

Bluetooth in Wi-Fi ključavnica za vrata

Seminarska naloga pri četrtem predmetu poklicne mature – Računalništvo

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

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

Kazalo vsebine

[Povzetek 5](#_Toc100693696)

[1. Uvod 6](#_Toc100693697)

[2. Razvoj 6](#_Toc100693698)

[3. Načrtovanje aplikacije v Fluttru 7](#_Toc100693699)

[3.1. Izgled 8](#_Toc100693700)

[3.2. Izbira BluLock 9](#_Toc100693701)

[3.2.1. Dodajanje knjižnice 10](#_Toc100693702)

[3.2.2. Inicializacija 10](#_Toc100693703)

[3.2.3. Dodajanje funkcij 11](#_Toc100693704)

[3.3. Izbira WebView 14](#_Toc100693705)

[3.3.1. Dodajanje knjižnice 15](#_Toc100693706)

[3.3.2. Implementacija 15](#_Toc100693707)

[4. Koda arduino in povezovanje na splet preko VPN-ja 16](#_Toc100693708)

[4.1. Povezovanje na ESP32 preko Bluetootha 17](#_Toc100693709)

[4.1.1. Dodajanje knjižnic 17](#_Toc100693710)

[4.1.2. Deklaracija in inicializacija 18](#_Toc100693711)

[4.1.3. Nastavitve Bluetootha 19](#_Toc100693712)

[4.2. Povezovanje na ESP32 preko VPN-ja na brezžično omrežje 20](#_Toc100693713)

[4.2.1. Nastavljanje VPN strežnika na mobilnem telefonu 20](#_Toc100693714)

[4.2.2. Deklaracija in inicializacija 21](#_Toc100693715)

[4.2.3. Nastavitve 21](#_Toc100693716)

[4.2.4. Kreiranje spletne strani 22](#_Toc100693717)

[5. Fizični izdelek 23](#_Toc100693718)

[5.1. Izdelava vezja 23](#_Toc100693719)

[5.2. Izdelava ohišja 24](#_Toc100693720)

Kazalo slik

[1 - Začetni meni 8](file:///G:\Other%20computers\My%20Computer\R4B\maturitetna%20naloga\Gaj_Škrjanc_maturitetna_naloga.docx#_Toc100693721)

[2 - BluLock 9](file:///G:\Other%20computers\My%20Computer\R4B\maturitetna%20naloga\Gaj_Škrjanc_maturitetna_naloga.docx#_Toc100693722)

[3 - WebView 14](file:///G:\Other%20computers\My%20Computer\R4B\maturitetna%20naloga\Gaj_Škrjanc_maturitetna_naloga.docx#_Toc100693723)

[4 - povečanje prostora za program 16](#_Toc100693724)

[5 - Ročno dodajanje knjižnice 17](#_Toc100693725)

[6 - VPN strežnik 20](file:///G:\Other%20computers\My%20Computer\R4B\maturitetna%20naloga\Gaj_Škrjanc_maturitetna_naloga.docx#_Toc100693726)

[7 - slika vezja 1 23](file:///G:\Other%20computers\My%20Computer\R4B\maturitetna%20naloga\Gaj_Škrjanc_maturitetna_naloga.docx#_Toc100693727)

[8 - slika vezja 2 23](file:///G:\Other%20computers\My%20Computer\R4B\maturitetna%20naloga\Gaj_Škrjanc_maturitetna_naloga.docx#_Toc100693728)

[9 - izgled aplikacije Tinkercad 24](#_Toc100693729)

[10 - Vsa izbira oblik 24](file:///G:\Other%20computers\My%20Computer\R4B\maturitetna%20naloga\Gaj_Škrjanc_maturitetna_naloga.docx#_Toc100693730)

# 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. 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…).

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, katerega komunikacija med spletnim mestom in mikrokrmilnikom ESP32 poteka preko VPNja Wiregurad.

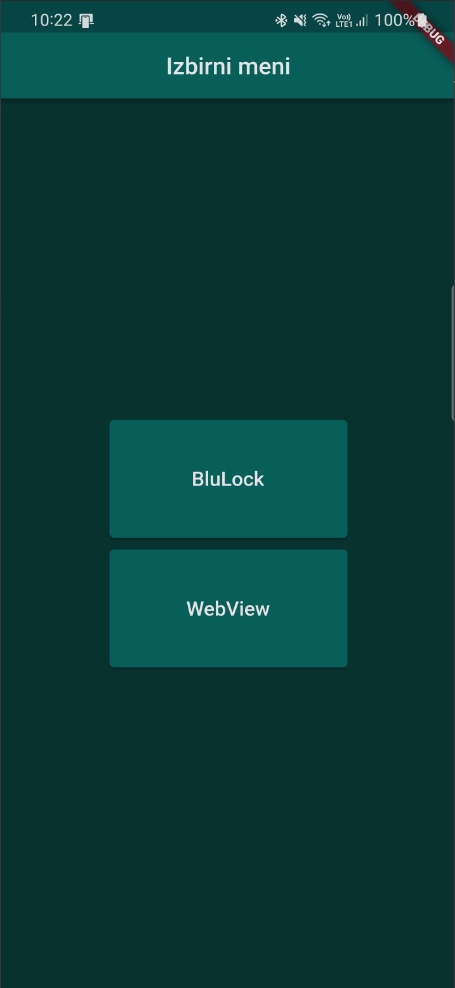
# Razvoj

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 in to sta izdelovanje mobilne aplikacije v fluttru in izdelovanje fizičnega izdelka z uporabo ESP32, ohišje za mikroprocesor pa 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 odprtokodni sistemsko razvojno orodje, ki se uporablja za razvoj mobilnih aplikacij na platformi android in IOS. Flutter deluje s programskim jezikom Dart, ki je podoben jeziku C++, katerega smo se učili v šoli. Vse se začne v funkciji void main(), v katero se potem dodajajo različne funkcije. Dart je preprost jezik za naučiti ker se vse dogaja znotraj gnezdenih funkcij (t.i. widget). To se uporabla od samega izgleda aplikacije do vse implementirane logike.

## Izgled



Ko odpremo aplikacijo se nam ponudita dve izbiri:

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

Gumba sta narejena v gnezdi funkciji »ElevatedButton()«, katera ob pritisku ustvarita povezavo na drugo stran. Obe izbiri ob pritisku ponudita dva dodatna gumba. Prvi je za odpiranje in drugi za zapiranje ključavnice.

onPressed: () {

                    Navigator.**push**(

                      context,

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

                    );

                  },

Tukaj se vidi kako ob pritisku na gumb aplikacija prikaže drugo vsebino, ki se nahaja v funkciji BluLock().

1 - Začetni meni

## Izbira BluLock



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

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 s to lažje dostopati do Bluetootha in ni podpiral možnosti null safety-ja, ki preprečuje da so nekatere inicializacije nastavljene na null.

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

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

### Inicializacija

Najprej sem inicializiral 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

Najprej sem moral napisati funkcije, ki so opisane v Readme datoteki na strani pub.dev za knjižnico flutter\_blue.

Najprej poiščemo dosegljive naprave s funkcijo startScan(), za to pa potrebujemu 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



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.

3 - WebView

### 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

### 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(

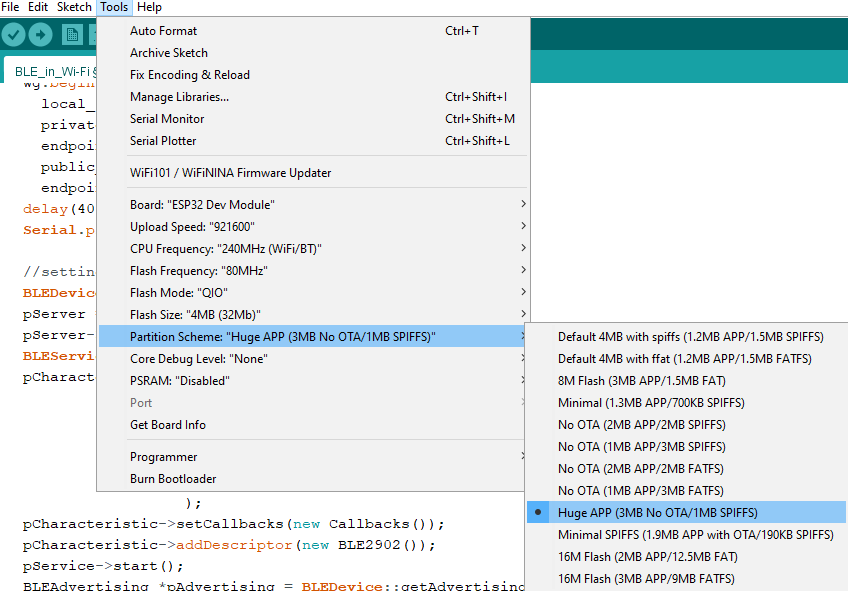
          javascriptMode: JavascriptMode.unrestricted,

          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 svoj program v Arduino IDE, ki sprejema vanj 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 povečati iz 1,2MB na 3MB.

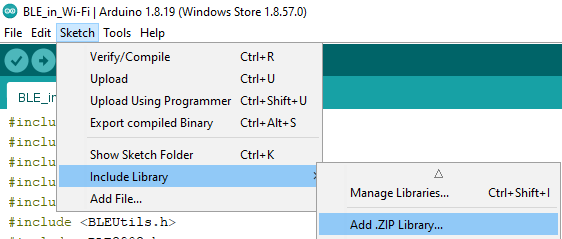


4 - povečanje prostora za program

## Povezovanje na ESP32 preko Bluetootha

### 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.



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, in druga 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;

    }

};

if (value[i] == '1') {

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

**digitalWrite**(ODKLENJENO, HIGH);

**digitalWrite**(ZAKLENJENO, LOW);

            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**();

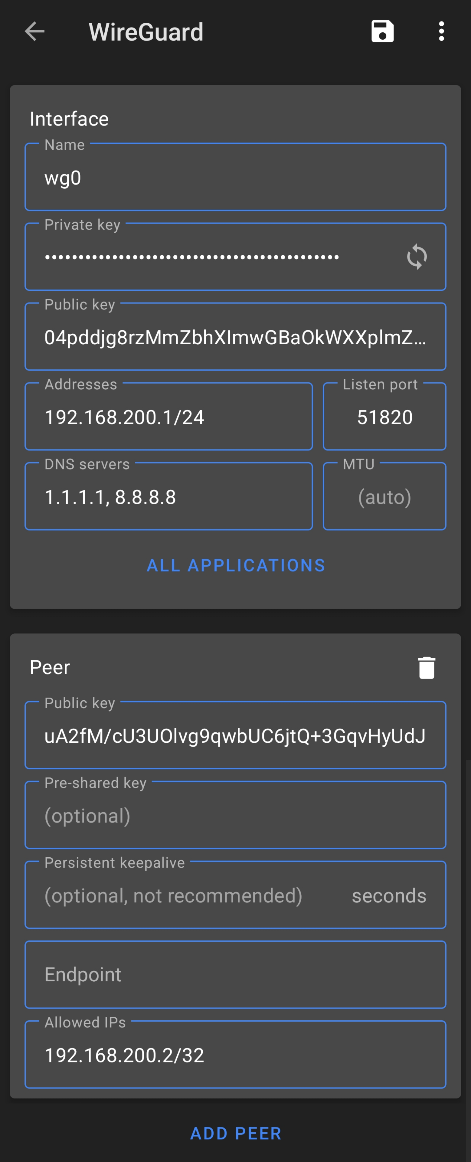
**digitalWrite**(ODKLENJENO, HIGH);

  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



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 prikazan.

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 ULR-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**();

  }

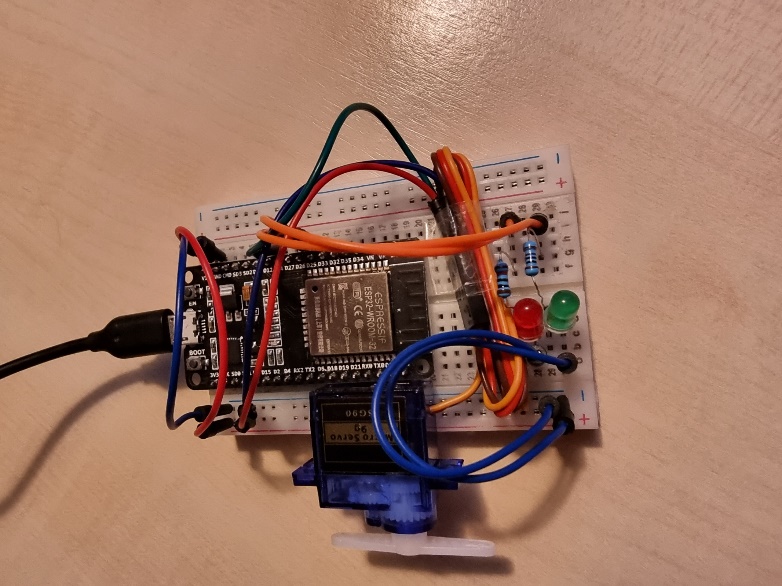
# Fizični izdelek

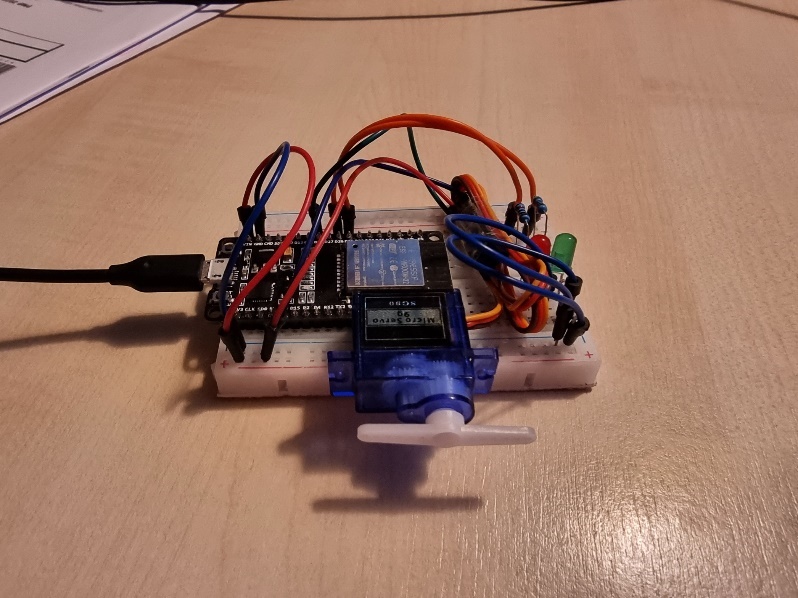
Del moje seminarske naloge je tudi izdelava fizične ključavnice. Trenutno je izdelek se v predstavitveni fazi. Sestavlja ga ESP32 s servo motorjem, drugi del fizičnega izdelka je pa ohišje z zaklepnim mehanizmom.

## Izdelava vezja

Na začetku sem želel uporabiti arduino uno in zunanji Bluetooth modul vendar sem se kasneje odločil za boljšo alternativo ESP32, ker je kompaktnejši ter ima že vgrajen Bluetooth in Wi-Fi.

Za vezje sem potreboval:

* EPS32 devkit,
* USB micro,
* Micro servo 9g,
* 7 žic,
* 2x1k Ohm upora ter,
* Dve LED diodi.

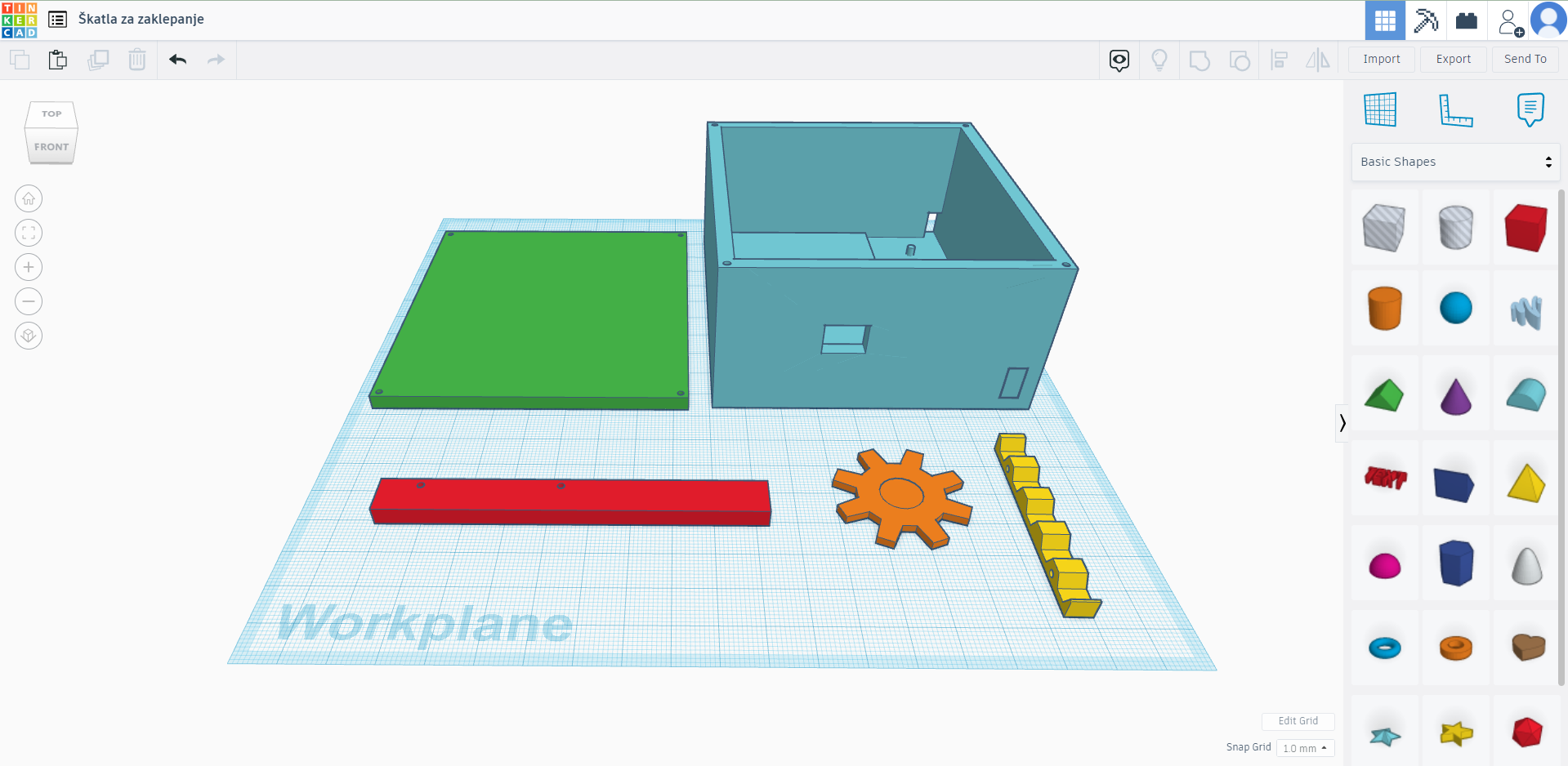


7 - slika vezja 1

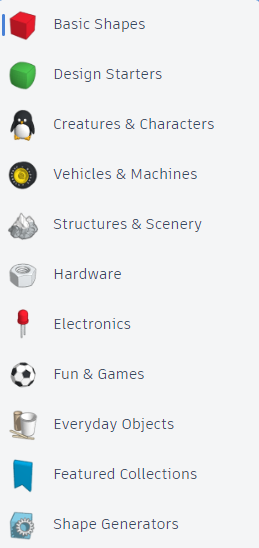
8 - slika vezja 2

## Izdelava ohišja

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.



9 - izgled aplikacije Tinkercad

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 designom predmetov, veliko jih je že narejeih (npr. zobnik)

10 - Vsa izbira oblik

# 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.