

Sistem gateway

Student: Stoica Andreea
grupa: 331AA

CUPRINS:

1.PREZENTAREA PROBLEMATICII - PG 3

2.SOLUTIA PROPUSA - PG 3

3.DESCRIEREA FUNCTIONALITATILOR - PG 4

**4.DESCRIERE SOFTWARE + COMPONENTE
HARDWARE - PG 7**

5.REFERINTE BIBLIOGRAFICE - PG 13

1. PREZENTAREA PROBLEMATICII

Odata cu evolutia rapida a tehnologiei, se remarca o dorinta colectiva de inserare a virtualului in domeniile adiacente existentei umane - fie ca este vorba de interiorul propriei locuinte sau de mediul industrial. Internet of Things este un concept ce incepe sa castige teren pe pietele actuale de desfacere, pliindu-se pe tendinta de automatizare a mediilor in care ne desfasuram activitatea sau cu ajutorul carora cream produse.

Un sistem gateway este un sistem folosit pentru coordonarea unei retele de senzori si elemente de achizitionare - practic un punct de control si de centralizare al informatiei. Aceste tipuri de sisteme reprezinta un avantaj major, deoarece se pot crea retele vaste de colectare a informatiei, iar prin tehnologia de Low Power elementele ce transmit au o durata de viata mare. Acest lucru duce implicit la acoperirea unei arii extinse de catre reseaua programata.

Elementele principale ale unui gateway sunt capacitatea de a comunica cu deviceuri mai putin inteligente (microcontrolere) si capacitatea de centralizare si stocare a informatiei obtinute. Gatewayul trebuie sa fie un sistem de sine statator - independent de alte sisteme de ordin superior si cu o putere de calcul relativ mare pentru a putea manageria functionalitatile dorite. Implementarea unei structuri de securitate la nivelul gatewayului, dar si prin deviceurile utilizate in retea, reprezinta un punct in plus pentru ansamblul astfel creat.

O aplicatie a unui astfel de sistem o constituie zona de Home Automation - proces prin care ne personalizam spatiile in functie de nevoile si dorintele noastre pentru a maximiza confortul. Am orientat tema proiectului inspre acest domeniu din fascinatie pe care o am pentru imbunatatirea traiului prin solutii tehnice - chiar daca poate fi considerata o nuanta usor nesemnificativa cu tehnologia disponibila actuala si nepopularitatea acesteia.

Totusi, se observa o tendinta crescuta inspre procesele embedded si cred ca domeniul reprezinta un punct de cotitura in evolutia umana si in felul in care percepem, ca specie, tehnologia si interactiunea acesteia cu viata cotidiana, de aici si motivatia mea pentru alegerea acestei problematice.

2. SOLUTIA PROPUSA

In urma unui research amanuntit a solutiilor de Home Automationa disponibile, pot clasifica acest domeniu in 2 mari ramuri: solutiile profesionale si solutiile neprofesionale - cele ce se pot realiza de acasa cu ajutorul unor senzori usor accesibili si a unor placi de programare - microcontrolere.

Exista solutii profesionale ce ofera o gama intreaga pentru automatizarea locuintei precum cei de la SALUS [6] - in special in zona de ambianta - reglarea temperaturii unei camere si monitorizarea si comanda consumului energetic. In general, aceste solutii sunt simplu de instalat si configurat, insa pretul este unul relativ mare pentru dispozitiv - mai ales in comparatie cu varianta neprofesionala.

Pentru solutia neprofesionala, exista chiar si platforme precum Mozilla IoT si EasyIoT ce iti ofera posibilitatea descarcarii unei imagini sau a unei arhive ce contine partea de setup a elementului de tip gateway. In ambele cazuri, elementul de tip gateway trebuie sa fie un Raspberry Pi.

Mozilla IoT este inca in stadiu incipient de popularizare si nu exista foarte multe resurse in mediul online legate de interactiunea dintre senzori si gateway.

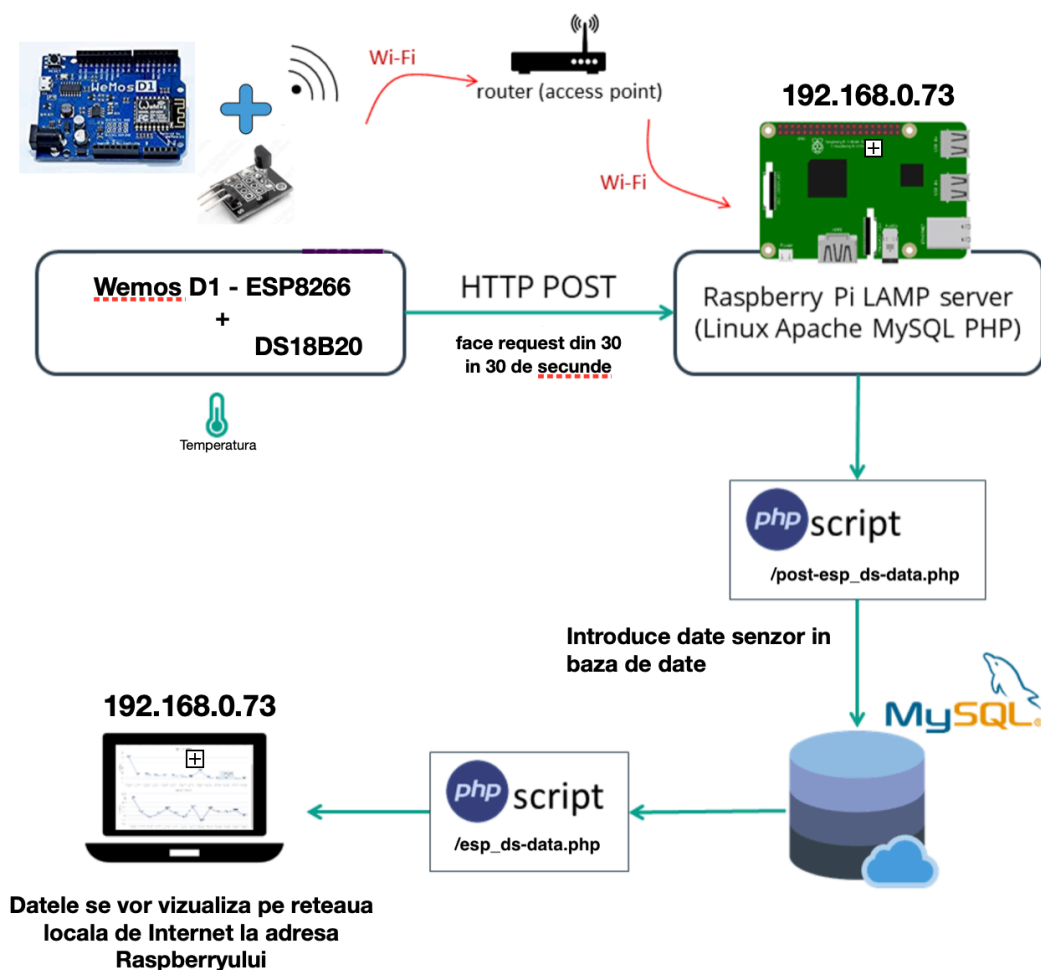
EasyIoT ofera o gama mai variata de tutoriale pentru diferiti senzori si elemente de actionare, insa din cauza faptului ca partea de programare a serverului si a paginii web a fost facuta de dezvoltator, se rateaza anumite puncte de configurare esentiale pentru un grad inalt de personalizare, in cazul in care aplicatia nu se pliaza exact pe tutorialele disponibile.

Astfel, solutia aleasa a fost crearea unui server web pe Raspberry Pi 4 si ulterior stabilirea comunicatiei de tip Ethernet prin intermediul ESP8266 si ESP32. Pentru partea de comunicatie fara internet am ales tehnologia nRF ce foloseste unde Radio pentru a transmite informatii. Consider ca aceasta solutie ofera un grad inalt de personalizare si ofera o optiune de dezvoltare mult mai potentata pentru proiecte viitoare.

3. DESCRIEREA FUNCTIONALITATILOR

Gatewayul, reprezentat de Raspberry Pi 4 va avea 2 functii principale - aceea de web server pentru comunicatia wireless prin Internet si aceea de receptor pentru comunicatia Radio. Pe langa acesta se adauga microcontrolerele la care sunt conectati senzorii.

Funcția de web server s-a realizat prin intermediul Apache. Am configurat o baza de date pentru acesta - MariaDB, iar pentru administrarea acesteia am folosit PhpMyAdmin, urmand ca interfata sa fie codata in PHP. Pentru o intelegere mai buna am introdus explicatii in diagrama de mai jos:



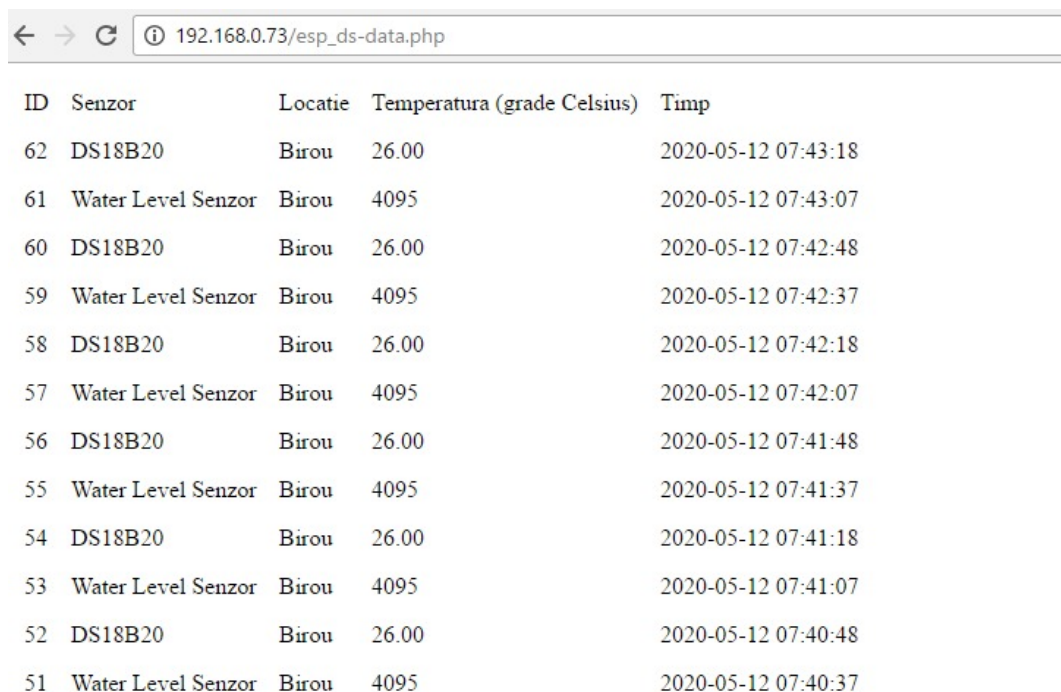
Partea de comunicatie prin intermediul unui protocol Ethernet este realizata cu ajutorul Wemos D1 - o placa de dezvoltare bazata pe Arduino Uno ce are integrat un chip ESP8266 - pentru comunicatie Wi-Fi. Chipul are un cost redus si poate folosi suita de protocoale de Internet TCP/IP.

Wemos D1 trimite valorile primite de la un senzor de temperatura (DS18B20) catre gateway - printr-un access point (routerul Wi-Fi). Protocolul folosit pentru trimiterea datelor este HTTP - un protocol prin care se distribuie si se colecteaza date de tip media prin intermediul unor hyperlinkuri. Acesta functioneaza pe baza procesului de tip client-server - in cazul meu serverul este reprezentat de Raspberry Pi, iar clientul care face request si post este Wemos D1.

Al doilea dispozitiv ce trimite prin internet este o placa de dezvoltare de la Heltec - Wifi Kit 32 - ce are un chip cp2102 bazat pe esp-32 ce foloseste acelasi principiu ca esp8266 si are performante asemanatoare. Placa este conectata la un senzor de nivel pentru apa (HW-038), preia nivelul lichidului si il trimite in acelasi mod catre gateway - foloseste tot protocolul HTTP. In diagrama de mai sus, putem inlocui Wemos D1 cu Heltec WiFi Kit 32 pentru a vizualiza procesul.

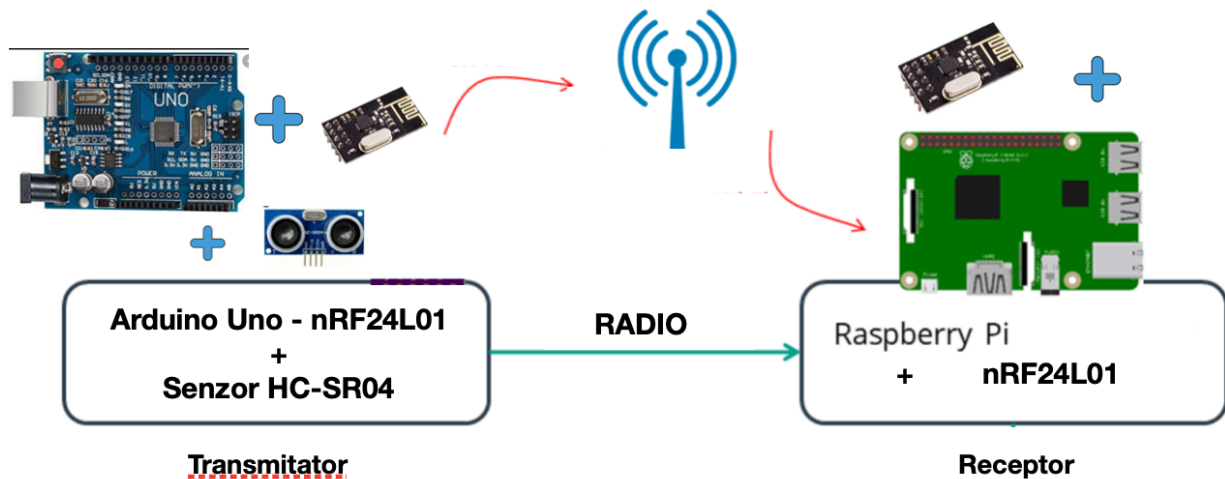
Gatewayul (pe post de web server) insereaza valorile primite (de la ambele dispozitive) in baza de date prin fisierul php post-esp_de-data.php. Afisarea se realizeaza prin intermediul unei alte pagini php ce preia datele din baza de date - esp_ds-data.php.

In diagrama de mai jos putem observa pagina php esp_ds-data.php prin intermediul careia se afiseaza datele trimise prin Wi-Fi de la cele 2 placi de dezvoltare:



ID	Senzor	Locatie	Temperatura (grade Celsius)	Timp
62	DS18B20	Birou	26.00	2020-05-12 07:43:18
61	Water Level Sensor	Birou	4095	2020-05-12 07:43:07
60	DS18B20	Birou	26.00	2020-05-12 07:42:48
59	Water Level Sensor	Birou	4095	2020-05-12 07:42:37
58	DS18B20	Birou	26.00	2020-05-12 07:42:18
57	Water Level Sensor	Birou	4095	2020-05-12 07:42:07
56	DS18B20	Birou	26.00	2020-05-12 07:41:48
55	Water Level Sensor	Birou	4095	2020-05-12 07:41:37
54	DS18B20	Birou	26.00	2020-05-12 07:41:18
53	Water Level Sensor	Birou	4095	2020-05-12 07:41:07
52	DS18B20	Birou	26.00	2020-05-12 07:40:48
51	Water Level Sensor	Birou	4095	2020-05-12 07:40:37

Functia de receiver Radio a gatewayului implica conectarea unui modul nRF24L01 la Raspberry Pi 4 si activarea interfetei SPI pentru a putea comunica in lungimea de banda de 2.4 Ghz oferita de modul. Astfel, Raspberrylul va avea rolul unui receptor - va trebui sa asculte pe un anumit canal si sa preia informatia receptionata de pe acesta, urmand ca aceasta sa fie afisata in terminal. Limbajul utilizat pentru acest lucru este C++.



Comunicatia se realizeaza prin intermediul tehnologiei nRF - o tehnologie bazata pe unde Radio de lungime de banda 2.4 GHz. Modulul nRF poate comunica pe un total de 125 de canale diferite, fiecare putand adresa 6 adrese separate. Acesta inregistreaza viteze intre 250 kbps pana la 2 Mbps (pot fi selectate prin cod) si poate ajunge la distante de 100 de metri in cazul in care viteza de transmisie este una redusa. Un alt avantaj al acestuia este faptul ca are un consum extrem de mic in timpul transmisiei - 12mA - mai putin decat un LED obisnuit.

Am conectat un modul nRF24L01 pe o placa de dezvoltare Arduino, urmand ca aceasta sa aiba rolul de transmitator. Modulul preia date de la senzorul cu ultrasunet (HC-SR04) conectat si semnaleaza receptorului daca in fata acestuia se afla sau nu un obiect.

Diagrama de mai jos reprezinta terminalul Raspberryului, in care se afiseaza datele trimise prin unde Radio de la senzorul ultrasonic:

```
===== SPI Configuration =====
CSN Pin      = CE0 (PI Hardware Driven)
CE Pin       = Custom GPIO17
Clock Speed   = 8 Mhz
===== NRF Configuration =====
STATUS       = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
RX_ADDR_P0-1 = 0x7878787878 0xc2c2c2c2c2
RX_ADDR_P2-5 = 0xc3 0xc4 0xc5 0xc6
TX_ADDR      = 0xe7e7e7e7e7
RX_PW_P0-6   = 0x20 0x00 0x00 0x00 0x00 0x00
EN_AA        = 0x3f
EN_RXADDR    = 0x03
RF_CH        = 0x05
RF_SETUP     = 0x05
CONFIG       = 0x0e
DYNPD/FEATURE = 0x3f 0x04
Data Rate    = 1Mbps
Model        = nRF24L01+
CRC Length   = 16 bits
PA Power     = PA_HIGH
Start listening...
Pipe : 0 Size : 32 Data : Nu este nimic in fata senzorului
Pipe : 0 Size : 32 Data : Nu este nimic in fata senzorului
Pipe : 0 Size : 32 Data : Senzorul a detectat o prezenta!
Pipe : 0 Size : 32 Data : Senzorul a detectat o prezenta!
Pipe : 0 Size : 32 Data : Senzorul a detectat o prezenta!
Pipe : 0 Size : 32 Data : Nu este nimic in fata senzorului
Pipe : 0 Size : 32 Data : Nu este nimic in fata senzorului
Pipe : 0 Size : 32 Data : Senzorul a detectat o prezenta!
Pipe : 0 Size : 32 Data : Nu este nimic in fata senzorului
Pipe : 0 Size : 32 Data : Senzorul a detectat o prezenta!
Pipe : 0 Size : 32 Data : Senzorul a detectat o prezenta!
^Z
[1]+  Stopped                  sudo ./receiver
```

O alta functionalitate importanta a Raspberryului este capacitatea de a porni direct - odata cu alimentarea, serverul web este gata pentru accesare si nu necesita utilizarea vreunei comenzi de activare. Totusi, pentru partea de comunicatie nRF nu am reusit sa implementez pornirea directa.

Fiecare element in parte are alimentarea portabila - Raspberry Pi prin intermediul unui powerbank, Wemos D1 printr-o mufa DC conectata la o baterie de 9V, WiFi Kit 32 printr-un powerbank si Arduino Uno printr-un circuit de alimentare realizat dintr-un fir pentru alimentarea Vcc si unul pentru GND, sudate intre ele si ulterior sudate la un capat de alimentare pentru o baterie de 9V.

4. DESCRIEREA SOFTWARE + COMPONENTE HARDWARE

Toate codurile pot fi regasite pe linkul de github de la sectiunea de referinte bibliografice [5]. Urmeaza sa ofer o descriere software pentru partile de interes - cele ce realizeaza comunicatia Wi-Fi si comunicatia Radio.

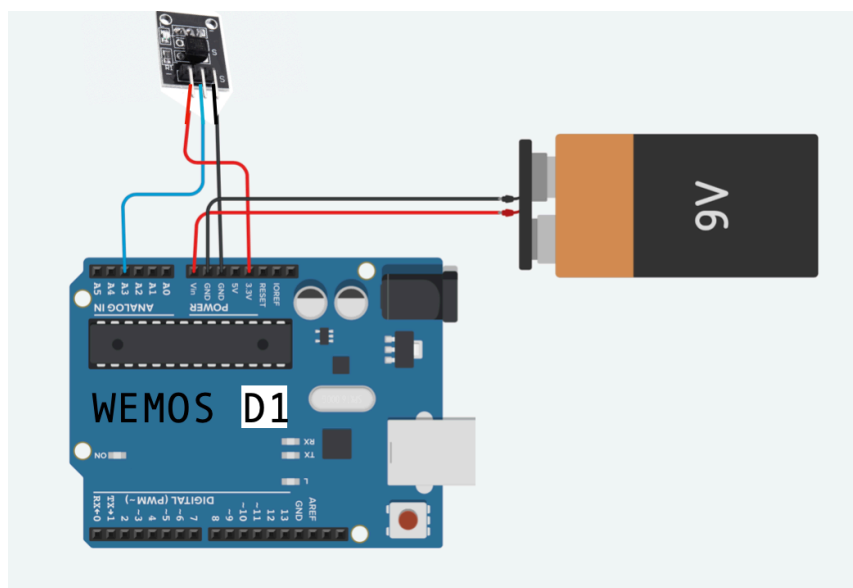
Portiunile de cod de mai jos contin comentarii pentru explicarea acestora - pentru o parcurgere mai usoara. Astfel, reiese din acestea implementarea functionalitatii dorite, punctele atinse reprezentand pasii ordonati pentru realizarea acesteia.

Setup web server Raspberry Pi

Pe Raspberry Pi am instalat raspbian OS - sistem de operare bazat pe Linux, iar configurarea apache, mariadb, mysql, phpmyadmin a fost facuta conform [1].

Wemos D1 + ESP8266 + DS18B20

Pentru o mai buna intelegere, voi incepe cu partea de Wemos D1 + senzor de temperatura. Am calibrat senzorul de temperatura conform [1] si am instalat esp2866 si bibliotecile necesare pentru a lucra cu microcontrolerul. Schema de conexiune electrica este prezentata in figura urmatoare:



```

/*aceste biblioteci sunt folosite pentru conectarea la
Wi-Fi si pentru lucrul cu HTTP - protocolul folosit pentru a transmite date
catre un Access Point (serverul web ce poate fi accesat prin URL sau adresa IP
in cazul meu)*/
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

const char* serverName = "http://192.168.0.73/post-esp_ds-data.php"; /*adresa
alocata raspberryului de catre routerul Wi-Fi este 192.168.0.73 - vom accesa
fisierul post-esp_ds-data.php pentru a insera datele in baza de date*/
WiFi.begin(ssid, password); /*conectarea la Wi-Fi cu ajotul usernamului si
parolei furnizate de router*/

HTTPClient http; //obiect pentru conexiunea prin protocolul HTTP
http.begin(serverName); /*conectare prin HTTP la web server/

sensors.requestTemperatures();
RawValue = sensors.getTempCByIndex(0); //se citeste temperatura de la senzor

/*se pregateste trimiterea requestului HTTP - acesta este un sir format din
valorile pe care doresc sa le trimit, in inceput continand API KEY (cheia de
conectare) la web server (mai exact interfata acestuia) - aceeasi se gaseste si
la fisierul ce va primi datele si reprezinta o metoda de securitate*/
String httpRequestData = "api_key=" + apiKeyValue + "&sensor=" + sensorName
                        + "&location=" + sensorLocation + "&value1=" +
                        String(Celcius) + "";

//trimite requestul HTTP
int httpResponseCode = http.POST(httpRequestData);

```

WiFi Kit 32 + ESP32 + Water Level Sensor HW-038

Schema de conexiune:

WiFi Kit32	Water Level Sensor - HW-038
GND	GND
5V	VCC
pin 25	Data

WiFi Kit 32	Powerbank
port micro USB	USB

Codul folosit este intr-o proportie mare acelasi, am schimbat partea de citire de la senzor si de trimitere a requestului HTTP pentru a trimite valoarea preluata de la senzorul de nivel de apa. Value reprezinta valoarea citita de la senzor.

```
String httpRequestData = "api_key=" + apiKeyValue + "&sensor=" + sensorName  
                        + "&location=" + sensorLocation + "&value1=" +  
                        String(value) + "";
```

Fisiere php Raspberry Pi - pentru inserare date in baza de date si afisare date din baza de date, conform [3].

Am creat o baza de date, prin intermediul phpmyadmin, in care voi stoca temperatura si nivelul apei.

Pentru a insera datele in baza de date, folosesc fisierul post-esp_ds-data.php.

```
//cheie de autentificare pentru accesul la interfata php
/*o va compara cu ce primeste de la Wemos + DS18B20, respectiv de la
WiFi Kit 32 + senzor nivel apa (aceasta cheie se afla si in fisierul de
pe placile de dezvoltare)*/
$api_key_value = "tPmAT5Ab3j7F9";

//daca serverul a trimis date prin http se testeaza validitatea API KEY
//ulterior preia valorile pentru a le insera in baza de date
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
    $api_key = test_input($_POST["api_key"]); //descria mai jos
    if($api_key == $api_key_value)
    {
        $sensor = test_input($_POST["sensor"]);
        $location = test_input($_POST["location"]);
        $value1 = test_input($_POST["value1"]);

        //crearea conexiunii la baza de date
        $conn = new mysqli($servername, $username, $password, $dbname);
        // verifica daca conexiunea s-a facut si printeaza eroarea in cazul in
care nu s-a realizat
        if ($conn->connect_error) {
            die("Connection failed: " . $conn->connect_error);
        }

        //insereaza valorile primite de la senzor in baza de date
        $sql = "INSERT INTO SensorData (sensor, location, value1)
VALUES ('" . $sensor . "', '" . $location . "', '" . $value1 . "')";

        //verifica daca queryul a fost executat cu succes
        if ($conn->query($sql) === TRUE)
            echo "New record created successfully!";
        else
            echo "Error: " . $sql . "<br>" . $conn->error;
```

PHP

```

        //inchide conexiunea la baza de date
        $conn->close();
    }
    else
        echo "Wrong API Key provided!"; //caz pentru care api keyul nu coincide
        cu cel trimise de esp8266 sau esp32
}
else
    echo "No data posted with HTTP POST!.";

/*preia valorile trimise de esp2866 sau esp32 si le retine in format specific
html*/
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

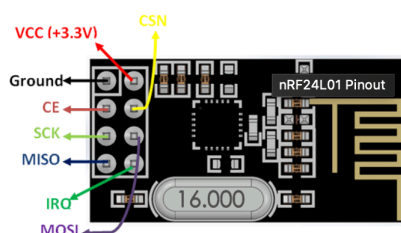
```

Pentru afisarea din baza de date a temperaturilor si a nivelelor inregistrate, folosesc fisierul de tip php esp_ds-data.php. Acesta se conecteaza la serverul gateway unde se afla baza de date "esp_ds_data" folosind username-ul "admin" si password "parola". Ulterior creaza un tabel in care va afisa valorile preluate din baza de date. Am atasat si acest fisier in linkul cu git ce contine tot proiectul [5].

Setup Raspberry Pi nRF24L01 conform [3]:

Am activat interfata SPI pentru comunicatii Radio cu comanda conform [4]:

Schema de conexiune:



nRF24L01+	Raspberry Pi
GND	6 / GND
VCC	1 / 3.3V
CE	11 / GPIO17
CSN	24 / GPIO8 / SPI_CE0_N
SCK	23 / GPIO11 / SPI_CLK
MOSI	19 / GPIO10 / SPI_MOSI
MISO	21 / GPIO9 / SPI_MISO
IRQ	—

Raspberry Pi va avea functia de receptor - pentru acest lucru am compilat fisierul simple_r.cpp si i-am schimbat numele in receiver.

Am downloadat si biblioteca RF24 pentru comunicatie radio prin modulul nRF24L01.

Codul din simple_r.cpp:

```
#include "../RF24/RF24.h" //am inclus libraria pentru a lucra cu
```

nRF24L01

```
int main() {
    RF24 radio(PIN_CE, PIN_CSN); //creeaza un obiect comandat de pinul GPIO17 de
    chip enable si cel de Chip Set Enable
    radio.begin(); //incepe comunicatia radio prin obiectul de tip rf24
    radio.setChannel(5); //seteaza canalul de ascultare (canalul 5)
    radio.setPALevel(RF24_PA_HIGH); /*seteaza puterea pe maxim, deoarece am dorit
    sa se poata comunica la cea mai mare distanta - putere max -> distanta max*/
    radio.setDataRate(RF24_1MBPS); //seteaza viteza transmisiei
    radio.enableDynamicPayloads(); /*functie ce ajuta la receptia ordonata a
    bitilor din sirul de caractere ce va fi transmis (un bit - un caracter)*/

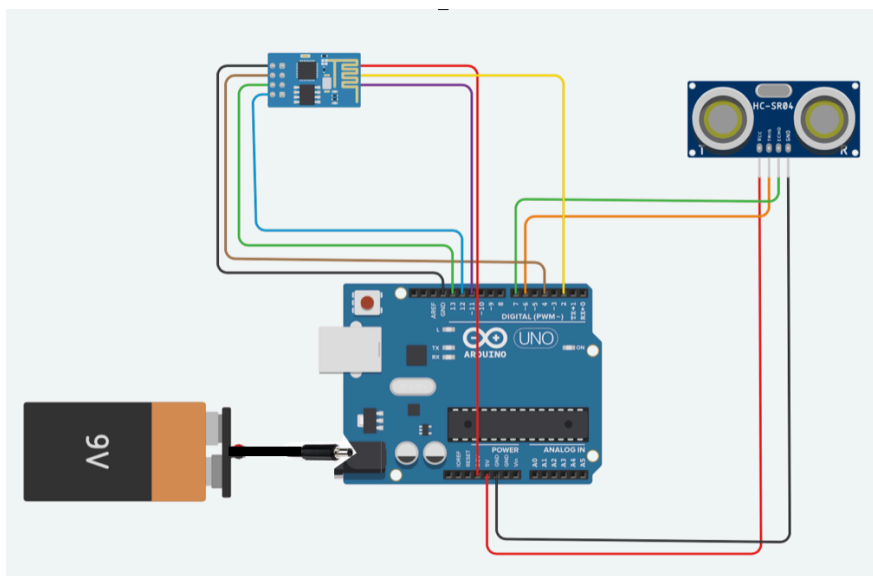
    radio.openReadingPipe(0, 0x7878787878LL); //deschide adresa la care se va face
    comunicatia
    radio.startListening(); //incepe sa asculte

    /*functia va prelua sirul de date bit cu bit si il va reconstrui ulterior
    while (true) {
        if (radio.available(&pipeNumber)) {
            payloadSize = radio.getDynamicPayloadSize();
            char payload[payloadSize];
            string receivedData;
            radio.read(&payload, sizeof(payload));

            for (uint8_t i = 0; i < payloadSize; i++)
                receivedData += payload[i]; /*construiesc sirul nou cu mesajul
            receptionat*/

            cout << "Data : " << receivedData << endl; //afiseaza mesajul receptionat
```

Arduino - nRF24L01 + HC-SR04 conform [3]



```

const uint64_t pipeNum = 0x7878787878LL; //adresa pe care o va
folosi modulul pentru comunicatie nRF

RF24 radio(PIN_CE, PIN_CSN); /*initializeaza un obiect de tip RF24 pentru
comunicatia RADIO*/

void setup() {

    bool var = radio.begin(); //incepe comunicatia

    radio.setChannel(5); //seteaza acelasi canal ca a receptorului
    radio.setDataRate (RF24_1MBPS); //seteaza viteza de transmisie a informatiei
    radio.setPALevel(RF24_PA_HIGH); /*seteaza puterea pe maxim, deoarece am dorit
sa se poata comunica la cea mai mare distanta - putere max -> distanta max*/
    radio.enableDynamicPayloads(); /*functie ce ajuta la receptia ordonata a
bitilor din sirul de caractere ce va fi transmis (un bit - un caracter)*/

    radio.openWritingPipe(pipeNum); /*activeaza scrisul la adresa indicata - este
aceeasi cu a receptorului (gatewayului)*/
}

/*am folosit o functie pentru a identifica cu ajutorul senzorului daca se afla
sau nu un obiect inaintea lui
se trimite catre receptor mesajul corespunzator - Senzorul a detectat o
prezenta/Nu este nimic in fata senzorului */
    if (k==0)
    {
        if (radio.write(&text0, sizeof(text0)))
            Serial.println("Delivered " + String(sizeof(text0)) + " byte");
        else
            Serial.println("Data not delivered");
    }
    else
    {
        if (radio.write(&text1, sizeof(text1)))
            Serial.println("Delivered " + String(sizeof(text1)) + "
byte");
        else
            Serial.println("Data not delivered");
    }

    delay(3000); //transmisia se face din 30 in 30 de secunde

```

6. REFERINTE BIBLIOGRAFICE

1. <https://www.instructables.com/id/Calibration-of-DS18B20-Sensor-With-Arduino-UNO/>
2. <https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2>
3. <https://randomnerdtutorials.com/esp32-esp8266-raspberry-pi-lamp-server/>
4. <https://www.hackster.io/wirekraken/connecting-an-nrf24l01-to-raspberry-pi-9c0a57>
5. <https://github.com/andreeastoica2212/gateway.git>
6. https://www.saluscontrols.ro/wp-content/uploads/Ghid-rapid-de-instalare-sistem-iT600-VS20_UGE600_TRV10RFM_CO10RF_RX10RF_KL08RF.pdf