```
//
//
//   マイクロキャット3号
// ------------------------------------------
//   サーボと超音波距離センサで障害を避けて走るロボット
//   回転するサーボヘッドに取り付けたセンサで距離を測る
//    前方に障害物がなければ走行開始
//     Ver-1.0 (2016/6/25 S.S)
//
//     Ver-1.1  サーボの駆動をライブラリに変更(2016/6/27)
//

// ヘッド関連の定義
#include <Servo.h>

Servo headservo;  // 超音波センサ回転用サーボのインスタンスを開始
Servo Rservo;     // 駆動用サーボのインスタンスを開始
Servo Lservo;

#define H_SERV  7 // 頭サーボ
#define TRIGGER 2 // トリガ出力
#define ECHO    3 // 超音波エコー受信ポート
#define HD_DELAY  200  // ヘッド動作待ちディレイタイム
#define FORWARD  89  // 機体前方角度
#define RIGHT   30  // 右方角度
#define LEFT   150  // 左方角度
#define D_RIGHT   45  // 斜め右方角度
#define D_LEFT   135  // 斜め左方角度
#define EMARGENCY 999 // 非常事態
#define EOA       888  // 配列の終わり記号

// ドライブ関係の定義
#define R_SERV  9    // サーボ#1接続ポート(R)
#define L_SERV  8    // サーボ#2接続ポート(L)
#define LED 13        // 障害物検出表示LED
#define BARRIER   30    // 障害物までの距離
#define BARRIER1  10    // 障害物までの距離(至近距離)

// 回転方向の定義
#define CW  150
#define CCW 10
#define STOPR 89
#define STOPL 88

// 移動時間の定義
#define T_MOVE  500
//#define T_MOVE  1000
#define T_LEFT  1000     // 左へ90度回転
```

```
#define T_RIGHT 1000      // 右へ90度回転
#define DELAY   10
#define T_STOP  300
#define OK 0
#define NG 1


//
// 超音波センサーを使って距離を測る関数
//  戻り値の単位は『cm』
//
int  measure() {
  int interval;
  int distance;

  digitalWrite( TRIGGER, HIGH );  // トリガーパルスを出力
  delayMicroseconds( 100 );
  digitalWrite( TRIGGER, LOW );

  // mesure the interval
  interval = pulseIn( ECHO, HIGH ); //エコーを受信
  distance = interval * 0.017;      // cmに変換

  return (distance);
}

//
// サーボを停止する関数
//
void  mv_stop( int stop_time ) {
  //act = ACK;
  //digitalWrite( LED, LOW );
  Lservo.write(STOPL);
  //delay( DELAY );
  Rservo.write(STOPR);
  delay( stop_time );
}

//
// 前進関数
//
void mv_forward( int move_time ) {
  Lservo.write(CW);
  //delay(DELAY);
  Rservo.write(CCW);
  delay(move_time);
}
```

```
//
// 後進関数
//
void mv_back( int move_time ) {
  Lservo.write(CCW);
  //delay(DELAY);
  Rservo.write(CW);
  delay(move_time);
}

//
// 左回転関数
//
void mv_left( int move_time ) {
  Lservo.write(STOPL);
  //delay(DELAY);
  Rservo.write(CCW);
  delay(move_time);
}

//
// 右回転関数
//
void mv_right( int move_time ) {
  Lservo.write(CW);
  //delay(DELAY);
  Rservo.write(STOPR);
  delay(move_time);
}

//
// 障害物回避処理
//   入力条件は障害物までの距離
//   結果：OK/NG(障害物を回避できた場合は『OK』)
//
int mv_avoid( int d ) {
  int l, r , resp;

  digitalWrite( LED, HIGH );
  resp = NG;

  headservo.write(D_LEFT);     // 左を確認
  delay(HD_DELAY);
  l = measure();
  headservo.write(D_RIGHT);      // 右を確認
  delay(HD_DELAY);
  r = measure();
```

```
  if (( l > d ) || ( l >= r )) {              // 左方に障害物が無ければ左回転
    headservo.write(D_LEFT);
    mv_left( T_LEFT );
    digitalWrite( LED, LOW );
    resp = OK;
  } else if (( r > d ) || ( r >= l )) {        // 右方に障害物が無ければ右回転
    headservo.write(D_RIGHT);
    mv_right( T_RIGHT );
    digitalWrite( LED, LOW );
    resp = OK;
  } else if (( r > d ) || ( l >= r )) {        // 左方に障害物が無ければ左回転
    headservo.write(D_LEFT);
    mv_left( T_LEFT );
    digitalWrite( LED, LOW );
    resp = OK;
  } else if (( l > d ) || ( r >= l )) {        // 右方に障害物が無ければ右回転
    headservo.write(D_RIGHT);
    mv_right( T_RIGHT );
    digitalWrite( LED, LOW );
    resp = OK;
  }
  return resp;
}

void setup() {
  headservo.attach(H_SERV);
  Lservo.attach( L_SERV );
  Rservo.attach( R_SERV );
  pinMode( TRIGGER, OUTPUT );
  pinMode( ECHO, INPUT );
  randomSeed(analogRead(7));       // 乱数発生器を初期化
  //Serial.begin( 9600 );
  //MsTimer2::set(500, check_barrier);     // 500ms毎に距離を測定
  //MsTimer2::start();             // タイマー割り込み開始
}

void loop() {
  int d;

  //digitalWrite( LED, LOW );
  headservo.write(FORWARD);      // 前を向く
  delay(HD_DELAY);               // ヘッドが動作完了するのを待つ

  d = measure();                 // 前方を確認
  if ( d > BARRIER ) {           // 前方に障害物が無ければ前進
    digitalWrite( LED, LOW );
    mv_forward( T_MOVE );
```

```
  } else if ( d < BARRIER1 ) {    // 障害物発見
    digitalWrite( LED, HIGH );    // 障害物まで至近距離
    mv_stop( T_STOP );
    mv_back( T_MOVE );            // とりあえずバックして避ける
    mv_avoid( BARRIER1 );         // 再度回避処理

    // 障害物は見つけたが、未だ距離はあるので左右へ回避を試みる
  } else if ( mv_avoid( BARRIER ) == NG) {
    // 回避できなかった。。。3方向全部障害(袋小路)
    mv_stop( T_STOP );
    mv_back( T_MOVE );          // バックしてリトライ
    mv_avoid( BARRIER1 );       // 再度回避処理
    headservo.write(FORWARD);   // ヘッドは前に向けておく
    delay(HD_DELAY);
  }
}
```