UREĐAJ ZA GENERIRANJE KRIPTIRANIH LOZINKI

Stević, Marin

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:886953

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-02-23**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

UREĐAJ ZA GENERIRANJE KRIPTIRANIH LOZINKI

Diplomski rad

Marin Stević

Osijek, 2021.

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. PREGLED POSTOJEĆIH RJEŠENJA 2

2.1. LastPass 2

2.2. YubiKey 3

2.3. Snopf 3

2.4. Idejno rješenje ovog rada 4

3. KORIŠTENI PROGRAMSKI ALATI I TEHNOLOGIJE 5

3.1. Android Studio 5

3.2. Arduino 6

3.1. Autodesk Eagle 6

3.2. ESP32-S2 7

3.3. HTTPS 8

4. REALIZACIJA SUSTAVA 9

4.1. Model sustava 9

4.1.1. Unos podataka 9

4.1.2. Slanje podataka 10

4.1.3. Primanje i obrada podataka 10

4.1.4. Generiranje lozinke 11

4.1.5. Unos lozinke 11

4.2. Sklopovsko rješenje 12

4.2.1. Prvi dizajn 13

4.2.2. Drugi dizajn 14

4.2.3. 3D model uređaja 16

4.3. Programsko rješenje 16

4.3.1. Priprema Arduino razvojnog okruženja 17

4.3.2. Pokretanje WiFi pristupne točke 20

4.3.3. Pokretanje HTTPS servera i generiranje certifikata 20

4.3.4. Upravljanje RGB LE diodom 22

4.3.5. Primanje podataka 23

4.3.6. Hashiranje 24

4.3.1. BASE64 kodiranje i slanje podataka putem USB veze 25

5. TESTIRANJE SUSTAVA 27

5.1. Programiranje uređaja 27

5.2. Instaliranje Android aplikacije 28

5.3. Korištenje uređaja 29

6. ZAKLJUČAK 34

LITERATURA 35

SAŽETAK 36

ABSTRACT 37

ŽIVOTOPIS 38

PRILOG 1. SHEMATSKI PRIKAZ UREĐAJA 39

1. UVOD

Pristup internetu nikada nije bio jednostavniji i pristupačniji nego u današnje vrijeme, ali za pristup sadržaju sve više stranica zahtjeva posjedovanje neke vrste osobnog računa. Za otvaranje računa potrebne su određene informacije poput: imena, prezimena, e-mail adrese u nekim slučajevima i kontakt broja i adrese prebivališta. Dok su sve te informacije često jednake i lako pamtljive jedan komad informacije koji je najvažniji je lozinka. Lozinka predstavlja tajnu kombinaciju slova, brojki i drugih znakova kojima pristupamo svojem e-računu. Korištenje iste lozinke za pristup više e-računa predstavlja lošu praksu zato što ne možemo znati kolika je razina sigurnosti pojedine stranice. Rješenje problema sigurnosti je vrlo jednostavno, koristiti unikatnu lozinku za pojedinu stranicu. Dok se to rješenje čini jednostavnim vrlo brzo dolazimo do dva velika problema: nemogućnosti pamćenja velike količine različitih lozinki i preklapanja među lozinkama.

Cilj ovoga rada je izraditi uređaj koji bi predstavljao jednostavno rješenje za prethodno navedene probleme. Uređaj bi trebao imati sljedeće mogućnosti: generiranje sigurnih i originalnih lozinki u stvarnom vremenu, spremanje odnosno ponovno generiranje iste lozinke za traženi e-račun, automatizirani unos lozinke i povrat svih lozinki u slučaju gubitka uređaja.

U idućem poglavlju su opisana tri različita sustava od kojih svaki rješava problem generiranja i spremanja lozinki na svoj jedinstven način. Treće poglavlje sadrži opis korištenih tehnologija za ostvarivanje zadatka zadanog ovim radom. Četvrto poglavlje prikazuje realizaciju sustava od modela do prototipa. Peto poglavlje pokazuje ispravnost i način rada sustava uz određene primjere.

1.1. Zadatak završnog rada

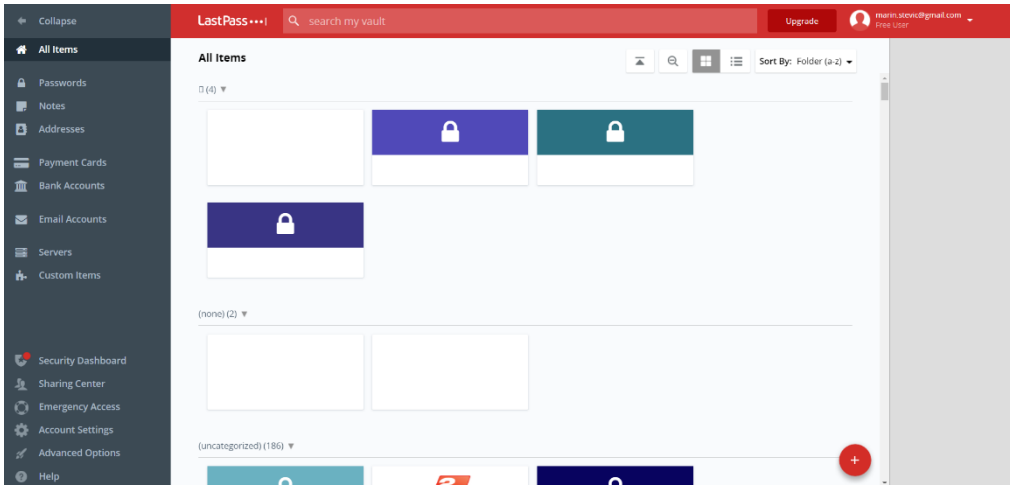
Zadatak ovog diplomskog rada je izraditi USB uređaj za generiranje kriptiranih lozinki. ESP32-S2 uređaj se treba koristiti za generiranje kriptirane lozinke korištenjem podataka primljenih s klijentskog uređaja putem sigurne bežične veze.

2. PREGLED POSTOJEĆIH RJEŠENJA

Postoje mnogobrojna programska i sklopovska rješenja generiranja i spremanja lozinki. U nastavku ovog poglavlja ću opisati tri takva rješenja od kojih su prva dva vrlo popularna i razvijena su od strane velikih kompanija, dok je treće rješenje pretežito otvorenog tipa. Kroz ove primjere ću također pokušati prikazati njihove prednosti i nedostatke koje su dovele do ideje za ovaj rad.

2.1. LastPass

LastPass je jedan od najpopularnijih programa za spremanje i generiranje lozinki za računala i mobilne uređaje. Ima podršku za sve veće operativne sustave: Windows, macOS, Linux, Android i iOS. Na računalima se koristi putem proširenja za web preglednik i automatski prepoznaje polja za korisničko ime i lozinku te ima mogućnost automatskog popunjavanja tih polja. U slučaju registracije nudi mogućnost generiranja nasumične lozinke koja se potom sprema u bazu spremljenih lozinki. Spomenuta baza je zaštićena pomoću glavne lozinke koja je poznata samo korisniku, u slučaju gubitka te lozinke korisnik gubi pristup cijeloj bazi. Baza se sinkronizira na svakom uređaju gdje je instalirana LastPass aplikacija i sve operacije enkripcije i dekripcije se odvijaju na korisničkoj strani. Za enkripciju baze LastPass koristi AES-256 enkripciju sa PBKDF2 SHA256 i posoljenim hashevima [1].



Sl. 2.1. Izgled LastPass sučelja.

LastPass je u operaciji od 2008. godine i od tada je imao mnogobrojne javne proboje sigurnosti. Najveći problem kod ovakvog servisa je online komponenta, iako su sve lozinke kriptirane dostupnost tih lozinki putem interneta predstavlja ogromnu slabu točku. Također

enkripcija cijele baze se vrši pomoću jedne lozinke koja se unosi unutar aplikacije i u slučaju napada ili proboja sigurnosti napadač može dobiti pristup cijeloj korisničkoj bazi.

2.2. YubiKey

YubiKey je uređaj proizveden od strane Yubico tvrtke i služi za zaštitu pristupa računalima, mrežama i mrežnim servisima koji podržavaju sljedeće standarde: jednokratne lozinke, kriptografiju i autentifikaciju pomoću javnog ključa, te U2F (engl. *Universal 2nd Factor*) i FIDO2 protokole. Uređaj omogućuje korisnicima pristup navedenim servisima serviranjem jednokratne lozinke ili javnog ključa generiranog na uređaju putem USB veze [2].

Sl. 2.2. Izgled YubiKey uređaja.

Problem kod dvofaktorske autentifikacije je što ga koristi jako mali skup poslužitelja koje normalni korisnik upotrebljava, iz tog razloga YubiKey podržava i spremanje statičkih lozinki. Ovaj tip lozinke se generira na način da korisnik sam unese početni dio šifre i potom uređaj nasumično dodaje slova iz ograničenog skupa [3]. Ovaj način spremanja lozinke nije pogodan jer ograničen skup slova rezultira manjem broju mogućih kombinacija. Također lozinke se spremaju direktno na samom uređaju što može predstavljati jednu vrstu ranjivosti u slučaju gubitka uređaja.

2.3. Snopf

Snopf je jednostavan ali učinkovit USB alat, sličan prethodno spomenutom YubiKey-u, koji služi za generiranje sigurnih i unikatnih lozinki. Uređaj generira lozinke tako da mu se putem USB veze i korisničkog sučelja pošalju određeni podaci koje on spaja sa 256-bitnom tajnom spremljenom u memoriji. Potom se taj skup podataka hashira i rezultantni niz nakon korištenja BASE64 kodiranja predstavlja generiranu šifru [4].

Sl. 2.3. Izgled Snopf uređaja.

Za razliku od YubiKey uređaja, Snopf se više oslanja na generiranje složenih i statičkih lozinki umjesto korištenja tehnologija privremenih lozinki ili javnih ključeva iz tog razloga je puno bolje rješenje jer pokriva veći broj slučajeva. Pozitivna strana ovog uređaja nasuprot raznih online servisa poput LastPass-a je što se kod njih sve lozinke spremaju na udaljenim serverima, dok korištenjem Snopf uređaja lozinka se uvijek generira lokalno na korisnikov zahtjev. Negativna strana ovog uređaja je mogućnost komunikacije s uređajem jedino putem USB veze i posebnog programa dostupnog samo za Linux operacijski sustav.

2.4. Idejno rješenje ovog rada

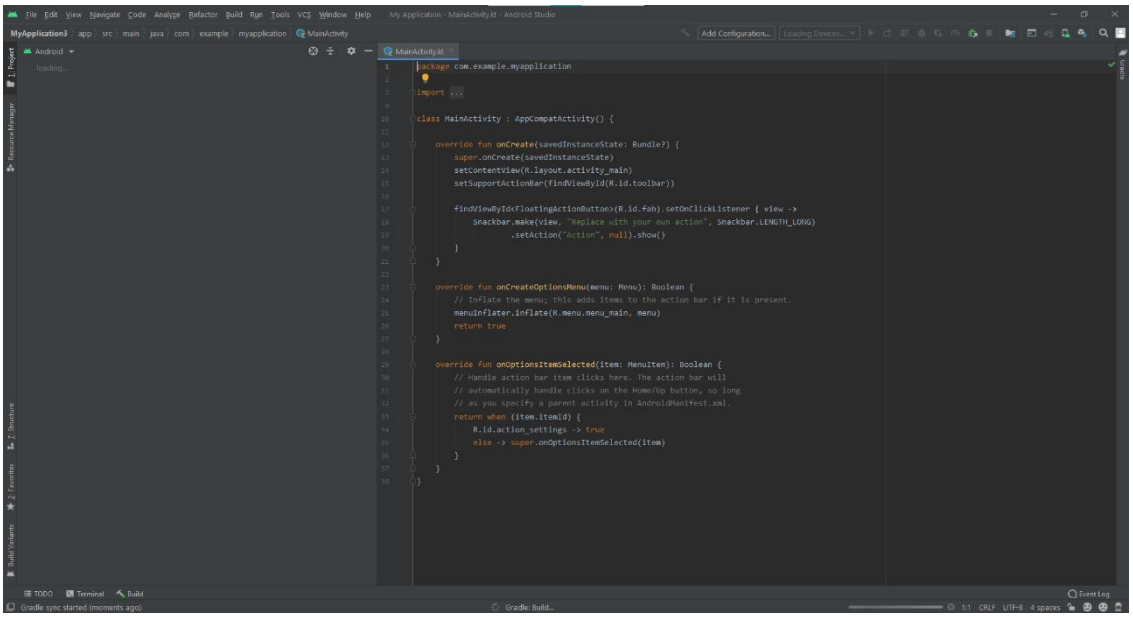
Analiza prethodnih rješenja je potaknula ideju za stvaranjem sustava koji bi mogao naslijediti sve njihove pozitivne karakteristike i otkloniti negativne. Sustav bi se sastojao od posebnog dizajniranog sklopovlja i korisničkog sučelja. Rad sustava bi prvenstveno trebao biti orijentiran na sigurnost ali u isto vrijeme jednostavan i brz za korištenje. Niti jedan sustav se ne može smatrati potpuno sigurnim, svakom računalnom sustavu je najčešće najveća mana ljudski element.

3. KORIŠTENI PROGRAMSKI ALATI I TEHNOLOGIJE

Sustav opisan u ovom radu može se podijeliti na dva dijela: klijentsku i poslužiteljsku stranu. Klijentska strana bi obuhvaćala sučelje koje se prikazuje korisniku u obliku Android aplikacije izrađene pomoću Android Studio razvojnog okruženja. Dok bi poslužiteljska strana predstavljala uređaj na koji se korisnik spaja pomoću navedene aplikacije i izvršava proces generiranja lozinke. Uređaj je posebno dizajniran sklop napravljen oko ESP32-S2 mikroupravljača. Komunikacija između aplikacije i sklopovlja se vrši putem bežične veze korištenjem HTTPS protokola.

3.1. Android Studio

Android Studio je službeno integrirano razvojno okruženje za razvoj aplikacija za mobilni operativni sustav Android čije sučelje možemo vidjeti na slici 3.1. Dostupan je za Windows, macOS i Linux operativne sustave.



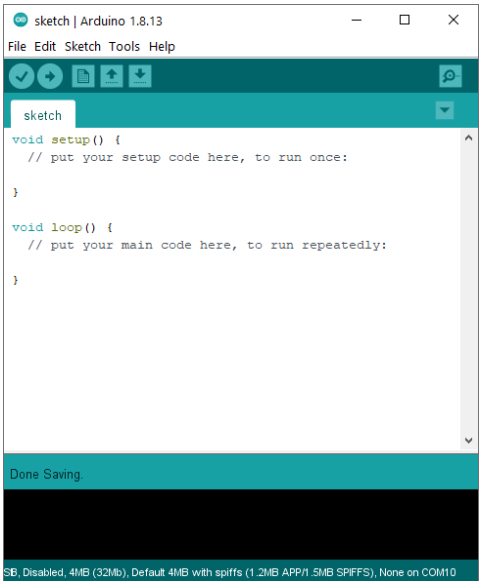
Sl. 3.1. Izgled Android Studio razvojnog okruženja.

Objavljen je 2013. godine na Google I/O konferenciji, te je prva stabilna verzija izašla u 2014. godini. 2019. godine Kotlin zamjenjuje Java programski jezik kao Google-ov preferirani jezik za razvoj Android mobilnih aplikacija unutar Android Studio-a [5]. Kao i većina drugih modernih razvojnih okruženja Android Studio ima mogućnosti automatskog prepoznavanja i ispravljanja grešaka, te popunjavanja dijelova koda što uvelike olakšava i ubrzava posao razvoja

aplikacija. Android Studio također podržava razvoj aplikacija ne samo za mobilne uređaje već i za pametne satove, televizore, tablete i druge pametne uređaje.

3.2. Arduino

Arduino je naziv za platformu otvorenog tipa koja služi za programiranje Arduino kompatibilnih uređaja. Stvorena je od strane talijanske tvrtke 2005. godine kao alat namijenjen za studente, koristeći jeftine 8-bitne mikroupravljače. Kroz godine Arduino platforma je postala jedna od najkorištenijih platformi u svijetu elektronike i ugradbenih sustava. Zbog svoje dokumentiranosti i široke podrške može se upotrebljavati za razne projekte, od svakodnevnih uređaja do kompleksnih znanstvenih instrumenata. Jedna od glavnih sastavnica ove platforme je njena zajednica koja svakim danom raste i proširuje njene mogućnosti [6].



Sl. 3.2. Izgled Arduino razvojnog okruženja.

Za programiranje Arduino kompatibilnih uređaja, odnosno mikroupravljača, razvijeno je Arduino integrirano razvojno okruženje (Arduino IDE) prikazano na slici 3.2. Arduino programski jezik korišten unutar Arduino IDE-a baziran je na AVR C programskom jeziku s određenim prethodno ugrađenim funkcijama za upravljanje sklopovljem. Također ga je moguće proširiti s raznim C++ bibliotekama.

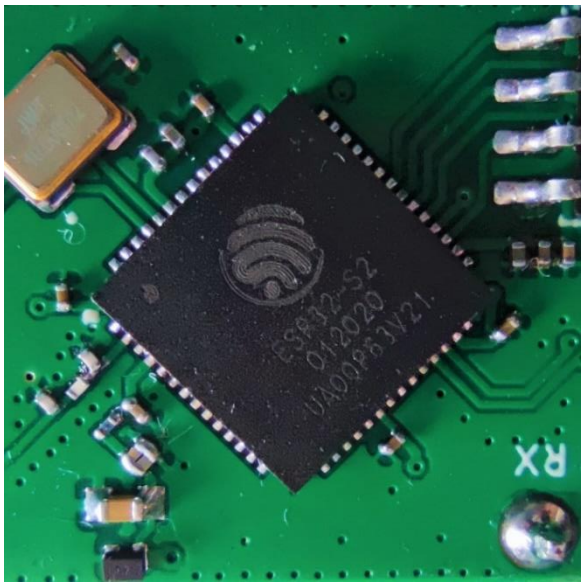
3.1. Autodesk Eagle

Autodesk Eagle je jedan od najpopularnijih programa za izradu shematskih prikaza i dizajniranja tiskanih pločica. Nastao je 1988. godine u Njemačkoj od strane tvrtke *CadSoft*

Computer GmbH kao program za DOS operacijski sustav. Prva verzija programa je omogućavala samo raspodjelu komponenti na tiskanim pločicama, tek s pojavom verzije 2.0 je omogućeno crtanje shematskih prikaza. 2016. godine Autocad najavljuje kupnju tvrtke *CadSoft Computer GmbH* te s uvodom verzije 8.0.0, u 2017. godini, uvodi pretplatnički tip korištenja programa. Postao je popularan upravo zbog svoje jednostavnosti, podrške zajednice i najvažnije besplatnog korištenja za studente [8]. Danas je Eagle kao samostalni program prestao s dobivanjem novih verzija i dolazi integriran unutar Autodesk Fusion360 programa koji služi za izradu i uređivanje 3D modela. Trenutno je Eagle dostupan za sve veće računalne operacijske sustave uključujući: Windows, macOS i Linux.

3.2. ESP32-S2

ESP32-S2 je visoko integrirani, jednoprocesorski Wi-Fi mikroupravljač male snage, dizajniran da bude siguran i isplativ, s visokim performansama i bogatim skupom ulaznih i izlaznih jedinica. Dolazi iz linije vrlo popularnih ESP32 mikroupravljača, nasljednika jednako popularnih ESP8266 mikroupravljača od tvrtke Espressif. ESP linija mikroupravljača je postala popularna prvenstveno zbog svoje cijene i mogućnosti komunikacije i spajanja na internet putem WiFi-a. ESP32-S2 je korišten u ovome radu upravo zbog toga što predstavlja iteraciju ESP32 mikroupravljača posebno fokusiranog na sigurnost sa značajkama poput: sigurnog pokretanja zasnovanog na RSA-3072, AES-XTS-256 enkripciji flash memorije, zaštite privatnog ključa i tajne uređaja od pristupa softvera, kriptografskih akceleratora za poboljšanje performansi i zaštite od fizičkih napada ubrizgavanjem grešaka [7].



Sl. 3.3. Izgled ESP32-S2 mikroupravljača.

3.3. HTTPS

HTTPS (engl. *HyperText Transfer Protocol Secure*) predstavlja proširenje standardnog HTTP protokola. Kod HTTPS-a komunikacijski protokol je kriptiran pomoću SSL/TLS protokola, stoga se nekad zove HTTP putem TLS-a ili HTTP putem SSL-a. Osnovna primjena HTTPS protokola je ostvarivanje sigurne komunikacije i razmjene podataka putem interneta. Štiti od napada poput čovjek u sredini (engl. *man-in-the-middle*) i zbog enkripcije u oba smjera onemogućava prisluškivanje ili čitanje toka podataka. Zbog toga što HTTPS protokol zahtjeva digitalni certifikat potpisan od povjerljive treće strane, što je povijesno bio skup poduhvat, koristio se samo u slučajevima gdje je komunikacija trebala biti strogo zaštićena poput servisa gdje se obavljaju novčane transakcije ili velikim tvrtkama. U današnje vrijeme HTTPS je postao općeprihvaćen i koristi se za veliku većinu javno dostupnih stranica na internetu [9].

4. REALIZACIJA SUSTAVA

Za realizaciju sustava zadanog zadatkom ovog rada potrebno je izraditi posebno dizajniran sklop baziran na ESP32-S2 mikroupravljaču i programski kod koji će obavljati sve funkcije potrebne da bi se uspješno generirala sigurna lozinka.

U sljedećim poglavljima će biti objašnjen cjelokupni proces izrade navedenog uređaja od definiranja svih funkcionalnosti do izrade i programiranja prvog prototipa.

4.1. Model sustava

Ovo potpoglavlje će dati detaljan pregled svih funkcionalnosti koje ovaj sustav treba obavljati. Dijagram koji prikazuje sve dijelove sustava i komunikaciju među njima je prikazan na slici 4.1.

Način rada sustava se može podijeliti u sljedećih pet cjelina:

- Unos podataka
- Slanje podataka
- Primanje i obrada podataka
- Generiranje lozinke
- Unos lozinke

4.1.1. Unos podataka

Unos podatak podrazumijeva postojanje korisničkog sučelja za unos podataka o e-računu izabranog od strane korisnika. Za generaciju sigurne i unikatne lozinke za pojedini e-račun od korisnika su potrebni sljedeći podaci:

1. Ime poslužitelja (engl. *hostname*)
2. Korisničko ime (engl. *username*)
3. Duljina lozinke
4. Iteracija
5. Glavni pin

Ime poslužitelja predstavlja oznaku kojom će korisnik moći raspoznati različite servise poput: Facebook, Gmail, Twitter, itd. Korisničko ime predstavlja ime/nadimak koji je korišten pri registraciji e-računa na određenom poslužitelju. Ime poslužitelja i korisničko ime predstavljaju osnovne podatke pomoću kojih će korisnik moći generirati lozinke za različite e-račune. Duljina

lozinke, kao što samo ime govori, predstavlja broj/količinu slova, brojki i drugih znakova koji tvore lozinku. U slučaju gubitka ili krađe lozinke jednostavnim povećanjem vrijednosti iteracije možemo dobiti potpuno novu i sigurnu lozinku. Glavni pin predstavlja tajni ključ koji korisnik može, ali ne mora, unijeti za dodatno osiguranje svojih lozinki. Glavni pin uz tajni ključ spremljen na uređaju predstavljaju dva komada informacije koje korisnik mora strogo čuvati jer bez njih je nemoguće ponovno generirati jednake lozinke.

4.1.2. Slanje podataka

Nakon što je korisnik unio sve potrebne podatke, sljedeći korak je dostaviti te podatke na uređaj. Kao što je prethodno spomenuto ovaj uređaj je dizajniran oko ESP32-S2 mikroupravljač upravo zbog svoje mogućnosti komuniciranja putem WiFi-a. Korisnička aplikacija koja služi prikupljanju prethodno navedenih podataka mora imati mogućnost spajanja na WiFi mrežu.

Pošto se radi o informacijama koje nisu kriptirane jer želimo imati apstrakciju korisničkog sučelja od uređaja s razlogom da se omogući korištenje drugih sučelja bez prethodne razmjene informacija odnosno postavljanja privatnih ključeva. Ovim postupkom smanjujemo mogućnost spajanja uređaja, s razlogom razmjene ključeva, u nesiguran sustav i mogućnosti komprimiranja samog uređaja. Iz tog razloga za razmjenu podataka koristimo postojeći i dobro razvijeni HTTPS protokol koji omogućava siguran prijenos podataka bez mogućnosti njihovog očitavanja ili promjene.

4.1.3. Primanje i obrada podataka

Primanje i obrada podataka se vrše na strani uređaja. Zbog visoke procesorske snage odabranog mikroupravljača on ima mogućnosti samostalnog pokretanja i održavanja privatnog HTTPS web server. Također uređaj može raditi u AP (engl. *access point*) načinu rada gdje stvara svoju osobnu privatnu WiFi pristupnu točku na koju se klijent potom može spojiti. Nakon spajanja klijenta s uređajem putem WiFi-a, korisnička aplikacija dobiva pristup spajanju na lokalni HTTPS server na koji šalje sve prethodno unesene podatke putem sigurne i kriptirane veze.

Podaci na uređaj pristižu u JSON (engl. *JavaScript Object Notation*) obliku. JSON je tekstualni format dizajniran za čitljiv i razumljiv prijenos strukture podataka ljudima i strojevima. Svi podaci unutar njega se spremaju u obliku parova ključa i vrijednosti (engl. *key-value*). Nakon što podatci stignu na uređaj vrlo je lako izdvojiti podatke potrebne za generiranje lozinke i podatke koji opisuju pravila za generiranja.

4.1.4. Generiranje lozinke

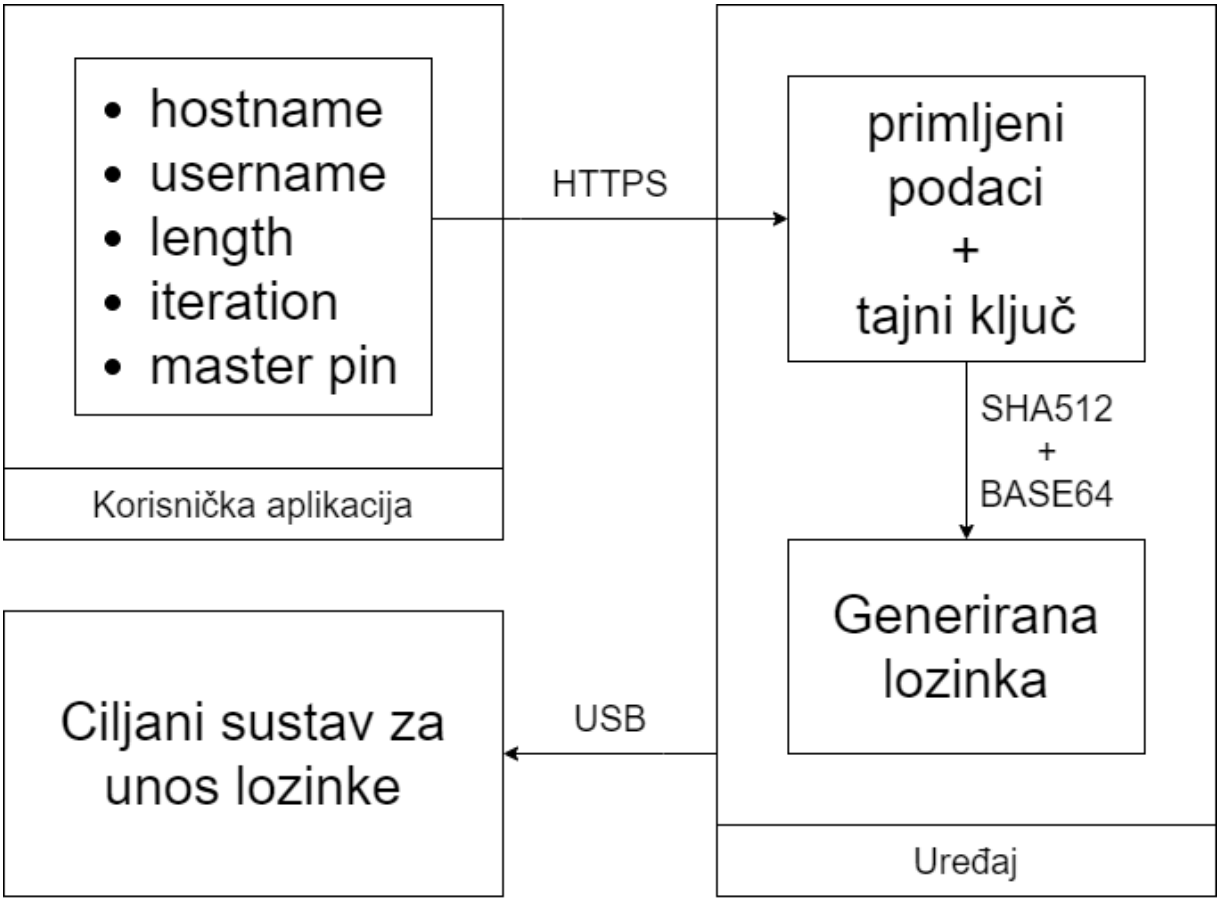
Za generiranje lozinke osim podataka primljenih putem korisničkog sučelja, potreban je i sigurnosni ključ spremljen u memoriji uređaja. Sigurnosni ključ predstavlja 256-bitni jedinstveni niz predstavljen pomoću 32 znaka koje korisnik mora unijeti prilikom programiranja uređaja. Ovaj skup znakova i glavni pin korisnik bi trebao zapamtiti ili zapisati na sigurno mjesto jer bez njih korisnik gubi mogućnost ponovnog generiranja identičnih lozinki. Također u slučaju gubitka uređaja ako korisnik ima pristup sigurnosnom ključu vrlo lako može izraditi novi uređaj.

Nakon što su svi podaci pripremljeni, uređaj spaja sljedeće podatke u jedan dugačak niz: ime poslužitelja, korisničko ime, glavni pin, iteraciju i sigurnosni ključ. Uređaj potom uzima taj niz i primjenjuje kriptografsku hash funkciju SHA512 nad tim podacima. Hash funkcija je bilo koja funkcija koja može primiti podatke proizvoljne veličine, a kao rezultat dati nit bitova fiksne veličine. Veličina izlaza uvjetovana je funkcijom koja se koristi. Izlaz hash funkcije će uvijek biti isti za isti ulaz, ako se promijeni i najmanji dio ulaznih podataka izlaz će biti potpuno različit. Ove funkcije se još nazivaju i jednosmjernima jer se iz izlaza ne može saznati ulaz (barem ne s trenutnom procesorskom snagom). Neke od poznatijih i najčešće korištenih hash funkcija su: SHA256, SHA512 i MD5. SHA512 funkcija koja je korištena na uređaju stvara izlazni niz duljine 512-bita. Za generiranje lozinke na uređaju je korištena upravo ova hash funkcija zbog toga što je prvenstveno jednosmjerna, ako netko sazna jednu od lozinki ne može doći do ulaznih podataka niti generirati druge lozinke. Drugo, jer s istim ulazom uvijek dobivamo isti izlaz što znači da lozinke ne moramo spremati nego ih uvijek možemo iznova generirati u stvarnom vremenu. I posljednje, da bi generirali lozinku moramo izlazni niz od 512-bita pretvoriti u niz znakova koji se smiju koristiti za lozinku, što znači da ako uzmemo skup od 64 znaka svaki znak se može opisati sa 7-bita. Ako uzmemo 512-bita i podijelimo na 7-bitne odjeljke dobiti ćemo maksimalnu duljinu lozinke koju možemo generirati i koja iznosi 73 znaka. Taj proces se naziva BASE64 kodiranje, ali u ovome slučaju korišten je prilagođen skup znakova koji se mogu naći na što više različitih rasporeda tipkovnice.

4.1.5. Unos lozinke

ESP32-S2 osim što ima mogućnost komuniciranja putem WiFi-a može koristiti i USB vezu za komuniciranje putem OTG (engl. *on-the-go*) standarda. USB OTG omogućuje ovom mikroupravljaču da se drugim priključnim uređajima predstavi kao USB *host* ili običan uređaj i da mijenja ulogu u bilo kojem trenutku. Ova implementacija je olakšala razvoj ovoga uređaja jer je omogućila direktno spajanje podatkovnih USB veza na pinove mikroupravljača bez potrebe za

dodatnom periferijom. Pri spajanju uređaja na računalo ili drugi sustav koji omogućuje USB vezu, uređaj se predstavlja kao HID (engl. *human interface device*) i može imitirati unos poput tipkovnice. Nakon što je završilo generiranje hash-a ulaznih podataka i sigurnosnog ključa, uzimamo generiranu lozinku i potom jedan po jedan znak šaljemo USB vezom pritom imitirajući tipkovnicu sve dok ne pošaljemo broj znakova određen duljinom lozinke.



Sl. 4.1. Pojednostavljeni blokovski prikaz modela sustava.

4.2. Sklopovsko rješenje

U ovome potpoglavlju opisat će se postupak realizacije sklopovskog rješenja uređaja opisanog zadatkom rada. Za izradu sklopa koristit će se program Autodesk Eagle. Prvi korak pri izradi sklopovskog rješenja za ovaj zadatak je izrada shematskog prikaza. Da bi se izradio shematski prikaz moraju se prvo definirati sve funkcionalnosti koje uređaj mora imati, to su:

- Komuniciranje putem USB veze
- Komuniciranje putem WiFi veze
- Prikaz trenutnog stanja
- Napajanje

Nakon što su određene sve funkcionalnosti potrebne za izradu ovog uređaja može se krenuti na crtanje shematskog prikaza i odabira komponenti. Prvi korak je odabir pakiranja ESP32-S2 mikroupravljača, koji se može naručiti u obliku modula ili samostalnog čipa. Za prvu verziju tiskane pločice korišten je modul zato što se tako privremeno izbjegava problem kompleksnijeg radiofrekvencijskog dizajna jer ovaj mikroupravljač podržava spajanje na WiFi i da bi ta veza bila što bolja treba se pridržavati određenih pravila i smjernica da bi sustav radio najoptimalnije. Za drugu verziju u cilju smanjenja volumena uređaja korišten je samo čip i sve potrebne periferne komponente za osnovni rad. Oba dizajna su opisana u sljedeća dva potpoglavlja.

Crtež shematskog prikaza zadnje verzije uređaja se može pronaći u prilogu 1.

4.2.1. Prvi dizajn

Prvi dizajn uređaja, prikazan na slici 4.2., koristio je ESP32-S2-WROVER modul koji na sebi osim što sadrži ESP32-S2 mikroupravljač i osnovne komponente za normalan i stabilan rad, sadrži i antenu. Antena je dizajnirana pomoću točno oblikovanog traga bakra na površini tiskane pločice modula. Da bi antena mogla normalno funkcionirati i odašiljati signal ispod nje se ne smiju nalaziti veće bakrene površine, stoga je na tom dijelu ostavljena prazna pločica.

Sl. 4.2. Izgled prvog dizajna uređaja.

Za normalno funkcioniranje opisanog modula potrebno je dovesti 3.3V napajanje i spojiti jedan 4.7kΩ *pull-up* otpornik na EN nožicu. Dovođenjem ovog napona postavljamo tu nožicu u visoko logičko stanje i uređaj će se upaliti. Također ovo možemo koristiti za resetiranje uređaja na

način da istu nožicu spojimo preko tipkala na GND, pritiskom tipkala spuštamo nožicu u nisko logičko stanje i uređaj se gasi.

Za izvor napajanja iskorištena je prednost dostupnosti USB sučelja čije je napajanje standardizirano na 5V i pomoću linearnog regulatora smanjujemo napon na traženih 3.3V. Korišteni linearni regulator je AMS1117-3.3 koji dolazi u SOT-223 pakiranju i ima mogućnosti kontinuiranog izlaza 1A struje. Uz njega su postavljena dva 10uF kondenzatora koji otklanjaju nagle poraste u potrošnji modula.

Uz tipkalo za resetiranje uređaja dodano je još jedno tipkalo koje služi za ulazak u način za programiranje uređaja. Jedna strana tipkala je spojena na GND dok je druga spojena na GPIO-0 nožicu modula koja je preko 4.7kΩ otpornika povučena na visoku logičku razinu. Za ulazak u način za programiranje uređaja mora se držati ovo tipkalo prilikom ponovnog pokretanja.

Komunikacija putem USB veze je ugrađena u ESP32-S2 mikroupravljač, stoga za ostvarivanje komunikacije putem USB-a moguće je direktnim spajanjem podatkovnih veza konektora sa modulom. Za signalizaciju o stanju uređaja korištene se dvije bijele LE diode. Također je na uređaj dodan i 2.54mm header s tri nožice koji je omogućavao komunikaciju putem serijske veze za otklanjanje pogrešaka.

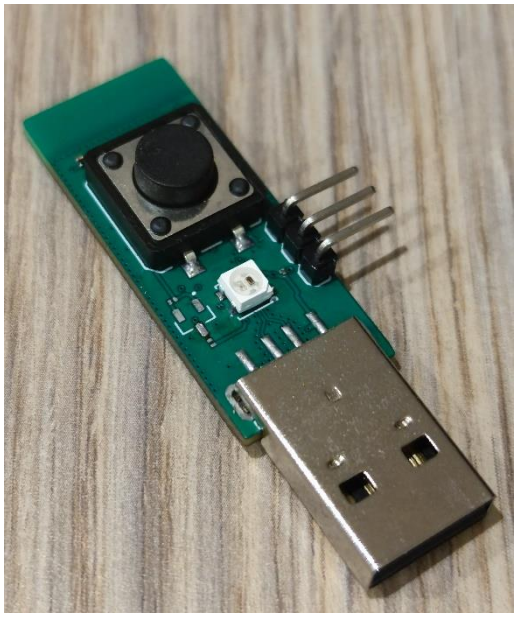
4.2.2. Drugi dizajn

Za drugu verziju dizajna uređaja cilj je bio smanjiti cjelokupni volumen. To se moglo postići samo prelaskom sa korištenja modula na korištenje samostalnog čipa. Razlike između prve i druge verzije su:

- Zamjena modula sa čipom
- 4-slojna pločica
- Manja antena
- Jedno tipkalo
- RGB LE dioda

Na slici 4.3. se može vidjeti izgled druge verzije uređaja. Za ostvarenje ovoga cilja potrebno je bilo preći sa korištenja dvoslojne pločice na korištenje četveroslojne zbog uštede prostora i veličine komponenti. Za razliku od korištenja modula, većina komponenti se morala koristiti i u ovome novom dizajnu, što je podrazumijevalo korištenje jako malih komponenti

veličine i do 0.6x0.3mm. Postavljanje ovakvih komponenti ručno je posao koji se mora izvršavati pomoću posebnih alata, ali se njihovim korištenjem podosta štedi na veličini uređaja.



Sl. 4.3. Izgled drugog dizajna uređaja.

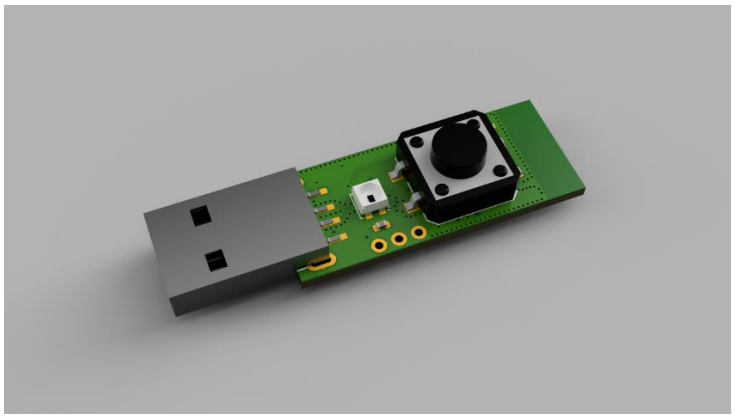
Dizajn tiskane pločice je morao sadržavati 4-sloja ne samo radi lakšeg rutiranja vodova nego i zbog radiofrekvencijskog dijela dizajna, odnosno vodova koji moraju nositi visokofrekvencijske signale i potrebni su im određeni karakteristike dizajna poput: debljine i duljine vodova, udaljenosti između bakra u slojevima i impedancije vodova. Zadnja karakteristika je pogotovo bila važna kod dizajniranja antene uređaja čija je impedancija 50Ω, dok je impedancija na izlazu iz nožice mikroupravljača (34+j5)Ω. Da bi se izjednačile impedancije potrebno je bilo koristiti π-tip mreže koji se u ovome slučaju sastojao od 2 kondenzatora vrijednosti 2.4pF i 3pF, te jedne zavojnice vrijednosti 2.2nH.

Nakon korištenja prve verzije uređaja zaključeno je da pošto se uređaj napaja putem USB konektora nije potrebno tipkalo za resetiranje, jer bi se isti efekt ostvario odvajanjem i ponovnim spajanjem uređaja. Zbog toga je samo ostavljeno jedno tipkalo koje služi za korisničku interakciju, a ulazak u način rada za programiranje je ostvaren putem zadržavanja tipkala prilikom spajanja uređaja.

Umjesto korištenja dvije odvojene LE diode, zamijenjene su sa jednom RGB LE diodom koja ima mogućnost prikazivanja različitih boja koje će lakše i brže signalizirati korisniku trenutno stanje uređaja. Dioda korištena u ovome dizajnu je WS2812B-MINI.

4.2.3. 3D model uređaja

Autodesk Eagle ima mogućnost automatske izrade 3D modela tiskane pločice s postavljenim komponentama. Za ostvarivanje ovog rezultata svaka komponenta korištena u dizajnu tiskane pločice treba imati postavljen 3D model. Većina proizvođača uz svoje komponente izdaje i CAD (engl. *Computer Aided Design*) modele koji se potom mogu uvesti u Eagle u raznim formatima poput: *stl, obj, step, itd.* Mogućnost izrade 3D modela može biti od velike pomoći pri izradi kućišta ili određenih analiza poput analize raspršivanja topline uređaja.



Sl. 4.4. 3D model uređaja izrađen u Fusion360 programu.

4.3. Programsko rješenje

Programsko rješenje sklopovlja uz njegov dizajn predstavlja integralni dio ovog rada. Za pisanje programskog koda korišteno je Arduino razvojno okruženje uz određene otvorene biblioteke i proširenja za rad s ESP32 obitelji mikroupravljača. Po osnovi svaki kod napisan unutar Arduino razvojnog okruženja sadrži dvije osnovne funkcije: *setup()* i *loop()*. Kod napisan unutar *setup* funkcije se izvršava samo jednom i to prilikom prvog pokretanja uređaja, dok se kod pisan unutar funkcije *loop* pokreće tek nakon izvršenja *setup* funkcije i vrti se beskonačno mnogo puta, odnosno do prestanka rada uređaja.

Funkcionalnosti opisane modelom sustava koje je potrebno implementirati u kodu su:

- Pokretanje WiFi pristupne točke
- Pokretanje HTTPS servera i generiranje certifikata
- Upravljanje RGB LE diodom
- Primanje podataka
- Hashiranje
- BASE64 kodiranje i slanje podataka putem USB veze

U sljedećim potpoglavljima će detaljno biti objašnjeni koraci i postupci koje korisnik mora proći za uspješno postavljanje i upotrebu uređaja. Koraci pretpostavljaju da korisnik ima pristup:

- uređaju opisanom ovim radom
- računalu s instaliranom najnovijom verzijom Arduino IDE-a

Programski kod se može u cijelosti preuzeti s GitHub repozitorija čija se poveznica nalazi u zaključku ovog rada.

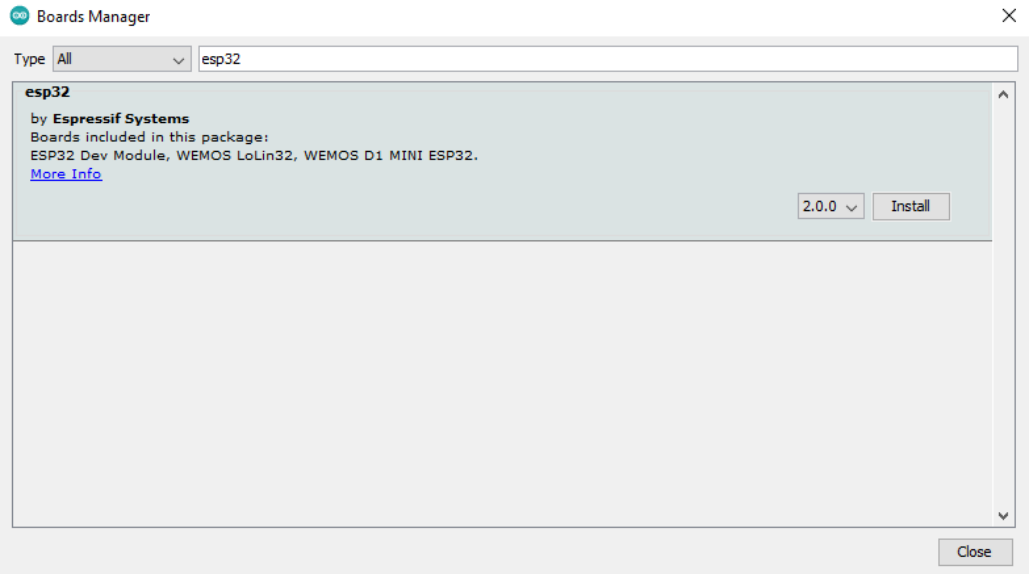
4.3.1. Priprema Arduino razvojnog okruženja

Prije početka programiranja potrebno je obaviti određene korake unutar Arduino razvojnog okruženja da bi imali mogućnost korištenja ESP32-S2 mikroupravljača i svih njegovih funkcija. Prvi korak je instalacija podrške za ESP32 mikroupravljače:

1. Otvoriti Arduino IDE
2. Unutar programa otići pod *File -> Preferences*
3. Pod *Additional Boards Manager URLs* dodati sljedeći link:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

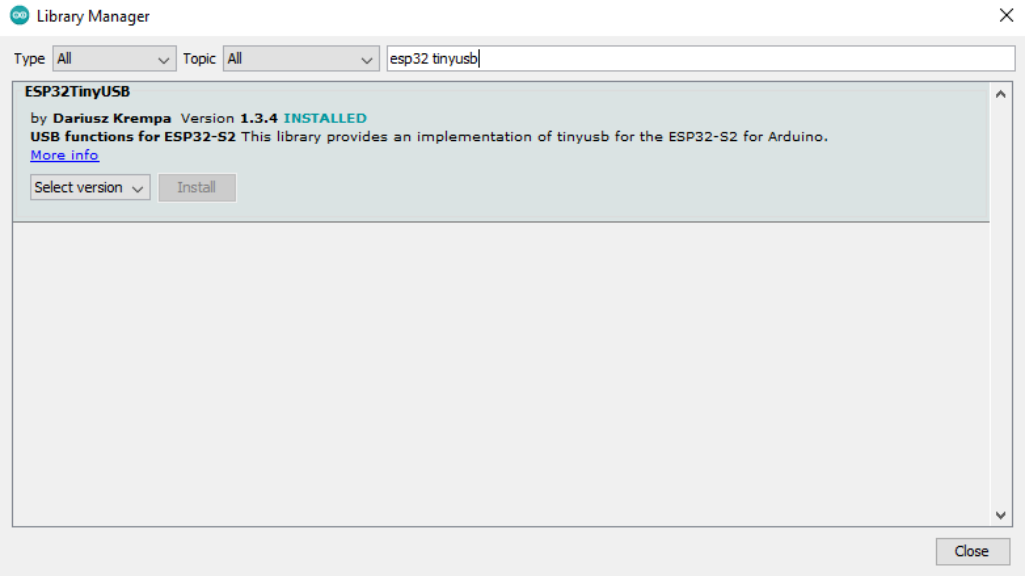
4. Nakon dodavanja linka zatvoriti otvorene prozore
5. Otići u *Boards Manager* koji se nalazi unutar *Tools -> Board*
6. U tražilicu unijeti esp32 i kliknuti *Install* (Slika 4.5)
7. Ponovno pokrenuti Arduino IDE



Sl. 4.5. Prikaz *Boards Manager* ekrana i *esp32* paketa koji treba instalirati.

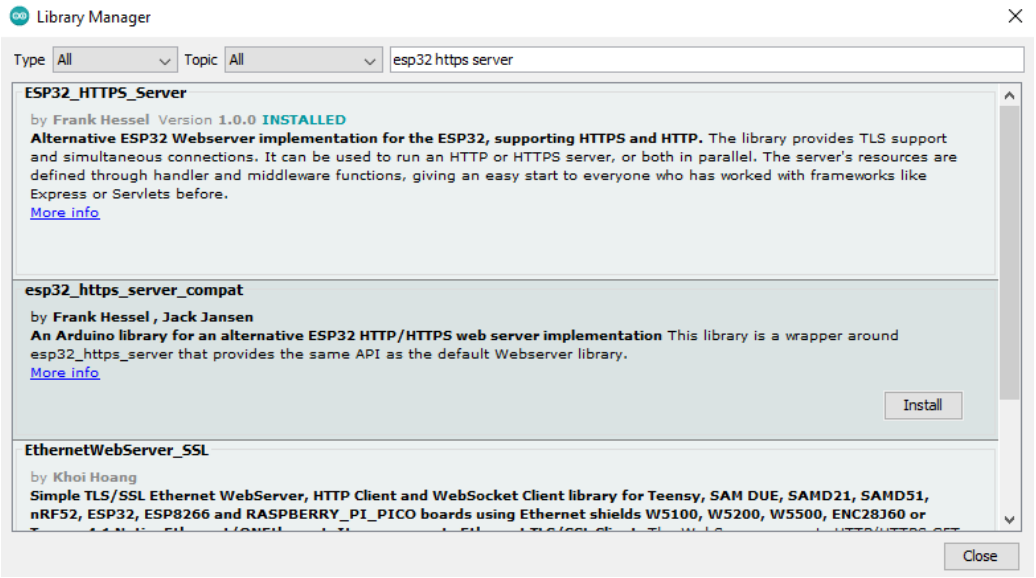
Nakon što je uspješno dodana podrška za ESP32 mikroupravljače potrebno je dodati i određene biblioteke koje pomažu za komunikaciju putem USB veze i upravljanjem HTTPS servera:

1. Unutar Arduino IDE-a otići na *Sketch ->Include Library -> Manage Libraries*
2. U tražilicu unijeti: „*esp32 tinyusb*“ i instalirati paket prikazan na slici 4.6



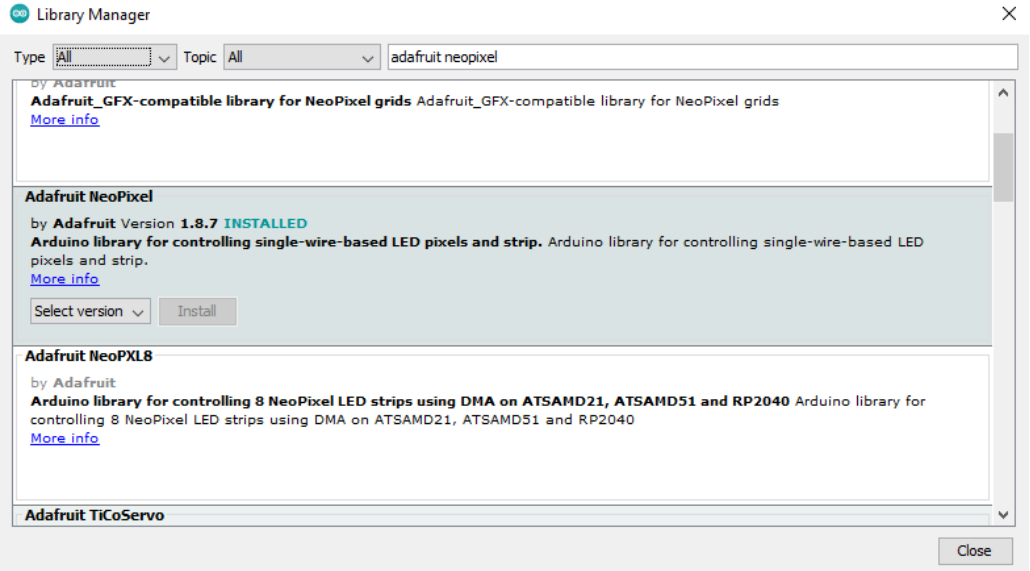
Sl. 4.6. Prikaz *Library Manager* ekrana i *esp32 tinyusb* paketa koji treba instalirati.

3. Ponovno kliknuti na tražilicu i upisati „*esp32 https server*“ i instalirati paket prikazan na slici 4.7. koji pored sebe ima oznaku *INSTALLED*



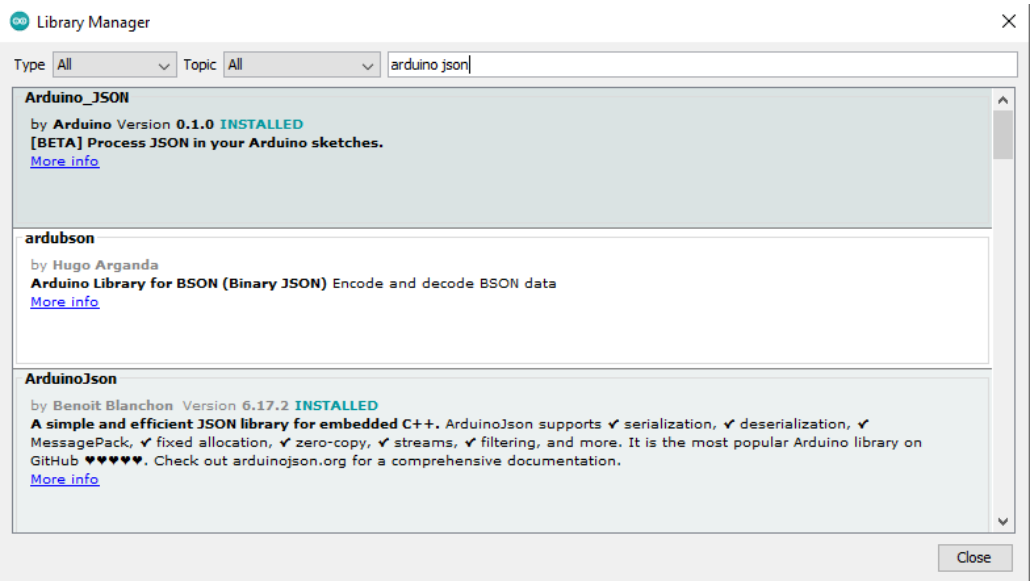
Sl. 4.7. Prikaz *Boards Manager* ekrana i *esp32 https server* paketa koji treba instalirati.

4. Ponovno kliknuti na tražilicu i upisati „*adafruit neopixel*“ i instalirati paket prikazan na slici 4.8. koji pored sebe ima oznaku *INSTALLED*



Sl. 4.8. Prikaz *Boards Manager* ekrana i *esp32 https server* paketa koji treba instalirati.

5. Ponovno kliknuti na tražilicu i upisati „*arduino json*“ i instalirati paket prikazan na slici 4.9. koji pored sebe ima oznaku *INSTALLED*



Sl. 4.9. Prikaz *Boards Manager* ekrana i *esp32 https server* paketa koji treba instalirati.

6. Zatvoriti *Library Manager*

Nakon završetka prethodnih koraka možemo krenuti na objašnjavanje napisanog koda.

4.3.2. Pokretanje WiFi pristupne točke

Prvi korak za pokretanje uređaja u AP načinu rada i odašiljanja privatne WiFi pristupne točke je dodavanje biblioteke *WiFi.h* (Slika 4.10.) koja omogućuje pristup tim funkcijama.

```
#include <WiFi.h>
```

Sl. 4.10. Dodavanje WiFi.h biblioteke.

Potom moramo definirati ime i lozinku AP-a uređaja što je prikazano kodom na slici 4.11.

```
const char* ssid = "PassVault";
const char* password = "123456789";
```

Sl. 4.11. Deklaracija imena i lozinka AP-a.

Nakon toga možemo na početku *setup* funkcije pokrenuti AP način rada pomoću naredbe *softAP* (Slika 4.12.) koja prima dva argumenta: *ssid* i *password* koji su prethodno definirani i postavljeni na svoje odgovarajuće vrijednosti. *SoftAP* funkcija može primiti više od ova dva osnovna argumenta, neki od drugih su: WiFi kanal, vidljivost SSID-a i maksimalan broj klijenata koji se može spojiti.

```
WiFi.softAP(ssid, password);
```

Sl. 4.12. Pokretanje WiFi AP-a na uređaju.

4.3.3. Pokretanje HTTPS servera i generiranje certifikata

Nakon što smo uspješno pokrenuli WiFi pristupnu točku i omogućili korisniku da se spoji na uređaj moramo pokrenuti HTTPS web server na koji će se slati korisnički podaci. Prvi korak je dodavanje potrebnih biblioteka (Slika 4.13.). Funkcionalnost ovih biblioteka i što se nalazi unutar njih neće biti ovdje objašnjene ali o njima se može više pročitati ovdje [10].

```
#include <HTTPSServer.hpp>
#include <SSLCert.hpp>
#include <HTTPRequest.hpp>
#include <HTTPResponse.hpp>
```

Sl. 4.13. Dodavanje biblioteka potrebnih za pokretanje HTTPS servera i generiranje certifikata

HTTPS server dolazi u drugom *namespace-u* stoga ga je dobro dodati ovdje za lakše korištenje (Slika 4.14.).

```
using namespace httpsserver;
```

Sl. 4.14. Deklaracija HTTPS server *namespace-a.*

ESP32 ima mogućnost generiranja certifikata na uređaju. To olakšava postupak namještanja uređaja jer ne moramo koristiti druge alate poput OpenSSL-a za generiranje potrebnih certifikata. Sigurnost generiranih certifikata nam nije problematičan jer se korisnik spaja direktno na uređaj koji ujedno radi kao i poslužitelj i sva komunikacija je jednosmjerna, odnosno usmjerena od korisnika prema uređaju. Na slici 4.15. možemo vidjeti kod koji služi za kreiranje objekta klase SSLCert koji će sadržavati informacije o certifikatu i objekta klase HTTPSServer koji će služiti za namještanje sigurnog servera.

```
SSLCert * cert;
HTTPSServer * secureServer;
```

Sl. 4.15. Deklaracija pokazivača za objekte klasa SSLCert i HTTPSServer.

Na idućoj slici 4.16. je prikazano instanciranje objekta klase SSLCert.

```
cert = new SSLCert();
```

Sl. 4.16. Instanciranje objekta klase SSLCert.

Nakon instanciranja objekta klase SSLCert možemo generirati certifikat. To ćemo učiniti pozivom funkcije *createSelfSignedCert* (Slika 4.17.) koja prima sljedeće parametre: referencu na naš SSLCert objekt, veličinu ključa i niz znakova koji predstavljaju naziv certifikata po x509 specifikacijama. Također ćemo provjeravati je li generacija certifikata prošla uspješno, tako što ćemo uspoređivati povratnu vrijednost funkcije koja treba biti 0.

```
int createCertResult = createSelfSignedCert(
    *cert,
    KEYSIZE_2048,
    "CN=myesp.local,O=acme,C=US");

if (createCertResult != 0) {
    Serial.printf("Error generating certificate");
    return;
}
```

Sl. 4.17. Primjer generiranja certifikata.

Nakon što smo uspješne generirali certifikat možemo ga poslati kao argument konstruktoru klase HTTPSServer da bi dobili njegov objekt (Slika 4.18.).

```
secureServer = new HTTPSServer(cert);
```

Sl. 4.18. Instanciranje objekta klase HTTPSServer s ranije generiranim certifikatom.

Nakon što smo uspješno kreirali HTTPSServer objekt moramo definirati rutu našeg servera. Zbog toga što šaljemo malu količinu podataka i samo za jednu upotrebu koristit ćemo

samo jednu rutu. Na slici 4.19. je prikazano postavljanje rute („/“) i njene registracije, također odmah poslije toga pokrećemo server i čekamo pomoću *while* petlje dok server ne bude dostupan.

```
ResourceNode* nodeEchoPost = new ResourceNode("/", "POST", &handleEcho);
secureServer.registerNode(nodeEchoPost);
secureServer.start();
while (!secureServer.isRunning()) {}
```

Sl. 4.19. Primjer registracije rute i pokretanja sigurnog servera.

Zadnja naredba koju moramo pokrenuti se pokreće unutar *loop* funkcije i ona služi za omogućavanje servera da prima podatke (Slika 4.20.).

```
secureServer.loop();
```

Sl. 4.20. Naredba kojom server ostaje aktivan.

4.3.4. Upravljanje RGB LE diodom

Za signalizaciju korisniku o trenutnom stanju uređaja koristi se jedna RGB LE dioda. Za korištenje ove LE diode potrebno je dodati biblioteku od kompanije Adafruit pod nazivom NeoPixel. Također moramo definirati objekt klase Adafruit_Neopixel koji prima 3 parametra: broj LE dioda, izlaz mikroupravljača na koji je spojena i LE diode (Slika 4.21.).

```
#include <Adafruit_NeoPixel.h>

Adafruit_NeoPixel pixels(1, 36, NEO_GRB + NEO_KHZ800);
```

Sl. 4.21. Dodavanje Adafruit_Neopixel biblioteke i deklaracija *pixels* objekta.

Na slici 4.22. je prikazan poziv funkcije *begin* na objektu *pixels* te nakon njega pozivamo funkciju *setBrightness* koja služi za postavljanje jačine svjetlosti LE diode, jačina je određena brojem od 0-255.

```
pixels.begin();
pixels.setBrightness(16);
```

Sl. 4.22. Primjer pokretanja RGB LE diode i postavljanja jačine svjetlosti.

Za prikaz određene boje na LE diodi moraju se upotrijebiti dvije funkcije: prva koja odabire boju i druga koja osvježava LE diodu i prikazuje boju. Za lakše korištenje napravljena je pomoćna funkcija *light*, prikazana na slici 4.23., koja služi za izvršavanje obje funkcije odjednom. Funkcija prima 3 argumenta koja predstavljaju vrijednost 3 boje: crvene, zelene i plave. Vrijednosti su u rasponu između 0-255.

```
void light(int red, int green, int blue) {
  pixels.setPixelColor(0, pixels.Color(red, green, blue));
  pixels.show();
}
```

Sl. 4.23. Primjer funkcije koja postavlja boju RGB LE diode.

Uređaj koristi 3 boje za signalizaciju trenutnog stanja:

- Zelena – uređaj je spreman za korištenje
- Plava – uređaj čeka korisničku interakciju
- Crvena – uređaj je u upotrebi

4.3.5. Primanje podataka

Na slici 4.24. je prikazan primjer funkcije *handleEcho* koja se poziva nakon što server primi podatke na prethodno definiranoj ruti. Nakon primitka podataka oni se potom spremaju u varijablu *buffer*. Podaci unutar varijable *buffer* su spremljeni u JSON formatu i moraju se deserijalizirati prije korištenja, odnosno spremiti u format razumljiv mikroupravljaču. Ako se deserijalizacija izvrši uspješno popunjavaju se pojedinačne varijable sa svojim odgovarajućim vrijednostima primljenih od korisnika.

```
void handleEcho(HTTPRequest* req, HTTPResponse* res) {
  res->setHeader("Content-Type", "text/plain");

  byte buffer[1024];
  char payload[1024];
  byte shaResult[64];

  while (!(req->requestComplete())) {
    size_t s = req->readBytes(buffer, 1024);
    res->write(buffer, s);

    [...]

    StaticJsonDocument<1024> doc;
    DeserializationError error = deserializeJson(doc, buffer);

    if (!error) {
      const char* hostname = doc["hostname"];
      const char* account = doc["account"];
      bool button = doc["button"];
      int len = doc["length"];
      const char* iteration = doc["iteration"];
      const char* pin = doc["pin"];
```

```
      const char* layout = doc["layout"];
      [...]
    }
  }
}
```

Sl. 4.24. Primjer *handleEcho* funkcije koja prima podatke poslane na server i primjer spremanja primljenih podataka.

4.3.6. Hashiranje

Da bismo mogli koristiti hash funkcije na primljenim podacima moramo prvo dodati biblioteku md.h (slika 4.25.).

```
#include "mbedtls/md.h"
```

Sl. 4.25. Dodavanje md.h biblioteke.

Prije nego što nastavimo s hashiranjem podataka moramo postaviti tajni ključ (Slika 4.26.) koji ostaje spremljen na uređaju i služi da bi se ulazni podaci mogli „posoliti“. Tajni ključ treba biti duljine 32 znaka odnosno 256-bita.

```
const char* secret = "WWbYy8hLMaCK2RkWCzBapCXNnXRupLZ3";
```

Sl. 4.26. Deklaracija tajnog ključa.

Zbog toga što želimo hashirati sve primljene podatke zajedno sa tajnim ključem, moramo ih prvo spojiti u jedan niz. Varijable iz koda koje će se spojiti i potom hashirati su: hostname, account, pin, interation i secret. One će tvoriti jedinstveni niz iz kojega će se generirati lozinka za korisnikov traženi e-račun. Na slici 4.27. se nalazi navedeni primjer spajanja svih podataka u jednu varijablu *payload* i njihovog hashiranja. Hash će potom biti spremljen u varijabli shaResult.

```
strcpy(payload, hostname);
strcat(payload, account);
strcat(payload, pin);
strcat(payload, iteration);
strcat(payload, secret);

const size_t payloadLength = strlen(payload);

mbedtls_md_init(&ctx);
mbedtls_md_setup(&ctx, mbedtls_md_info_from_type(md_type), 0);
mbedtls_md_starts(&ctx);
mbedtls_md_update(&ctx, (const unsigned char*)payload, payloadLength);
mbedtls_md_finish(&ctx, shaResult);
mbedtls_md_free(&ctx);
```

Sl. 4.27. Primjer spajanja vrijednosti varijabli u jedan niz i njihovo hashiranje.

4.3.1. BASE64 kodiranje i slanje podataka putem USB veze

Za slanje podataka, odnosno generirane lozinke, na sustav gdje ju trebamo upisati koristimo USB sučelje. Za korištenje tog sučelja moramo dodati biblioteke key_layout.h i hidkeyboard.h i deklarirati objekt klase HIDkeyboard koji moramo pokrenuti unutar *setup* funkcije (Slika 4.28.).

```
#include "key_layout.h"
#include "hidkeyboard.h"

HIDkeyboard dev;

void setup() {

  [...]

  dev.begin();


  [...]
}
```

Sl. 4.28. Dodavanje key_layout.h i hidkeyboard.h biblioteke, te deklaracija i pokretanje objekta klase HIDkeyboard.

Nakon što smo uspješno generirali hash svih podataka primljenih na uređaj, moramo izvršiti BASE64 kodiranje tog hasha. Postoje gotove funkcije koje bi mogle obaviti ovo kodiranje ali zbog toga što nam je bitno koje znakove možemo dobiti unutar zaporke, napravljen je poseban algoritam koji će koristiti znakove iz prethodno definiranog polja znakova. Na slici 4.28. je prikazan primjer prethodno navedenog algoritma koji izvršava BASE64 kodiranje. Kodiranje se vrši tako što se iz hasha redom uzima svakih 7-bitova koji predstavljaju poziciju znaka koji je odabran. Nakon što smo odabrali znak odmah ga prosljeđujemo funkciji *sendChar* koja putem USB veze šalje znak imitirajući pritisak tipke na tipkovnici koji pripada tom znaku (Slika 4.29.).

```
int c = 0;
for (int i = 0; i < len; i++) {
  byte p = 0;
  if (i % 4 == 0) {
    p = (int)shaResult[c * 3] >> 2;
  } else if (i % 4 == 1) {
    p = (int)shaResult[c * 3] & B11;
    p = p << 4;
    p = p | ((int)shaResult[c * 3 + 1] >> 4);
  } else if (i % 4 == 2) {
    p = (int)shaResult[c * 3 + 1] << 2;
    p = p & B00111111;
    p = p | ((int)shaResult[c * 3 + 2] >> 6);
```

```
  } else if (i % 4 == 3) {
    p = (int)shaResult[c * 3 + 2];
    p = p & B00111111;
    c++;
  }
  for (int j = 0; j < *(&keyLayout + 1) - keyLayout; j++) {
    if (!strcmp(keyLayout[j].locale, layout)) {
      dev.sendChar(keyLayout[j].charSet[(int)p]);
    }
  }
  delay(25);
}
```

Sl. 4.29. Primjer algoritma za BASE64 kodiranje i slanja dobivenih znakova putem USB sučelja.

5. TESTIRANJE SUSTAVA

Da bi se u potpunosti mogle testirati sve funkcionalnosti ovog sustava izrađeno je privremeno testno sučelje predstavljeno pomoću mobilne aplikacije napisane u Kotlin programskom jeziku za Android operativni sustav korištenjem Android Studio razvojnog okruženja. Android aplikacija služi samo kao prikaz jednog rješenja za klijentsko sučelje dizajniranog sklopovlja jer ključni dio ovog rada je sam uređaj koji provodi sve operacije za generiranje sigurne lozinke, dok aplikacija služi samo za unos i slanje podataka na uređaj putem sigurne bežične veze. Stoga će ovdje biti prikazan samo rad aplikacije, neće biti objašnjen njen programski kod.

Programski kod Android aplikacije se može u cijelosti preuzeti s GitHub repozitorija čija se poveznica nalazi u zaključku ovog rada.

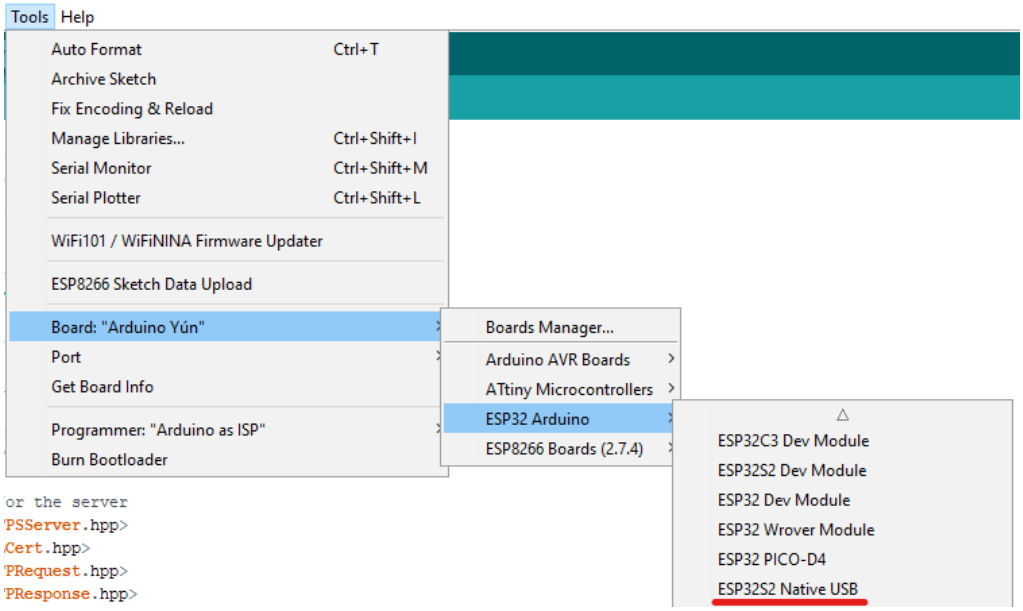
U nastavku ovog poglavlja će detaljno biti objašnjeni koraci i postupci koje korisnik mora proći za uspješno postavljanje i upotrebu sustava. Koraci pretpostavljaju da korisnik ima pristup sljedećem:

- Uređaju opisanom ovim radom
- Mobilnom telefonu s Android 11 operativnim sustavom
- Pristup računalu s instaliranim Arduino IDE-om verzije 1.8.16 ili novije
- Programskom rješenju uređaja

5.1. Programiranje uređaja

Prije programiranja uređaja važno je proći kroz korake iz poglavlja 4.3.1. da bi se kod mogao staviti na mikroupravljač bez pogrešaka. Programiranje uređaja je vrlo lagan proces jer Arduino IDE podržava programiranje ESP32-S2 mikroupravljača putem USB veze. Za programiranje uređaja potrebno je pratiti sljedeće korake:

1. Otvoriti Arudino IDE i programsko rješenje uređaja
2. Odabrati „ESP32S2 Native USB“ iz liste dostupnih razvojnih ploča iz izbornika *Tools -> Board -> ESP32 Arduino -> ESP32S2 Native USB* (Slika 5.1.)

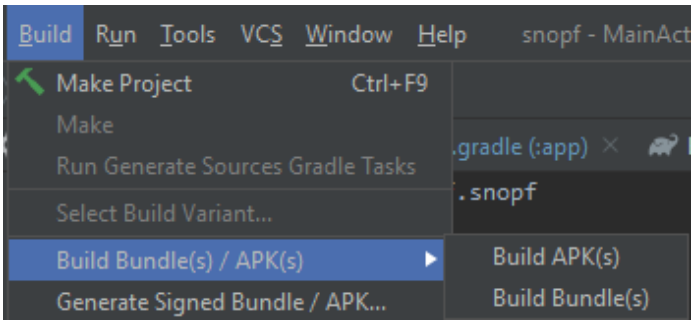


Sl. 5.1. Prikaz odabira *ESP32S2 Native USB* uređaja.

3. Pritisnuti tipkalo na uređaju i držati ga pritisnutim
4. Spojiti uređaj na računalo putem USB sučelja
5. Pustiti tipkalo
6. Pod *Tools -> Port* odabrati spojeni uređaj, u nazivu bi trebao imati *ESP32S2 Dev Module*
7. Unutar Arduino IDE-a odabrati *Sketch -> Upload*
8. Pričekati da se prijenos programa završi
9. Odspojiti uređaj

5.2. Instaliranje Android aplikacije

Android aplikacija koja služi za komunikaciju sa sklopovljem se može instalirati na dva načina: putem Android Studio razvojnog okruženja ili putem *.apk* datoteke. Upute za instaliranje aplikacije putem Android Studio razvojnog okruženja dostupne su ovdje [11]. Dok instaliranje aplikacije putem *.apk* datoteke se vrši tako da korisnik unutar Android Studio-a pritisne na opciju *Build APK(s)*, kao što je prikazano na slici 5.2.

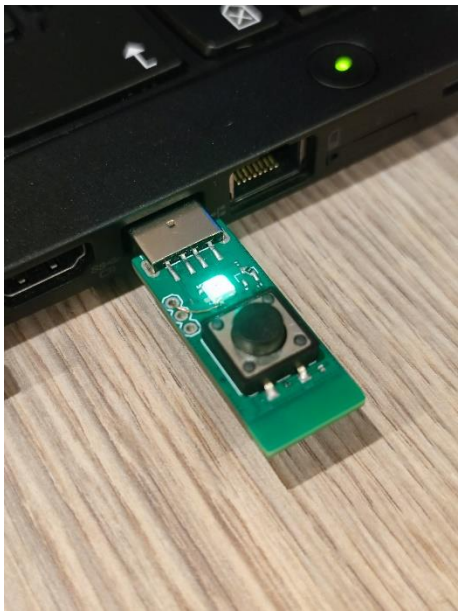


Sl. 5.2. Prikaz putanje do *Build APK(s)* opcije.

Nakon što Android Studio završi s izgradnjom *.apk* datoteke potrebno ju je prenijeti na Android uređaj na kojem se želi instalirati. Putanja do datoteke unutar glavne mape Android aplikacije je: *android\app\build\outputs\apk\debug\app-debug.apk*. Nakon prijenosa na Android uređaj potrebno ju je pokrenuti i instalirati.

5.3. Korištenje uređaja

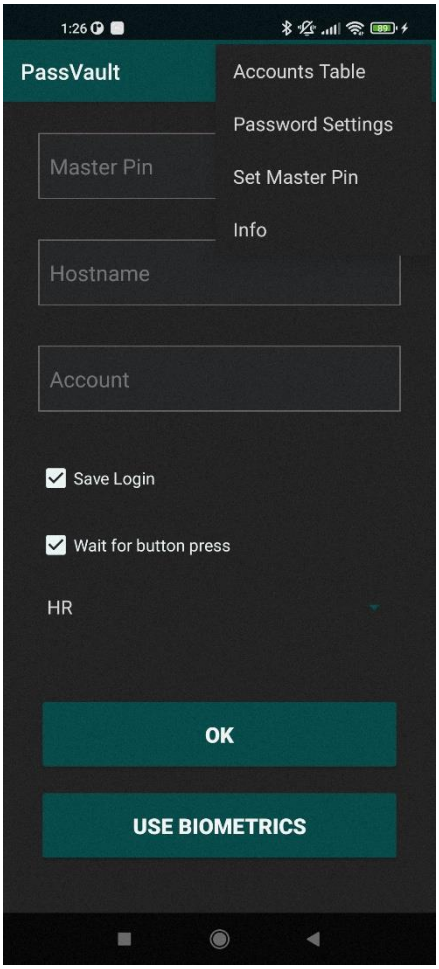
Prvi korak pri testiranju uređaja je njegovo spajanje na ciljani sustav putem USB sučelja gdje hoćemo unijeti generiranu lozinku. Na slici 5.3. je prikazan uređaj spojen na jedno od USB sučelja prijenosnog laptopa. Možemo primijetiti da uređaj na slici svijetli zelenom bojom što signalizira uspješno pokretanje AP načina rada i HTTPS servera.



Sl. 5.3. Prikaz spajanja uređaja.

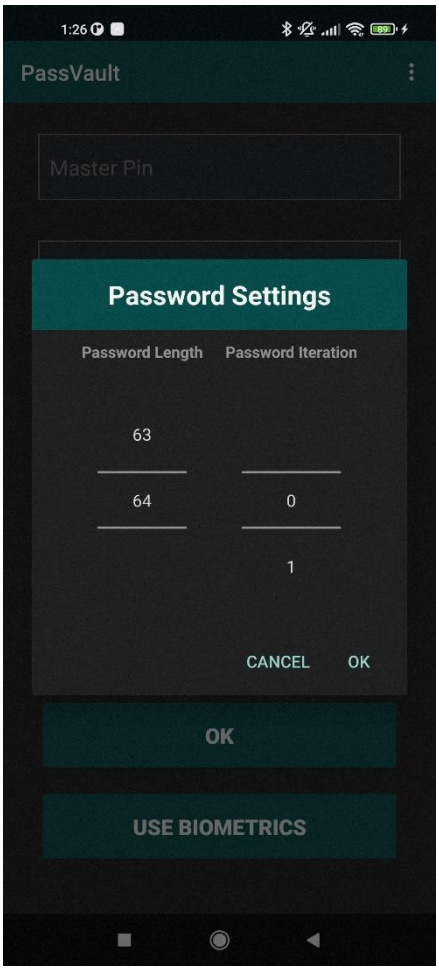
Uređaj nakon pokretanja osnovnih funkcija čeka na spajanje klijentske aplikacije. Prije pokretanja Android aplikacije potrebno je na mobilnom uređaju otići u postavke bežičnih mreža i

potražiti pristupnu točku s nazivom koji je definiran unutar programskog koda prenesenog na uređaj u poglavlju 5.1. Nakon uspješnog spajanja na uređaj možemo otvoriti aplikaciju. Na slici 5.4. možemo vidjeti izgled ekrana koji se prikazuje prilikom pokretanja aplikacije.



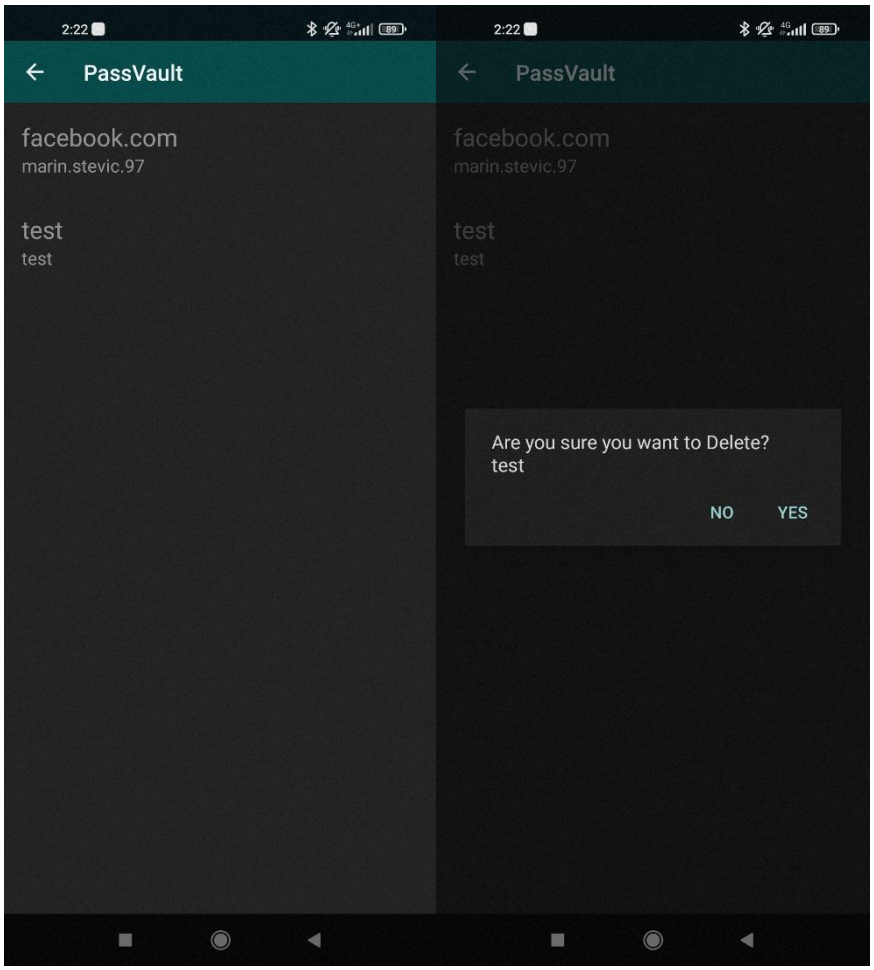
Sl. 5.4. Prikaz početnog ekrana Android aplikacije.

Iz priložene slike možemo vidjeti polja za unos podataka, opisanih u prethodnim poglavljima, potrebnih za generiranje lozinke na uređaju. Ti podaci su redom: glavni pin (*Master Pin*), ime poslužitelja (*Hostname*) i korisničko ime (*Account*). Podaci o duljini lozinke i iteraciji se mogu pristupiti otvaranjem dodatnog izbornika pritiskom na 3 točkice u desnom gornjem kutu ekrana. Na slici 5.5. je prikazan izgled izbornika za odabir duljine lozinke i iteracije.



Sl. 5.5. Prikaz izbornika za odabir duljine i iteracije lozinke.

Unutar prethodno spomenutog dodatnog izbornika imamo još dvije opcije: tablica računa (*Accounts Table*) i postavljanje glavnog pina (*Set Master Pin*). Svaki puta kada pomoću aplikacije zatražimo lozinku imamo mogućnost da odaberemo opciju spremanja računa u tablicu. Ako odaberemo tu opciju, spremit će se samo sljedeći podaci: ime poslužitelja, korisničko ime, duljina zaporke i iteracija. Tablica se sprema lokalno na Android uređaju u JSON formatu i moguće ju je prenijeti na drugi uređaj. U slučaju brisanja aplikacije, briše se i tablica računa. U tablicu se spremaju samo novi računi, ako račun već postoji unutar tablice neće se dodavati ponovno. Da bi se zatražio određeni račun potrebno je samo kliknuti na njega i svi podaci će se automatski popuniti. Zadržavanjem dodira na određenom računu dobiti ćemo mogućnost brisanja odabranog računa, što moramo potvrditi svojim odabirom. Na slici 5.6. možemo vidjeti izgled liste koja se generira iz tablice računa. Također možemo vidjeti i skočni prozor za potvrdu brisanja odabranog računa.



Sl. 5.6. Prikaz liste računa i mogućnosti brisanja pojedinog računa.

Jedini podatak koji se ne sprema unutar tablice računa je glavni pin koji je jedinstven i treba se čuvati na mjestu kojem se ne može pristupiti udaljeno. Za spremanje glavnog pina možemo koristiti prethodno navedenu opciju iz dodatnog izbornika. Odabirom opcije spremanja glavnog pina dobijemo mogućnost unosa i spremanja pina pomoću biometrijske zaštite. Ova opcija je dostupna samo na uređajima koji imaju neku vrstu takve zaštite poput: otiska prsta, prepoznavanja lica ili prepoznavanja šarenice.

Nakon što smo unijeli sve potrebne podatke ili odabrali prethodno spremljeni račun na početnom zaslonu, potrebno je pritisnuti tipku „*OK*“ ili „*Use Biometrics*“ ako smo postavili glavni pin pomoću biometrijske zaštite. Nakon odabira bilo koje od ovih dviju opcija podaci se šalju na uređaj putem sigurne bežične veze i u slučaju da smo odabrali opciju „*Wait for button press*“ na uređaju će LE dioda svijetliti plavom bojom sve dok ne pritisnemo tipkalo uređaja. Ako u određenom vremenu ne pritisnemo tipkalo, uređaj će preskočiti zahtjev i LE dioda će se vratiti u zelenu boju i čekat će se sljedeći poziv. Ako nismo odabrali opciju „*Wait for button press*“ ili smo pritisnuli tipkalo na vrijeme uređaj započinje proces dodavanja sigurnog ključa, hashiranja

podataka, generiranja lozinke i unosa na ciljani sustav imitirajući tipkovnicu. Prilikom prijenosa lozinke LE dioda uređaja svijetli crveno što signalizira da je sustav trenutno nedostupan i izvršava posao.

Na kraju testiranja sustava može se zaključiti da je uređaj potpuno ispravan i da izvršava sve funkcionalnosti prethodno navedene prvenstveno zadatkom, a potom proširene kroz ovaj rad.

6. ZAKLJUČAK

U ovom diplomskom radu izrađen je sustav za generiranje kriptiranih lozinki pomoću Android aplikacije i posebnog sklopovlja. Korištenje sustava omogućuje korisniku da uređaj priključi u drugi sustav gdje mu je potreban unos lozinke, potom pomoću aplikacije unosi tražene podatke za određeni e-račun i ti se podaci, poslani uređaju putem sigurne bežične veze, u stvarnom vremenu kriptiraju i iz njih se generira sigurna lozinka. Uređaj nakon generiranja lozinke unosi ju u priključeni sustav na odabrano polje pri tome imitirajući tipkovnicu.

Prednost ovog sustava je što uređaj sam stvara privatnu bežičnu pristupnu točku s upaljenim HTTPS serverom i zbog toga predstavlja integralni dio ovog sustava, dok se korisnička aplikacija može vrlo lagano implementirati na druge operacijske sustave. Za takvu implementaciju je potrebna mogućnost spajanja na bežične mreže i slanja podataka putem HTTPS protokola.

U svrhe testiranja i prikaza rada korisničko sučelje je napisano pomoću Kotlin programskog jezika unutar Android Studio razvojnog okruženja. Aplikacija je prilikom razvoja testirana na najnovijem javno dostupnom Android operativnom sustavu, Android 11. Shematski prikaz i dizajn tiskane pločice uređaja su napravljeni unutar programa Autodesk Eagle. Nakon ispitivanja sustava zaključeno je da iako sustav zadovoljava sve potrebe navedene u zadatku ovog rada, postoje mnoga poboljšanja koja bi omogućila lakše korištenje i povećala sigurnost uređaja. Neki od primjera nadogradnje sustava su: automatsko brisanje sigurnosnog ključa u slučaju krađe ili pokušaja provale, lakši izvoz spremljenih računa i povećanje sigurnosti pri slanju podataka putem bežične veze.

Sve datoteke za izradu uređaja su dostupne na sljedećoj poveznici:

https://github.com/MarinStevic/PassVault

LITERATURA

[1] „LastPass“, *Wikipedia*. tra. 20, 2021. Pristupljeno: ruj. 23, 2021. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=LastPass&oldid=1018847043

[2] „YubiKey“, *Wikipedia*. kol. 27, 2021. Pristupljeno: ruj. 23, 2021. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=YubiKey&oldid=1040971785

[3] „What is a Secure Static Password?“, *Yubico*. https://www.yubico.com/resources/glossary/static-password/ (pristupljeno ruj. 23, 2021).

[4] Hajo, *Snopf USB password token*. 2021. Pristupljeno: ruj. 23, 2021. [Na internetu]. Dostupno na: https://github.com/snopf/snopf

[5] „Android Studio“, *Wikipedia*. kol. 06, 2021. Pristupljeno: ruj. 23, 2021. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=Android_Studio&oldid=1037394287

[6] „What is Arduino?“ https://www.arduino.cc/en/Guide/Introduction (pristupljeno ruj. 23, 2021).

[7] „ESP32-S2 Wi-Fi MCU I Espressif“. https://www.espressif.com/en/products/socs/esp32-s2 (pristupljeno ruj. 23, 2021).

[8] „EAGLE (program)“, *Wikipedia*. kol. 14, 2021. Pristupljeno: ruj. 23, 2021. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=EAGLE_(program)&oldid=1038790627

[9] „HTTPS“, *Wikipedia*. ruj. 05, 2021. Pristupljeno: ruj. 23, 2021. [Na internetu]. Dostupno na: https://en.wikipedia.org/w/index.php?title=HTTPS&oldid=1042434026

[10] „esp32_https_server/src at master · fhessel/esp32_https_server“, *GitHub*. https://github.com/fhessel/esp32_https_server (pristupljeno ruj. 23, 2021).

[11] „Run your app“, *Android Developers*. https://developer.android.com/training/basics/firstapp/running-app (pristupljeno ruj. 23, 2021).

SAŽETAK

U ovom diplomskom radu razvijen je sustav sigurnog generiranja kriptiranih lozinki. Sustav se sastoji od Android mobilne aplikacije i posebno dizajniranog sklopovlja. Kada korisnik hoće zatražiti određenu lozinku ili generirati novu, u aplikaciju unosi sljedeće podatke: ime poslužitelja, korisničko ime, duljinu lozinke, iteraciju i glavni pin. Ti podaci se šalju na uređaj putem sigurne HTTPS veze, uređaj pomoću primljenih podataka i tajnog ključa spremljenog u svojoj memoriji generira sigurnu lozinku i prenosi ju na spojeni uređaj putem USB veze. Korisnik s unosom istih podataka će uvijek dobiti istu lozinku, ali s najmanjom promjenom će se generirati nova lozinka. Ovaj sustav ne rješava sve probleme oko sigurnosti korisničkih lozinki ali se može koristiti kao osnova za daljnji napredak.

Ključne riječi: Android, Arduino, generiranje lozinki, kriptiranje, sigurnost

ABSTRACT

Title: Device for generating encrypted passwords

In this thesis, a system for securely generating encrypted passwords has been developed. The system consists of an Android mobile application and a specially designed circuit. When a user wants to request a specific password or generate a new one, he enters the following data in the application: hostname, username, password length, iteration, and master pin. This data is sent to the device via a secure HTTPS connection, the device generates a secure password using the received data and the secret key stored in its memory and transfers it to the connected device via USB connection. A user will always receive the same password with the same input data, but with the slightest change a new password will be generated. This system does not solve all the problems around user password security, but it can be used as a basis for further progress.

Key words: Android, Arduino, encryption, password generation, security

ŽIVOTOPIS

Marin Stević, rođen 13.06.1997. godine u Osijeku, od 2004. do 2012. godine pohađao je osnovnu školu Augusta Šenoe u rodnom gradu. 2012. godine upisuje Prirodoslovno matematičku gimnaziju i za vrijeme svog školovanja odlazi na brojna natjecanja iz elektronike. Iste godine uručena mu je brončana medalja na 10. Međunarodnoj izložbi inovacija ARCA u Zagrebu. Također mu je uručena plaketa za iznimna postignuća i doprinos od osobitog značenja za razvitak tehničke kulture na području Osječko-baranjske županije. 2013. godine na Internacionalnom salonu inovacija u Ženevi uručena mu je brončana medalja i diploma Rumunjske znanstvene zajednice za visoku znanstvenu i tehnološku razinu izuma. 2016. godine završava Srednjoškolsko obrazovanje i upisuje preddiplomski smjer računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija. 2019. godine završava preddiplomski studij, te upisuje diplomski studij Informacijske i podatkovne znanosti.

_____

Potpis autora

PRILOG 1. SHEMATSKI PRIKAZ UREĐAJA