

## Inteligentná poštová schránka

Návod na zhotovenie inteligentnej poštovej schránky, ktorá bude posielať správy na e-mail pri otvorení dverí na poštovej schránke.

Cieľom našej práce je demonštrovať odosielanie správ prostredníctvom MQTT (Message Queue Telemetry Transport) s použitím mikrokontroléra ESP32 a Raspberry Pi v programe Node-Red.

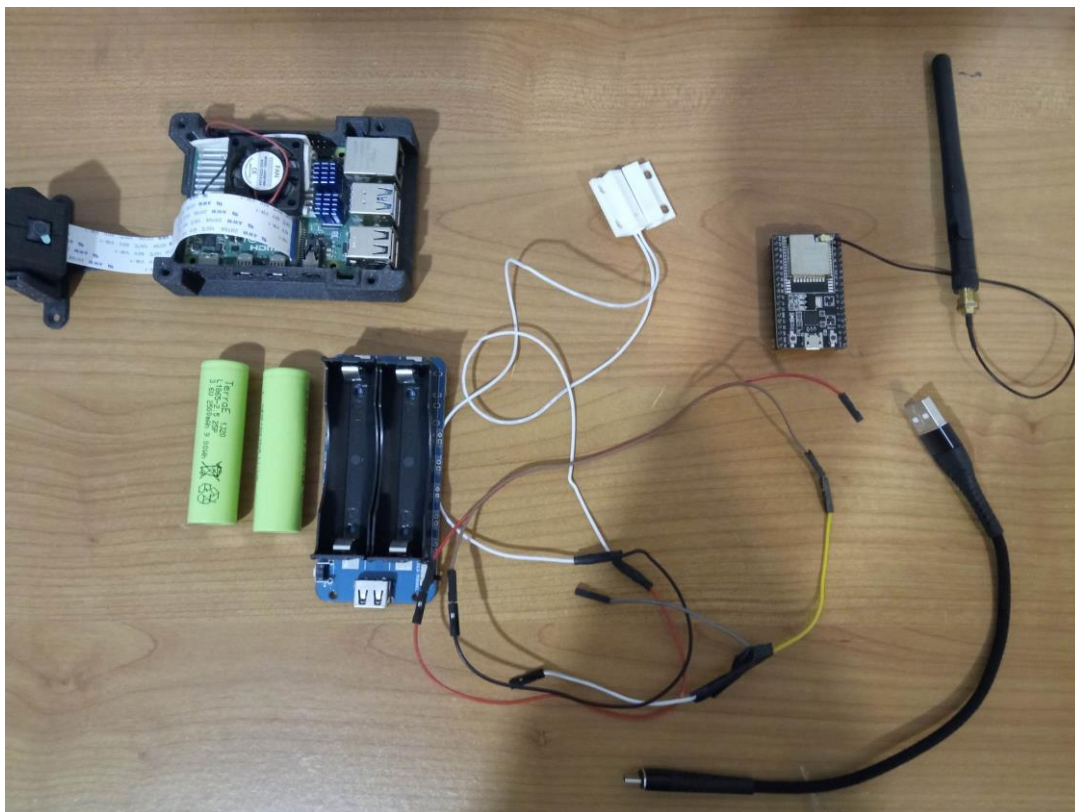
### Pomôcky:

#### Hardvér

- Raspberry Pi (SD karta 8 GB),
- Mikrokontrolér ESP32,
- Externý napájací zdroj pre ESP32,
- Kábel MicroUSB,
- Magnetický senzor,
- Prepojovacie vodiče,
- 10k rezistor.

#### Softvér

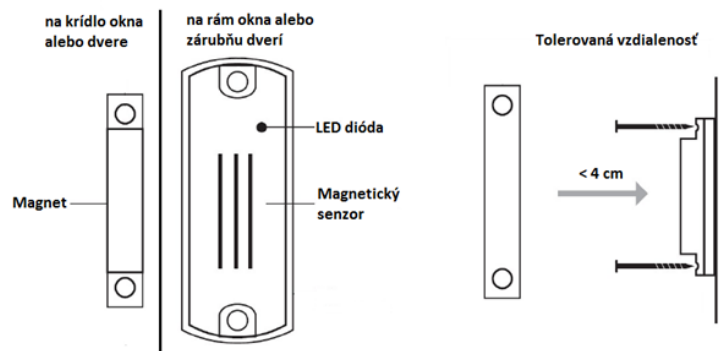
- Node-RED,
- Program Arduino IDE,
- MQTT server.



Obr.1 Pomôcky - hardvér

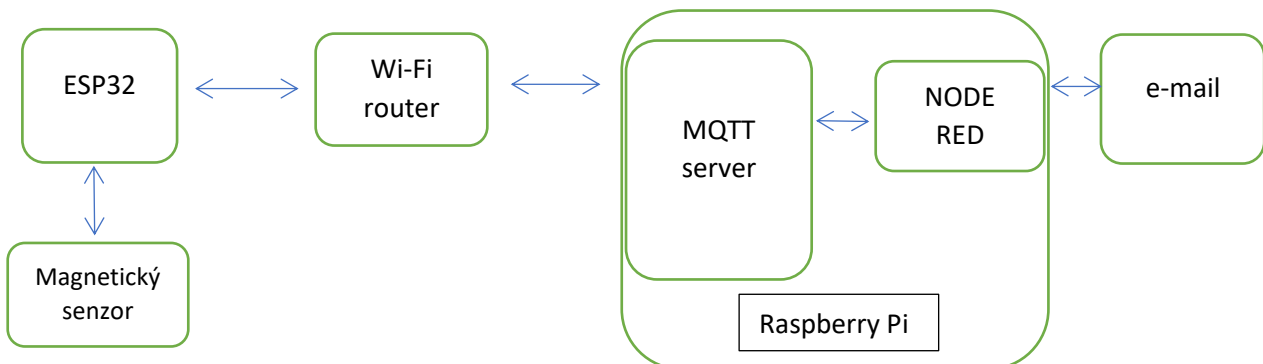
## Postup

Pri našej práci použijeme mikrokontrolér ESP32, ktorý bude slúžiť na posielanie správ o doručení pošty pri otvorení dverí na schránke. Princíp je založený na magnetickom senzore. Na dvierka schránky nainštalujeme pohyblivú časť magnetického senzora, druhú časť pripojíme na pevnú časť schránky. V momente ako sa magnetický senzor odpojí od svojej druhej časti, nie je tam žiaden magnet. Tým sa vyšle signál cez EPS32, že boli dierka otvorené. (Obr. 2)



Obr. 2 Magnetický senzor

MQTT je sprostredkovateľ a reakcia je, že bude doručený e-mail, ktorý nás informuje, že je niečo v poštovej schránke. Na prenos údajov použijeme MQTT a SMTP protokol. Prostredníctvom Raspberry Pi si vytvoríme na miestnej úrovni MQTT klienta, ktorý bude náš naprogramovaný mikrokontrolér ESP32 využívať. Na Raspberry Pi máme server Node-Red, ktorý nám bude odosielať správy pomocou SMTP protokolu na zvolený e-mail. Na obr.3 je zobrazená bloková schéma našej inteligentnej schránky.



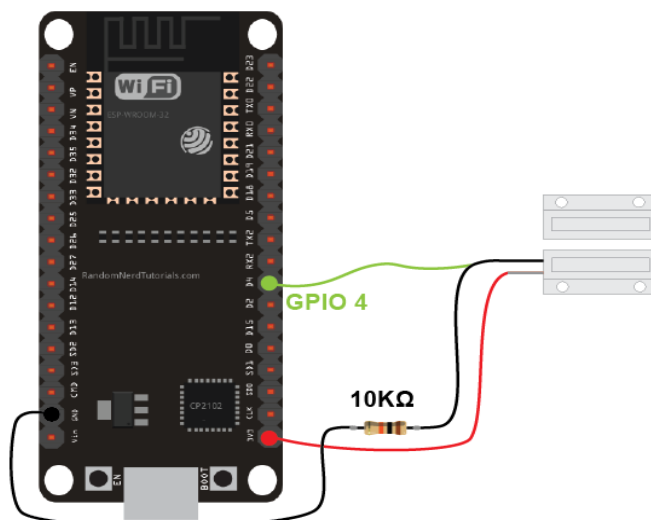
Obr.3 Bloková schéma zapojenia našej inteligentnej schránky.

1. Zapojíme mikrokontrolér EPS32 s magnetickým senzorom do obvodu podľa schémy.
2. V Arduino IDE nainštalujeme knižnice „<WiFi.h>“, „<PubSubClient>“, vyberieme v „board manager“ dosku ESP32, vytvoríme program a nahráme ho do EPS32.
3. Na Raspberry Pi nainštalujeme MQTT broker. Pred inštaláciou MQTT do Raspberry Pi si aktualizujeme operačný systém. Po dokončení aktualizácie systému nainštalujeme softvér Mosquitto a overíme, či je Mosquitto MQTT nainštalovaný a spustený.
4. Na Raspberry Pi spustíme program Node-Red, ktorý beží na porte 1880. Funkčné bloky Mqtt, Function, Email, Debug presunieme na plochu prostredia a nakonfigurujeme ich:
  - a) Nastavíme blok MQTT- zadáme „Topic“, na ktorý nám bude publikovať ESP32 správy. Ostatné nastavenia ako QoS a Output necháme v základnom stave.
  - b) Nastavíme MQTT broker. V záložke „Connection“ nastavíme „Server“. V našom prípade to bude lokálna IP adresa Raspberry Pi na ktorom nám beží už spomínaný MQTT Client.
  - c) Naprogramujeme funkčný blok funkcie. Do funkcie vstupuje správa z MQTT bloku. Naša funkcia porovnáva vstupné hodnoty s požadovanou hodnotou „1“ a to pomocou príkazu „if“. To má za následok, že ak je podmienka splnená na výstupe funkcie sa nastaví „msg.topic = Schranka Dom Krasna Horka“ a „msg.payload = Prisla posta“, ktoré vstupujú do funkčného bloku e-mailu.
  - d) Nastavíme blok e-mail. Tento blok je nutné doinštalovať cez možnosť „Manage palette“. Následne zvolíme záložku „Install“, kde vyhladáme „node-red-node-email“, ktorý následne nainštalujeme. nakonfigurujeme funkčný blok e-mailu. Po jeho rozkríknutí nastavíme odosielateľa a prijímateľa. Pre tento prípad budeme posilať sami sebe e-mail.
  - e) Do položky „To“ zadáme prijímateľa správy. Položky „Server“ a „Port“ necháme v preddefinovanom stave. Nakoniec nastavíme e-mail odosielateľa „Userid“ s jeho heslom „Password“. Keďže sa jedná o menej bezpečne nastavenie je nutné v nastaveniach google e-mailu povoliť zdroje od menej zabezpečených zariadení.
  - f) Pripojíme funkčný blok „debug“ k funkcii. Debugger nám slúži na výpis dát z funkcie. Pre uloženie a aktiváciu všetkých zmien použijeme tlačidlo „DEPLOY“ .
5. Program je pripravený na používanie. Pred demonštráciou funkcionality pripojíme ESP32 k zdroju energie. Pri rozpojení magnetického kontaktu snímača bude doručená sprava na e-mail.

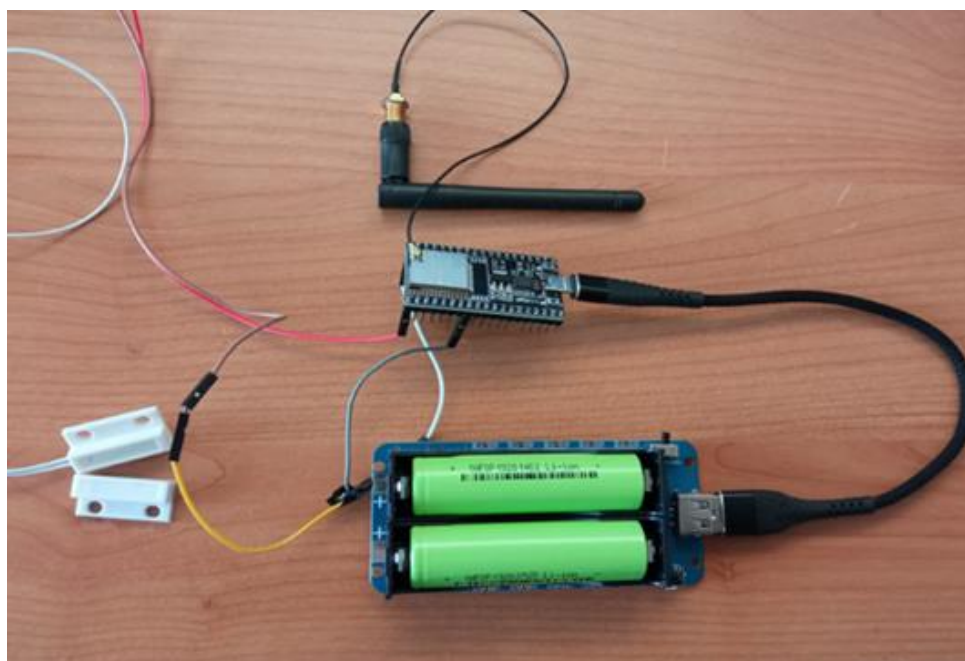
## 1. Mikrokontrolér ESP32

K mikrokontroléru ESP32 pripojíme podľa schémy magnetický senzor (obr.4). Prvý vodič zapojíme k zdroju napätia 3,3 V ( červený ). Druhý vodič musíme rozvetviť. Jeden z vodičov pripojíme priamo k GPIO4 a druhý vodič prepojíme cez rezistor (10kOhm), s pinom GND (čierny). Vodič musí byť v kombinácii s pull-up rezistorom, aby sme dosiahli stabilitu stavov HIGT a LOW.

Po zapojení elektrického obvodu prepojíme ESP32 pomocou USB s počítačom. V programovacom prostredí Arduino IDE máme nainštalované knižnice „<WiFi.h>“, „<PubSubClient>“. Následne vyberieme v „board manager“ dosku ESP32. (Ak ju Arduino IDE neobsahuje je nutné ju doinštalovať). Zvolíme Port na ktorom komunikuje naše ESP32. (ktorý mu pridelil počítač)



Obr.4 Schéma zapojenia ESP32 s magnetickým senzorom.



Obr.5 Praktické zapojenie ESP32 s magnetickým senzorom napájané externým zdrojom

## 2. Zdrojový kód schránky pre mikrokontrolér ESP32

1. Zahrnieme knižnice pre pripojenie sa k Wi-Fi sieti a knižnicu pre zasielanie správ cez MQTT. Zadefinujeme si GPIO pin 4 na ktorý je pripojený magnetický senzor a bitovú masku GPIO pinu, ktorá slúži na zobudenie EPS32 z hlbokého spánku.

```
Alena_klimcikova_schranka_esp32.ino  debug_custom.json

1 // ----- Knižnice -----
2 #include <WiFi.h> //Knižnica pre pripojenie sa ESP k Wi-fi
3 #include <PubSubClient.h> //Klientska knižnica pre zasielanie správ MQTT
4 // ----- Definovanie -----
5 #define schranka_senzor_GPIO_PIN 4 // RTC-GPIO pin. Počas hlbokého spánku môže koprocesor ULP použiť niektoré kolíky ESP32, konkrétne kolíky RTC_GPIO.
6
7 #define bitova_maska_GPIO_Pinu 0xA // 2^4 = 16 decimalne. (Použitý GPIO 4) v hex cisle je to A.
8
9 int status_schranky;
10
```

Obr.6 Časť kódu - knižnice

### 2. Nastavíme Wi-Fi a MQTT.

Preddefinujeme meno a heslo na Wi-Fi. Zadáme IP adresu servera RPI, nastavíme port 1883 kde bude počúvať MQTT. Zvolíme si používateľské meno a heslo MQTT servera. Nastavíme názov topik MQTT servera „ESP32/Schranka“ kde budeme naše dáta zdieľať.

```
10
11 // ----- Nastavenie Wi-Fi -----
12
13 const char* meno_wifi = "zadame nazov wifi"; //Použili sme teraz domácu adresu
14 const char* heslo_wifi = "zadame heslo"; // heslo
15
16
17 // ----- Nastavenie MQTT Servera -----
18 const char* mqttServer = "192.168.x.xxx"; //Adresa MQTT servera (v nasom prípade lokálna adresa RPI)
19 const int mqttPort = 1883; // Základný port MQTT
20
21 const char* mqttPouzivatel = "zadame meno pouzivatela"; //Meno používateľa MQTT
22 const char* mqtttheslo = "zadame heslo"; //Heslo používateľa MQTT
23
24 const char* topic_esp32 = "ESP32/Schranka"; //Topic MQTT
25
26 WiFiClient espClient;
27 PubSubClient client(espClient);
```

Obr.7 Časť kódu - nastavenia Wi-Fi, nastavenie MQTT servera.

### 3. Pripojíme sa k Wi-Fi.

Funkcia „void pripojenie\_wifi“, definuje pripojenie k Wi-Fi mikrokontroléra ESP32. Príkaz „WiFi.begin“ iniciuje pripojenie sa k Wi-Fi. Následne slučka „while“ overí, či je mikrokontrolér ESP32 pripojený k Wi-Fi. Pokiaľ nie je pripojený k Wi-Fi, bude čakať 0,5 sekundy a potom vypíše bodku. Ak je podmienka splnená a sme pripojený k Wi-Fi, tak v konzole sa vypíše lokálna IP adresa ESP32.

```
48 // ----- Pripojenie Wi-fi -----
49 void pripojenie_wifi() {
50
51     Serial.println(); //Prázdny riadok
52     Serial.print("Pripajanie k ");
53     Serial.println(meno_wifi); //Vypíše meno wifi
54
55     WiFi.begin(meno_wifi, heslo_wifi); //Začiatok pripajania sa k Wifi
56
57     while (WiFi.status() != WL_CONNECTED) { //Pokým nie je pripojeny k wifi každú pol sekundu načítava bodku.
58         delay(500);
59         Serial.print(".");
60     }
61
62     Serial.println();
63     Serial.println("WiFi pripojene");
64     Serial.println("IP adresa zariadenia ESP32: ");
65     Serial.println(WiFi.localIP()); //Vypíše lokálnu IP adresu ESP32
66 }
67 // ----- Zobudenie zariadenia -----
```

Obr.8 Časť kódu- pripojenie s k Wi-Fi

#### 4. Vytvoríme spojenie s MQTT serverom.

Zadefinujeme MQTT pripojenie cez funkciu „void mqtt\_pripojenie“. Príkaz „while“ vykonáva pripojenie, ak je podmienka „if“ splnená v konzole sa vypíše "Pripajam sa k MQTT Serveru ...". Ak sa podmienka nesplní, vykoná sa príkaz „else“, do konzoly vypíše text "Nepodarilo sa pripojiť k MQTT serveru" a čaká 2s, potom sa opakuje celá funkcia od začiatku slučky.

```
28 // ----- Pripojenie k MQTT serveru -----
29 void mqtt_pripojenie(){
30     client.setServer(mqttServer, mqttPort);
31     //client.setCallback(callback);
32     while (!client.connected()) { //Slučka while ktorá overuje či je pripojeny client (ESP32) k MQTT serveru (RPI)
33         Serial.println("Pripajam sa k MQTT Serveru ...");
34
35         if (client.connect("ESP32Client", mqttPouzivatel, mqttHeslo)) { //Overenie používateľa pripojenie sa k MQTT serveru (meno a heslo)
36
37             Serial.println("Pripojeny k MQTT");
38
39         } else {
40
41             Serial.print("Nepodarilo sa pripojiť k MQTT serveru");
42             Serial.print(client.state()); // Vypíše dôvod nepripojenia sa
43             delay(2000); // Ak sa nepripojí k MQTT zopakuje to o 2s
44         }
45     }
46 }
47 }
48 // ----- Pripojenie Wi-fi -----
```

Obr.9 Časť kódu- Pripojenie sa k MQTT serveru.

#### 5. Aktivujeme ESP32.

Funkcia „void dovod\_zobudenia\_zariadenia“ aktivuje naše ESP32 z hlbokého spánku ak sa zmení vstup na RTC pine. Príkaz „switch“ vyhodnotí znakový výraz v zátvorkách, hodnotu porovná postupne s hodnotami uvedenými za kľúčovými slovami „case“. Pri zhode sa vykoná príkaz uvedený za „: „ - "Schranka sa otvorila" alebo "Uplynul čas hlbokého spánku".



```
67 // ----- Zobudenie zariadenia -----
68 void dovod_zobudenia_zariadenia(){
69     esp_sleep_wakeup_cause_t zobudenie_dovod;
70
71     zobudenie_dovod = esp_sleep_get_wakeup_cause();
72
73     switch(zobudenie_dovod)
74     {
75     case ESP_SLEEP_WAKEUP_EXT0 : Serial.println("Schranka sa otvorila"); break;
76
77     case ESP_SLEEP_WAKEUP_TIMER : Serial.println("Uplynul cas hlbokého spanku"); break; // ma:
78
79     default : Serial.printf("Dovod zobudenia: %d\n",zobudenie_dovod); break;
80     }
81 }
82 }
83
```

Obr.10 Časť kódu - Aktivovanie zariadenia z režimu spánku

## 6. Nastavíme komunikáciu.

Funkciou „void setup()“ vykonáme nastavenia. Príkazom „Serial.begin“ nastavíme rýchlosť seriálovej komunikácie 115200 bitov za sekundu pre prenos dát. Inicializujeme Wi-Fi a MQTT komunikáciu. Pripojením sa privolá funkcia „dovod\_zobudenia\_zariadenia“, ktorá nám vypíše prečo sa zariadenie zobudilo.

```
83
84 // ----- Nastavenie -----
85 void setup() {
86
87     Serial.begin(115200); // Nastavenie rýchlosti seriálovej komunikácie
88
89     pripojenie_wifi(); //Privolanie funkcie pripojenia sa wifi
90
91     mqtt_pripojenie(); // Privolanie funkcie pripojenia sa k MQTT serveru
92
93     //Print the wakeup reason for ESP32
94     dovod_zobudenia_zariadenia(); //Vypíše dôvod zobudenia
95
96     esp_sleep_enable_ext0_wakeup(GPIO_NUM_4, 0); //1 = High, 0 = Low
97
98     pinMode(schranka_senzor_GPIO_PIN, INPUT_PULLUP); // Nastavenie ESP32 pin vstupu na pull-up mode
99     status_schranky = digitalRead(schranka_senzor_GPIO_PIN); // Vyčíta stav GPIO pinu 4
100
101     if (status_schranky == LOW)
102     {
103         Serial.println("Prisla posta");
104         client.publish(topic_esp32, "1"); //Odošle na MQTT server správu o hodnote 1 na topic ESP32/Schranka
105         delay(5000); //Čaká 5 s na to aby sa zavrela schránka a neprišla duplicitná správa
106     }
107 }
108
109
110
111
112
113
114
```

Obr.11 Časť kódu - Nastavenia komunikácie.

## 7. Uložíme do spánku ESP32.

```
114 |  
115 | // ----- SLEEP -----  
116 |  
117 | Serial.println("Ide do hlbokého spanku");  
118 |  
119 | esp_deep_sleep_start(); //Začiatok hlbokého spánku  
120 |  
121 | }  
122 | void loop(){  
123 | }  
124 |
```

Obr.12 Časť kódu – Uloženie sa do režimu spánku.

Nahráme do ESP32 náš vytrovený program. Po nahraní programu sa ESP32 pripojí k Wi-Fi routeru, pričom mu router prideli lokálnu IP adresu. Mikrokontrolér sa pripojí k MQTT serveru, kde odošle správu o otvorení schránky. Následne sa uspí, aby šetril energiu z batérií. MQTT broker beží na Raspberry Pi. Raspberry Pi ako náš lokálny server odošle e-mailovú správu pomocou SMTP protokolu cez program Node-Red vždy, keď budú dvierka schránky otvorené. V konzole programu sa zobrazí výpis údajov zo seriálového portu. (Pre overenie sme použili domácu IP adresu)

```
16:29:16.155 -> Pripajanie k rotwik  
16:29:16.605 -> .....  
16:29:18.568 -> WiFi pripojene  
16:29:18.613 -> IP adresa zariadenia ESP32:  
16:29:18.613 -> 192.168.1.108  
16:29:18.613 -> Pripaajam sa k MQTT Serveru ...  
16:29:18.784 -> Pripojeny k MQTT  
16:29:18.784 -> Schranka sa otvorila  
16:29:18.784 -> Ide do hlbokého spanku
```

Obr.13 Výpis údajov zo seriálového portu v programe Arduino IDE.

## 3. MQTT server

Na Raspberry Pi vytvoríme MQTT server. Pred inštaláciou MQTT do Raspberry Pi si aktualizujeme operačný systém pomocou príkazov:

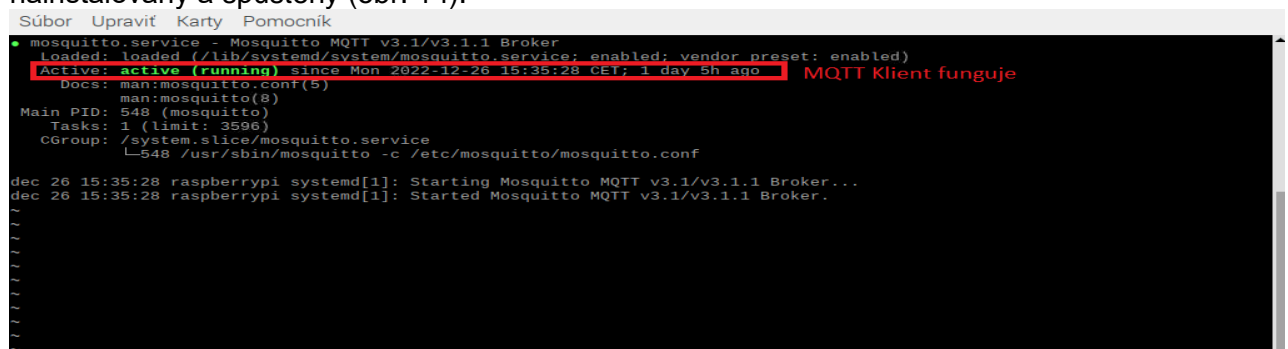
**sudo apt update**

**sudo apt upgrade**

Po dokončení aktualizácie systému nainštalujeme softvér Mosquitto pomocou príkazov:

**sudo apt install mosquitto mosquitto-clients**

Pomocou príkazu „**sudo systemctl status mosquitto**“ overíme, či je Mosquitto MQTT nainštalovaný a spustený (obr. 14).



```
Súbor  Upraviť  Karty  Pomocník  
● mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker  
   Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor preset: enabled)  
   Active: active (running) since Mon 2022-12-26 15:35:28 CET; 1 day 5h ago    MQTT Klient funguje  
     Docs: man:mosquitto.conf(5)  
    Main PID: 548 (mosquitto)  
      Tasks: 1 (limit: 3596)  
   CGroup: /system.slice/mosquitto.service  
           └─548 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf  
  
dec 26 15:35:28 raspberrypi systemd[1]: Starting Mosquitto MQTT v3.1/v3.1.1 Broker...  
dec 26 15:35:28 raspberrypi systemd[1]: Started Mosquitto MQTT v3.1/v3.1.1 Broker.
```

Obr.14 Overenie spustenia MQTT servera.

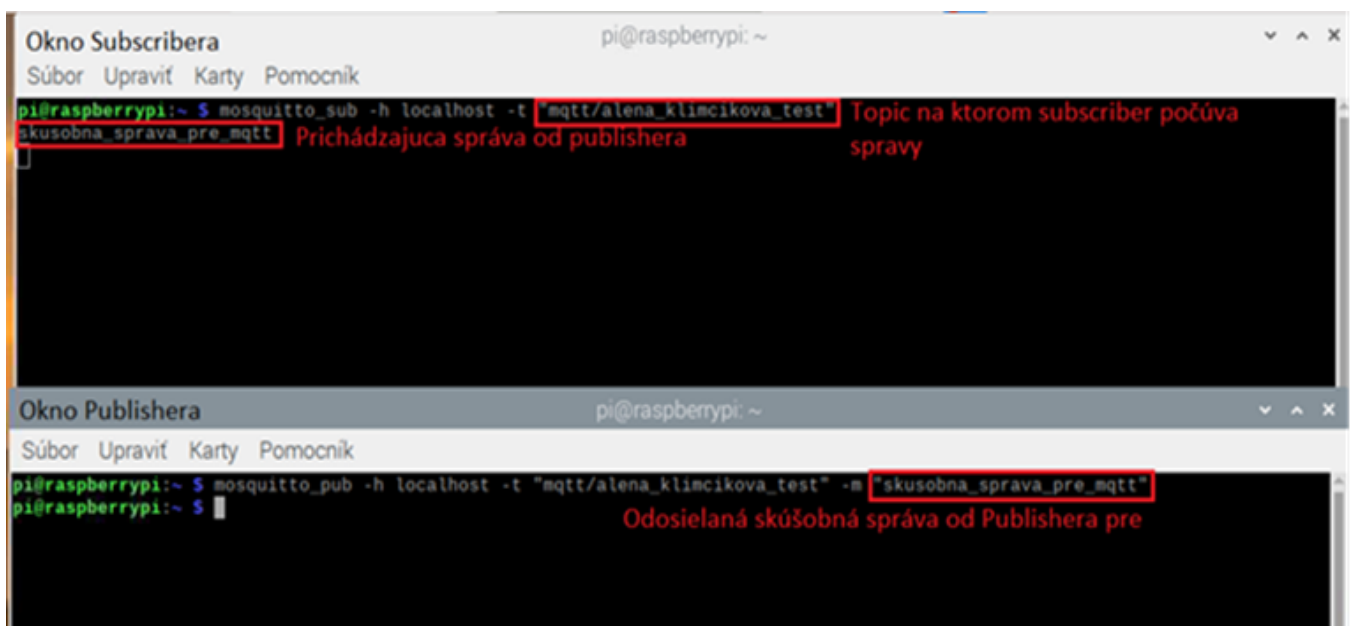


Pre overenie teraz začneme zdieľať správy so „Subscriberom“. Subscriber je to, čo bude počúvať nášho MQTT Brokera bežiaceho na Raspberry Pi.

Pripojíme sa k localhost pripojeniu príkazom „**mosquitto\_sub -h localhost -t „mqtt/alena\_klimcikova\_test“**“ a čakáme na správy od publishera z konkrétneho topicu „**mqtt/alena\_klimcikova\_test**“.

Teraz, keď máme klienta načítaného a počúvajúceho správy, skúsime mu jednu zverejniť. Spustíme nasledujúci príkaz „**mosquitto\_pub -h localhost -t "mqtt/alena\_klimcikova\_test" -m "skusobna\_sprava\_pre\_mqtt"** “.

zverejníme správu „**skusobna\_sprava\_pre\_mqtt**“ na našom localhost serveri pod topicom „**mqtt/alena\_klimcikova\_test**“. V prvom dialógovom okne sa nám zobrazí správa ktorú sme odoslali z druhého dialógového okna „**skusobna\_sprava\_pre\_mqtt**“ (obr. 15).



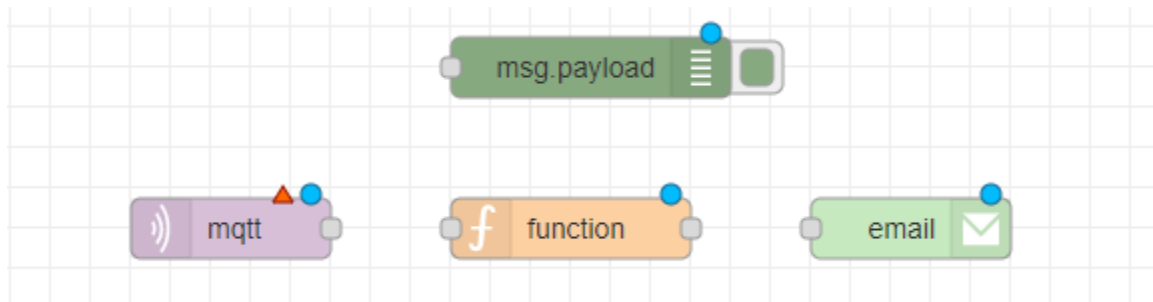
Obr.15 Overenie prijímania a odosielania správ cez MQTT.

## 4. Node-Red Server

Pre spustenie programu Node-Red zistíme lokálnu IP adresu servera (v našom prípade je to Raspberry PI), ktorý beží na porte 1880. Adresa nášho servera je <http://192.168.x.xxx:1880/> [1].

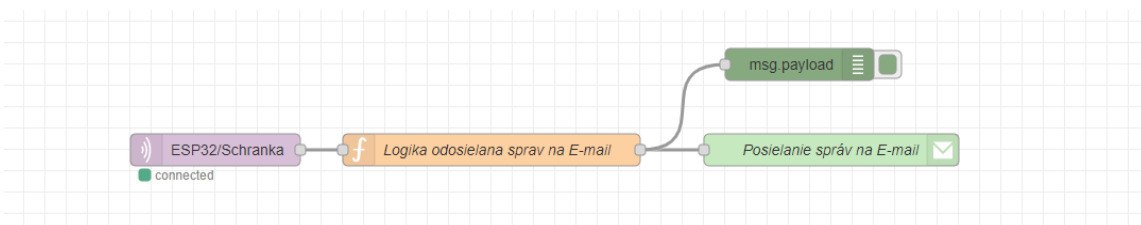
Po zadaní horeuvedenej IP adresy do nášho internetového prehliadača sa nám zobrazí programovacie prostredie Node-Red. Na vytvorenie odosielania správ do požadovaného e-mailu budeme potrebovať nasledujúce funkčné bloky (obr.16).

- Mqtt,
- Function,
- Email,
- Debug.



Obr.16 Použité funkčné bloky v Node-Red.

Ako vstup použijeme funkčný blok MQTT, ktorý si nájdeme pomocou vyhľadávacieho okna v ľavej hornej časti programovacieho prostredia. Následne tento blok presunieme na plochu programovacieho prostredia. Analogicky tak vyhľadáme a presunieme aj ostatné funkčné bloky, ktoré zapojíme podľa (obr. 17). Použité bloky si premenujeme.



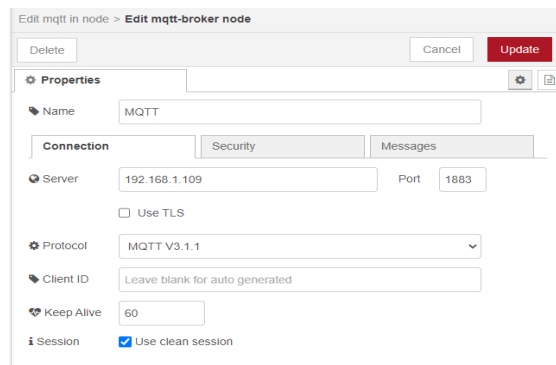
Obr.17 Schéma zapojenia funkčných blokov v programe Node-Red.

V ďalšom kroku budeme konfigurovať spomínané funkčné bloky. Ako prvý nastavíme MQTT. Po jeho rozkliknutí sa nám zobrazí dialógové okno v ktorom zadáme „Topic“, na ktorý nám bude publikovať ESP32 správy. Ostatné nastavenia ako QoS a Output necháme v základnom stave (obr. 18).

A screenshot of the 'Edit mqtt in node' dialog box in Node-Red. The dialog has a title bar 'Edit mqtt in node' and buttons for 'Delete', 'Cancel', and 'Done'. Below the buttons is a 'Properties' section with a settings icon. The properties are: 'Server' set to 'MQTT', 'Topic' set to 'ESP32/Schranka', 'QoS' set to '2', 'Output' set to 'auto-detect (string or buffer)', and 'Name' set to 'Name'. Each property has a corresponding icon and a dropdown or input field.

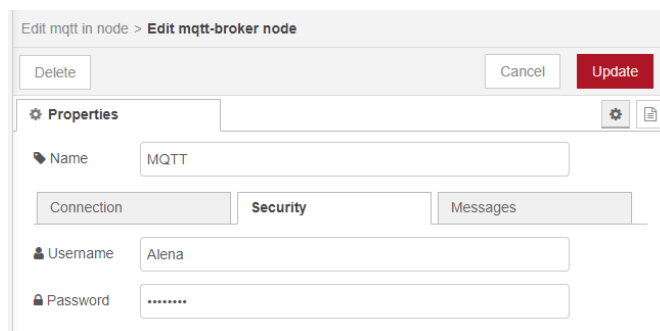
Obr.18 Konfigurácia funkčného bloku MQTT.

Nastavíme MQTT broker. V záložke „Connection“ nastavíme „Server“. V našom prípade to bude lokálna IP adresa Raspberry Pi na ktorom nám beží už spomínaný MQTT Client (obr.19).



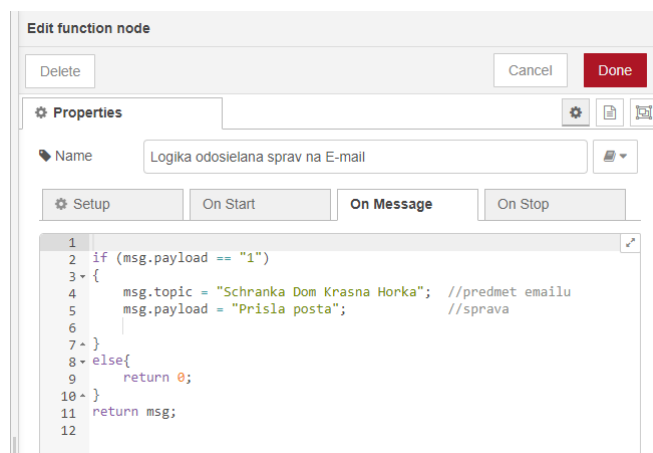
Obr.19 Nastavenie pripojenia funkčného bloku MQTT.

Posledným krokom je nastavenie zabezpečenia MQTT. Prejdeme do záložky „Security“, kde nastavíme používateľské meno a heslo (obr.20).



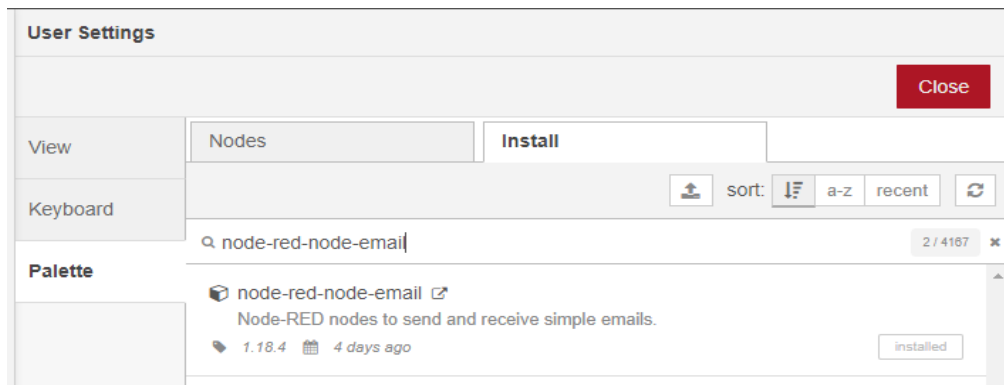
Obr.20 Nastavenie zabezpečenia MQTT.

Naprogramujeme funkčný blok funkcie podľa obr.21. Do funkcie vstupuje správa z MQTT bloku. Naša funkcia porovnáva vstupné hodnoty s požadovanou hodnotou „1“ a to pomocou príkazu „if“. To má za následok, že ak je podmienka splnená na výstupe funkcie sa nastaví „**msg.topic = Schranka Dom Krasna Horka**“ a „**msg.payload = Prisla posta**“, ktoré vstupujú do funkčného bloku e-mailu.



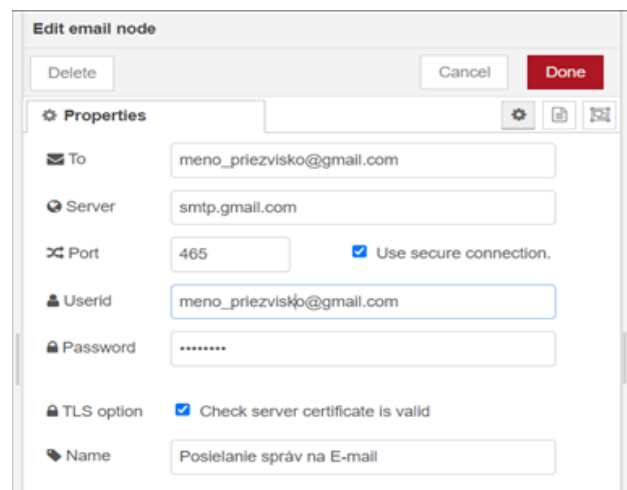
Obr. 21 Programovanie funkčného bloku function v programe Node-Red.

Posledný funkčný blok ktorý potrebujeme nastaviť je e-mail. Tento blok je nutné doinštalovať. Urobíme tak že klikneme v pravom hornom rohu na menu, kde vyberieme z ponuky možnosť „**Manage palette**“. Následne zvolíme záložku „**Install**“, kde vyhľadáme „**node-red-node-email**“, ktorý následne nainštalujeme (obr.22).



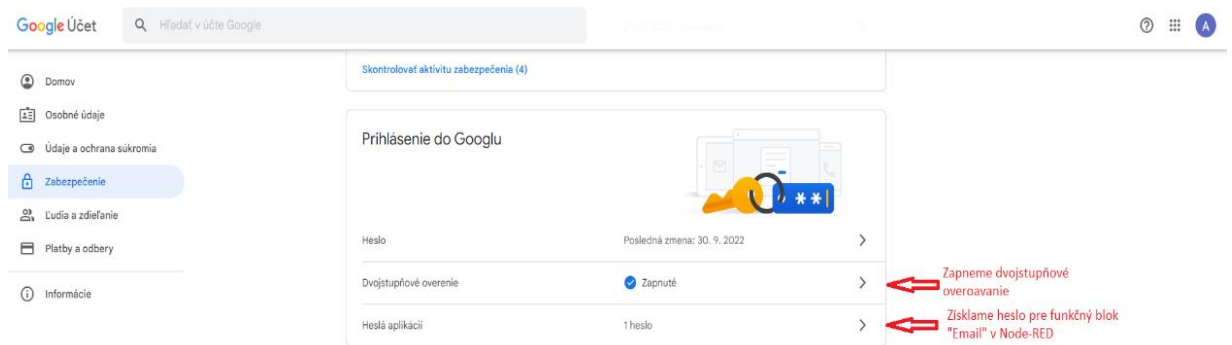
Obr. 22 Inštalácia node-red-node-email v programe Node-Red.

Po úspešnej inštalácii nakonfigurujem funkčný blok e-mailu. Po jeho rozkríknutí nastavíme odosielateľa a prijímateľa. Pre tento prípad budeme posilať sami sebe e-mail. Do položky „**To**“ zadáme prijímateľa správy. Položky „**Server**“ a „**Port**“ necháme v preddefinovanom stave. Nakoniec nastavíme e-mail odosielateľa „**Userid**“ s jeho heslom „**Password**“. Keďže sa jedná o menej bezpečne nastavenie je nutné v nastaveniach google e-mailu povoliť zdroje od menej zabezpečených zariadení ( obr.23).



Obr.23 Nastavenie funkčného bloku e-mailu v programe Node-Red.

Keďže sa jedná o menej bezpečne nastavenie je nutné v nastaveniach google e-mailu povoliť zdroje od menej zabezpečených zariadení. V nastavenia google e-mailu je potrebné povoliť dvojfaktorovú autentifikáciu (Obr 24)

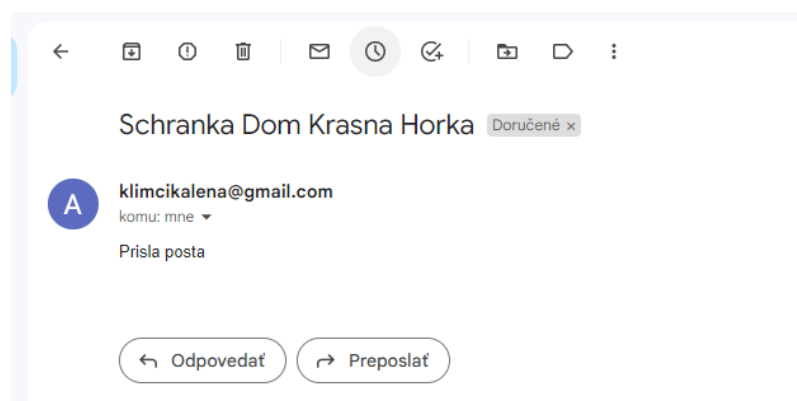


Obr 24. Nastavenie zabezpečenia

Vygenerujeme nové heslo ktoré použijeme vo funkčnom bloku „email“ a zadáme ho do „**Password**“ ( obr 25).

Ako posledný pripojíme funkčný blok „**debug**“ k funkcii. Debugger nám slúži na výpis dát z funkcie. Pre uloženie a aktiváciu všetkých zmien použijeme tlačidlo „**DEPLOY**“ .

Takto je program pripravený na používanie. Pred demonštráciou funkcionality pripojíme ESP32 k zdroju energie. Pri rozpojení magnetického kontaktu snímača bude doručená sprava na e-mail.



Obr.25 Doručenie správy o prijatí pošty na e-mail.

### **Použité zdroje:**

<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>

<https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/installing.html>

<https://nodered.org/docs/getting-started/raspberrypi>

<https://cookbook.nodered.org/mqtt/connect-to-broker>

<https://nodered.org/docs/user-guide/writing-functions>

<https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/>

<https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/>

<https://randomnerdtutorials.com/esp32-door-status-telegram/>

<https://randomnerdtutorials.com/esp32-cam-send-photos-email/#more-98272>

<https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/>

<https://www.hwlibre.com/sk/mqtt/>

[https://pimylifeup.com/raspberry-pi-mosquitto-mqtt-server/?fbclid=IwAR2BtbBds3QQYeEakRL0AigI36St4LJXdxNiyOd3DK7zSrY-nbZ\\_lg9tScc](https://pimylifeup.com/raspberry-pi-mosquitto-mqtt-server/?fbclid=IwAR2BtbBds3QQYeEakRL0AigI36St4LJXdxNiyOd3DK7zSrY-nbZ_lg9tScc)

[https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep\\_modes.html](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep_modes.html)

<https://support.google.com/accounts/answer/6010255?hl=en>

<https://uniot.sk/Downloads?sid=2>