[Ultrasonic + Husky Lens]

```
#include "HUSKYLENS.h"
#include "SoftwareSerial.h"

HUSKYLENS huskylens;
SoftwareSerial mySerial(3, 4);
void printResult(HUSKYLENSResult result);
int RightMotor_E_pin = 9;
int LeftMotor_E_pin = 10;
int trigPin = 6;
int echoPin = 5;
int E_carSpeed = 140;
void setup() {
    Serial.begin(115200);
    mySerial.begin(9600);
    TCCR1B = TCCR1B & B11111000 | B00000101;
    pinMode(echoPin, INPUT);    // echoPin 입력
    pinMode(trigPin, OUTPUT);  // trigPin 출력
    pinMode (9, OUTPUT);
    pinMode (10, OUTPUT);
    pinMode (8, OUTPUT);
    pinMode (11, OUTPUT);
    pinMode (12, OUTPUT);
    pinMode (13, OUTPUT);
    while (!huskylens.begin(mySerial))
    {
        Serial.println(F("fail"));
        delay(100);
    }
}
void loop()
{
  long duration, distance;
  digitalWrite(trigPin, HIGH);                          // trigPin에서 초음파 발생(echoPin
도 HIGH)
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);                    // echoPin 이 HIGH를 유지한 시
간을 저장 한다.
  distance = ((float)(340 * duration) / 1000) / 2;
```

```
  Serial.print("distance:");                              // 물체와 초음파 센서간 거리를 표시
  Serial.print(distance);
  Serial.println(" mm");
  delay(100);
    if (!huskylens.request())
Serial.println(F("Fail to request data from HUSKYLENS, recheck the connection!"));

    else if(!huskylens.isLearned()) {
Serial.println(F("Nothing learned, press learn button on HUSKYLENS to learn one!"));
stop();
    }
    else if(!huskylens.available()) {
Serial.println(F("No block or arrow appears on the screen!"));
stop();
}
    else
    {
        Serial.println(F("###########"));
        while (huskylens.available())
        {
            HUSKYLENSResult result = huskylens.read();
            printResult(result);
            driveBot(result,distance);
        }
    }
}
void printResult(HUSKYLENSResult result){
    if (result.command == COMMAND_RETURN_BLOCK){

Serial.println(String()+F("Block:xCenter=")+result.xCenter+F(",yCenter=")+result.yCenter+F(
",width=")+result.width+F(",height=")+result.height+F(",ID=")+result.ID);
    }
    else if (result.command == COMMAND_RETURN_ARROW){

Serial.println(String()+F("Arrow:xOrigin=")+result.xOrigin+F(",yOrigin=")+result.yOrigin+F(",x
Target=")+result.xTarget+F(",yTarget=")+result.yTarget+F(",ID=")+result.ID);
    }
    else{
        Serial.println("Object unknown!");
    }
}
```

```
void driveBot(HUSKYLENSResult result,float distance)
{
  if(result.width<=47)
  {
    if(distance <= 200)
    {
      stop();
    }
    else
      forward();
  }
  else if(result.xCenter<=65)
  {
       if(distance <= 200)
    {
      stop();
    }
    else
      left();
  }
  else if(result.xCenter>=235)
  {
       if(distance <= 200)
    {
      stop();
    }
    else
      right();
  }
  else if(result.width>47&&result.width<70)
  {
    stop();
  }
  else if (result.width>=70){
       if(distance <= 200)
    {
      stop();
    }
    else
       backward();
  }
```

```
}


void stop()
{
digitalWrite(8, LOW);
digitalWrite(11, LOW);
digitalWrite(12, LOW);
digitalWrite(13, LOW);
Serial.println("Stop");
}
void right()
{

analogWrite(LeftMotor_E_pin,140);
digitalWrite(12, HIGH);
digitalWrite(13, LOW);
Serial.println(" Rotate Left");
}
void left()
{
analogWrite(RightMotor_E_pin,140 );
digitalWrite(8, HIGH);
digitalWrite(11, LOW);
Serial.println(" Rotate Right");
}

void forward()
{
analogWrite(RightMotor_E_pin,160 );
analogWrite(LeftMotor_E_pin,160);
digitalWrite(8, HIGH);
digitalWrite(11, LOW);
digitalWrite(12, HIGH);
digitalWrite(13, LOW);

Serial.println("Forward");
}

void backward()
{
```

```
analogWrite(RightMotor_E_pin,160 );
analogWrite(LeftMotor_E_pin,160);
digitalWrite(8, LOW);

digitalWrite(11, HIGH);
digitalWrite(12, LOW);
digitalWrite(13, HIGH);
Serial.println("Backward");
}
```

[Wi-Fi Module _ Remote Adjustment]

```
#include "WiFiEsp.h"
#include "HUSKYLENS.h"
#include <SoftwareSerial.h>
#define rxPin 3
#define txPin 2
#define ledPin 13
#define digiPin 7

HUSKYLENS huskylens;
SoftwareSerial mySerial(2, 3);
SoftwareSerial esp01(txPin, rxPin); // SoftwareSerial NAME(TX, RX);
void printResult(HUSKYLENSResult result);
int RightMotor_E_pin = 9;
int LeftMotor_E_pin = 10;
int E_carSpeed = 140;
const char ssid[] = "bae";      // your network SSID (name)
const char pass[] = "gus30077";            // your network password
int status = WL_IDLE_STATUS;           // the Wifi radio's status
WiFiEspServer server(80);
uint8_t pin_val = 0; // 디지털 버튼 제어용 변수
String text = "";

void setup() {
       pinMode(RightMotor_E_pin, OUTPUT);
      pinMode(LeftMotor_E_pin, OUTPUT);
      pinMode (9, OUTPUT);
      pinMode (10, OUTPUT);
      pinMode (8, OUTPUT);
      pinMode (11, OUTPUT);
      pinMode (12, OUTPUT);
      pinMode (13, OUTPUT);

      Serial.begin(115200);
      esp01.begin(9600);    //와이파이 시리얼
      WiFi.init(&esp01);    // initialize ESP module
      while ( status != WL_CONNECTED) {    // 약 10초동안 wifi 연결 시도
      Serial.print(F("Attempting to connect to WPA SSID: "));
      Serial.println(ssid);
      status = WiFi.begin(ssid, pass);      // Connect to WPA/WPA2 network
```

```
    IPAddress ip = WiFi.localIP();
    Serial.print("IP Address: ");
    Serial.println(ip);
  }
  server.begin();
  TCCR1B = TCCR1B & B11111000 | B00000101;
}

const char HTTP_HEAD[] PROGMEM = "<!DOCTYPE html><html lang=\"ko\"><head>"
                                 "<meta charset=\"UTF-8\">"
                                 "<meta name=\"viewport\"content=\"width=device-width,initial-scale=1,user-scalable=no\"/>"
                                 "<link rel=\"icon\" href=\"data:,\">";
const char HTTP_STYLE[] PROGMEM = "<style>"
                                 "body{text-align:center;font-family:verdana;}"

"button{border:0;border-radius:0.3rem;background-color:#1fa3ec;color:#fff;line-height:2.4rem;font-size:1.2rem;width:100%} "
                                 "</style>";
const char HTTP_HEAD_END[] PROGMEM = "</head><body><div style=\"text-align:center;display:inline-block;min-width:260px;\">"
                                 "<h2>SMART CARhhhhT</h2>";
const char BUTTON_TYPE[] PROGMEM = "<p><button style=\"width:40%;background-color:#12cb3d;\">ON</button>"
                                 "<button style=\"margin-left:10%;width:40%;background-color:#1fa3ec;\">OFF</button></a></p>";
const char BUTTON_A_ON[] PROGMEM = "<p>Tracking mode</p><a href=\"/A/0\"><button style=\"background-color:#12cb3d;\">ON</button></a></p>";
const char BUTTON_A_OFF[] PROGMEM = "<p>Tracking mode</p><a href=\"/A/1\"><button style=\"background-color:#1fa3ec;\">OFF</button></a></p>";
const char BUTTON_B_ON[] PROGMEM = "<p>Navigation mode</p><a href=\"/B/0\"><button style=\"background-color:#12cb3d;\">ON</button></a></p>";
const char BUTTON_B_OFF[] PROGMEM = "<p>Navigation mode</p><a href=\"/B/1\"><button style=\"background-color:#1fa3ec;\">OFF</button></a></p>";
const char HTTP_END[] PROGMEM = "</div></body></html>\r\n";

bool button_a = LOW; // off
bool button_b = LOW; // off
```

```
void loop() {
  WiFiEspClient client = server.available();  // listen for incoming clients
  if (client) {                               // if you get a client,
    while (client.connected()) {              // loop while the client's connected
      if (client.available()) {               // if there's bytes to read from the client,
        String income_wifi = client.readStringUntil('\n');
        bool browser = false;
        if (income_wifi.indexOf("%%F0") != -1) {
            String wifi_temp = income_wifi.substring(income_wifi.indexOf("%%F0")+4,
income_wifi.indexOf("%%F1"));
            pin_val = wifi_temp.toInt();
            pin_control();
        } else if (income_wifi.indexOf(F("A/1")) != -1) {
          Serial.println(F("button_A on"));
          button_a = HIGH; browser = true;
          digitalWrite(ledPin, button_a);
        } else if (income_wifi.indexOf(F("A/0")) != -1) {
          Serial.println(F("button_A off"));
          button_a = LOW; browser = true;
          digitalWrite(ledPin, button_a);
        } else if (income_wifi.indexOf(F("B/1")) != -1) {
            Serial.println("Forward");
            while (!huskylens.begin(mySerial))
            {
                //Serial.println(F("fail"));
                delay(100);
            }
              while (huskylens.available())
          {
              HUSKYLENSResult result = huskylens.read();
              printResult(result);
              driveBot(result);
              if (income_wifi.indexOf(F("B/0")) != -1)break;
          }

          browser = true;

        } else if (income_wifi.indexOf(F("B/0")) != -1) {

          digitalWrite(8, LOW);
          digitalWrite(11, LOW);
```

```
            digitalWrite(12, LOW);
            digitalWrite(13, LOW);
            Serial.println("Stop");

            Serial.println(F("button_B off"));
            button_b = LOW; browser = true;
            digitalWrite(digiPin, button_b);


          } else {
            String wifi_temp = income_wifi.substring(income_wifi.indexOf("GET /")+5,
income_wifi.indexOf("HTTP/1.1"));
            if (wifi_temp != " ") {
              if (wifi_temp.indexOf("%20") != -1) {
                String space = "%20";
                String space_convert = " ";
                wifi_temp.replace(space, space_convert);
              }
              text = wifi_temp;
              Serial.println(text);
            } else {
              browser = true;
            }
          }
          client.flush();
          if (browser == true) {
            client.println(F("HTTP/1.1 200 OK")); // HTTP 프로토콜 헤더
            client.println(F("Content-type:text/html"));
            client.println(F("Connection: close"));
            client.println();
            String page;
            page  = (const __FlashStringHelper *)HTTP_HEAD;
            page += (const __FlashStringHelper *)HTTP_STYLE;
            page += (const __FlashStringHelper *)HTTP_HEAD_END;
            page += (const __FlashStringHelper *)BUTTON_TYPE;
            if (button_a == HIGH) {
              page += (const __FlashStringHelper *)BUTTON_A_ON;
            } else {
              page += (const __FlashStringHelper *)BUTTON_A_OFF;
            }
            if (button_b == HIGH) {
```

```
            page += (const __FlashStringHelper *)BUTTON_B_ON;
          } else {
            page += (const __FlashStringHelper *)BUTTON_B_OFF;
          }
          page += (const __FlashStringHelper *)HTTP_END;
          client.print(page);
          client.println();
          delay(1);
        }
        break;
      }
    }
    client.stop();
    Serial.println(F("Client disconnected"));
  }
}

void pin_control() {
  if (pin_val != 0) {
    switch (pin_val) {
      case 11: digitalWrite(ledPin, HIGH); // button 1 : on
               Serial.println("App Button_A ON");
                  break;
      case 10: digitalWrite(ledPin, LOW); // button 1 : off
               Serial.println("App Button_A OFF");
                  break;
      case 21: Serial.println("button 2 : on");
               Serial.println("App Button_B ON");
                  break;
      case 20: Serial.println("button 2 : off");
               Serial.println("App Button_B OFF");
                  break;
    }
  pin_val = 0;
  }
}
void printResult(HUSKYLENSResult result){
    if (result.command == COMMAND_RETURN_BLOCK){

Serial.println(String()+F("Block:xCenter=")+result.xCenter+F(",yCenter=")+result.yCenter+F(
",width=")+result.width+F(",height=")+result.height+F(",ID=")+result.ID);
```

```
    }
    else if (result.command == COMMAND_RETURN_ARROW){

Serial.println(String()+F("Arrow:xOrigin=")+result.xOrigin+F(",yOrigin=")+result.yOrigin+F(",x
Target=")+result.xTarget+F(",yTarget=")+result.yTarget+F(",ID=")+result.ID);
    }
    else{
        Serial.println("Object unknown!");
    }
}

void driveBot(HUSKYLENSResult result)
{
  if(result.width<=47)
  {
    forward();
  }
  else if(result.xCenter<=65)
  {
    left();
  }
  else if(result.xCenter>=235)
  {
    right();
  }
  else if(result.width>47&&result.width<70)
  {
    stop();
  }
  else if (result.width>=70){
    backward();
  }
}

void stop()
{
digitalWrite(8, LOW);
digitalWrite(11, LOW);
digitalWrite(12, LOW);
digitalWrite(13, LOW);
Serial.println("Stop");
```

```
}

void right()
{

analogWrite(LeftMotor_E_pin,170);
digitalWrite(12, HIGH);
digitalWrite(13, LOW);
digitalWrite(11, LOW);
digitalWrite(8, LOW);
Serial.println(" Rotate Left");
}
void left()
{
analogWrite(RightMotor_E_pin,170 );
digitalWrite(8, HIGH);
digitalWrite(11, LOW);
digitalWrite(12, LOW);
digitalWrite(13, LOW);
Serial.println(" Rotate Right");
}

void forward()
{
analogWrite(RightMotor_E_pin,190 );
analogWrite(LeftMotor_E_pin,190);
digitalWrite(8, HIGH);
digitalWrite(11, LOW);
digitalWrite(12, HIGH);
digitalWrite(13, LOW);
Serial.println("Forward");
}

void backward()
{
analogWrite(RightMotor_E_pin,190 );
analogWrite(LeftMotor_E_pin,190);
digitalWrite(8, LOW);
digitalWrite(11, HIGH);
digitalWrite(12, LOW);
digitalWrite(13, HIGH);
```

```
Serial.println("Backward");
}
```