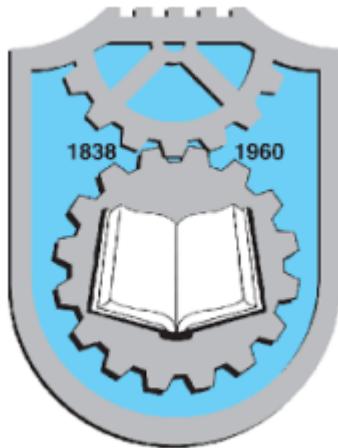


Универзитет у Крагујевцу

Факултет инжењерских наука



Предмет:

Програмирање мобилних апликација

Тема:

Мобилна апликација за даљинско управљање бравом
улаznих врата и осветљењем у просторијама куће

Студенти:

Вељко Максимовић 634/2018

Ђорђе Гачић 626/2018

Професор:

Др Ненад Грујовић

Асистент:

Вукашин Славковић, дипл. маш. инж.

Крагујевац, фебруар 2022.

Садржај:

1	Увод	3
2	Потребе реалног система	4
3	Опис коришћене технологије и компоненти	5
3.1	Опис хардверских компоненти	5
3.2	Софтверске потребе пројекта	8
4	Начин повезивања хардверских компоненти	10
5	Софтверска реализација пројекта	13
5.1	Програмирање мокроконтролера	13
5.2	Програмирање мобилне апликације	16
6	Израда макете куће	26
7	Изглед и демонстрација рада апликације	28
8	Литература	29

1 Увод

У данашње време, када су мобилни телефони и остали паметни уређаји свакодневно присутни у животу просечног човека, отварају се разне могућности за њихову употребу у многим сферама живота. Паметни телефони данас имају многе функције и, ако се на исправан начин употребљавају, могу много да олакшају живот и разне послове. Поред уobičajene употребе за размену позива и порука тј. за комуникацију уопште, паметни телефони се користе и за забаву, оријентацију у простору, фотографисање, снимање видео материјала, мобилно банкарство, праћење разних параметара и догађаја, опште и научно информисање као и уопштено за приступ Интернету. Поред свега наведеног, све више улазе у примену мобилне апликације које бежично (даљински) управљају (Wi-Fi, Bluetooth итд.) неким физичким уређајима или електронским компонентама. Једна од примена тих апликација јесте и у домаћинству како би се контролисао рад многих уређаја као нпр. кућни апарати (шпорети, фрижидери, веш машине), сијалице, браве итд.

У овом пројекту је реализована једна таква апликација чија је функција управљање целокупним осветљењем и бравом на улазним вратима куће. У оквиру пројекта су, поред саме апликације, физички присутне и уграђене све електронске компоненте које симулирају поменуте процесе, а израђена је и макета куће како би функционалност апликације била веродостојније приказана.

Мотивација за израду овог пројекта била је идеја да се повећа сигурност домаћинства тако што ће контрола неких његових делова бити на пар кликова од самог корисника. Тако бисмо преко неколико кликова могли бити сигурни да су сва светла у кући угашена и да су улазна врата закључчана без да идемо да проверавамо. Такође, ова апликација представља само део могућности које пружа технологија тзв. паметних дома (енг. Smart Home) и очекује се да у будућности мобилне апликације тог типа буду у свакодневној употреби код већине људи.

2 Потребе реалног система

Имплеменација реалног система на тему даљинског управљања бравом и осветљењем у домаћинству је прилично захтевна због великог броја компоненти које се морају уклопити у једну практичну и функционалну целину.

Реални систем захтева следеће компоненте:

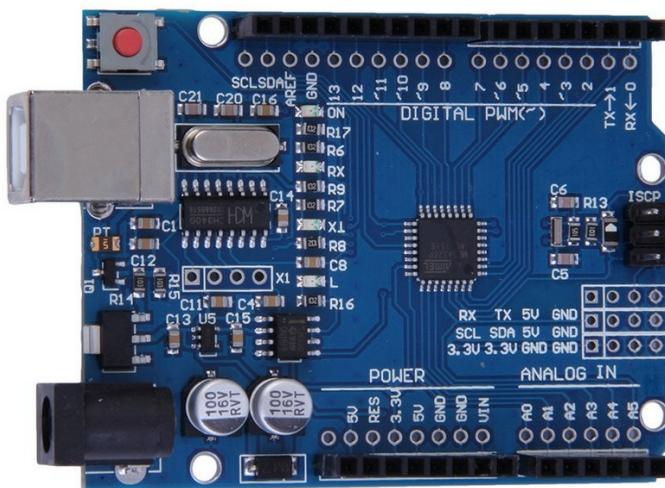
- Паметни уређај (паметни телефон, таблет, ...)
- Светлосна група - осветљење у свим просторијама
- Систем закључавања који подржава даљинско управљање
- Мотор који покреће систем закључавања
- Уређај за остваривање даљинске везе
- Микроконтролерски уређај за управљање целокупним системом
- Напајање за целокупни систем
- Ожичење целокупног система

Реални систем може захтевати још много помоћних компоненти али, с обзиром да се проектни задатак односи само на прототип једног таквог система, задржаћемо се на горе набројаним компонентама. У наредним поглављима ће бити детаљно приказан начин на који су поменуте компоненте уграђене у прототип система као и које су компоненте у питању. Такође ће бити детаљно објашњен начин на који је имплементирана сама мобилна апликација - софтверска реализација пројекта.

3 Опис коришћене технологије и компоненти

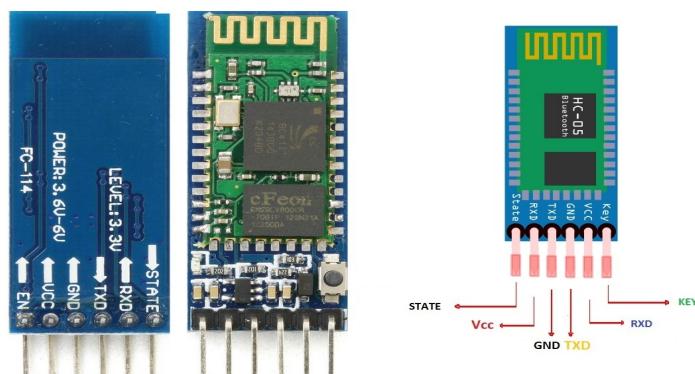
3.1 Опис хардверских компоненти

Микроконтролерски уређај за управљање прототипом система је [Arduino Uno R3](#). То је микроконтролерска плоча заснована на [ATmega328P](#). Има 16 дигиталних улазно/излазних пинова, 6 аналогних улаза, USB конекцију, прикључак за напајање итд. Садржи све што је потребно за подршку микроконтролеру. Уређај се преко USB кабла повезује на рачунар да би се испрограмирао при чему се и напаја преко истог кабла. Ако је потребно само напајање оно се, поред поменутог начина, може остварити и AC-DC адаптером или батеријом.



Слика 1: Arduino Uno R3

Уређај за остваривање даљинске (бежичне) везе је [HC-05 bluetooth модул](#) и представља једноставан за коришћење Bluetooth SPP (Serial Port Protocol) модул. Преко њега се може остварити двосмерна бежична функционалност тј. може се користити и као 'master' и као 'slave'. У оквиру овог пројекта користиће се само као 'slave' јер ће наредбе ићи само од апликације ка модулу а не и обрнуто.



Слика 2: HC-05 bluetooth модул

Светлосну групу у прототипу система чиниће шест светлећих диода (LED - Light Emitting Diode) црвене боје које ће да представљају осветљење у просторијама.



Слика 3: LED

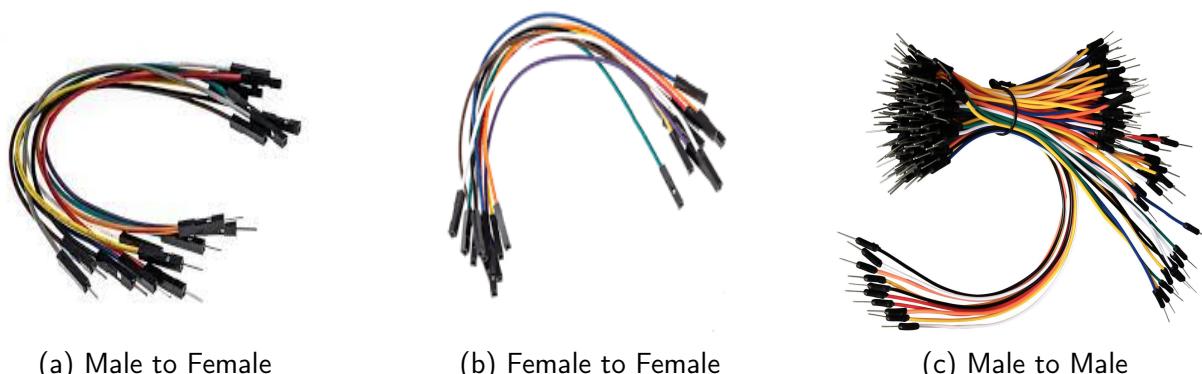
Мотор који покреће систем закључавања браве у овом пројекту је серво мотор ([Micro Servo SG90](#)) са једнокраким пропелером. Серво мотори су мотори високог обртног момента који се обично користе у роботици и неким другим апликацијама због чињенице да је лако контролисати њихову ротацију. Серво мотори имају зупчасту излазну осовину која се може електронски контролисати. Због контроле, за разлику од обичних DC мотора, серво мотори обично имају додатни пин поред два пина за напајање (VCC и GND) који је сигнални пин. Сигнални пин се користи за контролу серво мотора, окрећући његову осовину под било којим жељеним углом.



Слика 4: Micro Servo SG90

За ожичење су коришћене краткоспојне жицe са конекторима следећих типова:

- Female to Female
- Male to Female
- Male to Male



(a) Male to Female

(b) Female to Female

(c) Male to Male

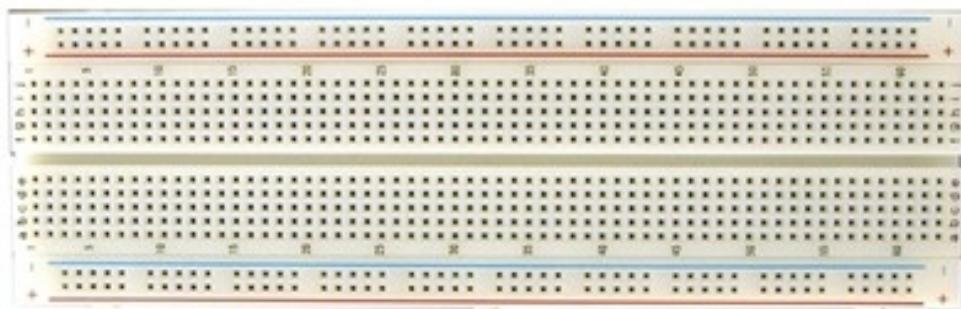
Слика 5: Жице са различитим комбинацијама конектора

За напајање се у оквиру реализованог пројекта користе батерија од 9 волти за напајање Arduino плоче и лежиште са три батерије од по 1.5 волти (тип AA) за напајање серво мотора, с тим да се у току реализације за напајање Arduino плоче користио USB каонектор преко кога се одвијало и програмирање микроконтролера.



Слика 6: Напајање: 3xAA 1.5V, 9V, USB конектор

Као простор за повезивање коришћена јеproto плоча (енг. [breadboard](#)) са 830 пинова. Једна таква је приказана на следећој слици:



Слика 7: Прото плоча

Паметни уређај на коме је инсталirана мобилна апликација у оквиру овог пројекта је паметни телефон са Android оперативним системом. Међутим, мобилна апликација у оквиру овог пројекта је имплементирана у Flutter мулти-платформском окружењу које омогућава израду апликација независно од оперативног система на коме се инсталира апликација, тако да би реализована апликација требала радити и на другим оперативним системима за паметне уређаје (нпр. iOS).

Претходно су приказане све компоненте које се налазе у хардверском делу реализованог пројекта и све се могу наћи у слободној прдаји. Начин повезивања свих компоненти биће објашњен и илустративно приказан у поглављу које се односи на сам начин реализације целокупног пројекта.

3.2 Софтверске потребе пројекта

У наставку ће бити набројани софтверски алати, радна окружења као и програмски језици који су били неопходни за реализацију читавог пројекта:

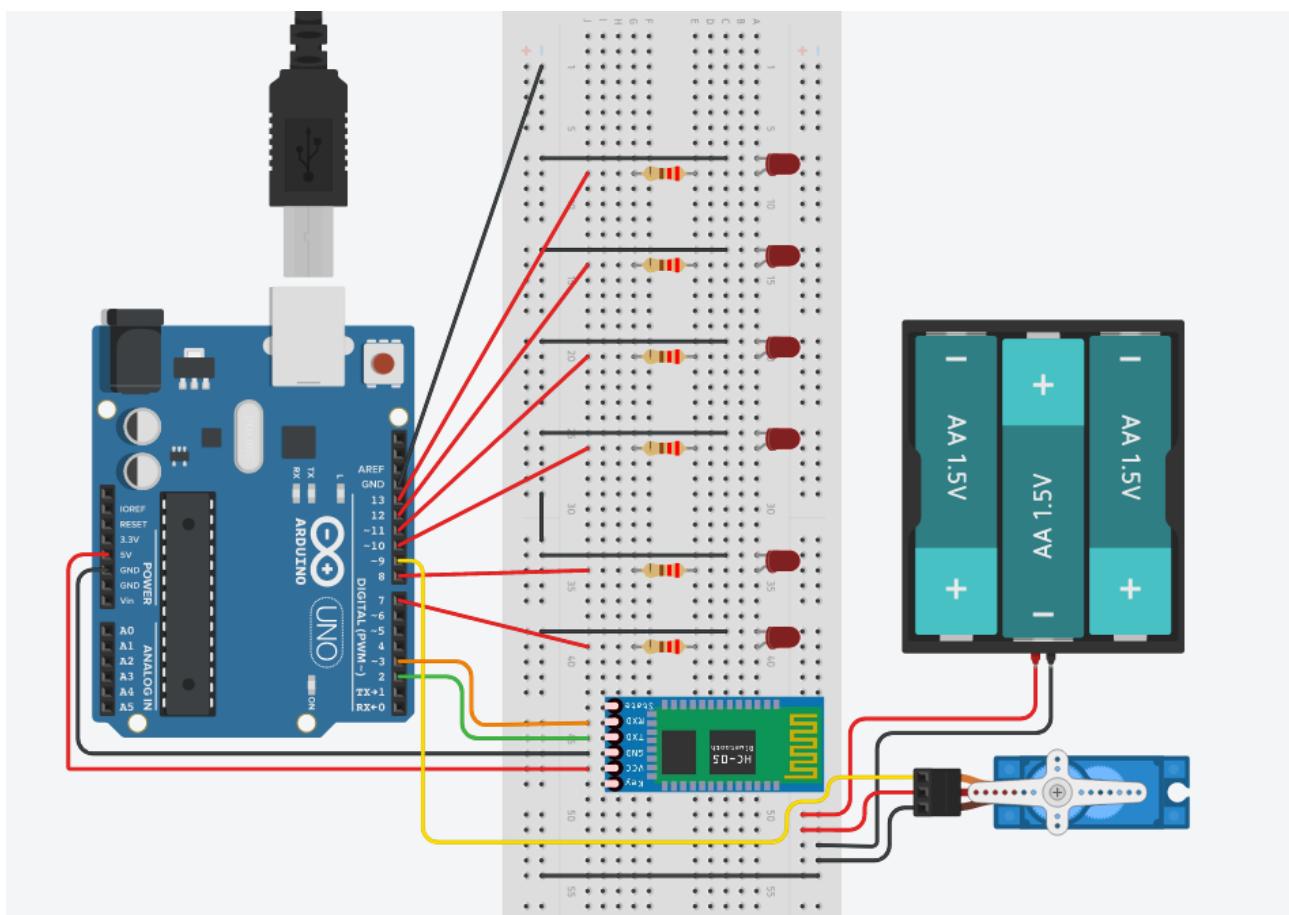
- [Arduino IDE 1.8.19](#) софтвер који омогућава компајлирање кода за Arduino и такође његово учитавање на плочу (енг. upload). Софтвер је бесплатан и доступан за све оперативне системе (Windows, Linux, Mac OS).
- [С програмски језик](#) - језик у коме се пише код за програмирање микроконтролера на Arduino плочи.
- [Flutter](#) - оквир за развој софтвера отвореног кода. Креиран је од стране Google-а. Користи се за развој мулти-платформских апликација за Android, iOS, Linux, Mac, Windows и интернет апликација (Web) и то из једне исте базе кода. Инсталација самог окружења је такође доступна за више оперативних система, а сам поступак инсталације и кораци који треба да се прате најбоље су објашњени у званичној [Flutter документацији](#).
- [Dart програмски језик](#) - језик који користи Flutter и у коме се пише код саме апликације. Инсталира се заједно са Flutter окружењем. Развио га је Google и намењен је развоју клијентских апликација, као што су интернет и мобилне апликације, а може се користити

и за израду сервер и десктоп апликација. У наредном поглављу биће приказани и објашњени поједини делови кода писаног у Dart програмском језику за потребе апликације у оквиру нашег пројекта.

- [Visual Studio Code](#) - уређивач изворног кода који је развио Microsoft за Windows, Linux и Mac OS. Софтвер је бесплатан и слободан за приватну или комерцијалну употребу. У нашем пројекту је коришћен за писање кода у оквиру Flutter окружења које се веома једноставно подешава за VS Code ако се прате [инструкције](#) које се такође налазе у оквиру званичне Flutter документације.
- [Latex](#) - коришћен за израду документације нашег пројекта. Представља описни језик и систем за припрему докумената. Latex је у широкој употреби у академији, а користи се за објављивање научних докумената у многим областима, укључујући математику, физику, информатику, статистику, економију, и политичке науке. Може се преузети са званичног [сајта](#), а може се користити и онлајн Latex уређивач при чemu не мора да се инсталира на рачунар. За израду ове документације коришћен је онлајн уређивач који се назива [Overleaf](#) и пружа заиста сјајан и прилагођен интерфејс и брзину генерсања документа.
- [Tinkercad](#) - онлајн уређивач који је у оквиру овог пројекта коришћен за цртање шеме повезивања хардверских компоненти. Све што је потребно јесте направити налог на овом сајту.

4 Начин повезивања хардверских компоненти

У претходном поглављу набројане су и описане све коришћене хардверске компоненте. У наставку ће бити приказана шема повезивања тих компоненти како би представљале функционалну целину када се изради апликација.



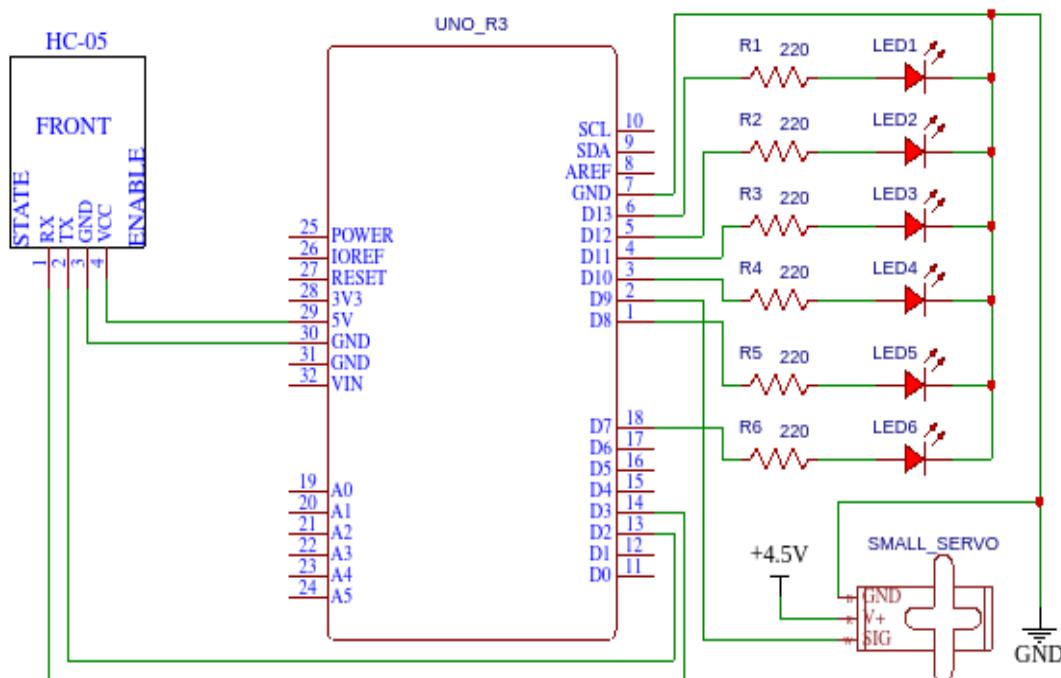
Слика 8: Шема са сликовитим приказом компоненти

На претходној шеми је приказан исправан начин повезивања компоненти. На шеми недостаје 9V батерија која напаја Arduino плочу, али је ту USB конектор који врши исту функцију. Разлог изосављања батерије јесте немогућност уређивача да прикаже Arduino плочу без USB конектора. Из сличног разлога је серво мотор приказан са пропелером који има два крака, а не један како је предвиђено у списку компоненти. Светлећих диода има шест и свака ће, како је већ поменуто, да служи као осветљење за једну просторију. То значи да је предвиђено да има шест просторија у макети куће, а оне су следеће:

- Дневни боравак (Living Room)
- Спаваћа соба (Bedroom)
- Дечија соба (Children's Room)

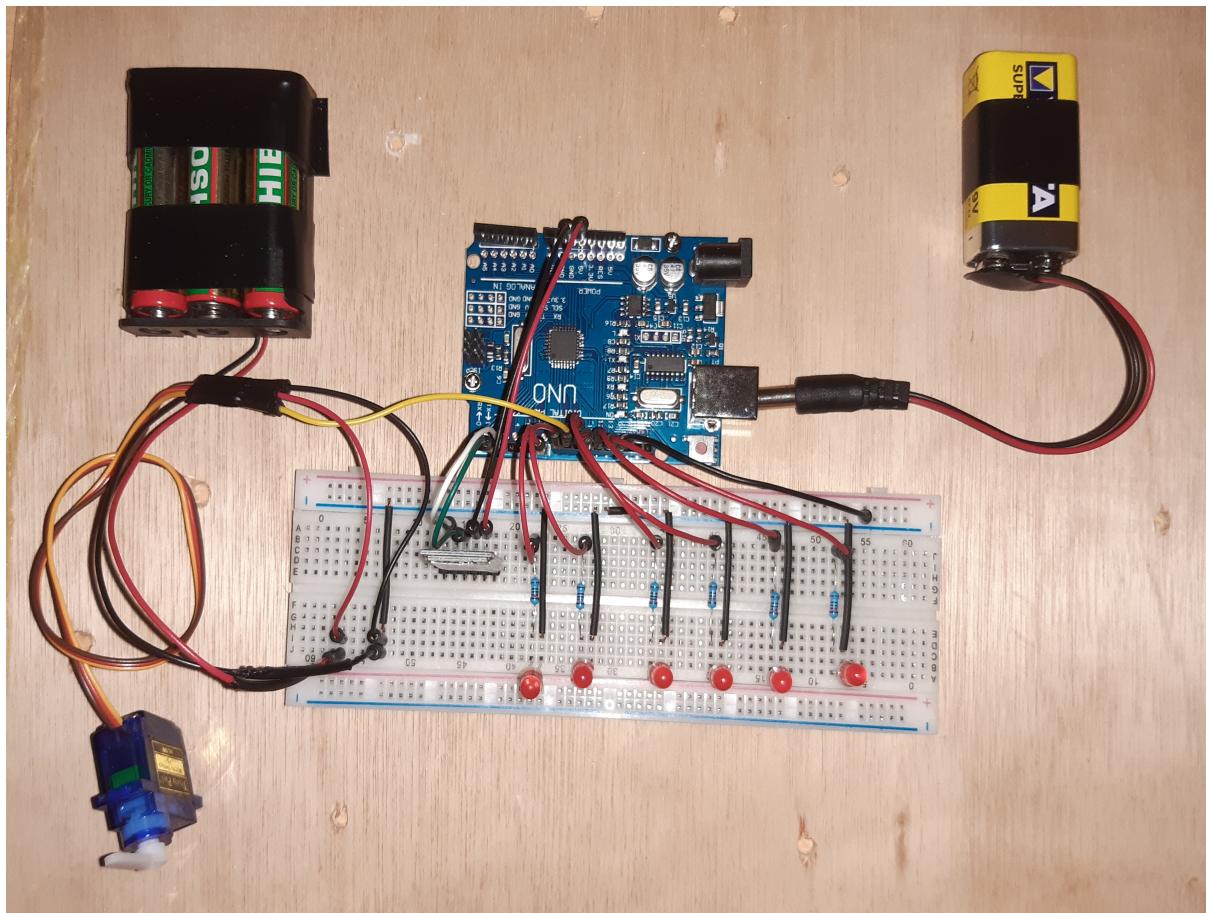
- Кухиња (Kitchen)
- Купатило (Bathroom)
- Ходник (Hallway)

Следи поједностављен приказ шеме на коме се јасно види види логика повезивања електронских компоненти. Шема је израђена помоћу [EasyEda](#) онлајн уређивача за шеме електронских компоненти (слично као Tinkercad или на нижем нивоу када је у питању изглед компоненти). Треба напоменути да и Tinkercad и EasyEda могу да послуже као симулатори електронских склопова, али су у оквиру овог пројекта коришћени само за израду шеме у циљу појашњења начина повезивања компоненти и нису рађене симулације.

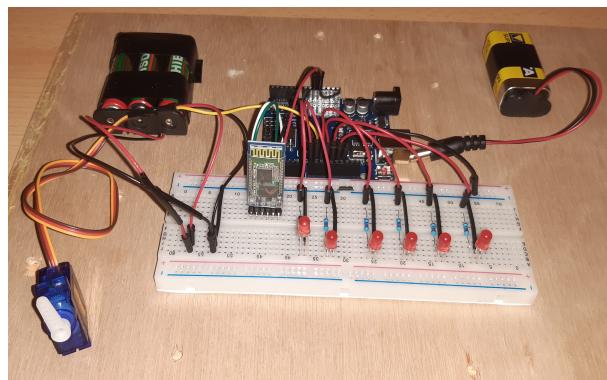


Слика 9: Поједностављен приказ шеме

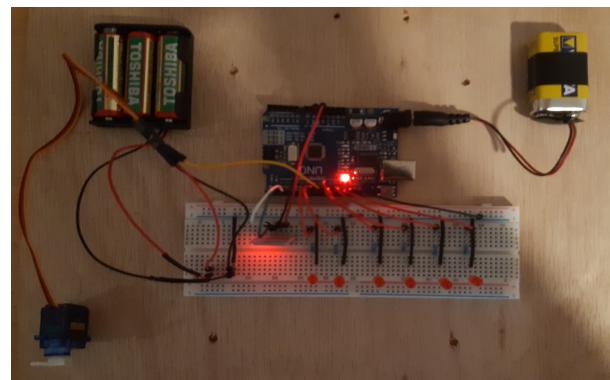
У следећем приказу (фотографија) јесте повезана шема свих реалних хардверских компоненти које су у саставу овог пројекта.



(a) Приказ одозго



(b) Приказ спреда



(c) Приказ - укључено

Слика 10: Фотографије израђене шеме повезивања компоненти

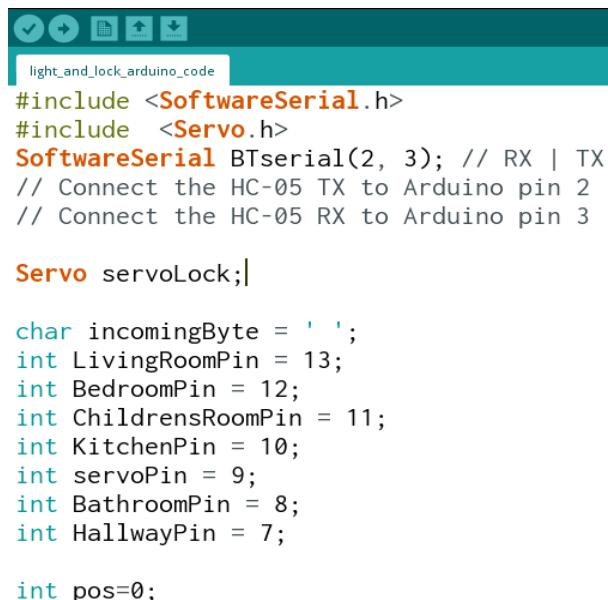
5 Софтверска реализација пројекта

5.1 Програмирање мокроконтролера

Како је већ речено, програмирање микроконтролера на Arduino плочи вршимо у окружењу Arduino IDE у програмском језику C. Код се налази у `light_and_lock_arduino_code.ino` датотеци. Да би се код учитао на Arduino потребно је повезати га са рачунаром помоћу USB конектора. Након покретања Arduino IDE окружења потребно је извршити следећа подешавања:

- Селектовати Tools > Board и одабрати Arduino Uno.
- Селектовати Tools > Port и одабрати USB порт на који је прикључен Arduino преко USB конектора (и нашем случају је то /dev/ttyUSB0).

У наставку ће бити објашњени најбитнији делови извornог кода:



```

#include <SoftwareSerial.h>
#include <Servo.h>
SoftwareSerial BTserial(2, 3); // RX | TX
// Connect the HC-05 TX to Arduino pin 2
// Connect the HC-05 RX to Arduino pin 3

Servo servoLock;

char incomingByte = ' ';
int LivingRoomPin = 13;
int BedroomPin = 12;
int ChildrensRoomPin = 11;
int KitchenPin = 10;
int servoPin = 9;
int BathroomPin = 8;
int HallwayPin = 7;

int pos=0;

```

Слика 11: Arduino изворни код 1

На самом почетку потребно је учитати библиотеку `SoftwareSerial` која омогућава серијску комуникацију са апликацијом, као и библиотеку `Servo` која омогућује управљање серво мотором. Након тога декларишемо Bluetooth везу тако што Arduino пинове 2 и 3 доделимо за TX и RX конекторе bluetooth модула, респективно. Затим се декларише променљива за серво мотор која је типа `Servo`. Променљива `incomingByte` типа `char` се користи да би се пријемни податак (наредба која долази са паметног уређаја) сместила унутар ове променљиве. Затим следе променљиве типа `int` помоћу којих касније дефинишемо које пинове развојног окружења користимо као излазне параметре. Помоћу њих вршимо контролу управљаних догађаја (осветљење и управљање бравом). Променљива `pos` је предвиђена за позицију серво мотора.

Затим следи метода `setup` која се позива само једном и то при извршавању кода тј. учитавању на Arduino:

```

void setup() {
    pinMode(LivingRoomPin, OUTPUT);
    pinMode(BedroomPin, OUTPUT);
    pinMode(ChildrensRoomPin, OUTPUT);
    pinMode(KitchenPin, OUTPUT);
    pinMode(BathroomPin, OUTPUT);
    pinMode(HallwayPin, OUTPUT);

    servoLock.attach(servoPin);
    digitalWrite(servoPin, LOW);

    digitalWrite(LivingRoomPin, LOW);
    digitalWrite(BedroomPin, LOW);
    digitalWrite(ChildrensRoomPin, LOW);
    digitalWrite(KitchenPin, LOW);
    digitalWrite(BathroomPin, LOW);
    digitalWrite(HallwayPin, LOW);

    // HC-05 default serial speed for commincation mode is 9600
    BTserial.begin(9600);
}

```

Слика 12: Arduino изворни код 2 - setup метода

Унутар методе `setup` позивају се `pinMode` функције преко којих се претходно дефинисани пинови означавају као излазни. Такође, врши се повезивање контроле серво мотора на пин предвиђен за то преко функције `attach`, а то је пин 9. Након тога позива се функција `digitalWrite` којом се на све пинове који ће да се користе шаље низак напон као почетна конфигурација. На крају се поставља и иницијализује серијска bluetooth комуникација као њена брзина која је изражена у `baud rate`-овима.

Следи функција `loop` која се позива стално након учитавања кода на Arduino, наравно све док се напајање не прекине и по томе се разликује од `setup` функције која се позива само једном.

```

void loop() {
    if (BTserial.available()) {
        incomingByte = BTserial.read();

        if (incomingByte == 'a') {
            digitalWrite(LivingRoomPin, HIGH);
        }
        if (incomingByte == 'b') {
            digitalWrite(LivingRoomPin, LOW);
        }
        if (incomingByte == 'c') {
            digitalWrite(BedroomPin, HIGH);
        }
        if (incomingByte == 'd') {
            digitalWrite(BedroomPin, LOW);
        }
        if (incomingByte == 'e') {
            digitalWrite(ChildrensRoomPin, HIGH);
        }
        if (incomingByte == 'f') {
            digitalWrite(ChildrensRoomPin, LOW);
        }
        if (incomingByte == 'g') {
            digitalWrite(KitchenPin, HIGH);
        }
        if (incomingByte == 'h') {
    
```

```
    digitalWrite(KitchenPin, LOW);
}
if (incomingByte == 'i') {
    digitalWrite(BathroomPin, HIGH);
}
if (incomingByte == 'j') {
    digitalWrite(BathroomPin, LOW);
}
if (incomingByte == 'k') {
    digitalWrite(HallwayPin, HIGH);
}
if (incomingByte == 'l') {
    digitalWrite(HallwayPin, LOW);
}
if (incomingByte == 'm') {
    digitalWrite(servoPin, HIGH);
    pos = map(90, 0, 180, 180, 0);
    servoLock.write(pos);
    digitalWrite(servoPin, LOW);
}
if (incomingByte == 'n') {
    digitalWrite(servoPin, HIGH);
    pos = map(0, 0, 180, 180, 0);
    servoLock.write(pos);
    digitalWrite(servoPin, LOW);
}
}
```

Слика 13: Arduino изворни код 3 - loop метода

Унутар методе `loop` се на почетку преко функције `available` проверава да ли је број бајтова (карактера) доступних за читање преко софтверског серијског порта већи од 0. Ако јесте, тај карактер се чита преко функције `read` и смешта у променљиву `incomingByte` и то ће нам представљати податак који је послат од стране мобилне апликације тј. 'master'-а ка `bluetooth` модулу тј. 'slave'-у. Затим се врши провера који је то карактер.

За потребе овог пројекта усвојени су карактери од 'а' до 'н' по абецедном реду. Логика је та да напр. ако стигне наредба 'а', преко функције digitalWrite се на пин који је предвиђен за контролу осветљења у дневном боравку (LivingRoomPin) шаље висок напон чиме се диода засветли, и светли све док не стигне низак напон на тај исти пин. Низак напон на тај исти пин стиче кад стигне наредба 'б'. Исти процес се одвија за све остале пинове који су задужени за осветљење само се шаљу други карактери као наредбе.

Оно што се разликује јесте како се реагује на наредбе 'm' и 'n' које служе као наредбе да се покрене серво мотор тј. да се откључа брава ако је примљена наредба 'm', а да се закључка ако је примљена наредба 'n'. Ако се прими било која од ове две наредбе, поставља се висок напон на пин који је предвиђен за контролу серво мотора (`servoPin`). Затим следи постављање жељене вредности за позицију пропелера серво мотора преко функције `map`. Након тога се преко функције `write` подешава позиција серво мотора и шаље се низак напон на `servoPin`.

Када се заврши са програмирањем, треба изабрати опцију Sketch > Verify/Compile што ће покренути процес компајлирања кода и уједно га верификовати (проверти да ли је исправан). Ако је код исправан, одради се следећи поступак: одабере се Sketch > Upload, чиме се код учита тј. инсталира на микроконтролерску плочу (Arduino Uno) и тиме је процес програмирања микроконтролера за овај проекат завршен.

5.2 Програмирање мобилне апликације

Као што је већ поменуто, мобилна апликација је израђена у Flutter окружењу при чему се користио Visual Studio Code уређивач текста. Након подешавања Flutter окружења у VS Code-у потребно је направити празан Flutter пројекат (апликацију) по [упутству](#) из документације.

При изради пројекта апликација је тестирана на Android паметном телефону. За инсталацију апликације на уређају је потребан USB кабл који ће да повезује рачунар и паметни телефон. Да би се проверило да ли Flutter препознаје уређај потребно је отићи на терминал у оквиру VS Code окружења, ући у директоријум пројекта (у нашем случају назив пројекта је `light_and_lock_bluetooth_control_flutter_app`) и унети команду `flutter devices`. Ако на стандардном излазу испише конектовани уређај, значи да ће апликација моћи да се инсталира уколико нема грешке. Да би се апликација инсталирала потребно је унети команду `flutter run`.

Да би реализована апликација радила радила исправно, потребно је извршити подешавања у две датотеке. То су `android/app/build.gradle` где је потребно пронаћи `defaultConfig` секцију и подесити `minSdkVersion` на 19:

```
defaultConfig {
    // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).
    applicationId "com.example.light_and_lock_bluetooth_control_flutter_app"
    minSdkVersion 19
    targetSdkVersion flutter.targetSdkVersion
    versionCode flutterVersionCode.toInt()
    versionName flutterVersionName
}
```

Слика 14: Модификација `build.gradle` датотеке

Након тога је потребно креирати `images` директоријум унутар директоријума Flutter пројекта и у њега убацити слике које ће се користити при изради графичког интерфејса апликације:



(a) `light_off.png`



(b) `light_on.png`



(c) `locked.png`



(d) `unlocked.png`

Слика 15: Илустрације потребне за израду графичког интерфејса апликације

Да би се слике исправно учитале потребно је отворити pubspec.yaml датотеку и у њој пронаћи flutter: assets: секцију и додати путање до слика као на примеру испод:

```

54 # The following section is specific to Flutter.
55 flutter:
56
57 # The following line ensures that the Material Icons font is
58 # included with your application, so that you can use the icons in
59 # the material Icons class.
60 uses-material-design: true
61
62 # To add assets to your application, add an assets section, like this:
63 # assets:
64 #   - images/a_dot_burr.jpeg
65 #   - images/a_dot_ham.jpeg
66 assets:
67   - images/light_on.png
68   - images/light_off.png
69   - images/locked.png
70   - images/unlocked.png
71
72 # An image asset can refer to one or more resolution-specific "variants", see
73 # https://flutter.dev/assets-and-images/#resolution-aware.

```

Слика 16: Модификација pubspec.yaml датотеке - учитавање слика

Поред тога, у pubspec.yaml датотеци потребно је пронаћи секцију dependencies: и додати зависности за bluetooth серијску комуникацију и Toast поруке:

```

23 # Dependencies specify other packages that your package needs in order to work.
24 # To automatically upgrade your package dependencies to the latest versions
25 # consider running 'flutter pub upgrade --major-versions'. Alternatively,
26 # dependencies can be manually updated by changing the version numbers below to
27 # the latest version available on pub.dev. To see which dependencies have newer
28 # versions available, run 'flutter pub outdated'.
29 dependencies:
30   flutter:
31     | sdk: flutter
32
33
34 # The following adds the Cupertino Icons font to your application.
35 # Use with the CupertinoIcons class for iOS style icons.
36 cupertino_icons: ^1.0.2
37 flutter_bluetooth_serial: ^0.4.0
38 fluttertoast: ^8.0.7

```

Слика 17: Модификација pubspec.yaml датотеке - dependencies

Најбитнија датотека у којој се пише комплетан код апликације јесте lib/main.dart и у наставку ће се проћи кроз објашњење најбитнијих делова кода који се ту налази.

```

1 import 'dart:convert';
2 import 'dart:typed_data';
3 import 'dart:async';
4
5 import 'package:flutter/material.dart';
6 import 'package:flutter_bluetooth_serial/flutter_bluetooth_serial.dart';
7 import 'package:fluttertoast/fluttertoast.dart';

```

Слика 18: Dart код - 1. део

На самом почетку се врши учитавање потребних библиотека. Најбитније су:

- `material` - за исправно функционисање widget-а (графичких компоненти у оквиру апликације)
- `flutter_bluetooth_serial` - за комуникацију са bluetooth модулом
- `fluttertoast` - омогућава испис Toast порука у оквиру апликације

```

Run | Debug | Profile
10 void main() {
11   runApp(const MyApp());
12 }
13
14 class MyApp extends StatelessWidget {
15   const MyApp({Key? key}) : super(key: key);
16
17   // This widget is the root of application.
18   @override
19   Widget build(BuildContext context) {
20     return MaterialApp(
21       title: 'Flutter App',
22       theme: ThemeData(
23         primarySwatch: Colors.blue,
24       ), // ThemeData
25       home: const MyHomePage(title: 'Light and Lock Remote Control App'),
26     ); // MaterialApp
27   }
28 }
```

Слика 19: Dart код - 2. део

Након учитавања библиотека позвали смо главну функцију `main()`. Унутар ње се позива уgraђена функција `runApp` чији је аргумент објекат `MyApp` класе која је креирана испод и која наслеђује `StatelessWidget` уgraђену класу. Унутар ње се предефинише `build` метода која се позива при креирању објекта класе и која враћа објекат типа `Widget`, а то ће у овом случају бити `MaterialApp` уgraђени `widget` која ће бити на врху хијерархије осталих `widget-а` у апликацији. У оквиру њега налази се `MyHomePage` `widget` који представља почетну и једину страницу апликације у овом пројекту.

```

30   |
31   class MyHomePage extends StatefulWidget {
32     const MyHomePage({Key? key, required this.title});
33
34     final String title;
35
36     @override
37     State<MyHomePage> createState() => _MyHomePageState();
38 }
```

Слика 20: Dart код - 3. део

Класа MyHomePage наслеђује StatefulWidget. Разлика између StatefulWidget и StatelessWidget је та да се у StatefulWidget-у може динамички мењати садржај у току коришћења апликације што омогућава израду ефикаснијег и квалитетнијег корисничког интерфејса. Да би се имплементирало такво понашање потребно је да се предефинише уградена функција createState() како би позвала објекат типа State који је у овом случају _MyHomePageState().

```

39  class _MyHomePageState extends State<MyHomePage> {
40
41    String _bulbImgPathLivingRoom = "images/light_off.png";
42    String _bulbImgPathBedroom = "images/light_off.png";
43    String _bulbImgPathChildrensRoom = "images/light_off.png";
44    String _bulbImgPathKitchen = "images/light_off.png";
45    String _bulbImgPathBathroom = "images/light_off.png";
46    String _bulbImgPathHallway = "images/light_off.png";
47    String _padlockImgPathFrontDoor = "images/locked.png";
48
49    Color _clrButtonLivingRoom = Colors.green;
50    Color _clrButtonBedroom = Colors.green;
51    Color _clrButtonChildrensRoom = Colors.green;
52    Color _clrButtonKitchen = Colors.green;
53    Color _clrButtonBathroom = Colors.green;
54    Color _clrButtonHallway = Colors.green;
55    Color _clrButtonFrontDoor = Colors.green;
56
57    String _txtButtonLivingRoom = "TURN ON";
58    String _txtButtonBedroom = "TURN ON";
59    String _txtButtonChildrensRoom = "TURN ON";
60    String _txtButtonKitchen = "TURN ON";
61    String _txtButtonBathroom = "TURN ON";
62    String _txtButtonHallway = "TURN ON";
63    String _txtButtonFrontDoor = "UNLOCK";
64
65    late BluetoothConnection connection;
66
67    String _connectedYesNo = "Loading...";
68    Color _colorConnectedYesNo = Colors.black;
69    String _txtButtonCheckReload = "CHECK";
70
71    _MyHomePageState(){
72      _connect();
73    }
74
75    bool get isConnected => (connection.isConnected);

```

Слика 21: Dart код - 4. део

Класа _MyHomePageState наслеђује State класу и садржи највећи део функционалности апликације. Све у наставку што садржи main.dart датотека јесте имплементација поменуте класе.

На самом почетку иницијализују се променљиве за путање слика, боје дугмади и текст дугмади на своје почетне вредности.

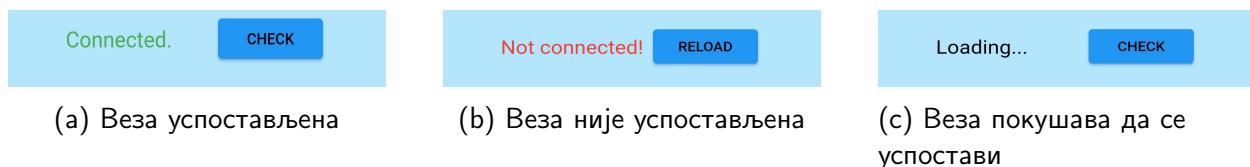
Затим се дефинише променљива connection типа BluetoothConnection која ће представљати објекат везе са bluetooth модулом. У конструктору класе се позива само функција _connect() која ће бити објашњена у наставку. Променљива isConnected ће садржати логичку вредност да ли је веза остварена или не, тако што ће се у њу, када год се користи, сместити резултат позива функције connection.isConnected.

```

77 Future<void> _connect() async {
78   try {
79     connection = await BluetoothConnection.toAddress("00:21:07:00:07:EE");
80     Fluttertoast.showToast( msg: 'Connected to the bluetooth device', );
81     print('Connected to the bluetooth device');
82     setState(() {
83       _connectedYesNo = "Connected.";
84       _colorConnectedYesNo = Colors.green;
85       _txtButtonCheckReload = "CHECK";
86     });
87   }
88   catch (exception) {
89     try {
90       if (isConnected){
91         Fluttertoast.showToast( msg: 'Already connected to the device', );
92         print('Already connected to the device');
93         setState(() {
94           _connectedYesNo = "Connected.";
95           _colorConnectedYesNo = Colors.green;
96           _txtButtonCheckReload = "CHECK";
97         });
98       }
99       else{
100         Fluttertoast.showToast( msg: 'Cannot connect, exception occurred', );
101         print('Cannot connect, exception occurred');
102         setState(() {
103           _connectedYesNo = "Not connected!";
104           _colorConnectedYesNo = Colors.red;
105           _txtButtonCheckReload = "RELOAD";
106         });
107       }
108     }
109     catch (e){
110       Fluttertoast.showToast( msg: 'Cannot connect, probably not initialized connection', );
111       print('Cannot connect, probably not initialized connection');
112       setState(() {
113         _connectedYesNo = "Not connected!";
114         _colorConnectedYesNo = Colors.red;
115         _txtButtonCheckReload = "RELOAD";
116       });
117     }
118   }
119 }
```

Слика 22: Dart код - 5. део

Функција `_connect()` је функција у којој се иницијализује bluetooth веза тако што се се позива функција `toAddress` из класе `BluetoothConnection`. Њој је потребно да се проследи као аргумент MAC адреса уређаја на који се повезује, а то је у овом случају MAC адреса bluetooth модула и она је следећа: 00:21:07:00:07:EE. Затим се корисник обавештава да је веза успостављена и подешава се динамички кориснички интерфејс тако што се позива уградена функција `setState` која се позива сваки пут када је потребно променити карактеристике неког widget-a динамички у току коришћења. У оквиру ње се реиницијализују променљиве које су коришћене од стране појединих widget-а да дефинишу одређене карактеристике. Такође, у оквиру функције `_connect()` се хватају и одређени изузети и то за случај да је корисник већ повезан, да се појавила грешка или да веза не може бити успостављена из разлога што неки од уређаја нема укључен bluetooth. Следи приказ графичког интерфејса ако је веза успостављена, ако није и ако покушава да се успостави:



Слика 23: Приказ графичког интерфејса за различита стања bluetooth везе

```
-- 
121 void waitLoading(){
122     setState(() {
123         _connectedYesNo = "Loading...";
124         _colorConnectedYesNo = Colors.black;
125         _txtButtonCheckReload = "CHECK";
126     });
127 }
128
129 void _reloadOrCheck(){
130     waitLoading();
131     _connect();
132 }
133
134 Future<void> _sendData(String data) async {
135     connection.output.add(Uint8List.fromList(utf8.encode(data))); // Sending data
136     await connection.output.allSent;
137 }
```

Слика 24: Dart код - 6. део

Функција `waitLoading` служи за подешавање графичких компоненти које се приказују док bluetooth веза покушава да се успостави. Унутар функције `_reloadOrCheck` позивају се функције `waitLoading` и `_connect` и служи да би уређај покушао поново да се повеже преко bluetooth-а или да провери постојећу bluetooth везу.

Најбитнија од ових функција је `_sendData` која као аргумент прима објекат типа `String`,

а прослеђиваће јој се само један карактер који ће да служи као наредба коју прима bluetooth модул и прослеђује је микроконтролеру који слуша и извршава наредбе.

```

139 void _setLightOrLockState(String _roomOrDoorType){
140
141     if(_connectedYesNo == "Connected."){
142
143         setState(() {
144
145             if (_roomOrDoorType == "Living Room"){
146                 if (_bulbImgPathLivingRoom == "images/light_off.png" && _clrButtonLivingRoom == Colors.green && _txtButtonLivingRoom == "TURN ON"){
147                     _bulbImgPathLivingRoom = "images/light_on.png";
148                     _clrButtonLivingRoom = Colors.red;
149                     _txtButtonLivingRoom = "TURN OFF";
150                     _sendData("a");
151                 }
152                 else{
153                     _bulbImgPathLivingRoom = "images/light_off.png";
154                     _clrButtonLivingRoom = Colors.green;
155                     _txtButtonLivingRoom = "TURN ON";
156                     _sendData("b");
157                 }
158             }
159         }  

229     }  

230     else if (_roomOrDoorType == "Front Door"){
231         if (_padlockImgPathFrontDoor == "images/locked.png" && _clrButtonFrontDoor == Colors.green && _txtButtonFrontDoor == "UNLOCK"){
232             _padlockImgPathFrontDoor = "images/unlocked.png";
233             _clrButtonFrontDoor = Colors.red;
234             _txtButtonFrontDoor = "LOCK";
235             _sendData("m");
236         }
237         else{
238             _padlockImgPathFrontDoor = "images/locked.png";
239             _clrButtonFrontDoor = Colors.green;
240             _txtButtonFrontDoor = "UNLOCK";
241             _sendData("n");
242         }
243     });
244 }
245 else{
246     Fluttertoast.showToast( msg: 'Cannot send data!\nYou are not connected.', );
247 }
248 }
```

Слика 25: Dart код - 7. део

Функција `_setLightOrLockState` позива се кад год се притисне неко од дугмади TURN ON, TURN OFF, LOCK, UNLOCK у графичком корисничком интерфејсусу. Као аргумент јој се прослеђује објекат типа String који представља назив просторије у којој се контролише осветљење или назив улаза на коме се врши контрола браве. На самом почетку унутар функције се проверава да ли је успостављена bluetooth веза и ако није корисник се о томе обавештава преко Toast поруке. Ако је веза успостављена позива се функција `setState` коју смо већ поменули. У оквиру њеног позива врши се провера који је String прослеђен функцији `_setLightOrLockState` као аргумент. На слици изнад је приказан само случај

када је у питању 'Living Room'. Затим се проверава да ли вредности променљивих указују на то да је светло у тој просторији (у овом случају дневном боравку) искључено и ако јесте, те исте променљиве ће да добију вредности које указују на упаљено светло и доћи ће до промене у графичком корисничком интерфејсу, а такође се преко bluetooth везе шаље наредба 'a' која значи да светло треба да се упали. Промене у графичком корисничком интерфејсу су приказане на следећој слици:



(a) Светло угашено

(b) Светло упаљено

Слика 26: Изглед графичких компоненти за случај упаљеног и угашеног светла у дневном боравку

Ако променљиве не указују на то да је светло упаљено то значи да је светло угашено и да га треба упалити па се ради супротан процес, а преко bluetooth везе се шаље наредба 'b' која подразумева гашење светла. На слици изнад није приказан код за случај сваке просторије јер је логика иста као за случај дневног боравка. Оно што је битно напоменути јесте да се увек шаљу различите наредбе (карактери) преко bluetooth везе за сваку операцију посебно тј. ићи ће редом a, b, c, d, ..., m, n. На слици испод приказан је и случај ако је функцији _setLightOrLockState као аргумент прослеђен стринг 'Front Door' који подразумева контролу браве. За откључавање браве шаље се наредба 'm', а за закључавање наредба (податак, карактер) 'n'. Следи приказ графичких компоненти апликације за случај закључчане и откључчане браве:



(a) Брава закључчана

(b) Брава откључчана

Слика 27: Изглед графичких компоненти за случај закључчане и откључчане браве

На слици испод је приказана имплементација функције _buildRow која враћа widget типа Container и служи да генерише графичке компоненте за један објекат којим се управља (брава, светло у свим просторијама посебно). Као аргументе јој је потребно проследити:

- String _roomOrDoorType - тип објекта којим ће се управљати тј. текст који ће бити исписан у средини widget-a.
- String _imagePath - слика која ће бити смештена са леве стране widget-a Container.
- Color _clrButton - боја дугмета које се налази унутар widget-a.

- String _txtButton - текст лабела дугмета која указује на то коју акцију дугме извршава.

```

250 Widget _buildRow(String _roomOrDoorType, String _imagePath, Color _clrButton, String _txtButton) {
251   return Container(
252     margin: const EdgeInsets.only(left: 10.0, right: 10.0, top: 10.0),
253     padding: const EdgeInsets.fromLTRB(0.0, 10.0, 10.0, 10.0),
254     decoration: const BoxDecoration(
255       color: Colors.white,
256       borderRadius: BorderRadius.all(Radius.circular(10))
257     ), // BoxDecoration
258     child: Row(
259       children: [
260         Image.asset(
261           _imagePath,
262           height: 50.0,
263           width: 50.0,
264         ), // Image.asset
265         Expanded(child: Text(_roomOrDoorType, style: const TextStyle(fontSize: 20))),
266         SizedBox(
267           width: 100.0,
268           child: ElevatedButton(
269             onPressed: () { _setLightOrLockState(_roomOrDoorType); },
270             child: Text(
271               _txtButton,
272               style: const TextStyle(color: Colors.black),
273             ), // Text
274             style: ButtonStyle(backgroundColor: MaterialStateProperty.all(_clrButton)),
275           ), // ElevatedButton
276         ), // SizedBox
277       ],
278     ), // Row
279   ); // Container
280 }

```

Слика 28: Dart код - 8. део

Пр. Ако претходно приказану функцију позовемо на следећи начин:

```
_buildRow("Bedroom", _bulbImgPathBedroom, _clrButtonBedroom, _txtButtonBedroom)
```

тада ће се генерисати следећа графичка компонента (widget):



Слика 29: Container widget за приказ једног предмета управљања - светло у спаваћој соби

Напомена: Још увек се налазимо у имплементацији класе `_MyHomePageState`.

У наставку је приказана предефинисана функција build у оквиру класе _MyHomePageState која враћа Scaffold widget који ће садржати главни део апликације тј. све оне графичке компоненте преко којих корисник може да управља хардверским компонентама и bluetooth везом.

```

283     @override
284     Widget build(BuildContext context) {
285         return Scaffold(
286             backgroundColor: Colors.lightBlue[100],
287             appBar: AppBar(
288                 title: Text(widget.title),
289             ), // AppBar
290             body: Container(
291                 color: Colors.lightBlue[100],
292                 child: SingleChildScrollView(
293                     child: Column(
294                         children: [
295                             _buildRow("Living Room", _bulbImgPathLivingRoom, _clrButtonLivingRoom, _txtButtonLivingRoom),
296                             _buildRow("Bedroom", _bulbImgPathBedroom, _clrButtonBedroom, _txtButtonBedroom),
297                             _buildRow("Children's Room", _bulbImgPathChildrensRoom, _clrButtonChildrensRoom, _txtButtonChildrensRoom),
298                             _buildRow("Kitchen", _bulbImgPathKitchen, _clrButtonKitchen, _txtButtonKitchen),
299                             _buildRow("Bathroom", _bulbImgPathBathroom, _clrButtonBathroom, _txtButtonBathroom),
300                             _buildRow("Hallway", _bulbImgPathHallway, _clrButtonHallway, _txtButtonHallway),
301                             _buildRow("Front Door", _padlockImgPathFrontDoor, _clrButtonFrontDoor, _txtButtonFrontDoor),
302                             Container(
303                                 margin: const EdgeInsets.only(left: 10.0, right: 10.0, top: 10.0),
304                                 padding: const EdgeInsets.fromLTRB(45.0, 10.0, 50.0, 10.0),
305                                 decoration: BoxDecoration(
306                                     color: Colors.lightBlue[100],
307                                 ), // BoxDecoration
308                                 child: Row(
309                                     children: [
310                                         Expanded(child: Text(_connectedYesNo, style: TextStyle(fontSize: 20, color: _colorConnectedYesNo))),
311                                         SizedBox(
312                                             width: 100.0,
313                                             child: ElevatedButton(
314                                                 onPressed: _reloadOrCheck,
315                                                 child: Text(
316                                                     _txtButtonCheckReload,
317                                                     style: const TextStyle(color: Colors.black),
318                                                 ), // Text
319                                                 ), // ElevatedButton
320                                             ), // SizedBox
321                                             1,
322                                         ), // Row
323                                         ), // Container
324                                         ],
325                                         ), // Column
326                                         ), // SingleChildScrollView

```

Слика 30: Dart код - 9. део

Овим смо дошли до краја самог main.dart кода тј. имплементација мобилне апликације је завршена.

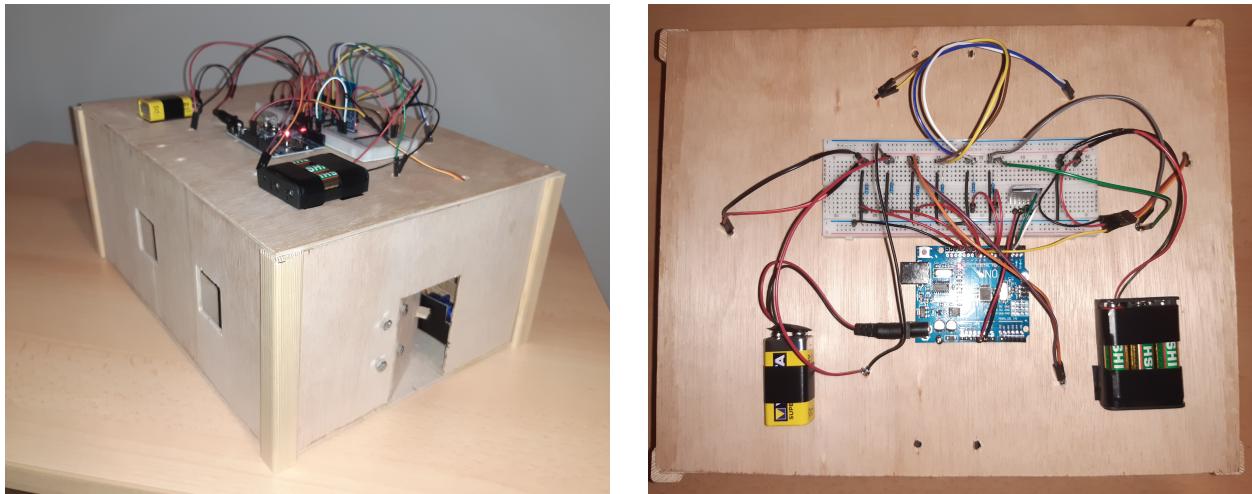
6 Израда макете куће

Макета куће која је потребна да би се употребнио прототип реалног система израђена је од комада шперплоче и изгледа као на следећим сликама:



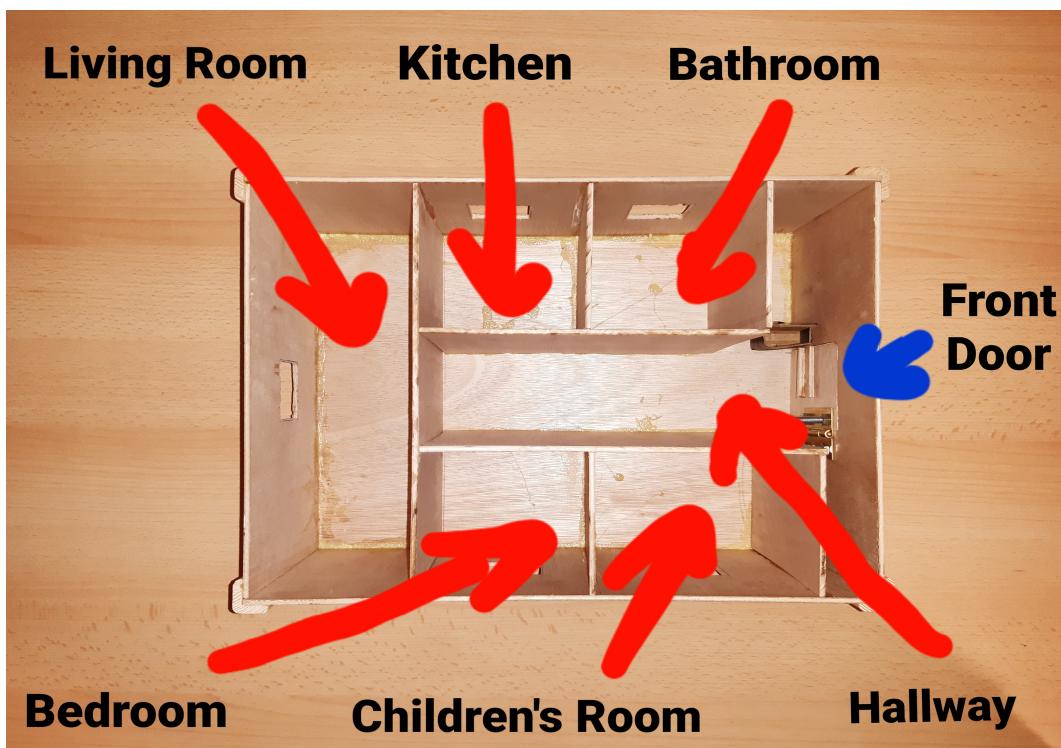
Слика 31: Изглед макете куће споља

Хардверске компоненте пројекта у оквиру макете куће налазе се на тавану и одатле су диоде и мотор спреведени на одговарајућа места. Кров макете може да се подигне како би се виделе хардверске компоненте које се налазе испод:



Слика 32: Положај хардверских компоненти на макети куће

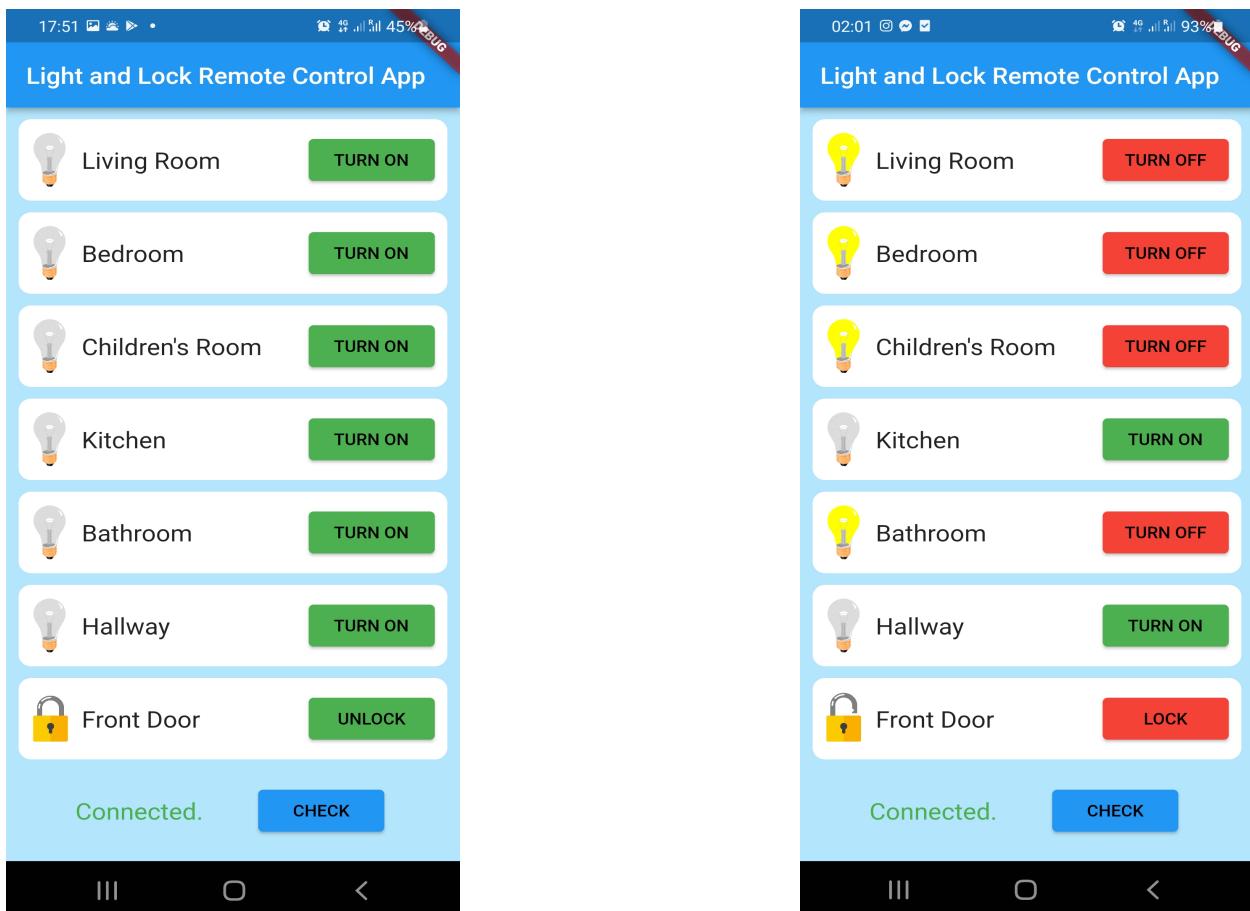
Плафон на коме се налазе хардверске компоненте такође може да се подигне да би се видео распоред просторија унутар макете куће. Направљене су преграде од шперплоче које имитирају више одвојених просторија:



Слика 33: Распоред просторија унутар макете куће - поглед одозго

7 Изглед и демонстрација рада апликације

Мобилна апликација имплементирана у оквиру овог пројекта је једноставна са више компоненти које динамички мењају свој изглед у току коришћења. Следи приказ два screenshot-а покренуте апликације:



Слика 34: Изглед апликације након покретања и у току коришћења

ДЕМОНСТРАЦИЈА РАДА АПЛИКАЦИЈЕ:

<https://youtu.be/4k8WbcMCyuA>

Github страница пројекта:

https://github.com/djoto/Flutter-Arduino_light_and_lock_bluetooth_control

8 Литература

- The Arduino Projects Book, Scott Fitzgerald and Michael Shiloh, Torino, Italy, September 2012
- http://moodle.fink.rs/pluginfile.php/36385/mod_resource/content/1/PMA%20-%20Dokumentacija.pdf - ПМА Документација, Бобан Срећковић, Крагујевац 2019
- <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontroller-Datasheet.pdf> - ATmega328P спецификација
- https://components101.com/sites/default/files/component_datasheet/HC-05-20Datasheet.pdf - HC-05 спецификација
- http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf - Micro Servo SG90 спецификација
- https://en.wikipedia.org/wiki/Light-emitting_diode
- <https://www.arduino.cc/en/software>
- [https://en.wikipedia.org/wiki/C_\(programming_language\)](https://en.wikipedia.org/wiki/C_(programming_language))
- [https://en.wikipedia.org/wiki/Flutter_\(software\)](https://en.wikipedia.org/wiki/Flutter_(software))
- [https://en.wikipedia.org/wiki/Dart_\(programming_language\)](https://en.wikipedia.org/wiki/Dart_(programming_language))
- <https://docs.flutter.dev/get-started/install>
- <https://code.visualstudio.com/>
- <https://docs.flutter.dev/get-started/editor?tab=vscode>
- <https://en.wikipedia.org/wiki/LaTeX>
- <https://www.latex-project.org/get/>
- <https://www.overleaf.com/project>
- <https://www.tinkercad.com/dashboard>
- <https://easyeda.com/>
- <https://www.arduino.cc/en/Reference/softwareSerial>
- <https://www.arduino.cc/reference/en/libraries/servo/>
- <https://en.wikipedia.org/wiki/Baud>
- <https://www.arduino.cc/reference/en/language/functions/math/map/>
- <https://docs.flutter.dev/get-started/test-drive?tab=vscode>

- https://pub.dev/packages/flutter_bluetooth_serial
- [https://api.flutter.dev/flutter/widgets/State setState.html](https://api.flutter.dev/flutter/widgets/State	setState.html)
- <https://stackoverflow.com/>
- <https://youtu.be/4k8WbcMCyuA> - YouTube линк демонстрације рада апликације
- https://github.com/djoto/Flutter-Arduino_light_and_lock_bluetooth_control
- Github страница пројекта