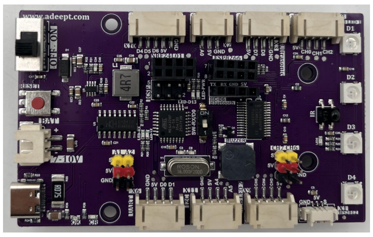
Lesson 16 Introduction to the Infrared Module

16.1 Overview

This course mainly introduces the relevant knowledge of infrared modules, including their working principles, hardware connections, and programming applications on the Arduino platform. Be able to master the working mechanism of the transmitting and receiving parts in infrared remote control systems, learn to use infrared modules to achieve remote control functions of equipment, and lay the foundation for developing various infrared control projects.

16.2 Required Components

Components	Quantity	Picture
Adeept Robot Control Board	1	
Type-C USB Cable	1	
IR remote control	1	

16.3 Principle Introduction

Infrared control is generally divided into two parts, namely the transmitting part and the receiving part, the IR receiver belongs to the receiving part, and the IR remote control belongs to the transmitting part.

There are two important components inside the IR receiver, namely IC and PD. IC is the processing element of the receiving head. It is mainly composed of silicon crystals and circuits. It is a highly integrated device. Its main functions include filtering, shaping, decoding, and amplification. PD is a photodiode, and its main function is to receive optical signals.

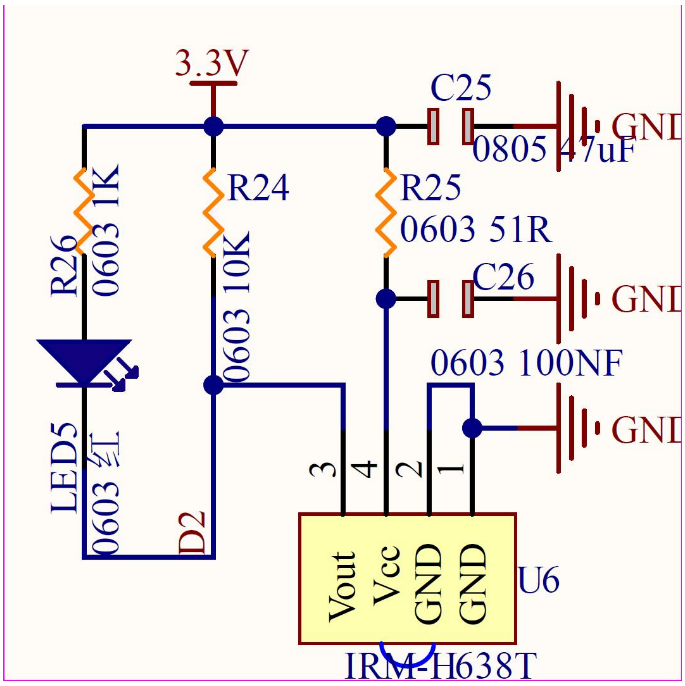
The transmitting circuit of the IR remote control uses infrared light-emitting diodes to emit modulated infrared light waves. Press a certain key of the remote control, the remote control will send out a series of modulated signals. After the signal is received by the infrared integrated module, it outputs the demodulated digital pulse. Each key corresponds to a different pulse, so it can identify different pulses. pulses to identify different keys.

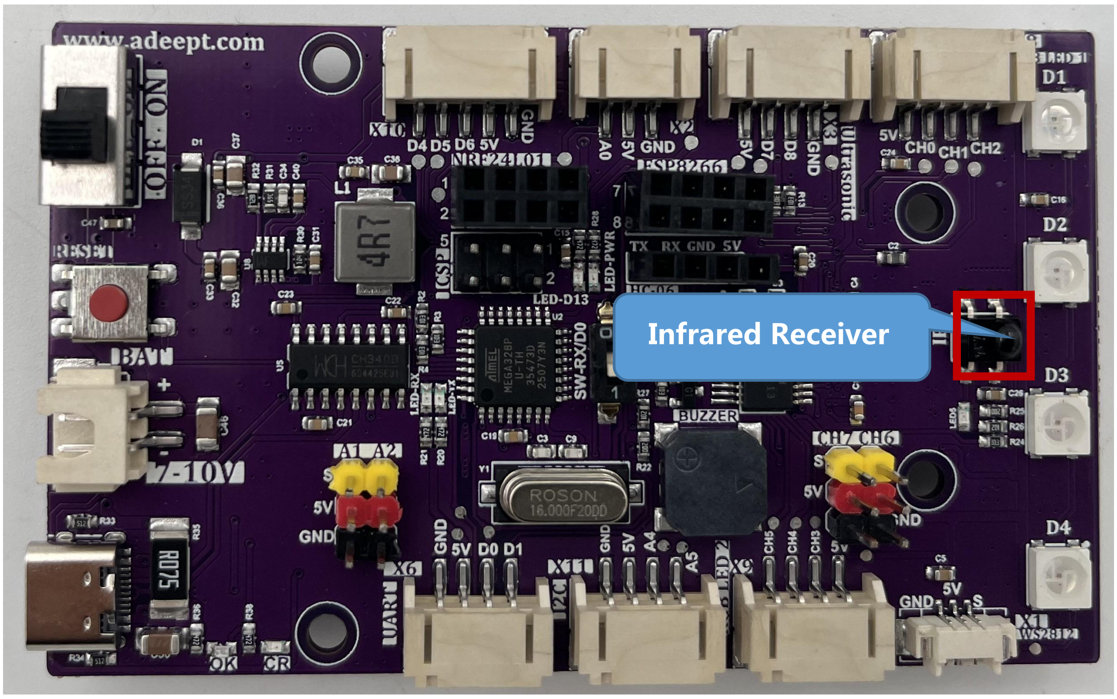
The following is the information in the corresponding program when each button of the infrared remote control is pressed.

Remote control button	Program value
0~9	0~9
*	10
#	11
UP	12
Down	13
Left	14
Right	15

OK	16

16.4 Wiring Diagram

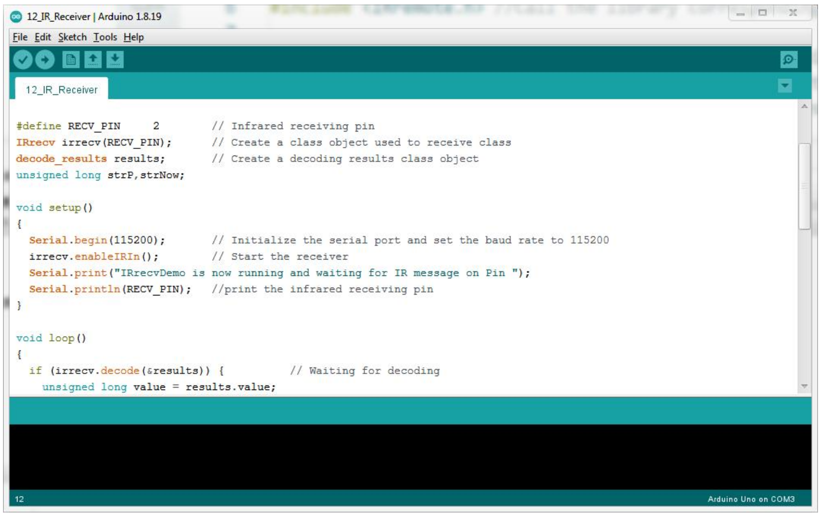



Infrared Receiver

16.5 Demonstration

1. Connect your computer and Adeept Robot Control Board (Arduino Board) with a USB cable.

2. Open "**12_IR_Receiver**" folder in "/Code", double-click "12_IR_Receiver.ino".



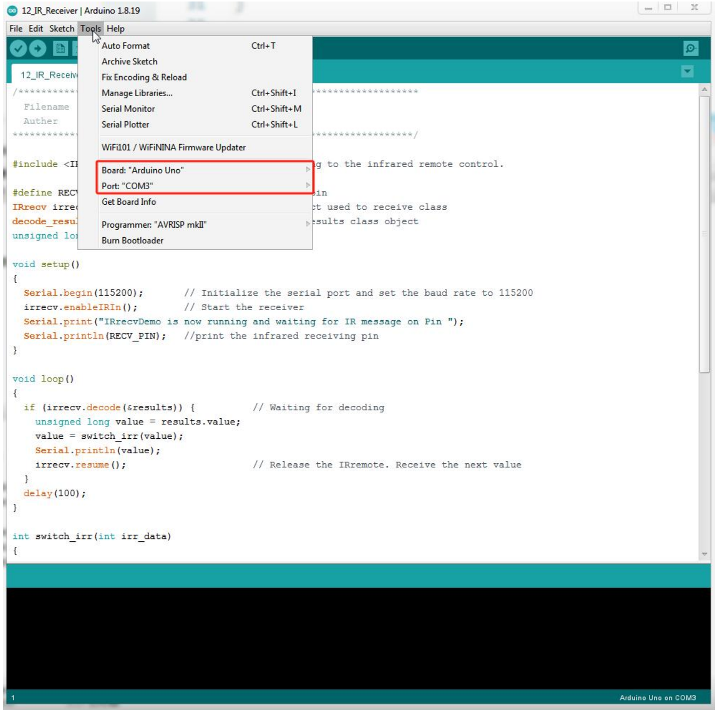
```
#define RECV_PIN     2        // Infrared receiving pin
IRrecv irrecv(RECV_PIN);      // Create a class object used to receive class
decode_results results;       // Create a decoding results class object
unsigned long strP,strNow;

void setup()
{
  Serial.begin(115200);       // Initialize the serial port and set the baud rate to 115200
  irrecv.enableIRIn();        // Start the receiver
  Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
  Serial.println(RECV_PIN);   //print the infrared receiving pin
}

void loop()
{
  if (irrecv.decode(&results)) {          // Waiting for decoding
    unsigned long value = results.value;
```

3. Select development board and serial port.

Board: Tools--->Board--->Arduino AVR Boards--->Arduino Uno

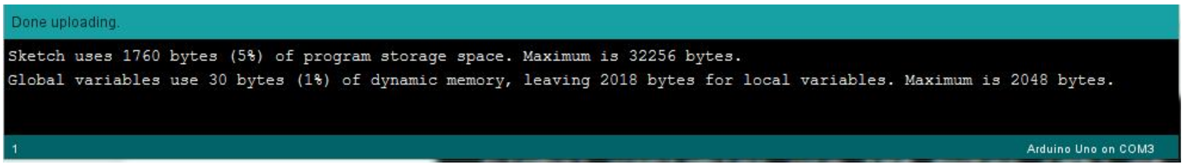
Port: Tools --->Port--->COMx

Note: The port number will be different in different computers.


12_IR_Receiver | Arduino 1.8.19
File Edit Sketch Tools Help
12_IR_Receiv
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
Board: "Arduino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
1
Arduino Uno on COM3

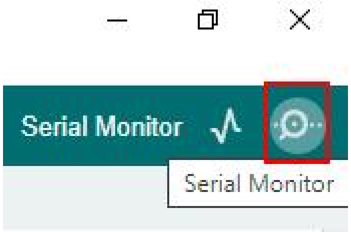
```
/***********                     *******************
  Filename
  Auther
***********                     ******************/

#include <II                      g to the infrared remote control.

#define RECV                      in
IRrecv irre                       t used to receive class
decode_resul                      sults class object
unsigned lo

void setup()
{
  Serial.begin(115200);       // Initialize the serial port and set the baud rate to 115200
  irrecv.enableIRIn();        // Start the receiver
  Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
  Serial.println(RECV_PIN);   //print the infrared receiving pin
}

void loop()
{
  if (irrecv.decode(&results)) {          // Waiting for decoding
    unsigned long value = results.value;
    value = switch_irr(value);
    Serial.println(value);
    irrecv.resume();                      // Release the IRremote. Receive the next value
  }
  delay(100);
}

int switch_irr(int irr_data)
{
```

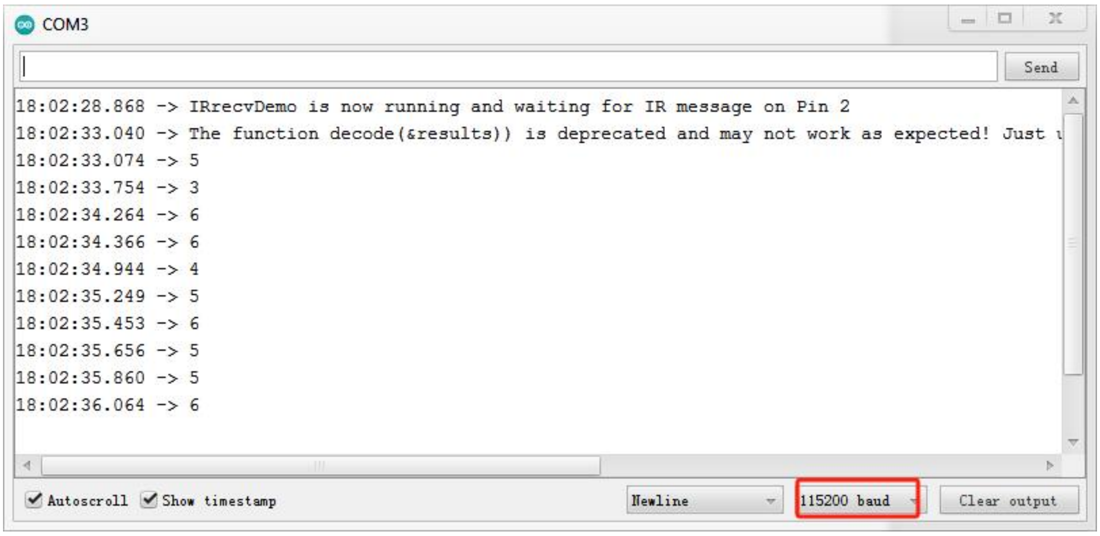
4. After opening, click → to upload the code program to the Arduino. If there is no error warning in the console below, it means that the Upload is successful.



```
Done uploading.
Sketch uses 1760 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 30 bytes (1%) of dynamic memory, leaving 2018 bytes for local variables. Maximum is 2048 bytes.
```

5. Click Serial Monitor, Set the baud rate as 115200. Use the IR remote to aim at the IR receiver on the expansion board. Press a button, and the corresponding button information can be displayed on the screen.





```
18:02:28.868 -> IRrecvDemo is now running and waiting for IR message on Pin 2
18:02:33.040 -> The function decode(&results)) is deprecated and may not work as expected! Just u
18:02:33.074 -> 5
18:02:33.754 -> 3
18:02:34.264 -> 6
18:02:34.366 -> 6
18:02:34.944 -> 4
18:02:35.249 -> 5
18:02:35.453 -> 6
18:02:35.656 -> 5
18:02:35.860 -> 5
18:02:36.064 -> 6
```

☑ Autoscroll ☑ Show timestamp Newline 115200 baud Clear output

16.6 Code

Complete code refer to 12_IR_Receiver.ino

```
/***********************************************************************
  Filename     : 12_IR_Receiver.ino
  Auther       : www.adeept.com
**********************************************************************/

#include <IRremote.h> //Call the library corresponding to the infrared remote control.

```

```
#define RECV_PIN     2        // Infrared receiving pin
IRrecv irrecv(RECV_PIN);      // Create a class object used to receive class
decode_results results;       // Create a decoding results class object
unsigned long strP,strNow;

void setup()
{
  Serial.begin(115200);       // Initialize the serial port and set the baud rate to 115200
  irrecv.enableIRIn();        // Start the receiver
  Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
  Serial.println(RECV_PIN);   //print the infrared receiving pin
}

void loop()
{
  if (irrecv.decode(&results)) {          // Waiting for decoding
    unsigned long value = results.value;
    value = switch_irr(value);
    Serial.println(value);
    irrecv.resume();                      // Release the IRremote. Receive the next value
  }
  delay(100);
}

int switch_irr(int irr_data)
{
  switch(irr_data)
  {
  case 16750695: return 0;
  case 16753245: return 1;
  case 16736925: return 2;
  case 16769565: return 3;
  case 16720605: return 4;
  case 16712445: return 5;
  case 16761405: return 6;
  case 16769055: return 7;
  case 16754775: return 8;
  case 16748655: return 9;
  case 16738455: return 10; // *
  case 16756815: return 11; // #
  case 16718055: return 12; // up
  case 16730805: return 13; // down
  case 16716015: return 14; // left
  case 16734885: return 15; // right
  case 16726215: return 16; // ok
  }
}

```

Code explanation

Initialization Stage:

In the setup function, first initialize the serial communication and set the baud rate to 115200 in order to view the infrared decoding results in the serial monitor.

Next, activate the infrared receiver to prepare it for receiving infrared signals.

Finally, print the pin information waiting to receive the infrared signal through the serial port to facilitate the confirmation of hardware connection.

Loop Control Process:

By waiting for the infrared signal to decode. When a valid infrared signal is received and decoded successfully, perform the following operations.

Stage 1: Retrieve key values from decoding results.

Stage 2: Call the switch_irr() function to convert key values and print the corresponding key values through the serial port.

Stage 3: Call irrecv.resume() to release the infrared receiver, allowing it to continue receiving the next infrared signal.

Stage 4: By delaying(100); Delay the program by 100ms to avoid excessive data processing and ensure stable system operation.