VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

ELEKTRONIKOS FAKULTETAS

ELEKTRONINIŲ SISTEMŲ KATEDRA

IŠMANIŲJŲ ĮRENGINIŲ PROGRAMAVIMO KURSINIS PROJEKTAS

Išmanieji IP įrenginiai

Kursinis darbas

Atliko: DKRf-22 gr. studentas Eimantas Dima

Tikrino: Dr. Tomas Cuzanauskas

VILNIUS 2025

ĮVADAS

Kursinio darbo tikslas – praktiškai įsisavinti Python programavimo kalbą, jos „Flask“ karkasą ir „paho-mqtt“ biblioteką, sujungiant du skirtingus pasaulius: REST interneto paslaugas ir lengvojo mašininio ryšio (MQTT) protokolą, dažnai naudojamą daiktų interneto (IoT) sprendimuose. Sukurta demonstracinė aplikacija leidžia:

• per naršyklę ieškoti angliškų žodžių reikšmių viešoje „Dictionary API“;

• gautą atsaką išsaugoti JSON faile ir publikuoti į MQTT temą;

• realiu laiku naršyklėje stebėti iš brokerio gaunamas MQTT žinutes.

Taip pademonstruotas web → IoT → web informacijos srautas bei įgytas praktinis darbo su HTTP užklausomis, WebSocket-iniu MQTT ryšiu ir asinchroniniu programavimu patyrimas.

PROGRAMINIO KODO TRUMPAS APRAŠYMAS

1. **2. Programinio kodo trumpas aprašymas**
2. **2.1 Architektūros apžvalga**

Programa realizuota kaip monolitinė Flask aplikacija su integruotais MQTT ir API klientais. Sistemos architektūra susideda iš šių pagrindinių komponentų:

Pagrindinės bibliotekos:

- Flask - web aplikacijos framework
- requests - HTTP užklausoms į Dictionary API
- paho-mqtt - MQTT protokolo implementacija
- json - JSON duomenų apdorojimui
- threading - asinchroniniam MQTT operacijų apdorojimui

3. **2.2 Funkcionalumo analizė**

Dictionary API integracija: Funkcija lookup_word_api(word) realizuoja API užklausas į https://api.dictionaryapi.dev/ paslaugą. Funkcija įgyvendina:

- HTTP GET užklausų siuntimą su timeout parametru
- Klaidų valdymą (404, connection errors)

- JSON atsakymų apdorojimą

MQTT funkcionalumas: Sistema implementuoja abu MQTT vaidmenis:

- **Publisher**: Siunčia Dictionary API duomenis į dictionary/words temą
- **Subscriber**: Kluso įeinančias žinutes ir jas saugo mqtt_messages sąraše

MQTT klientas konfigūruojamas setup_mqtt_client() funkcijoje su callback funkcijomis:

- on_connect() - prisijungimo valdymui
- on_message() - žinučių apdorojimui

Duomenų saugojimas: save_to_json_file() funkcija sukuria struktūrizuotą JSON objektą su:

- Laiko žyma (timestamp)
- Šaltinio informacija
- Kliento metaduomenimis
- Pagrindiniais API duomenimis

4. 2.3 Web sąsajos implementacija

Flask aplikacija pateikia šiuos maršrutus (endpoints):

Maršrutas	Metodas	Aprašymas
/	GET	Pagrindinis puslapis su HTML sąsaja
/lookup/<word>	GET	Žodžio paieška ir MQTT siuntimas
/api	GET	JSON failo atsisiuntimas
/mqtt_messages_json	GET	MQTT žinučių JSON gavimas
/test_mqtt	GET	MQTT funkcionalumo testavimas

Frontend sprendimas: HTML šablonas HTML_TEMPLATE integruoja:

- JavaScript fetch API realaus laiko duomenų gavimui
- Automatinį žinučių atnaujinimą kas 3 sekundes
- Responsyvų dizainą su CSS stiliais
- Interaktyvius elementus žodžių paieškai

5. 2.4 Duomenų srautų analizė

Tipinis naudojimo scenarijus:

1. Vartotojas įveda žodį web sąsajoje
2. Sistema siunčia užklausą į Dictionary API
3. Gauti duomenys apdorojami ir saugomi JSON faile
4. Duomenys siunčiami į MQTT brokerį
5. MQTT subscriber gauna žinutę ir atnaujina sąsają

6. Vartotojas mato rezultatus realiu laiku

Duomenų transformacija:

Dictionary API → JSON processing → File storage → MQTT publish → MQTT receive → Web display

6. 2.5 Klaidų valdymas ir stabilumas

Sistema implementuoja kelių lygių klaidų valdymą:

- **Network timeouts**: 10 sekundžių timeout API užklausoms
- **MQTT connection errors**: Graceful handling su error reporting
- **File I/O errors**: Exception catching su informaciniais pranešimais
- **JSON parsing errors**: Try-catch blokų naudojimas

Išteklių valdymas: cleanup() funkcija užtikrina korektišką MQTT klientų atjungimą programa pasibaigus.

IŠVADOS

Projekte pavyko sujungti dvi fundamentaliai skirtingas komunikacijos technologijas – sinchroninį REST ir asinchroninį MQTT – į bendrą, vartotojui draugišką aplikaciją. Darbe:

• įsisavintas Flask maršrutų kūrimas, HTML ir JS integracija;

• sukonfigūruotas MQTT klientas, gebantis vienu metu publikuoti ir prenumeruoti pranešimus;

• realizuotas minimalus log failas (api.json) ir bazinis pranešimų buferis;

• parodyta, kaip išoriniams įrenginiams (pvz., sensoriams) būtų galima siųsti arba gauti duomenis per tą pačią temą.