```
//センサーの反応をパソコンで確認　　P1~2
#define sensor1 2
#define sensor2 17
#define sensor3 3
#define sensor4 18
int sensorin1;
int sensorin2;
int sensorin3;
int sensorin4;
//A0～A5→D14～D19 変換可能
volatile int sens1;
volatile int sens2;


void setup() {
  pinMode(sensor1, INPUT_PULLUP);
  pinMode(sensor2, INPUT_PULLUP);
  pinMode(sensor3, INPUT_PULLUP);
  pinMode(sensor4, INPUT_PULLUP);
  attachInterrupt(0, Sensor1, CHANGE);
  attachInterrupt(1, Sensor2, CHANGE);
  Serial.begin(9600);

}

void loop() {
  serial();
}

void Sensor1() {
  if (digitalRead(sensor1) == HIGH) {
    sens1 = 1;
  }
  if (digitalRead(sensor1) == LOW) {
    sens1 = 0;
  }
}
void Sensor2() {
  if (digitalRead(sensor3) == HIGH) {
    sens2 = 1;
```

```
  }
  if (digitalRead(sensor3) == LOW) {
    sens2 = 0;
  }

}
void serial() {
  Serial.print("sensor1");
  Serial.print(" ");
  Serial.print(sens1);
  Serial.print(" ");
  Serial.print("sensor3");
  Serial.print(" ");
  Serial.println(sens2);
}
```

```
//IN と OUT のタイム分秒ミリ秒とパターンを表示 P3
void printer() {
  Serial.print("IN");
  Serial.print(" ");
  Serial.print(Icntm);
  Serial.print(" ");
  Serial.print(Icnts);
  Serial.print(" ");
  Serial.print(Itime2);
  Serial.print(" ");
  Serial.print("OUT");
  Serial.print(" ");
  Serial.print(Ocntm);
  Serial.print(" ");
  Serial.print(Ocnts);
  Serial.print(" ");
  Serial.print(Otime2);
  Serial.print(" ");
  Serial.print("INTimepattern");
  Serial.print(" ");
  Serial.print(INTimepattern);
  Serial.print(" ");
  Serial.print("pattern");
  Serial.print(" ");
  Serial.print(pattern);
  Serial.print(" ");
  Serial.println(IcntIN);
}
```

```
//timer2 サンプル　P4～12
//Sample using LiquidCrystal library
#include <LiquidCrystal.h>
#include <Boards.h>
#include <Firmata.h>

/*select the pins used on the LCD panel
      lcd の使っているピン番号
   LiquidCrystal(rs, enable, d4, d5, d6, d7)
   rs: LCD の RS ピンに接続する Arduino 側のピン番号
   rw: LCD の RW ピンに接続する Arduino 側のピン番号
   enable: LCD の enable ピンに接続する Arduino 側のピン番号
   d0～d7: LCD の data ピンに接続する Arduino 側のピン番号

   d0～d3 はオプションで、省略すると 4 本のデータライン(d4～d7)だけで制御します。 */
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

// define some values used by the panel and buttons
int pushbutton = 0;

//パターン
int INTimepattern = 4;
int OUTTimepattern = 4;
int pattern = 0;
/*I = IN
   O = OUT
   cnts = second
   cntm = minute
   分秒表示*/
int Icnts = 0;
int Icntm = 0;
int Ocnts = 0;
int Ocntm = 0;

//時間計算
unsigned long Itimemillis = 0;
unsigned long Otimemillis = 0;
unsigned long Itime1 = 0;
unsigned long Itime2 = 0;
unsigned long Itime3 = 0;
```

```
unsigned long Otime1 = 0;
unsigned long Otime2 = 0;
unsigned long Otime3 = 0;

//時間
unsigned long millis();

//時間その他
int IcntIN = 0;
int OcntOUT = 0;

//proccesing 変数宣言
int outTime_m = 0; // OUT コース:分
int outTime_s = 0; // OUT コース:秒
int outTime_c = 0; // OUT コース:センチ秒
int inTime_m = 0;   // IN コース:分
int inTime_s = 0;   // IN コース:秒
int inTime_c = 0;   // IN コース:センチ秒

//タイマー表示 2 周目 3 周目
int secondOTIME = 0;
int thirdOTIME = 0;
int secondITIME = 0;
int thirdITIME = 0;
int secondOcntM = 0;
int secondOcntS = 0;
int thirdOcntM = 0;
int thirdOcntS = 0;
int secondIcntM = 0;
int secondIcntS = 0;
int thirdIcntM = 0;
int thirdIcntS = 0;

//定義
#define LEFT     0
#define UP       1
#define DOWN     2
#define RIGHT    3
#define SELECT 4
#define NONE     5
```

```
//LCD キーパッドシールドスイッチ入力
#define analogswitch A0

// read the buttons
int button() {
  pushbutton = (analogRead(analogswitch) / 4);
  if (pushbutton >= 240) return NONE;//240
  if (pushbutton < 20) return   RIGHT;//20
  if (pushbutton < 70) return   UP;//70
  if (pushbutton < 120) return DOWN;//120
  if (pushbutton < 170) return LEFT;//170
  if (pushbutton < 240) return SELECT;//240
  //return NONE;
}


void timerIN() {
  pattern = button();
  switch (INTimepattern) {
    case 0:
      Itimemillis = millis();
      Itime2 = Itimemillis - Itime1;
      if (Itime2 > 999) {
        Itime1 = Itimemillis;
        Icnts += 1;
      }
      if (Icnts > 59) {
        Icnts = 0;
        Icntm += 1;
      }
      if (pattern == UP) {
        secondITIME = Itime2;
        secondIcntS = Icnts;
        secondIcntM = Icntm;
        inTime_c = secondITIME / 10;
        inTime_s = secondIcntS;
        inTime_m = secondIcntM;
      }
      IcntIN = 0;
```

```
      break;

    case 2:
      IcntIN++;
      if (pattern == LEFT) {
        IcntIN = 0;
        Itime2 = 0;
        INTimepattern = 4;
      }
      break;

    case 4:
      IcntIN++;
      Itimemillis = millis();
      Itime1 = Itimemillis;
      Itime2 = Itimemillis - Itime1;
      Icnts = 0;
      Icntm = 0;
      secondITIME = 0;
      secondIcntS = 0;
      secondIcntM = 0;
      inTime_c = secondITIME;
      inTime_s = secondIcntS;
      inTime_m = secondIcntM;
      if (IcntIN > 10 && pattern == DOWN) {
        Itime1 = Itimemillis;
        INTimepattern = 0;
      }
      break;
  }
}

void timerOUT() {
  pattern = button();
  switch (OUTTimepattern) {
    case 0:
      Otimemillis = millis();
      Otime2 = Otimemillis - Otime1;
      if (Otime2 > 999) {
        Otime1 = Otimemillis;
```

```
      Ocnts += 1;
    }
    if (Ocnts > 59) {
      Ocnts = 0;
      Ocntm += 1;
    }
    if (pattern == RIGHT) {
      secondOTIME = Otime2;
      secondOcntS = Ocnts;
      secondOcntM = Ocntm;
      outTime_c = secondOTIME / 10;
      outTime_s = secondOcntS;
      outTime_m = secondOcntM;
    }
    OcntOUT = 0;
    break;

  case 2:
    OcntOUT++;
    if (pattern == LEFT) {
      OcntOUT = 0;
      Otime2 = 0;
      OUTTimepattern = 4;
    }
    break;

  case 4:
    OcntOUT++;
    Otimemillis = millis();
    Otime1 = Otimemillis;
    Otime2 = Otimemillis - Otime1;
    Ocnts = 0;
    Ocntm = 0;
    secondOTIME = 0;
    secondOcntS = 0;
    secondOcntM = 0;
    outTime_c = secondOTIME;
    outTime_s = secondOcntS;
    outTime_m = secondOcntM;
    if (OcntOUT > 10 && pattern == DOWN) {
```

```
      Otime1 = Otimemillis;
      OUTTimepattern = 0;
    }
    break;
  }
}

void switcher() {
  pattern = button();   // read the buttons

  switch (pattern) {
    case LEFT:
      if (INTimepattern == 2) {
        Itimemillis = millis();
        Itime1 = Itimemillis;
        Itime2 = Itimemillis - Itime1;
        Icnts = 0;
        Icntm = 0;
      }
      if (OUTTimepattern == 2) {
        Otimemillis = millis();
        Otime1 = Otimemillis;
        Otime2 = Otimemillis - Otime1;
        Ocnts = 0;
        Ocntm = 0;
      }
      break;

    case UP:
      if (INTimepattern < 1) {
        INTimepattern = 2;
      }
      /*      if (INTimepattern > 0) {
              INTimepattern = 0;
            }*/
      break;

    case RIGHT:
      if (OUTTimepattern < 1) {
        OUTTimepattern = 2;
```

```
        }
        break;

      case NONE:

        break;
    }
}

void LCD() {
   lcd.setCursor(5, 1);
   lcd.print(Icntm);
   lcd.print(" ");
   lcd.print(Icnts);
   lcd.print(" ");
   lcd.print(Itime2);
   lcd.print("     ");
   lcd.setCursor(5, 0);
   lcd.print(Ocntm);
   lcd.print(" ");
   lcd.print(Ocnts);
   lcd.print(" ");
   lcd.print(Otime2);
   lcd.print("     ");
}
void printer() {
   Serial.print("IN");
   Serial.print(" ");
   Serial.print(Icntm);
   Serial.print(" ");
   Serial.print(Icnts);
   Serial.print(" ");
   Serial.print(Itime2);
   Serial.print(" ");
   Serial.print("OUT");
   Serial.print(" ");
   Serial.print(Ocntm);
   Serial.print(" ");
   Serial.print(Ocnts);
   Serial.print(" ");
```

```
  Serial.print(Otime2);
  Serial.print(" ");
  Serial.print("INTimepattern");
  Serial.print(" ");
  Serial.print(INTimepattern);
  Serial.print(" ");
  Serial.print("pattern");
  Serial.print(" ");
  Serial.print(pattern);
  Serial.print(" ");
  Serial.println(IcntIN);
}

void proccesing() {
  timerIN();
  timerOUT();
  Serial.print("H");        //  ヘッダ送信(先頭を示す文字)
  // Serial.write(highByte(outTime_c));// OUT コース;センチ秒データ送信
  // Serial.write(lowByte(outTime_c));// OUT コース;センチ秒データ送信
  // Serial.write(highByte(inTime_c)); // IN コースミリ秒データ送信
  // Serial.write(lowByte(inTime_c)); // IN コースミリ秒データ送信
  Serial.write(outTime_c);// OUT コース;センチ秒データ送信
  Serial.write(inTime_c);// IN コースミリ秒データ送信
  Serial.write(outTime_m);// OUT コース;分データ送信
  Serial.write(outTime_s);// OUT コース;秒データ送信
  Serial.write(inTime_m); // IN コース;分データ送信
  Serial.write(inTime_s); // IN コース;秒データ送信
  Serial.print('¥n');
}

void setup()
{
  Serial.begin(250000);
  lcd.begin(16, 2);                    // start the library
  lcd.setCursor(0, 0);
  lcd.print("OUT"); // print a simple message
  lcd.setCursor(0, 1);    // move to the begining of the second line
  lcd.print("IN");
}
```

```
void loop() {
  timerIN();
  timerOUT();
  switcher();
  LCD();
  //  printer();
  proccesing();
}
```

```
//Processing でタイマーを表示　P13～15　timer2 と併用可能
/*
 * MCR Gate System シリアル通信
 * Processing 側サンプル
 * 開発環境 Ver3.3
 */

// シリアルライブラリを取り入れる
import processing.serial.*;

// myPort（任意名）というインスタンスを用意
Serial myPort;

/*=================== 変数宣言 ===================*/
int x; //図形の X 座標の変数を用意
int NUM = 6; //センサーの数
int[] receivedData = new int[NUM];      //センサーの値を格納する配列
int secondIcntM = 0; // OUT コース:分
int secondIcntS = 0; // OUT コース:秒
int secondITIME = 0; // OUT コース:センチ秒
int inTime_m = 0;   // IN コース:分
int inTime_s = 0;   // IN コース:秒
int inTime_c = 0;   // IN コース:センチ秒

/*=================== 初期設定 ===================*/
void setup(){
  // 画面サイズ
  size(1080, 780);                    // 1024 x 768 だと下が切れる
  textFont(createFont("x", 15)); //  日本語を表示用。バージョンアップで不要？
//frameRate(10);                       // 1 秒間に 10 回 draw()実行（使うかもしれないので残す）

  // シリアルポートの設定
  // Arduino が接続されているシリアルポートにあわせて書き換える
  myPort = new Serial(this, "COM3", 250000);
}

/*================ 描画ループ ================*/
void draw(){

  // 受信バッファから描画用変数に格納
```

```
  secondIcntM = receivedData[2];
  secondIcntS = receivedData[3];
  secondITIME = receivedData[0];
  inTime_m = receivedData[4];
  inTime_s = receivedData[5];
  inTime_c = receivedData[1];

  // 分,秒,センチ秒のバラバラのデータを時計形式に結合
  String outTime = nf(secondIcntM, 2)+":"+nf(secondIcntS, 2)+":"+nf(secondITIME, 2);
  String inTime = nf(inTime_m, 2)+":"+nf(inTime_s, 2)+":"+nf(inTime_c, 2);
  println(outTime);            // デバッグ用に出力
  println(inTime);             // デバッグ用に出力

  background(255);              // 背景を指定色:白で塗りつぶす

  //===OUT コースタイム表示===
  textSize(90);                 // 文字サイズ
  fill(0, 0, 255);             // 文字色:BLUE
  text("OUT", 30, height*15/100); // 表示するテキスト, x 座標, y 座標

  textSize(220);                // 文字サイズ
  fill(0, 0, 255);             // 文字色:BLUE
  text(outTime, 80, height*40/100);



  //===IN コースタイム表示===
  textSize(90);                 // 文字サイズ
  fill(255, 0, 0);             // 文字色:RED
  text("IN", 30, height*65/100); // 表示するテキスト, x 座標, y 座標

  textSize(220);                // 文字サイズ
  fill(255, 0, 0);             // 文字色:RED
  text(inTime, 80, height*90/100);

}

void serialEvent(Serial p){
  if ( myPort.available() >= 7 ) {   // ヘッダ + 時間データ で合計 7 バイト
    if ( myPort.read() == 'H' ) {    // ヘッダ文字を見つけたところから読み取る
      receivedData[0] = myPort.read(); // OUT:分読み込み
```

```
        receivedData[1] = myPort.read(); // OUT:秒読み込み
        receivedData[2] = myPort.read(); // OUT:センチ秒読み込み
        receivedData[3] = myPort.read(); // IN:分読み込み
        receivedData[4] = myPort.read(); // IN:秒読み込み
        receivedData[5] = myPort.read(); // IN:センチ秒読み込み

        /*int high1 = myPort.read();
        int low1 = myPort.read();
        int high2 = myPort.read();
        int low2 = myPort.read();
        receivedData[0] = high1*256 + low1; // OUT:分読み込み
        receivedData[1] = high2*256 + low2; // OUT:秒読み込み
        receivedData[2] = myPort.read();// OUT:センチ秒読み込み
        receivedData[3] = myPort.read(); // IN:分読み込み
        receivedData[4] = myPort.read(); // IN:秒読み込み
        receivedData[5] = myPort.read(); // IN:センチ秒読み込み*/
    }
  }

}
```