UPUTSTVO ZA STUDENTE – IoT PROJEKTI (BESPLATNI ALATI)

Verzija V4: Dodate konkretne komande i primeri koda (MQTT, ThingsBoard, Node-RED, InfluxDB, Grafana).

0. Pretpostavke i preporu■eni OS

Preporuka: Ubuntu 22.04/24.04 (ili WSL na Windows-u). Sve komande su date za Linux terminal.

1. Pregled besplatnih alata

Alat	Uloga	Kako se koristi (kratko)
Mosquitto (MQTT broker)	Razmena poruka	mosquitto + mosquitto_sub/pub
Python (paho-mqtt)	Simulator senzora	skripta šalje JSON telemetriju
ThingsBoard CE	IoT platforma	Device + access token + dashboard + alarm
Node-RED	Logika/ETL	MQTT in → obrada → InfluxDB out
InfluxDB	Baza (time-series)	■uva telemetriju i omogu■ava upite
Grafana	Dashboard	poveže se na InfluxDB i crta grafikone

VEŽBA 1: MQTT + simulacija senzora (korak-po-korak)

Cilj: Pokrenuti lokalni MQTT broker, testirati publish/subscribe i slati simulirane senzorske podatke iz Python-a.

1.1 Instalacija Mosquitto brokera

```
# Ubuntu
sudo apt update
sudo apt install -y mosquitto mosquitto-clients
sudo systemctl enable --now mosquitto
sudo systemctl status mosquitto --no-pager
```

Ako je sve OK, servis je „active (running)“.

1.2 Test publish/subscribe u dva terminala

Terminal A (subscriber):

```
mosquitto_sub -h localhost -t iot/telemetry -v
```

Terminal B (publisher):

```
mosquitto_pub -h localhost -t iot/telemetry -m '{"temp":23.4,"hum":41,"ts":1700000000}'
```

U Terminalu A treba da se pojavi poruka.

1.3 Python simulator senzora (MQTT JSON telemetrija)

Instaliraj biblioteku:

```
python3 -m pip install --user paho-mqtt
```

Sa■uvaj kao **sensor_sim_mqtt.py**:

```
#!/usr/bin/env python3
import json, time, random
import paho.mqtt.client as mqtt

BROKER = "localhost"
PORT = 1883
TOPIC = "iot/telemetry"
DEVICE_ID = "sim-01"

client = mqtt.Client(client_id=DEVICE_ID, clean_session=True)
client.connect(BROKER, PORT, keepalive=60)
client.loop_start()

try:
    while True:
        payload = {
            "device": DEVICE_ID,
            "temp": round(random.uniform(18.0, 55.0), 2),
            "hum": round(random.uniform(25.0, 80.0), 1),
            "co": round(random.uniform(0.0, 50.0), 2),   # primer: CO ppm
            "ch4": round(random.uniform(0.0, 2.0), 3),   # primer: CH4 %vol
            "ts": int(time.time())
        }
        client.publish(TOPIC, json.dumps(payload), qos=0, retain=False)
        print("sent:", payload)
        time.sleep(5)
except KeyboardInterrupt:
    pass
finally:
    client.loop_stop()
    client.disconnect()
```

Pokretanje:

```
python3 sensor_sim_mqtt.py
```

U drugom terminalu (subscriber):

```
mosquitto_sub -h localhost -t iot/telemetry -v
```

VEŽBA 2: ThingsBoard CE – dashboard + alarm (MQTT)

Cilj: Povezati ure■aj na ThingsBoard (CE), prikazati telemetriju i napraviti alarm kada temperatura pre■e prag.

2.1 Brza instalacija ThingsBoard CE (Docker)

Najbrže za laboratoriju: Docker. (Ako Docker nije instaliran, instalirajte ga prema zvani■nom uputstvu.)

```
# Pokretanje ThingsBoard CE (jednostavan docker run primer)
# Napomena: Ovo je minimalni primer; u praksi se ■esto koristi docker-compose.

docker run -it -p 8080:8080 -p 1883:1883 --name tb-ce --restart unless-stopped thingsboard/tb-postgres
```

Otvorite u browser-u: http://localhost:8080 (default login: tenant@thingsboard.org / tenant)

2.2 Kreiranje Device + preuzimanje Access Token-a

U ThingsBoard UI: 1) **Devices** → **+** → Add new device (npr. „Deponija-Sim-01“) 2) Otvorite ure■aj → tab **Credentials** → kopirajte **Access token**

2.3 Slanje telemetrije na ThingsBoard MQTT endpoint

ThingsBoard MQTT topic za telemetriju je standardno:

```
v1/devices/me/telemetry
```

Test slanje iz terminala (zameni YOUR_TOKEN):

```
mosquitto_pub -h localhost -p 1883 -t v1/devices/me/telemetry -u "YOUR_TOKEN" -m '{"temp":42.1,"hum":33
```

Varijanta: iz Python simulatora (modifikuj parametre):

```
# U Python kodu promeni:
BROKER="localhost"
PORT=1883
TOPIC="v1/devices/me/telemetry"
# i prilikom connect-a koristi username=TOKEN:

client.username_pw_set("YOUR_TOKEN")
client.connect(BROKER, PORT, keepalive=60)
```

2.4 Dashboard (grafikon + status)

UI koraci: 1) **Dashboards** → Add new dashboard (npr. „Depo monitoring“) 2) Add widget → Charts → Time-series chart 3) Kao Data source izaberite ure■aj, key: **temp**, zatim **hum**, **co**, **ch4**

2.5 Alarm (Rule Chain)

Najjednostavnije: koristiti postoje■i rule chain i dodati „filter“. Primer logike: ako je **temp > 40** → kreiraj alarm „HIGH_TEMP“.

Prakti■an pristup (UI):

```
1) Rule Chains → Root Rule Chain
2) Dodaj node „Script“ ili „Filter“ (zavisno od verzije)
3) Uslov: msg.temp > 40
4) Poveži na node „Create alarm“ (alarm type: HIGH_TEMP)
```

Test: pošalji temp=45 i proveri u **Alarms** da je kreiran alarm.

VEŽBA 3: Node-RED → InfluxDB → Grafana (open-source stack)

Cilj: Prikupiti MQTT telemetriju, upisati je u InfluxDB i vizuelizovati u Grafani.

3.1 Najbrže: docker-compose (InfluxDB + Grafana + Node-RED + Mosquitto)

Sa■uvaj kao **docker-compose.yml** u praznom direktorijumu:

```
version: "3.8"
services:
  mosquitto:
    image: eclipse-mosquitto:2
    ports:
      - "1883:1883"

  influxdb:
    image: influxdb:2
    ports:
      - "8086:8086"
    environment:
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=admin
      - DOCKER_INFLUXDB_INIT_PASSWORD=adminadminadmin
      - DOCKER_INFLUXDB_INIT_ORG=iot-org
      - DOCKER_INFLUXDB_INIT_BUCKET=iot-bucket
      - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=MY_INFLUX_TOKEN
    volumes:
      - influxdb_data:/var/lib/influxdb2

  nodered:
    image: nodered/node-red:latest
    ports:
      - "1880:1880"
    depends_on:
      - mosquitto
      - influxdb

  grafana:
    image: grafana/grafana:latest
    ports:
      - "3000:3000"
    depends_on:
      - influxdb
    volumes:
      - grafana_data:/var/lib/grafana

volumes:
  influxdb_data:
  grafana_data:
```

Pokreni:

```
docker compose up -d
docker compose ps
```

Pristup:

```
Node-RED: http://localhost:1880
InfluxDB:  http://localhost:8086  (admin / adminadminadmin)
Grafana:   http://localhost:3000  (admin / admin)
```

3.2 Node-RED flow (MQTT in → JSON → InfluxDB out)

U Node-RED: 1) Menu → Manage palette → Install: **node-red-contrib-influxdb** 2) Dodaj node: **mqtt in** (topic: iot/telemetry, server: mosquitto:1883) 3) Dodaj node: **json** (pretvara string u objekt) 4) Dodaj node: **function** (format za Influx) 5) Dodaj node: **influxdb out** (InfluxDB v2, org=iot-org, bucket=iot-bucket, token=MY_INFLUX_TOKEN)

Primer Function node koda:

```
// Node-RED Function node
// o■ekuje msg.payload = {temp:..., hum:..., co:..., ch4:..., ts:...}
const p = msg.payload;
```

```
msg.payload = [{
  measurement: "telemetry",
  fields: {
    temp: Number(p.temp),
    hum: Number(p.hum),
    co:  Number(p.co),
    ch4: Number(p.ch4)
  },
  tags: {
    device: p.device || "sim-01"
  },
  timestamp: (p.ts ? new Date(p.ts * 1000).toISOString() : new Date().toISOString())
}];

return msg;
```

Pošalji test telemetriju (sa host-a):

```
mosquitto_pub -h localhost -t iot/telemetry -m '{"device":"sim-01","temp":31.2,"hum":44,"co":9.1,"ch4":
```

3.3 Grafana – Data source + Dashboard

U Grafani: 1) Connections → Data sources → Add data source → **InfluxDB** 2) Query language: **Flux** 3) URL: http://influxdb:8086 4) Org: iot-org, Token: MY_INFLUX_TOKEN, Default bucket: iot-bucket 5) Save & Test

Primer Flux upita za temperaturu:

```
from(bucket: "iot-bucket")
  |> range(start: -1h)
  |> filter(fn: (r) => r._measurement == "telemetry")
  |> filter(fn: (r) => r._field == "temp")
  |> aggregateWindow(every: 10s, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

Bonus: Alarm u Grafani (Alerting) – prag npr. temp > 40.

4. Checklist za odbranu projekta

Student treba da pokaže: 1) izvor podataka (senzor ili simulator), 2) komunikaciju (MQTT/HTTP) i primer poruke, 3) dashboard (ThingsBoard ili Grafana) sa bar 2 grafika, 4) alarm pravilo i demonstraciju okidanja, 5) kratak opis arhitekture (blok dijagram).