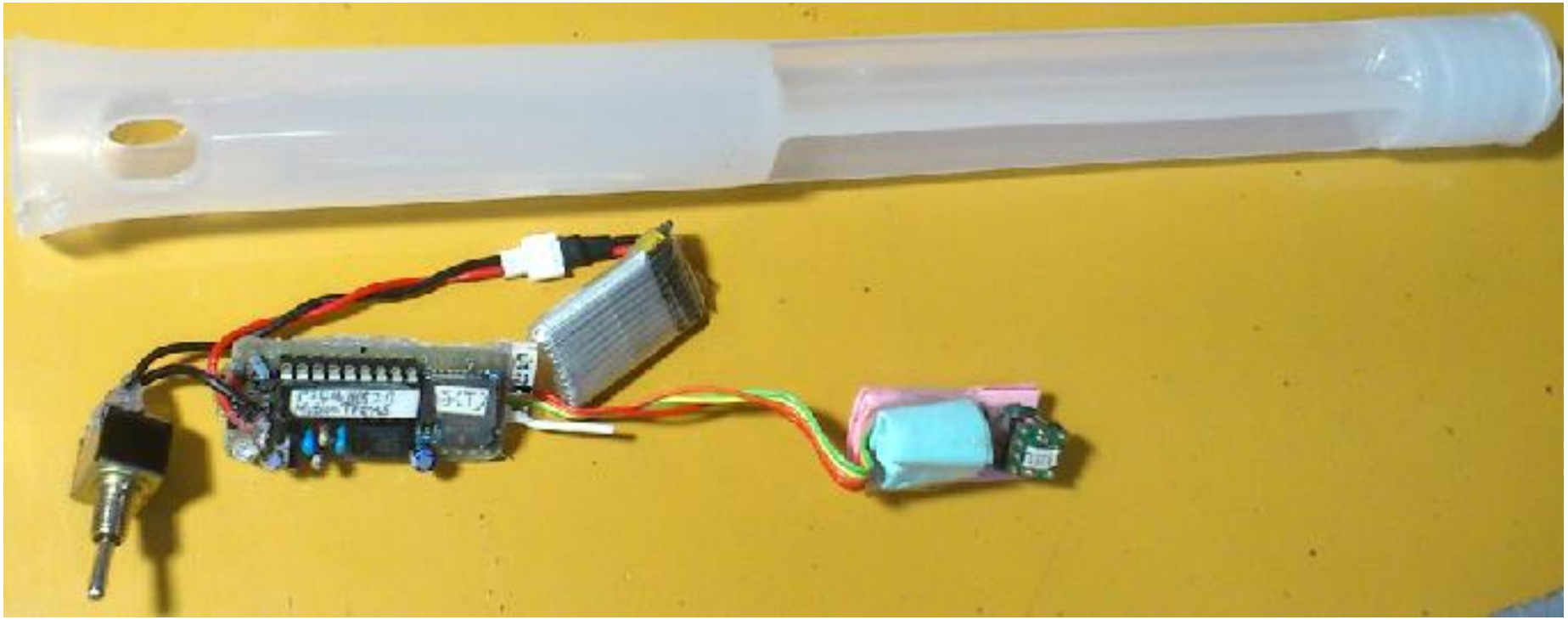
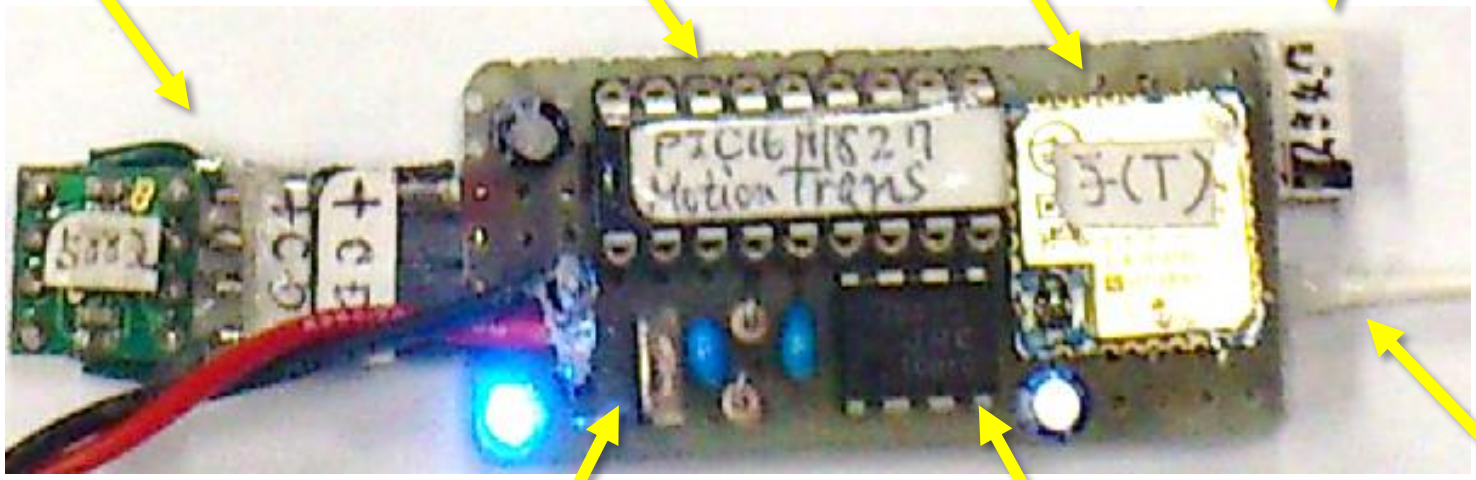
ワイヤレスモーションストランスミッタ　小型実装版

日用品の中に収納：
ケースは，100円の
柄杓（ひしゃく）



柄杓は，水を汲む部
分（合）を切り離し，
柄のみを利用．

ワイヤレスモーショントランスミッタ　小型実装版



ワイヤレスモーショントランスミッタ　小型実装版

TOCOS
Wireless Engine
TWE-001-Lite

図　無線モジュール

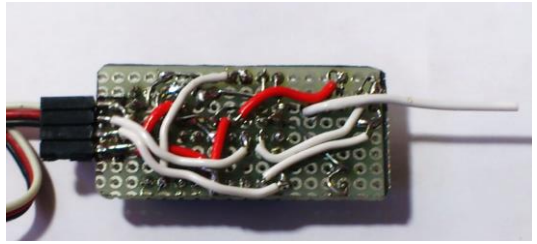


図　基板への実装（配線面）

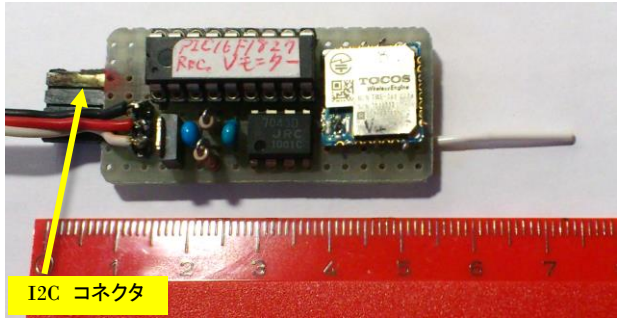
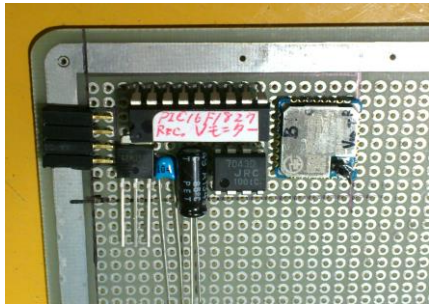


図　基板への実装（部品面）



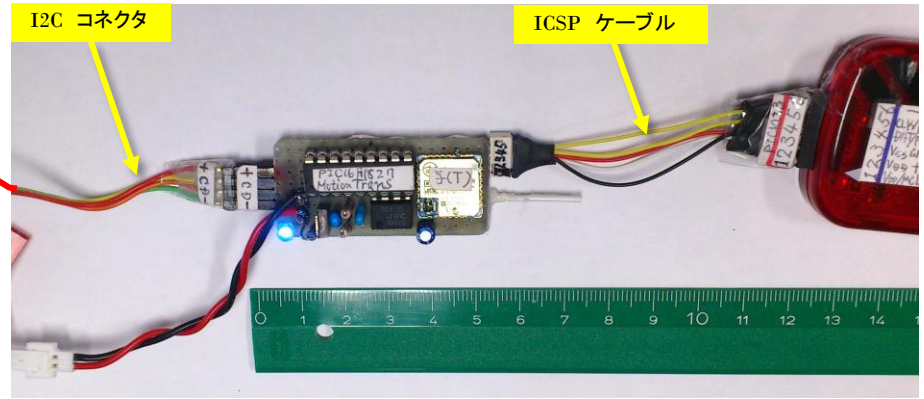
20～30年前の
基板なので半田の
“のり”が悪い.

図　使用基板（片面スルーホール）

I2Cコネクタを装備したのでLCDディスプレイを用い，現場で容易に動作確認できる.

図　熱収縮チューブで保護

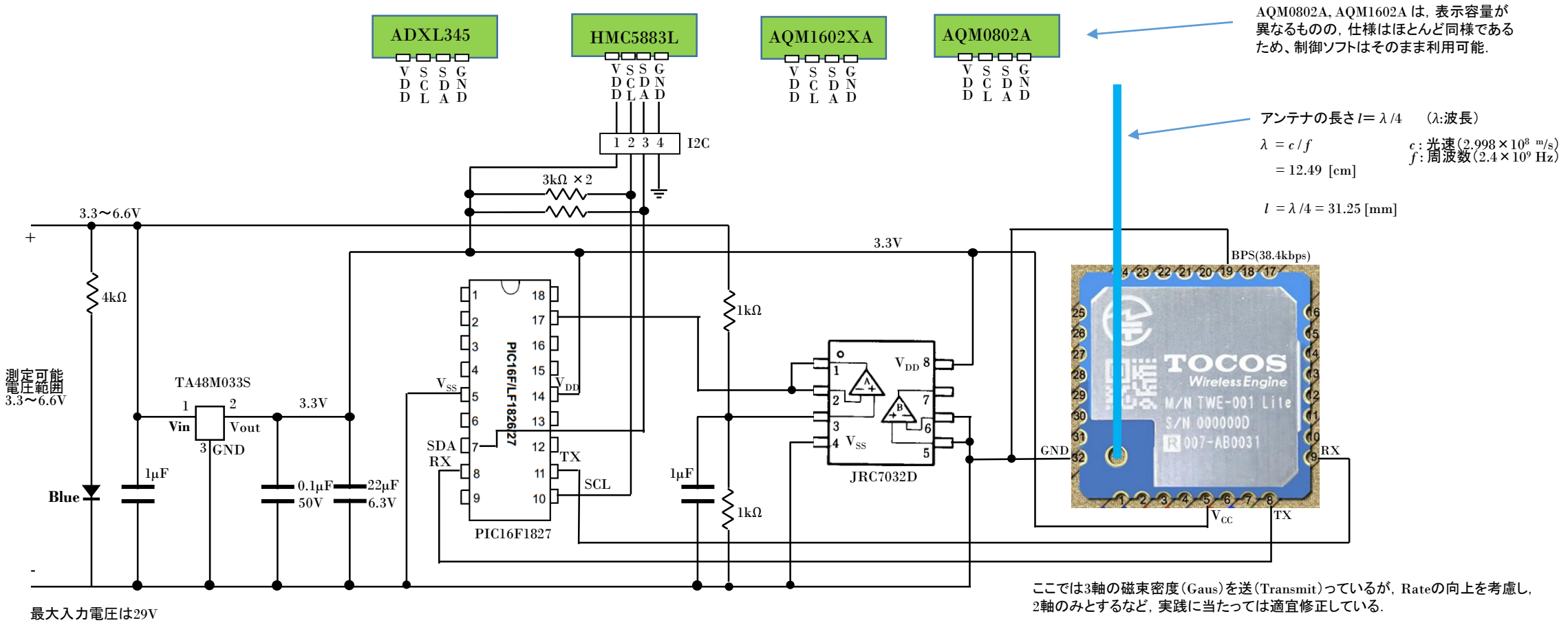
メモ：
I2Cコネクタを装備したので，磁気センサーや，温度，ジャイロ，気圧センサーなどを接続でき，応用範囲が広がった.
通信アプリ書き込み基板のトワイライタと受信用TWE-Lite DIP を含めて，費用はおよそ5500円であった．ただし，PIC開発費を除く.



メモ：2015.10.16(Fri.)
今回は、以下の様に修正した.

- **電源　→　1 cell Lipo 用ミニコネクタに変更**
- **ICSP：　*In Circuit Serial Programming***
  小型5Pコネクタ増設
- **3軸磁気センサーを用いて，個々の値を送信することとした.**
- **重力加速度を用いた手法も実験の予定.**

ワイヤレスモーショントランスミッタ　小型実装版



AQM0802A, AQM1602A は，表示容量が異なるものの，仕様はほとんど同様であるため、制御ソフトはそのまま利用可能．

アンテナの長さ $l = \lambda/4$　（λ:波長）

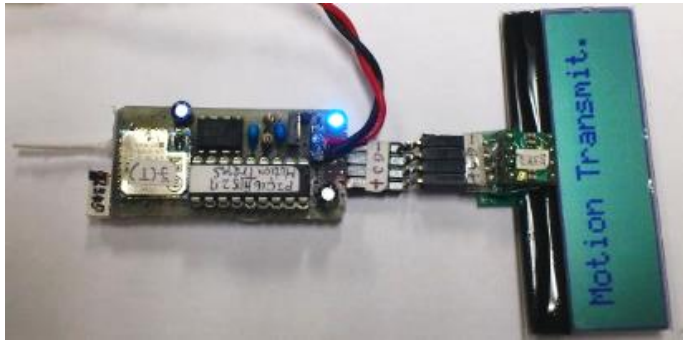
$\lambda = c/f$
$= 12.49$ [cm]

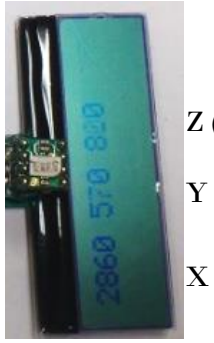
c: 光速（2.998×10^8 m/s）
f: 周波数（2.4×10^9 Hz）

$l = \lambda/4 = 31.25$ [mm]

最大入力電圧は29V

ここでは3軸の磁束密度（Gaus）を送（Transmit）っているが，Rateの向上を考慮し，2軸のみとするなど，実践に当たっては適宜修正している．

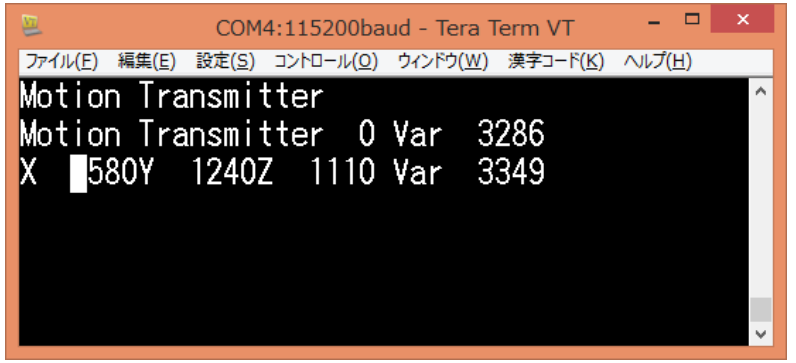




Z (gaus)

Y (gaus)

X (gaus)



ワイヤレスモーショントランスミッタ　送信プログラム

受信側の駆動用にラジコンサーボを想定している．そのため繰り返し周期（Rate）は15～16msとした．本プログラムはこのRateを得るためにdelay関数を用いてる．ちなみにインターナルタイマをも実験したが、目に見えては改善されない．これは，PICのclock(16MHz)とシリアルの速度38.4kbps　が関係している．このため外部タイマによる制御も実験の予定である．

```
// PIC16F1827 Motion Trans 2015.10.26,27 M.T
// Use I2C Port <<- HMC5883L, ADXL345,  AQM1602XA ...
// AQM0802A & AQM1602XA is command compatible
//  ===> AQM0802A      Akizuki @320
//  ===> AQM1602XA     Akizuki @550
// UART ===> TWE-Lite DIP

#include<16F1827.h>
#device ADC=10
#include<math.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,NOMCLR,BROWNOT    //
#use delay(clock=16000000)
#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_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);

////////////////////////////////////////////////
void ADT_init();
void ADT_data(unsigned char data);
void ADT_cmd(unsigned char data);
////////////////////////////////////////////////

static int16 d1,d2,d3,d4,d5;    // Effect only this position //

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(){
   int16  ADC;
   char title1[17] = "Motion Transmitt";
   double xx, yy, zz, deg;
   int16  degree, x, y, z;
   char cnt;

   delay_ms(1000);
   LCD_init_AQM();
   LCD_Clear_AQM();        // Infuluence to ACM160
   LCD_Setline_AQM(line_1_AQM);
   for(cnt=0 ; cnt < 16; cnt++){
      LCD_data_AQM( title1[cnt]);
   }

   printf("Motion Transmitt");
   setup_adc_ports(0x80);
   setup_adc(ADC_CLOCK_INTERNAL);
   delay_ms(2000);

   printf("%1c",0x0d);
   printf("                ");
   printf("%1c",0x0d);

   while(1){
      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();
      xx = x = (compass.x ^ 0xFFFF);      // Reverse Sign
      yy = 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;
            }
```

ワイヤレスモーショントランスミッタ　送信プログラム

```
else{
                xx = x;  yy = y;
                //deg = 57.30*ATAN(yy/xx) + 270;
            }
        }
    }

    x *= 10;
    y *= 10;
    z *= 10;
    //degree = deg * 10;
    //LCD_Clear_AQM();
    //LCD_Setline_AQM(line_1_AQM);
    //delay_ms(1);
    //printf("¥rX ");
    disp_val(x,0);
    printf(";");
    disp_val(y,0);
    printf(";");
    //disp_val(z,0);          // mod = 2 >>> EUSART Only
    //printf(":");
    printf("%1c",0x0d);

    //printf(" Var ");
    //disp_val(degree,1);     // mod = 0 >>> ACM1602
    //set_adc_channel(0);
    //delay_us(50);
    //ADC = 1.9995*3.265*read_adc();    // Small Type
    //ADC = 1.9995*2.9*read_adc(); // Develop.
    //ADC = 1.9995*3.25*read_adc();

    //LCD_Setline_AQM(line_2_AQM);
    //disp_val(ADC, 0);
    //printf("%1c",0x3b);  //delay_ms(50);// 0x3b == ';'  End of line
    //printf("%1c",0x0d);  //delay_ms(50);
    //LCD_data_AQM(" V");
    delay_ms(15);
  }
}
```

```
//---------------------------------
// 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);
}

// AQM0802 & AQM1602XA //
void LCD_com_AQM(unsigned char cmd){
    int16 u_t=20;
    i2c_start();            //delay_ms(1);
    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(1);
}
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);
}
```

ワイヤレスモーションストランスミッタ　送信プログラム

```
void LCD_data_AQM(unsigned char data){
    int16 u_t=26;

    i2c_start();            //delay_us(100);  //delay_ms(1);
    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_us(100);  //delay_ms(1);
}
void LCD_space_AQM(int8 n){
    int8 cnt;
    cnt = 0;
    while(cnt++ < n){
        LCD_data_AQM(' ');
    }
}

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

#ifdef ZERO
    if( x == 0xFFFF){
        printf("Ox FFFF");
    }
    else{
        if(  x & 0x8000){           // Minus ?
            printf("-");
            x = ( x ^ 0xFFFF)+1;
        }
        else{
            printf(" ");
        }
#endif

        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

#ifdef ZEROSUP
        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;
        }

#endif
        //printf("%1c",d1);     delay_ms(t_wait);
        printf("%1c",d2);     //delay_ms(t_wait);
        //printf("%1c",0x2e);  //delay_ms(t_wait); // 0x2c ==> '.'
        printf("%1c",d3);     //delay_ms(t_wait);
        printf("%1c",d4);     //delay_ms(t_wait);
        printf("%1c",d5);     //delay_ms(t_wait);
    }
```

ワイヤレスモーショントランスミッタ　送信プログラム　　タイマを用いたRate時間の生成

```
// PIC16F1827 Motion Trans 2015.10.26
// Use Internal Timer
// Use IIC Port <<- HMC5883L, ADXL345,  AQM1602XA ...
// AQM0802A & AQM1602XA is command compatible
//  ===> AQM0802A       Akizuki @320
//  ===> AQM1602XA      Akizuki @550
// UART ===> TWE-Lite DIP

#include<16F1827.h>
#device ADC=10
#include<math.h>
#fuses INTRC_IO,NOWDT,NOPROTECT,NOMCLR,BROWNOUT     //
#use delay(clock=16000000)
#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_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);

////////////////////////////////////////////////
void ADT_init();
void ADT_data(unsigned char data);
void ADT_cmd(unsigned char data);
////////////////////////////////////////////////
void disp_val(int16 x, int8 mod);

static int16 d1,d2,d3,d4,d5;    // Effect only this position //
static int16 icount;
static int16 degree, x, y, z;

#define TIC_1ms 0xee
#INT_TIMER0
void intval(){
   icount++;
   if(icount>=18){
      disp_val(x,0);
      printf(";");
      disp_val(y,0);
      printf(";");
      disp_val(z,0);         // mod = 2 >>> EUSART Only
      printf(":");
      printf("%1c",0x0d);
      icount = 0;
   }
   set_timer0(TIC_1ms);          // 1ms
}

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(){
   int16  ADC;
   char title1[17] = "Motion Transmitt";
   double xx, yy, zz, deg;

   char cnt;

   icount = 0;
   setup_timer_0(RTCC_INTERNAL | RTCC_DIV_256);
   set_timer0(TIC_1ms);          // 1ms
   enable_interrupts(INT_TIMER0);
   enable_interrupts(GLOBAL);

   delay_ms(1000);
   LCD_init_AQM();
   LCD_Clear_AQM();        // Infuluence to ACM1602
   LCD_Setline_AQM(line_1_AQM);
   for(cnt=0 ; cnt < 16; cnt++){
      LCD_data_AQM( title1[cnt]);
   }

   printf("Motion Transmitter");
   setup_adc_ports(0x80);
   setup_adc(ADC_CLOCK_INTERNAL);
   delay_ms(2000);

   while(1){
      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();
```

ワイヤレスモーションストランスミッタ　送信プログラム　　タイマを用いたRate時間の生成

```
 xx = x = (compass.x ^ 0xFFFF);      // Reverse Sign
       yy = 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;
             }
          }
          x *= 10;
          y *= 10;
          z *= 10;
       }
    }
}

//------------------------------
// 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);
}

// AQM0802 & AQM1602XA //
void LCD_com_AQM(unsigned char cmd){
    //int16 u_t=20;
    i2c_start();           //delay_ms(1);
    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(1);
}
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=26;

    i2c_start();           //delay_us(100);  //delay_ms(1);
    i2c_write(LCD_ADD_AQM); //delay_us(u_t);      // Slave Address
    i2c_write(LCD_DAT_AQM); //delay_us(u_t);      // Data Out Code
```

ワイヤレスモーショントランスミッタ　送信プログラム　　タイマを用いたRate時間の生成

```
 i2c_write(data);        //delay_us(u_t);      // Send Command
    i2c_stop();             //delay_us(100);  //delay_ms(1);
}

//void LCD_space_AQM(int8 n){
//    int8 cnt;
//    cnt = 0;
//    while(cnt++ < n){
//        LCD_data_AQM(' ');
//    }
//}

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

#ifdef ZERO
    if( x == 0xFFFF){
        printf("Ox FFFF");
    }
    else{
        if(  x & 0x8000){           // Minus ?
            printf("-");
            x = ( x ^ 0xFFFF)+1;
        }
        else{
            printf(" ");
        }
#endif

        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;

#ifdef ZEROSUP
        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;
        }

#endif
        //printf("%1c",d1);    delay_ms(t_wait);
        printf("%1c",d2);    //delay_ms(t_wait);
        //printf("%1c",0x2e);  //delay_ms(t_wait); // 0x2c ==> '.'
        printf("%1c",d3);    //delay_ms(t_wait);
        printf("%1c",d4);    //delay_ms(t_wait);
        printf("%1c",d5);    //delay_ms(t_wait);

        if(mod == 0){
            //LCD_data_AQM(d1);
            //LCD_data_AQM(d2);
            //LCD_data_AQM('.');
            //LCD_data_AQM(d3);
            //LCD_data_AQM(d4);
            //LCD_data_AQM(d5);
    }
}
```