Automated Composter Final Report

Prepared for:

Ed Casas

Head of Telecom and Networking

Susan Woo

COMM 2443 Instructor

BCIT

Prepared by:

Max Gibson

Sahil Nathani

Tom Walter

Telecommunications and Networking Option

21 April 2019

Contents

List of figures

Summary

The purpose of this report is to discuss the development of the Smart Bin prototype, its specifications, achievements and results, and future considerations that will continue to improve the prototype. The goal of our project was to design a smart composter that would address three measurable quantities and provide a user-friendly LCD screen that would accurately monitor and clearly display the quantities. The quantities are temperature, moisture, and bin capacity level. The compost bin would also be able to send this data over Wi-Fi to an MQTT test server and on a LCD screen attached to the bin where the data would be displayed for the owner/user of the compost bin. Details of the specifications are provided below and a full description of the Python Code is seen at Appendix A.

The report further outlines specific challenges that needed to be overcome throughout the development process. Coding on the Raspberry Pi, troubleshooting the code, adapting the size of the bin enclosure, and the inability to add some of the additional features that would have enhanced the bin's capabilities were challenges that were addressed throughout the process. Delays in choosing the project topic, waiting for the arrival of parts, and significant coding issues put the project behind schedule, and forced the team to rush to put all the final components in place.

In order to meet the budget restraints, parts were carefully chosen; the main cost was the Raspberry Pi, an essential component to the prototype.

Primarily because of time limitations, some of the additional and optional features could not be included in the design; however, the smart compost bin did meet the essential specifications, and the results proved the sensors measured the quantities accurately and the read outs were easily visible.

Finally, the smart bin composter was designed to assist users in regulating the appropriate nutrients in their compost by having accessible and ongoing data to assist them in preparing their compost and achieving the best results. Further enhancements based on the recommendations provided at the end of the report will optimistically create a desirable and usable product for gardeners that is eco-friendly and meets their growing needs.

Introduction

The final report outlines the key developments that were engineered to create a smart bin composter that would sense measurable quantities within the compost and regulate the most effective composting nutrients in the soil. The readership of this report may require familiarity with electronic components in order to understand the circuitry involved in the sensing devices. Additionally, this project's future includes further development of the smart bin software. This continuation of work can assist future BCIT computer science students with individual software innovations that can help advance both their own understanding of software design as well as improve the effectiveness and usefulness of eco friendly composting.

The stakeholder, Dhalathan Aiyathurai-Kandasamy, met with our project team on January 21, 2019 to discuss the development of the compost prototype. He outlined the specific measurement requirements for the prototype, and the team felt we could meet his expectations within the limited time period of one semester. As a key stakeholder, Dhal had the authority to advise, consult, and authorize budget and direction of the project. Dhal is currently a project leader at BCIT, and his key responsibilities are to create innovative projects for BCIT students and also feature projects at certain conventions related to eco friendly environmental advancements. The 2019 Eco World Summit, hosted by BCIT, is the convention at which Dhal plans to display our compost prototype. Communication 2443 instructor Susan Woo and ELEX 4560 instructor Ed Casas were available for ongoing assistance and provided the standards for technical communication and the criteria for the project development.

The design of the prototype is a regular size city compost bin found in most backyards. The difference between our prototype and what is already available on the market is our bin is complete with measurement access, connected to the bin itself versus devices already available that only have a measure stick inserted in the bin and not wholly connected within the bin. Organic waste that is currently not being recycled and reused as compost negatively impacts the environment and when disposed improperly adds to the massive waste accumulated in land fills. If there can be more household recycling of organic waste, it has the potential to reduce the impact on the environment. The smart bin will be a welcome and desirable product for the future: it is functional and purposeful and has efficiencies that can accurately assess the proper nutrient ratios required in rich soil that can be used for gardening and vegetable production. Larger scale use of the bin will further reduce organic waste and allow users to be self sufficient and proactive in determining how to best manage their garden waste for their own purposes.

After meeting with Dhal back in January 2019, we had a clear understanding of the scope of the project. We were able to meet all aspects of the necessary scope but were unable to complete some additional features that are listed in the recommendations section. Limitations of this project were purely the minimal budget allocated at the start of this project that prevented purchase of more sophisticated sensors.

Our mentor Ed Casas assisted enormously in all of the software requirements. He helped with both completing the project and understanding how the components worked. With his assistance, the team was able to overcome any obstacles and proceed with project delivery on time.

The report is written as a formal engineering report with appropriate sections. The main sections of the report are Project Description, Challenges, Schedule, Cost Analysis, Conclusions, and Recommendations. In addition, figures, references, and appendixes are listed to provide the reader with relevant information and additional documentation.

Project Description

Design and Operations

Our device needed to perform a number of tasks including taking relevant measurements from the compost and displaying the information on various platforms in a way that was simple and accessible to the user. The block diagram below illustrates these requirements.

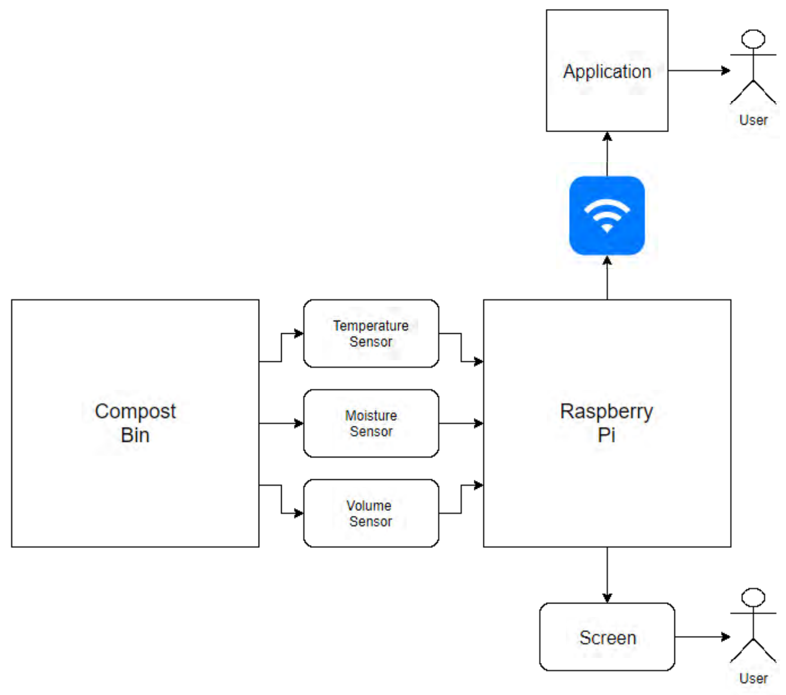


Figure 1: Block diagram:

In the functional block diagram above it can be seen that the sensors include temperature, moisture and volume measurements. We achieved this through the use of five temperature sensors and one capacitive moisture sensor. The moisture sensor measures relative moisture level and the temperature sensors measure the compost temperature at various levels within the bin. Parts of our code we sourced from Circuit Basics [1], TheRaspberryPiGuy [2], and the BCIT Telecom Wiki [3] We also used the temperature sensors to give a volume reading. To do this we oriented the top sensor so it would read the air temperature and compared the readings of the other sensors with this air temperature. Based on how close the other temperature readings were to this air temperature our code would decide how full the bin was. These measurements are sent over a Wi-Fi link to an MQTT test server as well as printed on an LCD screen that is on the enclosure. For an in depth look at how the code works see Appendix A: Python code.

The compost bin we used is from the City of Vancouver and stands 33 inches tall with a width of 31 inches. To place the sensors in the bin we decided to use a two-foot length of PVC pipe placed in the center of the bin to run our sensor cables through and have them sticking out of the pipe. The wires are sent to the bottom of the bin and back up another two-foot length of PVC pipe to the external enclosure where they are soldered to a breakout board that connects to the

Raspberry Pi with a ribbons cable. For a complete layout of the bin and soldering see Appendix B: Schematics and Diagrams.

To start the Raspberry Pi and run the program for the sensors the device must be plugged and once it finishes its boot-up process and connects to the internet the program begins running automatically. The sensor take their readings and the Raspberry Pi prints them to the LCD screen and MQTT screen. The moisture sensor puts out an analog signal which becomes digitized by an ADC before it reaches the Raspberry Pi. For details on how the components work see Appendix D: Datasheet links.

Specifications

For our main components that make up the readings inside the compost bin we used one moisture sensor and five different temperature sensors to make the basic measurements. We used the moisture sensor at the bottom of the bin so that it can survive heavy dumping of compost; most of the moisture is going to be at the bottom so it would be easy to know if there is too much water at the bottom. For the temperature sensors we have two different purposes for them: one is to tell the temperature inside the bin, two is to tell an approximation of how full the bin is. We can achieve measuring the approximate fullness of the bin by using our fifth temperature sensor as an air temperature that does not touch the compost and then the other four as the compost temperature sensor. So, if all the temperatures are the same then it will produce an empty reading if sensor four and sensor one in the bin are the same but sensor five is different then the bin is full but any other reading will cause the reading to be blank meaning there is still room in the bin. Once we have all of the measurements working, we then displayed it on an LCD display showing three measurements as shown below.

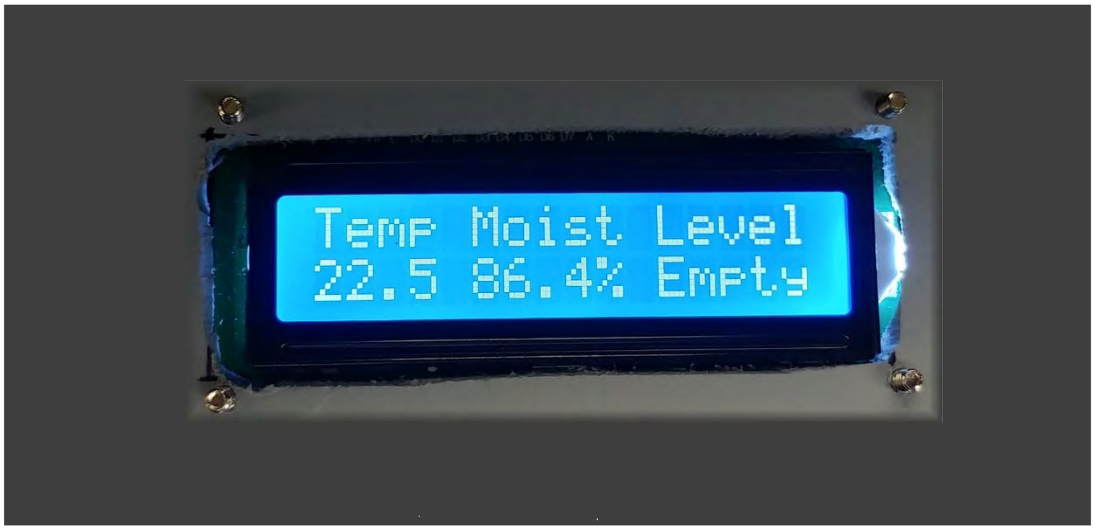


Figure 2: LCD Screen

It shows the average temperature of all four sensors that measure the compost, the relative moisture inside the bin to the compost, and the approximate fullness of the bin. The fullness can display three levels empty, nothing, and full.

The Raspberry Pi also sends the data it receives through a Wi-Fi link that is manually entered to the Raspberry Pi to an MQTT test server. To make the MQTT server to work we print the measurements to a specific directory that can be seen anywhere, but since it's a test server we only send small amounts of data for a temporary amount of time. The data that is on the server can be then used by any devise or websites.

Challenges and solutions

For the project we had a couple of challenges; our first challenge was creating a direct layout of how to go at completing the project from what specific sensors we need to make the composting process faster. We ended up with three sensors that would accomplish this temperature, moisture, and PH of the compost. For the PH sensor, we could not do it because all the sensors that can do what we want can only ship to the USA and they were all above fifty dollars and would come too late for us to make use of in time. In the end, we only could implement temperature and moisture to our compost bin measurements.

Our next challenge was coding on the Raspberry Pi and using it because none of us had any experience on how to use one or how to code on it, so our solution was to use example codes online and YouTube how to use it. The problem with relying on code online from a language none of us are familiar with is it's hard to troubleshoot on what exactly is wrong with the code when the compiler says it's good. To figure out how to troubleshoot we mainly tried to google what each function does and how to use them, and that worked for the most part except for our moisture sensor. Our moisture sensor was not as common as other types of sensors and we had to code using an analog to digital converter chip, so it was hard finding relevant and useful code for it. To fix this problem we spent hours with our mentor Ed Casas to find code that might work and Frankenstein those codes while adding other code that might work with it to make a finalize fourteen lines of code to read our sensor.

The compost bin, we had a problem with the size of the enclosure we bought and that we figured out two days before our project was due. To fix the problem Tom who was working on the enclosure found a much bigger box and remade all of the design cuts and mounting we needed in time for our due date.

The final week of our project our SD card would not work anymore with the Raspberry Pi, which caused us not to be able to run any code or access the Raspberry Pi through SSH. It took a long time and, in the end, nobody could fix it without spending hours on it so to fix the problem we got a different SD card and re-downloaded all of our files and copied our saved code from a USB we kept.

Our final challenge was time because to make a composter that looks and does all of the extra optional features, we had to finish our main code by the end of the reading break but we could have foreseen the time that coding would take us even with the extra time we gave it. In the end, time won and could only do the base requirements without the extra features such as battery it being battery powered.

Schedule

We started off delayed from the time we were supposed to choose our project because of the options and availabilities of different projects we took longer to decide. To get a rough idea of the scope of the project we talked to our sponsors and then talked to our mentor to see the viability of completing the different tasks. To get the final scope of the document it took about one month to complete from our three-month total time limit. After we got a finalized idea, we searched for parts that could do the job and to find the right parts we searched main for parts that would come in less than two weeks and we ended up getting most of our parts from Amazon. For a complete list of parts see Appendix C: Bill of Materials. Also, at this point found that buying a PH sensor and a regular portable battery was not a viable choice because it was too expensive for the PH and a portable battery pack would not last one year for our Raspberry Pi.

The whole process of buying parts and waiting for it to arrive took a total of one month two and a half weeks of searching and one and a half weeks of waiting for it to arrive. We finally got started coding and done with the planning stage on March 4th so there was no more room for changing the plans. The longest stage for us was coding on the Raspberry Pi, we started off coding the temperature sensors first and that took three days to complete from March 11th – March 13th. After coding the temperature sensor, we started working on the moisture sensor which the Raspberry Pi read off the analog to digital output. Coding the moisture sensor to read was one of the biggest delays because no matter what we tried as a group we could not get it to work and our mentor was not available until March 11th. The moisture sensor code was completed with a lot of help from Ed on March 20th the entire period for working on the moisture sensor was from March 12th – March 19th. Once we completed the sensors, we worked on coding the server to gather the measurements and that only took two days to complete. For the week of March 25th, we did final polishing and testing of the code before the assembly stage. On March 30th we have soldered the sensors on a breakout board but during the process of soldering and putting all the code together with the SD card on our Raspberry Pi broke so we were delayed for another three days until Ed could help us figure out the problem. During the final week of the project cycle, we had to rush and assemble all of the parts together and create the presentation within two days and finally on April 5th we presented and ended our full project.

Cost Analysis

The completed project was relatively cost effective. The components we purchased were not overly expensive. We were able to find sensors and even a compost bin that did exactly what we needed without breaking the bank. Our most expensive item was the Raspberry Pi which came in at about 70 dollars. Our total monetary cost was 194 dollars Canadian. For a complete list of costs please see Appendix C: Bill of Materials.

The largest cost to designing and building this composter was time. Each member of the team spent about 30 to 40 hours of their time in developing this composting system. The most time-consuming part of this project was easily the programming. This alone took about 20 hours of hour time.

Conclusions

In conclusion, our prototype has the capability to measure temperature, moisture, and approximate volume with measurements clearly displayed on the side of the bin. The sensors, microcontroller, and LCD screen are safely housed in a sturdy water protected enclosure box. The sensors are placed securely within the middle portion of the bin. All sensors and other electrical components are soldered to the Raspberry Pi for minimal chance of a poor connection.

The hardware housed all the components inside the bin securely, was cosmetically appealing to the user, and was positioned in the optimal location to record precise and accurate measurements. The data measurements can be sent over Wi-Fi for the user's ongoing awareness of the compost's progression.

The team wrote code for all sensors. The code ensured accurate measurements of both temperature and moisture sensors, and displayed the measurements on a screen. An MQTT web server connected The Raspberry Pi to the internet and all measured sensor data were placed on to an MQTT server.

The smart bin can continue to be modified by future students to include more sophisticated features that will enhance its functionality.

Now that the project team has finished with the compost bin, the prototype moves on to the second stage of development. The compost bin will now be used as a project of its own for CST students. Our project team had finished all the necessary hardware components of the bin, so the CST can solely focus on the software and web design portions of this ongoing development. Our results and achievements successfully met the expectations as proposed by Dhal at the beginning stage.

Once the bin has been tested fully with composting material deposited in the bin, approved for use, and seen to be both a profitable and desirable product, business partnerships could be forged with city waste and planning departments and the bins rolled out for more widespread uses. Many householders currently compost to improve their soils and gardens; if this prototype has the potential of increasing their commitment to ecofriendly habits, then it will be a successful product that will have succeeded in fulfilling its purpose.

Recommendations

With the short amount of time allocated to this project, we were unable to complete all of the features that would make an ideal automated composter. These further developments could assist in building the capacity of the Smart Bin in order to make it an effective, practical, and desirable product for users.

1. Add an additional a sensor that will measure PH levels. Adding an additional PH sensor would allow the customer to do an acidity test on the soil.
2. Add a battery powered (sleep mode) feature. Adding batteries to the compost bin instead of continuously being plugged into an outlet would greatly increase the efficiency of the bin.

3. Add a button to properly power down the Pi. As of right now the Pi must be left on to operate. Adding a power button would make the bin more efficient as well adding simplicity for the customer.
4. Create a user-friendly Webpage that provides composting information, tips on effective composting, and instructions on reading the bin readouts.
5. Work with city planners and waste departments to promote the smart bins and the positive impact.
6. Develop a mechanical arm to turn compost on a regular basis. This mechanism would save the user having to stir the compost manually and would be a convenient and practical addition to the bin.

References

[1] Circuit Basics, "Circuit Basics," 2012. [Online]. Available: http://www.circuitbasics.com/raspberry-pi-ds18b20-temperature-sensor-tutorial/. [Accessed March 2019].

[2] TheRaspberryPiGuy, "YouTube," 2016 March 30. [Online]. Available: https://www.youtube.com/watch?v=fR5XhHYzUK0. [Accessed March 2019].

[3] E. Casas, "BCIT ECET Telecom Wiki," 2019 March 18. [Online]. Available: https://tcom.bcit.ca/pi_hints. [Accessed March 2019].

Appendix A: Python code

```
import os
import glob
import time
from datetime import datetime
import lcddriver
import spidev
import numpy
import paho.mqtt.client as mqtt
import math

#########################
# Set up all variables #
#########################
```

```
#For setting server var and connection
client = mqtt.Client()
client.connect('test.mosquitto.org')

#something to do with lcd screen
display = lcddriver.lcd()

#allows use of the sensors
base_dir = '/sys/bus/w1/devices/'

#name the CSV file for the data log
out_filename = '/home/pi/Desktop/temp.csv'

#initialize variables for counting hot readings
i1 = 0
i2 = 0
i3 = 0
i4 = 0
i5 = 0
#insert device serial numbers here
sn1 = '28-011432e97dc1'
sn2 = '28-011433ca7137'
sn3 = '28-011432e747f3'
sn4 = '28-02131e15cfaa'
sn5 = '28-011432da8005'

#initialize all of the directories for the sensors
device_file1 = glob.glob(base_dir + sn1)[0] + '/w1_slave'
device_file2 = glob.glob(base_dir + sn2)[0] + '/w1_slave'
device_file3 = glob.glob(base_dir + sn3)[0] + '/w1_slave'
device_file4 = glob.glob(base_dir + sn4)[0] + '/w1_slave'
device_file5 = glob.glob(base_dir + sn5)[0] + '/w1_slave'

#################################
# Routines to read each sensor #
#################################
#Read Sensor 1
def read_temp_raw1():
	f = open(device_file1, 'r')
	lines1 = f.readlines()
	f.close()
	return lines1

def read_temp1():
	lines1 = read_temp_raw1()
	while lines1[0].strip()[-3:] != 'YES':
```

```
                time.sleep(0.2)
                lines1 = read_temp_raw1()
        equals_pos = lines1[1].find('t=')
        if equals_pos != -1:
      temp_string1 = lines1[1][equals_pos+2:]
                temp_c1 = int(temp_string1) / 1000.0
                temp_c1 = float(round(temp_c1, 3))
                return temp_c1

#Read Sensor 2
def read_temp_raw2():
        f = open(device_file2, 'r')
        lines2 = f.readlines()
        f.close()
        return lines2
def read_temp2():
        lines2 = read_temp_raw2()
        while lines2[0].strip()[-3:] != 'YES':
                time.sleep(0.2)
                lines2 = read_temp_raw2()
        equals_pos = lines2[1].find('t=')
        if equals_pos != -1:
                temp_string2 = lines2[1][equals_pos+2:]
                temp_c2 = int(temp_string2) / 1000.0
                temp_c2 = float(round(temp_c2, 3))
                return temp_c2

#Read Sensor 3
def read_temp_raw3():
        f = open(device_file3, 'r')
        lines3 = f.readlines()
        f.close()
        return lines3
def read_temp3():
        lines3 = read_temp_raw3()
        while lines3[0].strip()[-3:] != 'YES':
                time.sleep(0.2)
                lines3 = read_temp_raw3()
        equals_pos = lines3[1].find('t=')
        if equals_pos != -1:
                temp_string3 = lines3[1][equals_pos+2:]
                temp_c3 = int(temp_string3) / 1000.0
                temp_c3 = float(round(temp_c3, 3))
                return temp_c3

#Read Sensor 4
```

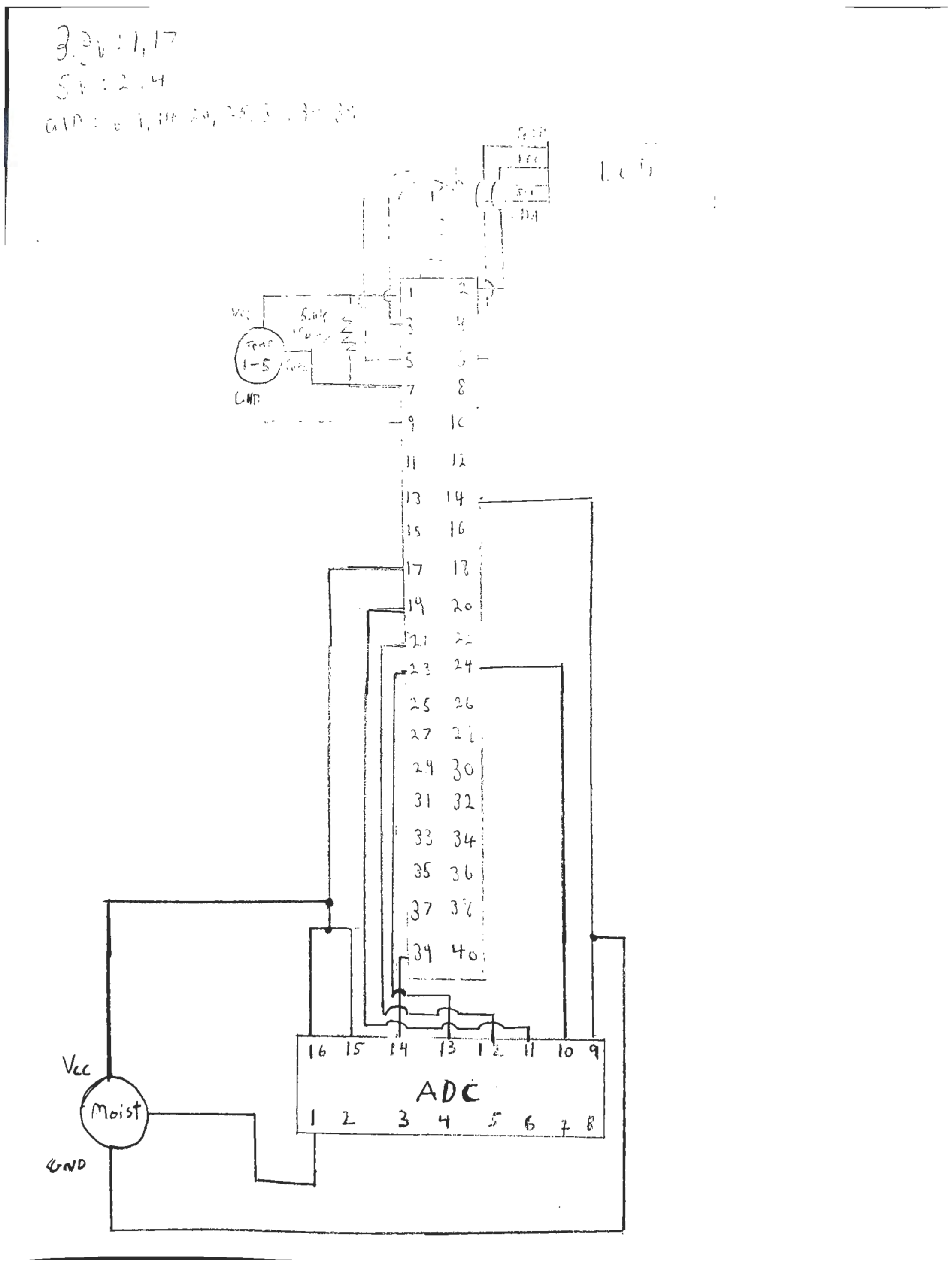
```
def read_temp_raw4():
        f = open(device_file4, 'r')
        lines4 = f.readlines()
        f.close()
        return lines4

def read_temp4():
        lines4 = read_temp_raw4()
        while lines4[0].strip()[-3:] != 'YES':
                time.sleep(0.2)
                lines4 = read_temp_raw4()
        equals_pos = lines4[1].find('t=')
        if equals_pos != -1:
                temp_string4 = lines4[1][equals_pos+2:]
                temp_c4 = int(temp_string4) / 1000.0
                temp_c4 = float(round(temp_c4, 3))
                return temp_c4

#Read Sensor 5
def read_temp_raw5():
        f = open(device_file5, 'r')
        lines5 = f.readlines()
        f.close()
        return lines5

def read_temp5():
        lines5 = read_temp_raw5()
        while lines5[0].strip()[-3:] != 'YES':
                time.sleep(0.2)
                lines5 = read_temp_raw5()
        equals_pos = lines5[1].find('t=')
        if equals_pos != -1:
                temp_string5 = lines5[1][equals_pos+2:]
                temp_c5 = int(temp_string5) / 1000.0
                temp_c5 = float(round(temp_c5, 3))
                return temp_c5

adc=spidev.SpiDev()

CH0 = 8     # single-ended CH0

adc.open(0,0)
adc.bits_per_word=8
adc.max_speed_hz=30517

#sets configuration bits for adc in the right order
```

```
def ReverseBits(byte):
 byte = ((byte & 0xF0) >> 4) | ((byte & 0x0F) << 4)
 byte = ((byte & 0xCC) >> 2) | ((byte & 0x33) << 2)
 byte = ((byte & 0xAA) >> 1) | ((byte & 0x55) << 1)
 return byte


def ave_temp():
      ave = str(round(((read_temp5() + read_temp4() + read_temp3() + read_temp2() +
read_temp1()) / 5), 1))
      return ave

#############
# Main Loop #
#############

while True:

#prints temperatures to cmd line
       print("T1 = ", read_temp1())
       print("T2 = ", read_temp2())
       print("T3 = ", read_temp3())
       print("T4 = ", read_temp4())
       print("T5 = ", read_temp5())

#reads moisture level and prints to cmd line
       data=adc.xfer2([ReverseBits(3),0,0])
       good = float((((data[1]&3)<<8) + data[2]))
       moist = str(round(((good / 1024) * 100), 1))
       print("moisture", moist)

#Checks the compst level and prints full, empty or blank based on the level
#Level is decided using the temperature sensors
#It is assumed that t5 is the top sensor and is in air
#Code runs comparing other temperatures to this air temperature
       t1 = read_temp1()
       t4 = read_temp4()
       t5 = read_temp5()
#Maximum temperature differnce is 2 degrees celcius
       maxdiff = 2
       empty = abs(t5 - t1) < maxdiff
       full = (not empty) and (abs(t4-t1) < maxdiff)
       if empty:
              display.lcd_display_string("Temp Moist Level", 1)
              display.lcd_display_string(ave_temp() + " " + moist + "% Empty", 2)
              client.publish("elex4560/compostbin/capacity", "Empty")
```

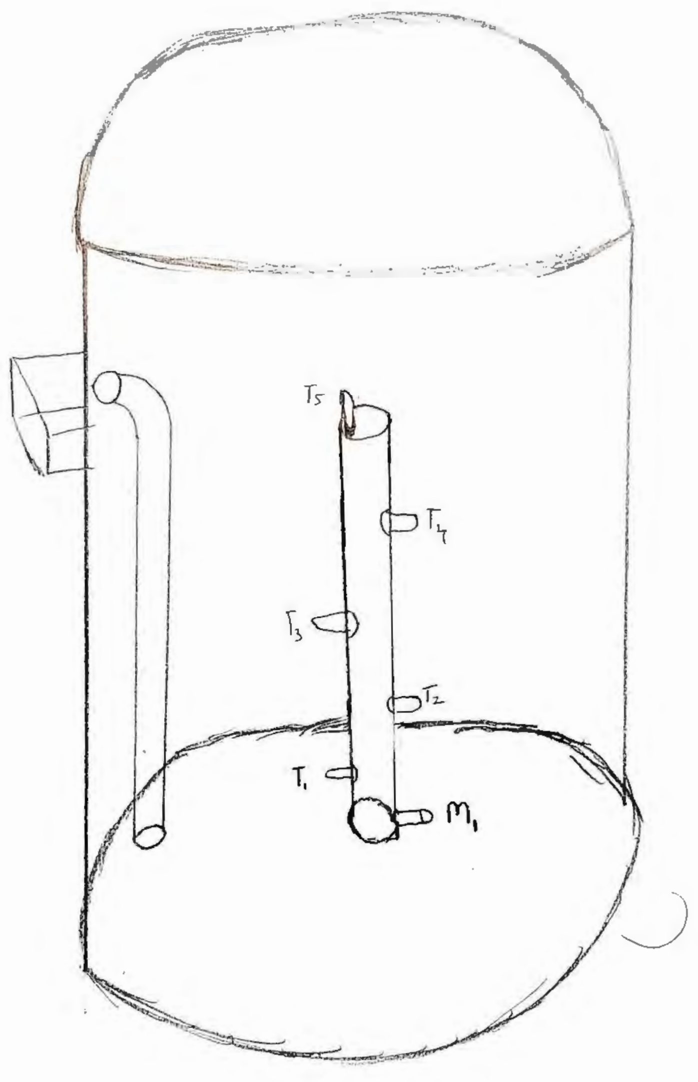
```
        elif full:
                display.lcd_display_string("Temp Moist Level", 1)
                display.lcd_display_string(ave_temp() + " " +  moist + "% Full ", 2)
                client.publish("elex4560/compostbin/capacity", "Full")
        else:
                display.lcd_display_string("Temp Moist Level", 1)
                display.lcd_display_string(ave_temp() + " " +  moist + "%       ", 2)
                client.publish("elex4560/compostbin/capacity", "     ")

#displays temp and moisture on lcd screen
        display.lcd_display_string("Temp:  Moisture:", 1)
        display.lcd_display_string(read_temp1() + "     " + good, 2)

#publishes moisture and temperature to mqtt server
            client.publish("elex4560/compostbin/temp1", read_temp1())
            client.publish("elex4560/compostbin/temp2", read_temp2())
            client.publish("elex4560/compostbin/temp3", read_temp3())
            client.publish("elex4560/compostbin/temp4", read_temp4())
            client.publish("elex4560/compostbin/temp5", read_temp5())
        client.publish("elex4560/compostbin/Average", ave_temp())
            client.publish("elex4560/compostbin/moisture", moist)
        time.sleep(1)
```

Appendix B: Schematics and Diagrams

Breakout board schematic



Bin Diagram



M = Moisture sensor

T = Temperature sensor

Appendix C: Bill of Materials

Item	Model/Type	Cost	Supplier
Temperature Sensors (Pack of 5)	DS18B20	22.99	Amazon
Moisture Sensors (Pack of 2)	EK1940	16.88	Amazon
Raspberry Pi	3 B+	69.99	Amazon
Compost Bin	N/a	25.00	City of Vancouver
PVC Pipe (6ft lenth)	N/a	19.95	RONA
Enclosure	1594ESGY	10.80	RP Electronics
Breakout kit	N/a	10.50	Amazon
LCD Screen	1602	12.99	Amazon
ADC	MCP3008	5.00	RP Electronics

Appendix D: Datasheet links

Raspberry PI 3 Model B+

https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

TC1602A LCD Screen

https://cdn-shop.adafruit.com/datasheets/TC1602A-01T.pdf

DS18B20 Temperature Sensor

https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

MCP3008 Analog to Digital Converter

https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf