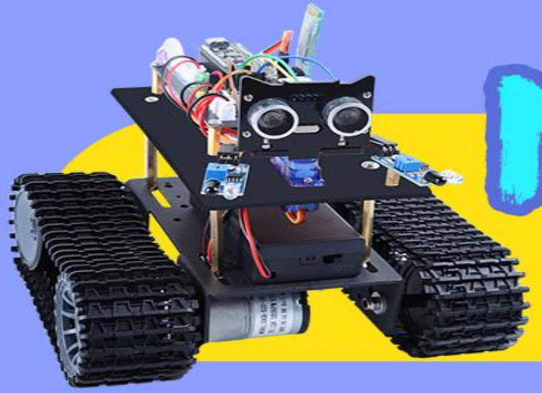


NANOBOT ASSEMBLY TUTORIAL

Lesson 4 Gesture Controlling Car

The Points of Section：

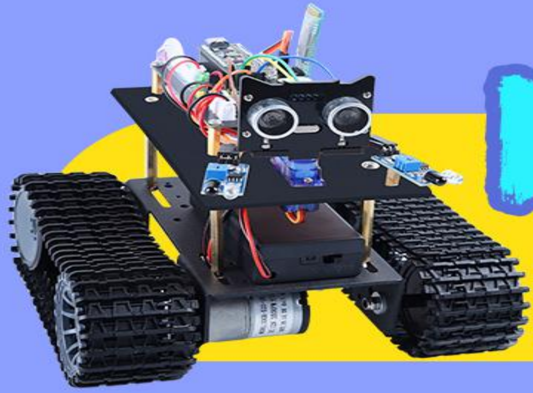
The principle of gesture control of the car is to continuously obtain information from the ADXL335 module through Arduino, receive and send information through the NRF24 module, and then analyze the acquired signals, and then control the car.

Learning Objectives:

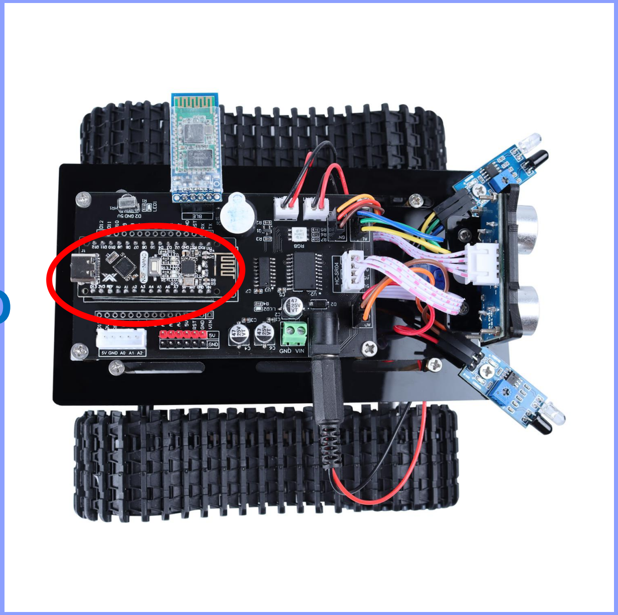
① Learn the principles of ADXL335
② Learn the wireless receiving and sending principle of NRF24 module
③ Learn to use gestures to control car direction

Preparations:

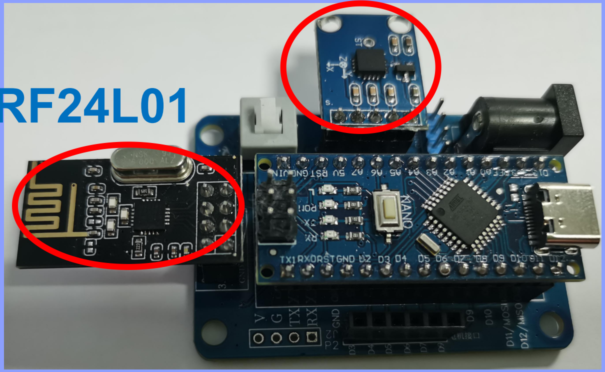
A car (with battery)
A Type-c cable
NRF24 Transmission module



NANOBOT
ASSEMBLY TUTORIAL



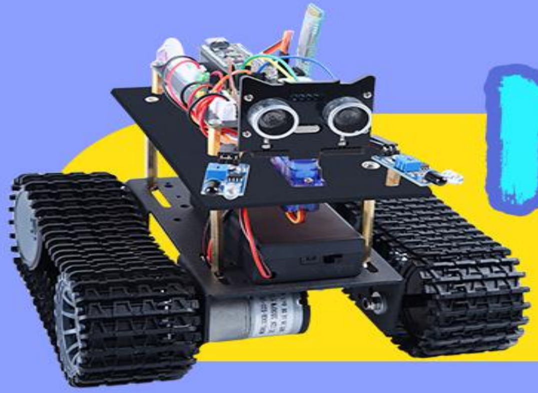
RF-NANO



ADXL335

NRF24L01

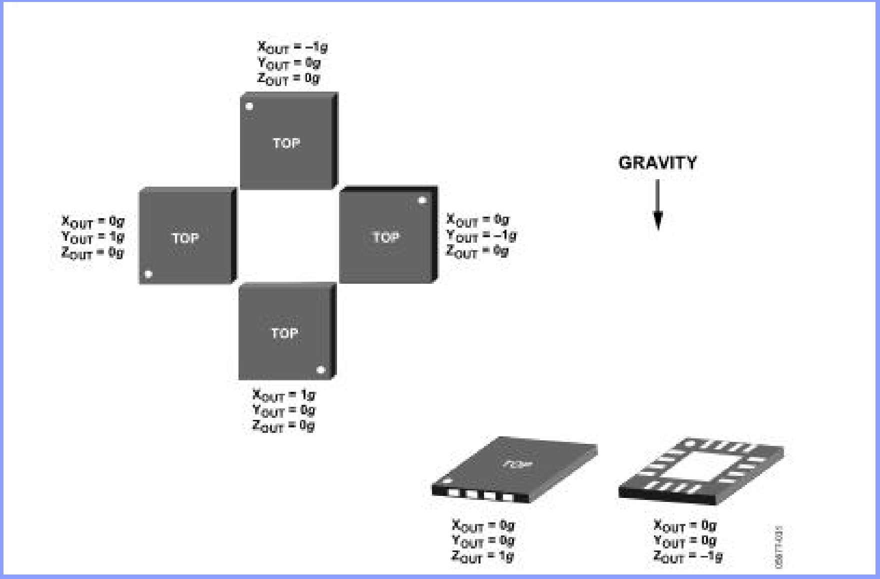
智懿 ZHIYI

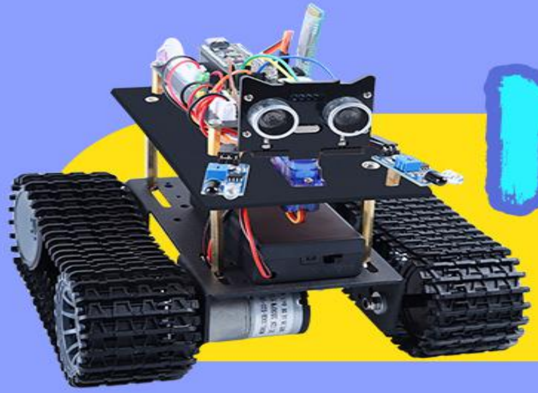


① ADXL335 Module

The data of ADXL335 Module is as below:

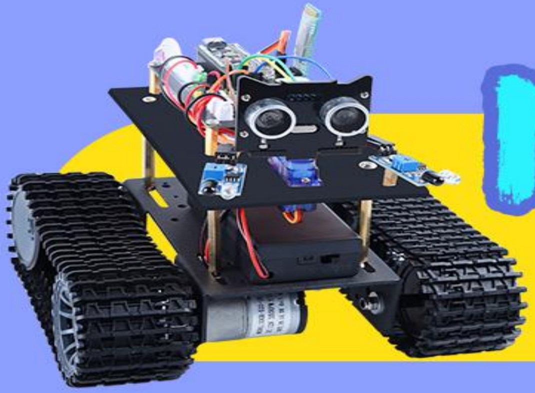
The user selects the accelerometer bandwidth using the capacitors Xout, Yout, and Zout on the Cx, CY, and CZ pins. Depending on the application, the appropriate bandwidth can be selected, ranging from 0.5 Hz to 1600 Hz for the X and Y axes, and 0.5 Hz to 550 Hz for the Z axis.





How does the module work:

The ADXL335 is a complete 3-axis acceleration measurement system. The ADXL335 has a measurement range of ±3 g minimum. It contains a polysilicon surface-micromachined sensor and signal conditioning circuitry to implement an open-loop acceleration measurement architecture. The output signals are analog voltages that are proportional to acceleration. The accelerometer can measure the static acceleration of gravity in tilt-sensing applications as well as dynamic acceleration resulting from motion, shock, or vibration.



peculiarity

1. Operating temperature range: -40 to 85° C

2. Sensitivity: 300mv/g

3. Sensitivity (%): ±10

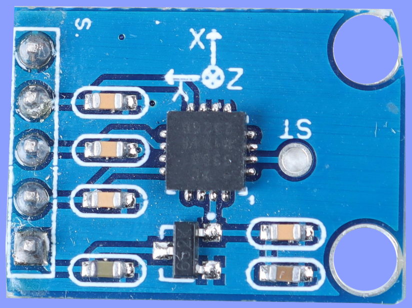
4. Output type: analog output

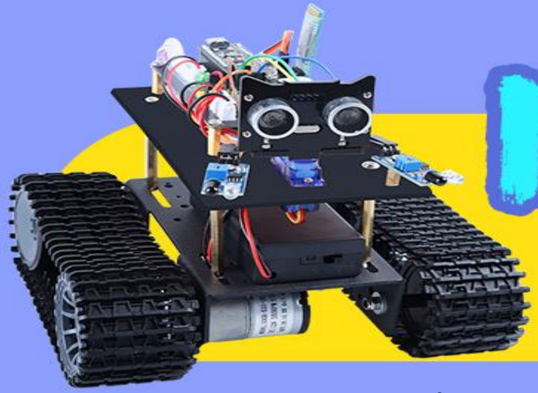
5. Working voltage: DC 5V

6. Low power consumption

7. Typical bandwidth: 500HZ (100nF capacitor connected to the X, Y, and Z terminals)

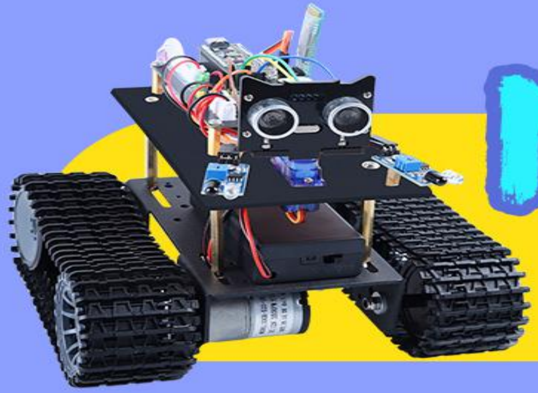
8. The full-scale acceleration measurement range is ±3 g



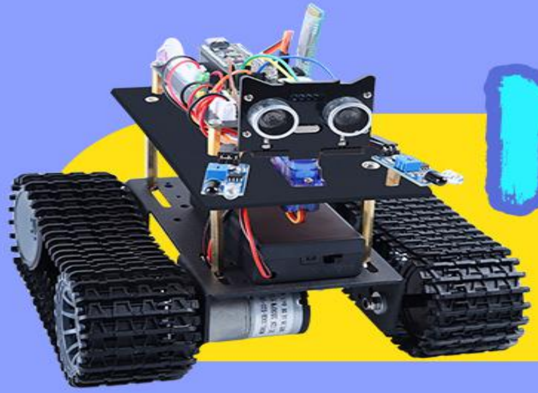


Next, let’s have a look at the RF24L01 sensor module.

The nRF24L01+ is a single chip 2.4GHz transceiver with an embedded baseband protocol engine(Enhanced ShockBurst™), suitable for ultra low power wireless applications. The nRF24L01+ is designed for operation in the world wide ISM frequency band at 2.400 - 2.4835GHz.To design a radio system with the nRF24L01+, you simply need an MCU (microcontroller) and a few exter-nal passive components.



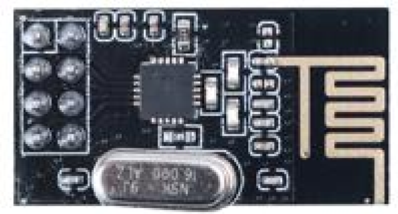
You can operate and configure the nRF24L01+ through a Serial Peripheral Interface (SPI). The register map, which is accessible through the SPI, contains all configuration registers in the nRF24L01+ and is accessible in all operation modes of the chip.The embedded baseband protocol engine (Enhanced ShockBurst™) is based on packet communication and supports various modes from manual operation to advanced autonomous protocol operation. Internal FIFOs ensure a smooth data flow between the radio front end and the system’s MCU. Enhanced Shock-Burst™ reduces system cost by handling all the high speed link layer operations.The radio front end uses GFSK modulation. It has user configurable parameters like frequency channel,output power and air data rate.

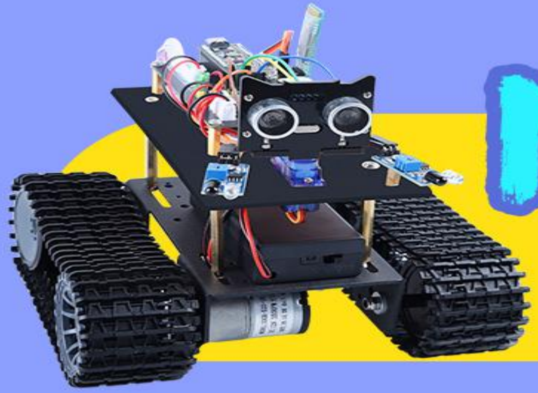


NRF24L01 series communication working principle:

The nRF24L01 series has a built-in state machine that controls the transition between the chip's operating modes. The state machine takes input from user-defined register values and internal signals.Operating mode, nRF24L01 series can be confirmed in shutdown mode, RX or TX mode.

Figure:

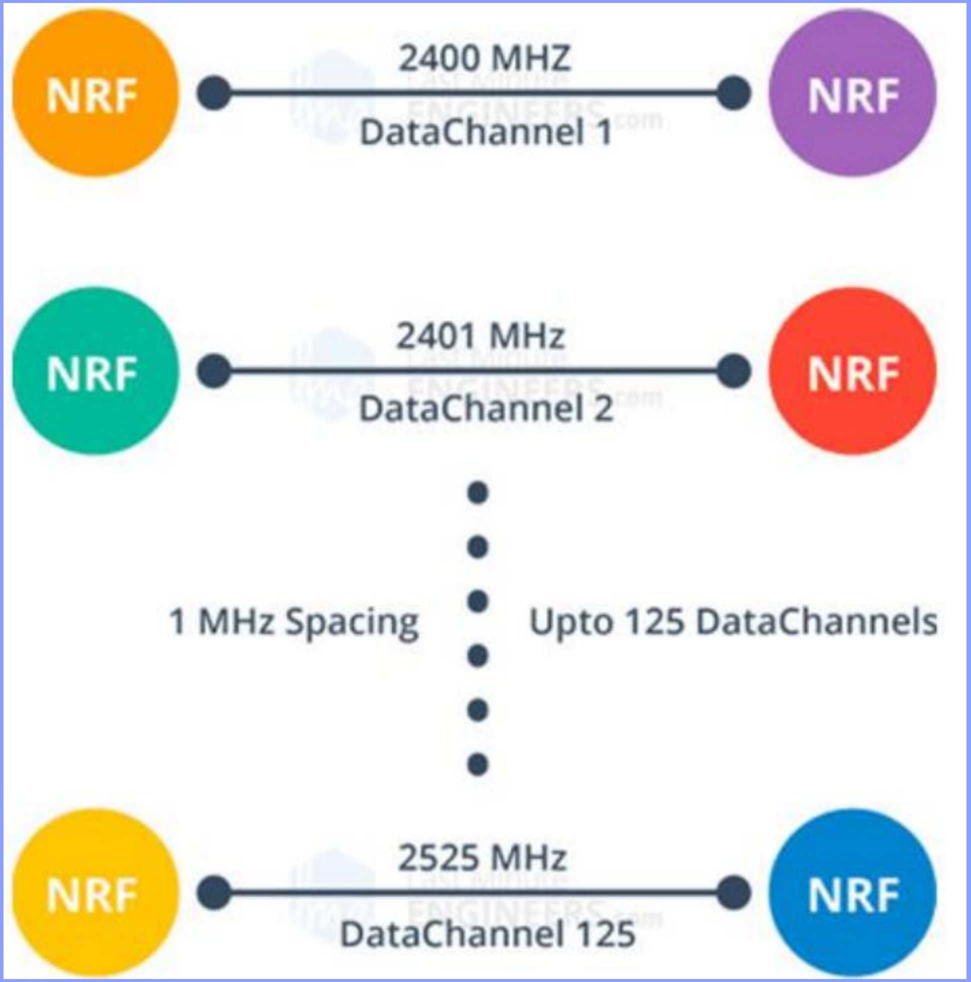




How does the RF24L01 + transceiver module work?

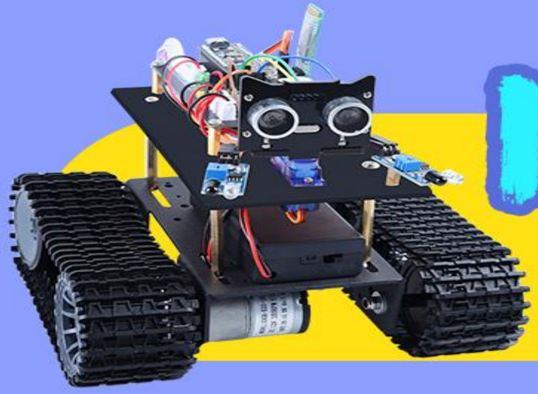
RF channel frequency

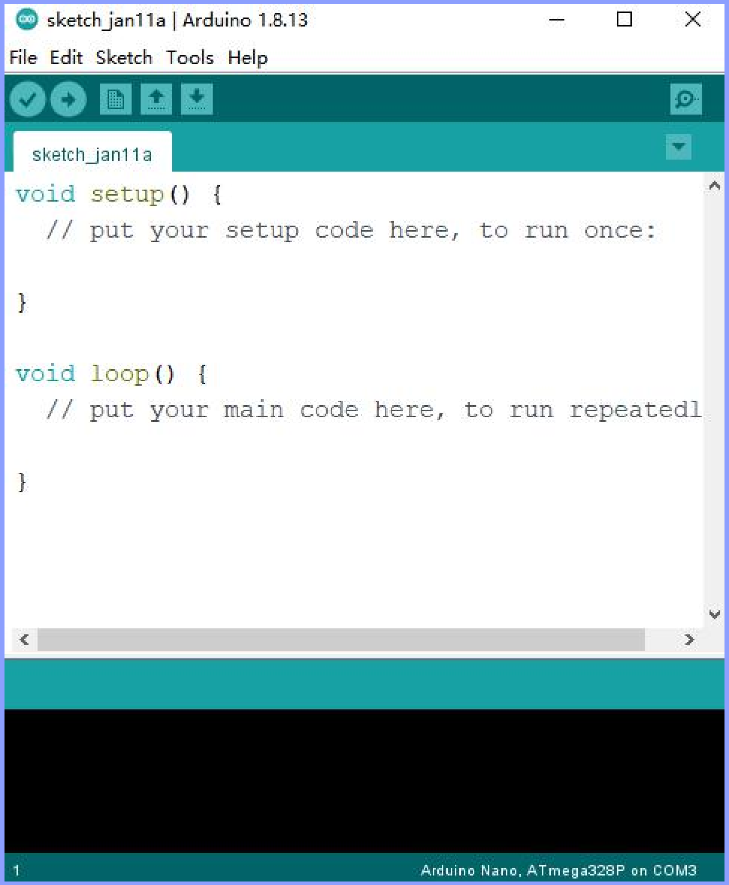
The nRF24L01+ transceiver module sends and receives data on a specific frequency called Channel. Likewise, in order for two or more transceiver modules to communicate with each other, they must be on the same channel. The channel can be any frequency in the 2.4 GHz ISM band, or more accurately, it can be between 2.400 to 2.525 GHz (2400 to 2525 MHz).



The bandwidth occupied by each channel is less than 1MHz. This allows us to provide 125 possible channels at 1MHz intervals.

Therefore, the module can use 125 different channels, making it possible to have a network of 125 independently working modems in one place.




sketch_jan11a | Arduino 1.8.13
File Edit Sketch Tools Help
sketch_jan11a

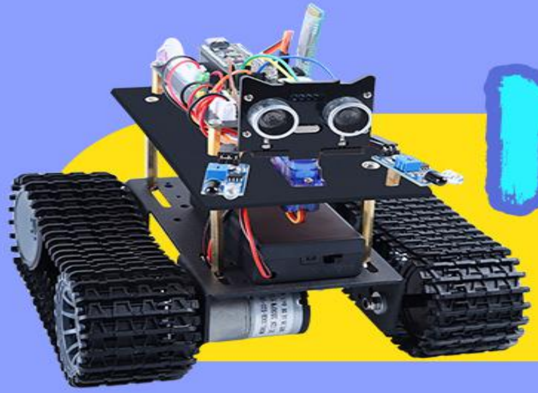
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedl

}
```

1
Arduino Nano, ATmega328P on COM3

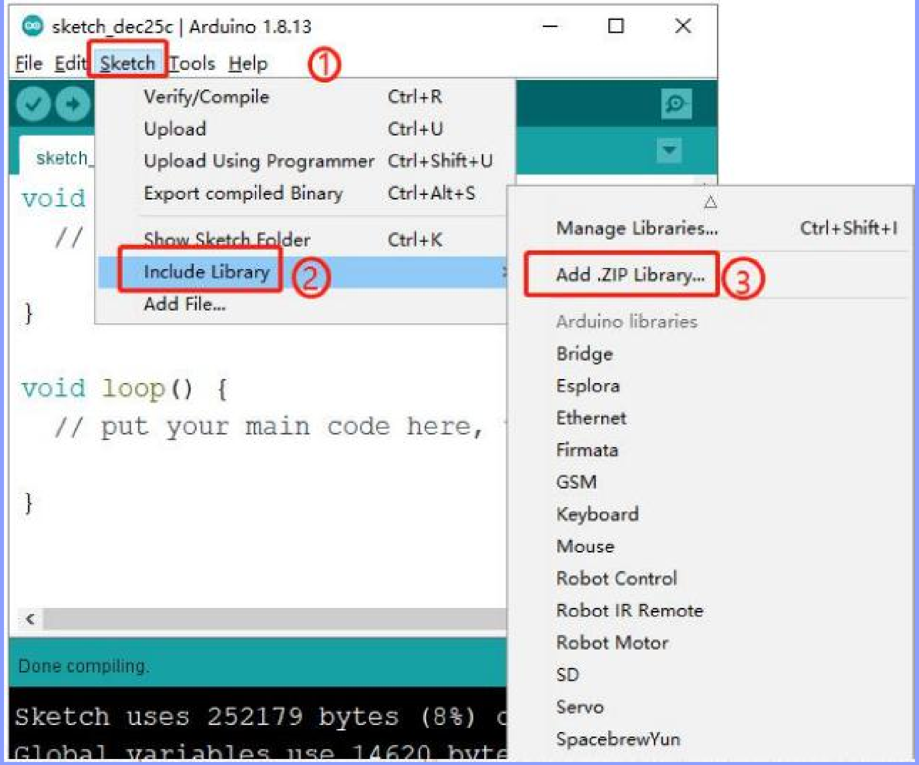
② Testing program

In this program, we need to use the library so that we need to add library file

First, connect the RF-NANO to the computer and open the Arduino IDE.

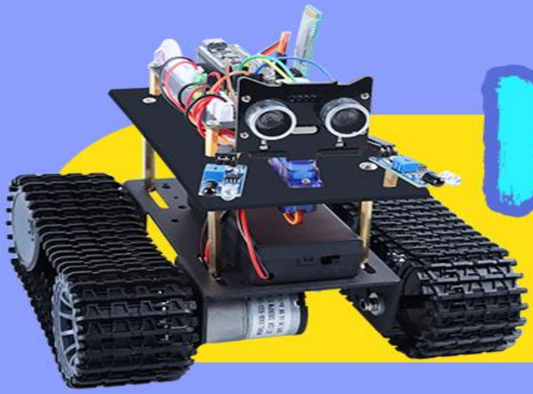


Click
Sketch --> Include Library --> Add .ZIP Library… --> then select the library as below.

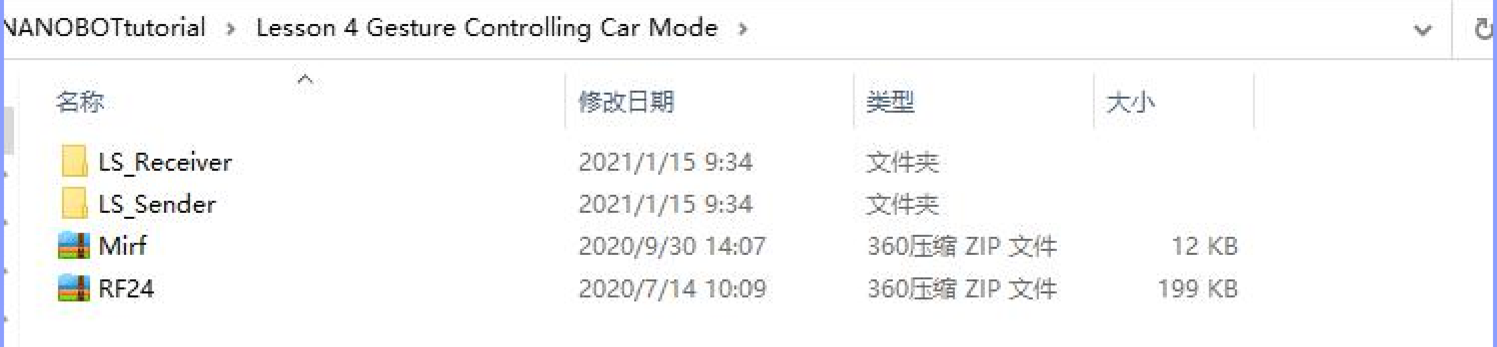


```
void
  //

}

void loop() {
  // put your main code here,

}
```

```
Sketch uses 252179 bytes (8%) c
Global variables use 14620 byte
```



The file name of ZIP library must be RF24 and Mirf.



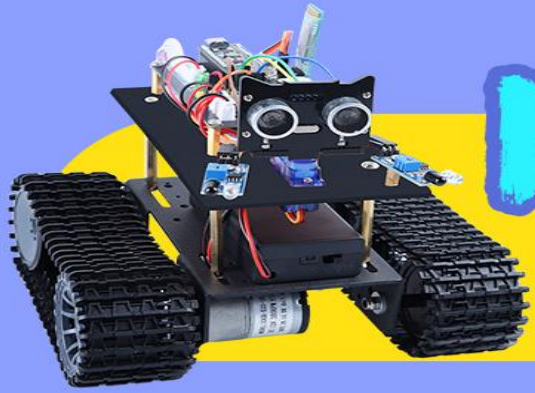
NANOBOTtutorial › Lesson 4 Gesture Controlling Car Mode ›

名称	修改日期	类型	大小
LS_Receiver	2021/1/15 9:34	文件夹	
LS_Sender	2021/1/15 9:34	文件夹	
Mirf	2020/9/30 14:07	360压缩 ZIP 文件	12 KB
RF24	2020/7/14 10:09	360压缩 ZIP 文件	199 KB

Open zip file of tutorial materials:

\Lesson 4 Gesture Controlling Car Mode\RF24. And add it.

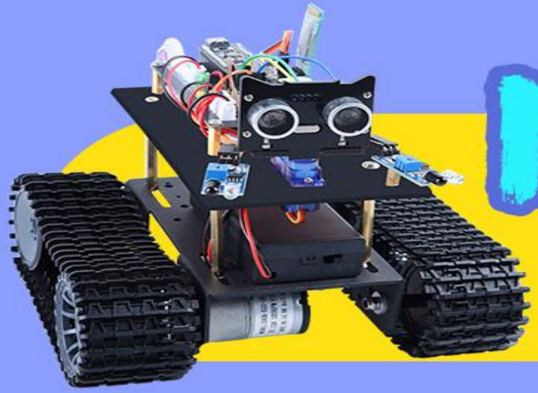
\Lesson 4 Gesture Controlling Car Mode\Mirf. And add it.



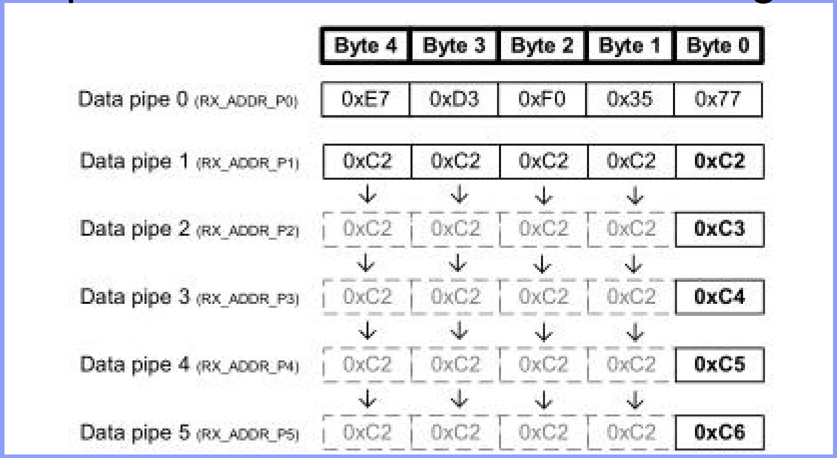
③ Introduction of principle

1.Working principle

The gesture control of the car is made up of RF-Nano (the development board that integrates the NRF24L01 module into the Nano board). The data of ADXL335 module is written to the NRF24L01 sender, and then the data is received through the NRF24L01 receiver. Finally, the received data is read through the RF-Nano to drive the direction of the car.

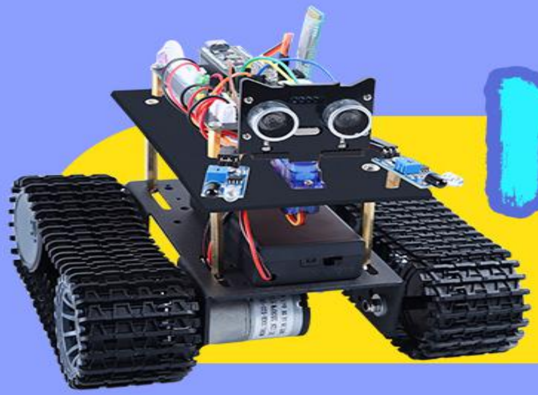


2. protocol Receive and transmitting

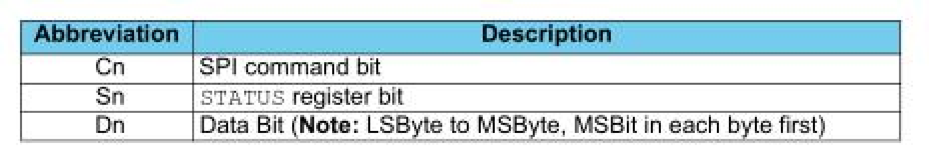


	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Data pipe 0 (RX_ADDR_P0)	0xE7	0xD3	0xF0	0x35	0x77
Data pipe 1 (RX_ADDR_P1)	0xC2	0xC2	0xC2	0xC2	**0xC2**
Data pipe 2 (RX_ADDR_P2)	0xC2	0xC2	0xC2	0xC2	**0xC3**
Data pipe 3 (RX_ADDR_P3)	0xC2	0xC2	0xC2	0xC2	**0xC4**
Data pipe 4 (RX_ADDR_P4)	0xC2	0xC2	0xC2	0xC2	**0xC5**
Data pipe 5 (RX_ADDR_P5)	0xC2	0xC2	0xC2	0xC2	**0xC6**

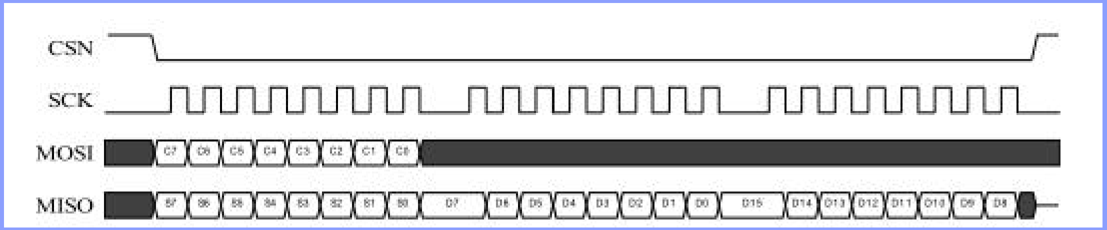
Each pipe can have up to 5 byte configurable address. Data pipe 0 has a unique 5 byte address. Data pipes 1-5 share the 4 most significant address bytes. The LSByte must be unique for all 6 pipes.

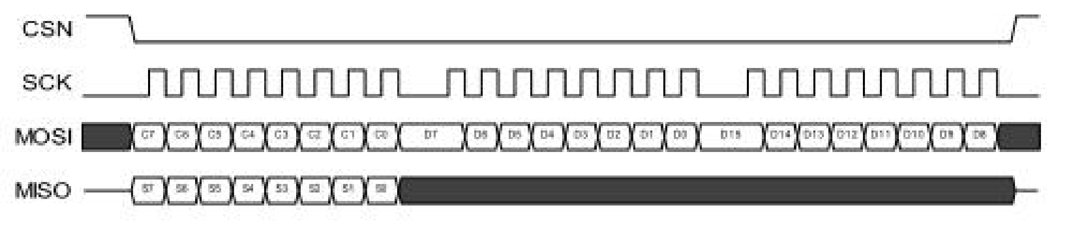


This is SPI Timing of receiving and transmitting NRF24L01

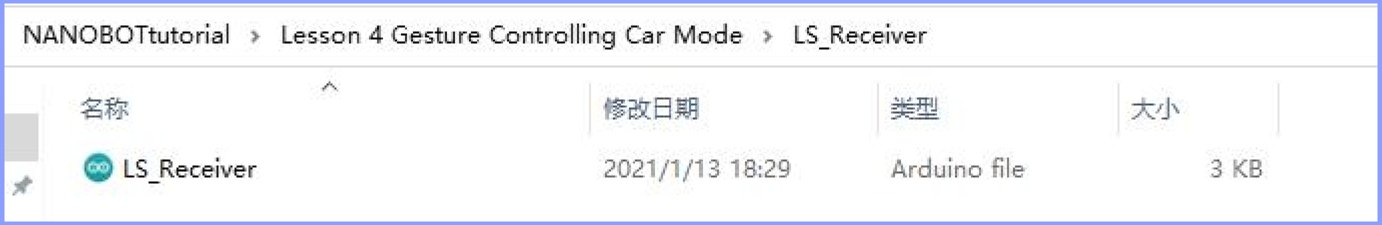


Abbreviation	Description
Cn	SPI command bit
Sn	`STATUS` register bit
Dn	Data Bit (**Note:** LSByte to MSByte, MSBit in each byte first)



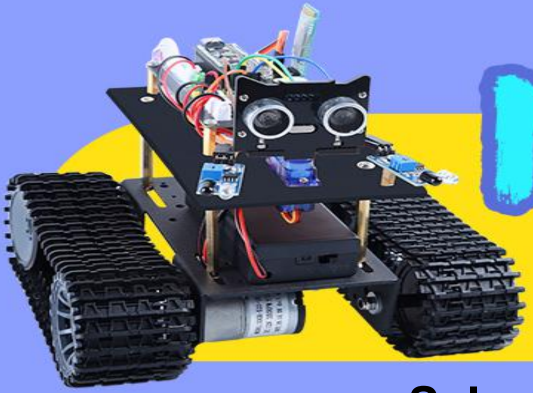


④ upload Receiver program
Open the code file in the path
“\Lesson 4 Gesture Controlling CarMode\LS_Receiver\LS_Receiver.ino”
and then upload the program as below to the car.

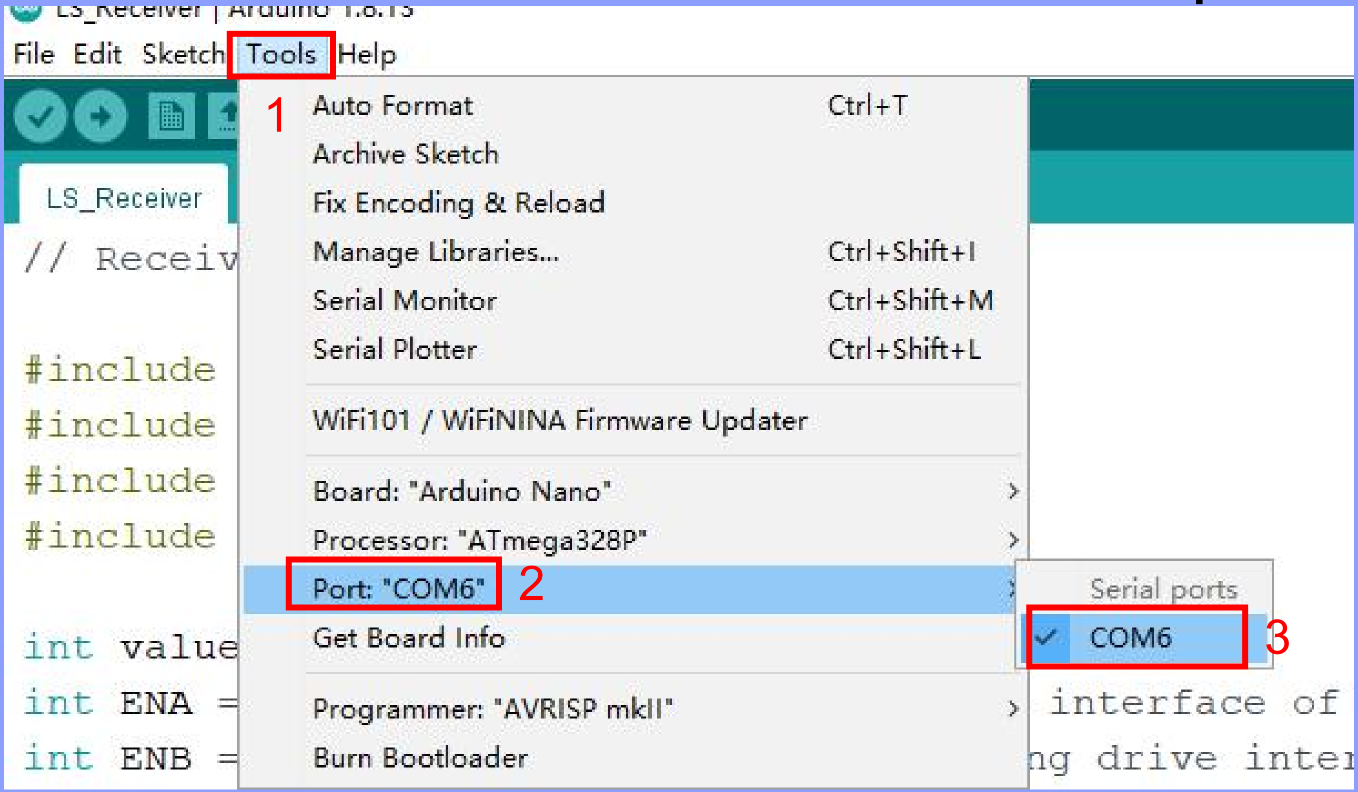


NANOBOTtutorial › Lesson 4 Gesture Controlling Car Mode › LS_Receiver

名称	修改日期	类型	大小
LS_Receiver	2021/1/13 18:29	Arduino file	3 KB



Select the Arduino NANO Board and Serial port.

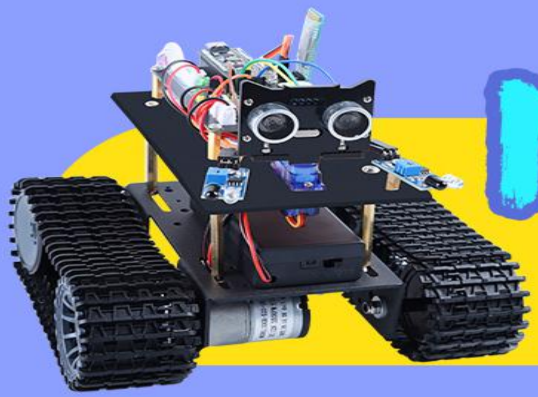


Board: "Arduino Nano"

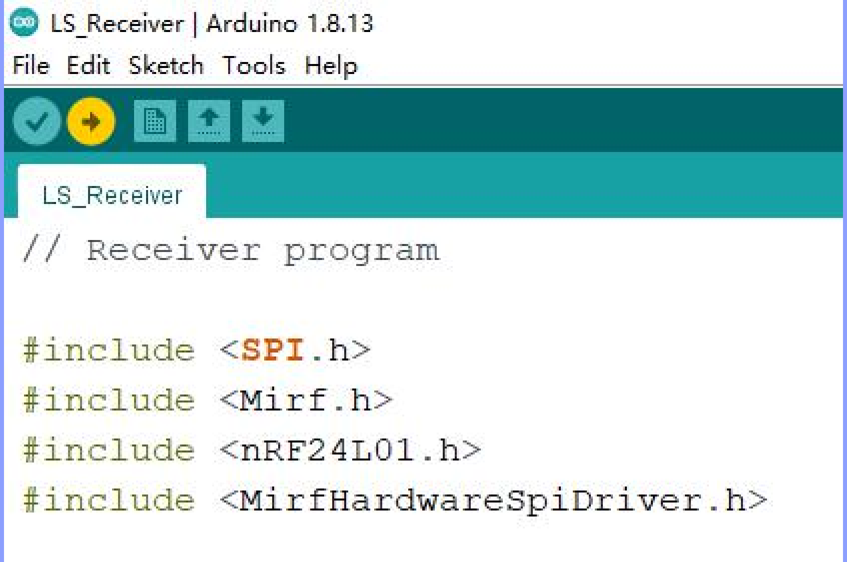
Processor: "ATmega328P"

Port: "COM6"

✓ COM6

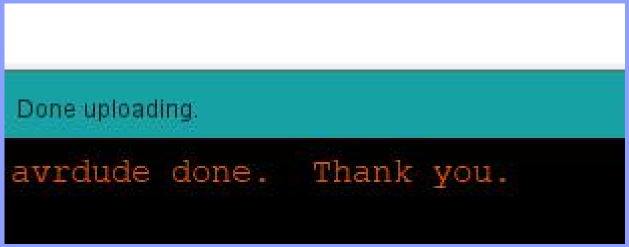


Press the upload button



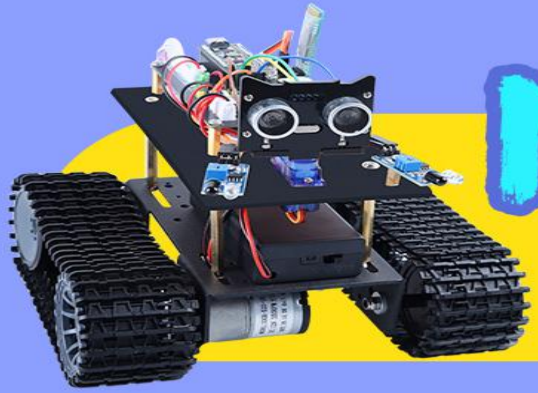
```
// Receiver program

#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>
```

succeed



Done uploading.

```
avrdude done.  Thank you.
```



Working principle of trolley：

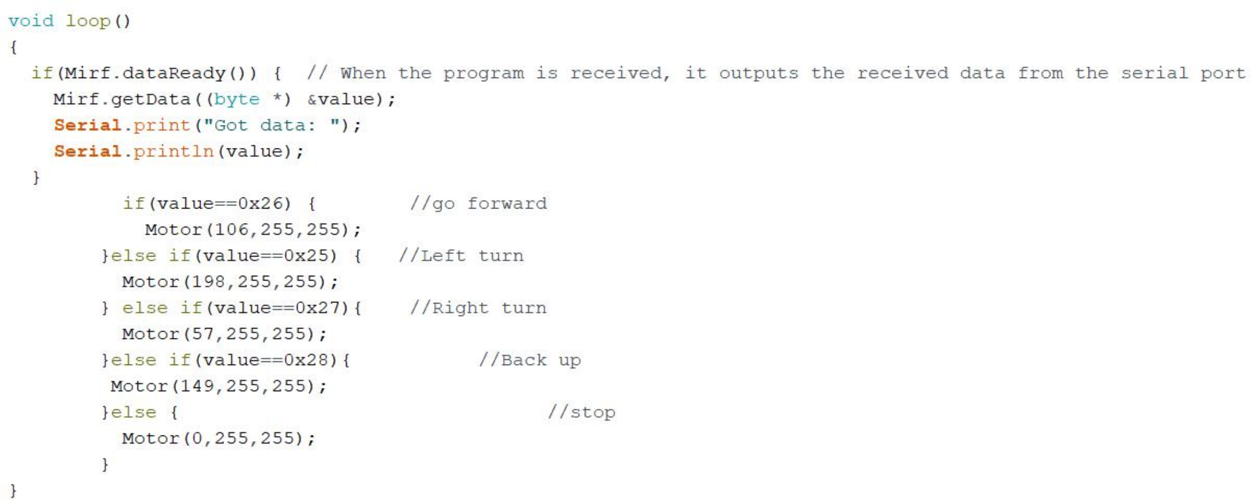
RF-Nano continuously reads the signals received by the RF24L01 module and makes corresponding actions according to the received signals.

When the received signal is 0X26, the car forward;

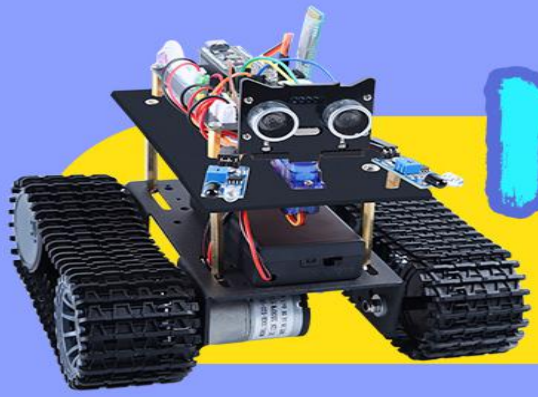
when the received data is 0x25, the car turns left;

when the received signal is 0x27, the car turns right;

when the received signal is 0x28, the car backward.

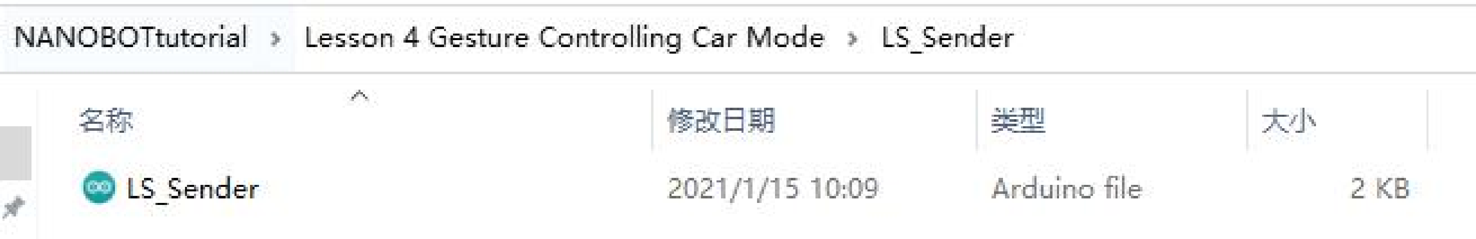


```
void loop()
{
  if(Mirf.dataReady()) {  // When the program is received, it outputs the received data from the serial port
    Mirf.getData((byte *) &value);
    Serial.print("Got data: ");
    Serial.println(value);
  }
          if(value==0x26) {         //go forward
            Motor(106,255,255);
        }else if(value==0x25) {    //Left turn
          Motor(198,255,255);
        } else if(value==0x27){     //Right turn
          Motor(57,255,255);
        }else if(value==0x28){            //Back up
         Motor(149,255,255);
        }else {                                 //stop
          Motor(0,255,255);
        }
}
```



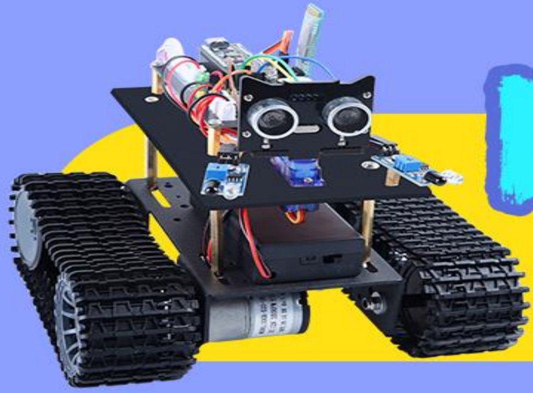
⑤ upload send program

Open the code file in the path “\Lesson 4 Gesture Controlling Car Mode\LS_Sender\LS_Sender.ino” and then upload the program as below to the car.

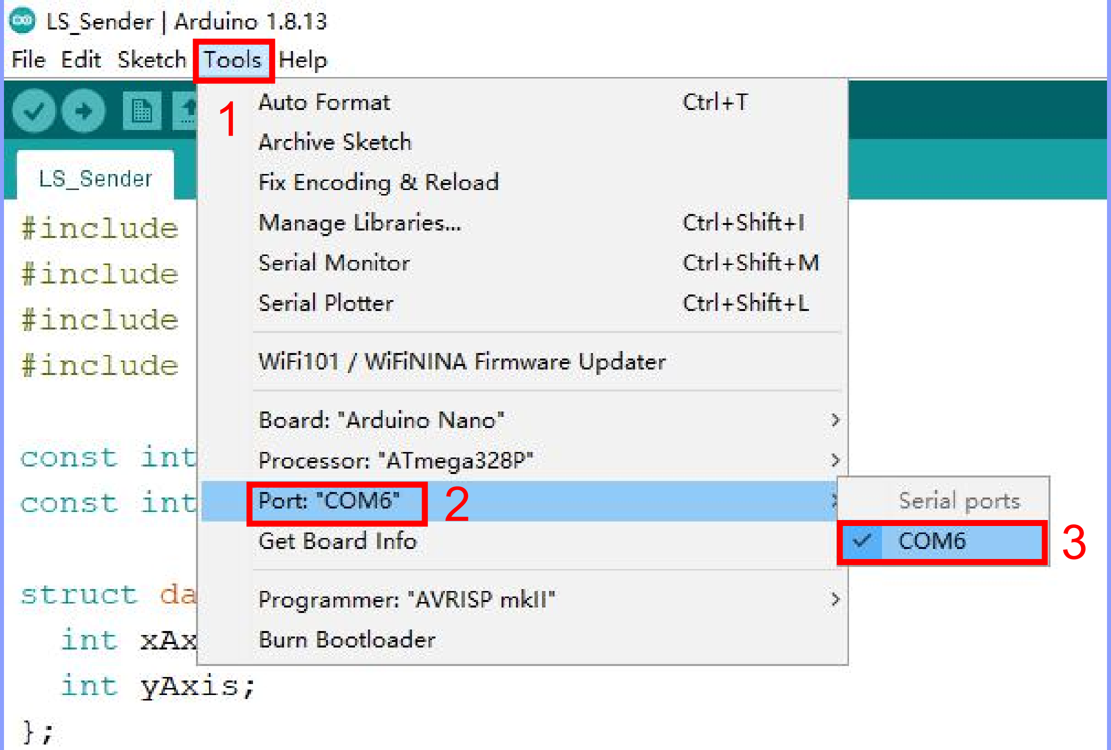


NANOBOTtutorial › Lesson 4 Gesture Controlling Car Mode › LS_Sender

名称	修改日期	类型	大小
LS_Sender	2021/1/15 10:09	Arduino file	2 KB

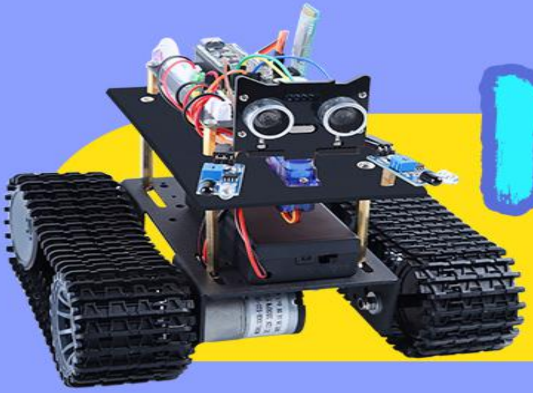


Select the Arduino NANO Board and Serial port.


LS_Sender | Arduino 1.8.13
File Edit Sketch Tools Help
1
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
2
Serial ports
3
LS_Sender

Board: "Arduino Nano"
Processor: "ATmega328P"
Port: "COM6"
✓ COM6

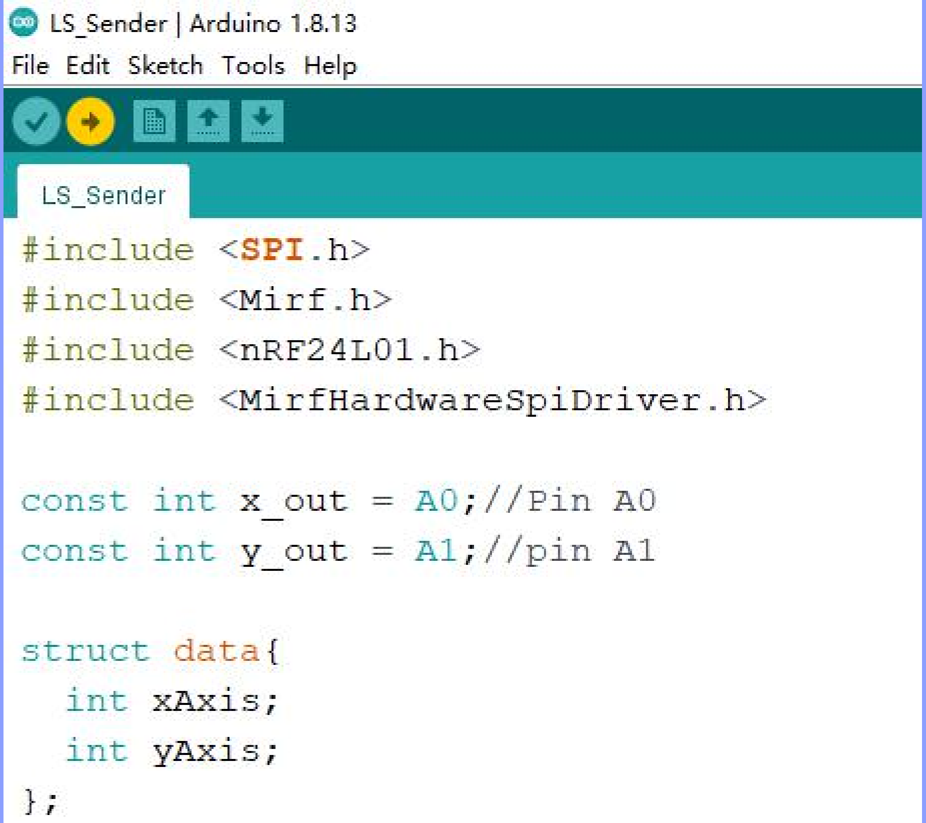
```
#include
#include
#include
#include

const int
const int

struct da
  int xAx
  int yAxis;
};
```



NANOBOT

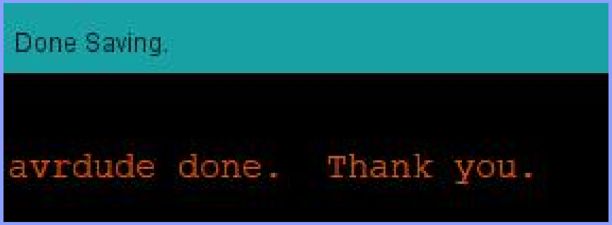
ASSEMBLY TUTORIAL

Press the upload button

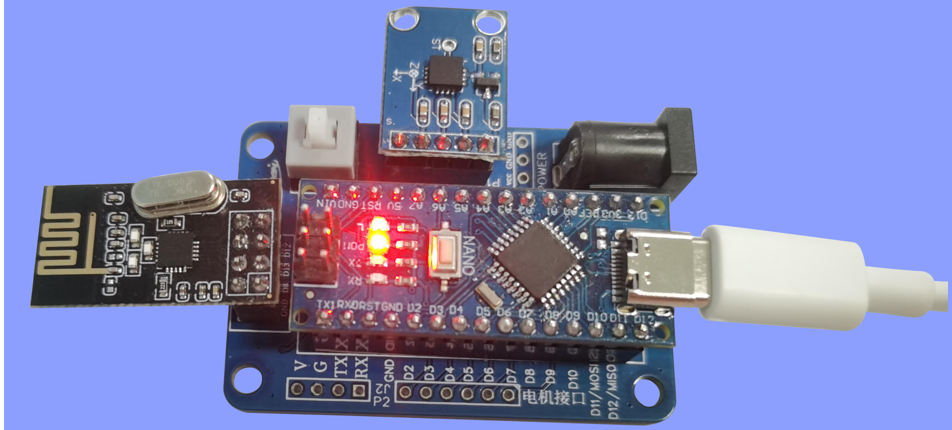


```
#include <SPI.h>
#include <Mirf.h>
#include <nRF24L01.h>
#include <MirfHardwareSpiDriver.h>

const int x_out = A0;//Pin A0
const int y_out = A1;//pin A1

struct data{
  int xAxis;
  int yAxis;
};
```

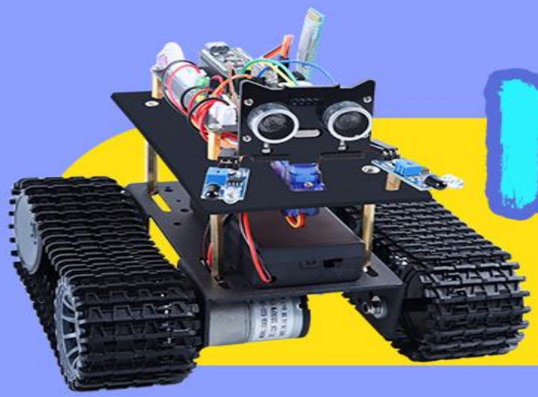
succeed



```
Done Saving.

avrdude done.  Thank you.
```



智懿
ZHIYI



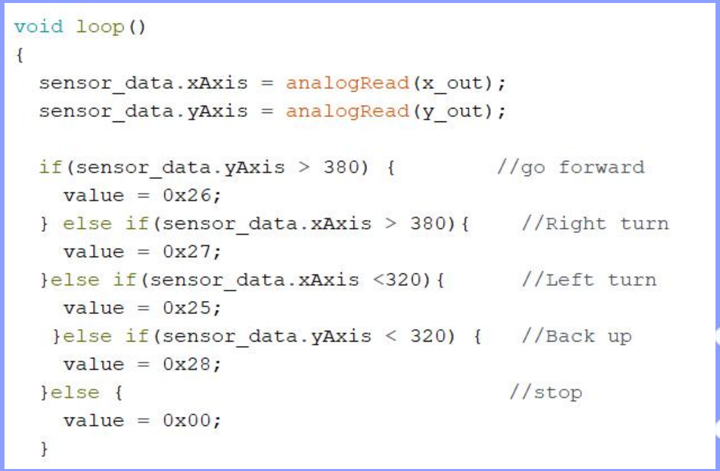
When Nano detects that the Y-axis Angle of ADXL335 sensor module is greater than 380, RF24L01 module is used to send data 0x26.

When Nano detects that the X axis Angle of ADXL335 sensor module is greater than 380, RF24L01 module is used to send data 0x25.

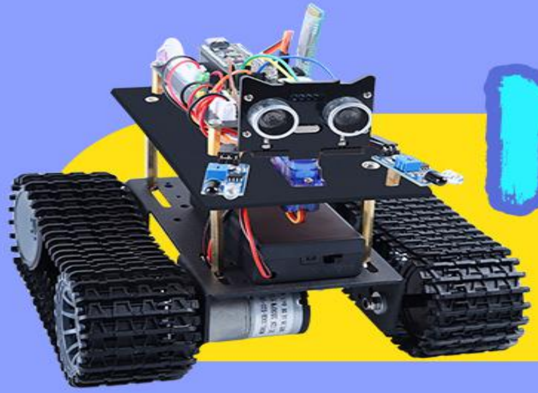
When Nano detects that the X axis Angle of ADXL335 sensor module is less than 320, RF24L01 module is used to send data 0x27.

When Nano detects that the Y axis Angle of ADXL335 sensor module is less than 320, RF24L01 module is used to send data 0x28.

Otherwise, data 0x00 is sent.



```
void loop()
{
  sensor_data.xAxis = analogRead(x_out);
  sensor_data.yAxis = analogRead(y_out);

  if(sensor_data.yAxis > 380) {         //go forward
    value = 0x26;
  } else if(sensor_data.xAxis > 380){     //Right turn
    value = 0x27;
  }else if(sensor_data.xAxis <320){       //Left turn
    value = 0x25;
   }else if(sensor_data.yAxis < 320) {    //Back up
    value = 0x28;
  }else {                                //stop
    value = 0x00;
  }
```



Thanks for watching!