

スタートゲートプログラム Arduino 側

スタートゲートプログラム Processing 側とセットでお使いください。

```
/*  
*****  
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 Itimemillis = 0;
unsigned long Itimemicros = 0;
unsigned long Otimemillis = 0;
unsigned long Otimemicros = 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 1IN ゲート 2OUT
#define sensor1 15 //ゲート 1 回路が少ないほう(出口)
#define sensor2 16 //ゲート 1 回路が多いほう(入口)
#define sensor3 17 //ゲート 2 回路が少ないほう(出口)
#define sensor4 18 //ゲート 2 回路が多いほう(入口)
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 = 12; // OUT コース:分
int outTime_s = 34; // OUT コース:秒
int outTime_c = 56; // OUT コース:センチ秒
int inTime_m = 01; // IN コース:分
int inTime_s = 23; // IN コース:秒
int inTime_c = 45; // IN コース:センチ秒

// read the buttons
int button() {
    pushbutton = (analogRead(analogswitch) / 4);
    if (pushbutton > 240) return NONE;
    if (pushbutton < 10) return RIGHT;
    if (pushbutton < 50) return UP;
    if (pushbutton < 100) return DOWN;
    if (pushbutton < 150) return LEFT;
    if (pushbutton < 200) 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;
}
OcntOUT = 0;
if (OUTsensorpattern == 5) {
    OUTTimepattern == 2;
}
break;

```

case 2:

```

if (pattern == 0) {
    OUTTimepattern = 4;
}
break;

```

case 4:

```

    Otimemillis = millis();
    Otime1 = Otimemillis;
    Otime2 = Otimemillis - Otime1;
    Ocnts = 0;
    Ocntm = 0;
    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 == 5) {  
    INTimepattern = 2;  
}
```

```
break;
```

```
case 2:
```

```
if (pattern == 0) {  
    INTimepattern = 4;  
}
```

```
break;
```

```
case 4:
```

```
Itimemillis = millis();  
Itime1 = Itimemillis;  
Itime2 = Itimemillis - Itime1;  
Icnts = 0;  
Icntm = 0;  
break;
```

```
}
```

```
}
```

```
void LCD() {  
    timerIN();  
    timerOUT();  
    lcd.setCursor(5, 1);  
    lcd.print(Ocntm);  
    lcd.print(" ");  
    lcd.print(Ocnts);  
    lcd.print(" ");  
    lcd.print(Otime2);  
    lcd.print("  ");  
    lcd.setCursor(5, 0);  
    lcd.print(Icntm);  
    lcd.print(" ");  
    lcd.print(Icnts);
```

```
lcd.print(" ");
lcd.print(Itime2);
lcd.print("  ");
}
```

```
void procesing() {
  timerIN();
  timerOUT();
  outTime_m = Icntm; // OUT コース:分
  outTime_s = Icnts; // OUT コース:秒
  outTime_c = Itime2; // OUT コース:ミリ秒
  inTime_m = Ocntm; // IN コース:分
  inTime_s = Ocnts; // IN コース:秒
  inTime_c = Otime2; // IN コース:ミリ秒
  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 sensori() {
  if (digitalRead(sensor1) == LOW) {
    sensorin1 = 1;
  }
  else {
    sensorin1 = 0;
  }
}
if (digitalRead(sensor2) == LOW) {
  sensorin2 = 1;
}
else {
  sensorin2 = 0;
}
```

```
}
```

```
}
```

```
void sensoro() {
```

```
{ if (digitalRead(sensor3) == LOW) {
```

```
    sensorin3 = 1;
```

```
}
```

```
else {
```

```
    sensorin3 = 0;
```

```
}
```

```
}
```

```
{ if (digitalRead(sensor4) == LOW) {
```

```
    sensorin4 = 1;
```

```
}
```

```
else {
```

```
    sensorin4 = 0;
```

```
}
```

```
}
```

```
}
```

```
void sensorOUT() {
```

```
    sensoro();
```

```
    if (sensorin4 == 0 && sensorin3 == 1) {
```

```
        Osensor = 1;
```

```
}
```

```
else {
```

```
    Osensor = 0;
```

```
}
```

```
}
```

```
void sensorIN() {
```

```
    sensori();
```

```
    if (sensorin2 == 1 && sensorin1 == 0) {
```

```
        Isensor = 1;
```

```
}
```

```
else {
```

```
    Isensor = 0;
```

```
}
```

```
}
```

```
void sensorOUTpattern() {
```

```

sensorOUT();
pattern = button();
switch (OUTsensorpattern) {
  case 0:
    sensorOUTcnt = 0;
    if ( pattern == 0 ) {
      gate2pattern = 1;
      OUTsensorpattern = 2;
    }
    break;

  case 1:
    sensorOUTcnt = 0;
    if (Osensor == 1) {
      OUTsensorpattern = 2;
    }
    break;
  case 2://1 回目通過
    sensorOUTcnt++;
    OUTTimepattern = 0;
    if (sensorOUTcnt > 1000) {
      sensorOUTcnt = 0;
      OUTsensorpattern = 3;
    }
    break;

  case 3://通過前クラッシュしたらスイッチを押して通過判断
    if (Osensor == 1 || pattern == 3) {
      sensorOUTcnt = 0;
      OUTsensorpattern = 4;
    }

    break;

  case 4://3 回目通過 (ゴール)
    sensorOUTcnt++;
    if (sensorOUTcnt > 500) {
      if (Osensor == 1 || pattern == 3) {
        sensorOUTcnt = 0;
        OUTTimepattern = 2;
      }
    }
  }
}

```



```
        OUTsensorpattern = 5;
    }
}
break;
```

```
case 5://ゴール後
    sensorOUTcnt++;
    if (sensorOUTcnt > 1000 && pattern == 0) {
        sensorOUTcnt = 0;
        gate2pattern = 0;
        OUTsensorpattern = 6;
    }
    break;
```

```
case 6:
    sensorOUTcnt++;
    if (sensorOUTcnt > 300) {
        OUTsensorpattern = 0;
    }
}
```

```
}
```

```
void sensorINpattern() {
    sensorIN();
    pattern = button();
    switch (INsensorpattern) {
        case 0:
            sensorINcnt = 0;
            if ( pattern == 0 ) {
                gate1pattern = 1;
                INsensorpattern = 2;//1 にすればゲート通過から測定が可能
            }
            break;
```

```
case 1:
    sensorINcnt = 0;
    if (Isensor == 1) {
        INsensorpattern = 2;
    }
    break;
```

case 2://1 回目通過

```
sensorINcnt++;  
INTimepattern = 0;  
if (sensorINcnt > 1000) {  
    sensorINcnt = 0;  
    INsensorpattern = 3;  
}  
break;
```

case 3://通過前クラッシュしたらスイッチを押して通過判断

```
if (Isensor == 1 || pattern == 4) {  
    sensorINcnt = 0;  
    INsensorpattern = 4;  
}  
  
break;
```

case 4://3 回目通過 (ゴール)

```
sensorINcnt++;  
if (sensorINcnt > 500) {  
    if (Isensor == 1 || pattern == 4) {  
        sensorINcnt = 0;  
        INTimepattern = 2;  
        INsensorpattern = 5;  
    }  
}  
break;
```

case 5://ゴール後

```
sensorINcnt++;  
if (sensorINcnt > 1000 && pattern == 0) {  
    sensorINcnt = 0;  
    gate1pattern = 0;  
    INsensorpattern = 6;  
}  
break;
```

case 6:

```
sensorINcnt++;
```

```

        if (sensorINcnt > 300) {
            INsensorpattern = 0;
        }
    }
}

void servoOUT() {
    sensorOUTpattern();
    switch (gate2pattern) {
        case 0:
            myservo2.write(90);
            break;

        case 1:
            myservo2.write(0);
            break;
    }
}

void servoIN() {
    sensorINpattern();
    switch (gate1pattern) {
        case 0:
            myservo1.write(90);
            break;

        case 1:
            myservo1.write(0);
            break;
    }
}

void setup() {
    Serial.begin(250000);
    pinMode(sensor1, INPUT_PULLUP);
    pinMode(sensor2, INPUT_PULLUP);
    pinMode(sensor3, INPUT_PULLUP);
    pinMode(sensor4, INPUT_PULLUP);
    myservo1.attach(3);
    myservo2.attach(11);
    lcd.begin(16, 2);           // start the library
}

```

```
lcd.setCursor(0, 0);  
lcd.print("2OUT"); // print a simple message  
lcd.setCursor(0, 1); // move to the beginning of the second line  
lcd.print("1IN");  
}
```

```
void loop() {  
  servoIN();  
  servoOUT();  
  timerIN(); //IN コース  
  timerOUT(); //OUT コース  
  proccesing(); //proccesing に送信  
  LCD(); //LCD に表示  
}
```

```
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);
}
```