Course 19---IR control

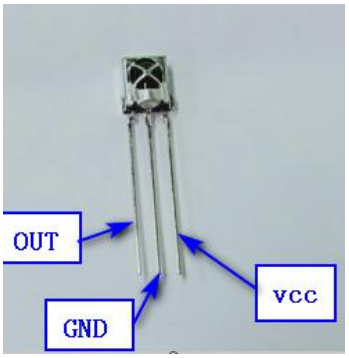
The purpose of the experiment:

In this experiment, we will make the IR remote controller communicate with the IR receiver sensor .

About the infrared remote control:

The signal from the IR remote controller is a series of binary pulse codes. In order to protect it from other infrared signals during wireless transmission. It is modulated on a specific carrier frequency ,and then transmitted by infrared emission sensor. The infrared receiving device need to filter out other waveform and receive the signal of the specific frequency and restore it to binary pulse code, this process is called demodulation.

The IR receiver sensor converts the optical signal emitted by the infrared emission sensor to a weak electrical signal. These signals are restored to the original encode by various circuits, finally outputs the signal to the control circuit.

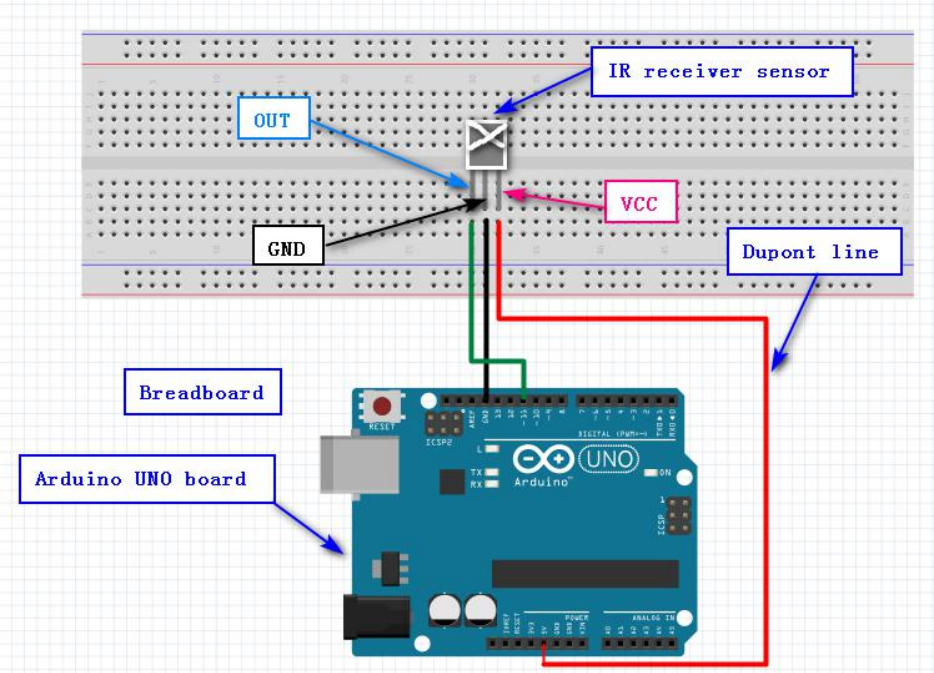


List of components required for the experiment:

Arduino UNO board *1
USB cable *1
IR receiver sensor *1
IR remote controller *1
Breadboard *1
Dupont line *1 bunch

Actual object connection diagram：

We need to connect the circuit as shown in the figure below.



Experimental code analysis:

```
#include <IRremote.h>//Including infrared library
int RECV_PIN = 11; // Declarations of port
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int LED6 = 7;
long on1    = 0x00FF6897;//Code the example to match the send
long off1 = 0x00ff30CF;
long on2 = 0x00FF9867;
long off2 = 0x00FF18E7;
long on3 = 0x00FFB04F;
long off3 = 0x00FF7A85;
long on4 = 0x00FF10EF;
long off4 = 0x00FF42BD;
long on5 = 0x00FF38C7;
long off5 = 0x00FF4AB5;
long on6 = 0x00FF5AA5;
long off6 = 0x00FF52AD;
IRrecv irrecv(RECV_PIN);
decode_results results;//Declarations of struct
// Dumps out the decode_results structure.
// Call this after IRrecv::decode()
```

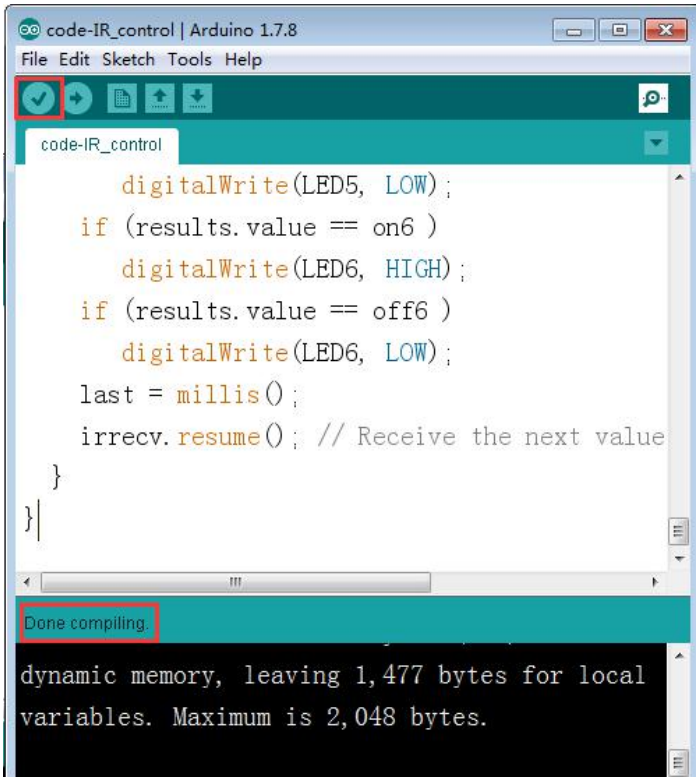
```
// void * to work around compiler issue
//void dump(void *v) {
//    decode_results *results = (decode_results *)v
void dump(decode_results *results)
{
      int count = results->rawlen;
      if (results->decode_type == UNKNOWN)
      {
            Serial.println("Could not decode message");
      }
      else
      {
            if (results->decode_type == NEC)
            {
                  Serial.print("Decoded NEC: ");
            }
            else if (results->decode_type == SONY)
                        {
                              Serial.print("Decoded SONY: ");
                        }
                        else if (results->decode_type == RC5)
                                    {
                                          Serial.print("Decoded RC5: ");
                                    }
                                    else if (results->decode_type == RC6)
                                                {
                                                      Serial.print("Decoded RC6: ");
                                                }
            Serial.print(results->value, HEX);
            Serial.print(" (");
            Serial.print(results->bits, DEC);
            Serial.println(" bits)");
      }
            Serial.print("Raw (");
            Serial.print(count, DEC);
            Serial.print("): ");

            for (int i = 0; i < count; i++)
              {
                if ((i % 2) == 1)
                {
                      Serial.print(results->rawbuf[i]*USECPERTICK, DEC);
                }
                else
```

```
            {
                Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC);
            }
            Serial.print(" ");
        }
        Serial.println("");
}

void setup()
{
    pinMode(RECV_PIN, INPUT);    //Defining the RECV port for the input port
    pinMode(LED1, OUTPUT);//Defining the LED1 port for the output port
    pinMode(LED2, OUTPUT);//Defining the LED2 port for the output port
    pinMode(LED3, OUTPUT);//Defining the LED3 port for the output port
    pinMode(LED4, OUTPUT);//Defining the LED4 port for the output port
    pinMode(LED5, OUTPUT);//Defining the LED5 port for the output port
    pinMode(LED6, OUTPUT);//Defining the LED6 port for the output port
    pinMode(13, OUTPUT);//Defining the port13 for the output port
    Serial.begin(9600); //The baud rate is 9600
    irrecv.enableIRIn(); // Start the receiver
}

int on = 0;
unsigned long last = millis();

void loop()
{
  if (irrecv.decode(&results)) //Calling the library function: decode
    {
     // If it's been at least 1/4 second since the last
     // IR received, toggle the relay
     if (millis() - last > 250)
        {
         on = !on;
         digitalWrite(13, on ? HIGH : LOW);
         dump(&results);
        }
     if (results.value == on1 )
         digitalWrite(LED1, HIGH);
     if (results.value == off1 )
         digitalWrite(LED1, LOW);
     if (results.value == on2 )
         digitalWrite(LED2, HIGH);
     if (results.value == off2 )
```

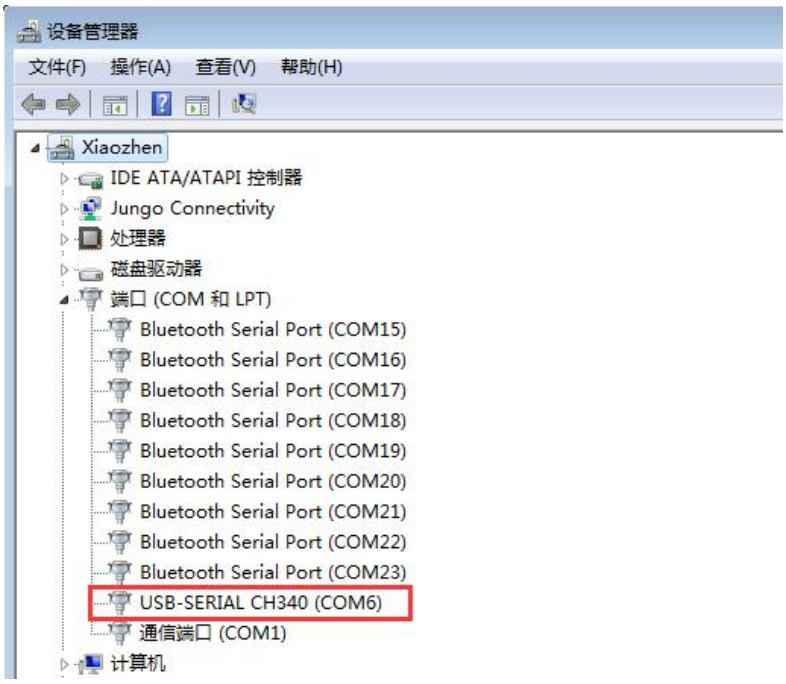
```
        digitalWrite(LED2, LOW);
    if (results.value == on3 )
        digitalWrite(LED3, HIGH);
    if (results.value == off3 )
        digitalWrite(LED3, LOW);
    if (results.value == on4 )
        digitalWrite(LED4, HIGH);
    if (results.value == off4 )
        digitalWrite(LED4, LOW);
    if (results.value == on5 )
        digitalWrite(LED5, HIGH);
    if (results.value == off5 )
        digitalWrite(LED5, LOW);
    if (results.value == on6 )
        digitalWrite(LED6, HIGH);
    if (results.value == off6 )
        digitalWrite(LED6, LOW);
    last = millis();
    irrecv.resume(); // Receive the next value
  }
}
```

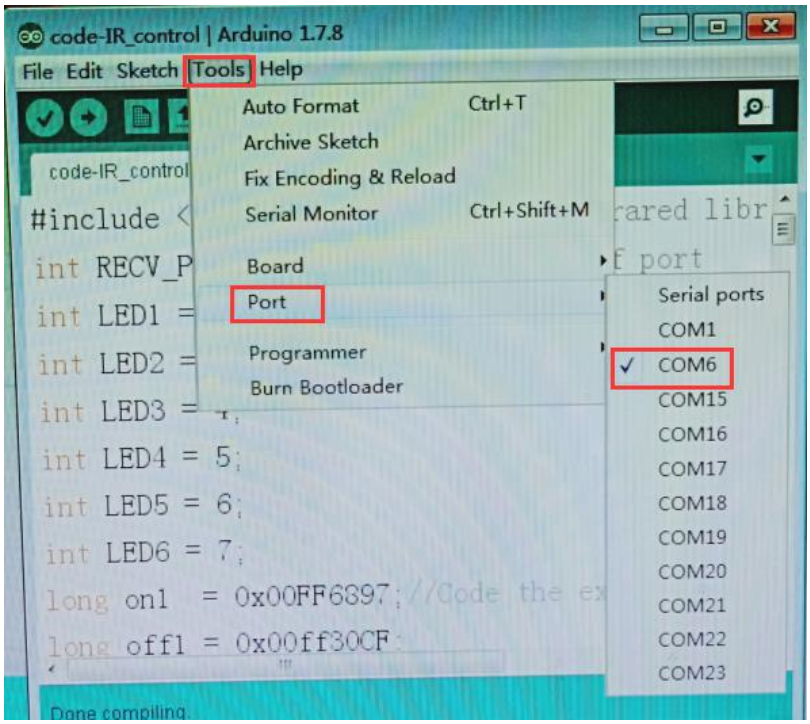
Experimental steps:

1.You need to open the code for this experiment: **code-IR_control.ino**, click "√"under the menu bar, compile the code, and wait for the words of **Done compiling** in the lower left corner, as shown in the following figure.

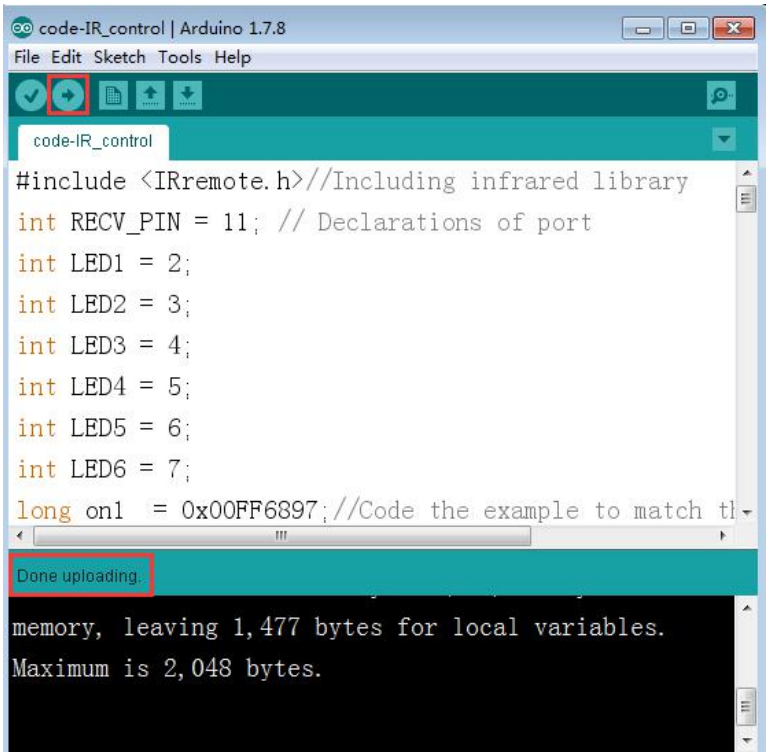


2. In the menu bar of Arduino IDE，select the 【Tools】---【Port】--- select the port that the serial number displayed by the device manager just now.for example:COM6,as shown in the following figure.

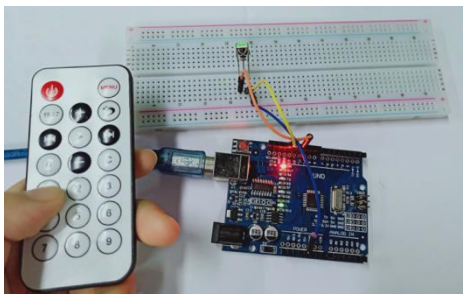


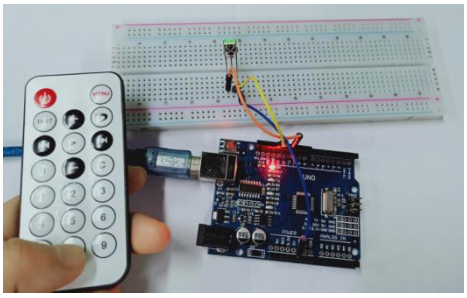


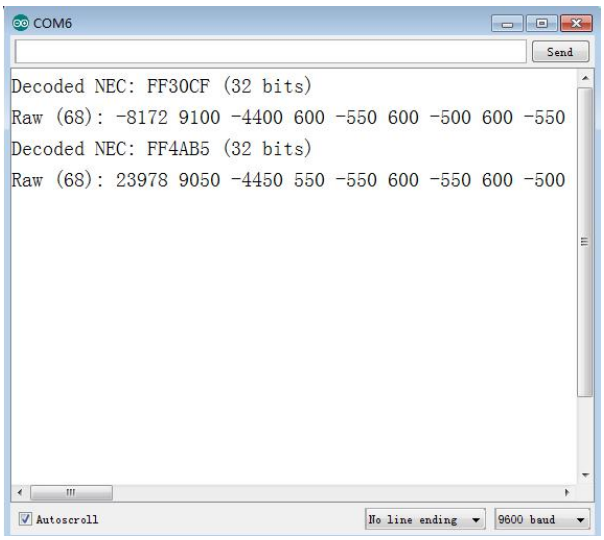
3. After the selection is completed, you need to click "→"under the menu bar,and upload the code to the Arduino UNO board, when appears to **Done uploading** on the lower left corner , that means that the code has been successfully uploaded to the Arduino UNO board, as shown in the following figure.



4. After the code is uploaded, we need to open the serial monitor of Arduino IDE, and set the baud rate to 9600. When we press the button on the infrared remote controller, we can see the code value of the corresponding button on the serial monitor, as shown below (Just for example).







The code of the experiment: