SVEPRISUTNO RAČUNARSTVO

Student: Teodora Kocić, 1457

Profesor: Dragan Stojanović

SADRŽAJ

Uvod

U okviru prvog projekta podaci koji se očitavaju sa senzora **Arduino**-a obrađuju se na *edge* platformi **OpenRemote**. Edge obrada vrši se na edge uređaju **Raspberry Pi 4**. Finalni proizvod projekata 2 i 3 je **Android** aplikacija *Smart Car* kojom se "upravlja" i kontroliše stanje pametnog vozila korisnika. Uslovi u vozilu simulirani su senzorskim podacima koji se čitaju sa Arduino-a. Obrada podataka vrši se na edge uređaju Raspberry Pi 4. Akcije koje se izvršavaju na aktuatorima (paljenje svetala, aktivacija kočionog sistema, parking senzori, detekcija osoba - DL model koji se izvršava na Arduino-u i Raspberry Pi 4), kao odgovor na promene stanja vozila, simulirane su slanjem komandi nazad na Arduino.

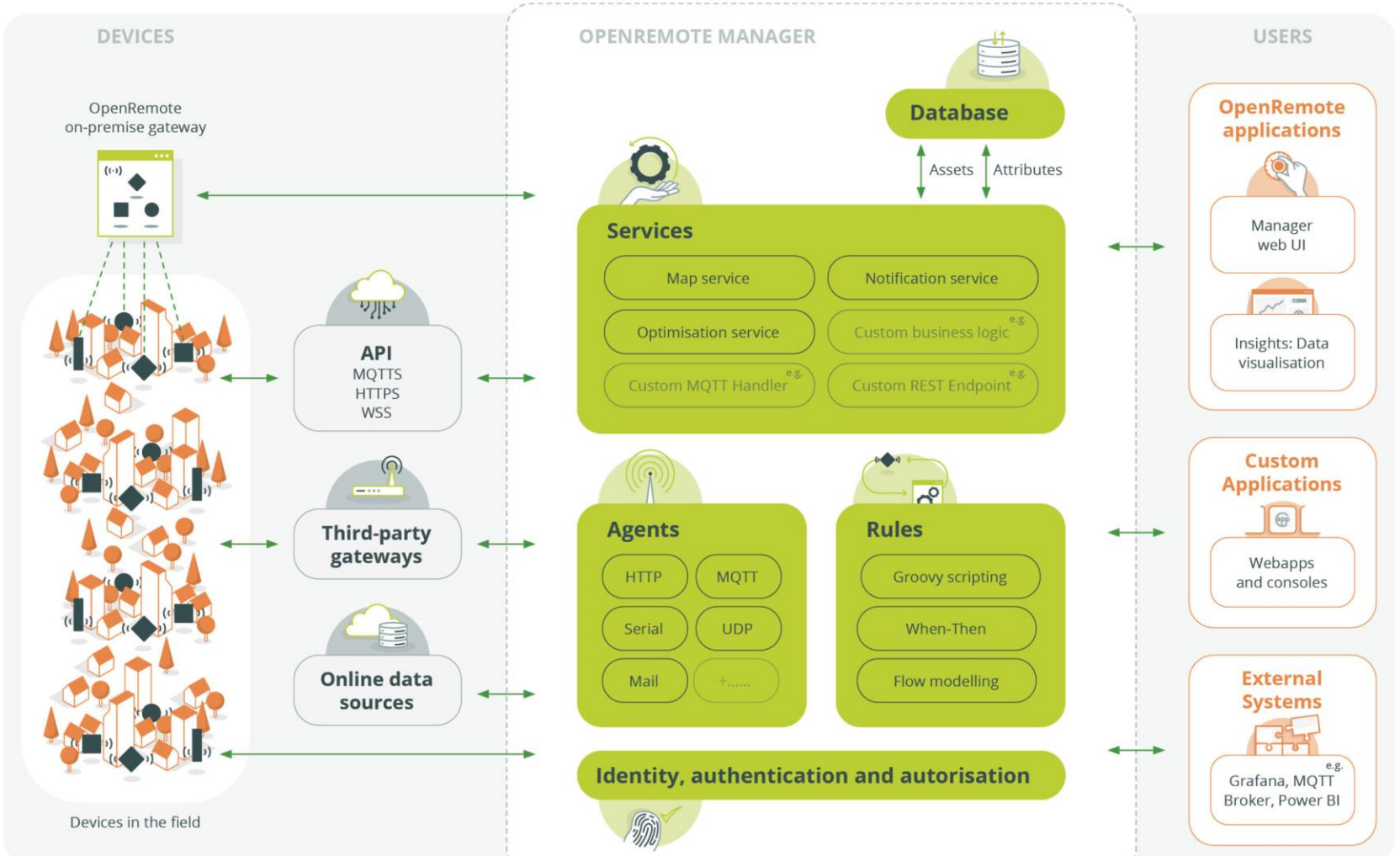
Korišćeni uređaji i tehnologije

Arduino	Raspberry Pi 4	OpenRemote	MQTT, InfluxDB, eKuiper i Grafana	Android
Koristi se mikrokontroler Arduino Nano 33 BLE Sense. Koriste se senzori LPS22HB (temperatura i pritisak), APDS9960 (boja, udaljenost i položaj), LSM9DS1 (linearno i rotaciono ubrzanje, kao i jačina magnetnog polja) i dodatno kamerica koja se koristi za detekciju osoba.	Mikroračunar na kojem se vrši obrada podataka na Edge-u. U prvom projektu se na njemu pokreće *open source* platforma na kojoj se podaci sa senzora filtriraju. Dok se kod 2. i 3. projekta na mikroračunaru vrši obrada podataka, njihov upis u bazu i vizuelizacija, kao i DL algoritam za detekciju osoba. Svaki servis koji se pokreće na edge-u je prethodno dokerizovan.	Platforma koja je u potpunosti *open source*. Obezbeđuje kreiranje kompletnog rešenja za upravljanje IoT uređajima, uključujući automatsko obezbeđivanje, kreiranje i upravljanje različitim vrstama atributa, kreiranje pravila (when-then, groovy i pravila toka), analiza podataka, povezivanje sa različitim protokolima (HTTP/REST, MQTT, WS).	MQTT broker se koristi kako bi se u svakom projekatu podaci sa Arduino-a slali na edge i nakon filtriranja/obrade vraćali natrag do klijenta, odnsono mobilne aplikacije. Podaci koji se šalju na mikroračunar upisuju se u Influx bazu podataka i kasnije je moguća vizuelizacija u Grafana-i. eKuiper pravila se koriste za obradu i filtriranje podataka koji se vraćaju mobilnoj aplikaciji.	Android mobilna aplikacija se koristi za vizuelizaciju obrađenih podataka koja se po eKuiper pravilima vraćaju korisniku. U okviru aplikacije postoje biblioteke za kreiranje MQTT klijenta koji direktno prima podatke poslate iz eKuiper-a, kao i kreiranje BLE klijenta kojim se ostvaruje BLE konekcija sa Arduino board-om. Arduino-u se iz aplikacije šalju komande kojima sa aktiviraju akcije na aktuatoru (Arduino-u).

PROJEKAT 1
OPENREMOTE

IoT open source platforma koja pojednostavljuje povezivanje mrežnih sredstava (assets) sa mobilnim i web aplikacijama

ARHITEKTURA PLATFORME



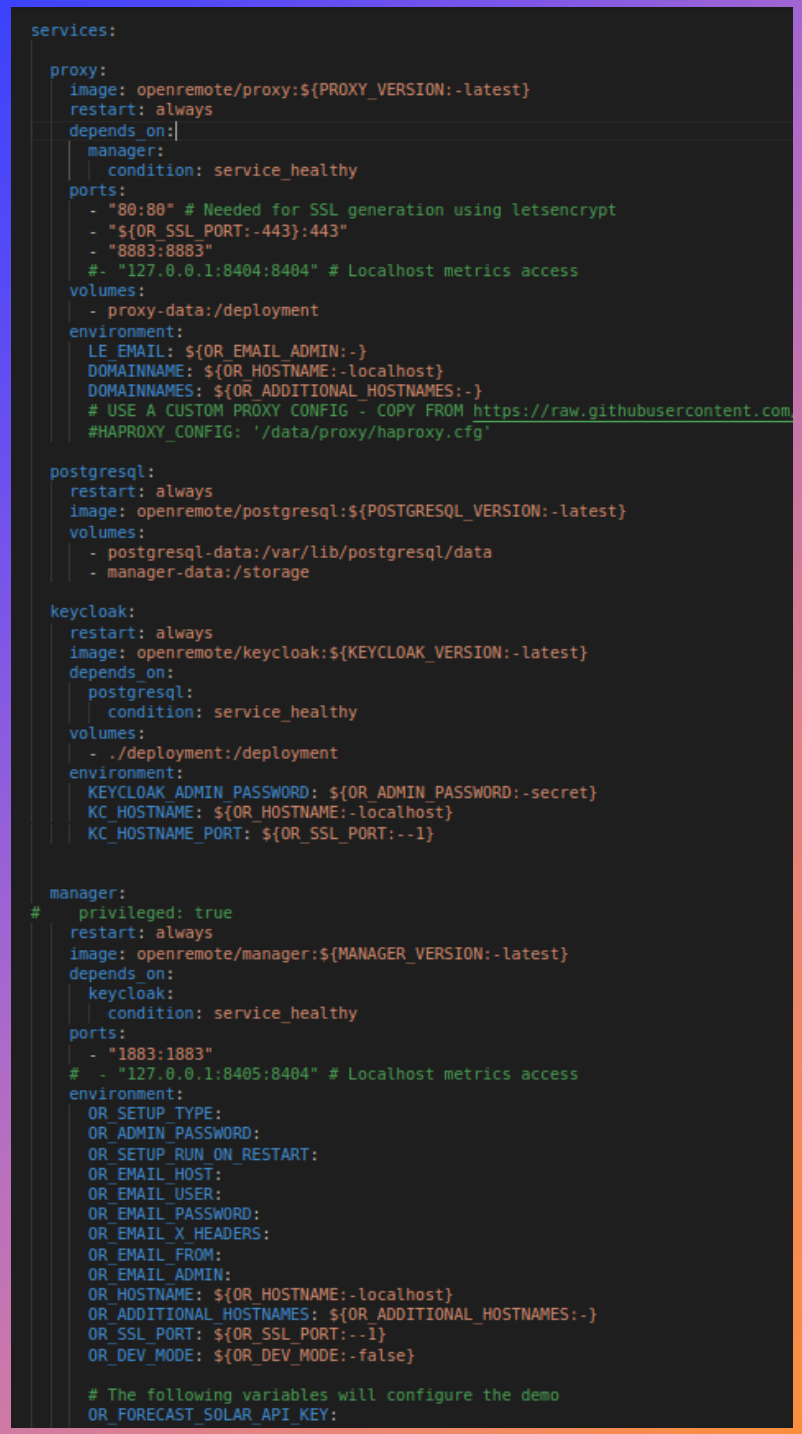
OPENREMOTE KOMPONENTE

- **ASSETS (SREDSTVA)** - SKUP SVIH ATRIBUTA ZA KOJE POSTOJI OPIS I ISTORIJA PROMENA. TAKOĐE, MOGUĆE JE DODAVATI NOVE ATRIBUTE I ODRADITI NJIHOVU KONFIGURACIJU

- **AGENTS (AGENTI) –** SPECIFIČNA VRSTA SREDSTAVA ČIJA JE NAMENA DA OMOGUĆE POVEZIVANJE SA EKSTERNIM SENZORIMA, AKTUATORIMA, ITD. (WEBSOCKET, HTTP, MQTT, TCP, UDP, SNMP, Z-WAVE, VELBUS). AGENTI MOGU DA SE POVEZUJU, FILTRIRAJU I GRUPIŠU KONTROLE I ATRIBUTE

- **RULES (PRAVILA) –** POSTOJE TRI TIPA PRAVILA: WHEN-THEN, GROOVY I FLOW PRAVILA. WHEN-THEN PRAVILA POKREĆU AKCIJU NAD ATRIBUTOM KOJA PRETHODI PROMENI NEKOG DRUGOG ATRIBUTA; FLOW PRAVILA OBRAĐUJU ATRIBUTE I KREIRAJU NOVE; GROOVY PRAVILIMA SE ATRIBUTI UPRAVLJAJU NAPREDNOM LOGIKOM

- **INSIGTHS (PREGLED)** - KORISTI SE ZA KREIRANJE DASHBOARD-OVA U OKVIRU MANAGER-A.



```yaml
services:

  proxy:
    image: openremote/proxy:${PROXY_VERSION:-latest}
    restart: always
    depends_on:
      manager:
        condition: service_healthy
    ports:
      - "80:80" # Needed for SSL generation using letsencrypt
      - "${OR_SSL_PORT:-443}:443"
      - "8883:8883"
      #- "127.0.0.1:8404:8404" # Localhost metrics access
    volumes:
      - proxy-data:/deployment
    environment:
      LE_EMAIL: ${OR_EMAIL_ADMIN:-}
      DOMAINNAME: ${OR_HOSTNAME:-localhost}
      DOMAINNAMES: ${OR_ADDITIONAL_HOSTNAMES:-}
      # USE A CUSTOM PROXY CONFIG - COPY FROM https://raw.githubusercontent.com/
      #HAPROXY_CONFIG: '/data/proxy/haproxy.cfg'

  postgresql:
    restart: always
    image: openremote/postgresql:${POSTGRESQL_VERSION:-latest}
    volumes:
      - postgresql-data:/var/lib/postgresql/data
      - manager-data:/storage

  keycloak:
    restart: always
    image: openremote/keycloak:${KEYCLOAK_VERSION:-latest}
    depends_on:
      postgresql:
        condition: service_healthy
    volumes:
      - ./deployment:/deployment
    environment:
      KEYCLOAK_ADMIN_PASSWORD: ${OR_ADMIN_PASSWORD:-secret}
      KC_HOSTNAME: ${OR_HOSTNAME:-localhost}
      KC_HOSTNAME_PORT: ${OR_SSL_PORT:--1}


  manager:
#    privileged: true
    restart: always
    image: openremote/manager:${MANAGER_VERSION:-latest}
    depends_on:
      keycloak:
        condition: service_healthy
    ports:
      - "1883:1883"
    #  - "127.0.0.1:8405:8404" # Localhost metrics access
    environment:
      OR_SETUP_TYPE:
      OR_ADMIN_PASSWORD:
      OR_SETUP_RUN_ON_RESTART:
      OR_EMAIL_HOST:
      OR_EMAIL_USER:
      OR_EMAIL_PASSWORD:
      OR_EMAIL_X_HEADERS:
      OR_EMAIL_FROM:
      OR_EMAIL_ADMIN:
      OR_HOSTNAME: ${OR_HOSTNAME:-localhost}
      OR_ADDITIONAL_HOSTNAMES: ${OR_ADDITIONAL_HOSTNAMES:-}
      OR_SSL_PORT: ${OR_SSL_PORT:--1}
      OR_DEV_MODE: ${OR_DEV_MODE:-false}

      # The following variables will configure the demo
      OR_FORECAST_SOLAR_API_KEY:
```

Srž sistema je **Manager –** Java aplikacija koja formira broker IoT konteksta (koji beleži trenutno stanje IoT sistema).
Sigurnost IoT sistema bazira se na industrijskom standardu za autentifikaciju: integracija Manager-a i **Keycloak**-a.
Podaci se skladište u **PostgreSQL** bazi podatak.

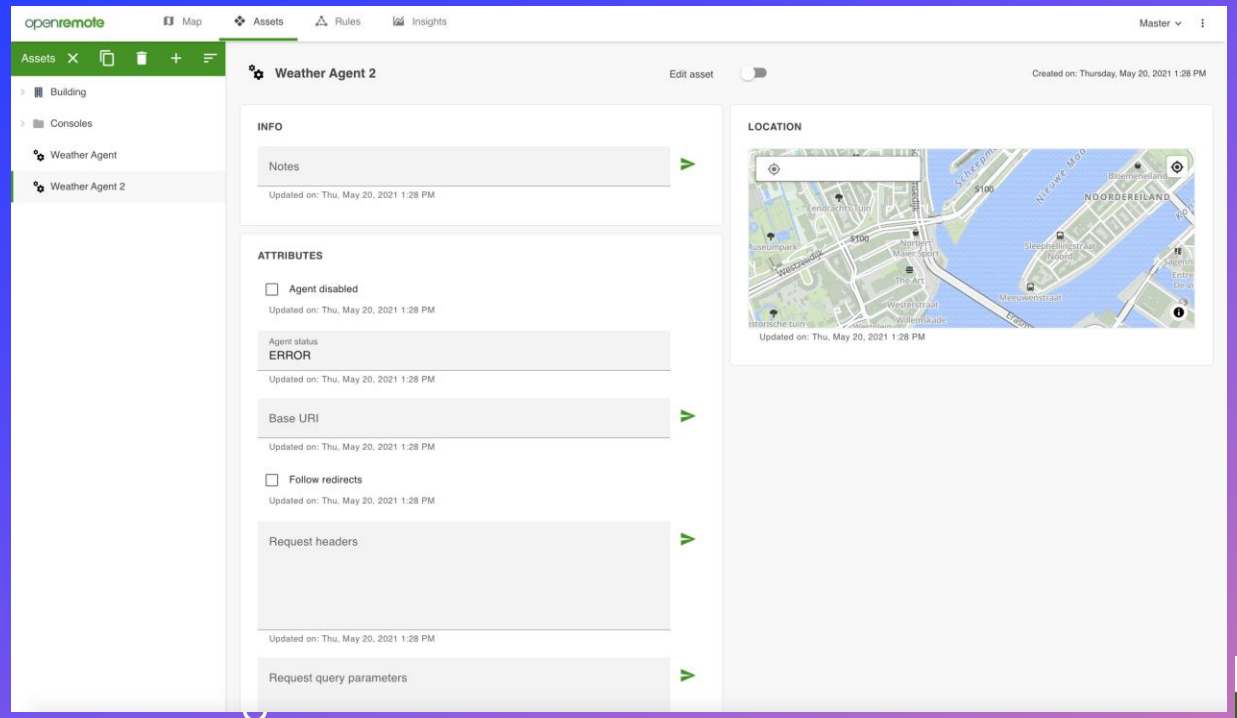
Povezivanje komponenti u Projektu 1

Arduino

- Arduino mikrokontroler povezan je na računar i preko *serijskog porta* šalje senzorske podatke
- Python skripta čita podatke direktno sa Arduino-a i šalje ih na odgovarajući MQTT topic (agent kreiran na OpenRemote platformi)

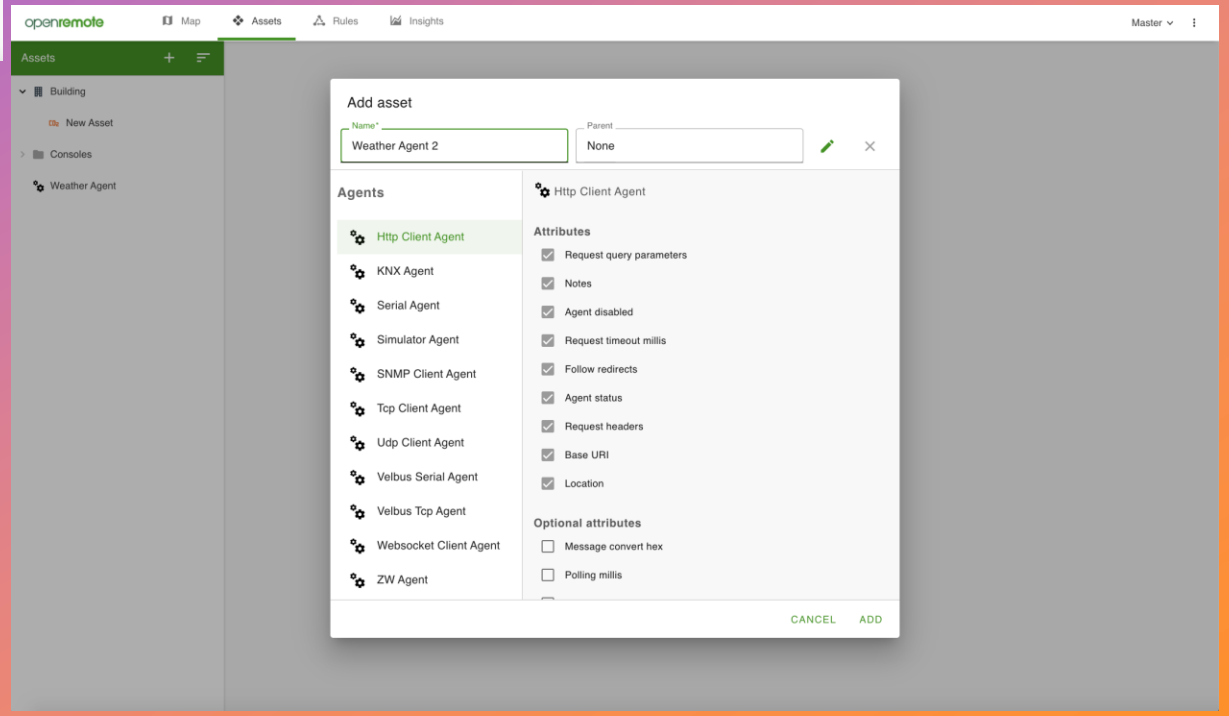
Raspberry Pi 4

- OpenRemote platforma je pokrenuta na mikroračunaru (edge-u)
- Kreirani su atributi koji prihvataju senzorske podatke poslate iz Python skripte
- Kreirana su pravila kojima se aktiviraju akcije i korisnik dobija povratnu informaciju
- Vizuelizacija atributa preko dashboard-ova kreiranih na samoj platformi



ATRIBUTI

SREDSTVA
(ASSETS)



- Podaci se šalju na topic:

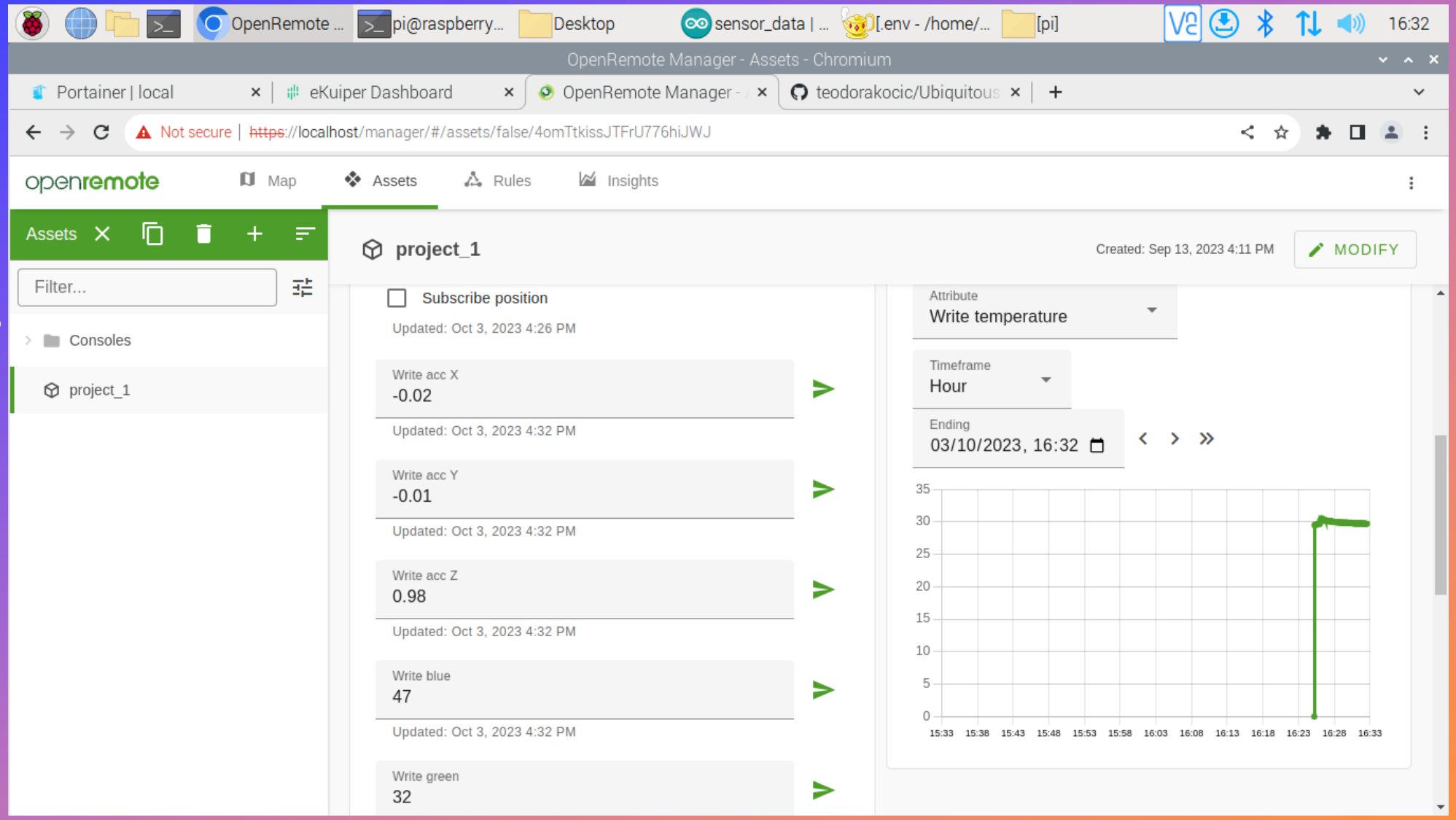
{realm}/{clientId}/writeattributevalue/{attributeName}/{assetId},

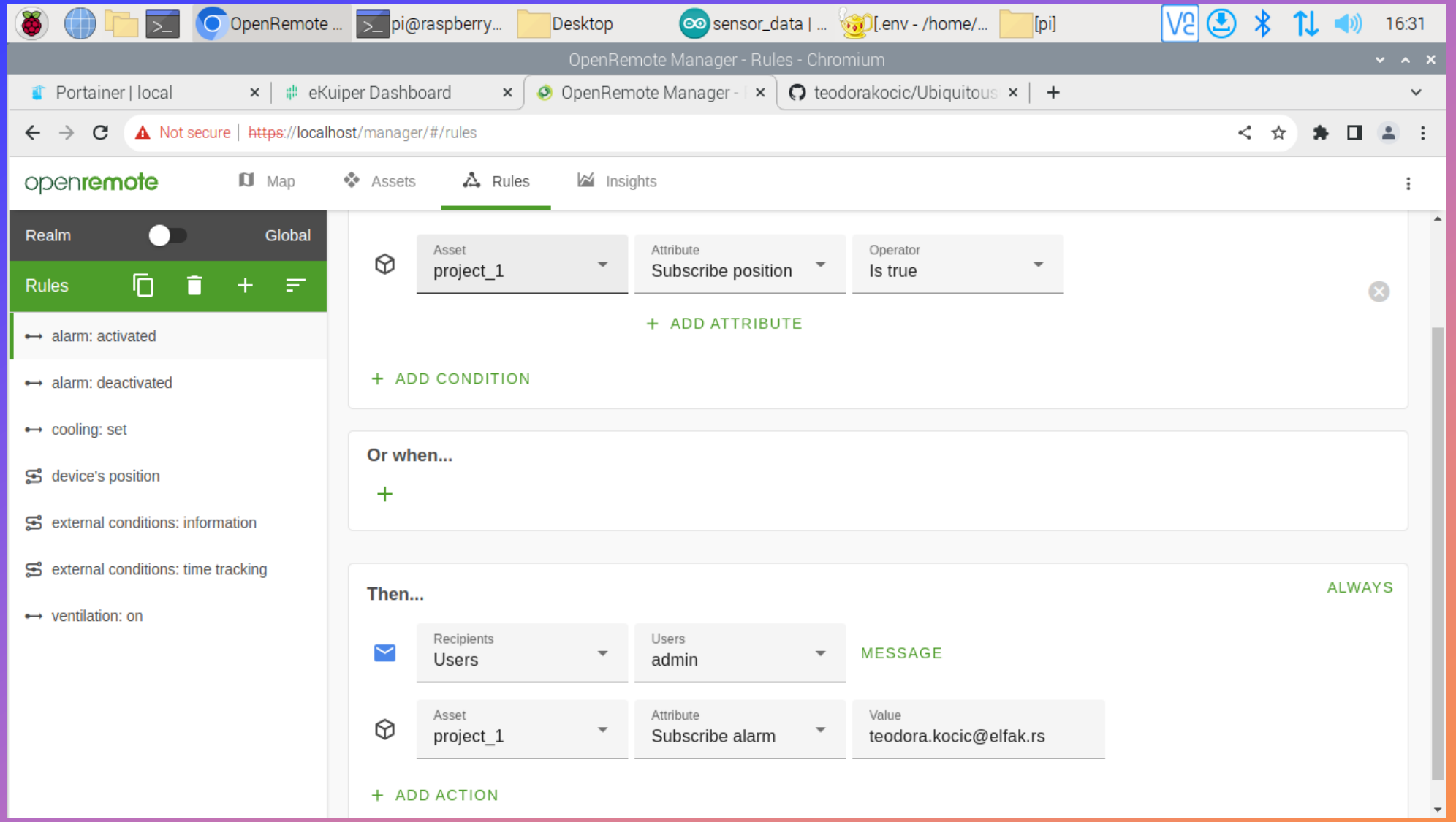
gde su *realm* i *clientId* vrednosti koje sam korisnik kreira, *attributeName* je ime atributa čije vrednosti šaljemo na topic, *assetId* se isčitava direktno iz uri-a (koristi se Manager UI – https://localhost:8080).

- Vrednosti novih atributa, kao i alarme koji su dobijeni kao rezultati kreiranih pravila na OpenRemote platformi korisnik može pratiti kako preko frontend-a (Manager UI), tako i korišćenjem same aplikacije koja šalje senzorske podatke na edge. Topic koji se osluškuje je:

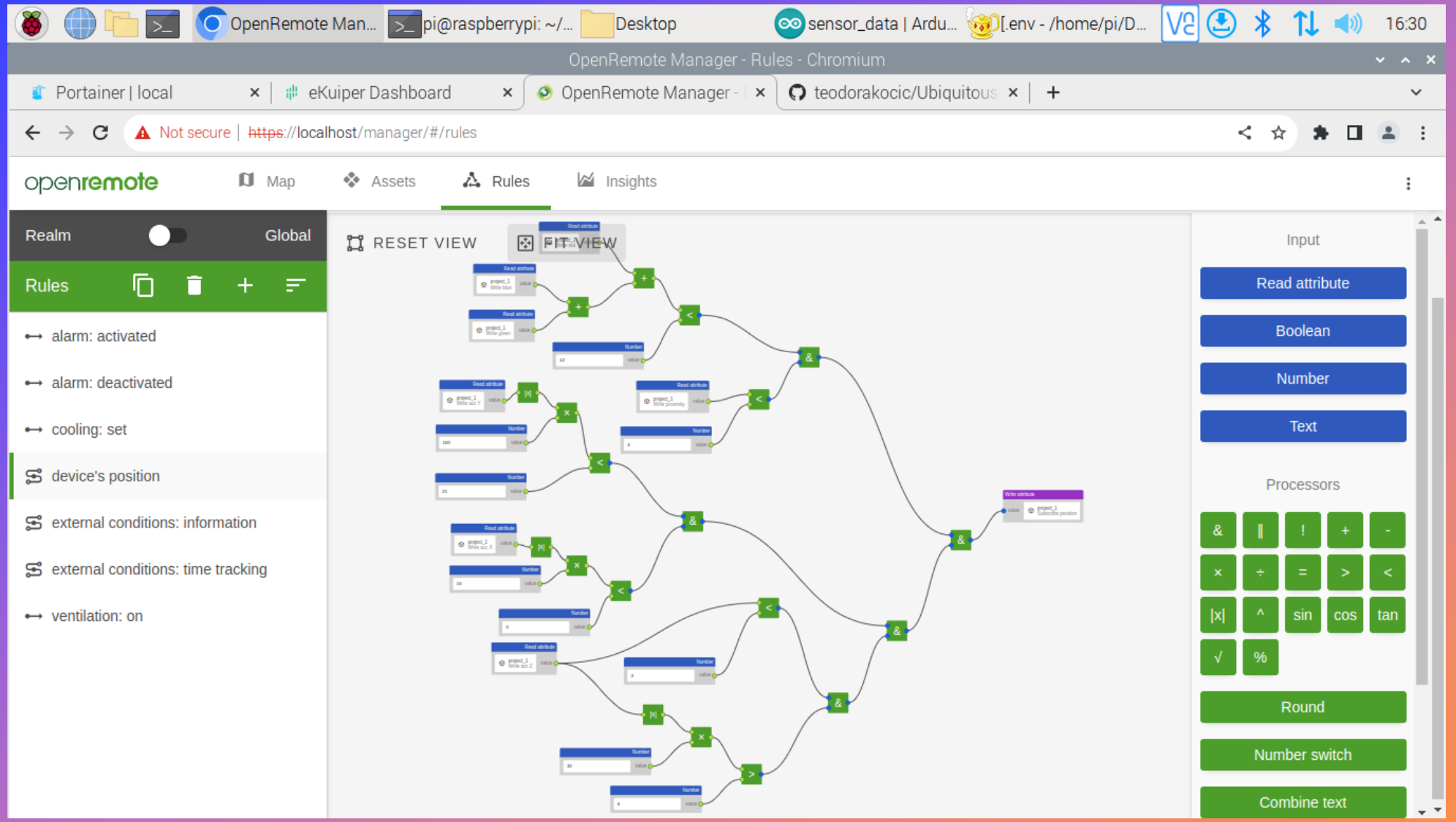
{realm}/{clientId}/attribute/{attributeName}/{assetId}

ATRIBUTI SA PRIKAZOM ISTORIJE



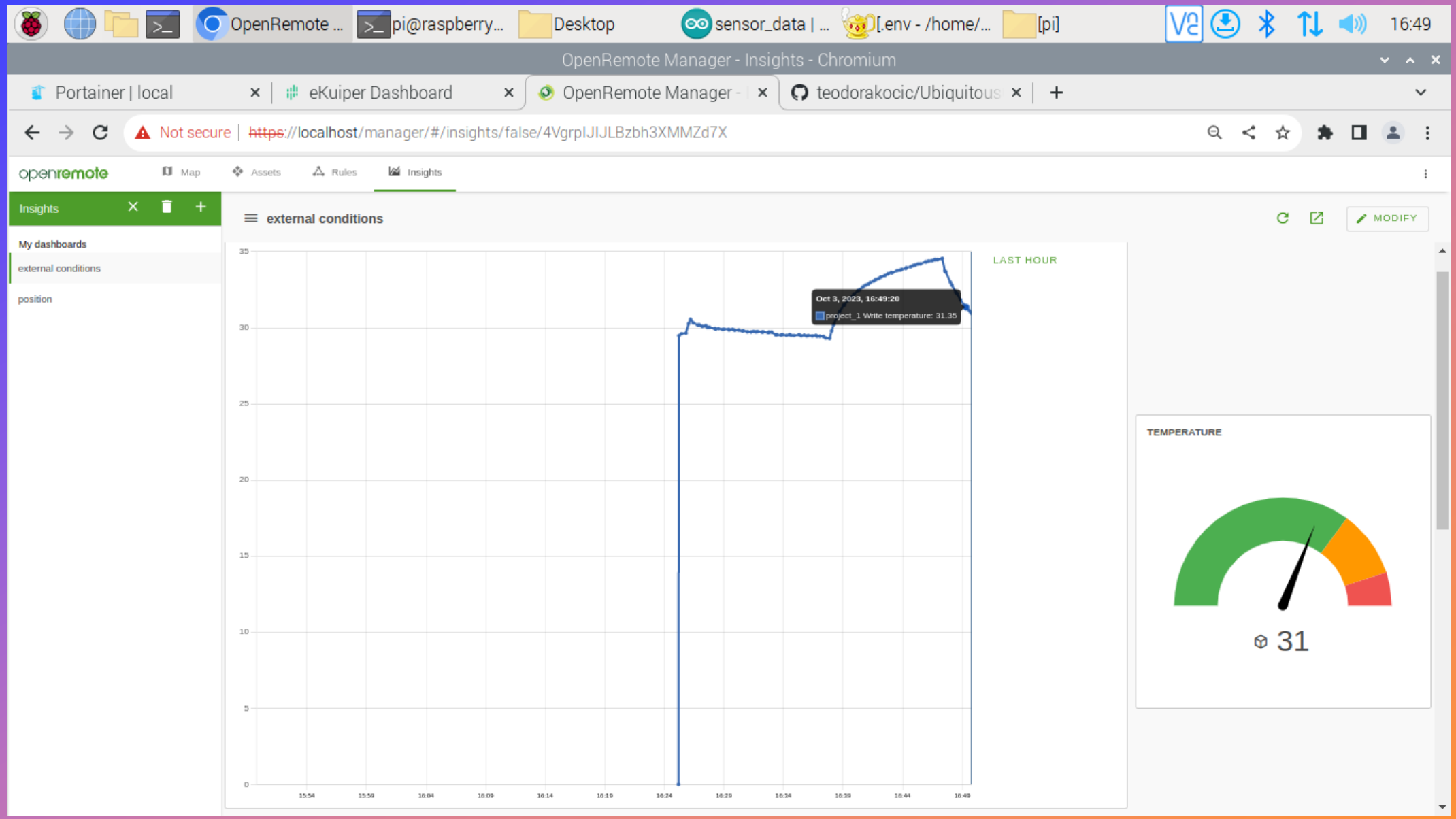


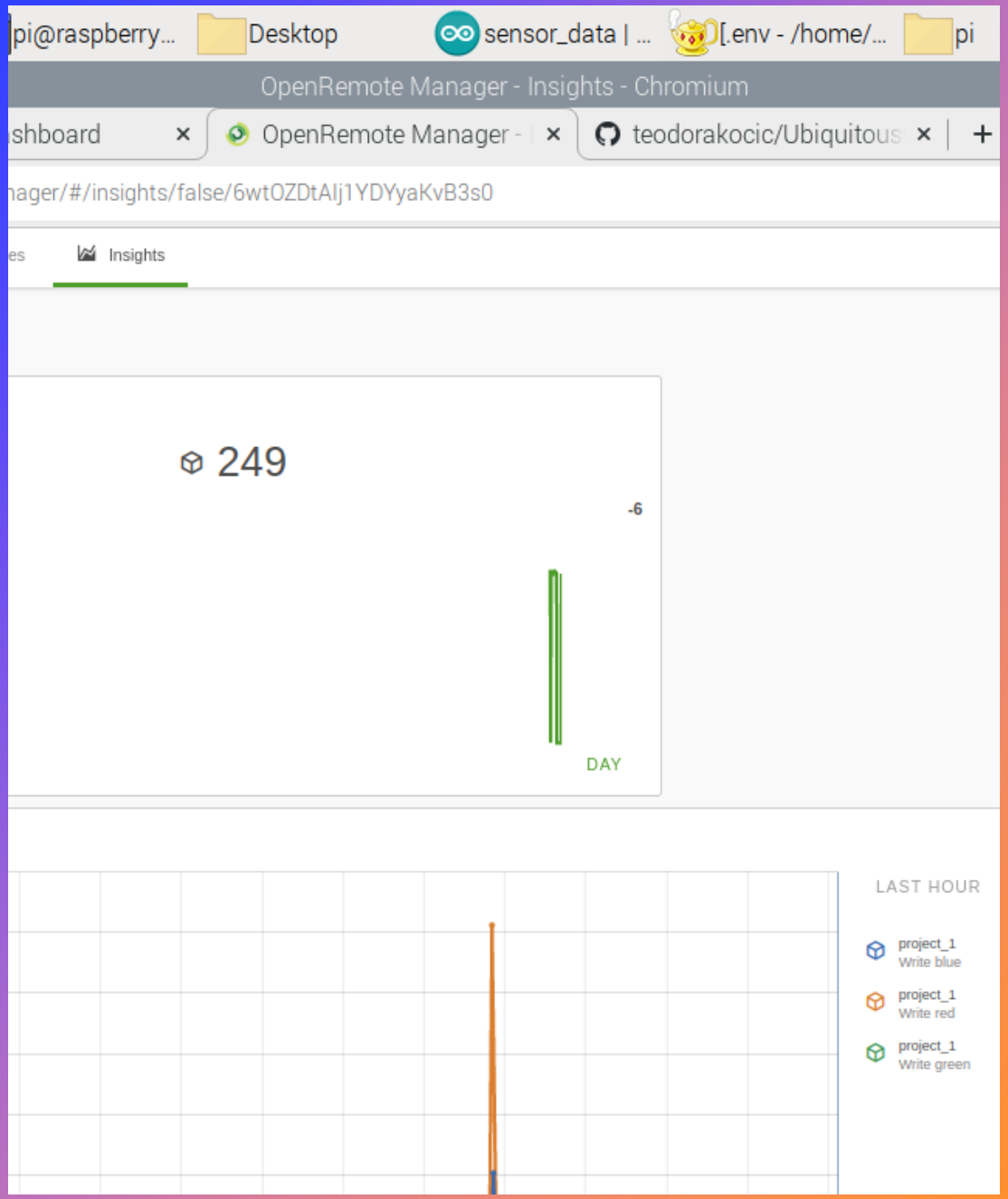
WHEN-THEN PRAVILO

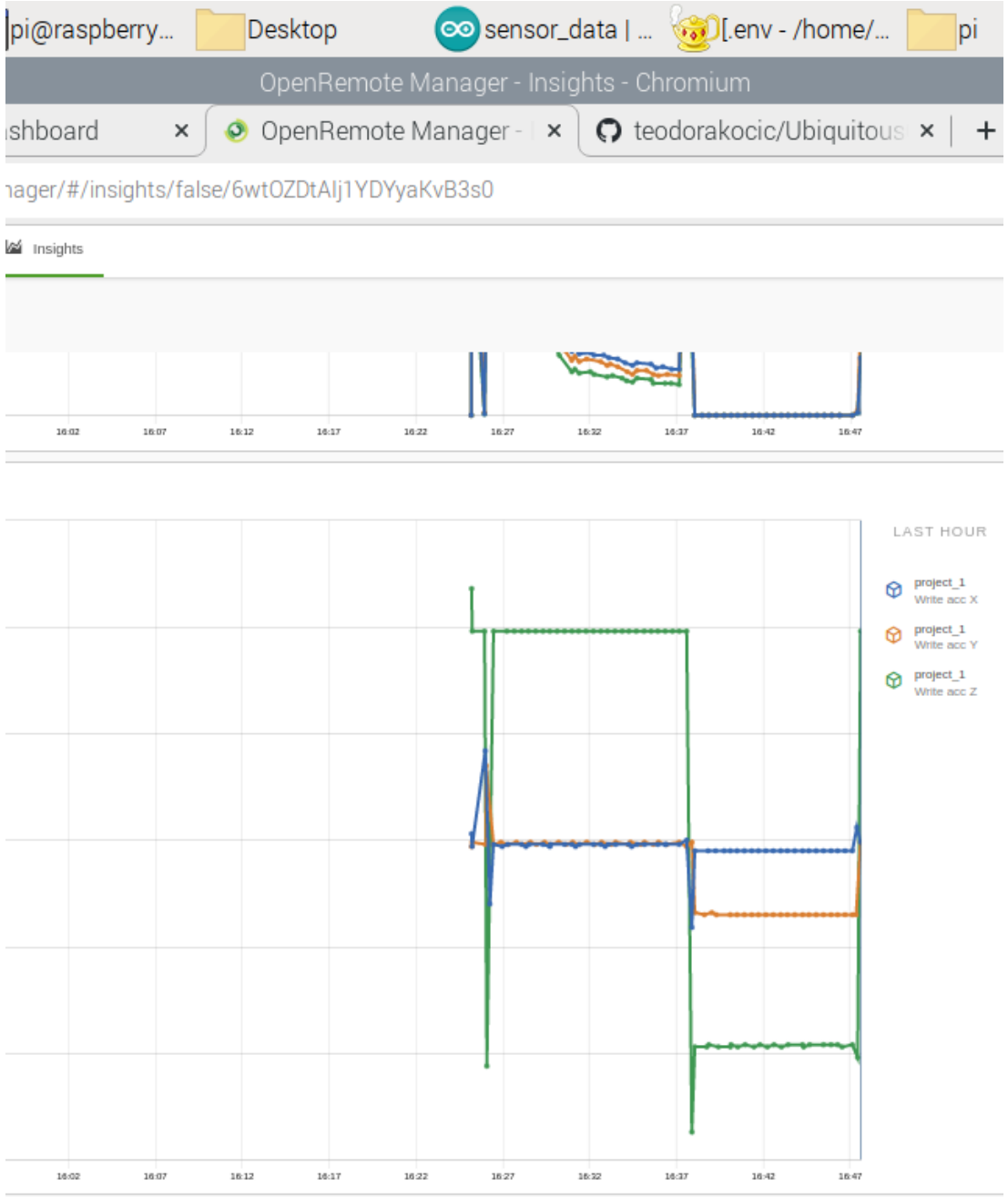


FLOW PRAVILO

DASHBOARD-OVI ZA TEMPERATURU, PROXIMITY I PROMENU AKCELERACIJE




pi@raspberry...
Desktop
sensor_data | ...
[.env - /home/...
pi
OpenRemote Manager - Insights - Chromium
shboard
OpenRemote Manager -
teodorakocic/Ubiquitous
nager/#/insights/false/6wtOZDtAlj1YDYyaKvB3s0
Insights
249
-6
DAY
LAST HOUR
project_1
Write blue
project_1
Write red
project_1
Write green


pi@raspberry...
Desktop
sensor_data | ...
[.env - /home/...
pi
OpenRemote Manager - Insights - Chromium
shboard
OpenRemote Manager -
teodorakocic/Ubiquitous
nager/#/insights/false/6wtOZDtAlj1YDYyaKvB3s0
Insights
LAST HOUR
project_1
Write acc X
project_1
Write acc Y
project_1
Write acc Z

PROJEKAT 2 I PROJEKAT 3

IoT sistem koji uključuje senzore i edge računarske uređaje, koji je povezan sa IoT mobilnom aplikacijom *Smart Car*

Delovi sistema Projekta 2

Arduino

- Očitani senzorski podaci šalju se na serijski port kroz kod koji se izvršava direktno na Arduino-u
- Python skripta podatke sa Arduino-a šalje na endpoint message broker-a

Raspberry Pi 4

- Na mikroračunaru pokrenuti su kontejneri: Mosquitto broker, InfluxDB, Grafana i eKuiper (+ manager) i izvršava se IoT aplikacija koja obezbeđuje komunikaciju između svih navedenih komponenti
- Podaci sa broker-a upisuju se u bazu Influx i mogu biti vizuelizovani korišćenjem Grafana-e
- Isti podaci šalju se do eKuiper-a gde se kroz pravila vrši filtriranje raw podataka

Mobilni telefon

- Android aplikacija koristi se za prikaz filtriranih senzorskih podataka
- MQTT klijent osluškuje topic-e na koje se šalju obrađeni podaci od strane eKuiper-a

File Edit Sketch Tools Help

arduino_read_data

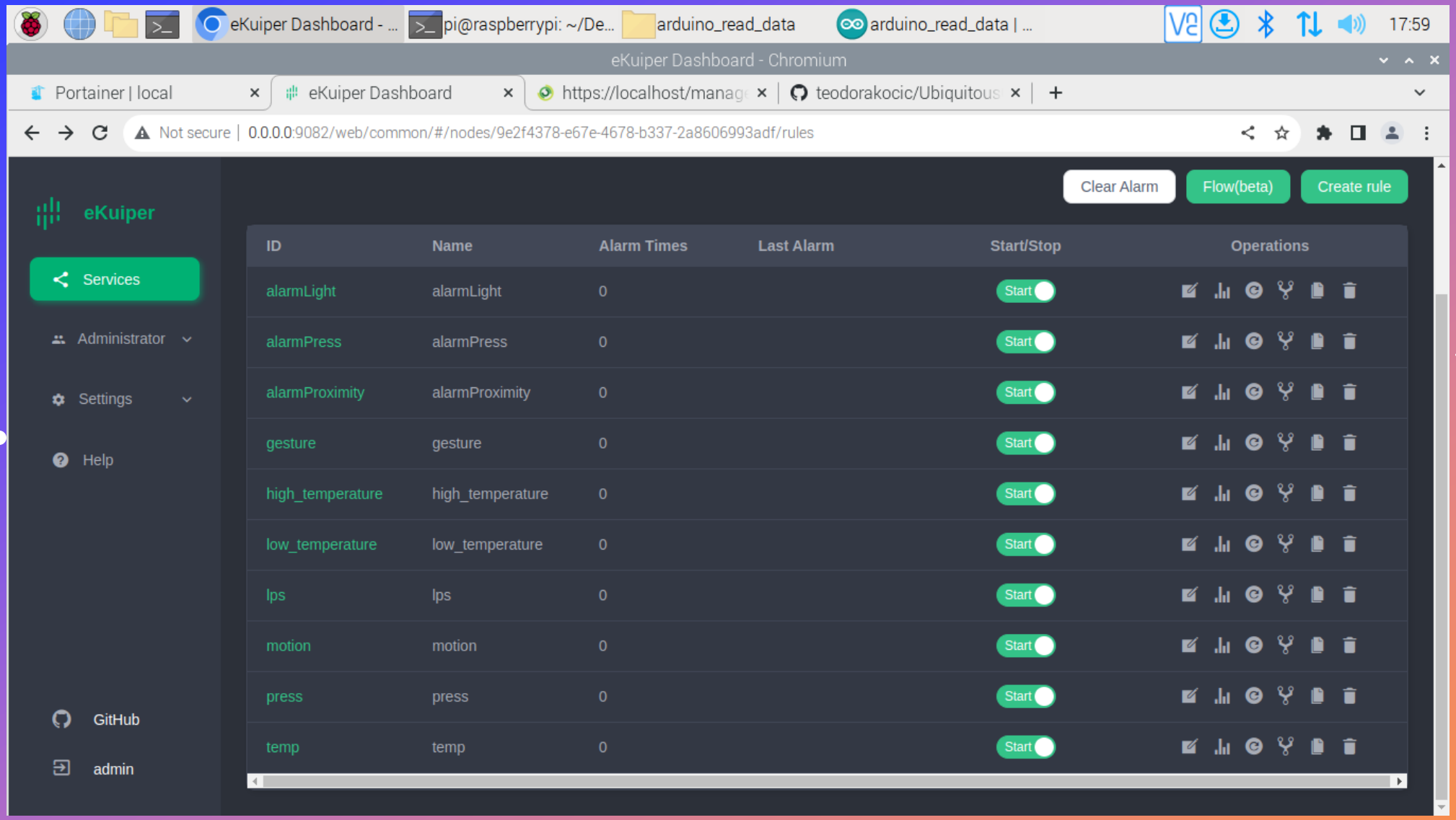
```
void offLED()
{
  digitalWrite(LEDG, HIGH);
  digitalWrite(LEDR, HIGH);
  digitalWrite(LEDB, HIGH);
}

void setup() {
  Serial.begin(9600);
  while (!Serial);

  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);

  if (!BLE.begin()) {
    while (1);
  }

  BLE.setLocalName("NANO 33 BLE");
  BLE.setAdvertisedService(ledService);

  ledService.addCharacteristic(ledCharacteristic);
  ledService.addCharacteristic(buttonCharacteristic);

  BLE.addService(ledService);

  ledCharacteristic.writeValue(0);
  buttonCharacteristic.writeValue(0);

  BLE.advertise();
//  Serial.println("Bluetooth® device active, waiting for connections...");/

  if (!BARO.begin()) {
    Serial.println("Failed to start the LPS22HB sensor.");
    while (1);
  }
```

File Edit Sketch Tools Help

arduino_read_data

```
void loop() {
  BLE.poll();

  char buttonValue = digitalRead(buttonPin);
  bool buttonChanged = (buttonCharacteristic.value() != buttonValue);

  if (buttonChanged) {
    ledCharacteristic.writeValue(buttonValue);
    buttonCharacteristic.writeValue(buttonValue);
  }

  if (ledCharacteristic.written() || buttonChanged)
  {
    if (ledCharacteristic.value())
    {
      if (ledCharacteristic.value() == 2)
      {
         blinkLED(ledPin);
      } else if (ledCharacteristic.value() == 3)
      {
         digitalWrite(LEDR, LOW);
         digitalWrite(LEDG, HIGH);
         digitalWrite(LEDB, HIGH);
         delay(500);
         blinkLED(LEDR);
         offLED();
      } else if (ledCharacteristic.value() == 4)
      {
        digitalWrite(LEDG, LOW);
        digitalWrite(LEDR, HIGH);
        digitalWrite(LEDB, HIGH);
        delay(500);
        blinkLED(LEDG);
        offLED();
```

Kod koji se izvršava na Arduino-u



PRAVILA NA EKUIPER-U KOJA SE ŠALJU NA MQTT TOPIC-E KOJE OSLUŠKUJE ANDROID APLIKACIJA

Resursi korišćeni u okviru Projekta 3

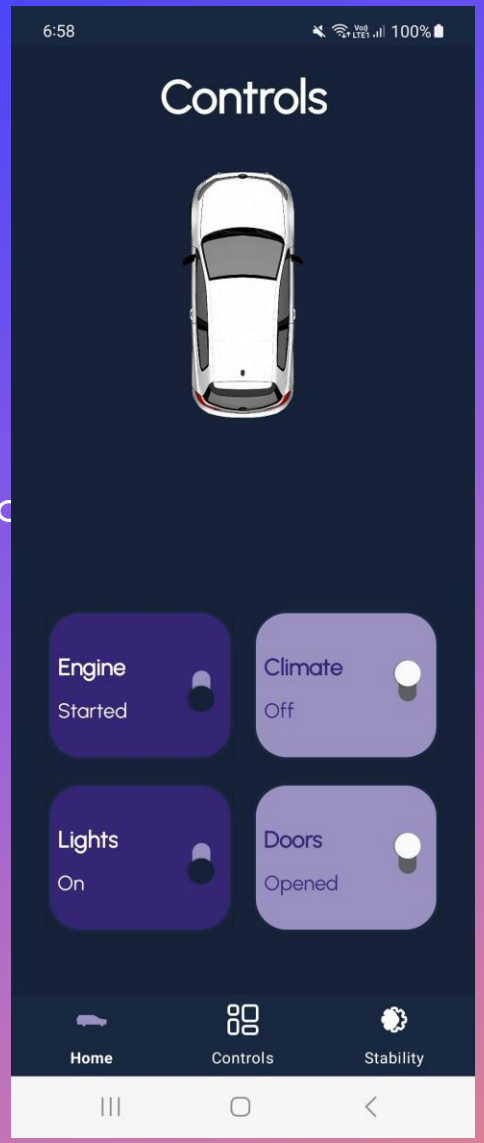
Postojećem sistemu iz Projekta 2 dodate su funkcionalnosti navedene u tabeli ispod.

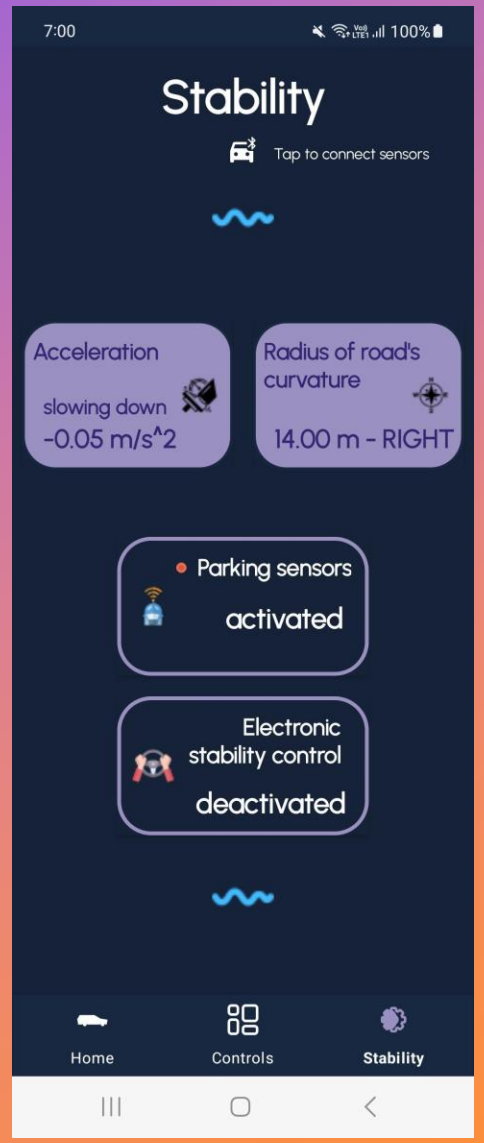
	Arduino	Raspberry Pi 4	Android application
Feature 1	Obezbeđivanje BLE konekcije sa perifernim uređajima	Izvršenje DL algoritma za prepoznavanje osoba (TensorFlow Light)	Dodavanje BLE klijenta radi čitanja i upisivanja BLE karakteristika
Feature 2	Dodavanje BLE servisa i karakteristika zbog akcija na aktuatorima	--	Slanje komandi Arduino-u u cilju aktivacije događaja na aktuatoru (Arduino)
Feature 3	Dodavanje kamerice i izvršavanje algoritma za detekciju osoba	--	Slanje notifikacije korisniku ukoliko je detektovana osoba

DL algoritam napisan je korišćenjem TensorFlow-a, koji se kovertuje u TensorFlow Light kako bi izvršavanje algoritma bilo moguće na edge mikroračunaru. Koristi se gotov model koji je istreniran nad odgovarajućim skupom podataka (slike osoba, mačaka i pasa).

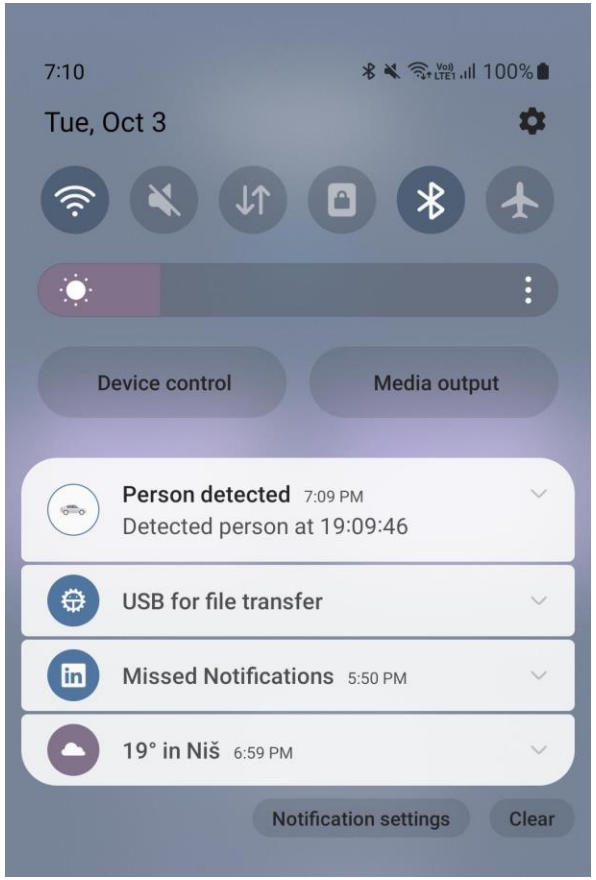
```
#img = tf.keras.utils.load_img(
#    image_path, target_size=(224, 224)
#)
#img_array = tf.keras.utils.img_to_array(img)
img = cv2.imread(image_path)
img = cv2.resize(img, (224,224))
img = img / 255.0
img_array = np.expand_dims(img, 0)
img_array = np.float32(img_array)
#print(img.shape)

interpreter = tf.Interpreter(model_path='tf_lite_model.tflite')

#print(interpreter.get_signature_list())


classify_lite = interpreter.get_signature_runner('serving_default')
classify_lite

predictions_lite = classify_lite(input_1=img_array)['dense_3']
#score_lite = tf.nn.softmax(predictions_lite)
score_lite = np.argsort(predictions_lite[0])

class_labels = ['cat', 'dog', 'person']
predicted_class = class_labels[np.argmax(score_lite)]
print(score_lite)
```

SLIKE EKRANA NAKON POKRETANJA ANDROID APLIKACIJE





BLE skeniranje i korisnička notifikacija nakon detekcije osobe



HVALA NA PAŽNJI!