RASPIGUARD

Submitted By: Vivek Socrates, Karel Tutsu, & Heakeme Williams  
Discipline: Computer Engineering Technology  
Date: N.A. / N.A. / 2018

# Declaration of Joint Authorship

Vivek Socrates, Karel Tutsu and Heakeme Williams confirm that this project is a joint collection of work between the three of us. Any and all outside sources that have been referenced or used will be cited and properly acknowledged at point of use.

# Proposal For Development Of RaspiGuard

Prepared by Vivek Socrates & Karel Tutsu

Computer Engineering Technology Students

<https://github.com/V-Socrates/RaspiGuard>

## Executive Summary

As a student in the Computer Engineering Technology program, I will be integrating the knowledge and skills I have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet-connected hardware will include a custom PCB with the following sensors and actuators, moisture sensor and magnetic contact switches. The database will store the status of the sensors, log activity information, and will be used to store user accounts for application authentication. The mobile device functionality will include the ability to monitor sensor state, change alarm settings, create and modify account information, add and remove monitoring units, view activity logs, as well as receiving notifications of sensor activity, and will be further detailed in the mobile application proposal. I will be collaborating with Vlad and Kelly in the prototype lab regarding the housing and deployment of the system. In the winter semester, I plan to form a group with the following students, who are also building similar hardware this term and working on the mobile application with me Vivek Socrates, Karel Tutsu and Heakeme Williams. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software Project. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project as a group of three.

## Background

The problem solved by RaspiGuard is that installing a home security system can be costly, but needing one and not having one can cost you even more. Fortunately, you will not have that dilemma with RaspiGuard. RaspiGuard is a cost-effective and easy to install security system that you can easily setup and deploy on your own. RaspiGuard is a home security/surveillance system that is used to monitor a room remotely using our android application. The current operational functionality is door and moisture sensors. We are currently working on the functionality of light sensors as well as various other sensors.

Existing products on the market include [1]. I have searched for prior art via Humber’s IEEE subscription selecting “My Subscribed Content”[2] and have found and read [3] which provides insight into similar efforts.

In the Computer Engineering Technology program, we have learned about the following topics from the respective relevant courses:

* Java Docs from CENG 212 Programming Techniques In Java,
* Construction of circuits from CENG 215 Digital And Interfacing Systems,
* Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
* Micro computing from CENG 252 Embedded Systems,
* SQL from CENG 254 Database With Java,
* Web access of databases from CENG 256 Internet Scripting; and,
* Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable me to build the subsystems and integrate them together as my capstone project.

## Methodology

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the fall semester. My coursework will focus on the first two of the 3 phases of this project:

### Phase 1 Hardware build

The hardware build will be completed in the fall term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

### Phase 2 System integration

The system integration will be completed in the fall term.

### Phase 3 Demonstration to future employers

This project will showcase the knowledge and skills that I have learned to potential employers. The brief description below provides rough effort and non-labour estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines. New moisture sensor due to current moisture sensor reading being imprecise.

## Concluding remarks

This proposal presents a plan for providing an IoT solution for Our product will be a small unit that is wall mountable and can be used to monitor door activity, moisture of plants and other various information that is currently being worked on. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects such as the initiative described by [3]. I request approval of this project.

# Abstract

The problem solved by this project is installing a home security system can be costly, but needing one and not having one can cost you more. Fortunately, you will not have that dilemma with RaspiGuard. RaspiGuard is a cost-effective and easy to install security system that you can easily setup and deploy on your own. A bit of background about this topic is RaspiGuard is a home security/surveillance system that is used to monitor a room remotely using our android application. The current operational functionality is door and moisture sensors. We are currently working on the functionality of light sensors as well as various other sensors.

# 1. Introduction

RaspiGuard is a home security/surveillance system that is used to monitor a room remotely using our android application. The current operational functionality is door and moisture sensors. We are currently working on the functionality of light sensors as well as various other sensors. Installing a home security system can be costly, but needing one and not having one can cost you more. Fortunately, you will not have that dilemma with RaspiGuard. RaspiGuard is a cost-effective and easy to install security system that you can easily setup and deploy on your own.

Contents

[Declaration of Joint Authorship II](#_Toc512198761)

[Proposal For Development Of RaspiGuard iii](#_Toc512198762)

[Executive Summary iii](#_Toc512198763)

[Background iv](#_Toc512198764)

[Methodology v](#_Toc512198765)

[Phase 1 Hardware build v](#_Toc512198766)

[Phase 2 System integration v](#_Toc512198767)

[Phase 3 Demonstration to future employers v](#_Toc512198768)

[Concluding remarks vi](#_Toc512198769)

[Abstract vii](#_Toc512198770)

[1. Introduction viii](#_Toc512198771)

[2. Project Description 13](#_Toc512198772)

[2.1 Problem 13](#_Toc512198773)

[2.2 Rationale Behind Project 14](#_Toc512198774)

[2.3 Project Scope 15](#_Toc512198775)

[2.4 Requirements Specifications 16](#_Toc512198776)

[2.4.1 Database 16](#_Toc512198777)

[2.4.2 Mobile Application 17](#_Toc512198778)

[2.4.3 Web Interface 18](#_Toc512198779)

[2.4.4 Networked Platform Communication Software 19](#_Toc512198780)

[2.4.5 Microprocessor Firmware 20](#_Toc512198781)

[2.4.6 Hardware Components 21](#_Toc512198782)

[2.5 Project Overview 22](#_Toc512198783)

[2.5.1 Bill of Materials 22](#_Toc512198784)

[2.5.2 PCB and Soldering 23](#_Toc512198785)

[2.5.3 Mechanical Assembly 24](#_Toc512198786)

[2.5.4 Micro SD Card Setup 26](#_Toc512198787)

[2.5.5 SQL Server Setup 27](#_Toc512198788)

[2.5.6 Android Application Setup 28](#_Toc512198789)

[2.5.7 Webpage Setup 29](#_Toc512198790)

[2.5.8 Raspbian Setup 30](#_Toc512198791)

[2.6 Problems Encountered 34](#_Toc512198792)

[2.6.1 PiTFT Resolution 34](#_Toc512198793)

[2.6.2 PiTFT Orientation 34](#_Toc512198794)

[2.6.3 MCP3008 Conflicts With PiTFT 35](#_Toc512198795)

[2.6.4 Sensor Cables Too Short 35](#_Toc512198796)

[2.6.5 Light Sensor Malfunction 35](#_Toc512198797)

[2.7 Accessing Data 36](#_Toc512198798)

[2.7.1 Android Phone Application 36](#_Toc512198799)

[2.7.2 Website 36](#_Toc512198800)

[3. Progress Reports 37](#_Toc512198801)

[3.1 Is-It-Moist 37](#_Toc512198802)

[3.1.1 Report 1 37](#_Toc512198803)

[3.2 RaspiGuard 40](#_Toc512198804)

[3.2.1 Report 1 40](#_Toc512198805)

[3.2.2 Report 2 43](#_Toc512198806)

[3.2.3 Report 3 45](#_Toc512198807)

[4. Conclusions 47](#_Toc512198808)

[5. Recommendations 49](#_Toc512198809)

[5.1 Hardware 49](#_Toc512198810)

[5.2 PCB 49](#_Toc512198811)

[5.3 Software 49](#_Toc512198812)

[6. Technical References 51](#_Toc512198813)

[7. Appendices 53](#_Toc512198814)

[7.1 RaspiGuard Unit 53](#_Toc512198815)

[7.1.1 RGsetup.sh 53](#_Toc512198816)

[7.1.2 RG.py 54](#_Toc512198817)

[7.2 SQL Database 60](#_Toc512198818)

[7.3 SQL Database PHP Scripts 60](#_Toc512198819)

[7.3.1 60](#_Toc512198820)

[7.3.2 60](#_Toc512198821)

[7.3.3 60](#_Toc512198822)

[7.3.4 60](#_Toc512198823)

[7.4 Website Code 60](#_Toc512198824)

[7.5 Android Phone Application 60](#_Toc512198825)

[7.5.1 60](#_Toc512198826)

# 2. Project Description

## 2.1 Problem

Installing a home security system can be costly, but needing one and not having one may cost you more. Fortunately you will not have that dilemma with RaspiGuard. RaspiGuard is a cost-effective and easy to install security monitoring system that you can easily setup and deploy on your own.

## 2.2 Rationale Behind Project

RaspiGuard is a home security/surveillance system that is used to monitor a room remotely using our android application. The current operational functionality is door and moisture sensors. We are currently working on the functionality of light sensors as well as various other sensors.

## 2.3 Project Scope

RaspiGuard has a $250.00 budget. The development time period for RaspiGuard is from September 11, 2017 to April 23, 2018. The functionality will include a stationary unit with an LCD display, which will display live readings. SQL database to store and log readings as well as user data. Readings and activity log will be accessible using website or Android application.

## 2.4 Requirements Specifications

### 2.4.1 Database

In our application we will be using the standard MySQL (version 5.0.27). The database will store user account information, the live status of sensors and logs sensor activity. All the commands and table operations are very basic such as INSERT, SELECT, UPDATE queries, which are supported in earlier versions as well. There will be a separate table for users, activity log and every sensor will have their own separate table in the database. Karel will lead this development.

### 2.4.2 Mobile Application

Our Android mobile application will be the primary end user interface to manipulate with sensors and data. The core functionality will include:

* Ability to add, modify or remove sensors from their account
* Ability to view the current status of a chosen sensor
* Ability to toggle sensor alarm feature ON/OFF
* Ability view the event history (activity log) of the sensor
* Ability to change user account information

Karel will lead this development.

### 2.4.3 Web Interface

The web interface will serve as a secondary interface to the user – the main benefit being portability. The user will be able to do all the same things as in the mobile application above just simply in a web browser. Karel will lead this development.

### 2.4.4 Networked Platform Communication Software

On the Broadcom BCM2837 (Raspberry Pi) device there will be software that will connect to an MySQL database hosted on cloud services provider [*DigitalOcean*](https://www.digitalocean.com/)*.* The device will be authenticated using credentials stored on the database and will update sensor fields at constant regular intervals. Vivek will lead this development.

### 2.4.5 Microprocessor Firmware

On the Broadcom device, there is a python script that will read raw data from analog and digital sensors. The data will be converted, formatted, and displayed using the standard measurement for each individual type of sensor. Moisture sensors will display humidity in percentage, door sensors will show status as “Open” or “Closed”, and light sensors will display luminous emittance in lux values. Vivek will lead this development.

### 2.4.6 Hardware Components

Components include:

* Raspberry Pi 3 Model B
* Raspberry Pi Power Adapter
* adafruit PiTFT 3.5” Touchscreen
* Magnetic Contact Switch
* SparkFun Moisture Sensor
* Photoresistor
* adafruit ADS1115
* MicroSD card
* Laser Cut Housing / Case

These components will be cased inside the laser cut housing. The components will be inaccessible to the consumer and will power up and start-up the custom written software upon the board being powered. HDMI, USB, audio, and Ethernet ports will be blocked, disabled and inaccessible. The sensors will be attached to the PCB that will be inside the case. Vivek will lead this development.

## 2.5 Project Overview

### 2.5.1 Bill of Materials

|  |  |
| --- | --- |
| Component | Cost |
| Raspberry Pi Model B | $39.95 |
| adafruit PiTFT Plus 480x320 3.5" | $44.95 |
| Raspberry Pi Power Adapter | $16.95 |
| adafruit ADS1115 | $14.95 |
| ADATA Premier 8GB microSDHC UHS-I Class 10 | $8.99 |
| ADATA microReader Ver.3 microSDHC | $4.99 |
| OctoPrint Raspberry Pi Rig 3.5" PiTFT Touch Display | $38.07 |
| SparkFun Soil Moisture Sensor | $5.95 |
| Photo Cell (CdS Photoresistor) | $0.95 |
| Magnetic Contact Switchs | $3.95 |
| Raspbian (OS) | $0.00 |
| 2x13 Female Header | $0.95 |
| 10K Resistor | $0.25 |
|  | |
| SUBTOTAL | $180.90 |
| TAX | $23.52 |
| TOTAL | $204.42 |

### 2.5.2 PCB and Soldering

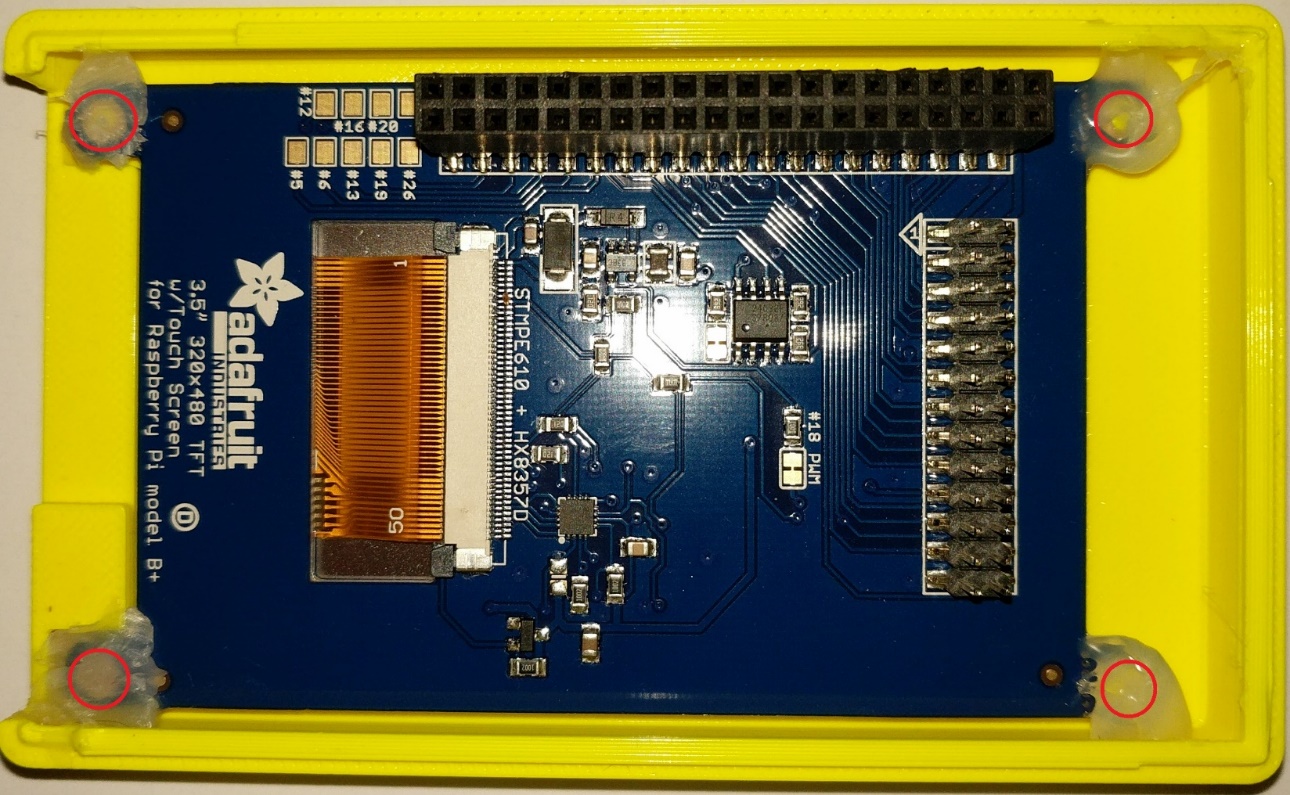
<TO BE FILLED>

### 2.5.3 Mechanical Assembly

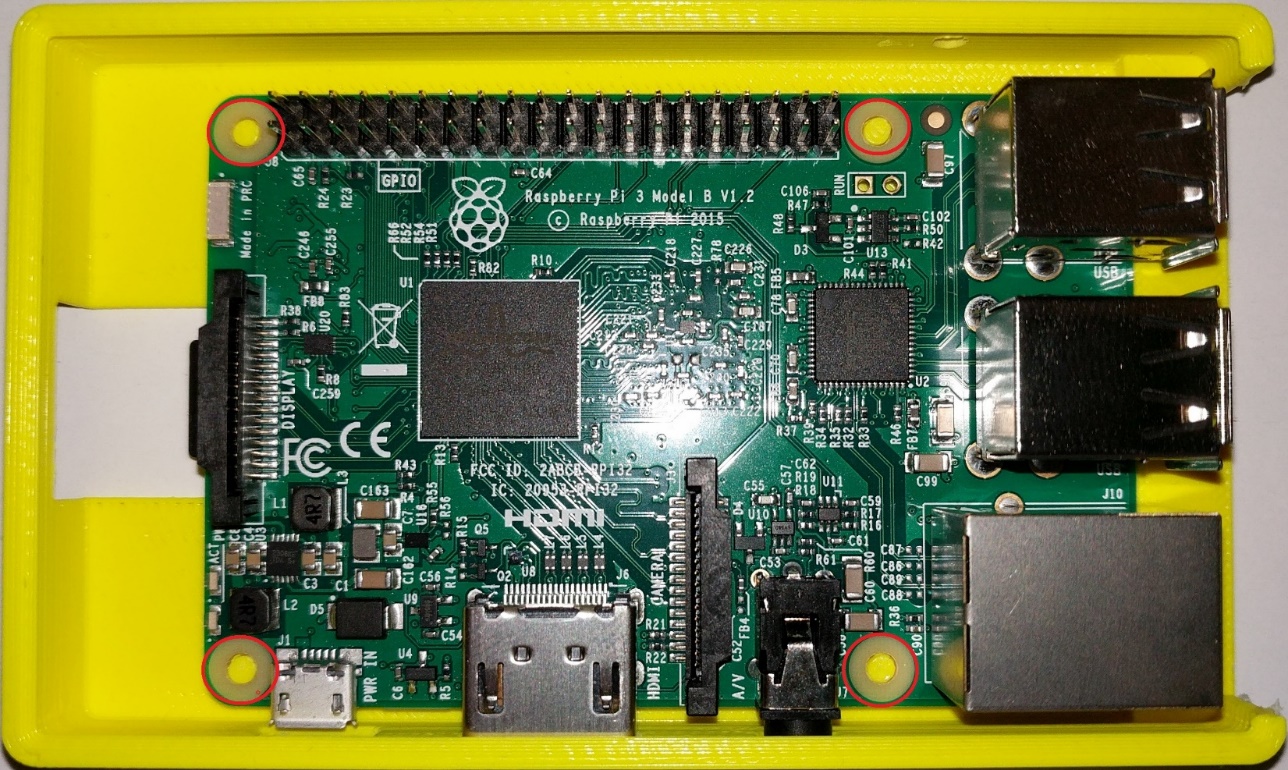




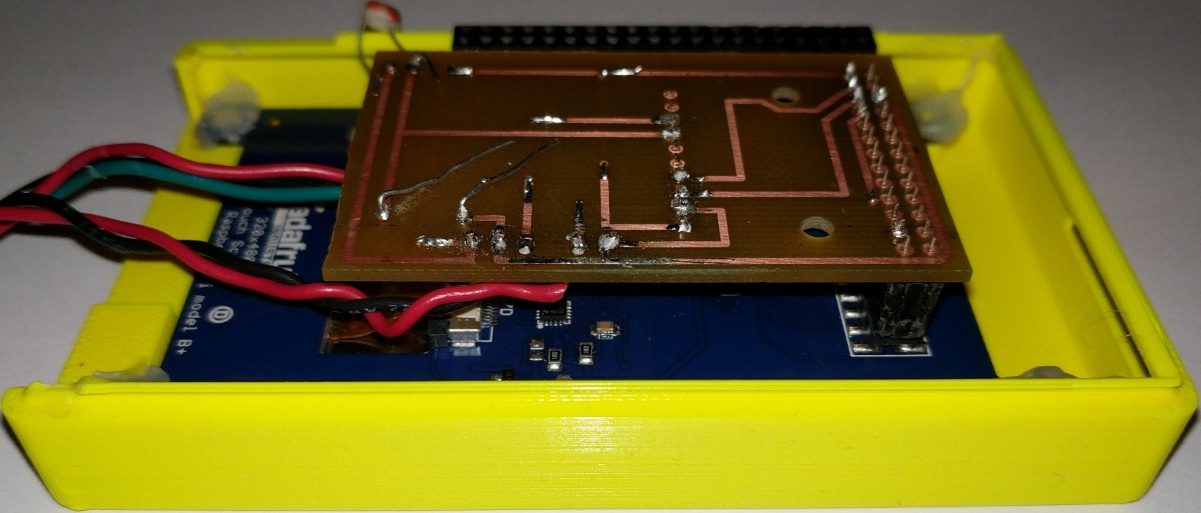
Drill small hole on the bottom side of the case in the space between the GPIO pins and the USB ports. Refer to above pictures.



Hot glue the PiTFT to the top of the case.



Hot glue the Raspberry Pi to the bottom case.



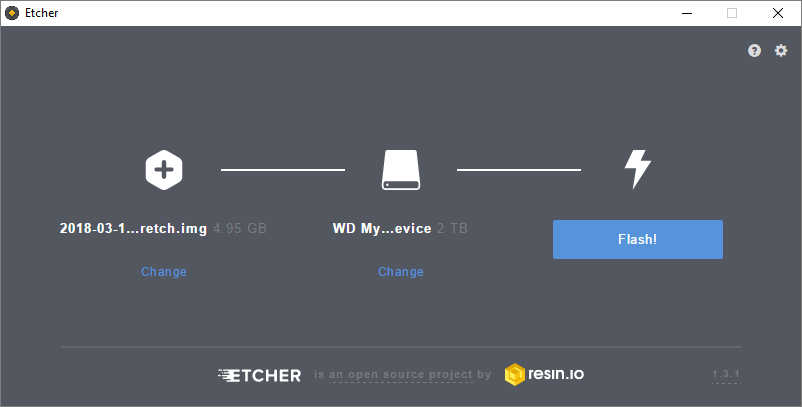
Connect the PCB to the bottom of the PiTFT.



Connect the PiTFT to the Raspberry Pi and close the case. Make sure the photo resistor is aligned with the hole that was drilled.

### 2.5.4 Micro SD Card Setup

Download a program called [Etcher](https://etcher.io/) and the latest [Raspbian](https://www.raspberrypi.org/downloads/raspbian/) OS with Desktop.



Open etcher and click “Select image”. Navigate to where you Raspbian OS is located and select “Open”. Next click on “Select drive” and choose your Micro SD card and click “Continue”. After completing there steps click “Flash!” and wait for you Micro SD card to be flashed with the image. After completion insert your Micro SD card into the Raspberry Pi.

### 2.5.5 Setting up Apache web-server Setup

Note: this goes without saying that you need root privileges to complete any setup on Linux

It’s always a good idea to make sure your system is up to date beforehand



Next install apache



Note: the default apache file path is /var/www/html/ <- this is where the web-server looks for your website files

### 2.5.6 Setting up MySQL

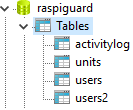
Download the package:



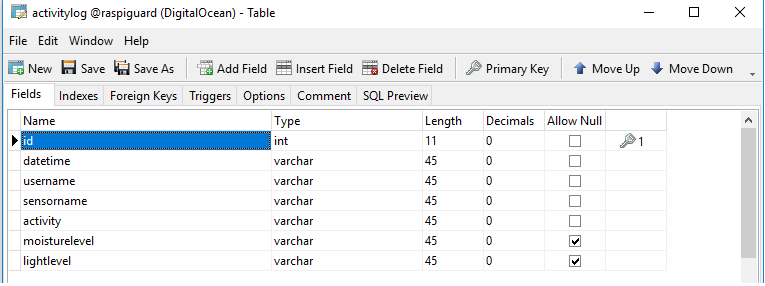
During the installation, your server will ask you to select and confirm a password for the MySQL "root" user. This is an administrative account in MySQL that has increased privileges. Think of it as being similar to the root account for the server itself (the one you are configuring now is a MySQL-specific account, however). Make sure this is a strong, unique password, and do not leave it blank.

#### 2.5.6.1 MySQL Table Structure

I’m going to show below the MySQL structure and tables that we used in our project

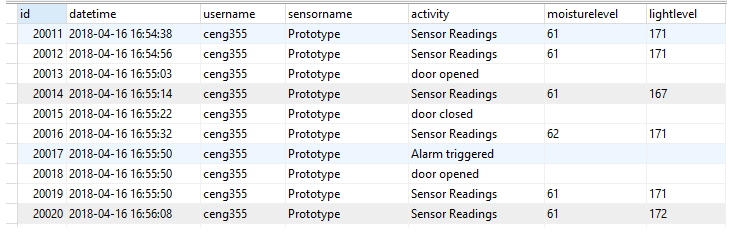


**Table activitylog**

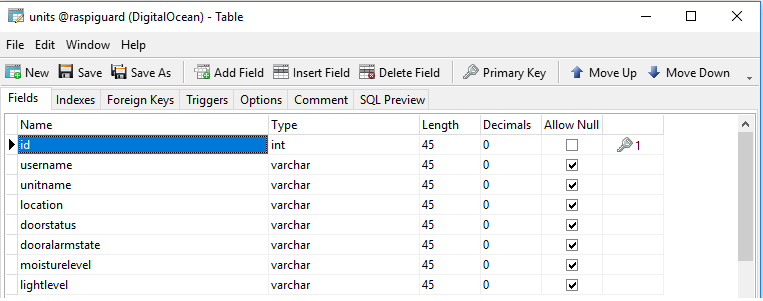


**Purpose**: logs events into the table

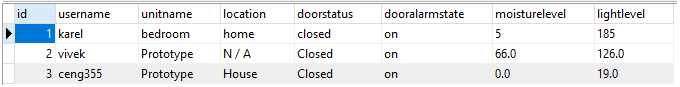
Sample data from our application:



**Table units**

****

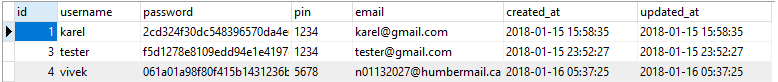
**Purpose: Every device is a unit, for each unit there is a separate record to hold its values**



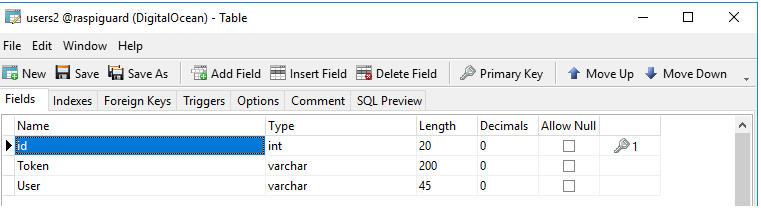
**Table users**



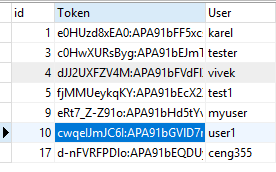
**Purpose: Authentication**



**Table users2**



**Purpose: Stores push notification tokens for users which are used to identify android devices. This enables us to send directed push notification messages.**



**Note that these values in the tables get inserted via our python script for the unit and android app.**

Example data entry **from python script** (code snippet, full code is available on github)

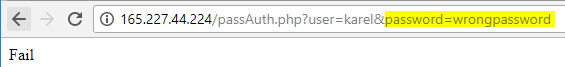
Pushing unit data into database

|  |
| --- |
| #Activity Log Entry |
|  | def ActivityLog(readings): |
|  |  |
|  | from datetime import datetime |
|  | curtime = datetime.now().strftime("%Y-%m-%d %H:%M:%S") |
|  |  |
|  | #Opening Connection |
|  | db = MySQLdb.connect(SQL\_Server, SQL\_User, SQL\_Pass, SQL\_Database) |
|  | cursor = db.cursor() |
|  |  |
|  | try: |
|  | cursor.execute("""INSERT INTO activitylog (datetime, username, sensorname, activity, moisturelevel, lightlevel) VALUES (%s, %s, %s, "door " %s, %s, %s) """, (curtime, SQL\_Username, SQL\_Unitname, readings[2].lower(), readings[1], readings[0])) |
|  | db.commit() |
|  |  |
|  | except: |
|  | db.rollback() |
|  |  |
|  | db.close() |
|  | return |

Example data entry **using the android-app**

Note: Android app **DOES NOT** directly communicate with MySQL server, instead the android app accesses a specific php file which in turn **generates json formatted outputs or plain text** which get parsed by the app.

**For example: Android app login system**



Source code snippet from passAuth.php file

if(isset($\_GET['user']) ){

$targetUser = $\_GET['user'];

$targetPass = $\_GET['password'];

$targetPass = md5($targetPass);

// Create connection

$conn = new mysqli($servername, $username, $password, $database );

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

$sql = "Select password FROM users WHERE username='". $targetUser."' ";

$result = mysqli\_query($conn, $sql);

if (mysqli\_num\_rows($result) > 0) {

while($row = mysqli\_fetch\_assoc($result)){

if ($targetPass == $row["password"]) {

echo "Success";

} else echo "Fail";

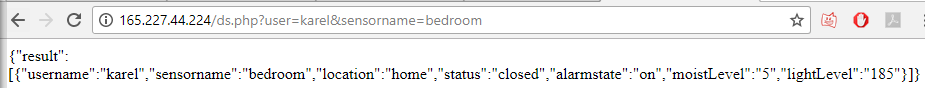
mysqli\_close($conn);

}

}

}

Here is another example of a json object output



The android app is able to parse the data and then we can display the values in the UI.

Example code snippet from DoorFragment.java code

**private void** getSensorData(String targetuser, String room) {  
 *//Creating a string request* StringRequest stringRequest = **new** StringRequest(Config.***DATA\_DOORSENSOR\_URL*** + **"?user="** + targetuser + **"&sensorname="** + room,  
  
  
 **new** Response.Listener<String>() {  
 @Override  
 **public void** onResponse(String response) {  
 JSONObject j = **null**;  
 **try** {  
 *//Parsing the fetched Json String to JSON Object* j = **new** JSONObject(response);  
  
 *//Storing the Array of JSON String to our JSON Array* **result2** = j.getJSONArray(Config.***JSON\_ARRAY***);  
  
 *//Calling method getsensors to get the sensors from the JSON Array* getSensors(**result2**);  
 } **catch** (JSONException e) {  
 e.printStackTrace();  
 }  
 }  
 },  
 **new** Response.ErrorListener() {  
 @Override  
 **public void** onErrorResponse(VolleyError error) {  
 }  
 });  
 *//Creating a request queue* RequestQueue requestQueue = Volley.*newRequestQueue*(getContext());  
  
 *//Adding request to the queue* requestQueue.add(stringRequest);  
}

**So to reiterate the android app uses plain text response (success/fail) for authentication and json object response for unit/sensor related data. These outputs are echoed out from specific php files.**

#### 2.5.6.2 PHP Config

File: config.php

<?php

define("DB\_HOST", "A.B.C.D");

define("DB\_USER", "username");

define("DB\_PASSWORD", "password");

define("DB\_NAME", "database\_schema");

?>

**Note, every php file that needs access to MySQL has to include config.php**

Eg: Opening a connection to MySQL

<?php

include\_once 'config.php';

class DbConnect{

private $connect;

public function \_\_construct(){

$this->connect = mysqli\_connect(DB\_HOST, DB\_USER, DB\_PASSWORD, DB\_NAME);

if (mysqli\_connect\_errno($this->connect))

{

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

}

public function getDb(){

return $this->connect;

}

}

Remember that there are no MySQL credentials in the java code because it DOES NOT directly communicate with MySQL, the php file has the credentials!

**public class** Config {  
 *//JSON URL* **public static final** String ***DATA\_DOORSENSOR\_URL*** = **"http://165.227.44.224/ds.php"**;  
 **public static final** String ***DATA\_ACTIVITYLOG\_URL*** = **"http://165.227.44.224/al.php"**;  
  
 *//Tags used in the JSON String* **public static final** String ***TAG\_USERNAME*** = **"username"**; *//raspiguard username*  
 **public static final** String ***TAG\_SENSORNAME*** = **"sensorname"**;   
 **public static final** String ***TAG\_LOCATION*** = **"location"**; *//eg bedroom,classroom*  
 **public static final** String ***TAG\_STATUS*** = **"status"**; *//eg door opened/closed*  
 **public static final** String ***TAG\_ALARMSTATE*** = **"alarmstate"**; *//eg alarm on/off*  
 **public static final** String ***TAG\_MOISTLEVEL*** = **"moistLevel"**; *//eg 50*  
 **public static final** String ***TAG\_LIGHTLEVEL*** = **"lightLevel"**; *//eg 150*  
 **public static final** String ***TAG\_DATETIME*** = **"datetime"**; *//timestamp*   
 **public static final** String ***TAG\_ACTIVITY*** = **"activity"**; *//eg Sensor Readings*  
  
 *//JSON array name* **public static final** String ***JSON\_ARRAY*** = **"result"**;  
}

The two URL constants defined here represent the base address, in their specific classes we are appending key value pairs to their tail to generate valid server requests.

### 2.5.7 Setting up PHP library for Apache

We can once again leverage the apt system to install our components. We're going to include some helper packages as well, so that PHP code can run under the Apache server and talk to our MySQL database:



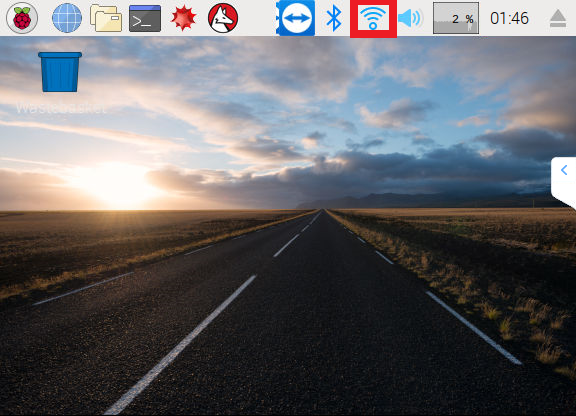
At this point you should have a working PHP web-server and a fresh installation of MySQL

### 2.5.8 Android Application Setup

### 2.5.9 Webpage Setup

### 2.5.8 Raspbian Setup

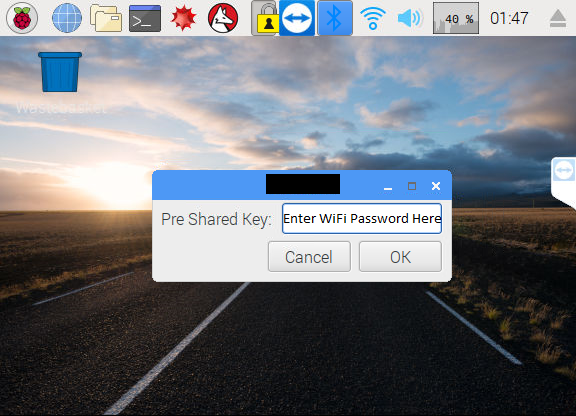
To setup the RaspiGuard unit plug a keyboard and mouse, then power on the unit.



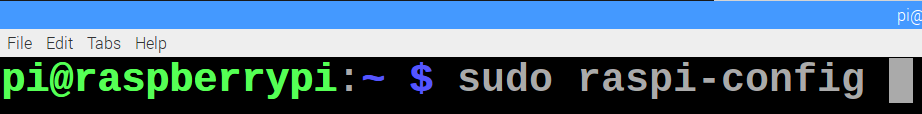
After Booting into the OS, setup the WiFi. If you are connected to the internet using Ethernet you may skip this step. Select the Wi-Fi button outlined in the image above.

[](https://raw.githubusercontent.com/V-Socrates/Is-It-Moist/master/Documentation/IMAGES-VIDEOS/Build%20Related/RPI2.png)

Select your connection from the dropdown list.



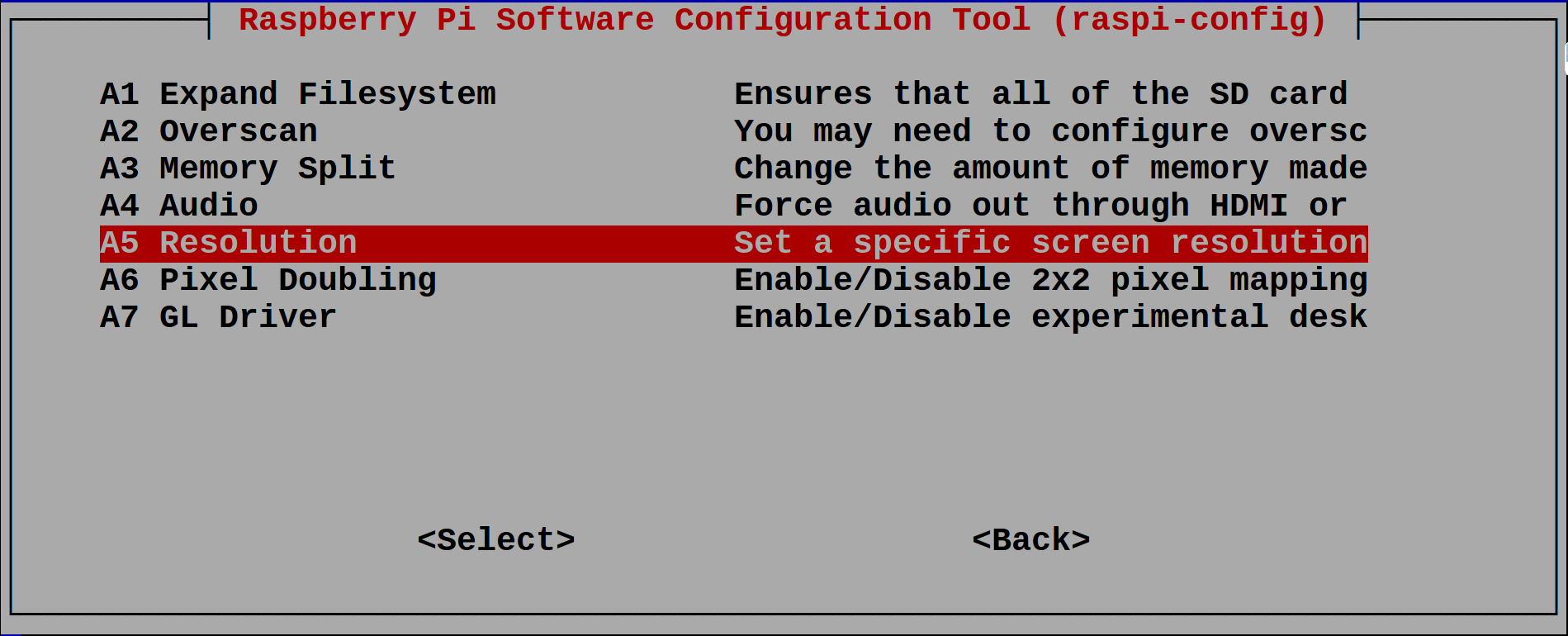
Enter you password and click OK.



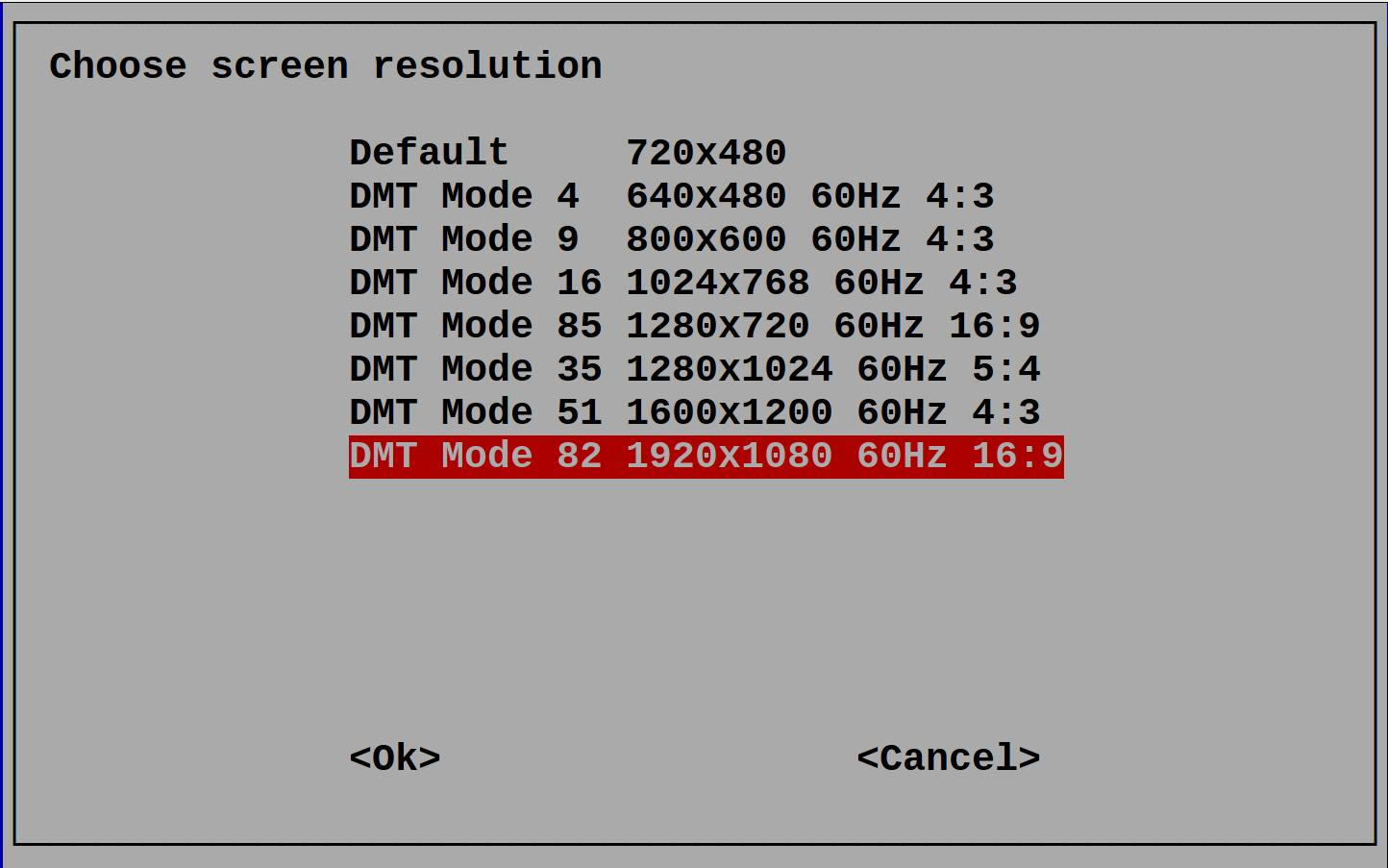
Open terminal and enter the command “sudo raspiconfig”.



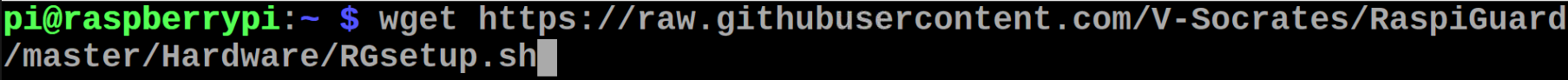
Select Advanced Options.



Select Resolution.



Select your desired resolution. The default for this project is “DMT Mode 82 1920x1080 60Hz 16:9”. After selecting your resolution, restart your unit.



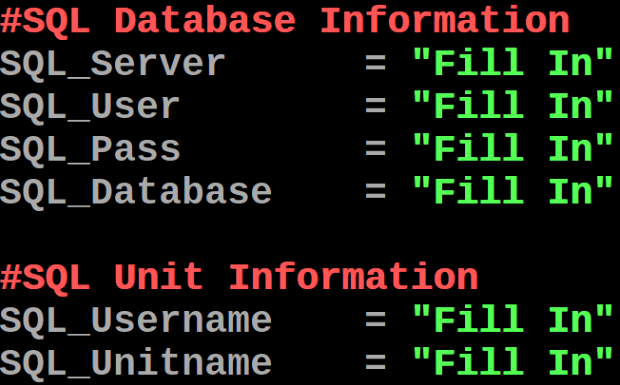
Enter “wget https://raw.githubusercontent.com/V-Socrates/RaspiGuard/master/Hardware/RGsetup.sh” in terminal to download the setup script.



After download has completed enter “sudo bash ./RGsetup.sh”. The script may take some time as it is getting everything to run the RG.py script. Follow any on screen instruction that may be displayed.



After the unit setup script has been completed enter “nano RG.py” to enter the SQL database and unit information.



Navigate to this part of the code (lines 23 - 31) and replace “Fill In” with your SQL database and unit information. Save the changes made to script and launch the script using the command “python RG.py”

If all steps are followed and the SQL database, PCB, Android application, and website are setup your unit is now complete.

## 2.6 Problems Encountered

### 2.6.1 PiTFT Resolution

**Problem:**

The PiTFT is set to 480p x 320p resoultion. The problem this presents is that the Raspbian GUI requires a larger resolution to be navigateable.

**Solution:**

The image provided by adafruit for the PiTFT is set to resolution 480p x 320 and cannot be changed. Problem solved by using the latest Raspbian image and manually installing PiTFT using adafruit's helper script.

### 2.6.2 PiTFT Orientation

**Problem:**

The PiTFT Rasbian image provided by adafruit is set to have the top of the screen on the side of the HDMI / Micro USB Power / AUX ports.

**Solution:**

Having converted to the latest Raspbian image the orientation of the PiTFT can be changed by editing "/boot/config.txt" and changing the dtoverlay option line.

### 2.6.3 MCP3008 Conflicts With PiTFT

**Problem:**

Raspberry Pi 3 SPI inputs are being used by PiTFT. Manually installing drivers using adafruit's helper script allows orientation selection. Currently set to 270 meaning that the top of the screen is on the side of the Raspberry Pi GPIO pins.

**Solution:**

New analog to digital converter is being used. The ads1115 uses the Raspberry Pi's I2C pins and does not conflict with the PiTFT.

### 2.6.4 Sensor Cables Too Short

**Problem:**

The door and moisture sensor must reach outside the case.

**Solution:**

Both door and moisture sensor cables striped and cut. New copper cables, twisted together for extra strength, soldered to sensors and the exposed copper and solder protected using heatshrink.

### 2.6.5 Light Sensor Malfunction

**Problem:**

Light resistor at times gave drastically incorrect results.

**Solution:**

Light resistor wires touching other wires changes values produced by the light. The problem was solved by using the copper wiring covers as jackets on the light resistor wires to avoid signal corruption.

## 2.7 Accessing Data

### 2.7.1 Android Phone Application

<TO BE FILLED>

### 2.7.2 Website

<TO BE FILLED>

# 3. Progress Reports

## 3.1 Is-It-Moist

### 3.1.1 Report 1

****Humber College

205 Humber College Blvd, Etobicoke, ON M9W 5L7

N01132027@humbermail.ca

**Progress Report**

**Project Summary**

|  |  |  |
| --- | --- | --- |
| **Report date** | **Project Name** | **Prepared By** |
| 1/8/2018 | Is it Moist | N01132027 |

**Status Summary**

“Is It Moist” sensor is currently in prototype form. Sensor is assembled, operational. Moisture is detected, and user is notified. Currently creating code for analog input. Testing and calibration continue.

**Project Overview**

|  |  |  |
| --- | --- | --- |
| **task** | **Progress** | **Notes** |
| Acquiring Components | 100% | All essential items received. |
| Component Testing | 100% | All required materials tested and functioning correctly. |
| Assembly | 100% | Product assembled and build video documented. |
| Function Testing | 75% | Product tested and functioning. |
| Prototyping | 100% | Prototype created and functioning. |
| Case Testing | 50% | Testing various use cases for product and collecting data. |
| Finalizing | 75% | Applying minor adjustments to hardware and software. |
| Documentation | 85% | Awaiting finalization to complete and publish documentation. |

**Budget Overview**

|  |  |  |  |
| --- | --- | --- | --- |
| **category** | **spent** | **Budget** | **notes** |
| Base Components | $ 119.23 | 59.62% | All items received on schedule. |
| Analog to Digital Components | $ 34.41 | 17.21% | Awaiting shipment. |

**CHALLENGES**

|  |  |
| --- | --- |
| **issue** | **plan / solution** |
| Calibrating Analog to Digital Sensor Signal | Potentiometer of Analog-Digital sensor needs further calibration to trigger at desired moisture level. |
| Products Software Status Response Time | Software currently checking for sensor status change too frequently. Measurement time needs to be configured for increased precision. |
| Reading Analog Signal | Currently creating code to read analog values of sensor |

**CONCLUSIONS / Subsequent plans**

“Is it Moist” prototype is currently operational and is in testing to calibrate hardware and software for increased precision. Awaiting components for integration of analog moisture values. Plans subsequent of finalizing project “Is it Moist” involve integration with project “RaspiGuard” which is targeted toward home/commercial security.

## 3.2 RaspiGuard

### 3.2.0 Report 0

Project status Report

Details

|  |  |  |
| --- | --- | --- |
| Report date | Project Name | Prepared By |
| 01/08/2018 | RaspiGuard | Karel Tutsu |

TASK Overview UP UNTIL THIS POINT

|  |  |
| --- | --- |
| task | notes |
| Research Topic & Proposal | Completed on the first week | |  |  |  |
| Schedule & Budget Planning | Completed by the fourth week | |  |  |  |
| Acquiring/Receiving Parts | Completed by the fifth week | |  |  |  |
| Blogging | Posted updates throughout the course | |  |  |  |
| Assembling Parts & Testing | Completed by the sixth week | |  |  |  |
| Video Script & Build Video | Completed by the ninth week | |  |  |  |
|  |  | |  |  |  |

Budget Overview

|  |  |  |  |
| --- | --- | --- | --- |
| category | spent | % of total | notes |
| Total Budget | $127.67 | 74% | The initial budget was $171.44 | |  |
|  |  |  |  | |  |

UPCOMING TASKS

|  |  |
| --- | --- |
| TASK | DUE DATE |
| Power Point Presentation | 01/15/2018 |  |
| Project Demonstration | 01/15/2018 |  |
| Build Instructions | 01/22/2018 |  |
|  |  |  |

PROJECT Summary

The project is up to date and I have completed all the requirements up until this point.

The blog has been on point.

### 3.2.1 Report 1

**Executive Summary**

Project RaspiGuard is currently on schedule and continuing with no significant difficulties. Project Is-It-Moist has been successfully integrated. The website design is complete and currently online. The hardware schematic design is completed and is currently being used to develop a custom PCB. Plans going forward include creating and testing PCB, Android application development, and hardware testing and scripting.

**Project Overview**

|  |  |  |
| --- | --- | --- |
| **Task** | **Progress** | **Developer** |
| Integrating Project Is-It-Moist | 100% | Vivek Socrates |
| Hardware Schematic | 100% | Vivek Socrates |
| PCB Schematic | 80% | Heakeme Williams |
| PCB Testing | 0% | Heakeme Williams |
| Website Design | 100% | Karel Tutsu |
| Website Testing | 70% | Karel Tutsu |
| Android Application Design | 50% | Karel Tutsu |
| Android Application Testing | 30% | Karel Tutsu |
| Hardware Scripting | 50% | Vivek Socrates |
| Hardware Script Testing | 50% | Vivek Socrates |
| MySQL Server Setup | 100% | Karel Tutsu |
| Case Design | 25% | Heakeme Williams |
| Case Assembly | 0% | Heakeme Williams |
| Prototype Device | 0% | Full Team |
| Prototype Device Testing | 0% | Full Team |
| Documentation | 20% | Full Team |

**Budget Overview**

|  |  |
| --- | --- |
| **Component** | **Cost** |
| Raspberry Pi 3 Model B | $48.98 |
| Raspberry Pi Power Adapter | $16.95 |
| ADATA Premier 8GB microSDHC UHS-I Class 10 | $8.99 |
| ADATA microReader Ver.3 microSDHC | $4.99 |
| adafruit PiTFT Plus 480x320 3.5" | $44.95 |
| adafruit ADS1115 | $14.95 |
| SparkFun Soil Moisture Sensor | $5.95 |
| Photo Cell (CdS Photoresistor) | $0.95 |
| Magnetic Contact Switchs | $3.95 |
| Piezo Buzzer - PS1240 | $1.50 |
| GPIO 2x13pin Ribbon Cable | $2.95 |
| Male To Male 2x13pin Header | $1.75 |
| Jumper Wire Cables | $1.95 |
| Raspbian Stretch with Desktop (OS) | $0.00 |
| **TAX** | **$20.65** |
| **TOTAL** | **$179.46** |

**Current Progress**

**Vivek Socrates**

Successfully integrated project Is-It-Moist into RaspiGuard. Light sensor, buzzer and LCD touchscreen display have been added to increase functionality. Hardware schematic has been created and is currently being tested. Currently writing and testing hardware scripts for accurate sensor readings.

**Karel Tutsu**

MySQL server setup and updated to support current hardware readings. Website template designed and created. Awaiting any functionality updates to modify website template. Currently testing website and working to adapt previously created android application to match current hardware specifications.

**Heakeme Williams**

After recently joining the team, is quickly becoming familiar with project details, and tasks. Undertaken the tasks of designing, creating and testing a custom PCB build based on hardware schematic. Currently working on the final stages of design and development for initial PCB prototype.

**Challenges, Problems and Troubleshooting**

**Vivek Socrates**

Initial analog to digital converter MCP3008 and adafruit PiTFT incompatible for simultaneous use. Issue resolved by changing converter to ADS1115.

Moisture sensor probe was tested and determined to be subpar. Probe has been replaced with superior product manufactured by SparkFun.

adafruit PiTFT having compatibility issues on Rasbian Stretch. Issue is currently being investigated, possible resolution may involve using older Rasbian operating system.

**Karel Tutsu**

MySQL database was initially designed to accommodate only two sensor readings. Tables and database have been updated to accommodate new hardware functionality.

Website template needed multiple versions due to hardware functionality being undetermined. Issue is resolved due to new definitive hardware specifications.

Previously written PHP scripts to access database are obsolete. New scripts currently in progress.

**Heakeme Williams**

Unfamiliarity with Fritzing software that was used in creating hardware schematic design. Issue being resolved by exploration of the software and online research.

Conversion of hardware schematic to creating PCB schematic using EAGLE software proved problematic due to default component libraries lacking necessary components. Resolved by searching for custom libraries online.

With updated functionality, PCB design needed to reflect new changes. Changes to PCB are currently in progress.

**Future Development**

The hardware is being assembled and tested using python scripts. PCB is nearing end of design phase and entering initial prototype phase. PCB will undergo extensive testing before final PCB is designed and produced. Website is being finalized and android app development due to begin.

Regards,

Vivek Socrates

### 3.2.2 Report 2

Dear Kristian,

Project RaspiGuard is still on schedule with no significant difficulties. Project Is-It-Moist has been successfully integrated. The website design has been completed to represent sensor data and is currently online. The hardware schematic design is completed and a PCB prototype has been printed.

**Integration Progress**

**Vivek Socrates**

**Recent Progress**

Successfully integrated project Is-It-Moist into RaspiGuard. Adafruit PiTFT having compatibility issues on Rasbian Stretch.

**Current Progress**

PiTFT touch screen was configured and working on latest Rasbian OS     Stretch. PiTFT was enabled to be now set to 1080p successfully.

**Upcoming Goal(s)**

The touchscreen is only calibrated to work 270 degrees, which poses as a problem for our project. Hence, we are working on calibrating the touchscreen to work regardless of its rotation.

**Karel Tutsu**

**Recent Progress**

Website template designed and created. Currently testing website and working to adapt previously created android application to match current hardware specifications.

**Current Progress**

Added login and session functionality to the website. This means that from now on users have to be logged in to access the full website.

Added account registration functionality for people without accounts.

Added change password/pin functionality for users.

**Upcoming Goal(s)**

Website data has to be specific to the user, thus I need to pull and display user specific data where as currently we are using placeholders with my account data (for everyone). Need to adjust data for visual graphs used on the website. Need to create interface to see the live status of the device. Need to create interface to toggle alarm state off/on.

**Heakeme Williams**

**Recent Progress**

There are a few updated functionalities to the initial schematic, hence, PCB design needs to reflect these new changes. Changes to PCB are currently in progress.

**Current Progress**

The changes to the schematic were made and the schematic was completed. The PCB prototype board was designed, routed and successfully printed.

**Upcoming Goal(s)**

While routing the board, because of the wires being a bit chaotic, some routes were made on top of the board, which poses a slight problem when soldering. Hence, we are working on getting all routes on the bottom.

**Conclusion**

We will be focusing on improving and executing the key deliveries while keeping our schedule and finance in mind. Currently, however, our finance is the same as before since no new purchases have been made.

Regards,

Heakeme Williams

### 3.2.3 Report 3

Dear Kristian, the following is our update for this week :

**Case**

The case hasn’t been designed yet, due to the fact that we are still taking measurements for the board. However, using the prototype PCB we were able to record the measurements, designs and orientation of the case as a predictive prototype test data.

**Hardware**

The Raspberry Pi is currently setup and is successfully reading all sensor components (Door, Light, and Moisture). The touchscreen is activated and displaying readings.

The touchscreen is calibrated to the orientation that is set and the issue is being worked on. The alarm component is currently connected and functioning however, the sound is not loud enough to currently be of any use. The alarm is put on low priority and focus is being placed on options that are more viable.

**PCB**

The prototype PCB is currently already soldered with all of the components and was tested with the raspberry pi. However, the test was unsuccessful and with further investigation, we came to a conclusion that there might have been a flaw with the routing in the schematic.

A new board is being designed for testing. This board will be an improvement on the prototype and will be able to plug in directly on the raspberry pi board.

**Sensor Readings**

The python script to read sensor data is currently operational. The script is setup to read and update the table on the MySQL database.

However currently no information is being entered to the activity log table in the database because we are in the process of determining the most efficient table update interval. We don’t want to spam the table with readings every few seconds because it’s not critical data and we are doing data analysis, so there more data there is the more time it takes to process. We are starting off with an interval of 1 update per minute but this is still to be determined.

**Website**

A demo user has been created for you to access the web UI. Username: **ceng355**, password: **ceng355**

<http://165.227.44.224/RaspiGuard/index.php> - this hyperlink is also available on our GitHub page.

**Android Application**

The android app just needs minor changes due to the additional light sensor that was added to the project this semester and the switch from binary output (moist/not moist) to raw analog readings.

**Financial**

We have not made any additional purchases since the last report, so we are on track with budget.

# 4. Conclusions

RaspiGuard is a monitoring system that is easy to setup and use. RaspiGuard offers multiple ways to track and monitor door and light activities, as well as the moisture level of plants. Users can access the information easily by navigating to our webpage or using our mobile android application. The alarm features on the mobile application keep user’s aware even while on go using notifications. The unit has a LCD touchscreen display showing live up-to-date readings with a built in alarm that can be set using the mobile application or webpage. A budget of $200.00 CAD was set which was not breached. The total cost to produce each unit is $---.-- CAD. This cost includes all components and shipping costs. RaspiGuard is a cost effective and simple solution to easily monitor any room.

# 5. Recommendations

## 5.1 Hardware

There are various hardware components that can be substituted to reduce cost of an individual unit.

1. Replace Raspberry Pi 3 Model B with Raspberry Pi Model B
2. Replace ads1115 with ads1015
3. Replace PiTFT with smaller, less expensive models

## 5.2 PCB

## 5.3 Software

# 6. Technical References

Williams, M. B. (2017, September 09). Raspberry Pi Security System with Motion Detection / Camera. Retrieved February 03, 2018, from: https://www.hackster.io/FutureSharks/raspberry-pi-security-system-with-motiondetection-camera-bed172

Institute of Electrical and Electronics Engineers. (2015, August 28). IEEE Xplore Digital Library [Online]. Available: https://ieeexplore.ieee.org/search/advsearch.jsp

Dechuan Chen and Meifang Wang, "A home security Zigbee network for remote monitoring application," 2006 IET International Conference on Wireless, Mobile and Multimedia Networks, hangzhou, China, 2006, pp. 1-4.

doi: 10.1049/cp:20061246 keywords: {MSP430F135 micro-controller;Zigbee;remote alarming;security system for house;wireles network},

URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5195407&isnumber=5195326

# 7. Appendices

## 7.1 RaspiGuard Unit

### 7.1.1 RGsetup.sh

#!/bin/bash

cd ~ && sudo apt-get -y update && sudo apt-get -y upgrade && wget https://raw.githubusercontent.com/adafruit/Adafruit-PiTFT-Helper/master/adafruit-pitft-helper2.sh && sudo apt-get install -y build-essential python-dev python-smbus python-pip python-mysqldb && git clone https://github.com/adafruit/Adafruit\_Python\_ADS1x15.git && sudo python setup.py install && cd ~ && sudo rm -r Adafruit\_Python\_ADS1x15 adafruit-pitft-helper2.sh && wget https://raw.githubusercontent.com/V-Socrates/RaspiGuard/master/Hardware/RG.py

### 7.1.2 RG.py

import Adafruit\_ADS1x15

import datetime

import MySQLdb

import RPi.GPIO as IO

import time

import webbrowser

##############################

# VARIABLES/SETTINGS #

##############################

#ADS1115 Settings

ADS = Adafruit\_ADS1x15.ADS1115()

GAIN = 1

#Sensor Data

formattedData = [0] \* 4

#Door Sensor

doorSensor = 4

IO.setmode(IO.BCM)

IO.setup(doorSensor, IO.IN, pull\_up\_down = IO.PUD\_UP)

#SQL Database Information

SQL\_Server = "Fill In"

SQL\_User = "Fill In"

SQL\_Pass = "Fill In"

SQL\_Database = "Fill In"

#SQL Unit Information

SQL\_Username = "Fill In"

SQL\_Unitname = "Fill In"

#####################

# FUNCTIONS #

#####################

#Update SQL Database

def UpdateSQLDatabase(readings):

#Opening Connection

db = MySQLdb.connect(SQL\_Server, SQL\_User, SQL\_Pass, SQL\_Database)

cursor = db.cursor()

update = "UPDATE units SET lightlevel = '%s', moisturelevel = '%s', doorstatus = '%s' WHERE username = '%s' AND unitname = '%s'" % (readings[0], readings[1], readings[2], SQL\_Username, SQL\_Unitname)

try:

cursor.execute(update)

db.commit()

except:

db.rollback()

db.close()

return

#Activity Log Entry

def ActivityLog(readings):

from datetime import datetime

curtime = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

#Opening Connection

db = MySQLdb.connect(SQL\_Server, SQL\_User, SQL\_Pass, SQL\_Database)

cursor = db.cursor()

try:

cursor.execute("""INSERT INTO activitylog (datetime, username, sensorname, activity, moisturelevel, lightlevel) VALUES (%s, %s, %s, "door " %s, %s, %s) """, (curtime, SQL\_Username, SQL\_Unitname, readings[2].lower(), readings[1], readings[0]))

db.commit()

except:

db.rollback()

db.close()

return

#Function To Print Sensor Data

def PrintSensorReadings():

#Variables To Store Data

rawData = [0] \* 2

#Reading Data

for i in range(2):

#Read the specified ADC channel using the previously set gain value.

rawData[i] = float(ADS.read\_adc(i, gain = GAIN))

GetAlarmStatus()

#Formatting Light Sensor Data

if i == 0:

formattedData[i] = abs(round(float((rawData[i] / 32767) \* 255), 0))

#Formatting Moisture Sensor Data

if i == 1:

formattedData[i] = abs(round(float((rawData[i] / 26500) \* 100), 0))

#Formatting Door Sensor Data

if IO.input(doorSensor) == True:

if formattedData[2] != "Opened":

formattedData[2] = "Opened"

ActivityLog(formattedData)

if IO.input(doorSensor) == False:

if formattedData[2] != "Closed":

formattedData[2] = "Closed"

ActivityLog(formattedData)

#Printing Formatted

print("\033[H\033[J")

print(" ")

print("\033[1;31;40m\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\033[1;31;40m\*\t\t\033[1;32;40m RaspiGuard \t\t\033[1;31;40m\*")

print("\033[1;31;40m\*---------------------------------------\*")

print("\033[1;31;40m\* Lux: \033[1;32;40m%7.2f\t\t\033[1;31;40mMoisture:\033[1;32;40m%04.2f%%\t\033[1;31;40m\*" % (formattedData[0], formattedData[1]))

print("\033[1;31;40m\* \*")

print("\033[1;31;40m\* \*")

#print("\033[1;31;40m\* \*")

print("\033[1;31;40m\* Door: \033[1;32;40m%s\t\t\033[1;31;40mAlarm:\t\033[1;32;40m %s\t\033[1;31;40m\*" % (formattedData[2], formattedData[3]))

print("\033[1;31;40m\* \*")

print("\033[1;31;40m\* \*")

#print("\033[1;31;40m\* \*")

print("\033[1;31;40m\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#Get Alarm State

def GetAlarmStatus():

#Opening Connection

db = MySQLdb.connect(SQL\_Server, SQL\_User, SQL\_Pass, SQL\_Database)

cursor = db.cursor()

alarm = "SELECT dooralarmstate FROM units WHERE username = '%s' AND unitname = '%s' LIMIT 1" % (SQL\_Username, SQL\_Unitname)

cursor.execute(alarm)

data = cursor.fetchone()

#Formatting Alarm Status Data

if data[0] == "on":

formattedData[3] = "On"

elif data[0] == "off":

formattedData[3] = "Off"

else:

formattedData[3] = data[3]

db.close()

return

################

# MAIN #

################

def main():

#Timer Variables

Update = 0

Log = 0

#Sensor Loop

while True:

#Getting And Printing Readings

PrintSensorReadings()

#Update SQL Database Every 3 Seconds

if Update == 6:

UpdateSQLDatabase(formattedData)

Update = 0

#log Sensor Details Every Minute

if Log == 30:

ActivityLog(formattedData)

Log = 0

#Updating Timers

Update += 1

Log += 1

time.sleep(0.5)

return

main()

## 7.2 SQL Database

<TO BE FILLED>

## 7.3 SQL Database PHP Scripts

### 7.3.1

To befilled

### 7.3.2

Tobefilled

### 7.3.3

Tobe filled

### 7.3.4

tobefilled

<TO BE FILLED>

## 7.4 Website Code

## 7.5 Android Phone Application

### 7.5.1