Infrared Remote Control

What is a infrared receiving header?

We have introduced infrared receiving tube in former passage, it is a kind of sensor which can identify the infrared. The integration of infrared sensor receives and modulates the 38kHZ infrared. In order to avoid in the process of wireless transmission from other infrared signal interference, the infrared remote control usually modulate the signal on a specific carrier frequency, and then launch out by the infrared emission diode. While the infrared receiving device needs to filter out other clutter, receives the specific frequency signal and restores it into binary pulse code, namely, demodulation.

Working Principle

The built-in receiving tube converts the light signal emitted by infrared transmitting tube to weak electrical signal, the signal is amplified through the internal IC, then restored to original code emitted by the infrared remote control through the automatic gain control, band-pass filtering, demodulation, waveform shaping, and input to the decoding circuit on electric appliance through the output pin of the receiving header

Infrared receiver header has three pins: connecting VOUT to analog interface, GND to the GND onboard and VCC to + 5 v.

Experiment Purpose

The aim is to ecode all the keys of the remote control by Ardunio and display them in a serial port.

Experiment Principle

The encoding of a remote control is required to be learned first if we want to decode it. The control code used in this product is: NEC protocol.

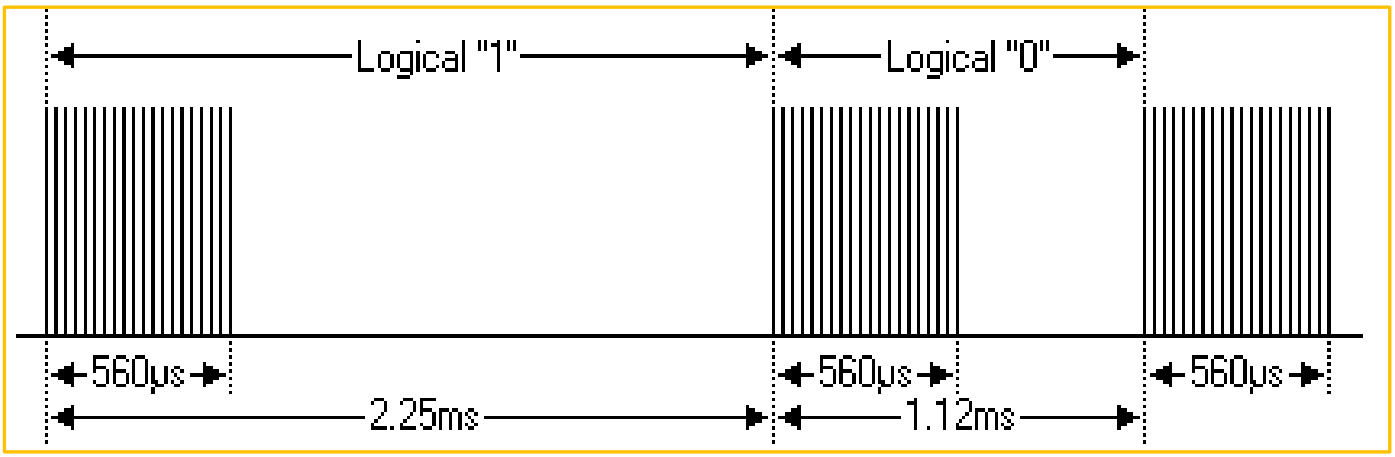
The following will introduce the NEC protocol:

Introduction of NEC

Characteristics

- 8 address bits, 8 command bits
- Address bits and command bits are transmitted twice in order to ensure reliability
- Pulse-position modulation
- Carrier frequency 38kHz
- Every bit lasts 1.125ms or 2.25ms

The definitions of logic 0 and 1 are as below

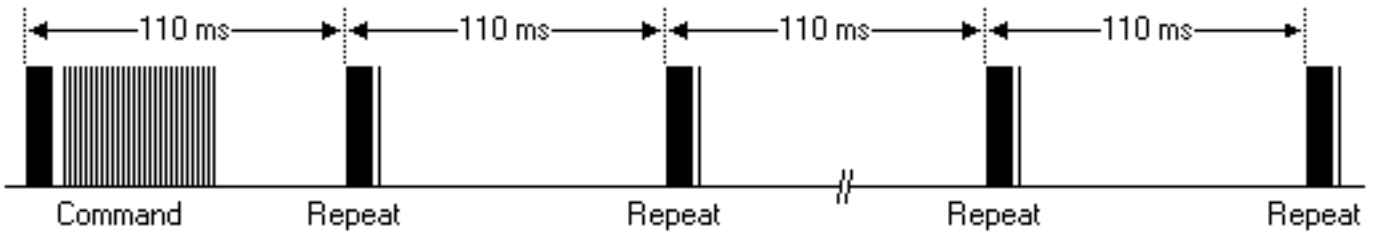


Transmitted pulse which the pressed button released immediately

The picture above shows a typical pulse sequence of the NEC protocol. Notice: The protocol of LSB (least significant) is firstly transmitted. In the above, pulse transmission address is 0x16 and the command is 0x59. A message starts from a 9ms high level, then followed by a 4.5ms low level, and by the address code and command code. The address and

command are transmitted twice. All bits flip in the second transmission, this can be used in the confirmation of the received message. The total transmission time is constant, because every bit repeats the flip length. If you are not interested, you can ignore this reliable inversion and expand the address and command every 16 bit as well !

Transmitted pulse which the pressed button last for a while



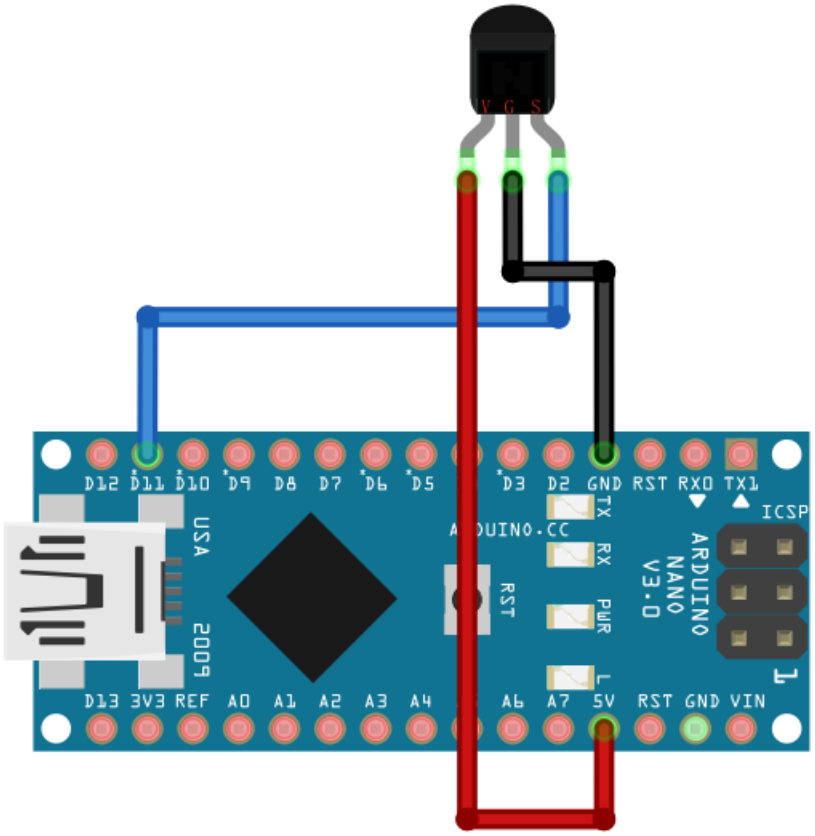
Even a button on the remote control is pressed again, the command is transmitted only once. When the button has been pressing, the first 110ms pulse is same as above, then the same code is transmitted every 110ms. The next repeated code consists of a 9ms high level pulse, a 2.25ms low level pulse and a 560μs high level pulse.

Notice: When the pulse received by the integrative header, the header needs to decode,amplify and shape the signal. So we should notice that the output is high level when the infrared signal is absent, otherwise, the output is low level, so the output signal level is reversed in transmitter. We can see the pulse of receiver through the oscilloscope and understand the program by waveform.

Component List

- Arduino Nano mainboard
- Breadboard
- USB cable
- NEC infrared remote control * 1
- The integrative infrared receiving header *1
- 1k Resistor *1
- Several jumper wires

Wiring of Circuit



Code

```
#include "IRremote.h"

IRremote ir(11);

unsigned char keycode;
char str[128];
void setup()
{
    Serial.begin(9600);
    ir.begin();
}

void loop()
{
    if (keycode = ir.getCode()) {
        String key_name = ir.getKeyMap(keycode);
        sprintf(str, "Get ir code: 0x%x key name: %s \n", keycode, (char
*)key_name.c_str());
        Serial.println(str);
    } else {
      //   Serial.println("no key");
    }
    delay(110);
}
```