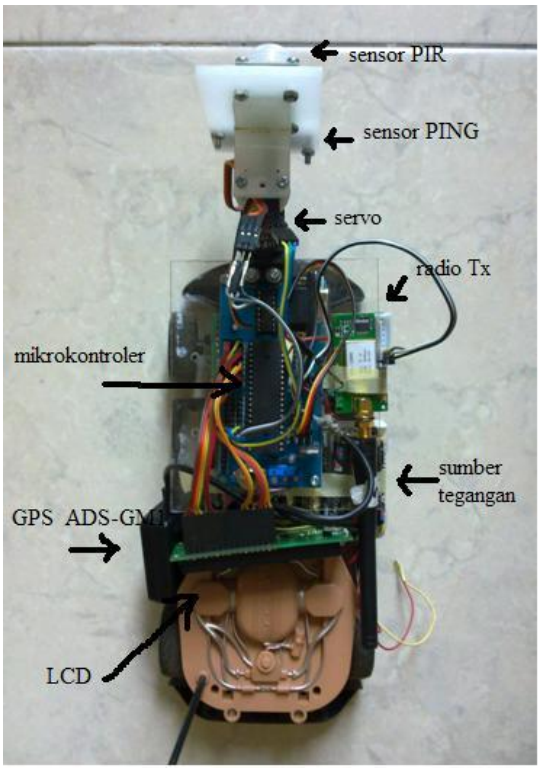
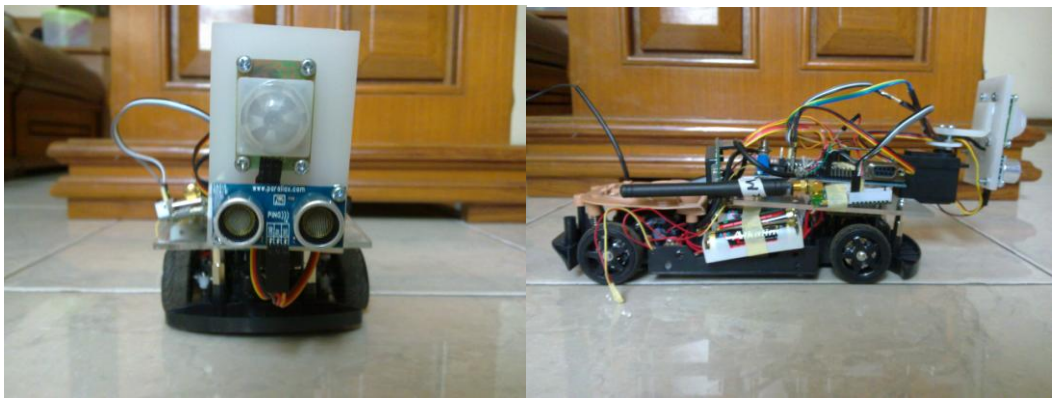
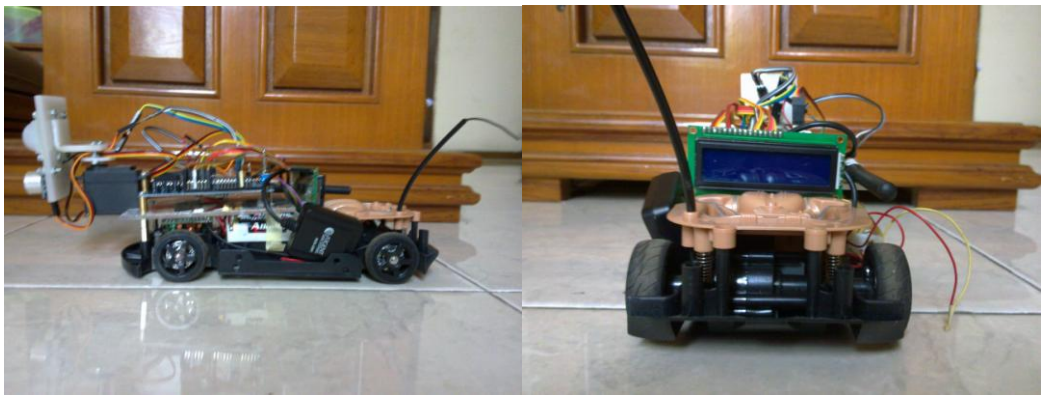
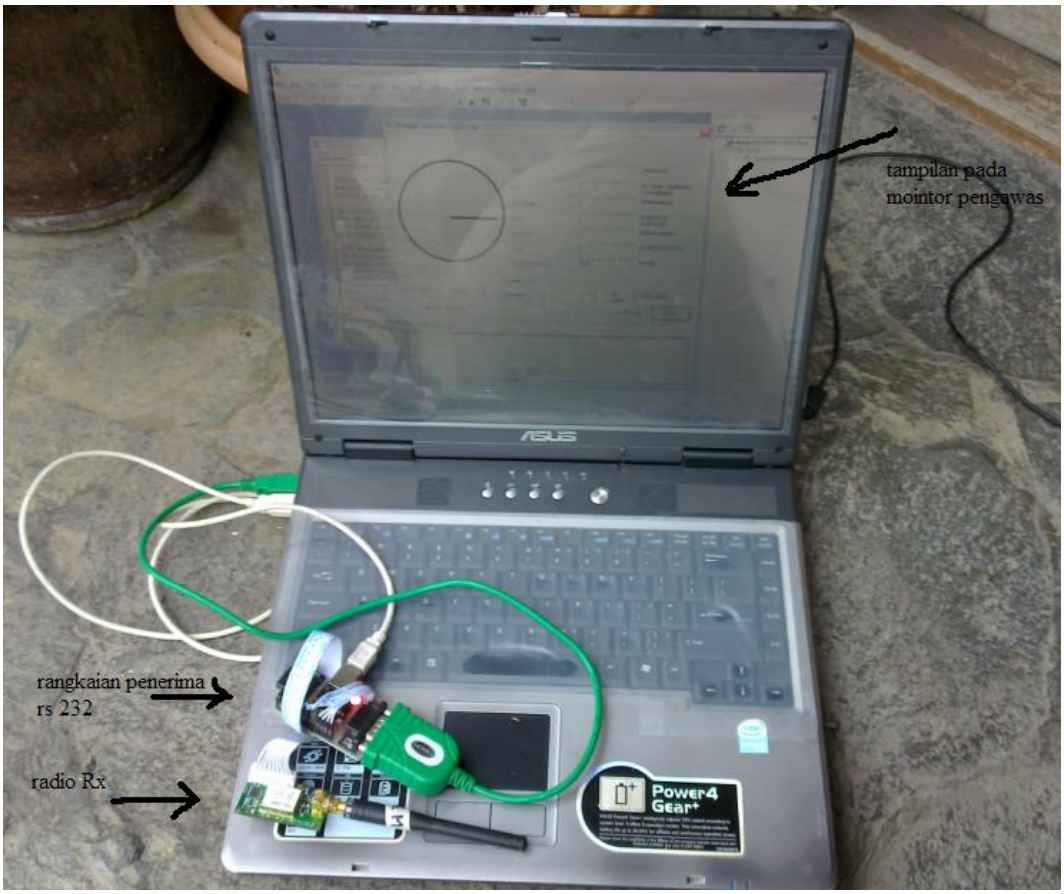
LAMPIRAN A
GAMBAR SISTEM



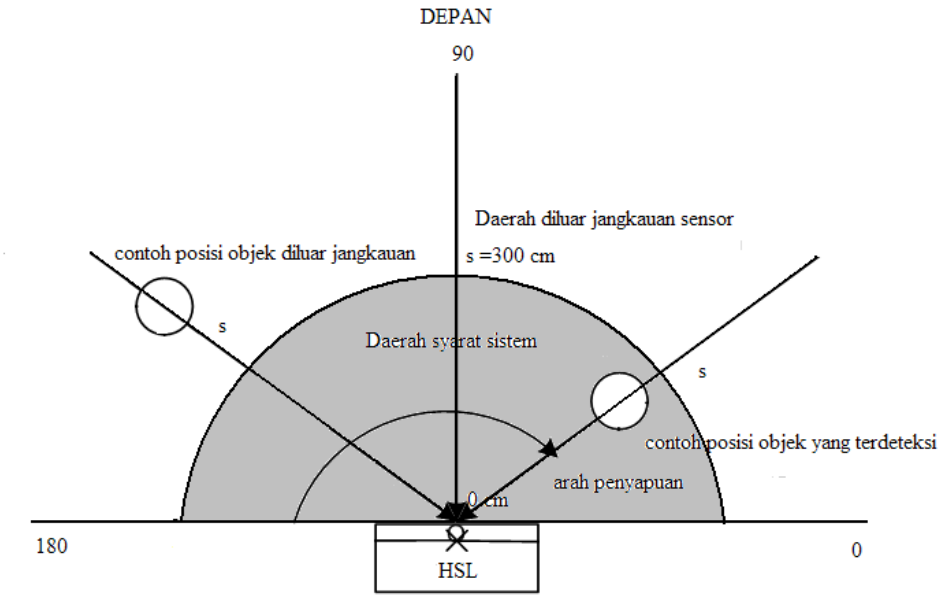




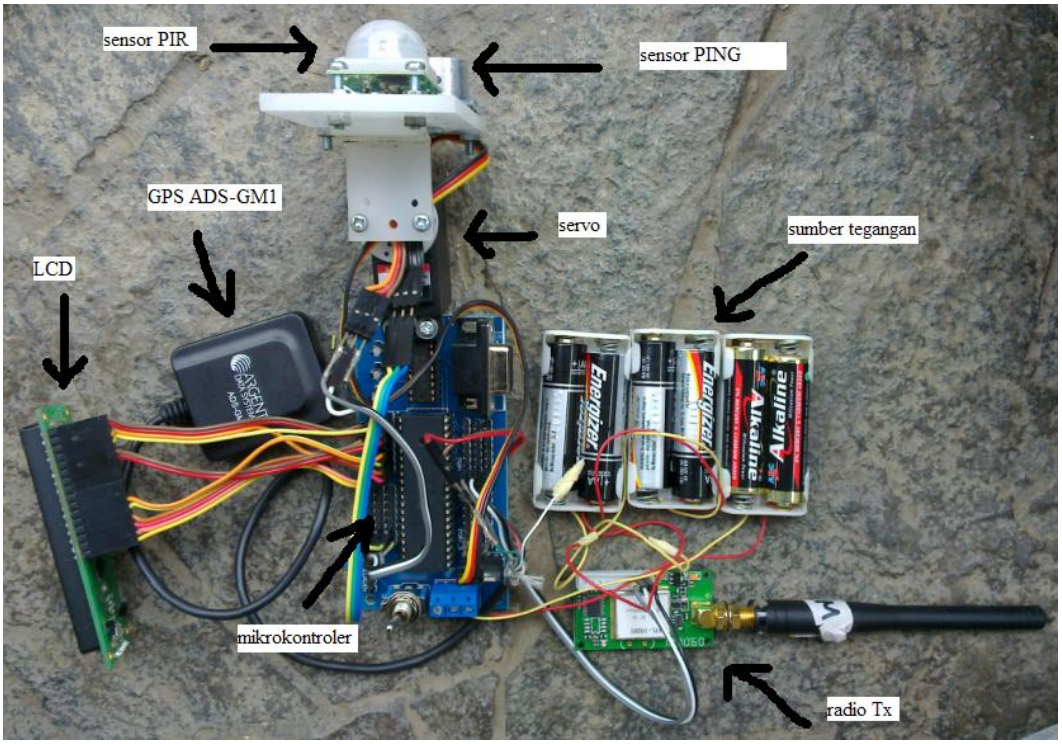
Perangkat Secara Keseluruhan



Penerima



Posisi objek terhadap sensor



Komponen Utama

LAMPIRAN B
PROGRAM AVR ATMEGA 16

```
/*****************************************************
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    : 11/16/2010
Author  : F4CG
Company : F4CG
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 <delay.h>
#include <stdio.h>
#include <lcd.h>
#include <string.h>

int t,s,count;
char text[32];
unsigned char string[100],text1[10];
unsigned char buf1,x;

#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)

// Get a character from the USART Receiver
#pragma used+

char getchar1(void)
{
char status,data;
```

```
while (1)
      {
      while (((status=UCSRA) & RX_COMPLETE)==0);
      data=UDR;
      if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
        return data;
      };
}

#pragma used-

// Write a character to the USART Transmitter
#pragma used+
void putchar1(char c)
{
while ((UCSRA & DATA_REGISTER_EMPTY)==0);
UDR=c;
}
#pragma used-

void sensor(void)
{
PORTA=0xff;
DDRA=0x00;

delay_ms(50000);

t=0;
      DDRA.0=1;
      PORTA.0=1;
      delay_us(5);
      PORTA.0=0;
      DDRA.0=0;
      PORTA.0=1;


      ulang:
      if(PINA.0==0)
      {
      delay_us(10);
      goto ulang;
      }


      while (PINA.0==1)
      {
      t++;
      delay_us(1);
      }
      s=0.034*t;
      sprintf(text," t=%d s=%d PIR=%d ",t,s,PINA.1);
      lcd_puts(text);
      delay_ms(500);
```

```
      lcd_clear();

      if (s<=300 && PINA.1==1)
      {
      lcd_putsf(" Kirim data S,PIR,GPS \n\r");

     buf1=0;
       while (buf1!='$') buf1=getchar1();
       getchar1();
       getchar1();
       getchar1();
       buf1=getchar1();

       if (buf1=='G')
       {
         for (x=0;x<2;x++)
         {
           buf1=0;
           while (buf1!=',') buf1=getchar1();
         }
         buf1=0;
         x=0;
         count=0;
         while (count<=3)
         {
           buf1=getchar1();
           string[x]=buf1;
           if(buf1==',')
           {
           count=count+1;
           }
           x++;
         }
         x--;
         string[x]=0;
         buf1=0;
       }
      }
}

// 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 0x00

// Read the AD conversion result
```

```
unsigned int 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 ADCW;
}

// Declare your global variables here

void main(void)
{

int i;

// 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=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
PORTC=0x00;
DDRC=0xFF;

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

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

```
TCNT0=0x00;
OCR0=0x00;

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

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

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

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

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud rate: 4800
```

```
UCSRA=0x00;
UCSRB=0x18;
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: AREF pin
// ADC Auto Trigger Source: None
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;

// LCD module initialization
lcd_init(16);

while (1)
      {
      // Place your code here

      for (i=0;i<50;i++)
      {
      PORTB=225;
      PORTB.1=1;
      delay_us(600);
      PORTB.1=0;
      delay_us(19400);
      }
      delay_ms(100);
      sensor();
      lcd_clear();
      sprintf(text1,"180 derajat %s",string);
      printf("S=%3d PIR=%d 180 derajat %s  \n\r",s,PINA.1,string);
      delay_ms(100);

      for (i=0;i<25;i++)
      {
      PORTB=225;
      PORTB.1=1;
      delay_us(700);
      PORTB.1=0;
      delay_us(19300);
      }
      delay_ms(100);
      sensor();
      lcd_clear();
```

```
sprintf(text1,"170 derajat %s",string);
printf("S=%3d PIR=%d 170 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(800);
PORTB.1=0;
delay_us(19200);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"160 derajat %s",string);
printf("S=%3d PIR=%d 160 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(900);
PORTB.1=0;
delay_us(19100);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"150 derajat %s",string);
printf("S=%3d PIR=%d 150 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(1000);
PORTB.1=0;
delay_us(19000);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"140 derajat %s",string);
printf("S=%3d PIR=%d 140 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
```

```
delay_us(1100);
PORTB.1=0;
delay_us(18900);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"130 derajat %s",string);
printf("S=%3d PIR=%d 130 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(1200);
PORTB.1=0;
delay_us(18800);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"120 derajat %s",string);
printf("S=%3d PIR=%d 120 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(1300);
PORTB.1=0;
delay_us(18700);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"110 derajat %s",string);
printf("S=%3d PIR=%d 110 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(1400);
PORTB.1=0;
delay_us(18600);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"100 derajat %s",string);
```

```
printf("S=%3d PIR=%d 100 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(1500);
PORTB.1=0;
delay_us(18500);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"90 derajat %s",string);
printf("S=%3d PIR=%d 090 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(1600);
PORTB.1=0;
delay_us(18400);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"80 derajat %s",string);
printf("S=%3d PIR=%d 080 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

{
for (i=0;i<25;i++)

PORTB=225;
PORTB.1=1;
delay_us(1700);
PORTB.1=0;
delay_us(18300);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"70 derajat %s",string);
printf("S=%3d PIR=%d 070 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
```

```
delay_us(1800);
PORTB.1=0;
delay_us(18200);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"60 derajat %s",string);
printf("S=%3d PIR=%d 060 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(1900);
PORTB.1=0;
delay_us(18100);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"50 derajat %s",string);
printf("S=%3d PIR=%d 050 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(2000);
PORTB.1=0;
delay_us(18000);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"40 derajat %s",string);
printf("S=%3d PIR=%d 040 derajat %s  \n\r",s,PINA.1,string);
delay_ms(100);

for (i=0;i<25;i++)
{
PORTB=225;
PORTB.1=1;
delay_us(2100);
PORTB.1=0;
delay_us(17900);
}
delay_ms(100);
sensor();
lcd_clear();
sprintf(text1,"30 derajat %s",string);
```

```
    printf("S=%3d PIR=%d 030 derajat %s  \n\r",s,PINA.1,string);
    delay_ms(100);

    for (i=0;i<25;i++)
    {
    PORTB=225;
    PORTB.1=1;
    delay_us(2200);
    PORTB.1=0;
    delay_us(17800);
    }
    delay_ms(100);
    sensor();
    lcd_clear();
    sprintf(text1,"20 derajat %s",string);
    printf("S=%3d PIR=%d 020 derajat %s  \n\r",s,PINA.1,string);
    delay_ms(100);

    for (i=0;i<25;i++)
    {
    PORTB=225;
    PORTB.1=1;
    delay_us(2300);
    PORTB.1=0;
    delay_us(17700);
    }
    delay_ms(100);
    sensor();
    lcd_clear();
    sprintf(text1,"10 derajat %s",string);
    printf("S=%3d PIR=%d 010 derajat %s  \n\r",s,PINA.1,string);
    delay_ms(100);

    for (i=0;i<25;i++)
    {
    PORTB=225;
    PORTB.1=1;
    delay_us(2400);
    PORTB.1=0;
    delay_us(17600);
    }
    delay_ms(100);
    sensor();
    lcd_clear();
    sprintf(text1,"0 derajat %s",string);
    printf("S=%3d PIR=%d 000 derajat %s  \n\r",s,PINA.1,string);
    delay_ms(100);

delay_ms(50);
    };
}
```

LAMPIRAN C
PROGRAM VISUAL BASIC

```
Public buff_ser As Byte

Option Explicit
Dim PanjangJarum As Integer                    ‘untuk panjang jarum
Dim PusatX As Integer                          ‘titik pusat
Dim PusatY As Integer

Sub HitungSkala()
Bingkai.Top = 600                              ‘top lingkaran
Bingkai.Left = 240                             ‘kiri lingkaran
PanjangJarum = (8 / 10 * 3495) \ 2 '(8 / 10 * Me.ScaleHeight) \ 2
'Hitung panjang jarum (80% dari diameter)  3495=tinggi dari pusat jari-jari

PusatX = 3500 \ 2                  'Hitung titik pusat Bingkai lebarnya
PusatY = 5000 \ 2                  'Hitung titik pusat Bingkai tingginya
End Sub

Private Sub AturJarum()
Dim SudutJarum As String
Dim x, y
SudutJarum = Text1(7).Text                     'Hitung sudut Jarum
x = PanjangJarum * Cos(SudutJarum * 3.14 / 180)   'Hitung koordinat Cartesius
y = PanjangJarum * Sin(SudutJarum * 3.14 / 180)
Jarum.X1 = PusatX                              'Atur jarum Jarum
Jarum.Y1 = PusatY
Jarum.X2 = PusatX + x
Jarum.Y2 = PusatY - y
End Sub

Private Sub Command10_Click()
RichTextBox1.SaveFile Text13.Text
MsgBox ("DATA TERSIMPAN")
```

```
End Sub

Private Sub Command5_Click()
If MSComm1.PortOpen = False Then MSComm1.PortOpen = True
End Sub

Private Sub Command6_Click()
If MSComm1.PortOpen = True Then MSComm1.PortOpen = False
End Sub

Private Sub Command8_Click()
End
End Sub

Private Sub Command9_Click()
MSComm1.CommPort = Text9.Text
MSComm1.Settings = Text12.Text & ",n,8,1"
MSComm1.Handshaking = comNone
End Sub

Private Sub Form_Load()
Call HitungSkala                    'Hitung Skala
Me.Visible = True
Bingkai.Visible = True              'Buat jadi Visible
Call AturJarum                      'Atur jarum
Jarum.Visible = True                'Buat jadi Visible
End Sub

Private Sub Form_Resize()
Call HitungSkala            'Hitung skala ketika terjadi perubahan ukuran Form
End Sub
```

```
Private Sub MSComm1_OnComm()
buff_ser = MSComm1.Input
End Sub

Private Sub RichTextBox1_Change()

End Sub

Private Sub Text1_Change(Index As Integer)

End Sub

Private Sub Text10_Change()

Dim a As String
Dim b As String
Dim c As String
Dim d As String
Dim e As String
Dim f As String
Dim g As String

a = Mid(Text10.Text, 3, 3)              ' indikasi PING
Text1(1).Text = a
b = Mid(Text10.Text, 11, 1)             ' indikasi PIR
Text1(2).Text = b
c = Mid(Text10.Text, 25, 9)             ' indikasi LATITUDE
Text1(3).Text = c
d = Mid(Text10.Text, 35, 1)             ' indikasi N/S
Text1(4).Text = d
e = Mid(Text10.Text, 37, 10)            ' indikasi LONGITUDE
Text1(5).Text = e
```

```
f = Mid(Text10.Text, 48, 1)              ' indikasi E/W
Text1(6).Text = f
g = Mid(Text10.Text, 13, 3)              ' indikasi SUDUT SERVO
Text1(7).Text = g

End Sub

Private Sub Text12_Change()

End Sub

Private Sub Text13_Change()

End Sub

Private Sub Text9_Change()

End Sub

Private Sub Timer1_Timer()

End Sub

Private Sub Timer2_Timer()
If MSComm1.PortOpen = True Then
Text10.Text = MSComm1.Input
RichTextBox1.Text = Time & " ; " & Text10.Text & " ; " & Chr$(10) &
RichTextBox1.Text
Call AturJarum
End If

End Sub
```

LAMPIRAN D
DATA SHEET

Penjelasan GPS NMEA 0813

NMEA-0183 adalah standar kalimat laporan yang dikeluarkan oleh GPS receiver. Standar NMEA memiliki banyak jenis bentuk kalimat laporan, di antaranya yang paling penting adalah koordinat lintang (latitude), bujur (longitude), ketinggian (altitude), waktu sekarang standar UTC (UTC time), dan kecepatan (speed over ground). Penjelasan lengkap www.SiRF.com.

Output GPS :

$GPGGA,002153.000,3342.6618,N,11751.3858,W,1,10,1.2,27.0,M,-34.2,M,,0000*5E

Name	Example	Unit	Description
Message ID	$GPGGA		GGA protocol header
UTC Time	002153.000		hhmmss.sss
Latitude	3342.6618		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	11751.3858		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Position Fix Indicator	1		See Table 1-4
Satellites Used	10		Range 0 to 12
HDOP	1.2		Horizontal Dilution of Precision
MSL Altitude	27.0	meters	
Units	M	meters	
Geoid Separation	-34.2	meters	Geoid-to-ellipsoid separation. Ellipsoid altitude = MSL Altitude + Geoid Separation.
Units	M	meters	
Age of Diff. Corr.		sec	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*5E		
<CR> <LF>			End of message termination

$GPGLL,3723.2475,N,12158.3416,W,161229.487,A,A*41

Name	Example	Unit	Description
Message ID	$GPGLL		GLL protocol header
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
UTC Time	161229.487		hhmmss.sss
Status	A		A=data valid or V=data not valid
Mode	*A*		*A=Autonomous, D=DGPS, E=DR* *N = Output Data Not Valid*
Checksum	*41		
<CR> <LF>			End of message termination

$GPGSA,A,3,07,02,26,27,09,04,15, , , , , ,1.8,1.0,1.5*33

Name	Example	Unit	Description
Message ID	$GPGSA		GSA protocol header
Mode 1	A		See Table 1-7
Mode 2	3		See Table 1-8
Satellite Used[1]	07		SV on Channel 1
Satellite Used[1]	02		SV on Channel 2
....			....
Satellite Used[1]			SV on Channel 12
PDOP[2]	1.8		Position Dilution of Precision
HDOP[2]	1.0		Horizontal Dilution of Precision
VDOP[2]	1.5		Vertical Dilution of Precision
Checksum	*33		
<CR> <LF>			End of message termination

$GPGSV,2,1,07,07,79,048,42,02,51,062,43,26,36,256,42,27,27,138,42*71

Name	Example	Unit	Description
Message ID	$GPGSV		GSV protocol header
Number of Messages[1]	2		Total number of GSV messages to be sent in this group
Message Number[1]	1		Message number in this group of GSV messages
Satellites in View[1]	07		
Satellite ID	07		Channel 1 (Range 1 to 32)
Elevation	79	degrees	Channel 1 (Maximum 90)
Azimuth	048	degrees	Channel 1 (True, Range 0 to 359)
SNR (C/N0)	42	dBHz	Range 0 to 99, null when not tracking
....			....
Satellite ID	27		Channel 4 (Range 1 to 32)
Elevation	27	degrees	Channel 4 (Maximum 90)
Azimuth	138	degrees	Channel 4 (True, Range 0 to 359)
SNR (C/N0)	42	dBHz	Range 0 to 99, null when not tracking
Checksum	*71		
<CR> <LF>			End of message termination

$GPMSS,55,27,318.0,100,1,*57

Name	Example	Unit	Description
Message ID	$GPMSS		MSS protocol header
Signal Strength	55	dB	SS of tracked frequency
Signal-to-Noise Ratio	27	dB	SNR of tracked frequency
Beacon Frequency	318.0	kHz	Currently tracked frequency
Beacon Bit Rate	100		bits per second
Channel Number	*1*		*The channel of the beacon being used if a multi-channel beacon receiver is used*
Checksum	*57		
<CR> <LF>			End of message termination

$GPRMC,161229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598, ,*10

Name	Example	Unit	Description
Message ID	$GPRMC		RMC protocol header
UTC Time	161229.487		hhmmss.sss
Status[1]	A		A=data valid or V=data not valid
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Speed Over Ground	0.13	knots	
Course Over Ground	309.62	degrees	True
Date	120598		ddmmyy
Magnetic Variation[2]		degrees	E=east or W=west
East/West Indicator[2]	E		E=east
Mode	*A*		*A=Autonomous, D=DGPS, E=DR, N = Output Data Not Valid*
Checksum	*10		
<CR> <LF>			End of message termination

$GPVTG,309.62,T, ,M,0.13,N,0.2,K,A*23

Name	Example	Unit	Description
Message ID	$GPVTG		VTG protocol header
Course	309.62	degrees	Measured heading
Reference	T		True
Course		degrees	Measured heading
Reference	M		Magnetic[1]
Speed	0.13	knots	Measured horizontal speed
Units	N		Knots
Speed	0.2	km/hr	Measured horizontal speed
Units	K		Kilometers per hour
Mode	*A*		*A=Autonomous, D=DGPS, E=DR, N = Output Data Not Valid*
Checksum	*23		
<CR> <LF>			End of message termination

$GPZDA,181813,14,10,2003,,*4F

Name	Example	Unit	Description
Message ID	$GPZDA		ZDA protocol header
UTC time	181813	hhmmss	The UTC time units are as follows: hh = UTC hours from 00 to 23 mm = UTC minutes from 00 to 59 ss = UTC seconds from 00 to 59 Either using valid IONO/UTC or estimated from default leap seconds
Day	14		Day of the month, range 1to 31
Month	10		Month of the year, range 1 to 12
Year	2003		1980 to 2079
Local zone hour[1]		hour	Offset from UTC (set to 00)
Local zone minutes[1]		minute	Offset from UTC (set to 00)
Checksum	*4F		
<CR> <LF>			End of message termination

ARGENT DATA SYSTEMS

Top » Catalog » GPS Receivers »

Categories

Trackers and TNCs (9)
Weather (1)
GPS Receivers (4)
GPS Accessories (7)
Radios (19)
Cables and Connectors (44)
Simplex Repeater (7)
Other Parts (19)

What's New?

Garmin 010-11232-00 FMI Cable
$45.00

Quick Find

Use keywords to find the product you are looking for.
Advanced Search

Information

Shipping & Returns
Privacy Notice
Conditions of Use
Gift Voucher FAQ
Contact Us

ADS-GM1 GPS Receiver

The ADS-GM1 is a high-sensitivity GPS receiver based on the SiRFstar III chipset. It has a 2-meter long cable terminated in a female DB9 connector that allows direct connection to an OpenTracker+, Tracker2, or any other device that provides regulated 5-volt power on pin 4.

- SiRFstar III high-sensitivity chipset
- Waterproof housing (IPX7 rating)
- Built-in low noise, high gain antenna
- Magnetic base
- LED clearly indicates positioning status
- No configuration required
- 20 channel all-in-view tracking

Sensitivity: -159 dBm typical
Accuracy: < 10 meters 2D RMS, < 7 meters WAAS corrected, time to 1 microsecond
Datum: WGS84
Acquisition Rate: 1 sec hot start, 42 sec cold start
Dynamic Limits: < 18,000 meters, < 1000 knots, < 4G acceleration
Power Supply: 5 VDC +/- 5%, 80 mA max, 55 mA typical
Interface: NMEA-0183 at 4800 baud, optional SiRF binary
NMEA Messages: GGA, GSA, GSV, RMC, and optionally VTG, GLL, and ZDA
Weight: 85 grams

LED flashes red to indicate valid GPS fix. Made in Taiwan.

Pin	Wire	Function
2	Red	RS-232 data out (+/- 6v)
3	Brown	RS-232 data in
4	Green	Power in
5	Blue	Ground
-	Yellow	TTL data out (0-3v)
-	Black	TTL data in

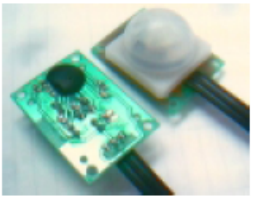
RoHS Compliant

KC7783R
PIR Module
Low Cost version

This is a low cost version of the PIR module. It is designed for cost sensitive consumer product.
Except the IC package format, all the mechanical and electrical spec is same as KC7783.

Features:

- ☐ IC soft package by dice bonding technique
- ☐ Small size: 25 x 35mm
- ☐ Ball lens is included as standard configuration
- ☐ 3 leads flat cable for easy connection
- ☐ 4 mounting holes on board
- ☐ High Sensitivity
- ☐ High immunity to RFI
- ☐ Power up delay to prevent from false triggering
- ☐ Output High for direct connect to control panel

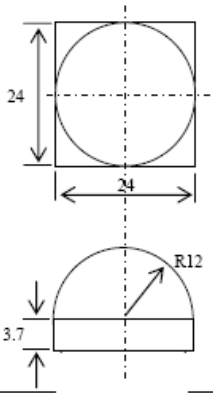


Specification

	Min	Typ	Max	Unit
Operation Voltage	4.7	5	12	V
Standby Current (no load)		300		μA
Output Pulse Width	0.5			Sec
Output High Voltage		5		V
Detection Range		5		M
Operation Temperature	-20	25	50	°C
Humidity Range			95	%

Note: 1. All other features and specification, please refer to KC778B
2. Minimum output pulse width can be customer specified.

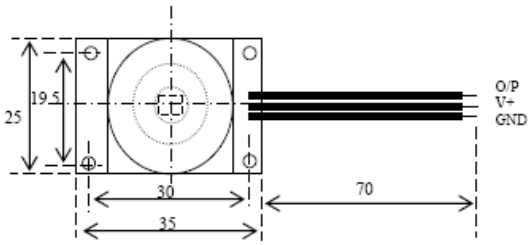
Lens Dimension (unit in mm)



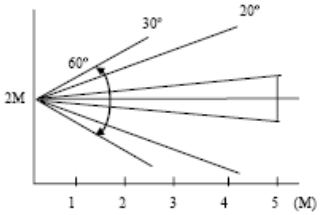
Standard Configuration

PIR controller	KC778B in dice form
PIR Sensor	RE200B by NICERA
Lens	Ball lens of 60° detection angle
Connector	3 leads flat cable, Power, GND, O/P

Mechanical Dimension



Vertical View Pattern



Application Note:

1. The PIR sensor is sensitive to the temperature change and therefore to prevent from operating the module in rapid environmental temperature changes, strong shock or vibration. Don't expose to the direct sun light or headlights of automobile. Don't expose to direct wind from heater or air conditioner.
2. This module is designed for indoor use. If using in outdoor, make sure to apply suitable supplemental optical filter and drop-proof, anti-dew construction
3. Detection range might be varied in different environmental temperature condition.

599 Menlo Drive, Suite 100
Rocklin, California 95765, USA
Office: (916) 624-8333
Fax: (916) 624-8003

General: info@parallax.com
Technical: support@parallax.com
Web Site: www.parallax.com
Educational: www.stampsinclass.com

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

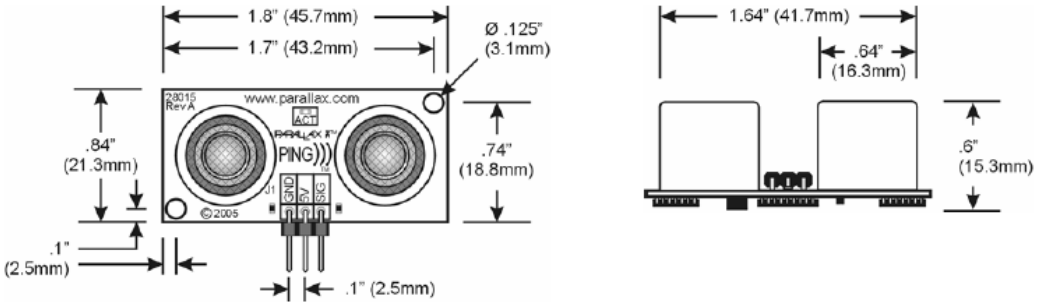
The Parallax PING))) ultrasonic distance sensor provides precise, non-contact distance measurements from about 2 cm (0.8 inches) to 3 meters (3.3 yards). It is very easy to connect to BASIC Stamp® or Javelin Stamp microcontrollers, requiring only one I/O pin.

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

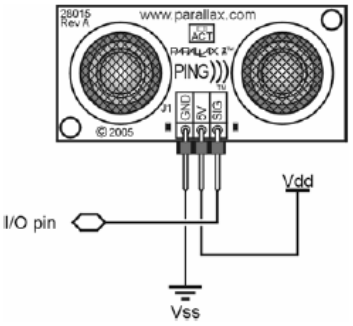
- Supply Voltage – 5 VDC
- Supply Current – 30 mA typ; 35 mA max
- Range – 2 cm to 3 m (0.8 in to 3.3 yrds)
- Input Trigger – positive TTL pulse, 2 uS min, 5 µs typ.
- Echo Pulse – positive TTL pulse, 115 uS to 18.5 ms
- Echo Hold-off – 750 µs from fall of Trigger pulse
- Burst Frequency – 40 kHz for 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)

Dimensions



Pin Definitions

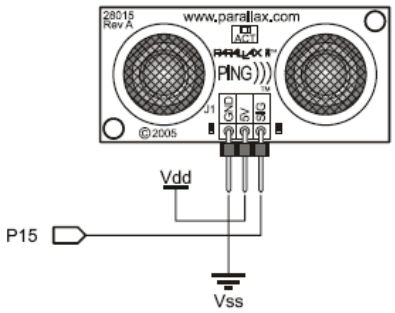
GND	Ground (Vss)
5 V	5 VDC (Vdd)
SIG	Signal (I/O pin)

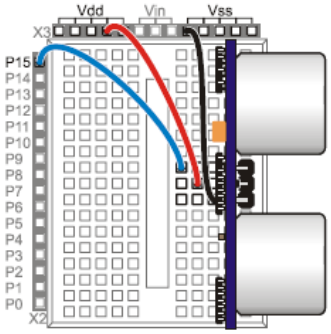


The PING))) sensor has a male 3-pin header used to supply power (5 VDC), ground, and signal. The header allows the sensor to be plugged into a solderless breadboard, or to be located remotely through the use of a standard servo extender cable (Parallax part #805-00002). Standard connections are show 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 program Ping_Demo.BS2 listed on page 7.

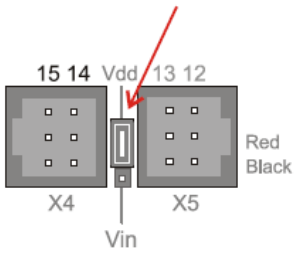




Servo Cable and Port Cautions

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

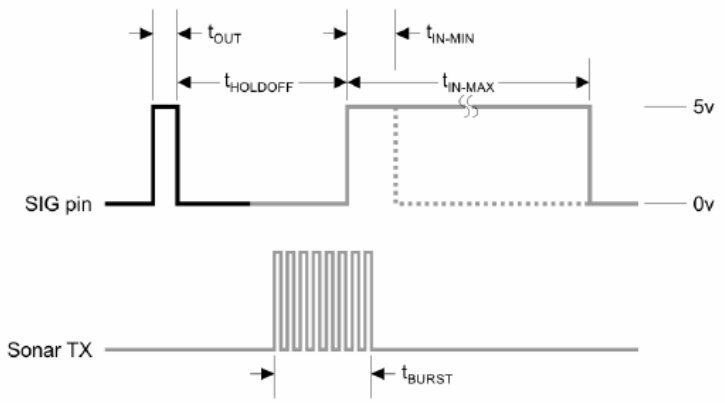
1. When plugging the cable onto the PING))) sensor, connect Black to GND, Red to 5 V, and White to SIG.
2. Check to see if your Board of Education servo ports have a jumper, as shown at right.
3. If your Board of Education servo ports have a jumper, set it to Vdd as shown.
4. If your Board of Education 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 servo cable directly to the breadboard with a 3-pin header. 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.



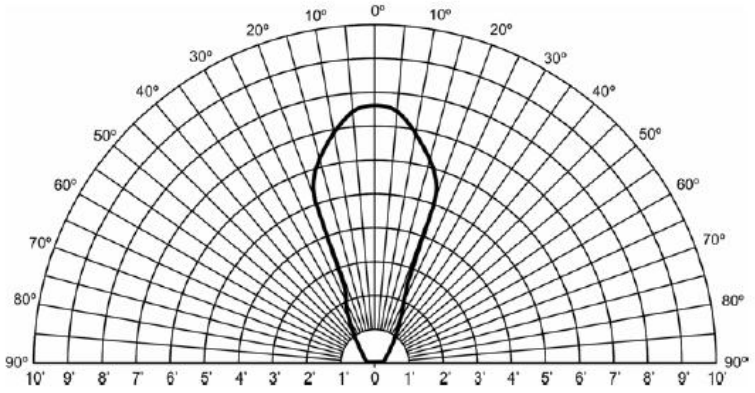
HOST	t_{OUT}	2 uS (min), 5 uS typical
PING	$t_{HOLDOFF}$	750 uS
	t_{BURST}	200 uS @ 40 kHz
	t_{IN-MIN}	115 uS
	t_{IN-MAX}	18.5 mS

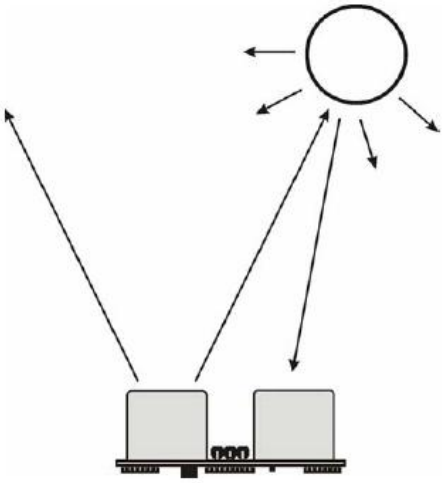
Test Data

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

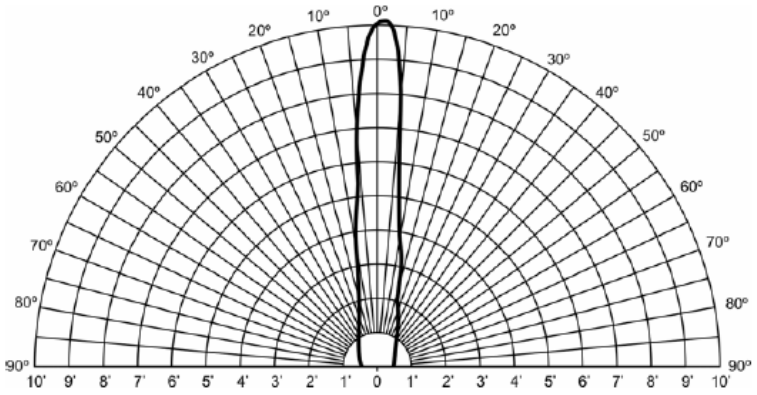
Sensor Elevation: 40 in. (101.6 cm)
Target: 3.5 in. (8.9 cm) diameter cylinder, 4 ft. (121.9 cm) tall – vertical orientation

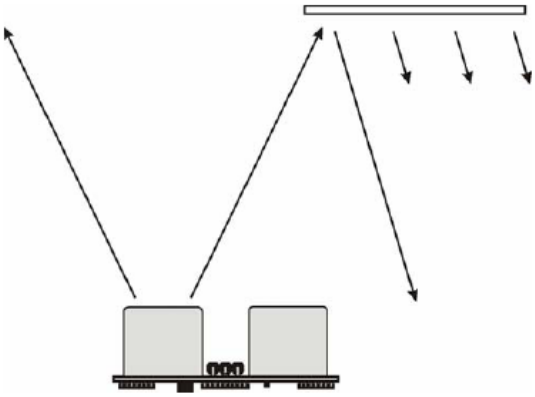




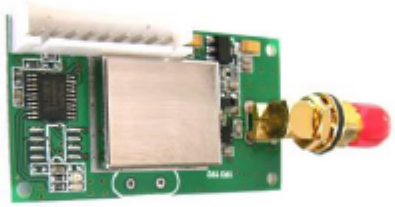
Test 2

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





YS-1020UB RF Data Transceiver



YS-1020 series Low power RF modules designed for the professional UART data transmission systems in short range. YS-1020 adopt Texas Instruments (Chipcon) CC1020 RF IC, works on ISM frequency band, half duplex integrated receiving and transmitting. Modules could directly connect with monolithic processors, PC, RS485 devices, and other UART components with RS-232, RS-485 and UART/TTL level interface port. Transparent data interface, nakedness, and wide temperature design handles most industrial application though indoor/outdoor environments.

1. Products Main Features

* Carrier frequency: 433/450/868/915MHz or ISM others optional, Free License;
* Interface: RS-232/ RS-485/ TTL optional;
* Multi-channels: 8 channels, expandable for 16/32 channels;
* Baud rate in air: 1200/2400/4800/9600/19200/38400bps, set before delivery;
* Transparent data transmission: What has been received is exactly what has been transmitted, suitable for any standard or non-standard user protocols;
* Interface format: 8N1/8E1/801 user-defined, or customization for other format interface;
* Modulation: GFSK. Based on the Gaussian Frequency Shift Keying (GFSK) modulation, High anti-interference and Low BER (Bit error Rate);
* Half duplex: Integration of receiver and transmitter，10ms auto change for receiving and sending;
* Low power consumption and sleep function;
* Widen Temperature: -35℃~+75℃ (-31~167 F);
* Working humidity: 10%~90% relative humidity without condensation;
* Impedance：50Ω (SMA antenna port, multiple antenna options available);
* Complying with EN 300220 and ARIB STD-T67.

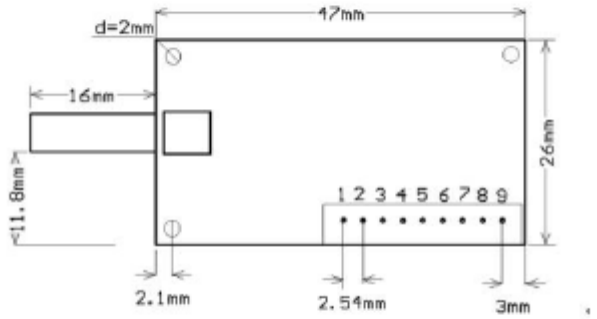
2. Application areas

* Automatic meter reading(AMR) and home automation ;
* Wireless smart terminal: POS, PDA,
* Wireless electronic display screen, LED display;
* Wireless remote control, Environment monitor, telemetry system;
* Check attendance system, Queue-management system and positioning in coal mine;
* RS-485 wire multi-drop System changeover wireless system;
* Industrial automatic data collection, Wireless Data Acquisition, Wireless sensor, SCADA;

3. Specifications

* RF power: ≤50mW/ 17dBm;
* Receiving current: <25mA;
* Transmitting current: <55mA;
* Sleep current: <20uA;
* Power supply: DC 5v or 3.3V;
* Receiving sensitivity: -115 dBm (@9600bps)
  -120 dBm (@1200bps);
* Size: 47mm×26mm×10mm (without antenna port);
* Range: ≤0.8Km (BER=10-^{3}@9600bps，when antenna is 2m above ground in open area),
  ≤1 Km (BER=10-^{3}@1200bps，when antenna is 2m above ground in open area).

4. Installation dimension:



5. Interface definition:

Pin	Pin name	Description	Level	Connection with terminal	Remands
1	GND	Grounding of power supply		Ground	
2	Vcc	Power supply DC	+3.3~5.5V		
3	RXD/TTL	Serial data receiving end	TTL	TxD	
4	TXD/TTL	Serial data transmitting end	TTL	RxD	
5	DGND	Digital grounding			
6	A(TXD)	A of RS-485 or TXD of RS-232		A(RxD)	
7	B(RXD)	B of RS-485 or RXD of RS-232		B(TxD)	
8	Sleep	Sleep control (input)	TTL	Sleep signal	Low level sleep
9	Test	Ex-factory testing			

NOTE: Generally the module is in receiving status, if the Sleep pin (No.8) continuously connects low level (>200millisecond), the module will be in sleep status, modules can not receive or transmit any data when sleep.

Only when the Sleep pin set in the state of high level (Vi=3.5V) or hangs/empty, module can be in receiving status again. The delay time for conversion between sleeping and receiving is less than 150mS.

6. Setting of channel, interface, and data format:

User can change or view the module's parameter setting (interface baud rate and channel) by testing software "YSPRG.EXE" in the CD (Free). Channel 6 is default value.

1) Corresponding frequency point: at 433MHz of 1~8 channel:

Channel	Frequency	Channel	Frequency	Channel	Frequency	Channel	Frequency
1	429.030SMHZ	2	430.030SMHZ	3	431.030SMHZ	4	432.030SMHZ
5	433.030SMHZ	6	434.030SMHZ	7	435.030SMHZ	8	436.030SMHZ

2) Corresponding frequency point: at 868MHz of 1~8 channel:

Channel	Frequency	Channel	Frequency	Channel	Frequency	Channel	Frequency
1	867.030SMHZ	2	868.030SMHZ	3	869.030SMHZ	4	870.030SMHZ
5	871.030SMHZ	6	872.030SMHZ	7	873.030SMHZ	8	874.030SMHZ

7. Antenna configuration:

Many appropriative antennas for low power RF modules are selected for meeting different user antenna configurations. Please ask our Sales office for further information about the antenna's dimension and performance. The main options of antennas are exterior flagelliform rubber antenna with helical SMA joint, magnetic car antenna.

Standard: A0# Helical SMA antennas, L0# 9pin line

Notes:

◢ Modules can share DC power supply with other equipment, Ensure the supply is stable (ideally <10mVpk ripple).

◢ Keep the module away from other EMF generating components.

◢ Match 50Ω, 1/4wave antenna, high mount the antenna as close to the module as possible. Set antenna more than 2m above ground in open area to reach optimal range.