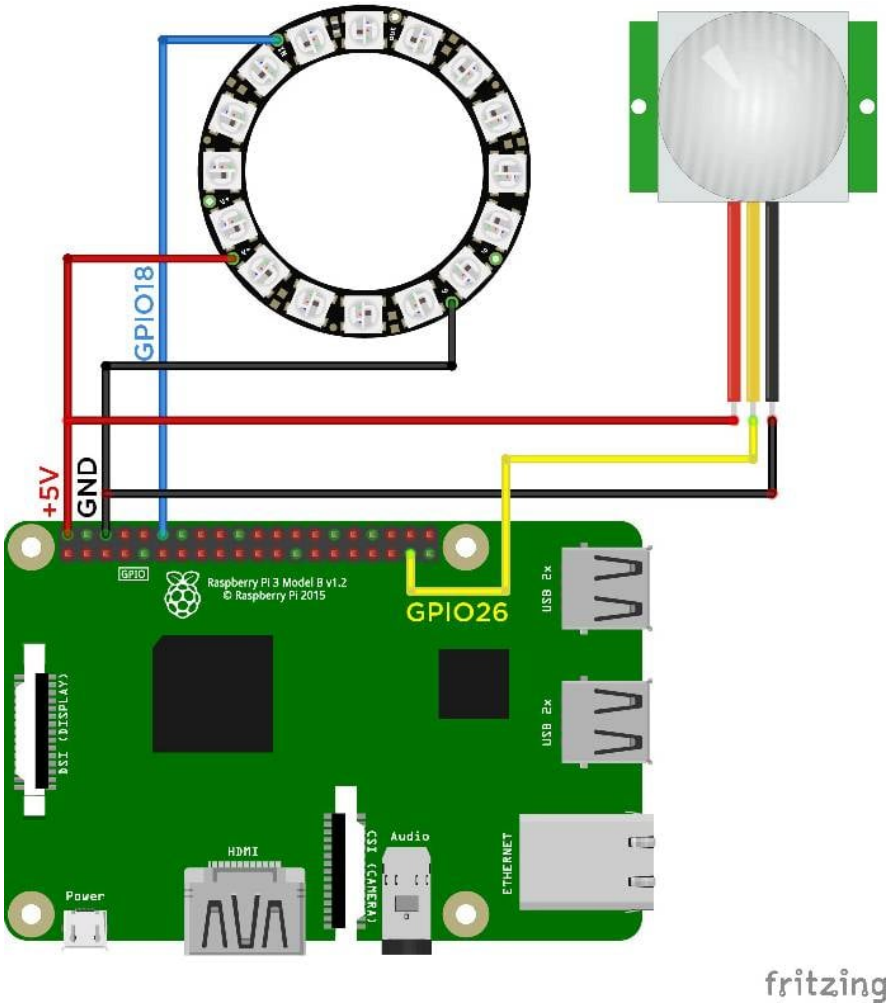
```
#include "Particle.h"
#include "ws2811.h"

#define TARGET_FREQ  WS2811_TARGET_FREQ
#define GPIO_PIN     18
#define DMA          5
#define STRIP_TYPE   SK6812_STRIP_RGBW    // SK6812RGBW (NOT SK6812RGB)
```

```
#define LED_COUNT    16

ws2811_t ledstring = {
    NULL,
    NULL,
    TARGET_FREQ,
    DMA,
    {
        {
            GPIO_PIN,
            0,
            LED_COUNT,
            STRIP_TYPE,
            NULL,
            255,
            0,
            0,
            0,
            0,
        },
        {
            0,
            0,
            0,
            0,
            NULL,
            0,
            0,
            0,
            0,
            0,
        },
    },
};


void rainbow(uint8_t wait);
uint32_t Wheel(byte WheelPos);

void setup()
{
  ws2811_init(&ledstring);
}
```

```
void loop() {
  rainbow(20);
}

uint8_t brightness = 32;
uint32_t color(uint8_t r, uint8_t g, uint8_t b, uint8_t w = 0) {
  return
    ((((uint32_t)w * brightness) >> 8) << 24) |
    ((((uint32_t)r * brightness) >> 8) << 16) |
    ((((uint32_t)g * brightness) >> 8) << 8) |
    ((((uint32_t)b * brightness) >> 8));
}

void rainbow(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256; j++) {
    for(i=0; i<LED_COUNT; i++) {
      ledstring.channel[0].leds[i] = Wheel((i+j) & 255);
    }
    ws2811_render(&ledstring);
    delay(wait);
  }
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  if(WheelPos < 85) {
   return color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos < 170) {
   WheelPos -= 85;
   return color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

## Connect and test the PIR sensor

We will use this particular motion (PIR) sensor for our project from Adafruit. Although the PIR sensor requires a 5V supply, its output is a Pi-friendly 3.3V, so it can be connected directly to a GPIO input.




The NeoPixel ring will light upon detecting motion and will remain off otherwise.

```
#include "Particle.h"
#include "ws2811.h"
#include "stdarg.h"

#define TARGET_FREQ  WS2811_TARGET_FREQ
#define GPIO_PIN     18
#define DMA          5
#define STRIP_TYPE   SK6812_STRIP_RGBW    // SK6812RGBW (NOT SK6812RGB)
#define LED_COUNT    16

ws2811_t ledstring = {
```

```
    NULL,
    NULL,
    TARGET_FREQ,
    DMA,
    {
        {
            GPIO_PIN,
            0,
            LED_COUNT,
            STRIP_TYPE,
            NULL,
            255,
            0,
            0,
            0,
            0,
        },
        {
            0,
            0,
            0,
            0,
            NULL,
            0,
            0,
            0,
            0,
            0,
        },
    },
};

uint32_t color(uint8_t r, uint8_t g, uint8_t b, uint8_t w);


bool pirState = LOW;

void setup()
{
  ws2811_init(&ledstring);
  pinMode(26,INPUT_PULLDOWN);
  for(uint8_t i=0; i<LED_COUNT; i++) {
      ledstring.channel[0].leds[i] = color(0,100,0,0);
    }
```

```
    ws2811_render(&ledstring);
    delay(500);
}


void loop()
{
    //This will run in a loop

    int value = digitalRead(26);

    if (value == HIGH) {    // check if the input is HIGH
      ////Turn on all the neopixles
      if (pirState == LOW) {
            // we have just turned on
            pirState = HIGH;
            for(uint8_t i=0; i<LED_COUNT; i++) {
              ledstring.channel[0].leds[i] = color(250,250,250,250);
            }
            ws2811_render(&ledstring);
            delay(20);
          }
    } else {
      //Turn off all the neopixles
      for(uint8_t i=0; i<LED_COUNT; i++) {
        ledstring.channel[0].leds[i] = color(0,0,0,0);
      }
      ws2811_render(&ledstring);
      delay(20);
      if (pirState == HIGH)
        pirState = LOW;
  }

}

uint8_t brightness = 32;
uint32_t color(uint8_t r, uint8_t g, uint8_t b, uint8_t w = 0) {
  return
    ((((uint32_t)w * brightness) >> 8) << 24) |
    ((((uint32_t)r * brightness) >> 8) << 16) |
    ((((uint32_t)g * brightness) >> 8) << 8) |
    ((((uint32_t)b * brightness) >> 8));
}
```

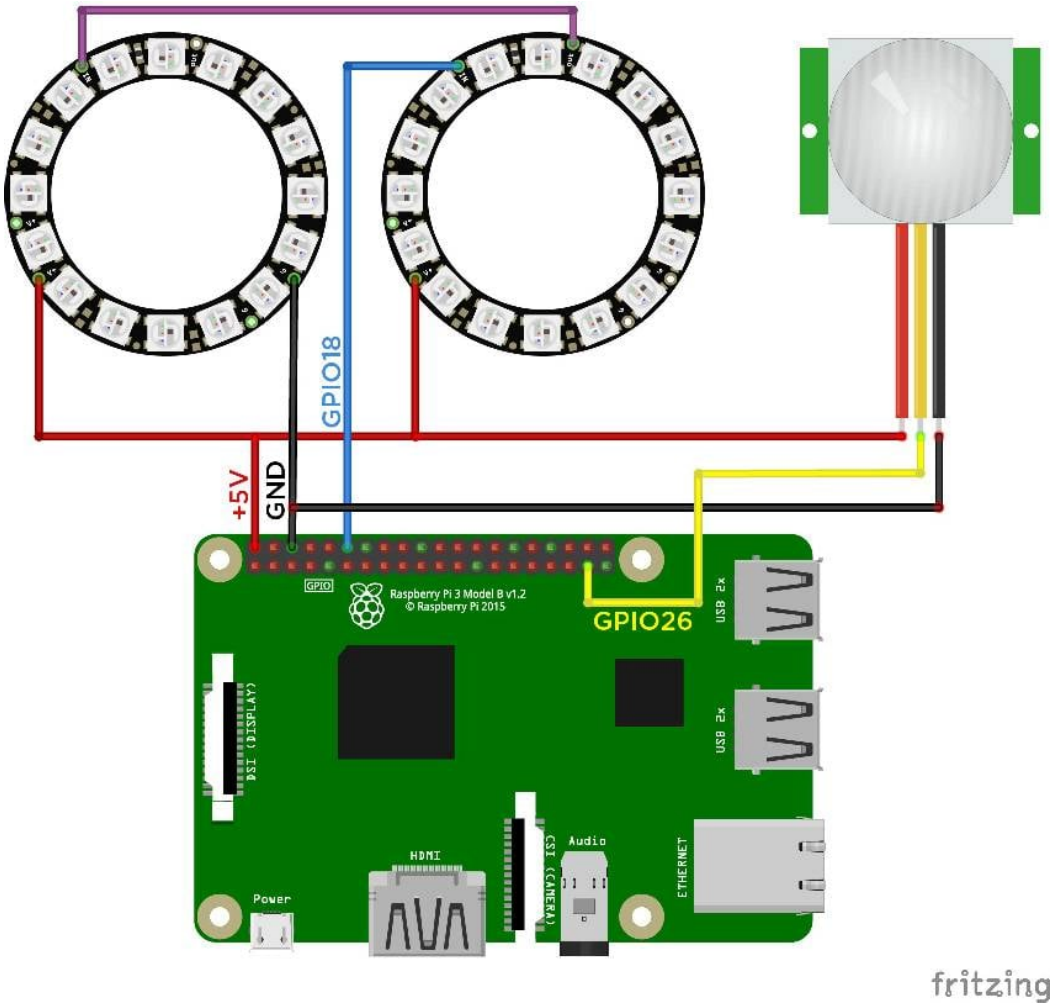
# Putting it all together

## Flash the firmware

Once you've verified that your hardware has been configured correctly, you'll need to flash the final firmware application that interacts with the hardware to your Raspberry Pi. You can view and download that application from our GitHub page for the project, located here:

https://github.com/particle-iot/particle-pi-camera/blob/master/firmware/application.cpp

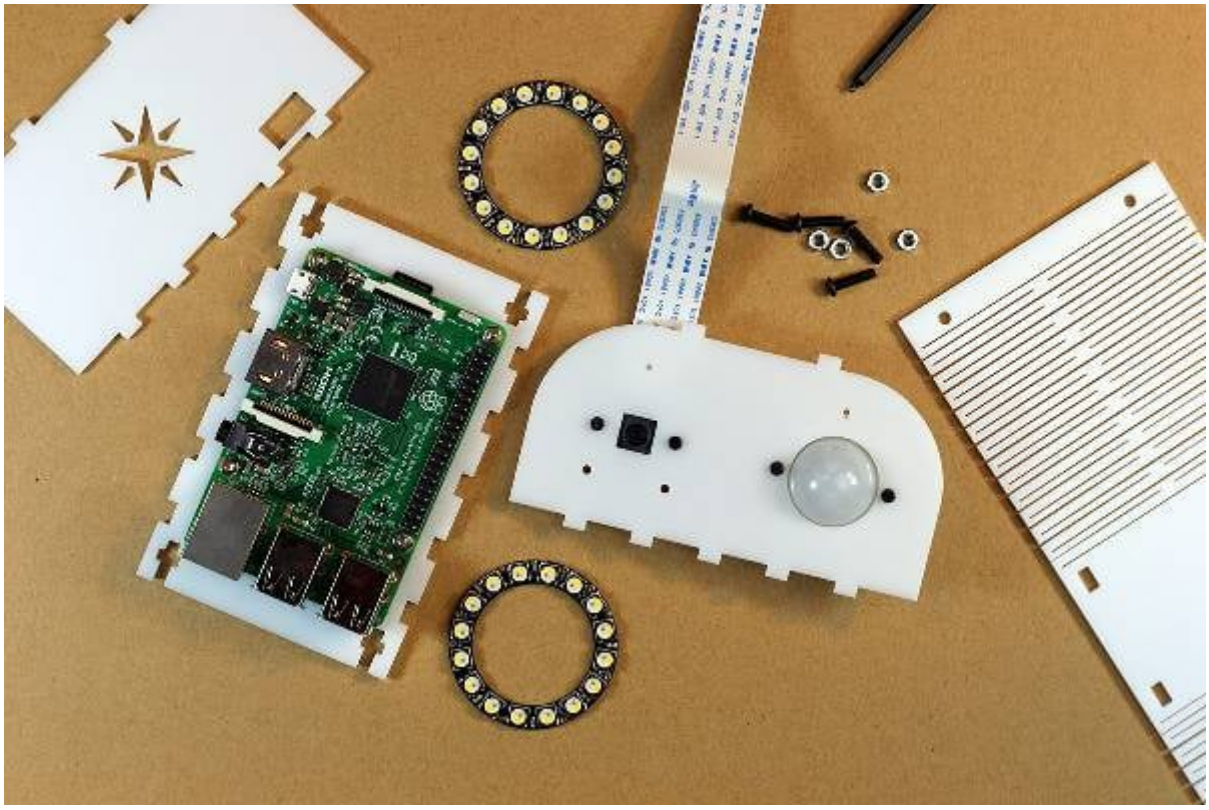
Here is the schematic of the complete project. Two NeoPixel rings are connected in series with the data out of the right ring connected to the data input of the left ring. The output of the PIR sensor is connected to GPIO26. Remember to plug in your pi-camera as well!



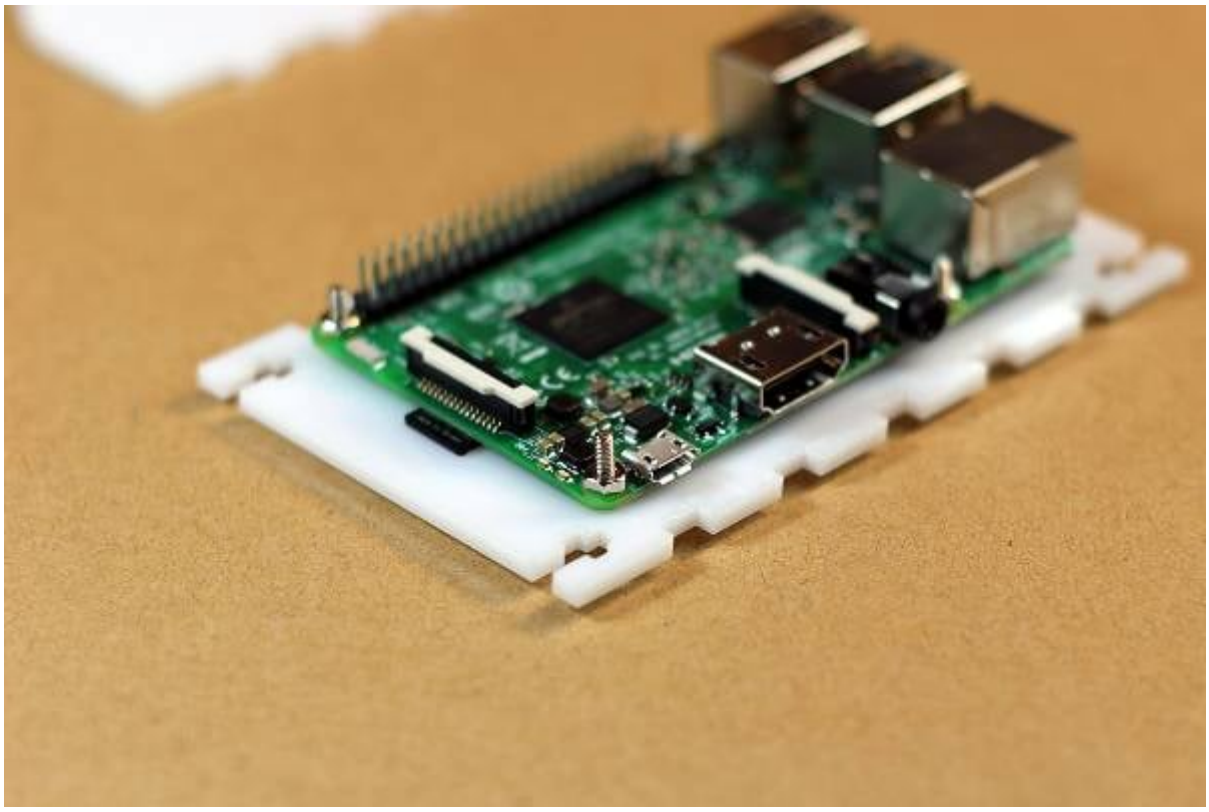
## Assemble the enclosure (optional)

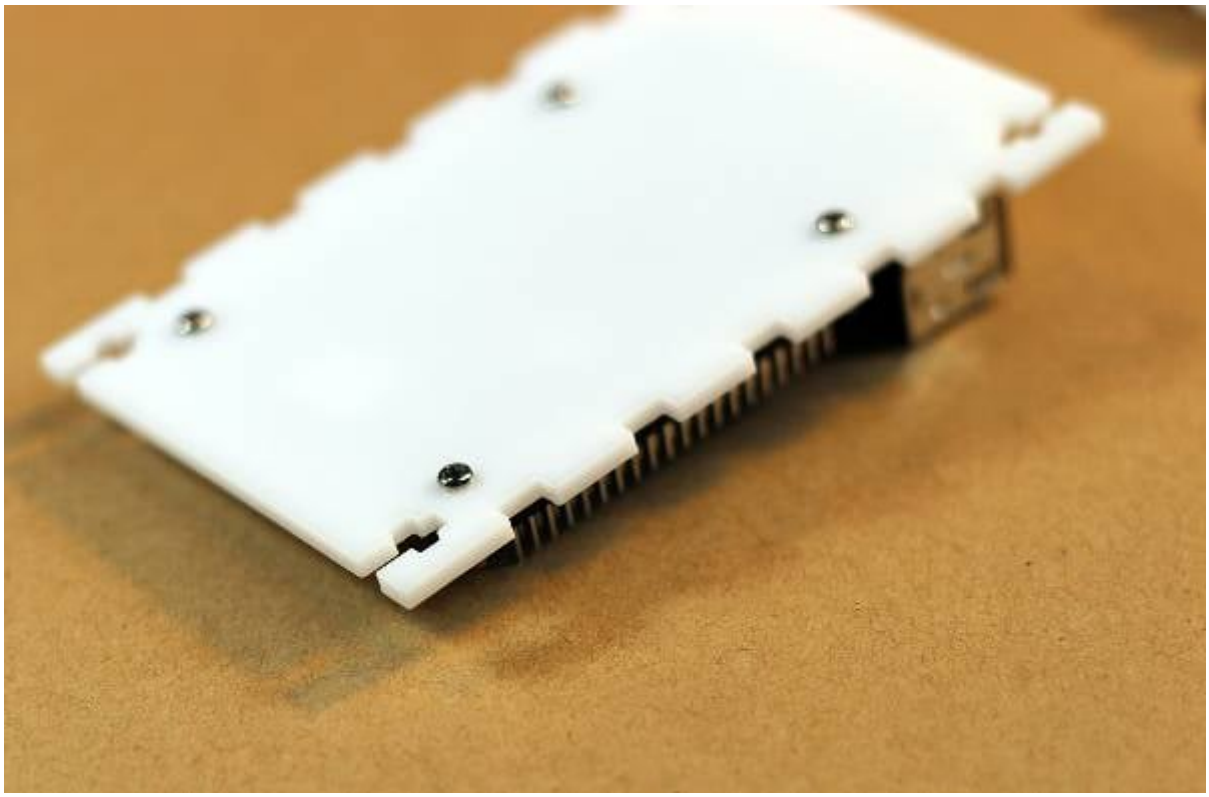
These instructions are for the assembly of an optional enclosure that you can build if you have access to a laser cutter. All of the pieces are cut from a 3mm white acrylic sheet, but you can feel free to use MDF or plywood instead.

You can download an illustrator file for the laser cutter here.



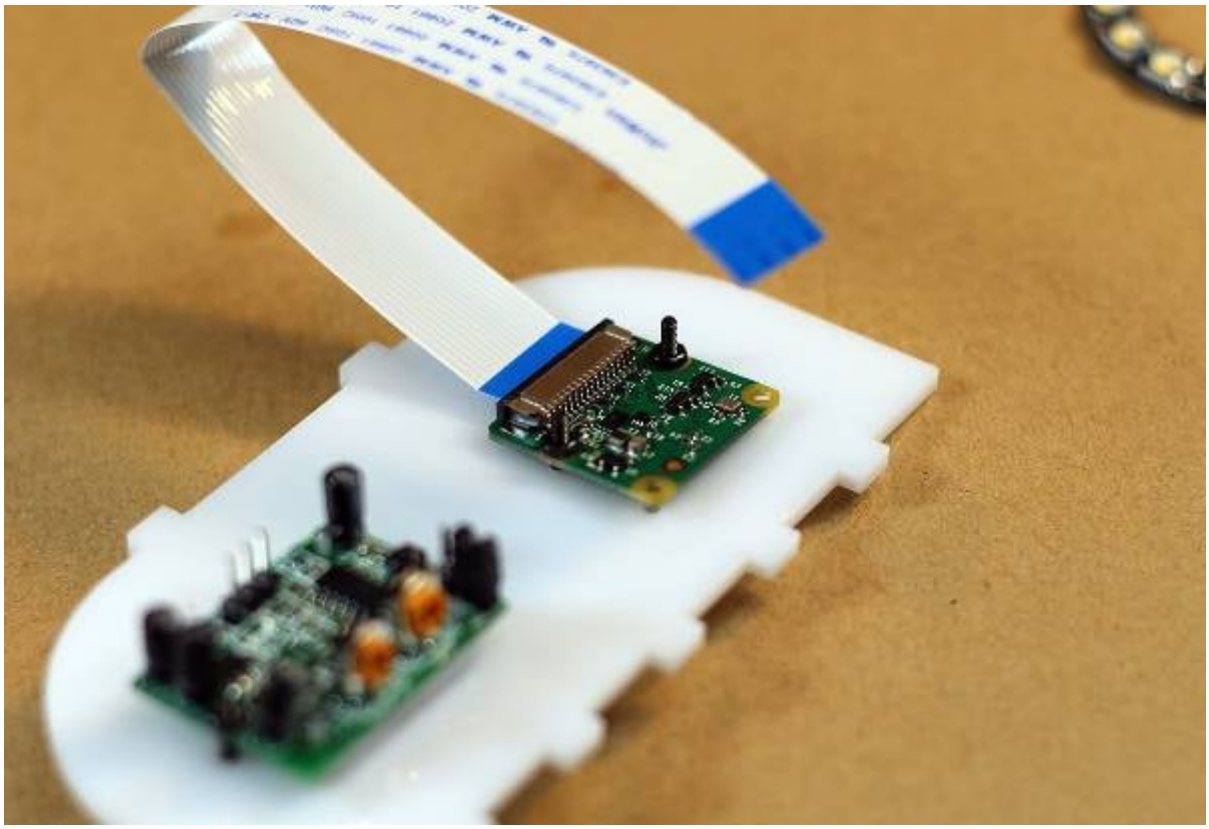
1. Attach the Pi to the base plate using M2.5 screws and nuts

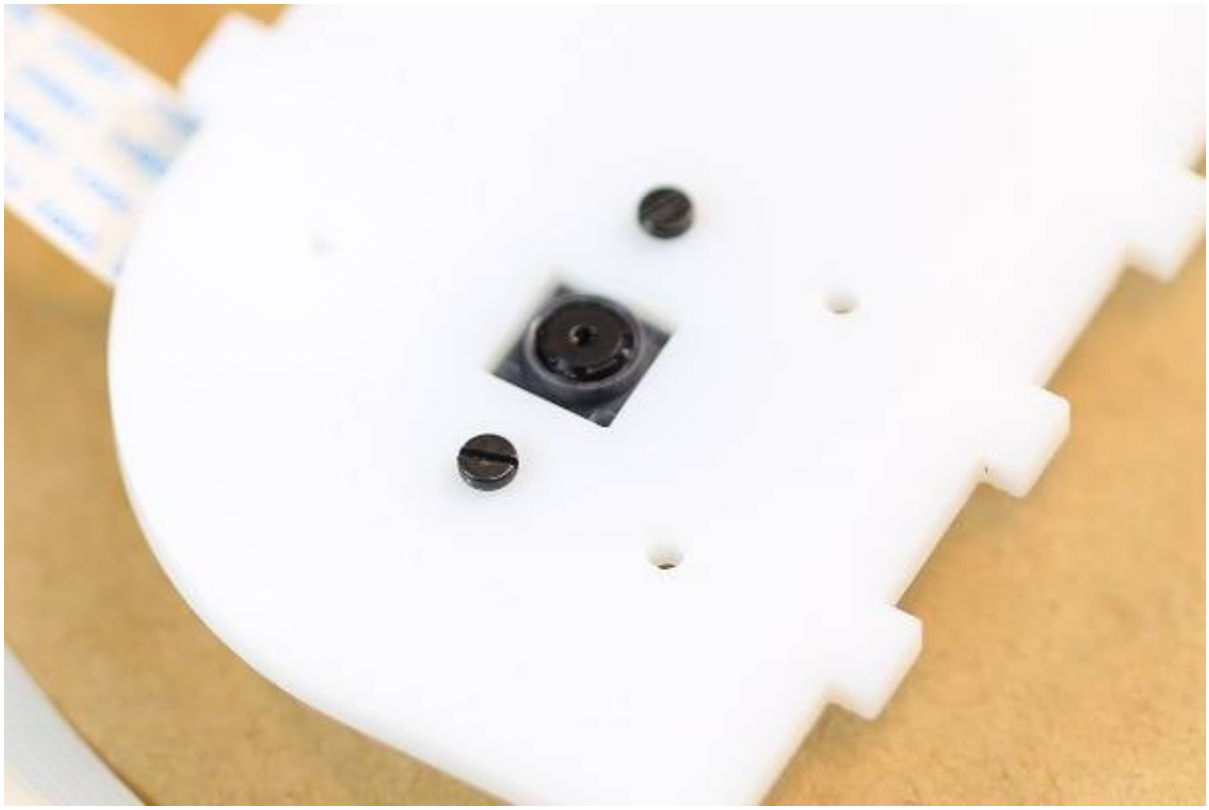




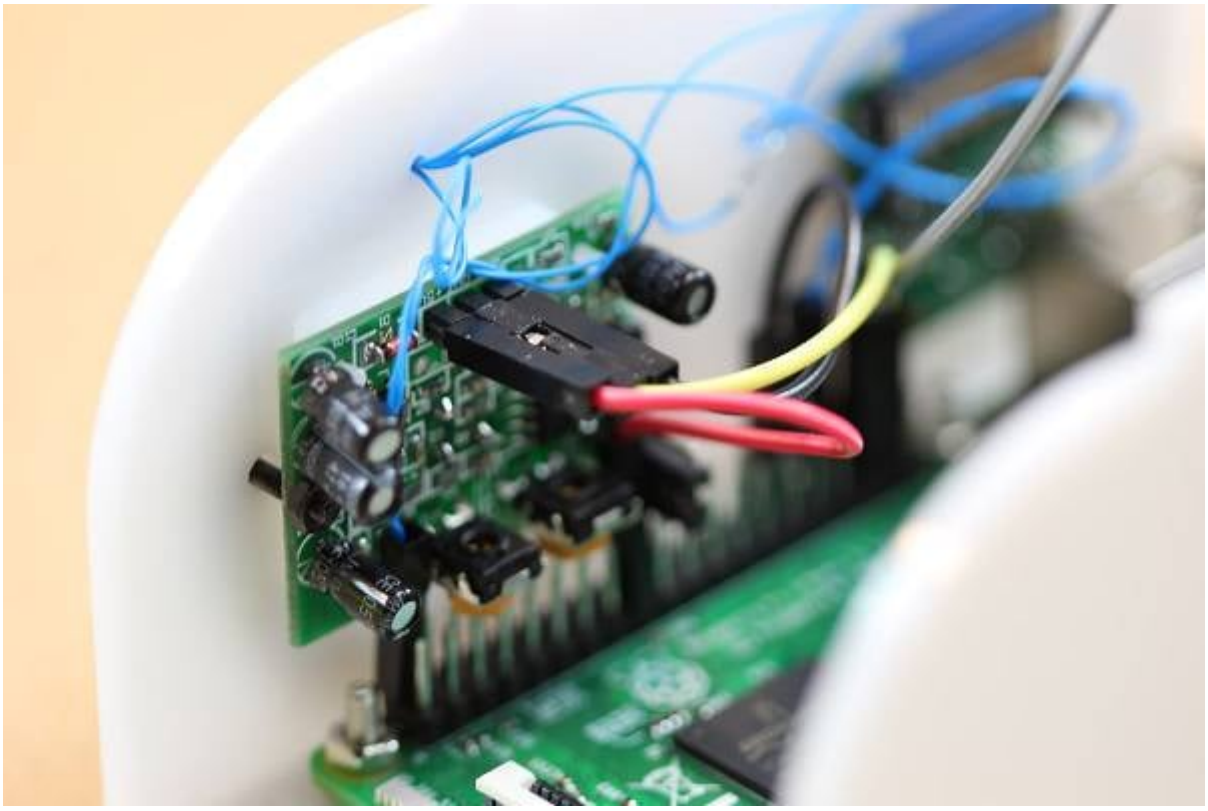
2. Attach the Pi camera and the PIR sensor using M2 screws and nuts

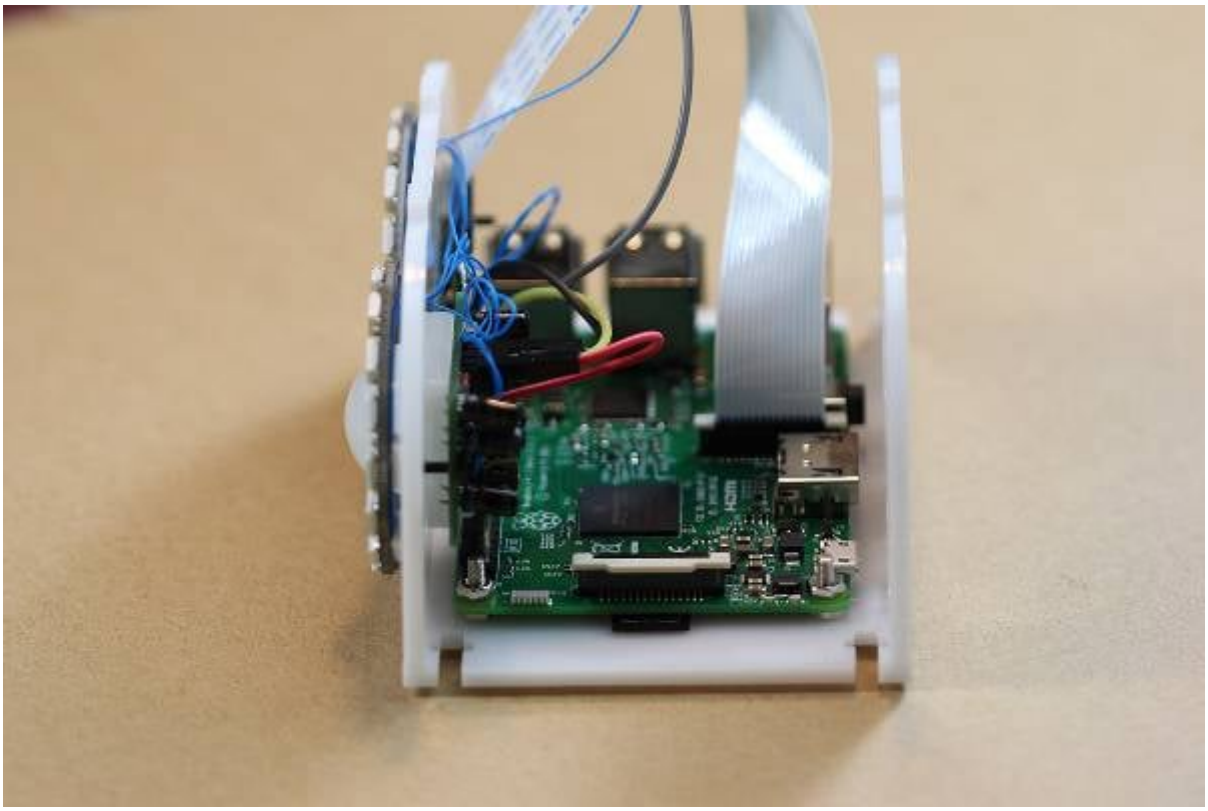

SENSITIVE
TIME

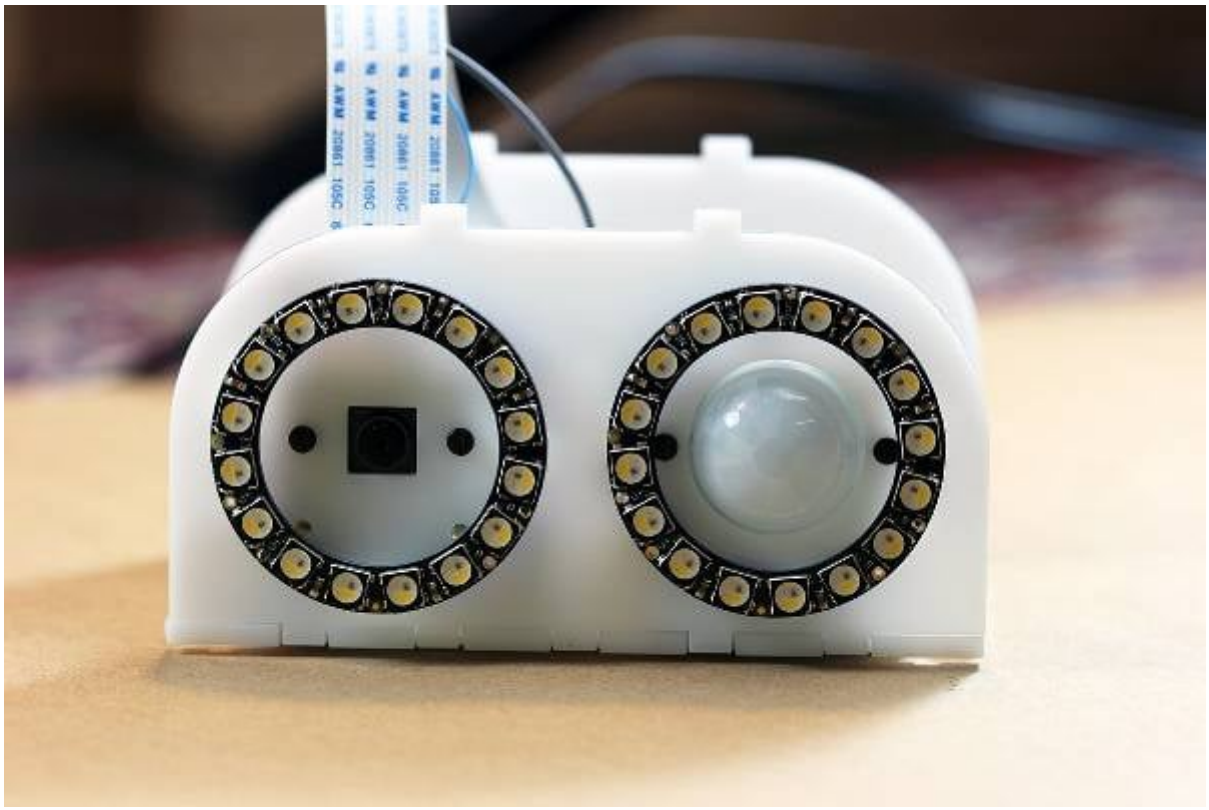




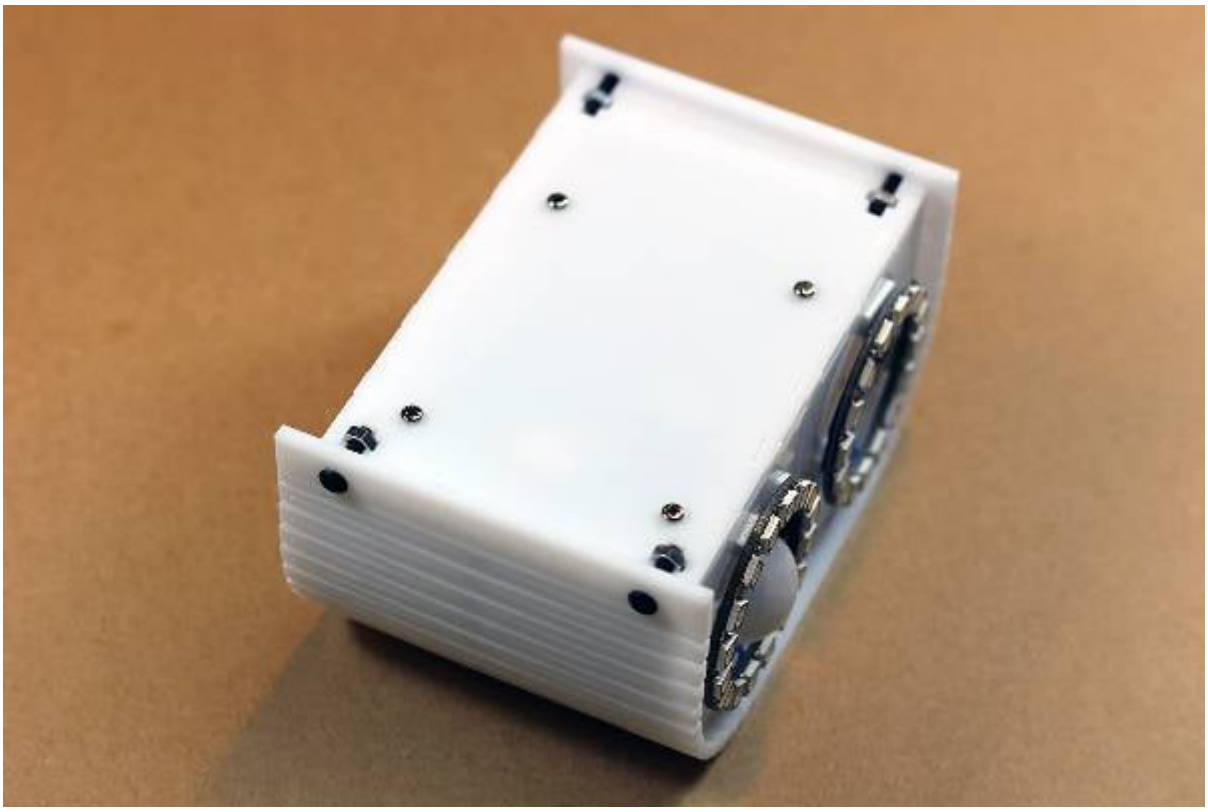
3. Glue the panels to the base plate using super glue or other fast curing epoxy glues.

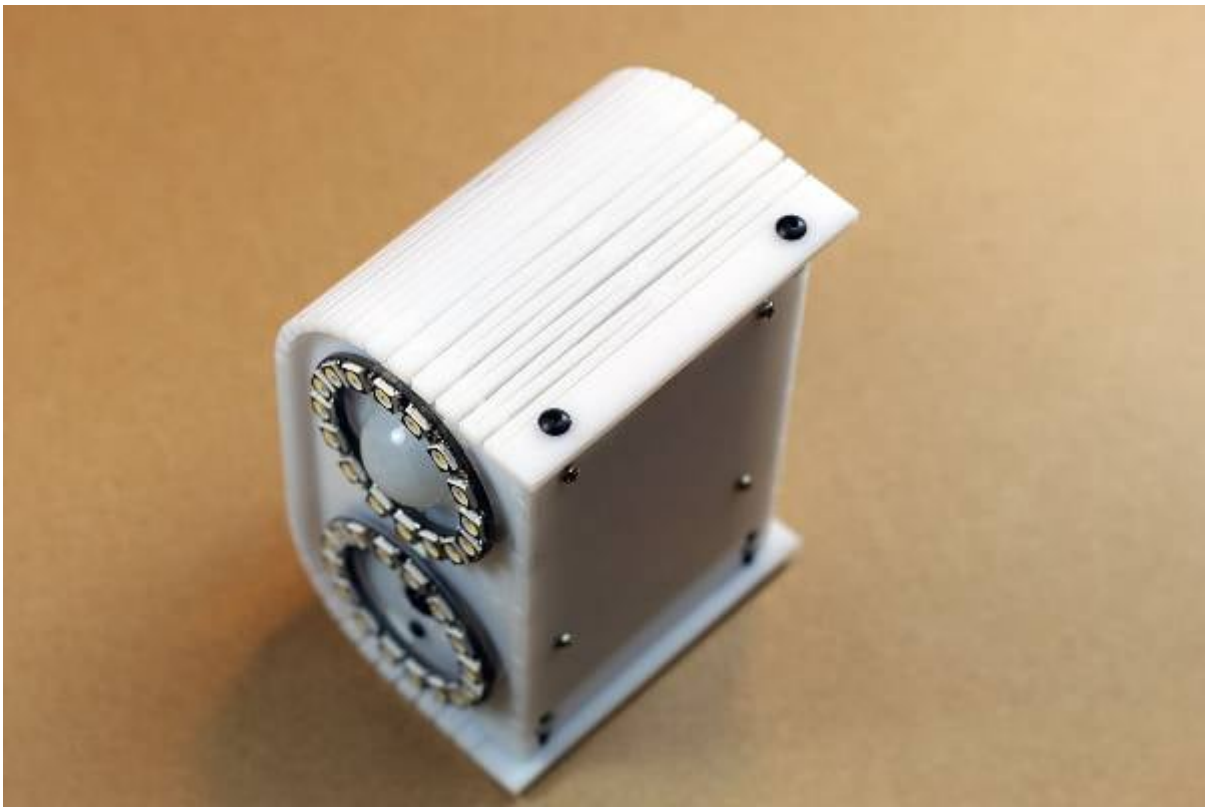






4. Attach the top plate and secure it to the bottom using M3 screws





5. Power it up with a USB cable

6. Mount it on a tripod and you are good to go! I glued a GoPro tripod mount bracket to the base of the enclosure.