

```

/*****
   スタートゲート制御基板用スケッチ
   LCD Keypad Shield 使用
   *****/

#include <LiquidCrystal.h> //LCD ライブラリ
#include <Boards.h> //LCD ライブラリ
#include <Servo.h> //サーボライブラリ

/*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
//timer2
int pushbutton;
//パターン
int INTimepattern = 4;
int OUTTimepattern = 4;
int pattern;
int gate1pattern = 0;
int gate2pattern = 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;
//センサーカウント
int sensorINcnt = 0;

```

```

int sensorOUTcnt = 0;
//時間計算
unsigned long Iitemillis = 0;
unsigned long Iitemicros = 0;
unsigned long Oitemillis = 0;
unsigned long Oitemicros = 0;
unsigned long Itime1 = 0;
int Itime2 = 0;
unsigned long Itime3 = 0;
unsigned long Otime1 = 0;
int Otime2 = 0;
unsigned long Otime3 = 0;
//時間
unsigned long micros();
//時間その他
int IcntIN = 0;
int OcntOUT = 0;
//定義
#define LEFT    0
#define UP      1
#define DOWN    2
#define RIGHT   3
#define SELECT  4
#define NONE    5

#define analogswitch A0

//sensor2 ゲート 1OUT ゲート 2IN
#define sensor1 2 //ゲート 1 回路が少ないほう(出口) 41
#define sensor2 A3 //ゲート 1 回路が多いほう(入口) 41D
#define sensor3 3 //ゲート 2 回路が少ないほう(出口) 41
#define sensor4 A4 //ゲート 2 回路が多いほう(入口) 41D
int sensorin1;
int sensorin2;
int sensorin3;
int sensorin4;
int INsensorpattern = 0;
int OUTsensorpattern = 0;
int Isensor = 0;
int Osensor = 0;

```

```
//A0~A5→D14~D19 変換可能
```

```
//servo2
```

```
Servo myservo1;
```

```
Servo myservo2;
```

```
//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;
```

```
// read the buttons
```

```
int button() {
```

```
    pushbutton = (analogRead(analogswitch) / 4);
```

```
    if (pushbutton > 240) return NONE;
```

```
    if (pushbutton < 10) return RIGHT;
```

```
    if (pushbutton < 60) return UP;
```

```
    if (pushbutton < 110) return DOWN;
```

```
    if (pushbutton < 170) return LEFT;
```

```
    if (pushbutton < 210) return SELECT;
```

```
    // return NONE;
```

```
}
```

```
void timerOUT() {  
  sensorOUTpattern();  
  pattern = button();  
  switch (OUTTimepattern) {  
    case 0:  
      //ミリ秒基準  
      Otimemillis = millis();  
      Otime2 = Otimemillis - Otime1;  
      if (Otime2 >= 1000) {  
        Otime1 = Otimemillis;  
        Ocnts += 1;  
      }  
      if (Ocnts >= 60) {  
        Ocnts = 0;  
        Ocntm += 1;  
      }  
      if (OUTsensorpattern == 31) {  
        secondOTIME = Otime2;  
        secondOcntS = Ocnts;  
        secondOcntM = Ocntm;  
        outTime_c = secondOTIME;  
        outTime_s = secondOcntS;  
        outTime_m = secondOcntM;  
      }  
      if (OUTsensorpattern == 60) {  
  
        OUTTimepattern = 2;  
      }  
      break;  
  
    case 2:  
  
      break;  
  
    case 4:  
      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;
    break;
}

}

void timerIN() {
    pattern = button();
    sensorINpattern();
    switch (INTimepattern) {
        case 0:
            Itimemillis = millis();
            Itime2 = Itimemillis - Itime1;
            if (Itime2 >= 1000) {
                Itime1 = Itimemillis;
                Icnts += 1;
            }
            if (Icnts >= 60) {
                Icnts = 0;
                Icntm += 1;
            }
            if (INsensorpattern == 31) {
                secondITIME = Itime2;
                secondIcntS = Icnts;
                secondIcntM = Icntm;
                inTime_c = secondITIME;
                inTime_s = secondIcntS;
                inTime_m = secondIcntM;
            }
            if (INsensorpattern == 60) {

                INTimepattern = 2;
            }
            break;

```

case 2:

break;

case 4:

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;

break;

}

}

void LCD() {

timerIN();

timerOUT();

lcd.setCursor(5, 1);

lcd.print(inTime\_m);

lcd.print(" ");

lcd.print(inTime\_s);

lcd.print(" ");

lcd.print(inTime\_c);

lcd.print(" ");

lcd.setCursor(5, 0);

lcd.print(outTime\_m);

lcd.print(" ");

lcd.print(outTime\_s);

lcd.print(" ");

lcd.print(outTime\_c);

lcd.print(" ");

}

```

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_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 sensoro() {
  { if (digitalRead(sensor1) == LOW) {
      sensorin1 = 1;
    }
    else {
      sensorin1 = 0;
    }
  }
}

```

```

void sensori() {
  { if (digitalRead(sensor3) == LOW) {
      sensorin3 = 1;
    }
    else {
      sensorin3 = 0;
    }
  }
}

```

```

void sensorOUT() {

```

```

sensoro();
if (sensorin1 == 0) {
    Osensor = 1;
}
else {
    Osensor = 0;
}
}

void sensorIN() {
    sensori();
    if (sensorin3 == 0 ) {
        Isensor = 1;
    }
    else {
        Isensor = 0;
    }
}

void sensorOUTpattern() {
    sensorOUT();
    pattern = button();
    switch (OUTsensorpattern) {
        case 0:
            sensorOUTcnt = 0;
            gate2pattern = 1;
            if ( pattern == 1) {
                gate2pattern = 0;
                OUTsensorpattern = 1;
            }
            break;

        case 1:
            sensorOUTcnt = 0;
            if ( pattern == 4 ) {
                gate2pattern = 1;
                OUTsensorpattern = 20;
            }
            break;

        case 10:

```



```

sensorOUTcnt = 0;
if (Osensor == 1) {
    OUTsensorpattern = 20;
}
break;

case 20://1 回目通過
sensorOUTcnt++;
OUTTimepattern = 0;
if (sensorOUTcnt >= 1700) {
    sensorOUTcnt = 0;
    OUTsensorpattern = 30;
}
break;

case 30://
if (pattern == 0) {
    sensorOUTcnt = 0;
    OUTsensorpattern = 70;
}
if (Osensor == 1) {
    sensorOUTcnt = 0;
    OUTsensorpattern = 31;//31 にすると 2 回目通過のタイムを PC に表示
}

break;

case 31:
sensorOUTcnt++;
if (sensorOUTcnt >= 1) {
    sensorOUTcnt = 0;
    OUTsensorpattern = 40;
}
break;

case 40://
sensorOUTcnt++;
if (sensorOUTcnt >= 1700) {
    sensorOUTcnt = 0;
    OUTsensorpattern = 30;
}

```

```
}  
break;  
  
case 50://ゴール後  
if (Osensor == 1 ) {  
    OUTTimepattern = 2;  
    sensorOUTcnt = 0;  
    OUTsensorpattern = 60;  
}  
break;  
  
case 60:  
    sensorOUTcnt++;  
    if (sensorOUTcnt >= 1) {  
        sensorOUTcnt = 0;  
        OUTsensorpattern = 61;  
    }  
    break;  
  
case 61:  
    if (pattern == 0) {  
        sensorOUTcnt = 0;  
  
        OUTsensorpattern = 70;  
    }  
    break;  
  
case 70:  
    OUTTimepattern = 2;  
    sensorOUTcnt++;  
    if (pattern == 4) {  
        sensorOUTcnt = 0;  
        OUTsensorpattern = 71;  
    }  
    break;  
  
case 71:  
    OUTTimepattern = 4;  
    sensorOUTcnt++;  
    if (sensorOUTcnt > 300) {
```

```

        gate2pattern = 0;
        sensorOUTcnt = 0;
        OUTsensorpattern = 1;
    }
    break;
}
}

```

```

void sensorINpattern() {
    sensorIN();
    pattern = button();
    switch (INsensorpattern) {
        case 0:
            sensorINcnt = 0;
            gate1pattern = 1;
            if ( pattern == 1) {
                gate1pattern = 0;
                INsensorpattern = 1;
            }
            break;

        case 1:
            sensorINcnt = 0;
            if ( pattern == 4 ) {
                gate1pattern = 1;
                INsensorpattern = 20;//10 にすればゲート通過から測定が可能
            }
            break;

        case 10:
            sensorINcnt = 0;
            if (Isensor == 1) {
                INsensorpattern = 20;
            }
            break;

        case 20://1 回目通過
            sensorINcnt++;
            INTimepattern = 0;
            if (sensorINcnt >= 1700) {

```

```
    sensorINcnt = 0;
    INsensorpattern = 30;
}
break;
```

case 30://通過前クラッシュしたらスイッチを押して通過判断

```
if ( pattern == 0 ) {
    sensorINcnt = 0;
    INsensorpattern = 70;
}
if (Isensor == 1 ) {
    sensorINcnt = 0;
    INsensorpattern = 31;
}
}
```

```
break;
```

case 31:

```
sensorINcnt++;
if (sensorINcnt >= 1) {
    sensorINcnt = 0;
    INsensorpattern = 40;
}
break;
```

case 40://3回目通過 (ゴール)

```
sensorINcnt++;
if (sensorINcnt > 1000 ) {
    sensorINcnt = 0;
    INsensorpattern = 30;
}
break;
```

case 50:

```
if (Isensor == 1 ) {
    INTimepattern = 2;
    sensorINcnt = 0;
    INsensorpattern = 60;
}
break;
```

```
case 60://ゴール後
    sensorINcnt++;
    if (sensorINcnt >= 1 ) {
        sensorINcnt = 0;
        INsensorpattern = 61;
    }
    break;
```

```
case 61:
    if (pattern == 0) {
        sensorINcnt = 0;

        INsensorpattern = 70;
    }
    break;
```

```
case 70:
    INTimepattern = 2;
    sensorINcnt++;
    if (pattern == 4) {
        sensorINcnt = 0;
        INsensorpattern = 71;
    }
    break;
```

```
case 71:
    INTimepattern = 4;
    sensorINcnt++;
    if (sensorINcnt > 300) {
        gate1pattern = 0;
        sensorINcnt = 0;
        INsensorpattern = 1;
    }
}
```

```
}
```

```
void servoOUT() {
```

```

sensorOUTpattern();
switch (gate2pattern) {
  case 0://閉じる
    myservo2.write(126);//126 で閉じる
    break;

  case 1:
    myservo2.write(39);//39 で開く
    break;
}
}

void servoIN() {
  sensorINpattern();
  switch (gate1pattern) {
    case 0:
      myservo1.write(124);//124 で閉じる
      break;

    case 1:
      myservo1.write(37);//37 で開く
      break;
  }
}

void setup() {
  Serial.begin(250000);
  pinMode(sensor1, INPUT_PULLUP);
  pinMode(sensor3, INPUT_PULLUP);
  attachInterrupt(0, sensori, FALLING );
  attachInterrupt(1, sensoro, FALLING );
  myservo1.attach(12);
  myservo2.attach(13);
  lcd.begin(16, 2);          // start the library
  lcd.setCursor(0, 0);
  lcd.print("1OUT"); // print a simple message
  lcd.setCursor(0, 1);    // move to the begining of the second line
  lcd.print("2IN");
}

```

```
void loop() {
  timerIN();
  timerOUT();
  servoIN();
  servoOUT();
  proccesing();
  LCD();
  timechange();
  sensortimekakunin();
  Serial.println(button());
  Serial.println(pushbutton);
}
```

```
void timeprinter() {
  timerIN();
  timerOUT();
  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(pattern);
  Serial.print(" ");
  Serial.println(IcntIN);
}
```

```
void sensorprinter() {
    sensori();
    sensoro();
    Serial.print("sensor1");
    Serial.print(sensorin1);
    Serial.print(" ");
    Serial.print("sensor2");
    Serial.print(sensorin2 );
    Serial.print(" ");
    Serial.print("sensor3");
    Serial.print(sensorin3 );
    Serial.print(" ");
    Serial.print("sensor4");
    Serial.println(sensorin4 );
}
```

```
void patternkakunin() {
    Serial.print(pattern);
    Serial.print(INTimepattern);
    Serial.println(OUTTimepattern);
}
```

```
void sensorkakunin() {
    sensoro();
    sensori();
    sensorIN();
    sensorOUT();
    Serial.print(sensorin1);
    Serial.print(sensorin2);
    Serial.print(sensorin3);
    Serial.print(sensorin4);
    Serial.print(INsensorpattern);
    Serial.println(OUTsensorpattern);
}
```

```
void sensortimekakunin() {
    sensorINpattern();
    sensorOUTpattern();
    Serial.print(Isensor);
    Serial.print(INTimepattern);
    Serial.print(INsensorpattern);
}
```



```
Serial.print(" ");
Serial.print(Osensor);
Serial.print(OUTTimepattern);
Serial.println(OUTsensorpattern);
}
void servokakunin() {
    servoIN();
    servoOUT();
    Serial.print(gate1pattern);
    Serial.println(gate2pattern);
}
void timechange() {
    timerIN();
    timerOUT();
    Serial.print(" ");
    Serial.print(outTime_c);
    Serial.print(" ");
    Serial.print(outTime_s);
    Serial.print(" ");
    Serial.print(outTime_m);
    Serial.print(" ");
    Serial.print(inTime_c);
    Serial.print(" ");
    Serial.print(inTime_s);
    Serial.print(" ");
    Serial.println(inTime_m);
}
```