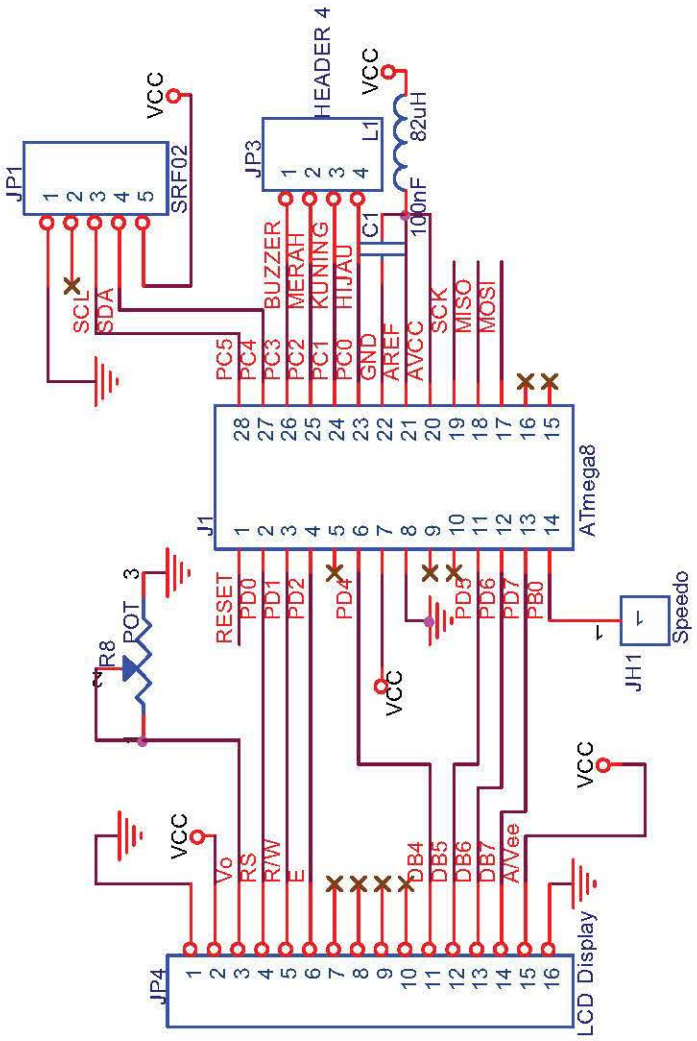
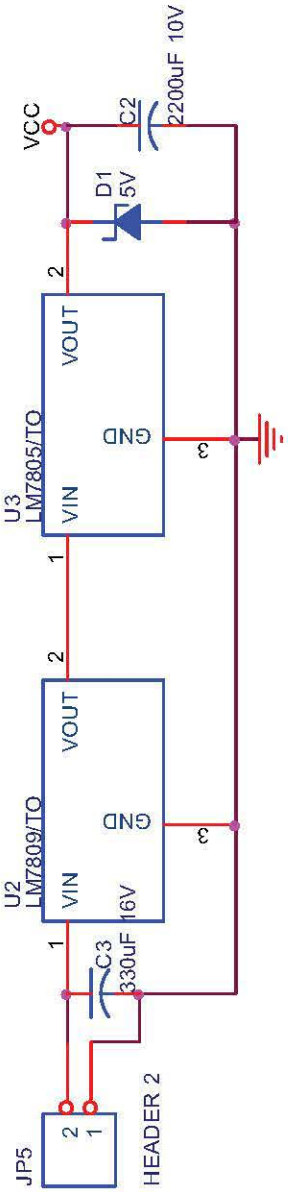
LAMPIRAN 1

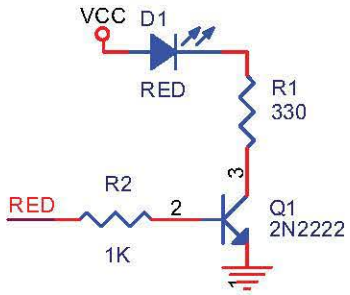
GAMBAR SKEMATIK LENGKAP



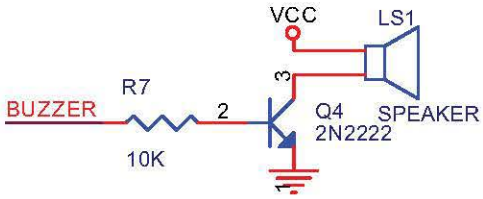
Gambar L.1 Rangkaian Utama (LCD, ATmega8, SRF02)



Gambar L.2 Rangkaian *Power Supply (Regulator)*



Gambar L.3 Rangkaian *driver* LED



Gambar L.4 Rangkaian *driver Piezo-buzzer*

LAMPIRAN 2

LISTING PROGRAM

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.4 Standard
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech
s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 16/07/2010
Author  :
Company :
Comments:


Chip type           : ATmega8
Program type        : Application
Clock frequency     : 8,000000 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 256
*****************************************************/

#include <mega8.h>

// I2C Bus functions
#asm
   .equ __i2c_port=0x15 ;PORTC
   .equ __sda_bit=4
   .equ __scl_bit=5
#endasm
#include <i2c.h>

// Alphanumeric LCD Module functions
#asm
   .equ __lcd_port=0x12 ;PORTD
```

```
#endasm


#include <lcd.h>
#include <delay.h>
#include <io.h>


#define ADC_VREF_TYPE 0xC0


// Declare your global variables here
unsigned int cacah;

// Timer 1 input capture interrupt service routine
interrupt [TIM1_CAPT] void timer1_capt_isr(void)
{
// Place your code here
cacah = ICR1;
TCNT1H=0x00;
TCNT1L=0x00;

}


void main(void)
{
// Declare your local variables here
unsigned char a,b;
unsigned int frek0,kec,Srl,Sidl,Srat;


// Input/Output Ports initialization
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In
Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T
State2=T State1=P State0=T
PORTB=0x02;
DDRB=0x00;

// Port C initialization
```

```
// Func7=In Func6=In Func5=In Func4=In Func3=Out
Func2=Out Func1=Out Func0=Out
// State7=T State6=T State5=T State4=T State3=0
State2=0 State1=0 State0=0
PORTC=0x00;
DDRC=0x0F;

// 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
TCCR0=0x00;
TCNT0=0x00;


// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 31,250 kHz
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Rising Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: On
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x44;
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
MCUCR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x20;
cacah = ICR1;


// I2C Bus initialization
i2c_init();

if(PINB.1==0)
{
// LCD module initialization
lcd_init(16);

lcd_gotoxy(0,0);
lcd_putsf("SAFETY DISTANCE ");
lcd_gotoxy(0,1);
lcd_putsf("INDICATOR(SED-i)");
delay_ms(3500);

lcd_gotoxy(0,0);
lcd_putsf("Oswin Arinaga S.");
lcd_gotoxy(0,1);
lcd_putsf("   5103006002   ");
```

```
delay_ms(2500);

lcd_gotoxy(0,1);
lcd_putsf("S=");
}

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

while (1)
      {


          //Pengukuran jarak melalui SRF02
            i2c_start();
            i2c_write(0xE0);
            i2c_write(0x00);
            i2c_write(0x51);
            i2c_stop();
            delay_ms(70);

            i2c_start();
            i2c_write(0xE0);
            i2c_write(0x02);
            i2c_stop();
            delay_us(10);

            i2c_start();
            i2c_write(0xE1);
            a = i2c_read(1);
            b = i2c_read(0);
            i2c_stop();

            Srl = (unsigned int)a * 256 + b;


            if(PINB.1==0)
            {
            lcd_gotoxy(2,1);
            lcd_putchar(Srl/100 %10 + 0x30);
            lcd_putchar(Srl/10 %10 + 0x30);
            lcd_putchar(Srl %10 + 0x30);
```

```
            lcd_putchar('c');
            lcd_putchar('m');
            lcd_putchar(' ');
            }

            frek0 = 31250/cacah;
            kec = (frek0-10)/11;

            if(kec>150)
            {
                kec = 0;
            }

//Penghitungan jarak ideal berdasarkan kecepatan
Sidl = kec;

//Penghitungan rasio jarak
if((kec>0)&&(kec<151))
{
     Srat = (Srl*10)/Sidl;
}

     if(PINB.1==0)
     {
     lcd_gotoxy(8,1);
     lcd_putsf("v=");
     lcd_putchar(kec/100 %10 + 0x30);
     lcd_putchar(kec/10 %10 + 0x30);
     lcd_putchar(kec %10 + 0x30);
     lcd_putchar('k');
     lcd_putchar('p');
     lcd_putchar('h');
     delay_ms(250);
     }

     //Proses indikasi status
     if(kec==0)
     {
          if(PINB.1==0)
          {
               lcd_gotoxy(0,0);
               lcd_putsf("    BERHENTI    ");
```

```
    }
    PORTC = 0b00000111;      //LED ON diam
}                            //+ buzzer OFF
else if((kec<=150)&&(kec>60))
{
    if(PINB.1==0)
    {
         lcd_gotoxy(0,0);
         lcd_putsf("LEBIH DR KEC MAX");
    }
    PORTC = 0b00001111;      //LED ON berkedip
    delay_ms(100);           //+buzzer beep 4x
    PORTC &=0b00000111;
    delay_ms(100);
    PORTC |=0b00001111;
    delay_ms(100);
    PORTC &=0b00000111;
    delay_ms(100);
    PORTC |=0b00001111;
    delay_ms(100);
    PORTC = 0b00000000;
    delay_ms(100);
    PORTC |=0b00001000;
    delay_ms(100);
    PORTC &=0b00000000;
    delay_ms(100);
}
else if((Srl>600)||(Srl<15))
{
    if(PINB.1==0)
    {
         lcd_gotoxy(0,0);
         lcd_putsf("JRK TAK TERUKUR ");
    }
    PORTC = 0b00001111;      //LED ON berkedip
    delay_ms(100);           //+buzzer beep 4x
    PORTC &=0b00000111;
    delay_ms(400);
    PORTC |=0b00001111;
    delay_ms(100);
    PORTC &=0b00000111;
    delay_ms(400);
```

```
    PORTC |=0b00001111;
    delay_ms(100);
    PORTC = 0b00000000;
    delay_ms(400);
    PORTC |=0b00001000;
    delay_ms(100);
    PORTC &=0b00000000;
    delay_ms(400);
}
else if(Srat>=100)
{
    if(PINB.1==0)
    {
         lcd_gotoxy(0,0);
         lcd_putsf("    = AMAN =    ");
    }
    PORTC = 0b00000001;     //LED Hijau ON
}                           //+ buzzer OFF
else if((Srat<100)&&(Srat>=70))
{
    if(PINB.1==0)
    {
         lcd_gotoxy(0,0);
         lcd_putsf(" = HATI-HATI! = ");
    }
    PORTC = 0b00001010;     //LED Kuning ON
    delay_ms(150);          //+ buzzer 2x
    PORTC &=0b00000010;
    delay_ms(100);
    PORTC |= 0b00001010;
    delay_ms(150);
    PORTC &= 0b00000010;
    delay_ms(900);
}
else
{
    if(PINB.1==0)
    {
         lcd_gotoxy(0,0);
         lcd_putsf(" = BAHAYA!!! =  ");
    }
    PORTC = 0b00001100;     //LED Merah ON
```

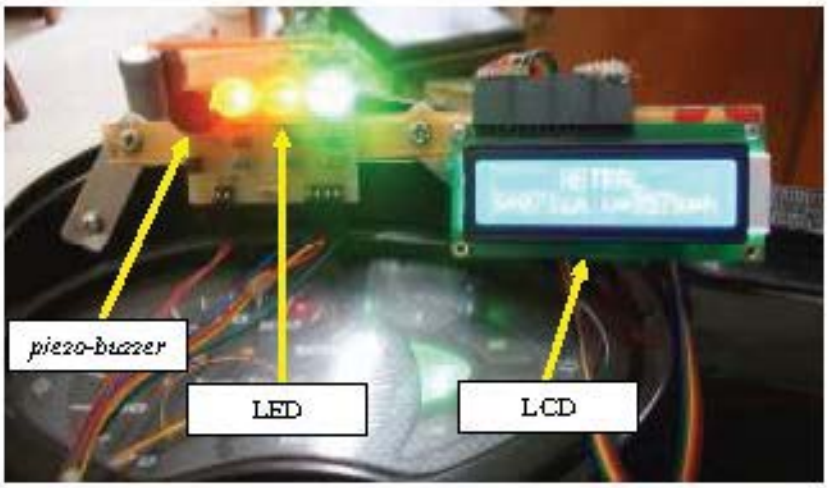
```
            }                                   //+ buzzer ON
        }

}
```

LAMPIRAN 3

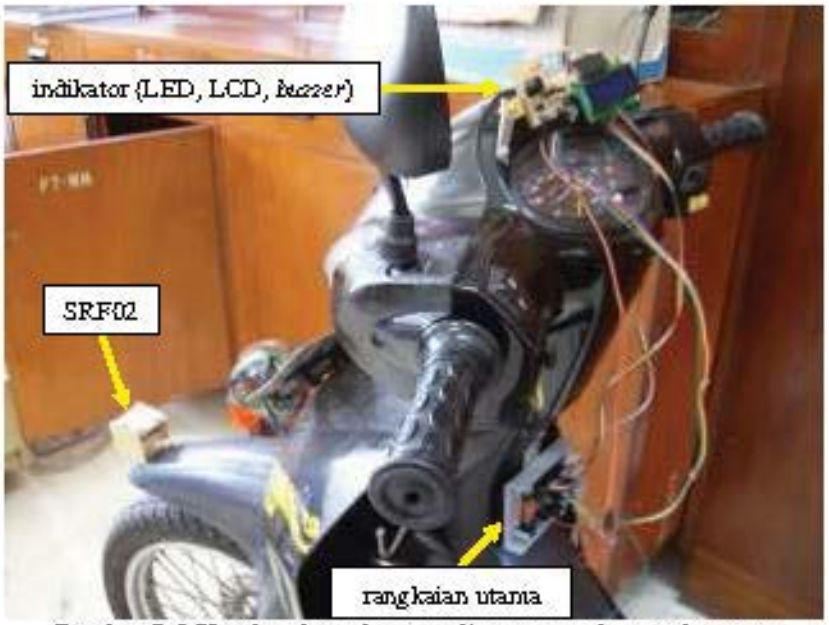
FOTO ALAT

Gambar L.5 Sensor SRF02 yang terletak di penutup ban depan

Gambar L.6 Rangkaian utama (tanpa penutup boks)



Gambar L.7 Indikator *piezo-buzzer*, LED, dan LCD



Gambar L.8 Keseluruhan alat saat dipasang pada sepeda motor

LAMPIRAN 4
PETUNJUK PENGGUNAAN
ALAT PEMANTAU JARAK AMAN BERKENDARAAN SEPEDA MOTOR

1. Pendahuluan

Alat ini berfungsi untuk mengetahui seberapa aman sepeda motor dan mengindikasikannya kepada pengemudi. Aman yang dimaksud adalah sepeda motor mampu terhindar dari tabrakan dengan kendaraan didepannya ketika kendaraan tersebut berhenti. Alat ini menghitung status aman tersebut dari 2 hal, yaitu: jarak kosong di depan dan kecepatan sepeda motor. Penyampaian status jarak aman kepada pengandara melalui 2 buah indikator, yaitu: LED (visual) dan *buzzer* (audio).

2. Pembacaan Indikator

Indikasi-indikasi yang ditunjukkan oleh LED dan *buzzer* memiliki arti yang berbeda-beda. Tabel L-4.1 akan menunjukan indikasi yang dapat muncul dan artinya.

3. Hal-hal yang Penting Untuk Diperhatikan

Berikut ini adalah beberapa yang harus diperhatikan tentang alat ini, supaya mengahsilkan kinerja yang maksimal, beberapa diantaranya dapat diindikasikan lewat indikator (lihat Tabel L-4.1):

- Alat ini terbatas oleh cuaca. Artinya tidak dapat digunakan ketika hujan deras, angin kencang, dan sebagainya.
- Jarak minimum dan maksimum yang mampu terdeteksi adalah 15 dan 600 cm (0,15 dan 6 meter).

- Kecepatan maksimum yang mampu dideteksi hingga 60 km/jam.
- Segala konektivitas pengkabelan dan perakitan telah didesain untuk digunakan pada sepeda motor Honda Karisma NF125D. Untuk penggunaan pada sepeda motor merk dan tipe lain diperlukan beberapa penyesuaian.

4. Spesifikasi

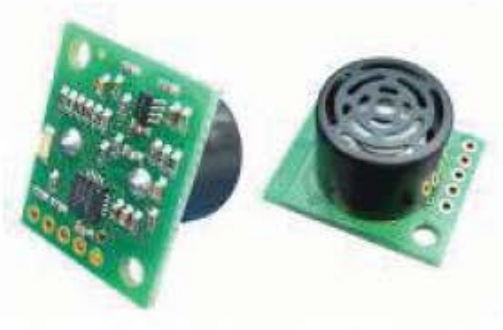
- Tegangan masukan: 10-14V
- Konsumsi arus: 11mA
- Konsumsi daya: 110-154mW

Tabel L-4.1 Indikasi yang muncul dan artinya bagi pengendara

Indikasi yang Muncul	Artinya
LED: hanya **hijau** menyala *Buzzer*: tidak berbunyi	Kondisi sepeda motor AMAN. Ketika kendaraan di depan berhenti mendadak, dapat terhindar dari tabrakan.
LED: hanya **kuning** menyala *Buzzer*: *beep* tiap 2x	Kondisi sepeda motor HATI-HATI. Ketika kendaraan di depan berhenti mendadak, mungkin terjadi tabrakan. Pengendara sebaiknya waspada dan mengurangi kecepatan.
LED: hanya **merah** menyala *Buzzer*: *beep* tanpa henti	Kondisi sepeda motor HATI-HATI. Ketika kendaraan di depan berhenti mendadak, sangat mungkin terjadi tabrakan. Pengendara harus waspada dan mengurangi kecepatan.
LED: **ketiganya** menyala berkedip *Buzzer*: *beep* 4x	Alat tidak dapat berfungsi dengan maksimal, karena jarak terukur di depan sepeda motor kurang dari 15 cm atau lebih dari 6 meter, ATAU kecepatan sepeda motor lebih dari 60 km/jam.
LED: **ketiganya** menyala diam *Buzzer*: tidak berbunyi	Sepeda motor dalam keadaan berhenti. Dapat pula berarti dalam posisi persneling netral (N). Kondisi sepeda motor AMAN.

SRF02 Ultrasonic range finder

Technical Specification



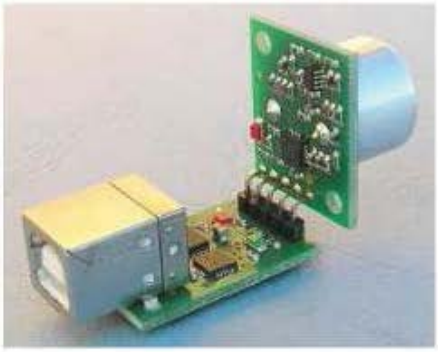
Overview

The SRF02 is a single transducer ultrasonic rangefinder in a small footprint PCB. It features both I2C and a Serial interfaces. The serial interface is a standard TTL level UART format at 9600 baud,1 start, 2 stop and no parity bits, and may be connected directly to the serial ports on any microcontroller. Up to 16 SRF02's may be connected together on a single bus, either I2C or Serial. New commands in the SRF02 include the ability to send an ultrasonic burst on its own without a reception cycle, and the ability to perform a reception cycle without the preceding burst. This has been as requested feature on our sonar's and the SRF02 is the first to see its implementation. Because the SRF02 uses a single transducer for both transmission and reception, the minimum range is higher than our other dual transducer rangers. The minimum measurement range is around 15cm (6 inches). Like all our rangefinders, the SRF02 can measure in uS, cm or inches.

Operating Modes

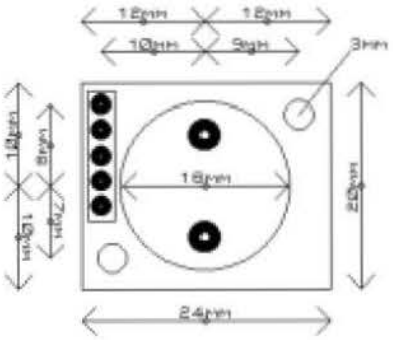
There are two operating modes for the SRF02. I2C mode and Serial Mode. This is set with the Mode pin, connected to 0v Ground for Serial Mode and left unconnected (or tied to +5v Vcc) for I2C Mode. These are documented on individual pages. For I2C Mode click here, and for Serial Mode click here.

SRF02 USB

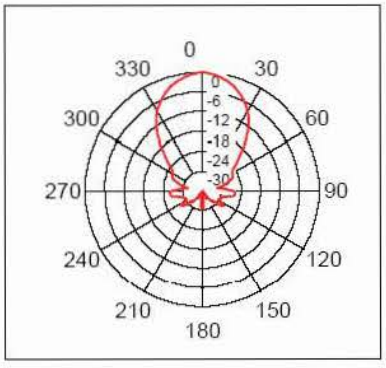


Connecting the SRF02 to your PC via USB is this easy. The USBI2C module supplies the SRF02 with power directly from the USB bus. The USB_I2C_SRF02 program can be downloaded here.

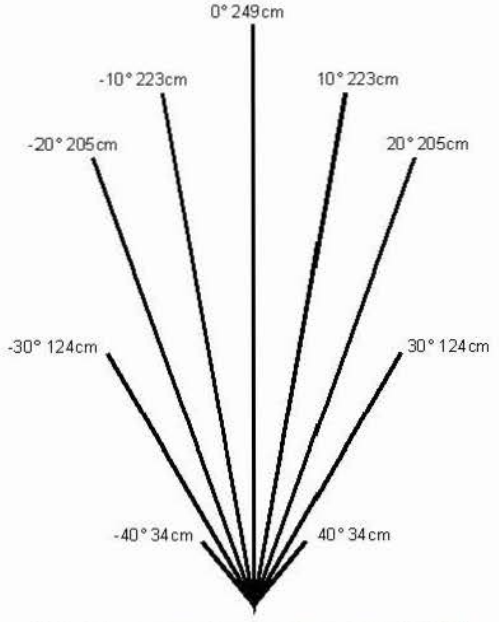
Dimensions



Beam Width



The manufacturers beam pattern, showing the sensitivity of the transducer in db.



This is the measured beam pattern for the SRF02, showing the maximum detection range of a 55mm diameter plastic pipe.

SRF02 Ultrasonic range finder

Technical Specification

I2C Mode

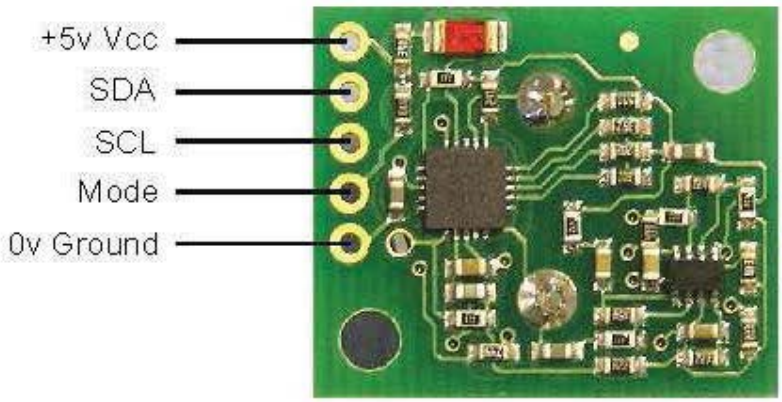
For Serial mode click here

I2C Communication

To use the SRF02 in I2C mode, make sure nothing is connected to the mode pin, it must be left unconnected.
The I2C bus is available on popular controllers such as the OOPic, Stamp BS2p, PicAxe etc. as well as a wide variety of micro-controllers. To the programmer the SRF02 behaves in the same way as the ubiquitous 24xx series EEPROM's, except that the I2C address is different. The default shipped address of the SRF02 is 0xE0. It can be changed by the user to any of 16 addresses E0, E2, E4, E6, E8, EA, EC, EE, F0, F2, F4, F6, F8, FA, FC or FE, therefore up to 16 sonar's can be used.

Connections

The connections to the SRF02 are identical to the SRF08 and SRF10 rangers. The "Mode" pin should be left unconnected, it has an internal pull-up resistor. The SCL and SDA lines should each have a pull-up resistor to +5v somewhere on the I2C bus. You only need one pair of resistors, not a pair for every module. They are normally located with the bus master rather than the slaves. The SRF02 is always a slave - never a bus master. If you need them, I recommend 1.8k resistors. Some modules such as the OOPic already have pull-up resistors and you do not need to add any more.



Registers

The SRF02 appears as a set of 6 registers.

Location	Read	Write
0	Software Revision	Command Register
1	Unused (reads 0x80)	N/A
2	Range High Byte	N/A
3	Range Low Byte	N/A
4	Autotune Minimum - High Byte	N/A
5	Autotune Minimum - Low Byte	N/A

Only location 0 can be written to. Location 0 is the command register and is used to start a ranging session. It cannot be read. Reading from location 0 returns the SRF02 software revision. The ranging lasts up to 65mS, and the SRF02 will not respond to commands on the I2C bus whilst it is ranging.

Locations, 2 and 3, are the 16bit unsigned result from the latest ranging - high byte first. The meaning of this value depends on the command used, and is either the range in inches, or the range in cm or the flight time in uS. A value of 0 indicates that no objects were detected. Do not initiate a ranging faster than every 65mS to give the previous burst time to fade away.

Locations, 4 and 5, are the 16bit unsigned minimum range. This is the approximate closest range the sonar can measure to. See the Autotune section below for full details.

Commands
The are three commands to initiate a ranging (80 to 82), to return the result in inches, centimeters or microseconds. Another set of three commands (86 to 88) do the same, but without transmitting the burst. These are used where the burst has been transmitted by another sonar. It is up to you to synchronize the commands to the two sonar's. There is a command (92) to transmit a burst without doing the ranging and also a set of commands to change the I2C address.

Command		Action
Decimal	Hex	
80	0x50	Real Ranging Mode - Result in inches
81	0x51	Real Ranging Mode - Result in centimeters
82	0x52	Real Ranging Mode - Result in micro-seconds
86	0x56	Fake Ranging Mode - Result in inches
87	0x57	Fake Ranging Mode - Result in centimeters
88	0x58	Fake Ranging Mode - Result in micro-seconds
92	0x5C	Transmit an 8 cycle 40khz burst - no ranging takes place
96	0x60	Force Autotune Restart - same as power-up. You can ignore this command.
160	0xA0	1st in sequence to change I2C address
165	0xA5	3rd in sequence to change I2C address
170	0xAA	2nd in sequence to change I2C address

Ranging
To initiate a ranging, write one of the above commands to the command register and wait the required amount of time for completion and read the result. The echo buffer is cleared at the start of each ranging. The ranging lasts up to 66mS, after this the range can be read from locations 2 and 3.

Checking for Completion of Ranging
You do not have to use a timer on your own controller to wait for ranging to finish. You can take advantage of the fact that the SRF02 will not respond to any I2C activity whilst ranging. Therefore, if you try to read from the SRF02 (we use the software revision number a location 0) then you will get 255 (0xFF) whilst ranging. This is because the I2C data line (SDA) is pulled high if nothing is driving it. As soon as the ranging is complete the SRF02 will again respond to the I2C bus, so just keep reading the register until its not 255 (0xFF) anymore. You can then read the sonar data. Your controller can take advantage of this to perform other tasks while the SRF02 is ranging. The SRF02 will always be ready 70mS after initiating the ranging.

LED
The red LED is used to flash out a code for the I2C address on power-up (see below). It also gives a brief flash during the "ping" whilst ranging.

Changing the I2C Bus Address

To change the I2C address of the SRF02 you must have only one sonar on the bus. Write the 3 sequence commands in the correct order followed by the address. Example; to change the address of a sonar currently at 0xE0 (the default shipped address) to 0xF2, write the following to address 0xE0; (0xA0, 0xAA, 0xA5, 0xF2). These commands must be sent in the correct sequence to change the I2C address, additionally, No other command may be issued in the middle of the sequence. The sequence must be sent to the command register at location 0, which means 4 separate write transactions on the I2C bus. When done, you should label the sonar with its address, however if you do forget, just power it up without sending any commands. The SRF02 will flash its address out on the LED. One long flash followed by a number of shorter flashes indicating its address. The flashing is terminated immediately on sending a command the SRF02.

Address		Long Flash	Short flashes
Decimal	Hex		
224	E0	1	0
226	E2	1	1
228	E4	1	2
230	E6	1	3
232	E8	1	4
234	EA	1	5
236	EC	1	6
238	EE	1	7
240	F0	1	8
242	F2	1	9
244	F4	1	10
246	F6	1	11
248	F8	1	12
250	FA	1	13
252	FC	1	14
254	FE	1	15

Take care not to set more than one sonar to the same address, there will be a bus collision and very unpredictable results.

Note - there is only one module address stored in the SRF02. If you change it, the equivalent Serial Mode address will also change:

```
0xE0, 0xE2, 0xE4, 0xE6, 0xE8, 0xEA, 0xEC, 0xEE, 0xF0, 0xF2, 0xF4, 0xF6,
0xF8, 0xFA, 0xFC, 0xFE I2C addresses
0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F Equivalent Serial addresses
```

AutoTune

The SRF02 does not require any user calibration. You power up and go right ahead and use the SRF02. Internally, there are tuning cycles happening automatically in the background. After the ultrasonic burst has been transmitted, the transducer keeps on ringing for a period of time. It is this ringing which limits the closest range the SRF02 can measure. This time period varies with temperature and from transducer to transducer, but is normally the equivalent of 11 to 16cm (4" to 6"), a bit more if the transducer is warm. The SRF02 is able to detect the transducer ring time and move its detection threshold right up to it, giving the SRF02 the very best performance possible. On power up, the detection threshold is set to 28cm (11"). The tuning algorithms quickly back this right up to the transducer ring. This happens within 5-6 ranging cycles - less than half a second at full scan speed. After this the tuning algorithms continue to monitor the transducer, backing the threshold up even further when possible or easing it out a bit when necessary. The tuning algorithms work automatically, in the background and with no impact on scan time.
The minimum range can be checked, if required by reading registers 4 and 5. This value is returned in uS, cm or inches, the same as the range. It is also possible to make the SRF02 re-tune by writing command 96 but you can ignore this command. It is used during our testing.

SRF02 Ultrasonic range finder

Technical Specification

Serial Mode

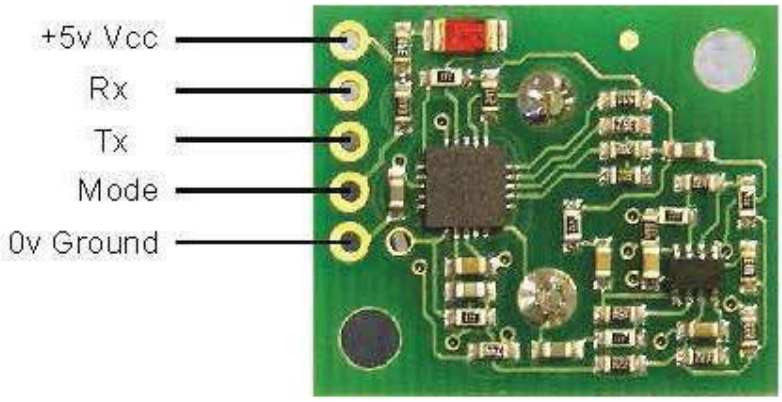
For I2C mode click here

Serial Communication

To use the SRF02 in Serial mode, make sure the Mode pin is connected to 0v Ground.
Serial data is fixed at 9600 baud 1 start, 2 stop and no parity bits. Serial data is a TTL level signal - It is NOT RS232. Do not connect the SRF02 to an RS232 port - you will destroy the module! If you would like to connect the SRF02 to your PC's RS232 port, you must use a MAX232 or similar device. It can also be used (in I2C mode) with the USBI2C module to make a self powered USB ranger, see the examples page for details. Many small controllers such as the OOPic, Stamp BS2p, PicAxe etc. as well as a wide variety of micro-controllers have serial ports. To communicate with the SRF02, you simply need to send two bytes, the address of the SRF02 (factory default is 0) and the command. The default shipped address can be changed by the user to any of 16 addresses 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, or 15, therefore up to 16 sonar's can be used.

Connections

The connections to the SRF02 are shown below. The "Mode" pin must be connected to 0v ground to place the SRF02 in serial mode. The Rx pin is data into the SRF02 and should be connected to the Tx pin on your controller. The Tx pin is data out of the SRF02 and should be connected to the Rx pin on your controller. If you're using multiple SRF02's, you can connect them all up to the same serial port on your controller. Connect the Tx from your controller to all the Rx pins on the SRF02's and connect the Rx pin on your controller to all the Tx pins on the SRF02's. This works because the Tx pins are high impedance (just a weak pull-up to 5v), except when actually sending data. Just make sure all the SRF02's are programmed to different addresses.



Commands

To send a command to the SRF02, you need to send two bytes. The first is the SRF02's address 0 to 15, (0x00 to 0x0F) and then the actual command itself - see below. The are three commands to initiate a ranging (80 to 82), to produce the result in inches, centimeters or microseconds. These three commands don't Tx the result back to your controller. You should wait 70mS and then use command 94 to get the result of the ranging. Another set of three commands (83 to 85) do the same, but also transmits the result of the ranging back to your controller as soon as it is available. Together, these six commands (80 - 85) are called "Real" because they perform a complete ranging. There is another set of six commands (86 - 91) called "Fake". They are the same as the "Real" commands except that they do not send the 8-cycle burst out. These are used where the burst has been transmitted by another sonar. It is up to you to synchronize the commands to the two sonar's. There is a command (92) to transmit a burst without doing the ranging.
Command 93 is used to get the firmware revision of the SRF02.
Command 94 gets returns two bytes (high byte first) from the most recent ranging. Put them together to make a 16-bit result.
Commands 95 and 96 are used by the Autotune algorithms - See the Autotune section below for details.

Command		Action
Decimal	Hex	
80	0x50	Real Ranging Mode - Result in inches
81	0x51	Real Ranging Mode - Result in centimeters
82	0x52	Real Ranging Mode - Result in micro-seconds
83	0x53	Real Ranging Mode - Result in inches, automatically Tx range back to controller as soon as ranging is complete.
84	0x54	Real Ranging Mode - Result in centimeters, automatically Tx range back to controller as soon as ranging is complete.
85	0x55	Real Ranging Mode - Result in micro-seconds, automatically Tx range back to controller as soon as ranging is complete.
86	0x56	Fake Ranging Mode - Result in inches
87	0x57	Fake Ranging Mode - Result in centimeters
88	0x58	Fake Ranging Mode - Result in micro-seconds
89	0x59	Fake Ranging Mode - Result in inches, automatically Tx range back to controller as soon as ranging is complete.
90	0x5A	Fake Ranging Mode - Result in centimeters, automatically Tx range back to controller as soon as ranging is complete.
91	0x5B	Fake Ranging Mode - Result in micro-seconds, automatically Tx range back to controller as soon as ranging is complete.
92	0x5C	Transmit an 8 cycle 40khz burst - no ranging takes place
93	0x5D	Get software version - sends a single byte back to the controller
94	0x5E	Get Range, returns two bytes (high byte first) from the most recent ranging.
95	0x5F	Get Minimum, returns two bytes (high byte first) of the closest range measurable - see Autotune section
96	0x60	Force Autotune Restart - same as power-up. You can ignore this command.
160	0xA0	1st in sequence to change I2C address
165	0xA5	3rd in sequence to change I2C address
170	0xAA	2nd in sequence to change I2C address

LED

The red LED is used to flash out a code for the I2C address on power-up (see below). It also gives a brief flash during the "ping" whilst ranging.

Changing the SRF02 Address

To change the address of the SRF02 you must have only one sonar connected. Write the 3 sequence commands in the correct order followed by the address. Example; to change the address of a sonar currently at 0 (the default shipped address) to 5, write the following to address 0; (0xA0, 0xAA, 0xA5, 0x05). These commands must be sent in the correct sequence to change the I2C address, additionally, No other command may be issued in the middle of the sequence. The sequence must be sent as four separate commands to the current address of the sonar. i.e. 0x00, 0xA0 then 0x00, 0xAA, then 0x00, 0xA5 and finally 0x00, 0x05. When done, you should label the sonar with its new address, however if you do forget, just power it up without sending any commands. The SRF02 will flash its address out on the LED. One long flash followed by a number of shorter flashes indicating its address. The flashing is terminated immediately on sending a command the SRF02.

Address		Long Flash	Short flashes
Decimal	Hex		
0	00	1	0
1	01	1	1
2	02	1	2

3	03	1	3
4	04	1	4
5	05	1	5
6	06	1	6
7	07	1	7
8	08	1	8
9	09	1	9
10	0A	1	10
11	0B	1	11
12	0C	1	12
13	0D	1	13
14	0E	1	14
15	0F	1	15

Take care not to set more than one sonar to the same address, there will be a bus collision and very unpredictable results.

Note - there is only one module address stored in the SRF02. If you change it, the equivalent I2C address will also change:

```
0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B,
0x0C, 0x0D, 0x0E, 0x0F Serial addresses
0xE0, 0xE2, 0xE4, 0xE6, 0xE8, 0xEA, 0xEC, 0xEE, 0xF0, 0xF2, 0xF4, 0xF6,
0xF8, 0xFA, 0xFC, 0xFE Equivalent I2C addresses
```

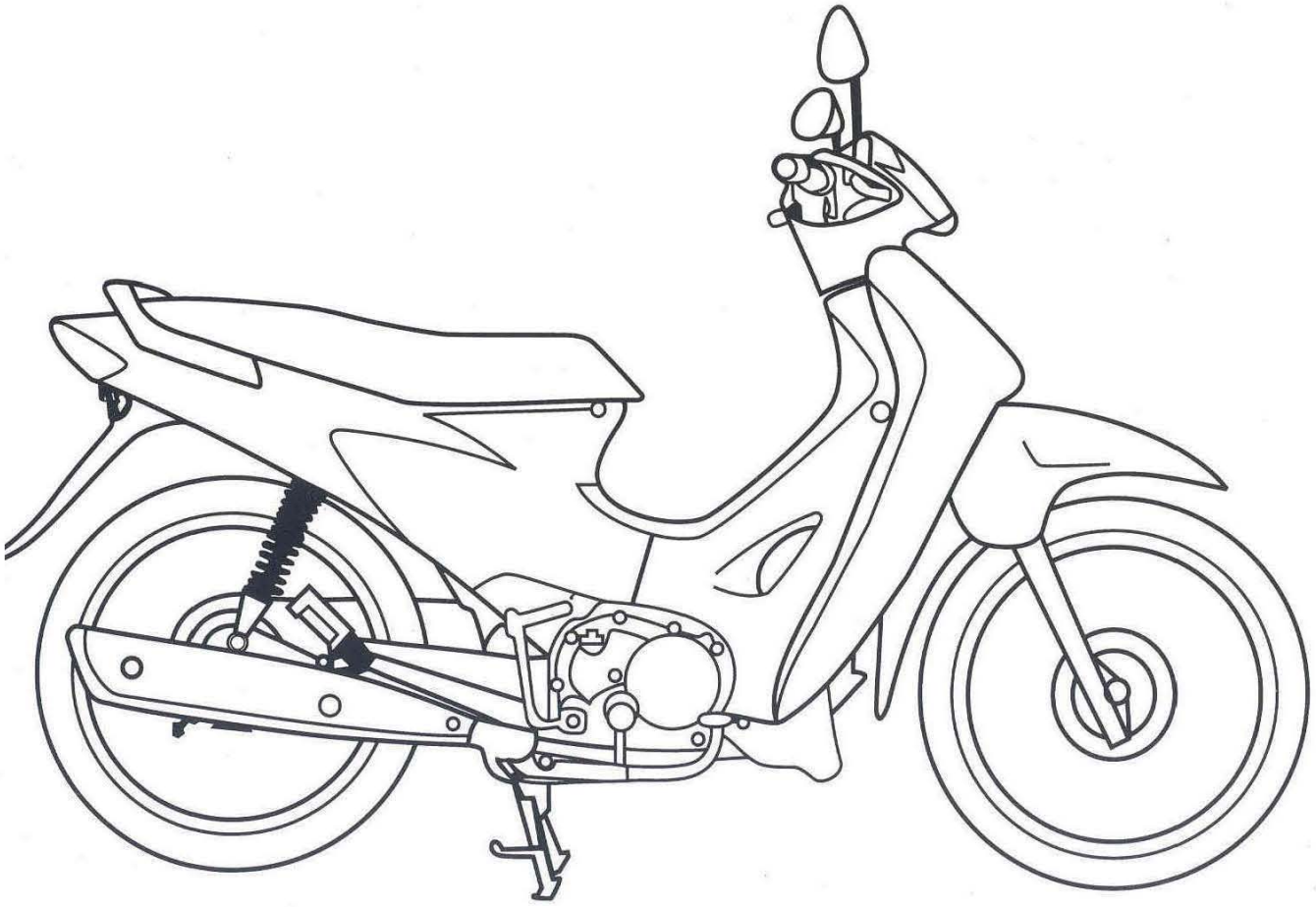
AutoTune

The SRF02 does not require any user calibration. You power up and go right ahead and use the SRF02. Internally, there are tuning cycles happening automatically in the background. After the ultrasonic burst has been transmitted, the transducer keeps on ringing for a period of time. It is this ringing which limits the closest range the SRF02 can measure. This time period varies with temperature and from transducer to transducer, but is normally the equivalent of 11 to 16cm (4" to 6"), a bit more if the transducer is warm. The SRF02 is able to detect the transducer ring time and move its detection threshold right up to it, giving the SRF02 the very best performance possible. On power up, the detection threshold is set to 28cm (11"). The tuning algorithms quickly back this right up to the transducer ring. This happens within 5-6 ranging cycles - less than half a second at full scan speed. After this the tuning algorithms continue to monitor the transducer, backing the threshold up even further when possible or easing it out a bit when necessary. The tuning algorithms work automatically, in the background and with no impact on scan time.

The minimum range can be checked, if required by sending command 95. This will return the closest measurable range in uS, cm or inches, the same as the range. It is also possible to make the SRF02 re-tune by writing command 96 but you can ignore this command. It is used during our testing.

HONDA

KARISMA

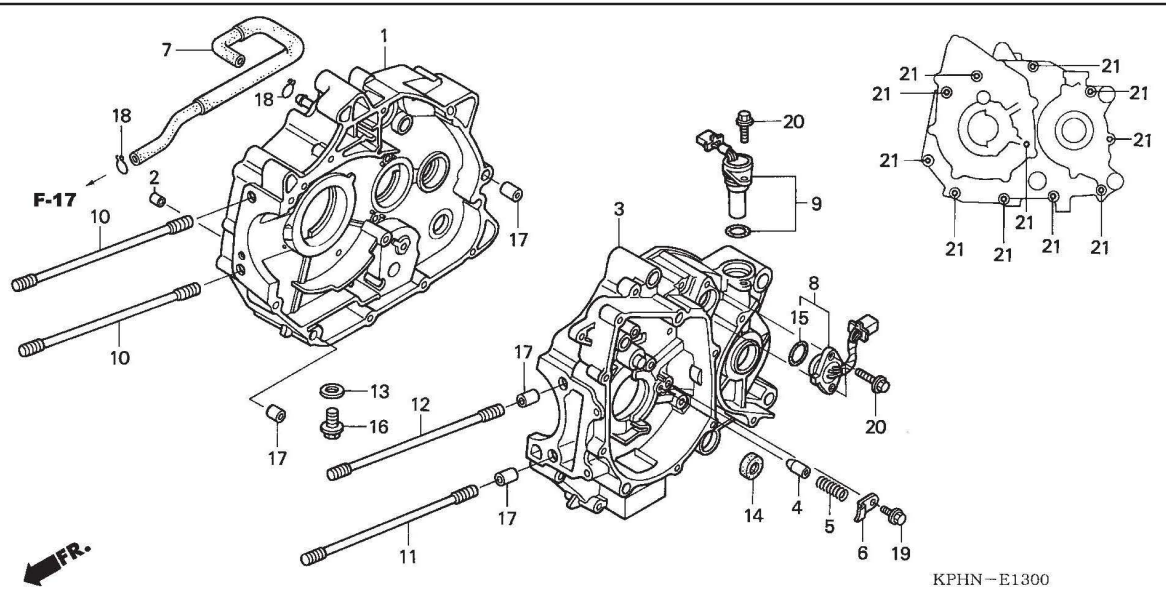


PARTS CATALOG

EDISI 1

Menu Utama

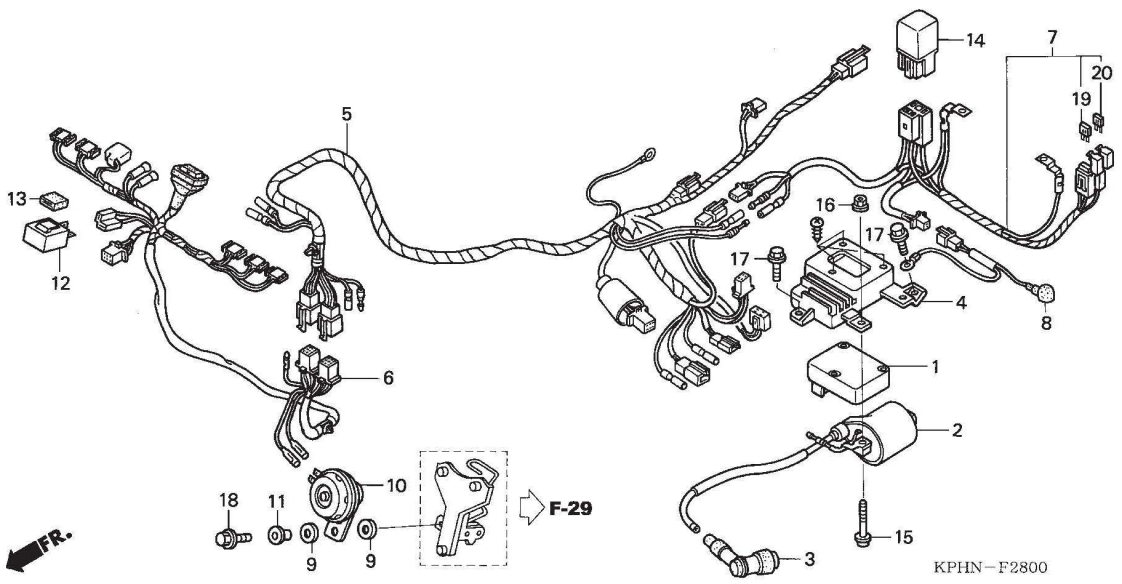
E-13 CRANKCASE



Service item	F.R.T.
SWITCH SET, CHANGE ·····	0.3
SENSOR ASSY., SPEED ·····	0.4
BOLT, CYLINDER STUD ·····	1.3
(ADD 0.1 FOR EACH.)	
(TAMBAHKAN 0.1 UNTUK MASING2)	
CRANKCASE COMP., R. ·····	*4.3
CRANKCASE COMP., L. ·····	*4.9

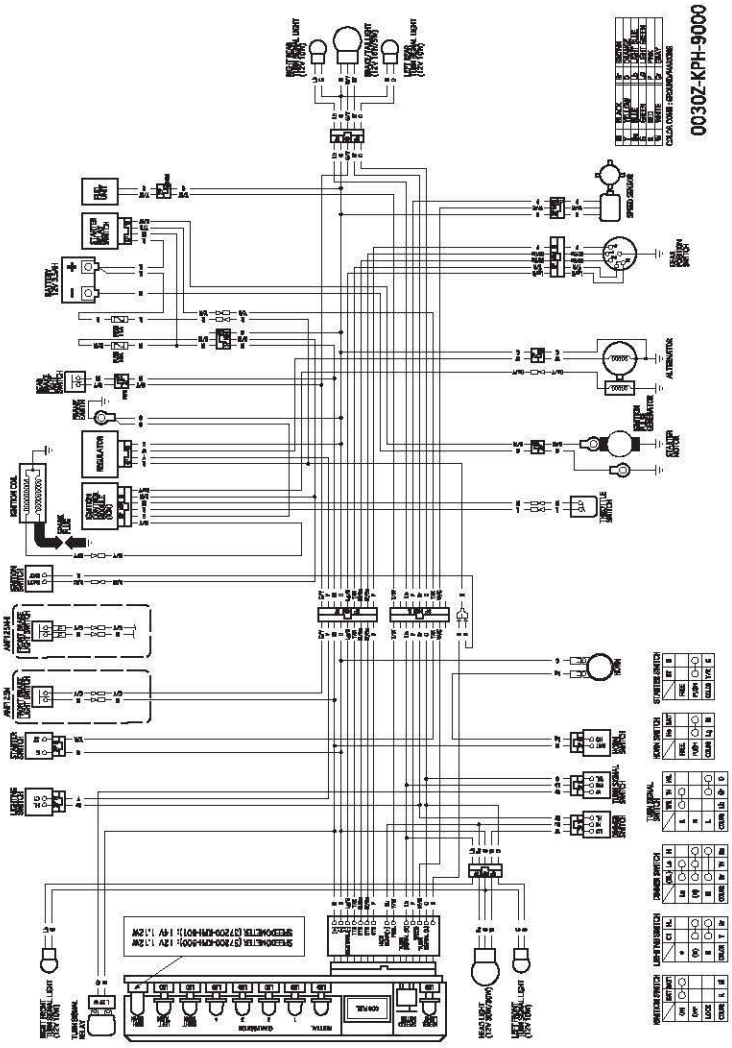
Ref. No.	Part No.	Description	Reqd. QTY NF 125/125D	Serial No.	Notes
1	11100-KPH-880	CRANKCASE COMP., R. ·····	1		
2	11133-KPH-900	JET, OIL, 0.8MM ·····	1		
3	11200-KPH-900	CRANKCASE COMP., L. ·····	1		
4	11215-KPH-900	PLUG, BEARING PUSH ·····	1		
5	11216-KPH-900	SPRING, BEARING PUSH ·····	1		
6	11217-KPH-900	PLATE, BEARING ·····	1		
7	15761-KPH-900	TUBE, BREATHER ·····	1		
8	35750-KPH-900	SWITCH SET, CHANGE ·····	1		
9	37700-KPH-901	SENSOR ASSY., SPEED ·····	1		
10	90031-KPH-900	BOLT, CYLINDER STUD ·····	2		
11	90032-KPH-900	BOLT, CYLINDER STUD ·····	1		
12	90033-KPH-900	BOLT, CYLINDER STUD ·····	1		
13	90407-259-000	PACKING, DRAIN COCK, 12.5X20	1		
14	91208-KPH-901	OIL SEAL, 11.6X24X10 ·····	1		
15	91305-KPH-900	O-RING, 21X2.3 ·····	1		
16	92800-12000	BOLT, DRAIN PLUG, 12MM ·····	1		
17	94301-10120	DOWEL PIN, 10X12 ·····	4		
18	95002-70000	CLIP, TUBE (C11) ·····	2		
19	95701-06014-00	BOLT, FLANGE, 6X14 ·····	1		
20	95701-06016-00	BOLT, FLANGE, 6X16 ·····	3		
21	96001-06060-00	BOLT, FLANGE, 6X60 ·····	11		

F-28 WIRE HARNESS



Service item	F.R.T.
HORN COMP. ·····	0.2
RELAY COMP., STARTER ·····	0.3
.RELAY COMP., WINKER	
SUB HARNESS, SPEEDOMETER ·····	0.4
C.D.I. UNIT ·····	0.5
.SUB HARNESS, BATTERY	
COIL ASSY., IGNITION ·····	0.6
.RECTIFIER COMP., REGULATOR	
HARNESS, WIRE ·····	1.4

Ref. No.	Part No.	Description	Reqd. QTY NF 125/125D	Serial No.	Notes
1	30410-KPH-881	C.D.I. UNIT ·····	1		
2	30500-KPH-900	COIL ASSY., IGNITION ·····	1		
3	30700-KPH-881	CAP ASSY., NOISE SUPPRESSOR ·	1		
4	31600-KPH-881	RECTIFIER COMP., REGULATOR ··	1		
5	32100-KPH-880	HARNESS, WIRE ·····	1		
6	32101-KPH-880	SUB HARNESS, SPEEDOMETER ····	1		
	32101-KPH-890	·····	1		
7	32103-KPH-880	SUB HARNESS, BATTERY ·····	1		
8	32410-KPH-900	CABLE, STARTING MOTOR ·····	1		
9	33714-KL3-620	RUBBER, TAILLIGHT BRACKET ···	2		
10	38110-KPH-881	HORN COMP. (HIGH) ·····	1		
11	38117-KPH-880	COLLAR, HORN SETTING ·····	1		
12	38301-KPH-881	RELAY COMP., WINKER ·····	1		
13	38306-KK4-000	SUSPENSION, WINKER RELAY (MITSUBA)	1		
14	38501-KPH-901	RELAY COMP., STARTER ·····	1		
15	90112-GK8-010	BOLT, FLANGE, 5X22 ·····	2		
16	94050-05000	NUT, FLANGE, 5MM ·····	2		
17	95701-06016-00	BOLT, FLANGE, 6X16 ·····	2		
18	95701-06020-00	BOLT, FLANGE, 6X20 ·····	1		
19	98200-31000	FUSE, BLADE (10A) ·····	1		
20	98200-31500	FUSE, BLADE (15A) ·····	1		


0030Z-KPH-9000

BIODATA

Nama	: Oswin Arinaga Soerjawinata
NRP	: 5103006002
Tempat Lahir	: Surabaya
Tanggal Lahir	: 3 Juli 1988
Agama	: Kristen
Alamat	: Jl. Baruk Utara VIII / 16 Perumahan Pondok Nirwana Surabaya

Riwayat Pendidikan:

- Tahun 2000, lulus SD Kr. Petra 13, Surabaya
- Tahun 2003, lulus SMP Kr. Petra 3, Surabaya
- Tahun 2006, lulus SMA Kr. Petra 2, Surabaya
- Tahun 2006 hingga biodata ini ditulis, tercatat sebagai mahasiswa Fakultas Teknik Jurusan Teknik Elektro di Universitas Katolik Widya Mandala Surabaya

* Kontestan Robot Cerdas Indonesia (KRCI) Regional IV 2009