```
// All Japan Micomcar Rally Robotrace Class TimeGate.
// 編集者　レッドインベーダー 2019.3.5
#include <MsTimer2.h>                  // タイマー割り込みを利用する為に必要なヘッダファイル
#include <LiquidCrystal.h>        // LCD 表示ライブラリ

/*=================== プロトタイプ宣言 ===================*/
#define SENSOR_IN            15 //A1
#define SENSOR_OUT           16 //A2
#define ON                   1
#define OFF                  0
#define INTERVAL_TIME    2000
#define btnRIGHT              0
#define btnUP                1
#define btnDOWN               2
#define btnLEFT              3
#define btnSELECT             4
#define btnNONE               5

/*=================== 変数宣言 ===================*/
char Pattern = 0;                  // 動作パターン

// variables will change:
int buttonState = 0;             // variable for reading the pushbutton status

//タイマ関連
int Time_m;              // 基準タイマ:分
int Time_s;              // 基準タイマ:秒
int Time_c;              // 基準タイマ:センチ秒
int outTime_m;          // OUT コース:分
int outTime_s;          // OUT コース:秒
int outTime_c;          // OUT コース:センチ秒
int outTime_m_old;          // OUT コース:分
int outTime_s_old;          // OUT コース:秒
int outTime_c_old;          // OUT コース:センチ秒

unsigned int IntervalTime; // IN コース車体通過待ちインターバルタイマ

LiquidCrystal lcd(8, 9, 4, 5, 6, 7); // select the pins used on the LCD panel
// define some values used by the panel and buttons
int lcd_key         = 0;
int adc_key_in    = 0;
unsigned long cnt = 0;
```

```
/*=================== 初期設定 ===================*/
void setup() {
  lcd.begin(16, 2);                 // start the lcd library
  Serial.begin(250000);             // set up Serial library at 250000 bps
  lcd.setCursor(0, 0);
  lcd.write("Time");
  lcd.setCursor(0, 1);
  lcd.write("OldTime");

  // initialize the pushbutton pin as an input:
  pinMode(SENSOR_IN, INPUT);
  pinMode(SENSOR_OUT, INPUT);

  //Timer2 割込設定
  MsTimer2::set(1, Timer2_Int);        // 1ms 毎に Timer2_Int()割込み関数を呼び出す様に設定
  MsTimer2::start();                    // タイマー割り込み開始　常にタイマー割り込みを行う
}

/*=================== メインループ ===================*/
void loop() {
  TimeGate();
  Processing();
}

/*************************************************************************
  タイムゲート処理
*************************************************************************/
void TimeGate() {
  buttonState = read_LCD_buttons();
  switch ( Pattern ) {
    case 0:
      lcd.setCursor(8, 0);
      lcd.print(outTime_m);
      lcd.print(":");
      lcd.print(outTime_s);
      lcd.print(".");
      lcd.print(outTime_c);
      lcd.print("    ");

      lcd.setCursor(8, 1);
      lcd.print("StandBye");
      /*lcd.print(outTime_m_old);
        lcd.print(":");
```

```
      lcd.print(outTime_s_old);
      lcd.print(".");
      lcd.print(outTime_c_old);
      lcd.print("     ");*/

    outTime_m_old = outTime_m = 0;
    outTime_s_old = outTime_s = 0;
    outTime_c_old = outTime_c = 0;

    if ((buttonState == btnLEFT) && (cnt >= 10)) {
      outTime_m_old = outTime_m = Time_m = 0;
      outTime_s_old = outTime_s = Time_s = 0;
      outTime_c_old = outTime_c = Time_c = 0;
      IntervalTime = 0;
      cnt = 0;
      Pattern = 10;
      break;
    }
    break;

  case 10://スタート待ち
    lcd.setCursor(8, 0);
    lcd.print(outTime_m);
    lcd.print(":");
    lcd.print(outTime_s);
    lcd.print(".");
    lcd.print(outTime_c);
    lcd.print("     ");

    lcd.setCursor(8, 1);
    lcd.print(outTime_m_old);
    lcd.print(":");
    lcd.print(outTime_s_old);
    lcd.print(".");
    lcd.print(outTime_c_old);
    lcd.print("     ");

    if ( (digitalRead(SENSOR_OUT) == ON) && (IntervalTime > INTERVAL_TIME) ) {
      outTime_m_old = outTime_m;
      outTime_s_old = outTime_s;
      outTime_c_old = outTime_c;
      outTime_m = Time_m;
      outTime_s = Time_s;
```

```
            outTime_c = Time_c;
            IntervalTime = 0;
        }
        if ( (digitalRead(SENSOR_IN) == ON) && (IntervalTime > INTERVAL_TIME) ) {
            Time_m = 0;
            Time_s = 0;
            Time_c = 0;
            outTime_m_old = outTime_m;
            outTime_s_old = outTime_s;
            outTime_c_old = outTime_c;
            outTime_m = Time_m;
            outTime_s = Time_s;
            outTime_c = Time_c;
            IntervalTime = 0;
        }
        if ((buttonState == btnRIGHT) && (cnt >= 10)) {
            cnt = 0;
            Pattern = 0;
            break;
        }
        break;

    default:
        break;
  }
}

/*********************************************************************
  Processing 処理
 *********************************************************************/
void Processing() {
  Serial.print("H");          //  ヘッダ送信(先頭を示す文字)
  Serial.write(outTime_c);// OUT コース;センチ秒データ送信
  Serial.write(outTime_s);// OUT コース;分データ送信
  Serial.write(outTime_m);// OUT コース;秒データ送信
  Serial.print('¥n');
}
/*********************************************************************
  TIMER2 割込処理
*********************************************************************/
void Timer2_Int()
{
  static int cnt10;
```

```
  IntervalTime++;
  cnt++;

  // 10ms 周期の処理
  cnt10++;
  if ( cnt10 >= 10 ) {
    Time_c++;
    if ( Time_c > 99 ) {
      Time_s++;
      Time_c = 0;
    }
    if ( Time_s > 59 ) {
      Time_m++;
      Time_s = 0;
    }
    if ( Time_m > 9 ) {
      Time_m = 0;
    }
    cnt10 = 0;
  }
}

/*********************************************************************
  LCD キーパットシールドボタン読み込み
*********************************************************************/
int read_LCD_buttons() {                    // read the buttons
  adc_key_in = analogRead(0);           // read the value from the sensor

  if (adc_key_in > 1000) return btnNONE;

  // For V1.1 us this threshold
  if (adc_key_in < 50)    return btnRIGHT;
  if (adc_key_in < 250)   return btnUP;
  if (adc_key_in < 450)   return btnDOWN;
  if (adc_key_in < 650)   return btnLEFT;
  if (adc_key_in < 850)   return btnSELECT;

  return btnNONE;                           // when all others fail, return this.
}
```