```
//Servo
#include <Servo.h>
#include <LiquidCrystal.h>

// select the pins used on the LCD panel
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;
   //return NONE;
}



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);
}
```

```
//sample_LCD_Keypad
#include <LiquidCrystal.h>

/*********************************************************

   This program will test the LCD panel and the buttons
   Mark Bramwell, July 2010

*********************************************************/

// select the pins used on the LCD panel
LiquidCrystal lcd(8, 9, 4, 5, 6, 7);

// define some values used by the panel and buttons
int lcd_key       = 0;
int adc_key_in    = 0;
#define btnRIGHT   0
#define btnUP      1
#define btnDOWN    2
#define btnLEFT    3
#define btnSELECT 4
#define btnNONE    5

// read the buttons
int read_LCD_buttons()
{
   adc_key_in = analogRead(0);         // read the value from the sensor
   // my buttons when read are centered at these valies: 0, 144, 329, 504, 741
   // we add approx 50 to those values and check to see if we are close
   if (adc_key_in > 1000) return btnNONE; // We make this the 1st option for speed reasons
since it will be the most likely result
   // 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;

   // For V1.0 comment the other threshold and use the one below:
   /*
      if (adc_key_in < 50)    return btnRIGHT;
      if (adc_key_in < 195)   return btnUP;
      if (adc_key_in < 380)   return btnDOWN;
```

```
    if (adc_key_in < 555)   return btnLEFT;
    if (adc_key_in < 790)   return btnSELECT;
  */


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

void setup()
{
  lcd.begin(16, 2);                    // start the library
  lcd.setCursor(0, 0);
  lcd.print("Push the buttons"); // print a simple message
}

void loop()
{
  lcd.setCursor(9, 1);                 // move cursor to second line "1" and 9 spaces over
  lcd.print(millis() / 1000);       // display seconds elapsed since power-up


  lcd.setCursor(0, 1);                 // move to the begining of the second line
  lcd_key = read_LCD_buttons();  // read the buttons

  switch (lcd_key)                     // depending on which button was pushed, we perform an
action
  {
    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;
        }
    }

}
```

```
//Ethernetl
#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 };
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);
}
```

```
//analogread_kakunin
#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("      ");

}
```

```
//servo
#include <Servo.h>
#include <LiquidCrystal.h>

// select the pins used on the LCD panel
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 = 0;
int pushbutton = 0;

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 setup() {
   myservo1.attach(19);
   myservo2.attach(11);
   Serial.begin(9600);
   lcd.begin(16, 2);                    // start the library
   lcd.setCursor(0, 0);
   lcd.write("button");

}
```

```
void loop() {
  pattern = button();
  switch (pattern) {
    case LEFT:
      myservo1.write(0);
      lcd.setCursor(0, 1);
      lcd.write("LEFT");
      lcd.write("      ");
      break;

    case UP:
      myservo1.write(45);
      lcd.setCursor(0, 1);
      lcd.write("UP");
      lcd.write("      ");
      break;

    case DOWN:
      myservo1.write(135);
      lcd.setCursor(0, 1);
      lcd.write("DOWN");
      lcd.write("      ");
      break;

    case RIGHT:
      myservo1.write(180);
      lcd.setCursor(0, 1);
      lcd.write("RIGHT");
      lcd.write("      ");
      break;

    case SELECT:
      myservo1.write(90);
      lcd.setCursor(0, 1);
      lcd.write("SELECT");
      lcd.write("      ");
      break;

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

```
    }
}
```