Contents

Technical Manual – SecuriPi Remote Surveillance

1 - Introduction:

1.1 - Overview

SecuriPi is a cost-effective home surveillance system, which utilizes only a Raspberry Pi, and an android application.

The Raspberry Pi contains python motion detection and image capturing code that can "arm" an attached web camera. This means that any movement the camera detects will cause it to begin capturing images rapidly, and uploading them to a user's Google Drive account. As well as this, once the device has detected motion, several other activities occur.

The device sends a push notification POST request to Google's Firebase Messaging web-app, which in turn, along with a unique Identification key of the user, which in turn sends a push notification to the SecuriPi Android Application on the user's mobile device. An SMS message is also sent to a specified phone number, in case the user does not have internet connection, to notify them that their camera has picked up movement.

As well as this, an alarm is triggered upon motion detection, and will only stop once the motion stops (or it is turned off by the application).

The previously mentioned application can be used as a "remote control" for the Raspberry Pi, and can control various aspects of the system such as setting the sound of the alarm. The app is also capable of accessing files uploaded to the drive and viewing them "in-app", turning off the alarm once it has been triggered, and even arming and disarming the system itself. The application achieves this by establishing an SSH connection with the Raspberry Pi, and executing various terminal commands to either find and replace strings in the python code, or to run/kill programs, depending on the task.

SecuriPi is the perfect solution for anybody who requires the full toolkit of a typical surveillance system with very minimal equipment, and without the typical financial commitment. The android application is a modern and convenient way for people to use a device that so many people are already so connected to, in order to monitor their home from anywhere on the planet.

1.2 - Glossary:

SSH - *"The Secure Shell Protocol (SSH) is a protocol for secure remote login and other secure network services over an insecure network."* - RFC4252

MIME Type - *"Multipurpose Internet Mail Extensions (MIME) is an Internet standard that extends the format of email to support text in character sets other than ASCII, non-text attachments, in non-ASCII character sets"* – Wikipedia

GPIO – *"General-purpose input/output (GPIO) is a generic pin on an integrated circuit or computer board whose behavior,including whether it is an input or output pin, is controllable by the user at run time."* - Wikipedia

RESTful Architecture – "*Representational State Transfer (REST) is a style of architecture based on a set of principles that describe how networked resources are defined and addressed."* – Service-architecture.com

2 - System Architecture

2.1 - Development Environment

The Application

The SecuriPi android application was developed in Android Studio version 2.2.3. The application itself consists of seven Java Classes, and various XML files for the UI. The team used a WAMP server which provided PHPMyAdmin, a SQL Database, and an Apache web server.

The application targets Android level 21 specifically, which is android Lollipop 5.0 and higher. The reason for this is two-fold. Firstly, Android Lollipop was the first distribution which allows users to access notifications from their lock screen. The development team believes this to be an important feature in terms of ease of use, as when the user is notified of camera detection, they can swipe the notification away, view the notification, or even open the app all from the lock screen.

The option to select priority applications in android 5.0 was another large factor in the team's choice. This feature allows users to be in "Do not disturb" mode (meaning they will not receive any notifications), but add exceptions for particularly important apps that demand such importance. SecuriPi, being a home security application, would most certainly qualify for such an order of precedence.

There were also many external libraries used in the development of the Android Studio project. Google services such as Play services and Firebase services, as well as the Google Drive Android API, which is the framework on which the authentication and image hosting services of the application are built on. The okhttp3 client was used to handle POST and GET requests to various google web apps. Jsch Network client was used to establish SSH connections with the raspberry Pi. Finally, AndroidJunitRunner and Espresso was used in the testing of the code.

Raspberry Pi

The code on the side of the Raspberry Pi was written using notepad++ and Sublime Text, and transferred to the pi using a mass storage device (USB flash drive). The motion capture functionality, sending of notifications and SMS messages, and uploading of files is run on the pi itself. This functionality is contained within 7 python files, and 1 json file used to handle user settings. The sound files are also stored here.

OpenCV 3.1 Library was used as the means of designing and implementing the motion capture from the webcam. This particular library was used as it is open source, python supported, and came highly recommended on many sites during our research into this project as a means of real time computer vision. Version 3.1 was chosen as it was stable and well documented, and also included better lighting detection for dimly lit spaces.

Python version 2.7 was used on the pi. As a team, we chose this version ahead of some newer versions mainly due to the vast number of online resources that are available to it, for what we needed to accomplish during this project there were no faults with this build. Many external python libraries were implemented in our development of this project, including a simplified version of google drive for our uploading and authentication purposes.

2.2 - System Design

The SecuriPi surveillance system was developed largely with security purposes in mind, for example the detection of an intruder. This is reflected by the various design elements of the system which provide the user with the necessary toolkit to identify such a situation, evaluate it, and take appropriate action based on it.

Application Remote Functionality

To access all features of this system remotely through an application was for the development team, the ideal sought after scenario, and something that has been successfully accomplished. The application (as well as some

clever scripting on the Raspberry Pi), completely removes the need to use a monitor or peripherals to control the device after initial set-up, meaning the system can be activated, de-activated, interacted with, and modified remotely through the app.

The Application acts a remote control for the SecuriPi system using SSH connections. The Start and Stop camera buttons combine a huge amount of functionality into the click of a simple button on the user's end. First, a session is created and passed the parameters of the username, host IP address and port number.

The session's password is then set to that of the Raspberry Pi. This session is connected and an execution channel is opened to gain access to the Pi's terminal. The execution command is then set to a string which will represent the terminal command to be sent to the Pi, which will change depending on the functionality of the button. For example, to start the camera the command will execute the pi_surveillance.py file with the relative Json scripts, whereas to stop it the string is set to "pkill -f python".

This channel is then connected, executed, and disconnected, making for a clean execution of the functionality the user required, with a very complex back end, but as far as the user is concerned, a simple tap of a button is sufficient.

The app also is the medium through which the user receives push notifications for movement detection, as well as allowing the user to choose the alarm to sound on the Raspberry Pi, when this occurs.

Application UI

The user interface of the application was designed with the primary goal of being simple and clean. The application is a tool and as such it's functionality needed to be all together in one place, easy to access, and not cluttered or distracting. As such, the main activity screen contains all the necessary functionality to arm and disarm the camera, call emergency services or contact a neighbor, view the drive to see uploaded images, or access configuration options through the options menu.

The center of the screen is a simple android logo, until the user opens a file from their drive, at which point this changes to the particular file that was opened.

The team wanted to ensure that the 7 principles of Universal design were adhered to, without getting carried away with adding exotic color schemes and flashy animations. These principles are:

1) Equitable use: The application is marketable to a wide range of users with varying abilities. This has been achieved through what is essentially a one-page, control panel layout for simplicity, as well as visual feedback on any change that is made or option that is selected, in the form of pop-up messages in the bottom of the screen.

2) Flexibility in use: This application accommodates a wide range of preferences, such as allowing the user to choose how to use the surveillance functionality. For example, by changing the alarm sound to the "Lullaby" option, the system would work as a monitor for a sleeping baby, while selecting the "Siren" is a perfect tool to use for security purposes in order to prevent burglary. "The Dog bark" option on the other hand may be useful to ward off intruders from outside the home, where the camera is monitoring a garden or outdoor entrance.

3) Intuitive use: Although the technology behind the system is complex, the language used to translate this functionality to the end user has been purposefully made very simple. Buttons and notifications are labelled with the non-tech savvy person in mind, in order to ensure user friendliness. Furthermore, the user is never more than one click away from the main control panel / home screen of the application, meaning the application does not require high concentration levels to navigate.

4) Perceptible information: While the application will display feedback to the user, no unnecessary information is displayed, to ensure the user does not become overwhelmed. Text is clear and concise, and the app avoids

presenting long strings of text which the user may find off-putting. Color has been used in places such as the toggle button, in order to give the user intuitive feedback when notifications are toggle on(green) or off(red).

5) Error handling: It is important that the end user knows when something has gone wrong, and is notified immediately. As such, if there happens to be loss of connection or a problem with Google's servers for example, the user will be notified appropriately. This functionality is carried out using the google drive SDK.

6) Low physical effort: As mentioned previously, the application is navigated through simple one click operations, with the user never being taken far away from their main control panel. This means navigating through the app never feels like a chore or something that is an inconvenience.

7) Spatial locality of functionality: Finally, the team wanted to ensure that the locality of each function in the application was intuitive and appropriately laid out. The Call 999 and Text neighbor options have been kept aligned on the left of the screen, while the remote toggling on and off of the motion detection software has been kept together on the right. Notifications can be toggled on or off through the toggle button in the top right of the main screen, meaning the user will not accidentally change this preference while accessing the rest of the app's functionality.

Motion Detection and notification

Raspberry Pi

A python script running on the pi uses OpenCV in order to detect movement from the connected webcam and output images that have detected movement to the user's Google Drive account, which is automatically authenticated after the first initial set-up.

A frame is initially grabbed from the camera feed and converted to a grayscale image. This is done as colour has no bearing on motion detection. The image is then blurred using Gaussian blurring to smooth the image. The average weight of all captured frames is accumulated, so that changes in the background i.e lighting conditions gradually changing, do not get detected as motion, and is then subtracted from the current frame's weight to give our frame's delta.

This delta is then thresholded to find areas containing substantial difference from the background, and can be corresponded to motion. In order to find regions that exceed the threshold contour detection is applied. The contours are looped over and if they exceed the pre-defined "min_area" then they are identified as motion, a box is drawn around them, and the text on the feed changes to "movement detected".

The images are then uploaded to the user's Google Drive account if the number of frames with consistent detected motion passes the pre-defined "min_motion_frames". Notifications are send to the app and also via SMS if so enabled by the user. The user must set up their phone number on installation and have an account with our 3rd party SMS provider "TextLocal".

Android App

On the application side, a notification is built in the FirebaseMessagingService class and FirebaseMessagingID class. The user will always receive a notification whether the application is running or not.

Upon receiving a notification, the user will also receive a text message from the Raspberry Pi as described above. This message will contain a clickable link to Google drive with the appended file ID generated in the python code, meaning in one click the user is taken to the image which triggered the system.

The open button in app can also be used to view this image. If the user opens drive in-app, a driveFileOpener Intent is built which gives the user only access to JPeg files, so they do not select a non-image file which could not be used as a Bitmap. Therefore, it is important that the python code on the Raspberry Pi specifies the MIME Type

of the file to create. When the user selects the image, a Bitmap is decoded and the imageview in the mainActivity XML file is set to that bitmap, allowing the user to get a larger view of the image in the application's main screen. Alternatively, a link has been set up for if the user chooses to open drive in their device's browser, which will take them directly to a thumbnail view of the drive, which may be useful for extended periods of motion capture, to allow the viewing all files uploaded without opening them individually.

2.3 - System Components

Google Developer's Console

The Developer's Console was used in the generating of SHA-1 and AUTHO2 Keys for the application, which along with the corresponding Json file inserted into the android studio project, allows for authentication and security to be handled. Although not present in the actual running of the system, it is the backbone on which the system operates, and is relied on to securely transfer the sensitive data that this, and indeed any security system deals with.

Google Drive

Google Drive is a cloud-based file storage system, accessible remotely from any computer around the world. This remote capability makes Google Drive an ideal platform on which to present users of SecuriPi their files in a cost effective and convenient manner. As such, in this system, the Drive acts as a storage bucket for Jpeg files generated on the Raspberry Pi once movement has been detected on an attached web camera. Google drive plays an integral part in the SecuriPi system, as it is not only the medium through which users can view images taken by their camera remotely, but also acts as an authentication mechanism for the application itself. The user simply signs in with their preexisting google account (or indeed create one), and can begin using the app.

Firebase Messaging

Firebase is a web application designed by google, facilitating push notifications in the SecuriPi system. A POST request is generated in python, and sent to google firebase's servers with a unique token. This token was generated in the android application's onCreate method. The application automatically uses an SSH connection to access the terminal of the Raspberry Pi, and assign the token to a variable in the python code. The POST request contains the message title and body, and is immediately delivered to the corresponding user's mobile device. Firebase can also be used to manually create messages to send out to specific users, users subscribed to a specific topic, or to all app users, which will no doubt be useful in notifying users of upcoming patches, bugs, or general news.

WAMP Server

WampServer refers to a web server software stack consisting of the Apache web server, OpenSSL, MySQL database, and PHPMyAdmin. This tool plays an important role in registering users for notifications with firebase messaging. Once a user connects to the application, the unique token mentioned above is generated and sent in a POST request to a PHP script located on the Apache web server, which stores the token in a mySQL database, and allows for the storage of unique user keys. These are the keys that are used to send user-specific notifications to the application.

VLC Player

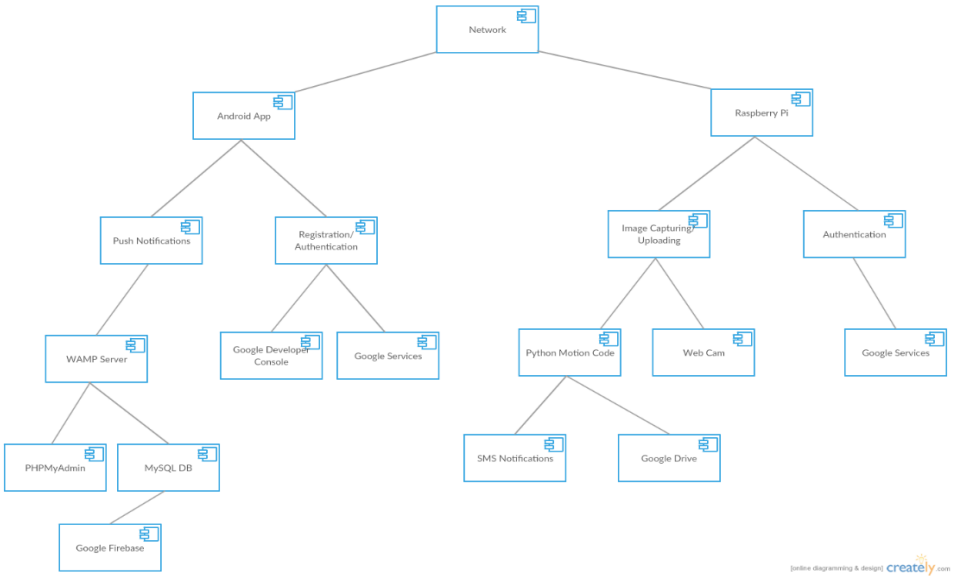
The sound output functionality on the pi relies on VLC player. The user can easily disable this functionality in the conf.json file if they so wish.

TextLocal

TextLocal is a 3rd party SMS service that allows us to send SMS notifications to the user's phone directly from the python code on the pi. The user must create an account with TextLocal which is free to do, they receive 10 free texts and from then on must purchase credit much like with a normal phone provider. To initialize this the user must simply enter their phone number, email address, and account-specific hash, into our conf.json file to set up their details. The user can disable this service in the conf.json file also if they do not wish to avail of it.

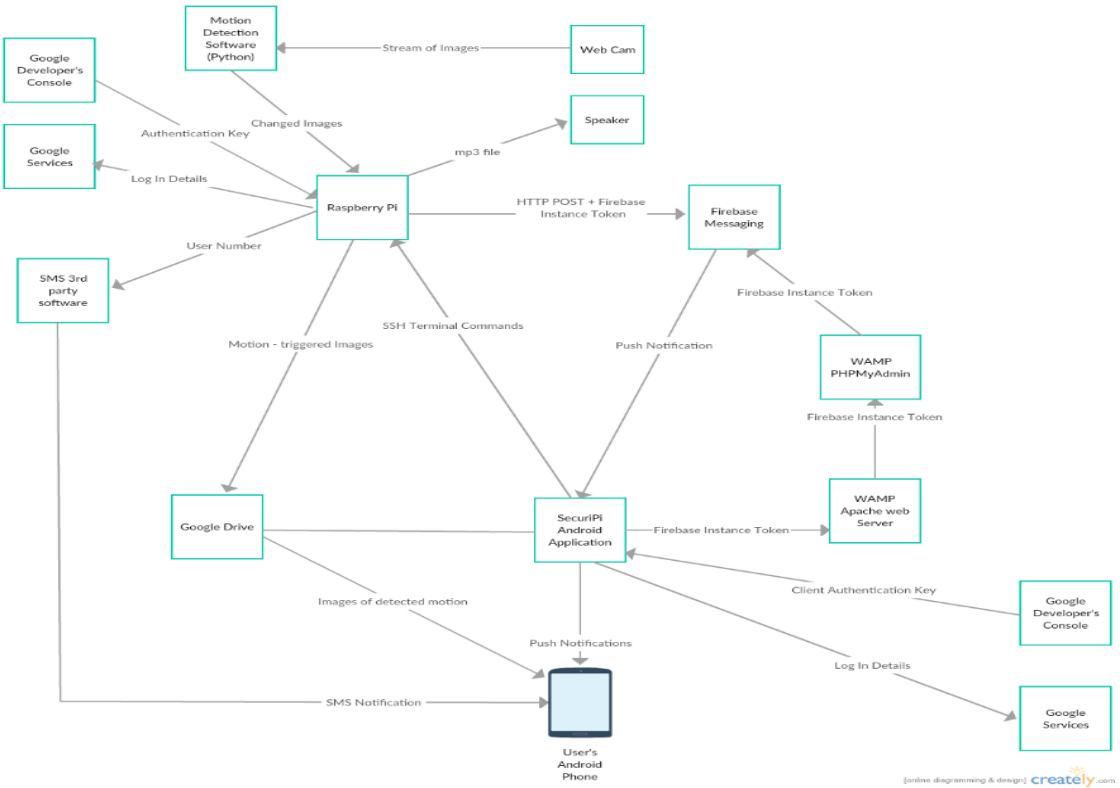
3 – High Level Design

3.1 - Component Model



The Component Model outlines how the system's various components break down into, or interact with other components in order to achieve the system's various tasks. If we look at the push notifications component for the Android App for example, it uses the WAMP server component, which provides the PHP and SQL components, which then interacts with the Firebase component in order to register the user to receive push notifications

3.2 - Data Flow Diagram



The data flow Diagram documents the various pieces of data sent between the components of the system. Above is a complete overview of the system's various components and the data they must send/receive for the system to function.

4 – Problems and Resolution

Alarm sounds freezing capture frame

The last component to be added to this project was the camera, which originally had been connected using the GPIO connectors. However, after realizing the sound limitations of this connection method, the team opted for a portable camera connected through a 3.5mm audio jack. An OS system call from python was originally used to execute OmxPlayer, a built-in music player on the pi, and force the mp3 file through local output using "local -o sound.mp3". Unfortunately, when the alarm would sound, the motion detection camera frame would freeze until the music had stopped playing and throw a resource busy warning the next time the camera was armed.

Solution: Used VLS Player instead, as OmxPlayer apparently utilizes the GPU of the device.

Dropbox support depreciated

As mentioned in the functional specification deliverable, Dropbox was originally to be used as the image-dump of the motion captured images. Unfortunately, Dropbox's previously well supported android SDK is no longer supported, and while the development API has been updated to a new version, it does not yet include a well-documented android library.

Solution: Google Drive has instead been used, which allows all the same functionality of Dropbox, and means there is less of a set up as far more people today have a google account.

Port forwarding for remote view of live stream server

SSH connections require that port 22 is open on the router that the Pi is connected to. As the team did not have access to this during development, either in student accommodation or indeed in DCU, it is not possible to set up a reliable SSH connection outside the local network for demonstration purposes.

Solution: Unfortunately, this is a limitation of the device in terms of demonstration. However, in a home network it is possible to set up a link to a live stream hosted on the pi, if port 22 is open.

Creating RESTful architecture and Web-hook for notifications

Originally in order to send push notifications when a file uploaded to drive, a web-hook was set up and a POST request sent to it. Google's servers would then poll the drive account, send a response to the web hook which would notify the application to create a notification. This method was very complicated and did not account for files being uploaded that were not related to this system, so any file uploaded to drive would notify the application.

Solution: The team worked around this issue by implementing the POST request on the Pi rather than from the application. Each time the Pi uploads the image, the POST request is sent to Google's Firebase server. This means that only SecuriPi related files notify the users device, and avoids the unnecessary construction of a RESTful API.

Notification Token needed to be hardcoded on Raspberry Pi:

A firebase token is required for the notification to be sent to a specific user when a drive file is uploaded. Unfortunately, this token is generated on the application, and as the notification process is now begun from the Pi, there needed to be some way to give the python code on the Pi the unique token that is generated each time a new client connects to the application.

Solution: An SSH session is established automatically in-app, on connection to a google drive account, which makes an "exec" connection and uses a "sed" terminal command to change the previous token (which is stored in the application as a string variable) to the newly generated token.

Motion detection alert if phone has no internet connection

One problem with the notification system was its reliance on a network connection to function. There may be a case when the user is out of data, not near a Wi-Fi spot, or simply has bad connection and does not receive a notification when the alarm is triggered.

Solution: A third-party SMS service has been used to send a pre-defined mobile phone number a notification via text, as well as a URL link to google drive with the appended file ID that has been created and uploaded in the python motion detection program.

Keep track of all client tokens for mass / selective notification

The application needed some way to store registration tokens in order to send to Firebase to identify a token as being a valid user, and send a notification.

Solution: An WAMP server was set up and used to host a PHP script and MySQL database which stores the user's uniquely generated tokens in a database.

User having to open the Pi-side code to change Alarm Sound

The application needed some way for the user to choose which alarm sound to play on motion detection based on what purpose the system will be used for be it monitoring a child, front door monitoring, security, etc. Originally this had to be done from the Raspberry Pi itself.

Solution: A sed command is used over SSH to search for the specific class path of the file, and change all occurrences of ".*mp3" to whichever option the user chooses, meaning the sound can be changed from the application eliminating the need for a monitor or peripherals on the Raspberry Pi itself.

Shutting down SecuriPi remotely

While arming the device through SSH was doable, there is no bash command to stop the motion capture and alarm sound if the user looked at the images being uploaded and determined that it was a false flag.

Solution: As it is the only python program running on the device, it was deemed safe to simply kill all python processes using an SSH connection from the application itself.

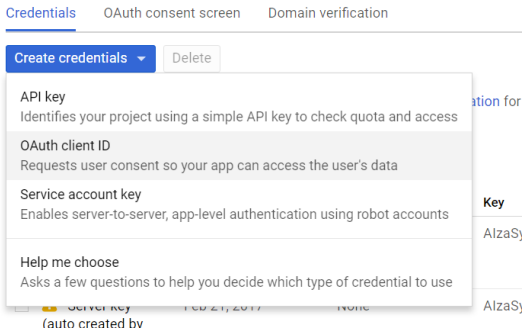
5 – Installation & Configuration

5.1 - Application Configuration & Initial Set up

The application can be downloaded from the Git Repository, or once it is published, from Google Play Store.

Once downloaded from Git, the application may be opened using android studio.

An SHA-1 Authentication key must be acquired from Google Developer Console in order to run this application from an emulator and gain access to google services. This requires the user to go to the development console and create a new project. Under the Credentials section, choose 0auth client ID:



Choose the Android application type, enter the package name from the AndroidManifest.xml file in the Android studio project, and enter the SHA1 Key.

This SHA1 key can be found by entering the following command:

```
C:\Program Files\Java\jdk1.8.0_60\bin>keytool -exportcert -alias androiddebugkey -keystore "C:\Users\ross\.android\debug.keystore" -list -v
Enter keystore password:
Alias name: androiddebugkey
Creation date: Jan 30, 2017
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: C=US, O=Android, CN=Android Debug
Issuer: C=US, O=Android, CN=Android Debug
Serial number: 1
Valid from: Mon Jan 30 18:48:02 GMT 2017 until: Wed Jan 23 18:48:02 GMT 2047
Certificate fingerprints:
         MD5:  40:77:6C:86:E8:CE:F9:00:A6:5B:E0:09:2C:2C:F5:39
         SHA1: AE:EA:70:37:E3:1B:C6:B2:F6:CC:5B:09:A5:21:DA:58:55:15:13:27
         SHA256: 4A:12:42:B8:7D:1F:D9:66:33:1E:6C:B7:A4:BE:1D:AC:95:41:8B:AB:49:EC:ED:A9:30:81:AE:D6:40:4B:8D:F7
         Signature algorithm name: SHA1withRSA
         Version: 1
```

Where the password is set to android by default. Now click create and the android studio project should be ready to run the application. Please ensure the Studio and Emulator versions are up to date, and that the Google Drive, Token Service, and Firebase APIs are enabled in the console's API manager. A Server will also be needed (WAMP if localhost), with the PHP script and a database written using the SQL code provided.

5.2 - Raspberry Pi Configuration & Initial Set up

Step 1: Expand Filesystem:

- "sudo raspi-config"
- Select first option "Expand File System", hit "Enter" and select "Finish".
- Use "sudo reboot" to reboot your pi.

Step 2: Install Dependencies:

- Upgrades and updates any existing packages.
  - "sudo apt-get install build-essential cmake pkg-config"
- Developer tool to configure OpenCV.
  - "sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev"
- Image I/O packages to load image formats.
  - "sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev"
  - "sudo apt-get install libxvidcore-dev libx264-dev"
- Video I/O to read video formats and allow video streams.
  - "sudo apt-get install libgtk2.0-dev"
- Enables use of OpenCV submodule "highgui" to allow images to be displayed to screen.
  - "sudo apt-get install libatlas-base-dev gfortran"
- Optimises certain OpenCV operations
  - "sudo apt-get install python2.7-dev"

3: Download OpenCV source Code

- Get source Code
  - "cd ~"
  - "wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.1.0.zip"
  - "unzip opencv.zip"

- Downloads and unzips OpenCV 3.1.0

4: Pip and NumPy

- Get source Code
    - "wget https://bootstrap.pypa.io/get-pip.py"
    - "sudo python get-pip.py"
- Install pip, python package manager.
    - "pip install numpy"
- Install NumPy, used for numerical processing.

5: Compile and Install OpenCV

- "cd ~/opencv-3.1.0/
- mkdir build
- cd build
- cmake -D CMAKE_BUILD_TYPE=RELEASE \

  -D CMAKE_INSTALL_PREFIX=/usr/local \

  -D INSTALL_PYTHON_EXAMPLES=ON \

  -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.1.0/modules \

  -D BUILD_EXAMPLES=ON .."
- "make –j4"
    - Compiles OpenCV. Allows use of 4 cores of raspberry pi for quicker install.
- "sudo make install"
- "sudo ldconfig"
    - Installs OpenCV 3.1.0.

6: Finish Installing OpenCV

- "ls –l /usr/local/lib/python2.7/site-packages/"

This command should show presence of "cv2.so". If not here check "dist-packages" instead.

7: Check Installation

- "python"
- "import cv2"
- "cv2.__version__"
- Should return your correctly running python 2.7.9 and CV 3.1.0

Download GIT files

"sudo git clone git@https://gitlab.computing.dcu.ie/franeyr3/2017-CA326-franeyr3-SecuriPi/tree/master/code/motion_Code securiPi"

- Allows the user to download all the code into a folder called securiPi.

Create API and Download Clients File (Google Authentication)

Step 1:

Go to this address: https://console.developers.google.com/flows/enableapi?apiid=drive and create a project. Use the go to credentials button.

Step 2:

On the "Add Credentials to your project" page click Cancel.

Step 3:

Go to the "OAuth consent screen" tab. Select an Email Address and Product Name and click Save.

Step 4:

Select the "Credentials" tab, click Create Credentials, and select OAuth client ID.

Step 5:

Select the application type "Other". Enter your Product Name and click Create.

Step 6:

Click the download icon and download the JSON file. Once downloaded, rename to "clients_secrets.json" and move it into your securiPi folder where the code is stored. This enables the authentication process for your Google Account.

Install VLC

- "sudo apt-get install vlc"
- Go to the following address:
    - http://git.videolan.org/?p=vlc/bindings/python.git;a=blob_plain;f=generated/vlc.py;hb=HEAD
- Copy the code into a file called "vlc.py", and place this file in your dist-packages for python.
    - "sudo mv /home/pi/vlc.py /usr/local/lib/python2.7/dist-packages"

Enabling Auto-Start of Process

- "cd ~/.config/lxsession/LXDE-pi"
- "sudo nano autostart"
- Add the line "@python /home/pi/securiPi/pi_surveillance.py –conf /home/pi/securiPi/conf.json
- To save and exit: "Ctrl + x", "Y", "Enter"

The SecuriPi development team would like to thank you for choosing our system, and are confident that the time spent with SecuriPi home surveillance system will be an enjoyable one.

Regards.

-Team SecuriPi