# 講習会

# Arduino

11回目

シールド（ライブラリと基板）

# 目的

- 自分でライブラリを入れることができるようになる。
- Arduinoに用意されたシールドを使えるようにする。

# シールド

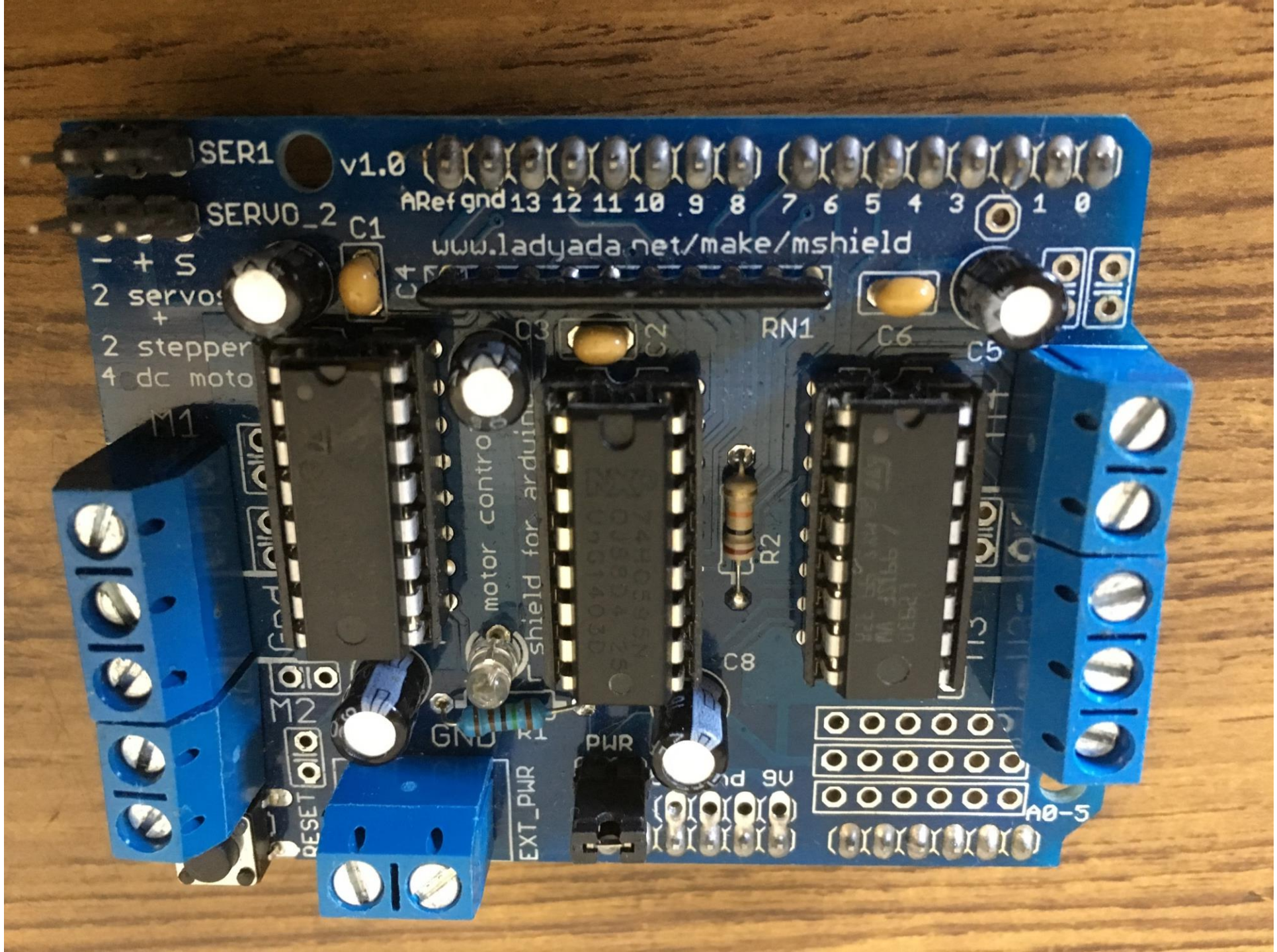
- Arduinoの周辺機器みたいなもの
- プリント基板に部品が配置、はんだ付けされておりArduino UNO等に挿すだけで使えるようになる。
- プログラムは自分で書かないといけないがサンプルがある。
- NanoやMicroでは挿せない
- 自作が可能

# 例

- LCD_Keypad_Shield
  （LCD画面に表示、ボタン操作あり）
- サーマルプリンターシールド
  （感熱紙に印刷）
- モータシールド
  （DCモータの操作）
- イーサネットシールド
  （有線LANの接続）

# モーターシールド

- 複数のDCモータを制御できるシールド
- ステッピングやサーボを制御できるのもある。


SER1
v1.0
SERVO_2
ARef gnd 13 12 11 10 9 8 7 6 5 4 3 1 0
www.ladyada.net/make/mshield
C1
C2
C3
C4
C5
C6
RN1
R2
C8
M1
M2
GND
PWR
EXT_PWR
RESET
A0-5

# 性能（例）

- 制御素子：PCA9685PW（16ch　12bit　PWM　I2C－bus）
- 駆動素子：TB6612FNG（２回路入Ｈブリッジ）×２素子
  モーター駆動電流：1.2A（3.2Aピーク）／Ｈブリッジ１回路あたり
  モーター駆動電圧：4.5～13.5V
- モーター駆動数：ＤＣモーター４個まで（独立に８ビットのスピード制御、正転／逆転制御）
  ：ステッピングモータ2個まで（バイポーラ／ユニポーラ）
  （独立に制御：1相励磁、2励磁、1-2相励磁、マイクロステッピング）
  ：サーボモーター２個まで（Arduinoから直接制御）
- スタッキング数：最大32シールド（I2Cアドレス空間：５ビット、ジャンパ設定による）
  （128個までのDCモーター、64個までのステッピングモーター、64個までのサーボーモーター）

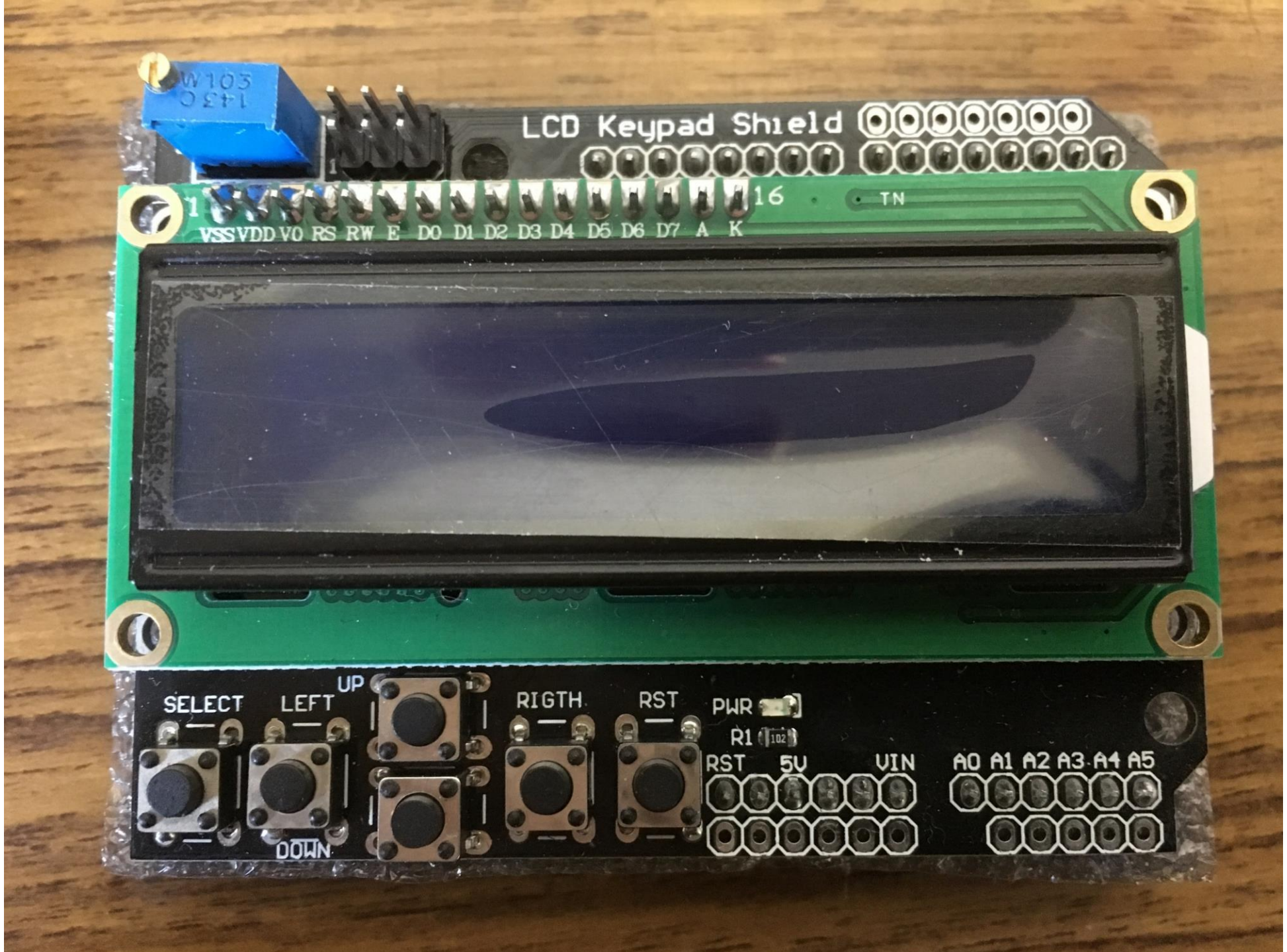

Digital I/O
PWMs
Adafruit Motor Shield v2.3
Steppers or
DC motors
GND
Analog In
Servo
Power
5-12V Motor
Power

- 使用電圧:5V～12V
- モーターコントローラ:L298P 2個のDCモーターもしくは1個のステッピングモーターを使用できます
- 最大電流:1チャンネル当り最大2A (外部電源使用時)
- 電流センサ:1.65V/A
- 使用ピン
  機能:チャンネル A,チャンネル B
  ・回転方向: D12,D13
  ・PWM: D3,D11
  ・ブレーキ: D9,D8
  ・電流センサ:A0,A1
  ブレーキ機能や電流センサ入力を使わない場合は、ジャンパパターンをカットする事で該当ピンを他の用途に使用できます。
- 端子
  ・ターミナル:モーター×2 外部電源×1
  ・TinkerKit互換アナログ入力端子×2 A2,A3
  ・TinkerKit互換アナログ出力端子×2 D5,D6
  ・TinkerKit互換TWI(I2C)端子入力×1 出力×1 SDA,SCL
  TWI端子は、Arduino R3以降のボードで使用できます。


RESET
ARDUINO
MOTOR
SHIELD

# LCD_Keypad_Shield

- LCD画面に文字を映すことができるシールド
- アナログ入力式のスイッチも使える


LCD Keypad Shield
VSS VDD VO RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
SELECT
LEFT
UP
DOWN
RIGTH
RST
PWR
R1
RST 5V VIN
A0 A1 A2 A3 A4 A5

# 性能（例）

- Arduino用のLCD＋キーパッドシールドです。キーパッドとして５個のタクトスイッチ（上下左右・セレクト）、リセット用タクトスイッチが実装されている。

- ＬＣＤ：16×2キャラクタディスプレイ[TC1602使用]
- ハイコントラスト青色バック白抜き液晶、白色LEDバックライト付き
- キーパッド：分圧抵抗による５ポジション設定(ArduinoのAD0のみを使用)
- 基板サイズ：80㎜×60㎜(実測値)


SELECT
LEFT
UP
DOWN
RIGHT
RST
DFROBOT
GND
5V

- 16文字2行LCDと5つのタクトスイッチを備えたArduinoシールドです。
- Arduino　Diecimila、Duemilanove、UNO、MEGA１２８０、ＭＥＧＡ２５６０で使用できます。
- 青地白抜きのバックライト付き液晶表示器で、視認性に優れています。
- ４bit　Arduino　LCD　ライブラリが使用できます。
- 5つのタクトスイッチは電圧の抵抗分割で認識しますので、「A0」１ポートのみの仕様です。

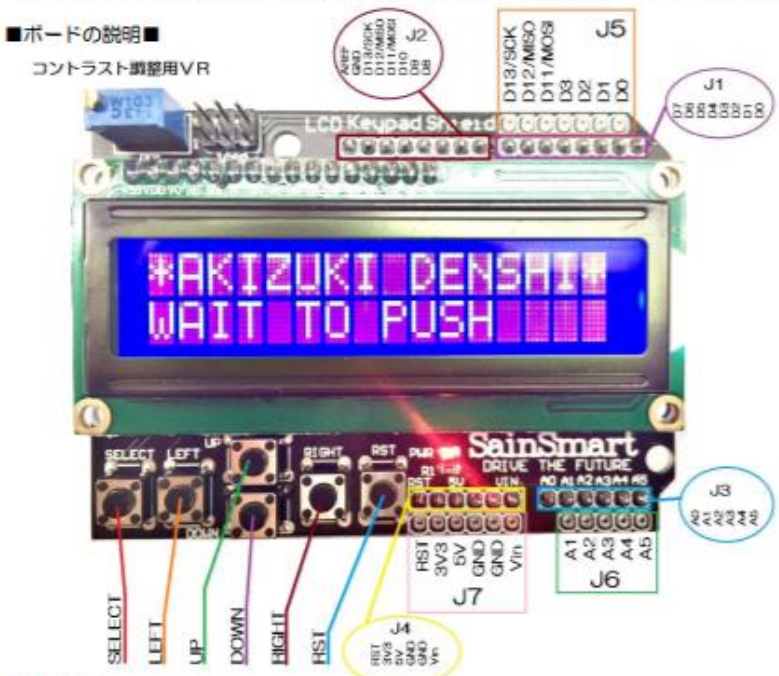
- リセットスイッチ「RST」　電源ＬＥＤ「PWR」付きです。


LCD Keypad Shield
Hello SainSmart.
SELECT
LEFT
UP
DOWN
RIGHT
RST
SainSmart
DRIVE THE FUTURE

# SainSmart
# LCD Kyepad Shield for Arduino

■特徴■
・１６文字２行ＬＣＤと５つのタクトスイッチを備えたＡｒｄｕｉｎｏシールドです。
・Ａｒｄｕｉｎｏ Ｄｉｅｃｉｍｉｌａ、Ｄｕｅｍｉｌａｎｏｖｅ、UNO、MEGA１２８０、
MEGA２５６０で使用できます。
・青地白抜きのバックライト付き液晶表示器で、視認性に優れています。
・４ｂｉｔ Ａｒｄｕｉｎｏ ＬＣＤ ライブラリが使用できます。
・５つのタクトスイッチは電圧の抵抗分割で認識しますので、「A0」１ポートのみの使用です。

■ボードの説明■




■使い方■
・Ｊ５、Ｊ６、Ｊ７が外部に引き出せる信号になっています。Ｊ１～Ｊ４はＡｒｄｕｉｎｏとの接続用です。
・ＬＣＤはD４～D９に接続され、４ｂｉｔデータでコントロールします。
・Ｋｅｙ（タクトスイッチ）は、A０に接続された抵抗で＋５Ｖを分割し、Ａ／Ｄコンバータの値を読み込んで
どのＫｅｙが押されているかを判断します。
・ご購入時には「コントラスト調整用ＶＲ」が調整されていませんので、何も表示しない事があります。
プログラムを走らせて表示させた状態でコントラストを調整してください。

サンプルは、メーカー提供のｚｉｐファイルを解凍し、「keypad_lcd」フォルダ内「keypad_1602for_1.0」
フォルダをArduinoインストールフォルダの「liblary」フォルダにコピーした後、下層「examples」フォル
ダ内「Key_Grab」を開き、スケッチを実行してください。

| Pin | Function |
| --- | --- |
| Analog 0 | Button (select, up, right, down and left) |
| Digital 4 | DB4 |
| Digital 5 | DB5 |
| Digital 6 | DB6 |
| Digital 7 | DB7 |
| Digital 8 | RS (Data or Signal Display Selection) |
| Digital 9 | Enable |
| Digital 10 | Backlit Control |

# LiquidCrystalライブラリ

- Arduinoボードを使って、日立HD44780(もしくは互換品)チップ搭載の液晶ディスプレイを制御するためのライブラリである。このチップは文字表示のためのほとんどの液晶ディスプレイによって採用されている。このライブラリは4ビットモードと8ビットモードのどちらでも動作する。すなわち、rs, enable,rw(rwはオプション)の制御ライン以外に4本もしくは8本のデータラインを使う。
- https://garretlab.web.fc2.com/arduino_reference/libraries/standard_libraries/LiquidCrystal/index.html

# 関数紹介（LCD_Keypad_Shield）

- LiquidCrystal()
- begin()
- setCursor()
- print()
- write()
- clear()

ライブラリはLiquidCrystal.h

# LiquidCrystal()

- LiquidCrystal型の変数を作成する。液晶ディスプレイは4本もしくは8本のデータラインで制御できる。4本のデータラインで制御するときは、d0からd3は未接続とする。RWピンはArduinoに接続せずに、接地してもいい。その場合は、この関数のパラメータのrwを省略する。
- LcdはLCDを表している。

# 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);

- rs
- 液晶ディスプレイのRSピンを接続したArduinoのピン番号
- rw
- 液晶ディスプレイのRWピンを接続したArduinoのピン番号(オプション)
- enable
- 液晶ディスプレイのenableピンを接続したArduinoのピン番号
- d0, d1, d2, d3, d4, d5, d6, d7
- 液晶ディスプレイのデータピンを接続したArduinoのピン番号。d0, d1, d2, d3は省略可能で、省略した場合は d4, d5,d6, d7の4本のデータラインで液晶ディスプレイを制御する。

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);

void setup()
{
  lcd.print("hello, world!");
}

void loop() {}
```

# begin()

- 液晶ディスプレイのインターフェイスを初期化する。また、ディスプレイの大きさ(幅と高さ)を指定する。begin()は、他のLCDライブラリの関数を利用する前に呼び出す必要がある。

cols　液晶ディスプレイの列数(1行あたりの文字数)
lines 液晶ディスプレイの行数
begin(cols,lines);

lcd.begin(16, 2);
16列2行のLCDを使用

# setCursor()

- 液晶ディスプレイのカーソル位置を設定する。以降の液晶ディスプレイへの文字はカーソルを設定した位置に出力される。

col　カーソル位置(列)　最初の列は0である。
row　カーソル位置(行)　最初の行は0である。

lcd.setCursor(col, row);

lcd.setCursor(0, 0);
0,0の位置に文字をセット
左基準なら一番左側になる。
左から右へ表示される。

# print()

- 液晶ディスプレイに文字列を書く

data 液晶ディスプレイに書き込む文字。
base 数字を表示する際の底。
　BIN　2進数
　DEC 10進数
　OCT 8進数
　HEX 16進数
digits小数点以下の桁数

lcd.print(data,base,digits);

lcd.print(inTime_m);
inTime_mをlcdに表示
lcd.print(" ");
1マスの空白を表示

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);

void setup(){
  lcd.print("hello, world!");
}

void loop() {}
```

# write()

- 液晶ディスプレイに文字を書く。

value液晶ディスプレイに書き込む文字。

lcd.write(value);

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 10, 5, 4, 3, 2);

void setup(){
  Serial.begin(9600);
}

void loop(){
  if (Serial.available()) {
    lcd.write(Serial.read());
  }
}
```

# clear()

- 液晶ディスプレイの表示を消し、カーソルを左上隅に配置する。

lcd.clear()

# 例

- 次の例は1秒ごとに数値を1繰り上げたものをLCDに表示するプログラムである。
- スイッチの処理を追加することで押したときのスイッチを表示することもできるようになる。

- #include <LiquidCrystal.h>

- LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

- #define btnNONE  1
- #define btnRIGHT  2
- #define btnUP  3
- #define btnDOWN  4
- #define btnLEFT  5
- #define btnSELECT 6

- int adc_key_in;
- int lcd_key;

```
int read_LCD_buttons() {
  adc_key_in = analogRead(0);
  if (adc_key_in > 1000) return btnNONE;
  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;
}

void setup() {
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("Push the buttons");
}
```

```
void loop() {
  lcd.setCursor(9, 1);
  lcd.print(millis() / 1000);

  lcd.setCursor(0, 1);
  lcd_key = read_LCD_buttons();
```

```
  switch (lcd_key)  {
    case btnRIGHT:
      lcd.print("RIGHT ");
      break;
    case btnLEFT:
      lcd.print("LEFT   ");
      break;
    case btnUP:
      lcd.print("UP    ");
      break;
    case btnDOWN:
      lcd.print("DOWN  ");
      break;
    case btnSELECT:
      lcd.print("SELECT");
      break;
    case btnNONE:
      lcd.print("NONE  ");
      break;
   }
}
```

- 次の例はスイッチを押したときの数値をLCDで表示するプログラムである。

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
#define Switch A0
int analoginput = 0;

void setup() {
   lcd.begin(16, 2);              // start the library    LCDは16*2マス
 lcd.setCursor(0,0);               //上から0    左から0
 lcd.print("Analogread"); // print a simple message    何を表示するか
}

void loop() {
 analoginput = (analogRead(Switch)/4);
 lcd.setCursor(0,1);
 lcd.print(analoginput);
 lcd.print("    ");
}
```

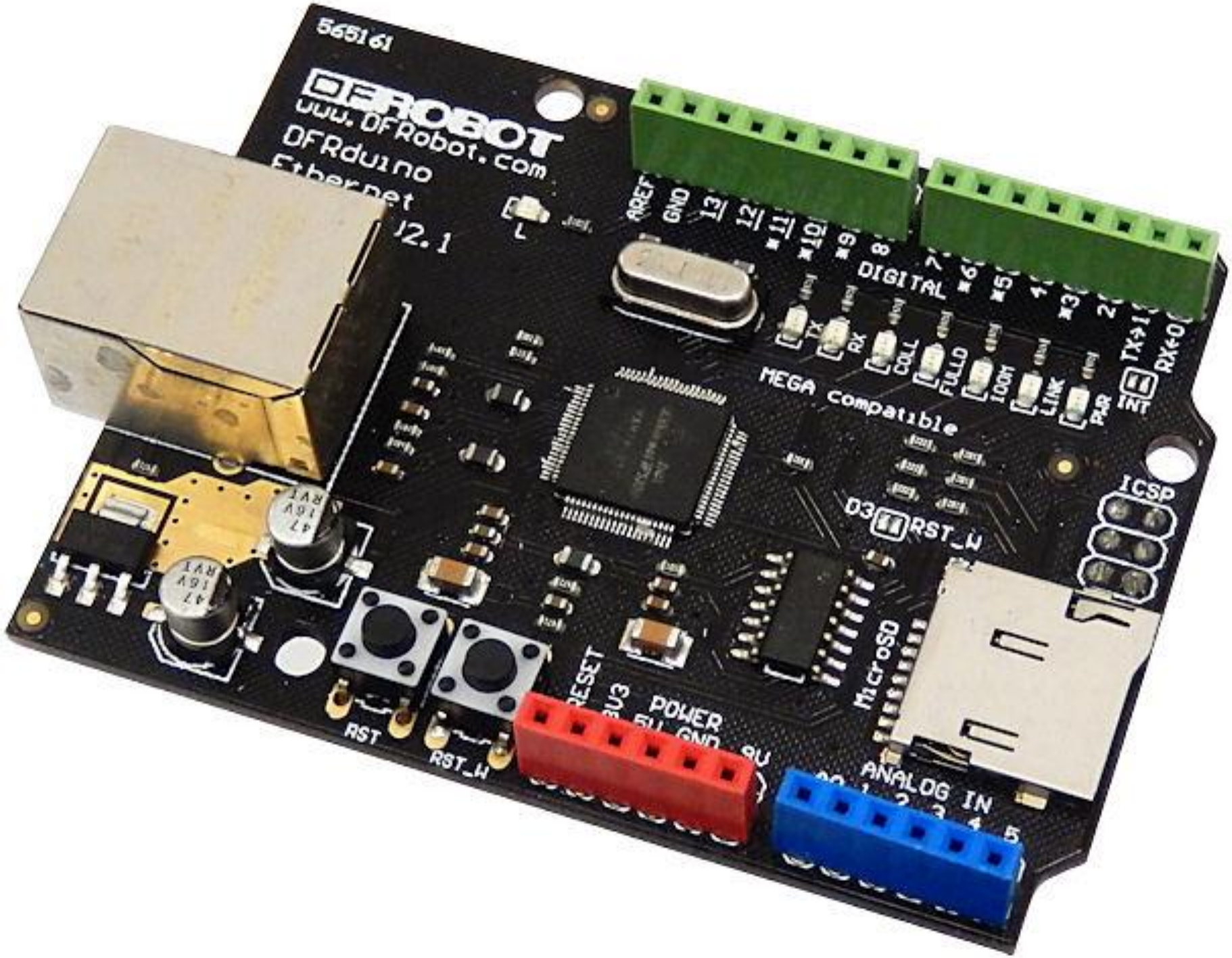
# イーサネットシールド

- LANケーブルを繋いでネットに接続できるシールド
- マイクロSDを挿せるものがある。

ANALOG IN
A5 4 3 2 1 0 A0 9V Gnd 5V 3V3 RESET
WP
RESET
CS 47 25V
CS 47 25V
HanRun
HR911105A
09/44
ICSP
MEGA compatible
PWR
LINK
100M
FULLD
COLL
RX
TX
INT
TX RX
0 1 2 3 4 5 6 7 8 9 10 11 12 13 GND AREF

# 性能（例）

- イーサネットシールドの主な仕様
  ・対応Ａｒｄｕｉｎｏ：ＵＮＯ、Ｍｅｇａ（１２８０＆２５６０）、Ｄｕｅｍｉｌａｎｏｖｅ
  ・イーサネットポート：ＲＪ－４５(緑／黄ＬＥＤ付き)
  ・外部メモリ：マイクロＳＤカードスロット
  ・通信ステータスＬＥＤ：７個
  ・リセットスイッチ：Ａｒｄｕｉｎｏ用、Ｗ５１００用
  ・基板サイズ：５６．２×４７．３ｍｍ（実測値）
  ※ＭＡＣアドレスは別途御用意する。


565161
DFROBOT
www.DFRobot.com
DFRduino
Ethernet
V2.1
MEGA compatible
DIGITAL
AREF
GND
13
12
11
10
9
8
7
6
5
4
3
2
TX→1
RX←0
TX
RX
TOOL
FULLD
100M
LINK
PWR
INT
ICSP
D3
RST_W
RESET
3V3
POWER
5V
GND
9V
RST
RST_W
MicroSD
ANALOG IN
0
1
2
3
4
5

# 関数紹介（イーサネット）

- EthernetServer()
- begin()

あとは省略

https://garretlab.web.fc2.com/arduino_reference/libraries/standard_libraries/Ethernet/index.html

# EthernetServer()

- 指定したポートへの接続を待ち受けるサーバを作成する。

port　待ち受けるポート
EthernetServer g_server(port);

EthernetServer g_server(80);

# begin()

- 入力される接続の待ち受けを開始するようサーバに指示する。

- g_server.begin();

# 例

- 次の例はパソコン画面からLEDを制御するためのプログラムである。
- 非常にわかりにくいものであるので調べたほうが早いかもしれない。

```
#include <SPI.h>    // for Ethernet.h
#include <Ethernet.h> // for Ethernet

#define LED1PIN (7)
#define LED2PIN (8)
#define LED3PIN (9)

#define LINEBUFFERSIZE  (128)
#define DELIMITER ("&")

unsigned char MACADDRESS[] = { 0x8C, 0x73, 0x6E, 0xE0, 0xB1, 0xED };//自分のPCのMACアドレスでできた。
unsigned char IPADDRESS[] = { 172, 17, 25, 199 };
EthernetServer g_server(80);

void setup(){
  Ethernet.begin(MACADDRESS, IPADDRESS);
  g_server.begin();
  pinMode(LED1PIN, OUTPUT);
  pinMode(LED2PIN, OUTPUT);
  pinMode(LED3PIN, OUTPUT);
//  Serial.begin(9600);
}
```

```
// HTML出力
void PrintHtml( EthernetClient& client, boolean led1, boolean led2, boolean led3, int speed ){
  client.println("HTTP/1.1 200 OK");
  client.println("Content-Type: text/html");
  client.println();

  client.println("<!DOCTYPE HTML PUBLIC ¥"-//W3C//DTD HTML 4.01 Transitional//EN¥"
¥"http://www.w3.org/TR/html4/loose.dtd¥">");
  client.println("<html lang=¥"ja¥">");
  client.println("<head>");
  client.println("<meta http-equiv=¥"Content-Type¥" content=¥"text/html; charset=UTF-8¥">");
  client.println("<meta http-equiv=¥"Content-Style-Type¥" content=¥"text/css¥">");
  client.println("<title></title>");
  client.println("</head>");
  client.println("<body>");

  client.println("<p>");
  client.print("LED1 : "); client.print( led1 ? "ON" : "off" ); client.println("<br>");
  client.print("LED2 : "); client.print( led2 ? "ON" : "off" ); client.println("<br>");
  client.print("LED3 : "); client.print( led3 ? "ON" : "off" ); client.println("<br>");
  client.println("<br>");
  client.println("速さ：");
  client.println(speed);
  client.println("</p>");

  client.println("<hr>");
```

```
  client.println("<form method=¥"post¥" action=¥"¥">");
  client.print("<input type=¥"checkbox¥" name=¥"led1¥" value=¥"1¥"""); if(led1){ client.print(" checked"); } client.println(">LED1<br>");
  client.print("<input type=¥"checkbox¥" name=¥"led2¥" value=¥"1¥"""); if(led2){ client.print(" checked"); } client.println(">LED2<br>");
  client.print("<input type=¥"checkbox¥" name=¥"led3¥" value=¥"1¥"""); if(led3){ client.print(" checked"); } client.println(">LED3<br>");
  client.println("<br>");
  client.println("速さ<br>");
  client.println("1 ");
  client.print("<input type=¥"radio¥" name=¥"speed¥" value=¥"1¥"""); if(1==speed){ client.print(" checked"); } client.println("> ");
  client.print("<input type=¥"radio¥" name=¥"speed¥" value=¥"2¥"""); if(2==speed){ client.print(" checked"); } client.println("> ");
  client.print("<input type=¥"radio¥" name=¥"speed¥" value=¥"3¥"""); if(3==speed){ client.print(" checked"); } client.println("> ");
  client.print("<input type=¥"radio¥" name=¥"speed¥" value=¥"4¥"""); if(4==speed){ client.print(" checked"); } client.println("> ");
  client.print("<input type=¥"radio¥" name=¥"speed¥" value=¥"5¥"""); if(5==speed){ client.print(" checked"); } client.println("> ");
  client.print("<input type=¥"radio¥" name=¥"speed¥" value=¥"6¥"""); if(6==speed){ client.print(" checked"); } client.println("> ");
  client.print("<input type=¥"radio¥" name=¥"speed¥" value=¥"7¥"""); if(7==speed){ client.print(" checked"); } client.println("> ");
  client.print("<input type=¥"radio¥" name=¥"speed¥" value=¥"8¥"""); if(8==speed){ client.print(" checked"); } client.println("> ");
  client.print("<input type=¥"radio¥" name=¥"speed¥" value=¥"9¥"""); if(9==speed){ client.print(" checked"); } client.println("> ");
  client.println(" 10<br>");
  client.println("<br>");
  client.println("<input type=¥"submit¥" value=¥"送信¥">");
  client.println("<input type=¥"reset¥" value=¥"リセット¥"></form>");
  client.println("<hr>");

  client.println("</body>");
  client.println("</html>");
}
```

```
boolean AnalyzeLineString( char* pszLine, boolean& rbLed1, boolean& rbLed2,
boolean& rbLed3, int& riSpeed ){
  char* pszToken = strtok(pszLine, DELIMITER);
  while(pszToken) {
    if( 6 == strlen(pszToken)
     && 0 == strncmp(pszToken, "led", 3) ) { // led?=1
      if(     '1' == pszToken[3] ){ rbLed1 = true; }
      else if( '2' == pszToken[3] ){ rbLed2 = true; }
      else if( '3' == pszToken[3] ){ rbLed3 = true; }
    }
    else if( 7 == strlen(pszToken)
     && 0 == strncmp(pszToken, "speed", 5) ) { // speed=?
      riSpeed = pszToken[6] - '0';
    }
    pszToken = strtok(NULL, DELIMITER);
  }
  return true;
}
```

```
boolean readHttpRequest( EthernetClient& client, boolean& rbLed1, boolean& rbLed2, boolean& rbLed3, int& riSpeed ){
  if( !client ) {
    return false;
  }

  // HTTPリクエスト空行（¥r¥n¥r¥n）で終わる。ので、空行を探す。
  boolean foundRNRN = false;
  boolean currentLineIsBlank = true;
  char szLine[LINEBUFFERSIZE];
  int iIndex = 0;
  while( client.connected() ) {
    if( !client.available() )   {
      //continue;
      break;
    }

    char c = client.read();
```

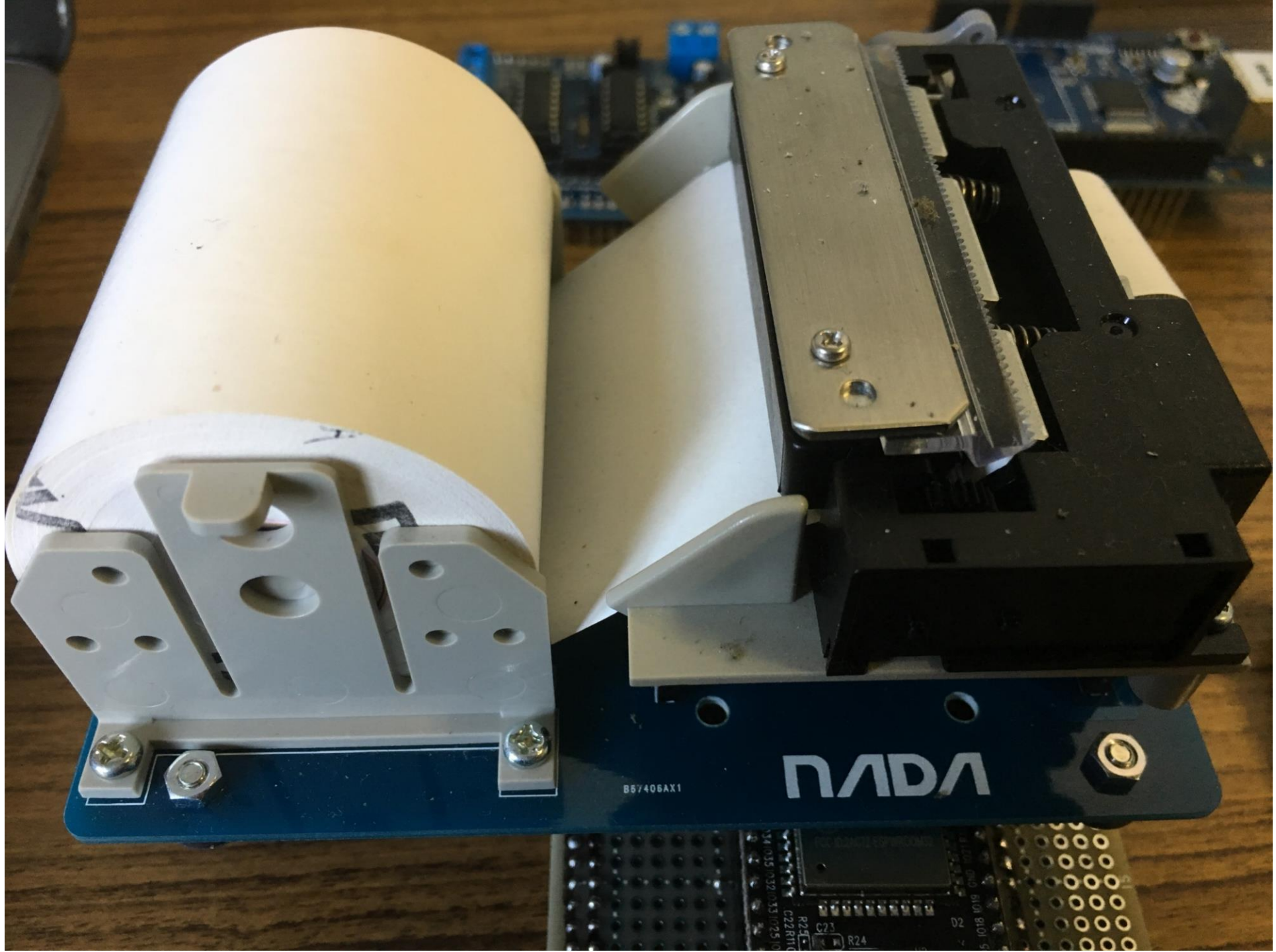
```
//    Serial.print(c);
      if( '¥n' == c && currentLineIsBlank ) { // 空行発見。
        foundRNRN = true;
      }
      if( '¥n' == c ) {
        // httpリクエストの１行の解析
        if( foundRNRN   && 0 != iIndex ) {
          szLine[iIndex] = '¥0';
          AnalyzeLineString( szLine, rbLed1, rbLed2, rbLed3, riSpeed );
        }
        // 新しい行の始まり。
        currentLineIsBlank = true;
        iIndex = 0;
      }
      else if( '¥r' != c ) { // この行は空行ではなかった。
        currentLineIsBlank = false;
        if( foundRNRN ) {
          if( LINEBUFFERSIZE - 1 > iIndex ) {
            szLine[iIndex] = c;
            ++iIndex;
          }
        }
      }
    }
```

```
    // httpリクエストの終端行の解析
    if( foundRNRN  && 0 != iIndex ) {
      szLine[iIndex] = '¥0';
      AnalyzeLineString( szLine, rbLed1, rbLed2, rbLed3, riSpeed );
    }
    return foundRNRN;
  }

  void loop(){
    static boolean s_led1 = false;
    static boolean s_led2 = false;
    static boolean s_led3 = false;
    static int s_interval = 0;

    EthernetClient client = g_server.available();
    boolean led1 = false;
    boolean led2 = false;
    boolean led3 = false;
    int speed = 5;
    if( readHttpRequest(client, led1, led2, led3, speed) ) {
      // Html出力
      PrintHtml(client, led1, led2, led3, speed);

      // Webブラウザに対して、データを取得するための時間を与える。
      delay(1);

      // コネクションを閉じる。
      client.stop();
```

```
  s_led1 = led1;
    s_led2 = led2;
    s_led3 = led3;
    // speed        |   1         9
    // 間隔[秒]     |   0.5       0.01
    // interval[ms] | 500         10
    s_interval = (int)(500L + (speed - 1) * (10L - 500L) / (9-1));
  }

  if( s_led1 ){ digitalWrite(LED1PIN, HIGH); }
  if( s_led2 ){ digitalWrite(LED2PIN, HIGH); }
  if( s_led3 ){ digitalWrite(LED3PIN, HIGH); }
  delay(s_interval);
  digitalWrite(LED1PIN, LOW);
  digitalWrite(LED2PIN, LOW);
  digitalWrite(LED3PIN, LOW);
  delay(s_interval);
}
```

# サーマルプリンターシールド

- サーマル（感熱）式のプリンター
- ArduinoのSerial通信で印刷ができる。
- ライブラリは特に必要ない。
- 電圧は5 Vで良いが、印刷する文字が多い場合や画像、バーコードを印刷する際は大電流が必要。（4 Aあるのが理想）
- 電源コードをACアダプターから自作する必要がある。
  (2~4 Aのものを使用)
- ライブラリがあるけど私は特に使用していない。
- シリアル通信でデータを受信して印刷をすることが可能。
- http://www.nada.co.jp/as289r2/


B57406AX1
NADA

## 仕様

| | |
|---|---|
| 型番 | AS-289R |
| RoHS | 対応 |
| 入力電圧 | 5V±5% |
| 平均電流 | 約2A (印字率25% 2分割時) |
| I/F | シリアル (TTL, 9600bps, 8Bit, NonParity, 1Stop) |
| 印字方式 | ラインサーマル方式 |
| ドット総数 | 384ドット/ライン |
| ドット密度 | 8ドット/mm |
| 印字有効幅 | 48mm |
| 印字速度 | 約25mm/秒 |
| 紙送り速度 | 約25mm/秒 |

| | |
|---|---|
| 文字コード | UTF-8 |
| 文字種類<br>及び<br>印字桁数 | 電源投入時は、ANK12×24、漢字24×24が初期値となります |

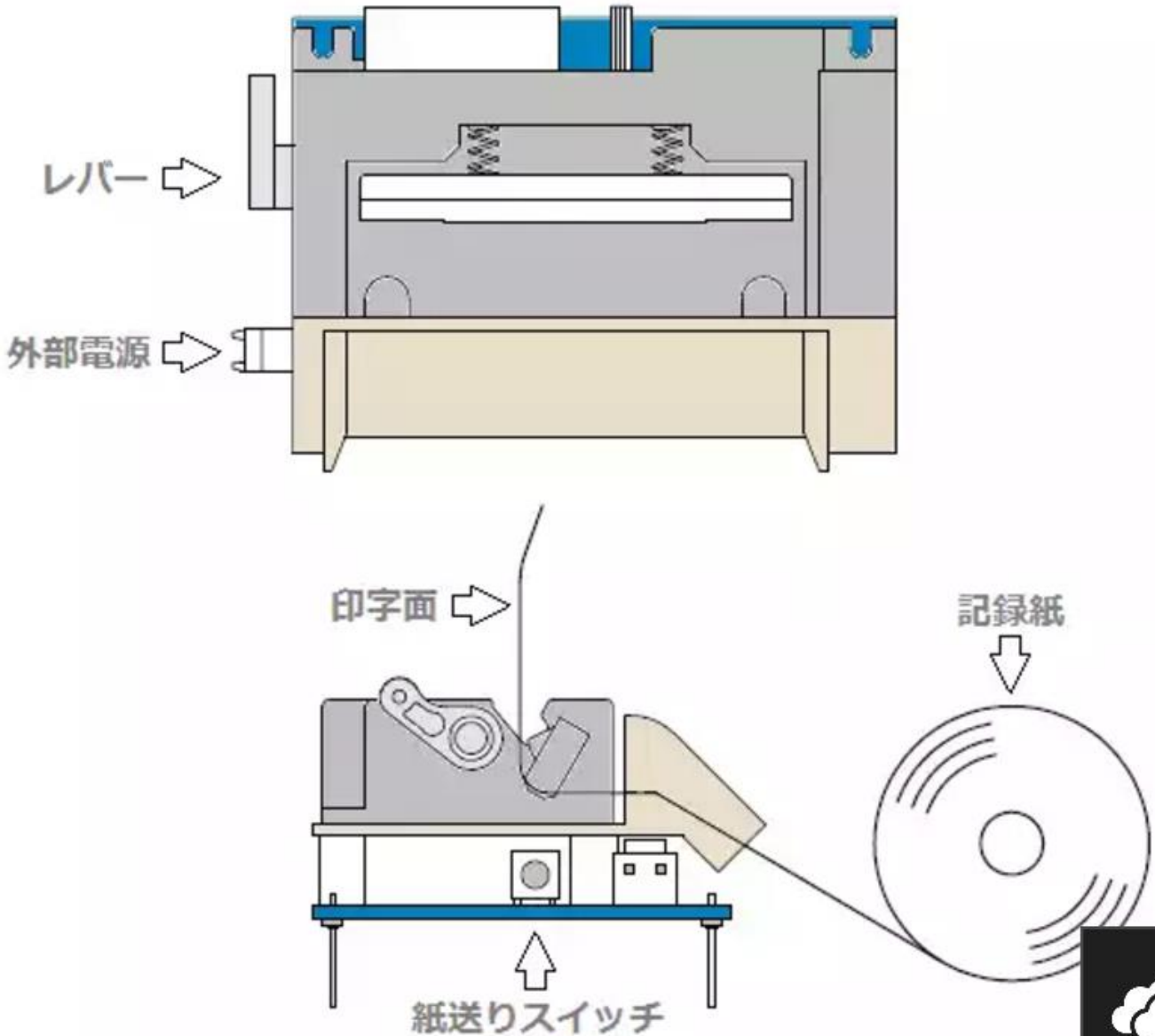
| | 文字寸法 | 桁数 | 桁間 |
|---|---|---|---|
| ANK JIS160文字 | 8×16 | 42 | 1 |
| | 12×24(ANK初期値) | 32 | 0 |
| | 16×16 | 24 | 0 |
| | 24×24 | 16 | 0 |
| 漢字 JIS C 6226-1983準拠<br>第一水準漢字2965文字<br>第二水準漢字3388文字<br>JIS非漢字524文字 | 16×16 | 24 | 0 |
| | 24×24(漢字初期値) | 16 | 0 |
| 大文字 0～9,A～Zの36文字 | 48×96 | 8 | 0 |

| | |
|---|---|
| バーコード | QR、JAN(13,8)、2of5(ITF)、2of7(NW7)、3of9(CODE39)、UPC-A |
| 重量 | シールド基板 25g<br>プリンタメカ 65g ※記録紙含まず |
| 動作環境 | 温度 0～50℃(但し印字保証は5～40℃)<br>湿度 35%～80%RH(非結露) |

| | | |
|---|---|---|
| 信頼性 | 常温25℃、常湿60%RH、印字率12.5%、記録紙NP-580使用時 | |
| | 寿命 | 600万行<br>磨耗故障時期に入り始めるポイントです |
| | MCBF | 1500万行<br>寿命600万行に至るまでの磨耗系故障、<br>偶発的故障を含めた総合的な平均故障間隔を表す |
| | ヘッド寿命 | 耐パルス性：1億パルス以上<br>耐磨耗性：50km以上 |
| 外形寸法 | 【PDF】AS-289R(シールド基板+プリンタメカ)外形図<br>【PDF】AS-289R(シールド基板)外形図<br>【PDF】AS-289R(プリンタメカ)外形図 | |

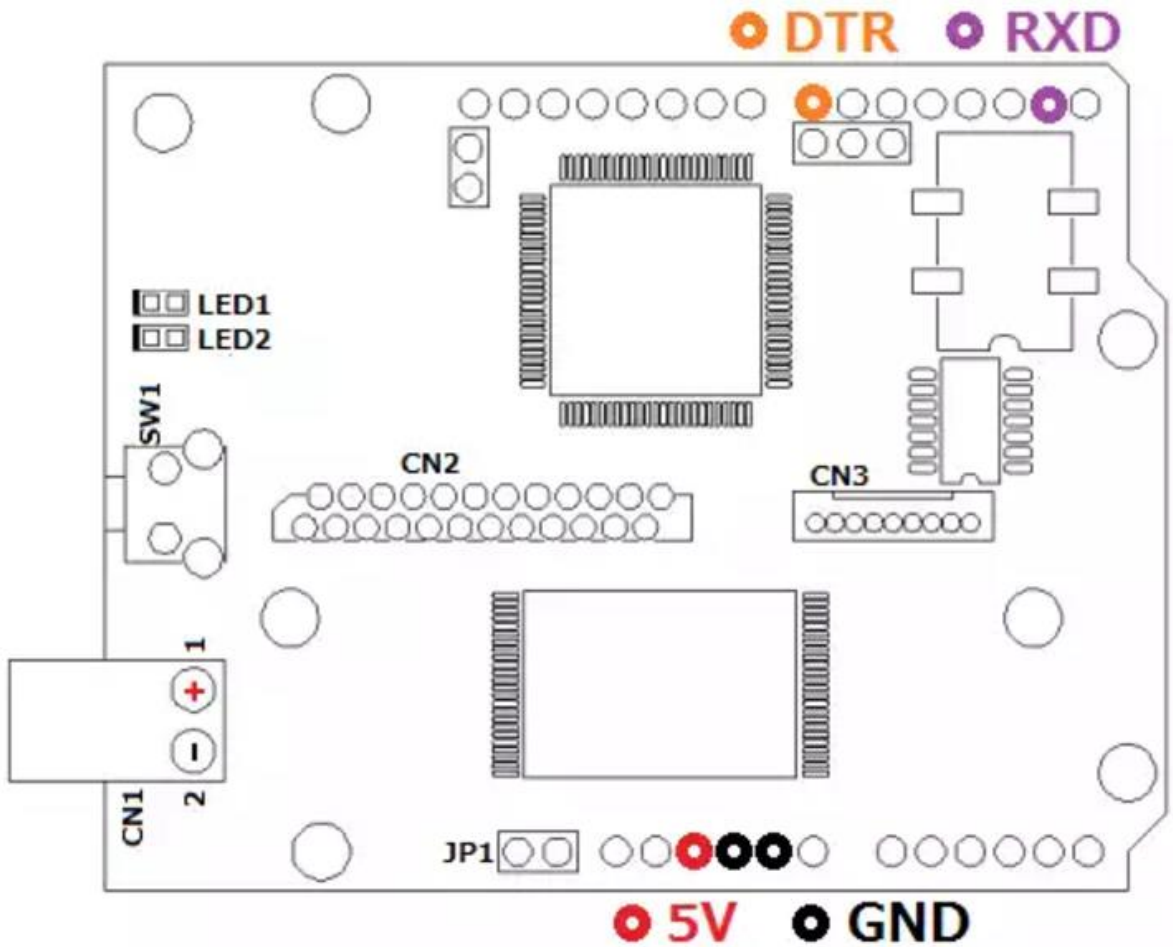
| 【付属品】 | |
|---|---|
| 記録紙 | 1巻　　型番：NP-580 (サーマルペーパー 幅:58mm 巻径:50mm 長さ:30m) |
| 電源ハーネス | 1本　　型番：CB-1116 (日本圧着端子製：VHR-2N 赤/黒 長さ:1m) |
| 記録紙ホルダ | 1対　　型番：ND131-108<br>【PDF】記録紙ホルダ外形図 |

プリンタメカ+基板




※記録紙ホルダについては【PDF】記録紙ホルダ外形図を参照して下さい

Arduinoとの接続ピンは、下記5箇所となっています。




**※5V端子は、JP1により切り替わります**

| | | |
|---|---|---|
| CN1 | 外部電源コネクタ 1番ピン:5V 2番ピン:GND | 日本圧着端子製 B2P-VH |
| CN2 | プリンタメカ接続端子 | MOLEX 52806-2410 |
| CN3 | プリンタメカ接続端子 | MOLEX 53047-0910 |
| SW1 | 紙送りスイッチ | |
| LED1 | 電源LED(緑色) | |
| LED2 | ステータスLED(アンバー色) | |
| JP1 | JP1をブリッジすると、外部電源コネクタ(CN1)の5VとAruduinoの5Vが接続されます | |

## 使用上の注意事項

1．記録紙がセットされていない状態で印字すると プリンタが損傷する可能性がある為、必ず記録紙をセットしてから印字を行ってください。
2．メカ部の主な板金部（プレス部）は メッキ鋼板を使用している為、端面に多少の錆が発生する場合があります。
3．低温時で使用する場合、印字開始時にサーマルヘッドが冷えているため、初期印字が薄くなることがあります。
4．高温時で使用する場合、印字ににじみ等が発生する場合があります。
5．連続印字（特に黒ベタ、千鳥格子）を行うと記録紙から発生する水蒸気により記録紙が汚染される場合があります。
6．製品に振動が加わる場所での使用はご相談ください。微弱な振動でも長時間加わると直接的な障害の他に二次的障害により予想外の不具合が発生する場合があります。
7．製品の周辺にて無線機を使用する場合、無線ノイズにより誤作動する場合があります。

## 禁止事項

1．印字中及び印字終了直後は、サーマルヘッド近傍、モータ表面は高温になるため、直接触れないこと。
2．結露状態での使用は行ってはならない。もし結露した場合は、結露がなくなるまでプリンタに通電しないこと。
3．記録紙及びプラテンに異物などの付着のないこと。
4．ヘッドダウン状態での紙の引き抜き（正逆方向）は行わないこと。

```
void setup(){
  Serial.begin(9600);
}
void loop(){
 /* Text Print */
 Serial.print("Thermal Printer Shield¥r");
 Serial.print("Text Printing.¥r");
 Serial.print("¥r¥r");     // Line Feed x 2
 /* QRcode Print */
 byte GsQr[] = { 0x1D,0x78,0x4C,0x04,0x54,0x45,0x53,0x54 };
 Serial.write(GsQr, 8);
 Serial.print("¥r¥r¥r¥r¥r¥r");  // Line Feed x 6
 /* Wait */
 delay(5000);  // 5sec
}
```

```
#include <AS289R2.h>
HardwareSerial monit = HardwareSerial(0);
HardwareSerial tprinter = HardwareSerial(2);
unsigned long moji = 0;
void setup() {
 monit.begin(250000);
 tprinter.begin(38400);
}
void loop() {
  if (monit.available() > 0) {
    moji = monit.read();
    monit.write(moji);
    tprinter.write(moji);
  }
}
```

# ライブラリ

- Arduinoのプログラミングを楽にするために使用される。
- <・・・.h>を使うことで対応する関数が使用可能になる。
  （・・・はライブラリ名）
- LCDやサーボなどはライブラリがないと使用できない
  （もしくはプログラミングがめちゃくちゃめんどくさくなる）
- Arduinoが簡単だとされている1つの要因
- 標準以外にもユーザー提供のものがある。

- ライブラリは、特別な機能を提供するもので、スケッチから利用する。例えば、ハードウェアの利用やデータの操作を簡単に行うことができます。ライブラリをスケッチから利用するには、Arduinoソフトウェアのツールバーから、Sketch →Import Library を選択する。
- ライブラリには標準ライブラリ、Esplora専用ライブラリ、ユーザ提供ライブラリの3つがある。

# Esplora専用ライブラリ

- Arduino Esploraには、ボードに搭載されたセンサやアクチュエータと簡単にインターフェイスをとるための関数群がある。Esploraクラスを通じて関数は利用可能である。
- ライブラリは、ボード上のセンサからのデータの簡単な読み取り手段と、出力状態の変更手段を提供する。
- ボードで利用できるセンサーは次の通り。

- 2軸アナログジョイスティック
- ジョイスティックのセンター押しボタン
- 4個の押しボタン
- マイク
- 光センサ
- 温度センサ
- 3軸加速度センサ
- 2個のTinkerKit入力端子

- ボードで利用できるアクチュエータは以下の通り。

- 明るいRGB LED
- ピエゾブザー
- 2個のTinkerKit出力端子

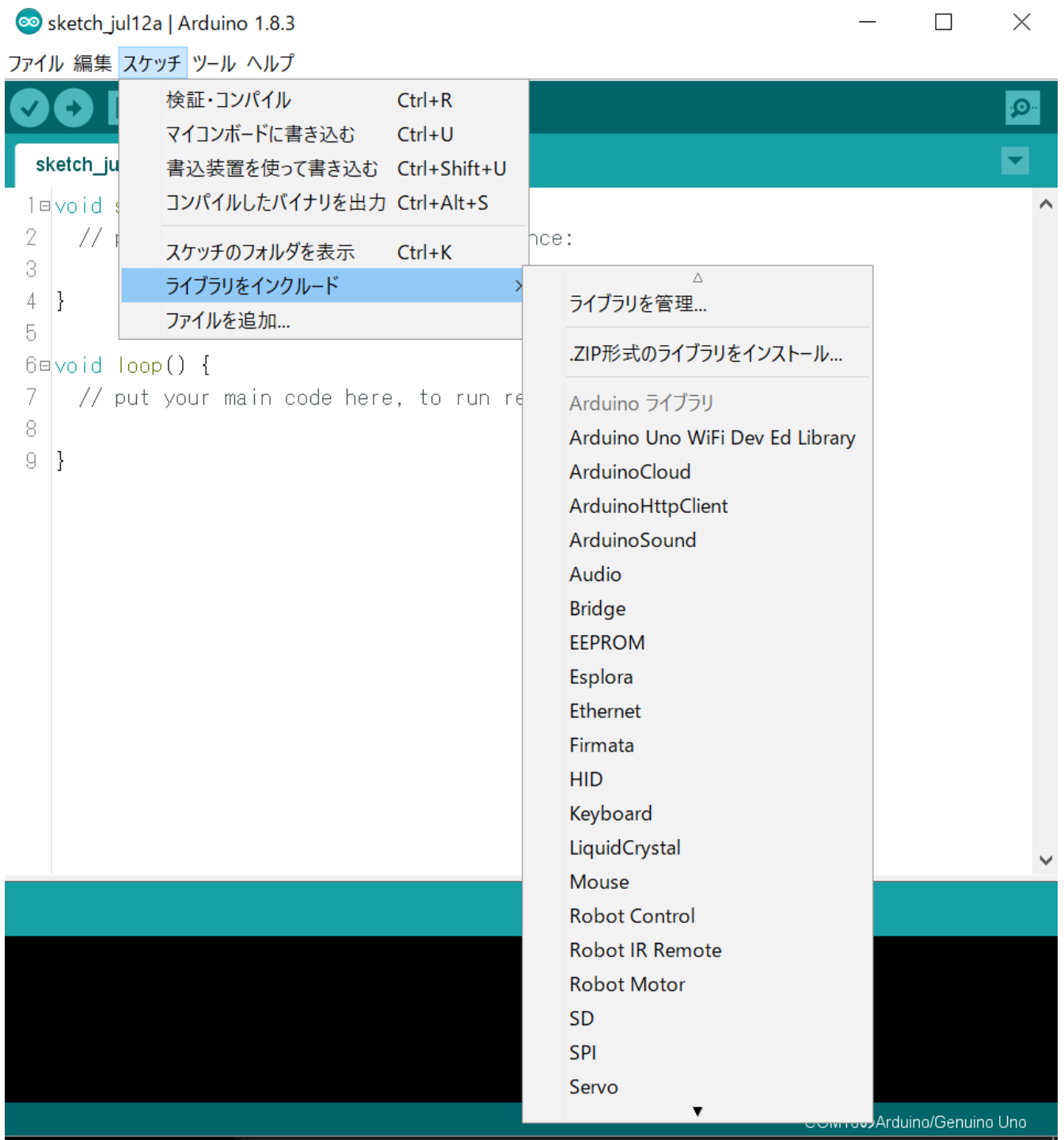
# ユーザ提供ライブラリ

- ユーザ提供ライブラリを利用するためには、まずインストールする必要がある。インストール方法は以下の通り。
- ライブラリをダウンロードする。
- ダウンロードしたライブラリを展開する。展開したフォルダには、通常、拡張子が.hのヘッダファイルと.cppのプログラム本体とが最低限含まれる。
- Arduinoのスケッチがあるフォルダを開く。そこにlibrariesというフォルダがなければ作成する。ダウンロードして展開したフォルダを、libraries配下に配置する。
- Arduinoをインストールしたフォルダに、librariesというフォルダがあるので、展開したフォルダをそこに配置する。
- Arduino環境を再起動する。
- Sketch → Import Library　メニューに、インストールしたライブラリが現れる。

- 標準ライブラリ
- https://garretlab.web.fc2.com/arduino_reference/libraries/standard_libraries/index.html
- Esplora専用ライブラリ
- https://garretlab.web.fc2.com/arduino_reference/libraries/esplora_only_library/index.html
- ユーザ提供ライブラリ
- https://garretlab.web.fc2.com/arduino_reference/libraries/contributed_libraries/index.html

# ライブラリを追加

- スケッチ→ライブラリをインクルード→「使いたいライブラリ」


sketch_jul12a | Arduino 1.8.3
ファイル 編集 スケッチ ツール ヘルプ
検証・コンパイル Ctrl+R
マイコンボードに書き込む Ctrl+U
書込装置を使って書き込む Ctrl+Shift+U
コンパイルしたバイナリを出力 Ctrl+Alt+S
スケッチのフォルダを表示 Ctrl+K
ライブラリをインクルード
ファイルを追加...
ライブラリを管理...
.ZIP形式のライブラリをインストール...
Arduino ライブラリ
Arduino Uno WiFi Dev Ed Library
ArduinoCloud
ArduinoHttpClient
ArduinoSound
Audio
Bridge
EEPROM
Esplora
Ethernet
Firmata
HID
Keyboard
LiquidCrystal
Mouse
Robot Control
Robot IR Remote
Robot Motor
SD
SPI
Servo
1 void
2 // p
3
4 }
5
6 void loop() {
7 // put your main code here, to run re
8
9 }
nce:
Arduino/Genuino Uno

# ライブラリのインストール、更新

- スケッチ→ライブラリをインクルード→ライブラリを管理→ライブラリ名を検索→選択しダウンロード

- スケッチ→ライブラリをインクルード→ライブラリを管理→タイプからアップデート可能を選択→ダウンロード

ライブラリマネージャ

タイプ 全て トピック 全て 検索をフィルタ…

**Arduino Low Power** by **Arduino**
**Power save primitives features for SAMD and nRF52 32bit boards** With this library you can manage the low power states of newer Arduino boards
More info

**Arduino SigFox for MKRFox1200** by **Arduino**
**Helper library for MKRFox1200 board and ATAB8520E Sigfox module** This library allows some high level operations on Sigfox module, to ease integration with existing projects
More info

**Arduino Uno WiFi Dev Ed Library** by **Arduino** バージョン**0.0.3** **INSTALLED**
**This library allows users to use network features like rest and mqtt. Includes some tools for the ESP8266.** Use this library only with Arduino Uno WiFi Developer Edition.
More info

**ArduinoCloud** by **Arduino** バージョン**1.0.0** **INSTALLED**
**Easly connect your Arduino/Genuino board to the Arduino Cloud** Easly connect your Arduino/Genuino board to the Arduino Cloud
More info

**ArduinoHttpClient** by **Arduino** バージョン**0.3.0** **INSTALLED**
**[EXPERIMENTAL] Easily interact with web servers from Arduino, using HTTP and WebSocket's.** This library can be used for HTTP (GET,

閉じる

- GitHub等で入手したライブラリを入れることもできる。
- ライブラリマネージャにない場合は自分で探して入れる。
- その場合は自分でどこに入れるかを探して見つけないとライブラリとして使えないため少し難しく感じる。
- 入れる場所としては
- Program Files→Arduino→librariesのような感じである。
- 画像を用意するのが面倒でした。

# 関数紹介（ライブラリインクルード）

- #include

# #include

- #includeは、スケッチ外のライブラリを挿入するために利用される。標準Cライブラリ(あらかじめ定義された関数群)やArduino専用に書かれたライブラリを利用することができる。
- #define同様、#includeも文の最後にセミコロンを書かない。

# ライブラリ紹介

- **LiquidCrystal.h（LCD_Keypad_Shieldで紹介済み）**
- **Servo.h**
- **SoftWareSerial.h**

# Servo.h

- このライブラリを使うとArduinoボードを使ってRCサーボモータを制御することができる。サーボは精密に制御できるギアとシャフトを統合したものである。一般的なサーボはシャフトを様々な角度にすることができ、それは、通常0度から180度である。連続回転サーボはシャフトの回転速度を制御することができる。
- サーボライブラリは、ほとんどのArduinoボードで12個のモータを制御できる。Arduino Megaでは48個のモータを制御できる。Mega以外のボードでは、サーボライブラリを使うと、サーボを接続していなくても、9番ピンと10番ピンのanalogWrite()(PWM)が使えなくなる。Megaでは、12個までPWMの機能を妨げずに利用できる。12から23までのモータを使うと11番ピンと12番ピンのPWMが使えなくなる。

- サーボモータは、電源、アース、制御信号の3線を持っている。電源線は多くの場合赤で、Arduinoボードの5Vピンに接続する。アース線は多くの場合黒か茶色で、ArduinoボードのGNDピンに接続する。制御信号線は黄色かオレンジ、白で、Arduinoボードのデジタルピンに接続する。サーボは大量の電力を消費するので、1個か2個よりも多くのサーボを制御する場合は、Arduinoの5Vピンではなく、外部から電源を供給する必要がある。この場合Arduinoと外部電源のアースを接続することを忘れないように。
- https://garretlab.web.fc2.com/arduino_reference/libraries/standard_libraries/Servo/index.html

# 関数紹介（Servo）

- attach
- write
- read
- detach

# attach

- Servo変数をピンに割り当てる。Arduino0016とそれ以前のバージョンでは、サーボライブラリは9番ピンと10番ピンの2つのピンしか使えない。

- attach(pin);
- pin　サーボが接続されているピン番号

```
void setup() {
  myservo1.attach(3);  //myservo1をピン3に割り当てる
  myservo2.attach(11); //myservo2をピン11に割り当てる
}
```

```
#include <Servo.h>

Servo myservo;

void setup() {
  myservo.attach(9);
}

void loop() {}
```

# write

- サーボに値を書き込み、シャフトの制御を行う。通常のサーボではシャフトの角度(度数法)を設定し、シャフトをその方向に動かす。連続回転サーボでは、サーボのスピードを設定する。この場合0がある方向での最大スピードで、180が逆方向への最大スピード、90付近が停止である。

- write(value)
- value　サーボに設定する値。0から180

```
void loop() {
 myservo1.write(90);  //myservo1を90にする（中心）
 myservo2.write(90);  //myservo2を90にする（中心）
}
```

```
#include <Servo.h>

Servo myservo;

void setup() {
  myservo.attach(9);
  myservo.write(90);  // set servo to mid-point
}

void loop() {}
```

# read

- 現在のサーボの値(最後にwrite()に渡した値)を読み出す。

- myservo.read();
- myservoは自分で作る

# detach

- サーボ変数をピンから切り離す。すべてのサーボ変数が切り離されると、9番ピンと10番ピンをanalogWrite()を使ってPWM出力に利用することができる。

- myservo.detach();
- myservoは自分で作る

# 例

- 次の例はLCDキーパッドシールドを使ってサーボを動かすプログラムである。
- 変な書き方をしているが問題はない。

- #include <Servo.h>
- #include <LiquidCrystal.h>

- LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
- Servo myservo1;
- Servo myservo2;

- #define LEFT  0
- #define UP  1
- #define DOWN  2
- #define RIGHT  3
- #define SELECT 4
- #define NONE  5

- #define analogswitch A0

- int pattern;
- int pushbutton;
- int a;
- int b;

```
int button() {
  pushbutton = (analogRead(analogswitch) / 4);
  if (pushbutton >= 240) return NONE;
  if (pushbutton < 20) return  RIGHT;
  if (pushbutton < 70) return  UP;
  if (pushbutton < 120) return DOWN;
  if (pushbutton < 170) return LEFT;
  if (pushbutton < 240) return SELECT;
}
```

```
void setup() {
  //myservo1.attach(12);
  //myservo2.attach(13);
  Serial.begin(250000);
  lcd.begin(16, 2);              // start the library
  lcd.setCursor(0, 0);
  lcd.write("button");
}
```

```
void loop() {
  pattern = button();
  a = myservo1.attached();
  b = myservo2.attached();
  if (pattern == NONE) {
    myservo1.detach();
    myservo2.detach();
  }
```

```
else {
   switch (pattern) {
     case LEFT:
       myservo1.attach(12);
       myservo2.attach(13);
       myservo1.write(34);
       myservo2.write(34);
       lcd.setCursor(0, 1);
       lcd.write("LEFT");
       lcd.write("    ");
       break;
```

- case UP:
- 	myservo1.attach(12);
- 	myservo2.attach(13);
- 	myservo1.write(60);
- 	myservo2.write(60);
- 	lcd.setCursor(0, 1);
- 	lcd.write("UP");
- 	lcd.write("  ");
- 	break;

```
case DOWN:
        myservo1.attach(12);
        myservo2.attach(13);
        myservo1.write(127);
        myservo2.write(127);
        lcd.setCursor(0, 1);
        lcd.write("DOWN");
        lcd.write("    ");
        break;
```

- case RIGHT:
- 	myservo1.attach(12);
- 	myservo2.attach(13);
- 	myservo1.write(150);
- 	myservo2.write(150);
- 	lcd.setCursor(0, 1);
- 	lcd.write("RIGHT");
- 	lcd.write("  ");
- 	break;

```
case SELECT:
        myservo1.attach(12);
        myservo2.attach(13);
        myservo1.write(180);
        myservo2.write(180);
        lcd.setCursor(0, 1);
        lcd.write("SELECT");
        lcd.write("     ");
        break;
```

```
  case NONE:
        lcd.setCursor(0, 1);
        lcd.write("NONE");
        lcd.write("     ");
        break;
    }
  }
  serial();
}
```

```
void serial() {
  Serial.print("pushbutton");
  Serial.print(" ");
  Serial.print(pushbutton);
  Serial.print(" ");
  Serial.print("a");
  Serial.print(" ");
  Serial.print(a);
  Serial.print(" ");
  Serial.print("b");
  Serial.print(" ");
  Serial.println(b);
}
```

- サーボの部分のみを書くとこうなる

```
#include <Servo.h>
Servo myservo1;
Servo myservo2;

void setup() {
  myservo1.attach(19);
  myservo2.attach(11);
}
```

```
• void loop() {
• switch (pattern) {
•     case LEFT:
•       myservo1.write(0);
• break;

•     case UP:
•       myservo1.write(45);
• break;

•     case DOWN:
•       myservo1.write(135);
• break;
```

```
 case RIGHT:
      myservo1.write(180);
break;

    case SELECT:
      myservo1.write(90);
break;

    case NONE:
   break;
  }
}
```

# SoftWareSerial.h

- Arduinoハードウェアでは、0番ピンと1番ピンでシリアル通信をサポートしている(USB接続でPCとの通信にも使われる)。このシリアル通信はチップに内蔵されたハードウェアを利用しており、UARTと呼ばれる。このハードウェアを使えば、64バイトのシリアルバッファに有余がある限りは、他のタスクを実行中でもAtmegaチップはシリアル通信を受信することができる。
- SoftwareSerialライブラリは、ソフトウェア実装により(なのでSoftwareSerialと名づけられた)、Arduinoの他のデジタルピンを使ってシリアル通信を利用できるようにするために開発された。115200bpsまでのスピードで、複数のソフトウェアシリアルポートを利用することができる。パラメータを設定することで、負論理(inverted signaling )のデバイスにも対応することができる。

このライブラリには以下の制約がある。

- 複数のソフトウェアシリアルポートを利用しても、同時には一つのデータしか受信できない。
- MegaとMega 2560では、全てのピンが入力の変化に対する割り込みをサポートしているわけではないので、以下のピンだけが受信(RX)に対応している：10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8(62), A9(63), A10(64), A11(65), A12(66), A13(67), A14(68), A15(69)。
- LeonardとMicroでは、全てのピンが入力の変化に対する割り込みをサポートしているわけではないので、以下のピンだけが受信(RX)に対応している：8, 9, 10, 11, 14(MISO), 15(SCK), 16(MOSI)。
- Arduino/Genuino 101では、現在の最大受信(RX)スピードは57600bpsである。
- Arduino/Genuino 101では、13番ピンでは受信できない。
- https://garretlab.web.fc2.com/arduino_reference/libraries/standard_libraries/SoftwareSerial/index.html

# 関数紹介（ソフトウェアシリアル）

- SoftwareSerial

それ以外は通常のSelialと同じで良い。

# SoftwareSerial

- SoftwareSerialはSoftwareSerialオブジェクトのインスタンスを作成するために利用される。オブジェクトの名前は下の例のように指定する。引数inverse_logicはオプションで、デフォルトはfalseである。詳細は下記を参照すること。複数のSoftwareSerialオブジェクトを作成することができるが、同時には一つのオブジェクトしか受信することができない。

- 通信を開始するには、SoftwareSerial.begin()を呼び出す必要がある。

- SoftwareSerial(rxPin, txPin, inverse_logic)

receivePin　シリアルデータを受信するピン番号

transmitPin　シリアルデータを送信するピン番号

inverse_logic　入力ビットを反転させる。デフォルトは通常のロジック。このパラメータを設定したときは、RXピンのLOW(通常は0V)を1(アイドル状態)、HIGH(通常は5V)を0として取り扱う。Txピンへの書き込みにも影響を与える。デフォルトはfalse。

# 課題

- LCD_Keypad_Shieldに文字を表示してみる。
- スイッチごとに表示する文字を変える。
- スイッチを押してサーボを動かしてみる。
- （押したスイッチをLCDで確認する。）

# 提出物

- LCD_Keypad_Shieldに文字を表示したときのプログラム
- スイッチごとに表示する文字を変えるプログラム
- それぞれの動作（動画）

# 提出方法

- いつも通り

# 参考

以下参考

# LCD_Keypad_Shieldスイッチ部分

```
int read_LCD_buttons(){
  adc_key_in = analogRead(0);
  if (adc_key_in > 1000) return btnNONE;
  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;
}
```

# 定義

- #define btnRIGHT  0
- #define btnUP    1
- #define btnDOWN   2
- #define btnLEFT   3
- #define btnSELECT 4
- #define btnNONE   5

# LCD

- lcd(8, 9, 4, 5, 6, 7);