# PEMOGRAMAN

```
#define btn_Ok 30       //Push Button Ok pada robot
#define btn_Up 31       //Push Button Up pada robot
#define btn_Down 32     //Push Button Down pada robot

int rotasi_x = 13;
int rotasi_y = 13;


#define bt Serial1
String inputString = "";         // a string to hold incoming data
boolean stringComplete = false;  // whether the string is complete


#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 depan1 22

#define depan2 23

#define vdepan 2

#define vbelakang 3

#define belakang1 25

#define belakang2 24

#define kiri1 27

#define kiri2 26

#define vkiri 4

#define vkanan 5

#define kanan1 28

#define kanan2 29



#include <DistanceGP2Y0A21YK.h>

DistanceGP2Y0A21YK depan, skndepan, skrdepan, kr, kn, sknblkg, skrblkg,
blkg;
```

```
int jarak_depan;
int jarak_skndepan;
int jarak_skrdepan;
int jarak_kr;
int jarak_kn;
int jarak_sknblkg;
int jarak_skrblkg;
int jarak_blkg;

int posX, posY;

int arah  = 0;
int gas_depan, gas_kiri, gas_kanan;
//0 depan
//1 kiri
//2 kanan

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;
int xTujuan = 0 ;
```

```
int yTujuan = 0;
long jumlah_x = 0;
long jumlah_y = 0;


#define waktu_batas 5000

#define jarak_aman 30


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

  lcd.setBacklightPin(BACKLIGHT_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);


  depan.begin(A0);
  skndepan.begin(A1);
  skrdepan.begin(A2);
```

```
kr.begin(A3);
kn.begin(A4);
sknblkg.begin(A5);
skrblkg.begin(A6);
blkg.begin(A7);



//buka komen ini jika ingin set semua variabel menjadi 0
//  Kp = 0;
//  Kd = 0;
//  Ki = 0;
//  Sp = 0;
//  Speed = 0;
//  simpan_semua();



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(HMC5883L_RANGE_1_3GA);

// Set measurement mode
compass.setMeasurementMode(HMC5883L_CONTINOUS);

// Set data rate
compass.setDataRate(HMC5883L_DATARATE_30HZ);

// Set number of samples averaged
compass.setSamples(HMC5883L_SAMPLES_8);

// Set calibration offset. See HMC5883L_calibration.ino
compass.setOffset(5, -358);

pinMode (depan1, OUTPUT) ;
pinMode (depan2, OUTPUT) ;
pinMode (belakang1, OUTPUT) ;
pinMode (belakang2, OUTPUT) ;
pinMode (vdepan, OUTPUT) ;
pinMode (vbelakang, OUTPUT) ;
pinMode (kiri1, OUTPUT) ;
pinMode (kiri2, OUTPUT) ;
pinMode (kanan1, OUTPUT) ;
pinMode (kanan2, OUTPUT) ;
pinMode (vkiri, OUTPUT) ;
pinMode (vkanan, OUTPUT) ;
```

```
}

void loop() {
  tampil_menu();
  // put your main code here, to run repeatedly:
  //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) baca_gas();
    else if (menu == 11) Setting_X();
    else if (menu == 12) Setting_Y();

  }

  if (digitalRead(btn_Up) == 0) {
    delay(200);
    menu = menu + 1;
    if (menu > 12)menu = 1;
    tampil_menu();
```

```
  }
  if (digitalRead(btn_Down) == 0) {
    delay(200);
    menu = menu - 1;
    if (menu < 1)menu = 12;
    tampil_menu();
  }
}
//
//  baca_depan();
//  delay(500);
//  baca_kanan();
//  delay(500);
//  baca_kiri();
//  delay(500);


long microsecondsToCentimeters (long microseconds)
{
  return microseconds / 29 / 2;
}

void baca_gas() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Baca Gas       ");

  while (digitalRead(btn_Ok) == 1) {
```

```
    lcd.setCursor(0, 1);
    lcd.print(gas_kanan); lcd.print("    ");
    lcd.setCursor(5, 1);
    lcd.print(gas_depan); lcd.print("    ");
    lcd.setCursor(10, 1);
    lcd.print(gas_kiri); lcd.print("    ");

  }


  delay(1000);

  tampil_menu();

}

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


  while (digitalRead(btn_Ok) == 1) {
```

```
    baca_sensor();


  }

  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.println(headingDegrees);
  //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(headingDegrees); lcd.print("    ");
  }

  delay(1000);


  tampil_menu();
}


void tampil_menu () {

  //lcd.clear();
  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("Baca Gas    ");
  delay(200);

}

if (menu == 11) {

  //Serial..println ("Setting X");
```

```
    lcd.setCursor (0, 0);
    lcd.print("SETTING X    ");
    delay(200);

  }

  if (menu == 12) {

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

}


void start_robot() {
  error = 0;
  jumlah_error = 0;
  selisih_error = 0;
  error_sebelumnya = 0;
  set_variabel();
  Serial.println(Kp);
  Serial.println(Kd);
  Serial.println(Ki);
  xTujuan = posX;
  yTujuan = posY;
```

```
int ySekarang = 0;
int xSekarang = 0;
long jumlah_x = 0;
long jumlah_y = 0;
bool lapangan[3][3];
lapangan[0][0] = true;
lapangan[0][1] = false;
lapangan[0][2] = false;

lapangan[1][0] = false;
lapangan[1][1] = false;
lapangan[1][2] = false;

lapangan[2][0] = true;
lapangan[2][1] = false;
lapangan[2][2] = false;



inputString = "";
stringComplete = false;
lcd.setCursor(0, 0);
lcd.print("ROBOT MOVE    ");
delay(2000);
for (int i = 0 ; i < 10 ; i++) {

  baca_kompas();
  Sp = headingDegrees;
}
```

```
  while (digitalRead(btn_Ok) == 1) {
    baca_sensor();
    baca_kompas();
    Serial.print(headingDegrees); Serial.print("  v   "); Serial.println(Sp);
    error = headingDegrees - Sp;
    if (error > 180 ) error = error - 360;
    if (error < -180 ) error = 360 + error ;

    bt.print(headingDegrees); bt.print(" ");
    bt.println(error);
    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;

    Serial.print(error); Serial.print(" ");
    Serial.println(PID);



    //kalau depan bisa jalan dan tujuan depan > 1 maka jalan depan terlebih dahulu
    //jika tidak bisa maka kekanan terlebih dahulu
    if (jarak_depan > 8 && yTujuan > ySekarang &&
lapangan[xSekarang][ySekarang + 1] == false) {
      arah = 0;
```

```
   } else if (jarak_kn > 8 && xTujuan > xSekarang && lapangan[xSekarang +
1][ySekarang] == false) {

     arah = 1;

   } else if (jarak_kr > 8 && xTujuan < xSekarang && lapangan[xSekarang -
1][ySekarang] == false) {

     arah = 2;

   } else if (jarak_blkg > 8 && yTujuan < ySekarang  &&
lapangan[xSekarang][ySekarang - 1] == false) {

     arah = 3;

   } else {

     //jika tidak menemukan jalan secara normal

     if (jarak_depan > 8 && lapangan[xSekarang][ySekarang + 1] == false) {

       arah = 0;

     } else if (jarak_kn > 8 && lapangan[xSekarang + 1][ySekarang] == false) {

       arah = 1;

     } else if (jarak_kr > 8  && lapangan[xSekarang - 1][ySekarang] == false) {

       arah = 2;

     } else if (jarak_blkg > 8  && lapangan[xSekarang][ySekarang - 1] == false) {

       arah = 3;

     } else {

       if (jarak_depan > 8) {

         arah = 0;

       } else if (jarak_kn > 8) {

         arah = 1;

       } else if (jarak_kr > 8) {

         arah = 2;

       } else if (jarak_blkg > 8) {

         arah = 3;

       }
```

```
  }
}



if (arah == 0 ) {
  jumlah_y++;
  motor_kiri(Speed + PID);
  motor_kanan(Speed - PID);
  motor_depan(0);
  motor_belakang(0);

} else if (arah == 1) {

  jumlah_x++;
  motor_kiri(0);
  motor_kanan(0);
  motor_depan(Speed + PID);
  motor_belakang(Speed - PID);

} else if (arah == 2) {
  jumlah_x--;
  motor_kiri(0);
  motor_kanan(0);
  motor_depan(-Speed + PID);
  motor_belakang(-Speed - PID);
} else {
  jumlah_y--;
```

```
  motor_kiri(-Speed + PID);
  motor_kanan(-Speed - PID);
  motor_depan(0);
  motor_belakang(0);
 }

 if (jumlah_x <= rotasi_x) {
  xSekarang = 0;
 } else if (jumlah_x > rotasi_x && jumlah_x <= (2 * rotasi_x)) {
  xSekarang = 1 ;
 } else if (jumlah_x > (2 * rotasi_x) ) {
  xSekarang = 2 ;
 }

 if (jumlah_y <= rotasi_y) {
  ySekarang = 0;
 } else if (jumlah_y > rotasi_y && jumlah_y <= (2 * rotasi_y)) {
  ySekarang = 1 ;
 } else if (jumlah_y > (2 * rotasi_y) ) {
  ySekarang = 2 ;
 }

 lapangan[xSekarang][ySekarang] = true;
 lcd.setCursor(0, 1);
 lcd.print("PID="); lcd.print(PID); lcd.print("   ");
  Serial.print("PID = "); Serial.print(PID); Serial.print("\t");
Serial.print(xSekarang); Serial.print("\t"); Serial.print(xTujuan); Serial.print("\t");
Serial.print(jumlah_x);
```

```
    Serial.print("Arah = "); Serial.print(arah); Serial.print("\t");
Serial.print(ySekarang); Serial.print("\t"); Serial.print(yTujuan); Serial.print("\t");
Serial.print(jumlah_y);


    if (xSekarang == xTujuan && ySekarang == yTujuan) {
      motor_kiri(0);
      motor_kanan(0);
      motor_depan(0);
      motor_belakang(0);
      lcd.clear();
      lcd.print("Sampai Di lokasi kance");
      delay(1000);
      break;
    }
    delay(10);
  }
  delay(1000);


  henti();
  tampil_menu();
}


void henti() {


  motor_kiri(0);
  motor_kanan(0);
  motor_depan(0);
  motor_belakang(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("SimpanOK");
}

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_kiri(int v) {

  if (v == 0) {
    digitalWrite(kiri1, 1);
    digitalWrite(kiri2, 1);

  } else if (v > 0) {
    if (v > 255)v = 255;
```

```
    digitalWrite(kiri1, 1);
    digitalWrite(kiri2, 0);
    analogWrite(vkiri, v);
  } else {
    v = abs(v);
    if (v > 255)v = 255;
    digitalWrite(kiri1, 0);
    digitalWrite(kiri2, 1);
    analogWrite(vkiri, v);
  }

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

}



void motor_kanan(int v) {

  if (v == 0) {
    digitalWrite(kanan1, 1);
    digitalWrite(kanan2, 1);

  } else if (v > 0) {
    if (v > 255)v = 255;
    digitalWrite(kanan1, 1);
    digitalWrite(kanan2, 0);
    analogWrite(vkanan, v);
  } else {
```

```
    v = abs(v);
    if (v > 255)v = 255;
    digitalWrite(kanan1, 0);
    digitalWrite(kanan2, 1);
    analogWrite(vkanan, v);
  }

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

}

void motor_depan(int v) {

  if (v == 0) {
    digitalWrite(depan1, 1);
    digitalWrite(depan2, 1);

  } else if (v > 0) {
    if (v > 255)v = 255;
    digitalWrite(depan1, 1);
    digitalWrite(depan2, 0);
    analogWrite(vdepan, v);
  } else {
    v = abs(v);
    if (v > 255)v = 255;
    digitalWrite(depan1, 0);
    digitalWrite(depan2, 1);
    analogWrite(vdepan, v);
```

```
  }

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

}

void motor_belakang(int v) {

  if (v == 0) {
    digitalWrite(belakang1, 1);
    digitalWrite(belakang2, 1);

  } else if (v > 0) {
    if (v > 255)v = 255;
    digitalWrite(belakang1, 1);
    digitalWrite(belakang2, 0);
    analogWrite(vbelakang, v);
  } else {
    v = abs(v);
    if (v > 255)v = 255;
    digitalWrite(belakang1, 0);
    digitalWrite(belakang2, 1);
    analogWrite(vbelakang, v);
  }

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

}
```

```
void baca_sensor() {
  jarak_depan = depan.getDistanceCentimeter();
  jarak_skndepan = skndepan.getDistanceCentimeter();
  jarak_skrdepan = skrdepan.getDistanceCentimeter();
  jarak_kr = kr.getDistanceCentimeter();
  jarak_kn = kn.getDistanceCentimeter();
  jarak_sknblkg = skndepan.getDistanceCentimeter();
  jarak_skrblkg = skrblkg.getDistanceCentimeter();
  jarak_blkg = blkg.getDistanceCentimeter();

  lcd.setCursor(2, 0);
  lcd.print(jarak_depan);
  lcd.print ("  ");

  lcd.setCursor(2, 1);
  lcd.print(jarak_skndepan);
  lcd.print ("  ");

  lcd.setCursor(5, 0);
  lcd.print(jarak_skrdepan);
  lcd.print ("  ");

  lcd.setCursor(5, 1);
  lcd.print(jarak_kr);
  lcd.print ("  ");

  lcd.setCursor(8, 0);
```

```
  lcd.print(jarak_kn);
  lcd.print ("  ");

  lcd.setCursor(8, 1);
  lcd.print(jarak_sknblkg);
  lcd.print ("  ");

  lcd.setCursor(11, 0);
  lcd.print(jarak_skrblkg);
  lcd.print ("  ");

  lcd.setCursor(11, 1);
  lcd.print(jarak_blkg);
  lcd.print ("  ");
  //delay(3000); //make it readable
}

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 serialEvent() {
  while (bt.available()) {
    // get the new byte:
    char inChar = (char)bt.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag
    // so the main loop can do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}

void Setting_X() {
  lcd.setCursor(0, 0);
  lcd.print("               ");
```

```
lcd.setCursor(0, 0);
lcd.print(posX);

while (digitalRead(btn_Ok) == 1) {
  if (digitalRead(btn_Up) == 0) {
    delay(1000);
    posX = posX + 1;
    if (posX > 2)posX = 2;
    lcd.setCursor(0, 0);
    lcd.print(posX);
    lcd.print("  ");
  }
  if (digitalRead(btn_Down) == 0) {
    delay(1000);
    posX = posX - 1;
    if (posX < 0)posX = 0;
    lcd.setCursor(0, 0);
    lcd.print(posX);
    lcd.print("  ");
  }
}
delay (1000);
lcd.setCursor(0, 0);
lcd.print("Set X Selesai");
delay (1000);


tampil_menu();
```

```
}

void Setting_Y() {
  lcd.setCursor(0, 0);
  lcd.print("                ");
  lcd.setCursor(0, 0);
  lcd.print(posY);

  while (digitalRead(btn_Ok) == 1) {
    if (digitalRead(btn_Up) == 0) {
      delay(1000);
      posY = posY + 1;
      if (posY > 2)posY = 2;
      lcd.setCursor(0, 0);
      lcd.print(posY);
      lcd.print("  ");
    }
    if (digitalRead(btn_Down) == 0) {
      delay(1000);
      posY = posY - 1;
      if (posY < 0)posY = 0;
      lcd.setCursor(0, 0);
      lcd.print(posY);
      lcd.print("  ");

    }
  }
  delay (1000);
```

```
  lcd.setCursor(0, 0);
  lcd.print("Set Y Selesai");
  delay (1000);


  tampil_menu();
}


void random_move() {

  bt.print("Inputkan nilai x dan y (x,y)");
  while (!stringComplete) {
    serialEvent(); //call the function
    bt.print(".");
  }
  bt.println(inputString);
  // print the string when a newline arrives:
  Serial.println(inputString);
  xTujuan = getValue(inputString, ',', 0).toInt();
  yTujuan = getValue(inputString, ',', 1).toInt();
  int ySekarang = 1;
  int xSekarang = 1;
  long jumlah_x = 0;
  long jumlah_y = 0;
  bool lapangan[3][3];
  lapangan[0][0] = true;
  lapangan[0][1] = false;
```

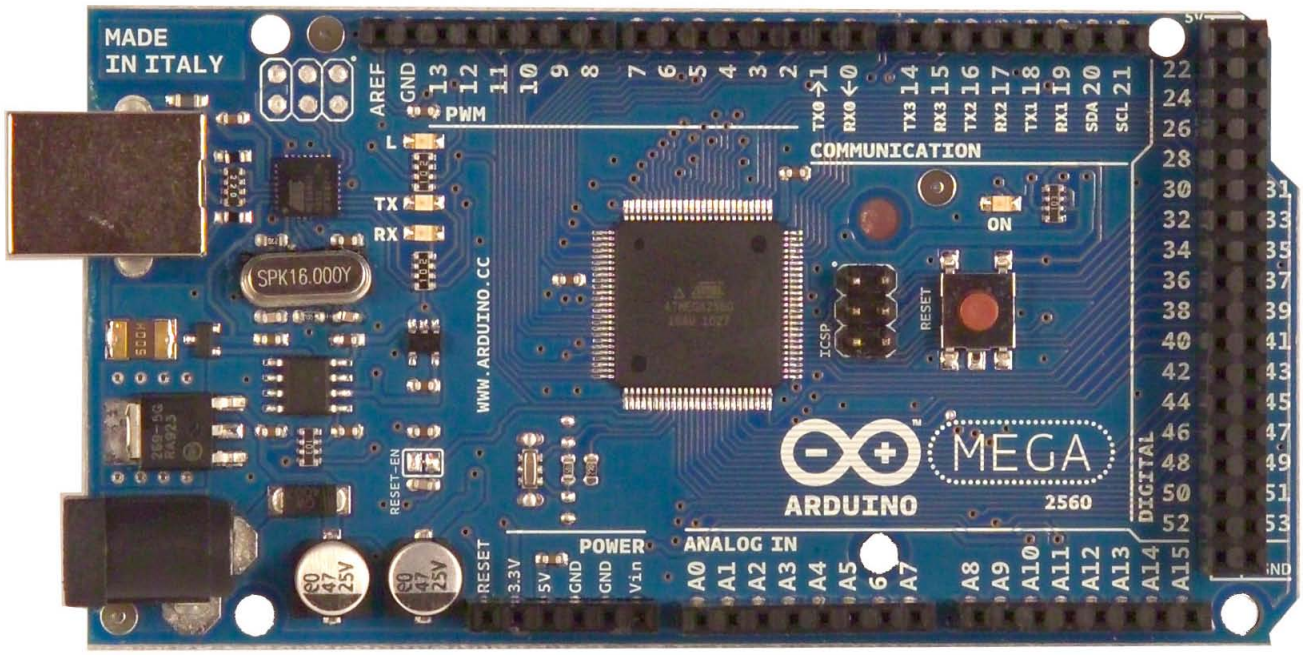
```
lapangan[0][2] = false;

lapangan[1][0] = false;
lapangan[1][1] = false;
lapangan[1][2] = false;

lapangan[2][0] = true;
lapangan[2][1] = false;
lapangan[2][2] = false;



inputString = "";
stringComplete = false;
lcd.setCursor(0, 0);
lcd.print("ROBOT MOVE    ");
delay(2000);
while (digitalRead(btn_Ok) == 1) {
  baca_sensor();
  baca_kompas();
  Serial.print(headingDegrees); Serial.print("  v   "); Serial.println(Sp);
  error = headingDegrees - Sp;
  bt.print(headingDegrees); bt.print(" ");
  bt.println(error);
  jumlah_error += (0.001 * error);
  selisih_error = error - error_sebelumnya;
  error_sebelumnya = error;

  P = Kp * error;
```

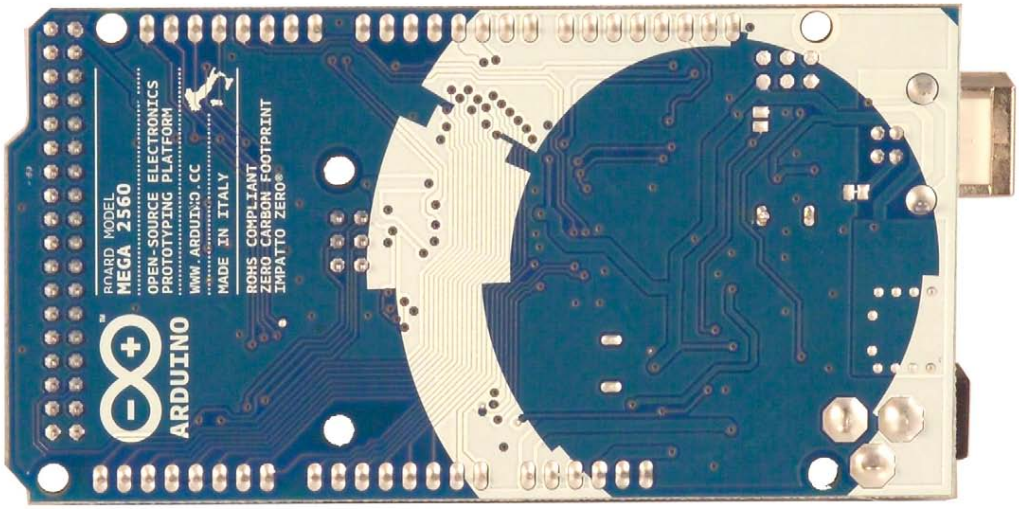
```
    D = Kd * selisih_error;
    I = Ki * jumlah_error;

    PID = P + I + D;

    Serial.print(error); Serial.print(" ");
    Serial.println(PID);



    //kalau depan bisa jalan dan tujuan depan > 1 maka jalan depan terlebih dahulu
    //jika tidak bisa maka kekanan terlebih dahulu
    if (jarak_depan > 50 && yTujuan > ySekarang &&
lapangan[xSekarang][ySekarang + 1] == false) {
      arah = 0;
    } else if (jarak_kn > 50 && xTujuan > xSekarang && lapangan[xSekarang +
1][ySekarang] == false) {
      arah = 1;
    } else if (jarak_kr > 50 && xTujuan < xSekarang && lapangan[xSekarang -
1][ySekarang] == false) {
      arah = 2;
    } else if (jarak_blkg > 50 && yTujuan < ySekarang  &&
lapangan[xSekarang][ySekarang - 1] == false) {
      arah = 3;
    }



    if (arah == 0 ) {
      jumlah_y++;
      motor_kiri(Speed + PID);
      motor_kanan(Speed - PID);
```

```
  motor_depan(0);
  motor_belakang(0);

} else if (arah == 1) {

  jumlah_x++;
  motor_kiri(0);
  motor_kanan(0);
  motor_depan(Speed + PID);
  motor_belakang(Speed - PID);

} else if (arah == 2) {
  jumlah_x--;
  motor_kiri(0);
  motor_kanan(0);
  motor_depan(-Speed + PID);
  motor_belakang(-Speed - PID);
} else {
  jumlah_y--;
  motor_kiri(-Speed + PID);
  motor_kanan(-Speed - PID);
  motor_depan(0);
  motor_belakang(0);
}

if (jumlah_x <= 13) {
  xSekarang = 1;
} else if (jumlah_x > 13 && jumlah_x <= 26) {
```

```
    xSekarang = 2 ;
   } else if (jumlah_x > 26 ) {
    xSekarang = 3 ;
   }

   if (jumlah_y <= 13) {
    ySekarang = 1;
   } else if (jumlah_y > 13 && jumlah_y <= 26) {
    ySekarang = 2 ;
   } else if (jumlah_y > 26 ) {
    ySekarang = 3 ;
   }

   lapangan[xSekarang][ySekarang] = true;
   lcd.setCursor(0, 1);
   lcd.print("PID="); lcd.print(PID); lcd.print("    ");
   delay(10);
  }


  henti();

  delay(2000);
  tampil_menu();
}
```

# Arduino Mega 2560




## 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.

## Schematic & Reference Design

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 |

## Power

The Arduino Mega 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 using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, 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.

- **$I^2C$: 20 (SDA) and 21 (SCL).** Support $I^2C$ (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the $I^2C$ pins on the Duemilanove or Diecimila.

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 Mega2560'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. For SPI communication, use the SPI library.

## Programming

The Arduino Mega 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.

The ATmega8U2 firmware source code is available in the Arduino repository. The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

## 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 Mega2560 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 Mega2560 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

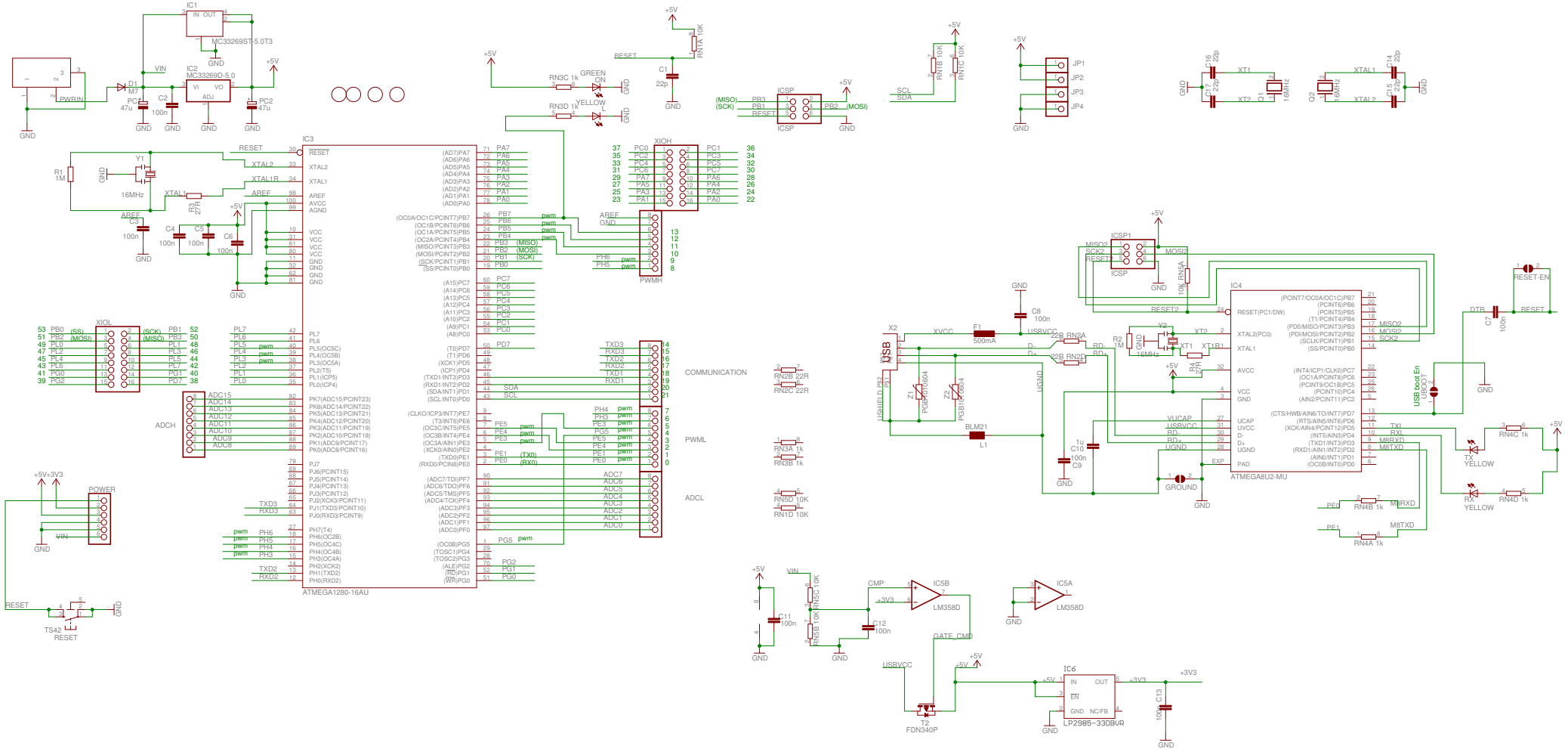
The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega2560 is designed to be compatible with most shields designed for the Uno, 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 Mega2560 and Duemilanove / Diecimila. *Please note that $I^2C$ is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).*

# Arduino™ Mega 2560 Reference Design

Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS". Arduino 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

Arduino 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." Arduino 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.



# GP2Y0A02YK0F

## Distance Measuring Sensor Unit
## Measuring distance: 20 to 150 cm
## Analog output type


## ■Description

**GP2Y0A02YK0F** is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector) , IRED (infrared emitting diode) and signal processing circuit.
The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method.
This device outputs the voltage corresponding to the detection distance. So this sensor can also be used as a proximity sensor.

## ■Features

1. Distance measuring range : 20 to 150 cm
2. Analog output type
3. Package size : 29.5×13×21.6 mm
4. Consumption current : Typ. 33 mA
5. Supply voltage : 4.5 to 5.5 V

## ■Agency approvals/Compliance

1. Compliant with RoHS directive (2002/95/EC)

## ■Applications

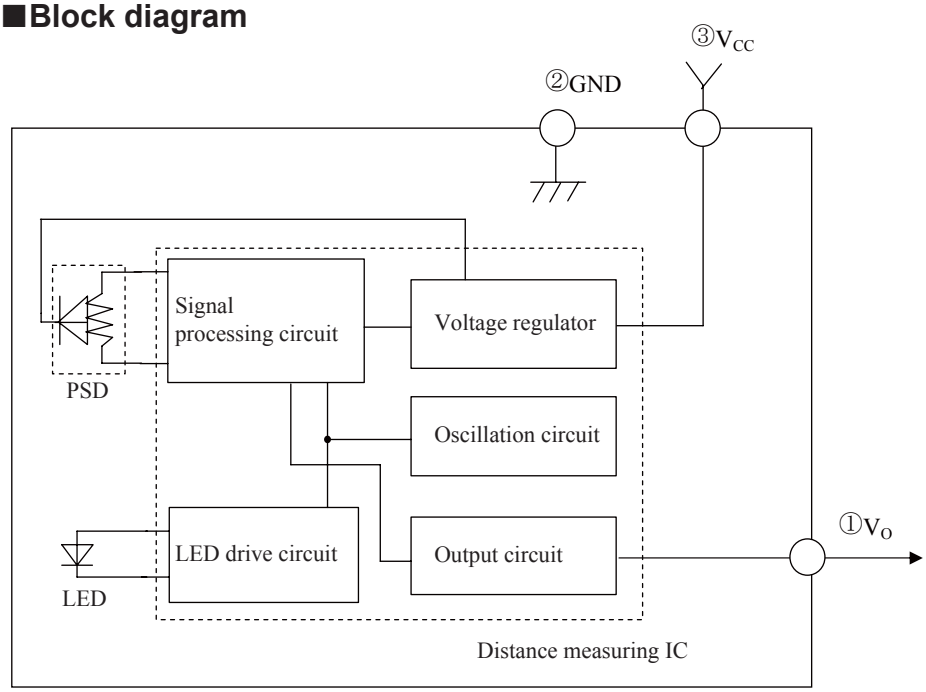
1. Touch-less switch
   (Sanitary equipment, Control of illumination, etc. )
2. Sensor for energy saving
   (ATM, Copier, Vending machine, Laptop computer, LCD monitor)
3. Amusement equipment
   (Robot, Arcade game machine)

Notice The content of data sheet is subject to change without prior notice.
In the absence of confirmation by device specification sheets, SHARP takes no responsibility for any defects that may occur in equipment using any SHARP devices shown in catalogs, data books, etc. Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device.



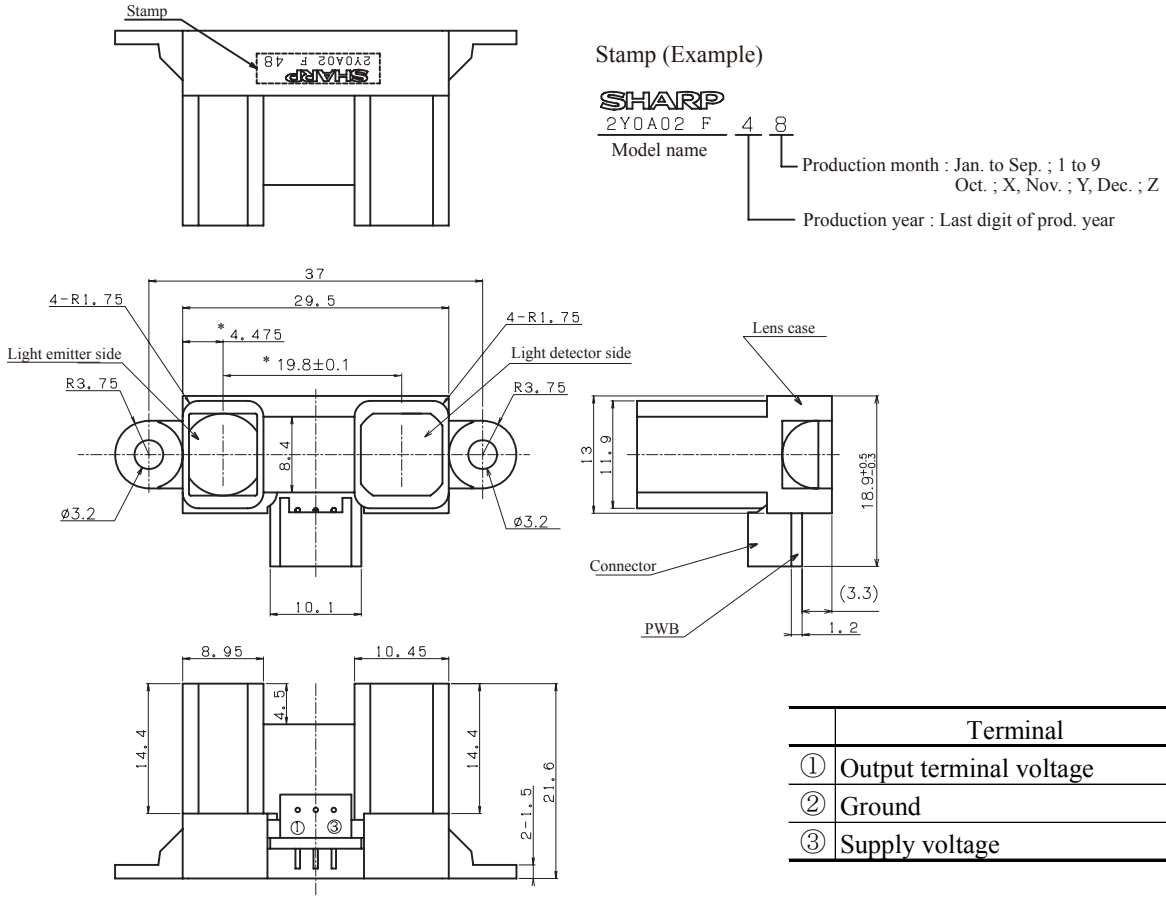
Sheet No.: E4-A00101EN
Date Dec.01.2006
©SHARP Corporation

## ■Block diagram




## ■Outline Dimensions

(Unit : mm)




| | Terminal | Symbol |
|---|---|---|
| ① | Output terminal voltage | $V_O$ |
| ② | Ground | GND |
| ③ | Supply voltage | $V_{CC}$ |

Note 1. Unspecified tolerances shall be ± 0.3 mm.
Note 2. The connector is made by J.S.T.TRADING COMPANY,LTD. and its part number is S3B-PH.
Note 3. The dimensions in parenthesis are shown for reference.
Note 4. The dimension marked by "*" show a distance from/to the center of an internal optical slit.

Product mass : approx. 4.8g

## ■Absolute Maximum Ratings

($T_a$=25℃,$V_{CC}$=5V)

| Parameter | Symbol | Rating | Unit |
|---|---|---|---|
| Supply voltage | $V_{CC}$ | -0.3 to +7 | V |
| Output terminal voltage | $V_O$ | -0.3 to $V_{CC}$+0.3 | V |
| Operating temperature | $T_{opr}$ | -10 to +60 | ℃ |
| Storage temperature | $T_{stg}$ | -40 to +70 | ℃ |

## ■Electro-optical Characteristics

($T_a$=25℃,$V_{CC}$=5V)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Average supply current | $I_{CC}$ | L=150cm (Note 1) | — | 33 | 50 | mA |
| Measuring distance range | ΔL | (Note 1) | 20 | — | 150 | cm |
| Output voltage | $V_O$ | L=150cm (Note 1) | 0.25 | 0.4 | 0.55 | V |
| Output voltage differential | $\Delta V_O$ | Output voltage difference between L=20cm and L=150cm (Note 1) | 1.8 | 2.05 | 2.3 | V |

* L : Distance to reflective object

Note 1 : Using reflective object : White paper (Made by Kodak Co., Ltd. gray cards R-27・white face, reflectance; 90%)

## ■Recommended operating conditions

| Parameter | Symbol | Conditions | Rating | Unit |
|---|---|---|---|---|
| Supply voltage | $V_{CC}$ | | 4.5 to 5.5 | V |

**Fig. 1 Timing chart**

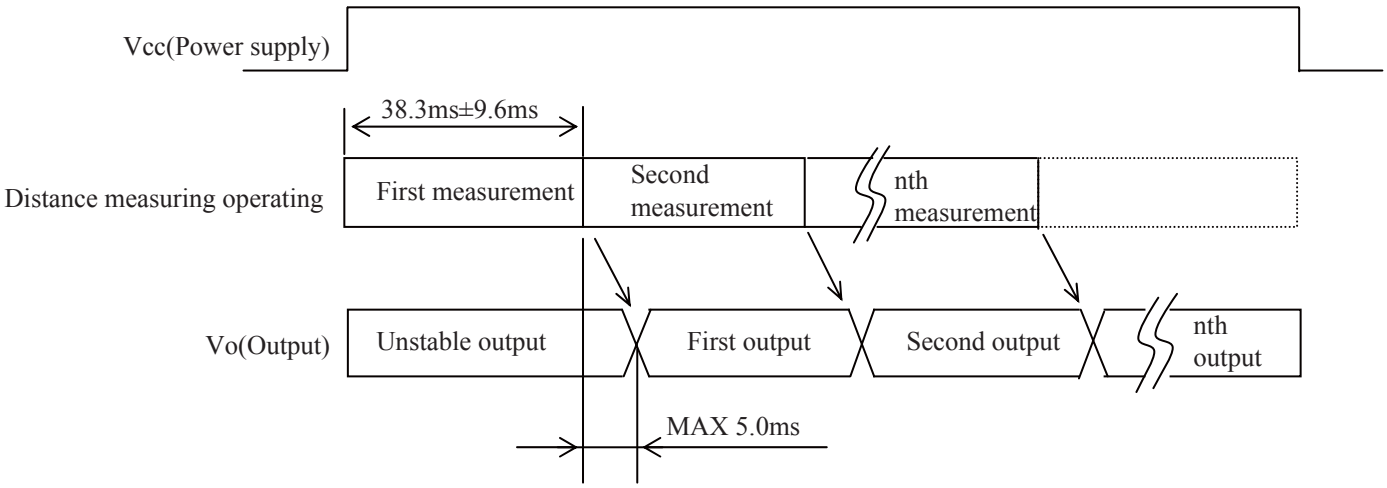
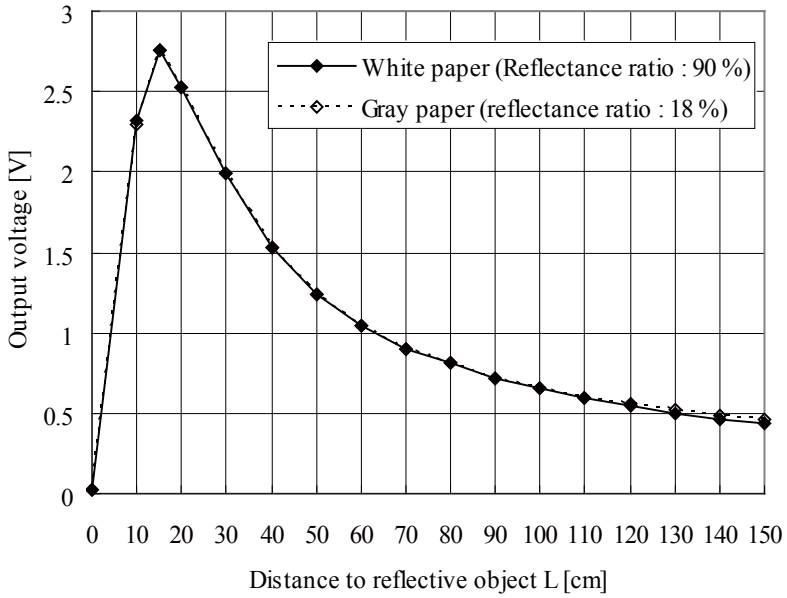
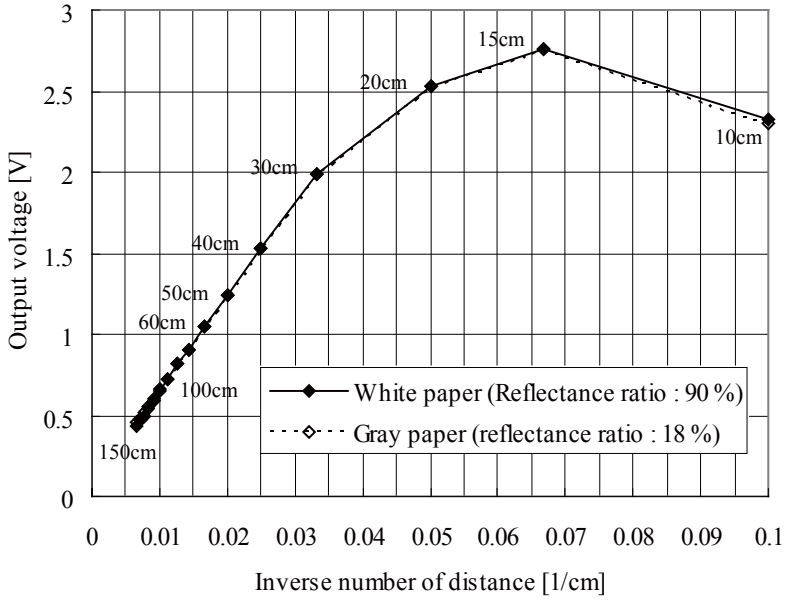


**Fig. 2 Example of distance measuring characteristics (output)**



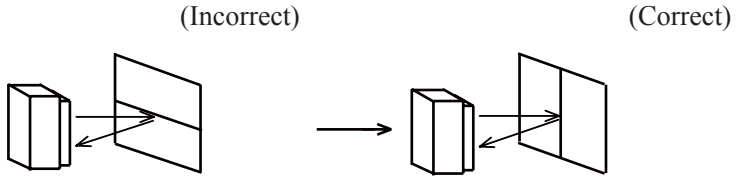
## ■Notes

### ●Advice for the optics

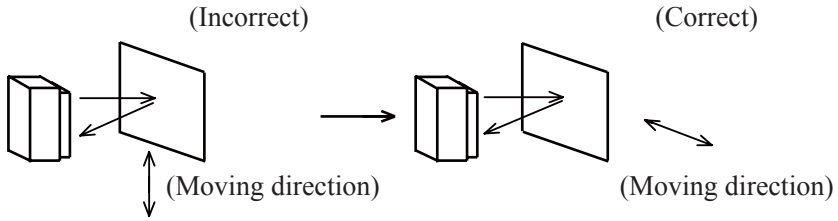
• The lens of this device needs to be kept clean. There are cases that dust, water or oil and so on deteriorate the characteristics of this device. Please consider in actual application.
• Please don't do washing. Washing may deteriorate the characteristics of optical system and so on. Please confirm resistance to chemicals under the actual usage since this product has not been designed against washing.

### ●Advice for the characteristics

• In case that an optical filter is set in front of the emitter and detector portion, the optical filter which has the most efficient transmittance at the emitting wavelength range of LED for this product (λ = 850 ± 70nm), shall be recommended to use. Both faces of the filter should be mirror polishing. Also, as there are cases that the characteristics may not be satisfied according to the distance between the protection cover and this product or the thickness of the protection cover, please use this product after confirming the operation sufficiently in actual application.
• In case that there is an object near to emitter side of the sensor between sensor and a detecting object, please use this device after confirming sufficiently that the characteristics of this sensor do not change by the object.
• When the detector is exposed to the direct light from the sun, tungsten lamp and so on, there are cases that it can not measure the distance exactly. Please consider the design that the detector is not exposed to the direct light from such light source.
• Distance to a mirror reflector can not be sometimes measured exactly.
In case of changing the mounting angle of this product, it may measure the distance exactly.
• In case that reflective object has boundary line which material or color etc. are excessively different, in order to decrease deviation of measuring distance, it shall be recommended to set the sensor that the direction of boundary line and the line between emitter center and detector center are in parallel.




• In order to decrease deviation of measuring distance by moving direction of the reflective object, it shall be recommended to set the sensor that the moving direction of the object and the line between emitter center and detector center are vertical.




### ●Advice for the power supply

• In order to stabilize power supply line, we recommend to insert a by-pass capacitor of 10μF or more between Vcc and GND near this product.

### ●Notes on handling

• There are some possibilities that the internal components in the sensor may be exposed to the excessive mechanical stress. Please be careful not to cause any excessive pressure on the sensor package and also on the PCB while assembling this product.

## ●Presence of ODC etc.

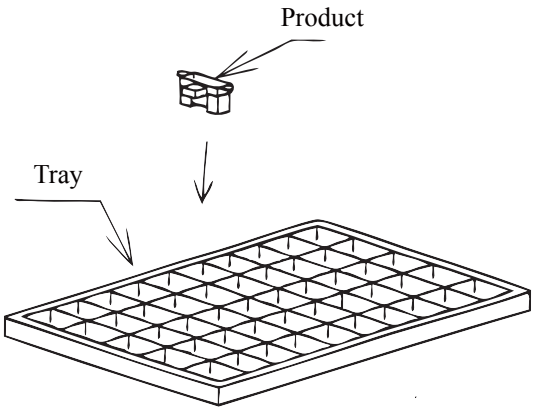
This product shall not contain the following materials.
And they are not used in the production process for this product.
Regulation substances : CFCs, Halon, Carbon tetrachloride, 1.1.1-Trichloroethane (Methylchloroform)

Specific brominated flame retardants such as the PBB and PBDE are not used in this product at all.

This product shall not contain the following materials banned in the RoHS Directive (2002/95/EC).

- Lead, Mercury, Cadmium, Hexavalent chromium, Polybrominated biphenyls (PBB), Polybrominated diphenyl ethers (PBDE).

## ■Package specification




MAX. 50 pieces per tray

## ■Important Notices

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

· Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device. SHARP reserves the right to make changes in the specifications, characteristics, data, materials, structure, and other contents described herein at any time without notice in order to improve design or reliability. Manufacturing locations are also subject to change without notice.

· Observe the following points when using any devices in this publication. SHARP takes no responsibility for damage caused by improper use of the devices which does not meet the conditions and absolute maximum ratings to be used specified in the relevant specification sheet nor meet the following conditions:

(i) The devices in this publication are designed for use in general electronic equipment designs such as:
- --- Personal computers
- --- Office automation equipment
- --- Telecommunication equipment [terminal]
- --- Test and measurement equipment
- --- Industrial control
- --- Audio visual equipment
- --- Consumer electronics

(ii) Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when SHARP devices are used for or in connection with equipment that requires higher reliability such as:
- --- Transportation control and safety equipment (i.e., aircraft, trains, automobiles, etc.)
- --- Traffic signals
- --- Gas leakage sensor breakers
- --- Alarm equipment
- --- Various safety devices, etc.

(iii) SHARP devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety such as:
- --- Space applications
- --- Telecommunication equipment [trunk lines]
- --- Nuclear power control equipment
- --- Medical and other life support equipment (e.g., scuba).

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

· This publication is the proprietary product of SHARP and is copyrighted, with all rights reserved. Under the copyright laws, no part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of SHARP. Express written permission is also required before any use of this publication may be made by a third party.

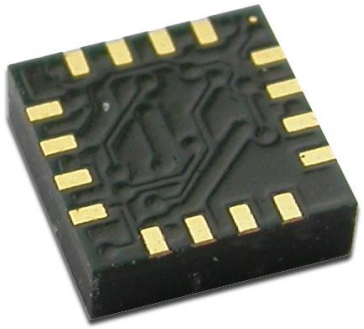
· Contact and consult with a SHARP representative if there are any questions about the contents of this publication.

# 3-Axis Digital Compass IC HMC5883L

Honeywell

*Advanced Information*

The Honeywell HMC5883L is a surface-mount, multi-chip module designed for low-field magnetic sensing with a digital interface for applications such as low-cost compassing and magnetometry. The HMC5883L includes our state-of-the-art, high-resolution HMC118X series magneto-resistive sensors plus an ASIC containing amplification, automatic degaussing strap drivers, offset cancellation, and a 12-bit ADC that enables 1° to 2° compass heading accuracy. The $I^2C$ serial bus allows for easy interface. The HMC5883L is a 3.0x3.0x0.9mm surface mount 16-pin leadless chip carrier (LCC). Applications for the HMC5883L include Mobile Phones, Netbooks, Consumer Electronics, Auto Navigation Systems, and Personal Navigation Devices.



The HMC5883L utilizes Honeywell's Anisotropic Magnetoresistive (AMR) technology that provides advantages over other magnetic sensor technologies. These anisotropic, directional sensors feature precision in-axis sensitivity and linearity. These sensors' solid-state construction with very low cross-axis sensitivity is designed to measure both the direction and the magnitude of Earth's magnetic fields, from milli-gauss to 8 gauss. Honeywell's Magnetic Sensors are among the most sensitive and reliable low-field sensors in the industry.

| FEATURES | BENEFITS |
|---|---|
| 3-Axis Magnetoresistive Sensors and ASIC in a 3.0x3.0x0.9mm LCC Surface Mount Package | Small Size for Highly Integrated Products. Just Add a Micro-Controller Interface, Plus Two External SMT Capacitors Designed for High Volume, Cost Sensitive OEM Designs Easy to Assemble & Compatible with High Speed SMT Assembly |
| 12-Bit ADC Coupled with Low Noise AMR Sensors Achieves 2 milli-gauss Field Resolution in ±8 Gauss Fields | Enables 1° to 2° Degree Compass Heading Accuracy |
| Built-In Self Test | Enables Low-Cost Functionality Test after Assembly in Production |
| Low Voltage Operations (2.16 to 3.6V) and Low Power Consumption (100 µA) | Compatible for Battery Powered Applications |
| Built-In Strap Drive Circuits | Set/Reset and Offset Strap Drivers for Degaussing, Self Test, and Offset Compensation |
| $I^2C$ Digital Interface | Popular Two-Wire Serial Data Interface for Consumer Electronics |
| Lead Free Package Construction | RoHS Compliance |
| Wide Magnetic Field Range (+/-8 Oe) | Sensors Can Be Used in Strong Magnetic Field Environments with a 1° to 2° Degree Compass Heading Accuracy |
| Software and Algorithm Support Available | Compassing Heading, Hard Iron, Soft Iron, and Auto Calibration Libraries Available |
| Fast 160 Hz Maximum Output Rate | Enables Pedestrian Navigation and LBS Applications |

## SPECIFICATIONS (* Tested at 25°C except stated otherwise.)

| Characteristics | Conditions* | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| ***Power Supply*** | | | | | |
| Supply Voltage | VDD Referenced to AGND<br>VDDIO Referenced to DGND | 2.16<br>1.71 | 2.5<br>1.8 | 3.6<br>VDD+0.1 | Volts<br>Volts |
| Average Current Draw | Idle Mode<br>Measurement Mode (7.5 Hz ODR;<br>No measurement average, MA1:MA0 = 00)<br>VDD = 2.5V, VDDIO = 1.8V (Dual Supply)<br>VDD = VDDIO = 2.5V (Single Supply) | -<br>- | 2<br>100 | -<br>- | µA<br>µA |
| ***Performance*** | | | | | |
| Field Range | Full scale (FS) | -8 | | +8 | gauss |
| Mag Dynamic Range | 3-bit gain control | ±1 | | ±8 | gauss |
| Sensitivity (Gain) | VDD=3.0V, GN=0 to 7, 12-bit ADC | 230 | | 1370 | LSb/gauss |
| Digital Resolution | VDD=3.0V, GN=0 to 7, 1-LSb, 12-bit ADC | 0.73 | | 4.35 | milli-gauss |
| Noise Floor<br>(Field Resolution) | VDD=3.0V, GN=0, No measurement average, Standard Deviation 100 samples<br>(See typical performance graphs below) | | 2 | | milli-gauss |
| Linearity | ±2.0 gauss input range | | | 0.1 | ±% FS |
| Hysteresis | ±2.0 gauss input range | | ±25 | | ppm |
| Cross-Axis Sensitivity | Test Conditions: Cross field = 0.5 gauss, Happlied = ±3 gauss | | ±0.2% | | %FS/gauss |
| Output Rate (ODR) | Continuous Measurment Mode<br>Single Measurement Mode | 0.75 | | 75<br>160 | Hz<br>Hz |
| Measurement Period | From receiving command to data ready | | 6 | | ms |
| Turn-on Time | Ready for I2C commands<br>Analog Circuit Ready for Measurements | | 200<br>50 | | µs<br>ms |
| Gain Tolerance | All gain/dynamic range settings | | ±5 | | % |
| $I^2C$ Address | 8-bit read address<br>8-bit write address | | 0x3D<br>0x3C | | hex<br>hex |
| $I^2C$ Rate | Controlled by $I^2C$ Master | | | 400 | kHz |
| $I^2C$ Hysteresis | Hysteresis of Schmitt trigger inputs on SCL and SDA - Fall (VDDIO=1.8V)<br>Rise (VDDIO=1.8V) | | 0.2*VDDIO<br>0.8*VDDIO | | Volts<br>Volts |
| Self Test | X & Y Axes<br>Z Axis | | ±1.16<br>±1.08 | | gauss |
| | X & Y & Z Axes (GN=5) Positive Bias<br>X & Y & Z Axes (GN=5) Negative Bias | 243<br>-575 | | 575<br>-243 | LSb |
| Sensitivity Tempco | $T_A$ = -40 to 125°C, Uncompensated Output | | -0.3 | | %/°C |
| ***General*** | | | | | |
| ESD Voltage | Human Body Model (all pins)<br>Charged Device Model (all pins) | | | 2000<br>750 | Volts |
| Operating Temperature | Ambient | -30 | | 85 | °C |
| Storage Temperature | Ambient, unbiased | -40 | | 125 | °C |

| Characteristics | Conditions* | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Reflow Classification | MSL 3, 260 °C Peak Temperature | | | | |
| Package Size | Length and Width | 2.85 | 3.00 | 3.15 | mm |
| Package Height | | 0.8 | 0.9 | 1.0 | mm |
| Package Weight | | | 18 | | mg |

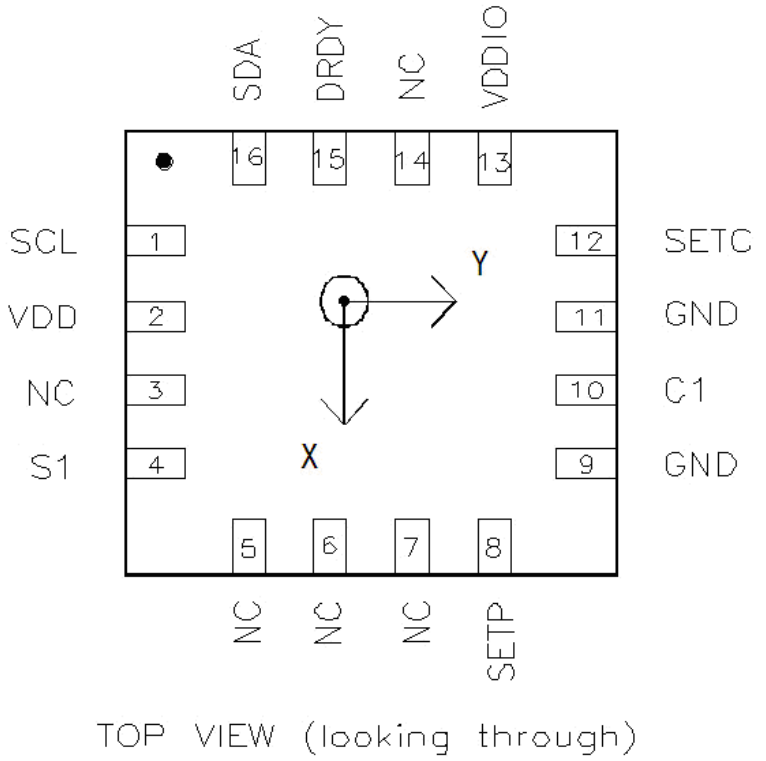
## Absolute Maximum Ratings (* Tested at 25°C except stated otherwise.)

| Characteristics | Min | Max | Units |
|---|---|---|---|
| Supply Voltage VDD | -0.3 | 4.8 | Volts |
| Supply Voltage VDDIO | -0.3 | 4.8 | Volts |

## PIN CONFIGURATIONS

| Pin | Name | Description |
|---|---|---|
| 1 | SCL | Serial Clock – $I^2C$ Master/Slave Clock |
| 2 | VDD | Power Supply (2.16V to 3.6V) |
| 3 | NC | Not to be Connected |
| 4 | S1 | Tie to VDDIO |
| 5 | NC | Not to be Connected |
| 6 | NC | Not to be Connected |
| 7 | NC | Not to be Connected |
| 8 | SETP | Set/Reset Strap Positive – S/R Capacitor (C2) Connection |
| 9 | GND | Supply Ground |
| 10 | C1 | Reservoir Capacitor (C1) Connection |
| 11 | GND | Supply Ground |
| 12 | SETC | S/R Capacitor (C2) Connection – Driver Side |
| 13 | VDDIO | IO Power Supply (1.71V to VDD) |
| 14 | NC | Not to be Connected |
| 15 | DRDY | Data Ready, Interrupt Pin. Internally pulled high. Optional connection. Low for 250 µsec when data is placed in the data output registers. |
| 16 | SDA | Serial Data – $I^2C$ Master/Slave Data |

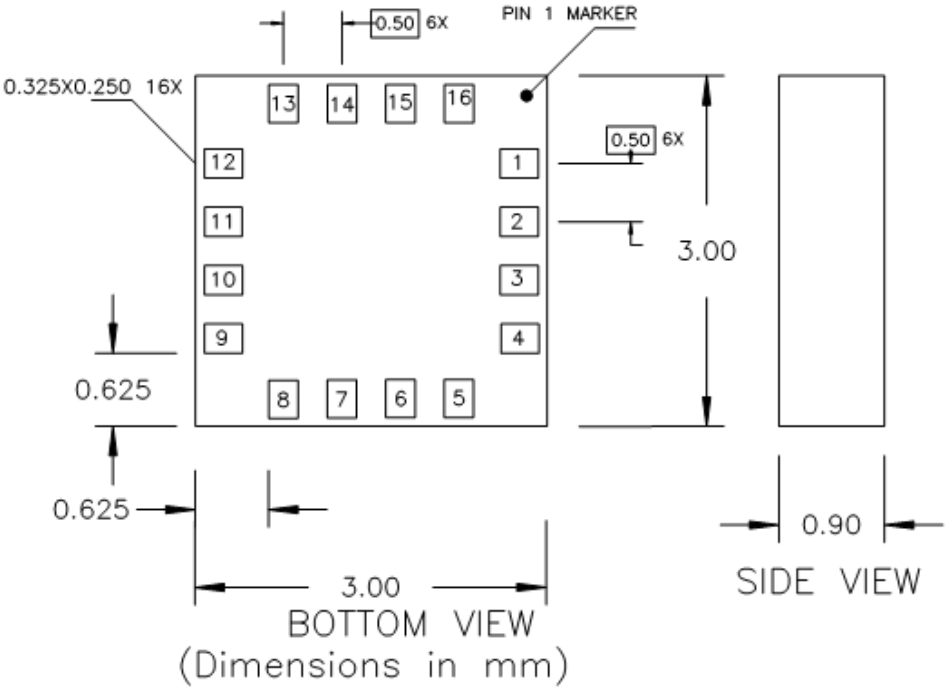
Table 1: Pin Configurations




Arrow indicates direction of magnetic field that generates a positive output reading in Normal Measurement configuration.

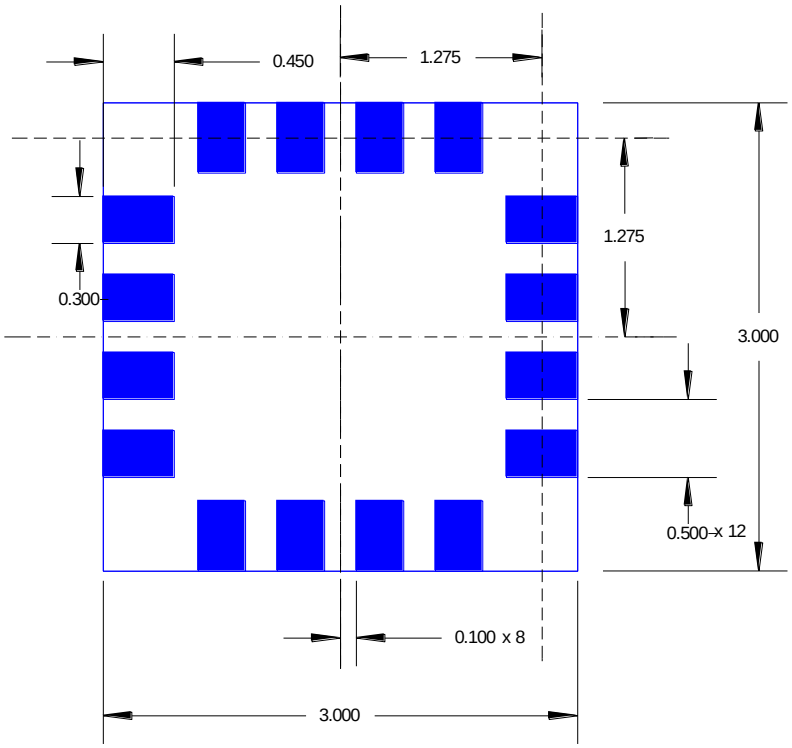
## PACKAGE OUTLINES

### PACKAGE DRAWING HMC5883L (16-PIN LPCC, dimensions in millimeters)




## MOUNTING CONSIDERATIONS

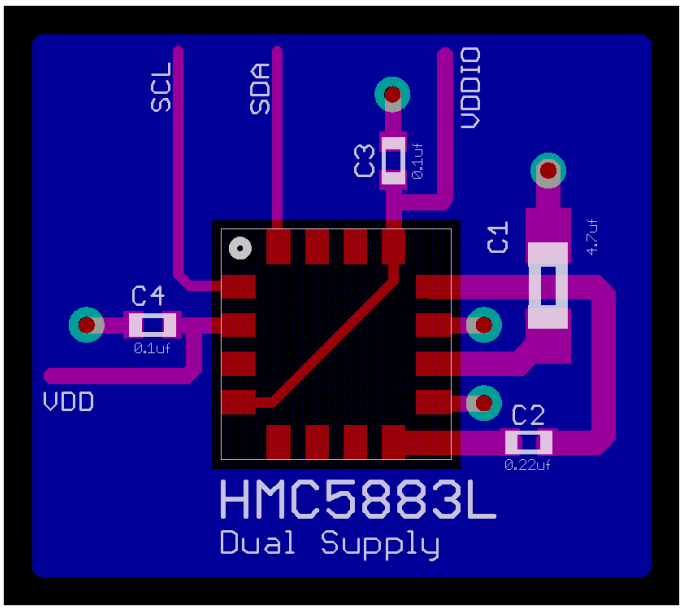
The following is the recommend printed circuit board (PCB) footprint for the HMC5883L.

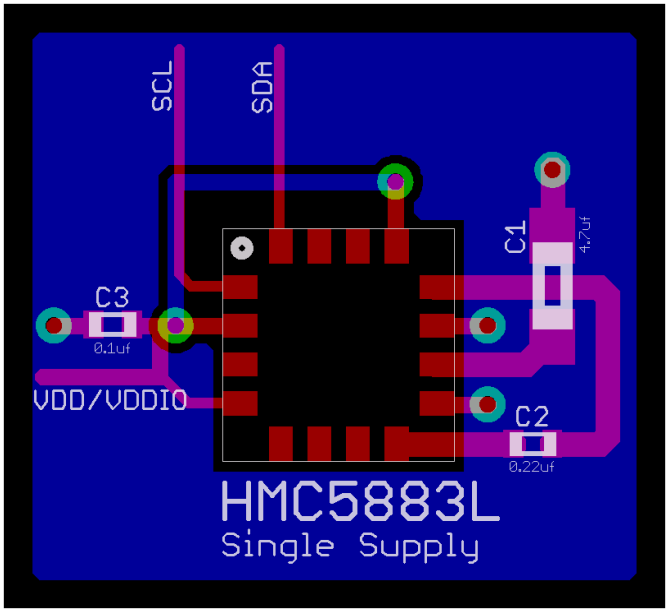



HMC5883 Land Pad Pattern
(All dimensions are in mm)

## LAYOUT CONSIDERATIONS

Besides keeping all components that may contain ferrous materials (nickel, etc.) away from the sensor on both sides of the PCB, it is also recommended that there is no conducting copper under/near the sensor in any of the PCB layers. See recommended layout below. Notice that the one trace under the sensor in the dual supply mode is not expected to carry active current since it is for pin 4 pull-up to VDDIO. Power and ground planes are removed under the sensor to minimize possible source of magnetic noise. For best results, use non-ferrous materials for all exposed copper coding.



### PCB Pad Definition and Traces

The HMC5883L is a fine pitch LCC package. Refer to previous figure for recommended PCB footprint for proper package centering. Size the traces between the HMC5883L and the external capacitors (C1 and C2) to handle the 1 ampere peak current pulses with low voltage drop on the traces.

### Stencil Design and Solder Paste

A 4 mil stencil and 100% paste coverage is recommended for the electrical contact pads.

### Reflow Assembly

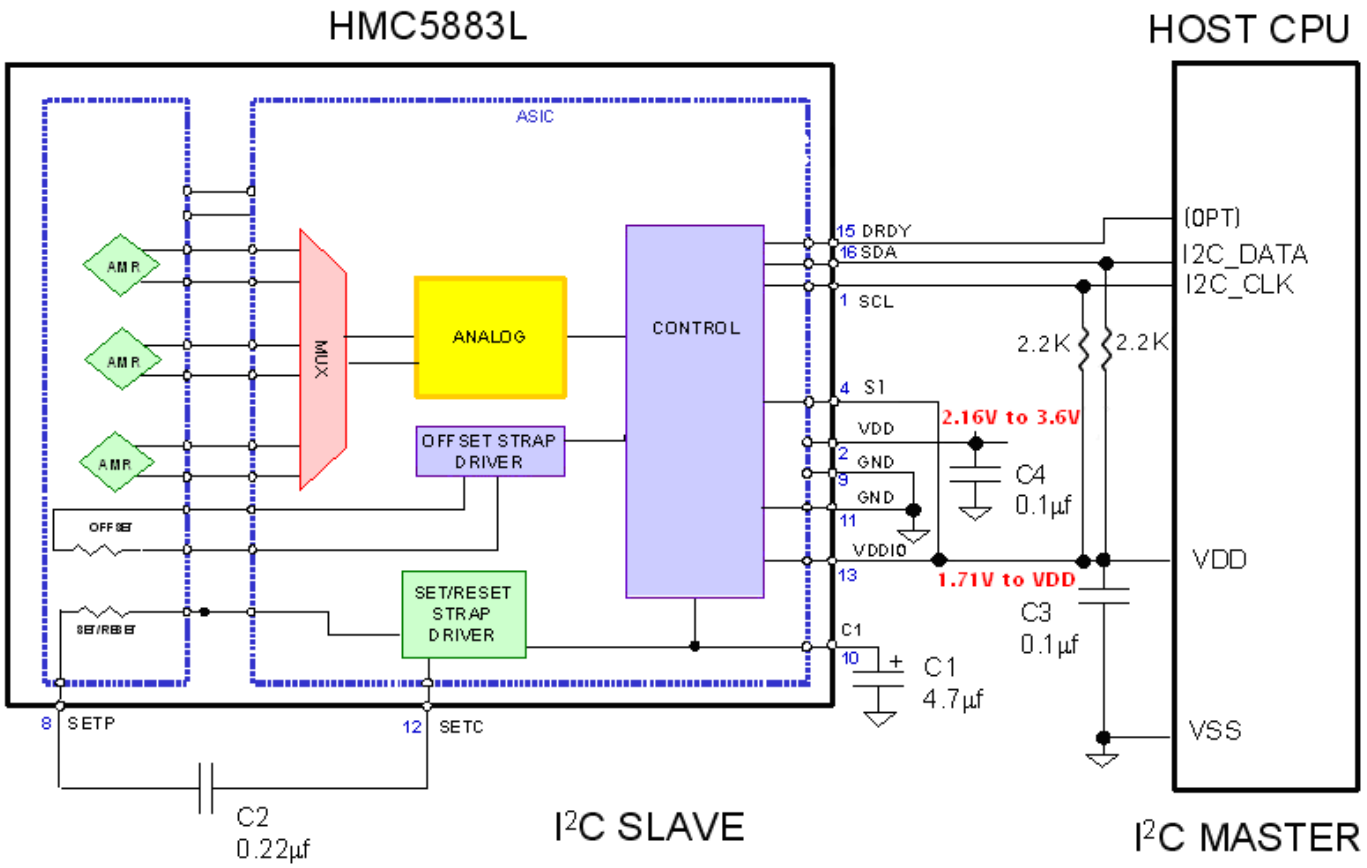
This device is classified as MSL 3 with 260°C peak reflow temperature. A baking process (125°C, 24 hrs) is required if device is not kept continuously in a dry (< 10% RH) environment before assembly. No special reflow profile is required for HMC5883L, which is compatible with lead eutectic and lead-free solder paste reflow profiles. Honeywell recommends adherence to solder paste manufacturer's guidelines. Hand soldering is not recommended. Built-in self test can be used to verify device functionalities after assembly.

### External Capacitors

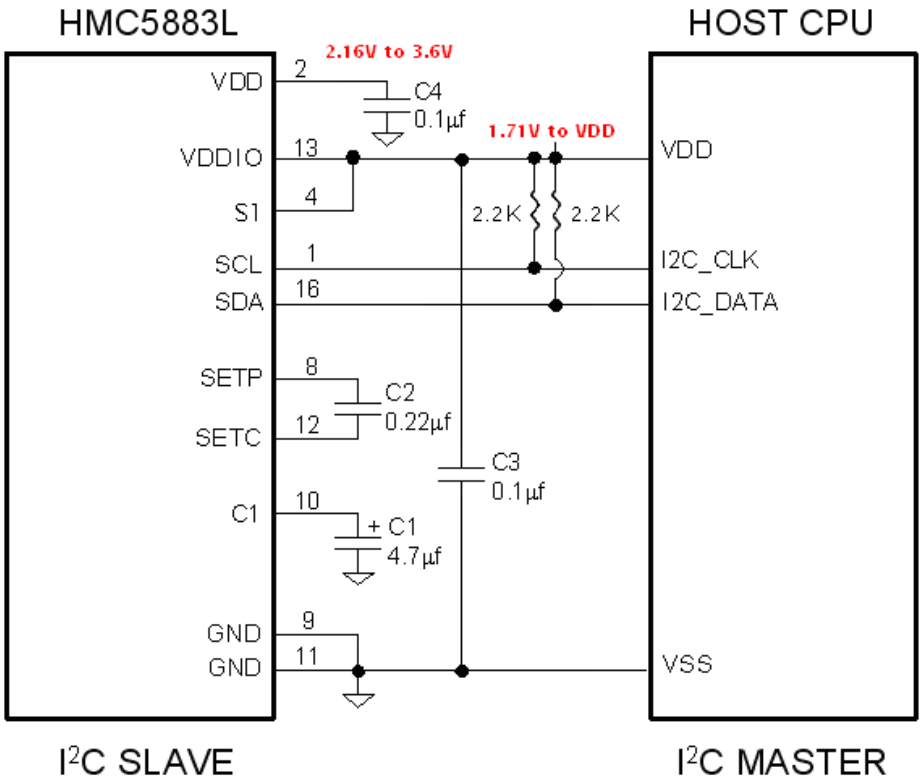
The two external capacitors should be ceramic type construction with low ESR characteristics. The exact ESR values are not critical but values less than 200 milli-ohms are recommended. Reservoir capacitor C1 is nominally 4.7 µF in capacitance, with the set/reset capacitor C2 nominally 0.22 µF in capacitance. Low ESR characteristics may not be in many small SMT ceramic capacitors (0402), so be prepared to up-size the capacitors to gain Low ESR characteristics.

## INTERNAL SCHEMATIC DIAGRAM

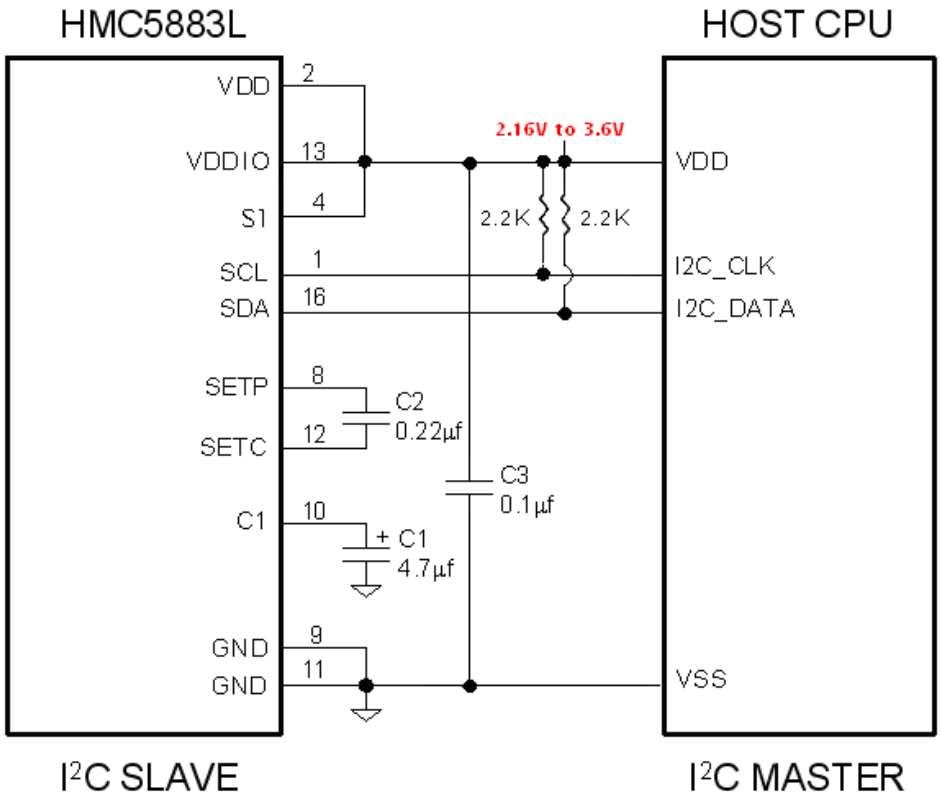
***HMC5883L***



## DUAL SUPPLY REFERENCE DESIGN



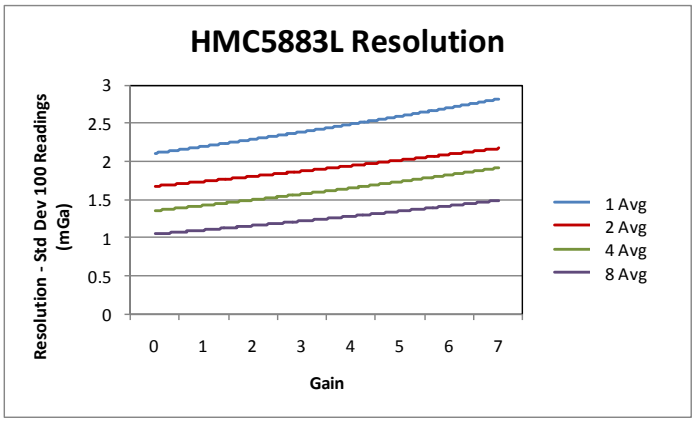

## SINGLE SUPPLY REFERENCE DESIGN

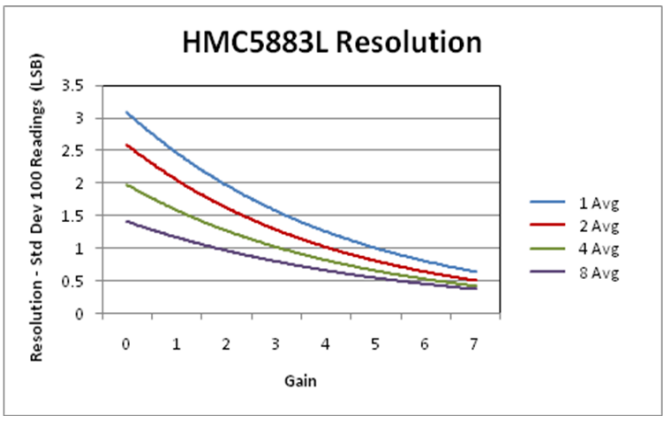


## PERFORMANCE

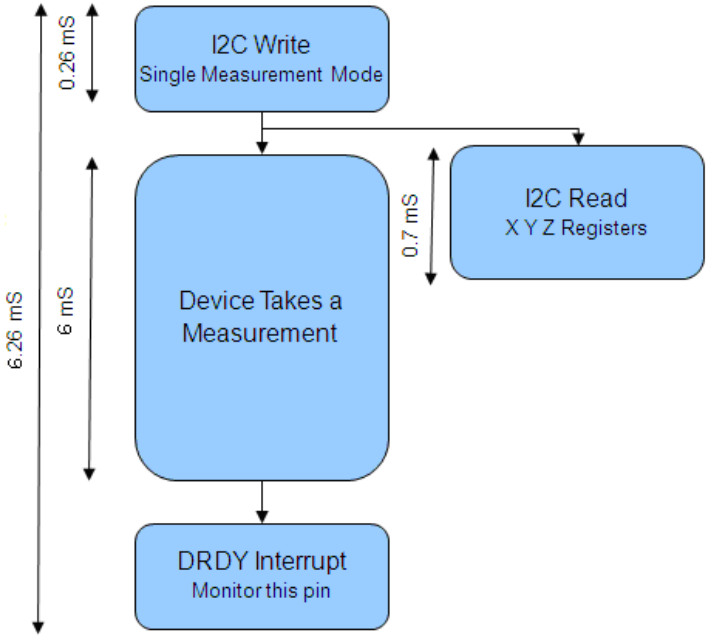
The following graph(s) highlight HMC5883L's performance.

### Typical Noise Floor (Field Resolution)




### Typical Measurement Period in Single-Measurement Mode




** Monitoring of the DRDY Interrupt pin is only required if maximum output rate is desired.*

## BASIC DEVICE OPERATION

### Anisotropic Magneto-Resistive Sensors

The Honeywell HMC5883L magnetoresistive sensor circuit is a trio of sensors and application specific support circuits to measure magnetic fields. With power supply applied, the sensor converts any incident magnetic field in the sensitive axis directions to a differential voltage output. The magnetoresistive sensors are made of a nickel-iron (Permalloy) thin-film and patterned as a resistive strip element. In the presence of a magnetic field, a change in the bridge resistive elements causes a corresponding change in voltage across the bridge outputs.

These resistive elements are aligned together to have a common sensitive axis (indicated by arrows in the pinout diagram) that will provide positive voltage change with magnetic fields increasing in the sensitive direction. Because the output is only proportional to the magnetic field component along its axis, additional sensor bridges are placed at orthogonal directions to permit accurate measurement of magnetic field in any orientation.

### Self Test

To check the HMC5883L for proper operation, a self test feature in incorporated in which the sensor is internally excited with a nominal magnetic field (in either positive or negative bias configuration). This field is then measured and reported. This function is enabled and the polarity is set by bits MS[n] in the configuration register A. An internal current source generates DC current (about 10 mA) from the VDD supply. This DC current is applied to the offset straps of the magneto-resistive sensor, which creates an artificial magnetic field bias on the sensor. The difference of this measurement and the measurement of the ambient field will be put in the data output register for each of the three axes. By using this built-in function, the manufacturer can quickly verify the sensor's full functionality after the assembly without additional test setup. The self test results can also be used to estimate/compensate the sensor's sensitivity drift due to temperature.

For each "self test measurement", the ASIC:

1. Sends a "Set" pulse
2. Takes one measurement (M1)
3. Sends the (~10 mA) offset current to generate the (~1.1 Gauss) offset field and takes another measurement (M2)
4. Puts the difference of the two measurements in sensor's data output register:

   **Output = [M2 – M1]** (i.e. output = offset field only)

See SELF TEST OPERATION section later in this datasheet for additional details.

### Power Management

This device has two different domains of power supply. The first one is VDD that is the power supply for internal operations and the second one is VDDIO that is dedicated to IO interface. It is possible to work with VDDIO equal to VDD; Single Supply mode, or with VDDIO lower than VDD allowing HMC5883L to be compatible with other devices on board.

### $I^2C$ Interface

Control of this device is carried out via the $I^2C$ bus. This device will be connected to this bus as a slave device under the control of a master device, such as the processor.

This device is compliant with *$I^2C$-Bus Specification*, document number: 9398 393 40011. As an $I^2C$ compatible device, this device has a 7-bit serial address and supports $I^2C$ protocols. This device supports standard and fast modes, 100kHz and 400kHz, respectively, but does not support the high speed mode (Hs). External pull-up resistors are required to support these standard and fast speed modes.

Activities required by the master (register read and write) have priority over internal activities, such as the measurement. The purpose of this priority is to not keep the master waiting and the $I^2C$ bus engaged for longer than necessary.

### Internal Clock

The device has an internal clock for internal digital logic functions and timing management. This clock is not available to external usage.

### H-Bridge for Set/Reset Strap Drive

The ASIC contains large switching FETs capable of delivering a large but brief pulse to the Set/Reset strap of the sensor. This strap is largely a resistive load. There is no need for an external Set/Reset circuit. The controlling of the Set/Reset function is done automatically by the ASIC for each measurement. One half of the difference from the measurements taken after a set pulse and after a reset pulse will be put in the data output register for each of the three axes. By doing so, the sensor's internal offset and its temperature dependence is removed/cancelled for all measurements. The set/reset pulses also effectively remove the past magnetic history (magnetism) in the sensor, if any.

For each "measurement", the ASIC:

1. Sends a "Set" pulse
2. Takes one measurement (Mset)
3. Sends a "Reset" pulse
4. Takes another measurement (Mreset)
5. Puts the following result in sensor's data output register:

**Output = [Mset – Mreset] / 2**

### Charge Current Limit

The current that reservoir capacitor (C1) can draw when charging is limited for both single supply and dual supply configurations. This prevents drawing down the supply voltage (VDD).

## MODES OF OPERATION

This device has several operating modes whose primary purpose is power management and is controlled by the Mode Register. This section describes these modes.

### Continuous-Measurement Mode

During continuous-measurement mode, the device continuously makes measurements, at user selectable rate, and places measured data in data output registers. Data can be re-read from the data output registers if necessary; however, if the master does not ensure that the data register is accessed before the completion of the next measurement, the data output registers are updated with the new measurement. To conserve current between measurements, the device is placed in a state similar to idle mode, but the Mode Register is not changed to Idle Mode. That is, MD[n] bits are unchanged. Settings in the Configuration Register A affect the data output rate (bits DO[n]), the measurement configuration (bits MS[n]), when in continuous-measurement mode. All registers maintain values while in continuous-measurement mode. The $I^2C$ bus is enabled for use by other devices on the network in while continuous-measurement mode.

### Single-Measurement Mode

This is the default power-up mode. During single-measurement mode, the device makes a single measurement and places the measured data in data output registers. After the measurement is complete and output data registers are updated, the device is placed in idle mode, and the Mode Register *is* changed to idle mode by setting MD[n] bits. Settings in the configuration register affect the measurement configuration (bits MS[n])when in single-measurement mode. All registers maintain values while in single-measurement mode. The $I^2C$ bus is enabled for use by other devices on the network while in single-measurement mode.

### Idle Mode

During this mode the device is accessible through the $I^2C$ bus, but major sources of power consumption are disabled, such as, but not limited to, the ADC, the amplifier, and the sensor bias current. All registers maintain values while in idle mode. The $I^2C$ bus is enabled for use by other devices on the network while in idle mode.

## REGISTERS

This device is controlled and configured via a number of on-chip registers, which are described in this section. In the following descriptions, *set* implies a logic 1, and *reset* or *clear* implies a logic 0, unless stated otherwise.

**Register List**
The table below lists the registers and their access. All address locations are 8 bits.

| Address Location | Name | Access |
|---|---|---|
| 00 | Configuration Register A | Read/Write |
| 01 | Configuration Register B | Read/Write |
| 02 | Mode Register | Read/Write |
| 03 | Data Output X MSB Register | Read |
| 04 | Data Output X LSB Register | Read |
| 05 | Data Output Z MSB Register | Read |
| 06 | Data Output Z LSB Register | Read |
| 07 | Data Output Y MSB Register | Read |
| 08 | Data Output Y LSB Register | Read |
| 09 | Status Register | Read |
| 10 | Identification Register A | Read |
| 11 | Identification Register B | Read |
| 12 | Identification Register C | Read |

Table2: Register List

**Register Access**

This section describes the process of reading from and writing to this device. The devices uses an address pointer to indicate which register location is to be read from or written to. These pointer locations are sent from the master to this slave device and succeed the 7-bit address (0x1E) plus 1 bit read/write identifier, i.e. 0x3D for read and 0x3C for write.

To minimize the communication between the master and this device, the address pointer updated automatically without master intervention. The register pointer will be incremented by 1 automatically after the current register has been read successfully.

The address pointer value itself cannot be read via the $I^2C$ bus.
Any attempt to read an invalid address location returns 0's, and any write to an invalid address location or an undefined bit within a valid address location is ignored by this device.

To move the address pointer to a random register location, first issue a "write" to that register location with no data byte following the commend. For example, to move the address pointer to register 10, send 0x3C 0x0A.

### Configuration Register A

The configuration register is used to configure the device for setting the data output rate and measurement configuration. CRA0 through CRA7 indicate bit locations, with *CRA* denoting the bits that are in the configuration register. CRA7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit.CRA default is 0x10.

| CRA7 | CRA6 | CRA5 | CRA4 | CRA3 | CRA2 | CRA1 | CRA0 |
|---|---|---|---|---|---|---|---|
| (0) | MA1(0) | MA0(0) | DO2 (1) | DO1 (0) | DO0 (0) | MS1 (0) | MS0 (0) |

Table 3: Configuration Register A

| Location | Name | Description |
|---|---|---|
| CRA7 | CRA7 | Bit CRA7 is reserved for future function. Set to 0 when configuring CRA. |
| CRA6 to CRA5 | MA1 to MA0 | Select number of samples averaged (1 to 8) per measurement output.<br>00 = 1(Default); 01 = 2; 10 = 4; 11 = 8 |
| CRA4 to CRA2 | DO2 to DO0 | Data Output Rate Bits. These bits set the rate at which data is written to all three data output registers. |
| CRA1 to CRA0 | MS1 to MS0 | Measurement Configuration Bits. These bits define the measurement flow of the device, specifically whether or not to incorporate an applied bias into the measurement. |

Table 4: Configuration Register A Bit Designations

The Table below shows all selectable output rates in continuous measurement mode. All three channels shall be measured within a given output rate. Other output rates with maximum rate of 160 Hz can be achieved by monitoring DRDY interrupt pin in single measurement mode.

| DO2 | DO1 | DO0 | Typical Data Output Rate (Hz) |
|---|---|---|---|
| 0 | 0 | 0 | 0.75 |
| 0 | 0 | 1 | 1.5 |
| 0 | 1 | 0 | 3 |
| 0 | 1 | 1 | 7.5 |
| 1 | 0 | 0 | 15 (Default) |
| 1 | 0 | 1 | 30 |
| 1 | 1 | 0 | 75 |
| 1 | 1 | 1 | Reserved |

Table 5: Data Output Rates

| MS1 | MS0 | Measurement Mode |
|---|---|---|
| 0 | 0 | Normal measurement configuration (Default). In normal measurement configuration the device follows normal measurement flow. The positive and negative pins of the resistive load are left floating and high impedance. |
| 0 | 1 | Positive bias configuration for X, Y, and Z axes. In this configuration, a positive current is forced across the resistive load for all three axes. |
| 1 | 0 | Negative bias configuration for X, Y and Z axes. In this configuration, a negative current is forced across the resistive load for all three axes.. |
| 1 | 1 | This configuration is reserved. |

Table 6: Measurement Modes

**Configuration Register B**

The configuration register B for setting the device gain. CRB0 through CRB7 indicate bit locations, with *CRB* denoting the bits that are in the configuration register. CRB7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit. CRB default is 0x20.

| CRB7 | CRB6 | CRB5 | CRB4 | CRB3 | CRB2 | CRB1 | CRB0 |
|---|---|---|---|---|---|---|---|
| GN2 (0) | GN1 (0) | GN0 (1) | (0) | (0) | (0) | (0) | (0) |

Table 7: Configuration B Register

| Location | Name | Description |
|---|---|---|
| CRB7 to CRB5 | GN2 to GN0 | Gain Configuration Bits. These bits configure the gain for the device. The gain configuration is common for all channels. |
| CRB4 to CRB0 | 0 | These bits must be cleared for correct operation. |

Table 8: Configuration Register B Bit Designations

The table below shows nominal gain settings. Use the “Gain” column to convert counts to Gauss. The “Digital Resolution” column is the theoretical value in term of milli-Gauss per count (LSb) which is the inverse of the values in the “Gain” column. The effective resolution of the usable signal also depends on the noise floor of the system, i.e.

Effective Resolution = Max (Digital Resolution, Noise Floor)

Choose a lower gain value (higher GN#) when total field strength causes overflow in one of the data output registers (saturation). Note that the very first measurement after a gain change maintains the same gain as the previous setting. **The new gain setting is effective from the second measurement and on.**

| GN2 | GN1 | GN0 | Recommended Sensor Field Range | Gain (LSb/ Gauss) | Digital Resolution (mG/LSb) | Output Range |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | ± 0.88 Ga | 1370 | 0.73 | 0xF800–0x07FF (-2048–2047 ) |
| 0 | 0 | 1 | ± 1.3 Ga | 1090 (default) | 0.92 | 0xF800–0x07FF (-2048–2047 ) |
| 0 | 1 | 0 | ± 1.9 Ga | 820 | 1.22 | 0xF800–0x07FF (-2048–2047 ) |
| 0 | 1 | 1 | ± 2.5 Ga | 660 | 1.52 | 0xF800–0x07FF (-2048–2047 ) |
| 1 | 0 | 0 | ± 4.0 Ga | 440 | 2.27 | 0xF800–0x07FF (-2048–2047 ) |
| 1 | 0 | 1 | ± 4.7 Ga | 390 | 2.56 | 0xF800–0x07FF (-2048–2047 ) |
| 1 | 1 | 0 | ± 5.6 Ga | 330 | 3.03 | 0xF800–0x07FF (-2048–2047 ) |
| 1 | 1 | 1 | ± 8.1 Ga | 230 | 4.35 | 0xF800–0x07FF (-2048–2047 ) |

Table 9: Gain Settings

### Mode Register

The mode register is an 8-bit register from which data can be read or to which data can be written. This register is used to select the operating mode of the device. MR0 through MR7 indicate bit locations, with *MR* denoting the bits that are in the mode register. MR7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit. Mode register default is 0x01.

| MR7 | MR6 | MR5 | MR4 | MR3 | MR2 | MR1 | MR0 |
|---|---|---|---|---|---|---|---|
| HS(0) | (0) | (0) | (0) | (0) | (0) | MD1 (0) | MD0 (1) |

Table 10: Mode Register

| Location | Name | Description |
|---|---|---|
| MR7 to MR2 | HS | Set this pin to enable High Speed I2C, 3400kHz. |
| MR1 to MR0 | MD1 to MD0 | Mode Select Bits. These bits select the operation mode of this device. |

Table 11: Mode Register Bit Designations

| MD1 | MD0 | Operating Mode |
|---|---|---|
| 0 | 0 | Continuous-Measurement Mode. In continuous-measurement mode, the device continuously performs measurements and places the result in the data register. RDY goes high when new data is placed in all three registers. After a power-on or a write to the mode or configuration register, the first measurement set is available from all three data output registers after a period of $2/f_{DO}$ and subsequent measurements are available at a frequency of $f_{DO}$, where $f_{DO}$ is the frequency of data output. |
| 0 | 1 | Single-Measurement Mode (Default). When single-measurement mode is selected, device performs a single measurement, sets RDY high and returned to idle mode. Mode register returns to idle mode bit values. The measurement remains in the data output register and RDY remains high until the data output register is read or another measurement is performed. |
| 1 | 0 | Idle Mode. Device is placed in idle mode. |
| 1 | 1 | Idle Mode. Device is placed in idle mode. |

Table 12: Operating Modes

**Data Output X Registers A and B**

The data output X registers are two 8-bit registers, data output register A and data output register B. These registers store the measurement result from channel X. Data output X register A contains the MSB from the measurement result, and data output X register B contains the LSB from the measurement result. The value stored in these two registers is a 16-bit value in 2's complement form, whose range is 0xF800 to 0x07FF. DXRA0 through DXRA7 and DXRB0 through DXRB7 indicate bit locations, with *DXRA* and *DXRB* denoting the bits that are in the data output X registers. DXRA7 and DXRB7 denote the first bit of the data stream. The number in parenthesis indicates the default value of that bit.

In the event the ADC reading overflows or underflows for the given channel, or if there is a math overflow during the bias measurement, this data register will contain the value -4096. This register value will clear when after the next valid measurement is made.

| **DXRA7** | **DXRA6** | **DXRA5** | **DXRA4** | **DXRA3** | **DXRA2** | **DXRA1** | **DXRA0** |
|---|---|---|---|---|---|---|---|
| (0) | (0) | (0) | (0) | (0) | (0) | (0) | (0) |
| **DXRB7** | **DXRB6** | **DXRB5** | **DXRB4** | **DXRB3** | **DXRB2** | **DXRB1** | **DXRB0** |
| (0) | (0) | (0) | (0) | (0) | (0) | (0) | (0) |

Table 13: Data Output X Registers A and B

**Data Output Y Registers A and B**

The data output Y registers are two 8-bit registers, data output register A and data output register B. These registers store the measurement result from channel Y. Data output Y register A contains the MSB from the measurement result, and data output Y register B contains the LSB from the measurement result. The value stored in these two registers is a 16-bit value in 2's complement form, whose range is 0xF800 to 0x07FF. DYRA0 through DYRA7 and DYRB0 through DYRB7 indicate bit locations, with *DYRA* and *DYRB* denoting the bits that are in the data output Y registers. DYRA7 and DYRB7 denote the first bit of the data stream. The number in parenthesis indicates the default value of that bit.

In the event the ADC reading overflows or underflows for the given channel, or if there is a math overflow during the bias measurement, this data register will contain the value -4096. This register value will clear when after the next valid measurement is made.

| **DYRA7** | **DYRA6** | **DYRA5** | **DYRA4** | **DYRA3** | **DYRA2** | **DYRA1** | **DYRA0** |
|---|---|---|---|---|---|---|---|
| (0) | (0) | (0) | (0) | (0) | (0) | (0) | (0) |
| **DYRB7** | **DYRB6** | **DYRB5** | **DYRB4** | **DYRB3** | **DYRB2** | **DYRB1** | **DYRB0** |
| (0) | (0) | (0) | (0) | (0) | (0) | (0) | (0) |

Table 14: Data Output Y Registers A and B

**Data Output Z Registers A and B**

The data output Z registers are two 8-bit registers, data output register A and data output register B. These registers store the measurement result from channel Z. Data output Z register A contains the MSB from the measurement result, and data output Z register B contains the LSB from the measurement result. The value stored in these two registers is a 16-bit value in 2's complement form, whose range is 0xF800 to 0x07FF. DZRA0 through DZRA7 and DZRB0 through DZRB7 indicate bit locations, with *DZRA* and *DZRB* denoting the bits that are in the data output Z registers. DZRA7 and DZRB7 denote the first bit of the data stream. The number in parenthesis indicates the default value of that bit.

In the event the ADC reading overflows or underflows for the given channel, or if there is a math overflow during the bias measurement, this data register will contain the value -4096. This register value will clear when after the next valid measurement is made.

| DZRA7 | DZRA6 | DZRA5 | DZRA4 | DZRA3 | DZRA2 | DZRA1 | DZRA0 |
|---|---|---|---|---|---|---|---|
| (0) | (0) | (0) | (0) | (0) | (0) | (0) | (0) |
| **DZRB7** | **DZRB6** | **DZRB5** | **DZRB4** | **DZRB3** | **DZRB2** | **DZRB1** | **DZRB0** |
| (0) | (0) | (0) | (0) | (0) | (0) | (0) | (0) |

Table 15: Data Output Z Registers A and B

**Data Output Register Operation**

When one or more of the output registers are read, new data cannot be placed in any of the output data registers until all six data output registers are read. This requirement also impacts DRDY and RDY, which cannot be cleared until new data is placed in all the output registers.

**Status Register**

The status register is an 8-bit read-only register. This register is used to indicate device status. SR0 through SR7 indicate bit locations, with *SR* denoting the bits that are in the status register. SR7 denotes the first bit of the data stream.

| SR7 | SR6 | SR5 | SR4 | SR3 | SR2 | SR1 | SR0 |
|---|---|---|---|---|---|---|---|
| (0) | (0) | (0) | (0) | (0) | (0) | LOCK (0) | RDY(0) |

Table 16: Status Register

| Location | Name | Description |
|---|---|---|
| SR7 to SR2 | 0 | These bits are reserved. |
| SR1 | LOCK | Data output register lock. This bit is set when:<br>1.some but not all for of the six data output registers have been read,<br>2. Mode register has been read.<br>When this bit is set, the six data output registers are locked and any new data will not be placed in these register until one of these conditions are met:<br>1.all six bytes have been read, 2. the mode register is changed,<br>3. the measurement configuration (CRA) is changed,<br>4. power is reset. |
| SR0 | RDY | Ready Bit. Set when data is written to all six data registers. Cleared when device initiates a write to the data output registers and after one or more of the data output registers are written to. When RDY bit is clear it shall remain cleared for a 250 µs. DRDY pin can be used as an alternative to the status register for monitoring the device for measurement data. |

Table 17: Status Register Bit Designations

**Identification Register A**

The identification register A is used to identify the device. IRA0 through IRA7 indicate bit locations, with *IRA* denoting the bits that are in the identification register A. IRA7 denotes the first bit of the data stream. The number in parenthesis indicates the default value of that bit.

The identification value for this device is stored in this register. This is a read-only register.
Register values. ASCII value *H*

| IRA7 | IRA6 | IRA5 | IRA4 | IRA3 | IRA2 | IRA1 | IRA0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

Table 18: Identification Register A Default Values

**Identification Register B**

The identification register B is used to identify the device. IRB0 through IRB7 indicate bit locations, with *IRB* denoting the bits that are in the identification register A. IRB7 denotes the first bit of the data stream.

Register values. ASCII value *4*

| IRB7 | IRB6 | IRB5 | IRB4 | IRB3 | IRB2 | IRB1 | IRB0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

Table 19: Identification Register B Default Values

**Identification Register C**

The identification register C is used to identify the device. IRC0 through IRC7 indicate bit locations, with *IRC* denoting the bits that are in the identification register A. IRC7 denotes the first bit of the data stream.

Register values. ASCII value *3*

| IRC7 | IRC6 | IRC5 | IRC4 | IRC3 | IRC2 | IRC1 | IRC0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |

Table 20: Identification Register C Default Values

## I$^2$C COMMUNICATION PROTOCOL

The HMC5883L communicates via a two-wire I$^2$C bus system as a slave device. The HMC5883L uses a simple protocol with the interface protocol defined by the I$^2$C bus specification, and by this document. The data rate is at the standard-mode 100kbps or 400kbps rates as defined in the I$^2$C Bus Specifications. The bus bit format is an 8-bit Data/Address send and a 1-bit acknowledge bit. The format of the data bytes (payload) shall be case sensitive ASCII characters or binary data to the HMC5883L slave, and binary data returned. Negative binary values will be in two's complement form. The default (factory) HMC5883L 8-bit slave address is 0x3C for write operations, or 0x3D for read operations.

The HMC5883L Serial Clock (SCL) and Serial Data (SDA) lines require resistive pull-ups (Rp) between the master device (usually a host microprocessor) and the HMC5883L. Pull-up resistance values of about 2.2K to 10K ohms are recommended with a nominal VDDIO voltage. Other resistor values may be used as defined in the I$^2$C Bus Specifications that can be tied to VDDIO.

The SCL and SDA lines in this bus specification may be connected to multiple devices. The bus can be a single master to multiple slaves, or it can be a multiple master configuration. All data transfers are initiated by the master device, which is responsible for generating the clock signal, and the data transfers are 8 bit long. All devices are addressed by I$^2$C's unique 7-bit address. After each 8-bit transfer, the master device generates a 9$^{th}$ clock pulse, and releases the SDA line. The receiving device (addressed slave) will pull the SDA line low to acknowledge (ACK) the successful transfer or leave the SDA high to negative acknowledge (NACK).

Per the $I^2C$ spec, all transitions in the SDA line must occur when SCL is low. This requirement leads to two unique conditions on the bus associated with the SDA transitions when SCL is high. Master device pulling the SDA line low while the SCL line is high indicates the Start (S) condition, and the Stop (P) condition is when the SDA line is pulled high while the SCL line is high. The $I^2C$ protocol also allows for the Restart condition in which the master device issues a second start condition without issuing a stop.

All bus transactions begin with the master device issuing the start sequence followed by the slave address byte. The address byte contains the slave address; the upper 7 bits (bits7-1), and the Least Significant bit (LSb). The LSb of the address byte designates if the operation is a read (LSb=1) or a write (LSb=0). At the 9th clock pulse, the receiving slave device will issue the ACK (or NACK). Following these bus events, the master will send data bytes for a write operation, or the slave will clock out data with a read operation. All bus transactions are terminated with the master issuing a stop sequence.

$I^2C$ bus control can be implemented with either hardware logic or in software. Typical hardware designs will release the SDA and SCL lines as appropriate to allow the slave device to manipulate these lines. In a software implementation, care must be taken to perform these tasks in code.

## OPERATIONAL EXAMPLES

The HMC5883L has a fairly quick stabilization time from no voltage to stable and ready for data retrieval. The nominal 56 milli-seconds with the factory default single measurement mode means that the six bytes of magnetic data registers (DXRA, DXRB, DZRA, DZRB, DYRA, and DYRB) are filled with a valid first measurement.

To change the measurement mode to continuous measurement mode, after the power-up time send the three bytes:

0x3C 0x02 0x00

This writes the 00 into the second register or mode register to switch from single to continuous measurement mode setting. With the data rate at the factory default of 15Hz updates, a 67 milli-second typical delay should be allowed by the $I^2C$ master before querying the HMC5883L data registers for new measurements. To clock out the new data, send:

0x3D, and clock out DXRA, DXRB, DZRA, DZRB, DYRA, and DYRB located in registers 3 through 8. The HMC5883L will automatically re-point back to register 3 for the next 0x3D query. All six data registers must be read properly before new data can be placed in any of these data registers.

Below is an example of a (power-on) initialization process for “continuous-measurement mode”:

1. Write CRA (00) – send **0x3C 0x00 0x70** (8-average, 15 Hz default, normal measurement)
2. Write CRB (01) – send **0x3C 0x01 0xA0** (Gain=5, or any other desired gain)
3. Write Mode (02) – send **0x3C 0x02 0x00** (Continuous-measurement mode)
4. Wait 6 ms or monitor status register or DRDY hardware interrupt pin
5. Loop
   Send **0x3D 0x06** (Read all 6 bytes. If gain is changed then this data set is using previous gain)
   Convert three 16-bit 2’s compliment hex values to decimal values and assign to X, Z, Y, respectively.
   Send **0x3C 0x03** (point to first data register 03)
   Wait about 67 ms (if 15 Hz rate) or monitor status register or DRDY hardware interrupt pin
   End_loop

Below is an example of a (power-on) initialization process for “single-measurement mode”:

1. Write CRA (00) – send **0x3C 0x00 0x70** (8-average, 15 Hz default or any other rate, normal measurement)
2. Write CRB (01) – send **0x3C 0x01 0xA0** (Gain=5, or any other desired gain)
3. For each measurement query:
   Write Mode (02) – send **0x3C 0x02 0x01** (Single-measurement mode)
   Wait 6 ms or monitor status register or DRDY hardware interrupt pin
   Send **0x3D 0x06** (Read all 6 bytes. If gain is changed then this data set is using previous gain)
   Convert three 16-bit 2’s compliment hex values to decimal values and assign to X, Z, Y, respectively.

## SELF TEST OPERATION

To check the HMC5883L for proper operation, a self test feature in incorporated in which the sensor offset straps are excited to create a nominal field strength (bias field) to be measured. To implement self test, the least significant bits (MS1 and MS0) of configuration register A are changed from 00 to 01 (positive bias) or 10 (negetive bias).

Then, by placing the mode register into single or continuous-measurement mode, two data acquisition cycles will be made on each magnetic vector. The first acquisition will be a set pulse followed shortly by measurement data of the external field. The second acquisition will have the offset strap excited (about 10 mA) in the positive bias mode for X, Y, and Z axes to create about a 1.1 gauss self test field plus the external field. The first acquisition values will be subtracted from the second acquisition, and the net measurement will be placed into the data output registers.

Since self test adds ~1.1 Gauss additional field to the existing field strength, using a reduced gain setting prevents sensor from being saturated and data registers overflowed. For example, if the configuration register B is set to 0xA0 (Gain=5), values around +452 LSb (1.16 Ga * 390 LSb/Ga) will be placed in the X and Y data output registers and around +421 (1.08 Ga * 390 LSb/Ga) will be placed in Z data output register. To leave the self test mode, change MS1 and MS0 bit of the configuration register A back to 00 (Normal Measurement Mode). Acceptable limits of the self test values depend on the gain setting. Limits for Gain=5 is provided in the specification table.

Below is an example of a “positive self test” process using continuous-measurement mode:

1. Write CRA (00) – send **0x3C 0x00 0x71** (8-average, 15 Hz default, positive self test measurement)
2. Write CRB (01) – send **0x3C 0x01 0xA0** (Gain=5)
3. Write Mode (02) – send **0x3C 0x02 0x00** (Continuous-measurement mode)
4. Wait 6 ms or monitor status register or DRDY hardware interrupt pin
5. Loop
   Send **0x3D 0x06** (Read all 6 bytes. If gain is changed then this data set is using previous gain)
   Convert three 16-bit 2’s compliment hex values to decimal values and assign to X, Z, Y, respectively.
   Send **0x3C 0x03** (point to first data register 03)
   Wait about 67 ms (if 15 Hz rate) or monitor status register or DRDY hardware interrupt pin
   End_loop
6. Check limits –
   If all 3 axes (X, Y, and Z) are within reasonable limits (243 to 575 for Gain=5, adjust these limits basing on the gain setting used. See an example below.) Then
   All 3 axes pass positive self test
   Write CRA (00) – send **0x3C 0x00 0x70** (Exit self test mode and this procedure)
   Else
   If Gain<7
   Write CRB (01) – send **0x3C 0x01 0x_0** (Increase gain setting and retry, skip the next data set)
   Else
   At least one axis did not pass positive self test
   Write CRA (00) – send **0x3C 0x00 0x70** (Exit self test mode and this procedure)
   End If

Below is an example of how to adjust the “positive self” test limits basing on the gain setting:

1. If Gain = 6, self test limits are:
   Low Limit = 243 * 330/390 = 206
   High Limit = 575 * 330/390 = 487

2. If Gain = 7, self test limits are:
   Low Limit = 243 * 230/390 = 143
   High Limit = 575 * 230/390 = 339

## SCALE FACTOR TEMPERATURE COMPENSATION

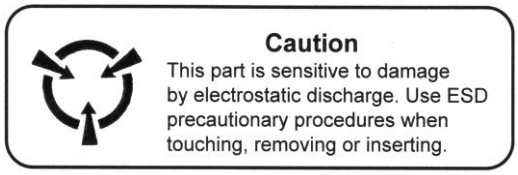
The built-in self test can also be used to periodically compensate the scaling errors due to temperature variations. A compensation factor can be found by comparing the self test outputs with the ones obtained at a known temperature. For example, if the self test output is 400 at room temperature and 300 at the current temperature then a compensation factor of (400/300) should be applied to all current magnetic readings. A temperature sensor is not required using this method.

Below is an example of a temperature compensation process using positive self test method:

1. If self test measurement at a temperature “when the last magnetic calibration was done”:
   X_STP = 400
   Y_STP = 410
   Z_STP = 420
2. If self test measurement at a different tmperature:
   X_STP = 300 (Lower than before)
   Y_STP = 310 (Lower than before)
   Z_STP = 320 (Lower than before)
   Then
   X_TempComp = 400/300
   Y_TempComp = 410/310
   Z_TempComp = 420/320
3. Applying to all new measurements:
   X = X * X_TempComp
   Y = Y * Y_TempComp
   Z = Z * Z_TempComp
   Now all 3 axes are temperature compensated, i.e. sensitivity is same as “when the last magnetic calibration was done”; therefore, the calibration coefficients can be applied without modification.
4. Repeat this process periodically or,for every Δt degrees of temperature change measured, if available.

## ORDERING INFORMATION

| Ordering Number | Product |
|---|---|
| HMC5883L-T | Cut Tape |
| HMC5883L-TR | Tape and Reel 4k pieces/reel |




**CAUTION: ESDS CAT. 1B**

## FIND OUT MORE

For more information on Honeywell’s Magnetic Sensors visit us online at www.magneticsensors.com or contact us at 1-800-323-8295 (763-954-2474 internationally).

The application circuits herein constitute typical usage and interface of Honeywell product. Honeywell does not warranty or assume liability of customer-designed circuits derived from this description or depiction.

Honeywell reserves the right to make changes to improve reliability, function or design. Honeywell does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

U.S. Patents 4,441,072, 4,533,872, 4,569,742, 4,681,812, 4,847,584 and 6,529,114 apply to the technology described

Honeywell
12001 Highway 55
Plymouth, MN 55441
Tel: 800-323-8295
www.magneticsensors.com

Form # 900405 Rev E
February 2013
©2010 Honeywell International Inc.

Honeywell

# Implementasi Sensor Compas HMC5883L Terhadap Gerak Robot *Micromouse* dengan Menggunakan Algoritma PID

**Della Diana*, Amperawan, Johansyah Al Rasyid**
Politeknik Negeri Sriwijaya, Palembang
E-mail: delladianaa@gmail.com

***ABSTRACT***

*Robot micromouse is one of the mobile robots that can pass through the labyrinth path to find the destination point that has been determined. This robot will move freely with 8 pieces of infrared sensors that will detect every passing labyrinth wall where the direction of movement of the robot is determined when there is a response to the object in front, right, left and behind the robot. This micromouse robot uses PID Algorithm and micromouse robot design is made using Arduino Mega 2560 as control system, compass sensor as maping or mapping robot direction and 2 motor drivers to move 4 dc motor using omni wheel.*

***Keywords**: Robot Micromouse, Infrared, PID Algorithm, Arduino Mega 2560*

**ABSTRAK**

Robot *micromouse* adalah salah satu *mobile* robot yang dapat melewati jalur labirin untuk menemukan titik tujuan yang telah ditentukan. Robot ini akan bergerak bebas dengan 8 buah sensor *infrared* yang akan mendeteksi setiap dinding labirin yang dilewati dimana arah pergerakan dari robot ini ditentukan ketika ada respon terhadap *obyek* di depan, kanan, kiri dan belakang robot. Robot *micromouse* ini menggunakan Algoritma PID dan rancang bangun robot *micromouse* ini dibuat dengan menggunakan Arduino Mega 2560 sebagai sistem kontrol, sensor kompas sebagai *maping* atau pemetaan arah robot dan 2 buah driver motor untuk menggerakan 4 buah motor dc menggunakan roda omni.

**Kata kunci***:* Robot *Micromouse, Infrared,* Algoritma PID, Arduino Mega 2560

## 1. PENDAHULUAN

Robot berasal dari bahasa Czech, robota, yang berarti pekerja. Robot terdiri dari beberapa komponen diantaranya adalah mekanik, elektronik, kendali berbasis prosesor dan pemograman. Robot merupakan alat mekanik yang dibuat untuk membantu pekerjaan manusia yang diharapkan mampu melakukan pekerjaan tugas fisik, baik menggunakan pengontrolan oleh manusia, ataupun menggunakan program yang telah dibuat [1]. Salah satu kasus yang menggunakan bantuan sebuah robot adalah dimana suatu keadaan untuk bekerja di daerah yang menyerupai labirin yang sulit untuk dijangkau oleh manusia untuk mencari jalan keluar atau tujuan. Pada saat itulah digunakanlah sebuah robot yang dapat mencari jalan keluar ataupun menemukan titik tujuan tertentu.

Robot yang dapat membantu kegiatan manusia dalam melewati kondisi atau keadaan di labirin untuk mencari titik tujuan ataup jalan keluar yang tidak dapat dijangkau oleh manusia yaitu *Robot Micromouse*. *Robot Micromouse* merupakan *mobile robot* yang memiliki tujuan untuk menyelesaikan lintasan berupa labirin. Pergerakan robot ini diatur oleh delapan buah sensor *infrared* untuk mendeteksi setiap dinding labirin yang dilewati. Arah pergerakan dari robot ini ditentukan ketika ada respon terhadap *obyek* di depan, kanan, kiri dan belakang robot.

Rancang bangun robot *micromouse* dibuat dengan menggunakan Arduino Mega 2560 sebagai sistem kontrol dengan algoritma PID, sensor kompas sebagai *maping* atau pemetaan arah robot dan dua buah driver motor untuk menggerakan empat buah motor dc menggunakan roda omni. Penggunaan algoritma PID bertujuan untuk memperbaiki kinerja sistem di mana masing-masing pengendali akan saling melengkapi dan menutupi dengan ke lemahan dan kelebihan masing-masing. PID digunakan dalam sebuah sistem loop tertutup yang melibatkan umpan balik dari output sistem untuk mencapai respon yang diinginkan. PID dapat mengendalikan variabel input dengan memanipulasi variabel output sehingga diperoleh variabel input baru agar menghasilkan output system yang sesuai [4].

## 2. DASAR TEORI

### *2.1 Micromouse*

Robot *Micromouse* adalah robot yang termasuk ke dalam jenis Robot *Mobile* yaitu *Autonomous Mobile* Robot yang pengendalian gerak robotnya berdasarkan program kendali yang telah diberikan sehingga robot tersebut seperti bergerak sendiri. Robot *Micromouse* ini merupakan robot yang pintar yang mampu bergerak dan berjalan dengan bebas di dalam sebuah labirin tanpa mengenai atau tersentuh dengan sel-sel dinding pada labirin tersebut, robot

Manuscript received May 27, 2017; Revised June 24, 2017; Accepted July 05, 2017. Date of publication July 26, 2017.
Digital Object Identifier 10.21063/JTE.2017.3133616. Copyright © 2017 ITP Press.

*micromouse* akan mengetahui harus ke arah mana bergerak dan harus berputar berapa derajat jika bertemu jalan buntu di dalam labirin [2].

### 2.2 Kontrol PID

PID merupakan pengabungan dari tiga macam pengendali, yaitu pengendali proporsional, pengendali integral, dan pengendali derivative. Aksi kontrol Proporsional memiliki Karakteristik dimana Proporsional dapat mengurangi waktu naik, menambah *overshoot*, dan mengurangi kesalahan keadaan tunak. Fungsi alih sistem dengan menambahkan aksi pengontrolan P menjadi:

$$\frac{p(s)}{q(s)} = \frac{K_p}{s^2 + 5s + (8 + K_p)} \quad (1)$$

Aksi kendali proporsional derivative memiliki karakteristik dimana *Proporsional Derivative* (PD) dapat mengurangi *overshoot* dan waktu turun. Fungsi alih sistem dengan aksi pengontrolan PD menjadi:

$$\frac{p(s)}{q(s)} = \frac{K_p + K_D s}{s^2 + (5 + K_D)s + (8 + K_p)} \quad (2)$$

Aksi kendali Proporsional *Integral Controller* memiliki karakteristik mengurangi waktu naik, menambah *Overshoot* dan waktu turun, serta menghilangkan kesalahan keadaan tunak. Fungsi alih sistem dengan penambahan aksi pengontrolan PI menjadi:

$$\frac{p(s)}{q(s)} = \frac{K_i + K_p s}{s^3 + 5s^2 + (8 + K_p)s + K_i} \quad (3)$$

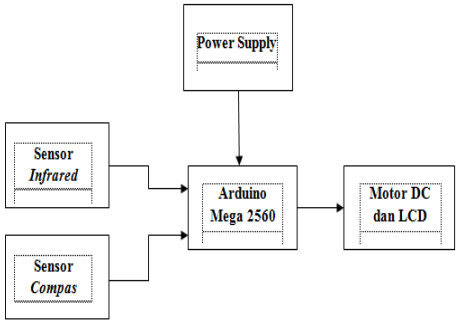
Aksi kontrol Proporsional – Integral – derivative**,** Aksi kontrol PID merupakan gabungan dari aksi P, I dan D dan fungsi alih sistem menjadi:

$$\frac{p(s)}{q(s)} = \frac{K_D s^2 + K_p s + K_i}{s^3 + (5 + K_D)s^2 + (8 + K_p)s + K_i} \quad (4)$$

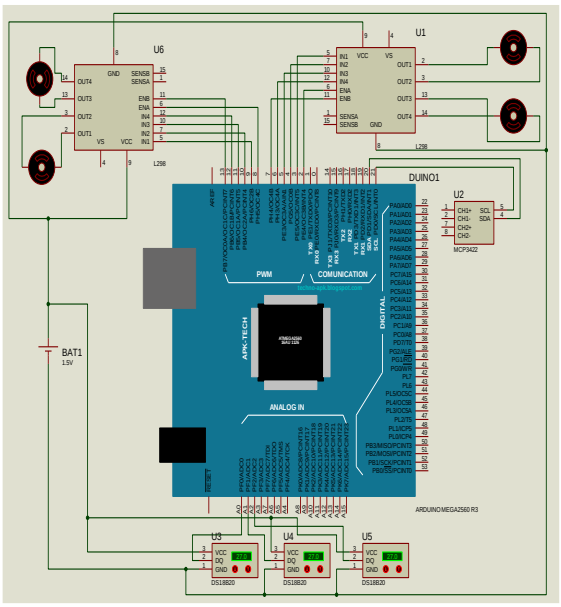
Dengan aksi kontrol P, I dan D, akan terbentuk kriteria sistem yang diinginkan dimana tanggapan sistem tidak memiliki *overshoot*, waktu naik yang cepat, dan kesalahan keadaan tunaknya sangat kecil mendekati nol [5].

### 2.3 Sensor COmpas HMC5883L

HMC5883L adalah sebuah sensor kompas digital yang menggunakan IC HMC5883L yang merupakan IC kompas digital 3 axis yang memiliki *interface*




(a)



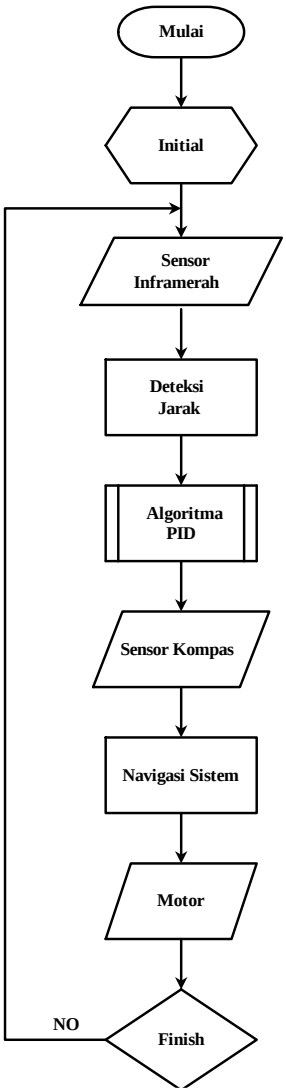

(b)

**Gambar** 1 Blok diagram dan skema rangakan robot *micromouse*

berupa 2 pin I2C. Sensor ini menggunakan medan magnet sebagai acuan dari pendeteksiannya sehingga sensor ini sangat sensitif terhadap rotasi dan arah hadap sensor. HMC5883L memiliki sensor magneto-resistive HMC118X series ber-resolusi tinggi, ditambah ASIC *dengan konten amplification, automatic degaussing strap driver, offset cancellation* dan 12bit ADC yang memungkinkan keakuratan kompas mencapat 1 sampai 2 derajat. Modul ini biasa digunakan untuk keperluan sistem navigasi otomatis, *mobile phone*, *netbook* dan perangkat navigasi personal. Modul ini memiliki 5 pin, diantaranya adalah VCC, Gnd, SDA, SCL, dan DRDY [3].

## 3. PERANCANGAN

Cara kerja keseluruhan robot yang akan dibuat dapat dilihat pada diagram blok sehingga kesuluruhan diagram blok akan menghasilkan suatu sistem yang dapat difungsikan atau dapat bekerja. Blok diagram dan skema rangkaian sistem robot micromouse ditunjukkan pada gambar 1.




**Gambar** 2 *Flowchart* proses percangan

Robot dirancang dengan desain berbentuk segi empat dengan 3 tingkatan sebagai berikut.

1. Pada tingkatan pertama atau yang paling bawah Robot *Micromouse* dipasang 4 buah motor DC dan 4 buah roda omni.
2. Masih pada tingkat pertama, dibagian atasnya dipasang 2 buah driver motor, 1 DC *step down* dan 1 buah baterai 1000mAh. 4 buah motor DC yang ada dibagian bawah akan dihubungkan ke driver motor kemudian dihubungkan ke DC step down.
3. Pada tingkat kedua dipasang Arduino Mega 2560. Pada tingkat kedua semua komponen yang digunakan harus dihubungkan pada pin-pin Arduino.
4. Pada tingkat ke 2 juga dipasang 8 *infared* SHARP 2Y0A21. *Infrared* tersebut dipasang bagian depan, serong kiri depan, serong kanan depan, kanan, kiri, serong kanan belakang, serong kiri belakang dan belakang dari robot *micromouse* tersebut.
5. Terakhir adalah tingkat 3, pada tingkat ke-3 ini dipasang LCD 16*2, sensor compas, dan switch.

## 4. ANALISA DATA

Tahap-tahap pengujian alat:

1. Atur Setpoint derajat compas pada robot.
2. Atur PID sesuai yang diingkan.
3. Pasangkan Bluetooth pada robot untuk pengambilan data error compas. Dimana data tersebut akan ditransfer ke aplikasi TERM Terminal.
4. Arahkan robot berlainan arah dengan derajat compas yang telah diatur.
5. Nyalakan robot, kemudian robot akan bergerak ke posisi derajat compas yang telah diatur.
6. Pada saat robot bergerak menuju posisi derajat compas yang diatur, data yang dibaca akan terukur dan dapat dilihat pada aplikasi TERM terminal.

Pada Proses pengujian alat, Sensor *compas* HMC5883L disetting dengan posisi $92.05^{0,}$, untuk melihat nilai error dari sensor tersebut maka posisi robot akan diputar ke berbagai sudut, sehingga saat robot kembali pada posisi semula yaitu sudut $92.05^{0}$ akan terlihat nilai error yang dialami sensor *compas* tersebut. Untuk melakukan pengujian yang efektif maka digunakan 3 buah percobaan dimana PID disetting berbeda – beda untuk menentukan PID yang menghasilkan nilai error ”0” pada pemetaan dan pembacaan sensor.

Pada tahap pertama PID diatur pada KP = 5, KI = 0, dan KD = 1, pada percobaan selanjutnya setting PID diubah pada konstanta KD menjadi 10 dan percobaan ketiga distting pada KP = 20, KI = 0, KD = 10.

**Tabel** 1 Setting PID 1

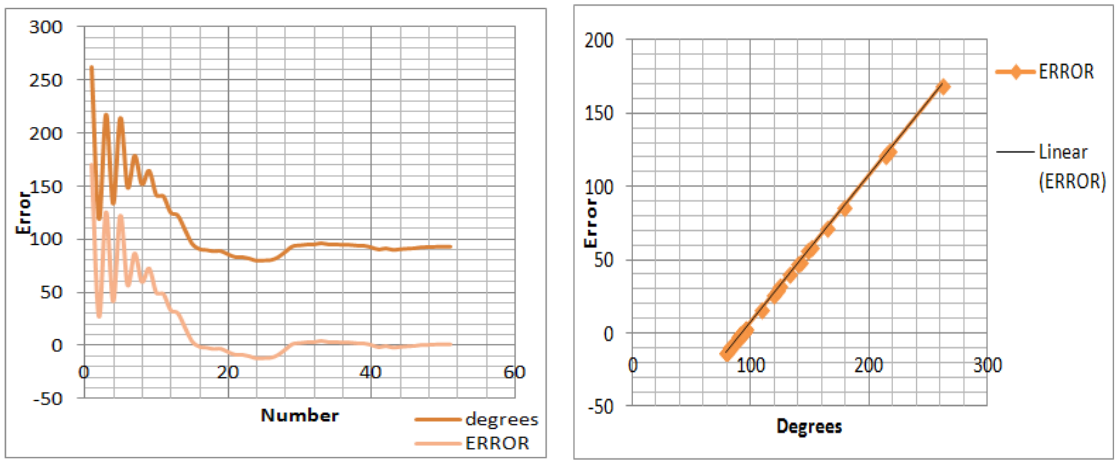
| Set Point | KP | KI | KD | Speed |
|---|---|---|---|---|
| $92.05^{0}$ | 5 | 0 | 1 | 100 |

**Tabel** 2 Setting PID 2

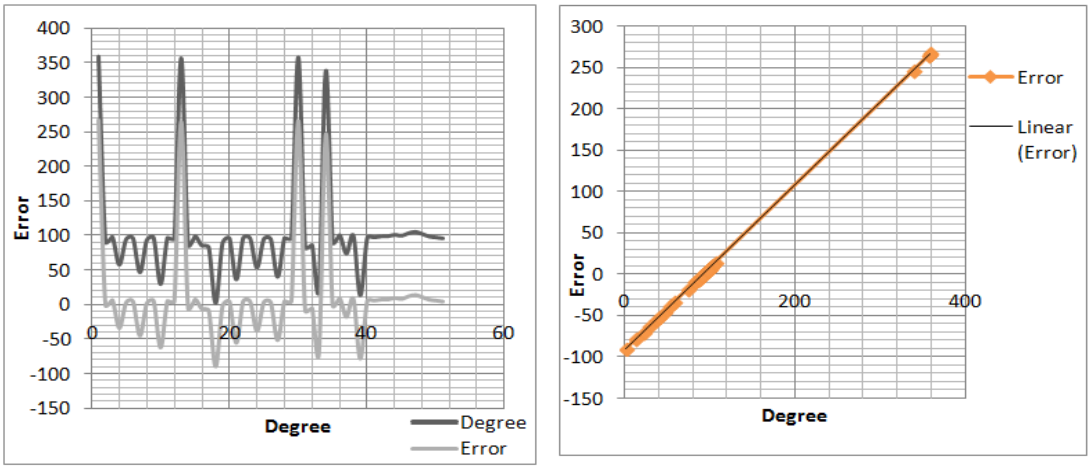
| Set Point | KP | KI | KD | Speed |
|---|---|---|---|---|
| $92.05^{0}$ | 5 | 0 | 10 | 100 |

**Tabel** 3 Setting PID 3

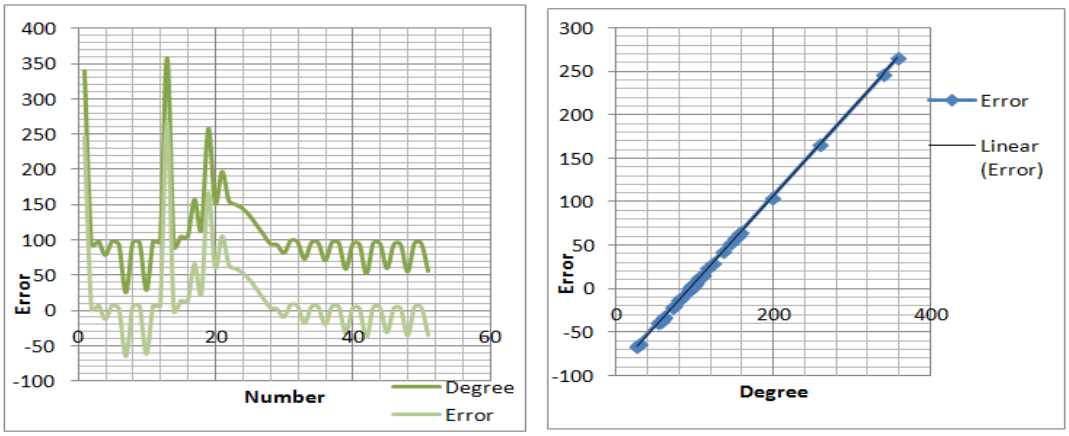
| Set Point | KP | KI | KD | Speed |
|---|---|---|---|---|
| $92.05^{0}$ | 20 | 0 | 10 | 100 |




**Gambar** 3 Grafik eror dan degree kompas serta hubungan liniernya untuk setting PID 1

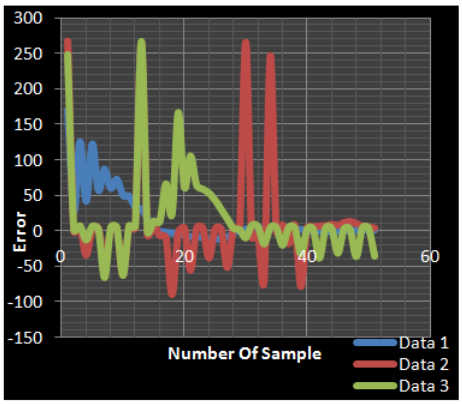



**Gambar** 4 Grafik eror dan degree kompas serta hubungan liniernya untuk setting PID 2




**Gambar** 5 Grafik eror dan degree kompas serta hubungan liniernya untuk setting PID 3

Dapat dianalisis bahwa nilai PID sangat mempengaruhi pembacaan nilai kompas dalam kestabilan menemukan derajat melalui set point yang telah ditentukan. Hal ini akan mempengaruhi maping robot *micromouse* semakin stabil PID yang telah ditentukan terhadap nilai set point maka mempercepat dan membuat kestabilan robot dalam menemukan titik derajat yang telah ditentukan. Dan nilai PID yang mendapat nilai kestabilan yang baik terdapat pada data 1 terlihat dari cepatnya respon terhadap kestabilan nilai error yang mencapai sekitar 0 dan stabil terhadap *set point* derajat kompas yang telah ditentukan. Perbandingan hasil dari tiap-tiap data ditunjukkan pada gambar 6.




**Gambar** 6 Grafik komparasi kestabilan robot

## 5. KESIMPULAN

Penerapan algoritma PID terhadap pergerakan *micromouse* robot mempengaruhi pembacaan kompas untuk menemukan stabilitas titik tujuan yang telah ditentukan. Nilai *error* PID data 2 dan 3 terhadap pembacaan kompas memiliki nilai *error* yang sulit untuk menemukan titik stabil. Sehingga robot sulit untuk menemukan tujuan atau *set point degree* kompas yang telah ditentukan.

## 6. DAFTAR PUSTAKA

[1] Mutijarsah, Kusprasapta. 2011. Robot Dalam Ruang Kelas. http://robotika.unit.itb.ac.id/lama/3dokumen/NEW%20Presentasi%20URO%202011.pdf.

[2] Anita, Nur Syafidtri. 2010. Robot MicromouseDengan Menggunakan Algoritma Depth First Search. http://www.gunadarma.ac.id/library/articles/graduate/computerscience/2010/Artikel_21105199.pdf.

[3] Putra, Yansyah, dkk. 2015. Rancang Bangun Sistem Data Logger Pergerakan Sepeda Motor Berbasis Mikrokontroler Atmega 328P.http://snete.unsyiah.ac.id/2015/prosiding/Naskah%2013.pdf

[4] Nur Rifai, isnan dan Saka Gilab Asa, Panji. 2014. Penerapan Algoritma Kendali Proportional Integral Derivative pada Sistem Real Time Untuk Mempelajari TanggapanTransien.https://repository.ugm.ac.id/135075/1/sentia%20isnan.pdf

[5] Ali, Muhammad. 2004. Pembelajaran Perancangan Sistem Kontrol Pid Dengan Software matlab. http://staffnew.uny.ac.id/upload/132256208/penelitian/Sistem+Kontrol+PID+Muhamad+Ali.pdf