Contents

Getting started

Home cage surveillance is a system to record video of your animals in their home cage.

It is designed to run on a Raspberry Pi computer using a Raspberry Pi NoIR camera.

Features

- Record video 24/7
- Automaticall controls day-time and night-time lights.
- Live video streaming to a web browser

Install

See the installing the software page.

Interface options

- **command line** - A command line interface.
- **web** - A point and click web browser interface.
- **REST** - A rest interface to communicate with a homecage server

Images

These are images of the homecage at various developmental stages

v0.0

Overview

##### Lights and camera ##### Rats nest

Install

Homecage requires the following libraries:

- Wiring Pi - Library that provides a command line interface to the GPIO pins. This should be installed by default.
- GPIO - Python library to control GPIO pins.
- flask - A python web server.
- uv4l - Library for live video streaming to a web browser
- Adafruit_DHT - (optional) Python library to read from a DHT temperature and humidity sensor.

1) Get a functioning Raspberry Pi

These instructions assume you have a functioning Raspberry Pi. To get started setting up a Pi from scratch, see our setup intructions.

2) Clone the repository

This will make a folder `homecage` in your root directory. You can always return to your root directory with `cd`

```
# if you don't already have git installed
sudo apt-get install git

git clone https://github.com/cudmore/homecage.git
```

3) Install python libraries

```
# if you don't already have pip installed
sudo apt-get install python-pip
```

```
pip install rpi.gpio
pip install flask

# if you run into errors then try installing
sudo apt-get install build-essential python-dev python-openssl
```

4) Install DHT temperature sensor (optional)

If you run into trouble then go to this tutorial.

```
cd
mkdir tmp
cd tmp
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
cd Adafruit_Python_DHT
sudo python setup.py install
```

5) Install uv4l for live video streaming (optional)

If you run into trouble, then follow this tutorial.

```
curl http://www.linux-projects.org/listing/uv4l_repo/lrkey.asc | sudo apt-key add -

## add the following line to /etc/apt/sources.list
## start editor with `sudo pico /etc/apt/sources.list`
deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/stretch stretch main

sudo apt-get update
sudo apt-get install uv4l uv4l-raspicam uv4l-server
```

6) Create the folder to save video files

```
cd
mkdir video
```

Video files will be saved to `/home/pi/video`. This can be changed in the web server configuration file `homecage/homecage_app/config.json`. If your going to save a lot of video, please mount a usb key and save videos there.

7) Start the web server at boot (optional)

Edit crontab

```
crontab -e
```

Add the following line to the end of the file (make sure it is one line)

```
@reboot (sleep 10; cd /home/pi/homecage/homecage_app && /usr/bin/python /home/pi/homecage/ho
```

Done installing !!!

At this point you can interact with the homecage either through the web or from the command line.

Web interface

Running the web server

At a command prompt, type:

```
cd
cd homecage/homecage_app
python homecage_app.py
```

Once `homecage_app.py` is running you can access the web server in a browser with the address:

```
http:[your_ip]:5000
```

Where [your_ip] is the IP address of your Pi.

To stop the homecage web server, use keyboard `ctrl+c`

Configuring the web server

The server can be configured by editing the `homecage/homecage_app/config.json` file.

```
cd
pico homecage/homecage_app/config.json
```

The default file is:

```
{
    "hardware":{
        "irLightPin": 7,
        "whiteLightPin": 8,
        "temperatureSensor": 9
    },
    "lights":{
        "sunrise": 6,
        "sunset": 18
    },
    "video":{
        "fps": 30,
        "resolution": [1024,768],
        "fileDuration": 6,
        "captureStill": true,
        "stillInterval": 2
    },
    "stream": {
        "streamResolution": [1024,768]
    }
}
```

REST interface

The homecage server will respond to the following REST calls.

Server Status:

Get runtime status of server

`/status`

Get user configured options

`/params`

Record

Start and stop video recording

```
/record/1
/record/0
```

Stream

Start and stop video streaming

```
/stream/1
/stream/0
```

Lights

Turn lights on and off

```
/irLED/1
/irLED/0
/whiteLED/1
/whiteLED/0
```

Images

```
/lastimage
```

Set user options

```
/set/fps/<int:value>
/set/fileDuration/<int:value>
```

Command line interface

1) Log in to the Pi

On a Mac, use the terminal application in /Applications/Utilities/terminal.app

```
# Type
ssh pi@10.16.80.162

# Enter password
[your_password]
```

2) Change into the homecage directory

At the command prompt, type

```
cd
cd homecage
```

3) Get command help

The commands allow you to start and stop a video stream and video recording. They can also be used to turn the white and IR lights on and off.

To get help, at the command prompt, type

```
./help
```

This returns

```
 Status
    status - check the status
 Video Recording
    record start
    record stop
 Video Streaming
    stream start
    stream stop
 IR Light
    light ir on
    light ir off
 White Light
    light white on
    light white off
 Online Manual
    http://blog.cudmore.io/homecage
```

4) Position the cages within a good field-of-view

Start a video stream and then view the stream in a web browser.

```
stream start

# Returns
View the stream at:
   http://10.16.80.162:8080/stream
```

In any browser, go to the address `http://10.16.80.162:8080/stream`

While positioning cages, turn the white and or IR LEDs on and off

```
# Turn the white lights on
light white on

# Turn the white lights off
light white off
```

When your happy with position, stop the video stream

```
stream stop
```

5) Start continuous video recording

```
record start
```

This will save recorded video into individual files, 5 minutes of video per file. This will also control the light cycle, at night (6 PM - 6 AM) the white light is off and the IR light is on. During the day (6 AM - 6 PM), the white light is on and the IR light is off. Both the duration of each video file at the timing of the light cycle can easily be changed.

6) Mount the file server to get your video files

On a Mac, use `Finder -> Go -> Connect To Server...` and log in as follows

```
afp://10.16.80.162
username: pi
password: [your_password]
```

Files are saved in the `/video/` folder. Video files have the .h264 extension. There are also text files (extension .txt) saved, these have a log of temperature and humidity as well as the time the lights were turned on and off.

7) Log out of the Pi

```
exit
```

Mounting the file server

MacOS

This assumes that apple-file-protocol (AFP) is installed and running on the Pi

```
afp://[IP]
```

Windows

This assumes Samba (SMB) is running and installed on the Pi

```
smb:\\[IP]
```

Wiring the system

Camera

Attach the camera to the Pi with a flat ribbon cable. The cable should have blue tabs on one side of each end.

The blue tab goes towards the ethernet port on the Pi and towards the back of the camera (away from the lens).

Lights

You want to use an external 12V AC/DC power supply. Never connect this directly to the Pi, instead use a relay switch.

- Wire 2x GPIO pins to a two-channel relay
- Connect the lights to the two-channel relay

Optional

- Wire the DHT temperature sensor (optional)
- Wire a IR light sensor
- Wire a visible light sensor

Parts list

- Build a box to hold cages, lights, and camera
- Strap the computer to the side, place on top or put inside the box. If placing inside the box, make sure to make an inner box to block LEDs on computer.

Computer

- Raspberry Pi 3
- 5V AC/DC power, 2A
- SD card, class 10, 16 GB (for system installation)
- USB key, 64 GB (to save video)
- ethernet cable
- case

Camera

- Raspberry Pi NoIR
- CSI Camera cable

Lights

- 12V AC/DC adapter
- 2-channel relay (to switch lights on/off)
- IR lights (< 900 nm)
- White lights

Environmental

- temperature and humidity sensor
- IR light sensor
- White light sensor

Troubleshooting

Converting h264 files to mp4

The Raspberry camera saves .h264 video files. This format is very efficient and creates small files (10 MB per 5 minutes) but does require conversion to mp4 to impose a time.

See this blog post

Troubleshoot video recording

Capture a single image

```
raspistill -o test.jpg
```

Troubleshoot video streaming

Run uv4l by hand

```
uv4l --driver raspicam --auto-video_nr --encoding h264 --width 640 --height 480 --enable-se
```

Browse the live stream at

```
http://[IP]:8080
```

Stop uv4l (make sure all browser windows are closed)

```
sudo pkill uv4l
```

Development notes

mkDocs

We use mkdocs to generate the documentation website from markdown files.

Install

```
pip install mkdocs

# we are using the material theme
pip install mkdocs-material
```

Serve locally

```
cd
cd homecage/docs
mkdocs serve
```

Push to github

```
cd
cd homecage/docs
mkdocs gh-deploy --clean
```

uv4l

uv4l is what we use to stream live video.

20171120 - Problem was that if streaming was on and we tried to stop it while there was still an opened browser window we would get an orphaned `<defunct>` process that can't actually be kill(ed). This was mucking up any future interaciton as `stream`, `record`, and `status` thought there was still a uv4l process.

Sent this to uv4l people

```
Hi there, great product and the best streaming I have ever seen.

I am running uv4l on a Raspberry Pi (Jessie) and it is working very well.
One problem is if I kill the stream with `sudo pkill uv4l` while there is
a browser window open (that is viewing the stream) I end up with a <defunct>
uv4l process that I can't seem to kill?

Can you suggest a server option I could use to stop this behavior?
I want to `sudo pkill uv4l` from the pi while some remote user still
has a stream window open in the browser? I've looked through the server
options and don't really know what I am looking for?
```

```
Thanks again for uv4l

p.s. Can you suggest an online forum for such questions?
```

Answer was to kill child processes first. Get child processes of PID with 'pstree -p PID'

Which eventually led to this

```
## get uv4l PID
PID = pgrep uv4l
## kill all processes in the same group, this includes children
## kills original and does NOT leave a `<defunct>` uv4l !
sudo kill -- -PID
```

Remove uv4l-raspicam-extras

```
sudo apt-get remove uv4l-raspicam-extras
```

ToDo

20171111

- finish index.html interface, mostly adding interface to change self.config
- split self.config (from config.json) and self.status (runtime variables)
- add in dht sensor code
- add in white and ir sensor code