

UNIVERSITY OF DUBLIN



TRINITY COLLEGE

RASPBERRY SPI

Ellen Marie Burke
B.A. (Mod.) Computer Science
Final Year Project April 2014
Supervisor: Fergal Shevlin

SCHOOL OF COMPUTER SCIENCE AND STATISTICS
O'REILLY INSTITUTE, TRINITY COLLEGE, DUBLIN 2, IRELAND

DECLARATION

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university

Ellen Marie Burke 23rd April 2014

ACKNOWLEDGEMENTS

Thank you to everyone who helped me throughout this project

ABSTRACT

Security systems set up in homes can be expensive and complex to set up. The cameras used can be bulky in size and therefore difficult to successfully hide. This project is to create a home security system using a Raspberry Pi and the Raspberry Pi camera module.

Contents

1	Introduction	2
1.1	Motivation/Problem	2
1.2	Background Research	3
1.2.1	Mobile App or Web Application	3
2	Design	5
2.1	Hardware	5
2.2	Software	12
2.3	Additional Requirements	14
3	Implementation	16
3.1	Web Application	16
3.1.1	Accessing the Web Application	17
3.2	Help Page	18
3.3	Real Time Photo	19
3.4	Motion Detection	20
3.4.1	How Motion Detection Works	20
3.4.2	Front End Motion Detection Web Page	22
3.4.3	Back End Motion Detection Web Page	25
3.5	Video Live Streaming	26
4	Testing	27
4.0.1	Testing Motion Detection	27
4.0.2	Testing Raspberry Spi Web Application	28
4.0.3	Testing Real Time Photo	28
4.0.4	Testing Live Streaming	28
4.0.5	Testing Photo Storage	28
5	Conclusion	29
5.0.6	Future Work	29

Chapter 1

Introduction

Home security systems have been available for a few years now. Most are quite expensive and are very obvious to spot. The camera used in them may not be able to supply 1080p video.

1.1 Motivation/Problem

The aim of this project is to create a home security system that is affordable, easy to setup and use by anyone. It will be affordable and easy to use for all kinds of end users.

The Raspberry Spi will allow users to set up a system and have complete control over it. It has endless possibilities of features that can be implemented as extras or change current features.

Raspberry Spi is different from other security systems with regards to size, cost and the control a user has over it. The Raspberry Spi approximately costs 30 Euro and the camera module is approximately 20 Euro. The dimensions of the Pi are 85.60mm x 56mm x 21mm. The camera module is even smaller.

This project could be used for multiple real life uses and applications. It is specifically aimed at being used at home but it could be used elsewhere as well.

- A user who is out working all day and they would like to know what their pet is getting up to throughout the day.
- Keeping an eye on small children while at home with them.
- Simply checking on a home throughout the day.

1.2 Background Research

The objective of background research was to see what features current surveillance systems have that are effective and popular and to see which of these features can be used for this project.

Most security systems have the ability to live stream a video. Some systems this is the their only feature and it would have the ability to record the entire footage. This can sometimes be a waste because no useful information can be gotten from the footage or simply nothing happens that would need to be recorded. For this project live streaming is a feature but the live stream footage by default is not saved. This can be changed however if the user wishes. Another popular feature is motion detection. Instead of live streaming motion detection would be constantly running and only when a change occurs will the images get saved.

Before this project Security systems could be bought either pre made or built by a person using a web camera. For a pre built home security system it just need to be set up and users could watch a feed that was provided. But it gave no control to the user. No changes could be made or features added to it. The downside to a prebuilt system would be that they can become quite expensive acquire, maintain and repair if needed. The camera used would not be able to be hidden due to its size and the system being prebuilt will not allow any customization made to it or even have features added.

Systems made at home that a user could create and set up themselves usually consist of a desktop or laptop with a USB connected webcam or simply just a IP camera which can be remotely accessed. Web cameras and IP cameras can be expensive and the higher the quality desired the higher the cost of them will be. But like pre built systems the user may not have control over them depending on the software used.

1.2.1 Mobile App or Web Application

The original concept was to create an Android app for the Raspberry Spi simply because apps are very popular these days. From this app the end user would control the Raspberry Spi. On the app the images would be viewed, motion would be started & stopped and the video footage of the live stream could be viewed.

If an Android app had of been used this would mean that this project would only be useful to users who had access to an Android phone. This project would be unavailable to iPhone users, Windows users and users who would prefer to access it through a computer's web browser.

Therefore the decision to make this project accessible on a web application would mean that no user would be unable to use this project and it would be available to all. It also reduces one element of setting up the Raspberry Spi. No app needs to be downloaded and configured.

Chapter 2

Design

One of the main aims of this project is to make it easy for others to set up. With this in mind there are not a lot of hardware components required for the Raspberry Pi. Additional hardware items such as a keyboard and mouse would only be needed for initial set up. But can be used if needed.

2.1 Hardware

The following is a list of hardware components:

- Raspberry Pi Model B
- Raspberry Pi Model B Case
- Raspberry Pi Camera Module
- USB Hub powered externally
- Wifi Adapter
- SD card

Raspberry Pi Model B:

The Raspberry Pi is described as a credit card sized computer. There are

two available models but this project was implemented using a Model B. The features available on a Model B Raspberry Pi are:

- ARM Processor capable of 700 MHz
- 512 MB (Shared with the GPU)
- HDMI
- Composite RCA connector
- CSI connector for the Raspberry Pi Camera
- 2 USB ports
- 3.5mm Audio Jack

Raspberry Pi Case:

Most Raspberry Pi's are not sold with an external case. The case is essential because otherwise the Pi will have no protection and can get damaged very easily. A model B case is also essential so there is space for the camera module to connect to the CSI connector on the Raspberry board. Model A cases do not have a space for the camera to connect. This will make connecting the camera very difficult.

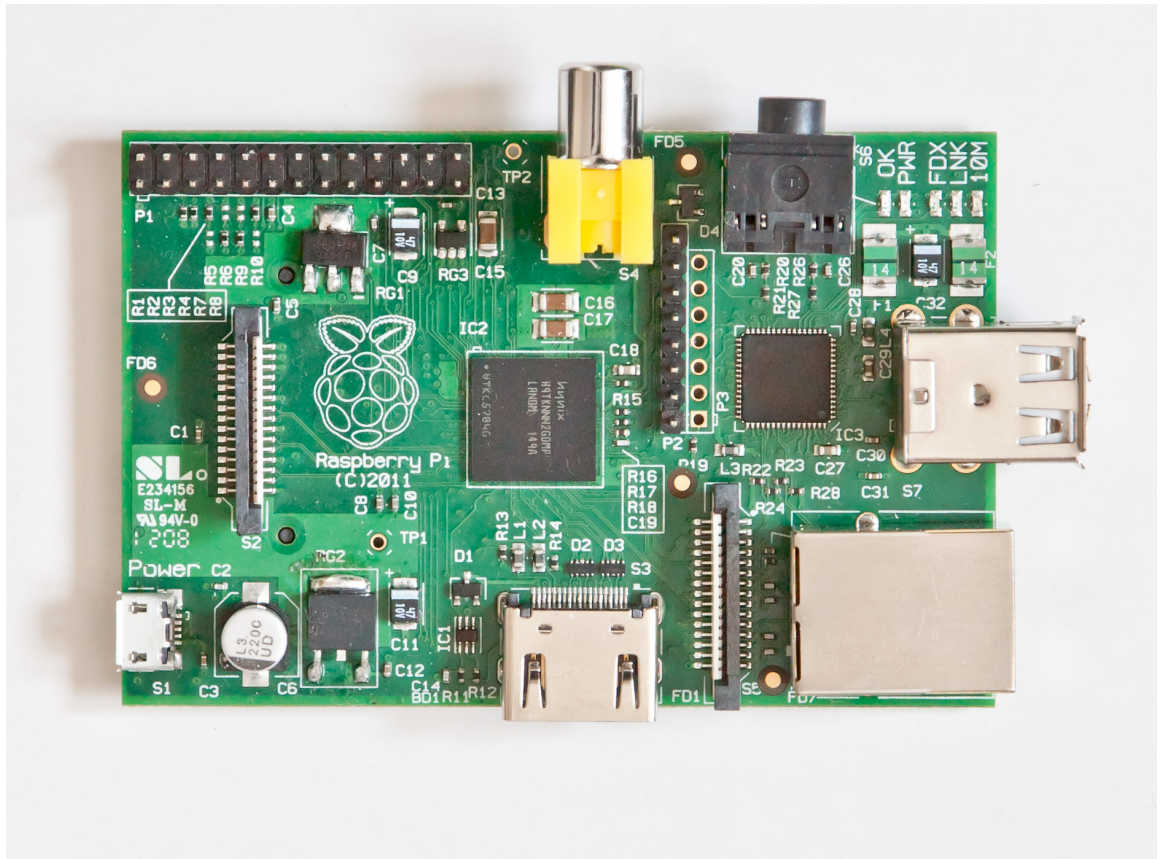


Figure 2.1: Raspberry Pi Model B board layout

Raspberry Pi Camera Module:

The camera module is a camera that has been designed specifically for the Raspberry Pi. It connects directly into the pi to the CSI connector rather than USB. The CSI connector exclusively carries pixel data and because of this it is capable of extremely high data rates.

The camera itself is not expensive. It costs roughly 20 Euro and will work instantly once it is properly attached to the Pi board. The camera is capable of outputting images and video of very high quality. For images the max resolution the camera is capable of is 2592 x 1944. For Video the camera is capable of being captured in 1080p quality. If a webcam was to be used it would be difficult to find one that would produce such high quality for such an affordable price.

There is an infrared filter on the camera and therefore it cannot detect IR. There is another camera module called the Pi Noir which is available.



Figure 2.2: Raspberry Pi Camera Module

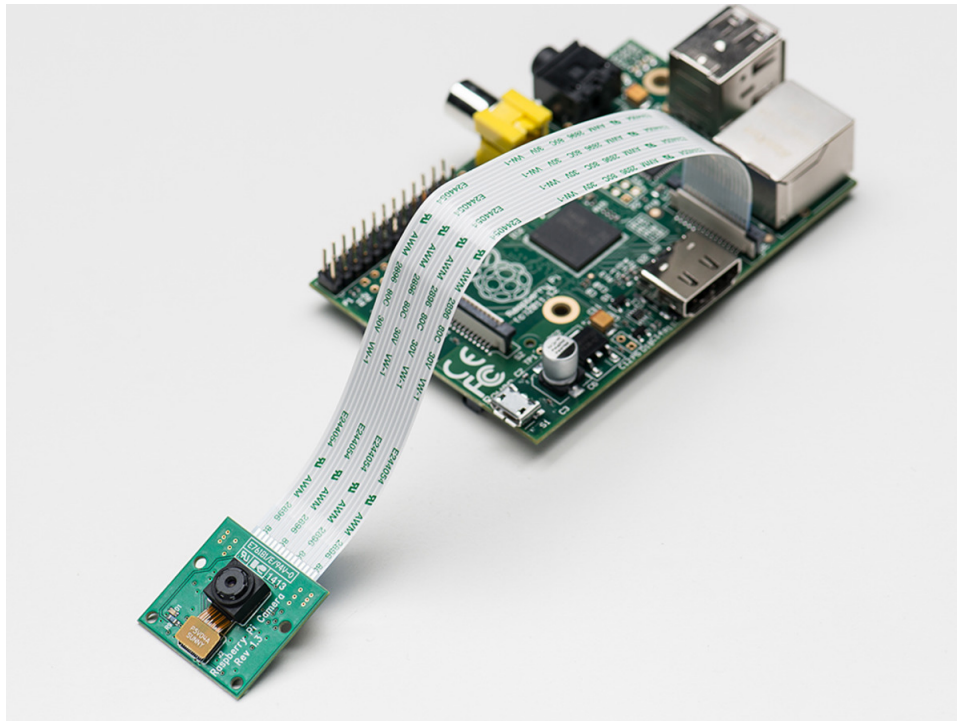


Figure 2.3: Camera Module attached to the Raspberry board with no casing

USB Hub powered externally:

A USB Hub is required to attach any USB components. If any component is connected by USB directly to the Pi, the Pi will power it. This will cause the Pi to heat up. If the Raspberry is left on for several hours, as this project aims to be, it could cause damage to the Pi. An externally powered USB hub will allow for the Pi to avoid heating up and create more USB slots available without affecting the Pi.

No particular external USB hub is needed any USB hub can be used.

Wifi Adapter:

When connecting to the internet a Wifi Adapter will be more useful than an Ethernet cable. By using an adapter it will avoid having to hide another cable and overall make the Pi neater to set up.

Using a Wifi Adapter is not a requirement of this project but is more of a recommendation. This is a decision the end user will make.

SD card:

An SD card will act as storage for the Pi. Most Raspberry Pi's do not come with an SD card and must be bought separately. The Raspberry Pi has no on board memory. An operating system will be loaded onto it and it will also store all images taken.

External hard drives can also be used to store the images but that will be another decision left to the end user. The storage device chosen depends on how frequent the Raspberry Pi will be used.

2.2 Software

The following is a list of all the software that will need to be installed:

- Operating System Raspian
- UV4L (Universal Video 4 Linux)
- OpenCV
- Apache
- Motion

Operating System Raspian:

With the Raspberry Pi there are several operating systems to choose from. This project uses Raspian because it is the most popular OS for the Pi. Being the most popular it is easy to debug problems and solve issues that arise when searching online.

UV4L (Universal Video 4 Linux):

The camera modules default driver is called RaspiCam. RaspiCam does not work properly with OpenCV. When OpenCV tries to locate a camera it looks for a device ID. The camera module does have a device ID but it's not an integer value. It's the value 'pi' because of this OpenCV cannot use the Camera Module with it's default drivers. To get around this driver problem UV4L (Universal Video 4 Linux) is used instead. UV4L allows OpenCV to locate and use the Camera.

When using this driver the UV4L initializing script must be run. It can be added to the Raspberry's start up script or can be called after each boot. The parameters passed in can be interchangeable.

OpenCV:

For all video and image processing the computer vision library OpenCV is used.

OpenCV will be responsible for taking a real time photo and motion detection.

Apache:

Apache web hosting will be used to host the Raspberry Spi website. Apache will continue running as a background process. While the Pi is handling motion detection, taking photos the website will be able to be accessed and process requests received.

Motion:

Motion is a program that will handle the video live streaming. Motion can be configured in many ways. From which port the video will stream, how many FPS (frames per second) are displayed to whether the live stream should be saved.

2.3 Additional Requirements

Aside from the mentioned hardware and software mentioned in the previous section there are other requirements and additional features that an end user can user.

Port Forwarding:

With the Raspberry Spi being set up on a home network port forwarding needs to be enabled. This project will not be able to work at all unless port forwarding is set up.

Port forwarding will different for each end users router. For this reason no explanation will be given and the end user will need to consult their router manual or search for the answer online.

A list of all ports that need to be open will be provided.

The following ports that will be used are:

Port	Port used for
80	HTTP Requests
8000/9000 or 8080	Video Live Streaming but can be changed
22	SSH into the Raspberry (Optional)

Port 80 is necessary and until this port is open the Raspberry Spi website will not be accessible. With port 80 open when the router receives a HTTP request it will know which device on the network to send the request to.

Port 22 is optional and will depend on how the end user will interact and make changes on the Raspberry Pi itself. If port 22 is open this will allow a user to SSH directly into it. SSHing into the Pi will let the user make changes from another computer instead of using the Raspberry and having to attach a keyboard and mouse.

Port 8000/9000 or 8080 is going to be used for live streaming. This port can be changed but will default to one of these ports. Changing the port that will be used for live streaming can be done in the Motion configuration file.

A browser with JavaScript enabled:

This project has a JavaScript function. Due to using JavaScript a browser that has JavaScript needs to be used when viewing the website.

If viewing the website on the Raspberry Pi itself the only browser currently that has JavaScript enabled is Chromium.

Chapter 3

Implementation

3.1 Web Application

How the site is made.

Once port forwarding has been set up the Raspberry Spi web application will be available on the external IP of the home network.

List of options available from the homepage:

- View Photos Currently stored on the Pi
- Take a real time photo
- Start & Stop Motion Detection
- Video Live Streaming
- Help Page

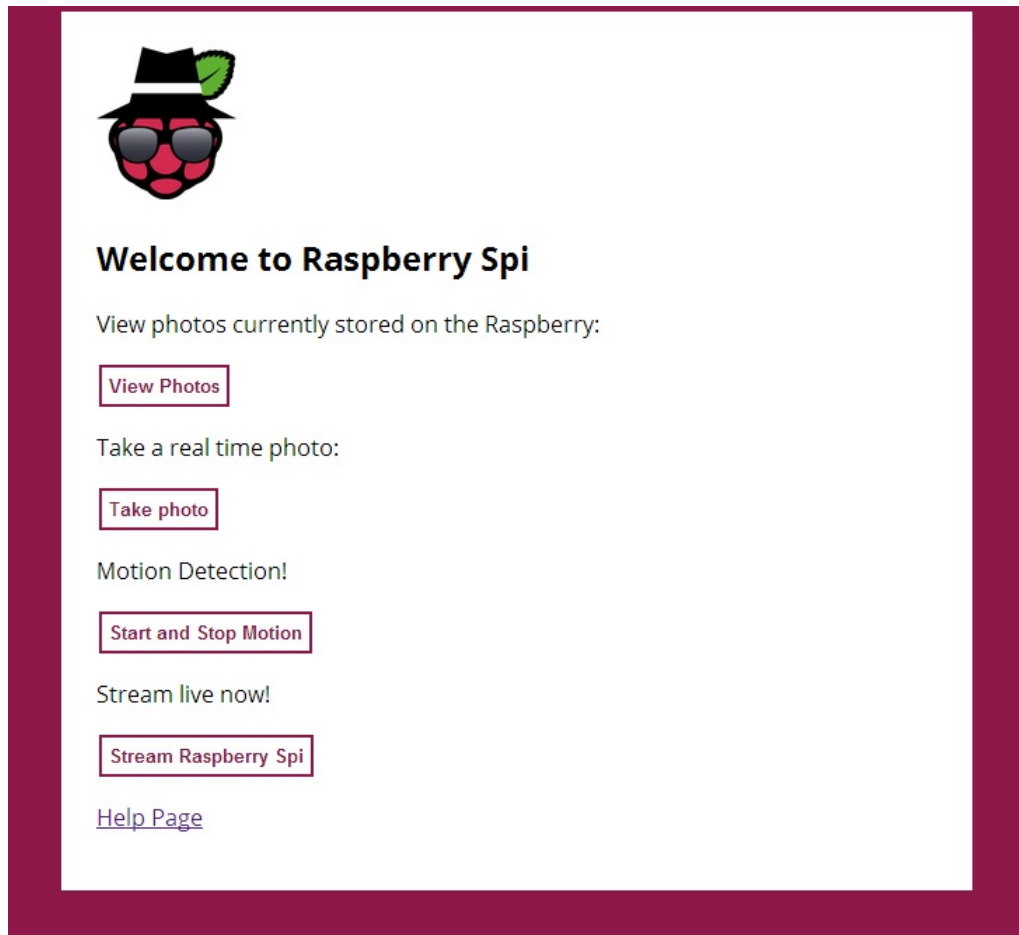


Figure 3.1: Home page with list of options

3.1.1 Accessing the Web Application

Accessing the website will require a username and password to view any pages. The access is controlled by a `.htaccess` file. This file is an Apache configuration file. In this file it will store what will be displayed when asking for the password. Currently the only information that will be displayed to the user before they enter a user name or password is the words Raspberry Spi to let the user know what they are logging in to.

Passwords are not kept in the same file or directory as the web pages. This is so they cannot be accessed on accident or viewed. The passwords and associated user names are stored in a `.htpasswd`. The passwords are encrypted and cannot be read. If a user would like to add a user or password it will need to be added from the actual Pi not the web application. The web application will

provide no hints to password or user name or any way to change either. This is done to avoid malicious attempts to access the Raspberry.

3.2 Help Page

A help page will be available on each page. It is there to answer problems that occurred most frequently when implementing and testing this project.

The questions are also available in a .txt file in case the web page is not accessible.

The questions are:

- The camera doesn't open?
- How does the motion detection work?
- The webpage is unavailable?
- When taking a live frame the webpage displays "Frame not read correctly"?
- How to SSH into the Raspberry Spi?
- Only a partial image was taken while using motion detection?
- where are the web pages stored on the pi?
- Source code for the Raspberry Spi?
- Question not here?

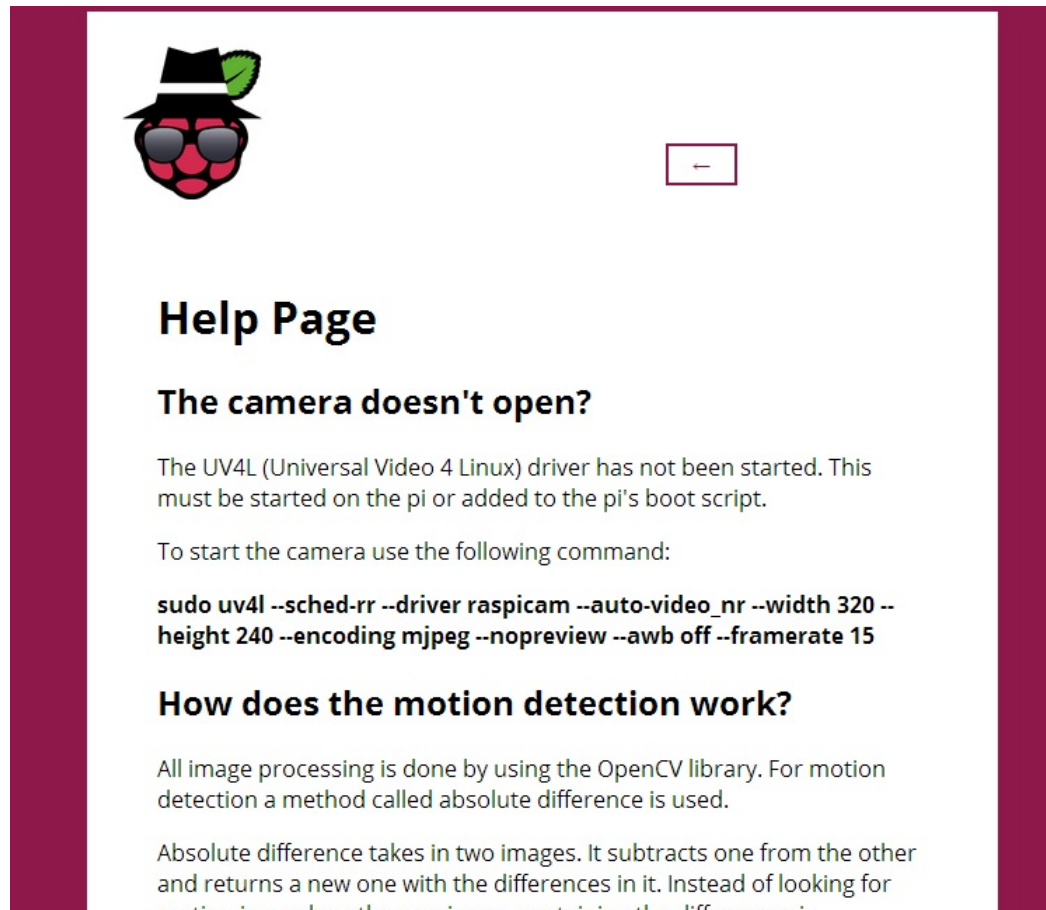


Figure 3.2: Help Page

3.3 Real Time Photo

Explain opencv works. Why CGI and how that works. [Example image output] Taking a real time photo is an option that allows the user to take a photo with the click of a button and then view it. This photo is taken by using a Common Gateway Interface CGI script.

CGI scripting is a way for a web user to click an option on a web page, the web server will then send the request server side where the application is then carried out. For this project the CGI application is done using C++.

The CGI opens the camera and will return one of these three options.

1. The camera did not open successfully
2. The frame taken is empty
3. The frame has been read correctly

Only if the frame is successful will a proper web page load. If the camera doesn't open or the frame is empty it will display the error to the user.

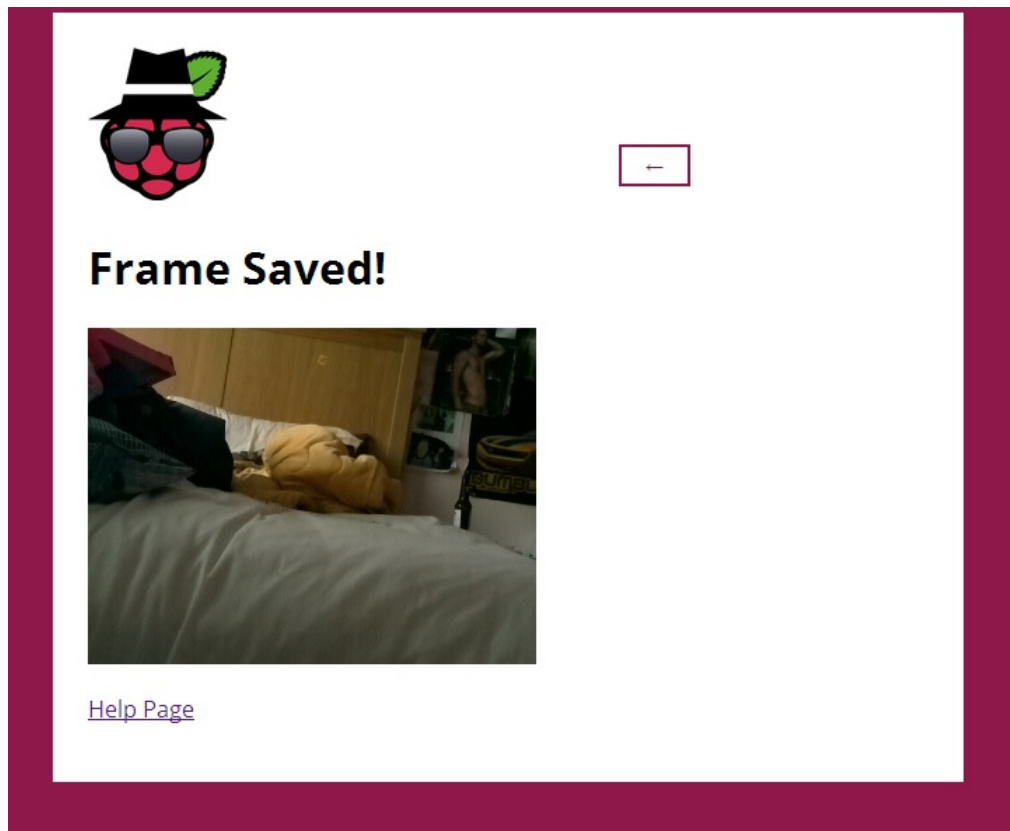


Figure 3.3: After taking a live frame

3.4 Motion Detection

3.4.1 How Motion Detection Works

Motion detection is done by using the OpenCV library. The method chosen for motion detection is Absolute Difference. This method will detect changes happening after new frames are read in. This method was chosen because it is

effective at detecting motion while not killing the pi.

OpenCV has an inbuilt function called *absdiff()*. This method takes in two RGB frames. The two frames are then subtracted from one another producing a new RGB image that contains the differences if any. RGB images are not useful when looking for changes so the difference image is converted to grayscale. This means that the image will only contain black and white pixels making it easier to see if there are any changes from one frame to the next. If there are no changes the image will only contain black pixels. When there are changes white pixels will appear. The white pixels are counted and if there are more pixels then a certain count then motion has been detected.

```
absdif(prevFrame - nextFrame = difference)
```

```
grayscale(difference)
```

```
checkWhiteCount (difference)
```

```
if WhiteCount is above threshold save images
```

If motion does get detected the RGB frames that were originally read in are saved. Since the difference image will not mean anything it does not get saved. If motion gets detected then the images are not saved and new images are read in.

The following is an example of the motion detection working using Absolute Difference.

Here are the two frames that have been read in. The first frame read in will represent *prevFrame* and the second from will represent *nextFrame*. It is clear from these two images that there is a difference between them.



Figure 3.4: prevFrame

Figure 3.5: nextFrame

This image is the difference image after it has been converted to grayscale. The white pixels represent the differences that absolute difference has found.



Figure 3.6: Difference image after it has been converted to grayscale

3.4.2 Front End Motion Detection Web Page

On the motion detection web page there are two options. Start motion and stop motion. If motion is currently running it will be displayed to the user. There will be text displaying if the motion is on or off and a small icon will appear as well.

Screenshots from the motion detection area on the web application

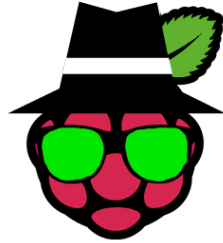


Figure 3.7: Motion Detection Start icon

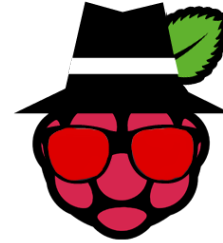


Figure 3.8: Motion Detection Stop icon

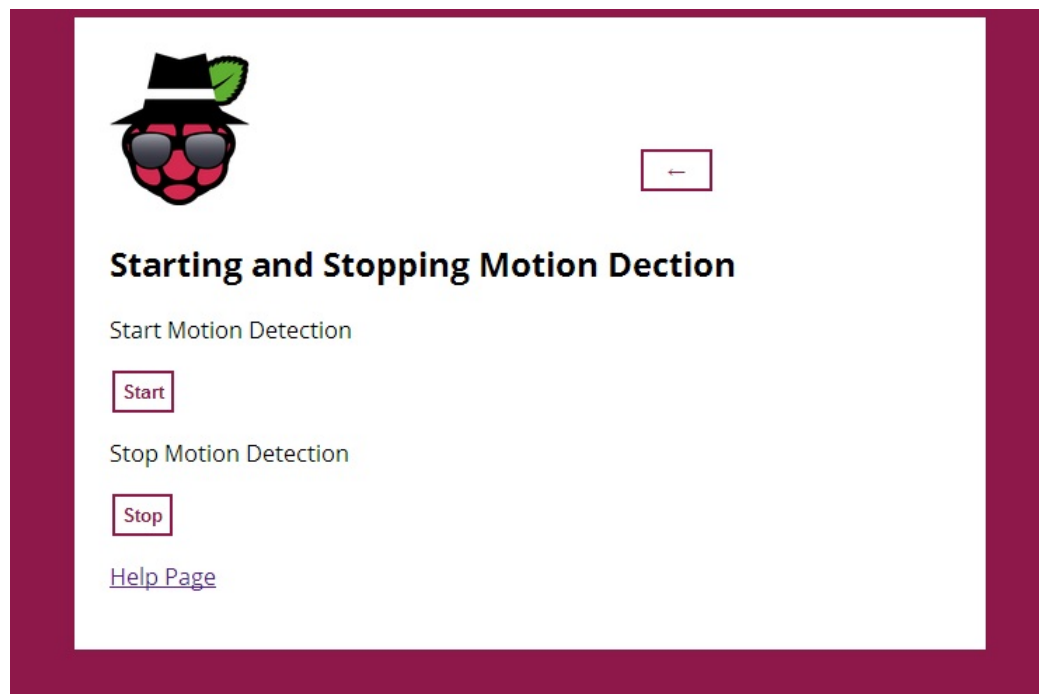


Figure 3.9: Motion Detection page

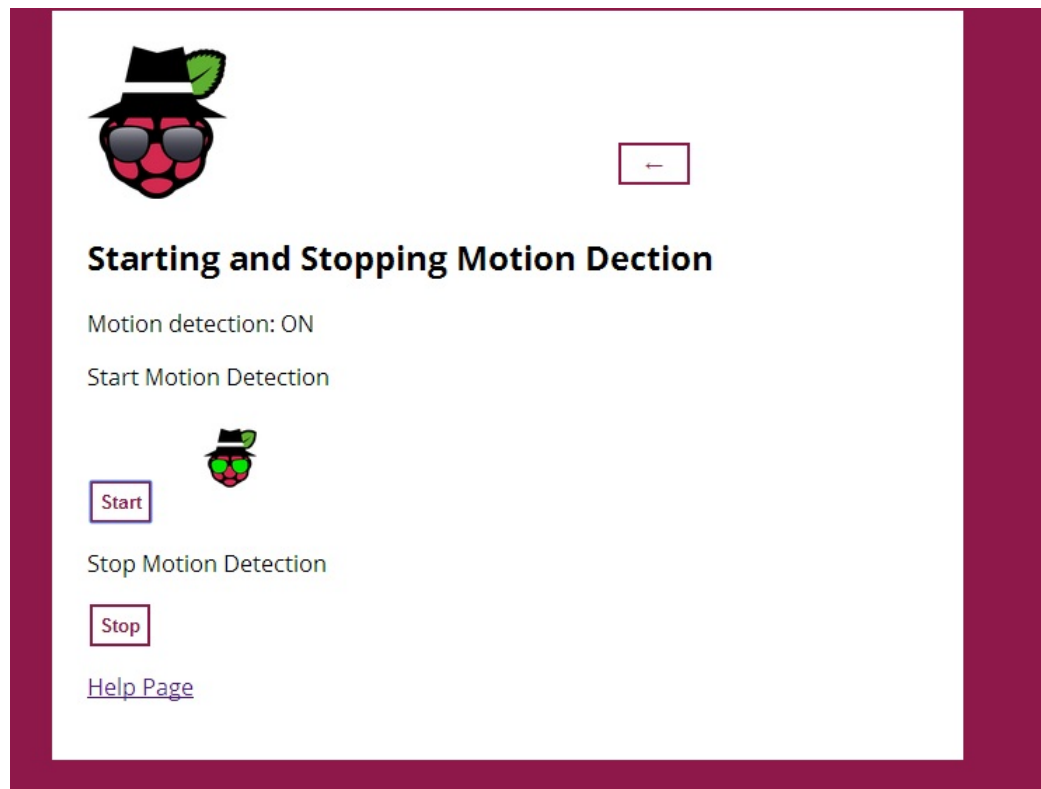


Figure 3.10: Motion Detection after being started

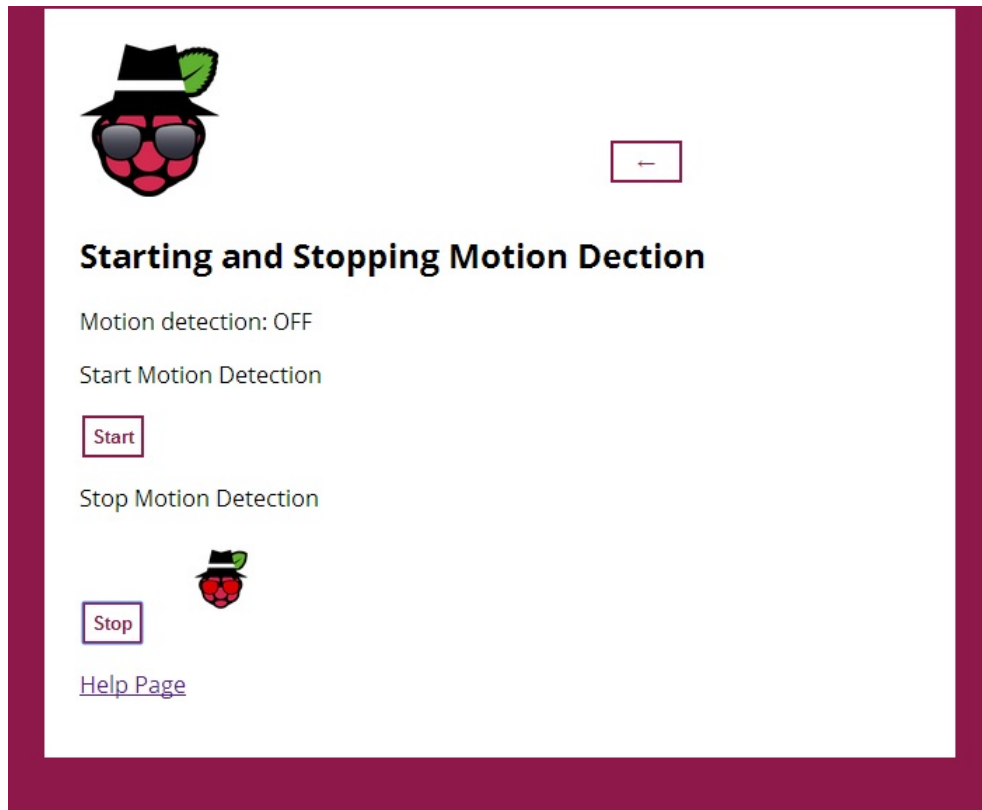


Figure 3.11: Motion Detection after being stopped

3.4.3 Back End Motion Detection Web Page

The back end of the motion detection web page is taken care of by a JavaScript function which sends an AJAX request to a PHP program which will call the correct bash script depending on which motion detection option is used.

When a button is pressed the value of either 'start' or 'stop' will be passed into a JavaScript function. This function will determine whether start or stop has been called. It will then use an AJAX request to send the value 'motion' to the correct PHP program. When the PHP program receives the request it will check that it receives the word 'motion'. If it does then it will call a bash script that will either start or stop the motion detection.

3.5 Video Live Streaming

Motion. How motion(live streaming) works. Motion is a program that is available to download. It sets up a video stream and will display images from the camera attached to the Pi to a given web page. It can also offer to do motion detection but since is been implemented using OpenCV this aspect of Motion is not in use.

Before Motion is used it is configured from how many frames it will display per second to the port the video is displayed on. Motion will take frames that are

This feature currently does not work properly or efficiently.

The video live streaming option on the home page of the web application does not start or stop streaming. It will display a blank page as to avoid effecting other features from working and effecting the pi connecting to the internet. Having this feature working properly would be an area of this project that could be expanded upon or changed in the future

Chapter 4

Testing

4.0.1 Testing Motion Detection

To get motion detection working effectively while not overpowering the Pi causing it to be unable to process anything else took some testing to get it just right. Motion Detection on a computer not a Pi would usually read in every frame from a video stream and would process each of the frames looking for motion. The Pi would be capable of such a task but for this project the Pi is expected to keep a website running and be able to process requests that it receives all while the motion detection is running.

To do this not every frame is read in but finding out the correct number of frames to skip and still be able to detect motion is where most of the testing took place. The same object of a book was used for each test. It was held in front of the camera and moved from one side of the frame to the other. The test was started at skip ten frames and read in that frame. Skipping ten frames was unsuccessful as it never caught any motion. If it did it only caught the end which is not very useful.

Ten frames was reduced down to five frames and the test was repeated. It proved to work much better but it still didnt catch all motion that was happening. Again the frames being skipped was reduced down to three frames. This test proved to be the most successful as motion was detected much better and it did not interfere with the web application.

The frames were reduced to having no skips but then motion detection started interfering with the web application and making the web pages have a delay in being displayed. This proved that skipping a set amount of frames helps.

4.0.2 Testing Raspberry Spi Web Application

After motion detection was found to be working as planned and correctly the website was tested to see if the Pi was still able to process requests while motion detection was running in the background.

For this test motion detection was started and different test users were asked to log into the website and go through all the images stored on the Pi. Objects were then moved on front of the camera so that motion detection would start saving images.

From this test images from the motion detection were successfully saved and no user had any issues or delays logging in or in accessing the photos

4.0.3 Testing Real Time Photo

Testing a real time photo was done by placing different objects on front of the camera and when a real time photo was taken seeing if they would appear.

All photos are saved with a timestamp but this test was to ensure that the photo was in fact recently taken but the image was updating when being viewed on the webpage.

4.0.4 Testing Live Streaming

Live Streaming was tested not using the web application but on a localhost connection. To test if the stream was live and if the the frames per second (FPS) were correct I walked from one side of the frame to the other. Live streaming was tested with the website as well but it caused several issues from taking down Apache and causing the web page to be unavailable to effecting the wifi connection on the Pi.

4.0.5 Testing Photo Storage

To test if photos were being stored on the Pi real time photos were taken and the time was taken so the it could be compared to the timestamp. The web address where all photos are stored was checked to see if the image appeared with the correct time stamp and the expected photo appeared. All photos taken from either feature appeared in a swift manner available to be viewed. Timestamps and dates on each photo were also always correct.

Chapter 5

Conclusion

The aim of this project was to build a different kind of security system that is affordable and easy to use but also effective. Raspberry Spi has achieved this. It is an affordable set up that is easy to control. Users are able to view photos previously taken or captured from motion detection, live stream video or start and stop motion detection.

Various changes can be made to match each user depending on their preference. It can have multiple real life uses for varying end users. Being set up on a home network allows for having full control over the system from who can access the Raspberry to the resolution of images being used.

All of the Raspberry Spi code is available to download from github which allows for endless possibilities of additional extras.

Raspberry Spi Code is available here:

<http://www.github.com/Smellen/RaspberrySpi>

5.0.6 Future Work

The video streaming does not work well or efficiently. To continue working on this project this area would involve either a completely new approach or continue using the program Motion but try to implement it working better alongside Apache.

Another area that could be improved is starting and stopping motion detection. JavaScript-¿Ajax Request-¿PHP-¿bas script is too much and could be made more efficient.

A different approach to this project would be to use the PiNoir camera which only works at night time and has no infrared filter. It would be a Raspberry Spi project for during the night.

Adding to this project even more would be adding in more Raspberry Pis. One Pi per room or as many as preferred. One Pi controlling the web page and the rest only taking photos, motion detection and live streaming.



Bibliography

- [1] John W. Dower *Readings compiled for History 21.479*. 1991.