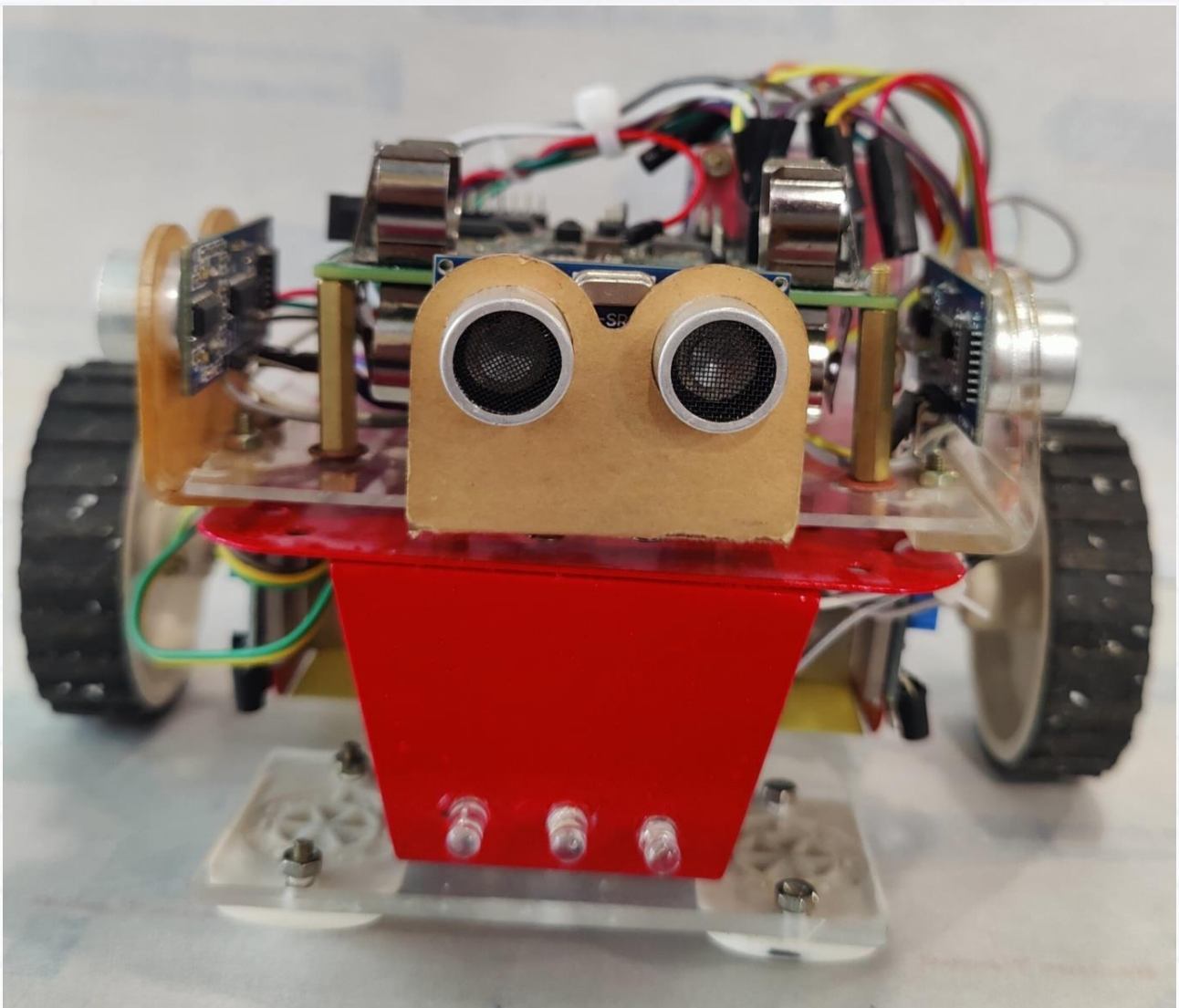
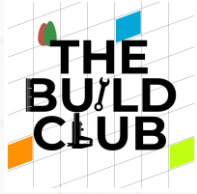


Build an autonomous robot that can navigate a maze!





Index

Prerequisites

Aim

Components

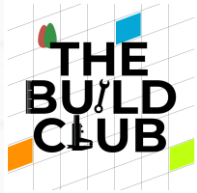
Part 1: Experimenting with the Sensors & Motors

- 1) IR Sensors
- 2) Ultrasonic Sensors
- 3) DC Motors

Part 2: Build a Maze Solving Robot

- Flowchart of the Program
- Encoding Each Step
- Algorithm of the final code
- Implementation

Challenges



Prerequisites

Topic	Resources
Rolling Display Project	Build Club website
Understanding Sensors & Actuators	Project Videos

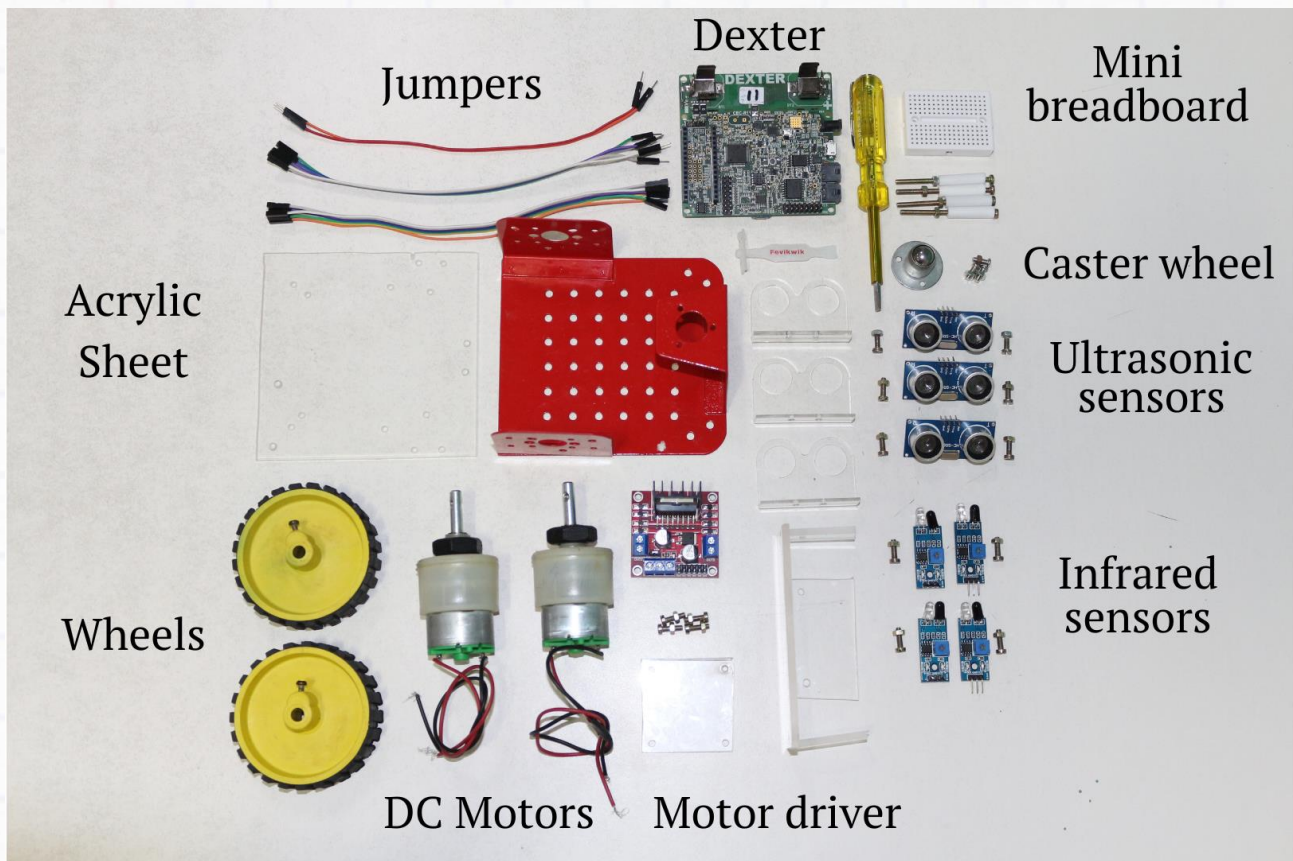
Aim

Build a Micro-mouse robot equipped with sensors and wheels which can autonomously navigate and solve a maze.

Components (*Provided in kits sent by Build Lab - IITM)

1. Dexter board
2. IR Sensors - 4
3. Ultrasonic sensors - 3
4. L298 Motor driver
5. 2 DC Motors
6. 2 Wheels
7. Caster wheel
8. Metal Chassis
9. Acrylic sheet cutouts - 8 pieces
10. Screws & nuts
11. Screwdriver
12. Superglue
13. Double sided tape
14. Mini Breadboard (170 point)
15. Jumper wires (male to female - 9 nos.)
16. Two 3.6V NMC cells

17. 2 USB cables - 1 for powering the Dexter and the other for uploading code from laptop to Dexter.

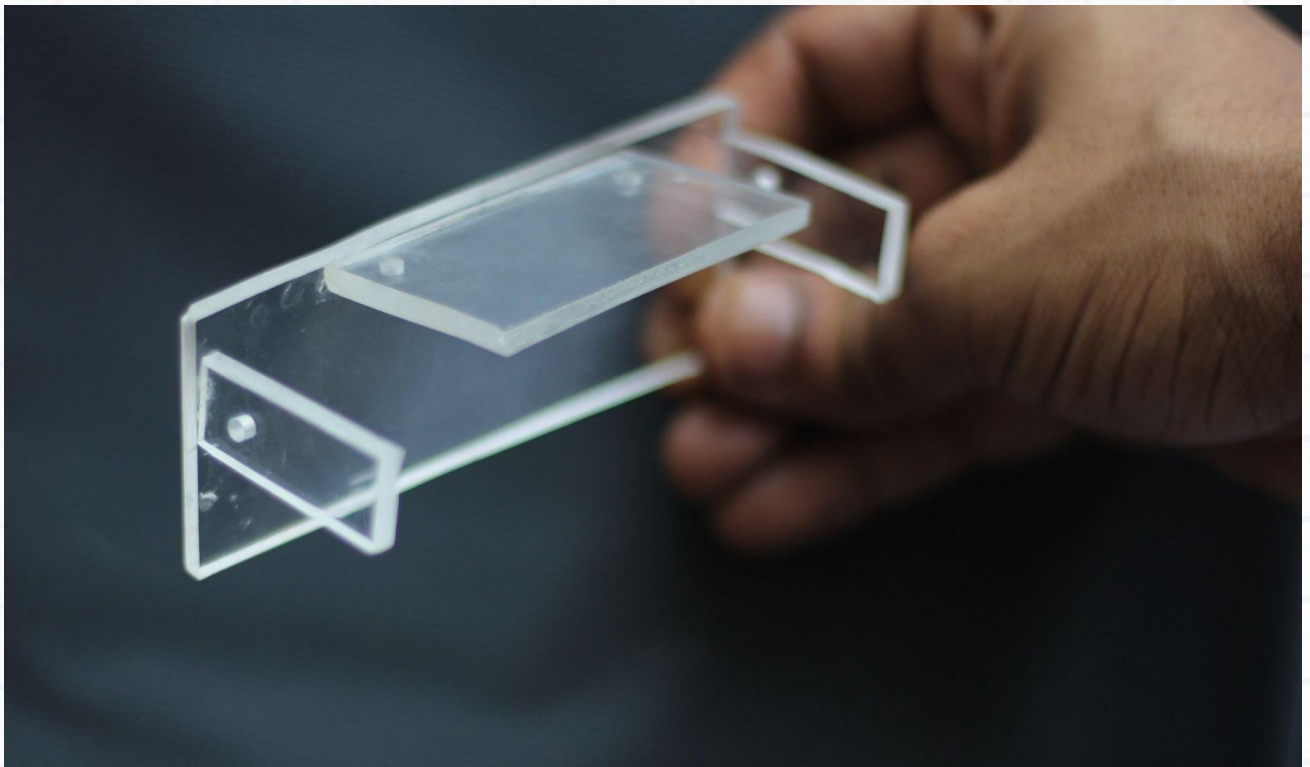


Part 1: Experimenting with the Sensors & Motors

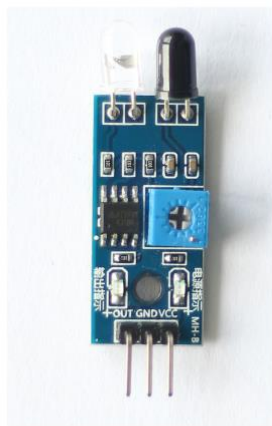
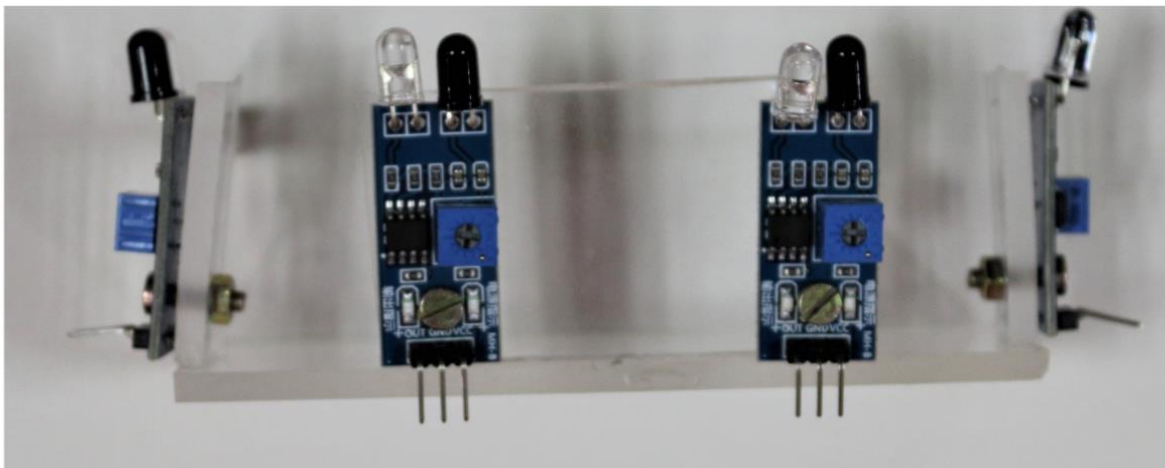
IR Sensors

Assembly:

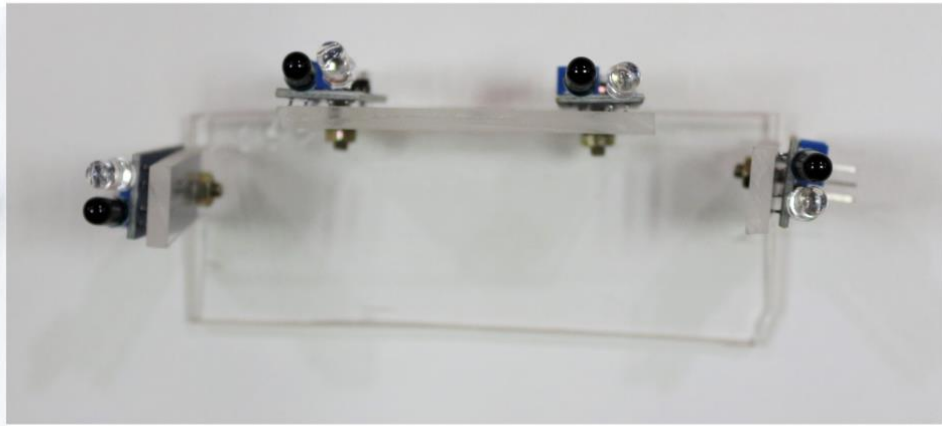
Take the acrylic cutouts for the IR sensors and superglue them to make the following shape:



Next, using 4 sets of screws & nuts, fix the 4 IR sensors on the setup as shown:



IR Sensor



IR Sensors connected on the acrylic structure

Connections:

NOTE: Before starting the connections, verify using a multimeter that all the jumper wires are working. Also ensure that the connections are strong, else the setup may not work. Make sure the connections are connected at respective pins.

Take 12 **female-to-male** jumper wires and connect them to the pins (OUT, GND, VCC) of the 4 IR sensors. Connect the other ends as per the below connections:

Important

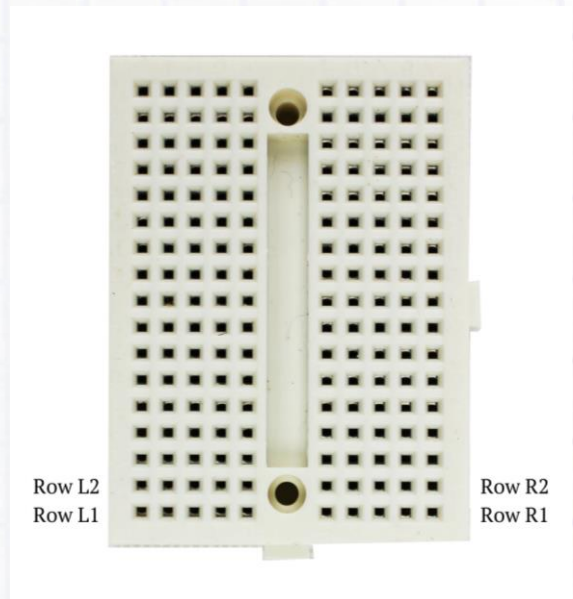


Sensor's Pin	Connect to...			
	Left sensor	Left_centre sensor	Right_centre sensor	Right sensor
OUT	PA6	PA7	PC6	PC7
GND	Breadboard Row R2	Breadboard Row R2	Breadboard Row R2	Breadboard Row R2
VCC	Breadboard Row R1	Breadboard Row R1	Breadboard Row R1	Breadboard Row R1

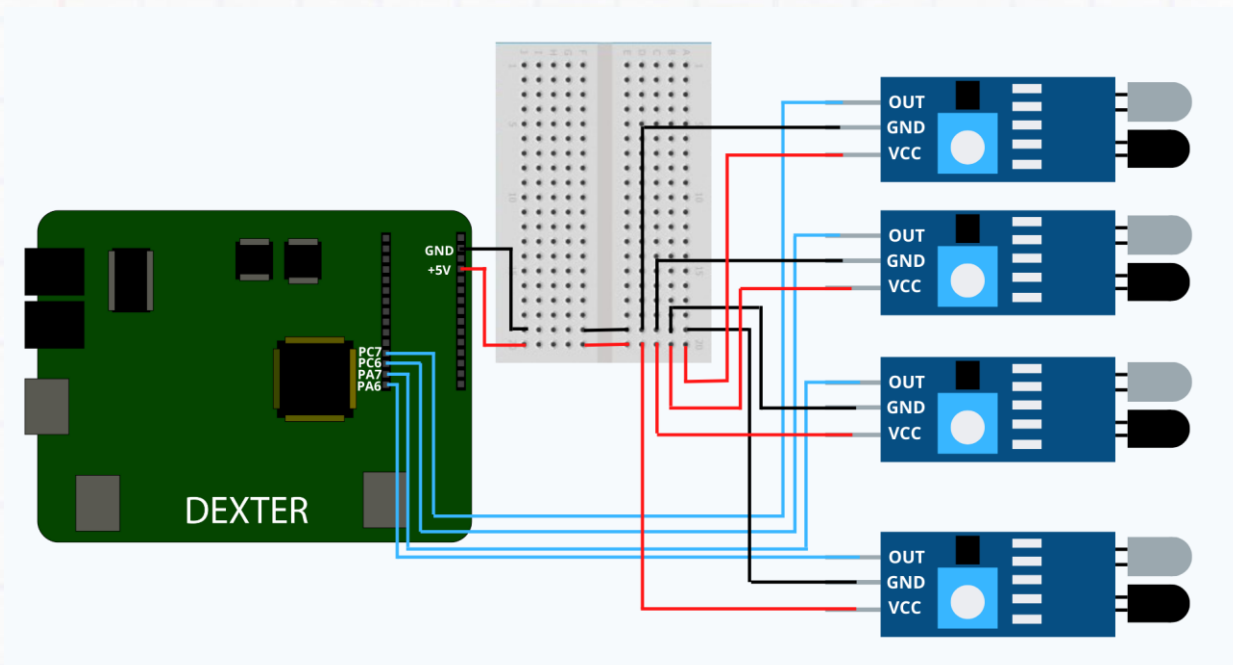
Take 2 male-to-male jumper wires and connect them as follows:

- **+5V** of Dexter to **Row R1** of breadboard
- **GND** of Dexter to **Row R2** of breadboard

Using 2 male-to-male jumpers, connect Row L1 to Row R1 and Row L2 to Row R2. This is known as 'shorting'.

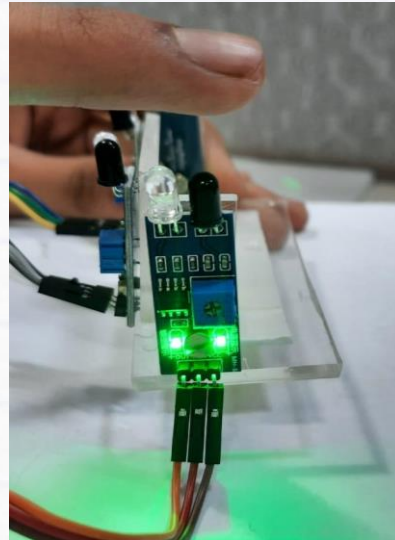
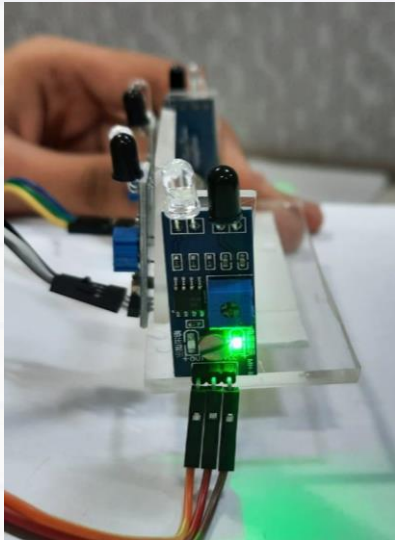


Your final circuit should look like this:



Once done, connect the Dexter to your computer using the two USB cables.

A green LED should glow on each IR sensor if the connections are correct. On hovering your hand over any of the sensors, another green LED should glow, indicating that an object is detected by the sensor.



Software & Testing:

Downloads & Setup

- 1) To download and install the STM32 Cube IDE (if you haven't already), follow the steps given in the Dexter Base Document, part 2.
- 2) Next, download the Project Workspace file '**Maze_Robot.zip**' given in the project page on the Build Club website.
- 3) In the C: drive on your computer, navigate to the '**Workspace**' folder created in previous Build projects. Inside it, create a new folder named '**Maze_Robot**'.
- 4) Now, as done in every project: i) Launch the STM IDE, ii) Select the **Maze_Robot** folder as workspace, iii) Import the **Maze_Robot.zip** file and iv) Navigate to **app.c**.

Code



There are 4 code functions for the IR sensors, one for each sensor. The functions return a '0' if an object is detected and a '1' otherwise.

```
Read_L_IR (); // Reads the Left IR sensor  
Read_LC_IR (); // Reads the Left Centre IR sensor  
Read_RC_IR (); // Reads the Right Centre IR sensor  
Read_R_IR (); // Reads the Right IR sensor
```

Eg. To store the status of Left IR sensor, you can save it in an int variable as shown:

```
int L_IR_sensor_value = Read_L_IR();
```

Testing

To test code or check whether devices are working, we use the 'Debug' mode in the STM32 IDE. Here, we will learn how to test and debug our IR sensor code.

In app.c, above the App() function, create 4 int variables named **L_IR_sensor_value**, **LC_IR_sensor_value**, **RC_IR_sensor_value** and **R_IR_sensor_value**.

Eg. `int L_IR_sensor_value;`

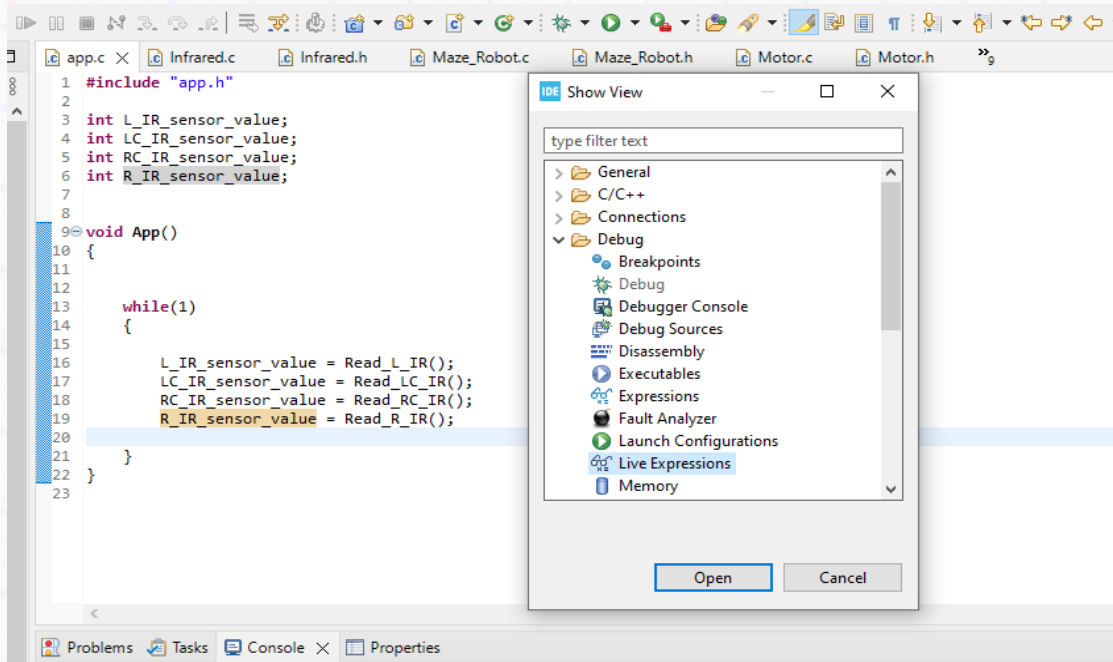
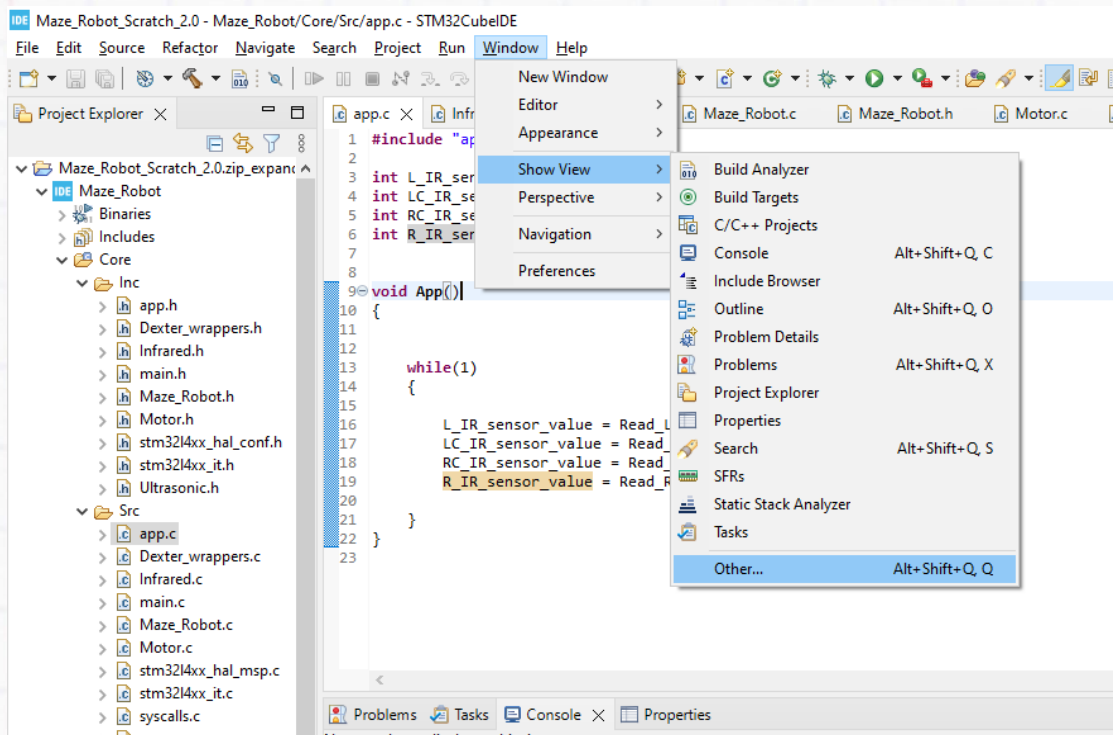
Now inside the App() function, create an infinite **while loop** like we have used in previous projects. Inside it, set the values of the variables created above to the values returned by the 'read IR' functions.

Eg. `L_IR_sensor_value = Read_L_IR ();`

Do similarly for the remaining 3 IR sensors.

```
app.c ×
1 #include "app.h"
2
3 int L_IR_sensor_value;
4 int LC_IR_sensor_value;
5 int RC_IR_sensor_value;
6 int R_IR_sensor_value;
7
8
9 void App()
10 {
11
12     while(1)
13     {
14         L_IR_sensor_value = Read_L_IR();
15         LC_IR_sensor_value = Read_LC_IR();
16         RC_IR_sensor_value = Read_RC_IR();
17         R_IR_sensor_value = Read_R_IR();
18     }
19
20 }
21
```

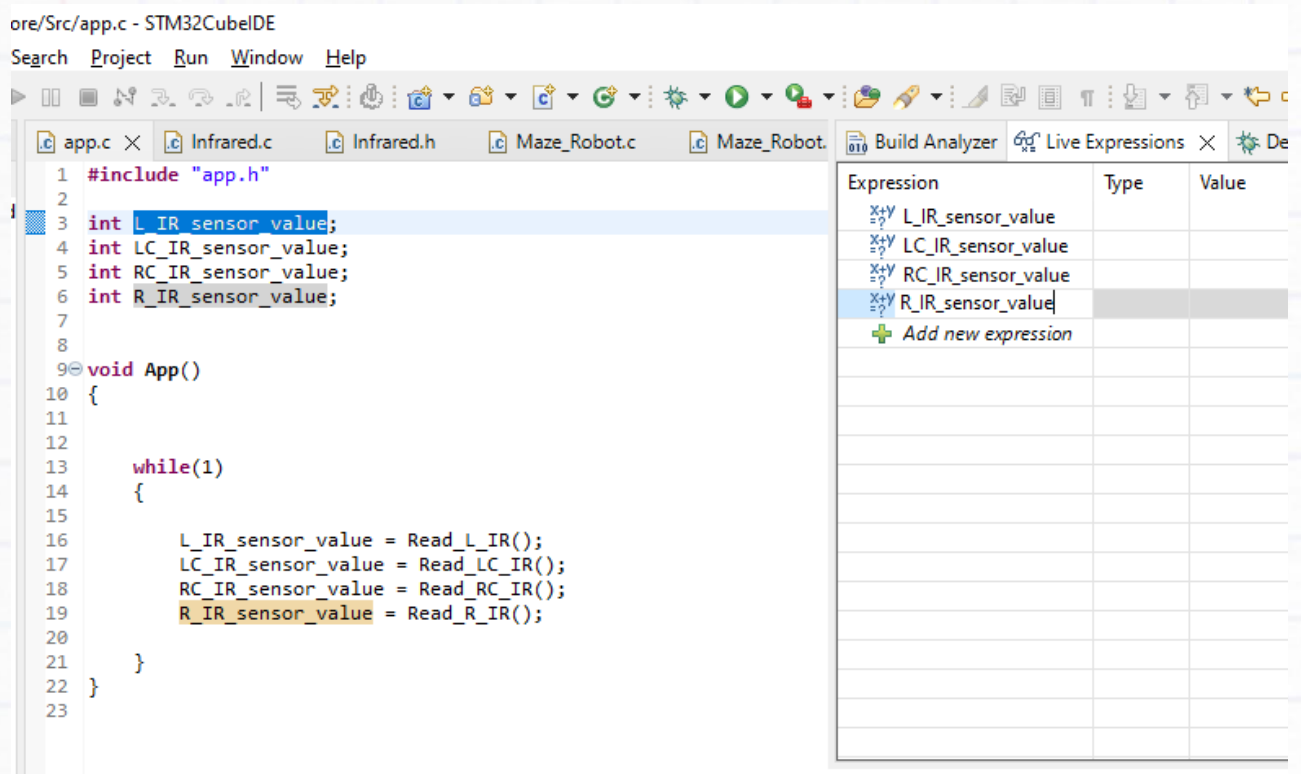
Click the Window tab on the top of the IDE and navigate to **Show View > Other > Debug > Live Expressions** and hit enter.



In the Live Expressions tab that opens, click *Add new expression* and paste the name of the variable **L_IR_sensor_value**. Do likewise for the remaining 3 variables.

ore/Src/app.c - STM32CubeIDE

Search Project Run Window Help

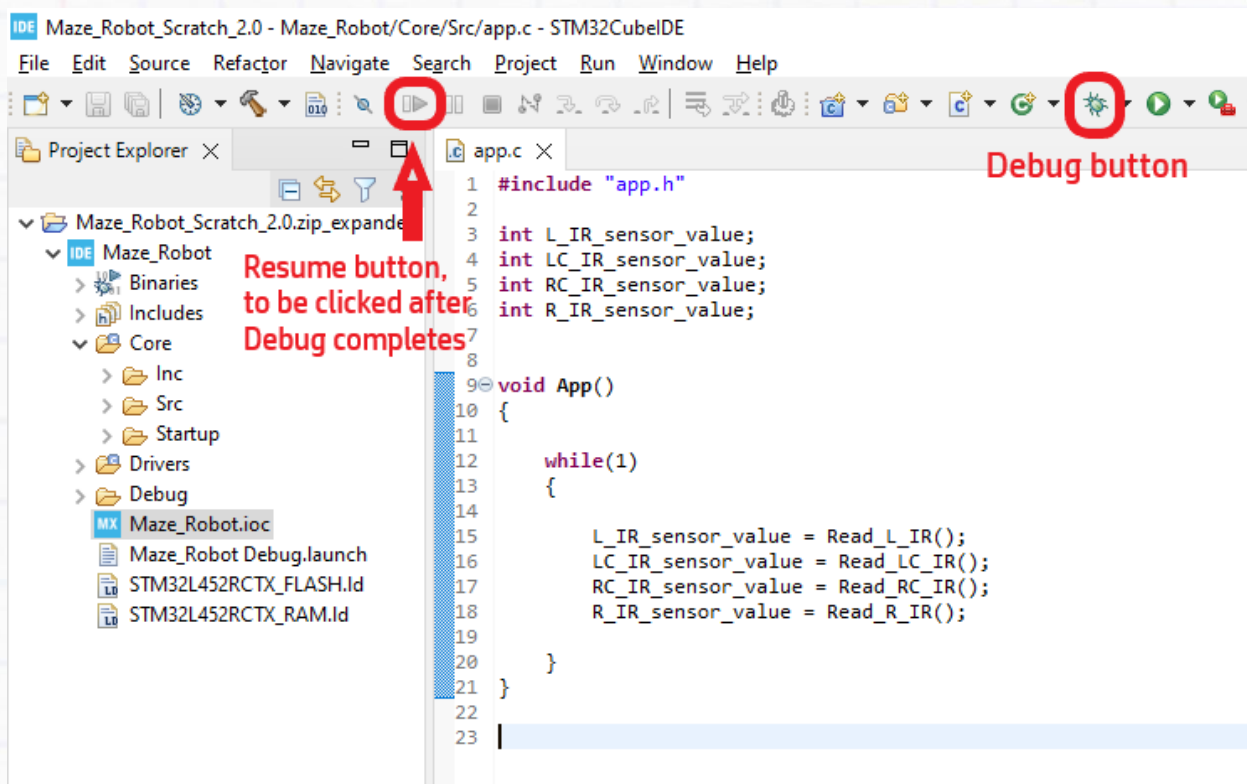


Expression	Type	Value
<code>L_IR_sensor_value</code>		
<code>LC_IR_sensor_value</code>		
<code>RC_IR_sensor_value</code>		
<code>R_IR_sensor_value</code>		
<code>+ Add new expression</code>		

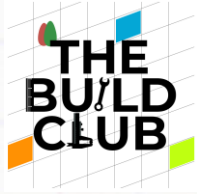
```

1 #include "app.h"
2
3 int L_IR_sensor_value;
4 int LC_IR_sensor_value;
5 int RC_IR_sensor_value;
6 int R_IR_sensor_value;
7
8
9 void App()
10 {
11
12
13     while(1)
14     {
15
16         L_IR_sensor_value = Read_L_IR();
17         LC_IR_sensor_value = Read_LC_IR();
18         RC_IR_sensor_value = Read_RC_IR();
19         R_IR_sensor_value = Read_R_IR();
20
21     }
22 }
23
  
```

Once done, hit the **Build** button on the IDE and ensure your code is error-free. Now, instead of clicking the Run button, we will be clicking the **Debug** button. After clicking Debug, if you receive a prompt asking to ‘Switch perspective’, click ‘Switch’.



Once the code has successfully uploaded to the Dexter, click the **Resume** button on the IDE to start Debugging. You will now be able to track in real-time the values of the IR sensors in the Live Expressions tab.



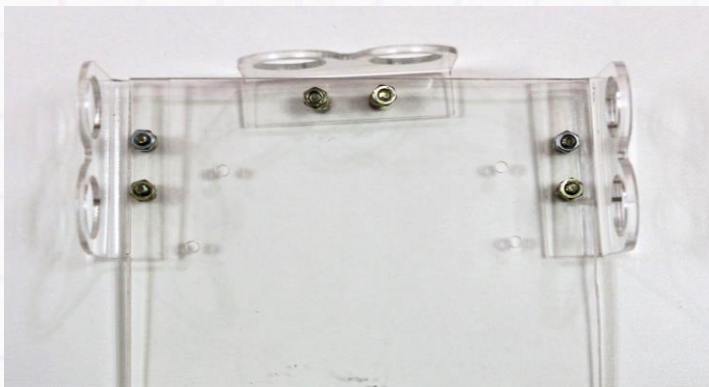
Tasks:

- 1) Put an object in front of any one of the sensors until the 2nd green LED on it turns ON. In the Live Expressions tab of the IDE, what value does the reading for that sensor show? Now remove the object - what is the value of that variable now? You will notice that when an object is detected, a '0' is shown and when removed, it changes to '1'. Test for all the four sensors and check if this same pattern is observed.
- 2) Take a white sheet of paper and see to what distance the sensor can detect it. Now stick a strip of non-reflective black tape or black paper in the centre of the sheet. Keeping an eye on the Live Expressions tab, hover this sheet from one end to the other. What did you notice when the black region passed over the sensor? Is the sensor able to sense it? The sensor doesn't detect the black colour because it is non-reflective. You may have to tune the sensor's potentiometer with a screwdriver to ensure this happens.
- 3) Gather small objects of different colours and reflectivity. Test whether the sensor is able to sense them and to what maximum distance.

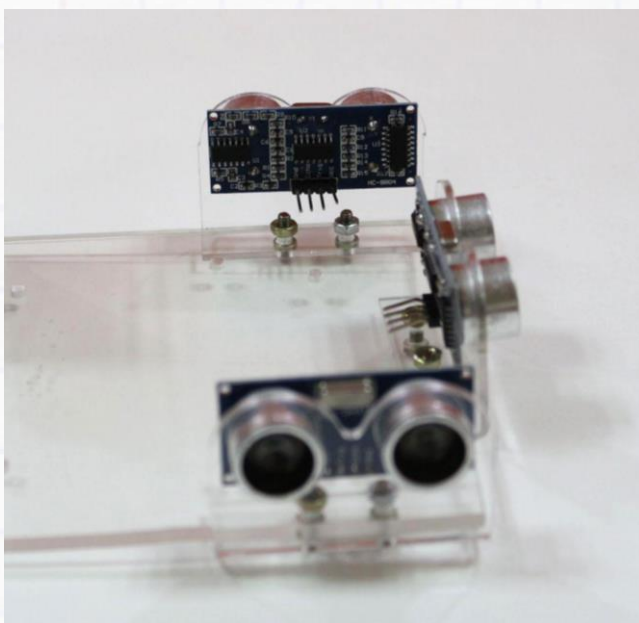
Ultrasonic Sensors

Assembly:

Take the acrylic cutouts for the Ultrasonic sensor. Using 6 sets of screws & nuts totally, fix the pieces to get the below shape:



Next, fit the 3 Ultrasonic sensors into the slots of the acrylic cutouts firmly.



Connections:

Take 16 **female-to-male** jumper wires and connect them to the pins (OUT, GND, VCC) of the 4 IR sensors. Connect the other ends as per the below connections:

Important

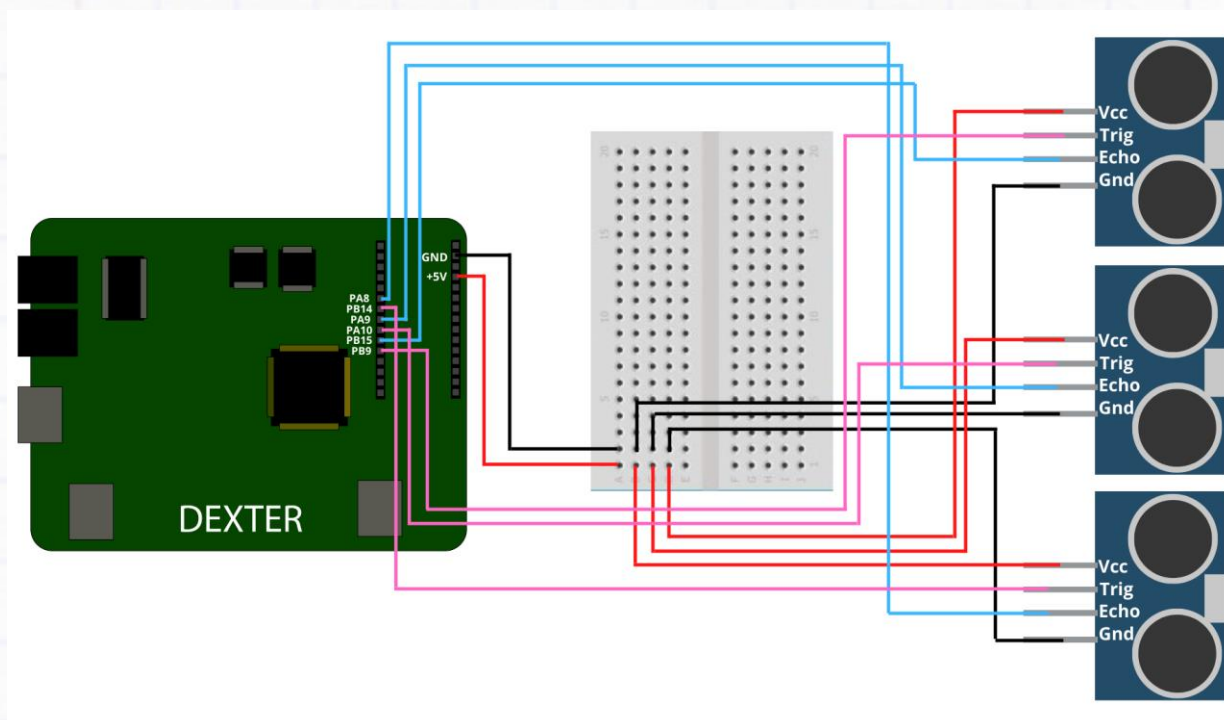


Sensor's Pin	Connect to...		
	Left sensor	Front sensor	Right sensor
GND	Breadboard Row L2	Breadboard Row L2	Breadboard Row L2
ECHO	PB15	PA9	PA8
TRIG	PB9	PA10	PB14
VCC	Breadboard Row L1	Breadboard Row L1	Breadboard Row L1

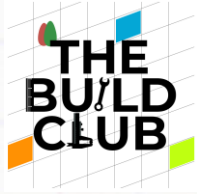
Take 2 male-to-male jumper wires and connect them as follows:

- **+5V** of Dexter to **Row L1** of breadboard
- **GND** of Dexter to **Row L2** of breadboard

Your final circuit should look like this:



Once done, connect the Dexter to your computer using the two USB cables.



Software & Testing:

Code

There are 3 code functions for the Ultrasonic sensors, one for each sensor. The functions return the distance in cm from an object as a decimal value.

```
Read_L_ultrasonic(); // Reads the Left Ultrasonic sensor
```

```
Read_F_ultrasonic(); // Reads the Front Ultrasonic sensor
```

```
Read_R_ultrasonic(); // Reads the Right Ultrasonic sensor
```

Eg. To store the value of Front Ultrasonic sensor, you can assign it to a float variable as shown:

```
float F_US_sensor_value = Read_F_ultrasonic();
```

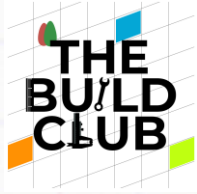
Testing

Like we did for the IR sensors, create 3 **float** variables named **L_US_sensor_value**, **F_US_sensor_value** and **R_US_sensor_value**.

Eg. float **F_US_sensor_value**;

Open the Live Expressions tab and add these variables there. Once again, create an infinite **while loop** inside the App() function and set the values of the variables created above to the values returned by the 'read ultrasonic' functions.

```
Eg. F_US_sensor_value = Read_F_ultrasonic();
```



Build your code and then hit **Debug**. The values you will now see in the Live Expressions tab are the distance in cm to the nearest objects/walls for each sensor.

NOTE: The sensors work accurately only when they are stationary and not shaking, so keep the setup on a stable flat surface.

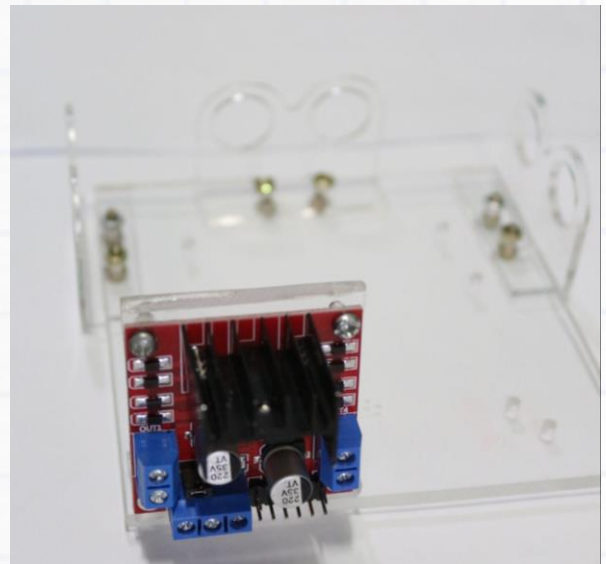
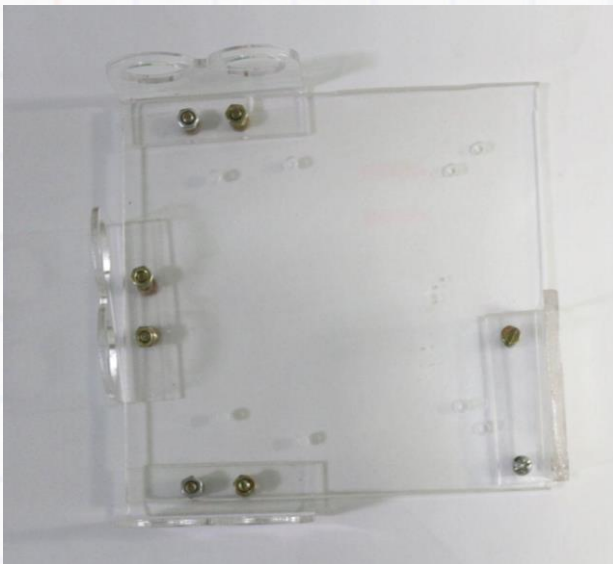
Tasks

- 1) Place a 30cm ruler on the floor perpendicular to a wall so that the 0cm mark touches the wall. Keep any one Ultrasonic sensor at the 10cm mark and verify if the reading in Live Expressions is also approximately 10cm. Take 3 readings for each sensor (say at 20cm, 30cm, 17cm) to verify that they are working properly.

If the distance readings for the sensors are not accurate, go to the top of the **app.c** file. In the Calibration Values section, you will find a variable called **ultrasonic_calibration** set to the value 53. Adjust this value until the readings are accurate (Note: You must test with different distances to ensure accuracy).
- 2) Keep one of the Ultrasonic sensors at the 20cm mark. Now slowly keep moving the sensor backwards. Note at what distance the readings lose their accuracy. This is the sensor's maximum range of functioning.

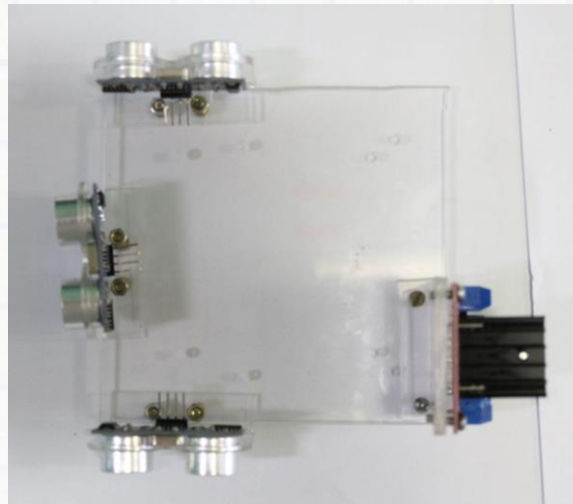
DC Motors

Assembly:

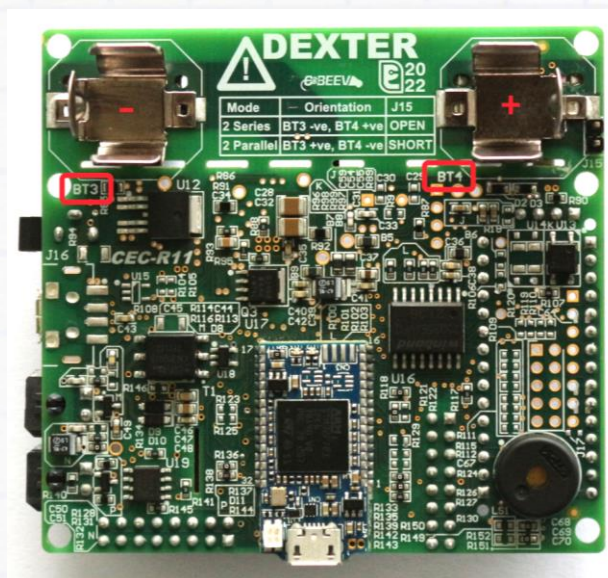
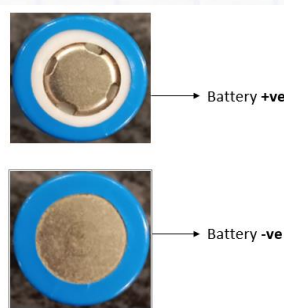


Take the acrylic piece for the L298N motor driver and fix it onto the acrylic sheet using screws. Ensure that the orientation of the piece is as shown in figure.

Screw the L298N Motor driver board onto it in the correct orientation using the 4 sets of screws & nuts. Fit the ultrasonic sensors back into their slots at the front.



From the components kit provided, take 1 NMC cell and insert it into the bottom battery slot of the Dexter board. The Negative terminal should be on the side of the 'BT3' label and the Positive terminal on the side of the 'BT4' label.



In the remaining 4 holes of the acrylic sheet, fit the 4 long screws and fasten them with the spacers and nuts on the other end. Peel the covering of the double-sided tape under the mini-breadboard and stick the breadboard between the Motor driver and screw mount at the rear. Now remove the nuts from the 4 long screws mounted on the acrylic sheet and fit them into the holes of the Dexter board.

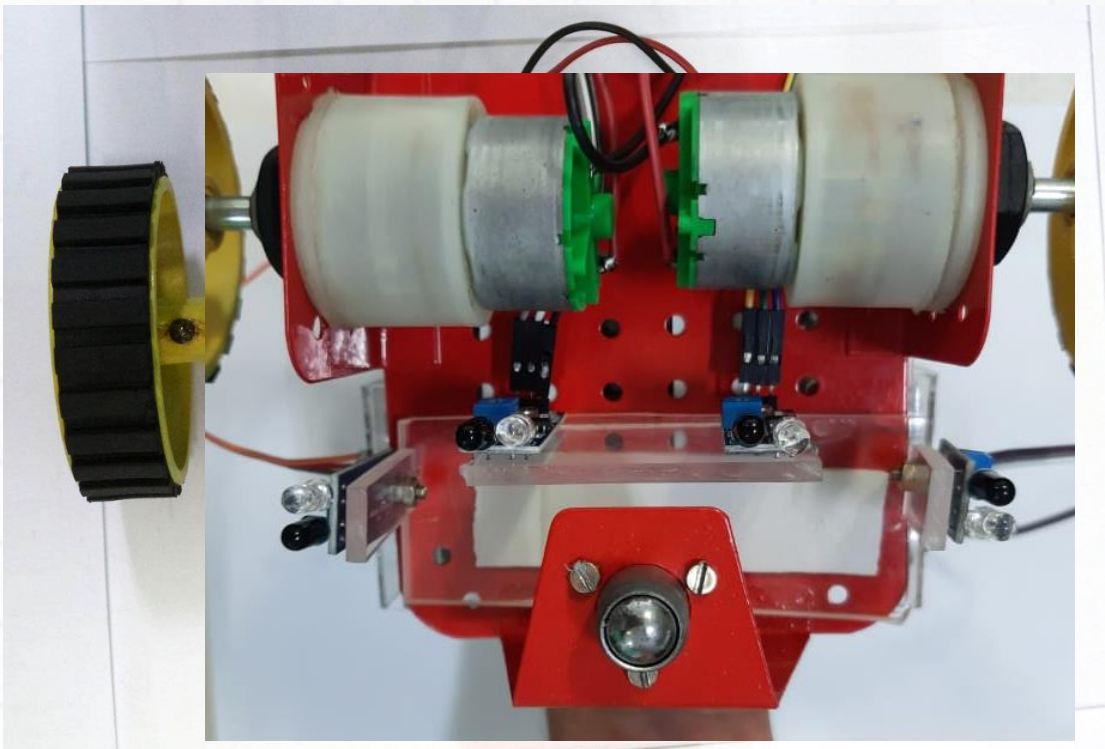
Retighten the nuts from the topside of the board, so that the Dexter is firmly fitted.



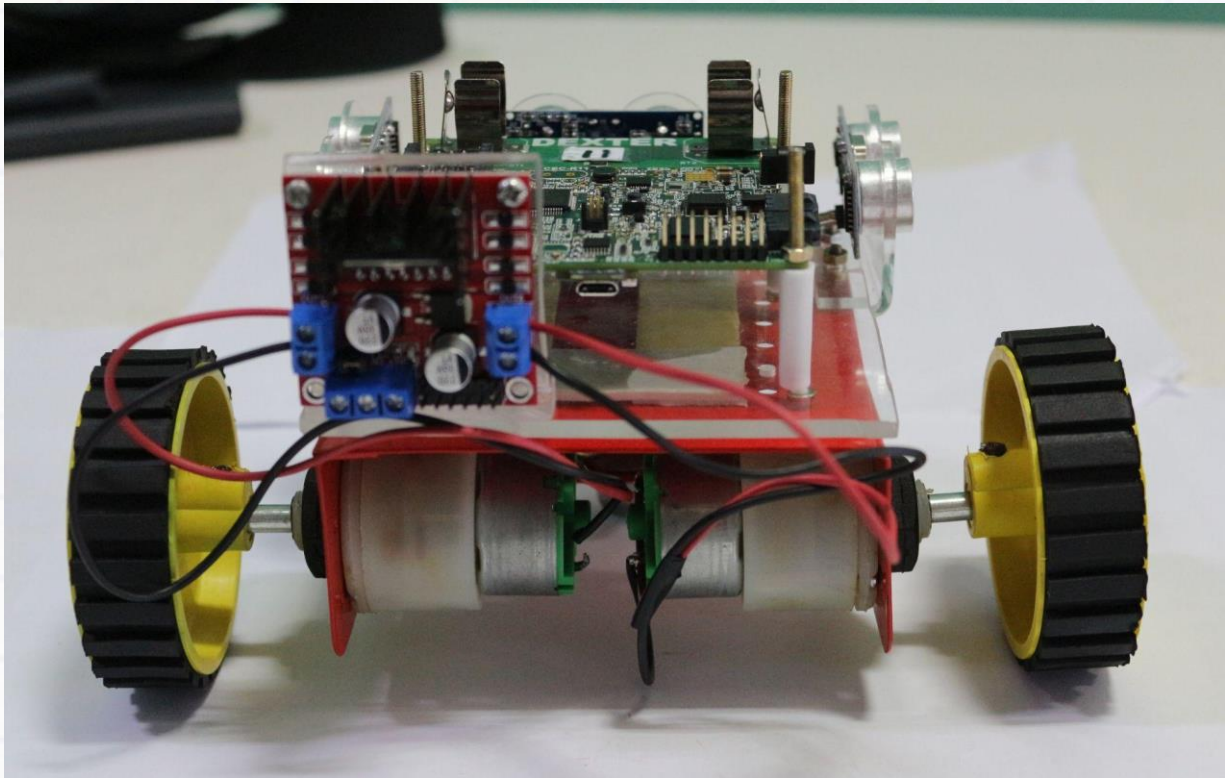
From the 2 DC Motors, remove the black plastic nuts, fit them through the slots in the metal chassis and refit the nut tightly on the other end. Insert the wheels into the ends of the motors and fix them by tightening the wheel's screw into the rotor hole. Make sure that the motors and wheels are aligned exactly straight.

Note that the Micro-mouse robot won't move accurately even if one of the motors / wheels is slightly tilted. Finally, fit the caster wheel into its slot in the front side and fix it to the chassis using 3 screws & nuts.

The IR sensor setup should be fixed using double-sided tape to the underside of the chassis, in front of the motors as shown.

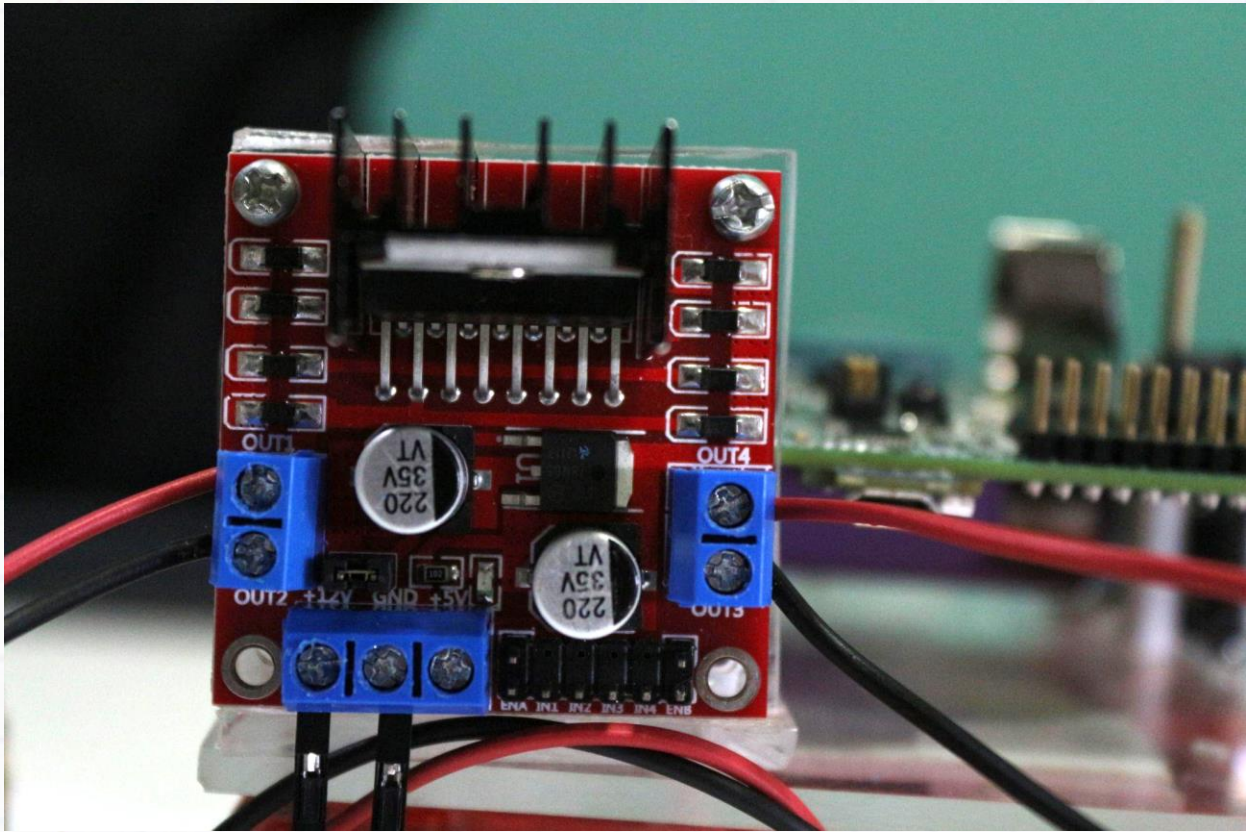


Fix the setup mounted on the acrylic sheet onto the metal chassis using multiple strips of double-sided tape. Make sure it is attached firmly and that the orientation of the chassis is correct.



Connections:

Connect the **red** wire of the **left** motor to **OUT1** and the **black** wire to **OUT2**, **black** wire of the **right** motor to **OUT3** and **red** wire to **OUT4**. Take 2 male-to-male jumpers and connect the +12V and GND terminals of the driver to VIN and GND pins of the Dexter. You will need a screwdriver to loosen the terminal screws, insert the wires/jumpers and re-tighten the screws.



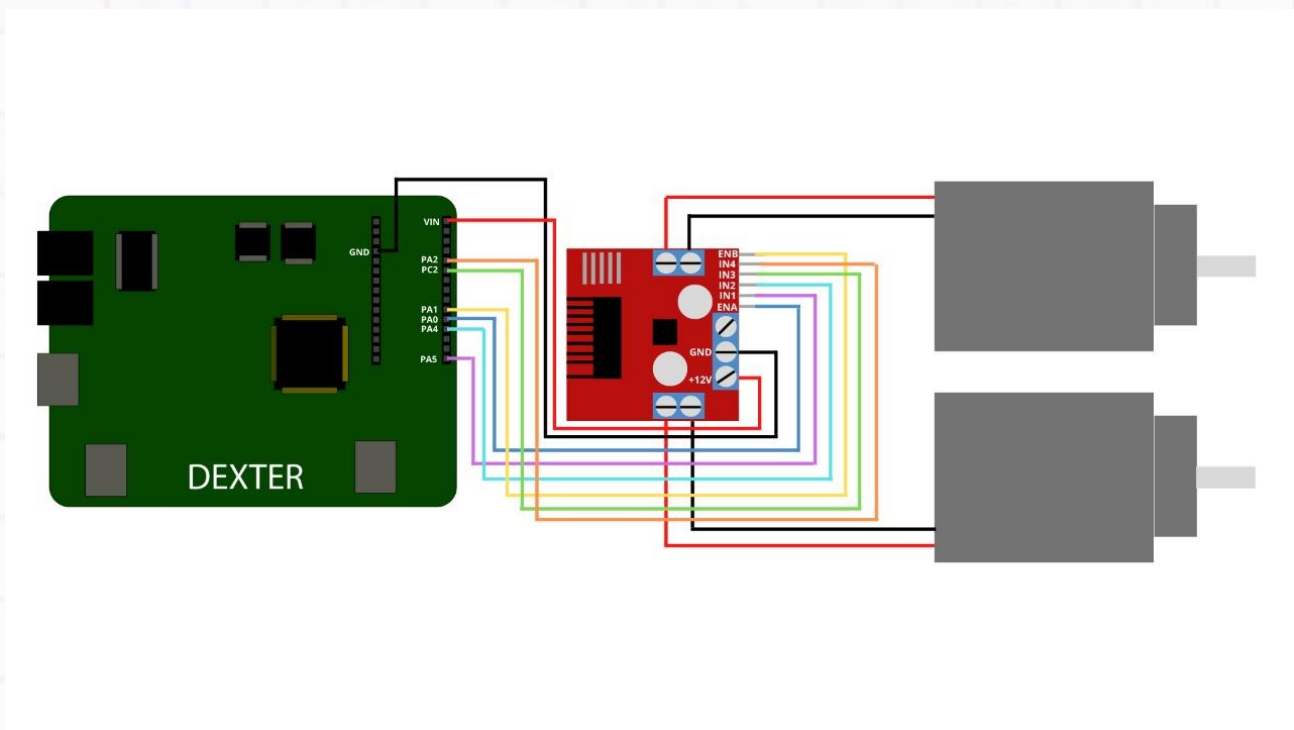
Take 6 **female-to-male** jumper wires and connect them as follows:

 **Important**

- **ENA** to **PA0** of Dexter
- **N1** to **PA5** of Dexter
- **IN2** to **PA4** of Dexter
- **IN3** to **PC2** of Dexter
- **IN4** to **PA2** of Dexter
- **ENB** to **PA1** of Dexter

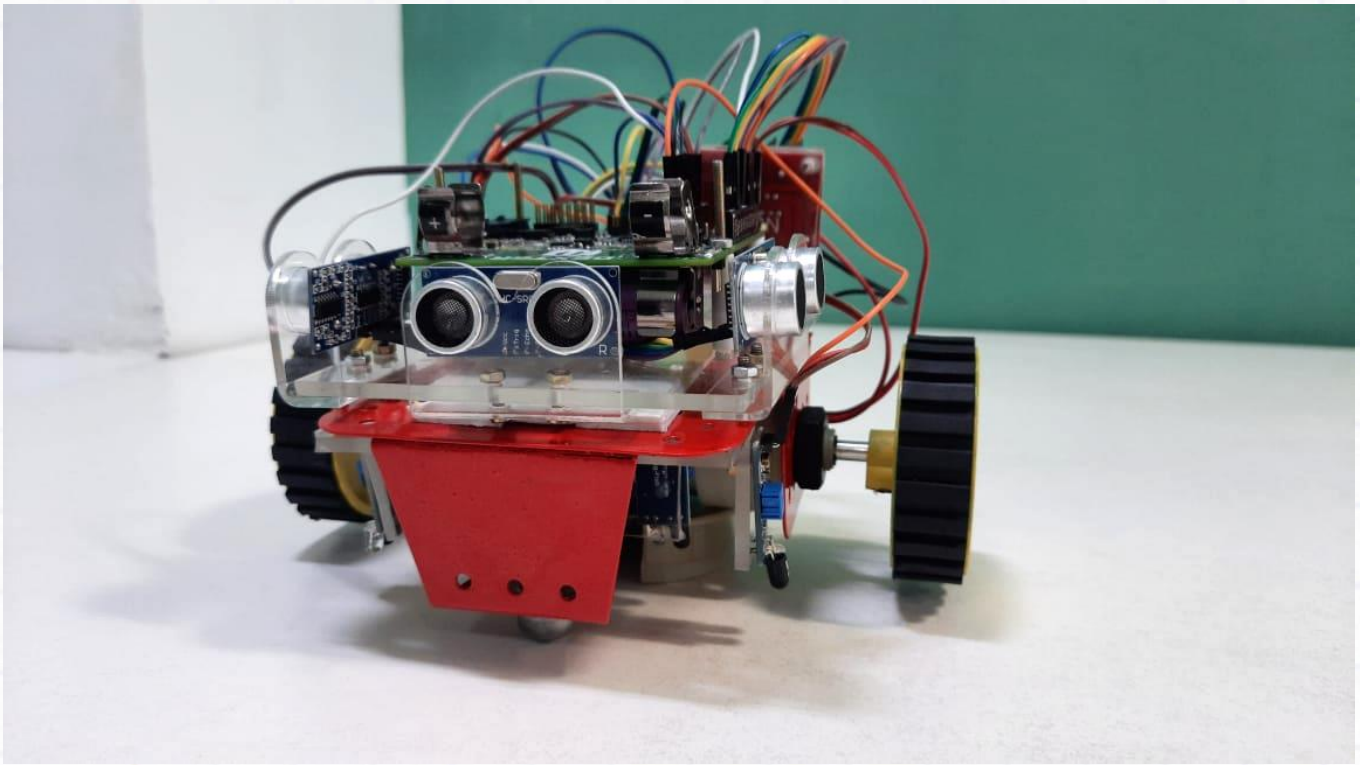
Use tape or cable ties to prevent the wires from interfering with the wheels or getting tangled.

Final circuit diagram for the DC Motors:



Complete the jumper connections for the DC motors, IR sensors and Ultrasonic sensors as done earlier. Use tape or cable ties to bunch the jumper wires and prevent them from dangling around.

WARNING: Make sure that each and every jumper connection is correct, most importantly for the VCC and GND connections. If even one is incorrect, there are high chances that the components will get damaged.

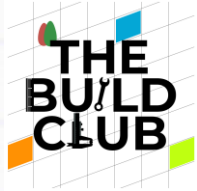


The Micro-mouse robot assembly is now complete!

Software & Testing:

From now on, follow this procedure to test or run any code on the Micro-mouse Robot:

- 1) Take a small flat object like a plastic/cardboard box on which you can safely place your Micro-mouse such that it is rested safely but its wheels don't touch the ground.
- 2) Connect the Dexter to your computer using the two USB cables. Run or debug the code you want to run onto the Robot.



- 3) Once the robot is working as desired, remove the 2 USB cables. Insert the 2nd NMC cell into the top side battery slot of the Dexter board. Its Negative terminal should be connected to the side having **-ve** label (BT1) and Positive to **+ve** (BT2). **BE VERY CAREFUL WITH THIS.** Wrong orientation can cause permanent damage to the board and components.

The Micro-mouse can now be placed on the floor to move.

NOTE: It is possible that the cells are low on charge and this will affect the Micro-mouse's performance. To charge the NMC cells, follow these steps:

How to charge the NMC cells

- 1) Remove the top NMC cell from the Dexter. The bottom one can remain mounted.
- 2) In the IDE, remove or comment out any code inside the App() function.
- 3) Connect the Dexter to the computer using the 2 USB cables and Run the blank code. This will prevent any connected actuator or sensor from draining power.
- 4) Now insert the 2nd NMC cell back into the top slot. A red LED on the Dexter will now start glowing, indicating that the cells are charging. The cells together will take around <Insert> hrs to charge from empty and once fully charged the LED indicator will <Insert>. The USB cable for uploading code can now be removed but the power cable is necessary to charge the cell. The computer must also remain ON to charge the cells.

Code functions

Below are the functions to control the DC motors of the Micro-mouse.

1) **Motor_init(motor);**

This function Initialises the motor specified. It must be called compulsorily for both motors, for them to work.

- **motor** input can have 2 values, for each motor - **Motor_L** or **Motor_R**

Eg. To initialise the Left DC motor: **Motor_init(Motor_L);**

2) **Motor_set_dir(motor, dir);**

This function sets the direction of rotation of the motor specified

- **dir** can have 2 values - **forward** or **backward**

Eg. To set direction of Right DC motor to forward:

Motor_set_dir(Motor_R, forward);

3) **Motor_set_speed(motor, speed);**

This function sets the speed of rotation of the motor specified

- **speed** can have values ranging from 0 to 100

Eg. To set speed of Right DC motor to Forward:

Motor_set_speed(Motor_R, 100);

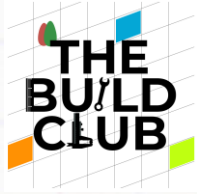
4) **Motor_run(motor, dir, speed);**

This function runs the motor with specified direction and speed.

Eg. To run Left DC motor backward at speed=50:

Motor_run(Motor_L, backward, 50);

5) **Motor_stop(motor);**



This function stops the specified motor.

Eg. To stop the Left DC motor: **Motor_stop(Motor_L);**

6) **Move_forward(speed);**

This function moves the Micro-mouse forward at the specified speed.

- **speed** can have values ranging from 0 to 100

Eg. To move Micro-mouse forward at speed = 100:

Move_forward(100);

7) **Move_backward(speed);**

This function moves the Micro-mouse backward at the specified speed.

- **speed** can have values ranging from 0 to 100

Eg. To move Micro-mouse backward at speed = 80:

Move_backward(80);

8) **Move(left motor dir, left motor speed, right motor dir, right motor speed);**

This function moves the Micro-mouse with direction and speed specified for each motor

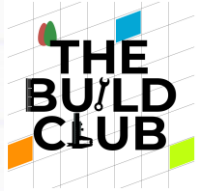
- **dir** can have 2 values - **forward** or **backward**
- **speed** can have values ranging from 0 to 100

Eg. To move Micro-mouse swerving slightly to the left:

Move(forward, 80, forward, 100);

9) **Turn_left_degrees(degrees);**

This function makes the Micro-mouse turn Left for specified degrees.



- **degrees** can have any positive degree value

Eg. To turn 90 degrees Left:

```
Turn_left_degrees(90);
```

10) Turn_right_degrees(degrees);

This function makes the Micro-mouse turn Right for specified degrees.

- **degrees** can have any positive degree value

Eg. To turn 90 degrees Right:

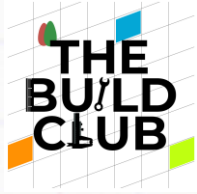
```
Turn_right_degrees(90);
```

Tasks

- 1) Initialise the Left DC Motor. Make it move forward at a speed of 100. Now reverse its direction and set its speed to 80. Repeat for Right DC Motor too.

In case you notice that a wheel is rotating in the opposite direction than expected, swap the wires connected to the **OUT** terminals of the L298N motor driver board.

- 2) Write a code to make the Micro-mouse move forward for 5 seconds, stop for 2 seconds and then move backward for 5 seconds.
- 3) Make the Micro-mouse turn 90 left. You may notice that it doesn't exactly turn 90. To calibrate this, go to the top of the **app.c** file. You will find a variable called **turn_calibration**, tune its value until the accuracy of turning is accurate.
- 4) Write a code to make the Micro-mouse travel in the following shapes:
 - a) Square
 - b) Triangle
 - c) S shape



- d) Circle of radius:
- i) 30cm
 - ii) 50cm

You have now mastered how to drive the Micro-mouse!

Part 2: Build a Maze Solving Robot

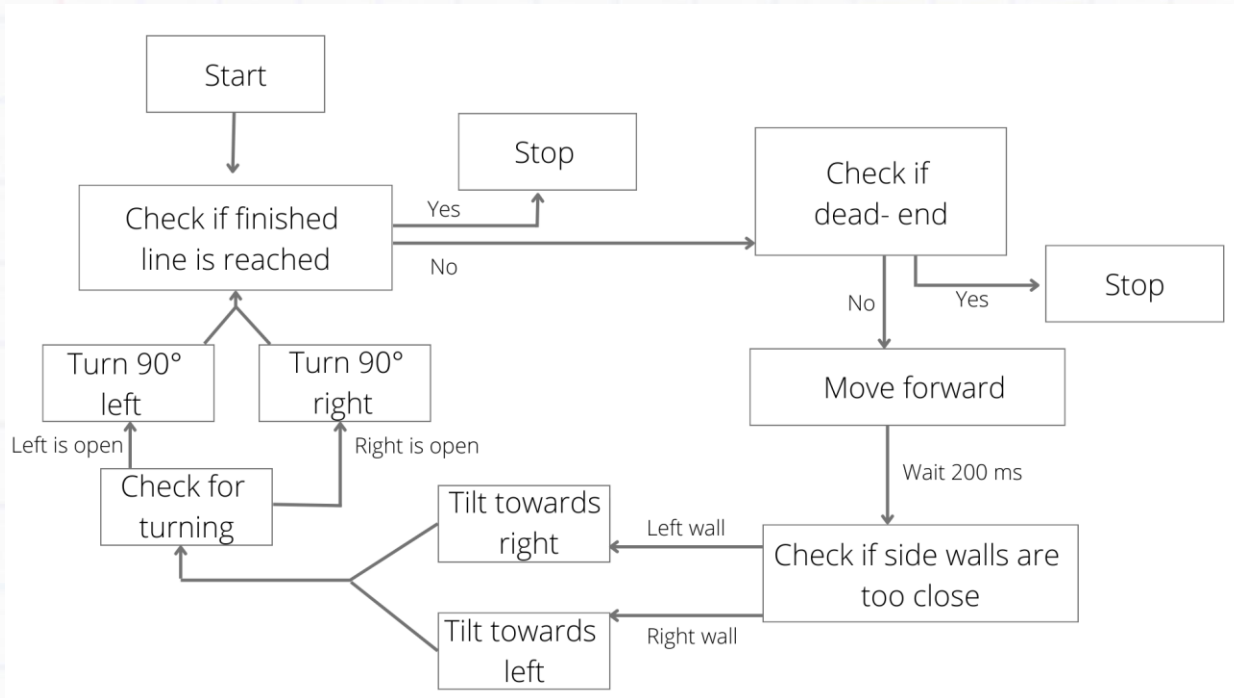
In this part, we will build a program to make our Micro-mouse navigate and solve a maze.

To write any complex program, the following procedure is helpful:

- 1) Write or draw out an algorithm / flowchart of steps that is required to accomplish the task.
- 2) For each small step in the algorithm, write a code that executes that step.
- 3) Translate the algorithm / flowchart into code by replacing each step with the code that accomplishes it.
- 4) Run the code. There will no doubt be some errors that you will face that you did not expect. Locate the cause for each and fix them one by one. To pinpoint the cause, examine across hardware, software and also the logic you have implemented.

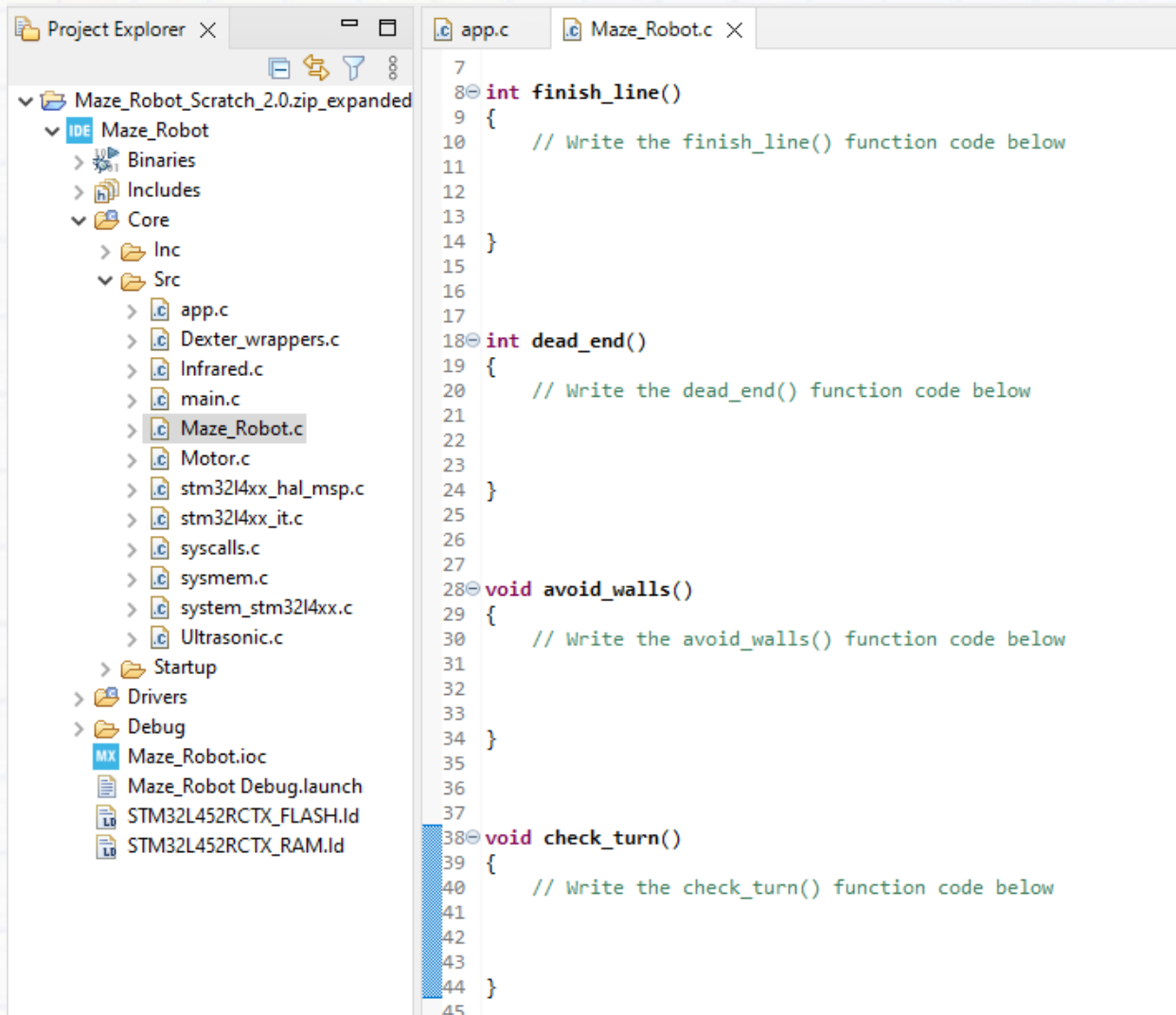
Flowchart of the Program

This flowchart diagram captures the flow of logical steps needed to implement the Maze solving robot project. Once this is firmly understood, code can be easily written to make the Micro-mouse follow these instructions.



Encoding Each Step

For the functions defined in the following pages, go to the Maze_Robot.c file in the IDE and complete the skeleton of the functions inside.



```

7
8 int finish_line()
9 {
10     // Write the finish_line() function code below
11
12
13
14 }
15
16
17
18 int dead_end()
19 {
20     // Write the dead_end() function code below
21
22
23
24 }
25
26
27
28 void avoid_walls()
29 {
30     // Write the avoid_walls() function code below
31
32
33
34 }
35
36
37
38 void check_turn()
39 {
40     // Write the check_turn() function code below
41
42
43
44 }
45

```

The following pages explain how to write each code command in detail.

a) Start

Initialise the 2 DC Motors using the Motor_init() function. This only needs to be done once, so it will be outside the while loop, in which the rest of the program will be inside.

```
Motor_init(Motor_L);  
Motor_init(Motor_R);
```

b) Check if Finish Line is reached

The Maze's finish line is a wide strip of black tape. When the Micro-mouse passes over it, all 4 IR sensors will detect 'black', ie. an absence of reflection. We want our program to run only as long as the finish line is not reached. Hence we will use a While loop, whose entry condition should be: 'finish line not detected'.

```
int finish_line()  
{  
    if ((Read_L_IR()==1) && (Read_LC_IR()==1) && (Read_RC_IR()==1) && (Read_R_IR()==1))  
    {  
        return 1;  
    }  
    else  
    {  
        return 0;  
    }  
}
```

```
while(!finish_line())  
{  
    // Remaining program  
}
```

c) Stop

Stop both the DC Motors.


```
Motor_stop(Motor_L);  
Motor_stop(Motor_R);
```

d) Check if dead-end

In case of a dead-end, all 3 Ultrasonic sensors - L, F, R - would sense a wall at a close distance. If a dead-end is detected, the Micro-mouse would need to halt immediately.

```
int dead_end()  
{  
    return ((Read_F_ultrasonic() < 12) && (Read_L_ultrasonic() < 12) && (Read_R_ultrasonic() < 12));  
}
```

```
if(!dead_end())  
{  
    // Remaining code  
}
```

e) Move Forward

If the coast is clear, move forward temporarily for 200ms.

```
Move_forward(100);  
Delay_ms(200);
```

f) Check if side walls are too close

This step uses the Ultrasonic sensors to check if the distance of the Micro-mouse from the walls of the maze has gone below a critical value. There are 2 scenarios:

- i) If the Micro-mouse comes less than 12cm close to a wall it must move away - towards the opposite direction.
- ii) If the Micro-mouse is in a corridor, with walls on either side, it must adjust its position to the centre of the lane so that it is equidistant from both walls.

```
void avoid_walls()
{
    if(Read_F_ultrasonic() < 12)
    {
        Motor_stop(Motor_L);
        Motor_stop(Motor_R);
    }

    else if(dead_end())
    {
        Motor_stop(Motor_L);
        Motor_stop(Motor_R);
    }

    else if(Read_L_ultrasonic() < 10)
    {
        Move(forward, 100, forward, 30);
    }

    else if(Read_R_ultrasonic() < 10)
    {
        Move(forward, 30, forward, 100);
    }

    else if(Read_L_ultrasonic() < Read_R_ultrasonic())
    {
        Move(forward, 100, forward, 30);
    }

    else if(Read_R_ultrasonic() < Read_L_ultrasonic())
    {
        Move(forward, 30, forward, 100);
    }
}
```

g) Check for Turnings:

A turning would come when there is a big opening on one side. This will be detected when the Ultrasonic sensor reading for that side exceeds a certain

value. In such a case, the Micro-mouse would have to turn 90 towards that side.

```
void check_turn()
{
    if((Read_L_ultrasonic() > 25) && (Read_R_ultrasonic() < 16) && (Read_F_ultrasonic() < 16))
    {
        Turn_left_degrees(90);
    }

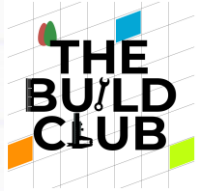
    else if((Read_R_ultrasonic() > 25) && (Read_L_ultrasonic() < 16) && (Read_F_ultrasonic() < 16))
    {
        Turn_right_degrees(90);
    }
}
```

With this, one cycle of the program is completed. This cycle would have to constantly repeat until the finish line is detected.

Algorithm of the final code

- 1) Initialise motors
- 2) While the finish line isn't reached, keep doing the following:
 - If not a dead-end, then:
 - Move forward, followed by a small delay
 - Stay away from side walls
 - Check for turns

Implementation



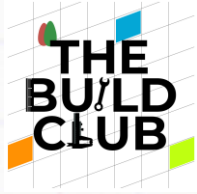
- 1) With the help of code taught before for each step, implement the above algorithm inside the **App()** function using the functions you have completed in the **Maze_Robot.c** file.
- 2) Using sheets of thermocol fixed together using double-sided tape, construct a maze for your Micro-mouse. Stick strips of black tape or black paper across the finish line of the maze, one after the other to form a black region of sufficient . This will serve as the finish line. Make sure that it is non-reflective so that the IR sensors are able to differentiate it from other colours.
- 3) Since the Maze setup dimensions will vary, you will have to adjust the values of wall distances for checking dead-ends, wall avoidance and accurate turnings, as per your Maze setup. Additionally, ensure turning calibration and ultrasonic sensor calibration is accurate.

NOTE: When testing the Micro-mouse on the maze, first place it at the start of the maze without the top NMC cell inserted. Once ready to begin, insert the cell and it should start moving.

Hooray!

You have completed the Maze Solving Robot build!

Challenges



1) Build the fastest and most efficient Maze Solver!

Write programs for implementing various popular maze solving algorithms such as:

- a) Left-hand following
- b) Right-hand following

Explore more such algorithms on the internet. Eg. [Wikipedia](#)
Note down how many attempts it takes to solve the maze for each algorithm. Which one was the best?

2) Build a Maze Learning Robot! Devise a program such that the Micro-mouse is made to repeatedly attempt the maze until it is solved. In the process of traversing the maze, the robot should learn the maze and map it in its memory. Therefore, once fully mapped, the Micro-mouse should be able to solve the maze without making any mistake in a single attempt.

3) Build a Line Following Robot. On a white surface, Draw a path using black tape. Using only the centre 2 IR sensors to detect the black line, the Micro-mouse should be able to follow it. Note: The black tape width must be less than the gap between the centre 2 IR sensors. Don't create abrupt turns in the black line, else the robot may not detect it.