Pi-Surveillance System Interim Report

DT211C/4

BSc. In Computer Science Infrastructure

Dean Ryan
Aneel Rahim

December 23rd 2015

Table of Contents

1. Project statement

The aim of this project is to develop an application for both web and android devices that will provide a live stream of video surveillance from a camera attached to a raspberry pi server. The project will be utilized by users who wish for a more affordable home monitoring system. The system includes a Raspberry Pi server a Raspberry Pi Camera, an application developed for android devices and a web application provided for media playback and streaming. An external storage device will be used for saving video and image files and an online database will be implemented to store user account credentials.

Project features include:

- Motion Detection
- Network Scanning
- Zoom Features
- Media Storage
- Notifications

2. Research

The following section provides information gathered from technologies and similar systems in an effort to achieve a more feasible project. The information obtained will be divided into subsections focusing specifically on the systems architecture, its relevant features and its functional and non-functional requirements. Nielsen's heuristics for interaction design will be practiced in order to correlate the associated systems with the apposed system [1].

2.1 Background research

In 1933 the world's first security camera was invented by an amateur photographer who went by the name Mr. Norbury. Whilst his invention was largely ignored around the time of creation, he has since been hailed as the father of camera surveillance. Using an ordinary box camera placed inside a chicken hut, he was able to capture pictures of the man who was stealing eggs for weeks prior to the capture. The photos were later used in court and justice was served [2]. It wasn't until the late 1940's that the systems were used on a commercial bases. Systems consisted of cameras and monitors that could only be used for live monitoring.

Figure 2.1: Box Camera

Eventually reel to reel systems were developed and recording was made possible. It wasn't until the emergence of the video cassette recordings (VCR) where surveillance systems became widely used. Users could review recorded footage whenever needed. Systems involved the use of analogue cameras which needed coax cables that had to be directly connect to the VCR.

Since then, the use of video surveillance, more commonly known as CCTV (closed-circuit television), has become very popular. We have seen solutions like multiplexing where syncing and recording between multiple cameras on one screen has been made possible.

Digital video recordings (DVRs) was also a huge step for CCTV as it was more user friendly and meant no more hassle of leads or video tapes.

Today, top companies like Hikvision use network video recording (NVR) systems. These systems work by encoding and processing video in cameras and then streaming the footage to NVRs for storage or remote viewing [3]. NVR systems can now include internet protocol cameras (IP). IP cameras can send and receive information over a network and internet connection. The problem with such systems is that the equipment can be very expensive to purchase and maintain.

However, IP cameras have now been found useful with other technologies such as web and mobile phone applications. Many different applications are now available to download which provide live stream from IP cameras over networks.

2.2 Alternative existing solutions to the problem you are solving

In order to develop a unique system, this research will describe multiple systems identifying the characteristics and features which lead to the success of each system. A further analysis will be provided on the systems functional requirements which will then be compared against Nielson’s Heuristics in order to rule out weaknesses which may be found in the interface design. The following systems are high rated developments which have been found throughout the research of this report.

2.2.1 Nielson’s Heuristics

A heuristic evaluation is a usability inspection method for computer software that helps to identify usability problems in the user interface (UI) design [5]. The evaluation consists of 10 principles which are each used to evaluate system features such as usability, efficiency and effectiveness.

1. **Visibility of system status**

   The system should always keep users informed about what is going on, through appropriate feedback within reasonable time [4].

2. **Match between system and the real world**

   The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order [4].

3. **User control and freedom**

   Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo [4].

4 Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions [4].

5 Error prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action [4].

6 Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate [4].

7 Flexibility and efficiency of use

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions [4].

8 Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility [4].

9 Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution [4].

10 Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large [4].

2.2.2 Raspberry Pi Camera Viewer

The application allows user to connect and stream video footage from their raspberry pi device [Figure 2.2]. Raspberry Pi Camera Viewer uses the Gstreamer framework which works on multiple OS's and supports a broad coverage of multimedia technologies. The application offers users to build their own pipeline on their raspberry pi which will still work with the application.

Weaknesses of this application:

Despite the attractive layout and simple UI provided by Raspberry Pi Camera Viewer, it does contain several flaws which users may find bleak. Flaws include:

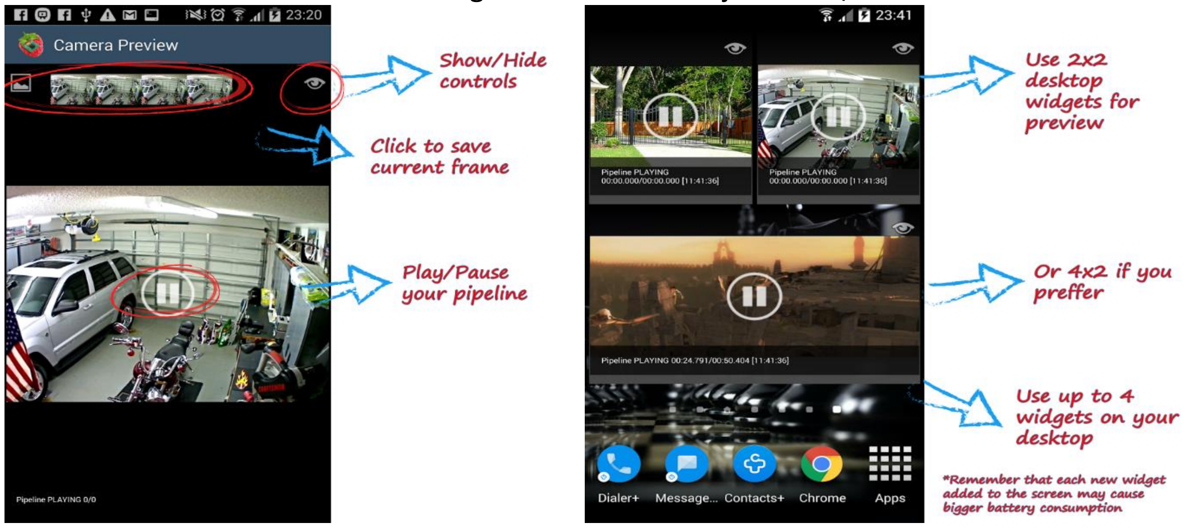
- In order to run the system, the user needs to install the Gstreamer framework on their raspberry pi.
- The user also needs the set the output of the stream on the raspberry pi in order for it to launch
- Full screen mode is unavailable
- Video cannot be recorded
- The application only works over local network
- Does not support image filtering or zoom features

Strengths of this application:

- The application provides users with the option to add multiple cameras
- Users can scroll between multiple cameras connected to the network
- The option to hide controls is available which represents a nice GUI
- Play and Pause is also available
- Users can setup widgets on the home screen for quick access to the cameras

Requirement's taken from this system:

As previously stated, the application provides a unique layout and a very attractive UI. I will definitely consider these features when developing my application. Another feature I may fuse with my application is the hidden controls option. I feel this gives the app a more pleasant look.


Camera Preview
Show/Hide controls
Click to save current frame
Play/Pause your pipeline
Pipeline PLAYING 0/0
Use 2x2 desktop widgets for preview
Or 4x2 if you preffer
Use up to 4 widgets on your desktop
*Remember that each new widget added to the screen may cause bigger battery consumption
Dialer+
Message...
Contacts+
Chrome
Apps

[Figure 2.2 Raspberry Pi Camera Viewer]

2.2.3 p2pCamViewer

P2pCamViewer is a free android application that provides live streaming for IP cameras over LAN and WAN connections [Figure 2.3]. Users can login and save cameras connected a network that is provided. The application includes a network scan feature which displays a list of available IP and ports on the network.

Weaknesses of this application:

Although the application provides a number of quality features, it lacks in other areas such as the layout and design of the application which is rather plain looking. Also, the application provides users with register and login accounts. However, it does not provide users with the ability to edit or delete their accounts. This shows a poor development flaw within the application. The application does not support filtering or home screen access to camera views like provided by Raspberry Pi Camera Viewer.

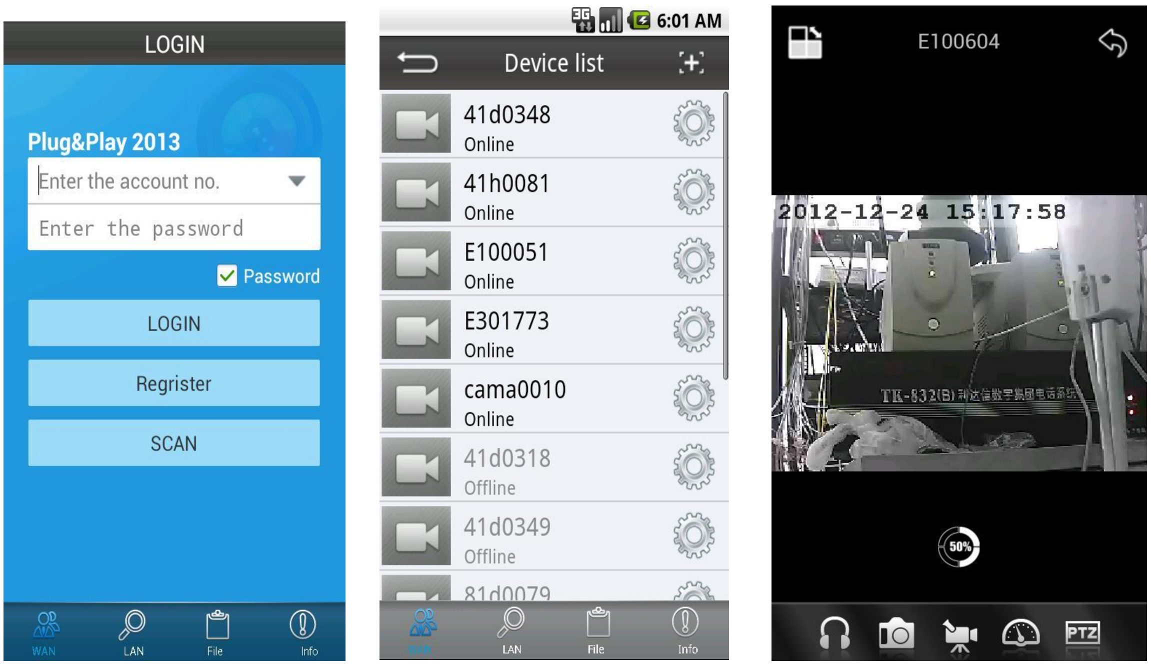
Strengths of this application:

Unlike Raspberry Pi Camera Viewer, the application allows the users to record video on multiple cameras. It includes an abundant of other features which include:

- PTZ options are available on each camera. This offers users with supported cameras the ability to zoom in and out of specified locations within the camera view.
- An effective quality of the application is its capability to access the IP camera over local and wide networks. This offers a great security feature to the application
- P2pCamViewer includes a listen and talk feature which can be very beneficial. For example is users are not home this would be of great use.
- The most effective quality in my opinion is the use of network scanning. Users can scan local networks for available IPs. This rules out complications and offers a more simplistic approach when needing to connect to cameras over a network

Requirement's taken from this system:

As previously stated, although the application does not provide an attractive layout and interface, it does contain a great value of quality features that I will definitely be using. Features like the ability to scan local network networks for camera IPs. Also, after researching a number of different applications, not many applications offer users access over both local and wide networks. I feel this would be a great attraction and would benefit my application greatly.



[Figure 2.3 P2pCamViewer]

2.2.4 Video Surveillance Ivideon

Ivideon is a cloud based video surveillance application available on both android and IOS devices [Figure 2.4]. It provides users with access and reliability without complexity. Users can record and view recordings made through the device. The applications supports the use of IP cameras over networks and provides the highest end of features an application can provide.

Weaknesses of this application:

Ivideon is a well-developed application that provides a near exact system which I wish to provide. However, there are some features that I won't be including in my application. The application provides the ability to record and play back video via Ivideon cloud server. I intend to support users with the use of external device storage like hard drives of memory sticks. This will guarantee a more secure approach to video storage and limit access to the unintended. The application also costs money to download. Yet another turn away feature for the application as many users will pick a free application before a priced application.

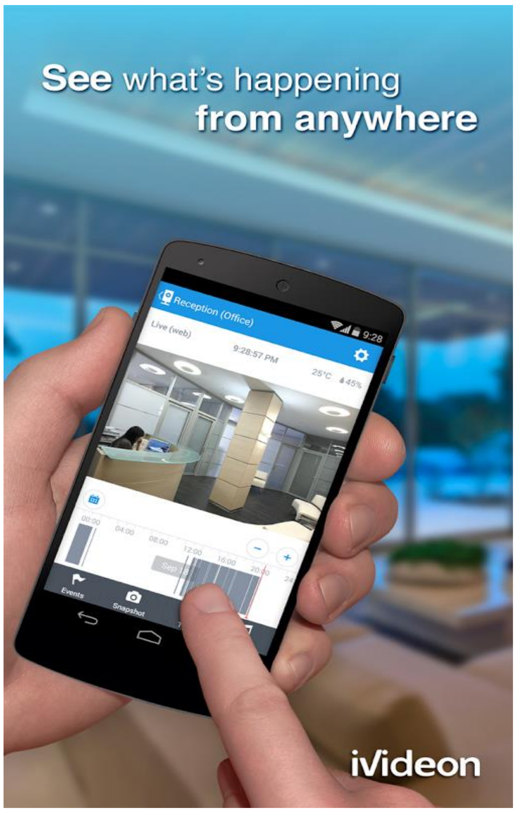
Strengths of this application:

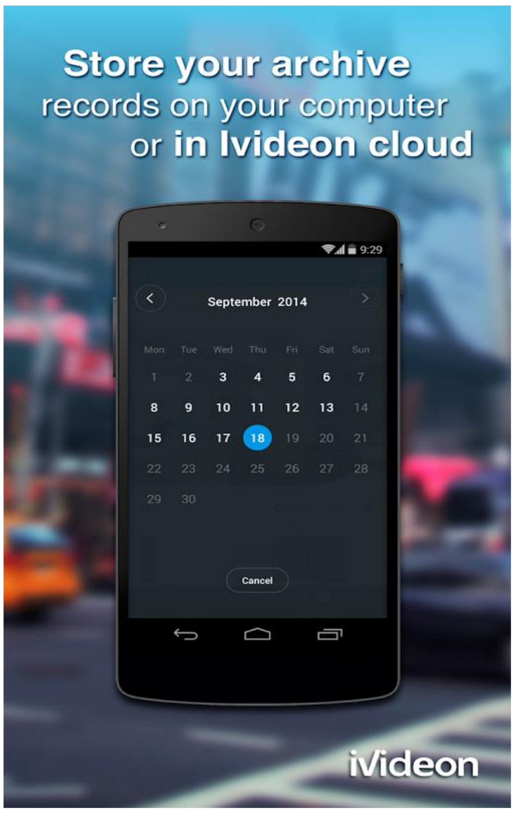
The application is the most similar application I have come across yet. It provides nearly every feature I wish to implement. Some features include:

- Record and store videos over the cloud. I intend to provide video storage through external devices connected to the server device (raspberry pi).
- Search and view recorded video surveillance. Users can playback video stored on the cloud.
- Just like P2pCamViewer, it provides users with the PTZ feature.
- Receive notifications via email of suspicious movements or sounds.
- Share camera links over social networks

Requirement's taken from this system:

The application provides a lot of features which I had already intended to implement with my application. A feature I find will be very beneficial is the use of record and playback features. Being able to instantly playback videos provides a more secure feeling for the users. Also Notifications are another star quality I feel will benefit the application. But instead of notifications via email, I intend to send notifications through the application. An example of this would be a notify sprite screen which will limit out the hassle of having to login to emails to view notifications.





[Figure 2.4 Video Surveillance Ivideon]

2.3 Similar Systems Conclusions

In conclusion to the investigation, its quiet clear that there is no exact application that provides the features I intend to use. However, the systems investigated provide a variety of features which will be used as a starting point to the project. Results using Nielsen's heuristics will now be documented and observed to guarantee a clean and friendly UI.

Visibility of Systems Status	✓
Match Between System and the Real World	x
User Control and Freedom	x
Consistency and Standards	✓
Error Prevention	✓
Recognition Rather Than Recall	✓
Flexibility and Efficiency of use	x
Aesthetic and Minimalist Design	✓
Help Users Recognise, Diagnose and Recover from errors	x
Help and Documentation	x

[Figure 2.5 Raspberry Pi Camera Viewer]

Visibility of Systems Status	x
Match Between System and the Real World	x
User Control and Freedom	x
Consistency and Standards	✓
Error Prevention	x
Recognition Rather Than Recall	✓
Flexibility and Efficiency of use	✓
Aesthetic and Minimalist Design	✓
Help Users Recognise, Diagnose and Recover from errors	x
Help and Documentation	x

[Figure 2.6 p2pCamViewer]

Visibility of Systems Status	✓
Match Between System and the Real World	✓
User Control and Freedom	✓
Consistency and Standards	✓
Error Prevention	x
Recognition Rather Than Recall	✓
Flexibility and Efficiency of use	✓
Aesthetic and Minimalist Design	✓
Help Users Recognise, Diagnose and Recover from errors	x
Help and Documentation	x

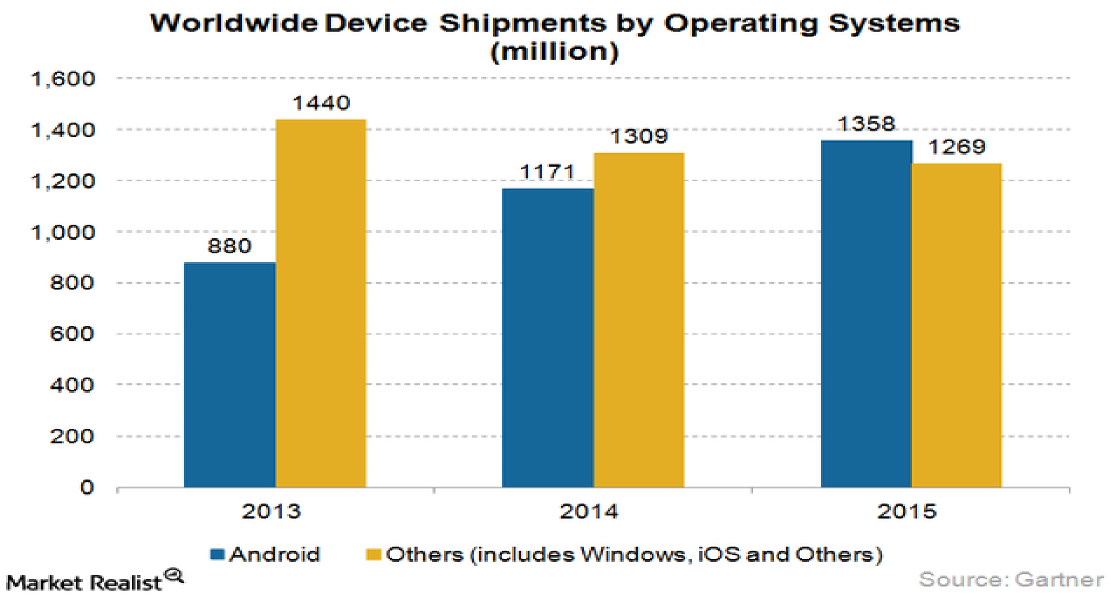
[Figure 2.7 Video Surveillance Lvideon]

2.4 Technologies researched

In order to determine a full effective system, I had to research and evaluate technologies that I felt may be of beneficial use to my system. The following is a list of well documented technologies that I will be using to develop and use with the project.

2.4.1 Android

The main reason I chose to develop my application in android is because not only do I own an android phone and would benefit me for testing purposes, but I also have previous experience developing applications for android. I also very much enjoy using java and feel I have a good understanding of the language. Technically speaking, android is easily ported and is supported on multiple operating systems.



[Figure 2.8 Yearly Shipment Orders Graph][6]

This immediately gives an advantage over window and apple devices. In order to develop for iOS, users would need to use a MAC. Also registration on the Apple App Store is more expensive in comparison to android.

2.4.2 Raspberry Pi 2 Model B

For the purpose of this project I will be using the Raspberry Pi 2 as the server. Originally I intended to use an Arduino model because of previous experience but enjoyed the idea of challenging a new technology. The Pi also provides a wider spec range; HDMI output for testing, USB ports for external storage (media files) and Ethernet port for networking, I knew the Raspberry Pi would be suited best for the project.

The Raspberry Pi provides support for multiple Oss which include Raspbian, Ubuntu Mate and Fedora to name a few. Subsequently I found that the most popular OS used is Raspbian.

	Arduino Uno	Raspberry Pi 2 Model B
Price	30	35
Size	7.6 x 1.9 x 6.4 cm	8.6 x 5.4 x 1.7cm
Memory	0.002MB	512MB
Clock Speed	16Mhz	700Mhz
On Board Network	None	10/100 wired Ethernet RJ45
Multitasking	No	Yes
Input Voltage	7 to 12 V	5 V
Flash	32KB	SD Card
USB	One, input only	4 peripherals
Operating System	None	Linux Distributions
Integrated Development Environment	Arduino	Linux support, IDLE, Scratch
HDMI	No	Yes
DSI	0	1

[Figure 2.9 Arduino Vs Raspberry Pi 2 B Spec]

2.4.3 Java For Android

For the purpose of this project I will be using java to develop the android application. Mainly because I have previous experience developing with java. I also plan on using Androids Network Service Discovery (NSD) which will allow users to search and select devices over local networks. Note that for the purpose of this project users will not have access to features provided by the raspberry pi server. Therefore the application will be used mainly as a view for user IP cams or self-developed similar systems.

2.4.4 Android Studio (IDE)

For the purpose of this project I have decided to use android studio. From previous experience, eclipse is unreliable and experiences many functional issues. Android studio was designed specifically for android development and I again enjoyed the idea of working with a new IDE.

2.4.5 Motion (Daemon)

Motion is an open source program developed in C which monitors the video signal from one or more cameras. It can detect if a significant part of a picture has changed (ie. motion). Motion is specifically designed for the Linux OS so I felt it would be of great benefit to my project. It can also be used to feed events to a MySQL database which I feel could be of great benefit to me. Motion also supports features like such as zoom and media storage capabilities.

2.4.6 Postfix (Daemon)

For the purpose of this project I plan on using an open source mail transfer agent called postfix. Developed in C, Postfix is used to route mail to other mail servers like Gmail (SMTP). I also understand there can be errors running two daemons at once on the Pi so I intend to create a script to check against the PID of each daemon to be sure both are running together.

2.4.7 Heirloom Mailx

For the purpose of this project I plan on using Heirloom mailx which is a mail user agent for UNIX systems. This will be the local mail client for my project that works with postfix to send mail to my Gmail account. A typical example of using Heirloom along with postfix might be: **echo 'hi' | mail –s Test <Email address>**

2.4.8 000webhost (Webhosting + MySQL) & No-IP (dynamic domain Service)

In order to stream and provide media playback, I will be developing a web application using HTML and PHP through the webhost service 000webhost. The web application will accept an IP and port number input by the user, and return a live stream of their IP cam/server. I will also be using this website to create a MySQL database to store user credentials. I found webhost to be a favourable option due to previous experience using the HTML, PHP and SQL languages. It is also a free service which is a bonus. For the purpose of this project I will also develop a media streaming server connected to my Raspberry Pi to view through the web application. I have had some trouble setting up a live stream from outside the local network. Therefore I will create a dynamic domain address using No-IP to eliminate issues. Note that users will not be able to use this feature as it is strictly for the use of this project.

2.4.9 PuTTY

Putty is a free implementation of SSH and Telnet for Windows and UNIX platforms [8]. I will use it to SSH into the server and develop the configuration and relevant files needed to complete the project.

2.4.10 XBMC (FTP)

XBMC (Kodi) is a free and open source media player developed by the XBMC foundation [10]. Available on multiple operating systems, XBMC provides users with a play and view feature of media saved locally. Media can also be viewed over the internet.

2.5 Other technologies

IPCamLive is a cloud based video broadcasting solution for IP cameras which embeds live video streaming to any webpage [9]. It's an alternative for IP cameras which do not have a video player component. The video player provided by IPCamLive is based off of flash and HTML5 and can be used on multiple platforms which include PCs, Mobiles, tablets and MACs to name a few. It uses HTTP aligned with RTSP to stream the video which can display MJPEG, MPEG4 and H.264 streams.

Once a user registers and signs in, he/she can add a camera given the cameras URL as reference. Port forwarding also has to be set up in order to view video streams. IPCamLive provides html snippets of the video streaming which gives access to multiple users at any given time.

For the purpose of this project I intend to create a very similar system. Instead users can sign in given their IP and port number. This will give immediate access to the camera in which they wish to view.

2.6 Resultant findings/requirements

2.6.1 Hardware & Software Requirements

- Raspberry Pi Model 2B
- Raspbian Jessie OS
- Pi Camera
- Android Device
- Android Studio
- Laptop (Windows 10 home) using Putty for SSH
- BASH
- MySQL
- HTML
- PHP
- Java

2.6.2 Functional Requirements

No.	Type	Description	User	Priority
Func-1	Android/Web	Users can login	Users	Medium
Func-2	Android/Web	Users are a logged in user	Users	Medium
Func-3	Android/Web	Users can log out	Users	Medium
Func-4	Android/Web	Edit user credentials	Users	Low
Func-5	Android/Web/Server	Delete user credentials	Users	Low
Func-6	Android	Scan local networks	Users	Low
Func-7	Android/Server	Save media content	Users	Medium
Func-8	Web	View saved media	Users	Medium
Func-9	Android/Web	View video stream	Users	High

[Figure 2.10 Functional Requirements]

2.6.3 Non-Functional Requirements

No.	Type	Description	Priority
Func-1	Performance	The android and web application will provide fast reliable stream at suitable frames per second	High
Func-2	Availability	The camera view will be available from anywhere	High
Func-3	Reliability	The system will be constantly refreshing to provide a stream of images	Medium
Func-4	Compatibility	Video streaming will provide multiple resolutions for clearer images	Low

[Figure 2.11 Non-Functional Requirements]

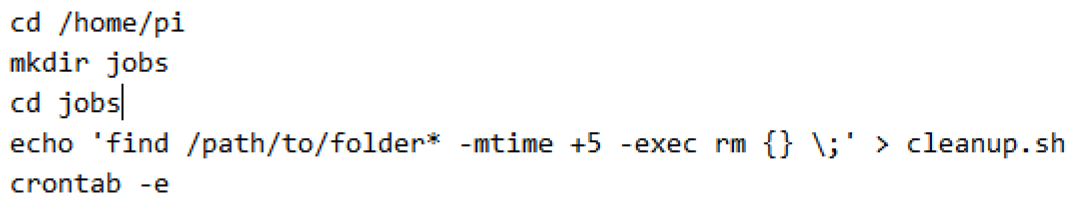
3. Analysis

The system will consist of three components. An Android application, a Web application and a Raspberry pi 2 (Server). Each component consists of key features which play a vital role for the system. I will use this section to discuss the components in further detail.

3.1 Pi Server

The development of the Pi server will consist of bash scripts and configuration files that correspond to the scripts. The server will always require a network connection in order to deliver its purpose. A Pi camera will be connected to the Pi server which will provide video stream for the android and web components. An external device will also be connected to the server to provide media storage for recorded videos/pictures.

The server will function off the operating system Raspbian Jessie. While my router runs off IPv4 settings, I will enable port forwarding so the android device and web application can connect to the server through internet communication from any location. The server will also be given a dynamic domain address provided by No-Ip. This allows us to access the camera from anywhere even if the routers IP changes. Upon boot of the server, a running daemon called motion will be enabled. This provides a number of different features which can be used with the Pi camera (ie. motion detection). Adding the external hard drives UUID (/etc/fstab folder) will automatically mount the device upon boot which gives users the privilege to save media to the device (permission has to be first granted – sudo chown –R pi:pi). Notifications will be enabled using postfix as the SMTP and Heirloom mailx as the client email for the server. When motion is detected, an email will be sent to the users email address notifying him/her (motion.conf). While the email is being sent, a video is also recorded based on the time the motion senses change in the images. Videos are saved directly to the external device and can be viewed through the web application. A simple script will be set up to delete video/images from the external device after a set amount of days. Finally, the server will run an open source media player known as XBMC. XBMC will allow users to stream media form a selected external drive. This is achieved by setting up port forwarding on a different port to the Pi camera and configuring vsftpd on the Pi.



```
cd /home/pi
mkdir jobs
cd jobs
echo 'find /path/to/folder* -mtime +5 -exec rm {} \;' > cleanup.sh
crontab -e
```

[Figure 3.0 Delete Videos Source]

3.2 Android Application

The android application will be developed using android studio. The application will built on the android platform 6.0 Marshmallow targeted at devices with a minimum of SDK 8 and maximum 22. I will use my HTC one device for testing the application.

The application will consist of 3 main pages: Login/Register, Camera View and Settings page.

The login/register page will first prompt the user to login with details they may have already saved (Email/Password). Alternatively the user can register their details. Information required will include: Email, username, password, IP address and port number. - An option will be given to conduct a network scan if users don't know the IP or port number of their camera. Once the user has registered his/her details, they will be redirected to the login/register page.

The Camera View page will provide users with the live video stream from their camera. The camera will be addressed by the IP and port number that they had entered upon registering. The activity will also consist of interactive buttons like the record and settings buttons.

The settings page will offer users access to their saved credentials. Users may wish to edit or delete their account. If users do edit or delete their account, they will be redirected to the login page to in order to update user credentials.

3.2 Web Application

Developed using 000webhost, the web application not only provides live video stream from the user's camera, but allows the user to administer his/her saved media. The application will also consist of three pages: The login/register, Cam/Media view and user settings page.

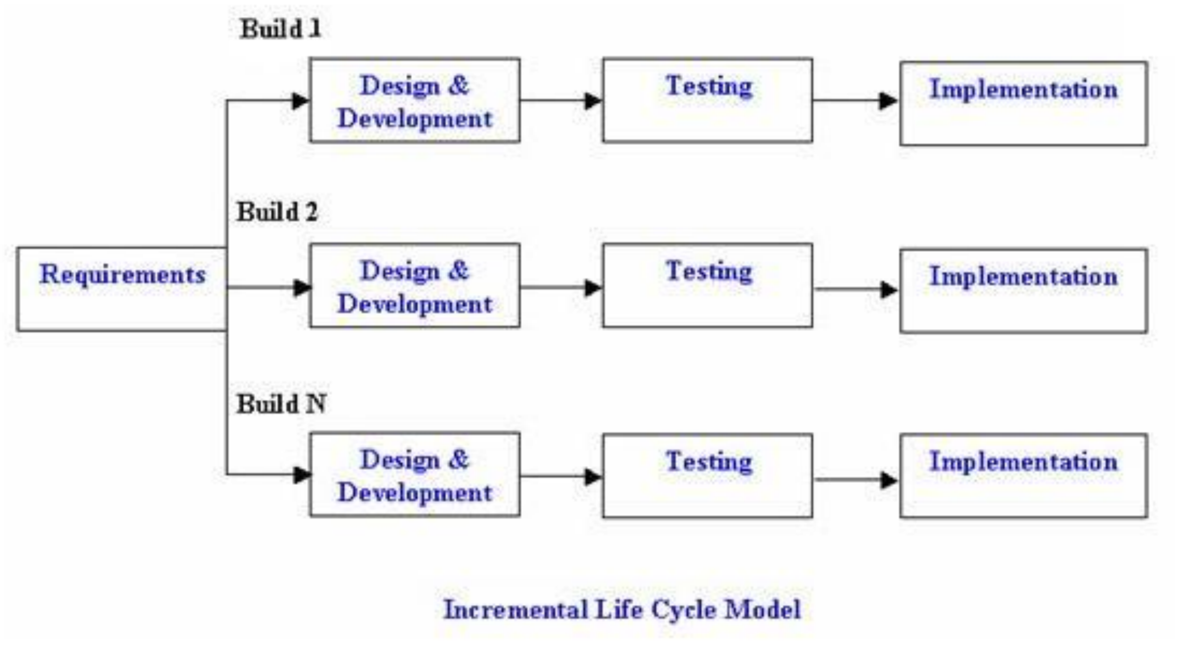
The login and register page will be very similar to the androids page. However, it will not offer users the ability to network scan.

The Cam/Media page will consist of two views. One for the live video stream and the other for the media files view. The Live video stream will provide users with zoom and monitoring features. The Media view will provide media playback of files saved to the external device of the server. XBMCs media viewer will be used to achieve this. A settings button will be implemented below the views.

The user settings page is also very similar to the android settings page. Users can edit and delete credentials from the MySQL database set up by 000webhost.

4. Approach and Methodology

The incremental methodology approach has been considered in order to develop the apposed system. The reason for this choice is the ability to control and not over engineer or add flexibility unless needed. Similar to following the waterfall module, the incremental approach is divided into smaller, more easily managed modules. User requirements are suspended when an increment is under construction. This gives the opportunity for other increments to develop and provide a greater sense of flexibility for the changing requirements. Requirements that contain the highest of priorities are considered and developed first. This gives an advantage over alternative methodologies as it guarantees a quality over quantity effect.



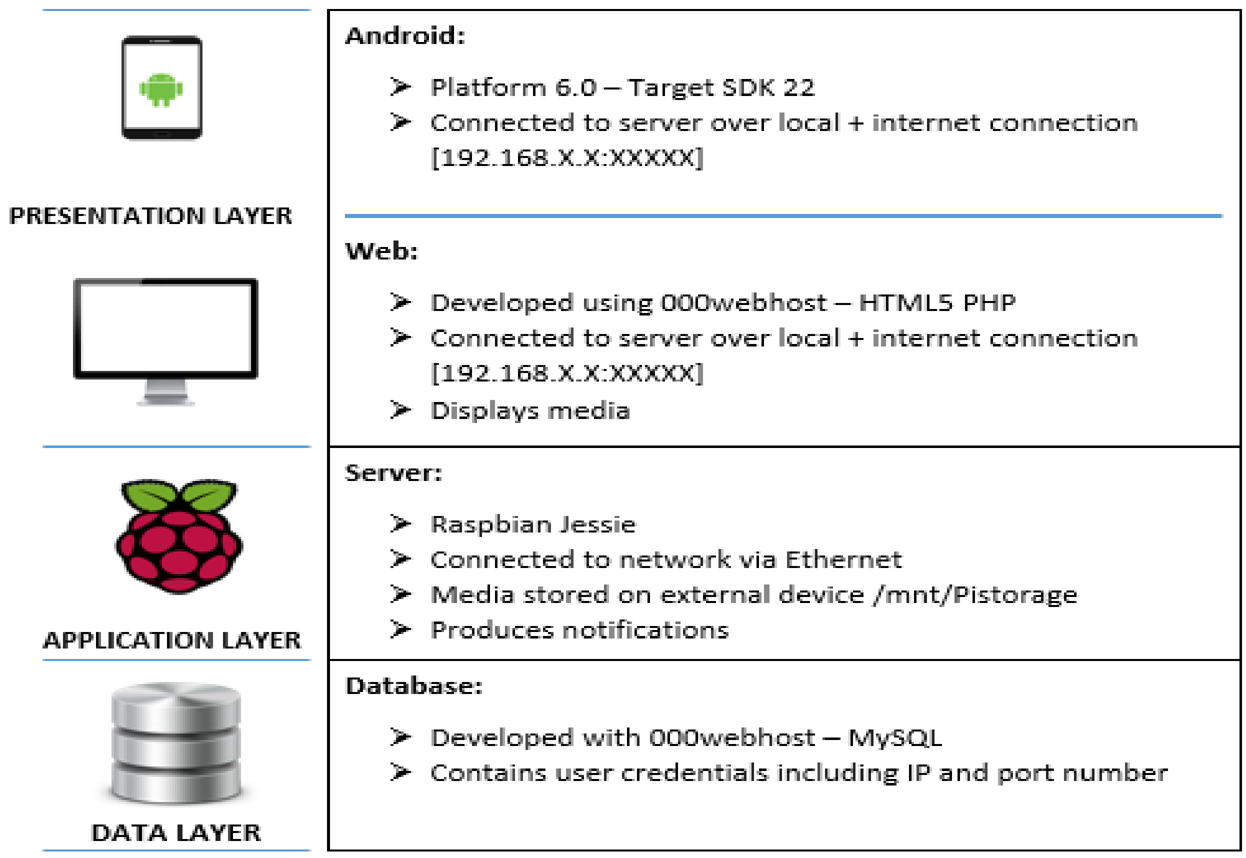
[Figure 4.0 Incremental Approach]

5. Design

The design of the system will be expressed using diagrams and tables in the following section. The diagrams and tables are intended to give a better understanding of the system.

Technical architecture diagram:

The following diagram is based on the main architecture of the system. Designed to give a more simple understanding of the system and display how it's put together. The presentation layer consists of both the android application and the web application. Each application is briefly described how it work and how it's developed. The application layer describes the setup of the server and how it works. Most of the configuration such as motion detection and sent notification is produced here. Finally the data layer briefly describes the database and what is used to develop it.



Android:

- Platform 6.0 – Target SDK 22
- Connected to server over local + internet connection [192.168.X.X:XXXXX]

Web:

- Developed using 000webhost – HTML5 PHP
- Connected to server over local + internet connection [192.168.X.X:XXXXX]
- Displays media

Server:

- Raspbian Jessie
- Connected to network via Ethernet
- Media stored on external device /mnt/Pistorage
- Produces notifications

Database:

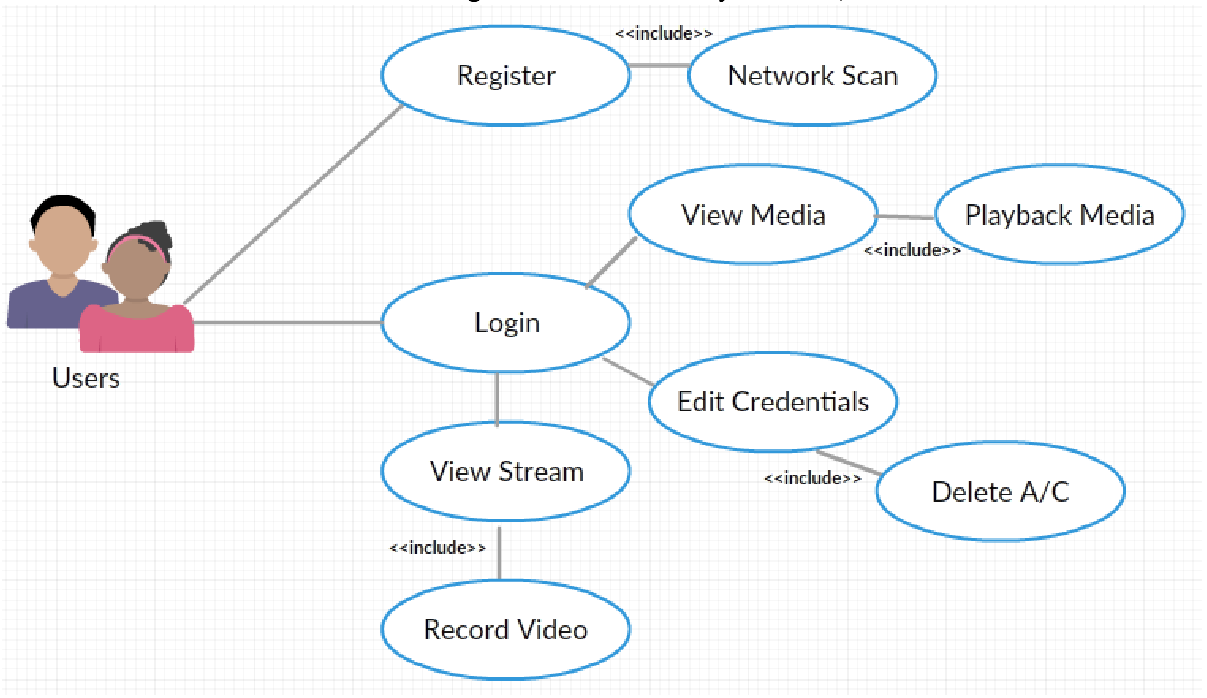
- Developed with 000webhost – MySQL
- Contains user credentials including IP and port number

[Figure 5.0: Technical Architecture]

Other design documents

<u>Use Case Diagram</u>

The following use case diagram exhibits how users interact with the system. Users are first prompted with register or login access. If users attempt to register over android, they may wish to search the network to add a camera. Once logged in, users have a number of different features to interact with. Users may wish to change the cameras address (192.168.x.x: xxxxx) or delete their account. This can be achieved using the settings menu.

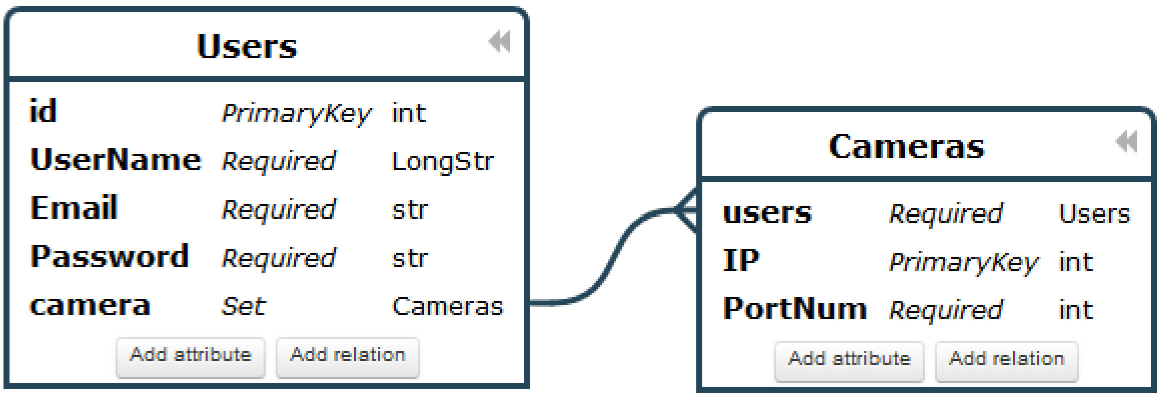


[Figure 5.1: Use Case Diagram]

<u>Entity Relationship Diagram</u>

The entity relationship will represent the design used for the database. The users table consists of the users credentials. Users will be referenced by ID to assign a login/logout system. For every user, multiple cameras can be added to their settings. **NOTE:** Users may save multiple cameras, but can only view one at any given instance.

The cameras table consists of the IP and port number which will be required to view the camera stream.



Users		
id	*PrimaryKey*	int
UserName	*Required*	LongStr
Email	*Required*	str
Password	*Required*	str
camera	*Set*	Cameras

Cameras		
users	*Required*	Users
IP	*PrimaryKey*	int
PortNum	*Required*	int

[Figure 5.2 ERD]

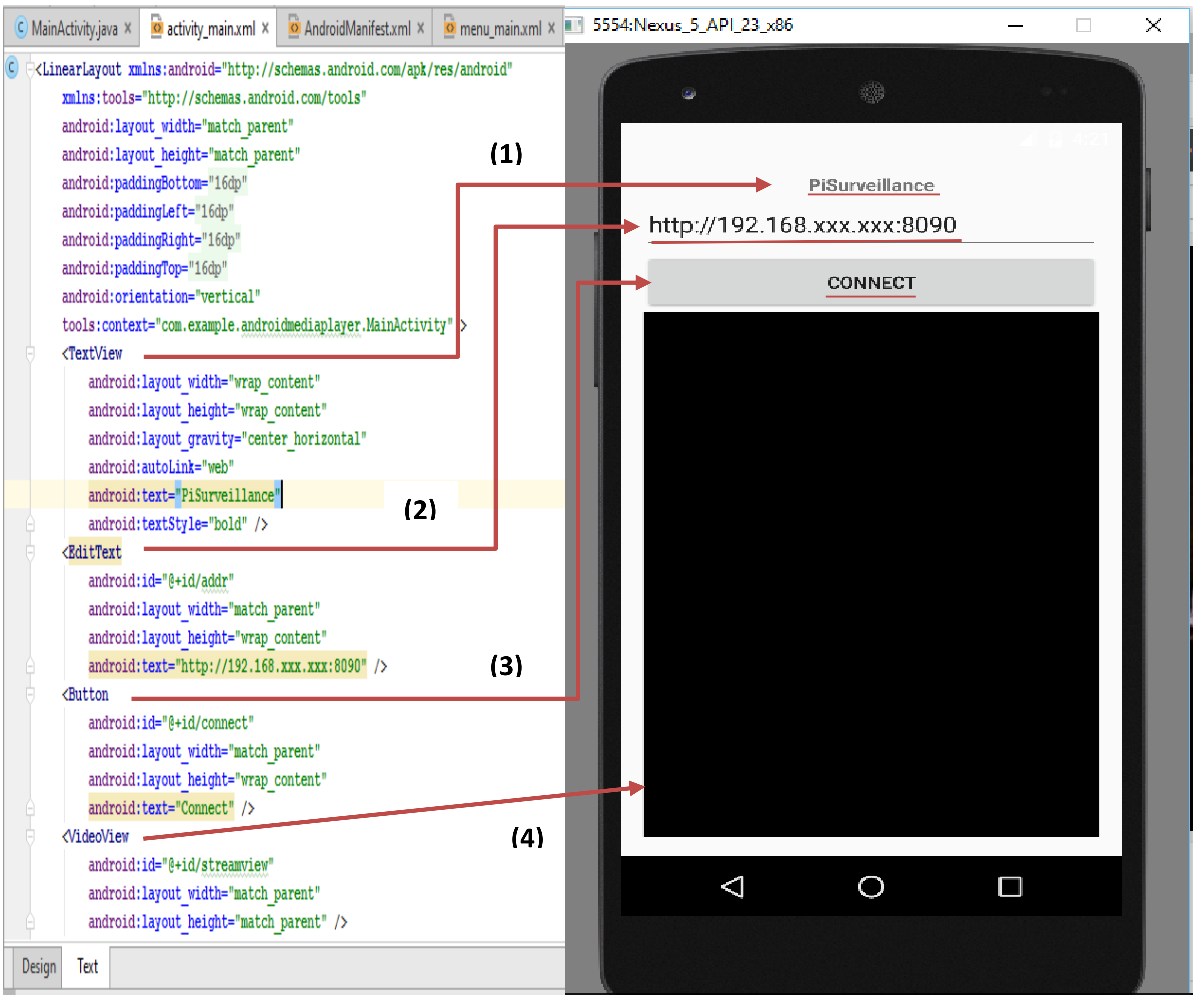
6. Prototyping and Development

The following section explains exactly what I have achieved for the system so far. To begin I will first describe the android application which I have working.

The android application developed is targeted for SDKs of versions 22. So far, I have the main screen developed. It's not fully complete and only shows a camera view. Users can enter there IP and port number as shown in **figure 6.0** (2). The android application uses HTTP to view image files that are refreshed on the server every 2 seconds. Users can view the stream locally and using the internet outside the network.

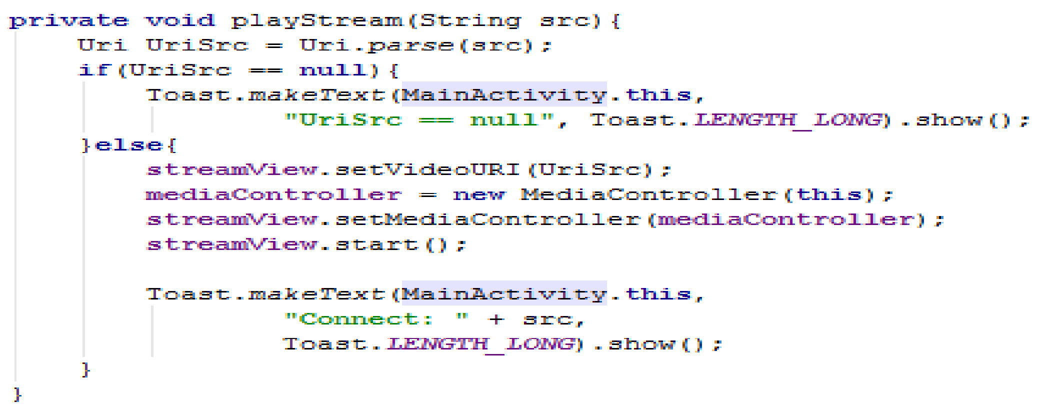
When a user is happy with his/her IP address and port number, they can process the feed from the camera by hitting connect **figure 6.0** (3).

As we can see by the xml file I am using a linear layout. The video views height and width is set to match parent to guarantee a correct positon of display to the user **figure 6.0** (4).Using the import android.net.Uri, the connect button calls the play stream function.



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:orientation="vertical"
    tools:context="com.example.androidmediaplayer.MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:autoLink="web"
        android:text="PiSurveillance"
        android:textStyle="bold" />
    <EditText
        android:id="@+id/addr"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="http://192.168.xxx.xxx:8090" />
    <Button
        android:id="@+id/connect"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Connect" />
    <VideoView
        android:id="@+id/streamview"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
```

[Figure 6.0: Android Application]

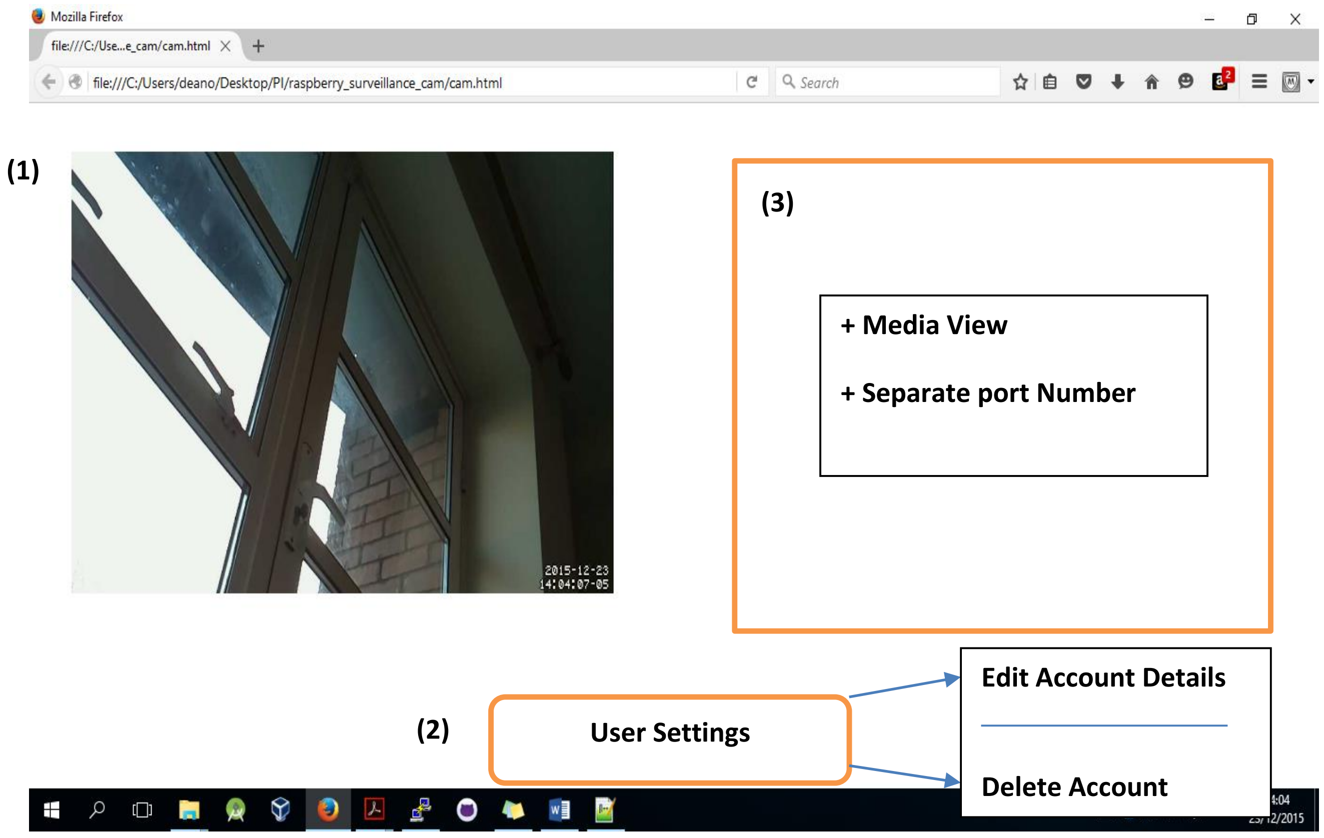


```
private void playStream(String src){
    Uri UriSrc = Uri.parse(src);
    if(UriSrc == null){
        Toast.makeText(MainActivity.this,
                "UriSrc == null", Toast.LENGTH_LONG).show();
    }else{
        streamView.setVideoURI(UriSrc);
        mediaController = new MediaController(this);
        streamView.setMediaController(mediaController);
        streamView.start();

        Toast.makeText(MainActivity.this,
                "Connect: " + src,
                Toast.LENGTH_LONG).show();
    }
}
```

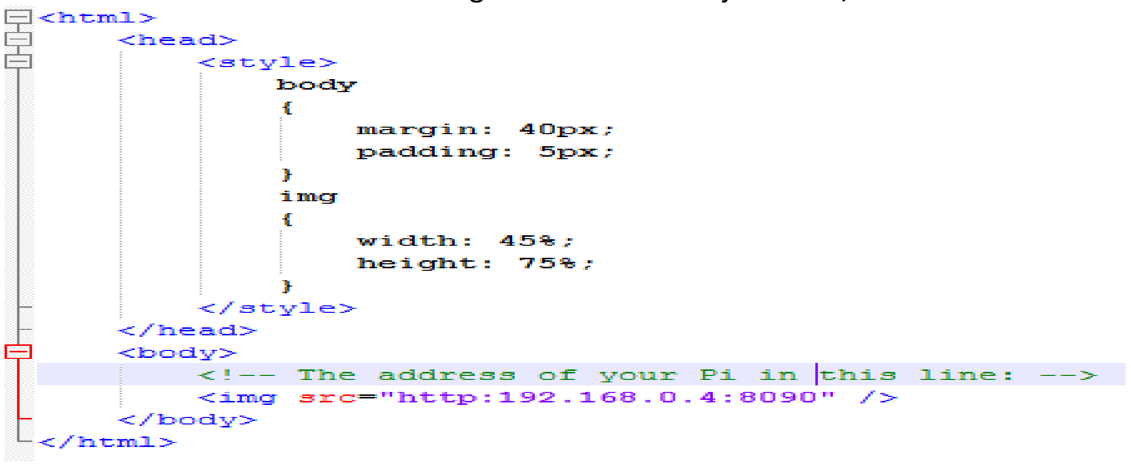
[Figure 6.1: Code Snippet – Connect Button]

Here, the set video URI is using the UriSrc input that the user has entered. If there is no stream found an error message is returned.



[Figure 6.2 Web Application]

The above figure 6.2 describes what progress has been made and what further intentions are in place for the application. (1) The camera view (192.168.0.4:8090) is currently streaming a live feed from the pi server. Motion is used to display the current date and time at the bottom the screen. The page is built on HTML as shown below:

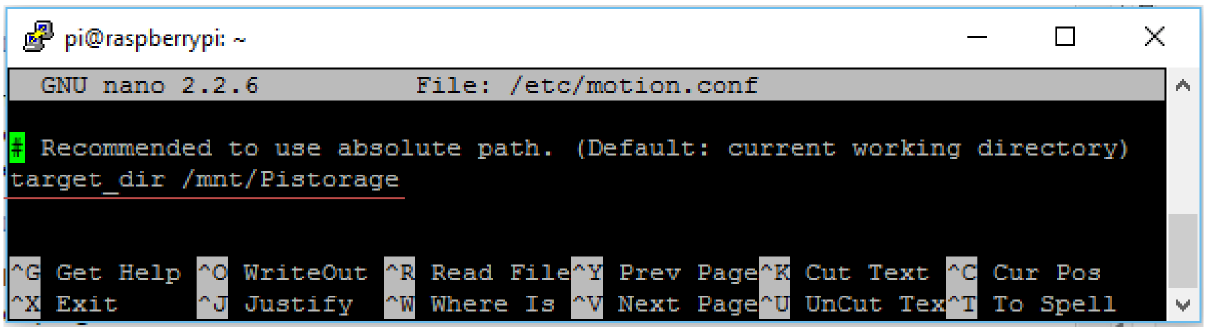


```
<html>
    <head>
        <style>
            body
            {
                margin: 40px;
                padding: 5px;
            }
            img
            {
                width: 45%;
                height: 75%;
            }
        </style>
    </head>
    <body>
        <!-- The address of your Pi in this line: -->
        <img src="http:192.168.0.4:8090" />
    </body>
</html>
```

(2) The user settings button will be provided for users after login. It offers users the ability to edit credentials such as their name or port number of the stream etc. It’s also offers users the option to delete their account if they no longer wish to use the applications. This implies that user permission will also be set up with the MySQL database.

(3) The media view will be provided over FTP using XBMC as the platform for the view. XBMC will be set up using separate port numbers to the Pi cam. Files that are saved to the external device will be available to view remotely.

In the next figure we can see the set value path of the saved media storage. When motion is detected, it saves the pictuers/.avi files to this location judged on the configuration of the framerates per second etc.



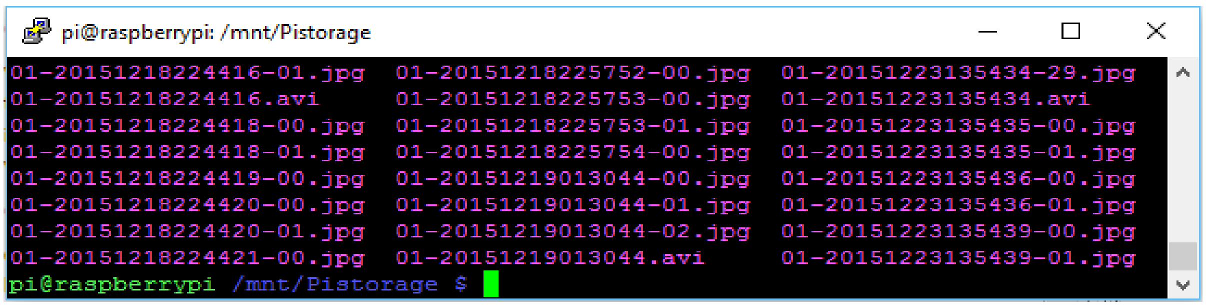
```
pi@raspberrypi: ~
  GNU nano 2.2.6            File: /etc/motion.conf

# Recommended to use absolute path. (Default: current working directory)
target_dir /mnt/Pistorage

^G Get Help  ^O WriteOut  ^R Read File^Y Prev Page^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is ^V Next Page^U UnCut Tex^T To Spell
```

[Figure 6.3 Motion Configuration File]

Images are saved for as long as motion is detected. Once the motion has finished, the images are combined to develop an .avi file. This is achieved using motion.



```
pi@raspberrypi: /mnt/Pistorage
01-20151218224416-01.jpg  01-20151218225752-00.jpg  01-20151223135434-29.jpg
01-20151218224416.avi     01-20151218225753-00.jpg  01-20151223135434.avi
01-20151218224418-00.jpg  01-20151218225753-01.jpg  01-20151223135435-00.jpg
01-20151218224418-01.jpg  01-20151218225754-00.jpg  01-20151223135435-01.jpg
01-20151218224419-00.jpg  01-20151219013044-00.jpg  01-20151223135436-00.jpg
01-20151218224420-00.jpg  01-20151219013044-01.jpg  01-20151223135436-01.jpg
01-20151218224420-01.jpg  01-20151219013044-02.jpg  01-20151223135439-00.jpg
01-20151218224421-00.jpg  01-20151219013044.avi     01-20151223135439-01.jpg
pi@raspberrypi /mnt/Pistorage $
```

[Figure 6.4 Motion Detected Files]

7. Testing

The following section describes the approach taken to test the functionality of the 3 system components: Android Application, Web Application and Pi Server.

Below are a number of test cases used to describe preconditions, expected results and post conditions that the system should produce.

ID: 1 Android	ID: 2 Android
Name: Register	**Name:** Login
Purpose: Register new user containing the required information and address that views his/her cam	**Purpose:** Login with user credentials that have already been registered to the systems database
Procedure steps: ➢ Open android application ➢ Hit register button ➢ Enter details	**Procedure steps:** ➢ Open android application ➢ Hit login button (Username + Password)
Expected Results: The registered user should be able to login with his/her details and view their camera provided it was the right IP and port number.	**Expected Results:** After logging in with the users saved credentials, he/she should be able to view the live camera feed from his/her camera

ID: 3 Android	ID: 4 Android
Name: Record	**Name:** Settings
Purpose: Users can record content to view at a later stage	**Purpose:** The settings feature allows users to edit and remove accounts.
Procedure steps: ➢ Open android application ➢ Hit login ➢ Hold record button for desired time	**Procedure steps:** ➢ Open Android application ➢ Login ➢ Hit settings button
Expected Results: Users should be able to review the recorded content on the web application	**Expected Results:** The settings button opens a view which contains the edit and delete buttons. Users can also add new cameras using the add button.

ID: 5 Android	ID: 6 Android
Name: Edit	**Name:** Delete
Purpose: The edit button displayed on the settings menu gives user access to edit user credentials	**Purpose:** The delete button displayed on the settings menu gives users the ability to delete their accounts
Procedure steps: ➢ Open android application ➢ Login ➢ Open settings ➢ Hit edit	**Procedure steps:** ➢ Open android application ➢ Login ➢ Open settings ➢ Hit delete
Expected Results: Once users configure their new credentials, they are redirected to the login page so that the new credentials can be refreshed	**Expected Results:** Once users hit delete, they are prompted with a confirmation message. If 'yes' is chosen, the users account is deleted and they are redirected to the login page.

ID: 7 Web	ID: 8 Web
Name: Register	**Name:** Login
Purpose: Register new user containing the required information and address that views his/her cam	**Purpose:** Login with user credentials that have already been registered to the systems database
Procedure steps: ➢ Navigate to web application ➢ Hit register button ➢ Enter details	**Procedure steps:** ➢ Navigate to web application ➢ Hit login button (Username + Password)
Expected Results: The registered user should be able to login with his/her details and view their camera provided it was the right IP and port number.	**Expected Results:** After logging in with the users saved credentials, he/she should be able to view the live camera feed from his/her camera. Media content will also be accessible.

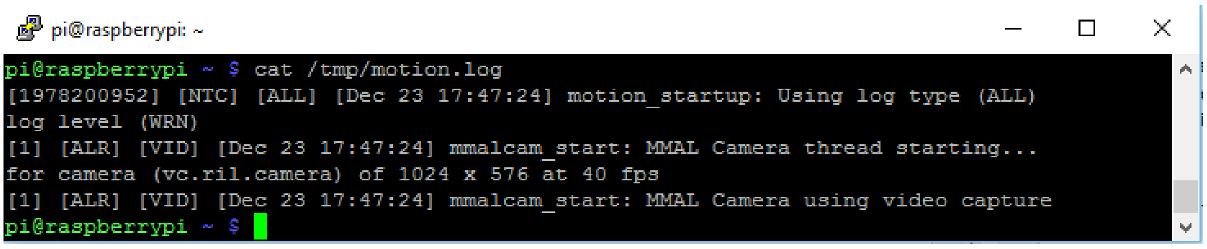
ID: 9 Web	ID: 10 Web
Name: Zoom	**Name:** Settings
Purpose: Users can zoom in and out of points located on the camera view	**Purpose:** The settings feature allows users to edit and remove accounts.
Procedure steps: ➢ Navigate to web application ➢ Login ➢ Click camera view to zoom in and out	**Procedure steps:** ➢ Navigate to web application ➢ Login ➢ Hit settings button
Expected Results: Users should be able to review the recorded content on the web application	**Expected Results:** The settings button opens a view which contains the edit and delete buttons. Users can also add new cameras using the add button.

ID: 11 Web	ID: 12 Web
Name: Edit	**Name:** Delete
Purpose: The edit button displayed on the settings menu gives user access to edit user credentials	**Purpose:** The delete button displayed on the settings menu gives users the ability to delete their accounts
Procedure steps: ➢ Navigate to web application ➢ Login ➢ Open settings ➢ Hit edit	**Procedure steps:** ➢ Navigate to web application ➢ Login ➢ Open settings ➢ Hit delete
Expected Results: Once users configure their new credentials, they are redirected to the login page so that the new credentials can be refreshed	**Expected Results:** Once users hit delete, they are prompted with a confirmation message. If 'yes' is chosen, the users account is deleted and they are redirected to the login page.

ID: 13 Web	ID: 14 Server
Name: Media playback	**Name:** Motion
Purpose: Users can view media that they have previously recorded	**Purpose:** When the camera senses a change in an image, it begins to record a video until the change has normalized. (ie. motion)
Procedure steps: ➢ Navigate to web application ➢ Login ➢ Interact with media view (scroll)	**Procedure steps:** ➢ Power on server ➢ Run motion (daemon) ➢ Activate Pi camera ➢ Refresh stream frame
Expected Results: Once logged in, if users have previous saved videos, they can view it using the media view on the main page.	**Expected Results:** Videos should save to the specified file path given in the motion configuration file. This provides users with media playback.

ID: 15 Server
Name: Notifications
Purpose: Provide notifications to the users email to notify him/her when motion is detected
Procedure steps: ➢ Power on server ➢ Run motion (daemon) ➢ Activate Pi camera ➢ If motion detected: call function to send email using postfix daemon
Expected Results: Users should receive email notifying him/her about motion detected

For the purpose of reviewing running errors using motion, I have created a **log file** motion.log. This has allowed me to fix some of the problems which have occurred using motion (explained section 8).



```
pi@raspberrypi ~ $ cat /tmp/motion.log
[1978200952] [NTC] [ALL] [Dec 23 17:47:24] motion_startup: Using log type (ALL)
log level (WRN)
[1] [ALR] [VID] [Dec 23 17:47:24] mmalcam_start: MMAL Camera thread starting...
for camera (vc.ril.camera) of 1024 x 576 at 40 fps
[1] [ALR] [VID] [Dec 23 17:47:24] mmalcam_start: MMAL Camera using video capture
pi@raspberrypi ~ $
```

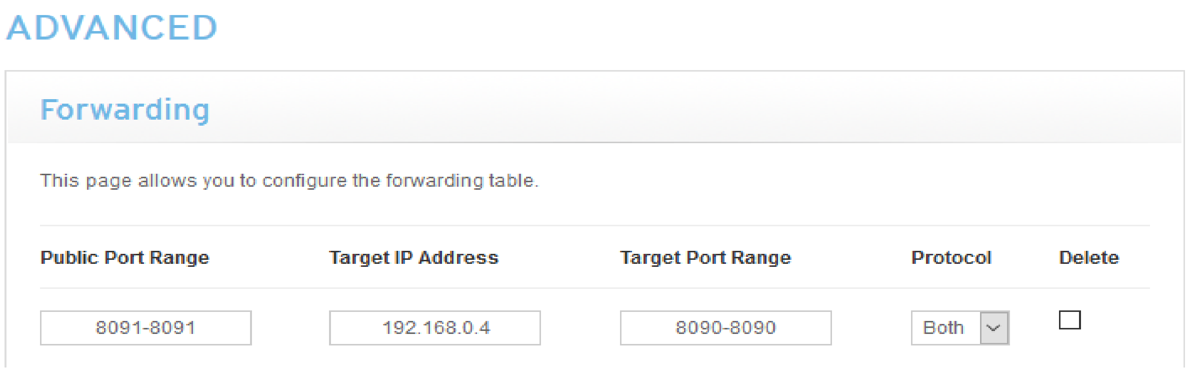
[Figure 7.0 Motion log no errors]

8. Issues and risks

The following section gives a brief description of the problems faced so far while developing the system.

IPv6 to IPv4

Due to the strict permissions of IPv6, in order to access the server and stream the media a call had to be made to the routers provider and request a switch back to IPv4. This granted admin access to the routers settings which meant one could enable port forwarding on the servers IP.



ADVANCED

Forwarding

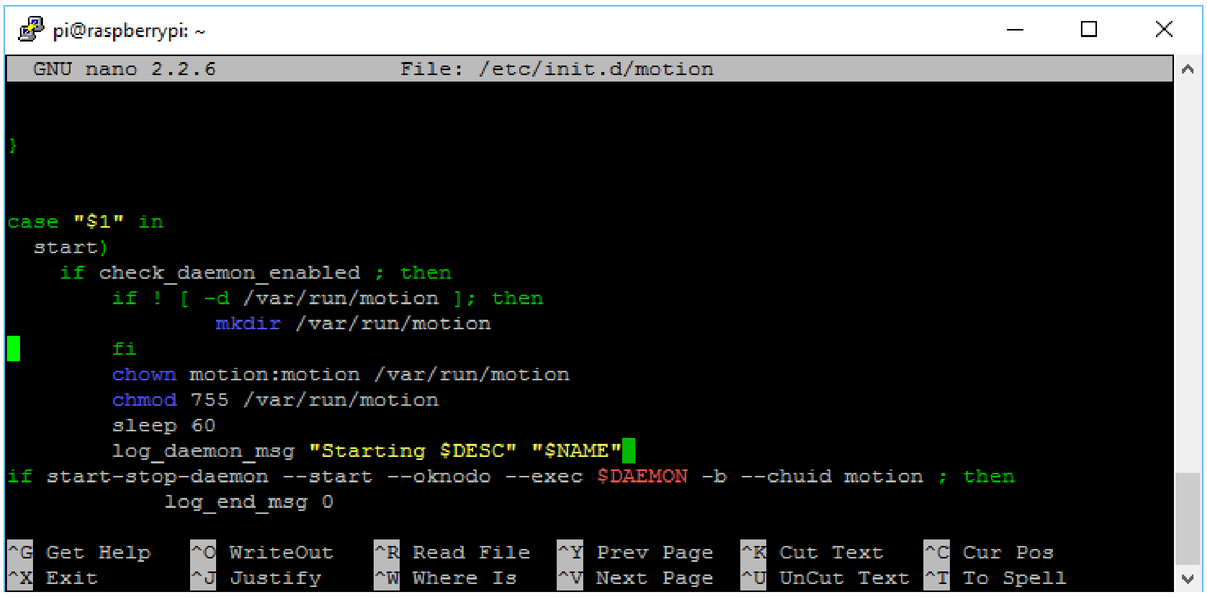
This page allows you to configure the forwarding table.

Public Port Range	Target IP Address	Target Port Range	Protocol	Delete
8091-8091	192.168.0.4	8090-8090	Both	☐

[Figure 8.0 Port Forwarding Enabled]

External Device

The external device that was connected to the server had had to be automatically mounted upon the system boot. Unfortunately problems were faced trying to configure motion to work with the device. Motion wouldn’t start because it could not recognise that the external device was mounted upon the system boot. Thankfully, a solution found online suggested to add a sleep function to the daemon file /etc/init.d/motion. This meant that the raspberry pi could start and mount the external device in time for motion to begin and recognise the mounted device.



```
pi@raspberrypi: ~
  GNU nano 2.2.6              File: /etc/init.d/motion


}


case "$1" in
  start)
    if check_daemon_enabled ; then
        if ! [ -d /var/run/motion ]; then
                mkdir /var/run/motion
        fi
        chown motion:motion /var/run/motion
        chmod 755 /var/run/motion
        sleep 60
        log_daemon_msg "Starting $DESC" "$NAME"
if start-stop-daemon --start --oknodo --exec $DAEMON -b --chuid motion ; then
            log_end_msg 0

^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Dynamic Domain Address No-IP

Upon developing the web and android application, issues were experienced with connecting to the server outside the network. A solution suggested that if one had set up a dynamic domain address, it would enable me to access the server from anywhere. Enabling the address meant that the IP set to the server would remain the same and would not need changing.

9. Plan and future work

Wifi Enabled

A simple yet effective solution that should be incorporate into the system design is the ability to place the camera anywhere around the house provided it can reach the routers connection. WIFI dongles can be bought from sites like amazon and eBay and are not expensive.

PTZ

Pan Tilt Zoom is a major feature to have involved with the apposed system. This would provide motion tracking which would be very beneficial for security purposes. Users could view and track people within distance of the cameras view. 2 servo motors have already been purchased and all that is required is a circuit board to control the power provided to the GPIO on the server.

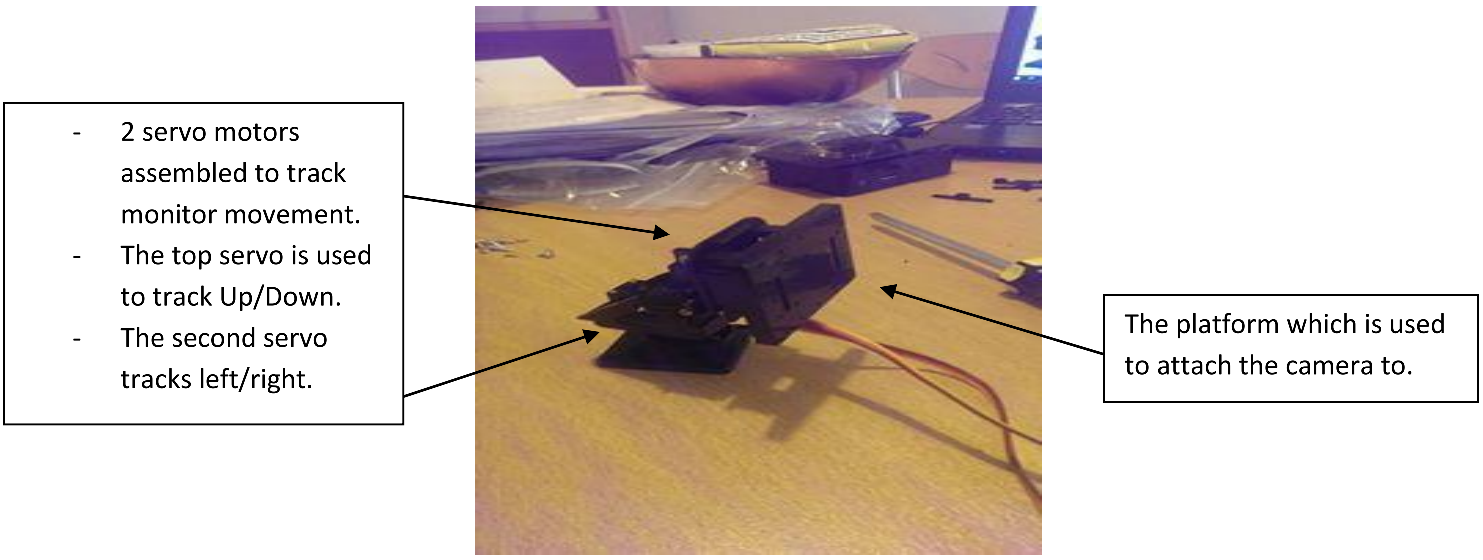

- 2 servo motors assembled to track monitor movement.
- The top servo is used to track Up/Down.
- The second servo tracks left/right.
The platform which is used to attach the camera to.

Image Processing

Finally, for the final feature which would also greatly benefit the security purposes of the system is face recognition. Image processing has been researched and undertaken as part of the college course in order to try and develop this feature into the opposed system. Open source libraries like OpenCV can be used to implement this feature.

10. Conclusions

The research undertaken throughout the report suggests great potential for the opposed system provided that the requirements displayed in section 2.6.2 are met. The main issues at hand is time management and needs to be dealt with accordingly. There is still a wide amount of research needed in order to complete the project which includes image processing as the main feature. Note that there is still also pre-existing areas of the system the need attending such as the development of the media server and the possibility of the user interacting with the server. I have full intentions to include as many features into the systems as possibly. My main goal is to acquire enough information needed to apply face recognition. To conclude, having seen how much I have progressed and completed with regards to the project and the features I have already functioning, I am confident that I am capable of delivering the final project that is opposed.

Bibliography

[1] 10 Heuristics for User Interface Design: Article by Jakob Nielsen. 2015. *10 Heuristics for User Interface Design: Article by Jakob Nielsen*. [ONLINE] Available at: http://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 23 December 2015].

[2] The Best Security Inventions In History. 2015. *The Best Security Inventions In History*. [ONLINE] Available at: http://www.slideshare.net/ajsteeldoors/best-security-inventionshistory. [Accessed 23 December 2015].

[3] *The history of CCTV - from 1942 to present | The latest news from the Computer and IT Industry | PC Retail*. [ONLINE] Available at: http://www.pcr-online.biz/news/read/the-history-of-cctv-from-1942-to-present/034658. [Accessed 23 December 2015].

[4] 10 Heuristics for User Interface Design: Article by Jakob Nielsen. 2015. *10 Heuristics for User Interface Design: Article by Jakob Nielsen*. [ONLINE] Available at: http://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 23 December 2015].

[5] 10 Heuristic Principles – Jakob Nielsen's ~ UXness . 2015. *10 Heuristic Principles – Jakob Nielsen's ~ UXness* . [ONLINE] Available at: http://www.uxness.in/2015/02/10-heuristic-principles-jakob-nielsens.html. [Accessed 23 December 2015].

[6] *Android could represent over 50% of worldwide devices by 2015 - Market Realist*. [ONLINE] Available at: http://marketrealist.com/2014/04/android-represent-50-worldwide-devices-2015/. [Accessed 23 December 2015].

[7] *WebHome < motion < Foswiki* Available at: http://www.lavrsen.dk/foswiki/bin/view/Motion (Accessed: 23 December 2015).

[8] PuTTY: a free SSH and Telnet client. 2015. *PuTTY: a free SSH and Telnet client*. [ONLINE] Available at: http://www.chiark.greenend.org.uk/~sgtatham/putty/. [Accessed 23 December 2015].

[9] *Embedding IP camera live video stream in web page* (no date) Available at: https://ipcamlive.com/ (Accessed: 23 December 2015).

[10] *Open source home theatre software* (2015) Available at: http://kodi.tv/ (Accessed: 23 December 2015).