# 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 has 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. (More needs to be added)

# Contents

# 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. (Add more)

## 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 operate 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 regard 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 has multiple real life uses and applications. It is specifically aimed for use at home but it could be operated elsewhere as well, for example:

- A user who is out working and would like to know what their pet is

getting up to throughout the day.

- Checking the quality of care of small children while the user is at work.

- Keeping an eye on small children without disturbing them while they are at sleep or play.

- Simply checking home security 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 only have the ability to live stream a video and record the entire footage, this can sometimes be a waste because no useful information can be obtained from the footage.

For this project live streaming is a feature but by default the entire footage captured is not saved. However this could be changed if the user wished.

Motion detection is a popular feature in most security systems and instead of live streaming motion detection would constantly run in the background and only when a change occurs would the image be saved.

Before this project Security systems could be bought either pre made or built by a person using a web camera. A pre built home security system when set up would provide a feed but gave no control to the user as no features or changes could be added to it.The downside to a pre built system is that they can be quite expensive to acquire and maintain or repair if needed. The cameras available could be difficult due to their size and the inability of the system to be customised.

Systems set up and created by a user at home usually consist of a desktop or laptop with a USB connected web camera or simply just a IP camera which can be remotely accessed. These web cameras and IP cameras again could be expensive as the higher the quality desired the higher the cost of them would be. Like pre built systems the user may not have as much control over them depending on the software used.

### 1.2.1 Mobile App or Web Application

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

If the Android app was the method used to control this project this would mean that only Android phone users could access this project excluding iPhone users, Windows users and anyone who wished to access it through a computer's web browser.

Therefore the decision was made to have this project controlled through a web application so that no user would be excluded.

The other factor in this decision was it reduced one element of setting up the Raspberry Spi, no app would need to be downloaded or configured.

# Chapter 2

# Design

One of the main aims of this project is to make it easy for others to set up regardless of their background. With this in mind there are not a lot of hardware components required for the Raspberry Spi. Additional hardware items such as a keyboard and mouse would only be needed for initial set up. But depending on how a user decides to make changes on the Pi they can be used if needed. Some hardware components are also just a suggestion to make it easier to control the Raspberry Spi.

## 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 Model A and Model B. But this project has been implemented using a Raspberry Pi Model B. The features available on a Model B 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

A Raspberry Pi Model A is still the same size as the Model B except it's slightly cheaper and it has different components. The Model A differs because it only has 256MB of RAM available, no Ethernet port and one USB slot. This project has also not been tested on a Model A version

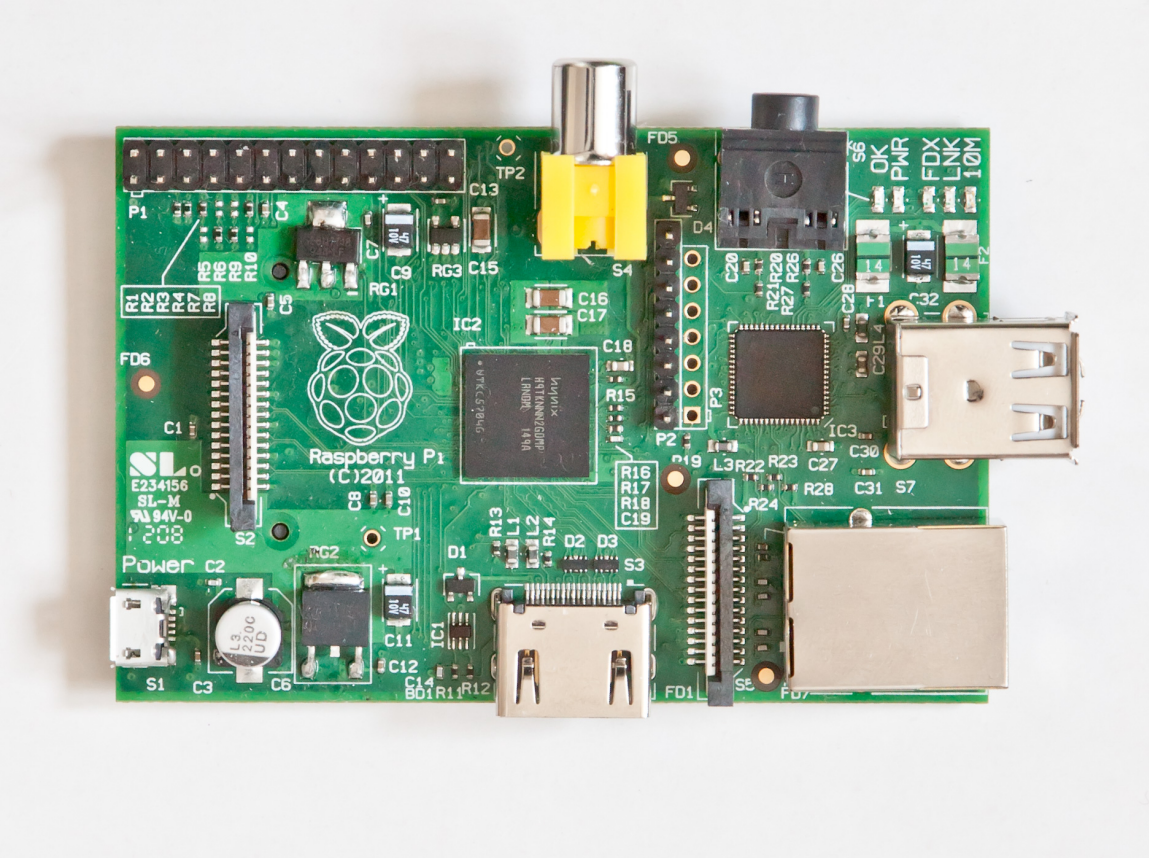



Figure 2.1: Raspberry Pi Model B board layout without a protective case

**Raspberry Pi Model B Case:**

Most Raspberry Pi's are not sold with a a protective 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 Pi board. Model A cases do not have a space for the camera to connect. This will make connecting the camera very difficult.

**Raspberry Pi Camera Module:**

The camera module is a camera that has been designed specifically for the Raspberry Pi in mind. It connects directly into the Pi to the CSI connector rather than USB by a ribbon cable. The CSI connector exclusively carries pixel data. This means that the camera module is capable of extremely high data rates and images would be sent to the Pi quicker then if a camera attached by USB would.

The camera itself is not expensive. It costs roughly 20 Euro and will work instantly once it is properly attached to the Pi board. To connect the camera the ribbon cable is simply inserted into the connector on the 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 capturing video in 1080p quality. If a web camera of this quality was to be used instead of this camera module for this this project it would be difficult to find one that would produce such high quality for such an affordable price and that can be hidden easily.

There is an infared 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

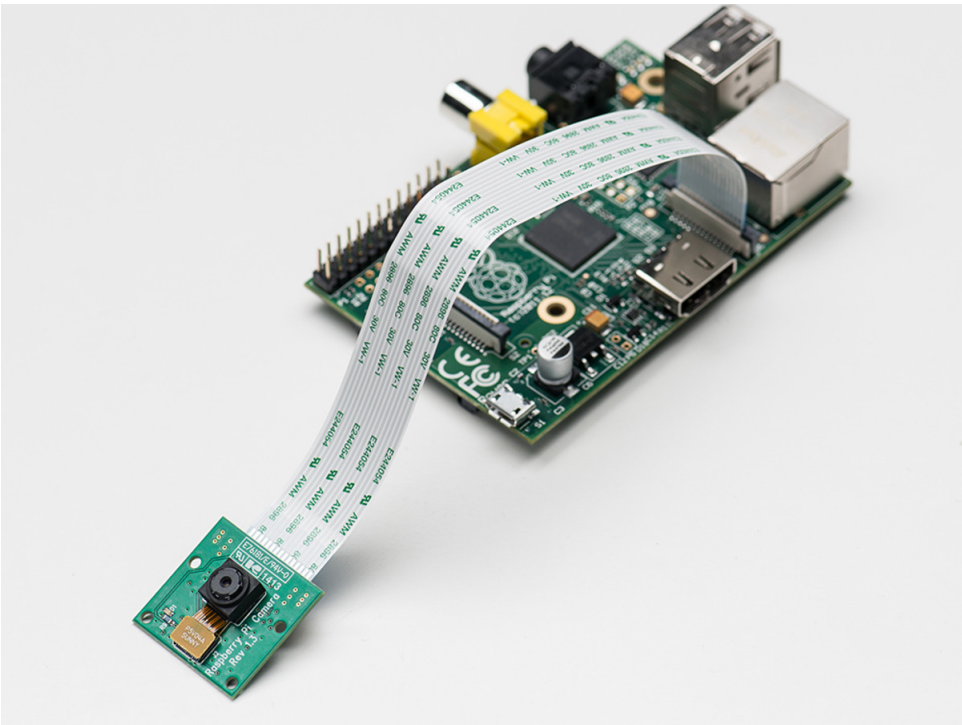


Figure 2.3: Camera Module attached to the Raspberry board with no casing

**USB Hub powered externally:**

A USB Hub is recommended when attaching any components to the Pi by USB. If any component is connected by USB in either of the two USB slots available on the Pi, they will draw power from the Pi itself. 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 or cause some features not to work properly. 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 recommended as any USB hub could be used.

**Wifi Adapter:**

When connecting to the internet it is recommended to use a Wifi Adapter as

it will be more useful then connecting to the internet by an Ethernet cable. By using an adapter it will avoid having to hide another cable and overall make this project 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 they must be bought separately. The Raspberry Pi has no on board memory so a form of storage is needed. 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 again that will be another decision left to the end user. The storage device chosen depends on how frequent the Raspberry Spi will be used and then amount of images a user wishes to keep stored.

## 2.2 Software

This chapter is about the software that will be needed throughout this project and why it is needed.

**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 that can be chosen. This project uses operating system 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.

Raspian is free to download and it is based off of the operating system Debian but it has been optimized to work on a Raspberry Pi.

**UV4L (Universal Video 4 Linux):**

The camera module has it's own default driver and it is called RaspiCam. RaspiCam does not work properly with OpenCV. Since OpenCV is used frequently throughout this project UV4L is needed.

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 is 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 will create a device node from the given driver Raspicam. OpenCV will not be able to tell the difference but it will be able to be found.

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:**

OpenCV is a free to use cross platform computer vision library that was created for the purpose of processing real time images and video.

For this project OpenCV will be used when taking a real time photo and during motion detection to determine if motion has actually occurred.

**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. Apache will run as soon as the Pi boots. This means that as long as the Pi is switched on the website will be available.

**Motion:**

Motion is an external program that will be installed and used to handle the video live streaming.

After Motion has been installed it 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.

Motion does have a motion detection feature that is available to use. This project will use OpenCV to take care of the motion detection and because of this that particular feature will not be used.

## 2.3 Additional Requirements

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

**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 unless port forwarding is set up.

A port is simply a communication channel or an endpoint. Port forwarding means that when a request is received by the router on a particular port it will know which device on the network to send the request on to.

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

A list of ports that need to be open will be provided. Ports for video streaming are the only ports that can be changed and port 22 is optional.

The following are ports that will be used:

| **Port** | **Port used for** |
|---|---|
| 80 | HTTP Requests (Website 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. Port 80 is responsible for all HTTP requests. 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. SSH means that they can use the IP address of the Pi and log into it from another machine. 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 which depending on where the Raspberry is set up could prove difficult.

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 in one of the web pages. Due to this using a browser that has JavaScript enabled needs to be used when viewing the website. The website will still appear but the function taken care of by the JavaScript will be unavailable without the user knowing.

This issues is only going to affect the user if they view the website on the Pi itself as most broswers for computers these days do in fact have JavaScript installed. If a user decides to view the website on the Raspberry Pi the only browser currently available and that has JavaScript enabled is Chromium.

# Chapter 3

# Implementation

## 3.1 Web Application

All of the pages on the website have been created using HTML and styled by using CSS. All pages use the same CSS style so that the website is consistent and the layout doesn't change as to not confuse the user.

Once port forwarding has been set up the Raspberry Spi web application will be available on the external IP of the home network. The page will not be accesible until a correct user name and password is entered.

**A 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 (Footer available on every page)

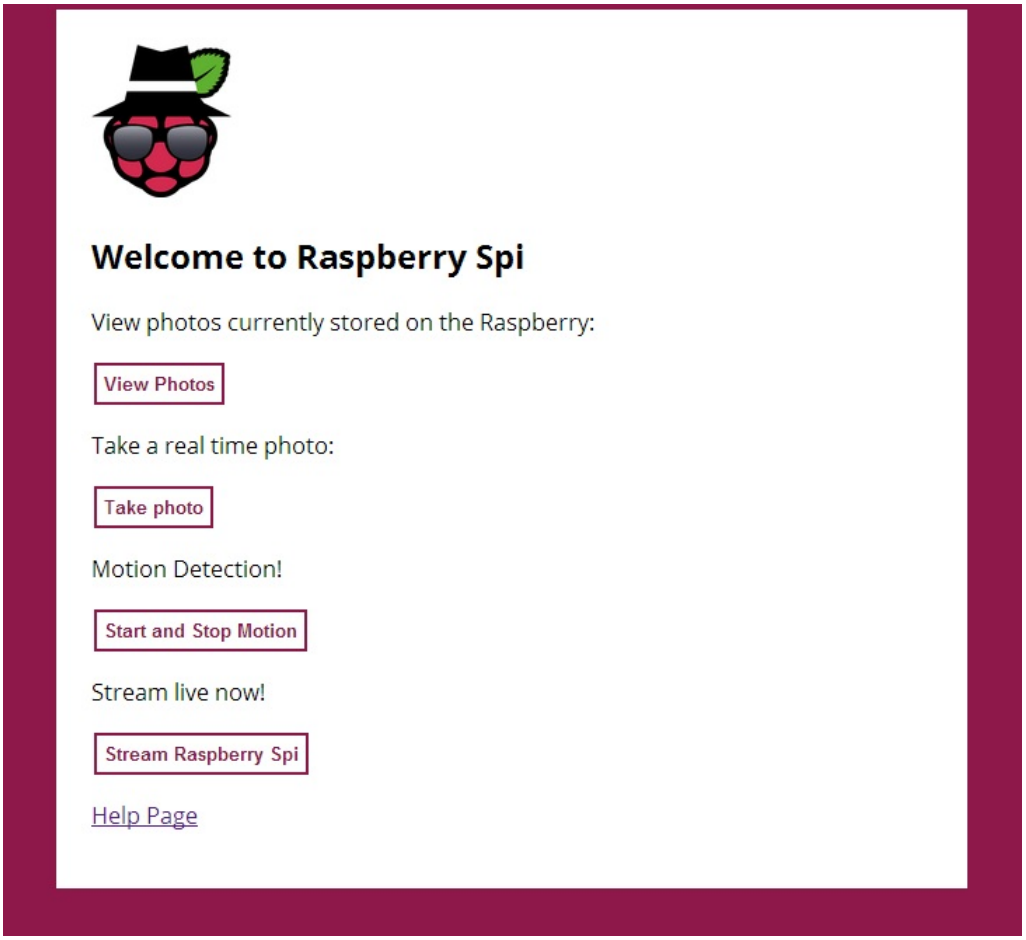



Figure 3.1: Home page of the web application with a list of available options

### 3.1.1 Accessing the Web Application

Accessing the website will require a user name and password to view any pages. The access of the site 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 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 by accident, viewed by accident or edited maliciously. The passwords and associated user names are stored in a .htpasswd. The passwords are encrypted and are not human readable. If a user would like to add a user name or password it will need to be added from the actual Pi not the web application. This is done to stop any fake attempts at accessing the site or spam requests being sent to the Pi. The web application will provide no hints to passwords or user names or any way to change either.

## 3.2 Help Page

A help page will be available in the footer on each page. It is there to give answers to 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 or a user would like to add to it.

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?

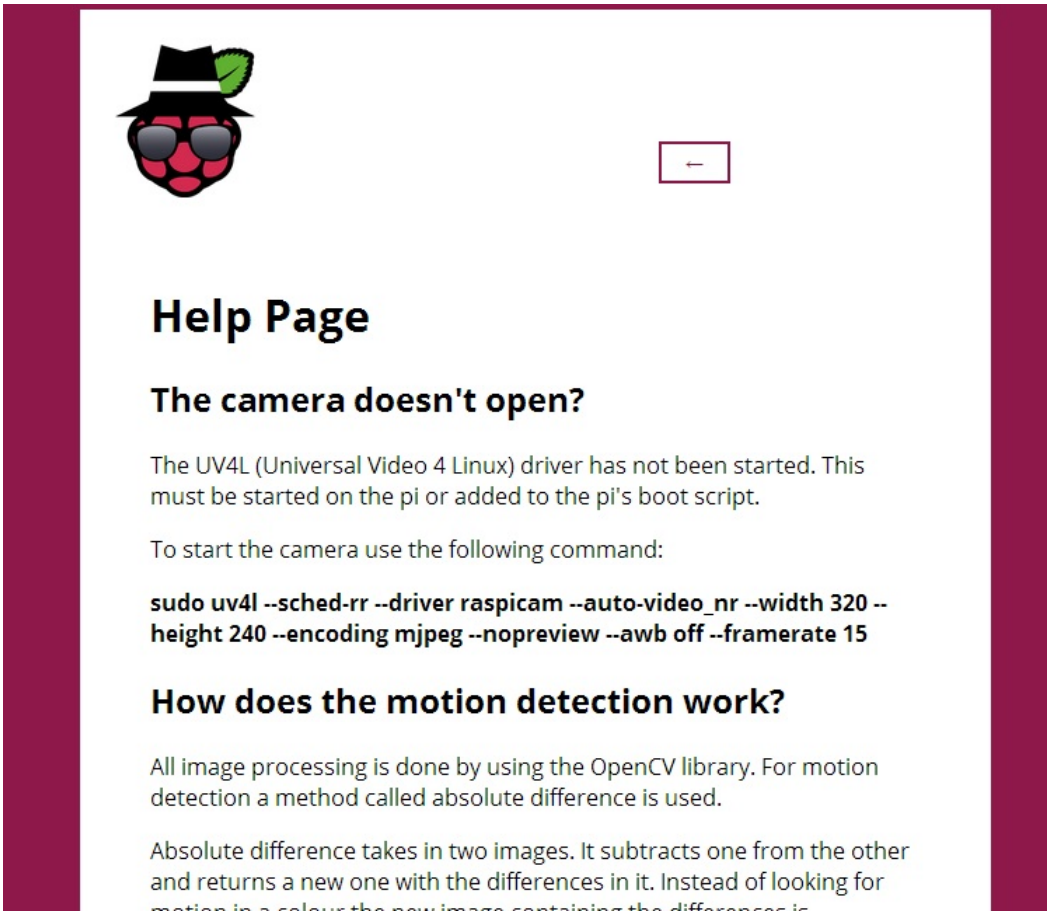



Figure 3.2: Help Page for the Raspberry Spi

## 3.3 Real Time Photo

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 on the web page. This photo is taken by using a Common Gateway Interface (CGI) script.

Common Gateway Interface 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. CGI scripts are stored in a .cgi file. For this project the CGI application is done using the language C++. When the C++ program is compiled the output of the program is set to a .cgi file

An example of this would be:

***g++ realTimePhoto.cpp -o realTimePhoto.cgi***

So when a user selects to take a real time photo that request is then sent to the Pi and it is asking to run the real time photo CGI script. The C++ program is then run and the results are returned and displayed on the web page. The layout and results shown to the user are taken from the C++ program.

**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 is will display the error to the user.

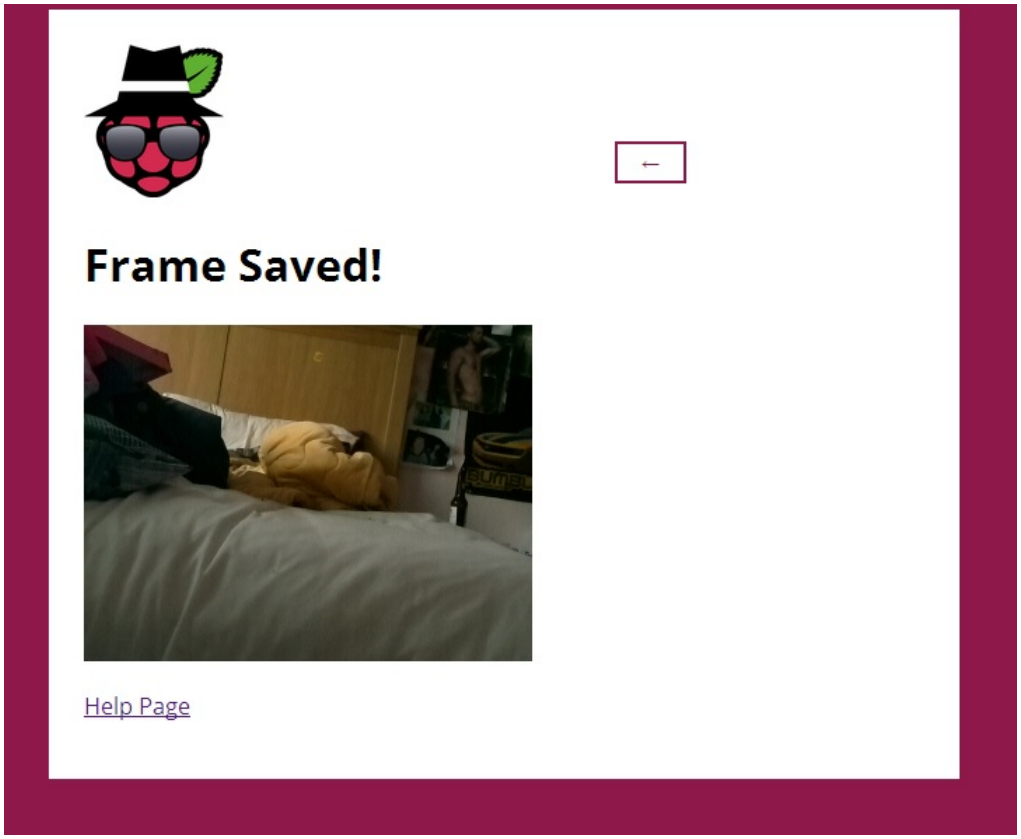



Figure 3.3: After taking a live frame

## 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 detection and stop motion detection. If motion detection is currently running it will be displayed to the user. There will be text displaying if the motion detection is on or off as well as a small icon which will appear. Depending on which option was selected a different icon will appear. These icons will

notify a user if starting or stopping motion detection was successful.

If a user does press start even though motion detection is running it does not affect the process in any way. Motion detection will continue to run as normal. This is also the same for pressing stop motion detection. If no motion detection is running pressing start will not cause any changes.

**Screen shots from the motion detection section on the web application**

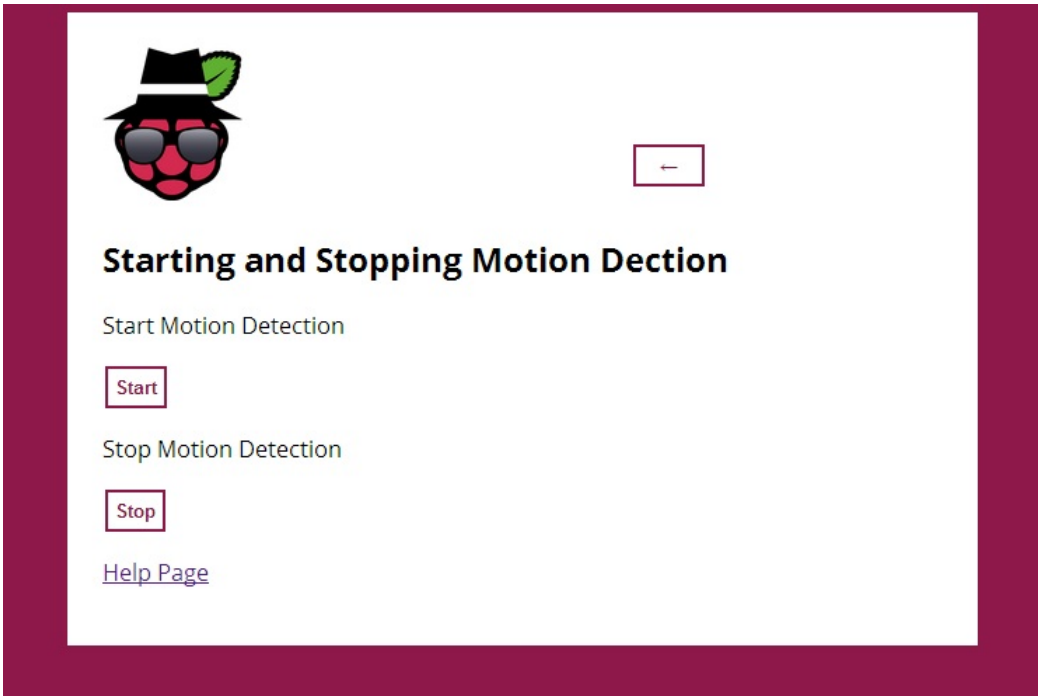



Figure 3.7: Motion Detection page

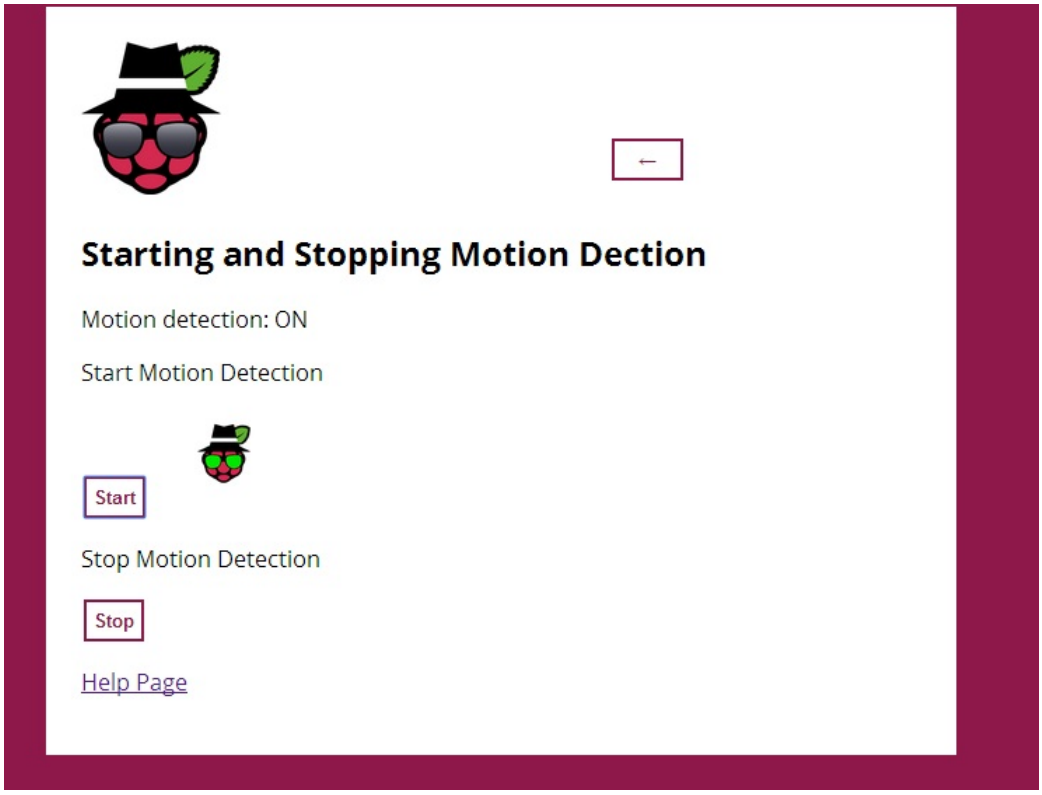



Figure 3.8: Motion Detection after being started

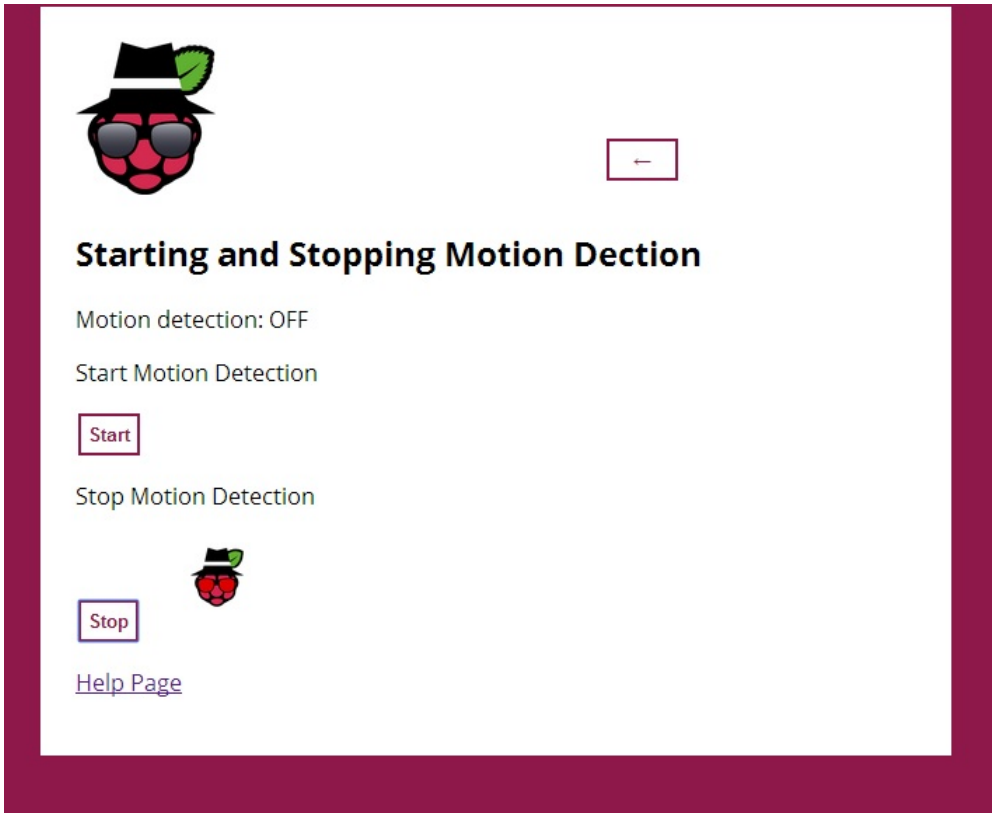



Figure 3.9: Motion Detection after being stopped

**Below are the icons in a larger form which are used to tell a user if motion detection is running or not.**

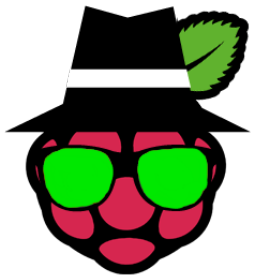


Figure 3.10: Motion Detection Start icon

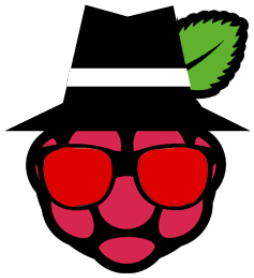


Figure 3.11: Motion Detection Stop icon

### 3.4.3 Back End of the Motion Detection Web Page

The back end of the motion detection web page is taken care of by a JavaScript function. This function will send an AJAX request to a PHP program when a button is pressed. PHP is a server side language so a request needs to be sent so that it will run. The PHP program will call the correct bash script which depending on the option selected will either start or stop motion detection.

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.

This motion detection JavaScript function is the reason why a JavaScript enabled browser is necessary.

## 3.5 Video Live Streaming

Motion is a program that is available to download and will be used to handle video live streaming. It sets up a video stream and will display the images that it receives from the camera attached to the Pi to a given web page on a particular port. It can also offer to do motion detection but since this has been implemented using OpenCV this aspect of Motion is not in use for this project.

Before Motion is used it needs to be set up by making changes to the configuration file. There are many changes that can be configured from how many frames it will display per second to the port the video is displayed on. Motion will take frames from the camera and display them onto the web page. This project is set up so that no live streaming is saved. This is done to avoid saving lengthy videos which would take up a lot of space to store which may not contain any useful information.

This feature currently does not work properly or efficiently. For future work this area could be greatly improved.

The video live streaming option on the home page of the web application does not start or stop streaming. The current problem with the live streaming is that when it is started it affects the website and sometimes the Pi itself connecting to the internet. It either causes the website to be entirely unavailable or disconnects the Pi from the internet. This could be due to Motion taking up a lot of memory on the Pi and the Pi simply can't handle it so it makes it unavailable.

# Chapter 4

# Testing

## 4.1 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.

## 4.2 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

## 4.3 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.

## 4.4 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.

## 4.5 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 built a new kind of security system that is affordable and easy to use but also effective. Raspberry Spi has achieved what was set out to create. It is an affordable set up that is both easy to use and 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 all from a web application that is successfully hosted on the Pi.

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 features can be changed if they do not act how an end user would like.

All of the Raspberry Spi code is completely open source and is available to download from github. This project has been made open source because it will allow for endless possibilities of additional extras to be added to it by other people. Many different people might see this project in a different way and could add to it their own completely unique and different ideas.
**Raspberry Spi Code is available here:**
http://www.github.com/Smellen/RaspberrySpi

## 5.1 Future Work

The video streaming does not work well or efficiently. To continue working on this project this area would involve either going into a completely new direction by trying a different way to do live streaming or continue using using the current method of the program Motion but try to implement it working better alongside Apache.

A different approach to this project would be to use the PiNoir camera which is specifically designed to only works at night time and has no infrared filter. The camera would connect in the exact same way the Raspberry Pi camera module does. It would be a Raspberry Spi project for during the night.

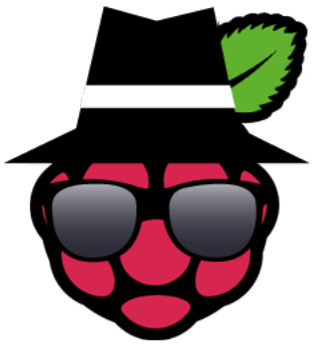
Lastly another small and simple feature of setting up notifications could be added in the future. When motion is started a user is most likely not going to be refreshing the web address where the photos are displayed waiting for some new images to appear. They would start motion detection, leave the web page and check back for any new images at a later stage. Notifications could be set up so if there is a lot of motion happening and being detected the Pi would simply send a notification of an email to the user that something has been detected and a photo of the motion that was captured could be attached .

## 5.2 Expansion Work

Raspberry Spi has been created as a system that is to be used in a home. But it could be set up in other places that could use features like motion detection, live video streaming and storing this information. An area like farming could use a project like this to keep an eye on a livestock building. A building of that size couldnt be monitored at all times but the Raspberry Spi would allow for monitoring to happen while there are no persons catering to the building.

Further expanding this project would be adding in more Raspberry Pis to the system. One Pi per room or as many as preferred. One Pi would act as a central server controlling the web pages and the other Pis on the system

would handle taking photos, motion detection and live streaming.



# Bibliography

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