SADRŽAJ

Popis slika **vi**

Popis isječaka koda **vi**

1. Uvod **1**

2. Opis sustava i tehničkih zahtjeva **2**

2.1. Opis problematike u poljoprivredi - WIP naslov potpoglavlja 2

2.2. Zahtjevi na sustav i opis predloženog rješenja 2

3. Razvojni sustav ESP32-C3-DevKitM-1 **3**

3.1. Wi-Fi 4

4. Amazon Web Services (AWS) **9**

4.1. AWS IoT Core 11

4.1.1. AWS IoT Fleet Provisioning 14

4.1.2. AWS IoT Jobs 18

4.1.3. AWS IoT OTA 18

4.1.4. AWS IoT Device Shadow 18

4.1.5. Ostale dostupne IoT usluge u sustavu AWS 18

5. Povezivanje razvojnog sustava i oblaka **21**

5.1. Programska potpora za mikrokontroler 21

5.1.1. Dinamičko povezivanje mikrokontrolera na Wi-Fi 22

5.1.2. Registracija u sustav AWS 27

5.1.3. Ažuriranje softvera 31

5.1.4. Očitavanje senzorskih mjerenja 31

5.1.5. Slanje očitanih podataka protokolom MQTT 31

5.2. Programska potpora za oblak 31

5.2.1. Dinamička registracija uređaja 31

6. Web aplikacija **34**

6.1. Infrastruktura aplikacije 34

6.2. Grafana 34

7. Zaključak **35**

Literatura **36**

POPIS SLIKA

2.1. Blok shema sustava 2

3.1. Konfiguracija razvojnog sustava ESP32-C3-DevKitM-1 [10] 3
3.2. Blok dijagram modula ESP32-C3 [8] 4
3.3. Wi-Fi RF standardi [9] 6
3.4. Primjer scenarija Wi-Fi povezivanja u načinu rada stanice [10] 7
3.5. Primjer scenarija Wi-Fi povezivanja u načinu rada pristupne točke [10] 7

4.1. Arhitektura usluga AWS-a za IoT [3] 10
4.2. Princip rada usluge AWS IoT Core [3] 11
4.3. Komponente usluge AWS IoT Core [3] 12

5.1. Arhitektura unificiranog provizioniranja [7] 23
5.2. Obavijest nakon skeniranja QR koda 28
5.3. Odabir dostupne Wi-Fi mreže u blizini 28
5.4. Tok registracije uređaja certifikatom zahtjeva [3] 29
5.5. Popis politika dodijeljenih registriranom uređaju 33

POPIS ISJEČAKA KODA

4.1. Odjeljak *parametri* u predlošku za registraciju 15
4.2. Odjeljak *resursi* u predlošku za registraciju 16
5.1. Stvaranje pristupne točke 24
5.2. Generiranje QR koda iz pristupne točke 25
5.3. Funkcija za prikaz QR koda na zaslonu 26
5.4. Spajanje certifikatom zahtjeva i zahtjev za novim certifikatom 30
5.5. Odjeljak *stvar* u predlošku za registraciju 32

1. Uvod

Moj uvod.

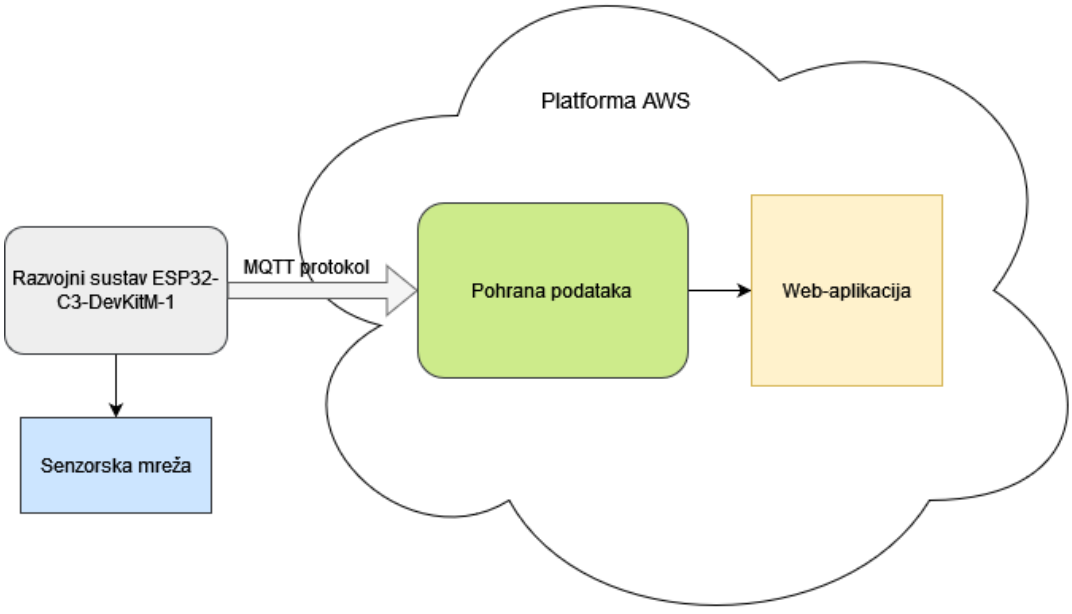
2. Opis sustava i tehničkih zahtjeva

2.1. Opis problematike u poljoprivredi - WIP naslov potpoglavlja

Moj problematični poljoprivredni sustav. Isto tako, možeš naći neke zanimljive radove i neka zanimljiva rješenja pa ih ovdje malo polinkati. Mislim da bi to bilo dosta kul.

2.2. Zahtjevi na sustav i opis predloženog rješenja

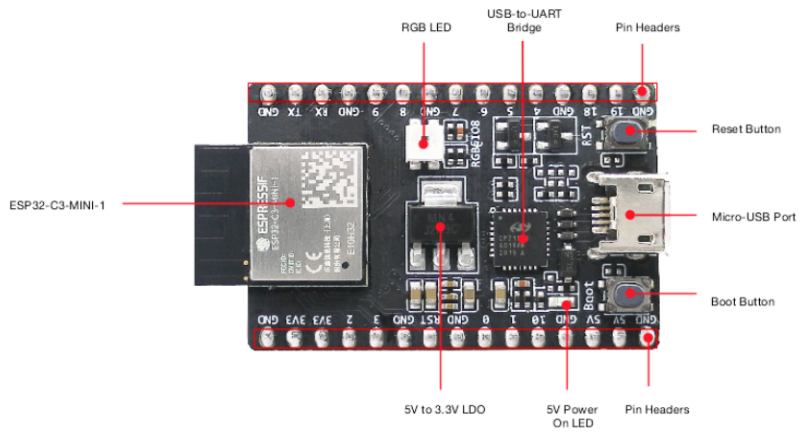
Ovdje bih mogla napraviti skicu sustava kao i prije 2 godine kako bih lijepo prikazala što želim postići svojim sustavom. Sad sam napravila probnu shemu, no kasnije bih mogla napraviti malo bolju, ovisno o tome kako ću na kraju projektirati sam rad.



Slika 2.1: Blok shema sustava

3. Razvojni sustav ESP32-C3-DevKitM-1

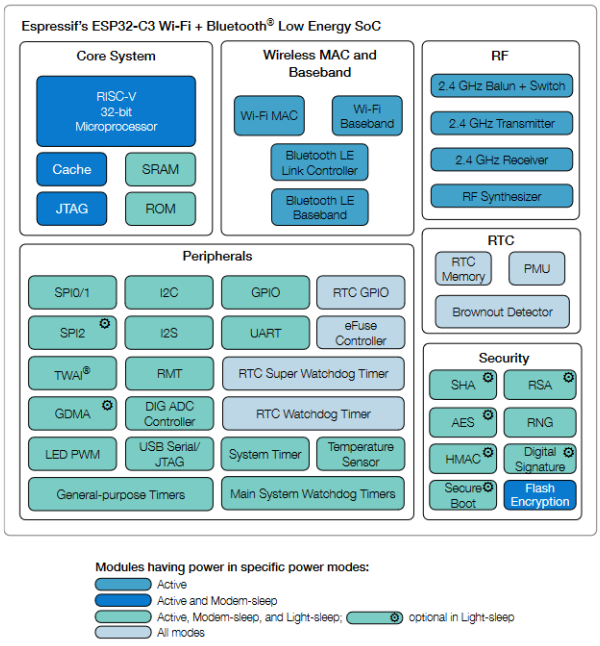
Razvojni sustav temelji se na modulu ESP32-C3-MINI-1. Modul je jedan u nizu ESP32-C3 serije SoC (engl. *System on Chip*) platformi tvrtke *Espressif*, te sadrži jednojezgreni 32-bitni procesor s RISC-V arhitekturom koji radi na frekvenciji do 160 MHz. Modul sadrži 400 KB memorije tipa SRAM (engl. *Static random-access memory*), od kojih je 16 KB rezervirano za priručnu memoriju (engl. *cache*), 384 MB memorije tipa ROM (engl. *Read-only memory*) te 4 MB memorije tipa *Flash*. Od periferije sadrži 22 programabilna GPIO pina (engl. *General Purpose Input Output*), te digitalna sučelja SPI, UART, I2C i I2S. Također sadrži upravljače za sučelja USB i JTAG koji se mogu koristiti za efikasnije otklanjanje pogrešaka u kodu (engl. *debugging*). Konfiguracija sustava prikazana je na slici 3.1. [8]



Slika 3.1: Konfiguracija razvojnog sustava ESP32-C3-DevKitM-1 [10]

Budući da modul ima funkciju RF (engl. *radio frequency*) primopredajnika, podržava bežično lokalno umrežavanje odnosno Wi-Fi, koji omogućava propusnost do 20 Mbps protokolom TCP te maksimalnu propusnost od 30 Mbps koristeći protokol UDP. Isto tako, podržava protokol Bluetooth s podrškom za velike udaljenosti.

Modul ESP32-C3-MINI-1 bežični je uređaj niske potrošnje energije (engl. *ultra-low-power*) primarno namijenjen razvoju aplikacija koje koriste Wi-Fi ili *Bluetooth Low Energy* (BLE) protokol. Na slici 3.2 nalazi se blok shema modula sa svim dostupnim značajkama.



Slika 3.2: Blok dijagram modula ESP32-C3 [8]

3.1. Wi-Fi

IEEE 802.11, skupina standarda za bežične lokalne mreže (engl. *WLANs*) [19], nudi nekoliko različitih načina bežične modulacije signala. Pojedini standardi označeni su slovima abecede. Za korisničke mreže postoje dva frekvencijska pojasa: 2,4 GHz i 5 GHz.

Prednosti pojasa od 2,4 GHz su veći doseg, bolje prolaženje kroz fizičke prepreke te bolja podrška jer više bežičnih uređaja koristi pojas od 2,4 GHz nego od 5 GHz. S druge strane, ovaj pojas ima manju propusnost i nudi manje kanala koji se ne preklapaju. Isto tako, može doći do zagušenja mreže jer kućni i Bluetooth uređaji koriste ovaj isti mrežni pojas.

Pojas od 5 GHz nudi brži protok, manje zagušenih kanala te ima više kanala koji se međusobno ne preklapaju. Ipak, ima kraći raspon u usporedbi s mrežama od 2,4 GHz jer teže prolazi kroz prepreke. [14]

U nastavku su opisani ključni standardi Wi-Fi tehnologije [6]:

– 802.11b - najsporiji i najjeftiniji standard, emitira u frekvencijskom pojasu od 2,4 GHz. Može prenijeti do 11 Mbps te koristi komplementarno šifriranje (engl. *complementary code keying - CCK*) radi poboljšanja brzine prijenosa.
– 802.11a - transmitira u pojasu od 5 GHz i može prenijeti do 54 Mbps. Koristi ortogonalno frekvencijsko multipleksiranje (engl. *orthogonal frequency-division multiplexing - OFDM*), što je efikasnija tehnika u odnosu na CCK koja dijeli radio signal u nekoliko podsignala prije slanja primatelju. Ova metoda značajno umanjuje interferenciju.
– 802.11g - poput standarda 802.11b, koristi frekvencijski pojas od 2,4 GHz. Međutim, može prenijeti do 54 Mbps jer koristi tehniku OFDM.
– 802.11n - kompatibilan je standard sa prethodno opisanim standardima. Nudi znatno poboljšanje u rasponu i brzini u odnosu na svoje prethodnike. Ovaj standard može prenijeti do četiri toka podataka, svaki maksimalno 150 Mbps, no većina usmjerivača (engl. *router*) dopušta dva ili tri toka.
– 802.11ac - radi isključivo u pojasu od 5 GHz, te je kompatibilan s prethodnim standardima. Manje je sklon interferenciji i brži je od prethodnih standarda s maksimalnim prijenosom od 450 Mbps jednim tokom.
– 802.11ax - najnoviji standard koji proširuje nekoliko ključnih mogućnosti svojih prethodnika. Usmjerivači koji podržavaju ovaj standard dopuštaju tok podataka do 9.2 Gbps, što je značajan porast u usporedbi s prethodnicima. Isto tako, moguće je postaviti više antena na jedan usmjerivač, čime je omogućen prihvat više veza odjednom bez usporavanja i interferencije.

Podsustav modula ESP32-C3 za Wi-Fi u skladu je sa standardom IEEE 802.111 te koristi nelicencirani pojas frekvencija od 2,4 GHz. U tom pojasu podržava propusnost od 20 i 40 MHz. Modul također podržava tehniku raznolikosti antena (engl. *antenna diversity*) za poboljšanje prijema i pouzdanosti signala korištenjem RF komutatora (engl. *switch*). Tim komutatorom upravljaju GPIO priključci i koristi se za odabir najbolje antene u kontekstu pouzdanosti i kvalitete signala. [9]

ESP32-C3 u potpunosti implementira protokol Wi-Fi na temelju standarda 802.11 b/g/n. Podržava osnovni skup (engl. *Basic Service Set - BSS*) operacija za značajke pristupne točke (engl. *software enabled access point - SoftAP*). Ovakve pristupne točke koriste softver kako bi omogućile uređajima kojima primarna svrha nije usmjeravanje prometa da postanu virtualni usmjerivač. [15] Također, upravljanje napajanjem odvija se automatski s minimalnom intervencijom domaćina kako bi se smanjila aktivnost uređaja.

Tvrtka *Espressif* također nudi biblioteke za povezivanje putem protokola TCP i IP te korištenje Wi-Fi *mesh* tehnologije. Pruža i podršku za protokole TLS 1.0, 1.1 i 1.2. Na slici 3.3 prikazani su Wi-Fi RF standardi koje koristi modul.

Name		Description
Center frequency range of operating channel [1]		2412 ~ 2484 MHz
Wi-Fi wireless standard		IEEE 802.11b/g/n
Data rate	20 MHz	11b: 1, 2, 5.5 and 11 Mbps 11g: 6, 9, 12, 18, 24, 36, 48, 54 Mbps 11n: MCS0-7, 72.2 Mbps (Max)
	40 MHz	11n: MCS0-7, 150 Mbps (Max)
Antenna type		PCB antenna and external antenna connector

Slika 3.3: Wi-Fi RF standardi [9]

ESP32 nudi nekoliko načina rada pri korištenju Wi-Fi tehnologije [20]:

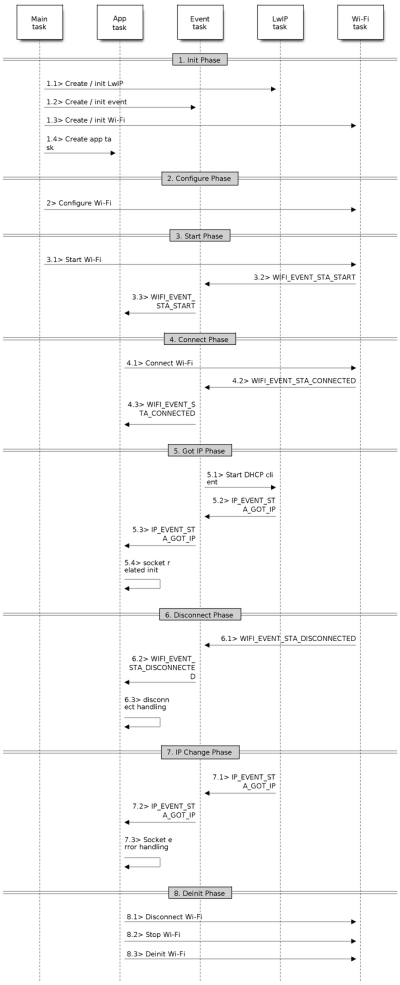
1. način rada stanice (engl. *station mode*) - ESP32 spaja se na točku pristupa,
2. način rada pristupne točke (engl. *SoftAP mode*) - druge se stanice spajaju na ESP32,
3. miješani - ESP32 radi kao stanica i pristupna točka spojena na drugu pristupnu točku.

U nastavku su opisani scenariji Wi-Fi povezivanja modula ESP32-C3 u načinu rada stanice i pristupne točke.

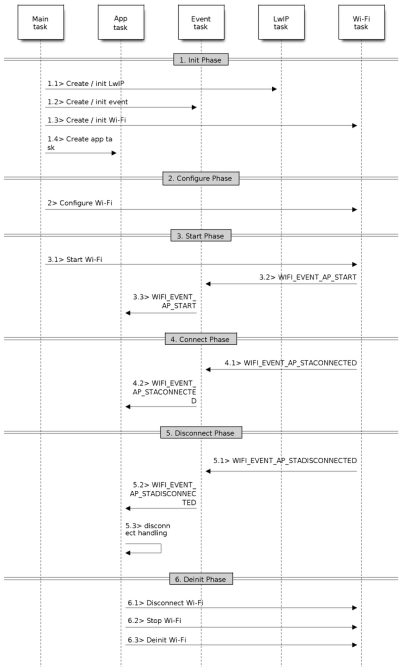
Na slici 3.4 prikazan je sekvencijski dijagram zadataka koje ESP32 obavlja u cijelom ciklusu spajanja i komunikacije s pristupnom točkom. Iz slike je vidljivo da se ciklus sastoji od osam faza. Prva faza služi za inicijalizaciju upravljačkih programa i pokretanje zadataka odnosno dretvi koje će obavljati zadatke vezane uz svoju dužnost. Glavni zadatak pokreće četiri različite dretve izvršavanja: aplikacijski zadatak, zadatak za događaje, zadatak za IP protokol, te zadatak za Wi-Fi. U drugoj fazi konfigurira se upravljački program za Wi-Fi. U sljedećoj se fazi pokreće upravljački program, nakon koje slijedi faza pretraživanja mreže i povezivanja na usmjerivač ili pristupnu točku. Nakon inicijalizacije DHCP klijenta, započinje faza dohvata IP adrese. Šesta faza odvija se nakon prekida Wi-Fi veze, čime se također uklanjaju i sve UDP i TCP konekcije. U aplikaciji se može omogućiti radno čekanje na ponovno uspostavljanje veze. Sedma faza pokreće se pri detekciji promjene IP adrese. Posljednja faza služi za programsko odspajanje s mreže i zaustavljanje upravljačkog programa za Wi-Fi.

Slika 3.5 modelira slučaj u kojem ESP32 ima ulogu pristupne točke. Scenarij je vrlo sličan ranije opisanom slijedu događaja, no razlikuje se u dvije faze i događajima

koji su pohranjeni u sustavu. Ovaj način rada nema fazu detekcije promjene IP adrese, jer je u ovom načinu ESP32 upravo taj uređaj čija se IP adresa može promijeniti. Isto tako, ne postoji faza dohvata IP adrese.



Slika 3.4: Primjer scenarija Wi-Fi povezivanja u načinu rada stanice [10]



Slika 3.5: Primjer scenarija Wi-Fi povezivanja u načinu rada pristupne točke [10]

U modulu ESP32 stavljen je veliki naglasak na mehanizme uštede energije, što se također preslikava na korištenje Wi-Fi veze. Modul pruža načine uštede energije i pri radu kao stanica i pristupna točka, no neke značajke nisu podržane u pristupnoj točki. Modul pri neaktivnosti može otići u stanje mirovanja (engl. *sleep mode*). Postoje dva načina uštede energije u načinu rada stanice: minimalna i maksimalna ušteda. Pri minimalnoj uštedi stanica se budi iz stanja mirovanja nakon svakog DTIM intervala (engl. *Delivery Traffic Indication Message*). Ovim se načinom ne gube globalno emitirane poruke (engl. *broadcast*) jer se one prenose nakon DTIM intervala. Međutim, ova metoda ne štedi puno energije ako je pristupna točka na koju je spojen modul postavila

malen interval. Pri maksimalnoj uštedi moguće je znatno produžiti vrijeme mirovanja u odnosu na DTIM interval, no ovime se riskira gubitak globalno emitiranih poruka.

4. Amazon Web Services (AWS)

Amazon usluge za web (engl. *Amazon Web Services - AWS*) sveobuhvatna je platforma za računarstvo u oblaku koju pruža tvrtka Amazon te sadrži brojne usluge u oblaku, uključujući infrastrukturu (engl. *Infrastructure as a Service - IaaS*), platformu (engl. *Platform as a Service - PaaS*) i softver (engl. *Software as a Service - SaaS*). AWS usluge nude organizacijske alate kao što su računalna snaga, baza podataka i usluge isporuke sadržaja [17].

AWS je podijeljen u više različitih usluga koje se mogu pojedinačno konfigurirati na temelju korisničkih potreba. Neke od usluga koje nudi AWS su: pohrana, baze podataka, monitoriranje, sigurnost, analitika, umjetna inteligencija te razvoj mobilnih aplikacija.

AWS pruža usluge iz mnogo podatkovnih centara (engl. *data center - DC*) koji su raspodijeljeni po zonama dostupnosti (engl. *availability zone - AZ*) diljem regija cijelog svijeta. Jedna regija obuhvaća nekoliko fizički bliskih zona povezanih mrežom niske latencije. Geografskom raspodijeljenošću AWS pruža višestruke prednosti [1]:

1. niska latencija: budući da su regije skupovi fizičkih mrežno povezanih bliskih zona, korisnički podaci i aplikacije mogu biti smješteni bliže krajnjim korisnicima što poboljšava performanse aplikacija,

2. visoka dostupnost: u slučaju kvara podatkovnog centra, korištenjem više zona dostupnosti unutar regije podaci mogu i dalje ostati dostupni,

3. otpornost i oporavak od katastrofe: podaci i aplikacije mogu se replicirati između više zona ili regija, što omogućava brzi oporavak u slučaju prirodnih katastrofa, tehničkih problema ili napada,

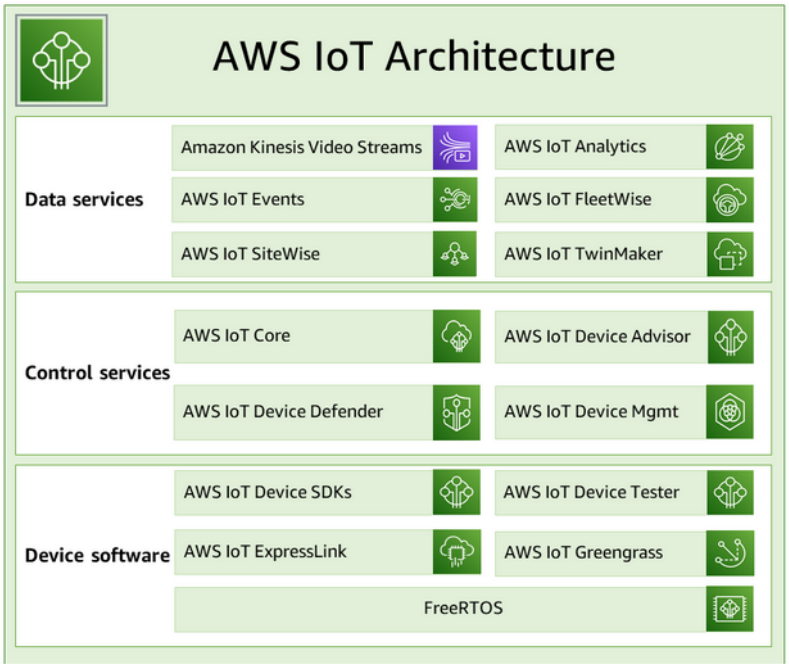
4. skalabilnost: klijenti mogu dinamički povećavati ili smanjivati resurse u različitim regijama ovisno o potražnji,

5. poboljšana sigurnost: fizički odvojeni podatkovni centri smanjuju rizik od pojedinačnih točaka neuspjeha i omogućavaju implementaciju složenijih sigurnosnih

strategija.

Također, nude se brojne mogućnosti za razvojne inženjere u sklopu AWS-a. Nudi alate naredbenog retka (engl. *command-line tools*) i pakete za razvoj programa (engl. *Software Development Kit - SDK*) za puštanje aplikacija u produkciju (engl. *deployment*) i upravljanje vlastitim uslugama i aplikacijama. Paketi za razvoj programa dostupni su u raznim programskim jezicima, uključujući programske jezike C++, Android, iOS, Java, Node.js, Python i Ruby.

AWS isto tako nudi brojne usluge za razvoj IoT sustava. Usluga AWS-a za IoT pruža platformu za upravljanje IoT uređajima te obradu podataka i njihovu pohranu na druge AWS usluge, poput baze podataka. AWS IoT pruža usluge u oblaku koje povezuju IoT uređaje s drugim uređajima i uslugama AWS-a u oblaku. Također pruža softver za uređaje, poput paketa za razvoj programa, za jednostavniju integraciju s uslugama AWS-a za IoT. Na slici 4.1 nalazi se prikaz arhitekture usluga koje AWS nudi za razvoj IoT sustava.



Slika 4.1: Arhitektura usluga AWS-a za IoT [3]

AWS podržava sljedeće komunikacijske protokole za IoT sustave:

- MQTT (engl. *Message Queuing Telemetry Transport*),
- HTTPS,
- LoRaWAN (engl. *Long Range Wide Area Network*),

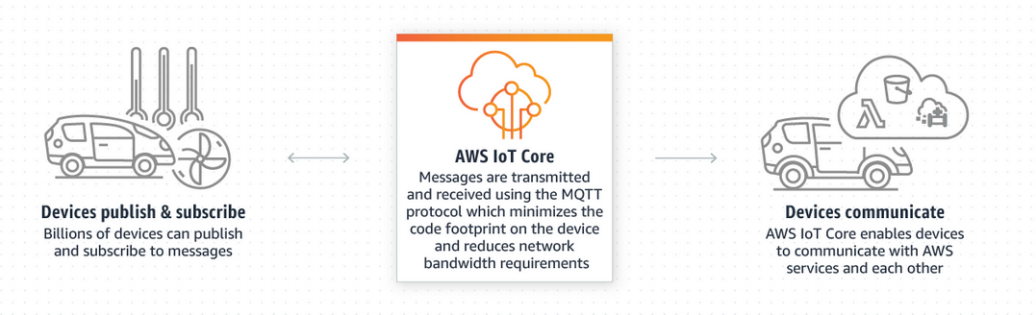
– TLS.

U nastavku su opisane usluge koje nudi AWS za razvoj IoT sustava.

4.1. AWS IoT Core

AWS IoT Core ključna je komponenta za integraciju oblaka i fizičkih uređaja. Omogućava povezivanje uređaja i preusmjeravanje poruka na usluge AWS-a. Koristi MQTT koji je standardni protokol za razmjenu poruka u IoT sustavima. To je lagan (engl. *lightweight*) protokol za prijenos poruka temeljen na objavi/pretplati sustavu u kojoj glavnu ulogu ima broker kao posrednik, te je pogodan za povezivanje udaljenih uređaja uz minimalnu potrošnju [16]. AWS IoT Core pruža paletu značajki za razmjenu poruka temeljenih na protokolu MQTT, koje pomažu pri izradi prilagodljive i skalabilne IoT arhitekture [3]. Komponenta također ima ugrađenu podršku za upravljanje MQTT brokerom koji podržava trajne, uvijek uključene veze, napredna pravila zadržavanja poruka te obrađuje više uređaja i tema istovremeno. Isto tako, AWS IoT Core podržava najnoviji standard MQTT 5 i kompatibilan je sa prethodnim standardom MQTT 3, što omogućuje učinkovito upravljanje heterogenim implementacijama s mnoštvom specifikacija. Nadalje, komponenta omogućava višestruke metode provjere autentičnosti i pristupne politike za zaštitu rješenja od ranjivosti. Štoviše, koristeći pomoć drugih usluga u sklopu platforme kao što je AWS IoT Core Device Advisor, moguće je pristupiti unaprijed izgrađenim paketima testova za provjeru MQTT funkcionalnosti uređaja tijekom faze razvoja.

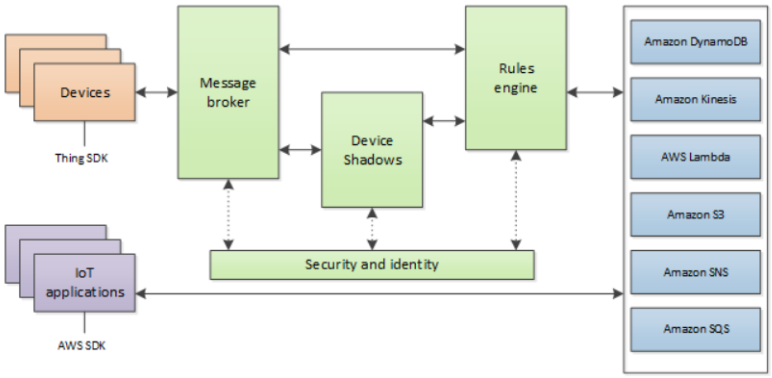
Na slici 4.2 nalazi se pregled rada usluge AWS IoT Core.



Slika 4.2: Princip rada usluge AWS IoT Core [3]

AWS IoT Core pruža usluge koje povezuju oblak AWS-a s IoT uređajima kako bi se ostale usluge u oblaku i aplikacije mogle međusobno komunicirati s tim uređajima. Na slici 4.3 nalaze se svi segmenti usluge AWS IoT Core te kako oni komuniciraju s

vanjskim dijelovima. Zelenom bojom označena je sama usluga, narančastom bojom fizički uređaji odnosno stvari (engl. *Things*), sivom bojom druge IoT aplikacije unutar AWS-a koje se mogu izravno spojiti na sustave u AWS-u, dok su plavom bojom označene ostale usluge odnosno sustavi koji se nalaze u ekosustavu AWS.



Slika 4.3: Komponente usluge AWS IoT Core [3]

U nastavku su ukratko opisane ključne usluge koje pokriva AWS IoT Core [3].

Usluge za slanje poruka

AWS IoT Core usluge za povezivanje pružaju sigurnu komunikaciju s IoT uređajima i upravlja porukama koje prolaze između uređaja i oblaka.

Prilazni uređaj omogućuje uređajima sigurnu i efikasnu komunikaciju sa sustavom AWS. Komunikacija je osigurana sigurnosnim protokolima koji koriste X.509 certifikate.

Broker za poruke pruža mehanizam uređajima i aplikacijama slanje i primanje poruka. Moguće je koristiti protokol MQTT ili direktno *WebSocket* za objavu i pretplatu na teme. Uređaji i klijenti koriste sučelje HTTP REST za objavu poruka brokeru. Broker zatim distribuira podatke na uređaje koji su se pretplatili na određene teme kao i na druge AWS aplikacije te usluge koje prate teme na brokeru.

AWS IoT Core za protokol LoRaWAN omogućava postavljanje privatne LoRaWAN mreže tako što poveže LoRaWAN te prilazne uređaje na AWS bez potrebe za razvojem mrežnog servera za LoRaWAN (engl. *LoRaWAN Network Server - LNS*). Poruke primljene od LoRaWAN uređaja šalju se na stroj za pravila (engl. *rules engine*) gdje se formatiraju i prosljeđuju ostalim AWS uslugama.

Stroj za pravila (engl. *rules engine*) povezuje podatke iz brokera s drugim AWS IoT uslugama za pohranu i dodatnu obradu. Primjerice, moguće je umetati ili pretraživati

po podatkovnim tablicama ili pozvati određene definirane funkcije na temelju izraza definiranog u stroju. Isto tako, moguće je obraditi te podatke i proslijediti novostvoreni format poruka drugim uslugama ili bazama podataka.

Upravljačke usluge

Upravljačke usluge AWS IoT Core komponente pružaju sigurnost uređaja te značajke za upravljanje i registraciju novih uređaja.

Moguće je definirati vlastite autorizatore radi upravljanja autentifikacijskim i autorizacijskim strategijama koristeći vlastiti servis za autentifikaciju i funkcije za računanje Lambda koje nudi AWS. Lambda je računalna usluga za slučajeve i aplikacije dinamičke skalabilnosti. Pokreće kod na infrastrukturi visoke dostupnosti i obavlja cjelokupnu administraciju računalnih resursa, uključujući održavanje poslužitelja i operativnog sustava, osiguravanje kapaciteta i automatsko skaliranje ovisno o trenutnim potrebama sustava. Lambda funkcije korisne su za obradu datoteka, tokova te HTTP zahtjeva. Funkcije su pogodne za obradu podataka prije preusmjeravanja na drugu funkciju ili pohranu.

Usluga za registraciju uređaja u sustav (engl. *provisioning*) omogućava konfiguriranje i prijavu uređaja u AWS koristeći predložak koji opisuje resurse potrebne uređaju: stvar, certifikat i nekoliko politika. Stvar (engl. *thing*) je unos u registar koji sadrži atribute opisa uređaja. Uređaji koriste certifikate za autentifikaciju sa sustavom AWS. Politike određuju koje operacije uređaj može izvršiti. Svaka politika ima definirani dokument s izjavom koja se sastoji od učinka, akcije i resursa nad kojim se akcija izvršava. Učinak može biti *Dopusti* ili *Uskrati*, akcija se odnosi nad radnje s MQTT brokerom i poslovima, a resurs se odnosi na regiju i račun u kojem politika vrijedi.

Isto tako, moguće je definirati grupe za lakšu kategorizaciju i upravljanje uređajima. Grupe također mogu imati podgrupe, i tako graditi hijerarhiju grupa. Sve akcije izvršene na roditeljima propagiraju se do najdubljih podgrupa. Dozvole dodijeljene grupi primjenjuju se na sve uređaje u toj grupi i svim njihovim podgrupama.

Usluga za poslove (engl. *jobs*) omogućava definiranje udaljenih operacija koje se pošalju i izvrše na fizičkim uređajima spojenih u oblak. Posao se može odnositi preuzimanje i instalaciju aplikacija, ažuriranje sustava, ponovno pokretanje, obnova certifikata i slično.

Usluga za sigurnost i identitet pruža dijeljenu odgovornost za sigurnost unutar AWS oblaka. Fizički uređaji moraju držati vjerodajnice na sigurnom kako bi se podaci mogli slati brokeru na siguran način. Značajke za sigurnost koriste i stroj za pravila

kao i broker za poruke radi sigurnog prijenosa podataka drugim uslugama unutar AWS ekosustava.

Usluge za uređaje

AWS IoT Core osigurava pouzdano aplikacijsko iskustvo iako uređaji nisu uvijek povezani.

Sjena uređaja (engl. *Device Shadow*) dokument je u JSON formatu koji se koristi za pohranu i dohvat trenutnog stanja i informacija o uređaju. AWS nudi uslugu koja održava stanje uređaja (engl. *Device Shadow service*) kako bi aplikacije mogle komunicirati s uređajem bez obzira je li uređaj na mreži ili ne. Kada uređaj nije priključen na mrežu, usluga sjene uređaja upravlja podacima za povezane uređaje. Kada se uređaj ponovno spoji na mrežu, sinkronizira stanje sa sjenom uređaja koja se nalazi u oblaku. Uređaji također mogu objavi svoje stanje usluzi u bilo kojem trenutku kako bi bilo na raspolaganju drugim aplikacijama i uređajima.

4.1.1. AWS IoT Fleet Provisioning

Kao što je ranije navedeno, AWS nudi uslugu registracije uređaja u sustav čime uređaj dobiva pristup ostalim uslugama unutar platforme, poput povezivanja na MQTT broker. Moguće je dinamički registrirati više uređaja pomoću privremenih ili trajnih certifikata, što se naziva provizioniranje flote (engl. *fleet provisioning*). Pomoću ove usluge AWS generira i sigurno dostavi certifikate te privatne ključeve na uređaje pri prvom povezivanju na platformu. AWS izdaje klijentske certifikate koji su potpisani certifikacijskim tijelom tvrtke Amazon (engl. *Certificate Authority - CA*) [3].

Tri su ključna koraka pri omogućavanju registracije više uređaja u sustav:

1. određivanje načina registracije,
2. definiranje upravljačke strukture nad uređajima,
3. kreiranje predložaka za registraciju.

Određivanje načina registracije u sustav povezano je s odabirom vrste certificiranja koja će se koristiti. Uređaji trebaju jedinstveni certifikat za povezivanje s platformom AWS i korištenje njenih usluga. Postoje tri metode certificiranja uređaja te odabir ovisi o mogućnostima proizvodnje uređaja i ovlastima samih korisnika:

- registracija uređaja vlastitim jedinstvenim certifikatima,
- registracija od strane korisnika od povjerenja,

– registracija certifikatom zahtjeva.

Registracija uređaja vlastitim jedinstvenim certifikatima podrazumijeva postojanje verificiranih certifikata na samom uređaju pri prvom povezivanju u sustav. Ovaj se način još naziva i pravodobno provizioniranje (engl. *just-in-time provisioning*). Problem s ovim pristupom jest što instalacija certifikata nije uvijek moguća prije dostave uređaja korisniku. Isto tako, čak iako je instalacija certifikata moguća, nije uvijek moguće instalirati certifikate sigurno na uređaj, stoga je potrebno koristiti alternativne metode.

U slučaju kada rana certifikacija uređaja nije moguća, autorizirani ili krajnji korisnici (engl. *trusted users*) mogu uz pomoć aplikacije registrirati uređaje prije njihova spajanja na platformu. Ovdje je potrebno korisnicima pružiti aplikaciju kojom bi konfigurirali uređaj prilikom instalacije, te *firmware* uređaja mora podržavati ovaj način registracije.

Treća opcija jest registracija pomoću certifikata zahtjeva (engl. *claim certificate*). Certifikati zahtjeva privremeni su certifikati koji se mogu instalirati na više uređaja, te se pri prvoj registraciji u sustav zamijene s novim, jedinstvenim certifikatom. Ova metoda zahtijeva dodatne sigurnosne provjere zbog mogućnosti curenja privremenih certifikata zahtjeva jer više uređaja dijeli isti certifikat, no prijetnja se može ublažiti redovitim rotiranjem privremenih certifikata te kreiranjem više certifikata zahtjeva koji se mogu koristiti. Isto tako, privremenim certifikatima potrebno je dodijeliti minimalan skup dozvoljenih radnji potrebnih za registraciju uređaja. Dodatna sigurnosna provjera može se izvršiti i u obliku Lambda funkcije dodatnom provjerom primljenih parametara s uređaja.

Upravljačka struktura nad uređajima je važna radi lakše organizacije uređaja u samom sustavu. Povezani uređaji predstavljeni su u platformi kao stvari tj. resursi koji se mogu organizirati i održavati. Osim ranije spomenutih stvari i grupa stvari, moguće je uređajima dodijeliti atribute po kojima se može pretraživati. Atributi se isto tako mogu dinamički dodijeliti na temelju podataka s uređaja prilikom njegove registracije. Oni se definiraju u predlošku za registraciju.

Predložak za registraciju dokument je u formatu JSON koji opisuje resurse, politike i dozvole koje je potrebno dodijeliti uređaju kada je registriran. Odjeljak *parametri* u predlošku definira resurse u odjeljku *resursi* koje uređaj mora koristiti pri interakciji s platformom. Svaki parametar definira naziv, vrstu i zadanu vrijednost koja nije obavezna. Zadana vrijednost koristi se kada rječnik proslijeđen s predloškom ne sadrži vrijednost za parametar. Odjeljak *parametri* izgleda na sljedeći način:

```
{
```

```
    "Parameters" : {
      "ThingName" : {
        "Type" : "String"
      },
      "SerialNumber" : {
        "Type" : "String"
      },
      "Location" : {
        "Type" : "String",
        "Default" : "HR"
      },
      "CSR" : {
        "Type" : "String"
      }
    }
  }
```

Isječak koda 4.1: Odjeljak *parametri* u predlošku za registraciju

Odjeljak *resursi* u predlošku definira resurse koji su potrebni uređaju za daljnji rad i komunikaciju sa sustavom: stvar, certifikat, jedna ili više politika. Svaki resurs specificira naziv, vrstu i skup svojstava. Vrsta resursa može biti jedna od tri vrijednosti: `AWS::IoT::Thing`, `AWS::IoT::Certificate`, te `AWS::IoT::Policy`. Resursi tipa certifikat imaju posebna svojstva koja ih definiraju, ovisno o tome koja je metoda registracije uređaja odabrana. Politike su vezane za već postojeće politike u sustavu AWS. Odjeljak *resursi* može izgledati na sljedeći način:

```
{
  "Resources" : {
    "thing" : {
      "Type" : "AWS::IoT::Thing",
      "Properties" : {
        "ThingName" : {"Ref" : "ThingName"},
        "AttributePayload" : { "version" : "v1", "
   serialNumber" :  {"Ref" : "SerialNumber"}},
        "ThingTypeName" :  "lightBulb-versionA",
        "ThingGroups" : ["v1-lightbulbs", {"Ref" : "
   Location"}]
```

```
      },
      "OverrideSettings" : {
        "AttributePayload" : "MERGE",
        "ThingTypeName" : "REPLACE",
        "ThingGroups" : "DO_NOTHING"
      }
    },
    "certificate" : {
      "Type" : "AWS::IoT::Certificate",
      "Properties" : {
        "CertificateSigningRequest": {"Ref" : "CSR"},
        "Status" : "ACTIVE"
      }
    },
    "policy" : {
      "Type" : "AWS::IoT::Policy",
      "Properties" : {
        "PolicyDocument" : "{ \"Version\": \"2012-10-17\"
  , \"Statement\": [{ \"Effect\": \"Allow\", \"Action\":
  [\"iot:Publish\"], \"Resource\": [\"arn:aws:iot:us-
  east-1:123456789012:topic/foo/bar\"] }] }"
      }
    }
  }
}
```

Isječak koda 4.2: Odjeljak *resursi* u predlošku za registraciju

Valja napomenuti kako je moguće u korisničkom sučelju platforme odabrati sve parametre, te kreiranje vlastitog dokumenta u JSON formatu nije potrebno, nego se automatski generira iz korisničkog odabira u sučelju.

4.1.2. AWS IoT Jobs

4.1.3. AWS IoT OTA

4.1.4. AWS IoT Device Shadow

4.1.5. Ostale dostupne IoT usluge u sustavu AWS

Uz ranije opisanu glavnu komponentu IoT Core koju nudi AWS za stvaranje IoT aplikacija, u samom ekosustavu nalazi se još mogućnosti za jednostavniju integraciju oblaka i fizičkih uređaja. U nastavku su ukratko opisane ostale usluge koje se mogu integrirati uz jezgrenu uslugu AWS IoT Core.

Važno je napomenuti kako nisu sve usluge dostupne u svim regijama unutar platforme AWS.

IoT Analytics

AWS IoT Analytics automatizira korake potrebne za analizu podataka prikupljenih od IoT uređajima. Filtrira, transformira i obogaćuje podatke prije nego ih pohrani u vremensku bazu podataka za daljnju analizu. Moguće je postaviti uslugu da prikuplja podatke s uređaja samo koji su potrebni, vrši matematičke operacije i dopunjava podatke raznim metapodacima, primjerice o lokaciji. Zatim se podaci mogu analizirati koristeći ugrađeni sustav za pretraživanje koji koristi SQL sintaksu ili pak vršiti kompleksniju analizu koristeći usluge umjetne inteligencije. Isto tako, ova usluga nudi vizualizaciju podataka integracijom s dodatnom uslugom Amazon QuickSight.

IoT Device Defender

AWS IoT Device Defender potpuna je usluga koja pomaže pri osiguranju IoT uređaja. Kontinuirano revidira IoT konfiguracije radi provjere jesu li sve u skladu s najboljim sigurnosnim praksama. Također pruža kontinuirano monitoriranje sigurnosnih metrika s uređaja i usluge AWS IoT Core kako bi se detektirale anomalije u ponašanju pojedinih uređaja.

Ova usluga također omogućuje slanje alarma na konzolu AWS IoT sustava i na uslugu za monitoriranje Amazon CloudWatch. Koriste se ugrađene mitigacijske akcije kako bi se izolirali nesigurni uređaji.

IoT Events

AWS IoT Events usluga služi za praćenje događaja u sustavu. Ova usluga prati ulazne podatke s više IoT uređaja i aplikacija radi prepoznavanja uzoraka i pokretanja prikladnih operacija na određene događaje. Moguće je pratiti ne samo fizičke uređaje, nego i druge AWS aplikacije integrirane u IoT sustav.

IoT FleetWise

AWS IoT FleetWise jest usluga koja se koristi za prikupljanje podataka od vozila i njihovu organizaciju u oblaku. Prikupljeni se podaci mogu koristiti za poboljšanje kvalitete, performansa i autonomije vozila. Također podržava više različitih protokola i podatkovnih formata. Ova usluga pomaže pri transformaciji poruka niske razine (engl. *low-level*) u oblik čitljiv čovjeku i standardizira podatke radi lakše analize u oblaku. Moguće je također definirati vrstu podataka i trenutak u kojem se ti podaci šalju u oblak.

Kada su podaci o vozilu u oblaku, mogu se koristiti u aplikacijama koje analiziraju zdravlje vozila. Ove informacije mogu pomoći pri identifikaciji potencijalnih problema u održavanju i pri unapređenju naprednih tehnologija poput autonomne i asistirane vožnje integracijom strojnog učenja.

IoT Greengrass

AWS IoT Greengrass jest usluga otvorenog koda (engl. *open source*) za računarstvo na rubu (engl. *edge computing*) i u oblaku koja pomaže pri izradi, objavi i upravljanju IoT aplikacija na uređajima. Može se koristiti za omogućavanje uređajima lokalno reagiranje na podatke koje generiraju, pokretanje modela strojnog učenja za predikciju, te filtriranje i agregaciju podataka s uređaja. Omogućava uređajima da prikupljaju i analiziraju podatke ne u oblaku, nego ili na samom uređaju ili drugom mjestu koje je bliže izvorištu tih podataka. Također može komunicirati na siguran način s uslugom AWS IoT Core i izvoziti podatke u oblak. Karakteristika računarstva u rubu, koje omogućava ova komponenta, jest približavanje računanja izvorišnim uređajima, čime se poboljšava vrijeme odziva i štedi propusnost [4].

IoT Roborunner

AWS IoT RoboRunner nova je usluga koja pruža infrastrukturu za optimizaciju robota iz jedne točke gledišta. Uz pomoć ove usluge moguće je izgraditi aplikacije za jednos-

tavniji međusobni rad robota. Namijenjena je za industrijske robote i automatizirane sustave za olakšano upravljanje opremom. Pruža centralne repozitorije podataka za pohranu te podržava različite podatkovne formate od raznih robota i autonomnih sustava.

IoT TwinMaker

AWS IoT TwinMaker usluga je za kreiranje operativnih digitalnih dvojnika fizičkih i digitalnih sustava. Stvara digitalne vizualizacije koristeći mjerenja i analize iz raznih senzora i kamera radi praćenja stvarnog stanja i uvjeta u kojima se objekt, zgrada ili kompleks nalazi. Podaci iz stvarnog svijeta se mogu koristiti za dijagnostiku i ispravljanje pogrešaka ili pak optimizaciju operacija.

Digitalni dvojnik (engl. *digital twin*) digitalna je reprezentacija sustava i svih njegovih fizičkih i digitalnih komponenti. Dinamički se ažurira primitkom novih podataka kako bi simulirao stvarno stanje i ponašanje sustava.

IoT SiteWise

AWS IoT SiteWise jest usluga koja skalabilno prikuplja, modelira, analizira i vizualizira podatke iz industrijske opreme. Usluga pruža kreiranje web aplikacija za operativne korisnike radi prikaza i analize industrijskih podataka u stvarnom vremenu. Moguće je dobiti uvide u podatke i operacije konfiguriranjem i praćenjem raznih metrika, primjerice efektivnost i efikasnost opreme. Ovu je uslugu moguće koristiti jedino uz ranije opisan IoT TwinMaker.

5. Povezivanje razvojnog sustava i oblaka

Oblak koju pruža platforma AWS i razvojni sustav ESP32-C3 dva su odvojena sustava koja moraju međusobno komunicirati i razmjenjivati podatke. Za ostvarenje njihove veze razvijena su dva programska rješenja:

1. programska potpora za mikrokontroler, koja će omogućiti dinamičko povezivanje na Wi-Fi, spajanje na platformu AWS te slanje podataka u oblak,
2. programska potpora za platformu AWS, koja će ostvariti umrežavanje uređaja u sustav, ažuriranje softvera na uređaju, pohranu primljenih podataka s uređaja te prikaz tih podataka u web aplikaciji.

5.1. Programska potpora za mikrokontroler

Programska potpora za uređaj ESP32-C3 sastoji se od nekoliko komponenti:

- dinamičko povezivanje na bežičnu mrežu,
- spajanje na platformu AWS,
- učitavanje novog softvera,
- očitavanje senzorskih mjerenja,
- slanje podataka u oblak protokolom MQTT.

Neke od navedenih komponenti izvršavaju se slijedno, dok se druge izvršavaju paralelno. Spajanje na Wi-Fi i povezivanje s platformom AWS ključni su koraci koji prethode bilo kakvom pokušaju slanja podataka u oblak. Isto tako, praćenje ažuriranja softvera i očitavanje mjerenja izvršavaju se paralelno u posebnim procesima budući da nisu sekvencijalni niti međusobno isključivi zadaci. U nastavku je pobliže opisan svaki navedeni segment programske potpore.

5.1.1. Dinamičko povezivanje mikrokontrolera na Wi-Fi

Radni okvir ESP-IDF nudi dinamičko spajanje na Wi-Fi mrežu pomoću zasebne komponente. Ovaj se postupak naziva provizioniranje (engl. *provisioning*). Ova komponenta pruža aplikacijska programska sučelja (engl. *Application Programming Interface - API*) koja kontroliraju pružanje usluge za primanje i konfiguriranje Wi-Fi vjerodajnica putem sigurnih komunikacijskih protokola. Sigurnosni protokoli definirani su u komponenti protokolne komunikacije (engl. *protocomm*) koja upravlja sigurnim sjednicama (engl. *sessions*) i pruža radni okvir za višestruki prijenos podataka. Također je moguće direktno koristiti sloj protokolne komunikacije radi implementacije specifične za aplikaciju [7].

Sloj protokolne komunikacije interno koristi mehanizam protokolnih međuspremnika (engl. *protocol buffers - protobuf*) za sigurno uspostavljanje sjednice. Protokolni međuspremnici namijenjeni su za serijalizaciju strukturiranih podataka neovisno o programskom jeziku i platformi. Koristan je pri izradi programa i sustava koji međusobno komuniciraju putem mreže zbog kompaktnosti i niske latencije [12].

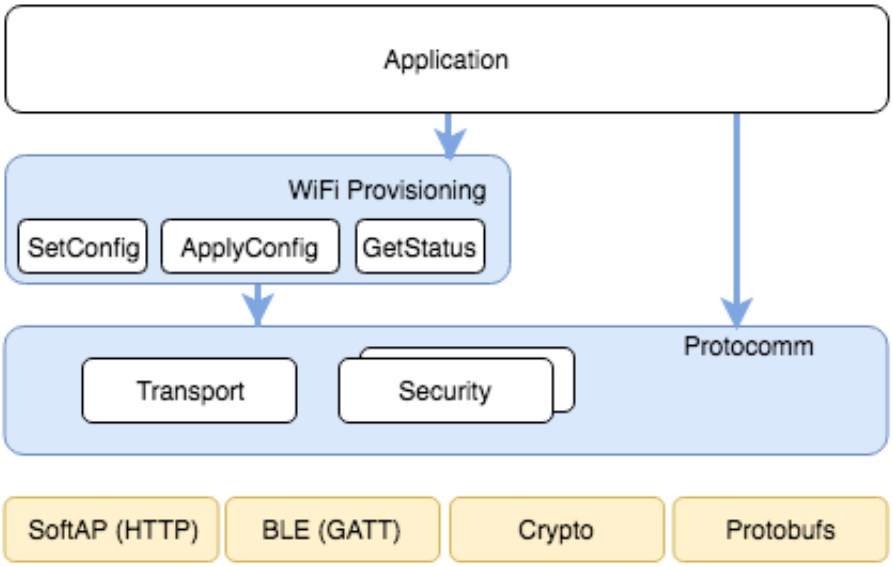
Sloj protokolne komunikacije pruža radni okvir za različite načine komunikacije:

1. protokol BLE,
2. Wi-Fi (SoftAP u kombinaciji s HTTP serverom).

Pružajući korisnicima okvir za ostvarivanje usluge dinamičkog povezivanja u mrežu, neovisno o načinu komunikacije, ovakva vrsta podrške naziva se unificirano provizioniranje (engl. *unified provisioning*). Ovakav način prijave uređaja na mrežu zahtijeva interakciju korisnika putem vanjskog uređaja za slanje vjerodajnica na mikrokontroler. Tvrtka *Espressif* pruža jednostavna mobilna rješenja koja se mogu koristiti gotova ili pak uklopiti u vlastitu mobilnu aplikaciju. Na slici 5.1 prikazana je arhitektura usluge.

Kao što je ranije opisano, arhitektura je bazirana na sloju protokolne komunikacije koji je odgovoran za prijenos podataka i sigurnost. Služi za jednostavne povratne pozive aplikaciji (engl. *callbacks*) i dobivanje Wi-Fi statusa. Sama aplikacija ima kontrolu nad implementacijom povratnih poziva.

Aplikacija stvara instancu protokolne komunikacije koja se preslikava na određeni prijenosni protokol i sigurnosnu shemu. Svaki prijenos podataka u sloju protokolne komunikacije ima koncept krajnje točke (engl. *endpoint*) koji odgovara logičkom komunikacijskom kanalu za određenu vrstu informacija. Primjerice, sigurnosno rukovanje (engl. *handshake*) odvija se na različitoj krajnjoj točki u odnosu na točku za Wi-Fi konfiguraciju. Svaka se krajnja točka identificira nizom znakova i mijenja se ovisno



Slika 5.1: Arhitektura unificiranog provizioniranja [7]

o internom prikazu krajnje točke. U slučaju prijenosa pomoću Wi-Fi veze odnosno SoftAP funkcionalnosti, krajnja točka prikazuje se kao URI, dok u slučaju prijenosa podataka putem protokola BLE odgovara GATT karakteristici sa specifičnim identifikatorom.

Oglašavanje i otkrivanje uređaja prepušteno je aplikaciji i ovisno o odabranom protokolu, vanjske aplikacije mogu odabrati odgovarajuću metodu za oglašavanje i otkrivanje. Za Wi-Fi prijenos obično se koristi ime mreže pristupne točke. Za prijenos putem protokola BLE može se koristiti ime samog uređaja.

Kao što je opisano, podržano je korištenje protokola BLE kao i Wi-Fi usluge za prijenos vjerodajnica. Pri odabiru prijenosnog kanala za spajanje uređaja u mrežu, potrebno je razmotriti nekoliko točaka. Za početak, prijenos temeljen na protokolu BLE prednost održavanja netaknutog komunikacijskog kanala između uređaja i klijenta tijekom prijenosa podataka, što osigurava pouzdanu povratnu informaciju. S druge strane, prijenos putem Bluetootha troši oko 110 KB memorije tijekom rada, što je na uređajima niskih resursa velika potrošnja. Korisno je što se korištena memorija može vratiti na hrpu (engl. *heap*) po završetku umrežavanja uređaja ukoliko se BLE funkcionalnosti više ne koriste. Prijenos temeljen na Wi-Fi mreži, odnosno SoftAP funkcionalnosti, vrlo je interoperabilan i ne troši dodatnu memoriju. Međutim, mikrokontroler koristi isti radio za emitiranje pristupne točke i za spajanje na željenu mrežu. Budući da se te akcije mogu odvijati na različitim kanalima, postoji mogućnost da se ažuriranja statusa veze ne dostave na mobilni uređaj. Također, mobilni se uređaj mora odspojiti s izvorne

Wi-Fi mreže radi privremenog spajanja na pristupnu točku mikrokontrolera. Uređaj će se spojiti na izvornu mrežu tak kada mikrokontroler ugasi pristupnu točku [7].

Za razvoj predloženog rješenja korišteno je slanje vjerodajnica pomoću Wi-Fi mreže, odnosno privremene pristupne točke. Kao što je ranije opisano, protokol BLE troši značajnu količinu *heap* memorije, a razvojni sustav ESP32-C3 nema dovoljno radne memorije koja bi pokrila prijavu u mrežu uz ostale radne procese. Mikrokontroler najprije stvori privremenu pristupnu točku na koju se mobilni uređaj spaja pomoću mobilne aplikacije. Zatim, nakon skeniranja dostupnih Wi-Fi mreža u blizini, u mobilnoj aplikaciji odabire se željena mreža i unese lozinka. Vjerodajnice se zatim pošalju putem Wi-Fi mreže, i mobilni uređaj može se odspojiti s privremene pristupne točke. Vjerodajnice se pohrane u memoriju tipa NVS (engl. *non-volatile storage*) koja ne zahtijeva konstantno napajanje kako bi se zadržala na uređaju. Ovime je omogućeno povezivanje uređaja u sustav čak i kada dođe do prekida napajanja [18]. Memorija tipa NVS može se jedino programski obrisati, te bi u idealnom izvedbenom rješenju postojao vanjski gumb spojen na mikrokontroler koji bi pokretao brisanje te memorije i tako omogućio ponovno spajanje na željenu mrežu. Sljedeći programski isječak prikazuje inicijalizaciju memorije NVS, mrežnog sučelja te stvaranje pristupne točke.

```
/* Init NVS partition */
esp_err_t ret = nvs_flash_init();
/* Init TCP/IP */
ESP_ERROR_CHECK(esp_netif_init());
/* Init the event loop */
ESP_ERROR_CHECK(esp_event_loop_create_default());
wifi_event_group = xEventGroupCreate();
/* Init Wi-Fi including netif with default config */
esp_netif_create_default_wifi_sta();
esp_netif_create_default_wifi_ap();
wifi_prov_mgr_config_t config = {
  .scheme = wifi_prov_scheme_softap,
  .scheme_event_handler = WIFI_PROV_EVENT_HANDLER_NONE
};
  /* Init provisioning manager with above config */
ESP_ERROR_CHECK(wifi_prov_mgr_init(config));
ESP_ERROR_CHECK(wifi_prov_mgr_start_provisioning(
 security, (const void *) sec_params, service_name,
```

```
   service_key));
  wifi_prov_print_qr(service_name, username, pop,
   PROV_TRANSPORT_SOFTAP, disp);
```

Isječak koda 5.1: Stvaranje pristupne točke

LCD zaslon

Za povezivanje mobilnog uređaja na privremenu pristupnu točku koju emitira razvojni sustav, potrebno je skenirati QR kod koji mikrokontroler generira. Budući da ESP32-C3 nema vlastito sučelje, na sustav je spojen zaslon OLED SSD1306 veličine 128×64 piksela. Uređaj sa zaslonom komunicira putem I2C sučelja, a za prikaz sadržaja na zaslonu korištena je biblioteka LVGL (engl. *Light and Versatile Graphics Library*). To je grafička biblioteka otvorenog koda namijenjena izradi aplikacija s grafičkim korisničkim sučeljem (engl. *Graphical User Interface - GUI*) za ugradbene sustave. Pruža radni okvir s mnogim značajkama, temama i paletama boja. Isto tako, biblioteka troši vrlo malo resursa, što je čini pogodnom za uređaje poput razvojnog sustava ESP32-C3 [2]. Generiranje QR koda obavlja se pomoću biblioteke *QR-Code-Generator* koja je prilagođena ESP32 uređajima.

```
static void wifi_prov_print_qr(const char *name, const
   char *usrname, const char *pop, const char *transport,
    lv_disp_t *disp) {
  char payload[150] = {0};
    snprintf(payload, sizeof(payload),
      "{\"ver\":\"%s\",\"name\":\"%s\",\"username\":\"%s
   \",\"pop\":\"%s\",\"transport\":\"%s\"}",
       PROV_QR_VERSION, name, usrname, pop, transport);
    esp_qrcode_config_t cfg = {
    .display_func = generate_qr_code_lcd,
    .max_qrcode_version = 10,
    .qrcode_ecc_level = ESP_QRCODE_ECC_LOW
  };
  esp_qrcode_generate(&cfg, payload);
}
```

Isječak koda 5.2: Generiranje QR koda iz pristupne točke

Prethodna funkcija povezuje pristupnu točku s QR kodom. Podaci o samoj pristupnoj točki učitaju se u privremenu varijablu, čiji se sadržaj prosljeđuje biblioteci za generiranje QR koda. Dobiveni se podaci zatim prosljeđuju funkciji za prikaz koda na zaslonu. QR kod prikazuje se na zaslonu piksel po piksel, skalirajući veličinu QR koda na temelju širine i duljine samog zaslona.

```
void generate_qr_code_lcd(esp_qrcode_handle_t qrcode)
{
  ESP_LOGI(TAG, "%s", "Started generate_qr_code_lcd...");

  int size = qrcodegen_getSize(qrcode);

  // Calculate the scale factor
  int scale = (int)fmin(EXAMPLE_LCD_H_RES / size,
   EXAMPLE_LCD_V_RES / size);

  // Calculate horizontal shift
  int shift_x = (EXAMPLE_LCD_H_RES - size * scale)/2;

  // Calculate vertical shift
  int shift_y = (EXAMPLE_LCD_V_RES - size * scale)/2;

  if (lvgl_port_lock(0)) {
    lv_obj_t *screen = lv_scr_act();
    lv_obj_clean(screen); // Clear the screen to ensure
   it's dark

    // Create a canvas object
    lv_obj_t *canvas = lv_canvas_create(screen);
    static lv_color_t cbuf[LV_CANVAS_BUF_SIZE_TRUE_COLOR(
   EXAMPLE_LCD_H_RES, EXAMPLE_LCD_V_RES)];
    lv_canvas_set_buffer(canvas, cbuf, EXAMPLE_LCD_H_RES,
    EXAMPLE_LCD_V_RES, LV_IMG_CF_TRUE_COLOR);
    lv_canvas_fill_bg(canvas, lv_color_white(),
   LV_OPA_COVER);
```

```
    // Draw the QR code on the canvas
    for (uint8_t y = 0; y < size; y++) {
      for (uint8_t x = 0; x < size; x++) {
        if (qrcodegen_getModule(qrcode, x, y)) {
          for (int dy = 0; dy < scale; dy++) {
            for (int dx = 0; dx < scale; dx++) {
              lv_canvas_set_px(canvas, shift_x + x *
   scale + dx, shift_y + y * scale + dy, lv_color_black()
   );
            }
          }
        }
      }
    }

    // Release the mutex
    lvgl_port_unlock();
  }
}
```

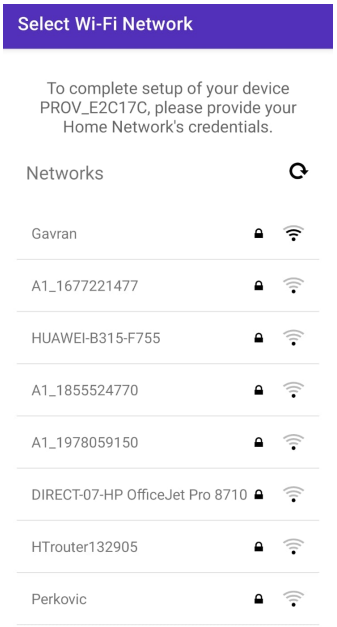
Isječak koda 5.3: Funkcija za prikaz QR koda na zaslonu

Na slikama 5.2 i 5.3 prikazana je mobilna aplikacija te trenutak nakon skeniranja QR koda. Mobilni uređaj zahtijeva spajanje na privremenu pristupnu točku, a iduća slika prikazuje dostupne Wi-Fi mreže u blizini mobilnog uređaja, ujedno i mikrokontrolera, na koje se razvojni sustav može spojiti. Odabirom jedne od mreža i unosom lozinke vjerodajnice se šalju na razvojni sustav.

5.1.2. Registracija u sustav AWS

Nakon uspješnog povezivanja na Wi-Fi, sljedeći je korak registracija uređaja na platformu AWS. Korištena je biblioteka za AWS IoT Fleet Provisioning koja se održava u sklopu projekta otvorenog koda *FreeRTOS* [11]. Biblioteka omogućuje mnoštvu odn. floti IoT uređaja registraciju i instalaciju jedinstvenih certifikata na platformu. Bibliotekom je moguće uređaje registrirati i pomoću autoriziranog korisnika, ali i certifikatima zahtjeva. Ova biblioteka ne ovisi o dodatnim bibliotekama osim standardne C biblioteke i stoga se može koristiti s bilo kojom bibliotekom za MQTT protokol.

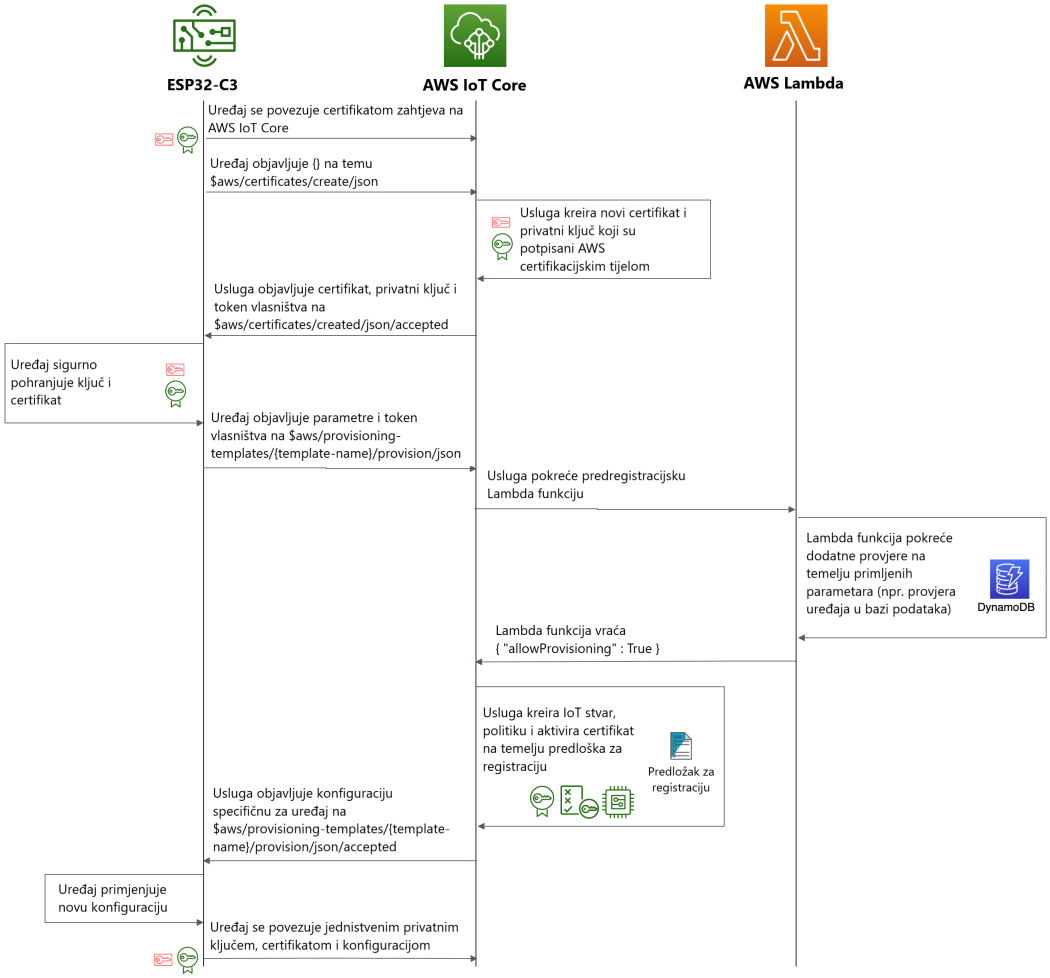
Slika 5.2: Obavijest nakon skeniranja QR koda



Slika 5.3: Odabir dostupne Wi-Fi mreže u blizini

Za potrebe ovog sustava odabrana je registracija pomoću certifikata zahtjeva. Na slici 5.4 prikazan je slijed događaja pri registraciji uređaja. Uređaj se najprije spaja na AWS certifikatom zahtjeva koji unaprijed postoji na mikrokontroleru te ostvaruje MQTT vezu. Zatim objavljuje praznu poruku na temu *$aws/certificates/create/json*, kojom zapravo podnosi zahtjev za kreiranjem vlastitog jedinstvenog certifikata. Usluga kreira traženi certifikat, pripadni ključ te generira značku vlasništva (engl. *ownership token*), što sve skupa i objavljuje na temu *$aws/certificates/created/json/accepted*, na koju je uređaj pretplaćen. Razvojni sustav zatim sigurno pohranjuje dobivene vjerodajnice te objavljuje vlastite parametre i dobivenu značku na temu predloška za registraciju. Ako je kreirana, u sustavu AWS zatim se pokreće Lambda funkcija koja obavlja dodatne provjere nad dobivenim parametrima i tako provjerava valjanost zahtjeva. Primjerice, uređaj pri slanju parametara šalje i hardversku tajnu, koja se može zatim provjeriti u bazi podataka sustava AWS s pohranjenim takvim tajnama i provjeriti valjanost te tajne. Pri uspješnoj provjeri, Lambda funkcija vraća `{"allowProvisioning" : True}`, čime daje konačno zeleno svjetlo za registraciju. Usluga posljedično kreira stvar i pripadnu politiku definiranu u predlošku te aktivira novostvoreni certifikat. Objavljuje novu konfiguraciju specifičnu za uređaj na temu za potvrdu uspješne registracije. Razvojni se sustav, nakon primjene nove konfiguracije, povezuje jedinstvenim privatnim ključem te certifikatom na AWS. Ova se

nova MQTT veza dalje koristi za komunikaciju i razmjenu podataka.



Slika 5.4: Tok registracije uređaja certifikatom zahtjeva [3]

Opisani tok ostvaren je uz pomoć ranije spomenute biblioteke, kao i knjižice *coreMQTT* za komunikaciju putem protokola MQTT s API-jem platforme AWS. Za manipulaciju certifikatima i privatnim ključevima, korištena je biblioteka *corePKCS11*. Ona koristi standard PKCS #11 (engl. *Public-key Certificate Standards*), što je široko korišten API za manipuliranje uobičajenim kriptografskim objektima. Funkcije koje navodi omogućuju aplikacijama korištenje, stvaranje, modificiranje i brisanje kriptografskih objekata, bez izlaganja tih objekata memoriji aplikacije. Primjerice, konkretna integracija s bibliotekom za registraciju uređaja koristi mali podskup PKCS #11 API-ja za, između ostalog, pristup privatnom ključu potrebnom za stvaranje mrežne veze koja je autentificirana i zaštićena protokolom TLS bez da aplikacija ikada dođe u doticaj s ključem [13].

Sljedeći programski odsječak prikazuje uspostavu MQTT sjednice pomoću certifikata zahtjeva te podnošenje zahtjeva za novim certifikatom. U odsječku se također

može primijetiti binarna serijalizacija podataka. Format CBOR (engl. *Concise Binary Object Representation*) Representation), za razliku od formata JSON, pretvara podatke u binarni oblik te ih tako šalje mrežom, što rezultira manjom latencijom i nosivošću (engl. *payload*) [5]. Pri prijenosu velike količine podataka pri ograničenim resursima, poput u IoT sustava, ušteda na veličini poslanih podataka znatno utječe na efikasnost sustava. Isto tako, binaran je format prilagodljiv u odnosu na JSON, gdje mora postojati unaprijed definirana shema za primitak podataka.

```
LogInfo( ( "Establishing MQTT session with claim
  certificate..." ) );
status = EstablishMqttSession(
  provisioningPublishCallback,
   *p11Session,
   pkcs11configLABEL_CLAIM_CERTIFICATE,
   pkcs11configLABEL_CLAIM_PRIVATE_KEY );
status = subscribeToCsrResponseTopics();
status = generateKeyAndCsr( *p11Session,
   pkcs11configLABEL_DEVICE_PRIVATE_KEY_FOR_TLS,
   pkcs11configLABEL_DEVICE_PUBLIC_KEY_FOR_TLS,
   csr,
   CSR_BUFFER_LENGTH,
   &csrLength );
/* Publish the CSR to CreateCertificatefromCsr API. */
PublishToTopic( FP_CBOR_CREATE_CERT_PUBLISH_TOPIC,
   FP_CBOR_CREATE_CERT_PUBLISH_LENGTH,
   ( char * ) payloadBuffer,
   payloadLength );
status = waitForResponse();
```

Isječak koda 5.4: Spajanje certifikatom zahtjeva i zahtjev za novim certifikatom

5.1.3. Očitavanje senzorskih mjerenja

Razvijeni sustav, osim zaslona, sadrži dva senzora koja mjere stanja iz okoline: senzor za temperaturu i vlagu zraka te senzor za vlažnost tla. Mjerenja ovih senzora šalju se na platformu AWS i simuliraju stvarna poljoprivredna mjerenja. Mjerenje temperature i vlage zraka vrši se pomoću senzora DHT11, dok se vlažnost tla mjeri senzorom

VMA303.

5.1.4. Slanje očitanih podataka protokolom MQTT

5.1.5. Ažuriranje softvera

5.2. Programska potpora za oblak

Programska potpora za platformu AWS nije kod u standardnom smislu, no kako bi se uređaj i platforma uopće mogli komunicirati međusobno, potrebno je omogućiti povezivanje i komunikaciju na samoj platformi. Sva se komunikacija odvija u istoj AWS regiji. Zbog dostupnosti većine IoT usluga i relativne geografske bliskosti, korištena je regija *eu-north-1* odnosno Stockholm, Švedska.

Isto tako, važno je istaknuti kako se sve opisane radnje u sustavu AWS mogu izvršavati pomoću naredbenog retka platforme AWS (engl. *Command-line interface - CLI*), no radi jednostavnosti i preglednosti, korišteno je korisničko sučelje platforme.

Programska potpora sastoji se od sljedećih segmenata:

- omogućavanje dinamičke registracije uređaja,
- slanje novog softvera na uređaj,
- pristup zadnjem stanju uređaja nakon gubitka veze,
- obrada podataka dobivenih protokolom MQTT,
- pohrana podataka.

5.2.1. Dinamička registracija uređaja

Kao što je ranije opisano, u razvijenom sustavu koriste se certifikati zahtjeva za prvotno spajanje na platformu AWS, stoga je potrebno kreirati odgovarajući predložak za registraciju. Predložak također mora imati dodijeljenu politiku koja autorizira certifikat zahtjeva i ta se politika dodjeljuje generiranim certifikatima zahtjeva. Odabiru se politike koje omogućavaju točno onoliko koliko je potrebno za spajanje u sustav, a to su dopuštenja za MQTT komunikaciju i spajanje na AWS. Također, potrebno je odabrati koji su certifikati valjani kao zahtjev. Poželjno je i dodijeliti Lambda funkciju kao predregistracijsku provjeru dodatne valjanosti zahtjeva i poslanih parametara. Za razvijeni sustav nije kreirana Lambda funkcija, što znači da je svaki zahtjev automatski odobren. Isto tako, moguće je automatski pri registraciji uređaja kreirati i pripadnu stvar (engl. *thing*), što je korisno za kasniju organizaciju i pregled certifikata. Svaka

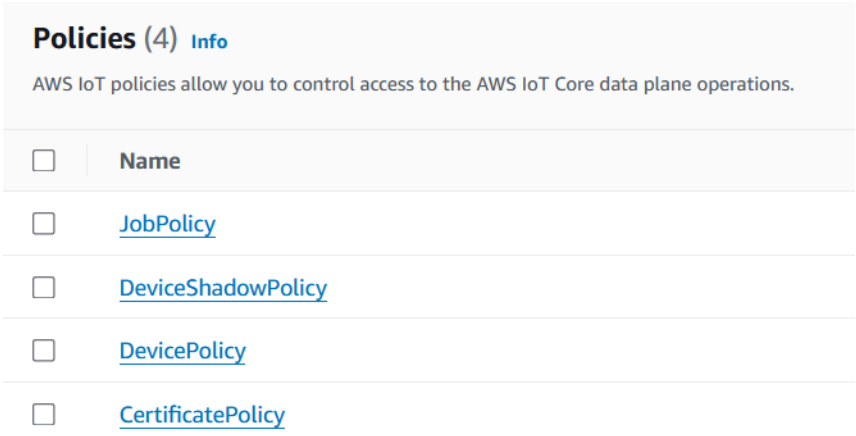
stvar može imati modularan naziv, ovisno o parametrima koji se pošalju. Za ovaj sustav postavljen je prefiks *ESP32Thing_* koji označava da su uređaji vrste ESP32, a ostatak naziva stvari ovisi o serijskom broju samog uređaja. To osigurava da svaki uređaj kreira jedinstvenu stvar u AWS-u. Moguće je odabrati i vrstu stvari (engl. *thing type*) koja će se automatski pridijeliti stvari, te u ovom slučaju kreirana je nova vrsta stvari naziva ESP32-C3. Stvarima se može dodijeliti i grupa, no u ovom sustavu nije bilo potrebe za kreiranjem dodatne grupe stvari budući da se povezuje samo jedna vrsta uređaja u sustav. Naposlijetku, potrebno je odabrati koje će se politike dodijeliti novogeneriranom jedinstvenom certifikatu, čime će uređaj dobiti pristup uslugama sustava AWS. Odabrane politike u ovom sustavu imaju sva dopuštenja radi lakše demonstracije funkcionalnosti. U nastavku se može vidjeti isječak kreiranog predloška, odnosno parametri stvari koja se kreira korištenjem predloška. U odsječku se isto tako može vidjeti funkcija za kreiranje imena stvari koja koristi predefinirani prefiks i serijski broj uređaja.

```
"thing": {
 "Type": "AWS::IoT::Thing",
 "OverrideSettings": {
   "AttributePayload": "MERGE",
   "ThingGroups": "DO_NOTHING",
   "ThingTypeName": "REPLACE"
 },
 "Properties": {
   "AttributePayload": {},
   "ThingGroups": [],
   "ThingName": {
     "Fn::Join": [
     "",
     [ "ESP32Thing_", { "Ref": "SerialNumber" } ]
     ]
   },
   "ThingTypeName": "ESP32-C3"
 }
}
```

Isječak koda 5.5: Odjeljak *stvar* u predlošku za registraciju

Na slici 5.5 nalazi se popis politika koje su dodijeljene uređaju registriranom u

sustav. Omogućene su sve politike, odnosno dodijeljene su sve dozvole koje uređaj može imati. Politika *DevicePolicy* odnosi se na komunikaciju MQTT protokolom, *DeviceShadowPolicy* na akcije vezane uz sjenu uređaja, *JobPolicy* na izvršavanje i dohvat poslova te *CertificatePolicy* dozvoljava povezivanje certifikatom.



Slika 5.5: Popis politika dodijeljenih registriranom uređaju

6. Web aplikacija

Ovdje opisati ukratko koja je svrha same web aplikacije i gdje je deployana.

6.1. Infrastruktura aplikacije

Ovdje opisati stvaranje mrežnog.. nečeg u AWS-u, load balancing i sve što sam već morala kreirati da bih podigla samu aplikaciju na koliko-toliko siguran način.

6.2. Grafana

Gotovo sigurno ću koristiti Grafanu budući da ima gotove vizualizacije koje su meni potrebne. Isto tako, malo opisati funkcionalnosti koje se nude u samoj Grafani, kako funkcionira alerting sustav. Napraviti nekoliko dashboarda, povezati s datasourceom, kreirati par alerta i pokazati kak su došli u obliku maila. Ponuditi proširenja za to - PagerDuty aplikacija recimo.

7. Zaključak

Moj zaključak.

LITERATURA

[1] Aws regions and availability zones. 2024. URL https://medium.com/my-experiments-with-aws/aws-regions-and-availability-zones-973b89f6f24c.

[2] *LVGL*, 2024. URL https://lvgl.io/.

[3] *AWS Documentation*. Amazon.com, Inc., 2024. URL https://docs.aws.amazon.com/index.html.

[4] Stephen J. Bigelow. What is edge computing? everything you need to know. 2021. URL https://www.techtarget.com/searchdatacenter/definition/edge-computing.

[5] C. Bormann. *CBOR*, 2020. URL https://cbor.io/.

[6] Marshall Brain i Talon Homer. How wifi works. 2021. URL https://computer.howstuffworks.com/wireless-network.htm.

[7] Espressif. *Provisioning API*, 2024. URL https://docs.espressif.com/projects/esp-idf/en/v5.1/esp32/api-reference/provisioning/provisioning.html.

[8] *ESP32-C3 Series Datasheet*. Espressif Systems, 2023. URL https://www.espressif.com/sites/default/files/documentation/esp32-c3_datasheet_en.pdf.

[9] *ESP32-C3-Mini 1 Datasheet*. Espressif Systems, 2023. URL https://www.espressif.com/sites/default/files/documentation/esp32-c3-mini-1_datasheet_en.pdf.

[10] *ESP-IDF Programming Guide*. Espressif Systems, 2023. URL https://docs.espressif.com/projects/esp-idf/en/v5.0.2/esp32c3/index.html.

[11] FreeRTOS. *AWS IoT Fleet Provisioning*, 2023. URL https://aws.github.io/Fleet-Provisioning-for-AWS-IoT-embedded-sdk/v1.1.0/.

[12] Google. *Protocol Buffers - Google's data interchange format*, 2024. URL https://protobuf.dev/.

[13] Akashdeep Kashyap. What you need to know about pkcs11? 2024. URL https://www.encryptionconsulting.com/what-you-need-to-know-about-pkcs11/.

[14] Microsoft. Wi-fi problems in your home. . URL https://support.microsoft.com/hr-hr/windows/problemi-s-wi-fijem-i-raspored-va%C5%A1ega-doma-e1ed42e7-a3c5-d1be-2abb-e8fad00ad32a.

[15] Microsoft. What is softap? . URL https://answers.microsoft.com/en-us/windows/forum/all/what-is-softap-what-does-it-do-where-do-i-get-it/e9e0385b-ad1a-446f-8b75-7973326c2629.

[16] *MQTT*. MQTT.org, 2022.

[17] A. S. Gillis N. Barney. Amazon web services (aws). URL https://www.techtarget.com/searchaws/definition/Amazon-Web-Services.

[18] Carol Sliwa Robert Sheldon. Non-volatile storage (nvs). 2021. URL https://www.techtarget.com/searchstorage/definition/nonvolatile-storage.

[19] *IEEE 802.11 Wireless Local Area Networks*. The Working Group for WLAN Standards, 2023. URL https://www.ieee802.org/11/.

[20] Random Nerd Tutorials. Esp32 useful wi-fi library functions. 2021. URL https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/.

Sustav za udaljeni nadzor u poljoprivredi temeljen na platformi ESP32-C3 i AWS-uslugama

Sažetak

Ovo je moj hrvatski sažetak.

Ključne riječi: prva, druga, treća, ključna, riječ

System For Remote Monitoring In Agriculture Based On ESP32-C3 Platform and AWS Services

Abstract

This is my english abstract.

Keywords: first, second, third, key, word