

UNIVERZA V MARIBORU
FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Denis Krajnc

**BELEŽENJE PRISOTNOSTI ZAPOSLENIH
Z UPORABO KRMILNIKA ESP32**

Diplomsko delo

Maribor, julij 2020

**BELEŽENJE PRISOTNOSTI ZAPOSLENIH Z UPORABO KRMILNIKA
ESP32**

Diplomsko delo

Študent: Denis Krajnc
Študijski program: Visokošolski strokovni
Računalništvo in informacijske tehnologije
Mentor: red. prof. dr. PETER KOKOL, univ. dipl. inž. el.

ZAHVALA

Zahvaljujem se mentorju dr. Petru Kokolu za pomoč, usmerjanja, mnenja in popravke pri pisanju diplomskega dela. Zahvala gre tudi vsem profesorjem in asistentom, ki so me vodili na dosedanji študijski poti.

Velika zahvala gre tudi moji družini, ki mi je omogočila študij.

Beleženje prisotnosti zaposlenih z uporabo krmilnika ESP32

Ključne besede: delovni čas, ESP32, MAC naslov, identifikacija

UDK:

Povzetek:

V diplomskem delu je predstavljena registracija delovnega časa z uporabo krmilnika ESP32, preko katerega zaznavamo identifikacijo z MAC naslovom. V uvodnih poglavjih smo se posvetili teoretičnemu delu naloge. Predstavili smo osnovno delovanje sistema, opisali uporabljene platforme ter predstavili različne vrste identifikacije. Nadaljujemo s podrobno predstavitvijo delovanja aplikacije in potekom dela. Predstavljena je tudi povezava in komunikacija krmilnika z mobilno aplikacijo. Na koncu dela sledi diskusija rezultatov in sklep.

Recording employee work attendance using the ESP32 controller

Keywords: work attendance, ESP32, MAC address, identification

UDK:

Povzetek:

The diploma thesis presents the registration of working hours using the ESP32 controller, which allows the identification with MAC address to be detected. In the introductory chapters we focused on the theoretical part of the thesis. We presented the basic operation of the system, described the platforms used and presented different types of identification. The following chapters give a detailed presentation of how the application works and how the work was organized and carried out. The connection and communication of the controller with the mobile application is also presented. This is followed by a discussion of the results and a conclusion in the final part of the thesis.

KAZALO VSEBINE

1. Uvod.....	1
2. Predstavitev obstoječih sorodnih programskih rešitev.....	3
3. Predstavitev lastne rešitve	5
4. Cilji	6
4.1 Sprotni cilji.....	6
5. Teze.....	6
6. Uporabljena orodja.....	7
6.1 Predstavitev platforme Android.....	7
6.2 Razvojno okolje Android Studio	8
6.3 Razvojno okolje Arduino IDE	9
6.4 MySQL	12
7. Podroben opis obravnavanega problema	14
8. Shema naprav in povezav	15
9. Uporabljena oprema in postopki povezave	16
9.1 Mikroračunalnik	16
9.2 Krmilnik ESP32.....	17
9.3 Povezava krmilnika.....	18
9.4 Protokol TCP/IP	19
9.5 Ustvarjanje zbirke in pripadajočih tabel	20
10. Predstavitev aplikacije	21
10.1 Predstavitev kode krmilnika in njena razlaga	22
10.2 Predstavitev kode, ki skrbi za shranjevanje v podatkovno bazo.....	29
10.3 Predstavitev mobilne aplikacije za generiranje QR kode.....	32
10.4 Predstavitev mobilne aplikacije za branje QR kode in komunikacija s krmilnikom	36
11. Testiranje aplikacije	40
11.1 »Sprejemni test« pri uporabnikih	40
11.2 Lastno testiranje.....	42
12. Sklep.....	44
13. Literatura	46

KAZALO SLIK

Slika 2.1 - Sistem Time&Space podjetja Špica	3
Slika 2.2 - Aplikacija Špica Mobile Time	4
Slika 6.1 - Android logotip	7
Slika 6.2 - Okolje Android Studio	8
Slika 6.3 - Namestitev razvojnega okolja Arduino 1	9
Slika 6.4 - Namestitev razvojnega okolja Arduino 2	9
Slika 6.5 - Namestitev razvojnega okolja Arduino 3	10
Slika 6.6 - Namestitev razvojnega okolja Arduino 4	10
Slika 6.7 - Namestitev razvojnega okolja Arduino 5	11
Slika 6.8 - Logotip MySQL.....	12
Slika 6.9 - Prenos namestitvene datoteke XAMPP	12
Slika 6.10 - Kontrolna plošča XAMPP	13
Slika 6.11 - Omogočanje funkcij MySQL v požarnem zidu.....	13
Slika 6.12 - Podatkovna baza MySQL v vmesniku phpMyAdmin.....	14
Slika 7.1 - Primer QR kode	14
Slika 8.1 - Shema povezav.....	15
<i>Slika 9.1 - Podrobna slika krmilnika [15].....</i>	<i>17</i>
Slika 9.2 - Slika osnovnega vezja	19
Slika 10.1 - Izpis na serijskem vmesniku ob uspešnem zagonu	27
Slika 10.2 - Izpis na zaslonu- začetno stanje	28
Slika 10.3 - Izpis na zaslonu ob zahtevku	28
Slika 10.4 - Podatki za povezavo do podatkovne zbirke.....	29
Slika 10.5 - Poskus vzpostavitve povezave	29
Slika 10.6 – Poizvedba, kjer prevrmo ali je zaposleni prisoten.....	29
Slika 10.7 – Stavek, kjer ustvarimo nov zapis	30
Slika 10.8 - Posodobimo obstoječi zapis	30
Slika 10.9 – Stavki, kjer računamo in posodobimo razliko prisotnosti v urah	31
Slika 10.10 - Obvestilo o uspešnem zaključku	31
Slika 10.11 - Android Studio- ustvarjanje novega projekta	32
Slika 10.12 - Android Studio- izbira imen in lokacije projekta	33
Slika 10.13 - Layout activity	33
Slika 10.14 - Knjižice.....	34
Slika 10.15 – Koda- generiranje QR kode.....	34
Slika 10.16 - Mobilna aplikacija- začetno okno.....	35
Slika 10.17 - Mobilna aplikacija- generirana QR koda	35
Slika 10.18 - Začetno okno mobilne aplikacije.....	36
Slika 10.19 – Knjižica- pomoč pri implementaciji mobilne aplikacije	36

Slika 10.20 - Koda startActivityForResult.....	37
Slika 10.21 - Koda onActivityResult	37
Slika 10.22 - Mobilna aplikacija- uspešno skeniranje QR kode	38
Slika 10.23 - Mobilna aplikacija- zahteva na strežnik	38
Slika 10.24 - Mobilna aplikacija- uspešna registracija	39
Slika 11.1 - Zapis v podatkovni zbirki ob enakem času	43

KAZALO TABEL

Tabela 9-1 Opis krmilnika	18
Tabela 11-1 - Starost in naklonjenost anketirancev	41
Tabela 11-2 Prijaznost aplikacije do uporabnikov	42

SEZNAM UPORABLJENIH SIMBOLOV IN KRATIC

NFC - Near Field Communication

MAC - Media Access Control address

IDE - Integrated Development Environment

QR - Quick Response code

SRAM - Static Random-Access Memory

TCP/IP - Transmission Control Protocol/Internet Protocol

1. Uvod

Napredek v razvoju pametnih naprav in moderen način življenja sta pripeljala do tega, da ljudje v vsakodnevnem življenju uporabljamo vedno več pametnih naprav.

Beležiti želimo prisotnost zaposlenih, kar je v nekaterih podjetjih še vedno velik problem, saj zakonodaja od delodajalca zahteva evidenco o izrabi delovnega časa, ne predpisuje pa načina [1]. Evidenca ni namenjena zgolj kadrovske službi in obračunu plač, temveč je zelo pomembna z vidika pravnega zagotavljanja delovnopравnih pravic, kot so odmori, počitki ter zelo pomembno področje, varnost pri delu, ki temelji na organiziranosti delovnega časa.

Čas je izredno dragocena stvar. Ne moremo ga nadomestiti, zavrteti nazaj ali celo povečati. Kadri so najpomembnejše delovno sredstvo, stroški dela pa največji strošek. Zato vsaka ura in minuta štejeta [1].

Namen evidence delovnega časa je redno spremljanje prihodov ter odhodov zaposlenih, evidentiranje porabe časa za malico, službene ali privatne izhode. Spremljajo se tudi dopusti ter bolniške. V nadaljevanju se bomo osredotočili le na prihode in odhode. Kvalitetno vodenje delovnega časa lahko znatno zmanjša stroške podjetja [1].

Če bi bil delovni čas za vse zaposlene enak, potem z vodenjem evidence ne bi bilo problema. Težava nastopi takoj, ko se delovni čas spreminja.

Delovni čas lahko evidentiramo poljubno. Ročno, z identifikacijskimi karticami oz. ključi, NFC, s pomočjo spletnih aplikacij. Danes najpogostejši način je identifikacija s pomočjo identifikacijskih kartic, kar pomeni, da zaposleni kartico približa terminalu, izbere željeno opcijo in avtomatsko se mu zabeleži prisotnost. Pojavljajo se mnoge zlorabe in iskanje drugih izhodov.

Poraja se kar nekaj vprašanj, kako preveriti ali je to res oseba, ki se je identificirala? S katere lokacije? Velikokrat zaposleni s seboj pozabijo vzeti ali pa izgubijo identifikacijski ključek oz. kartico, kako ukrepati v takšnem primeru? Kaj, če se kartica razmagneti?

Ena od rešitev vseh teh problemov in zastavljenih vprašanj je beleženje prisotnosti s pomočjo pametnega telefona, ki je dandanes naš zvesti spremljevalec. To je identifikacija z uporabo MAC naslova, katerega si bomo podrobneje pogledali v nadaljevanju.

2. Predstavitev obstoječih sorodnih programskih rešitev

Dandanes je na trgu mnogo podobnih informacijskih rešitev, kot so Špica, Četrta pot in mnoge druge [2]. Za začetek bom predstavil napreden sistem Time&Space, vodilnega slovenskega ponudnika sistema za registracijo delovnega časa Špica d.o.o.



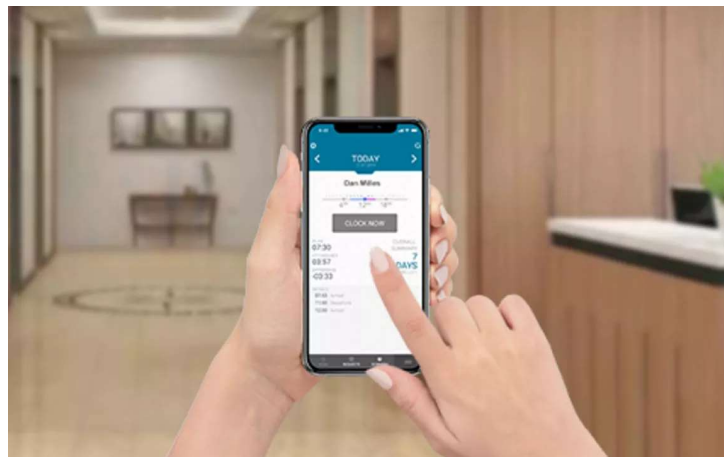
Slika 2.1 - Sistem Time&Space podjetja Špica

Podjetje registracijski terminal Zone Touch opiše kot kompakten in stroškovno učinkovit terminal za registracijo delovnega časa, ki je razvit za popolnoma usklajeno delovanje s programsko opremo. Zaposleni približa kartico terminalu in izbere željeno opcijo.

Po besedah javno dostopnih informacij podjetja natančna evidenca delovnih ur predstavlja osnovo za celovito planiranje, upravlja delovnega časa ter pripravo podatkov za obračun plač.

Sistem prinaša tudi spletno storitev, ki omogoča elektronsko potrjevanje odsotnosti, kar bistveno skrajša in olajša postopek potrjevanja, kot so obračuni plač, evidenca potnih nalogov, planiranje delovnega časa, poslovno analitiko ipd.

Ponujajo tudi mobilno aplikacijo za registracijo delovnega časa, imenovano Špica Mobile Time, ki zaposlenim omogoča natančno upravljanje vseh potrebnih aktivnosti neposredno z mobilne naprave. Prijavijo in odjavijo se lahko kadarkoli, tudi če dela ne opravljajo na običajni lokaciji.



Slika 2.2 - Aplikacija Špica Mobile Time

Največ podjetij se poslužuje identifikacije s pomočjo kartic. Tukaj se poraja kar nekaj pomanjkljivosti, vprašanj, pa tudi izboljšav, npr. kako preveriti identifikacijo zaposlenega, kaj narediti ob razmagnetenju ali zlorabi kartice, kaj če kartico pozabimo doma, s katere lokacije prihaja identifikacija in drugo. Prihaja tudi do množičnih zlorab med zaposlenimi.

Prednosti naprednega sistema podjetja Špica:

- beleženje vsega in povsod,
- upravljanje iz katerekoli lokacije,
- popolna vključenost.

Slabosti naprednega sistema podjetja Špica:

- zlorabe identitete;
- cenovno zahtevna storitev.

3. Predstavitev lastne rešitve

Ena od rešitev, da se zgoraj zastavljenim vprašanjem vsaj do neke mere izognemo, je identifikacija z MAC naslovom. Pomeni, da mora biti uporabnik s pametnim telefonom prisoten na določeni lokaciji v brezžičnem omrežju WIFI. Vsak pametni telefon ima ob povezavi v omrežje enoličen MAC naslov, katerega uporabimo za identifikacijo.

Mikrokontroler ESP32 prebere enoličen MAC naslov, s katerim se zaposleni identificira in podatke shrani v oddaljeno podatkovno bazo.

Ob morebitni prekinitvi povezave z omrežjem (kar pomeni, da je delavec odsoten) dodatno preverjamo prisotnost. Zaposleni s pametnim telefonom skenira generirano QR kodo. Torej se prisotnost zabeleži (prijava/odjava) ob pogoju, da je uporabnik prisoten v omrežju ter poskenira pravilno QR kodo. Posledično rešimo enega od mnogih problemov identifikacije s pomočjo kartic.

Vendar ima tudi ta rešitev kar nekaj pomanjkljivosti, npr. kako evidentirati čas zaposlenih, ki delajo terensko ali od doma.

Če pod problemom potegnemo črto, ugotovimo, da ima vsak način identifikacije po svoje tudi nekaj slabosti.

4. Cilji

Glavni cilj diplomskega dela je predstaviti delujočo aplikacijo za beleženje delovnega časa. Predstaviti teoretični in praktični vidik ter testirati in argumentirati dobljene rezultate.

4.1 Sprotni cilji

- Ustvariti ter povezati krmilnik v omrežje;
- Zaznavanje MAC naslova pametnega telefona s krmilnikom;
- Sprogramirati krmilnik, da bomo z njim lahko shranjevali v podatkovno bazo, beležili in spremljali delovni čas zaposlenih;
- Spisati protokol za komunikacijo med krmilnikom in aplikacijo;
- Testiranje ter argumentiranje rešitev.

5. Teze

Predpostavimo, da zaposleni izgubi povezavo z omrežjem, kar pomeni, da je odstoten. Zato uvedemo dodatno omejitev. Prisotnost (prihod/odhod) se zabeleži samo v primeru, ko je kot dodaten pogoj skenirana pravilna QR koda.

6. Uporabljena orodja

6.1 Predstavitev platforme Android

Android [3] je brezplačna odprtokodna mobilna platforma, ki temelji na operacijskem sistemu Linux. Vključuje operacijski sistem za pametne telefone in ostale prenosne naprave, uporabniške vmesnike, posredniške sisteme ter osnovne aplikacije [4]. Prav zato izbira platforme ni bila vprašljiva, saj večina pametnih naprav deluje na slednjem operacijskem sistemu.

Za razvoj je najzaslužnejše podjetje Google. Platforma je bila predstavljena leta 2007.

Prednosti platforme se kažejo z več vidikov, kot so odprtokodnost, omogoča tudi cenovno ugodno in enostavno razvijanje programov. Prednost občutijo tudi uporabniki, saj so programi in aplikacije za operacijski sistem večinoma brezplačne. Omogoča samodejno sinhronizacijo z Googlovimi storitvami, je odziven ter omogoča večopravilnost.

Operacijski sistem je sestavljen iz petih elementov:

- Aplikacije,
- ogrodja,
- knjižice,
- prevajalnika in
- Linux kernela.

Namen platforme je olajšati razvoj aplikacij na mobilnih napravah ter posledično znižati stroške razvoja. Vse aplikacije so napisane v programskem jeziku Java. [4]



Slika 6.1 - Android logotip

6.2 Razvojno okolje Android Studio

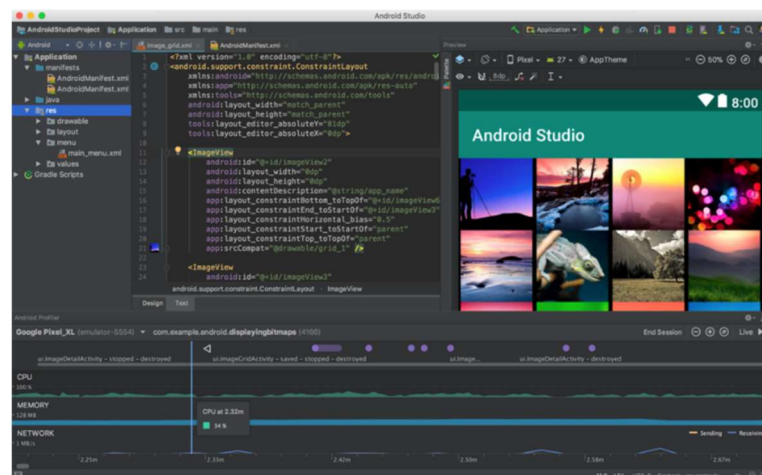
Android Studio [5] je integrirano brezplačno razvojno okolje za razvoj androidnih aplikacij, ki temelji na JetBrains IntelliJ IDEA [6]. Je izpopolnjeno, napredno razvojno okolje za hitro in strukturirano razvijanje mobilnih aplikacij. Je nadgradnja razvojnega okolja Eclipse [4].

Omogoča enostavno vključevanje knjižic in razvoj neodvisno od različice operacijskega sistema Android. Okolje smo podrobneje spoznali skozi izobraževanje na fakulteti, tako je bil razvoj aplikacije precej enostavnejši, saj smo bili že vpeljeni.

Vsebuje tudi preprost emulator, za vse tiste, ki nimajo na dosegu mobilnega telefona ali pa želijo aplikacijo na hitro preizkusiti kar na računalniku. Na voljo je za vse popularne operacijske sisteme Windows, macOS in Linux.

Struktura aplikacije:

- Android Manifest,
- Activity,
- Servis,
- sprejemnik objav,
- ponudnik vsebin.



Slika 6.2 - Okolje Android Studio

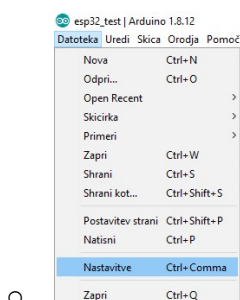
6.3 Razvojno okolje Arduino IDE

Programska koda krmilnika je napisana v programskem okolju Arduino [7]. Okolje temelji na programskem jeziku C. Zasnovo je na precej preprost način. Ukaze definiramo z angleškimi besedami, ki imajo enake pomene kot ukazi. Program pa nato ukaze prevede v programski jezik C.

Pred uporabo mikrokrmilnika ESP32 je potrebno prenesti in nastaviti ESP32 Arduino jedro ter namestiti ustrezne knjižice.

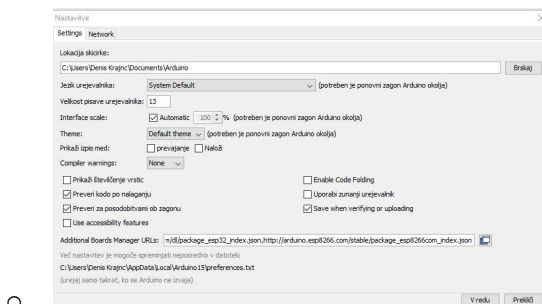
Potek namestitve [8]:

- V programskem okolju Arduino izberemo **Datoteka -> Nastavitve;**



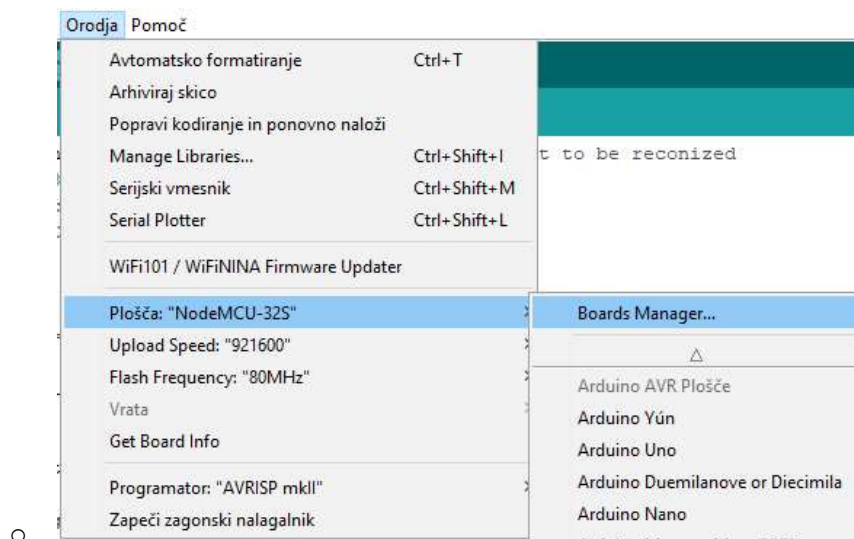
Slika 6.3 - Namestitev razvojnega okolja Arduino 1

- V oknu **Nastavitve** v polje “**Additional Board Manager URLs**” vstavimo spodnjo povezavo do knjižice: https://dl.espressif.com/dl/package_esp32_index.json;



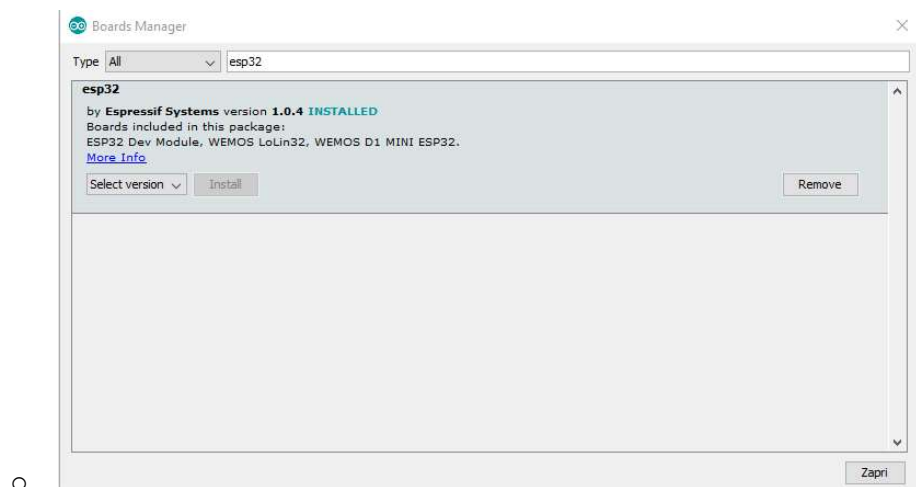
Slika 6.4 - Namestitev razvojnega okolja Arduino 2

- Če imamo v vnosnem polju že kakšno knjižico, jih preprosto ločimo z vejicami kot npr.:
 - https://dl.espressif.com/dl/package_esp32_index.json ,
http://arduino.esp8266.com/stable/package_esp8266com_index.json;
- V naslednjem koraku izberemo **Orodja -> Plošča -> Boards Manager...**;



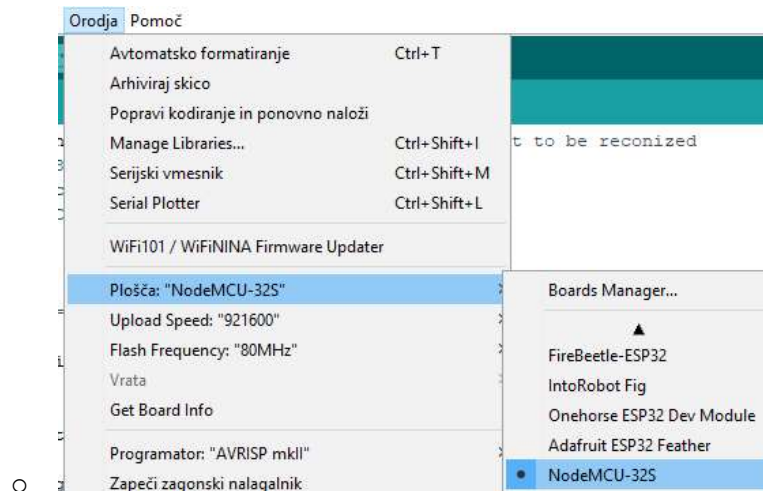
Slika 6.5 - Namestitev razvojnega okolja Arduino 3

- Nato v iskalno polje vpišemo željeno ploščo v našem primeru **ESP32** ter izberemo **namesti**;



Slika 6.6 - Namestitev razvojnega okolja Arduino 4

- Naslednji korak je izbira pravilne ploščice **Orodja -> Plošča**;



Slika 6.7 - Namestitev razvojnega okolja Arduino 5

- Ko vstavimo krmilnik v željena vrata računalnika, se nam samodejno prenese gonilnik, izbrati moramo le še pravilna vrata kot prikazuje zgornja slika;
- Tako imamo pripravljeno vse potrebno razvojno okolje.

Za samodejno izvajanje Arduino programa moramo definirati dve funkciji:

- `setup()` – služi kot začetna inicializacija nastavitvev,
- `loop()` – se izvaja ves čas delovanja.

6.4 MySQL

Ko govorimo o podatkovnih bazah, ki komunicirajo s PHP programsko kodo, ne moremo mimo izredno priljubljene opcije MySQL [9].

MySQL je odprtokoden sistem za upravljanje s podatkovnimi bazami. Omogoča implementacijo relacijske podatkovne baze, ki za delo s podatki uporablja jezik SQL. Omogoča hranjenje velike količine podatkov v pregledni obliki [10].

Deluje na principu odjemalec - strežnik. Strežnik lahko namestimo kot sistem, porazdeljen na več računalnikov [10].

Obstaja mnogo odjemalcev, zbirk ukazov in programskih vmesnikov za dostop do podatkovne baze. V našem primeru bomo uporabili »phpmyadmin« [11]. Vmesnik olajša delo s podatkovnimi zbirkami MySQL, saj ima odličen grafični vmesnik in je enostaven za uporabo.



Slika 6.8 - Logotip MySQL

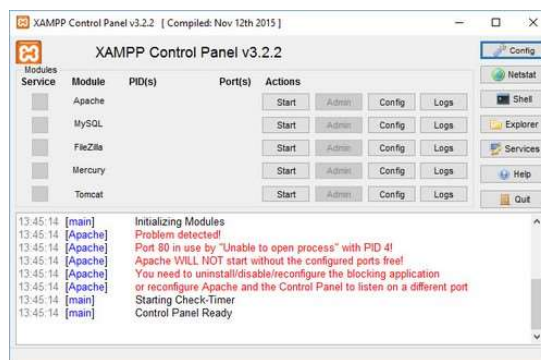
Vzpostavitev podatkovne zbirke pričnemo s prenosom programa s spletne strani apachefriends.org [12]. Izberemo XAMPP za vaš operacijski sistem.



Slika 6.9 - Prenos namestitvene datoteke XAMPP

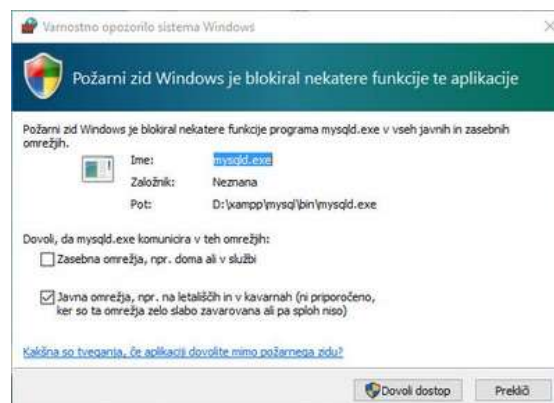
Sledi namestitve okolja. **XAMPP** namestimo v korensko mapo enega od diskov, izberemo namestitvene komponente, sledi izbira lokacije namestitve.

Po uspešni namestitvi okolja je potrebno ročno zagnati strežnik Apache in MySQL. V mapi xampp zaženemo datoteko xampp-control.exe, kliknemo na gumba Start ob oznakah Apache in MySQL.



Slika 6.10 - Kontrolna plošča XAMPP

Ob zagonu strežnika se nam pojavi pojavno okno, kjer omogočimo funkcije MySQL v požarnem zidu.



Slika 6.11 - Omogočanje funkcij MySQL v požarnem zidu

Namestimo še programsko orodje PhpMyAdmin. PhpMyAdmin je orodje v PHP jeziku, ki se uporablja za administracijo MySQL zbirke podatkov preko spletnega brskalnika. Je zelo priročno, saj vsebuje odličen grafični vmesnik.



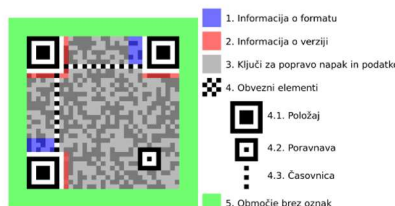
Slika 6.12 - Podatkovna baza MySQL v vmesniku phpMyAdmin

7. Podroben opis obravnavanega problema

Naš namen je beležiti prisotnost zaposlenih. Uporabnikovo prisotnost identificiramo z MAC naslovom.

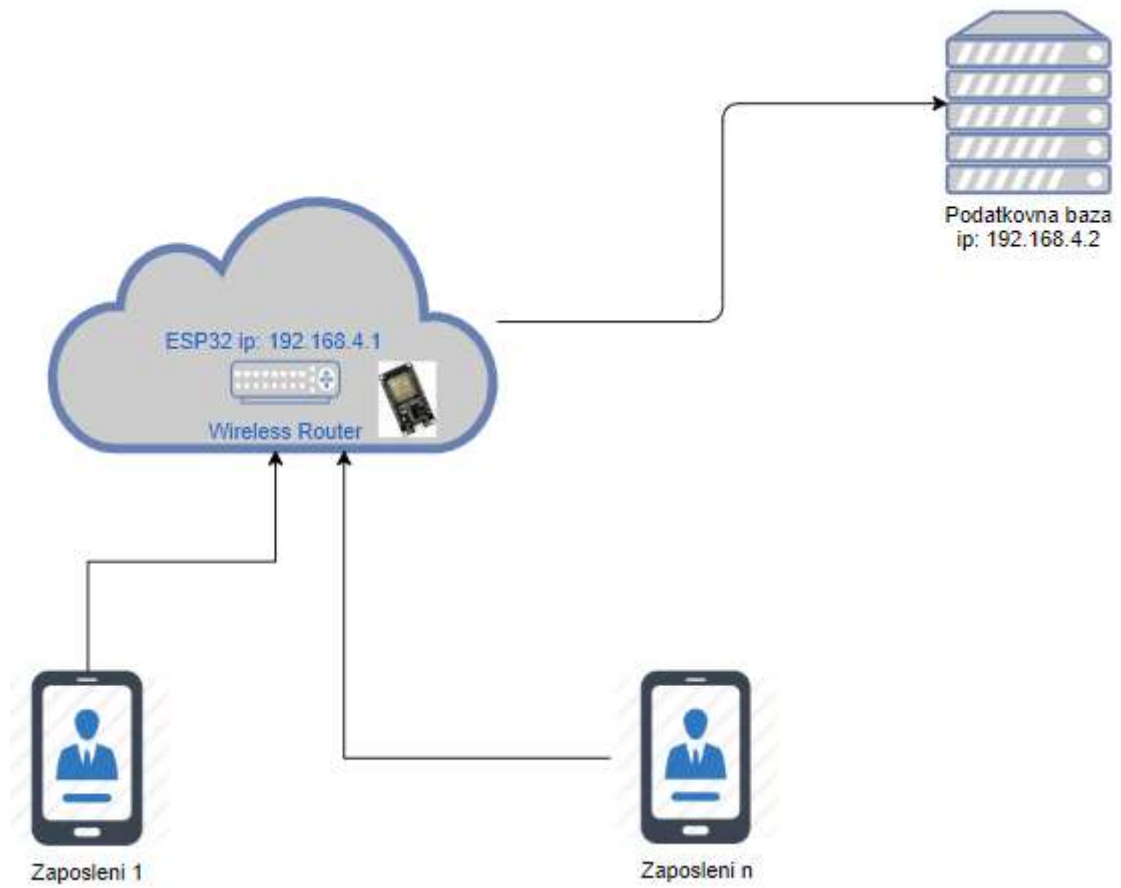
Ko se zaposleni s pametnim telefonom poveže v omrežje, krmilnik avtomatsko prebere enolični MAC naslov. Če se naslov ujema z vnaprej shranjenim uporabnikovim naslovom, se prisotnost prihod/odhod avtomatsko zabeleži v oddaljeno podatkovno bazo.

Da je identifikacija še bolj zanesljiva, mora uporabnik poskenirati spreminjajočo generirano QR kodo.



Slika 7.1 - Primer QR kode

8. Shema naprav in povezav



Slika 8.1 - Shema povezav

9. Uporabljena oprema in postopki povezave

9.1 Mikroračunalnik

»Mikroračunalnik je računalnik, zgrajen na osnovi mikroprocesorja. Sestavljajo ga mikroprocesor, programski in podatkovni pomnilnik ter periferni vmesniki.

Mikroprocesor, ki upravlja delovanje mikroračunalnika, je centralna procesna enota. Vsebuje krmilno enoto, enoto za računanje in obdelavo podatkov ter notranji pomnilnik.

Vsebuje tudi vmesnik za priključitev na zunanjo okolico.

Programski pomnilnik je pomnilnik, iz katerega mikroprocesor bere ukaze, jih interpretira ter izvaja.

Podatkovni pomnilnik je pomnilnik, v katerem so shranjeni podatki, ki jih procesor uporablja za obdelavo.

Periferni vmesniki so vmesniki, s katerimi mikroračunalnik vzpostavlja povezavo z okolico.

Mikrokrmilnik je računalnik, ki ima vse komponente mikroračunalnika vgrajene v en sam čip. Je sistem na integriranem vezju, ki ima poleg vsega naštetega še zunanje enote za komuniciranje z okolico.« [13]

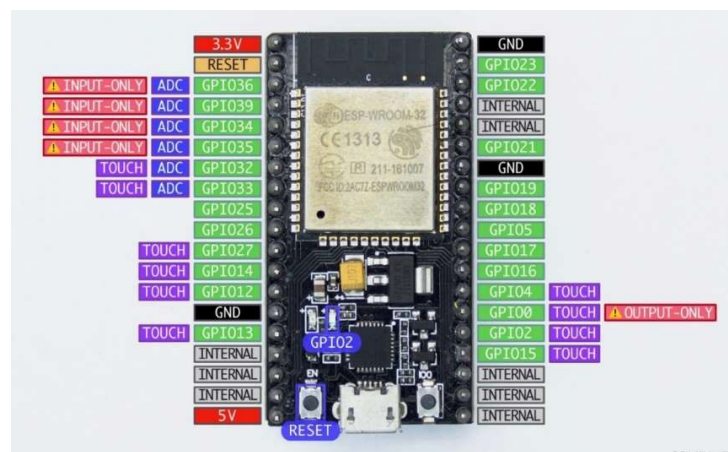
9.2 Krmilnik ESP32

»Na področju integriranih modulov imamo kar nekaj izbire. Razlikujejo se po funkcionalnostih in ceni.« [13]

Uporaba mikrokrmilnikov je čedalje večja in bolj razširjena. Priljubljenost programiranja v Arduino pa vse bolj narašča zaradi preprostosti in velikega nabora senzorjev ter knjižic, ki so na voljo brezplačno, skupaj z literaturo.

Med zelo priljubljenimi mikrokrmilniki je tudi dvojedrni, 32-bitni mikrokrmilnik ESP32. Modul je razvilo kitajsko podjetje Espressif Systems. Vgrajeno ima 520 KiB SRAM ter 16 MiB pomnilnika. Razlogov je več: enostavnost, velika zmožnost razvijanja kot tudi ugodna cena. Omogoča zmožnost brezžične komunikacije, kot sta WIFI in Bluetooth. Integriran vmesnik WIFI deluje v frekvenčnem območju 2.4 GHz in je popolnoma kompatibilen s standardi 802.11b, 802.11g, 802.11n. Podpira protokol TCP/IP in s tem sposobnost pošiljanja podatkov po brezžični povezavi. Omogoča implementacijo realno-časovne programske opreme ter ima analogne nožice [14].

Podpira privzeto programsko okolje ESP-IDF, vendar pa je precej enostavnejše programiranje v Arduino IDE. Omogoča tudi enostavnejši pregled. Velika prednost pa je tudi okolje, ki se neprestano posodablja.



Slika 9.1 - Podrobna slika krmilnika [15]

GPIO0	Se uporablja ob zagonu, lahko ga uporabljamo tudi kot izhodni pin.
GPIO34-GPIO39	Se uporabljajo kot izhodi.
GPIO32.GPIO39	Izhodi za analogno digitalne senzorje, merjenje napetosti.
GPIO2	Pini so povezani z modro LED diodo, podpira tudi binarne senzorje.
5V	Je povezan direktno na USB, kar zagotavlja napajalno napetost 5 V.

Tabela 9-1 Opis krmilnika

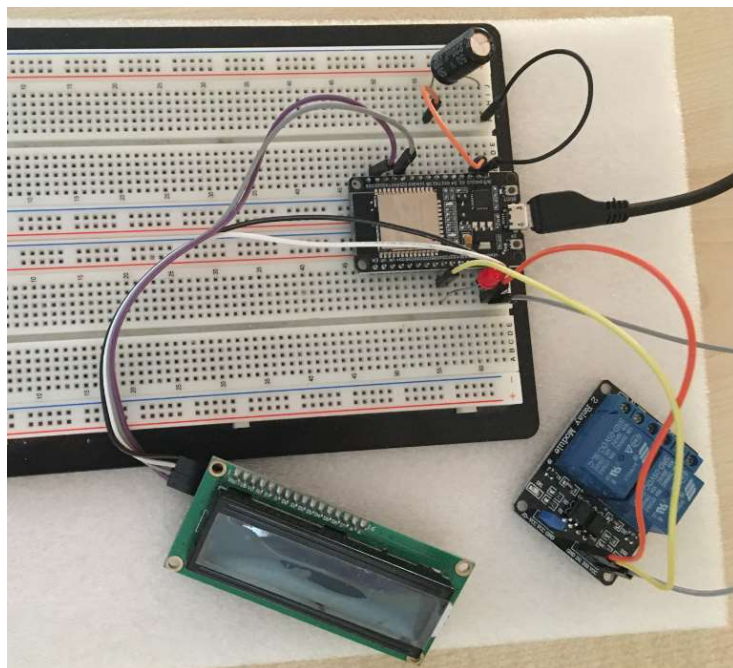
9.3 Povezava krmilnika

Na krmilnik smo priklopili 220uF kondenzator, ki služi za shranjevanje energije, saj krmilniku ob brezžični povezavi v omrežje primanjkuje energije in ob zagonu javlja napako »brownout detector«.

Priklopili smo tudi new LiquidCrystal_I2C display, ki služi za prikaz uspešnega vnosa ter prikaz morebitnih napak. Za delovanje potrebuje LiquidCrystal_I2C knjižico, ki smo jo predhodno namestili v Arduino IDE.

Na sliki vidimo tudi dvojni Relay modul, nanj priklopimo poljubno napravo. V našem primeru bi to lahko bila elektronska ključavnica za vrata. Ko se uporabnik uspešno registrira v sistem, se mu avtomatsko odprejo vhodna vrata podjetja.

Za lažjo simulacijo delovanja pa smo priklopili tudi LED diodo, ki uporabnika obvesti ob morebitnih napakah, kadar sistem ne deluje oz. je v težavah.



Slika 9.2 - Slika osnovnega vezja

Modul ESP32 ob povezavi v omrežje, preko DHCP, dobi IP naslov. Podatke s procesira in jih kot spletni strežnik/servis na vratih TCP 80 s pomočjo PHP kode shranjuje v oddaljeno podatkovno bazo. Shranjevanje krmilimo s pomočjo HTTP zahtevkov. Na strežniku, kjer teče podatkovna baza, moramo nastaviti statični IP naslov.

Modul lahko v omrežje konfiguriramo na več načinov. Problem nastane, ko program prenesemo v drugo brezžično okolje ali ko zamenjamo podatkovno zbirko. Obstaja možnost, dane bo delovalo, saj je skoraj nemogoče, da bi strežnik s podatkovno zbirko dobil enak IP naslov. Ob prenosu v drugo omrežje je najbolje, da na sami postavitvi sistema ponovno nastavimo potrebne parametre in ponovno sprogramiramo krmilnik.

9.4 Protokol TCP/IP

TCP/IP je protokol za nadzor prenosa. Skrbi za komunikacijo med napravami v omrežju. Večina omrežnega prometa poteka preko protokola TCP. Sporočila se preko protokola zaradi vzpostavljene povezave med servisom in odjemalcem prenašajo brezhibno. Informacije se med prenosom ne spreminjajo [16].

Sestavljata ga dva sloja: protokol nadzora nad prenosom TCP in internetni protokol IP.

IP določa način prenosa podatkov preko strojne opreme, obliko informacij ter zagotavlja način prepoznavanja posameznega računalnika v omrežju.

Najprej se vzpostavi povezava med odjemalcem in strežnikom. Pri povezavi se določi odjemalčev in strežnikov IP ter vrata. IP naslov povezan z vrati tvori vtičnico (socket) [16].

Odjemalčeva in strežnikova vtičnica tvorita povezavo TCP, ki je edinstveno določena. Podatki se pošiljajo s pomočjo paketov. Glava paketa vsebuje izvorni IP naslov, izvorna vrata, ciljni naslov in ciljna vrata, zaporedno številko paketa ter število potrditvene in kontrolne zastavice. Ob morebitni napaki pri prenosu, sistem samodejno poskrbi za ponovno pošiljanje informacije. Če so paketi preveliki, jih razbije na več manjših delov [16].

9.5 Ustvarjanje zbirke in pripadajočih tabel

S pomočjo vmesnika »phpmyadmin« smo ustvarili podatkovno bazo z imenom **esp32**, v kateri imamo tabelo **dostopi**. Vanjo shranjujemo vse potrebne podatke pri registraciji delovnega časa.

V njej hranimo:

- id - zaporedna številka zapisa, ki se avtomatsko povečuje;
- uporabnik – uporabniško ime;
- prihod_datum – datum prihoda;
- prihod_ura – ura prihoda;
- odhod_datum – datum odhoda, lahko je null, saj ko uporabnik zabeleži prihod, še ne poznamo njegovega odhoda;
- odhod_ura – ura odhoda;
- razlika – hranimo, koliko polnih ur je bil prisoten;
- prisoten – spremenljivka, v kateri beležimo ali je zaposleni trenutno prisoten.

The screenshot shows the phpMyAdmin interface for a database named 'dostopi'. The table structure is as follows:

#	Ime	Vrsta	Pravilo za razvrščanje znakov	Atributi	Null	Privzeto	Pripombe	Dodatno	Dejanje
1	id	int(11)			Ne	Brez		AUTO_INCREMENT	Spremeni Zavrzni Več
2	uporabnik	varchar(50)	utf8mb4_0900_ai_ci		Ne	Brez			Spremeni Zavrzni Več
3	prihod_datum	date			Ne	Brez			Spremeni Zavrzni Več
4	prihod_ura	time			Ne	Brez			Spremeni Zavrzni Več
5	odhod_datum	date			Da	NULL			Spremeni Zavrzni Več
6	odhod_ura	time			Da	NULL			Spremeni Zavrzni Več
7	razlika	int(11)			Da	NULL			Spremeni Zavrzni Več
8	prisoten	int(11)			Ne	Brez			Spremeni Zavrzni Več

Slika 9.5 - Podatkovna zbirka in pripadajoče tabele

10. Predstavitev aplikacije

Za programiranje krmilnika bomo uporabili kodo, ki jo s programskim okoljem Arduino IDE najprej preverimo, nato pa namestimo na krmilnik. S tem se rešimo morebitnih napak v programski kodi.

V programski kodi so prisotni tudi razni komentarji, saj tako najlažje sledimo, kje se je program ustavil oz. v čem je težava. Kodo smo gradili po zgoraj zastavljenih ciljih. Prvi cilj je bil povezati krmilnik v omrežje, nato smo nadaljevali z branjem MAC naslova, sledila je povezava krmilnika s podatkovno zbirko ter shranjevanje v le-to. Zaključili smo s povezavo mobilne aplikacije s krmilnikom.

V nadaljevanju smo podrobno opisali delovanje programske kode krmilnika [17], programsko kodo, ki skrbi za shranjevanje podatkov v podatkovno bazo ter uporabo knjižic pri mobilni aplikaciji, ki nam precej olajšajo delo.

Koda je komentirana in enostavno prenosljiva.

10.1 Predstavitev kode krmilnika in njena razlaga

//dodajanje potrebnih knjižic

```
#include "WiFi.h"
#include "esp_wifi.h"
#include "ESPAsyncWebServer.h"
#include <LiquidCrystal_I2C.h>
```

//inicializacija prikazovalnika

```
LiquidCrystal_I2C lcd (0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
```

//definiranje pina, na katerega je priklopljen rele

```
const int relay = 26;
```

//parametri brezžične povezave WIFI

```
const char *ssid = "esp32";
const char *password = "krmilnikesp32";
```

//naslov strežnika, na katerem je shranjena podatkovna baza

```
const char* host = "192.168.4.2";
```

//shranjeni MAC naslovi zaposlenih

```
String Tabela_naslovov[3][2] = {
  {"denis", "98:f1:70:30:e8:80"},
  {"peter", "bc:67:78:4c:35:12"},
  {"NAME", "MACADDRESS"}
  ...
};
```

```

//spremenljivka, v katero bomo shranili trenutni prebrani naslov
String naslov = "";

//funkcija, ki procesira MAC naslove
void PrintStations(String ime) //v spremenljivki ime hranimo (ime trenutno identificiranega uporabnika)
{
    wifi_sta_list_t stationList; //spremenljivka, v katero shranjujemo postaje
    esp_wifi_ap_get_sta_list(&stationList); //podatke shranimo v list
    Serial.print("N of connected stations: "); //pomoč pri izpisu
    Serial.println(stationList.num); //število naprav v omrežju

    for (int i = 0; i < stationList.num; i++) //zanka, kjer gremo skozi vse priključene postaje
    {
        wifi_sta_info_t station = stationList.sta[i]; //iz lista vzamemo i-to postajo
        for (int j = 0; j < 6; j++) // zanka
        {
            char str[3]; //polje, v katerega shranjujemo trenutni prebrani MAC naslov

            sprintf(str, "%02x", (int)station.mac[j]); //prva dva znaka
            Serial.print(str); //za pomoč si trenutno stanje izpišemo v serijski vmesnik
            naslov += str; //spremenljivki rekurzivno dodajamo trenutni prebrani niz

            if (j < 5) { //preverimo, če je naslov veljaven
                Serial.print(":"); //MAC naslov je predstavljen z vmesnimi dvopičji, zato jih dodamo
                naslov += ":"; //enako tudi tukaj dodamo podpičje
            }
        }
    }

    for (int i = 0; i < 3; i++) //zanka

```



```

{
    if (Tabela_naslovov[i][1] == naslov) //če se trenutni prebrani naslov ujema s prej
definiranim naslovom
    {
        Serial.println("\n" + Tabela_naslovov[i][0]); //na serijski vmesnik izpišemo ime
identificiranega uporabnika

        Serial.print("connecting to ");

        Serial.println(host); //obvestilo o povezavi na strežnik, kjer je shranjena podatkovna
baza

        WiFiClient client; // uporabimo WiFiClient razred, da ustvarimo TCP povezavo

        const int httpPort = 80; //uporabimo primarni port

        if (!client.connect(host, httpPort)) { //poskušamo se povezati

            Serial.println("connection failed"); //izpišemo morebitne napake pri povezavi

            return;

        }

        if (Tabela_naslovov[i][0] == ime) //če se podatki ujemajo, če se je identificiral registrirani
zaposleni

        {

            //pošljemo zahtevo strežniku s podatki, kje se nahaja datoteka, ki shranjuje podatke v
bazo ter imenom uporabnika, ki se želi registrirati

            client.print(String("GET http://your_hostname/esp32_test/connect.php?") +
               ("&uporabnik=") + Tabela_naslovov[i][0] +
                " HTTP/1.1\r\n" +
                "Host: " + host + "\r\n" +
                "Connection: close\r\n\r\n");

            //če se po določenem času ne odzove, obvestimo zaposlenega in prekinemo povezavo

            unsigned long timeout = millis();

            while (client.available() == 0) {

                if (millis() - timeout > 1000) {

                    Serial.println(">>> Client Timeout !");

                    client.stop();

```

```

    return;
}
}

//preberemo vse, kar nam vrne strežnik in izpišemo na serijski vmesnik
while (client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);
}
Serial.println();
Serial.println("closing connection");
}
}
}

    naslov = ""; //spremenljivko naslov ponastavimo, da lahko obdelamo morebitni naslednji naslov
    Serial.println();
}
Serial.println("-----"); //pomoč pri izpisu
}

void setup() { //začetek Arduino setup zanke
    Serial.begin(9600); //odpremo vrata in nastavimo hitrost na 9600
    WiFi.softAP(ssid, password); //ustvarimo WIFI dostopno točko
    Serial.println();
    Serial.print("IP address: ");
    Serial.println(WiFi.softAPIP()); //na serijski vmesnik izpišemo IP postaje

    pinMode(relay, OUTPUT); //pin z relejem nastavimo na izhod
    digitalWrite(relay, LOW); //postavimo na low
}

```

```

lcd.begin (16, 2); //nastavitveni parametri zaslona

lcd.write("ESP32 - delovni"); //kaj bo na zaslonu izpisano

lcd.setCursor(0, 1); //premknemo se v novo vrstico

lcd.write("cas");

}

void loop() { //zanka, ki se izvaja ves čas

  server.on("/vpis/on/denis", HTTP_GET , [](AsyncWebServerRequest * request) { //če
dobimo zahtevek z »/vpis/on/denis« potem izvedemo naslednje stavke

    request->send(200, "text/plain", "Uspešna registracija!"); //vrnemo uspešna registracija

    PrintStations("denis"); //v obdelavo funkcije pošljemo ime, ki smo ga prebrali

    digitalWrite(relay, HIGH); //vklopimo rele, kar pomeni, da se vrata odprejo

    lcd.clear(); //počistimo trenutni zapis na LCD

    lcd.write("Denis uspesna");

    lcd.setCursor(0, 1);

    lcd.write("registracija!");

    delay(4000); //počakamo 4 sekunde, toliko časa ima uporabnik, da odpre vrata

    lcd.clear();

    digitalWrite(relay, LOW); //vrata ponovno »zaklenemo«

  });

//enako bi naredili z drugega oz. n uporabnikov [18]

  server.on("/vpis/on/peter", HTTP_GET , [](AsyncWebServerRequest * request) {

    request->send(200, "text/plain", "Uspešna registracija!");

    Serial.println("Nt sem");

    PrintStations("peter");

    digitalWrite(relay, HIGH);

    lcd.clear();

    lcd.write("Peter uspesna");

    lcd.setCursor(0, 1);

```

```

lcd.write("registracija!");

delay(4000);

lcd.clear();

digitalWrite(relay, LOW);

});

server.begin(); //poženemo server
delay(1000); //počakamo 1 sekundo
}

```

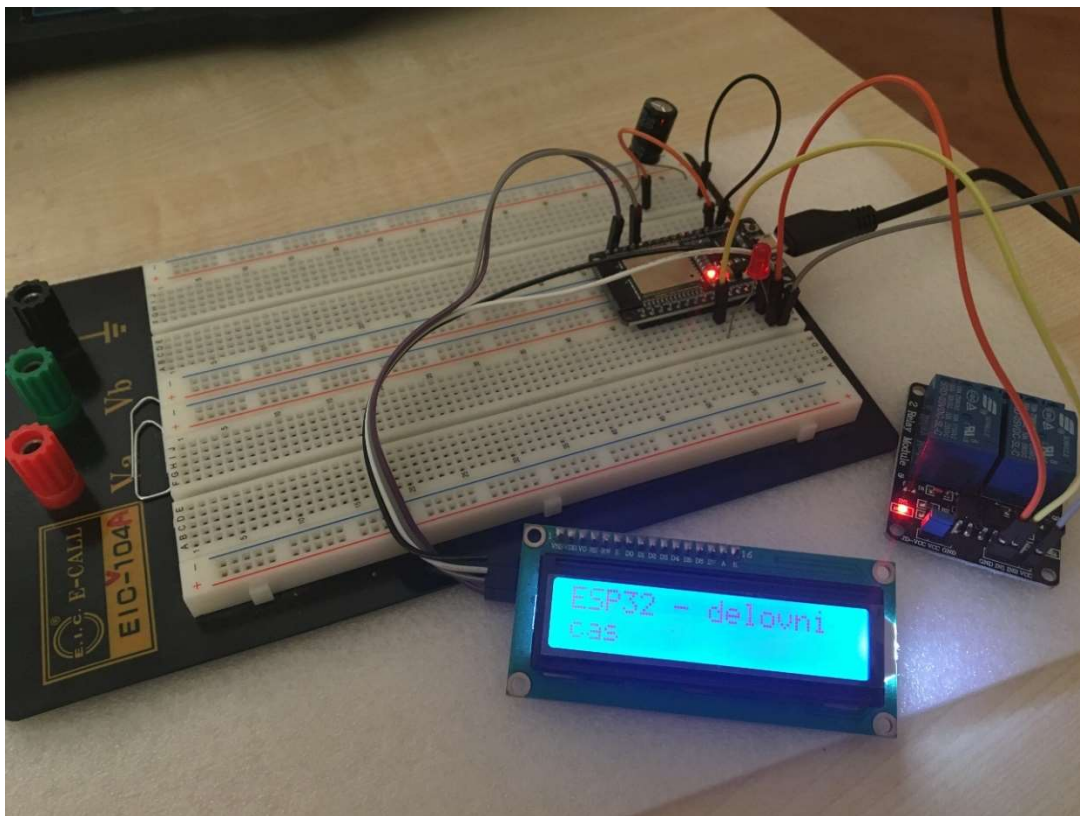
```

COM6
IP address: 192.168.4.1
Nt sem
N of connected stations: 2
cc:2f:71:e1:f7:b2
98:fl:70:30:e8:80
denis
connecting to 192.168.4.2
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Server: Microsoft-IIS/10.0
X-Powered-By: PHP/7.0.30
X-Powered-By: ASP.NET
Date: Mon, 06 Apr 2020 09:32:53 GMT
Connection: close
Content-Length: 88

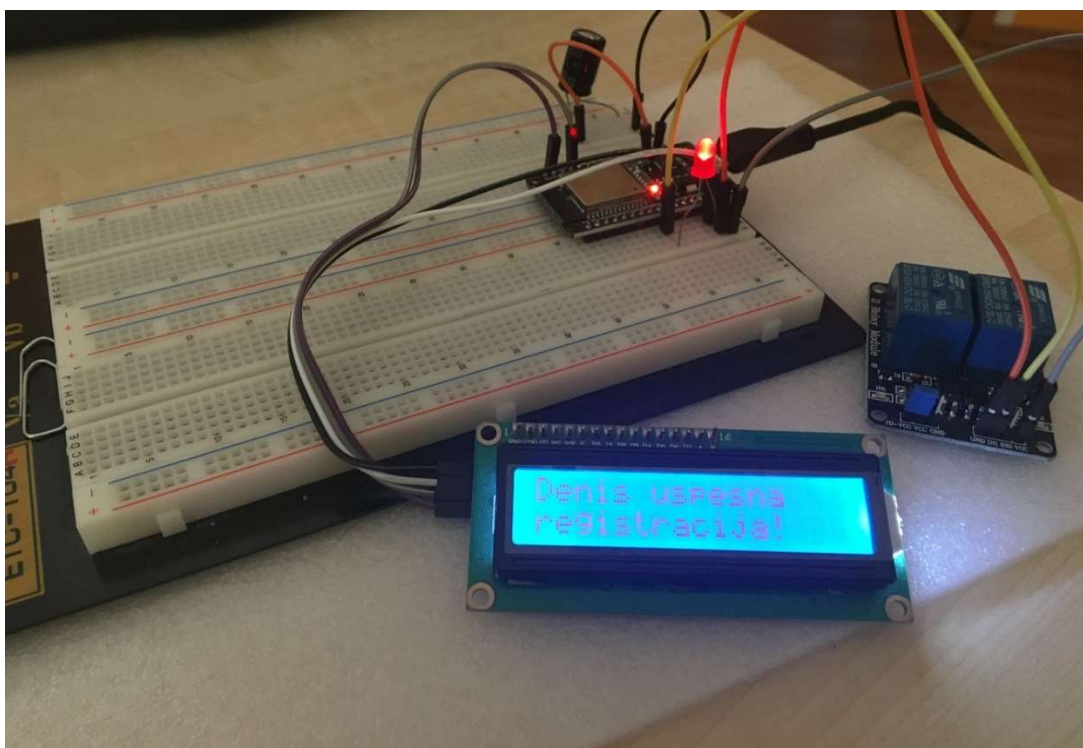
<html>
<body>
    Connection Success!<br><br>Insertion Success!<br></body>
</html>
closing connection
-----

```

Slika 10.1 - Izpis na serijskem vmesniku ob uspešnem zagonu



Slika 10.2 - Izpis na zaslonu- začetno stanje



Slika 10.3 - Izpis na zaslonu ob zahtevku

10.2 Predstavitev kode, ki skrbi za shranjevanje v podatkovno bazo

Dokument pričnemo z osnovno sintakso html. To nam omogoči, da lahko z enostavnim ukazom »echo« zaposlenega sprotno obveščamo o poteku registracije. Izvorna koda prikazuje osnovne podatke za dostop do zgoraj ustvarjene podatkovne baze. Hranimo tudi ime uporabnika, ki ga obdelujemo.

```
<html>
<body>
    <input type="text" value="" />?php
    $dbname = 'esp32';
    $dbuser = 'root';
    $dbpass = 'admin';
    $dbhost = 'localhost';
    $usn = $_GET["uporabnik"];
```

Slika 10.4 - Podatki za povezavo do podatkovne zbirke

Izvorna koda prikazuje vzpostavitev povezave s podatkovnim strežnikom. Uporabimo zgoraj navedene podatke. Ob morebitnih napakah obvestimo zaposlenega.

```
$povezava = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
if (mysqli_connect_errno()) {
    echo 'Ne morem vzpostaviti povezave: ' . mysqli_connect_error();
    exit();
}
```

Slika 10.5 - Poskus vzpostavitve povezave

Ustvarimo poizvedbo. Iz podatkovne zbirke preberemo zadnji zapis, ki se ujema z imenom ter ima parameter prisoten=2, kar pomeni, da je trenutno prisoten in bo šlo za odhod, nato izvedemo poizvedbo.

```
$query = "SELECT prisoten FROM dostopi WHERE uporabnik = '" . $usn . "' AND dostopi.prisoten=2 ORDER BY dostopi.id DESC LIMIT 1";
$result = $povezava->query($query);
```

Slika 10.6 – Poizvedba, kjer preverimo ali je zaposleni prisoten

Če takšnih vrstic ne dobimo, potem pomeni, da uporabnik ni prisoten, gre za prihod. Ustvariti moramo nov zapis. Nastavimo časovni pas ter format datuma in ure. Uporabnika obvestimo o uspešni povezavi ali morebitnih napakah.

V zbirko vstavimo naslednje podatke:

- ime uporabnika,
- datum prihoda,
- ura prihoda in
- prisoten.

```
if (mysql_num_rows($result) == 0) {
    $flag = true;

    $connect = @mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);

    if (!$connect) {
        echo "Error: " . mysqli_connect_error();
        exit();
    }

    echo "Connection Success!<br><br>";

    $uporabnik = $_GET["uporabnik"];
    date_default_timezone_set('Europe/Ljubljana');
    $datum = date('Y-m-d');
    $ura = date('G:i:s');

    $query = "INSERT INTO dostopi (uporabnik, prihod_datum, prihod_ura, prisoten) VALUES ('$uporabnik', '$datum', '$ura', 2)";
    $result = mysqli_query($connect, $query);
```

Slika 10.7 – Stavek, kjer ustvarimo nov zapis

Če takšne vrstice obstajajo, pomeni, da je zaposleni prisoten in pomeni odhod. Ponovno poskušamo vzpostaviti povezavo in javimo morebitne napake. Nastavimo časovni pas ter format datuma in ure. Posodobimo obstoječi zapis z datumom in uro odhoda. Posodobimo tudi parameter prisoten=1, ki pove, da uporabnik trenutno ni več prisoten.

```
else {
    $flag = false;
    $connect = @mysqli_connect($dbhost, $dbuser, $dbpass, $dbname);

    if (!$connect) {
        echo "Error: " . mysqli_connect_error();
        exit();
    }

    echo "Connection Success!<br><br>";

    $uporabnik = $_GET["uporabnik"];
    date_default_timezone_set('Europe/Ljubljana');
    $datum = date('Y-m-d');
    $ura = date('G:i:s');

    $query = "UPDATE dostopi SET dostopi.odhod_datum = '' . $datum . '', dostopi.odhod_ura = '' . $ura . '',
    dostopi.prisoten=1 WHERE dostopi.uporabnik = '' . $uporabnik . '' ORDER BY dostopi.id DESC LIMIT 1";
    $result = mysqli_query($connect, $query);
```

Slika 10.8 - Posodobimo obstoječi zapis

Posodobiti moramo le še razliko med časom prihoda in odhoda zaokroženo na polne ure. Iz zbirke preberemo uro prihoda in odhoda, pretvorimo v sekunde, izračunamo razliko ter zaokrožimo na polne ure. Na koncu le še posodobimo zapis v podatkovni bazi.

```
$poizvedba_odsoten_start = "SELECT prihod_ura FROM dostopi WHERE uporabnik = '$usn' . '' ORDER BY id DESC LIMIT 1";
$ura_odsoten_start = $povezava->query($poizvedba_odsoten_start);

$poizvedba_odsoten_konec = "SELECT odhod_ura FROM dostopi WHERE uporabnik = '$usn' . '' ORDER BY id DESC LIMIT 1";
$ura_odsoten_konec = $povezava->query($poizvedba_odsoten_konec);

$vrstica = $ura_odsoten_start->fetch_assoc();
$vrstica2 = $ura_odsoten_konec->fetch_assoc();

$parsed1 = date_parse($vrstica['prihod_ura']);
$parsed2 = date_parse($vrstica2['odhod_ura']);

$seconds1 = $parsed1['hour'] * 3600 + $parsed1['minute'] * 60 + $parsed1['second'];
$seconds2 = $parsed2['hour'] * 3600 + $parsed2['minute'] * 60 + $parsed2['second'];

//echo '<script type="text/javascript">alert("'.$seconds1.'");</script>';

$interval = $seconds2 - $seconds1;

$previousHour = floor($interval / 3600);

$zapisi_ure = "UPDATE dostopi SET dostopi.razlika = '$usn' . $previousHour . ''
WHERE dostopi.uporabnik = '$usn' . '' ORDER BY dostopi.id DESC LIMIT 1";
$result_ure = $povezava->query($zapisi_ure);
```

Slika 10.9 – Stavki, kjer računamo in posodobimo razliko prisotnosti v urah

Ob uspešnem vnosu ali posodobitvi vrstice uporabnika o vsem obvestimo. Sledi še zaključek html značke.

```
echo "Insertion Success!<br>";
```

Slika 10.10 - Obvestilo o uspešnem zaključku

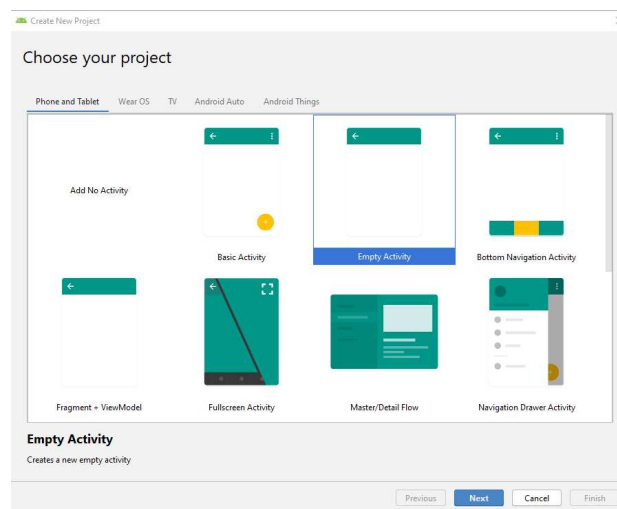
10.3 Predstavitev mobilne aplikacije za generiranje QR kode

Za rešitev hipoteze, ki je bila zastavljena, smo se odločili implementirati mobilno aplikacijo za dodatno preverjanje prisotnosti s skeniranjem QR kode. Vsak dan se avtomatsko generira nova QR koda.

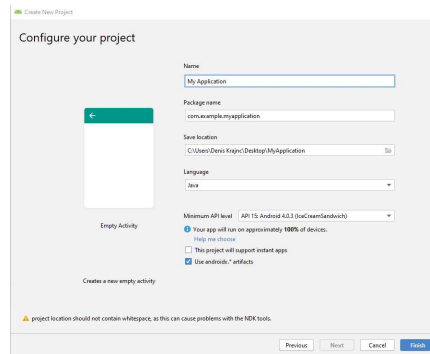
Najprej predstavimo sistem za generiranje QR kode. Lahko bi tudi uporabili katero od mnogih spletnih storitev, ki omogoča generiranje poljubne kode. Aplikacija je izdelana v okolju Android Studio.

V nadaljevanju bomo predstavili izdelavo mobilne aplikacije za generiranje QR kode.

V okolju Android Studio ustvarimo nov projekt. Vnesti moramo ime projekta, zapisati domeno ter izbrati lokacijo, kjer bo projekt shranjen.



Slika 10.11 - Android Studio- ustvarjanje novega projekta



Slika 10.12 - Android Studio- izbira imen in lokacije projekta

Aplikacija omogoča funkcijo. Vsebuje samo eno aktivnost. V postavitvi aktivnosti (layout) imamo:

- EditText – v polje vnesemo besedilo katerega želimo pretvoriti;
- Button – kliče funkcijo za generiranje;
- ImageView – služi za izris generirane QR kode.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="20dp"
    tools:context="com.example.admin.printqr.MainActivity">
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="400dp"
        android:layout_height="300dp"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="25dp"
        app:srcCompat="@mipmap/ic_launcher" />
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/imageView"
        android:layout_marginBottom="10dp"
        android:onClick="onClick"
        android:text="Generiraj" />
    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/button"
        android:layout_marginBottom="3dp"
        android:ems="10"
        android:hint="Enter Text to generate"
        android:inputType="textPersonName" />
</RelativeLayout>
```

Slika 10.13 - Layout activity

Uporabimo Googlove knjižice [19], ki nam precej olajšajo delo, saj so nam v pomoč pri generiranju kode.

```
import com.google.zxing.BarcodeFormat;
import com.google.zxing.MultiFormatWriter;
import com.google.zxing.WriterException;
import com.google.zxing.common.BitMatrix;
import com.journeyapps.barcodescanner.BarcodeEncoder;
```

Slika 10.14 - Knjižice

Ob kliku na gumb si najprej shranimo vneseno besedilo, nato pa ga z uporabo knjižic pretvorimo v sliko.

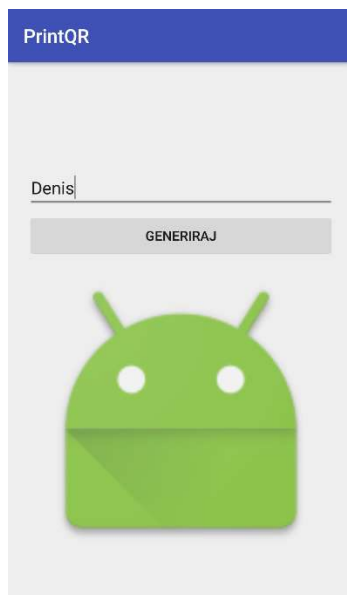
```
public void onClick(View view) {
    TextView text = (TextView) findViewById(R.id.editText);

    String besedilo = text.getText().toString().trim();

    if(text != null)
    {
        MultiFormatWriter multiFormatWriter = new MultiFormatWriter();
        try {
            BitMatrix bitMatrix = multiFormatWriter.encode(besedilo, BarcodeFormat.QR_CODE, width: 500, height: 500);
            BarcodeEncoder barcodeEncoder = new BarcodeEncoder();
            Bitmap bitmap = barcodeEncoder.createBitmap(bitMatrix);
            ImageView image = (ImageView) findViewById(R.id.imageView);
            image.setImageBitmap(bitmap);
        } catch (WriterException e) {
            e.printStackTrace();
        }
    }
}
```

Slika 10.15 – Koda- generiranje QR kode

Aplikacija omogoča vnos željenega teksta, ki ga želimo pretvoriti v QR kodo. S klikom na gumb »generiraj«, se nam generira zelena QR koda.



Slika 10.16 - Mobilna aplikacija- začetno okno

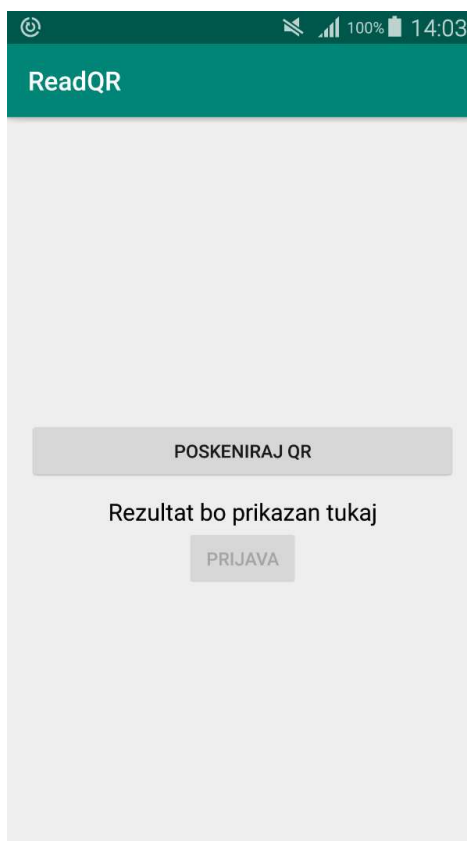


Slika 10.17 - Mobilna aplikacija- generirana QR koda

10.4 Predstavitev mobilne aplikacije za branje QR kode in komunikacija s krmilnikom

Aplikacija vsebuje dve okni, začetni zaslon in zaslon za zajem slike.

Na začetnem zaslonu imamo gumb poskeniraj QR, ki nam odpre drug activity za zajem slike. Sistem ob ujemanju kode avtomatsko zajame sliko in se s funkcijo `onActivityResult()` vrne na začetno okno.



Slika 10.18 - Začetno okno mobilne aplikacije

Enako kot pri prejšnji točki ustvarimo nov projekt ter izpolnimo zahtevana polja.

V pomoč pri skeniranju kode nam je knjižica, ki nam olajša delo.

```
import me.dm7.barcodescanner.zbar.ZBarScannerView;
```

Slika 10.19 – Knjižica- pomoč pri implementaciji mobilne aplikacije

Ob kliku na gumb kličemo `startActivityForResult()` z naslednjimi parametri.

```
btn.setOnClickListener((v) -> {  
    Intent intent = new Intent( packageContext MainActivity.this, ScanActivity.class);  
    startActivityForResult(intent, requestCode 1);  
});
```

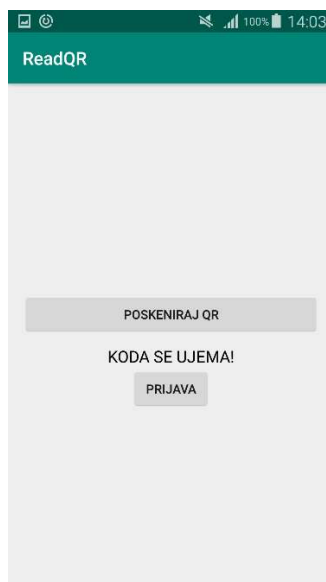
Slika 10.20 - Koda `startActivityForResult`

V funkciji `onActivityResult` pa preverimo ujemanje rezultatov. Če se rezultata ujemata, izpišemo obvestilo »KODA SE UJEMA« in omogočimo gumb »PRIJAVA«.

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
  
    if (requestCode == 1) {  
        if (resultCode == RESULT_OK) {  
            String pattern = "dd.MM.yyyy";  
            SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);  
            String date = simpleDateFormat.format(new Date());  
  
            if(koda.equals(date))  
            {  
                btn_vpis_dostopa.setEnabled(true);  
                tvresult.setText("KODA SE UJEMA!");  
            }  
        }  
        if (resultCode == RESULT_CANCELED) {  
            //napaka  
        }  
    }  
}
```

Slika 10.21 - Koda `onActivityResult`

Če se rezultata ujemata, izpišemo obvestilo »KODA SE UJEMA« in omogočimo gumb »PRIJAVA«. Seveda pa mora biti izpolnjen tudi drug pogoj, povezava z brezžičnim omrežjem.



Slika 10.22 - Mobilna aplikacija- uspešno skeniranje QR kode

Nato aktiviramo gumb prijava. Kliče se http zahtevek na server s podatki in imenom uporabnika.

```
final RequestQueue ExampleRequestQueue = Volley.newRequestQueue( context this );

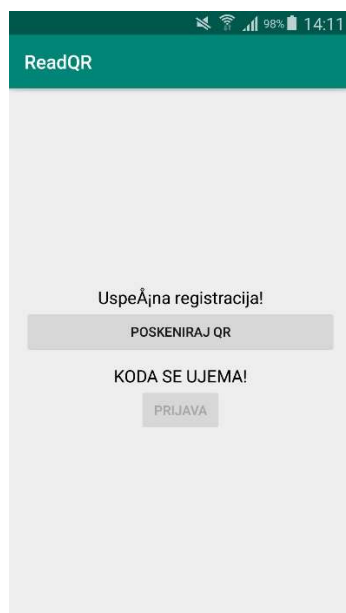
btn_vpis_dostopa.setOnClickListener( (v) -> {
    btn_vpis_dostopa.setEnabled( false );

    String url = "http://192.168.4.1/vpis/on/denis";
    StringRequest ExampleStringRequest = new StringRequest( Request.Method.GET, url, (response) -> {
        //This code is executed if the server responds, whether or not the response contains data.
        //The String 'response' contains the server's response.
        //You can test it by printing response.substring(0,500) to the screen.
        rezultat.setText( response.toString() );
    }, (error) -> {
        //This code is executed if there is an error.
        rezultat.setText( "Napaka" );
    } );

    ExampleRequestQueue.add( ExampleStringRequest );
});
```

Slika 10.23 - Mobilna aplikacija- zahteva na strežnik

Zaposlenega obvestimo o uspešni registraciji ali morebitnih napakah.



Slika 10.24 - Mobilna aplikacija- uspešna registracija

11. Testiranje aplikacije

Pri testiranju aplikacije smo se omejili le na nekaj zaposlenih lokalnega podjetja, ki se ne želijo izpostaviti, ter na lastno testiranje. Tako smo dobili takojšne preverljive rezultate. Za testiranje smo uporabili več pametnih telefonov, ki omogočajo povezavo z brezžičnim omrežjem WIFI.

11.1 »Sprejemni test« pri uporabnikih

Pri testiranju nas je zanimalo predvsem mnenje zunanjih delavcev. V intervjuju s pomočjo anketnega vprašalnika smo želeli preveriti, kako se aplikacija obnese v okolju. Ali služi svojemu namenu, ter predvsem ali je prijazna do uporabnikov in zakaj. Pri testiranju smo zajeli populacijo dvajsetih zaposlenih ter zbrali množico komentarjev.

Najprej smo želeli pridobiti starost anketirancev, tako smo želeli preveriti, kako so starostne skupine naklonjene uporabi modernejše mobilne tehnologije.

V spodnjem grafu vidimo, da je bilo v starostni skupini 20-30 šest anketirancev, kar predstavlja 30 %. V starostni skupini od 30-40 so bili štiri anketiranci, kar predstavlja 20 %. V starostni skupini od 40-50 je bilo sedem anketirancev, kar predstavlja 35 %. Sledi še starostna skupina 50-60, kjer so bili trije anketiranci, kar predstavlja 15 % vseh. Starejša populacija zaposlenih že v začetku ni bila navdušena nad predstavitev aplikacije. Trdili so, da je beleženje prisotnosti s pomočjo kartic bolj zanesljivo, bolj pregledno, enostavno ter predvsem bolj hitro. Medtem, ko je mlajša populacija aplikacijo sprejela odprtih rok, izpostavili so predvsem vpogled v delovni čas s poljubne lokacije.

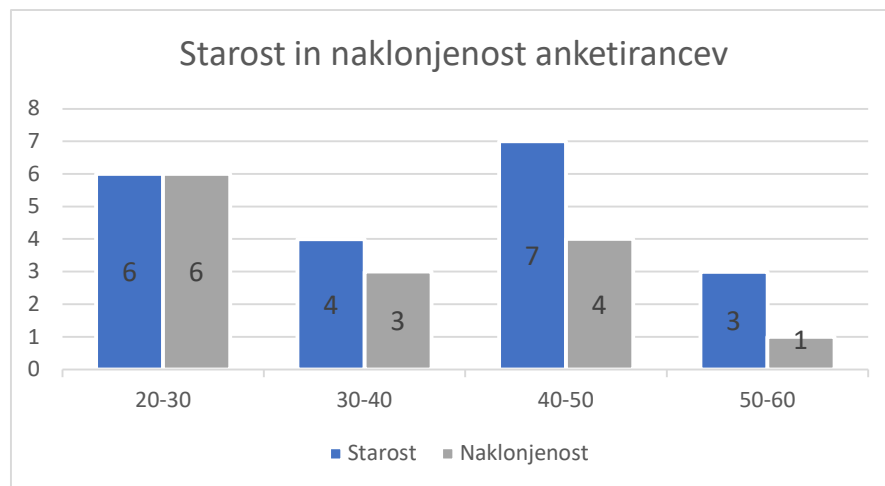


Tabela 11-1 - Starost in naklonjenost anketirancev

Preverili smo še enostavnost ter prijaznost do uporabnikov. Večina zaposlenih je naklonjenih aplikaciji, kar predstavlja 65 % vseh, vpis dostopa je potekal brez težav, izpostavili so predvsem preglednost ter varnost. Druga stran zaposlenih, ki aplikaciji niso naklonjeni, pa so izpostavili zahtevnost, saj je za uspešen vpis dostopa potrebno izpolniti več pogojev, kot so povezava v brezžično omrežje in pa skeniranje QR kode. Izpostavili so, da je vpis dostopa s pomočjo kartic hitrejši.

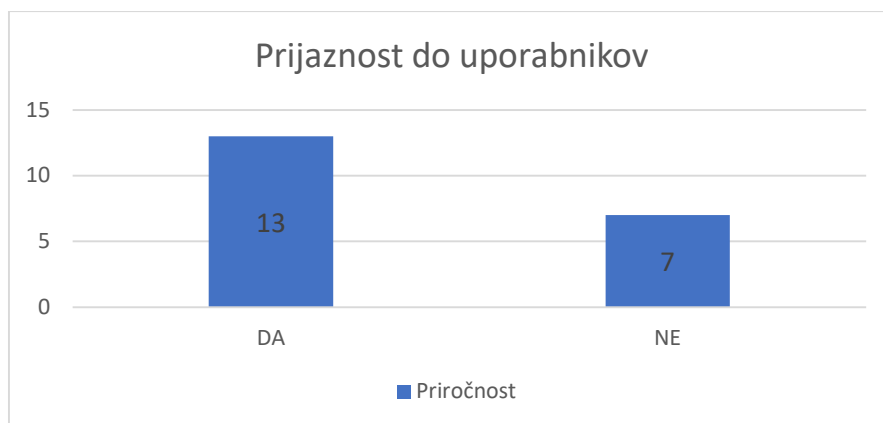


Tabela 11-2 Prijaznost aplikacije do uporabnikov

11.2 Lastno testiranje

Pri lastnem testiranju delovanja kode smo sledili sprotnim ciljem, ki smo jih navedli zgoraj. Pomagali smo si z izpisom raznih komentarjev, nekatere je mogoče videti v sami kodi. Tako smo v programu najlažje sledili, kje je prihajalo do težave. Komentarje izpisujemo v serijskem vmesniku.

Prvi večji cilj je bil uspešno povezati krmilnik v omrežje. S konfiguracijo smo imeli kar nekaj težav, saj krmilnik ob brezžični povezavi porabi precej več energije. V serijskem vmesniku se izpiše napaka »brownout detector«. S pomočjo spleta smo napako odpravili in na krmilnik povezali dodatni kondenzator, ki služi za skladiščenje energije.

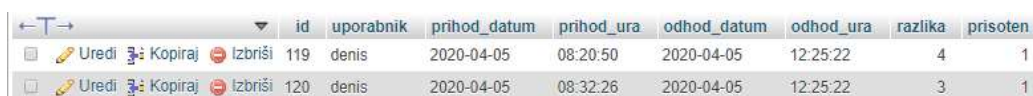
Ko je bil krmilnik uspešno povezan v omrežje, je bil naslednji večji cilj zaznavanje MAC naslova s pomočjo krmilnika. Najprej smo ob uspešnem prebrnem MAC naslovu

poskušali prižgati lučko ter vsemu skupaj s pomočjo komentarjev v serijskem vmesniku slediti. Če se MAC naslov ujema z vnaprej definiranim naslovom, lučko prižgemo.

Nato je sledila povezava s podatkovno bazo. Najprej smo ustvarili podatkovno bazo in vanj s pomočjo preprostejše aplikacije poskušali vstaviti nekaj zapisov. Tako smo se prepričali, da baza deluje in da so podatkovni tipi pravilni. Nato je sledila PHP koda, ki služi za shranjevanje podatkov. Najprej smo poskušali z osnovnim vstavljanjem stavkom »insert«. Nato so sledile razne poizvedbe in posodobitve polj, kot so »select«, »update« in ostalo.

Naslednji cilj je bil povezava krmilnika z mobilno aplikacijo. S pomočjo knjižic, ki olajšajo delo, smo spisali aplikacijo za generiranje in skeniranje QR kode. Nadaljevali smo s pogojem, če se skenirana koda ujema z vnaprej predpisano kodo, uporabnika obvestimo s sporočilom. Dodali smo le še http zahtevek na server in aplikacija je bila pripravljena.

Zaključni testi so pokazali, da je aplikacija enostavna za uporabo in prijazna uporabnikom. Beleženje poteka brez težav. Preizkusili smo tudi varianto kadar dva uporabnika ob enakem času zahtevata vpis dostopa. Vpis se izvede brezhibno, zabeleži pa se enak čas.



	id	uporabnik	prihod_datum	prihod_ura	odhod_datum	odhod_ura	razlika	prisoten
<input type="checkbox"/> Uredi Kopiraj Izbrisi	119	denis	2020-04-05	08:20:50	2020-04-05	12:25:22	4	1
<input type="checkbox"/> Uredi Kopiraj Izbrisi	120	denis	2020-04-05	08:32:26	2020-04-05	12:25:22	3	1

Slika 11.1 - Zapis v podatkovni zbirki ob enakem času

Testirali smo tudi možnost če zaposleni ni prisoten v omrežju, tedaj vpis dostopa seveda ni mogoč. Uporabnika uspešno obvestimo o napaki.

Lastna rešitev ima nekaj prednosti kot tudi slabosti.

Prednosti:

- se vidijo predvsem pri zlorabi identitete, česar se rešimo do določene mere, saj vsak uporablja le svoj pametni telefon;
- identifikacija je možna samo z lokacije podjetja, ki je pokrit z brezžičnim omrežjem.

Ugotovili pa smo tudi nekaj pomanjkljivosti:

- Kako identificirati zaposlenega, ki dela od doma ali pa opravlja terensko delo, in ni prisoten v omrežju. Lahko bi mu omogočili registracijo preko spletne aplikacije, vendar ima tudi ta sistem nekaj pomanjkljivosti;
- Časovna zahtevnost, saj mora biti izpolnjenih več pogojev;

12. Sklep

V diplomskem delu smo se v uvodnih poglavjih posvetili teoretičnemu delu projekta. Predstavili smo osnovno delovanje sistema in aplikacije ter opisali uporabljene platforme.

Sledil je opis praktičnega dela, kjer smo podrobneje predstavili potek razvoja projekta za registracijo delovnega časa z uporabo krmilnika ESP32. Sistem omogoča beleženje prihoda in odhoda zaposlenih. Zelo pomembna funkcionalnost aplikacije je dodaten pogoj skeniranja QR kode, s tem se rešimo mnogih zlorab identitete.

Sistem smo razvili v programskem okolju Arduino IDE in Android Studio. Izvorna koda je podrobno opisana in komentirana. Poskrbeli smo, da je razvojna koda preprosta in prenosljiva.

Nadaljevali smo s predstavitvijo rezultatov lastnega testiranja ter »sprejemnim testom« pri zaposlenih. Na koncu je sledil še sklep.

V prihodnosti nameravamo sistem nadgraditi. Možna je izboljšava grafičnega vmesnika mobilne aplikacije ter dodatne funkcionalnosti sistema.

Omogočili bomo beleženje

- službenih in privatnih izhodov,
- nadur,
- dopustov,
- bolniških odsotnosti,
- ...

Rezultati testiranja aplikacije kažejo primerljive rezultate z obstoječimi aplikacijami. Ima pa vsaka aplikacija po svoje določene slabosti. Težko je razviti sistem, ki bi pokril vse pomanjkljivosti in varnostne luknje.

V delu smo se podrobno spoznali z razvojnim okoljem za programiranje mikrokontrolerov ter okoljem za razvoj mobilnih aplikacij.

Skozi razvoj dela smo pridobili mnoga prepotrebna znanja in veščine ki jih bomo s pridom uporabili v prihodnosti.

13. Literatura

- [1] Univerza v Mariboru, Fakulteta za organizacijske vede. Diplomsko delo. Uporaba sodobnih informacijskih tehnologij za registracijo delovnega časa v podjetju Mojca Plankelj. Dostopno na: <https://core.ac.uk/download/pdf/67608839.pdf> [04.06.2020]
- [2] Špica sistem za registracijo delovnega časa. Dostopno na: <https://www.spica.si/registracija-in-evidenca-delovnega-casa> [04.06.2020]
- [3] Platforma Android. Dostopno na: <https://www.android.com/> [04.06.2020]
- [4] Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko. Diplomsko delo. Razvoj mobilne aplikacije »RUNAPP« v okolju Android Studio Damir Obrán. Dostopno na: <https://dk.um.si/Dokument.php?id=100360> [04.06.2020]
- [5] Razvojno okolje Android Studio. Dostopno na: <https://developer.android.com/studio> [04.06.2020]
- [6] Android Studio. Dostopno na: <https://svet-el.si/revija/programiranje/android-studio-1-programiranje-2/> [04.06.2020]
- [7] Razvojno okolje Arduino. Dostopno na: <https://www.arduino.cc/en/Main/Software> [04.06.2020]
- [8] Nameščanje ESP32 v okolje Arduino IDE. Dostopno na: <https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/> [04.06.2020]
- [9] Sistem MySQL. Dostopno na: <https://www.mysql.com/> [04.06.2020]
- [10] MySQL. Dostopno na: <https://sl.wikipedia.org/wiki/MySQL> [04.06.2020]
- [11] Orodje phpMyAdmin. Dostopno na: <https://www.phpmyadmin.net/> [04.06.2020]
- [12] XAMPP. Dostopno na: <https://nsa-splet.si/mysql/uvod/mysql-uvod-01.php> [04.06.2020]
- [13] Univerza v Ljubljani, Fakulteta za elektrotehniko. Diplomsko delo. Uporaba razvojnega okolja Arduino za izdelavo merilnega vozlišča na modulu ESP8266 Matej Selan. Dostopno na: <https://repositorij.uni-lj.si/Dokument.php?id=86499&lang=slv> [04.06.2020]

[14] Krmilnik ESP32. Dostopno na: <http://esp32.net/> [04.06.2020]

[15] NodeMCU ESP32. Dostopno na: https://esphome.io/devices/nodemcu_esp32.html
[04.06.2020]

[16] TCP/IP. Dostopno na: <https://sl.wikipedia.org/wiki/TCP/IP> [04.06.2020]

[17] ESP32 WiFi MAC Scanner/Sniffer. Dostopno na:
<https://create.arduino.cc/projecthub/p99will/esp32-wifi-mac-scanner-sniffer-promiscuous-4c12f4> [04.06.2020]

[18] Android IOT – Remotely Controlling ESP32. Dostopno na:
<https://hackaday.io/project/168149/instructions> [04.06.2020]

[19] Google knjižica ZXING. Dostopno na: <https://github.com/zxing/zxing> [04.06.2020]