Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної Інженерії

ЗВІТ

з дисципліни «Архітектура програмного забезпечення»

з лабораторної роботи №3

|  |  |
| --- | --- |
| Виконав | Перевірив: |
| ст. гр. ПЗПІ-20-7 | Старш. викл. кафедри ПІ |
| Крупчак Євгеній | Сокорчук І. П. |

Харків 2022

**ЛАБОРАТОРНА РОБОТА №3. РОЗРОБКА IOT АБО SMARTDEVICE ЧАСТИНИ**

**1.1 Мета роботи**

Розробити програмне забезпечення для  IoT або SmartDevice пристрою, реалізованого на базі будь-якої поширеної на  сьогодні платформи, придатної для реалізації вбудованих системдля проєкту за темою «Програмна система для автоматизації видачі боксів із їжею».

**1.2 Хід роботи**

IoT частина додатку написана за допомогою фреймворку Wiring та мови програмування Arduino С. У якості мікроконтролеру було використано ESP32. У якості замків використовуються сервомотори SG90. Використані бібліотеки:

1. ServoESP32 – для керування сервомоторами;
2. ESP\_WifiManager – для налаштування пристрою та підключення до WiFi;
3. MQTTPubSubClient – для підтримки протоколу MQTT;
4. ESP\_DoubleResetDetector – для відстеження входу до режиму налаштування.

На рисунку 1 зображено схему пристрою.

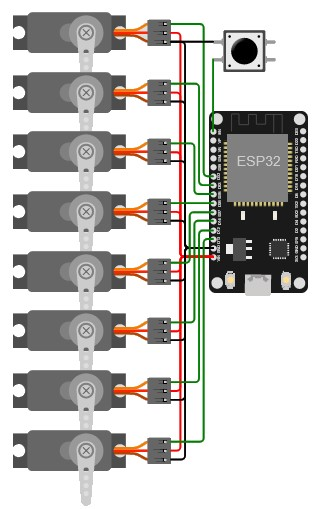


Рисунок 1 – Схема пристрою

На рисунку 2 зображено діаграму прецедентів для пристрою, на якій зображено відношення між акторами та прецедентами.

  
add admin

Рисунок 2 – Діаграма прецендентів пристрою

На рисунку 3 зображено діаграму взаємодії.

A screenshot of a computer

Description automatically generated with medium confidence

To appendix

Рисунок 3 – Діаграма взаємодії пристрою

На рисунку 4 зображено діаграму діяльности.

A screenshot of a computer screen

Description automatically generated with low confidence

To appendix

Рисунок 4 – Діаграма діяльности пристрою

На рисунку 5 зображено діаграму компонентів.

A picture containing text, screenshot, font

Description automatically generated

Рисунок 5 – Діаграма компонентів пристрою

Компонент «Main» являє собою головний файл, у якому зазначено 2 основні події: setup та loop. Вони відповідаюсь за стартове налаштування та цикл роботи пристрою відповідно. Код компоненту наведено у додатку А.

Компонент «Preferences MQTT client» являє собою бібліотеку, що відповідає за налаштування, підключення та зберігання налаштувань MQTT клієнта. Код компоненту наведено у додатку Б.

**1.3 Висновки**

У данній лабораторній роботі було розроблено програмне забезпечення для  IoT або SmartDevice пристрою для проєкту за темою «Програмна система для автоматизації видачі боксів із їжею».

Посилання на папку "DEMO" із відео та кодом на Google Drive: https://drive.google.com/drive/folders/1sBPljA5AFl66BZejiDRfry6l0LPELsV7?usp=share\_link

**ДОДАТОК А**

Програмний код файлу main.cpp

#include "main.h"

void configModeCallback(ESP\_WiFiManager \*wifiManager) {

Serial.println("Entered Configuration Mode");

Serial.print("Config SSID: ");

Serial.println(wifiManager->getConfigPortalSSID());

Serial.print("Config IP Address: ");

Serial.println(WiFi.softAPIP());

}

void startAP(bool (ESP\_WiFiManager::\*func)(const char \*, const char \*)) {

if (!(wm.\*func)("NexGenMeal Terminal", "12345678")) {

Serial.println("failed to connect and hit timeout");

delay(3000);

ESP.restart();

delay(5000);

}

}

void MQTTCallback(const String &payload, const size\_t size) {

Serial.print("Message arrived: ");

int i = payload.toInt();

Serial.println(i);

if (i < N\_CELLS) {

servos[i].write(45);

delay(delayMS);

servos[i].write(0);

}

}

void saveConfigCallback() {

shouldSaveConfig = true;

}

void setup() {

WiFi.mode(WIFI\_STA);

Serial.begin(115200);

for (int i = 0; i < N\_CELLS; i++) {

servos[i].attach(servoPins[i]);

servos[i].write(0);

}

pinMode(LED\_PIN, OUTPUT);

wm.setAPCallback(configModeCallback);

wm.setSaveConfigCallback(saveConfigCallback);

wm.setConfigPortalTimeout(TIMEOUT);

bool isEmpty = client.loadConfig();

ESP\_WMParameter broker\_text\_box("broker", "Enter MQTT Broker", client.getBroker(), 50);

char convertedValue[8];

sprintf(convertedValue, "%d", client.getPort());

ESP\_WMParameter port\_text\_box("port", "Enter MQTT port", convertedValue, 9);

ESP\_WMParameter topic\_text\_box("prefix", "Enter MQTT prefix", client.getPrefix(), 50);

wm.addParameter(&broker\_text\_box);

wm.addParameter(&port\_text\_box);

wm.addParameter(&topic\_text\_box);

if (!isEmpty || drd.detectDoubleReset())

startAP(&ESP\_WiFiManager::startConfigPortal);

else

startAP(&ESP\_WiFiManager::autoConnect);

Serial.println("");

Serial.println("WiFi connected");

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

client.setBroker(broker\_text\_box.getValue());

client.setPort(atoi(port\_text\_box.getValue()));

client.setPrefix(topic\_text\_box.getValue());

if (shouldSaveConfig)

client.saveConfig();

client.begin(MQTTCallback);

}

void loop() {

analogWrite(LED\_PIN, (WiFi.status() == WL\_CONNECTED) ? 255 : 0);

client.update();

drd.loop();

}

**ДОДАТОК Б**

Програмний код файлу Preferenced\_MQTT\_Client.cpp

#include <Preferenced\_MQTT\_Client.h>

Preferenced\_MQTT\_Client::Preferenced\_MQTT\_Client() {

preferences.begin("MQTT", false);

}

void Preferenced\_MQTT\_Client::saveConfig() {

preferences.putString("broker", broker);

preferences.putInt("port", port);

preferences.putString("topic", prefix);

}

bool Preferenced\_MQTT\_Client::loadConfig() {

port = preferences.getInt("port", 1883);

String brokerValue = preferences.getString("broker", "");

String prefixValue = preferences.getString("prefix", "");

strcpy(broker, brokerValue.c\_str());

strcpy(prefix, prefixValue.c\_str());

return strlen(broker) + strlen(prefix);

}

void Preferenced\_MQTT\_Client::begin(const std::function<void(const char \*, size\_t)> &cb) {

String topic = String(prefix) + '/' + String(SERIAL\_NUMBER);

Serial.print("Connecting ");

Serial.print(broker);

Serial.print(":");

Serial.print(port);

Serial.print(" on topic ");

Serial.println(topic);

client.connect(broker, port);

mqtt.begin(client);

mqtt.connect(topic);

mqtt.subscribe(topic, cb);

}