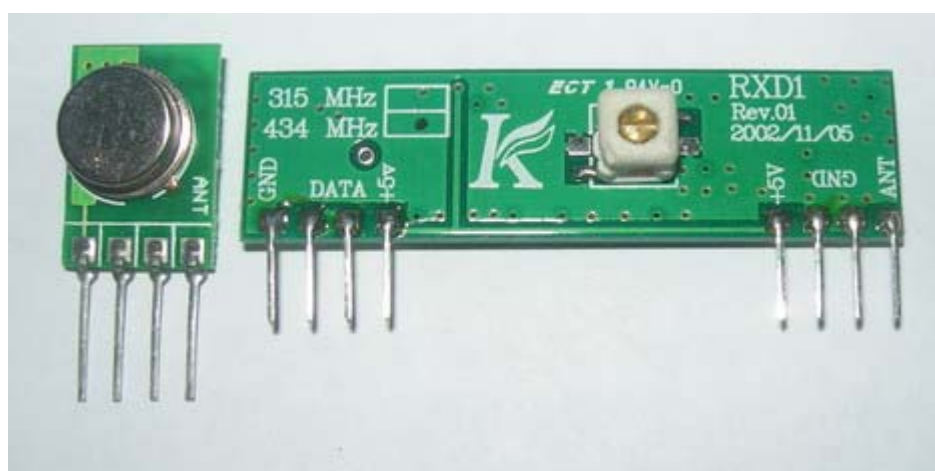


راه انداز سریع یک فرستنده و گیرنده رادیویی با کمک ماژول هار  
TX433 and RX433

نویسنده: مهدی زارعر

منبع: کویر الکترونیک



ساخت یک گیرنده و فرستنده با کمک RF modules.

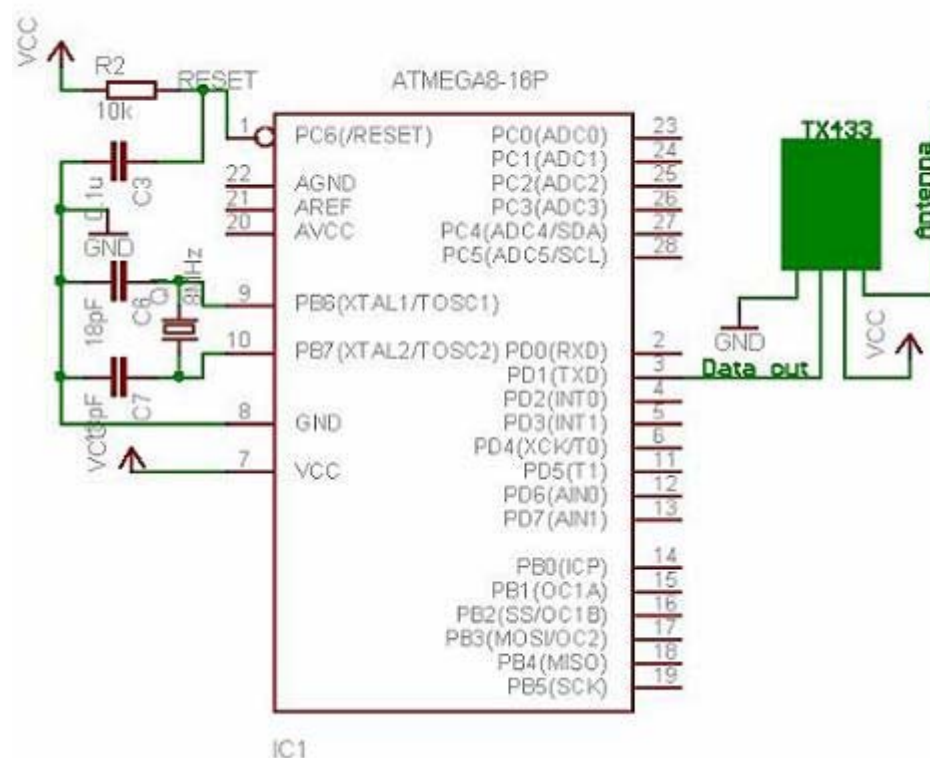
ماژول های RX433 and TX433

کار با این ماژول ها خیلی راحت می باشد و از طریق رابط سریال می توان ارتباط برقرار کرد. قیمت این ماژول ها خیلی کم بوده (در حد ۳۰۰۰/۴۰۰۰ تومان) و تا ۱۰۰ متری جواب می دهند.

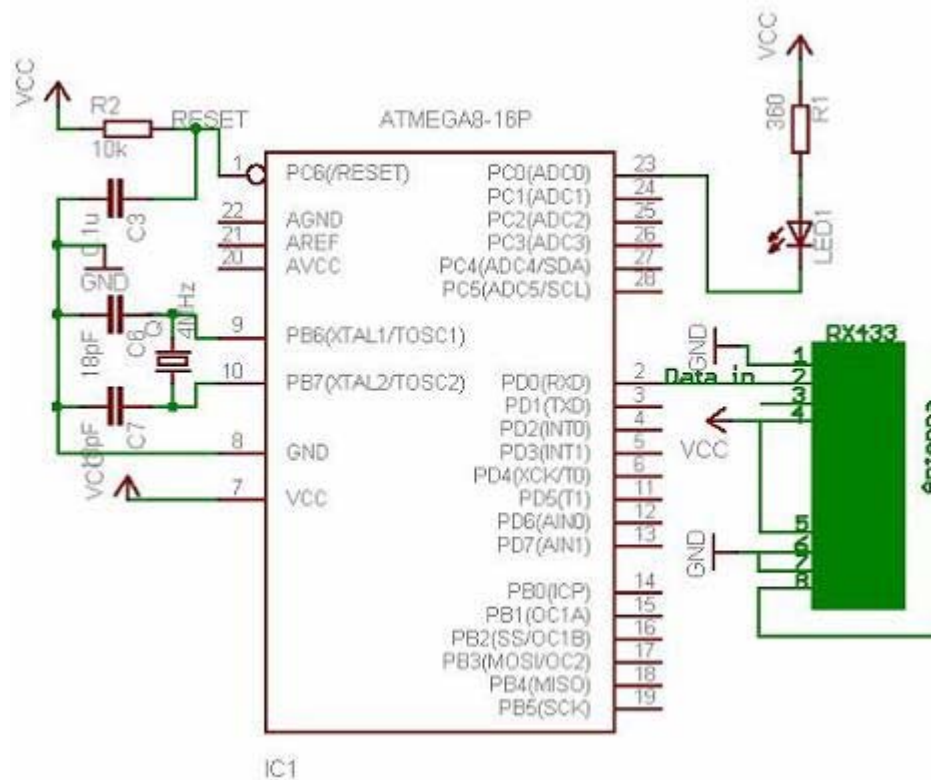
ولتاژ کاری این مازول بین ۳ تا ۱۲ ولت می باشد و چون نرمال در میکروکنترلرهای ۵ avr ولت استفاده می کنیم مشکل پیش نخواهد آمد.

فرستنده و گیرنده در فرکانس 433.92MHz تنظیم و به یک آنتن ۳۰-۳۵ سانتی نیاز دارند که می توانید از آنتن ریموت کنترل ها استفاده کنید.

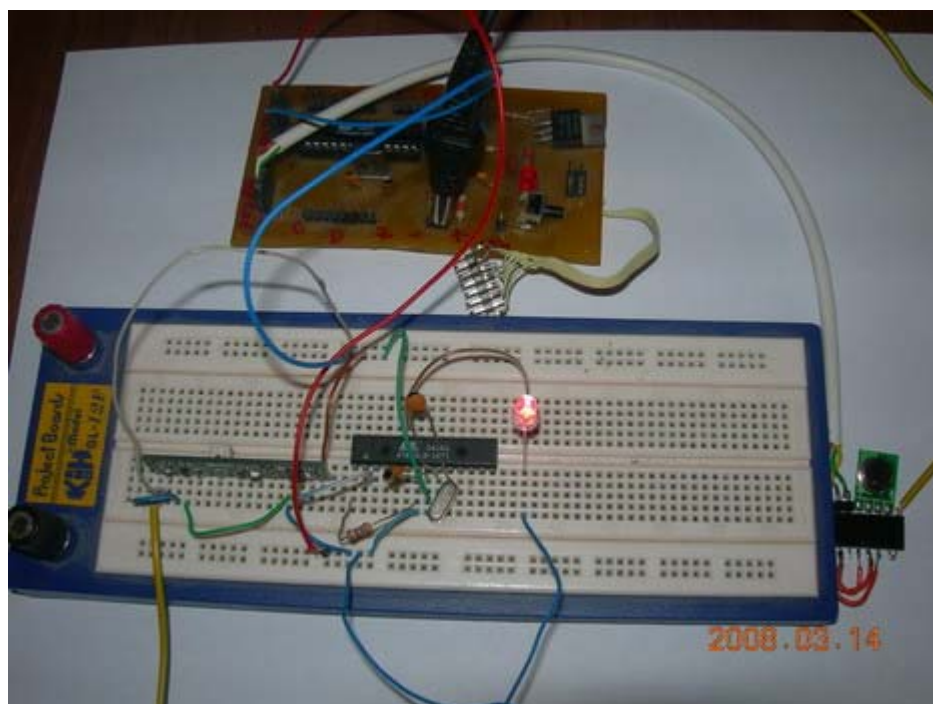
## شماتیک فرستنده



## شماتیک گیرنده



تست شده روی برد برد



بخش نرم افزار

به طور خلاصه برای ارسال دیتا به صورت رادیویی برای امنیت بالا نیاز به رمز دار کردن اطلاعات دارید که می توانید از ماژول های آماده استفاده کنید یا خیلی ساده به صورت نرم افزاری حل کنید.

در این مثال اطلاعات به صورت پکیج ۴ بیتی ارسال می گردد که به شرح زیر است:

- ارسال یک بایت برای هماهنگ سازی (10101010)
- یک بایت آدرس
- یک بایت دیتا
- یک بایت برای چک ارسال داده

شاید یک سوال پیش بیاید که چرا یک بایت برای هماهنگ سازی ارسال می گردد؟ دلیل آن هم اینست که برای جلوگیری از خطا موقعی که فرستنده دیتایی نمی فرستد و هک شدن یک بایت هماهنگ سازی ارسال می گردد.

سورس کد برای فرستنده

```
#include <avr/io.h>
#include <util/delay.h>
#ifndef F_CPU
//define cpu clock speed if not defined
#define F_CPU 8000000
#endif
//set desired baud rate
#define BAUDRATE 1200
//calculate UBRR value
#define UBRRVAL ((F_CPU/(BAUDRATE*16UL))-1)
//define receive parameters
#define SYNC 0xAA// synchro signal
#define RADDR 0x44
#define LEDON 0x11//switch led on command
#define LEDOFF 0x22//switch led off command
void USART_Init(void)
{
    //Set baud rate
    UBRRL=(uint8_t)UBRRVAL; //low byte
    UBRRH=(UBRRVAL>>8); //high byte
    //Set data frame format: asynchronous mode,no parity, 1 stop
    bit, 8 bit size
    UCSRC=(1<<URSEL)|(0<<UMSEL)|(0<<UPM1)|(0<<UPM0)|
        (0<<USBS)|(0<<UCSZ2)|(1<<UCSZ1)|(1<<UCSZ0);
    //Enable Transmitter and Receiver and Interrupt on receive
    complete
    UCSRB=(1<<TXEN);
}
void USART_vSendByte(uint8_t u8Data)
{
    // Wait if a byte is being transmitted
    while((UCSRA&(1<<UDRE)) == 0);
    // Transmit data
    UDR = u8Data;
}
void Send_Packet(uint8_t addr, uint8_t cmd)
{
    USART_vSendByte(SYNC);//send synchro byte
    USART_vSendByte(addr);//send receiver address
    USART_vSendByte(cmd);//send increment command
```

```

        USART_vSendByte((addr+cmd)); //send checksum
    }
    void delayms(uint8_t t) //delay in ms
    {
        uint8_t i;
        for(i=0; i<t; i++)
            _delay_ms(1);
    }
    int main(void)
    {
        USART_Init();
        while(1)
        {
            //endless transmission
            //send command to switch led ON
            Send_Packet(RADDR, LEDON);
            delayms(100);
            //send command to switch led OFF
            Send_Packet(RADDR, LEDOFF);
            delayms(100);
        }
        return 0;
    }
}

```

در برنامه بالا از Baud rate 1200 استفاده شده است. ما کمترین نرخ انتقال 8kb/s هست که در عمل و در فواصل مختلف بهترین و مطمئن ترین نرخ ارسال Baud rate 1200 می باشد.

فرستنده دو دستور (LEDON and LEDOFF) با فاصله زمانی ۱۰۰ میلی ثانیه به گیرنده می فرستد که گیرنده پس از دریافت بنا به دستور LED را روشن و یا خاموش می کند. در عمل باید LED به صورت منظم چشمک زن باشد و وقفه ای نداشته نباشد این نشانه درست عمل کردن مدار است.

سورس کد برای گیرنده:

```

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#ifndef F_CPU
//define cpu clock speed if not defined
#define F_CPU 4000000
#endif
//set desired baud rate
#define BAUDRATE 1200
//calculate UBRR value
#define UBRRVAL ((F_CPU/(BAUDRATE*16UL))-1)
//define receive parameters
#define SYNC 0XAA // synchro signal
#define RADDR 0x44
#define LEDON 0x11 //LED on command
#define LEDOFF 0x22 //LED off command
void USART_Init(void)
{
    //Set baud rate
    UBRR0L=(uint8_t)UBRRVAL; //low byte
    UBRR0H=(UBRRVAL>>8); //high byte
}

```

```

        //Set data frame format: asynchronous mode,no parity, 1 stop
bit, 8 bit size
        UCSRC=(1<<URSEL)|(0<<UMSEL)|(0<<UPM1)|(0<<UPM0)|
                (0<<USBS)|(0<<UCSZ2)|(1<<UCSZ1)|(1<<UCSZ0);
        //Enable Transmitter and Receiver and Interrupt on receive
complete
        UCSRB=(1<<RXEN)|(1<<RXCIE); //|(1<<TXEN);
        //enable global interrupts
    }
    uint8_t USART_vReceiveByte(void)
    {
        // Wait until a byte has been received
        while((UCSRA&(1<<RXC)) == 0);
        // Return received data
        return UDR;
    }
    ISR(USART_RXC_vect)
    {
        //define variables
        uint8_t raddress, data, chk;//transmitter address
        //receive destination address
        raddress=USART_vReceiveByte();
        //receive data
        data=USART_vReceiveByte();
        //receive checksum
        chk=USART_vReceiveByte();
        //compare received checksum with calculated
        if(chk==(raddress+data))//if match perform operations
        {
            //if transmitter address match
            if(raddress==RADDR)
            {
                if(data==LEDON)
                {
                    PORTC&=~(1<<0);//LED ON
                }
                else if(data==LEDOFF)
                {
                    PORTC|=(1<<0);//LED OFF
                }
                else
                {
                    //blink led as error
                    PORTC|=(1<<0);//LED OFF
                    _delay_ms(10);
                    PORTC&=~(1<<0);//LED ON
                }
            }
        }
    }
}
void Main_Init(void)
{
    PORTC|=(1<<0);//LED OFF
    DDRC=0X001;//define port C pin 0 as output;
    //enable global interrupts
    sei();
}
int main(void)
{
    Main_Init();
    USART_Init();
}

```

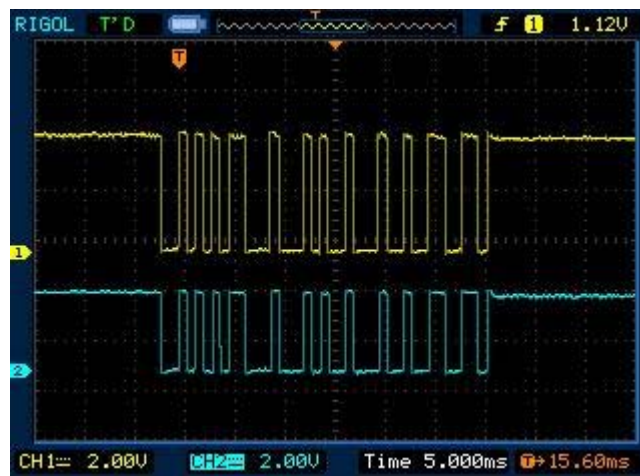
```

while(1)
{
}
//nothing here interrupts are working
return 0;
}

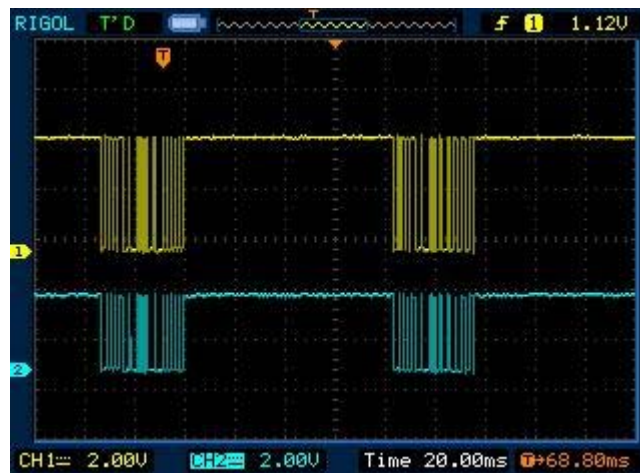
```

نکته مهم اینکه استفاده از آنتن ضروری می باشد.

در زیر می توانید سیگنال های فرستنده و گیرنده را ببینید سیگنال زرد رنگ مربوط به فرستنده (TX) و سیگنال آبی رنگ مربوط به گیرنده (RX) می باشد.



در شکل زیر هم نشان داده شده که بین پکیج های دیتا هیچ نویزی وجود ندارد



Source:

<http://winavr.scienceprog.com/example-avr-projects/running-tx433-and-rx433-rf-modules-with-avr-microcontrollers.html>

source code:

<http://winavr.scienceprog.com/download/Rf433.zip>