Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №4

з дисципліни «Технології розробки вбудованих систем IoT»

Виконав:
студент групи ІС-12
Канупа Максим

Перевірив:
Каплунов А. В.

Київ – 2025

1) Git repo: https://github.com/Aristocrab/Labs

2) Реалізація AgentMqttAdapter, data_processing

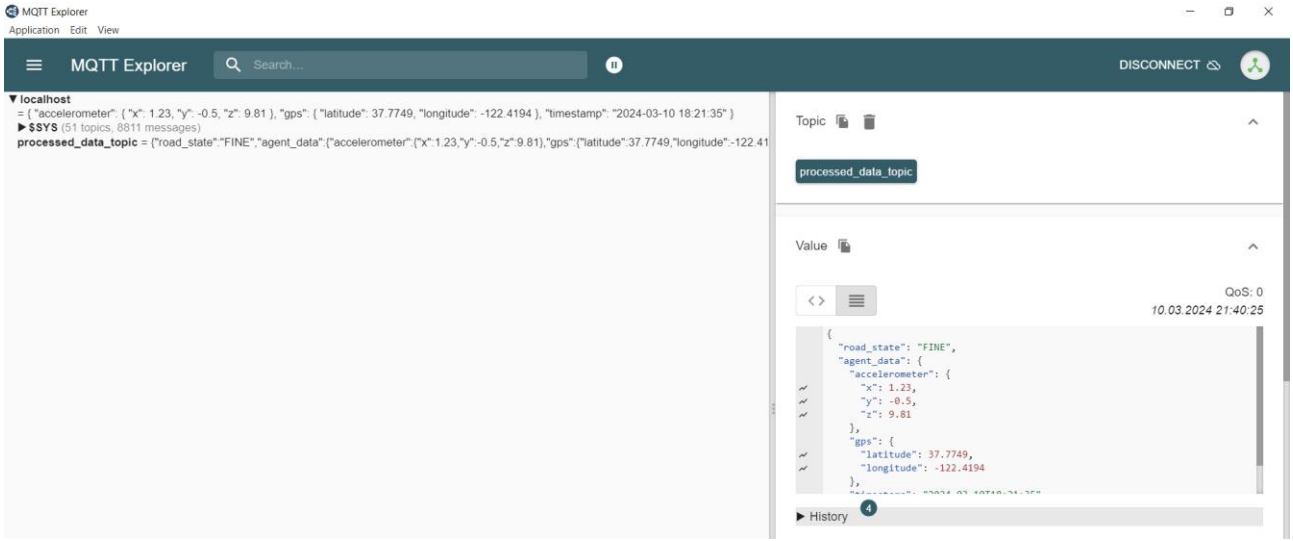
```
class AgentMQTTAdapter(AgentGateway):
    def __init__(
        self,
        broker_host,
        broker_port,
        topic,
        hub_gateway: HubGateway,
        batch_size=10,
    ):
        self.batch_size = batch_size
        # MQTT
        self.broker_host = broker_host
        self.broker_port = broker_port
        self.topic = topic
        self.client = mqtt.Client()
        # Hub
        self.hub_gateway = hub_gateway

    def on_connect(self, client, userdata, flags, rc):
        if rc == 0:
            logging.info("Connected to MQTT broker")
            self.client.subscribe(self.topic)
        else:
            logging.info(f"Failed to connect to MQTT broker
with code: {rc}")

    def on_message(self, client, userdata, msg):
        """Processing agent data and sent it to hub
gateway"""
        try:
            payload: str = msg.payload.decode("utf-8")
            # Create AgentData instance with the received
data
            agent_data =
AgentData.model_validate_json(payload, strict=True)
            # # Process the received data (you can call a use
case here if needed)
            processed_data = process_agent_data(agent_data)
```

```
            # # Store the agent_data in the database (you can
send it to the data processing module)
            if not
self.hub_gateway.save_data(processed_data):
                logging.error("Hub is not available")
        except Exception as e:
            logging.info(f"Error processing MQTT message:
{e}")

    def connect(self):
        self.client.on_connect = self.on_connect
        self.client.on_message = self.on_message
        self.client.connect(self.broker_host,
self.broker_port, 60)

    def start(self):
        self.client.loop_start()

    def stop(self):
        self.client.loop_stop()
```

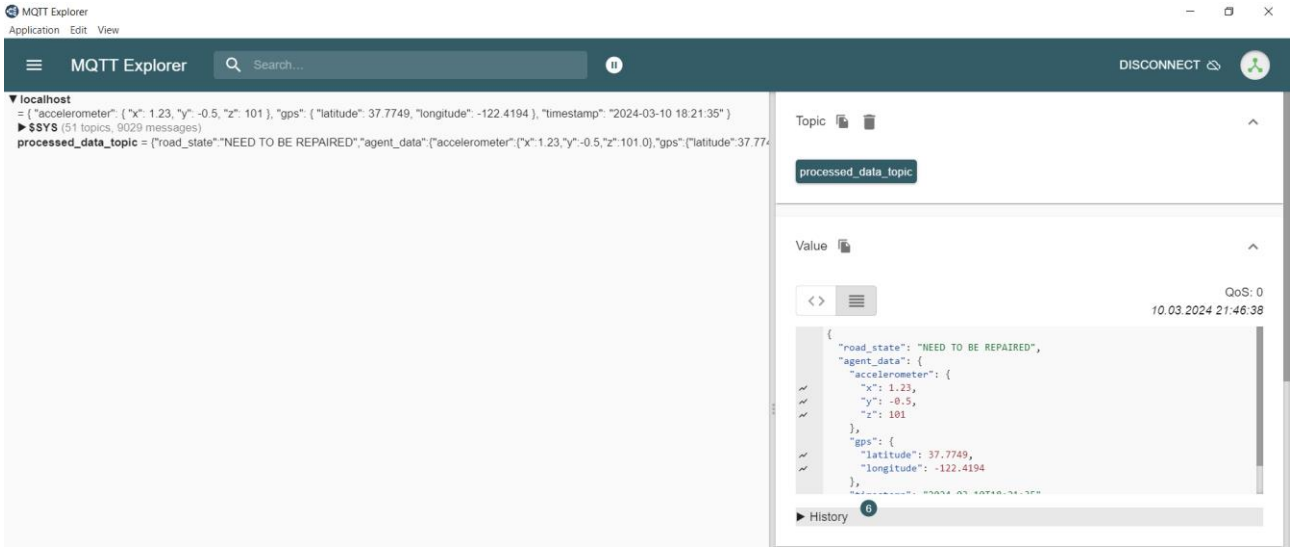
```
def process_agent_data(
    agent_data: AgentData,
) -> ProcessedAgentData:
    if (agent_data.accelerometer.z < 14000):
        return ProcessedAgentData(
            road_state = 'POTHOLE',
            agent_data = agent_data
        )
    elif (agent_data.accelerometer.z > 18000):
        return ProcessedAgentData(
            road_state = 'BUMP',
            agent_data = agent_data
        )

    return ProcessedAgentData(
        road_state = 'FINE',
        agent_data = agent_data
    )
```

3) MQTT

Z ϵ [−100, 100]



Інакше



Висновки:

Під час виконання даної лабораторної роботи проведено ознайомлення з можливостями MQTT-клієнта для прослуховування повідомлень, їх перевірки та обробки. Реалізовано обробку повідомлень мовою Python для визначення стану покриття.