1. Infrared receiving head introduction

一： what is infrared reception

Infrared remote control signals a series of binary pulse codes. In order to make it in the process of wireless transmission get rid of other infrared signal interference, is usually the first modulation on specific carrier frequency, and then by the infrared emission diode launch out, while the infrared receiving device to filter out other clutter, receives the specific frequency signal and to restore it into binary pulse code, namely demodulation.

二： Working principle

Built-in receiving tube will be emitted infrared transmitting tube light signal is converted to a weak electrical signals, the signal through the internal amplifier amplification IC, and then through the automatic gain control, band-pass filtering, demodulation, waveform shaping after reduction for the remote control launch out of the original code, through the output pin head of reception coding identification on the input to the electrical circuit.

三： The lead and attachment of infrared reception head

The infrared receiver has three pins as shown:
When using the VOUT, the VOUT received the GND on the test board, and VCC received the +5v on the test board.

**Infrared remote control experiment**

1. **Experimental devices**
   - Infrared remote control: 1
   - Infrared receiving head: 1
   - LED lights: 6
   - 220 Ω resistance: six
   - Colorful bread line: some

2. **Experimental connection**
   First connect the boards; Then the infrared receiver is connected with the above method, and the VOUT is received by the digit 11 pin, and the LED lamp is connected to the digital pin through the resistance. The pin is 2, 3, 4, 5, 6, 7. The return sample completes the connection of the circuit part.

3. **Experimental principle**
   To decode a remote, you have to understand the way the remote is encoded. The code of the control device used for this product is: NEC agreement. Here’s the NEC protocol:
   - **NEC protocol:** features: (1) 8-bit address and 8-bit command position
   - (2) the location of the reliability address and the command bit are transferred twice
   - (3) pulse position modulation
   - (4) carrier frequency 38khz
   - (5) each time of 1.125 ms 2.25 ms
   - • the definition of logic 0 and 1 is as follows

   **The agreement is as follows:**

   ![NEC protocol pulse sequence]

   • press button to release the firing pulse immediately:

   ![NEC protocol address transmission]

   The image above shows the typical pulse sequence of the NEC protocol. Note: this first sends the LSB (the lowest) protocol. In the above pulse transmission address 0 0 x16 x59
command. A message is started by a 9ms high level, followed by a 4.5ms low level, (the two levels are made up of a seeker code) and then an address code and a command code. The address and command are transferred twice. The second time all the bits are reversed and can be used to confirm the use of the received messages. The total transmission time is constant, because each point is repeated with it in reverse length. If you're not interested, you can ignore the reliability and expand the address and command to every 16 bits!

Press the button to release the pulse for a period of time:

```
-110 ms
```

A command is sent once, even if the button on the remote control is still pressed. When the button has been pressed, the first 110 ms pulse like above, after every 110 ms to repeat the code transmission. Return a duplicate code is written by a high level of ms pulse and a 2.25 ms low level and over 560 ms high level.

* repeat pulse

```
110 ms
```

Note: after receiving head, pulse waveform into the integration for the integration of receiving head to Wu decoding, signal amplification and plastic, so should pay attention to: in the absence of the infrared signal, its output is high level, a signal for the low level, so the output signal level the reverse of transmitter. The receiving end pulse can be seen through the oscilloscope and the waveform comprehension program seen in combination.

**Key value of remote control:**

A line of one = 0x0FFA25D; A line of two = 0x0FF629D; A line of three = 0x0FFE21D;
Two lines of one = 0x0FF22DD; Two lines of two= 0x0FF02FD; Two lines of three= 0x0FFC23D;
Three lines of one = 0x0FFE01F; Three lines of two = 0x0FFA857; Three lines of three = 0x0FF906F;
Four lines of one = 0x0FF6897; Four lines of two = 0x0FF9867; Four lines of three = 0x0FFB04F;

**Connection mode:**
LED wiring:

Note: all leds have negative 1K resistance to the ground

Infrared reception module

<table>
<thead>
<tr>
<th>LED</th>
<th>arduino</th>
</tr>
</thead>
<tbody>
<tr>
<td>Led1</td>
<td>D2</td>
</tr>
<tr>
<td>Led2</td>
<td>D3</td>
</tr>
<tr>
<td>Led3</td>
<td>D4</td>
</tr>
<tr>
<td>Led4</td>
<td>D5</td>
</tr>
<tr>
<td>Led5</td>
<td>D6</td>
</tr>
<tr>
<td>Led6</td>
<td>D7</td>
</tr>
</tbody>
</table>

| VCC  | 5V      |
| GND  | GND     |
| VOUT | D11     |
# The program code
/* This is our website www.weikedz.com
For bulk orders, please feel free to contact
sophie@weikedz.com. If any question, for orders,
for technical problems, pls contact us.
We will response you fastest time. */
#include <IRremote.h>
#include <IRremote.h>
int RECV_PIN = 11;
int LED1 = 2;
int LED2 = 3;
int LED3 = 4;
int LED4 = 5;
int LED5 = 6;
int LED6 = 7;
long on1 = 0x00FFA25D;
long off1 = 0x00FFE01F;
long on2 = 0x00FF629D;
long off2 = 0x00FFA857;
long on3 = 0x00FFE21D;
long off3 = 0x00FF906F;
long on4 = 0x00FF22DD;
long off4 = 0x00FF6897;
long on5 = 0x00FF02FD;
long off5 = 0x00FF9867;
long on6 = 0x00FFC23D;
long off6 = 0x00FFB04F;
IRrecv irrecv(RECV_PIN);
declare_results results;
// 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.println(results->bits, DEC);
Serial.println(" bits");

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);
    pinMode(LED1, OUTPUT);
    pinMode(LED2, OUTPUT);
    pinMode(LED3, OUTPUT);
    pinMode(LED4, OUTPUT);
    pinMode(LED5, OUTPUT);
}
pinMode(LED6, OUTPUT);
pinMode(13, OUTPUT);
Serial.begin(9600);

irrecv.enableIRIn(); // Start the receiver
}

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

void loop()
{
  if (irrecv.decode(&results))
  {
    // If it's been at least 1/4 second since the last
    // IR received, toggle the relay
    if (millis() - last > 250)
    {
      on = !on;
      //       digitalWrite(8, on ? HIGH : LOW);
      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