PROJECT REPORT

IoT Software Development

Higher Diploma in Science in Computing

Internet of Things Online Stream Sep 2017

Name:	Richard Seaman
Student Number:	17119111
Date:	07/05/2018
Lecturer:	Cristian Rusu

INDEX

1. INTRODUCTION

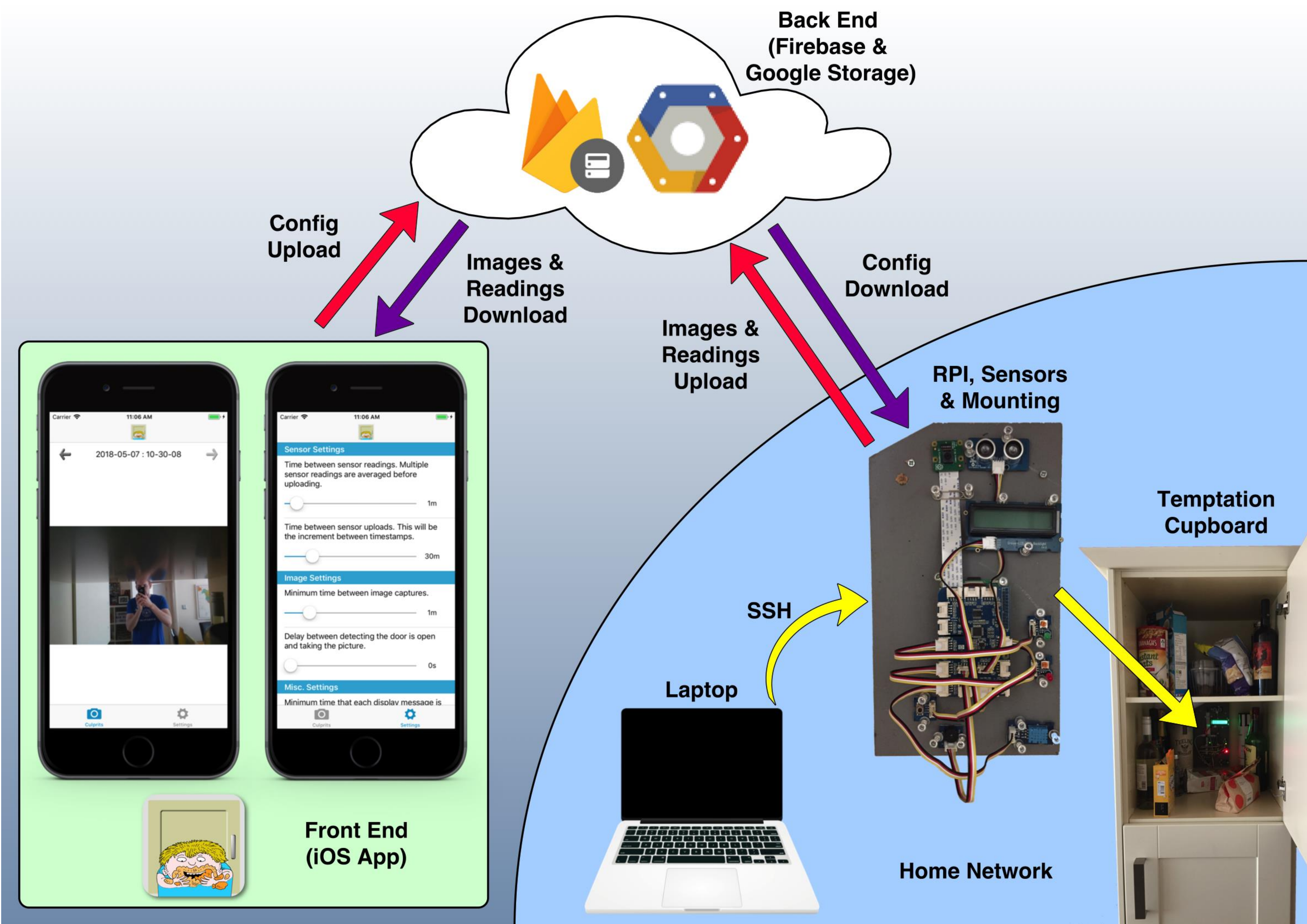
This report was written by Richard Seaman and forms part of the Project submission for the module IoT Software Development.

The project has not changed significantly from the Project Proposal. This report is intended to follow on from the proposal document.

A visualisation of the entire project and all of its components is provided overleaf.

The hardware, software and network components of the project are then outlined in turn and discussed in detail.

Finally, links to the source code and video presentations are provided at the end of the report.



Sensor Settings

Time between sensor readings. Multiple sensor readings are averaged before uploading.

1m

Time between sensor uploads. This will be the increment between timestamps.

30m

Image Settings

Minimum time between image captures.

1m

Delay between detecting the door is open and taking the picture.

0s

Misc. Settings

Minimum time that each display message is

2. HARDWARE

An actual image of the hardware used is shown overleaf. The hardware is laid out and fixed in position on the mounting. This image and the accompanying labels serve as the schematic representation of the hardware.

2.1 Mounting

The mounting is a simple off-cut of MDF, which is screwed in place at the rear of the cupboard. The hardware is attached to the mounting using thumbtacks.

2.2 Raspberry Pi 3 Model & GrovePi

The Raspberry Pi and GrovePi are mounted centrally, within reach of all other components. The raspberry pi is used as the main computational platform.

2.3 Raspberry Pi Camera Module V2

The camera is used to capture images of the cupboard culprits. It is positioned at the top of the mounting to allow a clear line of sight over any potential bottles / treats etc. This provides the best opportunity to capture a clear image of the person who opened the cupboard door.

The camera is the only component that doesn’t connect to the GrovePi interface, but directly to the raspberry pi. The camera is held in place using thumbtacks and a paper clip. This allows final, slight adjustments of the camera position to be made in situ.

2.4 Grove Ultrasonic Ranger Sensor

The ultrasonic ranger is used to detect when the cupboard door is open. Similarly, to the camera, it is positioned at the top of the mounting to provide the best line of sight to the cupboard door.

2.5 Grove RGB Backlit LCD

The LCD screen is used to provide visual feedback (text and colour) on the program operation and status including the door open (or “Raid”) count, configuration updates and image uploads.

2.6 Grove Temperature & Humidity Sensor (blue)

The temperature and humidity sensor is used to take temperature and humidity readings within the cupboard. It is positioned below the raspberry pi so that it will not be affected by any heat gains rising from it.

2.7 Grove Red / Green LED

The red LED turns on when the door is detected as open. It is used predominantly for troubleshooting (if something is blocking the ranger while the door is open for example).

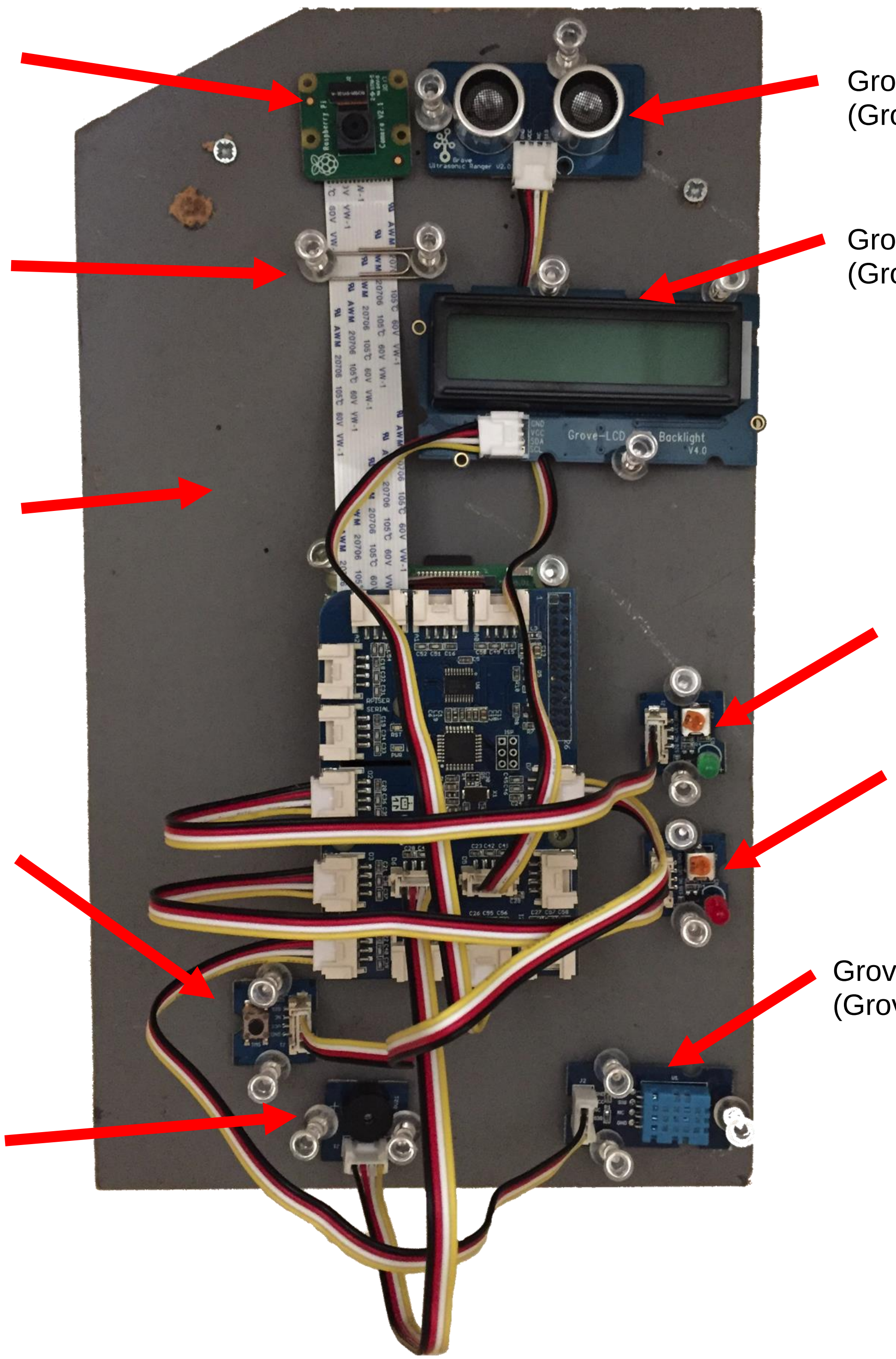
The green LED turns on when the required time between image captures has passed (i.e. ready to take a picture).

2.8 Grove Buzzer

The buzzer is used to provide a noise deterrent when the door is open and the maximum daily count has been exceeded.

2.9 Grove Button

The button is used to shut down the raspberry pi in a safe manner.



Raspberry Pi Camera Module V2 (camera connector on raspberry pi between Ethernet and HDMI ports)

Camera adjustment mechanism

Medium-density fibreboard (MDF) mounting, screwed to rear of cupboard.

Grove Button (GrovePi port: D7)

Grove Buzzer (GrovePi port: D6)

Grove Ultrasonic Ranger Sensor (GrovePi port: D5)

Grove RGB Backlit LCD (GrovePi port: I2C - 1)

Green LED (GrovePi port: D2)

Red LED (GrovePi port: D3)

Grove Temperature & Humidity Sensor (GrovePi port: D4)

3. SOFTWARE

A single program, “Main.py”, runs on the Raspberry Pi. This program is automatically executed when the system boots up. This is achieved by adding an @reboot command to the crontab file.

Main.py is written in python and is well documented. Please refer to the GitHub links provided at the end of this report.

A detailed flow diagram for Main.py is included overleaf and includes three main parts;

1. Initialisation – all of the initial setup and housekeeping required
2. Main Loop – an indefinite loop which carries out the main logic of the program
3. Image Processing Loop – an indefinite loop which runs in the background and checks for any saved images to process.

3.1 Operation

The operation of the program is evident from its source code. However, a brief overview of how the program operates is given below.

When the program begins, it creates a log file where all of it’s information and error messages will be logged. It then registers handlers for various signal interrupts so that it gracefully shuts down. Local “Images” and “Archive” folders are created if required. Firebase and Google Cloud Storage are initiated and authenticated. The image processing loop is kicked off in the background before beginning the main loop.

A number of time intervals are defined which control how often each of the periodic tasks within the main loop are carried out. If sufficient time has passed since the previous time a task was done, the task will be performed. Periodic tasks include; syncing configuration settings, updating the display, taking sensor readings and uploading average sensor readings.

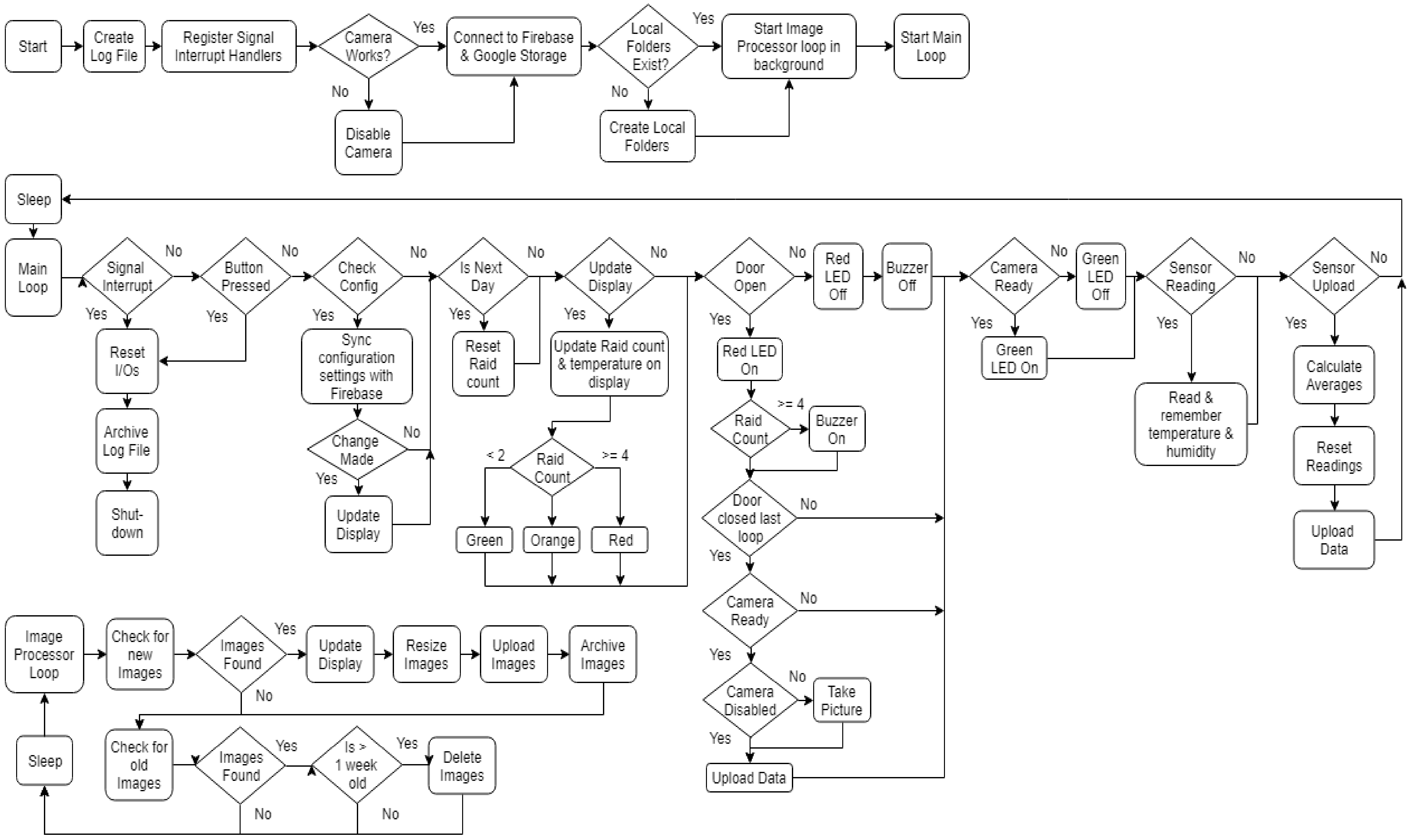
Regardless of how much time has passed, each cycle checks whether the door is open and performs the required tasks if it is or isn’t. The ultrasonic ranger seemed to give a number of false readings and confused the program into thinking the door was open. As a result, in order for the door to be considered genuinely open, a number of door open readings must be recorded in a row.

Images that are captured are processed by the background loop. They are resized (reducing their file size by approximately 98%), uploaded and archived. Any archived images older than a week are also deleted to save space.

3.2 Front End

An iOS application was developed to serve as the front end for the project. The app allows the configuration settings to be adjusted. It also allows the user to browse through the images of culprits. Due to time constraints, the temperature and humidity readings are not available in the application, although they can be viewed directly in firebase.

3.3 Main.py - Flow Diagram



4. NETWORK

Firebase and Google Cloud Storage are used for the backend of this project and provide the Network component.

4.1 Firebase Authentication

The data in the real time database and the images in google cloud storage are not made publicly available to everybody. Firebase authentication is used to ensure only users with the required privileges may access it.

The Raspberry Pi has a local copy of an authentication file which is used to grant access to the data. Note that this file is not included in the git repository but it’s path and use can be seen in the source code. The iOS app requires the user to sign in through Google in order to access the data.

4.2 Firebase Realtime Database

A real-time database is used to store the data for the project. The structure of a Firebase Realtime Database follows a JSON tree format.

There are three root nodes used; conditions, config and culprits. Example data from each node is provided overleaf.

Conditions

The conditions node contains the temperature and humidity readings. These are the average readings, periodically uploaded by Main.py.

Config

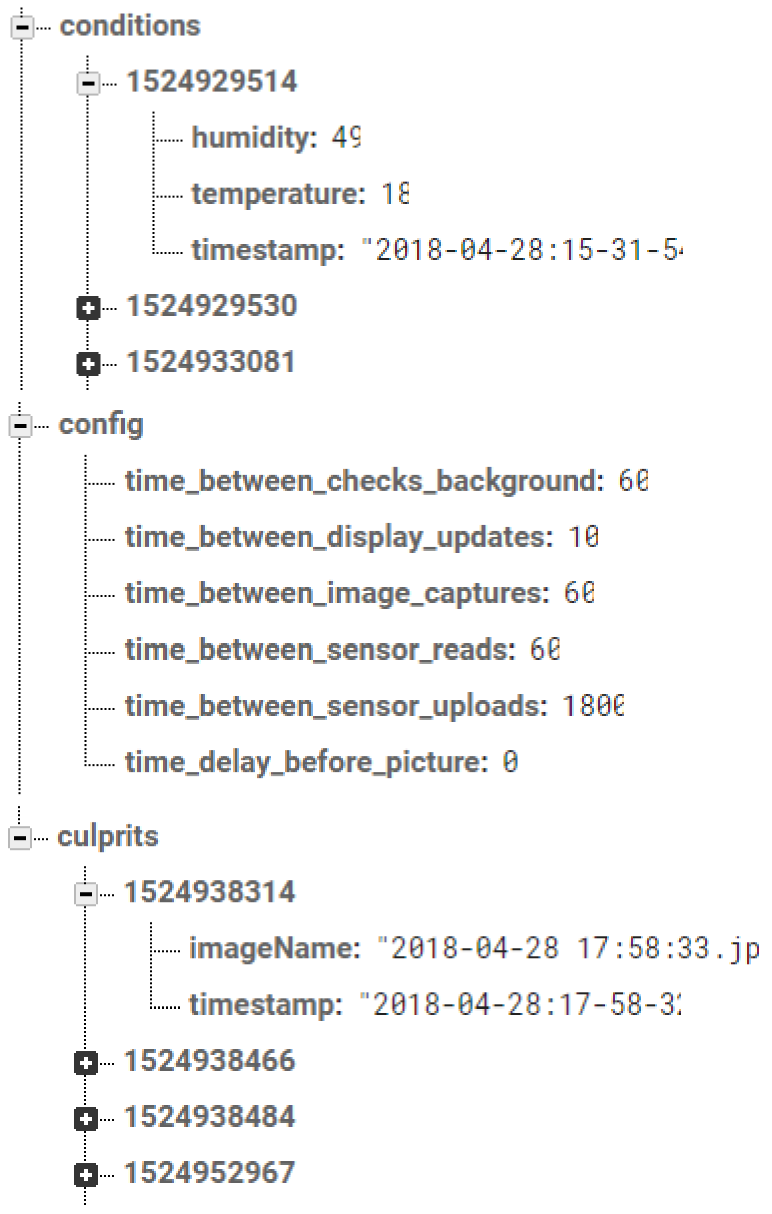
The config node contains all of the adjustable configuration values for the Main.py program. These values are set and uploaded by the iOS application (which includes validation). The values are then synced within the Main.py program.

Culprits

The culprits node contains the image file name and timestamp for each instance that Main.py detected and captured a culprit. This information is uploaded by Main.py. Note that the image file itself is not included here. This information is queried by the iOS application, which in turn knows where to download the actual image file, based on the image name.

4.3 Google Cloud Storage

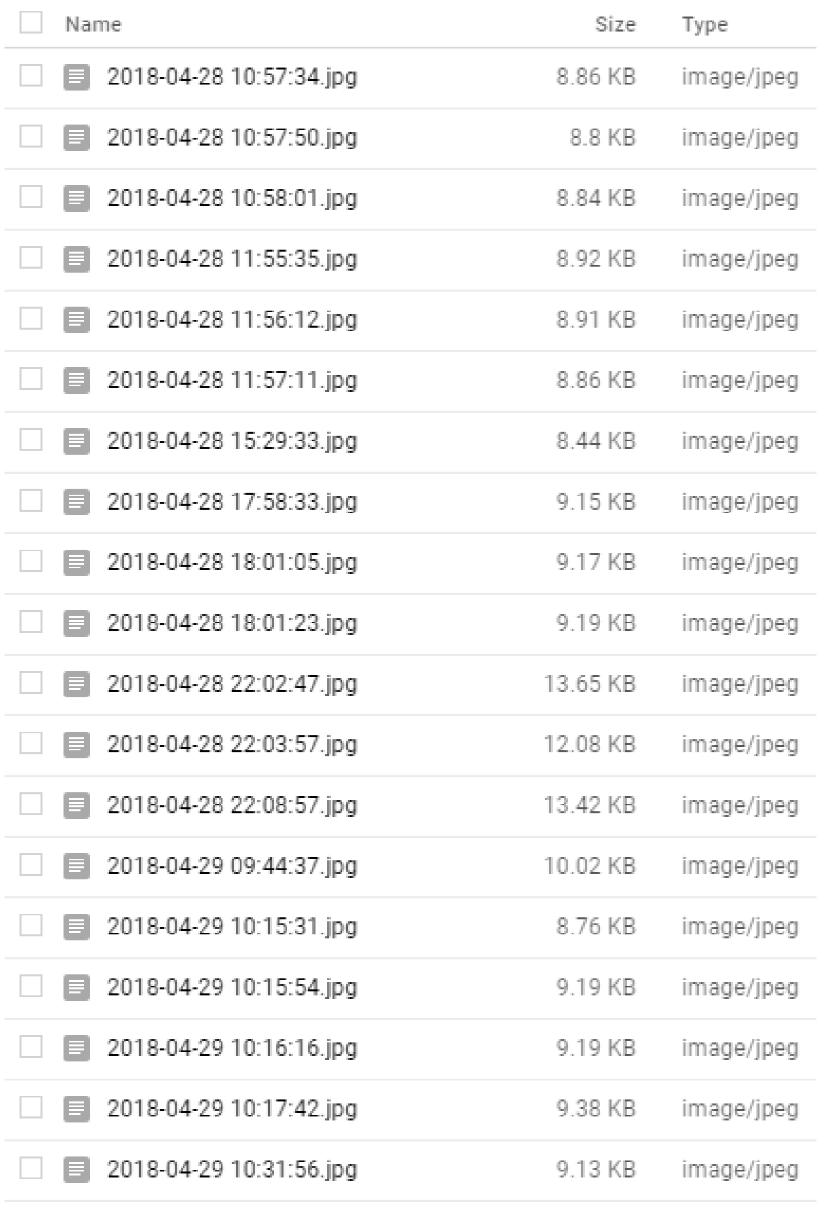
Google cloud storage is tightly integrated with Firebase and can share the authentication process. The background loop within Main.py uploads the resized images to a bucket on google cloud storage. The location and URL of this bucket is known by the iOS application so it can download any image it needs by simply appending the image file name.



```
conditions
    1524929514
        humidity: 49
        temperature: 18
        timestamp: "2018-04-28:15-31-5
    1524929530
    1524933081
config
    time_between_checks_background: 60
    time_between_display_updates: 10
    time_between_image_captures: 60
    time_between_sensor_reads: 60
    time_between_sensor_uploads: 1800
    time_delay_before_picture: 0
culprits
    1524938314
        imageName: "2018-04-28 17:58:33.jp
        timestamp: "2018-04-28:17-58-3
    1524938466
    1524938484
    1524952967
```

Example Data from Realtime Database

(Note: epoch time used as key for timeseries data)



Name	Size	Type
2018-04-28 10:57:34.jpg	8.86 KB	image/jpeg
2018-04-28 10:57:50.jpg	8.8 KB	image/jpeg
2018-04-28 10:58:01.jpg	8.84 KB	image/jpeg
2018-04-28 11:55:35.jpg	8.92 KB	image/jpeg
2018-04-28 11:56:12.jpg	8.91 KB	image/jpeg
2018-04-28 11:57:11.jpg	8.86 KB	image/jpeg
2018-04-28 15:29:33.jpg	8.44 KB	image/jpeg
2018-04-28 17:58:33.jpg	9.15 KB	image/jpeg
2018-04-28 18:01:05.jpg	9.17 KB	image/jpeg
2018-04-28 18:01:23.jpg	9.19 KB	image/jpeg
2018-04-28 22:02:47.jpg	13.65 KB	image/jpeg
2018-04-28 22:03:57.jpg	12.08 KB	image/jpeg
2018-04-28 22:08:57.jpg	13.42 KB	image/jpeg
2018-04-29 09:44:37.jpg	10.02 KB	image/jpeg
2018-04-29 10:15:31.jpg	8.76 KB	image/jpeg
2018-04-29 10:15:54.jpg	9.19 KB	image/jpeg
2018-04-29 10:16:16.jpg	9.19 KB	image/jpeg
2018-04-29 10:17:42.jpg	9.38 KB	image/jpeg
2018-04-29 10:31:56.jpg	9.13 KB	image/jpeg

Example Image Files on Google Cloud Storage

(resized prior to upload so very small)

5. LINKS TO SOURCE CODE & PRESENTATION

Please find links below to the various components of this project.

Main.py

The source code for the main python program which runs on the Raspberry Pi is available at the following link:

https://github.com/Richard-Seaman/CupboardCulprit

iOS Application

The source code (Swift) for the iOS application which is used as the front end of this project is available at the following link:

https://github.com/Richard-Seaman/CupboardCulpritIos

Video Presentation