

## Project 1: Line Tracking Sensor



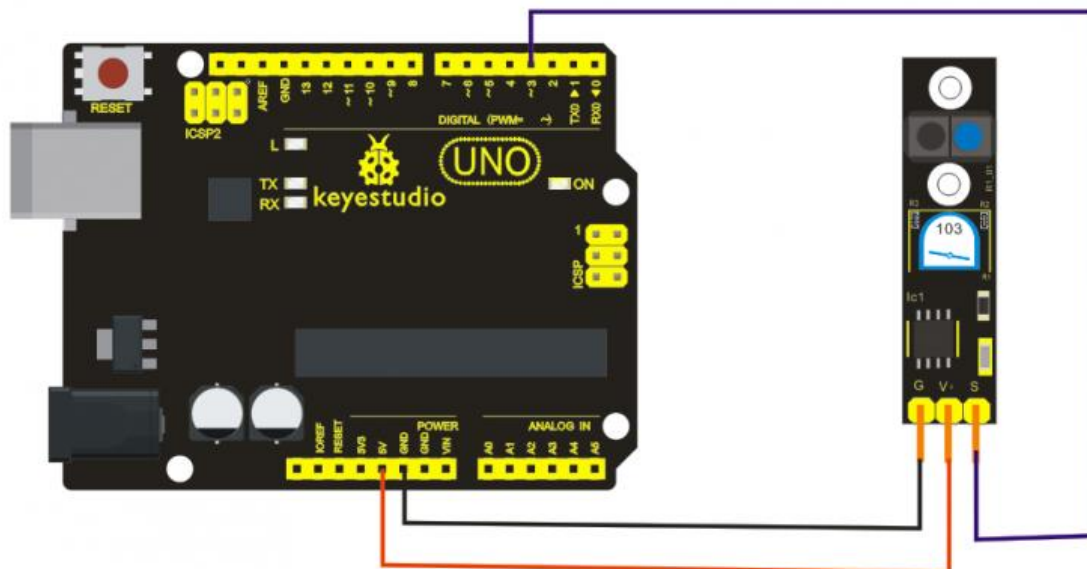
### Introduction:

This **Line Tracking Sensor** can detect white lines in black and black lines in white. The single line-tracking signal provides a stable output signal TTL for a more accurate and more stable line. Multi-channel option can be easily achieved by installing required line-tracking robot sensors.

### Specification:

- Power Supply: +5V
- Operating Current: <10mA
- Operating Temperature Range: 0°C ~ + 50°C
- Output Interface: 3-wire interface (1 - signal, 2 - power, 3 - power supply negative)
- Output Level: TTL level

### Connection Diagram:



### Sample Code:

```
const int sensorPin = 3;    // the number of the sensor pin
const int ledPin = 13;     // the number of the LED pin
int sensorState = 0;       // variable for reading the sensor status
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(sensorPin, INPUT); }

```

```
void loop(){
  // read the state of the sensor value:
  sensorState = digitalRead(sensorPin);
  // if the sensorState is HIGH:
  if (sensorState == HIGH) {
    digitalWrite(ledPin, HIGH);
  }
  else {digitalWrite(ledPin, LOW);
  }}
}
```

**Result:**

After power-on, power indicator D1 is on. When you block the sensing part of line tracking sensor with black paper, LED on the sensor is off as shown in Figure 1. When you block it with white paper, LED is on as shown in Figure 2.

**Figure 1:**

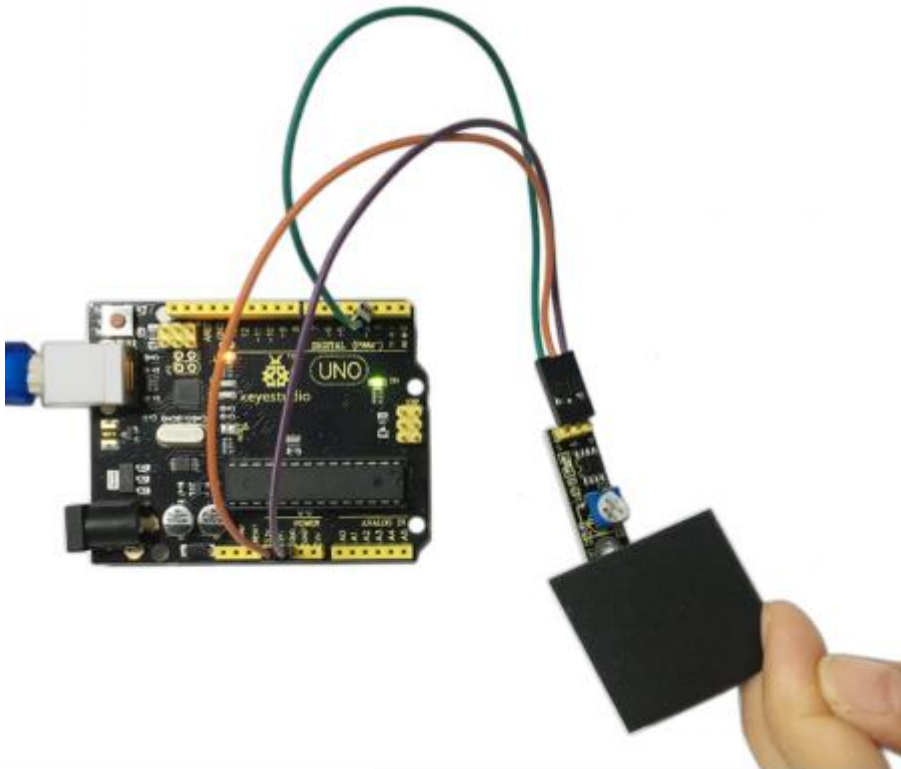
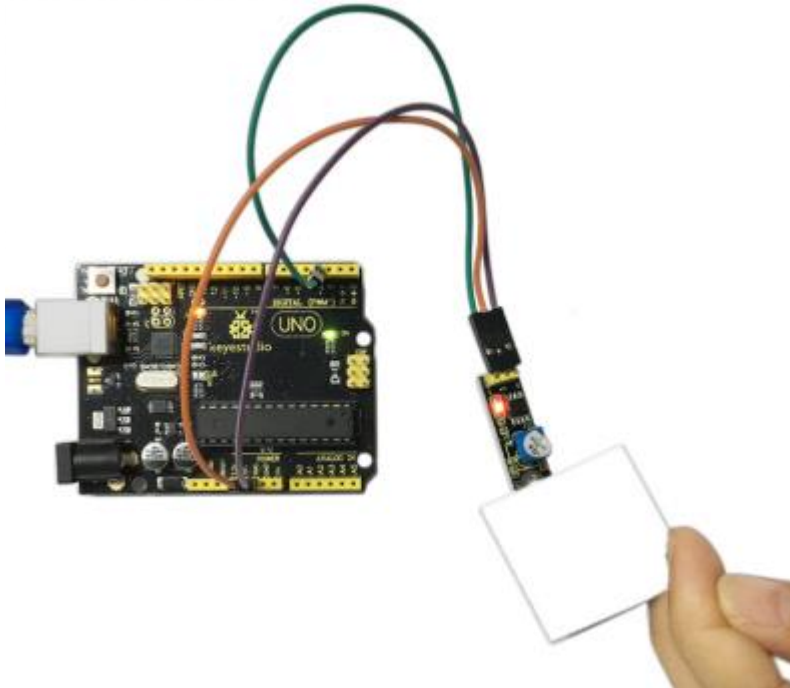


Figure 2:



## Project 2: Ultrasonic Sensor



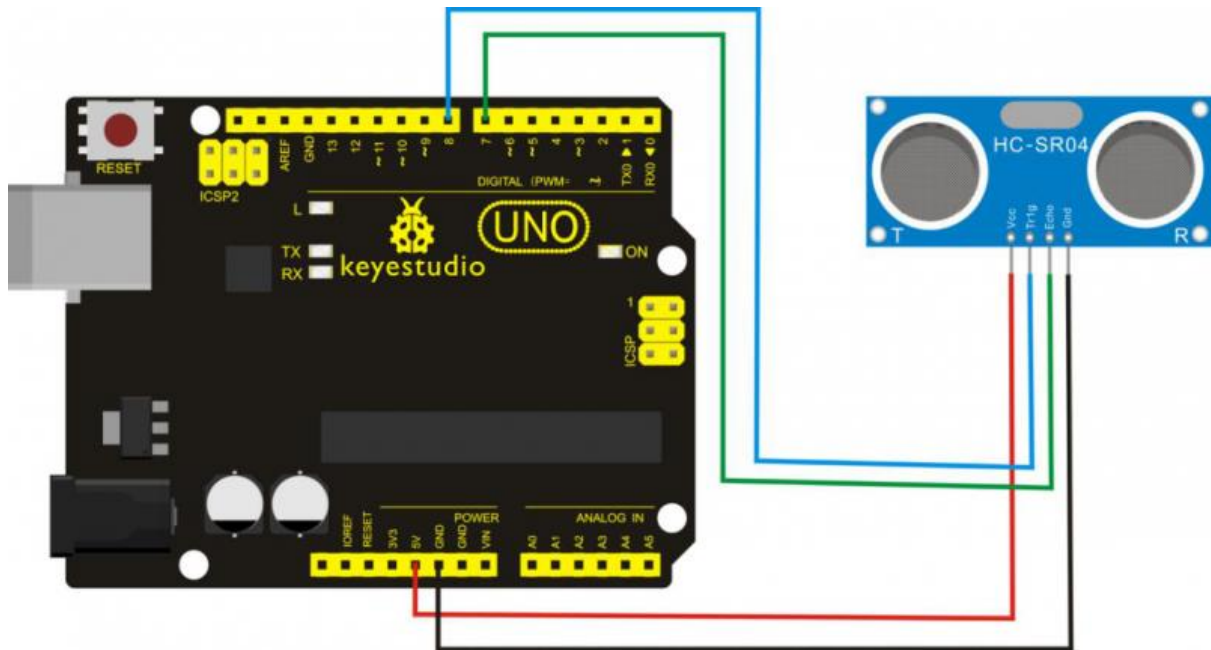
### Introduction:

The HC-SR04 Ultrasonic Sensor is a very affordable proximity/distance sensor that has been used mainly for object avoidance in various robotics projects. It essentially gives your Arduino eyes/ special awareness and can prevent your robot from crashing or falling off a table. It has also been used in turret applications, water level sensing, and even as a parking sensor. This simple project will use the HC-SR04 sensor with an Arduino and a processing sketch to provide a neat little interactive display on your computer screen.

### Specification :

- Working Voltage: DC 5V
- Working Current: 15mA
- Working Frequency: 40Hz
- Max Range: 4m
- Min Range: 2cm
- Measuring Angle: 15 degree
- Trigger Input Signal: 10 $\mu$ S TTL pulse
- Echo Output Signal Input TTL lever signal and the range in proportion
- Size: 46\*20.4mm
- Weight: 9g

### Connection Diagram:



### Sample Code:

VCC to arduino 5v  
 GND to arduino GND  
 Echo to Arduino pin 7  
 Trig to Arduino pin 8

```
#define echoPin 7 // Echo Pin
#define trigPin 8 // Trigger Pin
#define LEDPin 13 // Onboard LED

int maximumRange = 200; // Maximum range needed
int minimumRange = 0; // Minimum range needed
long duration, distance; // Duration used to calculate distance

void setup() {
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(LEDPin, OUTPUT); // Use LED indicator (if required)
}

void loop() {
  /* The following trigPin/echoPin cycle is used to determine the
  distance of the nearest object by bouncing soundwaves off of it. */
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
```

```
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);

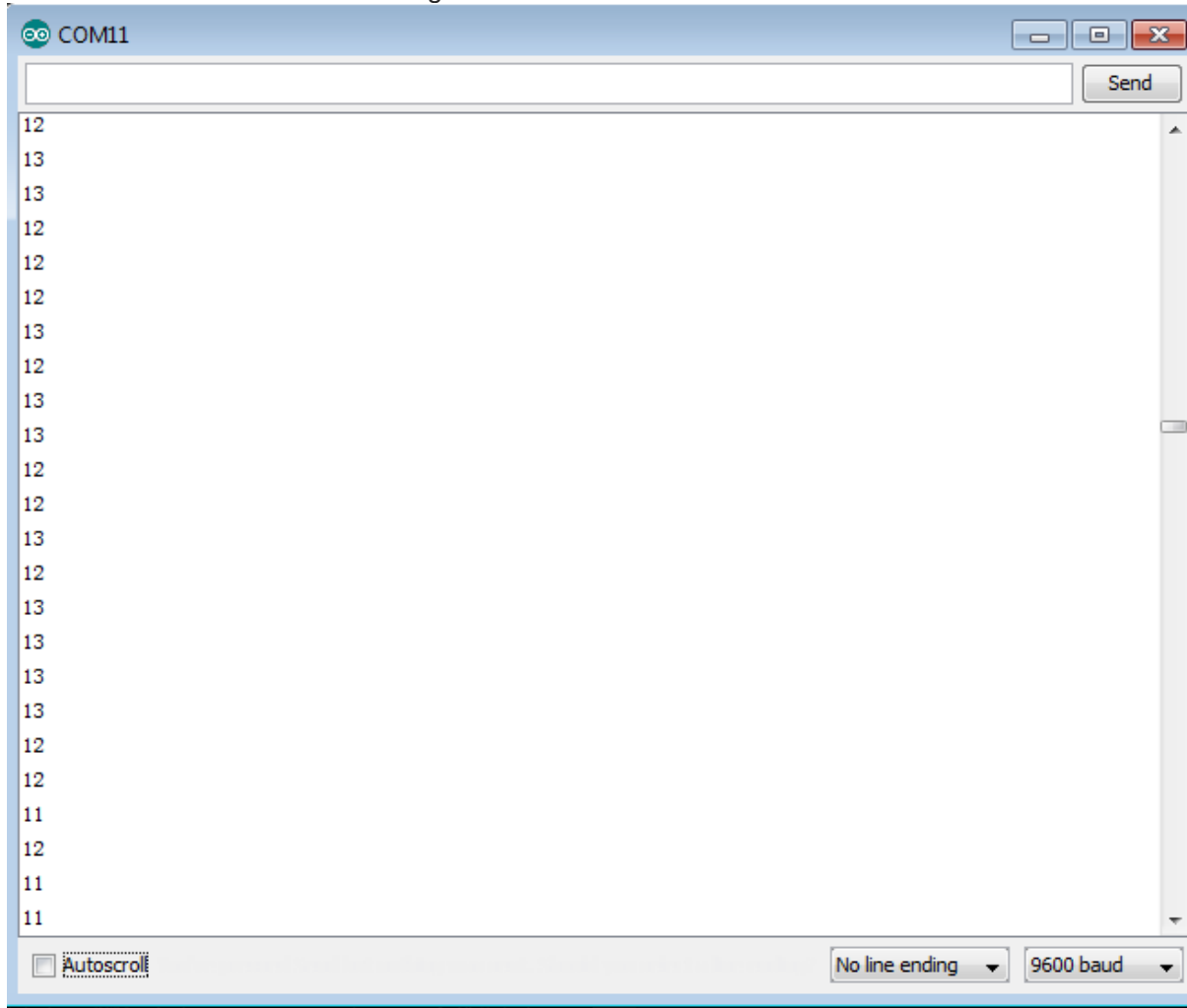
//Calculate the distance (in cm) based on the speed of sound.
distance = duration/58.2;

if (distance >= maximumRange || distance <= minimumRange){
/* Send a negative number to computer and Turn LED ON
to indicate "out of range" */
Serial.println("-1");
digitalWrite(LEDpin, HIGH);
}
else {
/* Send the distance to the computer using Serial protocol, and
turn LED OFF to indicate successful reading. */
Serial.println(distance);
digitalWrite(LEDpin, LOW);
}
//Delay 50ms before next reading.
delay(50);
}
```

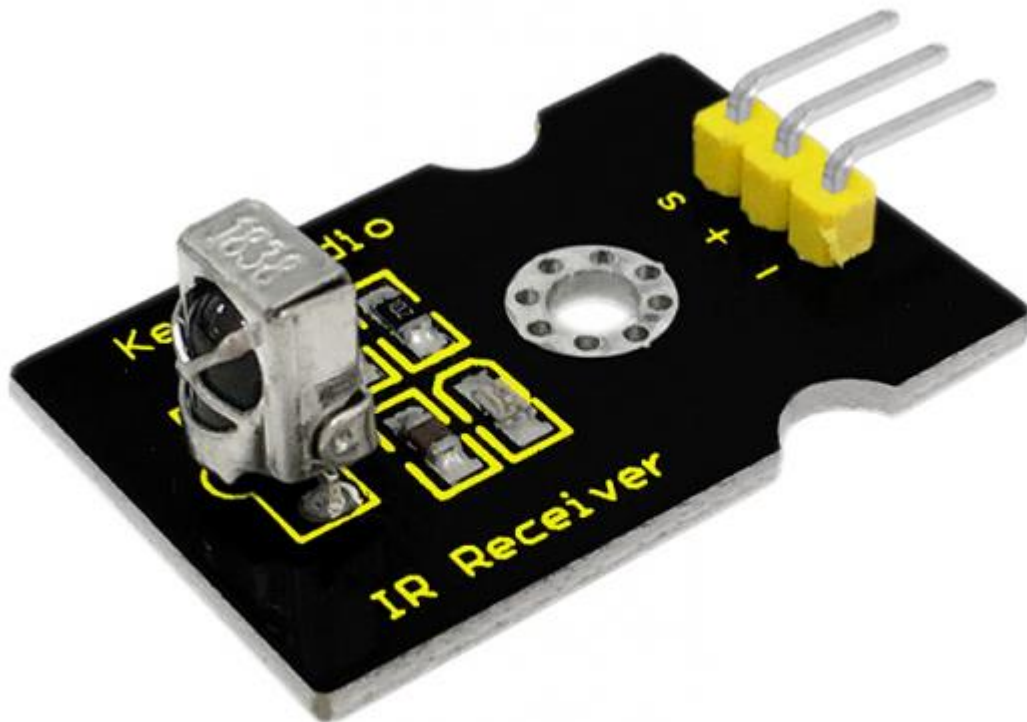
**Result:**

After connection and uploading, when ultrasonic sensor senses obstacle within sensing area, it is measuring the distance between itself and obstacle and the value of distance is displayed on

serial monitor as shown in below figure.



### Project 3: Digital IR Receiver Module

**Introduction:**

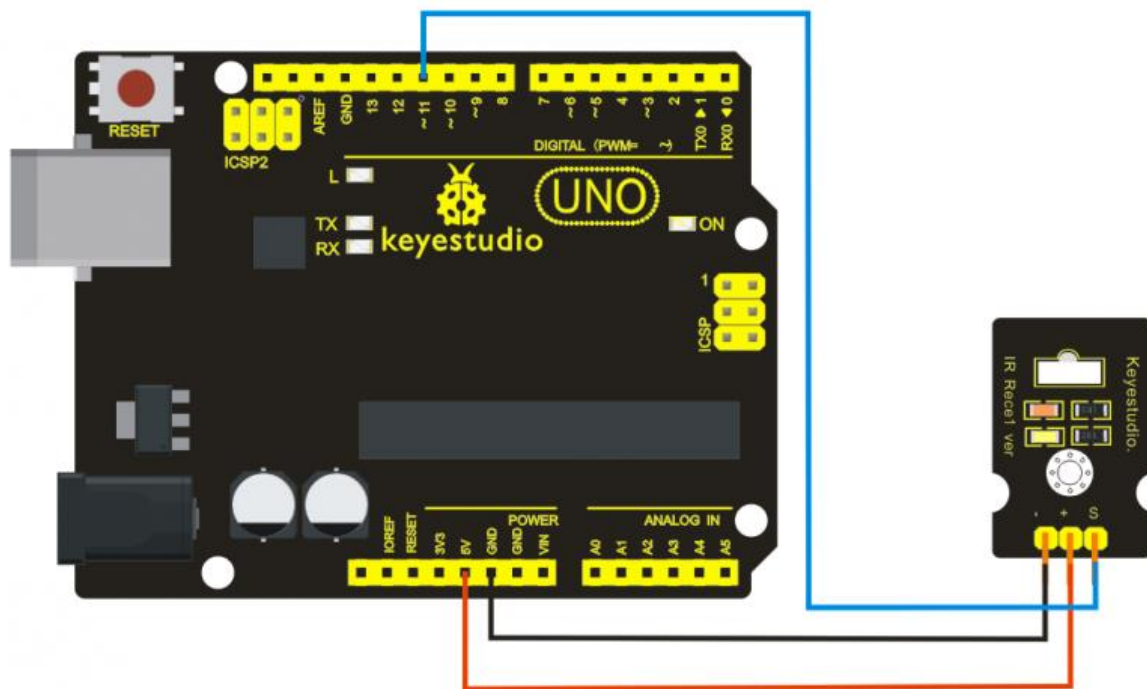
IR is widely used in remote control. With this [IR receiver](#) , Arduino project is able to receive command from any IR remoter controller if you have the right decoder. Well, it will be easy to make your own IR controller using IR transmitter.

**Specification:**

- Power Supply: 5V
- Interface: Digital
- Modulate Frequency: 38Khz
- Module Interface Socket: JST PH2.0

NOTE: In the sample code below Digital pin 11 is in use, you may either change your wiring or change the sample code to match.

**Connection Diagram :**



### Sample Code:

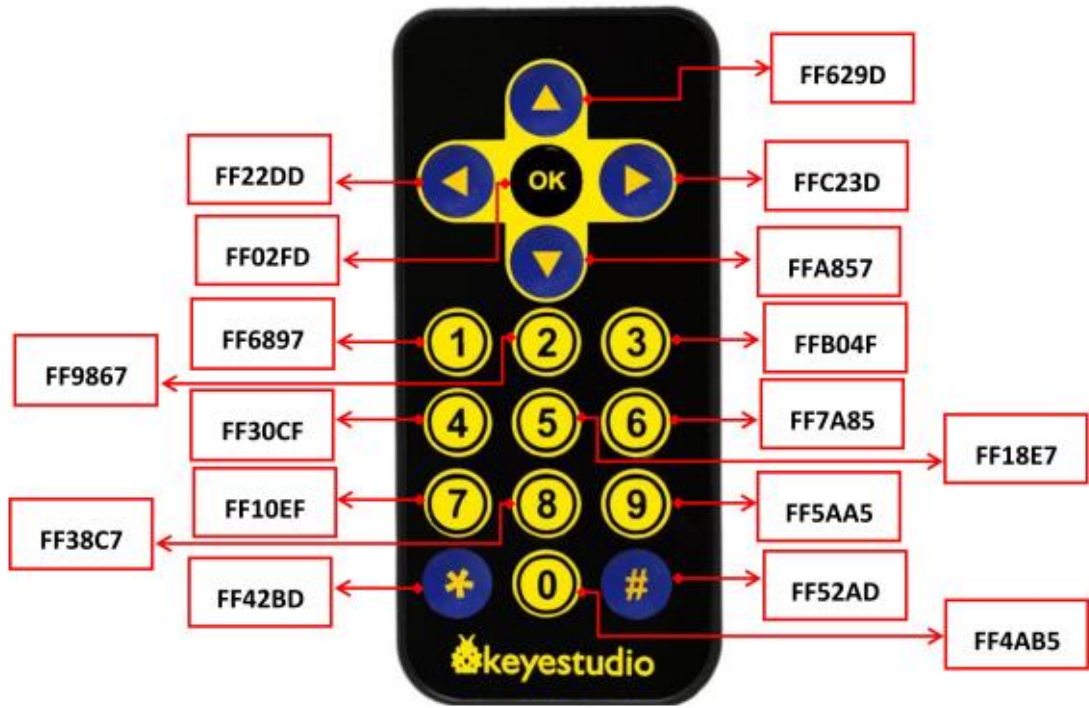
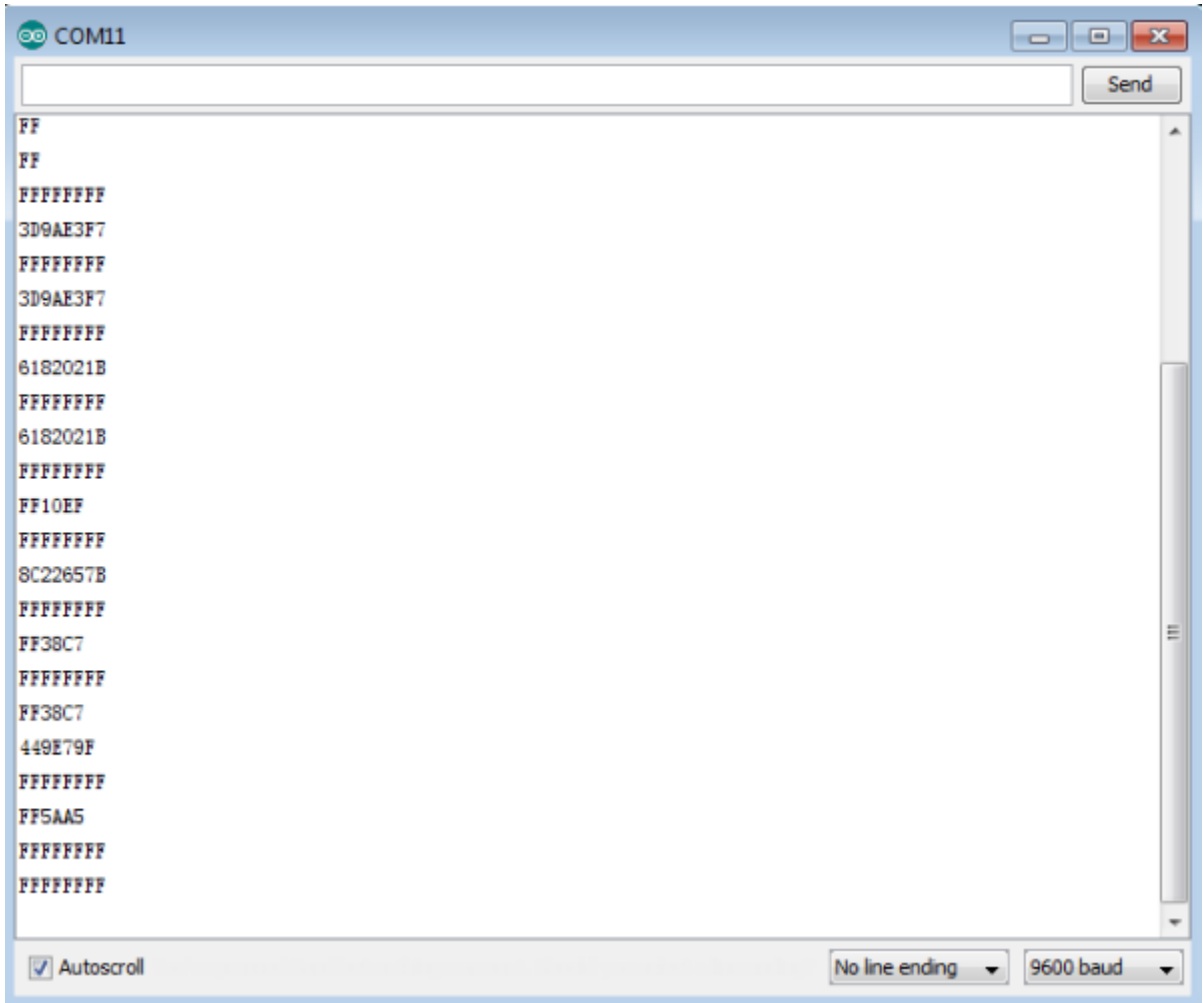
```
#include <IRremote.h>
int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
}
void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
  }
}
```

IR Remote Library includes some sample codes for sending and receiving.

<https://github.com/shirriff/Arduino-IRremote>

**Result:** In this project, we need to use a IR remote control which has 17 functional key and its launching distance is 8 meters at most, proper to control various devices indoors. This project is able to decode remote control signal. After connection and uploading codes, aim at IR receiving module and press the key, finally you can see the corresponding codes. If you press the key too long, it will show messy codes easily as shown in below figure.





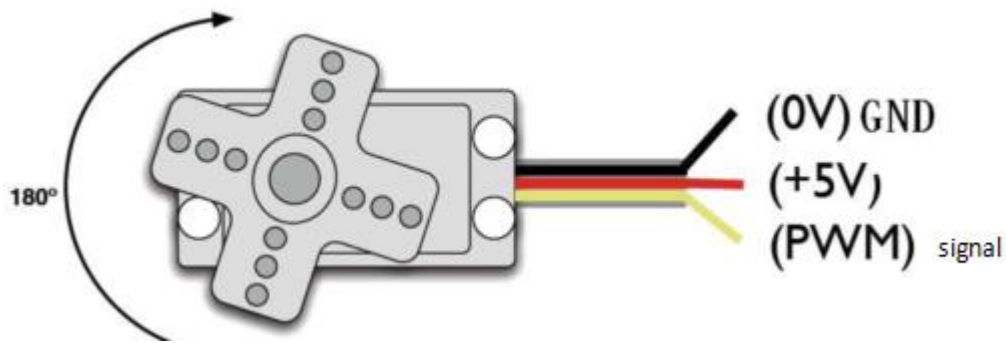
## Project 4: Servo Motor



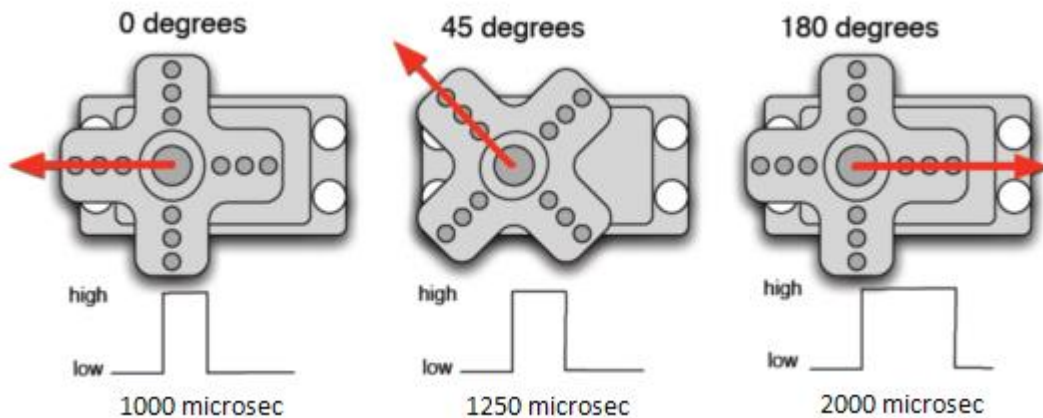
### Introduction:

Servomotor is a position control rotary actuator. It mainly consists of housing, circuit board, core-less motor, gear and position sensor. The receiver or MCU outputs a signal to the servomotor. The motor has a built-in reference circuit that gives out reference signal, cycle of 20ms and width of 1.5ms. The motor compares the acquired DC bias voltage to the voltage of the potentiometer and outputs a voltage difference. The IC on the circuit board will decide the rotate direction accordingly and drive the core-less motor. The gear then passes the force to the shaft. The sensor will determine if it has reached the commanded position according to the feedback signal. Servomotors are used in control systems that require to have and maintain different angles. When the motor speed is definite, the gear will cause the potentiometer to rotate. When the voltage difference reduces to zero, the motor stops. Normally, the rotation angle range is among 0-180 degrees.

Servomotor comes with many specifications. But all of them have three connection wires, distinguished by brown, red, orange color(different brand may have different color). Brown one is for GND, red one for power positive, orange one for signal Line.



The rotating angle of the servo motor is controlled by regulating the duty cycle of the PWM(Pulse-Width Modulation) signal. The standard cycle of the PWM signal is 20ms (50Hz) . Theoretically, the width is distributed between 1ms-2ms, but in fact, it's between 0.5ms-2.5ms. The width corresponds the rotate angle from 0° to 180°. But note that for different brand motor, the same signal may have different rotating angle.

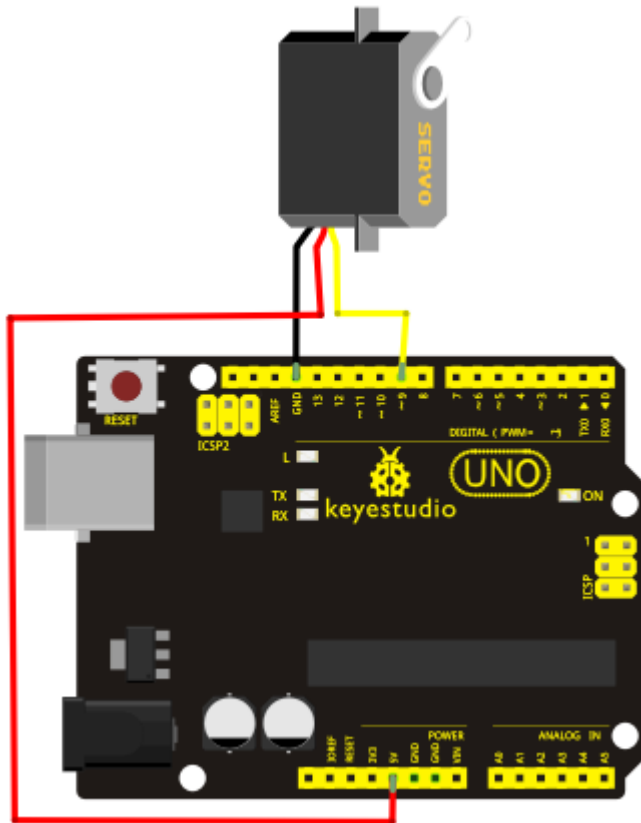


Mastering some basic knowledge, let's learn how to control a servomotor. In this experiment, you only need a servomotor and several jumper wires.

### Connection & Sample Program:

There are two ways to control a servomotor with Arduino. One is to use a common digital sensor port of Arduino to produce square wave with different duty cycle to simulate PWM signal and then use that signal to control the positioning of the motor. Another way is to directly use the Servo function of the Arduino to control the motor. In this way, the program will be easier, but it can only control two-contact motor because of the servo function, only digital pin 9 and 10 can be used. The Arduino drive capacity is limited. So, if you need to control more than one motor, you will need external power.

### Connection Diagram :



### Sample Code :

```
int servopin=9;// select digital pin 9 for servomotor signal line
int myangle;// initialize angle variable
int pulsewidth;// initialize width variable
int val;
void servopulse(int servopin,int myangle)// define a servo pulse
function
{
pulsewidth=(myangle*11)+500;// convert angle to 500-2480 pulse width
digitalWrite(servopin,HIGH);// set the level of servo pin as "high"
delayMicroseconds(pulsewidth);// delay microsecond of pulse width
digitalWrite(servopin,LOW);// set the level of servo pin as "low"
delay(20-pulsewidth/1000);
}
void setup()
{
pinMode(servopin,OUTPUT);// set servo pin as "output"
Serial.begin(9600);// connect to serial port, set baud rate at "9600"
Serial.println("servo=o_serai_simple ready" );
}
void loop()// convert number 0 to 9 to corresponding 0-180 degree
angle, LED blinks corresponding number of time
```

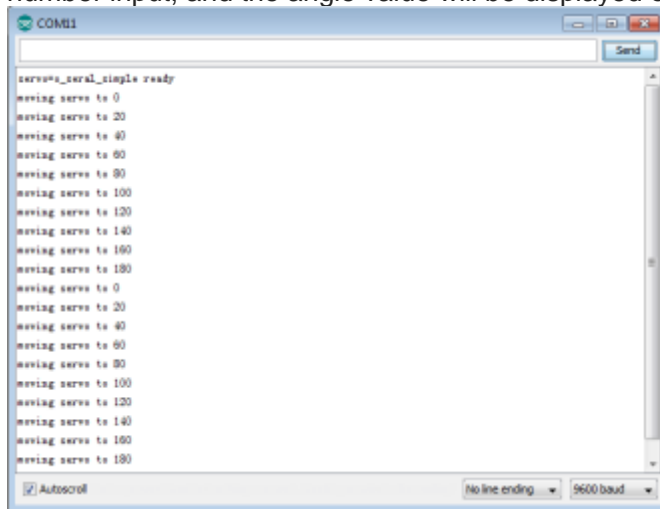
```

{
val=Serial.read();// read serial port value
if(val>'0'&&val<='9')
{
val=val-'0';// convert characteristic quantity to numerical variable
val=val*(180/9);// convert number to angle
Serial.print("moving servo to ");
Serial.print(val,DEC);
Serial.println();
for(int i=0;i<=50;i++) // giving the servo time to rotate to commanded
position
{
servopulse(servopin,val);// use the pulse function
}
}
}
}

```

### Result:

When you input a number on serial monitor, the motor rotates to an angle which is equal to the number input, and the angle value will be displayed on screen, as shown in below figure.



## Project 5: Bluetooth Module



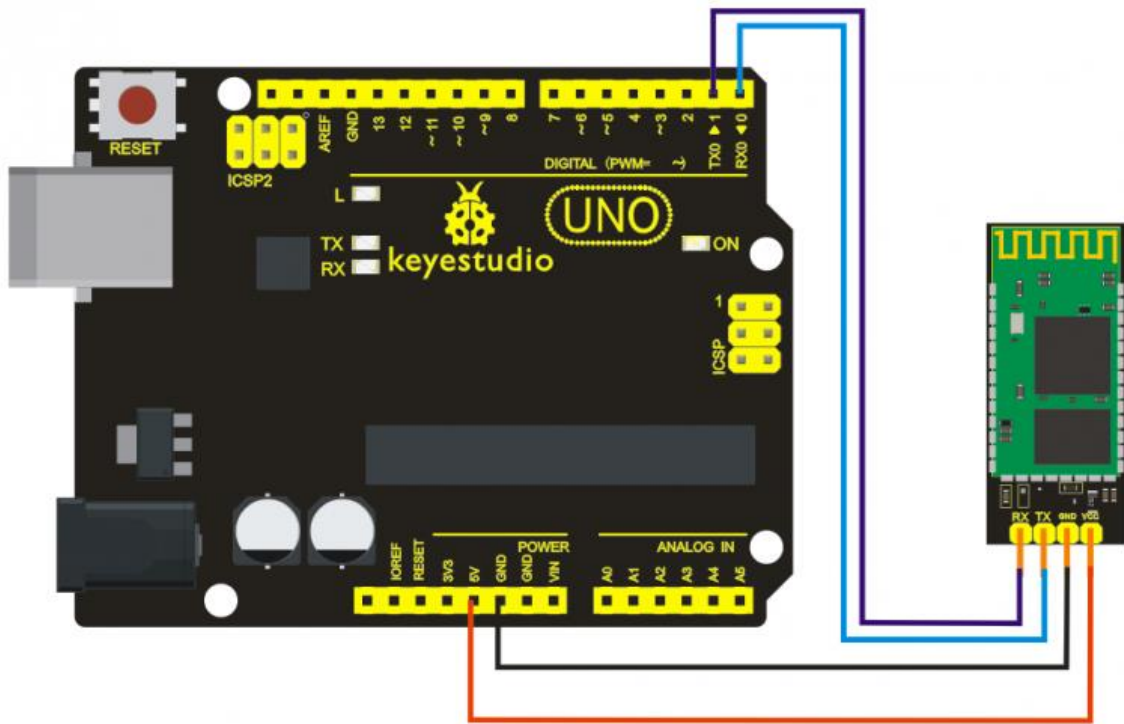
### **Introduction:**

This **Bluetooth module** can easily achieve serial wireless data transmission. Its operating frequency is among the most popular 2.4GHz ISM frequency band (i.e. Industrial, scientific and medical). It adopts Bluetooth 2.1+EDR standard. In Bluetooth 2.1, signal transmit time of different devices stands at a 0.5 seconds interval so that the workload of bluetooth chip can be reduced substantially and more sleeping time can be saved for bluetooth. This module is set with serial interface, which is easy-to-use and simplifying overall design/development cycle.

### **Specification :**

- Bluetooth Protocol: Bluetooth 2.1+ EDR Standard
- USB Protocol: USB v1.1/2.0
- Operating Frequency: 2.4GHz ISM Frequency Band
- Modulation Mode: Gauss Frequency Shift Keying
- Transmit Power:  $\leq 4\text{dBm}$ , Second Stage
- Sensitivity:  $\leq -84\text{dBm}$  at 0.1% Bit Error Rate
- Transmission Speed: 2.1Mbps(Max)/160 kbps(Asynchronous) ; 1Mbps/1Mbps(Synchronous)
- Safety Feature: Authentication and Encryption
- Supported Configuration: Bluetooth Serial Port (major and minor)
- Supply Voltage: 5V DC 50mA
- Operating Temperature: -20 to 55°C

### **Connection Diagram:**



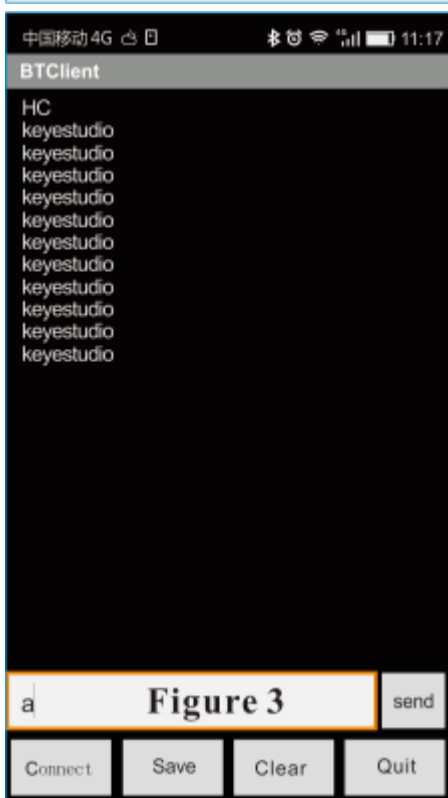
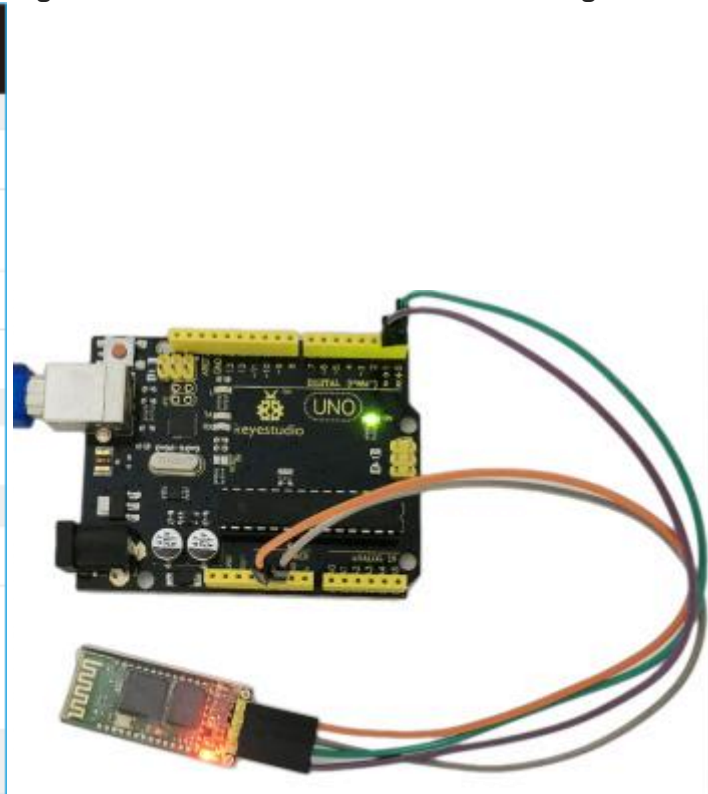
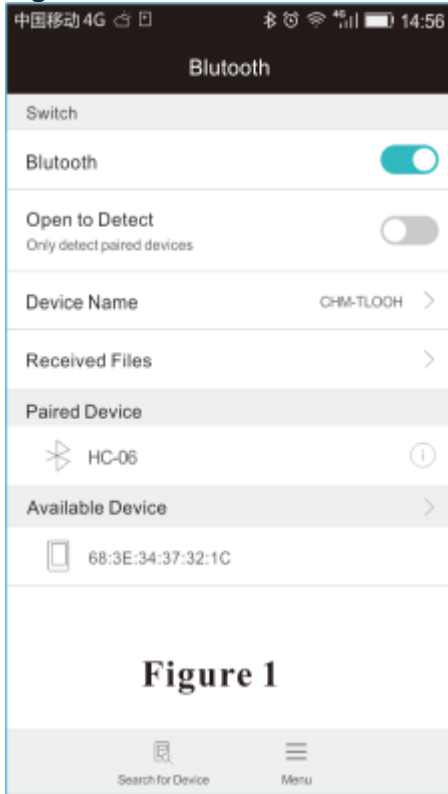
### Sample Code :

```
int val;
int ledpin=13;
void setup()
{
  Serial.begin(9600);
  pinMode(ledpin,OUTPUT);
}
void loop()
{ val=Serial.read();
if(val=='a')
{
digitalWrite(ledpin,HIGH);
delay(250);
digitalWrite(ledpin,LOW);
delay(250);
  Serial.println("keystudio");
}
}
```

### Result:

After power-on, power indicator D1 is on, and LED on Bluetooth module is blinking; open Bluetooth on mobile phone, pair them, input 1234, and finish pairing as shown in Figure 1 ; open APP—Bluetooth serial communication assistant, connect it to Bluetooth, select normal mode, complete connection, and LED on Bluetooth module is on as shown in Figure 2; input an “a” in the assistant, and display “keyestudio” in it as shown in Figure 3.

**Figure 1:**-----**Figure 2:**-----**Figure 3:**





## Project 6: L298N Motor Driver



### Introduction:

Using **L298N** made by ST Company as the control chip, the module has characteristics of strong driving ability, low calorific value and strong anti-interference ability.

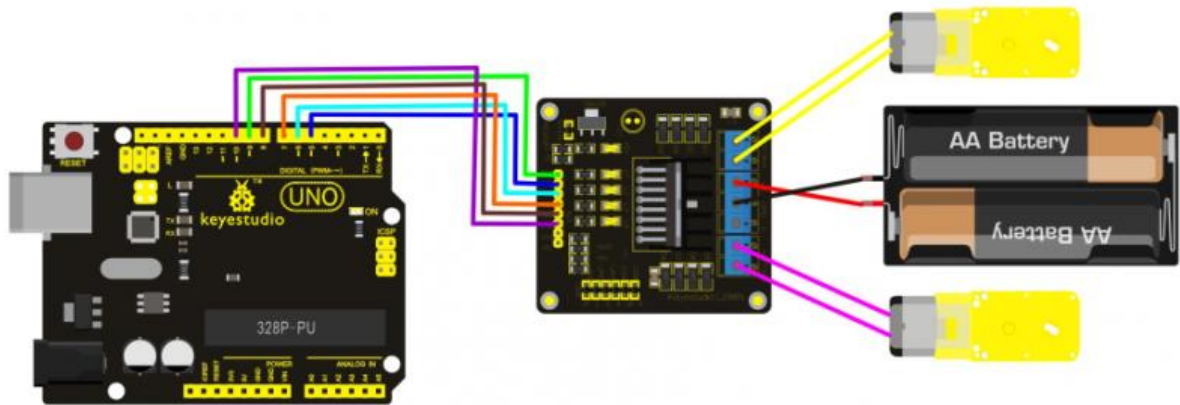
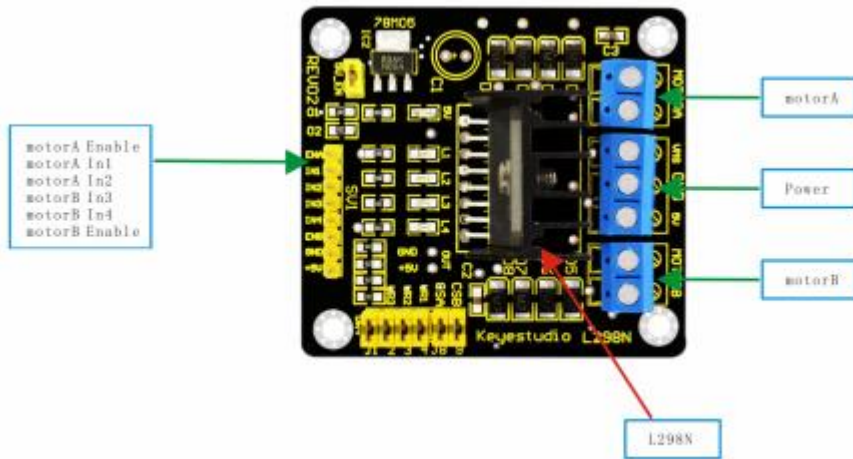
This module can use built-in 78M05 for electric work via a driving power supply part. But to avoid the damage of the voltage stabilizing chip, please use an external 5V logic supply when using more than 12V driving voltage.

Using large capacity filter capacitor, this module can follow current to protect diodes, and improve reliability.

### Specification:

- Working Mode: H bridge (double lines)
- Control Chip: L298N (ST)
- Logical Voltage: 5V
- Driving Voltage: 5V-35V
- Logical Current: 0mA-36mA
- Driving Current: 2A (MAX single bridge)
- Storage Temperature: (-20 °C)-(+135 °C)
- Maximum Power: 25W
- Weight: 30g
- Periphery Dimension: 43 x 43 x 27 mm(L x W x H)

### Circuit Connection:



### Sample Code:

```
int IN1=5;
int IN2=6;
int IN3=7;
int IN4=8;
int ENA=9;
int ENB=10;
void setup()
{
for (int i = 5; i <11; i ++)
{
pinMode(i, OUTPUT);
}
}
void loop()
{
// rotate CW
```

```
digitalWrite(IN1, LOW);  
digitalWrite(IN2, HIGH);  
analogWrite(ENA, 200);  
digitalWrite(IN3, LOW);  
digitalWrite(IN4, HIGH);  
analogWrite(ENB, 200);  
delay(1000);  
// pause for 1S  
analogWrite(ENA, 0);  
analogWrite(ENB, 0);  
delay(1000);  
// rotate CCW  
digitalWrite(IN1, HIGH);  
digitalWrite(IN2, LOW);  
analogWrite(ENA, 100);  
digitalWrite(IN3, HIGH);  
digitalWrite(IN4, LOW);  
analogWrite(ENB, 100);  
delay(1000);  
// pause for 1S  
analogWrite(ENA, 0);  
analogWrite(ENB, 0);  
delay(1000);  
}
```

**Result:**

After connection and power-on, two motors rotate clockwise for 1 second at a speed of 200 (PWM value is 200) and then stop for 1 second; two motors rotate anticlockwise for 1 second at a speed of 100 (PWM value is 100) and then stop for 1 second.

**Project 7: keystudio 1602 I2C Module**



### **Introduction:**

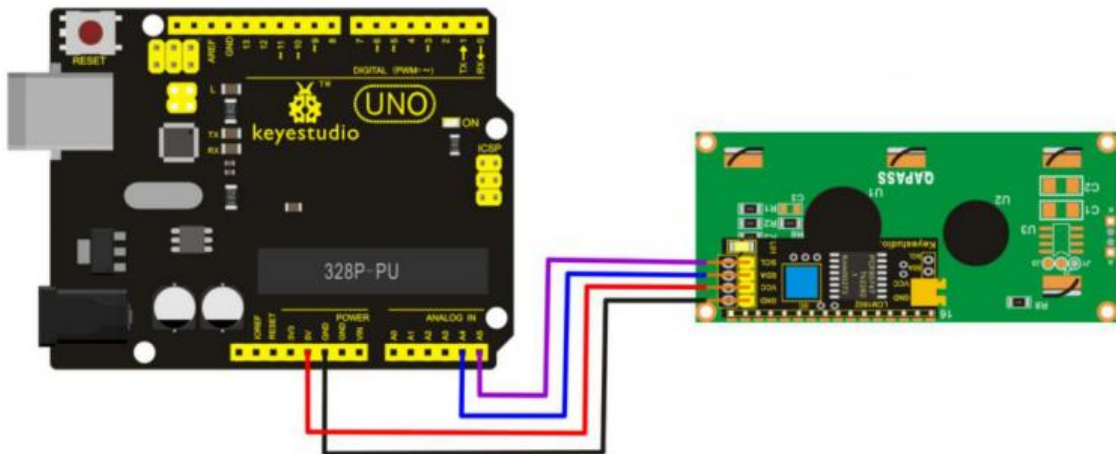
This is great **LCD display** compatible with arduino. With limited pin resources, your project will quickly run out of resources using normal LCDs. With this I2C interface LCD module, you only need 2 lines (I2C) to display the information. If you already have I2C devices in your project, this LCD module actually costs no more resources at all. The address can be set 0x27.

### **Specification :**

- I2C Address: 0x27
- Back Lit (Blue with white char color)
- Supply Voltage: 5V
- Interface: I2C/TWI x1, Gadgeteer interface x2
- Adjustable Contrast
- Size: 82x35x18 mm

### **Connection Diagram :**

I602 is equipped with 4 pins in total. SCL should be connected to analog 5, SDA to analog 4, VCC to +5V and GND to ground.



### Sample Code :

```
//Compatible with the Arduino IDE 1.0
//Library version:1.1
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address to 0x27 for a
16 chars and 2 line display
void setup()
{
  lcd.init(); // initialize the lcd
  lcd.init();
  // Print a message to the LCD.
  lcd.backlight();
  lcd.setCursor(3,0);
  lcd.print("Hello, world!");
  lcd.setCursor(2,1);
  lcd.print("keystudio!");
}
void loop()
{}
```

Get libraries of Wire and LiquidCrystal\_I2C  
from: <https://drive.google.com/open?id=15KOCyLn3Wwo-KICck6hYmm4uiKh0MGrW>

**Result:**

After connection and uploading codes, the 1602 I2C module will display as shown in below figure.



## Project 8: Line Tracking of Smart Car



**Introduction:**

This project introduces a simple and automatic line tracking system of a car based on Arduino microcontroller. This car, regarding UNO as main control, detects black line by IR photoelectric sensor and send the feedback to Arduino. Arduino will analyze the feedback signal and then control the driver motor to adjust the car diversion. Finally the car is able to go around the black line automatically. In addition, you can observe the state of the car through the1602 I2C module.

**Principle:**

- 1.Black absorbs most light. When the panel isn't black, most IR emitted by the sensor is reflected. So the sensor outputs low level at 0.
2. When there is a sensor above black line, since reflectivity of black is not good the IR is reflected back under demand that the sensor works. Therefore, the sensor output 1.
3. We just need to know the output of the sensor is 1 or 0 with Arduino to detect black line.
- 4.Arduino controls the motion of the car according to received signal. The system scheme is showed by the following picture 1-1.



Figure 1-1

5. The system is composed of main control circuit, power supply, IR detecting module, motor and driver module. The structure chart of the system is showed by picture 2-1.

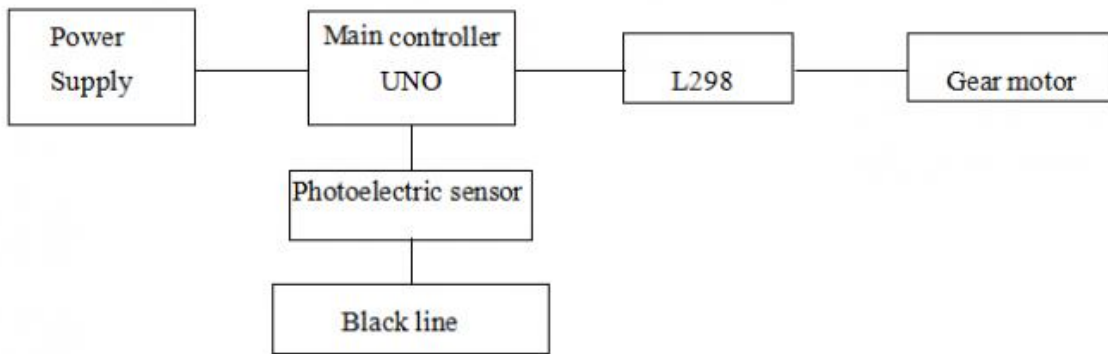
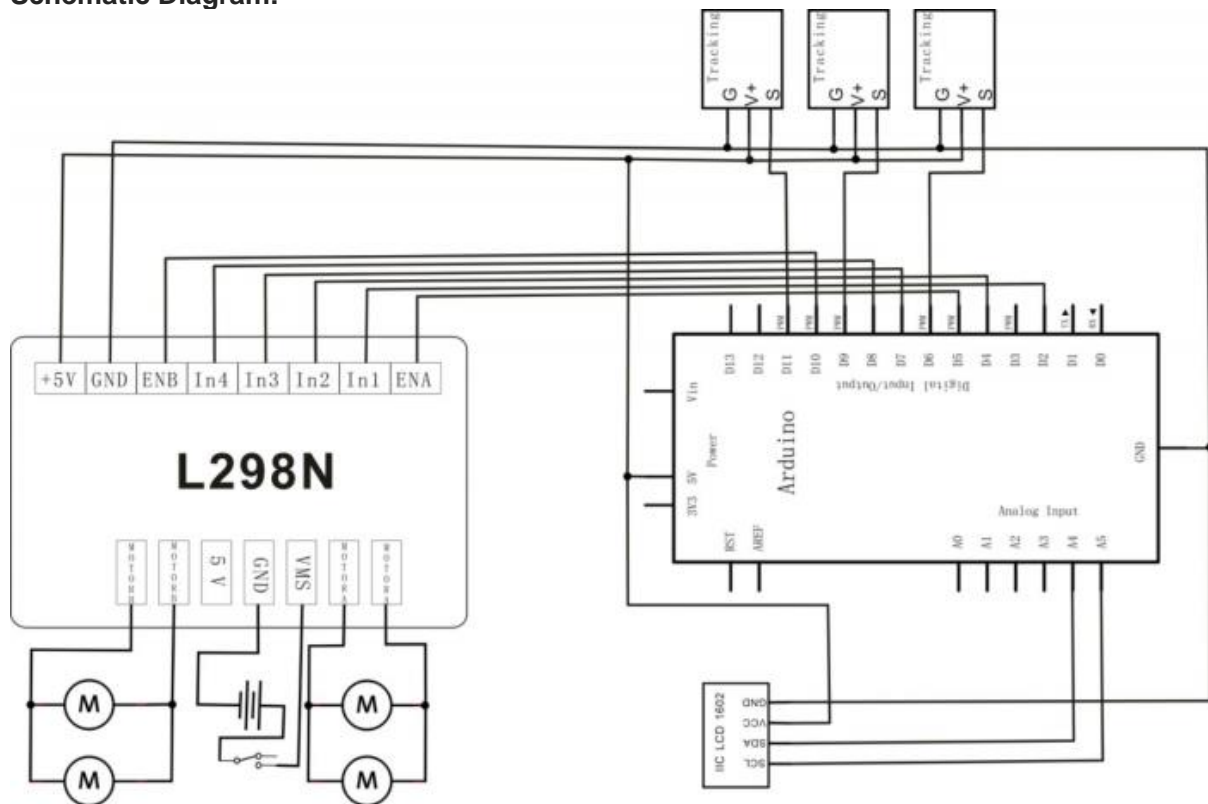


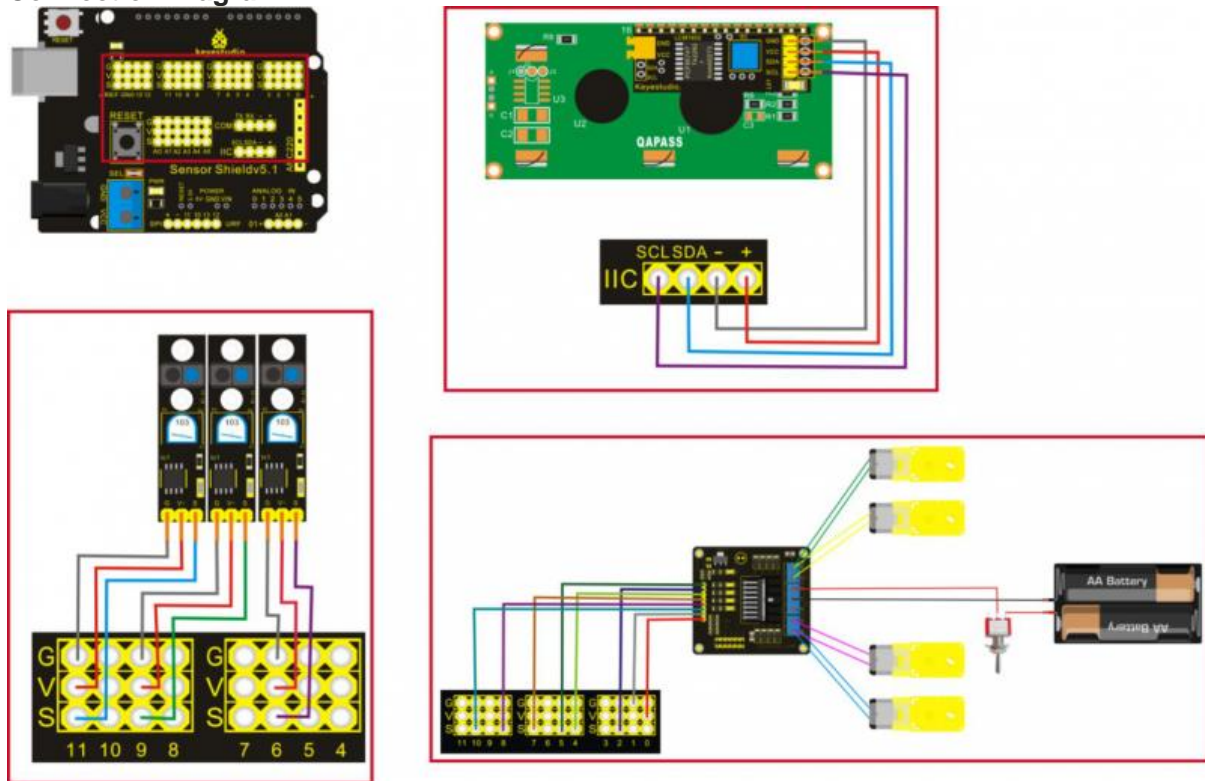
Figure2-1

**Schematic Diagram:**





## Connection Diagram:



## Sample Code:

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#define SensorLeft    6    //input pin of left sensor
#define SensorMiddle  9    //input pin of middle sensor
#define SensorRight   11   //input pin of right sensor
unsigned char SL;        //state of left sensor
unsigned char SM;        //state of middle sensor
unsigned char SR;        //state of right sensor
#define Lpwm_pin     5     //pin of controlling speed---- ENA of motor
driver board
#define Rpwm_pin     10    //pin of controlling speed---- ENA of motor
driver board
int pinLB=2;            //pin of controlling diversion----IN1 of motor
driver board
int pinLF=4;            //pin of controlling diversion----IN2 of motor
driver board
int pinRB=7;            //pin of controlling diversion----IN3 of motor
driver board
int pinRF=8;            //pin of controlling diversion----IN4 of motor
driver board
```

```

unsigned char Lpwm_val =180;//the speed of left wheel at 180 in
initialization
unsigned char Rpwm_val = 180;//the speed of right wheel at 180 in
initialization
int Car_state=0;           //state of car moving
LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address to 0x27 for a
16 chars and 2
void LCD1602_init(void)
{
    lcd.init();
    lcd.backlight();
    lcd.clear();
}
void Sensor_IO_Config()
{
    pinMode(SensorLeft,INPUT);
    pinMode(SensorMiddle,INPUT);
    pinMode(SensorRight,INPUT);
}
void Sensor_Scan(void)
{
    SL = digitalRead(SensorLeft);
    SM = digitalRead(SensorMiddle);
    SR = digitalRead(SensorRight);
}
void M_Control_IO_config(void)//initialized function of IO of motor
driver
{
    pinMode(pinLB,OUTPUT); // pin 2--IN1 of motor driver board
    pinMode(pinLF,OUTPUT); // pin 4--IN2 of motor driver board
    pinMode(pinRB,OUTPUT); // pin 7--IN3 of motor driver board
    pinMode(pinRF,OUTPUT); // pin 8--IN4 of motor driver board
    pinMode(Lpwm_pin,OUTPUT); // pin 5 (PWM) --ENA of motor driver board
    pinMode(Rpwm_pin,OUTPUT); // pin 10 (PWM) --ENB of motor driver board
}
void Set_Speed(unsigned char Left,unsigned char Right)//setting
function of speed
{
    analogWrite(Lpwm_pin,Left);
    analogWrite(Rpwm_pin,Right);
}
void advance()           // going forwards
{
    digitalWrite(pinRB,LOW); // making motor move towards right rear

```

```

    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,LOW); // making motor move towards left rear
    digitalWrite(pinLF,HIGH);
    Car_state = 1;
    show_state();
}
void turnR() //turning on the right(dual wheels)
{
    digitalWrite(pinRB,LOW); //making motor move towards right rear
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,HIGH);
    digitalWrite(pinLF,LOW); //making motor move towards left front
    Car_state = 4;
    show_state();
}
void turnL() //turning on the left(dual wheels)
{
    digitalWrite(pinRB,HIGH);
    digitalWrite(pinRF,LOW ); //making motor move towards right
front
    digitalWrite(pinLB,LOW); //making motor move towards left rear
    digitalWrite(pinLF,HIGH);
    Car_state = 3;
    show_state();
}
void stopp() //stop
{
    digitalWrite(pinRB,HIGH);
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,HIGH);
    digitalWrite(pinLF,HIGH);
    Car_state = 5;
    show_state();
}
void back() //back
{
    digitalWrite(pinRB,HIGH); //making motor move towards right rear
    digitalWrite(pinRF,LOW);
    digitalWrite(pinLB,HIGH); //making motor move towards left rear
    digitalWrite(pinLF,LOW);
    Car_state = 2;
    show_state() ;
}
void show_state(void) //showing current state of the car

```

```

{
  lcd.setCursor(0, 1); //showing from second row
  switch(Car_state)
  {
    case 1:lcd.print(" Go ");Serial.print("\n GO");
    break;
    case 2:lcd.print("Back ");Serial.print("\n Back");
    break;
    case 3:lcd.print("Left ");Serial.print("\n Left");
    break;
    case 4:lcd.print("Right");Serial.print("\n Right");
    break;
    case 5:lcd.print("Stop ");Serial.print("\n Stop");
    break;
    default:
    break;
  }
}
void setup()
{
  LCD1602_init();
  Sensor_IO_Config();
  M_Control_IO_config();          //motor controlling the initialization
of IO
  Set_Speed(Lpwm_val,Rpwm_val); //setting initialization of speed
  lcd.clear();
  lcd.setCursor(0, 0); //cursor set in first row and first column,
  lcd.print(" Wait Signal ");
  stopp();
}
unsigned char old_SL,old_SM,old_SR;
void loop()
{
  Sensor_Scan();
  if (SM == HIGH)// middle sensor in black area
  {
    if (SL == LOW & SR == HIGH) // black on left, white on right, turn left
    {
      turnR();
    }
    else if (SR == LOW & SL == HIGH) // white on left, black on right, turn
right
    {
      turnL();
    }
  }
}

```

```
}
else // white on both sides, going forward
{
advance();
}
}
else // middle sensor on white area
{
if (SL== LOW & SR == HIGH)// black on left, white on right, turn left
{
turnR();
}
else if (SR == LOW & SL == HIGH) // white on left, black on right, turn
right
{
turnL();
}
else // all white, stop
{
back();
delay(100);
stopp() ;
}
}
}
```

## **Project 9: Ultrasonic Obstacle Avoidance of Smart Car**



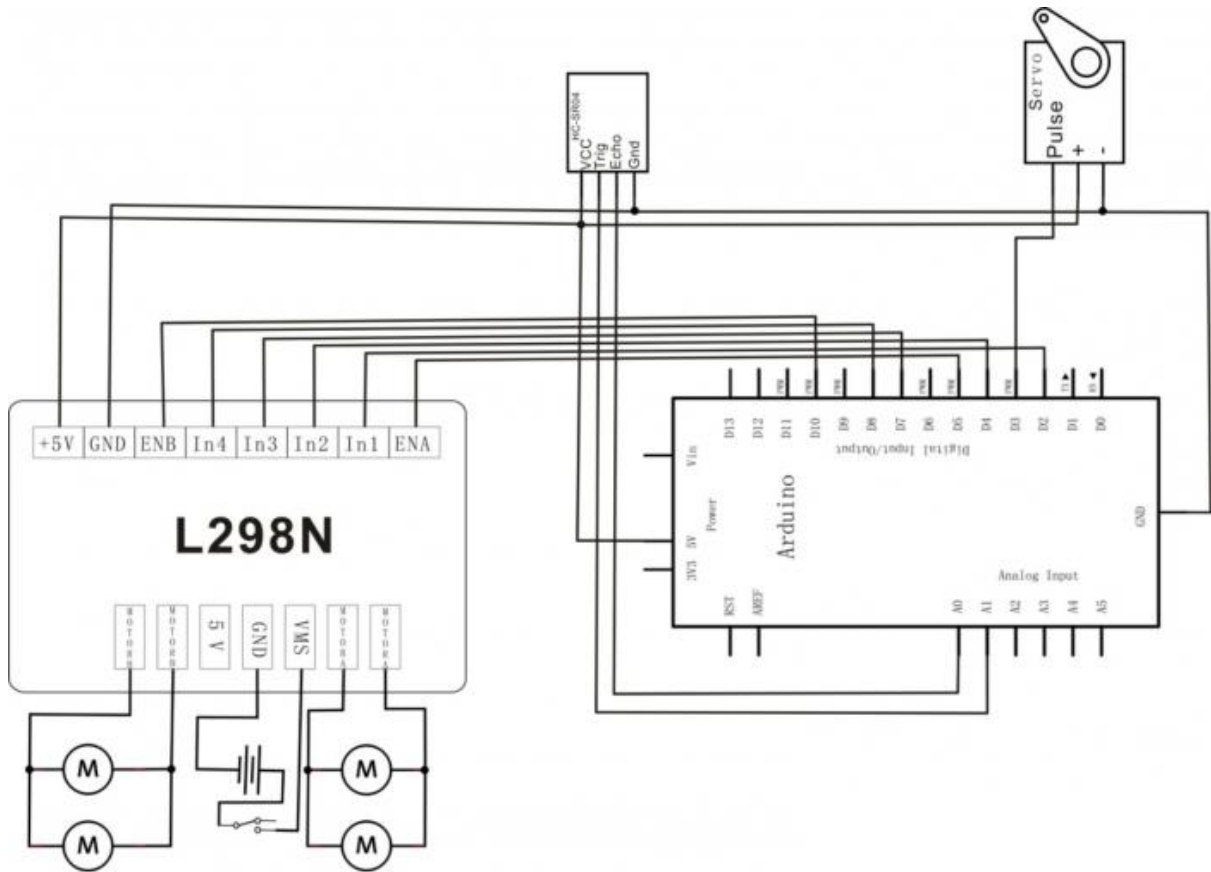
### **Introduction:**

This project is used to detect front obstacle by ultrasonic sensor and platform motor, and send the feedback to Arduino. Arduino will analyse the feedback signal and then control the driver motor to adjust the car diversion. Finally, the car is able to avoid obstacle automatically and to keep going.

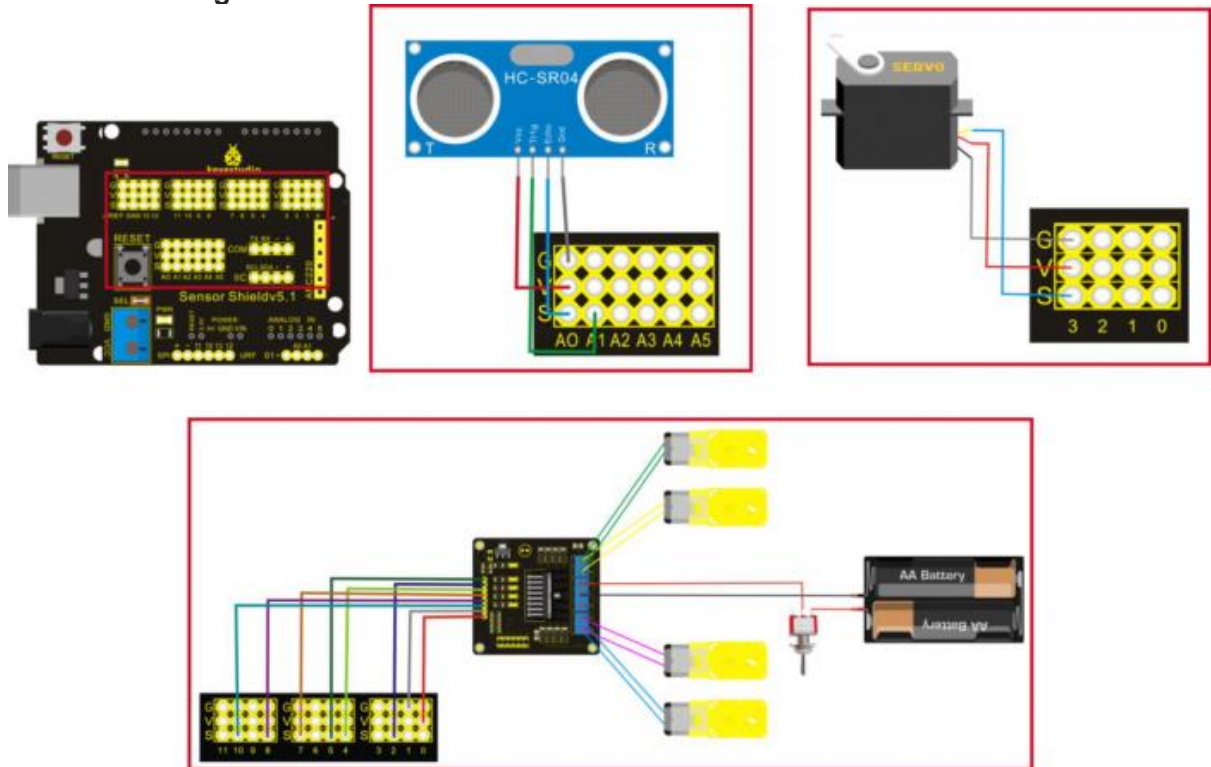
### **Principle:**

1. Ultrasonic detecting distance: one port emits high level more than 10 US. Once outputting level, open potentiometer to time. When the port becomes low level, read out current value. Use the time of detecting distance to calculate distance.
2. Use ultrasonic to detect the distance between obstacle and car, so that control the motion of the car according to the data.
3. If the distance between the car and obstacle is less than 20 cm, the car goes backward; if the distance is no less than 40 cm, the car goes forwards; if the distance is less than 40cm, the motor turns to detect the distance between car and left obstacle or right obstacle; if the distance between car and left obstacle, the distance between car and right obstacle are less than 15 cm, the car goes backward; if the distance between car and left obstacle is larger, the car turns left; if the distance between car and left obstacle is less than or equal to the distance between car and right obstacle, the car turns right.

### **Schematic Diagram:**



Connection Diagram:



Sample Code:

```

#include <Servo.h>
int pinLB = 2;      // defining pin 12
int pinLF = 4;      // defining pin 3
int pinRB = 7;      // defining pin 13
int pinRF = 8;      // defining pin 11
int Lpwm_pin = 5;   //adjusting speed
int Rpwm_pin = 10;  //adjusting speed //
unsigned char Lpwm_val = 200;
unsigned char Rpwm_val = 200;

int inputPin = A0;   // defining receiving pin of ultrasonic signal
int outputPin =A1;   // defining emitting pin of ultrasonic signal

int Fspeedd = 0;     // forward speed
int Rspeedd = 0;     // right speed
int Lspeedd = 0;     // left speed
int directionn = 0;  // front=8 back=2 left=4 right=6
Servo myservo;      // setting myservo
int delay_time = 250; // time for servo motor turning backward

int Fgo = 8;         // going forward
int Rgo = 6;         // turning right
int Lgo = 4;         // turning left
int Bgo = 2;         // turning backward

void setup()
{
  Serial.begin(9600); // defining output pin of motor
  pinMode(pinLB,OUTPUT); // pin 12
  pinMode(pinLF,OUTPUT); // pin 3 (PWM)
  pinMode(pinRB,OUTPUT); // pin 13
  pinMode(pinRF,OUTPUT); // pin 11 (PWM)
  pinMode(inputPin, INPUT); // defining input pin of ultrasonic
  pinMode(outputPin, OUTPUT); // defining output pin of ultrasonic
  myservo.attach(3); // defining output pin9 of motor
}

void advance() // going forward
{
  digitalWrite(pinLB,LOW); // right wheel going forward
  digitalWrite(pinRB, LOW); // left wheel going forward
  digitalWrite(pinLF,HIGH);
  digitalWrite(pinRF,HIGH);
}

```



```

void stopp()          //stop
{
    digitalWrite(pinRB,HIGH);
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,HIGH);
    digitalWrite(pinLF,HIGH);
}
void right()         //turning right(single wheel)
{
    digitalWrite(pinRB,LOW); //making motor move towards right rear
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,HIGH);
    digitalWrite(pinLF,LOW); //making motor move towards left front
}
void left()          //turning left(single wheel)
{
    digitalWrite(pinRB,HIGH);
    digitalWrite(pinRF,LOW ); //making motor move towards right
front
    digitalWrite(pinLB,LOW); //making motor move towards left rear
    digitalWrite(pinLF,HIGH);
}

void back()          //going backward
{
    digitalWrite(pinRB,HIGH); //making motor move towards right rear
    digitalWrite(pinRF,LOW);
    digitalWrite(pinLB,HIGH); //making motor move towards left rear
    digitalWrite(pinLF,LOW);
}
void detection()     //measuring 3 angles(0.90.179)
{
    int delay_time = 250; // time for servo motor turning backward
    ask_pin_F(); // reading out the front distance
    if(Fspeedd < 20) // assuming the front distance less than
10cm
    {
        stopp(); // clear output material
        delay(100);
        back(); // going backward for 0.2 second
        delay(200);
    }
}

```

```

        if(Fspeedd < 40)          // assuming the front distance less than
25cm
    {
        stopp();
        delay(100);              // clear output material
        ask_pin_L();             // reading out the left distance
        delay(delay_time);      // waiting servo motor to be stable
        ask_pin_R();             // reading out the right distance
        delay(delay_time);      // waiting servo motor to be stable

        if(Lspeedd > Rspeedd)    //assuming left distance more than
right distance
    {
        directionn = Lgo;        //turning left
    }

        if(Lspeedd <= Rspeedd)   //assuming left distance less than or
equal to right distance
    {
        directionn = Rgo;        //turning right
    }

        if (Lspeedd < 15 && Rspeedd < 15) //assuming both left
distance and right distance less than 10cm
    {
        directionn = Bgo;        //going backward
    }
    }
    else                          //assuming the front distance more
than 25 cm
    {
        directionn = Fgo;        //going forward
    }
}

void ask_pin_F() // measuring the front distance
{
    myservo.write(90);
    digitalWrite(outputPin, LOW); // ultrasonic launching low
voltage at 2µs
    delayMicroseconds(2);
    digitalWrite(outputPin, HIGH); // ultrasonic launching high
voltage at 10µs, at least at10µs
    delayMicroseconds(10);
}

```

```

        digitalWrite(outputPin, LOW);    // keeping ultrasonic launching
low voltage
        float Fdistance = pulseIn(inputPin, HIGH); // time of error
reading
        Fdistance= Fdistance/5.8/10;      // converting time into
distance (unit : cm)
        Fspeedd = Fdistance;             // reading-in Fspeedd(fore
speed) with distance
    }
void ask_pin_L() // measuring left distance
{
    myservo.write(5);
    delay(delay_time);
    digitalWrite(outputPin, LOW); // ultrasonic launching low
voltage at 2µs
    delayMicroseconds(2);
    digitalWrite(outputPin, HIGH); // ultrasonic launching high
voltage at 10µs, at least at10µs
    delayMicroseconds(10);
    digitalWrite(outputPin, LOW); // keeping ultrasonic launching
low voltage
    float Ldistance = pulseIn(inputPin, HIGH); // time of error
reading
    Ldistance= Ldistance/5.8/10;      // converting time into
distance (unit : cm)
    Lspeedd = Ldistance;             //reading-in Lspeedd(left
speed) with distance
}
void ask_pin_R() // measuring right distance
{
    myservo.write(177);
    delay(delay_time);
    digitalWrite(outputPin, LOW); // ultrasonic launching low
voltage at 2µs
    delayMicroseconds(2);
    digitalWrite(outputPin, HIGH); // ultrasonic launching high
voltage at 10µs, at least at10µs
    delayMicroseconds(10);
    digitalWrite(outputPin, LOW); // keeping ultrasonic launching
low voltage
    float Rdistance = pulseIn(inputPin, HIGH); // time of error
reading
    Rdistance= Rdistance/5.8/10;      // onverting time into
distance (unit : cm)

```

```

        Rspeedd = Rdistance;           // reading-in Rspeedd(right
speed) with distance
    }

void loop()
{
    myservo.write(90); //making motor regression, being ready for next
measurement
    detection();      //measuring angle and deciding which direction
it moves towards
    if(directionn == 2) //supposing direction = 2(back up)
    {
        back();
        delay(800);           // back up
        left() ;
        delay(200);           //moving slightly towards left(avoiding
locked)
    }
    if(directionn == 6)       //supposing direction = 6(turning
right)
    {
        back();
        delay(100);
        right();
        delay(600);           // turning right
    }
    if(directionn == 4)       //supposing direction = 4(turning left)
    {
        back();
        delay(600);
        left();
        delay(600);           // turning left
    }
    if(directionn == 8)       //supposing direction = = 8(going
forwards)
    {
        advance();           // going forwards normally
        delay(100);
    }
}
}

```

## Project 10: IR Remote Control of Smart Car

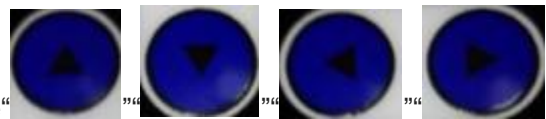


**Introduction:**

This project,uses an IR module to receive IR remote signal and send the signal to Arduino. Arduino will analyze the signal and then control the driver motor and the motion of the car with IR remote control. In addition, you can observe the state of the car through the 1602 I2C Module.

**Principle:**

1.Connecting Arduino to IR receiving module, Bluetooth module and IR receiving module communicating with IR remote control.



2.IR remote control will send these button message“



”to IR receiving module.

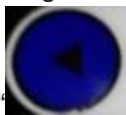
3.IR receiving module will send signal to Arduino, and it will control the motion of the car.



4. When Arduino receiving this message“



”, the car goes backward.



When receiving“



”,the car turns right.

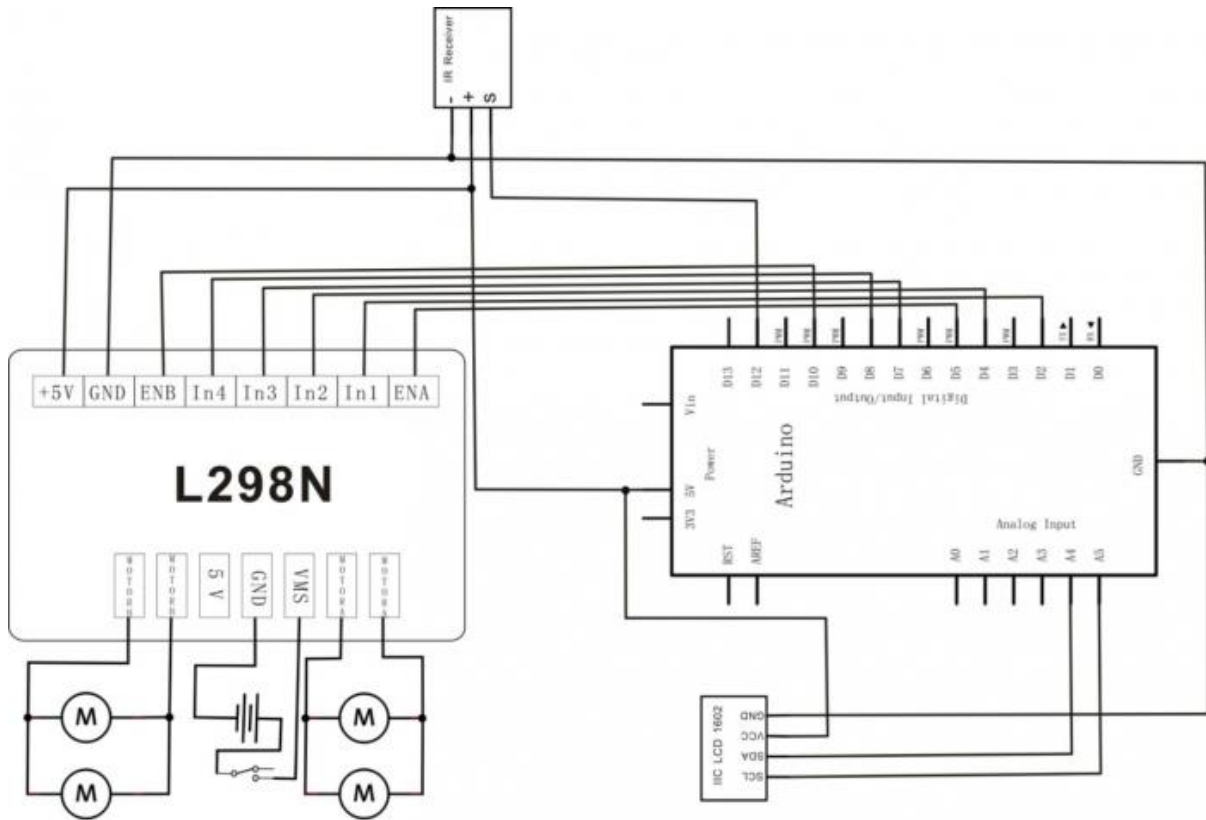


When receiving“

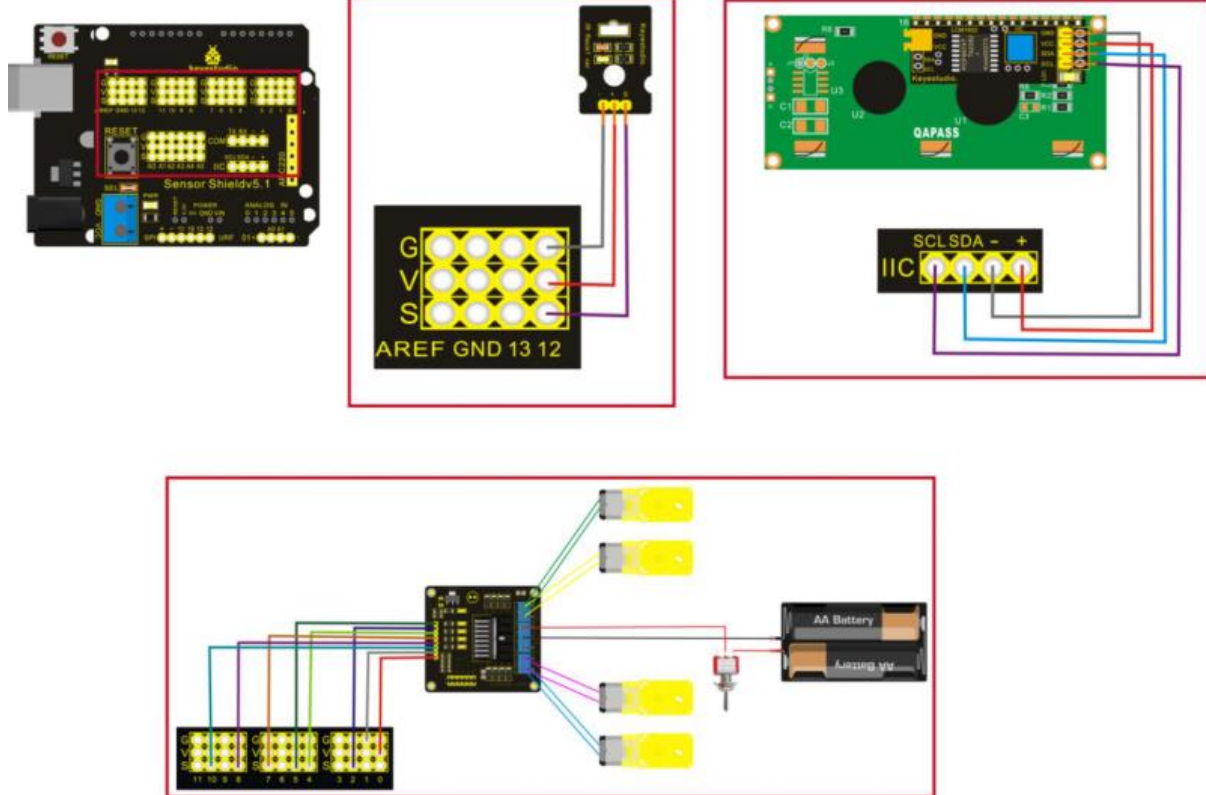


”, the car quits.

**Schematic Diagram:**



**Connection Diagram:**



**Sample Code:**

```

#include <LiquidCrystal_I2C.h>    //including libraries of I2C-LCD1602
liquid crystal
#include <Wire.h>                //including libraries of I2C
#include <IRremote.h>
int RECV_PIN = 12;
IRrecv irrecv(RECV_PIN);
decode_results results;
#define IR_Go      0x00ff629d
#define IR_Back   0x00ffa857
#define IR_Left   0x00ff22dd
#define IR_Right  0x00ffc23d
#define IR_Stop   0x00ff02fd
#define IR_ESC    0x00ff52ad
LiquidCrystal_I2C lcd(0x27,16,2); //defining liquid crystal
#define Lpwm_pin  5      //adjusting speed
#define Rpwm_pin  10     //adjusting speed //
int pinLB=2;        // defining pin2 left rear
int pinLF=4;        // defining pin4 left front
int pinRB=7;        // defining pin7 right rear
int pinRF=8;        // defining pin8 right front
unsigned char Lpwm_val = 200;
unsigned char Rpwm_val = 200;
int Car_state=0;
void M_Control_IO_config(void)
{
    pinMode(pinLB,OUTPUT); // pin2
    pinMode(pinLF,OUTPUT); // pin4
    pinMode(pinRB,OUTPUT); // pin7
    pinMode(pinRF,OUTPUT); // pin8
    pinMode(Lpwm_pin,OUTPUT); // pin11 (PWM)
    pinMode(Rpwm_pin,OUTPUT); // pin10 (PWM)
}
void Set_Speed(unsigned char Left,unsigned char Right)
{
    analogWrite(Lpwm_pin,Left);
    analogWrite(Rpwm_pin,Right);
}
void advance()      // going forward
{
    digitalWrite(pinRB,LOW); // making motor move towards right rear
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,LOW); // making motor move towards left rear
    digitalWrite(pinLF,HIGH);
}

```

```

    Car_state = 1;
    show_state();
}
void turnR()          //turning right(dual wheel)
{
    digitalWrite(pinRB,LOW); //making motor move towards right rear
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,HIGH);
    digitalWrite(pinLF,LOW); //making motor move towards left front
    Car_state = 4;
    show_state();
}
void turnL()          //turning left(dual wheel)
{
    digitalWrite(pinRB,HIGH);
    digitalWrite(pinRF,LOW ); //making motor move towards right
front
    digitalWrite(pinLB,LOW); //making motor move towards left rear
    digitalWrite(pinLF,HIGH);
    Car_state = 3;
    show_state();
}
void stopp()          //stop
{
    digitalWrite(pinRB,HIGH);
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,HIGH);
    digitalWrite(pinLF,HIGH);
    Car_state = 5;
    show_state();
}
void back()           //back up
{
    digitalWrite(pinRB,HIGH); //making motor move towards right rear
    digitalWrite(pinRF,LOW);
    digitalWrite(pinLB,HIGH); //making motor move towards left rear
    digitalWrite(pinLF,LOW);
    Car_state = 2;
    show_state() ;
}
void show_state(void)
{
    lcd.setCursor(0, 1);
    switch(Car_state)

```



```

    {
        case 1:lcd.print(" Go ");Serial.print("\n GO");
        break;
        case 2:lcd.print("Back ");Serial.print("\n Back");
        break;
        case 3:lcd.print("Left ");Serial.print("\n Left");
        break;
        case 4:lcd.print("Right");Serial.print("\n Right");
        break;
        case 5:lcd.print("Stop ");Serial.print("\n Stop");
        break;
        default:
        break;
    }
}
void LCD1602_init(void) //function of initialization of
liquid crystal
{
    lcd.init(); //invoking initialized function in LiquidCrystal_I2C.h
    delay(10); //delaying for10 millisecond
    lcd.backlight(); //open backlight of LCD1602
    lcd.clear(); //clear screen
}
void IR_Control(void)
{
    unsigned long Key;
    lcd.setCursor(0,0); //setting cursor in the first row and column
    lcd.print("IR_Ctr ");
    while(Key!=IR_ESC)
    {
        if(irrecv.decode(&results)) //judging if serial port receives data
        {
            Key = results.value;
            switch(Key)
            {
                case IR_Go:advance(); //UP
                break;
                case IR_Back: back(); //back
                break;
                case IR_Left:turnL(); //Left
                break;
                case IR_Right:turnR(); //Righ
                break;
                case IR_Stop:stopp(); //stop
            }
        }
    }
}

```

```

        break;
        default:
        break;
    }
    irrecv.resume(); // Receive the next value
}
}
lcd.clear();
lcd.setCursor(0, 0); //setting cursor in the first row and column,
lcd.print(" Wait Signal ");
stopp();
}
void setup()
{
    LCD1602_init();
    M_Control_IO_config();
    Set_Speed(Lpwm_val,Rpwm_val);
    irrecv.enableIRIn(); // Start the receiver
    Serial.begin(9600); //initializing serial port, Bluetooth used as
serial port, setting baud ratio at 9600
    lcd.setCursor(0, 0); //setting cursor at the first row and column
    lcd.print(" Wait Signal ");
    stopp();
}
void loop()
{
    if (irrecv.decode(&results)) {
        if(results.value == IR_Stop )IR_Control();
        irrecv.resume(); // Receive the next value
    }
}
}

```

## Project 11: Distance Detecting of Smart Car



### Introduction:

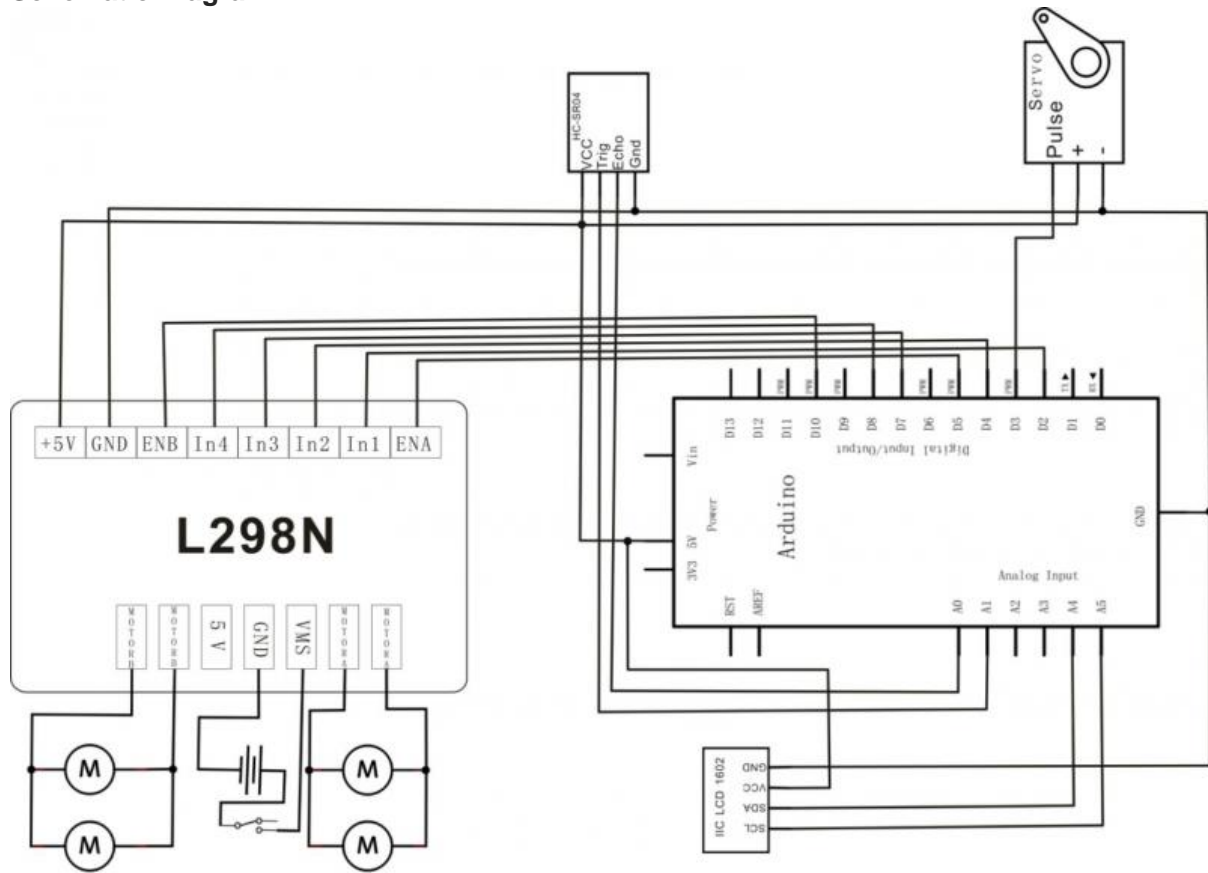
This project detects front obstacle by ultrasonic sensor and platform motor, and then send the feedback to Arduino. Arduino will analyse the feedback signal and then control the driver motor to adjust the car diversion. Finally, the car is able to avoid obstacle automatically and to keep going.

In addition, you can observe the state and speed of the car, the angle of motor and the distance between car and obstacle through the 1602 I2C Module.

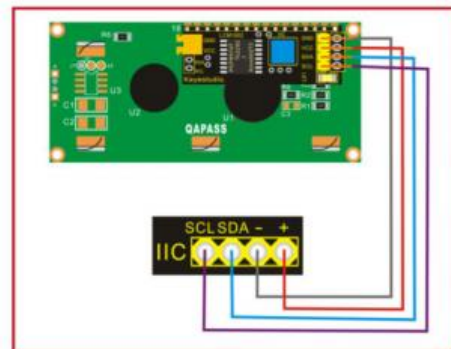
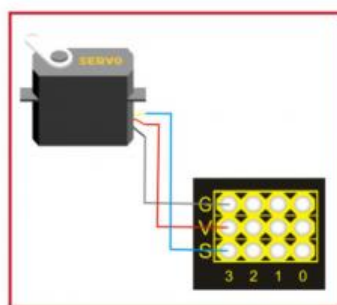
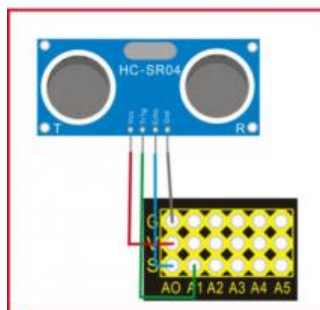
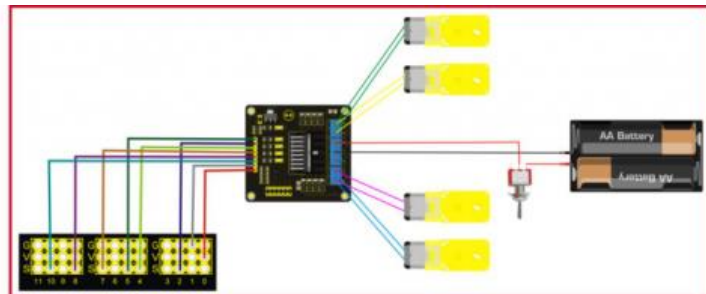
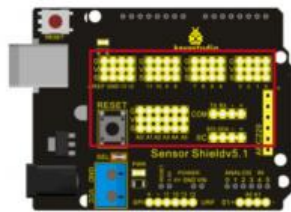
### Principle:

1. Ultrasonic detecting distance: one port emits high level more than 10 US. Once it is outputting level, open potentiometer to time. When the port becomes low level, read out current value. Use the time of detecting distance to calculate distance.
2. Use ultrasonic to detect the distance between obstacle and car, so that control the motion of the car according to the data.
3. If the distance between the car and obstacle is less than 35 cm, the car goes backward; if the distance is no less than 10 cm, the car goes forwards; if the distance is less than 60 cm, the motor turns to detect the distance between car and left obstacle or right obstacle; if the distance between car and left obstacle, the distance between car and right obstacle are less than 35 cm, the car goes backward; if the distance between car and left obstacle is larger, the car turns left; if the distance between car and left obstacle is less than or equal to the distance between car and right obstacle, the car turns right.

**Schematic Diagram :**



**Connection Diagram:**



**Sample Code :**

```

#include <LiquidCrystal_I2C.h>    //including libraries of I2C-LCD1602
liquid crystal
#include <Wire.h>                //including libraries of I2C
////////////////////////////////////
int inputPin=A0; // ultrasonic module  ECHO to A0
int outputPin=A1; // ultrasonic module  TRIG to A1
LiquidCrystal_I2C lcd(0x27,16,2); //defining liquid crystal
#define Lpwm_pin  5    //pin of controlling speed---- ENA of motor
driver board
#define Rpwm_pin  10   //pin of controlling speed---- ENB of motor
driver board
int pinLB=2;          //pin of controlling turning---- IN1 of motor
driver board
int pinLF=4;          //pin of controlling turning---- IN2 of motor
driver board
int pinRB=7;          //pin of controlling turning---- IN3 of motor
driver board
int pinRF=8;          //pin of controlling turning---- IN4 of motor
driver board
unsigned char Lpwm_val = 250; //initialized left wheel speed at 250
unsigned char Rpwm_val = 250; //initialized right wheel speed at 250
int Car_state=0;        //the working state of car
int servopin=3;         //defining digital port pin 3, connecting
to signal line of servo motor
int myangle;           //defining variable of angle
int pulsewidth;        //defining variable of pulse width
unsigned char DuoJiao=60; //initialized angle of motor at 60°

void servopulse(int servopin,int myangle) //defining a function of
pulse
{
pulsewidth=(myangle*11)+500; //converting angle into pulse width value
at 500-2480
digitalWrite(servopin,HIGH); //increasing the level of motor interface
to upmost
delayMicroseconds(pulsewidth); //delaying microsecond of pulse width
value
digitalWrite(servopin,LOW); //decreasing the level of motor interface
to the least
delay(20-pulsewidth/1000);
}
void Set_servopulse(int set_val)
{

```

```

    for(int i=0;i<=10;i++) //giving motor enough time to turn to
    assigning point
        servopulse(servopin,set_val); //invoking pulse function
    }
void M_Control_IO_config(void)
{
    pinMode(pinLB,OUTPUT); // /pin 2
    pinMode(pinLF,OUTPUT); // pin 4
    pinMode(pinRB,OUTPUT); // pin 7
    pinMode(pinRF,OUTPUT); // pin 8
    pinMode(Lpwm_pin,OUTPUT); // pin 11 (PWM)
    pinMode(Rpwm_pin,OUTPUT); // pin10 (PWM)
}
void Set_Speed(unsigned char Left,unsigned char Right) //function of
setting speed
{
    analogWrite(Lpwm_pin,Left);
    analogWrite(Rpwm_pin,Right);
}
void advance() // going forward
{
    digitalWrite(pinRB,LOW); // making motor move towards right rear
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,LOW); // making motor move towards left rear
    digitalWrite(pinLF,HIGH);
    Car_state = 1;
    show_state();
}
void turnR() //turning right(dual wheel)
{
    digitalWrite(pinRB,LOW); //making motor move towards right rear
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,HIGH);
    digitalWrite(pinLF,LOW); //making motor move towards left front
    Car_state = 4;
    show_state();
}
void turnL() //turning left(dual wheel)
{
    digitalWrite(pinRB,HIGH);
    digitalWrite(pinRF,LOW ); //making motor move towards right
front
    digitalWrite(pinLB,LOW); //making motor move towards left rear
    digitalWrite(pinLF,HIGH);

```

```

    Car_state = 3;
    show_state();
}
void stopp()          //stop
{
    digitalWrite(pinRB,HIGH);
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,HIGH);
    digitalWrite(pinLF,HIGH);
    Car_state = 5;
    show_state();
}
void back()          //back up
{
    digitalWrite(pinRB,HIGH); //making motor move towards right rear
    digitalWrite(pinRF,LOW);
    digitalWrite(pinLB,HIGH); //making motor move towards left rear
    digitalWrite(pinLF,LOW);
    Car_state = 2;
    show_state() ;
}
void show_state(void)
{
    lcd.setCursor(0, 1);
    switch(Car_state)
    {
        case 1:lcd.print(" Go ");Serial.print(" \r\n GO");
        break;
        case 2:lcd.print("Back ");Serial.print(" \r\n Back");
        break;
        case 3:lcd.print("Left ");Serial.print(" \r\n Left");
        break;
        case 4:lcd.print("Right");Serial.print(" \r\n Right");
        break;
        case 5:lcd.print("Stop ");Serial.print(" \r\n Stop");
        break;
        default:
        break;
    }
}
void LCD1602_init(void)          //including initialized function of
liquid crystal
{

```

```

    lcd.init(); //invoking initialized function of LCD in
LiquidCrystal_I2C.h
    delay(10); //delaying for 10 millisecond
    lcd.backlight(); //open backlight of LCD1602
    lcd.clear(); //clear screen
}
void Show_V(unsigned char V)
{
    lcd.setCursor(11, 0);
    lcd.print("V=   ");
    lcd.setCursor(13, 0);
    lcd.print(V,DEC);
    Serial.print("\n Speed = ");
    Serial.print(V,DEC);
}
void Show_DuoJiao(unsigned char Jiao)
{
    lcd.setCursor(6,1);
    lcd.print("C=   ");
    lcd.setCursor(8, 1);
    lcd.print(Jiao,DEC);
    Serial.print("\n JiaoDu = ");
    Serial.print(Jiao,DEC);
}
void Self_Control(void)//self-going, ultrasonic obstacle avoidance
{
    int H;
    lcd.setCursor(0, 0); //setting cursor in the first row and column
    lcd.print("Self_Ctr           ");
    Show_V(Lpwm_val);
    Set_servopulse(DuoJiao);
    Show_DuoJiao(DuoJiao);
    H = Ultrasonic_Ranging(1);
    delay(300);
    if(Ultrasonic_Ranging(1) < 35)
    {
        stopp();
        delay(100);
        back();
        delay(50);
    }

    if(Ultrasonic_Ranging(1) < 60)
    {

```



```

stopp();
delay(100);
Set_servopulse(5);
Show_DuoJiao(5);
int L = ask_pin_L(2);
delay(300);
  Set_servopulse(177);
  Show_DuoJiao(177);
int R = ask_pin_R(3);
delay(300);

if(ask_pin_L(2) > ask_pin_R(3))
{
  back();
  delay(100);
  turnL();
  delay(400);
  stopp();
  delay(50);
  Set_servopulse(DuoJiao);
  Show_DuoJiao(DuoJiao);
  H = Ultrasonic_Ranging(1);
  delay(500);
}

if(ask_pin_L(2) <= ask_pin_R(3))
{
  back();
  delay(100);
  turnR();
  delay(400);
  stopp();
  delay(50);
  Set_servopulse(DuoJiao);
  Show_DuoJiao(DuoJiao);
  H = Ultrasonic_Ranging(1);
  delay(300);
}
  if (ask_pin_L(2) < 35 && ask_pin_R(3) < 35)
  {
  stopp();
  delay(50);
  back();
  delay(50);

```

```

    }
}
else
{
advance();
}
}
int Ultrasonic_Ranging(unsigned char Mode)//function of ultrasonic
distance detecting , MODE=1, displaying, no displaying under other
situation

{
int old_distance;
digitalWrite(outputPin, LOW);
delayMicroseconds(2);
digitalWrite(outputPin, HIGH);
delayMicroseconds(10);
digitalWrite(outputPin, LOW);
int distance = pulseIn(inputPin, HIGH); // reading the duration of
high level
distance= distance/58; // Transform pulse time to distance
if(Mode==1){
lcd.setCursor(11, 1);
lcd.print("H= ");
lcd.setCursor(13, 1);
lcd.print(distance,DEC);
Serial.print("\n H = ");
Serial.print(distance,DEC);
return distance;
}
else return distance;
}
int ask_pin_L(unsigned char Mode)
{
int old_Ldistance;
digitalWrite(outputPin, LOW);
delayMicroseconds(2);
digitalWrite(outputPin, HIGH);
delayMicroseconds(10);
digitalWrite(outputPin, LOW);
int Ldistance = pulseIn(inputPin, HIGH);
Ldistance= Ldistance/58; // Transform pulse time to distance
if(Mode==2){
lcd.setCursor(11, 1);

```

```

        lcd.print("L=   ");
        lcd.setCursor(13, 1);
        lcd.print(Ldistance,DEC);
        Serial.print("\n L = ");
        Serial.print(Ldistance,DEC);
        return Ldistance;
    }
    else return Ldistance;
}
int ask_pin_R(unsigned char Mode)
{
    int old_Rdistance;
    digitalWrite(outputPin, LOW);
    delayMicroseconds(2);
    digitalWrite(outputPin, HIGH); //
    delayMicroseconds(10);
    digitalWrite(outputPin, LOW);
    int Rdistance = pulseIn(inputPin, HIGH);
    Rdistance= Rdistance/58;    // Transform pulse time to distance
    if(Mode==3){
        lcd.setCursor(11, 1);
        lcd.print("R=   ");
        lcd.setCursor(13, 1);
        lcd.print(Rdistance,DEC);
        Serial.print("\n R = ");
        Serial.print(Rdistance,DEC);
        return Rdistance;
    }
    else return Rdistance;
}

void setup()
{
    pinMode(servopin,OUTPUT); //setting motor interface as output
    LCD1602_init();           //initializing 1602
    M_Control_IO_config();    //motor controlling the initialization of
    IO
    Set_Speed(Lpwm_val,Rpwm_val); //setting initialized speed
    Set_servopulse(DuoJiao);      //setting initialized motor angle
    pinMode(inputPin, INPUT);     //starting receiving IR remote
    control signal
    pinMode(outputPin, OUTPUT);   //IO of ultrasonic module
    Serial.begin(9600);           //initialized serial port , using
    Bluetooth as serial port, setting baud

```

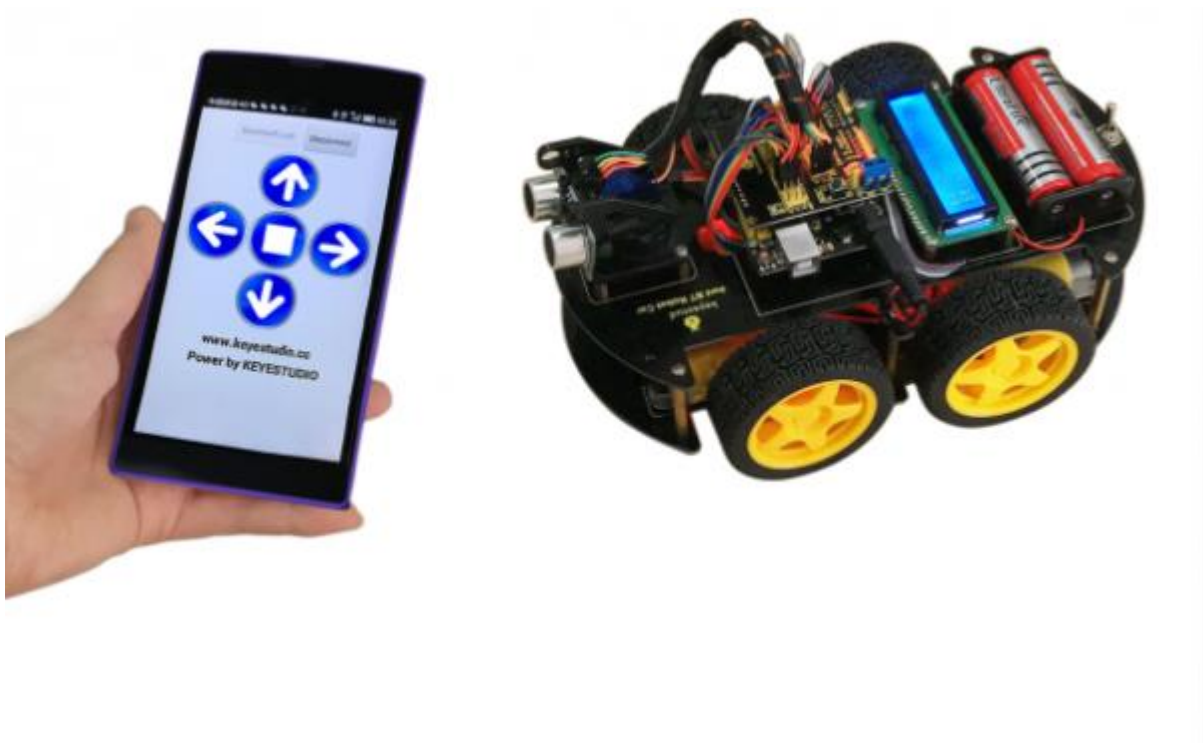
```

    lcd.setCursor(0, 0);          //setting cursor at 0.0
    lcd.print(" Wait Signal "); //LCD printing character string
    stopp();                      //stop
}
void loop()
{

    Self_Control();
}

```

## Project 12: Bluetooth Remote Control of Smart Car



### Introduction :

This project, regarding Arduino as main control, uses Bluetooth module to receive signal from mobile phone, and then send the signal to Arduino. Arduino will analyse the signal and then control the driver motor to control the motion of the car. In addition, you can observe the state and speed of the car, the angle of motor, and the distance between car and obstacle through the 1602 I2C Module.

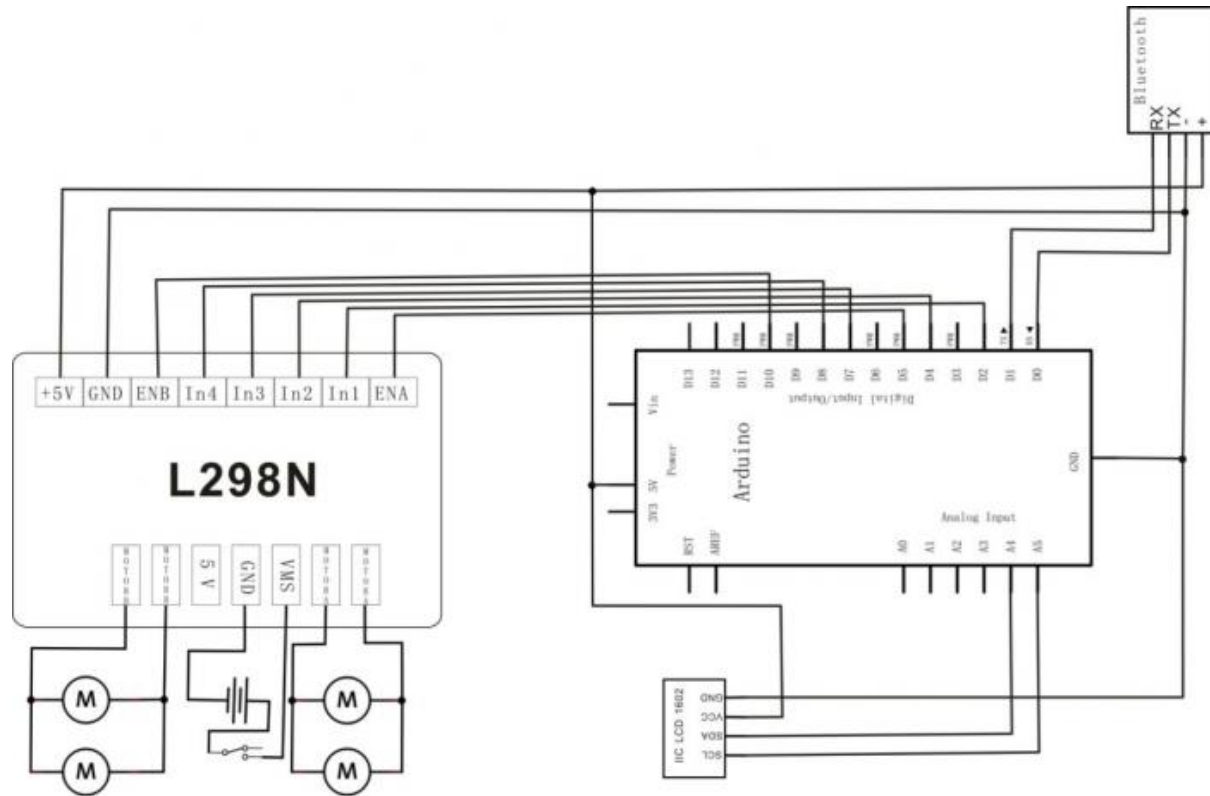
### Principle :

1. Connecting Arduino to Bluetooth module, this module communicates with APP on the mobile phone
2. APP sending "U" "D" "L" "R" "S" to Bluetooth module
3. Bluetooth module will send the received message to Arduino, and Arduino controls the car correspondingly.

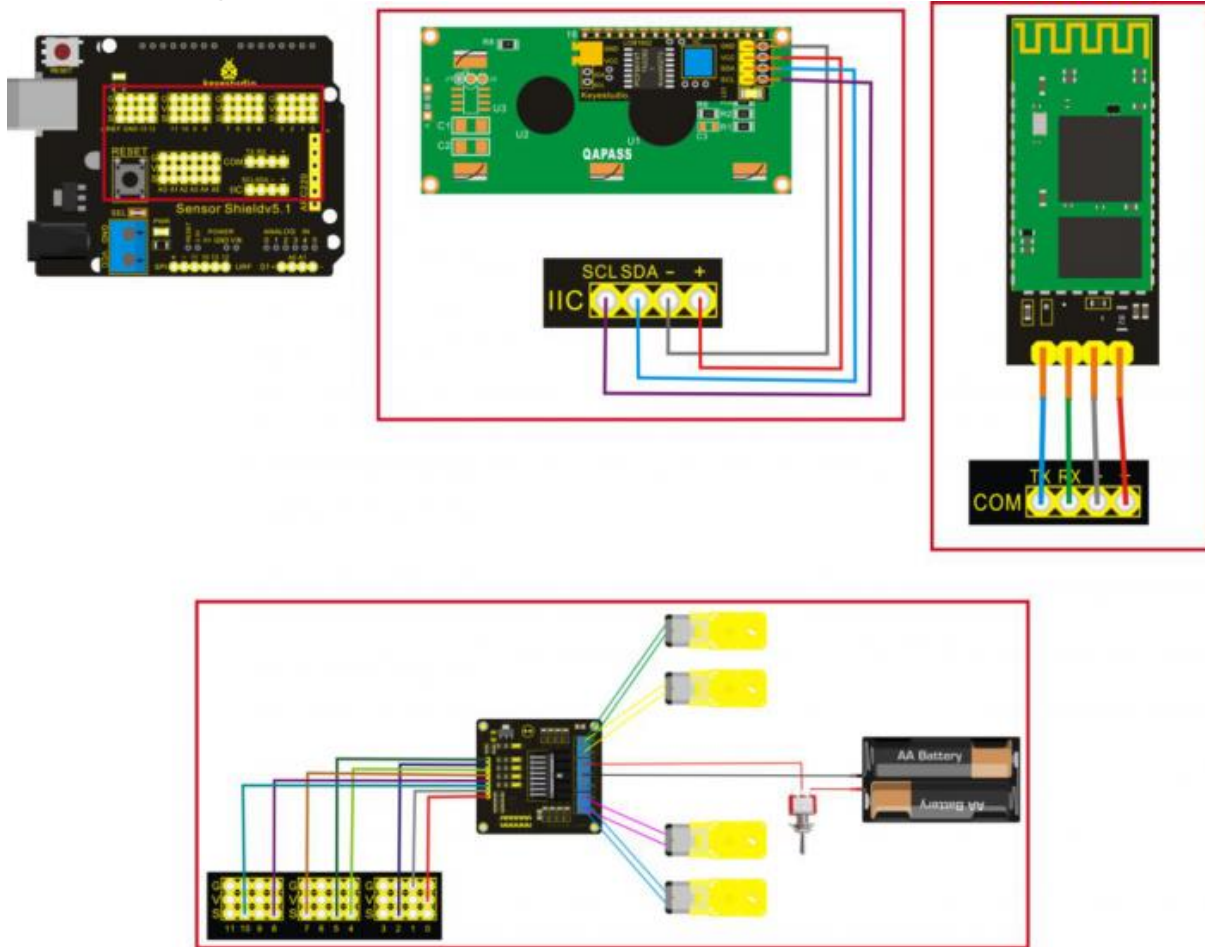
4. When Arduino receiving “U”, the car goes forward; when receiving “D”, it goes backward; when receiving “R”, it turns right; when receiving “S”, it stops moving.

**Schematic Diagram :**

Pay more attention that here you must first upload the code to the board and then plug in the Bluetooth module, otherwise the program fails to compile.



## Connection Diagram :



## Sample Code :

```
#include <LiquidCrystal_I2C.h> //including the libraries of I2C-  
LCD1602 liquid crystal  
#include <Wire.h> //including the libraries of I2C  
unsigned char Bluetooth_val; //defining variable val  
LiquidCrystal_I2C lcd(0x27,16,2); //defining liquid crystal  
#define Lpwm_pin 5 //adjusting speed  
#define Rpwm_pin 10 //adjusting speed //  
int pinLB=2; // defining pin2 left rear  
int pinLF=4; // defining pin4 left front  
int pinRB=7; // defining pin7 right rear  
int pinRF=8; // defining pin8 right front  
unsigned char Lpwm_val = 255;  
unsigned char Rpwm_val = 255;  
int Car_state=0;  
void M_Control_IO_config(void)  
{  
    pinMode(pinLB,OUTPUT); // pin 2
```

```

pinMode(pinLF,OUTPUT); // pin 4
pinMode(pinRB,OUTPUT); // pin 7
pinMode(pinRF,OUTPUT); // pin 8
pinMode(Lpwm_pin,OUTPUT); // pin 11 (PWM)
pinMode(Rpwm_pin,OUTPUT); // pin 10 (PWM)
}
void Set_Speed(unsigned char Left,unsigned char Right)
{
    analogWrite(Lpwm_pin,Left);
    analogWrite(Rpwm_pin,Right);
}
void advance() // going forward
{
    digitalWrite(pinRB,LOW); // making motor move towards right rear
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,LOW); // making motor move towards left rear
    digitalWrite(pinLF,HIGH);
    Car_state = 1;
    show_state();
}
void turnR() //turning right(dual wheel)
{
    digitalWrite(pinRB,LOW); //making motor move towards right rear
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,HIGH);
    digitalWrite(pinLF,LOW); //making motor move towards left front
    Car_state = 4;
    show_state();
}
void turnL() //turning left(dual wheel)
{
    digitalWrite(pinRB,HIGH);
    digitalWrite(pinRF,LOW ); //making motor move towards right
front
    digitalWrite(pinLB,LOW); //making motor move towards left rear
    digitalWrite(pinLF,HIGH);
    Car_state = 3;
    show_state();
}
void stopp() //stop
{
    digitalWrite(pinRB,HIGH);
    digitalWrite(pinRF,HIGH);
    digitalWrite(pinLB,HIGH);
}

```

```

    digitalWrite(pinLF,HIGH);
    Car_state = 5;
    show_state();
}
void back()          //back up
{
    digitalWrite(pinRB,HIGH); //making motor move towards right rear
    digitalWrite(pinRF,LOW);
    digitalWrite(pinLB,HIGH); //making motor move towards left rear
    digitalWrite(pinLF,LOW);
    Car_state = 2;
    show_state() ;
}
void show_state(void)
{
    lcd.setCursor(0, 1);
    switch(Car_state)
    {
        case 1:lcd.print(" Go ");
        break;
        case 2:lcd.print("Back ");
        break;
        case 3:lcd.print("Left ");
        break;
        case 4:lcd.print("Right");
        break;
        case 5:lcd.print("stop ");
        break;
        default:
        break;
    }
}
void LCD1602_init(void)          //including initialized function of
liquid crystal
{
    lcd.init(); //invoking initialized function of LCD in
LiquidCrystal_I2C.h
    delay(10); //delaying for 10 millisecond
    lcd.backlight(); //open backlight of LCD1602
    lcd.clear(); //clear screen
}
void setup()
{
    LCD1602_init();
}

```



```

M_Control_IO_config();
Set_Speed(Lpwm_val,Rpwm_val);
Serial.begin(9600); //initialized serial port , using Bluetooth as
serial port, setting baud at 9600
lcd.setCursor(0, 0); //setting cursor in the first row and column
lcd.print(" Wait Sigal");
stopp();
}
void loop()
{
  lcd.setCursor(0, 0); //setting cursor in the first row and column
  lcd.print("BluetoothControl");

  if(Serial.available()) //to judge whether the serial port receives
the data.
  {
    Bluetooth_val=Serial.read(); //reading (Bluetooth) data of serial
port,giving the value of val;
    switch(Bluetooth_val)
    {
      case 'U':advance(); //UP
      break;
      case 'D': back(); //back
      break;
      case 'L':turnL(); //Left
      break;
      case 'R':turnR(); //Right
      break;
      case 'S':stopp(); //stop
      break;
    }
  }
}
}

```

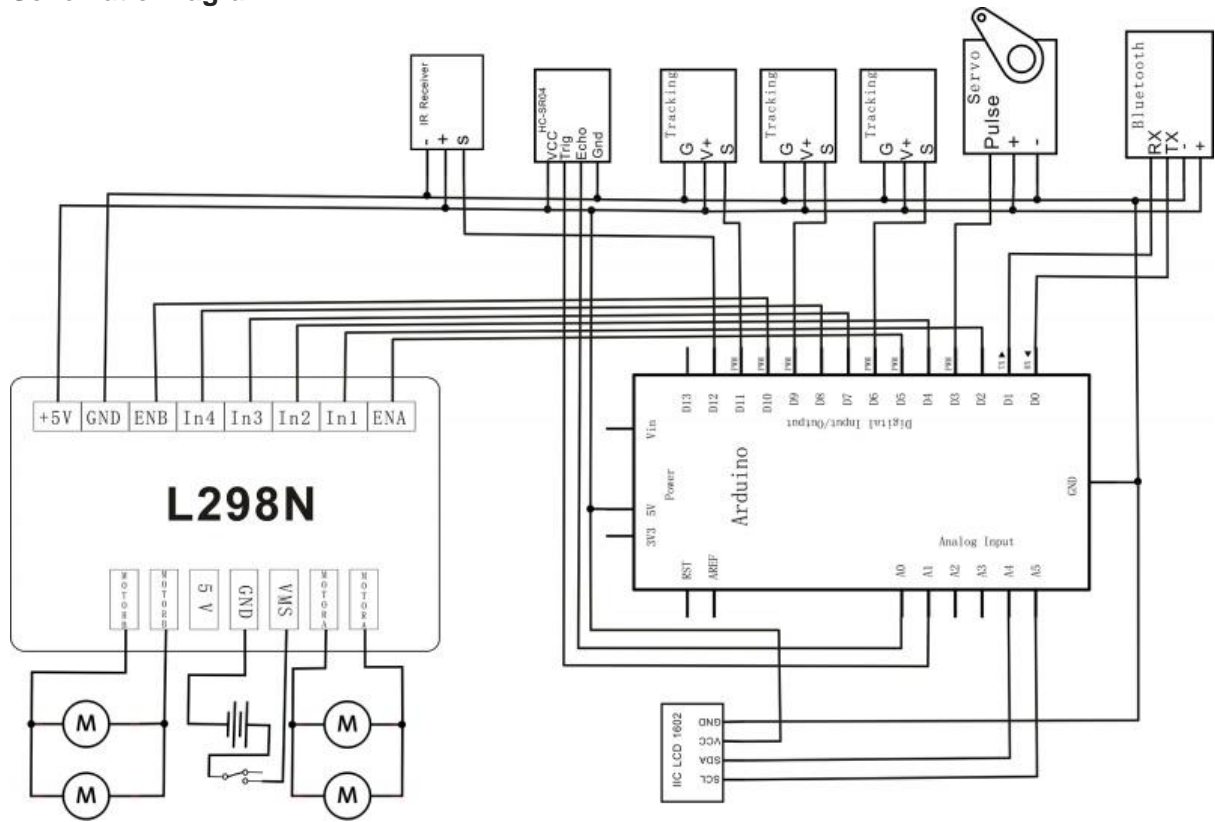
## **Project 13: 5 in 1 Multi-functional Car (Line Tracking, Obstacle Avoidance, Bluetooth and IR Remote Control, Distance Detecting)**



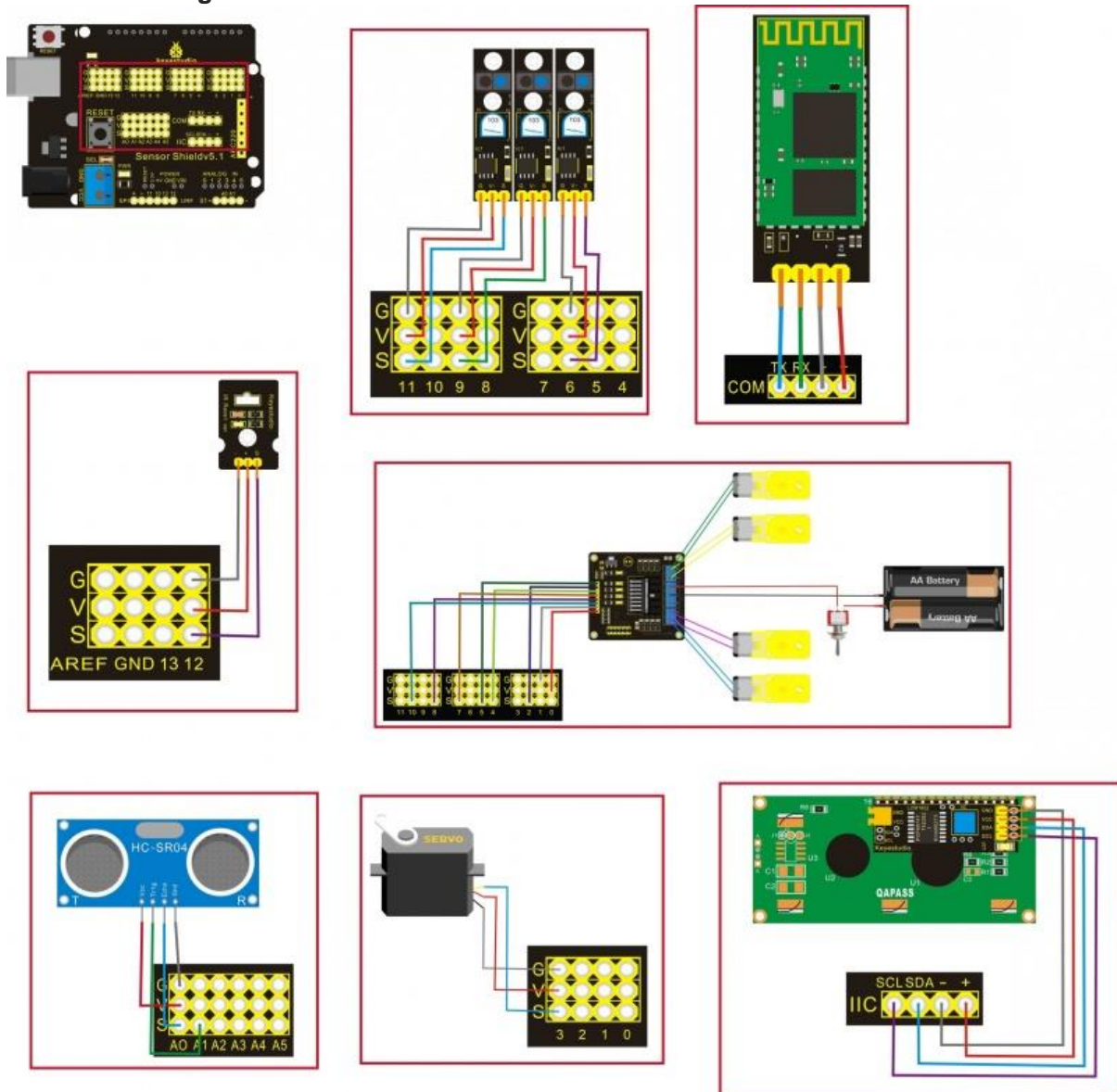
**Introduction :**

In this project, we will put five functions (namely line tracking, obstacle avoidance, Bluetooth and IR remote control, distance detecting) together into one to realize the working mode of the car. Press the button “5” of IR remote control to start line tracking, and button “6” to start ultrasonic obstacle avoidance and distance detecting. Press button “7” to start IR remote control, button “8” to start Bluetooth remote control, and press reset button to enter initialized mode.

Schematic Diagram :



## Connection Diagram :



## Sample Code :

```
#include <LiquidCrystal_I2C.h> //including libraries of I2C-LCD1602
liquid crystal
#include <Wire.h> //including libraries of I2C
#include <IRremote.h> //including libraries of remote control
#define RECV_PIN 12 //pin 12 of IR remoter control receiver
IRrecv irrecv(RECV_PIN); //defining pin 12 of IR remoter control
decode_results results; //cache of decode of IR remoter control
#define IR_Go 0x00ff629d //going forward
#define IR_Back 0x00ffa857 //going backward
#define IR_Left 0x00ff22dd //turning left
#define IR_Right 0x00ffc23d //turning right
```

```

#define IR_Stop      0x00ff02fd  //stop
#define IR_Servo_L   0x00ff6897  //motor turning left
#define IR_Servo_R   0x00ff9867  //motor turning right
#define IR_Speed_UP   0x00ffb04f  //increasing speed
#define IR_Speed_DOWN 0x00ff30cf  //decreasing speed
#define IR_XunJi_Mode 0x00ff18e7
#define IR_Self_Control 0x00ff7a85 //ultrasonic distance detecting
#define IR_IR_Control 0x00ff10ef
#define IR_Bluetooth_Control 0x00ff38c7
#define IR_ESC      0x00ff52ad  //quitting from remote control
////////////////////////////////////
#define SensorLeft   6   //sensor left pin of line tracking module
#define SensorMiddle 9   //sensor middle pin of line tracking module
#define SensorRight  11  //sensor right pin of line tracking module
unsigned char SL;    //state of left sensor of line tracking module
unsigned char SM;    //state of middle sensor of line tracking
module
unsigned char SR;    //state of right sensor of line tracking
module
int inputPin=A0;    // ultrasonic module  ECHO to A0
int outputPin=A1;   // ultrasonic module  TRIG to A1
unsigned char Bluetooth_val;    // ultrasonic module  TRIG to A1
LiquidCrystal_I2C lcd(0x27,16,2); //defining liquid crystal
#define Lpwm_pin  5    //pin of controlling speed---- ENA of motor
driver board
#define Rpwm_pin  10   //pin of controlling speed---- ENB of motor
driver board
int pinLB=2;        //pin of controlling turning---- IN1 of motor
driver board
int pinLF=4;        //pin of controlling turning---- IN2 of motor
driver board
int pinRB=7;        //pin of controlling turning---- IN3 of motor
driver board
int pinRF=8;        //pin of controlling turning---- IN4 of motor
driver board
unsigned char Lpwm_val = 250; //initialized left wheel speed at 250
unsigned char Rpwm_val = 250; //initialized right wheel speed at 250
int Car_state=0;      //the working state of car
int servopin=3;      //defining digital port pin 3, connecting
to signal line of servo motor
int myangle;         //defining variable of angle
int pulsewidth;     //defining variable of pulse width
unsigned char DuoJiao=60; //initialized angle of motor at 60°

```

```

void Sensor_IO_Config()      //IO initialized function of three line
tracking , all setting at input
{
    pinMode(SensorLeft,INPUT);
    pinMode(SensorMiddle,INPUT);
    pinMode(SensorRight,INPUT);
}
void Sensor_Scan(void) //function of reading-in signal of line tracking
module
{
    SL = digitalRead(SensorLeft);
    SM = digitalRead(SensorMiddle);
    SR = digitalRead(SensorRight);
}
void servopulse(int servopin,int myangle) //defining a function of
pulse
{
    pulsewidth=(myangle*11)+500; //converting angle into pulse width value
at 500-2480
    digitalWrite(servopin,HIGH); //increasing the level of motor interface
to upmost
    delayMicroseconds(pulsewidth); //delaying microsecond of pulse width
value
    digitalWrite(servopin,LOW); //decreasing the level of motor interface
to the least
    delay(20-pulsewidth/1000);
}
void Set_servopulse(int set_val)
{
    for(int i=0;i<=10;i++) //giving motor enough time to turn to
assigning point
        servopulse(servopin,set_val); //invoking pulse function
}
void M_Control_IO_config(void)
{
    pinMode(pinLB,OUTPUT); // /pin 2
    pinMode(pinLF,OUTPUT); // pin 4
    pinMode(pinRB,OUTPUT); // pin 7
    pinMode(pinRF,OUTPUT); // pin 8
    pinMode(Lpwm_pin,OUTPUT); // pin 11 (PWM)
    pinMode(Rpwm_pin,OUTPUT); // pin10 (PWM)
}
void Set_Speed(unsigned char Left,unsigned char Right) //function of
setting speed

```

```

{
  analogWrite(Lpwm_pin, Left);
  analogWrite(Rpwm_pin, Right);
}
void advance()    // going forward
{
  digitalWrite(pinRB, LOW); // making motor move towards right rear
  digitalWrite(pinRF, HIGH);
  digitalWrite(pinLB, LOW); // making motor move towards left rear
  digitalWrite(pinLF, HIGH);
  Car_state = 1;
  show_state();
}
void turnR()      //turning right(dual wheel)
{
  digitalWrite(pinRB, LOW); //making motor move towards right rear
  digitalWrite(pinRF, HIGH);
  digitalWrite(pinLB, HIGH);
  digitalWrite(pinLF, LOW); //making motor move towards left front
  Car_state = 4;
  show_state();
}
void turnL()      //turning left(dual wheel)
{
  digitalWrite(pinRB, HIGH);
  digitalWrite(pinRF, LOW ); //making motor move towards right
front
  digitalWrite(pinLB, LOW); //making motor move towards left rear
  digitalWrite(pinLF, HIGH);
  Car_state = 3;
  show_state();
}
void stopp()      //stop
{
  digitalWrite(pinRB, HIGH);
  digitalWrite(pinRF, HIGH);
  digitalWrite(pinLB, HIGH);
  digitalWrite(pinLF, HIGH);
  Car_state = 5;
  show_state();
}
void back()       //back up
{
  digitalWrite(pinRB, HIGH); //making motor move towards right rear

```

```

    digitalWrite(pinRF,LOW);
    digitalWrite(pinLB,HIGH); //making motor move towards left rear
    digitalWrite(pinLF,LOW);
    Car_state = 2;
    show_state() ;
}
void show_state(void)
{
    lcd.setCursor(0, 1);
    switch(Car_state)
    {
        case 1:lcd.print(" Go ");Serial.print(" \r\n GO");
        break;
        case 2:lcd.print("Back ");Serial.print(" \r\n Back");
        break;
        case 3:lcd.print("Left ");Serial.print(" \r\n Left");
        break;
        case 4:lcd.print("Right");Serial.print(" \r\n Right");
        break;
        case 5:lcd.print("Stop ");Serial.print(" \r\n Stop");
        break;
        default:
        break;
    }
}
void LCD1602_init(void) //including initialized function of
liquid crystal
{
    lcd.init(); //invoking initialized function of LCD in
LiquidCrystal_I2C.h
    delay(10); //delaying for 10 millisecond
    lcd.backlight(); //open backlight of LCD1602
    lcd.clear(); //clear screen
}
void Show_V(unsigned char V)
{
    lcd.setCursor(11, 0);
    lcd.print("V= ");
    lcd.setCursor(13, 0);
    lcd.print(V,DEC);
    Serial.print("\n Speed = ");
    Serial.print(V,DEC);
}
void Show_DuoJiao(unsigned char Jiao)

```



```

{
    lcd.setCursor(6,1);
    lcd.print("C=  ");
    lcd.setCursor(8, 1);
    lcd.print(Jiao,DEC);
    Serial.print("\n JiaoDu = ");
    Serial.print(Jiao,DEC);
}
void Xunji_Mode(void) //function of line tracking
{
    lcd.setCursor(0, 0); //setting cursor in the first row and column
    lcd.print("Xunji_Mode ");
    Sensor_Scan();
    if (SM == HIGH)// middle sensor in black area
    {
        if (SL == LOW & SR == HIGH) // black on left, white on right, turn left
        {
            turnR();
        }
        else if (SR == LOW & SL == HIGH) // white on left, black on right, turn
right
        {
            turnL();
        }
        else // white on both sides, going forward
        {
            advance();
        }
    }
    else // middle sensor on white area
    {
        if (SL== LOW & SR == HIGH)// black on left, white on right, turn left
        {
            turnR();
        }
        else if (SR == LOW & SL == HIGH) // white on left, black on right, turn
right
        {
            turnL();
        }
        else // all white, stop
        {
            back();
            delay(100);
        }
    }
}

```

```

stopp() ;
}
}
}
void Self_Control(void)//self-going, ultrasonic obstacle avoidance
{
    int H;
    lcd.setCursor(0, 0); //setting cursor in the first row and column
    lcd.print("Self_Ctr          ");
    Show_V(Lpwm_val);
    Set_servopulse(DuoJiao);
    Show_DuoJiao(DuoJiao);
    H = Ultrasonic_Ranging(1);
    delay(300);
    if(Ultrasonic_Ranging(1) < 35)
    {
        stopp();
        delay(100);
        back();
        delay(50);
    }

    if(Ultrasonic_Ranging(1) < 60)
    {
        stopp();
        delay(100);
        Set_servopulse(5);
        Show_DuoJiao(5);
        int L = ask_pin_L(2);
        delay(300);
        Set_servopulse(177);
        Show_DuoJiao(177);
        int R = ask_pin_R(3);
        delay(300);

        if(ask_pin_L(2) > ask_pin_R(3))
        {
            back();
            delay(100);
            turnL();
            delay(400);
            stopp();
            delay(50);
            Set_servopulse(DuoJiao);

```

```

    Show_DuoJiao(DuoJiao);
    H = Ultrasonic_Ranging(1);
    delay(500);
    }

if(ask_pin_L(2)  <= ask_pin_R(3))
{
    back();
    delay(100);
    turnR();
    delay(400);
    stopp();
    delay(50);
    Set_servopulse(DuoJiao);
    Show_DuoJiao(DuoJiao);
    H = Ultrasonic_Ranging(1);
    delay(300);
    }
    if (ask_pin_L(2)  < 35 && ask_pin_R(3)< 35)
    {
        stopp();
        delay(50);
        back();
        delay(50);
    }
}
else
{
    advance();
}
}

int Ultrasonic_Ranging(unsigned char Mode)//function of ultrasonic
distance detecting , MODE=1, displaying, no displaying under other
situation

{
    int old_distance;
    digitalWrite(outputPin, LOW);
    delayMicroseconds(2);
    digitalWrite(outputPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(outputPin, LOW);
    int distance = pulseIn(inputPin, HIGH);  // reading the duration of
high level

```

```

distance= distance/58;    // Transform pulse time to distance
if (Mode==1) {
    lcd.setCursor(11, 1);
    lcd.print("H=   ");
    lcd.setCursor(13, 1);
    lcd.print(distance,DEC);
    Serial.print("\n H = ");
    Serial.print(distance,DEC);
    return distance;
}
else return distance;
}
int ask_pin_L(unsigned char Mode)
{
int old_Ldistance;
digitalWrite(outputPin, LOW);
delayMicroseconds(2);
digitalWrite(outputPin, HIGH);
delayMicroseconds(10);
digitalWrite(outputPin, LOW);
int Ldistance = pulseIn(inputPin, HIGH);
Ldistance= Ldistance/58;    // Transform pulse time to distance
if (Mode==2) {
    lcd.setCursor(11, 1);
    lcd.print("L=   ");
    lcd.setCursor(13, 1);
    lcd.print(Ldistance,DEC);
    Serial.print("\n L = ");
    Serial.print(Ldistance,DEC);
    return Ldistance;
}
else return Ldistance;
}
int ask_pin_R(unsigned char Mode)
{
int old_Rdistance;
digitalWrite(outputPin, LOW);
delayMicroseconds(2);
digitalWrite(outputPin, HIGH); //
delayMicroseconds(10);
digitalWrite(outputPin, LOW);
int Rdistance = pulseIn(inputPin, HIGH);
Rdistance= Rdistance/58;    // Transform pulse time to distance
if (Mode==3) {

```

```

        lcd.setCursor(11, 1);
        lcd.print("R=   ");
        lcd.setCursor(13, 1);
        lcd.print(Rdistance, DEC);
        Serial.print("\n R = ");
        Serial.print(Rdistance, DEC);
        return Rdistance;
    }
    else return Rdistance;
}

void IR_Control(void) //remote control, when pressing"#", it quitting
from the mode
{
    unsigned long Key;
    lcd.setCursor(0,0); //setting cursor in the first row and column
    lcd.print("IR_Ctr   ");
    while(Key!=IR_ESC)
    {
        if(irrecv.decode(&results)) //to judge whether serial port receive
data
        {
            Key = results.value;
            switch(Key)
            {
                case IR_Go:advance(); //UP
                break;
                case IR_Back: back(); //back
                break;
                case IR_Left:turnL(); //Left
                break;
                case IR_Right:turnR(); //Righ
                break;
                case IR_Stop:stopp(); //stop
                break;
                case IR_Servo_L: if(DuoJiao<=180){ //motor turning left
                    DuoJiao+=10;
                    Set_servopulse(DuoJiao);
                    Show_DuoJiao(DuoJiao);}
                break;
                case IR_Servo_R: if(DuoJiao-10>=0){ //motor turning right
                    DuoJiao-=10;
                    Set_servopulse(DuoJiao);
                    Show_DuoJiao(DuoJiao);}
            }
        }
    }
}

```

```

        break;
        case IR_Speed_UP:if(Rpwm_val+10<=250&&Rpwm_val+10<=250){
//increasing speed
                Lpwm_val+=10; Rpwm_val+=10;
                Set_Speed(Lpwm_val,Rpwm_val);
                Show_V(Lpwm_val);
        }

        break;
        case IR_Speed_DOWN:if(Rpwm_val-10>=0&&Rpwm_val-10>=0){
//decreasing speed
                Lpwm_val-=10; Rpwm_val-=10;
                Set_Speed(Lpwm_val,Rpwm_val);
                Show_V(Lpwm_val);
        }

        break;
        default:
        break;
    }
    irrecv.resume(); // Receive the next value
}
}
lcd.clear();
lcd.setCursor(0, 0); //setting cursor in the first row and column
lcd.print(" Wait Signal ");
stopp();
}
void Bluetooth_Control() //Bluetooth remote control
{
    lcd.setCursor(0, 0); //setting cursor in the first row and column
    lcd.print("BluetoothControl");

    if(Serial.available()) //to judge whether serial port receive data
    {
        Bluetooth_val=Serial.read(); //reading value of Bluetooth serial
port, giving the value to val
        switch(Bluetooth_val)
        {
            case 'U':advance(); //UP
            break;
            case 'D': back(); //back
            break;
            case 'L':turnL(); //Left
            break;
            case 'R':turnR(); //Right

```

```

        break;
        case 'S':stopp();    //stop
        break;
    }
}

void setup()
{
    pinMode(servopin,OUTPUT); //setting motor interface as output
    LCD1602_init();           //initializing 1602
    M_Control_IO_config();    //motor controlling the initialization of
    IO
    Set_Speed(Lpwm_val,Rpwm_val); //setting initialized speed
    Set_servopulse(DuoJiao);     //setting initialized motor angle
    Sensor_IO_Config();          //initializing IO of line tracking
    module
    irrecv.enableIRIn();        //starting receiving IR remote
    control signal
    pinMode(inputPin, INPUT);   //starting receiving IR remote
    control signal
    pinMode(outputPin, OUTPUT); //IO of ultrasonic module
    Serial.begin(9600);         //initialized serial port , using
    Bluetooth as serial port, setting baud
    lcd.setCursor(0, 0);        //setting cursor at 0.0
    lcd.print(" Wait Signal "); //LCD printing character string
    stopp();                    //stop
}
void loop()
{

    if (irrecv.decode(&results))
    {
        //when receiving a button
        if(results.value == IR_XunJi_Mode)
        {
            while(IR_XunJi_Mode)
            {
                Xunji_Mode();
            } //pressing "OK" on remote controller, and entering remote
            control mode
        }
        if(results.value == IR_Self_Control)
        {
            while(IR_Self_Control)

```

```

    {
        Self_Control(); //pressing "OK" on remote controller, and
entering remote control mode
    }
}
if(results.value == IR_IR_Control)
{
    while(IR_IR_Control)
    {
        IR_Control(); //pressing "OK" on remote controller, and
entering remote control mode
    }
}
if(results.value == IR_Bluetooth_Control)
{
    while(IR_Bluetooth_Control)
    {
        Bluetooth_Control(); //pressing "OK" on remote controller, and
entering remote control mode
    }
}
irrecv.resume(); // Receive the next value
}
delay(10);
}

```

### **Result:**

The car is controlled by IR remote control and mobile phone Bluetooth together. After connection, press number 5 key of IR remote control, and the car enter into line tracking mode to go along with black line.

When press reset, and then press number 6 key, the car enter into ultrasonic obstacle avoidance mode and it goes with automatic obstacle avoidance, displaying current condition and distance between car and obstacle on LCD.

Press reset and then number 7 key, the car is in IR control, its motion is controlled by IR up, down, right and left key, "OK" means stop.

Number 1 and 2 key control rotation of motor, while 3 and 4 key control move speed. Reset again, the car goes into Bluetooth mode. If connecting to Bluetooth APP, you can control the motion of the car by APP.