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

Permission to Lend

I agree that the Library and other agents of the College may lend or copy this thesis upon request.

Ellen Marie Burke 23rd April 2014

Acknowledgements

I would like to thank my supervisor Fergal Shevlin for all his help throughout this project, Arissa Brown for reviewing my website and answering all my web development questions, Christina Lynch for proofreading this report and fixing my English, my cat Shinique for helping with motion detection testing, my family and everyone else who helped me get through this project by giving feedback or testing this project.

Abstract

Security systems are more frequently being set up in homes in current times due to the technology available. They can be bought pre built or created at home. Not all are being used for security purposes some are simply set up to check on pets while nobody is home throughout the day or for checking on small children without being intrusive while they are playing or sleeping.

This project will implement the most popular features such as motion detection, live streaming and media storage into an affordable system that can be set up in a home by a user enabling them to have full control over the system which allows for customization of features or adding on additional features.

Contents

1 Introduction **3**
- 1.1 Motivation/Problem . . . 3
- 1.2 Background Research . . . 4
  - 1.2.1 Mobile App or Web Application . . . 5

2 Design **6**
- 2.1 Hardware . . . 6
  - 2.1.1 Raspberry Pi Model B . . . 7
  - 2.1.2 Raspberry Pi Model B Case . . . 9
  - 2.1.3 Raspberry Pi Camera Module . . . 9
  - 2.1.4 USB Hub Powered Externally . . . 11
  - 2.1.5 Wifi Adapter . . . 12
  - 2.1.6 SD Card . . . 12
- 2.2 Software . . . 13
  - 2.2.1 Operating System - Raspbian . . . 13
  - 2.2.2 OpenCV . . . 14
  - 2.2.3 UV4L (Universal Video 4 Linux) . . . 14
  - 2.2.4 Apache . . . 14
  - 2.2.5 Motion . . . 15
- 2.3 Additional Requirements . . . 16
  - 2.3.1 Port Forwarding . . . 16
  - 2.3.2 A Browser with JavaScript Enabled . . . 17

3 Implementation **18**
- 3.1 Web Application . . . 18
  - 3.1.1 Accessing the Web Application . . . 19
- 3.2 Help Page . . . 20
- 3.3 Real Time Photo . . . 22

3.4 Motion Detection 24
3.4.1 How Motion Detection Works 24
3.4.2 Front End Motion Detection Web Page 26
3.4.3 Back End Motion Detection Web Page 28
3.5 Video Live Streaming 29

4 Testing **30**
4.1 Testing Motion Detection 30
4.2 Testing Raspberry Spi Web Application 31
4.3 Testing Real Time Photo 31
4.4 Testing Live Streaming 31
4.5 Testing Photo Storage and Timestamps 32

5 Conclusion **33**
5.1 Future Work 34
5.2 Expansion Work 34

DVD containing all project code **34**

Chapter 1

Introduction

Home security systems have been available for a number of years. They can be either pre built or created at home, but most can be quite expensive and can be easily visible around the household. In either set up the camera utilized may not have the potential to transmit 1080p video which would enable for high definition imagery, without the high definition imagery features on the system may not be effective or easy to get information from. Live streaming footage may be pixelated causing frames to be unclear and movement unclear or images detected by motion detection could be useless as nothing can be seen clearly.

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 users will have the ability to install the system and control it. It has a wide range of possibilities for adaptation allowing the user to change the current settings or extend the functionality.

In comparison to standard security systems, the size, cost and user control capability of The Raspberry Spi ensures that it is an excellent substitute. For the end user, the cost of the Pi is approximately 30 Euro and the camera module 20 Euro. The size of the Pi is significantly smaller than standard se-

curity systems with dimensions of 85.60mm x 56mm x 21mm and the camera module would be a minor addition.

There is a wide range of real life applications of this project. It is specifically aimed at uses within the home but can easily be adapted to other locations. The main uses for the application are:

- Allowing the user to observe pets within the home while out working or not in, by accessing through the web interface.
- 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 playing.
- 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 implemented in 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 as 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 give 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 for pre built systems could be difficult when setting up or controlled 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 would be. Like pre built systems the user may not have as much control over them depending on the software used. If pre installed software is used this may not allow a user to customise features to how they want them to behave.

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, the users would only require an internet browser.

Chapter 2

Design

One of the main aims of this project is to make it easy for users to set up regardless of their background. With this in mind there are relatively few hardware components required for the Raspberry Spi such as the Raspberry Pi Model B itself, Raspberry Pi Model B case, Raspberry Pi Camera Module and the SD card.

Additional items such as a keyboard and mouse are only needed for initial set up and depending on the users preference of how they will access the Pi to make changes. Some hardware components are also just additional extras for more control of the Raspberry Spi. There are also suggested hardware items such as a USB hub powered externally and a Wifi adapter should the user feel they are required.

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

Below is the listed components and how they will be used for this project.

2.1.1 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, 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

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

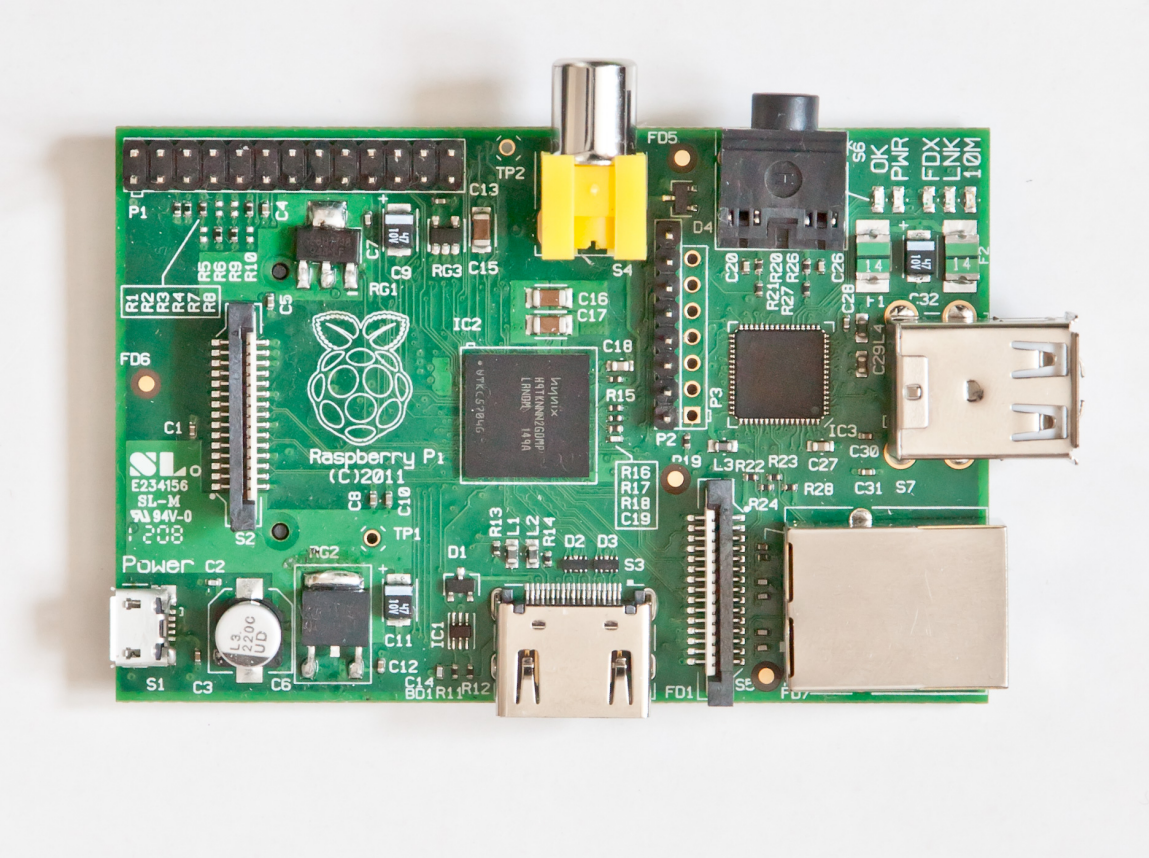


Figure 2.1: Raspberry Pi Model B board layout without a protective case[27]

2.1.2 Raspberry Pi Model B Case

Most Raspberry Pi's are not sold with a protective case. The case is essential because otherwise the Pi will have no protection and can get damaged very easily.

The other advantage to the Model B case is that there is space for the camera module to attach to the CSI connector on the Pi board. Model A cases do not have a space for the camera to attach to the CSI connector and this makes attaching the camera very difficult.

2.1.3 Raspberry Pi Camera Module

The camera module is a camera that has been designed specifically with the Raspberry Pi in mind. It attaches directly into the Pi to the CSI connector by a ribbon cable rather than USB. The CSI (Camera Serial Interface) connector exclusively carries pixel data. This means that the camera module is capable of extremely high data rates and images would travel to the Raspberry Pi quicker then a USB camera.

The camera itself is not expensive and costs approximately 20 Euro. It will work instantly once it is properly attached to the Pi board. To connect the camera, the ribbon cable is simply inserted into the CSI connector on the Pi board. The camera is capable of outputting images and video of very high quality, the max resolution the camera is capable of is 2592 x 1944. For video the camera is capable of capturing footage in 1080p quality.

For this project if a web camera of this quality was to be used instead of this camera module it would be difficult to find one to produce the same high quality at an affordable price that can be hidden easily.

There is an infrared (IR) filter on the camera and therefore it cannot detect IR. IR is not visible by the human eye it is a form of light emitting from hot objects and it has a shorter wavelength than visible light. There is another camera module called the Pi Noir which is available and it can detect IR. Using this camera would allow for night time surveillance.

Figure 2.2: Raspberry Pi Camera Module[28]

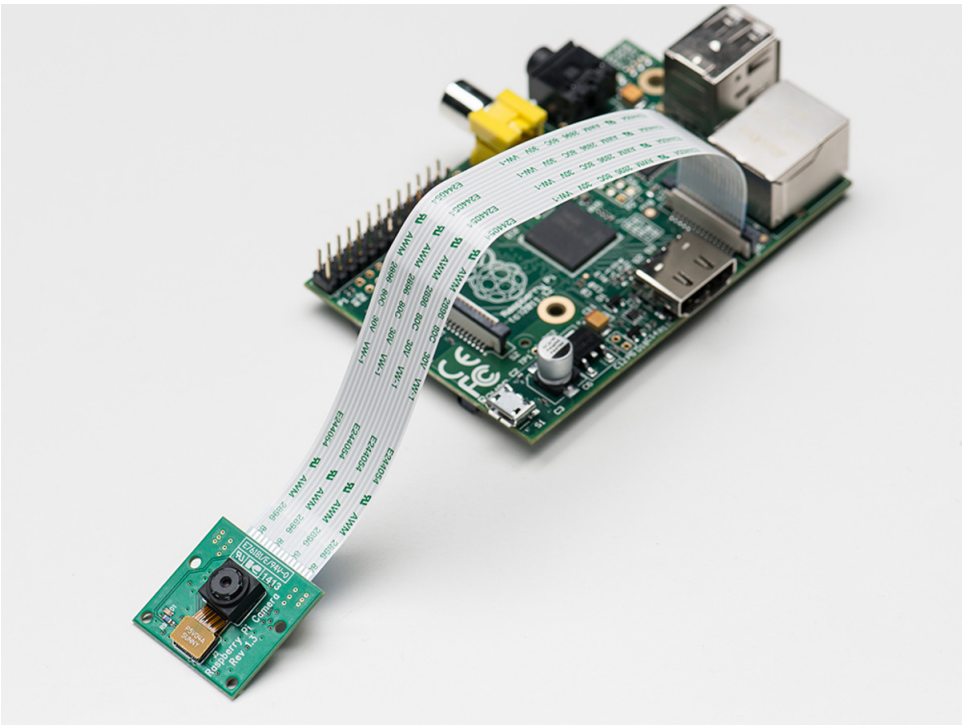


Figure 2.3: Camera Module attached to the Raspberry board with no casing[29]

2.1.4 USB Hub Powered Externally

A USB Hub is a device that expands one USB slot into several depending on how many are available on the device and is recommended when attaching any components to the Pi by USB. If any component is connected to the two USB slots on the Pi, it will draw power from the Pi itself and could cause the Pi to heat up and some features not to work properly.

If the Raspberry Pi is left on for several hours, as is the aim of this project, overheating may occur resulting in some features not working properly.

An externally powered USB hub will allow for the Pi to avoid overheating and create extra USB slots without affecting the Pi while it is operating.

There is no prerequisite USB Hub to be used.

2.1.5 Wifi Adapter

When connecting to the internet a Wifi Adapter is recommended as it will be more useful than connecting to the internet by an Ethernet cable. By using an adapter it would avoid hiding another cable and overall make this project neater to set up.

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

2.1.6 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 internal memory so an external storage component is required. An operating system will be loaded onto it and will also store all captured images.

Users have the option of implementing external hard drives if required for storing images.

2.2 Software

This section will discuss the software that will be needed throughout this project and it's requirements.

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

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

Below is the listed software required and what they will be used for in this project.

2.2.1 Operating System - Raspbian

With the Raspberry Pi there are several operating systems that can be chosen. This project uses the operating system (OS) Raspbian because it is the most popular OS for the Pi. As it is easy to debug problems and solve issues that arise due to the large amount of online resources.

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

2.2.2 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.

2.2.3 UV4L (Universal Video 4 Linux)

The camera module has it's own default driver and it is called RaspiCam. RaspiCam does not integrate fully with OpenCV. Since OpenCV is required 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' and because of this OpenCV cannot use the camera module with it's default drivers. To resolve this driver issue UV4L (Universal Video 4 Linux) is used instead. UV4L will create a device node from the given driver Raspicam. OpenCV will be unable to notice any difference but it will be able to find and access the camera module.

When using this driver the UV4L initializing script must be executed, by adding it to the Raspberry Pi's start up script or it can be called after each boot. The parameters passed in can be interchangeable.

2.2.4 Apache

Apache web hosting will be used to host the Raspberry Spi website. Apache will be permanently running as a background process. While the Raspberry Pi is handling motion detection and taking photos the website will be accessible and process requests received. Apache will be available, as soon as the Pi boots, this means that as long as the Pi is switched on the website will be available for users to access.

2.2.5 Motion

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

After Motion has been installed it can be configured in multiple ways, from which port the video will stream, how many (Frames Per Second)FPS are displayed, to whether the live stream footage should be saved and where to 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 therefore the motion detection feature will not be implemented.

2.3 Additional Requirements

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

2.3.1 Port Forwarding

With the Raspberry Spi being set up on a home network, port forwarding needs to be enabled. This is a requirement to ensure this application can work.

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 access the many online resources for further information.

A list of ports that need to be opened will be provided. Ports for video streaming are the only ports that can be changed and port 22 for SSHing 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 (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. This is responsible for all HTTP requests. With Port 80 open when the router receives a HTTP request it will know which device on the local network to send the request on to.

Port 22 is optional and is dependant on the end users requirements to interact or make changes on the Raspberry Pi itself. If Port 22 is open this will allow a user to SSH (Secure Shell) directly into it. SSH means that they can use the IP address of the Raspberry Pi and log into it from another machine. SSHing into the Raspberry Pi will allow the user to make changes from any computer instead of using the Raspberry Pi and having to attach a keyboard and mouse which depending on where the Raspberry Pi is set up could prove to be difficult.

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

2.3.2 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 certain features will be unavailable without the user knowing. This issue will only affect the user if they view the website on the Raspberry Pi itself as most browsers for modern day computers have JavaScript pre-installed. If a user decides to view the website on the Raspberry Pi the only browser currently available that has JavaScript enabled is Chromium.

Chapter 3

Implementation

3.1 Web Application

The website has been implemented using HTML and styled using CSS to ensure consistency. All pages use the same CSS style for easier usability and navigation.

Once port forwarding has been implemented the Raspberry Spi web application will be available on the external IP of the home network. The page will not be accessible 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 and Stop Motion Detection
- Video Live Streaming
- Help Page (Footer available on every page)

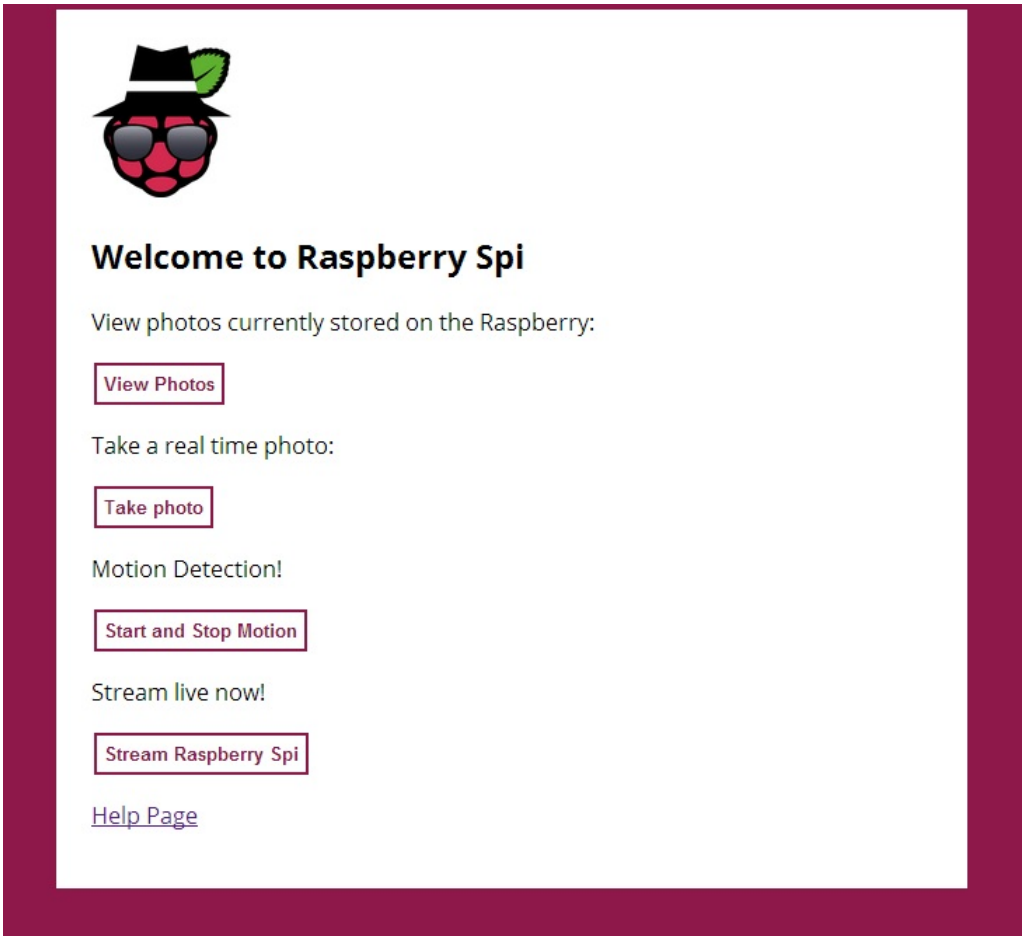


Figure 3.1: Home page of the web application with a list of available options

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. This file stores 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 stored separately from the files and directories for the web

pages. This prevents them from being accessed by accident, viewed by accident or edited maliciously. The passwords and associated user names are stored in a .htpasswd file. 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 Raspberry Pi not the web application. This is done to stop any phishing attempts at accessing the site or spam requests being sent to the Raspberry Pi. The web application will provide no hints to passwords or user names or options to change either.

3.2 Help Page

A help page will be available in the footer on each web 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 more information to it.

The questions on the help page currently are:

- The camera doesn't open?
- How does the motion detection work?
- The web page is unavailable?
- When taking a live frame the web page 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?

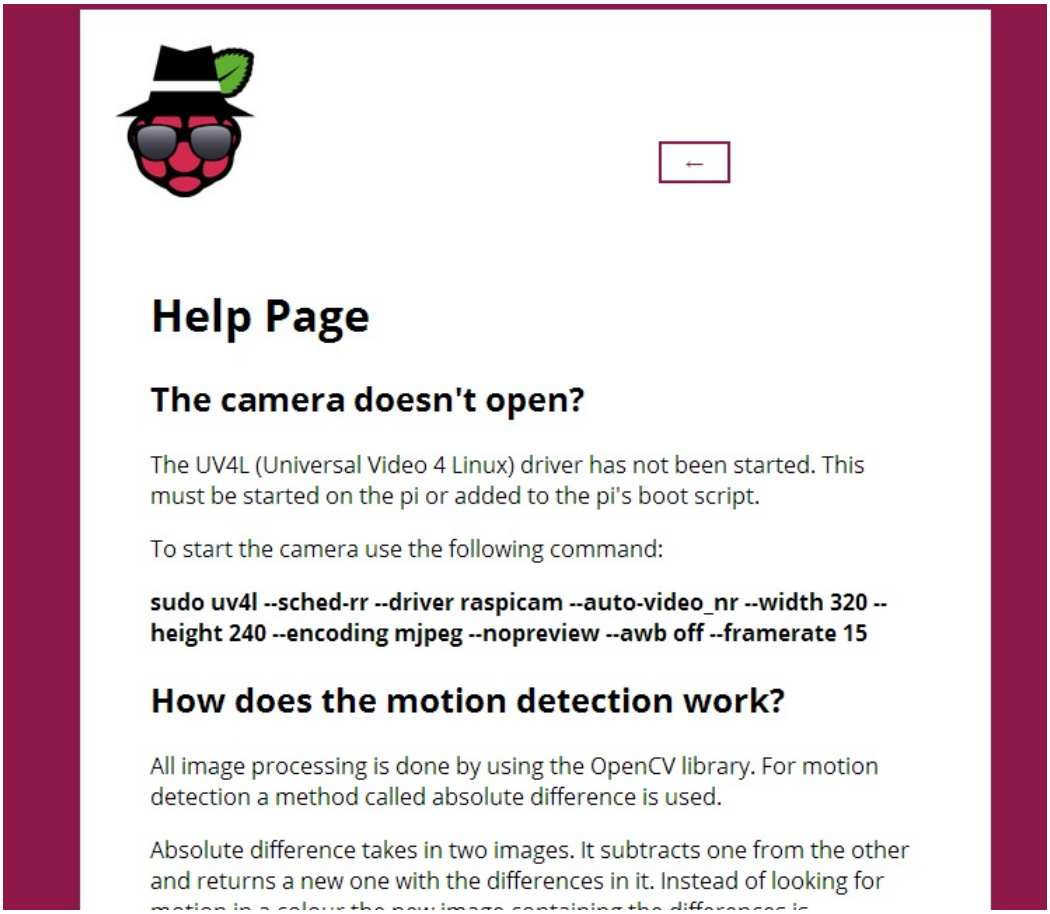


Figure 3.2: Help Page for the Raspberry Spi

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 that request server side. CGI scripts are stored in a .cgi file with the language C++ being utilized in this project. When the C++ program is compiled the output of this program is sent to a .cgi file.

An example of this would be:

g++ realTimePhoto.cpp -o realTimePhoto.cgi

When a user selects to take a real time photo, that request is then sent to the Raspberry Pi and it is asking to run the real time photo CGI script. The C++ program is then executed 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.

Any changes that are made to the C++ program will require compilation to happen again.

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, the web page will just display what has gone wrong.

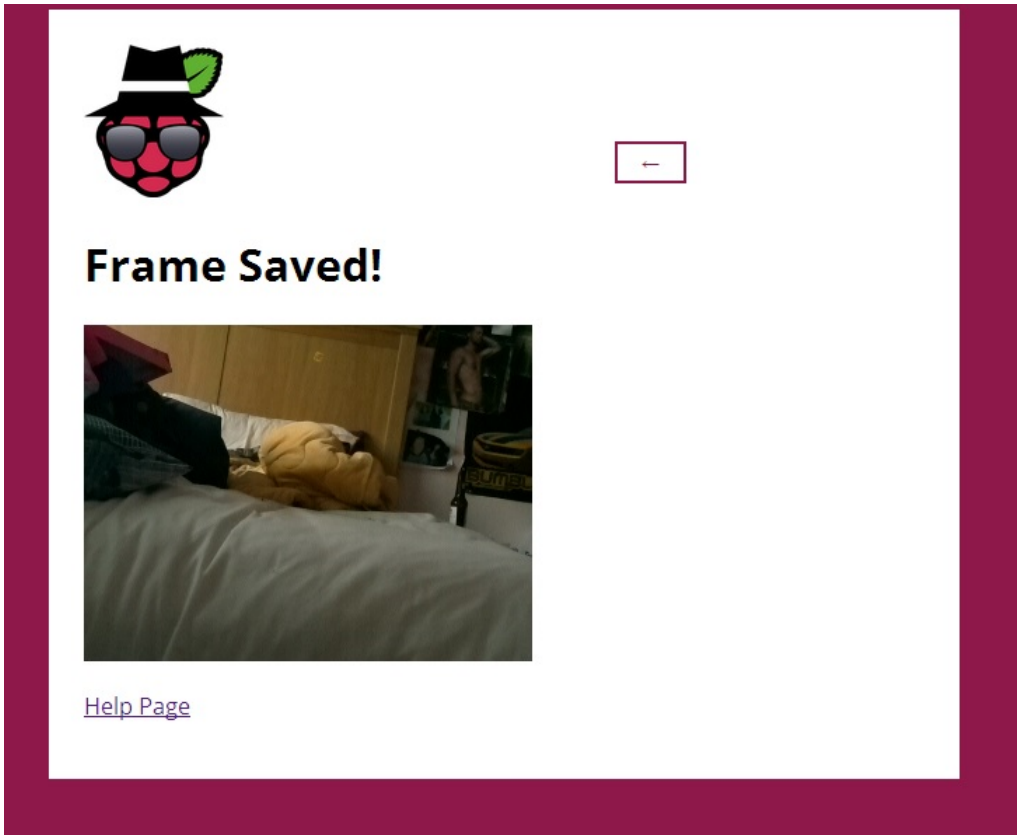


Figure 3.3: After a live frame has been taken

3.4 Motion Detection

3.4.1 How Motion Detection Works

Motion detection is achieved by using the OpenCV library. The method chosen for motion detection is Absolute Difference. This method will detect changes happening after frames are read in. This method was chosen because it is effective at detecting motion while not overpowering the Raspberry Pi.

OpenCV has an inbuilt function called *absdiff()*. This method takes in two coloured (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 set threshold then motion has been detected.

Pseudo code for motion detection:

absdif(prevFrame - nextFrame = difference)
grayscale(difference)
checkWhiteCount (difference)
if WhiteCount is above threshold save images

If motion is detected the RGB frames that were originally read in are saved. Since the difference image will not mean anything it will not be saved. If motion is not detected then the images are not saved and new images are read in and the process will repeat.

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 frame 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

Total time for this process to occur once defaults to 5FPS. Motion detection will skip every three frames read in but this can be changed.

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 that appears. Depending on the selected option 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 stop will not cause any changes.

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

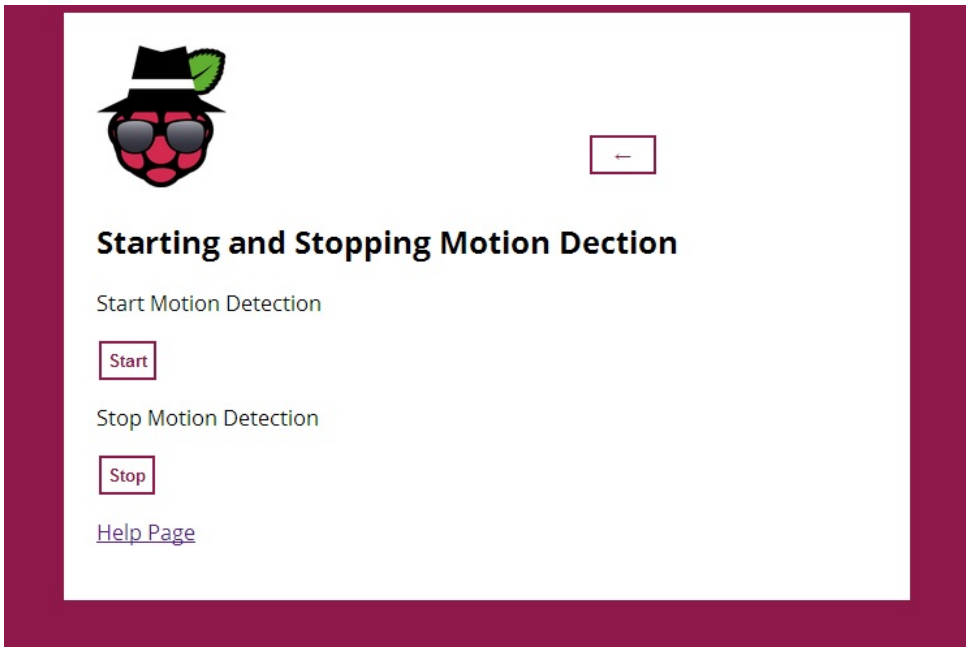


Figure 3.7: Motion Detection page

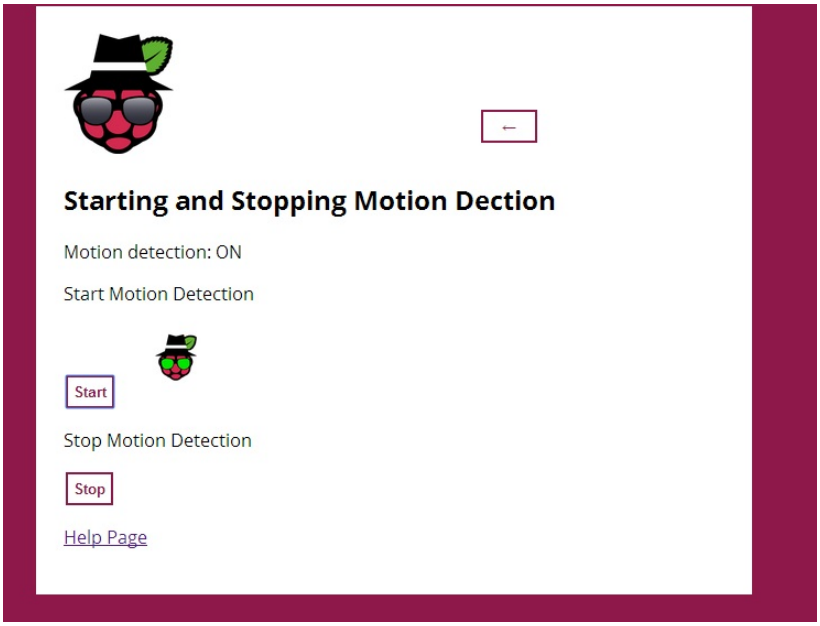


Figure 3.8: Motion Detection after it is started

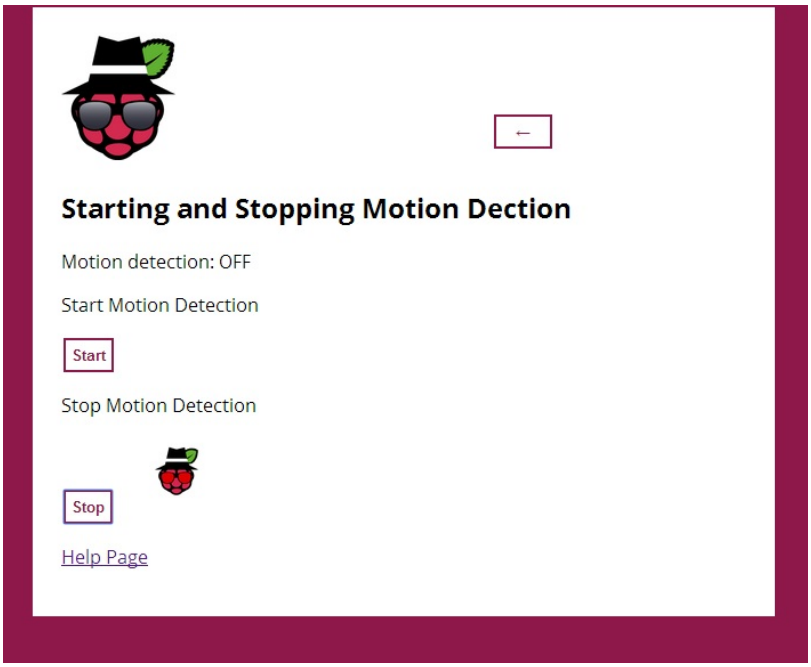


Figure 3.9: Motion Detection after it is stopped

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

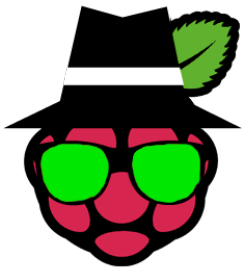


Figure 3.10: Motion Detection Start icon

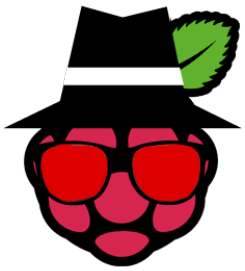


Figure 3.11: Motion Detection Stop icon

3.4.3 Back End Motion Detection Web Page

The back end of the motion detection web page is controlled 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 is required from the application to execute the PHP program. 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 and 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 Raspberry 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.

Parameters in the Motion config must be edited before runtime, an example of this would include 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 on to the web page. To avoid saving large video files that may not contain any useful information from taking up space, no live streaming imagery is stored. This feature currently does not work properly or efficiently. Future work for this project would be to tune this feature to ensure more accuracy and efficiency.

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 from 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 or the Pi simply can't handle it so it makes it unavailable.

Chapter 4

Testing

4.1 Testing Motion Detection

Testing for this application was highly complex to ensure the Raspberry Pi was not overpowered resulting in it being unable to process anything else. Motion Detection on a computer, not a Raspberry Pi, would in general read in every frame from a video stream and process each of the frames looking for motion. The Raspberry Pi is capable of such a task but for this project it is expected to keep a website running and be able to process all the requests that it receives as the motion detection is running in the background. For 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.

To test motion detection the object of a book was used. The book was held in front of the camera and moved from one side of the frame to the other. The test started at skipping ten frames which means that each following tenth frame was used. Skipping ten frames was unsuccessful as it occasionally only caught the end of any motion that was happening. Skipping ten frames was reduced down to skipping five frames and the test was repeated. It proved to work much better but it still did not catch all the motion that was happening. The frames being skipped was reduced down to three frames. This test proved to have the highest success rate for detecting motion and at the same time 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 caused a delay in the web pages being displayed. This proved that it was necessary to skip a certain set of frames.

4.2 Testing Raspberry Spi Web Application

Once motion detection was found to work correctly and as planned the website was tested to ensure if the Raspberry Pi was 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 in front of the camera so that motion detection would start saving images. From this test, motion detection captured images were successfully saved, no users had any issues or delays logging in or in accessing the photos.

4.3 Testing Real Time Photo

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

4.4 Testing Live Streaming

Live Streaming was tested on a localhost connection rather than through the web application. A subject walked across the screen to ensure the stream was live and the correctness of the FPS. This was a successful test.

When testing live streaming with the website it caused several issues such as taking down and stopping Apache, causing the web page to be unavailable and effecting the Wifi connection on the Raspberry Pi.

4.5 Testing Photo Storage and Timestamps

Real time photos were taken and the time was recorded so it can be cross checked with the timestamp on the photo. The web address where all photos are stored was checked to ensure the image appeared with the correct time stamp and the expected photo. All photos taken from either feature appeared in a swift manner and available for viewing, timestamps and dates on each photo were also always correct.

Chapter 5

Conclusion

The aim of this project was to build 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 Raspberry Pi.

Various changes can be made to match each user depending on their preference. It can have multiple real life applications for varying end users. Setting it 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 are fully customizable to the end users requirements.

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 may see this project in a variety of ways and could add their own completely unique and separate ideas.

Raspberry Spi Code is available here:
http://www.github.com/Smellen/RaspberrySpi

5.1 Future Work

The video streaming is not yet fully implemented or optimized. To further develop this area of the project would require a redesign of the live streaming implementation or further optimize the current Apache integration with the program Motion.

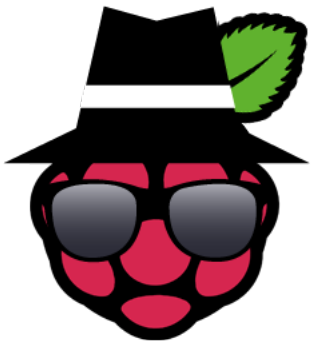
An alternative approach to this project could be to use the PiNoir camera which is specifically designed to only work at night time and has no infrared filter. The camera connection would be identical to the Raspberry Pi Camera Module. This would adapt the Raspberry Spi project for night time surveillance.

Additional proposed features, after user feedback would be the implementation of notifications. This could alert the user to the new images being captured to avoid manual refreshing of the web site. 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 being detected the Raspberry Pi would simply send a notification by 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 for use in the home, but it could be modified to work 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 shelter for livestock. A building of that size couldn't be monitored at all times, but the Raspberry Spi would allow for monitoring to happen while there is no one to keep watch on the building.

Further expanding of this project would involve adding in more Raspberry Pi's to the system, creating a distributed system. One Raspberry Pi per room or as many as preferred, one Raspberry Pi would act as a central server controlling the web pages and the other Raspberry Pi's on the system would handle taking photos, motion detection and live streaming.



Bibliography

[1] Ry Crist, *DIY security options offer smarter peace of mind*, (2013), available at http://www.cnet.com/uk/news/diy-security-options-offer-smarter-peace-of-mind/.

[2] ezwatch.com, *How to Wire and Power a Security Camera*, (2014), available at http://www.ezwatch.com/security-cameras/wiring-power-security-cameras.

[3] Gregory Karp, *Alarms should sound on deal*, (2012), available at http://articles.chicagotribune.com/2012-05-11/site/sc-cons-0510-karpspend-20120511_1_burglar-alarms-home-security-carbon-monoxide-alarms.

[4] Security Camera Warehouse, *Security Camera Systems*, (2014), available at http://www.security-camera-warehouse.com/security-camera-system/.

[5] Alex Nikolaidis, *How to make a DIY home alarm system with a raspberry pi and a webcam*, (2013), available at https://medium.com/random-things/2d5a2d61da3d.

[6] Mikael Albrecht, *Make your own DIY surveillance system*, (2013), available at http://safeandsavvy.f-secure.com/2013/06/06/make-your-own-diy-surveillance-system/#.U1Z7to-h5wA.

[7] element14.com, *Plug Into Your World with the Raspberry Pi*, (2014), available at http://www.element14.com/community/community/raspberry-pi.

[8] Ken Hess, *Raspberry Pi: How I spent almost $150 on a $35 computer*, (2013), available at http://www.zdnet.com/raspberry-pi-how-i-spent-almost-150-on-a-35-computer-7000020574/.

[9] elinux.org, *RPi Hub*, (2014), available at http://elinux.org/RPi_Hub.

[10] Lauren Orsini, *12 Cool Projects For Your Raspberry Pi*, (2014), available at http://readwrite.com/2014/01/21/raspberry-pi-great-projects#awesm=~oCaIqfhOd63QUU.

[11] raspberrypi.org/, *Raspberry Pi*, (2014), available at http://www.raspberrypi.org/.

[12] raspberrypi.org/, *Raspberry Pi OpenCV*, (2014), available at http://www.raspberrypi.org/tag/opencv/.

[13] Pierre Raufast /, *OpenCV and Pi Camera Board !*, (2013), available at http://thinkrpi.wordpress.com/2013/05/22/opencv-and-camera-board-csi/.

[14] Robert Castle /, *Installing OpenCV on a Raspberry Pi*, (2014), available at http://robertcastle.com/2014/02/installing-opencv-on-a-raspberry-pi/.

[15] Cdric Verstraeten /, *Install OpenCV on a Raspberry Pi*, (2014), available at http://blog.cedric.ws/install-opencv-on-raspberry.

[16] letsmakerobots.com /, *OpenCV on Raspberry Pi*, (2014), available at http://letsmakerobots.com/node/36947.

[17] OpenCV /, *All OpenCV references*, (2014), available at http://docs.opencv.org/.

[18] Peter Kirn /, *All OpenCV references*, (2009), available at http://createdigitalmotion.com/2009/02/processing-tutorials-getting-started-with-video-processing-via-opencv/.

[19] Peter Kirn /, *Setting up Wireless motion-detect cam*, (2013), available at http://rbnrpi.wordpress.com/project-list/setting-up-wireless-motion-detect-cam/.

[20] Mayank Sharma /, *Use the Raspberry Pi as a DIY Surveillance camera*, (2014), available at http://www.maketecheasier.com/raspberry-pi-as-surveillance-camera/.ra/

[21] www.w3schools.com /, *HTML*, (2014), available at http://www.w3schools.com/html/.

[22] www.w3schools.com /, *JavaScript*, (2014), available at http://www.w3schools.com/js/.

[23] www.w3schools.com /, *AJAX*, (2014), available at http://www.w3schools.com/ajax/.

[24] www.w3schools.com /, *PHP*, (2014), available at http://www.w3schools.com/PHP/.

[25] tldp.org /, *Bash Scripting*, (2014), available at http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html.

[26] UV4L/, *UV4L*, (2014), available at http://www.linux-projects.org/modules/sections/index.php?op=viewarticle&artid=14.

[27] readwrite.com /, *Raspberry Pi Model B board layout without a protective case*, (2014), available at http://readwrite.com/2014/01/20/raspberry-pi-everything-you-need-to-know#awesm=~oBwfeWDLWC0rMX.

[28] adafruit.com /, *Raspberry Pi Camera Module*, (2014), available at http://www.adafruit.com/products/1367.

[29] embeddedcomputer.nl /, *Camera Module attached to the Raspberry board with no casing*, (2014), available at http://embeddedcomputer.nl/media/catalog/product/cache/1/image/650x650/9df78eab33525d08d6e5fb8d27136e95/r/a/raspberry_pi_camera_board.jpg.