Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.3
DOI: https://doi.org/10.24867/26IH02Babic

SISTEM ZA MERENJE, PRAĆENJE I UPRAVLJANJE STANJIMA U BAŠTI

A SYSTEM FOR MEASURING, MONITORING AND MANAGING WEATHER CONDITIONS IN THE GARDEN

Jelena Babić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – MEHATRONIKA

Kratak sadržaj – *U ovom radu je implementiran MQTT protokol između Mosquitto brokera, jednog Raspberry Pi 4B računara i četiri ESP32-D0WDQ6 mikrokontrolera. Cilj zadatka je bio da se pomoću podataka sa senzora o temperaturi i vlažnosti vazduha, vlažnosti zemljišta, količini padavina i intenzitetu svetlosti upravlja ventilom za vodu i zaklonom od Sunca, i da se dati podaci skladište u bazi podataka na centralnom serveru i da se u realnom vremenu prikazuju na veb stranici Grafana.*

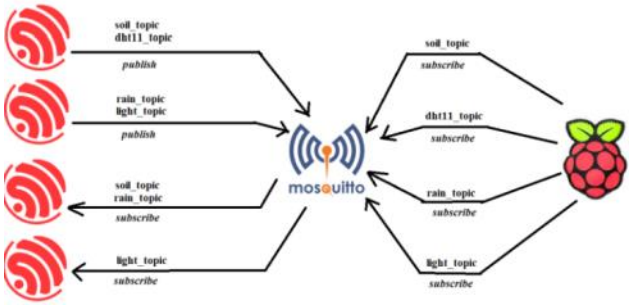
Ključne reči: *MQTT protkol, FreeRTOS OS u realnom vremenu, klijent server arhitektura, TCP/IP*

Abstract – *In this thesis, the MQTT protocol was implemented between the Mosquitto broker, one Raspberry Pi 4B single-board computer and four ESP32-D0WDQ6 microcontrollers. The goal was to control water valve and shade net based on the data from air temperature and humidity sensor, soil moisture sensor, rain drops sensor and light intesity sensor and to store these data in the database and displays it in real-time on the dashboard on the Grafana website.*

Keywords: *MQTT protocol, FreeRTOS real-time operating system, client server architecture, TCP/IP*

1. UVOD

U ovom radu je implementiran IoT sistem za privatnu baštu koji meri temperaturu i vlažnost vazduha, vlažnost zemljišta, količinu padavina i intenzitet svetlosti i na osnovu tih podataka upravlja ventilom za vodu i upravlja zaklonom od Sunca. Podaci koje daju ovi senzori se čuvaju na serveru koji se nalazi u kući, a u realnom vremenu se prate na kontrolnoj tabli kojoj se pristupa putem veb stranice. Na slici 1 data je blok šema celog sistema.



Slika 1 - Šema rada implementiranog IoT sistema

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bila dr Gordana Ostojić, red. prof.

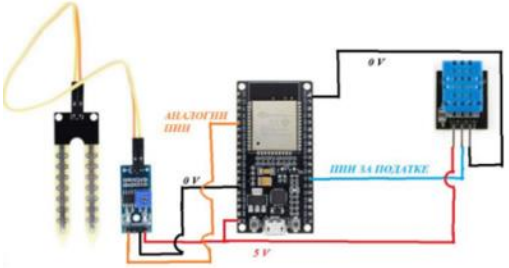
Na krajnjoj levoj strani mogu se videti četiri uređaja i oni se fizički nalaze u bašti. Predstavljaju, redom od gore prema dole, senzorski čvor za merenje temperature i vlažnosti vazduha i vlažnosti zemljišta, senzorski čvor za merenje količine padavina i intenziteta svetlosti, aktuatorski čvor za otvaranje i zatvaranje ventila za vodu i aktuatorski čvor za podešavanje zaklona od jakog sunca. Oni putem MQTT (eng. Message Queuing Telemetry Transport) protokola razmenjuju podatke i broker je prikazan u sredini. Sa desne strane se nalazi centralni softver koji dobija podatke putem istog protokola, obrađuje ih, skladišti i prikazuje na kontrolnoj tabli.

2. SENZORSKI I AKTUATORSKI ČVOROVI

Projekat se sastoji od četiri čvora koji međusobno komuniciraju putem MQTT protokola i u nastavku teksta će svaki biti posebno objašnjen.

2.1 Senzorski čvor za merenje temperature i vlažnosti vazduha i vlažnosti zemljišta

Senzorski čvor za merenje temperature i vlažnosti vazduha i vlažnosti zemljišta se sastoji od ESP32-WROOM-32 razvojne ploče, DHT11 senzora temperature i vlažnosti vazduha i HW-080 senzora vlažnosti zemljišta. DHT11 senzor u sebi ima NTC termistor koji je zapravo otpornik sa negativnim temperaturnim koeficijentom, što znači da kad temperatura raste, otpornost opada. Može da meri temperaturu od 0°C do 50°C i vlažnost od 20% do 90% sa preciznošću od ±1°C i ±1%. HW-080 senzor radi po principu merenja otpornosti i to tako što se dve sonde zabodu u zemljište i struja prolazi kroz njega. Na slici 2 je prikazana šema povezivanja ovog čvora.



Slika 2 - Šema povezivanja senzorskog čvora za merenje temperature i vlažnosti vazduha i vlažnosti zemljišta

Pošto su povezani na jednu razvojnu pločicu, u softveru je nameštено da merenje svakog senzora bude predstavljeno kao različiti zadatak. S obzirom da ESP32 mikrokontroler ima dva jezgra, taskovi mogu da se izvršavaju konkurentno na jednom jezgru ili paralelno na oba. U

ovom slučaju, pošto se koristi FreeRTOS, izabrano je konkurentno programiranje što može da se vidi sa slike 3.

```
void app_main(void)
{
    //Initialize NVS
    esp_err_t ret = nvs_flash_init();
    if (ret == ESP_ERR_NVS_NO_FREE_PAGES || ret == ESP_ERR_NVS_NEW_VERSION_FOUND) {
      ESP_ERROR_CHECK(nvs_flash_erase());
      ret = nvs_flash_init();
    }
    ESP_ERROR_CHECK(ret);

    wifi_init_sta();

    client = mqtt_app_start();

    xTaskCreate(vTask_Soil, "Task_Soil", 2048, NULL, 1, NULL);
    xTaskCreate(vTask_Dht11, "Task_Dht11", 2048, NULL, 1, NULL);
}
```

Slika 3 - Main funkcija senzorskog čvora za merenje temperature i vlažnosti vazduha i vlažnosti zemljišta

Sama implementacija MQTT biblioteke i korisničkih funkcija nije prikazana i može da se pogleda u softveru koji je dostupan uz ovaj rad. Zadatak u kojem je obrađeno merenje temperature i vlažnosti vazduha koristi gotovu biblioteku DHT11 i objavljuje podatke na „dht11_topic" temu. Ovaj zadatak prikazan je na slici 4.

```
void vTask_Dht11()
{
    // Configure DHT11 sensor
    DHT11_init(4);

    while(1)
        {
        printf("\n");
        temperature = DHT11_read().temperature;
        humidity = DHT11_read().humidity;
        printf("Temperature is %d \n", DHT11_read().temperature);
        printf("Humidity is %d\n", DHT11_read().humidity);
        printf("Status code is %d\n", DHT11_read().status);
        printf("\n");

        cJSON *json_root = cJSON_CreateObject();
        cJSON_AddNumberToObject(json_root, "temperature", temperature);
        cJSON_AddNumberToObject(json_root, "humidity", humidity);
        char *json_string = cJSON_Print(json_root);

        esp_mqtt_client_publish(client, "dht11_topic", json_string, 0, 1, 0);

        // Clean up cJSON objects and the JSON string when done
        cJSON_Delete(json_root);
        free(json_string);

        vTaskDelay(5 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
        }
    vTaskDelete(NULL);
}
```

Slika 4 - Zadatak za merenje temperature i vlažnosti vazduha

Zadatak za merenje vlažnosti zemljišta je dat na slici 5.

```
void vTask_Soil()
{
    // Configure ADC
        adc1_config_width(ADC_WIDTH_BIT_12);
        adc1_config_channel_atten(SOIL_SENSOR_ANALOG_CHANNEL, ADC_ATTEN_DB_11);

    while(1)
    {
        printf("\n");

        // Read the analog value from the light sensor
        soil_adc_value  = 0;
        for (int i = 0; i < 32; i++)
        {
            soil_adc_value  += adc1_get_raw(SOIL_SENSOR_ANALOG_CHANNEL);
            vTaskDelay(pdMS_TO_TICKS(10));
        }
        soil_adc_value  /= 32;
        soil_adc_value = 4095 - soil_adc_value;
        soil_adc_percentage = (100 * soil_adc_value) / 4096;

        // You can add code here to convert the analog value to brightness level
        if (soil_adc_percentage <= 15)
            soil_moisture_level = "It's dry!";
        else if (soil_adc_percentage <= 30)
            soil_moisture_level = "It's normal!";
        else if (soil_adc_percentage <= 50)
            soil_moisture_level = "It's wet!";
        else if (soil_adc_percentage <= 80)
            soil_moisture_level = "It's very wet!";
        else
            soil_moisture_level = "It's floaded!";

        printf(soil_moisture_level, "\n");

        cJSON *json_root = cJSON_CreateObject();
        cJSON_AddStringToObject(json_root, "soil moisture level", soil_moisture_level);
        cJSON_AddNumberToObject(json_root, "soil_adc_percentage", soil_adc_percentage);
        char *json_string = cJSON_Print(json_root);

        esp_mqtt_client_publish(client, "soil_topic", json_string, 0, 1, 0);

        // Clean up cJSON objects and the JSON string when done
        cJSON_Delete(json_root);
        free(json_string);

        vTaskDelay(5 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
    }
    vTaskDelete(NULL);
}
```

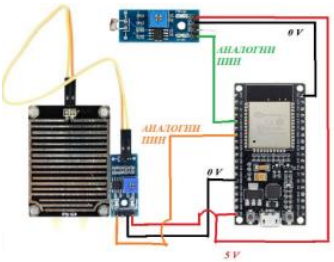
Slika 5 – Zadatak za merenje vlažnosti zemljišta

Zadatak u kojem je obrađeno merenje vlažnosti zemljišta, meri vrednost vlažnosti zemljišta na AD (eng. Analog to Digital Converter) konvertoru sa metodom sukcesivnih aproksimacija. Rezolucija AD konvertora je 12-bitna i na taj način se dobija integer vrednost od 0 do 4095 i zatim se, radi lakšeg prikazivanja, skalira na vrednost u procentima u iznosu od 0 do 100.

U slučaju da je vlažnost zemljišta manja ili jednaka 15%, smatra se da je zemljište suvo. Ako je veća od 15% i manje ili jednako 30% smatra se da je zemljište zadovoljavajuće vlažnosti. Ako je vlažnost zemljišta veća od 30%, a manja ili jednaka od 50% ono je vlažno, dok je za granicu između 51% i 80% veoma vlažno, a preko toga je zemljište poplavljeno. Podatke o vlažnosti zemljišta izraženim u procentima, kao i gore iznete komentare za svaku granicu, zadatak objavljuje na „soil_topic" temu.

2.2 Senzorski čvor za merenje količine padavina i intenziteta svetlosti

Senzorski čvor za merenje količine padavina i intenziteta svetlosti se sastoji od ESP32-WROOM-32 razvojne ploče, HW-028 senzora količine padavina i senzora intenziteta svetlosti. Na sledećoj slici 6 je prikazana šema povezivanja ovog čvora.



Slika 6 – Šema povezivanja senzorskog čvora za merenje količine padavina I intenziteta svetlosti

HW-028 senzor radi po principu promenjivog otpornika, odnosno u zavisnosti od količine vode na njemu, menja se otpornost. Senzor intenziteta svetlosti radi po principu merenja otpornosti jer sadrži fotorezistor.

Zadatak koji meri količinu padavina objavljuje na "rain_topic" temu, kao i kakva je kiša i koliko je padavina. Prikazan je na slici 7.

```
void vTask_Rain()
{
    // Configure ADC
    adc1_config_width(ADC_WIDTH_BIT_12);
    adc1_config_channel_atten(RAIN_SENSOR_ANALOG_CHANNEL, ADC_ATTEN_DB_11);

    while(1)
    {
        printf("\n");

        // Read the analog value from the rain sensor
        rain_adc_value  = 0;
        for (int i = 0; i < 32; i++)
        {
            rain_adc_value  += adc1_get_raw(RAIN_SENSOR_ANALOG_CHANNEL);
            vTaskDelay(pdMS_TO_TICKS(10));
        }
        rain_adc_value  /= 32;
        rain_adc_value = 4095 - rain_adc_value;
        rain_adc_percentage = (100 * rain_adc_value) / 4096;

        if (rain_adc_percentage <= 10)
            rainfall = "No rain!";
        else if (rain_adc_percentage <= 30)
            rainfall = "Light rain";
        else if (rain_adc_percentage <= 60)
            rainfall = "Moderate rain";
        else if (rain_adc_percentage <= 80)
            rainfall = "Heavy rain";
        else
            rainfall = "Very heavy rain!";
        printf(rainfall, "\n");

        cJSON *json_root = cJSON_CreateObject();
        cJSON_AddStringToObject(json_root, "rainfall", rainfall);
        cJSON_AddNumberToObject(json_root, "rain_adc_percentage", rain_adc_percentage);
        char *json_string = cJSON_Print(json_root);

        esp_mqtt_client_publish(client, "rain_topic", json_string, 0, 1, 0);

        // Clean up cJSON objects and the JSON string when done
        cJSON_Delete(json_root);
        free(json_string);

        vTaskDelay(5 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);

    }
    vTaskDelete(NULL);
}
```

Slika 7 – Zadatak za merenje količine padavina

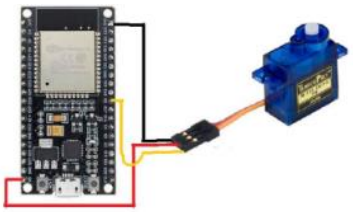
Zadatak koji meri intenzitet svetlosti objavljuje na “light_topic” temu koliki je intenzitet i kakva je jačina svetlosti. Zadatak je prikazan na slici 8.

```
void vTask_Light()
{
    // Configure ADC
        adc1_config_width(ADC_WIDTH_BIT_12);
        adc1_config_channel_atten(LIGHT_SENSOR_ANALOG_CHANNEL, ADC_ATTEN_DB_11);

    while(1)
    {
        printf("\n");

        // Read the analog value from the light sensor
        light_adc_value  = 0;
        for (int i = 0; i < 32; i++)
        {
            light_adc_value  += adc1_get_raw(LIGHT_SENSOR_ANALOG_CHANNEL);
            vTaskDelay(pdMS_TO_TICKS(10));
        }
        light_adc_value  /= 32;
        light_adc_value = 4095 - light_adc_value;
        light_adc_percentage = (100 * light_adc_value) / 4096;

        // You can add code here to convert the analog value to brightness level
        if (light_adc_percentage <= 5)
            brightness = "It's dark!";
        else if (light_adc_percentage <= 20)
            brightness = "It's dim!";
        else if (light_adc_percentage <= 50)
            brightness = "It's light!";
        else if (light_adc_percentage <= 80)
            brightness = "It's bright!";
        else
            brightness = "It's very bright!";

        printf(brightness, "\n");

        cJSON *json_root = cJSON_CreateObject();
        cJSON_AddStringToObject(json_root, "brightness", brightness);
        cJSON_AddNumberToObject(json_root, "light_adc_percentage", light_adc_percentage);
        char *json_string = cJSON_Print(json_root);

        esp_mqtt_client_publish(client, "light_topic", json_string, 0, 1, 0);

        // Clean up cJSON objects and the JSON string when done
        cJSON_Delete(json_root);
        free(json_string);

        vTaskDelay(5 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
    }
    vTaskDelete(NULL);
}
```

Slika 8 – Zadatak za merenje intenziteta svetlosti

2.3 Aktuatorski čvor za upravljanje navodnjavanjem

Aktuatorski čvorovi se sastoje od ESP32-WROOM-32 razvojne ploče i SG90s servo motora. S obzirom da su hardverski identični, na slici 9 je prikazana šema povezivanja oba aktuatorska čvora na jednom primeru.



Slika 9 – Šema povezivanja aktuatorskog čvora

Servo motor, pored žica za napajanje ima i ulaznu žicu za upravljanje odnosno PWM (eng. Pulse Width Modulation) žicu. PWM signal treba da ima frekvenciju od 50 Hz odnosno period od 20 ms. Da bi izlazno vratilo bilo u položaju koji predstavlja položaj od 0° potrebno je da period bude podešen na 1 ms, za položaj od 90° potrebno je da period bude podešen na 1,5 ms i za položaj od 180° potrebno je da period bude podešen na 2 ms.

Aktuatorski čvor za upravljanje navodnjavanjem je pretplaćen na „soil_topic“ i „rain_topic“ teme i u slučaju da nema kiše i da je zemljište suvo, zakreće vratilo za ugao od 90° i na taj način otvara ventil od pumpe koja navodnjava baštu vodom. U slučaju da pada kiša ili da je zemljište dovoljno vlažno, motor se vraća u prvobitni položaj od 0° i na taj način isključuje ventil.

Za ovaj aktuatorski čvor su implementirana dva zadatka, gde jedan upravlja servo motorom i prikazan je na slici 10, a drugi se pretplaćuje na gore navedene teme.

```
void vTask_Servo()
{
    // Set the LEDC peripheral configuration
        example_ledc_init();
        // Set duty to 5%
        ESP_ERROR_CHECK(ledc_set_duty(LEDC_MODE, LEDC_CHANNEL, LEDC_DUTY_0));
        // Update duty to apply the new value
        ESP_ERROR_CHECK(ledc_update_duty(LEDC_MODE, LEDC_CHANNEL));
    vTaskDelay(5 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);

    while(1)
    {
            if (valveOpen == false)
            {
                if (soil_int_value <= 25 && rain_int_value <= 10)
                {
                    // Set duty to 7.5%
                            ESP_ERROR_CHECK(ledc_set_duty(LEDC_MODE, LEDC_CHANNEL, LEDC_DUTY_90));
                            // Update duty to apply the new value
                            ESP_ERROR_CHECK(ledc_update_duty(LEDC_MODE, LEDC_CHANNEL));
                    valveOpen = true;
                    printf("opening valve \n");
                    vTaskDelay(2 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
                }
            }
            else
            {
                if (soil_int_value > 25 || rain_int_value > 10)
                {
                    // Set duty to 5%
                            ESP_ERROR_CHECK(ledc_set_duty(LEDC_MODE, LEDC_CHANNEL, LEDC_DUTY_0));
                             // Update duty to apply the new value
                             ESP_ERROR_CHECK(ledc_update_duty(LEDC_MODE, LEDC_CHANNEL));
                    valveOpen = false;
                    printf("closing valve \n");
                    vTaskDelay(2 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
                }
            }
        vTaskDelay(5 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
    }
    vTaskDelete(NULL);
}
```

Slika 10 – Zadatak za upravljanje servo motora koji otvara i zatvara ventil za vodu

2.4 Aktuatorski čvor za upravljanje zaštitnom mrežom od direktnih sunčevih zraka

Aktuatorski čvor za upravljanje zaštitnom mrežom od sunca je pretplaćen na „light_topic“ temu i u slučaju da je veoma velik intenzitet svetlosti, zakreće vratilo za ugao od 180° i na taj način prekriva gušće mrežom baštu. U slučaju da je velik intenzitet svetlosti, aktuatorski čvor zakreće vratilo za ugao od 90° i na taj način prekriva srednje gusto mrežom baštu, dok u slučaju da je normalan intenzitet svetlosti, zakreće vratilo za ugao od 0° i na taj način otkriva baštu. Mreža treba da bude napravljena tako da su joj delovi urađeni pod određenim uglom kako bi propuštala kišu, a smanjila intezitet svetlosti.

Za ovaj aktuatorski čvor su implementirana dva zadatka, gde jedan upravlja servo motorom, a drugi se pretplaćuje na „light_topic“ temu. Zadatak koji upravlja servo motorom, odnosno zaštitnom mrežom je dat na slici 11.

```
void vTask_Servo()
{
    // Set the LEDC peripheral configuration
        example_ledc_init();
        // Set duty to 5%
        ESP_ERROR_CHECK(ledc_set_duty(LEDC_MODE, LEDC_CHANNEL, LEDC_DUTY_0));
        // Update duty to apply the new value
        ESP_ERROR_CHECK(ledc_update_duty(LEDC_MODE, LEDC_CHANNEL));
    vTaskDelay(5 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);

    while(1)
    {
            if ((suncobran180 == false) && (light_int_value > 80))
            {
                // Set duty to 10%
                        ESP_ERROR_CHECK(ledc_set_duty(LEDC_MODE, LEDC_CHANNEL, LEDC_DUTY_180));
                        // Update duty to apply the new value
                        ESP_ERROR_CHECK(ledc_update_duty(LEDC_MODE, LEDC_CHANNEL));
                suncobran180 = true;
                suncobran90 = false;
                printf("180 stepeni \n");
                vTaskDelay(2 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
            }
            else if ((suncobran90 == false) && (light_int_value > 50) && (light_int_value <= 80))
            {
                // Set duty to 7.5%
                ESP_ERROR_CHECK(ledc_set_duty(LEDC_MODE, LEDC_CHANNEL, LEDC_DUTY_90));
                // Update duty to apply the new value
                ESP_ERROR_CHECK(ledc_update_duty(LEDC_MODE, LEDC_CHANNEL));
                suncobran180 = false;
                suncobran90 = true;
                printf("90 stepeni \n");
                vTaskDelay(2 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
            }
            else if (((suncobran90 == true) || (suncobran90 == true)) && (light_int_value <= 50))
            {
                // Set duty to 5%
                ESP_ERROR_CHECK(ledc_set_duty(LEDC_MODE, LEDC_CHANNEL, LEDC_DUTY_0));
                // Update duty to apply the new value
                ESP_ERROR_CHECK(ledc_update_duty(LEDC_MODE, LEDC_CHANNEL));
                suncobran180 = false;
                suncobran90 = false;
                printf("0 stepeni \n");
                vTaskDelay(2 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
            }
        vTaskDelay(5 * CONFIG_BLINK_PERIOD / portTICK_PERIOD_MS);
    }
    vTaskDelete(NULL);
}
```

Slika 11 – Zadatak koji upravlja servo motorom za nameštanje zaštitne mreže

2.3 Mikrokontroler

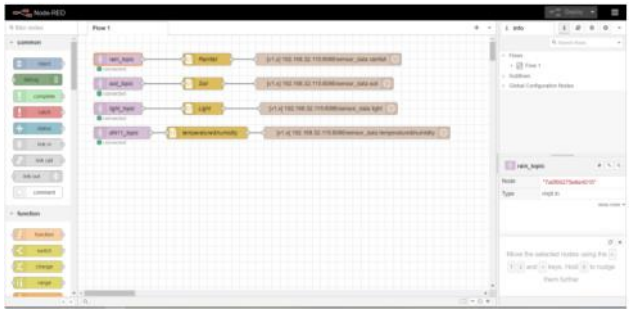
ESP32-WROOM-32 38-pinske razvojna ploča sadrži sadrži ESP32-D0WDQ6 mikrokontroler koji ima dva Xtensa® 32-bitni LX6 mikroprocesora, 520KB SRAM i 4MB Flash memorije.

2.4 FreeRTOS

FreeRTOS je operativni sistem u realnom vremenu koji pruža podršku za konkurentno programiranje putem zadataka gde svaki ima svoj prioritet, kod i prostor za stek. Svaki zadatak pruža nezavisni tok izvršavanja unutar aplikacije. Scheduler je baziran po principu prioriteta i preemptive je. Postoje mehanizmi za sinhronizaciju između zadataka kao što su redovi poruka, semafori, muteksi i grupe događaja [1].

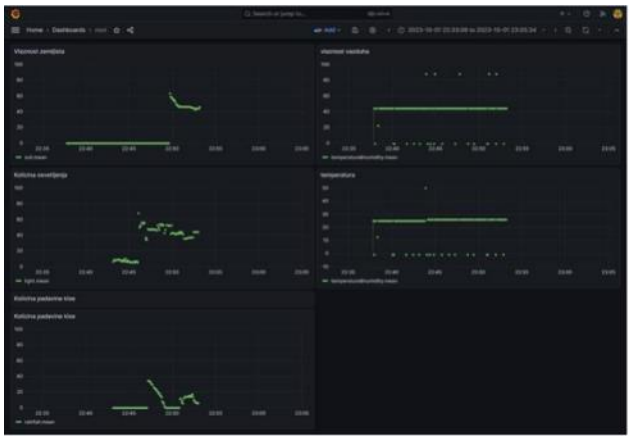
3. CENTRALNI SERVER

Raspberry Pi single-board računar služi kao centralni server. Zahvaljujući Docker-u, na njemu se nalazi kontejner sa NodeRED koji služi da se pretplati na sve teme MQTT brokera i da dobijene podatke obradi i pošalje u bazu podataka. Node-RED je open-source razvojni alat za vizualno programiranje i prikazan je na slici 12.



Slika 12 – Node-RED korisnički interfejs

U drugom kontejneru se nalazi InfluxDB baza podataka u kojoj se skladište podaci sa senzora. Ovoj bazi pristupa Grafana koja se nalazi u trećem kontejneru i prikazuje podatke u realnom vremenu na kontrolnoj tabli kojoj se pristupa preko veb stranice. Na slici 13 prikazana je kontrolna tabla.



Slika 13 – Granafa korisnički interfejs

Grafana je open-source softver koji se koristi za vizualizaciju, praćenje i analizu podataka iz različitih izvora u vidu interaktivnih grafikona i tabela [2].

4. PROTOKOLI IMPLEMENTIRANI U ZADATKU

Za razmenu podataka sa senzora izabran je MQTT protokol koji predstavlja protokol na aplikativnom sloju TCP/IP (eng. Transmission Control Protocol/Internet Protocol) modela. TCP/IP je skup standardizovanih protokola koji omogućavaju da uređaji komuniciraju preko mreže kao što je internet [3].

MQTT protokol je jednostavan protokol aplikativnog sloja namenjen za prenos kratkih poruka i baziran je na sistemu pretplate i objave. MQTT je klijent-server protokol koji omogućava razmenu poruka između klijenata preko centralnog servera koji se naziva MQTT broker. MQTT broker prima, rutira i dostavlja poruke. Klijenti mogu biti pretplatnici (eng. subscribers) ili objavljivači (eng. publishers), s tim da se prvi pretplaćuju na teme (eng. topics), a drugi objavljuju teme [4].

Senzorski čvorovi objavljuju podatke na teme, a aktuatorski čvorovi i centralni server su pretplaćeni na teme. Za broker je izabran Eclipse Mosquitto broker i nalazi se na računaru koji je u kući

5. ZAKLJUČAK

Konačni zaključak je da se okruženjem, odnosno stanjima u bašti, lako može upravljati i pratiti korišćenjem MQTT protokola i odgovarajućih komponenti.

Za pravce daljeg istraživanja, treba ispitati slučaj kada ima veći broj konektovanih klijenata na MQTT broker i kako to utiče na brzinu odziva brokera i pouzdanost sistema. Ovo bi podrazumevalo dodavanje senzora kao što su senzor za pritisak vazduha, ultrazvučni senzor za praćenje jačine vetra, senzor za detekciju dima, senzor za detekciju plamena u slučaju požara itd. Na osnovu podataka koje bi dali dodatni senzori, mogli bi se koristiti aktuatori za upravljanje zaštitnim platnom od vetra, upravljanje prskalicama protiv požara itd.

Svaki klijent koji je konektovan koristi memoriju i resurse brokerove centralne procesorske jedinice. Ako je više klijenata u realnom vremenu konektovano na broker, mogu da se zagube poruke ili da kasne. Više klijenata dovodi do preopterećenja u komunikaciji.

6. LITERATURA

[1] https://www.freertos.org

[2] https://petri.com/what-is-grafana/

[3] Douglas E. Come, „Internetworking with TCP/IP“, 2000

[4] https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

Kratka biografija:

Jelena Babić rođena je u Sremskoj Mitrovici 1997. god. Master rad na Fakultetu tehničkih nauka iz oblasti Mehatronika – Robotika i automatizacija odbranila je 2023.god.
kontakt: jelenababic1997@gmail.com