Arduino の開発環境を使った教育用マイクロマウス

熊本高専　葉山清輝（2012.9）

1. はじめに

近年，初心者でも容易に始められるマイコン利用のボードおよび開発環境として Arduino[1-2]が知られている．そこで， Arduino の開発環境を用いてハードウェアおよびソフトウェア演習を行うことができる自立型移動ロボット（マイクロマウス）を設計した．

2. マイクロマウスとは

マイクロコンピュータや各種センサなどを内蔵し，自分で判断しながら行動するロボットを自律型ロボットという．マイクロマウスとは，マイクロマウス自分の力で迷路を探索し，ゴールへ到達することができる自律型移動ロボットである．マイクロマウス競技は，1977 年に IEEE（米国電気電子学会）が提唱したことに始まり，日本では，1980 年より財団法人ニューテクノロジー振興財団が主催で毎年地区大会および全国大会が行われている．

3. 競技の概要

迷路の大きさは、１６×１６区画の通路から構成されている．四隅のいずれかの区間で設定された出発点に置かれたマウスが、自分だけの力で中央にあるゴールにどのようにして早くたどりつくか、時間を競う競技である．持ち時間は１０分以内で決まられ，走行回数は５回で，ゴール到達に要した最短時間を競う．多くのマウスは，最初に比較的ゆっくりした走行で迷路の探索走行を行い，その後は最短コースを計算して最速で迷路を走りぬける．ルールの詳細については，財団法人ニューテクノロジー振興財団のホームページ（http://www.robomedia.org/）をご参照されたい．

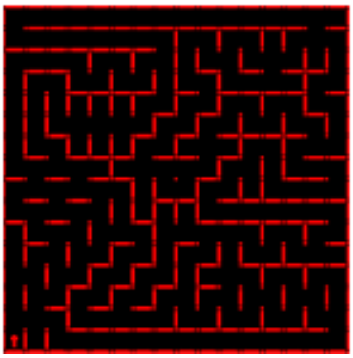

図１　マイクロマウスの迷路の例

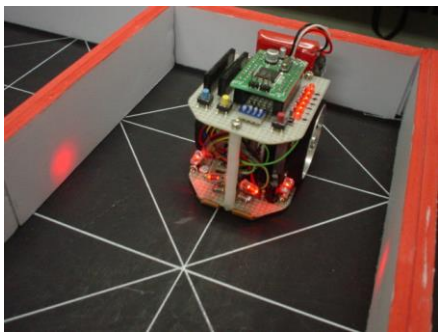

図２　迷路走行中のマイクロマウスの例

4. マイクロマウスのしくみ

4.1 マイクロマウスに使われるセンサ

マウスが迷路を通り抜けるためには、そこに壁があるかどうかを知らなければならない。そこで，マウスが周囲の状況を知るためのセンサが必要になる．光を使って壁の情報を知る光センサ，超音波の反射によって壁との距離を知る超音波センサ，壁との接触を直接検知するタッチセンサなどが使われる．

4.2 マイクロマウスの頭脳

マイクロマウスが周囲の状況を判断して行動を起こすためには頭脳が必要である．ワンボードやワンチップのマイクロコンピュータ（マイコン）がマイクロマウスではよく使われる．

4.3 マイクロマウスの動力

迷路内を移動するために，モータ，歯車，車輪などを使って自分自身で直進または回転する機構が必要である．マイクロマウスに使われる代表的なモータには，DC モータ，ステッピングモータなどがある．DC モータは小型・軽量で高回転だが，十分なトルクを得るためにはギヤによる減速が必要になる．また回転数や回転角を検知する機構と電力供給のコントロールが必要で，DC モータの制御には高度な技術が必要である．ステッピングモータは，与えられたパルスに応じて回転角を制御できるモータで，ロボットの世界ではよく使われる．DC モータと比較すると大型で重量もあり，加減速の制御が必要で高速回転をさせるのは難しいが，高トルクでギヤを使わず直接マイクロマウスを駆動できるものもある．早く走るには重さと動力（電力）のバランスが重要となる．初心者は制御が容易なステッピングモータを，上級者は高速走行可能な DC モータを使う傾向にある．

4.4 マイクロマウスの電力

センサ，マイコン，モータを駆動するには電力（バッテリー）が必要である．以前はマイコンの消費電力が比較的大きく，バッテリーの容量を決めるのに重要な要素となっていたが，最近の低消費電力のマイコンでは，その電力消費はあまり考えなくてよくなってきました．センサも極端に長距離での壁の検出をしない限り，消費電力を抑えて設計することがでる．最近では，走行性能と重量との関係がバッテリーの種類，重量，容量を決める最大の要素となっている．マイクロマウスに使われるバッテリーには従来より広く用いられている２次電池の NiCd，Ni:H バッテリーや，大容量で軽量なことから最近よく用いられるようになった Li イオンや Li ポリマー（LiPo）バッテリーなどがある．

5. マイクロマウスの製作

5.1 マイクロマウスの回路概略

様々な Arduino のボードが市販され，シールドと呼ばれる Arduino 基板上に連結して使用する各種基板も開発されている．市販の基板を用い，シールドおよび機体としてマイクロマウスを構成することもできるが，ここでは Arduino の開発環境を用いてソフトウェア開発を容易にし，かつ安価な機体とするために Arduino のブートローダを搭載した AVR マイコンを用いたマイクロマウス専用の基板および機体を開発した．

図 3 が開発した基板の回路図である．マイコンは Atmel 社の AVR マイコン（ATmega328P）を使用している．Arduino のブートローダは Arduino のサイトにも公開され，AVRISP などツールを持っていれば AVR マイコンへ自分で書き込みすることもできるが，スイッチサイエンス社[4]から書き込み済みの AVR が市販されているのでこれを利用しても良い．Arduino のブートローダが書込まれたものは，シリアルポートを介してオンボードでプログラムの書き込み，修正を行うことができる．

書き込みの際，自動リセットを行うための DTR 端子を含めたシリアル書き込み用の端子を 6 ピンヘッダで用意している．この配列は市販されている USB-シリアル変換ボードの端子（Sparkfun 社[3]，代理店：スイッチサイエンス社 [4]）に合わせている．ジャンパで電源をシリアルポートに供給することもできる．

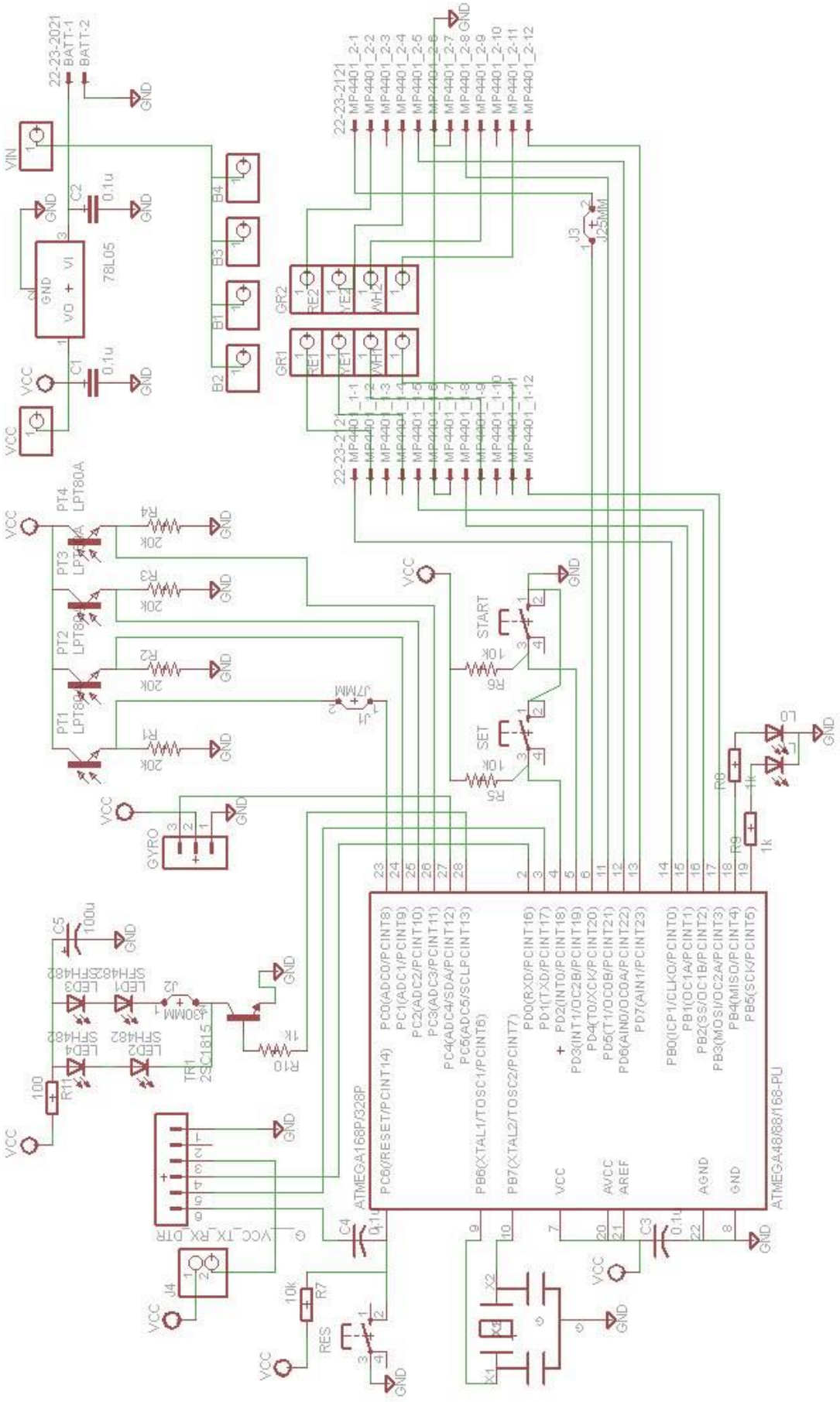

図 3　Arduino マイクロマウスの回路

シリアルポートに接続するデータロガーが市販されているので[5], このようなものを使う時にはジャンパを接続してシリアルポートから電源を供給する．また、オプションとしてジャイロセンサなどのアナログセンサを接続する３端子のコネクタも用意している．

制作するマイクロマウスは，①壁の有無および距離を計測する光学式センサ，②ステッピングモータおよびホイール，③表示用 LED・入力用スイッチを有する．以下，各々の回路について説明する．

①光学式センサ

発光部と受光部よりなる．発光部は，LED をパルス点灯している．電力ロスが少なく，４個の LED を電源電圧５V で高輝度に点灯させるため，２個直列を並列接続している．電源から一旦コンデンサに充電し，コンデンサに充電された電荷を負荷を通さずに直接 LED に加えバイポーラトランジスタでスイッチングする回路である．トランジスタのベース加えた信号（digital pin 19）で点滅する．

受光部はフォトトランジスタをエミッタフォロワとして使用し，２０ｋΩの抵抗に生じた電圧降下をマイコンのアナログ入力ポート（analog input 0 から３）に直接接続している．使用する光学センサの感度に応じて抵抗値を変えてもよい．感度が低い場合は抵抗値を大きくし，感度が高く，壁に近接したときにセンサの入力値が飽和してしまうようなら抵抗値を下げる．

アナログ入力ポート analog input4 は角度制御に用いるジャイロ入力用として準備してある． ３ピンのピンヘッダを取り付ければ，信号線， ５V 電源，GND の３本で接続することができる．

②ステッピングモータおよびホイール

ステッピングモータの駆動には，パワーMOSFET が４個入った IC である MP4401 を用いる．右モータはマイコンの digital pin 8,9,10,11 に，左モータはマイコンの digital pin ４,５,６,７に接続されている．Arduino ではステッピングモータの回転数や回転速度を制御するステッピングモータ制御用のライブラリが用意されているが２個のモータを同時に制御することはできない．そのためライブラリは用いず，マイコンの PB および PD ポートを直接制御してモータを駆動している．

③表示用 LED・入力用スイッチ

通信に用いる RXD,TXD を使用しなければ使用できるデジタル入出力の残りは４本であるので，入力と出力に２本ずつ割り当て，digital pin 2,3 を動作モード選択用の入力スイッチに，digital pin 12,13 を状態表示用 LED，L0,L1 に割り当てた．スイッチは通常 H レベルの入力となるようにプルアップされ，LED は正論理で直接ポートから抵抗を介して接続してある．

5.2 専用基板の設計と製作

図４は開発した専用プリント基板である．これに部品をはんだ付けして組み立てる．部品表は表１に示す通りである．基板の下辺に電源の GND が配置してある．電源のプラス側は３端子レギュレータとステッピングモータ用に配置されている．回路用の＋５V は基板裏面から分配されている．組立ての手順を以下に述べる．

１）ピンの方向に注意してマイコン用の IC ソケット取り付ける．

２）シリアルポート用６ピンヘッダを取り付け．（必要なら GYRO 用に３ピン，シリアルポートの電源供給に 2 ピンのピンヘッダを付けることができる.）

３）タクトスイッチを３個、４端子ともハンダ付けする．

４）抵抗値を確認して抵抗の取り付け．幅の狭い部分への取り付けは片側の線を１８０度折り曲げ縦に差して取り付ける．抵抗が倒れて他の配線に接触しないように取り付け方向に注意する．

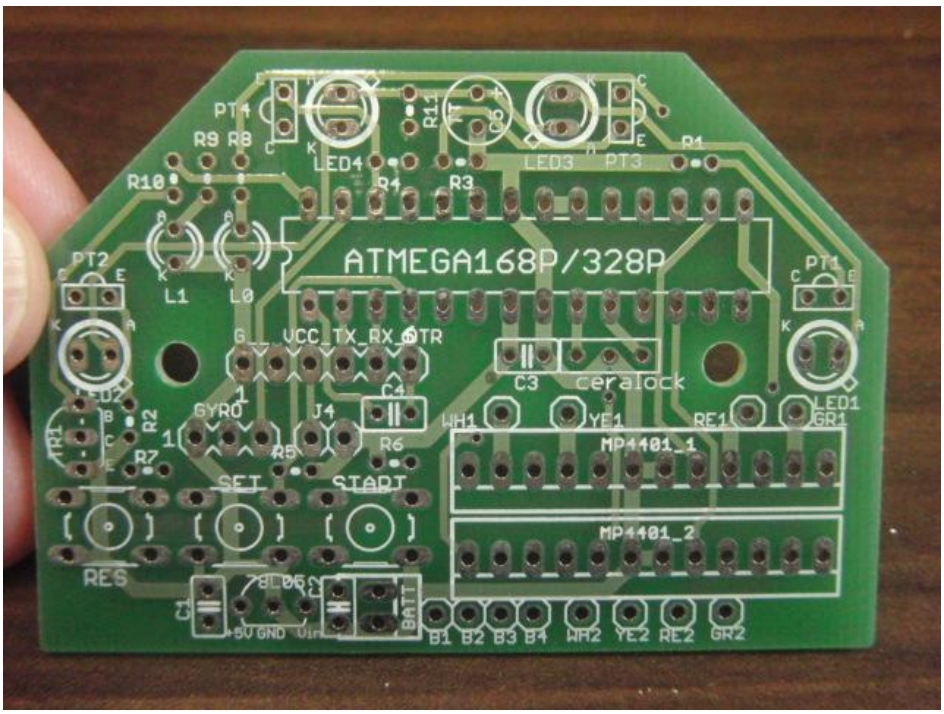

図４　開発した基板のイメージ

表１　部品表

	品名	基板上の記号
1	教育用マイクロマウス基板	
2	ATmega328P(Arduino ブートローダ書込済)	ATMEGA168/328P
3	カーボン抵抗（炭素皮膜抵抗）1／4W 100Ω	R11
4	カーボン抵抗（炭素皮膜抵抗）1／6W 10kΩ	R1,R2,R3,R4,R5,R6,R7
5	カーボン抵抗（炭素皮膜抵抗）1／6W 1kΩ	R8,R9,R10
6	電解コンデンサー 100μF 25V	C5
7	積層セラミックコンデンサー 0．1μF 50V	C1,C2,C3,C4
8	トランジスタ 2SC1815	TR1
9	3端子レギュレータ	78L05
10	照度センサ（フォトトランジスタ）	PT1,PT2,PT3,PT4
11	超高輝度赤LED 5mm OSHR5111A－TU	LED1,LED2,LED3,LED4
12	赤色LED 3mm OSDR3133A	L0,L1
13	パワーMOS－FETモジュール MP4401	MP4401-1,MP4401-2
14	タクトスイッチ	RES, SET, START
15	ﾋﾟﾝﾍｯﾀﾞ 1x6P	G_VCC_TX_RX_DTR
16	電池ボックス（単3X6・白・スナップ）	
17	バッテリースナップ（電池スナップ）プラスチック製	
18	24対1ユニポーラステッピングモータ	
19	自在タイヤ φ40 2個入り	
20	スペーサ	
21	アルミ板材 A4 版 A5052	
22	スベリ材用スポンジ	
23	スベリ材	

5）積層セラミックコンデンサ，セラロックを取り付ける．セラロックの中央のピンは GND で左右の極性はない．

6）極性のある部品については回路図と基板のパターンの対応を考えながら方向を間違えないように取

り付けること．
・電池スナップは基板内側をプラス，外側をマイナスに取り付ける．
・トランジスタ 2SC1815 は印字面を見て左から E,C,B となる．
・３端子レギュレータ 78L05 は印字面を見て左から＋５V，GND，Vin（電池のプラス側）となる．同等品でもピン配置が異なるものがあるので規格表で確認すること．
・C5 の電解コンデンサは基板外側がプラス，基板内側がマイナスとなるよう取り付ける．
・光センサの壁照射用の赤色 LED１から LED4 は A(アノード)側がプラス（電流が流入），K(カソード)側がマイナス（流出）になるようにする．フォトトランジスタ PT1 から PT4 は C(コレクタ)が＋５V に接続され，E（エミッタ）側が抵抗を介して GND に接続される．LED からの光が直接 PT に入らないように，それぞれ黒の熱収縮チューブで側面を覆うなどすること（図６の写真を参照）．LED とフォトトランジスタが組になったフォトリフレクタを用いる場合は、極性によりそのままでは使用できない場合があるので注意が必要である．
・MP4401 は印字面を見て左端が１番ピン，右端が１２番ピンである．
７）ステッピングモータの配線について，B1,B2,B3,B4 には，ステッピングモータの青色と黒色の線を 2 個分の計４本接続する．右のステッピングモータの白線を WH1，黄線を YE1，赤線を RE1，緑線を GR1 に配線する．緑モータ線に配線する．左のステッピングモータも同様に WH2,YE2,RE2,GR2 に配線する．

5.3 プログラムの書き込みと本体の組立

部品の取り付けが終わったら，モータに過大な電流が流れるなどで基板が損傷を受けるのを防ぐために，最初はモータを接続せずにプログラムの書き込みを行う．その様子を図５に示す．６ピンヘッダの左側が GND 右側が DTR となるように垂直に USB シリアル変換器を差して，Arduino の開発環境よりプログラムの書き込みを行った．ボードの設定は，Arduino Duemilanove w/ATmega328 を選択する．シリアルポートの番号は接続した USB シリアル変換器が認識されたポート番号を選ぶ．

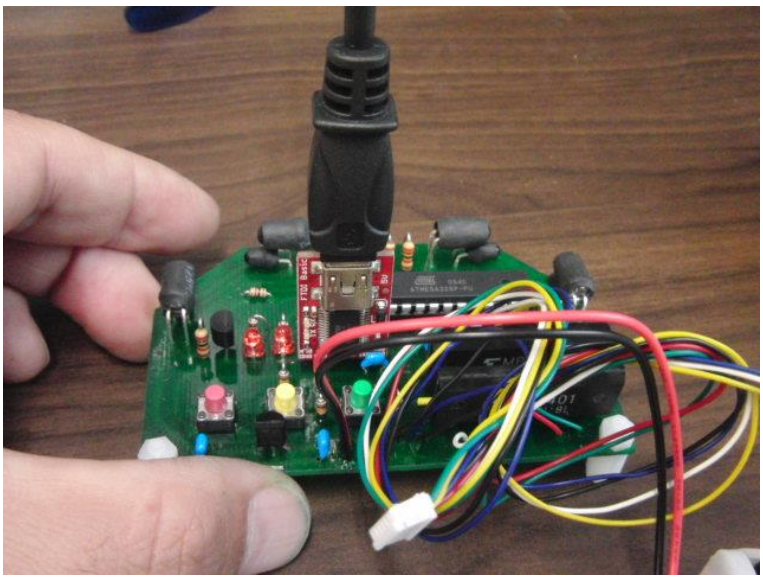

図５　製作した基板とプログラム書き込みの様子
（写真は初期試作品，シリアルポート電源ジャンパとジャイロポートなし）

書き込みが終わったら図面を元に本体の製作，モータの取り付け，タイヤ・ホイールの取り付け，基板の取り付けとモータへの配線を行う．最後に本体が水平になるように本体後部にスポンジとスベリ材で高さ調整をする．（図６参照）

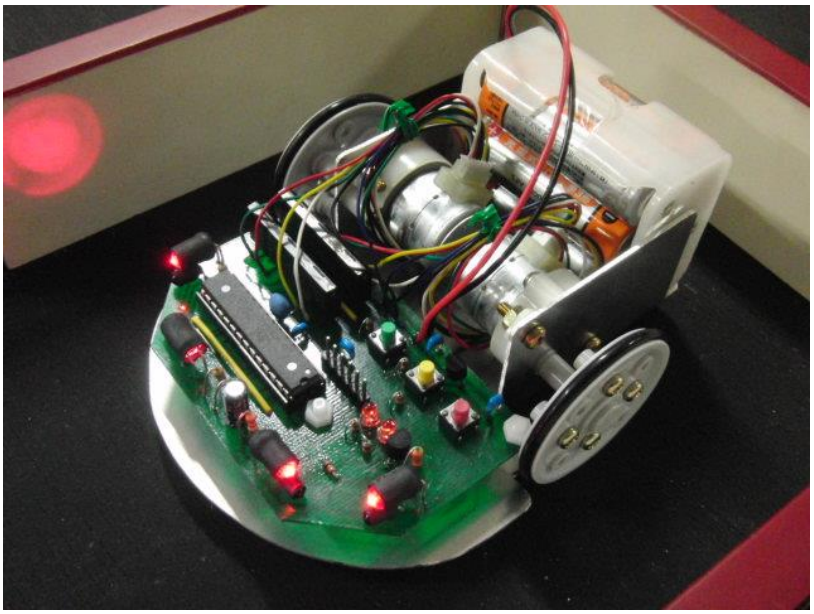

図６　開発したマイクロマウスの概観（写真は初期試作品）

6. マイクロマウスのソフトウェア

開発したマイクロマウスの全ソフトウェアは付録リスト１に示す．Arduino では setup()関数に書かれたことを最初に一度だけ実行するのでここに各種初期設定を記述した。loop()関数に書かれたことを繰り返すのでここに動作メニューを記述し、メニューからボタン操作で各種プログラムを呼び出すようにした。また MSTimer２ライブラリを用いて、タイマー割込みにより光学式センサおよびモータ制御を行う関数を呼び出した．詳細はリストを参照されたい．

操作手順について，簡単に説明する．

リセットボタン（写真では赤ボタン）を押すとメモリがクリアされ，プログラムが最初から実行される．電源を ON した状態でもリセットがかかるようになっている．

セットボタン（写真では黄色ボタン）を押すと動作モードが切り替わる．サンプルスケッチでは，

モード０（L１オフ，L0 オフ）：　センサチェック
モード１（L１オフ，L0 オン）：　U ターン走行
モード２（L１オン，L0 オフ）：　拡張左手法による迷路探索走行
モード３（L１オン，L0 オン）：　最短経路走行

となっている．

まずモード０でセンサの感度調整とセンサの基準値設定を行う．状態表示用 LED は２個しかないため，光学式センサの値を表示させることができない．Arduino ではシリアル通信用ライブラリを用いて PC とのシリアル通信は容易にできる．光学式センサの値はマイクロマウスと PC をシリアルポートで接続してセンサの読みを PC のターミナルソフトで表示させるようにした（図７）．

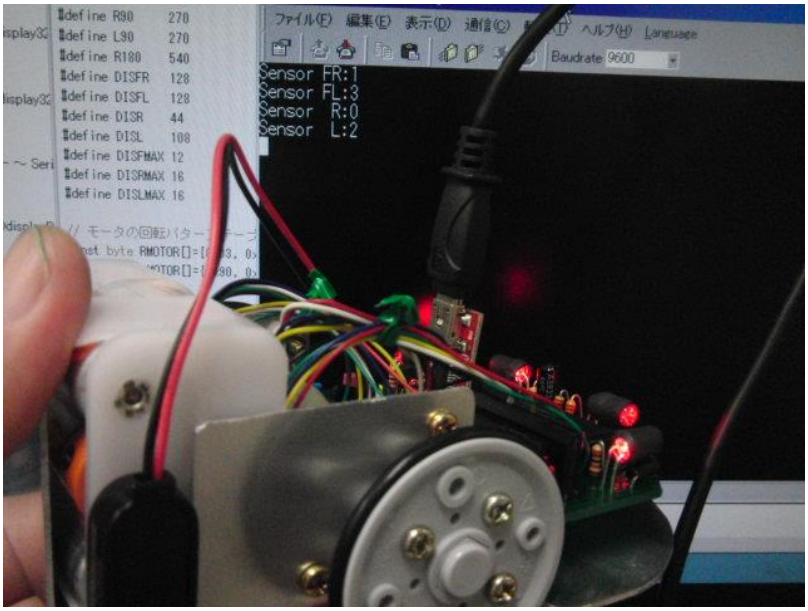

図７　PC との接続による光学センサの値表示

モード０でスタートボタン（赤色ボタン）を押すと，シリアルポートより PC へセンサの値が送信される．SensorFR が右前のセンサ値，SensorFL が左前，SensorR が右横，SensorL が左横のセンサの読みを１０ビット A/D 変換で読み取り１０進数で表示している．マイクロマウスを迷路内の区画中央（左右 2 個の車輪が中央に対称位置になるように）においてセンサ値がそれぞれ最大の値となるように LED とフォトトランジスタの光軸調整を行う．

最大の読みを基準値として，サンプルスケッチの以下の文の数値を書き換える．

```
#define R180      530                    // 右180度回転の歩数
#define DISFR     599                    // 前右壁の標準距離のセンサ値
#define DISFL     625                    // 前左壁の標準距離のセンサ値
#define DISR      200                    // 右壁の標準距離のセンサ値
#define DISL      190                    // 左壁の標準距離のセンサ値
```

基準値の設定が終わったらスケッチをコンパイルしてマイクロマウスにプログラムを書き込み，セットボタンでモードを１に設定してスタートボタンを押すと，マイクロマウスは前進し，前壁を検知したら U ターンをして前進を繰り返す走行を行う．左右の壁があれば右壁を優先に壁との距離を一定にした走行を行うので，蛇行するようならセンサの基準値の調整を行う．

次は，探索走行時の壁の有無を判断するしきい値調整を行う．マイクロマウスが探索走行の前進中に区画の２／３だけ進んだ地点でセンサの値を調べ，しきい値より大きいか小さいかで前壁，右壁，左壁の有無を判断しているので，センサの値の変化を見ながらしきい値を調整して以下の文の数値を書き換える．

```
#define DISFMAX 60                       // 前壁の最大距離のセンサ値(センサ値は最小)
#define DISRMAX 80                       // 右壁の最大距離のセンサ値(センサ値は最小)
#define DISLMAX 80                       // 左壁の最大距離のセンサ値(センサ値は最小)
```

しきい値の調整を設定したら再度コンパイルしてマイクロマウスに書き込み，モード２で探索走行ができまる．サンプルスケッチでは走行誤差に対する補正がほとんど入っていないので，簡単な迷路しか走行できないが，補正を加えれば難しい迷路でも探索できるようになる．

ゴール位置は（7， 7），（7， 8），（8， 7），（8， 8）の４箇所に設定されており，ゴールに到達後にスタート位置まで戻ってきたら，マウスの向きをスタート方向に向けなおしてモードを３に切替て再度スタートボタンを押すと最短走行を行い，ゴール位置で停止する．

参考文献

[1] http://ja.wikipedia.org/wiki/Arduino

[2] http://www.arduino.cc/

[3 http://www.sparkfun.com/

[4] http://www.switch-science.com/

[5] OpenLog 等，http://www.switch-science.com/products/detail.php?product_id=389

付録　リスト１

```
//*****************************************************************************
//
//                        KNCT-MMEdu for Arduino          熊本高専 葉山清輝
//
//  セットスイッチでメニュー選択. 00-11 まで LED に表示される
//   00:センサチェック, 01:予備, 10:拡張左手法, 11:最短走行
//   2010.11.28   Ver 1.0
//*****************************************************************************
#include <MsTimer2.h>

#define PT_FR   0                       // 右前センサ, アナログ入力
#define PT_FL   1                       // 左前センサ, アナログ入力
#define PT_R    2                       // 右センサ,  アナログ入力
#define PT_L    3                       // 左センサ,  アナログ入力
#define GYRO    4                       // ジャイロ用に予約, アナログ入力
#define LEDOUT 19                       // 壁センサ用 LED 出力(AN5 をデジタル 19 として使う)
#define SET     2                       // セットスイッチ, デジタル入力
#define START   3                       // スタートスイッチ, デジタル入力

// 各種基準値
#define LSPD     3                      // 低速走行のタイマーカウント値
#define HSPD     2                      // 高速走行のタイマーカウント値
#define STEP1    665                    // 1ステップの歩数
#define R90      265                    // 右90度回転の歩数
#define L90      265                    // 左90度回転の歩数
#define R180     530                    // 右180度回転の歩数
#define DISFR    599                    // 前右壁の標準距離のセンサ値
#define DISFL    625                    // 前左壁の標準距離のセンサ値
#define DISR     200                    // 右壁の標準距離のセンサ値
#define DISL     190                    // 左壁の標準距離のセンサ値
#define DISFMAX 60                      // 前壁の最大距離のセンサ値(センサ値は最小)
#define DISRMAX 80                      // 右壁の最大距離のセンサ値(センサ値は最小)
#define DISLMAX 80                      // 左壁の最大距離のセンサ値(センサ値は最小)

// モータの回転パターンテーブル, 右と左は逆向きなので逆回転
const byte RMOTOR[]={0x03, 0x06, 0x0C, 0x09, 0x00};   // 右モータは PORTB へ出力
const byte LMOTOR[]={0x90, 0xC0, 0x60, 0x30, 0x00};   // 左モータは PORTD へ出力

const byte DtoR[]={0,2,4,0,8,0,0,0,1};    // 現在の方向に対して右側の方向を与えるテーブル
```

```
const byte DtoL[]={0,8,1,0,2,0,0,0,4};    // 現在の方向に対して左側の方向を与えるテーブル

byte pmode=0;                                    // プログラムモード

// 割込み内で値を変化させる変数は Volatile で宣言が必要
volatile byte modeR=0, modeL=0;                    // モータの加減速・正逆転のモード
volatile int stepR, stepL;                         // モータの回転数の指示用変数
volatile byte patR=0, patL=0;                      // モータの励磁パターンの位相
volatile int cntR, cntL;                           // モータの回転数カウント
volatile int sensFR, sensFL, sensR, sensL;         // センサの値
volatile int sensFRB, sensFLB, sensRB, sensLB;     // LED 消灯時のセンサの値
volatile byte timR=0, timL=0;                      // 割り込み内で使う R,L モータのタイマー
volatile byte timS;                                // 割り込み内で使う光センサのタイマー
volatile byte fS=0;                                // 速度フラグ, 高速の時に壁との距離補正をかける
volatile byte fR=0, fL=0;                          // R,L モータの低速・高速切り替えフラグ 0:低速, 1:高速

union {                                            // マップを記憶する配列の構造定義
        byte all;                                  // mmap をバイト単位でアクセス
        struct { byte n:1;                         // 北方向の壁の有無(0:無, 1:有)
                 byte e:1;                         // 東方向の壁の有無(0:無, 1:有)
                 byte s:1;                         // 南方向の壁の有無(0:無, 1:有)
                 byte w:1;                         // 西方向の壁の有無(0:無, 1:有)
                 byte d:4;                         // 履歴(離脱方向)を4ビットで記憶領域確保
               };
       } mmap[16][16];

//--------------------------------------------------------------------------
//   LED 表示
//--------------------------------------------------------------------------
void dispLED(byte n)
{
 PORTB &= B11001111;                               // LED を一旦消す
 PORTB |= (n<<4);                                  // LED 表示
}

//--------------------------------------------------------------------------
//   タイマー割り込み処理
//--------------------------------------------------------------------------
void SensAndMotor(void) {

// モータ回転 mode = 0: フリー, 1: 前進, 2: 後退, 3: ブレーキ
// 右モータ処理
  if (timR>0) timR--;                              // timR をカウントダウン, timR=0 のとき処理
  if (timR==0) {
    if (fR==0) timR=LSPD; else timR=HSPD;
    if (modeR==1) {if (patR < 3) patR++; else patR = 0; }
    if (modeR==2) {if (patR > 0) patR--; else patR = 3; }
    cntR++;                                        // 右モータの歩数カウント
  }
// 左モータ処理
  if (timL>0) timL--;                              // timL をカウントダウン, timL=0 のとき処理
  if (timL==0) {
    if (fL==0) timL=LSPD; else timL=HSPD;
    if (modeL==1) {if (patL < 3) patL++; else patL = 0; }
    if (modeL==2) {if (patL > 0) patL--; else patL = 3; }
    cntL++;                                        // 左モータの歩数カウント
```

```
  }

  if (modeR==0 || modeL==0) { patR=4; patL=4; }   // mode0 のときはフリー
  PORTB &=0xf0; PORTB |= RMOTOR[patR];            // 右モータへパターンの出力
  PORTD &=0x0f; PORTD |= LMOTOR[patL];            // 左モータへパターンの出力

// センサ処理
// LED 消灯時の基準値を取り, LED 点灯後の差分でセンサ値を読み取る.
  if (timS<20) timS++; else timS=0;               // センサ読み取り周期カウンタ
  if (timS==0){
    sensFRB=analogRead(PT_FR);                    // LED 消灯時の初期値を入力
    sensFLB=analogRead(PT_FL);
    sensRB=analogRead(PT_R);
    sensLB=analogRead(PT_L);
    digitalWrite(LEDOUT, HIGH);                   // LED-ON
    delayMicroseconds(50);                        // LED の点灯に時間がかかるので少し待つ
    sensFR=analogRead(PT_FR)-sensFRB;             // LED 消灯時の初期値を入力
    sensFL=analogRead(PT_FL)-sensFLB;
    sensR =analogRead(PT_R) -sensRB;
    sensL =analogRead(PT_L) -sensLB;
    digitalWrite(LEDOUT, LOW);                    // LED-OFF
  }

// 左右センサによる距離の補正
// 右壁があるときは右壁のみを使い左壁無視, 左壁のみの時は左壁だけをつかう
// 標準の距離(センサ値)と現在の差をとり, 差が大きい時に補正する
  if (fS==1){                                     // 壁との距離補正する場合, 以下の処理
    fR=fL=1;                                      // 左右をまず高速設定
    if(sensR>DISRMAX){                            // 右壁が存在しているなら右壁のみで調整
      if ((sensR-DISR)>20) fL=0;                  // 右壁に近づきすぎた時, 左モータを減速
      if ((sensR-DISR)<-20) fR=0;                 // 右壁から遠すぎる時, 右モータを減速
    } else if(sensL>DISLMAX){                     // 左壁のみ存在している場合の調整
      if ((sensL-DISL)>20) fR=0;                  // 考え方は右壁と同じ
      if ((sensL-DISL)<-20) fL=0;
    }
  } else { fR=fL=0; }                             // fS=0 の時, 低速に設定

}

//--------------------------------------------------------------------------
// センサ調整, センサの値を LED に表示
//--------------------------------------------------------------------------
void check_sens() {
  while (1){
    Serial.print(0x0c,BYTE);   // 改ページ
    Serial.print("Sensor FR:"); Serial.println(sensFR); // 右前センサ値を出力
    Serial.print("Sensor FL:"); Serial.println(sensFL); // 左前センサ値を出力
    Serial.print("Sensor  R:"); Serial.println(sensR);  // 右センサ値を出力
    Serial.print("Sensor  L:"); Serial.println(sensL);  // 左センサ値を出力
    delay (500);
  }
}

//--------------------------------------------------------------------------
//          ブレーキ
//--------------------------------------------------------------------------
```

```
void run_break(){
  modeR=0; modeL=0;                          // モータ停止
}

//-----------------------------------------------------------------------
//        前壁による角度調整
//-----------------------------------------------------------------------
void adjust(){
  fS=0;                                      // 低速
  while(abs((sensFR-DISFR)-(sensFL-DISFL))>20){   // 前センサの値の差が大きい時補正する
    if ((sensFR-DISFR)>(sensFL-DISFL)) {
      modeR=2; modeL=1;                      // 右ターン
    } else {
      modeR=1; modeL=2;                      // 左ターン
    }
  }
  run_break();
}

//-----------------------------------------------------------------------
//        前進時に低速でスタート
//-----------------------------------------------------------------------
void slow_start(){
  fS=0;                                      // 最初は低速設定
  modeR=modeL=1;                             // モード設定 右:前進, 左:前進
  cntR=0; stepR=20;                          // 低速で50ステップ進む
  while (cntR<stepR);                        // 指定回転数まで待つ.
                                             // 何もしないと最適化で無視されてしまうので delay を入れている.
}

//-----------------------------------------------------------------------
//        ステップ前進
//-----------------------------------------------------------------------
void run_step(){
  slow_start();
  fS=1;                                      // 残りの距離を高速で進む
  cntR=0; stepR=STEP1-20;
  while (cntR<stepR) delay(1);
  run_break();
}

//-----------------------------------------------------------------------
//        右90度回転
//-----------------------------------------------------------------------
void run_R90(){
  fS=0;                                      // 低速設定
  cntR=0; stepR=R90;                         // 右 90 度回転の歩数セット
  modeR=2; modeL=1;                          // 右:後退, 左:前進
  while (cntR<stepR) delay(1);
  run_break();
}

//-----------------------------------------------------------------------
//        左90度回転
//-----------------------------------------------------------------------
void run_L90(){
```

```
  fS=0;                                          // 低速設定
  cntL=0; stepL=L90;                             // 左 90 度回転の歩数セット
  modeR=1; modeL=2;                              // 右:前進, 左:後退
  while (cntL<stepL);
  modeR=0; modeL=0;
  run_break();
}

//-----------------------------------------------------------------------
//        右180度回転
//-----------------------------------------------------------------------
void run_R180(){
  fS=0;                                          // 低速設定
  cntR=0; stepR=R180;                            // 右 90 度回転の歩数セット
  modeR=2; modeL=1;                              // 右:後退, 左:前進
  while (cntR<stepR);
  run_break();
}

//-----------------------------------------------------------------------
//        U ターンまたは右ターン走行,加速前進, 壁があったら減速, 距離補正後ターン
//-----------------------------------------------------------------------
void run_Turn(byte n){
  while (1){
    slow_start();
    fS=1;                                        // 残りの距離を高速で進む
    while (sensFR<DISFR);                        // 前(右)センサで標準距離まで走らせる
    adjust();                                    // 前壁による角度調整
    if (n==0) run_R180(); else run_R90();        // 右 180 度ターン または 右90度ターン
  }
}

//-----------------------------------------------------------------------
//        拡張左手法(履歴を記録)
//  左, 前, 右の優先順位で方向決定, 行き止まりなら U ターン
//-----------------------------------------------------------------------
void run_Hidarite(){
  byte wF, wR, wL;                               // 前, 右, 左に壁があるかどうかのフラグ
  byte wS;                                       // 壁センスを行ったかどうかのフラグ
  byte mapF, mapR, mapL;                         // マップ上の履歴を読み出す時に使う変数
  byte mx,my;                                    // マウスの x,y 座標, 初期は 0,0, 北にyが増分, 東にxが増分
  byte md;                                       // マウスの向いている方向, 北:1, 東:2, 南:4, 西:8

  mapF=0; mapR=0; mapL=0;                        // マップ読み出し変数の初期化
  mx=0; my=0; md=1;                              // マウスの初期座標は 0,0, 北向き
  wF=0; wR=1; wL=1;                              // スタート時点は前進から入る
  mmap[0][0].d=1;                                // 初期位置の履歴を北方向にセット

  while (digitalRead(START)==HIGH){              // 青ボタンが押されるまで繰り返す

  // 履歴の読み出し(mapF,mapR,mapL に前区画, 右区画, 左区画)
  // ただし, 現在地が外周の区画のときは mmap 配列からはみ出した区画をアクセスさせない.
    switch (md){
      case 1: if (my<15) mapF=mmap[my+1][mx].d; // 現方向が北向きのとき
              if (mx<15) mapR=mmap[my][mx+1].d;
              if (mx>0)  mapL=mmap[my][mx-1].d;
```

```
                break;
        case 2: if (mx<15) mapF=mmap[my][mx+1].d; //  現方向が東向きのとき
                if (my>0)   mapR=mmap[my-1][mx].d;
                if (my<15) mapL=mmap[my+1][mx].d;
                break;
        case 4: if (my>0)   mapF=mmap[my-1][mx].d; //  現方向が南向きのとき
                if (mx>0)   mapR=mmap[my][mx-1].d;
                if (mx<15) mapL=mmap[my][mx+1].d;
                break;
        case 8: if (mx>0)   mapF=mmap[my][mx-1].d; //  現方向が西向きのとき
                if (my<15) mapR=mmap[my+1][mx].d;
                if (my>0)   mapL=mmap[my-1][mx].d;
                break;
    }

// ここから左手法の条件判断
    if (wL ==0 && (mapL==0 || mapL==DtoL[md]))        //  左壁が無く,未進入または履歴で左折可の時に左折
      {run_L90(); md=DtoL[md]; }
    else if (wF==0 && (mapF==0 || mapF==md)){}        //  前壁が無く,進入可のとき前進(elseif 以下素通り)
    else if (wR==0 && (mapR==0 || mapR==DtoR[md]))
      {run_R90(); md=DtoR[md]; }                      //  右壁が無く,進入可の時右折
    else {run_R180(); md=DtoR[md]; md=DtoR[md];}      // U ターン

// ここから, センサで前後左右の壁を見ながら前進させる.
    wS=0; wF=0; wR=0; wL=0;                           //  壁判断のフラグをリセット
    slow_start();                                     //  低速スタート
    stepR=STEP1;
    fS=1;                                             //  高速設定
    while (cntR<stepR){                               //  1ステップ分回す.
      if (cntR > (STEP1*2/3) && wS==0){               //  歩数誤差による誤動作防止のためステップの 2/3 の所で壁検出
        wS=1;                                         // 1 度だけ壁検出するためフラグ(wS)を立てる
        if (sensR > DISRMAX) wR=1; else wR=0;         //  右壁検出
        if (sensL > DISLMAX) wL=1; else wL=0;         //  左壁検出
        if ((sensFR>DISFMAX || sensFL>DISFMAX)){ wF=1; break; }  //  前壁検出, 前壁があったらループを抜ける
      }
    }

    //  前壁を検出してループを抜けたとき, 前壁まで進む
    if (wF==1){
      while (sensFR<DISFR);                           //  壁で距離調整
      adjust();                                       //  前壁による角度調整
    }

// 履歴(離脱方向の逆方向)を進む前の座標に記録.
// 壁情報はマウスの方向に応じて座標を更新した後に記録.
    switch (md){
      case 1: mmap[my][mx].d=4; my++; mmap[my][mx].n=wF; mmap[my][mx].e=wR; mmap[my][mx].w=wL; break;
      case 2: mmap[my][mx].d=8; mx++; mmap[my][mx].e=wF; mmap[my][mx].s=wR; mmap[my][mx].n=wL; break;
      case 4: mmap[my][mx].d=1; my--; mmap[my][mx].s=wF; mmap[my][mx].w=wR; mmap[my][mx].e=wL; break;
      case 8: mmap[my][mx].d=2; mx--; mmap[my][mx].w=wF; mmap[my][mx].n=wR; mmap[my][mx].s=wL; break;
    }
    if (mx==0 && my==0) { run_break(); break; }       //  スタート地点に戻ったら探索終わり
  }
}
```

```
//------------------------------------------------------------------------
//          マップから最短距離を求め，最短走行
//------------------------------------------------------------------------
void run_saitan(){
  byte i,j,k,m;                          // 汎用変数
  byte smap[16][16];                     // 最短距離を求めるマップ
  byte run[256];                         // 最短走行パターンを入れる配列
  byte md;                               // マウスの向いている方向，北:1，東:2，南:4，西:8

// 最短距離マップをクリア，未探索区画は全てかべがあるものとする.
  for(i=0;i<16;i++){
    for(j=0;j<16;j++){
      smap[i][j]=0;
      if (mmap[i][j].d==0){
        mmap[i][j].n=1; if (i<15) mmap[i+1][j].s=1;
        mmap[i][j].e=1; if (j<15) mmap[i][j+1].w=1;
        mmap[i][j].s=1; if (i>0)  mmap[i-1][j].n=1;
        mmap[i][j].w=1; if (j>0)  mmap[i][j-1].e=1;
      }
    }
  }

// 歩数マップ作成
// ゴール位置に1をセット，m の初期値は1．全区画スキャンし，歩数mの位置なら移動可能な
// 区画に次の歩数(m+1)をセット.これを繰り返し，スタート地点に到達したらループを抜ける.

  smap[7][7]=1; smap[7][8]=1; smap[8][7]=1; smap[8][8]=1; // ゴールに1をセット
  m=1;                                                    // m の初期値セット
  for(k=0;k<255;k++){                                     // 最高 255 回繰り返し
    for(i=0;i<16;i++){
      for(j=0;j<16;j++){                                  // 全区画スキャン
        if (smap[i][j]==m){
          if (mmap[i][j].n==0 && i<15 && smap[i+1][j]==0) smap[i+1][j]=m+1;
          if (mmap[i][j].e==0 && j<15 && smap[i][j+1]==0) smap[i][j+1]=m+1;
          if (mmap[i][j].s==0 && i>0  && smap[i-1][j]==0) smap[i-1][j]=m+1;
          if (mmap[i][j].w==0 && j>0  && smap[i][j-1]==0) smap[i][j-1]=m+1;
        }
      }
    }
    m++;                                                  // 歩数を進める
    if (smap[0][0]!=0) break;                             // スタート地点まで歩数を埋めたらループから抜ける
  }

// 歩数マップを逆にたどって run[k] に最短経路の走行パターンを作る
// k はパターン数. 1:直進，2:右折，3:左折
  m=smap[0][0]-1;                                         // m にスタート位置の歩数を入れ，逆にたどる
  i=0; j=0; k=0;
  md=1;
  while (m>0){                                            // ゴールまで到達した後，m--された時点で終わり
    switch(md){
      case 1: if (mmap[i][j].n==0 && smap[i+1][j]==m && i<15) {run[k]=1; i++; m--;      break;}
              if (mmap[i][j].e==0 && smap[i][j+1]==m && j<15) {run[k]=2; md=DtoR[md]; break;}
              if (mmap[i][j].w==0 && smap[i][j-1]==m && j>0 ) {run[k]=3; md=DtoL[md]; break;}
      case 2: if (mmap[i][j].e==0 && smap[i][j+1]==m && j<15) {run[k]=1; j++; m--;      break;}
              if (mmap[i][j].s==0 && smap[i-1][j]==m && i>0 ) {run[k]=2; md=DtoR[md]; break;}
```

```
                if (mmap[i][j].n==0 && smap[i+1][j]==m && i<15) {run[k]=3; md=DtoL[md]; break;}
        case 4: if (mmap[i][j].s==0 && smap[i-1][j]==m && i>0 ) {run[k]=1; i--; m--;       break;}
                if (mmap[i][j].w==0 && smap[i][j-1]==m && j>0 ) {run[k]=2; md=DtoR[md]; break;}
                if (mmap[i][j].e==0 && smap[i][j+1]==m && j<15) {run[k]=3; md=DtoL[md]; break;}
        case 8: if (mmap[i][j].w==0 && smap[i][j-1]==m && j>0 ) {run[k]=1; j--; m--;       break;}
                if (mmap[i][j].n==0 && smap[i+1][j]==m && i<15) {run[k]=2; md=DtoR[md]; break;}
                if (mmap[i][j].s==0 && smap[i-1][j]==m && i>0 ) {run[k]=3; md=DtoL[md]; break;}
      }
      k++;
    }

// 最短経路を走行
    i=0;
    while (i<k){
      if (run[i]==1) { run_step(); i++; }
      if (run[i]==2) { run_R90();   i++; }
      if (run[i]==3) { run_L90();   i++; }
    }
}

//---------------------------------------------------------------------------
//                      初期化
//---------------------------------------------------------------------------
void setup()
{
    int i,j;

    DDRB |=B00111111;                       // PB0-4 右モータ, PB5,PB6 は LED の L0,L1
    DDRD |=B11110000;                       // PD4-7 左モータ, PD1 は TX, PD2 は SET,PD3 は START
    pinMode(LEDOUT, OUTPUT);                // 壁センサ用 LED 出力
    digitalWrite(LEDOUT, LOW);              // 最初は LED 消灯

// マップの初期化
    for (i=0; i<16;i++) for (j=0;j<16;j++) mmap[i][j].all=0;     // まずクリア
    for (i=0; i<16;i++){
        mmap[i][0].w=1; mmap[i][15].e=1;                        // 東西の外周の壁をセット
        mmap[0][i].s=1; mmap[15][i].n=1; }                      // 南北の外周の壁をセット
    mmap[0][0].e=1;

    MsTimer2::set(1, SensAndMotor);         // 周期設定, 1ms
    MsTimer2::start();                      // タイマースタート
    Serial.begin(9600);                     // シリアルポートを開く, 壁センサチェック用
}

//---------------------------------------------------------------------------
//                      メインループ
//---------------------------------------------------------------------------
void loop()
{
    while (digitalRead(START)==HIGH){
      if (digitalRead(SET)==LOW) {          // セットスイッチが押されたらプログラムモードを進める
        delay(10);   // チャタリング防止
        while(digitalRead(SET)==LOW);
        delay(10);   // チャタリング防止
        pmode++; if (pmode>3) pmode=0;
      }
```

```
      dispLED(pmode);
    }
    dispLED(0);
    delay(500);                                  // 0.5 秒待つ

    // プログラム実行
    switch(pmode){
/*
      case   0: run_step(); break;              // 1ステップ走行
      case   1: run_R90();   break;              // 右 90 度ターン
      case   2: run_L90();   break;              // 左 90 度ターン
      case   3: run_R180(); break;               // 180 度ターン
*/
      case   0: check_sens();     break;         // センサチェック
      case   1: run_Turn(0);       break;        // U ターン走行
      case   2: run_Hidarite(); break;           // 拡張左手法
      case   3: run_saitan();     break;         // 最短経路を走行
    }
}
```