磁気コンパスの試作　～データ送信の無線化　～液晶表示　　PIC16F1827　完成版

2015.1.30　倒立振子デモ
2015.1.22　倒立振子，グラフィッデモ
2014.12.18 グラフィックデモ(脈拍計)
2014.12.04 本件紹介，グラフィックデモ
　　マンデルブロ
2014.11.xx 技術交流開始

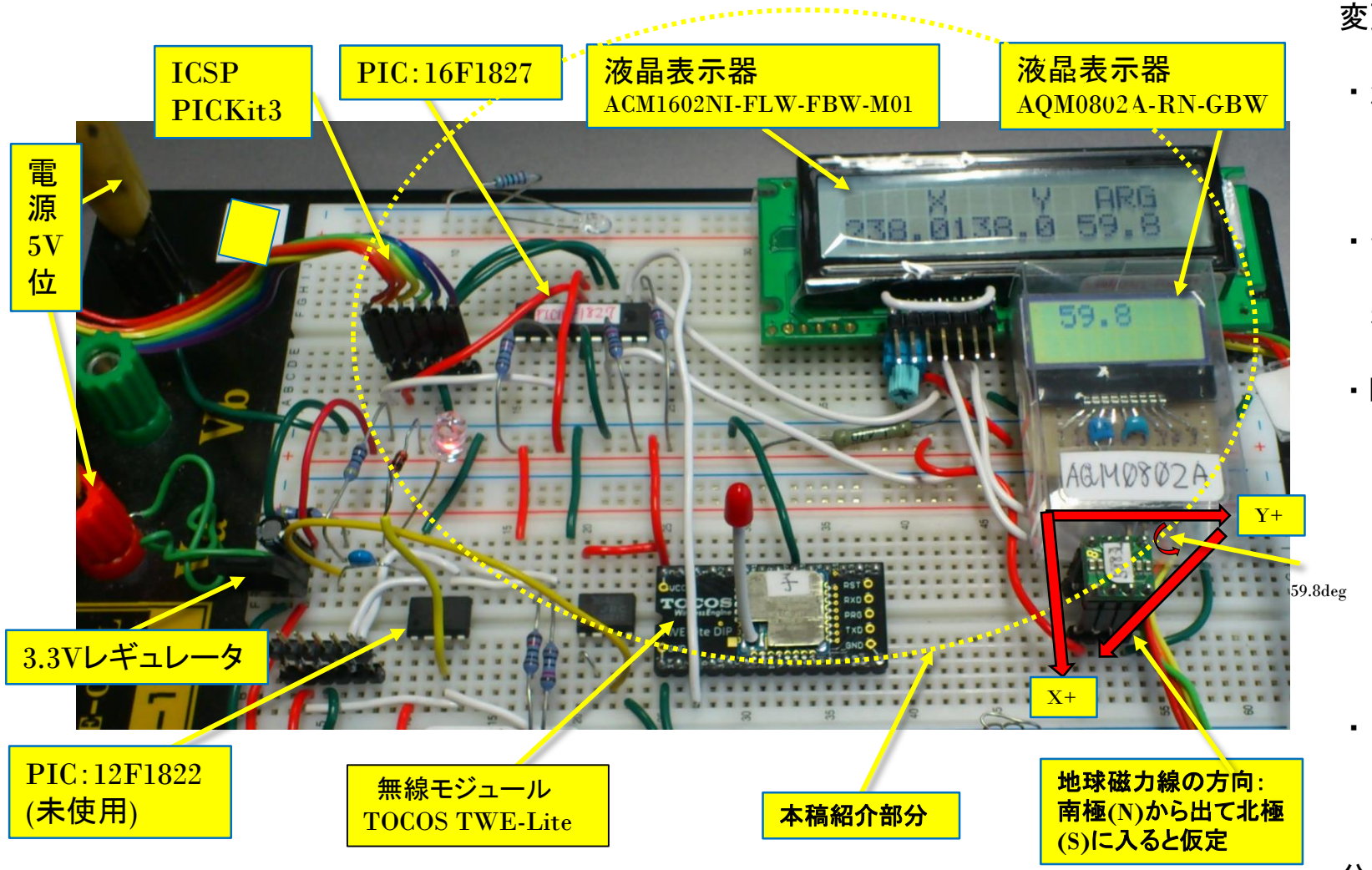

図1　磁気コンパス概要

変更点：2015. 1. 23

・液晶表示器を使用（試験のため2台使用）
　測定値を基板上で確認できる．
　小型器は即席な値確認に利用予定．

・センサーと無線モジュール、PICの電源を
　3. 3Vに統一した．
　利点：レベルコンバーターなどの回路が不要

・PICをPIC16F876からPIC16F1827に変更した．
　利点：低電圧(1.8V)から動作開始する．
　I2Cポート×2，USART×1ポートは，何かと便利．I2Cは液晶ディスプレイと磁気センサーに，USARTは無線モジュールに繋いだので，遠隔測定が可能となった．
　但し，PICKit2 は対応していないので，プログラミングできない．⇒PICKit3　ならば可能

・ブレッドボードを採用．
　永久性は無いけど，インスタントに体験するには最高．

分解能：
・コンパスは　0.1度の分解能を得た．
　センサーの13Bit幅とatan()計算精度に依存する．
温度測定はコーディングしていない．

PIC：12F1827

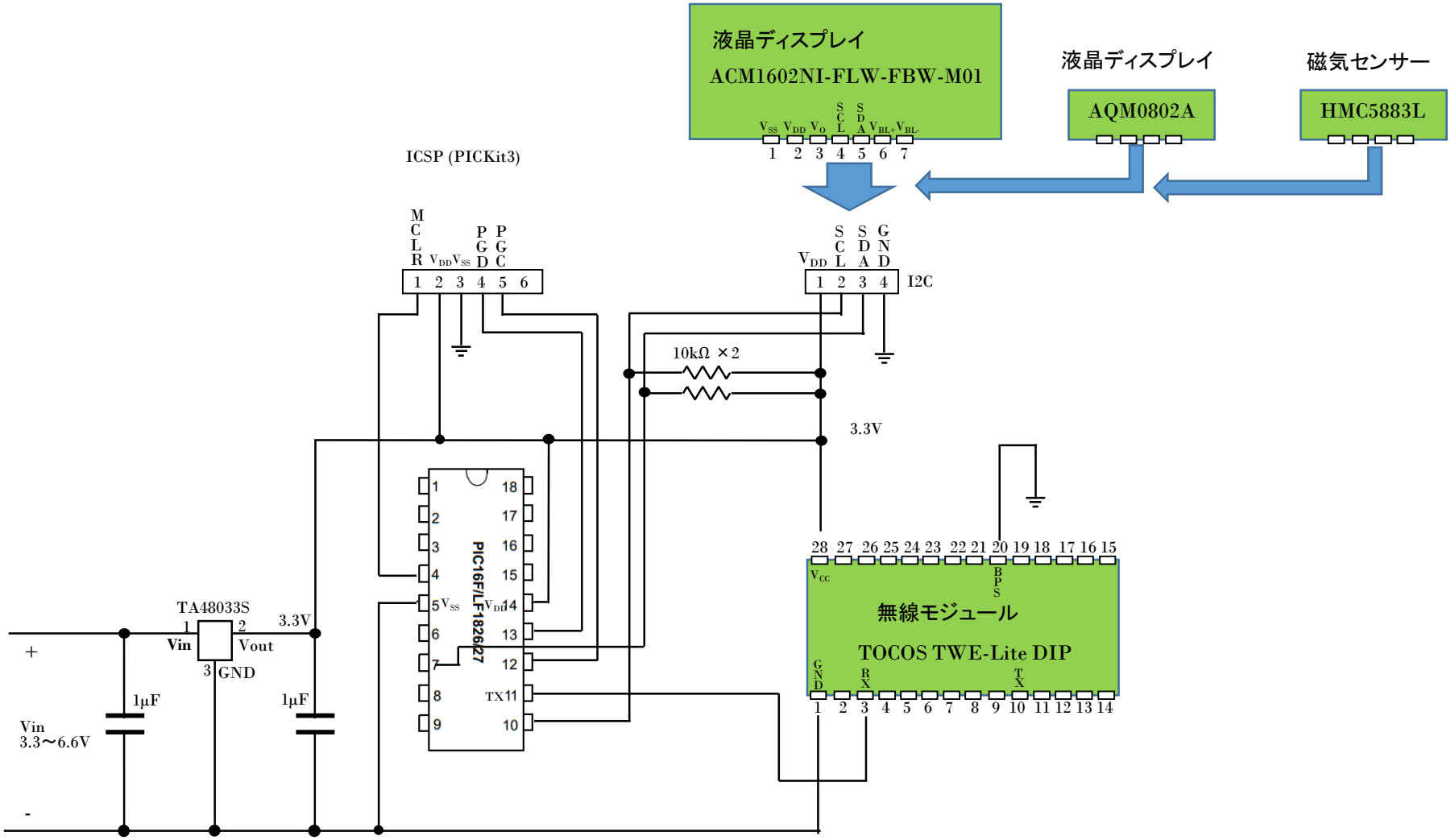

図2　回路

```
// Test of Magnetic Compass HMC5883L 2015.1.1
// Compass ===> LCD ACM1602N1-FLW-FBW M01 Akizuki
// Temperature ===> AQM0802A            Akizuki
// Send to PC by RS232C And TWE-Lite     M.T
#include<16F1827.h>
#include<math.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,NOMCLR,BROWNOUT        //
#use delay(clock=8000000)
#use i2c(MASTER,SDA=PIN_B1,SCL=PIN_B4,FAST,NOFORCE_SW)
#use RS232(PARITY=N, BAUD=38400,XMIT=PIN_B5, RCV=PIN_B1)

// i2c slave addresses
#define HMC5883L_WRT_ADDR   0x3C
#define HMC5883L_READ_ADDR  0x3D
// ACM1602
#define LCD_ADD          0xA0   // ACM1602 Slave Address
// AQM0802A
#define LCD_ADD_AQM       0x7c   // AQM0802 Slave Address
// AQM W/R Mode val
#define LCD_CMD_AQM       0x80   // Instruction Write Mode
#define LCD_DAT_AQM       0xC0   // Data Write Mode
#define line_1_AQM        0x00   // first line
#define line_2_AQM        0xC0   // second line = 0x80 + 0x40

// Register addresses
#define HMC5883L_CFG_A_REG  0x00
#define HMC5883L_CFG_B_REG  0x01
#define HMC5883L_MODE_REG   0x02
#define HMC5883L_X_MSB_REG  0x03

// mtsw 21014.11.11
#define HMC5883L_STR      0x09   //Status Register A
#define HMC5883L_IRA      0x0A   //Identification Register A = 0x48
#define HMC5883L_IRB      0x0B   //Identification Register B = 0x34
#define HMC5883L_IRC      0x0C   //Identification Register C = 0x33

void LCD_cmd(unsigned char cmd);
void LCD_Clear();
void LCD_Setline(unsigned char line);
void LCD_init();
void LCD_data(unsigned char data);
void LCD_space(int8 n);

void LCD_com_AQM(unsigned char cmd);
void LCD_Clear_AQM();
void LCD_Setline_AQM(unsigned char line);
void LCD_init_AQM();
void LCD_data_AQM(unsigned char data);
void LCD_space_AQM(int8 n);

#define line_1          0x80
#define line_2          0xC0
#define data_out         0x80
#define cmd_code         0x00

static int16 d1,d2,d3,d4,d5;

void hmc5883l_write_reg(int8 reg, int8 data);
int8 hmc5883l_read_reg (int8 reg);
void hmc5883l_read_data();

void disp_val(int16 x, int8 mod);

//------------------------------
typedef struct{
   int16 x;
   int16 y;
   int16 z;
}hmc5883l_result;

// This global structure holds the values read
// from the HMC5883L x,y,z registers.

hmc5883l_result compass = {0,0,0};
//  ======================================
void main(){
   double xx, yy, zz, deg;
   int16  degree, x, y, z;
   char cnt;

   LCD_init_AQM();
   LCD_Clear_AQM();       // Infuluence to ACM1602
   LCD_Setline_AQM(line_1_AQM);
   LCD_space_AQM(5);
   LCD_data_AQM('V');
   LCD_data_AQM('a');
   LCD_data_AQM('r');
   delay_ms(1000);

   LCD_init();
   LCD_Setline(line_1);
   cnt = 0;
   while(cnt < 16){
      LCD_data(0x30 + cnt++);
   }
   delay_ms(1000);
   LCD_Setline(line_2);
   cnt = 0;
   while(cnt < 16){
      LCD_data(0x40 + cnt++);
   }
   delay_ms(1000);
   LCD_Clear();
   LCD_Setline(line_1);

   LCD_space(4);  LCD_data('Y');
   LCD_space(2);
   LCD_data('V'); LCD_data('a'); LCD_data('r');
   delay_ms(1000);

// Meas & Display
   while(1) {

      LCD_Setline_AQM(line_2_AQM);
      LCD_data_AQM(' ');LCD_data_AQM(' ');

      hmc5883l_write_reg(HMC5883L_CFG_A_REG, 0x70);   //0x70 CCS=0x70
      hmc5883l_write_reg(HMC5883L_CFG_B_REG, 0x80);   //0xA0 CCS=0xA0
      hmc5883l_write_reg(HMC5883L_MODE_REG,  0x00);   //0x00 CCS=0x00
      hmc5883l_read_data();

      x = (compass.x ^ 0xFFFF);      // Reverse Sign
      y = (compass.y ^ 0xFFFF);      // Reverse Sign
      zz = z =  compass.z;
```

```
if( x & 0x8000 ){
      x *= -1;
      if( y & 0x8000){
        y *= -1;
        xx = x;  yy = y;
        deg = 57.30*ATAN(yy/xx) + 90;
      }
      else{
        xx = x;  yy = y;
        deg = 57.30*ATAN(xx/yy);
      }
    }
    else{
      if( x > 0){
        if( y & 0x8000){
          y *= -1;
          xx = x;  yy = y;
          deg = 57.30*ATAN(xx/yy) + 180;
        }
        else{
          xx = x;  yy = y;
          deg = 57.30*ATAN(yy/xx) + 270;
        }
      }
    }
    xx *= 10;
    yy *= 10;
    zz *= 10;
    degree = deg * 10;

    LCD_Setline(line_1);
    LCD_data('X');LCD_data(' ');
    printf("¥rX ");
    disp_val(xx,0);
    LCD_space(5);
    LCD_data('V');LCD_data('a');LCD_data('r');

    LCD_Setline(line_2);
    LCD_data('Y');LCD_data(' ');
    printf("  Y");
    disp_val(yy,0);
    printf("  Z ");
        disp_val(zz,2);        // mod = 2 >>> EUSART Only

        LCD_space(2);
        printf(" Var ");
        disp_val(degree,0);    // mod = 0 >>> ACM1602
        disp_val(degree,1);    // mod = 1 >>> AQM0802

        delay_ms(30);
      }
}
// ACM1602NI-FLW-FBW-M01
void LCD_cmd(unsigned char cmd){
    int16 u_t=10;
    i2c_start();       delay_ms(10);
    i2c_write(LCD_ADD); delay_us(u_t);    // Slave Address
    i2c_write(cmd_code);delay_us(u_t);    // Command Mode
    i2c_write(cmd);    delay_us(u_t);    // Send Command
    i2c_stop();        delay_ms(10);
}
void LCD_Clear(){
    LCD_cmd(0x01);
    delay_ms(3);
}
void LCD_Setline(unsigned char line) {
    LCD_cmd(line);
}
void LCD_init(){
    delay_ms(40);
    LCD_cmd(0x01);    //
    delay_ms(1);
    LCD_cmd(0x38);    //
    delay_ms(1);
    LCD_cmd(0x0C);    //
    delay_ms(1);
}
void LCD_data(unsigned char data){
    int16 u_t=100;
    i2c_start();       delay_ms(1);
    i2c_write(LCD_ADD); delay_us(u_t);    // Slave Address
    i2c_write(data_out);delay_us(u_t);    // Data Out Code
    i2c_write(data);   delay_us(u_t);    // Send Command
    i2c_stop();        delay_ms(1);
}
void LCD_space(int8 n){
    int8 cnt;
    cnt = 0;
    while(cnt++ < n){
      LCD_data(' ');
    }
}

// AQM0802 //
void LCD_com_AQM(unsigned char cmd){
    int16 u_t=50;
    i2c_start();          delay_ms(5);
    i2c_write(LCD_ADD_AQM); delay_us(u_t);    // Slave Address
    i2c_write(LCD_CMD_AQM);delay_us(u_t);     // Command Mode
    i2c_write(cmd);       delay_us(u_t);    // Send Command
    i2c_stop();           delay_ms(5);
}
void LCD_Clear_AQM(){
    LCD_com_AQM(0x01);
    delay_ms(20);
}
void LCD_Setline_AQM(unsigned char line) {
    LCD_com_AQM(line);
}
void LCD_init_AQM(){
    delay_ms(60);
    LCD_com_AQM(0x38);     // Function set
    delay_us(30);
    LCD_com_AQM(0x39);     // Function set
    delay_us(30);
    LCD_com_AQM(0x14);     // Internal OSC frequency
    delay_us(30);
    LCD_com_AQM(0x70);     // Contrast
    delay_us(30);
```

```
 LCD_com_AQM(0x56);     // Power/ICON/Contrast Control
   delay_us(30);
   LCD_com_AQM(0x6c);     // Follow Control
   delay_ms(300);
   LCD_com_AQM(0x38);     // Function set
   delay_us(30);
   LCD_com_AQM(0x0c);     // Display ON/OFF Control
   delay_us(30);
   LCD_com_AQM(0x01);     // Clear Display
   delay_ms(2);
}
void LCD_data_AQM(unsigned char data){
   int16 u_t=100;

   i2c_start();       delay_ms(5);
   i2c_write(LCD_ADD_AQM); delay_us(u_t);    // Slave Address
   i2c_write(LCD_DAT_AQM);delay_us(u_t);     // Data Out Code
   i2c_write(data);    delay_us(u_t);        // Send Command
   i2c_stop();        delay_ms(5);
}
void LCD_space_AQM(int8 n){
   int8 cnt;
   cnt = 0;
   while(cnt++ < n){
      LCD_data_AQM(' ');
   }
}

//------------------------------
// Low level routines for HMC5883L
//------------------------------
void hmc5883l_write_reg(int8 reg, int8 data){
   i2c_start();
   i2c_write(HMC5883L_WRT_ADDR);
   i2c_write(reg);
   i2c_write(data);
   i2c_stop();
}
//------------------------------
int8 hmc5883l_read_reg(int8 reg){
   int8 retval;
   i2c_start();
   i2c_write(HMC5883L_WRT_ADDR);
   i2c_write(reg);
   i2c_stop();        // mtsw
   i2c_start();
   i2c_write(HMC5883L_READ_ADDR);
   retval = i2c_read(0);
   i2c_stop();
   return(retval);
}
//------------------------------
void hmc5883l_read_data(){
   int8 x_lsb;    int8 x_msb;
   int8 y_lsb;    int8 y_msb;
   int8 z_lsb;    int8 z_msb;

   i2c_start();
   i2c_write(HMC5883L_WRT_ADDR);
   i2c_write(HMC5883L_X_MSB_REG);  // Point to X-msb register = 0x03
   i2c_stop();

   delay_ms(1);               // mtsw
   i2c_start();
   i2c_write(HMC5883L_READ_ADDR);

   x_msb = i2c_read();
   x_lsb = i2c_read();

   z_msb = i2c_read();
   z_lsb = i2c_read();

   y_msb = i2c_read();
   y_lsb = i2c_read(0);

   i2c_stop();
// Combine high and low bytes into 16-bit values.
   compass.x = make16(x_msb, x_lsb);
   compass.y = make16(y_msb, y_lsb);
   compass.z = make16(z_msb, z_lsb);
}

void disp_val(int16 x, int8 mod){          // Disp Digits
   int8 sw;

   if( x == 0xFFFF){
      printf("Ox FFFF");
   }
   else{
      if(  x & 0x8000){          // Minus ?
         printf("-");
         x = ( x ^ 0xFFFF)+1;
      }
      else{
         printf(" ");
      }
      d5 = x % 10 + 0x30;
      x /= 10;
      d4 = x % 10 + 0x30;
      x /= 10;
      d3 = x % 10 + 0x30;
      x /= 10;
      d2 = x % 10 + 0x30;
      x /= 10;
      d1 = x % 10 + 0x30;

      sw = 0;
      if(d1 == 0x30){
         d1 = 0x20;
      }
      else{
         sw = 1;
      }
      if( (d2 == 0x30)&&(sw == 0)){
         d2 = 0x20;
      }
```

```
 else{
        sw = 1;
      }
      if((d3 == 0x30)&&(sw == 0)){
        d3 = 0x20;
      }
      else{
        sw = 1;
      }
      if((d4 == 0x30)&&(sw == 0)){
        d4 = 0x20;
      }

      if(mod == 2){
        printf("%1c%1c%1c%1c.%1c",d1,d2,d3,d4,d5);
      }
      else{
        if(mod == 1){
        LCD_data_AQM(d1); LCD_data_AQM(d2); LCD_data_AQM(d3);
        LCD_data_AQM(d4); LCD_data_AQM('.');LCD_data_AQM(d5);
        }
        else{
          printf("%1c%1c%1c%1c.%1c",d1,d2,d3,d4,d5);
          LCD_data(d1); LCD_data(d2); LCD_data(d3);
          LCD_data(d4); LCD_data('.');LCD_data(d5);
        }
      }
    }
}
```