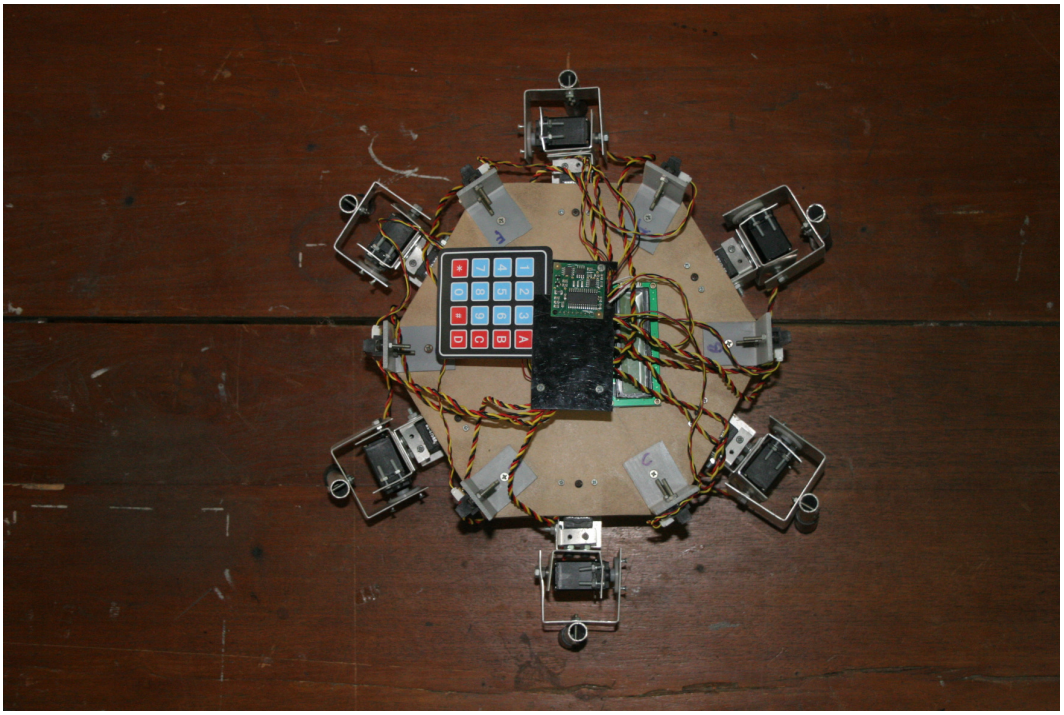
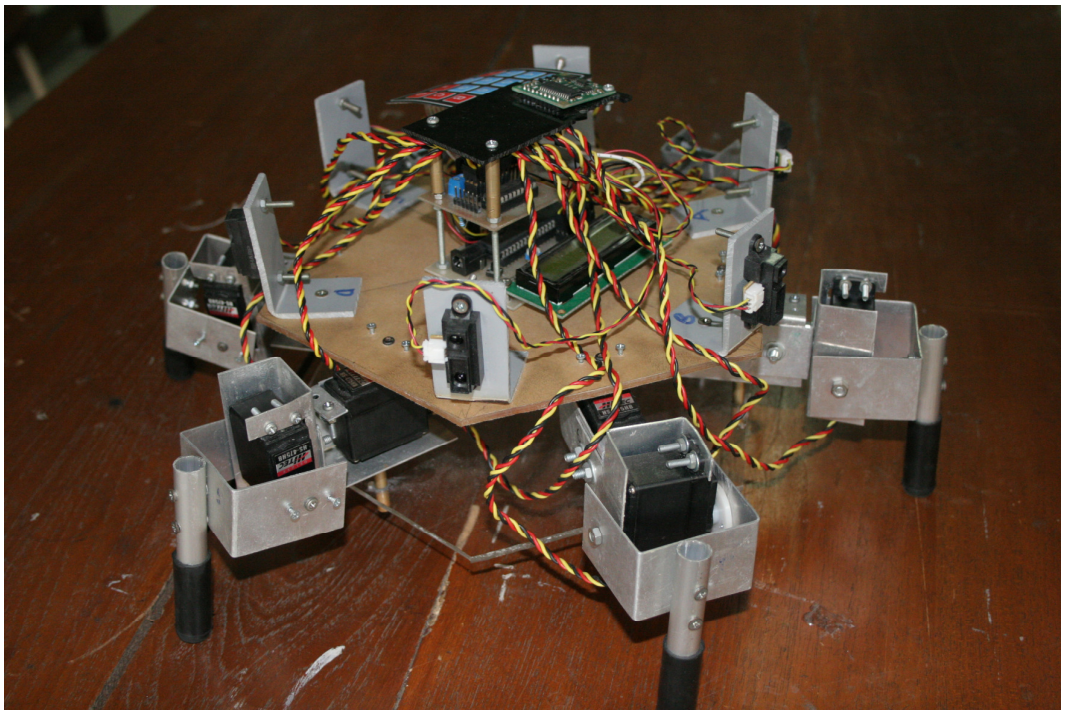
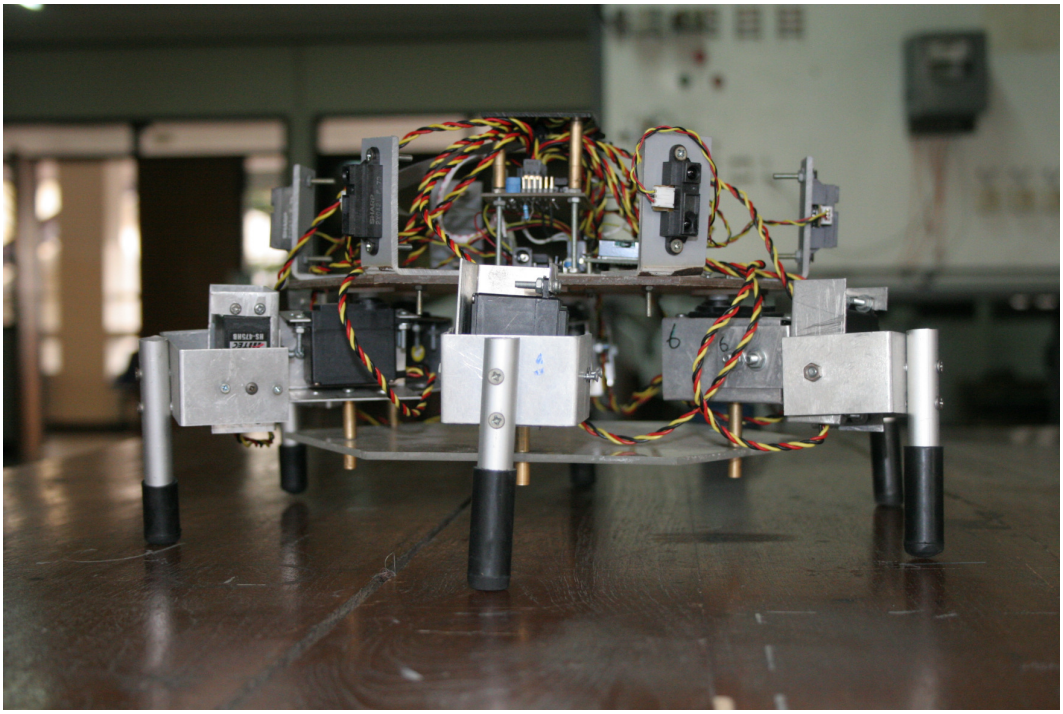
LAMPIRAN A
FOTO ROBOT BERKAKI ENAM







LAMPIRAN B
PROGRAM PADA PENGONTROL
ATMEGA16 DAN ATTINY2313

1. Robot mampu berjalan sesuai dengan langkah dan arah yang dimasukkan melalui *keypad*.

```
ATMEGA16
/*****************************************************
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 :
Version :
Date    : 2/6/2007
Author  : Laboratorium
Company : Fisika dan Instrumentasi
Comments:
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 <stdio.h>
#include <delay.h>
#include <math.h>
#include <scankeypadB.h>

int sudut,i;
unsigned char text1[32],text2[32],text3[32];
int data1,data2;
unsigned int kps;
char lngkh;

// I2C Bus functions
#asm
   .equ __i2c_port=0x1B ;PORTA
   .equ __sda_bit=1
   .equ __scl_bit=0
#endasm
#include <i2c.h>

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

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

#define ADC_VREF_TYPE 0x60

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}
// Declare your global variables here
int keytonum(void)
{
long val;
char datal[33];
char keypadchar;
val = 0x00;
while (keypadchar != '#')
```

```
{
keypadchar = scan_keypadB();
switch (keypadchar)
   {
   case 1: val = val +1;
   break;
   case 2: val = val +2;
   break;
   case 3: val = val +3;
   break;
   case 4: val = val +4;
   break;
   case 5: val = val +5;
   break;
   case 6: val = val +6;
   break;
   case 7: val = val +7;
   break;
   case 8: val = val +8;
   break;
   case 9: val = val +9;
   break;
   case '0': val = val;
   break;
   case '*': val = 0;
   break;
   }
if (keypadchar != '*' & keypadchar != '#' & keypadchar != 0 & keypadchar != 'A' & keypadchar != 'B' & keypadchar != 'C' &
keypadchar != 'D')
       {
       val = val * 10;
       }

keypadchar = 0;
lcd_clear();
sprintf (datal,"ANGKA=%u \n*=clear #=OK ", val/10);
lcd_puts (datal);
delay_ms(300);
keypadchar = scan_keypadB();
}
return val/10;
}

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=T State0=T
PORTB=0x00;
DDRB=0x00;

// 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=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;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x08;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x47;

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

// ADC initialization
// ADC Clock frequency: 691.200 kHz
// ADC Voltage Reference: AVCC pin
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;
// I2C Bus initialization
```

```
i2c_init();

// LCD module initialization
lcd_init(16);

while (1)
      {
      // Place your code here

       lcd_clear();
       lngkh=keytonum();
       lcd_putsf("langkah=");
       delay_ms(2000);
       lcd_clear();
       sprintf(text3,"langkah=%d",lngkh);
       lcd_puts(text3);
       putchar (lngkh);

      delay_ms(3000);

      i2c_start();
      i2c_write(0xC0);
      i2c_write(0x02);
      i2c_start();
      i2c_write(0xC1);
      data1=i2c_read(1);
      data2=i2c_read(0);
      i2c_stop();
      lcd_clear();
      kps=((float)data1*256+data2)/10;
      sprintf(text1,"kps=%d",kps);
      lcd_puts(text1);
      delay_ms(2000);

      lcd_clear();
      sudut=keytonum();
      lcd_putsf("Sudut=");
      delay_ms(2000);
      lcd_clear();
      sprintf(text2,"sudut=%d",sudut);
      lcd_puts(text2);

       if (  (kps>337.5)  &&  (kps<=359.9)  ||  (kps>=0) && (kps<=22.5)  )
      {
      switch (sudut) {
        case 0:       {putchar('A');} break;
        case 45:      {putchar ('B');} break;
        case 90:      {putchar ('C');} break;
        case 135:     {putchar ('D');} break;
        case 180:     {putchar ('E');} break;
        case 225:     {putchar ('F');} break;
        case 270:     {putchar ('G');} break;
        case 315:     {putchar ('H');} break;};}

    if (  (kps>22.5)  &&  (kps<=67.5)  )
    {
    switch (sudut) {
      case 0:      {putchar ('H');}break;
      case 45:     {putchar ('A');}break;
      case 90:     {putchar ('B');}break;
      case 135:    {putchar ('C');}break;
      case 180:    {putchar ('D');}break;
      case 225:    {putchar ('E');}break;
      case 270:    {putchar ('F');}break;
      case 315:    {putchar ('G');}break;};}

    if (  (kps>67.5)  &&  (kps<=112.5)  )
    {
    switch (sudut) {
      case 0:      {putchar ('G');}break;
      case 45:     {putchar ('H');}break;
      case 90:     {putchar ('A');}break;
      case 135:    {putchar ('B');}break;
      case 180:    {putchar ('C');}break;
      case 225:    {putchar ('D');}break;
```

```
    case 270:     {putchar ('E');}break;
    case 315:     {putchar ('F');} break;};}

  if (  (kps>112.5)  &&  (kps<=157.5)  )
  {
  switch (sudut) {
     case 0:       {putchar ('F');}break;
     case 45:      {putchar ('G');}break;
     case 90:      {putchar ('H');}break;
     case 135:     {putchar ('A');}break;
     case 180:     {putchar ('B');}break;
     case 225:      {putchar ('C');}break;
     case 270:      {putchar ('D');}break;
     case 315:      {putchar ('E');}break;};}

   if (  (kps>157.5)  &&  (kps<=202.5)  )
   {
   switch (sudut) {
     case 0:      {putchar ('E');}break;
     case 45:      {putchar ('F');}break;
     case 90:     {putchar ('G');}break;
     case 135:     {putchar ('H');}break;
     case 180:     {putchar ('A');}break;
     case 225:     {putchar ('B');}break;
     case 270:     {putchar ('C');}break;
     case 315:     {putchar ('D');}break;};}

    if (  (kps>202.5)  &&  (kps<=247.5)  )
    {
    switch (sudut) {
     case 0:       {putchar ('D');}break;
     case 45:      {putchar ('E');}break;
     case 90:      {putchar ('F');}break;
     case 135:     {putchar ('G');}break;
     case 180:     {putchar ('H');}break;
     case 225:     {putchar ('A');}break;
     case 270:     {putchar ('B');}break;
     case 315:    {putchar ('C');}break;};}

    if (  (kps>247.5)  &&  (kps<=292.5)  )
    {
    switch (sudut) {
     case 0:       {putchar ('C');}break;
     case 45:      {putchar ('D');}break;
     case 90:      {putchar ('E');}break;
     case 135:      {putchar ('F');}break;
     case 180:     {putchar ('G');}break;
     case 225:      {putchar ('H');}break;
     case 270:      {putchar ('A');}break;
     case 315:      {putchar ('B');}break;};}

    if (  (kps>292.5)  &&  (kps<=337.5)  )
    {
    switch (sudut) {
     case 0:       {putchar ('B');}break;
     case 45:      {putchar ('C');}break;
     case 90:      {putchar ('D');}break;
     case 135:     {putchar ('E');}break;
     case 180:     {putchar ('F');}break;
     case 225:     {putchar ('G');}break;
     case 270:     {putchar ('H');}break;
     case 315:     {putchar ('A');}break;};}

    };

}
```

ATTINY2313

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

Project :
Version :
Date    : 3/11/2009
Author  : Lab Instrumentasi
Company : UKM
Comments:

Chip type           : ATtiny2313
Clock frequency     : 8.000000 MHz
Memory model        : Tiny
External SRAM size  : 0
Data Stack size     : 32
*****************************************************/

#include <tiny2313.h>
#include <delay.h>
#include <stdio.h>
char langkah;
char servo,a;
int i;
int posisi45;
int posisi90;
int posisi110;
int posisi135;
int posisid45;
int posisid90;
int posisid110;
int posisid135;

#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;
     };
  };
  //servo=getchar();

}

#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

char naik(char servo_no)
{
posisi45 = posisi45|servo_no;
posisi90 = posisi90|servo_no;
posisi110 = posisi110|servo_no;
posisi135 = posisi135|servo_no;
}

char turun(char servo_no)
{
posisi135 = posisi135&(~servo_no);
posisi110 = posisi110&(~servo_no);
posisi90 = posisi90&(~servo_no);
posisi45 = posisi45&(~servo_no);
}

char tengah(char servo_no)
{
posisi135 = posisi135&(~servo_no);
posisi110 = posisi110&(~servo_no);
posisi45 = posisi45|servo_no;
posisi90 = posisi90|servo_no;
}

char naikd(char servo_no)
{
posisid45 = posisid45|servo_no;
posisid90 = posisid90|servo_no;
posisid110 = posisid110|servo_no;
posisid135 = posisid135|servo_no;
}

char turund(char servo_no)
{
posisid135 = posisid135&(~servo_no);
posisid110 = posisid110&(~servo_no);
posisid90 = posisid90&(~servo_no);
posisid45 = posisid45&(~servo_no);
}

char tengahd(char servo_no)
{
posisid135 = posisid135&(~servo_no);
posisid110 = posisid110&(~servo_no);
posisid45 = posisid45|servo_no;
posisid90 = posisid90|servo_no;
}
```

```
// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here

PORTB=0xFF;
PORTD=0xFF;
delay_us(1250);
PORTB=posisi45;
PORTD=posisid45;
delay_us(50);
PORTB=posisi90;
PORTD=posisid90;
delay_us(200);
PORTB=posisi110;
PORTD=posisid110;
delay_us(200);
PORTB=posisi135;
PORTD=posisid135;
delay_us(150);
PORTB=0x00;
PORTD=0x00;
}

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

// Declare your global variables here

void sudut0(void)
{
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);
    naikd(0x20);
    delay_ms(500);
    naikd(0x04);
    delay_ms(500);
    turun(0x04);
    delay_ms(500);
    naik(0x20);
    delay_ms(500);
    turund(0x20);
    delay_ms(500);
    turund(0x04);
    delay_ms(500);
    naik(0x40);
    delay_ms(500);
    naikd(0x10);
    delay_ms(500);
    naik(0x04);
    delay_ms(500);
    turun(0x20);
    delay_ms(500);
    turun(0x40);
    delay_ms(500);
    turund(0x10);
    delay_ms(500);
    tengah(0x20);
    delay_ms(500);
    tengah(0x04);
    delay_ms(500);
    tengahd(0x20);
    delay_ms(500);
    tengahd(0x04);
    delay_ms(500);
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);

    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);
```

```
    naikd(0x40);
    delay_ms(500);
    naikd(0x08);
    delay_ms(500);
    turun(0x08);
    delay_ms(500);
    naik(0x01);
    delay_ms(500);
    turund(0x40);
    delay_ms(500);
    turund(0x08);
    delay_ms(500);
    naik(0x40);
    delay_ms(500);
    naikd(0x10);
    delay_ms(500);
    naik(0x08);
    delay_ms(500);
    turun(0x01);
    delay_ms(500);
    turund(0x10);
    delay_ms(500);
    turun(0x40);
    delay_ms(500);
    tengah(0x01);
    delay_ms(500);
    tengah(0x08);
    delay_ms(500);
    tengah(0x40);
    delay_ms(500);
    tengahd(0x08);
    delay_ms(500);
}

void sudut45(void)
{
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);
    naikd(0x20);
    delay_ms(500);
    naikd(0x04);
    delay_ms(500);
    turun(0x04);
    delay_ms(500);
    naik(0x20);
    delay_ms(500);
    turund(0x04);
    delay_ms(500);
    turund(0x20);
    delay_ms(500);
    naikd(0x40);
    delay_ms(500);
    naikd(0x08);
    delay_ms(500);
    naik(0x04);
    delay_ms(500);
    turun(0x20);
    delay_ms(500);
    turund(0x40);
    delay_ms(500);
    turund(0x08);
    delay_ms(500);
    tengah(0x20);
    delay_ms(500);
    tengah(0x04);
    delay_ms(500);
    tengahd(0x20);
    delay_ms(500);
    tengahd(0x04);
    delay_ms(500);
}

void sudut90(void)
{
```

```
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);
    naik(0x40);
    delay_ms(500);
    turun(0x02);
    delay_ms(500);
    turun(0x40);
    delay_ms(500);
    naik(0x02);
    delay_ms(500);
    tengah(0x02);
    delay_ms(500);
    tengah(0x40);
    delay_ms(500);
    naikd(0x10);
    delay_ms(500);
    naik(0x10);
    delay_ms(500);
    turund(0x10);
    delay_ms(500);
    turun(0x10);
    delay_ms(500);
    tengah(0x10);
    delay_ms(500);
    tengahd(0x10);
    delay_ms(500);
}

void sudut135(void)
{
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);
    naikd(0x40);
    delay_ms(500);
    naikd(0x08);
    delay_ms(500);
    turun(0x01);
    delay_ms(500);
    naik(0x08);
    delay_ms(500);
    turund(0x40);
    delay_ms(500);
    turund(0x08);
    delay_ms(500);
    naikd(0x04);
    delay_ms(500);
    naikd(0x20);
    delay_ms(500);
    naik(0x01);
    delay_ms(500);
    turun(0x08);
    delay_ms(500);
    turund(0x04);
    delay_ms(500);
    turund(0x20);
    delay_ms(500);
    tengah(0x01);
    delay_ms(500);
    tengah(0x08);
    delay_ms(500);
    tengahd(0x40);
    delay_ms(500);
    tengahd(0x08);
    delay_ms(500);
}

void sudut180(void)
{
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);
    naikd(0x20);
    delay_ms(500);
```

```
    naikd(0x04);
    delay_ms(500);
    naik(0x04);
    delay_ms(500);
    turun(0x20);
    delay_ms(500);
    turund(0x20);
    delay_ms(500);
    turund(0x04);
    delay_ms(500);
    naik(0x40);
    delay_ms(500);
    naikd(0x10);
    delay_ms(500);
    turun(0x04);
    delay_ms(500);
    naik(0x20);
    delay_ms(500);
    turun(0x40);
    delay_ms(500);
    turund(0x10);
    delay_ms(500);
    tengah(0x20);
    delay_ms(500);
    tengah(0x04);
    delay_ms(500);
    tengahd(0x20);
    delay_ms(500);
    tengahd(0x04);
    delay_ms(500);
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);

    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);

    naikd(0x40);
    delay_ms(500);
    naikd(0x08);
    delay_ms(500);
    naik(0x08);
    delay_ms(500);
    turun(0x01);
    delay_ms(500);
    turund(0x40);
    delay_ms(500);
    turund(0x08);
    delay_ms(500);
    naik(0x40);
    delay_ms(500);
    naikd(0x10);
    delay_ms(500);
    turun(0x08);
    delay_ms(500);
    naik(0x01);
    delay_ms(500);
    turund(0x10);
    delay_ms(500);
    turun(0x40);
    delay_ms(500);
    tengah(0x01);
    delay_ms(500);
    tengah(0x08);
    delay_ms(500);
    tengah(0x40);
    delay_ms(500);
    tengahd(0x08);
    delay_ms(500);
}

void sudut225(void)
{
    tengah(0xFF);
```

```
    tengahd(0xFF);
    delay_ms(500);
    naikd(0x20);
    delay_ms(500);
    naikd(0x04);
    delay_ms(500);
    naik(0x04);
    delay_ms(500);
    turun(0x20);
    delay_ms(500);
    turund(0x20);
    delay_ms(500);
    turund(0x04);
    delay_ms(500);
    naikd(0x40);
    delay_ms(500);
    naikd(0x08);
    delay_ms(500);
    turun(0x04);
    delay_ms(500);
    naik(0x20);
    delay_ms(500);
    turund(0x40);
    delay_ms(500);
    turund(0x08);
    delay_ms(500);
    tengah(0x04);
    delay_ms(500);
    tengah(0x20);
    delay_ms(500);
    tengahd(0x04);
    delay_ms(500);
    tengahd(0x20);
    delay_ms(500);
}

void sudut270(void)
{
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);
    naik(0x40);
    delay_ms(500);
    naik(0x02);
    delay_ms(500);
    turun(0x40);
    delay_ms(500);
    turun(0x02);
    delay_ms(500);
    tengah(0x02);
    delay_ms(500);
    tengah(0x40);
    delay_ms(500);
    naikd(0x10);
    delay_ms(500);
    turun(0x10);
    delay_ms(500);
    turund(0x10);
    delay_ms(500);
    naik(0x10);
    delay_ms(500);
    tengah(0x10);
    delay_ms(500);
    tengahd(0x10);
    delay_ms(500);
}


void sudut315(void)
{
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(500);
    naikd(0x40);
    delay_ms(500);
```

```
    naikd(0x08);
    delay_ms(500);
    naik(0x01);
    delay_ms(500);
    turun(0x08);
    delay_ms(500);
    turund(0x40);
    delay_ms(500);
    turund(0x08);
    delay_ms(500);
    naikd(0x04);
    delay_ms(500);
    naikd(0x20);
    delay_ms(500);
    turun(0x01);
    delay_ms(500);
    naik(0x08);
    delay_ms(500);
    turund(0x04);
    delay_ms(500);
    turund(0x20);
    delay_ms(500);
    tengah(0x01);
    delay_ms(500);
    tengah(0x08);
    delay_ms(500);
    tengahd(0x40);
    delay_ms(500);
    tengahd(0x08);
    delay_ms(500);
}

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

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

// Input/Output Ports initialization
// Port A initialization
// Func2=In Func1=In Func0=In
// State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

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

// Port D initialization
// Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0x7F;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0A output: Disconnected
// OC0B output: Disconnected
TCCR0A=0x00;
TCCR0B=0x00;
TCNT0=0x00;
OCR0A=0x00;
OCR0B=0x00;
```

```
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 125.000 kHz
// Mode: CTC top=OCR1A
// 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: On
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x0B;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0xD8;
OCR1AH=0x09;
OCR1AL=0xC4;
OCR1BH=0x00;
OCR1BL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// Interrupt on any change on pins PCINT0-7: Off
GIMSK=0x00;
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x40;

// Universal Serial Interface initialization
// Mode: Disabled
// Clock source: Register & Counter=no clk.
// USI Counter Overflow Interrupt: Off
USICR=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x90;
UCSRC=0x06;
UBRRH=0x00;
UBRRL=0x33;

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

// Global enable interrupts
#asm("sei")

while (1)
      {
      // Place your code here
     a=getchar();
     if(a>64)
     {
     servo=a;
     delay_ms(100);
       langkah=getchar();
     }
     else
     {
     langkah=a;
       delay_ms(100);
       servo=getchar();
     }
```

```
    if(servo=='A')
    {
          for(i=0;i<langkah;i++)
            {
            sudut0();
            }
      }
   if(servo=='B')
   {
            for(i=0;i<langkah;i++)
            {
            sudut45();
            }
   }
   if(servo=='C')
   {
            for(i=0;i<langkah;i++)
            {
            sudut90();
            }
   }
   if(servo=='D')
   {
            for(i=0;i<langkah;i++)
            {
            sudut135();
            }
   }
   if(servo=='E')
   {
            for(i=0;i<langkah;i++)
            {
            sudut180();
            }
   }
   if(servo=='F')
   {
            for(i=0;i<langkah;i++)
            {
            sudut225();
            }
   }
   if(servo=='G')
   {
            for(i=0;i<langkah;i++)
            {
            sudut270();
            }
   }
   if(servo=='H')
   {
            for(i=0;i<langkah;i++)
            {
            sudut315();
            }
    }

    };
}
```

2. Robot berjalan menuju ruangan-ruangan yang terdapat pada *maze*.

```
ATMEGA16
/*****************************************************
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 :
Version :
Date    : 2/6/2007
Author  : Laboratorium
Company : Fisika dan Instrumentasi
Comments:

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 <stdio.h>
#include <delay.h>
#include <math.h>
#include <scankeypadB.h>

int data1,data2;
unsigned int kps,kompas;
void sensorjarak(void);
int a,b,c,d,e,f;
unsigned char text[32],text1[32];

// I2C Bus functions
#asm
   .equ __i2c_port=0x1B ;PORTA
   .equ __sda_bit=1
   .equ __scl_bit=0
#endasm
#include <i2c.h>

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

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

#define ADC_VREF_TYPE 0x60

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCH;
}


// Declare your global variables here


void sensorjarak(void)
```

```
{
     a=read_adc(7);
     a=2141.72055 * (pow(a,-1.078867));
     b=read_adc(6);
     b=2141.72055 * (pow(b,-1.078867));
    c=read_adc(5);
    c=2141.72055 * (pow(c,-1.078867));
    d=read_adc(4);
    d=2141.72055 * (pow(d,-1.078867));
     e=read_adc(3);
     e=2141.72055 * (pow(e,-1.078867));
     f=read_adc(2);
     f=2141.72055 * (pow(f,-1.078867));
     lcd_clear();
     sprintf(text,"a=%d b=%d c=%d d=%d e=%d f=%d ",a,b,c,d,e,f);
     lcd_puts(text);
     delay_ms(200);
}


void koreksi(void)
{
     i2c_start();
     i2c_write(0xC0);
     i2c_write(0x02);
     i2c_start();
     i2c_write(0xC1);
     data1=i2c_read(1);
     data2=i2c_read(0);
     i2c_stop();
     lcd_clear();
     kps=((float)data1*256+data2)/10;
     sprintf(text1,"kps=%d",kps);
     lcd_puts(text1);
     delay_ms(200);

     kompas=220;
     if (kps<(kompas-5)) putchar ('R');
     if (kps>(kompas+5)) putchar ('L');

     i2c_start();
     i2c_write(0xC0);
     i2c_write(0x02);
     i2c_start();
     i2c_write(0xC1);
     data1=i2c_read(1);
     data2=i2c_read(0);
     i2c_stop();
     lcd_clear();
     kps=((float)data1*256+data2)/10;
     sprintf(text1,"kps=%d",kps);
     lcd_puts(text1);
     delay_ms(200);
}
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=T State0=T
PORTB=0x00;
DDRB=0x00;

// 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=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;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 9600 (Double Speed Mode)
UCSRA=0x02;
UCSRB=0x08;
UCSRC=0x86;
UBRRH=0x00;
UBRRL=0x8F;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
```

```
ACSR=0x80;
SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 691.200 kHz
// ADC Voltage Reference: AVCC pin
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// I2C Bus initialization
i2c_init();

// LCD module initialization
lcd_init(16);

while (1)
      {
      // Place your code here

      lcd_putsf("ruangan=");

      switch(scan_keypadB())
      {
      case 1:
//      ruangan 1
        {
       sensorjarak();
      while (  b<35 && e>80 )
      {
      koreksi();
      if (f<10) putchar ('E');
      if (d<10) putchar ('A');
      else putchar('G');
      sensorjarak();
      }

        sensorjarak();
      while ( a > 18 && f > 18 )
      {
       koreksi();
       if ( b < 10 ) putchar ('G');
       if ( e < 10 ) putchar ('C');
       else putchar ('A');
       sensorjarak();
      }

      sensorjarak();
      while ( ( a < 18 || f < 18 ) && b >15 )
      {
       koreksi();
       if(f<10 || a<10) putchar ('E');
       if(c<10 || d<10) putchar ('A');
       else putchar ('C');
         sensorjarak();
      }}
      break;

      case 2:
//       Ruangan 2
      {
        sensorjarak();
      while ( a > 18 && f > 18 )
      {
       koreksi();
       if ( b < 10 ) putchar ('G');
       if ( e < 10 ) putchar ('C');
       else putchar ('A');
       sensorjarak();
      }

      sensorjarak();
      while ( ( a < 18 || f < 18 ) && b >15 )
```

```
      {
       koreksi();
       if(f<10 || a<10) putchar ('E');
       if(c<10 || d<10) putchar ('A');
       else putchar ('C');
         sensorjarak();
       }

      sensorjarak();
      while(a<20 && f<20 && b<20 && c>35 && d>35)
      {
      koreksi();
      putchar ('E');
      sensorjarak();
      }}
      break;

      case 3:
//     ruangan 3
       {
          sensorjarak();
      while (b<150 && a>15 && f>15 && c>15 && d>5)
      {
      koreksi();
      if(c<15) putchar ('A');
      else putchar ('C');
      sensorjarak();
      }

       sensorjarak();
      while ( a > 18 && f > 18 )
      {
       koreksi();
       if ( b < 10 ) putchar ('G');
       if ( e < 10 ) putchar ('C');
       else putchar ('A');
       sensorjarak();
      }

      sensorjarak();
      while ( ( a < 18 || f < 18 ) && b >15 )
      {
       koreksi();
       if(f<10 || a<10) putchar ('E');
       if(c<10 || d<10) putchar ('A');
       else putchar ('C');
         sensorjarak();
       }

       sensorjarak();
      while (a>18 && f >18 && c>20 && d>20)
      {
      koreksi();
      putchar ('E');
      sensorjarak();
      }

      sensorjarak();
      while(b<20 && c>20 && d>20)
      {
      koreksi();
      putchar ('E');
      sensorjarak();
      }}
      break;

      case 4:
//      ruangan 4
       {
       sensorjarak();
       while ( a>18 && f>18 && c>15 && d>15)
       {
       koreksi();
       if (b<10) putchar('G');
       if (e>25) putchar ('G');
```

```
      else putchar ('E');
      sensorjarak();
      }

        sensorjarak();
      while ( a > 18 && f > 18 )
      {
       koreksi();
       if ( b < 10 ) putchar ('G');
       if ( e < 10 ) putchar ('C');
       else putchar ('A');
       sensorjarak();
      }

      sensorjarak();
      while ( ( a < 18 || f < 18 ) && b >15 )
      {
       koreksi();
       if(f<10 || a<10) putchar ('E');
       if(c<10 || d<10) putchar ('A');
       else putchar ('C');
         sensorjarak();
      }}
      break;

      };

      };

}
```

ATtiny2313

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

Project :
Version :
Date    : 3/11/2009
Author  : Lab Instrumentasi
Company : UKM
Comments:

Chip type           : ATtiny2313
Clock frequency     : 8.000000 MHz
Memory model        : Tiny
External SRAM size  : 0
Data Stack size     : 32
*****************************************************/

#include <tiny2313.h>
#include <delay.h>
#include <stdio.h>
char langkah;
char servo,a;
int i;
int posisi45;
int posisi90;
int posisi110;
int posisi135;
int posisid45;
int posisid90;
int posisid110;
int posisid135;

#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;
    };
  };

}

#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

char naik(char servo_no)
{
posisi45 = posisi45|servo_no;
posisi90 = posisi90|servo_no;
posisi110 = posisi110|servo_no;
posisi135 = posisi135|servo_no;
}
char turun(char servo_no)
{
posisi135 = posisi135&(~servo_no);
posisi110 = posisi110&(~servo_no);
posisi90 = posisi90&(~servo_no);
posisi45 = posisi45&(~servo_no);
}
char tengah(char servo_no)
{
posisi135 = posisi135&(~servo_no);
posisi110 = posisi110&(~servo_no);
posisi45 = posisi45|servo_no;
posisi90 = posisi90|servo_no;}
char naikd(char servo_no)
{
posisid45 = posisid45|servo_no;
posisid90 = posisid90|servo_no;
posisid110 = posisid110|servo_no;
posisid135 = posisid135|servo_no;
}
char turund(char servo_no)
{
posisid135 = posisid135&(~servo_no);
posisid110 = posisid110&(~servo_no);
posisid90 = posisid90&(~servo_no);
posisid45 = posisid45&(~servo_no);
}
char tengahd(char servo_no)
{
posisid135 = posisid135&(~servo_no);
posisid110 = posisid110&(~servo_no);
posisid45 = posisid45|servo_no;
posisid90 = posisid90|servo_no;
}

// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here
PORTB=0xFF;
PORTD=0xFF;
delay_us(1200);
```

```
PORTB=posisi45;
PORTD=posisid45;
delay_us(100);
PORTB=posisi90;
PORTD=posisid90;
delay_us(200);
PORTB=posisi110;
PORTD=posisid110;
delay_us(200);
PORTB=posisi135;
PORTD=posisid135;
delay_us(50);
PORTB=0x00;
PORTD=0x00;
}
// Standard Input/Output functions
#include <stdio.h>
// Declare your global variables here
void sudut0(void)
{
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(100);
    naikd(0x20);
    delay_ms(100);
    naikd(0x04);
    delay_ms(100);
    turun(0x04);
    delay_ms(100);
    naik(0x20);
    delay_ms(100);
    turund(0x20);
    delay_ms(100);
    turund(0x04);
    delay_ms(100);
    naik(0x40);
    delay_ms(100);
    naikd(0x10);
    delay_ms(100);
    naik(0x04);
    delay_ms(100);
    turun(0x20);
    delay_ms(100);
    tengah(0x40);
    delay_ms(100);
    tengahd(0x10);
    delay_ms(100);
    tengah(0x20);
    delay_ms(100);
    tengah(0x04);
    delay_ms(100);
    tengahd(0x20);
    delay_ms(100);
    tengahd(0x04);
    delay_ms(100);

    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(100);

    naikd(0x40);
    delay_ms(100);
    naikd(0x08);
    delay_ms(100);
    turun(0x08);
    delay_ms(100);
    naik(0x01);
    delay_ms(100);
    turund(0x40);
    delay_ms(100);
    turund(0x08);
    delay_ms(100);
    naik(0x40);
    delay_ms(100);
    naikd(0x10);
```

```
    delay_ms(100);
    naik(0x08);
    delay_ms(100);
    turun(0x01);
    delay_ms(100);
    tengahd(0x10);
    delay_ms(100);
    tengah(0x40);
    delay_ms(100);
    tengah(0x01);
    delay_ms(100);
    tengah(0x08);
    delay_ms(100);
    tengah(0x40);
    delay_ms(100);
    tengahd(0x08);
    delay_ms(100);
}
void sudut90(void)
{
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(100);
    naik(0x40);
    delay_ms(100);
    naikd(0x10);
    delay_ms(100);
    turun(0x02);
    delay_ms(100);
    naik(0x10);
    delay_ms(100);
    turun(0x40);
    delay_ms(100);
    turund(0x10);
    delay_ms(100);
    naikd(0x20);
    delay_ms(100);
    naikd(0x04);
    delay_ms(100);
    naik(0x02);
    delay_ms(100);
    turun(0x10);
    delay_ms(100);
    tengahd(0x20);
    delay_ms(100);
    tengahd(0x04);
    delay_ms(100);
    tengah(0x02);
    delay_ms(100);
    tengah(0x10);
    delay_ms(100);
    tengah(0x40);
    delay_ms(100);
    tengahd(0x10);
    delay_ms(100);

    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(100);
    naik(0x40);
    delay_ms(100);
    naikd(0x10);
    delay_ms(100);
    turun(0x02);
    delay_ms(100);
    naik(0x10);
    delay_ms(100);
    turun(0x40);
    delay_ms(100);
    turund(0x10);
    delay_ms(100);
    naikd(0x40);
    delay_ms(100);
    naikd(0x08);
    delay_ms(100);
```

```
    naik(0x02);
    delay_ms(100);
    turun(0x10);
    delay_ms(100);
    tengahd(0x40);
    delay_ms(100);
    tengahd(0x08);
    delay_ms(100);
    tengah(0x02);
    delay_ms(100);
    tengah(0x10);
    delay_ms(100);
    tengah(0x40);
    delay_ms(100);
    tengahd(0x10);
    delay_ms(100);

}

void sudut180(void)
{
     tengah(0xFF);
    tengahd(0xFF);
    delay_ms(100);
    naikd(0x20);
    delay_ms(100);
    naikd(0x04);
    delay_ms(100);
    naik(0x04);
    delay_ms(100);
    turun(0x20);
    delay_ms(100);
    turund(0x20);
    delay_ms(100);
    turund(0x04);
    delay_ms(100);
    naik(0x40);
    delay_ms(100);
    naikd(0x10);
    delay_ms(100);
    turun(0x04);
    delay_ms(100);
    naik(0x20);
    delay_ms(100);
    tengah(0x40);
    delay_ms(100);
    tengahd(0x10);
    delay_ms(100);
    tengah(0x20);
    delay_ms(100);
    tengah(0x04);
    delay_ms(100);
    tengahd(0x20);
    delay_ms(100);
    tengahd(0x04);
    delay_ms(100);

    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(100);

    naikd(0x40);
    delay_ms(100);
    naikd(0x08);
    delay_ms(100);
    naik(0x08);
    delay_ms(100);
    turun(0x01);
    delay_ms(100);
    turund(0x40);
    delay_ms(100);
    turund(0x08);
    delay_ms(100);
    naik(0x40);
    delay_ms(100);
```

```
    naikd(0x10);
    delay_ms(100);
    turun(0x08);
    delay_ms(100);
    naik(0x01);
    delay_ms(100);
    tengahd(0x10);
    delay_ms(100);
    tengah(0x40);
    delay_ms(100);
    tengah(0x01);
    delay_ms(100);
    tengah(0x08);
    delay_ms(100);
    tengah(0x40);
    delay_ms(100);
    tengahd(0x08);
    delay_ms(100);
}

void sudut270(void)
{
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(100);

    naik(0x40);
    delay_ms(100);
    naikd(0x10);
    delay_ms(100);
    naik(0x02);
    delay_ms(100);
    turun(0x10);
    delay_ms(100);
    turun(0x40);
    delay_ms(100);
    turund(0x10);
    delay_ms(100);
    naikd(0x40);
    delay_ms(100);
    naikd(0x08);
    delay_ms(100);
    turun(0x02);
    delay_ms(100);
    naik(0x10);
    delay_ms(100);
    tengahd(0x40);
    delay_ms(100);
    tengahd(0x08);
    delay_ms(100);
    tengah(0x02);
    delay_ms(100);
    tengah(0x10);
    delay_ms(100);
    tengah(0x40);
    delay_ms(100);
    tengahd(0x10);
    delay_ms(100);
    tengah(0xFF);
    tengahd(0xFF);
    delay_ms(100);
    naik(0x40);
    delay_ms(100);
    naikd(0x10);
    delay_ms(100);
    naik(0x02);
    delay_ms(100);
    turun(0x10);
    delay_ms(100);
    turun(0x40);
    delay_ms(100);
    turund(0x10);
    delay_ms(100);
    naikd(0x04);
    delay_ms(100);
```

```
    naikd(0x20);
    delay_ms(100);
    turun(0x02);
    delay_ms(100);
    naik(0x10);
    delay_ms(100);
    tengahd(0x04);
    delay_ms(100);
    tengahd(0x20);
    delay_ms(100);
    tengah(0x02);
    delay_ms(100);
    tengah(0x10);
    delay_ms(100);
    tengah(0x40);
    delay_ms(100);
    tengahd(0x10);
    delay_ms(100);
}

void koreksi1(void)
{
      tengah(0xFF);
      tengahd(0xFF);
      delay_ms(100);
      naik(0x40);
      delay_ms(100);
      naikd(0x10);
      delay_ms(100);
      naik(0x02);
      delay_ms(100);
      naik(0x10);
      delay_ms(100);
      turun(0x40);
      delay_ms(100);
      turund(0x10);
      delay_ms(100);
      naikd(0x04);
      delay_ms(100);
      naikd(0x20);
      delay_ms(100);
      turun(0x02);
      delay_ms(100);
      turun(0x10);
      delay_ms(100);
      tengah(0x02);
      delay_ms(100);
      tengah(0x10);
      delay_ms(100);
      tengah(0x40);
      delay_ms(100);
      tengahd(0x10);
      delay_ms(100);

      tengah(0xFF);
      tengahd(0xFF);
      delay_ms(100);
      naik(0x40);
      delay_ms(100);
      naikd(0x10);
      delay_ms(100);
      naik(0x02);
      delay_ms(100);
      naik(0x10);
      delay_ms(100);
      turun(0x40);
      delay_ms(100);
      turund(0x10);
      delay_ms(100);
      naikd(0x40);
      delay_ms(100);
      naikd(0x08);
      delay_ms(100);
      turun(0x02);
      delay_ms(100);
```

```
        turun(0x10);
        delay_ms(100);
        tengah(0x02);
        delay_ms(100);
        tengah(0x10);
        delay_ms(100);
        tengah(0x40);
        delay_ms(100);
        tengahd(0x10);
        delay_ms(100);
}

void koreksi2(void)
{
        tengah(0xFF);
        tengahd(0xFF);
        delay_ms(100);
        naik(0x40);
        delay_ms(100);
        naikd(0x10);
        delay_ms(100);
        turun(0x02);
        delay_ms(100);
        turun(0x10);
        delay_ms(100);
        turun(0x40);
        delay_ms(100);
        turund(0x10);
        delay_ms(100);
        naikd(0x04);
        delay_ms(100);
        naikd(0x20);
        delay_ms(100);
        naik(0x02);
        delay_ms(100);
        naik(0x10);
        delay_ms(100);
        tengah(0x02);
        delay_ms(100);
        tengah(0x10);
        delay_ms(100);
        tengah(0x40);
        delay_ms(100);
        tengahd(0x10);
        delay_ms(100);

}

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

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

// Input/Output Ports initialization
// Port A initialization
// Func2=In Func1=In Func0=In
// State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;

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

// Port D initialization
// Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
```

```
// State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
PORTD=0x00;
DDRD=0x7F;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0A output: Disconnected
// OC0B output: Disconnected
TCCR0A=0x00;
TCCR0B=0x00;
TCNT0=0x00;
OCR0A=0x00;
OCR0B=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 125.000 kHz
// Mode: CTC top=OCR1A
// 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: On
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x0B;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0xD8;
OCR1AH=0x09;
OCR1AL=0xC4;
OCR1BH=0x00;
OCR1BL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// Interrupt on any change on pins PCINT0-7: Off
GIMSK=0x00;
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x40;

// Universal Serial Interface initialization
// Mode: Disabled
// Clock source: Register & Counter=no clk.
// USI Counter Overflow Interrupt: Off
USICR=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
// USART Baud rate: 9600
UCSRA=0x00;
UCSRB=0x90;
UCSRC=0x06;
UBRRH=0x00;
UBRRL=0x33;

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

// Global enable interrupts
#asm("sei")
```

```
while (1)
      {
      // Place your code here
      servo=getchar();
      if(servo=='A')
      {
      sudut0();
      }
     if(servo=='C')
     {
     sudut90();
     }
     if(servo=='E')
     {
     sudut180();
     }
     if(servo=='G')
     {
     sudut270();
     }
      if(servo=='L')
     {
     koreksi1();
      }
      if(servo=='R')
     {
     koreksi2();
      }
```

LAMPIRAN C
DATASHEET

FEATURES

- Analog output
- Effective Range: 10 to 80 cm
- LED pulse cycle duration: 32 ms
- Typical response time: 39 ms
- Typical start up delay: 44 ms
- Average current consumption: 33 mA
- Detection area diameter @ 80 cm: 6 cm

DESCRIPTION

The GP2D12 is a distance measuring sensor with integrated signal processing and analog voltage output.

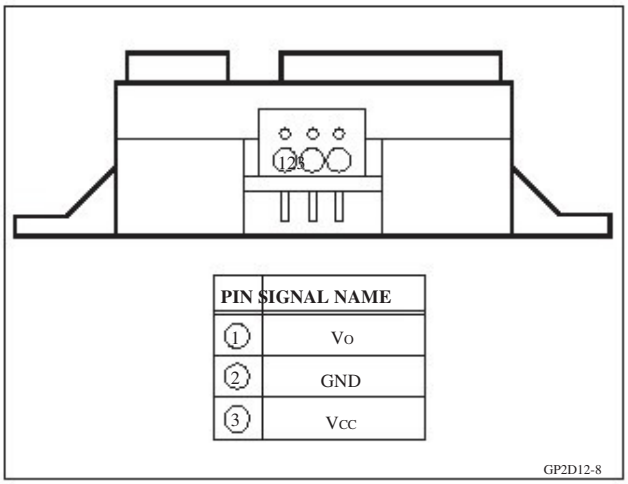


Figure 1. Pinout

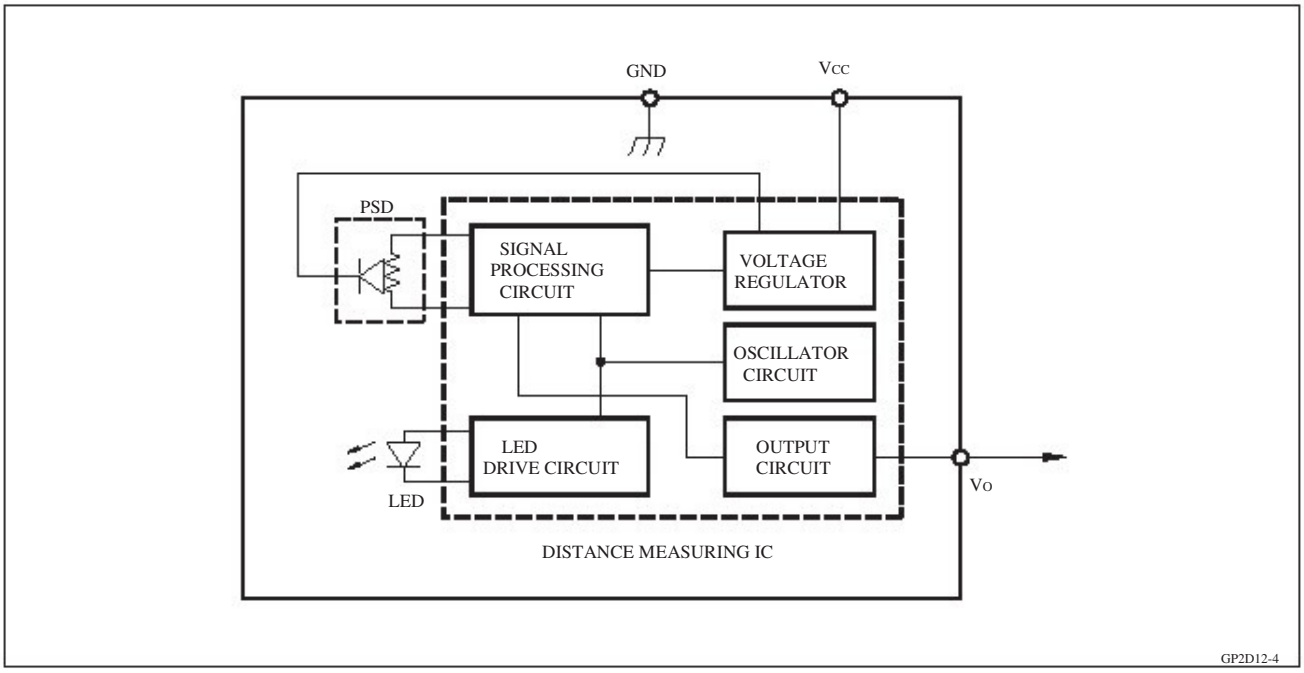


Figure 2. Block Diagram

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings

Ta = 25°C, V_{CC} = 5 VDC

PARAMETER	SYMBOL	RATING	UNIT
Supply Voltage	V_{CC}	-0.3 to +7.0	V
Output Terminal Voltage	V_O	-0.3 to (V_{CC} + 0.3)	V
Operating Temperature	Topr	-10 to +60	°C
Storage Temperature	Tstg	-40 to +70	°C

Operating Supply Voltage

PARAMETER	SYMBOL	RATING	UNIT
Operating Supply Voltage	V_{CC}	4.5 to 5.5	V

Electro-optical Characteristics

Ta = 25°C, V_{CC} = 5 VDC

PARAMETER	SYMBOL	CONDITIONS	MIN.	TYP.	MAX.	UNIT	NOTES
Measuring Distance Range	ΔL		10	-	80	cm	1, 2
Output Voltage	V_O	L = 80 cm	0.25	0.4	0.55	V	1, 2
Output Voltage Difference	ΔV_O	Output change at L change 1.75 (80 cm - 10 cm)		2.0	2.25	V	1, 2
Average Supply Current	I_{CC}	L = 80 cm	-	33	50	mA	1, 2

NOTES:
1. Measurements made with Kodak R-27 Gray Card, using the white side, (90% reflectivity).
2. L = Distance to reflective object.

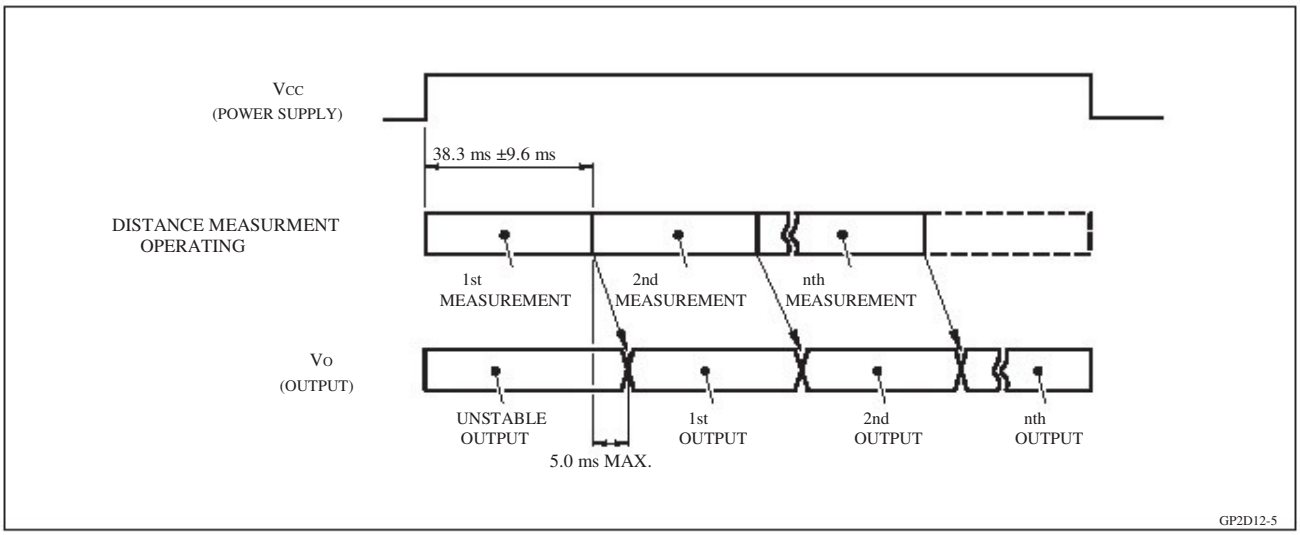


Figure 3. Timing Diagram

RELIABILITY

The reliability of requirements of this device are listed in Table 1.

Table 1. Reliability

TEST ITEMS	TEST CONDITIONS	FAILURE JUDGEMENT CRITERIA	SAMPLES (n), DEFECTIVE (C)
Temperature Cycling	One cycle -40°C (30 min.) to +70°C in 30 minutes, repeated 25 times	Initial × 0.8 > V_O V_O > Initial × 1.2	n = 11, C = 0
High Temperature and High Humidity Storage	+40°C, 90% RH, 500h		n = 11, C = 0
High Temperature Storage	+70°C, 500h		n = 11, C = 0
Low Temperature Storage	-40°C, 500h		n = 11, C = 0
Operational Life (High Temperature)	+60°C, V_{CC} = 5 V, 500h		n = 11, C = 0
Mechanical Shock	100 m / s_2, 6.0 ms 3 times / ±X, ±Y, ±Z direction		n = 6, C = 0
Variable Frequency Vibration	10-to-55-to-10 Hz i n 1 minute Amplitude: 1.5 mm 2 h i n e a c h X, Y, Z direction		n = 6, C = 0

NOTES:
1. Test conditions are according to Electro-optical Characteristics, shown on page 2.
2. At completion of the test, allow device to remain at nominal room temperature and humidity (non-condensing) for two hours.
3. Confidence level: 90%, Lot Tolerance Percent Defect (LTPD): 20% / 40%.

MANUFACTURER'S INSPECTION

Inspection Lot

Inspection shall be carried out per each delivery lot.

Inspection Method

A single sampling plan, normal inspection level II based on ISO 2859 shall be adopted.

Table 2. Quality Level

DEFECT	INSPECTION ITEM and TEST METHOD	AQL (%)
Major Defect	Electro-optical characteristics defect	0.4
Minor Defect	Defect to appearance or dimensions (crack, split, chip, scratch, stain)*	1.0

NOTE: *Any one of these that affects the Electro-optical Characteristics shall be considered a defect.

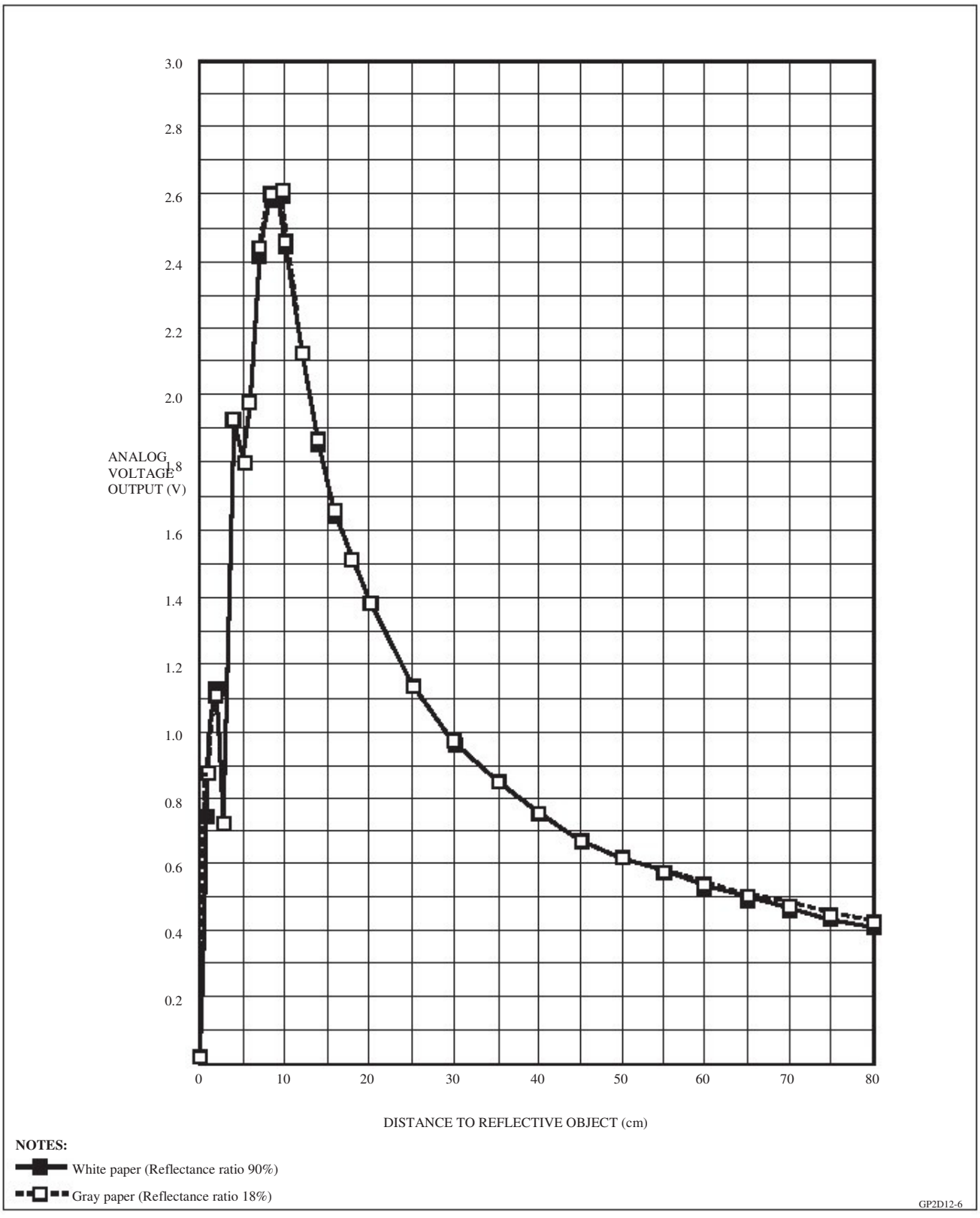


Figure 4. GP2D12 Example of Output/Distance Characteristics

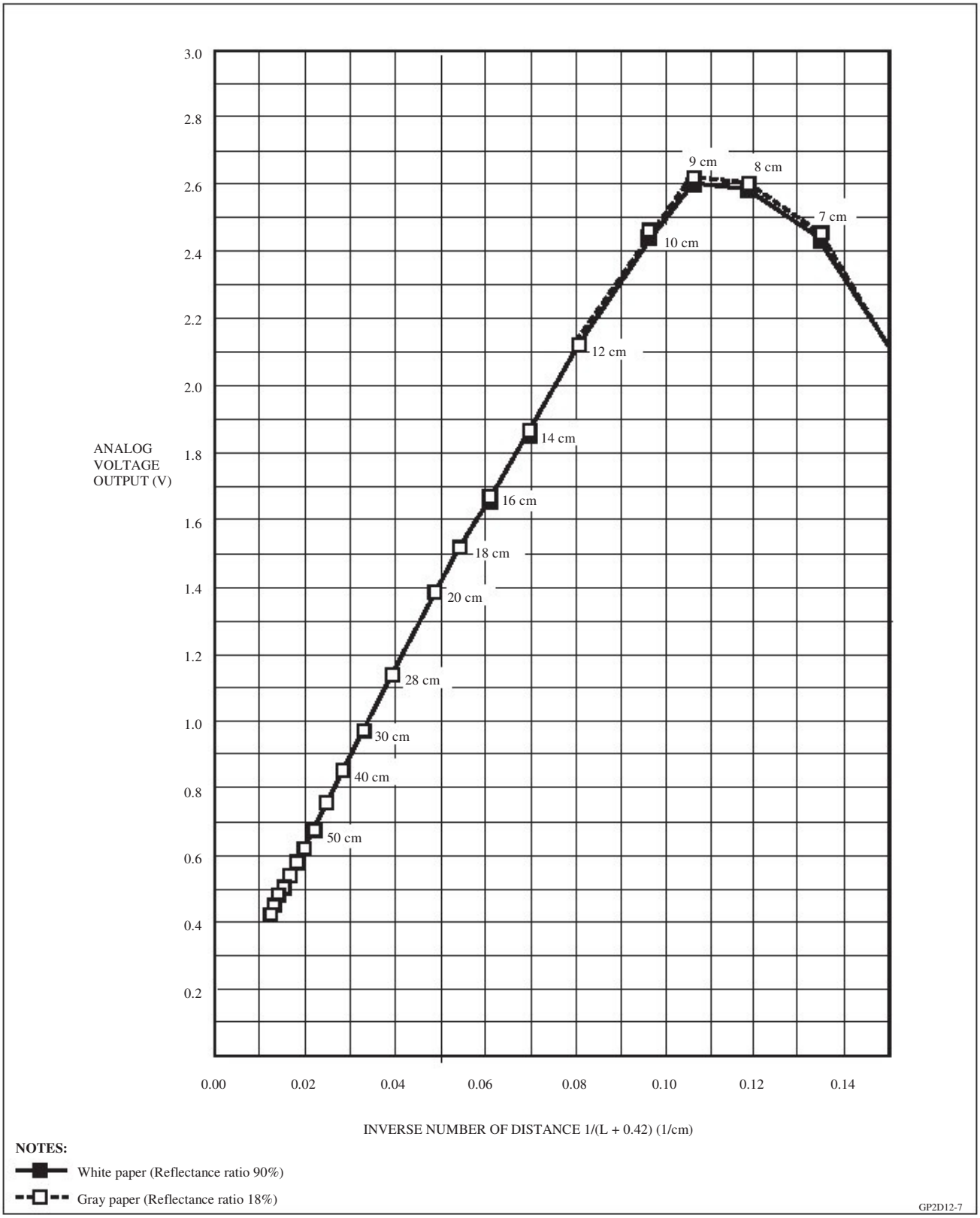
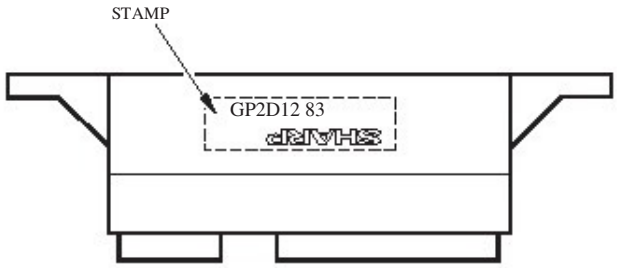
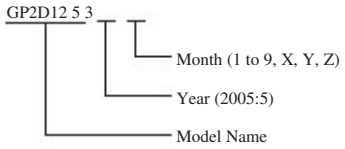


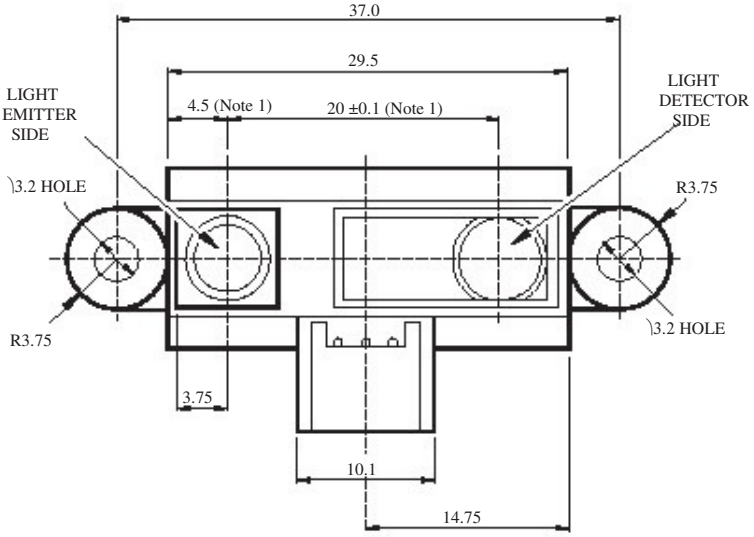
Figure 5. GP2D12 Example of Output Characteristics with Inverse Number of Distance

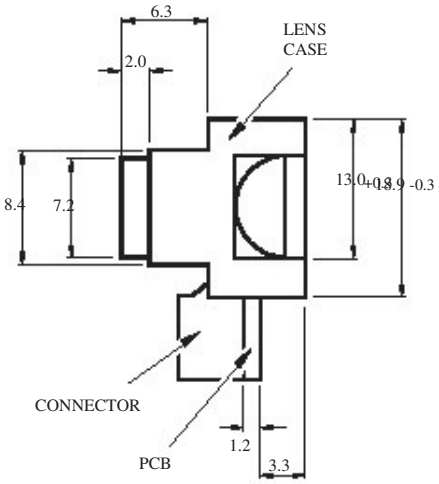
PACKAGE SPECIFICATIONS

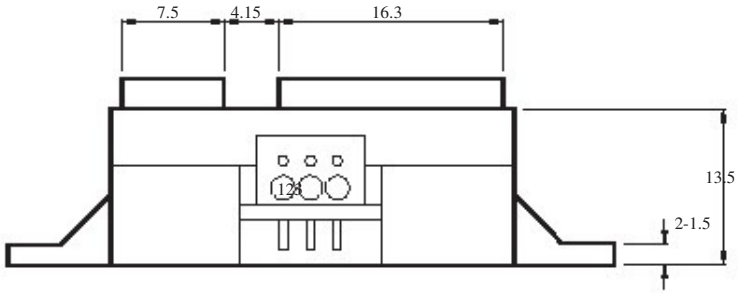


STAMP EXAMPLE









CONNECTOR SIGNAL

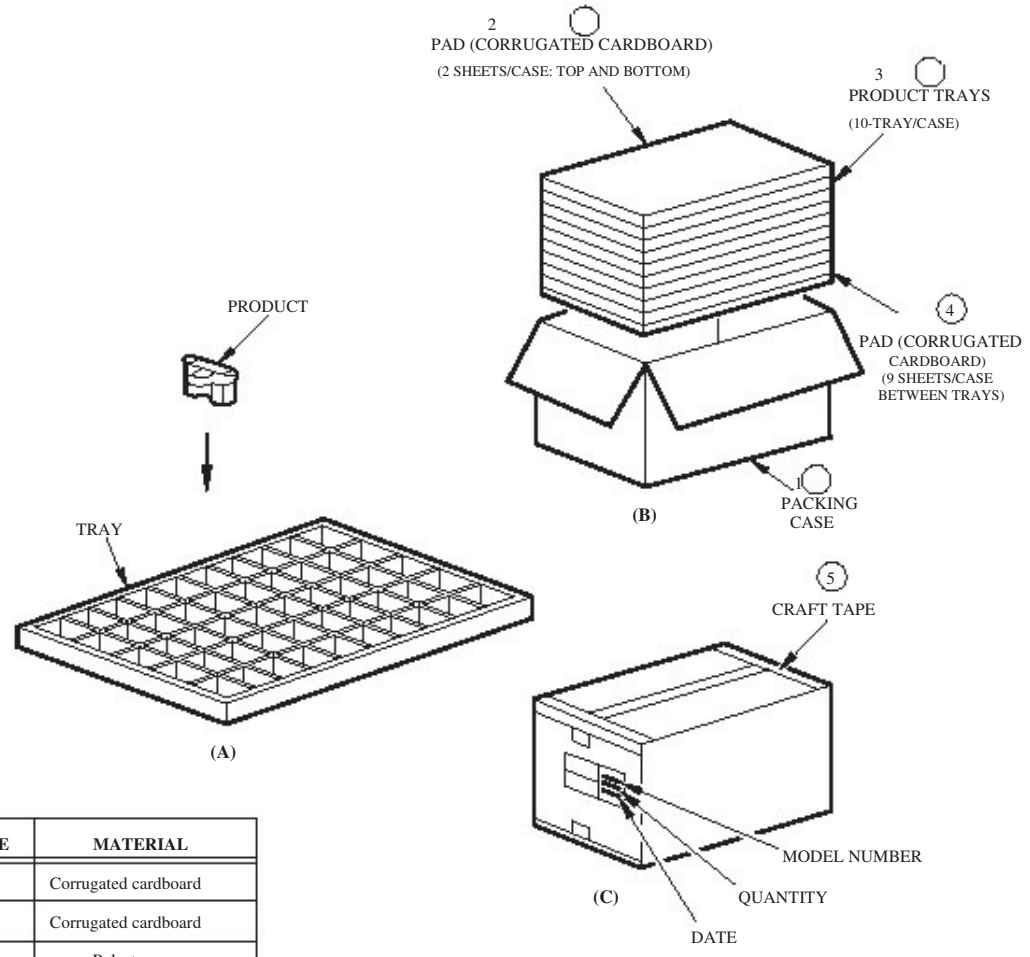
PIN	SIGNAL NAME
①	V_O
②	GND
③	V_{CC}

Connector: J.S.T. Trading Company, LTD
S3B-PH

NOTES:
1. Dimensions shall reference lens center.
2. Unspecified tolerance shall be ±0.3 mm.
3. Scale: 2/1, dimensions are in mm.

GP2D12-3

PACKING SPECIFICATION



PART NAME	MATERIAL
Packing case	Corrugated cardboard
Pad	Corrugated cardboard
Tray	Polystyrene

PACKING METHOD

1. Each tray holds 50 pieces. Packing methods are shown in (A).
2. Each box holds 10 trays. Pads are added to top and bottom, and between layers, as in (B).
   top and bottom. Put pads between each tray (9 pads total) see above drawing (B).
3. The box is sealed with craft tape. (C) shows the location of the Model number, Quantity, and Inspection date.
4. Package weight: Approximately 4 kg.

GP2Y0A02YK-8

NOTES

- Keep the sensor lens clean. Dust, water, oil, and other contaminants can deteriorate the characteristics of this device. Applications should be designed to eliminate sources of lens contamination.
- When using a protective cover over the emitter and detector, ensure the cover efficiently transmits light throughout the wavelength range of the LED (λ = 850 nm ± 70 nm). Both sides of the protective cover should be highly polished. Use of a protective cover may decrease the effective distance over which the sensor operates. Ensure that any cover does not negatively affect the operation over the intended application range.
- Objects in proximity to the sensor may cause reflections that can affect the operation of the sensor.
- Sources of high ambient light (the sun or strong artificial light) may affect measurement. For best results, the application should be designed to prevent interference from direct sunlight or artificial light.
- Using the sensor with a mirror can induce measurement errors. Often, changing the incident angle on the mirror can correct this problem.
- If a prominent boundary line exists in the surface being measured, it should be aligned vertically to avoid measurement error. See Figure 6 for further details.
- When measuring the distance to objects in motion, align the sensor so that the motion is in the horizontal direction instead of vertical. Figure 7 illustrates the preferred alignment.
- A 10 µF (or larger) bypass capacitor between V_{CC} and GND near the sensor is recommended.
- To clean the sensor, use a dry cloth. Use of any liquid to clean the device may result in decreased sensitivity or complete failure.
- Excessive mechanical stress can damage the internal sensor or lens.

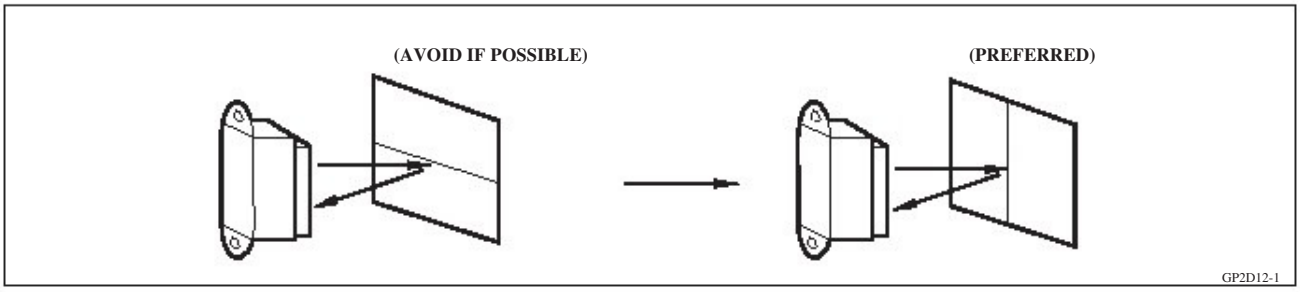


Figure 6. Proper Alignment to Surface Being Measured

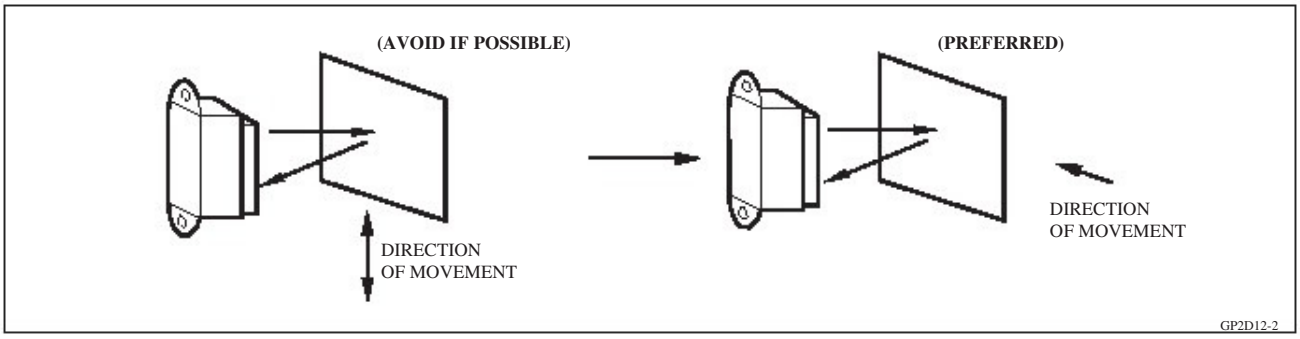


Figure 7. Proper Alignment to Moving Surfaces

NOTICE

The circuit application examples in this publication are provided to explain representative applications of SHARP devices and are not intended to guarantee any circuit design or license any intellectual property right. SHARP takes no responsibility for any problems related to any intellectual property right of a third party resulting from the use of SHARP devices.

SHARP reserves the right to make changes in the specifications, characteristics, data, materials, structures and other contents described herein at any time without notice in order to improve design or reliability.

Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device. Manufacturing locations are also subject to change without notice.

In the absence of confirmation by device specification sheets, SHARP takes no responsibility for any defects that occur in equipment using any SHARP devices shown in catalogs, data books, etc.

The devices listed in this publication are designed for standard applications for use in general electronic equipment. SHARP's devices shall not be used for or in connection with equipment that requires an extremely high level of reliability, such as military and aerospace applications, telecommunication equipment (trunk lines), nuclear power control equipment and medical or other life support equipment (e.g. Scuba). SHARP takes no responsibility for damage caused by improper use of device, which does not meet the conditions for use specified in the relevant specification sheet.

If the SHARP devices listed in the publication fall within the scope of strategic products described in the Foreign Exchange and Foreign Trade Law of Japan, it is necessary to obtain approval to export such SHARP devices.

Contact and consult with a SHARP representative if there are any questions about the contents of this publication.