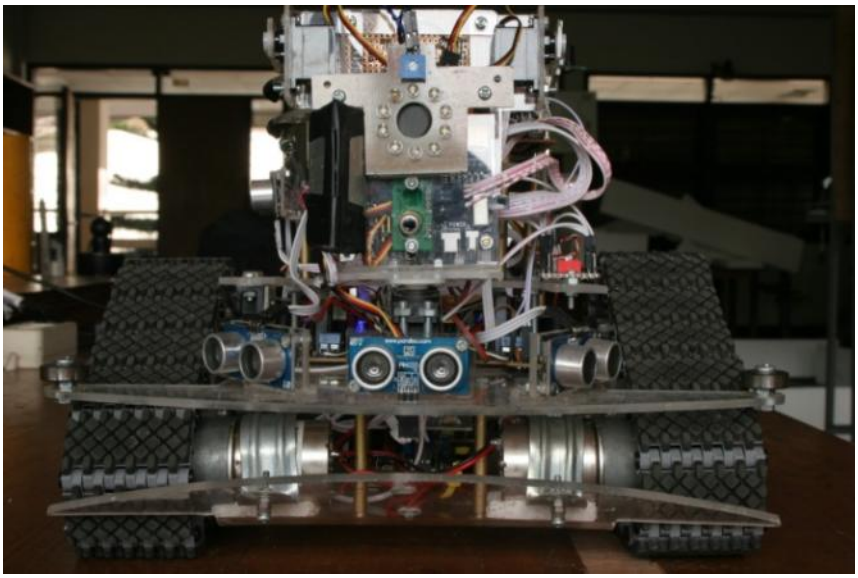
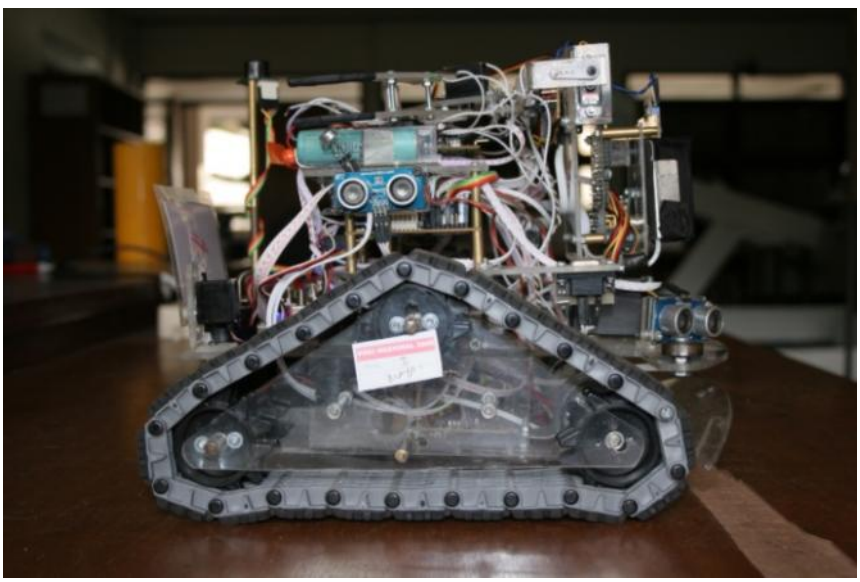
LAMPIRAN A
FOTO ROBOT MOBIL TANK

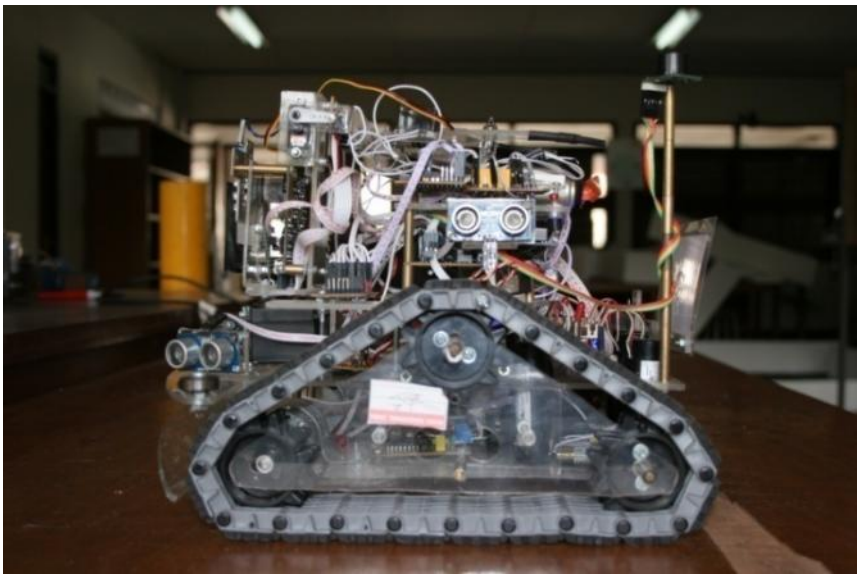
TAMPAK DEPAN



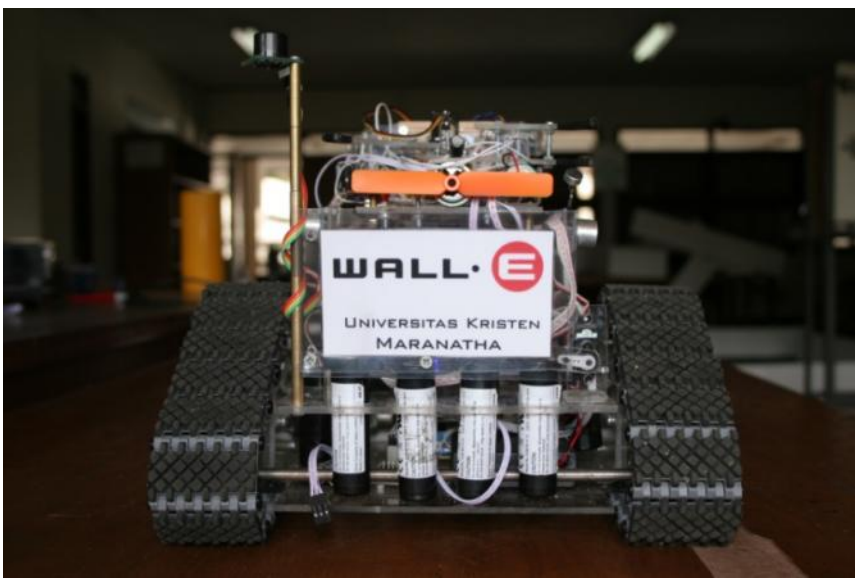
TAMPAK SAMPING KANAN



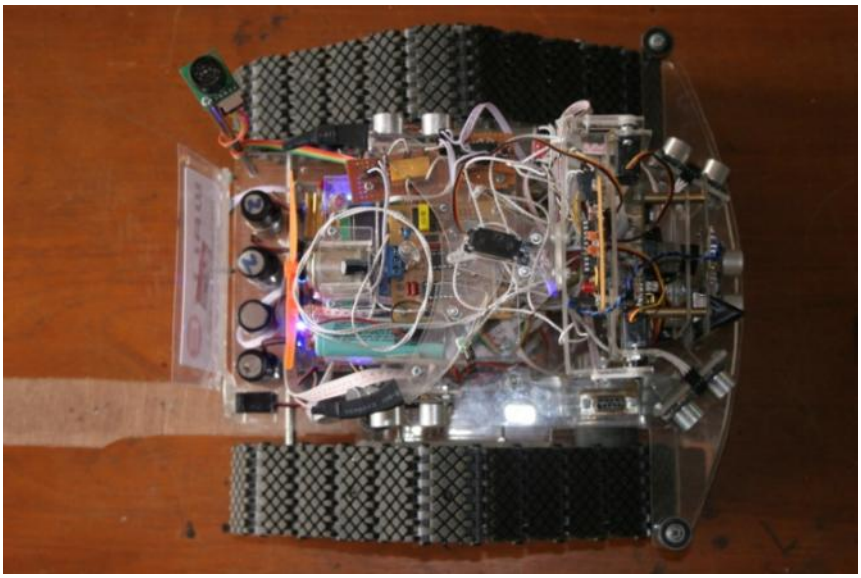
TAMPAK SAMPING KIRI



TAMPAK BELAKANG



TAMPAK ATAS



LAMPIRAN B
PROGRAM PADA PENGONTROL MIKRO ATMEGA16

PROGRAM UTAMA

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.3 Standard
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project : WALL_E
Version : 1
Date    : 3/3/2009
Author  : SUFENDI && WIWIK && TEAM
Company : LAB FISIKA
Comments: KRCI EXPERT SINGLE 2009


Chip type           : ATmega16
Program type        : Application
Clock frequency     : 11.059200 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 256
*****************************************************/

#include <mega16.h>
#include <delay.h>
#include <stdio.h>

// I2C Bus functions
#asm
   .equ __i2c_port=0x18 ;PORTB
   .equ __sda_bit=4
   .equ __scl_bit=0
#endasm
#include <i2c.h>

unsigned char text[32],dat2[32],dat[32];
unsigned int i,t,ki,kik,te,kak,ka,z,x,ar,pos,ats,atas1,atas2,b,a,c,d;
unsigned char kps,kps2,temp,temp1,temp2,temp3,temp4,temp5,temp6,servo;
char tangga(void);
char bayi(void);
char lilin(void);
char cek(void);
bit ada2,ada;

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
```

```
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
char status,data;
status=UCSRA;
data=UDR;
if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
   rx_buffer[rx_wr_index]=data;
   if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
   if (++rx_counter == RX_BUFFER_SIZE)
      {
      rx_counter=0;
      rx_buffer_overflow=1;
      };
   };
   if(data==255) x=0;// carry return
   dat[x]=data;
   x++;
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
char data;
while (rx_counter==0);
data=rx_buffer[rx_rd_index];
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;
#asm("sei")
return data;
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

// Timer 0 output compare interrupt service routine
interrupt [TIM0_COMP] void timer0_comp_isr(void)
{
```

```
// Place your code here

}

// Timer 2 output compare interrupt service routine
interrupt [TIM2_COMP] void timer2_comp_isr(void)
{
// Place your code here

}

// Declare your global variables here

void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=Out Func4=In Func3=In Func2=In Func1=Out Func0=Out
// State7=P State6=T State5=0 State4=T State3=P State2=T State1=0 State0=0
PORTA=0x88;
DDRA=0x23;


// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=Out Func2=Out Func1=Out Func0=In
// State7=T State6=T State5=T State4=T State3=0 State2=0 State1=0 State0=T
PORTB=0x00;
DDRB=0x0E;

// Port C 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
PORTC=0x00;
DDRC=0x00;

// Port D initialization
// Func7=Out Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=0 State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x80;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: 10.800 kHz
// Mode: Fast PWM top=FFh
// OC0 output: Non-Inverted PWM
TCCR0=0x6D;
TCNT0=0x00;
OCR0=0xFF;

// 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: 10.800 kHz
// Mode: Fast PWM top=FFh
// OC2 output: Non-Inverted PWM
ASSR=0x00;
TCCR2=0x6F;
TCNT2=0x00;
OCR2=0xFF;

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

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x82;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 115200
UCSRA=0x00;
UCSRB=0x98;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x05;

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

// I2C Bus initialization
i2c_init();

// LCD module initialization
lcd_init(16);

// Global enable interrupts
#asm("sei")
pos=0;
ada=0;//bayi
ada2=0;//api
b=0;
while (1)
      {

//-----------------------------------------------------------------------------------------------
//                  lantai 1
//-----------------------------------------------------------------------------------------------
//------------------inisialisasi awal-----------------------------------------------------------
ar=160; //arah tangga
switch (pos) {
```

```
case  0:


//cek sound activation.....................
//--------------------------------------
     OCR0=0x00;
     OCR2=0x00;
     lcd_clear();
      sprintf(dat2,"    mana tepuk   tangannya!!!!!");
       lcd_puts(dat2);
       wait:
      if(PINA.6==1  ) goto next;
      else goto wait;
 next:

//setting kamera
  printf("RS \r");
  delay_ms(100);
  printf("sv 0 120\r");
  delay_ms(100);
  printf("RM 3 \r");
  delay_ms(100);
  printf("PM 0 \r");
  delay_ms(100);

goto no;
//--------------------bila kondisi dibawah tangga-----------------------------------------------------------

  bawahtangga:  kompas();
              kps2=kps;
              //patok kanan bawah tangga
              if(kps2>=ar || kps2<=(ar-128) )
              {
                    while(pos==0)
                    {
                    if(PINA.7==1) cek();
                    if(PINA.7==0) c=0;
                    if(pos==1) goto end1;
                    OCR2=0xcc; //kanan
                    if(kak>=40) {kak=55;OCR2=0xee; }
                    if(kak<=10) kak=10;
                    PORTB.2=0;
                    PORTB.1=0;
                    OCR0=(-5.1*kak)+306; //kanan
                    lcd_clear();
                    sprintf(dat2,"bwh kak=%3d ocr0=%3d",kak,OCR0);
                    lcd_puts(dat2);
                    kanank();
                    tengah();
                    kirik();
                          if(te<=8 && kak<=20 && kik<=20)
                          {
                                while(kps<(ar-5) || kps>(ar+5))//putar 180 derajat
                                {
                                PORTB.2=0;
                                PORTB.1=1;
                                OCR0=0xdd;
                                OCR2=0xdd;
                                kompas();
                                }
                                while(pos==0) //patok kiri
                                {
                                if(PINA.7==1) cek();
                                if(PINA.7==0) c=0;
                                if(pos==1) goto end1;
                                OCR0=0xcc; //kanan
                                if(kik>=40) {kik=55;OCR0=0xee; }
```

```
                    if(kik<=10) kik=10;
                        if( te<=8 || kak<=15)
                        {
                            while(te<=8 || kak<=15)
                            {
                            PORTB.2=0;//kanan
                            PORTB.1=1;///kiri
                            OCR0=0xdd;
                            OCR2=0xdd;
                            tengah();
                            kanank();
                            }
                        }
                    PORTB.2=0;
                    PORTB.1=0;
                    OCR2=(-5.1*kik)+306; //kiri
                    lcd_clear();
                    sprintf(dat2," bwh1 kik=%3d ocr2=%3d",kik,OCR2);
                    lcd_puts(dat2);
                    kirik();
                    tengah();
                    kanank();
                    }
                }
            }
}
//patok kiri bawah tangga
else if((ar-128)<kps2<ar)
{
    while(pos==0)
    {
    if(PINA.7==1) cek();
    if(PINA.7==0) c=0;
    if(pos==1) goto end1;
    OCR0=0xcc; //kanan
    if(kik>=40) {kik=55;OCR0=0xee; }
    if(kik<=10) kik=10;
    PORTB.2=0;
    PORTB.1=0;
    OCR2=(-5.1*kik)+306; //kiri
    lcd_clear();
    sprintf(dat2," bwh kik=%3d ocr2=%3d",kik,OCR2);
    lcd_puts(dat2);
    kirik();
    tengah();
    kanank();
        if(te<=8 && kak<=20 && kik<=20)
        {
            while(kps<(ar-5) || kps>(ar+5))//putar 180 derajat
            {
            PORTB.2=0;
            PORTB.1=1;
            OCR0=0xdd;
            OCR2=0xdd;
            kompas();
            }
            while(pos==0) //patok kanan
            {
            if(PINA.7==1) cek();
            if(PINA.7==0) c=0;
            if(pos==1) goto end1;
            OCR2=0xcc; //kanan
            if(kak>=40) {kak=55;OCR2=0xee; }
            if(kak<=10) kak=10;
                if( te<=8 || kak<=15)
                {
                    while(te<=8 || kik<=15)
```

```
                                    {
                                    PORTB.2=1;//kanan
                                    PORTB.1=0;///kiri
                                    OCR0=0xdd;
                                    OCR2=0xdd;
                                    tengah();
                                    kirik();
                                    }
                                }
                            PORTB.2=0;
                            PORTB.1=0;
                            OCR0=(-5.1*kak)+306; //kanan
                            lcd_clear();
                            sprintf(dat2,"bwh1 kak=%3d ocr0=%3d",kak,OCR0);
                            lcd_puts(dat2);
                            kanank();
                            tengah();
                            kirik();
                            }
                        }
                    }
                }

//----------------------akhir dari kondisi dibawah tangga-----------------------------------------------
//-----------------tidak ada  tangga-------------------------------------------------------------------
no:     lcd_clear();
        sprintf(dat2,"ga dapat tangga");
        lcd_puts(dat2);
        while( kps<(ar-5) || kps>(ar+5))//putar kearah tangga
        {
        PORTB.2=0;
        PORTB.1=1;
        OCR0=0xdd;
        OCR2=0xdd;
        kompas();
        lcd_clear();
        sprintf(dat2,"ar=%d kps=%d",ar,kps);
        lcd_puts(dat2);
        }
        tengah();
        kirik();
        kanank();
        ulang:
        while(te>10)//maju osilasi
        {
        if(PINA.7==1) cek();
        if(PINA.7==0) c=0;
        if(pos==1) goto end1;
        tengah();
        kirik();
        kanank();
        if(kik>=20) kik=20;
        if(kik<=10) kik=10;
        if(kak>=20) kak=20;
        if(kak<=10) kak=10;
        PORTB.2=0;
        PORTB.1=0;
        OCR2=(25.5*kak)-255; //kiri
        OCR0=(25.5*kik)-255; //kanan
        }
        tengah();
        kirik();
        kanank();
        if(te<15 && kik<40 && kak<40)
        goto terus;
        else goto ulang;
        terus:
```

```
        atas();
        delay_ms(200);
        while(te<30 || kik<12)//putar kanan awal
        {
        OCR0=0xff;
        OCR2=0xff;
        PORTB.2=1;//kanan
        PORTB.1=0;//kiri
        tengah();
        kirik();
        }
//----------------proses perulangan di lantai satu-----------------------------------------------------
loop:
        atas();
        tengah();
        kiri();
        kanan();
        kirik();
        kanank();
        if(ats<25) goto bawahtangga;
        if(PINA.7==1) cek();
        if(PINA.7==0) c=0;
        if(pos==1) goto end1;

        if(PINA.3==1)
        {
                if(te<20 || kak<35)
                {
                OCR0=0x00;
                OCR2=0x00;
                lilin();
                        while(dat[8]>=10 || dat[9]>=10 || kik<12)
                        {
                        PORTB.2=1;//kanan
                        PORTB.1=0;//kiri
                        OCR0=0xff;
                        OCR2=0xff;
                        lilin();
                        kirik();
                        lcd_clear();
                        sprintf(dat2,"%d %d kik=%d",dat[8],dat[9],kik);
                        lcd_puts(dat2);
                        }
                }

        }
        //depan atau kiri ada halanganm putar kanan
        if(te<8 || kik<12)
        {
                while(te<8 || kak<12 ||kik<12)
                {
                PORTB.2=1;//kanan
                PORTB.1=0;//kiri
                OCR0=0xff;
                OCR2=0xff;
                tengah();
                kanank();
                kirik();
                lcd_clear();
                sprintf(dat2,"kak=%d te=%d kik=%d",kak,te,kik);
                lcd_puts(dat2);
                }
        }
        //kanan ada halangan
        if(kak<10)
        {
                if(kik>12)
```

```
            { //putar kiri menghindar
                while(te<10 || kak<12)
                {
                OCR0=0xff;
                OCR2=0xff;
                PORTB.2=0;//kanan
                PORTB.1=1;//kiri
                tengah();
                kanank();
                lcd_clear();
                sprintf(dat2,"kanan=%3d kiri>12",kak);
                lcd_puts(dat2);
                }
            }
            else
            {//putar kanan menghindar
                 while(te<10 || kak<12 || kik<12)
                {
                OCR0=0xff;
                OCR2=0xff;
                PORTB.2=0;//kanan
                PORTB.1=1;//kiri
                tengah();
                kirik();
                kanank();
                lcd_clear();
                sprintf(dat2,"kanan=%3d kiri=%3d",kak,kik);
                lcd_puts(dat2);
                }
            }
        }
        //kondisi normal,patok kiri
        else
        {
        a=0;
        OCR0=0xcc; //kanan
        if(kik>=40) {kik=55;OCR0=0xee;}
        PORTB.2=0;//kanan
        PORTB.1=0;//kiri
        OCR2=(-5.1*kik)+306; //kiri
        lcd_clear();
        sprintf(dat2,"GA ADA kik=%3d ocr2=%3d ",kik,OCR2);
        lcd_puts(dat2);
        }
        goto loop;

end1:

 break;


//-----------------------------------------------------------------------------------------------------------
//              lantai 2
//-----------------------------------------------------------------------------------------------------------

case  1:
        OCR0=0x00;
        OCR2=0x00;
        for(z=0;z<70;z++)//bendera turun
        {
        PORTA.0=1;
        delay_us(1800);
        PORTA.0=0;
        delay_ms(18);
        }
        printf("RS \r"); //setting kamera
        delay_ms(100);
```

```
        printf("sv 0 120\r");
        delay_ms(100);
        printf("RM 3 \r");
        delay_ms(100);
        printf("PM 0 \r");
        bayi();
        delay_ms(100);



loop1:

if(PINA.2==1 && ada==0) ada2=1;//bila global mendeteksi api  sebelum bayi diselamatkan
if(PINA.3==1 && ada==1) goto api;

//----------------------------------program gerak ikut dinding kanan-----------------------------------------------------

if(ada==1 && ada2==1)//api ditemukan terlebih dahulu sebelum bayi dan bayi telah diselamatkan
{
tengah();
kiri();
kanan();
kanank();
kirik();
OCR2=0xcc; //kanan
if(kak>=40) {kak=55;OCR2=0xee; }
PORTB.2=0;//kanan
PORTB.1=0;//kiri
OCR0=(-5.1*kak)+306; //kiri
lcd_clear();
sprintf(dat2,"lnt2 kak=%3d ocr0=%3d %d %d %d",kak,OCR0,PINA.3);
lcd_puts(dat2);
//program menghindar dan perlu revisi*****************************************
       if(te<8)
       {//putar kiri
              if(PINA.2==1) goto api;
              while(te<30 || kak<12)
              {
              OCR0=0xff;
              OCR2=0xff;
              PORTB.2=0;//kanan
              PORTB.1=1;//kiri
              tengah();
              kanank();
             }
      }
      if(kak<10 && kik>18)
      {//putar kiri
              while(kak<12)
              {
              OCR0=0xcc;
              OCR2=0xcc;
              PORTB.2=0;//kanan
              PORTB.1=1;//kiri
              kanank();
              }
      }
      if(kik<10)
      {
              if(kak>12)
              {//putar kanan dikit
                     while(kik<12)
                     {
                     OCR0=0xcc;
                     OCR2=0xcc;
                     PORTB.2=1;//kanan
```

```
                PORTB.1=0;//kiri
                kirik();
                }
            }
            else
            {//putar kiri
                while(kak<12)
                {
                OCR0=0xcc;
                OCR2=0xcc;
                PORTB.2=0;//kanan
                PORTB.1=1;//kiri
                kanank();
                }
            }
        }

}

//-----------------------------------program gerak ikut dinding kiri-----------------------------------------------


else
{
//-----bila bayi belum ditemukan dan cek bayi
      if(ada==0)
      {
            if(dat[8]>=10 && dat[9]>=7 )//ada warna  bayi
            {
            OCR0=0x00; //kanan
            OCR2=0x00; //kiri
            printf("RS \r"); //setting kamera
            delay_ms(100);
            printf("sv 0 120\r");
            delay_ms(100);
            printf("RM 3 \r");
            delay_ms(100);
            printf("PM 0 \r");
            bayi();
            delay_ms(100);
            kompas();
            kps2=kps;
                while(dat[2]<=34 || dat[2]>=54 || dat[8]<=2 || dat[9]<=2)
                {
                OCR0=0x00;
                OCR2=0x00;
                if(kps<=kps2-30 || kps>kps2+30) goto loop1;
                    if(dat[2]<=54)
                    {
                    PORTB.2=1;//kanan
                    PORTB.1=0;//kiri
                    }
                    if(dat[2]>=34)
                    {
                    PORTB.2=0;//kanan
                    PORTB.1=1;//kiri
                    }
                printf("RS \r"); //setting kamera
                delay_ms(100);
                printf("sv 0 120\r");
                delay_ms(100);
                printf("RM 3 \r");
                delay_ms(100);
                printf("PM 0 \r");
                bayi();
                delay_ms(100);
                OCR0=0xff;
```

```
        OCR2=0xff;
        lcd_clear();
        sprintf(dat2,"%d %d %d kps2=%d kps=%d",dat[2],dat[8],dat[9],kps2,kps);
        lcd_puts(dat2);
        kompas();
        delay_ms(75);
        }
tengah();
kirik();
kanank();
ceklagi:
        while(te>=8)
        {
                if(dat[2]>=24 && dat[2]<=64)
                {
                lcd_clear();
                sprintf(dat2,"%d %d %d kps2=%d kps=%d",dat[2],dat[8],dat[9],kps2,kps);
                lcd_puts(dat2);
                tengah();
                kirik();
                kanank();
                //if(kik>=20) kik=20;
                if(kik<=10) kik=10;
                //if(kak>=20) kak=20;
                if(kak<=10) kak=10;
                PORTB.2=0;
                PORTB.1=0;
                OCR2=(25.5*kak)-255; //kiri
                OCR0=(25.5*kik)-255; //kanan
                if(kik>=20) OCR0=0xaf;
                if(kak>=20) OCR2=0xaf;
                }
                else goto loop1;
        }
OCR0=0x00; //kanan
OCR2=0x00; //kiri
tengah();
if(te>=8) goto ceklagi;
printf("RS \r");
tpa81();
delay_ms(100);
OCR0=0x00; //kanan
OCR2=0x00; //kiri
        for (servo=100;servo<140;servo++)
        {
         tpa81();
                if(temp1>=100 || temp2>=100 || temp3>=100 || temp4>=100 || temp5>=100 || temp6>=100  ) //temp ada
                {
                        for(z=0;z<100;z++)//bendera naikkan
                        {
                        PORTA.0=1;
                        delay_us(700);
                        PORTA.0=0;
                        delay_ms(19);
                        }
                        //angkat bayi
                        PORTA.5=1;
                        delay_ms(500);
                        DDRA.5=0;
                        PORTA.5=1;
                        while(PINA.5==0)
                        {
                        lcd_clear();
                        sprintf(dat2,"angkat bayi");
                        lcd_puts(dat2);
                        delay_ms(100);
                        OCR0=0x00; //kanan
```

```
                OCR2=0x00; //kiri
                }


            printf("sv 0 120\r");
            ada=1;
            //-----menghindari bayi----------------------
            tengah();
            kirik();
            kanank();
            while( te<25 || kik<12 || kak<12)
            {
            tengah();
            kirik();
            kanank();
            OCR0=0xcc;
            OCR2=0xcc;
            PORTB.2=1;//kanan
            PORTB.1=1;//kiri
            }
            OCR0=0xcc;
            OCR2=0xcc;
                if(ada==1 && ada2==1)
                {
                PORTB.2=0;//kanan
                PORTB.1=1;//kiri
                }
                else
                {
                PORTB.2=1;//kanan
                PORTB.1=0;//kiri
                }
                delay_ms(800);
            //---------------------------------------
            //---mendeteksi api setelah ada bayi
                if(PINA.2==1)
                {
                PORTB.2=0;//kanan
                PORTB.1=1;//kiri
                    while(PINA.3==0)
                    {
                    OCR0=0xcc;
                    OCR2=0xcc;
                    delay_ms(100);
                    OCR0=0x00;
                    OCR2=0x00;
                    delay_ms(100);
                    }
                    goto lanjut;
            //---------------------------------------
                }
                else goto loop1;
            }
        lcd_clear();
        sprintf(dat2,"srv=%d temp=%d %d %d %d %d %d %d %d" ,servo ,temp ,temp1 ,temp2 ,temp3 ,temp4
         ,temp5 ,temp6 ,temp6);
        lcd_puts(dat2);
        sprintf(text,"sv 0 %u\r",(servo));
        puts(text);
        delay_ms(200);
        }
printf("sv 0 120\r");
delay_ms(100);
printf("RM 3 \r");
delay_ms(100);
printf("PM 0 \r");
bayi();
```

```
            delay_ms(100);
            }
        }

//-----ikuti dinding kiri
        tengah();
        kiri();
        kanan();
        kirik();
        kanank();
        OCR0=0xcc; //kanan
        PORTB.2=0;//kanan
        PORTB.1=0;//kiri
        if(kik>=40) {kik=55;OCR0=0xee;}
        OCR2=(-5.1*kik)+306; //kiri
        lcd_clear();
        sprintf(dat2,"lnt2 kik=%d ocr2=%d %d %d %d %d %d",kik,OCR2,dat[8],dat[9],PINA.3,PINA.2,ada2,ada);
        lcd_puts(dat2);
        //program menghindar **********perlu revisi (bila bayi telah ditemukan cek api)===

            if(PINA.3==1 && PINA.2==1 && ada==0)
            {
                if(te<20 || kak<35)
                {
                OCR0=0x00;
                OCR2=0x00;
                        while(te<15 || kik<35 || kak<10)
                        {
                        tengah();
                        kirik();
                        kanank();
                        PORTB.2=1;//kanan
                        PORTB.1=0;//kiri
                        OCR0=0xff;
                        OCR2=0xff;
                        delay_ms(100);
                        }
                    }
            }
            if(te<8)
            {//putar kanan
                if(PINA.2==1 && ada==1) goto api;
                while(te<30 || kik<12)
                {
                OCR0=0xff;
                OCR2=0xff;
                PORTB.2=1;//kanan
                PORTB.1=0;//kiri
                tengah();
                kirik();
                }
            }
            if(kik<10 && kak>18)
            {//putar kanan
                while(kik<12)
                {
                OCR0=0xcc;
                OCR2=0xcc;
                PORTB.2=1;//kanan
                PORTB.1=0;//kiri
                kirik();
                }
            }
            if(kak<10)
            {
                if(kik>12)
                {//putar kiri dikit
```

```
                    while(kak<12)
                    {
                    OCR0=0xcc;
                    OCR2=0xcc;
                    PORTB.2=0;//kanan
                    PORTB.1=1;//kiri
                    kanank();
                    }
                }
                else
                {//putar kanan
                    while(kik<12)
                    {
                    OCR0=0xcc;
                    OCR2=0xcc;
                    PORTB.2=1;//kanan
                    PORTB.1=0;//kiri
                    kirik();
                    }
                }
            }


}

goto loop1;



api:    printf("sv 0 122\r");
        lcd_clear();
        sprintf(dat2,"api %d %d",PINA.2,PINA.3);
        lcd_puts(dat2);
        while(PINA.2==1)//global mendetksi api
        {
            if(PINA.3==0)
            {
            lcd_clear();
            sprintf(dat2,"cari api depan");
            lcd_puts(dat2);
                if(ada==1 && ada2==1)
                {
                PORTB.2=0;//kanan
                PORTB.1=1;//kiri
                }
                else
                {
                PORTB.2=1;//kanan
                PORTB.1=0;//kiri
                }
            OCR0=0xcc;
            OCR2=0xcc;
            delay_ms(100);
            OCR0=0x00;
            OCR2=0x00;
            delay_ms(50);
            }
            else goto lanjut; //dapat api ke label lanjut
        }
        goto loop1;


        lanjut:     lilin();
                    tengah();
                    while(te>23)//maju keapi dengan osilasi
                    {
                    tengah();
```

```
                lilin();
//                  if(dat[2]<=34 || dat[2]>=54 || dat[8]<=2 || dat[9]<=2)
//                  {
                    while(dat[2]<=24 || dat[2]>=64)
                    {
                        if(dat[2]<=64)
                        {
                        PORTB.2=1;//kanan
                        PORTB.1=0;//kiri
                        OCR0=0xcc;
                        OCR2=0xcc;
                        delay_ms(100);
                        OCR0=0x00;
                        OCR2=0x00;
                        }
                        if(dat[2]>=24)
                        {
                        PORTB.2=0;//kanan
                        PORTB.1=1;//kiri
                        OCR0=0xcc;
                        OCR2=0xcc;
                        delay_ms(100);
                        OCR0=0x00;
                        OCR2=0x00;
                        }
                    lilin();
                    }
//                  }
//                  else
//                  {
                tengah();
                kirik();
                kanank();
                PORTB.2=0;
                PORTB.1=0;
                if(kik<=10) kik=10;
                if(kak<=10) kak=10;
                OCR2=(25.5*kak)-255; //kiri
                OCR0=(25.5*kik)-255; //kanan
                if(kik>=20) OCR0=0xaf;
                if(kak>=20) OCR2=0xaf;
//                  }
            lcd_clear();
            sprintf(dat2,"%d %d %d te=%d ",dat[2],dat[8],dat[9],te);
            lcd_puts(dat2);
            }
        OCR0=0x00;
        OCR2=0x00;
        if( PINA.3==1 && PINA.2==1)
        {
        tpa81();
        printf("RS \r");
        delay_ms(200);
        for (servo=80;servo<160;servo++)
        {
        tpa81();
            if(temp>110||temp1>110||temp2>110||temp3>110||temp4>110||temp5>110||temp>110) //temp ada
            {
            kompas();
            kps2=kps;
            tengah();
            kirik();
            kanank();
                while(te<15 || kik<10 || kak<10) //mundur dikit
                {
                PORTB.2=1;//kanan
                PORTB.1=1;//kiri
```

```
                    OCR0=0xcc;
                    OCR2=0xcc;
                    tengah();
                    kirik();
                    kanank();
                    }
                    //balik badan
                    while(kps<=kps2-138 || kps>=kps2-128)//128
                    {
                    PORTB.2=0;//kanan
                    PORTB.1=1;//kiri
                    OCR0=0xff;
                    OCR2=0xff;
                    delay_ms(100);
                    OCR0=0x00;
                    OCR2=0x00;
                    delay_ms(100);
                    kompas();
                    lcd_clear();
                    sprintf(dat2,"kps2=%d kps=%d ",kps2,kps);
                    lcd_puts(dat2);
                    }
                    PORTA.1=1;
                    OCR0=0xcc;
                    OCR2=0xcc;
                    PORTB.2=1;//kanan
                    PORTB.1=0;//kiri
                    delay_ms(800);
                    while(PINA.2==1)
                    {
                    PORTA.1=1;
                    OCR0=0xcc;
                    OCR2=0xcc;
                    PORTB.2=0;//kanan
                    PORTB.1=1;//kiri
                    delay_ms(1000);
                    PORTB.2=1;//kanan
                    PORTB.1=0;//kiri
                    delay_ms(1000);
                    }
                    PORTA.1=0;
                    goto loop1;
                }
            sprintf(text,"sv 0 %u\r",(servo));
            puts(text);
            lcd_clear();
            sprintf(dat2,"srv=%d temp=%d %d %d %d %d %d %d %d", servo, temp, temp1, temp2, temp3, temp4, temp5,
            temp6, temp6);
            lcd_puts(dat2);
            delay_ms(100);
            }
            }
        goto api;
        pos=1;


 break;
 }

     };
}


char tangga(void)
{
printf("RS \r");
delay_ms(100);
```

```
printf("sv 0 120\r");
delay_ms(100);
printf("RM 3 \r");
delay_ms(100);
printf("PM 0 \r");
delay_ms(100);
printf("tc 20 35 46 70 20 55\r");
delay_ms(200);
}
char bayi(void)//00255
{
printf("L1 1\r");
delay_ms(100);
printf("L1 0\r");
delay_ms(100);
printf("tc 16 30 16 40 50 240 \r");//@#$%^&*percobaab
delay_ms(100);
}
char lilin(void)
{
printf("RS \r"); //setting kamera
delay_ms(100);
printf("sv 0 120\r");
delay_ms(100);
printf("RM 3 \r");
delay_ms(100);
printf("PM 0 \r");
delay_ms(100);
printf("tc 240 240 240 240 240 240  \r");
delay_ms(100);
}
char cek(void) //warna tangga luar dan dalam beda
{
c++;
lcd_clear();
sprintf(dat2,"c=%d ",c);
lcd_puts(dat2);
delay_ms(100);
if(c>=6)
{
      OCR0=0x00;
      OCR2=0x00;
      kiri();
      kanan();
      if((ki+ka)<=60 && ki<30 && ka<30)
      {
      kompas();
            if(kps>(ar-10) && kps<(ar+10))
            {
            lcd_clear();
            sprintf(dat2,"tangga");
            lcd_puts(dat2);
            delay_ms(1000);
            d++;
            }
      }
      if(d>=3) pos=1;
}


}
```

SUBPROGRAM PENGGUNAAN SENSOR - SENSOR

```
void kiri(void) // Sensor Ping Kiri
{
mulai:
     t=1;
     DDRD.6=1;
     PORTD.6=1;
     delay_us(5);
     PORTD.6=0;
     DDRD.6=0;
     PORTD.6=1 ;
       for(i=0;i<1050;i++)
       {
       if(PIND.6==1)
       goto coun;
       }
goto mulai;
coun:
       if(PIND.6==0)
       goto hitung;
       t=t+1 ;
       delay_us(1);
goto coun;
hitung:
       ki=460*0.0001*t;
return;
}

void kirik(void) // Sensor Ping Serong Kiri 45^0
{
mulai:
     t=1;
     DDRD.5=1;
     PORTD.5=1;
     delay_us(5);
     PORTD.5=0;
     DDRD.5=0;
     PORTD.5=1 ;
       for(i=0;i<1050;i++)
       {
       if(PIND.5==1)
       goto coun;
       }
goto mulai;
coun:
       if(PIND.5==0)
       goto hitung;
       t=t+1 ;
       delay_us(1);
goto coun;
hitung:
       kik=460*0.0001*t;
return;
}

void tengah(void) // Sensor Ping Tengah
{
```

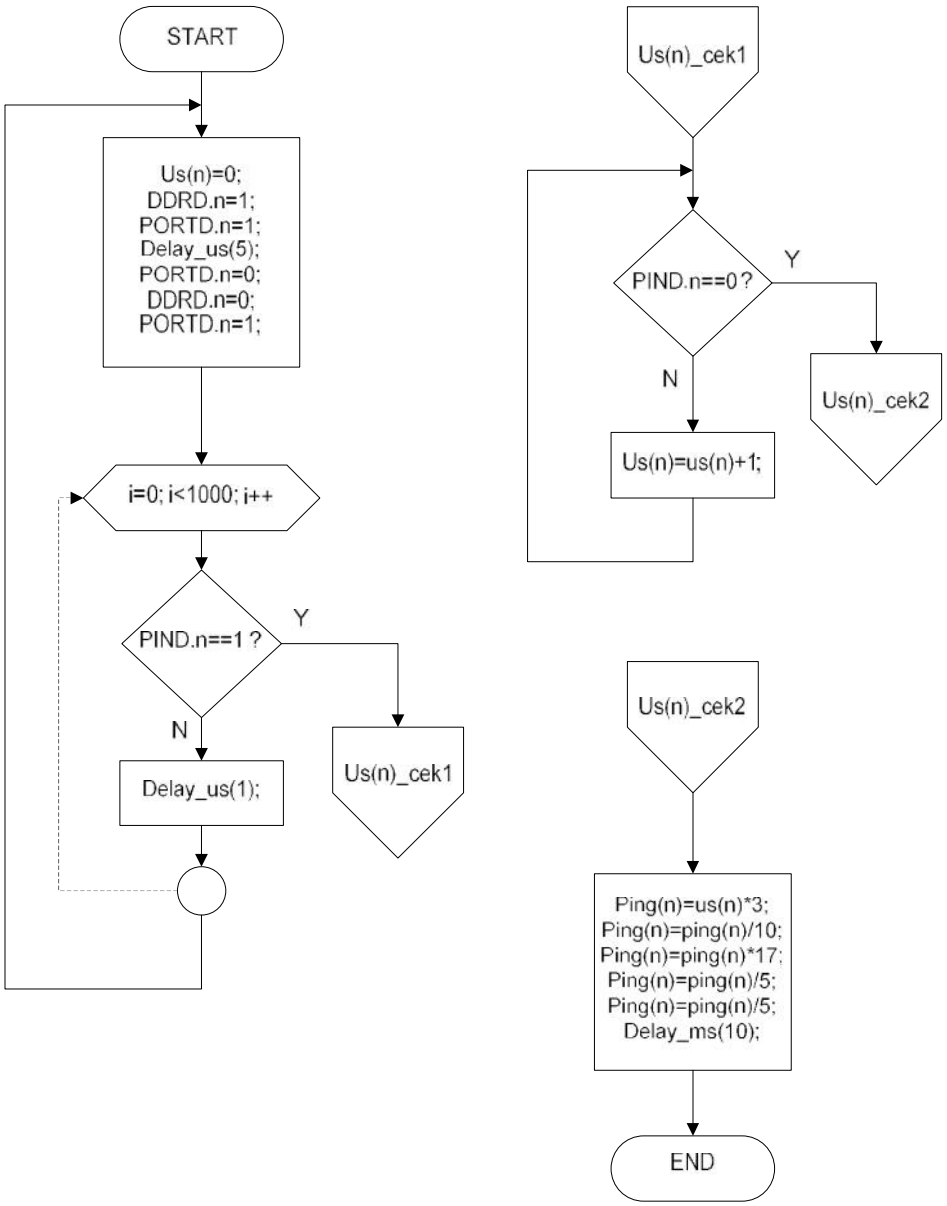
```
mulai:
     t=1;
     DDRD.4=1;
     PORTD.4=1;
     delay_us(5);
     PORTD.4=0;
     DDRD.4=0;
     PORTD.4=1 ;
       for(i=0;i<1050;i++)
       {
       if(PIND.4==1)
       goto coun;
       }
goto mulai;
coun:
       if(PIND.4==0)
       goto hitung;
       t=t+1 ;
       delay_us(1);
goto coun;
hitung:
       te=460*0.0001*t;
return;
}

void kanank(void) // Sensor Ping Serong Kanan 45^0
{
mulai:
     t=1;
     DDRD.3=1;
     PORTD.3=1;
     delay_us(5);
     PORTD.3=0;
     DDRD.3=0;
     PORTD.3=1 ;
       for(i=0;i<1050;i++)
       {
       if(PIND.3==1)
       goto coun;
       }
goto mulai;
coun:
       if(PIND.3==0)
       goto hitung;
       t=t+1 ;
       delay_us(1);
goto coun;
hitung:
       kak=460*0.0001*t;
return;
}

void kanan(void) // Sensor Ping Kanan
{
mulai:
     t=1;
     DDRD.2=1;
     PORTD.2=1;
     delay_us(5);
     PORTD.2=0;
     DDRD.2=0;
     PORTD.2=1 ;
       for(i=0;i<1050;i++)
       {
       if(PIND.2==1)
       goto coun;
       }
```

```
goto mulai;
coun:
      if(PIND.2==0)
      goto hitung;
      t=t+1 ;
      delay_us(1);
goto coun;
hitung:
      ka=460*0.0001*t;
return;
}

void atas()  //Sensor SRF02
{
i2c_start();
i2c_write(0xE0);
i2c_write(0x00);
i2c_write(0x51);
i2c_start();
i2c_write(0xE0);
i2c_write(0x02);
i2c_start();
i2c_write(0xE1);
atas1=i2c_read(1);
atas2=i2c_read(0);
ats=(atas1*256)+atas2;
i2c_stop();
}

void kompas() // Sensor CMPS03
{
i2c_start();
i2c_write(0xC0);
i2c_write(0x01);
i2c_start();
i2c_write(0xC1);
kps=i2c_read(0);
i2c_stop();
}

void tpa81() // Sensor Thermal Infrared
{
i2c_start();
i2c_write(0xD0);
i2c_write(0x01);
i2c_start();
i2c_write(0xD1);
temp=i2c_read(1);
temp1=i2c_read(1);
temp2=i2c_read(1);
temp3=i2c_read(1);
temp4=i2c_read(1);
temp5=i2c_read(1);
temp6=i2c_read(0);
i2c_stop();
}
```

LAMPIRAN C

DIAGRAM ALIR PROGRAM SENSOR-SENSOR

DIAGRAM ALIR PROGRAM SENSOR PING

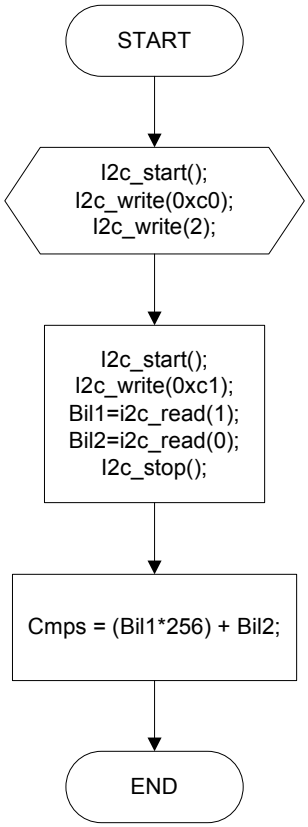


Keterangan :

n =2,3,4,5,6 (2 = Sensor Ping Kiri, 3 = Sensor Ping Serong Kiri 45^0, 4 = Sensor Ping Depan, 5 = Sensor Ping Serong Kanan 45^0, 6 = Sensor Ping Kanan)

i = variabel pengulangan.

DIAGRAM ALIR PROGRAM SENSOR CMPS03



Keterangan :

Bil1 = variabel yang berisi data register 2.

Bil2 = variabel yang berisi data register 3.

Cmps = data sudut arah mata angin dalam 1 *word*.

LAMPIRAN D
DATASHEET

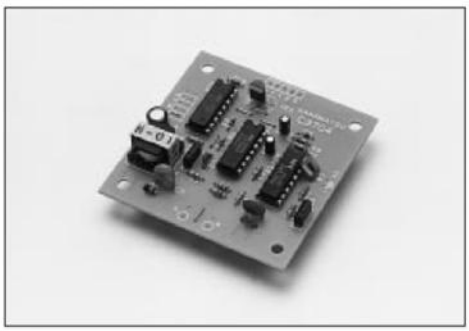
SENSOR API LILIN (HAMAMATSU UVTRON)

Compact, Lightweight, Low Current Consumption, Low Cost Operates as High Sensitivity UV Sensor with UV TRON Suitable for Flame Detectors and Fire Alarms

Hamamatsu C3704 series UV TRON driving circuits are low current consuming, signal processing circuits for the UV TRON, well known as a high sensitivity ultraviolet detecting tube. The C3704 series can be operated as a UV sensor by connecting the UV TRON and applying DC low voltage, as they have both a high-voltage power supply and a signal processing circuit on the same printed circuit board.

Since background discharges of the UV TRON caused by natural excitation lights (such as a cosmic ray, scattered sunlight, etc.) can be cancelled in the signal processing circuit, the output signals from the C3704 series can be used without errors.

When the high sensitivity sensor "UV TRON R2868" (sold separately) is used, the flame from a cigarette lighter (flame length: 25mm) can be detected even from a distance of more than 5m.



APPLICATIONS

- **Flame detectors for gas and oil lighters**
- **Fire alarms**
- **Combustion monitors for burners**
- **Electric spark detector**
- **UV photoelectric counter**

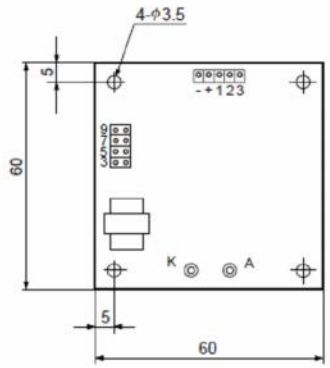
SPECIFICATIONS

Dimensional outline ..... Figure 1
Weight ..... Approx. 20g
Output signal ..... Open collector Output (50 V, 100 mA Max.)
10 ms width pulse output (Note : 1)
UV TRON supply voltage ..... DC 350 V (Note : 2)
Quenching time ..... Approx. 50 ms
Operating temperature ..... -10 to +50°C
(with no condensation)
Suitable UV TRON ..... Low voltage operation UV TRON
(such as R2868)

	C3704	C3704-02	C3704-03
Input Voltage	10 to 30 Vdc	5Vdc ± 5%	6 to 9 Vdc
Current consumption	3 mA Max.	300µA Max.	300µA Max.

Note 1: The output pulse width can be extended up to about 100s by adding a capacitor to the circuit board.
Note 2: Since the output impedance of this power supply is extremely high, an ordinary voltmeter cannot be used. Use a voltmeter that has an input impedance of more than 10 GΩ.

Figure 1: Dimensional Outline (Unit : mm)



<Top View>

<Side View>

TPT A002HEA

Figure 2: Schematic Diagram

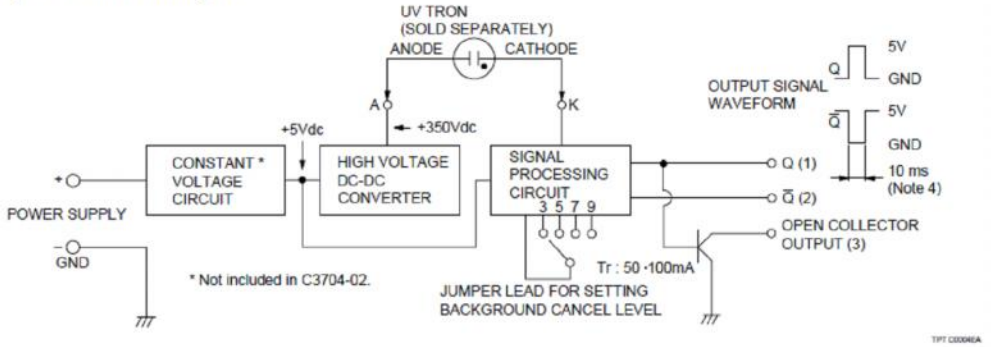


Figure 3: Method of Connection

POWER SUPPLY INPUT AND SIGNAL OUTPUT POINT
– : GND
+ : POWER SUPPLY INPUT
1 : SIGNAL OUTPUT Q
2 : SIGNAL OUTPUT Q̄ (Note 3)
3 : OPEN COLLECTOR OUTPUT

JUMPER LEAD FOR SETTING
BACKGROUND CANCEL LEVEL

Cx : POINT FOR ADDING A CAPACITOR TO
EXTEND OUTPUT PULSE WIDTH (Note 4)

This background cancel circuit outputs signal pulses of 10ms width only when 3 to 9 pulses enter the circuit with a time interval of 2 seconds or less from the UV TRON. Cancel level can be set in 4 steps (number of pulses: 3, 5, 7, 9) by this jumper lead. Connect to "3" for general use. When using where much natural excitation light is present, set the cancel level at a higher step (5, 7, 9).

CONNECTING POINT OF UV TRON
A : Connecting Anode of UV TRON
K : Connecting Cathode of UV TRON

Note 3: No load can be driven by an output from points "1" and "2" because these signals are output from the only C-MOS IC directly. When a load such as a buzzer and a relay is connected to this circuit, it should be connected to the point open collector output. The transistor ratings of the open collector is 50V, 100mA. Be careful not to exceed the ratings.

Note 4: The output pulse width is set to 10ms at shipping. If the pulse width needs to be extended, add a capacitor to this point. (When using an electrolytic condenser, make sure the polarity is correct.)
e.g. CX = 1 μF: Pulse Width ≃ 1s, CX=10 μF : Pulse Width ≃ 10s

PRECAUTIONS FOR USE

- Since the operation impedance is extremely high, the UV TRON should be connected as close as possible to the circuit board within 5 cm.
- Take care to avoid external noise since a C-MOS IC is used in the circuit. It is recommended that the whole PC board be put in the shield box when it is used.
- To reduce current consumption, oscillating frequency is very low (approx. 20 Hz) in this DC-DC converter. Thus, the output impedance of the high voltage power supply is extremely high. If the surrounding humidity is high, electrical leakage on the PC board surface may lead to a drop in the supply voltage to the UV TRON. This voltage drop may result in lowered detection performance, so a moistureproof material (silicone compound, etc.) should be applied at the connecting point of the UV TRON, etc., if using the unit in a humid environment.

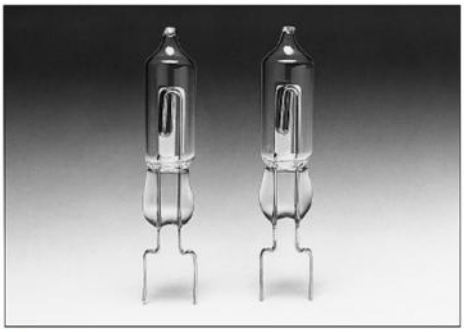
- A model equipped with a flame sensor (R2868) is also available.

Quick Detection of Flame from Distance, Compact UV Sensor with High Sensitivity and Wide Directivity, Suitable for Flame Detectors and Fire Alarms.

Hamamatsu R2868 is a UV TRON ultraviolet detector that makes use of the photoelectric effect of metal and the gas multiplication effect. It has a narrow spectral sensitivity of 185 to 260 nm, being completely insensitive to visible light. Unlike semiconductor detectors, it does not require optical visible-cut filters, thus making it easy to use.

In spite of its small size, the R2868 has wide angular sensitivity (directivity) and can reliably and quickly detect weak ultraviolet radiations emitted from flame due to use of the metal plate cathode (eg. it can detect the flame of a cigarette lighter at a distance of more than 5 m.).

The R2868 is well suited for use in flame detectors and fire alarms, and also in detection of invisible discharge phenomena such as corona discharge of high-voltage transmission lines.



APPLICATIONS

- **Flame detectors for gas/oil lighters and matches**
- **Fire alarms**
- **Combustion monitors for burners**
- **Inspection of ultraviolet leakage**
- **Detection of discharge**
- **Ultraviolet switching**

GENERAL

Parameters	Rating	Units
Spectral Response	185 to 260	nm
Window Material	UV glass	—
Weight	Approx. 1.5	g
Dimensional Outline	See Fig. 3	—

MAXIMUM RATINGS

Parameters	Rating	Units
Supply Voltage	400	Vdc
Peak Current [1]	30	mA
Average Discharge Current [2]	1	mA
Operating Temperature	-20 to +60	°C

CHARACTERISTICS (at 25°C)

Parameters	Rating	Units
Discharge Starting Voltage (with UV radiation)	280	Vdc Max.
Recommended Operating Voltage	325±25	Vdc
Recommended Average Discharge Current	100	µA
Background [3]	10	cpm Max
Sensitivity [4]	5000	cpm Typ.

NOTES:

1) This is the maximum momentary current that can be handled if its full width at half maximum is less than 10 µs.
2) If the tube is operated near this or higher, the service life is noticeably reduced. Use the tube within the recommended current values.
3) Measured under room illuminations (approximately 500 lux) and recommended operating conditions. Note that these values may increase if the following environmental factors are present.
   1. Mercury lamps, sterilization lamps, or halogen lamps are located nearby.
   2. Direct or reflected sunlight is incident on the tube.
   3. Electrical sparks such as welding sparks are present.
   4. Radiation sources are present.
   5. High electric field (including static field) generates across the tube.
4) These are representative values for a wavelength of 200 nm and a light input of 10 pW/cm². In actual use, the sensitivity will vary with the wavelength of the ultraviolet radiation and the drive circuitry employed.

Figure 1: UV TRON's Spectral Response and Various Light Sources

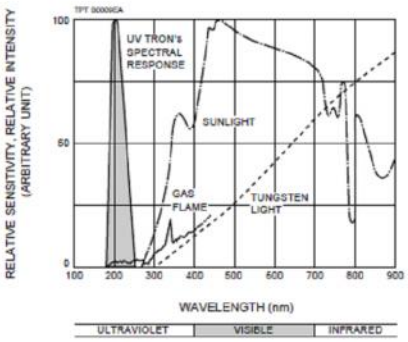


Figure 2: Angular Sensitivity (Directivity)

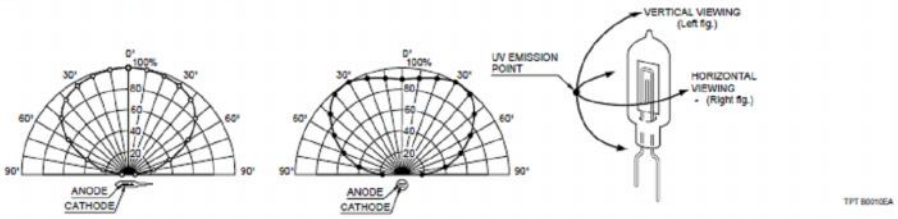


Figure 3: Dimensional Outline (Unit: mm)

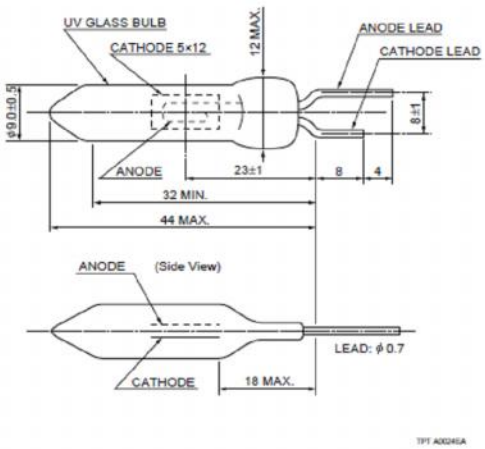
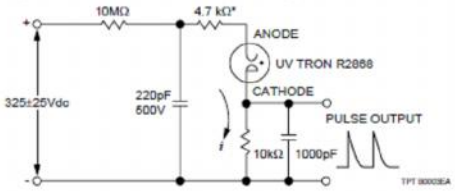
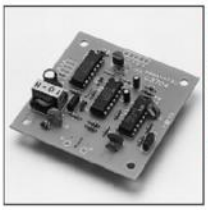


Figure 4: Recommended Operating Circuit



* Be sure to connect the 4.7 kΩ resistor within 2.5 cm from the anode lead end of UV TRON.

- UV TRON Driving Circuit C3704 series (Option)



Hamamatsu also provide the driving circuit C3704 series for R2868 operation.
C3704 series include a high voltage power supply and a signal processing circuit in printed circuit board, which allows to operate R2868 easily as a flame sensor with the low input voltage (DC 6 to 30 V) only.
For the details, please refer to the datasheet of C3704 series.

PRECAUTIONS FOR USE

- **Ultraviolet Radiation**

The UV TRON itself emits ultraviolet radiation in operation. When using two or more UV TRONs at the same time in close position, care should be taken so that they do not optically interfere with each other.

- **Vibration and Shock**

The UV TRON is designed in accordance with the standards of MIL-STD-202F (Method 204D/0.06 inch or 10g, 10- 500Hz, 15 minutes, 1 cycle) and MIL-STD-202F (Method 213B/100g, 11ms, Half-sine, 3 times). However, should a strong shock be sustained by the UV TRON (e.g. if dropped), the glass bulb may crack or the internal electrode may be deformed, resulting in deterioration of electrical characteristics. So extreme care should be taken in handling the tube.

- **Polarity**

Connect the UV TRON with correct polarity. Should it be connected with reverse polarity, operating errors may occur.

WARRANTY.

The UV TRON is covered by a warranty for a period of one year after delivery. The warranty is limited to replacement of any defective tube due to defects traceable to the manufacturer.

SENSOR JARAK ULTRASONIK (PING)

PING)))™ Ultrasonic Distance Sensor (#28015)

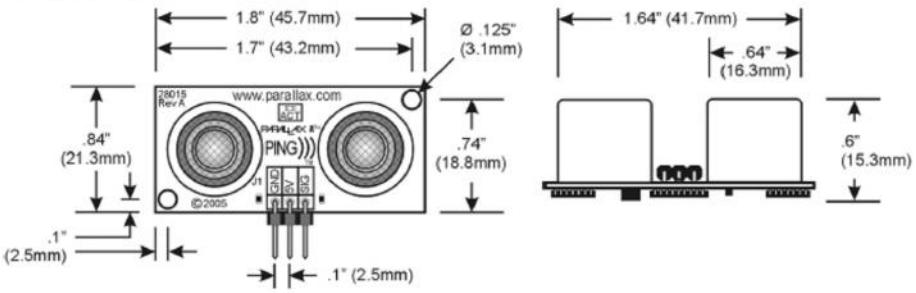
The Parallax PING))) ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to BASIC Stamp® or Javelin Stamp modules, SX or Propeller chips, or other microcontrollers, requiring only one I/O pin.

The PING))) sensor works by transmitting an ultrasonic (well above human hearing range) burst and providing an output pulse that corresponds to the time required for the burst echo to return to the sensor. By measuring the echo pulse width, the distance to target can easily be calculated.

Features

- Supply voltage – +5 VDC
- Supply current – 30 mA typ; 35 mA max
- Range – 2 cm to 3 m (0.8 inches to 3.3 yards)
- Input trigger – positive TTL pulse, 2 µs min, 5 µs typ.
- Echo pulse – positive TTL pulse, 115 µs to 18.5 ms
- Echo hold-off – 750 µs from fall of trigger pulse
- Burst frequency – 40 kHz for 200 µs
- Burst indicator LED shows sensor activity
- Delay before next measurement – 200 µs
- Size – 22 mm H x 46 mm W x 16 mm D (0.84 in x 1.8 in x 0.6 in)
- Operating temperature: 0 – 70° C.

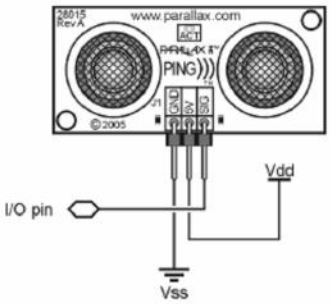
Dimensions



Pin Definitions

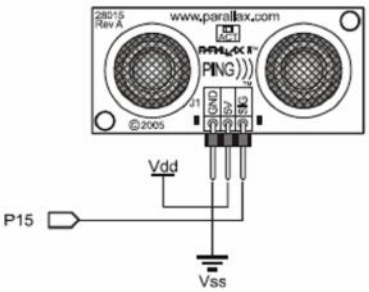
GND	Ground (Vss)
5 V	5 VDC (Vdd)
SIG	Signal (I/O pin)

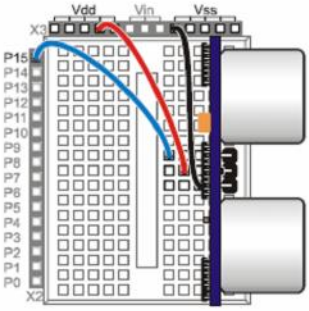
The PING))) sensor has a male 3-pin header used to supply ground, power (5 VDC) and signal. The header allows the sensor to be plugged into a solderless breadboard, or to be located remotely through the use of a standard servo extender cable (Parallax part #805-00002). Standard connections are shown in the diagram to the right.



Quick-Start Circuit

This circuit allows you to quickly connect your PING))) sensor to a BASIC Stamp® 2 via the Board of Education® breadboard area. The PING))) module's GND pin connects to Vss, the 5 V pin connects to Vdd, and the SIG pin connects to I/O pin P15. This circuit will work with the example BASIC Stamp program listed below.

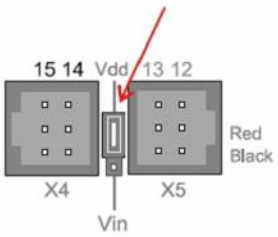




Extension Cable and Port Cautions

If you want to connect your PING))) sensor to a Board of Education platform using an extension cable, follow these steps:

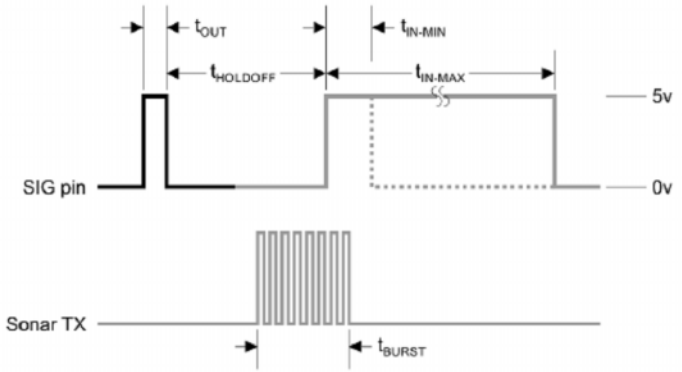
1. When plugging the cable onto the PING))) sensor, connect Black to GND, Red to 5 V, and White to SIG.
2. Check to see if your Board of Education servo ports have a jumper, as shown at right.
3. If your Board of Education servo ports have a jumper, set it to Vdd as shown. Then plug the cable into the port, matching the wire color to the labels next to the port.
4. If your Board of Education servo ports do not have a jumper, **do not use them with the PING))) sensor**. These ports only provide Vin, not Vdd, and this may damage your PING))) sensor. Go to the next step.
5. Connect the cable directly to the breadboard with a 3-pin header as shown above. Then, use jumper wires to connect Black to Vss, Red to Vdd, and White to I/O pin P15.



Board of Education Servo Port Jumper, Set to Vdd

Theory of Operation

The PING))) sensor detects objects by emitting a short ultrasonic burst and then "listening" for the echo. Under control of a host microcontroller (trigger pulse), the sensor emits a short 40 kHz (ultrasonic) burst. This burst travels through the air at about 1130 feet per second, hits an object and then bounces back to the sensor. The PING))) sensor provides an output pulse to the host that will terminate when the echo is detected, hence the width of this pulse corresponds to the distance to the target.



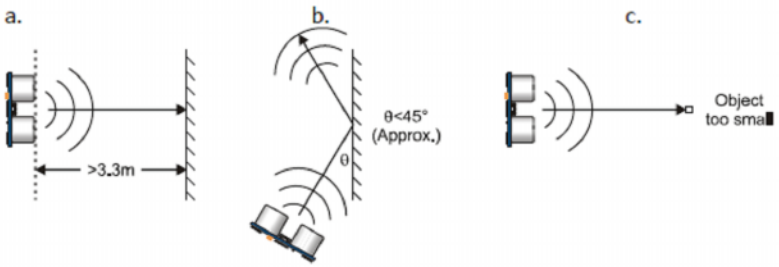
Host Device
PING))) Sensor

t_{OUT}	2 µs (min), 5 µs typical
$t_{HOLDOFF}$	750 µs
t_{BURST}	200 µs @ 40 kHz
t_{IN-MAX}	115 µs
t_{IN-MIN}	18.5 µs

Practical Considerations for Use

Object Positioning

The PING))) sensor cannot accurately measure the distance to an object that: a) is more than 3 meters away, b) that has its reflective surface at a shallow angle so that sound will not be reflected back towards the sensor, or c) is too small to reflect enough sound back to the sensor. In addition, if your PING))) sensor is mounted low on your device, you may detect sound reflecting off of the floor.



Target Object Material

In addition, objects that absorb sound or have a soft or irregular surface, such as a stuffed animal, may not reflect enough sound to be detected accurately. The PING))) sensor will detect the surface of water, however it is not rated for outdoor use or continual use in a wet environment. Condensation on its transducers may affect performance and lifespan of the device. See the Water Level with PING))) document on the 28015 product page.

Air Temperature

Temperature has an effect on the speed of sound in air that is measurable by the PING))) sensor. If the temperature (°C) is known, the formula is:

$$C_{air} = 331.5 + (0.6 \times T_C)\ m/s$$

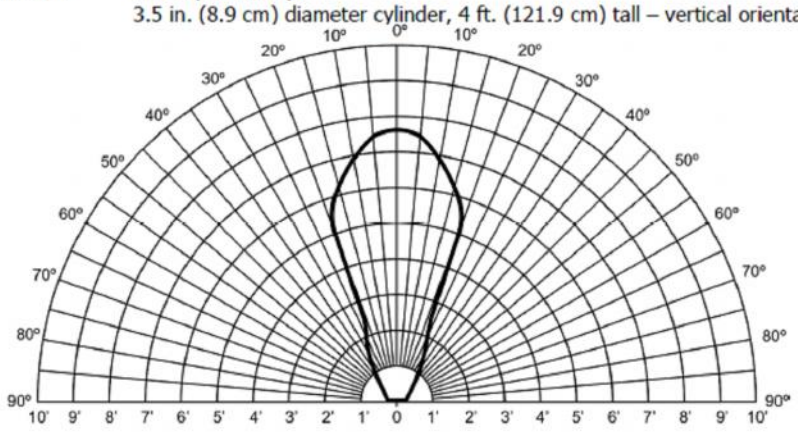
The percent error over the sensor's operating range of 0 to 70 ° C is significant, in the magnitude of 11 to 12 percent. The use of conversion constants to account for air temperature may be incorporated into your program (as is the case in the BS2 program below). Percent error and conversion constant calculations are introduced in Chapter 2 of *Smart Sensors and Applications*, a Stamps in Class text available for download from the 28029 product page.

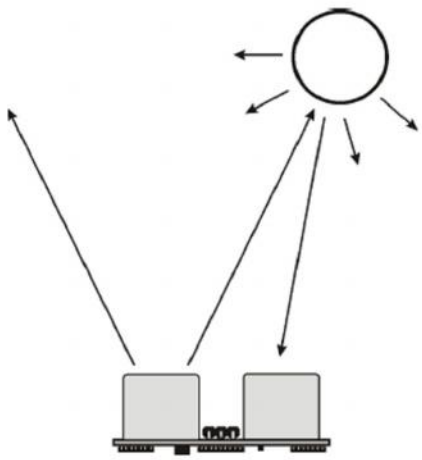
Test Data

The test data on the following pages is based on the PING))) sensor, tested in the Parallax lab, while connected to a BASIC Stamp microcontroller module. The test surface was a linoleum floor, so the sensor was elevated to minimize floor reflections in the data. All tests were conducted at room temperature, indoors, in a protected environment. The target was always centered at the same elevation as the PING))) sensor.

Test 1

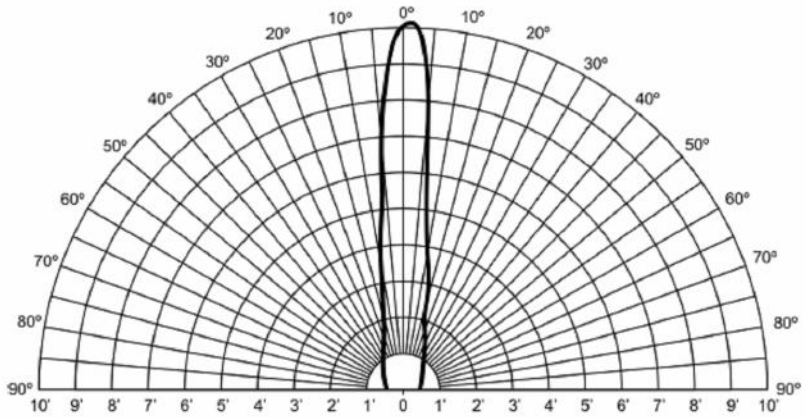
Sensor Elevation: 40 in. (101.6 cm)
Target: 3.5 in. (8.9 cm) diameter cylinder, 4 ft. (121.9 cm) tall – vertical orientation

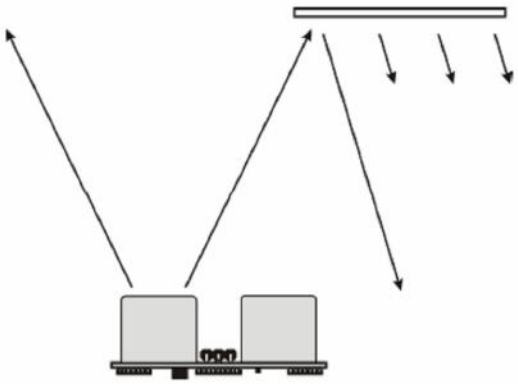




Test 2

Sensor Elevation: 40 in. (101.6 cm)
Target: 12 in. x 12 in. (30.5 cm x 30.5 cm) cardboard, mounted on 1 in. (2.5 cm) pole
Target positioned parallel to backplane of sensor





SENSOR THERMAL INFRARED

(THERMAL ARRAY TPA 81)

TPA81 Thermopile Array

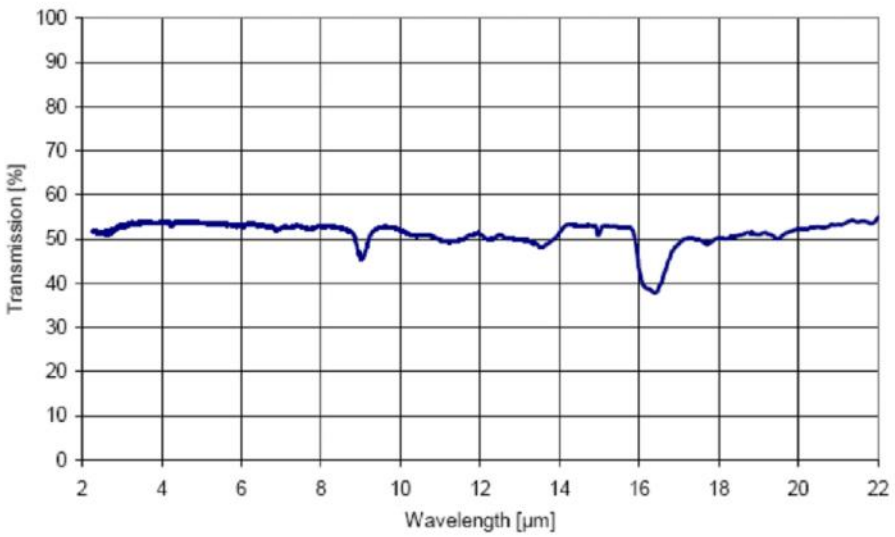
Technical Specification

Introduction

The TPA81 is a thermopile array detecting infra-red in the 2um-22um range. This is the wavelength of radiant heat. The Pyro-electric sensors that are used commonly in burglar alarms and to switch on outside lights, detect infra-red in the same waveband. These Pyro-electric sensors can only detect a change in heat levels though - hence they are movement detectors. Although useful in robotics, their applications are limited as they are unable to detect and measure the temperature of a static heat source. Another type of sensor is the thermopile array. These are used in non-contact infra-red thermometers. They have a very wide detection angle or field of view (FOV) of around 100° and need either shrouding or a lens or commonly both to get a more useful FOV of around 12°. Some have a built in lens. More recently sensors with an array of thermopiles, built in electronics and a silicon lens have become available. This is the type used in the TPA81. It has a array of eight thermopiles arranged in a row. The TPA81 can measure the temperature of 8 adjacent points simultaneously. The TPA81 can also control a servo to pan the module and build up a thermal image. The TPA81 can detect a candle flame at a range 2 metres (6ft) and is unaffected by ambient light!

Spectral Response

The response of the TPA81 is typically 2µm to 22µm and is shown below:



Field of View (FOV)

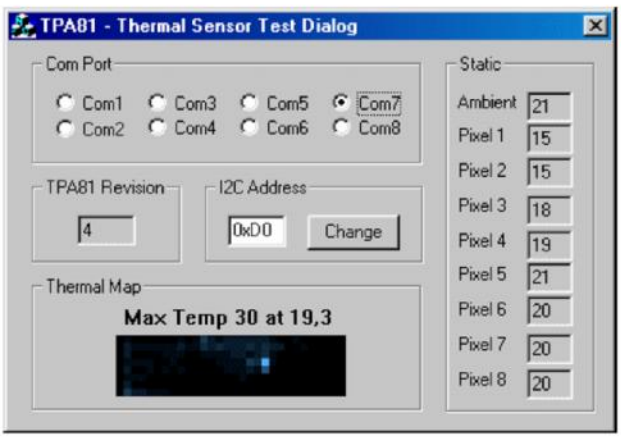
The typical field of view of the TPA81 is 41° by 6° making each of the eight pixels 5.12° by 6°. The array of eight pixels is orientated along the length of the PCB - that's from top to bottom in the diagram below. Pixel number one is nearest the tab on the sensor - or at the bottom in the diagram below.

Sensitivity

Here's some numbers from one of our test modules:
For a candle, the numbers for each of the eight pixels at a range of 1 meter in a cool room at 12°C are:
11 10 11 12 12 29 15 13 (All °C)
You can see the candle showing up as the 29°C reading. At a range of 2 meters this reduces to 20°C - still around 8° C above ambient and easily
detectable. At 0.6 meter (2ft) its around 64°C. At 0.3 meter (1ft) its 100°C+.

In a warmer room at 18°C, the candle measures 27°C at 2 meters. This is because the candle only occupies a small part of the sensors field of view and the candles point heat source is
added to the back ground ambient - not swamped by it. A human at 2 meters will show up as around 29°C with a background 20°C ambient.

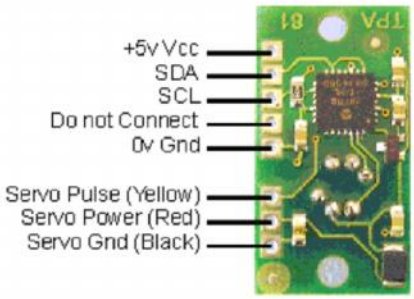
The following is a snapshot of our test program. It displays a 32x8 bitmap produced by using a servo to pan the sensor. If you want a copy of this windows based program, its here, but you will need an RF04/CM02 to connect the TPA81 to your PC. Here you can see a candle flame about a meter away showing up as the bright spot.



Connections

All communication with the TPA81 is via the I2C bus. If you are unfamiliar with the I2C bus, there is a tutorial which will help. The TPA81 uses our standard I2C 5 pin connection layout. The "Do Not Connect" pin should be left unconnected. It is actually the CPU MCLR line and is used once only in our workshop to program the PIC16F88 on-board after assembly, and has an internal pull-up resistor. The SCL and SDA lines should each have a pull-up resistor to +5v somewhere on the I2C bus. You only need one pair of resistors, not a pair for every module. They are normally located with the bus master rather than the slaves. The TPA81 is always a slave - never a bus master. If you need them, I recommend 1.8k resistors. Some modules such as the OOPic already have pull-up

resistors and you do not need to add any more. A servo port will connect directly to a standard RC servo and is powered from the modules 5v supply. We use an HS311. Commands can be sent to the TPA81 to position the servo, the servo pulses are generated by the TPA81 module.



Registers

The TPA81 appears as a set of 10 registers.

Register	Read	Write
0	Software Revision	Command Register
1	Ambient Temperature °C	Servo Range (V6 or higher only)
2	Pixel 1 Temperature °C	N/A
3	Pixel 2	N/A
4	Pixel 3	N/A
5	Pixel 4	N/A
6	Pixel 5	N/A
7	Pixel 6	N/A
8	Pixel 7	N/A
9	Pixel 8	N/A

Only registers 0, and 1 can be written to. Register 0 is the command register and is used to set the servo position and also when changing the TPA81's I2C address. It cannot be read. Reading from register 0 returns the TPA81 software revision. Writing to register 1 sets the servo range - see below. It cannot be read back, reading register 1 reads the ambient temperature.

There are 9 temperature readings available, all in degrees centigrade (°C). Register 1 is the ambient temperature as measured within the sensor. Registers 2-9 are the 8 pixel temperatures. Temperature acquisition is continuously performed and the readings will be correct approx 40mS after the sensor points to a new position.

Servo Position

Commands 0 to 31 set the servo position. There are 32 steps (0-31) which typically represent 180° rotation on a Hitec HS311 servo. The calculation is SERVO_POS*60+540uS. So the range of the servo pulse is 0.54mS to

2.4mS in 60uS steps. Writing any other value to the command register will stop the servo pulses.

Command		Action
Decimal	Hex	
0	0x00	Set servo position to minimum
nn	nn	Set servo position
31	0x1F	Set servo position to maximum
160	0xA0	1st in sequence to change I2C address
165	0xA5	3rd in sequence to change I2C address
170	0xAA	2nd in sequence to change I2C address

Firmware Version 6 or higher.

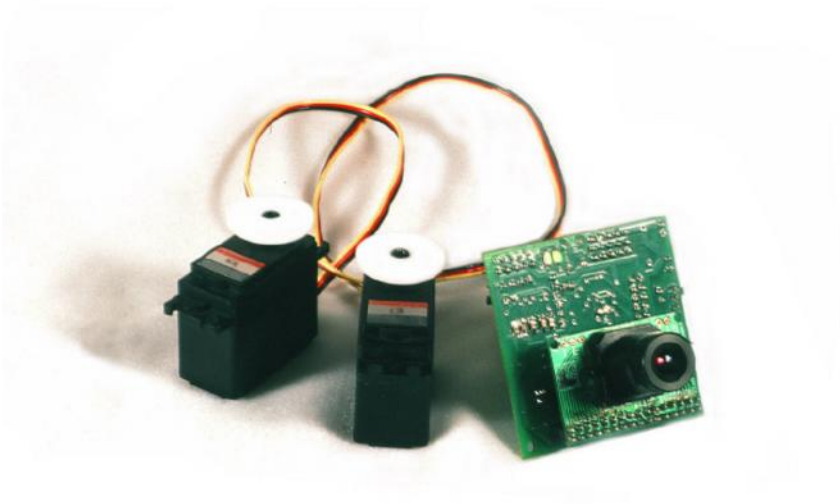
As from Version 6 (March 2005) we have added a new write only register (register 1) to allow you to vary the range of the servo stepping. It defaults to the same 180 degree range on a Hitec HS311 servo as earlier versions. You can write values from 20 to 120 to the range register. If you attempt to write a value less than 20, it will be set to 20. If you attempt to write a value greater than 120, it will be set to 120. The calculation for the range in uS is ((31*ServoRange)/2). Setting a range of 20 will give a range of (31*20)/2 or 310uS. Setting a range of 120 will give a range of (31*120)/2 or 1860uS. In all cases the available range is centered on the servo's mid position of 1500uS. So in the first example, the 310uS range will be from 1345uS to 1655uS (Or 1.345 to 1.655mS if you prefer). The second example of 1860uS range centered on 1500uS gives a range of 570uS to 2430uS. On power up the range register is set to 120, which give the same range as earlier versions.

Changing the I2C Bus Address

To change the I2C address of the TPA81 you must have only one module on the bus. Write the 3 sequence commands in the correct order followed by the address. Example; to change the address of a TPA81 currently at 0xD0 (the default shipped address) to 0xD2, write the following to address 0xD0; (0xA0, 0xAA, 0xA5, 0xD2). These commands must be sent in the correct sequence to change the I2C address, additionally, No other command may be issued in the middle of the sequence. The sequence must be sent to the command register at location 0, which means 4 separate write transactions on the I2C bus. Additionally, there MUST be a delay of at least 50mS between the writing of each byte of the address change sequence. When done, you should label the sensor with its address, if you lose track of the module addresses, the only way to find out what it is to search all the addresses one at a time and see which one responds. The TPA81 can be set to any of eight I2C addresses - 0xD0, 0xD2, 0xD4, 0xD6, 0xD8, 0xDA, 0xDC, 0xDE. The factory default shipped address is 0xD0.

SENSOR KAMERA(CMUCam2+)

CMUcam2 Vision Sensor

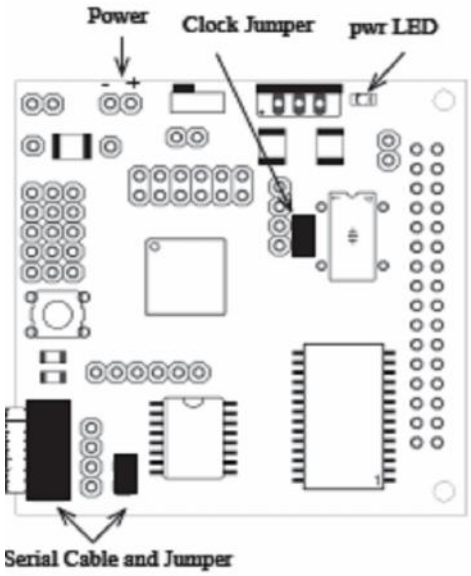


User Guide

Getting Started

Setting Up the Hardware

In order to initially test your CMUcam2, you will need a serial cable, a power adapter and a computer. The CMUcam2 can use a power supply which produces anywhere from 6 to 15 volts of DC power capable of supplying at least 200mA of current. This can be provided by either an AC adapter (possibly included) or a battery supply. These should be available at any local electronics store. The serial cable should have been provided with your CMUcam2. Make sure that you have the CMOS sensor board connected to the CMUcam2 board so that it is in the same orientation as the picture shows on the cover of this manual.



First, connect the power. Make sure that the positive side of your power plug is facing away from the main components on the board. If the camera came with an AC adapter, make sure that the connector locks into the socket correctly.

Now that the camera has power, connect the serial link between the camera and your computer. This link is required initially so that you can test and focus your camera. The serial cable should be connected so that the ribbon part of the cable faces away from the board. You must also connect the serial pass through jumper.

Check to make sure that the clock jumper is connected. This allows the clock to actively drive the processor.

Once everything is wired up, try turning the board on. The power LED should illuminate green and only one LED should remain on. Both LEDs turn on upon startup, and one turns off after the camera has been sucessfully configured.

Testing the Firmware

Once you have set the board up and downloaded the firmware, a good way to test the system is to connect it to the serial port of a computer.

Step 1: If one does not already exist, build a serial and/or power cable

Step 2: Plug both of them in.

Step 3: Open the terminal emulator of your choice.

Step 4: Inside the terminal emulator set the communication protocol to 115,200 Baud, 8 Data bits, 1 Stop bit, no parity, local echo on, no flow control and if possible turn on "add line feed" (add \n to a received \r). These setting should usually appear under "serial port" or some other similar menu option.

Step 5: Turn on the CMUcam2 board; the Power LED should light up and only one of the two status LEDs should remain on.

Step 6: You should see the following on your terminal emulator:

```
CMUcam2 v1.0 c6
:
```

If you have seen this, the board was able to successfully configure the camera and start the firmware.

Step 7: Type gv followed by the enter key. You should see the following:

```
:gv
ACK
CMUcam2 v1.0 c6
:
```

This shows the current version of the firmware. If this is successful, your computer's serial port is also configured correctly and both transmit and receive are working.

Serial Commands

The serial communication parameters are as follows:

- 1,200 to 115,200 Baud
- 8 Data bits
- 1 Stop bit
- No Parity
- No Flow Control (Not Xon/Xoff or Hardware)

All commands are sent using visible ASCII characters (123 is 3 bytes "123"). Upon a successful transmission of a command, the ACK string should be returned by the system. If there was a problem in the syntax of the transmission, or if a detectable transfer error occurred, a NCK string is returned. After either an ACK or a NCK, a \r is returned. When a prompt ('\r' followed by a ':') is returned, it means that the camera is waiting for another command in the idle state. White spaces do matter and are used to separate argument parameters. The \r (ASCII 13 carriage return) is used to end each line and activate each-command. If visible character transmission causes too much overhead, it is possible to use varying degrees of raw data transfer.

Alphabetical Command Listing

\r

This command is used to set the camera board into an idle state. Like all other commands, you should receive the acknowledgment string "ACK" or the not acknowledge string "NCK" on failure. After acknowledging the idle command the camera board waits for further commands, which is shown by the ':' prompt. While in this idle state a \r by itself will return an "ACK" followed by \r and : character prompt. This is how you stop the camera while in streaming mode.

Example of how to check if the camera is alive while in the idle state:

```
:
ACK
:
```

BM active \r

This command sets the mode of the CMUcam's frame buffer. A value of 0 (default) means that new frames are constantly being pushed into the frame buffer. A value of 1, means that only a single frame remains in the frame buffer. This allows multiple processing calls to be applied to the same frame. Instead of grabbing a new frame, all commands are applied to the current frame in memory. So you could get a histogram on all three channels of the same image and then track a color or call get mean and have these process a single buffered frame. Calling **RF** will then read a new frame into the buffer from the camera. When **BM** is off, **RF** is not required to get new frames.

Example of how to track multiple colors using buffer mode:

```
:BM 1
ACK
:PM 1
ACK
:TC 200 240 0 30 0 30
ACK
T 20 40 10 30 30 50 20 30
:RF
ACK
:TC 0 30 200 240 0 30
ACK
T 30 50 20 40 40 60 22 31
```

CR [reg1 value1 [reg2 value2 ... reg16 value16]]\r

This command sets the Camera's internal Register values directly. The register locations and possible settings can be found in the Omnivision CMOS camera documentation. All the data sent to this command should be in decimal visible character form unless the camera has previously been set into raw mode. It is possible to send up to 16 register-value combinations. Previous register settings are not reset between CR calls; however, you may overwrite previous settings. Calling this command with no arguments resets the camera and restores the camera registers to their default state. This command can be used to hard code gain values or manipulate other low level image properties.

Register		Value	Effect
5	Contrast	0-255	
6	Brightness	0-255	
18	Color Mode		
		36	YCrCb Auto White Balance On
		32	YCrCb Auto White Balance Off
		44	RGB Auto White Balance On
		40	*RGB Auto White Balance Off
17	Clock Speed		
		0	*50 fps
		1	26 fps
		2	17 fps
		3	13 fps
		4	11 fps
		5	9 fps
		6	8 fps
		7	7 fps
		8	6 fps
		10	5 fps
19	Auto Exposure		
		32	Auto gain off
		33	*Auto gain on

* indicates the default state

Example of switching into YCrCb mode with White Balance off

```
:CR 18 32
ACK
:
```

CP boolean \r

This command toggles the **C**amera module's **P**ower. A value of 0, puts the camera module into a power down mode. A value of 1 turns the camera back on while maintaining the current camera register values. This should be used in situations where battery life needs to be extended, while the camera is not actively processing image data. Images in the frame buffer may become corrupt when the camera is powered down.

CT boolean \r

This command toggles the **C**amera **T**ype while the camera is in slave mode. Since the CMUcam2 can not determine the type of the camera without communicating with the module, it is not possible for it to auto-detect the camera type in slave mode. A value of 0, sets the CMUcam2 into ov6620 mode. A value of 1 sets it into ov7620 mode. The default slave mode startup value assumes the ov6620.

DC value \r

This command sets the **C**hannel that is used for frame **D**ifferencing commands. A value of 0, sets the frame differencing commands LF and FD to use the red (Cr) channel. A value of 1 (default) sets them to use the green (Y) channel, and 2 sets them to use the blue (Cb) channel.

DM value \r

This command sets the **D**elay **M**ode which controls the delay between characters that are transmitted over the serial port. This can give slower processors the time they need to handle serial data. The value should be set between 0 and 255. A value of 0 (default) has no delay and 255 sets the maximum delay. Each delay unit is equal to the transfer time of one bit at the current baud rate.

DS x_factor y_factor \r

This command allows **D**own **S**ampling of the image being processed. An x_factor of 1 (default) means that there is no change in horizontal resolution. An x_factor of 2, means that the horizontal resolution is effectively halved. So all commands, like send frame and track color, will operate at this lower down sampled resolution. This gives you some speed increase and reduces the amount of data sent in the send frame and bitmap linemodes without clipping the image like virtual windowing would. Similarly, the y_factor independently controls the vertical resolution. (Increasing the y_factor downsampling gives more of a speed increase than changing the x_factor.) The virtual window is reset to the full size whenever this command is called.

Example of down sampling the resolution by a factor of 2 on both the horizontal and vertical dimension.

```
:DS 2 2
ACK
:GM
ACK
S 89 90 67 5 6 3
S 89 91 67 5 6 2
```

FD threshold \r

This command calls **Frame Differencing** against the last loaded frame using the **LF** command. It returns a type T packet containing the middle mass, bounding box, pixel count and confidence of any change since the previously loaded frame. It does this by calculating the average color intensity of an 8x8 grid of 64 regions on the image and comparing those plus or minus the user assigned *threshold*. So the larger the threshold, the less sensitive the camera will be towards differences in the image. Usually values between 5 and 20 yield good results. (In high resolution mode a 16x16 grid is used with 256 regions.)

FS boolean \r

This command sets the **Frame Streaming** mode of the camera. A value of 1, enables frame streaming, while a 0 (default) disables it. When frame streaming is active, a send frame command will continuously send frames back to back out the serial connection.

GB \r

This command Gets a Button press if one has been detected. This command returns either a 1 or a 0. If a 1 is returned, this means that the button was pressed sometime since the last call to Get Button. If a 0 is returned, then no button press was detected.

GH <channel> \r

This command Gets a Histogram of the *channel* specified by the user. The histogram contains 28 bins each holding the number of pixels that occurred within that bin's range of color values. So bin 0 on channel 0 would contain the number of red pixels that were between 16 and 23 in value. If no arguments are given, get histogram uses the last channel passed to get histogram. If get histogram is first called with no arguments, the green channel is used. The value returned in each bin is the number of pixels in that bin divided by the total number of pixels times 256 and capped at 255.

GI \r

This command Gets the auxiliary I/O Input values. When get inputs is called, a byte is returned containing the values of the auxiliary IO pins. This can be used to read digital inputs connected to the auxiliary I/O port. The aux I/O pins are internally lightly pulled high. See page 22 for pin numbering. Note that the pins are pulled up internally by the processor.

Example of how to read the auxiliary I/O pins. (in this case, pins 0 and 1 are high, while pins 2 and 3 are low).

```
:GI
3
ACK
:
```

GM \r

This command will Get the Mean color value in the current image. If, optionally, a subregion of the image is selected via virtual windowing, this function will only operate on the selected region. The mean values will be between 16 and 240 due to the limits of each color channel on the CMOS camera. It will also return a measure of the average absolute deviation of color found in that region. The mean together with the deviation can be a useful tool for automated tracking or detecting change in a scene. In YCrCb mode RGB maps to CrYCb.

This command returns a Type S data packet that by default has the following parameters:

S Rmean Gmean Bmean Rdeviation Gdeviation Bdeviation\r

Example of how to grab the mean color of the entire window:

```
:SW 1 1 40 143
ACK
:GM
ACK
S 89 90 67 5 6 3
S 89 91 67 5 6 2
```

GS servo \r

This command will Get the last position that was sent to the Servos.

Example of how to use get servo:

```
:GS 1
ACK
128
:
```

GT \r

This command Gets the current Track color values. This is a useful way to see what color values track window is using.

This example shows how to get the current tracking values:

```
:TW
ACK
T 12 34 ....
:GT
ACK
200 16 16 240 20 20
:
```

GV \r

This command Gets the current Version of the firmware and camera module version from the camera. It returns an ACK followed by the firmware version string. c6 means that it detects an OV6620, while c7 means that it detected an OV7620.

Example of how to ask for the firmware version and camera type:

```
:GV
ACK
CMUcam2 v1.00 c6
```

GW \r

This command Gets the current virtual Windowing values. This command allows you to confirm your current window configuration. It returns the x1, y1, x2 and y2 values that bound the current window.

HC #_of_bins scale \r

This command lets you Configure the Histogram settings. The first parameter takes one of three possible values. A value of 0 (default) will cause GH to output 28 bins. A value of 1 will generate 14 bins and a value of 2 will generate 7 bins. The scale parameter (default 0) allows you to better examine bins with smaller counts. Bin values are scaled by 2^{scale} where scale is the second parameter of the command.

#_of_bins

Input	Bins
0	28
1	14
2	7

HD boolean \r

This command enables or disables HiRes frame Differencing. A value of 0 (default) disables the high resolution frame differencing mode, while a value of 1 enables it. When enabled, frame differencing will operate at 16x16 instead of 8x8. The captured image is still stored internally at 8x8. The extra resolution is achieved by doing 4 smaller comparisons against each internally stored pixel. This will only yield good results when the background image is relatively smooth, or has a uniform color.

HR state \r

This sets the camera into HiRes mode. This is only available using the OV6620 camera module. A *state* value of 0 (default) gives you the standard 88x143, while 1 gives you 176x287. HiRes mode truncates the image to 176x255 for tracking so that the value does not overflow 8 bits.

HT boolean \r

This command enables or disables Histogram Tracking. When histogram tracking is enabled, only values that are within the color tracking bounds will be displayed in the histograms. This allows you to select exact color ranges giving you more detail, and ignoring any other background influences. A value of 0 (default) will disable histogram tracking, while a value of 1 will enable it. Note that the tracking noise filter applies just like it does with the TC and TW commands.

L0 boolean \r
L1 boolean \r

These commands enable and disable the two tracking LEDs. A value of 0 will turn the LED off, while a value of 1 will turn it on. A value of 2 (default) will leave the LED in automatic mode. In this mode, LED 1 turns on when the camera confidently detects an object while tracking and provides feedback during a send frame. In automatic mode, LED 0 does nothing, so it can be manually set.

LF \r

This command Loads a new Frame into the processor's memory to be differenced from. This does not have anything to do with the camera's frame buffer. It simply loads a baseline image for motion differencing and motion tracking.

LM type mode \r

This command enables Line Mode which transmits more detailed data about the image. It adds prefix data onto either T or S packets. This mode is intended for users who wish to do more complex image processing on less reduced data. Due to the higher rate of data, this may not be suitable for many slower microcontrollers. These are the different types and modes that line mode applies to different processing functions:

Type	Mode	Effected Command	Description
0	0	TC TW	Default where line mode is disabled
0	1	TC TW	Sends a binary image of the pixels being tracked
0	2	TC TW	Sends the Mean, Min, Max, confidence and count for every horizontal line of the tracked image.
1	0	GM	Default where line mode is disabled
1	1	GM	Sends the mean values for every line in the image
1	2	GM	Sends the mean values and the deviations for every line being tracked in the image
2	0	FD	Default where line mode is disabled
2	1	FD	Returns a bitmap of tracked pixels much like type 0 mode 0 of track color
2	2	FD	Sends the difference between the current image pixel value and the stored image. This gives you delta frame differenced images.
2	3	LF FD	This gives you the actual averaged value for each element in a differenced frame. It also returns these values when you load in a new frame. This can be used to give a very high speed gray scale low resolution stream of images.

Note, that the "mode" of each "type" of linemode can be controlled independently.

Data Packet Description

When raw mode is disabled all output data packets are in ASCII viewable format except for the F frame and prefix packets.

ACK

This is the standard acknowledge string that indicates that the command was received and fits a known format.

NCK

This is the failure string that is sent when an error occurred. The only time this should be sent when an error has not occurred is during binary data packets.

Type **F** data packet format:

1 Xsize Ysize 2 r g b r g b ... r g b r g b 2 r g b r g b ... r g b r g b 3

1 - new frame 2 - new row 3 - end of frame
RGB (CrYCb) ranges from 16 - 240
RGB (CrYCb) represents two pixels color values. Each pixel shares the red and blue.
176 cols of R G B (Cr Y Cb) packets (forms 352 pixels)
144 rows
To display the correct aspect ratio, double each column so that your final image is 352x144

Type **H** packet:

H bin0 bin1 bin2 bin3 ... bin26 bin27 \r

This is the return packet from calling get histogram (GH). Each bin is an 8 bit value that represents the number of pixels that fell within a set range of values on a user selected channel of the image.

Bin0 – number of pixels between 16 and 23
Bin1 – number of pixels between 24 and 31
.
.
.
Bin27 – number of pixels between 232 and 240

Type **T** packet:

T mx my x1 y1 x2 y2 pixels confidence\r

This is the return packet from a color tracking or frame differencing command.

mx - The middle of mass x value
my - The middle of mass y value
x1 - The left most corner's x value
y1 - The left most corner's y value
x2 - The right most corner's x value
y2 -The right most corner's y value
pixels –Number of Pixels in the tracked region, scaled and capped at 255: (pixels+4)/8
confidence -The (# of pixels / area)*256 of the bounded rectangle and capped at 255

Type S data packet format:

S Rmean Gmean Bmean Rdeviation Gdeviation Bdeviation \r

This is a statistic packet that gives information about the camera's view

Rmean - the mean Red or Cr (approximates r-g) value in the current window
Gmean - the mean Green or Y (approximates intensity) value found in the current window
Bmean - the mean Blue or Cb (approximates b-g) found in the current window
R*deviation* - the *deviation of red or Cr found in the current window
G*deviation*- the *deviation of green or Y found in the current window
B*deviation*- the *deviation of blue or Cb found in the current window

*deviation: The mean of the absolute difference between the pixels and the region mean.