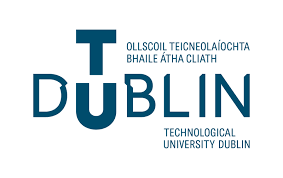
******

**SCHOOL OF ELECTRICAL AND ELECTRONIC ENGINEERING**

**BEng Tech (Ord) Electronics and Communications Engineering Level 7**

**Programme Code: (DT008)**

**<YEAR 3>**

**Name of Supervisor (Derek Gillmor)**

**Project Logbook**

**Student Name** Sajjad Ullah, **\_\_**C17344483**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Project Title** Arduino Robot \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DECLARATION**

This report is submitted in partial fulfilment of the requirements of the Degree in Electronic and Communications Engineering (DT008) of the Technological University Dublin.

I, the undersigned, declare that this report is entirely my own written work, except where otherwise accredited, and that it has not been submitted for a degree or other award to any other university or institution.

A close up of a logo

Description automatically generated

Signature of student: ………………………………….

Date: …………20/04/20………………………………

**Contents**

[Introduction 5](#_Toc37103435)

[Proposed solution 5](#_Toc37103436)

[Research 6](#_Toc37103437)

[H-Bridge L293D 6](#_Toc37103438)

[Pulse Width Modulation (PWM) 8](#_Toc37103439)

[Ultrasonic sensor HC-SR04 9](#_Toc37103440)

[Arduino uno 10](#_Toc37103441)

[ESP8266 Node MCU 11](#_Toc37103442)

[Week 1 – Single wheel simulation 12](#_Toc37103443)

[Week 2 – Wheels simulation 14](#_Toc37103444)

[Week 3 –Ultrasonic sensors simulation 18](#_Toc37103445)

[Week 4 -Soldering onto proto shield 23](#_Toc37103446)

[Week 5 – Hardware Testing 25](#_Toc37103447)

[Week 6 – WiFi module Testing 26](#_Toc37103448)

[Week 7 – Soldering onto copper strip board 29](#_Toc37103449)

[Week 8 – Code improvements 30](#_Toc37103450)

[Week 9 – Testing the Arduino robot with Wi-Fi 32](#_Toc37103451)

[Proposed modifications 35](#_Toc37103452)

[Alternative solutions 35](#_Toc37103453)

[Conclusion 35](#_Toc37103454)

[References 36](#_Toc37103455)

**Table of Figures**

[Figure 1 Example of an Operating Environment 5](file:///C:\Users\sajja\Documents\1.AProjest%20code\Reports%5e%5e%5e%5e%5e%5e%5e%5e8%5e%5e%5e\DT008_3%20Final%20Report%20Sajjad_Ullah.mm%5bLogbook%5d.docx#_Toc37103456)

[Figure 2 H-Bridge circuit [1] 6](#_Toc37103457)

[Figure 3 L23d pinout [2] 7](#_Toc37103458)

[Figure 4 Pulse Width Modulation graph [4] 8](#_Toc37103459)

[Figure 5 HC-SR04 pinout [6] 9](#_Toc37103460)

[Figure 6 Working principle of HC-SR04 [6] 9](#_Toc37103461)

[Figure 7 Arduino uno pinout [7] 10](#_Toc37103462)

[Figure 8 ESP8266 Mode MCU pinout [9] 11](#_Toc37103463)

[Figure 9 Simulation of single wheel 12](#_Toc37103464)

[Figure 10 Test code for single wheel 13](#_Toc37103465)

[Figure 11 Simulation of two wheels 14](#_Toc37103466)

[Figure 12 Class for Motor Direction 15](#_Toc37103467)

[Figure 13 Function behaviours for Class Motor Direction 16](#_Toc37103468)

[Figure 14 Testing both wheels using class object 17](#_Toc37103469)

[Figure 15 Hardware testing of both wheels 17](#_Toc37103470)

[Figure 16 Simulation for 4x ultrasonic sensors 18](#_Toc37103471)

[Figure 17 Class for Distance Sensor 18](#_Toc37103472)

[Figure 18 Function behaviours for Class Distance Sensor 19](#_Toc37103473)

[Figure 19 defining pins and creating sensor objects 20](#_Toc37103474)

[Figure 20 Setting pin modes and using sensor class 20](#_Toc37103475)

[Figure 21 Getting distance measurements on multiple HC-SR04 21](#_Toc37103476)

[Figure 22 Distance results on multiple HC-SR04 21](#_Toc37103477)

[Figure 23 real distance value(Left) and returned value(Right) 22](#_Toc37103478)

[Figure 24 de-soldered amicus18 companion shield 23](#_Toc37103479)

[Figure 25 soldered components on new proto shield 24](#_Toc37103480)

[Figure 26 Code for parallel movement against a wall 25](#_Toc37103481)

[Figure 27 Top and side view (Left & Middle) of setup and Testing (Right) 25](#_Toc37103482)

[Figure 28 modified HTML webserver code 26](#_Toc37103483)

[Figure 29 ESP webserver(Left) and Serial monitor result (Right) 26](#_Toc37103484)

[Figure 30 ESP webserver code [6] 28](#_Toc37103485)

[Figure 31 Top view of final copper stripboard 29](#_Toc37103486)

[Figure 32 Bottom view of final copper stripboard 29](#_Toc37103487)

[Figure 33 Added functions in MotorDirection class 30](#_Toc37103488)

[Figure 34 behaviour for new functions 30](#_Toc37103489)

[Figure 35 modified setDutycycle 31](#_Toc37103490)

[Figure 36 Testing the new functions 31](#_Toc37103491)

[Figure 37 The setup Before(left) shifting and After(Right) shifting 31](#_Toc37103492)

[Figure 38 Top view of finished robot 32](#_Toc37103493)

[Figure 39 Side view of finished robot 33](#_Toc37103494)

[Figure 40 Serial Monitor input test [10] 34](#_Toc37103495)

# Introduction

In this logbook we will be keeping track of the progress made during the Arduino Robot project.

The project is an Arduino enabled robot capable of navigating and enclosed area such as a warehouse, for the purpose of reaching a desired destination and travelling back with the desired contents. Online services that will be used to achieve this are the Arduino Integrated Development Environment (IDE) and the Tinkercad design software.

# Proposed solution

The solution proposed makes use of Arduino hardware e.g. Boards & shields and the Arduino IDE to create a two wheeled robot capable of navigating an enclosed area with the utilization of ultrasonic sensors to detect walls, shelfs and obstacle detection. Initial work on creating code for the wheels and verifying its intended behaviour will be done in Tinkercad where hardware simulation will be the quickest route to achieving the aims for that aspect of the project. A grid system is to be used for identifying unique shelf sections.

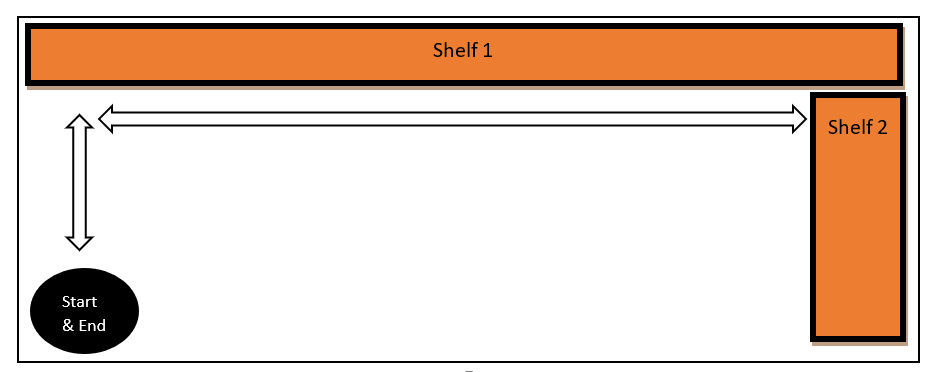


Figure 1 Example of an Operating Environment

Using the figure 1 layout as a basic implementation the robot will start and end at the same designated spot, once a user has entered in the desired shelf coordinate, it will begin using its ultrasonic sensors at regular intervals for the purpose of identifying its relative distance from walls, ensuring it maintains a parallel path along the shelfs until a time when it identifies a turn it must make. After it makes this turn it will continue along this pathway until the correct shelf location has been reached.

The proposed dimensions of the environment in Figure 1 is 3 meters x 2 meters.

# Research

Insight on how to implement parts of the project such as the wheels, ultrasonic sensors and wifi was done by viewing the work of people who have posted their schematic diagrams and code on the Arduino community project hub for others to view. For instances where hardware component information was required, the manufacturers data sheet was obtained.

## H-Bridge L293D

To start we began by getting information on the pinout of the l233d and created a table with descriptions on the function of each pin from information obtained from *microcontrollerslab.com*.

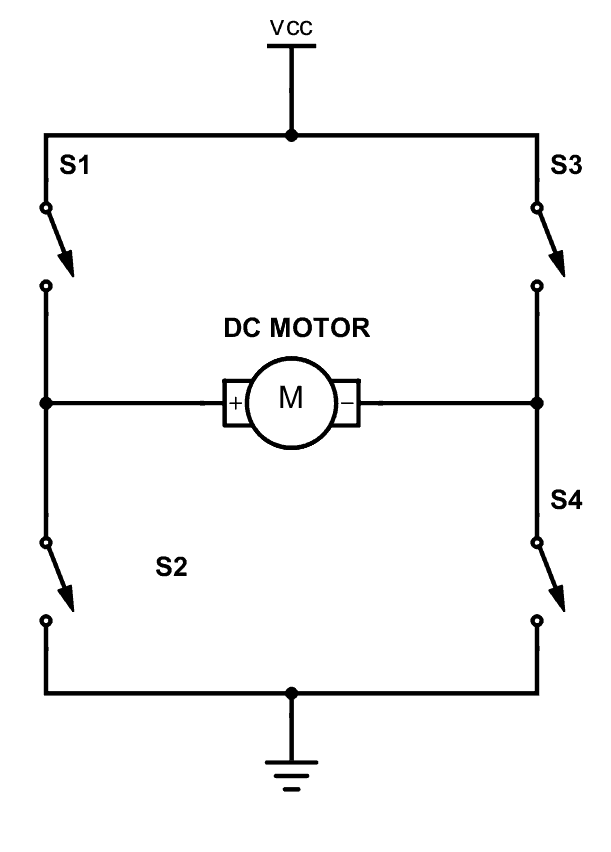


Figure 2 H-Bridge circuit [1]

The internal workings of the l293d contain H-bridge circuits like the one shown in figure 2 that are used to control the direction of spin. For example, the circuit here would require S1 and S4 to close for the wheels to spin in one direction and S3 and S2 fit it to spin on the reverse direction [1].

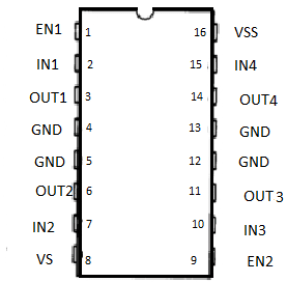


Figure 3 L23d pinout [2]

The pinout id the chip is shown in Figure 3 with short descriptions about the purpose of each pin in Table 1.

Table 1 L293d pinout descriptions [3]

|  |  |
| --- | --- |
| **Pin-[Name]** | **Description of pin** |
| 1 -Enable Left | When High/Low turns on/off pins 3 & 6. |
| 2 -Input 1 | When High, current goes through output 1. |
| 3 -output 1 | Should be connected to a motor terminal |
| 6 -output 2 | Should be connected to a motor terminal |
| 7 -Input 2 | Shen High, current goes through output 2. |
| 8 -Vcc2 | This is the power supply to the motors |
| 16-Vcc1 | This is the power supply to the chip.[should be 5v] |
| 15-Input 4 | When High, current goes through output 4. |
| 14-Output 4 | Should be connected to a motor terminal |
| 11-output 3 | Should be connected to a motor terminal |
| 10-Input 3 | When High, current goes through output 3. |
| 9-Enable Right | When High/Low turns on/off pins 11 & 14. |
| 4,5,12,13-GND | Ground connector pins |

Now that the purpose of each pin on the l293d are known we can make informed pin connections to connect and control two wheels using this chip.

## Pulse Width Modulation (PWM)

So far, we can only control the direction of spin with the l293d but not the speed of rotation, this will be done with PWM techniques.

Pulse width modulation is a method for producing analogue results with digital techniques, the PWM method works by producing a square wave where the ratio of time on and time off can produce any voltage value between 5 volts and 0 volts on the Arduino PWM compatible output [4].

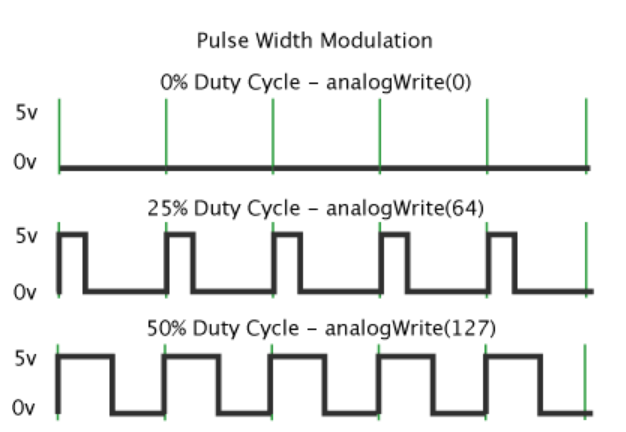


Figure 4 Pulse Width Modulation graph [4]

Figure 4 shows how PWM is implemented in the Arduino board, a desired duty cycle where the signal is only on for 25% of the period will result in 25% of the power to be sent to the output. This would require an *analogueWrite(64)* command which is calculated by:

, *where*

This command is a useful method for employing variable voltages to the wheels without having to utilise different voltage sources.

## Ultrasonic sensor HC-SR04

The sensor type HC-SR04 uses ultrasonic sound to produce distance measurements, its measuring angle is 15 degrees and was chosen for its ability to detect distances of up to 400cm [5] for which the small-scale operating environment shown in figure 1 would be ideal.

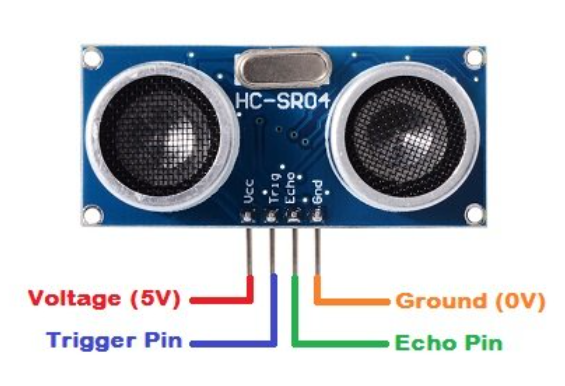


Figure 5 HC-SR04 pinout [6]

Table 2 HC-SR04 pinout descriptions [6]

|  |  |
| --- | --- |
| **Pin** | **Description** |
| VCC | 5v power for the device |
| Trig | Sends an echo signal |
| Echo | Detects the reflected echo signal |
| GND | Ground pin |

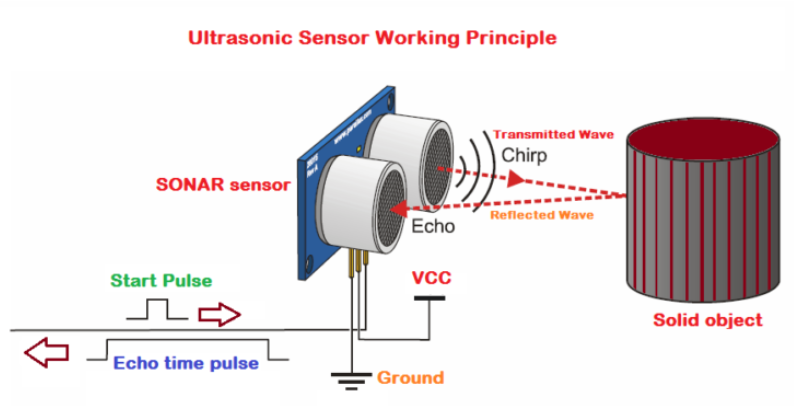


Figure 6 Working principle of HC-SR04 [6]

The sensor operates at 40 MHz with two main parts transmitter(Trig) and receiver(Echo), where the first is responsible for transmitting the wave out to an object for which the pin must be kept HIGH for 10 µs to initiate [6]. The next steps are listed below:

The waves that travel out at 40 MHz will be in 8 cycle sonic bursts.

These bursts that are detected by the receiver and can be converted into a distance measurement.

Conversion is possible because we know the speed at which the waves travel is the speed of sound and the time it takes to be received is the echo time so multiplying these values give the round trip distance but we divide by two to get the object distance.

## Arduino uno

The Arduino uno was an ideal board to start the project with by testing the sensors and the l293d with wheels on it. This is because its capabilities and pin functionality was already known since it was used during several lab activities in the past years.

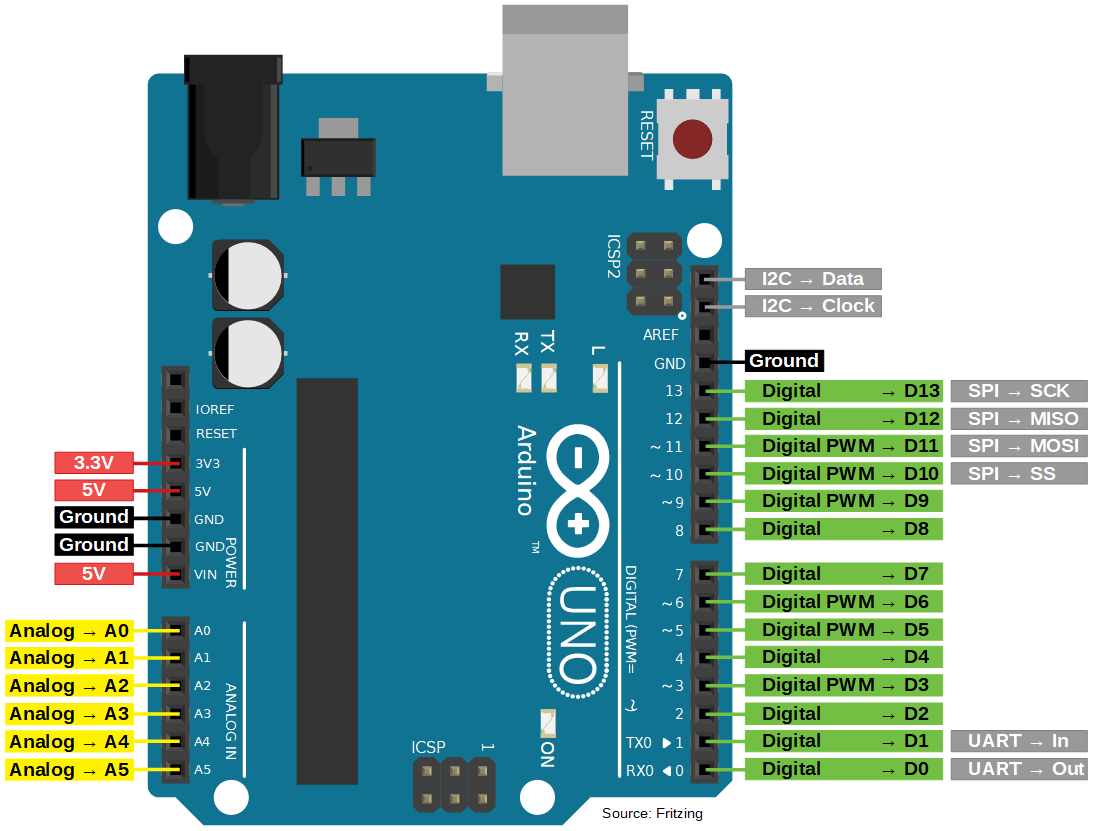


Figure 7 Arduino uno pinout [7]

From Figure 9 we can see it has 14 digital pins and 6 analogue pins which can also be used as digital pins. This is plenty of pins for testing the multiple components we plan on using.

## ESP8266 Node MCU

Towards the end of the project a decision was made to switch from a standard Arduino uno to the ESP8266 which contains onboard Wi-Fi capabilities but while it has 16 General Purpose Input/output (GPIO) pins only eleven can be used for input/output (I/O) functions [8].

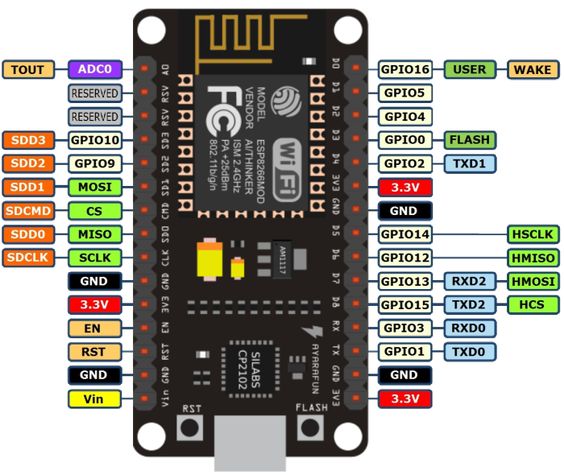


Figure 8 ESP8266 Mode MCU pinout [9]

When setting pin modes on the ESP the pin names have alias names in the Arduino IDE which are the ones that must be used, this is particularly true for the TX and RX pins if they are to be used as I/O pins, shown in table 3.

Table 3 ESP8266 pin mapping for IDE [8]

|  |  |  |
| --- | --- | --- |
| **Pin name** | **label** | **Alias Name in Arduino IDE** |
| D0 | GPIO 16 | D0 |
| D1 | GPIO 5 | D1 |
| D2 | GPIO 4 | D2 |
| D3 | GPIO 0 | D3 |
| D4 | GPIO 2 | D4 |
| D5 | GPIO 14 | D5 |
| D6 | GPIO 12 | D6 |
| D7 | GPIO 13 | D7 |
| D8 | GPIO 15 | D8 |
| RX | GPIO 3 | D9 |
| TX | GPIO 1 | D10 |

# Week 1 – Single wheel simulation

The project was started by testing one motor connected to the Arduino through the bridge. The aim of week one was to start work on the wheels by creating code for the L293D H-bridge chip that would be controlling the two wheels.

The connections made are listed in Table 4 and shown in Figure 9.

Table 4 Configuration for wheel testing

|  |  |  |
| --- | --- | --- |
| **Pins of l293d** | **Arduino Pin** | **Motor** |
| 8 | 3.3v |  |
| 16 | 5v |  |
| 4 | GND |  |
| 5 | GND |  |
| 3 |  | positive terminal of motor |
| 6 |  | negative terminal of motor |
| 2 | Digital pin 3 |  |
| 7 | Digital pin 4 |  |
| 1 | Digital pin 2 |  |

We signed up to Tinkercad and created a new project where it was possible to connect virtual components to a virtual Arduino board and hardware simulation of the code could be achieved.

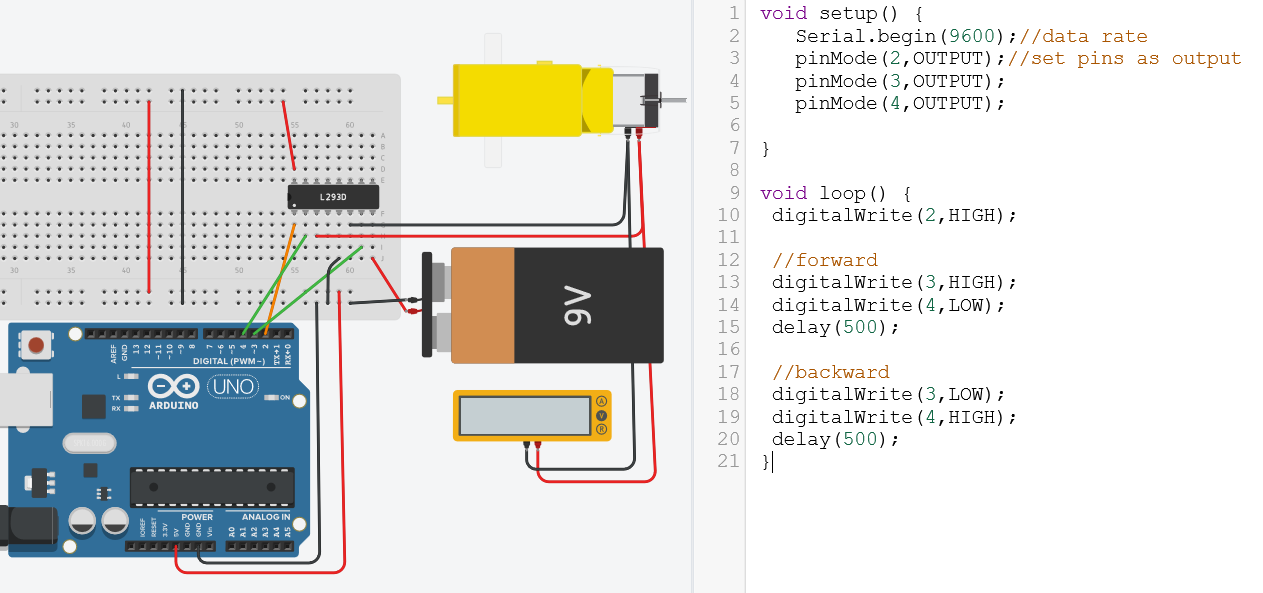


Figure 9 Simulation of single wheel

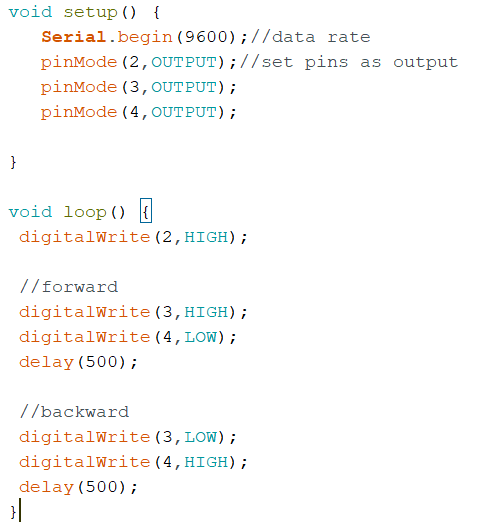


Figure 10 Test code for single wheel

Figure 10 code description: The code will set the data rate and set pin 2 as output in the setup. Then in the loop it will turn on pin 2 (the left side of the chip), pin 3 is set high and pin 4 low to turn the wheel forwards with a small delay of 500ms before it goes to the last three lines which will make it spin backwards with another delay of 500ms.

While the code worked as expected an observation on the motor revolutions per minute (RPM) provided by the simulation showed a fixed speed. In the context of the project it would be more desirable to have the ability to change the speed of the wheels for example if the sensors detect an object the robot could slow down therefore PWM would be used to achieve this by changing the duty cycle of the signal.

# Week 2 – Wheels simulation

Work resumed on Tinkercad and to be able to set different motor speed via the duty cycle we decided to create a user defined data structure in C++, referred to as class called *MotorDirection* containing functions to execute the code required for the desired wheel behaviour. Pin connections between the Arduino and the H-bridge chip were moved where necessary to PWM compatible pins, the components added to the current project are shown in figure 11 where digital oscilloscopes were added to show the PWM signals.

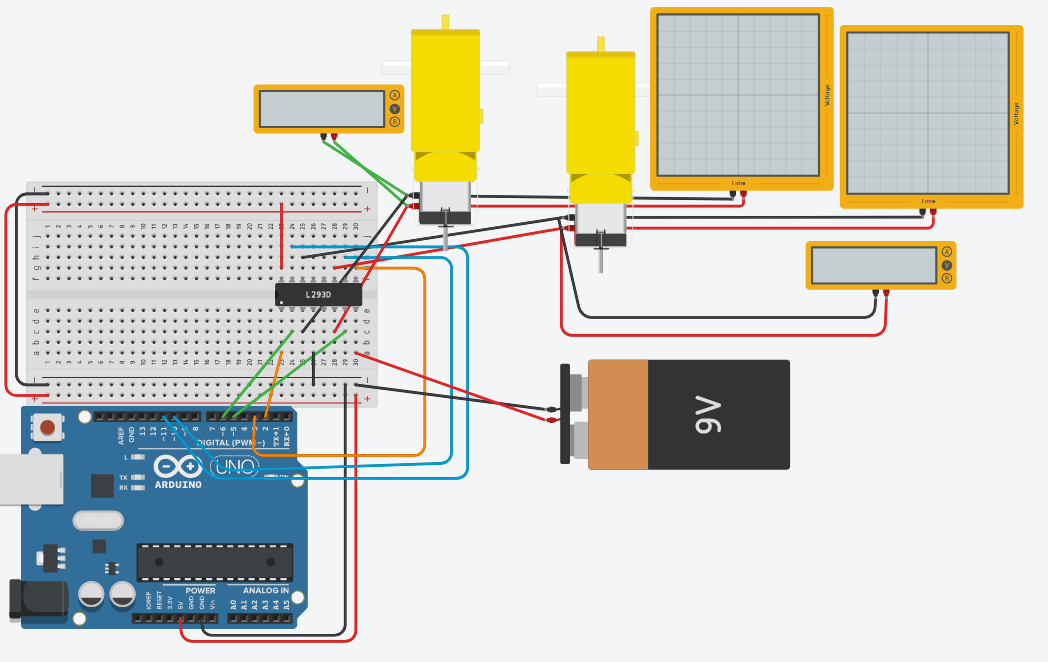


Figure 11 Simulation of two wheels

The class that was created is shown in figure 12 with all the attributes and behaviours that will be required.

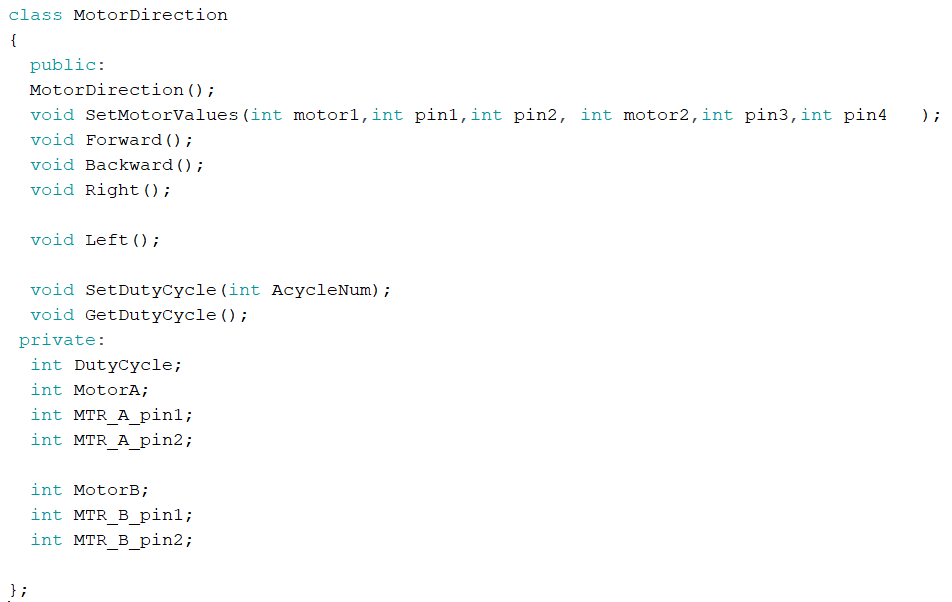
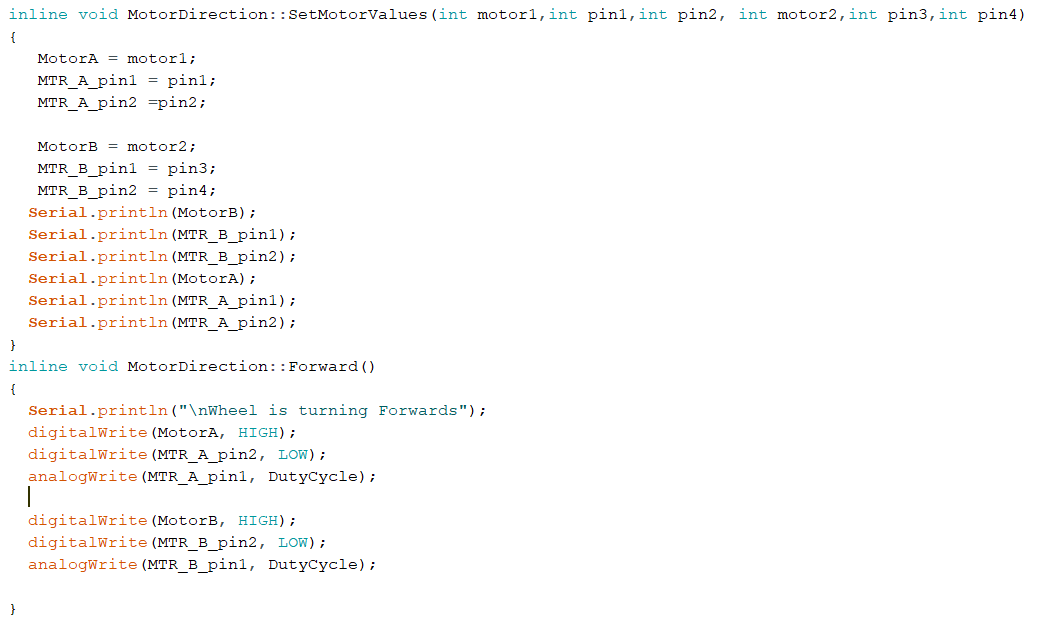


Figure 12 Class for Motor Direction

Below in figure 13 are the corresponding functions for the newly created class.





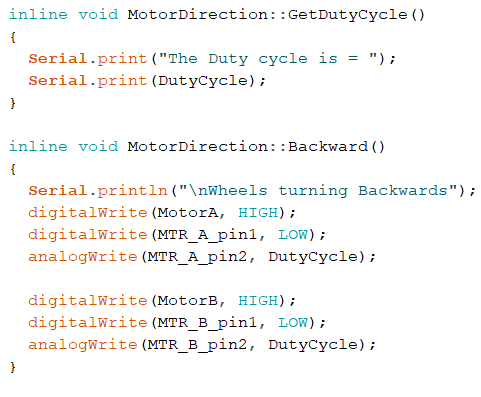


Figure 13 Function behaviours for Class Motor Direction

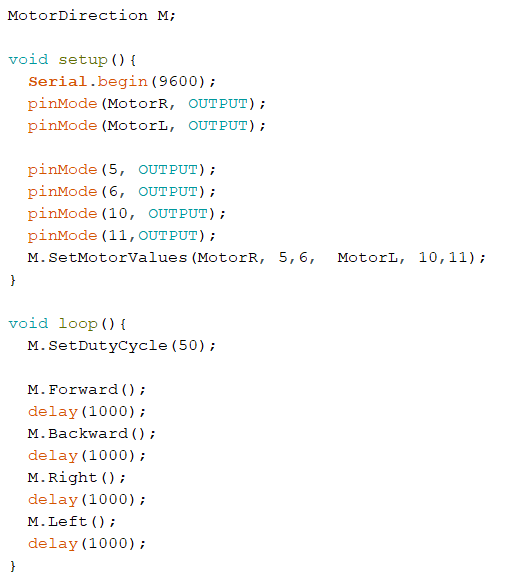


Figure 14 Testing both wheels using class object

In figure 14, to use the class there needs to be an object e.g. *MotorDirection M* then pin values for the motor are set using *M.SetMotorValues* and functions can be called that turn the pins high or low for the corresponding movements. With this code the wheels rotated using PWM and their speed can easily be altered by changing the duty cycle value. The running simulation works as expected and rotates the wheels in the order seen in the loop.

Once the simulation results showed satisfactory results the design was validated by hardware testing, shown in figure 15.

A close up of a device

Description automatically generated

Figure 15 Hardware testing of both wheels

# Week 3 –Ultrasonic sensors simulation

After discussion with the supervisor, four ultrasonic modules of HC-SR04 was decided as the means by which the robot would use to navigate around its environment and for obstacle avoidance.

Week 3 was spent creating another class called *DistanceSensor,* this was done so that multiple objects of the same type could be used in the final program this meant that the student could create functions for calculating the distance inside the class one and simply call this function for any number of objects created.

The four sensors were added to our project in Tinkercad as seen below in Figure 16.

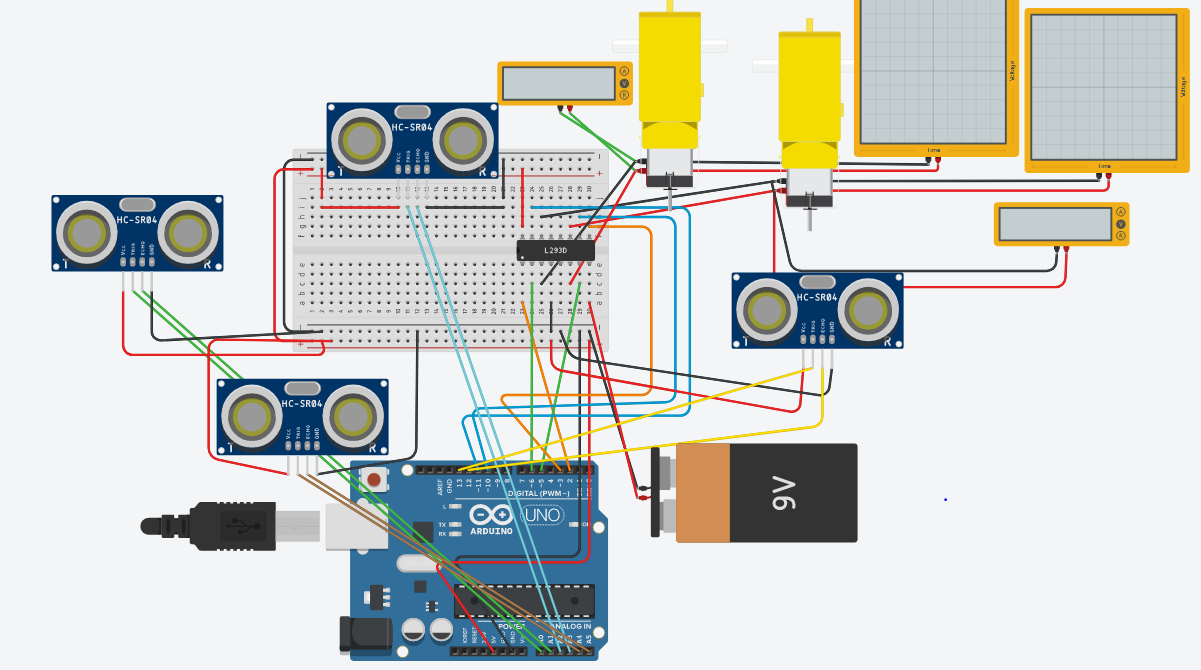


Figure 16 Simulation for 4x ultrasonic sensors

All the attributes and function of the newly created class is shown in figure 17.

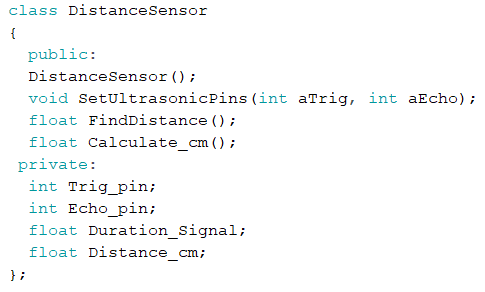


Figure 17 Class for Distance Sensor

Its behaviours are shown below in figure 18.



Figure 18 Function behaviours for Class Distance Sensor

As we can see above in the *FindDistance* function will send out the echo from the trigger pin and the echo pin will detect the returning pulse, inside we also call the *calculate\_cm* function so that the distance in centimetres is calculated and returned to main.

Due to the nature of this project the sensors would only detect objects which are only centimetres away so the *calculate\_cm* converts the signal into units of cm. This is done by knowing the medium in which the wave travels is air and the speed of sound in this is 343 m/s this is needed to convert into units of cm & microseconds:

Then the result is divided by two because only the bounce back time of the signal from the object is desired.

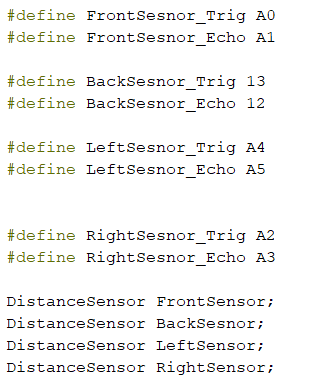


Figure 19 defining pins and creating sensor objects

In Figure 19 the pin values for the sensors can be defined at the top of the program making it easy to change the pin values if needed instead of manually changing values throughout the code. The setup loop (Figure 20) is used to set the pin modes and call the *setUltrasonicPins* function for each sensor.

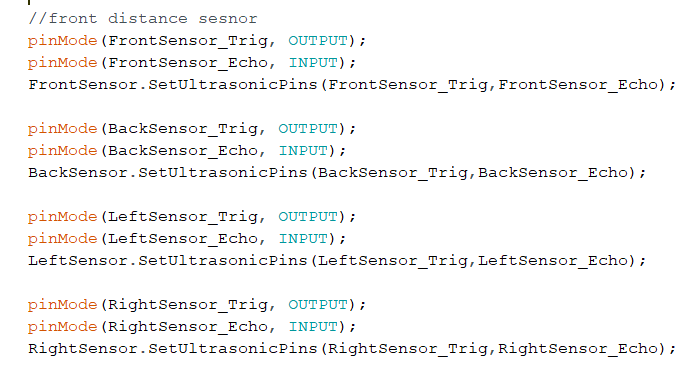


Figure 20 Setting pin modes and using sensor class

To check if the class was working, a simple test to verifying if the functions execute and return the correct distance measurements is done in figure 21.

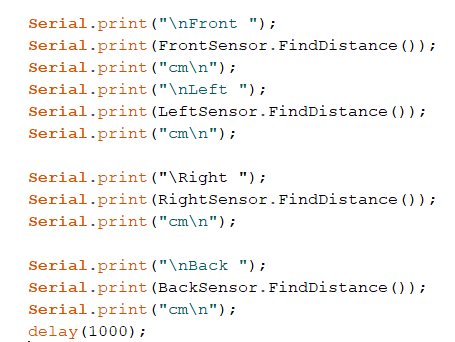


Figure 21 Getting distance measurements on multiple HC-SR04

The serial monitor showed correct measurement values of the virtual objects that were placed in front of each sensor. In Figure 23 the front sensor has a virtual object placed 35.9cm away and the output monitor (Figure 22) shows this object to be 35.19cm, a very close result.

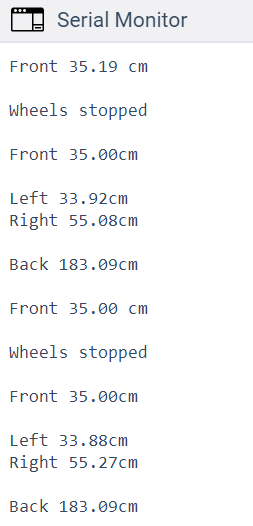


Figure 22 Distance results on multiple HC-SR04

It would be benifical to know how much error we can expect from the returned measurement compared to the actual distance so a quick calculation to check this:

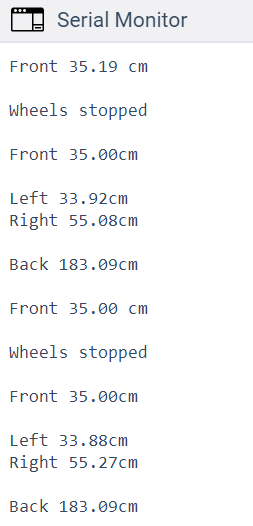
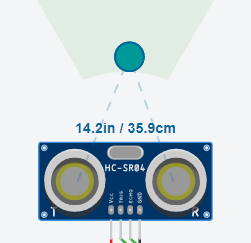


Figure 23 real distance value(Left) and returned value(Right)

The distance result generates an error of 1.97% which is more than accurate for this project. This code can now be combined with the wheels code allowing for the distance measurements to dictate what movements the robot should perform.

# Week 4 -Soldering onto proto shield

After discussion with the supervisor on the topic of adding wireless communication it was realised that there would not be enough pins left over for a wireless module so the four sensors would need to be reduced to three, eliminating the back sensor.

Work commenced on de-soldering an old amicus18 companion shield because it already had a module for the l293d to be placed on and its pins were already in line with the Arduino uno pins thus could be placed onto it like a shield.

A circuit board

Description automatically generated

Figure 24 de-soldered amicus18 companion shield

During the de-soldering process pin connections became damaged and it appeared better to find a replacement board rather than continuing with the current one. After consultation with lab technicians about the board, one was helpful in providing a new proto shield type rev3 to use for the project.

Later that week we mapped out the connections for the new board and started soldering. In figure 25 is the new board with the ultrasonic sensors mounted and placed onto the Arduino.

A circuit board

Description automatically generated

Figure 25 soldered components on new proto shield

When testing the sensors, we found that the front sensor was the only one not providing any distance measurements on the serial monitor window. To figure out what could be the issue we first checked the code and ensured that when defining the pin names their correct pin values were used.

Next, a multimeter was used to check the voltage across each of the pins and their corresponding paths back to the Arduino. The issue was found to be that the *Vin* pin for the sensor which was not receiving the full power that was output by the pin from the Arduino, to fix this we added more solder across the connections for that path.

# Week 5 – Hardware Testing

This week was spent creating code for the robot to move in a straight path parallel to a wall. The code created would work by maintaining a 15cm distance to the left & right and adjust itself when necessary to correct itself while it moves forward and when the front sensor detected an object 20cm away it will stop.

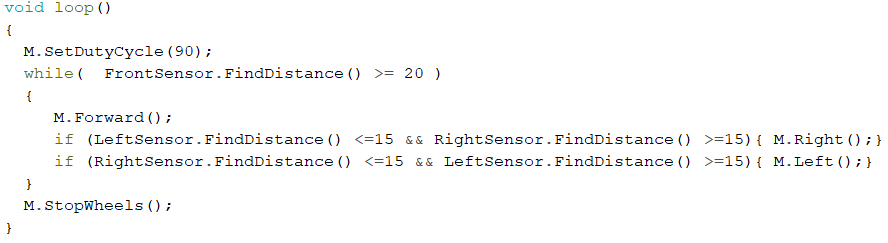


Figure 26 Code for parallel movement against a wall

A picture containing indoor, wall, floor

Description automatically generatedA picture containing indoor, wall, floor, sitting

Description automatically generated

Figure 27 Top and side view (Left & Middle) of setup and Testing (Right)

The movenent of the robot during the course corretions can be described as eratic because it would abrubtly stop and begin turing one wheel left/right, the frequency of this quick change is amplified due to the robots tendency to deviate from a straight path.

The robot was sucessful in maintaing a parallel path with the help of course corrections but sudden course corrections would cause the sensor array to fall off, then testing without walls at either side showed the robot would start off in a straight line but the more distance was travelled a deviation was noticed likely due to a slight imbalance of power that is sent to the wheels or different gear ratios used by the gearboxes or a difference in friction encountered by the wheels.

# Week 6 – WiFi module Testing

To incorporate wifi communication for the remote tramsmission of grid coordinates to the arduino, the superviser suggested switching to an ESP8266 Node MCU board which has WiFi cababilites. By switching the we would eliminate complexity of connecting two arduinos (second one would be for wifi) but the only drawback was that this new board had eleven pins for general purpose inlut/output so one of the ultrasonic sesnors would need to be removed from the design.

A decision was made to create a web server on the ESP where a user could enter in x and y coordinates this could then be saved on the board, from which movement commands could be sent to the l293d chip. Since we did not have experience coding in the HTML language, we were able to find an online source which had already created the necessary code to enable a web server on the ESP at *circuits4you.com.*

The layout and text shown for the webpage was modified as needed for the project, this saved into a string to be sent to the server while it is shown as multiple lines in figure 28 the entire string was required to be placed on a single line.

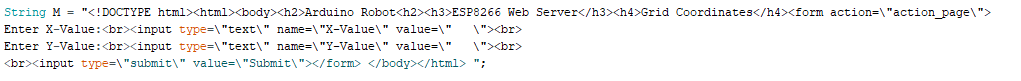


Figure 28 modified HTML webserver code

The original HTML code was replaced with figure 28 code and two attributes were defined at the top of the program (Xval & Yval) to store the received user value and be used in the rest of the program.

A close up of text on a white background

Description automatically generatedA close up of text on a white background

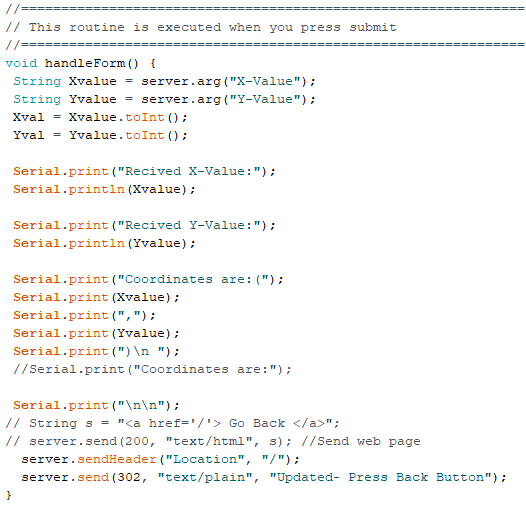
Description automatically generated

Figure 29 ESP webserver(Left) and Serial monitor result (Right)

Figure 29 shows the output to the serial monitor where the first three lines are the ESP webserver connection and setup, this process produces an Internet Protocol (IP) address which can be used to connect into the ESP webserver. In the webpage an X-value of 27 and Y-value of 543 have been inputted and this result is shown in the serial monitor as intended.

This is the webserver code.





Continued to next page.

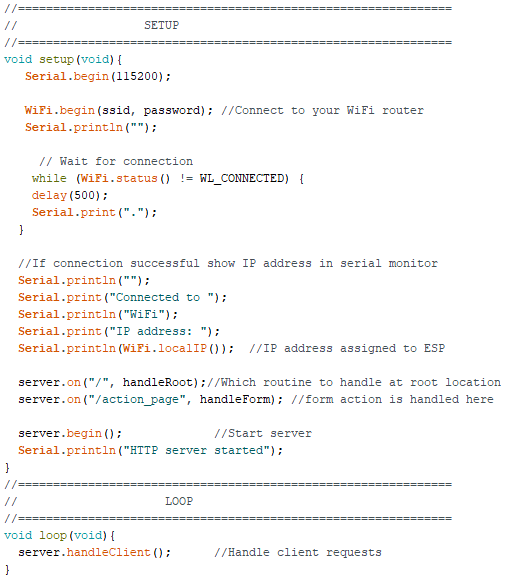


Figure 30 ESP webserver code [9]

In figure 30 most of the webserver code is contained in functions such as *handleRoot* and *handleForm* that are called during the *server.on* operations*.*

# Week 7 – Soldering onto copper strip board

To place the two sensors, ESP8266 and connection for l293d chip first wire connections were placed onto to the copper stripboard shown in figure 31 then sub-sections were created by cutting tracks in order to isolate some connections and prevent unwanted cross connections as seen in figure 32.

A circuit board

Description automatically generated

Figure 31 Top view of final copper stripboard

The wire connections between each component makes use of the horizontal copper pathways and only uses wires for the vertical connections to the components.

Section 3

Section 2

Section 1

A picture containing electronics, circuit, clock

Description automatically generated

Figure 32 Bottom view of final copper stripboard

In Figure 32 other than carvings to separate parallel pins for the ESP and l293d we can see that that stripboard has been split into three main sections with the cravings. Section 1 is the connections for the ESP board that connect into the l293d. Section 2 is the L293d connections while the last section is used for connecting two sensors to the ESP.

# Week 8 – Code improvements

From previous testing of the wheels code we noted that the robot had a natural tendency to deviate from a straight path, due to this it would perform adjustments which require it to stop and make left/right turn. In order to make these adjustments look like smooth transitions the motor direction class will be improved upon by adding in a function that will set different duty cycles to both wheels.

In order to avoid naming confusion with the current *left* and *right* functions we will called the new ones *RightR* and *LeftL* where an integer value will be passed to the function, Figure 33.

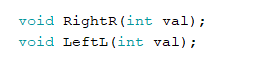


Figure 33 Added functions in MotorDirection class

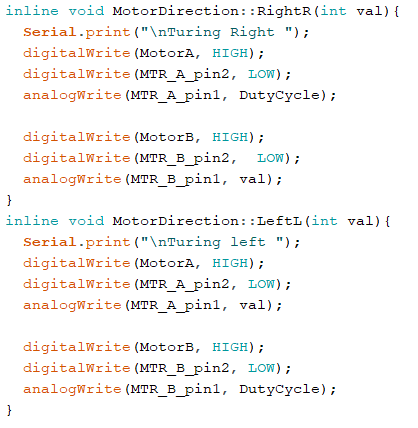


Figure 34 behaviour for new functions

The new functions shown in figure 34 will use the value passed into them as the duty cycle value for the second wheel while the first wheel uses the duty cycle set at the beginning of the program.

The *setDutycycle* was modified so that instead of having to compute and enter in a value between 0-256 mapped from 0%-100% PWM signal ratio, we can instead enter in the signal ratio as a percentage and use a calculation to convert into the 8-bit range of 256. This is more convenient because if we want for example 50% of the power to be at the wheels we can enter this in, and it will be calculated into 128.

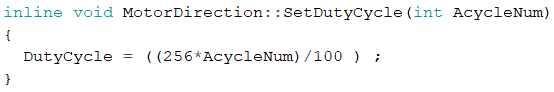


Figure 35 modified setDutycycle

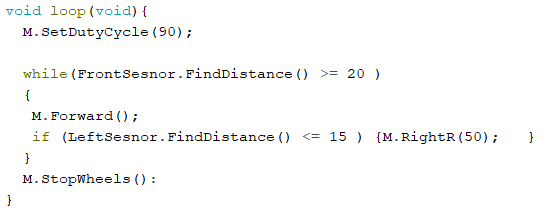


Figure 36 Testing the new functions

The new code in figure 36 was used for testing where the duty cycle is set to 90% and the behaviour of the robot had much improved with two duty cycles, one for each wheel causing the movement to be smoother and more consistent.

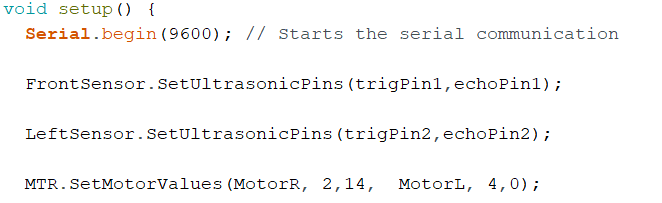
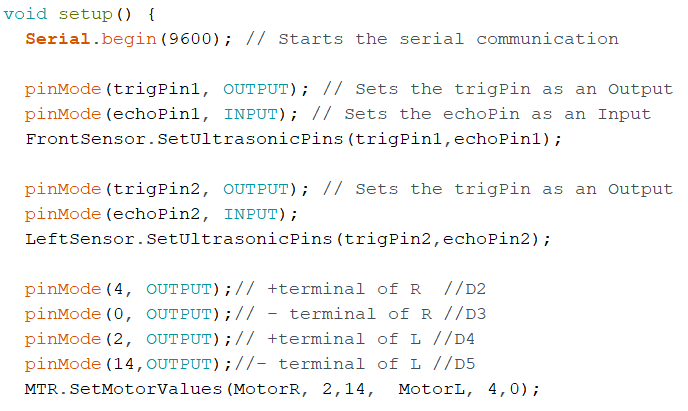


Figure 37 The setup Before(left) shifting and After(Right) shifting

Other improvements made by the student was shifting the *pinMode* commands into their respective set functions for the sensor and wheel class, this resulted in a decluttered setup shown in figure 36.

# Week 9 – Testing the Arduino robot with Wi-Fi

The code segments created so far were now integrated as a single program, the new setup is shown in figure 38 where the two sensors and the wheels and the Wi-Fi connection is now done.

**A circuit board

Description automatically generated**

Figure 38 Top view of finished robot

All the components were put onto the copper stripboard for final testing, shown in figure 38. This is a cleaner setup than during the wheel testing from figure 27.

In Figure 28 we have placed the battery pack for the ESP8266 under the copper stripboard and have attached the connections for the two wheels and the external power supply for the motors to the three screw treminals with an elastic band around the stripbaord to mainatain it in a fixed position.

**A picture containing yellow, truck, parked, street

Description automatically generated**

Figure 39 Side view of finished robot

After the program was uploaded the robot was unresponsive to the transmitted coordinates, this prompted a double check of the program and hardware (checking if connections are loose/unconnected) but the issue was found to be the ESP which was damaged possibly due to sensors sending 5v signals back to the ESP pins which only allow max of 3.3v input.

However, this should not have happened as we are using the 3.3v outputs of the ESP to power them and the 9v battery is only connected to the l293d.

After this setback all the final code adjustments were made on the Adriano uno set up from Figure 27. Since it won’t be possible to transmit coordinates to the uno because it has no wireless capabilities, we will use the serial monitor to input coordinates during testing.

Understanding of how to get serial monitor input was found at [***www.c-sharpcorner.com***](http://www.c-sharpcorner.com). This was used for collecting the character input from the user before we convert them to integer values for use in the main program.

Code for getting serial monitor input is shown in Figure 40.

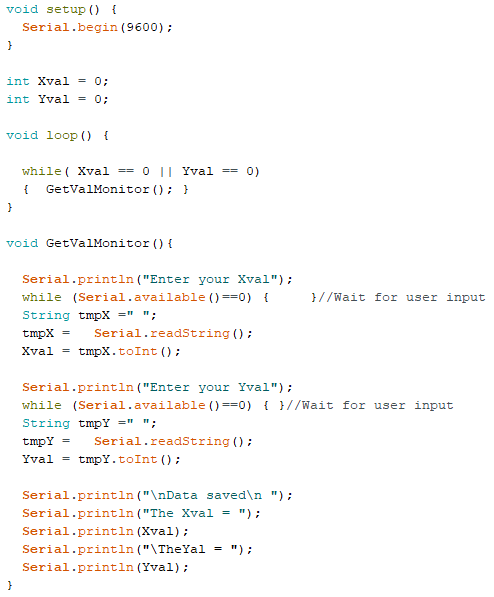


Figure 40 Serial Monitor input test [10]

# Proposed modifications

If given the opportunity to improve the project we could have a total of six ultrasonic sensors, one front and back one top left & right and one bottom left & right. The two sensors at each side would allow for the ability to accurately position the robots body parallel to shelfs by ensuring both sensors are equal distance from the wall. An Arduino mega would be suitable for the increased number of sensors and as for the code both classes the student created can be reused by creating multiple objects of the same type. The wheel array could be improved by replacing them with stepper motors which can rotate the wheels at accurate angles, allowing for precise positioning of the robot along a shelf.

As for the Wi-Fi communication the webserver that has been set up is working as intended but some feedback about the package could be sent back to the website to better inform the user of package details and possibly the robots status on fetching the package.

# Alternative solutions

Alternative solutions could entail local GPS navigation within a confined area and use four stepper motor wheels for accurate positioning and a robotic arm for picking up the package with cameras for object recognition/detection. The circuit for this setup could be put onto a custom printed circuit board.

# Conclusion

As the project progressed, we had to remove two sensors due to time and hardware constraints but managed to complete all the important aspects of the project and got a demonstration of the proposed solution working as intended. There was always a steady progress during the project period during which we kept a good record of all the progress made during the project and used research skills to find the right components to used. We used software concepts such as classes to create user defined data types. Where hardware and software issues were encountered our code debugging, and electronic debugging skills were used.

# References

1. Y. Dahl, "What is an H-Bridge? - Build Electronic Circuits", *Build Electronic Circuits*, 2018. [Online]. Available: https://www.build-electronic-circuits.com/h-bridge/. [Accessed: 03- Feb- 2020].
2. "L293D Motor Driver IC with Arduino | H-bridge IC", Electronics Hobbyists. [Online]. Available: https://electronicshobbyists.com/l293d-motor-driver-ic-arduino-h-bridge-ic/. [Accessed: 01- Feb- 2020].
3. "L293D Motor Driver IC introduction, pinouts and how to use", Microcontrollers Lab. [Online]. Available: https://microcontrollerslab.com/l293d-motor-driver-ic-introduction-pinouts-example/. [Accessed: 02- Feb- 2020].
4. T. Hirzel, "Arduino - PWM", Arduino.cc. [Online]. Available: https://www.arduino.cc/en/tutorial/PWM. [Accessed: 07- Feb- 2020].
5. "Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino | Random Nerd Tutorials", Random Nerd Tutorials. [Online]. Available: https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/. [Accessed: 14- Feb- 2020].
6. S. Nasir, "Ultrasonic Sensor Arduino Interfacing - The Engineering Projects", *The Engineering Projects*, 2017. [Online]. Available: https://www.theengineeringprojects.com/2017/08/ultrasonic-sensor-arduino-interfacing.html/. [Accessed: 21- Feb- 2020].
7. "Arduino Uno Tutorial [Pinout] - DIYI0T", *DIYI0T*. [Online]. Available: https://diyi0t.com/arduino-uno-tutorial/. [Accessed: 10- Feb- 2020].
8. "ESP8266 NodeMcu Pinout - ESP8266 Shop", *ESP8266 Shop*. [Online]. Available: https://esp8266-shop.com/esp8266-guide/esp8266-nodemcu-pinout/. [Accessed: 03- Mar- 2020].
9. M. Thakur, "ESP8266 Node MCU Handling HTML web forms data | Circuits4you.com", *Circuits4you.com*, 2019. [Online]. Available: https://circuits4you.com/2019/03/20/esp8266-receive-post-get-request-data-from-website/. [Accessed: 07- Mar- 2020].

[10].D. Dash, "Reading Input From Serial Monitor In Arduino", *C-sharpcorner.com*, 2016. [Online]. ///////Available: https://www.c-sharpcorner.com/article/reading-input-from-serial-monitor-in-|||||||arduino/. [Accessed: 25- Mar- 2020].