**A Project Report**

**on**

**SMART GREENIFY DEVICE**

**For the award of**

**ONTARIO GRADUATE CERTIFICATE**

Submitted by

**Amonjot Singh Chhina**

**Anna Joy**

**Manu Simriti Parbhakar**

**Vy Nguyen**

Under the guidance of

**Dr. Mike Aleshams**

**ESE-4009 Embedded System Design Project**

(May 2021 - Aug 2021)

**Abstract**

The agenda of the project is to make a smart device that can monitor and even control the elements needed for a plant without much exertion and time from the user. Totally user friendly as it can be accessed by the customer from anywhere through an application. No prerequisite knowledge of programming needed as a person of any age group can operate it.

Dealing with Embedded Systems and analyzing related technology made us think of this concept. Investigation on similar products lead to the conclusion that our project not only eliminates their drawback but also provides new possibilities. Our product runs on Raspberry pi 4 with Raspbian as our operating system. Our product runs a Linux kernel and Debian distro.

A thorough review was done after implementation by testing the capabilities and usability of the product. Eventually, a user’s guide was prepared to assist the early users.

**Acknowledgments**

We would like to express our heartfelt gratitude to everyone at Lambton College's Embedded Systems Design Engineering Class for allowing us to accomplish this project. Dr. Mike Aleshams, our project guide, deserves special thanks for his involvement and contribution in helping us plan our project and file the final project report.

We would also like to extend our appreciation to Mike Aleshams, Course Coordinator, Embedded Systems Design Engineering for providing necessary cooperation during troubleshooting times.

Lambton College in Toronto is a fantastic place to learn and work, thanks to the supportive professors. Thank you to all of my classmates who have supported and encouraged our proposal.

Amonjot Singh Chhina

Anna Joy

Manu Simriti Parbhakar

Vy Nguyen

Table of Contents

# List of Figures

|  |  |
| --- | --- |
| **Figure** | **Description** |
| **2.1** | **Pie Chart on Plant Care** |
| **2.2** | **The survey conducted by the groupmates throughout the class and peers** |
| **3.1** | **Block Diagram of Project development** |
| **4.1** | **Connection between Raspberry pi 4 and Si7021** |
| **4.2** | **Ultrasonic sensor HC-SR04 with voltage divider** |
| **4.3** | **Raspberry pi Camera V2 Module** |
| **4.4** | **LCD Communication** |
| **4.5** | **Connection of camera to Raspberry pi 4** |
| **4.6** | **LCD - MCP23008 - Raspberry Pi 4 Connection** |
| **4.7** | **Schematic Design** |
| **4.8** | **PCB design** |
| **8.1** | **Plant placed in the frame** |
| **8.2** | **Raspberry Pi setup** |
| **8.3** | **Putty software window** |
| **8.4** | **App installation and App UI.** |

# 

# Chapter I

# Introduction

**Overview.**

## **Concept Introduction**

### **A Study Case:**

More and more people are having a hobby of growing plants, herbs indoors, etc. What is the reason behind this trend? In this section, we will be exploring the benefits of interior plants.

Benefits ment, according to NASA research (NASA, n.d.)

* Humidity: One of the studies (Lohr, 2010, 2) states that foliage plants can increase the relative humidity to healthier and more comfortable levels in interior spaces
* Dust: Adding plants into the room will reduce dust by at least 20% (Lohr, 2010, 2). A study shows that when the plants were in self-watering containers that watered the plants from below, the growing medium surface was dehydrated and dusty. Documenting that interior plants were associated with reduced dust under such circumstances was especially important because it allayed that the ever-increasing medium in containers might make interiors dustier.

More importantly, studies showing that keeping plants indoors also provides some psychological benefits:

* Stress: Interior plants have been associated with reduced stress, increased pain tolerance, and improved people’s productivity. A study showing that when people are in a room with a few containerized interior plants, even when their attention is not drawn to the plants
* Feeling: In a study of people working on computer tasks, there were significant differences in the item "I feel attentive or concentrating" (Lohr et al., 1996). When foliage plants were in the room, people reported feeling more attentive than did people in the room without plants.

Productivity: Productivity is higher when plants are present—the computer task study mentioned above. People responded significantly more quickly when plants were in the room than when the plants were absent, and there was no increase in error rate associated with the faster response (Lohr, 2010, 3).of growing plants indoors:

* Air: Plants contribute to a cleaner, healthier air for us, hence giving us a better living environment

**Problem Statement**

However, these plants need attention and maintenance as well. They need to be monitored and taken care of closely, which requires the amount of time from the owner and knowledge about plants. On the other hand, people are busy with their activities: working, studying, relaxing, etc. therefore, this is one reason people may lose interest in taking care of their little garden. Another reason is: they get bored; many people give up their passion for their hobbies for various reasons (Heshmat, 2017, 1). People like to enjoy their hobbies, especially when they do not have to do much work to care for them. Some new plant owners don’t know how to care for a specific plant to keep them alive.

**Goal and Objectives**

**Goal**

To have a device that takes care of the plants by observing the vitals of soil and give water and lighting accordingly.

**Objective**

The device will have sensors to detect the water level, humidity, and temperature, and all these parameters can be controlled by the particular plant’s requirement. Grow lights will be used to give artificial sunlight to the plants. We will have a log of data of the different varieties of plant species and their requirements stored in the memory. All the data and controls can be accessed on a handheld device wirelessly over Bluetooth. We also will have a camera to monitor the Plant’s health.

**The Scope of the Project**

**Deliverable**

* Device that hosts the plants with sensors inside the pot
* Wi-Fi, Bluetooth, live video monitoring capabilities
* Water supply and lighting controlled by microcontroller

**Milestones**

|  |  |  |  |
| --- | --- | --- | --- |
| Task | Day start | Day End | Person in charge |
| Project Proposal | May 10 | June 3 | Group |
| Testing components | Jun 10 | Jun 14 | Manu |
| Setting up Raspberry Pi | Jun 14 | Jun 17 | Amonjot |
| Interfacing humidity and temperature sensor | Jun 17 | Jun 24 | Vy |
| Interfacing Water level sensor | Jun 17 | Jun 24 | Anna |
| Interfacing Moisture sensor | Jun 17 | Jun 24 | Manu |
| Interfacing Display | Jun 17 | Jun 24 | Vy |
| Interfacing USB camera | Jun 24 | Jun 30 | Anna |
| Interfacing Speaker | Jun 24 | Jun 30 | Amonjot |
| Interfacing with Video Streaming | Jun 30 | July 14 | Anna |
| Interfacing grow lights and relay board | Jun 30 | July 14 | Manu |
| Interfacing water pump and switch control | Jun 30 | July 14 | Amonjot |
| Google cloud interfacing | July 14 | Aug 5 | Anna |
| Smartphone interfacing | July 14 | Aug 5 | Vy |
| Schematic design for the whole system | Aug 5 | Aug 12 | Vy |
| PCB layout design | Aug 5 | Aug 12 | Manu |
| Integrating & real-time programming the whole system | Aug 12 | Aug 15 | Amonjot |
| Report writing | July 29 | Aug 15 | Group |
| Demonstration | Aug 16 | Aug 19 | Group |

**Limitations**

* Inbuilt soil pods are not present in the current product to grow plants
* The brightness of grow lights cannot be adjusted but can only switch off and switch on.
* Automatic disease detection is not introduced

**Outcomes and Benefits**

* The user will get a device that takes care of their plants
* A display screen that displays the humidity and moisture of the soil
* No input from the user is required once the device is set up

**Facilities and Resources**

· **Laboratory**

Multimeter and Linux installed Laptop.

· **Intellectual Resources**

Exploring Raspberry Pi from Derek Molloy.

**Procedure and Methodology**

1. Install latest Raspian image that can be obtained from raspberrypi.org on Raspberry pi 4
2. Attach moisture sensor and proximity sensor and obtain the readings
3. Attach water pump and lighting that will be controller according to the readings of the sensors
4. Connect display to the raspberry pi and display the vital readings on the display
5. Attach a camera so that will stream the live video to an IP address
6. Providing connectivity to the user smartphone through BLE

# 

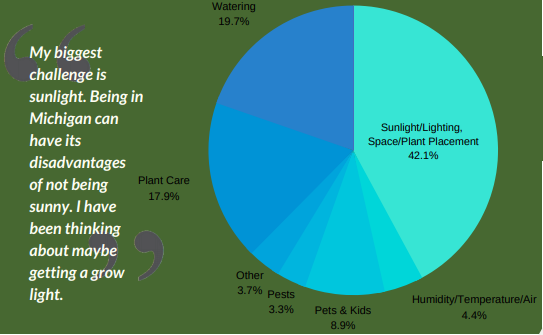
# Chapter II

# Literature Review

Plants contribute to a cleaner, healthier air for us, hence giving us a better living environment, according to NASA research (NASA, n.d.)

One of the studies (Lohr, 2010, 2) states that foliage plants can increase the relative humidity to healthier and more comfortable levels in interior spaces

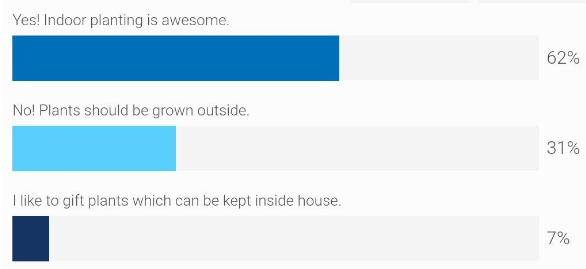
Adding plants into the room will reduce dust by at least 20% (Lohr, 2010, 2). A study shows that when the plants were in self-watering containers that watered the plants from below, the growing medium surface was dehydrated and dusty. Documenting that interior plants were associated with reduced dust under such circumstances was especially important because it allayed that the ever-increasing medium in containers might make interiors dustier. In a study of people working on computer tasks, there were significant differences in the item "I feel attentive or concentrating" (Lohr et al., 1996). When foliage plants were in the room, people reported feeling more attentive than did people in the room without plants. Productivity is higher when plants are present—the computer task study mentioned above. People responded significantly more quickly when plants were in the room than when the plants were absent, and there was no increase in error rate associated with the faster response (Lohr, 2010, 3). However, these plants need attention and maintenance as well. They need to be monitored and taken care of closely, which requires the amount of time from the owner and knowledge about plants. On the other hand, people are busy with their activities: working, studying, relaxing, etc. therefore, this is one reason people may lose interest in taking care of their little garden. Another reason is: they get bored; many people give up their passion for their hobbies for various reasons (Heshmat, 2017, 1). People like to enjoy their hobbies, especially when they do not have to do much work to care for them. Some new plant owners don’t know how to care for a specific plant to keep them alive.



*Figure2.1: Pie Chart on Plant Care*

**Survey Results:**

We surveyed to find out whether people like to grow plants indoors and the results were as follows:



*Figure2.2: The survey conducted by the groupmates throughout the class and peers*

From the above survey results, it is evident that many people like to grow plants inside. But people find it difficult to keep their plants alive inside the house. So, we need a solution to help people keep their plants healthy inside.

# Chapter III

# Requirements / Analysis

**Hardware**

* Raspberry pi 4
* Peristaltic Liquid Pump
* Plastic Water Solenoid
* Tubing PVC 8MM
* Humidity Sensor
* Capacitive moisture sensor
* Ultrasonic sensor
* LED light strip
* LCD display
* 12V Adapter
* 5V Adapter
* USB type C

**Software**

* Eclipse IDE for C/C++ programing
* Raspbian for Raspberry pi
* Putty on the host machine
* EasyEDA
* Latest raspbian image

**Power Requirement**

There are three main power-consuming components. Their power requirements are as follows:

1. Raspberry Pi 4 –15W at 5V

2. LED Strip Maximum 12V @ 60mA draw per strip segment

3. Water pump 5V @ 500mA

**Block Diagram**

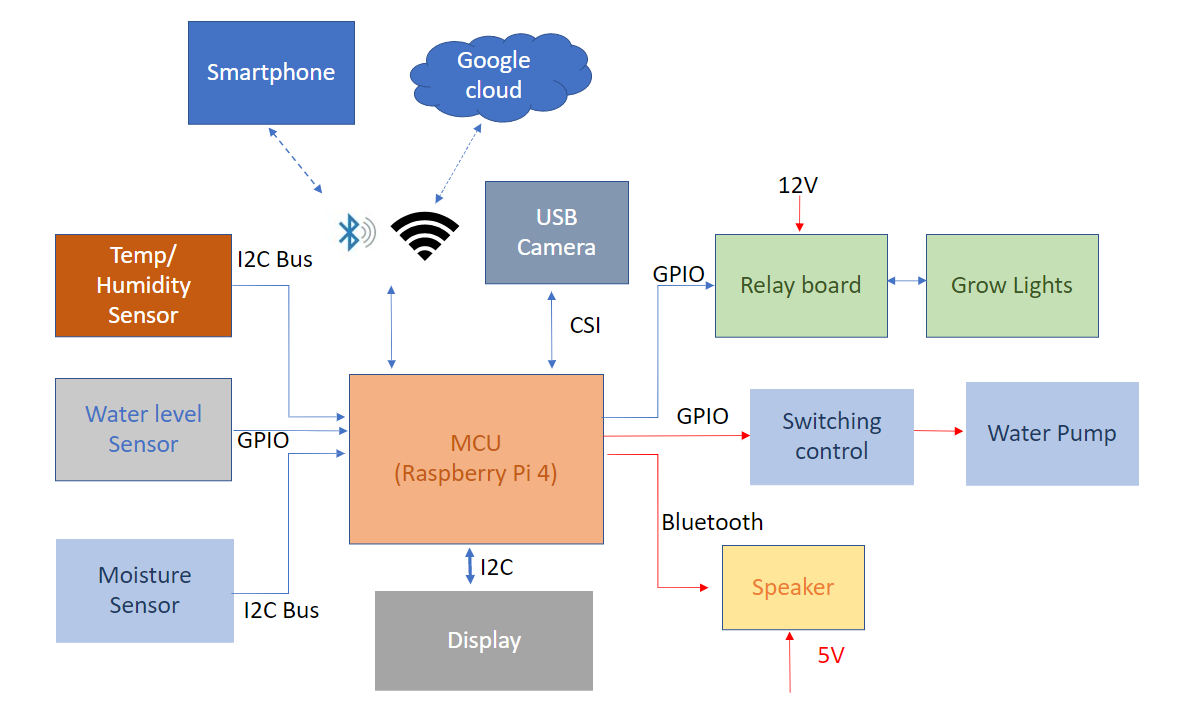


Figure 3.1 *Block Diagram of Product Development*

The above figure shows the flow of product development. All testing of the device tree was performed under Raspbian OS provided by Raspberrypi.org for Raspberry Pi 4.

# 

# Chapter IV

# Design

**Humidity and Temperature Sensor: Si7021**

Si7021 can be connected easily to Raspberry pi 4.The connection is shown in the following diagram.

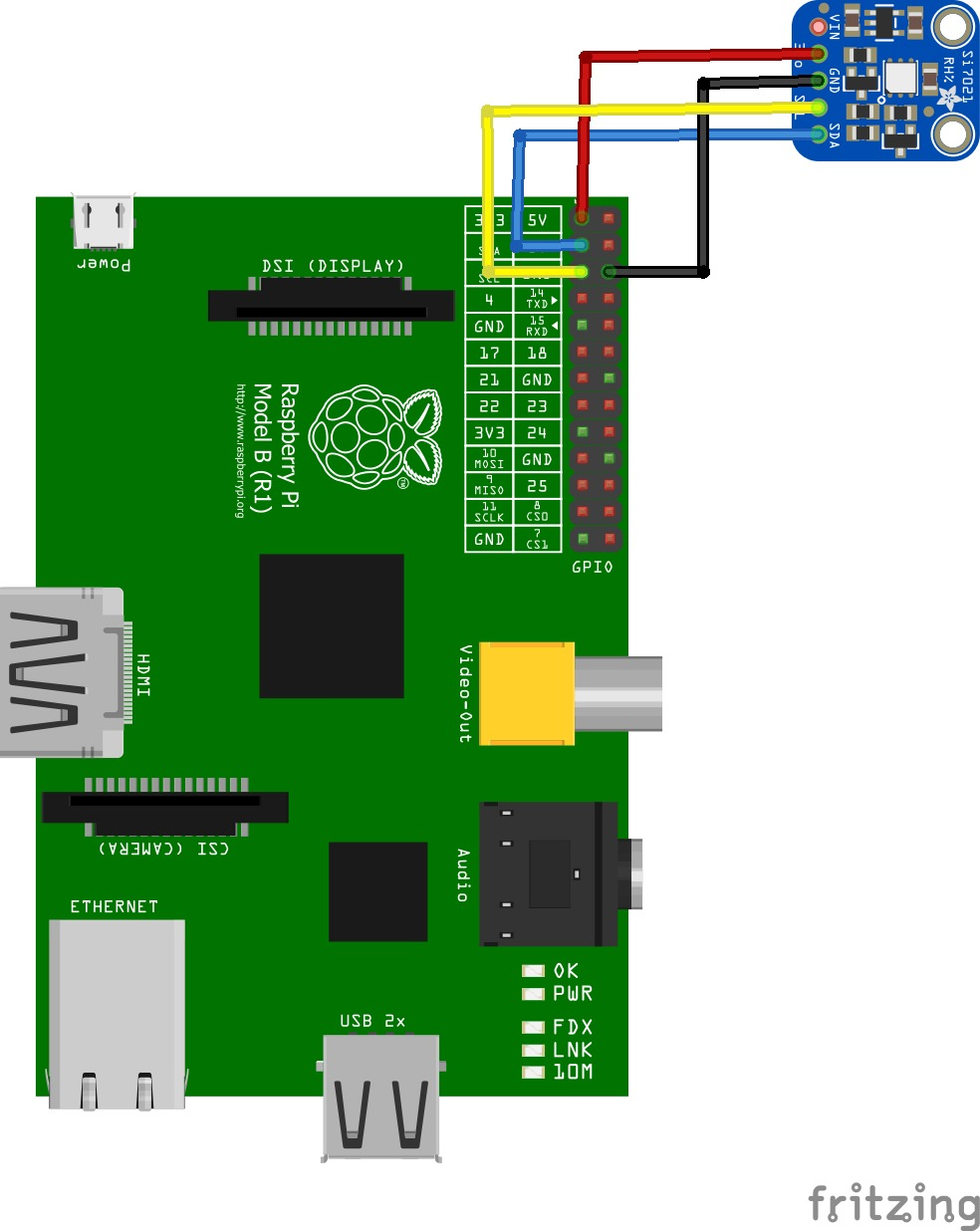


Figure 4.1 *Connection between Raspberry pi 4 and Si7021*

As per the diagram, the connections are from GND to the negative pin of Si7021, GPIO 2 Serial Data (I2C) to the DA pin of the sensor and GPIO 3 Serial Clock (I2C) to the CLK pin of the sensor.

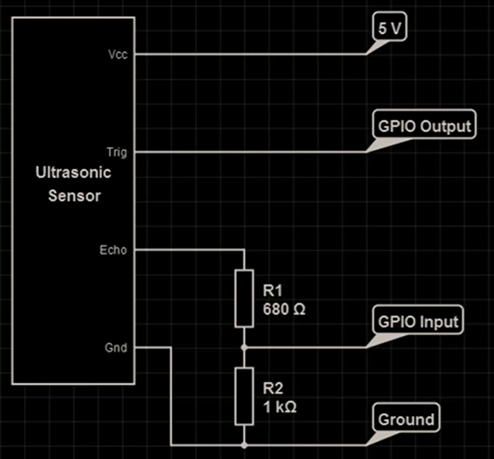
**Raspberry Pi 4 and Ultrasonic Sensor (water level detection)**

In this project, we are using Ultrasonic Sensor HC-SR04 as the main sensor to detect the water level of the tank.

Because the sensor is working at 5V, hence we need to lower this signal before it goes to the Raspberry Pi 4 (GPIO pins only work with 3.3V) to protect the MCU.

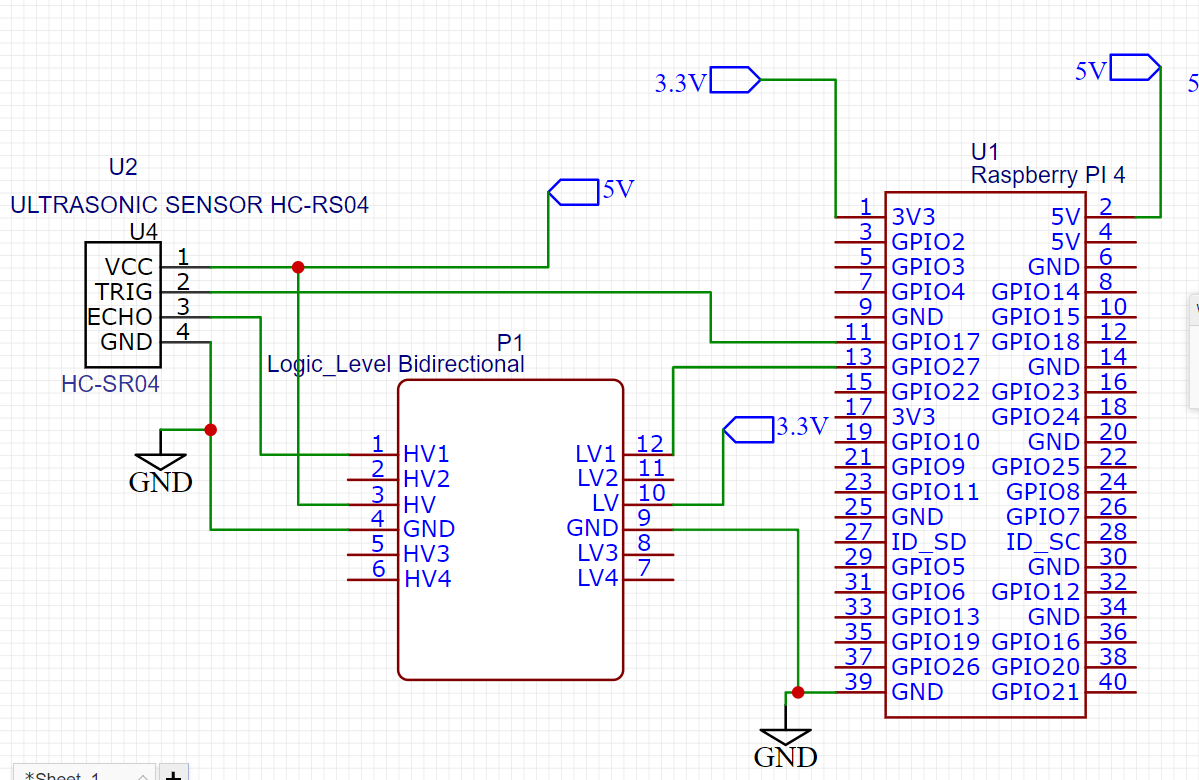
There are two ways to lower the input signals coming from the HC-SR04. Using voltage divider, or a logic level bidirectional converter.

* Circuit with voltage divider:



*Figure 4.2: Ultrasonic sensor HC-SR04 with voltage divider*

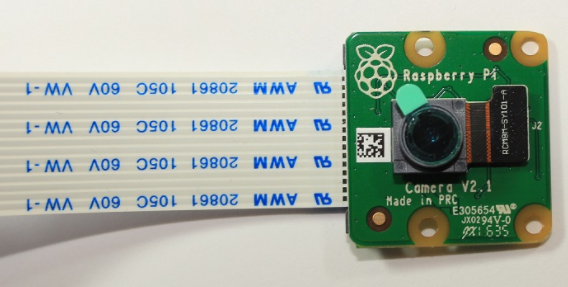
* Circuit with logic bidirectional converter:



*Figure 4.3: Ultrasonic sensor HC-SR04 with Logic level Bidirectional converter.*

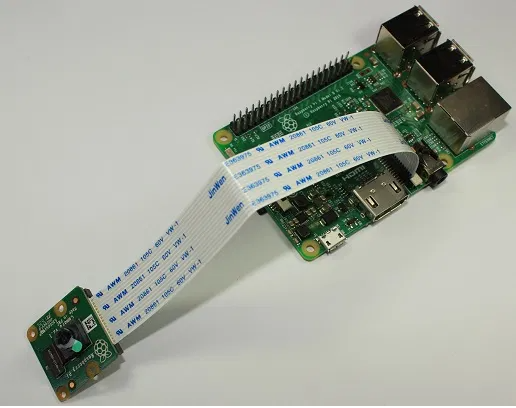
**Raspberry Pi 4 Camera V2 Module**

The camera used in this is Raspberry pi V2 Module camera which can be directly connected to Raspberry pi 4.



*Figure 4.3:Raspberry pi Camera V2 Module*

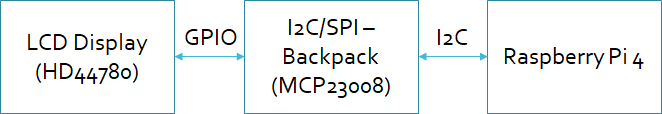
The connection should be done with the Raspberry pi in shutdown.Connect the camera to the Pi CSI port.Make sure the camera is connected in the right orientation with the ribbon blue letters facing up as shown in the next figure.



*Figure 4.4: Connection of camera to Raspberry pi 4*

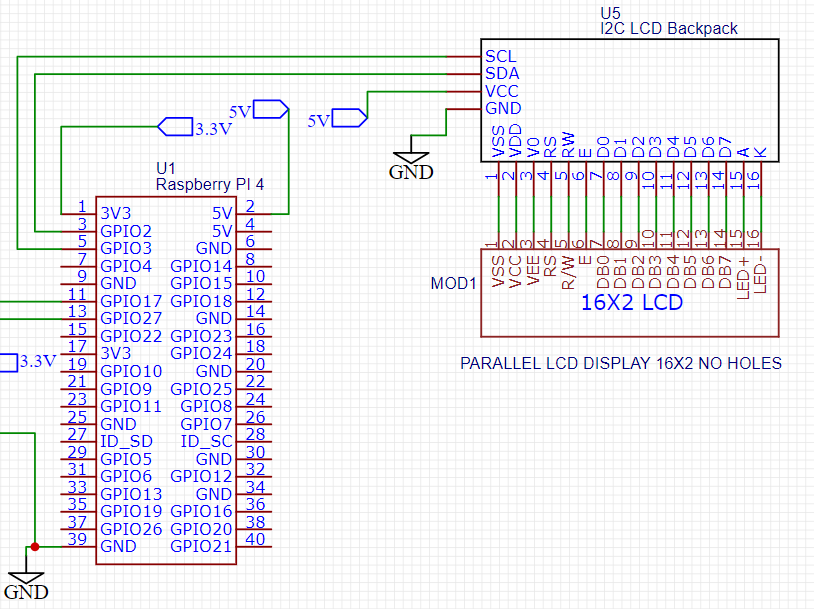
**Raspberry Pi 4 and LCD.**

LCD 16x2 with HD44780 chip is used in this project. However, to reduce the GPIO pins connecting directly to the Raspberry Pi 4. We are using the I2C/SPI Backpack module (with MCP23008 chip) as the bridge communication, the figure below illustrates how this works.



*Figure 4.5 LCD communication.*

The circuit is shown in the figure below:

****

*Figure 4.6 LCD - MCP23008 - Raspberry Pi 4 Connection*

**Raspberry Pi 4 and SmartPhone control via BLE.**

Raspberry Pi 4 has Bluetooth 5.0 and Bluetooth Low Energy technology. From this, we can make the Raspberry pi work as a client (end device), and use a smartphone as a server to control the GPIO and read sensor data.

Using the Blynk App, we can design an application for our own project.

**Project Schematic:**

Schematic design of our project.

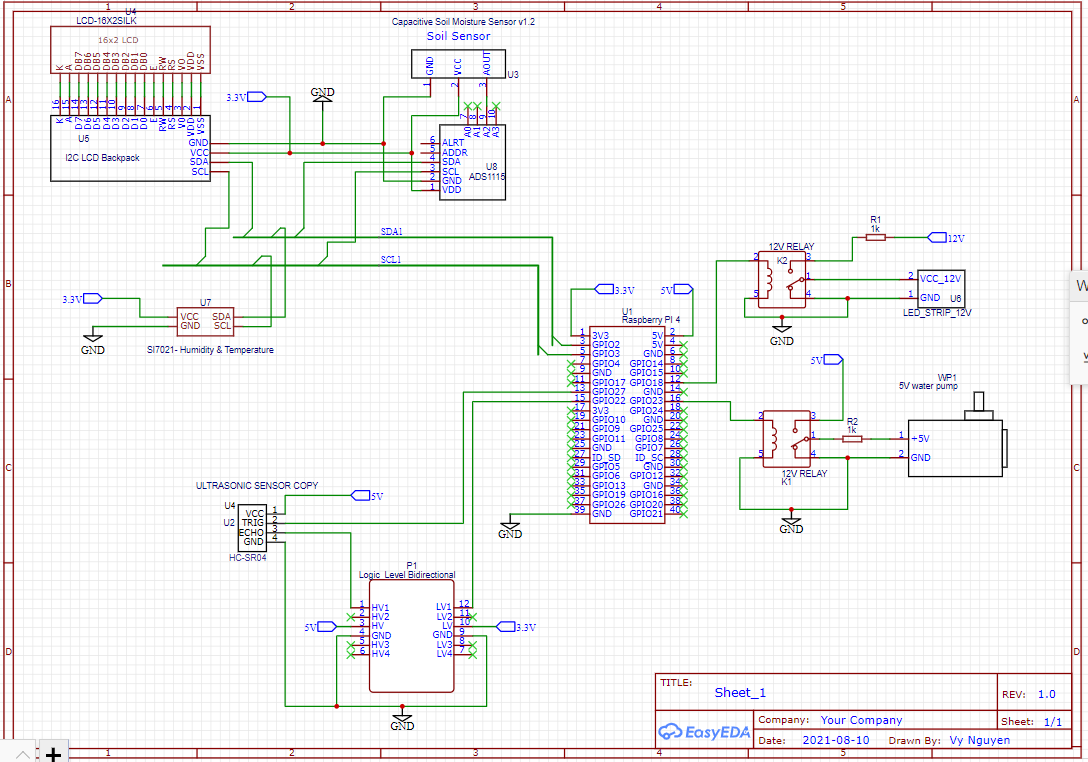
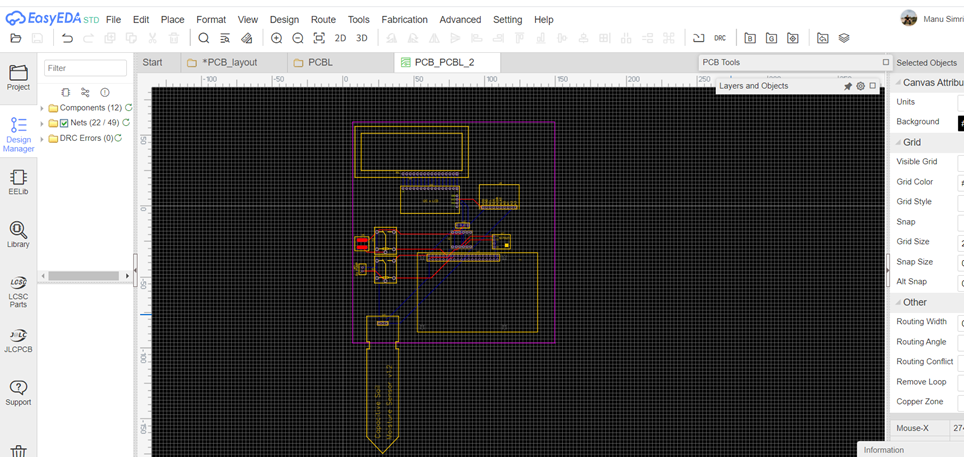
****

Figure 4.7 Schematic Design.

**PCB Design:**

The PCB can be realised after designing the schematics for this project EasyEDA. Below is the design of PCB:

****

*Figure 4.8 PCB Design.*

**Software Selection:**

Geany IDE is a powerful, stable and lightweight programmer's text editor that provides tons of useful features without bogging down your workflow. It runs on Linux, Windows and MacOS, is translated into over 40 languages, and has built-in support for more than 50 programming languages.

By using Geany IDE, we can directly program and compile the code for our project.

Other Software we are using to design Schematic and PCB is EasyEDA.

EasyEDA is a web-based EDA tool suite that enables hardware engineers to design, simulate, share - publicly and privately - and discuss schematics, simulations and printed circuit boards. Other features include the creation of a bill of materials, Gerber files and pick and place files and documentary outputs in PDF, PNG and SVG formats.

EasyEDA allows the creation and editing of schematic diagrams, SPICE simulation of mixed analogue and digital circuits and the creation and editing of printed circuit board layouts and, optionally, the manufacture of printed circuit boards.

# Chapter V

# Implementation and Test

**Raspberry Pi 4 and Si7021(Temperature and Humidity Sensor).**

|  |
| --- |
| // Distributed with a free-will license. // Use it any way you want, profit or free, provided it fits in the licenses of its associated works. // SI7021 // This code is designed to work with the SI7021\_I2CS I2C Mini Module available from ControlEverything.com. // https://www.controleverything.com/content/Humidity?sku=SI7021\_I2CS#tabs-0-product\_tabset-2  #include <stdio.h> #include <stdlib.h> #include <unistd.h> #include <linux/i2c-dev.h> #include <sys/ioctl.h> #include <fcntl.h>  void main() {  // Create I2C bus  int file;  char \*bus = "/dev/i2c-1";  if((file = open(bus, O\_RDWR)) < 0)  {  printf("Failed to open the bus. \n");  exit(1);  }  // Get I2C device, SI7021 I2C address is 0x40(64)  ioctl(file, I2C\_SLAVE, 0x40);   // Send humidity measurement command(0xF5)  char config[1] = {0xF5};  write(file, config, 1);  sleep(1);  // Read 2 bytes of humidity data  // humidity msb, humidity lsb  char data[2] = {0};  if(read(file, data, 2) != 2)  {  printf("Error : Input/output Error \n");  }  else  {  // Convert the data  float humidity = (((data[0] \* 256 + data[1]) \* 125.0) / 65536.0) - 6;   // Output data to screen  printf("Relative Humidity : %.2f RH \n", humidity);  }   // Send temperature measurement command(0xF3)  config[0] = 0xF3;  write(file, config, 1);  sleep(1);   // Read 2 bytes of temperature data  // temp msb, temp lsb  if(read(file, data, 2) != 2)  {  printf("Error : Input/output Error \n");  }  Else  {  // Convert the data  float cTemp = (((data[0] \* 256 + data[1]) \* 175.72) / 65536.0) - 46.85;  float fTemp = cTemp \* 1.8 + 32;   // Output data to screen  printf("Temperature in Celsius : %.2f C \n", cTemp);  printf("Temperature in Fahrenheit : %.2f F \n", fTemp);  } } |

**Raspberry Pi 4 and Moisture Sensor.**

The moisture is connected to arduino uno through jumper wires and then the coding is done in Arduino IDE where the values of water and air have been collected and by doing the calculations the percentage of water is calculated.

**Code for Arduino**

|  |
| --- |
| **const int AirValue = 620;  const int WaterValue = 310;  int soilMoistureValue = 0; int soilmoisturepercent=0; void setup() {  Serial.begin(9600); // open serial port, set the baud rate to 9600 bps } void loop() { soilMoistureValue = analogRead(A0); //put Sensor insert into soil Serial.println(soilMoistureValue); soilmoisturepercent = map(soilMoistureValue, AirValue, WaterValue, 0, 100); if(soilmoisturepercent >= 100) {  Serial.println("100 %"); } else if(soilmoisturepercent <=0) {  Serial.println("0 %"); } else if(soilmoisturepercent >0 && soilmoisturepercent < 100) {  Serial.print(soilmoisturepercent);  Serial.println("%");   }  delay(2500); }** |

After uploading the code, the Arduino is connected to Raspberry Pi through USB and the baud rate is set as the same as the Arduino which is 9600. The commands are used in python to connect Raspberry Pi4 with Arduino. When the code is run, the output of Arduino is shown in Raspberry Pi.

|  |
| --- |
| #!/usr/bin/env python3 import serial if \_\_name\_\_ == '\_\_main\_\_': ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1) ser.flush() while True: if ser.in\_waiting > 0: line = ser.readline().decode('utf-8').rstrip() print(line) |

**Raspberry Pi 4 and Ultrasonic Sensor (Water level detection).**

The code interfacing between raspberry pi and the ultrasonic sensor is developed like this. We are using the wiring PI library to read and write digital signals.

|  |
| --- |
| #include <stdio.h> #include <stdlib.h> #include <wiringPi.h>   #define TRUE (1==1)  // HC-SR04 ultrasonic sensor on Raspberry pi 4 #define TRIG 4 #define ECHO 5  static volatile long startTimeUsec; static volatile long endTimeUsec; double speedOfSoundMetersPerSecond = 340.29;  void recordPulseLength() {  startTimeUsec = micros();  while (digitalRead(ECHO) == HIGH);  endTimeUsec = micros(); }  void setupUltrasonic() {  wiringPiSetup();  pinMode(TRIG, OUTPUT);  pinMode(ECHO, INPUT);   // TRIG pin must start LOW  // Initialize the sensor's trigger pin to low. If we don't pause  // after setting it to low, sometimes the sensor doesn't work right.  digitalWrite(TRIG, LOW);  delay(500); // .5 seconds }  int getCM() {  // Send trig pulse  // Triggering the sensor for 10 microseconds will cause it to send out  // 8 ultrasonic (40Khz) bursts and listen for the echos.  digitalWrite(TRIG, HIGH);  delayMicroseconds(10);  digitalWrite(TRIG, LOW);   int now = micros();  // Wait for echo start  // The sensor will raise the echo pin high for the length of time that it took  // the ultrasonic bursts to travel round trip.  while (digitalRead(ECHO) == LOW && micros() - now < 30000);  recordPulseLength();   long travelTimeUsec = endTimeUsec - startTimeUsec;  double distanceMeters = 100 \* ((travelTimeUsec / 1000000.0) \* 340.29) / 2;   return distanceMeters ; }  int main(void) {  setupUltrasonic();   while (1) {  if (getCM() <= 3)  {  printf("Tank is full\n");  printf("Distance: %dcm\n", getCM());  delay(1000); // 0.5 second  }  else if (getCM() > 10)  {  printf("Tank is empty\n");  printf("Distance: %dcm\n", getCM());  delay(1000); // 0.5 second  }  else  {  printf("Safe zone, Distance: %dcm\n", getCM());  delay(1000);  }  }  return 0; } |

**Raspberry Pi 4 and LED.**

In the code, the GPIO of Raspberry Pi used is 17 and the LED strip is connected through a relay. The code is written in C through the help of wiring pi. The pin mode is defined for output.

|  |
| --- |
| #include <wiringPi.h>  int main(void) // main function {  // LED strip : Physical pin 11, BCM GPIO17, and WiringPi pin 0.  const int ledstrip = 0;   wiringPiSetup(); pinMode(ledstrip, OUTPUT); //setting the pin to be output   digitalWrite(ledstrip, low); // setting the output to be HIGH   return 0; } |

**Raspberry Pi 4 and LCD**

LCD interfacing is programmed with the following code, In this code, we are using wiringPI library with the LCD, and mcp23008 libraries to convert the LCD communication to I2C.

|  |
| --- |
| #include <stdio.h> #include <stdlib.h> #include <wiringPi.h> #include <mcp23008.h> #include <lcd.h>  //Defines for LCD #define AF\_BASE 100 #define AF\_E (AF\_BASE + 2) //Enable pin  //#define AF\_RW (AF\_BASE + 5) //Read/ write  //#define AF\_RW (AF\_BASE + 14) //Read/ write  #define AF\_RS (AF\_BASE + 1) //Resister selec  #define AF\_DB4 (AF\_BASE + 3) //Data pin 4 #define AF\_DB5 (AF\_BASE + 4) //Data pin 5 #define AF\_DB6 (AF\_BASE + 5) //Data pin 6 #define AF\_DB7 (AF\_BASE + 6) //Data pin 7  //static volatile  int lcd;  //Initialise  void lcd\_setup()  {  wiringPiSetup();    mcp23008Setup(AF\_BASE, 0x20);  //lcd\_init();  lcd = lcdInit (2, 16, 4, AF\_RS, AF\_E, AF\_DB4, AF\_DB5, AF\_DB6, AF\_DB7, 0, 0, 0, 0);  lcdClear(lcd);  pinMode(AF\_BASE + 7, OUTPUT);  digitalWrite(AF\_BASE + 7, HIGH);   while(1)  {  lcdPosition(lcd,0,0);  lcdPuts(lcd,"Hello there");  lcdPosition(lcd,0,1);  lcdPuts(lcd,"How are you");  delay(2000);  lcdClear(lcd);  lcdPosition(lcd,0,0);  lcdPuts(lcd,"Bye Bye");  delay(2000);  lcdClear(lcd);  } }   int main(void) {  //wiringPiSetup():  lcd\_setup();  //lcd\_input(); } |

**Raspberry Pi 4 and Smartphone control**

The Blynk code for the application is based on the code below. This code introduces how to interfacing with a button, using a GPIO pin and a virtual pin.

Thanks to this example, we can develop our application:

|  |
| --- |
| // Blynk "gp" numbers are BCM numbers, so gp17 is physical pin 11  // #define BLYNK\_DEBUG #define BLYNK\_PRINT stdout #ifdef RASPBERRY  #include <BlynkApiWiringPi.h> #else  #include <BlynkApiLinux.h> #endif #include <BlynkSocket.h> #include <BlynkOptionsParser.h>  static BlynkTransportSocket \_blynkTransport; BlynkSocket Blynk(\_blynkTransport);  #include <BlynkWidgets.h>  unsigned int uptime; // 1 second intervals unsigned int pinStatus; // status of BCM 17 unsigned int lastpinStatus = 0; // to toggle  void myTimerEvent() // button widget on V0 or direct access gp17 button {  uptime = (millis() / 1000);  Blynk.virtualWrite(V1, uptime);  pinStatus = digitalRead(17);  if(pinStatus != lastpinStatus){  lastpinStatus = pinStatus;  printf("GP17 pin status: %i\n", pinStatus);  if(pinStatus == 1){ // this is to synchronise V1 button if gp17 button is pressed  Blynk.virtualWrite(V0, 1);  }  else{  Blynk.virtualWrite(V0, 0);  }  } }  void setup() {  //nothing to go here yet }  BLYNK\_WRITE(V0) // button set at PUSH frequency {  if(param[0] == 1){  printf("V1 turned device ON\n");  digitalWrite (17, HIGH) ;   }  else{  printf("V1 turned device OFF\n");  digitalWrite (17, LOW) ;  } }  void loop() {  Blynk.run();  if(millis() >= uptime + 1){ // 1 second intervals  myTimerEvent();  } }  int main(int argc, char\* argv[]) {  const char \*auth, \*serv;  uint16\_t port;  parse\_options(argc, argv, auth, serv, port);  Blynk.begin(auth, serv, port);  while(true) {  loop();  }  return 0; } |

**Raspberry Pi 4 and Video Streaming**

|  |
| --- |
| # Web streaming example # Source code from the official PiCamera package # http://picamera.readthedocs.io/en/latest/recipes2.html#web-streaming  import io import picamera import logging import socketserver from threading import Condition from http import server  PAGE="""\ <html> <head> <title>Raspberry Pi - Surveillance Camera</title> <style> h1 {  -webkit-text-stroke: 1px black;  margin: 2px; } </style> </head> <body style="background-image: url('https://www.hortibiz.com/fileadmin/\_processed\_/3/1/csm\_Top\_25\_vertical\_farming\_companies\_cca6015a56.jpg');"> <center><h1 style="color:red; border:2px outset silver;">Raspberry Pi - Surveillance Camera</h1></center> <center> <img style="border:10px outset silver;" src="stream.mjpg" width="800" height="560" /> </center> </body> </html> """  class StreamingOutput(object):  def \_init\_(self):  self.frame = None  self.buffer = io.BytesIO()  self.condition = Condition()   def write(self, buf):  if buf.startswith(b'\xff\xd8'):  # New frame, copy the existing buffer's content and notify all  # clients it's available  self.buffer.truncate()  with self.condition:  self.frame = self.buffer.getvalue()  self.condition.notify\_all()  self.buffer.seek(0)  return self.buffer.write(buf)  class StreamingHandler(server.BaseHTTPRequestHandler):  def do\_GET(self):  if self.path == '/':  self.send\_response(301)  self.send\_header('Location', '/index.html')  self.end\_headers()  elif self.path == '/index.html':  content = PAGE.encode('utf-8')  self.send\_response(200)  self.send\_header('Content-Type', 'text/html')  self.send\_header('Content-Length', len(content))  self.end\_headers()  self.wfile.write(content)  elif self.path == '/stream.mjpg':  self.send\_response(200)  self.send\_header('Age', 0)  self.send\_header('Cache-Control', 'no-cache, private')  self.send\_header('Pragma', 'no-cache')  self.send\_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')  self.end\_headers()  try:  while True:  with output.condition:  output.condition.wait()  frame = output.frame  self.wfile.write(b'--FRAME\r\n')  self.send\_header('Content-Type', 'image/jpeg')  self.send\_header('Content-Length', len(frame))  self.end\_headers()  self.wfile.write(frame)  self.wfile.write(b'\r\n')  except Exception as e:  logging.warning(  'Removed streaming client %s: %s',  self.client\_address, str(e))  else:  self.send\_error(404)  self.end\_headers()  class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):  allow\_reuse\_address = True  daemon\_threads = True  with picamera.PiCamera(resolution='640x480', framerate=24) as camera:  output = StreamingOutput()  #Uncomment the next line to change your Pi's Camera rotation (in degrees)  #camera.rotation = 90  camera.start\_recording(output, format='mjpeg')  try:  address = ('', 8000)  server = StreamingServer(address, StreamingHandler)  print("Stream Started")  server.serve\_forever()  finally:  camera.stop\_recording() |

**Raspberry pi with Google cloud**

|  |
| --- |
| import gspread from google.oauth2.service\_account import Credentials import blynklib import time import threading  import os os.chdir("/home/pi/")  while True:  try:   BLYNK\_AUTH = 'aHVFrTW05xL7iERk6myGWbm4E2wugjQj' #insert your Auth Token here   scope = ['https://spreadsheets.google.com/feeds','https://www.googleapis.com/auth/drive']  creds = Credentials.from\_service\_account\_file("client\_secret.json", scopes=scope)  client = gspread.authorize(creds)  gs = client.open("test\_sheet")  sheet = gs.worksheets()  path\_to\_file = "/home/pi/data\_log.txt"  file = open(path\_to\_file,'rb')   #Uncomment the line below to share the Spreadsheet with a mail  #gs.share('chhinajotub3@gmail.com', perm\_type='user', role='reader')      blynk = blynklib.Blynk(BLYNK\_AUTH)  break   except Exception as e:  f = open("test.txt", 'a')  f.write("\n 1."+str(e)+"\n")  f.close()  time.sleep(5)  # register handler for virtual pin V11 reading @blynk.handle\_event('write V11') def read\_virtual\_pin\_handler(pin,value):  file = open('/home/pi/data.txt','w')  print('Got V11 Blynk:',value)  file.write(str(value[0]))  file.close()  # register handler for virtual pin V12 reading @blynk.handle\_event('write V12') def read\_virtual\_pin\_handler(pin,value):  file = open('/home/pi/data2.txt','w')  print('Got V12 Blynk:',value)  file.write(str(value[0]))  file.close()   def blynk\_run():  print("Initialized Blynk")  while True:  blynk.run()   def background\_audio():  print("Background Audio Started")  while True:  os.system("omxplayer -o both /home/pi/sample.mp3")  print("Music Looped")     t1 = threading.Thread(target=blynk\_run) t1.start() t2 = threading.Thread(target=background\_audio) t2.start()  while 1:  with open(path\_to\_file,'rb') as f:  contents = f.read()   contents = str(contents)  #print(contents)  contents = contents.split("\\r\\n")[0]  temp = contents.split('TEMP:')[1].split(" ")[0]  humi = contents.split('HUM:')[1].split(" ")[0]  mois = contents.split('MOIST:')[1].split(" ")[0]  us = contents.split('US:')[1].split(" ")[0]  print("Updating Values to Sheets:", temp,humi,mois,us)  for ii in sheet:  #print(ii.title)  curr\_time = time.localtime()  time\_str = time.strftime("%m/%d/%Y %H:%M:%S",curr\_time)  ii.append\_row([time\_str,temp,humi,mois,us])  time.sleep(10) |

**Raspberry Pi 4 - Real Time Operate integration - main Function:**

|  |
| --- |
| #include <iostream>  extern "C" {  #include <stdio.h> #include <string.h> #include <errno.h> #include <wiringSerial.h> #include <wiringPi.h> #include <malloc.h> #include <stdlib.h> #include <wiringPiI2C.h> #include <pthread.h> #include <sched.h> // Scheduling library for prioritization of threads }  using namespace std;  int g, s, x, y;   // Define some device parameters #define I2C\_ADDR 0x20// 0x27 // I2C device address  // Define some device constants #define LCD\_CHR 1 // Mode - Sending data #define LCD\_CMD 0 // Mode - Sending command  #define LINE1 0x80 // 1st line #define LINE2 0xC0 // 2nd line  #define LCD\_BACKLIGHT 0x08 // On // LCD\_BACKLIGHT = 0x00 # Off  #define ENABLE 0b00000100 // Enable bit  void lcd\_init(void); void lcd\_byte(int bits, int mode); void lcd\_toggle\_enable(int bits);   void typeInt(int i); void typeFloat(float myFloat); void lcdLoc(int line); //move cursor void ClrLcd(void); // clr LCD return home void typeln(const char \*s); void typeChar(char val); int fd, fdl; // seen by all subroutines  int tmp = 0; int hum = 0; int moist = 0; int us = 0;  const int lightPin = 0; //GPIO 17 const int pumpPin = 2; //GPIO 27   void \*checkMoisture(void \*arg) {  FILE \*fp;  char ch[1];  int pumpflag = 2;  while(1)  {    fp=fopen("/home/pi/data.txt","r");  fscanf(fp,"%[^\n]",ch);  fflush(fp);  fclose(fp);    if ((moist > 450 || strcmp(ch,"1") == 0 ) && (pumpflag == 0 || pumpflag == 2))  {  digitalWrite (pumpPin, HIGH) ;   printf("Water Pump ON.\n");  pumpflag = 1;  delay(1000);  }  if(moist < 430 && (pumpflag == 1 || pumpflag == 2))//|| strcmp(ch,"0") == 0)  {  digitalWrite (pumpPin, LOW) ;  printf("Water Pump OFF.\n");  pumpflag = 0;  }  delay(1);  }  return 0; }   void \*checkTemp(void \*arg) {  int lightflag = 2;    FILE \*fp2;  char ch2[1];  while(1)  {    fp2=fopen("/home/pi/data2.txt","r");  fscanf(fp2,"%[^\n]",ch2);  fflush(fp2);  fclose(fp2);      if ((tmp < 90 || strcmp(ch2,"1") == 0) && (lightflag == 0 || lightflag == 2))  {  digitalWrite (lightPin, HIGH) ;   printf("Lights ON.\n");  lightflag = 1;  delay(1000);  }  else if (tmp >= 90 && (lightflag == 1 || lightflag == 2))  {  digitalWrite (lightPin, LOW) ;  printf("Lights OFF.\n");  lightflag = 0;  }  delay(1);  }  return 0; }   int main () {    printf("Code Started \n");    if(wiringPiSetup() == -1)  return 1;    printf("Setting Up GPIO Pins \n");  pinMode (lightPin, OUTPUT) ;  pinMode (pumpPin, OUTPUT) ;      printf("Connectiong to LCD \n");  fdl = wiringPiI2CSetup(I2C\_ADDR);      printf("Initializing LCD \n");  lcd\_init(); // setup LCD  lcdLoc(LINE1);  typeln("SMART GREENIFY");  lcdLoc(LINE2);  typeln("DEVICE");   printf("Trying Connecting to Arduino \n");    if ((fd = serialOpen ("/dev/ttyUSB0", 9600)) < 0)  {  fprintf (stderr, "Unable to open USB0 serial device: %s\n", strerror (errno)) ;  if ((fd = serialOpen ("/dev/ttyUSB1", 9600)) < 0)  {  fprintf (stderr, "Unable to open USB1 serial device: %s\n", strerror (errno)) ;  if ((fd = serialOpen ("/dev/ttyACM0", 9600)) < 0)  {  fprintf (stderr, "Unable to open USB2 serial device: %s\n MAKE SURE ARDUINO IS CONNECTED.\n", strerror (errno)) ;  }  }    }    serialFlush(fd);    int available = 0;  printf("Starting the Streaming \n"); system("/usr/bin/python3 /home/pi/streamingTest.py &"); printf("\n Video Streaming GStarted at: "); system("hostname -I"); printf(" port: 8000 \n");  printf("Starting Google Sheets, Blynk & Music \n"); system("/usr/bin/python3 /home/pi/google\_sheet.py &");  delay(5000);   pthread\_attr\_t moisture\_thread\_attr, temp\_thread\_attr;  pthread\_t moisture\_thread,temp\_thread;  sched\_param param1,param2;    // pthreads   pthread\_attr\_init (&moisture\_thread\_attr);  pthread\_attr\_init (&temp\_thread\_attr);    pthread\_attr\_getschedparam (&moisture\_thread\_attr, &param1);  pthread\_attr\_getschedparam (&temp\_thread\_attr, &param2);    param1.sched\_priority = 1;  pthread\_attr\_setschedparam (&moisture\_thread\_attr, &param1);   param2.sched\_priority = 2;  pthread\_attr\_setschedparam (&temp\_thread\_attr, &param2);      pthread\_create(&moisture\_thread, &moisture\_thread\_attr, &checkMoisture, NULL);  pthread\_create(&temp\_thread, &temp\_thread\_attr, &checkTemp, NULL);     while(1)  {  char data[500];  printf("Checking Available Serial Data... ");  available = serialDataAvail(fd);  printf(" Available Characters: %d\n", available);  if (available > 95)  {  serialFlush(fd);  continue;  }            if (available > 35)  {  printf("GOT DATA:");  for(int i=0; i < available; i++)  {  data[i] = serialGetchar(fd);  }      for(int j = 0; j < available; j++)  {  printf("%c",data[j]);  }    printf("\n");  FILE \*fr;  fr=fopen("/home/pi/data\_log.txt","w");  fprintf(fr,"%s",data);  fflush(fr);  fclose(fr);    int t = strstr(data, "TEMP:")-data;  int h = strstr(data, "HUM:")-data;  int m = strstr(data, "MOIST:")-data;  int u = strstr(data, "US:")-data;      char temp [5];  strncpy(temp,data+t+5,4);  printf("\nTEMP VAL: ");  tmp = atoi(temp);  printf("%d \n",tmp);        char hm [5];  strncpy(hm,data+h+4,4);  printf("\nHUM VAL: ");  hum = atoi(hm);  printf("%d \n",hum);        char mst [5];  strncpy(mst,data+m+6,4);  printf("\nMOIST VAL: ");  moist = atoi(mst);  printf("%d \n",moist);        char uls [5];  strncpy(uls,data+u+3,4);  printf("\nULTRASONIC VAL: ");  us = atoi(uls);  printf("%d \n",us);    printf(" \n");    //ClrLcd();    lcdLoc(LINE1);  typeln("Temp:");  typeInt(tmp);    typeln("F Hum:");  typeInt(hum);  typeln(" ");    lcdLoc(LINE2);  typeln("Mois:");    if(moist > 450) typeln("LO");  else if(moist > 250) typeln("OK");  else typeln("HI");    typeln(" Tank:");  if (us > 20) typeln("LO ");  else if(us > 10) typeln("OK ");  else typeln("FL ");      //checkTemp();  //checkMoisture();      delay(200);     }  else  {  printf("Serial Data not yet Ready.\n");  delay(200);    }   }   }      // float to string void typeFloat(float myFloat) {  char buffer[20];  sprintf(buffer, "%4.2f", myFloat);  typeln(buffer); }  // int to string void typeInt(int i) {  char array1[20];  sprintf(array1, "%d", i);  typeln(array1); }  // clr lcd go home loc 0x80 void ClrLcd(void) {  lcd\_byte(0x01, LCD\_CMD);  lcd\_byte(0x02, LCD\_CMD); }  // go to location on LCD void lcdLoc(int line) { lcd\_byte(line, LCD\_CMD); }  // out char to LCD at current position void typeChar(char val) { lcd\_byte(val, LCD\_CHR); }   // this allows use of any size string void typeln(const char \*s) { while ( \*s ) lcd\_byte(\*(s++), LCD\_CHR); }  void lcd\_byte(int bits, int mode) {   //Send byte to data pins  // bits = the data  // mode = 1 for data, 0 for command  int bits\_high;  int bits\_low;  // uses the two half byte writes to LCD  bits\_high = mode | (bits & 0xF0) | LCD\_BACKLIGHT ;  bits\_low = mode | ((bits << 4) & 0xF0) | LCD\_BACKLIGHT ;   // High bits  wiringPiI2CReadReg8(fdl, bits\_high);  lcd\_toggle\_enable(bits\_high);   // Low bits  wiringPiI2CReadReg8(fdl, bits\_low);  lcd\_toggle\_enable(bits\_low); }  void lcd\_toggle\_enable(int bits) {  // Toggle enable pin on LCD display  delayMicroseconds(500);  wiringPiI2CReadReg8(fdl, (bits | ENABLE));  delayMicroseconds(500);  wiringPiI2CReadReg8(fdl, (bits & ~ENABLE));  delayMicroseconds(500); }   void lcd\_init() {  // Initialise display  lcd\_byte(0x33, LCD\_CMD); // Initialise  lcd\_byte(0x32, LCD\_CMD); // Initialise  lcd\_byte(0x06, LCD\_CMD); // Cursor move direction  lcd\_byte(0x0C, LCD\_CMD); // 0x0F On, Blink Off  lcd\_byte(0x28, LCD\_CMD); // Data length, number of lines, font size  lcd\_byte(0x01, LCD\_CMD); // Clear display  delayMicroseconds(500); } |

In this final integrated real time code we have created two threads i.e moisture thread and temperature threads using pthreads library and we have assigned them the priorities . The moisture thread has priority level 1 and temperature thread has priority level 2. These threads will execute in real time along with the main thread

**Plug and Play mode:**

To use this function, we can use the Autostart in Raspberry pi. It is the best way to run GUI-based Raspberry Pi programs on startup. It works by ensuring that both the X Window system and the LXDE desktop environment are available before the system runs any of the scheduled programs.

To run any of the GUI-based programs/applications at startup on your Raspberry Pi, we need to schedule them to run using autostart.  
Steps:

1. Open the terminal and edit this: **sudo nano /etc/xdg/autostart/display.desktop.**
2. In the .desktop file. Adding the file needed to run

[Desktop Entry]

Name=autostart

Exec=/usr/bin/python3 /home/pi/autostart.py

1. Save and reboot the system.

# 

# Chapter VI

# Evaluation

**Introduction**

This chapter will illustrate the solution success related to the user requirement. It will also evaluate the methods through which we tested during the project development. This chapter discusses the success of a solution meeting the user requirement. Finally, the chapter will proceed further to show the solution we found in measuring soil. The communication protocol we changed and how it affected our project.

**Minimum Requirements**

Moisture Sensor, Minimum product requirement is to check the water content in soil and light incoming for the plant. The moisture sensor which we worked on was showing discrepancies in its reading. In order to reach its solution, we studied the sensor characteristics through which we came to know about the corrosion happening in the sensor tip due to the presence of water.

**Troubleshooting**

This section is related to the troubleshooting of the moisture sensor. The sensor used in the project was based on resistance which measured the resistance offered by the soil and the resistance varied incase of the variation of the amount of water. The method was very successful however with constant contact of the metal with water lead to corrosion and the reading differed.

The trouble was cleared when we switched to a new type of sensor known as Capacitive Type Sensor which uses the capacitance of the soil to calculate the moisture content. Stemma Capacitive Moisture sensor was used. The sensor used showed no output so the sensor was switched with another capacitive sensor known as Moisture Sensor v1.2. However, the new sensor was analog which was further interfaced with Raspberry Pi 4 through the components which already worked on it in our third semester: Arduino Uno

Following is the link of the conversation of the group mates with our instructor:

<https://github.com/vyhoangquocnguyen/CapstoneProject_G1/issues/8>

.

# Chapter VII

# Conclusion

The aim of the project is to develop a device that no longer seeks human involvement to take care of the plants. The device is made of a strong 1.5 ft x 1ft square case which can accommodate up to 6 plants or 12 small seeds. The light-induced will not heat up the plants and the moisture will be in control

The project can be developed more into the agricultural sector by working into more concepts like soil fertility, disease detection & light variation. It can be achieved by working with faster MCU/CPUs like Pine H64. The LEDs such as WRGBICs etc.

**Future Work**

As discussed above, the future scopes of the project can be working more into the fields which can not only increase the plant growth but also make the user free from further involvement. These are as following:

1). LED lights: The lights used for the project are white which are sufficient for the growth and flowering. However, if we go further into the plant growth the light rich in blue component helps in the growth while the red component helps in flowering and reproduction. This can be achieved by introducing WRGB led strips.

2). Image Processing: By introducing image processing the users will get notification about the color changes on the leaves of the plants and also when to harvest the vegetables if it is a kitchen garden. However, this needs a huge amount of research on different kinds of plants and the diseases mainly found in them. This will be a smart way of disease detection in plants.

# Chapter VIII

# User’s Guide

**Setting Up**

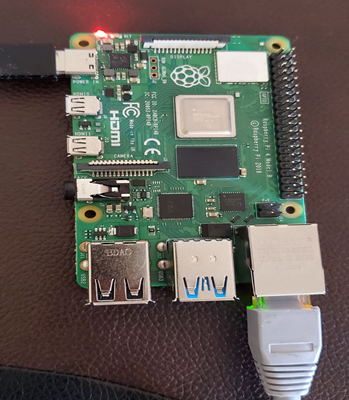
Place a plant in the frame. Insert the sensors into the pot carefully. The setup is shown in the figure below.



*Figure 8.1: Plant placed in the frame*

**Powering Up**

Connect the C-type USB of Raspberry to the power source and there is an additional power source for LEDs and a Water Pump of 12V. The power connections can be seen from the figure below.



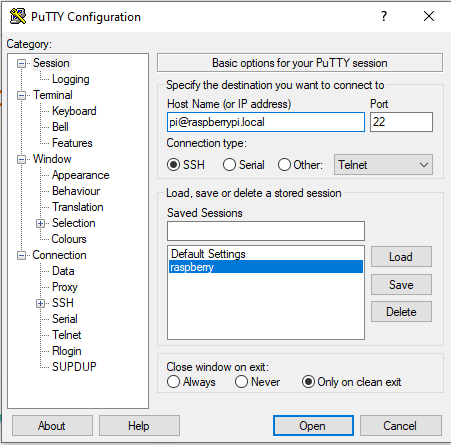
*Figure 8.2: Raspberry Pi Setup*

If the connections are successful you will see the small LEDs flashing on Raspberry Pi4..

**Connecting to Server**

In case if we need to debug the device, we can use Putty to connect to the Raspberry Pi from the desktop

* Connect Raspberry Pi with the computer.
* Download PUTTY software on the computer.
* Open PUTTY software and type in host- pi@raspberrypi.local. Look at the figure8.3 below.

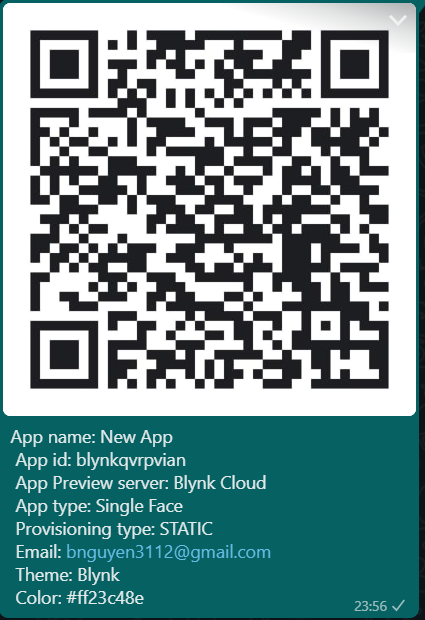
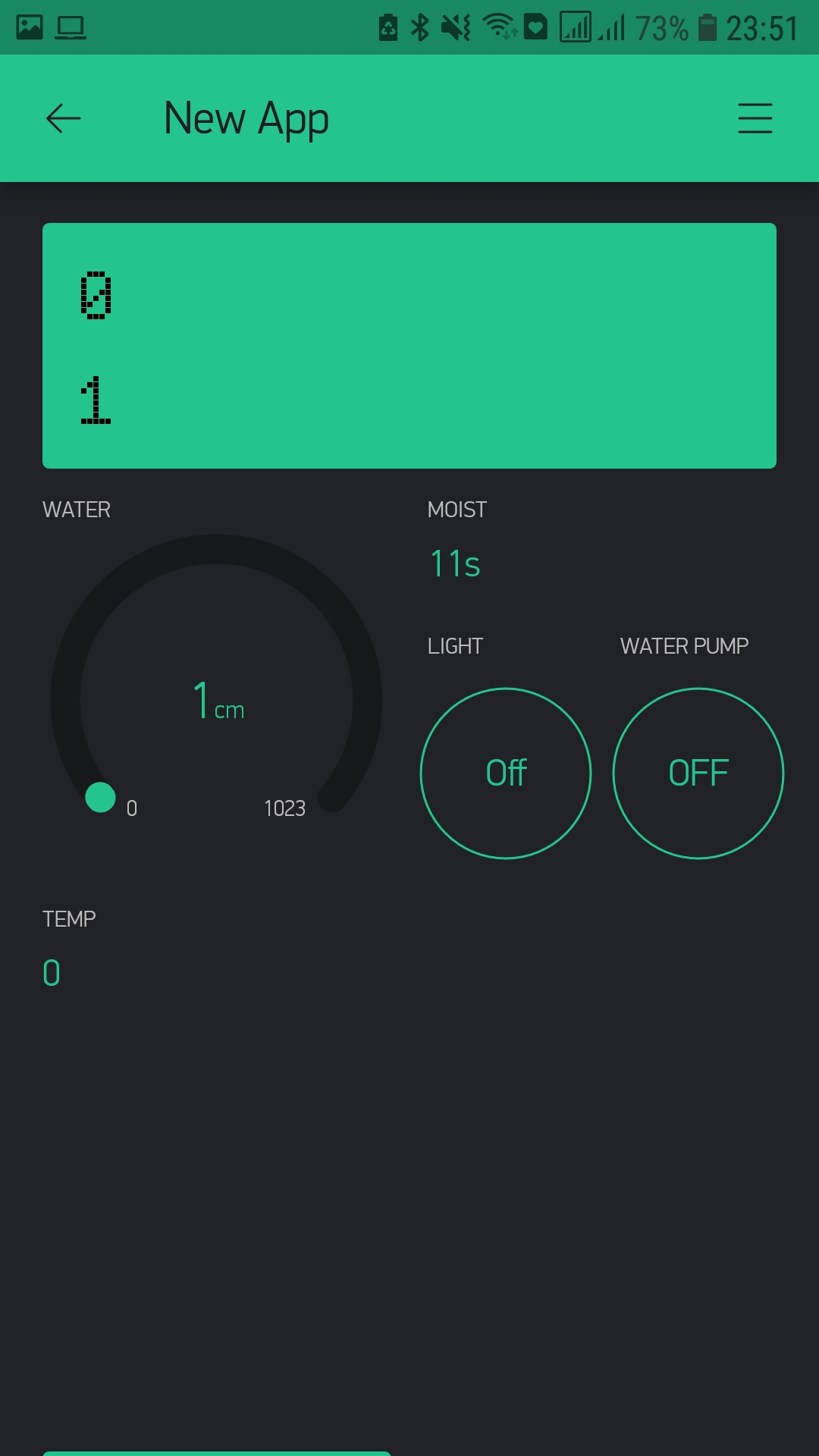


*Figure8.3: Putty software window*

* The putty terminal will open and type the password as Raspberry
* Now type ifconfig
* This will show the IP address of Raspberry Pi 4
* Use this IP address to open the link by typing http://**<Your\_Pi\_IP\_Address>**:8000
* Thus, you will be able to see the video of your plants.

**Connect to the App.**

After plug in and power up the device, go to the App Store and get our Greenify App. The app will look like this:

****

*Figure 8.4 App installation and App UI.*

# Chapter IX

# References

# 

1. Randomnerdtutorials.com. 2021. *Video Streaming Raspberry Pi Camera | Random Nerd Tutorials*. [online] Available at: <https://randomnerdtutorials.com/video-streaming-with-raspberry-pi-camera/>
2. Picamera.readthedocs.io. 2021. *4. Advanced Recipes — Picamera 1.13 Documentation*. [online] Available at: <https://picamera.readthedocs.io/en/latest/recipes2.html>
3. Author, M., 2021. *Raspberry Pi Video Streaming*. [online] Instructables. Available at: <https://www.instructables.com/Raspberry-Pi-Video-Streaming/>
4. 4. Raspberry Pi Stack Exchange. 2021. *How to stream video from a Raspberry Pi camera and watch it live*. [online] Available at: <https://raspberrypi.stackexchange.com/questions/23182/how-to-stream-video-from-raspberry-pi-camera-and-watch-it-live>
5. Medium. 2021. *Cloud IoT step-by-step: Connecting Raspberry PI + Python*. [online] Available at: <https://medium.com/google-cloud/cloud-iot-step-by-step-connecting-raspberry-pi-python-2f27a2893ab5>
6. Accenture.com. 2021. *What is Cloud Computing & Why is it Important? | Accenture*. [online] Available at: <https://www.accenture.com/ca-en/insights/cloud-computing-index?c=acn_glb_brandexpressiongoogle_12261974&n=psgs_0621&gclid=CjwKCAjwos-HBhB3EiwAe4xM904FqOZ1-m9tG-5x4rd3qn5J2HEmFifahotfX8ihYUSsC6fmD94vJxoCaTUQAvD_BwE&gclsrc=aw.ds>
7. Google Cloud. 2021. *Cloud IoT Core | Google Cloud*. [online] Available at: <https://cloud.google.com/iot-core>
8. Pi, R. and NodeJS, C., 2021. *Connect Raspberry Pi to Google Cloud IoT (GCP IoT) using NodeJS*. [online] SwA. Available at: <https://www.survivingwithandroid.com/connect-raspberry-pi-to-google-cloud-iot-nodejs/>
9. User, S., 2021. *Raspberry Pi And The IoT In C - I2C Bus*. [online] Iot-programmer.com. Available at: <https://www.iot-programmer.com/index.php/books/22-raspberry-pi-and-the-iot-in-c/chapters-raspberry-pi-and-the-iot-in-c/61-raspberry-pi-and-the-iot-in-c-i2c-bus?start=3>
10. Linux.die.net. 2021. *sleep(3): sleep for specified number of seconds - Linux man page*. [online] Available at: <<https://linux.die.net/man/3/sleep>
11. GitHub. 2021. *ControlEverythingCommunity/SI7021*. [online] Available at: <<https://github.com/ControlEverythingCommunity/SI7021/blob/master/C/SI7021.c>>
12. 4.Silabs.com. 2021. [online] Available at: <https://www.silabs.com/documents/public/data-sheets/Si7021-A20.pdf>
13. Ada, L., Dicola, T., & Rembor, K. (n.d.). *I2C/SPI LCD Backpack*. Adafruit. Retrieved June 28, 2021, from <https://learn.adafruit.com/i2c-spi-lcd-backpack/overview>
14. Adafruit. (n.d.). *Raspberry Pi Char LCD Plate*. Adafruit. Retrieved June 28, 2021, from <https://learn.adafruit.com/character-lcd-with-raspberry-pi-or-beaglebone-black/raspberry-pi-char-lcd-plate>
15. Loflin, L. (2018). *I2C LCD to Raspberry Pi in C*. bristolwatch. Retrieved June 28, 2021, from <https://www.bristolwatch.com/rpi/i2clcd.htm>
16. Monk, S., & Ada, L. (n.d.). *Adafruit's Raspberry Pi Lesson 4. GPIO Setup*. Adafruit. Retrieved June 28, 2021, from <https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/the-gpio-connector>