



INSTITUTE *of*
TECHNOLOGY

CARLOW

Institiúid Teicneolaíochta Cheatharlach

Secure SCADA/IoT System to track live data

Final Report

Student: Neil Kane

Student Number: C00242418

Supervisor: James Egan

Abstract

I think the biggest upside to the final year project was the wide variety of learning. We had the opportunity to learn new skills, implement what we learned through college and test ourselves of our knowledge. All the issues that arose throughout the year I feel like I learned from and took something away from it. Although there was bumps in my project along the way, I was happy to try and overcome these issues. I didn't get to implement everything I intended to but I still learned a great deal from that and understand why I didn't achieve everything I wanted. With this being the case, I have to say I am still happy with what I achieved throughout the year and what I learned along the way.

My project uses Flask webserver for the SCADA dashboard. Python was the main language used as well as some HTML for the dashboard. I used the Sunfounder sensor kit for the sensors used throughout this project. I also used two Raspberry Pi's and a Pi camera. As I used two Pi's I will refer to the sensor Pi as Pi1 and the camera Pi as Pi2. Each Pi has its own system and feature and perform different tasks, although they would work together in a combined real world scenario.

Acknowledgements

I would like to thank all the lectures involved in the projects for any help I received throughout the year and the feedback received from my presentations.

I would especially like to thank my supervisor James Egan who helped guide me through the year and gave me great advice along the way.

Table Of Contents

| | |
|--|----|
| Abstract | 1 |
| Acknowledgements | 2 |
| Table Of Contents | 3 |
| Outline of Working Project..... | 4 |
| Raspberry Pi-1 | 4 |
| Sensors | 4 |
| Flask | 6 |
| Raspberry Pi-2..... | 7 |
| Camera..... | 7 |
| Security..... | 8 |
| Combing Raspberry Pi's | 9 |
| General Issues | 10 |
| Problems Encountered | 10 |
| What I Achieved | 10 |
| Security..... | 11 |
| Facial Recognition | 11 |
| Flask | 11 |
| What I did not Achieve | 11 |
| What I learned..... | 12 |
| What I would do differently if starting over | 12 |
| Update..... | 13 |
| Remote Desktop and Locked Out..... | 13 |
| Libraries | 14 |
| Flask Libraries..... | 15 |
| Conclusion | 16 |

Outline of Working Project

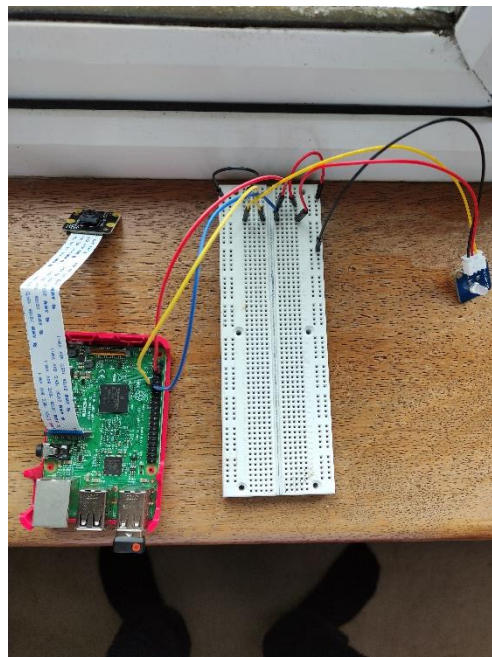
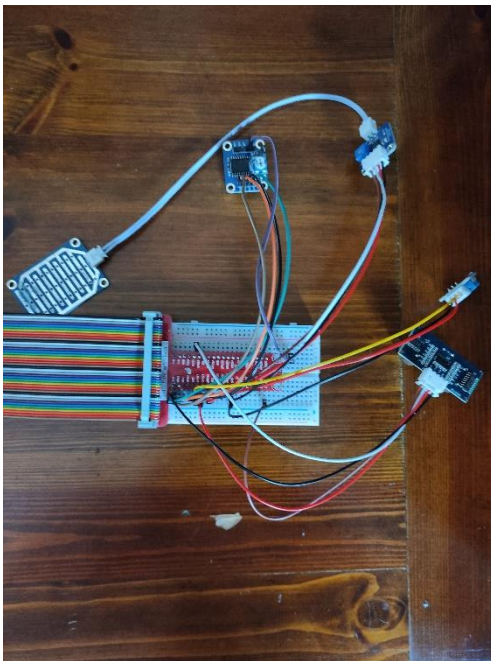
In this section I will briefly talk about how the project works together and show screenshots along the way of the system and sensors connected.

Raspberry Pi-1

Sensors

The first Pi system uses the sensors from the sunfounder sensor kit. To use these sensors Sunfounder has created a guide which leads to their github, and we can get the python code for the sensors. I had tested a range of individual sensors in my research but hadn't created a system in which I connected multiple random sensors together. I used the humidity sensor which reads the humidity and temperature. I also tested out the rain sensor which is a simple program that tells us if its "raining" or "not raining". This sensor had to be put into a cup of water to register the "raining" reading.

The image on the left is the Pi with the Sunfounder sensors including humidity, ultrasonic-range and rain detector. The image on the right has the camera connected and the LED light connected to the breadboard.



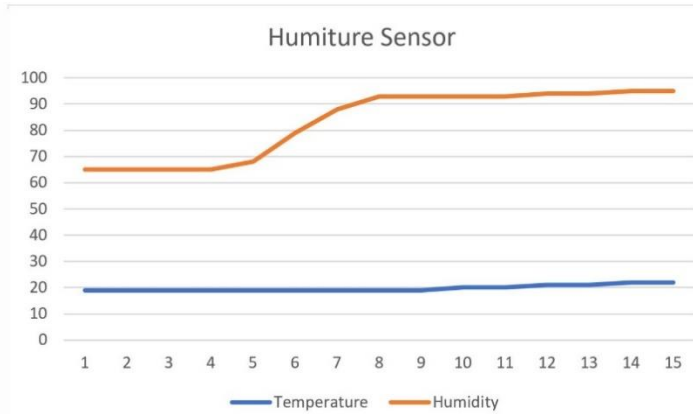
As I wanted to somehow monitor the system I also included reading the CPU temperature, so we could monitor if the system was working to heavy.

My plan was to send this data to a CSV file which is basically an excel file. Through Flask which hosts the webserver, I could then import the data from the CSV file and using the pandas and matplotlib libraries to create the charts monitoring the readings. This caused problems which I will discuss below. Instead of this I had to

just upload screenshots to demonstrate what I wanted the dashboard to look like. I ran the Flask project on both my Windows machine and the Raspberry Pi, and the Pi would always freeze, so I decided to stick with the windows machine.

The below image shows the readings from the humiture sensor alongside a graph with the data.

| | |
|-------------------|---------------|
| Temperature: 19°C | Humidity: 65% |
| Temperature: 19°C | Humidity: 65% |
| Temperature: 19°C | Humidity: 65% |
| Temperature: 19°C | Humidity: 65% |
| Temperature: 19°C | Humidity: 68% |
| Temperature: 19°C | Humidity: 79% |
| Temperature: 19°C | Humidity: 88% |
| Temperature: 19°C | Humidity: 93% |
| Temperature: 19°C | Humidity: 93% |
| Temperature: 20°C | Humidity: 93% |
| Temperature: 20°C | Humidity: 93% |
| Temperature: 21°C | Humidity: 94% |
| Temperature: 21°C | Humidity: 94% |
| Temperature: 22°C | Humidity: 95% |
| Temperature: 22°C | Humidity: 95% |



The below image shows the ultrasonic range sensor with the readings on the left and data inputted in a chart on the right.

```
>>> %Run ultrasonic_ranging.py
Ultrasonic range
203.81498336791992 cm

Ultrasonic range
18.36872100830078 cm

Ultrasonic range
41.70656204223633 cm

Ultrasonic range
42.890071868896484 cm

Ultrasonic range
13.549566260920090 cm

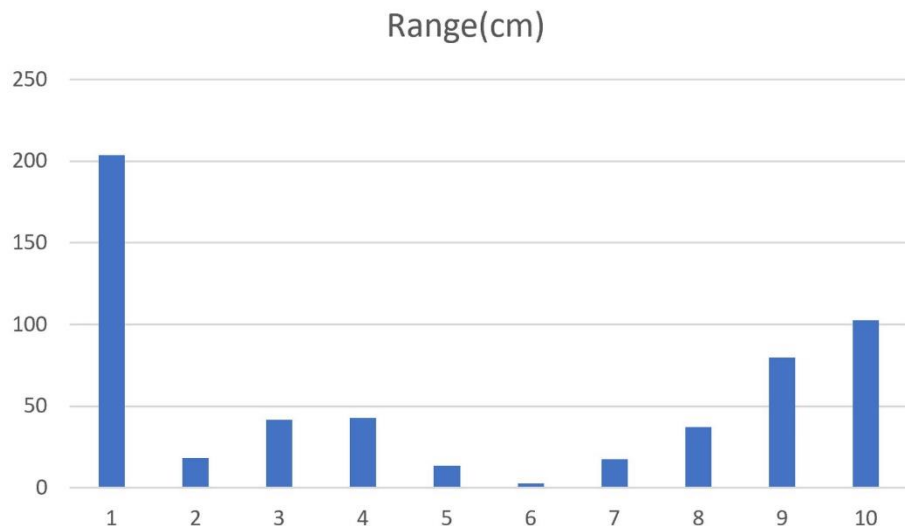
Ultrasonic range
2.889871597290039 cm

Ultrasonic range
17.60260211364746 cm

Ultrasonic range
37.34946250915527 cm

Ultrasonic range
79.63156700134277 cm

Ultrasonic range
102.62084007263184 cm
```



Flask

As stated above I ran the Flask project on both my Windows machine and the Raspberry Pi, and the Pi would always freeze, so I decided to stick with the windows machine. I set up Flask on Visual Studio code and using the terminal installed the necessary libraries, like Flask, Flask_Login and SQL_Alchemy. I also used Jinja to add python to individual HTML pages. These all installed no problem and I created a sign up, login and home page. The home page is the dashboard which shows the user the graphs from the readings of the sensors. The Sign-up page has conditions that must be met. The email has to be an unused email. The passwords have to match and contain at least 8 characters. If these conditions are not met, the user will be prompted with a message declaring passwords don't match, email already used etc. The passwords are hashed. Once signed in the user can see the data.

Login Sign Up

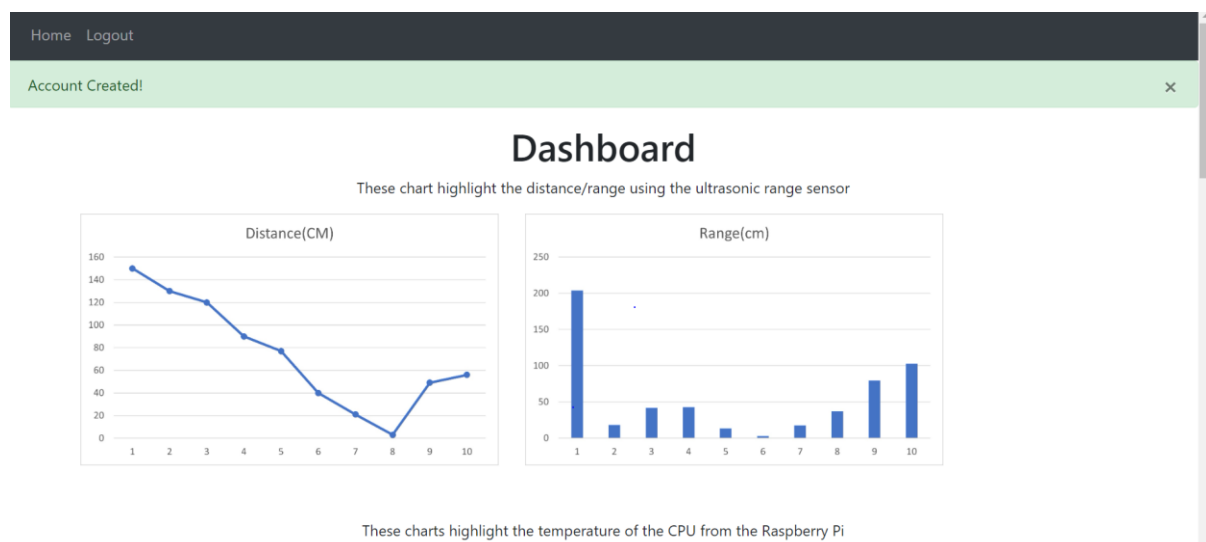
Sign Up

Email Address

First Name

Password

Confirm Password



Raspberry Pi-2

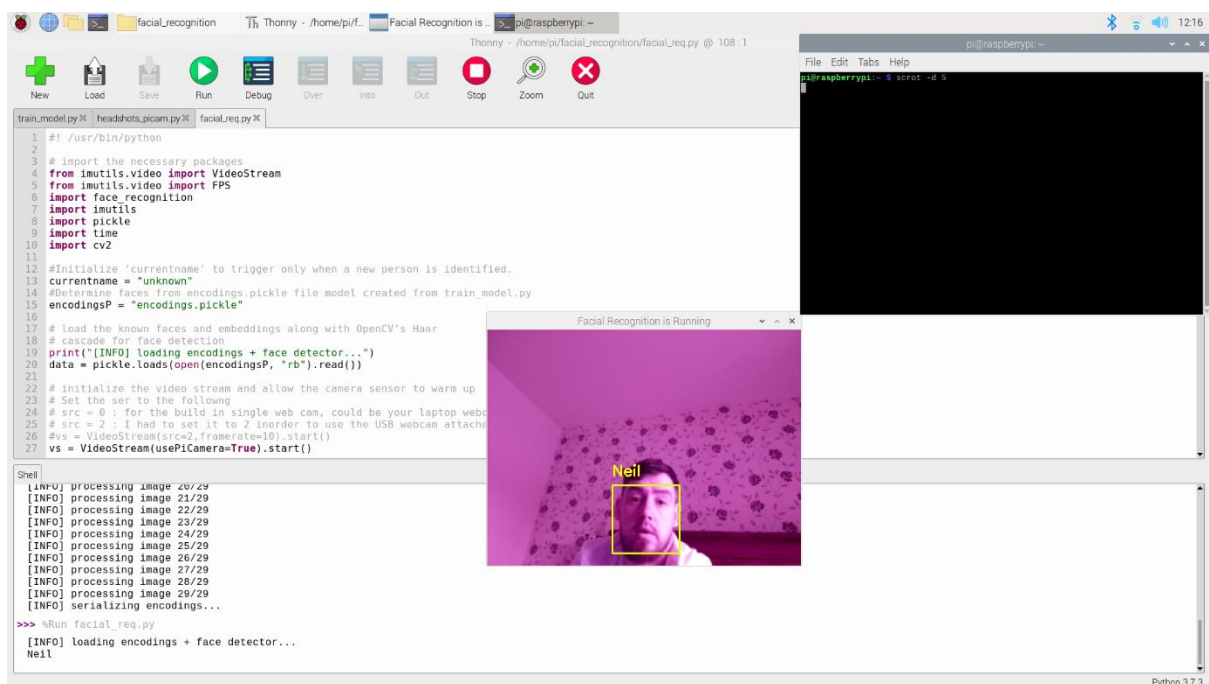
Camera

I will now discuss the Raspberry Pi2 which is the camera Pi.

This feature got added later as the energy monitor didn't work out which I will discuss further down. I had access to a camera and wanted to implement it some way. I came across a facial recognition program created by Caroline Dunn, which I implemented on the Pi. I then added an LED sensor that upon a successful recognition would turn green. This could represent any sensor as it works the same way. Once a person inputted into the system and recognised the sensor will take effect. This could be a range of sensors, I just used the LED to demonstrate.

The system works as follows, we teach the program about a user by creating a folder calling it by our name. We then run a part of the code which will take photos of us using the spacebar, we move around and get different angles of our face. When we run the program it then uses these photos as a comparison to identify the user. I just inputted my own image as a demo and then showed the camera a picture from my phone of someone else which it did not recognise, proving the system worked. The frame rate of the camera is extremely low but I think that was to do with the version of the Pi, there is newer Pi's out with greater computing power that seems to run these type of programs smoother.

The camera I had access to was a NoIR, which stands for "No Infrared Filter", this leaves a purple hue over the picture as the camera was designed for use at night time. I still used this camera as it still works and it was just to demonstrate that it all works. The Open-CV library was used to perform the above, it is a tool used for real time computer vision. I also used the facial_recognition library and imutils which is used for image processing.



```
train_model.py% | headshots_picam.py% | facial_req.py%  
1 #!/usr/bin/python  
2  
3 # import the necessary packages  
4 from imutils.video import VideoStream  
5 from imutils.video import FPS  
6 import face_recognition  
7 import imutils  
8 import pickle  
9 import time  
10 import cv2  
11  
12 #Initialize 'currentname' to trigger only when a new person is identified.  
13 currentname = "unknown"  
14 #Obtainme faces from encodings.pickle file model created from train_model.py  
15 encodingsP = "encodings.pickle"  
16  
17 # load the known faces and embeddings along with OpenCV's Haar  
18 # cascade for face detection  
19 print("[INFO] loading encodings + face detector...")  
20 data = pickle.loads(open(encodingsP, "rb").read())  
21  
22 # initialize the video stream and allow the camera sensor to warm up  
23 # Set the ser to the following  
24 # src = 0 : for the build in single web cam, could be your laptop webcam  
25 # src = 2 : I had to set it to 2 in order to use the USB webcam attached  
26 #vs = VideoStream(src=2, framerate=10).start()  
27 vs = VideoStream(usePiCamera=True).start()  
  
Shell  
[INFO] processing image 20/29  
[INFO] processing image 21/29  
[INFO] processing image 22/29  
[INFO] processing image 23/29  
[INFO] processing image 24/29  
[INFO] processing image 25/29  
[INFO] processing image 26/29  
[INFO] processing image 27/29  
[INFO] processing image 28/29  
[INFO] processing image 28/29  
[INFO] serializing encodings...  
  
>>> %Run facial_req.py  
[INFO] loading encodings + face detector...  
Neil
```


Security

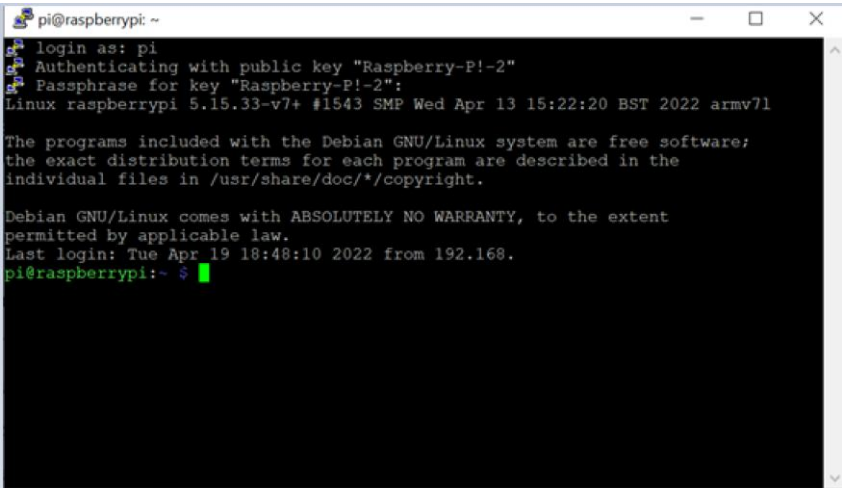
I secured the Pi2 in a number of ways following my research document. The first thing I done was changed the password on the Pi. I installed Fail2Ban which helps fight against Brute Force attacks. We set the number of sign in attempts from an IP address and after this set number is hit, the IP is banned for a duration of our choosing. I chose 5 attempts in 10 minutes, would ban you for 10 minutes. This will slow an attacker trying to brute force by a considerable amount. We can change this by opening and editing the config file with : `sudo nano /etc/fail2ban/jail.conf` and setting the limits to more or less time.

I installed the UFW firewall. This allows us to create a basic IP table, to deny or allow traffic to and from the Pi. I only allowed SSH from my windows machine and denied all other IP addresses.

We can also set ports to allow traffic through, and close access to unused features. This includes HTTP.

I changed the standard port from port 22 for SSH, as this is a known port and will be the targeted port.

As I needed SSH on the have remote access, I also set up SSH Authentication. Using Puttygen I generated a public and private key. One of the keys is put on the raspberry pi and when we try SSH we no longer need a password as we are authenticated using these keys. I used Putty to SSH and here is where we type in the port number, the Pi's IP address and import the private key that is stored in a safe location on the computer. I also added a passphrase as an extra layer of security, although this is not required. We can see below on the second line Authenticating with public key "Raspberry-P1-2" and the third line asks for a passphrase.

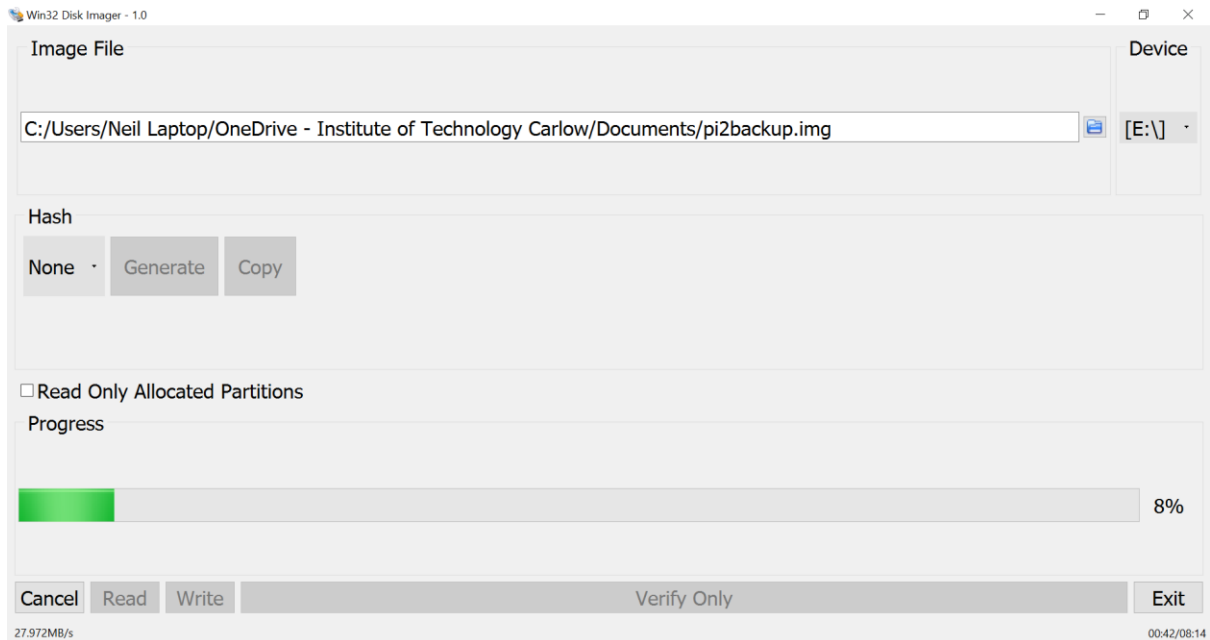
A terminal window titled 'pi@raspberrypi: ~' showing the process of logging in via SSH. The output includes: 'login as: pi', 'Authenticating with public key "Raspberry-P1-2"', 'Passphrase for key "Raspberry-P1-2":', system information 'Linux raspberrypi 5.15.33-v7+ #1543 SMP Wed Apr 13 15:22:20 BST 2022 armv7l', a copyright notice for Debian GNU/Linux, a warranty disclaimer, and the last login time 'Last login: Tue Apr 19 18:48:10 2022 from 192.168.'. The prompt 'pi@raspberrypi:~ \$' is visible at the bottom.

```
pi@raspberrypi: ~
login as: pi
Authenticating with public key "Raspberry-P1-2"
Passphrase for key "Raspberry-P1-2":
Linux raspberrypi 5.15.33-v7+ #1543 SMP Wed Apr 13 15:22:20 BST 2022 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Apr 19 18:48:10 2022 from 192.168.
pi@raspberrypi:~ $
```

We should also backup our raspberry pi, this can be done a few ways but I chose to insert the SD card from the Pi into my laptop and use a program called Win32 Disk Imager. We can read the SD card to a folder on your computer naming the file with the img extension. If we need to use this backup, we use the same program and write back on an SD card



For the purpose of creating a demo video, I also installed XRDP on the Pi. This allows us to use Remote Desktop Connection from our Windows machine so we can remote in and see the screen of the Pi not just the command line. This installed without any issues on Pi2 but on Pi1 it seemed to cause problems, which I will discuss below.

Combing Raspberry Pi's

I believe the features of the two Pi's demonstrate a system that could be used in a real world situation for an industrial industry. If we had a SCADA system and we only wanted certain users access to the machine we could implement the facial recognition to allow access to a room. I wasn't originally planning on using a camera but after using it, I believe it's a great extension to use with a Raspberry Pi and can be incorporated into a number of ways including a live feed.

General Issues

This section explain problems that arose through the final year project. I will talk about what I achieved and things I didn't achieve. I will discuss what I learned and if starting over what I would do differently.

Problems Encountered

I have to admit I encountered a range of problems, some I overcame and other I had to change up the project due to these problems. As mentioned in the UPDATE section on my research document. The original idea was to create a energy monitor. Through more research and attempted implementation, this was not a viable option. I asked an electrical engineer to look at the idea I was going with. Based off projects and reports I had researched, I had he idea of what needed to be done. When the engineer had a look I was informed that the setup was dangerous as it was to gain information from live cables. He had run a test to see it working and had informed me with the equipment I had the readings were so small that it was not worth attempting. The time spent on energy monitors took up a good chunk of valuable time that could have been used elsewhere. This is when I chose to include the camera.

I struggled with installing some libraries on the raspberry Pi. As I was learning about Pythion and Raspberry Pi's as the project progressed, I think this led to many delays with features I had hoped to include. A simple early problem was familiarising myself with using sensors. In first year we worked with Arduinos and sensors and that was the only time I had ever used them. The kit I was using had a great guide book and with just practice and time I became comfortable working with the sensors.

Most of my problems came from installing libraries. I had to find a work around on numerous occasions for a simple library install. This happened for a number of reasons, including the python version I was using. A big fix to many of the issues was to manually upgrade the Python version. I had performed this and it said it was successful but my version didn't change from Python 3.7. This created some problems as it didn't allow me to use the libraries I needed to finalize the project the way I had hoped.

Once this was done I still had the second Raspberry Pi, so I started looking into other projects, I came across a facial recognition program and this became the replacement for the energy monitor. This still tied into security which included the camera for actual security and securing the Pi. The time spent on gathering the parts for the energy monitor and reading similar documents was valuable time that could have been better put to use.

What I Achieved

I think overall I achieved what I set out to do. I created a webserver using flask with a user sign up and login page. This page would allow access to the dashboard. I got multiple sensors working on a Raspberry Pi, which I was worried about as initially I

only connected one sensor at a time. I also successfully had two Raspberry Pi's up and running, I'm not sure if it would have made more sense to try and do everything on the one but with limited capabilities of a Pi, I thought it best to use two. I added security features to one of the Pi's and had a wide range of different sensors working

Security

I secured one of the Pi's for SSH using public and private keys for SSH connection. This was done using PuttyGen, it gives us a public and private key. We store the private key on our windows machine and the public key on our raspberry pi. When we SSH using Putty we then copy in our Private key and when we connect we will authenticate with the keys. I also added a passphrase just to show you could have the extra layer of security. I also installed security features on the Camera Pi. This included changing the port number as port 22 is the standard used and anyone trying to gain access would use this port. I installed Fail2Ban which is a tool to prevent brute force attacks. I also installed a firewall on this Pi and set it that only my IP address can SSH into the Pi.

Facial Recognition

I installed a facial recognition program that highlights the user with their name if they have been implemented in the system which if successful turns an LED light green. Although I didn't have the sensor, there is a locking system that could have been used here and my LED is just representing any sensor that could have been used upon successful recognition. I managed to get one program sending the input to a csv file, this would have allowed me to share the readings to create a live chart had I had the correct libraries installed in time. The facial recognition is from GitHub and many users have tried to tweak it or add their own bit to it. The installation for this caused plenty of problems, as some of the libraries would get stuck on 99%. I spent 12 hours trying to just install some of the libraries for this. There was also a few different suggestions of how to install certain libraries, with some needing "sudo" and others recommending something else.

Flask

I installed Flask on to my windows machine and found it to run better. I created the webserver and made the dashboard which allows for user signup and login. Once logged in the user is directed to the dashboard.

What I did not Achieve

I did not get to implement the full scale SCADA system I had hoped to achieve this was down to two things, timing being a big one and the struggle of installing the correct libraries needed to implement such a system. I needed to install pandas to work with a range of other libraries. I planned on using matplotlib and csv files. Although there is features missing from my finished project I think most of this came down to timing and how I approached the project. I definitely thought I had more time to work on certain aspects but I got side tracked with certain features and in turn this led to other features being neglected. As stated above I also didn't get to implement the energy monitor, this didn't feel like something I didn't get to achieve as much as wasted resources. The fact that I could add the camera feature, I think made up for the loss of this. I also would have liked to add more security features.

What I learned

I learned a great deal in the development of the final year project. Almost all aspects of this project were new to me. I learned from every area of this project and sharpened my skills and knowledge. One of the biggest things for me on this project was not finalising my idea quicker, although the research continued throughout the course of the project. I think now, looking back, I should have finalised my project quicker and started implementing much sooner. I was not prepared for certain aspects of the project to take so much time and maybe in some cases for little achievement.

I also learned about the endless possibilities regarding Raspberry Pi's. As I had never used a Pi before, everything was new to me. I had used Linux before in small doses throughout college but never to the extent I did during this project. An almost downside to the Pi was having so many features and projects as I spent too long looking up what people had achieved using them. Throughout the year I was constantly finding new projects and learning bits from everywhere. This is obviously a good thing as we have a huge collection to research from but also delayed me as I would go down a rabbit hole of videos, and in some cases didn't have anything to do with what I needed. I learned Python for this project, which was completely new to me as I had never used it before.

I also got more familiar with Linux, performing tasks with it I had never done before. This gave me a greater appreciation for Linux overall and I hope to continue learning and using it in the future. I also learned a great deal about the sensors used in this project and I had tested other sensors from the kit to get familiar with. When I started I was just using a guidebook to connect sensors but by the end of it I had a greater understanding of them and how they are used in the correct pins and code.

Although it does not seem it, I would usually consider myself to be great with time management. I will never again underestimate the time frame of a project. As this project was ongoing for a college year I think I convinced myself I had more time than I actually did. I have definitely taking away a more strict approach to time keeping and meeting deadlines.

Below I have added an Update section, as some of the features that were not working initially, I got them going in some capacity. It might have been too late to add them to the overall project, but I wanted to include this as I also learned from it. It showed me stepping away from a problem and approaching it with fresh eyes can result in the desired outcome. I think I got everything working individually, I just ran out time to time to implement it to its best possible result

What I would do differently if starting over

The scope of this project was much bigger than I had predicted and one of the early problems I encountered was finalising the project. Looking back now it seems strange because I can now clearly see what I wanted it to be, but early on there was so much research done into Raspberry Pi style projects. This gave me many ideas and I had probably wanted to use a bit of everything. I should have stuck to my initial

plan more and not got side tracked with other things I could try implement. A big thing I would do differently starting over would be to create a list of every library, program and anything else I might need installed and install them all immediately. I would backup this Pi and use this on the second Pi. An issue I ran into was thinking I had a library installed on Pi1 and it was actually Pi2. This occurred for reasons like installing a library early in the project and losing track of where I installed it. It would make more sense to have the both Pi's the same and then later into the project if a new library was needed I could just add it.

I would also be much more strict with the time management, giving myself deadlines for certain aspects to be complete and this I think was my biggest mistake, as I said I wanted to achieve more than I did and I believe I would have with better time management.

I would also log my issues more and take screenshots of problems that arose. This would have made it easier to demonstrate these issues and how they were overcome.

I would familiarise myself with python and raspberry pi systems much sooner. This project was very much learning as you go, the wide range of libraries and things to do on a Pi is endless. I also stuck to much to the Sunfounder code, from reading I realised there was easier ways to implement the code and sensors but I was stuck in my ways of using this as I had the code. The Adafruit_DHT library, would have been another possibility for some of these sensors and could make it much easier to use.

Update

Remote Desktop and Locked Out

Up above I mentioned an issue with XRDP on the PI one. This happened when the project was nearly due. I was creating a demo video and wanted to remote into the Pi's. Pi2 had no issues and remotng in wasn't a problem. In my video I had to run all my commands by SSH.

From what I have read this problem can consist for a number of reasons, one being installing XRDP. I also changed the password which again, upon reding up on it is known to cause this issue. When I installed XRDP on Pi1 it installed no problem, I tried to remote in and got an error, this continued to happen. I thought it was an issue with my password, so I changed my password on the Pi. I then rebooted the Pi and it got stuck on a loop on the sign in page. I would enter my password, the screen flashes black and then the pop up for username and password was back. If I typed in the incorrect password it would tell me this, which is what is expected. There were many solutions on forums and for some people they worked, but the majority of answers was a fresh install on the SD card. I couldn't afford to do this as I needed to demo my project.

I was still able to SSH in to the Pi, I checked the logs for what was being mentioned online but everything seemed ok. I then decided to see if I created a new account what would happen. I done this through the SSH command and rebooted the Pi.

When it came back on I was able to login as the new user. I had to change privileges and allow for sudo with this new account. I was also able to gain access to the Pi users folders through this new account. Although I still had issues with some files, I was now back in and able to run my sensors. I had access to all the screenshots I had taken and code snippets that may be needed.

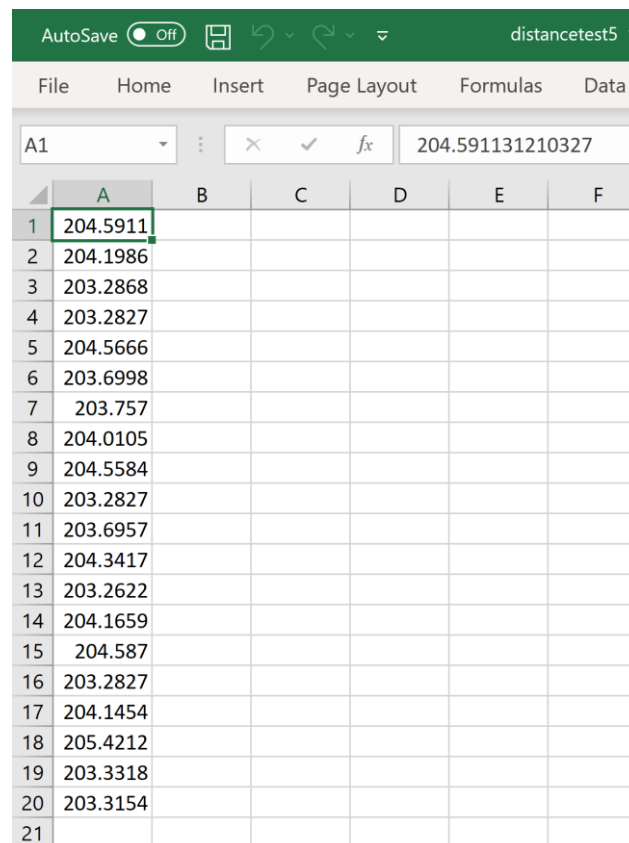
Libraries

As stated I had issues with CSV files, this seemed strange as everything I was reading online, consisted of a few lines of code, it was nothing overly complicated. I think the biggest issue was the sunfounder python code and sending the data to a csv file. I had tried different variations of what I seen online. I was running out of time and had to finish my reports and video.

After the video I went back to this and I managed to get 20 readings from the ultrasonic sensor readings going to a CSV file. Although I didn't have enough time to implement this, I have include it here as I did get it working.

```
f = open("distancetest5.csv", "w", newline="")
rc=csv.writer(f)

for x in range(20):
    rc.writerow([distcheck()])
f.close()
```

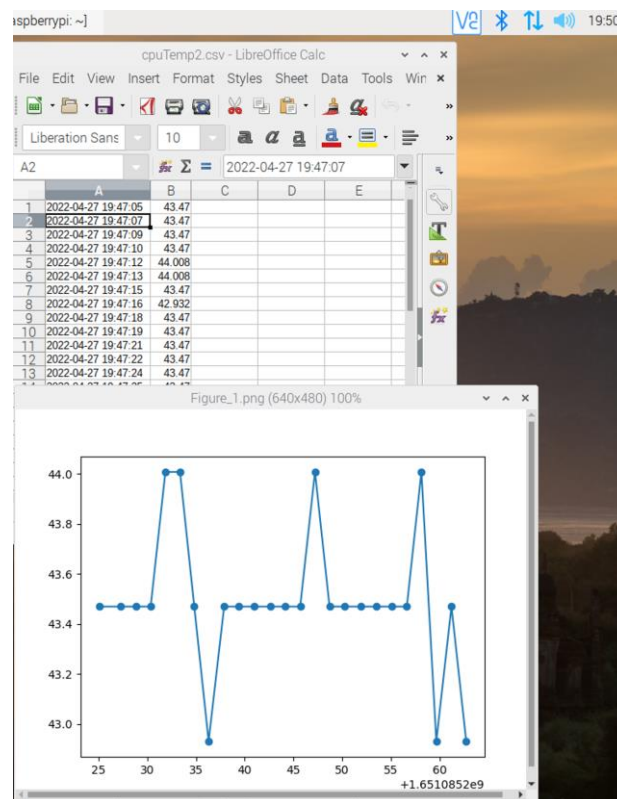


The screenshot shows a Microsoft Excel spreadsheet with the following data:

| | A | B | C | D | E | F |
|----|----------|---|---|---|---|---|
| 1 | 204.5911 | | | | | |
| 2 | 204.1986 | | | | | |
| 3 | 203.2868 | | | | | |
| 4 | 203.2827 | | | | | |
| 5 | 204.5666 | | | | | |
| 6 | 203.6998 | | | | | |
| 7 | 203.757 | | | | | |
| 8 | 204.0105 | | | | | |
| 9 | 204.5584 | | | | | |
| 10 | 203.2827 | | | | | |
| 11 | 203.6957 | | | | | |
| 12 | 204.3417 | | | | | |
| 13 | 203.2622 | | | | | |
| 14 | 204.1659 | | | | | |
| 15 | 204.587 | | | | | |
| 16 | 203.2827 | | | | | |
| 17 | 204.1454 | | | | | |
| 18 | 205.4212 | | | | | |
| 19 | 203.3318 | | | | | |
| 20 | 203.3154 | | | | | |
| 21 | | | | | | |

The issues that arose from the libraries I think were due to the way I was installing them. There is many ways to install the libraries, like sudo install, pip and pip3. These were installing the libraries for certain versions of Python and I may have installed the other libraries a different way. I think the biggest issue that had me mixed up, was if I installed something like matplotlib it would not be recognised and tell me “module not found”, yet if I checked it was installed. The libraries also seemed to be installing in different locations for this reason so although ey were installed they could not communicate with each other.

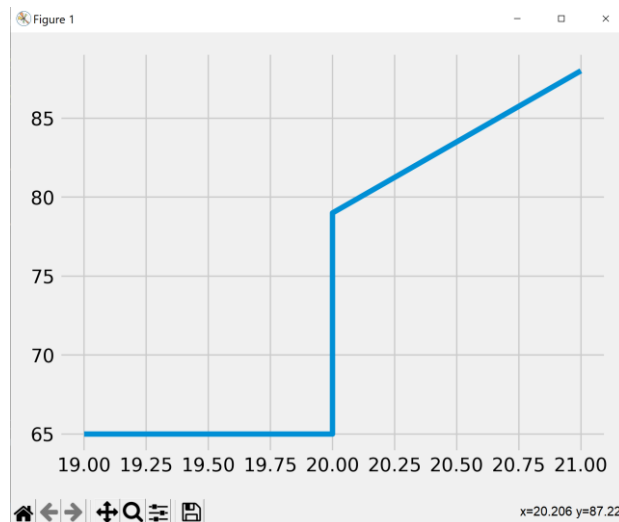
As my project had a CPU temperature monitor, I found basic ways to plot this. This code was used just to prove everything worked. The readings and time and data is sent to a CSV file and a live graph is used to plot the temperature.



Flask Libraries

I also had the same issue as mentioned above with Flask on my Windows machine. I installed a library and it still would not work. This was down to the interpreter used in Visual Studio Code. I had seen this and knew about it from early in the process of the Flask project. I had changed it and all my Flask libraries were working, when it came time to use Pandas or Matplotlib, these would not work. If I changed the interpreter, Pandas might be recognised but then Flask would stop working. I attempted this one last time and I installed all the libraries again with the interpreter that Flask was running with, this worked. I don't know if I had a clearer head the day I done this but I had tried all this previously and it did not work. Either way the libraries now all worked in my Visual Studio Code.

As I was running out of time I just wanted to test this was working. I used static values to test if the graph would appear. I inputted values representing temperature and humidity. The humidity is represented on the left while the temperature is at the bottom.



I also used the animation library which allows for a live graph with new input updating the live graph. I opened a CSV file which was being written and read from another script and inputting the data from the CSV file to the graph.

I believe with more time I would have incorporated this to my project and it would have been a stronger dashboard and represent the SCADA system better. The reason I included this section was just to show I did have everything working in the end, just not altogether.

Conclusion

Although I have stated throughout this document issues that arose along the way and the things I didn't achieve to the extent I would have liked. Overall I am happy with my project. I created two Raspberry Pi systems, which perform different features. Each Pi offers something unique from the other but I think between implementation and documents my project demos the potential of what can be achieved. It also highlights security features we need to implement on such devices for SSH, firewalls or rules for allowing traffic from certain IP addresses. I know I didn't achieve everything I hoped but between thinking back and looking over this report I was on the right track and just didn't get to implement it in time.

If I had the libraries working earlier I believe I would have had better readings. I am still proud of what I managed to achieve as nearly all aspects of the project were new to me and I learned a great deal. I am glad of the different technologies used and that I gained knowledge with hardware and programming languages I knew very little about.

I was extremely happy finding a way around some issues that arose. The things I talk about in the update section, were all done under pressure and I was proud to demonstrate my determination of getting these working in some capacity.

Although it may not mean anything, I did enjoy learning new things and once I am done college I plan to purchase a Raspberry Pi as I would be more comfortable working on one from scratch and I know the possibilities with working on them.