**We’re going to make our own trailcam, using a Raspberry Pi Zero W. This guide is from** [**https://peaknature.co.uk/blog/how-to-build-a-raspberry-pi-trail-cam--part-1-introduction**](https://peaknature.co.uk/blog/how-to-build-a-raspberry-pi-trail-cam--part-1-introduction) **although I have modified it a wee bit.**

**You’ll need:**

* Raspberry Pi (we’re using the Raspberry Pi Zero W)
* micro-SD card with Raspbian OS installed
* PIR sensor
* Pi Zero Cam (either standard, or NOIR, for shooting at night)
* 3 x jumper wires
* USB drive
* Power supply - we’re using mains adapter whilst we’re building the trailcam and a portable power-bank when its in the field.

**1.) Accessing the Pi**

Insert the micro-SD card into the Pi, and the power supply into the micro-USB port labelled **PWR.** You should see a little green light come on.

If you wanted to, you could connect the Pi to a monitor, keyboard, and mouse, and then use it like a normal PC. However, we’re going to do it the hard (but cheap) way. Accessing the Pi without any peripherals attached is called going ‘headless’.

To access the Pi headlessly, we need something called an SSH (Secure Shell) client on our computer. If you have a Mac, you already have one - it’s called Terminal. If you have a Windows machine, you’ll need to download one. The go-to SSH client for Windows is called PuTTY, and you can get it here: <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

On your laptop, open your SSH client. This has a ‘command line’ interface. It might look a bit scary at first, but it is your friend.

Your Pi has a name either the one your gave it or raspberrypi (default).

**Type in command line (change YOUR\_PI\_NAME to whatever your Pi is called):**

ssh pi@YOUR\_PI\_NAME.local

It will ask for a password. The default password is “raspberry”. Change the password to one you can remember.

**2.) Updating software and enabling the camera**

Before we do anything, it is a good idea to update the software on the Pi. Since ours are new, it probably won’t take too long - a couple of minutes, hopefully. The command ‘sudo’ for some reason gives us the authority to make major changes to the Pi, a bit like having system admin rights on a shared computer.

**Type/paste in command line (one at a time) then press ENTER:**

sudo apt-get update

sudo apt-get upgrade -y

Now we need to install some libraries to let us write our trailcam program. This might take a little while - maybe 10 minutes in total.

**Type/paste in command line (one at a time - note there are 3 lines here, the first one is a bit long!) then press ENTER:**

sudo apt-get install python3 python3-dev git i2c-tools python3-pip python3-gpiozero python-gpiozero python-picamera -y

sudo pip3 install gpiozero picamera RPi.GPIO spidev

sudo apt-get install gpac -y

Finally for the initial set-up, we need to enable the Pi Cam:

**Type in command line:**

sudo raspi-config

This will open a GUI. Use the keyboard to select [5. Interfacing Options], and then hit **ENTER** to select Camera, select ‘YES’ and click **ENTER** to enable. Then select finish. It will ask you if you want to reboot. Select ‘NO’.

Now we’ll turn off the Pi.

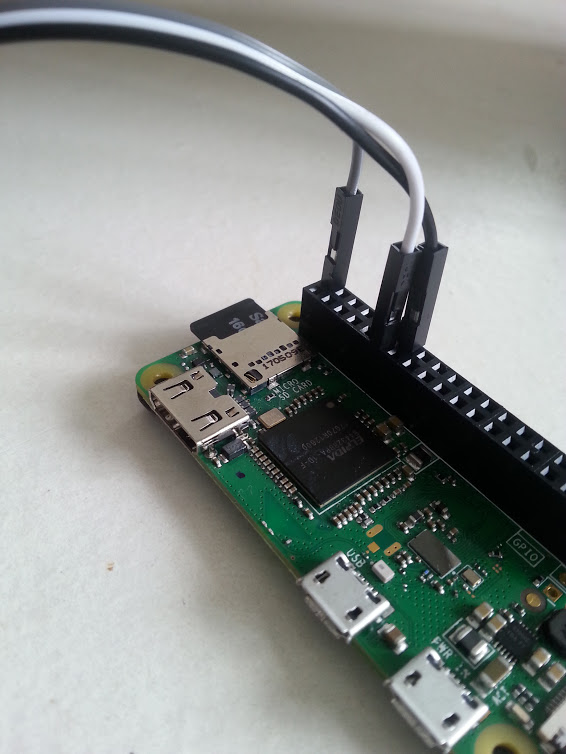
**Type in command line:**

sudo halt

Once the green light has stopped blinking, disconnect the power supply from the Pi.

**3. Attach the hardware**

Ok, good work, now let’s stick some stuff on it. We’ve only got two things to attach, so it should be a breeze. First, let’s attach the PIR sensor:



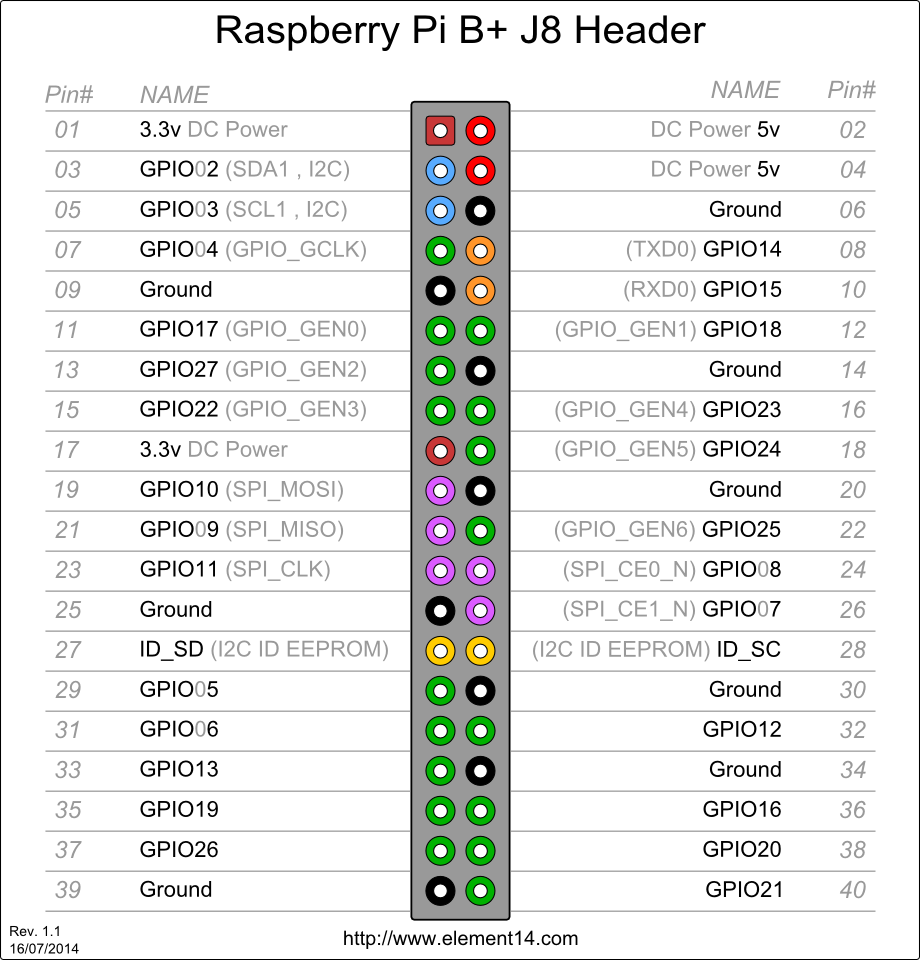
Connect three jumper wires to the three pins on the bottom of the PIR sensor (left pic). To keep things tidy, you can keep the three wires connected to each other rather than separating them.

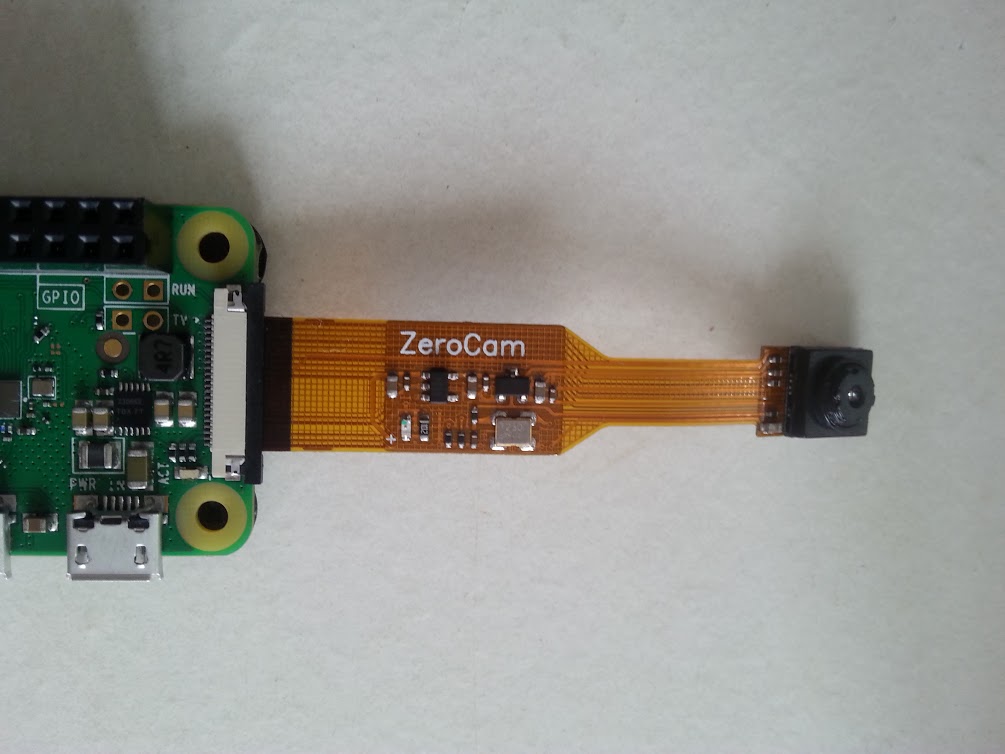
Make sure you connect the wires to the correct GPIO pins on the Pi (General Purpose In/Out):

Connect as follows: Power (grey): Pin 2

Signal (white):Pin 11

Ground (black): Pin 14



Now \*carefully\* attach the camera to the camera port. It should just slot in, but it can be a bit fiddly.

**4. Write the trailcam program**

Great! Now we have our camera and sensor attached, so we should make and test our trailcam program in Python. Power up your Pi and SSH into it as in step 1. We’re going to create an empty Python file for our program.

**Type in command line:**

nano trailcam.py

Nano is a simple text editor for writing code. To create the trailcam program, copy and paste all the stuff between the dashed lines below. Have a read through the code and see if you can follow the annotations (in red) and see what it does:

--------------------------------

**# Import the libraries needed for the program**

**from** gpiozero **import** MotionSensor

**import** logging

**from** datetime **import** datetime

**from** subprocess **import** call

**import** picamera

**import** time

**import** os

**# Create somewhere to log file data**

logfile = **"/home/pi/trailcam\_log/trailcam\_log-"**+str(datetime.now().strftime(**"%Y%m%d-%H%M"**))+**".csv"**

logging.basicConfig(filename=logfile, level=logging.DEBUG,

format=**'%(asctime)s %(message)s'**,

datefmt=**'%Y-%m-%d, %H:%M:%S,'**)

**# Tell the Pi what the PIR sensor is**

pir = MotionSensor(17)

**print**(**'Starting'**)

logging.info(**'Starting'**)

**# Wait an initial duration to allow PIR to settle**

time.sleep(10)

**# The actual code, telling the Pi to start recording**

**# when it gets a signal from the PIR, record for a set**

**# duration, then save it.**

**while** True:

pir.wait\_for\_motion()

logging.info(**'Motion detected'**)

**print**(**'Motion detected'**)

**while** pir.motion\_detected:

**print**(**'Taking photo'**)

ts = **'{:%Y%m%d-%H%M%S}'**.format(datetime.now())

logging.info(**'Taking photo: '**+ str(ts)+**'.jpg'**)

**with** picamera.PiCamera() **as** cam:

cam.resolution=(1024,768)

cam.annotate\_background = picamera.Color(**'black'**)

cam.start\_recording(**'/home/pi/video.h264'**)

start = datetime.now()

**while** (datetime.now() - start).seconds < 10:

cam.annotate\_text = **"BES 2018 "**+datetime.now().strftime(**'%d-%m-%y %H:%M:%S'**)

cam.wait\_recording(0.2)

cam.stop\_recording()

time.sleep(5)

timestamp = datetime.now().strftime(**'%d-%m-%y\_%H-%M-%S'**)

input\_video = **"/home/pi/video.h264"**

logging.info(**'Attempting to save image'**)

**if** os.path.isdir(**'/mnt/usb1/videos'**):

logging.info(**'Saving to /mnt/usb1/videos/'**)

output\_video = **"/mnt/usb1/videos/{}.mp4"**.format(timestamp)

**else**:

logging.info(**'Saving to /home/pi/videos/'**)

output\_video = **"/home/pi/videos/{}.mp4"**.format(timestamp)

call([**"MP4Box"**, **"-add"**, input\_video, output\_video])

time.sleep(10)

**print**(**'Motion Ended'**)

logging.info(**'Motion Ended'**)

**----------------------------------**

To save our program and exit, press **CTRL X*,*** then type **y**, then press **ENTER.**

Note that at the moment, our program films for 10 seconds once the PIR detects movement. You can change this to a longer time if you like. The program ‘sleeps’ for 5 seconds after recording. This is to give the Pi time to save the file - don’t make this value much smaller, or it may start filming again before it finishes saving the file, which means we’ll lose the video.

**5. Set directories for saving the videos**

In our program, we save our videos to a USB drive if one is available, otherwise it saves onto the SD card. We need to make sure our Pi knows where the USB drive is, and also make some of the directories (folders) that the code calls upon, and set the permissions which allow the program to write into them.

First make the directory for the logging files:

sudo mkdir /home/pi/trailcam\_log

Set ‘permissions’ for the directory:

sudo chown -R pi:pi /home/pi

Now plug your USB drive into the socket labelled **USB**, using the adapter.

We’re going to look for the details of the USB drive, so we can tell the Pi where to keep the photos.

sudo blkid

There should be some details of your USB drive. It’s probably in the port called /dev/sda1. Make a note of the UUID - mine was 7245-EF1E.

Next we’re going to create a directory to store our pictures in on the USB drive, then we’re going to tell the Pi where that USB drive is.

sudo mkdir /mnt/usb1

sudo mkdir /mnt/usb1/videos

Set ‘permissions’ for the directory:

sudo chown -R pi:pi /mnt/usb1

This bit makes it so that the drive automatically mounts when plugged in. First, open this file from the command line:

sudo nano /etc/fstab

Add this line to the bottom of the file. You’ll need to change the UUID to the one for your USB that you carefully made a note of earlier:

UUID="7245-EF1E" /mnt/usb1 vfat auto,users,rw,uid=1000,gid=100,umask=0002 0 0

Save and exit: press **CTRL X*,*** then type **y**, then press **ENTER.**

**6. See if it works**

Nearly there! Let’s reboot the Pi and then test our program!

sudo reboot

You’ll have to SSH back into the Pi again, as we did in Step 1, from the command line. Once you’re in, try running the program.

**Type in command line:**

python trailcam.py

This command tells the Pi to open our program file using Python. The program should start, and should begin recording when you wave your hand in front of the sensor. You should see a light go on on the camera when it’s filming.

If it doesn’t work, don’t worry, the Pi will spit out something called ‘Traceback’ which will help us (along with trusty old Google) to pinpoint where and what went wrong.

It should tell you in the command line when a video file has been saved to the USB. Press **CTRL Z** to stop the program from running - make sure you do this when the red camera light is OFF. If the red light is on, then the camera will stay on and you’ll have to reboot the Pi before running the program again. Remove the USB drive from the Pi and check to see whether the files have been saved.

Apparently, you can adjust the sensitivity of the PIR by playing with the little orange dials. One dial (or potentiometer, if you want to be pedantic) adjusts the delay between detecting movement and sending the signal, and the other adjusts the sensitivity of the sensor. I think the former is on the left (if you are looking at the bit where the screwdriver goes in, and all the components are sticking towards you) and the latter on the right. Have a fiddle with them if you like: see how little you have to move/how close you can get to the sensor before the light comes on. Then we can think of the best way to set them in the field.

**7. Make the trailcam run automatically**

We won’t be able to tell the Pi to run the trailcam program when we’re in the field, so we need to set it up so that it starts automatically when the Pi powers up. Don’t worry, this is the final bit!

Create a new file which loads when the Pi boots:

sudo nano /lib/systemd/system/startcam.service

This file tells the Pi to execute a command (load python, then load our trailcam program) after everything else has loaded (that’s the ‘multi-user.target’ bit in the below). Copy and paste the following code into the file:

[Unit]

Description=Start Trailcam on Boot

After=multi-user.target

[Service]

Type=idle

ExecStart=/usr/bin/python /home/pi/trailcam.py

[Install]

WantedBy=multi-user.target

Save and exit: press **CTRL X*,*** then type **y**, then press **ENTER.**

Give this its permission:

sudo chmod 644 /lib/systemd/system/startcam.service

Finally, we need to ‘enable’ our command that starts after startup. I don’t know why we have to give it permission AND enable it - sometimes computers lack a bit of initiative:

sudo systemctl daemon-reload  
sudo systemctl enable startcam.service

Power down your Pi:

sudo halt

That’s it! Congratulations, you’ve just made a fieldcam! Now you just need to make a case to keep the weather out...