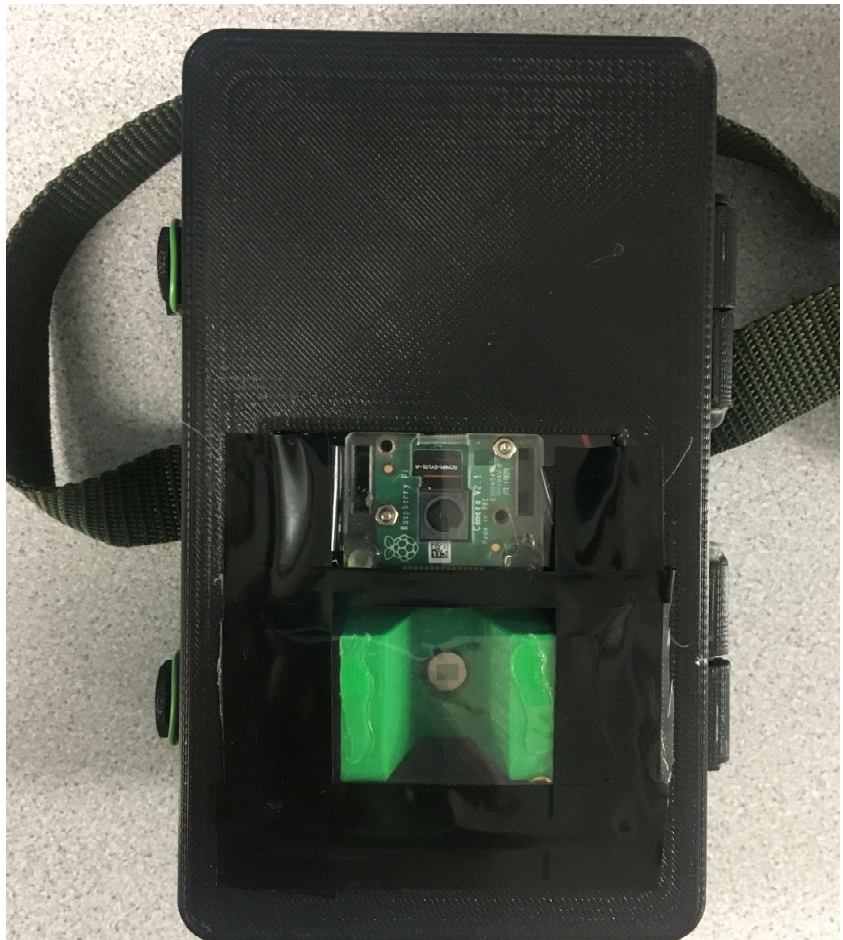


Integrated Camera Trap

Installation Guide and Instruction Manual



A Wi-Fi connected Raspberry Pi
Camera Trap Network

Author: Anamitra Datta
Updated: Dylan Logan

Contents

1. Introduction	3
a. Background	3
b. Network Design	3
c. Hardware	4
d. Software and Networking	4
2. Unix	5
a. Introduction to Unix	5
b. Basic commands	5
i. ls	5
ii. cd	5
iii. mv	5
iv. cp	5
v. rm	5
vi. mkdir	5
vii. pwd	5
Permissions	6
viii. chmod	6
ix. chown	6
c. Processes	6
i. ps	6
ii. kill	6
d. Power	6
i. reboot	6
ii. poweroff	6
e. Startup services	6
i. rc.local	6
ii. Systemd	7
iii. Cron	7
3. Installation and Setup	8
Camera Trap Raspberry Pi	8
a. Supplies	8

b. OS Installation.....	9
c. Setting up the Raspberry Pi.....	10
i. Add components and start up OS.....	10
ii. Wi-Fi Setup	12
iii. Installation software and enabling interfaces	13
Access Point Raspberry Pi.....	14
i. Supplies	14
ii. Setup and installation.....	15
d. How to run the code	17
e. Setting up mobile app	21
f. Setting up mobile server.....	22
g. Setting up RTC (Real time clock).....	24
h. Setting up the enclosure.....	28
3. Setup from Image Files.....	28

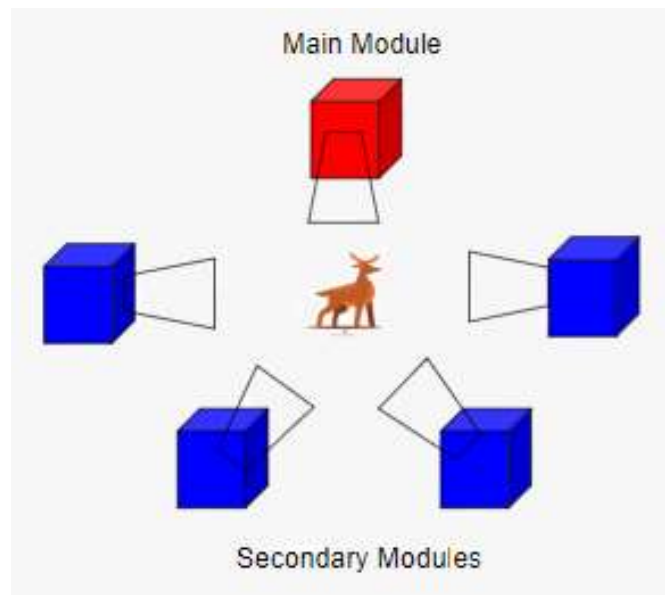
1. Introduction

a. Background

Camera Traps have been used by biologists and wildlife researchers to observe animals through the use of photos and 3D modeling rather than dangerous and disruptive physical techniques such as tranquilization and traps which can disturb and harm animals. However, the problem with current camera traps is that they are neither connected nor synchronized. This becomes an issue when photo sets generated by the camera traps are used for 3D modeling, resulting in poor, disconnected, and fragmented models as a result of desynchronized photos. Our system aims to improve this issue by connecting the camera traps through Wi-Fi and ensuring photo capturing synchronization through timed messages sent over Wi-Fi, telling each camera trap when to take a photo.

b. Network Design

Our network uses a master-slave paradigm in which one “master”/main device sends messages to all the of the “slave”/secondary devices when to take a photo. The figure below shows how to the system is set up.



The camera traps should be arranged in a circle around a single point/area of focus where an animal is likely to appear. There should only be one “master” device in the network and the rest of the devices should be “slave” devices.

c. Hardware

Our camera trap runs on a Raspberry Pi Zero WH (Wireless with Headers) as shown below



A Raspberry Pi is a small single-board computer. This board functions like a normal computer in which you can run software and even install physical components.

More information about the Raspberry Pi can be found on the official Raspberry Pi website:
<https://www.raspberrypi.org/>

d. Software and Networking

The system runs on Wi-Fi through the use of either a router or a wireless access point that can be set up using a Raspberry Pi.

The messages sent within the system are through the use of MQTT (Message Queuing Telemetry Transport). A “broker” is used to send or “publish” messages to all devices listening or “subscribing” to an MQTT “topic” or channel within a network. The broker/publisher and subscribers can be set up using Mosquitto, a free, open-source software for MQTT. More information about Mosquitto can be found on the official Mosquitto website:
<https://mosquitto.org/>

The code running the camera synchronization and photo capturing script is written in Python. More information about Python can be found on the official Python website:
<https://www.python.org/>

2. Unix

a. Introduction to Unix

The Raspbian OS that the Raspberry Pi uses is part of the Linux/Unix OS family. Knowledge of this OS and how to use it is necessary for this system. This chapter covers all of the basic topics of the Raspbian OS that are needed for this system to function.

b. Basic commands

The main way people use a Linux/Unix based operating system is through the command line or terminal. In this section we will cover some basic command line functions so you can get used to working in a Linux/Unix environment. Documentation for Linux commands:

<http://man7.org/linux/man-pages/index.html> . Also type “*man <command name>*” to get information about a particular Linux command.

i. `ls`

This lists the contents of the directory you are currently in or the directory you specify

ii. `cd`

This changes the directory you are currently in. If you want to change to a different directory type this command.

iii. `mv`

Move a file to a destination directory optionally with a different name and delete the old file.

iv. `cp`

Copy a file to a destination directory optionally with a different name and preserve the old file.

v. `rm`

Remove a file or directory

vi. `mkdir`

Make a directory

vii. `pwd`

Print the current working directory you are in.

Permissions

Permissions control who can read, write, or execute a file or directory. Root has the highest permission in Unix subsystems. To be a root user, you must in the sudoers list in the operating system (you are automatically in it by default on Raspbian). Next you must type in the command “*sudo su*” to change to the root user. Sometimes you need a password to become the root user. If you ever need root privilege to execute a command type “*sudo*” before your whole command.

viii. `chmod`

Change read, write, and execute permissions of a file.

ix. `chown`

Change who owns a file or directory.

c. Processes

Programs and other software that run on an Operating System are referred to as processes. When you execute a program or command, it will run as a process. Here are commands that deal with processes:

i. `ps`

Lets you view all process running on the operating system. Alternatives include `top` or `htop`

ii. `kill`

Kill or stop a process from executing

d. Power

i. `reboot`

Restart/reboot the Pi

ii. `poweroff`

Shutdown the Pi.

e. Startup services

In our system, we have programs that must start up immediately when the Raspberry Pi boots up. There are many different ways to start programs up on boot.

i. `rc.local`

This is a file that starts just after all normal services have been started. This is just a bash script file which can run shell commands or any software commands. It is located in the `/etc/` folder.

For example, we can start the mobile app with the command “*python -u ct_mobile_server.py > output.txt 2>error.txt </dev/null &*” and put it in the `rc.local` file. This starts the mobile app

server as a background process and writes the output and errors to separate files on startup, as long as it connected to a proper Wi-Fi network.

ii. Systemd

<https://www.freedesktop.org/wiki/Software/systemd/>

<https://www.raspberrypi-spy.co.uk/2015/10/how-to-autorun-a-python-script-on-boot-using-systemd/>

Systemd is a service manager which can control which process can run on startup and run them as a service file.

To set a python program (for example, the CT slave program) as a systemd service, write the service file with the command *“sudo nano /lib/systemd/ctslave.service”*
(Note: these can also be saved to the /lib/systemd/services directory)

Add in the following to the ctslave.service file and save it:

[Unit]

Description=Camera Trap Slave Service

After=network-online.target

Wants=network-online.target

[Service]

Type=idle

User=pi

ExecStart=python -u /home/pi/ct_client_slave.py

[Install]

WantedBy=network-online.target

Give the file root permissions with the command *“sudo chmod 777 /lib/systemd/ctslave.service”*
Type in the commands *“sudo systemctl daemon-reload”* and *“sudo systemctl enable ctslave.service”* and then reboot the pi. The program should successfully run on startup. You can check using *“sudo systemctl status ctslave.service”*. You can check the output of the service using journalctl.

You can start the mobile app server or the slave or master CT programs using systemd

iii. Cron

<http://man7.org/linux/man-pages/man5/crontab.5.html>

Cron is a software utility which provides a time based job scheduler. You can start scripts or programs at specific times. Here is a guide on how to write a crontab to automatically start a program or command: <https://help.ubuntu.com/community/CronHowto>

3. Installation and Setup

Camera Trap Raspberry Pi

a. Supplies

The supplies we used are listed below, but you can use any components that are similar and fit within the enclosure

PIR Motion Sensor	Adafruit	HC-SR501	\$10.00
Raspberry Pi Zero WH	Raspberry Pi	Zero WH	\$14.00
Raspberry Pi Camera Module V2 (8 Megapixel, 1080p)	Raspberry Pi	V2, 8MP, 1080p	~\$25.00
Portable Charger	Sethruki	20000mAh, T-28	~\$25.00

Raspberry Pi Zero WH (Wireless With Headers)



HC-SR501 PIR Motion Sensor



Raspberry Pi Camera Module V2 (8 Megapixel, 1080p)



Small Portable Charger (small enough to fit in enclosure)



MicroSD card (16-32 GB) and SD Card Adapter



Micro USB to USB adapter or USB Hub (for USB Wireless Mouse and Keyboard)



Mini HDMI to HDMI cable
or Mini HDMI adapter with
HDMI to HDMI cable



3 x Female to Female Jumper
Wires



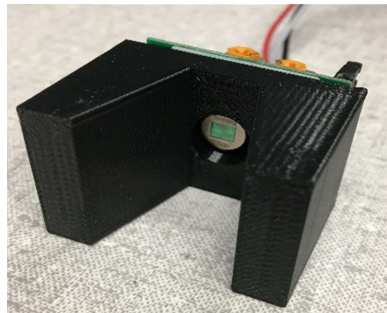
(Power cable) Micro USB
to USB cable



Raspberry Pi Zero Camera
Cable (22 pin to 15 pin)



Sensor Shell



Enclosure



b. OS Installation

Install Raspbian Install Raspbian OS from the Official Raspberry Pi Website :

<https://www.raspberrypi.org/downloads/raspbian/>

We recommend the version with Desktop.

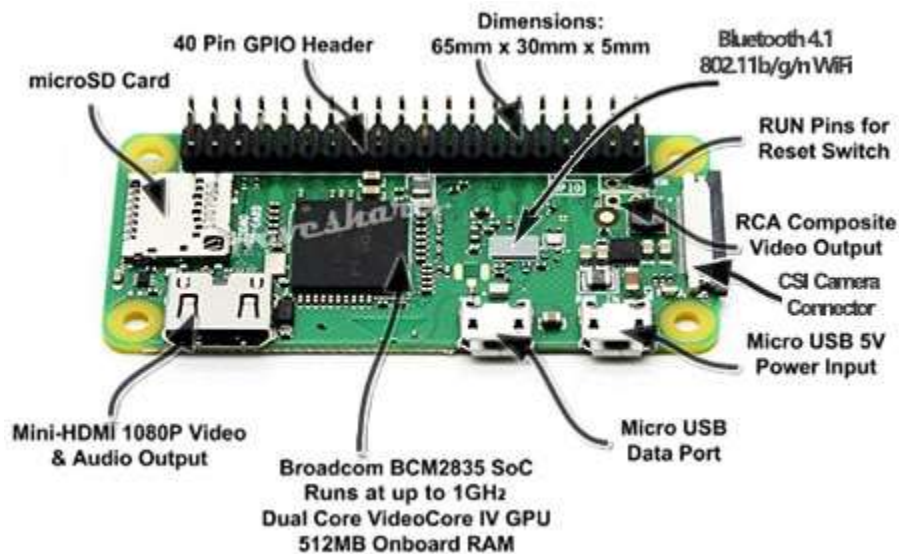
Setup for the Banana Pi is the same just ensure you use the Clean_Banana_Pi.img for the OS.
Also note that the Banana Pi requires HDMI connection to the monitor. In testing using the mini-HDMI adapter to a HDMI worked reliable.

Put the microSD card in the SD Card adapter and put the SD card adapter into your SD card slot on your computer.

First, unzip the downloaded folder to get an .img file of the Operating system you want to install on the Pi. To install the Raspbian OS on the microSD card, we recommend to use Etcher : <https://www.balena.io/etcher/> . Flash the image or .img file onto the SD card and remove the microSD card from the SC card adapter when the installation is complete.

c. Setting up the Raspberry Pi

i. Add components and start up OS

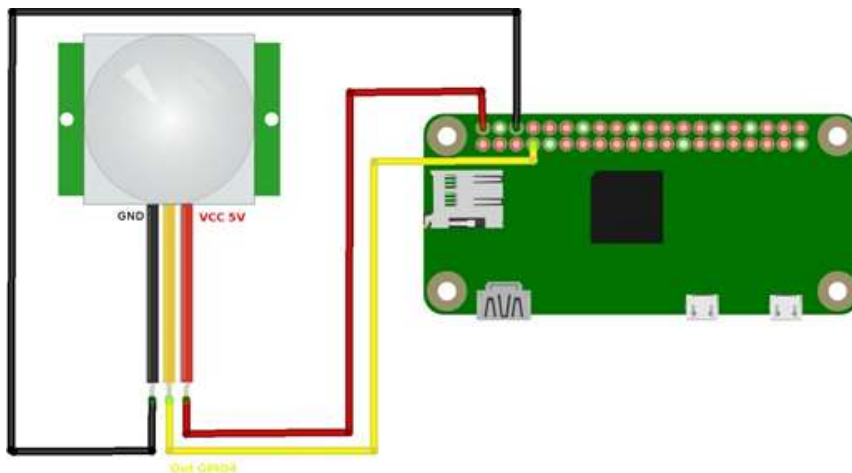


Place the microSD card into the microSD card slot on the Raspberry Pi

Put the 22 pin side of the wire on the bottom slot of the camera module and the 15 pin side of the wire in the Raspberry Pi CSI camera connector on the end of the Pi. Make sure the dark or white part of the wire faces up. Make sure the camera is attached as shown below.



Attach the PIR Sensor with the female to female jumper wire cables to the Raspberry Pi Zero as shown below. Make sure the pins match with each other.



Plug in the Micro USB to USB adapter/hub in the Micro USB data port and attach your wireless USB Mouse and Keyboard

Plug in the mini HDMI to HDMI cable or mini HDMI adapter to the mini HDMI port and attach an HDMI cable to a computer Desktop monitor

Finally, plug in the power cable to the Pi in the micro USB power input port and attach it to a power outlet or a small portable charger

A few Raspberry Pi logos, a rainbow-colored screen, and maybe some startup scripts should appear. This means that the Raspbian OS is booting up and you had successfully installed the OS on the Pi. If it does not work, make sure you have followed all of the steps, make sure the Pi works properly and make sure the image is installed on the microSD card correctly and make sure the microSD card is working properly.

The default username is "pi" and the default password is "raspberry"

You can change the password for user "pi" if you want, but it is not necessary. Follow these instructions: <https://www.raspberrypi.org/documentation/linux/usage/users.md>

If you want to change the password, use the command "passwd" or type in "*sudo raspi-config*" and select option *1. Change User Password*

If you are using a Headless Raspbian OS without a GUI/Desktop, edit XKBLAYOUT line in /etc/default/keyboard from "gb" to "us" using nano with the command "*sudo nano /etc/default/keyboard*"

ii. Wi-Fi Setup

Follow these instructions to set up Wi-Fi depending on your setup:

<https://www.raspberrypi.org/documentation/configuration/wireless/>

You need to connect to the Internet for the next few steps.

Desktop:

In the upper right corner, there should be network icon that looks like this:



Click it, and click on the Wi-Fi network you want to connect to and type in the password of the Wi-Fi network. When the icon becomes fully blue, you should be fully connected to Wi-Fi

Command Line:

Type in "*sudo raspi-config*" and select *2. Network Options* and then *2. Wi-Fi*. Type in the ssid of the network you want to connect to and its password. After that, reboot the Pi and you should be connected to the Wi-Fi network you specified.

Or

You can edit the /etc/wpa_supplicant/wpa_supplicant.conf file to add this line at the bottom:

```
network={
    ssid="<name of Wifi Network>"
    psk="<Password for that Wifi Network>"
}
```

Then reboot the Pi and should you be able to connect to the Wi-Fi network specified.

Depending on your Wi-Fi network configurations, you might need to add extra parameters to connect, see the documentation for more details,

http://w1.fi/cgit/hostap/plain/wpa_supplicant/wpa_supplicant.conf -
Documentation for wpa_supplicant.conf

To set up a static IP address for the Raspberry Pi, edit the `dhcpcd.conf` file using the command *"sudo nano /etc/dhcpcd.conf"* and add the following line with the ip address you want to set the Pi to specifically (24 is the subnet mask)

```
static ip_address=192.168.5.1/24
```

iii. Installation software and enabling interfaces

When you are connected to the internet, type in the following in the command line and type "y" when asked to install the following:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install mosquitto mosquitto-clients
sudo apt-get install python-pip
sudo pip install paho-mqtt
sudo apt-get install ftpd
sudo pip install gpiozero
sudo pip install picamera
sudo apt-get install fbi
sudo apt-get install sshpass
sudo apt-get install python-paramiko
```

SSH (protocol to log into Pi remotely) is disabled by default, we need to enable SSH to enable SSH:

1. type *"sudo raspi-config"* in the command line
2. Select *5. Interfacing Options*
3. Select *P2 SSH*
4. reboot the pi by typing *"sudo reboot"* in the command line

SSH is now enabled and you can SSH into the Pi from any device using a software client or through command line (<https://man.openbsd.org/ssh>)

FTP or SCP files from <https://github.com/anamitradata/Integrated-Camera-Trap> for each Pi. We recommend an FTP or SCP client:

WinSCP (Windows only)
Filezilla
Cyberduck

The camera is disabled by default, we need to enable Camera

to enable camera:

1. type "*sudo raspi-config*" in the command line
2. Select *5.Interfaces Options*
3. Select *PI Camera*
4. reboot the pi by typing "*sudo reboot*" in the command line

The Camera is now enabled and working

If you want to test the camera, run `test_camera.py` to see if it can take a photo "test_photo.jpg"

In the main source code folder, type in the command "*sudo chmod 777 ./**" and "*sudo chown pi ./**" to run all files as user pi without root permissions

Make a directory called cameraTrapPhotos in the /home/pi directory.

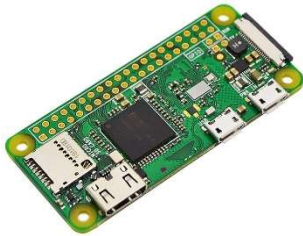
Python should be pre-installed on Raspbian OS. Make sure python or python3 is installed on your OS by typing in "*python --version*", if returns nothing, then you need to install python for Linux <https://www.python.org/downloads/source/>

Access Point Raspberry Pi

In our system, we will be using and setting up a Raspberry Pi Zero W to act as an access point or router in our system so we do not need the use of a physical router or switch in our network.

i. Supplies

Raspberry Pi Zero W or WH



Small Portable Charger (small enough to fit in enclosure)



Mini HDMI to HDMI cable or Mini HDMI adapter with HDMI to HDMI cable



Micro USB to USB adapter or USB Hub (for USB Wireless Mouse and Keyboard)



(Power cable) Micro USB
to USB cable

MicroSD card (16-32 GB) and SD Card Adapter



Set up the Raspbian Operation System with Desktop as you did before for the camera trap Raspberry Pi and connect to the Internet.

We will now set up the Raspberry Pi as an access point (AP).

Here is a guide to setting up the Pi as an access point, which we will cover in this section:

<https://learn.sparkfun.com/tutorials/setting-up-a-raspberry-pi-3-as-an-access-point/all>

All files are also available on the Github if needed for reference under the AP_Files folder.

We will need to use nano (an inbuilt text editor) to change some configuration files. To save a file in nano, press ctrl + o and then ctrl + x to exit the file. Nano documentation:

<https://www.nano-editor.org/docs.php>

When you are connected to the internet, type in the following in the command line and type "y" when asked to install the following:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install ftpd
sudo apt-get install hostapd
sudo apt-get install dnsmasq
```

First make a backup of all files we will be editing with the commands:

```
sudo cp /etc/dhcpd.conf /etc/dhcpd.conf.backup
sudo cp /etc/network/interfaces /etc/network/interfaces.backup
sudo cp /etc/hostapd/hostapd.conf /etc/hostapd/hostapd.conf.backup
sudo cp /etc/dnsmasq.conf /etc/dnsmasq.conf.backup
```

We are going to set a static IP address on our AP Raspberry Pi. Edit the dhcpd.conf file with the command "*sudo nano /etc/dhcpd.conf*". Go to the bottom of the file and add the following line and save the file:

```
denyinterfaces wlan0
```

Next, edit the interfaces file with the command "*sudo nano /etc/network/interfaces*"

Go to the bottom of the file and add the following and save the file:


```

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.5.1
    netmask 255.255.255.0
    network 192.168.5.0
    broadcast 192.168.5.255

```

Next we have to configure hostapd, which allows the Raspberry Pi's WiFi chip to broadcast an SSID and allow Wi-Fi connections on a certain channel

Edit the hostapd.conf file with the command *“sudo nano /etc/hostapd/hostapd.conf”*

Enter the following into the file and save the file:

```

interface=wlan0
ssid=CTWifi
hw_mode=g
channel=7
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=cameratrapp
rsn_pairwise=CCMP
wpa_pairwise=TKIP

```

Feel free to change the ssid and wpa_passphrase to whatever you like. These are your Wi-Fi name and the Wi-Fi password. In our example, we use CTWifi as the SSID and cameratrapp as the password for the CTWifi network.

Edit the hostapd startup script with the command *“sudo nano /etc/default/hostapd”*

Find the line where it says `#DAEMON_CONF=""` and replace it with:
`DAEMON_CONF="/etc/hostapd/hostapd.conf"`

Next we have to set up the DHCP server which hands out IP addresses to different devices connected to the network.

Edit the dnsmasq.conf file with the command *“sudo nano /etc/dnsmasq.conf”*

Add the following to the bottom of the file and save it:

```
interface=wlan0
listen-address=192.168.5.1
bind-interfaces
server=8.8.8.8
domain-needed
bogus-priv
dhcp-range=192.168.5.100,192.168.5.200,24h
```

You can also assign static IPs to specific devices in dnsmasq.conf, for example if the device with MAC address 00:00:5e:00:53:42 was designated as the slave#2, we could set up a static IP for that device by adding the following line to the dnsmasq.conf file:

```
dhcp-host=00:00:5e:00:53:42,ctslave2,192.168.5.102
```

This will have the DNS server assign a name to this device and give a static IP of 192.168.5.102 as well. You can optionally do this for the devices, so you can refer the device as a name rather than an IP address.

You can edit the dhcp range from anything from 192.168.5.2 to 192.168.5.254. But remember, the AP Pi uses 192.168.5.1, so do not assign a Pi with that IP address.

Now reboot your Pi. You can test if the Wi-Fi access point works from your computer. You can connect to it by typing in the password you have in the hostapd.conf file.

You have now successfully built the Access Point Pi for the camera trap network.

You can connect the other Pis to the AP network by either using the WiFi icon/raspi-config and inputting the password or editing the wpa_supplicant.conf file with the ssid and password you chose for the AP.

d. How to run the code

In the main source code folder, type in the command *"sudo chmod 777 ./*"* and *"sudo chown pi ./*"* to run all files as user pi without root permissions. Make a directory called cameraTrapPhotos in the /home/pi directory

Python should be pre-installed on Raspbian OS. Make sure python or python3 is installed on your OS by typing in *"python --version"*, if returns nothing, then you need to install python for Linux <https://www.python.org/downloads/source/>. Make whatever changes to parameters are needed in the code. Type *"python <name of python file>"* to run the code

You must turn on the master Pi before the slave Pis and make sure they are all running on the same network. Run the slave Pis first with the command *"python ct_client_slave.py"*, and then when all the slave Pis are running, run the Master code with the command *"python ct_master_multi.py"*. (Note: the updated slave code is called ct_slave_23.py or dylan_slave.py if it has not been renamed on the pi)

Either run the single sensor photo triggering or the multiple sensor photo triggering. Do not run both at the same time. *(Note: the multisensory version should be used for more reliability)*

There should only be one master running and multiple slaves running. Make sure the slave Pi programs have the IP address of the master Pi in their code and are connected to the network.

In this section, we will show how to run the single sensor camera trap program.

First run the slave Pis. Be sure the master Pi is on and has the software installed. The message below should appear on the slave Pis if everything is set correctly.

```
pi@raspberrypi:~ $ python ct_client_slave.py
Connected with result code 0
Started the photo synchronization program on slave 3 Waiting for a signal from the master module
```

When all the slave devices are on and connected, run the master pi. This message should appear.

```
pi@raspberrypi:~ $ python ct_publish_master.py
Photo synchroniztion program: master device running
Resting the program for a few seconds...
```

After a few seconds, the master Pi will wait for the motion sensor to be triggered. When the motion sensor senses something, it will start the photo session.

```
pi@raspberrypi:~ $ python ct_publish_master.py
Photo synchroniztion program: master device running
Resting the program for a few seconds...
Done resting
False
False
False
```

```
False
False
True
Motion has been detected. Taking camera 1 Photo1
successfully created the directory /home/pi/cameraTrapPhotos/set1/
Camera 1 photo number 1 taken
Photo synchroniztion program: master device running
Resting the program for a few seconds...
```

The slave device will receive a message when motion has been detected from the master Pi and will also take a photo and send it to the master device as shown in the screenshot below. Here we only have one slave device running.

```

pi@raspberrypi:~ $ python ct_client_slave.py
Connected with result code 0
Started the photo synchronization program on slave 3 Waiting for a signal from t
he master module
Received message: Take Synced Photo 1
sucessfully created the directory /home/pi/cameraTrapPhotos/set1/
('Photo Number', 1, ' taken on Slave 3')
Moving from Cam 3 to Master
finished moving photo from Cam 3 to master
closed ftp connection

```

On the master Pi, in the cameraTrapPhotos directory, you should have some photo sets with photos from all cameras running as shown below

```

pi@raspberrypi:~ $ ls -la cameraTrapPhotos/
total 12
drwxrwxrwx 3 pi pi 4096 Mar 29 20:46 .
drwxr-xr-x 8 pi pi 4096 Mar 29 19:40 ..
drwxr-xr-x 2 pi pi 4096 Mar 29 20:46 set1
pi@raspberrypi:~ $ ls -la cameraTrapPhotos/*
total 7720
drwxr-xr-x 2 pi pi 4096 Mar 29 20:46 .
drwxrwxrwx 3 pi pi 4096 Mar 29 20:46 ..
-rw-r--r-- 1 pi pi 3186692 Mar 29 20:46 set1_camera1.jpg
-rw-r----- 1 pi pi 4703980 Mar 29 20:46 set1_camera3.jpg

```

The multiple sensor camera trap program runs the same way.

First, make sure the master device is on and run the slave Pis and the message below should appear if everything is set correctly.

```

pi@raspberrypi:~ $ python ct_client_slave_multisensors.py
Connected with result code 0
Started multisensors photo synchronization program on slave 3, waiting for signa
l from master module

```

Then run the master Pi when all the slave devices have started running.

The command to run the master code is: `python ct_master_multi.py`

The message below should appear

```

pi@raspberrypi:~ $ python ct_publish_master_multisensors.py
Started master multisensor photo sync program
Resting...

```

After a few minutes, the master device should get a response from all slave devices as shown below. Here we only have one slave device running.

```
pi@raspberrypi:~ $ python ct_publish_master_multisensors.py
Started master multisensor photo sync program
Resting...
Starting
Socket is listening
('Got connection from', ('192.168.5.103', 43516))
```

Then the master device will get the sensor readings from all slave devices. If the percentage of “true” sensor readings from all sensor devices is greater than some threshold (in this example, we used 67%), it will trigger a photo, else it will not and will rest and repeat the process.

```
pi@raspberrypi:~ $ python ct_publish_master_multisensors.py
Started master multisensor photo sync program
Resting...
Starting
Socket is listening
('Got connection from', ('192.168.5.103', 43516))
['192.168.5.101', 'False']
['192.168.5.103', 'False']
ratio: 0.0
Resting...
Starting
Socket is listening
('Got connection from', ('192.168.5.103', 43518))
['192.168.5.101', 'False']
['192.168.5.103', 'True']
ratio: 0.5
Resting...
```

```
Resting...
Starting
Socket is listening
('Got connection from', ('192.168.5.103', 43522))
['192.168.5.101', 'True']
['192.168.5.103', 'True']
ratio: 1.0
successfully created the directory /home/pi/cameraTrapPhotos/set1/
Camera 1 photo number 1 taken
Resting...
```

When the master starts a photo session, it will send a message to the slave devices to take a photo and send it to the master device.

```
pi@raspberrypi:~ $ python ct_client_slave_multisensors.py
Connected with result code 0
Started multisensors photo synchronization program on slave 3, waiting for signal from master module
successfully created the directory /home/pi/cameraTrapPhotos/set1/
Photo Number 1 taken on Slave 3
Moving from Cam 3 to Master
Finished moving photo from Cam 3 to master
closed FTP connection
```

e. Setting up mobile app

To set up the mobile app, we first need to install Android Studio:

<https://developer.android.com/studio>

Download the mobile app folder from Github

In Android Studio, go to the “File” tab, and select “import project”. Select the mobile app folder you downloaded.

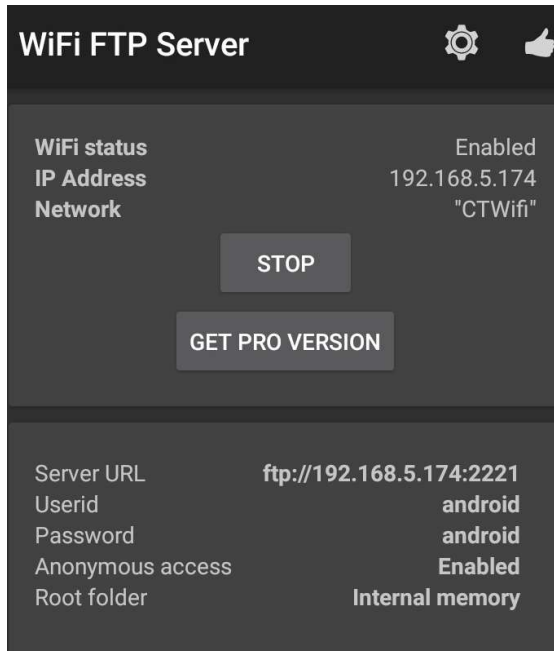
Plug in your Android phone via USB into your computer. To download the app onto your phone, you must go into your phone’s settings and enable USB debugging.

Run the app by clicking on the run button or go to the “Run” tab and click “Run App”
The app should be installed on your phone

On your phone, you must also install an FTP server app, we recommend this app:

https://play.google.com/store/apps/details?id=com.medhaapps.wiftpserver&hl=en_US

Make sure that the port number is 2221 and the username and password are set to the ones in the code. The default using our recommended app is username:android, password:android . Start the FTP server on. Make sure the settings are similar to the one shown below



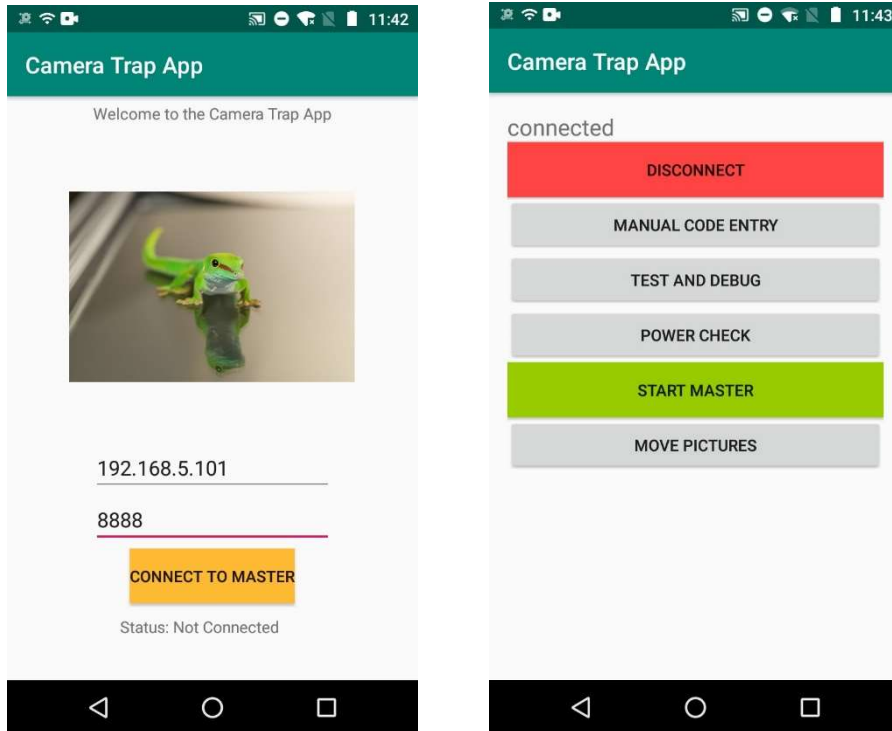
f. Setting up mobile server

Download the `ct_mobile_server.py` from Github. Make sure the IP address, client port, and the username and password for the FTP server are the same as the one you are running on the phone.

Run the mobile server code on the Master Pi using the command `python3 ct_mobile_server.py`. Make sure you are connected to the same network the master Pi is on. Make sure the host IP in the code is correctly set to the IP address of the Master Pi. This message should appear on the Pi.

```
pi@raspberrypi:~ $ python3 ct_mobile_server.py
Starting up Camera Trap Mobile Server
```

Run the FTP server and mobile app on the phone. Type in the IP address of the Master Pi and the port for the mobile server (default port = 8888) as shown below. You should see a connected text on the mobile app on the next page if you successfully set it up.

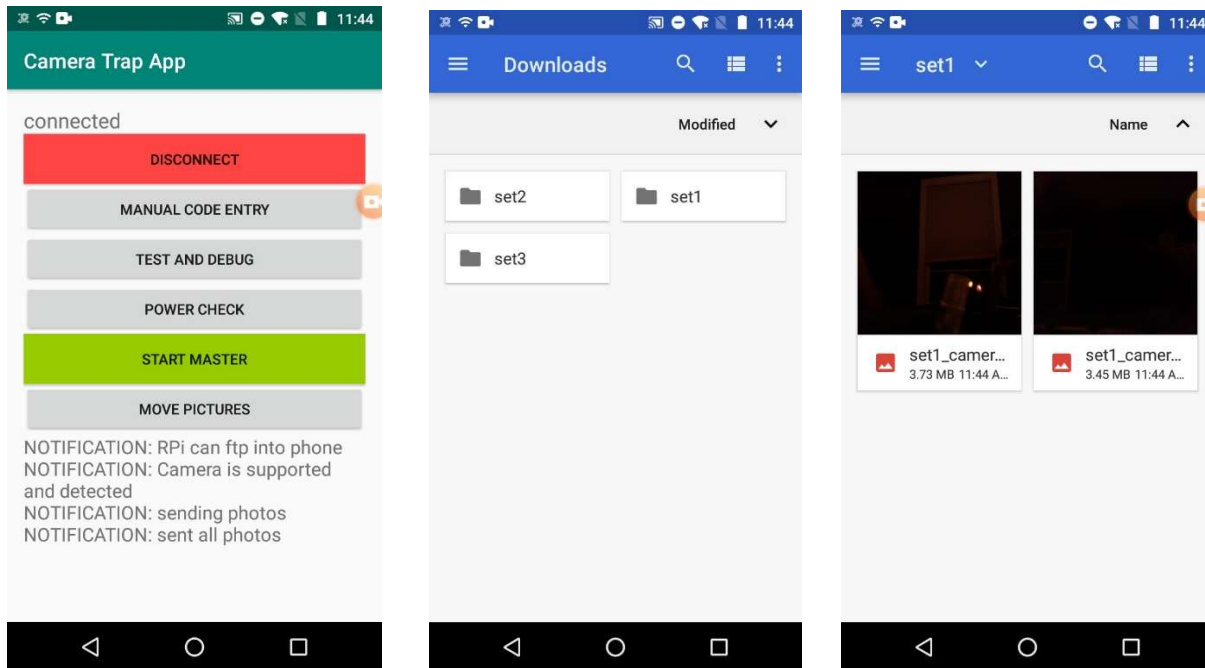


The phone should be connected now and a message will appear on the master Pi as shown below if successful. From there, as long as the mobile server is running on the master Pi, you can do many options, including downloading the photos onto an SD on your phone. For this, make sure the FTP server is running on your phone. If you are disconnected from the socket connection and cannot connect to the Pi through the phone, restart both the mobile app and the mobile server on the Pi.

```
pi@raspberrypi:~ $ python3 ct_mobile_server.py
Starting up Camera Trap Mobile Server
Connected by 192.168.5.174
```

To download the photos from the Master Pi, select the “Move Pictures” box.

It will print a notification of “sending pictures” and “sent all photos” as shown below. To see the photos, go to the Downloads directory and you should see your photo sets as shown below



The master Pi will also give notifications (as shown below for image transfer) for different commands.

```
pi@raspberrypi:~ $ python3 ct_mobile_server.py
Starting up Camera Trap Mobile Server
Connected by 192.168.5.174
sending photos
sending photo number 1 out of 6
sent photo number 1out of 6 to phone
sending photo number 2 out of 6
sent photo number 2out of 6 to phone
sending photo number 3 out of 6
sent photo number 3out of 6 to phone
sending photo number 4 out of 6
sent photo number 4out of 6 to phone
sending photo number 5 out of 6
sent photo number 5out of 6 to phone
sending photo number 6 out of 6
sent photo number 6out of 6 to phone
done. sent all photos to phone
```

g. Setting up RTC (Real time clock)

Optionally, you can install an RTC (real time clock) on the Access Point Raspberry Pi. The Pi does not have a hardware clock to keep track of time. It has a software clock synchronized with an NTP server over the Internet to keep the time. In this section, we will add a hardware clock which has a timer to the Pi. This will be helpful for maintaining time synchronization within the system.

Here is a guide to setting up an RTC on the Raspberry Pi. <https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi>

Make sure you are connected to the Internet. Set the time of the Pi synced with the NTP server by typing the command `"sudo raspi-config"`, then select *4. Localisation Options* then *I2. Change Timezone*, and select the time zone you are currently in. This will set the Pi's time to the time controlled by the NTP server for the time zone you selected.

We recommend using the PCF8523 RTC (<https://www.adafruit.com/product/3386>), but feel free to use any RTC of your choosing

Put the RTC on the GPIO pins on the end near the SD Card like in the figure below



First ,enable I2C by typing `"sudo raspi-config"`, then *5. Interfacing Options*, then *P5. I2C*, then reboot the Pi.

Run the command `"sudo apt-get install python-smbus i2c-tools"`

Then, type `"sudo i2cdetect -y 1"`. You should see ID # 68 appear, which is the address of the RTC

```

Serial-COM7 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
[Icons]
pi@raspberrypi:~$ sudo i2cdetect -y 0
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi:~$
Ready      Serial: COM7  11, 19  11 Rows, 58 Cols  Xterm

```

Next, edit the `/boot/config.txt` file with the command “`sudo nano /boot/config.txt`”

Add the line:

`dtoverlay=i2c-rtc, pcf8523`

To the end of the file and save it. Change the RTC name depending on which one you use.

Reboot the Pi and then run “`sudo i2cdetect -y 1`” again and you should see UU where it showed #68 before



```

pi@raspberrypi:~$ sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
40:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- UU -- -- -- -- -- --
70:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi:~$

```

Disable the software clock by running the commands:

`sudo apt-get -y remove fake-hwclock`

`sudo update-rc.d -f fake-hwclock remove`

`sudo systemctl disable fake-hwclock`

Now, edit the file `/lib/udev/hwclock-set` by running the command

“`sudo nano /lib/udev/hwclock-set`”

Comment out these lines by putting a `#` sign in front of them

`if [-e /run/systemd/system] ; then`

`exit 0`

`fi`

Your file should look like this:

```
#!/bin/sh
# Reset the System Clock to UTC if the hardware clock from which it
# was copied by the kernel was in localtime.

dev=$1

# if [ -e /run/systemd/system ] ; then
#     exit 0
# fi

if [ -e /run/udev/hwclock-set ]; then
    exit 0
fi

if [ -f /etc/default/rcS ] ; then
    . /etc/default/rcS
fi

# These defaults are user-overridable in /etc/default/hwclock
```

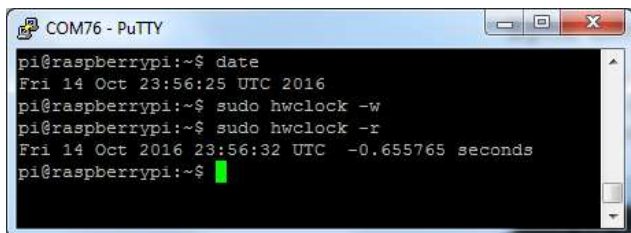
Also comment out these two lines:

```
/sbin/hwclock --rtc=$dev --systz --badyear
/sbin/hwclock --rtc=$dev --systz
```

```
if [ yes = "$RANDOM" ] ; then
# /sbin/hwclock --rtc=$dev --systz --badyear
# /sbin/hwclock --rtc=$dev --hctosys --badyear
else
# /sbin/hwclock --rtc=$dev --systz
# /sbin/hwclock --rtc=$dev --hctosys
fi

# Note 'touch' may not be available in initramfs
> /run/udev/hwclock-set
```

Then reboot the Pi again. To read the time from the RTC use the command “*sudo hwclock -D -r*”. The current time on the hardware clock is invalid. To set the time, first run the command “*date*” to verify the time is correct. Then run the command “*sudo hwclock -w*” to write the time to the RTC. Then “*sudo hwclock -r*” to read the time from the RTC. If the time from the RTC is correctly synced, you have successfully set up the RTC.



```
COM76 - PuTTY
pi@raspberrypi:~$ date
Fri 14 Oct 23:56:25 UTC 2016
pi@raspberrypi:~$ sudo hwclock -w
pi@raspberrypi:~$ sudo hwclock -r
Fri 14 Oct 2016 23:56:32 UTC -0.655765 seconds
pi@raspberrypi:~$
```

You can sync the times of all of the Pis in the network through the Access Point Pi which has the RTC using the timesyncssh.sh file in the AP_Files folder. The access point Pi uses the script's awk utility to get the IP addresses of all Pis in the network and then ssh individually into each Pi and set the time from its RTC onto each Pi's time.

h. Setting up the enclosure

1. See separate enclosure setup document

3. Setup from Image Files

Saved to the github repository are image files from the working Raspberry Pis. These are exact copies of the working modules. In order to utilize these, you must have a micro-SD card the of at least 16 gigabytes and ideally the same brand and model as used here to avoid problems when flashing the image. SanDisk Ultra 16GB was used. This recommendation is here because sometimes problems can arise where different brand cards are slightly different sizes and will then not allow the files to be flashed.

Plug the card into a computer and use an etching tool (balena etcher was used) to copy the correct boot files from the repository. This method is the same as downloading the OS and is detailed above. Connect the PIR sensor and camera and the Pi should be ready for use.

Note under 3.c.ii you set up a static IP address for the Pis. Ensure you don't set up the Pis with the same static address when copying from the image file or allow the AP Pi to assign IP addresses as they connect.