Renewable Energy Studio A

Interim Report - Team Project

Washing Machine

Group 06
11/06/2020
12878930 - Lucien Tran
12907960 - Manish Nath
12935084 - Jeong Bin Lee

Manish Nath, Jeong Bin Lee, Lucien Tran

Table of Contents

1. Introduction

The purpose of the project is to develop a DC motor application, such as an appliance that is driven by a motor. The application will be a derivation of power electronic concepts and techniques learnt to date. This is the full report of our project with project management, design, testing data, learning experience recorded throughout the subject and more. This report contains everything about the project including team proposal, prototype 1 and prototype 2.

1.1 Task allocation (Project)

The task allocation section is a brief table of each task that has to be performed by each member. Further information can be found in the Trello board where specific tasks are assigned to ensure clear communication and task allocation.

MEMBERS	MAIN TASKS	SUPPORT TASKS
Lucien Tran	PCB design for footprints, schematics and board	Software (Arduino)
Jeong Bin Lee	Hardware, intended DC motor application, and general design concept.	Software (Arduino) PCB design support (datasheet analysis)
Manish Nath	Gantt Chart, presentation and software with coding of the Arduino.	PCB design support (datasheet analysis)

1.1 Task allocation (Project Proposal)

MEMBERS	MAIN TASKS	SUPPORT TASKS
Lucien Tran	PCB design for footprints, schematics and board	Presentation
Jeong Bin Lee	Intended DC motor application, and general design concept.	Presentation and Report PCB design support (datasheet analysis)
Manish Nath	Gantt Chart and presentation	PCB design support (datasheet analysis) Presentation and Report

1.2 Task allocation (Prototype 1)

MEMBERS	MAIN TASKS	SUPPORT TASKS
Lucien Tran	Project Management, task allocation	Software (Arduino) Partial design
Jeong Bin Lee	Partial design and prototype, soldering of PCB	Software (Arduino)
Manish Nath	Software code (Arduino), video editing	Report and presentation

1.3 Task Allocation (Prototype 2)

MEMBERS	MAIN TASKS	SUPPORT TASKS
Lucien Tran	Task allocation, Video Editing, interrupt and delaySeconds (Software)	Software (Arduino) Debugging Prototype 2 hardware
Jeong Bin Lee	Prototype 2 hardware, Debugging	Software (Arduino)
Manish Nath	Software code (Arduino), Project Management	Debugging Hardware (Buttons)

2. Design

2.1 Intended DC Motor Application

The team agreed upon the project goal to go with the Washing Machine given in the project guideline. As outlined in the guideline, the DC motor will have 3 different cycles.

1. The washing cycle, where the motor will spin in both directions, which will allow the clothes to be washed thoroughly. After spinning one way, the motor will come to a complete stop before changing directions to make sure no electrical and mechanical equipment is damaged.
2. Next is the rinsing cycle where the motor will spin in a single direction at relatively constant speed despite the increase and decrease in load. This is when the water enters the

machine to dilute the detergent and flows out again constantly changing the load of the machine.

3. The final step is the spinning cycle where the moisture from the clothes will be drained. The motor will spin at different speeds, starting slow, then increasing speed up to a limited maximum speed and slowing down until it comes to a complete stop. This will be achieved by changing the duty cycle from 0% to 40%, then from 40% to 80%, and finally 0% duty cycle, letting the motor slow down to a complete stop.

A feature the project will also include is to have an emergency button to stop all functions without damaging components and stopping the motor within a short time. The team also discussed that this will be more like a pause button so that if the button is pressed within a timeframe, the machine will continue the cycle process.

2.2 Design Concept

There were two sections to consider in the project, one electrical and one mechanical. These include the power electronic circuit, the generator connected to the DC motor and washing machine (see figure 2.2 below).

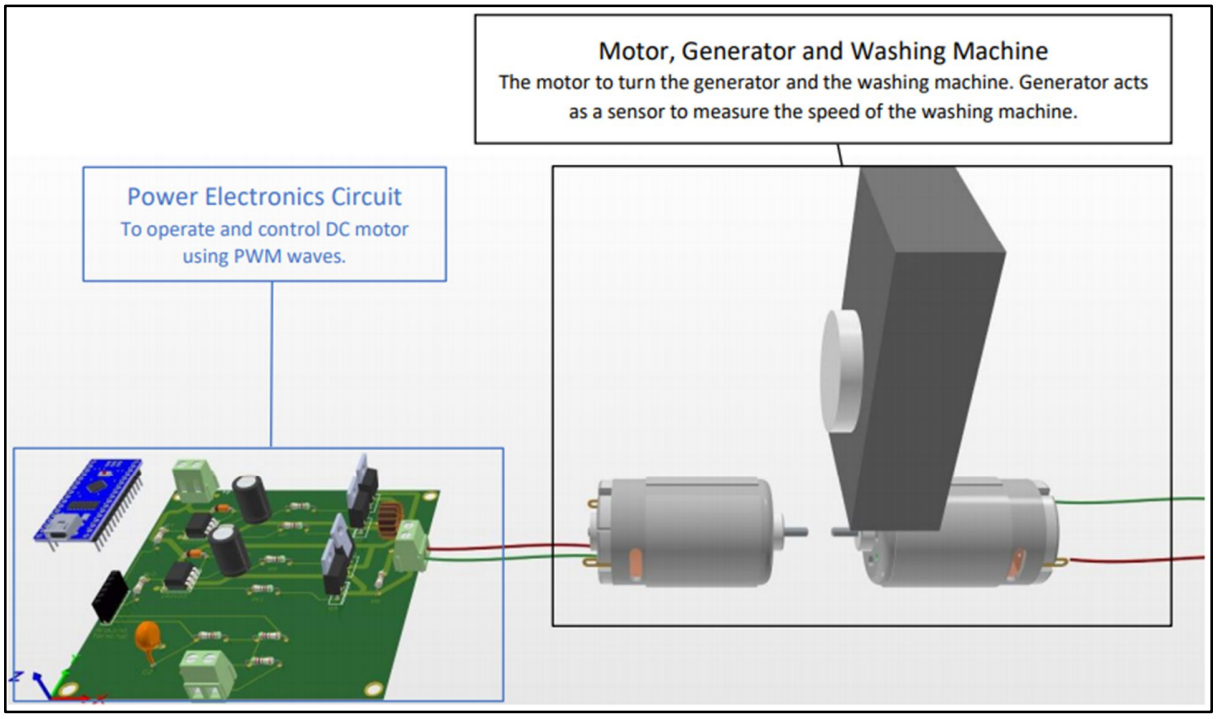
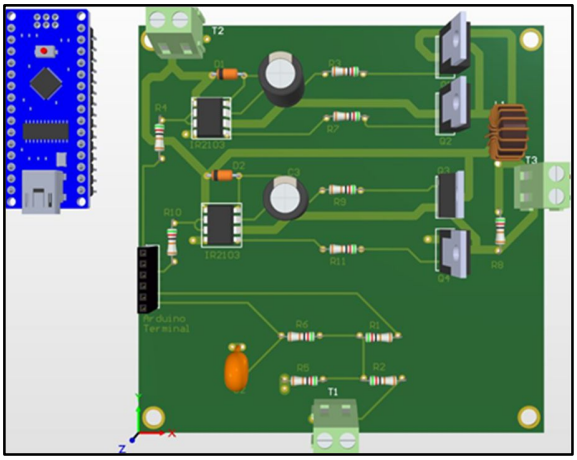
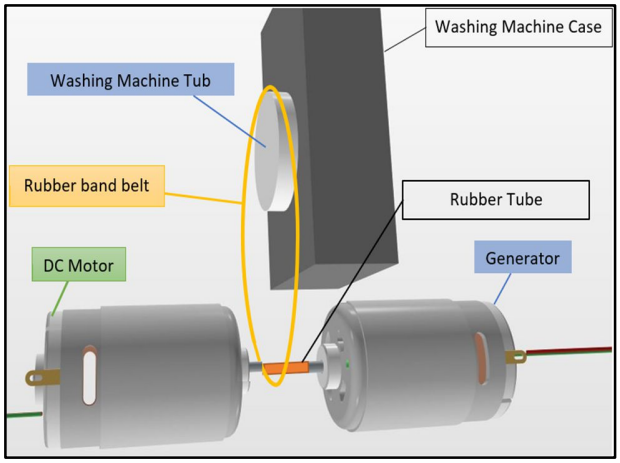


Figure 2.2 Electrical and Mechanical components of the project.



Power Electronic Circuit



Motor, Generator and Washing Machine

2.2.1 Power Electronic Circuit

The power circuit was designed with the program Altium as it was the most comfortable for the team to use. The circuit elements used are from the list given specifically for the project, no additional components were required.

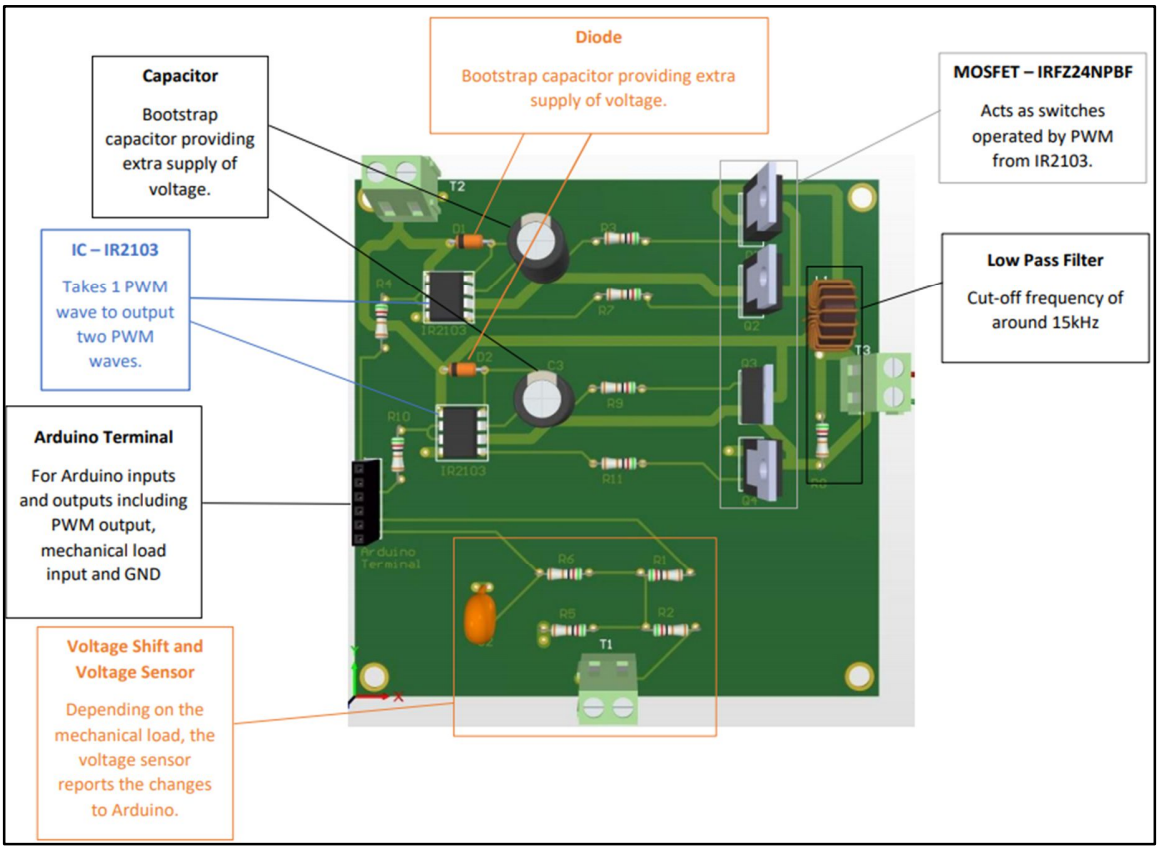


Figure 2.2.1 3D model of the designed Power Electronic Circuit.

2.2.2 DC Motor, Generator and Washing Machine

From the circuit, power is provided to run the DC motor. The DC motor is connected to the generator by a rubber tube that has enough friction to hold it in place. The washing machine tub that rotates is connected to the rubber tube that connects the motor and generator to have maximum friction for efficient energy transfer. The generator outputs the voltage generated from the motor which connects to the voltage shifter and voltage sensor. Higher the voltage, the faster the washing machine is operating at. With load in the washing machine, speed is reduced which will notify the arduino to change duty cycle accordingly.

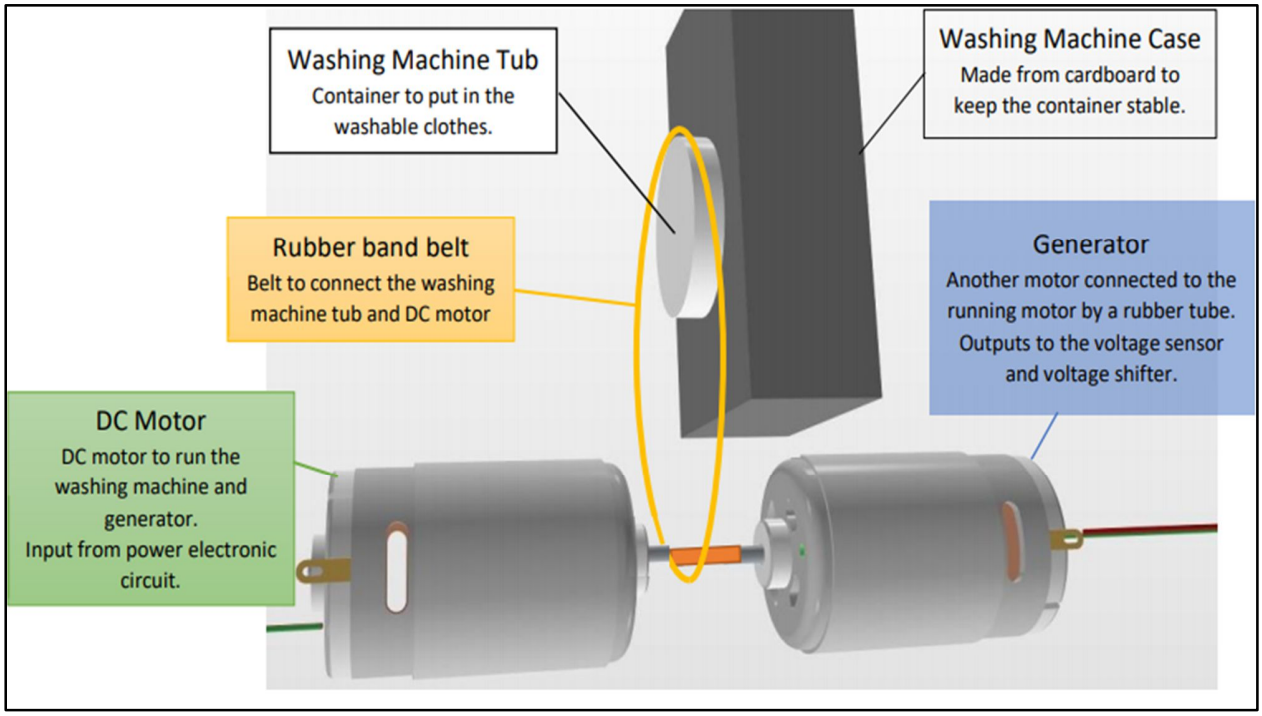


Figure 2.2.2 3D model of mechanical connection design

2.3 PCB

2.3.1 Circuit Design

Designing a PCB requires several steps. The first step is choosing the appropriate program: for this project, it was decided to use Altium Designer.
The second step of the PCB design is to check all the components and their datasheet in order to design their footprint. All components are through-hole, the opposite of surface mount.Through-hole components are meant to through a hole designed on the PCB and later soldering and cutting the legs of the components. To design through-hole footprints on Altium, simply place a pad for a hole. Thus, a resistor will consist of two pads. The diameter of the hole needs to be adjusted depending on the width of the leg and the distance between the two pads adjusted for the length of the component. The figure below demonstrates how the footprint is designed for the IR2103. It is good practice to add a sign on the top overlay to show where pin 1 is and therefore avoiding confusion when soldering the different components.

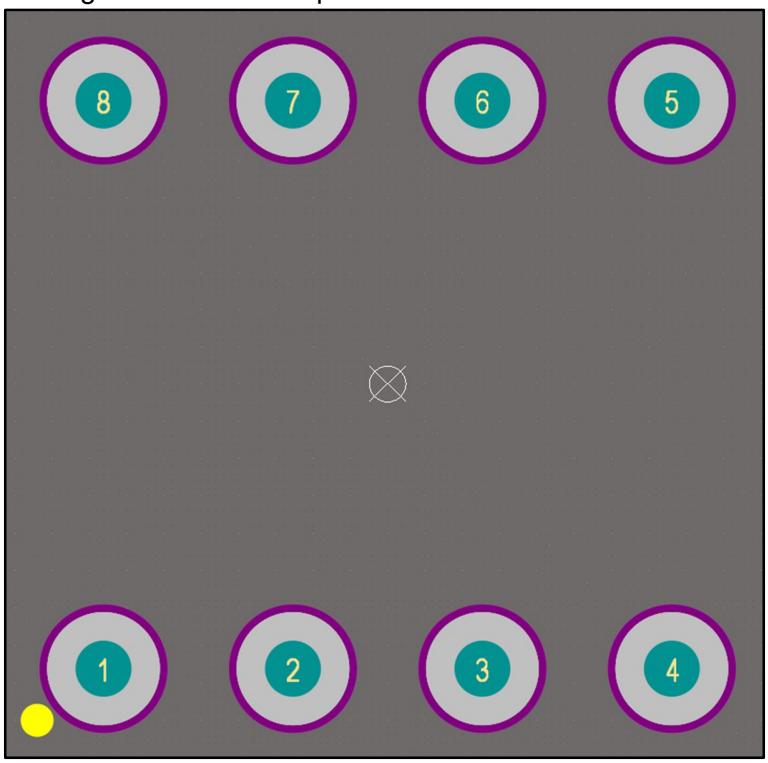


Figure 2.3.1.1. IR2103 footprint in Altium

The next step is creating the schematic symbol for each component and linking it to the corresponding component. The step consists of creating a schematic library for the schematic. Following the creation of IR2103, find below its schematic symbol. The pin needs to be added at that point to differentiate them.

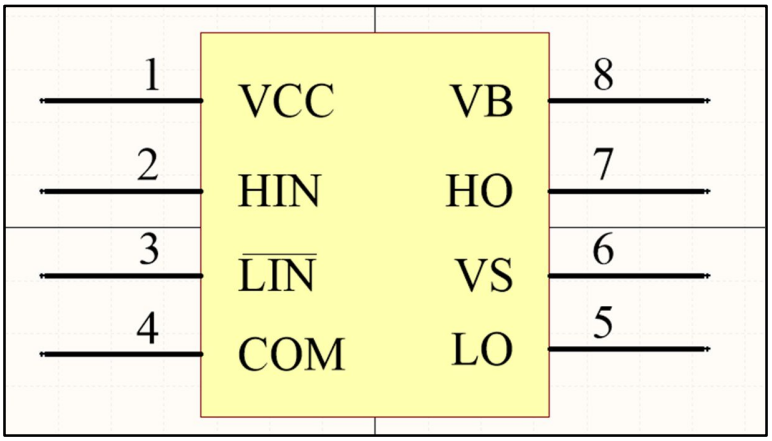


Figure 2.3.1.2.IR2103 schematic symbol

Once all schematic symbols are created, the schematic document needs to be created as well as its routing.

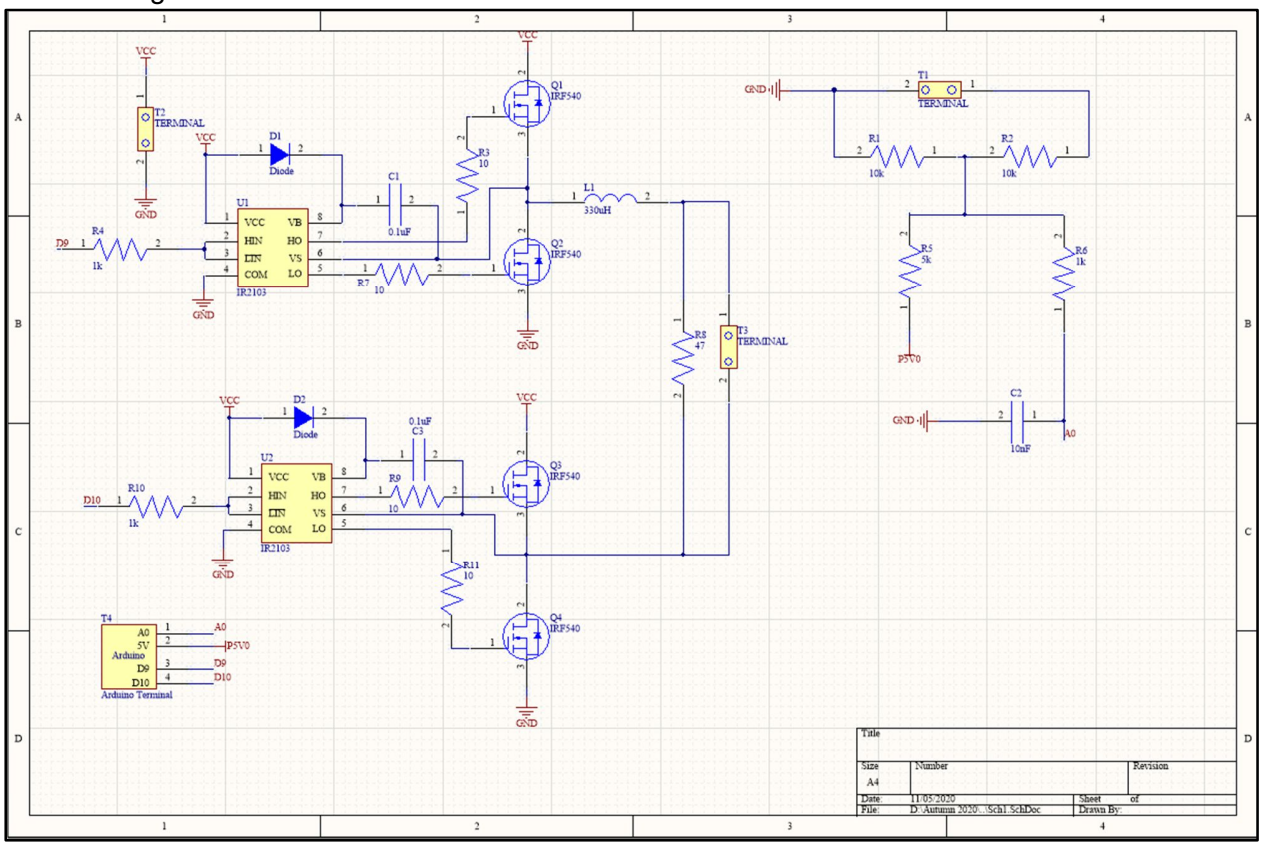


Figure 2.3.1.3. Schematic document of the PCB

Two different circuits were created: the input signal to operate the washing machine and the feedback that is connected to the arduino and further analysed by the microcontroller to output a more suitable signal.

Finally, the actual PCB needs to be designed. The PCB board cannot be bigger than 10x10cm according to the requirements. Another design requirement is routing with thicker wire where the voltage is high. Thus, all ground and Vcc are significantly thicker than other connections.

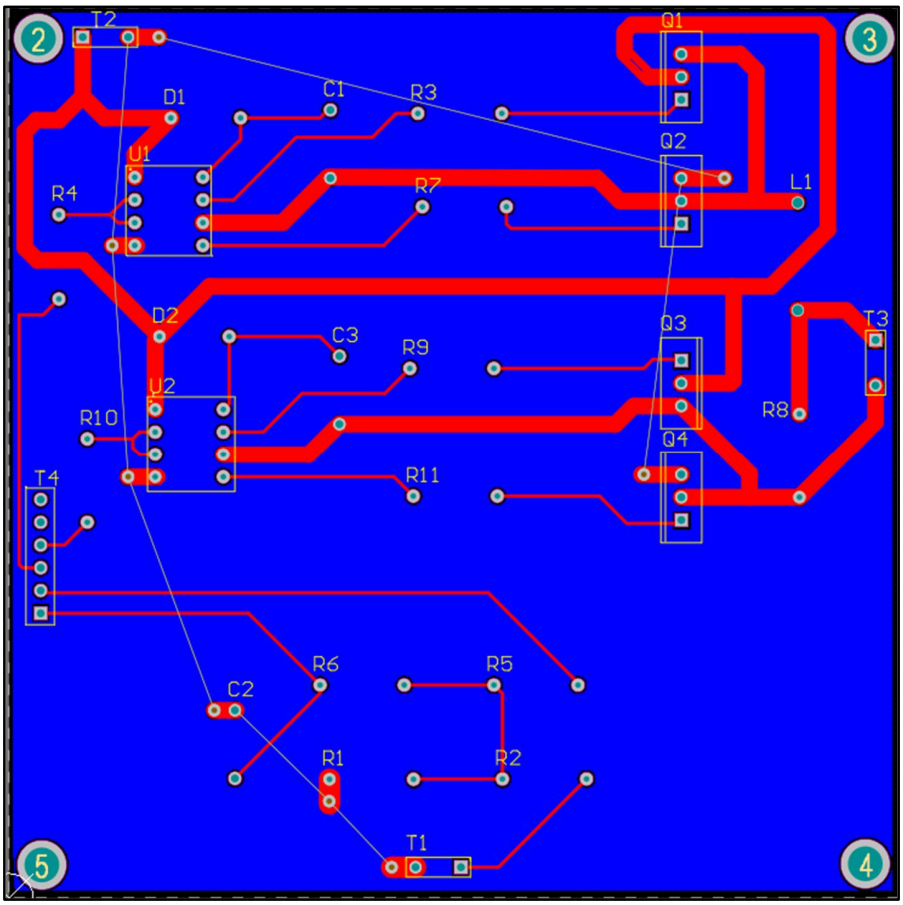


Figure 2.3.1.4 PCB design Altium

The red colour in Figure 2.3.1.4 shows the first layer of the PCB where all the connections are done. The second layer is designed for a ground plane. To do so, an option in Altium called ‘polygon pour’ was used. The second layer was filled and all ground are connected. It allows us to ground one pin only rather than all of them.

2.3.2 Soldering

Soldering did not take much time or effort to finish. The only trouble during soldering was getting the direction of the MOSFET correctly and noticing small flaws in the PCB design. For example, the 100uF capacitor between Vcc and COM was initially thought unnecessary until the actual practical experiment of a full-bridge converter was conducted. To solve this problem the 100uF capacitor was soldered in as shown in figure 2.3.2.1 below.

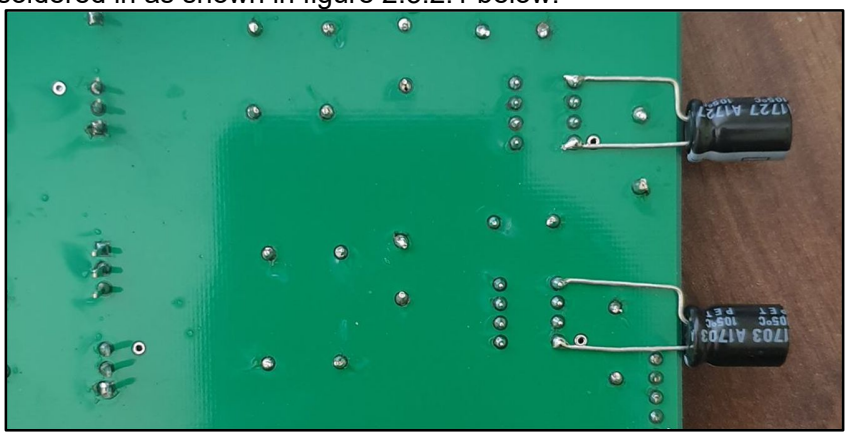


Figure 2.3.2.1 100uF capacitor soldering design.

During the initial testing phase of the PCB, the motor did not run at all because the ground was not connected internally on the 2nd layer of the circuit board as designed on Altium. It was only after connecting the ground on the separate breadboard the motor started to run. This is believed to be a fault in manufacturing. An idea to solve this problem is to solder the vias of the PCB to the port which will then connect to the Arduino NANO ground to avoid additional wiring on the breadboard. The final soldered circuit board can be seen in figure 2.3.2.2.

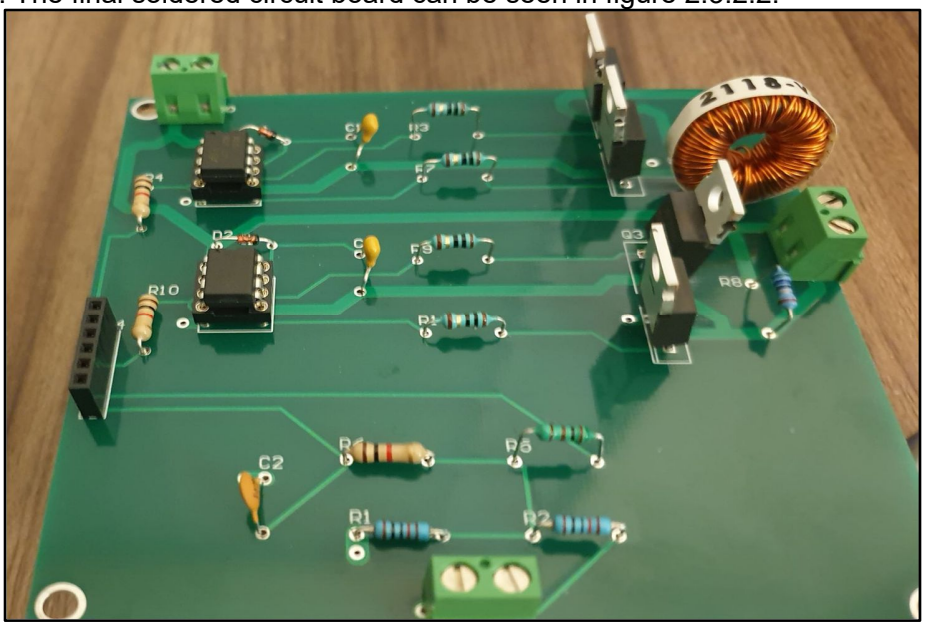


Figure 2.3.2.1 Soldered PCB.

2.4 Prototype

2.4.1 Prototype 1

From the 2.2 Design Concept, the washing machine is made using an empty cardboard box, a can for the washing machine tub and a pole for the rotor connected to the motor bought from an arts and crafts shop. The design consists of minimum friction between the rotor and the box and maximum friction between the motor and the rotor by using a rubber band. However, during the initial test the load on the motor was too high, the distance between the motor and rotor had to be reduced from 15cm to around 9cm. After the adjustment the motor ran without any problem, a little improvement on the mechanical design such as changing the cardboard box to a wooden box for more stability would significantly improve the overall performance and aesthetics. See figure 2.4.2.1 for the fully operational prototype 1.

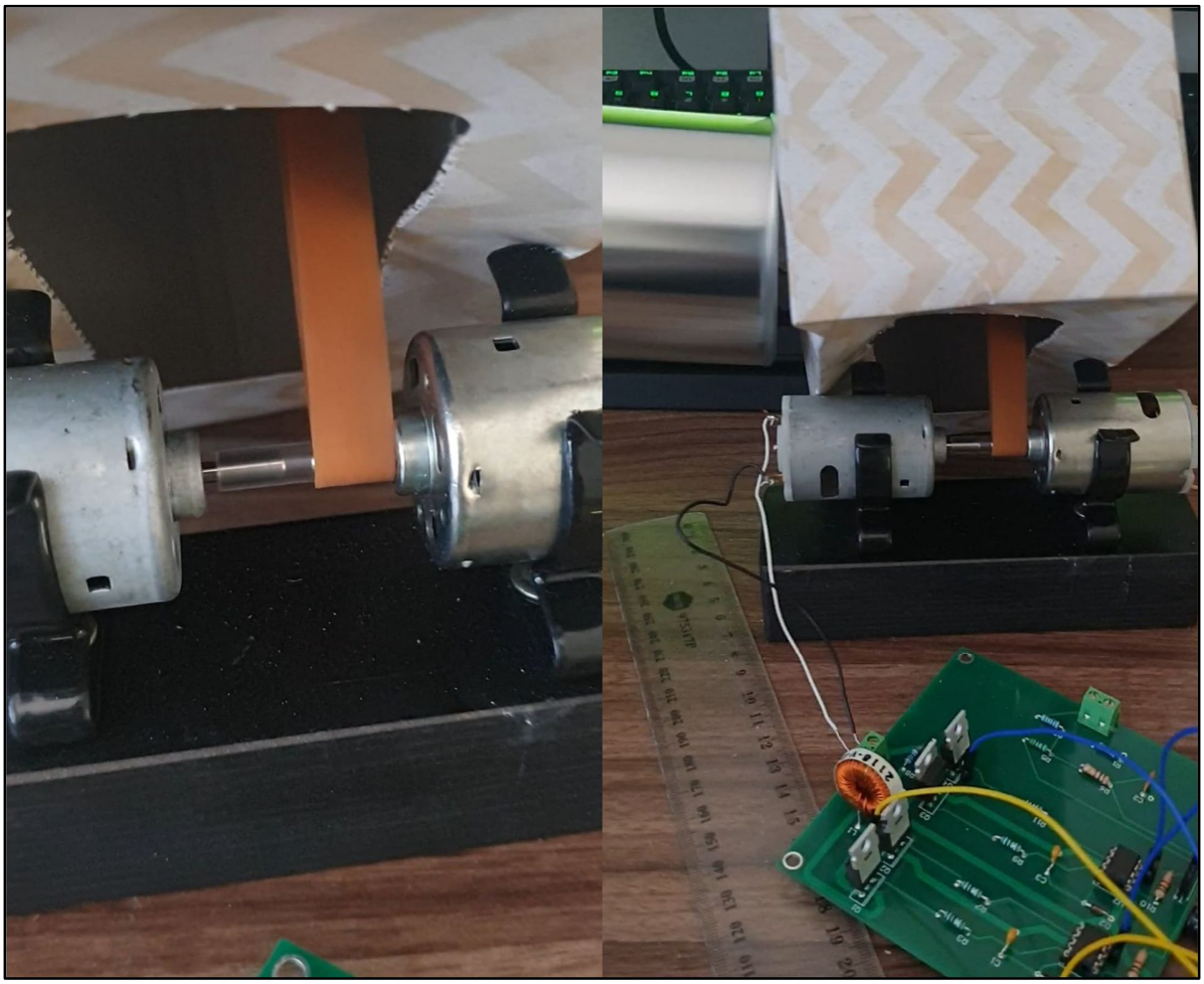
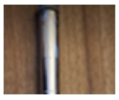


Figure 2.4.2.1 electrical and mechanical design prototype 1.

2.4.1.1 Prototype 1 Bill of Materials

Each component in the mechanical design cost $2.80 from an arts and crafts store. These include a metal tin as a washing machine tub, a metal pole connected to the tub acting as a rotor and rubber band as a belt between the motor and the washing machine. Additionally, some circuit components such as diodes and MOSFET were damaged during the testing phase. To prepare for the worst case scenario, additional components were bought at the nearest Jaycar.

Item Number	Item Name	Image	QTY	Cost	Total Cost
1	Pen holder metal can		1	$2.80	$2.80
2	Chrome plated rod		1	$2.80	$2.80
3	String Cyam Acry (super glue)		1	$2.80	$2.80
4	Wide rubber band		1	$2.80	$2.80
5	40 PIN female header strip		1	$2.50	$2.50
6	Resistor 10R		1	$0.55	$0.55
7	Resistor 1K0		1	$0.55	$0.55
8	Capacitor 100uF		2	$0.95	$1.90

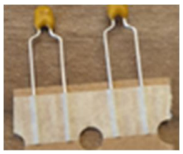
9	Capacitor 10nF		2	$0.35	$0.70
10	Capacitor 100nF		4	$0.35	$1.40
11	Terminal PCB		2	$1.35	$2.70
12	MOSFET IRF1405		4	$5.95	$23.80
13	8 pin IC holder		4	$1.25	$5.00
				Total Cost:	**$50.30**

Table 2.4.1.1 List of materials purchased for project prototype 1.

2.4.2 Prototype 2

2.4.2.1 Mechanical Design Improvements

There were several improvements to the mechanical design from prototype 1. The cardboard box from prototype 1 (see 2.4.1 for prototype 1) was too weak, it could barely hold the washing machine tub and the rotor in place, so in prototype 2, rather than using a wooden box with added friction, a cardboard box was used instead. This worked out well since it was light and easy enough to cut and change the properties of the design, but strong enough to hold the washing machine in place while the motor is spinning at maximum speed. Additionally, the rotor and the tub was too heavy, it was putting too much load on the motor, so the team decided it was best to change it to something lighter. A wooden rod and an empty metal can was chosen to reduce the amount of load on the motor (see figure 2.4.2.1).

Figure 2.4.2.1 Working mechanical design of prototype 2.

2.4.2.2 Electrical Design Improvements

In terms of the mechanical working design, there has been no change from prototype 1. However, for circuit design, there have been multiple component failures in the PCB, after damaging the second circuit board, the team decided it would be safer if the circuit was done on a breadboard as shown in figure 2.4.2.2. Since two circuits have been damaged, all the components on the circuit board were soldered in, hence could not be used in prototype 2. MOSFET IRF1403 with similar characteristics as IRF540Z were purchased as backup during prototype 1, and was used in prototype 2 which in fact worked better than the given MOSFET.

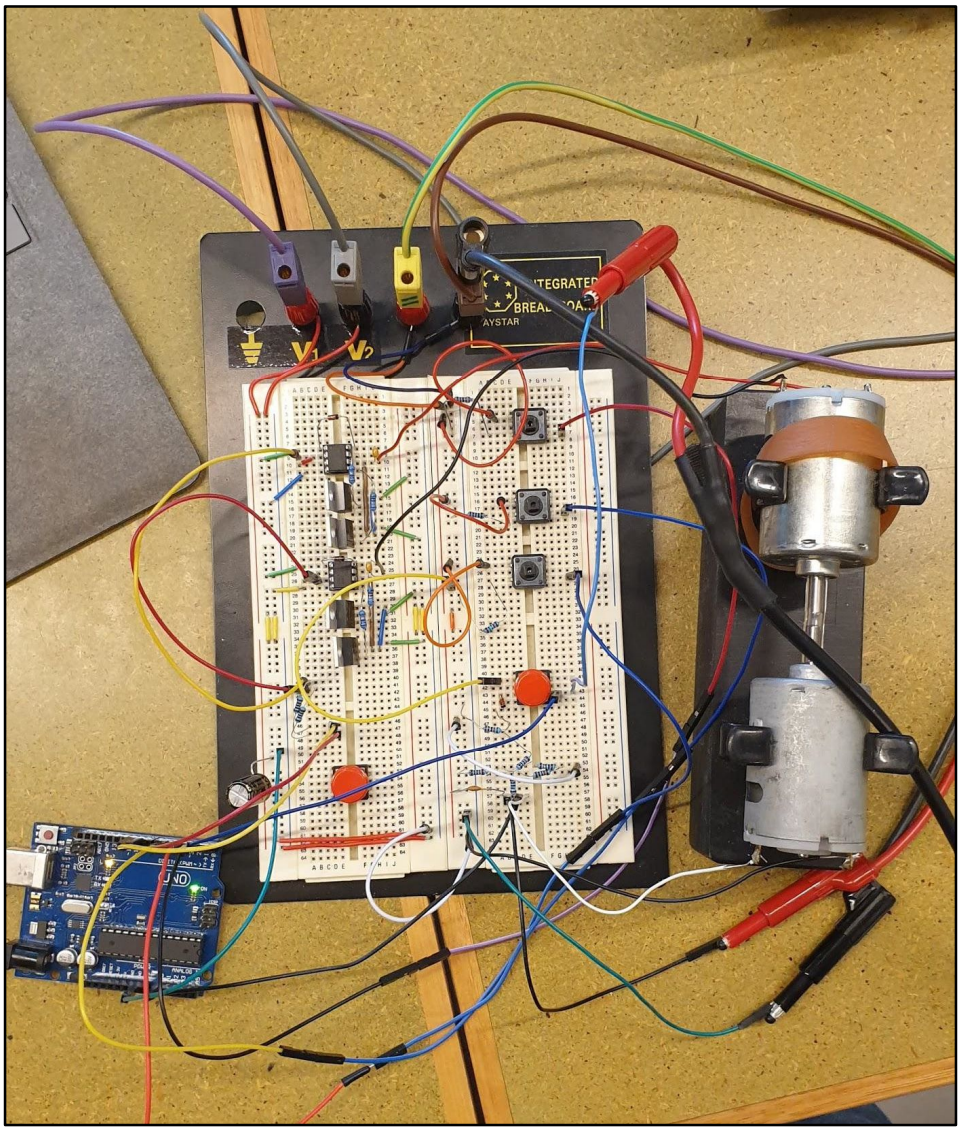


Figure 2.4.2.2 Prototype 2 circuit on breadboard.

3. Arduino Code

3.1 Flowchart of application operation program

PREVIOUS FLOWCHART

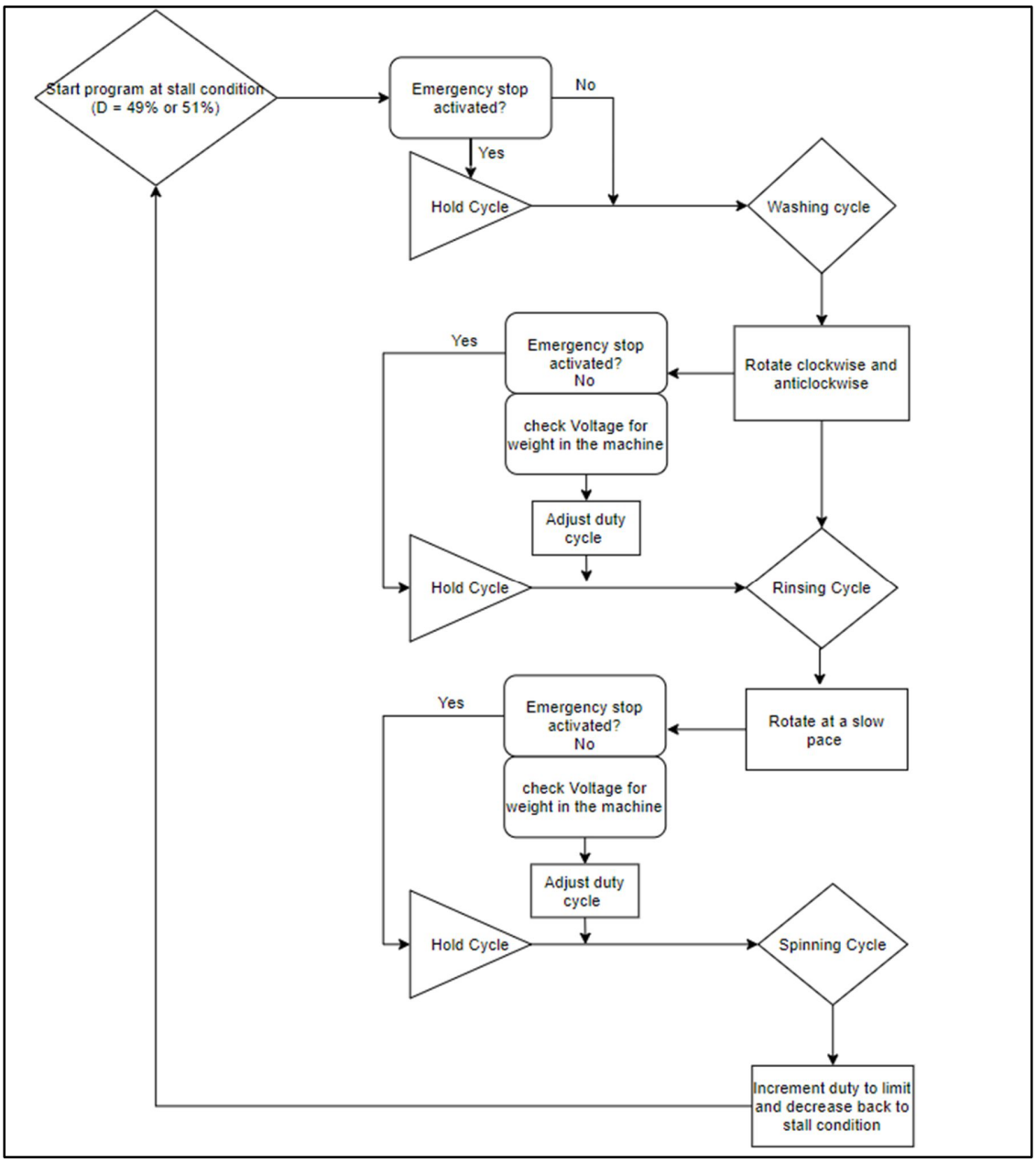


Figure 3.1.1 OLD FLOWCHART

NEW FLOWCHART OF OPERATION

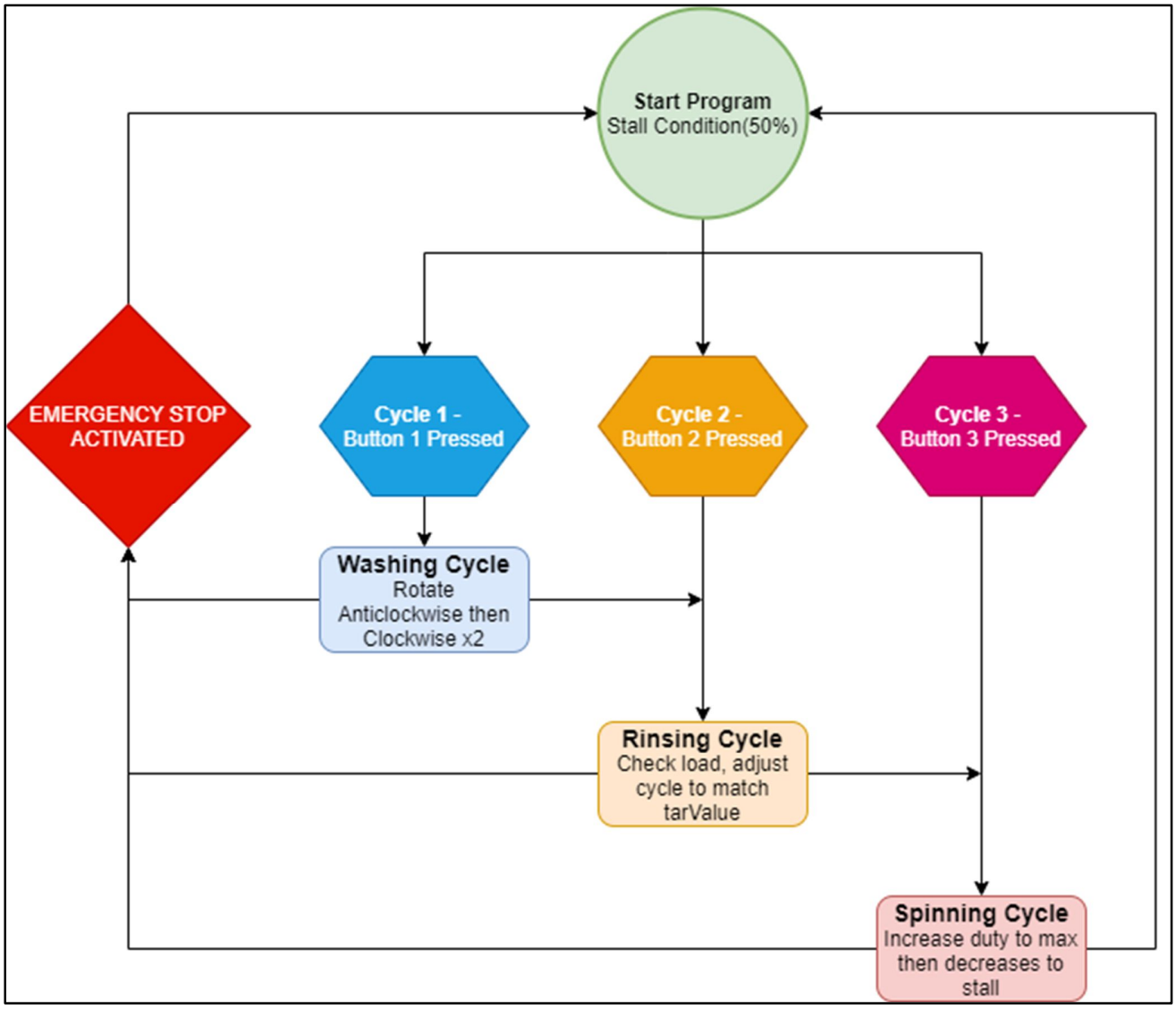


Figure 3.1.2 NEW Operation Flowchart

The washing machine is designed to function as an home appliance, based on this each cycle is designed accordingly. Cycle 1 is designed for full functionality where the washing machine will go through all the cycles, washing cycle, rinsing cycle and spinning cycle. This cycle was set up so that it doesn't require the user's input for any given cycle and runs like a normal washing machine. Cycle 2 consists only of the rinsing cycle and spinning cycle, this option is used either if the user engaged the emergency stop after or during the washing cycle, or only requires the use of the rinsing and spinning cycles. Finally Cycle 3 consists only of the spinning cycle, which is used to dry the clothes so it's not as damp. The machine is also implemented with an emergency stop button that can be triggered at any point and will STALL the motor and return to the main loop ready for the user to input what cycle they want.

3.2 Code Explanation

3.2.1 Main function and buttons

```
void loop()
{
  sensorValue = analogRead(A0); //reading analog value
  interruptValue = digitalRead(2); //reading digital value from pin2
  delaySeconds(100);
  voltage = sensorValue * (5 / 1023); //actual voltage of the PIN

  washingState = digitalRead(washingButton); //State value of washingButton whether high or low
  rinsingState = digitalRead(rinsingButton); //"^"
  spinningState = digitalRead(spinningButton); //"^"

  Serial.print("A0 Value:"); //prints sensorvalue on the serial monitor
  Serial.println(sensorValue);
  Serial.print("duty: "); //prints duty on the serial monitor
  Serial.println(duty);
  Serial.print("Interrupt Value:"); // value of whether the interrupt is pressed or not
  Serial.println(interruptValue);


  //if statements for sequential cycles
  if (washingComplete == 1) //if washing cycle has completed then proceed
  {
    rinsingState = HIGH;
    //Rinsig Button state to high to trigger rinsing cycle to begin after washing cycle is complete
    washingComplete = 0; //resets washingComplete value back to 0
  }

  if (rinsingComplete == 2) //if rinsing cycle has completed then proceed
  {
    spinningState = HIGH;
    //SPinning Button state to high to trigger spinning cycle to begin after rinsing cycle is complete
    rinsingComplete = 0;
  }
```

Figure 3.2.1.1 Main Function Pt1

Above is part of the main function in this section the main elements to highlight are the *washingState, rinsingState, spinningState*, and the *if* statements. The state variables are to measure the states of the buttons to determine whether they have been pressed or not. The buttons are connected to digital pins rather than analog, which requires us to use *digitalRead* to determine whether they are *HIGH* or *LOW*. If conditions are based on the completion of either the washing cycle or the rinsing cycle, these will allow for sequential cycles without the need for the users inputs.

```
  if (washingState == HIGH) //HIGH means on
  {
    Serial.println("WASHING CYCLE");
    washingComplete = 1; //when washing triggered it sets to value to true for the if statement
    WASHING_CYCLE(); //calls washing cycle
    Serial.println("END WASHING CYCLE");


  }
  else if (rinsingState == HIGH)
  {
    Serial.println("RINSING CYCLE");
    rinsingComplete = 2; //sets value to true for if statment
    RINISING_CYCLE(); //calls rinsing cycle
    discharge(); //stalls the motor after rinsing
    Serial.println("END RINSING CYCLE");


  }
  else if (spinningState == HIGH)
  {
    Serial.println("SPINNIG CYCLE");
    SPINNING_CYCLE(); //calls spinning cycle
    Serial.println("END SPINNING CYCLE");
  }
  else {
    Serial.println(" NO CYCLE");
    if (!delaySeconds(1000))
    {
      return; //returns back to the start of void loop
    }
  }

}
```

Figure 3.2.1.2 Main Function Pt2

There are 4 cases from the main menu, 3 of which are selected from the user. The cycles and what each input results in can be seen in *Figure 3.1.2,* the flowchart. At the end of the washing cycle it changes the value of *washingComplete* to true so that when the *if* statement is exited it meets the condition of sequential *if* statement changing the value of *b2State*, running rinsing cycle without any input. The same is true for the end of the rinsing cycle.

3.2.2 Delays

```
boolean delaySeconds(int interval)
{
  timer = millis();
  while((interrupted == false) && (millis() < (timer + interval)))
  {
  }
  if(interrupted)
  {
    Serial.println("interrupted worked!");
    interrupted = !interrupted;
    duty = STALL;
    update_pwm_bipolar(duty);
    b1State = b2State = b3State = LOW;
    Serial.println("BACK TO STALL");
    return false;
  }
  else
  {
    return true;
  }
}
```

Figure 3.2.2 - delaySeconds function

The first prototype and first version of the code was implemented using the function delay(). However, delay() is deemed to be a bad practice for the microcontroller and interrupts will not work. In fact, delay is using interrupt and it is not possible to have an interrupt inside another Interrupt Service Routine (ISR). Thus, delay cannot be used in our emergency button ISR either.

In Prototype 2, the team has decided to implement a user function that takes an interval in milliseconds as a parameter. Using the function millis(), we managed to create a delay function where interrupts are possible.

3.2.3 Interrupt

```
void STOP() //when digitalInterrupt is triggered STOP function is called
{
  if(digitalRead(2)) //if the value of Pin2 is true = 1, then run interrupt
  // only works if button pressed not for random triggers
  {
    interrupted = true; //changes value of interrupted for delaySeconds function

    //Resets all the Cycle Completion indicators
    washingComplete = 0;
    rinsingComplete = 0;
    spinningComplete = 0;
  }
}
```

Figure 3.2.3 - ISR routine for the emergency button

A fundamental rule about interrupts is to have as less code as possible. Therefore, our emergency button ISR is only raising a flag. That flag will break the while loop of the delay and bring the Prototype to a stall condition (see Figure 3.2.2). After it is done, the interrupt flag is cleared and the washing machine is off.

It was not deemed necessary to shut down the motor in the ISR as each line of code would take less than 50 us, which is too fast for the user to perceive. Thus, bringing the motor to a stall is only done in the delaySeconds where the interrupt breaks the delay loop and will bring the washing machine to its starting state.

3.2.4 Washing Cycle

```
void WASHING_CYCLE()
{
  //Washing cycle operation
  for (int i = 0; i <= 1; i++) //will run the washing cycle 2 times
  {
    duty = DUTY1; //Anti - Clockwise for 3 secs at duty1
    update_pwm_bipolar(duty);
    Serial.print("DUTY: ");
    Serial.println(duty);
    if (!delaySeconds(3000)) //run for 3 secs
    {
      return;
    }

    duty = STALL; //Stall for 1 sec
    update_pwm_bipolar(duty);
    Serial.print("DUTY: ");
    Serial.println(duty);
    if (!delaySeconds(1000))
    {
      return;
    }

    duty = DUTY2; //Clockwise for 3 secs at duty2
    update_pwm_bipolar(duty);
    Serial.print("DUTY: ");
    Serial.println(duty);
    if (!delaySeconds(3000))
    {
      return;
    }

    duty = STALL; //Stall for 2 second
    update_pwm_bipolar(duty);
    Serial.print("DUTY: ");
    Serial.println(duty);
    if (!delaySeconds(2000))
    {
      return;
    }
  }

}
```

Figure 3.2.4 Washing Cycle

The washing cycle operation is simple and operates as follows: the tub will spin anti-clockwise for a few revolutions then stall then spin clockwise for a few revolutions, finally reaching stall, this whole cycle would be repeated 2 times. Our goal for the washing cycle was to achieve a similar experience to how a real washing machine would operate. During our washing cycle in the anti-clockwise direction the duty cycle had to be set to a duty that would provide a low tumble spin for a few revolutions, the number of revolutions was determined by the duration of the *delaySeconds*. The most optimal duration was determined to be 3 seconds, which was controlled by the *delaySeconds* variable. As 500 is equivalent to 50% and the duty ranges from 0-1000, the range for anti-clockwise operation is between 0-500, and from 500-1000 is for clockwise operation. In practice the anti-clockwise rotation begins around 450 with the load of the tub, as there isn't enough power to move the tub until then, the vice versa is true for clockwise rotation where it begins at 550. The directions of the motors can be seen below in the figure attached.

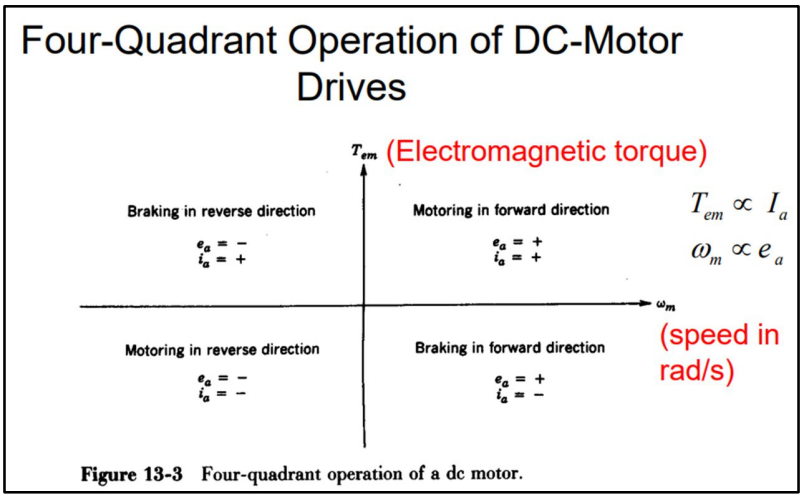


Figure 3.2.4.1 DC motor Operation

Knowing these variables the we calculated the duty cycles of the based on the Vo, as according to the equation $Vo = Vin \cdot D + (-Vin)(1 - D)$. The equivalent duty cycle that was used for the anti-clockwise operation was 420, resulting in 580 for the clockwise direction.

3.2.5 Rinsing Cycle

```
void RINISING_CYCLE()
{
  //Rinising Cycle Operation
  //Kick starts the duty at 5

  Serial.print("Sensor Value: ");
  Serial.println(sensorValue);
  duty = 540;
  update_pwm_bipolar(duty);

  while (tarValue != sensorValue) //within the while loop where tarValue != sensorValue
  {
    if (tarValue > sensorValue)//will increase duty if the tarValue greater than the sensor value
    {
      if (duty < MAX_DUTY) // won't exceed the maximum duty
      {
        duty += 2; //duty will increase by 2,

      }
    }
    else //will decrease duty if the tarValue less than the sensor value
    {
      if (duty > MIN_DUTY) //won't exceed the minimum duty
      {
        duty -= 2; //duty will decrease by 2,
      }
    }

    Serial.print("DUTY: ");
    Serial.println(duty);
    update_pwm_bipolar(duty);

    sensorValue = analogRead(A0); //read sensorValue every cycle
    Serial.print("Sensor Value: ");
    Serial.println(sensorValue);

    if (!delaySeconds(200))
    {
      return;
    }
  }

  //matched tarValue, exit while loop, duty set, run for 10 secs
  update_pwm_bipolar(duty);

  if (!delaySeconds(10000))
  {
    return;
  }

  duty = STALL;

  if (!delaySeconds(1000))
  {
    return;
  }


}
```

Figure 3.2.5 Rinsing Cycle

The aim of the rinsing cycle is to achieve the same speed regardless of the load that is in the tub. This is to replicate the rinsing cycle of an actual washing machine where there would be clothes and it would be full of water, and would be able to maintain the same speed for any load. The load in the tub is measured through the analog pin on the arduino UNO (*A0*). The *sensorValue* that is read from the tub is measured for every cycle within the while loop, the aim of the while loop is to cycle through increasing/ decreasing the duty cycle until the *tarValue* is equal to the *sensorValue.* The *tarValue* is the target voltage we want to achieve in order to maintain a constant speed, for the duty increments the duty is increased by 0.2% with every iteration so that it's easier for the *sensorValue* to match the *tarValue.*

Initially the duty was increasing by 5 but that proved to be too large of an increment where it would bounce up and down within a range for too long before the *sensorValue* matched the *tarValue.* Afterwards the duty increment was set to 0.1% this worked really well when matching the *sensorValue* as there was a much smaller range but took really long before the motor started spinning, thus leading us to use 0.2% increments. The maximum and minimum duty of the rinsing cycle is the maximum duty that we set for the motor as it would be unsafe to exceed these duty cycles.

3.2.6 Spinning Cycle

```
void SPINNING_CYCLE()
{
  //SPINNING CYCLE
  //kick start duty at 550
  duty = 550;
  while (duty != MAX_DUTY) //as long as duty is not equal to max duty
  {
    duty += 10; //increases duty by 10 every cycle
    if (duty > MAX_DUTY) //if exceeds will set to max duty
    {
      duty = MAX_DUTY;
    }
    Serial.print("DUTY: ");
    Serial.println(duty);
    update_pwm_bipolar(duty);
    if (!delaySeconds(2000)) //will increase duty at 2s increments
    {
      return;
    }
  }

  if (!delaySeconds(3000))//run motor at max speed for 3 secs
  {
    return;
  }

  while (duty != STALL) //where duty is not Stall
  {
    duty -= 10; //decrease duty by 10
    if (duty < STALL) //if below stall then sets to stall
    {
      duty = STALL;
    }

    Serial.print("DUTY: ");
    Serial.println(duty);
    update_pwm_bipolar(duty);
    if (!delaySeconds(2000)) //decrease duty ever 2s
    {
      return ;
    }
  }

}
```

Figure 3.2.6 Spinning Cycle

The spinning cycle in a real washing machine would be used to dry the clothes to a dampened state where there isn't an excessive amount of water on the load. To replicate this our spinning cycle is designed to run from *STALL* to the maximum duty cycle, then back down to *STALL.* In our machines case is 70%, while it is possible to increase the duty cycle to 80%, from a functional and hardware standpoint it was more than sufficient to run at 70% as any greater lead to a lot of noise from the tub and rattling. The duty cycle will increment by 1% every cycle which occurs every 2 secs, this means every 2 secs the duty cycle increases by 10, once maximum is reached the machine will run for 3 secs before starting its descent back to *STALL.*

4. Project Management

4.1 Gantt Chart

The Gantt chart breaks down each deliverable into which tasks need to be accomplished within a certain given time. Deliverable 1 is the team proposal which is broken down into DC motor application, Dc motor design concept, application operation, interim report and the presentation. The other two deliverables are the prototype 1 and 2. Prototype 1 consists mainly of designing the mechanical parts for the washing machine, creating a BOM and purchasing those parts, this can be seen in Figure 4.1.1. The main objective is to create a rough design for the deliverable. Prototype 2 is where the final design has to be ready for submission and must be fully operational, this consists of doing the programming for the Arduino, having the operational design and conduction testing on it so that it achieves all the guidelines we have specified. The final report and presentation will cover the entirety of the project difficulties we faced and what we were able to achieve. The Gantt chart can be found in the appendix Figure 4.1.

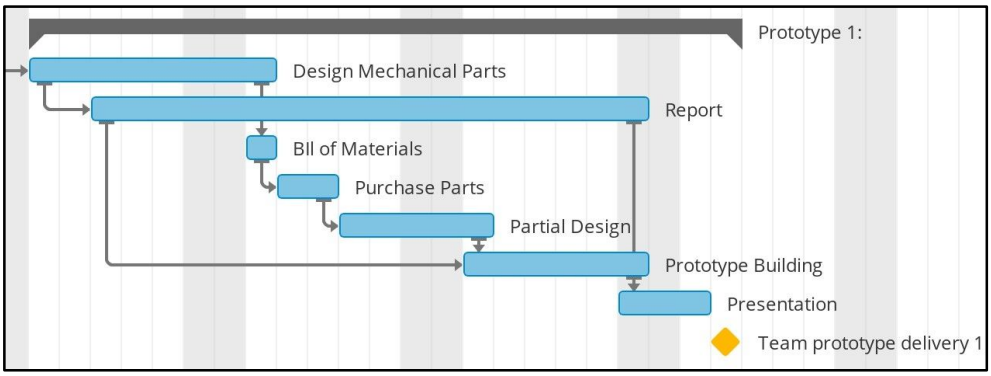


Figure 4.1.1. Prototype 1 Gantt Chart

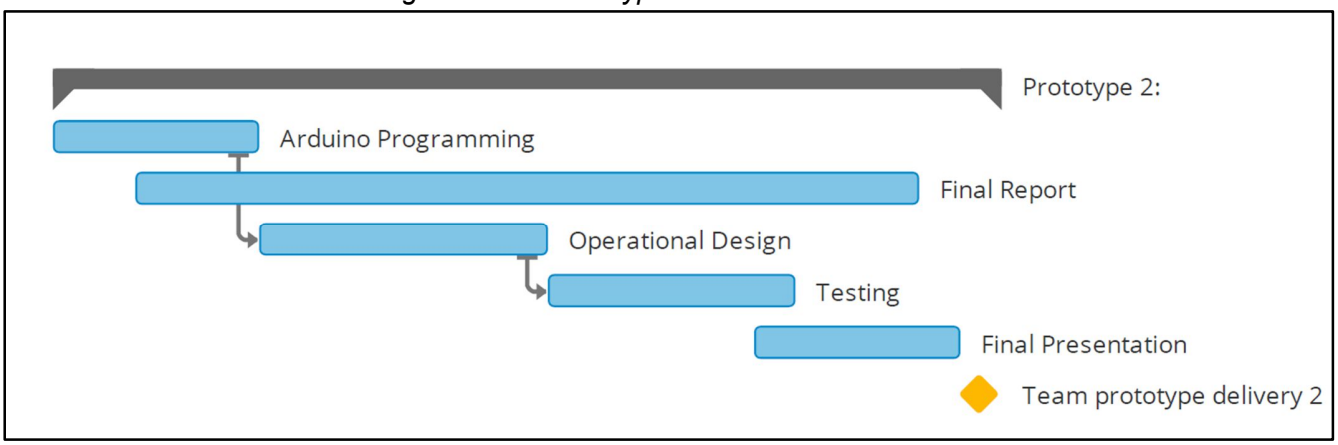


Figure 4.1.2 Prototype 2 Gantt Chart

4.2 Milestones

The milestones are the gold diamond markers, which indicate the major parts of the project are due. These dates were pulled from the project guidelines and are representative of when the PCB is due for fabrication, team proposal, prototype 1 and 2 are due. For the team proposal the majority of the time was spent on the PCB drawing and the design concept. In most of the sections majority of the time is spent on the designing and construction due to us being so far away and having limited in person interaction due to covid-19.

4.2.1 Progression (Prototype 1)

As the software that was used for the Gantt chart was only for a trial period the Gantt chart cannot be updated visually, but the Trello board has been adequate enough to monitor the progress of what has been completed and what is yet to be completed for each deliverable. Each task is based on the Gantt chart which has an expected duration for how long a task should take, the durations are flexible except for those that are dependent on each other such as the BOM and purchasing of parts to build the prototype. As per the submission of this report the team is making substantial progress having collected the PCB and soldered it, built a prototype from rough parts, and has begun testing the PCB. Figure 4.2 in the appendix shows the progress and Gantt chart of Prototype 1 using Excel Spreadsheet.

4.2.2 Progression (Prototype 2)

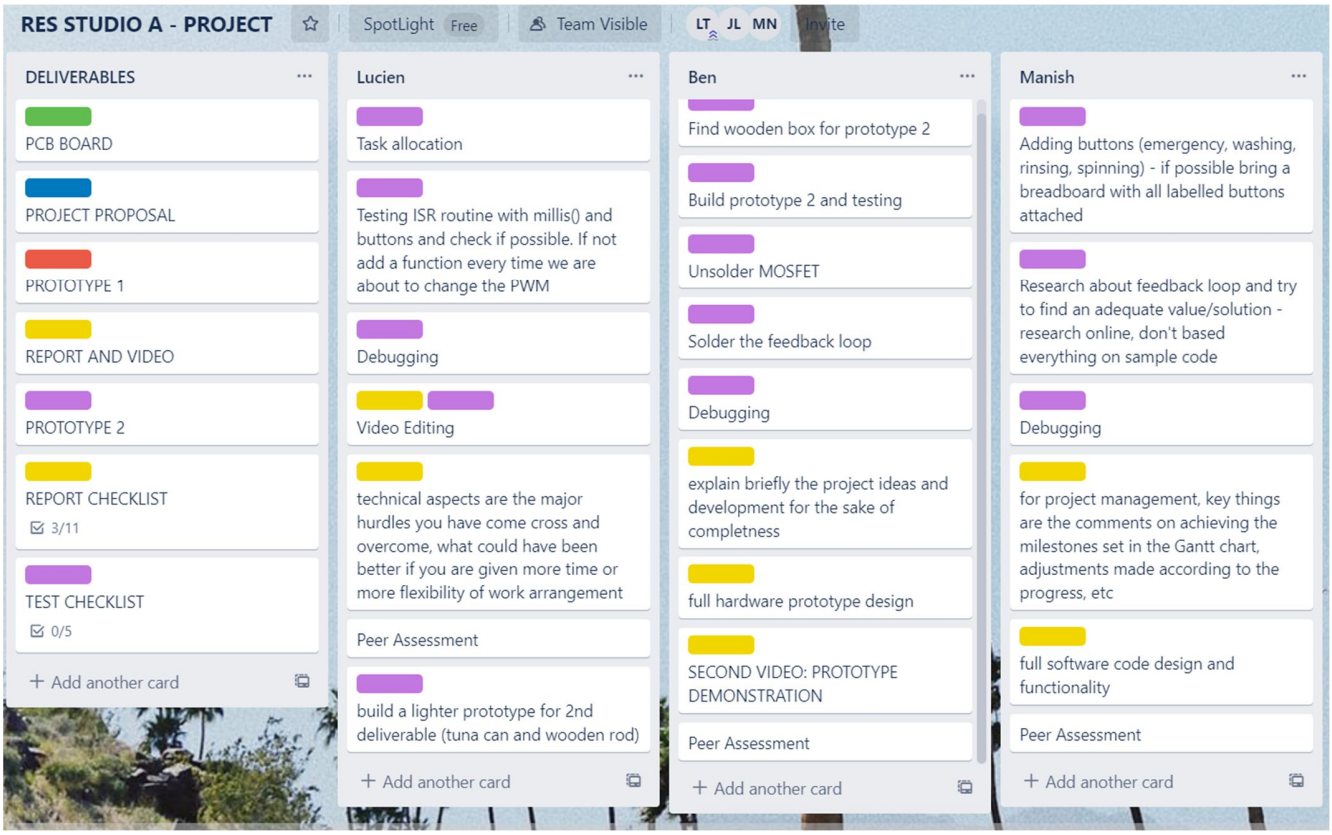


Figure 4.2.2 Trello Board

As the software that was used for the Gantt chart was only for a trial period the Gantt chart cannot be updated visually, but the Trello board has been adequate enough to monitor the progress of what has been completed and what is yet to be completed for each deliverable. The Trello board has been broken down to all the major tasks and when completed transfer to the done list. As this is the end of prototype 2 where all the deliverables must be met, the progress of the project can officially be labelled as complete.

4.3 Challenges

4.3.1 Challenges of Prototype 1

Building the washing machine will prove to be difficult due to the covid-19 situation where we are forced to be in lockdown, but based on recent emails from uts it seems as if the restrictions are being lowered therefore allowing us to interact more. Also delivery of products if not available in store, also of PCB.

One major challenge we faced is the PCB manufacturing as the PCB that was sent over was missing the second layer where ground and other internal connections were missing. This has led to us having to solder our own wire at the back of the board to ensure that all ground pins are connected and avoid having a wire hanging for each ground. Furthermore, one of the laptops was not detecting the kit despite downloading the appropriate firmware and troubleshooting on github. A software challenge that we encountered is the expiration free trial of the Gantt chart program. Excel Spreadsheet has been used as an alternative but it has limitations and is not designed specifically for Gantt chart creation.

After soldering the kit and testing it out, the motor could only rotate in one direction. After checking all connections such as the inputs and outputs, changing the IR2103, and the different PWMs, it was found that one of the diodes was faulty. The diode of the first IR2103 had a voltage of 0.5 while the second one was overloading (OL). In turns, the faulty equipment has affected our results and the way the circuit behaves, challenging us in being meticulous with testing the circuits and ensuring that the state of all components is satisfactory for the circuit to run accurately. Further tests have been run and a faulty MOSFET has been found. While trying to replace the MOSFET, the low heat capacity of the PCB has challenged us.

4.3.2 Challenges of Prototype 2

Several challenges have been encountered during the development of Prototype 2. The major challenge was finding a second faulty MOSFET on the PCB. It impeded the performance of the motor in anticlockwise rotation. The solution to the problem was to unsolder the MOSFET without damaging the PCB like it was previously in Prototype 1. After fixing the hardware part of the project, we found out that our prototype 2 was too heavy and the load was too high for the motor to operate in a wide range. Thus, we have decided to design a smaller prototype with a wooden rod and a smaller tub.

Another challenge that we encountered was meeting each other for task allocation and debugging. We could not rely on one team member to debug both software and hardware obstacles and had to meet. However, each one of us lives 1.5 hours from university and even further away from each others' houses. Thus, meeting at UTS was the best solution, however social distancing and security at UTS has reminded us to keep more distance than needed and has significantly slowed

down our progress. Personal challenges have also risen. One of our team members had to move due to personal reasons and is in the process of moving out during the period of prototype 2 and has delayed us in the timeline expected.

In addition, the implementation of interrupts has failed as we have too many delay functions in our code. The different cycles cannot function without the delay function and therefore we had to give up implementing the ISR routine. Another obstacle was not being able to use the delay function in the ISR routine but using millis function which has deeply affected the other delay functions. On top of that, the Arduino Nano given to us has encountered some problems and new codes cannot be uploaded anymore. Fortunately, several of our team members had their own microcontrollers and could continue the project with not much obstacle. After extensive debugging, all delay functions have been replaced with a user defined delaySeconds function, using millis(). In this case, using interrupts for the emergency button was possible.

4.4 Proactive Plan (Prototype 1)

4.4.1 Progress vs Plan

As mentioned above in the Gantt chart of Prototype 1, we have indeed completed the tasks that were allocated. However, several challenges such as the motor turning only in one direction have troubled us.The team is on track as we have purchased the different part of the motor as well as built the PCB and its connections and coded the three different cycles with emergency button: washing, rinsing and spinning

4.4.1 How is the team tackling challenges?

The team has tackled the different challenges with utmost calm. No one was blamed and no one pointed fingers towards each other. Several meetings have been conducted to allow us to identify the challenges as well as how to solve them and allocate new tasks while considering the timetable of each of us. All team members have agreed and rigorously completed their part and written down what has to be conducted next. The use of Trello board has allowed the team to allocate efficiently each task to each team member and allowed us to judge if each one of us has the same amount of work.

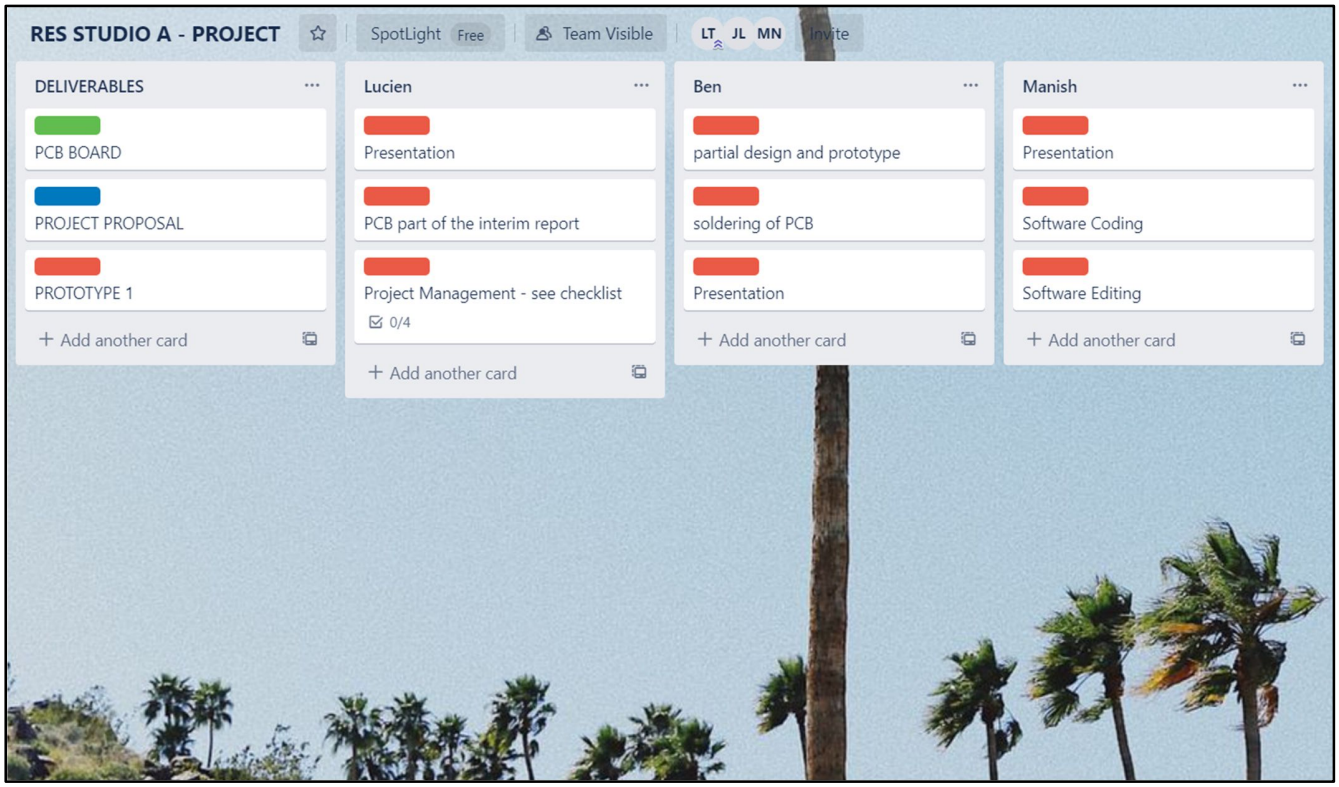


Figure 4.4.1. Trello board with all tasks allocated (Prototype 1)

Furthermore, each team member has several main tasks to complete but also several support tasks. Support tasks have been incorporated in our project management as it is easy to be overwhelmed by the main task. When one team member is in trouble, the person with the support task has to help them and if further help is required, the whole team is working towards the same goal. That tactic has been proven to be effective in both PCB design, hardware and software aspects of the project. Finally, the bond that each team member shares grows stronger by helping each other.

4.4.2 Possible adjustments to get back on track (Prototype 1)

As discussed above, the cardboard box of the prototype was not stable enough and a wood box will be needed to ensure a better performance. The wooden box will not only increase the performance but will also look more esthetically pleasing to the eye.

While all tasks have been completed, the team deemed that another prototype needed to be built with new components purchased by us. It has slightly changed the timing of Prototype 2 however additional effort will be given by each team member and the team strongly believes that it will enhance the quality of the prototype. All components were already purchased and a new prototype is already being built. It reveals that further testing will also be needed and all team members are happy to help each other in those conditions.

5. Learning Experience

5.1 Testing Prototype 1

The mechanical design was completed by connecting the rubber band as a belt between the motor and the rotor, the rotor was connected to the washing machine tub by puncturing a hole in a metal can which served as a washing machine tub and a rubber tube was placed between the motor and generator. The materials for the washing machine were bought from a local arts and crafts store for high quality finish. However, the cardboard box was not as strong as expected, although it was effective enough to keep to the rotor in place.

Firstly, the completed PCB was tested by connecting the output terminal to the motor to ensure that the circuit design was working. Secondly, the duty cycle that generated the maximum torque was calculated by trial and error. Thirdly, the mechanical design and electrical design was connected. While testing the prototype 1, the washing machine did not run at all due to the high friction between the motor and the rotor, so the distance between the two components was reduced from 15cm to 9cm. Additionally when the motor started to spin the washing machine, the rubber band started to move around between the motor and the generator which caused the design to slow down at times. To fix this problem, the rotor had to be adjusted so that it was in parallel with the table. Despite the flaws in the mechanical design, overall prototype 1 ran successfully. For prototype 2, the team agreed that the cardboard box is to be replaced with a more stable design such as a wooden box. The images below show the output waveforms measured on the Labrador. These waveforms were used to test the prototype 1 with 10kHz frequency and from 20% 40% and 50% duty cycle outputs from Vs, Vb and Arduino NANO.

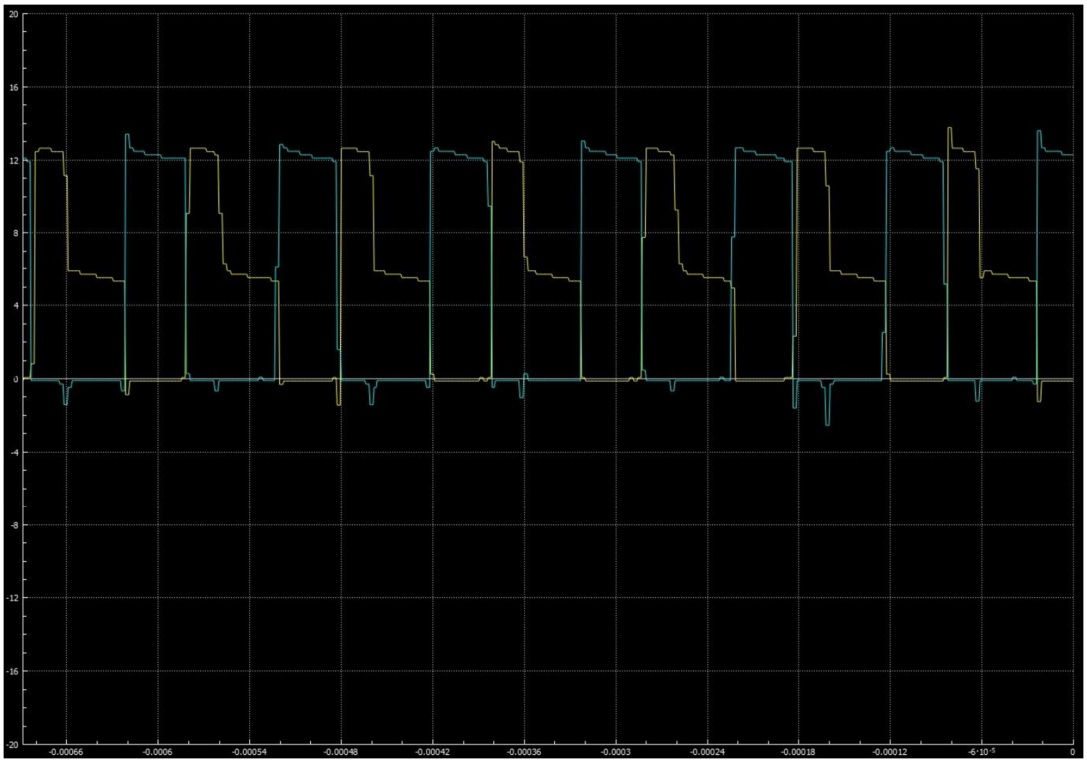


Figure 5.2.1 40% duty cycle, output from IR2103 Vs, stall condition

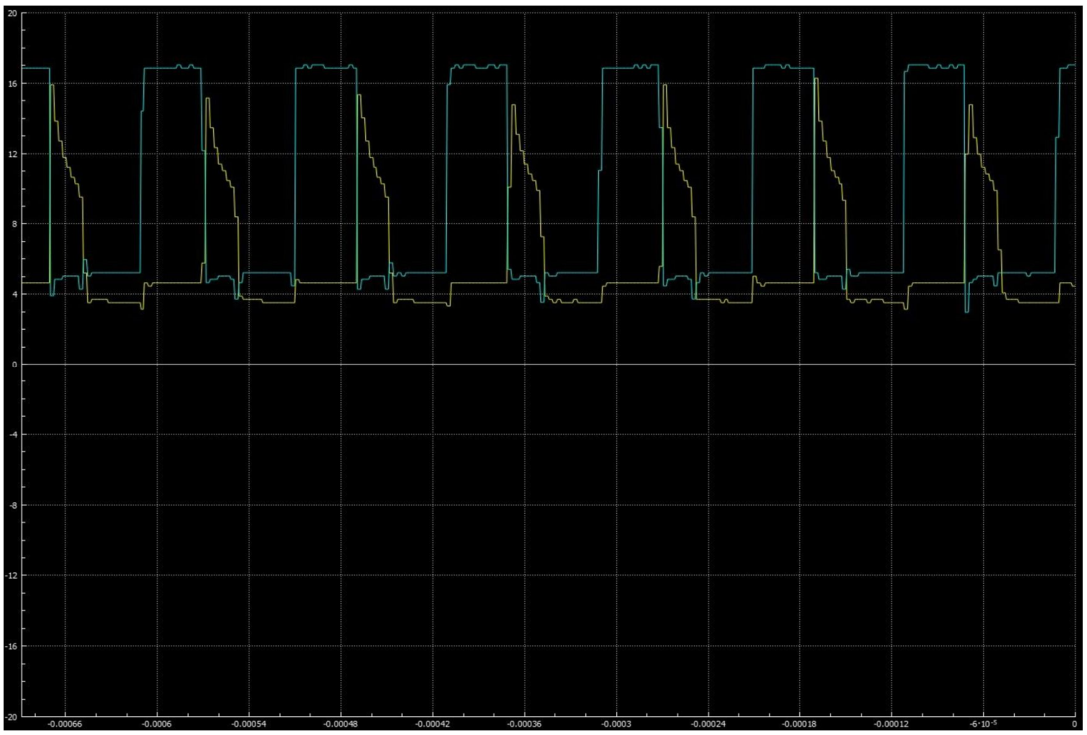


Figure 5.2.2 40% duty cycle, output from IR2103 Vb with -8V offset in both channels

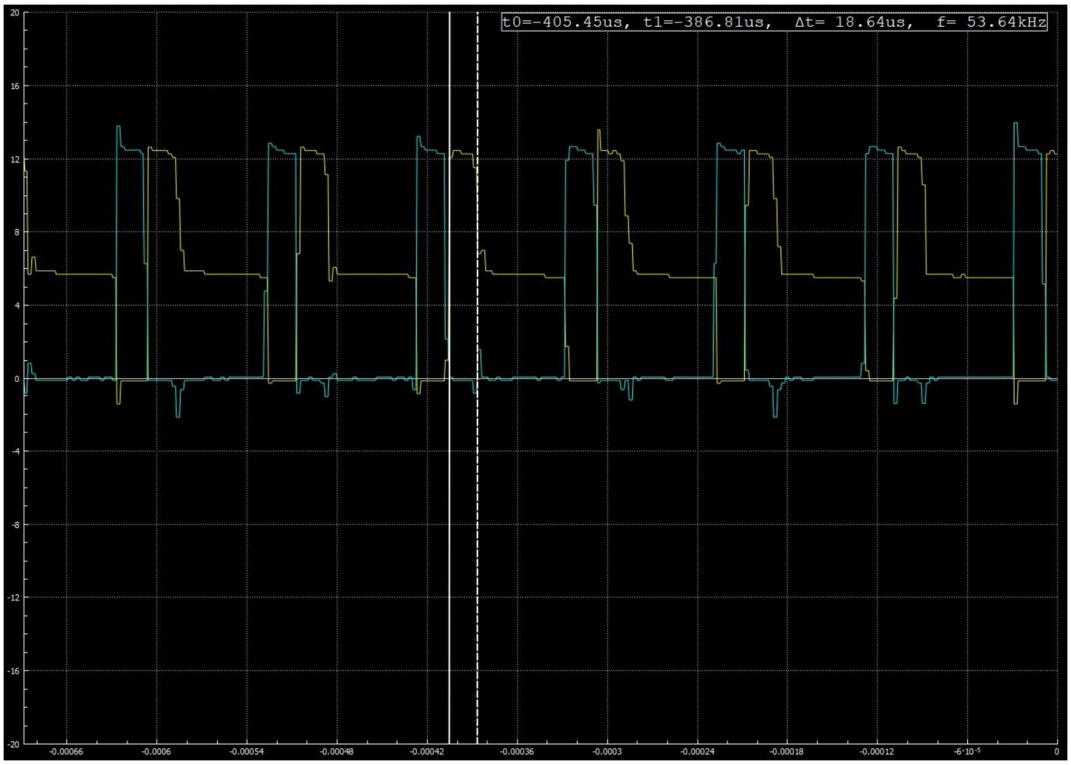


Figure 5.2.2 20% duty cycle, output from IR2103 Vs.

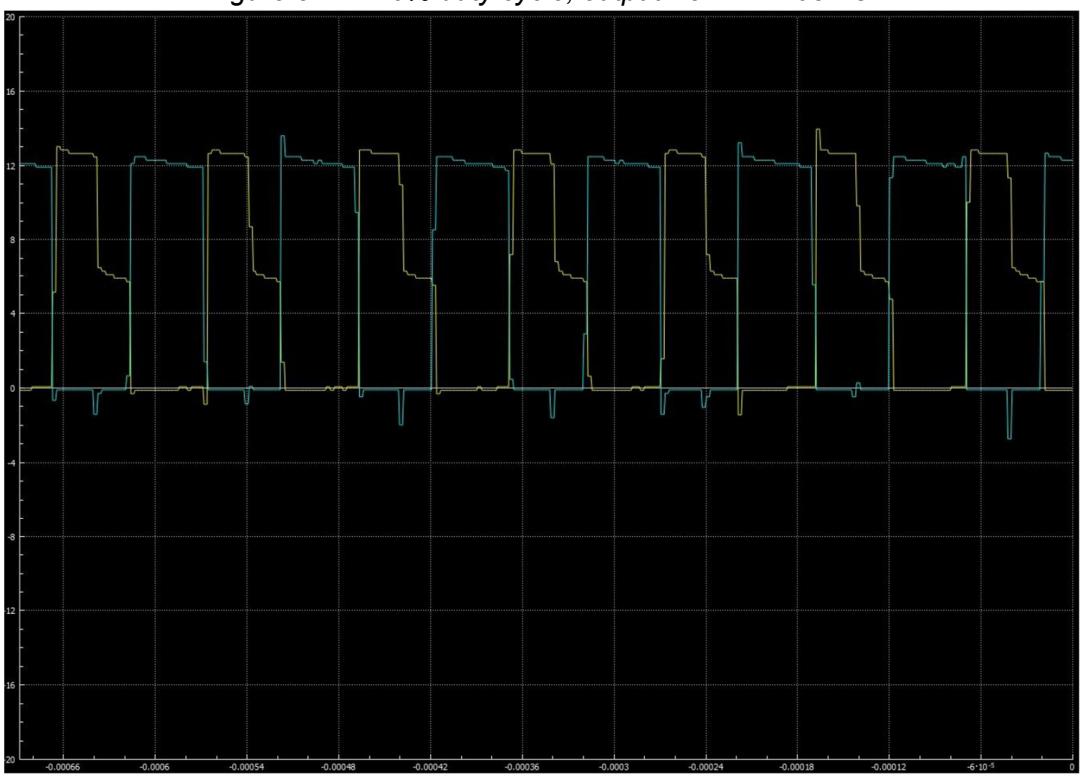


Figure 5.2.3 50% duty cycle, output from IR2103 Vs.

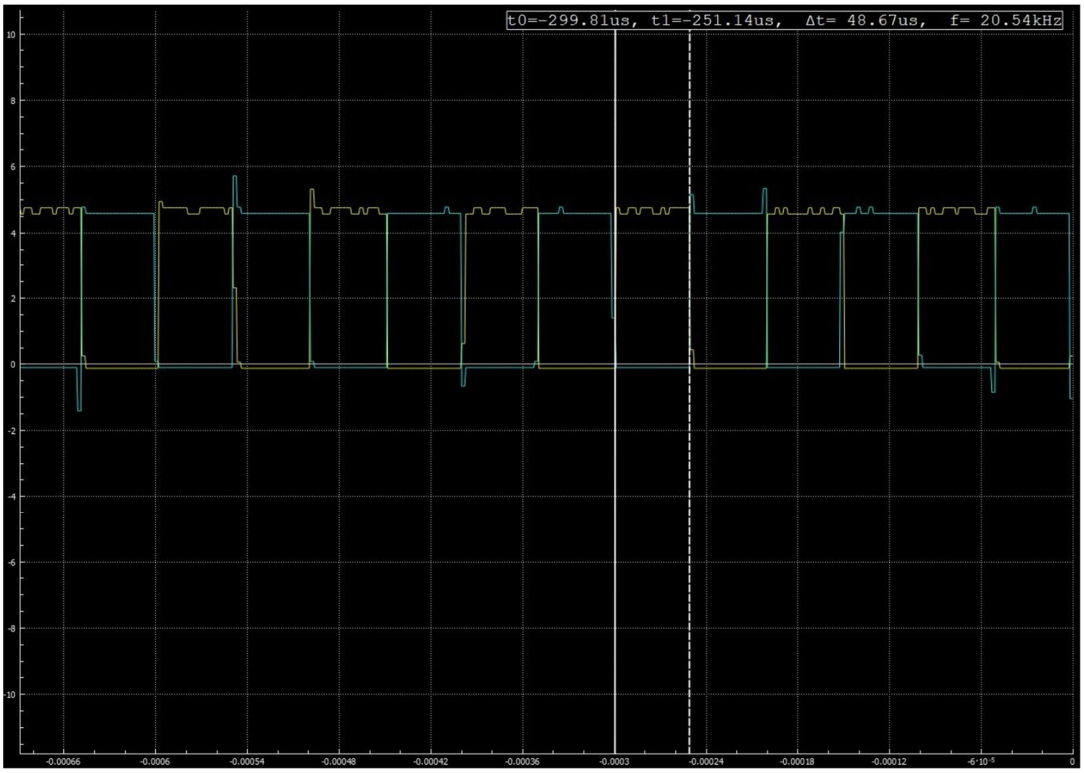


Figure 5.2.3 50% duty cycle, output from Arduino NANO 10kHz.

5.2 Testing Prototype 2

Testing prototype 2 took additional care compared to prototype 1 due to limited resources. After implementing the new design, (highlighted in 2.4.2 Prototype 2), the team decided to meet and work on the project together and debug any problems with the code and the mechanical design. The initial testing was to perfect the washing cycle, spinning cycle and emergency stop without the mechanical load. After getting the code to work with the motor, the mechanical components of the project were integrated to see the effects it made on the motor.

The washing cycle was the most difficult to test since the components on the PCB failed multiple times and in the end the circuit board was replaced with a breadboard. When the motor was tested using PCB, the motor would only spin in one direction and not both directions. The team still does not understand why the circuit board is not working as designed, but replacing the PCB with breadboard fixed all problems.

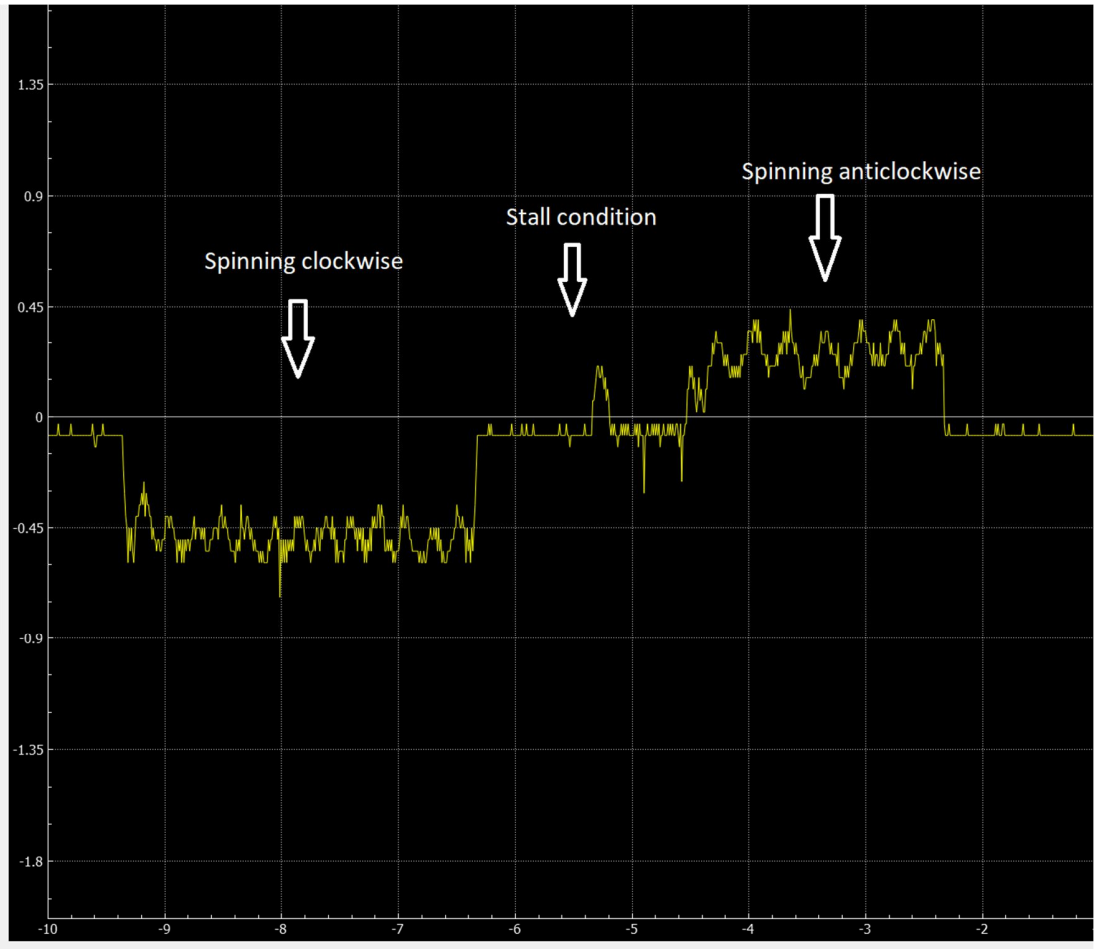


Figure 5.2.1 Washing Cycle waveform, spinning in both direction

Rinsing cycle used a feedback generator to control the constant speed with changing load. When the motor spins the generator connected to the motor by a rubber tube spins at the same speed feeding the voltage back to the arduino. The arduino coding recognises the analogue input and adjusts the duty cycle accordingly. See the image below, showing feedback voltage produced by the generator (see 3.2.4 Washing Cycle for Arduino Coding).

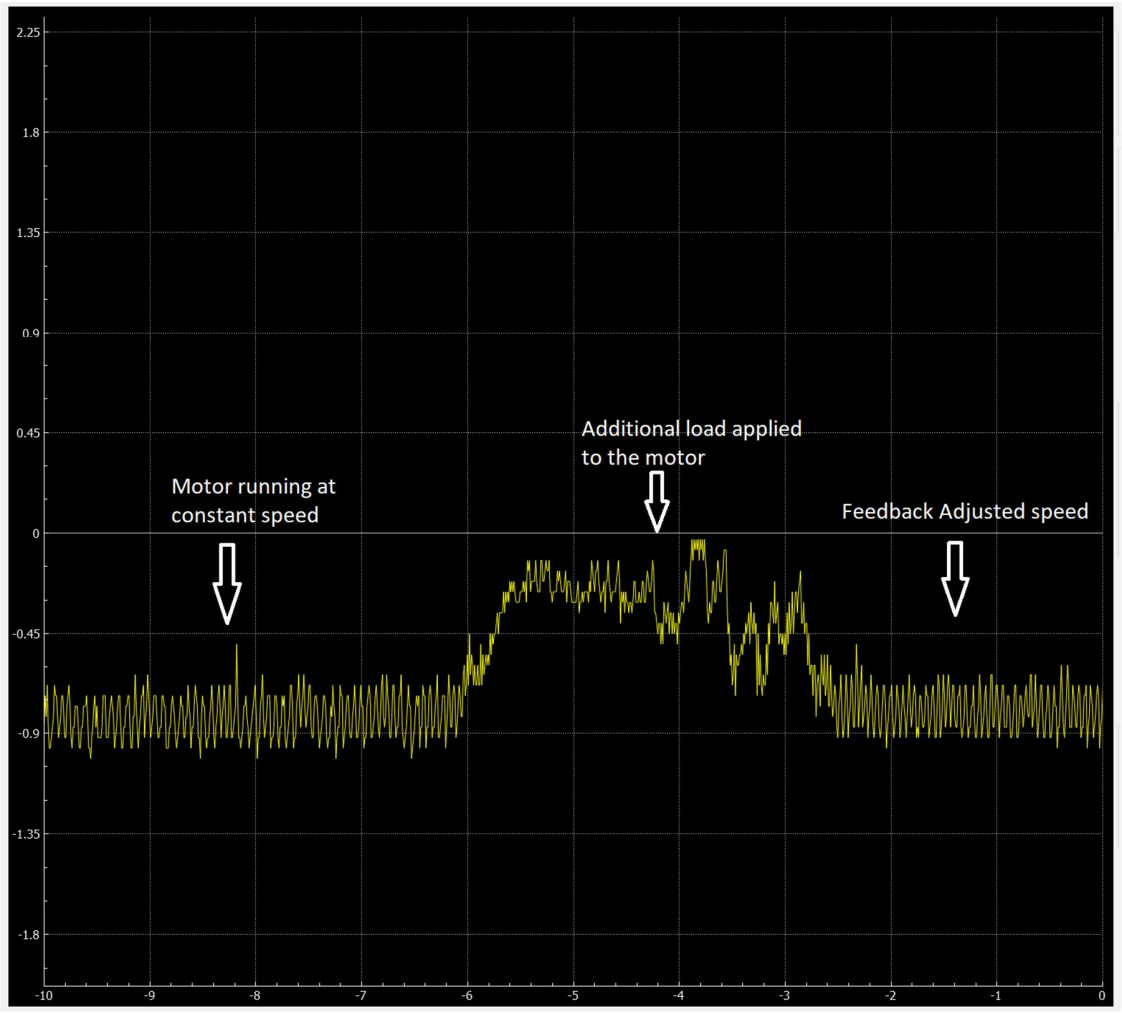


Figure 5.2.2 Rinsing cycle with load applied half way.

Spinning cycle with mechanical load had some problems as the motor load was initially too large to operate the spinning cycle efficiently. This problem was fixed by replacing the mechanical components with lighter and smaller components and a better overall design (see 2.4.2 for prototype 2 design). Spinning cycle was one of the easiest to test and implement as it was just increasing speed with constant load to the maximum speed, then it climbed back down to stall condition. One problem when implementing the function was the interrupt for the emergency button was going off at random times as the spinning cycle was running. This problem was fixed using a diode from 5V power to the emergency button to only allow current to flow one way. The team believes this problem was caused by the charged inductors inside the motor and the generator.

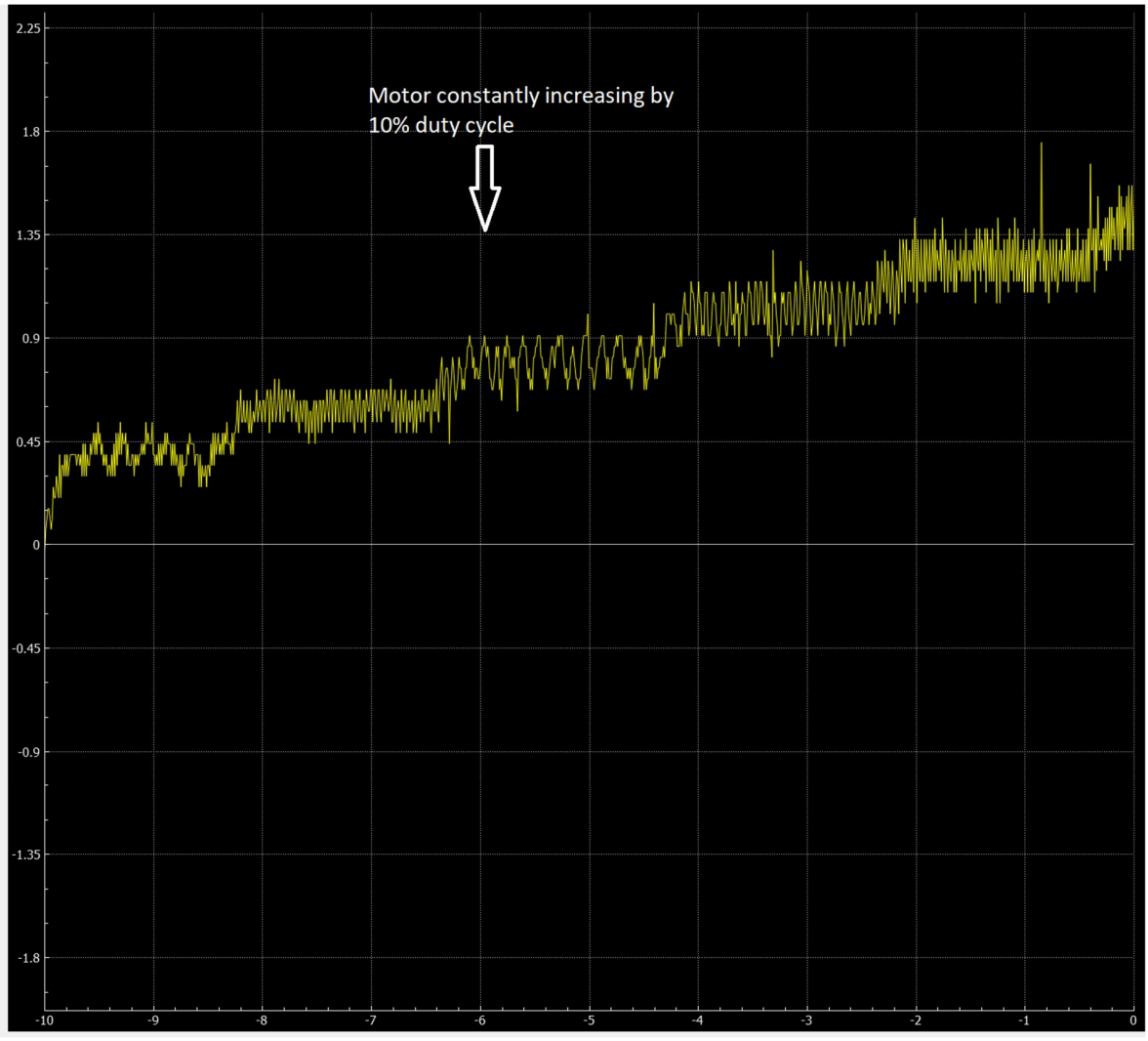


Figure 5.2.3 Spinning cycle waveform, increasing by 10% duty cycle.

5.3 Teamwork & Communication

Deciding on the project application was based on a team decision. Overall there were three possible applications discussed. First was to do an original idea of a wind turbine to generate electricity and convert the electrical power to charge a mobile phone. But seeing as it required extensive research and including the time it takes to practically gather and build the project, the team decided to push for one of the given potential project topics. The Second possible application was a Passenger Lift given in the potential project topics in the assessment guideline. Once again, it was not in the teams best interest to go into the depth of building a working lift. Finally, the second given project in the assessment guide required the least amount of tinkering with non electrical components, in which the team decided to go with the Washing Machine potion.

The team got together quickly and considered multiple options carefully before deciding on the project. This was achieved with effective communication using video chat and social media websites as the main platform to connect. As a team we also helped each other out on understanding the main objective of the project and by taking initiative, we were able to achieve further in the time given.

5.4 New Learnings

Participating in a group of three in this project has helped us split the workload in between each one of us. A lot of topics were covered in the project itself and bringing everything together, we have listed three big headings in our learning:

- PCB Design - PCB design is an important part of the electrical degree in our opinion. Researching and troubleshooting in both KiCad and Altium has helped us understand the different ways to design a PCB as well as its different techniques. To design a PCB, being able to understand each component's datasheet, creating footprints, symbols and schematic as well as routing are essential skills to learn.
- Software - coding a microcontroller is a piece of skill needed to learn in electronics. Using interrupts and researching how to replace the delay function with the millis function was a major hurdle. Furthermore, coding and debugging has also helped us understand the logic behind the circuit.
- Hardware - We have learned that theory and experimental results are really different and dealing with those situations has taught us to problem solve under time constraint. Building the prototype and soldering the PCB has also given us a tremendous amount of insight in terms of full-bridge converter and motor operations.

5.5 Possible Improvements

The mechanical design from prototype 1 to prototype 2 was supposed to change from a thin cardboard box to a wooden box, however no suitable wooden box was found so a stronger cardboard box was used instead. Although it worked as expected the team agrees that it would have worked more effectively if additional weight was carried by the mechanical design itself. Additionally, the distance of the rotor and the motor had to be changed multiple times due to the change in weight of the washing machine which had a simple solution but the team exhausted other options before considering changing the material itself.

When changing the material for the mechanical design, aesthetics of the design was not heavily considered, hence the design could have looked better than a brown box and a tin can attached to a wooden pole. Although it does not contribute at all to the working principle of the washing machine, it was something the team considered to be important.

Once again the printed circuit board did not work as intended which made the Q3 MOSFET burn out, and an IR2103 faulty. So the team decided it would be best to use the purchased circuit elements on a breadboard, instead of using the third PCB and breaking the circuit components again. All functionality of the circuit was included in one breadboard as shown in figure 2.4.2.2, these include, full-bridge converter, buttons and the feedback circuit. Although it was efficient to do this, using a PCB would definitely make the aesthetics of the design much better.

6. Peer Assessment

6.1 Peer Assessment from Lucien Tran

Team Member	**Feedback**
Jeong Bin Lee	Ben has worked really hard and knows how to reach for help when needed. He was in charge of the hardware section and has outdone himself in debugging and finding materials to constantly improve the prototype every week.
Manish Nath	Manish has always handed in all his tasks on time and is also happy to help when asked to. He was in charge of the software coding and project management with myself and we have worked well together, reviewing each other's work.
General Feedback on the team	I personally think that each one of us has put in as much work as each other. It was a pleasure working with both of them. Workload was distributed evenly and communication was easy as we knew each other since first year of university and worked together in several projects.

6.2 Peer Assessment from Jeong Bin Lee

Team Member	**Feedback**
Lucien Tran	Lucien was in charge of project management and circuit design. Always gets the job done as soon as possible and helps out the rest of the team as needed. Lucien has excellent knowledge on the project in terms of electrical, mechanical and software design. He is always looking for additional ways to improve the project design and efficiency. Great team player and project manager.
Manish Nath	Manish was in charge of Arduino programming. Always finishes the given work on time and provides helpful feedback on all aspects of the project. Manish dedicated himself to make the project work until the end and has always been a great team player. He has consistently improved his coding skills and project management skills throughout the semester.
General feedback on the team	Overall the team works very efficiently and effectively. As engineering students, I believe we learnt as much as we could from this subject in terms of teamwork, communication and all the technical skills provided in this subject. It has been a great experience being part of this team.

6.3 Peer Assessment from Manish Nath

Team Member	**Feedback**
Lucien Tran	Lucien always works to deliver the best of his ability, striving for the best quality work. If assistance is ever needed during the project he is always quick to clarify and help out. Proved to be an integral part of the team when it came to PCB design, software support and being the project manager. His knowledge of electrical and software really meant he was able to help both aspects of the project.
Jeong Bin Lee	Ben was in charge of the hardware for the project, and proved to be really good at what he did, always providing new insight into what issues could be causing the hardware to malfunction and furthermore. Ben always delivered high quality work at a timely manner and was able to assist with other aspects of the project as well. Also provided creative ways to improve the hardware and software.
General feedback on the team	Overall the team works really effectively together always delivering and striving for the best, and possible ways to improve. It has been a great opportunity to work with them over the semester, and having previously known each other communication was very easy, such as pointing out errors, ways to improve, how to split the workload and more, thus allowing us to function optimally.

Appendix

Figure 2.3.1.3 Schematic of the PCB design

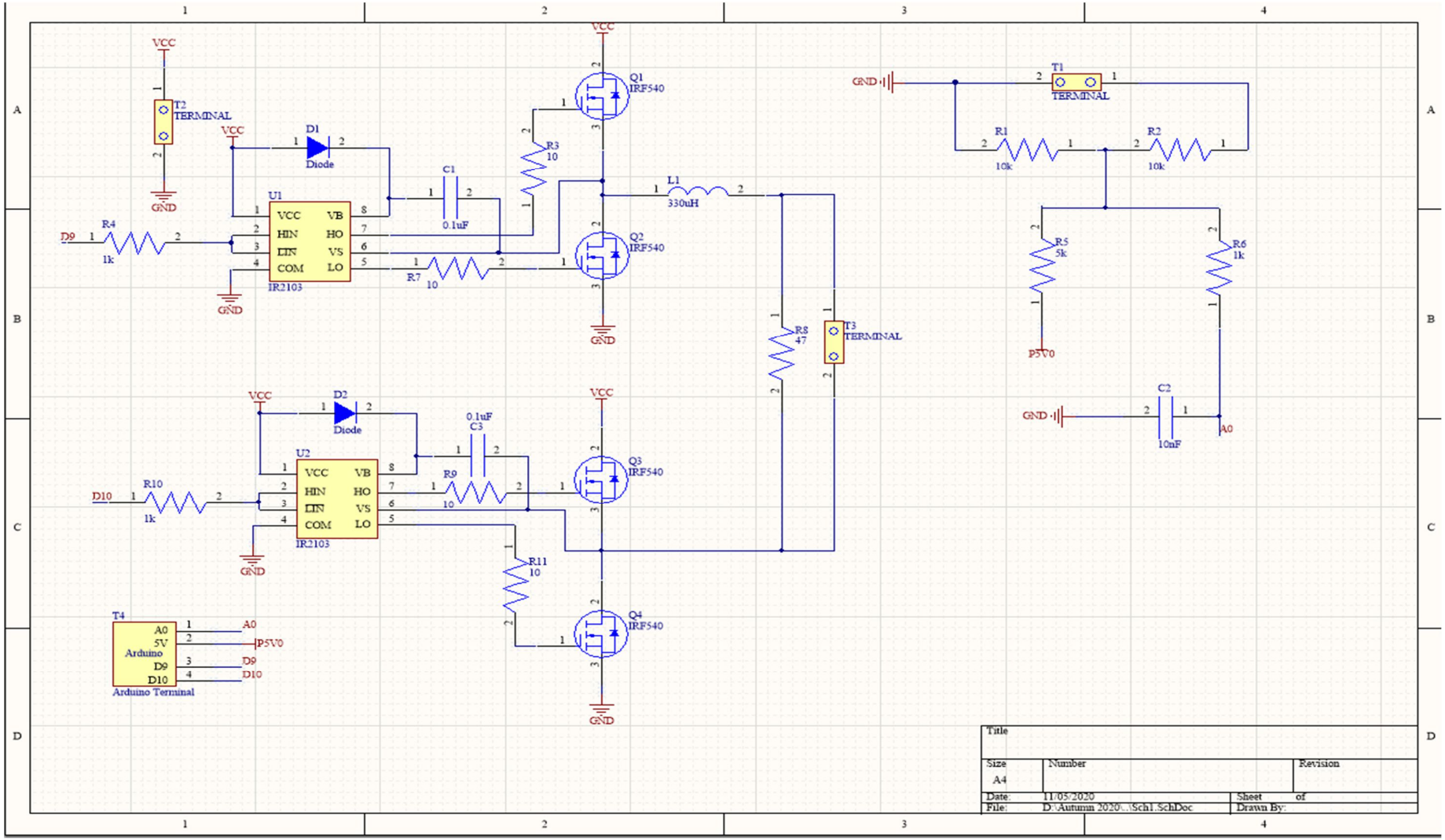


Figure 3 Arduino Code


```
// DEF
#define F_CLK 16000000UL // Main clock frequency
#define F_PWM 10000 //PWM frequency in Hz
#define N 1 // Timer1 clcok prescaler

//VALUE DEF
#define MAX_DUTY 700
#define MIN_DUTY 300
#define STALL 500
#define DUTY1 420
#define DUTY2 580

#define TIMER_TOP (unsigned long) (F_CLK/(2*N*F_PWM)-1)

//PIN DEF
#define PIN1_PWM 9
#define PIN2_PWM 10
#define PIN_LED 13


// Global variables
int duty = STALL;
int32_t ocr;
int sensorValue; //weight of the load from A0
int interruptValue; //Value of when the emergency stop button is pressed
int tarValue = 475; //target voltage value for the rinising cycle which is weight dependant
int washingState; //button 1 state
int rinsingState; //button 2 state
int spinningState; //button 3 state
float voltage;
int interruptPin = 2; //Pin connected to the interrupt
volatile byte stopState = LOW; //state of switch
boolean interrupted = false;

int washingComplete;
int rinsingComplete;
int spinningComplete;

volatile unsigned long timer; // how long the program has been running for



//CONSTANT PINS
const int washingButton = 3; //washing cycle
const int rinsingButton = 4; //Rinsing
const int spinningButton = 6; //Spinning


//Init
void setup()
{
  //I/O pins config
  pinMode(PIN_LED, OUTPUT);
  digitalWrite(PIN_LED, HIGH);
  pinMode(PIN1_PWM, OUTPUT);
  pinMode(PIN2_PWM, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(interruptPin), STOP, RISING); //interrupt used as emergency stop
  //(interruptPin is Pin 2, STOP is the function thats called, RISING is the mode which triggers the interrupt)
  pinMode(washingButton, INPUT);
  pinMode(rinsingButton, INPUT);
  pinMode(spinningButton, INPUT);

  Serial.begin(9600);

  washingComplete = 0;
  rinsingComplete = 0;
  spinningComplete = 0;


  //Timer config
  //DO NOT CHANGE THESE
  TCCR1A = _BV(COM1A1) | _BV(COM1A0) | _BV(COM1B1); //OC1A and OC1B
  TCCR1B = _BV(WGM13) | _BV(CS10); // phase and frequency corect PWM, N =1
  ICR1 = TIMER_TOP;
}

void loop()
```

```
{
  sensorValue = analogRead(A0); //reading analog value
  interruptValue = digitalRead(2); //reading digital value from pin2
  delaySeconds(100);
  voltage = sensorValue * (5 / 1023); //actual voltage of the PIN

  washingState = digitalRead(washingButton); //State value of washingButton whether high or low
  rinsingState = digitalRead(rinsingButton); //"^"
  spinningState = digitalRead(spinningButton); //"^"

  Serial.print("A0 Value:"); //prints sensorvalue on the serial monitor
  Serial.println(sensorValue);
  Serial.print("duty: "); //prints duty on the serial monitor
  Serial.println(duty);
  Serial.print("Interrupt Value:"); // value of whether the interrupt is pressed or not
  Serial.println(interruptValue);


  //if statements for sequential cycles
  if (washingComplete == 1) //if washing cycle has completed then proceed
  {
    rinsingState = HIGH;
    //Rinsig Button state to high to trigger rinsing cycle to begin after washing cycle is complete
    washingComplete = 0; //resets washingComplete value back to 0
  }

  if (rinsingComplete == 2) //if rinsing cycle has completed then proceed
  {
    spinningState = HIGH;
    //SPinning Button state to high to trigger spinning cycle to begin after rinsing cycle is complete
    rinsingComplete = 0;
  }


  if (washingState == HIGH) //HIGH means on
  {
    Serial.println("WASHING CYCLE");
    washingComplete = 1; //when washing triggered it sets to value to true for the if statement
    WASHING_CYCLE(); //calls washing cycle
    Serial.println("END WASHING CYCLE");


  }
  else if (rinsingState == HIGH)
  {
    Serial.println("RINSING CYCLE");
    rinsingComplete = 2; //sets value to true for if statment
    RINISING_CYCLE(); //calls rinsing cycle
    discharge(); //stalls the motor after rinsing
    Serial.println("END RINSING CYCLE");


  }
  else if (spinningState == HIGH)
  {
    Serial.println("SPINNIG CYCLE");
    SPINNING_CYCLE(); //calls spinning cycle
    Serial.println("END SPINNING CYCLE");
  }
  else {
    Serial.println(" NO CYCLE");
    if (!delaySeconds(1000))
    {
      return; //returns back to the start of void loop
    }
  }

}

void STOP() //when digitalInterrupt is triggered STOP function is called
{
  if (digitalRead(2)) //if the value of Pin2 is true = 1, then run interrupt
    // only works if button pressed not for random triggers
  {
    interrupted = true; //changes value of interrupted for delaySeconds function

    //Resets all the Cycle Completion values
```

```
      washingComplete = 0;
      rinsingComplete = 0;
      spinningComplete = 0;
    }
  }

  //function generates two complementary PWM signals
  void update_pwm_bipolar(unsigned int duty_new)
  {
    ocr = duty_new * TIMER_TOP / 1000;
    OCR1A = ocr;
    OCR1B = ocr;
  }

  void WASHING_CYCLE()
  {
    //Washing cycle operation
    for (int i = 0; i <= 1; i++) //will run the washing cycle 2 times
    {
      duty = DUTY1; //Anti - Clockwise for 3 secs at duty1
      update_pwm_bipolar(duty);
      Serial.print("DUTY: ");
      Serial.println(duty);
      if (!delaySeconds(3000)) //run for 3 secs
      {
        return;
      }

      duty = STALL; //Stall for 1 sec
      update_pwm_bipolar(duty);
      Serial.print("DUTY: ");
      Serial.println(duty);
      if (!delaySeconds(1000))
      {
        return;
      }

      duty = DUTY2; //Clockwise for 3 secs at duty2
      update_pwm_bipolar(duty);
      Serial.print("DUTY: ");
      Serial.println(duty);
      if (!delaySeconds(3000))
      {
        return;
      }

      duty = STALL; //Stall for 2 second
      update_pwm_bipolar(duty);
      Serial.print("DUTY: ");
      Serial.println(duty);
      if (!delaySeconds(2000))
      {
        return;
      }
    }

  }

  void RINISING_CYCLE()
  {
    //Rinising Cycle Operation
    //Kick starts the duty at 5

    Serial.print("Sensor Value: ");
    Serial.println(sensorValue);
    duty = 540;
    update_pwm_bipolar(duty);

    while (tarValue != sensorValue) //within the while loop where tarValue != sensorValue
    {
      if (tarValue > sensorValue)//will increase duty if the tarValue greater than the sensor value
      {
        if (duty < MAX_DUTY) // won't exceed the maximum duty
        {
          duty += 2; //duty will increase by 2,

        }
```

```
      }
      else //will decrease duty if the tarValue less than the sensor value
      {
        if (duty > MIN_DUTY) //won't exceed the minimum duty
        {
          duty -= 2; //duty will decrease by 2,
        }
      }

      Serial.print("DUTY: ");
      Serial.println(duty);
      update_pwm_bipolar(duty);

      sensorValue = analogRead(A0); //read sensorValue every cycle
      Serial.print("Sensor Value: ");
      Serial.println(sensorValue);

      if (!delaySeconds(200))
      {
        return;
      }
    }

    //matched tarValue, exit while loop, duty set, run for 10 secs
    update_pwm_bipolar(duty);

    if (!delaySeconds(10000))
    {
      return;
    }

    duty = STALL;

    if (!delaySeconds(1000))
    {
      return;
    }


  }

  void SPINNING_CYCLE()
  {
    //SPINNING CYCLE
    //kick start duty at 550
    duty = 550;
    while (duty != MAX_DUTY) //as long as duty is not equal to max duty
    {
      duty += 10; //increases duty by 10 every cycle
      if (duty > MAX_DUTY) //if exceeds will set to max duty
      {
        duty = MAX_DUTY;
      }
      Serial.print("DUTY: ");
      Serial.println(duty);
      update_pwm_bipolar(duty);
      if (!delaySeconds(2000)) //will increase duty at 2s increments
      {
        return;
      }
    }

    if (!delaySeconds(3000))//run motor at max speed for 3 secs
    {
      return;
    }

    while (duty != STALL) //where duty is not Stall
    {
      duty -= 10; //decrease duty by 10
      if (duty < STALL) //if below stall then sets to stall
      {
        duty = STALL;
      }

      Serial.print("DUTY: ");
      Serial.println(duty);
```

```
      update_pwm_bipolar(duty);
      if (!delaySeconds(2000)) //decrease duty ever 2s
      {
        return ;
      }
    }

  }

  boolean delaySeconds(int interval)
  {
    timer = millis();
    while ((interrupted == false) && (millis() < (timer + interval)))
    {
    }
    if (interrupted)
    {
      Serial.println("interrupted worked!");
      interrupted = !interrupted;
      duty = STALL;
      update_pwm_bipolar(duty);
      washingState = rinsingState = spinningState = LOW;
      Serial.println("BACK TO STALL");
      return false;
    }
    else
    {
      return true;
    }
  }

  void discharge()
  {
    if (sensorValue != 0)
    {
      interrupted = true;
    }
  }
```

FIgure 7.1 PROTOTYPE 1 CODE PROTOTYPE 1

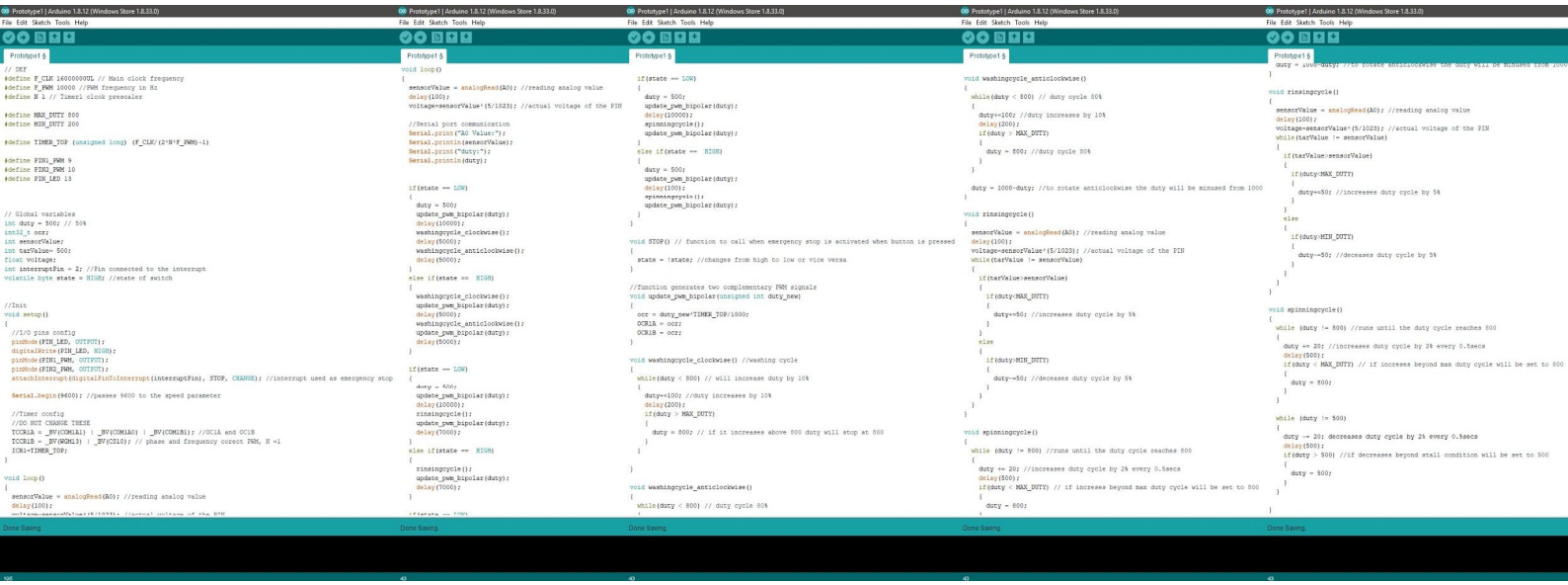


Figure 7.2 Rinsing Cycle PROTOTYPE 1

```
void rinsingcycle()
{
  sensorValue = analogRead(A0); //reading analog value
  delay(100);
  voltage=sensorValue*(5/1023); //actual voltage of the PIN
  while(tarValue != sensorValue)
  {
    if(tarValue>sensorValue)
    {
      if(duty<MAX_DUTY)
      {
        duty+=50; //increases duty cycle by 5%
      }
    }
    else
    {
      if(duty>MIN_DUTY)
      {
        duty-=50; //deceases duty cycle by 5%
      }
    }
  }
}
```

Figure 7.3 Spinning Cycle PROTOTYPE 1

```
void spinningcycle()
{
  while (duty != 800) //runs until the duty cycle reaches 800
  {
    duty += 20; //increases duty cycle by 2% every 0.5secs
    delay(500);
    if(duty < MAX_DUTY) // if increses beyond max duty cycle will be set to 800
    {
      duty = 800;
    }
  }

  while (duty != 500)
  {
    duty -= 20; decreases duty cycle by 2% every 0.5secs
    delay(500);
    if(duty > 500) //if decreases beyond stall condition will be set to 500
    {
      duty = 500;
    }
  }

}
```

FIgure 7.4 Washing Cycle Anticlockwise PROTOTYPE 1

```
void washingcycle_anticlockwise()
{
  while(duty < 800) // duty cycle 80%
  {
    duty+=100; //duty increases by 10%
    delay(200);
    if(duty > MAX_DUTY)
    {
      duty = 800; //duty cycle 80%
    }
  }

  duty = 1000-duty; //to rotate anticlockwise the duty will be minused from 1000
}
```

Figure 7.5 Washing Cycle Clockwise PROTOTYPE 1

```
void washingcycle_clockwise() //washing cycle
{
  while(duty < 800) // will increase duty by 10%
  {
    duty+=100; //duty increases by 10%
    delay(200);
    if(duty > MAX_DUTY)
    {
      duty = 800; // if it increases above 800 duty will stop at 800
    }
  }

}
```

Figure 7.6 Stop Function PROTOTYPE 1

```
void STOP() // function to call when emergency stop is activated when button is pressed
{
  state = !state; //changes from high to low or vice versa
}
```

Figure 3.2.7 Arduino Uno Schematics

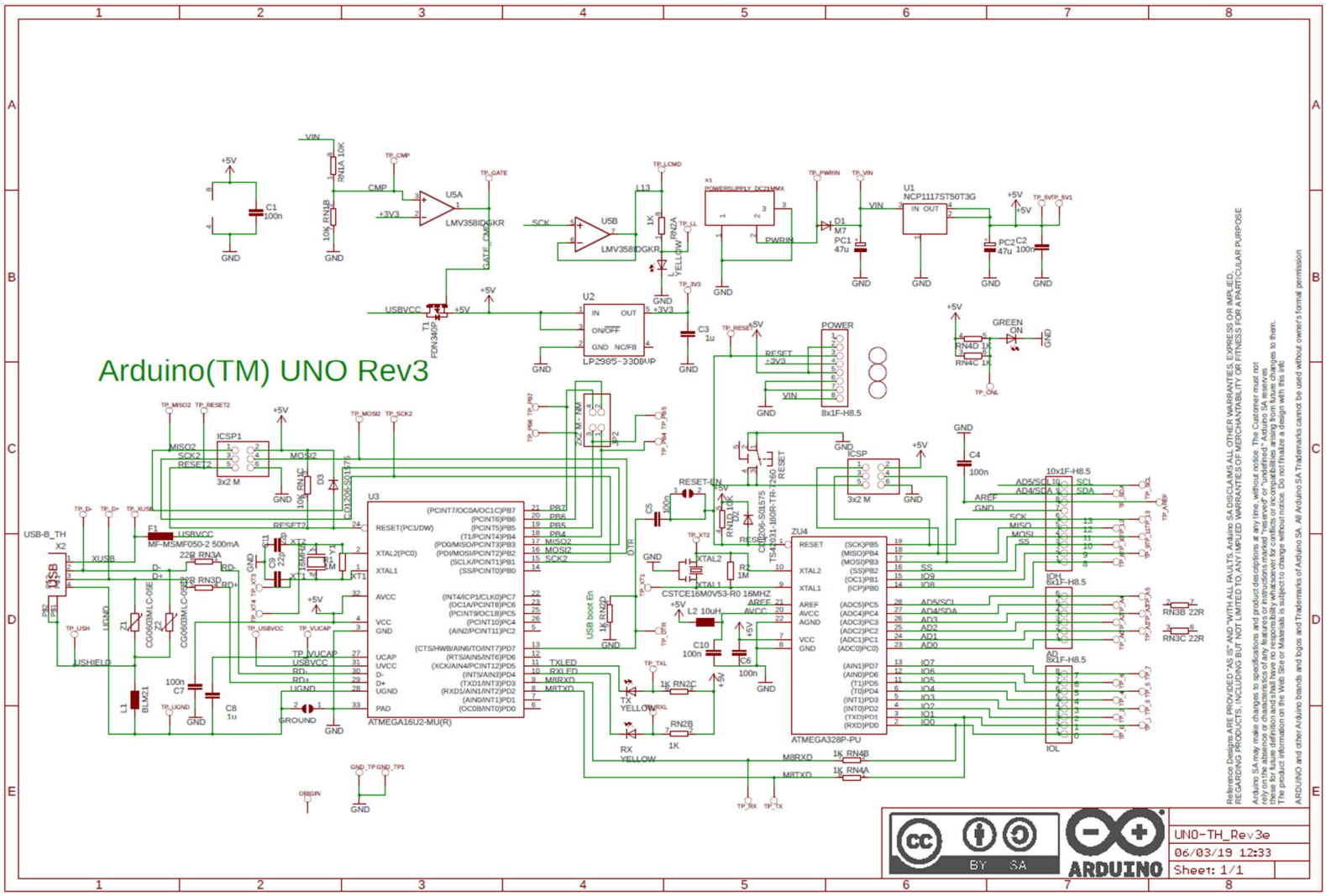


Figure 4.1 Gantt Chart, Milestones and Progression (Project Proposal)

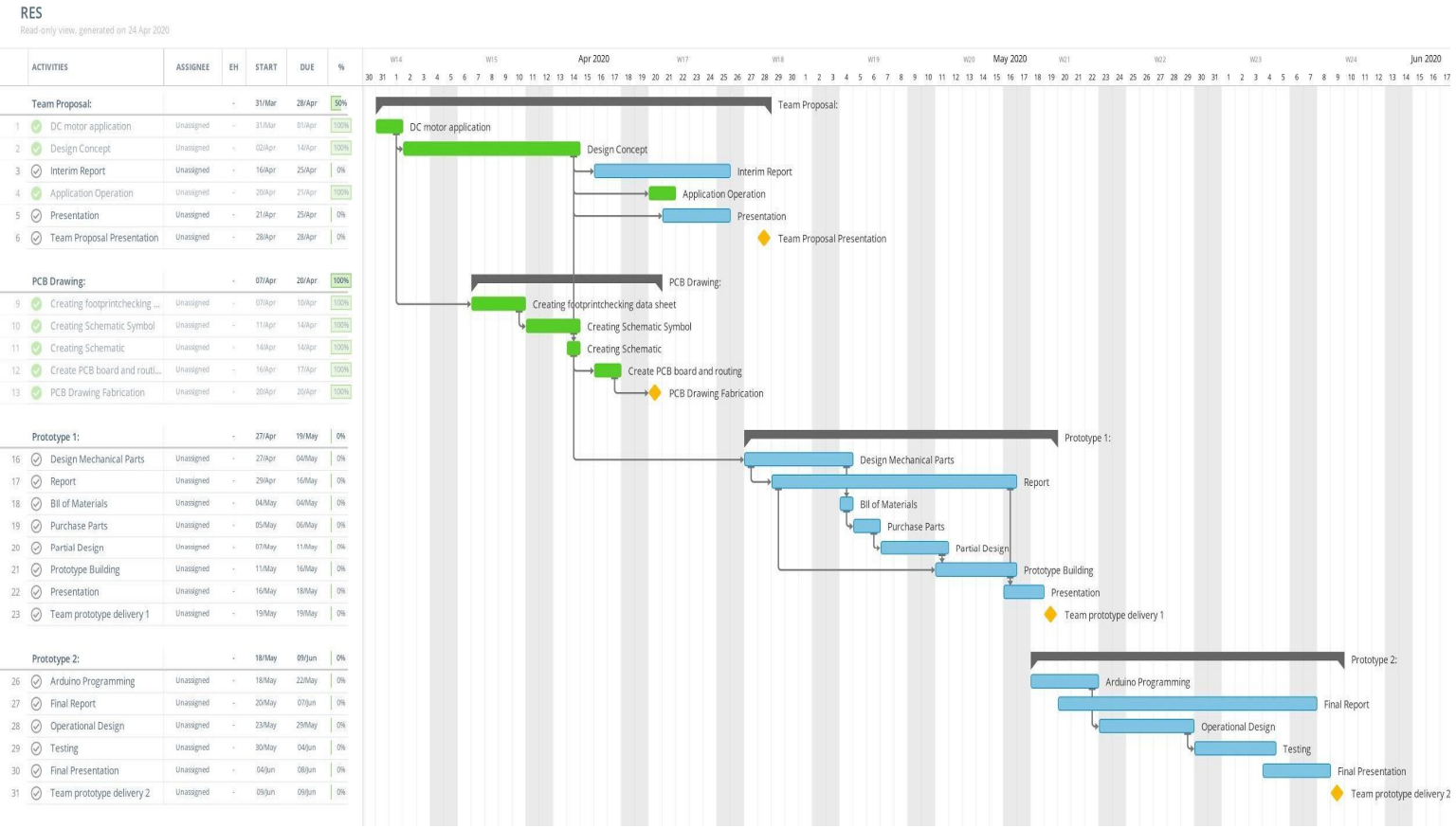


Figure 4.2 Gantt Chart of Prototype 1 - Excel