Android Controlled Robot for Beginners (A to Z)

by **taifur** on April 24, 2016

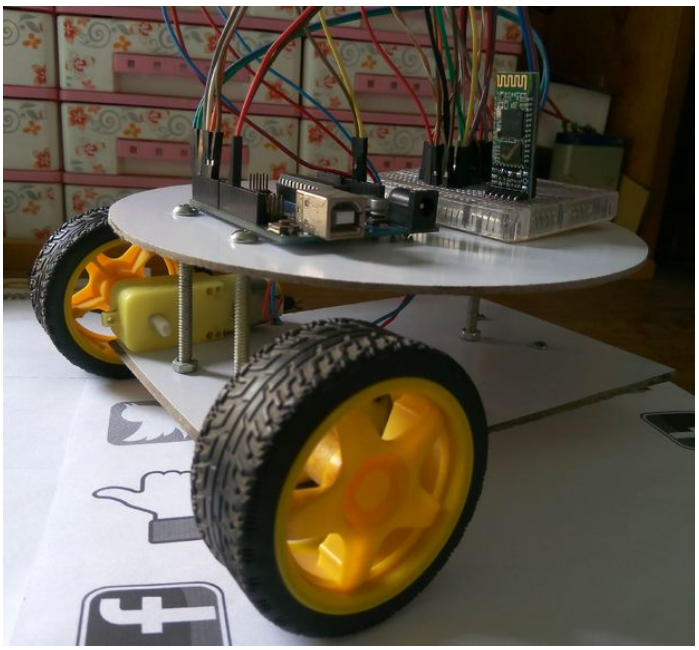
Table of Contents

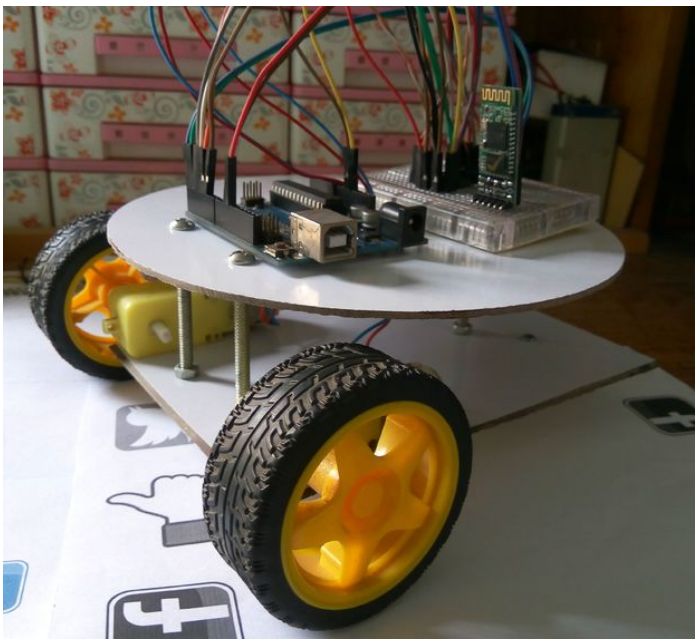
Author:taifur
I like to learn, like to make, like to share.

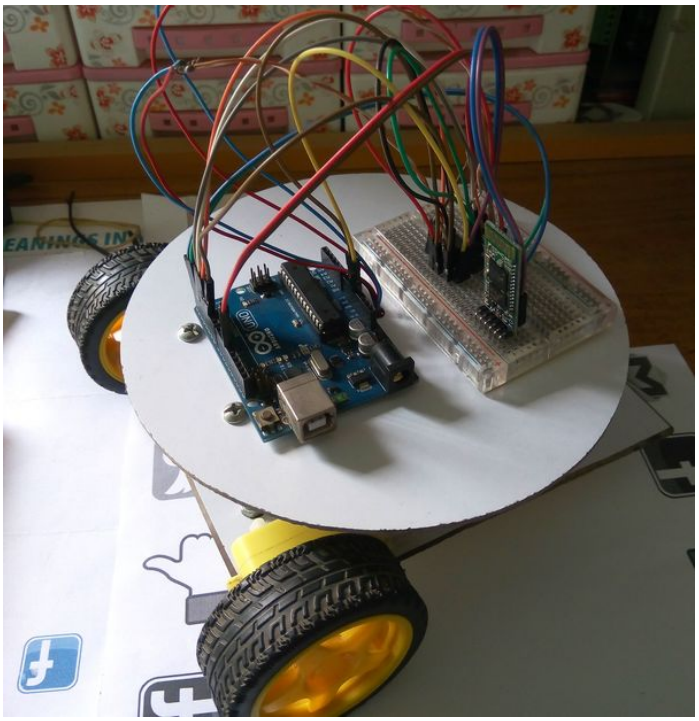
Intro: Android Controlled Robot for Beginners (A to Z)

Do you like Robots? Are you new in Robotics? Do you thinking to build a Robot from scratch? If yes, this instructables is for you. In this instructables I will show you how you can build a robot from scratch, I also show you how you can control you robot using your Android phone. I will guide you to build an android application to control your robot via bluetooth. Don't worried, you don't need any programming experience to build android app because I will use MIT App Inventor to build an android app.

I am sure If you follow this, you will able to build Line Following Robot, Obstacle Avoiding Robot and so on. So, let's start working.







Step 1: Stuff required

Hardware :

Arduino / Arduino Clone or make your own custom arduino board with this tutorial.

Two DC gear motors like this (Robotshop).

A ball caster (Sparkfun)

Two robot wheels (Sparkfun)

Chassis , usually a small acrylic board will do.

HC-05 Bluetooth Module (Amazon)

Four AA batteries and battery holder.

L293D Motor Driver IC

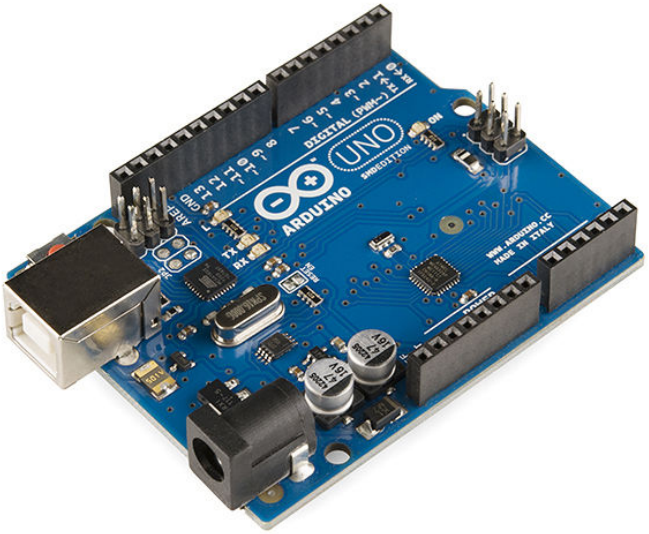
Small breadboard

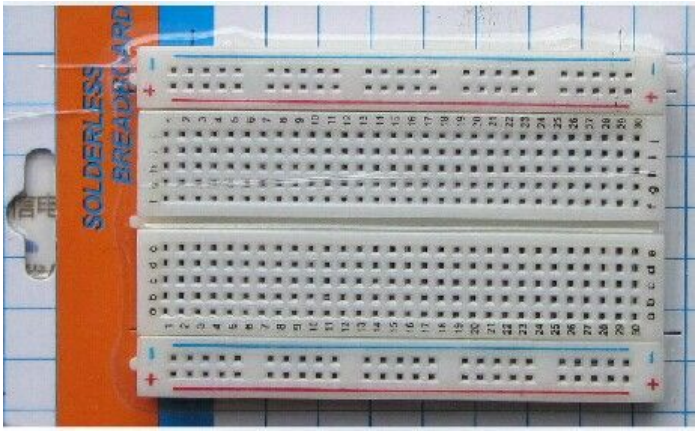
Some nuts and bolts

Software :

Arduino IDE : Arduino

MIT App Inventor 2




SOLDERLESS
BREADBOARD

Step 2: Make Chassis (part 1)

Readymade chassis that are available in the market are really good, cheap and for all you know they might have been subjected to a R&D before fabrication. In spite of all these advantages I still see the following issues with readymade chassis.

1. They are not available the shape and size that I feel fit for my robot. Most of the time I am forced to restrict the robot size to the size of the chassis that is available.
2. They do not give the flexibility that a custom design can have.
3. You cannot cast your artistic touch (if you have one that is) over the design.
4. Above all, they are not unique. If you go for an event with high hopes and find that there are a dozen robots that looks exactly like yours, then you will understand the importance of being unique.

So the only other alternative is to build your own chassis. There are wide varieties of materials to choose from when it comes to building your own chassis. I have listed out the commonly available materials over which robots are built.

Wood: Wood is the easiest and cheapest material that is also strong enough to support the weight of a large robot. Most of the time robots are built on wood. Wood is preferred because it is easily available. You can walk into a building under construction and you will have enough and more wood to for all the robot's you can possibly build in a month's time. The reason they are chosen is that they are really easy to work with.

Plastics: Most people like to use acrylic Plexiglas sheets in place of glass for transparent applications and small robot bases because it is easy to drill and tap and can be cut with a jigsaw. PVC (pipe) can be useful for projects where lightweight strength is needed. Plexiglas, PVC, and most other plastic can be formed or shaped using heat gun.

Metal: It is hard to beat metal for building a robot frame. It is extremely strong, durable, and can be joined by either welding or using bolts/nuts. Cutting is a bit more difficult, requiring either a hacksaw (and some elbow grease) or a reciprocating saw with a fine-tooth metal blade. Once built, a metal frame will last for years and will not warp or change shape.

Fiberglass: Fiberglass is an outstanding material for creating specific shapes that would be nearly impossible to make from metal or wood. It is also extremely strong and rigid once set, as well as waterproof. The process involves laying a fiberglass cloth and then applying a two-part resin on top of the cloth. It only takes about 1 hour to harden, but it does make some strong fumes.

Details: http://embedjournal.com/

Though wood has more advantages this post will cover the use of acrylic as the material for building chassis. Acrylic is available in all shapes and sizes with a wide range of thickness to choose from. One other compelling factor is that it available in vibrant colors which gives a new visual appeal to you robot.

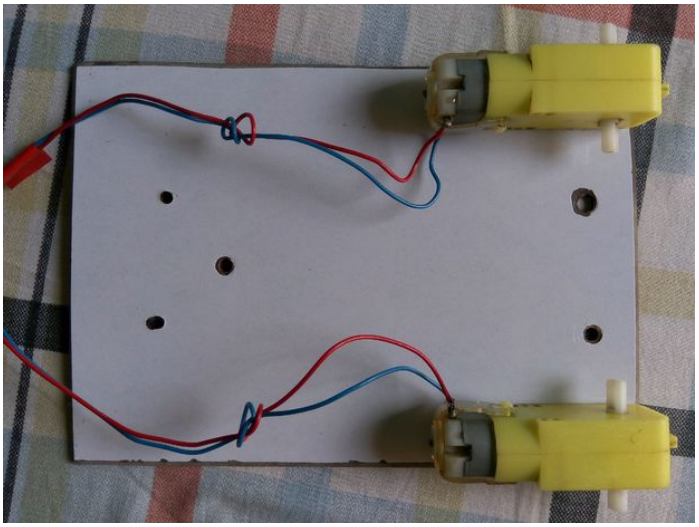
Take two pieces of acrylic sheet, one is round shape and another is rectangular shape as shown in above image. Make drill and connect together using nuts and bolts. Use glue to connect motor to the rectangular box.

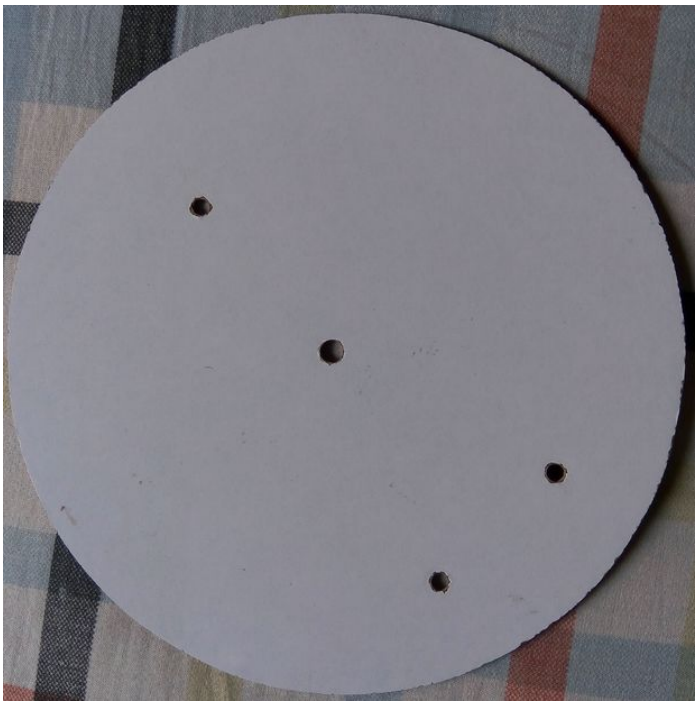
Brushed DC motors are ideally suited to robotics projects because they are small, powerful and easy to control at low voltages commonly provided by battery packs while still being highly efficient and reliable.

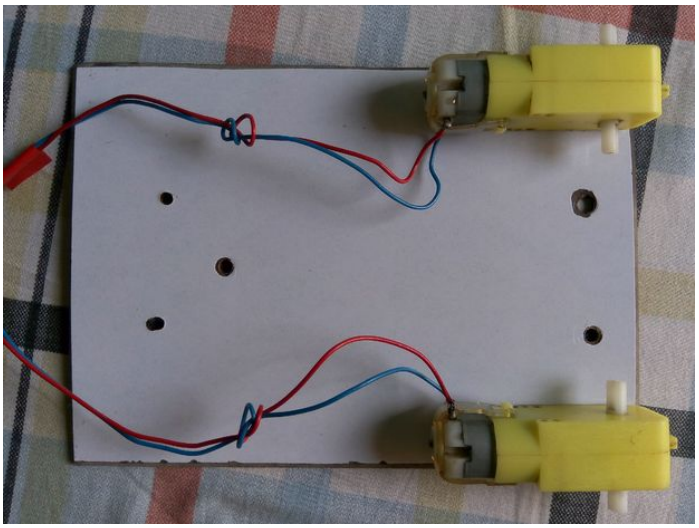
A gearmotor is just a electric motor with a gearbox attached. The torque and speed specifications quoted are for the output shaft. Often the same DC motor is used with different gearboxes to provide a range of available characteristics. Our motors work at voltages suitable for powering from battery packs and low voltage AC-DC switch mode power supplies.

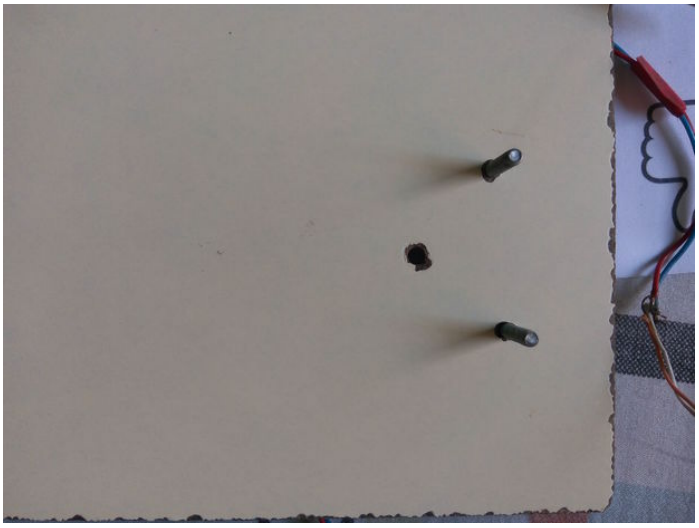
L293 Motor Driver and H-Bridges

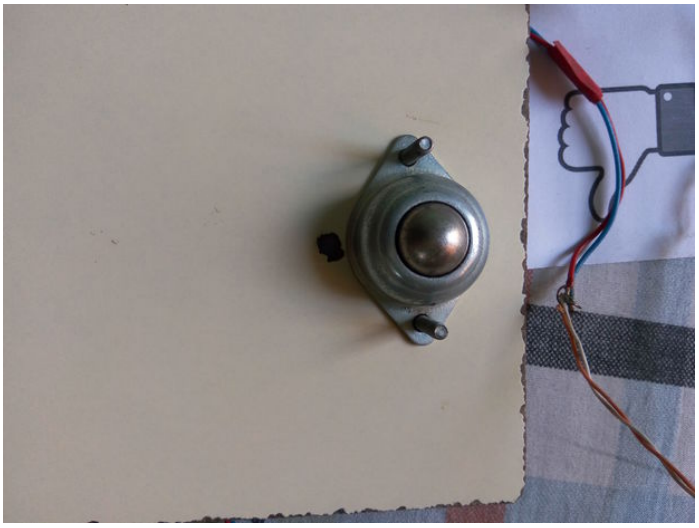
The most common method to drive DC motors in two directions under control of a computer is with an H-bridge motor driver. H-bridges can be built from scratch with bi-polar junction transistors (BJT) or with field effect transistors (FET), or can be purchased as an integrated unit in a single integrated circuit package such as the L293. The L293 is simplest and inexpensive for low current motors, For high current motors, it is less expensive to build your own H-bridge from scratch.
ITP Physical Computing has a terrific tutorial on using an Arduino and an L293 to control a bi-directional motor. The Twin Cities Robotics Club has an *excellent* tutorial on H-bridges, and complete detail on how to build your own $5.00 H-bridge good for several amps. From the same source is a detailed tech note on PWM speed control of a motor using an H-bridge and a PIC microcontroller The L293 is an integrated circuit motor driver that can be used for simultaneous, bi-directional control of two small motors. Small means small. The L293 is limited to 600 mA, but in reality can only handle much small currents unless you have done some serious heat sinking to keep the case temperature down. Unsure about whether the L293 will work with your motor? Hook up the circuit and run your motor while keeping your finger on the chip. If it gets too hot to touch, you can't use it with your motor. (Note to ME2011 students: The L293 should be OK for your small motor but is not OK for your gear motor.) The L293 comes in a standard 16-pin, dual-in line integrated circuit package. There is an L293 and an L293D part number. Pick the "D" version because it has built in flyback diodes to minimize inductive voltage spikes. The L293D can be purchased for somewhere between $2 and $3 (quantity one) fromwww.mouser.com (PN 511-L293D) or www.digikey.com (PN 296-9518-5-ND). For complete information, consult the Unitrode L293 data sheet (PDF file, 626Kb).

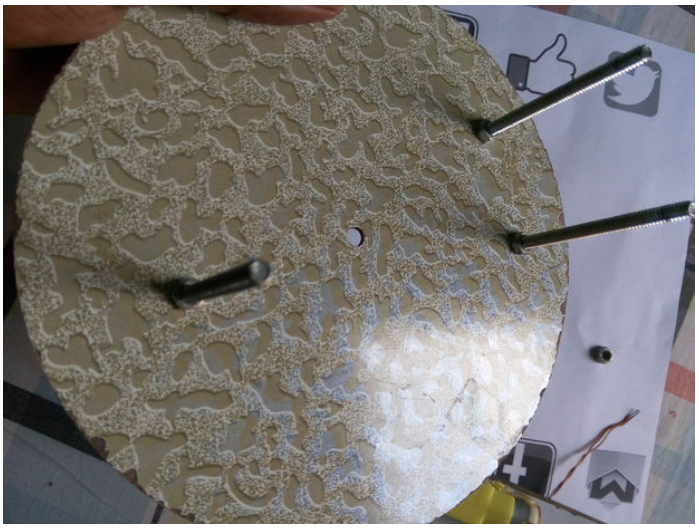


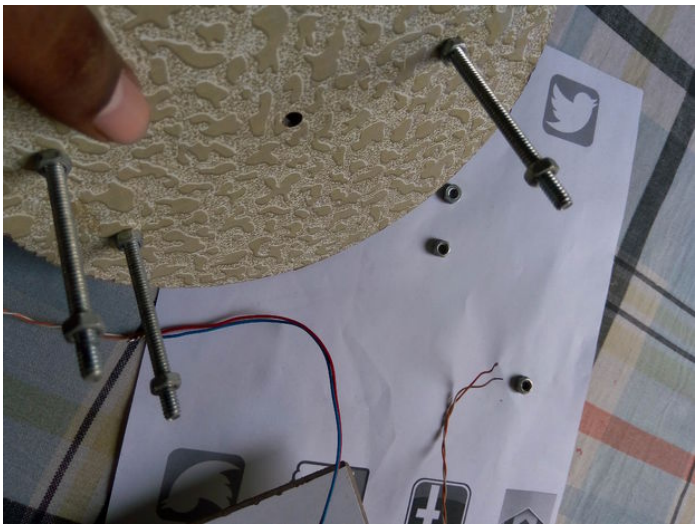


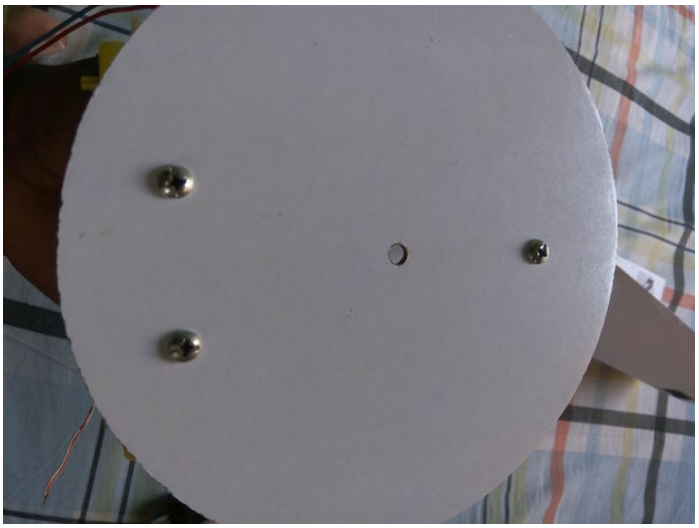










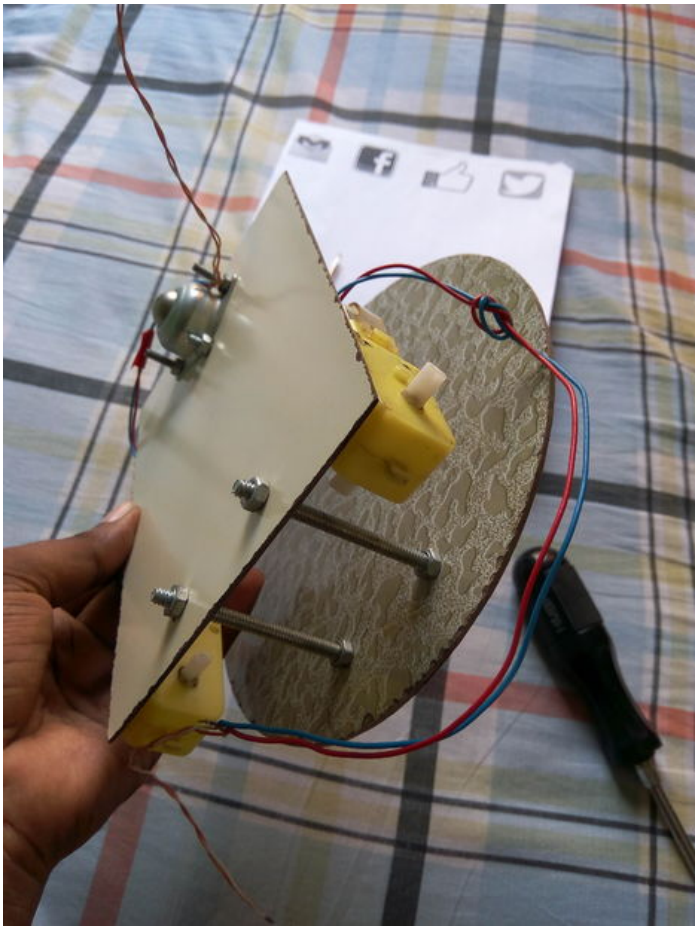


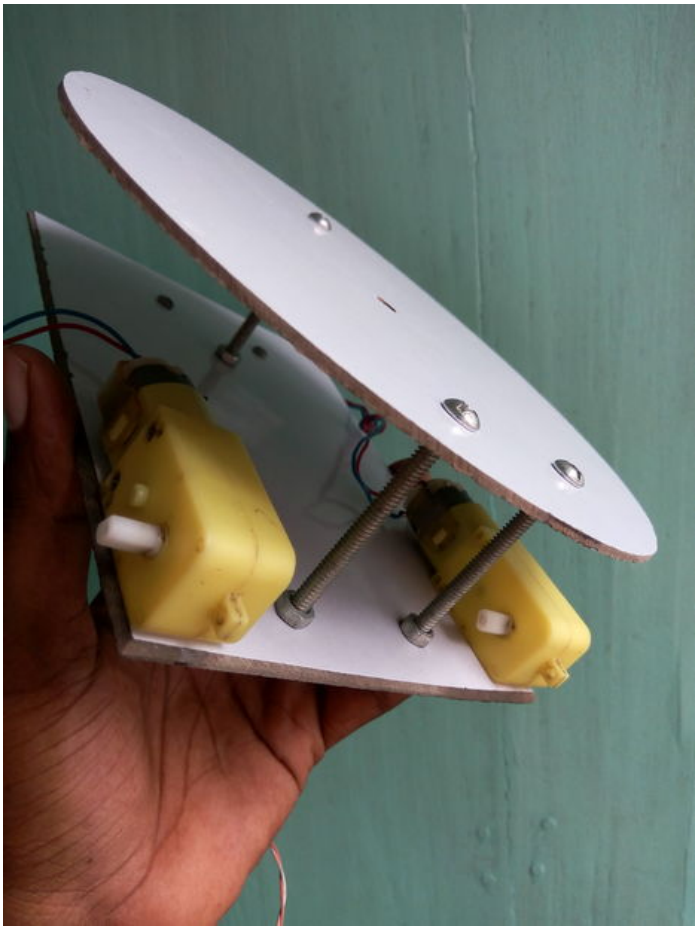
Step 3: Make Chassis (part 2)

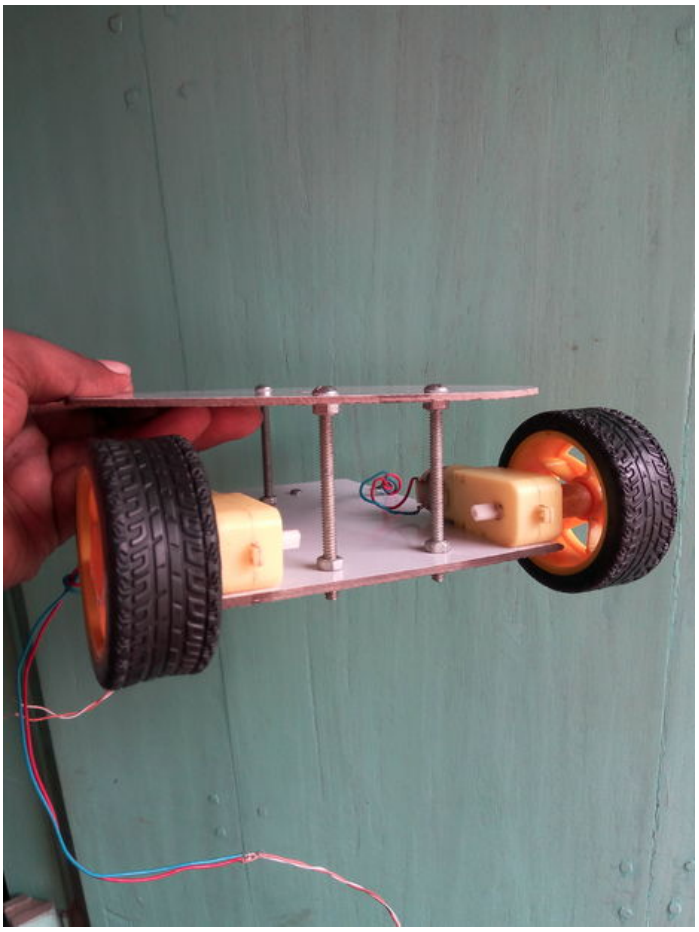
Using 3 inches nuts fix two board together as like as figure. Nuts can be smaller then 3 inch or bigger than 3 inches, it deepens on your requirements and choice. Use supper glue to connect two gear motor to the bottom board. After connecting two boards and motor connect the wheel to the shaft of the motor.

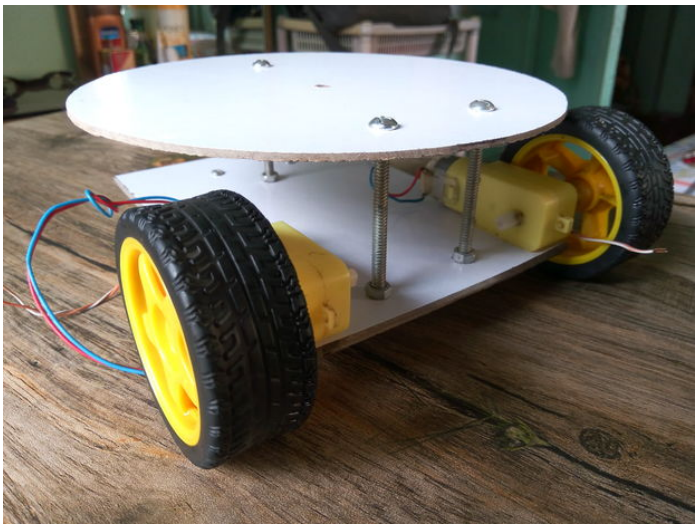
Description: HC-05 is a class-2 bluetooth module with Serial Port Profile, which can configure as either Master or slave. a Drop-in replacement for wired serial connections, transparent usage. You can use it simply for a serial port replacement to establish connection between MCU, PC to your embedded project and etc.

HC-05 Specification: Bluetooth protocal: Bluetooth Specification v2.0 EDRFrequency: 2.4GHz ISM bandModulation: GFSK(Gaussian Frequency Shift Keying)Emission power: â‰¤4dBm, Class 2Sensitivity: â‰¤-84dBm at 0.1% BERSpeed: Asynchronous: 2.1Mbps(Max) / 160 kbps, Synchronous: 1Mbps/1MbpsSecurity: Authentication ar encryptionProfiles: Bluetooth serial portPower supply: 3.3VDC 50mAWorking temperature: -20 ~ 75CentigradeDimension: 26.9mm x 13mm x 2.2 mm







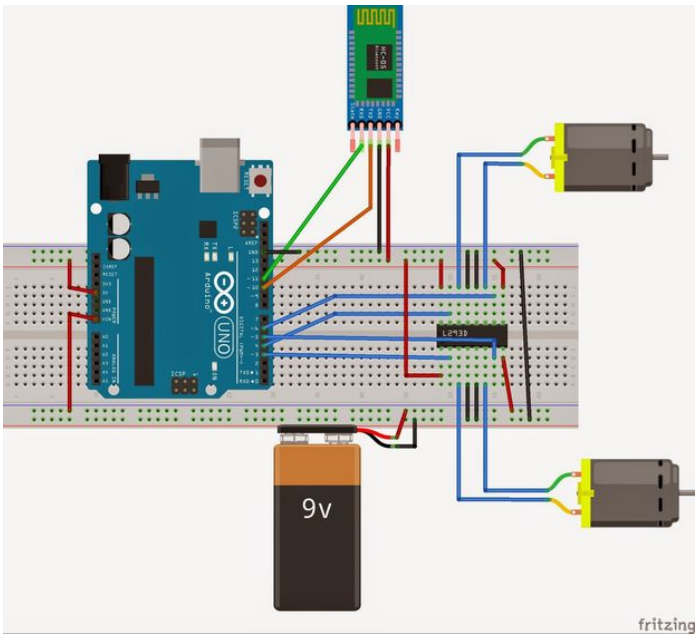


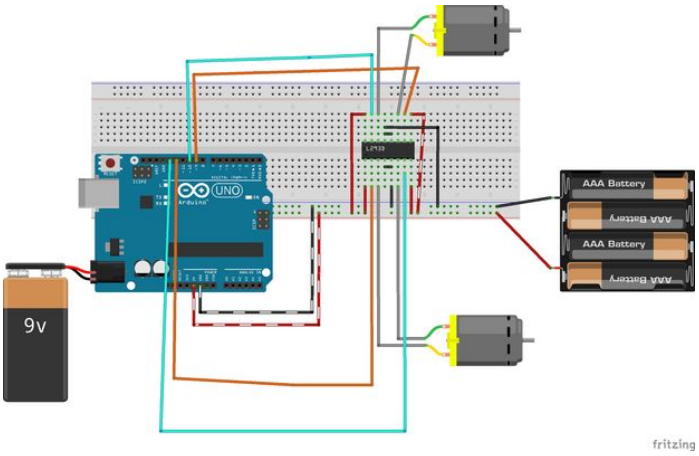
Step 4: Schematic & Layout

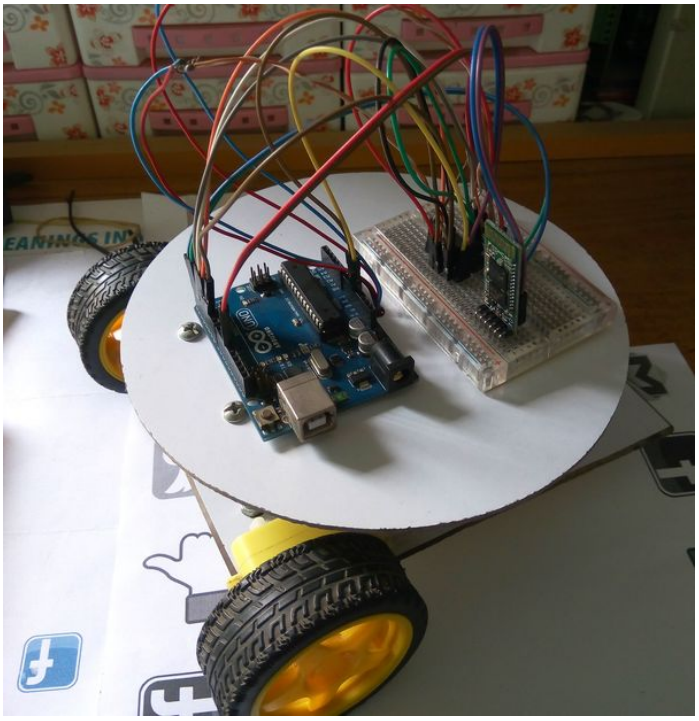
Circuit connection is very simple for the robot. We need to connect two motor to the Arduino board but one important thing is that Arduino pin can not drive a motor directly. So, we need to add transistor between them. Both speed & direction control are required for a robot and for that we need 4 transistors to drive a motor. It is easy to use a motor driver IC instead of transistor. I used most popular dual H-bridge motor driver IC L293D here for driving two motors. This IC has two enable input which need to connect any pwm pin of Arduino to control the speed of a motor.

For receiving command from the android phone I used bluetooth HC-05 module with the arduino. You need to connect TX pin of Arduino pin with the RX pin of bluetooth receiver & RX pin of arduino pin with the TX pin of bluetooth receiver. This seems a little bit confusing, but logic is simple. Data transmitted by bluetooth module is received by Arduino (TX-RX) and data transmitted by arduino is received by bluetooth module.

Fritzing layouts are attached. A 9V battery is connected to powering the robot.

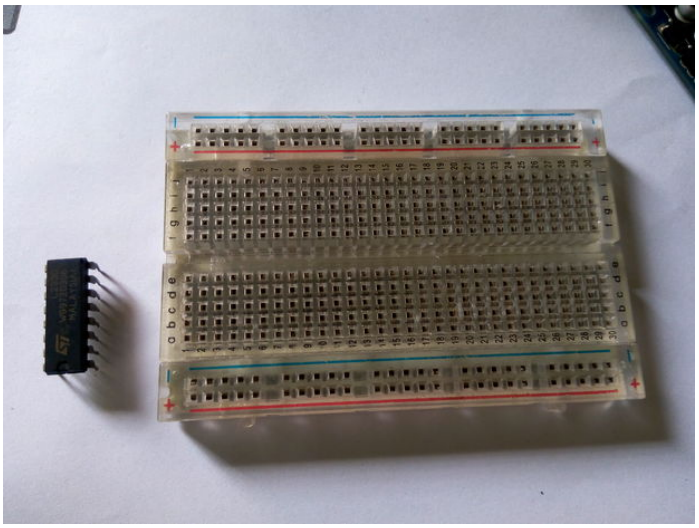


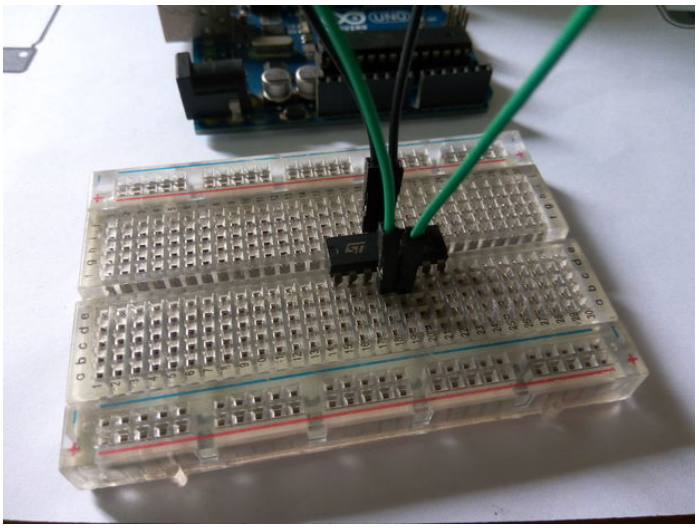


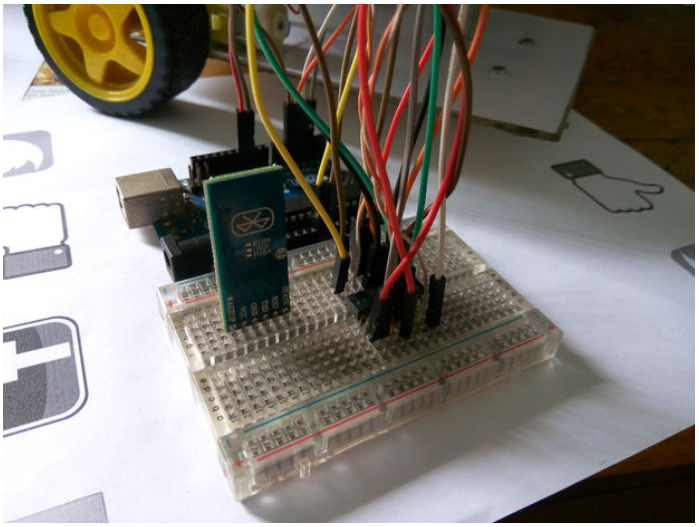


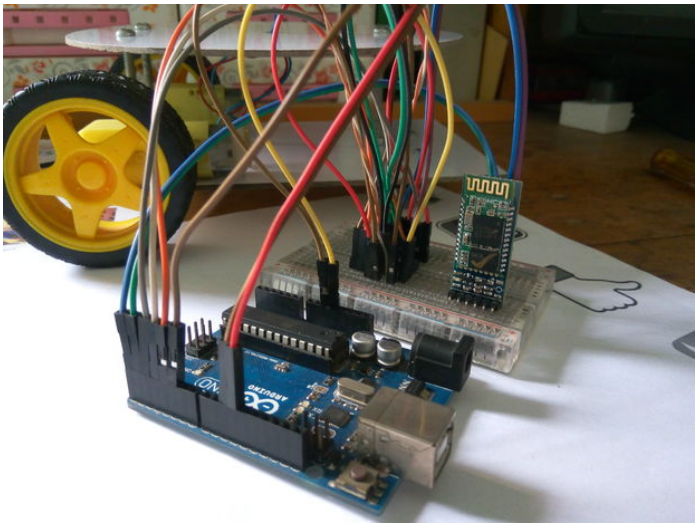
Step 5: Connecting all Electronics

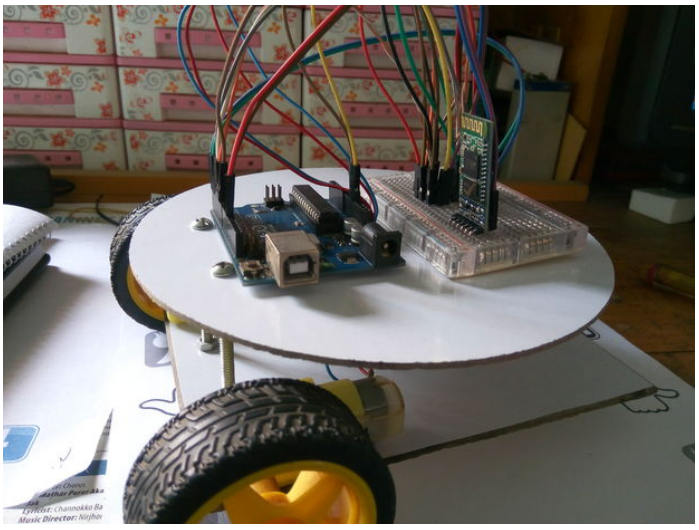
In this step we will make the circuit. Take a mini breadboard and attached motor driver IC first. Then connect the Arduino board with the motor driver IC using jumper wires. Make sure about the connections. Then connect bluetooth module to the arduino board. At last, connect two motors to the motor driver IC. Put the complete circuit into the chassis.











Step 6: Making App

Bluetooth is the communications technology with a funny name.[1] Bluetooth is actually named for a long ago Danish king who worked to unite groups of people, which is similar to Bluetooth's goal of interconnecting different devices. The King's real name was "Harald" but he had a nickname that translates as "Bluetooth" – no one knows for sure why he had this nickname but one thought is he had one dark tooth that may have appeared black or blue. And that is certainly an obscure way to choose a name for new technologies!

Bluetooth establishes a very low power, short range (up to 10 meters) communications link between two devices. Bluetooth uses the same frequency band (2.4 Ghz) as Wi-Fi, but uses different technology. Both Bluetooth and Wi-Fi use forms of spread spectrum radio links that result in signals moving around within a wide band in ways that enable sharing of the spectrum by multiple devices. But the two technologies serve different purposes, are not identical, and cannot communicate with one another. Bluetooth applications include common wireless headsets for wired and cellular phones, and in-ear cordless adapters for phones. Bluetooth is also used by cordless headphones and to exchange address cards between devices, and for industrial applications where sensors collect and send data into a network. There are two forms of Bluetooth – classic Bluetooth, which we use in the sample applications, and a newer version known as Bluetooth low energy, Bluetooth BLE, Bluetooth LE or Bluetooth Smart – all referring to the same new technology. The newest Android devices running Android 4.3 or newer, usually support the newest Bluetooth Smart technology. Regardless, we use classic Bluetooth which is backwards compatible to older phones, and is the technology supported by App Inventor. Setting up a Bluetooth devices involves "pairing" the two devices and establishing a connection. This will be covered later in this tutorial.

There are two separate apps for Bluetooth communications – one is a "server" app that runs on one device, and the other is a "client" app that runs on a second device.
Bluetooth must be enabled on both devices, and the devices need to be paired before running these apps. (How to do this is explained near the end of this tutorial.) The server must be run first on one device and then the client app on the 2nd device connects to the server before data can be sent between the two devices. More on this later in this tutorial.

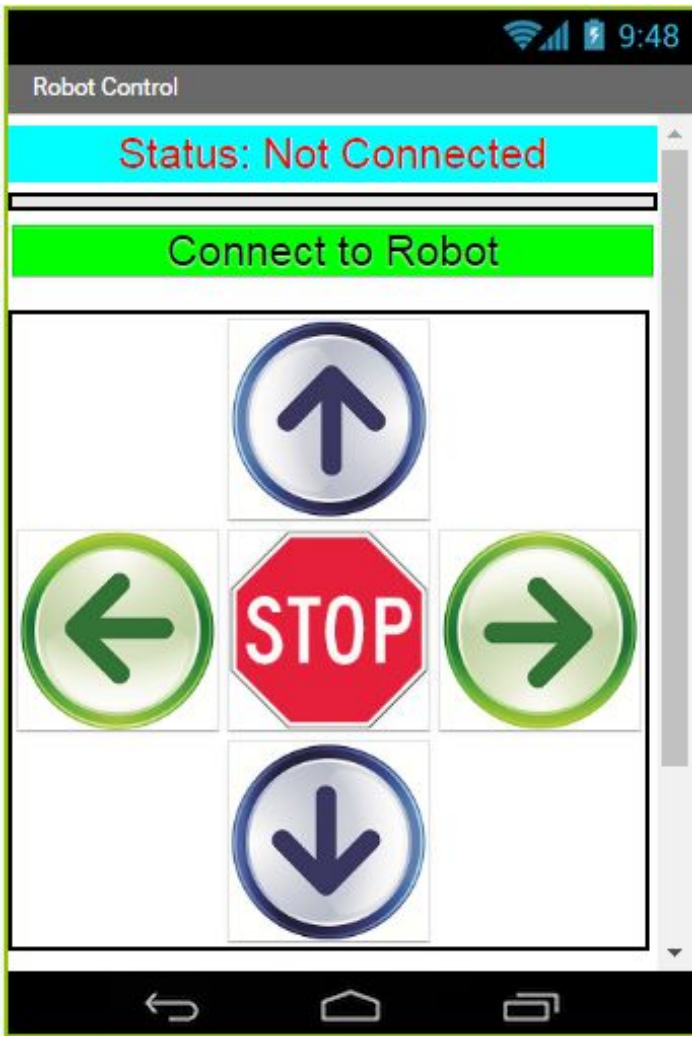
The main components of the server interface design are:
Accept Connection Button – press this to set the server to accept a connection from another device. Connections are not possible until the Accept Connection service is started.

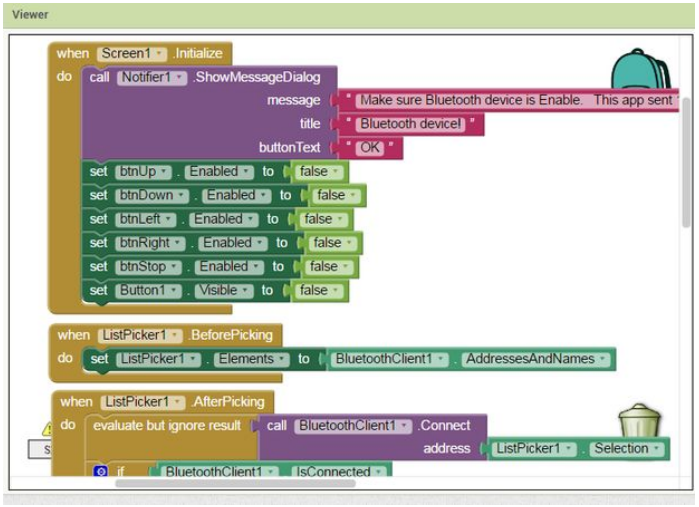
Send the following text Button – the text is the following text box is sent to the other Bluetooth device.

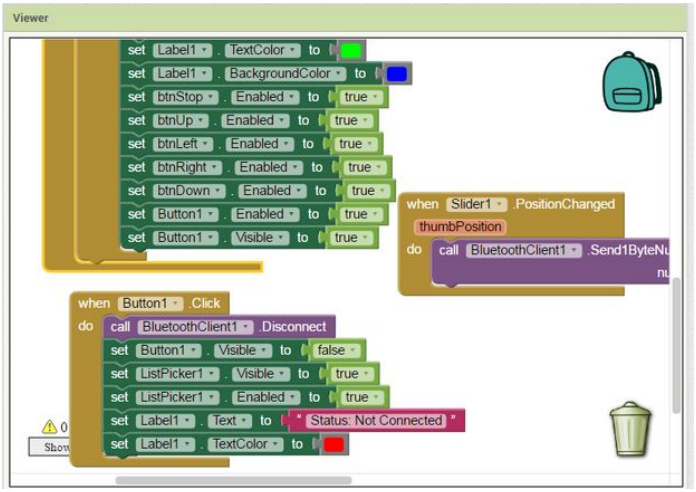
Disconnect Button

Status messages – Status about the communications link, and any messages received from the other device are shown on the display

Non-visible components – The apps use a clock to cause activities to occur at a preset interval. The Notifier1 component is used to display error messages (see tutorial on the use of Notifier), and BluetoothServer1 provides the Bluetooth support. The BluetoothClient1 and BluetoothServer1 components are located in the Connectivity section of the Designer palette. http://appinventor.pevest.com/?p=520







Step 7: Uploading Program

You made all the connection? Just upload the program to your Arduino Program

```
/*
* created by Md. Khairul Alam
* Control 2 DC motors with Smartphone via bluetooth
* 2015
*/
int motor1Pin1 = 3; // pin 2 on L293D IC
int motor1Pin2 = 4; // pin 7 on L293D IC
int motor1EnablePin = 6; // pin 1 on L293D IC
int motor2Pin1 = 8; // pin 10 on L293D IC
int motor2Pin2 = 9; // pin 15 on L293D IC
int motor2EnablePin = 11; // pin 9 on L293D IC
int state;
int Speed = 100;
int flag=0;        //makes sure that the serial only prints once the state
int stateStop=0;
void setup() {
   // sets the pins as outputs:
   pinMode(motor1Pin1, OUTPUT);
   pinMode(motor1Pin2, OUTPUT);
   pinMode(motor1EnablePin, OUTPUT);
   pinMode(motor2Pin1, OUTPUT);
   pinMode(motor2Pin2, OUTPUT);
   pinMode(motor2EnablePin, OUTPUT);
   // sets enable1Pin and enable2Pin high so that motor can turn on:
   //digitalWrite(Motor1EnablePin, HIGH);
   //digitalWrite(Motor2EnablePin, HIGH);
   // initialize serial communication at 9600 bits per second:
   Serial.begin(9600);
}

void loop() {
   //if some date is sent, reads it and saves in state
   if(Serial.available() > 0){
     state = Serial.read();
     if(state > 5){
        Speed = state;}
     flag=0;
   }
   // if the state is '1' the DC motor will go forward
   if (state == 1) {
       forword();
       if(flag == 0){
         Serial.println("Go Forward!");
         flag=1;
       }
   }

   // if the state is '2' the motor will turn left
   else if (state == 2) {
       turnLeft();
       if(flag == 0){
         Serial.println("Turn LEFT");
         flag=1;
       }
       delay(1500);
       state=3;
       stateStop=1;
   }
   // if the state is '3' the motor will Stop
   else if (state == 3 || stateStop == 1) {
       Stop();
       if(flag == 0){
         Serial.println("STOP!");
         flag=1;
       }
       stateStop=0;
   }
```

```
   // if the state is '4' the motor will turn right
   else if (state == 4) {
       turnRight();
       if(flag == 0){
         Serial.println("Turn RIGHT");
         flag=1;
       }
       delay(1500);
       state=3;
       stateStop=1;
   }
   // if the state is '5' the motor will Reverse
   else if (state == 5) {
       backword();
       if(flag == 0){
         Serial.println("Reverse!");
         flag=1;
       }
   }
   //For debugging purpose
   //Serial.println(state);
}

void forword(){
   digitalWrite(motor1Pin1, HIGH);
   digitalWrite(motor1Pin2, LOW);
   digitalWrite(motor2Pin1, LOW);
   digitalWrite(motor2Pin2, HIGH);
   analogWrite(motor1EnablePin, Speed);
   analogWrite(motor2EnablePin, Speed);
   //Serial.println("Go Forward!");
}

void backword(){
   digitalWrite(motor1Pin1, LOW);
   digitalWrite(motor1Pin2, HIGH);
   digitalWrite(motor2Pin1, HIGH);
   digitalWrite(motor2Pin2, LOW);
   analogWrite(motor1EnablePin, Speed);
   analogWrite(motor2EnablePin, Speed);
   //Serial.println("Go Reverse!");

}

void turnRight(){
   digitalWrite(motor1Pin1, LOW);
   digitalWrite(motor1Pin2, LOW);
   digitalWrite(motor2Pin1, LOW);
   digitalWrite(motor2Pin2, HIGH);
   analogWrite(motor1EnablePin, Speed);
   analogWrite(motor2EnablePin, Speed);
   //Serial.println("Turn Right");

}

void turnLeft(){
   digitalWrite(motor1Pin1, HIGH);
   digitalWrite(motor1Pin2, LOW);
   digitalWrite(motor2Pin1, LOW);
   digitalWrite(motor2Pin2, LOW);
   analogWrite(motor1EnablePin, Speed);
   analogWrite(motor2EnablePin, Speed);
   //Serial.println("Turn Left");

}

void Stop(){
   digitalWrite(motor1Pin1, LOW);
   digitalWrite(motor1Pin2, LOW);
   digitalWrite(motor2Pin1, LOW);
   digitalWrite(motor2Pin2, LOW);
   analogWrite(motor1EnablePin, Speed);
   analogWrite(motor2EnablePin, Speed);
   //Serial.println("Stop");

}
```

File Edit Sketch Tools Help

Control_2_DC_Motors_Via_Bluetooth

```
int motor1Pin2 = 4; // pin 7 on L293D IC
int motor1EnablePin = 6; // pin 1 on L293D IC
int motor2Pin1 = 8; // pin 10 on L293D IC
int motor2Pin2 = 9; // pin 15 on L293D IC
int motor2EnablePin = 11; // pin 9 on L293D IC
int state;
int Speed = 100;
int flag=0;        //makes sure that the serial only prints once t
int stateStop=0;
void setup() {
    // sets the pins as outputs:
    pinMode(motor1Pin1, OUTPUT);
    pinMode(motor1Pin2, OUTPUT);
    pinMode(motor1EnablePin, OUTPUT);
    pinMode(motor2Pin1, OUTPUT);
    pinMode(motor2Pin2, OUTPUT);
    pinMode(motor2EnablePin, OUTPUT);
    // sets enable1Pin and enable2Pin high so that motor can turn
```

Control_2_DC_Motors_Via_Bluetooth

```
void loop() {
    //if some date is sent, reads it and saves in state
    if(Serial.available() > 0){
      state = Serial.read();
      if(state > 5){
         Speed = state;}
      flag=0;
    }
    // if the state is '1' the DC motor will go forward
    if (state == 1) {
        forword();
        if(flag == 0){
          Serial.println("Go Forward!");
          flag=1;
        }
    }

    // if the state is '2' the motor will turn left
```

Control_2_DC_Motors_Via_Bluetooth

```
void turnRight(){
    digitalWrite(motor1Pin1, LOW);
    digitalWrite(motor1Pin2, LOW);
    digitalWrite(motor2Pin1, LOW);
    digitalWrite(motor2Pin2, HIGH);
    analogWrite(motor1EnablePin, Speed);
    analogWrite(motor2EnablePin, Speed);
    //Serial.println("Turn Right");

}

void turnLeft(){
    digitalWrite(motor1Pin1, HIGH);
    digitalWrite(motor1Pin2, LOW);
    digitalWrite(motor2Pin1, LOW);
    digitalWrite(motor2Pin2, LOW);
    analogWrite(motor1EnablePin, Speed);
    analogWrite(motor2EnablePin, Speed);
    //Serial.println("Turn Left");

}

void Stop(){
    digitalWrite(motor1Pin1, LOW);
    digitalWrite(motor1Pin2, LOW);
```

File Downloads

Control_2_DC_Motors_Via_Bluetooth.ino (3 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'Control_2_DC_Motors_Via_Bluetooth.ino']

Related Instructables

DIY RC Robot with your Android Phone Tutorial Part 1: from Android phone to the microcontroller by Practical Techonlogy

myRobot - DIY Robot by aneophyte

Robotics guide by vonPongrac

DIY RC Robot with Your Android Phone Tutorial Part 2: Bluetooth/Microc Parts & Wiring by Practical Techonlogy

[DIY] Spider Robot - PART II - Remote control by RegisHsu

Robot Attiny2313 With Bluetooth HC06 and Stepper Motors by Bogus Cichocki

Comments