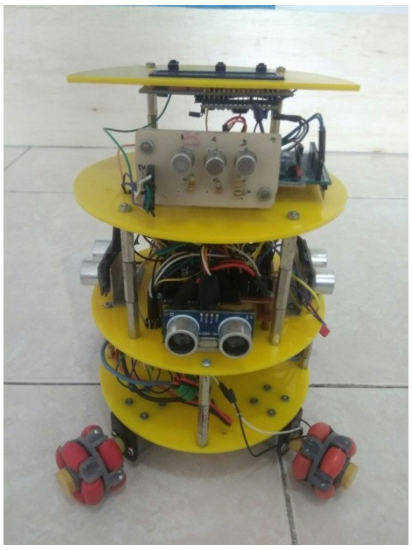
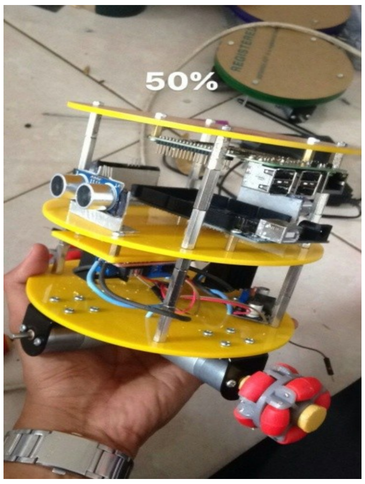
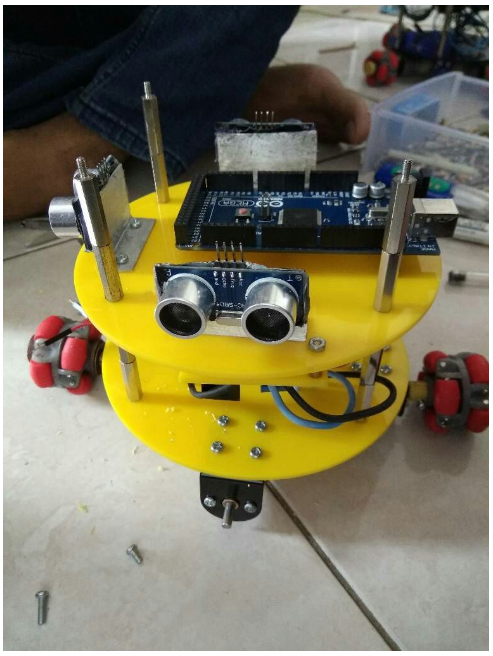
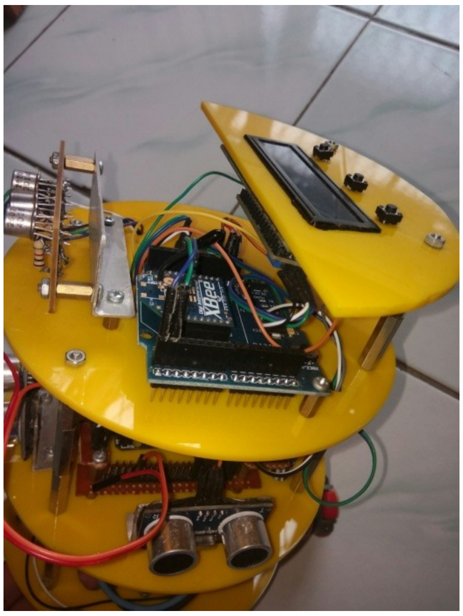
LAMPIRAN

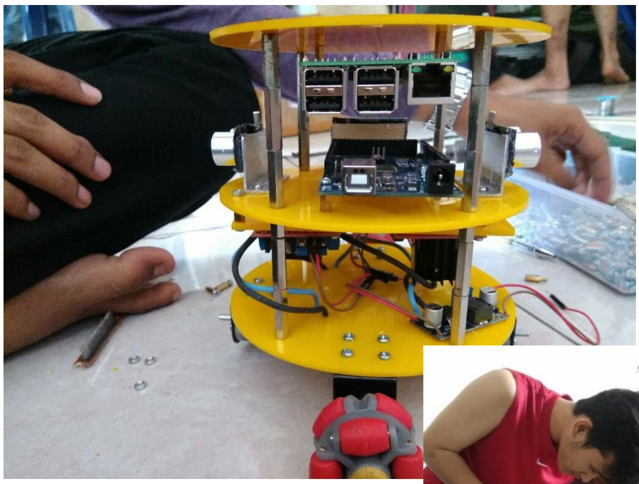
Gambar Alat

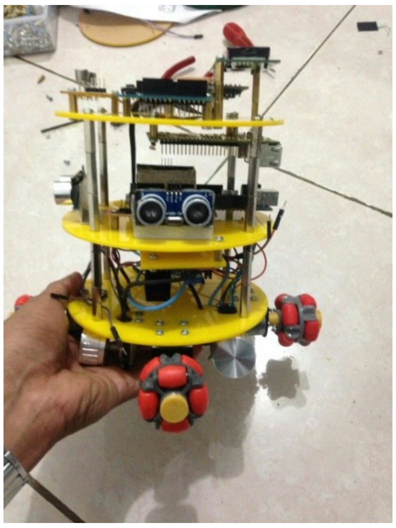












Koding Keseluruhan *Mobile Robot*

```
#define btn_Ok
50       //Push Button Ok
pada robot
#define btn_Up
52       //Push Button Up
pada robot
#define btn_Down
48     //Push Button
Down pada robot

#define tgs2600 A0
#define tgs2602 A1
#define tgs2620 A2

#define xbee Serial3
#define raspi Serial2
#define pinRSSI 53

#define speedb 40

#include <EEPROM.h>
#include <Wire.h>
#include <LCD.h>
#include
<LiquidCrystal_I2C.h>  //
Library LCD I2C
//#include "MPU6050.h"
//#include "I2Cdev.h"

#include
<HMC5883L.h>           //
Library Compass


#define I2C_ADDR
0x3F        //LCD (jika
tidak bisa 3F ganti 27F)
#define BACKLIGHT_PIN
3
#define En_pin  2
#define Rw_pin  1
#define Rs_pin  0
#define D4_pin  4
#define D5_pin  5
#define D6_pin  6
#define D7_pin  7

LiquidCrystal_I2C
lcd(I2C_ADDR, En_pin,
Rw_pin, Rs_pin, D4_pin,
D5_pin, D6_pin, D7_pin);


#define trig_depan 25
#define echo_depan 24
#define trig_kanan 23
#define echo_kanan 22
#define trig_kiri 27
#define echo_kiri 26

#define in1_krB 30
#define in2_krB 31
#define in3_knD 29
#define in4_knD 28
#define ena_krB 2
#define enb_knD 3

#define in1_knB 32
#define in2_knB 33
#define in3_krD 35
#define in4_krD 34
#define ena_knB 4
#define enb_krD 5


int jarak_robot = 0;
int
jarak_depan;
//variabel bilangan bulat
int jarak_kanan;
int jarak_kiri;
int arah  = 0;

int best_robot = 0 ;
long best_x = 0 ;
long best_y = 0 ;
int best_2600 = 0 ;
int best_2602 = 0 ;
int best_2620 = 0 ;
int best_kiri = 0 ;
int best_depan = 0 ;
int best_kanan = 0 ;
int best_kompas = 0 ;


long millis_mulai;
String prediksi="";
#define waktu_kirim 500


long tempuh_x ,
tempuh_y;


//0 depan
//1 kiri
//2 kanan
//3 mundur
HMC5883L compass;


float Kp, Ki, Kd, Speed,
Sp;  //variabel berkoma
float error, jumlah_error,
selisih_error,
error_sebelumnya;
float P, I, D, PID;
byte menu = 1;
float heading;
float headingDegrees;
long waktu_mulai;


#define waktu_batas
5000

#define jarak_aman 10

String raspiString = "";
// a string to hold
incoming data
boolean raspiComplete
= false;  // whether the
string is complete


String xbeeString = "";
// a string to hold
incoming data
boolean xbeeComplete
= false;  // whether the
string is complete

int t2600 = 0;
int t2602 = 0;
int t2620 = 0;


void setup() {
  // put your setup code
here, to run once:
  Serial.begin(9600);
  xbee.begin(9600);
  raspi.begin(9600);
  lcd.begin (16, 2);


lcd.setBacklightPin(BACK
LIGHT_PIN,
POSITIVE);   //LCD
  lcd.setBacklight(HIGH);
  lcd.home ();

  pinMode(btn_Ok,
INPUT_PULLUP);
//Push Button
  pinMode(btn_Up,
INPUT_PULLUP);
  pinMode(btn_Down,
INPUT_PULLUP);
```

```
  pinMode(trig_depan,
OUTPUT);
  pinMode(echo_depan,
INPUT);
  pinMode(trig_kanan,
OUTPUT);
  pinMode(echo_kanan,
INPUT);
  pinMode(trig_kiri,
OUTPUT);
  pinMode(echo_kiri,
INPUT);

  pinMode(tgs2600,
INPUT);
  pinMode(tgs2602,
INPUT);
  pinMode(tgs2620,
INPUT);



  pinMode(in1_knB,
OUTPUT);
  pinMode(in2_knB,
OUTPUT);
  pinMode(in3_krD,
OUTPUT);
  pinMode(in4_krD,
OUTPUT);
  pinMode(ena_knB,
OUTPUT);
  pinMode(enb_krD,
OUTPUT);



  pinMode(in1_krB,
OUTPUT);
  pinMode(in2_krB,
OUTPUT);
  pinMode(in3_knD,
OUTPUT);
  pinMode(in4_knD,
OUTPUT);
  pinMode(ena_krB,
OUTPUT);
  pinMode(enb_knD,
OUTPUT);

  //buka komen ini jika
ingin set semua variabel
menjadi 0
  //  Kp = 20;
  //  Kd = 0;
  //  Ki = 0;
  //  Sp = 30;
  //  Speed = 0;
  //  simpan_semua();
  //
  //
  // while(1){
  //  baca_depan();
  //  if(jarak_depan<20){
  //      motor_knB(-150);
  //      motor_knD(-150);
  //      motor_krB(-150);
  //      motor_krD(-150);
  //   }else{
  //      motor_knB(150);
  //      motor_knD(150);
  //      motor_krB(150);
  //      motor_krD(150);
  //
  //   }
  //
  //}


  set_variabel();
  // Initialize Initialize
HMC5883L
  Serial.println("Initialize
HMC5883L");
//Kompas
  while (!
compass.begin())
  {
    Serial.println("Could
not find a valid
HMC5883L sensor, check
wiring!");
    delay(500);
  }

  // Set measurement
range

compass.setRange(HMC
5883L_RANGE_1_3GA);

  // Set measurement
mode

compass.setMeasureme
ntMode(HMC5883L_CON
TINOUS);

  // Set data rate

compass.setDataRate(H
MC5883L_DATARATE_30
HZ);

  // Set number of
samples averaged

compass.setSamples(HM
C5883L_SAMPLES_8);

  // Set calibration offset.
See
HMC5883L_calibration.in
o
  compass.setOffset(50,
-130);



//while (1) {
//motor_knD(50);
//  delay(2000);
//  motor_knD(-100);
//  delay(1000);
//
//  }

}

void loop() {
  tampil_menu();
  // put your main code
here, to run repeatedly:
  while (1) {
//Baca depan sensor
ultrasonik
    if (digitalRead(btn_Ok)
== 0) {
      delay(1000);
      if (menu ==
2)start_robot();
      else if (menu ==
3)setKp();
      else if (menu ==
4)setKd();
      else if (menu ==
5)setKi();
      else if (menu ==
6)setSp();
      else if (menu ==
7)setSpd();
      else if (menu == 8)
tampil_sensor();
      else if (menu == 9)
tampil_kompas();
      else if (menu == 10)
test_raspi();


    }

    if (digitalRead(btn_Up)
== 0) {
      delay(200);
      menu = menu + 1;
      if (menu > 10)menu
= 1;
      tampil_menu();
    }
    if
(digitalRead(btn_Down)
== 0) {
      delay(200);
      menu = menu - 1;
      if (menu < 1)menu =
10;
      tampil_menu();
    }
  }
  //
  //  baca_depan();
  //  delay(500);
  //  baca_kanan();
  //  delay(500);
  //  baca_kiri();
  //  delay(500);
}

long
microsecondsToCentimet
ers (long microseconds)
{
  return microseconds /
29 / 2;
}


void tampil_sensor() {
//tampil sensor
ultrasonik
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Baca Sensor
");


  while
(digitalRead(btn_Ok) ==
1) {
    baca_kanan ();
    baca_depan ();
    baca_kiri ();

      t2600 =
analogRead(tgs2600);
    t2602 =
analogRead(tgs2602);
    t2620 =
analogRead(tgs2620);
    lcd.setCursor(0,0);

lcd.print(t2600);lcd.print
("  ");

lcd.print(t2602);lcd.print
("  ");

lcd.print(t2620);lcd.print
("  ");

    lcd.setCursor(0, 1);
    lcd.print(jarak_kiri);
lcd.print("  ");
    lcd.setCursor(5, 1);
    lcd.print(jarak_depan);
lcd.print("  ");
    lcd.setCursor(10, 1);
    lcd.print(jarak_kanan);
lcd.print("  ");
    delay(200);
  }

  delay(1000);

  tampil_menu();
}


void baca_kompas()
//baca sensor kompas
{
  Vector norm =
compass.readNormalize(
);

  // Calculate heading
  heading =
atan2(norm.YAxis,
norm.XAxis);

  // Set declination angle
on your location and fix
heading
  // You can find your
declination on:
http://magnetic-
declination.com/
  // (+) Positive or (-) for
negative
  // For Bytom / Poland
declination angle is
4'26E (positive)
  // Formula: (deg +
(min / 60.0)) / (180 /
M_PI);
  float declinationAngle
= (4.0 + (26.0 / 60.0)) /
(180 / M_PI);
  heading +=
declinationAngle;

  // Correct for heading <
0deg and heading >
360deg
  if (heading < 0)
  {
    heading += 2 * PI;
  }

  if (heading > 2 * PI)
  {
    heading -= 2 * PI;
  }

  // Convert to degrees
  headingDegrees =
heading * 180 / M_PI;

  // Output
  //  Serial.print("
Heading = ");
  //  Serial.print(heading);
  //  Serial.print(" Degress
= ");
  //
Serial.print(headingDegr
ees);
  //  Serial.println();

  //  delay(100);
}

void tampil_kompas() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Baca Kompas
");


  while
(digitalRead(btn_Ok) ==
1) {
    baca_kompas ();
    lcd.setCursor(0, 1);
    lcd.print(heading);
lcd.print("    ");
    lcd.setCursor(5, 1);

lcd.print(headingDegree
s); lcd.print("    ");
  }

  delay(1000);



  tampil_menu();
}


void tampil_menu () {
  if (menu == 1) {
    //Serial.println ("C-
Fire");
```

```
    lcd.setCursor(0, 0);
    lcd.print("C-Balancer
");
    delay(200);
    //lcd.print("==
Bismillah ==");
    //lcd.setCursor(0,1);
    delay(20);

  }

  if (menu == 2) {
    //Serial.println ("Start
Kiri");
    lcd.setCursor(0, 0);
    lcd.print("Start Robot
");
    delay(200);

  }


  if (menu == 3) {

    //Serial..println ("KP");
    lcd.setCursor (0, 0);
    lcd.print("Setting KP
");
    delay(200);
  }


  if (menu == 4) {

    //Serial..println ("KD");
    lcd.setCursor (0, 0);
    lcd.print("Setting KD
");
    delay(200);
  }


  if (menu == 5) {

    //Serial..println ("KI");
    lcd.setCursor (0, 0);
    lcd.print("Setting KI
");
    delay(200);
  }


  if (menu == 6) {

    //Serial.println ("Set
Point");
    lcd.setCursor (0, 0);
    lcd.print("Set Point
");
    delay(200);
  }


  if (menu == 7) {

    //Serial..println
("Speed");
    lcd.setCursor (0, 0);
    lcd.print("Speed
");
    delay(200);
  }


  if (menu == 8) {

    //Serial..println ("Baca
Sensor");
    lcd.setCursor (0, 0);
    lcd.print("Baca Sensor
");
    delay(200);
  }

  if (menu == 9) {

    //Serial..println ("Baca
Kompas");
    lcd.setCursor (0, 0);
    lcd.print("Baca
Kompas    ");
    delay(200);

  }

  if (menu == 10) {

    //Serial..println ("Baca
Kompas");
    lcd.setCursor (0, 0);
    lcd.print("Test SVM
");
    delay(200);

  }

  if (menu == 11) {

    //Serial..println ("Baca
Kompas");
    lcd.setCursor (0, 0);
    lcd.print("Baca Gas
");
    delay(200);

  }
}

void baca_gas(){
  delay(1000);
  while
(digitalRead(btn_Ok) ==
1) {
      t2600 =
analogRead(tgs2600);
    t2602 =
analogRead(tgs2602);
    t2620 =
analogRead(tgs2620);
    lcd.setCursor(0,1);

lcd.print(t2600);lcd.print
("  ");

lcd.print(t2602);lcd.print
("  ");

lcd.print(t2620);lcd.print
("  ");

  }
  tampil_menu();
  delay(1000);

}

void start_robot() {
  error = 0;
  jumlah_error = 0;
  selisih_error = 0;
  error_sebelumnya = 0;
  tempuh_x = 0 ;
  tempuh_y = 0;
  set_variabel();
  lcd.setCursor(0, 0);
  lcd.print("Bismillah
");
  delay(2000);
  best_robot = 0;
  bool mutar = true;
  int temp_2600 = 0;
  int temp_2602 = 0;
  int temp_2620 = 0;
  float temp_heading = 0
;
  long waktu_jalan =
millis();
  while
(digitalRead(btn_Ok) ==
1) {
    jarak_robot =
pulseIn(48, LOW, 500);

Serial.println(jarak_robot
);
```

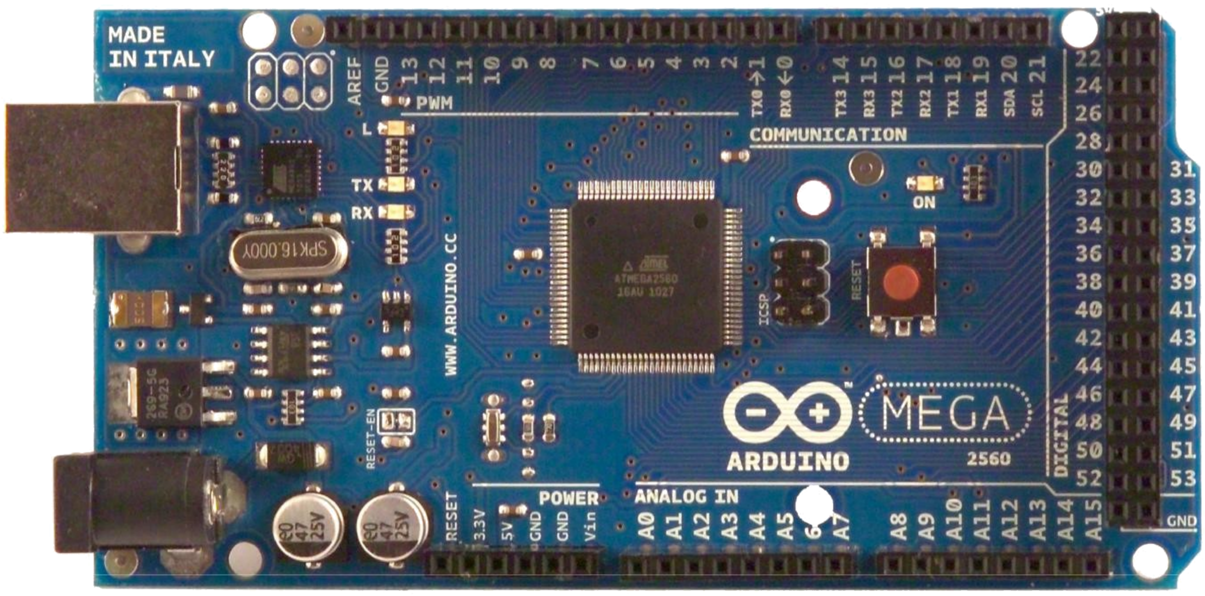
```
    //xbee.print("rssi:");
xbee.println(jarak_robot)
;
    //delay(2000);
    baca_depan();
    baca_kiri();
    baca_kanan();
    t2600 =
analogRead(tgs2600);
    t2602 =
analogRead(tgs2602);
    t2620 =
analogRead(tgs2620);
    baca_kompas();
    Serial.print("Set Point
="); Serial.println(Sp    );

    if (t2600 > 400 ||
t2602 > 400 || t2620 >
400){
     prediksi =
uji_sample(t2600,t2602,t
2620);
    }

    if (mutar) {
      motor_krB(speedb);
      motor_krD(speedb);
      motor_knB(-speedb);
      motor_knD(-speedb);
      if (t2600 >
temp_2600 && t2602 >
temp_2602 && t2620 >
temp_2620 ) {
        temp_heading =
headingDegrees;
        temp_2600 =
t2600;
        temp_2602 =
t2602;
        temp_2620 =
t2620;

      }
      float
sudut_stop_putar = Sp -
10 ;
      lcd.setCursor(0,0);

lcd.print(t2600);lcd.print
("  ");

lcd.print(t2602);lcd.print
("  ");

lcd.print(t2620);lcd.print
("  ");
      lcd.setCursor(0,1);

lcd.print(temp_2600);lcd
.print("  ");

lcd.print(temp_2602);lcd
.print("  ");

lcd.print(temp_2620);lcd
.print("  ");

      if (sudut_stop_putar
< 0 )sudut_stop_putar
+= 360;
      if
(abs(headingDegrees -
sudut_stop_putar) < 5 ||
millis() - waktu_jalan >
10000) {
        mutar = false;
        waktu_jalan =
millis();
         Sp =
temp_heading;
      }
    } else {
      if (millis() -
waktu_jalan > 20000){
        mutar = true;
        temp_2600 = 0;
        temp_2602 = 0;
        temp_2620 = 0;
        temp_heading = 0 ;
        lcd.clear();
        waktu_jalan =
millis();
      }
      if (jarak_depan <
jarak_aman) {
        lcd.setCursor(0,0);
        lcd.print("depan
keno");
            motor_krB(-
speedb);
            motor_krD(-
speedb);
            motor_knB(-
speedb);
            motor_knD(-
speedb);
            delay(1000);
        if (jarak_kiri <
jarak_kanan) {
          //arah  = 2;
          arah = 0;
          Sp = Sp + 45 +
random(45);
          if (Sp > 360)Sp =
Sp - 360;
          while (abs(Sp -
headingDegrees) > 10) {
            lcd.setCursor(0,
0);
            lcd.print(Sp);
lcd.print("   ");
            lcd.setCursor(0,
1);

lcd.print(headingDegree
s); lcd.print("   ");

motor_krB(speedb);

motor_krD(speedb);
            motor_knB(-
speedb);
            motor_knD(-
speedb);
            baca_kompas();

          }
          //delay(2000);

        } else {
          //arah  = 1;
          arah = 0;
          Sp = Sp - 45 -
random(45);
          if (Sp < 0)Sp =
360 + Sp;
          while (abs(Sp -
headingDegrees) > 10) {
            lcd.setCursor(0,
0);
            lcd.print(Sp);
lcd.print("   ");
            lcd.setCursor(0,
1);

lcd.print(headingDegree
s); lcd.print("   ");
            motor_krB(-
speedb);
            motor_krD(-
speedb);

motor_knB(speedb);

motor_knD(speedb);
            baca_kompas();

          }
        }

      }
      if (jarak_kiri <
jarak_aman) {
        //        if
(jarak_depan >
jarak_aman) {
        //          arah  = 0;
        //        } else {

lcd.setCursor(0,0);
        lcd.print("kiri
keno");
        motor_krB(-
speedb);

motor_krD(speedb);

motor_knB(speedb);
            motor_knD(-
speedb);
            delay(1000);
        Sp = Sp + 45 +
random(45);
        if (Sp > 360)Sp =
Sp - 360;
        while (abs(Sp -
headingDegrees) > 10) {
          lcd.setCursor(0,
0);
          lcd.print(Sp);
lcd.print("   ");
          lcd.setCursor(0,
1);

lcd.print(headingDegree
s); lcd.print("   ");

motor_krB(speedb);

motor_krD(speedb);
          motor_knB(-
speedb);
          motor_knD(-
speedb);
          baca_kompas();

        }
        //        }
      }
      if (jarak_kanan <
jarak_aman) {
        lcd.setCursor(0,0);
        lcd.print("kanan
keno");
        //        if
(jarak_depan >
jarak_aman) {
        //          arah  = 0;
        //        } else {

motor_krB(speedb);
            motor_krD(-
speedb);
            motor_knB(-
speedb);

motor_knD(speedb);
            delay(1000);
        arah = 0;
        Sp = Sp - 45 -
random(45);
        if (Sp < 0)Sp = 360
+ Sp;
        while (abs(Sp -
headingDegrees) > 10) {
          lcd.setCursor(0,
0);
          lcd.print(Sp);
lcd.print("   ");
          lcd.setCursor(0,
1);

lcd.print(headingDegree
s); lcd.print("   ");
          motor_krB(-
speedb);
          motor_krD(-
speedb);

motor_knB(speedb);

motor_knD(speedb);
          baca_kompas();
        }
        //        }
      }



      //Sp =
load_eeprom(12);
      pergerakan_robot();
    }




    if (millis() -
millis_mulai >
waktu_kirim) {
      xbee.println();
      millis_mulai =
millis();
    }

    xbeeEvent();
    // print the string
when a newline arrives:
    if (xbeeComplete) {

Serial.println(xbeeString)
;
      if
(xbeeString.length() <=
10 ) {
        if
(xbeeString.charAt(0)
== '1') { //robot 1
          //
          // format robot
no_robot,compass,x,y,tg
s2600,tgs2602,tgs2620,j
arak_kiri,jarak_depan,jar
ak_kanan;

Serial.print("Hadir");
          xbee.print("1,");

xbee.print(headingDegre
es);
          xbee.print(",");

xbee.print(tempuh_x);
          xbee.print(",");

xbee.print(tempuh_y);
          xbee.print(",");
          xbee.print(t2600);
          xbee.print(",");
          xbee.print(t2602);
          xbee.print(",");
          xbee.print(t2620);
          xbee.print(",");

xbee.print(jarak_kiri);
          xbee.print(",");

xbee.print(jarak_depan);
          xbee.print(",");

xbee.println(jarak_kanan
);
          xbee.print(",");

xbee.println(prediksi);
          }
      } else {

Serial.println("masuk sini
nah");

Serial.println(getJumlah(
xbeeString, ',', 1));
        if
(getJumlah(xbeeString,
',', 1) >= 4) {
        // format robot
no_robot,compass,x,y,tg
s2600,tgs2602,tgs2620,j
arak_kiri,jarak_depan,jar
ak_kanan;
        best_robot =
getValue(xbeeString, ',',
0).toInt();
        best_kompas =
getValue(xbeeString, ',',
1).toInt();
        best_x =
getValue(xbeeString, ',',
2).toInt();
        best_y =
getValue(xbeeString, ',',
3).toInt();
        best_2600 =
getValue(xbeeString, ',',
4).toInt();
        best_2602 =
getValue(xbeeString, ',',
5).toInt();
        best_2620 =
getValue(xbeeString, ',',
6).toInt();
        best_kiri =
getValue(xbeeString, ',',
7).toInt();
        best_depan =
getValue(xbeeString, ',',
8).toInt();
        best_kanan =
getValue(xbeeString, ',',
9).toInt();


      }
    }
    xbeeString = "";
    xbeeComplete =
false;

   }

  }


  henti();
  tampil_menu();
}

void pergerakan_robot()
{
  error =
headingDegrees - Sp;
  jumlah_error += (0.001
* error);
  selisih_error = error -
error_sebelumnya;
  error_sebelumnya =
error;

  P = Kp * error;
  D = Kd * selisih_error;
  I = Ki * jumlah_error;

  PID = P + I + D;

  if (arah == 0 ) {
    motor_krB(Speed -
PID);
    motor_krD(Speed -
PID);
    motor_knB(Speed +
PID);
    motor_knD(Speed +
PID);
    tempuh_y ++;
  } else if (arah == 2) {
    motor_krB(-Speed -
PID);
    motor_krD(Speed -
PID);
    motor_knB(Speed +
PID);
    motor_knD(-Speed +
PID);
    tempuh_x ++ ;
  } else if (arah == 1) {
    motor_krB(Speed -
PID);
    motor_krD(-Speed -
PID);
    motor_knB(-Speed +
PID);
    motor_knD(Speed +
PID);
    tempuh_x -- ;
  } else {
    motor_krB(-Speed -
PID);
    motor_krD(-Speed -
PID);
    motor_knB(-Speed +
PID);
    motor_knD(-Speed +
PID);
    tempuh_y --;
  }

  lcd.setCursor(0, 0);
  lcd.print("E");
lcd.print(error);
lcd.print("  ");
lcd.print(tempuh_x);
lcd.print("  ");
lcd.print(tempuh_y);
  lcd.setCursor(0, 1);
  lcd.print("P");
lcd.print(P); lcd.print("
");
  lcd.print("D");
lcd.print(D); lcd.print("
");
  lcd.print("I");
lcd.print(P); lcd.print("
");

}

int getJumlah(String
data, char separator, int
index)
{
  int found = 0;
  int strIndex[] = {0, -1};
  int maxIndex =
data.length() - 1;

  for (int i = 0; i <=
maxIndex; i++) {
    if (data.charAt(i) ==
separator || i ==
maxIndex) {
      found++;
      //strIndex[0] =
strIndex[1] + 1;
      //strIndex[1] = (i ==
maxIndex) ? i + 1 : i;
    }
  }
  return found;
}

void henti() {
  motor_krB(0);
  motor_knD(0);
  motor_krD(0);
  motor_knB(0);

}

void save_eeprom(int
direccion, float num)
{
  long valor = num *
10000;

  byte cuatro = (valor &
0xFF);
  byte tres = ((valor >>
8) & 0xFF);
  byte dos = ((valor >>
16) & 0xFF);
```

```
  byte uno = ((valor >>
24) & 0xFF);


EEPROM.write(direccion,
cuatro);
  EEPROM.write(direccion
+ 1, tres);
  EEPROM.write(direccion
+ 2, dos);
  EEPROM.write(direccion
+ 3, uno);
}

float load_eeprom(long
direccion)
{

  long cuatro =
EEPROM.read(direccion);
  long tres =
EEPROM.read(direccion
+ 1);
  long dos =
EEPROM.read(direccion
+ 2);
  long uno =
EEPROM.read(direccion
+ 3);

  float num = ((cuatro
<< 0) & 0xFF) + ((tres
<< 8) & 0xFFFF) + ((dos
<< 16) & 0xFFFFFF) +
((uno << 24) &
0xFFFFFFFF);
  return (num / 10000);
}

void simpan_semua ()
{


  save_eeprom (0, Kp);
  save_eeprom (4, Kd);
  save_eeprom (8, Ki);
  save_eeprom (12, Sp);
  save_eeprom (16,
Speed);




//Serial.println("SimpanO
K");
}

void set_variabel()
{

  Kp = load_eeprom(0);
  Kd = load_eeprom(4);
  Ki = load_eeprom(8);
  Sp = load_eeprom(12);
  Speed =
load_eeprom(16);



//Serial.println("Variables
inicializadas");
}

void setKp() {
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(Kp);

  while
(digitalRead(btn_Ok) ==
1) {
    if (digitalRead(btn_Up)
== 0) {
      delay(20);
      Kp = Kp + 0.01;
      lcd.setCursor(0, 0);
      lcd.print(Kp);
      lcd.print("  ");
    }
    if
(digitalRead(btn_Down)
== 0) {
      delay(20);
      Kp = Kp - 0.01;
      lcd.setCursor(0, 0);
      lcd.print(Kp);
      lcd.print("  ");
    }
  }
  delay (1000);
  lcd.setCursor(0, 0);
  lcd.print("Set KP
Selesai");
  simpan_semua();
  delay(1000);


  tampil_menu();

}

void setKd() {
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(Kd);

  while
(digitalRead(btn_Ok) ==
1) {
    if (digitalRead(btn_Up)
== 0) {
      delay(20);
      Kd = Kd + 0.01;
      lcd.setCursor(0, 0);
      lcd.print(Kd);
      lcd.print("  ");
    }
    if
(digitalRead(btn_Down)
== 0) {
      delay(20);
      Kd = Kd - 0.01;
      lcd.setCursor(0, 0);
      lcd.print(Kd);
      lcd.print("  ");
    }
  }
  delay (1000);
  lcd.setCursor(0, 0);
  lcd.print("Set KD
Selesai");
  simpan_semua();
  delay (1000);


  tampil_menu();
}

void setKi() {
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(Ki);

  while
(digitalRead(btn_Ok) ==
1) {
    if (digitalRead(btn_Up)
== 0) {
      delay(20);
      Ki = Ki + 0.01;
      lcd.setCursor(0, 0);
      lcd.print(Ki);
      lcd.print("  ");
    }
    if
(digitalRead(btn_Down)
== 0) {
      delay(20);
      Ki = Ki - 0.01;
      lcd.setCursor(0, 0);
      lcd.print(Ki);
      lcd.print("  ");
    }
  }
  delay (1000);
  lcd.setCursor(0, 0);
  lcd.print("Set KI
Selesai");
  simpan_semua();
  delay (1000);


  tampil_menu();
}

void setSp() {
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(Sp);

  while
(digitalRead(btn_Ok) ==
1) {
    if (digitalRead(btn_Up)
== 0) {
      delay(20);
      Sp = Sp + 0.01;
      lcd.setCursor(0, 0);
      lcd.print(Sp);
      lcd.print("  ");
    }
    if
(digitalRead(btn_Down)
== 0) {
      delay(20);
      Sp = Sp - 0.01;
      lcd.setCursor(0, 0);
      lcd.print(Sp);
      lcd.print("  ");
    }
  }
  delay (1000);
  lcd.setCursor(0, 0);
  lcd.print("Set Sp
Selesai");
  simpan_semua();
  delay (1000);


  tampil_menu();
}

void setSpd() {
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(Speed);

  while
(digitalRead(btn_Ok) ==
1) {
    if (digitalRead(btn_Up)
== 0) {
      delay(20);
      Speed = Speed + 1;
      lcd.setCursor(0, 0);
      lcd.print(Speed);
      lcd.print("  ");
    }
    if
(digitalRead(btn_Down)
== 0) {
      delay(20);
      Speed = Speed - 1;
      lcd.setCursor(0, 0);
      lcd.print(Speed);
      lcd.print("  ");
    }
  }
  delay (1000);
  lcd.setCursor(0, 0);
  lcd.print("Set Spd
Selesai");
  simpan_semua();
  delay(1000);


  tampil_menu();
}


void motor_krB(int v) {

  if (v == 0) {
    digitalWrite(in1_krB,
1);
    digitalWrite(in2_krB,
1);

  } else if (v > 0) {
    if (v > 255)v = 255;
    digitalWrite(in1_krB,
1);
    digitalWrite(in2_krB,
0);
    analogWrite(ena_krB,
v);
  } else {
    v = abs(v);
    if (v > 255)v = 255;
    digitalWrite(in1_krB,
0);
    digitalWrite(in2_krB,
1);
    analogWrite(ena_krB,
v);
  }

  // put your main code
here, to run repeatedly:

}

void motor_knD(int v) {

  if (v == 0) {
    digitalWrite(in3_knD,
1);
    digitalWrite(in4_knD,
1);

  } else if (v > 0) {
    if (v > 255)v = 255;
    digitalWrite(in3_knD,
0);
    digitalWrite(in4_knD,
1);
    analogWrite(enb_knD,
v);
  } else {
    v = abs(v);
    if (v > 255)v = 255;
    digitalWrite(in3_knD,
1);
    digitalWrite(in4_knD,
0);
    analogWrite(enb_knD,
v);
  }

  // put your main code
here, to run repeatedly:

}

void motor_krD(int v) {

  if (v == 0) {
    digitalWrite(in3_krD,
1);
    digitalWrite(in4_krD,
1);

  } else if (v > 0) {
    if (v > 255)v = 255;
    digitalWrite(in3_krD,
1);
    digitalWrite(in4_krD,
0);
    analogWrite(enb_krD,
v);
  } else {
    v = abs(v);
    if (v > 255)v = 255;
    digitalWrite(in3_krD,
0);
    digitalWrite(in4_krD,
1);
```

```
    analogWrite(enb_krD,
v);
  }

  // put your main code
here, to run repeatedly:

}

void motor_knB(int v) {

  if (v == 0) {
    digitalWrite(in1_knB,
1);
    digitalWrite(in2_knB,
1);

  } else if (v > 0) {
    if (v > 255)v = 255;
    digitalWrite(in1_knB,
1);
    digitalWrite(in2_knB,
0);
    analogWrite(ena_knB,
v);
  } else {
    v = abs(v);
    if (v > 255)v = 255;
    digitalWrite(in1_knB,
0);
    digitalWrite(in2_knB,
1);
    analogWrite(ena_knB,
v);
  }

  // put your main code
here, to run repeatedly:

}


void baca_depan() {
  digitalWrite(trig_depan,
LOW);
  delay(2);
  digitalWrite(trig_depan,
HIGH);
  delay(5);
  digitalWrite(trig_depan,
LOW);
  jarak_depan =
pulseIn(echo_depan,
HIGH);
  jarak_depan =
microsecondsToCentimet
ers(jarak_depan);
  if (jarak_depan <= 0)
jarak_depan =
jarak_aman + 1 ;
  Serial.print("Depan =
");
Serial.println(jarak_depa
n);
}

void baca_kanan() {
  digitalWrite(trig_kanan,
LOW);
  delay(2);
  digitalWrite(trig_kanan,
HIGH);
  delay(5);
  digitalWrite(trig_kanan,
LOW);
  jarak_kanan =
pulseIn(echo_kanan,
HIGH);
  jarak_kanan =
microsecondsToCentimet
ers(jarak_kanan);
  if (jarak_kanan <= 0)
jarak_kanan =
jarak_aman + 1;
  Serial.print("Kanan =
");
Serial.println(jarak_kana
n);
}

void baca_kiri() {
  digitalWrite(trig_kiri,
LOW);
  delay(2);
  digitalWrite(trig_kiri,
HIGH);
  delay(5);
  digitalWrite(trig_kiri,
LOW);
  jarak_kiri =
pulseIn(echo_kiri, HIGH);
  jarak_kiri =
microsecondsToCentimet
ers(jarak_kiri);
  if (jarak_kiri <= 0)
jarak_kiri = jarak_aman
+ 1 ;
  Serial.print("Kiri = ");
Serial.println(jarak_kiri);
}

String getValue(String
data, char separator, int
index)
{
  int found = 0;
  int strIndex[] = {0, -1};
  int maxIndex =
data.length() - 1;

  for (int i = 0; i <=
maxIndex && found <=
index; i++) {
    if (data.charAt(i) ==
separator || i ==
maxIndex) {
      found++;
      strIndex[0] =
strIndex[1] + 1;
      strIndex[1] = (i ==
maxIndex) ? i + 1 : i;
    }
  }

  return found > index ?
data.substring(strIndex[
0], strIndex[1]) : "";
}
/*
  SerialEvent occurs
whenever a new data
comes in the
 hardware serial RX.
This routine is run
between each
 time loop() runs, so
using delay inside loop
can delay
 response.  Multiple
bytes of data may be
available.
 */
void raspiEvent() {
  while (raspi.available())
{
    // get the new byte:
    char inChar =
(char)raspi.read();
    // add it to the
inputString:
    if (inChar == '\r') {

    } else if (inChar ==
'\n') {
    raspiComplete = true;
    } else {

      raspiString +=
inChar;
    }
    // if the incoming
character is a newline,
set a flag
```

```
    // so the main loop
can do something about
it:

  }
}


void xbeeEvent() {
  while (xbee.available())
{
    // get the new byte:
    char inChar =
(char)xbee.read();
    // add it to the
inputString:
    if (inChar == '\r') {
      xbeeComplete =
true;
    } else if (inChar ==
'\n') {

    } else {

      xbeeString +=
inChar;
    }
    // if the incoming
character is a newline,
set a flag
    // so the main loop
can do something about
it:

  }
}


void test_raspi() {
  lcd.setCursor(0, 0);
  lcd.print("RASPI TEST");

  while
(digitalRead(btn_Ok) ==
1) {
    t2600 =
analogRead(tgs2600);
    t2602 =
analogRead(tgs2602);
    t2620 =
analogRead(tgs2620);

    raspi.print(t2600);
    raspi.print(",");
    raspi.print(t2602);
    raspi.print(",");
    raspi.println(t2620);

    delay(1000);

    raspiEvent(); //call the
function
    // print the string
when a newline arrives:
    if (raspiComplete) {

Serial.println(raspiString)
;
      lcd.setCursor(0, 1);
      lcd.print(raspiString);
      lcd.print("     ");

      raspiString = "";
      raspiComplete =
false;
    }
  }
  delay (1000);
  lcd.setCursor(0, 0);
  lcd.print("TEST SVM
Selesai");
  simpan_semua();
  delay(1000);

  tampil_menu();
}

String uji_sample(int a,
int b, int c) {
  //  String
  raspiString = "";
  raspiComplete = false;

  raspi.print(a);
  raspi.print(",");
  raspi.print(b);
  raspi.print(",");
  raspi.println(c);

  delay(200);

  raspiEvent(); //call the
function
  // print the string when
a newline arrives:
  if (raspiComplete) {

Serial.println(raspiString)
;
    return raspiString;

  }

}
```

Arduino MEGA 2560



Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

Technical Specifications — Page 2

How to use Arduino
Programming Enviroment, Basic Tutorials — Page 6

Terms & Conditions — Page 7

Enviromental Policies
half sqm of green via Impatto Zero® — Page 7

radiospares RADIONICS ALLIED ELECTRONICS AN ELECTROCOMPONENTS COMPANY

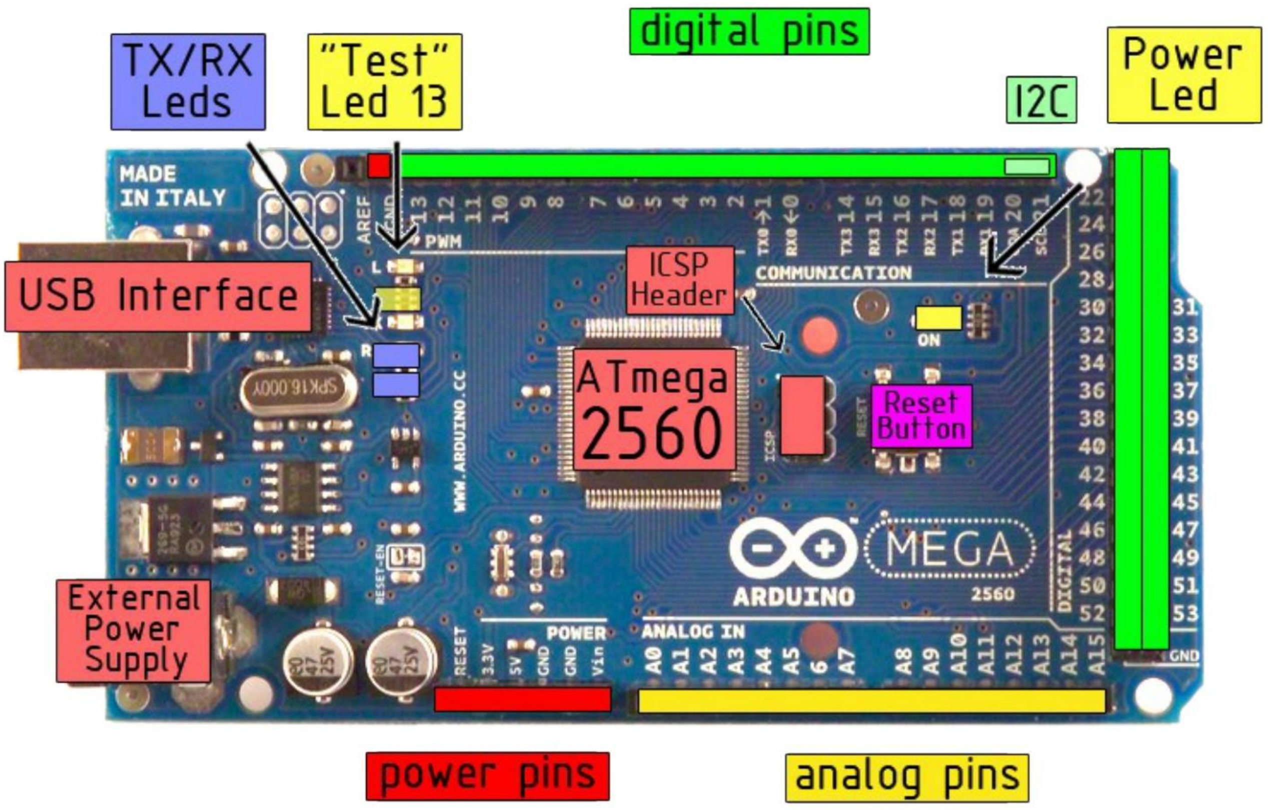
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode() , digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I^{2}C: 20 (SDA) and 21 (SCL).** Support I$_2$C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I$_2$C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

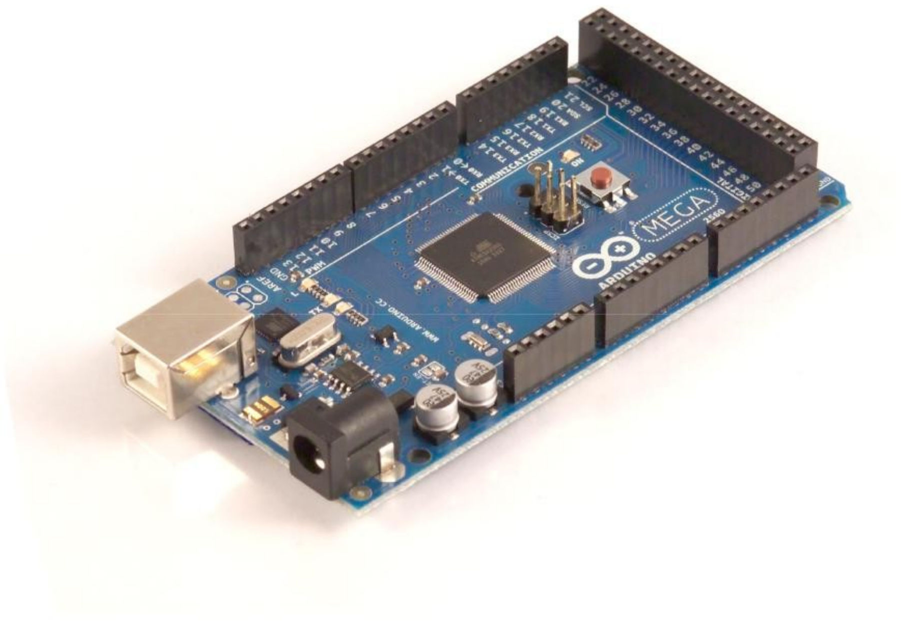
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.



Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto -reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I^{2}C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install **Windows Install** **Mac Install**

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

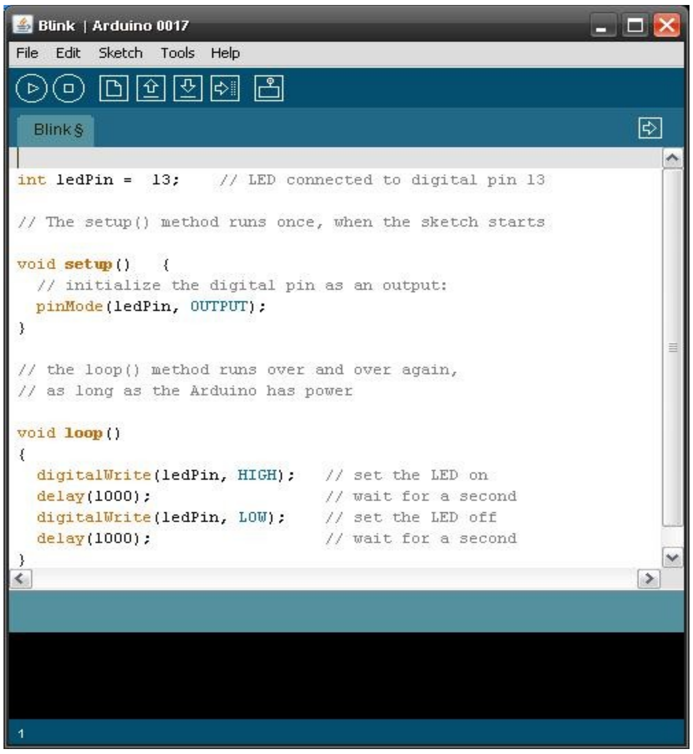
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

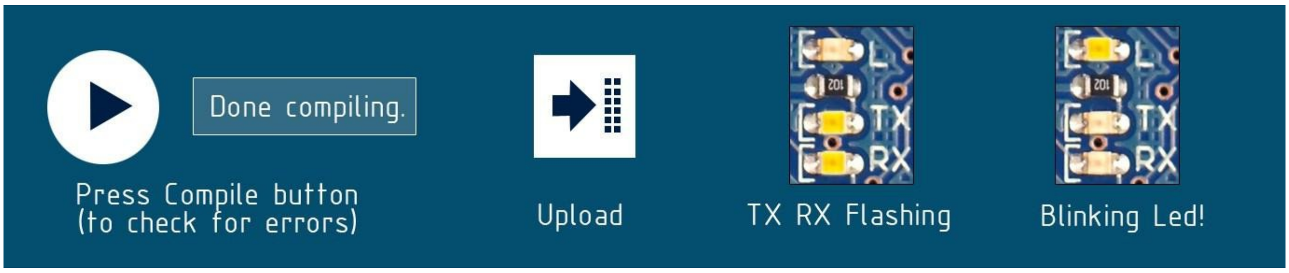
Once you have your skecth you'll see something very close to the screenshot on the right.

In **Tools>Board** select MEGA

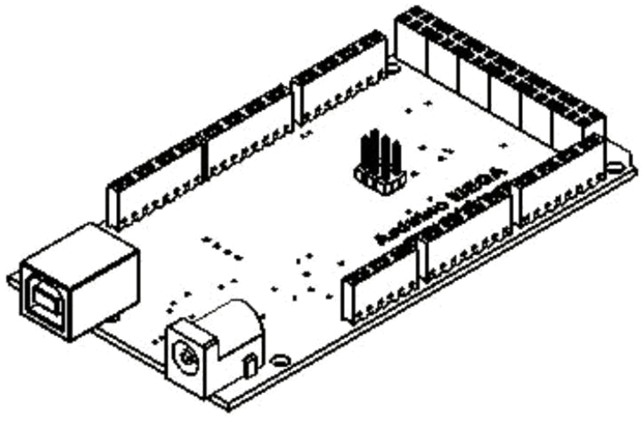
Now you have to go to
Tools>SerialPort
and select the right serial port,
the one arduino is attached to.

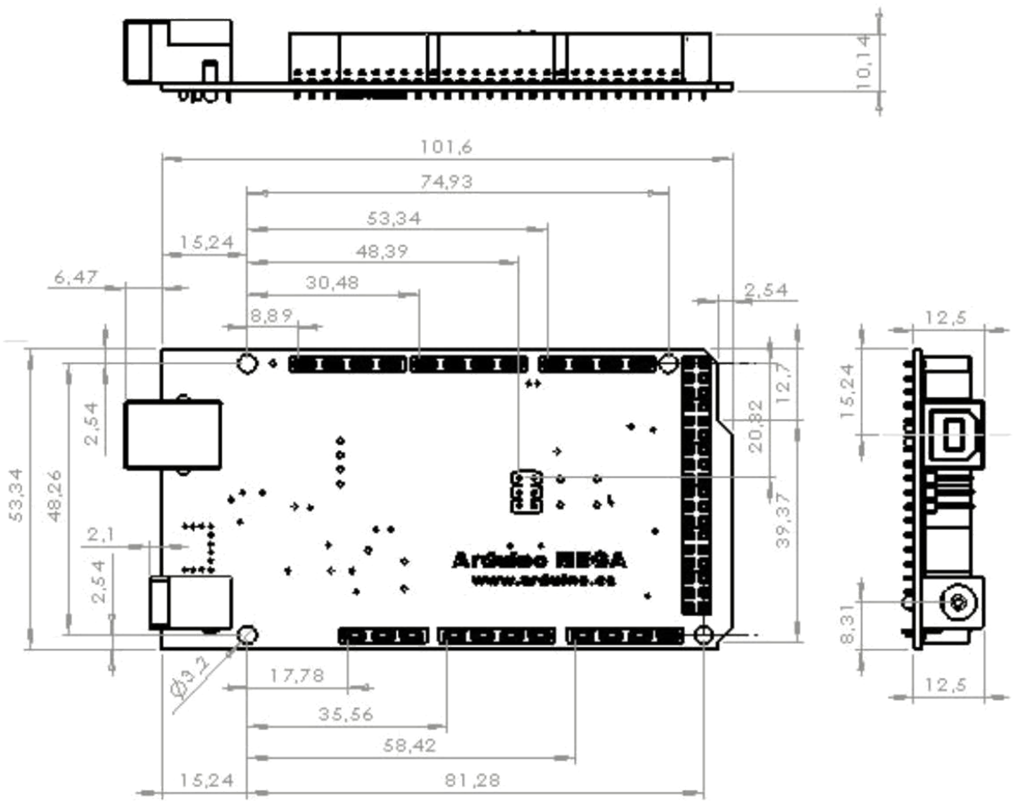


```
int ledPin =  13;    // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

void setup()   {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power

void loop()
{
  digitalWrite(ledPin, HIGH);   // set the LED on
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // set the LED off
  delay(1000);                  // wait for a second
}
```



RS radiospares RADIONICS ALLIED ELECTRONICS AN ELECTROCOMPONENTS COMPANY




10,14
101,6
74,93
53,34
48,39
30,48
15,24
8,89
6,47
2,54
12,5
12,7
20,32
15,24
39,37
53,34
48,26
2,54
2,1
2,54
8,31
12,5
Ø3,2
17,78
35,56
58,42
15,24
81,28
Arduino MEGA
www.arduino.cc

Terms & Conditions

1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

Enviromental Policies

The producer of Arduino ™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.

Ultrasonic Ranging Module HC - SR04

Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:

- Using IO trigger for at least 10us high level signal,
- The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
- IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.

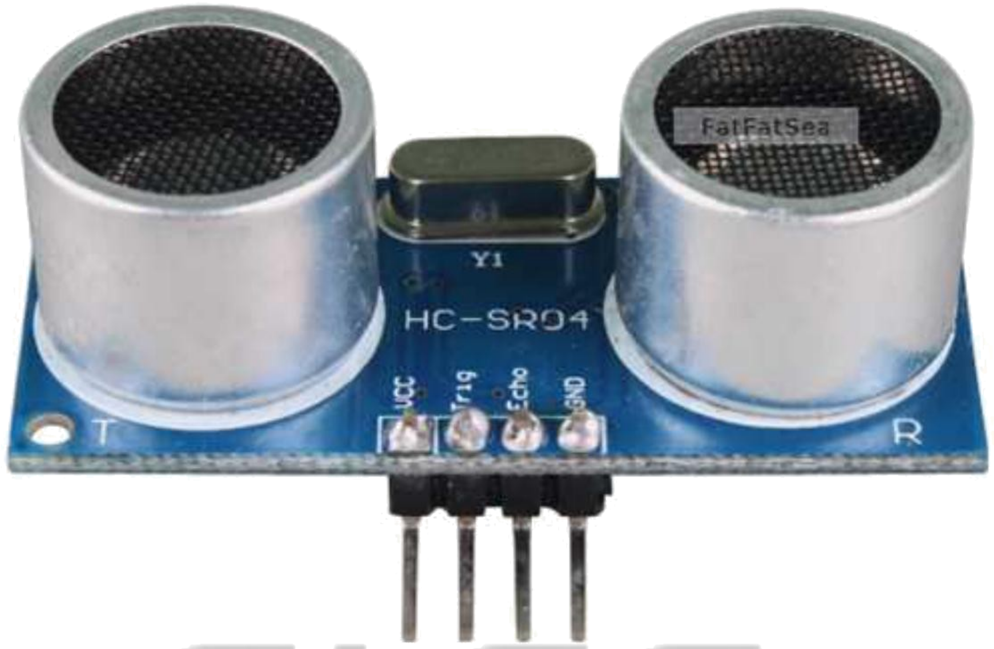
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

5V Supply
Trigger Pulse Input
Echo Pulse Output
0V Ground

Electric Parameter

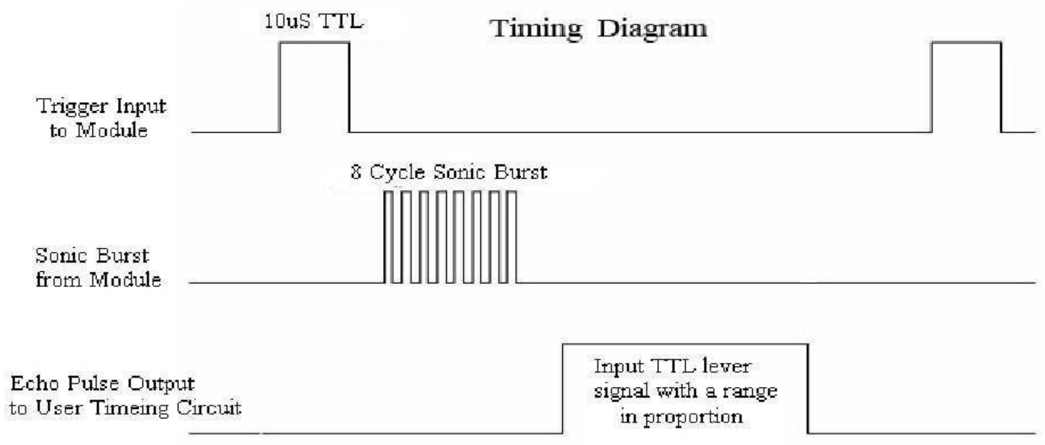
Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**



Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.


10uS TTL
Timing Diagram
Trigger Input to Module
8 Cycle Sonic Burst
Sonic Burst from Module
Echo Pulse Output to User Timeing Circuit
Input TTL lever signal with a range in proportion

Attention:

The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise,
it will affect the normal work of the module.

When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

www.Elecfreaks.com

FIGARO

PRODUCT INFORMATION

TGS 2600 - for the detection of Air Contaminants

Features:

* Low power consumption
* High sensitivity to gaseous air contaminants
* Long life and low cost
* Uses simple electrical circuit
* Small size

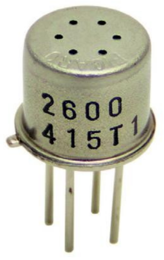
Applications:

* Air cleaners
* Ventilation control
* Air quality monitors

The sensing element is comprised of a metal oxide semiconductor layer formed on an alumina substrate of a sensing chip together with an integrated heater. In the presence of a detectable gas, the sensor's conductivity increases depending on the gas concentration in the air. A simple electrical circuit can convert the change in conductivity to an output signal which corresponds to the gas concentration.

The **TGS 2600** has high sensitivity to low concentrations of gaseous air contaminants such as hydrogen and carbon monoxide which exist in cigarette smoke. The sensor can detect hydrogen at a level of several ppm.

Due to miniaturization of the sensing chip, TGS 2600 requires a heater current of only 42mA and the device is housed in a standard TO-5 package.



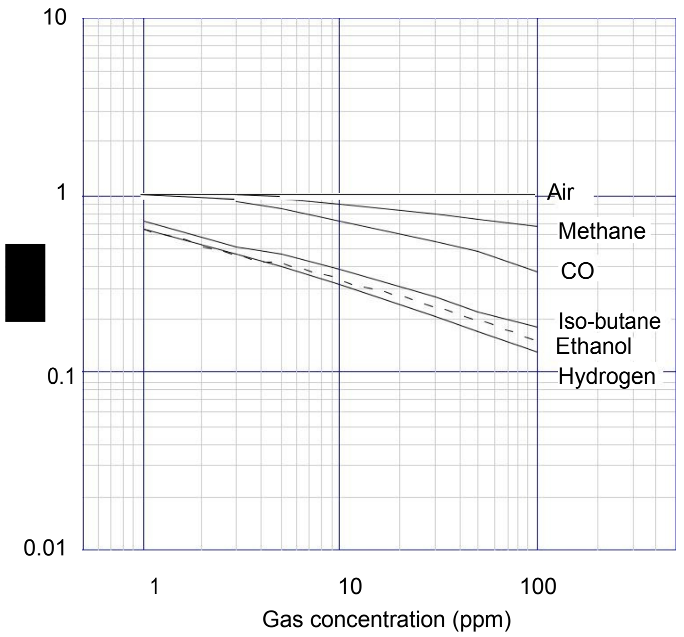
The figure below represents typical sensitivity characteristics, all data having been gathered at standard test conditions (see reverse side of this sheet). The Y-axis is indicated as sensor resistance ratio (Rs/Ro) which is defined as follows:

Rs = Sensor resistance in displayed gases at various concentrations
Ro = Sensor resistance in fresh air

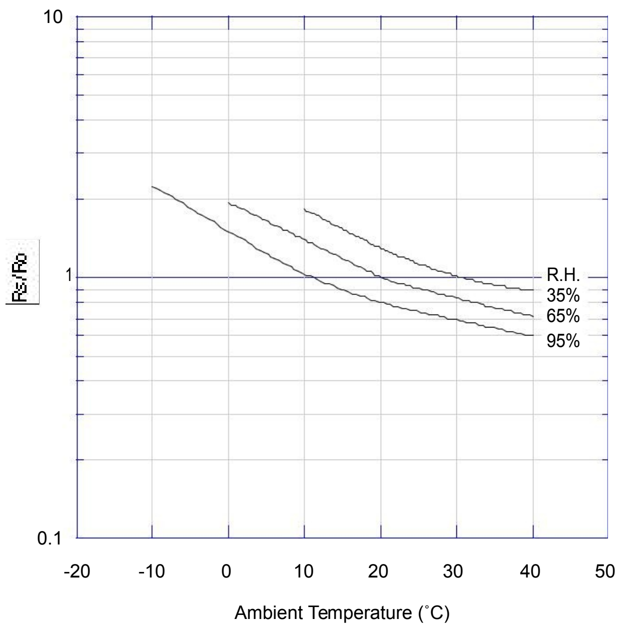
The figure below represents typical temperature and humidity dependency characteristics. Again, the Y-axis is indicated as sensor resistance ratio (Rs/Ro), defined as follows:

Rs = Sensor resistance in fresh air at various temperatures/humidities
Ro = Sensor resistance in fresh air at 20°C and 65% R.H.

Sensitivity Characteristics:



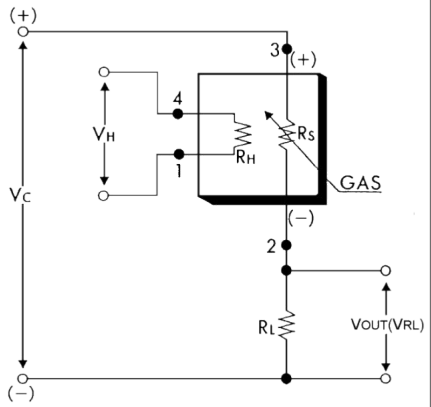
Temperature/Humidity Dependency:



IMPORTANT NOTE: OPERATING CONDITIONS IN WHICH FIGARO SENSORS ARE USED WILL VARY WITH EACH CUSTOMER'S SPECIFIC APPLICATIONS. FIGARO STRONGLY RECOMMENDS CONSULTING OUR TECHNICAL STAFF BEFORE DEPLOYING FIGARO SENSORS IN YOUR APPLICATION AND, IN PARTICULAR, WHEN CUSTOMER'S TARGET GASES ARE NOT LISTED HEREIN. FIGARO CANNOT ASSUME ANY RESPONSIBILITY FOR ANY USE OF ITS SENSORS IN A PRODUCT OR APPLICATION FOR WHICH SENSOR HAS NOT BEEN SPECIFICALLY TESTED BY FIGARO.

Basic Measuring Circuit:

The sensor requires two voltage inputs: heater voltage (V_H) and circuit voltage (V_C). The heater voltage (V_H) is applied to the integrated heater in order to maintain the sensing element at a specific temperature which is optimal for sensing. Circuit voltage (V_C) is applied to allow measurement of voltage (V_{OUT}) across a load resistor (R_L) which is connected in series with the sensor. DC voltage is required for the circuit voltage since the sensor has a polarity. A common power supply circuit can be used for both V_C and V_H to fulfill the sensor's electrical requirements. The value of the load resistor (R_L) should be chosen to optimize the alarm threshold value, keeping power consumption (P_S) of the semiconductor below a limit of 15mW. Power consumption (P_S) will be highest when the value of Rs is equal to R_L on exposure to gas.



Specifications:

Model number			TGS2600-B00	
Sensing principle			MOS type	
Standard package			TO-5 metal can	
Target gases			Air contaminants (hydrogen, ethanol, etc.)	
Typical detection range			1 ~ 30ppm of H_2	
Standard circuit conditions	Heater voltage	V_H	5.0±0.2V AC/DC	
	Circuit voltage	V_C	5.0±0.2V DC	P_S≤15mW
	Load resistance	R_L	variable	0.45kΩ min.
Electrical characteristics under standard test conditions	Heater resistance	R_H	approx 83Ω at room temp. (typical)	
	Heater current	I_H	42±4mA	
	Heater power consumption	P_H	210mW	V_H=5.0V DC
	Sensor resistance	R_S	10kΩ ~ 90kΩ in air	
	Sensitivity (change ratio of Rs)		0.3~0.6	$\frac{\text{Rs (10ppm of } H_2)}{\text{Rs air}}$
Standard test conditions	Test gas conditions		normal air at 20±2˚C, 65±5%RH	
	Circuit conditions		Vc = 5.0±0.01V DC V_H = 5.0±0.05V DC	
	Conditioning period before test		7 days	

The value of power consumption (P_S) can be calculated by utilizing the following formula:

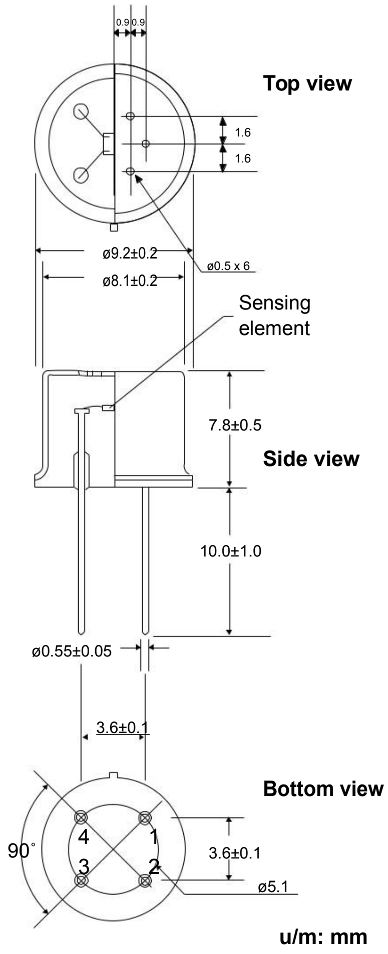
$$P_S = \frac{(V_C - V_{RL})^2}{R_S}$$

Sensor resistance (Rs) is calculated with a measured value of $V_{OUT}(V_{RL})$ by using the following formula:

$$R_S = \left(\frac{V_C}{V_{RL}} - 1\right) \times R_L$$

All sensor characteristics shown in this brochure represent typical characteristics. Actual characteristics vary from sensor to sensor. The only characteristics warranted are those in the Specification table above.

Structure and Dimensions:



Pin connection:

1:	Heater
2:	Sensor electrode (-)
3:	Sensor electrode (+)
4:	Heater

FIGARO

PRODUCT INFORMATION

TGS 2602 - for the detection of Air Contaminants

Features:

* High sensitivity to VOCs and odorous gases
* Low power consumption
* High sensitivity to gaseous air contaminants
* Long life
* Uses simple electrical circuit
* Small size

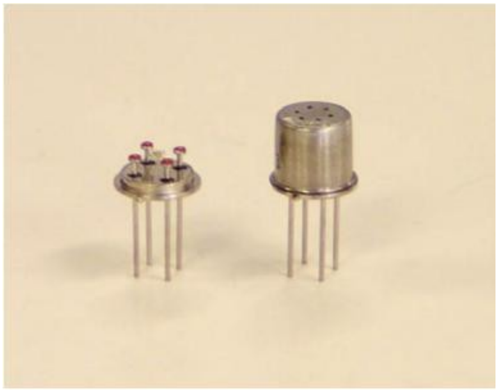
Applications:

* Air cleaners
* Ventilation control
* Air quality monitors
* VOC monitors
* Odor monitors

The sensing element is comprised of a metal oxide semiconductor layer formed on the alumina substrate of a sensing chip together with an integrated heater. In the presence of detectable gas, sensor conductivity increases depending on gas concentration in the air. A simple electrical circuit can convert the change in conductivity to an output signal which corresponds to the gas concentration.

The **TGS 2602** has high sensitivity to low concentrations of odorous gases such as ammonia and H_2S generated from waste materials in office and home environments. The sensor also has high sensitivity to low concentrations of VOCs such as toluene emitted from wood finishing and construction products. Figaro also offers a microprocessor (FIC02667) which contains special software for handling the sensor's signal for appliance control applications.

Due to miniaturization of the sensing chip, TGS 2602 requires a heater current of only 42mA and the device is housed in a standard TO-5 package.



The figure below represents typical sensitivity characteristics, all data having been gathered at standard test conditions (see reverse side of this sheet). The Y-axis is indicated as sensor resistance ratio (Rs/Ro) which is defined as follows:

Rs = Sensor resistance in displayed gases at various concentrations

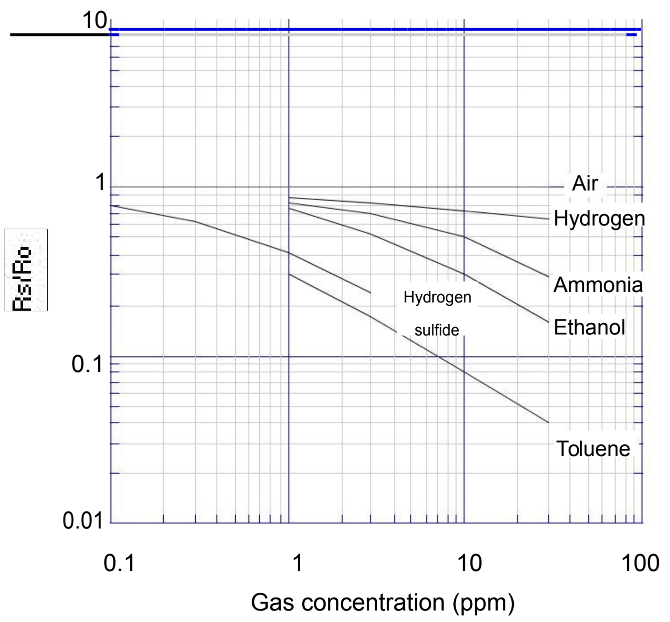
Ro = Sensor resistance in fresh air

The figure below represents typical temperature and humidity dependency characteristics. Again, the Y-axis is indicated as sensor resistance ratio (Rs/Ro), defined as follows:

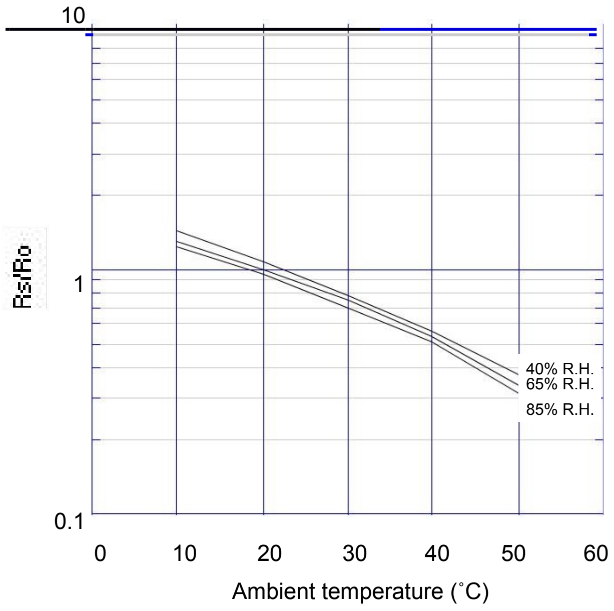
Rs = Sensor resistance in fresh air at various temperatures/humidities

Ro = Sensor resistance in fresh air at 20°C and 65% R.H.

Sensitivity Characteristics :

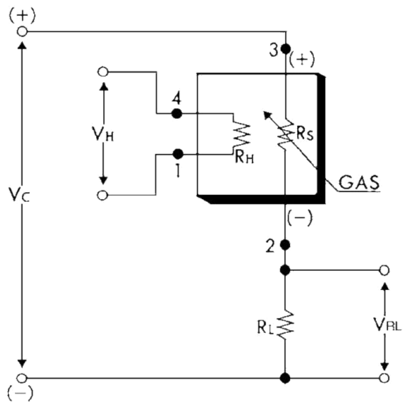


Temperature/Humidity Dependency:



Basic Measuring Circuit:

The sensor requires two voltage inputs: heater voltage (V_H) and circuit voltage (V_C). The heater voltage (V_H) is applied to the integrated heater in order to maintain the sensing element at a specific temperature which is optimal for sensing. Circuit voltage (V_C) is applied to allow measurement of voltage (Vout) across a load resistor (R_L) which is connected in series with the sensor. DC voltage is required for the circuit voltage since the sensor has a polarity. A common power supply circuit can be used for both V_C and V_H to fulfill the sensor's electrical requirements. The value of the load resistor (R_L) should be chosen to optimize the alarm threshold value, keeping power consumption (P_S) of the semiconductor below a limit of 15mW. Power consumption (P_S) will be highest when the value of Rs is equal to R_L on exposure to gas.



Specifications:

Model number			TGS 2602-B00	
Sensing element type			D1	
Standard package			TO-5 metal can	
Target gases			Air contaminants	
Typical detection range			1 ~ 30 ppm of EtOH	
Standard circuit conditions	Heater voltage	V_H	5.0±0.2V DC/AC	
	Circuit voltage	V_C	5.0±0.2V DC	Ps≤15mW
	Load resistance	R_L	Variable	0.45kΩ min.
Electrical characteristics under standard test conditions	Heater resistance	R_H	approx. 59Ω at room temp.	
	Heater current	I_H	56±5mA	
	Heater power consumption	P_H	280mW (typical)	
	Sensor resistance	Rs	10k~100kΩ in air	
	Sensitivity (change ratio of Rs)		0.15~0.5	$\frac{\text{Rs (10ppm of EtOH)}}{\text{Rs (air)}}$
Standard test conditions	Test gas conditions		normal air at 20±2˚C, 65±5%RH	
	Circuit conditions		V_C = 5.0±0.01V DC V_H = 5.0±0.05V DC	
	Conditioning period before test		7 days	

The value of power consumption (P_S) can be calculated by utilizing the following formula:

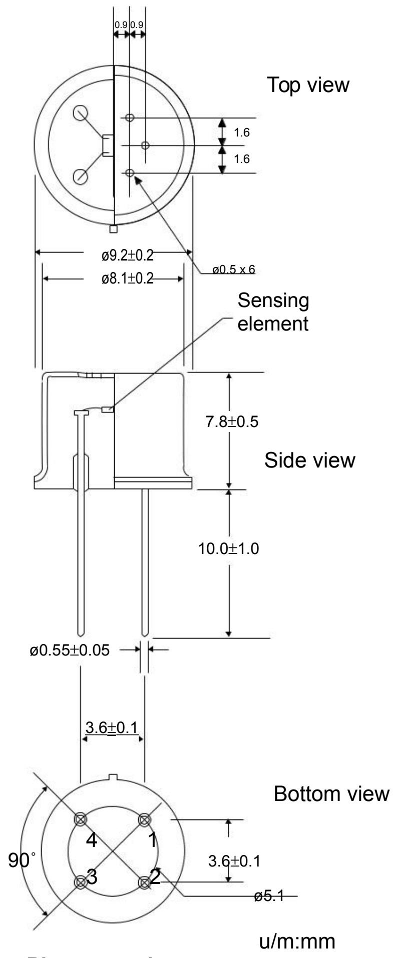
$$P_S = \frac{(V_C - V_{out})^2}{R_S}$$

Sensor resistance (Rs) is calculated with a measured value of Vout by using the following formula:

$$R_S = \frac{V_C \times R_L}{V_{out}} - R_L$$

For information on warranty, please refer to Standard Terms and Conditions of Sale of Figaro USA Inc. All sensor characteristics shown in this brochure represent typical characteristics. Actual characteristics vary from sensor to sensor. The only characteristics warranted are those in the Specification table above.

Structure and Dimensions:



Pin connection:
1: Heater
2: Sensor electrode (-)
3: Sensor electrode (+)
4: Heater

FIGARO USA, INC.
121 S. Wilke Rd. Suite 300
Arlington Heights, IL 60005
Phone: (847)-832-1701
Fax: (847)-832-1705
e-mail: figarousa@figarosensor.com

FIGARO

REV: 01/05

PRODUCT INFORMATION

TGS 2620 - for the detection of Solvent Vapors

Features:

* Low power consumption
* High sensitivity to alcohol and organic solvent vapors
* Long life and low cost
* Uses simple electrical circuit

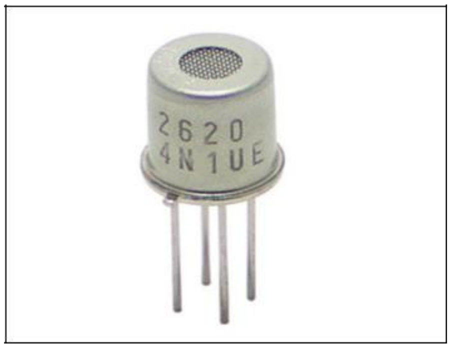
Applications:

* Alcohol testers
* Organic vapor detectors/alarms
* Solvent detectors for factories, dry cleaners, and semiconductor industries

The sensing element is comprised of a metal oxide semiconductor layer formed on an alumina substrate of a sensing chip together with an integrated heater. In the presence of a detectable gas, the sensor's conductivity increases depending on the gas concentration in the air. A simple electrical circuit can convert the change in conductivity to an output signal which corresponds to the gas concentration.

The **TGS 2620** has high sensitivity to the vapors of organic solvents as well as other volatile vapors, making it suitable for organic vapor detectors/alarms.

Due to miniaturization of the sensing chip, TGS 2620 requires a heater current of only 42mA and the device is housed in a standard TO-5 package.



The figure below represents typical sensitivity characteristics, all data having been gathered at standard test conditions (see reverse side of this sheet). The Y-axis is indicated as sensor resistance ratio (Rs/Ro) which is defined as follows:

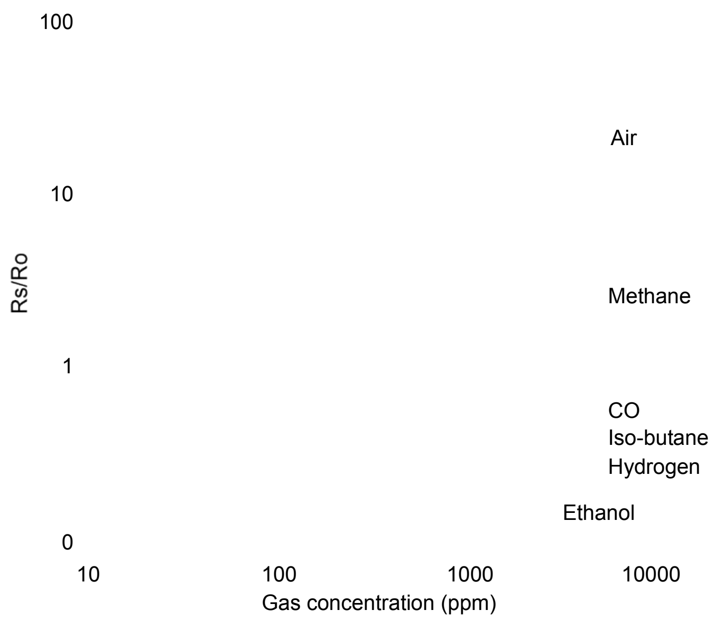
Rs = Sensor resistance in displayed gases at various concentrations
Ro = Sensor resistance in 300ppm of ethanol

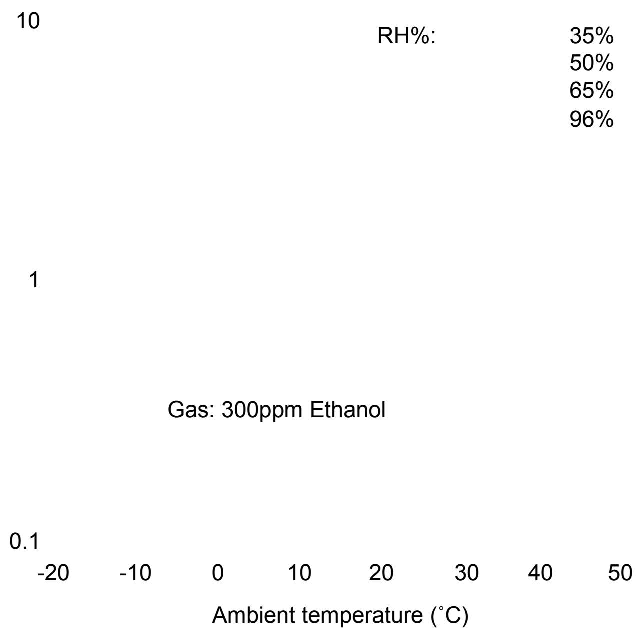
Sensitivity Characteristics:

The figure below represents typical temperature and humidity dependency characteristics. Again, the Y-axis is indicated as sensor resistance ratio (Rs/Ro), defined as follows:

Rs = Sensor resistance in 300ppm of ethanol at various temperatures/humidities
Ro = Sensor resistance in 300ppm of ethanol at 20°C and 65% R.H.

Temperature/Humidity Dependency:

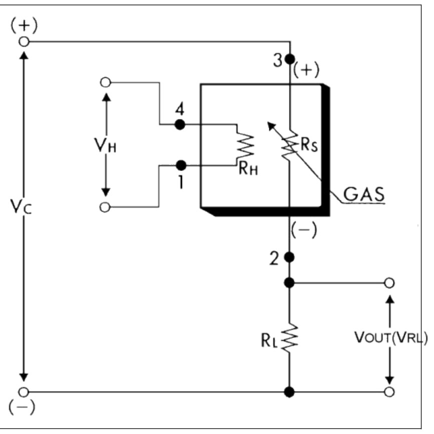




IMPORTANT NOTE: OPERATING CONDITIONS IN WHICH FIGARO SENSORS ARE USED WILL VARY WITH EACH CUSTOMER'S SPECIFIC APPLICATIONS. FIGARO STRONGLY RECOMMENDS CONSULTING OUR TECHNICAL STAFF BEFORE DEPLOYING FIGARO SENSORS IN YOUR APPLICATION AND, IN PARTICULAR, WHEN CUSTOMER'S TARGET GASES ARE NOT LISTED HEREIN. FIGARO CANNOT ASSUME ANY RESPONSIBILITY FOR ANY USE OF ITS SENSORS IN A PRODUCT OR APPLICATION FOR WHICH SENSOR HAS NOT BEEN SPECIFICALLY TESTED BY FIGARO.

Basic Measuring Circuit:

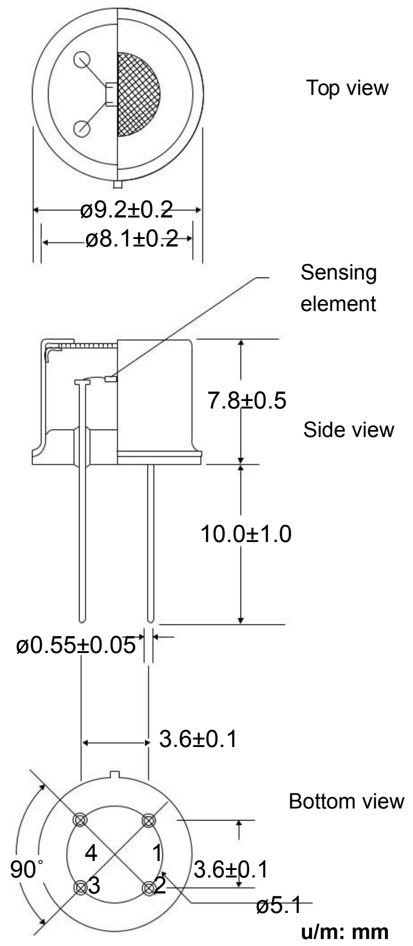
The sensor requires two voltage inputs: heater voltage (V_H) and circuit voltage (V_C). The heater voltage (V_H) is applied to the integrated heater in order to maintain the sensing element at a specific temperature which is optimal for sensing. Circuit voltage (V_C) is applied to allow measurement of voltage (V_{OUT}) across a load resistor (R_L) which is connected in series with the sensor. A common power supply circuit can be used for both V_C and V_H to fulfill the sensor's electrical requirements. The value of the load resistor (R_L) should be chosen to optimize the alarm threshold value, keeping power consumption (P_S) of the semiconductor below a limit of 15mW. Power consumption (P_S) will be highest when the value of Rs is equal to R_L on exposure to gas.



Specifications:

Model number			TGS2620-C00	
Sensing principle			MOS-type	
Standard package			TO-5 metal can	
Target gases			Alcohol, Solvent apors	
Typical detection range			50 ~ 5,000ppm EtOH	
Standard circuit conditions	Heater voltage	V_H	5.0±0.2V AC/DC	
	Circuit voltage	V_C	5.0±0.2V DC	Ps≤15mW
	Load resistance	R_L	variable	0.45kΩ min.
Electrical characteristics under standard test conditions	Heater resistance	R_H	83Ω at room temp. (typical)	
	Heater current	I_H	42±4mA	
	Heater power consumption	P_H	210mW (typical)	
	Sensor resistance	R_S	1kΩ ~ 5kΩ in 300ppm ethanol	
	Sensitivity (change ratio of Rs)		0.3~0.5 in ethanol	$\frac{\text{Rs (300ppm)}}{\text{Rs (50ppm)}}$
Standard test conditions	Test gas conditions		Ethanol vapor in air at 20±2˚C, 65±5%RH	
	Circuit conditions		V_C = 5.0±0.01V DC V_H = 5.0±0.05V DC	
	Conditioning period before test		7 days	

Structure and Dimensions:



Pin connection:

1:	Heater
2:	Sensor electrode (-)
3:	Sensor electrode (+)
4:	Heater

The value of power dissipation (P_S) can be calculated by utilizing the following formula:

$$P_S = \frac{(V_C - V_{RL})^2}{R_S}$$

Sensor resistance (Rs) is calculated with a measured value of $V_{OUT}(V_{RL})$ by using the following formula:

$$R_S = \left(\frac{V_C}{V_{RL}} - 1\right) \times R_L$$