

Infrared Controlling Car

Infrared Controlling Car

Infrared Controlling Car



Infrared Controlling Car

The points of section




Infrared remote control is a widely used method for remote control.

The car has been equipped with infrared receiver and thus allows it to be controlled using the infrared remote controller.

Learning parts:

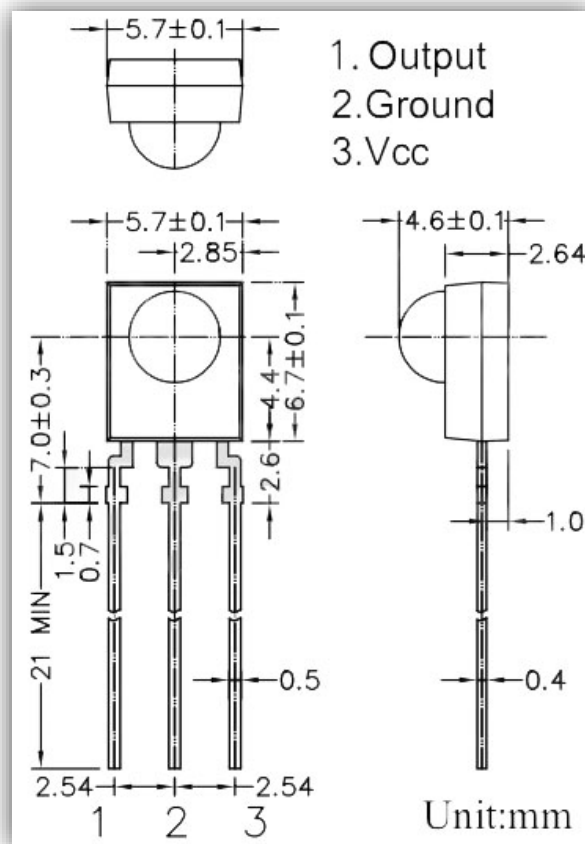
-  *Understand the infrared remote controller and the receiver*
-  *Understand the remote control principles*

Preparations:

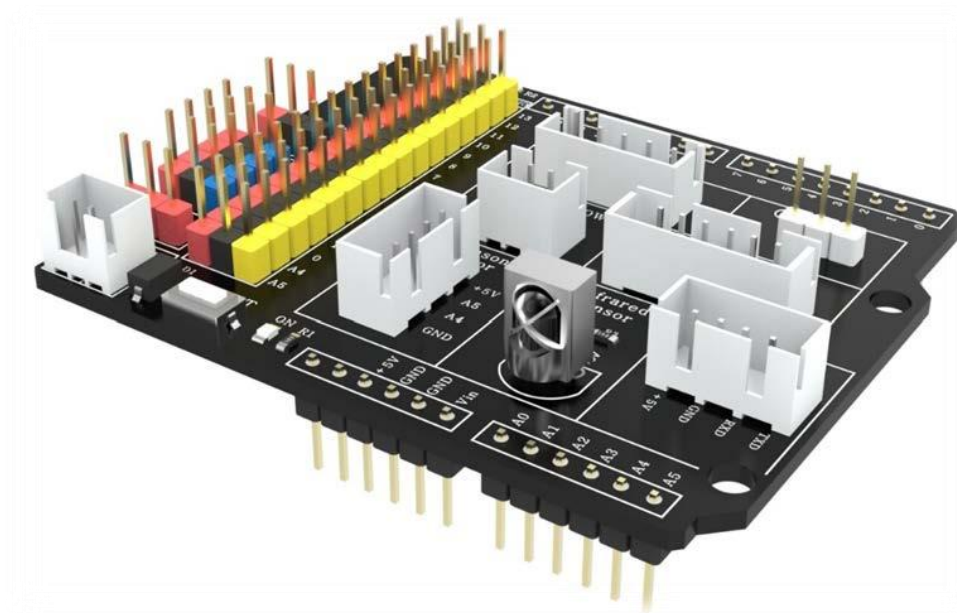
-  *A car (with battery)*
-  *A USB cable*
-  *IR receiving module and IR remote*

I . IR Receiving Module and IR Remote

The data of IR receiver sensor is as below:



The connection of receiver module is as below:

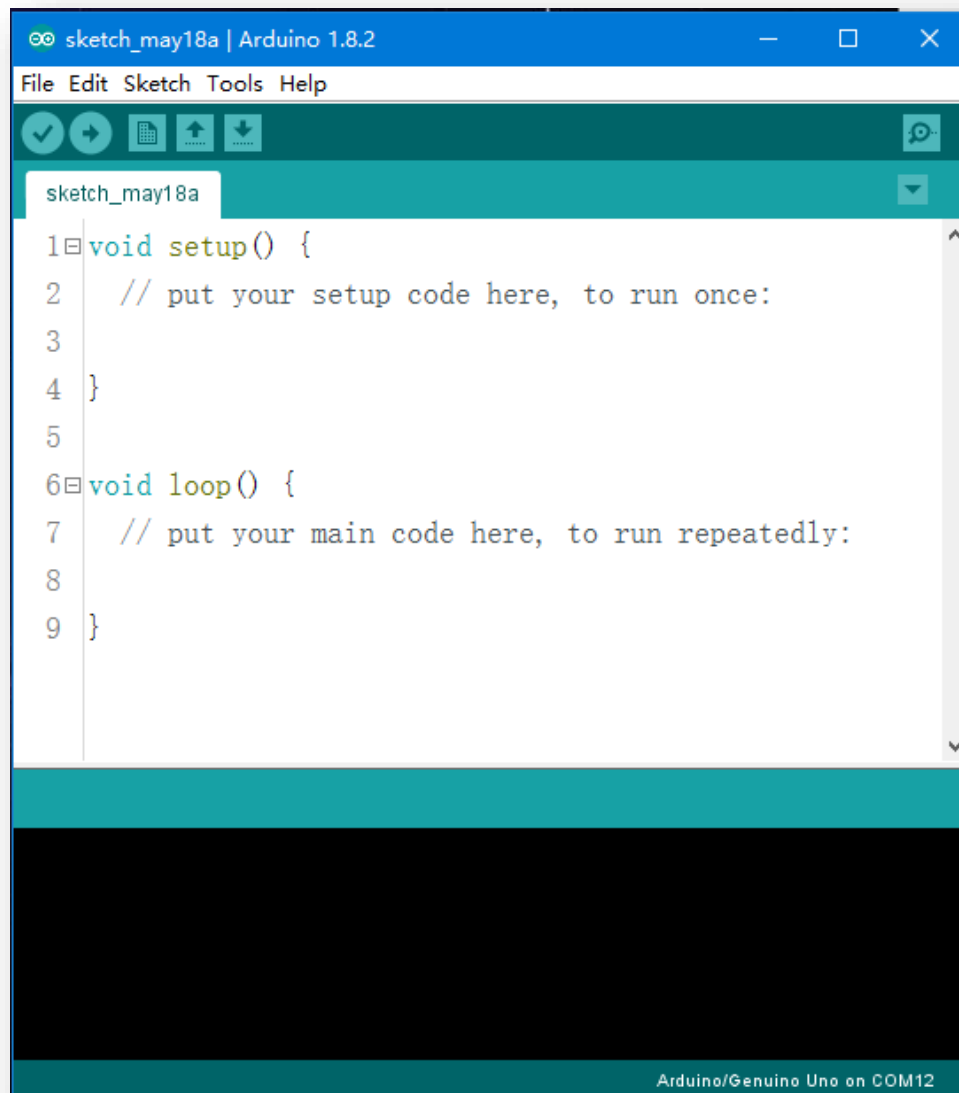


This is IR remote:

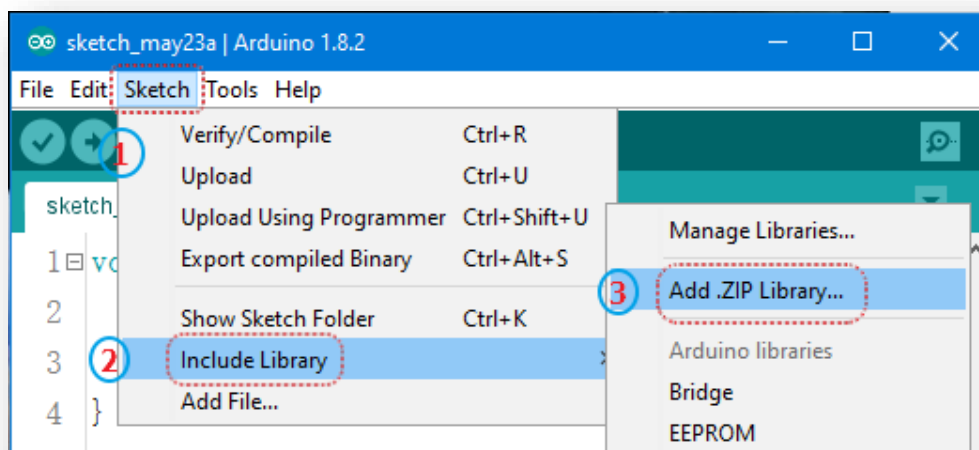


II. Testing program

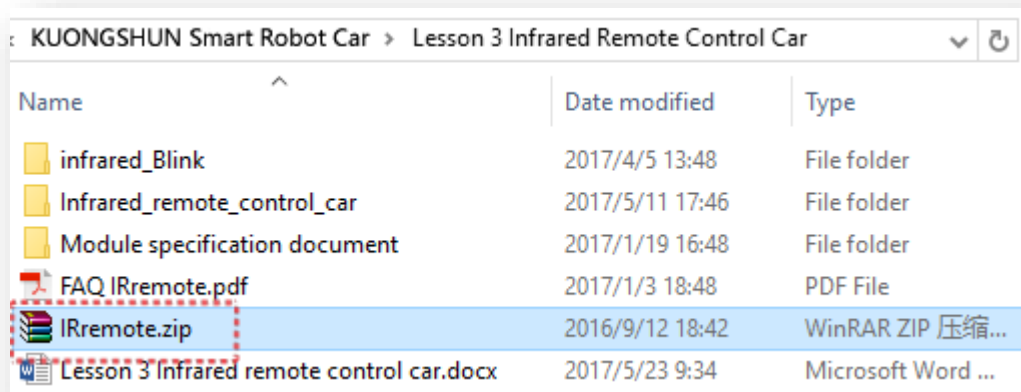
Because in this program, we need to use the library so that we need to add library file at first. Connect the UNO to the computer and open the Arduino IDE.



Click Sketch --> Include Library --> Add .ZIP Library... --> then select the library as blow.

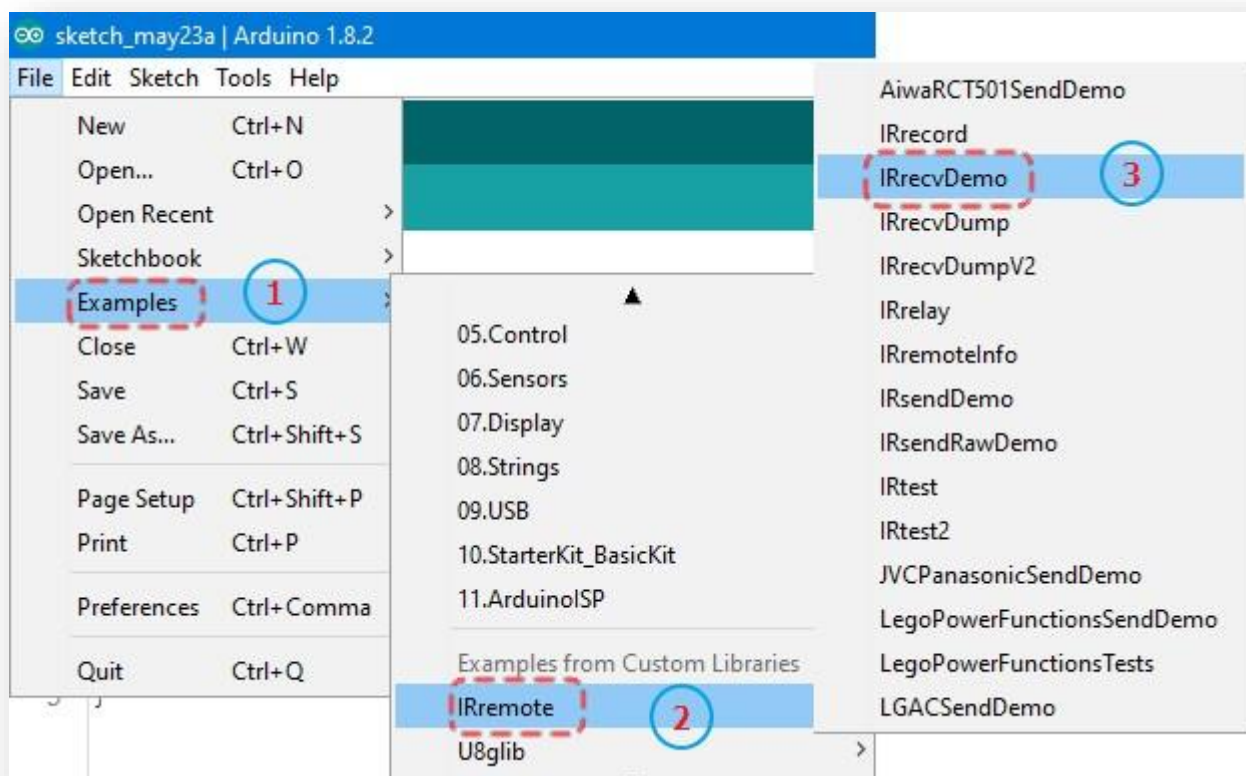


The file name of ZIP library must be IRremote.zip.

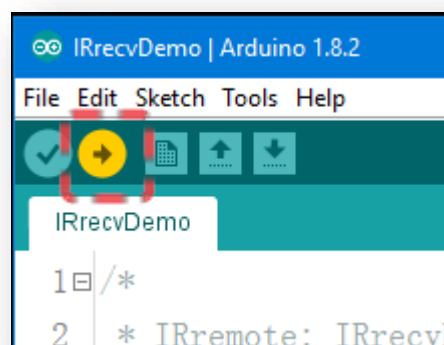


Must be compiled with this library file, which is a specially modified library file.

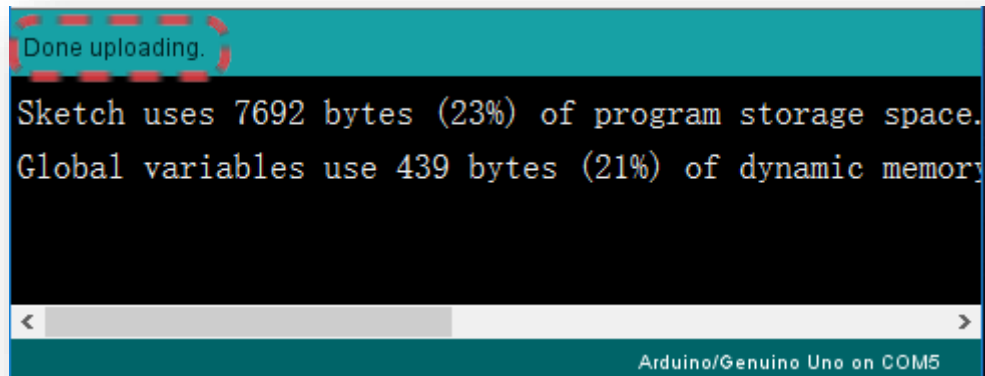
Select an IRremote example



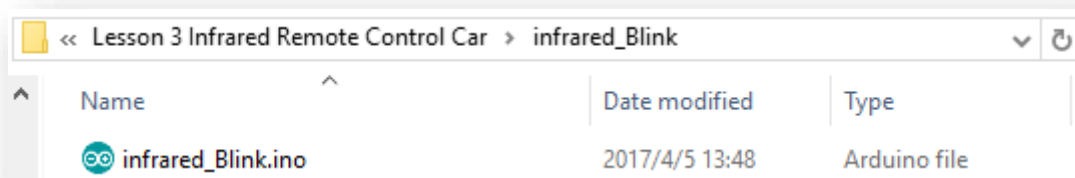
Click compile button.



Done compiling. If not, that the IRremote library was not installed successfully. Please go to add the IRremote library again.



Open the code file in the path “\KUONGSHUN Smart Robot Car Kit V3.0\Lesson 3 Infrared Remote Control Car\infrared_Blink\infrared_Blink.ino” and upload the program to the controller board.



Code preview :

```
#include

<IRremote.h> #define

//Infrared signal receiving pin

#define LED      13      //define LED pin
#define L        16738455
#define UNKNOWN L 1386468383

bool state = LOW;      //define default input mode
unsigned long val;

IRrecv irrecv(RECV_PIN); //initialization
decode_results results; //Define structure type

void stateChange() {
    state = !state;
    digitalWrite(LED, state);
}

void setup() {
```

```

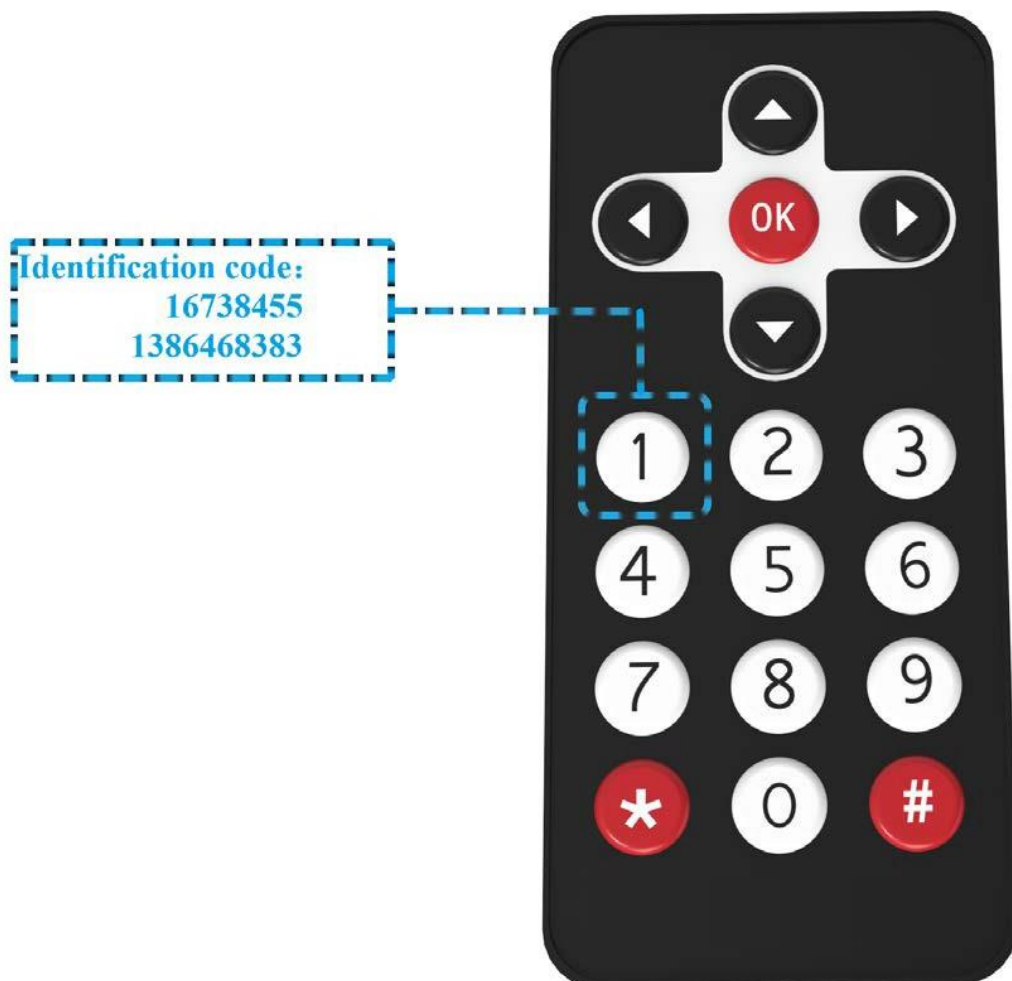
pinMode(LED, OUTPUT); //initialize LED as an output
Serial.begin(9600); // debug output at 9600 baud
irrecv.enableIRIn(); // Start receiving
}

void loop() {
  if (irrecv.decode(&results)) {
    val = results.value;
    Serial.println(val);
    irrecv.resume(); // Receive the next value
    delay(150);
    if(val == L || val == UNKNOWN_L) {
      stateChange();
    }
  }
}

```

After disconnecting the car to the computer, you can turn on the power switch and put the car on the ground.

Press the button "1" facing toward the car, observe the car, and you will find the LED with a label "L" on the extension board turn off.



III. Introduction of principle

1. Working principle

The universal infrared remote controlling system consists of two parts: sending and receiving, the sending part consists of an IR remote controller, the receiving part consists of an infrared receiving tube. The signals that sent by IR remote controlling is a serial of binary pulse code. In order to be free from distraction of other infrared signals during wireless transportation, it's general to modulate it at given carrier frequency, and then launch it through infrared emitted phototransistor. Infrared receiving tube filters out other noise waves, only receives signals of given frequency and restores them to binary pulse code that is demodulation. Built-in receiving tube transform light signals that are sent from infrared light-emitting diode to weak electric signals, signals are enlarged through amplifier inside IC, and through automatic gain controlling, band-pass filtering, demodulation, wave shaping and be restored to original encoding sent by remote control, recognize the circuit by coding that is input to electric appliance through signal output pin of infrared receiving module.

2. Protocol of infrared remote controlling

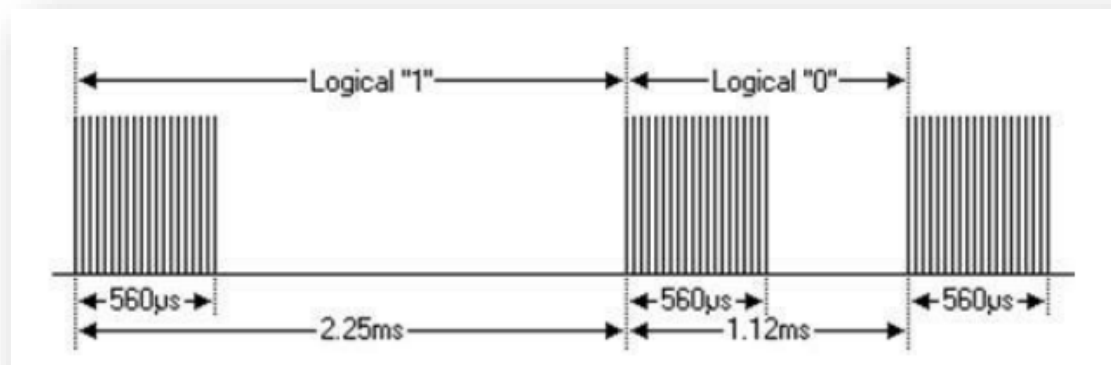
The coding scheme of matched IR remote controlling is: NEC protocol.

Next, let's learn what NEC protocol is.

Features:

- (1) 8 address bit, 8 order bit
- (2) Address bit and order bit are transmitted twice in order to guarantee reliability
- (3) Pulse position modulation
- (4) Carrier frequency is 38kHz
- (5) Time of every bit is 1.125ms or 2.25ms

Definitions of logical 0 and 1 are as below:



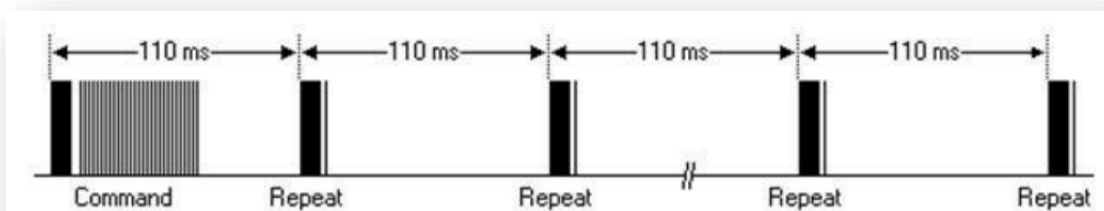
The protocol is as below:

Press instant loosen transmission pulse:



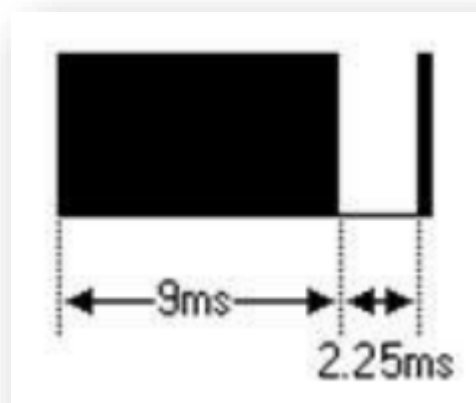
Note: This is protocol of sending LSB (least-significant bit) firstly. Transport address of the above pulse is 0x59, order is 0x16. One message starts from a high level of 9ms, the follow is a low level of 4.5ms, (Two level forms guidance code) and through address code and order code. Address and order are transmitted twice. In the second time, all bits are inverted the opposite, can be used to confirm the receiving messages to be used. Total sending time is fixed, if you are not interested in it, you can ignore reliability of invert, and can expand address and order at 16 bit! Because the fact that length repeat if every bit is opposite.

Press transmitted pulse loosened after a time.



Once a command was sent, even if the button of remote controlling is pressed. When button is still pressed, the pulse of first 110ms is different from above, duplicated code is transmitted after every 110ms. Duplicated code is made up of a high level pulse of 9ms and a low level of 2.25 and a high level of 560μs.

Repeat pulse:



Note: After impulse waveform enters into integration of sensor, owing to the fact that integration of sensor should be decoded, signal magnified and plastic, you should note the time when there are no infrared signals, its output terminal is high level, is low level when there are signals. So the level of output signal is opposite to transmitting terminal. Everybody can see receiver pulse through oscilloscope, understand program with wave form seen.

3. The idea of programming remote control car

According to the characteristic of NEC code and wave of receiving-end, this experiment divides wave of receiving-end into four parts: leading code (Pulse of 9ms and 4.5ms)、address code (including 8-bit address code and 8-bit address fetch)、

16-bit address code (including 8-bit address code and 8-bit address fetch)、16-bit order code(including 8-bit order code and 8-bit order fetch)、repeat code(be made up of pulse of 9ms、 2.25ms、 560us).

Exploit the timer to test high level and low level of wave received, being distinguished according to the time tested: logical“01”、 logical”1”、 leading pulse、 repeat pulse. Leading code and address code are judged whether correct, not be stored, owing to the fact that order code of each key is different, action is carried out by order code.

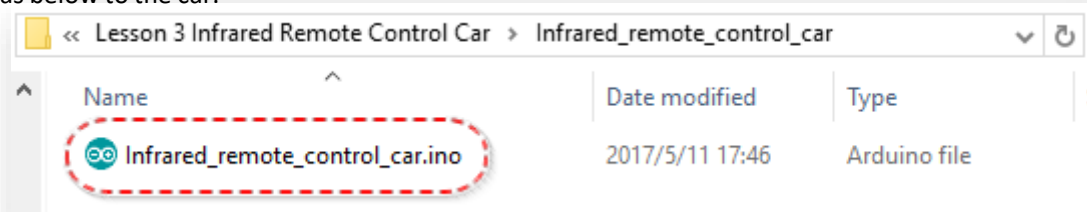
During car experiment, we just need to control the car to go forward and backward ,turn left and right, and stop ,which means we would need 5 keys and the value of them are as below:

Remote control character	key value
Middle red button	16712445, 3622325019
Above triangle	16736925, 5316027
Below triangle	16754775, 2747854299
Left triangle	16720605, 1386468383
Right triangle	16761405, 553536955

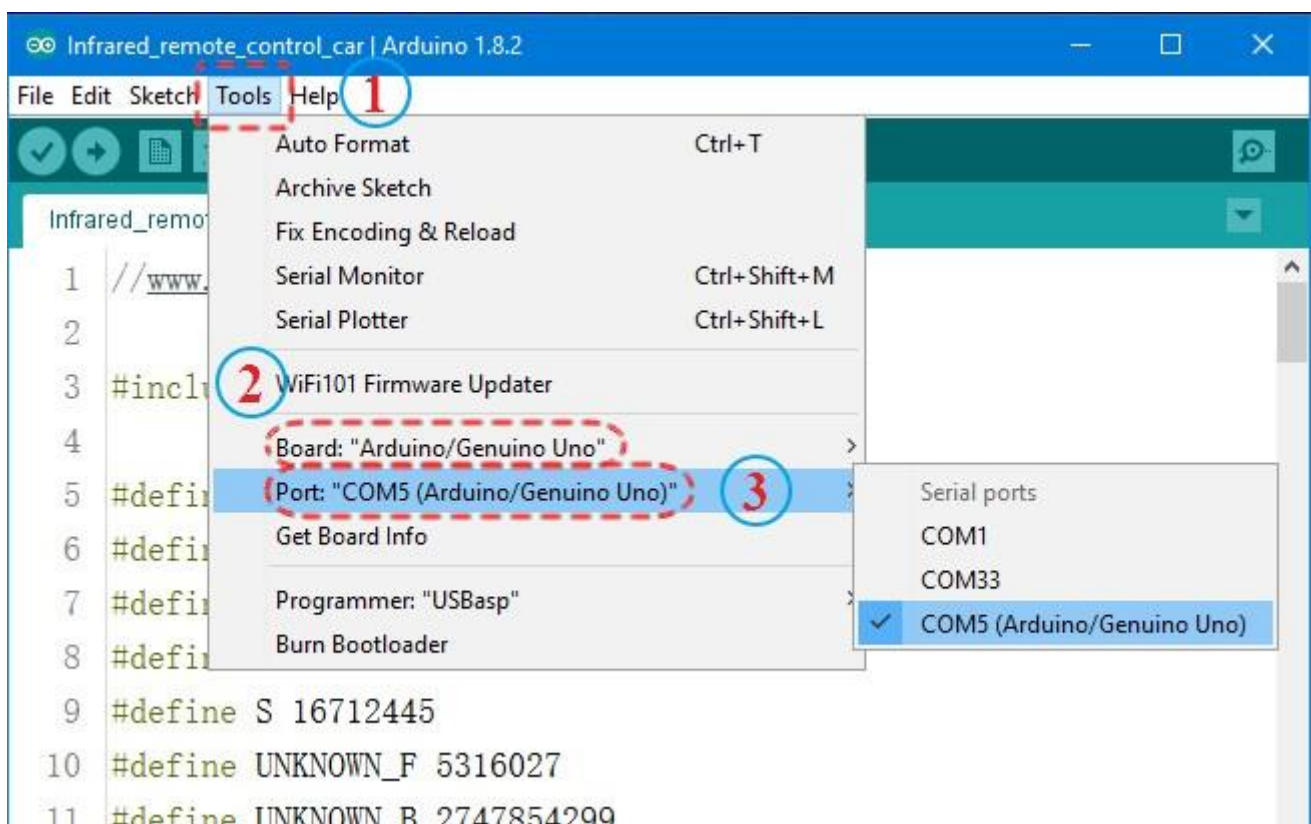


IV. Make a remote controlling car

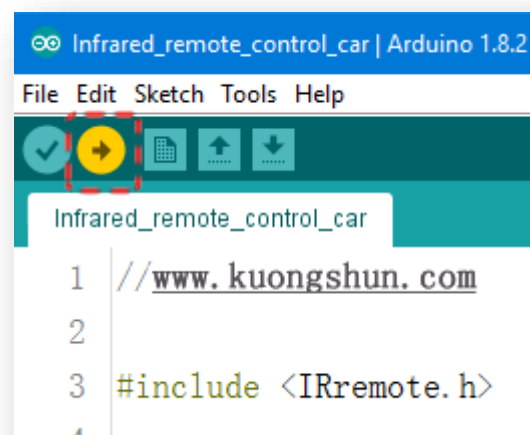
Open the code file in the path“\KUONGSHUN Smart Robot Car Kit V3.0
\Infrared_remote_control_car\ Infrared_remote_control_car.ino” and then upload the program as below to the car.



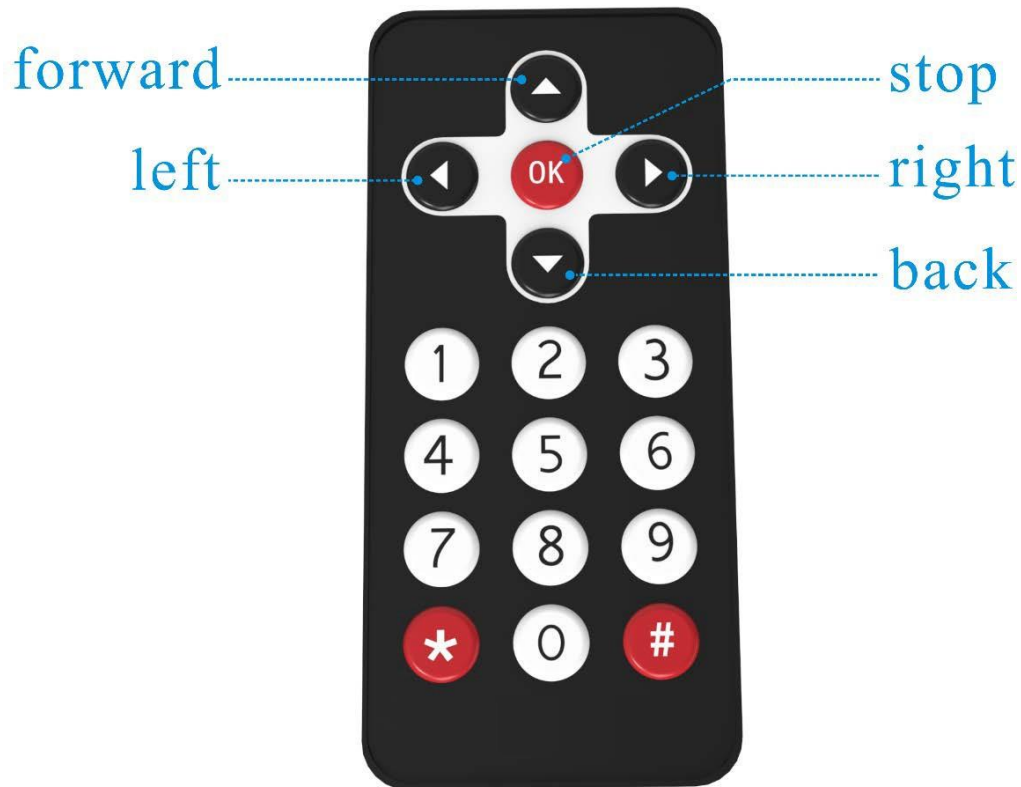
Select the Arduino Uno Board and Serial port.



Press the upload button



When done uploading, disconnecting the car to the computer. Then turn on the power switch and put the car on the ground. Press the button on the remote and you can see the car move accordingly as you command.



Voila, now you can play with the IR controlling car happily.

Code preview:

```
#include <IRremote.h>

#define F 16736925
#define B 16754775
#define L 16720605
#define R 16761405
#define S 16712445

#define UNKNOWN_F 5316027
#define UNKNOWN_B 2747854299
#define UNKNOWN_L 1386468383
#define UNKNOWN_R 553536955
#define UNKNOWN_S 3622325019
```

```

#define RECV_PIN 12
#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11
#define carSpeed 150

IRrecv irrecv(RECV_PIN);
decode_results results;
unsigned long val;
unsigned long preMillis;

void forward() {
    digitalWrite(ENA,HIGH);
    digitalWrite(ENB,HIGH);
    digitalWrite(IN1,HIGH);
    digitalWrite(IN2,LOW);
    digitalWrite(IN3,LOW);
    digitalWrite(IN4,HIGH);
    Serial.println("go forward!");
}

void back() {
    digitalWrite(ENA,HIGH);
    digitalWrite(ENB,HIGH);
    digitalWrite(IN1,LOW);
    digitalWrite(IN2,HIGH);
    digitalWrite(IN3,HIGH);
    digitalWrite(IN4,LOW);
    Serial.println("go back!");
}

void left() {
    analogWrite(ENA,carSpeed);
    analogWrite(ENB,carSpeed);
    digitalWrite(IN1,LOW);
    digitalWrite(IN2,HIGH);
    digitalWrite(IN3,LOW);
    digitalWrite(IN4,HIGH);
    Serial.println("go left!");
}

void right() {
    analogWrite(ENA,carSpeed);

```

```

    analogWrite(ENB,carSpeed);
    digitalWrite(IN1,HIGH);
    digitalWrite(IN2,LOW);
    digitalWrite(IN3,HIGH);
    digitalWrite(IN4,LOW);
    Serial.println("go right!");
}

void stop(){
    digitalWrite(ENA, LOW);
    digitalWrite(ENB, LOW);
    Serial.println("STOP!");
}

void setup() {
    Serial.begin(9600);
    pinMode(IN1,OUTPUT);
    pinMode(IN2,OUTPUT);
    pinMode(IN3,OUTPUT);
    pinMode(IN4,OUTPUT);
    pinMode(ENA,OUTPUT);
    pinMode(ENB,OUTPUT);
    stop();
    irrecv.enableIRIn();
}

void loop() {
    if (irrecv.decode(&results)){
        preMillis = millis();
        val = results.value;
        Serial.println(val);
        irrecv.resume();
        switch(val){
            case F:
            case UNKNOWN_F: forward(); break;
            case B:
            case UNKNOWN_B: back(); break;
            case L:
            case UNKNOWN_L: left(); break;
            case R:
            case UNKNOWN_R: right();break;
            case S:
            case UNKNOWN_S: stop(); break;
            default: break;
        }
    }
}

```

```
}  
else{  
    if(millis() - preMillis > 500){  
        stop();  
        preMillis = millis();  
    }  
}  
}
```