#include "Adafruit\_VL53L0X.h"

#include <LCD\_I2C.h>

#include <ESP8266WiFi.h>

#include <ESP8266HTTPClient.h>

#include <WiFiClient.h>

#include <ArduinoJson.h>

const char\* ssid = "iot";

const char\* password = "12345678";

const char\* serverName = "<http://iotcloud22.in/crack_detection/post_value.php>";

const int IN1 = D3;

const int IN2 = D4;

const int IN3 = D5;

const int IN4 = D7;

LCD\_I2C lcd(0x27, 16, 2);

Adafruit\_VL53L0X lox = Adafruit\_VL53L0X();

int ldr;

WiFiClient client;

HTTPClient http;

String httpRequestData;

int dis;

void setup() {

Serial.begin(9600);

lcd.begin();

lcd.backlight();

lcd.setCursor(0, 0);

lcd.print(" Early crack ");

lcd.setCursor(0, 1);

lcd.print("detection system");

pinMode(D6, INPUT);

pinMode(D0, OUTPUT);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(IN3, OUTPUT);

pinMode(IN4, OUTPUT);

while (!Serial) {

delay(1);

}

Serial.println("Adafruit VL53L0X test");

if (!lox.begin()) {

Serial.println(F("Failed to boot VL53L0X"));

while (1)

;

}

WiFi.begin(ssid, password);

Serial.println(serverName);

Serial.println("Connecting");

while (WiFi.status() != WL\_CONNECTED) {

Serial.print(".");

delay(500);

}

Serial.println("");

Serial.print("Connected to WiFi network with IP Address: ");

Serial.println(WiFi.localIP());

// power

Serial.println(F("VL53L0X API Simple Ranging example\n\n"));

delay(2000);

lcd.clear();

}

void loop() {

VL53L0X\_RangingMeasurementData\_t measure;

Serial.print("Reading a measurement... ");

lox.rangingTest(&measure, false); // pass in 'true' to get debug data printout!

ldr = digitalRead(D6);

Serial.println(ldr);

if (measure.RangeStatus != 4) { // phase failures have incorrect data

Serial.print("Distance (mm): ");

Serial.println(measure.RangeMilliMeter);

dis=measure.RangeMilliMeter;

lcd.setCursor(0, 0);

lcd.print("Distance (mm):");

lcd.setCursor(14, 0);

lcd.print(measure.RangeMilliMeter);

}

if (ldr == 0) {

lcd.setCursor(0, 1);

lcd.print(" Crack detected ");

digitalWrite(D0,HIGH);

delay(500);

// lcd.clear();

}

else {

Serial.println(" out of range ");

lcd.setCursor(0, 1);

lcd.print("crack not detect");

delay(100);

// lcd.clear();

}

digitalWrite(D0,LOW);

sending\_to\_db();

json();

}

void sending\_to\_db() {

if (WiFi.status() == WL\_CONNECTED) {

http.begin(client, serverName);

http.addHeader("Content-Type", "application/x-www-form-urlencoded");

String httpRequestData = "&value1=" + String(ldr) + "&value2=" + String(dis) + "";

Serial.print("httpRequestData: ");

Serial.println(httpRequestData);

int httpResponseCode = http.POST(httpRequestData);

if (httpResponseCode > 0) {

Serial.print("HTTP Response code: ");

Serial.println(httpResponseCode);

} else {

Serial.print("Error code: ");

Serial.println(httpResponseCode);

}

http.end();

} else {

Serial.println("WiFi Disconnected");

}

//Send an HTTP POST request every 3 seconds

delay(100);

}

void json() {

if (WiFi.status() == WL\_CONNECTED) {

//HTTPClient http; //Object of class HTTPClient

http.begin(client, "<http://iotcloud22.in/crack_detection/light.json>");

int httpCode = http.GET();

//Check the returning code

if (httpCode > 0) {

// Parsing

}

}

StaticJsonDocument<256> doc;

DeserializationError error = deserializeJson(doc, http.getString());

Serial.println(http.getString());

if (error) {

Serial.print(F("deserializeJson() failed: "));

Serial.println(error.f\_str());

return;

}

//StaticJsonDocument<256> doc;

// deserializeJson(doc, json);

// auto error = deserializeJson(doc, json);

if (error) {

Serial.print(F("deserializeJson() failed with code "));

Serial.println(error.c\_str());

return;

}

String robot = doc["robot"]; // "on"

if (robot == "backward") {

digitalWrite(IN2, HIGH);

digitalWrite(IN4, HIGH);

Serial.println("Backward");

} else if (robot == "forward") {

digitalWrite(IN1, HIGH);

digitalWrite(IN3, HIGH);

Serial.println("Forward ");

}

else if (robot == "stop") {

digitalWrite(IN1, LOW);

digitalWrite(IN2, LOW);

digitalWrite(IN4, LOW);

digitalWrite(IN3, LOW);

Serial.println("Forward ");

}

http.end(); //Close connection

//Serial.println(flag);

delay(100);

}