# Dizajn i prototipiranje integracije interneta stvari i mobilnih aplikacija

**Godek, Tomislav**

**SVEUČILIŠTE U ZAGREBU**

**FAKULTET ORGANIZACIJE I INFORMATIKE**

**V A R A Ž D I N**

**Tomislav Godek**

# DIZAJN I PROTOTIPIRANJE INTEGRACIJE INTERNETA STVARI I MOBILNIH APLIKACIJA

**ZAVRŠNI RAD**

**Varaždin, 2020.**

**SVEUČILIŠTE U ZAGREBU**

**FAKULTET ORGANIZACIJE I INFORMATIKE**

**V A R A Ž D I N**

**Tomislav Godek**

**Matični broj: 45967/17–R**

**Studij: Informacijski sustavi**

# DIZAJN I PROTOTIPIRANJE INTEGRACIJE INTERNETA STVARI I MOBILNIH APLIKACIJA

**ZAVRŠNI RAD**

**Mentor:**

Doc. dr. sc. Stapić Zlatko

**Varaždin, rujan 2020.**

*Tomislav Godek*

**Izjava o izvornosti**

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

*Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi*

_____

# Sažetak

U današnje vrijeme Internet stvari (eng. *Internet of Things - IoT*) je gotovo svugdje oko nas. Na što mislim kada pišem o IoT uređajima. To su svi objekti koji su spojeni na neku zajedničku mrežu i međusobno komuniciraju i dijele podatke jedan s drugim. Interakcija s njima je često implementirana preko jednostavnog sučelja na samom uređaju ili preko mobilne aplikacije. Ovaj rad prikazuje postupak dizajniranja i implementacije IoT-a u integraciji s mobilnom aplikacijom. Nakon usporedne analize, u radu će biti odabran jedan skup tehnologija za koje će biti prikazan dizajn i razvoj prototipa jednostavnog sustava podržanog Internetom stvari i upravljanog mobilnom aplikacijom pisanom u programskom jeziku Kotlin. Cilj ovog završnog rada nije detaljno objasniti sveukupnu sliku IoT-a, nego prikazati njenu primjenu i mogućnosti u izradi mobilnih aplikacija. Rezultat ovog rada je prikaz modularne IoT aplikacije bazirane na Bluetooth tehnologiji čija zadaća je olakšati korisniku postupak zalijevanja biljaka u kućanstvu. Odabir tehnologije i uređaja koji se koristili u ovom radu je odabran na temelju statističke analize trenutno najpopularnijih tehnologija te metodom procjenjivanja i prosuđivanja tehnologija koje bi bile najprikladnije za mali domet i relativno brz prijenos podataka.

**Ključne riječi:** dizajn, razvoj, razvoj mobilnih aplikacija, Internet stvari, IoT, Kotlin

# Sadržaj

# 1. Uvod

Tema ovog završnog rada je dizajn i prototipiranje jednostavnog sustava podržanog Internetom stvari (eng. *Internet of Things* - IoT). Detaljno će biti objašnjeno što je to Internet stvar (IoT), kako se danas primjenjuje u stvarnom svijetu i na koji način se pristupa razvoju mobilne aplikacije koja objedinjuje Internet stvari u svojem životnom ciklusu.

Kako bi uopće počeli razvijati IoT sustav potrebno je prvo imati određeni cilj na umu. Razvojni tim ili pojedinac mora prvo odrediti:

- problem koji želi riješiti
- što želi postići rješavanjem tog problema
- koji je najbolji pristup rješavanju takvog problema

Zatim je potrebno provesti istraživanje o postojećim IoT sustavima i kako su implementirani u područjima nad kojim se želi primijeniti naš sustav. Time se smanjuje trošak testiranja koji bi bio potreban ako tim ili pojedinac nije siguran o vrsti senzora, uređaju ili tehnologiji koje će se koristiti u izgradnji vlastitog IoT sustava. Jedan od najvećih problema razvoja IoT sustava je odabir pravog uređaja i tehnologije. Potrebno je imati na umu da upravo odabir IoT platforme definira način i postupak implementacije cijelog IoT sustava. Nagla promjena platforme za vrijeme razvoja aplikacije može biti vrlo skup postupak pa je potrebno na samom početku imati jasnu predodžbu o cilju i potrebama korisnika za koje se kreira IoT sustav.

Prototipiranje IoT sustava nije jednostavan postupak. Zahtijeva razumijevanje uređaja i tehnologija koji se koriste u fazi prototipiranja. Jedan od glavnih ciljeva u toj fazi je ostvariti uspješnu komunikaciju između uređaja, prikupiti podatke pomoću senzora i prezentirati informacije pomoću korisničkog sučelja na ciljanom uređaju (najčešće pametnom telefonu). Prototip nije proizvod spreman za tržište, već robusna verzija sustava sa svim potrebnim komponentama koja vodi prema definiranom cilju koji se želi postići. Danas postoji mnogo predložaka za gotovo svaku IoT tehnologiju što čini razvoj IoT aplikacije mnogo pristupačnijim nego prije. Kasnije u razvoju potrebno je kreirati vlastite principe i rješenja kako bi se što bolje optimizirao sustav i smanjio trošak razvoja aplikacije.

Cilj završnoga rada je sistematizirati i prikazati mogućnosti integracije Interneta stvari s mobilnom aplikacijom. Rezultat takve integracije je mobilna aplikacija koja će integrirati IoT funkcionalnosti i korisnicima donijeti određenu korist i udobnost u svakodnevnom korištenju. U ovom radu je prikazana izrada mobilne aplikacije koja daje korisniku mogućnost zalijevanja

biljaka uporabom intuitivnog i jednostavnog sučelja. Mogućnosti za kasniju nadogradnju su brojne zbog odabira vrlo moćnog mikroračunala i modernog programskog rješenja.

Ovaj završni rad sastoji se od 8 glavnih poglavlja. Uvodno poglavlje opisuje svrhu i cilj diplomskog rada te daje kratak sažetak rada. Drugo poglavlje sadrži opis metoda i tehnika koji su primijenjeni u ovom radu. U trećem poglavlju objašnjeno je što je Internet stvari i njihova povezanost s mobilnim aplikacijama. Na kraju poglavlja dan je osvrt na očekivanja u budućnosti Interneta stvari. Četvrto poglavlje prikazuje postupke u dizajnu mobilne aplikacije koja integrira IoT sustav te sve probleme i poteškoće koje pojedinac ili tvrtka može susresti u fazi prototipiranja IoT sustava. U petom poglavlju opisana je odabrana tehnologija koja se koristila u ovom radu te način pristupa prilikom implementiranja takve tehnologije. Šesto poglavlje daje uvid u razne platforme koje se mogu koristiti u izradi IoT sustava. U sedmom poglavlju prikazan je način izrade određenog prototipa IoT aplikacije. Objašnjen je način ostvarivanja komunikacije i slanja podataka između uređaja te je analiziran programski kôd za mobilnu aplikaciju i IoT uređaj. Zadnje poglavlje sadrži zaključak odnosno sažeti opis cijelog rada i komentar autora na izradu završnog rada.

# 2. Metode i tehnike rada

U izradi ovog završnog rada korištena je metoda istraživanja različitih tema vezanih za pojmove Interneta stvari (IoT) te dizajniranja mobilnih aplikacija za IoT. Metoda pregleda literature korištenja je u poglavlju Interneta stvari i svih njegovih podpoglavlja. Metodom analize dan je opis budućih IoT sustava u poglavlju 3.2 Budućnost IoT-a. Za izradu poglavlja Dizajn i prototipiranje integracije Interneta stvari i mobilnih aplikacija korištena je metoda pregleda literature.

Heurističkim pristupom odabran je ESP32 mikrokontroler kao jedan od uređaja prilikom izrade IoT aplikacije. Metodom analize i pregleda literature opisano je 4. poglavlje *Smart Control* aplikacija. Sva poglavlja vezana uz opis programskog rješenja izrađena su uz pomoć tehnike programiranja i dizajna programskog rješenja. Kotlin kôd je izrađen metodom analize postojećih Kotlin programskih rješenja te analizom službene dokumentacije Kotlin programskog jezika za Android platformu.

Za samu izradu aplikacije korišten je Android Studio, službeni IDE za Googleov operativni sustav Android napisan u programskom jeziku Kotlin. Platforma koja se koristila za upravljanje EPS32 sustava je Arduino. Arduino IDE omogućuje pisanje kôda u programskom jeziku C prilagođenom za upravljanje mikrokontrolerske pločice i svih komponenti spojenih na pločici. Glavna tehnologija korištena za ostvarivanje komunikacije između uređaja je klasični Bluetooth.

# 3. Internet stvari

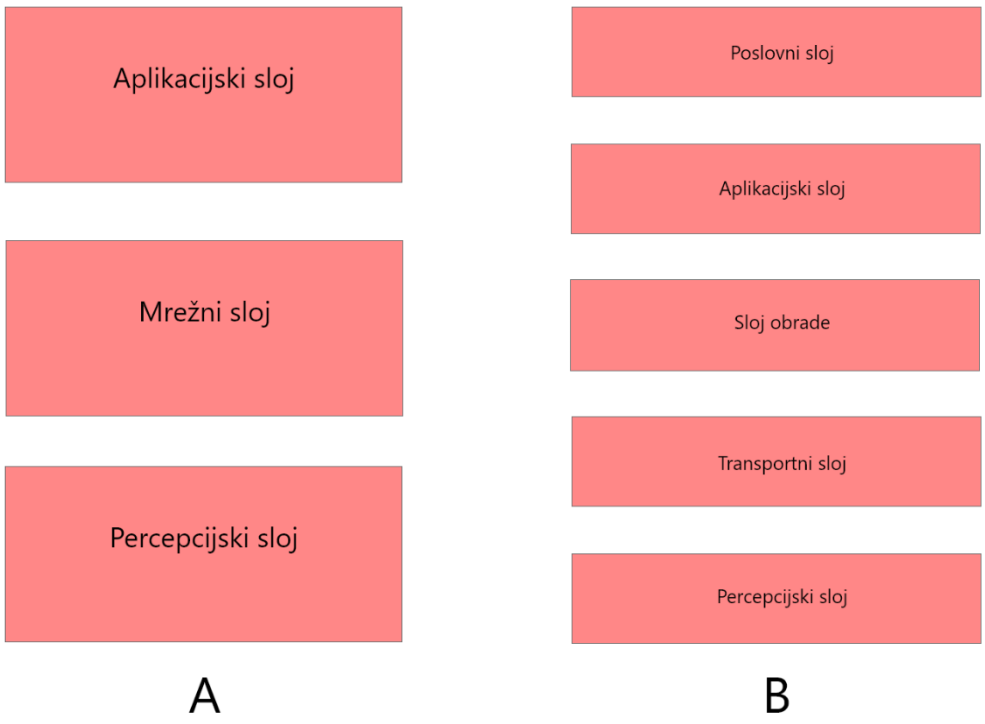
Danas je Internet stvari (IoT) vrlo poznata riječ. Nema službene definicije jer označava veliki skup stvari. Iako je Kevin Ashton među prvima definirao IoT, mnogi drugi su izveli svoje vizije o definiciji IoT-a. Zajedničko svojstvo svim definicijama je ideja da prva verzija Interneta stvari stoji za svu kolekciju podataka prikupljenih od ljudi, dok druga verzija stoji iza svih podataka stvorenih od stvari (Somayya Madakam et al., 2015). U časopisu *Internet of Things (IoT): A Literature* autori *Review* Somayya Madakam, R. Ramaswamy, i Siddharth Tripathi izveli su vrlo dobru definiciju u kojoj pišu da je IoT otvorena i sveobuhvatna mreža inteligentnih objekata koji imaju sposobnost automatske organizacije, dijeljenja informacija, podataka i resursa te reagiranja i djelovanja u situacijama i promjenama u okolišu (Somayya Madakam et al., 2015).

Često se IoT povezuje s različitim sustavima i tehnologijama. U nekim slučajevima se smatra da je samo web baziran, da pruža spajanje realnog i virtualnog svijeta ili mogućnost povezivanja uređaja. Sve navedeno spada u IoT i moguće je stvoriti jedinstveni sustav koji povezuje nekoliko različitih sustava ili implementira samo jedan od sustava. Tako se IoT sastoji od mreže fizičkih objekata, uređaja i vozila koje integriraju razne senzore, programsku podršku i mrežnu povezanost kako bi omogućile tim objektima kolekciju i razmjenu podataka (Muntjir et al., 2017). Internet stvari zapravo čine most između fizičkog svijeta i računalno-baziranih sustava u kojem se stvara bolja produktivnost, efikasnost i ekonomski dobitak. Kada se u IoT sustav uz funkcionalnost razmjene podataka pridodaju senzori i aktuatori tada se radi o instanci pametnog sustava u kojoj se kreiraju pametne kuće, pametne mreže, pametni transport te pametni gradovi (Muntjir et al., 2017).

Kako bi se IoT uspješno implementirao potrebna je dostupna aplikacija za jednostavno upravljanje sustavom, zaštita podataka i privatnost, učinkovita potrošnja energije uređaja unutar sustava, laka iskoristivost aplikacije za krajnje korisnike te pristup otvorenom i interoperabilnom sustavu u oblaku (*cloud system*) (Somayya Madakam et al., 2015). Uspješno rješavanje svih navedenih uvjeta doprinijet će kreiranju vrlo sofisticiranog i robusnog IoT sustava koji će stvoriti profit i zadovoljstvo svojim korisnicima.

Arhitektura IoT-a se prema  Qian Xiao Cong i Zhang Jidong može podijeliti na 3 sloja (Muntjir et al., 2017). Percepcijski sloj predstavlja najniži sloj (eng. *Perception layer*), a služi za prepoznavanje i prikupljanje informacija o objektima koji se promatraju. U sredini je mrežni sloj (eng. *Network layer*), koji se sastoji od fonda otvorene povezanosti (eng. *Open Connectivity Foundation*), mreže mobilnih i fiksnih telefona te zatvorenih IP podatkovnih mreža. Na samom vrhu je aplikacijski sloj (eng. *Application layer*) gdje aplikacije upravljaju ostatkom sustava. Kako je IoT napredovao, a potreba za sigurnosti je postala sve veća, predložena je 5-slojna arhitektura. Na samom vrhu iznad aplikacijskog dodan je poslovni sloj (eng. *Business layer*)

koji upravlja aplikacijama i uslugama IoT-a, a odgovoran je za sva istraživanja vezana uz IoT (Muntjir et al., 2017). Na tom sloju sastavljaju se različiti poslovni modeli za učinkovito donošenje poslovnih strategija. Između aplikacijskog i percepcijskog sloja dodan je transportni sloj i sloj obrade. Transportni sloj (eng. *Transport layer*) je zadužen za primanje informacija u obliku digitalnih signala iz sloja percepcije i njihovo prosljeđivanje u sustave za obradu uz pomoć transmisijskih medija poput WiFi-a, Bluetooth-a, GSM-a (eng. *Global System for Mobile Communications*), itd (Muntjir et al., 2017). Sloj obrade (eng. *Processing layer*) obrađuje podatke prikupljene od percepcijskog sloja. Neke od tehnologija koje su uključene u taj sloj su obrada u oblaku (eng. *Cloud processing*) i sveprisutno računanje (Muntjir et al., 2017). Na kraju se podaci obrađuju i poduzima se neka vrsta automatizacije na temelju obrađenih informacija. Na slici 1 prikazana je 3-slojna i 5-slojna arhitektura interneta stvari.




Slika 1: IoT arhitektura (A: 3-slojna i B: 5-slojna) (Muntjir et al., 2017)

## 3.1. IoT tehnologija

IoT je iniciran od RFID (*Radio-frequency identification*) zajednice čiji članovi su tvrdili da se može pronaći podatak o označenom objektu pregledavajući Internet adresu ili bazu podataka koji odgovara traženom RFID-u ili sličnoj stvari koja koristi tehnologiju bliskog polja (Somayya Madakam et al., 2015). RFID je bežična upotreba elektromagnetskih polja za prijenos podataka u svrhu automatskog prepoznavanja i praćenja označenih predmeta. Primjenjiv je u širokom spektru aplikacija poput nadzora pacijenta, procesu distribucije te vojnih aplikacija. RFID zajednica je došla do zaključka da je ključ IoT-a RFID, zajedno sa senzorskom tehnologijom, nano tehnologijom i tehnologiji ugrađenoj u inteligenciju (*Artificial intelligence* - AI) (Farooq et al., 2015). Kasnije se pojavila komunikacija niskog polja (eng. *Near Filed Communication* - NFC) koja se koristila na mobilnim uređajima za očitavanje NFC oznaka. Uz pomoć NFC-a pojednostavilo se obavljanje transakcija te omogućilo jednostavnu razmjenu digitalnog sadržaja. Nakon uspostave komunikacijskih mreža i globalnog roaminga to je predstavilo novi val tehnologija (Somayya Madakam et al., 2015)

Tehnologija Interneta stvari primarno je bazirana na bežičnim mrežama. Takve mreže omogućuju ljudima komunikaciju te pristup aplikacijama i informacijama bez upotrebe žice. Bežične mreže možemo podijeliti u četiri kategorije:

- Bežična osobna mreža (engl. *Wireless Personal Area Networks* - WPAN)
- Bežična lokalna mreža (eng. *Wireless Local Area Network* – WLAN)
- Bežična gradska mreža (eng. *Wireless Metropolitan Area Networks* - WMAN)
- Bežična mreža širokog područja (eng. Wireless Wide Area Networks - WWAN) (ComputerNetworkingNotes, 2018)

Tablica 1. prikazuje sve kategorije bežičnih mreža i tipična svojstva za svaku od tehnologija.

Tablica 1: Usporedna bežičnih tehnologija

| Tip | Pokrivenost | Brzina prijenosa | Standard | Primjena |
|---|---|---|---|---|
| WPAN | Na dohvat osobe | Umjerena | Standardni Bluetooth, IEEE 802.15 i zamjena IrDa kabela za periferne uređaje | Zamjena kabela za periferne uređaje |
| WLAN | Unutra zgrade ili kampusa | Visoka | IEEE 802.11, Wi-Fi i HiperLAN | Mobilno proširenje žičanih mreža |
| WMAN | Unutar grada | Visoka | Privatni, IEEE 802.16, i WIMAX | Fiksna bežična veza između domova i tvrtki |

| | | | | |
|---|---|---|---|---|
| WWAN | Širom svijeta | Niska | CDPD i Mobilni 2G, 2.5G, i 3G | Mobilni pristup internetu |

Izvor: (ComputerNetworkingNotes, 2018)

Bežične mreže možemo podijeliti prema dometu u dvije kategorije:

- Bežična tehnologija kratkog dometa (eng. *Short-range wireless technology* - SRWT)
- Bežična tehnologija velikog dometa (eng. *Long-range wireless technology* - LRWT)

U najpopularnije bežične tehnologije kratkog dometa spadaju: NFC, RFID, WiFi, ZigBee, Z-Wave.

WiFi je mrežna tehnologija koja omogućuje računalima i drugim uređajima komunikaciju putem bežičnog signala (Somayya Madakam et al., 2015). Među prvim bežičnim proizvodima ubraja se WaveLan koji nudi brzine od 1Mbps do 2Mbps (Somayya Madakam et al., 2015). Danas se WiFi tehnologija može pronaći na većini električnih uređaja te je implementirana u gotovo sve veće gradove. Najpopularnija topologija koja se koristi u Wi-Fi-ju je topologija zvijezde u kojoj čvorovi mogu međusobno komunicirati samo putem središnjeg čvorišta (Afaneh, 2020).

Bluetooth je vrlo jeftina bežična tehnologija vrlo kratkog dometa. Uglavnom se koristi za spajanje na računalo, pametni telefon, printere, slušalice i razne druge uređaje. Može prenositi različite vrste podataka, kao što su tekst, slika, zvuk i video. Među novim tehnologijama ubraja se Bluetooth niske potrošnje (*Bluetooth low energy - BLE*) koji nudi 10x brži prijenos podataka i vrlo malu potrošnju energije za cijenu dometa.

ZigBee je tehnologija izrađena na temelju IEEE 802.15.4 standarda. Neke od karakteristika ZigBee-a su niska cijena, mala brzina prijenosa, kratak domet, skalabilnost, pouzdanost i fleksibilnost (Farooq et al., 2015). Često se može pronaći u sustavima poput kućne automatizacije, digitalne poljoprivrede i medicinskim nadzoru.

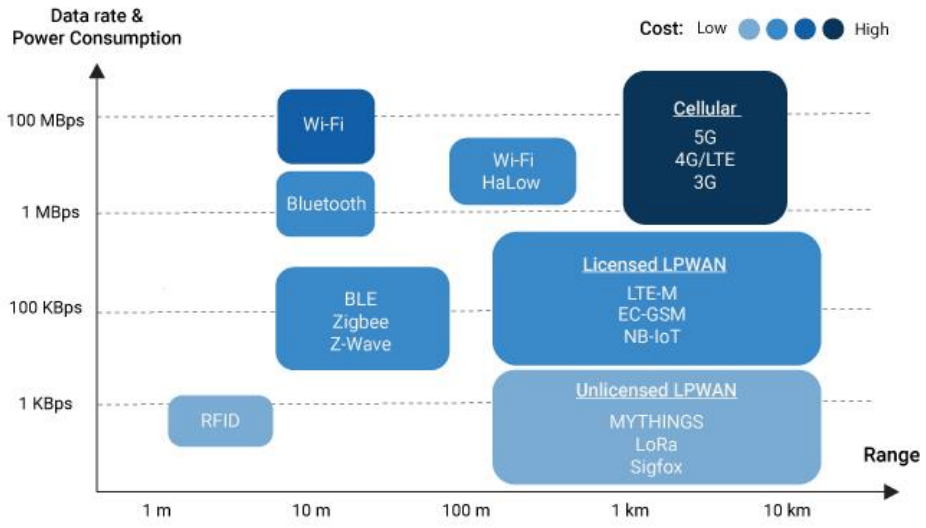
Z-Wave je inicijalno dizajniran kao protokol za upravljanje rasvjetnih sustava, a evoluirao je u protokol kućne automatizacije kojim upravlja Z-Wave savez[1] (Afaneh, 2020). Unutar Europe radi na frekvenciji od 868 MHz, a u SAD-u radi u opsegu između 908 i 915 MHz-a (Afaneh, 2020).

[1] https://z-wavealliance.org/

Širokopojasna mreža niske potrošnje (eng. *Low-power wide-area networking* - LPWAN) i mobilna širokopojasna mreža niske potrošnje (*Cellular Low-Power Wide Area Network* - CLPWAN) su bežične tehnologije s ciljem ostvarivanja velikog dometa pri maloj brzini prijenosa podataka (BehrTech, bez dat.). LPWAN uređaji se mogu povezati sa bilo kakvom vrstom senzora te imaju vrlo malu potrošnju energije. Jedan od glavnih nedostatka je što mogu slati male količine podataka po malim brzinama, pa su prikladniji za slučajeve koji ne zahtijevaju veliku propusnost i nisu vremenski osjetljivi (BehrTech, bez dat.). Na slici 2 nalazi se graf najpopularnijih tehnologija s prikazom po širini dometa, brzini prijenosa i potrošnji energije.

Dugoročna evolucija strojeva (eng. *Long Term Evolution of Mashines* – LTE-M) i uski pojas Interneta stvari (eng. *NarrowBand IoT* – NB-IoT) su tehnologije bazirane na temelju CLPWAN-a (Afaneh, 2020). NB-IoT je idealan za aplikacije s malom potrošnjom energije te malom propusnošću, dok LTE-M nudi veću brzinu prijenosa podataka čiju primjenu možemo pronaći u aplikacijama realnog vremena (eng. *Real-time Applications*) (Afaneh, 2020).

Dalekometna mreža širokog područja (eng. *Long Range Local Area Network* - LoRaWAN) i Sigfox su CLPWAN bazirane tehnologije. LoRaWAN omogućuje dugoročnu komunikaciju između uređaja, a istovremeno održava nisku potrošnju energije (Afaneh, 2020). Sigfox je LPWAN tehnologija koja djeluje kao mrežni operater za tu tehnologiju (Afaneh, 2020). Navedene tehnologije glavnu primjenu nalaze u sustavima poput pametnih gradova, pametnih mjerača komunalnih usluga i pametnih uređaja za parkiranje (Afaneh, 2020).




Slika 2: Najpopularnije IoT tehnologije (BehrTech, bez dat.)

Slika 3 prikazuje usporedbu svih gore opisanih tehnologija u odnosu na domet, propusnost, potrošnju energije, tekući trošak, trošak po jedinici modela, topologiji i količini prodanih primjeraka u 2019 godini.

| Atributi | Bluetooth niske potrošnje (BLE) | Wi-Fi | Z-Wave | IEE 802.15.4 | LTE-M | NB-IoT | Sigfox | LoRaWAN |
|---|---|---|---|---|---|---|---|---|
| Domet | 10 m - 1.5 km | 10m - 100 m | 30 m - 50 m | 30 m - 100 m | 1 km - 10 km | 1 km - 10 km | 3 km - 50 km | 2 km - 20 km |
| Propusnost | 125 kbps - 2 Mbps | 54 Mbps - 1.3 Gbps | 10 kbps - 100 kbps | 20 kbps - 250 kbps | Do 1 Mbps | Do 200 kbps | Do 100 bps | 10 kbps - 50 kbps |
| Potrošnja energije | Niska | Srednja | Niska | Niska | Srednja | Niska | Niska | Niska |
| Tekući trošak | Jednom | Jednom | Jednom | Jednom | Ponavljajući | Ponavljajući | Ponavljajući | Jednom |
| Trošak po modelu | Ispod 5$ | Ispod 10$ | Ispod 10$ | 8$ - 15$ | 8$ - 20$ | 8$ - 20$ | Ispod 5$ | 8$ - 15$ |
| Topologija | P2P, zvijezda isprepleteno, difuzijska | Ispod 5$ | Isprepletena | Isprepletena | Zvijezda | Zvijezda | Zvijezda | Zvijezda |
| Broj pošiljaka u 2019. | ≈ 3500 | ≈ 3200 | ≈ 120 | ≈ 420 | ≈ 7 | ≈ 16 | ≈ 10 | ≈ 45 |

Slika 3: Usporedba IoT tehnologija (ComputerNetworkingNotes, 2018)

Navedene tehnologije su temelj za izgradnju IoT sustava te svaka od tih tehnologija ima svoje prednosti i mane ovisno o području primjene. Zbog toga je važno proučiti koju će geografsku širinu poprimiti ciljani IoT sustav te koju vrstu podatka i kojom brzinom želimo slati podatke unutar sustava.

## 3.2. Budućnost IoT-a

Vizija IoT-a je omogućiti spajanje ljudi i stvari u bilo kojem trenutku, u bilo koje vrijeme i na bilo kojem mjestu koristeći bilo koji uređaj ili mrežu koja mu je trenutno na raspolaganju (Vermesan et al., 2011). Sve što bi pripadalo u neku grupu, zajednicu, proizvode, podatke, servise koristilo bi komunikaciju podržanu od strane pametnih stvari, a sama povezanost bi bila moguća uz vrlo niske troškove te ne bi pripadala nekoj privatnoj tvrtki (Vermesan et al., 2011). Svake godine IoT postaje sve zreliji i jedan od najviše rastućih i iščekivanih IT koncepata u svijetu. IoT možemo također promatrati kao globalnu mrežu koja pruža komunikaciju između čovjeka i stvari, čovjeka i čovjeka, te stvari sa stvari (Somayya Madakam et al., 2015). Pitanje je kako uspostaviti takav jedan sustav koji bi razumio svaki proslijeđeni podatak koristeći siguran komunikacijski kanal i time omogućio potpunu zaštitu korisničkih podataka od štetnih napada. Kako je svakim danom moguće sve lakše prikupiti podatke iz stvarnog svijeta putem različitih senzora i proslijediti ih različitim servisima, što se u puno slučajeva događa na aplikacijskoj razini, možemo zaključiti da će se kroz nekoliko godina povećati korištenje kućnih uređaja povezanih na internet.

Svaki IoT upravljani sustav bio bi u mogućnosti komunicirati s cijelom mrežom i time stvoriti jednu povezanu cjelinu u kojem se svaka informacija međusobno dijeli. S takvom vizijom Internet stvari bi autonomno mogle upravljati svojim procesima i optimizirati svoj proces obrade i dijeljenja podataka (Vermesan et al., 2011). Takav sustav bi imao mogućnost prilagoditi se novom okruženju i uspješno uspostaviti vezu sa novim stvarima koje bi bile povezane u zajedničku mrežu (Vermesan et al., 2011).

## 3.3. Povezanost IOT-a i mobilnih aplikacija

Internet stvari (IoT) i mobilne aplikacije idu ruku uz ruku. Uz pomoć IoT-a ljudima je na drugačiji način predstavljen rad i komunikacija sa strojevima i ostalom elektronikom. Mobilna aplikacija je najčešći odabir za posrednika koji omogućuje takvu komunikaciju.

Na samom početku razvoja mobilnih aplikacija za IoT, internet nije bio glavno sredstvo za prijenos podataka (Ratner, 2020). Često se komunikacija odvijala preko Bluetootha, a kasnije razvojem tehnologije je uključen i internet kao jedno od mogućnosti povezivanja uređaja sa mobilnim aplikacijom. Bluetooth je svojom pojavom ostvario određenu kompleksnost u mobilnom programiranju (Ratner, 2020). Justin Ratner u svojem članku *4 Ways IoT is Changing Mobile App Development* piše da je osnovna specifikacija i komunikacija Bluetootha opisana preko 3000 stranica, što je programerima stvorilo dodatnu muku u programiranju mobilnih aplikacija. Uporabom određenih biblioteka za rukovanje komunikacije preko Bluetooth-a programerima se olakšao posao, ali su one često bile pune grešaka i nepotpune (Ratner, 2020). Također, važno je naglasiti sigurnost korisničkih podataka preko takve komunikacije. Česta praksa je izbjegavati slanje važnih podatka preko Bluetootha, ali kada je to jedini način slanja postavlja se pitanje kako zaštiti te podatke i na koji način. Neki od načina da se korisniku da detaljna korisnička dokumentacija o načinu spajanja preko Bluetooth-a je enkripcija podataka i autentifikacija korisnika te uspješno prekidanje komunikacije preko Bluetooth-a, kako bi se smanjio utrošak baterije na uređaju (Ratner, 2020). Na sve to i još mnogo toga programerski tim mora obratiti pažnju kada je u postupku dizajniranja programskog rješenje za IoT aplikaciju.

Kada je riječ o dizajniranju mobilnih aplikacija pokretanih uz pomoć IoT-a, komunikacija korisnika s aplikacijom mora imati svoj uzrok i posljedicu. Glavni fokus više nije sama aplikacija kao u standardom razvoju mobilne aplikacije, već se na istoj razini mora promatrati korisnik, sklopovlje i programska podrška. Mobilna aplikacija bi trebala tijekom svojeg životnog ciklusa ponuditi korisniku određenu pomoć u postizanju svojih ciljeva ili biti u mogućnosti automatski ponavljati određene zadatke i time korisniku dati više vremena za bitnije stvari.

Većina današnjih aplikacija je *pametna,* ali nije u stanju komunicirati s bilo kojom drugom aplikacijom ili stvari. Većina ih je bazirana na interakciji tipa čovjek-aplikacija. Dodavanjem funkcionalnosti u aplikaciju koja će automatizirati neku ljudsku radnju stvara korisniku određenu vrijednost i omogućuje mu fokusiranje na bitnije stvari. Za primjer možemo uzeti Google-ovu IoT aplikaciju Android Auto. Omogućuje korisniku da vidi uživo promet na cesti, prikazuje okruženje vozila i nudi korisniku bitne smjernice tijekom vožnje u prometu. Također omogućuje korisniku kreiranje odlaznih poziva i sviranje glazbe s mobilnog uređaja kao i mnogo drugih stvari. Time takvu aplikaciju možemo svrstati u Internet stvari (IoT) jer olakšava korisniku vožnju i pruža mu važne informacije, a istovremeno u pozadini komunicira sa ostalim

IoT uređajima. IoT aplikacije osim u prometu mogu biti vrlo korisne i u ostalim sustavima. Tako u zdravstvu uz pomoć raznih senzora možemo pratiti otkucaje srca, tlak, temperaturu i razinu šećera pacijenta unutar i izvan bolnice. Svi prikupljeni podaci se šalju na mobilni uređaj liječniku kao i ostalim članovima obitelji kako bi se u pravo vrijeme moglo pomoći pacijentu u slučaju hitne pomoći. U agrokulturi se može pratiti vlažnost tla, temperatura i količina svjetla dostupna biljkama. Na temelju prikupljenih podataka u slučaju suhog tla automatski se uključuje pumpa za protok vode ili se zbog male količine svijetla upali žarulja. Sva statistika mogla bi se pratiti preko mobilne aplikacije. Sustav upravljanja zvan pametna kuća omogućuje svojim korisnicima upravljanje raznih stvari uz pomoć aplikacije na pametnom uređaju. Neke od funkcionalnosti su: podešavanje termostata, kontrola rasvjete, zaključavanje brave, praćenje preko nadzorne kamere i mnogo drugog. IoT aplikacije ne samo da poboljšavaju našu udobnost, već nam daju i veću kontrolu za pojednostavljivanje rutinskih i osobnih zadataka (Muntjir et al., 2017).


Slika 4: Android Auto aplikacija (Android, bez dat.)

# 4. Dizajn i prototipiranje integracije Interneta stvari i mobilnih aplikacija

Razviti moderan i cjelovit IoT sustav nije jednostavan postupak. Prvi korak je imati određenu ideju, viziju i cilj koja će pojedinca ili tvrtku usmjeriti prema pravom putu. Sljedeći korak je osmisliti jednostavnu skicu koja će voditi prema prvom prototipu. Svi postupci u dizajnu i prototipiranju IoT aplikacije objašnjeni su u sljedeća dva poglavlja.

## 4.1. Dizajn IoT mobilnih aplikacija

Kako bi kreirali sofisticirano IoT rješenje potrebno je dizajnirati aplikaciju koja bi savršeno funkcionirala sa svim povezanim uređajima. U pozadini IoT sustava možemo imati vrlo kvalitetan IoT uređaj, ali loš dizajn korisničkog sučelja (eng. *User Interface* – UI) može značajno utjecati na prezentaciju cjelokupnog proizvoda i time uništiti sav uloženi trud i novac razvojnog tima.

IoT sustavi se razvijaju vrlo brzo. To može reflektirati na promjenu fizičkog dizajna uređaja, razvoju novih funkcionalnosti i mogućnosti uređaja te kreiranju potpuno novih uređaja i objekata. Takve promjene se odražavaju na mobilnu aplikaciju i zahtijevaju brzo prilagođavanje dizajna aplikacije. Iako su ljudi naviknuti raditi u skladu s mobilnom aplikacijom, IoT stalno donosi nova iskustva i mogućnosti kako za ljude tako i za poduzeća (Lazarevich, 2020). Jedan od najvećih izazova u razvoju IoT dizajna je implementirati nova iskustva na korisniku razumljiv način i jednostavan za naučiti (Lazarevich, 2020).

Kada govorimo o velikim IoT sustavima poput sustava za nadzor skladišta, obično podrazumijevamo mnogo korisnika s kontrolom pristupa zasnovanom na ulogama i dozvolama. Prilikom izrade dizajna za IoT aplikacije s više grupa krajnjih korisnika, važno je prilagoditi korisničko iskustvo i funkcionalnost aplikacije njihovim zahtjevima i očekivanjima.

U glavne funkcionalnosti IoT mobilnih aplikacija ubraja se daljinsko upravljanje uređaja i konfiguracija sustava (Lazarevich, 2020). Time se korisniku pruža brz pristup potrebnim funkcijama i olakšava interakcija s IoT sustavom.

Implementacija notifikacija u mobilnu aplikaciju predstavlja jedan od najtežih izazova prilikom dizajna korisničkog sustava (Lazarevich, 2020). Granica između suvišnih notifikacija i onih koje su korisne je vrlo tanka. Ovisno o čimbenicima poput vremena i mjesta, obavijesti nekih sustava trebale bi biti u mogućnosti utišati se ili pojačati (Lazarevich, 2020). Za primjer možemo uzeti aplikaciju koja obavještava liječnika o kritičnom stanju svojeg pacijenta. Takva notifikacija bi uvijek trebala biti maksimalno uočljiva za razliku od sustava notifikacije korisnika

prilikom uspješnog punjenja električnog automobila usred noći koja je suvišna u trenutnom vremenu i situaciji korisnika.

Većina IoT aplikacija u pozadini procesuira različite vrste podataka. Ti podaci se korisniku trebaju prezentirati na jednostavan i koristan način. Za razliku od stolnih računala i laptopa, zaslon pametnog telefona je znatno manji. Zbog toga je potrebno donijeti važne odluke u organizaciji i načinu prikaza prikupljenih podataka. Korištenje filtera, grafova, tablica te brzo premještanje s jednog pogleda na drugi samo su neki od načina kreiranja preglednog prikaza informacija. Ovisno o specifičnostima sustava, trebamo osigurati da aplikacija prikazuje točno one podatke koji su potrebni korisniku i ne zauzimaju bespotrebno prostor (Lazarevich, 2020).

Skalabilnost je također jedno od važnih svojstva IoT aplikacije. Važno je razviti pametno sučelje koje se može prilagoditi sve većem broju uređaja i korisniku uključiti određene funkcionalnosti ovisno o kompletu uređaja koji su trenutno implementirani u postojećem IoT sustavu (Lazarevich, 2020).

## 4.2. Prototipiranje IoT mobilnih aplikacija

Prototipiranje IoT sustava nije jednostavan postupak. Zahtjeva razumijevanje uređaja i tehnologije koji se koriste u fazi prototipiranja. Jedan od glavnih ciljeva u toj fazi je ostvariti uspješnu komunikaciju između uređaja, prikupiti podatke pomoću senzora i prezentirati informacije pomoću korisničkog sučelja na ciljanom uređaju (najčešće pametnom telefonu). Prototipiranje mobilne aplikacije i prototipiranje IoT mobilne aplikacije nije sasvim isto. Prototip mobilne aplikacije je u većini slučaja sličan krajnjem proizvodu. Naprotiv, IoT proizvodi su puno kompliciraniji. Razlog tome je što sklopovlje koji se koristi tijekom izrade prototipa značajno može razlikovati od onog koji se koristiti za masovno stvaranje IoT proizvoda (van Diessen). Programska podrška također postaje sve zahtjevnija kako proizvod doseže završnu fazu.

Prilikom izrade prototipa odabir komunikacijske tehnologije nije presudan ako se radi o malom IoT sustavu. Glavna svrha faze prototipiranja je provjera valjanosti poslovanja putem ciljnih korisnika, a komunikacijska tehnologija se kasnije može lako prilagoditi na temelju traženih zahtjeva ispitanika (Dunkels, 2019). Ako se radi o velikom području koje će IoT aplikacija pokriti, važno je u startu odrediti način povezivanja jer bi izmjena takve tehnologije mogla financijski značajno oštetiti proizvođača.

Za uspješnu nadogradnju prototipa, potrebno je konstantno preispitivati postojeće tehnologije i funkcionalnost sustava. Kako se mobilni uređaji fizički i softverski razlikuju jedan od drugog i svake godine predstavljaju se novi uređaji sa različitim veličinama i oblikom ekrana, potrebno je provesti testiranja na što većem broju uređaja. Također, korisno je napraviti test nad raznom dobnom skupinom ljudi u kojem će biti zadano nekoliko scenarija koji polaznici

moraju riješiti. To nam omogućuje da, kao vlasniku proizvoda, proučimo što korisnici smatraju važnim i time nam skrenuti pažnju na stvari koje su u početku izgledale nebitne.

Odličan izbor za prototipiranje IoT solucija je korištenje prototipskih pločica. Takve pločice imaju sve potrebne fizičke komponente za jednostavno kreiranje IoT solucija. Danas postoji mnogo različitih proizvođača, a među najpoznatijim je tvrtka Arduino[2] koja ima veliki izbor prototipskih pločica za bilo koju namjenu. Također imaju razvijenu programsku podršku s velikom zajednicom korisnika koja konstantno nadopunjuje razvojno okruženje raznim bibliotekama za komunikaciju s različitim senzorima i aktuatorima.

Potreban je veliki napor da se prototip poboljša do te mjere da bi mogao poslužiti kao proizvod za masovnu proizvodnju (van Diessen). Ako je pojedinac ili tvrtka ograničena s resursima potrebno je donijeti pametne odluke na samom početku kako bi se izbjegli budući problemi o odabiru loše komunikacijske tehnologije ili lošem odabiru senzora. Rob van Diessen je u svojem članku *IoT Prototyping: Everything you Need to Know* opisao kreiranje prvog prototipa jednako teškim kao trčanje maratona po prvi puta. Drugim riječima prototipiranje IoT bazirane aplikacije zahtjeva puno vještine, znanja i dobrog planiranja unaprijed.

[2] https://www.arduino.cc/

# 5. Odabir Bluetooth-a kao komunikacijskog sredstva

Kako je u ovom radu pokrivena IoT aplikacija bazirana komunikacijom preko Bluetooth-a, potrebno je objasniti dizajniranje jedne takve aplikacije s gledišta programera. Način na koji se pristupa razvoju Bluetooth aplikacije može se poistovjetiti s razvojem web aplikacije (Huang & Rudolph, 2005). Razlog tome je što Bluetooth pokriva puno pojmova koje susrećemo u svijetu interneta. Tako je npr. u oba slučaja potrebno je odabrati uređaj s kojim se želi komunicirati i način na koji se želi komunicirati. Također, potrebno je uspostaviti vezu s drugim uređajem, prihvatiti dolaznu vezu te na kraju međusobno razmjenjivati podatke. Iako takav model nije točan za svaki pristup web programiranju, dobar je temelj za određivanje zajedničkih točaka između Bluetooth i web tehnologija (Huang & Rudolph, 2005).

Bluetooth je puno detaljniji te slično kao i kod protokola za prijenos zvuka (eng. *Voice-over-IP* - VOIP) pokriva sve što se fizički izvodi u komunikaciji između dva uređaja, kako se uspostavi signal, na koji način se prekine signal i kako se prenosi zvuk između uređaja (Huang & Rudolph, 2005). Programer ne treba poznavati svaki sloj Bluetooth-a, niti treba poznavati sve tehnologije, već samo jedan mali dio, a to je kako povezati jedan Bluetooth uređaj sa drugim Bluetooth uređajem. Također mora poznavati kako će prenositi podatke s jednog uređaja na drugi i obrnuto.

Postavlja se pitanje kako doznati adresu drugog Bluetooth uređaja s kojim se želi uspostaviti veza? Kako u web tehnologiji postoji MAC adresa, tako svaki Bluetooth uređaj sadrži svoju globalnu jedinstvenu 48-bitnu adresu koja se često zove *Bluetooth adresa* ili *adresa uređaja* (Huang & Rudolph, 2005). Određuje se prilikom proizvodnje uređaja te jedna adresa ostaje neizmijenjena sve do završetka životnog ciklusa uređaja (Huang & Rudolph, 2005). Prilikom programiranja definira se kao osnovna adresa za komuniciranje uređaja s drugim uređajima te se koristi na svim slojevima Bluetooth komunikacijskog procesa (Huang & Rudolph, 2005). Kada gledamo sa web programske strane serverska strana bi imala dodijeljenu IP (eng. *Internet Protocol*) adresu, koja je zamijenjena s nekom adresom domenskog sustava imena (eng. *Domain Name Server address* – DNS address). Također u svijetu Bluetooth-a često se adresa uređaja zamjenjuje s nekim jednostavnim nazivom, poput naziva samog uređaja. Kasnije u primjerima bit će prikazan način imenovanja Bluetooth uređaja te sam postupak spajanja sa klijentom. Važno je napomenuti da se sama adresa Bluetooth uređaja koristi za komunikaciju, a dodijeljeno ime služi samo za lakše korištenje i pamćenje svih uređaja. Tako na primjer mobilni uređaji koji imaju Bluetooth omogućuju korisniku preimenovanje samog uređaja za lakše raspoznavanje kada se nalaze u okolini s više istih ili sličnih uređaja.

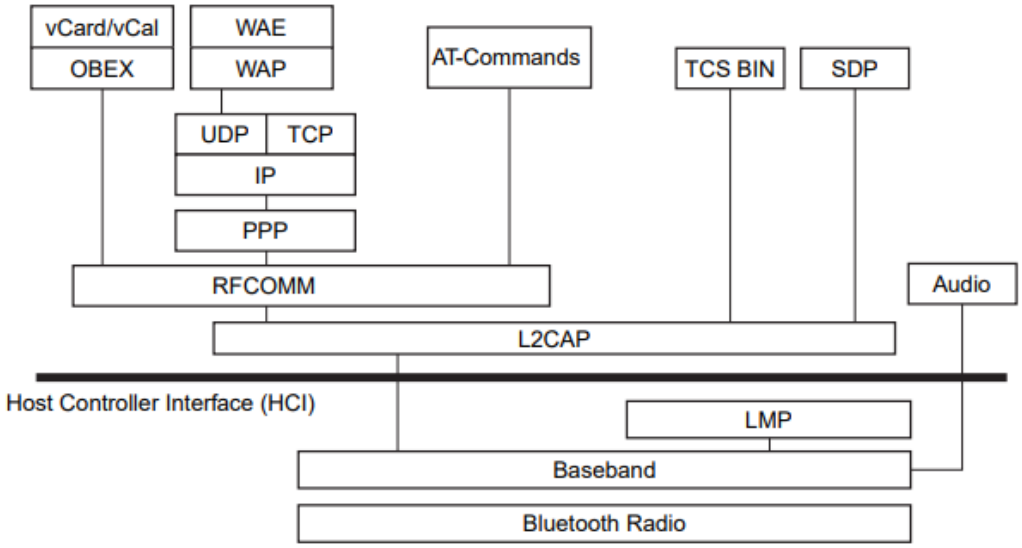
## 5.1. Bluetooth protokoli

Kako bi razumjeli način na koji funkcionira Bluetooth, potrebno je prvo objasniti koje sve protokole koristi i koji protokoli služe za koju namjenu. Organizacija Bluetooth SIG[3] definira glavne protokole te je glavni zaštitni znak iza Bluetooth tehnologije.

Jedna od glavnih funkcija Bluetooth-a je *Bluetooth protocol stack*. Zadaća mu je da definira i pruža različite vrste slojeva i funkcionalnosti Bluetootha. Bluetooth Framework se može implementirati preko različitih slojeva protokola, ali svaki od tih protokola koristi isti fizički sloj (Soft Service Company, bez dat.). Slika 5 prikazuje sve slojeve Bluetooth protokola i veze između svakog od tih protokola.

Bluetooth protokol za komunikaciju preko radijskih frekvencija (eng. *Radio frequency communication* - RFCOMM) je jednostavan set transportnih protokola koji se nalazi iznad protokola za kontrolu i prilagodbu logičke veze (eng. *Logical link control and adaptation protocol* - L2CAP), a služi za simuliranje RS-232 serijskog porta (Soft Service Company, bez dat.). Možemo ga shvatiti kao TCP (eng. *Transmission Control Protocol*) protokol, jer pruža približno iste usluge i pouzdanost kao i TCP. Jedna od najvećih razlika između RFCOMM i TCP je što TCP pruža 65535 otvorenih portova, dok RFCOMM pruža samo 30 (Huang & Rudolph, 2005). Često se RFCOMM naziva i SPP što stoji za profil serijskog porta (eng. *Serial Port Profile)*. SPP je jedan od najčešće korištenih Bluetooth profila čime ga ne možemo potpuno zamijeniti sa RFCOMM protokolom jer RFCOMM koriste i drugi profili poput OPP (eng. *Object Push Profile*), FTP (eng. *File Transfer Profile*) i mnogo drugih Bluetooth profila (Soft Service Company, bez dat.). Bluetooth Framework prirodno podržava RFCOMM protokol u klijentskom i serverskom načinu rada (Soft Service Company, bez dat.). Tako možemo raditi na uređaju koji se želi spojiti s ostalim Bluetooth uređajima (klijentska strana), kao i prihvaćati konekciju s ostalim Bluetooth uređajima (serverska strana).

[3] https://www.bluetooth.com/




Slika 5: Bluetooth Protocol Stack (Pandey & Bagwe, 2014)

## 5.2. Bluetooth profili

Kako bi uređaj mogao koristiti Bluetooth tehnologije, mora biti kompatibilan s podskupom Bluetooth profila. (Soft Service Company, bez dat.). Ovisno o podržanim profilima na uređaju, moguće je koristiti one tehnologije koje su vezane za određeni Bluetooth profil (Soft Service Company, bez dat.). Svaki Bluetooth profil je izgrađen za sebi vezani Bluetooth protokol. Android podržava puno Bluetooth profila, a jedan od njih je SPP. ESP32 mikrokontroler također podržava SPP Bluetooth profil pa oba uređaja mogu vrlo lako komunicirati i međusobno slati podatke.

Postavlja se pitanje na koji način programeri mogu iskoristiti određeni Bluetooth profil te kako uopće znaju koji profil koriste. Tu dolazi UUID (eng. *Universally Unique ID*) koji na jedinstven način definira svaki Bluetooth profil. Programer može sam napraviti svoj 128-bitni UUID u tijeku dizajniranja aplikacije (Huang & Rudolph, 2005). Kada se pokrene program, kreirani ID se registrira na SDP (eng. *Service Discovery Protocol*) serveru za određeni uređaj. Klijentska aplikacija koja pokušava pronaći određeni proces odnosno profil, zatraži od SDP poslužitelja da provjeri UUID uređaja i usporedi ga s traženim UUID-em (Huang & Rudolph, 2005).

Osim prilagođenih UUID, postoje i rezervirani UUID-evi. Slično kako određeni protokoli imaju rezervirane portove tako i Bluetooth profili imaju rezervirane UUID-eve. Najčešće se koriste za identificiranje unaprijed definiranih klasa usluga (Huang & Rudolph, 2005). Obično dolaze kao 16-bitne ili 32-bitne vrijednosti, ali potpuno odgovaraju 128-bitnim UUID-evima (Huang & Rudolph, 2005). Kako bi doznali cijeli 128-bitni UUID potrebno je uzeti bazni UUID (npr. 00000000-0000-1000-8000-00805F9B34FB) koji ćemo koristiti u ovom radu, te nakon

toga zamijeniti krajnji lijevi segment sa 16-bitnom ili 32-bitnom vrijednošću (Huang & Rudolph, 2005).

Aplikacija *Smart Control* će koristiti SPP Bluetooth profil, jedan je od najčešće korištenih Bluetooth profila pokraj bluetooth LE. Jedna od bitnih razlika između ta dva profila je način na koji se ostvaruje povezivanje dva uređaja. Proces se često naziva *pairing mode*.

Bluetooth SPP zahtjeva da se prvo uređaj upari sa operacijskim sustavom ciljanog uređaja. Tako npr. u Android operacijskom sustavu potrebno je prvo u postavkama uređaja stvoriti konekciju. Nakon toga potrebno je povezati se sa samom aplikacijom što će biti prikazano u analizi programskog rješenja u Kotlinu. S druge strane Bluetooth LE (poznat kao bluetooth 4.0) zahtjeva samo povezivanje s aplikacijom. Važno je napomenuti da Bluetooth SPP nije dostupan na IOS uređajima, već je potrebno koristiti Appleov profil zvan *Bluetooth iAP/MFi* (SerialIO, bez dat.). Jedna je od značajki SPP-a je što nudi vrlo brz i stabilan prijenos podataka između dva uređaja, što je i više nego dovoljno za izradu *Smart Control* aplikacije. Bluetooth LE je orijentiran više prema optimizaciji utroška električne energije i brzini prijenosa, čime se žrtvuje daljina prijenosa podataka (SerialIO, bez dat.).

Nakon što je objašnjen Bluetooth protokol i odabran profil, potrebno je odabrati platformu koja će korisniku omogućiti uspostavu komunikacije između uređaja. Nakon toga je potrebno kreirati korisničko sučelje i testirati funkcionalnost IoT sustava.

# 6. Odabir platforme za prototipiranje IoT sustava

Prvi korak u prototipiranju IoT sustava, nakon definiranja ciljeva i svrhe IoT sustava, je odabrati platformu koja će biti temelj za izgradnju osnovnih funkcionalnosti IoT sustava. Odabir dobre platforme može značajno utjecati na implementaciju, održavanje, praćenje i upravljanje IoT uređaja i time osigurati stalni rast i sigurnost poslovanja.

Arduino i Raspberry Pi su dvije najpoznatije platforme za prototipiranje IoT sustava. Raspberry Pi je malo računalo bazirano na Linux operacijskom sustavu. Budući da pokreće cijeli operacijski sustav može pokretati bilo koji program za stolna računala, ali je zbog toga kompliciraniji za korištenje u odnosu na Arduino. Odličan je izbor za složene IoT projekte koji zahtijevaju puno računalne obrade i veliki kapacitet pohrane podataka (Teel, 2017). Ima podršku za programske jezike poput: Python, C/C++, Java, Perl i Ruby.

Arduino sadrži programibilan mikrokontroler i posvećen alat za njegovo programiranje. Cijeli sustav je jednostavan za korištenje, a razlog tome je velika kolekcija biblioteka koja omogućuje upravljanje različitim vrstama senzora. Arduino ima veliku zajednicu koja konstantno nadograđuje platformu novim funkcionalnostima kreirajući biblioteke za upravljanje mikrokontrolera i senzora. Arduino platforma je idealna za robusne, fleksibilne i jeftine IoT solucije bazirane na C i C++ programskom jeziku (Teel, 2017). U ovom radu je korišten upravo Arduino sustav za upravljanje fizičkim dijelom IoT sustava i ostvarivanje komunikacije s mobilnom aplikacijom.

Sljedeći korak u izradi prototipa IoT sustava je odabrati IoT uređaj koji je Arduino kompatibilan i sadrži Bluetooth tehnologiju.

## 6.1. ESP32

ESP32 je mikrokontroler niske potrošnje električne energije s integriranom Wi-Fi i Bluetooth tehnologijom što ga čini savršenom platformom za izradu raznih IoT aplikacija. Izrađen je od strane Espressif[4] tvrtke koja se bazira na izradi AIoT (eng. *Artificial Intelligence of Things*) platformi. Espressif se fokusira na izradi najnovije Wi-Fi i Bluetooth tehnologije, uređaja niske potrošnje te kreiraju modernih IoT i AI (eng. *Artificial Intelligence*) solucija (Espressif Systems, bez dat.). Jedan od njihovih izuma je ESP32 čip, nasljednik popularnog ESP8266 čipa. Broj 32 u nazivu znači da se radi o 32 bitnom čipu koji radi na 240 MHz (Espressif Systems, bez dat). Ima vrlo malu potrošnju, a ukupna memorija za pohranu je 520 KiB. Sadrži podršku za SPI (eng. *Serial Peripheral Interface*), $I^2C$ (eng. *Inter-Integrated Circuit*),

[4] https://www.espressif.com/

$I^2S$ (eng. *Inter-IC Sound*) i UART (*Universal asynchronous receiver-transmitter*) komunikaciju, *Ethernet* sučelje i mnogo drugog. Budući da je za IoT aplikacije vrlo važna i sigurnost, ESP32 nudi IEE 802.11 standardnu sigurnost sa podrškom za WPA/WPA2 (eng. *Wi-Fi Protected Access* / *Wi-Fi Protected Access 2*) i WAPI (eng. *WLAN Authentication and Privacy Infrastructure*) algoritmima, kao i kriptografske protokole poput AES (eng. *Advanced Encryption Standard*), SHA-2 (eng. *Secure Hash Algorithm 2*) i RSA (*Rivest–Shamir–Adleman*) (Espressif Systems, bez dat.). Sa svim nabrojenim mogućnostima ESP32 mikrokontroler je savršen izbor za IoT aplikaciju koja će biti obrađena u ovom završnom radu.

Kako bi lakše iskoristili sve funkcionalnosti ESP32 mikrokontrolera često se koristi određena platforma koja sve funkcionalnosti implementira na jednoj prototipskoj pločici. Ona služi kao mozak cijelog sustava i komunicira sa svim međusobno povezanim elektroničkim komponentama (Teel, 2017). Time je omogućen pomak sa sklopovskog programiranja na niskoj razini na programiranje više razine (C/C++, Python) pomoću standardiziranog serijskog komunikacijskog sučelja (eng. *Universal Serial Bus interface* – USB interface) (Teel, 2017).

Croduino Nova32 je jedna od mnogih ESP32 baziranih prototipskih pločica. Proizvedena je od strane E-radionice[5], tvrtke koja se bavi izradom prototipskih pločica kompatibilnih sa Arduino sustavom. Neke od značajki pločice su direktna komunikacija preko ugrađenog USB konektora, punjač za litij-ionsku bateriju kao dodatna opcija napajanja i easyC konektor koji omogućava jednostavno povezivanje mikrokontrolerskih pločica, senzora i aktuatora bez mogućnosti pogreške prilikom spajanja. Na tržištu se nalaze razne ESP32 bazirane prototipske pločice s različitim ugrađenim funkcionalnostima. Na primjer Wemos Lolin32 prototip pločica osim standardnih ESP32 funkcionalnosti nudi ugrađeni OLED (eng. *Organic Light Emitting Diode*) display, dok ESP32S NodeMCU ima prilagođeni dizajn eksperimentalnoj pločici, ali nema podršku za ugradnju baterije niti ugrađeni display. Croduino Nova32 je idealno rješenje za ovaj završni rad jer sadrži sve osnovne ESP32 funkcionalnosti te nudi autonomni rad na bateriji što će biti vrlo korisno u kasnijim faza prototipiranja kada će se testirati radni vijek sustava. EasyC konektor bi također mogao biti koristan u fazi implementiranja senzora i testiranja novih funkcionalnosti sustava.

[5] https://e-radionica.com/

## 6.2. Android Studio razvojno okruženje

Nakon što je odabrana platforma za upravljanje fizičkog dijela sustava, potrebno je odabrati platformu koja će se pobrinuti za dizajn i implementaciju korisničkog sučelja mobilne aplikacije. U ovom radu je pokrivena android mobilna aplikacija razvijena u Android Studio razvojnom okruženju. Službeno je podržana od strane Google-a, te nudi razvoj aplikacije u programskom jeziku Java i Kotlin. Platforma nudi prikaz vizualnog rasporeda elementa aplikacije, inteligentni uređivač kôda, mogućnost izrade više varijanti aplikacije za različite uređaje unutar jednog projekta, mogućnost korištenja emulatora za testiranje na različitim uređajima, mogućnosti za smanjenje veličine android aplikacije pregledavanjem sadržaja APK datoteke aplikacije i mnogo drugog. Android Studio sadrži posebno aplikacijsko programsko sučelje (eng. *Application programming interface* - API) koje omogućuje uporabu različitih Bluetooth funkcionalnosti. Time je moguće uspostaviti komunikaciju preko Bluetooth-a između mobilne aplikacije i drugog IoT uređaja te omogućiti međusobno prosljeđivati podataka.

Nakon utvrđivanja ciljeva i svrhe IoT sustava, korištene komunikacijske tehnologije, odabira platforme za upravljanje fizičkog dijela IoT sustava i korisničkog sučelja te početnog modela potrebno je kreirati prvu instancu prototipa kako bi IoT projekt mogao napredovati prema novoj fazi.

# 7. *Smart Control* aplikacija

U ovom poglavlju prikazan je način izrade prototipa IoT mobilne aplikacije *Smart Control*. Uvodni dio pokriva ciljeve aplikacije i kratak opisa rada. Zatim je prikazan postupak u izradi dizajna i arhitekture sustava. Nakon toga će biti riječ o prototipiranju i testiranju sustava te na kraju opis programskog rješenja mobilne aplikacije.

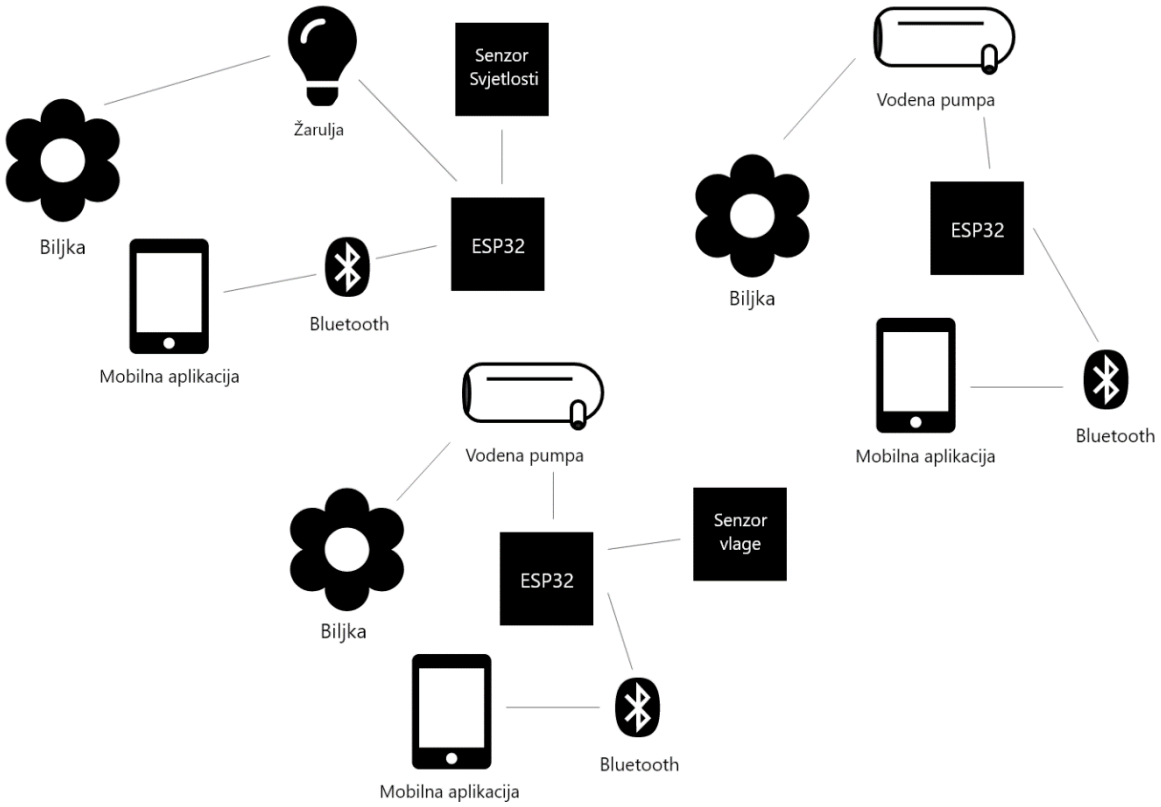
*Smart Control* je mobilna aplikacija koja omogućuje korisniku zalijevanje biljaka preko svojeg mobilnog uređaja. Cilj aplikacije je zamijeniti svakodnevnu brigu ljudi oko biljaka u kućanstvu s jednostavnim klikom preko mobilnog uređaja. Time se korisnik može fokusirati na druge stvari te uštedjeti vrijeme i trud oko procesa zalijevanja biljaka. Misija *Smart Control* aplikacije je uspostaviti pametni sustav u području kućne agrokulture koja će automatizirati određene procese te svojim korisnicima omogućiti potpunu kontrolu i uvid u svoj pametni vrt.

Komunikacija između uređaja odvija se preko Bluetooth-a, a uređaj koji je zadužen za upravljanje malog motora za protok vode je ESP32 mikrokontroler, baziran na Croduino Nova32 prototipskoj pločici. Aplikacija je razvijena za Android sustav i dizajnirana za jednostavno korištenje gdje je većina funkcionalnosti skrivena u pozadini.

## 7.1. Dizajn IoT sustava

Postupak dizajniranja IoT aplikacije nije jednostavan. Za razliku od dizajniranja mobilne aplikacije gdje je u fokusu samo mobilna aplikacija, dizajniranje IoT sustava osim mobilne aplikacije zahtjeva i dizajn fizičkog dijela sustava. U samom početku glavni cilj u dizajnu IoT sustav u ovom radu bio je osmisliti rješenje za jednostavnu brigu oko biljaka u kućanstvu. Nakon prevedenog istraživanja odlučeno je da se sustav treba sastojati od mobilne aplikacije, uređaja s kojim će mobilna aplikacija komunicirati, određenog komunikacijskog kanala i neka vrsta aktuatora ili senzora kojom će se upravljati. Za početni prototip potrebno je dizajnirati sustav s ciljem implementacije funkcionalnosti koja će pridonijeti najveću korist korisniku sustava. Na slici ispod prikazana su tri potencijalna modela sustava. Prvi model predstavlja sustav u kojem mobilna aplikacija prikazuje količinu svjetlosti u prostoriji s biljkom. Podatke sa senzora svjetlosti prikuplja ESP32 mikrokontroler o kojem je bilo riječi u prethodnim poglavljima. Ako je količina svjetlosti premala, uključuje se žarulja i biljka se opskrbljuje potrebnom količinom svjetlosti. Drugi i treći sustav su vrlo slični, ali postoji bitna razlika. Oba sadrže mobilnu aplikaciju, ESP32 mikrokontroler i pumpu za protok vode, ali jedan sadrži i senzor vlage. Time bi se uključivanje i isključivanje pumpe moglo kontrolirati na temelju vlage u zemlji. Problem u takvom pristupu je što senzori vlage imaju kratak vijek trajanja zbog dodira s vodom koju ispušta pumpa. Također oštećivanjem senzora može doći do stvaranja

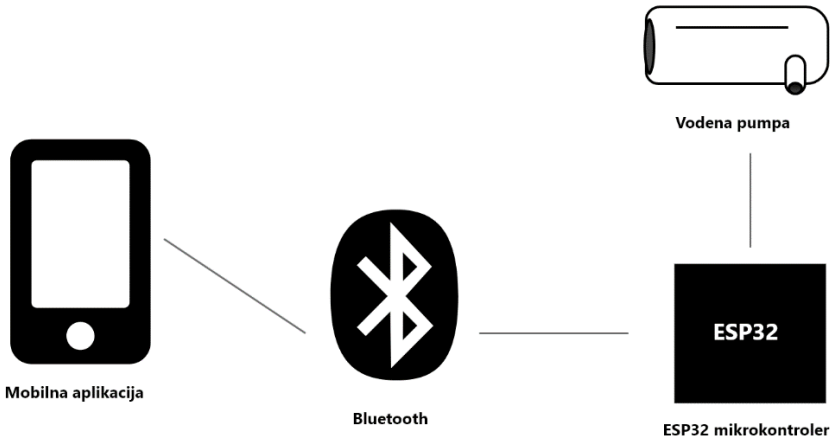
određenih kemikalija i trovanja biljke. Bolji i sigurniji pristup je implementirati sustav u kojem se uz pomoć mobilne aplikacije ručno kontrolira protok vode. Time je korisnik u potpunoj kontroli sustava i siguran je da će se sustav reagirati onako kako on želi. Potrebno je razmisliti koji model će biti implementiran u prvoj fazi testiranja. Budući da je voda glavni faktor uzgoja biljke, a sustav je ciljan kućnoj uporabi u kojoj nema prirodnog izvora vode odlučeno je da će arhitektura sustava implementirati pumpu koja će omogućiti opskrbu biljke vodom. U sljedećem poglavlju je prikazana odabrana arhitektura IoT sustava za ovaj rad.




Slika 6: Potencijalni modeli sustava

## 7.2. Arhitektura IoT sustava

Arhitektura Interneta stvari za ovaj rad je prikazana na slici 6. Sustav se sastoji od korisničke aplikacije, Bluetooth komunikacijskog kanala, malog mikrokontrolera i jedne pumpe. Mobilna aplikacija šalje naredbe ESP32 mikokontroleru preko Bluetooth veze. Mikrokontroler obrađuje naredbe te ovisno o naredbi pali ili gasi pumpu.




Slika 7: Arhitektura sustava

## 7.3. Testiranje sustava

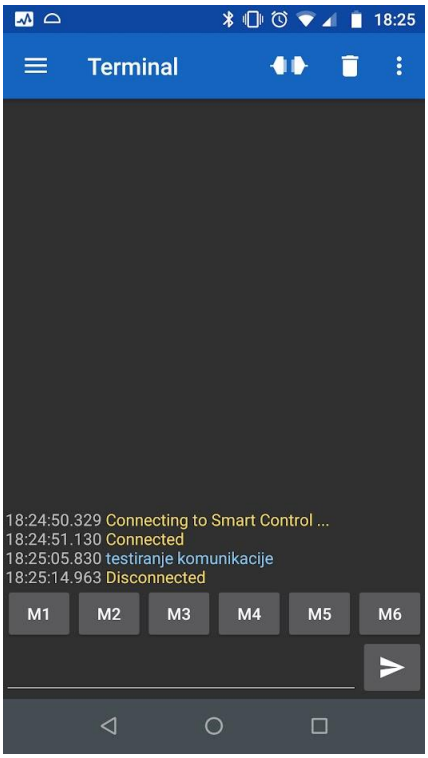
Prije izrade mobilne aplikacije potrebno je testirati rad EPS32 mikrokontrolera. Na Google-ovoj službenoj trgovini aplikacija (eng. *Google Play*) dostupna je aplikacija pod nazivom *Serial Bluetooth Terminal*. Omogućuje testiranje osnovnih Bluetooth funkcionalnosti na mikrokontrolerima koji sadrže klasičnu Bluetooth tehnologiju ili Bluetooth LE (eng. *Low Energy*) tehnologiju preko mobilne aplikacije. Na slici 8. možemo vidjeti neke od testiranih funkcionalnosti. Prilikom otvaranja *Terminal* kartice aplikacija pokušava uspostaviti konekciju s EPS32 uređajem. Porukom *Connected* aplikacija obavještava korisnika da je ostvarena konekcija između uređaja. Unosom bilo kakvog znakovnog niza u predviđeno polje za unos, aplikacija šalje unesene podatke preko Bluetooth-a i ispisuje ih na zaslonu preko serijskog sučelja. Na slici 9. možemo vidjeti ispis poruke na Arduino serijskom zaslonu.

Uključivanjem i isključivanjem led diode provedena je simulacija upravljanja pumpe za navodnjavanje. Navedena simulacija testirana je preko *Serial Bluetooth Terminal* aplikacije unosom određenog znaka u za to predviđeno polje nakon čega se upalila integrirana led dioda na Croduino Nova32 pločici. Provedeno je također testiranje maksimalne udaljenosti između uređaja. Udaljenost do 20 metara unutar i izvan kuće nije predstavljala problem. Udaljenosti veće od 20 metara su uzrokovale sporije slanje podataka te prekid konekcije.

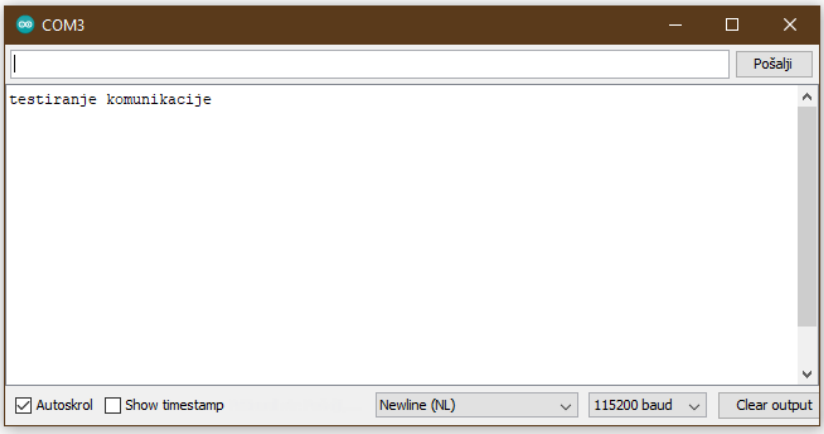
Aplikacija *Serial Bluetooth Terminal* pridonijela je boljem razumijevanju zahtjeva sustava te samom načinu pristupa dizajnu mobilne aplikacije. Potrebno je izbjegavati sustav slanja poruka za kontrolu IoT sustava. Takav način je vrlo nespretan i zahtijeva od korisnika poznavanje ključnih riječi za kontroliranje IoT uređaja. Bolji način je koristiti isključivo tipke za

kontroliranje IoT uređaja čime se sprječava unos krivih naredbi i smanjuje vrijeme potrebno za interakciju s uređajem.

Nakon što je definirana arhitektura i dizajn sustava te provedeni različiti testovi potrebno je odrediti funkcionalne i nefunkcionalne zahtjeve. Opis zahtjeva je prikazan u sljedećem poglavlju.




Slika 8: Aplikacija Serial Bluetooth Terminal

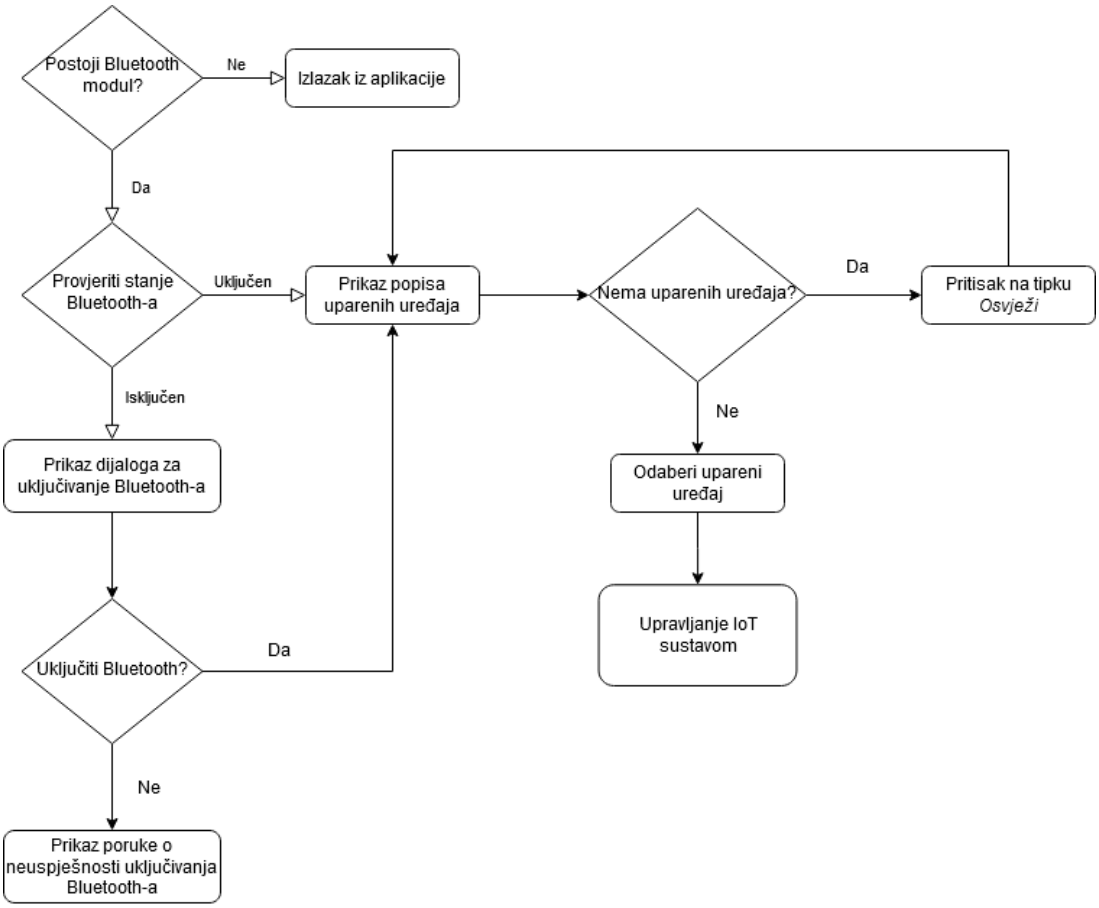



Slika 9: Prikaz uspješne komunikacije između uređaja

## 7.4. Zahtjevi sustava

### 7.4.1. Funkcionalni zahtjevi sustava

Funkcijski zahtjevi aplikacije prikazani su jednostavnim dijagramom toka na slici 9.




Slika 10: Dijagram toka funkcionalnih zahtjeva aplikacije

Pristup funkcionalnostima aplikacije imaju samo korisnici koji na svojim pametnim uređajima posjeduju Bluetooth modul. Nadalje aplikacija će ispravno raditi ako je korisnik uključio Bluetooth i uparo mobilni uređaj sa IoT uređajem (ESP32 mikrokontroler). Ako je korisnik učinio sve prethodne korake, sljedeći zahtjev aplikacije je odabir IoT uređaja iz popisa uparenih Bluetooth uređaja. Nakon toga korisnik može koristiti sve funkcionalnosti koje su mu trenutno dostupne unutar aplikacije.

### 7.4.2. Nefunkcionalni zahtjevi sustava

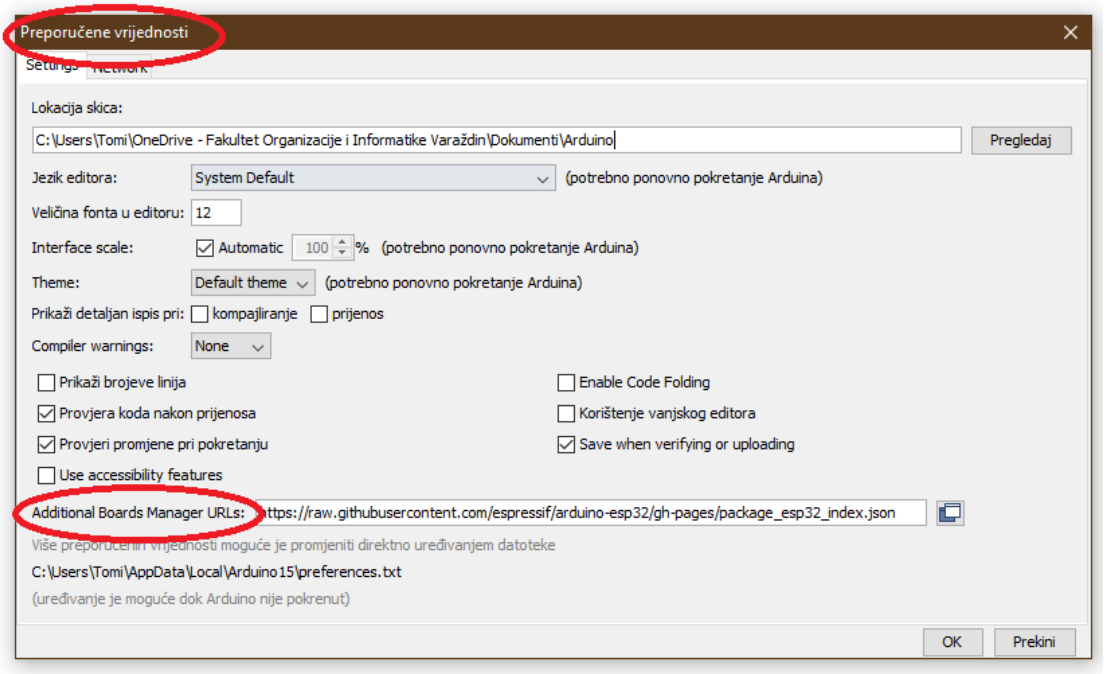
Mobilnu aplikaciju *Smart Control* moguće je koristiti ako su zadovoljeni sljedeći zahtjevi:

- posjedovanje Android mobilnog uređaja
- instalirana Android verzija 5.0 ili više
- posjedovanje sukladnog IoT uređaja
- IoT uređaj priključen na izvor električne energije
- ispravno povezan IoT uređaj sa vodenom pumpom
- maksimalna udaljenost između uređaja je 20 metara

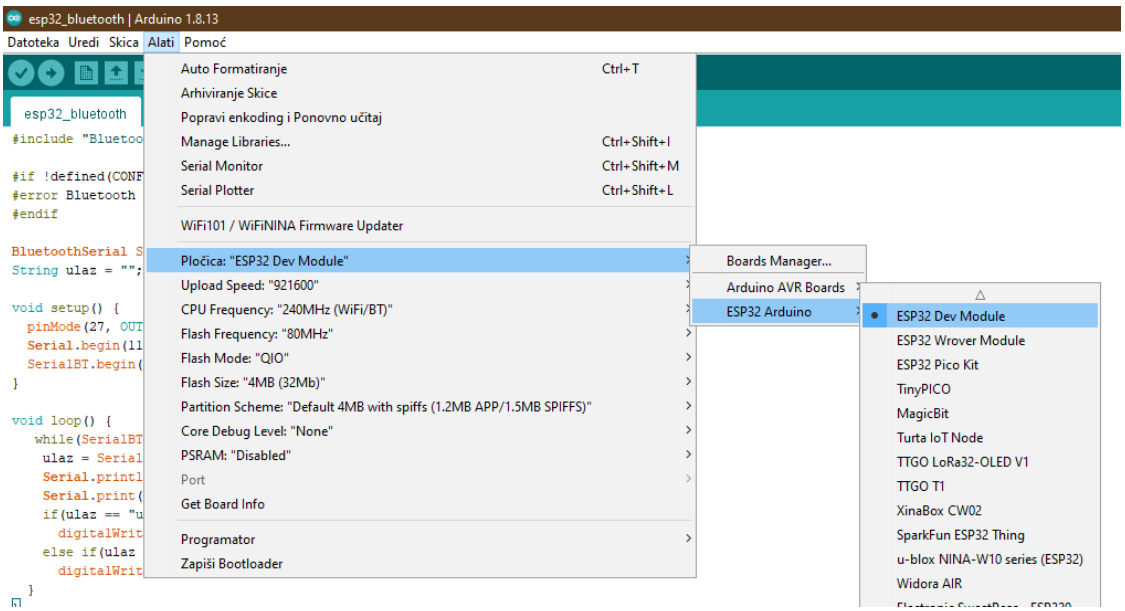
Nakon što su definirani funkcionalni i nefunkcionalni zahtjevi sustava možemo započeti s programiranjem EPS32 mikrokontrolera. Postavljanje ESP32 mikrokontrolera po prvi puta u Arduino razvojnom okruženju i način programiranja bilo kojeg drugog Arduino kompatibilnog mikrokontrolera s mogućnošću povezivanja preko Bluetootha je opisano u sljedećih nekoliko poglavlja.

## 7.5. Arduino IDE i početne postavke mikrokontrolera

Arduino IDE je integrirano razvojno okruženje koje se koristi za pisanje i prijenos programa za mikrokontrolerske pločice koje su Arduino kompatibilne. Program je *open-source*, što znači da autor svojim korisnicima daje pravo pregleda, promjene i distribucije svojeg programa u bilo koju svrhu. Program je dostupan na Arduino.cc službenoj web stranici i moguće ga je skinuti za Windows, Mac i Linux operativni sustav. Budući da je Croduino Nova32 Arduino kompatibilna vrlo je lako postaviti radno okruženje za rad s ESP32 mikrokontrolerom. Budući da sama pločica nije službeno podržana potrebno ju je ručno uključiti u Arduino IDE. Na kartici Datoteka (eng. *File*) potrebno je otvoriti preporučene vrijednosti (eng. *Preferences*) te u polje *Additional Boards Manager URLs* zalijepiti najnoviju verziju json datoteke sa službene Espressif Github stranice. Nakon toga pod karticom Alati->Pločica (eng. *Tools->Boards*) odaberemo *ESP32 Dev Module* i Croduino Nova32 je spremna za korištenje.




Slika 11: Dodavanje ESP32 mikrokontrolera u Arduino

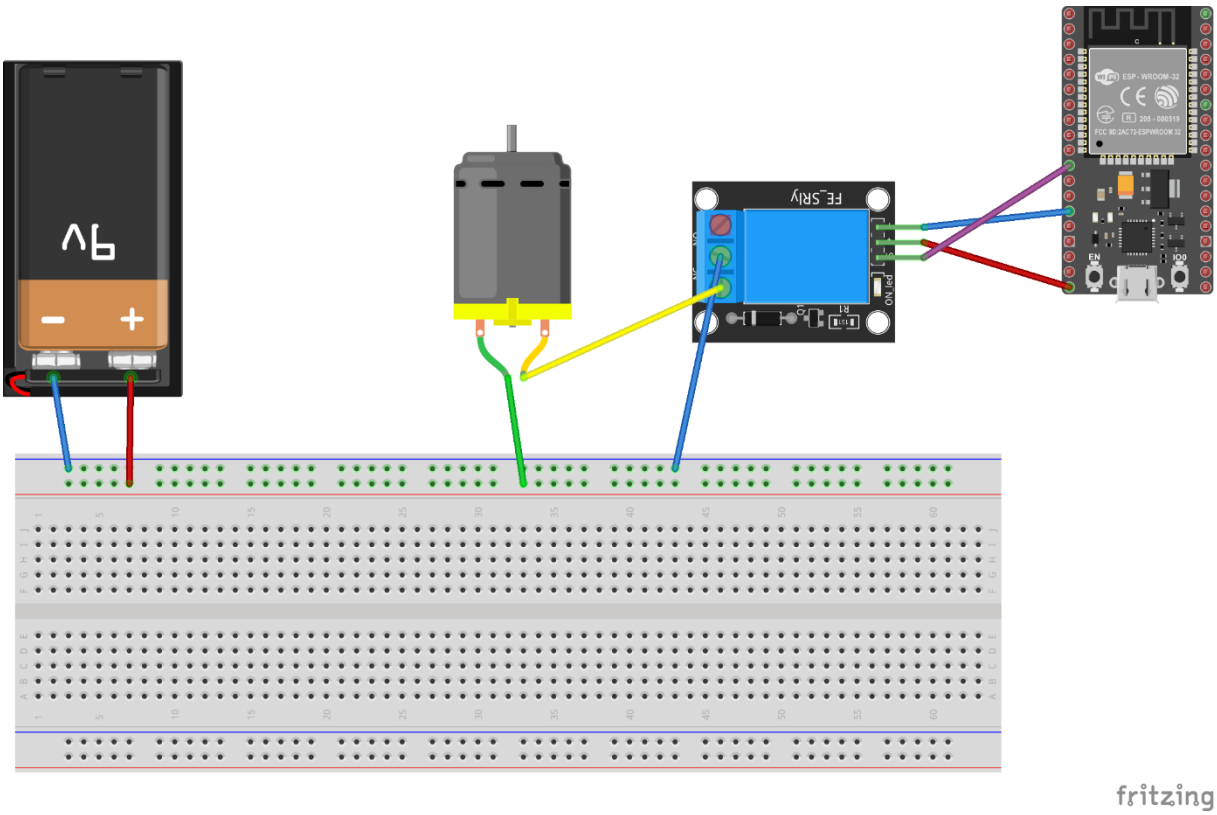



Slika 12: Odabir kompatibilnog ESP32 modela

## 7.6. Način povezivanja ESP32 sa ostalim komponentama

Kako bi uz pomoć *Smart Control* aplikacije mogli uspješno upravljati motorom uz pomoć kojeg ćemo zalijevati biljke, potrebno je prvo ispravno povezati motor s mikrokontrolerom. Na slici 12. možemo vidjeti sve fizičke komponente koje smo koristili u projektu te na koji način smo povezali sve te komponente. Bitno je napomenuti da se na slici 13. ne nalazi Croduino Nova 32, već jedna druga vrsta pločice koja također sadrži ESP32 mikrokontroler. Razlog tome je što program *Fritzing*, koji se koristi za dokumentaciju prototipa, trenutno ne podržava Croduino Nova 32 model, ali broj pinova i njihov raspored je isti kao i u Croduino nova 32 pločici pa ga time možemo lako nadomjestiti prikazanom pločicom.

Kako ESP32 može raditi samo na 3.3V i 5V, a motor radi na većem naponu, potrebno je implementirati relej. Relej je elektromagnetski prekidač pomoću kojeg možemo s manjim naponom kontrolirati protok struje većeg napona. Time smo stvorili dva nezavisna strujna kruga, jedan u kojem ESP32 komunicira sa relejom te drugi u kojem je relej povezan sa baterijom i motorom. Jednostavnim signalom s Croduino Nova 32 pločice možemo relejom upravljati poput prekidača gdje je u jednom stanju otvoren protok struje, a u drugom je protok isključen. U nastavku će biti opisano kako programski možemo upravljati s opisanim komponentama te na koji način uspostavljamo Bluetooth komunikaciju s ESP32 mikrokontrolerom u Arduino razvojnom okruženju.




Slika 13: Prototip ESP32 projekta

## 7.7. Arduino kôd

Programski kôd napisan u Arduino IDE naziva se skica. Za pretvorbu skice u računalu razumljiv skup naredbi koristi se C/C++ kompajler pa je sintaksa vrlo slična programima C i C++.

U svakoj Arduino skici postoje dvije standardne funkcije, a to su `setup()` i `loop()`. Obje funkcije su tipa *void* što znači da ne vraćaju nikakvu vrijednost. Obavezno moraju biti prisutne kako bi program ispravno radio. Funkcija `setup()` izvodi se samo jednom, a služi za definiranje i inicijalizaciju početnih varijabli. Funkcija `loop()` izvršava sve naredbe beskonačno mnogo puta.

### 7.7.1. Biblioteka *BluetoothSerial.h*

Kako bi mogli iskoristiti Bluetooth modul u EPS32 mikrokontroleru, uključena je biblioteka zvana `BluetoothSerial.h` razvijena od strane Espressif-a. Uključuje se pomoću naredbe `#include` na samom početku skice. Nakon uspješnog uključivanja biblioteke u projekt možemo iskoristiti sve mogućnosti i funkcije koje nam nudi biblioteka. Jedna od naredbi je `BluetoothSerial` koja stvara objekt klase *BluetoothSerial*, a omogućava inicijalizaciju ESP32 bluetooth sučelja i uspostavu serijske komunikacije preko Bluetooth veze. Metode klase `BluetoothSerial` koji se koriste u ovom primjeru su:

- `Begin()`
- `Avaible()`
- `readString()`

### 7.7.2. Opis *Setup()* bloka

Unutar `Setup()` bloka moramo definirati one vrijednosti za koje znamo da će se izvršiti samo jednom. Metoda `pinMode(prvi_parametar, drugi_parametar)` konfigurira zadani pin na mikrokontroleru kao ulaz ili izlaz. Prvi parametar određuje o kojem se točno pinu radi, u našem slučaju je to pin 27, a drugi parametar može poprimiti vrijednost *INPUT* ili *OUTPUT*. Budući da je na pin 27 spojen relej koji kontrolira mali motor, potrebno je odabrati *OUTPUT* način rada. Klasa `Serial` se koristi za komunikaciju između mikrokontrolera i računala preko serijskog priključka. Uz pomoć `Serial.begin(parametar)` funkcije možemo pratiti prijenos svih podataka na definiranoj frekvenciji serijskog kanala. Zadnjom naredbom u `setup()` funkciji nad kreiranim `BluetoothSerial` objektom inicijaliziramo Bluetooth vezu tako da kao parametar unesemo string koji će reprezentirati ESP32 kao Bluetooth uređaj. Kada će neki

Bluetooth uređaj pokrenuti proces otkrivanja drugih uređaja, ESP32 uređaj će biti otkriven kao *Smart Control* među ponuđenim rezultatima.

```
7     BluetoothSerial SerialBT;
8     String ulaz = "";
9
10     void setup() {
11          pinMode(27, OUTPUT);
12          Serial.begin(115200);
13          SerialBT.begin("Smart Control");
14    }
```

Isječak kôda 1: Postavljanje i inicijalizacija varijabli unutar *setup()* metode

### 7.7.3. Opis *Loop()* bloka

Isječak kôda 2 prikazuje `loop()` funkciju u kojoj se koristi `SerialBT.available()` metoda koja dopušta izvršavanje sljedećeg bloka naredbi ako postoji komunikacija između uređaja. Varijabla *ulaz* služi za pohranu svih bitova pročitanih s Bluetooth komunikacijskog kanala. Ako varijabla ulaz poprimi niz znakova *upali*, tada će pin 27 na ESP32 mikrokontroleru primiti napon od 3.3V zbog parametra *HIGH* u metodi *`digitalWrite()`*. Ako je *ulaz* jednak nizu znakova *ugasi*, tada pin 27 neće primiti napon. Naredba `println(ulaz)` klase `Serial` ispisuje sve dolazne podatke pročitane s Bluetooth komunikacijskog kanala na serijski priključak kao čitljiv ASCII tekst s prelaskom u novu liniju. Kasnije u analizi Kotlin kôda postavit ćemo da se na pritisak gumba *UPALI* pošalje znak *upali*, a na pritisak tipke *UGASI* pošalje znak *ugasi*.

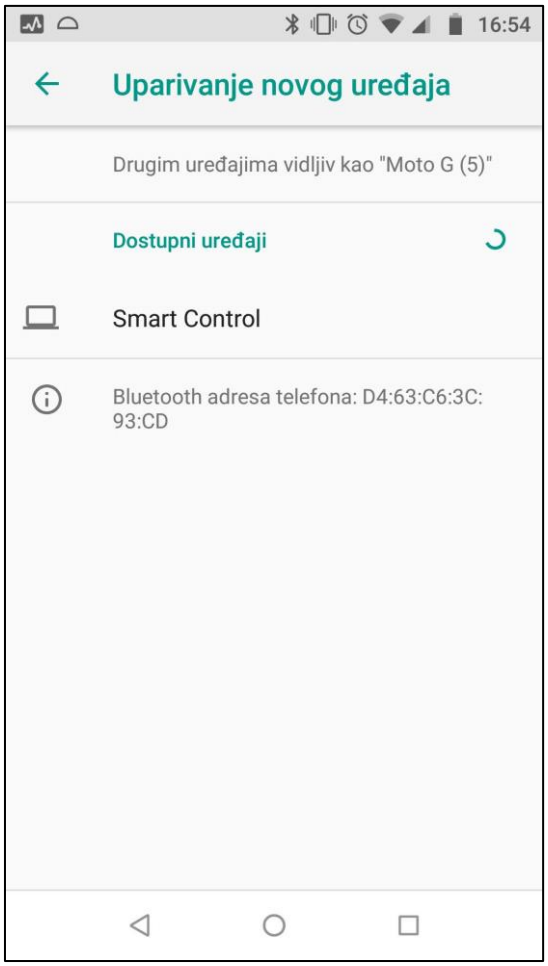
```
16    void loop() {
17          while(SerialBT.available()){
18                ulaz = SerialBT.readString();
19                Serial.println(ulaz);
20                Serial.print("\n");
21                if(ulaz == "upali")
22                      digitalWrite(27, HIGH);
23                else if(ulaz == "ugasi")
24                      digitalWrite(27, LOW);
25          }
26     }
```

Isječak kôda 2: Prikaz *loop()* metode za upravljanje ESP32 mikrokontrolera i pumpe

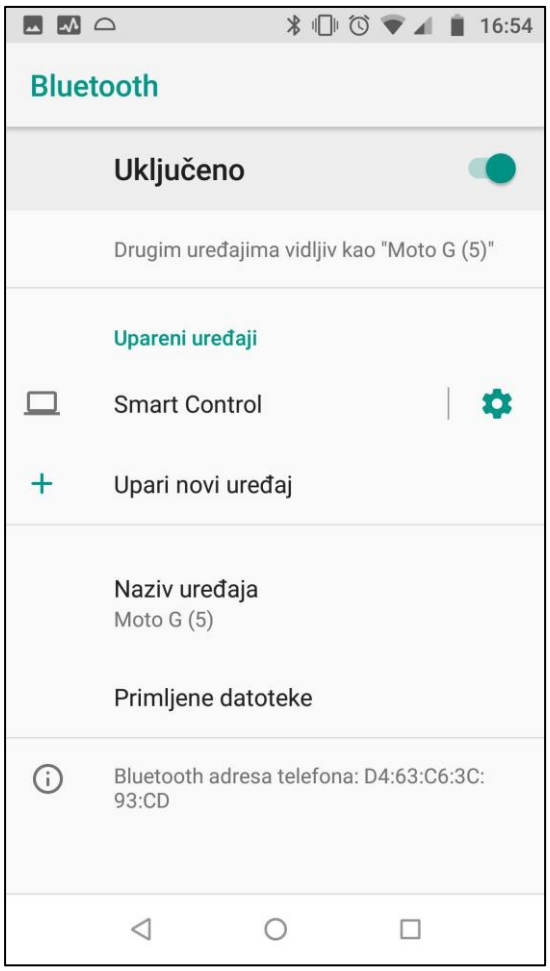
### 7.7.4. Postupak uparivanja uređaja

Na sljedećim slikama možemo vidjeti da je ESP32 na mobilnom uređaju prepoznat kao Bluetooth uređaj na kojega se možemo spojiti. Slike 14. i 15. prikazuju dijaloge sustava za uparivanje. Prva slika prikazuje proces pronalaska i uparivanja svih dostupnih Bluetooth uređaja. Uređaj *Smart Control* je zapravo EPS32 mikrokontroler, a postupak dodjele imena opisan je u prethodnom poglavlju. Klijent, u ovom slučaju pametni uređaj, emitira upit za pronalazak ostalih Bluetooth uređaja i svaki detektirani uređaj pita za svoje korisničko ime. Pritiskom na željeni uređaj započinje proces uparivanja uređaja, a uspješno ostvarenu konekciju možemo vidjeti na slici 15. Pod kategorijom upareni uređaji primjećujemo da se nalazi uređaj naziva *Smart Control* što potvrđuje da su uređaji uspješno upareni i mogu međusobno razmjenjivati podatke koristeći Bluetooth sučelje.

Time je zaključen programski dio sa strane ESP32 sustava, a u nastavku će biti objašnjena izrada mobilne aplikacije te način komunikacije Android mobilnog uređaja s drugim Bluetooth uređajem.



Slika 14:Otkrivanje ESP32 uređaja



Slika 15: EPS32 spojen sa mobitelom

## 7.8. Izrada Kotlin aplikacije za Android platformu

Za izradu mobilne aplikacije koja će komunicirati s ESP32 sustavom preko Bluetooth-a izabran je Kotlin programski jezik. Neki od razloga su što je moderan programski jezik otvorenog kôda, službeno je podržan od Google-a za izradu Android aplikacija te ga je vrlo lako uključiti u projekt unutar Googleovog službenog integriranog razvojnog okruženja zvanog Android Studio. Za razliku od Java programskog jezika, koji je originalno podržan u Android Studio IDE, nudi kraće pisanje kôda čime je sama struktura bolje čitljiva. Ako programeri žele iskoristiti neke od Java funkcionalnosti, Kotlin nudi interoperabilnost s Javom, bez potrebe za migracijom čitavog kôda. Danas je Kotlin zajednica vrlo razvijena i prema Google-u 60% svih aplikacija na Google Store-u je Kotlin bazirano (Kotlin Foundation, bez dat.).

### 7.8.1. Bluetooth API

Android platforma ima vrlo detaljno razrađenu podršku za Bluetooth API. Sva dokumentacija je dostupna na Googleovoj stranici za pomoć developerima. Bluetooth API je automatski dostupan u Android Studio IDE, a programeru omogućuje:

- skeniranje drugih Bluetooth uređaja
- mogućnost spajanja sa drugim Bluetooth uređajima preko usluge otkrivanja
- uspostavljanje RFCOMM kanala
- prijenos podataka na i s drugih uređaja
- upravljanje višestrukim vezama (Google Developers, 2020)

Da bi uređaji mogli međusobno prenosili podatke, prvo moraju formirati komunikacijski kanal pomoću postupka uparivanja. Jedan uređaj mora biti dostupan za dolazne zahtjeve dok drugi uređaj ostale uređaje pomoću usluge otkrivanja. Nakon što otkriveni uređaj prihvati zahtjev za uparivanje, dva uređaja dovršavaju postupak spajanja gdje razmjenjuju sigurnosne ključeve. Kako bi se smanjilo vrijeme čekanja prilikom kasnijeg povezivanja, uređaji pohranjuju ključ za ponovnu upotrebu u memoriju. Nakon što je postupak uparivanja i spajanja završen, uređaji su spremni za razmjenu informacija. Kada je sesija završena, uređaj koji je pokrenuo zahtjev za uparivanje zatvara kanal s drugim uređajem. To ne znači da je veza potpuno prekinuta, već je privremeno zatvorena. Time se uređaji mogu automatski ponovno povezati tijekom buduće sesije sve dok su u međusobnom dometu i klijentski uređaj nije uklonio dugog uređaja u popisu uparenih uređaja (Google Developers, 2020).

### 7.8.2. Dozvole za korištenje Bluetooth-a

Kako bi mogli koristiti Bluetooth u aplikaciji potrebno je unutar AndroidManifest.xml datoteke uključiti dvije dozvole. Dozvole su aplikaciji potrebne za pristup zaštićenim dijelovima sustava ili drugih aplikacija. Sljedeći isječak kôda prikazuje sva potrebna dopuštenja za korištenje Bluetooth funkcionalnosti, a to su BLUETOOTH i BLUETOOTH_ADMIN (Google Developers, 2020). Prva dozvola je potrebna za obavljanje bilo kakve Bluetooth komunikacije, poput zahtjeva za povezivanjem, prihvaćanja veze i prijenosa podataka (Google Developers, 2020). Drugu dozvolu je potrebno uključiti ako preko aplikacije želimo pokrenuti proces otkrivanja uređaja ili manipulirati postavke Bluetooth-a (Google Developers, 2020).

```
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
```

Isječak kôda 3: Dozvole za uključivanje Bluetooth funkcionalnosti preko aplikacije

### 7.8.3. Uključivanje Bluetootha

Prije nego uspostavimo komunikaciju s drugim Bluetooth uređajem potrebno je provjeriti je li uključen Bluetooth. Ako nije uključen potrebno ga je uključiti, što možemo napraviti uz pomoć `BluetoothAdapter` klase. Ta klasa predstavlja Bluetooth adapter lokalnog uređaja, a omogućuje izvršavanje osnovnih Bluetooth zadataka, kao što su pokretanje otkrivanja uređaja, ispitivanje popisa povezanih (uparenih) uređaja i uspostavljanje Bluetooth uređaja pomoću poznate adrese uređaja (Google Developers, 2020).

`BluetoothAdapter` klasa potrebna je za bilo kakvu aktivnost vezanu za Bluetooth. Kako bi dohvatili lokalni adapter samog uređaja, potrebno je pozvati `getDefaultAdapter()` metodu. Unutar isječka kôda 4 možemo primijetiti prvi *if* blok na liniji 31 u kojem provjeravamo podržava li uređaj Bluetooth funkcionalnosti. To radimo tako da provjerimo je li objekt `bluetoothAdapter` jednak nuli. Ako je uvjet istinit, prikazuje se kratka obavijest koja je definirana preko `toast` klase.

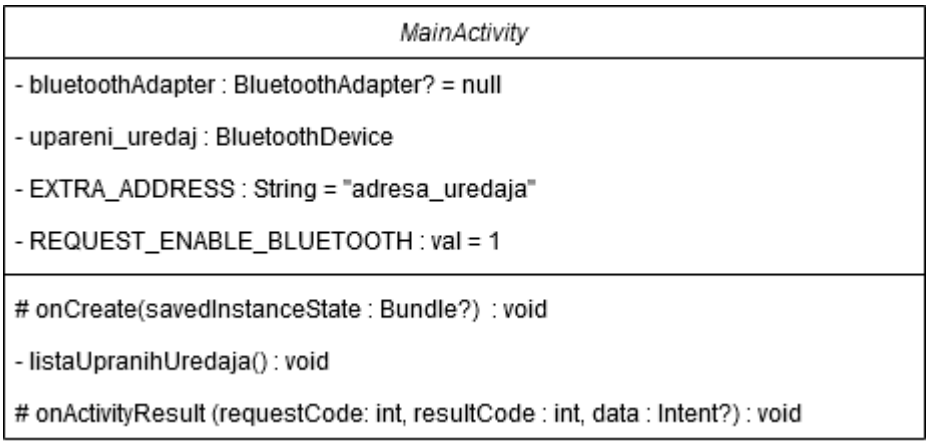
```
26      override fun onCreate(savedInstanceState: Bundle?) {
27          super.onCreate(savedInstanceState)
28          setContentView(R.layout.activity_main)
29
30          bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()
31          if(bluetoothAdapter == null) {
32              toast("Ovaj uređaj ne podržava Bluetooth")
33              return
```

```
34          }
35          if(!bluetoothAdapter!!.isEnabled) {
36              val enableBluetoothIntent =
37              Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE)
38              startActivityForResult(enableBluetoothIntent,
39              REQUEST_ENABLE_BLUETOOTH)
40          }
41
42          select_device_refresh.setOnClickListener{listaUperanihUredaja()}
43      }
```

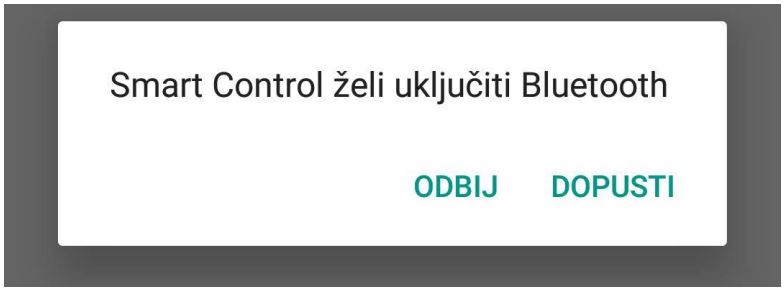
Isječak kôda 4: Uključivanje Bluetooth-a preko aplikacije

Drugi *if* blok provjerava postoji li Bluetooth funkcionalnost na uređaju. Ako postoji, pokreće se nova aktivnost u kojoj korisnik može uključiti Bluetooth uređaj.

Aplikacija je zamišljena tako da se prilikom pokretanja automatski provjerava uključenost Bluetooth-a, zbog čega se navedena dva *if* bloka nalaze unutar `onCreate` funkcije koja se pokreće prilikom pokretanja aktivnosti. Aktivnost predstavlja jedan prozor aplikacije, slično kao u web pregledniku gdje možemo imati različite kartice, a na svakoj od tih kartica se nalazi drugačiji sadržaj. Na slici 16 je prikaz UML (eng. *Unified Modeling Language*) dijagrama klase za navedeni isječak kôda.




Slika 16: UML dijagram klase *MainActivity*




Slika 17: Pop up prozor

Na slici 17. možemo vidjeti prikaz dijaloga nakon izvršenog drugog *if* bloka. Odabirom opcije *DOPUSTI* uključuje će se Bluetooth i aplikacija obavještava korisnika o uspješnom uključivanju Bluetooth-a.

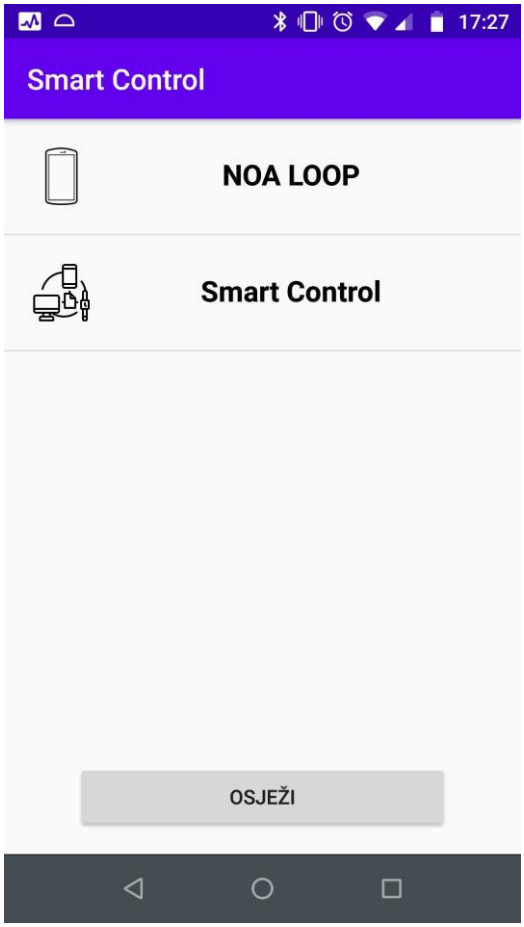
### 7.8.4. Konekcija uređaja

Nakon što je uspješno uključen Bluetooth na mobilnom uređaju otvara se početna aktivnost prikazana na slici 18. Tipka *Osvježi* služi za prikaz svih Bluetooth uređaja koji su upareni s mobilnim uređajem. Kako bi dobili popis svih uređaja koji su upareni potrebno je koristiti metodu `getBondedDevices()`. Ta metoda vraća skup `BluetoothDevice` objekata koji predstavljaju uparene uređaje. Ako postoji upareni uređaj, unutar `for` petlje dohvatit će se ime, adresa i tip uparenog uređaja. Ako niti jedan od uređaja nije uparen, aplikacija će prikazati kratku obavijest o navedenom problemu.

```
46      private fun listaUperanihUredaja() {
47          var btIme : ArrayList<String> = ArrayList()
48          var btSlika : ArrayList<Drawable> = ArrayList()
49          var btAdresa : ArrayList<String> = ArrayList()
50          uparen_uredaj = bluetoothAdapter!!.bondedDevices
51
52          if (!uparen_uredaj.isEmpty()) {
53             for (device: BluetoothDevice in uparen_uredaj) {
54                  btIme.add(device.name)
55                  btAdresa.add(device.address)
56                  btSlika.add(dohvatiSlikuKlaseUredaja
57                     (device.bluetoothClass.majorDeviceClass)!!)
58                  Log.i("device", "" + device)
59             }
60          } else {
61              toast("Nije pronađen niti jedan bluetooth uređaj")
62          }
```

Isječak kôda 5: Metoda za dodavanje imena, adrese i tipa uparenog uređaja u određenu listu

Kako bi uređaji mogli međusobno razmjenjivati podatke, potrebno je stvoriti komunikaciju između uređaja. Za ostvarivanje takve komunikacije, potreba je adresa drugog uređaja. Na liniji 49 u isječku kôda 5 inicijalizirali smo listu objekata tipa `String` u koju ćemo kasnije pohraniti sve adrese uparenih uređaja. Prolaskom kroz cijelu listu uparenih uređaja u varijablu *btAdresa* na liniji 55 dodajemo sve adrese uparenih uređaja. Na isti način smo dohvatili ime i tip uparenog uređaja.



Slika 18: Popis svih uperenih uređaja

Slika 18. prikazuje aktivnost aplikacije nakon pritiska tipke *Osvježi*. Jedan element u listi sastoji se od slike i naziva uparenog uređaja. Trenutno aplikacija može raspoznati da li se radi o mobilnom uređaju ili nekom drugom Bluetooth uređaju. Na slici 18. vidimo prikaz u kojem su uparena dva uređaja. Jedan je mobilni uređaj pod nazivom *NOA LOOP*, a drugi je *Smart Control* IoT sustav. Klasa koja definira konstante glavnih klasa uređaja spojenih preko Bluetooth-a je `BluetoothClass.Device.Major`. Isječak kôda 6 prikazuje metodu koja kao rezultat vraća sliku ovisno o unesenom argumentu. Uneseni argument je upravo neka od konstanti `Major` klase, a može biti npr. računalo koje je definirano konstantom 256 ili pametni telefon definiran konstantom 512. Ostali tipovima klasa uređaja koji su definirani `Major` klasom su: AUDIO_VIDEO, HEALTH, IMAGING, MISC, NETWORKING, PERIPHERAL, TOY, UNCATEGORIZED, WEARABLE.

```
74  private fun dohvatiSlikuKlaseUredaja(tip: Int): Drawable? {
75      return when (tip) {
76          BluetoothClass.Device.Major.COMPUTER -> ContextCompat.getDrawable(
77              this,
78              R.drawable.ic_smart_device
79          )
80          BluetoothClass.Device.Major.PHONE -> ContextCompat.getDrawable(
81              this,
82              R.drawable.ic_phone
83          )
84          else -> ContextCompat.getDrawable(this, R.drawable.ic_smart_device)
85      }
86  }
```

Isječak kôda 6: Metoda koja vraća sliku tipa uređaja ovisno o konstanti uređaja

Budući da se na pritisak određenog elementa u listi uređaja otvara nova aktivnost, potrebno je proslijediti adresu kao parametar novoj aktivnosti. U Kotlinu to radimo preko `companion` objekt-a. Unutar tog objekta potrebno je definirati sve varijable koje želimo proslijediti sljedećoj aktivnosti. Isječak kôda 7 na liniji 22 i 23 prikazuje na koji način to radimo. U liniji 66 pozivamo `onItemClickListener()` metodu nad `AdapterView` objektom, a služi za definiranje radnji nakon pritiska na objekt nad kojim je pozvana. Metoda sadrži četiri parametra (`parent, view, position, id`). Nama je potreban samo `position` parametar zbog toga što taj parametar određuje poziciju svakog Bluetooth uređaja unutar definirane liste. Poznavanjem pozicije uređaja unutar liste lagano možemo saznati adresu određenog uređaja. Na liniji 68 u varijablu `adresa` pohranjujemo adresu nakon pritiska na određeni uređaj. Metodom `putExtra` prosljeđujemo dohvaćenu adresu sljedećoj aktivnosti.

```
22    companion object {
23        val EXTRA_ADDRESS: String = "adresa_uredaja"
24    }
…

64    val adapter = PrikazBluetoothUredaja(this, btIme, btSlika, btAdresa)
65    lvPopisUredaja.adapter = adapter
66    lvPopisUredaja.onItemClickListener = AdapterView.OnItemClickListener { _, _,
67    position, _ ->
68    val adresa = btAdresa[position]
69    val intent = Intent(this, ControlActivity::class.java)
70    intent.putExtra(EXTRA_ADDRESS, adresa)
71    startActivity(intent)
```

Isječak kôda 7: Prosljeđivanje varijable drugoj aktivnosti

Najčešći tip Bluetooth utičnice je RFCOMM i podržava ga Bluetooth API. Za uspješnu uspostavu konekcije potrebno je kreirati `BluetoothSocket` objekt. Nakon toga potrebno je pozvati `createInsecureRfcommSocketToServiceRecord(UUID)` metodu koja će stvoriti

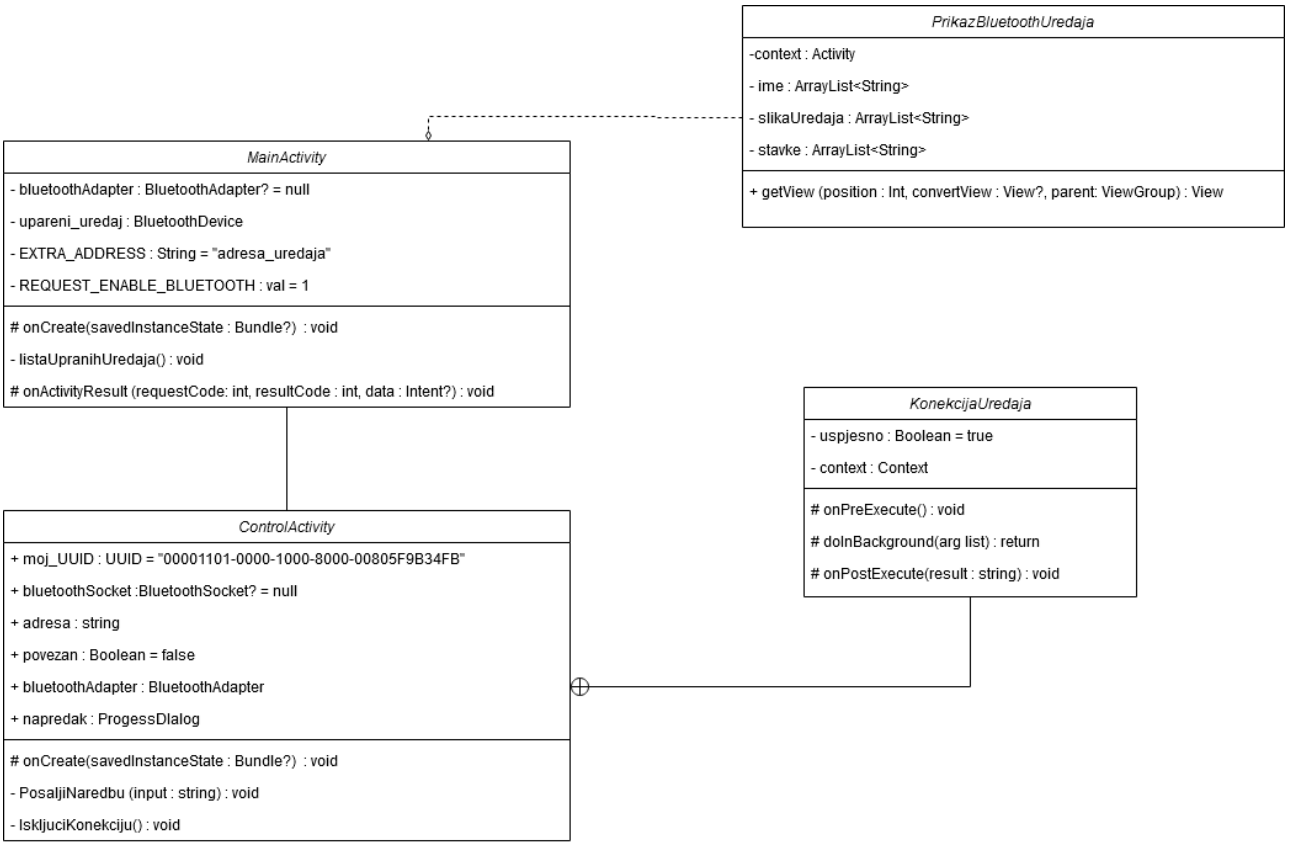
nesigurnu odlaznu vezu (Google Developers, 2020). Nesigurnu vezu preporučeno je kreirati ako ne šaljemo osjetljive informacije preko Bluetooth komunikacijskog signala. Ako šaljemo osjetljive informacije i ne želimo mogućnost napada *srednjeg čovjeka* (eng. man-in-the-middle) tada je potrebno koristiti `createRfcommSocketToServiceRecord(UUID)` metodu (Google Developers, 2020). Takav kanal je kriptiran i potrebna je uspostava autentifikacije. Budući da se preko *SmartControl* aplikacije šalju samo naredbe drugom Bluetooth uređaju dovoljno je koristiti nesigurnu vezu. Zadnji korak je pozvati `connect()` metodu nad `BluetoothSocket` objektom kako bi uspostavili konekciju. Metoda će također napraviti SDP (eng. *Service Discovery Protocol*) pretragu zadanog UUID-a. Metodu `cancelDiscovery()` potrebno je pozvati kako bi zaustavili pretragu za drugim Bluetooth uređajima. Proces otkrivanja uređaja vrlo je zahtjevan proces pa se njegovim isključivanjem oslobađa memorija, a vrijeme uspostave konekcije znatno se smanjuje. U priloženom isječku kôda 8 možemo primijetiti da je na liniji 83 pozvana metoda `cancelDiscovery()` nad `BluetoothAdapter` objektom te u sljedećoj liniji pozvana metoda `connect()` nad instancom objekta `bluetoothSocket` kako bi uspostavili odlaznu konekciju.

```
75   override fun doInBackground(vararg p0: Void?): String? {
76     try {
77        if(bluetoothSocket == null || !povezan) {
78           bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()
79           val device: BluetoothDevice =
80            bluetoothAdapter.getRemoteDevice(adresa)
81           bluetoothSocket =
82           device.createInsecureRfcommSocketToServiceRecord(moj_UUID)
83           BluetoothAdapter.getDefaultAdapter().cancelDiscovery()
84            bluetoothSocket!!.connect()
85        }
86     } catch (e: IOException) {
87        uspjesno = false
88        e.printStackTrace()
89    }
90    return null
91 }
```

Isječak kôda 8: Povezivanje Bluetooth uređaja preko *AsyncTask* klase

Proces povezivanja odrađen je u pozadini aplikacije uz pomoć `AsyncTask` klase. Klasa mora biti nadjačana s barem jednom metodom koja se zove `doInBackground()`. Unutar metode je odrađen cijeli postupak povezivanja s drugim Bluetooth uređajem, a implementacija je prikazana u isječku kôda 8. Kako bi izvršili pozadinski zadatak unutar aplikacije, potrebno je pozvati metodu `execute()` nad definiranom klasom. Slika 19 prikazuje UML dijagram klasa za navedene isječke kôda. Klasa *MainActivity* implementira sve funkcionalnosti vezane za uključivanje Bluetooth-a na uređaju i prikaz uparenih uređaja. Klasa *ControlActivity*

implementira sve funkcionalnosti vezane za kontrolu uparenog IoT uređaja. *ControlActivity* također ugnježđuje klasu *KonekcijaUređaja* u kojoj se ostvaruje komunikacija između uređaja.




Slika 19: UML dijagram klasa *MainAcitvity*, *ControlActivity*, *KonekcijaUredaja*

### 7.8.5. Komunikacija s drugim Bluetooth uređajem

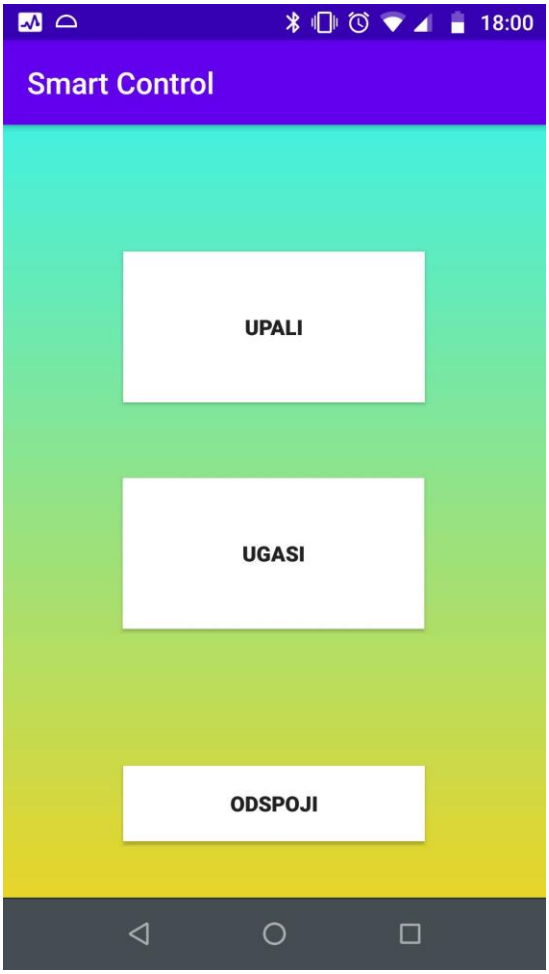
Nakon uspješno kreiranog komunikacijskog kanala moguće je prosljeđivati podatke između Bluetooth uređaja. Nad kreiranim `BluetoothSocket` objektom možemo pozvati klase poput `InputStream` i `OutputStream`. Klasa `InputStream` zadužena je za primanje podataka o drugom uređaju, a klasa `OutputStream` za slanje podataka drugom uređaju. U ovom primjeru koristimo metodu `OutputStream.write()` za slanje podataka drugom uređaju.

Slika 20. prikazuje jednostavno sučelje za kontroliranje ESP32 uređaja i svih komponenti spojenih na mikrokontroler. Metoda `setOnClickListener()` unutar isječka kôda 9 poziva metodu `PosaljiNaredbu()` nakon pritiska tipke *UPALI* ili *UGASI*. Metoda `PosaljiNaredbu()` kao parametar uzima niz znakova i šalje podatke mikrokontoroleru preko RFCOMM kanala. Na mikrokontroleru prenesen je Arduino kôd koji sluša dolazne signale preko Bluetooth-a. Ako zaprimi signal s porukom *upali,* pokrenut će pumpu i time započeti proces zalijevanja. U suprotnom, ako zaprimi signal sa porukom *ugasi* isključit će pumpu i zaustaviti proces zalijevanja.

```
34      motor_on.setOnClickListener { PosaljiNaredbu("upali") }
35      motor_off.setOnClickListener { PosaljiNaredbu("ugasi") }

…
39      private fun PosaljiNaredbu(input: String) {
40          if (bluetoothSocket != null) {
41              try{
42                  bluetoothSocket!!.outputStream.write(input.toByteArray())
43              } catch(e: IOException) {
44                  e.printStackTrace()
45              }
46          }
47      }
```

Isječak kôda 9: Metoda za slanje naredbi drugom Bluetooth uređaju




Slika 20: Sučelje za upravljanje pumpe

### 7.8.6. Prekidanje Bluetooth komunikacije

Kada želimo prekinuti komunikaciju s drugim uređajem, potrebno je pozvati metodu `close()` nad `BluetoothSocket` objektom. Sljedeći blok kôda prikazuje metodu `IskljuciKonekciju()`. Na liniji 50 unutar isječka kôda 10 provjerava se postojanje aktivne konekcije. Ako postoji konekcija, instanca `BluetoothSocket` objekta postavlja se na *null* vrijednost i oslobađaju se svi povezani interni resursi vezani za komunikaciju preko Bluetooth-a. Na kraju trebamo pozvati metodu `finish()` koja služi za izlaz iz trenutne aktivnosti i povratka na početni zaslon.

```
49      private fun IskljuciKonekciju() {
50          if (bluetoothSocket != null) {
51              try {
52                  bluetoothSocket!!.close()
53                  bluetoothSocket = null
54                  povezan = false
55              } catch (e: IOException) {
56                  e.printStackTrace()
57              }
58          }
59          finish()
60      }
```

Isječak kôda 10: Prekidanje Bluetooth komunikacije

Ovime je završeno poglavlje *Smart Control* aplikacije. Prikazan je osnovni koncept izrade Android mobilne aplikacije koja koristi Bluetooth kao komunikacijsko sredstvo za upravljanje IoT uređajem. Objašnjen je odabir dizajna sustava i koraci u prototipiranju i testiranju aplikacije. Prikazane su i objašnjene glavne funkcionalnosti sustava preko analize programskog kôda. Preko Arduino razvojnog okruženja prikazani su početni koraci uspostave prototipa ESP32 sustava, a analizom Kotlin programskog jezika objašnjen je način otkrivanja uparenog uređaja, definiranje tipa uparenog uređaja te sam proces povezivanja i slanja podatka drugom Bluetooth uređaju. U sljedećem poglavlju dan je osvrt na ciljeve i metode rada te autorov komentar o temi i postignutim ciljevima rada.

# 8. Zaključak

Internet stvari kao tehnološki koncept puno je napredovao u zadnjih nekoliko godina i sigurno će dalje napredovati puno većom brzinom. Implementacija modernih IoT aplikacija u razne sustave poput medicine, prometa, agrokulture, obrazovanja ima za cilj poboljšati i unaprijediti sustave. To omogućuje stvaranje modernog svijeta u kojem ljudski i IoT stvari djeluju u harmoniji. Glavni nedostatak Interneta stvari je veliki izbor tehnologija i protokola što može zbuniti proizvođače i programere u izradi IoT sustava.

Cilj ovog rada je opisati procese dizajniranja i prototipiranja mobilne aplikacije u integraciji Interneta stvari uz pomoć jednostavnog sustava. Kroz razvoj *Smart Control* aplikacije prikazana je uporaba Bluetooth-a kao komunikacijskog sredstva. Predstavljen je funkcionalni prototip sa svim osnovnim funkcionalnostima poput uspostave konekcije između uređaja i prosljeđivanja podataka. Prototip sklopovskog dijela aplikacije omogućio je razumijevanje IoT sustava na nižoj razini programiranjem malog mikrokontrolera. Mobilnom aplikacijom prikazani su različiti problemi i načini upravljanja IoT uređajem. Ovim diplomskim radom pokazao sam kako je moguće uspostaviti funkcionalni IoT sustav bez prevelikog znanja o IoT uređajima i mobilnom programiranju.

Što se tiče budućnosti *Smart Control* aplikacije, sustav bi mogao implementirati senzor temperature i vlažnosti za kvalitetnije praćenje biljaka i prikaz informacija u realnom vremenu. Testiranjem i implementacijom novih tehnologija poput Bluetooth LE (eng. *Low Energy*) omogućio bi se brži prijenos i manja potrošnja IoT uređaja. Drugim riječima, broj mogućih pravaca i rješenja kojim može težiti neki IoT projekt je velik kao ocean.

Programskom kôdu *Smart Control* aplikaciji možete pristupiti na sljedećem linku: https://github.com/tgodek/Smart-Control

# Popis slika

# Popis tablica

# Popis kôdova

# Popis literature


Afaneh, M. (2020). *Wireless Connectivity Options for IoT Applications – Technology Comparison* [Blog post]. Preuzeto 6.9.2020. s https://www.bluetooth.com/blog/wireless-connectivity-options-for-iot-applications-technology-comparison/

Android (bez dat.). *AndroidAuto*. Preuzeto 8.9.2020. s https://www.android.com/auto/#next-steps

BehrTech. (bez dat.). *6 Leading Types of IoT Wireless Tech and Their Best Use Cases* [Blog post]. Preuzeto 6.9.2020. s https://behrtech.com/blog/6-leading-types-of-iot-wireless-tech-and-their-best-use-cases/

ComputerNetworkingNotes. (2018). *Types of Wireless Network Explained with Standards*. Preuzeto 6.9.2020. s https://www.computernetworkingnotes.com/ccna-study-guide/types-of-wireless-network-explained-with-standards.html

Dunkels, A. (2019). *What is an IoT Prototype?* [Blog post]. Preuzeto 6.9.2020. s https://www.thingsquare.com/blog/articles/iot-prototype/

Espressif Systems. (bez dat.). *About Espressif*. Preuzeto 2.9.2020. s https://www.espressif.com/en/company/about-espressif

Espressif Systems. (bez dat.). *ESP32 Series Datasheet*. Preuzeto 2.9.2020. s https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Farooq, M. U., Waseem, M., Mazhar, S., Khairi, A., & Kamal, T. (2015). A Review on Internet of Things (IoT). *International Journal of Computer Applications*, *113*, 1–7. https://doi.org/10.5120/19787-1571

Google Developers. (2020). *Bluetooth overview*. Preuzeto 2.9.2020. s https://developer.android.com/guide/topics/connectivity/bluetooth

Huang, A., & Rudolph, L. (2005). *Bluetooth for Programmers*. Cambridge University Press.

Kotlin Foundation. (bez dat.). *Using Kotlin for Android Development*. Preuzeto 8.9. 2020. s https://kotlinlang.org/docs/reference/android-overview.html

Lazarevich, K. (2020). *How to Design IoT Apps: UX/UI Design for IoT Mobile Apps* [Blog post]. Preuzeto 6.9.2020 s https://www.digiteum.com/design-iot-apps-ux-ui-mobile-apps

Muntjir, M., Rahul, M., & Alhumiany, H. (2017). An Analysis of Internet of Things(IoT): Novel Architectures, Modern Applications,Security Aspects and Future Scope with Latest Case Studies. *Building Services Engineering Research and Technology*, *06*.

Pandey, A., & Bagwe, S. (2014). *Bluetooth Protocol Stack*. Preuzeto 8.9.2020. s https://www.ques10.com/p/2700/bluetooth-protocol-stack-1/

Ratner, J. (2020). 4 Ways IoT is Changing Mobile App Development [Blog post]. Preuzeto 2.9.2020. s https://www.verypossible.com/insights/4-ways-iot-is-changing-mobile-app-development

SerialIO. (bez dat.). *What's the difference between Bluetooth LE and Bluetooth SPP (BLE vs SPP)?* Preuzeto 1.9.2020. s https://www.serialio.com/faqs/whats-difference-between-bluetooth-le-and-bluetooth-spp-ble-vs-spp

Soft Service Company. (bez dat.). *Bluetooth Framework and RFCOMM Protocol*. Preuzeto 2.9.2020 s https://www.btframework.com/rfcomm.htm

Somayya Madakam, Ramaswamy, R., & Tripathi, S. (2015). Internet of Things (IoT): A Literature Review. *Journal of Computer and Communications*, *3*, 164–173. https://doi.org/10.4236/jcc.2015.35021

Teel, J. (2017). *How to Choose the Best Development Kit: The Ultimate Guide for Beginners* [Blog post]. Preuzeto 7.9.2020. s https://predictabledesigns.com/how-to-choose-the-best-development-kit-the-ultimate-guide-for-beginners/

van Diessen, R. (bez dat.). *IoT Prototyping: Everything you Need to Know* [Blog post]. Preuzeto 6.9.2020. s https://www.applyanze.com/iot-prototyping-everything-you-need-to-know/

Vermesan, Dr. O., Friess, Dr. P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, Dr. A., Jubert, I. S., Mazura, Dr. M., Harrison, Dr. M., Eisenhauer, Dr. M., & Doody, Dr. P. (2011). Internet of Things Strategic Research Roadmap. In *Internet of Things—*

*Global Technological and Societal Trends From Smart Environments and Spaces to Green ICT* (p. 316). Danska, River Publishers.