Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 621.38
DOI: https://doi.org/10.24867/13BE02Savic

INTEGRACIJA MAINFLUX PLATFORME U SISTEM ZA UPRAVLJANJE ENERGETSKOM POTROŠNJOM U PAMETNOJ KUĆI

INTEGRATION OF MAINFLUX PLATFORM INTO THE SMART HOUSE ENERGY CONSUMPTION SYSTEM

Filip Savić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – VAŠ ODSEK

Kratak sadržaj – *Zadatak rada jeste integracija Mainflux platforme u sistem za upravljanje energetskom potrošnjom u pametnoj kući. Mainflux platforma je korišćena zajedno sa Bash scripting-om da obezbijedi dostupnost podataka potrebnih za generisanje realne potrošnje električne energije. Simulacija je rađena pomoću Typhoon HIL softvera. Python programski jezik je korišćen za dobavljanje podataka sa Mainflux-a, stavljanje podataka u Typhoon HIL simulaciju, te upis realne energetske potrošnje u CSV datoteku.*

Ključne reči: *Mainflux, Typhoon HIL, pametna kuća, Python, IoT*

Abstract – *This paper deals with the integration of the Mainflux platform into a smart house energy counsumption system. Mainflux platform was used together with Bash scripting to provide the necessary data for more real energy consumption simulation. The simulation is done using Typhoon HIL software. Python programming language has been used for software support, as a main integration component. It fetches data from the Mainflux platform, puts it into the Typhoon HIL simulation, and at the end writes the generated energy estimation in a CSV file.*

Keywords: *Mainflux, Typhoon HIL, smart home, IoT*

1. UVOD

Zadatak rada predstavlja integracija Mainflux platforme sa TyphoonHIL simulatorom u cilju ubacivanja realnih podataka o temperaturi i solarnoj iradijaciji u Typhoon HIL simulaciju pametne kuće (eng. „smart home“). Ovaj rad predstavlja dio sistema koji softverskim agentima povezanim na realne pametne uređaje ima za cilj da smanji energetsku potrošnju u pametnoj kući.

Prvenstveno je bilo potrebno pribaviti i instalirati neophodan softver za korišćenje Mainflux [1, 4] platforme i Typhoon HIL Control Center-a [2].

2. OPIS KORIŠĆENIH TEHNOLOGIJA

Konačno softversko rješenje kreirano je uz pomoć Typhoon HIL Control Center-a i Typhoon HIL API-ja, Mainflux platforme čije su komponente pokretane pomoću Docker [3] softverskog alata, te upotrebom Bash scripting-a i programskog jezika Python. Podaci o temperaturi i iradijaciji preuzeti su sa sajta australijske kompanije Solcast [5].

Skripta za pribavljanje podataka sa Mainflux platforme i upravljanje Typhoon HIL simulacijom pisane su u programskom jeziku Python. Typhoon-HIL-API [6] je Python modul koji podržava upravljanje simulacijom.

2.1 Mainflux platforma

Mainflux predstavlja skalabilnu, bezbjednu, patent-free i open-source platformu za Internet of Things, napisanu u programskom jeziku Go, koja se distribuira putem softverskih paketa („kontejnera“) Docker tehnologije.

Glavne karakteristike Mainflux-a su:

- brza i skalabilna mikro-servisna arhitektura,
- dobro formiran API (HTTP, MQTT, WebSocket),
- interakcija sa uređajima kroz različite protokole i
- lako pokretanje i distribucija putem Docker alata.

Ove karakteristike omogućavaju brz i jednostavan razvoj pametnih IoT rješenja.

U daljem tekstu će se, zbog jednostavnosti, za Mainflux platformu navoditi samo Mainflux.

Mainflux predstavlja skup servisa od kojih svaki ima važnu ulogu:

- Users – upravljanje korisnicima i autentifikacijom,
- Things – registrovanje novih uređaja i aplikacija, pravljenje komunikacionih kanala između njih, te kontrola komunikacije,
- Protocol adapters – (HTTP, MQTT, WebSocket) adapteri koji obezbjeđuju interfejse za pristup komunikacionim kanalima,
- Normalizer – normalizacija SenML [7] poruka i slanje istih ka tokovima za dalju obradu podataka i
- Custom consumers – svi servisi koji vrše upis u skladišta podataka (eng. „data store“) i čitanje normalizovanih poruka kroz HTTP API.

Tri glavna entiteta kojima Mainflux upravlja su:

1. Thing – stvar – predstavlja sve entitete koji komuniciraju kroz Mainflux,
2. Channel – kanal – predstavlja komunikacioni kanal. Služi kao tema razmjene poruka (eng. „message topic“)
3. User – korisnik – pravi korisnik sistema (čovjek). Predstavljen je svojom adresom e-pošte i lozinkom.

Messaging – razmjena poruka

Mainflux koristi NATS [8] kao glavni stub komunikacije, zbog svojih performansi i malog zauzeća resursa.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Milan Vidaković, red. prof.

Mainflux nudi više protokola za razmjenu poruka, od kojih su u realizaciji sistema korišćeni HTTP i MQTT. O MQTT-u i SenML-u će biti nekoliko riječi, a HTTP neće biti opisivan zbog svoje rasprostranjenosti.

SenML

SenML (Sensor Markup Language) specifikacija definiše standardizovan format poruka za opisivanje senzora i njihovih mjerenja. Poruke mogu biti u mnogim formatima, od kojih je u projektu, zbog konciznosti i rasprostranjenosti, korišćen JSON (JavaScript Object Notation). Primjer SenML poruke dat je u Listingu 1.

```
[ {"n":"temperature","u":"Cel","v":23.1} ]
```

Listing 1. *Primjer SenML poruke*

Vrijednosti „n“, „u“ i „v“ predstavljaju naziv uređaja, jedinicu, te vrijednost izmjerene veličine, redom. Svako mjerenje se, makar bilo i samo jedno, mora nalaziti unutar liste, koja se u JSON notaciji označava uglastim zagradama - [ ].

MQTT

MQTT (Message Queuing Telemetry Transport) predstavlja M2M (Mahcine to Machine) komunikacioni protokol. Dizajniran je za podršku izuzetno lakog (eng. „lightweight“) transporta poruka po modelu „publish/ subscribe“. Neki od primjera upotrebe su kod senzora koji komuniciraju sa brokerom putem satelitske veze ili u raznim scenarijima automatizacije domaćinstava i malih uređaja.

Docker

Docker je tehnologija za kreiranje softverskih paketa, odnosno „kontejnera” za samostalne aplikacije. U njima se nalaze biblioteke i sve ostalo potrebno za pokretanje i izvršavanje aplikacije.

Docker kontejneri omogućavaju „pakovanje” aplikacije i svega što joj je potrebno za rad u „sliku“ kontejnera (eng. „container image“).

„Slika kontejnera“ je šablon za izvršavanje kontejnera. Nakon što se kontejner kreira, njegovim izvršavanjem upravlja „Docker Engine“, poznat i pod nazivom „Docker Daemon“. Interakcija sa Docker Engine-om omogućena je putem komande „docker“, odnosno interfejsa komandne linije (eng. „command line interface“, CLI).

Instalacija i pokretanje Mainflux-a

Za postupak instalacije koji je opisan u nastavku, potreban je softver cURL i git CLI. Da bi instalirali platformu, prvo je potrebno preuzeti i instalirati Docker i docker compose [9], koji služi za pokretanje aplikacija sastavljenih iz više Docker kontejnera.

Kada su Docker i docker-compose instalirani, moguće je preuzeti i pokrenuti Mainflux platformu komandama sadržanim u listingu 2.

```
> git clone https://github.com/mainflux/mainflux.git
> cd mainflux
> docker-compose -f docker/docker-compose.yml -f
docker/addons/influxdb-reader/docker-compose.yml -f
docker/addons/influxdb-writer/docker-compose.yml up -d
```

Listing 2. *Komande za instalaciju Mainflux platforme*

Mainflux CLI

Mainflux CLI [10] predstavlja alat za upravljanje korisnicima, stvarima, kanalima i porukama. Alat je moguće preuzeti zasebno sa github stranice izdanja (eng. „releases“) preuzeti tar.gz datoteku te je otpakovati u željenom direktorijumu. Listing 3 prikazuje set komandi za ovaj tip instalacije.

```
> wget
https://github.com/mainflux/mainflux/releases/downloa
d/v0.11.0/mainflux-cli_v0.11.0_linux-amd64.tar.gz
> tar xvf mainflux-cli_v0.11.0_linux-amd64.tar.gz
```

Listing 3. *Komande za ručno preuzimanje i instalaciju Mainflux CLI alata*

Nakon instalacije moguće je koristiti Mainflux CLI ako je pokrenut glavni Mainflxux servis. Komande korišćene u ovom projektu su `messages` za poruke, `things` za upravljanje stvarima te `users` za upravljanje korisnicima.

2.2. Bash scripting

Bash scripting [11] je korišćen za skriptu koja kreira čitač CSV datoteke (Mainflux-ov CSV Reader). Skripta se koristi za pravljenje Mainflux entiteta koji su potrebni za pokretanje reader-a. Kreiraju se korisnik, kanal i stvar koja predstavlja sam čitač.

Bash skripta predstavlja datoteku koja sadrži set komandi koje se izvršavaju u komandnoj liniji (eng. „command line“). Bilo koja komanda koja se stavi u skriptu radi potpuno istu stvar kao i kada se samostalno pokrene u komandnoj liniji. Neke od osnovnih komandi su echo za ispis u konzolu, if za kontrolu toka programa te for za programske petlje. Svaka bash skripta u kao prvu liniju shebang (#!) koji označava da slijedi putanja do interpetera ili programa koji će izvršiti skriptu. Putanja do interpretera je obično /bin/bash.

2.3. Typhoon HIL Control Center

Typhoon HIL Control Center [12] predstavlja softverski alat kompanije Typhoon HIL za inženjering sistema zasnovanih na modelu (eng. „Model-Based System Engineering“). Sastoji se od četiri modula: Schematic Editor, HIL SCADA, Typhoon Test IDE i Test Suite. U ovom zadatku Control Center je korišćen za simulaciju pametne kuće i energetske potrošnje u njoj.

2.4. Python i upravljanje simulacijom

Za upravljanje Typhoon HIL simulacijom korišćena je Python biblioteka Typhoon-HIL-API. Pruža razne funkcije koje su podijeljene u četiri grupe: za kontrolu inicijalizacije, za mijenjanje stanja, za preuzimanje informacija iz simulacije te razne Utility funkcije. Listing 4 prikazuje Python skriptu koja uvozi biblioteku, učitava model, pokreće simulaciju te postavlja neke vrijednosti unutar simulacije.

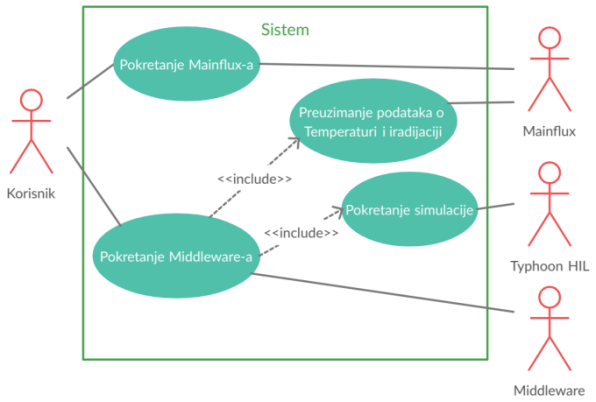
```
import typhoon.api.hil as hil
hil.load_model(file=r'/home/filip/master/typhoon_hil/
Target files/model.cpd')
hil.start_simulation()
hil.set_analog_output(5,'V(Va)',scaling=150.0,offset=
5.0)
```

Listing 4. *Osnovne komande Typhoon HIL API-ja*

3. SPECIFIKACIJA APLIKACIJE

Zadatak predstavlja integraciju Mainflux-a sa Typhoon HIL simulatorom. Cilj ovog rada jeste nadzor, mjerenje i upravljanje potrošnjom električne energije u pametnoj kući. Pošto su trenutno vrijednosti temperature i iradijacije u simulaciji generisane, učitavanje realnih podataka omogućava bolju simulaciju energetske potrošnje.

Podaci o temperaturi i solarnoj iradijaciji ubačeni su u Mainflux iz CSV datoteke, a učitavanje tih podataka u simulaciju realizovano je korišćenjem Python skripte. Slika 1 prikazuje Use case dijagram sistema.



Slika 1. *Use case dijagram*

4. OPIS IMPLEMENTACIJE

Za potrebe zadatka korišćeni su Mainflux platforma, programski jezik Python te Typhoon HIL API. Mainflux je preuzet sa github stranice Mainflux projekta, a pokretan je docker i docker compose alatima. Integracija je urađena koristeći Python. Typhon HIL softver za Linux operativni sistem dostavljen je od strane zaposlenih kompanije Typhoon HIL dok je biblioteka Typhoon HIL API preuzeta je sa PyPI repozitorijuma.

4.1. Odabir podataka

Podaci o temperaturi i iradijaciji preuzeti su sa sajta Solcast kompanije.Sa stranice „Historical & TMY“ (eng. „Typical meteorological year“) web sajta Solcast kompanije moguće je preuzeti podatke o temperaturi i solarnoj iradijaciji. Preuzimanje podataka vrši se kreiranjem zahtjeva. Potrebno je unijeti lokaciju za koju nas podaci interesuju te označiti period u godinama ili od jednog do drugog konkretnog datuma. Zatim se biraju parametri kao što su horizontalna iradijacija, azimut, temperatura vazduha, brzina vjetra itd. Podatke je moguće preuzeti u intervalima od po 5, 10, 15, 30 i 60 minuta. Za svaki skup podataka preuzima se odvojena CSV datoteka.

4.2. Uloga Mainflux-a

Mainflux je u ovom zadatku služio kao repozitorijum podataka te lako proširivo rješenje u slučaju prelaska na realan sistem pametne kuće. Čuva podatke o temperaturi i iradijaciji i čini ih dostupnim za dobavljanje preko interneta. Glavni doprinos Mainflux-a za ovaj zadatak jeste CSV čitač (eng. „reader”). CSV reader se kreira kao Mainflux stvar prikačena na Mainflux kanal. Da bi se kreirao bilo koji Mainflux entitet, potrebno je da postoji korisnik. Za potrebe testiranja rada kreirano je mnogo korisnika, kanala i reader-a, te je proces automatizovan u jednu bash skriptu koja zbog svoje veličine neće biti prikazana u ovom radu. Skripta kreira korisnika, uzima njegov token, te ga prosljeđuje prilikom kreiranja kanala. Zatim korisnikov token i ID kreiranog kanala šalje u JSON za kreiranje CSV reader-a kao Mainflux stvari. JSON osim toga sadrži i putanju do CSV datoteke te kolone koje se čitaju iz nje.

Da bi se kreirani CSV reader pokrenuo, potrebno je preuzeti kôd za CSV reader preuzima se sa github naloga Darka Draškovića [13]. Zatim je potrebno pokrenuti Mainflux core (glavni servis) i Mongo writer koji predstavlja Mainflux dodatak za skladištenje (eng. „storage“) podataka. Nakon što se pokrenu ovi servisi, može se pokrenuti i sam reader. U terminalu se treba navigirati do csv-dbreader direktorijuma u Mainflux-u sa github-a Darka Draškovića, te pokrenuti komande prikazane na listingu 5.

```
export MF_CSV_ADAPTER_CONFIG_FILE =
/home/filip/Faks/Master/readers.cfg
go run main.go
```

Listing 5. *Pokretanje CSV reader-a*

Prva komanda postavlja putanju do konfiguracijone datoteke za reader-e „readers.cfg“, a druga pokreće reader pisan u jeziku Go. Nakon ovoga u terminalu u kome je pokrenut Mongo writer biće prikazano mnoštvo linija koje su indikator da je sve iz CSV datoteke upisano u bazu. Nakon ovoga treba samo pokrenuti Mongo reader koji će omogućiti preuzimanje podataka iz baze preko API-ja.

4.3. Uloga Typhoon HIL-a

U ovom zadatku korišćen je Typhoon HIL Control Center te Typhoon HIL API. Typhoon HIL se koristio za simuliranje pametne kuće i energetske potrošnje unutar nje, ali su podaci o temperaturi i iradijaciji bili generisani. Bilo je potrebno izmijeniti inicijalni model pametne kuće dobijen od strane zaposlenih kompanije Typhoon HIL da bi se u njega mogli ubaciti podaci o temperaturi i iradijaciji. Ovo je urađeno u Schematic Editor-u korišćenjem SCADA Input komponente koja ima mogućnost da joj se programski mijenja vrijednost, čak i tokom trajanja simulacije.

4.4. Kreiranje Middleware-a

Nakon završene pripreme Mainflux i Typhoon HIL strana za komunikaciju, sve što je potrebno za ubacivanje podataka jeste GET zahtjev na Mainflux-ov Mongo Reader za dobavljanje podataka, te ubacivanje dobijenih podataka u simulaciju preko SCADA Input-a. Slijedi listing 6 koji prikazuje kôd middleware-a.

```
import requests, json
import typhoon.api.hil as hil
TEMPERATURE_FIELD = 'AirTemp'
IRRADIATION_FIELD = 'Ghi'
CHANNEL_ID = 'e9880be5-44e5-4739-848a-5aee83c99b12'
THING_KEY = 'ae3fc247-e790-483b-b588-e3abd003fb22'
MSGS_PER_CSV_ROW = 6
compiled_model_path =
'/home/filip/Faks/Master/TyphoonHIL/modeli/house
Target files/house.cpd'
model_loaded = hil.load_model(compiled_model_path,
vhil_device=True)
message_row_number = 0
while True:
```

```
  OFFSET = message_row_number * MSGS_PER_CSV_ROW

  url =
'http://localhost:8904/channels/'+CHANNEL_ID+'/messag
es?limit='+str(MSGS_PER_CSV_ROW)+'&offset='+str(OFFSE
T)
  headers = {'Authorization': THING_KEY}
  req = requests.get(url, headers=headers)
  json_response = json.loads(req.text)
  if json_response["messages"] == []:
    break

  for message in json_response["messages"]:
    if message["subtopic"]==TEMPERATURE_FIELD:
      temperature = message["value"]
    if message["subtopic"]==IRRADIATION_FIELD:
      irradiation = message["value"]

  hil.set_scada_input_value("TemperatureInput",
value = temperature)
  hil.set_scada_input_value("IrradiationInput",
value = irradiation)
  hil.start_simulation(var_sim_define_step =
hil.get_sim_step() * 15)
  P_grid_net = hil.read_analog_signal( name =
"P_grid_net")
  message_row_number += 1
```

Listing 6. *Programski kôd Middleware-a*

Prve linije listinga prikazuju komande učitavanje biblioteka za slanje zahtjeva, rad sa JSON-om te Typhoon HIL API-ja. Nakon toga slijede definicije konstanti koje predstavljaju nazive kolona u CSV datoteci, tj. podtemu (eng. „subtopic“) na Mainflux kanalu pod kojom su temperatura i iradijacija upisane u bazu podataka. Konstanta `MSGS_PER_CSV_ROW` predstavlja broj poruka koje su upisane u bazu za svaki red CSV datoteke, tj. za svaki period od 15 minuta. Sljedeće dvije linije listinga predstavljaju putanju do kompajliranog modela te programsko učitavanje modela. Lokalna promjenljiva `message_row_number` predstavlja redni broj reda CSV datoteke koji čitamo.

Nakon toga slijedi while petlja u kojoj se učitavaju podaci, postavljaju vrijednosti u simulaciju, te preuzima energetska potrošnja. Konstanta `OFFSET` utiče na to od koje poruke u bazi će početi čitanje, dok se prethodno definisana konstanta `MSGS_PER_CSV_ROW` koristi kao ograničenje koliko poruka se preuzima sa Mainflux-a u jednoj iteraciji petlje, što je u ovom slučaju 6.

Slijedi definisanje URL-a te formiranje zaglavlja postavljanjem ključa stvari, samog CSV reader-a, za autorizaciju zahtjeva ka Mongo Reader-u. Promjenljiva `req` predstavlja rezultat poziva *get* metode biblioteke requests kojoj se proslijede prethodno definisani URL te zaglavlje za autorizaciju.

Iz odgovora se uzima tekstualna reprezentacija i pretvara u JSON pomoću metode *loads*. Nakon linije za uslov prekida while petlje, a to je da smo došli do posljednjeg upisa u bazi, slijedi uzimanje vrijednosti temperature i iradijacije iz poruka pomoću ključa value.

Nakon ovoga se, pomoću Typhoon HIL API-ja, dobijene vrijednosti postavljaju u simulaciju koja se pokreće te vraća energetsku potrošnju.

5. ZAKLJUČAK

Ovaj rad predstavio je integraciju Mainflux platforme sa Typhoon HIL simulatorom u cilju ubacivanja realnih podataka o temperaturi i solarnoj iradijaciji u simulaciju radi tačnijeg generisanja potrošnje električne energije. Rad doprinosi rješenju šireg opsega koje se bavi pravljenjem i upotrebom softverskih agenata koji se trenutno povezuju na simulirane pametne uređaje, a cilj je da rade i sa realnim, te upravljaju njihovim ponašanjem.

Kako je rješenje predstavljeno u ovom radu dio većeg rješenja, unapređenje bi bilo stvaranje jednostavnog API-ja za pokretanje Middleware-a preko interneta.

Takođe, unapređenje unutar samog sistema bilo bi automatizacija pokretanja svih komponenata. Mogle bi se napraviti dvije Bash skripte, jedna za pokretanje Mainflux-a za komunikaciju sa Middleware-om, druga za pokretanje svega potrebnog za kreiranje CSV reader-a.

6. LITERATURA

[1] Mainflux platforma, https://www.mainflux.com/ (pristupljeno u oktobru 2020.)

[2] Typhoon HIL Control Center, https://www.typhoon-hil.com/products/hil-software/ (pristupljeno u oktobru 2020.)

[3] Docker, https://www.docker.com/ (pristupljeno u oktobru 2020.)

[4] Mainflux github, https://github.com/mainflux/mainflux (pristupljeno u oktobru 2020.)

[5] Solcast, https://solcast.com/ (pristupljeno u oktobru 2020.)

[6] Typhoon HIL API, https://www.typhoon-hil.com/documentation/typhoon-hil-api-documentation/hil_api.html (pristupljeno u oktobru 2020.)

[7] SenML, https://tools.ietf.org/html/draft-ietf-core-senml-13 (pristupljeno u oktobru 2020.)

[8] NATS, https://nats.io/documentation/ (pristupljeno u oktobru 2020.)

[9] Docker compose, https://docs.docker.com/compose/ (pristupljeno u oktobru 2020.)

[10] Mainflux CLI, https://mainflux.readthedocs.io/en/latest/cli/ (pristupljeno u oktobru 2020.)

[11] Bash scripting, https://help.ubuntu.com/community/Beginners/BashScripting (pristupljeno u oktobru 2020.)

[12] Typhoon HIL Control Center, https://www.typhoon-hil.com/products/hil-software/ (pristupljeno u oktobru 2020.)

[13] Github Darka Draškovića, https://github.com/darkodraskovic/mainflux/tree/dbreader/cmd/csv-dbreader (pristupljeno u oktobru 2020.)

Kratka biografija:

Filip Savić rođen je u Šapcu 1995. god. Master rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva – Softversko inženjerstvo i informacione tehnologije odbranio je 2020. god.

kontakt: filipsavic1995@yahoo.com