Univerzitet u Sarajevu
Elektrotehnički fakultet

Ugradbeni sistemi (RI/TK) 2024/25

Sistem za upravljanje pristupom garaži pomoću IoT tehnologija

Dokumentacija implementacije

Članovi tima:

Dalila Tanković	19665
Emina Hadžić	19798
Emina Zubetljak	19676

30. 6. 2025. god.

Sadržaj

1 Uvod

Kao što je prethodno navedeno u specifikaciji, cilj ovog projekta je bio da se implementira sistem pametne garaže, koja rješava mnoge korisničke probleme koji se pojavljuju sa korištenjem standardnog sistema. Opremljena sa cjelokupnim sigurnosnim sistemom, garaža omogućava višestruk način za validaciju identiteta vozača. U slučaju pokušaja nedozvoljenog pristupa, aktivira se alarmni sistem koji zaključava garažu, odnosno onemogućen je bilo koji način pristupa istoj. Sistem se uvezuje korištenjem Android mobilne aplikacije, kroz koju je moguće gasiti aktivirani alarm. Također je moguće otvarati i zatvarati vrata bez obzira gdje se korisnik nalazi, i analizirati historiju ulaza u garažu, što postaje naročito korisno u slučaju više korisnika.

Pri tehničkoj implementaciji projekta je posebna pažnja stavljena na efektivnu implementaciju svih dijelova projekta, budući da je zbog postavke problema u mnogim scenarijima neophodno da sistem istovremeno radi više operacija, odnosno da je spreman da odgovori na različite zahtjeve. Također, sam sistem se sastoji od dva mikrokontrolera, odnosno picoETF razvojna sistema, koji kontrolišu vanjski i unutrašnji dio garaže respektivno. Stoga je bilo neophodno omogućiti efektivnu komunikaciju između njih, da bi sistem djelovao kao cjelina. Komunikacija cjelokupnog sistema je bazirana na MQTT protokolu.

U nastavku je detaljno objašnjena izvedba tehničkog dijela, razdvojena po segmentima sistema.

1.1 Korištene biblioteke

U okviru sistema se koriste biblioteka `machine`, za rad sa GPIO razvojnog sistema. U okviru nje su korištene klase `PWM`, `Timer`, te `Pin`.

Korištene su i biblioteke `umqtt.simple`, `network` i `time`, zbog realizacije MQTT protokola i kontrole toka programa sa pauzama.

Dodatne biblioteke uključuju `urandom` za nasumične brojeve i `mfrc522` za rad sa RFID čitačem.

2 MQTT protokol

Kao što je već spomenuto, sistem se sastoji iz dva mikrokontrolera koja međusobno komuniciraju, a oba komuniciraju i sa mobilnom aplikacijom. Za način realizacije spomenute komunikacije odabran je MQTT protokol.

Na oba mikrokontrolera su definisane funkcije `connect_wifi`, koje služe da se mikrokontroler spoji na definisanu WiFi mrežu. Za ovu svrhu se koristi klasa `WLAN` iz biblioteke `network`, zajedno sa metodama `active`, `isconnected` i `connect`.

Navedena funkcija se poziva iz funkcije `mqtt_setup`, u okviru koje se i konfiguriše MQTT klijent. Postavlja mu se odgovarajući callback, u okviru kojeg se parsiraju poruke i njihove vrijednosti, zajedno sa temom na koju su poslane, te se vrše odgovarajuće akcije u sistemu nakon primanja istih. Dakle, poziva se metoda `set_callback` i `connect`. Spajanje se vrši u okviru try bloka, budući da se može desiti da MQTT spajanje ne uspije.

Nakon spajanja je neophodno pretplatiti se na sve teme sa kojih odgovarajući mikrokontroler prima korisne informacije. Za pretplatu na temu se koristi metoda `subscribe`.

Nakon odgovarajuće konfiguracije, mikrokontroler je spreman da reaguje na pristigle poruke kroz poziv metode `check_msg` u glavnoj petlji. U slučaju kada mikrokontroler treba objaviti poruku na neku temu, koristi se metoda `publish`. U dijelovima koji slijede će biti definisana sva komunikacija koja se zapravo odvija u okviru sistema.

3 Validacija pristupa

Sistem može validirati pristup kroz validnu RFID karticu, ili unos tačnog PIN-a na matričnoj tastaturi. U ovoj fazi sistem izvršava nekoliko operacija istovremeno. Naime, sistem mora biti spreman da registruje RFID karticu koja je prislonjena, a također mora reagovati na eventualne pritisnute tipke matrične tastature, koje onda prikazuje na 7-segmentnom displeju. Jasno je da prosto korištenjem pristupa *polling-a*, odnosno prozivanja svake komponente individualno, sistem ne bi bio u stanju da radi efektivno. To se dešava iz razloga što bi se sekvencijalno radila jedna, pa druga operacija. Ova činjenica je pogoršana time da je skeniranje RFID kartice zapravo *blokirajući poziv*, a ne asinhroni. Odnosno, svaki put kad se poziva procedura za čitanje sa RFID čitača, sistem mora *čekati* na rezultat. U slučaju lošeg tajminga bi se vrlo lako moglo dešavati da sistem reaguje presporo, odnosno da eventualno ne registruje pritiske nekih tipki ili skeniranja kartice jer se u tom trenutku „bavio" onom drugom operacijom, pogotovo kad se uzme u obzir da je potrebno softverski riješiti problem *debouncing-a* tipki na matričnoj tastaturi, koje bi uvodile dodatne pauze u okviru izvršavanja glavne petlje. Uz to sve se mora i uzeti u obzir da prikaz cifara na displeju mora biti konzistentan, i dešavati se iznimno brzo, da ne bi došlo do treptanja koje je vidljivo ljudskom oku.

Iz svih navedenih razloga je odlučeno da se problematika riješi primarno korištenjem **sistema prekida.**

3.1 Unos PIN-a

3.1.1 Matrična tastatura

Jasno je da je za unos na matričnoj tastaturi potrebno velikom brzinom skenirati matričnu tastaturu, da bi se prepoznala pritisnuta tipka. Budući da ne želimo da blokiramo glavnu petlju programa, ova procedura se vrši posredstvom *periodičnog tajmera.*

Vođeni iskustvom stečenim u okviru rada na laboratorijskim vježbama, primijećeno je da pri korištenju standardne procedure za čitanje matrične tastature u kombinaciji

sa timerima za displej i unos često izaziva problem *ghosting-a*. Ghosting na matričnoj tastaturi nastaje zbog nepoželjnih povratnih strujnih puteva u nediodiranoj matričnoj tastaturi. U praksi, dešava se da struja „iscuri" kroz susjedne linije i izazove lažni signal na nekom drugom presjeku, te se registruje pritisak više tipki, iako je u realnosti aktiviran samo jedan taster. Stoga je u okviru ovog projekta implementirana nešto kompleksnija, ali znatno pouzdanija „**ghost-free**" implementacija.

Implementacija izvedena u projektu kombinuje nekoliko ideja koje doprinose pouzdanijem skeniranju matrične tastature, a one uključuju:

- **„two-way cross check"**; ova ideja podrazumijeva dvostruko provjeravanje pritisnute tipke, pri čemu se u prvom prolazu pinovi redova koriste kao izlazni i kolone kao ulazni. U drugom prolazu se izvrši *rekonfiguracija* pinova, te se sada redovi koriste kao ulazni pinovi, a kolone kao izlazni.
- **„N-sample streak debounce"**; podrazumijeva da se tastatura skenira više puta uzastopno velikom brzinom, te da se pritisnuta tipka prizna samo ukoliko je u N prolaza registrovana ista tipka svaki put.

Prvi dio je implementiran u okviru pomoćne funkcije `single_scan`, koja služi za jedan „two-way" prolaz matrične tastature. Ona kao povratnu vrijednost vraća **uređenu trojku**, koja predstavlja indeks reda, indeks kolone, te tipku koja se nalazi u tom redu i koloni matrične tastature. Kao što je već rečeno, prvi prolaz služi za odabir kandidata za pritisnutu tipku. Jedan prolaz podrazumijeva standardno skeniranje matrične tastature, u kojem se postepeno svaki red, koji je trenutno izlazni pin, postavlja na high, provjerava se da li je aktivna neka od kolona, koja je deklarisana kao ulazni pin, te ukoliko je pronađen kandidat za tipku, izlazi se iz petlje. Ukoliko nije pronađen nikakav kandidat, ili ukoliko je više kolona detektovano kao aktivno, odmah se izlazi iz funkcije uz vraćanje uređene trojke gdje su svi elementi `None`, budući da je u tom scenariju ili detektovan ghosting, ili nijedna tipka nije aktivirana.

Nakon prvog prolaza se vrši rekonfiguracija pinova na idući način:

```
# rekonfiguracija pinova
for rp in row_pins: rp.init(Pin.IN, Pin.PULL_DOWN)
for cp in col_pins: cp.init(Pin.OUT)
```

Sada se vrši drugi prolaz, pri čemu se u obzir uzima samo kolona koja sadrži kandidata iz prethodnog prolaza. Provjerava se da li je u drugom prolazu detektovan isti kandidat, te se pinovi se vraćaju na izvornu konfiguraciju. Što se tiče detekcije istog kandidata, ukoliko jeste, vraća se trojka sa tim redom, kolonom i znakom, a ukoliko nije, ponovo se vraća trojka `None`.

Rezultat prethodne funkcije koristi funkcija `scan_keypad`, koja predstavlja glavni ISR za matričnu tastaturu, i ona se i prosljeđuje timeru. U okviru ove funkcije se vrši jedan „two-way scan" pozivom pomoćne funkcije, te se globalna varijabla `_streak_count` *postavlja na jedan* ukoliko je ova tipka detektovana po prvi put, odnosno vrijednost `_streak_count` se *uvećava za jedan* ukoliko je ta tipka već detektovana u prethodnom

prolazu. Ovo se vrši posredstvom poređenja sa varijablom `_last_rc`. Zatim, ukoliko je vrijednost `_streak_count` dostigla vrijednost zahtijevanu kroz konstantu `DEBOUNCE_STREAK` (u ovom slučaju 4), zaista se prihvaća ta tipka, uz provjeru da li je u pitanju uzlazna ivica, zbog debouncinga.

Dakle da se sumira, za prihvaćanje tipke se matrična tastatura zapravo skenira *četiri* puta, gdje jedno skeniranje podrazumijeva duplu provjeru uz rekonfiguraciju pinova. Mada se ovo na prvu možda čini suvišno, ovakva implementacija se pokazala kao najbolja što se tiče pouzdanosti. Budući da se period tajmera postavlja na poprilično malu vrijednost od samo 20 ms, procedura se izvršava dovoljnom brzinom da se za korisnika tipka registruje instantno.

Nakon što se tipka registruje, iz ISR se poziva funkcija `handle_key` kojoj se proslijedi registrovana tipka. U okviru ove funkcije se vrši sama logika prepoznavanja PIN-a. Dakle, trenutna tipka se dodaje na uneseni pin, te ako je dužina unesenog pina jednaka predviđenoj od 4 znaka, i ako se pritisne tipka „#", vrši se poređenje sa ispravnim PIN-om.

U slučaju ispravno unesenog PIN-a se isto vizuelno indicira na displeju tako što se 2 sekunde prikazuje blinkanje decimalnih tački. Nakon toga se pušta kratki zvuk sa generatora, i komunicira se sa unutrašnjim mikrokontrolerom da se vrata trebaju otvoriti. Ovo se vrši tako što se na odgovarajuću temu pošalje poruka. Također se šalje vrijednost koja naznačava koji korisnik je pristupio ulazu, da bi se ista informacija mogla zabilježiti na mobilnoj aplikaciji.

U slučaju pogrešno unesenog pina se dešava blinkanje znaka „-" na displeju i pamti se koja je to greška unosa po redu. Ako se desi treća uzastopna greška unosa, pali se alarmni sistem pozivom funkcije `start_alarm`, o čemu će detaljnije biti riječi u nastavku.

3.1.2 7-segmentni displej

Svaka pritisnuta tipka se treba adekvatno i prikazivati na displeju, što se također vrši posredstvom odvojenog tajmera koji je zadužen za ovu funkcionalnost.

Njegov callback je definisan kroz funkciju `display_callback`. Ova funkcija je prosto zadužena za prikaz cifre koja je *trenutno na redu*, što se pamti kroz globalnu varijablu `current_digit`. Drugim riječima, vrši se multipleksiranje za prikaz cifara. Na početku funkcije se gase sve cifre i segmenti. Zatim se ulazni pin parsira tako što se dopuni prazninama ukoliko je dužina trenutno manja od četiri. Trenutna cifra se ažurira tako što se postave odgovarajuće vrijednosti za segmente, koje su definisane u mapi `DISPLEJ_MAPA`. Ona kao ključ prima cifru, a kao rezultat vraća 7 bita koje predstavljaju odgovarajuće vrijednosti za svaki segment. Nakon postavljanja segmenata se pali cifra po redu i prelazi se na iduću cifru ažuriranjem globalne varijable.

Period ovog tajmera je postavljen na 5 ms, da bi se na ovaj način prikaz ažurirao dovoljno brzo da se ne uočava nikakvo treptanje prikaza ljudskim okom.

Što se tiče specijalnih slučajeva za vizuelni prikaz ispravnog i pogrešnog pina, definisane su funkcije `flash_decimal_points` i `blink_minus`. U okviru ovih funkcija se najprije

deinicijalizira tajmer za displej. Zatim se 2s vrši prikazivanje odgovarajućih znakova u `while` petlji analognim pristupom kao ranije. Na kraju funkcija se ponovo inicijalizira tajmer. Na ovaj način je lakše implementirana logika prikazivanja na displeju, budući da je tajmer odgovoran samo za *prikaz pina*, a *ne i za rezultat* završenog unosa. To je urađeno iz razloga što se rezultat unosa treba vizuelno prikazivati 2s, što bi bilo teže uklopiti u logiku postojećeg tajmera.

3.2 RFID čitač

Biblioteka koja je korištena za realizaciju čitača je `mfrc522`, koja odgovara našem modulu. U okviru ove biblioteke se nalazi istoimena klasa, koja se koristi za inicijalizaciju čitača. Njoj se u konstruktoru proslijedi ID za korištene SPI pinove, tj. 0 u ovom slučaju, kao i odgovarajući pinovi.

Na početku funkcije `rfid_callback` se poziva metoda `init` koja resetuje MFRC522 i podešava osnovne registre (takt, modulaciju, timeout), te uključuje antenu za čitanje. Zatim se poziva metoda `request` sa odgovarajućim parametrom. Ona šalje ISO14443-A naredbu „REQIDL" da vidi da li se kartica nalazi u polju. Kao rezultat vraća `stat` (OK ili ERR) i `tag_type` (broj primljenih bita, tj. vrsta kartice). U slučaju da je detektovana kartica, što se prepoznaje po `stat`, poziva se metoda `SelectTagSN`. Ona kombinuje antikoliziju i selekciju, te kao rezultat vraća `uid` (4-bajtni jedinstveni ID kartice) i `stat`, koji ponovo govori je li UID uspješno pročitan.

Ako je UID ispravno pročitan, obrada se nastavlja dalje parsiranjem broja kartice. Lista bajtova `uid` se spakuje u 32-bitni cijeli broj zapisan u little-endian formatu. Zatim se vrši usporedba sa dozvoljenom karticom koja je registrovana u sistemu. Ovisno od toga da li je kartica validna ili ne, vrši se slična procedura kao i kod unosa pina.

Za validnu karticu se pušta kratki zvuk na generatoru, te se na odgovarajuće teme objavljuju poruke o otvaranju vrata, te o korisniku koji je pristupio ulazu. Ako kartica nije registrovana u sistemu, pali se alarm.

Sama procedura se poziva u glavnoj petlji programa svakih 1.5 s. Razlog zašto ova funkcionalnost nije prepuštena tajmeru je već spomenut - funkcije za čitanje u okviru korištene biblioteke su blokirajući pozivi. Stoga, ne bismo zapravo ništa „dobili" korištenjem tajmera. Taj tajmer bi zapravo remetio unos odnosno prikaz pina na displeju, zato jer se presporo izvršava. Budući da je ovo jedina operacija koja se izvršava u glavnoj petlji, pored provjeravanja da li je došla MQTT poruka na neku od tema, i da se ona posredstvom statusnih varijabli zaista izvršava samo kada je neophodno, sistem je i dalje u stanju obavljati sve poslove efikasno.

3.3 Alarm

Aktiviranje alarma se vrši na vrlo jednostavan način kroz funkciju `start_alarm`. Prosto se vrijednost pina postavi na 1, ažurira se statusna varijabla `alarm_active`, te se također deinicijaliziraju aktivni tajmeri za tastaturu i displej, da bi se onemogućio unos

pina dok traje alarm. Statusna varijabla `alarm_active` služi da se onemogući čitanje RFID kartice u glavnoj petlji.

Također se na odgovarajuću temu objavljuje poruka da je alarm trenutno aktivan. Ova poruka služi da bi se posredstvom mobilne aplikacije aktivirala notifikacija koja obavještava korisnika. Notifikacija stiže uz vibraciju mobitela, da bi se lakše uočila. Notifikacija dolazi na uređaj neovisno da li je mobilna aplikacija aktivna u tom momentu ili ne, budući da ona pozadinski sluša na upravo ovu MQTT poruku da bi pokrenula notifikaciju.

Gašenje alarma se vrši klikom na dugme u mobilnoj aplikaciji. Time se zapravo postavlja poruka na temu na koju je vanjski mikrokontroler pretplaćen. Dobijanjem ovakve poruke se aktivira callback za MQTT klijenta definisan u okviru funkcije `on_mqtt`. U slučaju da je primljena ispravna poruka, alarm se gasi funkcijom `stop_alarm`, u kojoj se također sistem vrati na prvobitno stanje sa aktivnim tajmerima i RFID čitanjem.

4 Senzor pokreta

Gledajući funkcionalnosti sistema, senzor pokreta nije neophodan da sistem radi. Razlog zašto je dodan senzor pokreta je zbog *uštede energije*. Naime, pri validaciji unosa sistem mora vršiti nekoliko operacija istovremeno, što kao rezultat može biti poprilično zahtjevno za CPU. Korištenjem senzora pokreta je moguće detektovati momenat kada se vozilo približi, te tek tada aktivirati proces validacije unosa.

Senzor pokreta je zapravo spojen na unutrašnji mikrokontroler, iz razloga što su svi pinovi vanjskog već bili zauzeti. Ultrazvučni senzor ima dva ključna pina, *trigger* i *echo*. Kada senzor primi kratki puls na `trigger`, on transmituje kratki ultrasonični puls, te se nakon toga sluša na eho. `echo` pin dobija vrijednost high koja traje do 25 ms ovisno od distance objekta, a ako se nikako ne detektuje objekat, pin ostane high do maksimalno 38 ms. Mjerenjem *vremena* za koje je vrijednost echo pina jednaka 1, u kombinaciji sa *brzinom zvuka* u zraku, moguće je izračunati udaljenost objekta.

Samo mjerenje distance se vrši u okviru funkcije `mjeri`. Funkcija započinje tako što se šalje kratki puls na pin trigger. Zatim se koristi funkcija `time_pulse_us` iz biblioteke `machine` da se računa ukupno vrijeme za koje je vrijednost echo pina jednaka 1. Distanca se računa kao proizvod tog vremena i brzine zvuka, podijeljeno sa 2, jer nam treba samo distanca u jednom smjeru. Kao rezultat se dobija distanca u cm. Ako distanca padne ispod 7 cm, to je znak da se automobil približio u pogled senzora. Tada se na MQTT temu šalje obavještenje, na osnovu kojeg vanjski mikrokontroler inicijalizira tajmere za unos i počinje sa validacijom.

5 Kontrola garažnih vrata

Fizičko otvaranje i zatvaranje vrata se vrši posredstvom servo motora kojeg kontroliše unutrašnji mikrokontroler. Sam servo motor je deklarisan kao `PWM` izlaz, da bi se mogla kontrolisati željena vrijednost ugla.

Najprije je definisana funkcija koja služi za mapiranje intervala. Servo motor ima opseg ugla od 0°do 180°. Željenu vrijednost ugla je potrebno mapirati na interval [0, 65535], budući da se *duty cycle* zadaje kao nepredznačni broj od 16b. U funkciji `mapiranje_intervala` se primjenjuje vrlo jednostavna formula, gdje su sve navedene vrijednosti parametri iste:

```
return (x - ulazni_min) *
        (izlazni_max - izlazni_min)/(ulazni_max - ulazni_min)
        + izlazni_min
```

Za pravu kontrolu servo motora je definisana funkcija `pisi_servo`. Njoj se kao parametar šalje vrijednost željenog ugla, koji je u već navedenom opsegu. Standardni mikro servo motori, kao onaj koji je korišten u projektu, očekuju PWM signal sa fiksnom frekvencijom od 50 Hz, i sa podesivom širinom impulsa između oko 0.5 ms i 2.5 ms, koje odgovaraju vrijednostima 0°i 180°respektivno. U funkciji se najprije vrši mapiranje ugla na odgovarajuću širinu impulsa, te se vrijednost čuva u varijabli `puls`. Ovaj `puls` je u suštini vrijednost širine impulsa u milisekundama koju servo „razumije" kao svoj ciljni ugao. Zatim se ta širina impulsa pretvara u 16-bitni duty cycle, i ta vrijednost se zapravo postavlja kao duty na pinu servo motora.

Za otvaranje i zatvaranje vrata su definisane funkcije `open_door` i `close_door`. U okviru njih se prosto za zatvaranje vrata poziva funkcija `pisi_servo` sa parametrom 80, odnosno sa parametrom 180 za otvaranje vrata. Ove vrijednosti su se pokazale kao najadekvatnije za fizički servo koji je korišten za ovu realizaciju.

Otvaranje i zatvaranje vrata se primarno kontroliše primanjem MQTT poruka, o čemu je već bilo govora. U unutrašnjosti garaže se još nalazi touch senzor koji pruža alternativni način za postizanje iste svrhe. Na njegovom pinu je omogućeno izazivanje IRQ, a kao handler je napravljena funkcija `touch_senzor`. U njoj se prosto provjerava stanje vrata i izvrši odgovarajuća operacija; ako su vrata trenutno otvorena poziva se funkcija `close_door`, a ako su zatvorena poziva se `open_door`. Stanje vrata se prepoznaje kroz globalnu varijablu `door_open` čija se vrijednost ažurira u funkcijama adekvatno.

Zatvaranjem vrata se također inicijalizira tajmer koji je odgovoran za ultrazvučni senzor, budući da smo tada došli u stanje u kojem bi moglo biti neophodno ponovo čekati na približavanje vozila i unos vrijednosti.

Naposljetku, kao dodatni sigurnosni mehanizam je dodan tajmer nakon otvaranja vrata. Otvaranje vrata inicijalizira „one-shot" tajmer sa periodom 2 minute. U okviru njegovog callback-a se prosto provjerava da li su vrata i dalje otvorena, te ako jesu, zatvaraju se. Razlog za dodavanje ove mogućnosti je u slučaju da korisnik *zaboravi* fizički zatvoriti vrata nekom od akcija, vrata će se zatvoriti sama.

6 Mobilna aplikacija

Detalji implementacije mobilne aplikacije ovdje neće biti u potpunosti navedeni, budući da je ista obavljena u Kotlin programskom jeziku, i suštinski je nevezana za ovaj

predmet.

Aplikacija se sastoji od dva pogleda. Na glavnom pogledu su 4 dugmeta kojim se mogu obavljati operacije već spomenute u prethodnim dijelovima; otvaranje i zatvaranje vrata, gašenje aktivnog alarma, te pregled historije ulaza u garažu. Sve ove opcije koriste MQTT komunikaciju.

Na pogledu historije korištenja se prikazuje lista korisnika uz datum i vrijeme njihovog ulaza. U kodu je definisano ime korisnika koje je vezano za validnu RFID karticu u sistemu. U slučaju da korisnik pristupi garaži koristeći PIN, identitet mu je tada nepoznat, te se registruje samo da je ulaz izvršen posredstvom ukucavanja PIN-a.

Alarmni sistem funkcioniše tako što se korisnik obavijesti o istom bez obzira da li je aplikacija aktivna ili ne. Naime, na prvom pokretanju aplikacije se inicijalizira MQTT klijent i u pozadini se održava konekcija s istim. Stoga, aplikacija može reagovati na pristiglu poruku i kada nije upaljena direktno. Na ovaj način korisnik pouzdano dobija notifikacije kada se desi pokušaj neovlaštenog pristupa.