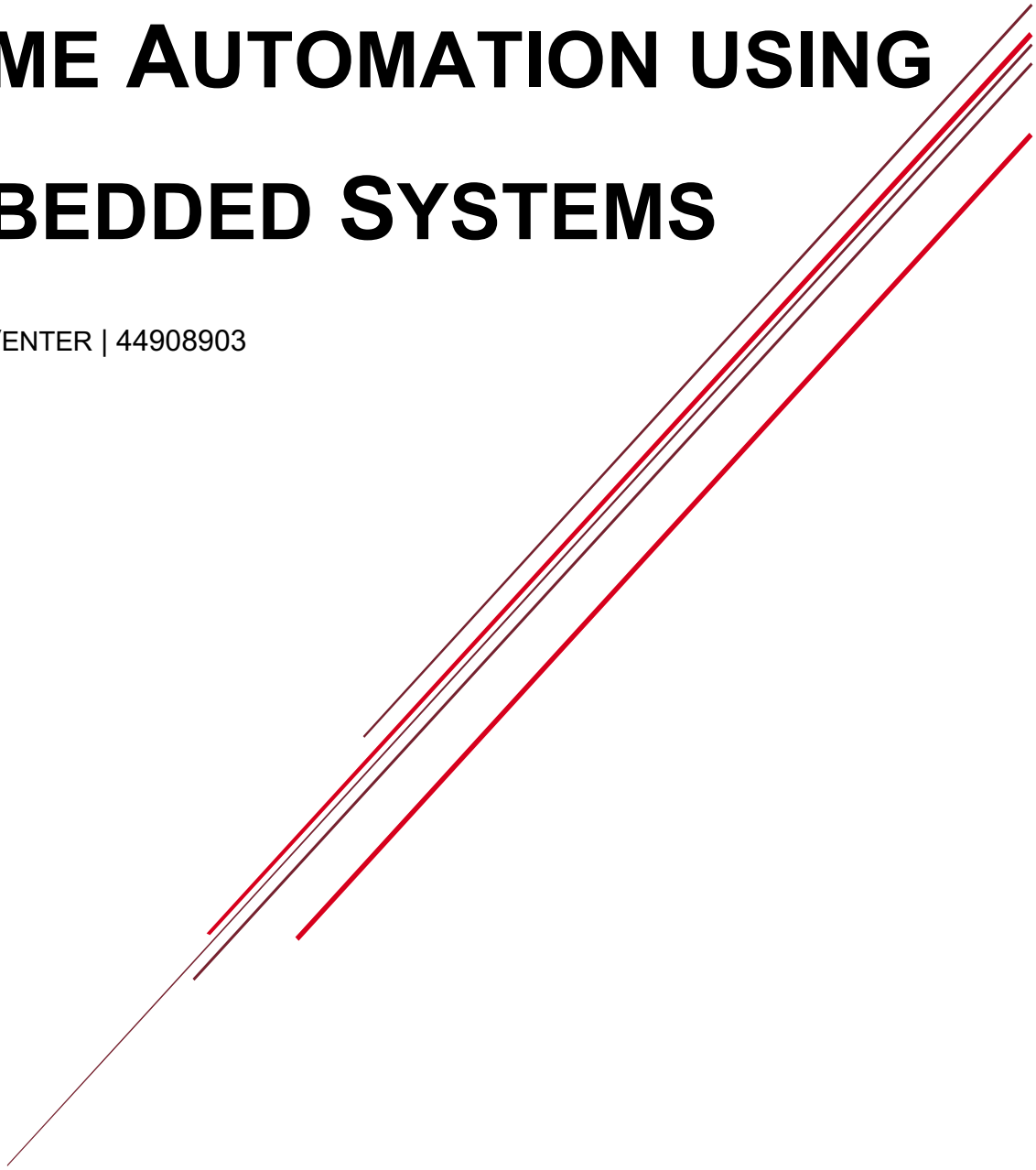
HOME AUTOMATION USING EMBEDDED SYSTEMS

HANNES VENTER | 44908903



MACQUARIE University
SYDNEY·AUSTRALIA

Table of Contents

1. Introduction

This project aimed to enable home-automation using embedded systems. It consisted of two smaller projects, both centred around home-automation. Appendix I offers explanations of additional project considerations.

1.1 Ceiling Vents

The first project was to create a system which draws hot air out of the ceiling, thus cooling the house. A fan is placed inside the ceiling space with its exhaust directing outwards through a vent. The fan is enabled once the temperature inside the ceiling reaches above a certain threshold, and runs until the ceiling is sufficiently cooled. The system will adjust the fan's power using the data from an ultraviolet (UV) sensor, to ensure that the fan is quiet during the night.

1.2 Automatic Blinds

The second project was to enable external blinds to be gradually lowered in front of windows as the sun sets. UV sensors are used to track the location of the sun. The system integrates into an existing electrical blinds system manufactured by Somfy [1]. The system sends commands using the existing remote, to gradually lower the blinds as the sun sets in the afternoon. This ensures that the room remains cool during summer and reduces the glare of the sun throughout the room.

2. Motivation

Motivation for this project arose after reflecting on the multitude of software engineers which exist. Through thought and research, I tried to determine the type of software engineer that I want to be. Embedded systems were found to be most interesting. The aspect of problem solving and creating projects with physical interactions have always been particularly enjoyable. Reflection on the work completed in ENGG200 and ENGG300 instigated the initial idea to undertake a project on embedded systems. There is a significant amount of knowledge and experience that can be personally gained in this field. These projects were thus chosen with a goal of self-improvement, above and beyond the endeavours completed in ENGG200 and ENGG300. Appendix II offers further explanations of motivations for this topic.

3. Project Plan

3.1 Microcontroller/Processor

This project does not require tremendous amounts of processing power. Arduino was selected due to its ease of use, low cost and reduced processing power. The choice of Arduino hardware was narrowed to the Arduino Uno, Arduino Leonardo and Arduino Mega 2560.

The Arduino Uno (Figure 1) was the microcontroller board used in this project. It has sufficient memory and processing power, and has enough input/output pins for the requirements of this project. It is also the most inexpensive and compact out of the three Arduino boards.

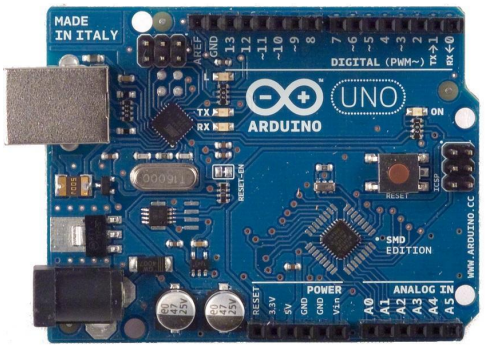


Figure 1 - Arduino Uno Microcontroller Board [2]

It "has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analogue inputs, a USB connection, a power jack, an ICSP header and a reset button" [2].

3.2 Software

C++ was used as the primary language for this project. Using this language removed some limitations of the Arduino language, and enabled the functionality of custom-made libraries. The Arduino language still had to be used in the `INO` file which contained the main functions of the program.

The goal of this project was to program as professionally and efficiently as possible. Libraries reduced the amount of code required in the main files and facilitated OO Programming which enabled multiple sensor objects to be created and controlled using a single library.

3.3 Sensors and Additional Hardware

3.3.1 Ultraviolet Sensor Module

Figure 2 shows the XC-4518 Ultraviolet Sensor which was selected to detect the location of the sun. An ultraviolet sensor was chosen over an ordinary light sensor as its readings would be notably more accurate in detecting sunlight. “This module measures the UV light and adjusts the output voltage depending on the UV intensity” [3]. It is “capable of measuring over a wide spectral range of 200nm-370nm”, has an output voltage 0-1200mV and a current 0.06-0.1mA [3].

Figure 2 - XC-4518 Ultraviolet Sensor [3]

3.3.2 Temperature Module

Figure 3 shows the XC-4520 Temperature and Humidity Sensor Module which was acquired to measure the temperature in the selected locations. It was the only temperature sensor available, however its additional humidity sensor can be valuable for future expandability. It has a temperature range of 0º C - 50º C (+/- 2º C) and a humidity range of 20 – 80% (+/- 5%). [4]

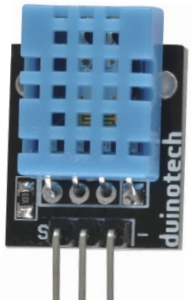


Figure 3 - XC-4520 Temp. & Humidity Sensor [4]

3.3.3 Relay Board

Figure 4 shows the XC-4419 Arduino Compatible 5V Relay Board which was selected to electronically trigger the switches on the existing remote. This module can switch up to 10A per channel and includes back-EMF protection. [5]

Figure 4 - Arduino Compatible 5V Relay Board [5]

3.3.4 Somfy Remote

The SITUO 1 RTS [6] remote communicates at a radio frequency of 433.42 MHz. It has three buttons to control the movement of the blinds:

1. UP, raises the blinds until fully retracted
2. DOWN, lowers the blinds until fully extended
3. MY has two functionalities:
    a. If pressed while the blinds are in motion, it will stop the blinds.
    b. If pressed while the blinds are stopped, it will extend or retract the blinds to a location pre-set by the user.

3.4 Project Diagrams

3.4.1 Automatic Blinds

Figure 5 is a diagram which provides an overview of the hardware placement for the Automatic Blinds project. Sensors `UV0`, `UV1` and `UV2` each have a 5 Volt supply, a Ground and a Sensor/Data cable connected from the Uno.

Sensor `UV0` is placed along the bottom of the windows and when sunlight reaches sensor `UV0`, the blinds will lower until it no longer detects UV light.

Sensors `UV1` and `UV2` are placed outside, at the top of the windows and will be used to detect when the sun has set. When neither sensor detects UV light, the blinds will be raised. Two sensors are used as a form of redundancy. If a tree branch were to cast a shadow on one sensor, the other sensor would still offer accurate readings.

There are three relays controlled by the Uno. Each relay corresponds to a different button on the remote (i.e. UP, DOWN, MY).

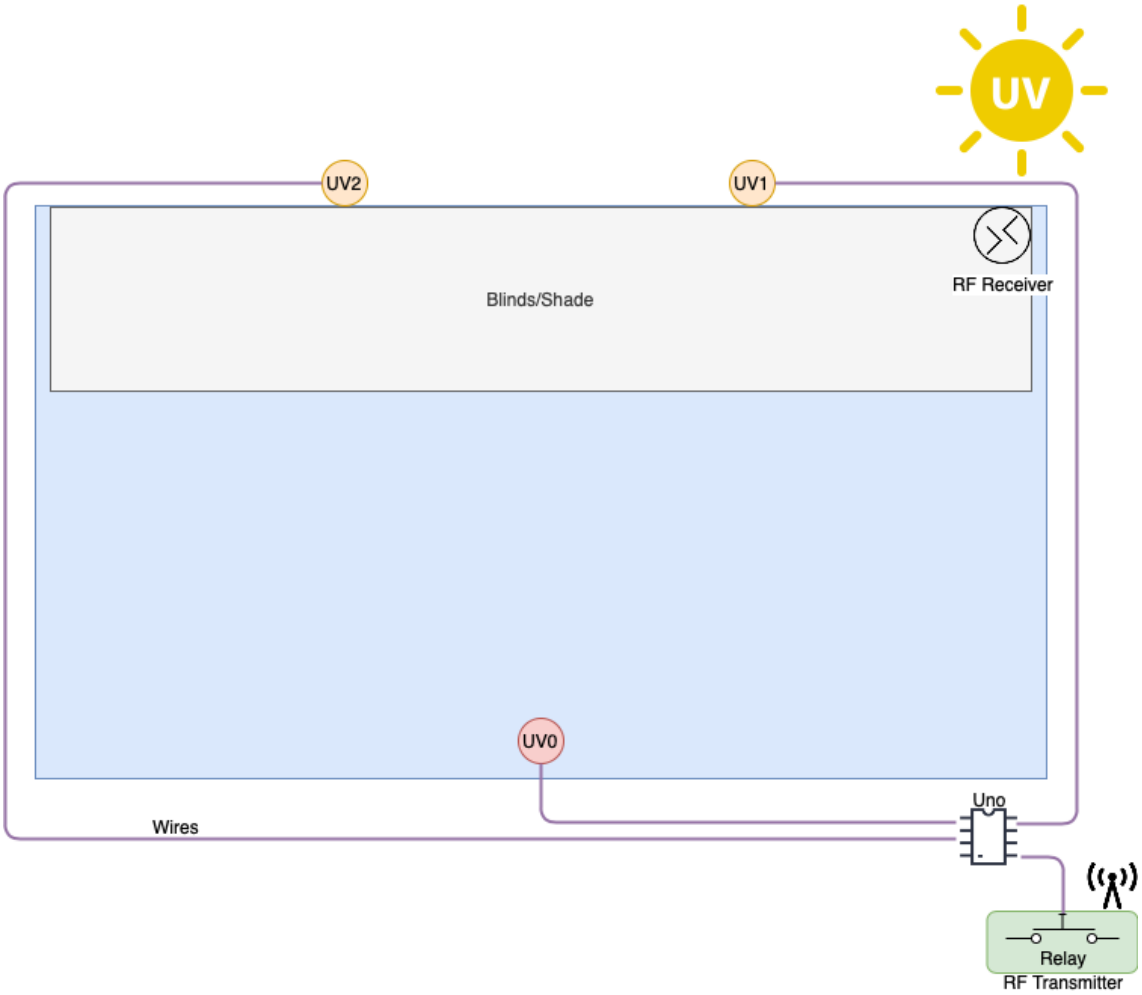


Figure 5 - Wiring/Sensor Diagram of Blinds Project

3.4.2 Ceiling Fan/Vent

Figure 6 is a diagram which provides an overview of the hardware placement of the Ceiling Fan/Vent project. Sensors `UV` and `Temp` each have a 5 Volt supply, a Ground and a Sensor/Data cable connected from the Uno. Sensor `Temp` is a Temperature and Humidity Sensor and sensor `UV` is a UV Sensor.

Sensor `Temp` is placed inside the ceiling space to monitor the temperature of the internal area. Once the temperature reaches above a certain threshold, the Uno triggers the relay which powers the fan. Sensor `Temp` will continue to monitor the temperature, and if the temperature decreases below the threshold, the Uno will trigger the relay to remove power to the fan.

Sensor `UV` is placed outside. Once the sensor stops detecting UV light, the Uno will trigger the relay to remove power to the fan.

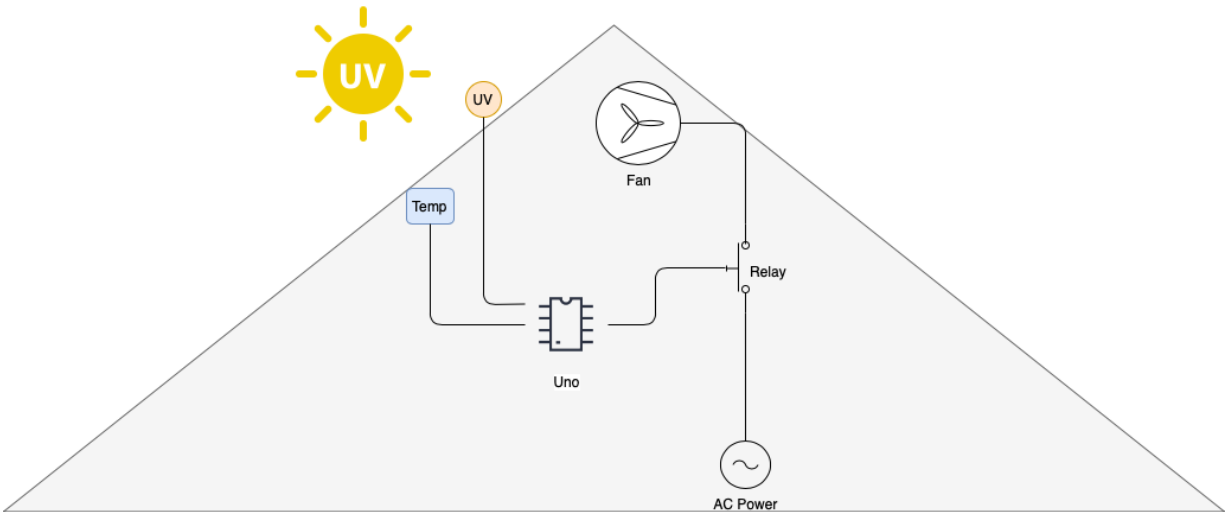


Figure 6 - Wiring/Sensor Diagram of Fan/Vent Project

3.5 Timeline

This project had a timeline of six weeks. Initial work began with research to select the most appropriate components for the project, which was followed by research into the hardware and software of the chosen components. The next stage involved prototyping and testing, before the build and integration of the components into the larger system was completed. The final stage was the completion which finalised the system. Figure 7 provides an overview of the project over the dedicated six weeks.

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
Research						
Acquiring Components						
Small Scale Prototyping						
System Protoyping						
Testing						
Building						
Completion						

Figure 7 - Gantt Chart for 6 weeks of the project

4. Progression of the Build

4.0 Week 0

Week 0 was not technically included in this 6-week project, however it was critical in ensuring the project plan was finalised. This week consisted primarily of research, planning and thought.

The research ensured that a broad variety of ideas were considered for this project. Multiple options were studied and compared such that the final choices could be selected with confidence. This research determined which processors, sensors and additional components were the most appropriate options for the project.

4.1 Week 1

Week 1 consisted of initial research into the project and any selected components. All initial research was conducted before any products were handled.

The pinout diagrams of the Uno were researched to determine what each pin does. The capabilities of the board was noted.

JayCar was the store used to purchase all the Arduino components for this project. It was selected as they had a wide variety of compatible components, and their stores were still open during this period of lockdown. The parts were purchased after sufficient research had been conducted.

4.1.1 Ultraviolet Sensors

Four XC-4518 Ultraviolet Sensors were purchased to detect the location of the sun. The supporting documentation from JayCar was only a single Data Sheet with some basic specifications about the project. No supporting libraries or code examples could be found for this sensor module. After further research, it was noted that the sensor's electronics operate in a manner similar to a potentiometer. Altering the code of a potentiometer seemed to provide raw data output from the sensor module.

4.1.2 Temperature and Humidity Sensor Modules

Two XC-4520 Temperature and Humidity Sensor Modules were purchased to measure the temperature in the selected locations. This sensor had a compressed file containing its supporting documentation. It also had a single `ino` file which seemed to contain the code required to operate this product. When uploading and running the code, both sensor modules displayed an unchanging temperature of 255.55 degrees Celsius and an unchanging humidity

of 255.55%. The code and wiring were thoroughly examined and altered, yet no further progress was made.

The decision was made to find alternate sources of code which would enable the sensors to work. It was found that these sensors were called DHT11 Temperature and Humidity Sensors in other countries. With further research, a DHT sensor library [7] was found on GitHub which handled the low-level functionality of the sensor module. A sensor object could then be created, and the corresponding temperature accessed using `dht.temperature` and the corresponding humidity accessed using `dht.humidity`. This seemed to offer accurate readings.

4.1.3 Transmitter Module

A ZW-3100 Transmitter Module was purchased as it was initially selected to communicate with the existing electrical blinds. Supporting documentation for this was also incredibly limited, with only a single PDF datasheet available. The datasheet offered significant detail about the electronics of the device, however there was no information about the software of the device. Code was not provided and initial attempts to find external libraries were not successful. It seemed that this would be a significant barrier in the project, and initial contingency plans were developed.

4.2 Week 2

Week 2 saw the commencement of initial prototyping using the individual components. Each component was connected to the Uno ensuring that the appropriate pins are matched on both.

4.2.1 Temperature and Humidity Sensor Modules

First, a single Temperature and Humidity Sensor Module was connected to pin `7`, and basic test code was uploaded to the Uno. The following code reveals how the DHT library was utilised to retrieve the readings from the sensor:

```
DHT.read11(DHT11_PIN);
Serial.print("Temperature = ");
Serial.println(DHT.temperature);
Serial.print("Humidity = ");
Serial.println(DHT.humidity);
```

This DHT library seemed to work flawlessly and its readings were constant and accurate. A secondary digital thermometer was used to compare the readings.

4.2.2 Ultraviolet Sensors

Next, a single Ultraviolet Sensor was connected to pin `A0`, and basic test code was uploaded to the Uno. Its raw value was displayed using the following code:

```
Serial.println(analogRead(A0));
```

Some formulas were found which offered to convert the output from the sensors into the appropriate UV index. The following was tried to calculate a voltage reading from the sensor:

```
int voltage = (sensorValue * (5.0 / 1023.0))*1000;
```

This calculation seems to under-value the voltage by a considerable margin. This meant that the derived UV index never changed from `0`, even in direct sunlight. As a true UV index was not required for this project, the raw values were used. Measurements were taken in multiple areas with varying levels of sunlight. These readings formed the baseline for the code.

The goal of this project was to code efficiently. The creation of libraries would facilitate this as it reduces the amount of repeated code. The Blinds project required multiple UV sensors, which could be implemented with the use of a UV library. The libraries are created using C++ classes which are split up into two files. The header file (`UV.h`) has the extension of `.h` and contains class definitions and functions. The implementation of the class (`UV.cpp`) goes into the `.cpp` file and implements the functions defined in the header file. This meant that multiple different UV sensor objects could be created in the `INO` file. Calling the `checkUV()` function on each object would return the corresponding reading from the sensor. This significantly reduced the amount of code needed in the main `INO` file.

4.2.3 Small Scale Prototype

To create the first prototype, both sensors were connected to the Uno using a breadboard (see Figure 8). The code was combined and uploaded to run in conjunction with each other. An LED was used to simulate an event happening. In this scenario, it would illuminate when the temperature reached above 20º Celsius. This simulation worked perfectly which meant that further work could commence.

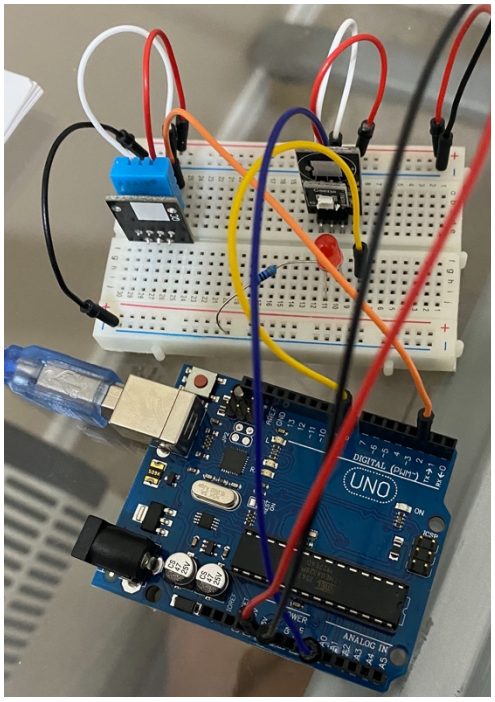


Figure 8 - Initial Week 2 CPD Prototype

4.2.4 Transmitter Module

In Week 2, some substantial problems were realised with the transmitter module. The transmitter chip that was purchased claimed to transmit at 433MHz. This was correct, however its accurate transmission occurs at the frequency of 433.92MHz. The Somfy blinds use an RF transmitter which communicates at 433.42 MHz.

This difference seemed relatively insignificant and some people [8] reported that it works fine. If this did not work, the recommendation is that the crystal resonator should be replaced with a 433.42 MHz resonator [9]. This was too complicated for goals of this project, so an alternative solution had to be found.

The most promising contingency plan was to purchase an additional Somfy remote and wire the Uno to the remote. Each button on the remote will correspond to a different pin on the Uno. The Uno will then trigger a `HIGH` on the button that it needs to use.

4.2.5 Blinds Library

In anticipation of the blinds' control mechanism working, a master control library (`Blinds.h` and `Blinds.cpp`) was created. Theis managed all the low-level functionalities, and removed any excess code from the `INO` file.

The following code shows the three primary methods that can be called once a blinds object has been created:

```
void lower();
void raise();
void stop();
```

At this stage, these methods had not been implemented.

4.2.6 Sleep Function

To improve the efficiency of the system, a sleep function [10] was implemented which places the Uno in an idle mode for a certain period of time. The Arduino would complete a single loop, and then sleep for 8 seconds using the `SLEEP_8S` call. This lowers the power consumption of the Uno and reduces the load on the CPU. This reduced load should improve the longevity of the Uno and the system as a whole.

4.3 Week 3

At the commencement of Week 3, the project was slightly ahead of schedule.

4.3.1 Relays

Due to the complications with the transmitter module, the alternative solution was to purchase an additional Somfy remote control which will be controlled using the Arduino. The plan was to simulate a button push by sending a pulse of power to the corresponding button. This did not work as the negative/ground for the Arduino was different to the negative/ground of the remote.

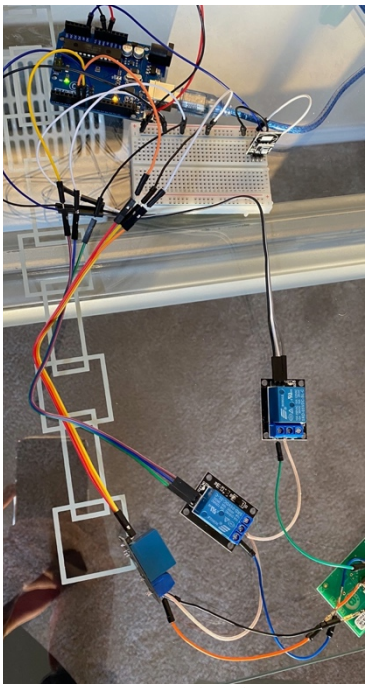


Figure 10 - Uno with Relays connected

As such, three 5V relays were purchased from JayCar and integrated into the system. Figure 9 shows the solution to replace the buttons with relays which can be triggered from the Uno. A push-button works by closing the circuit and letting power flow through. Figure 11 shows how both sides of the button were connected to the relays. The respective pins on the relay could then be connected to the Uno. Triggering a `HIGH` would close the relay simulating a button push, and a `LOW` would open the relay again.

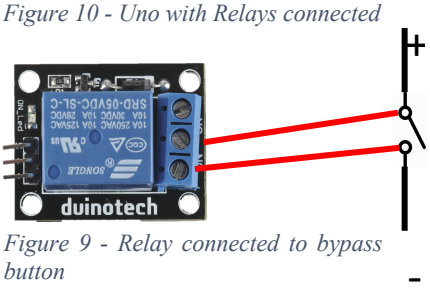


Figure 9 - Relay connected to bypass button

4.3.2 Blinds Library

Once the relays were installed, the implementation of the Blinds library could be completed. The code below shows how the `lower()` method operates:

```
    if(blindsState != 1) {
        digitalWrite(pinDown, HIGH);
        blindsState = 1;
        delay(200);
    }
    digitalWrite(pinDown, LOW);
}
```

The `blindsState` variable tracks the current state of the blinds. `1` means that the blinds are lowering, `0` means the blinds are stopped, and `-1` means the blinds are raising. To prevent a button being pushed multiple times, a relay will only be enabled if the blinds are not currently

in the required state. The relay will remain `HIGH` for 200 milliseconds, and then return to the `LOW` state. This simulates the push of a button.

4.3.3 Primary Code

The code that performs the main operation for the Blinds project is shown below.

```
1.void readUV() {
2.  if(!UV1.inSunlight() && !UV2.inSunlight()) {
3.   blinds.raise();
4.  }
5.  else if(UV0.inSunlight()) {
6.    blinds.lower();
7.    readUV();
8.  }
9.  else {
10.   blinds.stop();
11. }
12.}
```

The first condition (line 2) checks whether the two external sensors (`UV1` and `UV2`) are in sunlight. If this returns `TRUE`, it means the sun has set and the blinds can be raised (line 3).

If that is `FALSE`, it means they are detecting UV light. The next condition (line 5) checks whether the main UV sensor (`UV0`) is in sunlight. If this is `TRUE`, it will trigger the blinds to be lowered (line 6), and then recursively repeat the process (line 7) until it returns `FALSE`. If all those conditions are `FALSE` (line 9), the blinds will stop (line 10).

To prevent the system constantly making minute adjustments, a 500 millisecond delay was implemented in `Blinds.cpp`, before the stop button was triggered. This allowed the blinds to be lowered marginally past `UV0`, and slightly reduced the number of adjustments required throughout the day.

An issue which occurred in the process was when the blinds raise for the night, they remain in the raised state (`blindsState == -1`). When `UV1` and `UV2` detect sunlight the following day, yet `UV0` does not, the blinds would enter the stopped state (`blindsState == 0`). Since the blinds were not previously in the stopped state, the stop (MY) button would be pressed, which lowers the blinds to a pre-set height. To overcome this, the stop (MY) button would only be pressed if the blinds were not in either the stopped or raising states.

```
if(blindsState != 0 && blindsState != -1)
```

4.4 Week 4

In Week 4, the building of the final system commenced after successful testing and performance of the prototypes. The majority of time was spent installing the sensors in their dedicated positions and running wires from the Uno to the sensors. Figure 11 shows the creation of the Blinds system's housing and the placement of the wires.

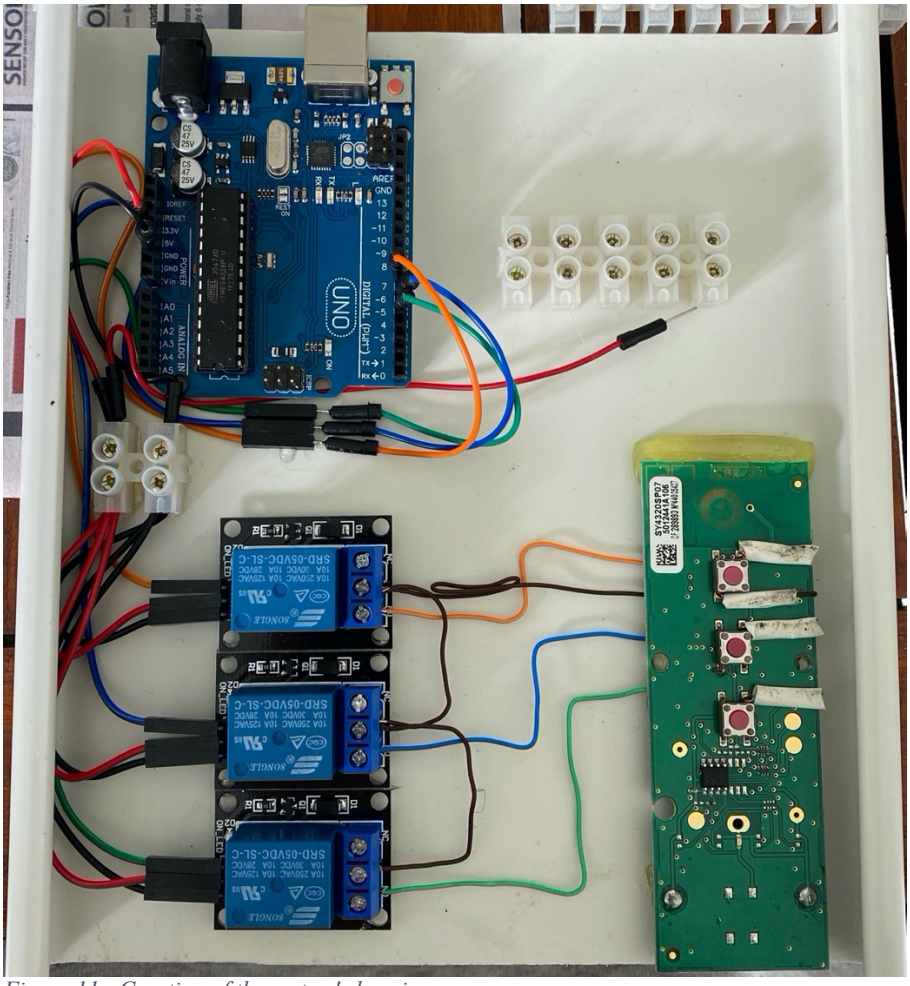


Figure 11 - Creating of the system's housing

Regular tests were conducted at every stage of the build, to ensure that each of the prior steps had been connected correctly. Prototype code which successively triggered each relay, was used to test the proper functioning of each button. Extended wires were created for each of the three UV sensors. Measurements were taken to ensure that each sensor's cable had sufficient length to reach their optimal positions.

4.5 Week 5

The wiring was finalised in Week 5. Figure 12 shows the completed housing for the Arduino, the relays, the remote and the wiring.

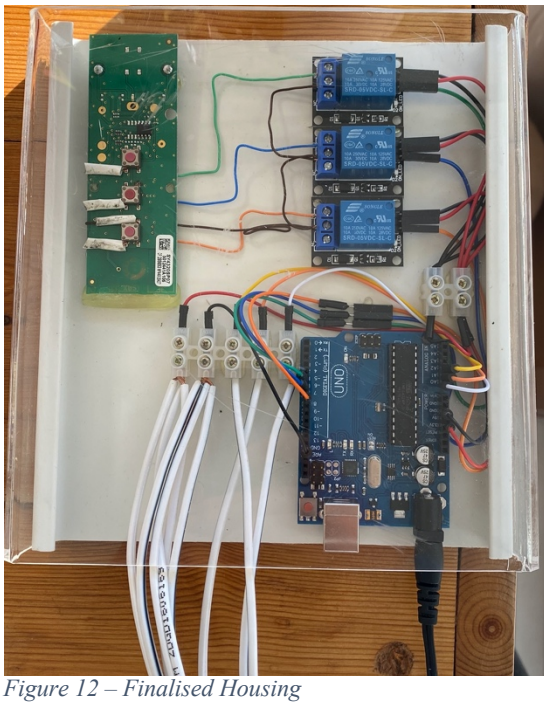


Figure 12 – Finalised Housing

Each UV sensor was temporarily placed in its selected position, and the whole system was tested over the course of a few days. Many days were overcast and had minimal sunlight which made testing difficult.

The days which had full sunlight proved valuable for any final adjustments. The readings from each sensor was slightly different, however they all displayed a reading when placed in sunlight. The raw value used to determine whether the sensors are `inSunlight()`, is a reading greater than 5. Making the tolerance higher than 0 reduces the likelihood of the system being activated in false readings.

4.6 Week 6

Week 6 saw the conclusion of the project. To protect the UV sensors which are placed outside, Perspex Acrylic Plexiglas was used create a case. When the sensors were placed and sealed in their cases, the sensors stopped detecting UV light. After numerous tests to ensure the wiring was still correct, it was noted that some types of Perspex block UV light. The final case (Figure 13) was made of a type of glass which does not block any UV light.

The wiring and sensors were placed in their final position, and the system was finalised and prepared for presentation.

Figure 13 – Glass Case for UV Sensors

5. Results

The code for this project can be found on GitHub [11].

5.1 Automatic Blinds

The automatic blinds system was completed as expected. The system functions optimally and does not require any intervention. A time-lapse video of the system can be found in the Documentation directory on GitHub [11].

5.2 Ceiling Vents

The ceiling vents project was not fully completed. Due to budget restraints and the closure of stores, an extractor fan could not be purchased. All other components are completed, and perform as expected. A relay is enabled when the temperature rises above a pre-determined threshold and disables when the temperature descends below the threshold. The only component missing is the fan which must be connected to the relay.

6. Future Expandability

Both projects have possibilities for future expansion.

6.1 Automatic Blinds

The blinds project could be expanded with the addition of the temperature module. During winter, the temperature module can be utilised to prevent the blinds from lowering until the room has reached a certain temperature. This reduces the heating requirements as the room is naturally heated by the sun.

A wind sensor would be a valuable addition. It can be used to raise the blinds during strong winds to prevent any damage to the blinds.

Additionally, an LCD screen would be helpful to monitor the system. It could display the current state of the system and the readings from all the implemented sensors.

6.2 Ceiling Vents

Once completed, the fan/vent project could be expanded with additional temperature sensors. In this project, the readings from the humidity sensors are ignored and only the temperature readings are accounted for. The accuracy of the system could be improved with a combination of additional sensors and enabling the humidity sensors.

7. Conclusion

This venture was a project for continuing professional development. Although not all aspects of the project were completed exactly as planned, the main goal of the project was achieved. The code created was efficient, and the use of appropriate commenting improves the professionalism and enables anyone else to understand and utilise this code.

The most challenging aspects of this project was the wiring and electrical work required. The creation of code which interacts with hardware is particularly challenging and has given me a greater appreciation of Embedded Systems Engineering. I have thoroughly enjoyed working on a project completely of my own creation. It has been challenging, however I have learned a considerable amount through the undertaking of the project. I had the sole responsibility for the fulfilment of this project. Through its completion I have learnt how to manage time and complications, as well as improving my ability to be self-sufficient and overcome obstacles. Through research, I have gained valuable knowledge in the field of Embedded Systems Engineering. Besides the continued professional development experienced in this project, the most rewarding aspect of this endeavour was that this project taught me how to be autonomous.

8. References

[1] Somfy, “Somfy,” Somfy, 2020. [Online]. Available: https://www.somfy.com.au. [Accessed 28 March 2020].

[2] Arduino, “ARDUINO UNO REV3,” 2020. [Online]. Available: https://store.arduino.cc/usa/arduino-uno-rev3. [Accessed 27 March 2020].

[3] JayCar, “Arduino Compatible Ultraviolet Sensor Module,” 2020. [Online]. Available: https://www.jaycar.com.au/arduino-compatible-ultraviolet-sensor-module/p/XC4518. [Accessed 28 March 2020].

[4] JayCar, “Arduino Compatible Temperature and Humidity Sensor Module,” 2020. [Online]. Available: https://www.jaycar.com.au/arduino-compatible-temperature-and-humidity-sensor-module/p/XC4520. [Accessed 28 March 2020].

[5] JayCar, “Arduino Compatible 5V Relay Board,” 2020. [Online]. Available: https://www.jaycar.com.au/arduino-compatible-5v-relay-board/p/XC4419. [Accessed 20 May 2020].

[6] Somfy, “SITUO 1 RTS SILVER,” Somfy, 2020. [Online]. Available: https://www.somfy.com.au/products/1800460/situo-1-rts-silver. [Accessed 26 May 2020].

[7] Adafruit Industries, “DHT sensor library,” 2020. [Online]. Available: https://github.com/adafruit/DHT-sensor-library. [Accessed 14 April 2020].

[8] vsky279, “Somfy module,” 27 09 2016. [Online]. Available: https://nodemcu.readthedocs.io/en/master/modules/somfy/. [Accessed 14 April 2020].

[9] Nickduino, “Somfy Remote,” 2019. [Online]. Available: https://github.com/Nickduino/Somfy_Remote. [Accessed 14 April 2020].

[10] L. P. Moh, “LowPower Library,” 2020. [Online]. Available: https://github.com/rocketscream/Low-Power/blob/master/LowPower.cpp. [Accessed 18 April 2020].

[11] H. Venter, “Embedded Systems: CPD Project for COMP4060,” 2020. [Online]. Available: https://github.com/hventer/comp4060.

[12] L. Pasqualis, “19 Types of Developers Explained,” 16 August 2017. [Online]. Available: https://www.coderhood.com/19-types-of-developers-explained/. [Accessed 19 March 2020].

Appendix I: Initial Project Ideas

After reflection and research of what kind of SE I want to be, I decided to complete a project on embedded systems.

NFC Door lock

Initials thoughts were to create an NFC enabled door lock that unlocks when a registered phone is tapped. An adverse realisation was that there are already too many systems that offer this and would perform significantly better. Another unfavourable insight was the extreme security requirements of this project, since it will control access to the house.

Automatic Lights

Another thought was to create a system that controls lights in the garden. The lights would turn on when the sun sets and it becomes dark, and then turn off when it becomes light again. This idea was disregarded as it seemed too elementary.

Automatic Blinds

Our family room has windows on three of its walls, including the western-facing wall, meaning that the sun is particularly prominent in the afternoon. Existing external blinds can be triggered to lower and raise using a remote control. We often lower it in stages as the sun sets in the afternoon. Once the sun has set, we raise it again.

This produced the idea to create a system that uses light sensors at certain increments to lower the blinds gradually as the sun sets and raises it again once the sun has set. In winter, the sun naturally warms the room reducing the need for heaters. The light sensors could thus be combined with a thermometer, which only lowers the blinds once the room reaches above a certain temperature.

Ceiling Vents

The final idea was to place a vent/fan in the upstairs ceiling which turns on once the temperature in the ceiling space rises above a certain threshold.

Appendix II: Project Motivation

I have always enjoyed problem solving. Most of software engineering has some form of problem solving in it.

This past week, I have intensely thought about the type of software engineer that I want to be. I thought back about the subjects I did and which ones I enjoyed most. COMP247 stood out, however this was more because it was directly relatable my work as an IT consultant at a school. Through my work there, I have often had to change the VLANs on switch-ports, change the IP address of various devices and interact with servers, routers and switches via the command line interface.

Something that I've admired greatly is the new metro. I am in awe of the software and hardware that makes that system work. It would be nice to be able to know that I had a role in something such as that.

To support my reflections, I decided to do some research on the different types of software engineers and what they all actually do [12].

I noticed the difference between a high-level developer and a low-level developer. Low-level seems to interact more with hardware and seemed interesting.

"This is a general term for a developer who writes code that is very close to the hardware, in low-level languages such as assembly and C. Embedded developers are often low-level developers, but not always." [12]

The main one that really stood out to me was an embedded developer.

"These developers work with hardware that isn't commonly classified as computers. For example, microcontrollers, real-time systems, electronic interfaces, set-top boxes, consumer devices, iOT devices, hardware drivers, and serial data transmission fall into this category. [12]

Embedded developers often work with languages such as C, C++, Assembly, Java or proprietary technologies, frameworks, and toolkits." [12]

Both ENGG200 and ENGG300 have shown me how enjoyable it is to work with embedded systems. I really enjoy seeing physical actions from my code. It is also heavily centred around problem solving and actually working with real things.