VILNIAUS VERSLO KOLEGIJA

INFORMACINIŲ TECHNOLOGIJŲ KATEDRA
PROGRAMAVIMAS IR INTERNETINĖS TECHNOLOGIJOS

PIT-21-I-NT

MIKROKOMPIUTERIŲ IR VALDIKLIŲ PROGRAMAVIMAS

PROJEKTIS DARBAS

APSAUGOS SISTEMA

Darbą atliko: **Audrius Ivko**
Darbo vadovas: **doc. dr. Aleksandr Igumenov**

Vilnius, 2023

TURINYS

ĮVADAS

Darbo projekto pavadinimas – Apsaugos sistema skirta namų ar panašaus objekto apsaugai, valdoma per WEB sąsaja, apsaugota dviejų faktorių autentifikacija (2FA) ir rodmenų duomenų baze.

Darbo užduotis:

Pagrindinė užduotis: Surinkti pateiktą schemą ir pateikti apsaugos sistemos prototipą valdomą per WEB vartotojo sąsaja su integruota dvigubos autentifikacijos sistema, pateikti sketch.ino failą su veikiančiu sprendimu. Vartotojo sąsajoje turi būti galimybė valdyti objekto apsaugos sistema:

1. Įjungti / išjungti apsaugos sistema;

2. Fiksuot apsaugos sistemos objekto statusą ir pokyčius sistemoje realiuoju laiku;
3. Valdyti atskirus objekto mazgus (atrakint / užrakint vartus, duris);
4. Įjungti / išjungti papildomu modulius neatjungiant pagrindinių sistemų;
5. Duomenų ir pokyčių valdymo fiksavimas įrašant ir nuskaitant duomenis į duomenų bazės serverius naudojant MySQL duomenų baze;
6. Vartotojo sąsaja (realizuota su responsive web design);
7. Įrenginio sąsaja;
8. Apsaugos sistemos įrenginys naudoja Wifi duomenų perdavimo ryši prisijungimui prie interneto;
9. Duomenų perdavimas į duomenų baze iš įrenginio pusės vyksta per html POST formos užklausą įrašant duomenys tiesiai į duomenų bazes;
10. Įrenginio duomenų nuskaitymui iš duomenų bazių duomenys pateikiami JSON formatu.
11. Vartotojas per vartotojo sąsaja įgalina programa įrašyti duomenis į duomenų baze.

Schemą sudaro šie elementai:

1. ESP8266 NodeMCU įrenginys;
2. SparkFun CD74HC4067 analoginis / skaitmeninis multiplekserio modulis - 16 kanalų
3. WEB serveris vartotojo sąsajai;
4. Duomenų bazės serveris dirbantis su MySQL duomenų baze;
5. 4 LED diodai;
6. 4 mygtukai daviklių imitacijai;
7. 4 varžos 220Ω LED diodų pajungimui;
8. 4 varžos 1kΩ mygtuku pajungimui;
9. Laidai schemos sujungimui.

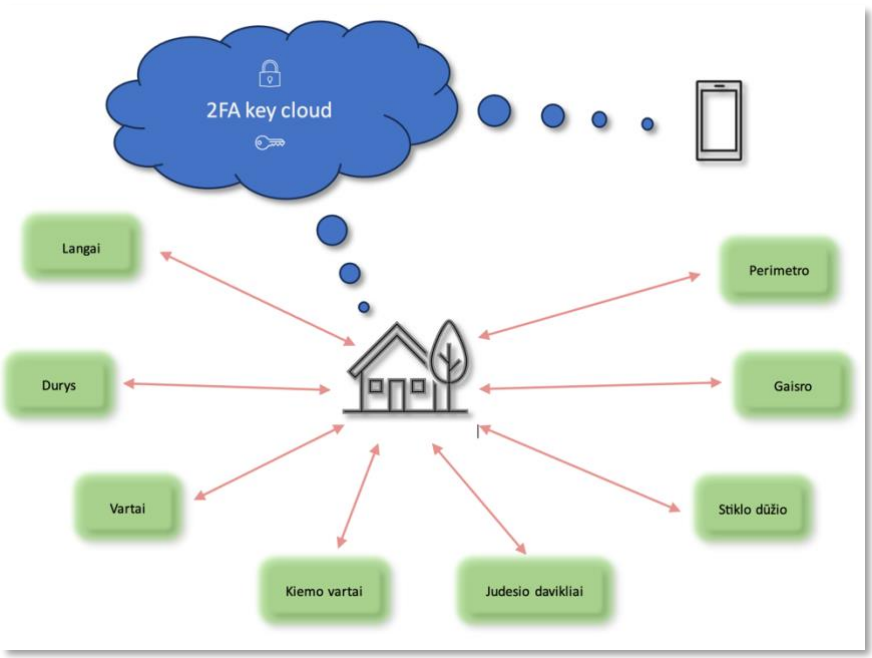
Naudojama įranga: Personalinis kompiuteris Macbook.

Darbo eiga:

1. Darbo eigos rezultatų fiksavimas pateikiant vaizdo įrašą su veikiančios schemos realizavimu;
2. „Arduino“ *security.ino* failo pateikimas.

VEIKIMO PRINCIPAS

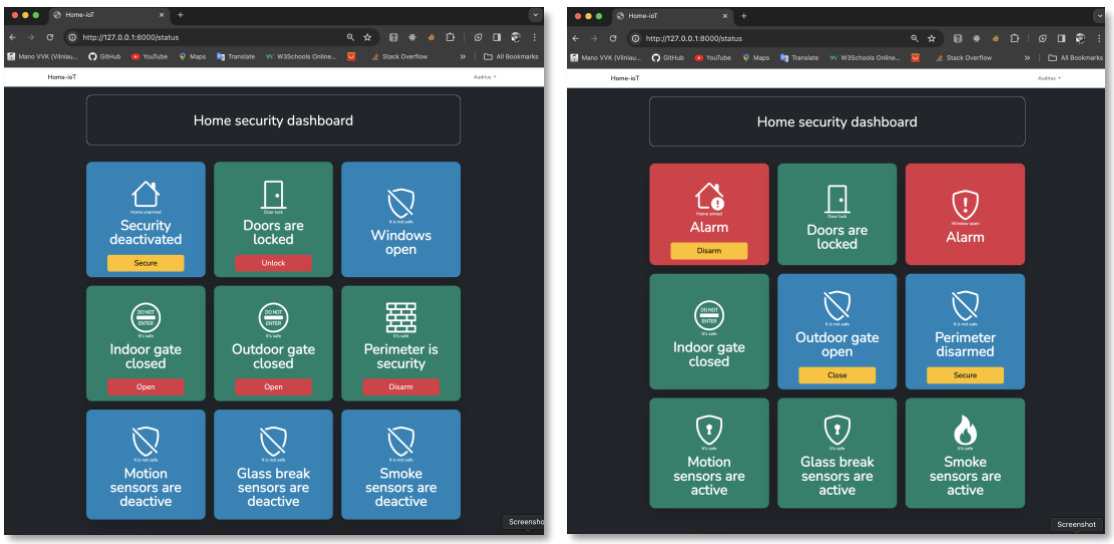
Daviklių, jutiklių ir valdiklių sistema nuskaito duomenų rodmenis. Surinktus duomenis siunčia interneto pagalbą į MySQL duomenų bazės serverius. Duomenų bazes serveris atlieka duomenų įpašymą į domenų baze. Prieiga prie duomenų ir apsaugos sistemos vartotojas gali valdyti tik atlikęs dvigubos autentifikacijos procedūrą. Sistema tikrina langų, durų, vartų, kiemo vartų, patalpų judesio daviklių, stiklo dūžių, gaisro bei perimetro jutiklių rodmenys. Sistema valdo durų, vartų ir perimetro sistema aktyvioje ir neaktyvioje sistemos būsenoje t.t. gali būti valdomi šie mazgai nepriklausomai nuo pagrindinės sistemos statuso (**pav. 1**).



***pav.** 1 Schema.*

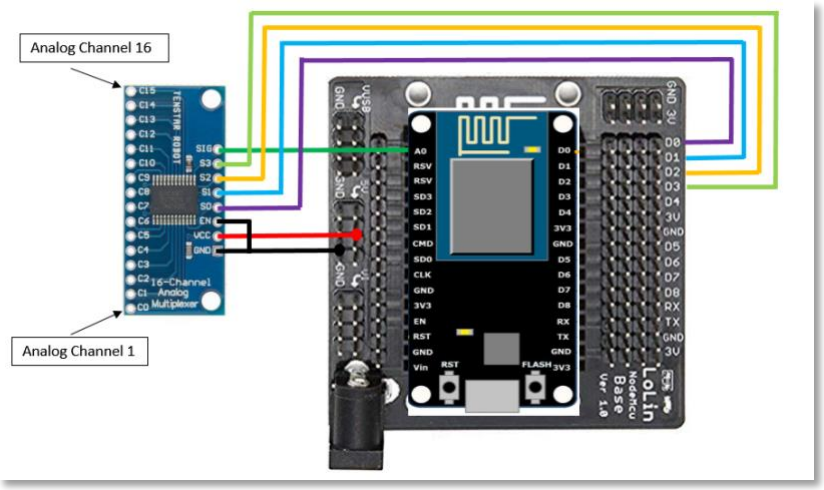
REALIZAVIMAS

Vartotojo sąsajos kodas realizuotas pasitelkiant LARAVEL v10.25.2 plugin v0.8.1 versija. Vartotojo sąsajos vaizdas (pav. 2). Vartotojo sąsajoje sistemos įjungimas / išjungimas. Pokyčiu atvaizdavimas skirtingų spalvų jungikliais, bei informaciniais užrašais.



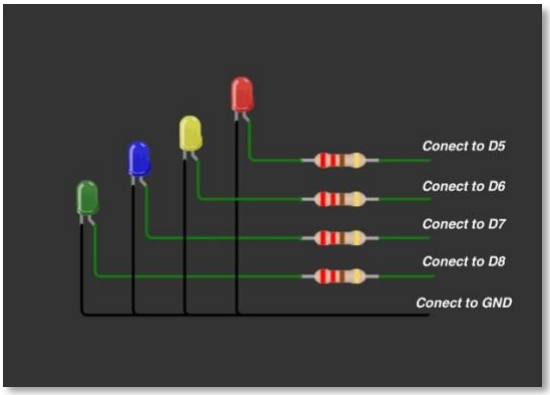
***pav.** 2 Vartotojo sąsajos vaizdas.*

Padidiname NodeMCU analoginių jungčių kieki iki 16 prijungiant skaitmeninį multiplekserio modulį per A0 skaitmeninę jungtį ir jungtis D0, D1, D2, D3 pagal pateiktą schemą (pav. 3).



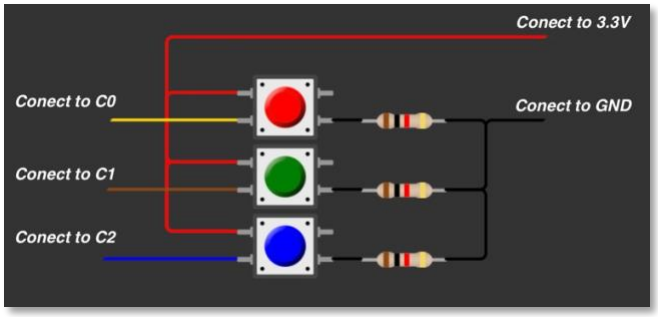
pav. *3 SparkFun CD74HC4067 analoginis / skaitmeninio multiplekserio modulio - 16 kanalų pajungimas prie NodeMCU*

Papildomu NodeMCU jungčių D5, D6, D7, D8 pajungimas prie LED diodų per 220Ω varžą schema. (pav. 4).



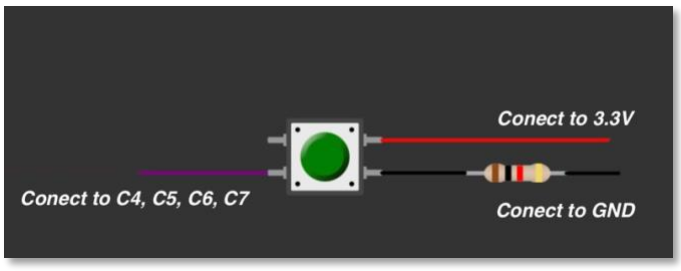
pav. *4 LED pajungimas D5, D6, D7, D8 jungtys*

Skaitmeninio multiplekserio modulio kontaktų C0, C1, C3 pajungimas prie kontrolinių jungiklių imituojančių suveikimo jungiklius (pav. 5).



pav. 5 *Skaitmeninio multiplekserio modulio kontaktų pajungimas*

Skaitmeninio multiplekserio modulio kontaktų C4, C5, C6, C7 pajungimas prie kontrolinių jungiklių imituojančių suveikimo jungiklius, pajungimas per vieną jungiklį (pav. 6).



pav. 6 *Skaitmeninio multiplekserio modulio kontaktų C4, C5, C6, C7 pajungimas*

Užduoties realizavimo kodas, (**pav.** 8).

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <ArduinoJson.h>
//-----------------------------------------
#define ON_Board_LED 2  //--> Defining an On Board LED (GPIO2 = D4), used for indicators when the process of connecting to a wifi router, read and write db

//------------------SSID and Password of your WiFi router.
const char* ssid = "*********";      //--> wifi name or SSID.
const char* password = "*********";  //--> wifi password.

//---------------------------------------Web Server address / IPv4
String host_or_IPv4 = "http://192.168.0.178/";
String Destination = "";
String URL_Server = "";
String URL_Server_status = "";
//-----------------------------------------
String getData = "";
String getData_status = "";
String payloadGet = "";
//-----------------------------------------
HTTPClient http;  //--> Declare object of class HTTPClient
WiFiClient client;
//-----------------------------------------
DynamicJsonDocument doc(1024);
int start = 0;
String sensor_name[] = { "Doors", "Windows", "Indoor gate", "Outdoor gate", "Motion", "Glass break", "Perimeter", "Smoke" };
int value_sensor[8];
int sensor[8];      /* Assign the sensor0-15 as analog output value from Channel C0-C15 */
int sensor_now[8];   /* Assign the sensor0-15 value now as analog output value from Channel C0-C15 */
int sensor_cloud[8]; /* Assign the sensor0-15 value from SQL db now as analog output value from Channel C0-C15 */
int disarm[] = { 1, 0, 1, 1, 0, 0, 0, 1 }; /* defoult sensor value */
int z;                              /* delay time ms*/
int zc;                              /* delay time ms for loop*/
int statusCloud = 0;
int doorLock = 2;      /* D4 pin */
int perimeterLock = 14; /* D5 pin */
int inGateLock = 12;    /* D6 pin */
int outGateLock = 13;   /* D7 pin */
int sensor_name_qty;    /* Sensor name qty */
```

```
#define S0 D0  /* Assign Multiplexer pin S0 connect to pin D0 of NodeMCU */
#define S1 D1  /* Assign Multiplexer pin S1 connect to pin D1 of NodeMCU */
#define S2 D2  /* Assign Multiplexer pin S2 connect to pin D2 of NodeMCU */
#define S3 D3  /* Assign Multiplexer pin S3 connect to pin D3 of NodeMCU */
#define SIG A0 /* Assign SIG pin as Analog output for all 16 channels of Multiplexer to pin A0 of NodeMCU */

void setup() {
  sensor_name_qty = sizeof(sensor_name) / sizeof(sensor_name[0]);

  pinMode(S0, OUTPUT); /* Define digital signal pin as output to the Multiplexer pin SO */
  pinMode(S1, OUTPUT); /* Define digital signal pin as output to the Multiplexer pin S1 */
  pinMode(S2, OUTPUT); /* Define digital signal pin as output to the Multiplexer pin S2 */
  pinMode(S3, OUTPUT); /* Define digital signal pin as output to the Multiplexer pin S3 */
  pinMode(SIG, INPUT); /* Define analog signal pin as input or receiver from the Multiplexer pin SIG */

  pinMode(D5, OUTPUT); /* Output to the pin D5 */
  pinMode(D6, OUTPUT); /* Output to the pin D6 */
  pinMode(D7, OUTPUT); /* Output to the pin D7 */
  pinMode(D8, OUTPUT); /* Output to the pin D8 */

  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(115200);
  delay(z * 2);
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");
  pinMode(ON_Board_LED, OUTPUT);     //--> On Board LED port Direction output
  digitalWrite(ON_Board_LED, HIGH);  //--> Turn off Led On Board
  //----------------------------------------Wait for connection
  Serial.print("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
  //----------------------------------------Make the On Board Flashing LED on the process of connecting to the wifi router.
    digitalWrite(ON_Board_LED, LOW);
    delay(z);
    digitalWrite(ON_Board_LED, HIGH);
    delay(z);
  }
  //----------------------------------------
  digitalWrite(ON_Board_LED, HIGH);  //--> Turn off the On Board LED when it is connected to the wifi router.
  //If successfully connected to the wifi router, the IP Address that will be visited is displayed in the serial monitor
  Serial.println("");
  Serial.print("Successfully connected to : ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  //----------------------------------------
  delay(z * 5);
}

void loop() {
  if (statusCloud == 0) {
    z = 250;
    zc = 50;
  } else {
    z = 1;
    zc = 1;
```

```
}
if (statusCloud == 0) {
  external();
}
Serial.println("--------------------------------------------------------------------------------");
Serial.println("Reading data from Server");
getdbData();

if (statusCloud != 0) {
  if (start == 0) {
    for (int i = 0; i < sensor_name_qty; i++) {
      value_sensor[i] = 1;
      sensor_now[i] = 1;
      sensor[i] = 0;
    }
    for (int i = 0; i < 30; i++) {
      Serial.print(".");
      delay(z);
    }
    Serial.println("Object security");
    statusCloud = 0;
    sensorRead();
    getdbData();
    start = 1;
  }

  digitalWrite(D5, HIGH); /* Define analog signal pin as output to the pin D5 - door lock */
  delay(200);
  digitalWrite(D6, HIGH); /* Define analog signal pin as output to the pin D6 - indoor gate close */
  delay(200);
  digitalWrite(D7, HIGH); /* Define analog signal pin as output to the pin D7 - outdoor gate close */
  delay(200);
  digitalWrite(D8, HIGH); /* Define analog signal pin as output to the pin D8 - perimeter armed */
  delay(200);

  Serial.println("Sending data to Server:");
  sensorRead();
  Serial.println();
} else {
  if (start == 1 && statusCloud == 0) {
    for (int i = 0; i < sensor_name_qty; i++) {
      value_sensor[i] = disarm[i];
      sensor_now[i] = disarm[i];
      Serial.print(" value_sensor[i] = ");
      Serial.print(value_sensor[i]);
      Serial.print("; disarm[i] = ");
      Serial.println(value_sensor[i]);
      putDataByGet(i);
    }
    start = 0;
    getdbData();
  }
  Serial.println("System disarmed");
}
for (int i = 0; i < sensor_name_qty; i++) {
  delay(zc);
  digitalWrite(ON_Board_LED, LOW);
```

```
    delay(zc);
    Serial.print(sensor_name[i]);
    Serial.print(" => ");
    Serial.print(sensor_cloud[i]);
    Serial.println(";");
    digitalWrite(ON_Board_LED, HIGH);
  }
}
void putDataByGet(int id) {
  digitalWrite(ON_Board_LED, LOW);
  delay(z);
  digitalWrite(ON_Board_LED, HIGH);
  Serial.println("Reading data from sensor ");
  Serial.print(sensor_name[id]);
  Serial.print(" => ");
  Serial.print(id);
  Serial.print(" = ");
  Serial.print("Status : ");
  Serial.print(statusCloud);
  Serial.println(";");
  value_sensor[id] = sensor_now[id];
  Serial.println(value_sensor[id]);
  getData = "id=" + String(id + 1) + "&value=" + String(value_sensor[id]);
  Destination = "arduino_php/getSecureStatus.php/?";
  URL_Server = host_or_IPv4 + Destination + getData;
  http.begin(client, URL_Server);                                //--> Specify request destination
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");  //Specify content-type header
  int httpCodeGet = http.POST(Destination);                     //--> Send the request
  getdbData();
}
void getdbData() {
  digitalWrite(ON_Board_LED, LOW);
  delay(z);
  digitalWrite(ON_Board_LED, HIGH);
  Destination = "arduino_php/getSecureData.php";
  URL_Server = host_or_IPv4 + Destination;
  http.begin(client, URL_Server);                                //--> Specify request destination
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");  //Specify content-type header
  int httpCodeGet = http.POST(Destination);   //--> Send the request
  payloadGet = http.getString();    //--> Get the response payload from server
  Serial.print("Response Code : "); //--> If Response Code = 200 means Successful connection, if -1 means connection failed.
  Serial.println(httpCodeGet);
  Serial.println("Returned data from Server: ");
  Serial.println(payloadGet);  //--> Print request response payload
  decodeJASON(payloadGet);
}
void decodeJASON(String input) {
  /*Decode db value JASON form db server */
  Serial.println();
  Serial.println("Decode JASON:");
  Serial.print("Status : ");
  Serial.print(statusCloud);

  Serial.println("; before JSON");
  JsonObject obj = doc.as<JsonObject>();
  deserializeJson(doc, input);
  statusCloud = obj[String("status")];
```

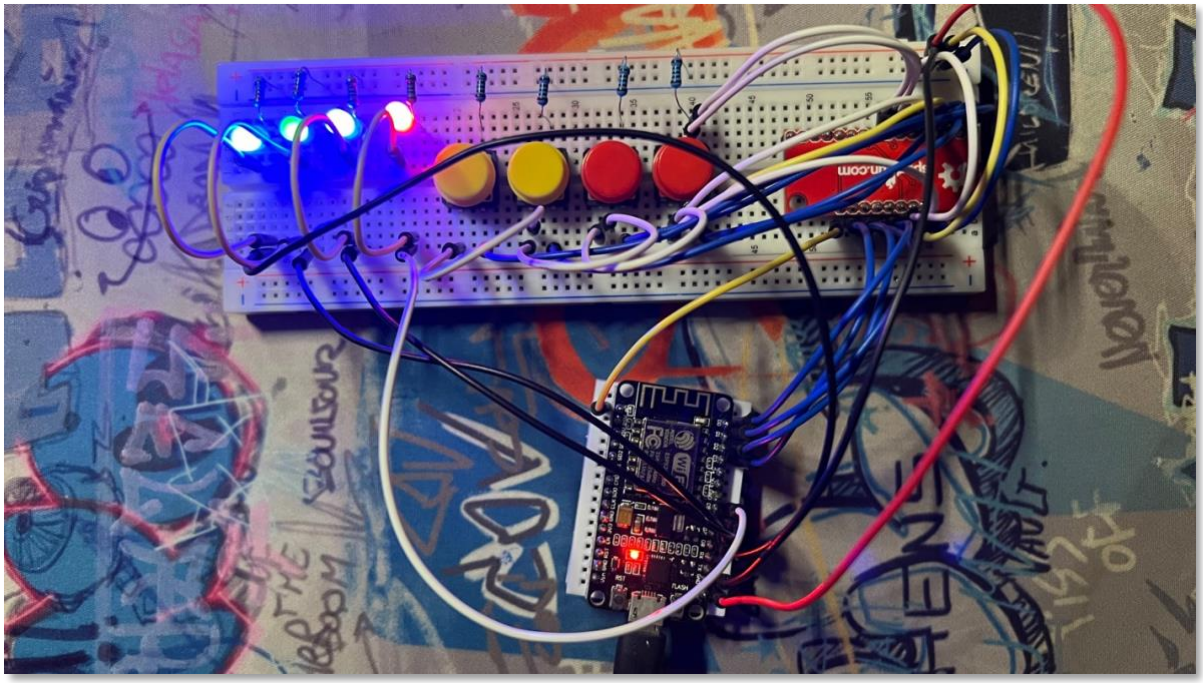
```
  Serial.print("Status : ");
  Serial.print(statusCloud);
  Serial.println("; after JSON");

  sensor_cloud[0] = obj[String("doors")];
  sensor_cloud[1] = obj[String("windows")];
  sensor_cloud[2] = obj[String("indoor_gate")];
  sensor_cloud[3] = obj[String("outdoor_gate")];
  sensor_cloud[4] = obj[String("motion")];
  sensor_cloud[5] = obj[String("glass_break")];
  sensor_cloud[6] = obj[String("perimeter")];
  sensor_cloud[7] = obj[String("smoke")];
}
void sensorRead() {
  // Channel 0 (C0 pin - binary output 0,0,0,0)
  digitalWrite(S0, LOW);
  digitalWrite(S1, LOW);
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  sensor[0] = analogRead(SIG);
  sensor_now[0] = sensor_status(sensor[0], sensor_now[0]);
  putDataByGet(0);

  // Channel 1 (C1 pin - binary output 1,0,0,0)
  digitalWrite(S0, HIGH);
  digitalWrite(S1, LOW);
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  sensor[1] = analogRead(SIG);
  sensor_now[1] = sensor_status(sensor[1], sensor_now[1]);
  putDataByGet(1);

  // Channel 2 (C2 pin - binary output 0,1,0,0)
  digitalWrite(S0, LOW);
  digitalWrite(S1, HIGH);
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  sensor[2] = analogRead(SIG);
  sensor_now[2] = sensor_status(sensor[2], sensor_now[2]);
  putDataByGet(2);

  // Channel 3 (C3 pin - binary output 1,1,0,0)
  digitalWrite(S0, HIGH);
  digitalWrite(S1, HIGH);
  digitalWrite(S2, LOW);
  digitalWrite(S3, LOW);
  sensor[3] = analogRead(SIG);
  sensor_now[3] = sensor_status(sensor[3], sensor_now[3]);
  putDataByGet(3);

  // Channel 4 (C4 pin - binary output 0,0,1,0)
  digitalWrite(S0, LOW);
  digitalWrite(S1, LOW);
  digitalWrite(S2, HIGH);
  digitalWrite(S3, LOW);
  sensor[4] = analogRead(SIG);
  sensor_now[4] = sensor_status(sensor[4], sensor_now[4]);
  putDataByGet(4);
```

```
  // Channel 5 (C5 pin - binary output 1,0,1,0)
  digitalWrite(S0, HIGH);
  digitalWrite(S1, LOW);
  digitalWrite(S2, HIGH);
  digitalWrite(S3, LOW);
  sensor[5] = analogRead(SIG);
  sensor_now[5] = sensor_status(sensor[5], sensor_now[5]);
  putDataByGet(5);

  // Channel 6 (C6 pin - binary output 0,1,1,0)
  digitalWrite(S0, LOW);
  digitalWrite(S1, HIGH);
  digitalWrite(S2, HIGH);
  digitalWrite(S3, LOW);
  sensor[6] = analogRead(SIG);
  sensor_now[6] = sensor_status(sensor[6], sensor_now[6]);
  putDataByGet(6);

  // Channel 7 (C7 pin - binary output 1,1,1,0)
  digitalWrite(S0, HIGH);
  digitalWrite(S1, HIGH);
  digitalWrite(S2, HIGH);
  digitalWrite(S3, LOW);
  sensor[7] = analogRead(SIG);
  sensor_now[7] = sensor_status(sensor[7], sensor_now[7]);
  putDataByGet(7);

  /* state value for sensor 0 - 7 */
  for (int i = 0; i < sensor_name_qty; i++) {
    delay(zc);
    Serial.print("Sensor ");
    Serial.print(i);
    Serial.print(" : ");
    Serial.println(sensor[i]);
  }
}
int sensor_status(int sensor_value, int status_value) {
  /* SQL value 2 - alarm,  1 - armed/lock, 0 - disarmed/unlock, ( switch on sensor[i] value 1024 )*/
  if (sensor_value > 1000 && status_value == 1) {
    return 2;
  }
  if (sensor_value > 1000 && status_value == 2) {
    return 2;
  }
  if (sensor_value < 1000 && status_value == 1) {
    return 1;
  }
  if (sensor_value < 1000 && status_value == 2) {
    return 2;
  }
  return 0;
}

void external() {
  /* External devices are controlled via the analog port D5 - D8 */
  if (sensor_cloud[0] == 1) {
    digitalWrite(D5, HIGH);
```

```
} else if (sensor_cloud[0] == 0) {
  digitalWrite(D5, LOW);
 }
 if (sensor_cloud[2] == 1) {
  digitalWrite(D6, HIGH);
 } else if (sensor_cloud[2] == 0) {
  digitalWrite(D6, LOW);
 }
 if (sensor_cloud[3] == 1) {
  digitalWrite(D7, HIGH);
 } else if (sensor_cloud[3] == 0) {
  digitalWrite(D7, LOW);
 }
 if (sensor_cloud[6] == 1) {
  digitalWrite(D8, HIGH);
 } else if (sensor_cloud[6] == 0) {
  digitalWrite(D8, LOW);
 }
}
```

***pav.** 8 WiFi_esp_8266_db_PHP_1_1.ino faile esantis kodas.*

Užduoties realizavimo nuotrauka, (**pav.** 9).



***pav.** 9 projekto nuotrauka.*

Užduoties realizavimo vaizdo įrašas (darbalaukis ir vaizdo kamera):

https://youtu.be/5rds_DHeRBs

Užduoties realizavimo vaizdo įrašas tik vaizdo kamera:

https://youtu.be/KxZX_QDN6JQ

Vartotojo sąsajos GitHub saugykla:

https://github.com/Audriusvilnius/Two-Factor-Authentication

Įrenginio sąsajos GitHub saugykla:

https://github.com/Audriusvilnius/arduino_php

NodeMCU ESP8266 realizuoto kodo GitHub saugykla:

https://github.com/Audriusvilnius/WiFi_ESP_8266_db_PHP

LITERATŪRA

Internetiniai šaltiniai:

1. https://www.arduino.cc/reference/en/

PRIEDAI

1. *WiFi_esp_8266_db_PHP_1_1.zip;*