Arduino Starter kit Users Manual

“ Infrared IR Remote LCD 1602 DIY Kit ”

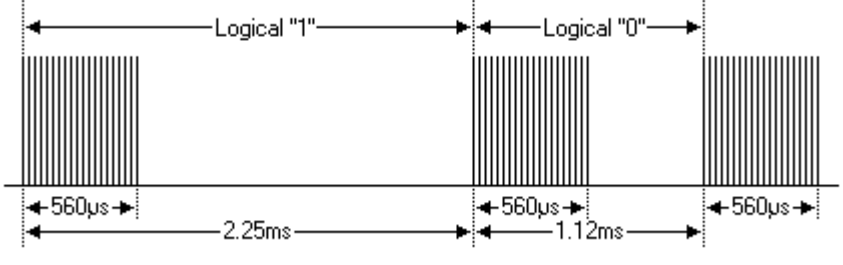
1. What is Infrared Receiver (IRM_3638) ?

To know more about how to control IR Remote Control. You should have a knowledge of the NEC Protocol

NEC Protocol Features :(http://www.sbprojects.com/knowledge/ir/nec.htm)

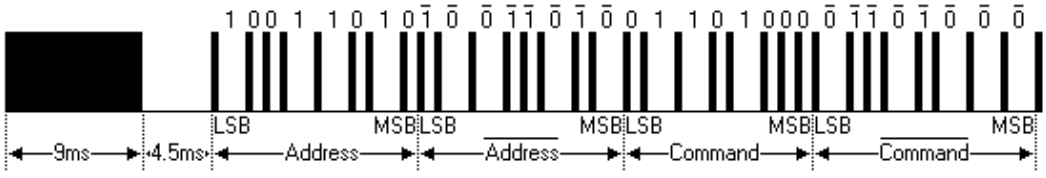
- 8 bit address and 8 bit command length
- Address and command are transmitted twice for reliability
- Pulse distance modulation
- Carrier frequency of 38kHz
- Bit time of 1.125ms or 2.25ms

Modulation:

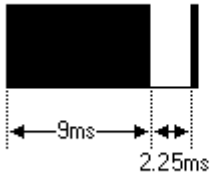


The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.

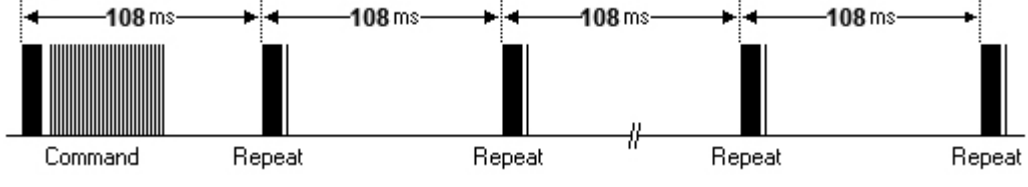
Protocol :



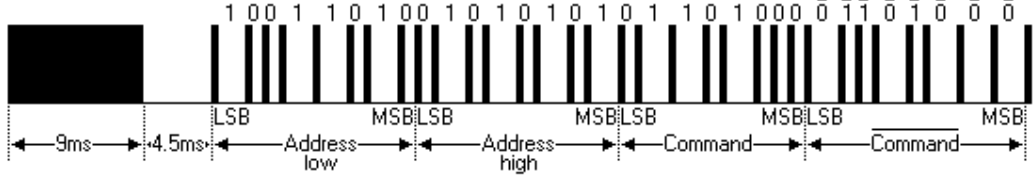
The picture above shows a pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address $59 and Command $16 is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed by a 4.5ms space, which is then followed by the Address and Command. Address and Command are transmitted twice. The second time all bits are inverted and can be used for verification of the received message. The total transmission time is constant because every bit is repeated with its inverted length. If you're not interested in this reliability you can ignore the inverted values, or you can expand the Address and Command to 16 bits each!



A command is transmitted only once, even when the key on the remote control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9ms AGC pulse followed by a 2.25ms space and a 560µs burst.



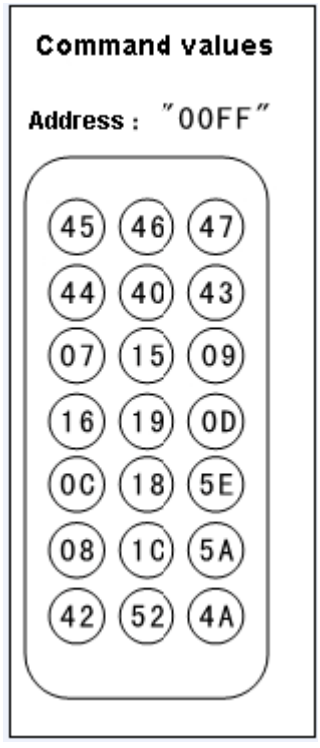
In this Experiment, we will use a little different IR Remote Control , which use WD6122 chip . (Extended NEC protocol)



Pay attention: **When there is no infrared signals, the receiver's output is High ; while, it receives an signal, its output is LOW . We can check the received pulse through the**

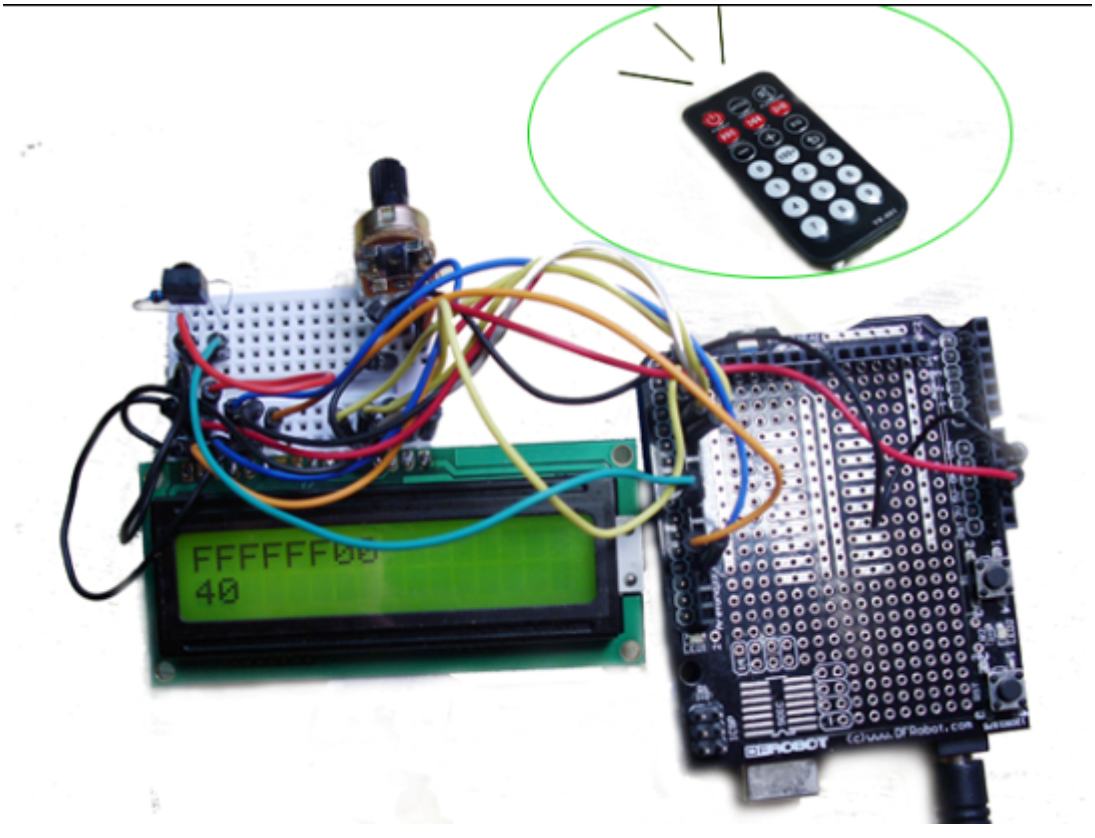
Oscilloscope, analyse the program according to the waveform.

The Command Values of the IR Remote Control's key like the following picture.

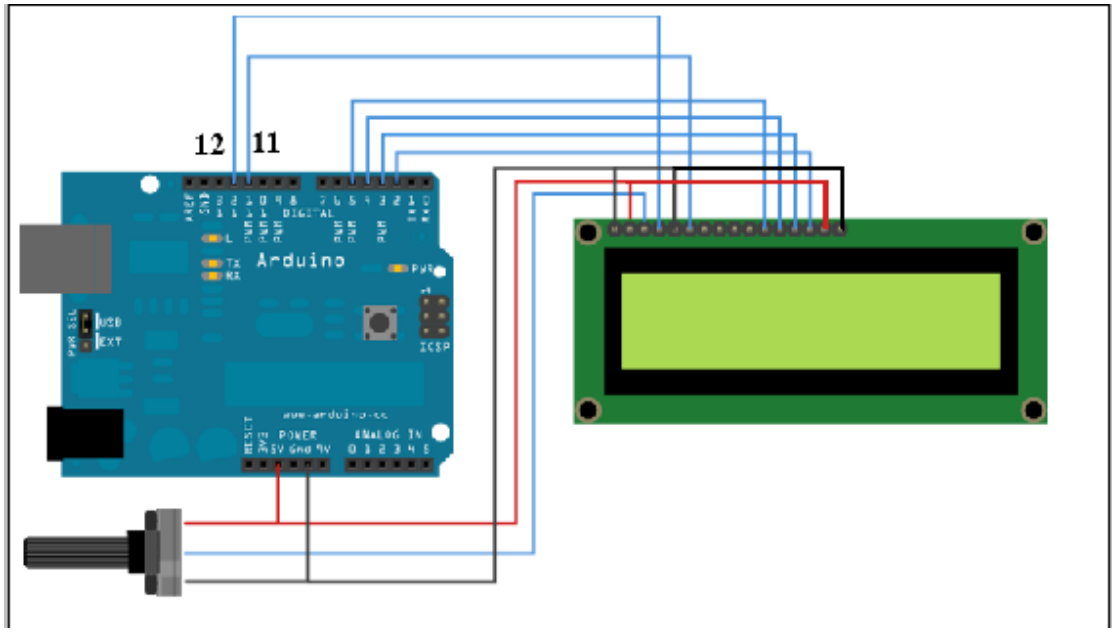


First Experiment:

Prepare: 1 X Infrared Receiver (IRM_3638) , 1 X IR Remote Control , 1 X 10 kΩ Resistors , some Jumper cable.

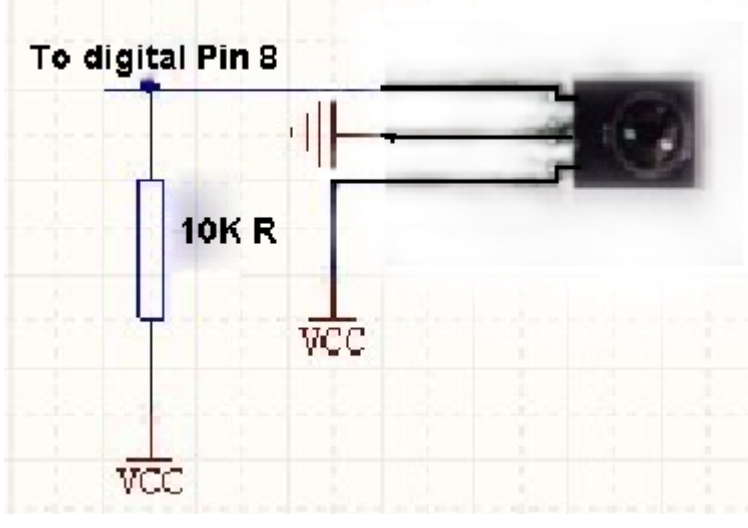


Please clip the following picture to see the large picture , to wire your LED screen to your Arduino



.

Attach the Infrared Receiver's Pin Vout to your arduino's digital Pin 8 , Please check the following picture.



Code : open the folder IR , and open the IR.pde

In the program, when you press the key of the IR Remote Control, the 1602 LCD will show the commond value of the key.

```
#include <LiquidCrystal.h>

#define IR_IN   8   //红外接收
```

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int Pulse_Width=0;//存储脉宽
int ir_code=0x00;//  用户编码值
char    adrL_code=0x00;//命令码
char    adrH_code=0x00;//命令码反码

void timer1_init(void)//定时器初始化函数
{
   TCCR1A = 0X00;
   TCCR1B = 0X05;//给定时器时钟源
   TCCR1C = 0X00;
   TCNT1 = 0X00;
   TIMSK1 = 0X00;      //禁止定时器溢出中断
}
void remote_deal(void)//执行译码结果函数
{
         //数据显示
      lcd.clear(); //清屏
      delay(1);
      lcd.setCursor(0,0);
      lcd.print(ir_code,HEX);//16 进制显示
      lcd.setCursor(0,1);
      lcd.print(adrL_code,HEX);//16 进制显示

}
char logic_value()//判断逻辑值“0”和“1”子函数
{
   TCNT1 = 0X00;
   while(!(digitalRead(8))); //低等待
   Pulse_Width=TCNT1;
   TCNT1=0;
   if(Pulse_Width>=7&&Pulse_Width<=10)//低电平 560us
   {
      while(digitalRead(8));//是高就等待
      Pulse_Width=TCNT1;
      TCNT1=0;
      if(Pulse_Width>=7&&Pulse_Width<=10)//接着高电平 560us
         return 0;
      else if(Pulse_Width>=25&&Pulse_Width<=27) //接着高电平 1.7ms
         return 1;
   }
   return -1;
}
```

```
void pulse_deal()//接收地址码和命令码脉冲函数
{
   int i;
   int j;
   ir_code=0x00;//  清零
   adrL_code=0x00;//  清零
   adrH_code=0x00;//  清零

   //解析遥控器编码中的用户编码值
   for(i = 0 ; i < 16; i++)
   {
      if(logic_value() == 1) //是 1
            ir_code |= (1<<i);//保存键值
   }
   //解析遥控器编码中的命令码
   for(i = 0 ; i < 8; i++)
   {
      if(logic_value() == 1) //是 1
         adrL_code |= (1<<i);//保存键值
   }
   //解析遥控器编码中的命令码反码
   for(j = 0 ; j < 8; j++)
   {
      if(logic_value() == 1) //是 1
            adrH_code |= (1<<j);//保存键值
   }
}
void remote_decode(void)//译码函数
{
   TCNT1=0X00;
   while(digitalRead(8))//是高就等待
   {
      if(TCNT1>=1563)   //当高电平持续时间超过 100ms，表明此时没有按键按下
      {
         ir_code=0x00ff;//  用户编码值
         adrL_code=0x00;//键码前一个字节值
         adrH_code=0x00;//键码后一个字节值
         return;
      }
   }

   //如果高电平持续时间不超过 100ms
   TCNT1=0X00;
```

```
    while(!(digitalRead(8))); //低等待
    Pulse_Width=TCNT1;
    TCNT1=0;
    if(Pulse_Width>=140&&Pulse_Width<=141)//9ms
    {

        while(digitalRead(8));//是高就等待
        Pulse_Width=TCNT1;
        TCNT1=0;
        if(Pulse_Width>=68&&Pulse_Width<=72)//4.5ms
        {
            pulse_deal();
            return;
        }
        else if(Pulse_Width>=34&&Pulse_Width<=36)//2.25ms
        {
            while(!(digitalRead(8)));//低等待
            Pulse_Width=TCNT1;
            TCNT1=0;
            if(Pulse_Width>=7&&Pulse_Width<=10)//560us
            {
                return;
            }
        }
    }
}
void

setup()
{
    unsigned char i;
    pinMode(IR_IN,INPUT);//设置红外接收引脚为输入
    lcd.begin(16, 2);
}
void loop()
{
    timer1_init();//定时器初始化
    while(1)
    {
        remote_decode();    //译码
        remote_deal();      //执行译码结果
    }
}
```