Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
Кафедра Програмної Інженерії

ЗВІТ

з дисципліни «Архітектура програмного забезпечення»

з лабораторної роботи №3

Виконав
ст. гр. ПЗПІ-20-7
Крупчак Євгеній

Перевірив:
Старш. викл. кафедри ПІ
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 C. 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.

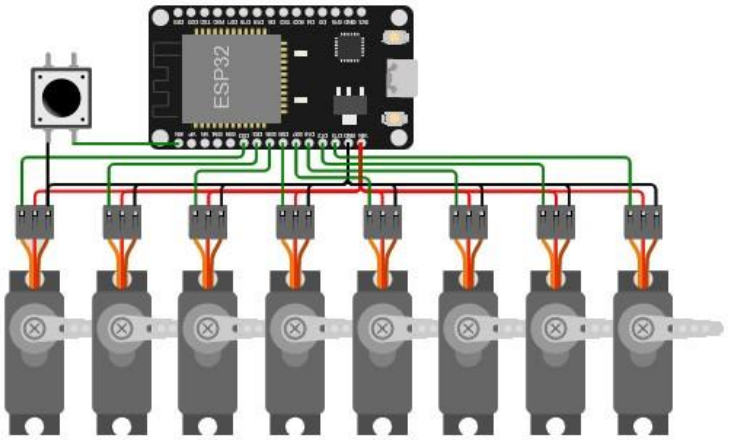

Рисунок 1 – Схема пристрою

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

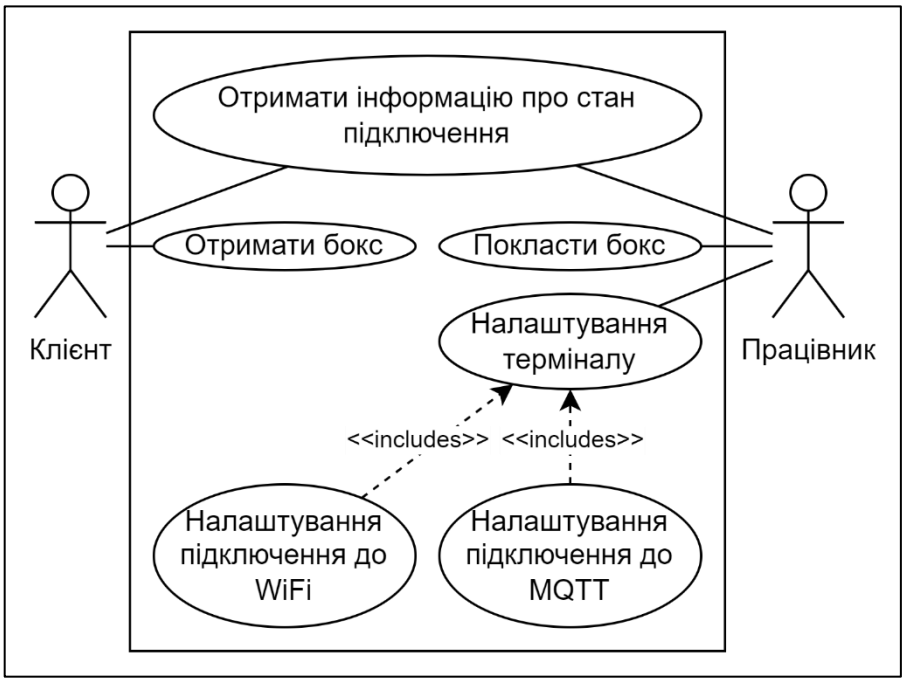

Рисунок 2 – Діаграма прецендентів пристрою

На рисунку 3 зображено діаграму взаємодії.

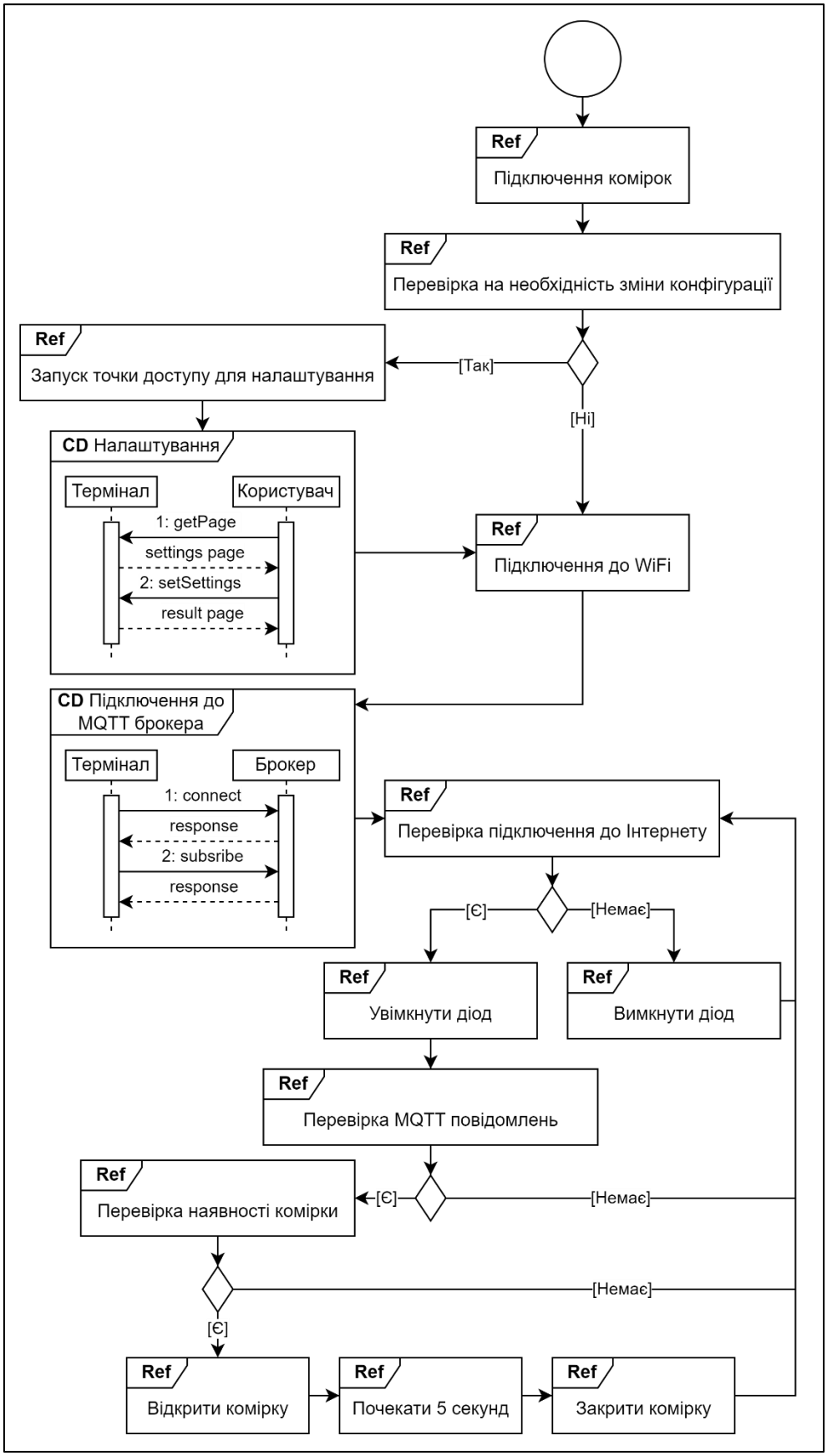

Рисунок 3 – Діаграма взаємодії пристрою

На рисунку 4 зображено діаграму діяльності.

Рисунок 4 – Діаграма діяльности пристрою

На рисунку 5 зображено діаграму компонентів.

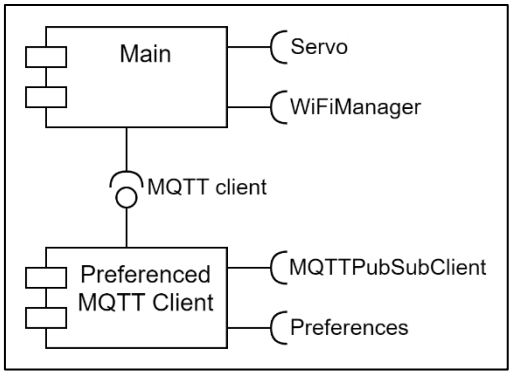

Рисунок 5 – Діаграма компонентів пристрою

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);

```

```
47    bool isEmpty = client.loadConfig();
48
49    ESP_WMParameter broker_text_box("broker", "Enter MQTT Broker",
50  client.getBroker(), 50);
51    char convertedValue[8];
52    sprintf(convertedValue, "%d", client.getPort());
53    ESP_WMParameter port_text_box("port", "Enter MQTT port",
54  convertedValue, 9);
55    ESP_WMParameter topic_text_box("prefix", "Enter MQTT prefix",
56  client.getPrefix(), 50);
57
58    wm.addParameter(&broker_text_box);
59    wm.addParameter(&port_text_box);
60    wm.addParameter(&topic_text_box);
61    if (!isEmpty || drd.detectDoubleReset())
62      startAP(&ESP_WiFiManager::startConfigPortal);
63    else
64      startAP(&ESP_WiFiManager::autoConnect);
65
66    Serial.println("");
67    Serial.println("WiFi connected");
68    Serial.print("IP address: ");
69    Serial.println(WiFi.localIP());
70
71    client.setBroker(broker_text_box.getValue());
72    client.setPort(atoi(port_text_box.getValue()));
73    client.setPrefix(topic_text_box.getValue());
74
75    if (shouldSaveConfig)
76      client.saveConfig();
77    client.begin(MQTTCallback);
78  }
79
80  void loop() {
81    analogWrite(LED_PIN, (WiFi.status() == WL_CONNECTED) ? 255 : 0);
82    client.update();
83    drd.loop();
84  }
```

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);
}

```