Data Acquisition System to Measure and Monitor Temperatures and Atmospheric Air Parameters

David N. Long, James Madison University

David N. Long is a recent graduate of James Madison University's, Integrated Science and Technology program. David studied energy and production systems where he worked on the Water Out Of Thin Air project. The aim of the project was to provide clean water with low energy inputs. David concentrated on the Data Collection System which was designed and built to record the key parameters of the project in Costa Rica.

Dr. Karim Altaii, James Madison University

Dr. Altaii holds a Ph.D. in mechanical engineering, and received his doctorate from The City University of New York. He is a professor in the College of Integrated Science and Engineering (CISE) at James Madison University. He is a registered Professional Engineer and holds five patents in solar energy applications and irrigation system. He is the director of CISE Energy and Environmental Projects- an international summer program in Costa Rica. He is the Director of the Advanced Thermal-Fluids laboratory. His primary interests are in renewable energy applications, fluid-thermal sciences, and international education.

Data Acquisition System to Measure and Monitor Temperatures and Atmospheric Air Parameters

Abstract

The optimization and performance of many systems require data collection for the temperature variation of solids, liquids, and air parameters that vary from one location to another. For example, monitoring the performance of an Atmospheric Water Generator Device that utilizes geothermal heat rejection to condense vapor in the humid air. This device requires air parameters, water temperatures, and soil temperature variation to be measured to understand the system. This paper presents a capstone project that involved the design and implementation of an inexpensive data acquisition system that utilizes off-the-shelf components. Twelve parameters are collected: air inlet and outlet temperatures and humidity, water inlet and outlet temperatures, and soil temperature from two to eight feet at one-foot increments. The Raspberry Pi 4 with MCP-9600 chips, which runs on the I2C protocol, was used in this project. The I2C works by using an address system that allows the Raspberry Pi to distinguish between multiple sensors. The sensors, setup, circuitry, and code details are presented here. The system's performance, possible improvements, and the educational experience of an international capstone are also discussed.

Introduction

This system measured several key parameters for an atmospheric water generator (AWG) in Costa Rica. The AWG was part of a multi-year capstone project for three groups of engineering and Integrated Science and Technology (ISAT) students at James Madison University [1,2]. The data was critical for monitoring system performance and measuring the underground soil temperature and for future modeling of the system. However, the design laid out here could be used for other cases. The design of this system is also beneficial for undergraduate education, especially for class or capstone projects that require data acquisition of such parameters. The potential broad scope of applications and knowledge that is required includes schematics, wiring, programming, and networking. While this project was done for a specific purpose, there is no limit to the applications a system such as this one could perform.

The project described here was part of a multi-year capstone project that two previous teams worked on and then were taken over and expanded upon by a third group of three students. The capstone educational experience in the ISAT program at James Madison University consists of a four-course sequence (6 credits) in the 3rd and 4th academic years.

During the fall semester, students enroll in a one-credit hour course taught by two faculty members. The primary goal is to develop a research proposal on an engineering, science, or technological problem facing society. A second one-credit course is taken in the spring semester of the junior year when students do a deeper dive into the relevant literature, develop a detailed plan for executing the project during their senior year, and prepare a poster on their project that is presented at a symposium. In the senior year, students take a 2-credit hour course in the fall and spring, undertaking the research developed and proposed in their junior year. For the project described here, students performed an independent research project in Costa Rica. Through this

project, students analyze a technology-based problem, develop alternative solutions, recommend the best solution, and provide a written and oral technical report. As part of this international experience, students are able to demonstrate their ability to define and manage a project, identify goals, track and report progress, deliver results on time, and clearly report results. Specifically, students have:

1. Develop innovative solutions to significant, real-world problems.
2. Work with others, such as team members, project sponsors, and faculty members.
3. Situate their work in the relevant social context(s).
4. Develop and deliver a clear, convincing oral presentation and
5. Write an extensive professional report.

The team for this project consisted of three students (one female and two male students). Two of the three students had never been outside the United States, so cultural and extensive logistical preparations were made before the trip. The team decided to redesign and implement a system to address water scarcity, which required technical skills and creative problem-solving. As mentioned before, this was part of an international study abroad program. The initial plan was to install the system in the summer of 2020. The trip was canceled due to the pandemic. In the winter of 2021, the study abroad program was given the green light, and the project implementation period in Costa Rica was three weeks. This paper will focus on a specific portion of the capstone project which is the data acquisition system.

Methodology

The overall function of the data acquisition system (DAS) was to measure underground soil temperatures at six depths, measure two different humidity and temperature values, and measure two different water temperatures; Table 1 shows the breakdown. This was done using a Raspberry Pi 4 [3] because of its compact size, community support, header pins, and connectivity. The Raspberry Pi is connected to the sensors using the onboard GPIO (General Purpose Input Output) pins. Several different sensors were used, each with their respective benefits and weaknesses. We will highlight the MCP-9600 because of its I2C protocol.

Table 1: List of Materials for DAS

Sensor	Location	Parameter
1. AHT-20	Air Inlet	Temperature/Humidity
2. AHT-20	Air Outlet	Temperature/Humidity
3. MCP-9600	Water Inlet	Temperature
4. MCP-9600	Water Outlet	Temperature
5. MCP-9600	4 Foot	Temperature
6. MCP-9600	5 Foot	Temperature
7. MCP-9600	6 Foot	Temperature
8. MCP-9600	7 Foot	Temperature
9. MCP-9600	8 Foot	Temperature
10. MCP-9600	9 Foot	Temperature

In our use case, we had the Raspberry Pi sample the sensors once every minute and average the values over that hour. Once the average is collected, it is sent to Thing Speak [4], an online database run by MATLAB. This data can then be used for analysis anywhere with an internet connection.

The Raspberry Pi is the brain of the DAS. Raspberry Pi 4 was used in this capstone project and implemented in Costa Rica with the following specifications: 64Bit at 1.5GHz, Wireless Connectivity, Bluetooth, two HDMI-mini ports, and a 40 Pin GPIO Header. The Raspberry Pi is also 2.2 x 3.4 x 0.6 inches making it very small. The most crucial piece for the project is the 40 Pin GPIO Header. The Header is a series of 40 pins each capable of handling power, a ground, or data (digital or analog) to the Raspberry Pi. Figure 1 below shows the Raspberry Pi and labels what each pin does. For our purpose, the General-Purpose Input Output (GPIO) pins and the pins for the I2C protocol were used. It is important to note that these pins are fixed and with a few exceptions, the wiring must be connected to specific pins on the Header.

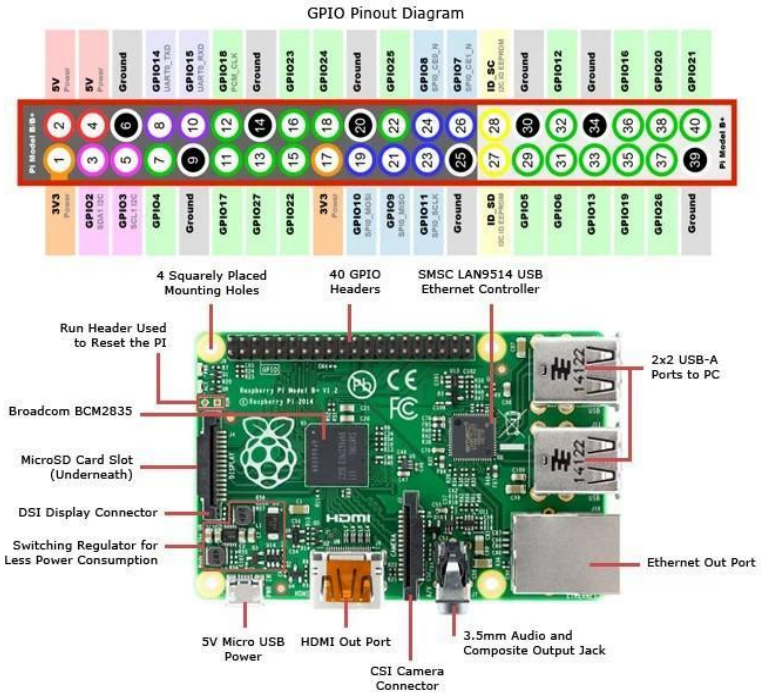

Figure 1: The figure above displays a Raspberry Pi 4 pinout diagram [5].

The reason the MCP-9600 [6] Chip was selected is that it runs on the I2C protocol. This allows the Raspberry Pi to run multiple identical chips in series while only using four of the headers on the Raspberry Pi, simplifying the design. The I2C protocol uses a 4-wire setup shown in Figure 2. Each chip is given power (VDD) and ground (GND). Serial Clock (SCK) functions to synchronize the timing of the chips to the Raspberry Pi. The Data Line (SDA) transmits the data to the Raspberry Pi.

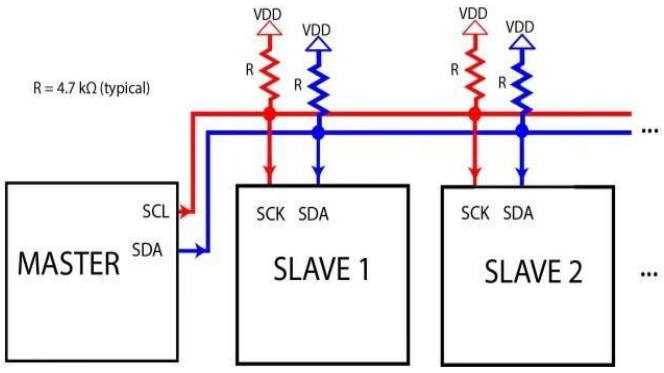

Figure 2: The figure above displays the wiring schematic of the I2C protocol.

The I2C works by using an address system that allows the Raspberry Pi to distinguish between multiple sensors. This protocol works like the way a switch works in networking. This function will enable us to wire all the chips in parallel, and as long as each has a unique address, the Raspberry Pi can call to and receive information from the sensors. The MCP-9600 comes with an address of 0X60. This address can be changed using the address pin on the chip with a voltage divider. Figure 3 shows the addressing protocol called out on the MCP-9600 datasheet [7]. The data sheet shows that this chip allows up to 8 unique addresses. There are ways to increase the number of chips further by using an I2C multiplexer which, in theory, could allow for an infinite string of chips to be connected.

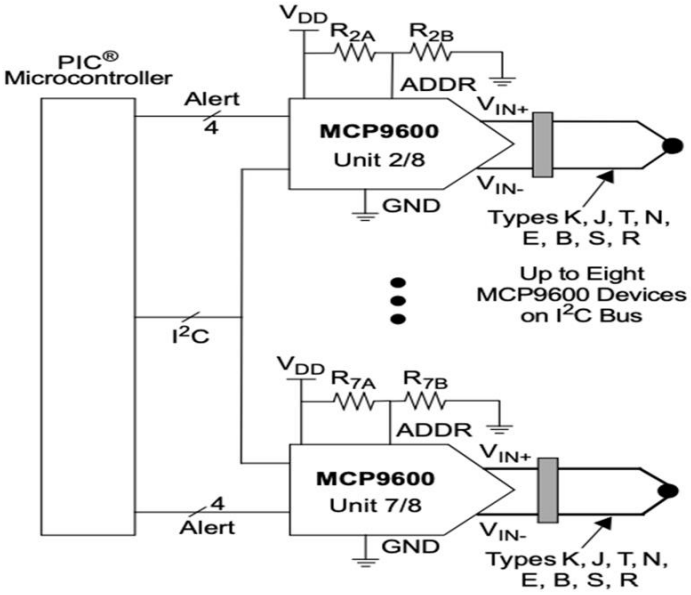

Figure 3: The figure above shows an I2C addressing diagram [8].

Using the information about wiring an I2C sensor to a Raspberry Pi and the address selection protocol of the MCP9600, there was a custom PCB designed using a free online resource called EasyEDA [9] and made through JLCPCB [10].

Figure 4 shows a schematic designed by one of the authors and uses different value resistors to change the address value of each sensor.

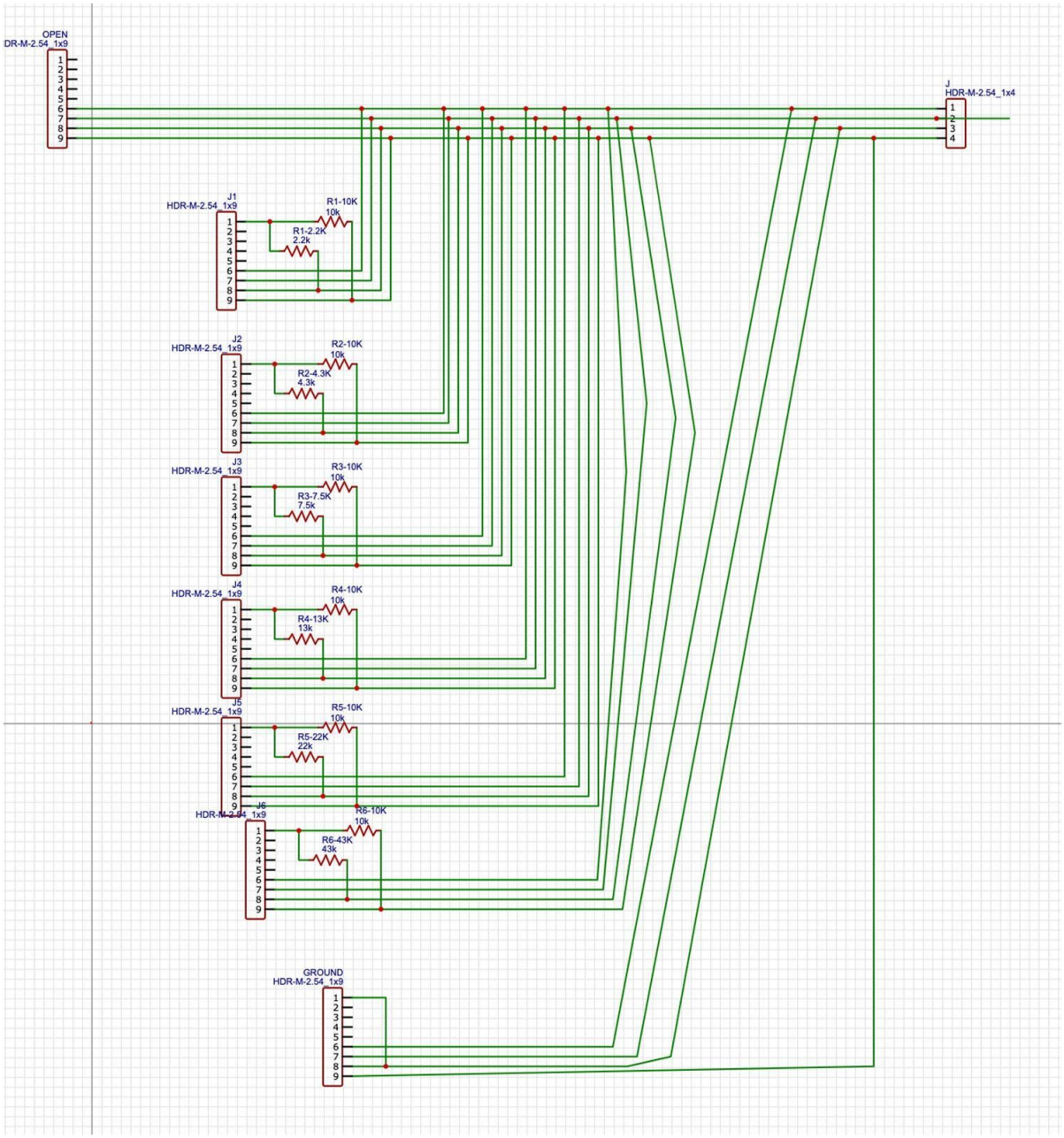

Figure 4: The figure above shows the PCB circuit.

The schematic in Figure 4 is then converted to a PCB gerber file which denotes the actual routes of the circuits, specifies the part numbers for the various ports and resistors, and much more. The gerber file contains all the necessary information for a manufacturer to make the part.

Figure 5 shows the PCB layout. Figure 6 shows the PCB 3D model and Figure 7 shows the PCB final product.

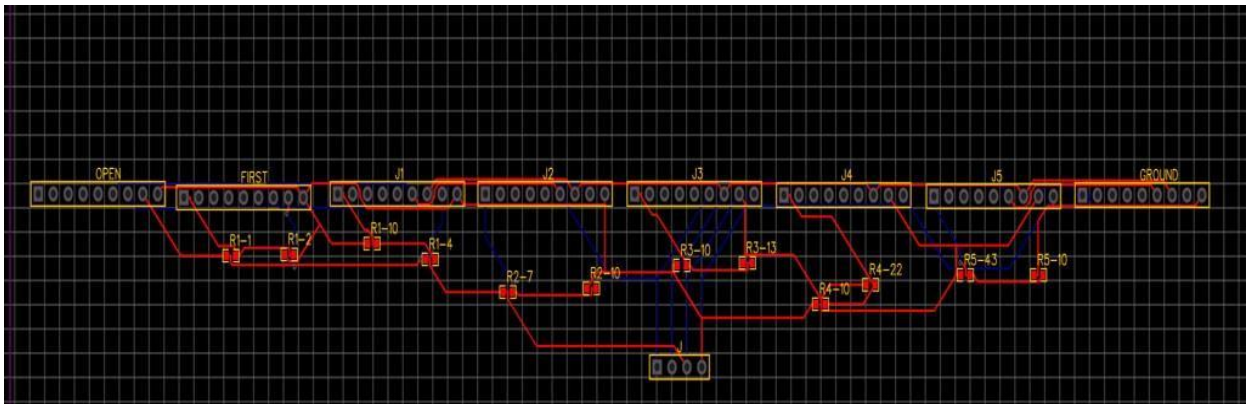

Figure 5: The figure above shows the PCB layout.

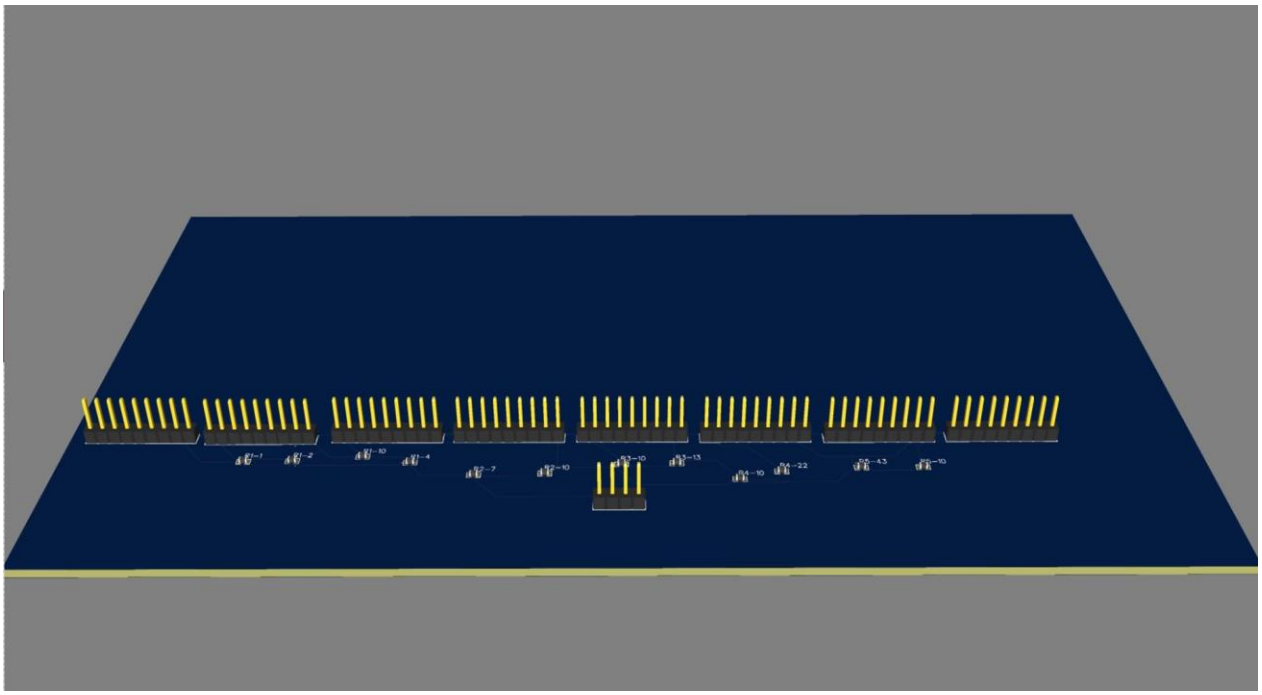

Figure 6: The figure above shows the PCB 3D model.

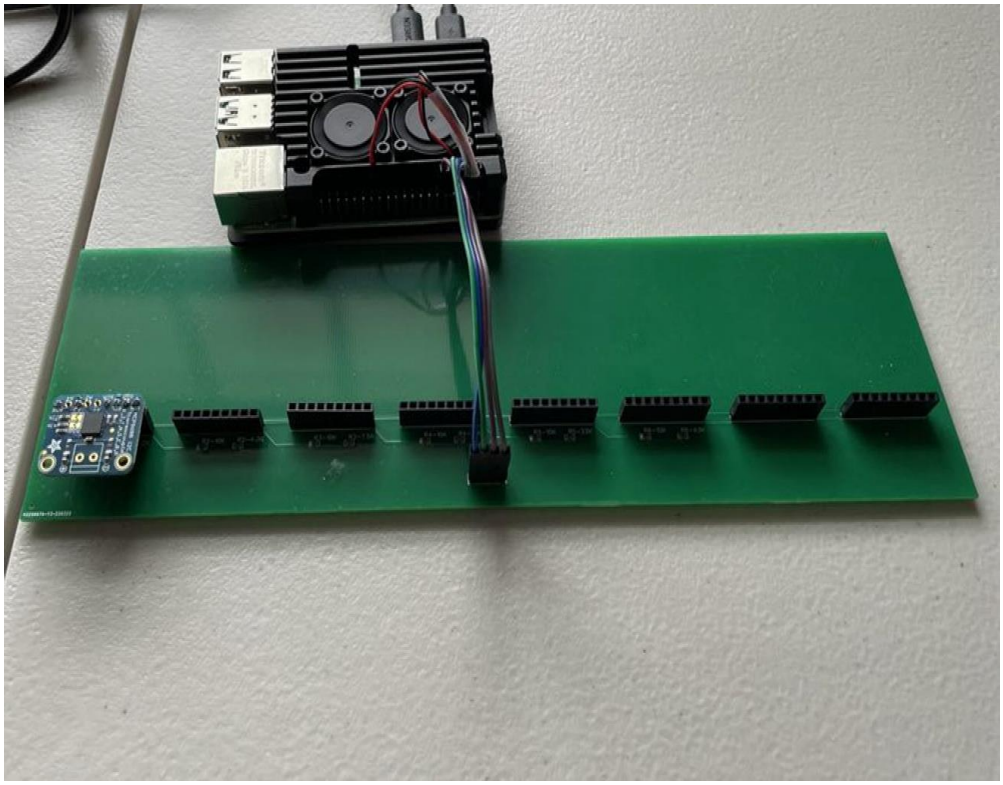

Figure 7: The figure above shows the PCB final product.

There were several key software programs that were used in an attempt at redundancy since this system wasn't going to be easily accessible. The first program that is activated is a program called watchdog [11]. This program watches the CPU and GPU to find when the Raspberry Pi is stalled out. When this happens, watchdog will automatically resets the system. This is the main defense against anything happening to the Raspberry Pi that could cause severe damage, such as overheating or some bug locking the team out from accessing the Raspberry Pi. There is a command that you can input into the terminal that detects devices on the I2C bus. The problem was that the Raspberry Pi only sees the chips a certain percentage of the time. This is due to a bug with the Raspberry Pi operating system and if this issue is faced, the team needs to change the I2C bus speed. This is done by editing the config.txt file. The last software downloaded was to have remote access to the Raspberry Pi from anywhere in the world. The software is called VNC Viewer, and the way that it works is to download the software onto both the Raspberry Pi and the computer used to access the Raspberry Pi. The software allows the user to access the screen of the Raspberry Pi anywhere in the world with an internet connection. The last step in using the Raspberry Pi is downloading the necessary packages to use the chip. Adafruit is an online store with information on setting up all its products. The Adafruit site [12] was used to download the necessary packages and code.

The code used is a mixture of one of the authors and examples found on the Adafruit website. The code can be found in the Appendix. The code is 150 lines and collects and stores data every minute for an hour, then creates an hourly average that it sends to a remote database. Thingspeak is an online database with easy integration with Python scripts.

Below is a summary of some of the key lines of Python code:

1) Lines 1-14: Imports the Necessary Packages for chips and other commands.
2) Lines 19-37: Establishes the name for each chip and its relative wiring on the Raspberry Pi.
3) Lines 40-53: Creates a variable that starts at zero. This is for the average data.
4) Lines 58-63: Calibration curves for the underground data.
5) Lines 65-76: Adds each data point to the variables on lines 40-53.
6) Lines 79-93: This is the data that prints once a minute on the screen of the Raspberry Pi, mainly used to make sure the data makes sense.
7) Lines 96-108: Calculates the average of the data for that hour and rounds the data to 2 decimal points.
8) Lines 111-122: This code sends the data to Thingspeak for later analysis.
9) Lines 139-150: Resets the variables on lines 40-53 to zero.

In our case, this system was implemented outdoors in Costa Rica. This required that the system have some level of protection from the elements, so it was encased in a water-tight box. The team also installed a keyboard and monitor in the box for potential troubleshooting if the need arose. The main worry was a change in internet settings, which wouldn't allow the Raspberry Pi to connect to the VNC Viewer hence losing access to the computer. Therefore, in the event of this happening, someone locally could easily correct the network settings returning access to the team.

Figure 8 shows a photo of the setup of Data Acquisition System (DAS). Figure 9 shows the final location of the DAS within the AWG system.

Figure 8: The figure above shows a photo of the setup of Data Acquisition System (DAS).

Figure 9: The figure above shows a photo of the final location of the DAS.

Discussion

This multi-year project was a great educational and learning experience for all stakeholders: faculty, students, resort management, and employees.

There are important considerations for students participating in a study abroad program, especially one focused on a specific project like the one in Costa Rica. To expand on some points:

1. Cultural preparation is essential for any study abroad program. It is important for students to understand the culture, customs, and expectations of the place they are visiting. This can help avoid misunderstandings and cultural insensitivity. Communicating with people from different cultures and languages is a valuable skill that can benefit students in many aspects of their lives. Learning how to appreciate and understand different cultures can help avoid misunderstandings and create more meaningful relationships.
2. This can also be applied to learning about different organizational cultures they may find themselves in during their careers.

3. Employability skills are also important for students to develop. Direct interaction with an international client can help students build skills in communication, negotiation, problem-solving, and project management.
4. Planning and preparation are key for any project, especially when resources are limited. Students who had to plan, purchase, and haul their materials likely gained valuable experience in project management and resource allocation.
5. Learning from mistakes and planning for multiple scenarios is a crucial skill in any field. Students who experienced shortcomings in machinery or materials likely learned how to problem-solve and adapt to unexpected situations.
6. Sustainable solutions are becoming increasingly important in many fields, and learning about sustainable practices in the resort can help students become more environmentally conscious and responsible.
7. Time management is essential in any project, and students who had to complete their work within a three-week time frame likely gained valuable experience in prioritizing tasks and managing their time effectively.
8. Weather constraints can be unpredictable and difficult to manage, but learning how to plan for potential weather-related issues can help students become more prepared and adaptable.
9. Understanding the priorities and constraints of the client is essential in any project. Students who learn how to work within these constraints can gain valuable experience in navigating complex professional relationships.

The students successfully connected concepts and found solutions to each problem they encountered. Despite the team's accomplishments, there were lessons learned that our team gained after the end of the project. This multi-disciplinary project challenged the team to broaden their focus to encompass every aspect, from technical to project management skills. In this section, we will cover the items the team succeeded in as well as items that could have been done differently.

On the technical aspects, the system outlined above is beneficial for several key reasons. The main one is the cost-effectiveness of the system. Data loggers of this type exist commercially, but they start at around 500 dollars for a unit with only two to three inputs. This system using 25-dollar chips can technically be daisy-chained using an I2C multiplexer. Another issue with commercial data loggers is the ability to network and remote access. The data collected using this device can be viewed and downloaded from anywhere with a network connection making the system very flexible. Also, the flexibility when using the Raspberry Pi is unparalleled. For instance, in our use case, we needed to know the temperature of the water going out of the system. If one wanted to change out the temperature sensor for a flow rate sensor, this would be easy with our system. The system does have some downsides when compared to a commercial grade system, mainly the accuracy of the system. The chips themselves have a margin of error of around two degrees Celsius, which depending on the application, could be an issue. Another issue, especially in a place like Costa Rica, is Raspberry Pi's tendency to run hot. This can cause worsened performance and, in some cases, lead to outright failure. The team did add a fan and heat sink in an attempt to lessen this effect. This decreased the tendency to overheat but never eradicated the issue. The format outlined above calls for the designer's creativity to create a

system that can fit perfectly to the requirements of a project and the Raspberry Pi gives you the tools to create a perfect fit. This design is from the ground up and encompasses a wide range of techniques and knowledge in order to pull off. Still, this knowledge is easily accessible, allowing anyone from beginners to experts to learn something and have a system that fits their needs. A project like this requires skills in programming, circuitry design, implementation, knowledge of software requirements, hardware specifications, and networking. Most importantly, and sometimes painfully, integration of all these subsystems. However, there is community support every step of the way allowing for self-learning once the fundamentals are understood. Online forums and guides are plentiful, from Python to the hardware allowing most problems to be easily solved. The wide gambit of information required means that this system can be perfect as an educational tool for a student or a professional system in the workplace.

Conclusion

The data acquisition system for an AWG in Costa Rica measures and monitors several key parameters. This cost-effective system provides the ability to network and remote access from anywhere with a network connection. This system uses over-the-shelf components and is easily accessible to beginners and experts. Implementing this project took skills, including programming, circuitry design, and more. This project also serves as an educational tool, providing students with hands-on technical skills, self-learning, and community support.

Participating in an international capstone project can provide students with a unique and valuable educational experience that can help them develop important skills and prepare them for a globalized workforce.

Acknowledgments

We want to give thanks to everyone who helped our team along our capstone journey. Starting with the first group that established this multi-year project: Kiara Pontious, Brad Weidner, Nima Guerin, Andrew Dates, and Dr. Olga Pierrakos. Next, we thank the second group who helped transition us into taking over this project: Declan Tyranski, Devin Simons, and Zachary High. We also want to acknowledge Brian Tang for his dedicated and meticulous work in designing and building the AWG. Maria Liu for setting up the weather station and processing the EES modeling. In addition, we want to acknowledge our lab and electronics technicians Mark Showalter and Joe Rudmin for their time and effort in assisting us as well as James Madison University, The College of Integrated Science and Engineering, and the ISAT Department for supporting us as we implemented our project during a study abroad trip in Costa Rica. Many thanks go to Rod MacDonald for his help in editing this paper. Last but not least, we are indebted and grateful to Boris Gordienko, Joe Calderon, and the entire Punta Leona Hotel and Resort workers for working with us to make this project possible.

References

[1] Pontious, K., Weidner,B., Guerin, N., Dates,v A., Pierrakos, O., & Altaii, K. Design of an atmospheric water generator: Harvesting water out of thin air. In 2016 IEEE Systems and Information Engineering Design Symposium (SIEDS), 6-11, 2016.

[2] Simons, D. P., Tyranski, D. R., High, Z. D., & Altaii, K. Water Out of Thin Air: Designing an Atmospheric Water Generator to Address Water Scarcity. Retrieved from IEEE: https://doi.org/10.1109/SIEDS52267.2021.9483753, 2021

[3] https://www.adafruit.com/product/4564

[4] https://thingspeak.com

[5] https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/

[6] https://www.adafruit.com/product/4101

[7] https://www.microchip.com/en-us/product/MCP9600

[8] https://ww1.microchip.com/downloads/en/DeviceDoc/MCP960X-Data-Sheet-20005426.pdf

[9] https://easyeda.com/

[10] https://jlcpcb.com/?from=VGB&gclid=CjwKCAiAuaKfBhBtEiwAht6H76VOCfcYvTQMPLw-v3PwBQ7-DSosBLVOsGBir8QxekVk-YdTMRXoMhoCHS8QAvD_BwE

[11] https://diode.io/raspberry%20pi/running-forever-with-the-raspberry-pi-hardware-watchdog-20202

[12] https://learn.adafruit.com/adafruit-mcp9600-i2c-thermocouple-amplifier/python-circuitpython

Appendix

```python
import board
import busio
import adafruit_mcp9600
import time
import datetime
import adafruit_ahtx0
import digitalio
import adafruit_max31855
import adafruit_max31856
import random
import urllib.request
import requests
import threading
from adafruit_htu21d import HTU21D

i2c = busio.I2C(board.SCL, board.SDA, frequency=100000)
spi = board.SPI()

#Water Inlet Temperature
cs_inlet = digitalio.DigitalInOut(board.D5)
water_inlet = adafruit_max31855.MAX31855(spi, cs_inlet)

#Water Outlet Temperature
cs_outlet = digitalio.DigitalInOut(board.D6)
water_outlet = adafruit_max31856.MAX31856(spi, cs_outlet)

#Temperature and Humidity Sensor
air_inlet = adafruit_ahtx0.AHTx0(i2c,address= 0X38)
air_outlet = HTU21D(i2c,address = 0X40)

#UnderGround
mcp_0 = adafruit_mcp9600.MCP9600(i2c, address= 0X60)
mcp_2 = adafruit_mcp9600.MCP9600(i2c, address= 0X62)
mcp_3 = adafruit_mcp9600.MCP9600(i2c, address= 0X63)
mcp_5 = adafruit_mcp9600.MCP9600(i2c, address= 0X65)
mcp_6 = adafruit_mcp9600.MCP9600(i2c, address= 0X66)
mcp_7 = adafruit_mcp9600.MCP9600(i2c, address= 0X67)

#Begin Variable Averege For Underground
five = 0
six = 0
seven = 0
eight = 0
nine = 0
ten = 0

#Begin Varible For Averege Above Ground
enter_temp = 0
enter_humid = 0
exit_temp = 0
exit_humid = 0
fluid_enter = 0
fluid_exit = 0
while True:
    for x in range(60):
        time.sleep(59)
        now = datetime.datetime.now()
        five_ft= round(((mcp_0.temperature*.668)+5.84),2)
        six_ft= round(((mcp_2.temperature*.666)+6.666),2)
        seven_ft= round(((mcp_3.temperature*.682)+5.5),2)
        eight_ft= round(((mcp_5.temperature*.663)+7.2),2)
        nine_ft= round(((mcp_6.temperature*.663)+7.2),2)
        ten_ft= round(((mcp_7.temperature*.684)+4.875),2)

        five += five_ft
        six += six_ft
        seven += seven_ft
        eight += eight_ft
        nine += nine_ft
        ten += ten_ft
        enter_temp += air_inlet.temperature
        enter_humid += air_inlet.relative_humidity
        exit_temp += air_outlet.temperature
        exit_humid += air_outlet.relative_humidity
        fluid_enter += water_inlet.temperature
        fluid_exit += water_inlet.temperature

        date_time = (now.strftime("Date:%Y/%m/%d Time:%H:%M:%S"))
```

```
            print ("Data Collection 5ucces f ull , Collected at: "+da t e _t i me)
            pri nt ( "" )
            pri nt ( " Run Number" , x+1 )
            pri nt ( " UnderGround Temperature: \n SFt:{} C \n 6Ft:{} C \ n 7Ft:{} C \n 8Ft:{} C \n 9Ft:{} C\n 10Ft :{} C " . fo rm at (fi ve_f t , si x_ft , seven _ft , ei ght _ft , ni ne _ft , t en _ft ) )
            pri nt ( "" )
            print ("Inl et   Tme  perat   ure   and  Humi dit  y:"  )
            print (" Temperature: %0 .1 f C % a i r _ i nl e t . t empera t ure)
            pri nt ( " Humidity: %0 .lf %%" % a i r _i nl et. r-e l a ti ve_humi dit y)
            pri nt ( "" )
            print ("Out let Temperat ure and Humi dit y:" )
            pri  nt  ( "    Tme  perat   ure:    %0 .lf    C    %  a i  r _ o ut  let.   t empera  t ur-e)
            pri nt ( " Humidity: %0.lf %%" % a i r _ o ut l e t . re l a t i ve _humi di t y)
            pri nt ( "" )
            print ("Inl et a nd Outlet Water Temperat ure :" )
            pri nt ( " I nl et :{} C \n Out l et : { } C' . f or mat (w a t er _ i nl et .t emperat ure, round ((wat er _ ou t l et . t empera t ure), 2 ) ) )

        #The 3 is the Number of Data point and needs to be accurate
        fi ve_avg=r-o und ( (fi ve/ (x +l ) ), 2 )
        si x_avg = rou nd( (s i x/ (x +l )), 2 )
        seven  _ avg = rou nd( (seven  / (x +l ) ) , 2 )
        eight  _ avg = rou nd( ( ei ght / (x +l ) ) , 2 )
        ni ne_avg =r-o und ( (n i ne / (x +l )), 2 )
        t en_avg = rou nd( (t en / (x +l )), 2 )

        ent er _t emp_avg =round ( (en t er _t emp/ (x +l ) ) , 2 )
        en t er _ humi d_ avg = round ( ( en t er _ humid / (x +l )), 2 )
        exit _t emp_avg = round (( exi t _t emp/ (x +l ) ), 2 )
        exit _humi d_avg =round ( (exi t _ humi d / (x +l ) ) , 2 )
        fl ui d_ ent er _ avg = round ( (fl ui d_ ent er / (x +l )), 2 )
        fl ui d_ ex i t_ avg =round ( (fl ui d_ exi t / (x +l ) ) , 2 )

            #Below Ground
    def   t hi ngs peak () :
            #timerbelow is how often to upload in seconds
            URL = · htt ps : / / a pi . t hi ngspe ak. com/ updat e ?a pi _ key = ·
            KEY= ' '
            #field 1 and 2 are created on thing speak and the values come
            #from val 1 and val 2
            HEADE R= ' &fiel dl ={ }&field2={}&f i el d3 ={ } &fi e ld4={}&field5 ={ }&f i e l d6 ={} · . format (fi ve _avg, si x_ av g, seven _ avg,  ei ght _ avg, ni ne _ avg , t en _ avg}
            new_URL =URL+K EY+HEADER
            new_UR L=UR L+K E Y+HEADER
            v=urll i b. request. urlopen (new _ UR L )
            pri nt (v}
    if       name == _ m a i n_ · :
            t hi ngspea k ()

            #Above Ground
    def   t hi ngspea k () :
            ttimerbelow is how often to upload in seconds
            URL =· ht t ps : / /api. t hi ngspeak . com/ updat e? a pi _key='
            KE Y= '
            #field 1 and 2 are created on thing speak and the values come
            #from val 1 and val 2
            HEADER = ' &fieldl={}&field2={}&field3={}&field4={}&f i el d5= {} &fi e l d6={}' . fo rm at (en t er _ tem p_avg , en t er _ humi d_ avg,exi t _t emp_ avg, ex i t _humid _avg , fl ui d_ ent er _ avg, fl ui d_ ex i t _ avg)
            new_ UR L=UR L+K EY+HEADER
            v=ur llib. request. urlopen (new _ UR L )
            pri nt (v)
            pri nt ( " Average Data has been se nt \ n" , (x +l ), " Runs Were Ave r aged" )
            pri nt ( " For Ref er e nce The Five Foot Aver a ge was: ", f i ve_avg)
    if       name == m a i n
            t hi ngspea k ()
        f ive = 0
        six = 0
        seven = 0
        ei gh t = 0
        nine = 0
        ten = 0
        en t er _t emp = e
        en t er _ humi d = 0
        exit  _t em p = 0
        exi t _ humid = e
        fl ui d_en t er = 0
        fl ui d_exit = 0
```