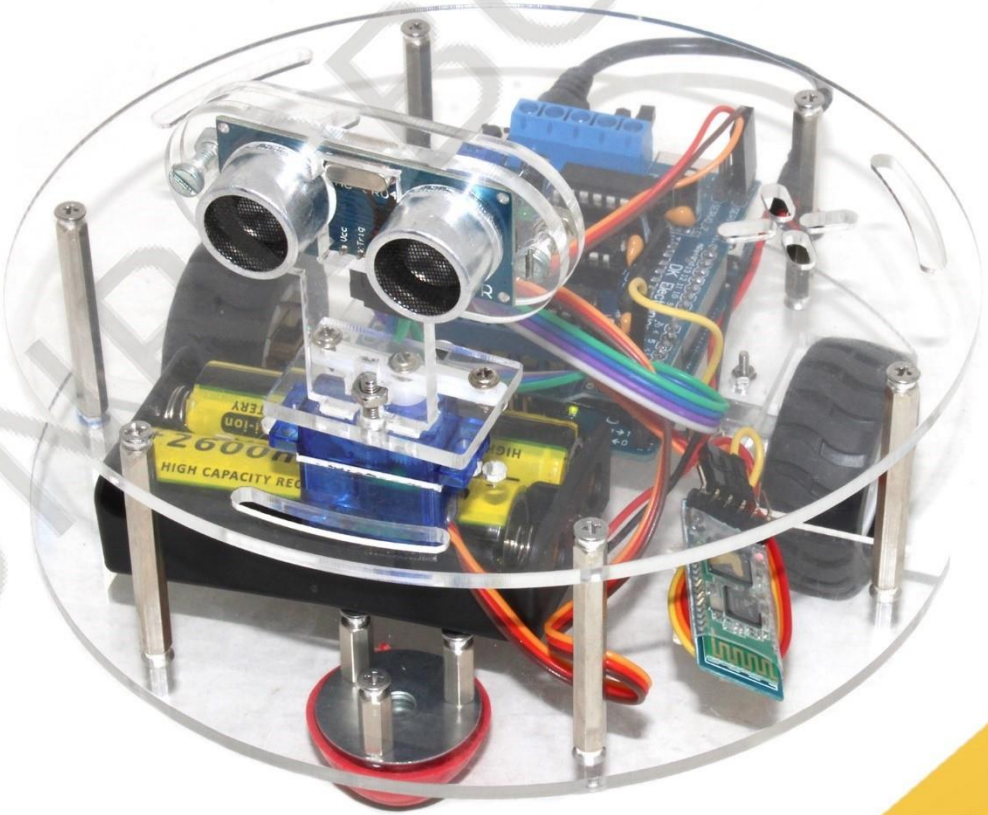
Circle Bot Mini DIY Robotics kit



SUNROBOTICS

www.sunrobotics.in

SunRobotics Technologies
C-301, Sumel Business Park 6, Dudheshwar Road, Ahmedabad-380004, Gujarat-India
Ph no: 079 26608128, 95588-27732 Email: support@sunrobotics.in

Preface

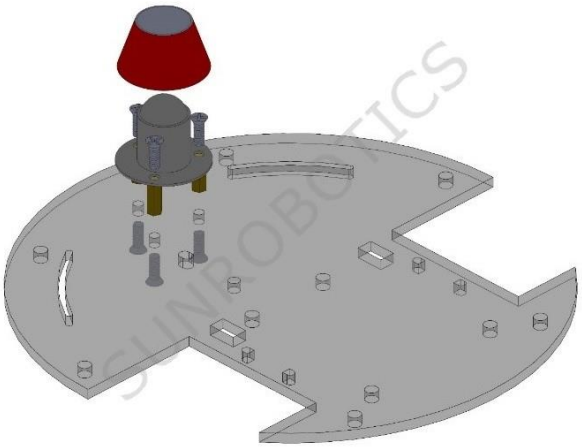
Sun Robotics is a technical service team of open source software and hardware. Dedicated to applying the Internet and the latest industrial technology in open source area, we strive to provide best hardware support and software service for general makers and electronic enthusiasts around the world. We aim to create infinite possibilities with sharing. No matter what field you are in, we can lead you into the electronic world and bring your ideas into reality.

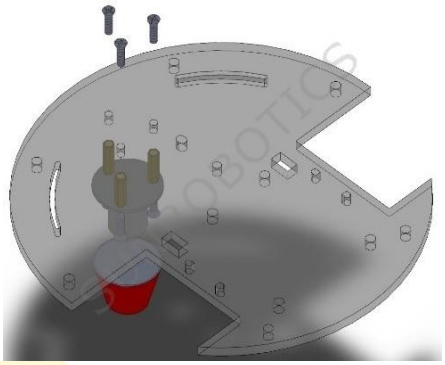
This is a Professional Level kit for Arduino. Some common electronic components and sensors are included. Through the learning, you will get a better understanding of Arduino, and be able to make fascinating works based on Arduino.

Contents

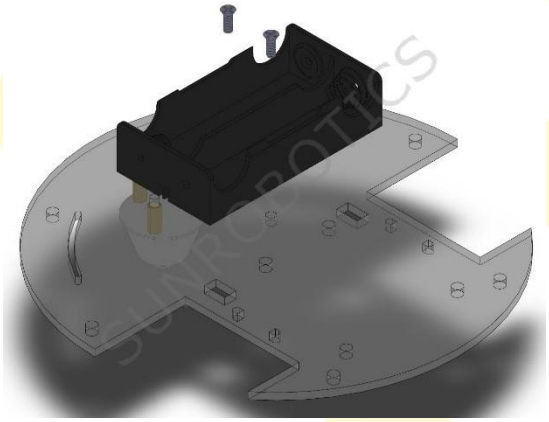
Connect all Modules and Chassis

1. Ball Caster wheel place on Lower Plat-form

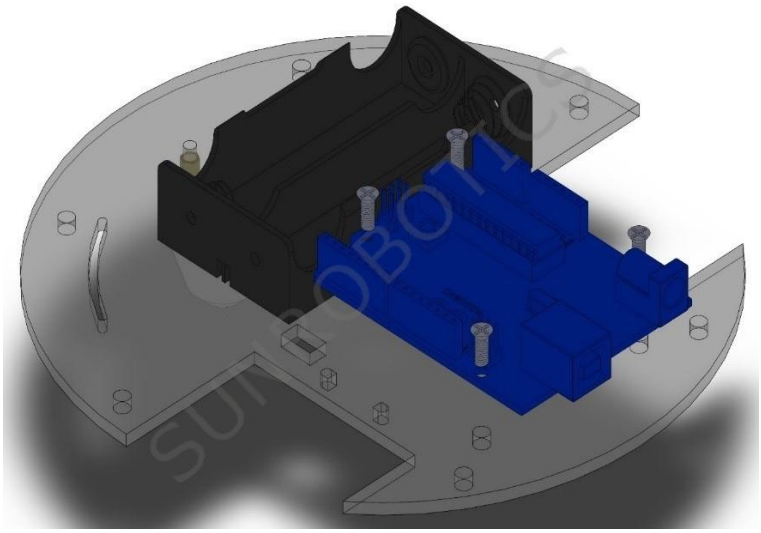




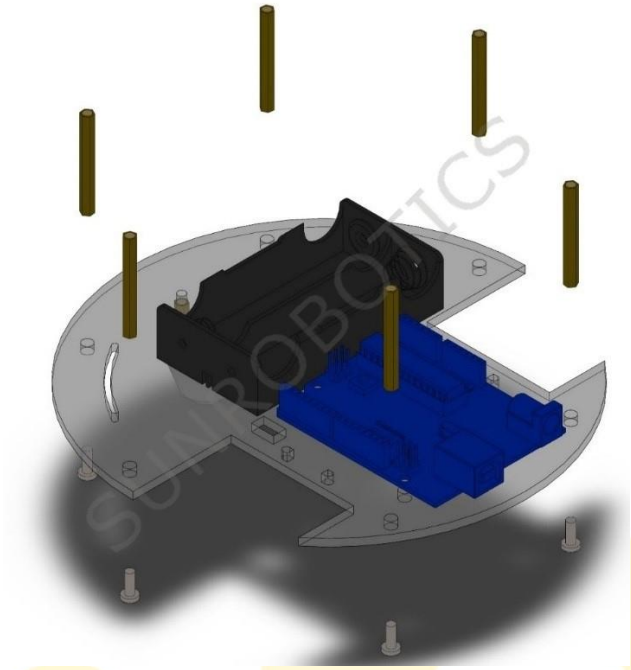
2. Place Battery Holder on Lower Plat-form



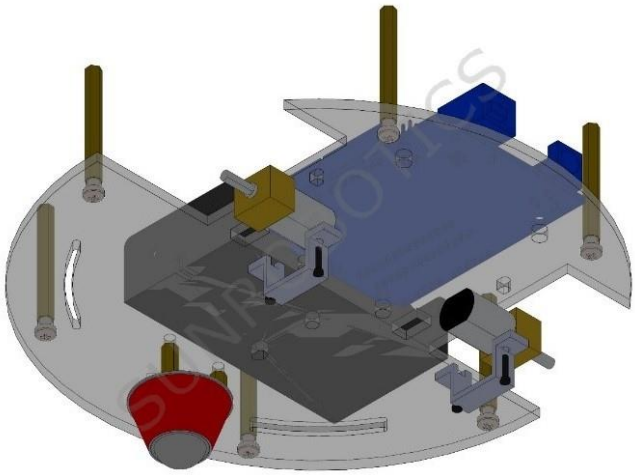
3. Place Arduino Board



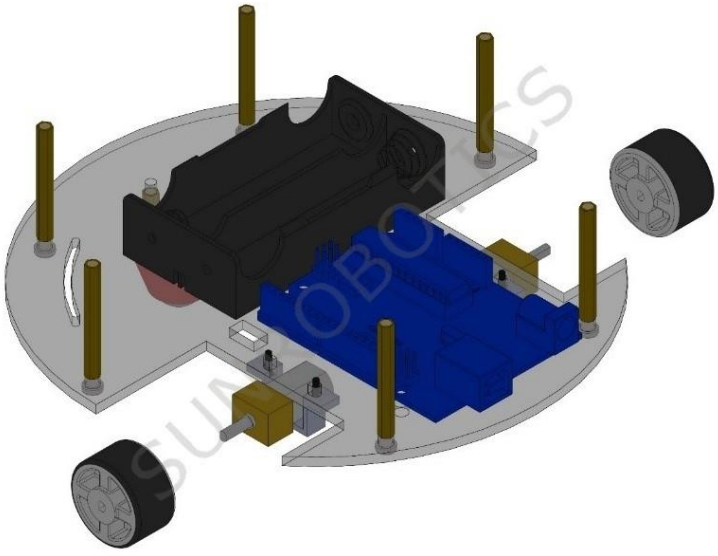
4. Place stud for upper platform



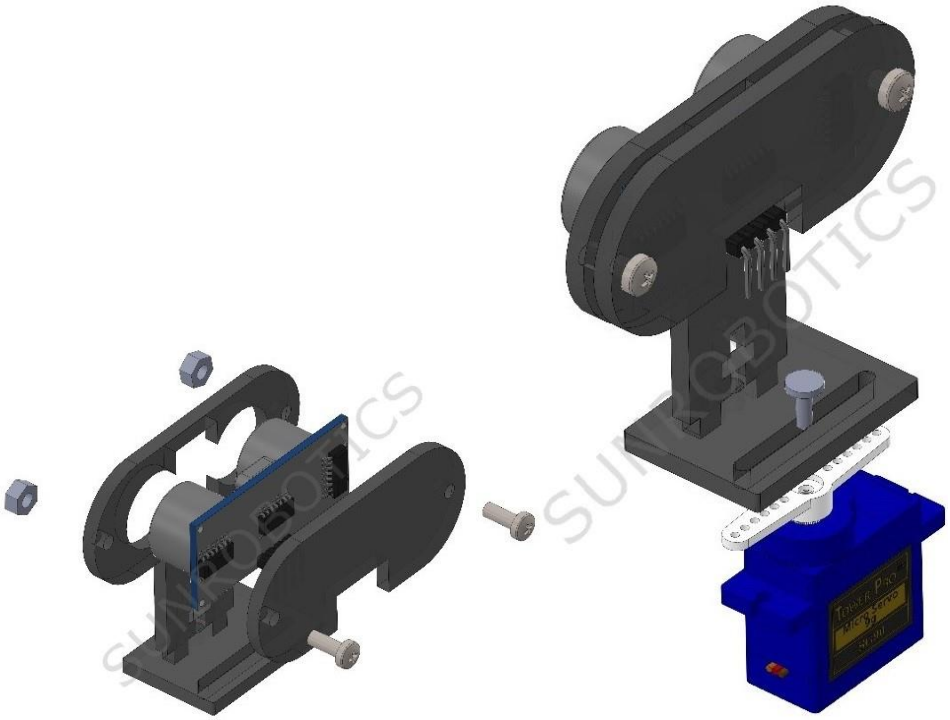
5. Place DC Motors



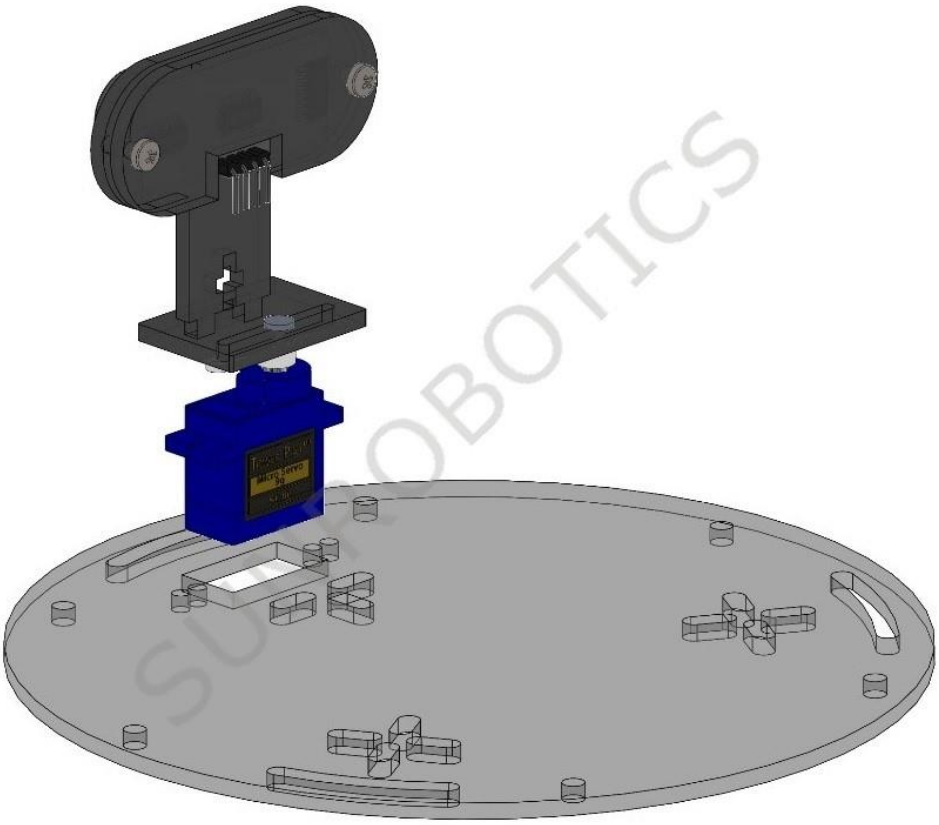
6. Place DC Motors wheels



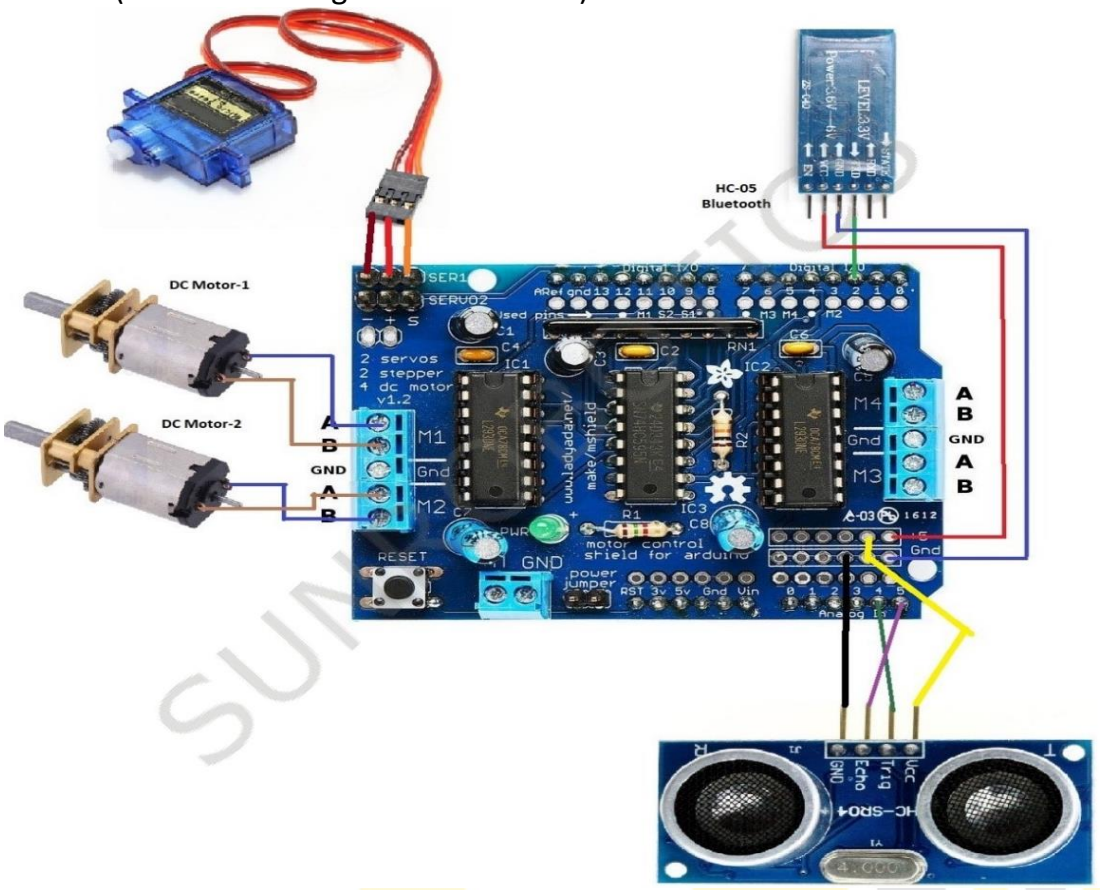
7. Place Ultrasonic Distance Sensor in its Chassis and place on Servo Motor

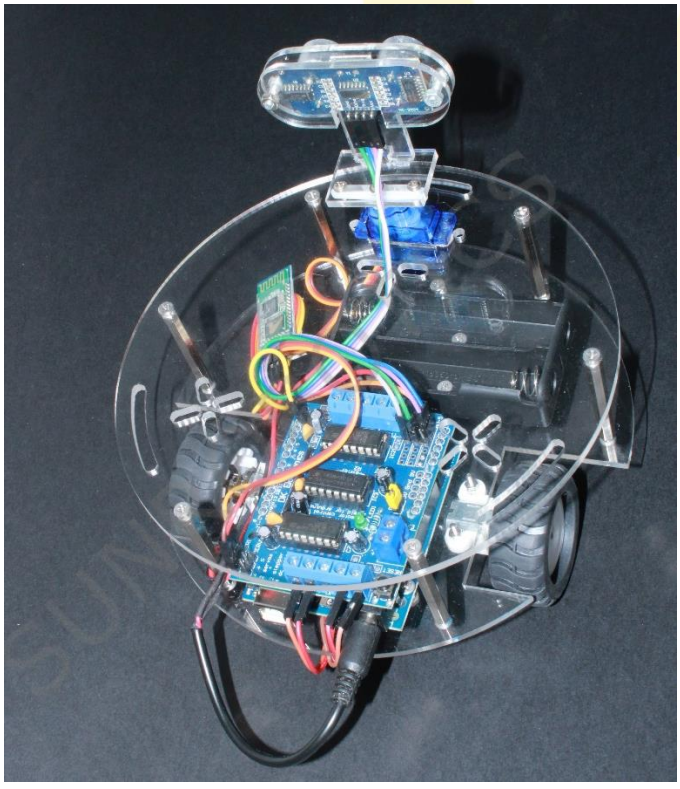


8. Place Ultrasonic Distance Sensor and Servo Motor Combine module on Upper Plat form disk

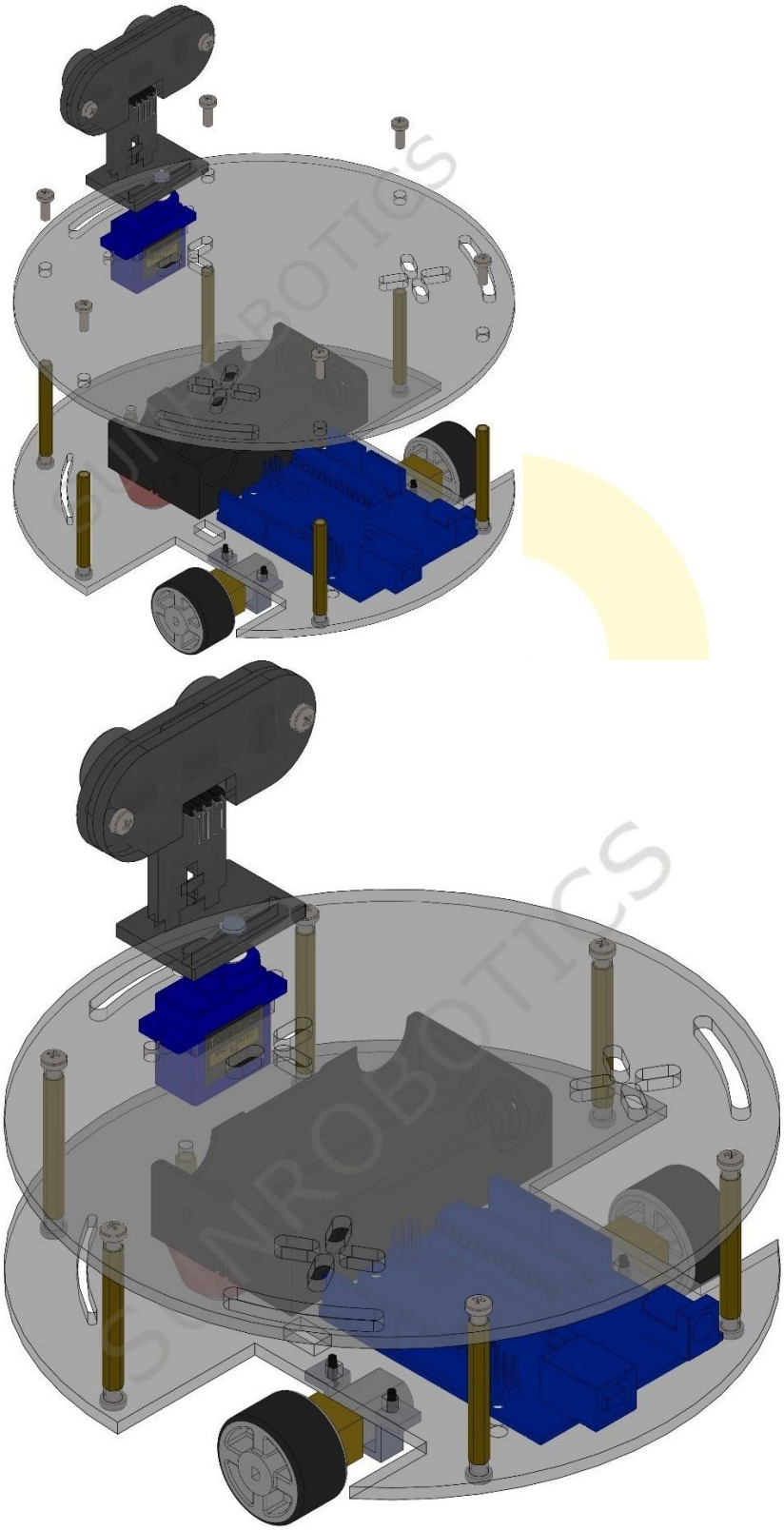


9. L293D Motor Driver Shield Place on Arduino Board connect Bluetooth module, two DC motors wire, servo motors wire and Ultrasonic Distance sensor wire(follow this image for connection)





10. Place Upper Plat form disk to Lower Plat-form



Getting Started with Arduino

What is an Arduino?

Arduino is an open-source physical computing platform designed to make experimenting with electronics more fun and intuitive. Arduino has its own unique, simplified programming language, a vast support network, and thousands of potential uses, making it the perfect platform for both beginner and advanced DIY enthusiasts.

www.arduino.cc

A Computer for the Physical World:

The friendly blue board in your hand (or on your desk) is the Arduino. In some ways you could think of Arduino as the child of traditional desktop and laptop computers. At its roots, the Arduino is essentially a small portable computer. It is capable of taking inputs (such as the push of a button or a reading from a light sensor) and interpreting that information to control various outputs (like a blinking LED light or an electric motor). That's where the term "physical computing" is born - an Arduino is capable of taking the world of electronics and relating it to the physical world in a real and tangible way. Trust us - this will all make more sense soon.

Arduino UNO SMD R3

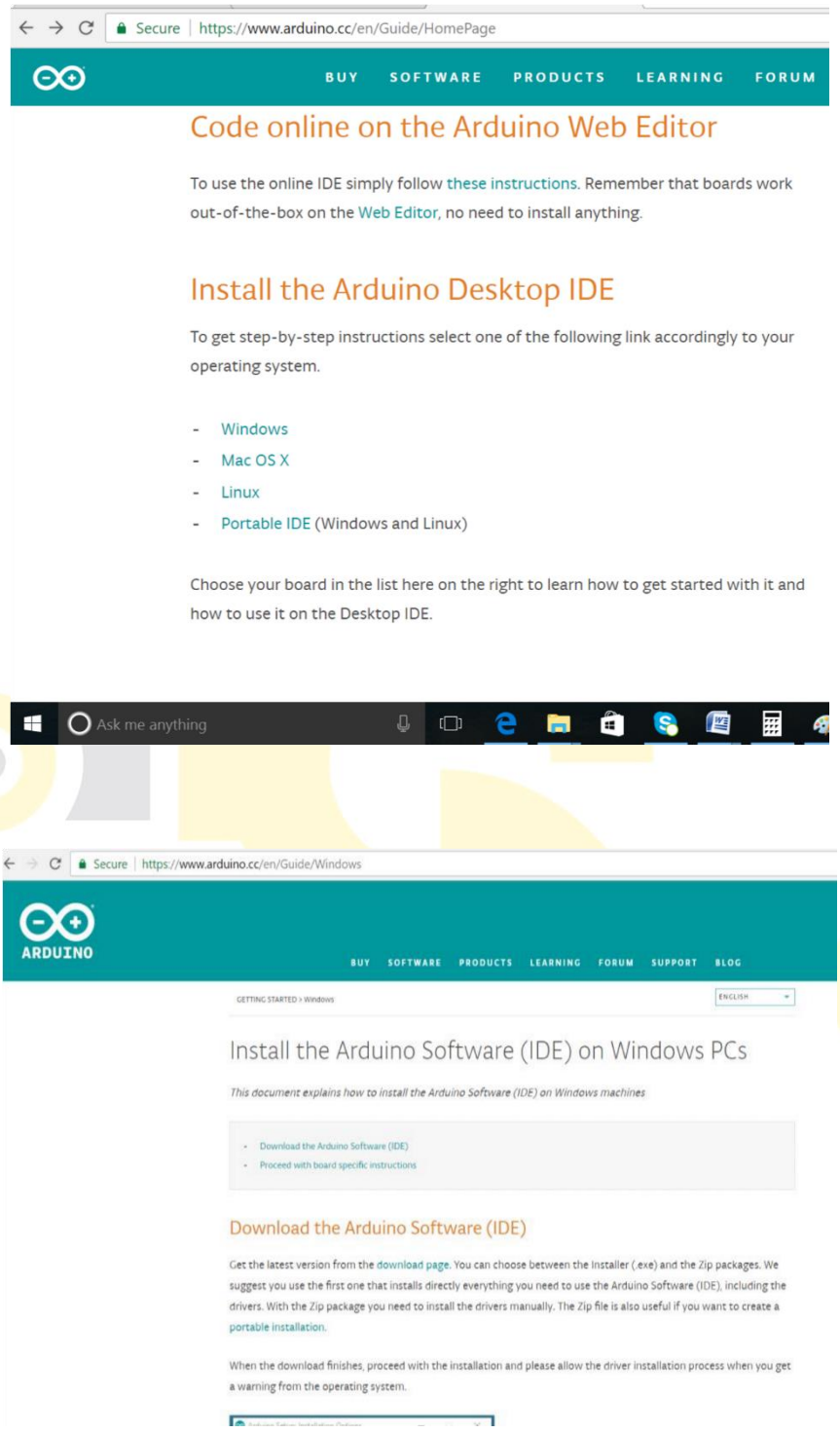
The Arduino Uno is one of several development boards based on the ATmega328. We like it mainly because of its extensive support network and its versatility. It has 14 digital input/output pins (6 of which can be PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. Don’t worry, you’ll learn about all these later.

Installing Arduino IDE and Using Uno R3 board

STEP-1: Download the Arduino IDE (Integrated Development Environment)

Access the Internet: In order to get your Arduino up and running, you'll need to download some software first from www.arduino.cc (it's free!). This software, known as the Arduino IDE, will allow you to program the Arduino to do exactly what you want. It’s like a word processor for writing programs. With an internet-capable computer, open up your favorite browser and type in the following URL into the address bar:

www.arduino.cc/en/Main/Software



For different operating system platforms, the way of using Arduino IDE is different. Please refer to the following links: Windows User ： http://www.arduino.cc/en/Guide/Windows Mac

OS
User:http://www.arduino.cc/en/Guide/MacOSXLinuxUserhttp://playground.arduino.cc/Learning/Linux For more detailed information about Arduino IDE, please refer to the following link: **http://www.arduino.cc/en/Guide/HomePage**

STEP-2: Connect your Arduino Uno to your Computer:

Use the USB cable provided in the kit to connect the Arduino to one of your computer's USB inputs.

STEP-3: Install Drivers

Depending on your computer's operating system, you will need to follow specific instructions. Please go to the URLs below for specific instructions on how to install the drivers onto your Arduino Uno.

Windows Installation Process: Go to the web address below to access the instructions for installations on a Windows-based computer.

http://arduino.cc/en/Guide/Windows

Macintosh OS X Installation Process: Macs do not require you to install drivers. Enter the following URL if you have questions. Otherwise proceed to next page.

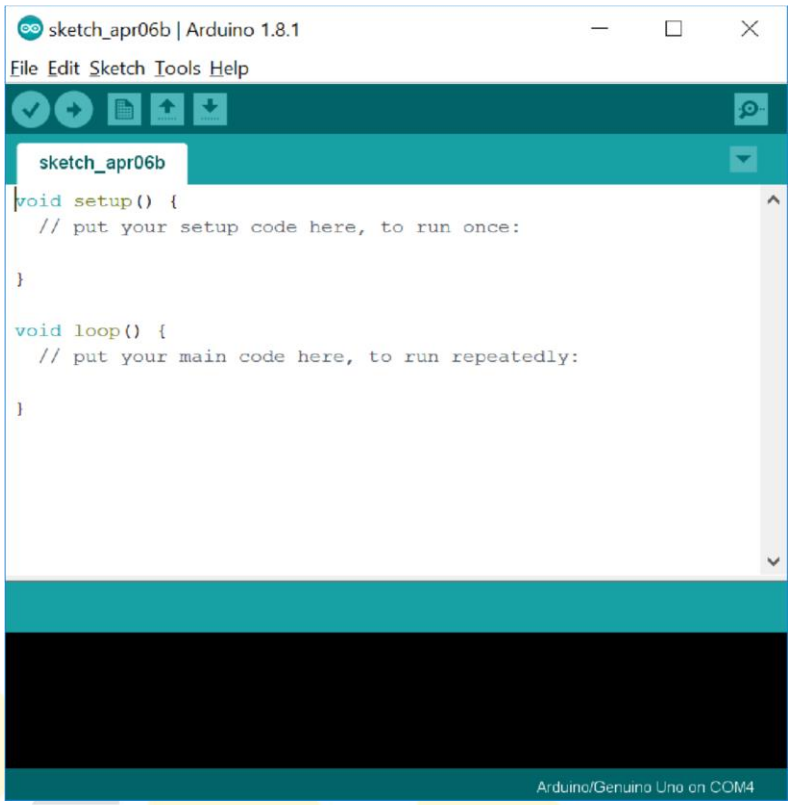
http://arduino.cc/en/Guide/MacOSX

Linux: 32 bit / 64 bit, Installation Process Go to the web address below to access the instructions for installations on a Linux-based computer.

http://www.arduino.cc/playground/Learning/Linux

STEP-4: Open the Arduino IDE

Open the Arduino IDE software on your computer. Poke around and get to know the interface. We aren't going to code right away, this is just an introduction. The step is to set your IDE to identify your Arduino Uno.



GUI (Graphical User Interface)

Verify

Checks your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board. See uploading below for Details .Note: If you are using an external programmer with your board, you can hold Down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer" New Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within The current window overwriting its content. Note: due to a bug in Java, this menu Doesn’t scroll; if you need to open a sketch late in the list, use the File | Sketch book Menu instead.

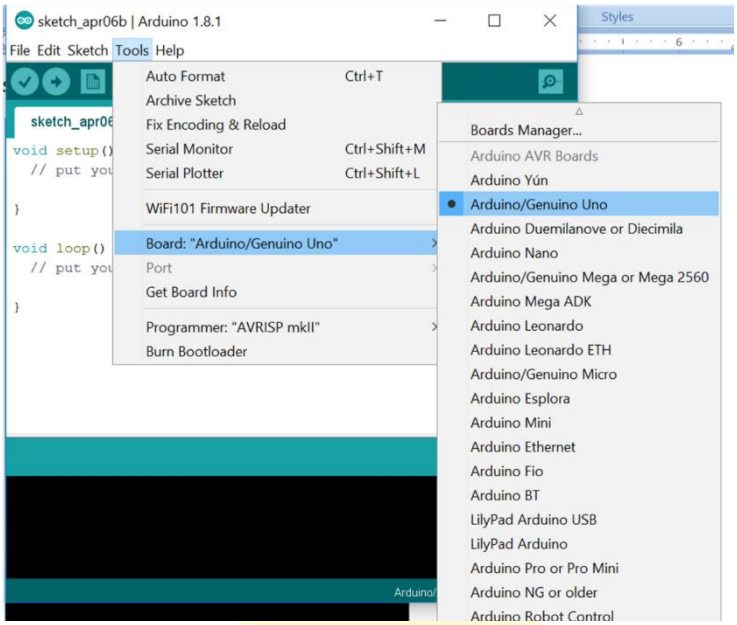
Save

Saves your sketch.

Serial Monitor

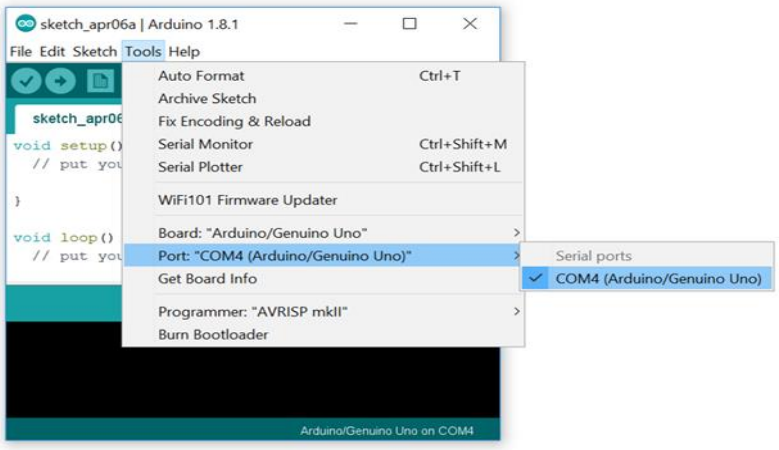
Opens the serial monitor.

STEP-5: Select your board: Arduino Uno



STEP-6: Select your Serial Device

Windows: Select the serial device of the Arduino board from the Tools | Serial Port menu. This is likely to be com3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.



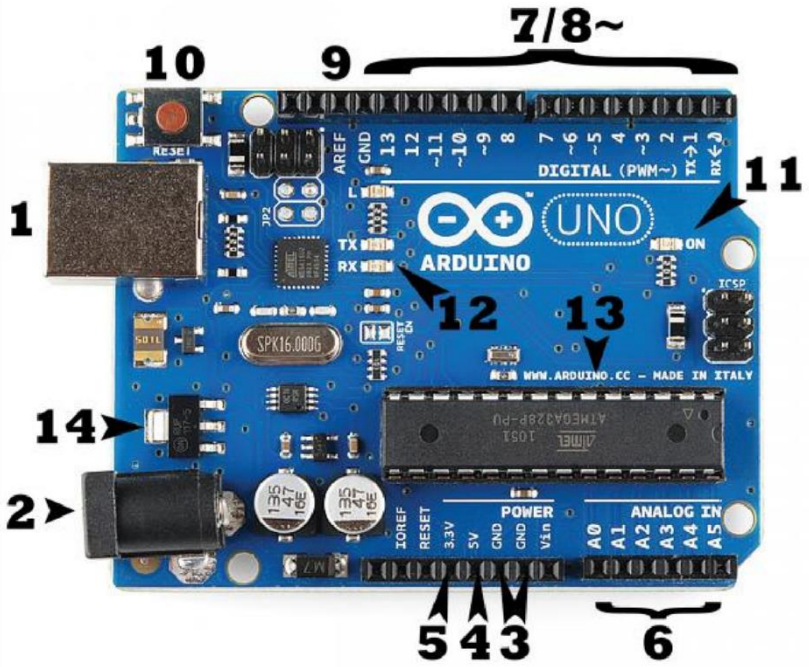
Mac OS: Select the serial device of the Arduino board from the Tools > Serial Port menu. On the Mac, this should be something with /dev/tty.usbmodem (for the Uno or Mega 2560) or /dev/tty.usbserial (for older boards) in it.

Linux: http://playground.arduino.cc/Learning/Linux

About Arduino Uno R3 board

What's on the board?

There are many varieties of Arduino boards that can be used for different purposes. Some boards look a bit different from the one below, but most Arduino have the majority of these components in common:



Power (USB / Barrel Jack)

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. In the picture above the USB connection is labeled **(1)** and the barrel jack is labeled **(2).** .

NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts. Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire. They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

- **GND (3):** Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- **5V (4) & 3.3V (5):** As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
- **Analog (6):** The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.

- **Digital (7)**: Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- **PWM (8)**: You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).
- **AREF (9)**: Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Reset Button

Just like the original Nintendo, the Arduino has a reset button **(10)**. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn't usually fix any problems.

Power LED Indicator

Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON' **(11)**. This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!

TX RX LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs **(12)**. These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

Main IC

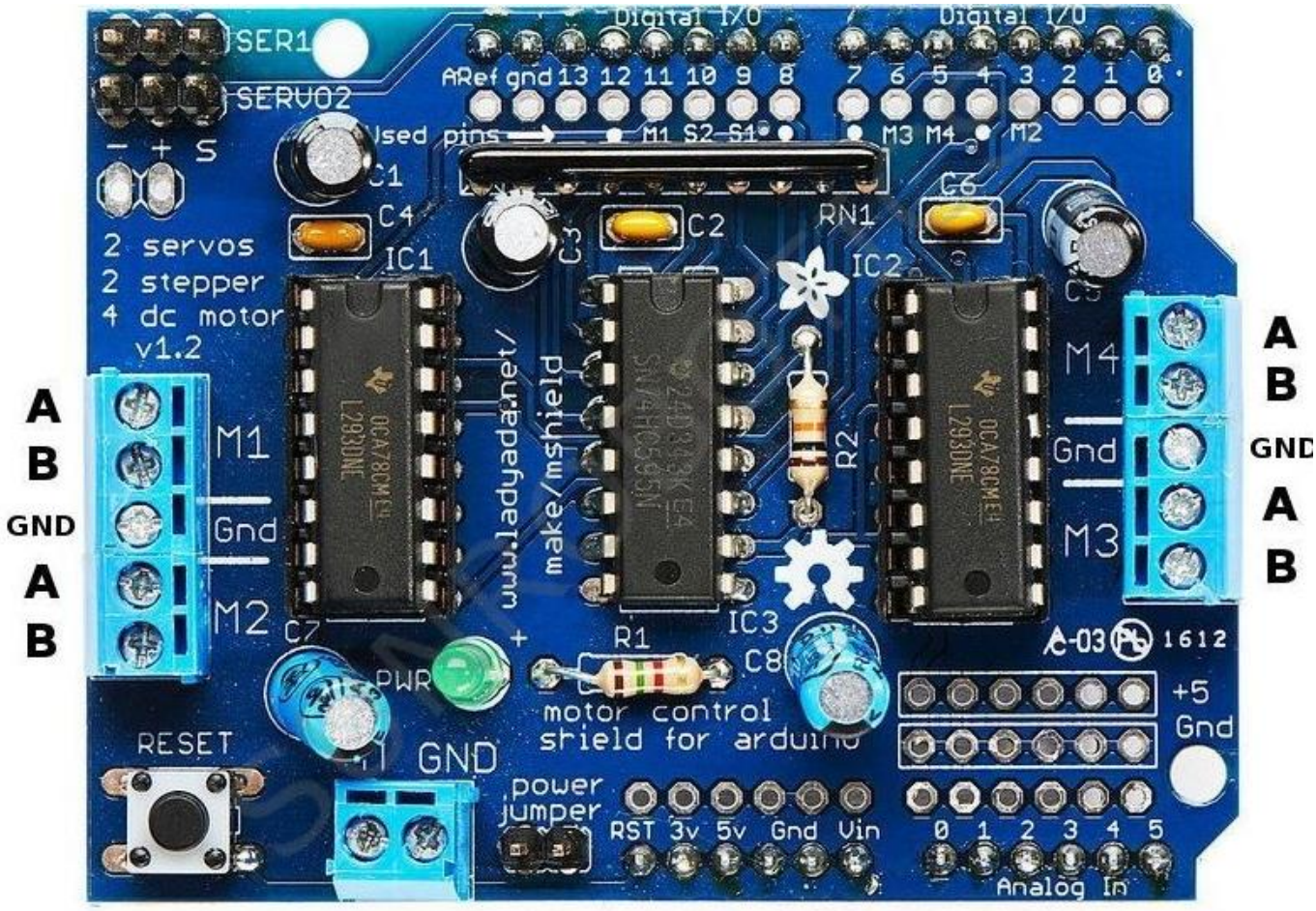
The black thing with all the metal legs is an IC, or Integrated Circuit **(13)**. Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the AT mega line of IC's from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

Voltage Regulator

The voltage regulator **(14)** is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might

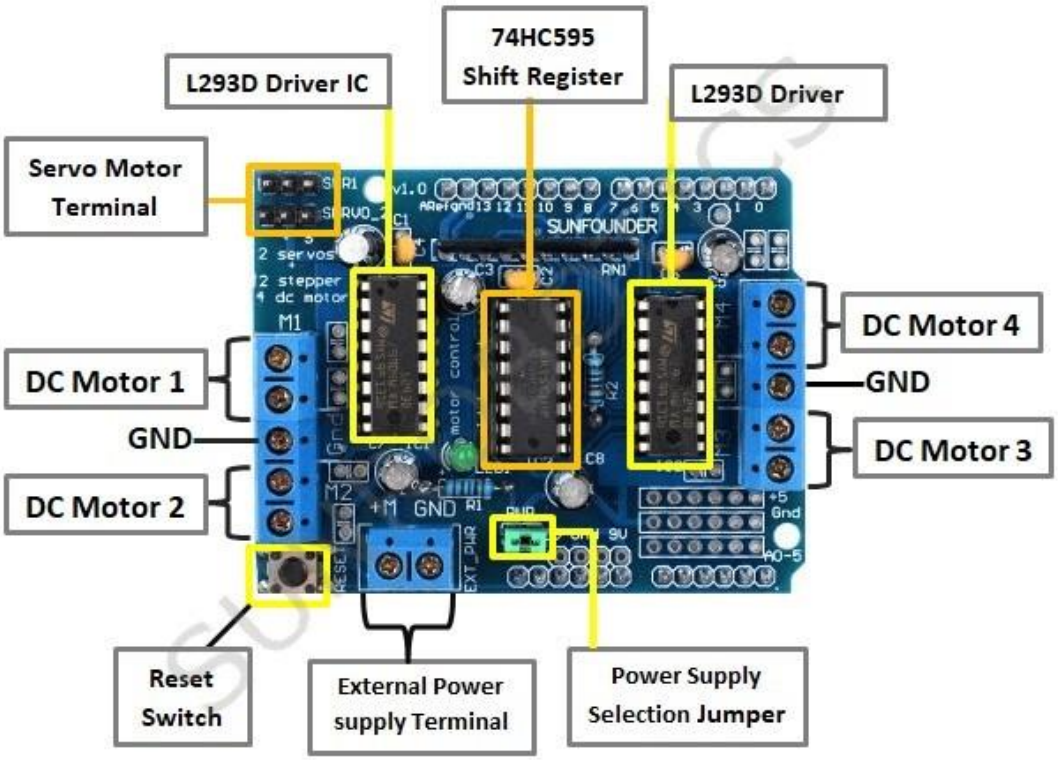
harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

About L293D motor Driver Shield Board



A motor driver is an integrated circuit chip which is usually used to control motors in autonomous robots. Motor driver act as an interface between Arduino and the motors .The most commonly used motor driver IC's are from the L293D. Here we used L293D Motor Driver Shield.It can drive 4 bi-directional DC motors with 8-bit speed selection(0-255),2 stepper motors (unipolar or bipolar) with single coil, double coil, interleaved or micro-stepping, 2 servo motors.

Schematic:



- **L293D Motor Driver & 74HC595 Shift Register**

The L293D is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors or single stepper motor. As the shield comes with two L293D motor driver chipsets that means it can individually drive up to four DC motors making it ideal for building four-wheel robot platforms.

The shield offers total 4 H-Bridges and each H-bridge can deliver up to 0.6A to the motor. The shield also comes with a 74HC595 shift register that extends 4 digital pins of the Arduino to the 8 direction control pins of two L293D chips.

- **Power Supply**

There exists three scenarios when it comes to supplying power for the motors through shield. Single DC power supply for both Arduino and motors: If you would like to have a single DC power supply for both Arduino and motors, simply plug it into the DC jack on the Arduino or the 2-pin EXT_PWR block on the shield. Place the power jumper on the motor shield.

You can employ this method only when motor supply voltage is less than 12V. (Recommended) Arduino powered through USB and motors through a DC power supply: If you would like to have the Arduino powered off of USB and the motors powered off of a DC power supply, plug in the USB cable. Then connect the motor supply to the EXT_PWR block on the shield. Do not place the jumper on the shield.

Two separate DC power supplies for the Arduino and motors: If you would like to have 2 separate DC power supplies for the Arduino and motors. Plug in the supply for the Arduino into the

DC jack, and connect the motor supply to the EXT_PWR block. Make sure the jumper is removed from the motor shield.

Warning:

DO NOT supply power to the EXT_PWR input when jumper is in place. It may damage the motor shield and also your Arduino!

1. As a bonus, the shield offers below features:
2. The shield comes with a pull-down resistor array to keep motors switched off during power-up.
3. The on-board LED indicates the motor power supply is Okay. If it is not lit, the motors will not run.
4. The RESET is nothing but Arduino's reset button. It just brought up top for convenience.

Output Terminals

The output channels of both the L293D chips are broken out to the edge of the shield with two 5-pin screw terminals viz. M1, M2, M3 & M4. You can connect four DC motors having voltages between 4.5 to 25V to these terminals.

Each channel on the module can deliver up to 600mA to the DC motor. However, the amount of current supplied to the motor depends on system's power supply.

You can also connect two stepper motors to output terminals. One stepper motor to motor port M1-M2 and other to M3-M4.

The GND terminal is also provided if you happen to have a unipolar stepper motor. You can connect the center taps of both stepper motors to this terminal.

The shield brings out the 16bit PWM output lines to two 3-pin headers to which you can connect two servo motors.

Lesson- 1 Two DC Motor Control Using L293D Motor driver shield with Arduino

Overview:

In this lesson, we will learn how to control DC Motors using L293 motor Driver shield device.

Components

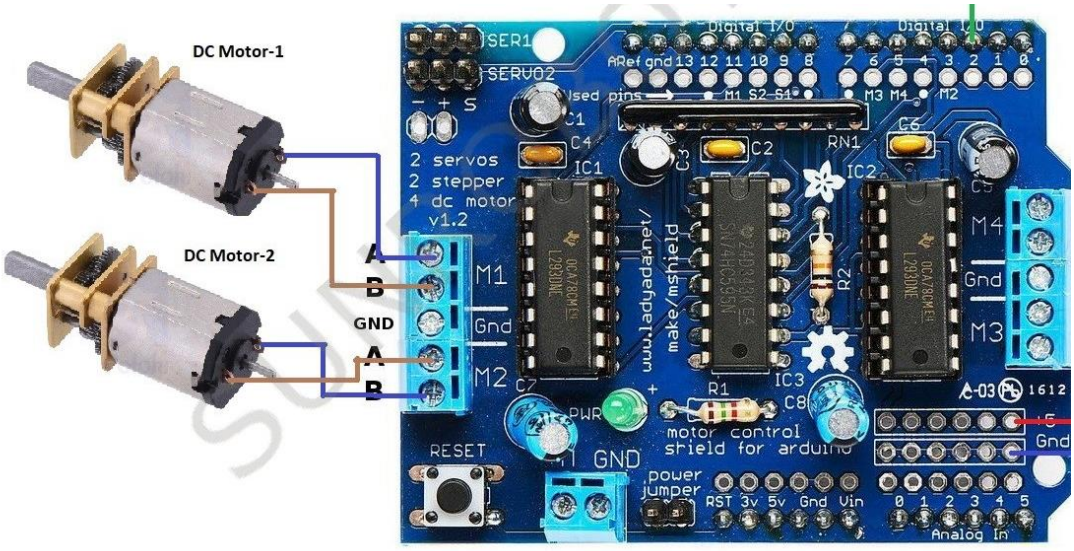
- 1 x Arduino UNO
- 1 x USB Cable
- 1 x L293D motor Driver Shield Module
- 2 x 5v DC Motor
- 2 x3.7v Battery Cell
- 1 x battery holder(two port)
- Jumper Wires

Principle

A motor driver is an integrated circuit chip which is usually used to control motors in autonomous robots. Motor driver act as an interface between Arduino and the motors .The most commonly used motor driver IC's are from the L293D. Here we used L293D Motor Driver Shield.It can drive 4 bi-directional DC motors with 8-bit speed selection(0-255),2 stepper motors (unipolar or bipolar) with single coil, double coil, interleaved or micro-stepping, 2 servo motors.

Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the "CODE" Folder.(for this code copy adafruit motor shield library in arduino library folder)

```
#include <AFMotor.h>
AF_DCMotor motor1(1);
AF_DCMotor motor2(2);

void setup() {
  motor1.setSpeed(250);
  motor2.setSpeed(250);
  motor1.run(RELEASE);
  motor2.run(RELEASE);
 }
void loop() {
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  delay(5000);
  delay(5000);

  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  delay(5000);
  delay(5000);

  motor1.run(FORWARD);
  motor2.run(BACKWARD);
  delay(5000);
  delay(5000);

  motor1.run(BACKWARD);
  motor2.run(FORWARD);
  delay(5000);
  delay(5000);
 }
```

Step 3: Compile the program and upload to Arduino UNO board

Lesson 2- Controlling a Servo Using Arduino

Overview:

In this lesson, we will introduce a new electronic device (Servo) to you, and tell you how to control it with the Arduino UNO using L293D Motor Driver Shield.

Components:

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x Servo
- L293D Motor Driver Shield
- Several jumper wires

Principle:

Driving the servos with L293D shield is as easy as pie.the motor shield actually breaks out Arduino's 16bit PWM output pins #9 & #10 to the edge of the shield with two 3-pin headers.
Power for the Servos comes from the Arduino's on-board 5V regulator, so you don't have to connect anything to the EXT_PWR terminal.

1. Servo motor

The servo motor has three wires: power, ground, and signal. The power wire is typically red, and should be connected to the 5V pin on the Arduino board. The ground wire is typically black or brown and should be connected to a ground pin on the Arduino board. Usually the signal pin is yellow, orange or white, and should be connected to a digital pin on the Arduino board. Note that the servo motor draws a considerable amount of power, if you need to drive more than one or two servos, you'll probably need to power them with an extra supply (i.e. not the +5V pin on your Arduino). Be sure to connect the grounds of the Arduino and external power supply together.

2. Servo library

This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

3. Key functions:

attach()

Attach the Servo variable to a pin. Note that in Arduino 0016 and earlier, the Servo library supports only servos on only two pins: 9 and 10.
Syntax:
servo.attach(pin)
servo.attach(pin, min, max)
Parameters:
servo: a variable of type Servo
pin: the number of the pin that the servo is attached to
min (optional): the pulse width, in microseconds, corresponding to the minimum (0-degree) angle on the servo (defaults to 544)

max (optional): the pulse width, in microseconds, corresponding to the maximum (180-degree) angle on the servo (defaults to 2400)

Procedure:

Step 1: Build the circuit.

Step 2: Program:

> **NOTE:**
> Before you upload the program, you need to install the Servo.zip library. Otherwise the program will throw an error message.

Step 3: Compile the program and upload to Arduino UNO board

```
#include <Servo.h>
Servo myservo;  // create servo object to control a servo
int pos = 0;  // variable to store the servo position

void setup()
{
  // attaches the servo on pin 10 to the servo object
  myservo.attach(10);
}
void loop()
{
  myservo.write(15);//goes to 15 degrees
  delay(1000);//wait for a second

  myservo.write(30);//goes to 30 degrees
  delay(1000);//wait for a second.33
```

```
  myservo.write(45);//goes to 45 degrees
  delay(1000);//wait for a second.33

  myservo.write(60);//goes to 60 degrees
  delay(1000);//wait for a second.33

  myservo.write(75);//goes to 75 degrees
  delay(1000);//wait for a second.33

  myservo.write(90);//goes to 90 degrees
  delay(1000);//wait for a second

  myservo.write(75);//back to 75 degrees
  delay(1000);//wait for a second.33

  myservo.write(60);//back to 60 degrees
  delay(1000);//wait for a second.33

  myservo.write(45);//back to 45 degrees
  delay(1000);//wait for a second.33

  myservo.write(30);//back to 30 degrees
  delay(1000);//wait for a second.33

  myservo.write(15);//back to 15 degrees
  delay(1000);//wait for a second

  myservo.write(0);//back to 0 degrees
  delay(1000);//wait for a second

  for(int num=0;num<=180;num++)
  {
    myservo.write(num);//back to 'num' degrees(0 to 180)
    delay(10);//control servo speed
  }
  for(int num=180;num>=0;num--)
  {
    myservo.write(num);//back to 'num' degrees(180 to 0)
    delay(10);//control servo speed
  }
}
```

Now, you should see the servo rotate from 0 to 180 degrees, and then do it in the opposite direction.

Lesson 3 – HC05 UART Communication with arduino

Overview:
In this tutorial we will study about the arduino using Bluetooth module HC-05.

Components:

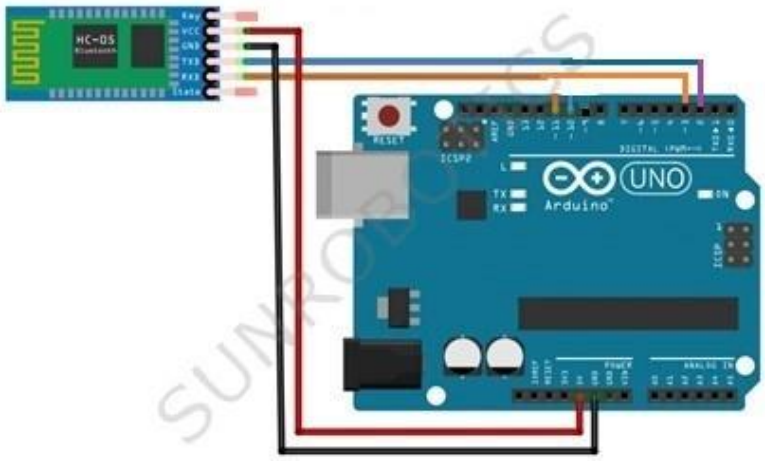
- 1 x Arduino Uno
- 1 x USB Cable
- 1 x Bluetooth Module HC-05
- Jumper wires

Principle:

Arduino can communicate with other devices via Bluetooth using the module HC-05 (master/slave). It enables the Arduino to connect and exchange data with other devices such as Smartphone, computer or other microcontrollers. Bluetooth communication can be used to control a robot remotely, Display and store data on your computer or on your smartphone, for instance.

Schematic:
Vcc – Arduino 5v
GND – Arduino GND
TXD – Arduino Pin 2
RXD – Arduino Pin 3

Procedure:
Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*
  HC05 - Bluetooth AT-Command mode
*/
#include "SoftwareSerial.h"
SoftwareSerial MyBlue(2, 3); // RX | TX
char c = ' ';
void setup()
```

```
{
  Serial.begin(9600);
  MyBlue.begin(9600);  //Baud Rate for AT-command Mode.
  Serial.println("***AT commands mode***");
}
void loop()
{
  //from bluetooth to Terminal.
  if (MyBlue.available())
  {
    c = MyBlue.read();
    Serial.write(c);
  }
  //from termial to bluetooth
  if (Serial.available())
  {
    c = Serial.read();
    MyBlue.write(c);
  }
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Step 4: “Bluetooth Terminal HC-05” free App Download In your phone from Play Store. Btn command set anything “B” ,“A” etc. then when app button press command send to arduino terminal via Bluetooth.

i. Open application

s

ii. Scan devices from this pair and connect

iii. Set btn Command and then press button

iv. this Command output also see in arduino serial monitor

AT Commands

In general, typing the command **AT+<command>?** will prompt the saved parameter (ex: AT+PSWD? will display the module PIN code). If you enter **AT+<command>=<Param>**, you can set the parameter value(ex: AT+PWSD=0000 to modify the PIN code to 0000).
Here are some of the AT commands:
To test communication,enter **AT** in the serial monitor of the Arduino IDE. If everything is setup correctly it should display OK.
To modify the module name, enter **AT+NAME=<Param>**. The module should answer OK (Defaut HC-05, Ex: To change the name to BTM1 enter AT+NAME=BTM1).
To modify the PIN code, enter **AT+PSWD=<Param>** . The module should answer OK(Default 1234 Ex: To change the PIN to 0000 enter AT+PSWD=0000).
AT+ROLE=<Param> to midy the role of the module as slave or master (Default 0, Ex: to change the role as master enter AT+ROLE=1, as slave enter AT+ROLE=0).
To modify the baudrate, enter **AT+UART=<Param1>,<Param2>,<Param3>** with Param1, 2 and 3 serial communication parameters: baudrate, stop bit and parity bit respectively (By default,set to 9600,0,0. Ex: to modify the baudrate to 115200 enter AT+UART=115200,0,0).
Other AT commands exist for the Bluetooth module HC-05 that you can find following the link.

Slave Configuration
To set the module as a slave, you can change the name as AT+NAME=HC05-Slave and choose the communication parameters and the PIN code that you want. You'll need to make sure that master and slave as the same communication parameters.
AT returns OK
AT+NAME=HC05-Slave
AT+UART=9600,0,0
AT+ROLE=0
Enter AT+ADDR to obtain the module address (ex: +ADDR:98d3:32:21450e)

Master Configuration
To set the module as master, you need to change the role and set the same communication parameter as the slave module.
AT returns OK
AT+NAME=HC05-Master
AT+UART=9600,0,0
AT+ROLE=1
The slave module address must be enter in the master module to allows it to appair: AT+BIND=98d3,32,21450e (replace dots ":" by coma ",")

Lesson- 4 Ultrasonic Distance sensor with Arduino

Overview:

In this tutorial we will study about the arduino and Ultrasonic Distance sensor based obstacle distance measurement.

Components:

- 1 x Arduino Uno
- 1 x USB Cable
- 1 x Ultrasonic Module(HC-SR04)
- Jumper wires

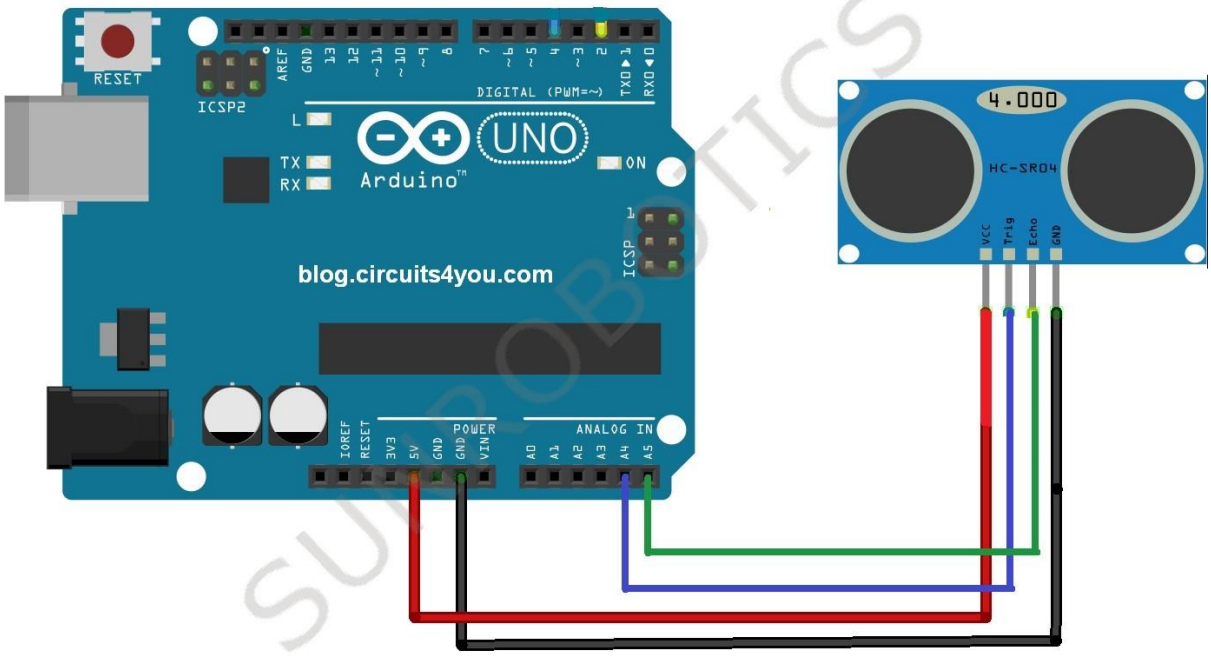
Principle:

Ultrasonic sensors are useful for measuring distances. Ultrasonic waves are transmitted and whenever these strike an obstacle and return back in the form of an echo. Difference of outgoing sound and returning echo gives us the distance.

Schematic:

Ultrasonic sensor HC-SR04 Consists of 4PINS:
VCC → Connect with 5V
Triggering (Signal IP/OP) → Connect with A4
ECHO Signal → Connect with A5
Ground → Connect with Ground

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
#define trigger A4 //18
#define echo A5 //M19
float time = 0, distance = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
}

void loop()
{
  long dur;
  long tocm;
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  dur = pulseIn(echo, HIGH);
  tocm = microsecondsToCentimeters(dur);
  Serial.print("Distance: ");
  Serial.print(tocm);
  Serial.println("cm");
  delay(100);
}
long microsecondsToCentimeters(long microseconds)
{
  return microseconds / 29 / 2;
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Step 4: Output will show on Terminal/ arduino serial monitor

Lesson 5 – POLY Robot control Using android app

Overview:

In this tutorial we will study about the arduino and HC-05 based Poly Robot control using Android app

Components:

- 1 x Arduino Uno
- 1 x USB Cable
- 1 x Bluetooth Module HC-05
- 1 x Poly Robo Chasis
- 2 x 3.7 v Battery cell
- 1 x battery Holder
- 1 x L293D Motor driver Shield Module
- 2 x 5v DC Motor
- Jumper wires

Principle:

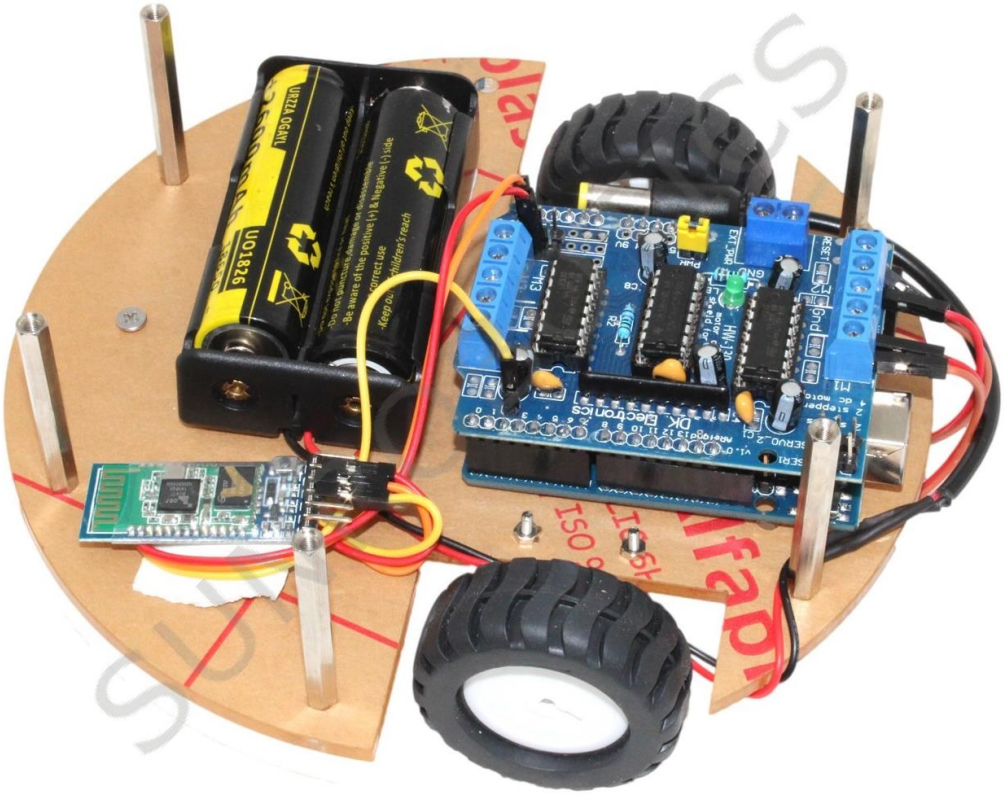
For the android communication with our Bluetooth module I've used the "Arduino Bluetooth Robot Car "android app, It's completely free, so you just need to go to "Play store" and download it. Then you just need to connect your smartphone with the Bluetooth module. We can see in application ↑ ↓ ← → arrow and Lock menu.

Working:-

- when Lock on –system work in continuity mode. i.e. when up arrow press one time system will lock this arrow position and robot car continue move in forward direction.
- when Lock off- system work when key continues press. When release robot car will be stop.
- In lock condition app send command to arduino using hc-05 (when ↑ press send "F", when ↓ press send "B", when ← press send "L", when → press send "R")
- In Un-lock condition app send command to arduino using hc-05 (when ↑ press send "F" after release send "S", when ↓ press send "B" after release send "S", when ← press send "L" after release send "S", when → press send "R" after release send "S")

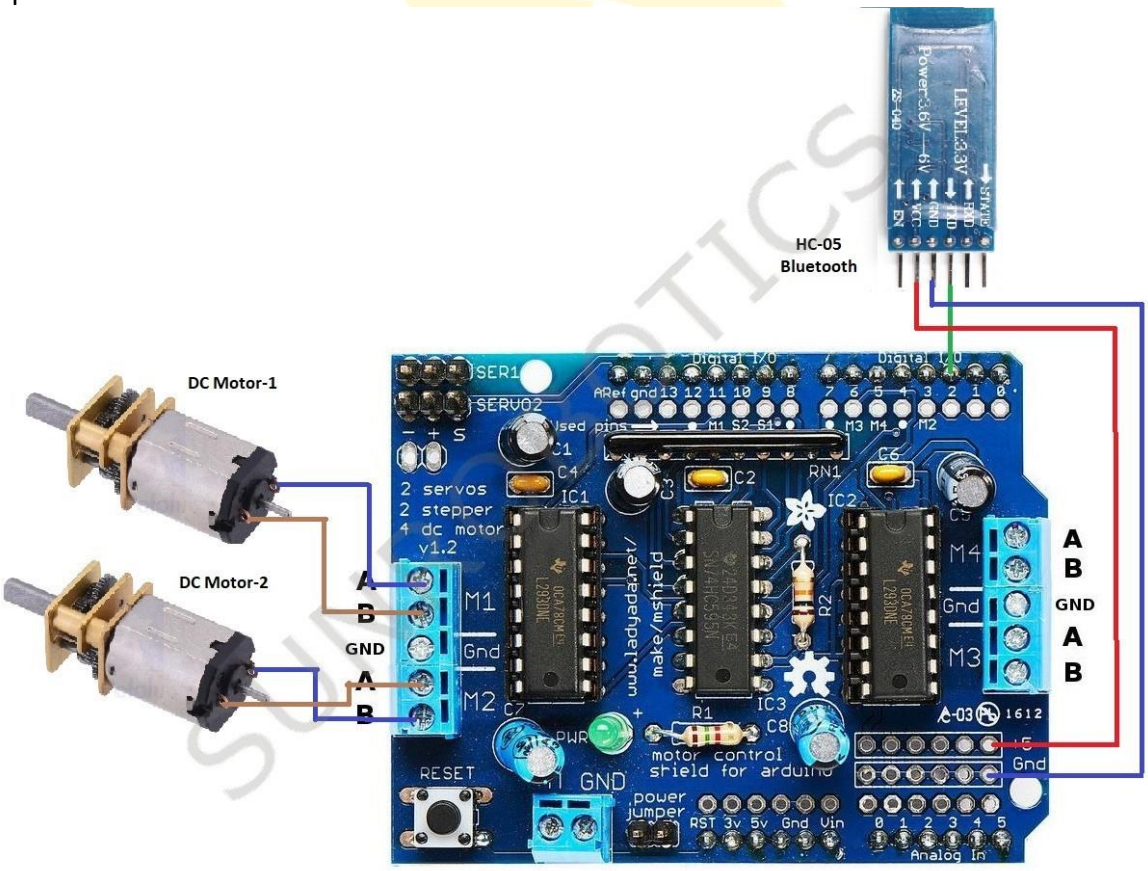
Schematic:

Vcc – Arduino 5v
GND – Arduino GND
TXD – Arduino Pin 2
L293D Motor Driver Shield place on Arduino Board
DC motor connect with L293D Motor Driver Module



Procedure:

Step 1: Build the circuit



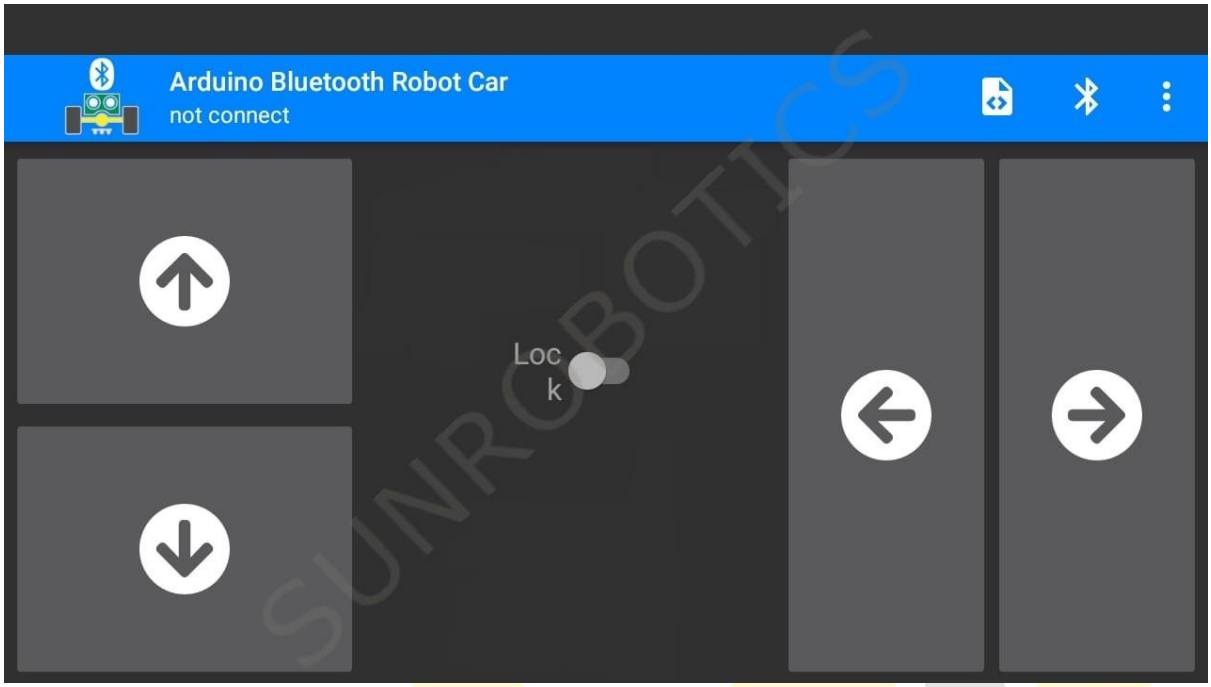
Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*
  HC05 - Bluetooth AT-Command mode
  Download "Android Control robot car" applicationin your Mobile Phone from PlayStore
*/
#include "SoftwareSerial.h"
#include <AFMotor.h>

AF_DCMotor motor1(1);
AF_DCMotor motor2(2);
SoftwareSerial MyBlue(2, 3); // RX | TX

int state;

void setup()
{
  Serial.begin(9600);
  MyBlue.begin(9600);  //Baud Rate for AT-command Mode.
  motor1.setSpeed(255);
  motor2.setSpeed(255);
  motor1.run(RELEASE);
  motor2.run(RELEASE);
}
void loop()
{
  if (MyBlue.available() > 0)
  {
    state = MyBlue.read();
  }
  if (state == 'S')
  {
    motor1.run(RELEASE);
    motor2.run(RELEASE);
  }
  if (state == 'F')
  {
    motor1.run(FORWARD);
    motor2.run(FORWARD);
  }
  if (state == 'B')
  {
    motor1.run(BACKWARD);
    motor2.run(BACKWARD);
  }
  if (state == 'L')
  {
    motor1.run(BACKWARD);
    motor2.run(FORWARD);
  }
  if (state == 'R')
```

```
  {
    motor1.run(FORWARD);
    motor2.run(BACKWARD);
  }
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Step 4: application Download in your phone. (Arduino Bluetooth Robot car)



Lesson- 6 Ultrasonic Servo based POLY Robot control Using android App(Arduino Control Robot car)

Overview:

In this tutorial we will study about the arduino, servo, ultrasonic distance sensor and HC-05 based Poly Robot control using Android app

Components:

- 1 x Arduino Uno
- 1 x USB Cable
- 1 x Bluetooth Module HC-05
- 1 x Poly Robo Chassis
- 2 x 3.7 v Battery cell
- 1 x battery Holder
- 1 x L293D Motor driver Shield Module
- 2 x 5v DC Motor
- 1 x Ultrasonic Module(HC-SR04)
- 1 x Servo Motor
- Jumper wires

Principle:

For the android communication with our Bluetooth module we've used the "Arduino Bluetooth Robot Car "android app, It's completely free, so you just need to go to "Play store" and download it. Then you just need to connect your smartphone with the Bluetooth module (HC-05). We can see in application ↑ ↓ ← → arrow and Lock menu. Ultrasonic distance through measure object to poly robo distance and scan in 180° degree by servo motor.

Working:-

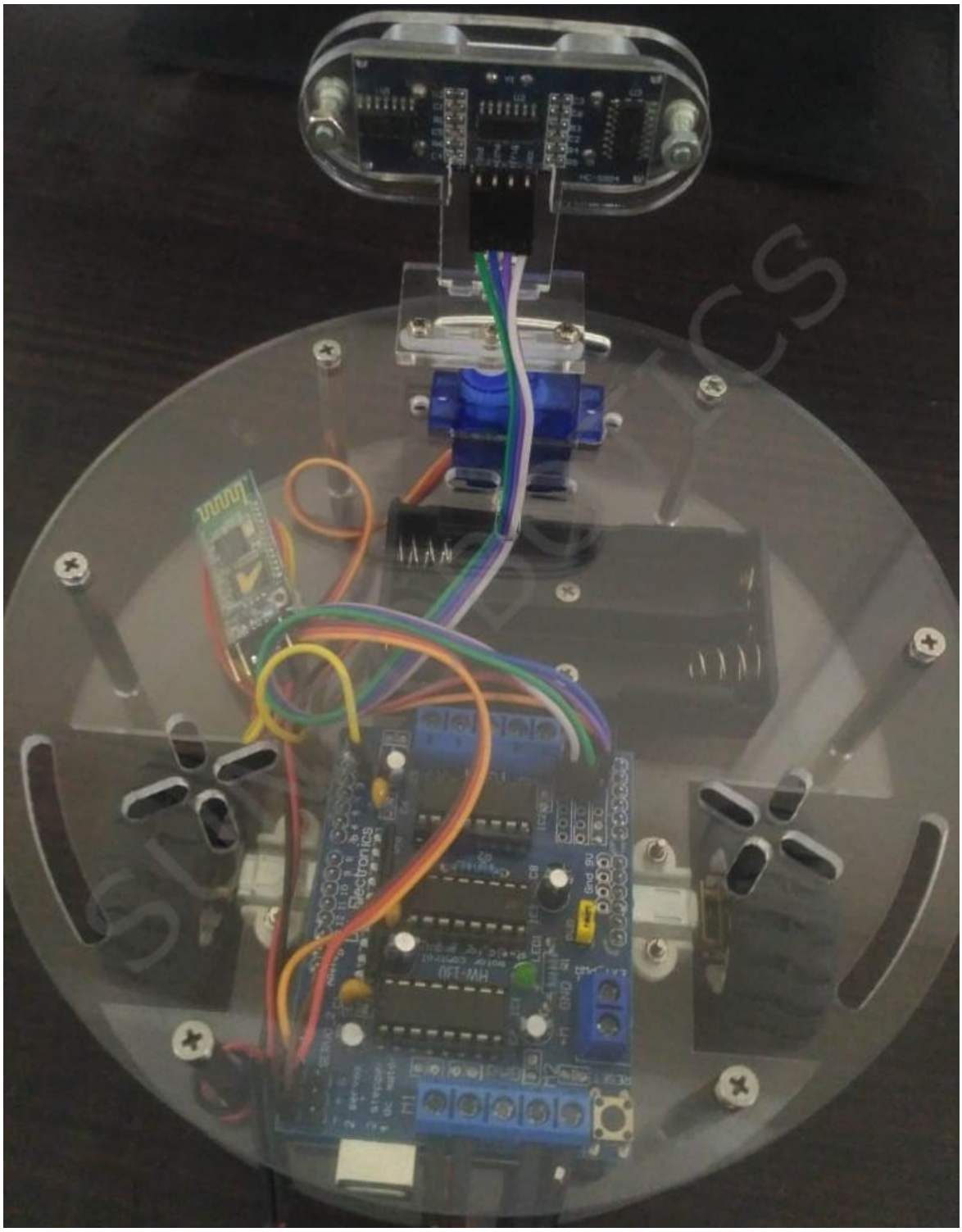
- When lock on –system work in continuity mode. I.e. when up arrow press one time system will lock this arrow position and robot car continue move in forward direction.
- When lock off- system work when key continues press. When release robot car will be stop.
- In lock condition app send command to arduino using hc-05 (when ↑ press send "F", when ↓ press send "B", when ← press send "L", when → press send "R")
- In Un-lock condition app send command to arduino using hc-05 (when ↑ press send "F" after release send "S", when ↓ press send "B" after release send "S", when ← press send "L" after release send "S", when → press send "R" after release send "S")
- In our case robo move In forward direction and if object detect in distance of less than 18cm (measure by ultrasonic distance sensor) robo will stop and after 1 second it move in backward direction for 3 seconds and then scanning will start(using servo motor)

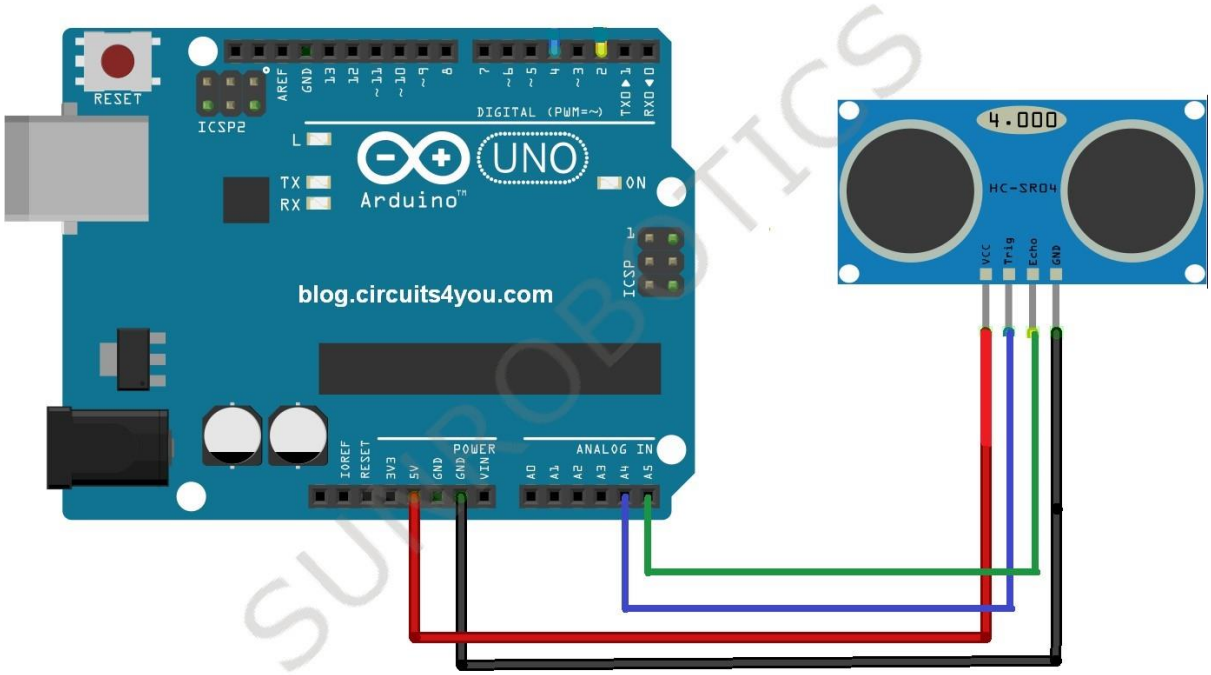
Schematic:

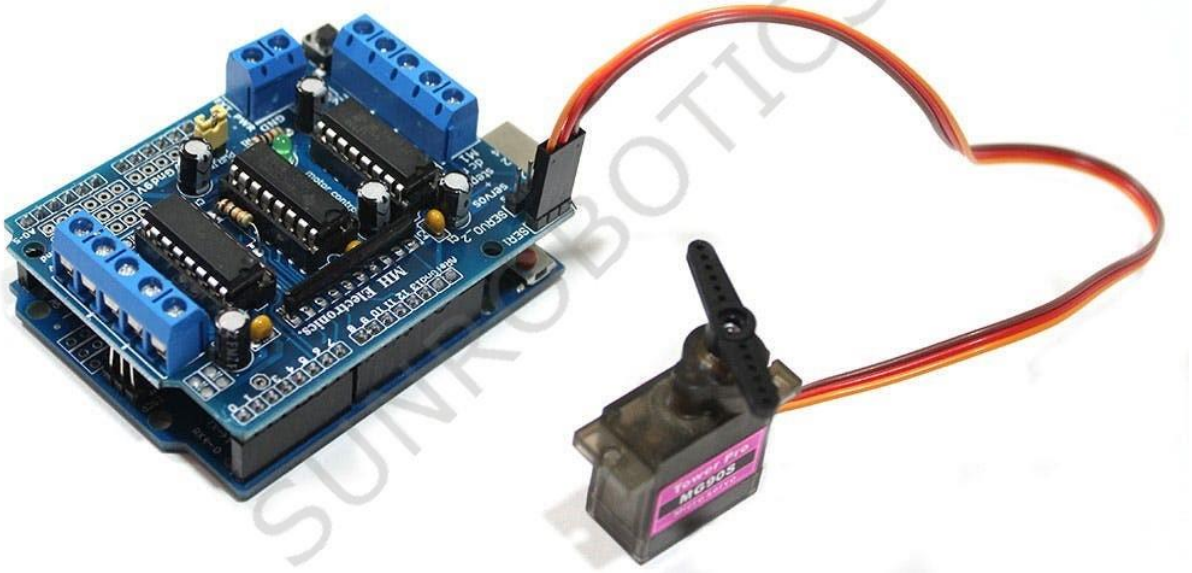
1. Ultrasonic sensor HC-SR04 Consists of 4PINS:
   VCC → Connect with 5V
   Triggering (Signal IP/OP) → Connect with A4
   ECHO Signal → Connect with A5
   Ground → Connect with Ground
2. Bluetooth Connecion:
   Vcc – Arduino 5v
   GND – Arduino GND

TXD – Arduino Pin 2
RXD – Arduino Pin 3

3. Servo Motor Connection:
   L293D module “SER1”(follow below image)
4. DC Motors Connection:
   L293D module “M1” and “M2”(follow below image)







Step 2: Program: Open /Copy the code from the “CODE” Folder

```
#include "SoftwareSerial.h"
#include <AFMotor.h>
#include <Servo.h>

#define trigger A4 //18
#define echo A5 //M19
```

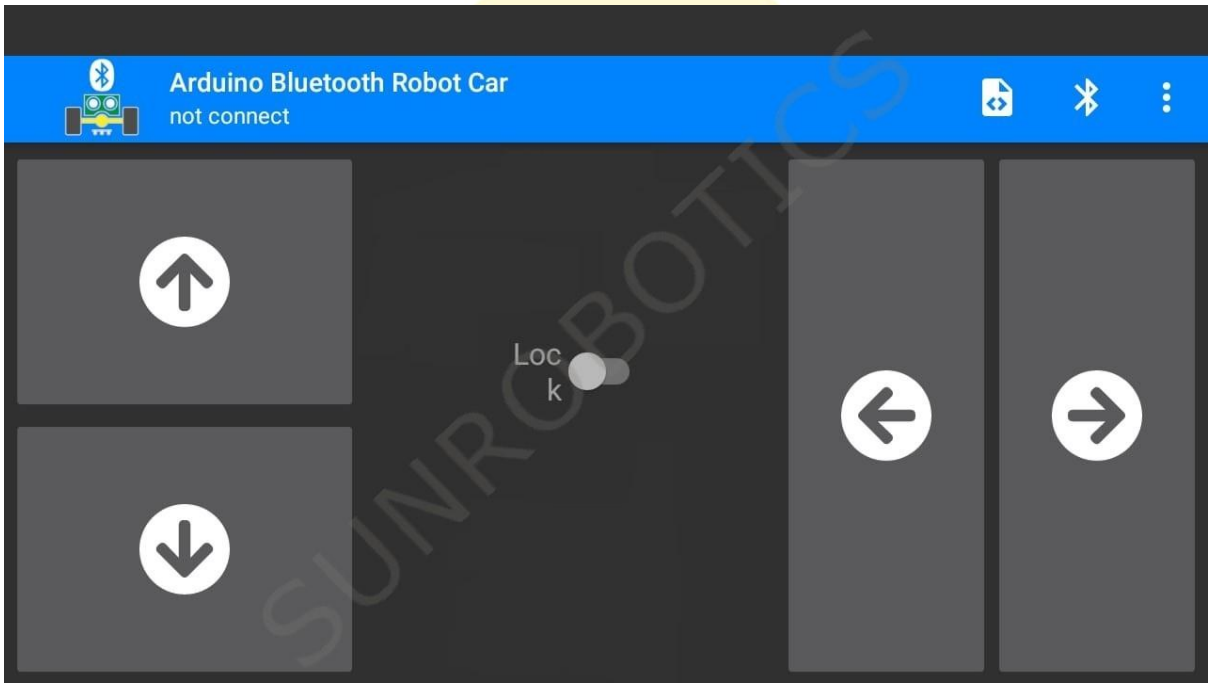
```
float time = 0, distance = 0;
int state;
int num = 0;  // variable to store the servo position
long dur;
long tocm;
bool UDSscanFlag = false;
void UDS_read();
void BLE_cmd_read();

AF_DCMotor motor1(1);
AF_DCMotor motor2(2);
Servo myservo;  // create servo object to control a servo
SoftwareSerial MyBlue(2, 3); // RX | TX

void setup()
{
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
  myservo.attach(10);
  Serial.begin(9600);
  MyBlue.begin(9600);  //Baud Rate for AT-command Mode.
  motor1.setSpeed(255);
  motor2.setSpeed(255);
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  myservo.write(90);
}

void loop()
{
  UDS_read();
  BLE_cmd_read();
}
void UDS_read()
{
  digitalWrite(trigger, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger, LOW);
  //  delayMicroseconds(2);
  dur = pulseIn(echo, HIGH);
  tocm = dur * 0.0340 / 2;
  Serial.print("distance : ");
  Serial.println(tocm);
  delay(100);
  if (UDSscanFlag == true)
  {
    if ((tocm > 9) && (tocm < 16)) {
      motor1.run(RELEASE);
```

```
    motor2.run(RELEASE);
    delay(1000);
    motor1.run(BACKWARD);
    motor2.run(BACKWARD);
    delay(3000);
    motor1.run(RELEASE);
    motor2.run(RELEASE);
    for (int num = 90; num >= 0; num--)
    {
     myservo.write(num);//back to 'num' degrees(180 to 0)
     delay(50);//control servo speed
    }
    for (num = 90; num <= 180; num++)
    {
     myservo.write(num);//back to 'num' degrees(0 to 180)
     delay(50);//control servo speed
    }
    myservo.write(90);
   }
  }
}

void BLE_cmd_read()
{
  if (MyBlue.available() > 0)
  {
   state = MyBlue.read();
   Serial.println(state);
   if (state == 'S')
   {
    UDSscanFlag = false;
    motor1.run(RELEASE);
    motor2.run(RELEASE);
   }
   if (state == 'F')
   {
    UDSscanFlag = true;
    motor1.run(FORWARD);
    motor2.run(FORWARD);
    delay(500);
   }
   if (state == 'B')
   {
    UDSscanFlag = true;
    motor1.run(BACKWARD);
    motor2.run(BACKWARD);
   }
   if (state == 'L')
   {
    UDSscanFlag = true;
    motor1.run(BACKWARD);
```

```
    motor2.run(FORWARD);
   }
   if (state == 'R')
   {
    UDSscanFlag = true;
    motor1.run(FORWARD);
    motor2.run(BACKWARD);
   }
  }
}
```

Step 3: Compile the program and upload to Arduino UNO board.

Step 4: application Download in your phone. (Arduino Bluetooth Robot car)




SUNROBOTICS
Thank
You!
www.sunrobotics.in