Detekcija pokreta i obaveštenje putem push notifikacije

predmet : Umreženi Embeded sistemi

student : Milin Ivan E1-79/2023

Opis projekta :

- Sistem se sastoji od dva Raspberry Pi uređaja koji su bežično povezani na istu mrežu putem Wifi.
Jedan Raspberry Pi je u ulozi Edge node-a, a drugi u ulozi Cloud-a.

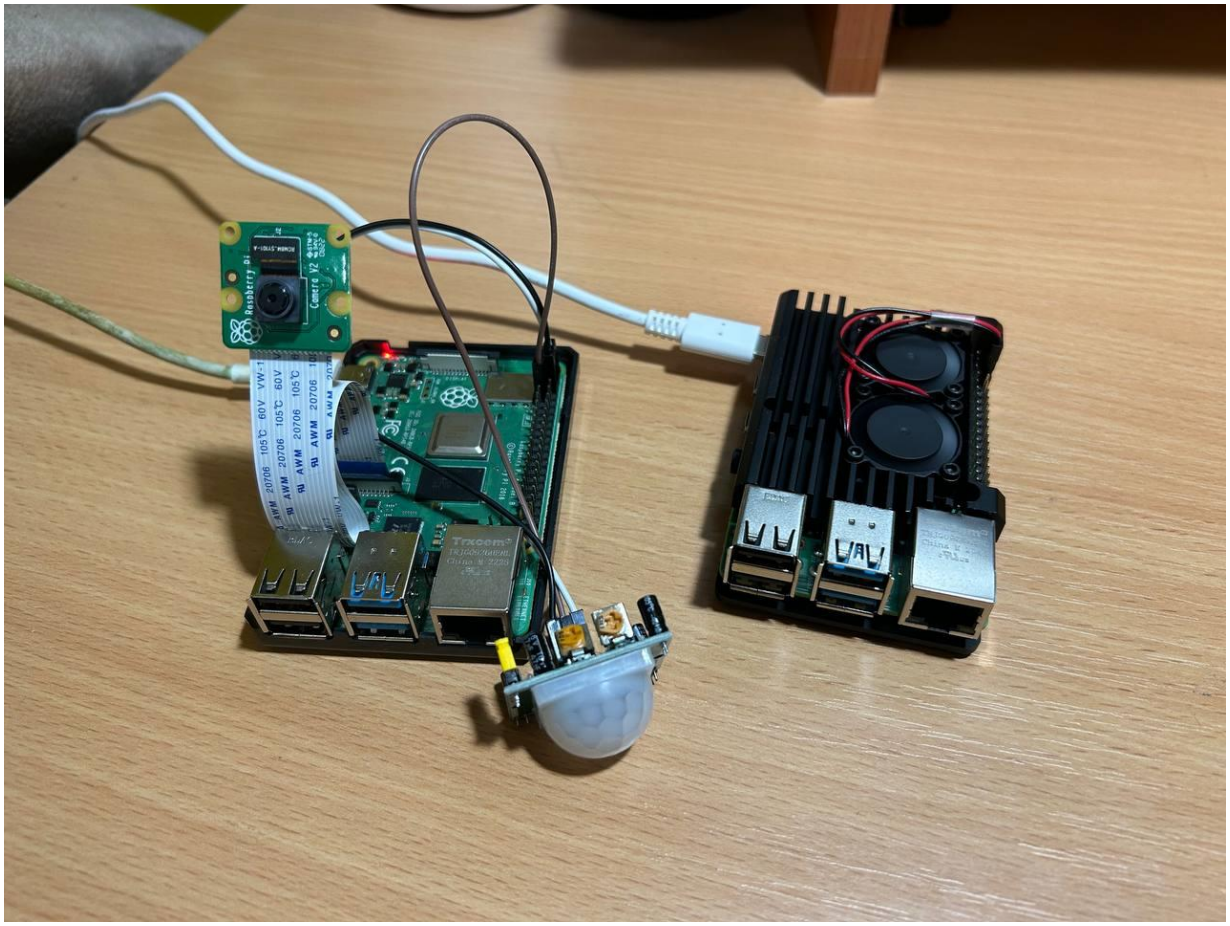
- Raspberry Pi u ulozi Edge node-a na sebi ima povezanu kameru i IR senzor koji detektuje pokret u prostoriji.

- U slučaju detekcije pokreta, edge uređaj šalje snimak ka Cloud-u (drugi Raspberry Pi) pomoću HTTP zahteva, na kom se čuvaju snimci.

- Cloud šalje push notifikaciju na Telegram mobilnu aplikaciju kao vid upozorenja da je detektovano kretanje.

- Snimcima je moguće pristupiti kroz web aplikaciju implementiranu pomoću Streamlit Python biblioteke.

- Pristup snimcima moguć je samo unutar lokalne mreže, dok je slanje poruke ka Telegram aplikaciji moguće i van lokalne mreže (npr. telefon je povezan na mobilne podatke SIM kartice).



1. RaspberryPi u ulozi Edge node i Cloud-a

Na slici 1. prikazana su dva Raspberry Pi uređaja, levi uređaj je u ulozi Edge Node-a na koji je povezana Raspberry Pi Camera Module 2 kao i IR senzor detekcije pokreta HC-SR501. Desni Raspberry Pi je u ulozi Clouda.

```cpp
#include <wiringPi.h>
#include <iostream>
#include <cstdlib>
#include <string>
#include <chrono>
#include <thread>
#include <curl/curl.h>
#include <sys/stat.h>
#include <limits.h>

using namespace std;

// PIN konfiguracija
const int pirPin = 7;  // wiringPi 7 = BCM 4

// IP servera
const string server_url = "http://192.168.0.36:5000/upload/";

// Provera da li fajl postoji i da nije prazan
bool file_ready(const string &path) {
    struct stat st;
    return (stat(path.c_str(), &st) == 0 && st.st_size > 0);
}

// Funkcija koja salje fajl preko HTTP POST
bool send_file_http(const string &server_url, const string &file_path) {
    char abs_path[PATH_MAX];
    if (!realpath(file_path.c_str(), abs_path)) {
        cerr << "Ne mogu da dobijem apsolutnu putanju za fajl: " << file_path << endl;
        return false;
    }

    CURL *curl = curl_easy_init();
    if (!curl) return false;

    curl_mime *mime = curl_mime_init(curl);
    curl_mimepart *part = curl_mime_addpart(mime);
    curl_mime_name(part, "file");
    curl_mime_filedata(part, abs_path);

    curl_easy_setopt(curl, CURLOPT_URL, server_url.c_str());
    curl_easy_setopt(curl, CURLOPT_MIMEPOST, mime);
    curl_easy_setopt(curl, CURLOPT_TIMEOUT, 120L);
    curl_easy_setopt(curl, CURLOPT_FOLLOWLOCATION, 1L);

    CURLcode res = curl_easy_perform(curl);
    long http_code = 0;
    curl_easy_getinfo(curl, CURLINFO_RESPONSE_CODE, &http_code);

    curl_mime_free(mime);
    curl_easy_cleanup(curl);

    if (res != CURLE_OK) {
        cerr << "curl_easy_perform() failed: " << curl_easy_strerror(res) << endl;
        return false;
    }
    if (http_code != 200) {
        cerr << "Server returned HTTP code " << http_code << endl;
        return false;
    }

    return true;
}
```

```cpp
int main() {
    if (wiringPiSetup() == -1)
    {
        cerr << "Greska: wiringPi setup nije uspeo!" << endl;
        return 1;
    }

    pinMode(pirPin, INPUT);


    bool recording = false;
    int motion= 0;
    string current_file;

    while (true)
    {
        cout << "Sistem spreman. Cekam pokret..." << endl;
        this_thread::sleep_for(chrono::milliseconds(1000));

        motion = digitalRead(pirPin);

        if (motion && !recording)
        {
            recording = true;
            cout << "Pokret detektovan! Pokrecem snimanje..." << endl;

            // Kreiraj ime fajla sa human-readable timestamp-om
            char buffer[100];
            time_t now = time(nullptr);
            tm *ltm = localtime(&now);
            strftime(buffer, sizeof(buffer), "videos/video_%Y-%m-%d-%H-%M-%S.h264", ltm);
            string filename(buffer);

            current_file = filename;

            // Pokreni snimanje u pozadini, sve logove utisaj
            string cmd = "rpicam-vid --timeout 20000 --width 1920 --height 1080 --framerate 30 -o " + filename + " > /dev/null 2>&1";
            system(cmd.c_str());

            cout << "Snimanje zavrseno: " << filename << endl;

            int tries = 0;
            while (tries < 10 && !file_ready(current_file))
            {
                this_thread::sleep_for(chrono::milliseconds(300));
                tries++;
            }

            if (file_ready(current_file))
            {
                cout << "Saljem fajl na server..." << endl;
                if (send_file_http(server_url, current_file))
                {
                    cout << "Fajl uspesno poslat!" << endl;
                    remove(current_file.c_str()); // obrisi fajl sa diska
                    cout << "Fajl obrisan sa diska." << endl;
                }
                else
                {
                    cerr << "Slanje fajla nije uspelo!" << endl;
                }
            }
            else
            {
                cerr << "Fajl nije spreman ili je prazan: " << current_file << endl;
            }

            recording = false;
            motion = 0;
            delay(10000);
        }

        this_thread::sleep_for(chrono::milliseconds(500));
    }

    return 0;
}
```

2. Program koji se izvršava na Edge Node

Edge Node

Na slici 2. prikazan je program koji se izvršava na Edge Node Raspberry Pi. Program je implementiran pomoću C++ programskog jezika. Njime se detektuje pokret pomoću IR senzora i snima se video pomoću Raspberry Pi kamere.

Na početku programa pozivaju se sve biblioteke čije se funkcije u nastavku koda koriste, definiše se broj PINa na koji je povezan izlaz IR senzora i navodi se IP adresa servera na koji se šalje snimak.

- **bool file_ready(const string &path)** pre slanja snimka funkcija proverava da li je video izgenerisan i sačuvan u memoriji.
- **bool send_file_http(const string &server_url, const string &file_path)** šalje video snimak pomoću HTTP POST zahteva, funkcija iz **libcurl** biblioteke

- Unutar **int main()** implementirana je glavna funkcionalnost programa. Na početku se PIN samog Raspberry-a definiše kao ulazni da bi sa IR senzora stizali podaci. Unutar beskonačne while(true) petlje vrši se detekcija pokreta, generiše ime fajla i pokreće komanda kojom se vrši snimanje ukoliko se detektuje pokret. Po završetku snimka poziva se funkcija **send_file_http** kojom se snimak zatim šalje ka serveru.

U cilju ušetede memorijskog prostora, po slanju snimka on se briše iz memorije.

Da bi program mogao da se pokrene potrebno je instalirati **wiringPi** biblioteku pozivom narednih komandi u terminal

```
1. git clone https://github.com/WiringPi/WiringPi.git
2. cd WiringPi
3. ./build
```

Radi lakšeg pokretanja programa kreiran je Makefile, prvo je potrebno pokrenuti komandu **make,** a zatim **./motion**

```
TARGET = motion
SRC = motion_detection_pt3.cpp
CXX = g++
CXXFLAGS = -Wall
LIBS = -lwiringPi -lcurl –lpthread

$(TARGET): $(SRC)
	$(CXX) $(CXXFLAGS) -o $(TARGET) $(SRC) $(LIBS)
clean:
	rm -f $(TARGET)
```

Cloud

Na Cloudu implementirana su dva programa pomoću Python programskog jezika.

Jedan program predstavlja serversku stranu sistema za detekciju pokreta koji prima snimke od Edge Node-a i šalje notifikacije ka Telegram mobilnoj aplikaciji svaki put kad se detektuje pokret.

Drugi program predstavlja web aplikaciju implementiranu pomoću Streamlit biblioteke kojom je omogućen pregled snimaka koje je Edge Node poslao ka serveru.

Na slici 3. prikazan je Python kod kojim je implementiran prijem snimaka i slanje Telegram notifikacije. Na početku programa uvoze se potrebne bibloteke, kreira instanca FastAPI aplikacije i kreira se folder u koji će se čuvati snimci.

BOT_TOKEN i CHAT_ID generiše se unutar Telegram aplikacije, to su kredencijali kojima se definiše gde će bot slati poruke.

Funkcija **send_telegram_message(message: str)** poziva se kada se šalje poruka ka Telegram aplikaciji.

Radi lakše reprodukcije snimka u web aplikaciji kreirana je funkcija **convert_h264_to_mp4** snimak se konvertuje iz .h264 u .mp4

Pomoću @app.post("/upload/") se definiše da kada server primi POST zahtev na adresu /upload/ da se tada pokrene funkcija upload_file i uzme se fajl koji je klijent poslao. Unutar te funkcije poziva se funkcija za konverziju u mp4 format, šalje se obaveštenje ka Telegram aplikaciji i vraća JSON odgovor ka Edge Node-u.

Da bi program mogao da se pokrene potrebno je instalirati Python virtuelno okruženje i u terminalu pokrenuti komandu **pip install -r requirements.txt**. Nakon toga se pokreće komanda **uvicorn server:app --host 0.0.0.0 --port 5000 –reload**.

```python
# uvicorn server:app --host 0.0.0.0 --port 5000 --reload
from fastapi import FastAPI, File, UploadFile, HTTPException
from fastapi.responses import JSONResponse
import os
import subprocess
import shutil
import requests
from datetime import datetime

app = FastAPI()

MP4_DIR = "uploads_mp4"
os.makedirs(MP4_DIR, exist_ok=True)

# Telegram bot podaci
BOT_TOKEN = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXxx"
CHAT_ID = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXxx"

def send_telegram_message(message: str):
    url = f"https://api.telegram.org/bot{BOT_TOKEN}/sendMessage"
    data = {"chat_id": CHAT_ID, "text": message}
    try:
        response = requests.post(url, data=data)
        response.raise_for_status()
        print("Poruka poslata na Telegram.")
    except requests.exceptions.RequestException as e:
        print(f"Ne mogu poslati poruku na Telegram: {e}")

def convert_h264_to_mp4(h264_path: str) -> str:
    base_name = os.path.splitext(os.path.basename(h264_path))[0]
    mp4_path = os.path.join(MP4_DIR, f"{base_name}.mp4")

    cmd = ["ffmpeg","-y","-i", h264_path,"-c:v", "copy",mp4_path]
    subprocess.run(cmd, check=True)
    return mp4_path

@app.post("/upload/")
async def upload_file(file: UploadFile = File(...)):
    # privremeno sacuvaj .h264
    temp_path = os.path.join(MP4_DIR, file.filename)
    with open(temp_path, "wb") as buffer:
        shutil.copyfileobj(file.file, buffer)

    try:
        # konvertuj u mp4
        mp4_path = convert_h264_to_mp4(temp_path)
        os.remove(temp_path)  # obrisi originalni fajl
    except subprocess.CalledProcessError as e:
        raise HTTPException(status_code=500, detail=f"Konverzija nije uspela: {e}")

    # Posalji Telegram notifikaciju
    timestamp = datetime.now().strftime("%H:%M:%S")
    send_telegram_message(f"Dogodio se pokret u {timestamp}")

    return JSONResponse(content={
        "status": "success",
        "mp4_file": os.path.basename(mp4_path),
        "message": "Fajl uspesno uploadovan i sacuvan kao MP4"
    })
```

3. Cloud server

Na slici 4. prikazan je kod web aplikacije, na početku se uključuje biblioteka Streamlit i folder iz kog se prikazuju snimci. Definiše se naslov i kreira se lista snimaka koji su sortirani po datumu, od najstarijeg do najnovije snimljenog. Kreiran je video player kojim je moguće pustiti odabrani snimak.

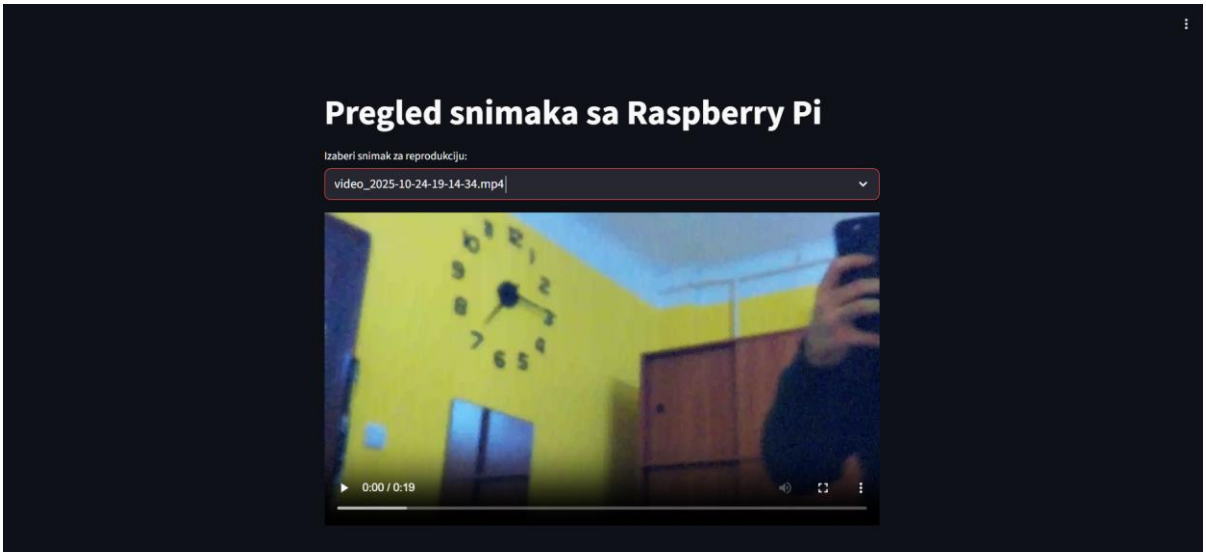
Web aplikacija se otvara tako što se upiše **http://<IP-adresa-uredjaja>:8501**

Na slici 5. prikazan je izgled web aplikacije.

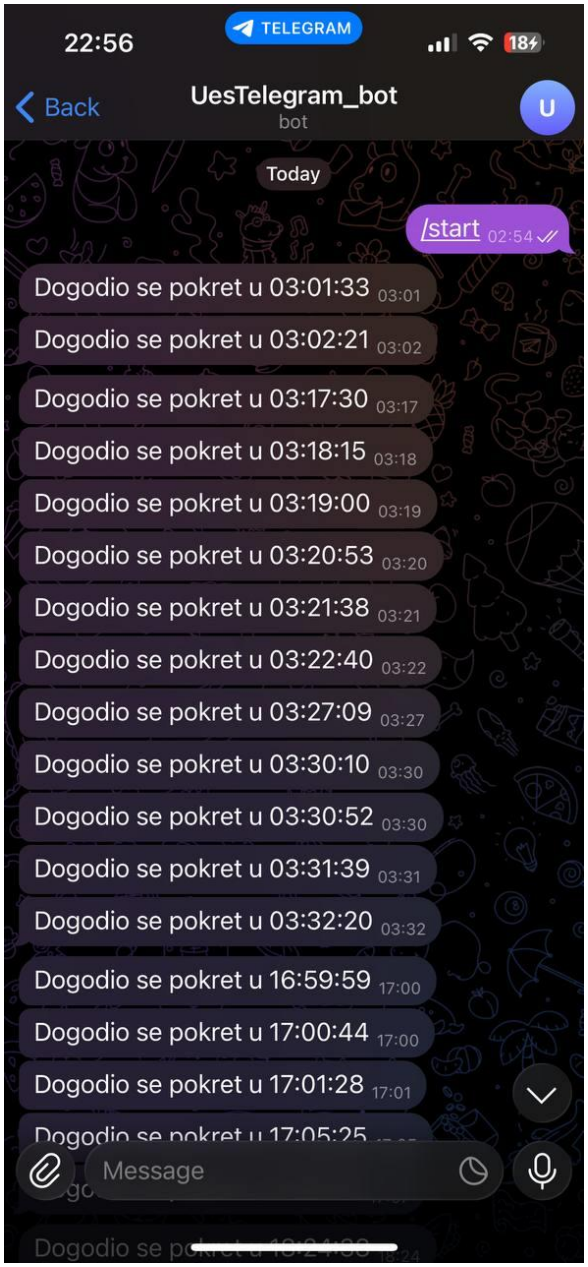
Na slici 6. prikazane su poruke unutar Telegram aplikacije koje je server poslao.

```python
# streamlit run webpage_streamlit.py
import streamlit as st
import os

MP4_DIR = "uploads_mp4"  # folder sa snimcima

st.title("Pregled snimaka sa Raspberry Pi")

# Uzmi sve mp4 fajlove iz foldera
mp4_files = [f for f in os.listdir(MP4_DIR) if f.endswith(".mp4")]

if not mp4_files:
    st.write("Nema snimaka u folderu.")
else:
    # Sortiraj fajlove po datumu kreiranja/modifikacije (od najstarijeg do najmladeg)
    mp4_files.sort(key=lambda f: os.path.getmtime(os.path.join(MP4_DIR, f)))

    # Drop-down lista
    selected_file = st.selectbox("Izaberi snimak za reprodukciju:", mp4_files)

    if selected_file:
        video_path = os.path.join(MP4_DIR, selected_file)
        st.video(video_path)  # reprodukcija video fajla
```

4. Python kod Streamlit web aplikacije



5. Izgled web aplikacije



6. Primljene poruke u Telegram aplikaciji