Infrared Remote Control Experiment

Introduce to Infrared receiver tube

Infrared receiver tube, it is a kind of sensor that can identify infrared.Integrated reception and modulation of 38kHZ infrared sensor.In order to avoid the interference of other infrared signals in the wireless transmission process, the infrared remote control usually modulates the signal on a specific carrier frequency, and then it is sent out by the infrared emitting diode.When the infrared receiver needs to filter out other clutter, it receives a specific frequency signal and returns it to binary pulse code, that is, demodulation.

Working Principle

The built-in receiving tube converts the optical signal sent by the infrared transmitter tube into weak current signal, which is amplified by internal IC, and then restored to the original code sent by the infrared remote control through automatic gain control, bandpass filtering, demodulation, waveform shaping, and the decoding circuit input to the electrical appliance through the output pin of the receiving head

Experiment Purpose

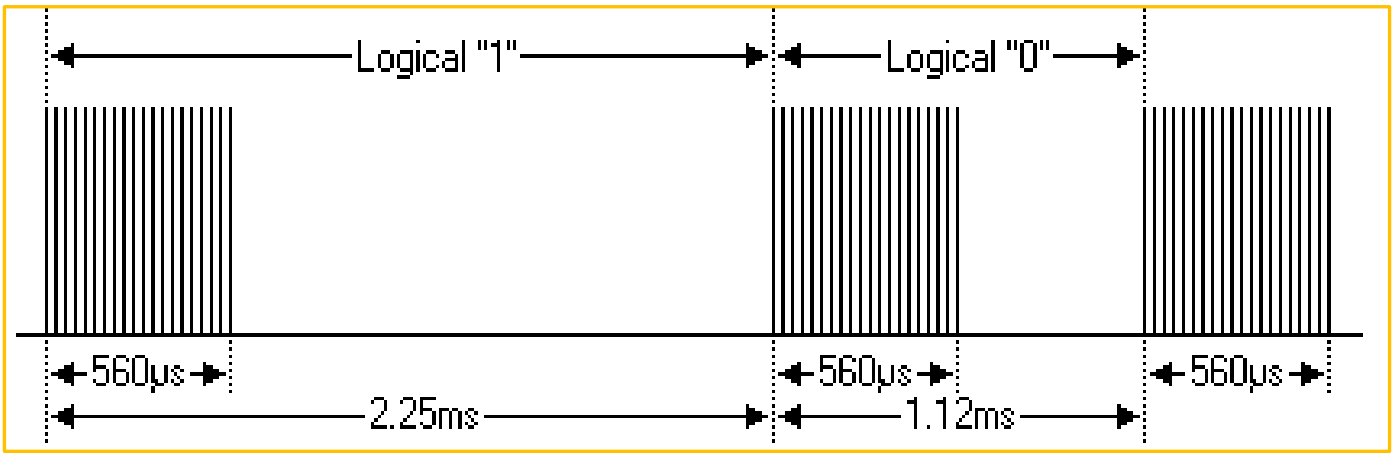
- The buttons of the remote control are coded through Ardunio
- Arduino UNO main control board communicates with infrared receiver. If the "<" button of the remote control is pressed, the fan turn left; if the ">" button is pressed, the fan turn right “ok” button is pressed, fan start run, when “ok” pressed again, fan stop.

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

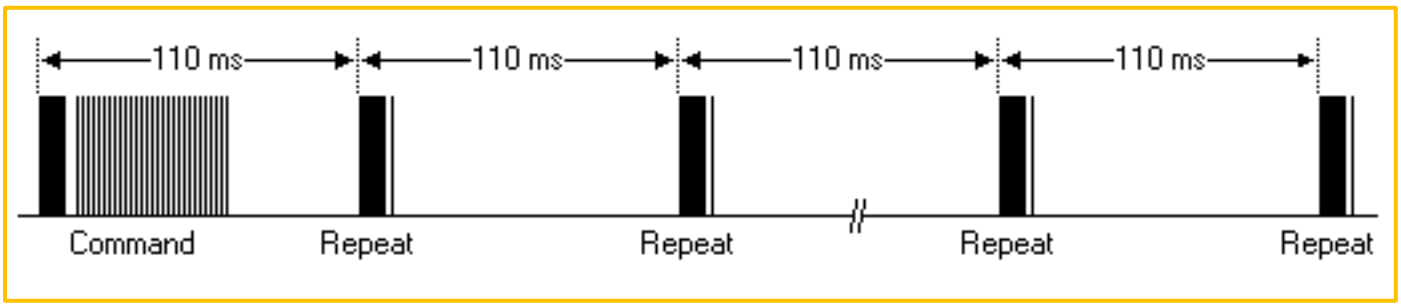
The definitions of logic 0 and 1 are as below



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.

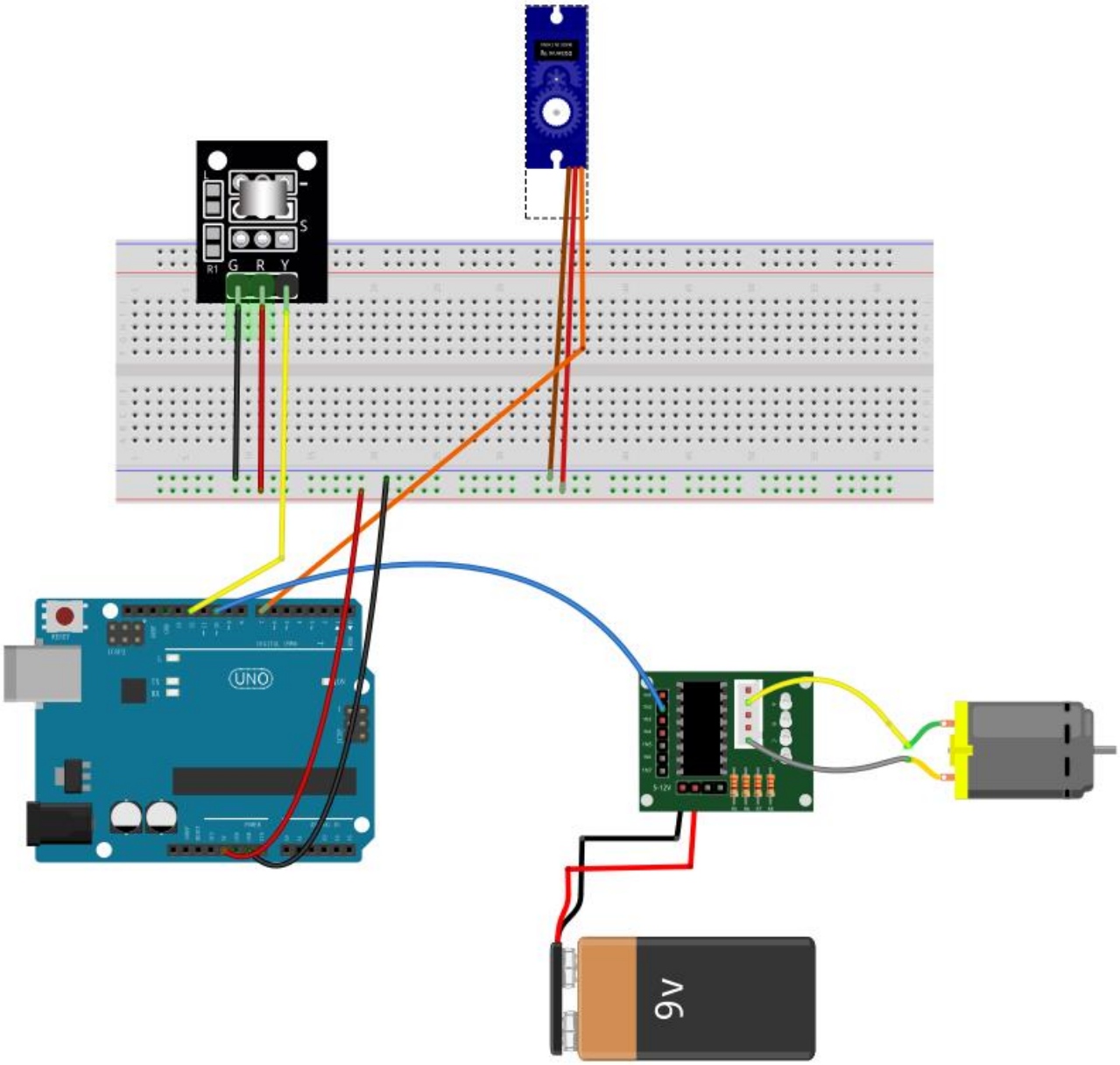
Component List

- Keywish Arduino UNO R3 motherboard
- Bread plate
- USB cable
- Infrared remote control * 1
- Integrated infrared receiving head module * 1
- Dc motor *1
- Fan * 1
- Steering gear * 1
- Motor drive board *1
- Motor bracket kit *1
- Battery * 1
- A number of jumpers

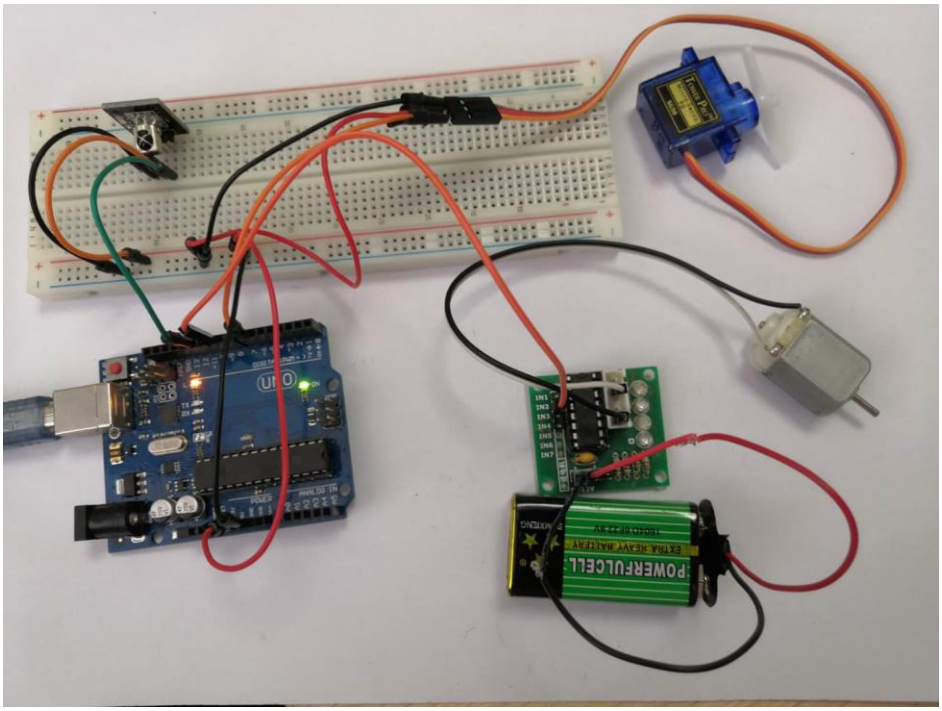
Wiring of Circuit

Motor drive board	Steering gear module
5V	VCC
GND	GNG
7	S
Arduino mianboard	Infrared remote control
5V	+
GND	-
12	S
Arduino mianboard	Motor drive board
5V	5V(+)
GND	GND(-)

10	IN4
Motor drive board	Dc motor
VCC	+
OUT4	-



Physical connection

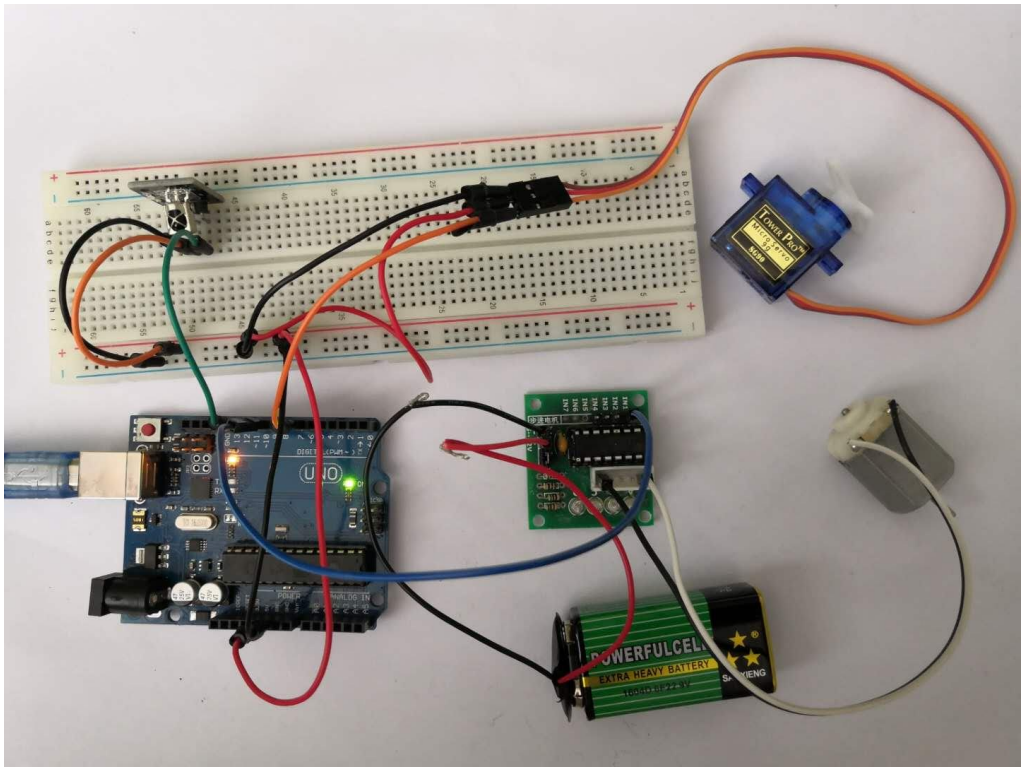


Code

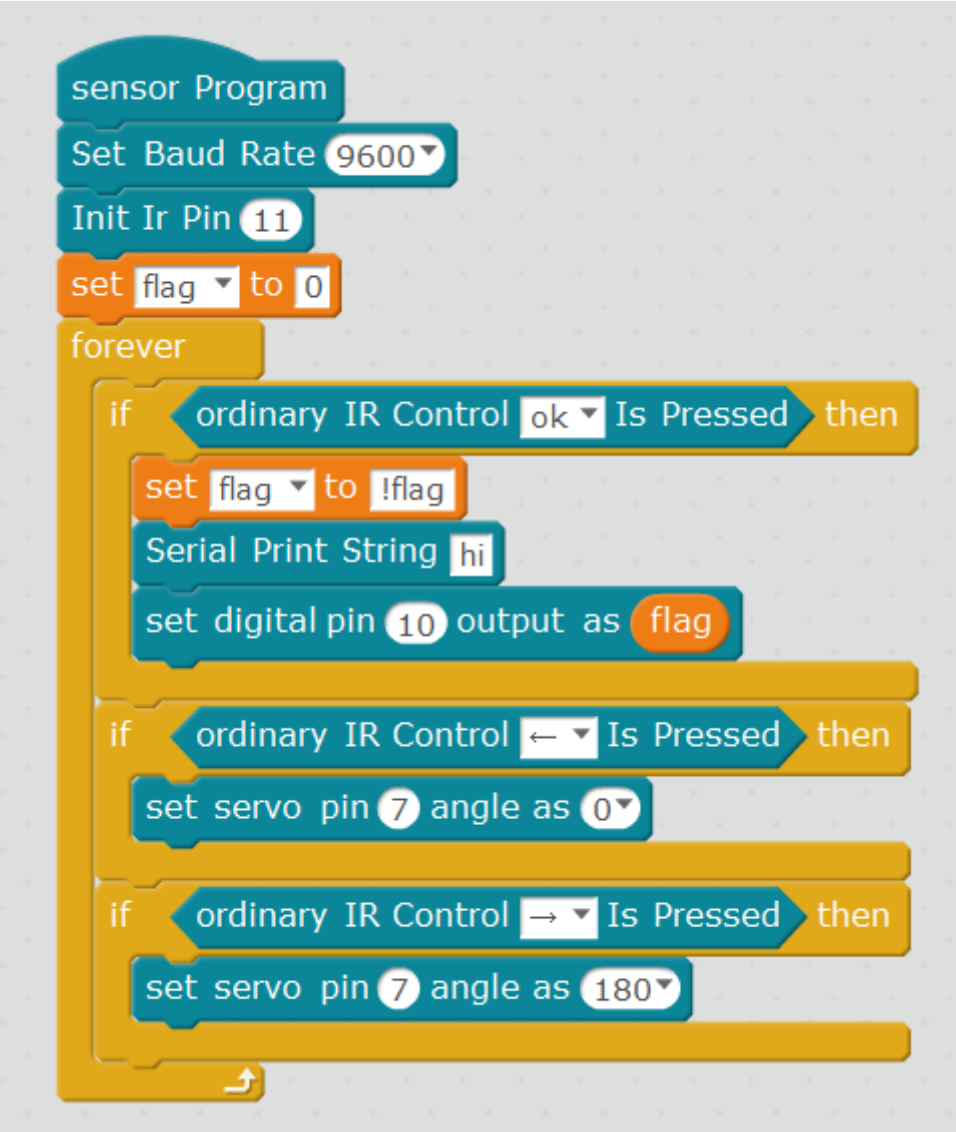
```
#include "IR_remote.h"
#include <Servo.h>
Servo myservo;
int servopin = 7;// Set the servo interface to 7
int MotorPin = 10;// Set the motor interface to 10
int flag = 0;// Set the flag

IRremote ir(12);// Set the infrared receiving interface to 12

unsigned char keycode;

void setup()
{
    Serial.begin(9600);// Set the serial port baud rate to 9600
    ir.begin();
    myservo.attach(7);
    myservo.write(90);
    delay(1000);
    pinMode(MotorPin, OUTPUT);
    digitalWrite(MotorPin, 0);
}
```

```
void loop()
{
   byte ir_key = ir.getCode();
   switch (ir.getIrKey(ir_key)) {// Determine which button is pressed and execute the
corresponding program
    case IR_KEYCODE_OK:
         Serial.println("IR_KEYCODE_OK key");
         flag = !flag;
         digitalWrite(MotorPin, flag);// Control the motor
         Serial.println(flag);
         break;
    case IR_KEYCODE_LEFT:
         myservo.write(0);// Control the steering gear to turn to 0 degree
         Serial.println("IR_KEYCODE_OK left");
         break;
    case IR_KEYCODE_RIGHT:
         myservo.write(180);//Control the steering gear to turn 180 degrees
         Serial.println("IR_KEYCODE_OK right");
         break;
   }
   delay(110);
}
```

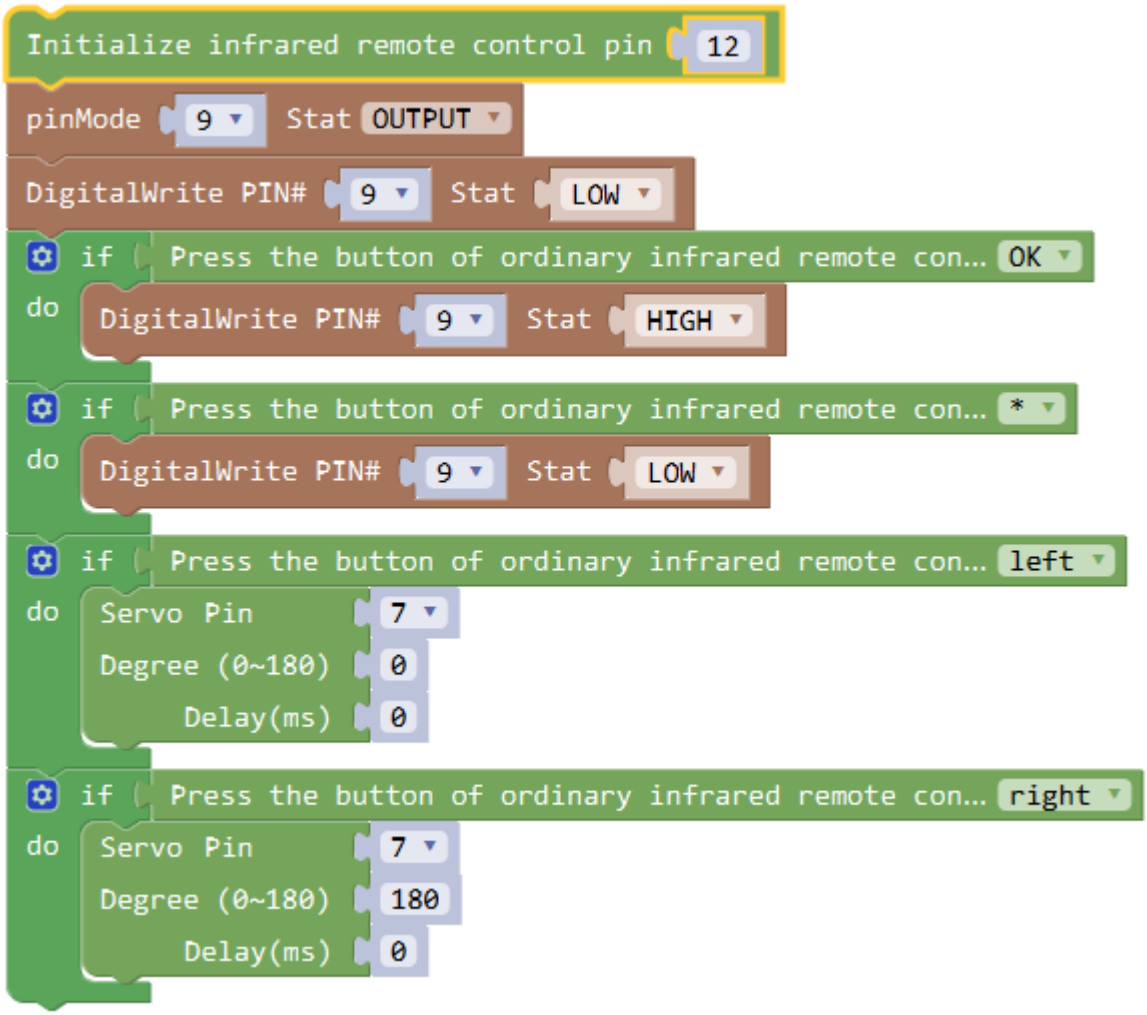
Result



Mblock programming program



Mixly programming program



MagicBlock programming program