



Project *Report*

M00668318

Hamza Bari



Contents

Introduction	2
A description of the system concept	2
Sensors & Actuators	2
Robotic Schematic Diagram	2
Robotic System Description	4
Requirements	4
Key decisions taken and why	5
Implementation	15
Sensor Fusion.....	15
Hardware Building Process	16
Datasheets of the Hardware Components Used.....	25
Arduino Uno	25
DHT11 Temperature & Humidity Sensor	26
4WD Arduino Uno Extension board.....	27
Servo motor	28
Ultrasonic Sensor.....	29
WS2812B_LED.....	30
Geared Motors	31
Software Building Process.....	32
Temperature Sensor Arduino Code	32
Robotic Arduino Main Code.....	33
Testing	46
Further Improvements	47

Introduction

In this written report I will cover the following areas: a description of my system concept - showing the requirements, the sensors and actuators used, showing a schematic diagram of the system, key decisions I took and why – the changes I made to my plan, the reasons why I chose to use those sensors, a record of implementation – image showing the hardware process and the software code with annotations, screenshots of the sensor used datasheet, and the test procedures by testing the robot by the requirements to see what works and what doesn't and how can the robot be improved further if it doesn't meet the requirement and what other things could of also been added for further improvements for the second version of the system as this is only the first version.

A description of the system concept

The main purpose of this project was to find out the different temperature in the rooms so people, in their houses know what the temperature is in different areas and, it can be solving issues as well for example, if one area is hot all the time, then that issue can be investigated because the robot keeps navigating out of that area constantly. Other sensors are used for precautions as there are a lot of sensor 360 degrees around the room as, a result the ultrasonic sensor has been used with the servo motor rotating it 180 degrees as, this way it should avoid hitting objects while navigating out of the area.

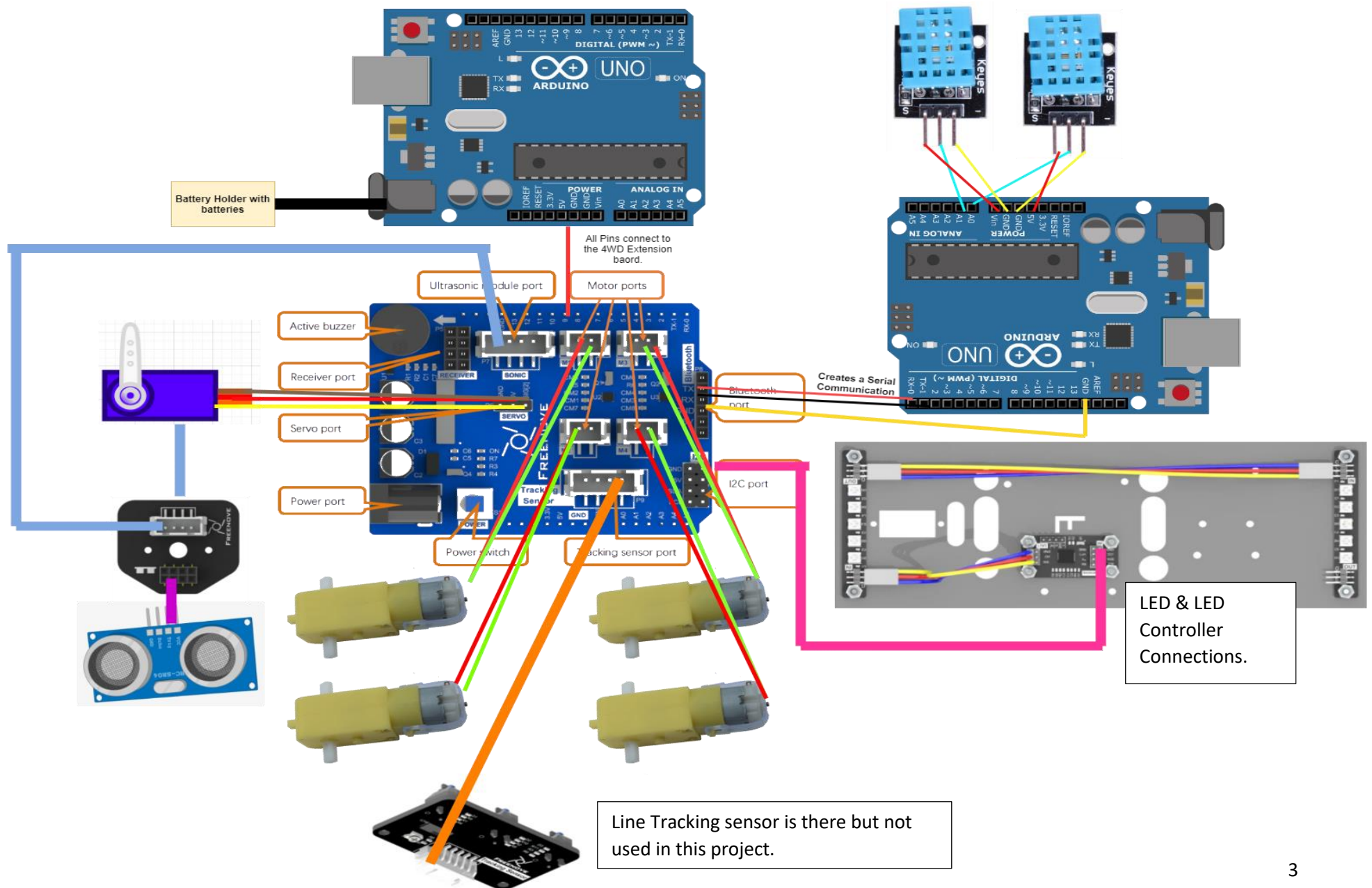
Sensors & Actuators

The sensors and actuators used in this robotic system for this project only are the following:

- 2x Arduino Uno
- 4x gear motors
- 4x wheels
- 2x DHT11 Temperature & Humidity sensor
- 1x 4WD Arduino Uno Extension board
- 1x Servo motor
- 1x Ultrasonic module connector
- 1x Ultrasonic sensor
- 1x WS2812B_LED_controller
- 2x WS2812B_LED
- Double-sided and Single-Sided Jump wire pins for sensor and actuators connections
- 2x 18650 3.7V batteries
- Batter holder with power jack connection to the Arduino Uno.

Robotic Schematic Diagram

You can witness the robotic schematic diagram in the page below which shows how the sensors and actuators are connected together and how they communicate with each other. In simple words, it shows how the sensors are connected to the Arduino. This diagram was very useful when it came to implementation because, the connection of the robotic system was easily done without any hassle.



Robotic System Description

All the sensors and actuators are connected together which can be witnessed in the symmetric diagram above. The concept of this system is to find out the temperature in a room using the DHT11 temperature sensors. The robot will use the gears and wheels to navigate around the room, as, the robot is navigating the two temperature sensors will collect the temperatures. Those two temperatures are very likely to have a different value therefore, using the software algorithms the robot will make a decision whether its too hot in this area or too cold, and depending on the statement and decision making the robot will use its wheels to navigate out of that area, the path to move area in the room is set in the if statement algorithm.

The robot has led lights therefore, when the temperature is too hot then, it will flash the red led light on the robot while navigating as well and, if the temperature is cold then it will flash the green led light on the robot while navigating.

The robot will use its wheels to navigate out of the area but, the room or area will have many objects and obstacles as, a result we have added the ultrasonic sensor to avoid hitting obstacles objects. The ultrasonic sensor is connected to the ultrasonic module and, the ultrasonic module is connected to the servo motor, this way the servo motors move the ultrasonic module and sensor constantly for 180 degrees to check for any objects around therefore, if an object is there then it can be avoided. There are algorithms written for obstacles avoidance and those methods are getting executed when the temperature decision has been made. You'll understand this more when we come to the implementation section.

In a very likely case, if both temperatures are equal from the temperature sensors, then the blue light will flash as well as, the robot will move around in a circle in the room.

The software programming algorithms were written using the Arduino code which basically is C++.

In this project more time was spent on configuring the hardware making sure its connected properly, and hardware was more sophisticated in this project, compared to the time and challenge on the software. Overall, this is a project which is integrated with hardware and software because there, hardware sensors which have been used and there are software algorithms used for controlling them which makes it an integrated system.

Requirements

Below, is the table which lists all the requirements that is expected from this robotic system to perform.

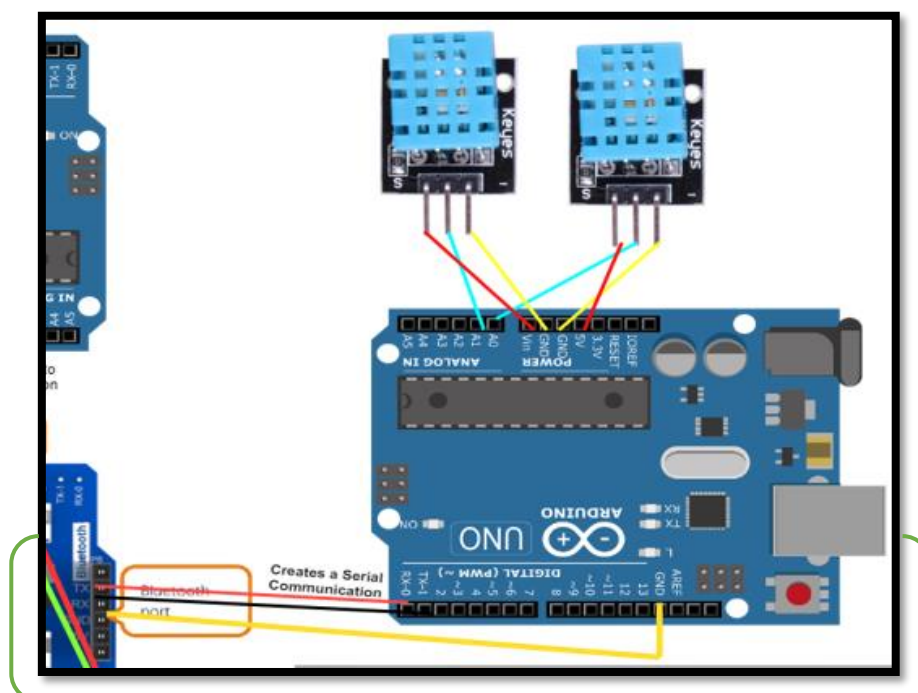
1.)	If the first temperature sensor is higher than the second temperature sensor, then flash the red led lights.
2.)	When the red led lights are flashing, navigate out of that room/area using the wheels.
3.)	If the second temperature sensor is lower than the first temperature sensor, then flash the green led lights.
4.)	When the green led lights are flashing, navigate out of that room/area using the wheels.
5.)	When navigating the ultrasonic module should move 180 degrees using the servo motor.
6.)	If the object is near then the ultrasonic sensor should capture that and stop the robot from hitting the object.

7.)	If first and second temperature both match together, then the robot should make a circle in the room flashing the blue led light.
8.)	At the start and end of the navigation path there should be a sound of the buzzer to indicate that the path has completed navigating and its moving to the next path.
9.)	Serial communication between both Arduino's to send the temperature value from the Arduino connected with temperature sensor to the main robot system Arduino.
10.)	Everything should happen but, with short pauses and delays as, this way the robot should be in more control.

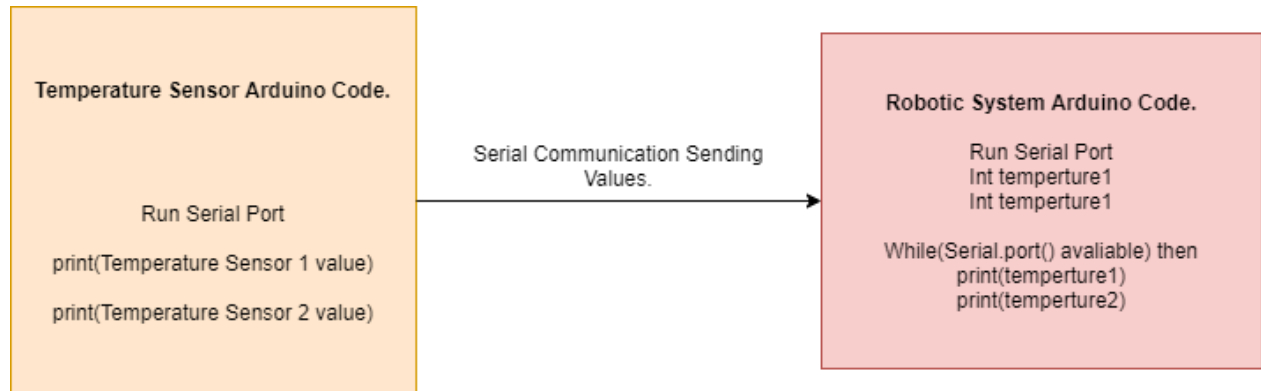
Key decisions taken and why

The key decision I took was to use another Arduino to connect both of my temperature sensors since there was no space on the robot Arduino and the reason for that was because, the extension board was connected to the while Arduino and, the extension board only had pins limited sensors as, a result I made a key decision here to use another Arduino.

Therefore, I connected both of my temperature sensor using the other Arduino and programmed the code for that, which managed to get the temperatures from the sensors but, the challenge here was how can I transfer this second Arduino temperature data to the main Arduino on the robot. Therefore, another key decision I made was 2 use Bluetooth module to transfer the data as, there was a pin available for the Bluetooth module component. The Bluetooth idea couldn't work as well since it was causing errors due to the different versions therefore, it didn't manage to work. After that, I decided that I will connect a database which would send and retrieve data but, that idea didn't work as well because for that to work the Arduino needs to be connected to the PC machine since, databases run on PC machines. Therefore, I finally made a key decision to use a serial communication between the two Arduino that managed to work, the diagram can be witnessed below which makes it easier to understand.



A part of symmetric diagram is shown above, you can witness the green label where its shown how the serial communication is performed. RX connected to TX, TX connected to RX, and GND connected to GND, once all of this is connected that, then makes a serial communication between the two Arduinos. Therefore, you can see how the serial communication works in the small software architecture pseudo code diagram below.



This way I could transfer the temperatures values from one Arduino to the other Arduino and this, is the reason why I finally decided to use this method of doing it.

The reason why I chose to use the DHT11 temperature sensor was because it's an excellent quality sensor, and it gives you a faster response compared to other sensors which measure temperature and, it uses lower power of +5V and since, I was using a lot of other sensors so it was essential that I use sensors which have lower power anyway so it doesn't affect the power which can cause the system to fail and, it has a signal transmission distance up to 20 meters which is a good distance for measuring the temperature in a room.

This is the following way of how both temperature is calculated from DHT11 Temperature sensors:

First a request is sent to both DHT11 sensors. After that both, sensor will be sensing a responsive pulse. After that, the sensor will send 40 bits data to the Arduino. For starting communication the pulse needs to be sent to the DHT 11 sensor, and also pull down the data pin minimum 18ms and then pull up.

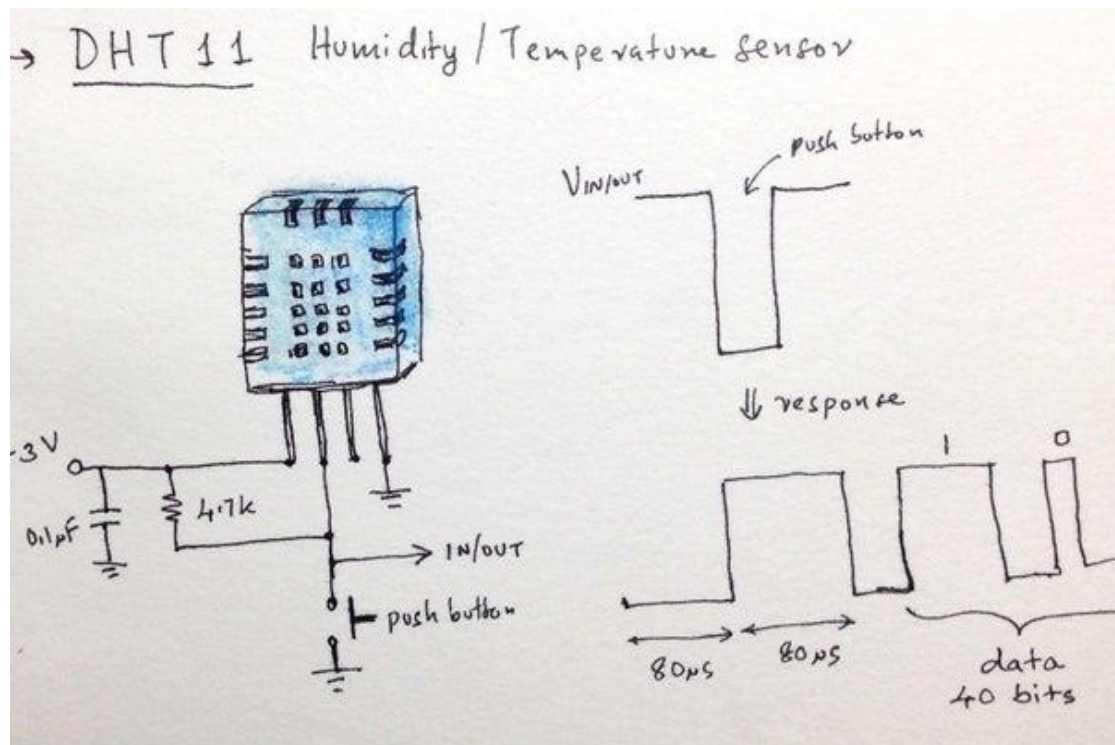
Once, the pulse is received from Arduino then, the DHT11 sensor will send the response pulse to the Arduino which means it will show that the temperature sensors have received the starting pulse. The pulse responded with will be low at 54us but it will go higher to about 80us.

Then, after sending the response pulse from the Arduino to the DHT11, then the DHT11 will send the data which is the temperature and humidity values obviously for this project we only require temperature but, in general it will send both with its checksum. The data from in total is 40 bits which contains 5 bytes, and each byte is 8-bit long. In the 5 segments the first two contain the humidity integer value. But we need the temperature there the 1st 8-bits which are the integer part and the next 8 bits are segment contains the temperature value in an integer value. The temperature is given to you in the Celsius from and this is another reason why I decided to use the DHT11 sensor.

The final segment is known as the checksum which is holding the check sum of the four segments. The checksum byte is known as the direct addition of the temperature value. Thus way we can verify it in the Arduino to check is the value is the same or not the same. If its not the same then, there

must have been an error, else the temperature value is correct. This is another reason why I chose this was due to its accuracy and it has a method to check accuracy which was needed to make sure the temperatures were accurate.

After the Arduino receives the data then the DHT11 sensor pins will go into the lower power consumption mode till the Arduino sends the start pulse rate. The Arduino will usually send pulse rate after a delay of few seconds as, I made a decision to add delays in this part of code and the reason for that is because it keeps the DHT11 sensors are more in control and it makes it less likely for a mistake compared to, if everything was at a faster process with no breaks.



Since, there was no Bluetooth module to print data onto the serial monitor without Arduino-USB connection since there is no space left on the Arduino pins therefore, I decided to use the WS2812B_LED_Controller and WS2812B_LED Strip. Therefore, if the temperature was hot then it will flash red LED, or if the temperature is cold, then it will flash green LED, or if they are both equal, then it will flash. This way it will indicate if the temperature is hot or cold, or just equal. This is the reason why I decided to use WS2812B_LED Strip. Another reason why it was decided to use a LED strip because, it uses lower power. In the images below you can witness the LEDs flashing.

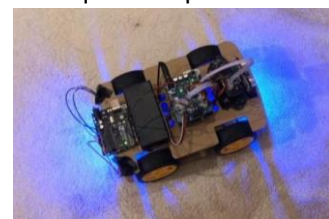
Hot Temperature

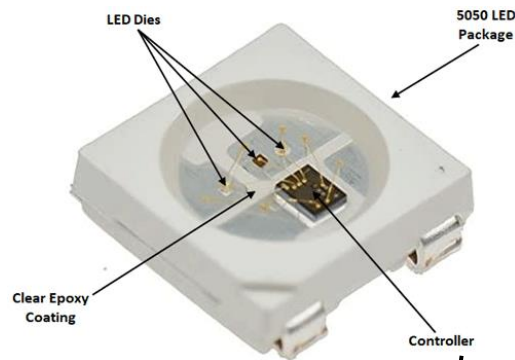


Cold Temperature

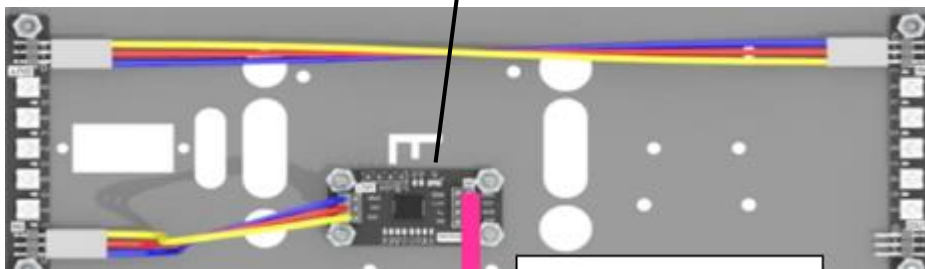


Equal Temperature





The WS2812B has 3 LEDs which are RGB and it has a controller in the package. The controller has a 24-bit register which is taking the data from the pin and stored, then display it onto the LED. Therefore, in our project when the temperature is hot then, the controller takes the data from the pin and, then displays it onto the LED which is the colour red.



G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

The 24-bit register has been divided into three parts, the first part is 8 bits long which is holding the different level of brightness for each colours. As, there are already 8 bits therefore, there are 256 brightness values for each LEDs, as there will be three colours and, this way a total number are 17 million can be made. This is another reason why this was used in this project as, it has a lot of variety and LEDs usually have less power compared to other light as a, result it wouldn't affect the power too much.

As, you can witness two LED strips are used. Each Strip uses a few LEDs therefore, the output of the LED controller will be buffered for maintaining the signal and the quality brightness of the LED lights when, there are too many LEDs used. This is the reason why it was beneficial for our project.

There is also no way that we could identify when the navigation paths finishes but, there was a solution found to that where I took a key decision of using the Arduino sound buzzer to play sound each time the navigation path completes. This way the user can easily identify when the navigation path has completed.



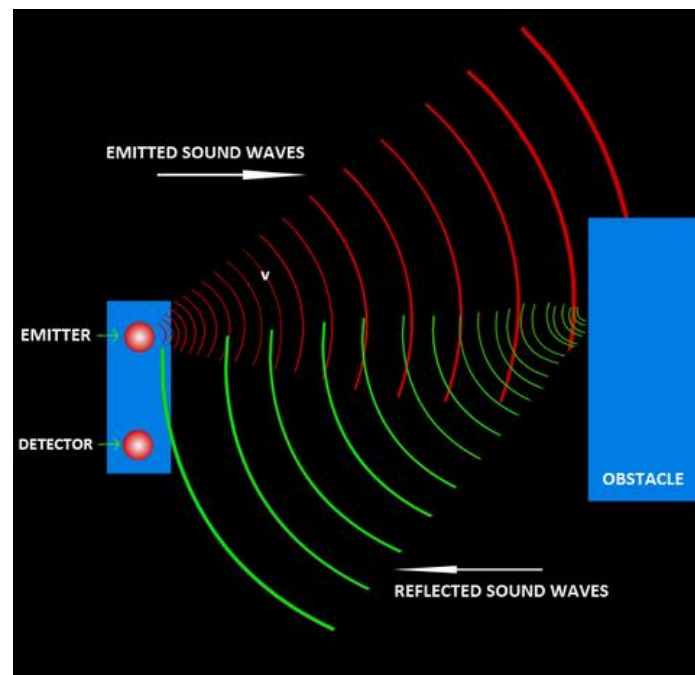
Arduino Buzzer on the 4WD Extension Board.

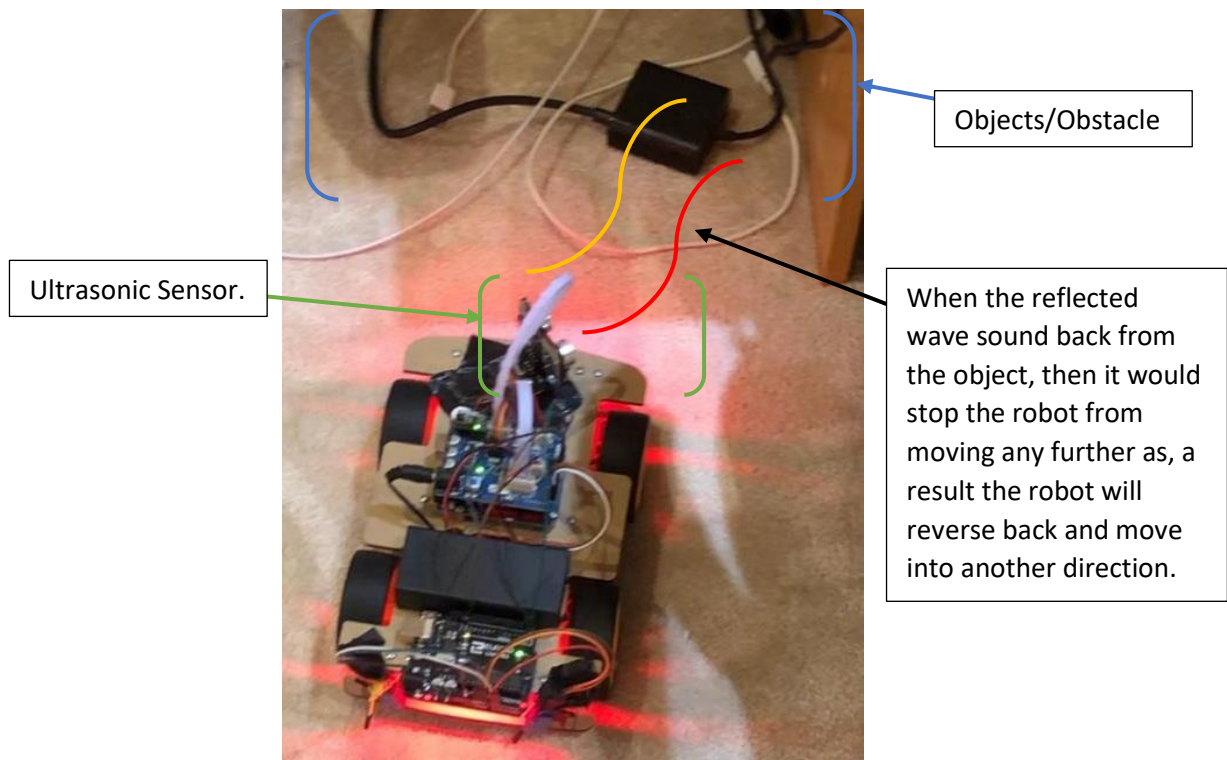
A buzzer is known as an audio component in the Arduino. There are active and passive types of buzzers. In this project the active buzzers have been used. Active buzzers use oscillator inside. Active

buzzers will work till the power is being supplied to the robotic system. Active buzzers are straightforward to use not that complexed that's why its been used as for a small task soothing easy is good. Active buzzers make specific frequency sound therefore, in the project when the buzzer sounds it will make just specific frequencies. It requires very low battery as well only 4.5V for the minimum. This way easy because the buzzer is fixed in the 4WD extension board therefore, when the board is powered using the batter holder power jack, then the buzzer is also powered nothing external required that's another reason why this was considered.

The reason why it was decided to use the ultrasonic sensor is because when, the robot is navigating it could hit obstacles on its way as, a result the ultrasonic sensor will detect the object which is in front of the sensor and, then it will reverse back from where the object is located.

Ultrasonic sensor has two main components which are called the following: receiver: which is used to encounter the sound once it has travelled to and then from the target, and it also has a transmitter that can emit sound by using the piezoelectric crystals. To calculate the distance between the Ultrasonic sensor and the object in the room, the sensor will measure the time it has taken between the emission of the sound by the transmitter to its contact with the receiver. The formula to calculate this is $\text{distance} = \frac{1}{2} \text{Times} * \text{Speed of the sound}$ and, it returns the result in meters/second. As, example would be if an ultrasonic sensor were aimed at the object, then, and it took 0.025 second for the sound to bounce back, the distance between the ultrasonic sensor would be e.g., $D = 0.5 \times 0.025 \times 343$ which would result overall in 4.2875 meters.





The yellow and red are the emitting sound waves. The yellow wave is the emitter wave, and the red wave is the reflected sound wave.

To calculate the distance between the Ultrasonic sensor and the object in the room, the sensor will measure the time it has taken between the emission of the sound by the transmitter to its contact with the receiver. The formula to calculate this is $\text{distance} = 1/2 \text{ Times} * \text{Speed of the sound}$ and, it returns the result in meters/second. The example for this image is calculated below:

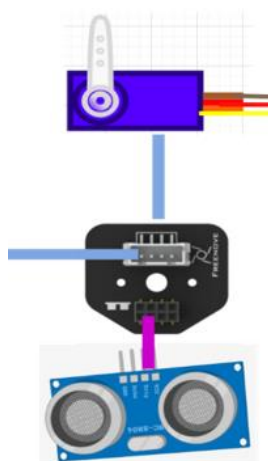
Speed of sound = 340 m/s

Time = 0.025

Distance = $0.5 \times 0.025 \times 340$

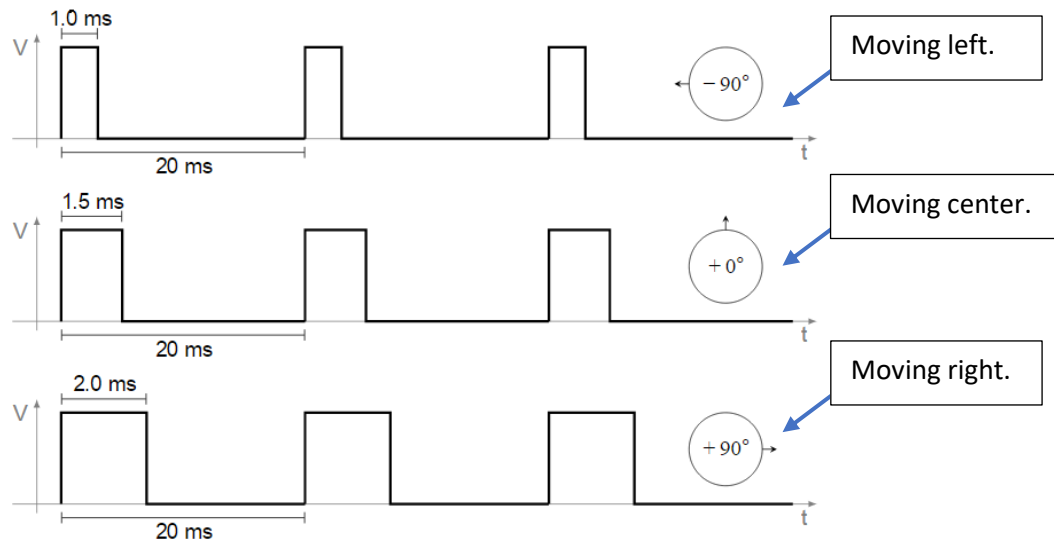
$4.25 = 0.5 \times 0.025 \times 340$

It had taken 0.025 seconds for the wave to be bounced back from the object. The distance which has calculated between the ultrasonic sensor and the box is **4.25 meters**.



The servo motor is connected to the ultrasonic module, and the ultrasonic sensor is connected to the ultrasonic module. The reason why the servo motor has been used because it will rotate the ultrasonic module by 180 degrees while the robot is navigating. The reason why this is important is because there is only one ultrasonic sensor used, and the objects are 360 degrees around the room as, a result its important that the ultrasonic sensor moves 180 degrees constantly therefore, it can bounce back waves from different angles and this way this servo motor will eventually stop the robot from hitting the object from at least the center, left, and right angle as, it will send and bounce back waves from the left and right angle as well.

The servo motor would turn 90 degrees in either right or left direction but, including both it makes a movement of 180 degrees. This is one of the reasons why I decided to use it for my project. The servo motor is controlled by sensing the PWM signals through the 3 wires which are connected to the 4WD extension board. The pulse will be sent every 20 milliseconds. The width of the pulse will determine the rotation of the servo motor. In this project when a pulse of 1ms is sent to the then, it will move -90 degrees to the left angle and if the pulse of 1.5ms is sent then it will return back to the center, and if the pulse of 2ms is sent then it will move +90 degrees to the right side. These pulses will be sent very quickly therefore, the ultrasonic sensor moves faster and recognises obstacles in its way. This makes very minimal chances for the robot to crash into the objects.



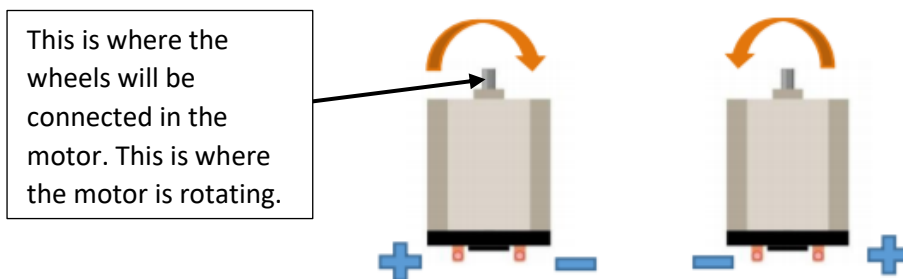
Below are the following advantages of the servo motor:

- High efficiency
- High output power relative to their size
- More constant torque at higher speed
- Closed-loop control
- Quiet operation
- Highly reliable
- High ratio of torque to inertia
- High acceleration
- High-speed performance
- Torque control
- Smooth running
- High accuracy
- Small in size
- Well suited to varying load applications

Servo motor has a controlled torque which smoothly rotates, torque control was one of the main reasons why I choose to use this servo motor because, the force needs to be smooth when rotating causing less damage to the system, and another advantage was more constant torque at a higher speed because, due to the ultrasonic sensor the servo needs to go at a higher speed therefore, its very good if the torque will be constant at a higher speed. It's very useful in robots where high accuracy, lower weight for the speed and power, and quite reliable in terms of its size. This is what is

required for the project and the servo motor will meet my expectations. All other advantages weight as well in the making of my decision to use the servo motor.

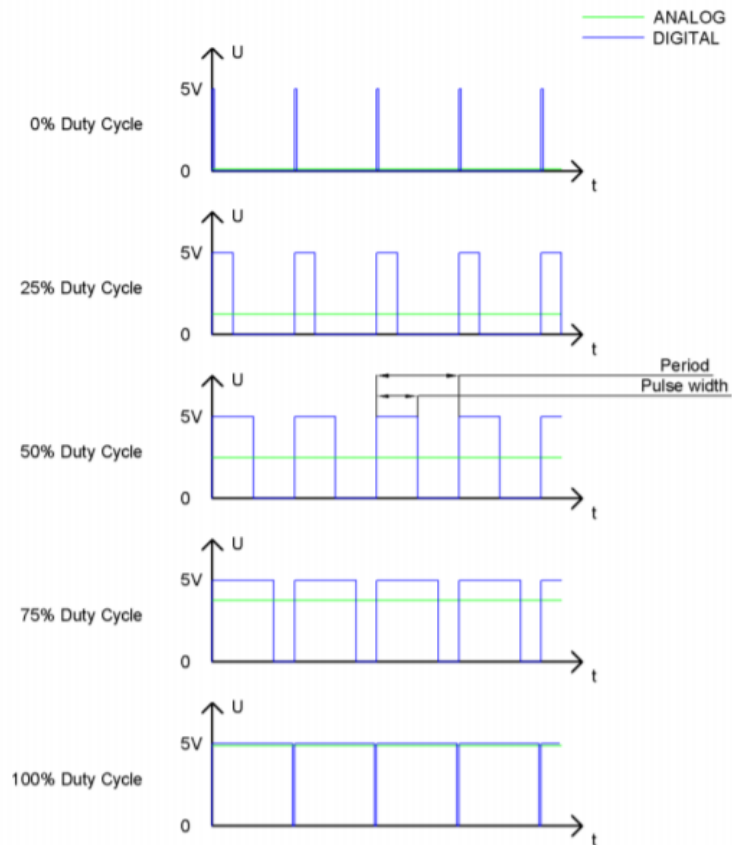
I decided to use the TT Geared Motors to control the wheels of the robot. The reason why I decided to use the TT Geared Motors is because the motor it provides more torque as, a result my robotic system us going to turn with a stronger force, and stronger force is required as the robot is going to move in a lot of different direction and taking a lot of left and right turns as well as movie around in a circle which will use a lot of rotation of the geared motors. This is especially useful because, the robot can move effectively with carrying load as, there is some load on the robot.



The speed of the motor will depend on the voltages between the two last ends of the geared motor. The higher the voltage the higher the speed will be of the geared motor.

The PWM is using digital pins for sending some frequencies of the waves that is the output of the higher and the lower levels. These waves will last for a while. The total time for setting the higher levels and lower levels will be fixed which is recognised as the period. The time for the high levels is called pulse width, and the duty cycle is the percentage for the ration of the pulse time, or the pulse width to the total period from waveform.

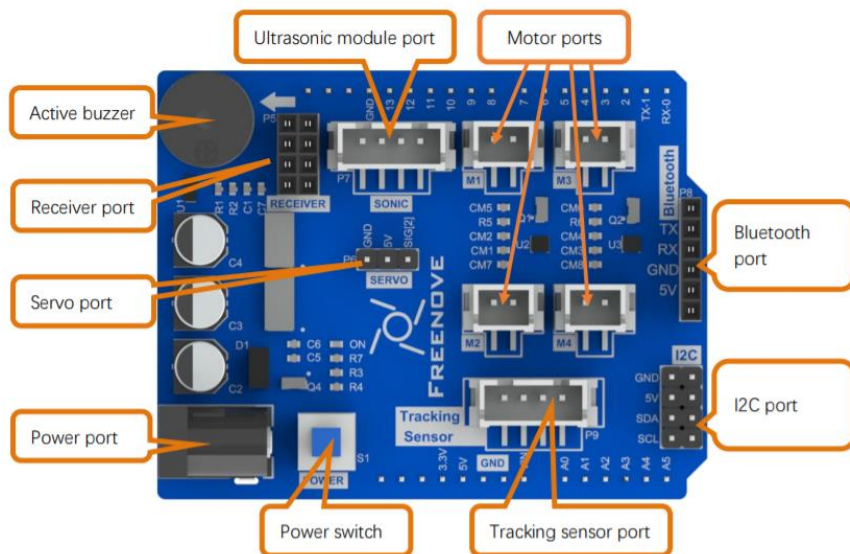
If the output lasts high for a long then then the large duty cycle, and the higher the corresponding voltage in the analogue signal. In the image below it can be witnessed how analogue signal voltage can vary between 0 volts to 5 volts with 5 being the highest, the analogue signal voltage corresponds to the pulse width which will either be 0% or 100%.



Longer the PWM duty cycle, then that will result in a higher power output which can be witnessed from the image above. That's the reason why we have used PWM to control the speed of our geared motors because, using this the speed of the wheels can be controlled. This way it will make the robotic navigation more controlled.

Relationship between extension board and the control board is as below.

Pins of Arduino	Ports of extension	Pin multiplexing	Description
0	UART-Bluetooth		Bluetooth
1			
2	Servo		
3	Direction-Right		Direction of right motor
4	Direction-Left		Direction of left motor
5	Motor-Right		Speed of right motor -PWM
6	Motor-Left		Speed of right motor -PWM
7	Trig		Ultrasonic module
8	Echo		
9	SPI-NRF24L01	IR-remote	CE/IR-remote
10			CS
11	SPI-NRF24L01		MOSI
12			MISO
13			SCK
A0	Battery voltage	Buzzer	The ADC uses an internal 5V reference and the acquisition voltage is the battery voltage *1/4. Output to the Buzzer through the comparator. When higher than the 4.5V, output HIGH.
A1	Line-tracking sensor		Left
A2			Middle
A3			Right
A4	I2C		SDA
A5			SCL



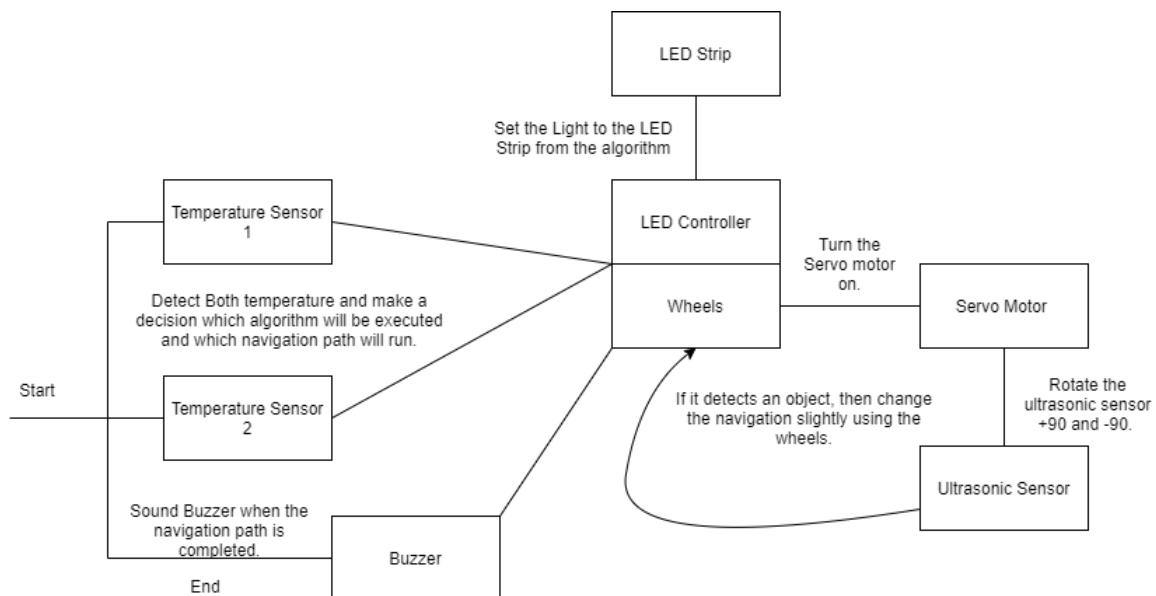
You can see in the table above where the Arduino pins correspond to the extension board ports, and you can witness which sensor wire will be inserted into which port. The reason why this has been used because it makes the system more concise and cleaner. This way it's also easier to understand as, you can easily tell which wires are connected to which sensor ports. usually without the extension board, therefore, would have been a lot of wires therefore, that could made things tougher to understand if there was a problem and it would of look very messy and jumbled up, this is the reason why we have decided to use the extension board so it's easier to understand, solving errors and mistakes. There is only one disadvantage, you can easily run out of spaces such as I couldn't add 2 DHT11 sensors in this Arduino as, I had to use another Arduino with the serial communication.

The processors I've used in this project are Arduino Uno's. Arduino has a larger community and it includes a lot of software library which makes it easier to write algorithms for controlling the robot. It's easier to use and more reliable, and it has its own software IDE which makes the programming environment easier to understand and it has all the necessary controls from before in terms of uploading the code to the Arduino by just clicking one button for example, and these are the things which are ease of use. Its cross-platform which means it can run on any different operating system. The operating voltage is only 5V which is not much compared to other microcontrollers. It has many things built on to it for example the buzzers and in addition to that it has so many pins. There are so many ways of making the system work for example the serial communication I've implemented between both Arduinos. These are the main reasons why I decided to use the Arduino Uno for this project.

Implementation

Sensor Fusion

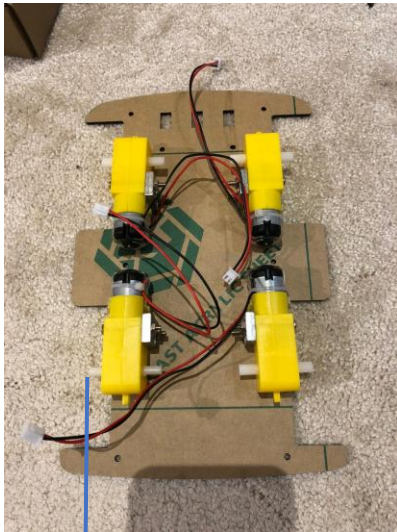
The Sensor Fusion will be used to combine the sensory data from the different sensors and actuators. There will be algorithms used in the program to perform this action.



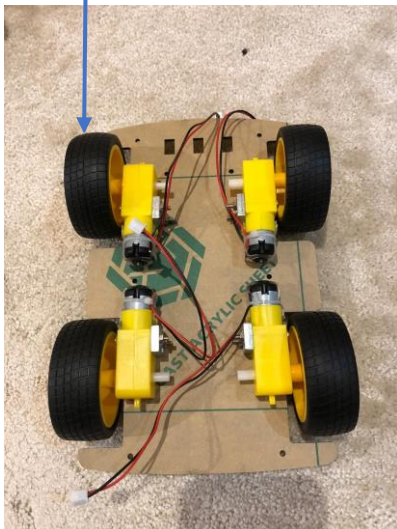
In the diagram above you can witness how the sensors will be combined together to perform this project robotic system task. Both temperature sensors are detected, then a decision is made if they are hot, cold, or equal. After that, based on the temperature the LED controller will set the colour and using the wheels the robot will navigate according to the navigate path set in the algorithm. When navigating the servo motor will be turned on which will rotate the ultrasonic sensor by +90 to -90 and the ultrasonic sensor will then detect if there is an object in front or not, and if there is then it will change the navigation path slightly where the wheels will move slightly different. After the navigation completes, then it will sound the buzzer to alert that this navigation path has completed and it will get the temperature data values and make a decision again to keep this process ongoing.

Hardware Building Process

Two Acrylic sheets have been used. Below, you can witness the development of Acrylic 1 sheet.



Checking all the gear motors and then using the screws to fix each gear motor on every side. These gear motors will be used for controlling the wheels of the robot and, the next stage will be adding the wheels. Both, Acrylic sheets have position and screw positions such as, where to fit which sensor, and which screw to use in order to make it tighter and fix it.

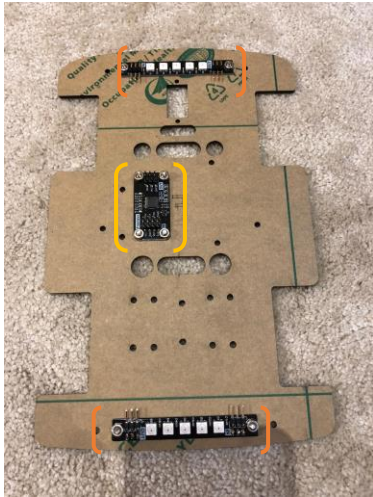


The blue arrow shows where the wheels are inserted into the gear motor. Wheels has a hole in the middle, the hole in the wheel is inserted into the motor gear which can be witnessed in the blue arrow. The wheels are installed to the system therefore, it can be used for navigation around and changing area in the room depending on the temperature. It will have more control when its turning in terms of the torque.

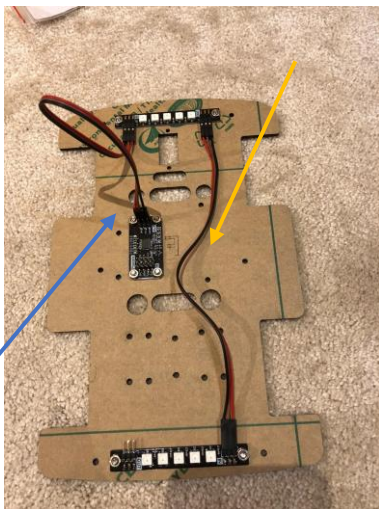


The orange label shows that the tracking module sensor has been inserted into the acrylic sheet location. After that, has been tightened by using the screws. The tracking module sensor is used for following a black line navigation path using PWM but, its only part of the robot sensor list but, it's not used in the project.

We have moved onto fixing sensor into the second acrylic sheet, same as the first one this has its holes and screw positions for fixing each sensors.



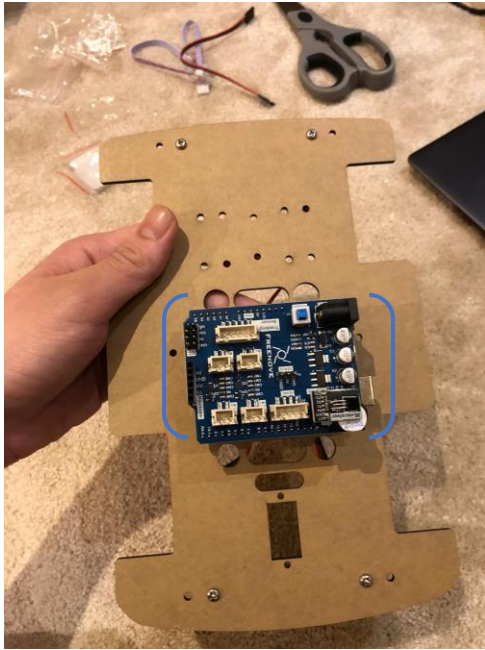
At the top and bottom of the acrylic sheet each LED strip has been installed and fixed which can be witnessed in the orange labels. The LED controlled which is used for controlling the RGB colours of those LED strips are fixed at its position in the middle using the screws which can be witnessed in the yellow labels.



To configure the LED controller and LED strips, they have been connected to each other using jump wires. The blue arrow shows, one jump wire connected from LED Strip to the LED controller and, the other wire has been connected from LED strip to the second LED strip. After these connections, the LED is installed to the system as a result, it can be used for displaying red if the temperature is too hot, green if the temperature is cold, or blue if the temperatures are equal.



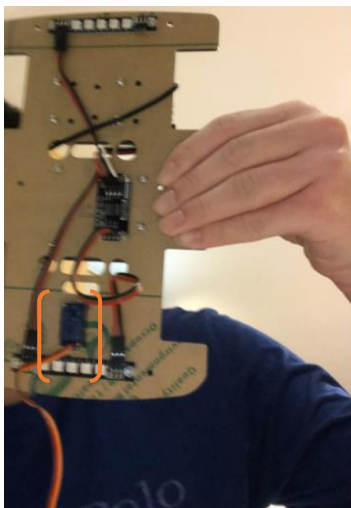
The second acrylic sheet has now been flipped over, and the first Arduino Uno for the robot has been inserted into its location which can be seen in the orange label.



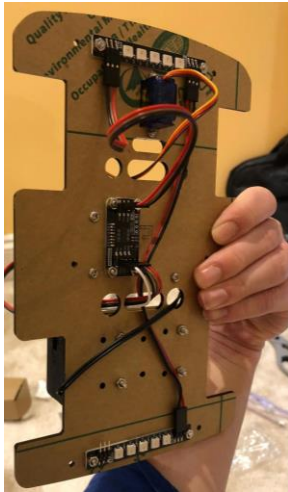
The 4WD extension board has been inserted on top of the Arduino Uno by using the Arduino Uno pins. It can be seen in the blue label. This is where all the jump wires will be connected of all the hardware sensors which have already been installed into the system.



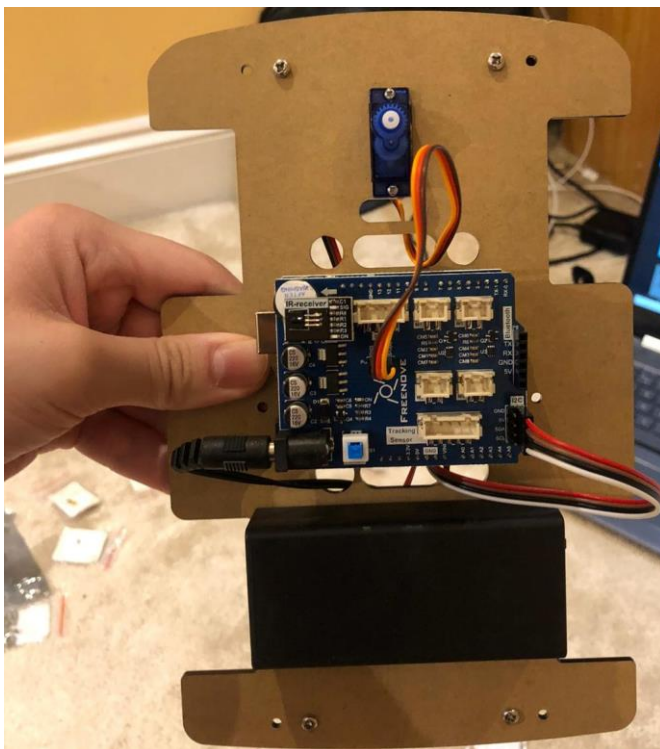
The batter holder has been fixed into its position. The battery holder power jack wire has been inserted into the 4WD extension board power jack. When the battery is inserted, then it will power the Arduino and 4WD extension board meaning, the programs will be able to run on the system port once uploaded but, still the sensor port connection are required.



In the orange label it can be seen that the servo motor is fixed into its location using the screws. The servo motor will be used for rotating the ultrasonic module 90 degrees to the left and 90 degrees to the right.



The servo motor has been installed into the system and the blue arrow shows that the jump wire is sent to the other side to be connected to the extension board servo port.



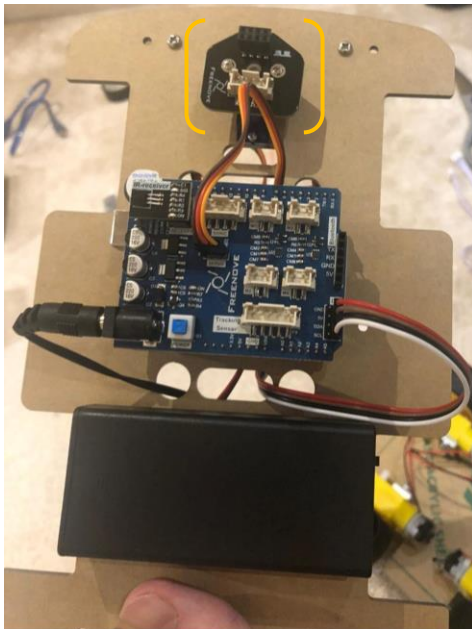
The servo motor jump wire has been connected to the servo port in the extension board therefore, the servo motor is now ready to run and also code has been used for testing to see if it works and it's been working okay.



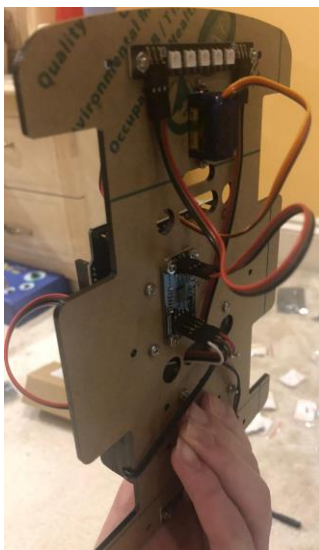
The power from the extension board has been turned after inserting the batteries into the battery holder. As, you can see after testing it, the power is on which means the Arduino and extension board is working.



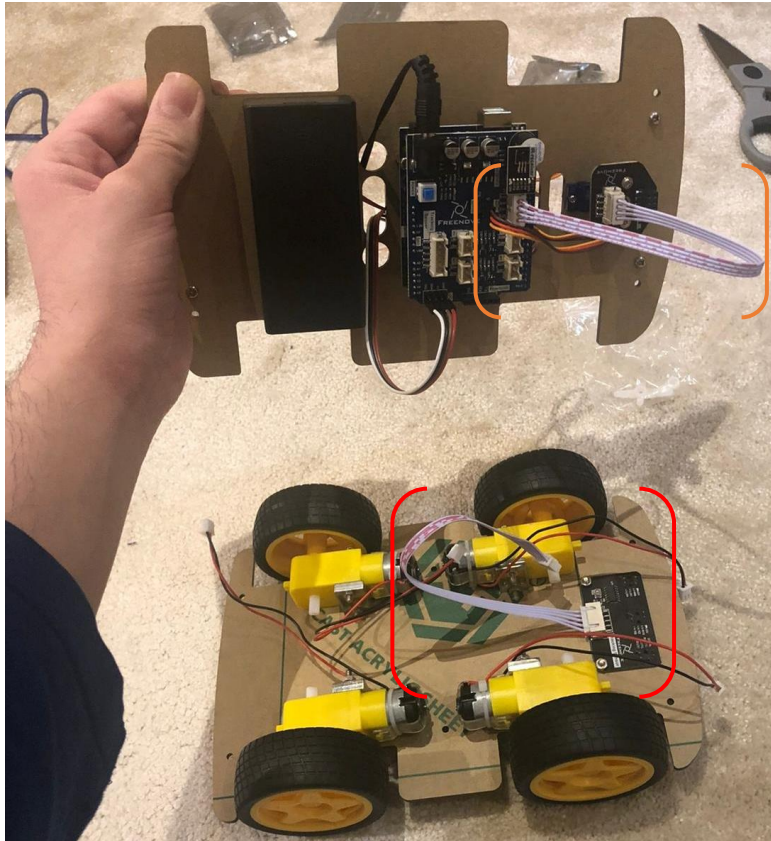
Once the is power on, it can also be witnessed that the LED controller is also on now which means the LED lights would turn on if the code is uploaded to the Arduino.



In the yellow label it can be seen that the ultrasonic module has been fixed on top of the servo motor. Therefore, it would rotate left and right.

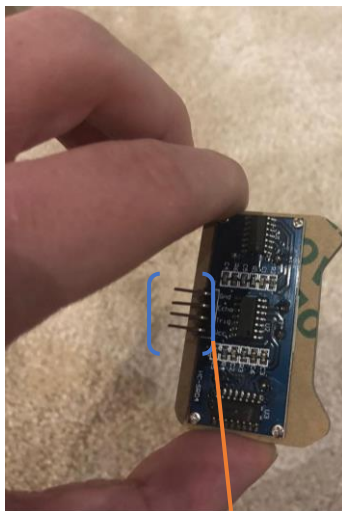


All wires were checked to see if they were connected to the right places and just making sure if every sensor is installed and configured in the right place or not before we fix both acrylic sheets together using screws.



The orange label shows the port wire is connected to the ultrasonic module to the ultrasonic port. This way the ultrasonic module is powered and its ready to be used if code is uploaded to the Arduino using its libraries.

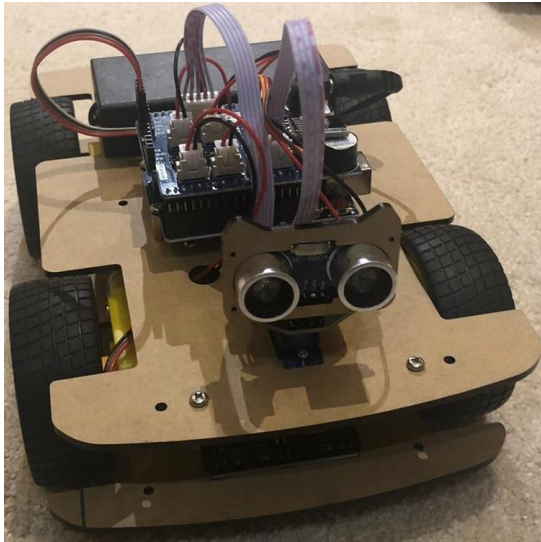
The first acrylic sheet in the red label shows the port wire is connected to the line tracking module and it will be connected to the extension board line tracking port.



The ultrasonic sensor has been added into the ultrasonic holder and where you see the holes in the green label is where its been fixed using the screws. The front and the back side are witnessed from the images such as how it looks.



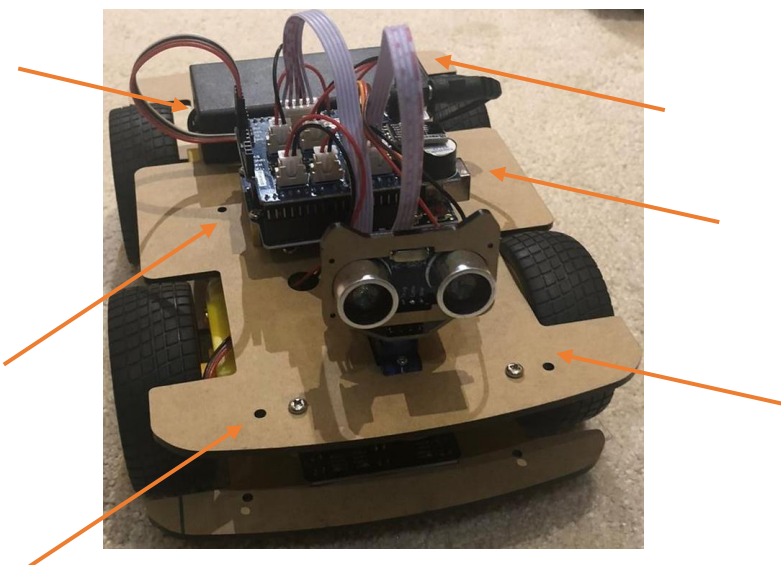
The ultrasonic sensor has been connected to the ultrasonic module using the ultrasonic sensor pins. Therefore, the ultrasonic sensor is powered and can be used once the code is uploaded to the Arduino. Now, when the servo moves the ultrasonic module left and right 90 degrees each, then ultimately it will move the ultrasonic sensor as well as, they are all connected together. When the ultrasonic sensor moves left and right while the robot I navigation to change the location in the room when the temperature is hot then, it should detect objects and obstacles in the way and avoid those and change path in duration of the navigation which stops the robot from being damaged.



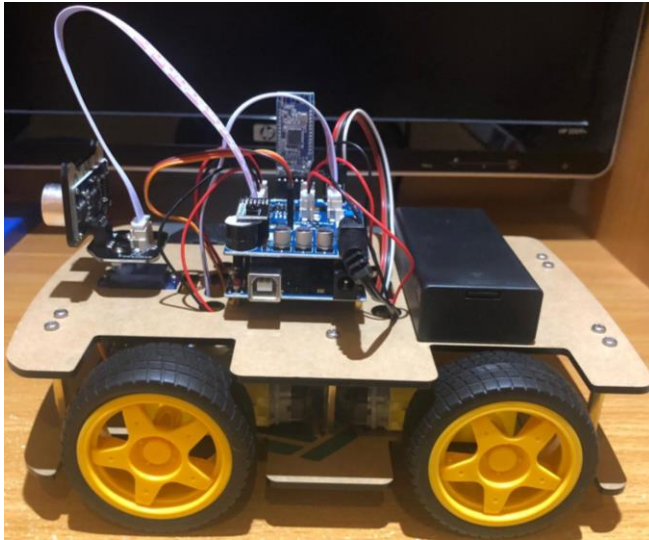
The second acrylic sheet would be put on top of the first acrylic sheet. Now, they will have to be fixed with tight screws therefore, both acrylic sheet is connected together which makes one robot together. After that, the robot can be used for navigation since everything will be fitted strongly.



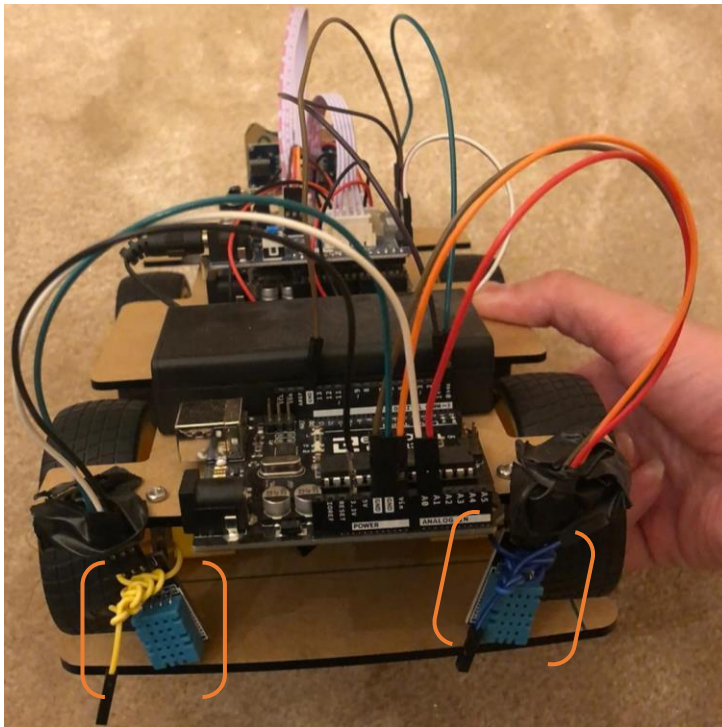
In the orange label you can witness that the nuts are used for titthing the edge of the robot. They are tightened using screws.



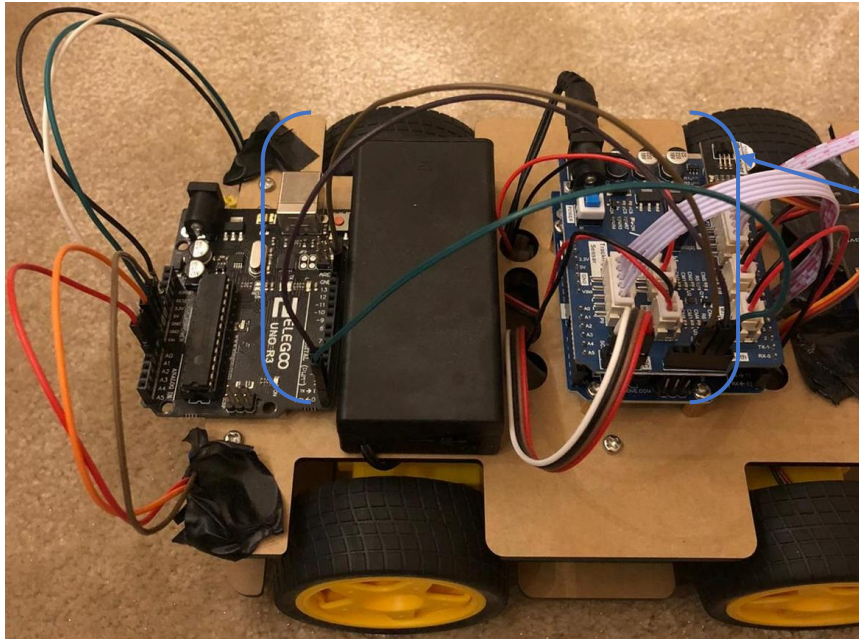
In the orange label you can witness six pointers and, all of these pointers show where the nuts will be used for tightening the edges. All nuts were added and tightened using screws in these orange arrow positions.



Once, all the nuts were fixed, then the robot was solid therefore, some code was uploaded for checking if the wheels were working and if everything was alright with this part of the implementation. After testing, we could witness that the robot was solid and navigating alright. Therefore, the first part of the implementation was completed. The second part for the temperature sensor was the next and the final bit of hardware implementation.



The second Arduino was stuck at the back of the robot using some glue. After that, in the orange label you witness two DHT11 sensors which are stuck on the back nuts of the robot using black tape and tied with some wires. After that, the jump wires have been used to connect both DHT11 temperature sensors to the Arduino pins which, you can see from this image on the left-hand where they are connected. After that, using the USB cable for the second Arduino the program to output temperature was uploaded onto the Arduino as, a result it returned back with the temperature values. This was the first main program of the system to get the temperature values and, it gets them successfully at this stage. The next challenge was how to transfer this temperature data to the main robot program where it uses algorithms to perform the main system task which is described in the **'A description of the system concept'** section.



The blue label shows the serial communication connection between both Arduinos.

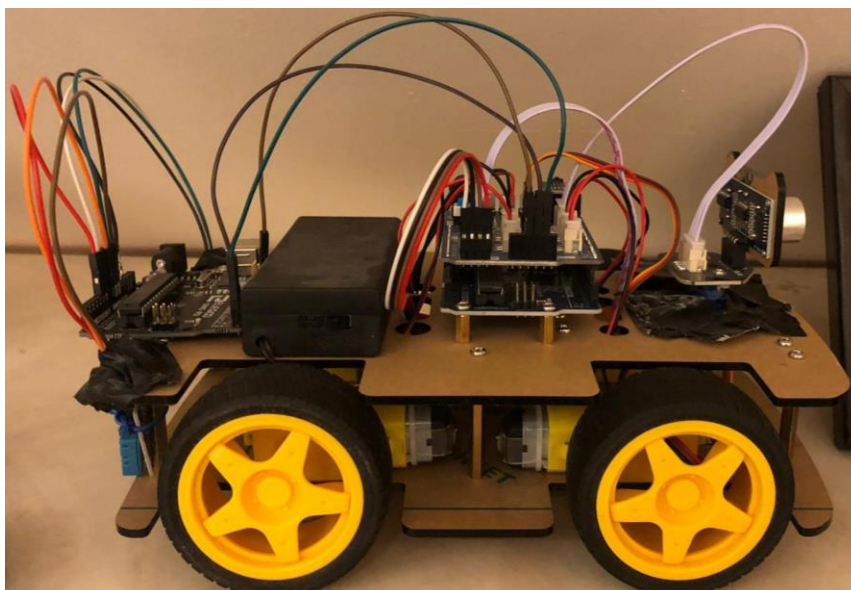
Arduino – Main Robotic Arduino

RX Pin – TX Pin

TX Pin – RX Pin

GND Pin – GND Pin

After that, the power is turned on and battery also loads the temperature sensor Arduino power as well through the serial communication connections between both Arduinos. After that, in the main task program serial port availability function is written for testing if the main program is receiving the temperature values from the second Arduino. After the test, we found out that the values are being received in the main program though the serial communication. Therefore, we used those values to develop robotic task algorithms in the main system program after that, it was able to perform its task the task which is described in the '**A description of the system concept**' section.

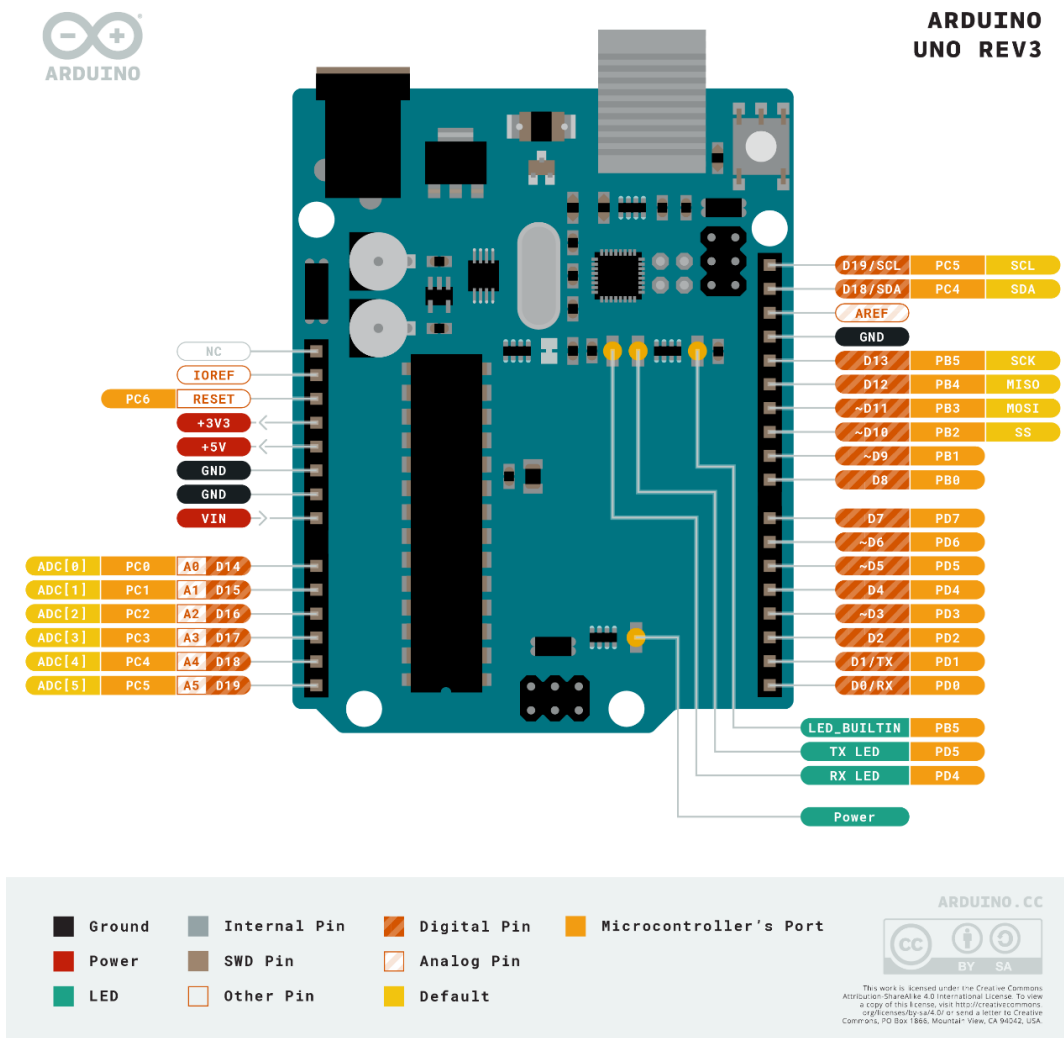


The image on the left shows how the final product looks like after all the robotic hardware is built, configured, implemented, and tested (testing meaning short codes were just uploaded for testing if each sensor is working or not but, all sensors worked). Now, the code can be run on the Arduino for the robot to perform the system task which is described in the '**A description of the system concept**' section.

Datasheets of the Hardware Components Used

These are all the datasheets for the hardware components used in the robot.

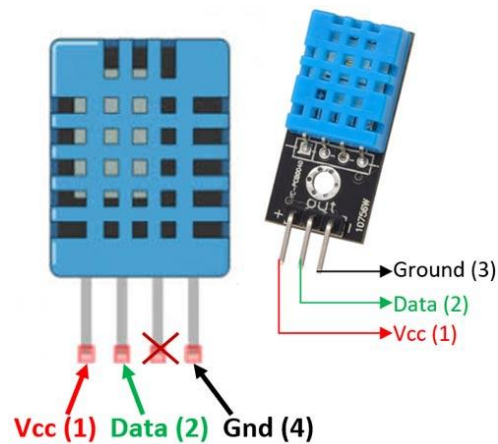
Arduino Uno



- **Serial Pins 0 (Rx) and 1 (Tx):** Rx and Tx pins are used to receive and transmit TTL serial data. They are connected with the corresponding ATmega328P USB to TTL serial chip.
- **External Interrupt Pins 2 and 3:** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM Pins 3, 5, 6, 9 and 11:** These pins provide an 8-bit PWM output by using `analogWrite()` function.
- **SPI Pins 10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK):** These pins are used for SPI communication.
- **In-built LED Pin 13:** This pin is connected with an built-in LED, when pin 13 is HIGH – LED is on and when pin 13 is LOW, its off.

The Arduino datasheet above supported in recognising which pins represents what. Therefore, using the schematic diagram and this datasheet it was easily figured out which pin the wire needed to be connected to and in addition to the recognising the pins for the power and ground.

DHT11 Temperature & Humidity Sensor



No:	Pin Name	Description
For DHT11 Sensor		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	NC	No Connection and hence not used
4	Ground	Connected to the ground of the circuit
For DHT11 Sensor module		
1	Vcc	Power supply 3.5V to 5.5V
2	Data	Outputs both Temperature and Humidity through serial Data
3	Ground	Connected to the ground of the circuit

DHT11 Specifications:

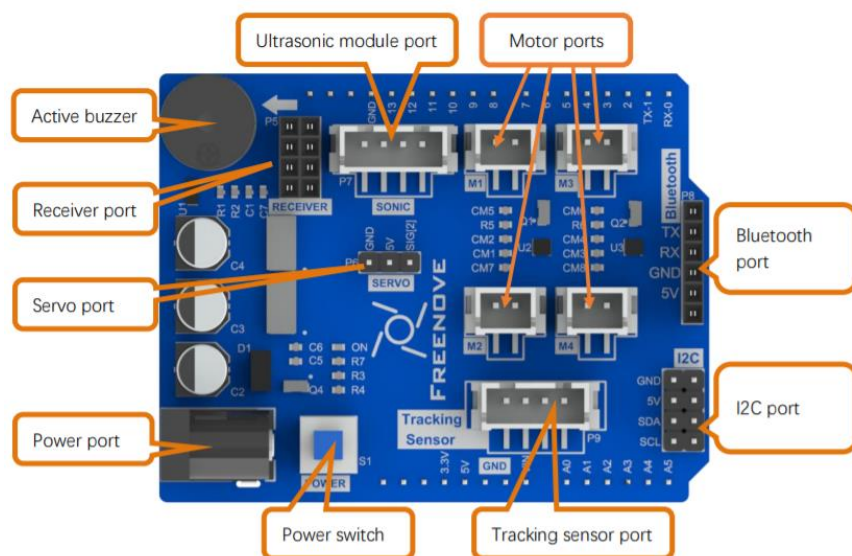
- Operating Voltage: 3.5V to 5.5V
- Operating current: 0.3mA (measuring) 60uA (standby)
- Output: Serial data
- Temperature Range: 0°C to 50°C
- Humidity Range: 20% to 90%
- Resolution: Temperature and Humidity both are 16-bit
- Accuracy: $\pm 1^\circ\text{C}$ and $\pm 1\%$

The datasheet used above is for the DHT11 temperature sensor. It supported in showing which pins are used for reasons. It shows all its specification such as power, and what type of data will be outputted. This way it became easier to use the DHT11 temperature sensor for this project.

4WD Arduino Uno Extension board

Relationship between extension board and the control board is as below.

Pins of Arduino	Ports of extension	Pin multiplexing	Description
0	UART-Bluetooth		Bluetooth
1			
2	Servo		
3	Direction-Right		Direction of right motor
4	Direction-Left		Direction of left motor
5	Motor-Right		Speed of right motor -PWM
6	Motor-Left		Speed of right motor -PWM
7	Trig		Ultrasonic module
8	Echo		
9	SPI-NRF24L01	IR-remote	CE/IR-remote
10			CS
11	SPI-NRF24L01		MOSI
12			MISO
13			SCK
A0	Battery voltage	Buzzer	The ADC uses an internal 5V reference and the acquisition voltage is the battery voltage *1/4. Output to the Buzzer through the comparator. When higher than the 4.5V, output HIGH.
A1	Line-tracking sensor		Left
A2			Middle
A3			Right
A4	I2C		SDA
A5			SCL

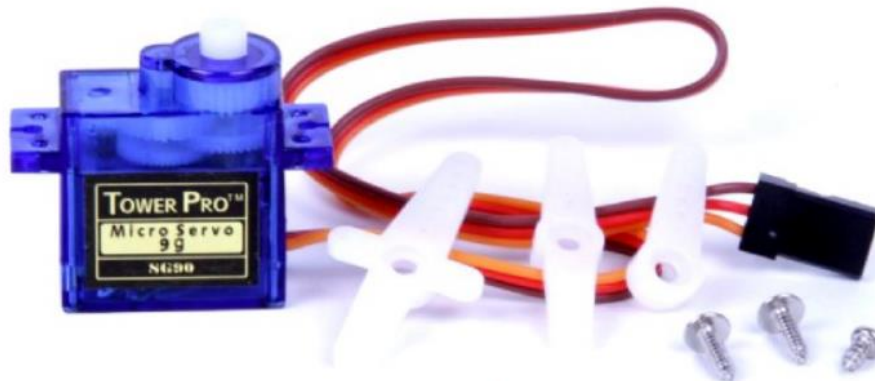


You can see in the table above where the Arduino pins correspond to the extension board ports, and you can witness which sensor wire will be inserted into which port. The reason why this has been used because it makes the system more concise and cleaner. This way it's also easier to understand as, you can easily tell which wires are connected to which sensor ports.

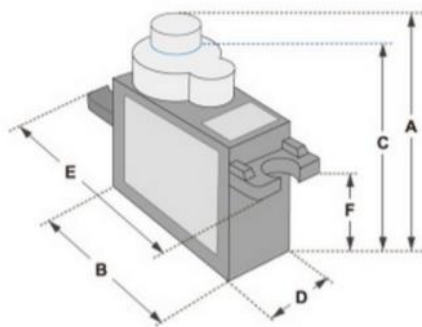
Servo motor

SERVO MOTOR SG90

DATA SHEET



Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.

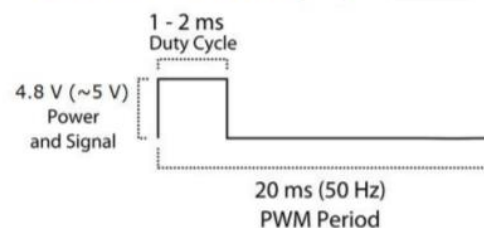


Dimensions & Specifications

A (mm) : 32
B (mm) : 23
C (mm) : 28.5
D (mm) : 12
E (mm) : 32
F (mm) : 19.5
Speed (sec) : 0.1
Torque (kg-cm) : 2.5
Weight (g) : 14.7
Voltage : 4.8 - 6

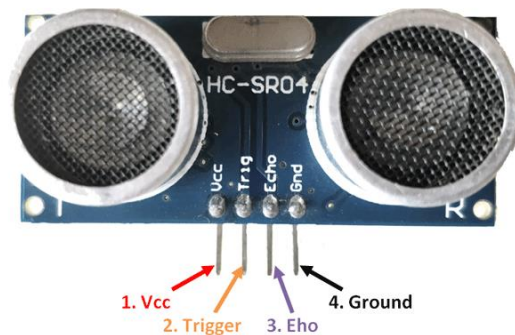
Position "0°" (1.5 ms pulse) is middle, "90°" (~2ms pulse) is middle, is all the way to the right, "-90°" (~1ms pulse) is all the way to the left.

PWM=Orange (⏏)
Vcc=Red (+)
Ground=Brown (-)



This is the datasheet for the servo motor used. It indefinites the pins and shows all the power requirement and more importantly the torque such as when rotating the ultrasonic sensor how much force it will have. All these decisions were reviewed before selecting the servo motor in this project.

Ultrasonic Sensor



Ultrasonic Sensor Pin Configuration

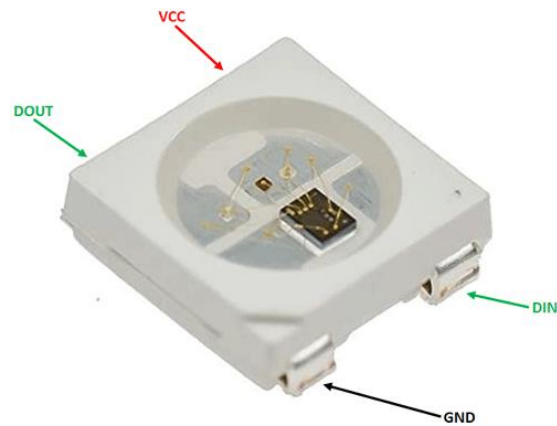
Pin Number	Pin Name	Description
1	Vcc	The Vcc pin powers the sensor, typically with +5V
2	Trigger	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
3	Echo	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4	Ground	This pin is connected to the Ground of the system.

HC-SR04 Sensor Features

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm
- Practical Measuring Distance: 2cm to 80cm
- Accuracy: 3mm
- Measuring angle covered: $<15^\circ$
- Operating Current: $<15\text{mA}$
- Operating Frequency: 40Hz

This is the datasheet used for the ultrasonic sensor. It shows where the pins needed to be connected to, gives information about the power. After, reviewing all this information, it was decided that we would use the ultrasonic sensor. This will be connected to the ultrasonic module connector.

WS2812B_LED



Pin Description of WS2812B LED

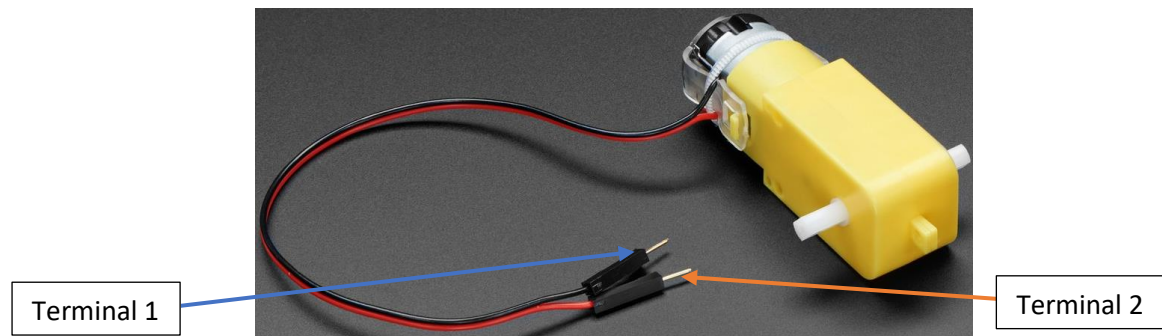
Pin Number	Pin Name	Description
1	V _{DD}	LED power supply pin
2	D _{OUT}	Data signal output pin
3	GND	Ground reference supply pin
4	D _{IN}	Data signal input pin

WS2812B LED Specifications

- Supply voltage - 3.5V to 5.3V
- Signal input voltage - 0.5V to V_{CC} + 0.5V
- Input capacitance - 15pF
- Signal supply current - 1uA

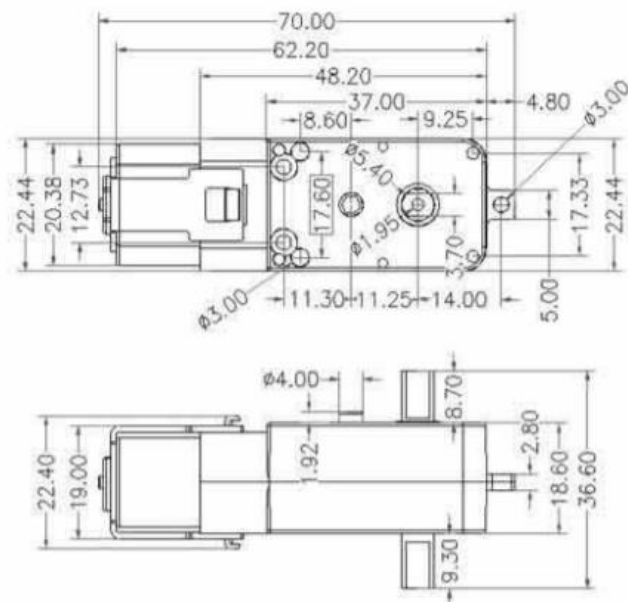
This is the datasheet for the WS2818B LED. This gives information about the LED power and shows which pins it has. After reviewing this information it was decided to use this WS2818B LED. These pins will be connected to the WS2818B LED controller.

Geared Motors



TECHNICAL DETAILS

- Rated Voltage: 3~6V
- Continuous No-Load Current: 150mA +/- 10%
- Min. Operating Speed (3V): 90+/- 10% RPM
- Min. Operating Speed (6V): 200+/- 10% RPM
- Torque: 0.15Nm ~0.60Nm
- Stall Torque (6V): 0.8kg.cm
- Gear Ratio: 1:48
- Body Dimensions: 70 x 22 x 18mm
- Wires Length: 200mm & 28 AWG
- Weight: 30.6g



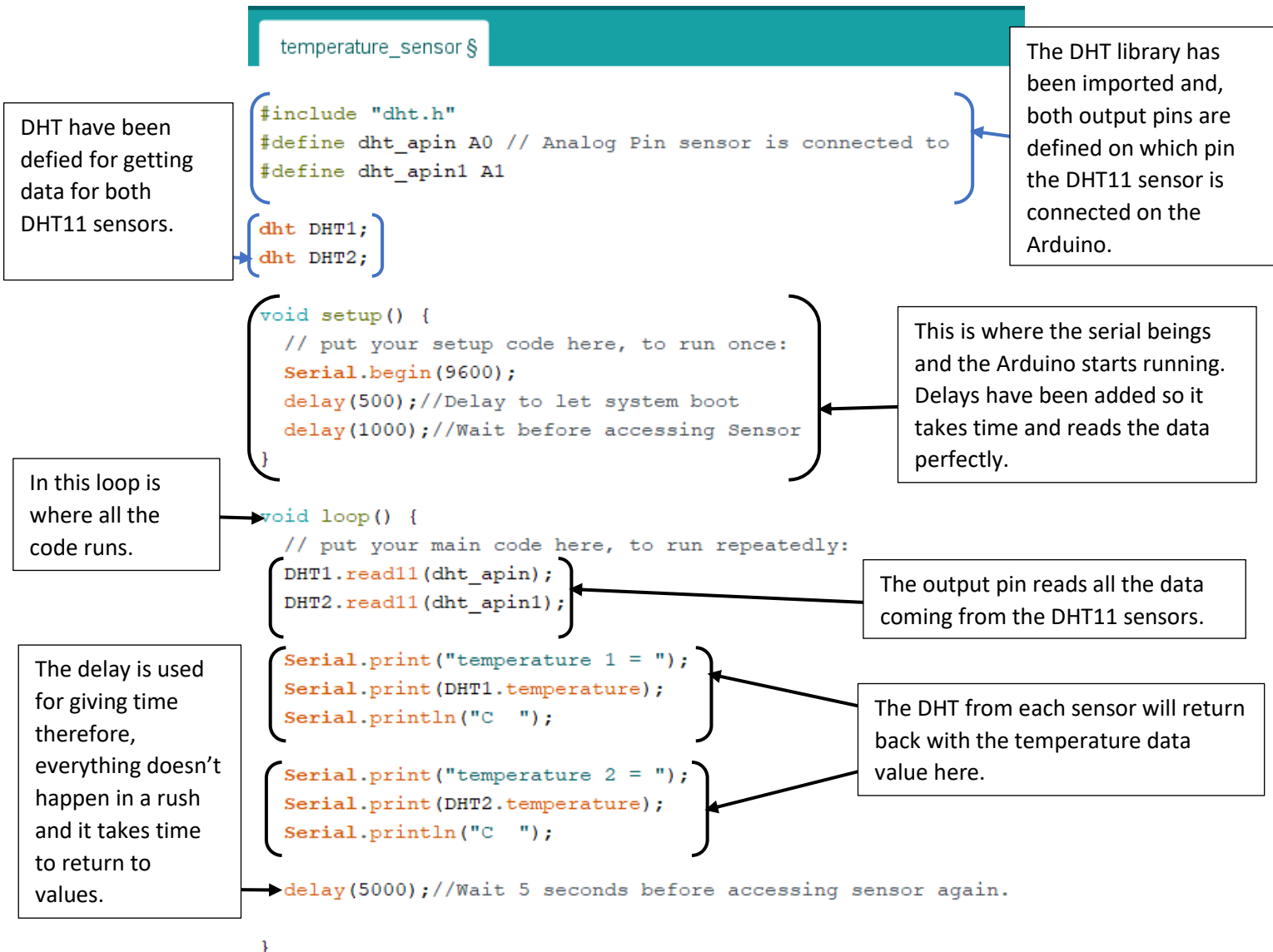
Product Weight: 30.6g / 1.1oz

This datasheet was used for the geared motors which, shows the pin configuration, the specification such as the torque for when the wheels are rotating, then the amount of force they are using therefore, there is enough torque in this geared motor. The power isn't too much as well. After, all this was reviewed, then it was decided to use the geared motor for this project.

Software Building Process

This is where I will show the annotations the code such as what algorithms are used for controlling which sensor, and how the robot responds to the sensor data and how it will navigate.

Temperature Sensor Arduino Code



Robotic Arduino Main Code

The code is long so its pasted direct from the Arduino IDE.

Start

```
#include "Freenove_WS2812B_RGBLED_Controller.h"
```

The LED library has been imported here.

```
#include <Servo.h>
```

```
#define PIN_SERVO
```

2

The Servo library and the output pin have been defined here.

```
#define I2C_ADDRESS 0x20
```

```
#define LEDS_COUNT 10 //it defines number of LEDs.
```

Defines the address of the LED for the colour combination bits. Defines how many LED are being used in the strip.

```
#define PIN_DIRECTION_RIGHT 3
```

```
#define PIN_DIRECTION_LEFT 4
```

```
#define PIN_MOTOR_PWM_RIGHT 5
```

```
#define PIN_MOTOR_PWM_LEFT 6
```

Defines the direction and PWM for the geared motors.

```
#define PIN_BATTERY A0
```

```
#define PIN_BUZZER A0
```

Defines the battery buzzer pin which is A0 according to the extension board port.

```
#define PIN_SONIC_TRIG 7
```

```
#define PIN_SONIC_ECHO 8
```

Defines the ultrasonic sensor pin which are connected to the Arduino Extension port.

```
#define OBSTACLE_DISTANCE 40
```

```
#define OBSTACLE_DISTANCE_LOW 15
```

Defines the distance for the obstacle. This will be used in the methods related to obstacle avoidance.

```
#define MAX_DISTANCE 300 //cm
```

```
#define SONIC_TIMEOUT (MAX_DISTANCE*60)
```

```
#define SOUND_VELOCITY 340 //soundVelocity: 340m/s
```

These calculations for the ultrasonic sensor will be used in the methods related to obstacle avoidance.

```
Servo servo;
```

```
byte servoOffset = 0;
```

Importing a servo here.

```
int speedOffset;//batteryVoltageCompensationToSpeed
```

```
{
int tempSensorOne;
int tempSensorTwo;
}
```

Initializes both temperature values therefore, when they are received from the serial communication then, they are stored to those variables.

```
{ Freenove_WS2812B_Controller strip(I2C_ADDRESS, LEDS_COUNT, TYPE_GRB); //initialization }
```

Initializes the LED Strip.

```
void setup() {
  Serial.begin(9600);
  pinMode(PIN_DIRECTION_LEFT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_LEFT, OUTPUT);
  pinMode(PIN_DIRECTION_RIGHT, OUTPUT);
  pinMode(PIN_MOTOR_PWM_RIGHT, OUTPUT);
  pinMode(PIN_BUZZER, OUTPUT);
  servo.attach(PIN_SERVO);
  calculateVoltageCompensation();
}
```

This is where the Arduino serial begins and, all the pin mode has been set and active therefore, the robot can navigate around. The Servo motor is also attached which means that will be active once the Arduino serial beings. It gets the amount of voltage used by the calculateVoltageCompensation() method.

```
void TempHigher() {
  //Move back
  motorRun(-100, -100);
  delay(1000);
  motorRun(-100, -100);
  delay(1000);
  motorRun(-100, -100);
  delay(1000);
  motorRun(-100, -100);
  delay(1000);
  motorRun(0, 0);
}
```

This method sets the navigation path of the robot when the temperature is high. The motor run method is used to set the moving direction of the robot with a speed in the second parameter. The delays are added so it keeps things calm as, if its too fast, then there are more chances for the system crashing. In the comments you can witness the navigation path such as where the robot is going to move e.g., // Move Back.

```
//Left motors rotate to one direction
```

```
digitalWrite(PIN_DIRECTION_LEFT, HIGH);
analogWrite(PIN_MOTOR_PWM_LEFT, 100);
delay(1000);

//Stop
analogWrite(PIN_MOTOR_PWM_LEFT, 0);
delay(1000);

//left motors rotate to opposite direction
digitalWrite(PIN_DIRECTION_LEFT, LOW);
analogWrite(PIN_MOTOR_PWM_LEFT, 255);
delay(1000);

//Stop
analogWrite(PIN_MOTOR_PWM_LEFT, 0);
delay(1000);

//Move forward
motorRun(80, 80);
delay(1000);
motorRun(80, 80);
delay(1000);
motorRun(0, 0);

//Left motors rotate to one direction
digitalWrite(PIN_DIRECTION_LEFT, HIGH);
analogWrite(PIN_MOTOR_PWM_LEFT, 100);
delay(1000);

//Stop
analogWrite(PIN_MOTOR_PWM_LEFT, 0);
delay(1000);

//left motors rotate to opposite direction
```

```
digitalWrite(PIN_DIRECTION_LEFT, LOW);
analogWrite(PIN_MOTOR_PWM_LEFT, 255);
delay(1000);
//Stop
analogWrite(PIN_MOTOR_PWM_LEFT, 0);
delay(1000);
```

```
//Move forward
motorRun(130, 130);
delay(1000);
motorRun(130, 130);
delay(1000);
motorRun(0, 0);
```

```
delay(2000);
```

```
{
digitalWrite(PIN_BUZZER, HIGH); //turn on buzzer
delay(50);
digitalWrite(PIN_BUZZER, LOW); //turn off buzzer
}
```

The buzzer pins are passed into the parameter which will sound the buzzer once the navigation has been completed, this is used to alert that the navigation has completed.

```
void TempLower() {
```

```
//Move back
motorRun(-100, -100);
delay(1000);
motorRun(-100, -100);
delay(1000);
motorRun(-100, -100);
delay(1000);
motorRun(-100, -100);
```

This method sets the navigation path of the robot when the temperature is cold. The motor run method is used to set the moving direction of the robot with a speed in the second parameter. The delays are added so it keeps things calm as, if it's too fast, then there are more chances for the system crashing. In the comments you can witness the navigation path such as where the robot is going to move e.g., // Move Back.

```
delay(1000);
motorRun(0, 0);
delay(1000);
motorRun(0, 0);
delay(1000);
motorRun(0, 0);

//Right motors rotate to one direction
digitalWrite(PIN_DIRECTION_RIGHT, HIGH);
analogWrite(PIN_MOTOR_PWM_RIGHT, 100);
delay(1000);
analogWrite(PIN_MOTOR_PWM_RIGHT, 0);
delay(1000);

//Right motors rotate to opposite direction
digitalWrite(PIN_DIRECTION_RIGHT, LOW);
analogWrite(PIN_MOTOR_PWM_RIGHT, 255);
delay(1000);
analogWrite(PIN_MOTOR_PWM_RIGHT, 0);
delay(1000);

//Left motors rotate to one direction
digitalWrite(PIN_DIRECTION_LEFT, HIGH);
analogWrite(PIN_MOTOR_PWM_LEFT, 100);
delay(1000);
//Stop
analogWrite(PIN_MOTOR_PWM_LEFT, 0);
delay(1000);

//left motors rotate to opposite direction
digitalWrite(PIN_DIRECTION_LEFT, LOW);
```

```
analogWrite(PIN_MOTOR_PWM_LEFT, 255);  
delay(1000);  
//Stop  
analogWrite(PIN_MOTOR_PWM_LEFT, 0);  
delay(1000);
```

```
//Move forward  
motorRun(80, 80);  
delay(1000);  
motorRun(80, 80);  
delay(1000);  
motorRun(0, 0);
```

```
//Move forward  
motorRun(130, 130);  
delay(1000);  
motorRun(130, 130);  
delay(1000);  
motorRun(0, 0);
```

```
delay(2000);
```

```
{  
  digitalWrite(PIN_BUZZER, HIGH); //turn on buzzer  
  delay(50);  
  digitalWrite(PIN_BUZZER, LOW); //turn off buzzer  
}
```

The buzzer pins are passed into the parameter which will sound the buzzer once the navigation has been completed, this is used to alert that the navigation has completed.

```
void MakeACircle() {
```

This method just goes in a circle in short delays. This will be used when both, temperature values are equal.

```
//Right motors rotate to one direction  
digitalWrite(PIN_DIRECTION_RIGHT, HIGH);
```



```
analogWrite(PIN_MOTOR_PWM_RIGHT, 100);  
delay(1000);  
analogWrite(PIN_MOTOR_PWM_RIGHT, 0);  
delay(1000);
```

```
//Right motors rotate to opposite direction  
digitalWrite(PIN_DIRECTION_RIGHT, LOW);  
analogWrite(PIN_MOTOR_PWM_RIGHT, 255);  
delay(1000);  
analogWrite(PIN_MOTOR_PWM_RIGHT, 0);  
delay(1000);
```

```
//Right motors rotate to one direction  
digitalWrite(PIN_DIRECTION_RIGHT, HIGH);  
analogWrite(PIN_MOTOR_PWM_RIGHT, 100);  
delay(1000);  
analogWrite(PIN_MOTOR_PWM_RIGHT, 0);  
delay(1000);
```

```
//Right motors rotate to opposite direction  
digitalWrite(PIN_DIRECTION_RIGHT, LOW);  
analogWrite(PIN_MOTOR_PWM_RIGHT, 255);  
delay(1000);  
analogWrite(PIN_MOTOR_PWM_RIGHT, 0);  
delay(1000);
```

```
//Right motors rotate to one direction  
digitalWrite(PIN_DIRECTION_RIGHT, HIGH);  
analogWrite(PIN_MOTOR_PWM_RIGHT, 100);  
delay(1000);  
analogWrite(PIN_MOTOR_PWM_RIGHT, 0);
```

```
delay(1000);

//Right motors rotate to opposite direction
digitalWrite(PIN_DIRECTION_RIGHT, LOW);
analogWrite(PIN_MOTOR_PWM_RIGHT, 255);
delay(1000);
analogWrite(PIN_MOTOR_PWM_RIGHT, 0);
delay(1000);

//Right motors rotate to one direction
digitalWrite(PIN_DIRECTION_RIGHT, HIGH);
analogWrite(PIN_MOTOR_PWM_RIGHT, 100);
delay(1000);
analogWrite(PIN_MOTOR_PWM_RIGHT, 0);
delay(1000);


//Right motors rotate to opposite direction
digitalWrite(PIN_DIRECTION_RIGHT, LOW);
analogWrite(PIN_MOTOR_PWM_RIGHT, 255);
delay(1000);
analogWrite(PIN_MOTOR_PWM_RIGHT, 0);
delay(1000);

}
```

```
void loop() {
```

```
  while (!strip.begin());
```

This is where the LED strip begins and it gets all the LED values.



```
    if (Serial.available() > 0) {
      tempSensorOne = Serial.read();
      Serial.println(tempSensorOne);
      Serial.println("");
    }
```

```

delay(2000);

tempSensorTwo = Serial.read();
Serial.println(tempSensorTwo);
Serial.println("");
delay(2000);
}

```

This is the main serial communication part where it's getting the values from the temperature sensor Arduino. First it checks if the serial is available once it's available, then using `Serial.read()` it passes each temperature value into the integer variables which were defined for each different temperature sensor value earlier. After that, the temperature values were received in the program. The way we checked that is by inserting the USB cable and then, printing the values to the serial monitor for checking if the values were being received and the temperature values were received.

```

if (tempSensorOne >= tempSensorTwo) {
    Serial.println("The Temperature is Hot...");
    strip.setAllLedsColor(0xFF0000); //Set all LED color to red
    updateAutomaticObstacleAvoidance();
    TempHigher();
    updateAutomaticObstacleAvoidance();
    Serial.println("");
} else if (tempSensorOne <= tempSensorTwo) {
    Serial.println("The Temperature is Cold...");
    strip.setAllLedsColor(0x00FF00); //set all LED color to green
    updateAutomaticObstacleAvoidance();
    TempLower();
    updateAutomaticObstacleAvoidance();
    Serial.println("");
} else if (tempSensorOne == tempSensorTwo) {
    Serial.println("Both Temperature Sensor Are Equal");
    strip.setAllLedsColor(0x0000FF); //set all LED color to blue
    updateAutomaticObstacleAvoidance();
    MakeACircle();
    Serial.println("");
}

```

Using the temperature values it checks if the temperature one is greater than temperature two and if true then, that means it's in the hot category where it will execute the higher temperature method to change the robot navigation path. It will set all LEDs to red. The obstacle avoidance method will be used for checking if there are any obstacles in the way and, if they are then slightly change navigation path to avoid that.

Using the temperature values it checks if the temperature one is less than temperature two and if true then, that means it's in the cold category where it will execute the lower temperature method to change the robot navigation path. It will set all LEDs to green. The obstacle avoidance method will be used for checking if there are any obstacles in the way and, if they are then slightly change navigation path to avoid that.

Using the temperature values it checks if the temperature one is equal to temperature two and if true then it makes a circle around the room/area by using the method. It will set all LEDs to blue. The obstacle avoidance method will be used for checking if there are any obstacles in the way and, if they are then slightly change navigation path to avoid that.

```

delay(2000);

```

Delays are used for giving pauses so everything doesn't happen in a rush.

```
}
```

```
void motorRun(int speedl, int speedr) {
```

This is the motor run method. It uses algorithms, to calculate and set the speeds and directions.

```
    int dirL = 0, dirR = 0;
```

```
    if (speedl > 0) {
```

```
        dirL = 0;
```

```
    }
```

```
    else {
```

```
        dirL = 1;
```

```
        speedl = -speedl;
```

```
    }
```

```
    if (speedr > 0) {
```

```
        dirR = 1;
```

```
    }
```

```
    else {
```

```
        dirR = 0;
```

```
        speedr = -speedr;
```

```
    }
```

```
    digitalWrite(PIN_DIRECTION_LEFT, dirL);
```

```
    digitalWrite(PIN_DIRECTION_RIGHT, dirR);
```

```
    analogWrite(PIN_MOTOR_PWM_LEFT, speedl);
```

```
    analogWrite(PIN_MOTOR_PWM_RIGHT, speedr);
```

```
}
```

The direction and speed value calculated from the if statement will be passed to the geared motor PWM. Therefore, this is applied onto the robotic wheels when navigating.

```
void updateAutomaticObstacleAvoidance() {
```

```
    int distance[3], tempDistance[3][5], sumDisntance;
```

```
    static u8 leftToRight = 0, servoAngle = 0, lastServoAngle = 0; //
```

```
    const u8 scanAngle[2][3] = { {150, 90, 30}, {30, 90, 150} };
```

```
    for (int i = 0; i < 3; i++)
```

This algorithm is written for avoiding the obstacle which comes in front of the ultrasonic sensor. It checks for obstacles straight, left, and right and depending on the if statement it changes the angles slightly so the robot doesn't crash into the object. To slightly move the robot it uses the motor run method which, we already discussed about earlier above.

```
{
  servoAngle = scanAngle[leftToRight][i];
  servo.write(servoAngle);
}
```

This piece of code is rotating the servo motor which has the ultrasonic sensor on it. It rotates it +90 to right, and -90 to left.

```
if (lastServoAngle != servoAngle) {
  delay(130);
}
lastServoAngle = servoAngle;
for (int j = 0; j < 5; j++) {
  tempDistance[i][j] = getSonar();
  delayMicroseconds(2 * SONIC_TIMEOUT);
  sumDisntance += tempDistance[i][j];
}
```

The getSonor method is used for calculating the distance between the sensor and the object.

```
if (leftToRight == 0) {
  distance[i] = sumDisntance / 5;
}
else {
  distance[2 - i] = sumDisntance / 5;
}
sumDisntance = 0;
}
leftToRight = (leftToRight + 1) % 2;
```

In the obstacle avoidance method using those comments you can check what the if statement is being used for. This is just one example of a comment shown but there many throughout this method.

```
if (distance[1] < OBSTACLE_DISTANCE) { //Too little distance ahead
  if (distance[0] > distance[2] && distance[0] > OBSTACLE_DISTANCE) { //Left distance is greater
    than right distance
    motorRun(-(150 + speedOffset), -(150 + speedOffset)); //Move back
    delay(100);
    motorRun(-(150 + speedOffset), (150 + speedOffset));
  }

  else if (distance[0] < distance[2] && distance[2] > OBSTACLE_DISTANCE) { //Right
    distance is greater than left distance
    motorRun(-(150 + speedOffset), -(150 + speedOffset)); //Move back
```



```

    delay(100);
    motorRun((150 + speedOffset), -(150 + speedOffset));
}
else {          //Get into the dead corner, move back, then turn.
    motorRun(-(150 + speedOffset), -(150 + speedOffset));
    delay(100);
    motorRun(-(150 + speedOffset), (150 + speedOffset));
}
}
else {          //No obstacles ahead
    if (distance[0] < OBSTACLE_DISTANCE_LOW) {    //Obstacles on the left front.
        motorRun(-(150 + speedOffset), -(150 + speedOffset)); //Move back
        delay(100);
        motorRun((180 + speedOffset), (50 + speedOffset));
    }
    else if (distance[2] < OBSTACLE_DISTANCE_LOW) {    //Obstacles on the right front.
        motorRun(-(150 + speedOffset), -(150 + speedOffset)); //Move back
        delay(100);
        motorRun((50 + speedOffset), (180 + speedOffset));
    }
    else {          //Cruising
        motorRun((80 + speedOffset), (80 + speedOffset));
    }
}
}

```

```

float getSonar() {
    unsigned long pingTime;
    float distance;

```

This method is used for the ultrasonic sensor. it used the pins which have been defined earlier meaning which pin is the sensor connected to in the Arduino. This calculates the distance between the ultrasonic sensor and the object and it returns back the distance.

```

    digitalWrite(PIN_SONIC_TRIG, HIGH); // make trigPin output high level lasting for 10µs to trigger
    HC_SR04,

```

```
delayMicroseconds(10);

digitalWrite(PIN_SONIC_TRIG, LOW);

pingTime = pulseIn(PIN_SONIC_ECHO, HIGH, SONIC_TIMEOUT); // Wait HC-SR04 returning to the
high level and measure out this waiting time

if (pingTime != 0)

    distance = (float)pingTime * SOUND_VELOCITY / 2 / 10000; // calculate the distance according to
the time

else

    distance = MAX_DISTANCE;

return distance; // return the distance value
}
```

```
void calculateVoltageCompensation() {
    float voltageOffset = 8.4 - getBatteryVoltage();
    speedOffset = voltageOffset * 20;
}
```

← This method
calculates the
amount of voltage
used.

```
float getBatteryVoltage() {
    pinMode(PIN_BATTERY, INPUT);

    int batteryADC = analogRead(PIN_BATTERY);

    float batteryVoltage = batteryADC / 1023.0 * 5.0 * 4;

    return batteryVoltage;
}
```

← This method by getting the battery
pin calculates how much battery
voltage has been used.

Testing

The first type of testing was done using the following approach: when each sensor was installed to the robot, then the program code was tested onto it for checking if the sensor is working or not. This approach was used for all the testing and, all the tests managed to pass which proved two things first, the sensors have no problems, and second the sensors were wired accurately onto the Arduino controller.

Below, is the test plan based on the requirements of the robotic system, in this you can witness which tests have passed and which tests have failed. The way to figure that out if that the green highlighted tests will be passed whereas, the red highlighted tests will fail.

Test No.	Expected Outcome	Actual Outcome
1.)	If the first temperature sensor is higher then, the second temperature sensor, then flash the red led lights.	When the first temperature sensor is higher then, the second temperature sensor, then the red LED lights are flashed.
2.)	When the red led lights are flashing, navigate out of that room/area using the wheels.	When the red lights are flashing, then the wheels are used to navigate out of the room area.
3.)	If the second temperature sensor is lower then, the first temperature sensor, then flash the green led lights.	When the second temperature sensor is higher then, the first temperature sensor, then the green LED lights are flashed.
4.)	When the green led lights are flashing, navigate out of that room/area using the wheels.	When the green lights are flashing, then the wheels are used to navigate out of the room area.
5.)	When navigating the ultrasonic module should move 180 degrees using the servo motor.	When the robot is navigating, then the servo motors turn the ultrasonic module 90 degrees to the left and right.
6.)	If the object is near then the ultrasonic sensor should capture that and stop the robot from hitting the object.	This is where the problem is when the ultrasonic sensor captures the object it still hits the object. The obstacle avoidance algorithm still executes are tries to move the robot away from the object but, its not fast enough or the algorithm doesn't seem to be accurate as, a result the robot does crash into the object. That is why for the current robotic version its recommended to use in rooms and areas where there are less objects.
7.)	If first and second temperature both match together, then the robot should make a circle in the room flashing the blue led light.	When both temperatures are equal, then it flashes the blue LED lights and rotates around in a circle around the area/room.
8.)	At the start and end of the navigation path there should	At the start and end of every navigation function the buzzer

	be a sound of the buzzer to indicate that the path has completed navigating and its moving to the next path.	sounds to alert users that the function has completed.
9.)	Serial communication between both Arduino's to send the temperature value from the Arduino connected with temperature sensor to the main robot system Arduino.	Serial communication is sending data from Arduino temperature code to the main robotic code which is, then used in algorithms.
10.)	Everything should happen but, with short pauses and delays as, this way the robot should be in more control.	The robotic code delays are working perfectly fine which can, be seen when the robot is navigating as, you see from the demonstration that it will pause and, then perform the next manoeuvre.

All, the tests have passed apart from one test which is the object avoidance requirement. Therefore, when working on improvements from the second version this requirement needs to be researched more and then met.

Further Improvements

Below, I'm going to talk about three major improvements that should be met for the next version of this robotic system after further amendments. You can see those improvements listed below:

- As, it can be witnessed from the test plan that object avoidance requirement number 6 has failed. Therefore, to improve that we need to research the area in more detail and come down to the actual problem and tackle it. So far, we know there is no issues with the ultrasonic sensor since, all sensors were individually tested using code which had all passed. Therefore, we know the error or bug is in the object avoidance algorithm. Therefore, we need to break the algorithm down and test each line of code in steps and see where the error is being cause e.g., check by which line of code the robot is still crashing into the object. Therefore, once we find those disinfected lines of code, then try to make amendments to it and try different examples to see if it works or not. Even try a different technique such as insert the navigation code such as temp higher method into the object avoidance algorithm where the motor run method is used as, a result this trial and error might work. Also try researching and doing more detailed work on the different types of object avoidance algorithms therefore, when we come to write the object avoidance algorithm for the second time, then hopefully it should work.

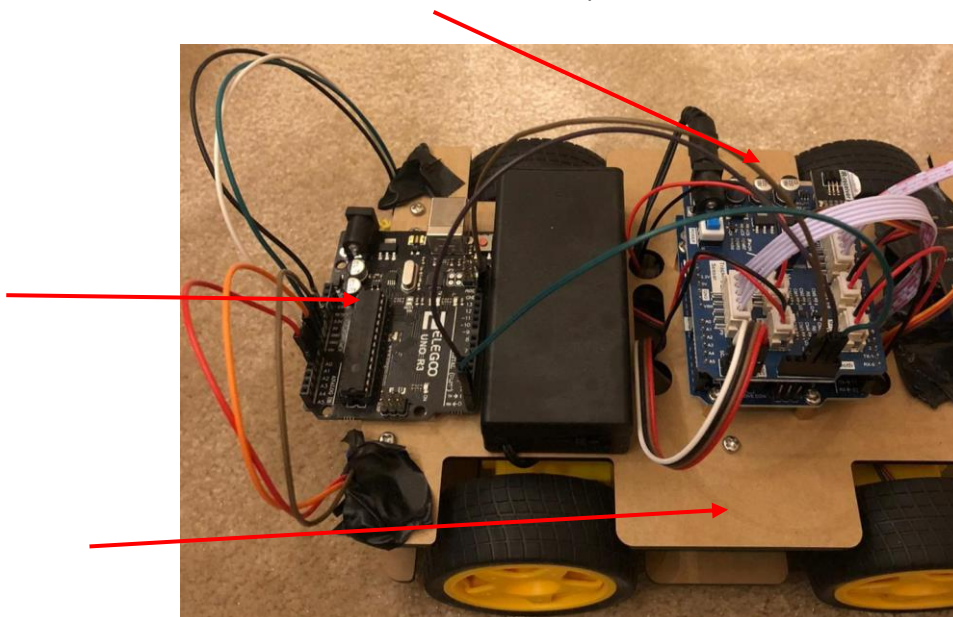
```

if (distance[1] < OBSTACLE_DISTANCE) {           //Too little distance ahead
  if (distance[0] > distance[2] && distance[0] > OBSTACLE_DISTANCE) {      //Left
    motorRun(-(150 + speedOffset), -(150 + speedOffset)); //Move back
    delay(100);
    motorRun(-(150 + speedOffset), (150 + speedOffset));
  }
  else if (distance[0] < distance[2] && distance[2] > OBSTACLE_DISTANCE) {
    motorRun(-(150 + speedOffset), -(150 + speedOffset)); //Move back
    delay(100);
    motorRun((150 + speedOffset), -(150 + speedOffset));
  }
  else {                                           //Get into the dead corner, move back, then turn.
    motorRun(-(150 + speedOffset), -(150 + speedOffset));
    delay(100);
    motorRun(-(150 + speedOffset), (150 + speedOffset));
  }
}
}

```

The image above shows the code which is a part of object avoidance. As, you can see the motor run function is being used here therefore, this is where the trial and error can be performed by trying to add the robotic navigation path e.g., temp higher into this area of the object avoidance algorithm. This is just one example of a solution. The others I've already mentioned above for this point.

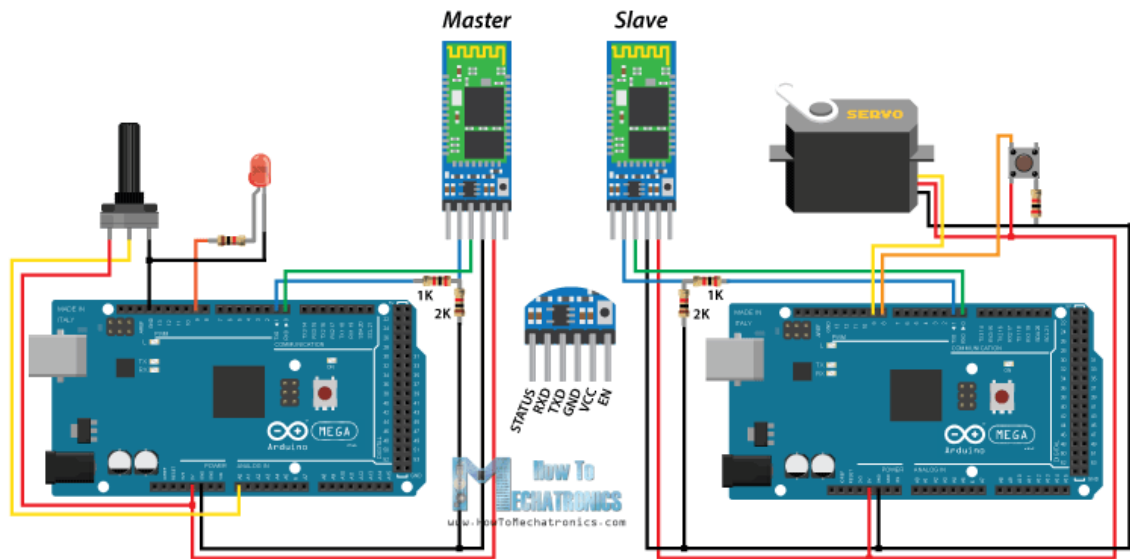
- The next further development is to add 3 more ultrasonic sensors around the robot. The example can be seen below where you can see red arrows and these red arrows indicate where the other 3 ultrasonic sensors should be placed:



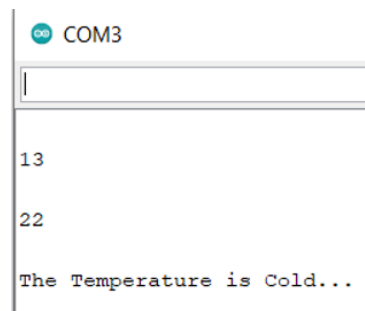
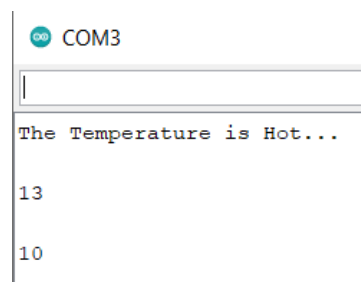
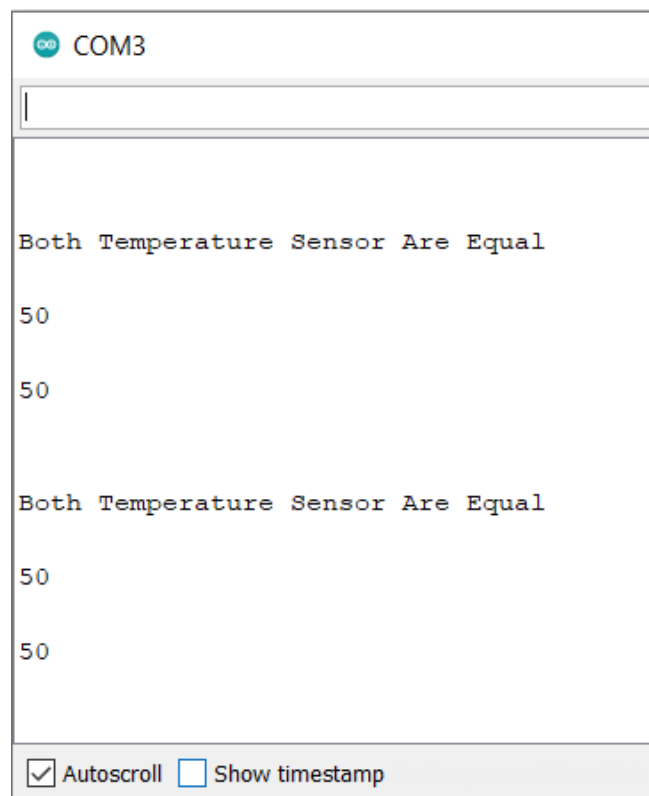
Therefore, after installing the 3 servo motors and ultrasonic sensor to these positions, then there will be ultrasonic sensor on the front, right, left, and the back side of the robot. The reason why this is needed is because, the robot navigation algorithms give a lot of instruction in terms of turning back, left, and right therefore, the front ultrasonic sensor can only move 90 degrees on each side to capture objects as, a result it can't really detect the objects on the left right, and especially not at the back. As, a result these 3 extra algorithms will do the same job as they will turn 90 degrees each side to check if there are any objects on the left, right, or the back side as, result using the algorithm it will stop the robot from

hitting the object from any side as, a result this will keep the robot protected and not let it get damaged. Therefore, this is a really good further development after the obstacle avoidance algorithm has been fixed as, once the obstacle avoidance algorithm is corrected then, extend that algorithm further to the 3 other ultrasonic sensors for making this development working for the next improved version of the robot.

- The final improvement is to use a Bluetooth module to communicate between the two Arduino wirelessly. You can witness this in the image below:



There are two reasons for that first, there will be less jump wires used because, right now it can be witnessed that the jump wires are too much and they are stretched when they are pinned to the RX and TX pins and that can cause the wire to break causing the whole system to break. Therefore, by using the Bluetooth module there will be less wires which makes it easier to understand such as the development team for the second version can easily understand what is happening and the robot connections will look organised. The second, thing is that at the moment is we want to see the actual temperature values, then we have to connect the Arduino robot to the PC via USB and, then open serial monitor to witness the temperature values for both DHT11 sensors. This is a drawback as; you have to stop your system in terms of the navigation just to check the temperature values. Therefore, if the Bluetooth module is connected, then the Bluetooth can be accessed using your mobile phones or laptop using an external application or an external serial monitor as, a result the users can see the DHT11 temperature values as the same time while the robot is navigating. This way it will do both things at the same time, which is more reliable for uses as, they can actually witness the temperature values themselves for each area in the room which is not applicable right now. Therefore, this was the next further development which has been considered for the second version of the robot.



The image above shows the temperature sensor printing on the serial monitor. This is happening because the robot Arduino is connected via USB cable as, without that this would not work on the Arduino IDE serial monitor. That is why if we use a Bluetooth module, then the serial printing would work the same way as showing in the image above without using the USB cable. This is another reason why the Bluetooth module was considered as an improvement. Therefore, this was the next further development which has been considered for the second version of the robot.