USER GUIDE

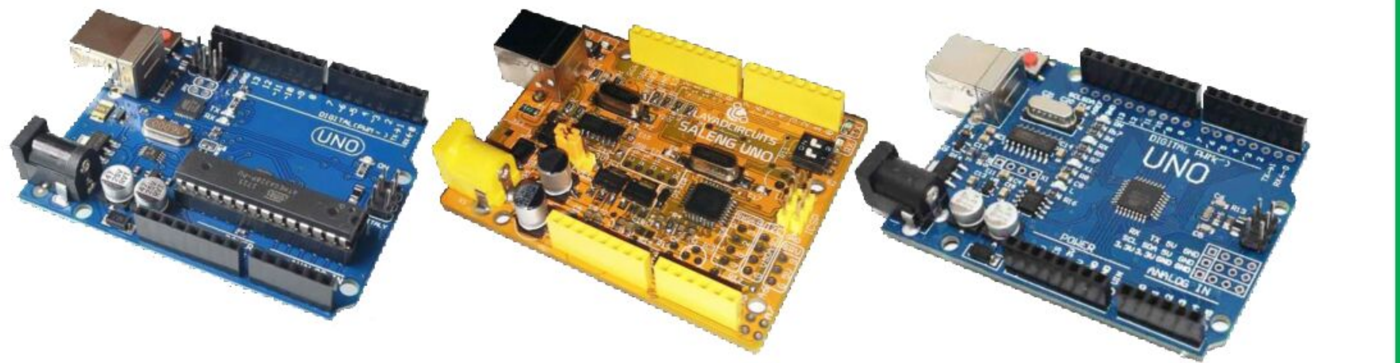
KIT008: Basic 2WD Robot Kit

Thank you for purchasing Layad Circuits' KIT008 or the Basic 2WD Robot Kit. This kit is designed as an introductory kit for microcontroller-based mobile robots. Aside from the kit specifications, this user guide will cover instructional information on how the robot is assembled and shall also provide exercises that demonstrates the capabilities of the kit. This document is prepared to cater to beginners as a step by step guide in building and programming a working mobile robot. It may also serve as reference for advanced users. Mechanical assembly, electronics and some degree of C programming are the ideal prerequisites in performing these exercises.

KIT CONTENTS

ARDUINO/SALENG UNO

At the heart of the kit is a microcontroller board from one of the possible options below. All these boards are based on the ATmega328 microcontroller chip and all work with the software tool called Arduino IDE.

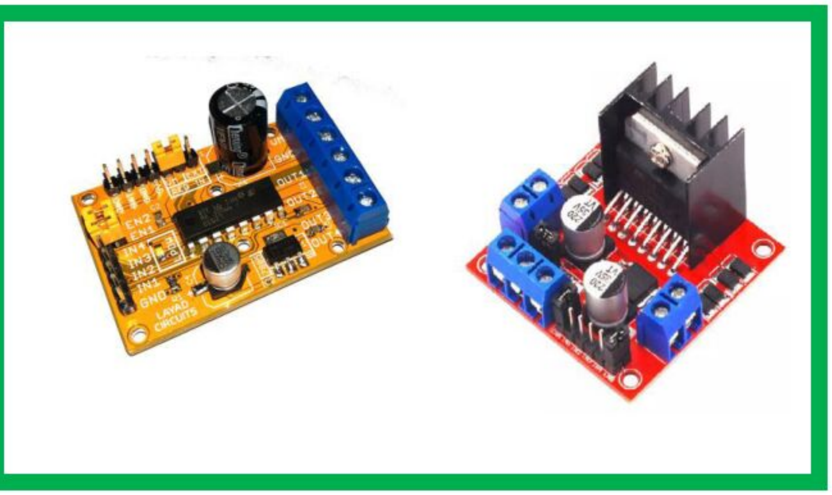

Option A: Arduino Uno DIP version Option B: Saleng Uno Option C: Arduino Uno SMD version

USB CABLE

A type-A-to-type-B USB cable is provided mainly as a means to program the Arduino/Saleng board from a computer. It may also be used to interface a computer to the microcontroller.

MOTOR DRIVER MODULE

Your kit will come with either the L298N or the L293D (Kimat L293D by Layad Circuits) motor driver modules. The motor driver module is a modular circuit used to control the DC motors using the Arduino's General Purpose I/O (GPIO) pins. There are two such circuits, called a full H-bridge circuit, built into the module. This module in combination with the Arduino allows direction and speed control of the DC motors.



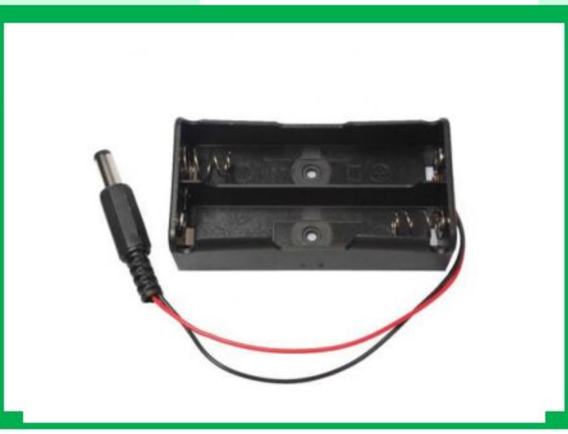
18650 LITHIUM-ION BATTERIES

The kit includes two 18650 Lithium Ion (li-Ion) batteries rated at least 800mAh. These batteries may come in a variety of colors and either button or flat top. The batteries have a nominal voltage rating of 3.7V and maximum full charge voltage of 4.2V.

- Do not short circuit
- Dispose of properly
- Do not incinerate
- Do not disassemble

BATTERY HOLDER

The battery holder accepts two Li Ion batteries and connects them in series forming a total maximum voltage of 8.4V at its DC plug.



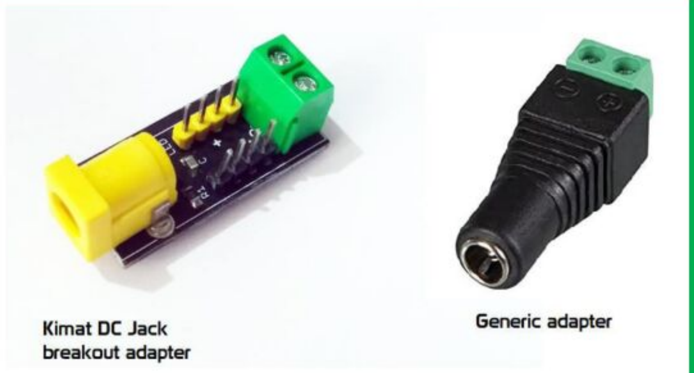
ULTRASONIC SENSOR

The Arduino may use the ultrasonic sensor module to determine distance following sonar time of flight technique. That is, to emit ultrasonic waves and determine the time they bounce back into the sensor. Distance is then determined in the program by simple math given the time and known speed of sound in air. This sensor will be used in this kit for detecting obstacles.

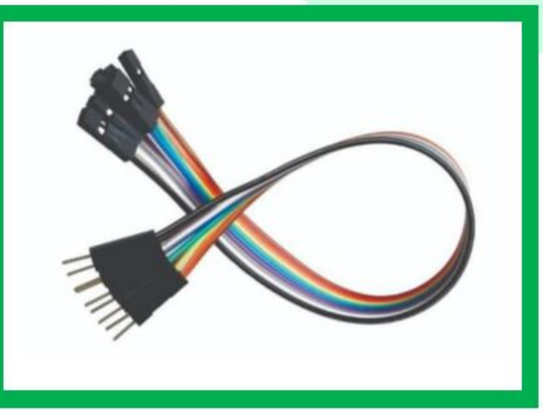
DC JACK BREAKOUT

The kit will contain either a Kimat DC Jack Breakout board or a an equivalent adapter and shall perform the following functions:

- Accepts power from the battery's DC plug
- Allows connection of the power to the motor driver with its terminal block



CONNECTORS



A set of connectors for 0.1" pin headers are provided for easier connection of signal lines. Cut-off the female headers of 2 wires as this will be needed for the power connection of the Arduino.

POWER WIRE

A two-wire cable is also included to connect the battery to the motor driver via the terminal of the DC jack breakout. This is thicker than the connectors since it is expected to carry higher current.

CABLE TIES

The cable ties are used to secure the ultrasonic sensor and other components to the acrylic base of the robot chassis.

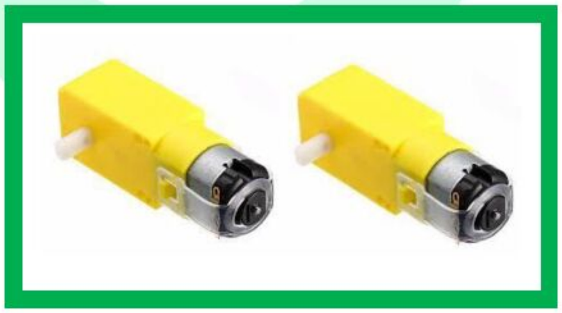
DOUBLE SIDED TAPE

The double sided foam tape is used to secure the following into the acrylic base:

- Arduino/Saleng Uno
- Motor Driver
- Battery holder
- DC Jack breakout

DC MOTORS

Two DC motors are included. The motors have built-in gear boxes for added torque. The motor shafts are fitted with plastic coupling for ready interface into the wheels. If you need wires soldered into the motor terminals, contact us at info@layadcircuits.com or facebook.com/layadcircuits for the customization of your kit.



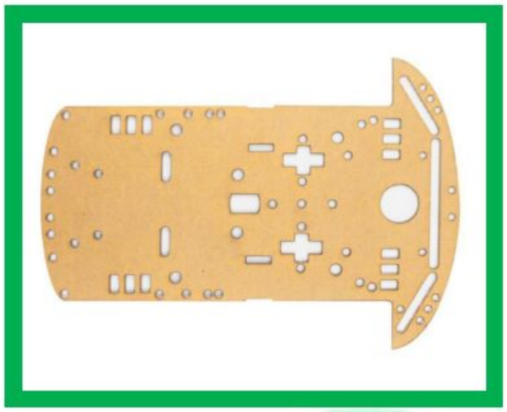
WHEELS

The plastic wheels are made of light-weight material and are ready to fit into the DC motor shafts. Care should be taken when inserting the wheel into the shaft

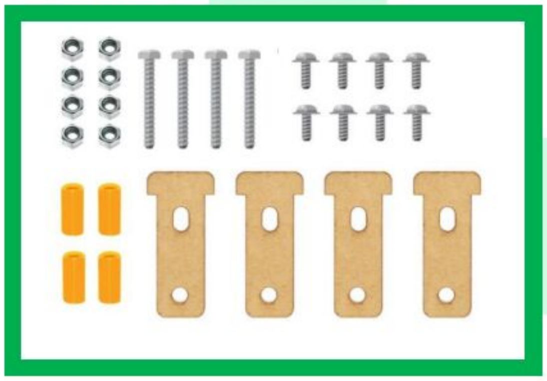
ACRYLIC BASE BOARD

The base board holds everything in place. Underneath the board, the motors and wheels are attached using the provided acrylic clips and screws. The rest are installed on top.

The base board is made with multiple holes for maximum options in the installation of the different boards and components.



BASE BOARD ATTACHMENTS



The kit also comes with different screws, standoff and acrylic clips for the attachment for the wheels and motors into the base board

THIRD WHEEL

A smaller caster wheel is also provided as a support for the body. This is not driven by the motors but merely as a support to maintain the base board horizontal

ARDUINO OVERVIEW

The Arduino/Saleng Uno board contains a microcontroller unit (MCU), the ATMega328P, installed within the printed circuit board.

The board contains the ATmega328P and the essential circuits such as:

- Regulator circuit – converts the higher voltage input from the DC jack to 5V which is used by the rest of the circuit. In the Saleng Uno, a 3.3V regulator circuit is also integrated allowing for 3.3V operation.
- Oscillator circuit – this is the main clock source of the MCU and is normally running at 16MHz.
- USB-to-Serial converter – this circuit allows the board to communicate directly with a computer via the USB. This is used for the programming of the board and for exchanging Serial data between MCU and computer.
- LED's – there maybe several LEDs in the board but there are 2 important LEDs:
    - The power LED – labeled as "PWR" in the Saleng Uno and may be unlabeled in the Ardino Uno board. This lights up when power is available. This is hardwired to the output of the regulator circuit.
    - The "L" LED – labaled as "L" in the Arduino or Saleng Uno board is controlled by digital pin 13 (D13) and hence may be controlled by the software.

The MCU is an Integrated Circuit that contains 3 major elements:

1. A processor – this performs the mathematical, logical operations that is central to a computer
2. Memories – the ATMega328P integrates 3 memory types:

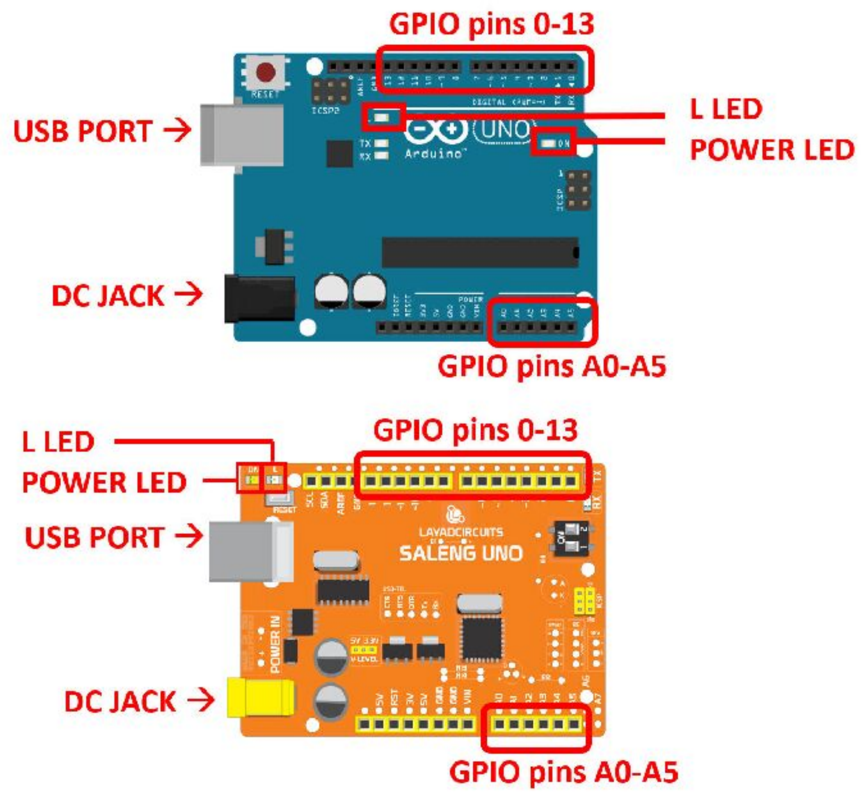
    2.1 Flash – this 32Kb memory holds the software ("program" or "code" or "sketch" in Arduino parlance) or the set of instructions that the user wants the MCU to perform. This is normally writable only during programming of the MCU ("uploading").

    2.2 RAM – specifically, a 2Kb SRAM is the MCU's "scratch pad" or working memory used during run time. Among those that get allocated into the RAM are global variables declared in the software, these, the user may access this memory any time during run time.

    2.3 EEPROM – this is a 1KB memory for storing data that are not part of the instructions but may be accessed and edited during runtime. This is good for storing settings, serial numbers, and similar. In this document, we will not be using the EEPROM.

    In terms of data volatility, the memories are either volatile or non-volatile:

Volatile Memory	Non-Volatile Memory
Data maintained only when power is available. Data is erased upon power cycle Example: RAM	Data is retained even when power is not available. Example: Flash and EEPROM



3. General Purpose Input/Output (GPIO) – are the "pins" or I/O of the MCU that are assigned by the software to function as either an input or output pin. They may also be classified according to the circuit behind the pin and the user defined function:

Digital Input	Accepts digital logic or HIGH (logic 1 / 5V) and LOW (logic 0 / 0V). All of the 20 GPIO pins may be used as digital outputs.
Digital Output	Outputs either HIGH or LOW. All of the 20 GPIO pins may be used as digital outputs.
Analog Input	Accepts a varying signal between 0V and 5V and represents the actual votlage as a number between 0 and 1023 in the program. Only pins A0 – A5 may be used as analog input pins.

PROGRAMMING

The basic development setup is composed of 3 elements:

- The Arduino/Saleng Uno – with or without the application circuit.
- USB cable
- Computer – where the Arduino IDE software is running. In this document, we will only consider Windows operating system.

Avoid using pins 0 and 1 as these are shared with the programming port. If it cannot be avoided, disconnect whatever circuit is connected during programming. Replace after programming.

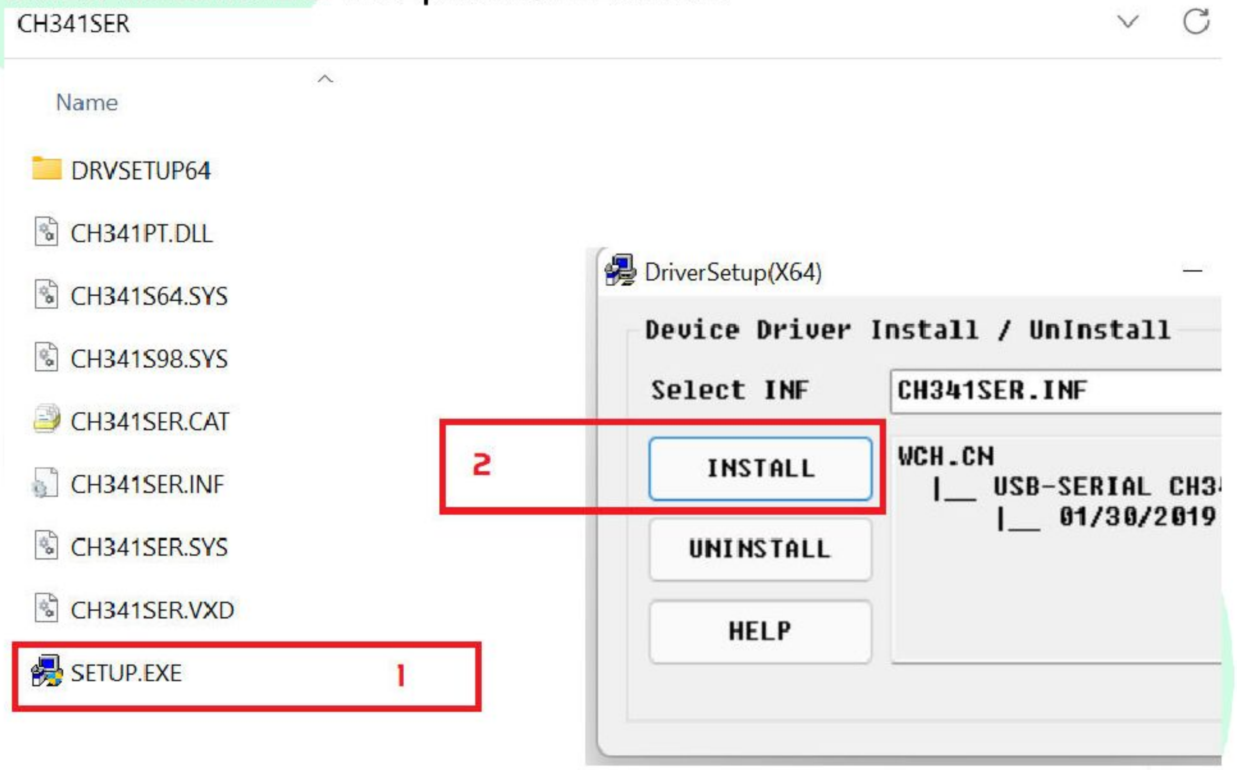
The program (code) is developed in the computer and then transfer to the Arduino/Saleng Uno during the "upload" process. In this document, we shall refer to the program/sketch simply as code.

The code is written in a slightly modified version of C and C++ programming language. The user is expected to have some knowledge of this as it is beyond the scope of this document. Important programming topics may be discussed to present certain concepts. Check with Layad Circuits if you are in need of other kits covering programming. User who may not be interested in learning C/C++ may simply copy the codes provided and perform the upload.

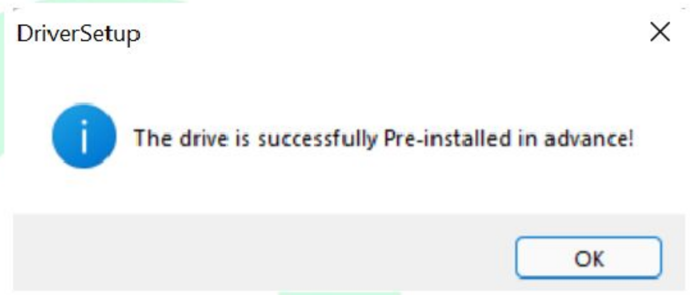
PREPARING YOUR COMPUTER

Before any programming can be done, there are a few things to prepare in the computer.

1. Kits with the Arduino Uno SMD version or Saleng Uno will need to install the USB driver separately. Kits with the Arduino Uno DIP version may skip this step as the USB drivers are installed in the succeeding steps.
    a. Download the driver from the link below:
    http://layadcircuits.com/ds/SalengUno/drivers/CH341SER.ZIP
    b. Extract the contents of the zip file. Run SETUP.EXE



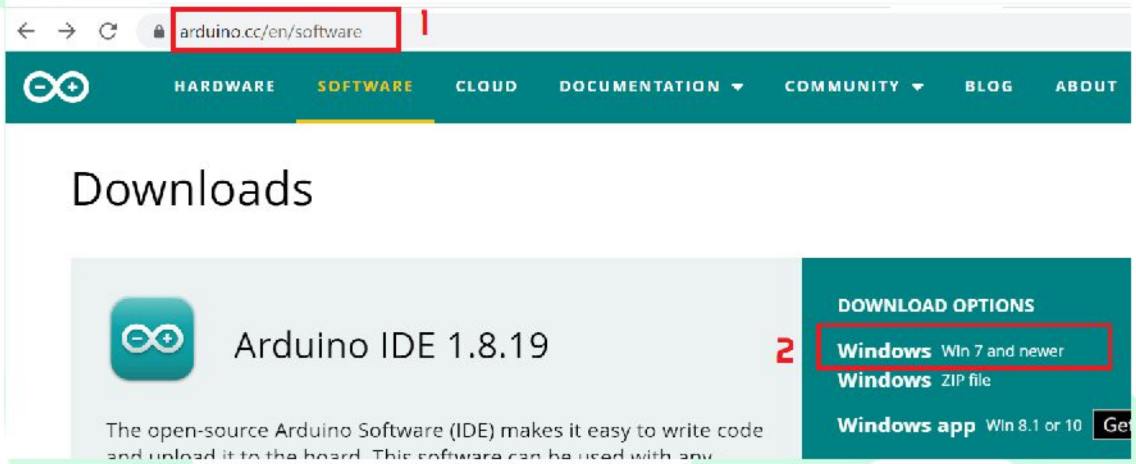
    c. Install the driver. You should see the following upon successful installation. Incase you have already installed the driver previously, installation may fail, in this case, uninstall first if you want to re-install


DriverSetup

The drive is successfully Pre-installed in advance!

d. Restart your computer

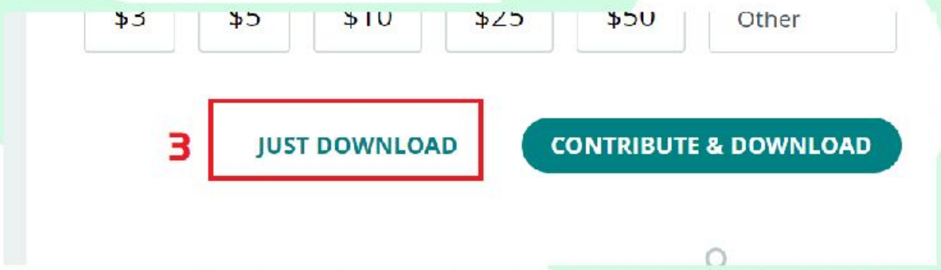
2. Download and Install the Arduino IDE software
   a. Download the installer from the link below
   https://www.arduino.cc/en/software



Downloads

Arduino IDE 1.8.19

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any

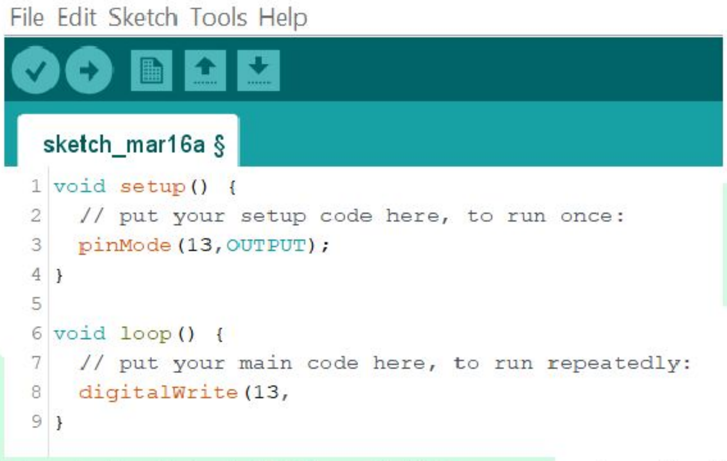


   b. Double click on the installer after downloading. Follow onscreen instructions.

ACTIVITY 1: FIRST UPLOAD

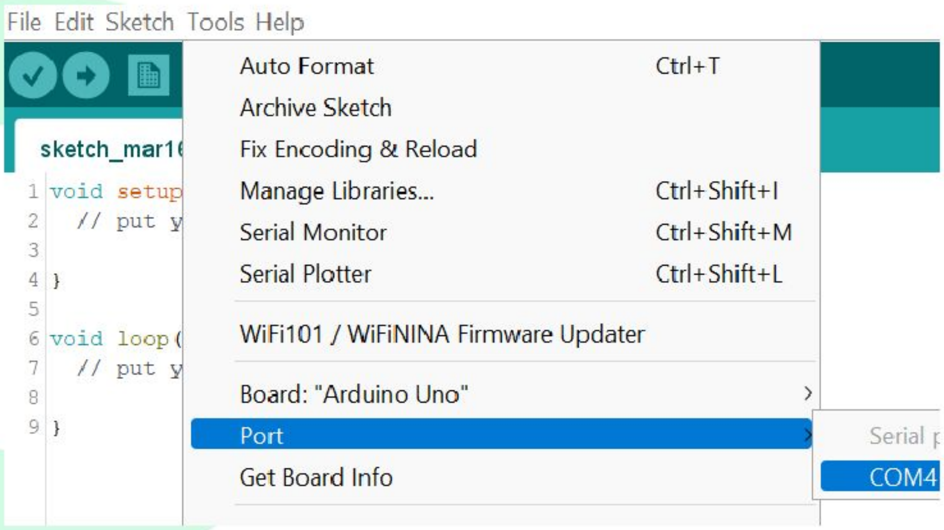
1. **Connect the Arduino/Saleng Uno into the computer using the USB cable. The computer should indicate with a notification tone that the hardware has been recognized.**

2. **Launch the Arduino IDE installed in the computer**



```
void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13,
}
```

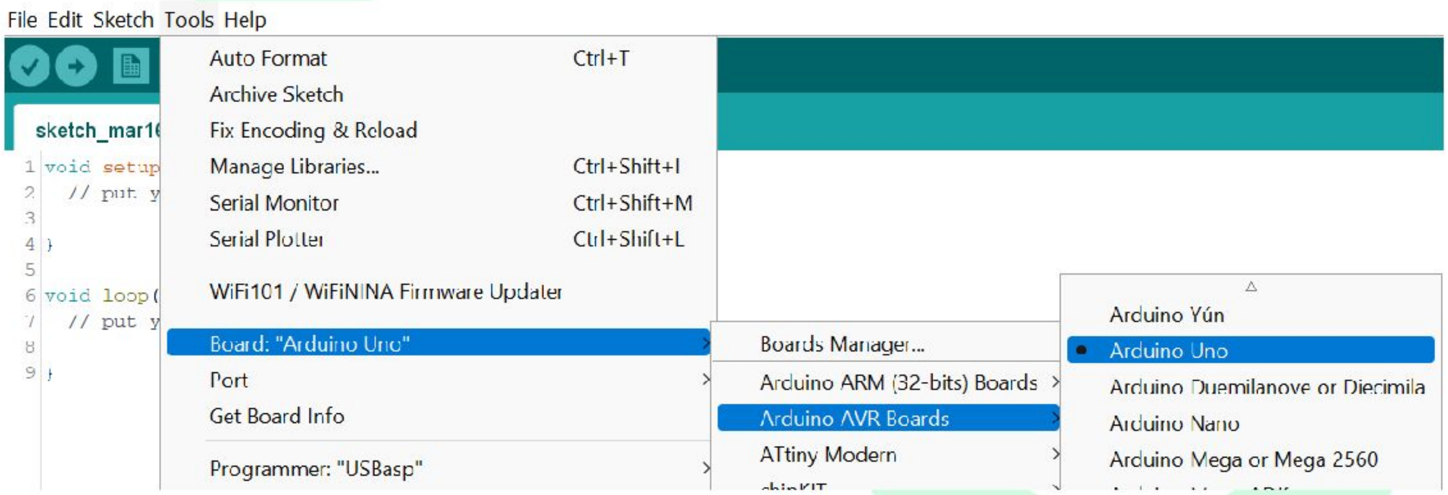
3. **Go to tools>ports and check the list of COM port(s). Your computer should assign a COM port number to your Arduino if it is detected properly. In the picture below, the COM number is COM4 and may not be the same for other computers.**


File Edit Sketch Tools Help
sketch_mar1
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Serial p
COM4

```
void setup
  // put y

}

void loop(
  // put y

}
```

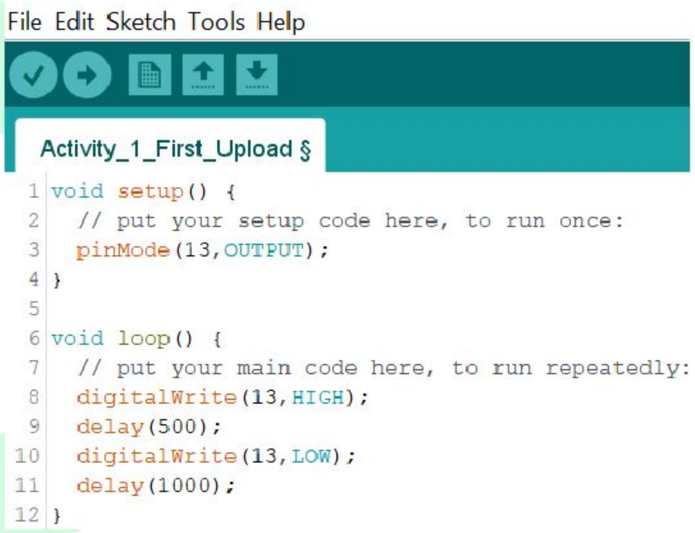
Note: If you cannot detect the hardware, please check that you have installed the USB driver if you are using Arduino Uno SMD or Saleng Uno. You may also want to check other USB ports incases when a port may be damaged. Finally, you could confirm if your hardware is detectable using another computer.

4. **Go to tools>board>Arduino AVR boards and select Arduino Uno**


File Edit Sketch Tools Help
sketch_mar1
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "USBasp"
Boards Manager...
Arduino ARM (32-bits) Boards
Arduino AVR Boards
ATtiny Modern
Arduino Yún
● Arduino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino Mega or Mega 2560

```
void setup
  // put y

}

void loop(
  // put y

}
```

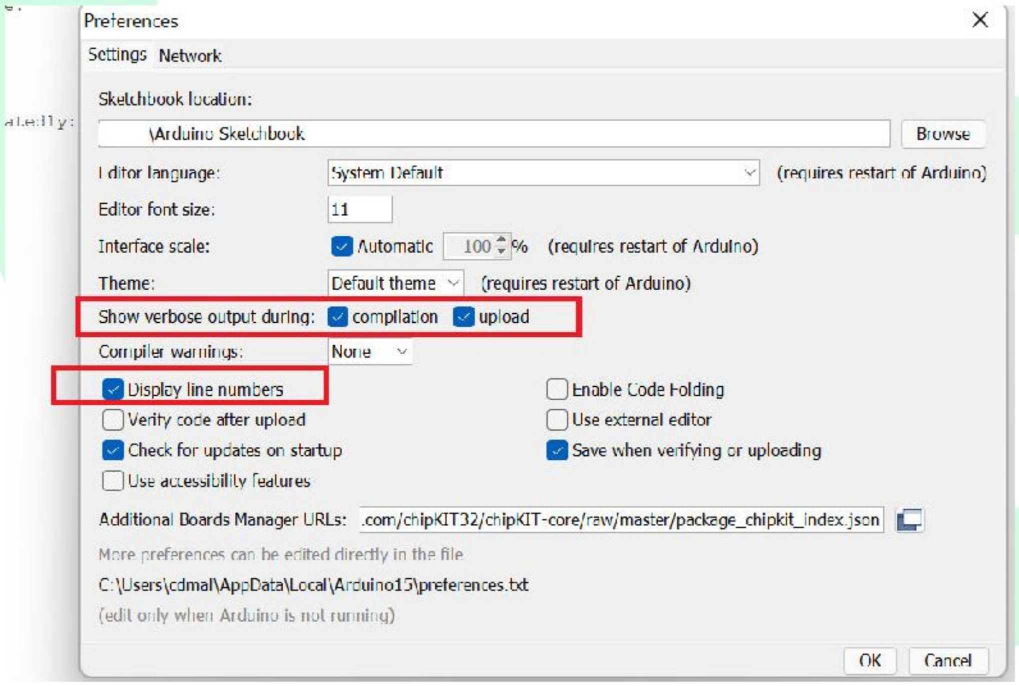
5. **Now we are set to write our first code. In this exercise, copy the program below into the Arduino IDE editor.**



Activity_1_First_Upload §

```
 1 void setup() {
 2   // put your setup code here, to run once:
 3   pinMode(13,OUTPUT);
 4 }
 5 
 6 void loop() {
 7   // put your main code here, to run repeatedly:
 8   digitalWrite(13,HIGH);
 9   delay(500);
10   digitalWrite(13,LOW);
11   delay(1000);
12 }
```

Notice that the picture shows the line numbers of the code. This is a good reference when discussing the code. To activate this feature, go to File>Preferences and this tick on the "Display line number" option. You may want to enable verbose output during compilation and upload. Follow the red boxes below:



Sketchbook location: \Arduino Sketchbook

Editor language: System Default (requires restart of Arduino)

Editor font size: 11

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: [✓] compilation [✓] upload

Compiler warnings: None

[✓] Display line numbers

[ ] Verify code after upload

[✓] Check for updates on startup

[ ] Use accessibility features

[ ] Enable Code Folding

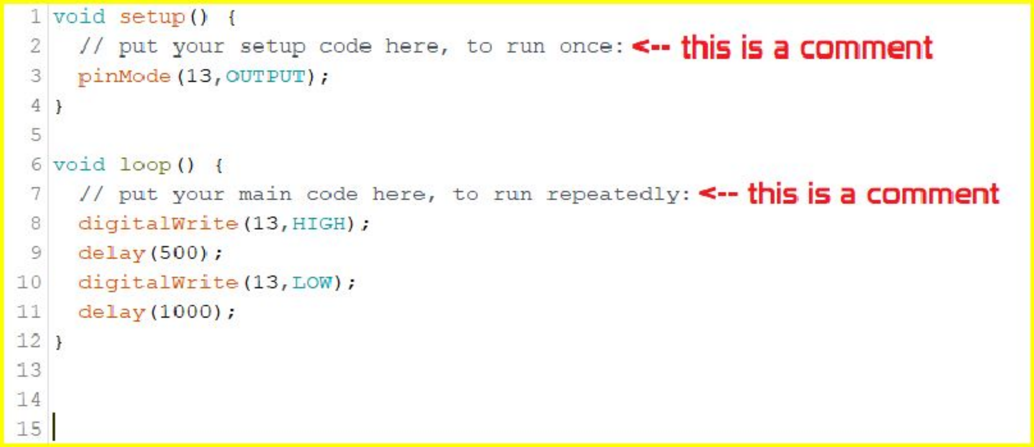
[ ] Use external editor

[✓] Save when verifying or uploading

Additional Boards Manager URLs: .com/chipKIT32/chipKIT-core/raw/master/package_chipkit_index.json

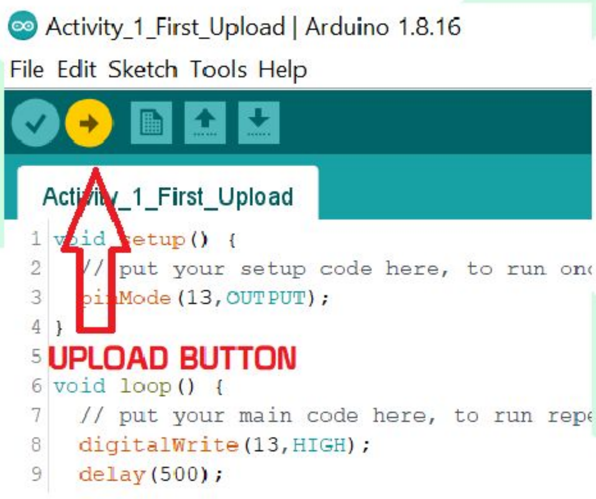
More preferences can be edited directly in the file
C:\Users\cdmal\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

REVIEW: Comments inC/C++ may be added using // followed whatever text to the right. It may also be done by enclosing the text in /* and */. Comments do not have an effect on the instructions to be uploaded into the microcontroller. These are used as helpful text for the reader of the code and may be used for purposes like improving readability, providing notes about the algorithms, and many other purposes. In our default editor screen, you should notice the two comments already on screen. Deleting these lines will not affect the code itself.



```
void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13,HIGH);
  delay(500);
  digitalWrite(13,LOW);
  delay(1000);
}
```

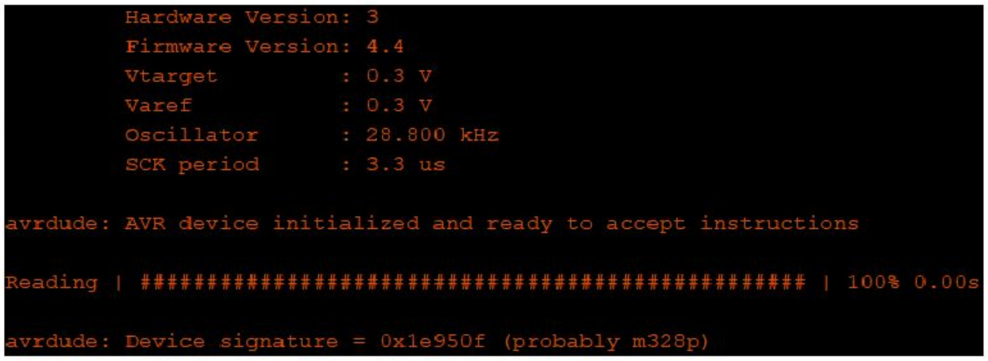
6. Upload the code by clicking on the right arrow icon or going to Sketch>Upload



Activity_1_First_Upload | Arduino 1.8.16
File Edit Sketch Tools Help
Activity_1_First_Upload
UPLOAD BUTTON

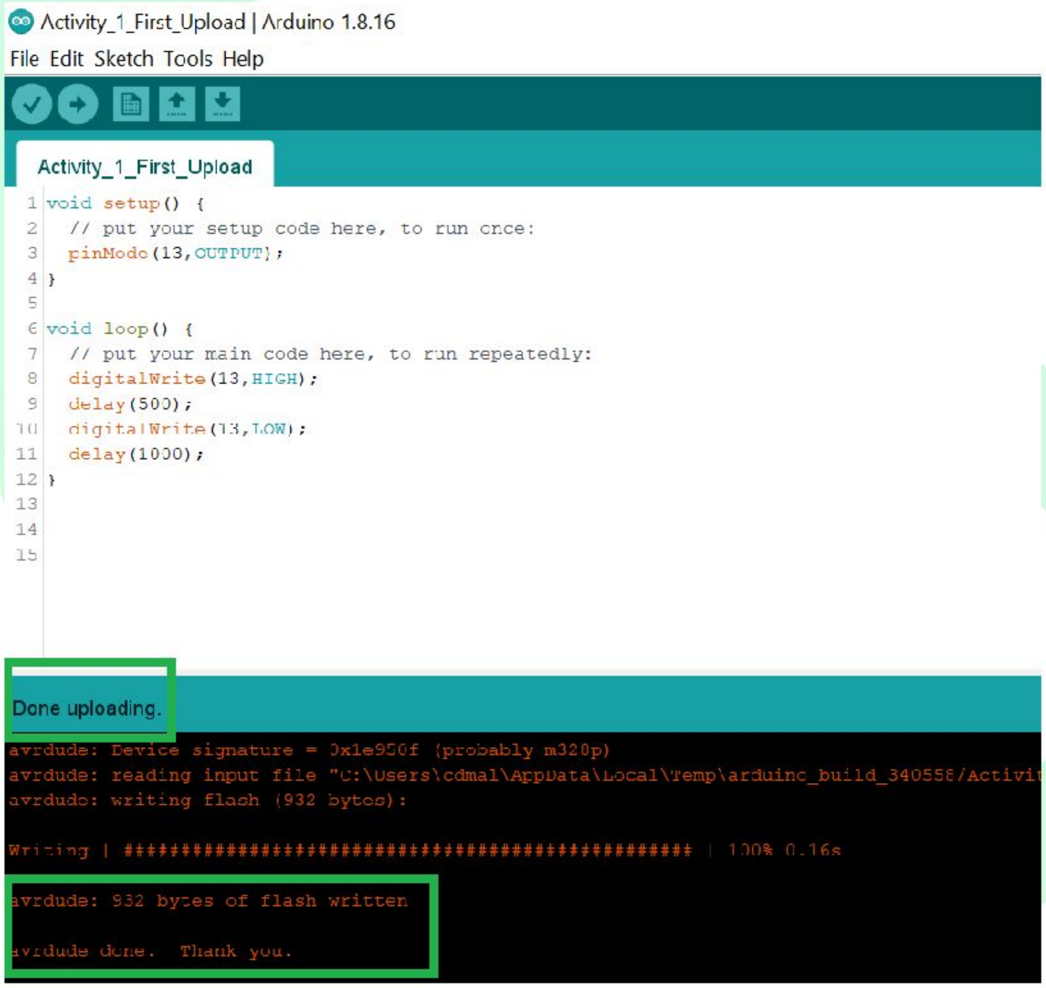
```
void setup() {
  // put your setup code here, to run onc
  pinMode(13,OUTPUT);
}

void loop() {
  // put your main code here, to run repe
  digitalWrite(13,HIGH);
  delay(500);
```

7. Wait for the upload to complete. You should see some messages at the console area during uploading



```
         Hardware Version: 3
         Firmware Version: 4.4
         Vtarget          : 0.3 V
         Varef            : 0.3 V
         Oscillator       : 28.800 kHz
         SCK period       : 3.3 us

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e950f (probably m328p)
```

8. **After a successful upload, you will see a screen similar to below**


Activity_1_First_Upload | Arduino 1.8.16
File Edit Sketch Tools Help
Activity_1_First_Upload

```
void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13,HIGH);
  delay(500);
  digitalWrite(13,LOW);
  delay(1000);
}
```

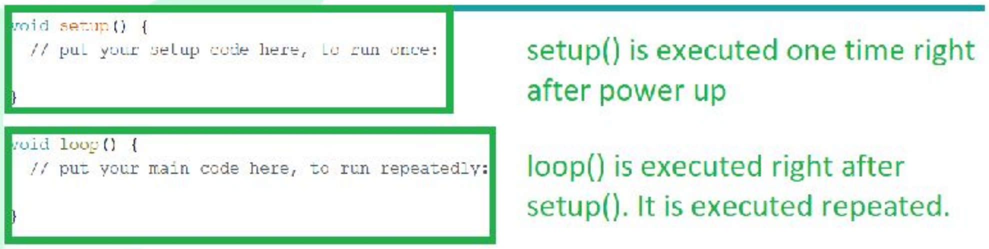
Done uploading.

```
avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: reading input file "C:\Users\cdmal\AppData\Local\Temp\arduino_build_340558/Activit
avrdude: writing flash (932 bytes):

Writing | ################################################## | 100% 0.16s

avrdude: 932 bytes of flash written

avrdude done.  Thank you.
```

9. Congratulations! You have just uploaded your first code. Now check the "L" LED on the Arduino/Saleng Uno. It will turn on for 0.5 (500 milliseconds) second and turn off for 1 second (1000 milliseconds).

BASIC CODE STRUCTURE

Writing the code follow basic rules and structure that must be followed. When a new editor window is opened, the basic elements are automatically created as in below. Here are the basic guidelines:



```
void setup() {
  // put your setup code here, to run once:

}
```

setup() is executed one time right after power up

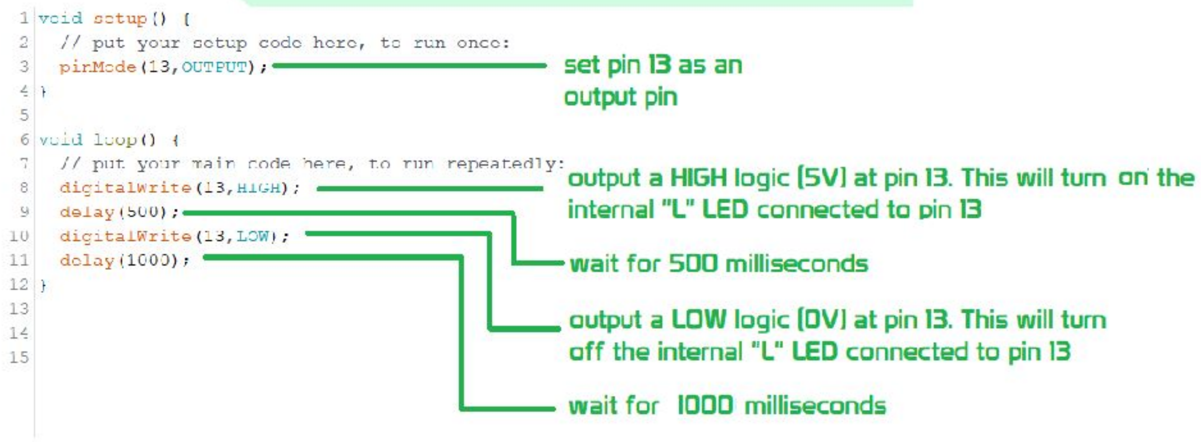
```
void loop() {
  // put your main code here, to run repeatedly:

}
```

loop() is executed right after setup(). It is executed repeated.

- The code is written and read from top to bottom, left to right.
- On power up, whatever is written inside the setup() function is executed first.
- After this , the contents of the loop() function is executed repeatedly from top to bottom and then repeating at the top.
- All other functions will be written by the author and shall be called from setup() or loop()

THE "ARDUINO LANGUAGE"

Looking back at activity 1, we could dissect the code as follows:



```
void setup() {
  // put your setup code here, to run once:
  pinMode(13,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13,HIGH);
  delay(500);
  digitalWrite(13,LOW);
  delay(1000);
}
```

set pin 13 as an output pin

output a HIGH logic (5V) at pin 13. This will turn on the internal "L" LED connected to pin 13

wait for 500 milliseconds

output a LOW logic (0V) at pin 13. This will turn off the internal "L" LED connected to pin 13

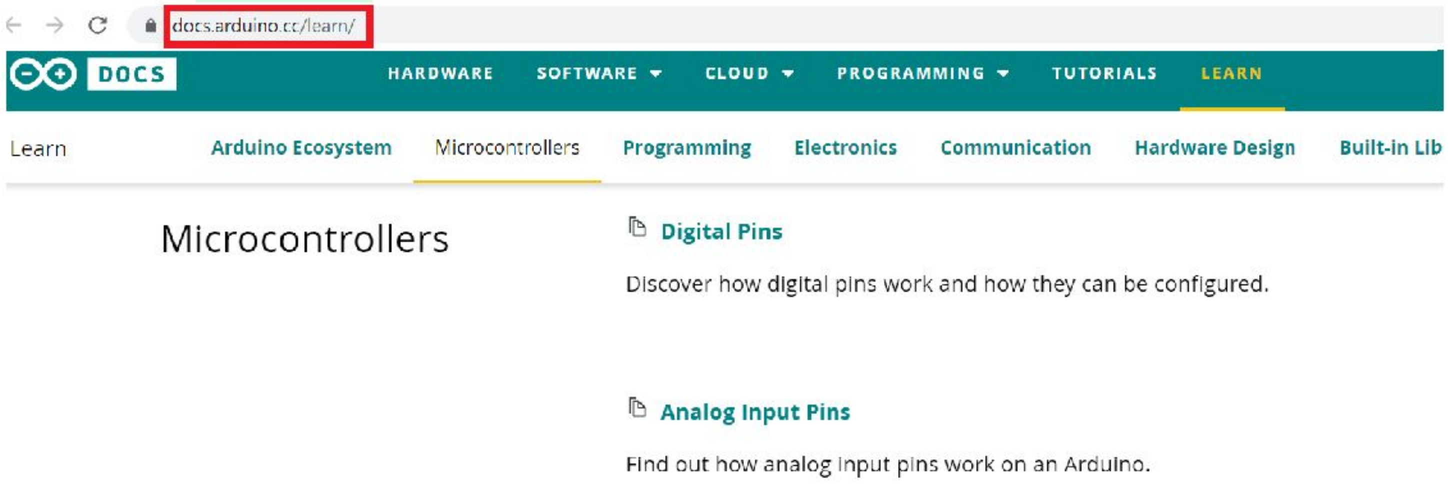
wait for 1000 milliseconds

The Arduino environment implements an abstraction layer that simplifies a lot of complexities related to microcontrollers. It also implements hardware specific functions. As such, the "Arduino" programming language is not 100% standard ANSI C or standard C++. As stated, this document does not cover the whole subject of programing and programming language. Instead, the reader may find time to research on the basic Arduino related functions:

- **pinMode(pin,direction)** – pinMode() sets an I/O pin as either an output or an input.
- **digitalWrite(pin, state)** – this sets the pin to a HIGH or LOW logic state
- **digitalRead(pin)** – returns the logic state of pin
- **analogRead(pin)** – returns the analog value of the voltage applied to pin as an integer from 0 to 1023 where 0 represents 0V and 1023 is 5V. This is applicable for pins A0 to A5 only. For extended reading, search for "Analog to Digital Conversion"
- **delay(ms)** – tells the processor to wait and do nothing for the time defined by ms in milliseconds.

For extended reading, go through the resources in docs.Arduino.cc/learn/

You may also go through https://www.arduino.cc/reference/en/ as a quick function reference



Microcontrollers

Digital Pins

Discover how digital pins work and how they can be configured.

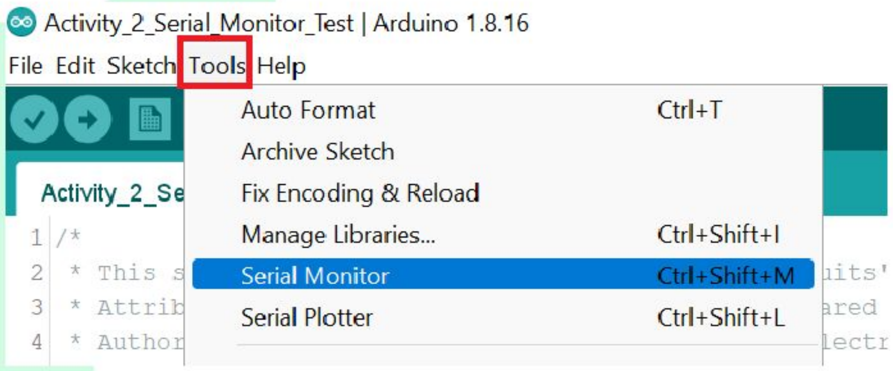
Analog Input Pins

Find out how analog input pins work on an Arduino.

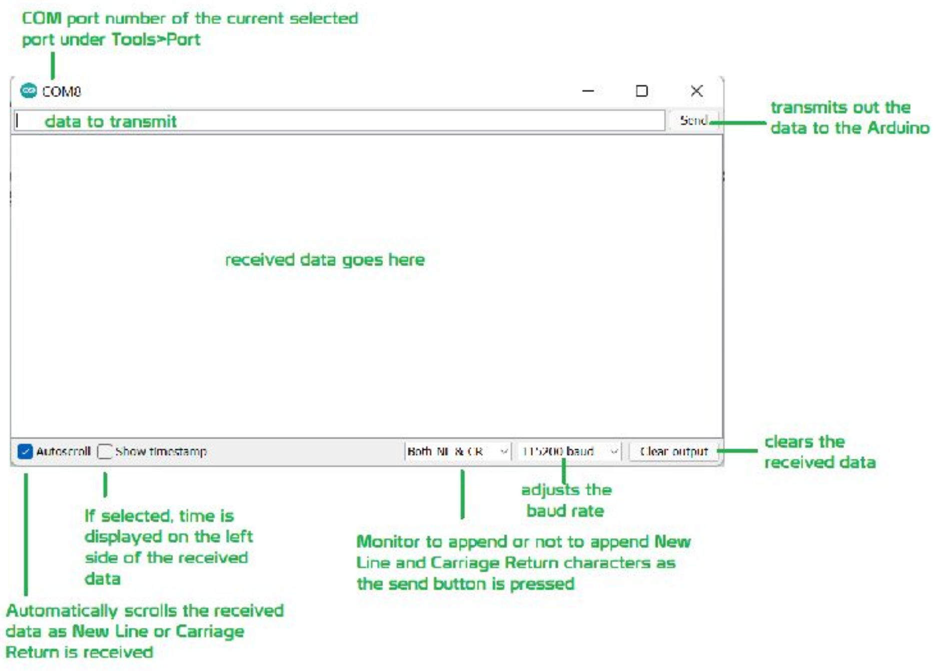
ACTIVITY 2: SERIAL MONITOR TESTING

The Arduino IDE includes a software tool called "Serial Monitor". It is used to talk to any COM port as emulated in the computer by your Arduino/Saleng Uno hardware. This gives the user the ability to communicate to the Arduino from the computer using simple text (ASCII) characters.

The Serial Monitor can be launched from tools>Serial Monitor in the Arduino IDE or with the keyboard shortcut Ctrl+Shift+M.


Activity_2_Serial_Monitor_Test | Arduino 1.8.16
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
Activity_2_Se

A new window launches as shown below:


COM port number of the current selected port under Tools>Port
COM8
data to transmit
Send
transmits out the data to the Arduino
received data goes here
Autoscroll
Show timestamp
Both NL & CR
115200 baud
Clear output
clears the received data
adjusts the baud rate
If selected, time is displayed on the left side of the received data
Monitor to append or not to append New Line and Carriage Return characters as the send button is pressed
Automatically scrolls the received data as New Line or Carriage Return is received

As an exercise, upload the code below :

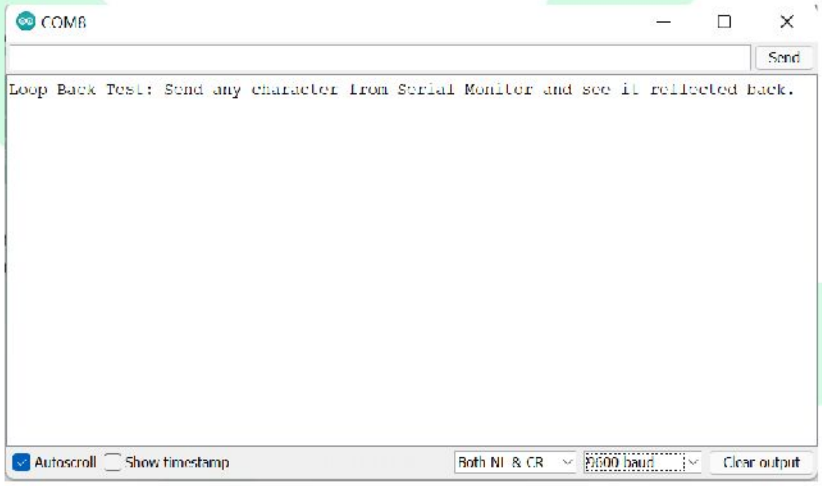
```
void setup() {
  Serial.begin(9600);
  Serial.println("Loop Back Test: Send any character from Serial Monitor and see it reflected back.");
}

void loop() {
  if(Serial.available())
  {
    Serial.write(Serial.read());
  }
}
```

Follow the steps in uploading:

REVIEW: steps in uploading sketches/code:

1. **Connect cable into Arduino and Computer**
2. **Tools>Board>Arduino AVR Boards>Arduino Uno**
3. **Tools>Port>COM port of your Arduino**
4. **Click on Upload Button**
5. **Wait until "Upload Done" id shown**

After uploading, set the baud rate of the Serial Monitor to 9600. Note that every time baud rate is changed in the Serial Monitor, the Arduino resets the code from start.



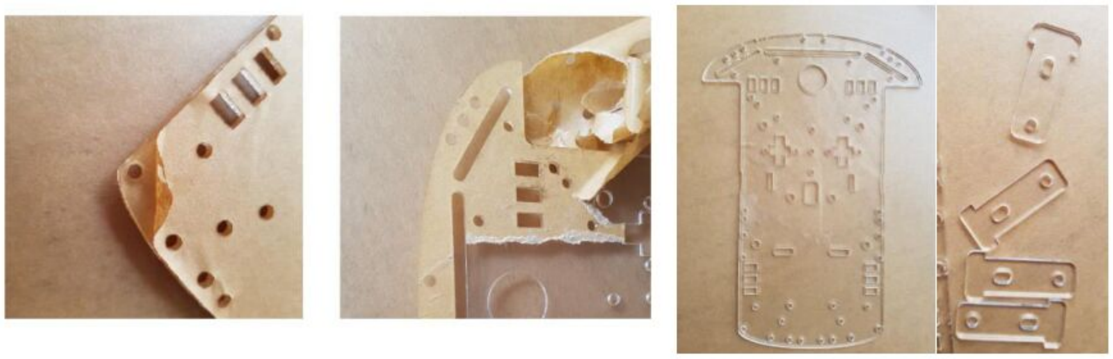
```
Loop Back Test: Send any character from Serial Monitor and see it reflected back.
```

Any data you send shall be echoed back into the received data display. This simple exercise demonstrates how the Serial Monitor may be used to communicate with the Arduino.

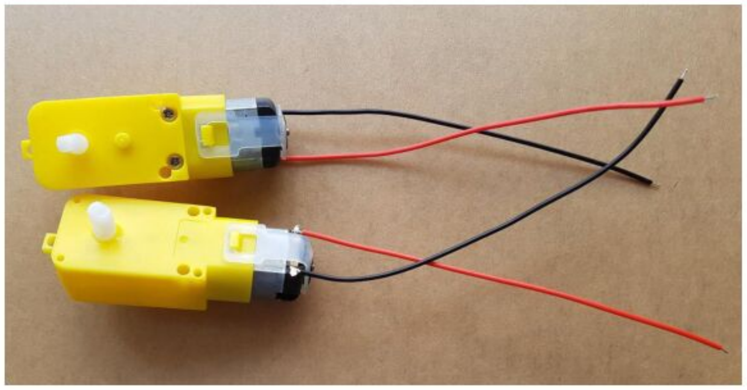
CAR ASSEMBLY

Follow these steps to assemble everything:

1. **Peel off the protective film on both sides of the Arcylic base board and motor clips. This is an optional step as some users may want to keep the protective film on. However, this will weaken the use of double sided tape as the film may give way in time.**

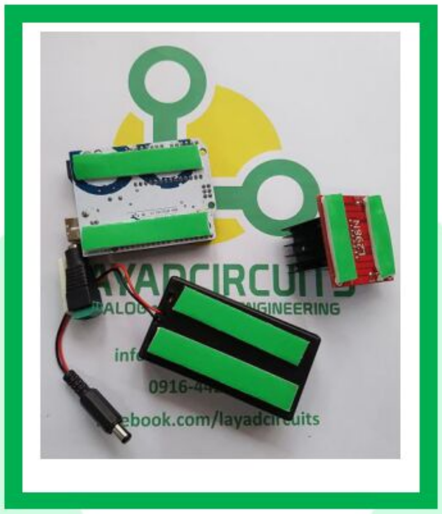


2. **In some kit versions, you will need to solder the wires into the motor terminals. If you need a kit with the wires soldered, contact Layad Circuits to customize the kit. Since the motors do not have a polarity, you may use any wire on the terminals. However, for consistency, this document will use the same color orientation below.**



3. Attach the motors and third wheel into the base board. You may refer to the assembly video from our YT channel found in this link: www.youtube.com/watch?v=L8PzoMVqsNE

   Use the brass hex standoffs to support the third wheel. Use the 4 acrylic clips to support the motors. The long screws are used to attach the motors into the acrylic clips. If you are looking for a kit that already implements this step, contact Layad Circuits for kit customization.

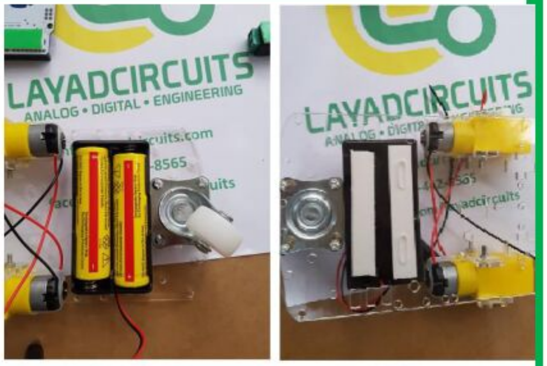


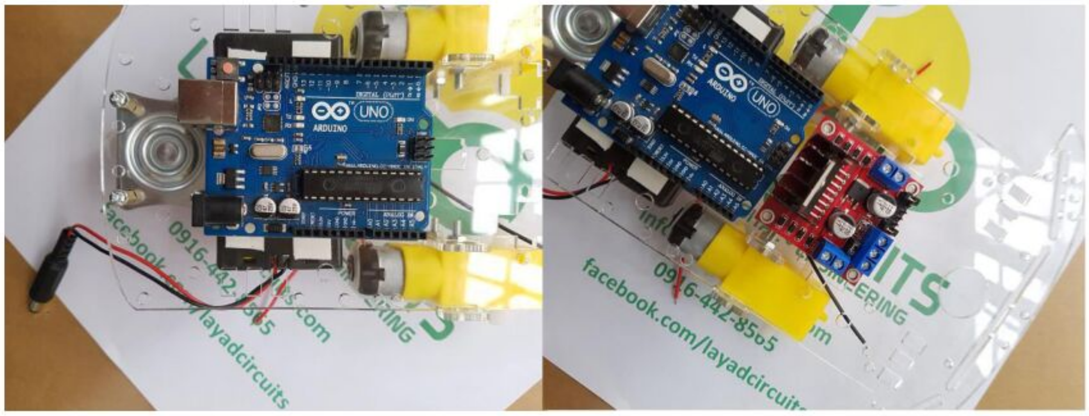
4. Apply double sided tape into the following components
   - Battery Holder
   - Arduino/Saleng Uno
   - Motor Driver
   - DC Jack adapter

5. Install the above components into the base board. Remove sticker backing as you install each component.

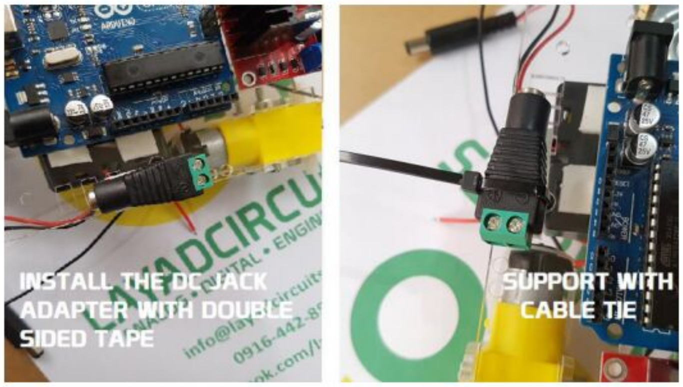
While there is no correct location of installation for the components, it is noteworthy to consider the notes below. You may also copy how the installation was done in this document by following the images in this section.

- Provide access to the Arduino Uno USB port for easy access when programming
- Provide access to the DC jack adapter
- The battery plug must have sufficient wire length to reach the DC jack as this is the method used in this kit to turn the robot car on and off.
- The wires from the motors should have sufficient length to reach the terminal blocks.

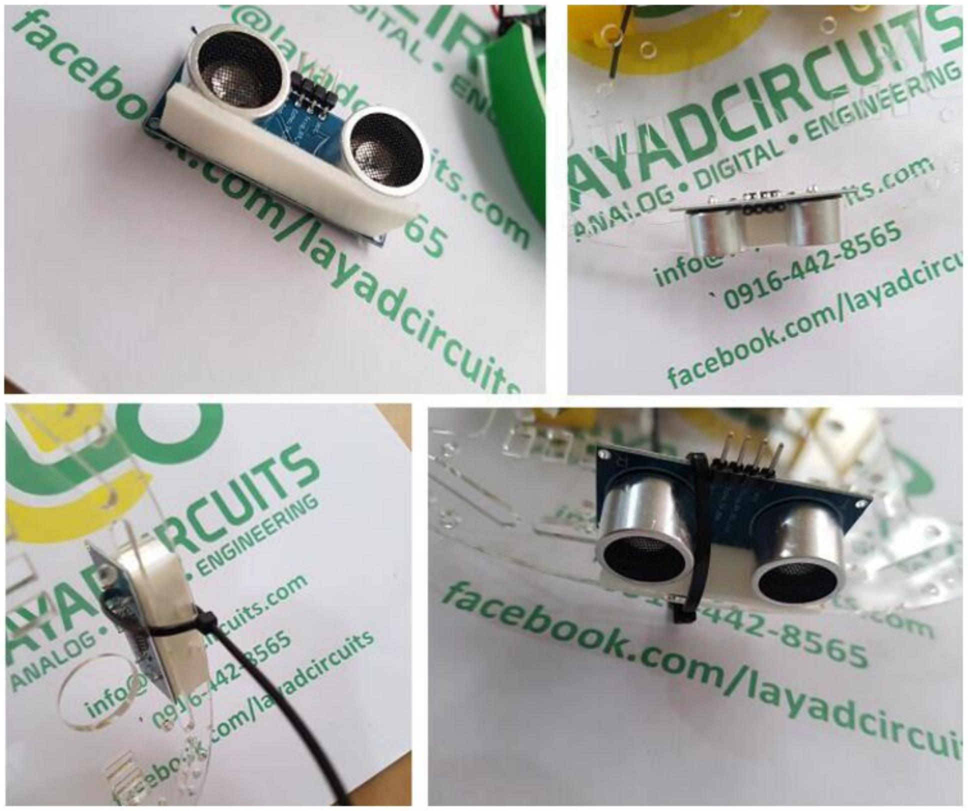




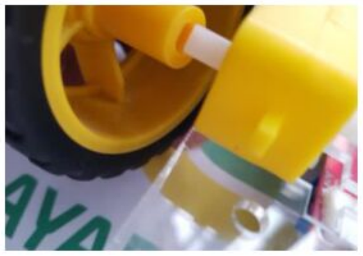
- The distance between the DC jack adapter and motor driver should be close enough for the power wire.
- In our suggested assembly, we have installed the battery holder at the bottom of the base board together with the motors and third wheel. The rest are installed on top.



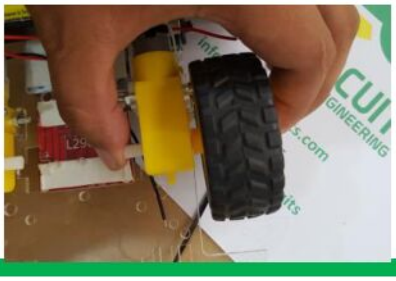
6. Install the ultrasonic sensor module into the base board, to do this, use double sided tape underneath the sensing elements (cylindrical components) and support this with a cable tie around the module and into the holes of the base board. Keep the sensor terminals on top for easy access of the connecting wires. Cut off the excess cable tie.



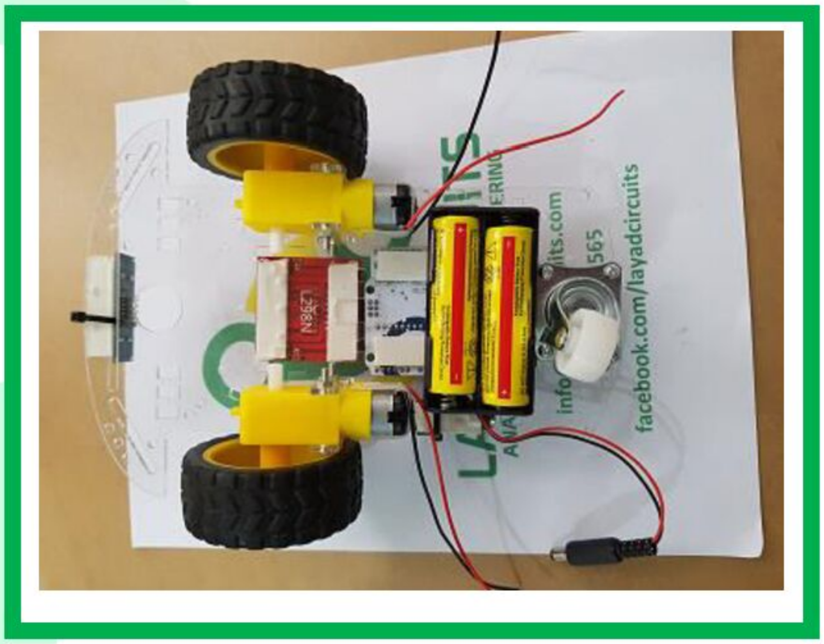
7. Attach the main wheels into the left and right motors
   - Align the wheel coupling into the motor shaft



   - Push the wheel into the shaft. For double shaft motors, support the opposite side of the shaft. Avoid excessive force to prevent damaging the shaft.



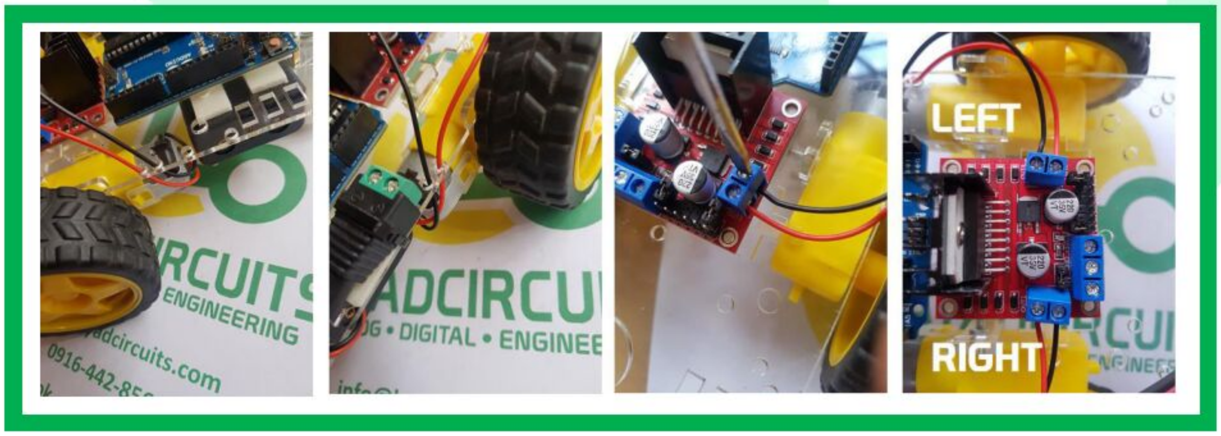
At this point, the assembly should be similar to the image below:



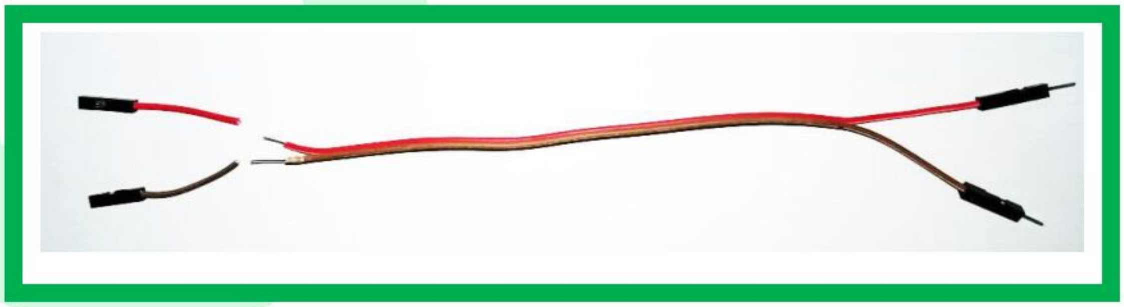
8. Insert the motor wires into available holes on the base board from the bottom side to the top side towards the terminals of the Motor Driver board. For consistency, this document will follow the table below

Motor Driver Terminals	Wire Color, Motor side
OUT1	RED, RIGHT MOTOR
OUT2	BLACK, RIGHT MOTOR
OUT3	RED, LEFT MOTOR
OUT4	BLACK, LEFT MOTOR

Use a 3mm or similarly sized flat head or Philips screw driver to connect wires into the terminal blocks



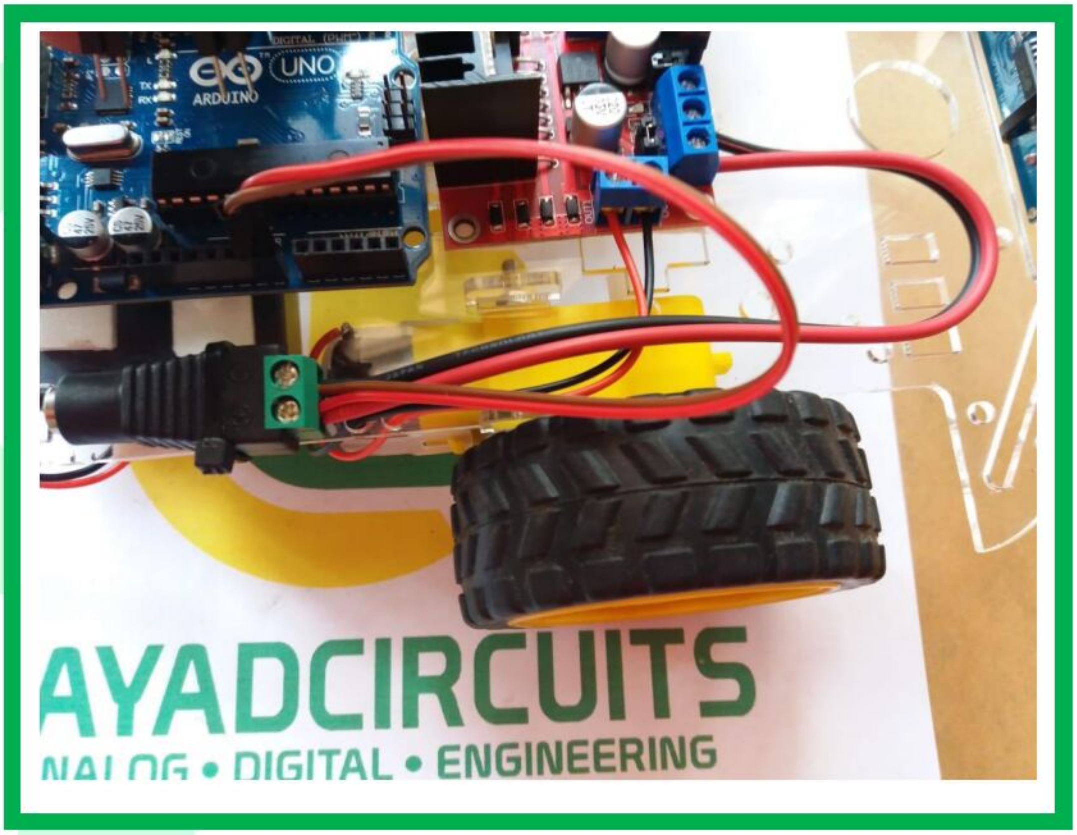
Connect the power wire and a red-brown pair from the connectors in parallel to the DC jack adapter. Strip the female header portion of the 2 connectors and strip off around 5-10mm of the insulator. This will serve as the connector for the power of the Arduino via the VIN (and GND) pin.



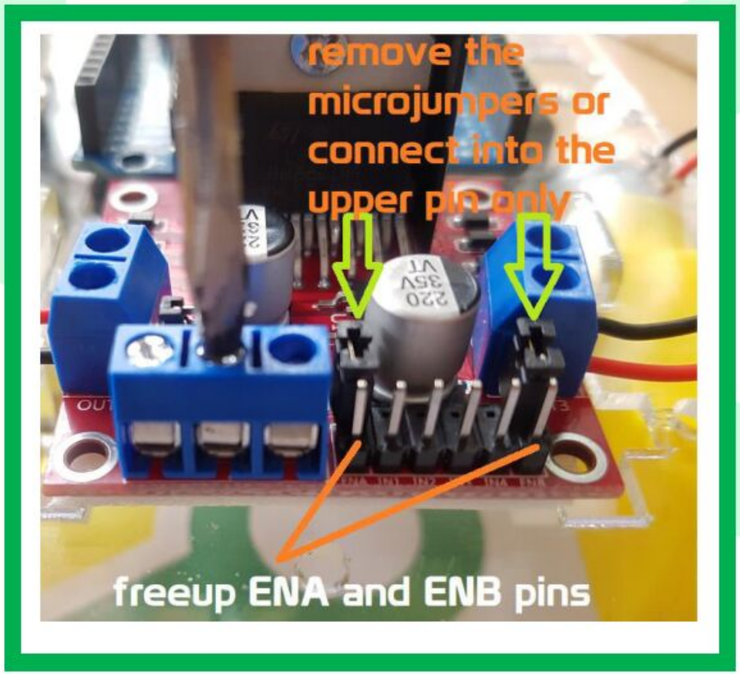
The stripped end of this 2 wires will be connected in parallel with the power cable or the DC jack connector. Follow the color conventions where the red wires connect to the positive terminal and the black/brown wires connect to the negative terminal.

The other end of the power wire goes to the 12V (+, red) and GND (- , black) terminals of the motor driver. On the other hand, the connecting wires coming from the positive side (red wire) goes into the VIN header of the Arduino. The brown connector goes to any GND pin within the Arduino. Follow the wiring diagram in the succeeding page/s.

Connect the 2 set of wires into the DC jack adapter using the screw terminals. A 3mm flathead or Philips screw driver is required for this.



9. Remove or place the microjumpers of the motor driver into one pin. We will need the ENA and ENB pins.

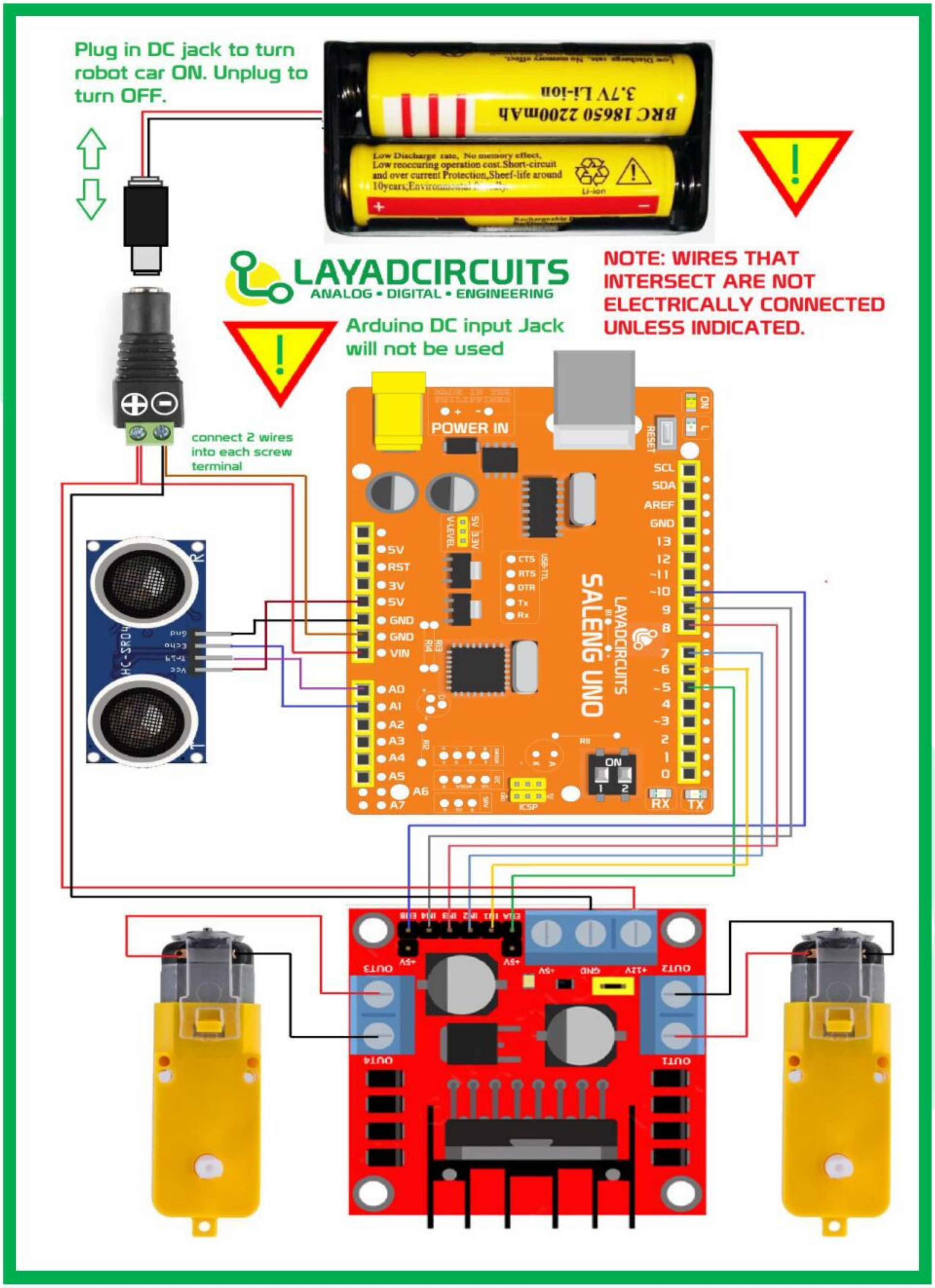


10. Connect the other connectors:
    - Motor driver IN1-IN4 and ENA and ENB to Arduino
    - Ultrasonic headers to Arduino

    Follow the wiring diagram in the succeeding page/s.

WIRING DIAGRAM

A wiring diagram is provided here as a reference for the electrical connections. The lines in the diagram represent physical wires. Lines that intersect in the diagram does not mean physical or electrical connection unless it is mentioned with text of indicated by a dot. A soft copy can be downloaded from our website at layadcircuits.com/ds/2WD_basic_kit_KIT008/


Plug in DC jack to turn robot car ON. Unplug to turn OFF.
NOTE: WIRES THAT INTERSECT ARE NOT ELECTRICALLY CONNECTED UNLESS INDICATED.
Arduino DC input Jack will not be used
connect 2 wires into each screw terminal
LAYADCIRCUITS ANALOG • DIGITAL • ENGINEERING
3.7V Li-ion
BRC 18650 2200mAh
POWER IN
5V
RST
3V
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
A6
A7
SCL
SDA
AREF
GND
13
12
~11
~10
9
8
7
~6
~5
4
~3
2
1
0
RX
TX
ICSP
SALENG UNO
Gnd
Echo
Trig
Vcc
HC-SR04
ENB IN4 IN3 IN2 IN1 ENA
+5V
GND
+12V
OUT1
OUT2
OUT3
OUT4

ACTIVITY 3: DISTANCE SENSING

At this point, the ultrasonic sensor is connected to the Arduino following the pin assignments below:

Arduino Pins	Ultrasonic Sensor Pins
5V	VCC
GND	GND
A0	TRIG
A1	ECHO

Note that although we are using the "Analog" pins A0 and A1, we shall use them as digital I/O in this exercise.

For this exercise, we do not need to connect the battery as we will not use the motors. The USB will provide sufficient power for the Arduino and ultrasonic sensor.

Upload the following code into the Arduino:

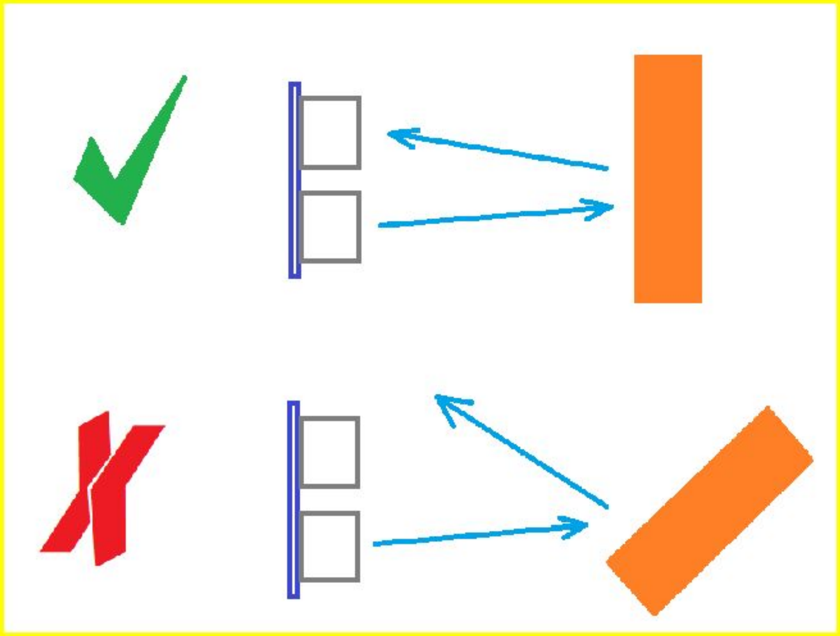
```
const byte PIN_ULTRASONIC_TRIG = A0;
const byte PIN_ULTRASONIC_ECHO = A1;
const byte PIN_LED_BUILTIN = 13;
unsigned long time_of_flight;
unsigned int distance;


void setup() {
  pinMode(PIN_ULTRASONIC_TRIG, OUTPUT);
  pinMode(PIN_ULTRASONIC_ECHO, INPUT);
  pinMode(PIN_LED_BUILTIN, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // Generate trigger pulse
  digitalWrite(PIN_ULTRASONIC_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(PIN_ULTRASONIC_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_ULTRASONIC_TRIG, LOW);

  // Measure the time of flight of the ultrasonic waves
  // We shall limit the distance to 1 meter and therefore the time to 30ms max.
  time_of_flight = pulseIn(PIN_ULTRASONIC_ECHO, HIGH, 30000);

  // Compute distance from time of flight
  // Note that the time of flight is round trip time
  // so we divide this by 2. We assume the speed of sound
  // is 346 m/sec (at 25C in air). The unit of time_of_flight
  // is in microseconds while our target output unit is centimeters.
  // Therefore we shall have:
  distance = time_of_flight*0.0346/2;

  // limit the valid distance between 2cm and our limit of 100cm
  if(distance <2 || distance >100) distance = 100;

  // send this distance  to the Serial Monitor
  Serial.print(distance);
  Serial.println(" cm");// include the unit and a new line (println())

  // turn on the builtin LED when we detect less than 30cm and
  // turn if off when we are over 30cm
  if(distance > 30) digitalWrite(PIN_LED_BUILTIN,LOW);
  else digitalWrite(PIN_LED_BUILTIN,HIGH);

  delay(100); // prevent very fast printing
}
```

To see the distance, open the Serial Monitor and set to 9600 baud rate. The sensed distance should be displayed on screen. A few things to note:

- We have set the max distance to 100cm as this is the practical distance that our robot will be working on. Whenever a distance is sensed to be longer than this, the display is set to 100cm
- When an object is too close, or when there are issues with the reflected ultrasonic wave, or if there is no return wave in the case of very far obstacles, then we sill still get 100cm. We forced this behavior in the code to prevent issues in other activities in this document.

REVIEW: Ultrasonic Ranging uses the time of flight of ultrasonic waves and uses this to compute approximate distance of the obstacle assuming the speed of sound in air. However, we should consider possible factors that will affect the accuracy of the readings:

- Ultrasonic waves may be deflected by angled obstacle. Ideally, we want a perpendicular obstacle to the sensor elements



- Certain materials may absorb or scatter ultrasonic waves, hence, the return signal may be too weak to detect
- Obstacles too close, at about 5cm or closer, may not be reliably detected by the sensor. On paper, a 2cm minimum distance is specified for the sensor. In practice that may be around 5cm. Limiting readings to a certain minimum is a good way to handle these errors.
- Other ultrasonic sources such as another sensor that is emitting ultrasonic waves within the field of view of a sensor may interfere with the readings
- Although with minimal effect, ambient environment also affects the speed of the waves. These include temperature and humidity.

All of these affect sensor readings, consider them whenever writing your code.

ACTIVITY 4: DIRECTION CONTROL

In this activity, we shall demonstrate how to control the direction of the motors. Because the activity requires the motors, the battery will be connected to test the result. However, during upload, we must keep the batter disconnected. We only connect if after the upload is completed and the USB cable is removed.

Because the motors require a lot more power that what one Arduino IO pin can offer (40mA at 5V), we will need the motor driver module to perform the heavy lifting but still controlled by the Arduino.

The following are the control pins of the L298N/L293D:

Motor Driver Pin Name	Function	Affected Motor
ENA	Enable/Disable pin – may be used for speed control of OUT1/OUT2	Right Motor
IN1	Output control pin for OUT1	Right Motor
IN2	Output control pin for OUT2	Right Motor
IN3	Output control pin for OUT3	Left Motor
IN4	Output control pin for OUT4	Left Motor
ENB	Enable/Disable pin – may be used for speed control of OUT3/OUT4	Left Motor

To control direction of a motor, the Arduino must drive the input pins following the table below

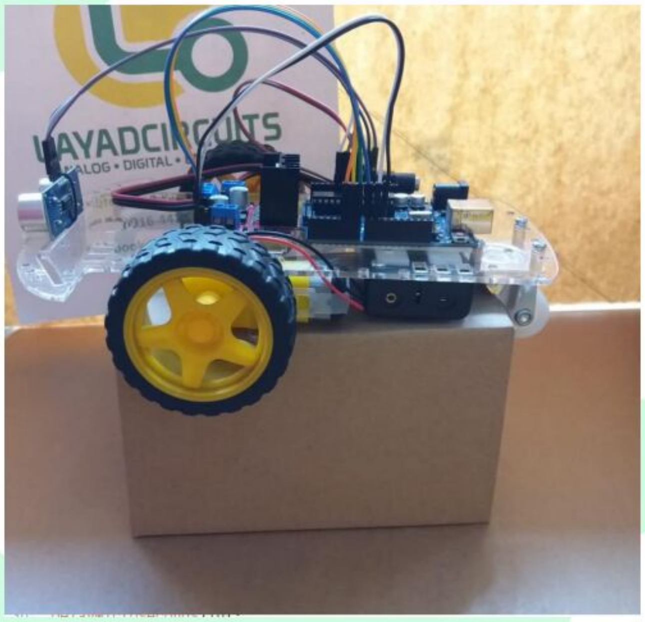
IN1	IN2	RESULTING DIRECTION OF **RIGHT** MOTOR
		FORWARD
		BACKWARD

IN3	IN4	RESULTING DIRECTION OF **LEFT** MOTOR
		FORWARD
		BACKWARD

The tables above are for the motors individually. We could combine them into a single table to show the logic we need to make the robot move forward, backward, left and right.

IN3	IN2	IN3	IN4	RESULTING DIRECTION OF WHOLE ROBOT CAR
HIGH				FORWARD
				BACKWARD
				TURNS LEFT
				TURNS RIGHT
LOW	LOW	LOW	LOW	CAR STOPS
HIGH	HIGH	HIGH	HIGH	CAR STOPS

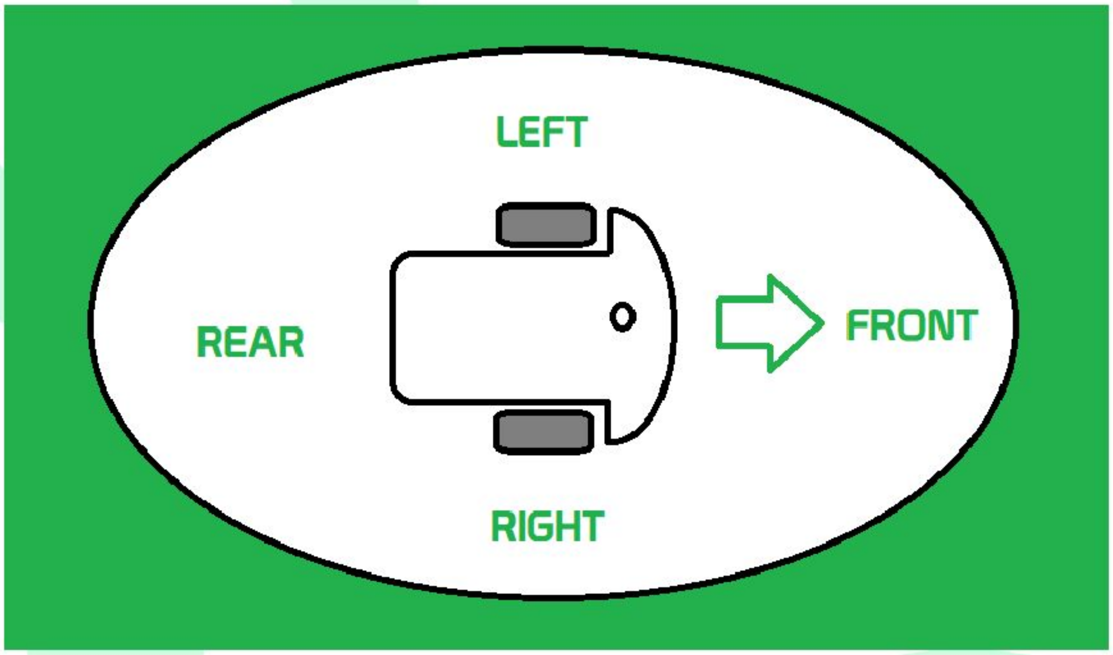
In this exercise we shall use the assembled robot car. Put an object/s underneath the car excluding the wheels. This will keep the robot from moving while we check the direction of the motors.



Disconnect the DC plug from the battery when uploading code!

Although the Arduino/Saleng Uno will have a circuit to allow both its DC jack and USB to be powered at the same time, the risk of the battery voltage entering the lower voltage of the USB bus is eliminated by using either the battery alone or the USB alone.

Through this document, we shall reference the front, left and right sides of the car following the topview diagram below:



The arc of the acrylic base board is the front side. This is the FORWARD direction. Left and right sides are referenced with respect to the robot car.

Upload the following code:

```
/*
 * This software is written in support of Layad Circuits' kit/s. It is provided for free and on an "as is" basis.
 * Attribution and this heading is requested when this software is shared or used in public.
 * Author: Layad Circuits Staff for Layad Circuits Electronics Engineering Supplies and Services
 * Contact for Support:
 *         B314 Lopez bldg Session Rd. corner Assumpton Rd., Kabayanihan Brgy.
 *         Baguio City, Benguet, Philippines
 *         +63 916 442 8565
 *         info@layadcircuits.com *
 * Copyright 2022 © Layad Circuits All Rights Reserved
 */

const byte PIN_MOTORDRIVER_ENA = 5;
const byte PIN_MOTORDRIVER_IN1 = 6;
const byte PIN_MOTORDRIVER_IN2 = 7;
const byte PIN_MOTORDRIVER_IN3 = 8;
const byte PIN_MOTORDRIVER_IN4 = 9;
const byte PIN_MOTORDRIVER_ENB = 10;
const byte PIN_ULTRASONIC_TRIG = A0;
const byte PIN_ULTRASONIC_ECHO = A1;
const byte PIN_LED_BUILTIN = 13;


void robot_stop()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN2,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN3,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN4,LOW);
}

void robot_forward()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN2,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN3,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN4,LOW);
}


void robot_backward()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN2,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN3,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN4,HIGH);
}

void robot_right()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN2,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN3,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN4,LOW);
}

void robot_left()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN2,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN3,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN4,HIGH);
}

void set_speed_leftmotor(byte val)
{
  analogWrite(PIN_MOTORDRIVER_ENB,val);
}


void set_speed_rightmotor(byte val)
{
  analogWrite(PIN_MOTORDRIVER_ENA,val);
}
```

```
void set_speed_robot(byte val)
{
  set_speed_leftmotor(val);
  set_speed_rightmotor(val);
}

void setup() {
  pinMode(PIN_MOTORDRIVER_ENA, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN1, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN2, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN3, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN4, OUTPUT);
  pinMode(PIN_MOTORDRIVER_ENB, OUTPUT);

  set_speed_robot(64); // keep the speed low to be able to observe the direction

  delay(3000); // add start up delay
}

void loop() {
  robot_forward();
  delay(2000);
  robot_stop();
  delay(1000);
  robot_backward();
  delay(2000);
  robot_stop();
  delay(1000);
}
```

Power Up the robot by inserting the DC plug from the battery into the DC jack adapter.

Notice how we encoded our truth table in the code as functions named:

- `robot_stop()`
- `robot_forward()`
- `robot_backward()`
- `robot_right()`
- `robot_left()`

In the loop(), we alternately turn the right motor forward and backward. Edit the code if you want to try this on the left motor.

The controlling the motors in one direction and shifting to the opposite direction immediately induces high currents that will stall the motor driver. Excessive currents could potentially damage the motors or the electronics. To avoid this, we first need to stop the motor, wait for a short time and then move to the opposite direction. This preventive method is shown in this exercise

ACTIVITY 5: SPEED CONTROL

As already stated in the previous exercise, we could control the speed of the robot car by applying PWM signals at the ENx pin of the motor controller.

REVIEW: Pulse Width Modulation (PWM) is one technique readiliy available in the Arduino to control power applied to a certain load , motors in this case.

The Arduino has built in PWM peripherals hardwired into 6 pins: pins **3,5,6,9,10 and 11.** These pins can generate an 8-bit PWM signal using the `analogWrite()` function.

`analogWrite()` accepts 2 parameters, the first if the pin number and the second is a value between 0 to 255 where 255 is the fastest speed (100% duty cycle) and 0 will render the motors to stop (0% duty cycle). Example, if we want to set pin 3 to 50% duty cycle, we can call this: `analogWrite(3,128);`

Upload the following code into the Arduino. Be sure the disconnect the battery before uploading.

```
/*
 * This software is written in support of Layad Circuits' kit/s. It is provided for free and on an "as is" basis.
 * Attribution and this heading is requested when this software is shared or used in public.
 * Author: Layad Circuits Staff for Layad Circuits Electronics Engineering Supplies and Services
 * Contact for Support:
 *         B314 Lopez bldg Session Rd. corner Assumpton Rd., Kabayanihan Brgy.
 *         Baguio City, Benguet, Philippines
 *         +63 916 442 8565
 *         info@layadcircuits.com *
 * Copyright 2022 © Layad Circuits All Rights Reserved
 */
const byte PIN_MOTORDRIVER_ENA = 5;
const byte PIN_MOTORDRIVER_IN1 = 6;
const byte PIN_MOTORDRIVER_IN2 = 7;
const byte PIN_MOTORDRIVER_IN3 = 8;
const byte PIN_MOTORDRIVER_IN4 = 9;
const byte PIN_MOTORDRIVER_ENB = 10;
const byte PIN_ULTRASONIC_TRIG = A0;
const byte PIN_ULTRASONIC_ECHO = A1;
const byte PIN_LED_BUILTIN = 13;


void robot_stop()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN2,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN3,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN4,LOW);
}

void robot_forward()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN2,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN3,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN4,LOW);
}


void robot_backward()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN2,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN3,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN4,HIGH);
}

void robot_right()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN2,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN3,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN4,LOW);
}

void robot_left()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN2,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN3,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN4,HIGH);
}

void set_speed_leftmotor(byte val)
{
  analogWrite(PIN_MOTORDRIVER_ENB,val);
}


void set_speed_rightmotor(byte val)
{
  analogWrite(PIN_MOTORDRIVER_ENA,val);
}


void set_speed_robot(byte val)
{
  set_speed_leftmotor(val);
  set_speed_rightmotor(val);
}

void setup() {
  pinMode(PIN_MOTORDRIVER_ENA, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN1, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN2, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN3, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN4, OUTPUT);
  pinMode(PIN_MOTORDRIVER_ENB, OUTPUT);

  delay(3000); // add start up delay

  robot_forward(); // keep the motors running forward

  set_speed_robot(90); // initialize speed a low value
}

void loop() {
  set_speed_robot(90);
  delay(1000);
  set_speed_robot(120);
  delay(1000);
  set_speed_robot(150);
  delay(1000);
  set_speed_robot(180);
  delay(1000);
  set_speed_robot(210);
  delay(1000);
  set_speed_robot(255); //max speed
  delay(1000);
  set_speed_robot(0); // motors are disabled and will stop
  delay(2000);
}
```

Power Up the robot by inserting the DC plug from the battery into the DC jack adapter.

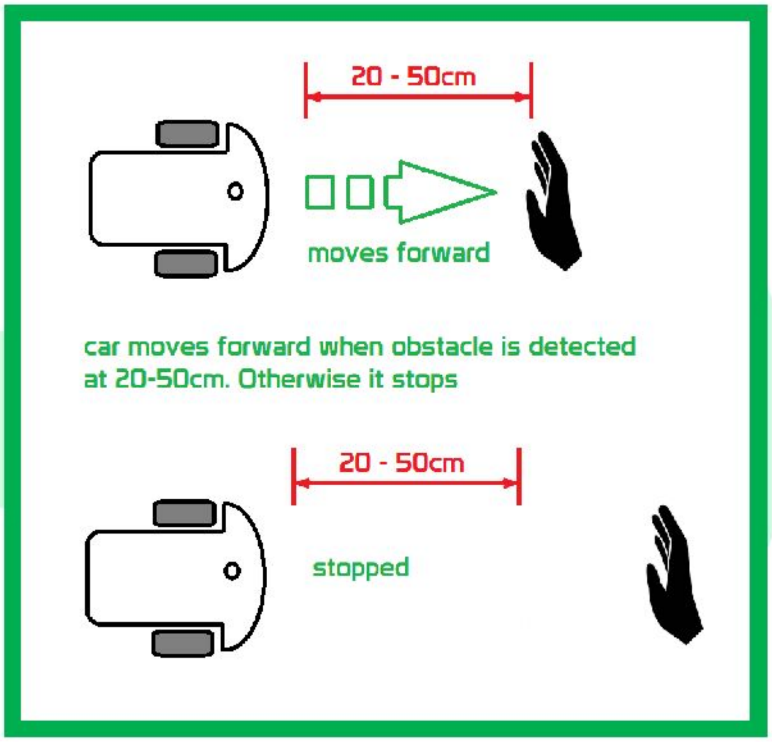
See how the speed of the wheels gradually increases and how easy this can be done with `set_speed_robot()`.

Play around with the value input in `set_speed_robot()` and observe how it affects the speed of the motor.

The PWM value of 0-255 is a scaled representation of the duty cycle. Note however that below a certain PWM value, the power applied to the motors may not be sufficient to even rotate the motors. This can vary between motors and battery charge conditions.

ACTIVITY 6: HAND FOLLOWER

In the output of this activity, the car is expected to move forward or to "follow" the palm of the user (or any obstacle) when it is presented between 20 and 50cm from the sensor. Outside of this range, the robot will stop. Only the loop() function is modified from the previous activity.


20 - 50cm
moves forward
car moves forward when obstacle is detected at 20-50cm. Otherwise it stops
20 - 50cm
stopped

Upload the following code. Keep the battery disconnected while doing the upload. Reconnect the battery plug once upload is complete to start the car.

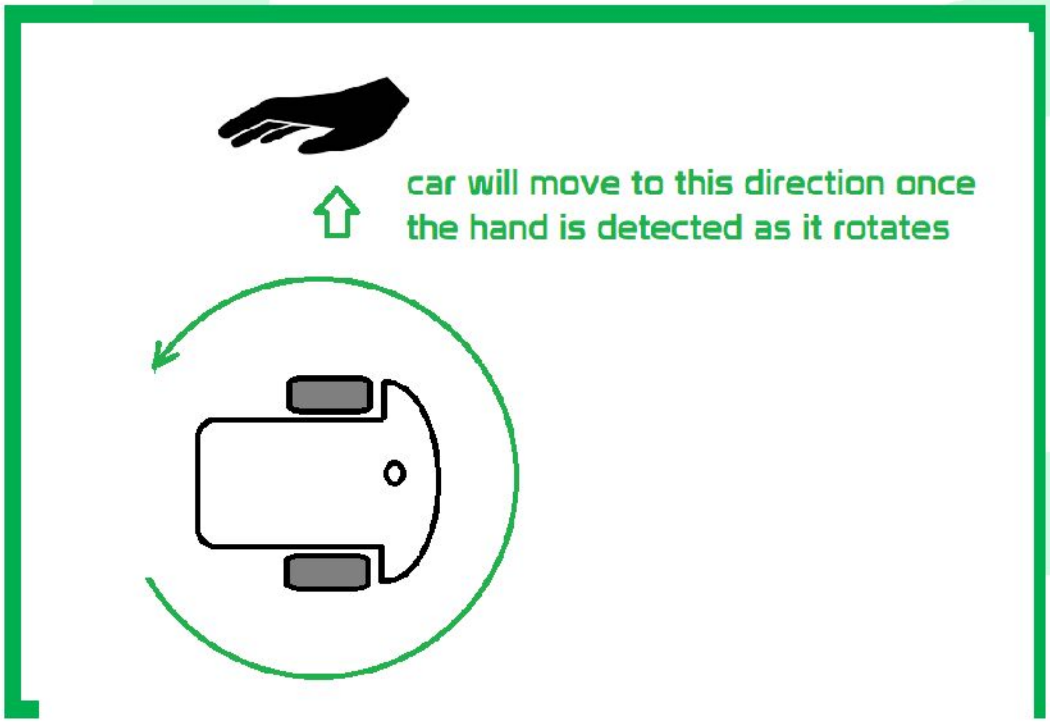
```
/*
 * This software is written in support of Layad Circuits' kit/s. It is provided for free and on an "as is" basis.
 * Attribution and this heading is requested when this software is shared or used in public.
 * Author: Layad Circuits Staff for Layad Circuits Electronics Engineering Supplies and Services
 * Contact for Support:
 *         B314 Lopez bldg Session Rd. corner Assumpton Rd., Kabayanihan Brgy.
 *         Baguio City, Benguet, Philippines
 *         +63 916 442 8565
 *         info@layadcircuits.com *
 * Copyright 2022 © Layad Circuits All Rights Reserved
 */
const byte PIN_MOTORDRIVER_ENA = 5;
const byte PIN_MOTORDRIVER_IN1 = 6;
const byte PIN_MOTORDRIVER_IN2 = 7;
const byte PIN_MOTORDRIVER_IN3 = 8;
const byte PIN_MOTORDRIVER_IN4 = 9;
const byte PIN_MOTORDRIVER_ENB = 10;
const byte PIN_ULTRASONIC_TRIG = A0;
const byte PIN_ULTRASONIC_ECHO = A1;
const byte PIN_LED_BUILTIN = 13;


void robot_stop()
{
  digitalWrite(PIN_MOTORDRIVER_IN1, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN2, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN3, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN4, LOW);
}

void robot_forward()
{
  digitalWrite(PIN_MOTORDRIVER_IN1, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN2, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN3, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN4, LOW);
}


void robot_backward()
{
  digitalWrite(PIN_MOTORDRIVER_IN1, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN2, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN3, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN4, HIGH);
}

void robot_right()
{
  digitalWrite(PIN_MOTORDRIVER_IN1, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN2, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN3, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN4, LOW);
```

```
}

void robot_left()
{
  digitalWrite(PIN_MOTORDRIVER_IN1, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN2, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN3, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN4, HIGH);
}

void set_speed_leftmotor(byte val)
{
  analogWrite(PIN_MOTORDRIVER_ENB, val);
}


void set_speed_rightmotor(byte val)
{
  analogWrite(PIN_MOTORDRIVER_ENA, val);
}


void set_speed_robot(byte val)
{
  set_speed_leftmotor(val);
  set_speed_rightmotor(val);
}

void setup() {
  pinMode(PIN_MOTORDRIVER_ENA, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN1, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN2, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN3, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN4, OUTPUT);
  pinMode(PIN_MOTORDRIVER_ENB, OUTPUT);
  pinMode(PIN_ULTRASONIC_TRIG, OUTPUT);
  pinMode(PIN_ULTRASONIC_ECHO, INPUT);

  delay(3000); // add start up delay

  set_speed_robot(90); // initialize speed at a low value
}

void loop() {
  unsigned long time_of_flight;
  int distance;
  // Read the distance
  digitalWrite(PIN_ULTRASONIC_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(PIN_ULTRASONIC_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_ULTRASONIC_TRIG, LOW);
  time_of_flight = pulseIn(PIN_ULTRASONIC_ECHO, HIGH, 30000);
  distance = time_of_flight * 0.0346 / 2;
  if (distance < 2 || distance > 100) distance = 100;

  // decide what to do. If distance is more than 30, we stop.
  //if distance is 30 or below, we move forward.

  if (distance >= 20 && distance <= 50) robot_forward();
  else robot_stop();

  delay(50);
}
```

Adjust the distance range (20 and 50) to get the desired results. Keeping a low speed will also help.

ACTIVITY 7: MODIFIED HAND FOLLOWER

This activity is very similar to the previous one where the car follows the palm of the hand. We will modify the code so that the palm can be placed anywhere around the car along the floor plane. Instead of just stopping, we could rotate the car to the right until it "sees" an obstacle to follow. Removing this obstacle within the allowable distance range will bring the robot to its rotation state.


car will move to this direction once
the hand is detected as it rotates

Upload the following code. Keep the battery disconnected while doing the upload. Reconnect the battery plug once upload is complete to start the car.

```
/*
 * This software is written in support of Layad Circuits' kit/s. It is provided for free and on an "as is" basis.
 * Attribution and this heading is requested when this software is shared or used in public.
 * Author: Layad Circuits Staff for Layad Circuits Electronics Engineering Supplies and Services
 * Contact for Support:
 *         B314 Lopez bldg Session Rd. corner Assumpton Rd., Kabayanihan Brgy.
 *         Baguio City, Benguet, Philippines
 *         +63 916 442 8565
 *         info@layadcircuits.com *
 * Copyright 2022 © Layad Circuits All Rights Reserved
 */
const byte PIN_MOTORDRIVER_ENA = 5;
const byte PIN_MOTORDRIVER_IN1 = 6;
const byte PIN_MOTORDRIVER_IN2 = 7;
const byte PIN_MOTORDRIVER_IN3 = 8;
const byte PIN_MOTORDRIVER_IN4 = 9;
const byte PIN_MOTORDRIVER_ENB = 10;
const byte PIN_ULTRASONIC_TRIG = A0;
const byte PIN_ULTRASONIC_ECHO = A1;
const byte PIN_LED_BUILTIN = 13;


void robot_stop()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN2,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN3,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN4,LOW);
}

void robot_forward()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN2,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN3,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN4,LOW);
}


void robot_backward()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN2,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN3,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN4,HIGH);
}

void robot_right()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN2,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN3,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN4,LOW);
}

void robot_left()
{
   digitalWrite(PIN_MOTORDRIVER_IN1,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN2,HIGH);
   digitalWrite(PIN_MOTORDRIVER_IN3,LOW);
   digitalWrite(PIN_MOTORDRIVER_IN4,HIGH);
}

void set_speed_leftmotor(byte val)
{
  analogWrite(PIN_MOTORDRIVER_ENB,val);
}


void set_speed_rightmotor(byte val)
{
  analogWrite(PIN_MOTORDRIVER_ENA,val);
}


void set_speed_robot(byte val)
{
  set_speed_leftmotor(val);
  set_speed_rightmotor(val);
}

void setup() {
  pinMode(PIN_MOTORDRIVER_ENA, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN1, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN2, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN3, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN4, OUTPUT);
  pinMode(PIN_MOTORDRIVER_ENB, OUTPUT);
  pinMode(PIN_ULTRASONIC_TRIG, OUTPUT);
  pinMode(PIN_ULTRASONIC_ECHO, INPUT);

  delay(3000); // add start up delay

  set_speed_robot(90); // initialize speed at a low value
}

void loop() {
  unsigned long time_of_flight;
  int distance;
  // Read the distance
  digitalWrite(PIN_ULTRASONIC_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(PIN_ULTRASONIC_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_ULTRASONIC_TRIG, LOW);
  time_of_flight = pulseIn(PIN_ULTRASONIC_ECHO, HIGH, 30000);
  distance = time_of_flight*0.0346/2;
  if(distance <2 || distance >100) distance = 100;

  // decide what to do. If distance is more than 30, we stop.
  //if distance is 30 or below, we move forward.

  if(distance >= 20 && distance <= 50) robot_forward();
  else robot_right();

  delay(50);
}
```

After upload, connect the battery and test by placing your palm (or any obstacle) any where around the robot car along the floor plane. Expect the robot to follow. Remove your hand to keep the motor rotating. In this activity, the robot car could follow the hand in any direction.

ACTIVITY 8: OBSTACLE AVOIDANCE

This activity uses the same template as previous program, only the contents of the loop() are modified. The output shall be a mobile robot car that moves to the left or right whenever an obstacle is detected. Otherwise, it moves forward.

Upload the following code. Unplug the battery before uploading.

```
/*
   This software is written in support of Layad Circuits' kit/s. It is provided for free and on an "as is" basis.
   Attribution and this heading is requested when this software is shared or used in public.
   Author: Layad Circuits Staff for Layad Circuits Electronics Engineering Supplies and Services
   Contact for Support:
           B314 Lopez bldg Session Rd. corner Assumpton Rd., Kabayanihan Brgy.
           Baguio City, Benguet, Philippines
           +63 916 442 8565
           info@layadcircuits.com
   Copyright 2022 © Layad Circuits All Rights Reserved
*/

const byte PIN_MOTORDRIVER_ENA = 5;
const byte PIN_MOTORDRIVER_IN1 = 6;
const byte PIN_MOTORDRIVER_IN2 = 7;
const byte PIN_MOTORDRIVER_IN3 = 8;
const byte PIN_MOTORDRIVER_IN4 = 9;
const byte PIN_MOTORDRIVER_ENB = 10;
const byte PIN_ULTRASONIC_TRIG = A0;
const byte PIN_ULTRASONIC_ECHO = A1;
const byte PIN_LED_BUILTIN = 13;
```

```
void robot_stop()
{
  digitalWrite(PIN_MOTORDRIVER_IN1, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN2, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN3, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN4, LOW);
}

void robot_forward()
{
  digitalWrite(PIN_MOTORDRIVER_IN1, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN2, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN3, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN4, LOW);
}


void robot_backward()
{
  digitalWrite(PIN_MOTORDRIVER_IN1, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN2, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN3, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN4, HIGH);
}

void robot_right()
{
  digitalWrite(PIN_MOTORDRIVER_IN1, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN2, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN3, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN4, LOW);
}

void robot_left()
{
  digitalWrite(PIN_MOTORDRIVER_IN1, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN2, HIGH);
  digitalWrite(PIN_MOTORDRIVER_IN3, LOW);
  digitalWrite(PIN_MOTORDRIVER_IN4, HIGH);
}

void set_speed_leftmotor(byte val)
{
  analogWrite(PIN_MOTORDRIVER_ENB, val);
}


void set_speed_rightmotor(byte val)
{
  analogWrite(PIN_MOTORDRIVER_ENA, val);
}


void set_speed_robot(byte val)
```

```
{
  set_speed_leftmotor(val);
  set_speed_rightmotor(val);
}

void setup() {
  pinMode(PIN_MOTORDRIVER_ENA, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN1, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN2, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN3, OUTPUT);
  pinMode(PIN_MOTORDRIVER_IN4, OUTPUT);
  pinMode(PIN_MOTORDRIVER_ENB, OUTPUT);
  pinMode(PIN_ULTRASONIC_TRIG, OUTPUT);
  pinMode(PIN_ULTRASONIC_ECHO, INPUT);

  delay(3000); // add start up delay

  set_speed_robot(140); // initialize speed at a low value
}

void loop() {
  unsigned long time_of_flight;
  int distance;
  // Read the distance
  digitalWrite(PIN_ULTRASONIC_TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(PIN_ULTRASONIC_TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(PIN_ULTRASONIC_TRIG, LOW);
  time_of_flight = pulseIn(PIN_ULTRASONIC_ECHO, HIGH, 30000);
  distance = time_of_flight * 0.0346 / 2;
  if (distance < 2 || distance > 100) distance = 100;

  // decide what to do. If distance is more than 30, we stop.
  //if distance is 30 or below, we move forward.

  if (distance < 20) {
    if (random(0, 2) == 0) {
      robot_right();
      delay(100);
    }
    else {
      robot_left();
      delay(100);
    }
  }
  else {
    robot_forward();
  }

  delay(50);
}
```

SPECIAL NOTES AND INSTRUCTIONS

1. **BATTERY CARE AND USAGE**
   - If the robot car does not receive power, check that the Lithium Ion batteries are in physical contact with the battery holder terminals. Move the battery along the slot to get a better contact.
   - When ever the motors are moving, the current consumption is higher hence, it could drain the batteries faster.
   - After several charge-discharge cycles, the battery capacity may wear out. In this case, replace it with an 18650 Lithium Ion battery.
   - To conserve the charge-discharge cycle, you may use an AC-DC power supply wall adapter with a voltage output between 7V and 9V and current rating of at least 2A (2000mA) and an output plug od 5.5x2.1mm. This can be used during programming/debugging when robot is not moving.
   - Each battery shall reach around 4.2V when fully charged. Therefore, expect around 8.4V at the DC plug of the holder when both batteries are installed.
   - If the batteries are to be stored for a long time, keeping the batteries at a voltage of around 3.9V helps keep it healthy while in storage. Recharge to this level if voltage drops over time.

2. **MOTOR DIRECTION**
   - Code provided in this document assumes that the user has followed the same orientation of the motor terminals.
   - Incase the terminals are different as that here, you can adjust either in the hardware or code side:
     - Hardware adjustment – swap the wire terminals in OUT1-OUT2 or OUT3-OUT4
     - Software adjustment – change the logic state of the INx pins inside the direction control functions.

3. **MOTOR MINIMUM SPEED**
   - Although you may control the motor speed using PWM, there is a minimum value at which the motor will reliably operate. Experiment and determine this value in your case as different motors and drivers may have differences in this value.

4. **MOTOR SPEED DIFFERENCE**

- The motors provided in this kit are basic DC motors which may come with speed differences at lower PWM values. Avoid this situation by using higher PWM values in speed control code.
- You may also tune the PWM value between left and right motor to match the speed. Note that the difference may change as you adjust the speed.

5. UPLOAD ISSUES

- Common causes of failed uploads are listed here:
  - Wrong selection in Tools>Board
  - Wrong selection in Tools>Port
  - Port is not detected by the computer – install USB driver or use another USB port
  - CH341 driver has just been installed – reboot your computer
  - Pins 0 and 1 are connected to a circuit – remove the circuit during upload

6. REFERENCES and TROUBLESHOOTING

- The official Arduino website (Arduino.cc) is one of the best references for Arduino. There are several topics under it that is worth reading.
- For inquiries on this kit, pls contact us by mobile at 09164428565, email at info@layadcircuits.com, or at our official FB page at facebook.com/layadcircuits
- For troubleshooting, refer to common problems discussed in the website: https://www.arduino.cc/en/pmwiki.php?n=guide/troubleshooting

IMPORTANT NOTICE