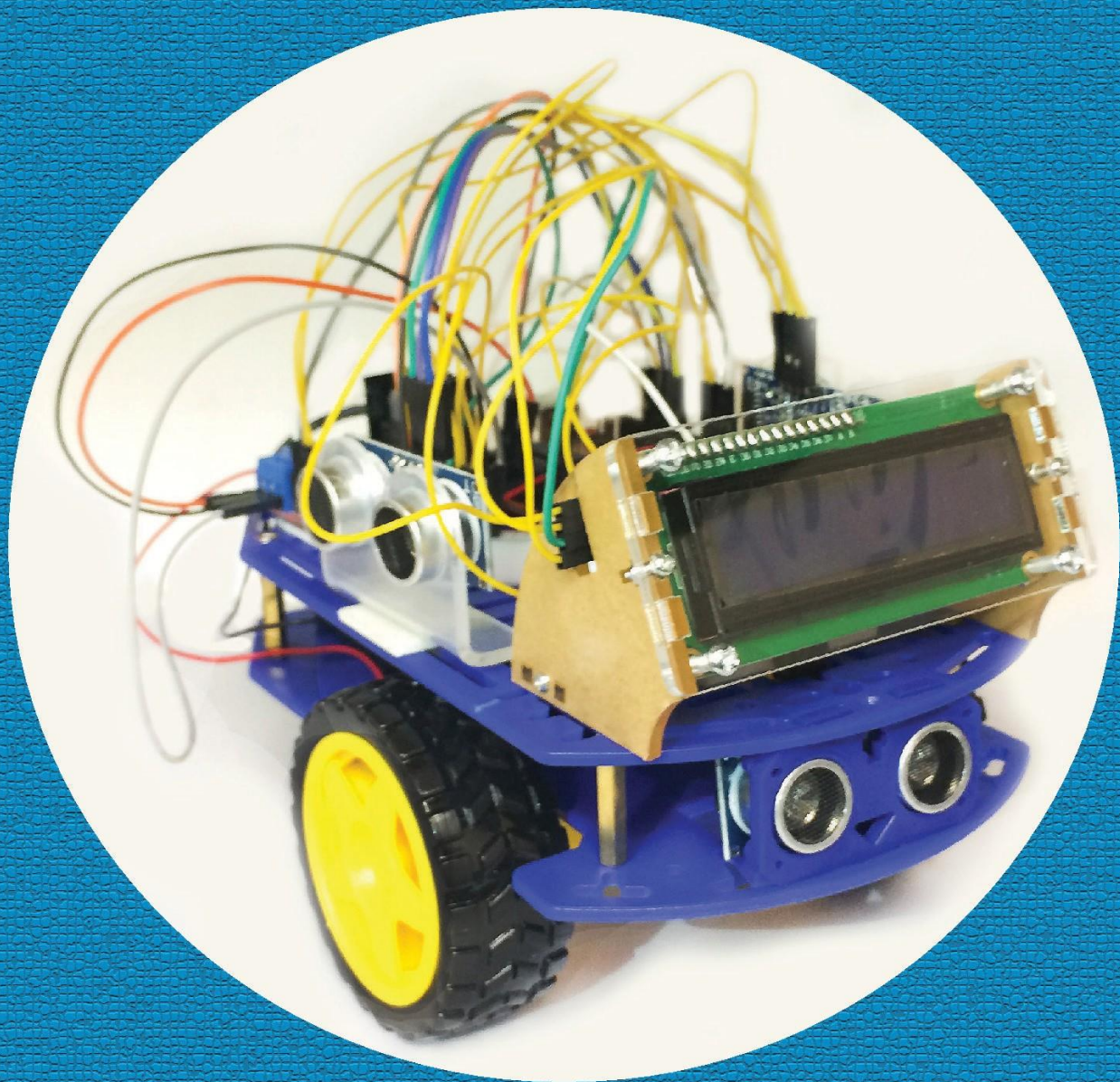


Maze Solving Smart Car



AK-CAR-05

Table of Contents

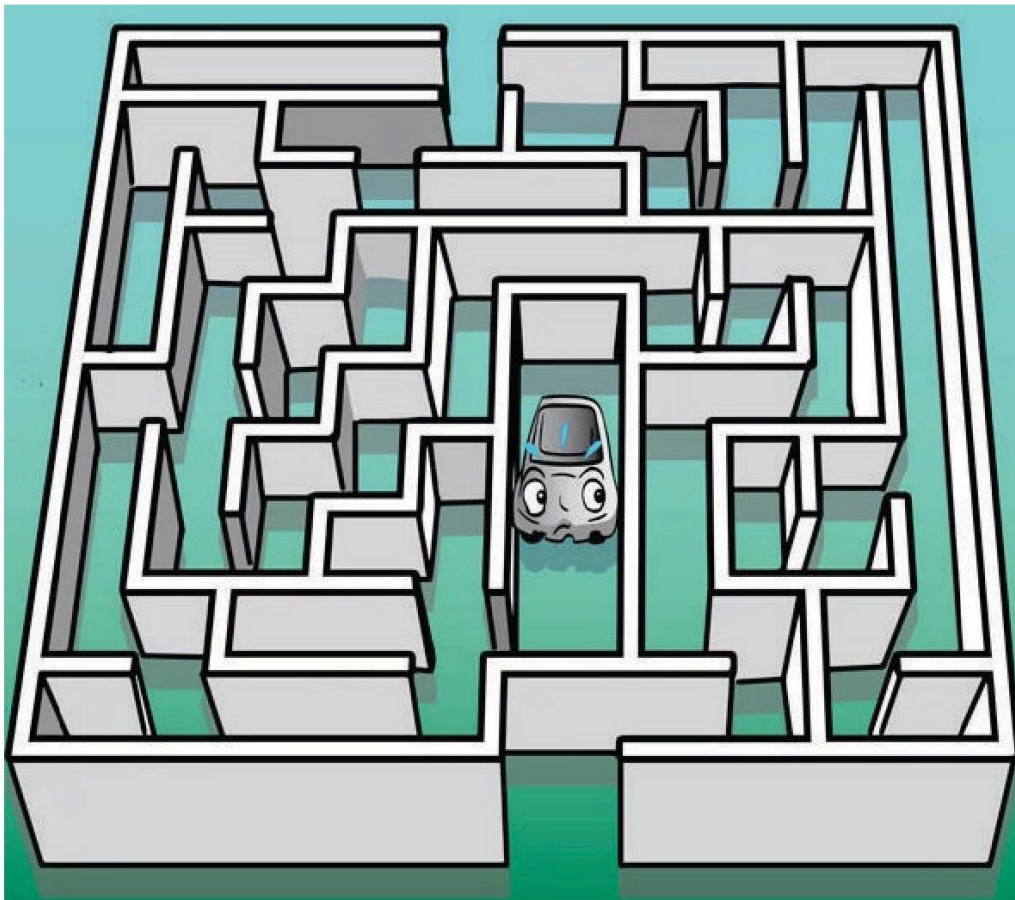
Introduction	3
What Is A Maze Solving Robot?	3
List of Parts.....	4
About The Components.....	5
ABRAUNO	5
L298N Motor Driver	5
Ultrasonic Sensor	6
LCD Display with I2C backpack - 16x2.....	7
Chassis & Motors Kit	7
Blocks For Maze Solving Robot.....	8
CIRCUIT DIAGRAM.....	8
MAZE SOLVING ROBOT ASSEMBLY	9
FRAME CONSTRUCTION.....	9
ASSEMBLY OF ELECTRONICS	13
Connections.....	14
Coding	15
Software Required	15
Algorithms Used	15
Code	17
Complete Code.....	18
Strategies For Constructing Arena.....	25
What Else We Can Do With This Kit?	26
Obstacle Avoiding Robot	26
Wall Follower Robot.....	26
Object Distance Calculation Robot	26

Introduction

The field of robotics has advanced greatly in recent years, having a large field of research in finding solutions to every problem, such as problems solving mazes and test new designs. The main objective of this kit is to make students get familiar with the principles, the algorithm used in solving the complex mazes and making autonomous robot solving without human interference. After completing the robot using ABRA MAZE SOLVING KIT, students will increase their qualitative and quantitative analytical thinking and will be able to solve any puzzles.

What Is A Maze Solving Robot?

Maze is a network of paths, typically from an entrance to exit. The concept of maze was invented approximately a thousand years ago in Egypt. From then, many mathematicians made various algorithm to solve maze. This maze solving robot move in the complex puzzled path made by walls and reach the end on time. We can also program to return it in the shortest path by using flooding algorithm. There are many different types of maze solving robot depend on the algorithm used. To solve the maze, this robot will apply wall following algorithms such as left- or right-hand rule which is explained clearly in the algorithm section.



List of Parts

ABRA Part #	Description	QTY	Checkbox
ABRAUNO	Arduino Uno R3 Compatible Microcontroller with Cable	1	
MOT-L298	L298N Dual H Bridge DC Controller Board	1	
HC-SR04	Ultrasonic Sensor Ranging Module	3	
SENS-BR1	Ultrasonic Sensor Acrylic Mounting Bracket	2	
SG92R	Micro servo 9G 2.5KG Torque	1	
SENS-BR-C	Sweeping rangefinder mounting brackets – clear acrylic	1	
DG007	Magician Chassis (Includes 2x MOT-305 DC Motors)	1	
LCD-MOD-13	16x2 LCD Display with I2C backpack - White on Blue	1	
LCD-MOD-C	D.I.Y - 1602 LCD Display Acrylic Holder Enclosure	1	
30-448	AA Size Panasonic Heavy Duty Battery Pkg/4	1	
826-ADA	Premium Female/Male 'Extension' Jumper Wires - 40 x 6"	1	
MISC	Premium Male/Male 'Extension' Jumper Wires	5	
TAPE-DSF-1CM-W	1cm Wide - Double Sided Foam Tape (white)	1	
ABRA-6	BreadBoards 400 Tie Points	1	
TOOL-200	Philips Screwdriver - #0	1	

Did you open the box? Please check the parts you received.



Tools Required: (Not Included)



Soldering Iron (SI-3090)



Scissor



Wire Cutter (HT-109P)



Nylon Zip Warps (CT-130-SC)



Tape

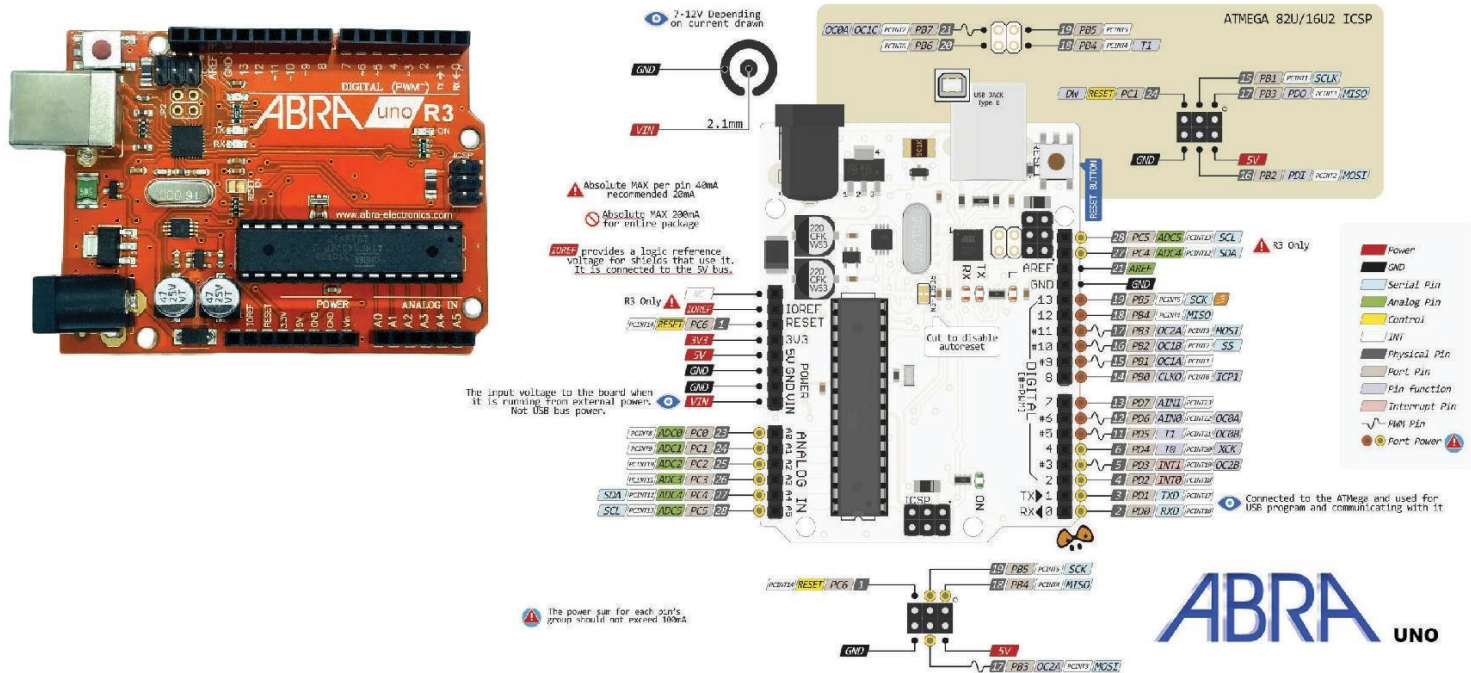


**Heatsink
(Optional)**

About The Components

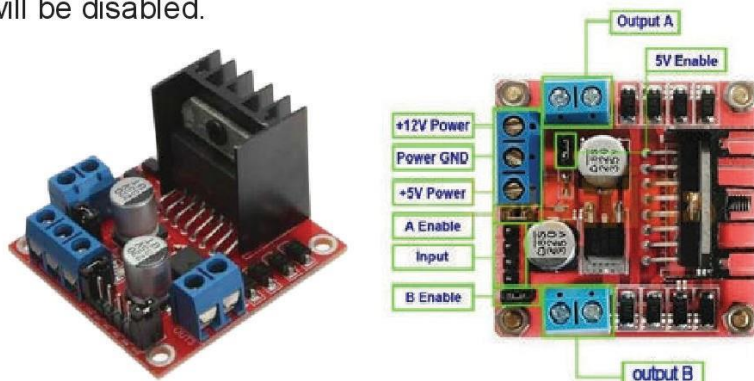
ABRAUNO

It is an Arduino UNO R3 compatible microcontroller which can be programmed using Arduino IDE. It is a microcontroller board based on the ATmega 328P chip. With the help of this board, you can program and give instructions to the output devices. The ABRAUNO pinout consists of 14 digital pins, 6 analog pins, a power jack, USB connection and ICSP header. The versatility of the pinout provides many different options such as driving motors, LEDs, reading sensors and more. Regarding the power supply, ABRAUNO can be powered three ways, Barrel jack, VIN pin and USB cable. We can use Barrel jack and VIN to power the board up to 12VDC supply and USB port up to 5V at 500mA. The board includes regulator which converts the voltage to required 5V for the microcontroller.



L298N Motor Driver

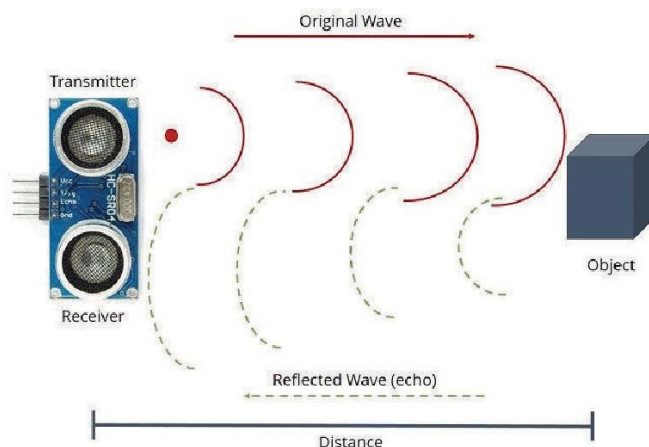
The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time. This module can drive DC motors that have voltages between 5 and 35V with a peak current up to 2A. This module is also responsible for making higher voltage motors compact with 5V microcontroller instructions. This module has two screw terminal blocks for the motor A and B, and another screw terminal block for the ground pin, the VCC for motor and a 5V pin which can either be an input or output. The module has an onboard 5V regulator which is either enabled or disabled using a jumper. The main disadvantage of the module is, the voltage drop across the IC is 2V and the output voltage of the motor will not be same as the input voltage. The enable A and enable B pins on the board is used for controlling the speed of the motor. If the jumper is present on the pin, that means that motor will be enabled and work at maximum speed. If you remove the jumper, you can connect a PWM input and control the speed of the motor. If you connect these pins to ground, the motor will be disabled.



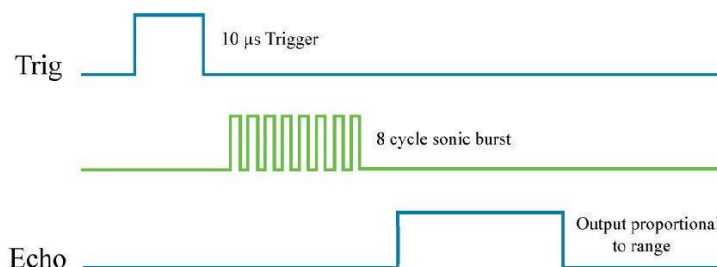
Ultrasonic Sensor

An ultrasonic sensor is a device that can measure the distance to an object by using sound wave at a specific frequency and listening for that sound wave to bounce back. By recording the elapsed time between the sound wave being generated and the sound wave bouncing back, we can identify the object and its distance from the sensor.

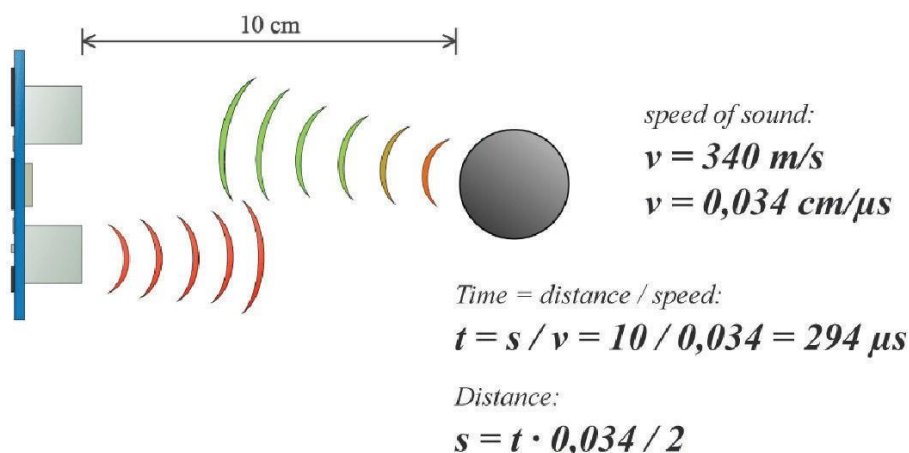
Working principle: It emits an ultrasound at 400000Hz which travels through the air and if there is an object or obstacle on its path, it will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.



The HC-SR04 Ultrasonic module has 4 pins, ground, VCC, Trig and Echo. The ground and the VCC pins of the module needs to be connected to the ground and the 5 volts pins on the ABRAUNO board respectively and the trig and echo pins to any digital I/O on the ABRAUNO board. In order to generate the ultrasound, you need to set the trig on a high state for 10 μ s. That will send out an 8-cycle sonic burst which will travel at the speed of sound and it will be received in the Echo pin. The Echo pin output the time (in microseconds) the sound wave travelled



For example, if the object is 10cm away from the sensor, and the speed of the sound is 340m/s or 0.034 cm/ μ s. thus, the distance travelled by the sound wave is 294 μ s seconds. However, this is the time taken to travel forward and bounce backwards. In order to get the distance in cm, you need to multiply the received travel time value from the echo pin by 0.034 and divide by 2.

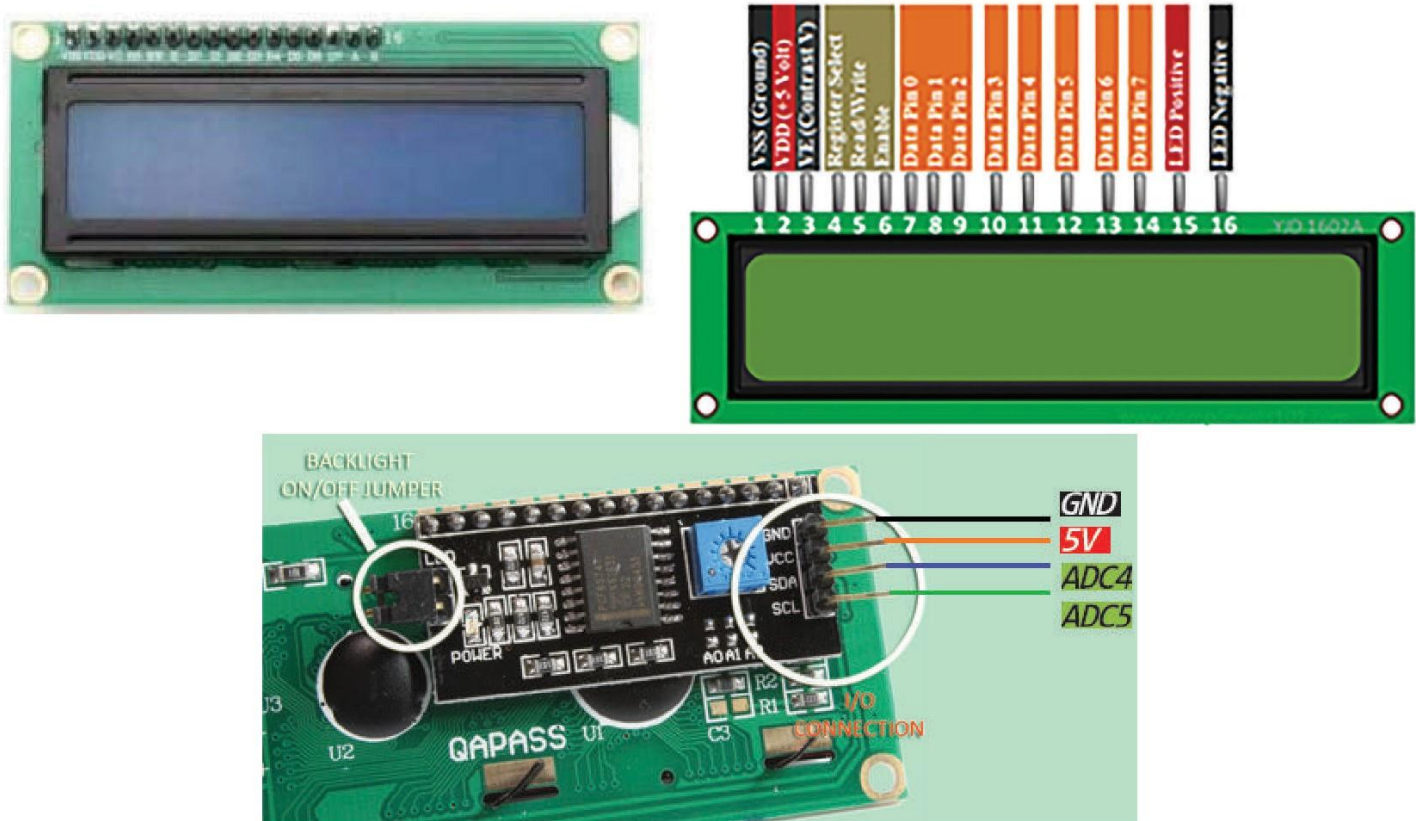


LCD Display with I2C backpack - 16x2

LCD modules are very commonly used in most embedded projects, the reason being it is programmer friendly, a less expensive option, and easy to acquire. Most of us would have come across these displays, such as part of a printer or a calculator.

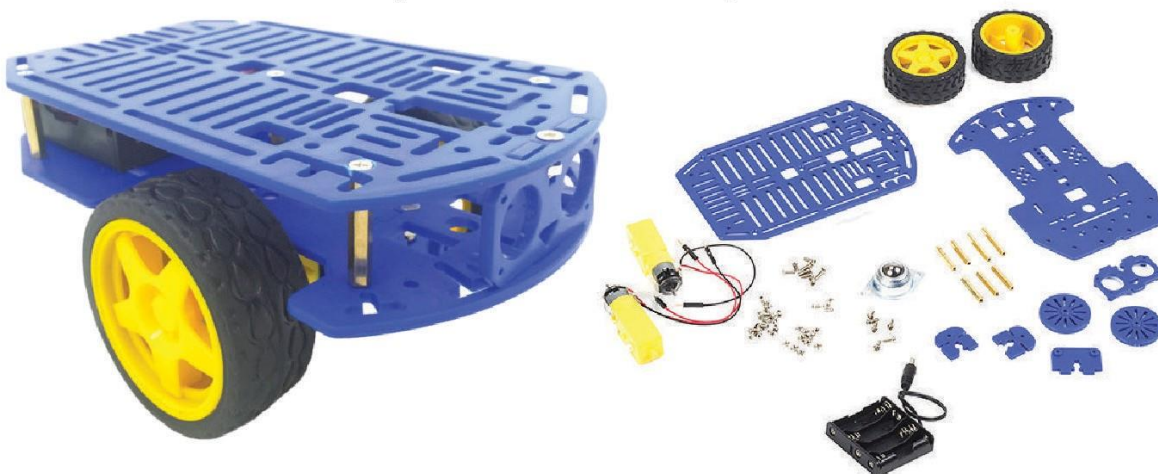
16x2 LCD stands for 16 columns and 2 rows. There are many combinations available such as 8x1, 8x2, 10x2, 16x1 etc. Although, the most commonly used one is the 16x2 LCD. This LCD has 32 characters in total and each character will be made up of 5x8 pixel dots.

The LCD that is included in this kit is coming with I2C backpack. This LCD display with I2C backpack converts 16pins I/O to 4pins I/O. Instead of soldering 16pins, it's easy to use 4 pins to communicate with the LCD and potentiometer is embedded on the I2C pack by which you can adjust the brightness of LCD backlight. This LCD display is great for beginners. The appearance and the pinouts are shown below.



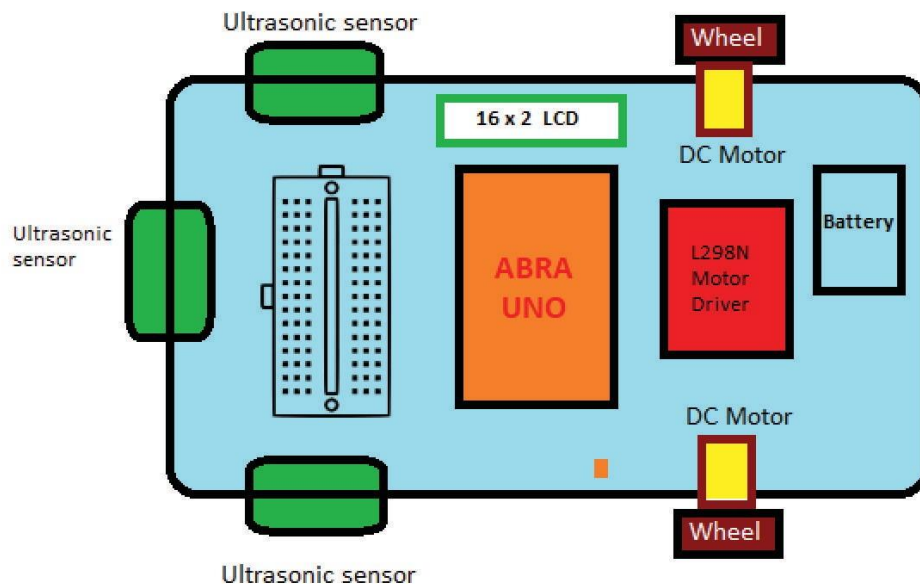
Chassis & Motors Kit

This kit comes with a 110x174mm Chassis board along with spacers, screws, 2 DC motors with mounts and two 65mm wheels. The chassis is made up of plastic with a wide variety of mounting holes for sensors, motors and other devices. The input of the DC motor is 6VDC and can handle current up to 250mA. This kit also includes a caster wheel which helps the bot to move freely in all directions.



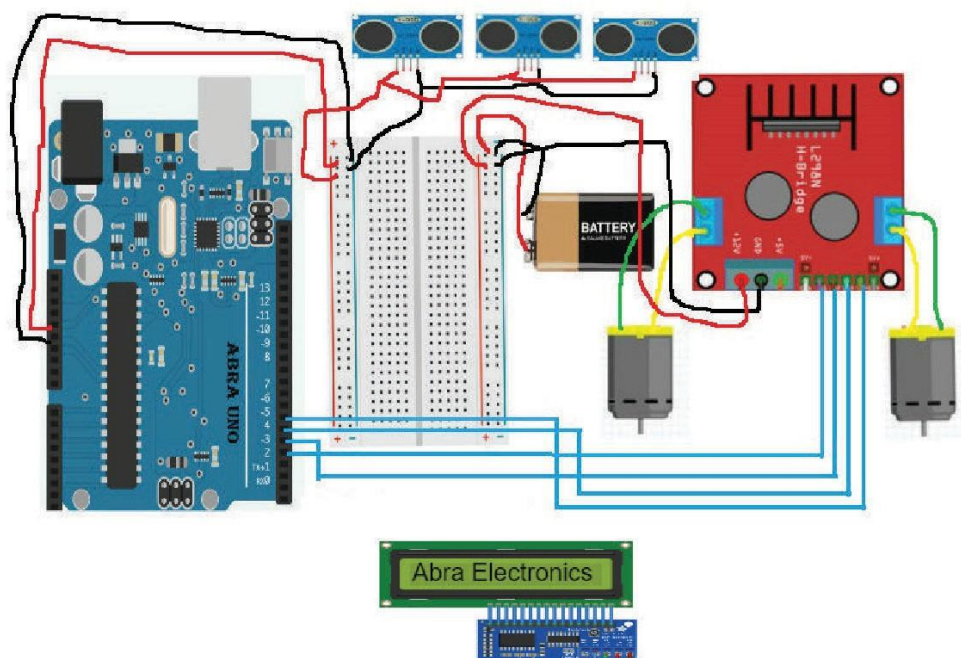
Blocks For Maze Solving Robot

Below is the block diagram of the maze solving robot.



CIRCUIT DIAGRAM

Below is the block diagram of the maze solving robot.

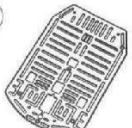

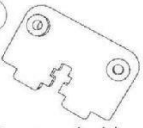
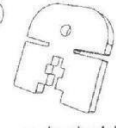
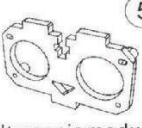


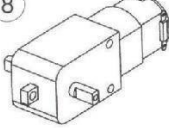
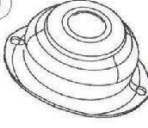

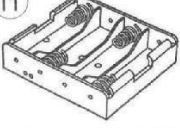
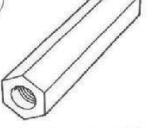
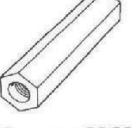
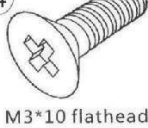








MAZE SOLVING ROBOT ASSEMBLY

Now that you are familiar with each component, let's begin the construction of the robot car.

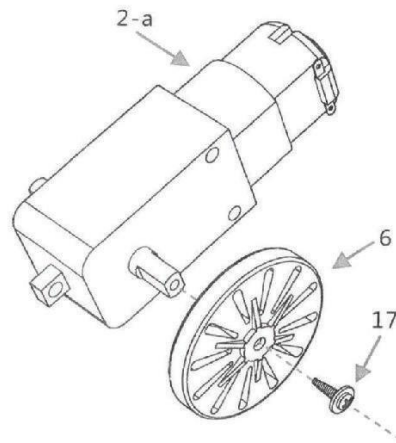
FRAME CONSTRUCTION

Partlist :

1  Chassis-up 1PC	2  Chassis-bottom 1PC	3  IR sensor holder 1PCS	4  encoder holder 2PCS	5  Ultrasonic module holder 1PCS
6  Speed board holder 2PCS	7  Motor holder 4PCS	8  Motor 2PCS	9  Omni wheel 1PC	10  Wheel 2PCS
11  Battery holder 1PC	12  L30 spacer 2PCS	13  L25 spacer 5PCS	14  M3*10 flathead screw 6PCS	15  M3*8 screw 4PCS
16  M3 nut 10PCS	17  M2.3*8self-tapping screw with flange 2PCS	18  M3*6 flathead screw 12PCS	19  M3*6 screw 2PCS	20  M3*30 screw 4PCS

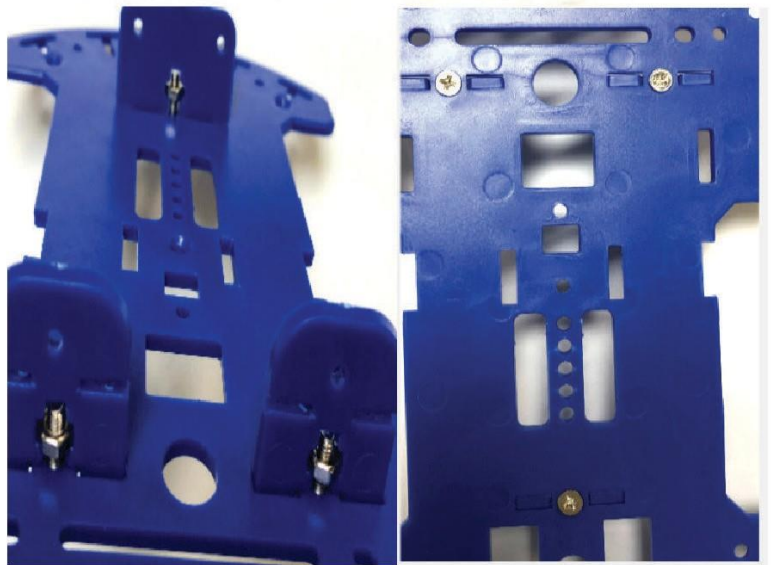
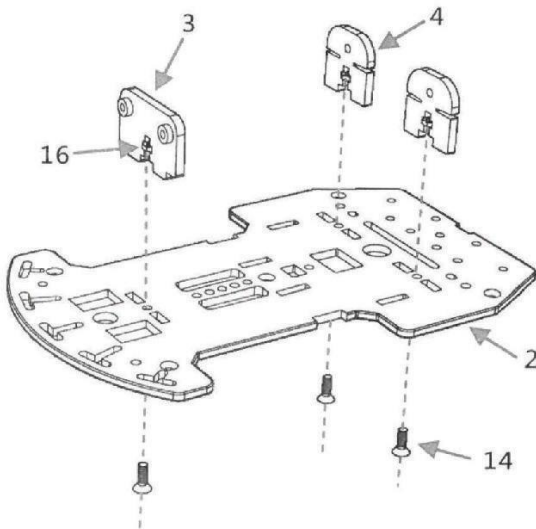
Step1

In the first part of the frame assembly, you shall start with motors. Affix the encoders to the motors as shown in the image using a screwdriver. Cross check the parts using the numbers provided in the left part of the image with the list of parts.



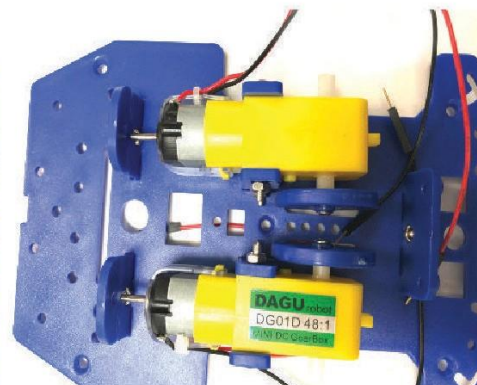
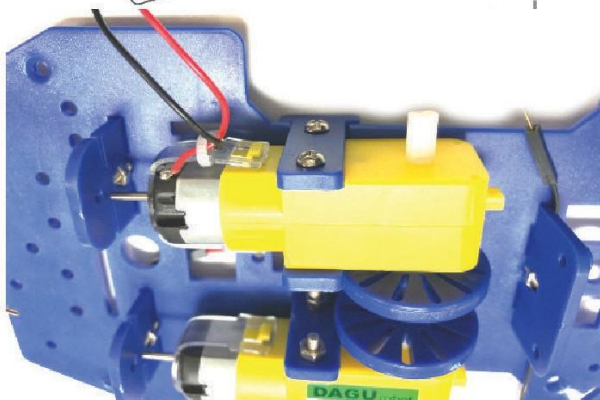
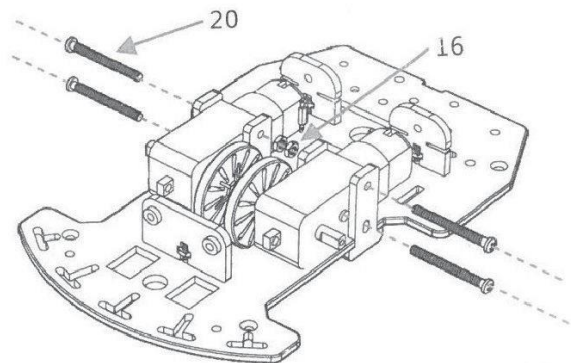
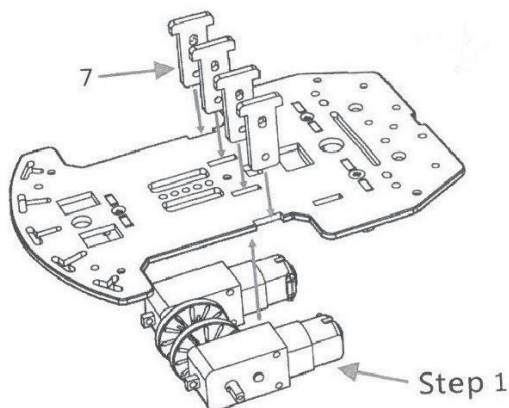
Step2

In the next step, assemble the mounts as shown in the figure below. Make sure you are affixing the mounts on the ride side of the frame. Keep some white surface below the robot so that you will not miss any small parts.



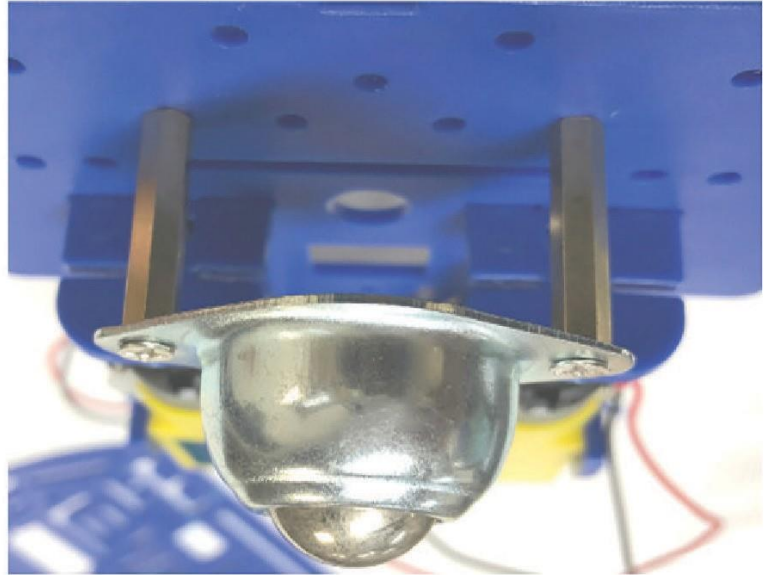
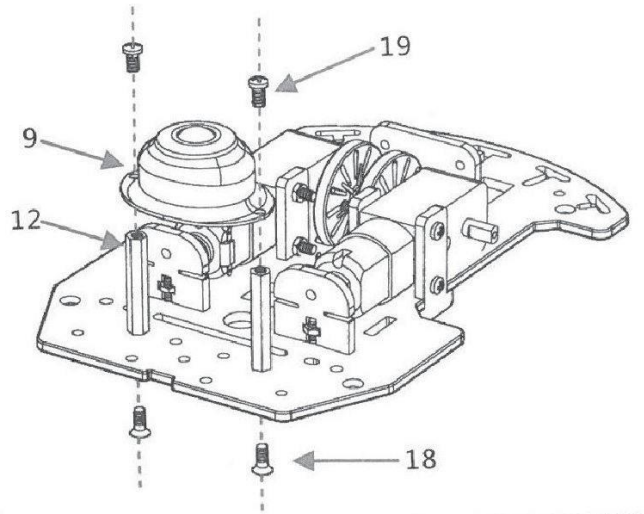
Step3

Now its time to assemble the motor mounts along with the motors. There is 4 big screws provided in the pack. Using these screws, affix the motors along with mounts as shown in the figure below.



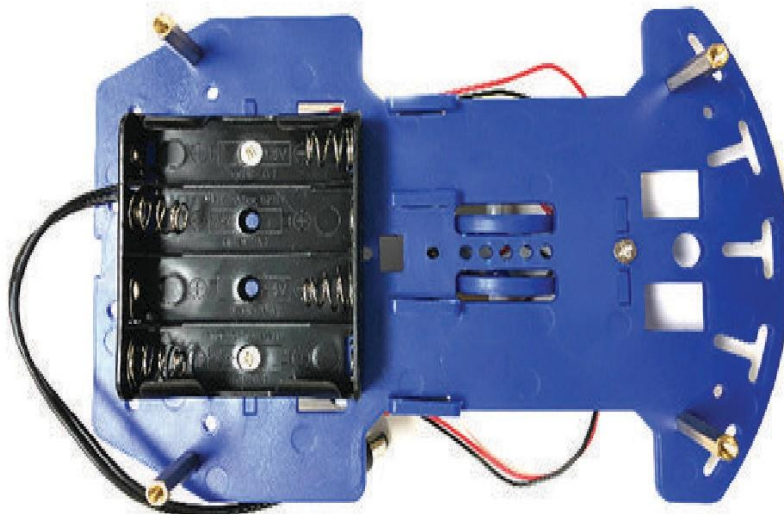
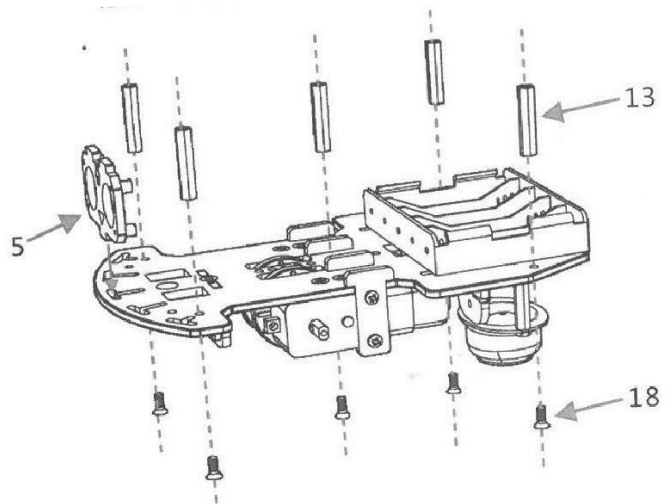
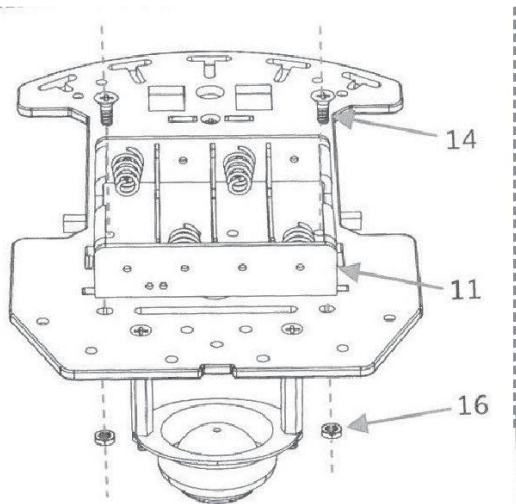
Step4

After you have attached the motors, Affix the caster wheel on the same side of the motors using the spacers.



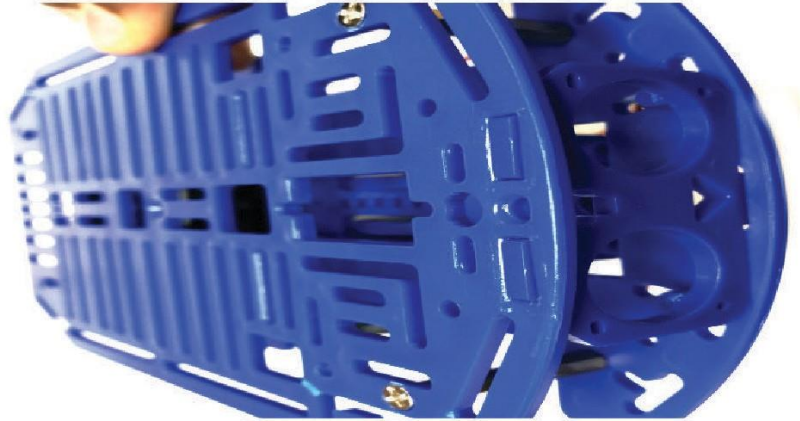
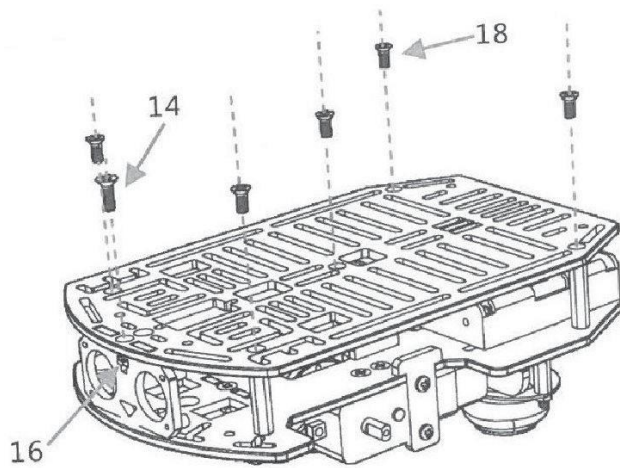
Step5

Next, flip your frame and affix the AA battery holder with the help of small screws.



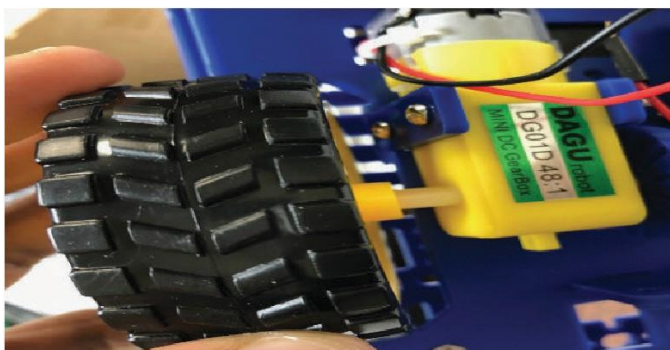
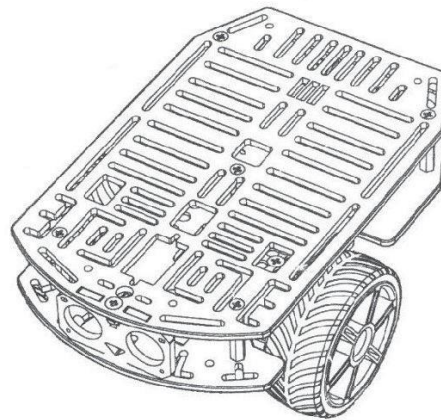
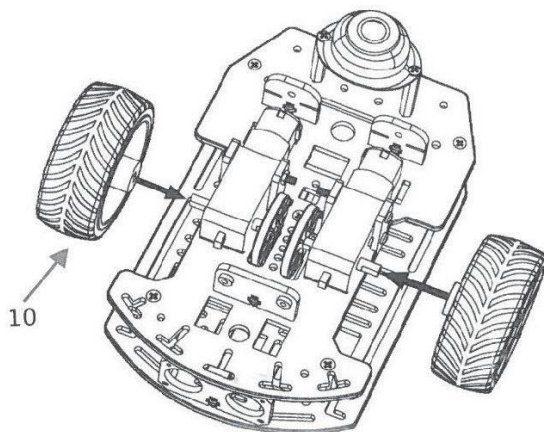
Step6

Finally, mount the upper part of the frame using spacers and one ultrasonic sensor mount gave with the frame.



Step7

Finally, Affix the wheels to the motors. There is no need for any screws for these wheels. You can see the hole in the wheel and fit it to the motor perfectly.



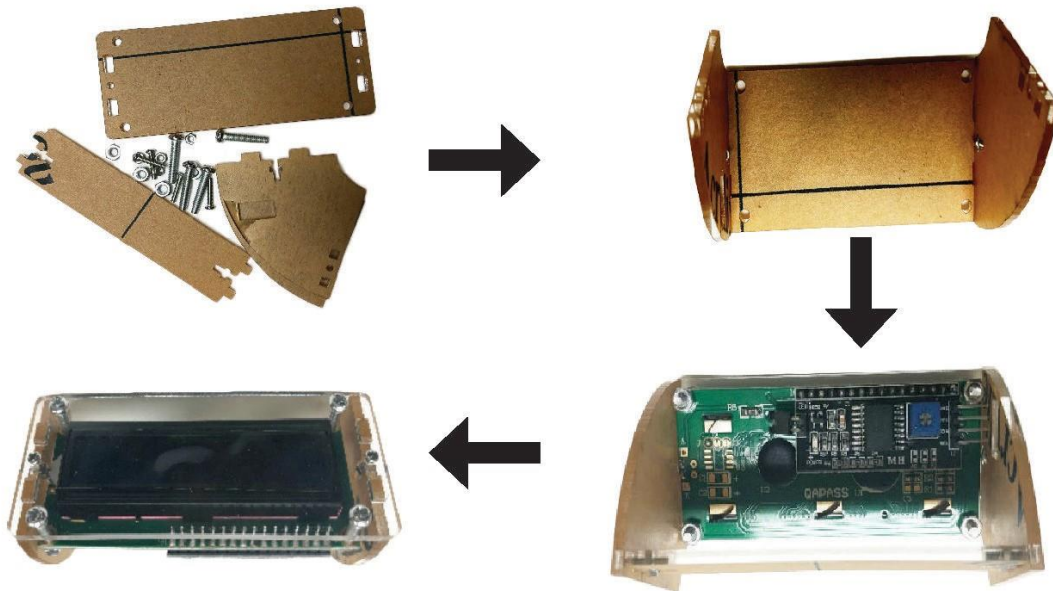
Next, you need to go for electronic mounts and connections.

ASSEMBLY OF ELECTRONICS

Now it is time to mount all the electronic components.

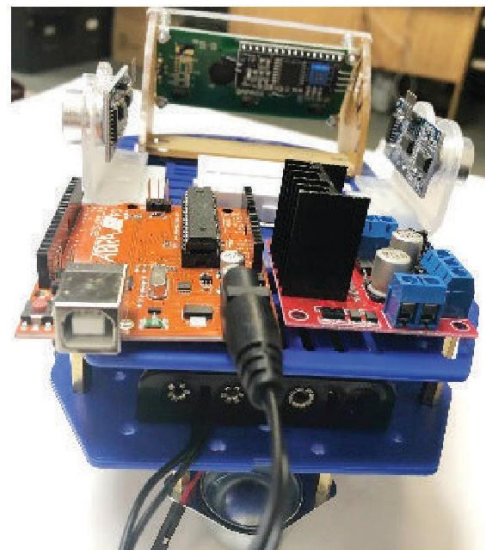
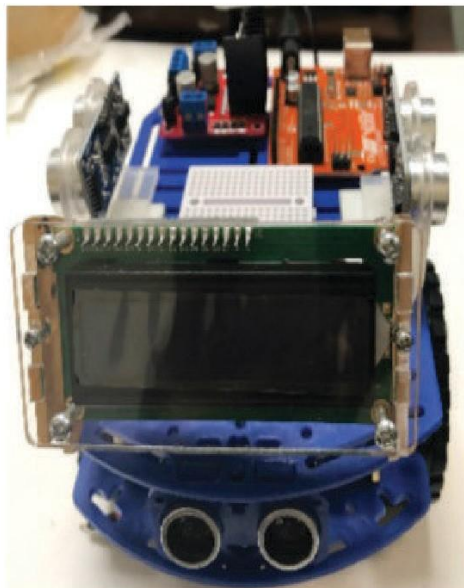
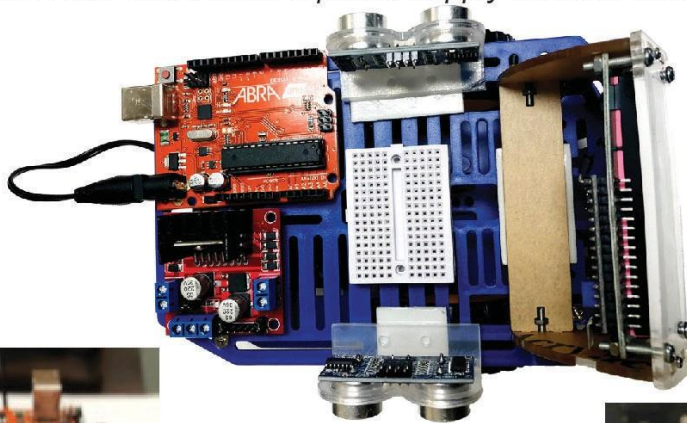
Step1

The first step is to affix the LCD provided in this kit to the frame. Remove the protective film on the LCD before mounting. Now Assemble the acrylic holder as shown below and mount the LCD.



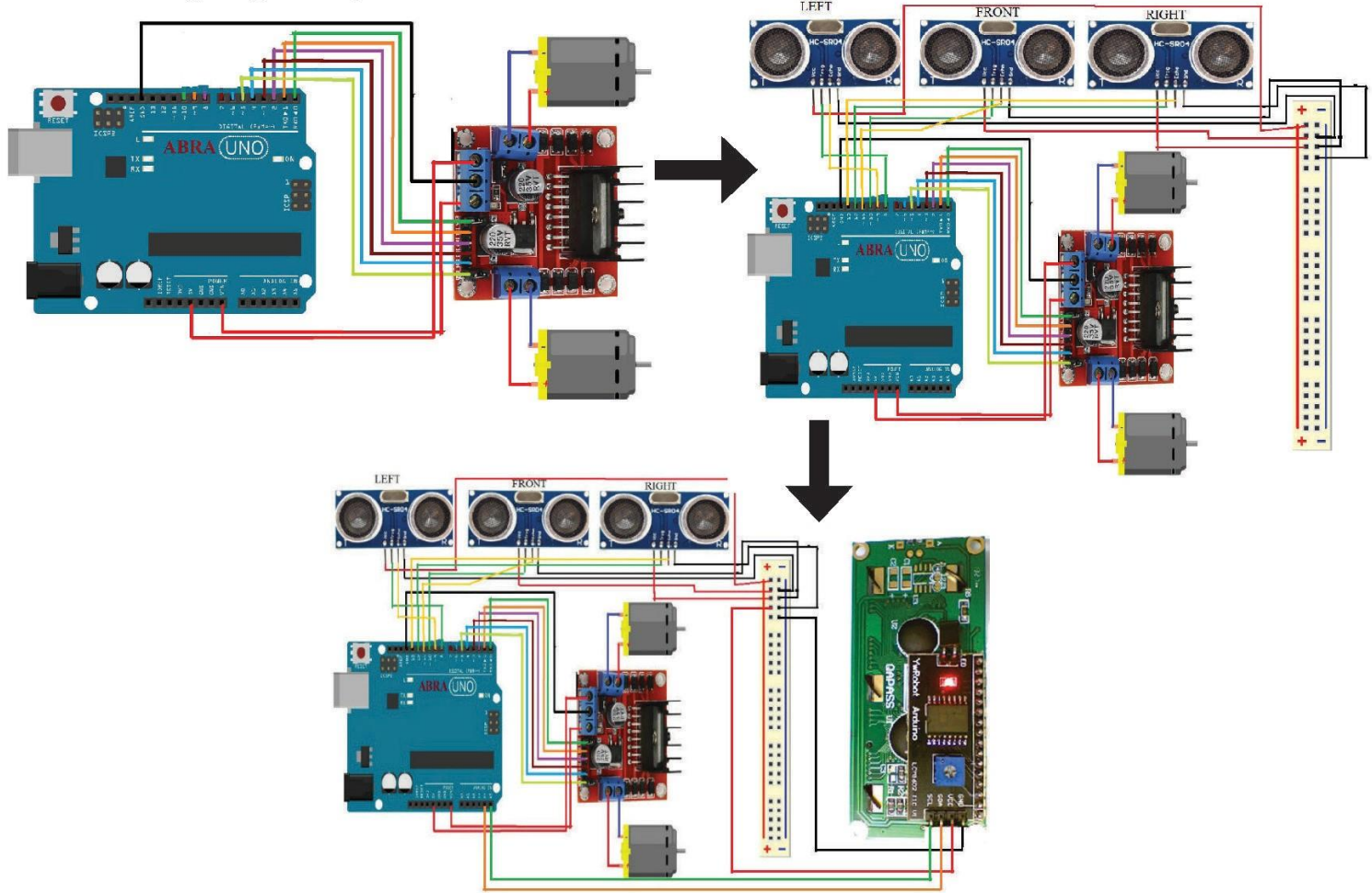
Step2

Now it is time to affix the LCD on the robot. Mount the LCD on the top of the robot by using the double-sided tape. Then affix the ABRAUNO, Motor Driver, Ultrasonic Sensors, and the Breadboard as shown below. One of the ultrasonic sensors goes between the frame and the other two goes on the top of the frame. *Note: Make sure the direction of the ABRAUNO's power supply is to the backside of the board.*

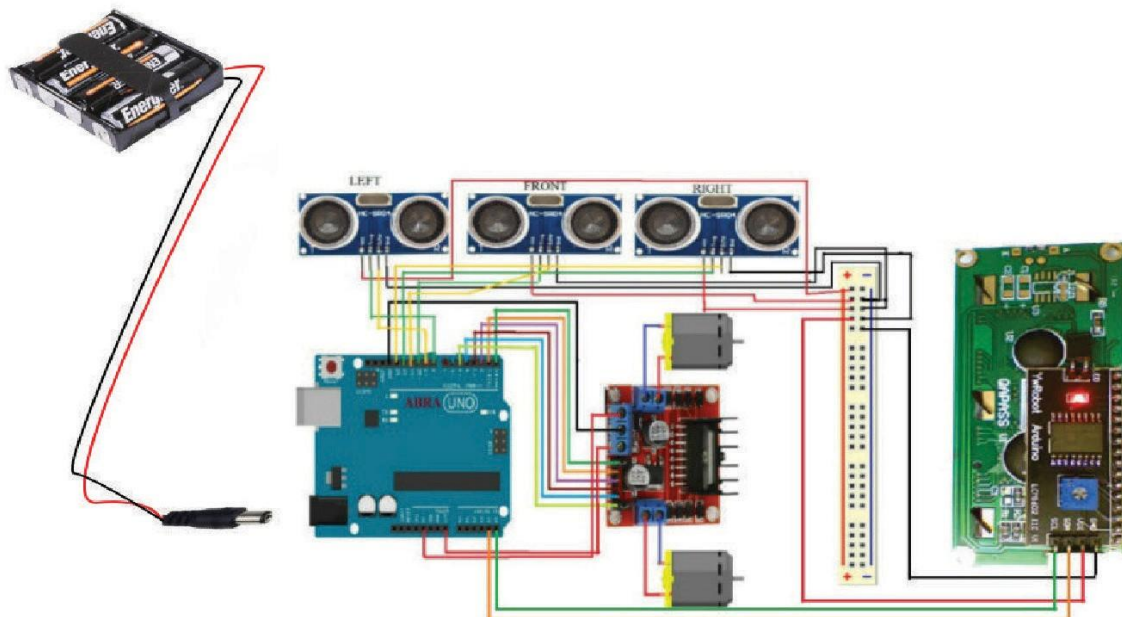


Connections

Now you must connect the motors to the driver, then the sensors, LCD and the motor driver to the ABRAUNO. The connecting diagram is provided below. Follow the connection directions in the pictures.



- For the first connection, we connected the motors to the motor driver and the driver to the ABRAUNO. Make sure the connections are exactly like the diagram shown above without any parallax errors.
- In the second connection, the 3 ultrasonic sensors which are named as Left, Right and Front are connected to the ABRAUNO with the help of the breadboard for multiple bus.
- Finally, the LCD display is connected to the ABRAUNO as shown above.
- At last check all the connections and also check the pin diagrams in component description of this manual.



Coding

The next step in this kit is to program the ABRAUNO.

Software Required

Firstly, you will need to install the software on your computer. Arduino IDE is the software required to program the ABRAUNO, follow the software installation guide below.

1. Visit <http://www.arduino.cc/en/main/software> to download the latest Arduino IDE version for your computer's operating system. It is compatible with Windows, Mac, and Linux systems.
2. On the download page click on the "Windows installer" option for the easiest installation.
3. Save the .exe file to your hard drive.
4. Open the .exe file.
5. Click the button to agree to the licensing agreement.
6. Decide which components to install, then click "Next"
7. Select which folder to install the program to, then click "Install"
8. Wait for the program to finish installing, then click "close".
9. Now find the Arduino shortcut on your desktop and click on it. The IDE will open and you'll see the code editor
10. The next thing to do is to make sure the software is set up for your Arduino board. Go to the "tools" drop-down menu and find "Board". Another menu will appear, where you can select from a list of Arduino models. You have to choose **Arduino Uno R3** for our ABRAUNO.

For more info on how to install, please visit:

<https://startingelectronics.org/software/arduino/installing-arduino-software-windows-10/>

Algorithms Used

Stuck on what is an algorithm?

It is a set of rules to be followed to solve the complex maze. As we discussed above, the robot does not know which type of maze it is and the length of the maze. For your robot to know which type of maze, you need to write a set of rules or instructions so the robot take decisions accordingly. The commonly used algorithm for maze solving is Wall Follower algorithm.

What is Wall follower algorithm?

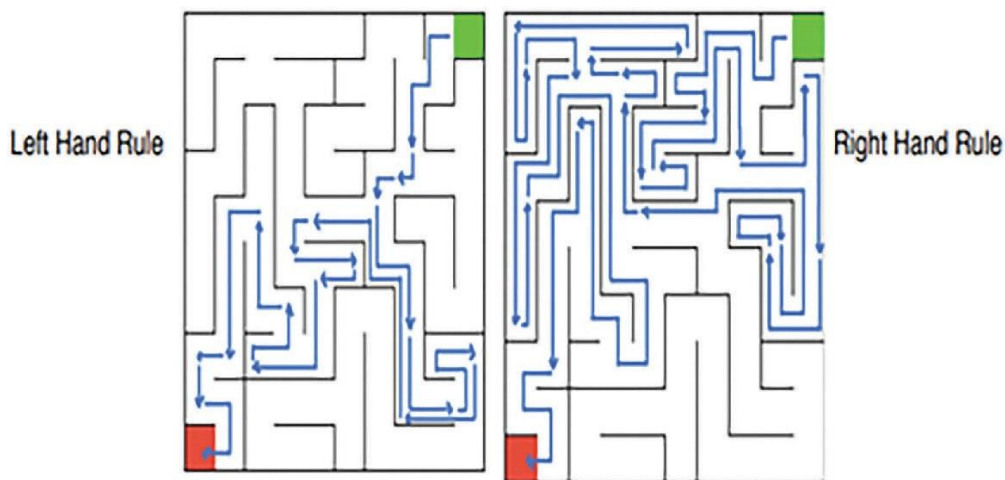
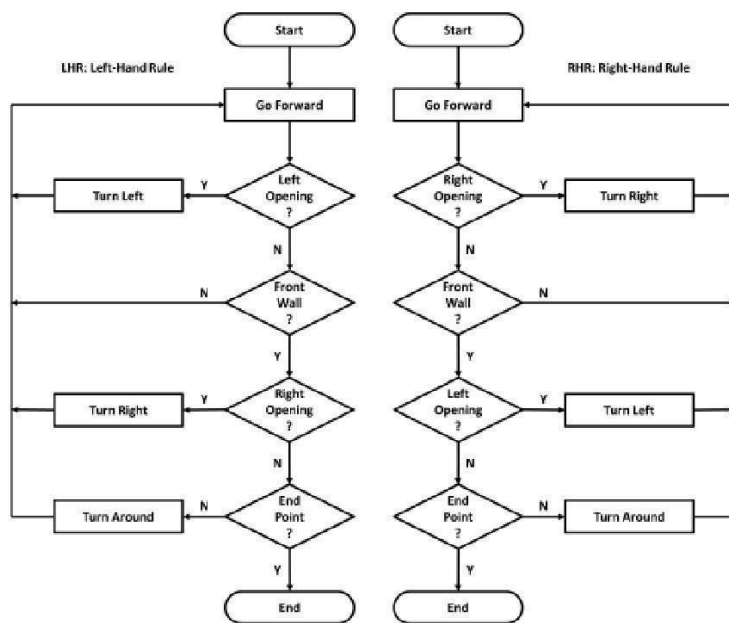
As the name of the algorithm implies, the robot keeps an eye of its surroundings. This means that whether there is an opening for the robot, it chooses that path. While the robot takes different paths, the robot records the walls that it has encountered and maps out the maze. This algorithm works best in a situation where the walls are connected to each other. The robot will have a tough time when it encounters a 4-way road since there is no walls to follow in the centre of the road. This algorithm has two subsets in terms of solving the maze: "**Left-hand rule**" and "**Right-hand rule**".

Left-hand Rule-LHR

The principle of the Left-hand Rule is the robot would follow the path of the maze referencing the walls on the left side of the robot. Even if the LHR leads the robot to a dead end, the robot will travel to that point regardless. This rule does not aim perfection or does not aim to find the shortest distance. The LHR with the “Right-hand rule” is designed to map the maze and calculate the shortest path.

Right-hand Rule-RHR

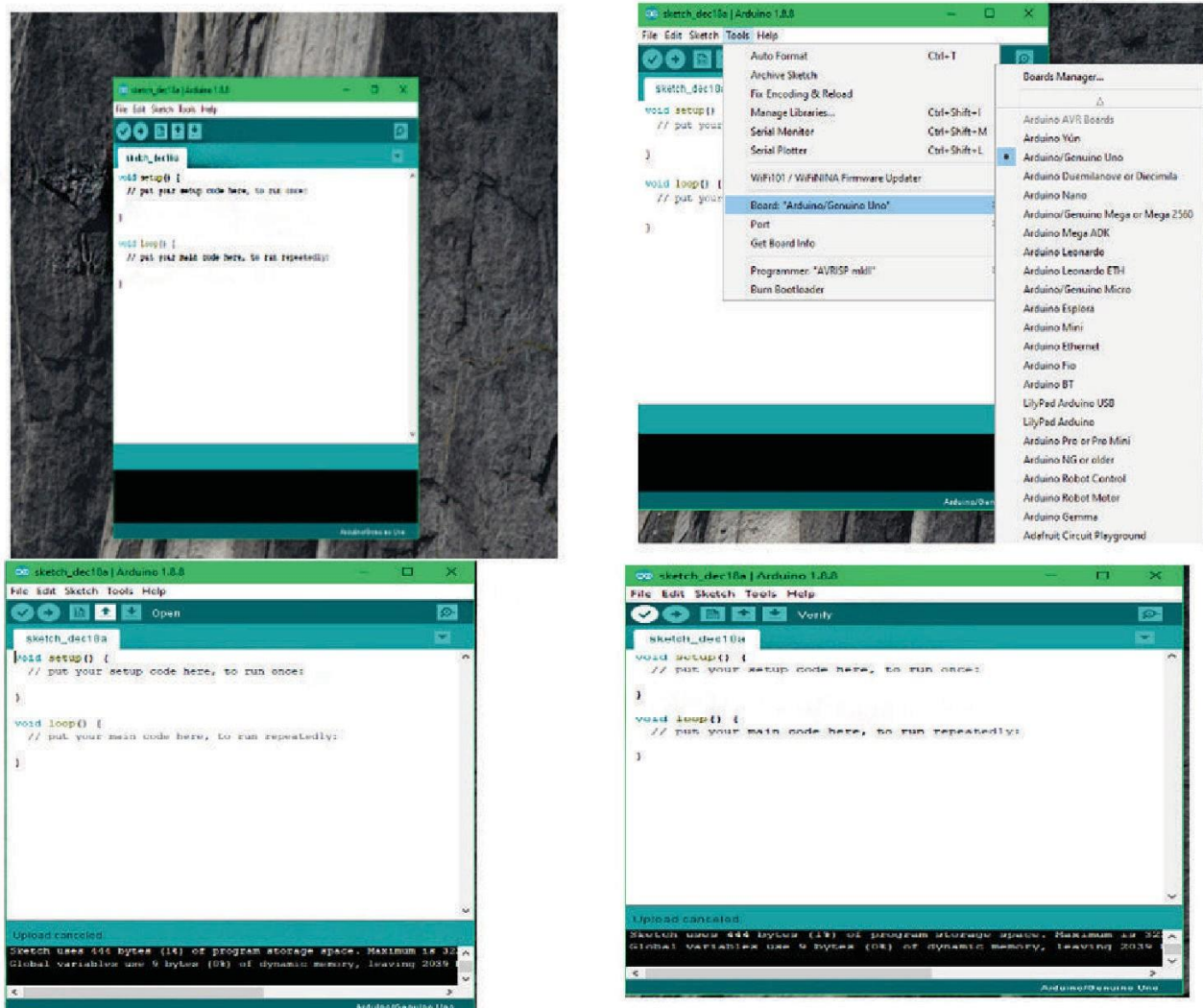
The principle of the “Right-hand rule” is similar to the LHR but instead the RHR prioritizes turning right instead of left. With understanding of this logic you can to make our maze solving robot. A helpful demonstration of the flowchart and a comparison of the two rules can be seen below.



Code

The full code is provided down and is also available at www.abra-electronics.com. Search for “AK-CAR-05”, and the code will be under attachments. This code includes programming instructions for the maze solving robots with three ultrasonic sensors and LCD display.

Follow the instructions below on how to upload the code to the ABRAUNO.



Open Arduino IDE → Select board (Arduino Uno) → open file → verify → Upload

If there is any error with NewPing, it means that libraries are not installed in your PC.

Go to **Sketch** → **Include Library** → **Manage Libraries** → **Search for Newping** → **Install**.

Install all the required libraries.

Complete Code

```
#include <Wire.h>
#include <LiquidCrystal_PCF8574.h>

LiquidCrystal_PCF8574 lcd(0x27); // set the LCD address to 0x27 for a 16 chars and 2 line display
//Install all required libraries
int motorL1=4;
int motorL2=5;
int motorR1=2;
int motorR2=3; // assigning motors pins
int echo_front=13;
int trig_front=12;
int echo_left=9;
int trig_left=8;
int echo_right=11;
int trig_right=10; // assigning Sensors pins
int pt=0;
long
time_front,distance_front,minimum_front,distance_left,minimum_left,time_left,distance_right,minimum_right,time_right;

void setup()
{
  Serial.begin(115200); // See http://playground.arduino.cc/Main/I2cScanner
  Wire.begin();
  Wire.beginTransmission(0x27);
  lcd.begin(16, 2); // initialize the lcd
  Serial.begin(9600);
  pinMode(trig_front,OUTPUT);
  pinMode(echo_front,INPUT);
  pinMode(trig_left,OUTPUT);
  pinMode(echo_left,INPUT);
  pinMode(trig_right,OUTPUT);
  pinMode(echo_right,INPUT);

  pinMode(motorL1,OUTPUT);
  pinMode(motorL2,OUTPUT);
  pinMode(motorR1,OUTPUT);
  pinMode(motorR2,OUTPUT);

  // indicating all input output pins
}

void loop()
{
  // main loop stars from here

  lcd.setBacklight(255);
  lcd.home(); lcd.clear();
  lcd.print("I AM YOUR ROBOT");
```

```

lcd.setCursor(0,1); lcd.print("WELCOME"); // initialising LCD display
sensor_front();
sensor_left();
sensor_right();

if(distance_front<=minimum_front)
{
  stop_robot();
  Serial.println("Robot Stop");
  lcd.clear();
  lcd.print("STOP");
  delay(100);
  if((distance_right<=minimum_right)&&!(distance_left<=minimum_left))
  {
    reverse_robot();
    Serial.println("Robot Back-1");
    lcd.clear();
    lcd.print("MOVING BACK");
    delay(100);
    turn_left();
    pt=-1;
    Serial.println("Robot Left1");
    lcd.clear();
    lcd.print("MOVING LEFT");
    delay(100);
    forward_robot();
    Serial.println("Robot Forward1");
    lcd.clear();
    lcd.print("MOVING FORWARD");
    delay(100);

  }
  else if( (distance_left<=minimum_left)&&!(distance_right<=minimum_right))
  {
    reverse_robot();

    Serial.println("Robot Back0");
    lcd.clear();
    lcd.print("MOVING BACK");
    delay(100);
    turn_right();
    pt=1;
    Serial.println("Robot Right1");
    lcd.clear();
    lcd.print("MOVING RIGHT");delay(500);
    forward_robot();
    Serial.println("Robot Forward11");
    lcd.clear();
    lcd.print("MOVING FORWARD");delay(500);

  }
  else if((distance_left>=minimum_right)&&(distance_right>=minimum_left))

```



```

{
    reverse_robot();

    Serial.println("Robot Back1");
    lcd.clear();
    lcd.print("MOVING BACK");
    delay(100);
    turn_left();
    pt=-1;
    Serial.println("Robot Left2");
    lcd.clear();
    lcd.print("MOVING LEFT");delay(500);
    forward_robot();
    Serial.println("Robot Forward111");
    lcd.clear();
    lcd.print("MOVING FORWARD");delay(500);
}
else
{
    do
    {
        reverse_robot();
        Serial.println("Robot Back2");
        lcd.clear();
        lcd.print("MOVING BACK");delay(500);
        sensor_front();
        sensor_left();
        sensor_right();
    }
    while((distance_left<=minimum_left)&&(distance_right<=minimum_right));
    stop_robot();
    goto deadend;
}
}
else if((distance_right<=minimum_right)&&!(distance_left<=minimum_left))
{
    reverse_robot();
    Serial.println("Robot Back22");
    lcd.clear();
    lcd.print("MOVING BACK");delay(100);
    turn_left();
    pt=-1;
    Serial.println("Robot Left4");
    lcd.clear();
    lcd.print("MOVING LEFT");delay(100);
    forward_robot();
    Serial.println("Robot Forward1111");
    lcd.clear();
    lcd.print("MOVING FORWARD");delay(100);
}
else if((distance_left<=minimum_left)&&!(distance_right<=minimum_right))
{

```

```

reverse_robot();
turn_right();
pt=1;
Serial.println("Robot Right2");
lcd.clear();
lcd.print("MOVING RIGHT");delay(100);
forward_robot();
Serial.println("Robot Forward11111");
lcd.clear();
lcd.print("MOVING FORWARD");delay(100);
}
else if((distance_left>=minimum_left)&&(distance_right>=minimum_right))
{
reverse_robot();
turn_left();
forward_robot();
pt=-1;
Serial.println("Robot Left5");
lcd.clear();
lcd.print("MOVING LEFT");delay(100);
forward_robot();
Serial.println("Robot Forward1.6");
lcd.clear();
lcd.print("MOVING FORWARD");delay(100);
}
else
{
forward_robot();
Serial.println("Robot Forward2");
lcd.clear();
lcd.print("MOVING FORWARD");delay(100);
}
if(0)
deadend:
{
if((distance_left>=minimum_left)&&(distance_right>=minimum_right))
{
if(pt==1)
{
turn_left();
pt=-1;
Serial.println("Robot Left6");
lcd.clear();
lcd.print("MOVING LEFT");delay(100);
forward_robot();
Serial.println("Robot Forward3");
lcd.clear();
lcd.print("MOVING FORWARD");delay(100);
}
else if(pt==1)
{
turn_right();

```



```

    pt=1;
    Serial.println("Robot Right3");
    lcd.clear();
    lcd.print("MOVING RIGHT");delay(100);
    forward_robot();
    Serial.println("Robot Forward4");
    lcd.clear();
    lcd.print("MOVING FRONT");delay(100);
    }
}
else if((distance_left>=minimum_left))
{
    turn_left();
    pt=-1;
    Serial.println("Robot Left7");
    lcd.clear();
    lcd.print("MOVING LEFT");delay(100);
    forward_robot();
    Serial.println("MOVING FRONT");
    lcd.clear();
    lcd.print("MOVING FRONT");delay(100);
    }
else if((distance_right>=minimum_right))
{
    turn_right();
    pt=1;
    Serial.println("Robot Right4");
    lcd.clear();
    lcd.print("MOVING RIGHT");delay(100);
    forward_robot();
    Serial.println("Robot Forward6");
    lcd.clear();
    lcd.print("MOVING FRONT");delay(100);
    }
}
}
}

```

```

void sensor_front()
{
    digitalWrite(trig_front, LOW);
    delayMicroseconds(5);
    digitalWrite(trig_front, HIGH);
    delay(20);
    digitalWrite(trig_front, LOW);
    time_front=pulseIn(echo_front,HIGH);
    distance_front=time_front/58.2;
    Serial.println("distance front=");
    Serial.println(distance_front);
    delay(10);
    minimum_front=30;
}

```

```

void sensor_left()
{
  digitalWrite(trig_left, LOW);
  delayMicroseconds(5);
  digitalWrite(trig_left, HIGH);
  delay(20);
  digitalWrite(trig_left, LOW);

  time_left=pulseIn(echo_left,HIGH);
  distance_left=time_left/58.2;
  Serial.println("distance left=");
  Serial.println(distance_left);
  delay(10);
  minimum_left=30;
}

void sensor_right()
{
  digitalWrite(trig_right, LOW);
  delayMicroseconds(5);
  digitalWrite(trig_right, HIGH);
  delay(20);
  digitalWrite(trig_right, LOW);

  time_right=pulseIn(echo_right,HIGH);
  distance_right=time_right/58.2;
  Serial.println("distance right=");
  Serial.println(distance_right);
  delay(10);
  minimum_right=30;
}

void forward_robot()
{
  digitalWrite(motorL1,LOW);
  digitalWrite(motorL2,HIGH);
  digitalWrite(motorR1,LOW);
  digitalWrite(motorR2,HIGH);// if motors wont move front ,rearrange the pins
  delay(2000);
}

void reverse_robot()
{
  digitalWrite(motorL1,HIGH);
  digitalWrite(motorL2,LOW);
  digitalWrite(motorR1,HIGH);
  digitalWrite(motorR2,LOW);
  delay(2000);
}

void stop_robot()
{
  digitalWrite(motorL1,LOW);
  digitalWrite(motorL2,LOW);

```



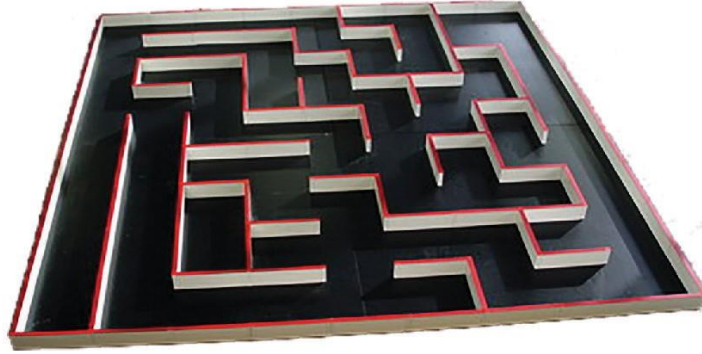
```
digitalWrite(motorR1,LOW);  
digitalWrite(motorR2,LOW);  
delay(2000);  
}
```

```
void turn_left()  
{  
  digitalWrite(motorL1,LOW);  
  digitalWrite(motorL2,LOW);  
  digitalWrite(motorR1,LOW);  
  digitalWrite(motorR2,HIGH);  
  delay(2000);  
}
```

```
void turn_right()  
{  
  digitalWrite(motorL1,LOW);  
  digitalWrite(motorL2,HIGH);  
  digitalWrite(motorR1,LOW);  
  digitalWrite(motorR2,LOW);  
  delay(2000);  
}
```

Strategies For Constructing Arena

After finishing making the maze solving smart car, now you must make your own arena. You can make the walls using any type of material like wood, foam, cardboard, etc. Although the easiest way to make the walls is to use foam and tape. The Walls should be a minimum height of 108.27mm, or taller than the ultrasonic sensors on the car. The reason being is that the sensors should send signals and receive it back. The thickness of the wall does not matter. The path of the maze can be built using any algorithms, either the Left-Hand, or Right-Hand, or both. An example of a built maze is shown below.



For more ideas for maze solving arenas watch some videos below.

Link1: https://www.youtube.com/watch?time_continue=2&v=WUtdfhdl8H4

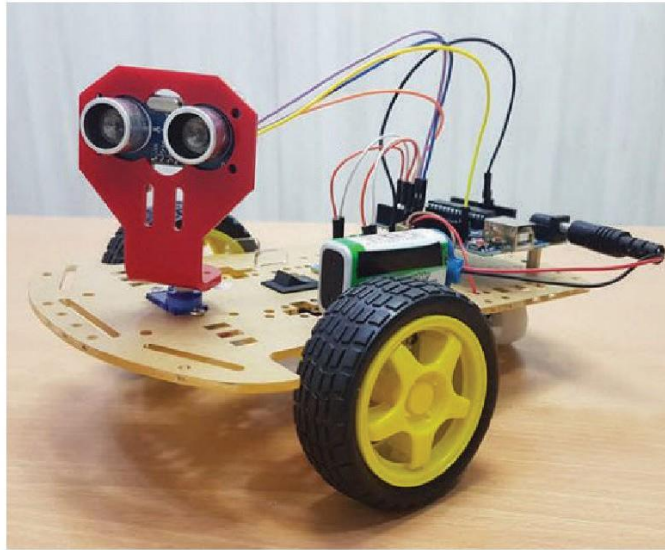
Link2: https://www.youtube.com/watch?v=ekeUSu_ubf8

What Else We Can Do With This Kit?

Here are some other ideas you can do with this kit

Obstacle Avoiding Robot

This robot keeps away from the obstacle. By using the components provided in this kit, you can build this robot. You can also display the direction of the object on the LCD. The chassis construction will be the same as the maze solving robot. Additionally, you need to add one ultrasonic sensor along with a servo motor provided in the kit. By using the program in the ABRAUNO, you can make the robot move away from an object or obstacle. This is a great project for students.

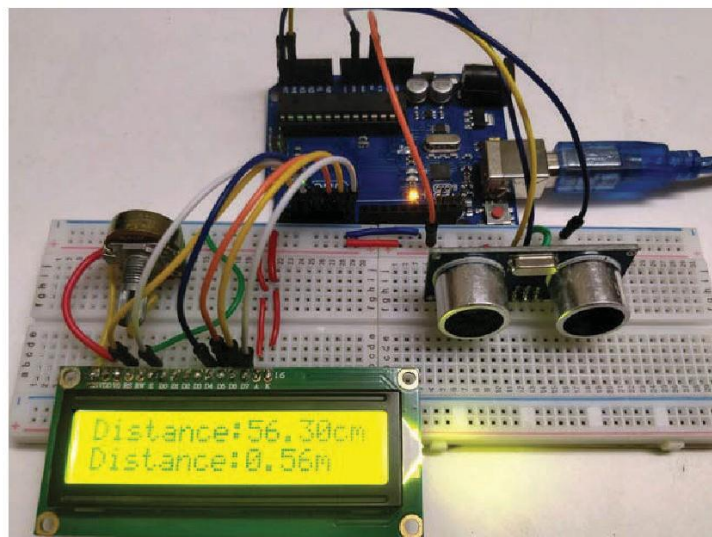


Wall Follower Robot

This robot just follows a wall or moves along the wall side. For this robot, you need to keep one ultrasonic sensor on the side of the robot and you need to program ABRAUNO according to the requirements. Also, display the distance travelled by bot on the LCD.

Object Distance Calculation Robot

This robot helps you to find the distance of an object and go towards it. The robot detects the object and finds the distance from the reference point of the bot. Then it moves towards the object and updates the distance on the LCD screen of the robot.



To know more about these models and its code, please go to www.abra-electronics.com and search for "AK-CAR-05".