

Bluetooth in Wi-Fi ključavnica za vrata

Seminarska naloga pri četrtem predmetu poklicne mature – Računalništvo

Mentor: Aleš Volčini Avtor: Gaj Škrjanc, R 4.b

Zahvala

Iskreno se zahvaljujem mentorju Alešu Volčiniju za pomoč in nasvete pri izdelavi fizičnega izdelka in kode projekta.

Kazalo vsebine

[1. Uvod 5](#_Toc100872684)

[2. Razvoj 5](#_Toc100872685)

[3. Načrtovanje aplikacije v Fluttru 6](#_Toc100872686)

[3.1. Izgled 7](#_Toc100872687)

[3.2. Izbira BluLock 8](#_Toc100872688)

[3.2.1. Dodajanje knjižnice 9](#_Toc100872689)

[3.2.2. Deklaracija in inicializacija 9](#_Toc100872690)

[3.2.3. Dodajanje funkcij 10](#_Toc100872691)

[3.3. Izbira WebView 13](#_Toc100872692)

[3.3.1. Dodajanje knjižnice 14](#_Toc100872693)

[3.3.2. Implementacija 14](#_Toc100872694)

[4. Koda arduino in povezovanje na splet preko VPN-ja 15](#_Toc100872695)

[4.1. Povezovanje na ESP32 preko Bluetootha 16](#_Toc100872696)

[4.1.1. Dodajanje knjižnic 16](#_Toc100872697)

[4.1.2. Deklaracija in inicializacija 17](#_Toc100872698)

[4.1.3. Nastavitve Bluetootha 18](#_Toc100872699)

[4.2. Povezovanje na ESP32 preko VPN-ja na brezžično omrežje 19](#_Toc100872700)

[4.2.1. Nastavljanje VPN strežnika na mobilnem telefonu 19](#_Toc100872701)

[4.2.2. Deklaracija in inicializacija 20](#_Toc100872702)

[4.2.3. Nastavitve 20](#_Toc100872703)

[4.2.4. Kreiranje spletne strani 21](#_Toc100872704)

[5. Fizični izdelek 22](#_Toc100872705)

[5.1. Izdelava vezja 22](#_Toc100872706)

[5.2. Izdelava ohišja 23](#_Toc100872707)

[5.2.1. Aplikacija Tinkercad 23](#_Toc100872708)

[5.2.2. Izgled končanega izdelka 24](#_Toc100872709)

[6. Zaključek 25](#_Toc100872710)

[7. Viri 26](#_Toc100872711)

Kazalo slik

[Slika 1: Začetni meni 7](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872712)

[Slika 2: BluLock 8](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872713)

[Slika 3: WebView 13](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872714)

[Slika 4: Povečanje prostora za program 15](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872715)

[Slika 5: Ročno dodajanje knjižnice 16](#_Toc100872716)

[Slika 6: VPN strežnik 19](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872717)

[Slika 7: Slika vezja 1 22](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872718)

[Slika 8: Slika vezja 2 22](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872719)

[Slika 9: Izgled aplikacije 23](#_Toc100872720)

[Slika 10: Izbira designov 23](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872721)

[Slika 11: Ptičja perspektiva izdelka 24](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872722)

[Slika 12: Stranski pogled izdelka 24](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872723)

[Slika 13: Zaklenjena ključavnica 24](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872724)

[Slika 14:Odklenjena ključavnica 24](file:///C:\Users\gajsk\Desktop\Gaj_Škrjanc_Bluetooh_in_Wi-Fi_ključavnica_poklicna_matura.docx#_Toc100872725)

Povzetek

V tem strokovnem poročilu sem opisal potek izdelovanja zaklepnega mehanizma za vrata z mikroprocesorjem ESP32 in aplikacije v odprtokodnem kompletu za razvoj programske opreme Flutter s katero se ta izdelek uporablja. Zabeležil sem tudi ugotovitve in težave, ki so se mi dogajale med izdelovanjem aplikacije in fizičnega izdelka.

Ključne besede: Flutter, ESP32, Tinkercad, Dart

# Uvod

Za maturitetni izdelek sem si izbral izdelavo ključavnice za vrata. Takšno nalogo sem si izbral da se naučim povezovati znanja iz različnih področij tehnologije kar sem vedno hotel znati. Deluje preko tehnologije Bluetooth in domačega omrežja Wi-Fi. Komunikacija med spletnim mestom in mikrokrmilnikom ESP32 poteka preko VPN serverja Wireguard na mojem mobilnem telefonu.

Kratice in strokovni pojmi ki so omenjeni v seminarski nalogi pričakujem da so vam že razumljive, drugače jih vedno lahko poiščete na spletu (npr. UUID, widget, SDK…).

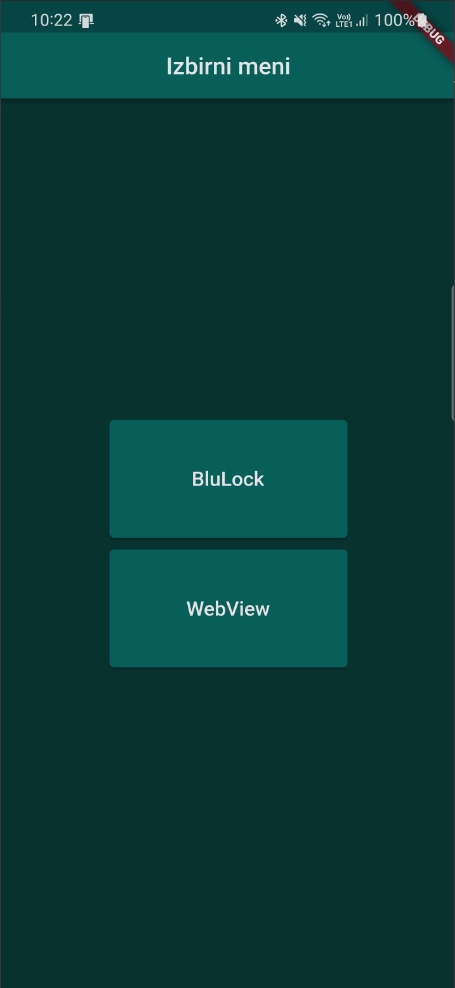
# Razvoj

Cilj naloge je bil omogočanje odklepanja in zaklepanja katerikoli vrat v hiši z uporabo telefona.­ Takšna naloga je zahtevala uporabo in povezovanje več znanj, nekatera sem pridobil v šoli, druga sem se moral naučiti sam. Projekt se deli na dva dela. To sta izdelovanje mobilne aplikacije v Fluttru in fizičnega izdelka z uporabo ESP32, katerega sem programiral v aplikaciji Arduino IDE. Ohišje za mikroprocesor sem načrtoval v aplikaciji Tinkercad, komponente pa sem sprintal v šoli z uporabo 3D tiskalnika.

# Načrtovanje aplikacije v Fluttru

Za začetek moramo približno razumeti kako Flutter deluje. Flutter je odprtokodno sistemsko razvojno orodje, ki se uporablja za razvoj mobilnih aplikacij na platformi android in IOS. Flutter sloni na programskim jezikom Dart, ki je podoben jeziku C++. Izvajanje se začne v funkciji void main(), v katero se potem dodajajo različne funkcije. Dart je preprost jezik ker se vse dogaja znotraj gnezdenih funkcij (t.i. widget). To se uporablja od samega izgleda aplikacije do vse implementirane logike.

## Izgled



Ko odpremo aplikacijo se nam ponudita dve izbiri:

* Do ESP-ja lahko dostopamo preko povezave Bluetooth,
* Do ESP-ja lahko dostopamo preko domačega omrežja Wi-fi.

Gumba sta narejena s pripomočkom »ElevatedButton()«,ki ob pritisku ustvarita povezavo na drugo stran. Na vsaki od obeh strani se pojavita dva gumba. Prvi je za odpiranje in drugi za zapiranje ključavnice.

Slika 1: Začetni meni

onPressed: () {

                    Navigator.**push**(

                      context,

                      MaterialPageRoute(builder: (context) => const BluLock()),

                    );

                  },

Tu je prikazan del programa, ki poskrbi, da se ob pritisku na tipko BluLock, odpre druga stran.

## Izbira BluLock



Tu se nam ponudi izbira da odklenemo ali zaklenemo ključavnico. Da sem omogočil Fluttru dostop do uporabe Bluetooth-a sem moral dodati dodaten SDK, ki se imenuje flutter\_blue.

Slika 2: BluLock

### Dodajanje knjižnice

Za dodajanje knjižnic v Flutter odpremo datoteko pubspec.yaml in vanj dodamo želeno SDK. To storimo sledeče:

  flutter\_blue:

    git:

      url: https://github.com/espresso3389/flutter\_blue

Za to knjižnico sem našel alternativo originalu, ker je bilo z njo lažje dostopati do Bluetootha in ni podpiral funkcije null safety.

Omogočiti sem moral tudi vse funkcije Bluetootha v AndroidManifest.xml (ker sem uporabljal le android in nimam nobene Apple naprave, nastavitev za IOS nisem spreminjal).

    <uses-permission android:name="android.permission.BLUETOOTH\_SCAN" android:usesPermissionFlags="neverForLocation" />

    <uses-permission android:name="android.permission.BLUETOOTH\_ADVERTISE" />

    <uses-permission android:name="android.permission.BLUETOOTH\_CONNECT" />

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

    <uses-permission android:name="android.permission.BLUETOOTH\_ADMIN" />

Ko dodamo knjižnico jo lahko začnemo uporabljati v datoteki main.dart, ki je glavna datoteka za izvajanje programa.

### Deklaracija in inicializacija

Najprej sem podal vse potrebne spremenljivke

final String SERVICE\_UUID = "def8570d-1d17-4908-8bcb-082e56f566b9";

  final String CHARACTERISTIC\_UUID = "f1226eca-6253-40e6-9c8b-1e0e5406e689";

  final String TARGET\_DEVICE\_NAME = "ESP32 get noti from device";

  FlutterBlue flutterBlue = FlutterBlue.instance;

  StreamSubscription<ScanResult>? scanSubScription;

  BluetoothDevice? targetDevice;

  BluetoothCharacteristic? targetCharacteristic;

  String conncetionText = "";

### Dodajanje funkcij

Prvo sem moral dodati vse funkcije, ki so bile navedene na spletni strani pub.dev, za knjižnico flutter\_blue, pod zavihkom Readme.

Najprej poiščem dosegljive naprave s funkcijo startScan(), za to pa potrebujem tudi funkcijo, ki iskanje ustavi.

**startScan**() {

**print**("starting scan");

**setState**(() {

      conncetionText = "Start scanning";

    });

    scanSubScription = flutterBlue.**scan**().**listen**((scanResult) {

      if (scanResult.device.name == TARGET\_DEVICE\_NAME) {

**print**("DEVICE FOUND");

**stopScan**();

**setState**(() {

          conncetionText = "Found Target Device!";

        });

        targetDevice = scanResult.device;

**connectToDevice**();

      }

    }, onDone: () => **stopScan**());

  }

**stopScan**() {

    flutterBlue.**stopScan**();

    scanSubScription?.**cancel**();

    scanSubScription = null;

  }

Po tem sem dodal funkcijo ki poišče pravi UUID:

**discoverServices**() async {

    if (targetDevice == null) return;

    List<BluetoothService>? services = await targetDevice?.**discoverServices**();

    services?.**forEach**((service) {

      if (service.uuid.**toString**() == SERVICE\_UUID) {

        for (var characteristic in service.characteristics) {

          if (characteristic.uuid.**toString**() == CHARACTERISTIC\_UUID) {

            targetCharacteristic = characteristic;

**writeData**("Hi thereeeeee, esp");

**setState**(() {

              conncetionText = "All Ready With ${*targetDevice*?.*name*}";

            });

          }

        }

      }

    });

  }

Ter funkciji za povezavo na želeno napravo:

**connectToDevice**() async {

    if (targetDevice == null) return;

**setState**(() {

      conncetionText = "Device Connecting...";

    });

    await targetDevice?.**connect**();

**print**("DEVICE CONNECTED");

**setState**(() {

      conncetionText = "Device Connected!";

    });

**discoverServices**();

  }

**disconnectFromDevice**() {

    if (targetDevice == null) return;

    targetDevice?.**disconnect**();

**setState**(() {

      conncetionText = "Device Disconnected!";

    });}

Funkcija za pošiljanje podatkov preko Bluetootha pa je sledeča:

**writeData**(String data) async {

    if (targetCharacteristic == null) return;

    List<int> bytes = utf8.**encode**(data);

    targetCharacteristic?.**write**(bytes);

  }

Ko pa pritisnemo gumb »odkleni« pa se v povezano napravo pošlje 1 ali 0. Ti številki premikata servo motor.

onPressed: () => **writeData**("1"),

onPressed: () => **writeData**("0"),

## Izbira WebView



Slika 3: WebView

Da sem dostopal do ESP-ja preko brezžičnega omrežja sem moral najprej v Flutter implementirati možnost dostopanja do brskalnika. To sem dosegel s knjižnico webview\_flutter.

### Dodajanje knjižnice

Knjižnico webview se doda v Flutter enako kot vse ostale knjižnice. V pubscpec.yaml dodamo ime knjižnice in verzijo.

  webview\_flutter: ^3.0.1

V tem primeru sem dodal originalno knjižnico, ker je vsebovala vse kar sem potreboval.

### Implementacija

Dostop do brskalnika je v aplikacijo dodati veliko lažje kot implementacija Bluetootha. Potrebujemo samo nov razred, kateremu v telo vstavimo funkcijo WebView(). Znotraj pa povemo katera spletna stran se prikaže na zaslonu.

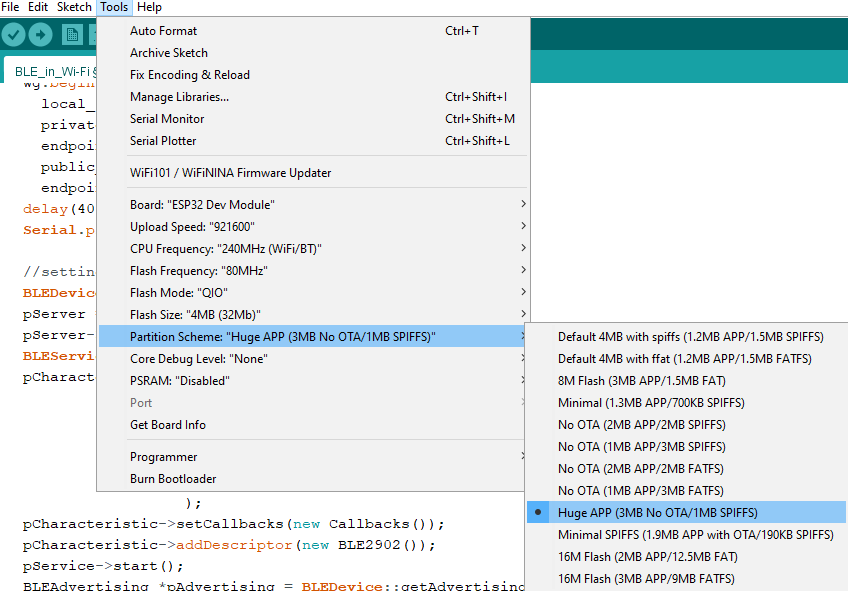
body: const WebView(

          javascriptMode: JavascriptMode.unrestricted,

          initialUrl: '192.168.200.2',

        ));

# Koda arduino in povezovanje na splet preko VPN-ja

Da sem se z aplikacijo povezal na ESP32, sem moral napisati program v aplikaciji Arduino IDE. Ta bo sprejemala poslano 1 ali 0 iz Bluetooth aplikacije ali pa iz spletne strani prebere končnico URL-ja in primerno temu premakne servo motor. Tu sem imel manjše težave, ker je koda zavzemala več prostora kot ga je bilo na razpolago tako da sem moral spremeniti particijo, zadolženo za aplikacijo in jo povečati iz 1,2MB na 3MB.

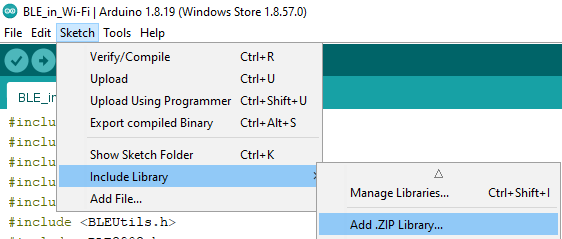
Slika 4: Povečanje prostora za program

## Povezovanje na ESP32 preko Bluetootha

V tem delu bom predstavil kodo, ki nam omogoči sprejemanje nizov preko povezave Bluetooth.

### Dodajanje knjižnic

Za omogočanje Bluetootha na ESP-ju sem moral najti knjižnico na spletu, ker v programu Arduino IDE ni ponujena. To sem naredil tako, da sem prenesel .zip datoteko, ki sem jo potem ročno dodal v aplikacijo.



Slika 5: Ročno dodajanje knjižnice

### Deklaracija in inicializacija

Najprej sem moral generirati svoj UUID za strežbo in značilnost, to je potrebno zato, da oddajnik ESP32 prepoznajo druge naprave. Dodati je bilo potrebno še dve povratni funkciji (to sta funkciji, ki ju podamo kot parameter v drugo funkcijo, kjer se potem ta funkcija izvede). Prva poskrbi, da ESP32 začne obveščati, da je na voljo za povezavo. Druga pa bere podatke, ki so bili poslani v mikrokrmilnik.

#define **SERVICE\_UUID**        "def8570d-1d17-4908-8bcb-082e56f566b9"

#define **CHARACTERISTIC\_UUID** "f1226eca-6253-40e6-9c8b-1e0e5406e689"

class ServerCallbacks: public BLEServerCallbacks {

    void **onConnect**(BLEServer\* *pServer*) {

      deviceConnected = true;

      BLEDevice::**startAdvertising**();

    };

    void **onDisconnect**(BLEServer\* *pServer*) {

      deviceConnected = false;

    }

};

class Callbacks: public BLECharacteristicCallbacks {

    void **onWrite**(BLECharacteristic \**pCharacteristic*) {

      std::string value = *pCharacteristic*->**getValue**();

      if (value.**length**() > 0) {

        for (int i = 0; i < value.**length**(); i++) {

          if (value[i] == '1') {

            Serial.**println**("Odklepam");

**digitalWrite**(ODKLENJENO, HIGH);

**digitalWrite**(ZAKLENJENO, LOW);

            angle = 0;

            servo.**write**(angle);

**delay**(15);

          }

          if (value[i] == '0') {

            Serial.**println**("Zaklepam");

**digitalWrite**(ODKLENJENO, LOW);

**digitalWrite**(ZAKLENJENO, HIGH);

            angle = 180;

            servo.**write**(angle);

**delay**(15);

          }}}}};

### Nastavitve Bluetootha

V tem delu kode sem nastavil Bluetooth strežnik. Teh nastavitev je kar nekaj, zato jih ne bom posebej opisoval. Ob koncu izseka kode je ESP32 nastavljen tako, da se lahko nanjo povežemo z odjemalčevo napravo (npr. mobilnim telefonom).

  BLEDevice::**init**("ESP32 get noti from device");

  pServer = BLEDevice::**createServer**();

  pServer->**setCallbacks**(new **ServerCallbacks**());

  BLEService \*pService = pServer->**createService**(SERVICE\_UUID);

  pCharacteristic = pService->**createCharacteristic**(

                      CHARACTERISTIC\_UUID,

                      BLECharacteristic::PROPERTY\_READ   |

                      BLECharacteristic::PROPERTY\_WRITE  |

                      BLECharacteristic::PROPERTY\_NOTIFY |

                      BLECharacteristic::PROPERTY\_INDICATE

                    );

  pCharacteristic->**setCallbacks**(new **Callbacks**());

  pCharacteristic->**addDescriptor**(new **BLE2902**());D

  pService->**start**();

  BLEAdvertising \*pAdvertising = BLEDevice::**getAdvertising**();

  pAdvertising->**addServiceUUID**(SERVICE\_UUID);

  pAdvertising->**setScanResponse**(false);

  pAdvertising->**setMinPreferred**(0x0);

  BLEDevice::**startAdvertising**();

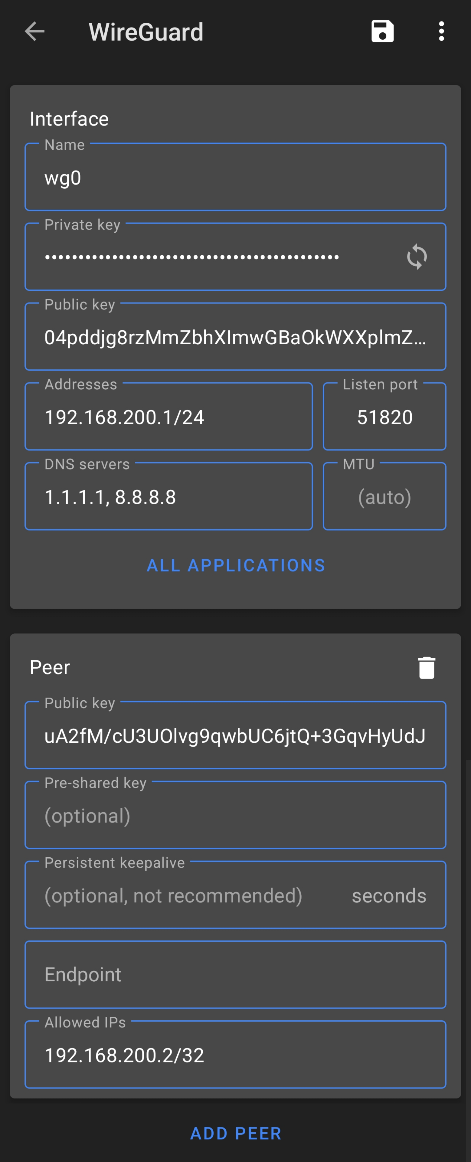
**digitalWrite**(ODKLENJENO, HIGH);

  Serial.**println**("Bluetooth is set, starting advertising...");

## Povezovanje na ESP32 preko VPN-ja na brezžično omrežje

Da sem poskrbel za varnost pri povezavi na brezžično omrežje sem se odločil, da med strežnik in odjemalca postavim VPN. To sem storil z aplikacijo Wireguard na mobilnem telefonu. Pri vzpostavitvi VPN-ja sem imel težave, zato mi je pri tem delu naloge pomagal sošolec Žiga Kralj. Nalogo sem potem nadaljeval v aplikaciji Arduino IDE kjer sem dokončal vse potrebno, da se je ESP povezal na internet preko Wireguard serverja na mojem mobilnem telefonu.

### Nastavljanje VPN strežnika na mobilnem telefonu



Slika 6: VPN strežnik

Ta del naloge mi je pomagal narediti Žiga Kralj. Takoj ko sem ustvaril nov server se avtomatsko generirata privatni in javni ključ. Oknu za uporabniški vmesnik dodamo IP naslov ki ga želimo uporabljati, DNS serverje in vrata, na katerem komunikacija poteka.

Po tem pa sem dodal enega uporabnika. V mojem primeru bo to ESP32. Dodelil sem mu njegov javni ključ in dovolil, da se nanj poveže le naprava, ki ima IP 192.168.200.2

To so vse potrebne nastavitve za strežnik na telefonu. Sedaj se lahko nanj povežem z mikrokrmilnikom.

### Deklaracija in inicializacija

Najprej sem definiral vse spremenljivke ki sem jih potreboval. Imena spremenljivk dobro opišejo njihov namen in zato ne bom šel v podrobno razlago. Zaradi varnosti, geslo domačega brezžičnega omrežja ni prikazano.

WiFiServer **server**(80);

static WireGuard wg;

static HTTPClient httpClient;

char ssid[] = "Gaj\_Wifi";

char password[] = "\*\*\*\*\*\*\*\*";

char private\_key[] = "0Pw/m0VlbsN1qkpNeGxnCRzQI4bcRnMHNyyWM7NXlWE=";

char public\_key[] = "04pddjg8rzMmZbhXImwGBaOkWXXplmZM6aEcwfuzESw=";

char endpoint\_address[] = "192.168.1.5";

IPAddress **local\_ip**(192, 168, 200, 2);

int endpoint\_port = 51820;

### Nastavitve

Nastavitve Wi-Fija so nekoliko krajše od nastavitev Bluetootha. Po koncu izseka programa je brezžično omrežje prižgano in povezano na Wireguard strežnik na mobilnem telefonu.

WiFi.**begin**(ssid, password);

  while ( !WiFi.**isConnected**() ) {

**delay**(1000);

  }

  server.**begin**();

**configTime**(1 \* 60 \* 60, 0, "ntp.jst.mfeed.ad.jp","ntp.nict.jp","time.google.com");

  wg.**begin**(

    local\_ip,

    private\_key,

    endpoint\_address,

    public\_key,

    endpoint\_port);

**delay**(4000);

### Kreiranje spletne strani

V naslednjem izseku kode sem na spletni strežnik naložil spletno stran. Na njej sta dva gumba, ki dodajata niz (odkleni ali zakleni) na konec URL-ja, katerega program prebere in temu primerno zavrti servo motor.

WiFiClient client = server.**available**();

  if (client) {

    Serial.**printf**("connected: %d", povezani);

    povezani++;

    Serial.**println**("New Client.");

    String currentLine = "";

    while (client.**connected**()) {

      if (client.**available**()) {

        char c = client.**read**();

        Serial.**write**(c);

        if (c == '\n') {

          if (currentLine.**length**() == 0) {

          client.**print**(" <!DOCTYPE html> "); *// HTML spletne strani*

            client.**println**();

            break;

          } else {

          currentLine = "";

          }

        } else if (c != '\r') {

          currentLine += c;

        }

        if (currentLine.**endsWith**("GET /zakleni/")) {

          Serial.**println**("Zaklepam");

**digitalWrite**(ZAKLENJENO, HIGH);

**digitalWrite**(ODKLENJENO, LOW);

          angle = 0;

          servo.**write**(angle);

        }

        if (currentLine.**endsWith**("GET /odkleni/")) {

          Serial.**println**("Odklepam");

**digitalWrite**(ZAKLENJENO, LOW);

**digitalWrite**(ODKLENJENO, HIGH);

          angle = 180;

          servo.**write**(angle);

        }

      }

    }

    client.**stop**();

  }

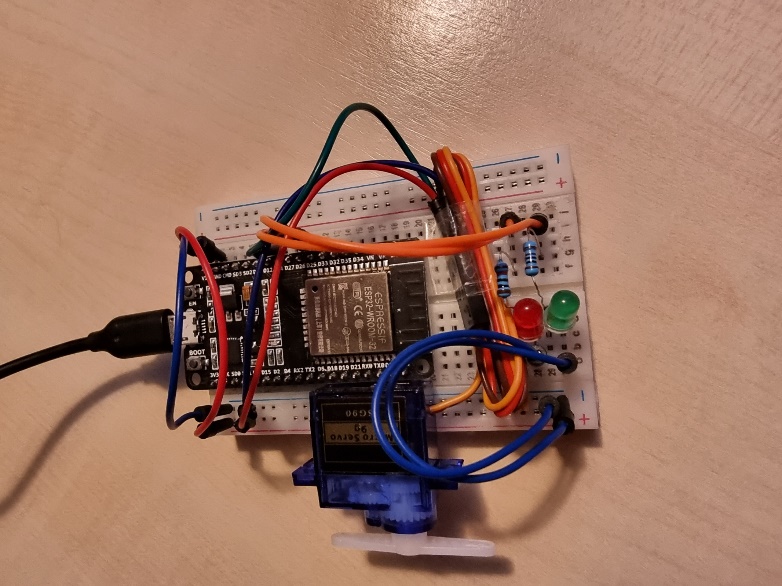
# Fizični izdelek

Del moje seminarske naloge je tudi izdelava fizične ključavnice. Trenutno je izdelek še prototip. Sestavlja ga ESP32 s servo motorjem, drugi del fizičnega izdelka je pa ohišje z zaklepnim mehanizmom.

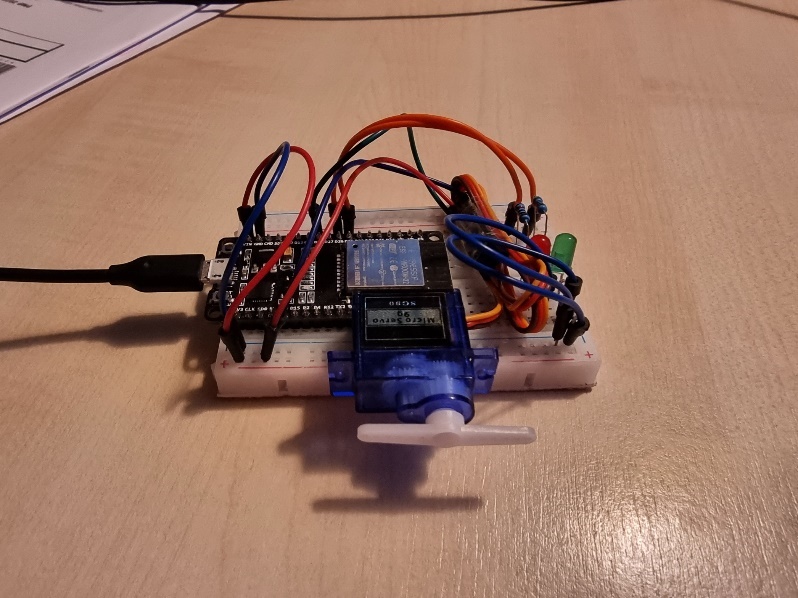
## Izdelava vezja

Na začetku sem želel uporabiti arduino uno in zunanji Bluetooth modul, vendar sem se kasneje odločil za boljšo alternativo in sicer ESP32, ki je kompaktnejši ter ima že vgrajeno strojno podporo za protokola Bluetooth in Wi-Fi.

Za vezje sem potreboval:

* Modul EPS32 devkit,
* Kabel s priključkom mikro USB,
* Micro servo 9g,
* 7 žic,
* 2 x 1k Ohm upora ter,
* Dve LED diodi.

Slika 7: Slika vezja 1



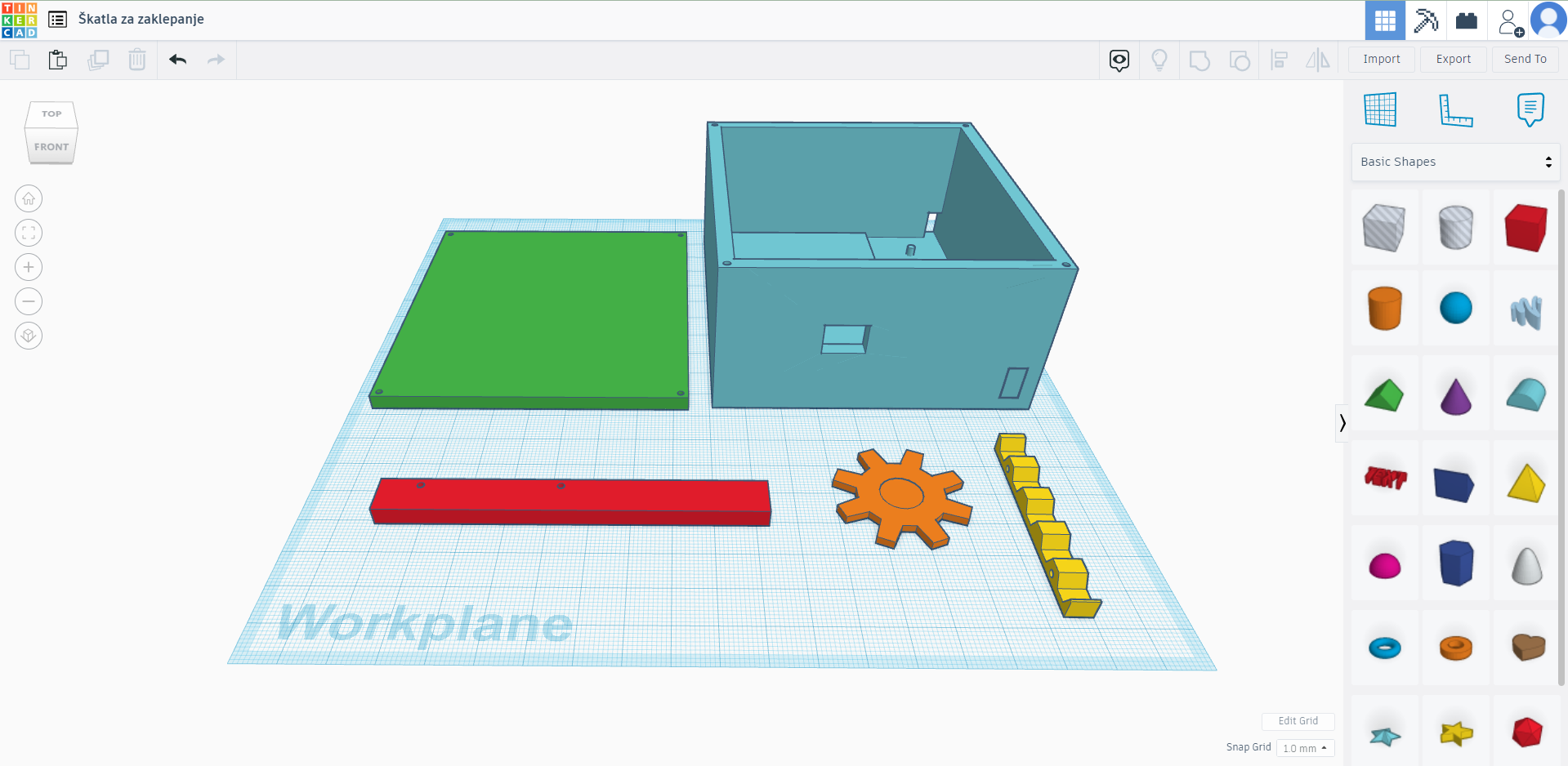
Slika 8: Slika vezja 2

## Izdelava ohišja

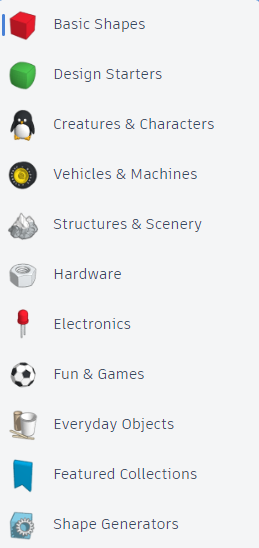
V zadnjem delu naloge bom predstavil spletno aplikacijo Tinkercad in končni izgled fizičnega izdelka.

### Aplikacija Tinkercad

Da sem oblikoval ohišje za ESP32 sem si izbral zastonj aplikacijo za 3D oblikovanje Tinkercad. Aplikacija mi je zelo všeč in jo zato mislim uporabiti tudi v drugih projektih. Uporaba je zelo preprosta tako da jo lahko na kratko razložim.



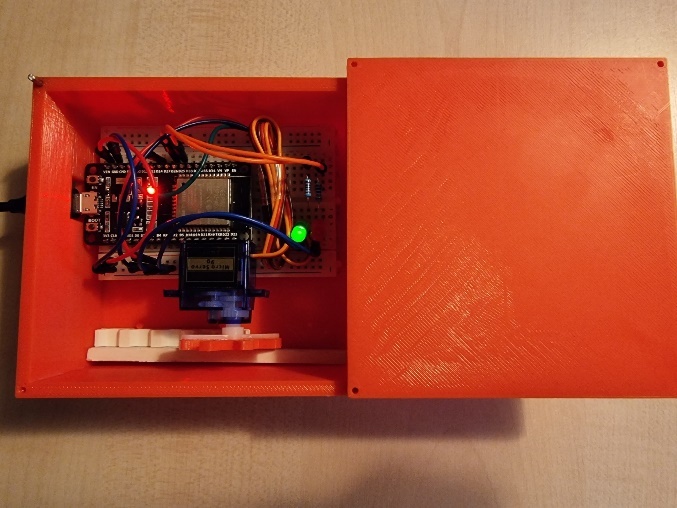
Slika 9: Izgled aplikacije

Zgoraj levo se nahajajo kontrole za premikanje kamere kot tudi možnosti za lepljenje, kopiranje in brisanja komponent. V zgornji desni strani pa se nahaja orodje za združevanje in razdruževanje komponent. V skrajnem desnem kotu pa imamo možnost za uvoz, izvoz in pošiljanje projekta. Pod tem imamo merilo, šablono in komentarje, pod tem pa lahko izbiramo tako med osnovnimi kot med kompleksnejšimi designi predmetov, katerih veliko je že narejenih (npr. zobnik)

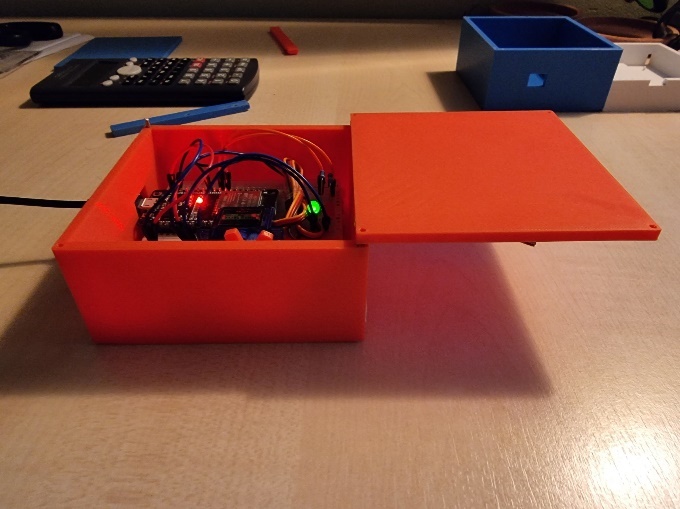
Slika 10: Izbira designov

### Izgled končanega izdelka

Za vse komponente izdelka (oranžne in bele) sem uporabil šolski 3D tiskalnik.



Slika 11: Ptičja perspektiva izdelka



Slika 12: Stranski pogled izdelka

****



Slika 13: Zaklenjena ključavnica

Slika 14:Odklenjena ključavnica

# Zaključek

Naloga, ki sem si jo izbral, mi je bila všeč. Dosegel sem skoraj vse cilje, ki sem si jih zadal. V nadaljnjem bi želel narediti Wireguard strežnik v oblaku, da bi se nanj povezal s telefonom in z ESP-jem in bi tako lahko odklepal vrata iz kjerkoli.

# Viri

<https://stackoverflow.com/>

<https://docs.flutter.dev/get-started/install>

<https://pub.dev/packages/flutter_blue>

<https://pub.dev/packages/webview_flutter>

<https://create.arduino.cc/projecthub/karmette/basic-led-setup-for-beginners-0a124a>

<https://www.youtube.com/watch?v=1ukSR1GRtMU&list=PL4cUxeGkcC9jLYyp2Aoh6hcWuxFDX6PBJ>

<https://docs.flutter.dev/cookbook/navigation/navigation-basics>

<https://github.com/espresso3389/flutter_blue>

<https://stackoverflow.com/questions/56637375/navigating-to-another-page-from-drawer-menu-and-setting-title-to-app-bar>

<https://medium.com/codechai/playing-with-appbar-in-flutter-3a8abd9b982a>

<https://api.flutter.dev/flutter/material/Icons-class.html>

<https://api.flutter.dev/flutter/material/AppBar/automaticallyImplyLeading.html>

<https://docs.flutter.dev/cookbook/navigation/returning-data>

<https://stackoverflow.com/questions/60525482/problem-with-ble-module-exception-another-scan-is-already-in-progress>

<https://stackoverflow.com/questions/43822671/how-do-i-set-the-background-color-of-my-main-screen-in-flutter>

<https://stackoverflow.com/questions/60557641/flutter-webview-empty-page-using-provider>

<https://www.wireguard.com/quickstart/#key-generation>

<https://techoverflow.net/2021/12/31/esp32-wireguard-example-with-http-access-over-wireguard-platformio/>

<https://github.com/micahmo/WireGuardServerForWindows>

Izjava o avtorstvu

Izjavljam, da je strokovno poročilo o Bluetooth in Wi-Fi ključavnici za vrata izdelano samostojno s pomočjo navedene literature in pod vodstvom mentorja.

14.4.2022 Gaj Škrjanc