

Тема: Интерфейс интернета вещей

Задание: 10.11.23 и 17.11.23.

1. Создать графические интерфейсы для смартфона и для ПК. Продумать информацию которая будет передаваться (графическая, численная и др.). Информация не должна повторяться. Интерфейсы могут быть реализованы на базе любого ПО Java.
2. Продумать вид интерфейса и обеспечить его масштабирование под размер рабочего окна смартфона и ПК.
3. Возможна реализация аппаратного проекта на базе системы ESP и модуля NodeMcu (предоставляется по запросу в количестве 1-2 шт на группу) или приобретается самостоятельно. При использовании этих систем - можно использовать например вывод температуры из одной части комнаты в графический интерфейс реализованный на ПК.
4. Сгенерировать файлы проектов в виде пусковых файлов. Загрузить их в ответ на это задание. Включить в отчёт (ворд или пдф) задание, скриншоты программ, листинги кодов. Отчёт загружается в ответ на это задание.

Ссылка на гит с проектом:

https://github.com/nameunique/DaGOI_lab2

Скриншот программы:

ПК:



Смартфон:

ESP32 Web Server

LED on pin13 state:

ON

OFF

Potentiometer Value: 4095

Rotation Control: rotate stop

Clockwise

Counter-Clockwise

Stop Rotation

Speed Control:



Speed: 100

192.168.1.4



Листинги:

Скетч ардуино:

```
#include "WiFi.h"
```

```
#include "ESPAsyncWebServer.h"
```

```
#include "SPIFFS.h"
```

```
const char* ssid = "Artego";
```

```
const char* password = "1111111111";
```

```
const int ledPin = 13;
```

```
const int potens = 36;
```

```
const int PIN_STEP = 23;
```

```
const int PIN_DIR = 22;
```

```
const int PIN_EN = 21;
```

```
AsyncWebServer server(80);
```

```
String potentiometerValue = "0";
```

```
bool isMotorOn = false;
```

```
bool isClockwise = true;
```

```
int currentSpeed = 100;
```

```
String processor(const String& var) {
```

```
    if (var == "STATE") {
```

```
        if (digitalRead(ledPin)) {
```

```
            return "ON";
```

```
        } else {
```

```
            return "OFF";
```

```
    }  
  } else if (var == "POTENTIOMETER") {  
    return potentiometerValue;  
  }  
  return String();  
}
```

```
void setup() {  
  Serial.begin(115200);  
  pinMode(ledPin, OUTPUT);  
  pinMode(potens, INPUT);  
  pinMode(PIN_STEP, OUTPUT);  
  pinMode(PIN_DIR, OUTPUT);  
  pinMode(PIN_EN, OUTPUT);  
  
  if (!SPIFFS.begin(true)) {  
    Serial.println("An Error has occurred while mounting SPIFFS");  
    return;  
  }  
}
```

```
WiFi.begin(ssid, password);  
while (WiFi.status() != WL_CONNECTED) {  
  delay(1000);  
  Serial.println("Connecting to WiFi..");  
}
```

```
Serial.println(WiFi.localIP());
```

```
digitalWrite(PIN_EN, HIGH);
```

```
server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {  
    request->send(SPIFFS, "/index.html", String(), false, processor);  
});
```

```
server.on("/style.css", HTTP_GET, [](AsyncWebServerRequest *request) {  
    request->send(SPIFFS, "/style.css", "text/css");  
});
```

```
server.on("/led", HTTP_GET, [](AsyncWebServerRequest *request) {  
    if (request->hasParam("is_on")) {  
        String temp = request->getParam("is_on")->value();  
        if(temp == "true")  
        {  
            digitalWrite(ledPin, LOW);  
            request->send(200, "text/plain", "On");  
        }  
        else if(temp == "false")  
        {  
            digitalWrite(ledPin, HIGH);  
            request->send(200, "text/plain", "Off");  
        }  
    }  
});
```

```
server.on("/getPotentiometer", HTTP_GET, [](AsyncWebServerRequest  
*request) {  
    request->send(200, "text/plain", potentiometerValue);  
});
```

```

server.on("/rotate", HTTP_GET, [](AsyncWebServerRequest *request) {
    if (request->hasParam("clockwise")) {
        String temp = request->getParam("clockwise")->value();
        isClockwise = temp == "true";
    }
    isMotorOn = true;
    digitalWrite(PIN_EN, LOW); // Включаем мотор
    String str_to_send = "motor started " + String(isClockwise ? "clockwise" :
"counter clockwise");
    request->send(200, "text/plain", str_to_send);
});

server.on("/stopRotation", HTTP_GET, [](AsyncWebServerRequest *request) {
    isMotorOn = false;
    digitalWrite(PIN_EN, HIGH); // Выключаем мотор
    request->send(200, "text/plain", "motor stopped");
});

server.on("/speed_motor", HTTP_GET, [](AsyncWebServerRequest *request) {
    if (request->hasParam("speed")) {
        int speedd = request->getParam("speed")->value().toInt();
        currentSpeed = speedd;
    }
    request->send(200, "text/plain", "");
});

server.begin();
}

```

```
void loop() {  
  int analogValue = analogRead(potens);  
  potentiometerValue = String(analogValue);  
  if (isMotorOn) {  
    digitalWrite(PIN_DIR, isClockwise ? HIGH : LOW);  
    digitalWrite(PIN_STEP, HIGH);  
    delay(currentSpeed);  
    digitalWrite(PIN_STEP, LOW);  
    delay(currentSpeed);  
  }  
}
```

Скрипт index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" type="text/css" href="style.css">
  <script>
    var potentiometerValue = 0; // Изначальное значение потенциометра
    var currentSpeed = 10; // Изначальная скорость вращения

    function updatePotentiometerValue(value) {
      document.getElementById("potentiometer-value").textContent = "Potentiometer
Value: " + value;
    }

    function updateSpeedSliderValue(value) {
      document.getElementById("speed-value").textContent = "Speed: " + value;
    }

    function updatePotentiometer() {
      fetch('/getPotentiometer')
        .then(response => response.text())
        .then(data => {
          potentiometerValue = data;
          updatePotentiometerValue(potentiometerValue);
        });
    }

    function updateSpeed(speed) {
      currentSpeed = speed;
      updateSpeedSliderValue(speed);

      fetch('/speed_motor?speed=' + speed)
        .then(response => response.text())
        .then(data => {

        });
    }

    function startRotation(clockwise) {
      fetch('/rotate?clockwise=' + clockwise)
        .then(response => response.text())
        .then(data => {
          document.getElementById("rotation-state-value").textContent = "Rotation
Control: " + data;
        });
    }

    function ledOnOff(is_on) {
      fetch('/led?is_on=' + is_on)
```



```

        .then(response => response.text())
        .then(data => {
            document.getElementById("led13-state").textContent = "LED on pin13
state: " + data;
        });
    }

    function stopRotation() {
        fetch('/stopRotation')
        .then(response => response.text())
        .then(data => {
            document.getElementById("rotation-state-value").textContent = "Rotation
Control: " + data;
        });
    }

    setInterval(updatePotentiometer, 1000);
</script>
</head>
<body>
    <h1>ESP32 Web Server</h1>
    <p id="led13-state">LED on pin13 state:</p>
    <button class="button" onclick="ledOnOff(true)">ON</button></a>
    <button class="button button2" onclick="ledOnOff(false)">OFF</button></a>
    <p id="potentiometer-value">Potentiometer Value: 0</p>
    <p id="rotation-state-value">Rotation Control: rotate stop</p>
    <button class="button" onclick="startRotation(true)">Clockwise</button>
    <button class="button button2" onclick="startRotation(false)">Counter-
Clockwise</button>
    <p><button class="button" onclick="stopRotation()">Stop Rotation</button></p>
    <p>Speed Control:</p>
    <input type="range" id="speed-slider" min="1" max="100" step="1" value="100"
oninput="updateSpeed(this.value)">
    <p id="speed-value">Speed: 100</p>
</body>
</html>

```

Скрипт style.css:

```

html {
    font-family: Helvetica;
    display: inline-block;
    margin: 0px auto;
    text-align: center;
}

h1 {
    color: #0F3376;
    padding: 2vh;
}

```

```
}

p {
  font-size: 1.5rem;
}

.button {
  display: inline-block;
  background-color: #008CBA;
  border: none;
  border-radius: 4px;
  color: white;
  padding: 16px 40px;
  text-decoration: none;
  font-size: 30px;
  margin: 2px;
  cursor: pointer;
}

@media only screen and (max-width: 600px) {
  .button {
    font-size: 16px;
    padding: 8px 20px;
  }
}

.button2 {
  background-color: #f44336;
}
```