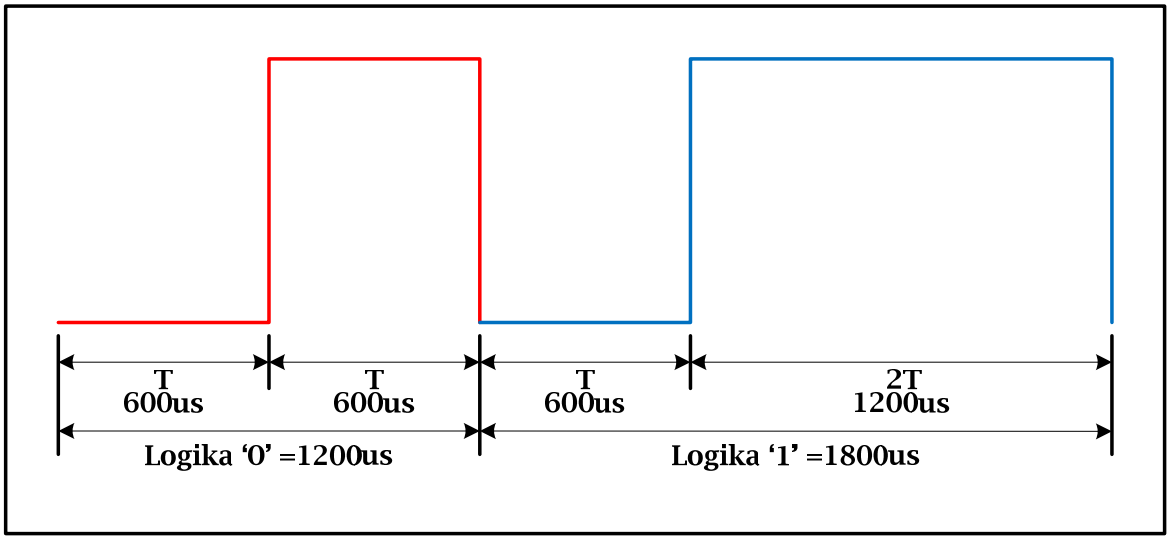
Pendahuluan

Standar Remote Kontrol yang mudah untuk dimengerti dan diaplikasikan adalah standar SIRC atau lebih dikenal dengan standar SONY. Bagian terkecil dari sinyal pembacaan pada standar ini adalah **T** yang memiliki periode sebesar 600us.

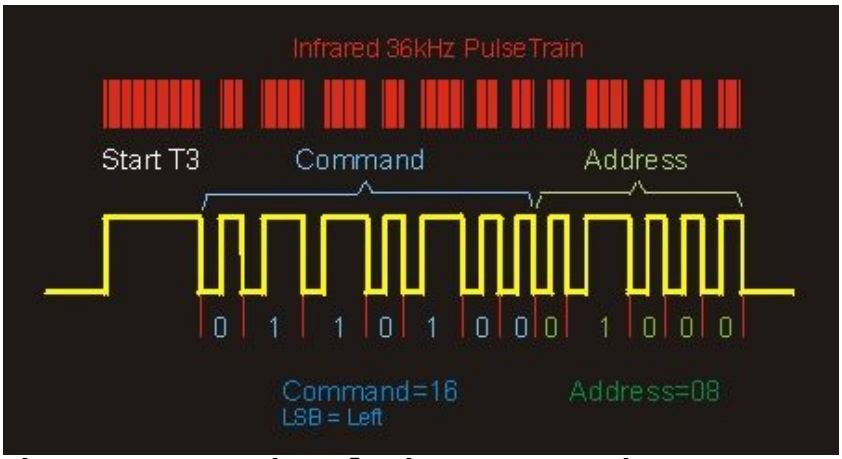
Sistem pengkodean yang digunakan pada remote kontrol standar SONY ini adalah ***Pulse Code***, jadi yang membedakan logika *high* atau *low* terletak pada panjang pulsanya.



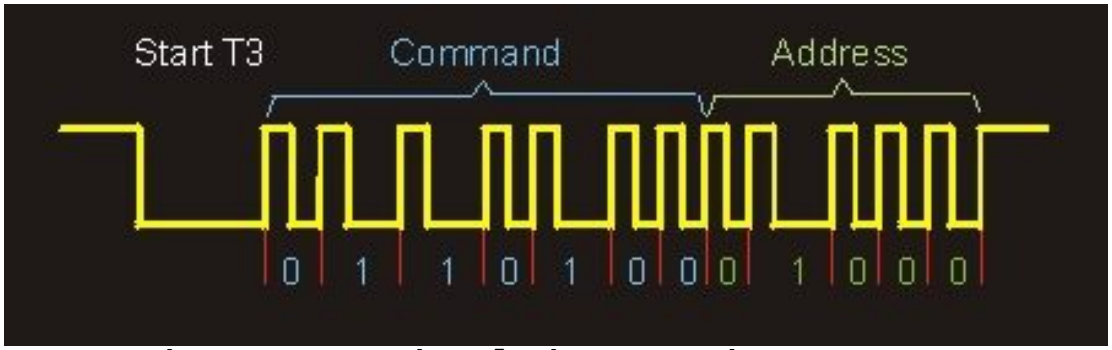
Gambar 3.1 Panjang Pulsa untuk tiap Level Logika

Format Data Remote Standar SONY

Pada dasarnya pembacaan remote kontrol SONY ini terdiri dari start bit dengan panjang pulsa 2.4 ms, dimana untuk logika '1' panjang pulsanya 1.2 ms dan logika '0' panjang pulsanya 0.6 ms yang masing-masing dipisahkan oleh jarak sebesar 0.6 ms. Format data SONY ada 3 macam yaitu 12 bit, 15 bit dan 20 bit, tapi dalam pembahasan kali ini yang akan digunakan adalah format 12 bit, karena format ini banyak digunakan pada remote kontrol SONY yang terdapat dipasaran.



Gambar 3.2 Bentuk Gelombang Transmit Remote SONY



Gambar 3.3 Bentuk Gelombang Receive Remote SONY

Pada saat data dikirim pertama kali yang dikeluarkan adalah header dengan panjang pulsa 3T (2400ms) setelah itu *command* 8 bit dan yang terakhir *address device* 4 bit, untuk *command* urutan datanya LSB sebelah kiri sedangkan *address* LSB sebelah kanan, seperti yang ditunjukkan oleh gambar 3.2. Setelah data dipancarkan maka akan diterima oleh receiver dengan kondisi pulsa dibalik atau *active low*, seperti yang ditunjukkan oleh gambar 3.3.

Pembacaan Remote Standar SIRC

Teknik pembacaan untuk standar ini adalah sebagai berikut:

1. Deteksi pulsa low selama 2400 ms apabila tidak sama maka bukan standar SIRC.
2. Deteksi pulsa low untuk level logika, apabila 0.6 ms = ‘0’ tapi apabila 1.2 ms = ‘1’.
3. Untuk no. 2 ulangi sebanyak 12 kali dan simpan hasilnya.

Aplikasi Penerima dengan ATmega8535

Pemrograman mikrokontroller ATmega8535 sebagai penerima remote standar SIRC ini tekniknya sama hanya perlu ada beberapa penyesuaian agar didapatkan hasil yang maksimal. Penyesuaian tersebut perlu dilakukan karena ada beberapa hal yang mempengaruhi diantaranya adalah kecepatan mikrokontroller serta komponen yang digunakan. Dalam penyesuaian ini yang dilakukan hanya menambah atau mengurangi sedikit dari timer, yang digunakan untuk mendeteksi periode pulsa yang diterima. Berikut ini akan disampaikan program dengan menggunakan codevision untuk mendeteksi pulsa yang diterima untuk remote jenis SONY atau SIRC.

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.3 Evaluation
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 8/4/2008
Author  : Freeware, for evaluation and non-commercial use only
Company :
Comments:


Chip type           : ATmega32
Program type        : Application
Clock frequency     : 11.059200 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 512
*****************************************************/
#include <mega32.h>
#include <stdio.h>
#include <delay.h>

unsigned                                                    char
count,data1,data2,data3,data4,data5,data6,data7,data8;
unsigned char data11,data12,data13,data14;
unsigned char start;
void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
//  Func7=In  Func6=In  Func5=In  Func4=In  Func3=In  Func2=In
Func1=In Func0=In
//  State7=T  State6=T  State5=T  State4=T  State3=T  State2=T
State1=T State0=T
PORTA=0x00;
DDRA=0x00;

// Port B initialization
//  Func7=In  Func6=In  Func5=In  Func4=In  Func3=In  Func2=In
Func1=In Func0=In
//  State7=T  State6=T  State5=T  State4=T  State3=T  State2=T
State1=P State0=P
PORTB=0x03;
```

```
DDRB=0x00;

// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
//  State7=1  State6=1  State5=1  State4=1  State3=1  State2=1
State1=1 State0=1
PORTC=0xFF;
DDRC=0xFF;

// Port D initialization
//  Func7=In  Func6=In  Func5=In  Func4=In  Func3=In  Func2=In
Func1=In Func0=In
//  State7=T  State6=T  State5=T  State4=T  State3=T  State2=T
State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;
```

```
// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

while (1)
      {
      while (PINB.0==1) {};
      while (PINB.0==0)
       {
         delay_us(100);
        count++;
        };
      start=count;
      count=0;
      if (start>=24 && start<=27) //jika header kurang dari 24
                      //atau lebih dari 27 deteksi ulang
      {
      while (PINB.0==1) {};
      while (PINB.0==0)
        {
         delay_us(100);
        count++;
        };
      data1=count;
      count=0;
     while (PINB.0==1) {};
```

```
 while (PINB.0==0)
   {
    delay_us(100);
   count++;
   };
 data2=count;
 count=0;
while (PINB.0==1) {};
while (PINB.0==0)
   {
    delay_us(100);
   count++;
   };
 data3=count;
 count=0;
while (PINB.0==1) {};
while (PINB.0==0)
   {
    delay_us(100);
   count++;
   };
 data4=count;
 count=0;
while (PINB.0==1) {};
while (PINB.0==0)
   {
    delay_us(100);
   count++;
   };
 data5=count;
 count=0;
while (PINB.0==1) {};
while (PINB.0==0)
   {
    delay_us(100);
   count++;
   };
 data6=count;
 count=0;
while (PINB.0==1) {};
while (PINB.0==0)
   {
    delay_us(100);
   count++;
   };
 data7=count;
 count=0;
 while (PINB.0==1) {};
 while (PINB.0==0)
```

```
        {
         delay_us(100);
        count++;
        };
       data8=count;
       count=0;
       while (PINB.0==1) {};
       while (PINB.0==0)
        {
        delay_us(100);
        count++;
        };
       data11=count;
       count=0;
      while (PINB.0==1) {};
      while (PINB.0==0)
        {
        delay_us(100);
        count++;
        };
       data12=count;
       count=0;
      while (PINB.0==1) {};
      while (PINB.0==0)
        {
        delay_us(100);
        count++;
        };
       data13=count;
       count=0;
       while (PINB.0==1) {};
       while (PINB.0==0)
        {
        delay_us(100);
        count++;
        };
       data14=count;
       count=0;
       data1=((data1/5)-1)*1;//data hasil deteksi pulsa dibagi
5 dikurangi 1
       data2=((data2/5)-1)*2;//jadi   jika   datanya   6   maka
outputnya akan = 0
       data3=((data3/5)-1)*4;//sedangkan jika datanya 12 maka
outputnya akan = 1
       data4=((data4/5)-1)*8;//lalu  hasil  tersebut  dikalikan
dengan nilai2 bit
       data5=((data5/5)-1)*16;
       data6=((data6/5)-1)*32;
       data7=((data7/5)-1)*64;
```

```
      data8=((data8/5)-1)*128;
      PORTC=data8+data7+data6+data5+data4+data3+data2+data1;//
hasilnya dimasukkan ke PORTC
     }
      };
}
```

Kesimpulan

Aplikasi diatas dapat dikembangkan menjadi remote control untuk pengendali alat rumah tangga secara *wireless*, pengendalian robot, komunikasi data dan masih banyak lagi.

Semoga bermanfaat_____(wspambudi2008)