Poročilo o strukturi projekta in pričakovanih težavah

pri predmetu Sistemska Programska Oprema

Člani skupine:

Simona Zhirova

Marija Jovanova

Nik Terglav

Luka Lamprečnik

Vizualizacija diagnostičnih podatkov o delovanju celotnega sistema

Sistem za upravljanje z gestami je zasnovan kot modularna aplikacija, ki integrira zaznavanje gest, analizo okolja in zaznavanje utrujenosti voznika. Za dosego zanesljive komunikacije, spremljanja zmogljivosti in vizualizacije podatkov sistem uporablja MQTT protokol, Prometheus za zbiranje metrik in Grafano za prikaz podatkov. Celotna arhitektura je implementirana neposredno na operacijskem sistemu Windows brez uporabe Dockerja, kar omogoča prilagodljivo upravljanje odvisnosti in ročno konfiguracijo.

MQTT povezava

Komunikacija med komponentami sistema poteka prek MQTT (Message Queuing Telemetry Transport) protokola, ki omogoča lahkotno in zanesljivo izmenjavo sporočil. MQTT posrednik (broker) je nameščen z uporabo odprtokodnega orodja Eclipse Mosquitto neposredno na gostiteljskem računalniku (z ZeroTier IP-jem 172.25.70.243), ki deluje na vratih 1883 za standardno komunikacijo. Aplikacije, implementirane v Python skriptah, kot so gesture_control_gui.py, detect_env_data.py in drowsiness_detector.py, uporabljajo knjižnico paho-mqtt za povezavo z Mosquitto posrednikom. Povezava je konfigurirana ročno z uporabo ZeroTier omrežja, kjer se aplikacije povezujejo z brokerjem prek naslova 172.25.70.243, ki je dodeljen gostiteljskemu računalniku.

Python skripte na oddaljenih računalnikih, kot je merging_gui.py (npr. z ZeroTier IP-jem 172.25.209.178), pošiljajo in prejemajo MQTT sporočila, razdeljena po temah (topics), kot so gesture/control, environment/data in drowsiness/status, za izmenjavo ukazov in podatkov (npr. "vklopi radio", zaznane vrednosti okolja). Na primer, ko skripta zazna gesto ali drug dogodek, pošlje sporočilo na ustrezen MQTT topic, ki ga druge komponente sprejmejo in obdelajo. Ta arhitektura omogoča asinhrono komunikacijo med moduli prek ZeroTier omrežja, kar povečuje robustnost in prilagodljivost sistema, pri čemer je skupina ročno zagotovila povezljivost z uporabo ukazov, kot je zerotier-cli join 6ab565387ace79b4.

Prenos podatkov do Prometheusa

Za spremljanje zmogljivosti sistema so v vsako Python skripto (gesture_control_gui.py, detect_env_data.py, drowsiness_detector.py, merging_gui.py) integrirane Prometheus metrike. Vsaka skripta izpostavlja metrike prek HTTP strežnika, ki teče na specifičnih vratih: 8003 za merging_gui.py na oddaljenem računalniku, medtem ko so na gostiteljskem računalniku uporabljena vrata 8000 za zaznavanje gest, 8001 za analizo okolja in 8002 za zaznavanje utrujenosti. Metrike, kot so gesture_processing_time (čas obdelave geste), environment_prediction_total (število okoljskih napovedi) in drowsiness_confidence_summary (povprečna zanesljivost zaznavanja utrujenosti), so implementirane z uporabo knjižnice prometheus-client.

Prometheus je nameščen neposredno na en računalnik (gostiteljski z IP 172.25.70.243) in konfiguriran v datoteki prometheus.yml za periodično strganje metrik z naslovov, kot so 172.25.209.178:8003 za merging_gui.py na oddaljenem računalniku ter 172.25.70.243:8000–8002 za ostale skripte na gostiteljskem računalniku. Struktura scrape_configs v prometheus.yml je ročno prilagojena za vsako skripto, organizirana po delovnih mestih (npr. gesture_control, environment_analysis), kar olajša poizvedovanje in analizo.

Prikaz metrik na Grafani

Grafana je nameščena neposredno na gostiteljskem računalniku in dostopna na vratih 3000, kjer služi kot vizualizacijsko orodje za prikaz Prometheus metrik. Grafana je povezana s Prometheusi, nameščenim na gostiteljskem računalniku, kot podatkovnim virom, kar omogoča dinamično poizvedovanje metrik prek ZeroTier omrežja. Posodobljena nadzorna plošča, definirana v JSON formatu, vključuje panele za vse ključne komponente sistema:

- **Zaznavanje gest**: Prikaz metrik, kot so število napovedi po modelu (gesture_model_predictions_total), povprečna zanesljivost gest (gesture_confidence_summary) in čas obdelave gest (gesture_processing_time).
- **Analiza okolja**: Vizualizacija števila okoljskih napovedi (environment_prediction_total), povprečne zanesljivosti (environment_confidence_summary) in časa obdelave slik (image_processing_time).
- **Zaznavanje utrujenosti**: Prikaz skupnega števila napovedi (drowsiness_prediction_total), zanesljivosti zaznavanja (drowsiness_confidence_summary) in časa obdelave okvirjev (frame_processing_time).
- **Glavna aplikacija**: Spremljanje izvajanja skript (script_execution_total) in napak (script_execution_errors) za merging_gui.py.

Nadzorna plošča vključuje različne tipe prikazov, kot so časovne vrste, merilniki in stolpični diagrami, kar omogoča celovit vpogled v delovanje sistema. Dodana je tudi spremenljivka component za filtriranje po posameznih delovnih mestih, kar povečuje prilagodljivost analize. JSON datoteka za nadzorno ploščo je bila ročno posodobljena z dodatnimi paneli za MQTT specifične metrike, kot so obremenitev CPU, poraba pomnilnika, število sporočil in obdelana sporočila, kar smo uresničili z ročnim urejanjem datoteke dashboard.json.

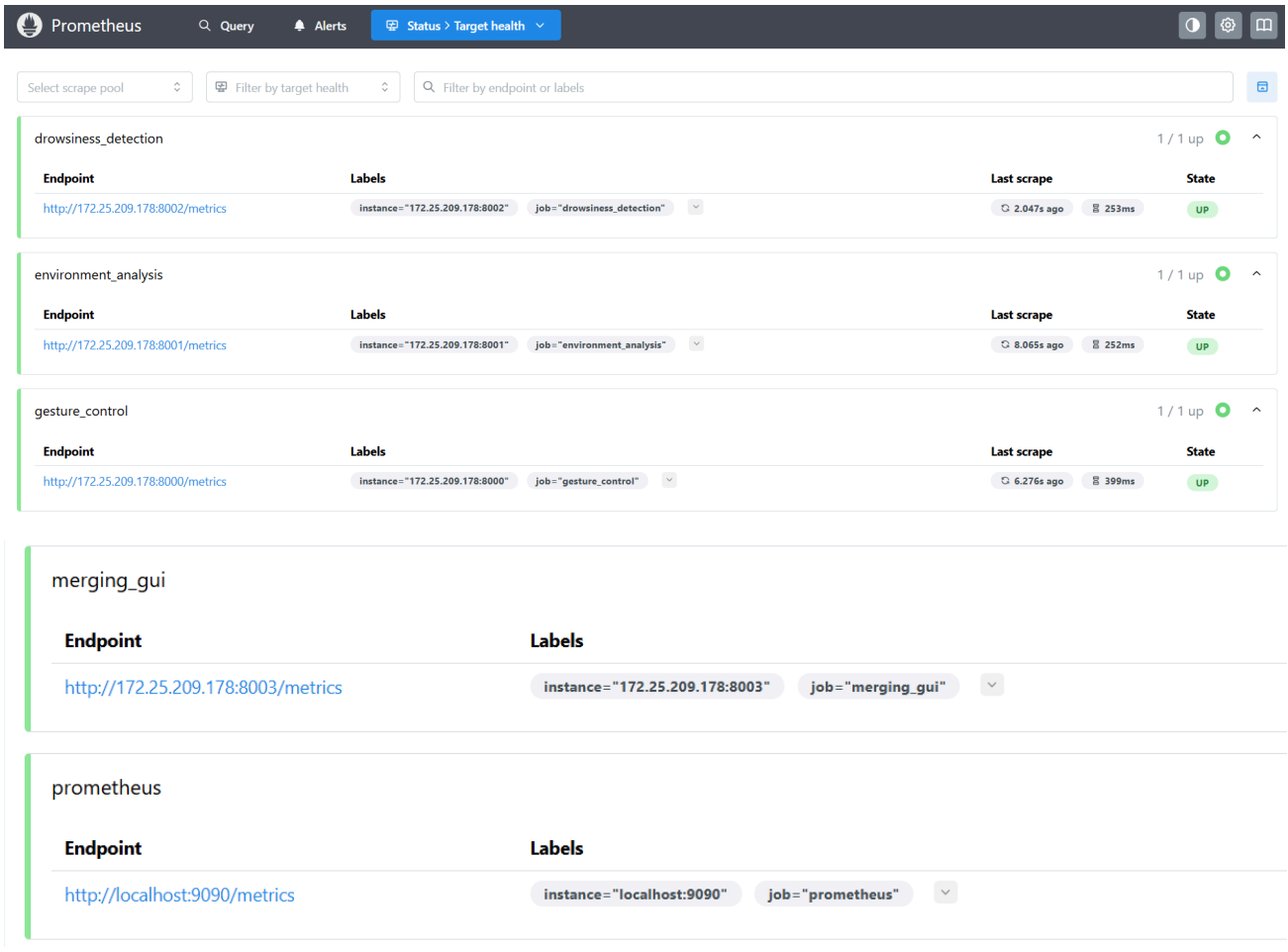
Postopek ročne namestitve in konfiguracije

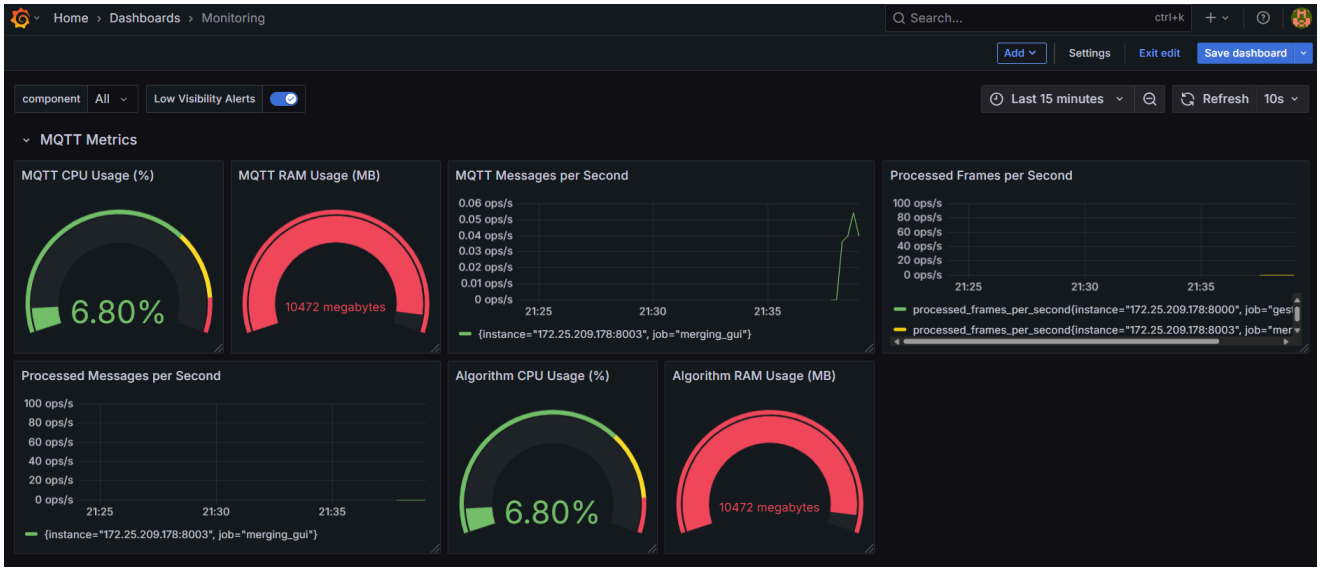
Sistem je bil ročno nameščen in konfiguriran na računalnikih s strani štiričlanske skupine. Na gostiteljskem računalniku (172.25.70.243) smo namestili Mosquitto z uporabo izvedljive datoteke mosquitto.exe, Prometheus z izvedljivo datoteko prometheus.exe in Grafano z grafana-server.exe, pri čemer smo prilagodili konfiguracijske datoteke (npr. prometheus.yml) za delovanje prek ZeroTier omrežja. Na oddaljenih računalnikih ostalih članov smo namestili samo potrebne Python skripte (npr. merging_gui.py) in jih konfigurirali za povezavo z

gostiteljskim Mosquittom na 172.25.70.243:1883. ZeroTier omrežje (ID 6ab565387ace79b4) je bilo ročno vzpostavljeno z ukazom zerotier-cli join, kar je omogočilo povezljivost med 172.25.70.243 in npr. 172.25.209.178. Firewalls na obeh računalnikih so bili ročno prilagojeni z netsh ukazi za odpiranje vrat 1883 (MQTT), 8000–8003 (metrike) in 9090 (Prometheus).

Testiranje sistema je potekalo z ročnim zagonom skript in preverjanjem povezav z ukazi, kot so ping 172.25.70.243, curl http://localhost:8003/metrics na oddaljenem računalniku in mosquitto_pub/sub za preverjanje MQTT komunikacije. Napake, kot je bil prvotno down status Prometheusa na oddaljenem računalniku, smo odpravili z ročnim urejanjem prometheus.yml na gostiteljskem računalniku in zagotavljanjem, da so vrata odprta. Mosquitto dnevniki (mosquitto.log) so potrdili uspešno povezavo z oddaljenimi računalniki, kar je omogočilo nadaljnje testiranje in optimizacijo. Skupina je dodala odpravljanje težav, kot je bila prilagoditev neskončne zanke v merging_gui.py z uvedbo periodičnih posodobitev metrik prek metode update_metrics, ter zagotovila, da se metrike pravilno izpostavljajo in strgajo v Prometheusu na gostiteljskem računalniku.

Rezultati


Prometheus
Query
Alerts
Status > Target health
Select scrape pool
Filter by target health
Filter by endpoint or labels
drowsiness_detection
1 / 1 up
Endpoint
Labels
Last scrape
State
http://172.25.209.178:8002/metrics
instance="172.25.209.178:8002"
job="drowsiness_detection"
2.047s ago
253ms
UP
environment_analysis
1 / 1 up
Endpoint
Labels
Last scrape
State
http://172.25.209.178:8001/metrics
instance="172.25.209.178:8001"
job="environment_analysis"
8.065s ago
252ms
UP
gesture_control
1 / 1 up
Endpoint
Labels
Last scrape
State
http://172.25.209.178:8000/metrics
instance="172.25.209.178:8000"
job="gesture_control"
6.276s ago
399ms
UP
merging_gui
Endpoint
Labels
http://172.25.209.178:8003/metrics
instance="172.25.209.178:8003"
job="merging_gui"
prometheus
Endpoint
Labels
http://localhost:9090/metrics
instance="localhost:9090"
job="prometheus"


Home > Dashboards > Monitoring
Search...
ctrl+k
Add
Settings
Exit edit
Save dashboard
component
All
Low Visibility Alerts
Last 15 minutes
Refresh
10s
MQTT Metrics
MQTT CPU Usage (%)
6.80%
MQTT RAM Usage (MB)
10472 megabytes
MQTT Messages per Second
0.06 ops/s
0.05 ops/s
0.04 ops/s
0.03 ops/s
0.02 ops/s
0.01 ops/s
0 ops/s
21:25
21:30
21:35
{instance="172.25.209.178:8003", job="merging_gui"}
Processed Frames per Second
100 ops/s
80 ops/s
60 ops/s
40 ops/s
20 ops/s
0 ops/s
21:25
21:30
21:35
processed_frames_per_second{instance="172.25.209.178:8000", job="ges
processed_frames_per_second{instance="172.25.209.178:8003", job="mer
Processed Messages per Second
100 ops/s
80 ops/s
60 ops/s
40 ops/s
20 ops/s
0 ops/s
21:25
21:30
21:35
{instance="172.25.209.178:8003", job="merging_gui"}
Algorithm CPU Usage (%)
6.80%
Algorithm RAM Usage (MB)
10472 megabytes