Upravljanje rasvjetom i električnom bravom korištenjem mobilnih uređaja i Bluetooth komunikacije

Korunić, Danijel

Undergraduate thesis / Završni rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University North / Sveučilište Sjever**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:122:004370

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-20**

Repository / Repozitorij:

University North Digital Repository

Sveučilište
Sjever

Završni rad br. 373/EL/2016

Upravljanje rasvjetom i električnom bravom korištenjem mobilnih uređaja i Bluetooth komunikacije

Danijel Korunić, 5039/601

Varaždin, rujan 2016. godine

Sveučilište Sjever

Odjel za elektrotehniku

Završni rad br. 373/EL/2016

Upravljanje rasvjetom i električnom bravom korištenjem mobilnih uređaja i Bluetooth komunikacije

Student

Danijel Korunić, 5039/601

Mentor

Miroslav Horvatić, dipl. ing.

Varaždin, rujan 2016. godine

Sveučilište Sjever
Sveučilišni centar Varaždin
104. brigade 3, HR–42000 Varaždin

UNIVERSITY NORTH

Prijava završnog rada

Definiranje teme završnog rada i povjerenstva

ODJEL: Odjel za elektrotehniku

PRISTUPNIK: Danijel Korunić | MATIČNI BROJ: 5039/601

DATUM: 02.06.2016. | KOLEGIJ: Automatsko upravljanje

NASLOV RADA: Upravljanje rasvjetom i električnom bravom korištenjem mobilnih uređaja i Bluetooth komunikacije

NASLOV RADA NA ENGL. JEZIKU: Control of lighting and electric lock using mobile devices and Bluetooth communication

MENTOR: Miroslav Horvatić, dipl. ing. | ZVANJE: predavač

ČLANOVI POVJERENSTVA:
1. mr. sc. Ivan Šumiga, dipl. ing.
2. mr. sc. Matija Mikac, dipl. ing.
3. Miroslav Horvatić, dipl. ing.
4.
5.

Zadatak završnog rada

BROJ: 373/EL/2016

OPIS

Realizirati upravljanje RGB LED trakom, žaruljom i električnom bravom korištenjem Arduino razvojnog sustava i Bluetooth komunikacije. Razviti mobilnu aplikaciju kojom će se vršiti upravljanje. Sustav upravljanja treba imati osjetila za mjerenje temperature i osvjetljenja te mogućnost pohranjivanja podataka na mobilni uređaj. Mobilna aplikacija treba omogućiti promjenu intenziteta i boje RGB LED trake. Osjetilo osvjetljenja iskoristiti za mjerenje svjetlosti žarulje. Električna brava treba biti zaštićena četveroznamenkastom brojčanom ili znakovnom šifrom. Mobilna aplikacija treba pohraniti vrijeme otključavanja i zaključavanja brave te omogućiti promjenu šifre.

U radu je potrebno:
- opisati upravljanje LED trakom, žaruljom i električnom bravom,
- opisati korištenje osjetila, realizaciju Bluetooth komunikacije i Arduino razvojni sustava,
- osmisliti programski dio sustava,
- realizirati sklopovski i programski dio sustava,
- razviti i detaljno opisati mobilnu aplikaciju kojom će se vršiti upravljanje,
- detaljno opisati rad gotovog sustava upravljanja.

REPUBLIKA HRVATSKA 5 KOPRIVNICA SVEUČILIŠTE SJEVER

ZADATAK URUČEN: 07.06.2016. | POTPIS MENTORA: Jm

Predgovor

Zahvaljujem mentoru Miroslavu Horvatiću, dipl. ing, na prenijetom znanju, nesebičnoj i velikoj pomoći te kontinuiranosti praćenja procesa nastanka ovog završnog rada. Također zahvaljujem na susretljivosti kod odabira teme i sve dobronamjerne savjete kojima je učinio ovaj rad uspješnim.

Zahvaljujem svim profesorima Sveučilišta Sjever u Varaždinu te svim profesorima Tehničke škole Čakovec koji su svojim strpljenjem, trudom i entuzijazmom prenijeli svoje znanje, što je veoma pomoglo kod izrade ovog završnog rada.

Također se zahvaljujem svojoj obitelji na velikoj i neizmjernoj potpori te pružanju mogućnosti obrazovanja na Sveučilištu Sjever u Varaždinu. Zahvaljujem firmi TMT d.o.o. na financijskoj i materijalnoj pomoći te svima ostalim kolegama i prijateljima koji su na bilo koji način pomogli oko obrazovanja i uspješne izrade ovog završnog rada.

Sažetak

U ovom završnom rada opisan je koncept i realizacija udaljene kontrole RGB LED trakom, žaruljom i električnom bravom korištenjem *Arduino* razvojnog sustava. Opisan je programski alat *App Inventor* za izradu *Android* aplikacije i *Bluetooth* bežična tehnologija. Detaljno su opisane korištene upravljačke komponente te njihove funkcije. Rad električne brave simuliran je pomoću dvije LED diode (otključano / zaključano). *Arduino UNO* razvojna platforma zadovoljava zadatke ovog završnog rada pa je prema tome odabrana kao glavni dio sustava za upravljanje navedenim uređajima. Odabrana sklopovska i programska platforma omogućavaju kasniju modifikaciju i nadogradnju.

Maketa se sastoji od upravljačke jedinice (*Arduino UNO*) sa RGB shieldom, *Bluetooth* modula, LED trake, relejnog modula, žarulje, temperaturnog senzora, fotootpornika te dvije LED diode koje simuliraju rad električne brave. Prije prvog pokretanja aplikacije korisnik mora upariti svoj mobilni uređaj sa *Bluetooth* modulom. Uređaji se mogu upariti u postavkama mobilnog uređaja tako da se unese lozinka za identifikaciju. Svakim pokretanjem aplikacije dobiva se obavijest za brzo uključivanje Bluetooth komunikacije, kako bi aplikacija mogla raditi. Korisnik nakon spajanja na *Bluetooth* modul dobije obavijest o uspješnom spajanju i tek nakon toga može početi s radom (upravljanjem) u aplikaciji. Aplikacija je realizirana korištenjem četiri kartice (*eng. tab*) te se ovisno o odabranoj kartici uključuje/isključuje vidljivost ostalih. Korisniku se daje na izbor regulacija intenziteta i promjena boje LED trake, uključivanje i isključivanje žarulje uz moguće očitavanje i pohranjivanje podataka o trenutnom osvjetljenja, provjera trenutne temperature te spremanje istog u tekstualnu datoteku. Korisnik može upravljati bravom koja je zaštićena četveroznamenkastom znakovnom lozinkom uz mogućnost mijenjanja lozinke te također pohranjivanje podataka o otključavanju i zaključavanju u tekstualnu datoteku ovisno o vremenu u kojem je određena radnja napravljena.

KLJUČNE RIJEČI: Android, aplikacija, Arduino RGB shield, Arduino UNO, Bluetooth modul, električna brava, fotootpornik, LED traka, mikrokontroler, relejni modul, temperaturni senzor, žarulja

Popis korištenih kratica

COM	(communication port) komunikacijski port
DC	(direct current) istosmjerna struja
AC	(alternating current) izmjenična struja
PWM	(pulse-width modulation) pulsno-širinska modulacija
Vcc	(positive supply voltage) pozitivni polaritet napona
GND	(ground) uzemljenje
RGB	(red - green - blue) crveno - zeleno - plavo
USB	(universal serial bus) univerzalna serijska sabirnica
MAC	(media acces control) fizička adresa
OS	(operating system) operacijski sustav
NC	(normally closed) normalno zatvoren
NO	(normally open) normalno otvoren
LED	(light emitting diode) svjetleća dioda

Sadržaj

1. Uvod 6
2. Arduino razvojni sustav 7
   2.1. Arduino IDE sučelje 8
   2.2. RGB SHIELD 10
      2.2.1. Specifikacije i Sheme Velleman RGB Shielda 10
3. Bluetooth tehnologija i Android OS 12
   3.1. Bluetooth modul HC-06 13
   3.2. Android operacijski sustav 15
   3.3. Komunikacijski protokol i blok dijagram izvršavanja koda 15
4. Realizacija sklopa 18
   4.1. RGB LED traka 19
   4.2. Relejni modul 21
   4.3. Temperaturni senzor DS18B20 22
   4.4. Fotootpornik 23
   4.5. Rad električne brave 24
5. Izrada aplikacije 26
   5.1. APP Inventor 26
   5.2. Dizajn aplikacije 28
   5.3. Logički dio aplikacije 37
      5.3.1. LED traka 40
      5.3.2. Relejni modul (Žarulja) 43
      5.3.3. Temperatura 47
      5.3.4. Brava 49
6. Zaključak 55
7. Literatura 57
8. Prilozi 58

1. Uvod

Razvojem tehnologije ljudima se pokušava stvoriti što jednostavniji život, kako van doma tako i unutar njega. Automatizacija je jedan od načina poboljšanja kvalitete življenja unutar doma. Pomoću glavne upravljačke jedinice (*Arduino UNO*), senzora, *Bluetooth* modula, relejnog modula i *Android* mobilnog uređaja zaduženog za komunikaciju između korisnika i upravljačke jedinice, moguće je modernizirati svakodnevni život.

Upravo je taj primjer tema ovog završnog rada. Senzori koji se koriste u završnome radu su temperaturni senzor i fotootpornik. Interakcija je ostvarena pomoću aplikacije koja je grafički programirana i dizajnirana u programskom alatu *MIT App Inventor* tako da korisnik na jednostavan način zadaje naredbe upravljačkoj jedinici. Ulazni elementi ovog sklopa su *Android* mobilni telefon, odnosno aplikacija koja preko *Bluetooth* modula šalje naredbe upravljačkoj jedinici te korišteni senzori. Izlazni elementi su LED traka, relejni modul koji se koristi za upravljanje žaruljom i dvije LED diode koje simuliraju električnu bravu. Postoji mnogo mogućnosti nadogradnje sklopa kao npr. zamjena *Bluetooth* modula sa *wi-fi* shieldom.

Završni rad je podijeljen na 8 poglavlja u kojima se detaljno opisuje realizacija upravljanja rasvjetom i električnom bravom korištenjem mobilnog uređaja i *Bluetooth* komunikacije. Također su opisani programski alati koji su korišteni za realizaciju ovog završnog rada.

Nakon uvoda, u drugom poglavlju ukratko je opisan Arduino razvojni sustav. U trećem poglavlju ukratko je opisana Bluetooth tehnologija i Android OS. U četvrtom poglavlju opisana je fizička realizacija sklopa te svi korišteni elektronički elementi. U petom poglavlju detaljno je opisan proces izrade mobilne aplikacije.

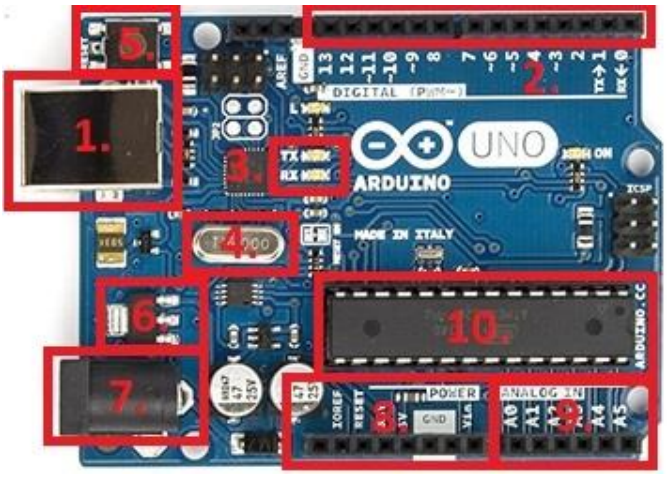
2. Arduino razvojni sustav

Arduino UNO je platforma otvorenog koda koja se koristi za razne projekte iz područja elektronike. Sastoji se od programskog i sklopovskog dijela. Programski dio *Arduino UNO* razvojnog sustava sastoji se od integriranog razvojnog okruženja, tzv. IDE (eng. Integrated Development Environment) koje se koristi za pisanje i uređivanje programa te za prevođenje programskog jezika C u asemblerski kod. Sklopovski dio *Arduino UNO* razvojnog sustava sastoji se od tiskane pločice koja sadrži programirljivi mikrokontroler koji razumije samo asemblerski kod. Kada se prevedeni programski kod prenese u memoriju mikrokontrolera, izvršava se tako dugo dok ima napajanje. Nestankom napajanja ili resetiranja, programski kod počinje iznova. Potrebno vrijeme za obradu pojedinih podataka mikrokontrolera kraće je od milisekunde, što omogućava relativno brzu obradu podataka. Postoji nekoliko modela *Arduino* platforme, a model korišten u ovom radu je *ArduinoUno R3.* Navedeni model sadrži 8 bitni mikrokontroler *Atmega 328P*.

Radni napon *ArduinoUNO R3* pločice je 5 V, a pločica ukupno ima 14 digitalnih pinova od kojih se 6 može koristiti kao *PWM* izlazi. Razvojna pločica ima i 6 analognih pinova. Pinovi mogu biti konfigurirani kao ulazni ili izlazni. Analogni pinovi su spojeni na 10 bitni analogno-digitalni pretvornik (2^{10} = 1024). Tako se analogna vrijednost napona u rasponu od 0 - 5 V pomoću analogno-digitalnog pretvornika pretvara u digitalnu vrijednost u rasponu od 0 - 1023, odnosno postoji 1024 vrijednosti. Kod digitalnih pinova treba posebno spomenuti pinove *Rx (eng. receive)* i *Tx. Rx* služi za primanje podataka, a *Tx (eng. transmit)* pin služi za slanje podataka. Korištenjem navedenih pinova odvija se serijska komunikacija s *Bluetooth* modulom.

Dijelovi *Arduino UNO* R3 sklopovskog razvojnog sustava:

1. USB priključak
2. Digitalni pinovi
3. Tx i Rx signalizacijske LED diode
4. Kristalni oscilator
5. RESET tipka
6. Regulator napona
7. Priključak za napajanje
8. Pinovi napajanje i mase
9. Analogni pinovi
10. Mikrokontroler



Slika 1. Prikaz komponenti Arduino Uno R3 sklopovskog razvojnog sustava [1]

Mikrokontroler	Atmega328P
Frekvencija procesora	16 MHz
Radni napon	5 V
Ulazni napon (preporučeno)	7-12V
Ulazni napon (granica)	6-20V
Digitalni I/O pinovi	14 (od toga moguće 6 PWM izlaza)
Analogni ulazni pinovi	6
DC struja po I/O pinu	20 mA
DC struja za 3.3V pin	50 mA
Flash memorija	32 KB (0.5 KB rezervirano za bootloader)
SRAM	2 KB
EEPROM	1 KB

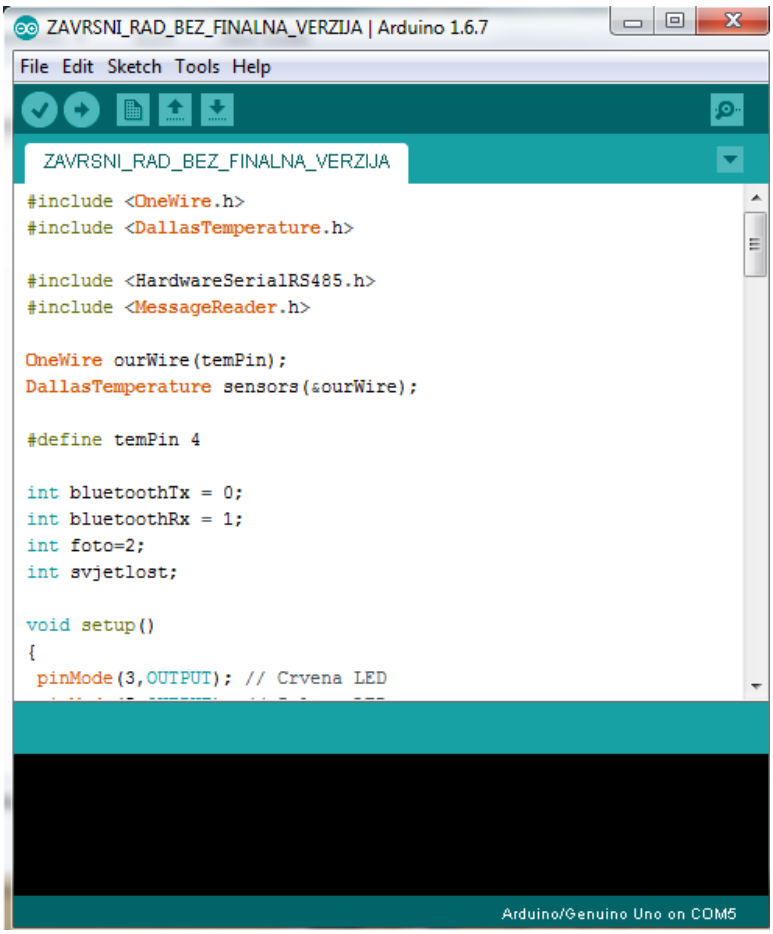
Tablica 1. Specifikacije Arduino UNO razvojne platforme

Mikrokontroler radi na naponskoj razini od 5V te kako bi se izbjegla oštećenja treba paziti prilikom spajanja vanjskih elemenata da naponska razina ne prelazi radnu naponsku razinu. Programiranje mikrokontrolera vrši se u *Arduino IDE* programskom sučelju, pa se upravljački program u mikrokontroler učitava putem USB priključka.

2.1. Arduino IDE sučelje

Arduino programsko sučelje čini besplatna razvojna okolina koja se može preuzeti na službenoj web stranici [1]. Programski jezik je kombinacija C/C++ jezika. Najprije se trebalo upoznati s pojedinim elementima elektroničkog sklopa te na temelju toga napisati funkcije pojedinih elemenata koje se kasnije spajaju u cjelinu.

Napisan program, tzv. *sketch* prvo treba proći provjeru ispravnosti napisane sintakse. Nakon uspješne provjere, odnosno ispravne sintakse, program se kompajlira (*eng. compile*) i nakon toga se može prenijeti na mikrokontroler. Ukoliko dođe do greške u pisanju programa ili sintaksi konzola precizno prikazuje mjesto i detalje o grešci. Također nakon kompajliranja programa ako se uključi *Serial monitor* može se pratiti prikaz podataka koji se šalju ili primaju putem serijske komunikacije.

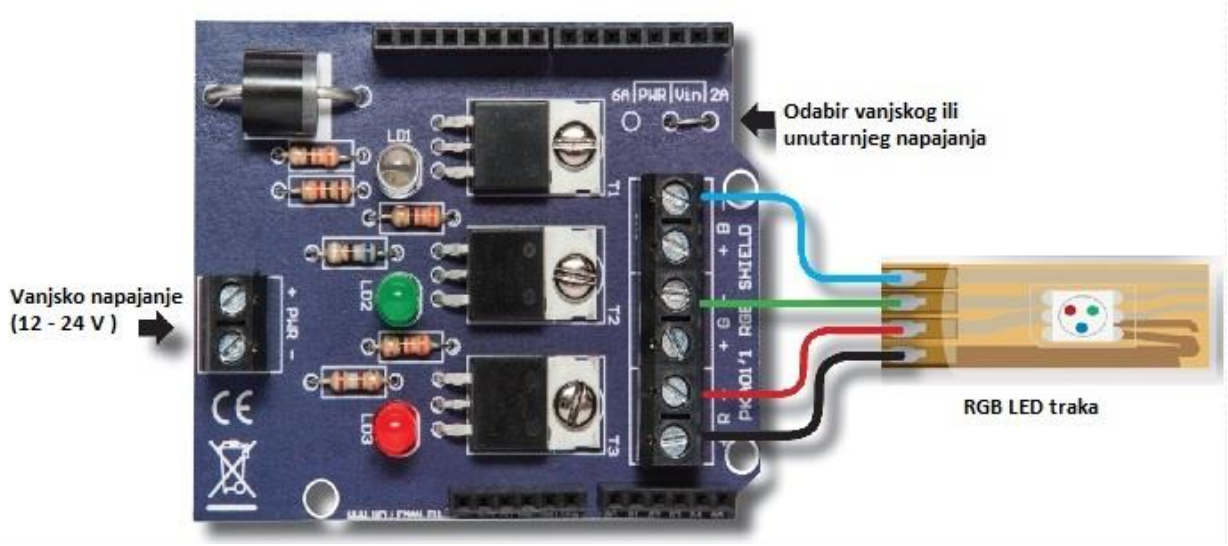


Slika 2. Arduino IDE programsko sučelje

Programi su spremljeni sa ekstenzijom .ino. Prilikom pisanja programa potrebno je najprije uključiti knjižnice koje se koriste za pojedine elemente. Unutar knjižnice se nalaze prethodno definiran način rada pojedinog elementa te se one koriste radi lakšeg pisanja koda.

2.2. RGB SHIELD

Red Green Blue shield je jedna od brojnih pločica za nadogradnju *Arduino* razvojnog sustava. Mikrokontroler *Arduino* razvojnog sustava na svojim pinovima ne može dati dovoljnu veliku struju za napajanje LED trake. Zbog toga će se u ovom radu koristiti RGB Shield pločica koja LED traci osigurava dovoljan iznos struje. RGB shield ima svoje posebno napajanje te je također moguće upravljati s tri kanala na shieldu uz mogućnost prigušenja (*eng. dimmer*) signala. RGB Shield koristi tri pina s *Arduino* pločice za svoja tri kanala pa je tako pin 3 za crvenu, pin 5 za zelenu i pin 6 za plavu. Kod lemljenja treba pripaziti na polarizaciju svjetlećih dioda (*slika 4.* - LD1, LD2, LD3) i Diode (*slika 4.* - D1).

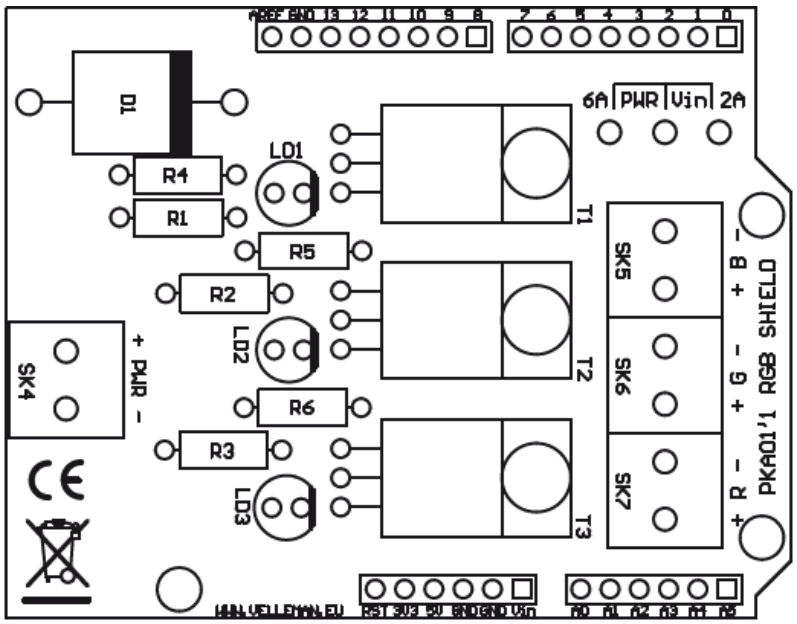


Slika 3. Velleman RGB Shield [2]

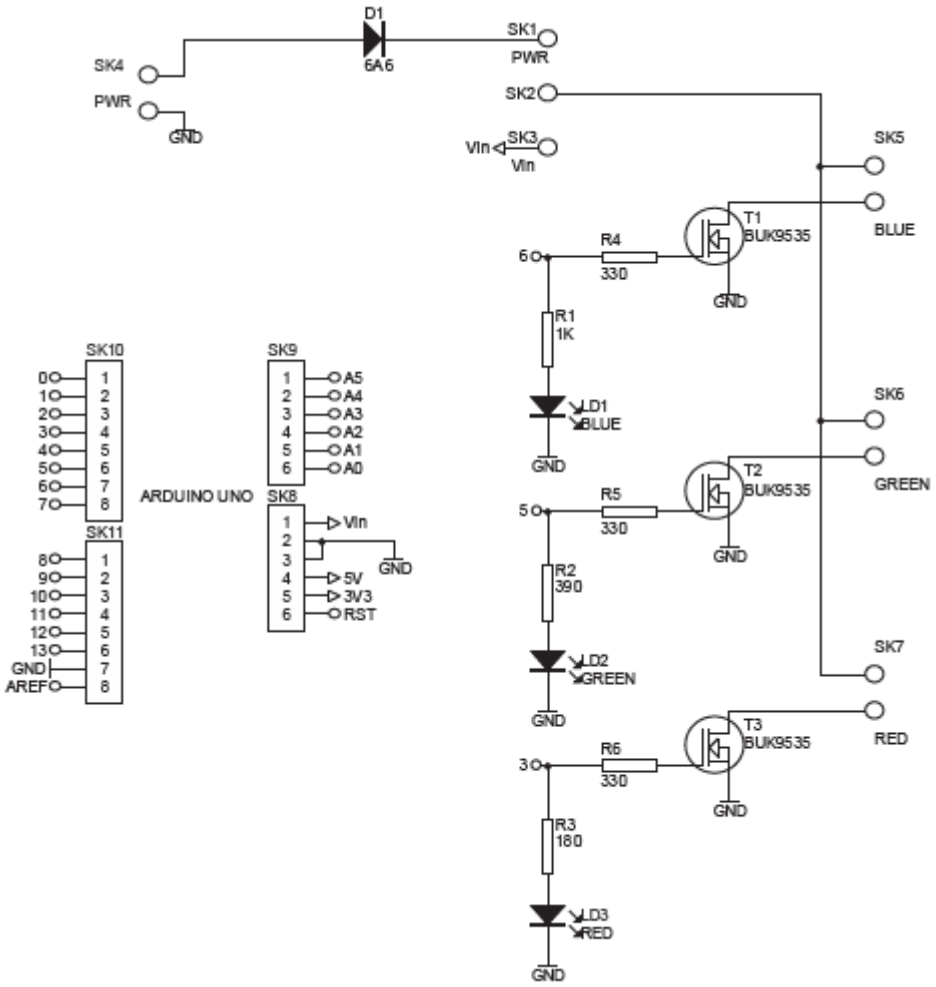
2.2.1. Specifikacije i Sheme Velleman RGB Shielda

Maksimalna struja	2A po kanalu
Maksimalan ulazni napon	50 VDC
Dimenzije	68 x 53 mm

Tablica 2. Specifikacije Velleman RGB Shielda



Slika 4. Montažna shema Velleman RGB Shielda [9]



Slika 5. Elektronička shema Velleman RGB Shielda [9]

3. Bluetooth tehnologija i Android OS

Bluetooth je standard bežične komunikacije koji se koristi za razmjenu podataka na malim udaljenostima, a razvijen je 1994. godine u kompaniji Ericsson. Glavne značajke *Bluetooth* tehnologije su niska potrošnja energije, niska cijena te mali domet. Zbog korištenja radio veze uređaji koji se povezuju ne moraju biti međusobno usmjereni, a veza se može ostvariti u promjeru od otprilike 10 metara oko uređaja. *Bluetooth* omogućava brzinu prijenosa reda veličine 1 Mbit/s, novije verzije i do 24 Mbit/s te koristi nelicencirani frekvencijski pojas od 2.4 do 2.485 GHz, odnosno koristi ISM područje (*engl. Industrial, Scientific and Medical*) koje je frekvencijski usklađeno na svjetskoj razini.

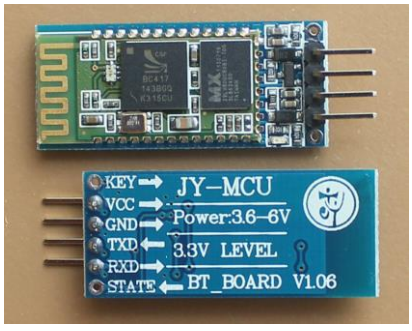
Za komunikaciju se koristi primopredajnik koji stalno, u skokovima (frequency hop) mijenja radnu frekvenciju kako bi se smanjile smetnje i pogreške pri prijenosu. Koristi se dvosmjerni prijenos podataka. Kako bi se pojednostavila izvedba primopredajnika, koristi se binarna frekvencijska modulacija pri prijenosu podataka. Kroz komunikacijski se kanal informacija prenosi u paketima, koji se šalju svaki na svojoj frekvenciji (svaki paket koristi jedan frekvencijski skok). Komunikacijski kanal je podijeljen u slotove (vremenske odsječke) od kojih svaki slot ima nominalnu duljinu od 625 μs. Paket se normalno smješta u svoj slot, ali pojedini paketi mogu zauzeti do 5 slotova.

3.1. Bluetooth modul HC-06

Bluetooth modul HC-06 omogućuje serijsku komunikaciju između *Arduino* sustava i drugoga *Bluetooth* uređaja putem bežične veze. Glavna značajka serijske komunikacije je ta da se bitovi prenose jedan za drugim. Bitovi predstavljaju podatak te se tako prikazuju *TTL* (*eng. Transistor Transistor Logic*) metodom, odnoso različitim naponskim razinama. Za ostvarivanje komunikacije koriste se dva kanala (*Tx i Rx*) kojima se omogućuje dvosmjerna komunikacija. *Tx* pin koji šalje podatke s Arduino sustava, priključuje se na *Rx* pin *Bluetooth* modula tako da modul može primati podatke koje Arudino šalje. Paralelno tome, na *Rx* pin Arduino sustava priključuje se *Tx* pin *Bluetooth* modula kako bi *Arduino* mogao primati podatke koje *Bluetooth* modul šalje. Fizička (*eng. Media Acces Control*) adresa HC-06 modula zapisana je u heksadecimalnom obliku i konkretno u ovom slučaju taj zapis ima oblik 10:14:05:22:12:62. *Bluetooth* modul podržava verziju 2.0, a uređaj na kojem se nalazi aplikacija, odnosno mobilni uređaj podržava verziju 4.0. Iako je verzija na mobilnom uređaju novija ona je kompatibilna s verzijom *Bluetooth* modula.

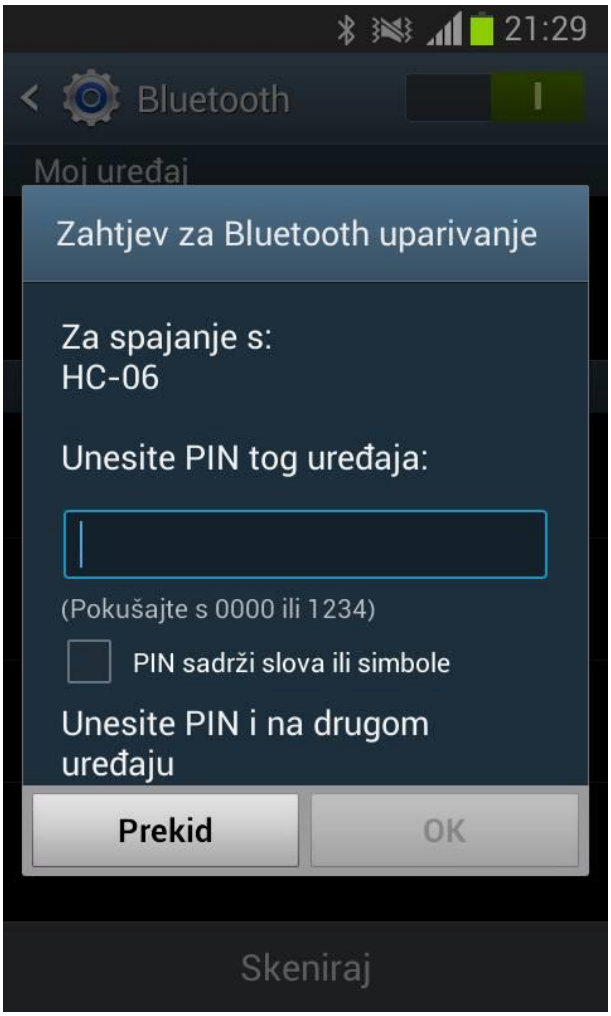
Bluetooth v3.0 riješio je problem male brzine prijenosa podataka pa je ostalo neriješeno pitanje potrošnje energije koja je na današnjim *Bluetooth* uređajima relativno velika. *Bluetooth v4.0* izrađen je upravo s namjerom da riješi ovaj problem. Tako je u sklopu *Bluetooth v4.0* protokola predstavljen i dodatni mod koji se naziva *BLE – Bluetooth low energy*. Uređaji u ovom modu mogu raditi i do nekoliko godina koristeći relativno mali izvor napajanja. To je omogućeno na način da dok su uređaji upareni, a ne dijele veću količinu podataka, odlaze u način mirovanja koji troši neznatnu količinu energije. U slučaju da uređaji moraju intenzivnije izmjenjivati podatke, veza odlazi u normalan način koji također troši mnogo manje energije u usporedbi sa starijim verzijama *Bluetooth* protokola.

Napon napajanja *Bluetooth* modula iznosi 3.6 V - 6 V, dok napon na pinovima za primanje i slanje podataka (*Rx i Tx*) iznosi 3.3 V. *Bluetooth* modul je 2. klase, a broj klase modula predstavlja domet veze. U prikazanom slučaju to iznosi desetak metara.



Slika 6. Bluetooth modul HC-06 [3]

Prvi korak kod spajanja dvaju *Bluetooth* uređaja je uparivanje. Pritom treba provjeriti da li je na oba uređaja *Bluetooth* uključen tj. dostupan i u području dometa. Uparivanje se pokreće ručno od strane korisnika. Da bi uspješno uparili uređaje potrebno je razmijeniti tj. upisati iste lozinke na oba uređaja. Konkretno, u prikazanom slučaju upisana je zadana (*eng. default*) lozinka „1234“.

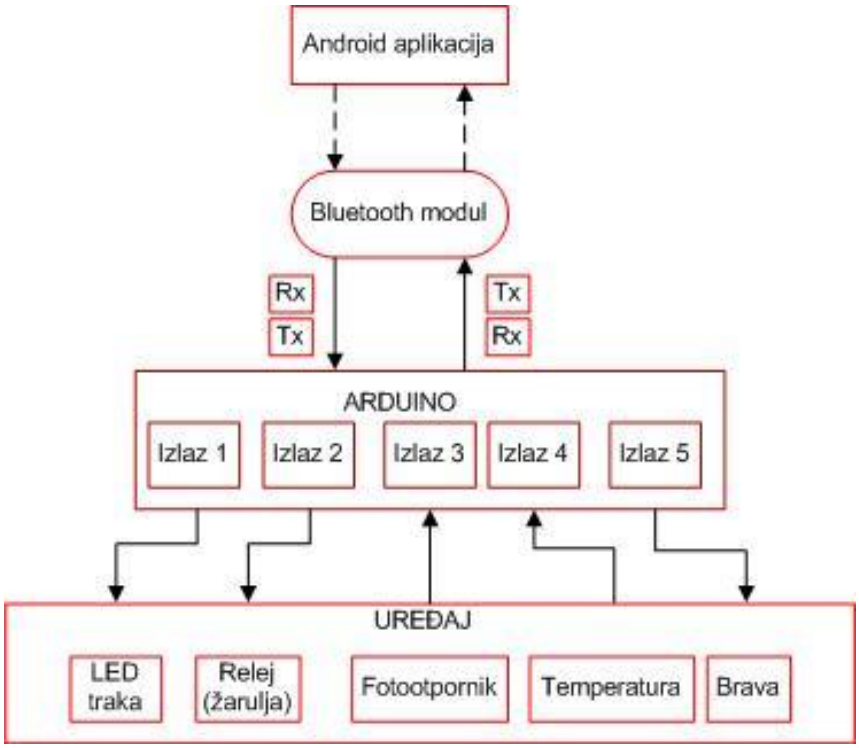


Slika 7. Uparivanje s Bluetooth modulom HC-06

3.2. Android operacijski sustav

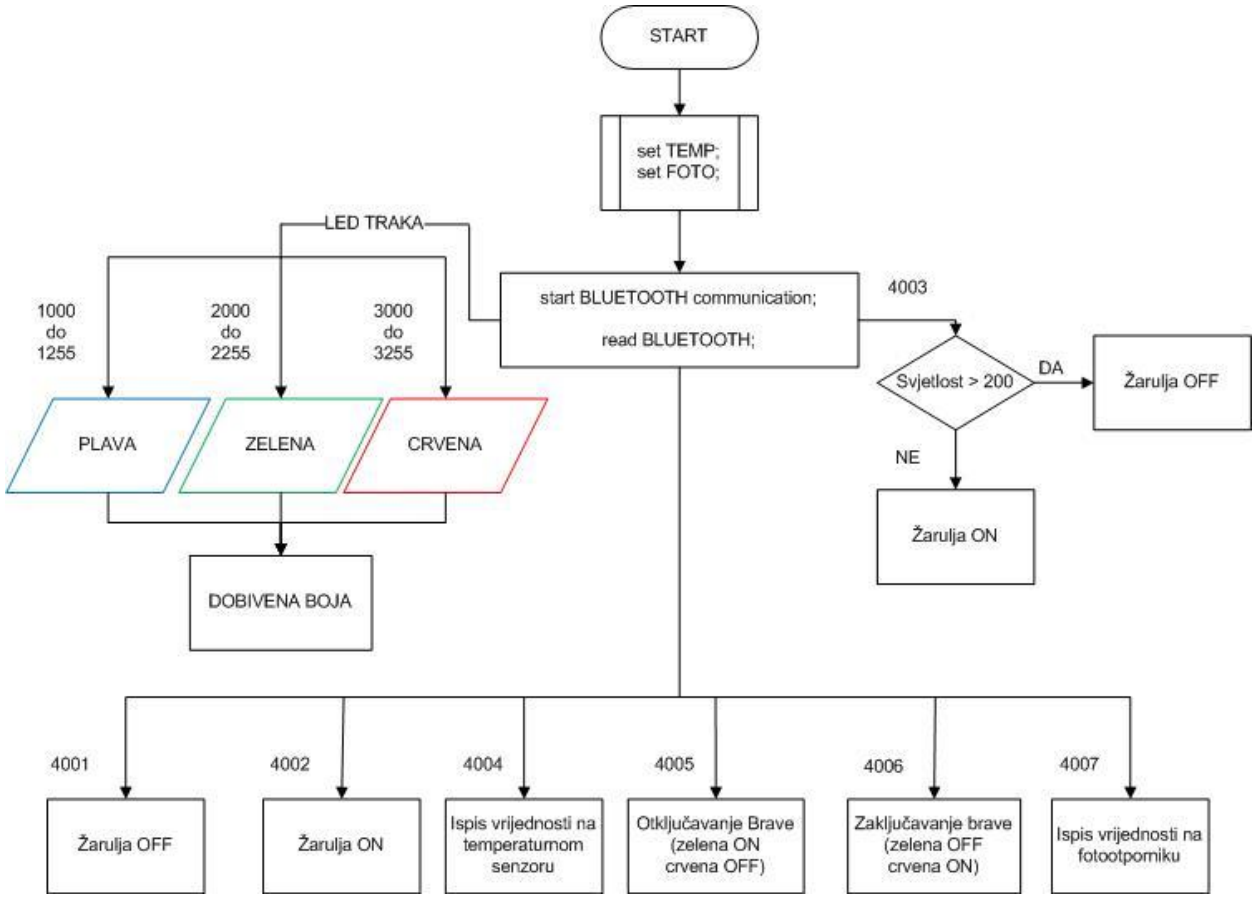
Google Android je prvi otvoreni operacijski sustav za mobilne uređaje (mobilni telefoni, tableti, netbook računala, *Google TV*) pokrenut od strane *Google Inc*. i vođen od strane *Open Handset Alliance* - grupe koja danas broji preko 80 tehnološki kompanija među kojima su HTC, T-Mobile, Intel itd. *Android Inc.* su osnovali Andy Rubin, Rich Miner, Nick Sears i Chris White u listopadu 2003. god., kako bi razvijali programe za pametne mobilne uređaje. Nakon dvije godine rada, *Google* je odlučio kupiti *Android* te počinju priče o ulasku *Google*-a na tržište pametnih telefona. Osnivači i programeri, uz pomoć *Google*-ovh programera na tržište donose mobilnu platformu koja je potpuno prilagodljiva zahtjevima korisnika. *Android* je modularan i prilagodljiv pa postoje slučajevi njegovog prenošenja na razne uređaje. Aplikacija je zapravo jedna datoteka aplikacijskog paketa – *.apk* datoteka. Da bi se neslužbene aplikacije mogle instalirati na mobilni uređaj potrebno je u postavkama uređaja omogućiti instalaciju aplikacija koje nisu službeno odobrene od strane *Google*-a.

3.3. Komunikacijski protokol i blok dijagram izvršavanja koda



Slika 8. Komunikacijski protokol

Bluetooth modul je komunikacijski posrednik između mobilnog uređaja, tj. aplikacije i *Arduino sustava*. Nakon uspostavljanja veze između aplikacije i *Bluetooth* modula, putem modula *Arduino sustav* i aplikacija mogu komunicirati. *Android* aplikacijom šalju se određeni podaci, *Arduino* ih čita te na temelju njih upravlja svojim izlazima. Podaci na temelju kojih *Arduino* upravlja izlazima su brojevi.



Slika 9. Blok dijagram izvršavanja koda

Arduino čita brojeve 1000 - 1255 (plava), 2000 - 2255 (zelena), 3000 - 3255 (crvena) te na temelju tih brojeva uključuje LED traku i bira boju kojom će traka svijetliti te određuje jakost pojedine boje. Kad dobije broj 4002 uključuje relej (žarulju) i drži ga uključenog sve dok ne dobije 4001 za isključivanje. Također se u samoj kartici za upravljanje žaruljom može poslati 4003 za automatsko uključivanje releja ukoliko je preslabo osvjetljenje ili isključivanje ukoliko je prejako osvjetljenje. Slanjem 4007 šalje se zahtjev Arduinu za očitavanje vrijednosti na fotootporniku tj. jakost svijetla u određenoj okolini te nakon što *Arduino* očita vrijednost šalje tu vrijednost nazad aplikaciji tj. mobilnom uređaju. Aplikacija na mobilnom uređaju ima

mogućnost prikazivanja i spremanja vrijednosti na mobilnom uređaju. Uz fotootpornik u sustavu se nalazi i temperaturni senzor, pa se tako slanjem 4004 šalje zahtjev za očitavanjem vrijednosti na temperaturnom senzoru te se na isti način informacija vraća do korisnika kao i sa fotootpornikom. Uz LED traku, žarulju i senzore u sustavu se nalazi električna brava, a njome se upravlja slanjem 4005 za otključavanje i 4006 za zaključavanje.

Na koji način se korištenjem aplikacije ovi podaci šalju biti će objašnjeno pod opisom programskog dijela aplikacije, za svaku karticu (*eng. tab*) posebno. Bitno je napomenuti da *Arduino* stanja izlaza konstantno čita u petlji (*loop*) i konstantno ih šalje aplikaciji, a u aplikaciji se po potrebi šalju podaci *Arduinu*, na temelju kojih on može kontrolirati izlaze.

4. Realizacija sklopa

Zadatak završnog rada je upravljanje rasvjetom i električnom bravom korištenjem mobilnog uređaja. Dakle, skup manjih zadataka koji se spoje u cjelinu. Potrebna je realizacija upravljanja LED trakom, žaruljom, bravom te senzorima. Prilikom realizacije upravljanja potrebno je izraditi odgovarajuću programsku aplikaciju za mobilni uređaj i programski kod za Arduino platformu. Korišteni hardver i programski kod potrebni za ovaj rad opisani su u nastavku.

Ključni dio programskog koda je naredba koja se dobije preko *Bluetooth* modula. Ovisno o tome s čime korisnik koji koristi aplikaciju želi upravljati šalju se različiti brojevi preko *Bluetooth* modula koje *Arduino* čita i upravlja određenim elementima. Kako bi se mogle čitati poslane naredbe, u Arduino programskom okruženju potrebno je pročitati i spremiti nekoliko podataka.

```
if(Serial.available()>= 2 )  {
int BTnaredba1 = Serial.read();
int BTnaredba2 = Serial.read();
int BTnaredba = (BTnaredba2 *256) + BTnaredba1;
```

Na početku se provjerava ako ima podataka na serijskom portu te se nakon toga podaci spremaju u *BTnaredba1* i *BTnaredba2*. Konačno *BTnaredba* je kombinacija prethodno pročitanih podataka.

4.1. RGB LED traka

LED trake SMD flex su posebno dizajnirane na fleksibilnoj samoljepljivoj PCB traci, a koriste se u interijerima i eksterijerima, najčešće za dekorativnu i indirektnu rasvjetu. Koncept RGB (crvena, zelena i plava) LED traka je da se "miješanjem" različitih kombinacija ove tri boje dobiva čitav spektar različitih boja. Napajanje koje se dobiva putem USB porta na *Arduino* razvojnoj pločici nije dovoljno za LED traku pa tako postoji potreba za dodatnim napajanjem. Konkretno u završnom radu za napajanje LED trake od 1 m koristi se ispravljač s nazivnim podacima:

Model	HJ-12V200
Ulazni napon (AC)	100 - 240 V
Ulazna frekvencija	50/60 Hz
Izlazni napon (DC)	12 V
Izlazna struja	2 A

Tablica 3. Nazivni podaci ispravljača za napajanje LED trake

Slika 10. SMD Flex RGB LED traka [4]

Kod definiranje LED trake u *Arduino IDE* programskom okruženju uzima se u obzir da su za LED traku potrebna 3 pina za crvenu, zelenu i plavu boju, što znači da se sva tri pina trebaju definirati. Prema standardu RGB Shielda to su digitalni pinovi 3 za crvenu, 5 za zelenu i 6 za plavu boju.

Promjena boje obavlja se korištenjem tri if petlje:

```
      // PLAVA
if (BTnaredba >= 1000 && BTnaredba <1255){
      int plava = BTnaredba;
      plava = map(plava, 1000,1255,0,255);
      analogWrite(6,plava);
      delay(10); }

      // ZELENA
if (BTnaredba >=2000 && BTnaredba <2255){
      int zelena = boja;
      zelena = map(zelena,2000,2255,0,255);
      analogWrite(5,zelena);
      delay(10);}

      // CRVENA
if (BTnaredba >=3000 && BTnaredba < 3255){
      int crvena = BTnaredba;
      crvena = map(crvena, 3000, 3255,0,255);
      analogWrite(3,crvena);
      delay(10);}
```

Kako bi se spriječile neželjene kolizije, jer svaka od tih tri boja ima na raspolaganju vrijednosti od 0 do 255, svaka boja koristi svoju tisućicu. Plava boja koristi brojeve od 1000 do 1255, zelena od 2000 do 2255 i crvena od 3000 do 3255.

4.2. Relejni modul

Jednokanalni relejni modul s optoizolatorima koristi se za uključivanje i isključivanje uređaja čija je nazivna struja veća od one koju može dati *Arduino*. Konkretno u ovom završnom radu koristi se za uključivanje i isključivanje žarulje. Dovođenjem električnog signala na određeni digitalni pin aktivira se relej. Primarni (upravljački) strujni krug radi na 5V DC, a sekundarni (izvršni) strujni krug može podnijeti i do 220V AC. Optoizolatori imaju ulogu električnog odvajanja strujnog kruga mikrokontrolera od strujnog kruga elektromagneta releja. Pojedini relej ima tri izlazne stezaljke: *NO* (normalno otvorenu), *NC* (normalno zatvorenu) i zajedničku *COM* stezaljku. U slučaju normalno otvorenog kontakta (*NO*), kada se napon dovede na elektromagnet, on će privući kotvu i zatvoriti strujni krug. Kod normalno zatvorenog kontakta (*NC*) slučaj je obrnut.

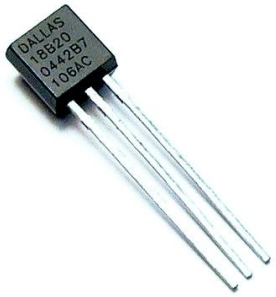
Slika 11. Relejni modul 5V 10 A s optoizolatorima

Definicija u programskom kodu je jednostavna i nije potrebno preuzimanje ili uključivanje nove knjižnice (*eng. library*). Najprije se definira pin na koji će se spojiti upravljačka žica relejnog modula te se zatim definira da je taj pin izlazni, naredbom `pinMode(8,OUTPUT);`
Relejni modul uključuje i isključuje žarulju ovisno o tome koju naredbu *Arduino* dobije preko *Bluetooth* modula.

```
// Uključivanje
if (BTnaredba==4002) {
digitalWrite(8,HIGH); }

// Isključivanje
if (BTnaredba==4001) {
digitalWrite(8,LOW);}
```

4.3. Temperaturni senzor DS18B20

DS18B20 je digitalni senzor koji na izlazu daje 9 bitni do 12 bitni zapis o temperaturi u °C. Senzor koristi 1-wire protokol tj. koristi samo jednu liniju (*DQ*) za prijenos podataka, što znači da se na jedan pin može staviti 127 osjetila i mjeriti temperaturu na 127 mjesta. Osjetilo ima 64-bitni ROM kodni zapis koji nam omogućuje definiciju i pronalazak osjetila kod korištenja 1-wire protokola. Ulaz *DQ* se spaja na određeni pin te preko tzv. "*pull-up*" otpornika na pozitivni napon iznosa 5V. Otpornik služi za održavanje logičke razine. Prije korištenja digitalnog osjetila preko 1-wire protokola, potrebno je preuzeti odgovarajuće knjižnice (*eng. library*) te ih uključiti na početku programa (`#include <OneWire.h>` i `#include <DallasTemperature.h>`) u *Arduino IDE* okruženju.



Slika 12. Temperaturni senzor DS18B20 [5]

Navedeni senzor može mjeriti temperaturu u opsegu od -55°C do 125°C sa točnošću od 0.5°C, na intervalu od -5°C do 95 °C, prema tome najveća točnost je na temperaturama oko 30°C do 40°C.

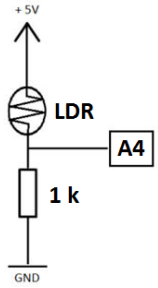
Nakon što su uključene odgovarajuće knjižnice, definira se pin na koji će se spojiti DQ linija temperaturnog senzora, korištenjem definicije `#define temPin 4.` Nakon toga se definira komunikacija sa svim OneWire elementima `OneWire oneWire(temPin);` Također je potrebno započeti mjerenja senzora sa naredbom `sensors.begin();`

U ovom završnom radu ispisivanje vrijednosti koja se nalazi na temperaturnom senzoru obavlja se ukoliko *Arduino* preko *Bluetooth* modula dobije naredbu "4004"

```
// TEMPERATURA
if (BTnaredba == 4004) {
sensors.requestTemperatures();
Serial.println(sensors.getTempCByIndex(0));}
```

4.4. Fotootpornik

Fotootpornik (*LDR - light dependent resistor*) je otpornik čiji se otpor mijenja ovisno o jakosti svjetla koje pada na njega. Ako je svjetlo jakog intenziteta tada je otpor fotootpornika malog iznosa, a ako je svjetlo slabije tada je otpor veliki. Izrađuje se od poluvodiča s velikim električnim otporom. Fotootpornici nisu polarizirani što znači da ne treba posebno paziti kod spajanja u strujni krug. Najčešće se spajaju serijski s otporom jer takav spoj čini naponsko dijelilo. Sklop se priključuje na 5V te na analogni pin Arduino sustava. Analogno- digitalnom pretvorbom signala dobiva se brojčana vrijednost u rasponu od 0 do 1023. Ukoliko je svjetlo slabije na *Arduino mikrokontroleru* se dobiva mala brojčana vrijednost, a u suprotnom slučaju velika brojčana vrijednost.



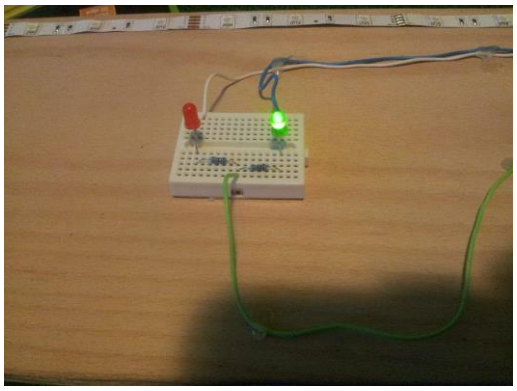
Slika 13. Shema naponskog djelila za priključenje fotootpornik

Fotootpornici imaju manju osjetljivost od foto-dioda i foto-tranzistora. Oni su pasivni elementi kojima nedostaje jedan PN spoj. Vrlo su osjetljivi na promjenu temperature tako da ako se intenzitet svijetlosti održava konstantnim, otpor još uvijek može varirati zbog promjene temperature. Definicija u programskom kodu je jednostavna i nije potrebno preuzimanje ili uključivanje nove knjižnice. Najprije se definira analogni pin na koji će se spojiti fotootpornik, programska linija `int foto=2;` uz to je potrebno definirati i varijablu za spremanje vrijednosti koja opisuje količinu svjetlosti, programska linija `int svjetlost;`.
Fotootpornik konstantno očitava vrijednosti `svjetlost = analogRead(foto);`. Osim samog mjerenja svjetlosti, zadaća fotootpornika u ovo slučaju je provjeravanje da li je žarulja uključena ili isključena. Ukoliko korisnik preko *Bluetooth* modula pošalje naredbu "4003" (u aplikaciji tipka "Optimalno svjetlo"), na temelju mjerenja osvjetljenja okoline fotootpornik odlučuje da li bi bilo dobro uključiti žarulju ili je isključiti.

```
// FOTOOTPORNIK (optimalno svjetlo)
if (BTnaredba == 4003 && svjetlost < 200 && svjetlost > 0 )
{digitalWrite(8,HIGH);}
if (BTnaredba == 4003 && svjetlost > 200)
{digitalWrite(8,LOW); }
```

4.5. Rad električne brave

Rad električne brave simuliran je pomoću dvije svjetleće diode. Zelena dioda označava da je brava u otključanom stanju, a crvena da je brava u zaključanom stanju. Početno stanje brave definirano je kao otključano iz razloga što je sustav predviđen za sobu u kući pa bi bilo nezgodno ostati zaključan u sobi ukoliko korisnik na početku korištenja ne zna točnu lozinku. Izrada kartice (*eng. tab*) za bravu u aplikacije je kompleksan proces o kojem će više pisati u nastavku ovog završnog rada.



Slika 14. Električna brava simulirana s dvije LED žarulje

Najprije se definiraju pinovi na kojima će biti spojene anode *LED* žarulja te se zatim definiraju da su pinovi izlazni, korištenjem naredbi:

```
pinMode(12,OUTPUT) // Brava on;
pinMode(13,OUTPUT) // Brava off;
```

, nakon toga se odredi početno stanje *LED* žarulja (brave):

```
digitalWrite(12,HIGH); // upaljena zelena, otkljucano
digitalWrite(13,LOW); // ugasena crvena
```

te na kraju *Arduino* uključuje i isključuje (otključava i zaključava bravu) *LED* žarulje ovisno o tome koju naredbu *Arduino* dobije preko *Bluetooth* modula:

```
// BRAVA
if (BTnaredba == 4005) {   // OTKLJUCAVANJE
digitalWrite(12,HIGH);     //upaljena zelena, otkljucano
digitalWrite(13,LOW);  }   //ugasena crvena

if (BTnaredba == 4006) {   //ZAKLJUCAVANJE
digitalWrite(12,LOW);      //ugasena zelena
digitalWrite(13,HIGH);}    //upaljena  crvena,  zakljucano
```

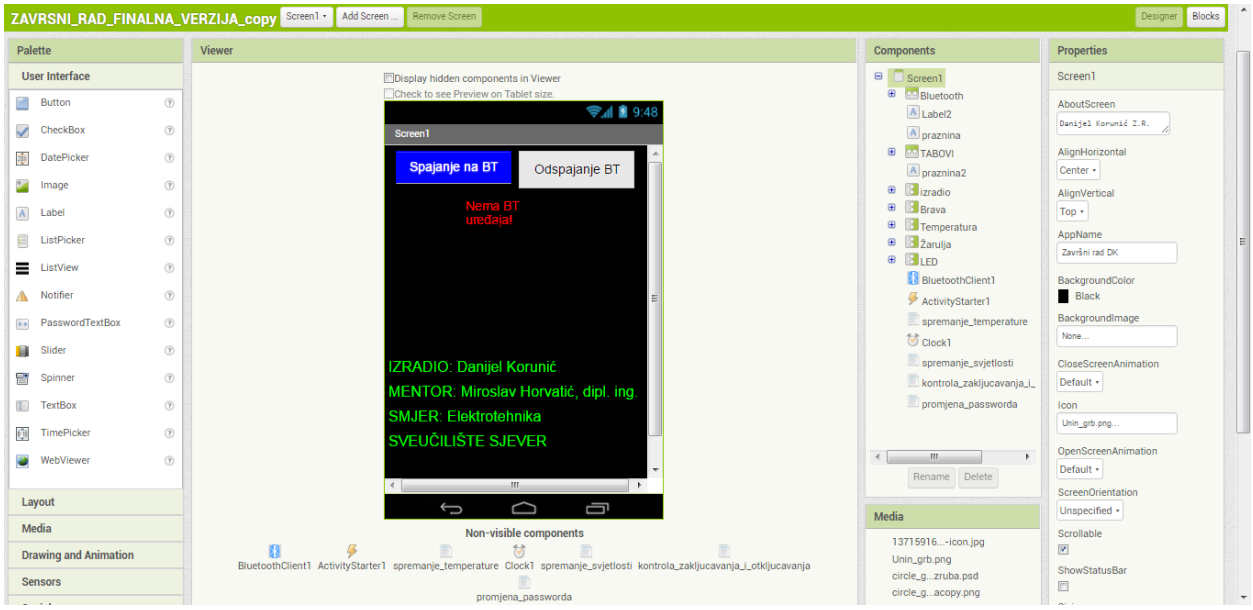
5. Izrada aplikacije

Zadatak završnog rada je upravljanje rasvjetom i električnom bravom korištenjem mobilnog uređaja. Da bi se upravljalo prethodno navedenim elementima preko mobilnog uređaja, potrebno je izraditi mobilnu aplikaciju koja će preko *Bluetooth* modula komunicirati s *Arduino* razvojnom pločicom. U ovom završnom radu za izradu aplikacije koristi se razvojno okruženje *App Inventor*. Aplikacija mora biti pregledna, jednostavna i izrađena tako da se njome može upravljati prethodno navedenim elementima. Način izrade mobilne aplikacije potrebne za ovaj rad opisan je u nastavku.

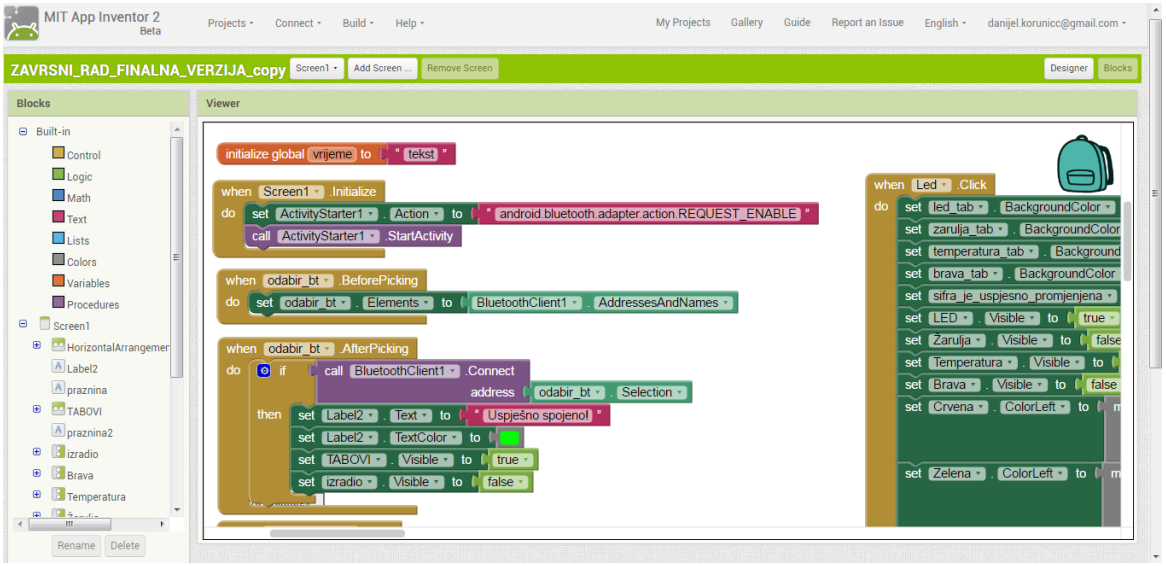
5.1. APP Inventor

App Inventor je *open source* web aplikacija koju razvija istraživačko sveučilište *MIT* za multinacionalnu korporaciju *Google* i trenutno nosi naziv *MIT App Inventor 2*. *App Inventor* koristi *JAVA-u* kao programski jezik. Programiranje je olakšano pomoću blokova koji predstavljaju određeni programski kod. Zbog toga je alat namijenjen svima i za njegovo korištenje nije potrebno poznavanje programskog jezika. Alat se sastoji od dva dijela: *designer* i *blocks*. U designer dijelu se kreira vizualni izgled aplikacije. Komponente na designer dijelu mogu biti vidljive (tekst, slike, gumbi itd.) ili nevidljive (sat, *Bluetooth* klijent, datoteka - spremanje itd.). U Block dijelu stvara se programska logika pomoću blokova. Blokovi određuju kako se komponente trebaju ponašati prilikom neke radnje s njima.

Pristup alatu se obavlja prijavom ili registriranjem vlastitog *Google* korisničkog računa na stranici: *http://appinventor.mit.edu/explore/.*

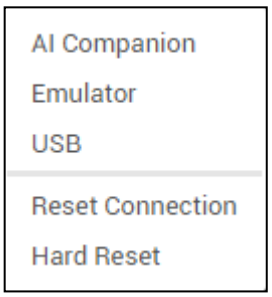


Slika 15. MIT App Inventor 2 Designer



Slika 16. MIT App Inventor 2 Blocks

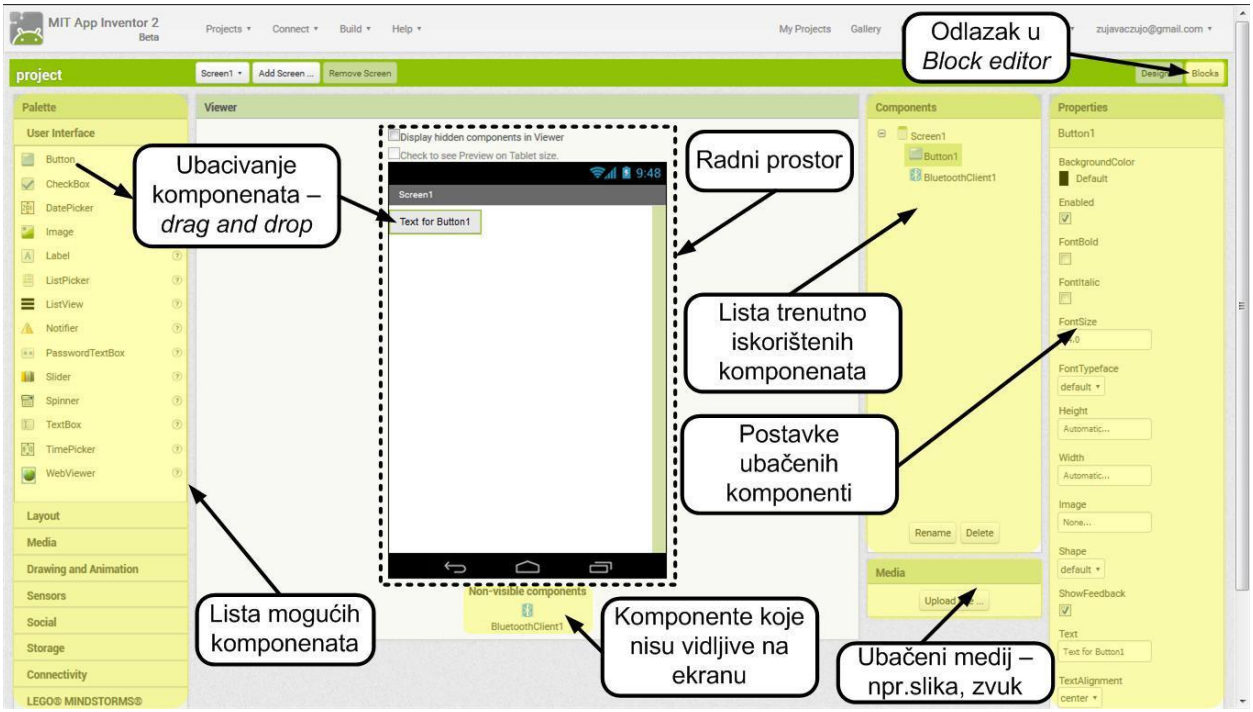
App Inventor također pruža mogućnost testiranja aplikacije kako se izgrađuje. Testiranju se pristupa pritiskom na karticu Connect, nakon čega se može birati između nekoliko načina (*slika 17.).* Najbolji način testiranja je pomoću besplatne aplikacije za *Android* mobilne uređaje pod nazivom "*MIT AI2 Companion*". Nakon što se u kartici *Connect* odabere *AI Companion*, *App Inventor* dodijeljuje jedinstveni kod i *QR* kod koji se mogu upisati ili skenirati u aplikaciju na mobilnom uređaju. Ukoliko korisnik nema mobilni uređaj, aplikaciju je moguće testirati pomoću *Android* emulatora, softvera na računalu koji se ponaša isto kao i mobilni uređaj.



Slika 17. Načini testiranja prilikom izrade aplikacije

5.2. Dizajn aplikacije

Postupak izrade aplikacije započinje definiranjem njezinog izgleda. Raspored vidljivih i nevidljivih komponenata te njihova pozicija na ekranu *Android* uređaja uređuje se unutar dizajnera komponenata. *Slika 18.* prikazuje izgled dizajnera komponenata. Sve moguće komponente koje se mogu iskoristiti za izradu aplikacije smještene su na paleti (eng. *Palette*). Paleta je podijeljena na nekoliko sekcija. Korisnici imaju na raspolaganju najčešće korištenu *User Interface* sekciju koja sadrži: klizače, gumbe, tekstualne oznake, itd. Za izradu ovog završnog rada korištene su još sekcije *Layout* za grupiranje i urednije postavljanje komponenata, *Drawing and Animation* za platno (*eng. canvas*) komponentu, *Sensors* za sat (*eng. clock*) komponentu, *Storage* za datoteka (*eng. file*) komponentu za spremanje podataka na mobilni uređaj i *Connectivity* sekcija za *BluetoothClient* komponentu.



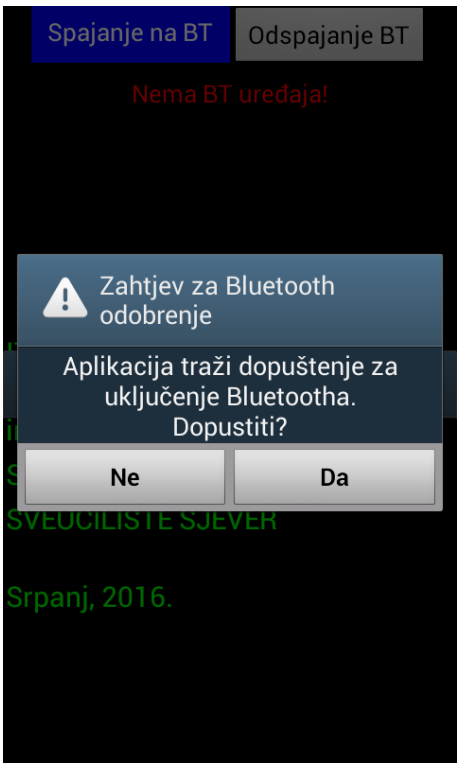
Slika 18. Dizajner komponenta MIT App Inventora 2

Važno je napomenuti da je *App Inventor* još uvijek u Beta verziji, tako da neke stvari još uvijek ne rade kako bi trebale. Postoji opcija za dodavanje i brisanje zaslona kako bi se primjerice pritiskom na neku tipku promijenio zaslon pa bi time aplikacija dobila lijepu preglednost. Jedan od glavnih problema ovog završnog rada bila je *Bluetooth* veza prilikom promjene zaslona. Prilikom svake promjene zaslona trebalo bi ponovo povezati *Bluetooth*. Takav način nije nimalo praktičan. Zbog toga je bilo potrebno pronaći novo rješenje. Rješenje je pronađeno u ostajanju na

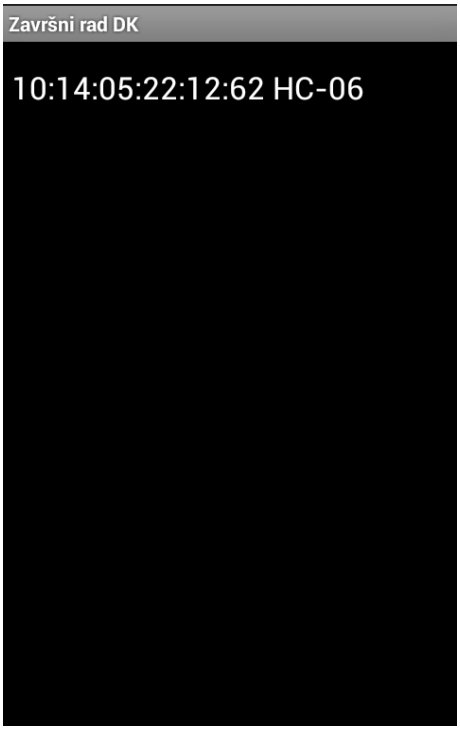
jednom zaslonu uz naizmjenično uključivanje i isključivanje vidljivosti svih komponenata u aplikaciji, ovisno o odabranoj komponenti. Samim time se sve dodatno komplicira, ali uz puno pažnje je izvedivo. Izgled aplikacije je baziran na principu kartica (*eng. tab*), jer je tako najlakše uključivati i isključivati vidljivosti za pojedine komponente.

Slika 19. Početni zaslon aplikacije

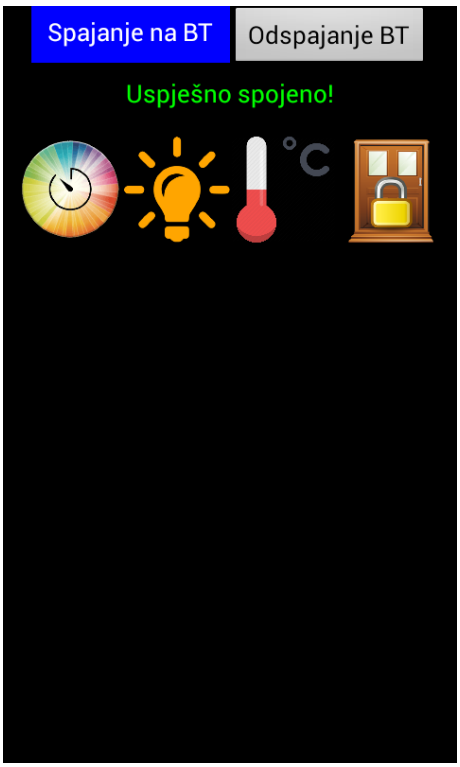
Na samom početku korištenja aplikacije vide se samo informacije o aplikaciji i tipka za spajanje na *Bluetooth* modul.



Slika 20. Activity starter traži uključenje Bluetooth komunikacije

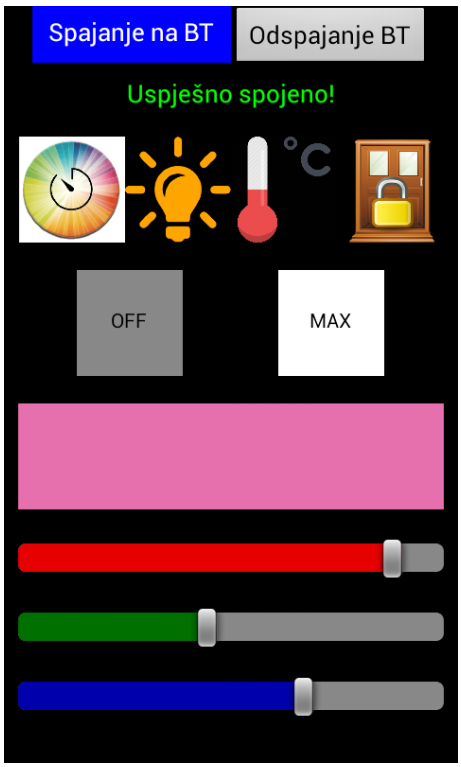


Slika 21. Odabir Bluetooth modula



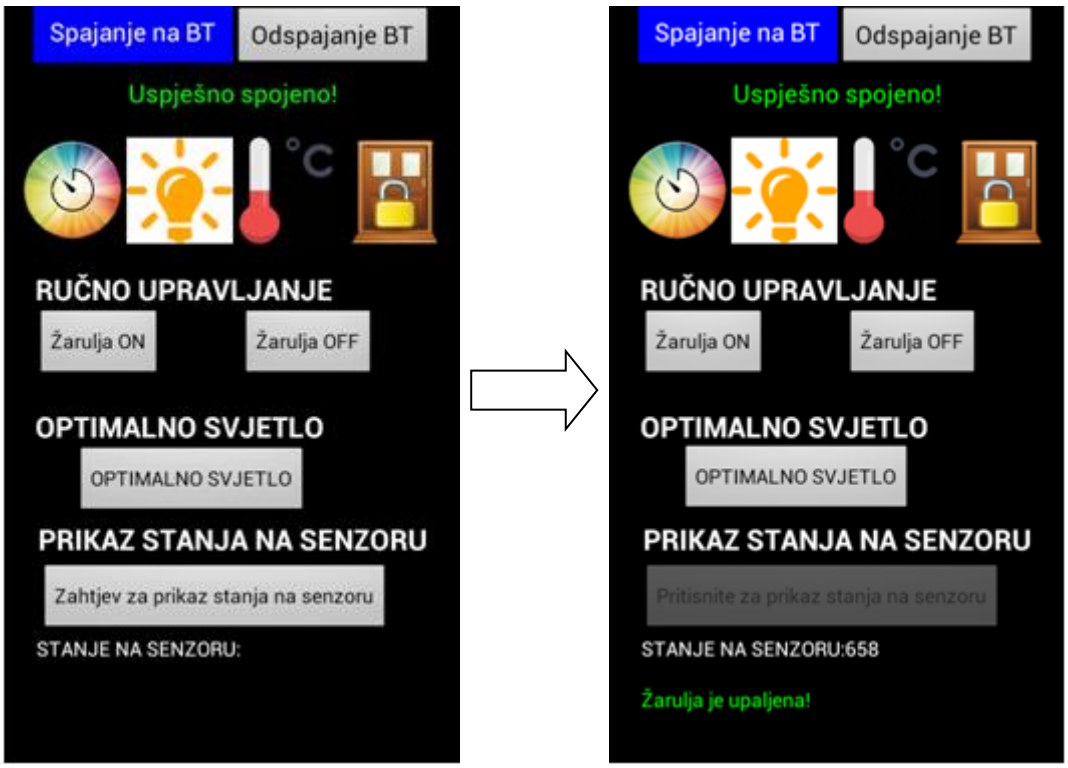
Slika 22. Nakon uspješnog spajanja na Bluetooth modul

Sve komponente su skrivene tako dugo dok korisnik ne uključi *Bluetooth* (*slika 20.*) i tek kada se spoji na određeni *Bluetooth* modul (*slika 21.*) , uključuje se vidljivost kartica (*slika 22.*). Nakon uspješnog spajanja na *Bluetooth* modul tekst "Nema BT uređaja" se mijenja u "Uspješno spojeno!" , a informacije o aplikaciji (*slika 19.*) tj. početni zaslon, se vraća kad korisnik odluči prekinuti *Bluetooth* vezu pritiskom na tipku "Odspajanje BT".



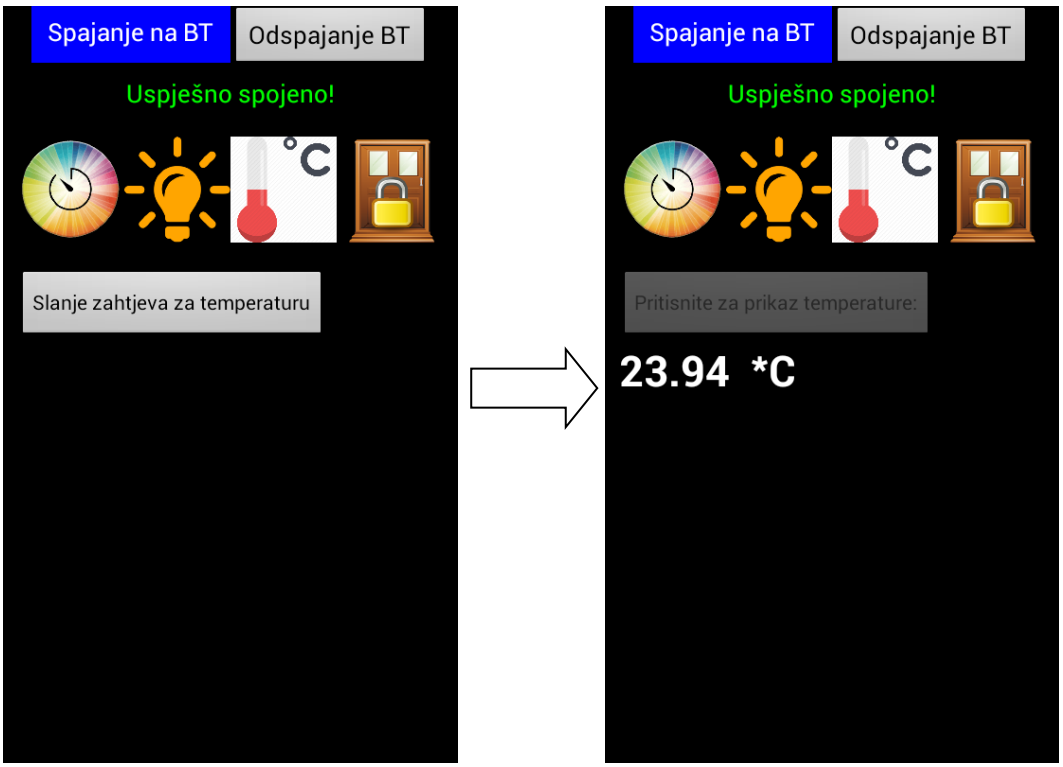
Slika 23. Kartica za upravljanje LED trakom

Ako korisnik odabere karticu za upravljanje LED trakom, pozadina kartice promijeni boju u bijelu i sve komponente za upravljanje LED trakom postanu vidljive dok su sve ostale skrivene. Kod upravljanja LED trakom koriste se tri klizača (*eng. slider*) za pojačavanje intenziteta svake od tri glavnih boja (crvene, zelene i plave). Iznad klizača nalazi se platno (*eng. canvas*), na kojem korisnik preko mobilnog uređaja može vidjeti trenutnu boju na LED traci, što je praktično jer samim time korisnik uopće ne mora fizički vidjeti LED traku. Iznad platna se nalaze dvije tipke čijim se pritiskom klizači automatski postavljaju na određene položaje, prvenstveno radi bržeg upravljanja klizačima. Tako se korisniku daju na izbor dvije tipke, tipka za brzo gašenje (tipka "OFF") i tipka za maksimalno osvjetljenje (tipka "MAX").



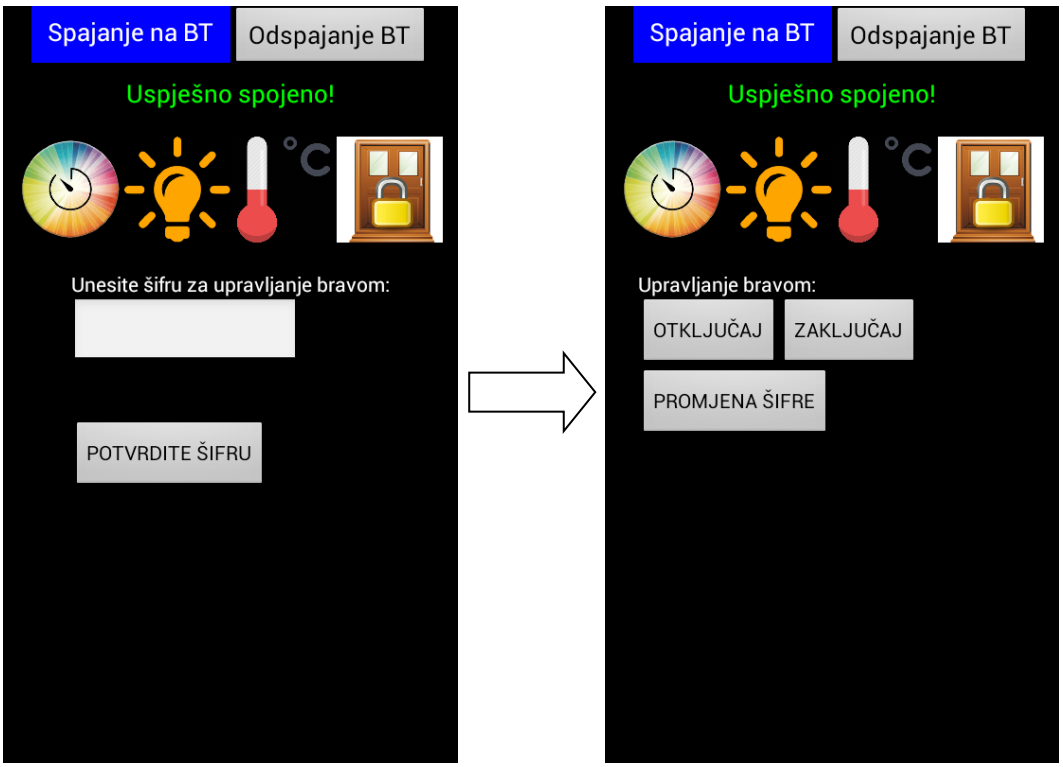
Slika 24. Kartica za upravljanje žaruljom i prikaz stanja na fotootporniku

Ukoliko pak korisnik odabere karticu za upravljanje žaruljom, pozadina kartice promijeni boju u bijelu i sve komponente za upravljanje žaruljom postanu vidljive dok su komponente ostalih kartica skrivene. Kartica za upravljanje žaruljom je podijeljena na tri dijela. U dijelu RUČNO UPRAVLJANJE su dvije tipke za uključivanje (tipka "Žarulja ON") i isključivanje (tipka "Žarulja OFF") žarulje. Kod dijela OPTIMALNO SVJETLO, ukoliko korisnik pritisne na tipki "OPTIMALNO SVJETLO" tada se provjerava stanje na fotootporniku te ako je premalo svijetlosti u okolini, relej uključi žarulju, u suprotnom je isključi. U dijelu PRIKAZ STANJA NA SENZORU, ukoliko korisnik pritisne "Zahtjev za prikaz stanja na senzoru" (*slika 24.* lijevo), na *Arduino* se šalje zahtjev za ispisivanje trenutno stanja na fotootporniku, nakon toga se prijašnja tipka zamjenjuje tipkom "Pritisnite za prikaz stanja na senzoru" (*slika 24.* desno). Ukoliko korisnik pritisne ovu tipku, ispod tipke se ispiše stanje te se ujedno sprema u tekstualnu datoteku na mobilnom uređaju i tipka se ne može ponovo pritisnuti tako dugo dok se barem jedanput ne promijeni kartica (*eng. tab*), nakon čega se korisniku ponovo vraća tipka za slanje zahtjeva za ispisivanje stanja na fotootporniku. Ispod prikaza stanja na senzoru nalazi se tekst koji korisniku daje povratnu informaciju o stanju izlaza, tj. žarulje.



Slika 25. Kartica za prikaz stanja na temperaturnom senzoru

Ako korisnik odabere karticu za prikaz stanja temperature, pozadina kartice promijeni boju u bijelu i sve komponente za prikaz stanja na temperaturnom senzoru postanu vidljive, dok su komponente ostalih kartica skrivene. Ukoliko korisnik želi da se ispiše stanje na temperaturnom senzoru, prvo mora pritisnuti tipku "Slanje zahtjeva za temperaturu" (*slika 25.* lijevo), čime se na Arudino šalje zahtjev za ispisivanje stanja na temperaturnom senzoru. Nakon toga se tipka slanja zahtjeva zamjenjuje tipkom "Pritisnite za prikaz temperature:" (*slika 25.* desno). Pritiskom na novu tipku, ispod se ispisuje stanje na temperaturnom senzoru. Tipku je nemoguće pritisnuti više od jedanput iz razloga što bi se opet ispisalo isto stanje kao i prvi put. Zbog toga je potrebno ponovo vratiti tipku za slanje zahtjeva za prikaz temperature. Proces je moguće ponoviti promjenom kartice i vraćanjem nazad na istu.



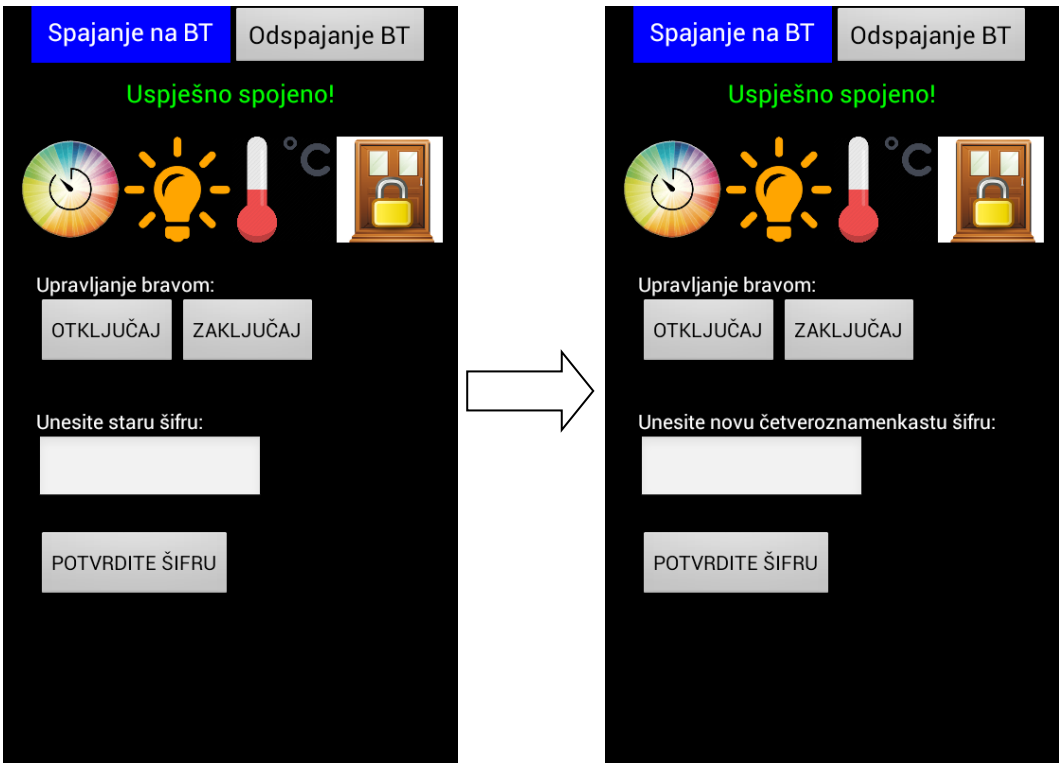
a) b)

Slika 26. Kartica za upravljanje bravom

a) Zaštićena lozinkom

b) Nakon upisane točne lozinke

Kartica za upravljanje bravom (*slika 26.* a) se razlikuje od prijašnjih kartica zato što se tipke za upravljanje bravom pojavljuju tek nakon upisivanja točne lozinke za pristup kartici. Ukoliko se upiše pogrešna lozinka korisnik dobije povratnu informaciju za ponovni pokušaj unosa lozinke. Na kartici se nalaze dvije tipke za upravljanje bravom (*slika 26. b*). Tipka "OTKLJUČAJ" za otključavanje brave (uključivanje zelene LED diode) i tipka "ZAKLJUČAJ" za zaključavanje brave (uključivanje crvene LED diode).



Slika 27. Promjena lozinke

Ukoliko korisnik odluči promijeniti pristupnu lozinku te pritisne na tipku "PROMJENA ŠIFRE", pojavljuje se tekstni okvir za unos stare lozinke zbog sigurnosne provjere identiteta korisnika koji želi promijeniti lozinku (*slika 27.* lijevo). Nakon točnog unosa stare lozinke i potvrde unosa, pojavljuje se novi okvir za unos lozinke u koji se unosi nova željena lozinka (*slika 27.* desno). Nova lozinka mora sadržati kombinaciju od četiri slova, znaka ili broja. Ukoliko lozinka nije četveroznamenkasta, korisnik dobije povratnu informaciju za ponovni unos.

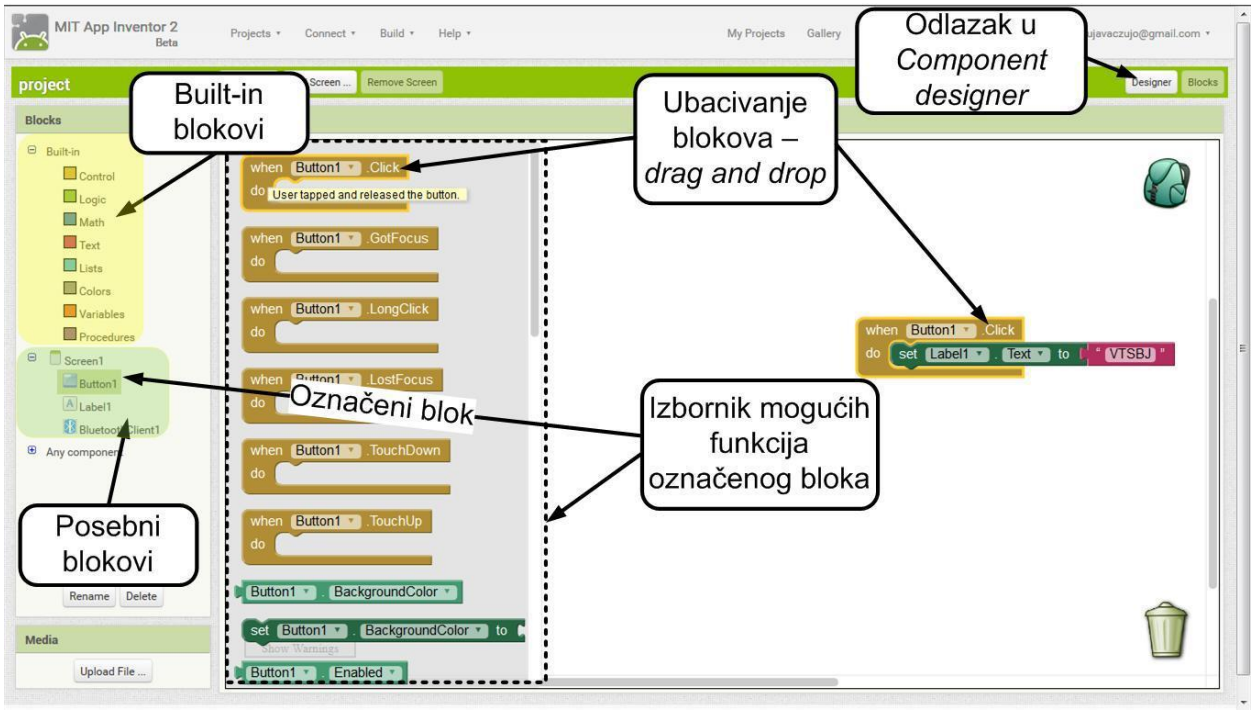
5.3. Logički dio aplikacije

Klikom na *Blocks* karticu ulazi se u uređivač blokova (eng. *Block editor*). Zadaća uređivača blokova je stvaranje programske logike, tj. dodjeljivanje radnji komponentama koje su ubačene u dizajneru komponenata. Blokovi određuju kako se komponente trebaju ponašati prilikom neke radnje s njima te se kombinacijom blokova kreiraju funkcije potrebne za izvršavanje zadaće aplikacije. Pomoću blokova se početnicima bez poznavanja programskog jezika olakšava stvaranje aplikacije. Unutar uređivača definira se funkcionalnost komponenata ubačenih u dizajneru komponenata.

Blokovi su podijeljeni u dvije kategorije:

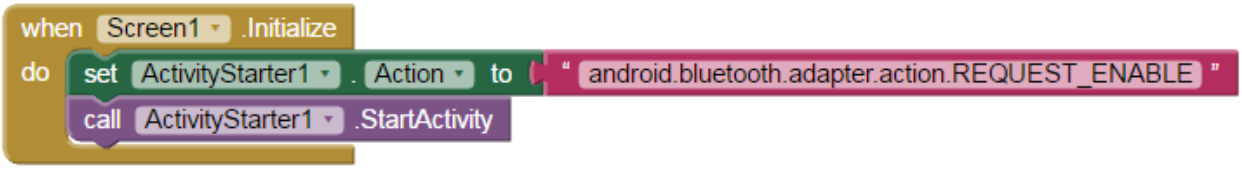
1) *Built-in* blokovi – uvijek su dostupni

2) posebni blokovi – dostupni tek kada se ubaci pojedina komponenta unutar dizajnera.



Slika 28. Blocks komponenta MIT App inventora 2

Na samom početku korištenja blokova potrebno je definirati događaje prilikom učitavanja početnog zaslona aplikacije.



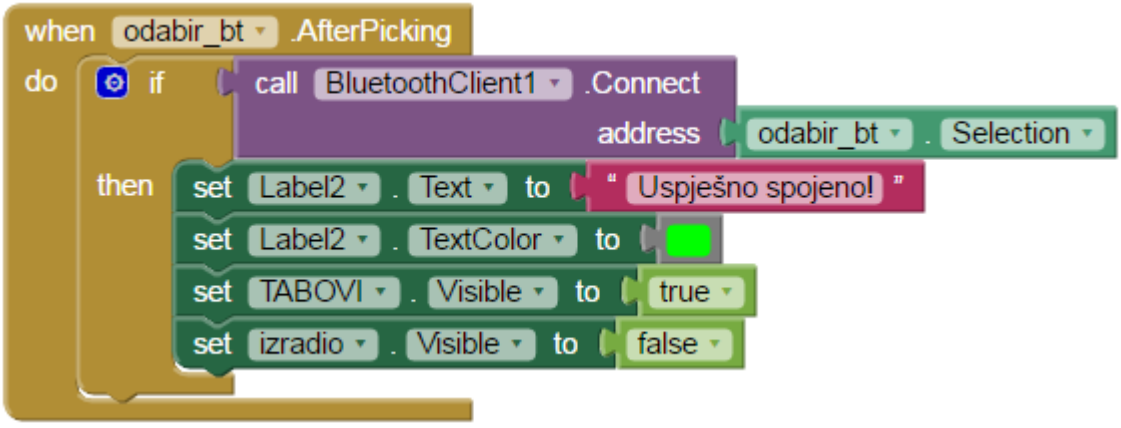
Slika 29. Brzo uključivanje Bluetooth veze

Prilikom učitavanja početnog zaslona aplikacije koristi se naredba (*android.bluetooth.adapter.action.REQUEST_ENABLE*) za brzo uključivanje *Bluetooth* veze na mobilnom uređaju ukoliko on već nije uključen.

Nakon uključivanja *Bluetooth* veze potrebno je spojiti mobilni uređaj s *Bluetooth* modulom. Ta veza se ostvaruje pritiskom na listu "Spajanje na BT". Zbog navedenog, aplikacija mora učitati popis uparenih *Bluetooth* uređaja prije nego korisnik pritisne na listu "Spajanje na BT".

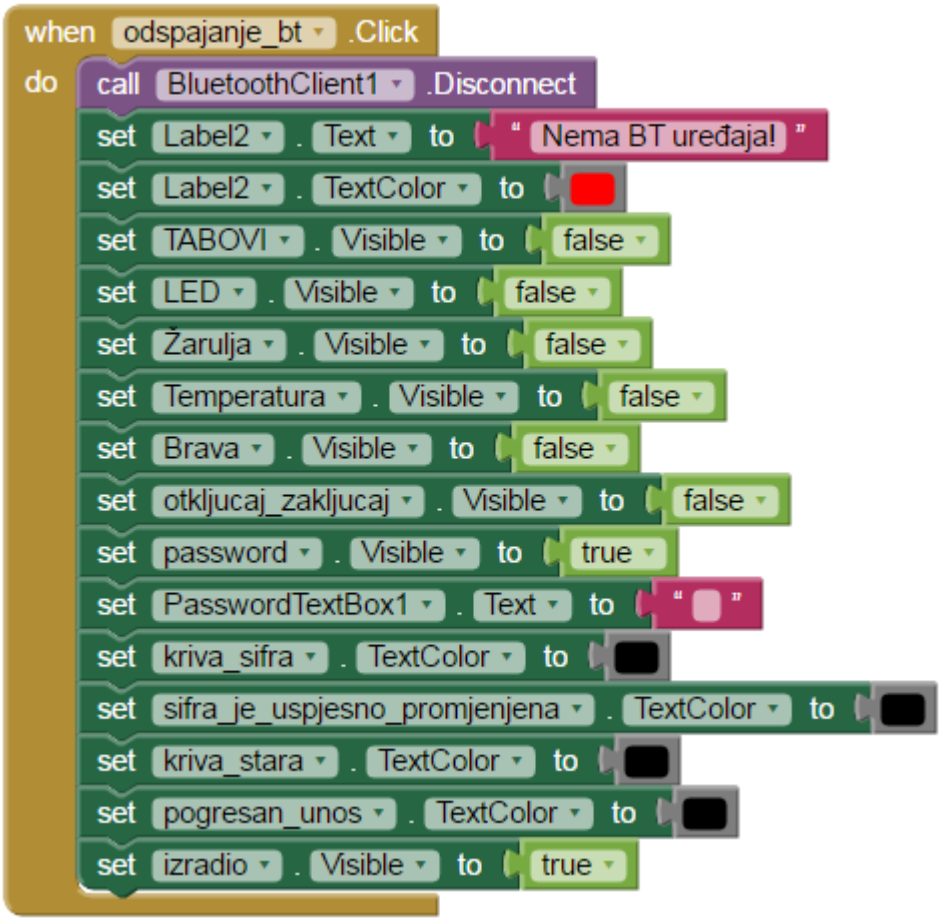
Slika 30. Učitavanje popisa uparenih Bluetooth uređaja na mobilnom uređaju

Zatim je potrebno definirati događaje koji će slijediti nakon što se odabere željeni *Bluetooth* modul. Dakako, najbitnije je spajanje na odabrani *Bluetooth* modul. Nakon toga slijedi tekstna povratna informacija o stanju veze (*Slika 31. set Label2*) i postavljanje vidljivosti za određene komponente (Kartice postanu vidljive, a informacije o izradi aplikacije nevidljive).



Slika 31. Uspostavljanje Bluetooth veze i događaji nakon toga

Također valja definirati tipku za prekid *Bluetooth* veze i događaje nakon pritiska na tipku "Odspajanje BT".

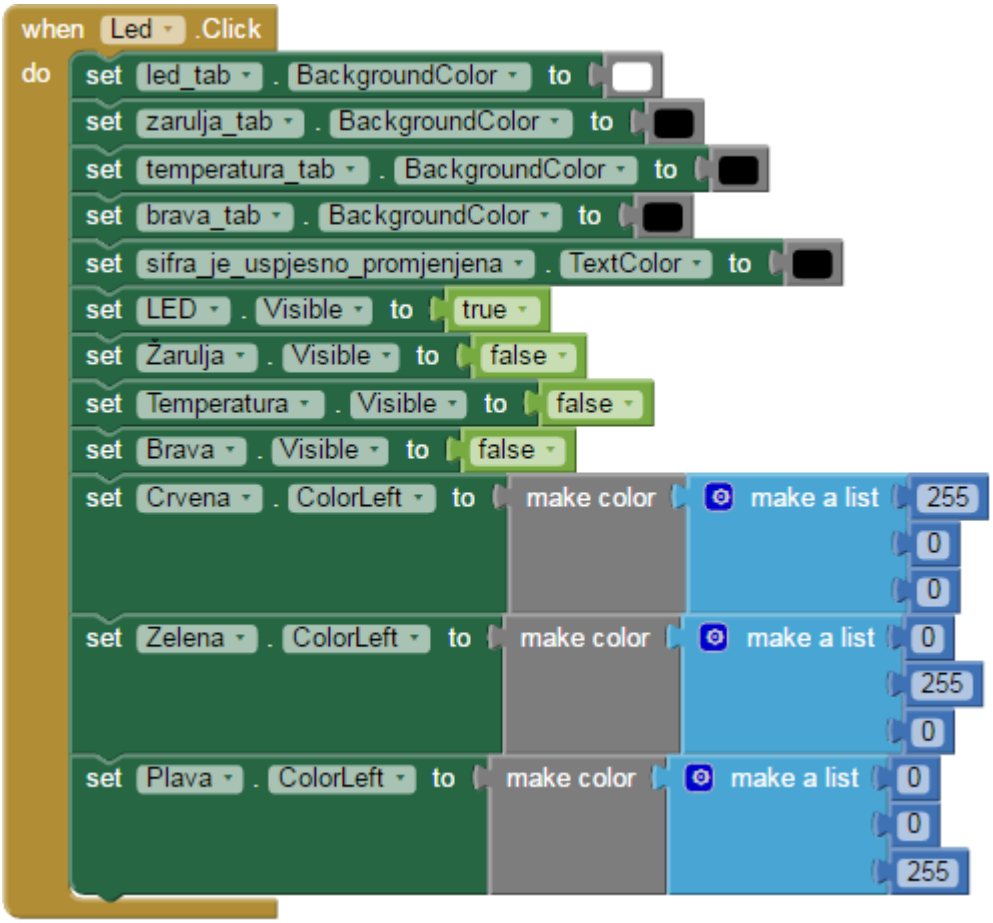


Slika 32. Prekid Bluetooth veze

Ukoliko korisnik pritisne na tipku "Odspajanje BT" prekida se *Bluetooth* komunikacija između mobilnog uređaja i *Bluetooth* modula (*call BluetoothClient1.Disconnect*). Tekst o stanju veze se postavlja na početno stanje (*set Label2.Tex to "Nema BT uređaja!"*). Isključuju se vidljivosti za sve komponente koje nisu na početnom zaslonu prilikom uključivanja aplikacije, a informacije o aplikaciji ponovo postanu vidljive. Također treba sakriti sve povratne informacije upozorenja ukoliko je prilikom korištenja došlo do njih te je potrebno očistiti tekstni okvir za unos lozinke radi sigurnosti.

5.3.1. LED traka

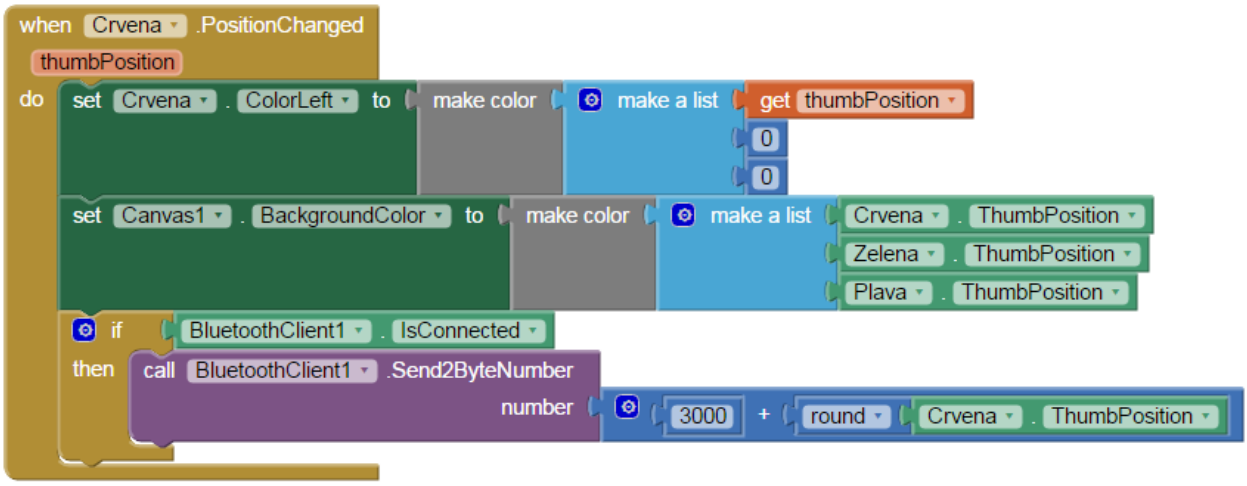
Definiranje blokova LED trake započinje promjenom boje pozadine kartice (*eng. tab*) LED trake, zatim se uključuje vidljivost komponenata potrebnih za kontrolu LED trake i isključuje vidljivost ostalih komponenata. Nakon toga, potrebno je definirati liste klizača triju glavnih boja (crvene, zelene i plave).



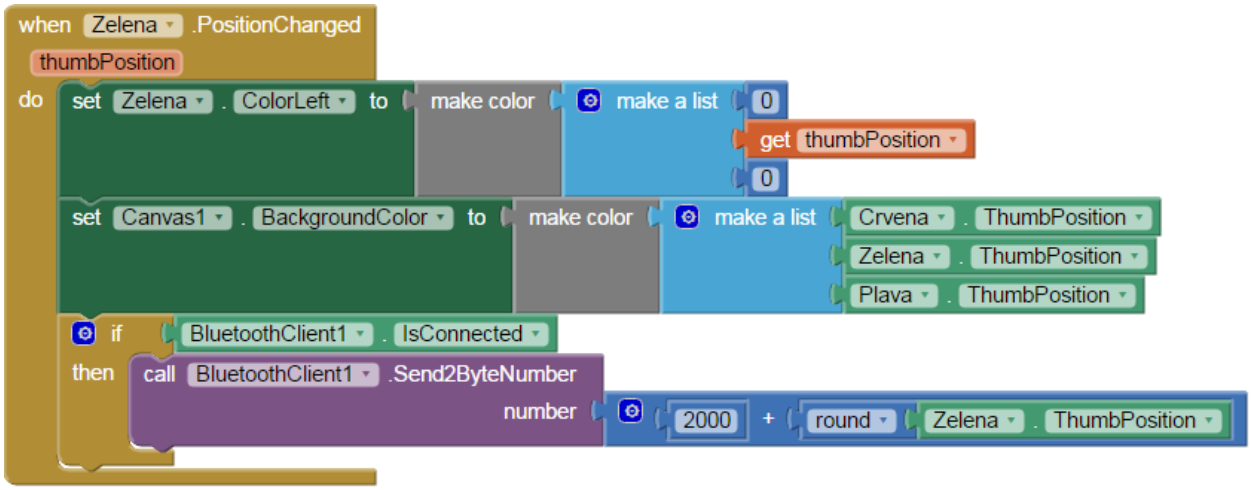
Slika 33. Definiranje funkcija pritiskom na karticu za upravljanje LED trakom

Budući da LED kartica ima tri klizača, svaki za jednu od tri glavne boje, potrebno je definirati svaki klizač posebno. Kod definiranja klizača potrebno je za svaku boju posebno uzimati vrijednosti sa klizača ovisno o položaju u kojem se nalazi tj. položaju u koji ga je korisnik stavio preko ekrana na dodir (definirano blokom "*get thumbPosition*"). Uz promjenu položaja trenutno korištenog klizača, aplikacija mora pamtiti položaje preostalih dvaju klizača. Iznad klizača nalazi se platno (*eng. canvas*), na kojem korisnik preko mobilnog uređaja može vidjeti trenutnu boju na LED traci. Kako bi se definirala boja koja će se nalaziti na platnu, uzimaju se vrijednosti svih triju klizača i "miješanjem" boja, odnosno provjerom jačine

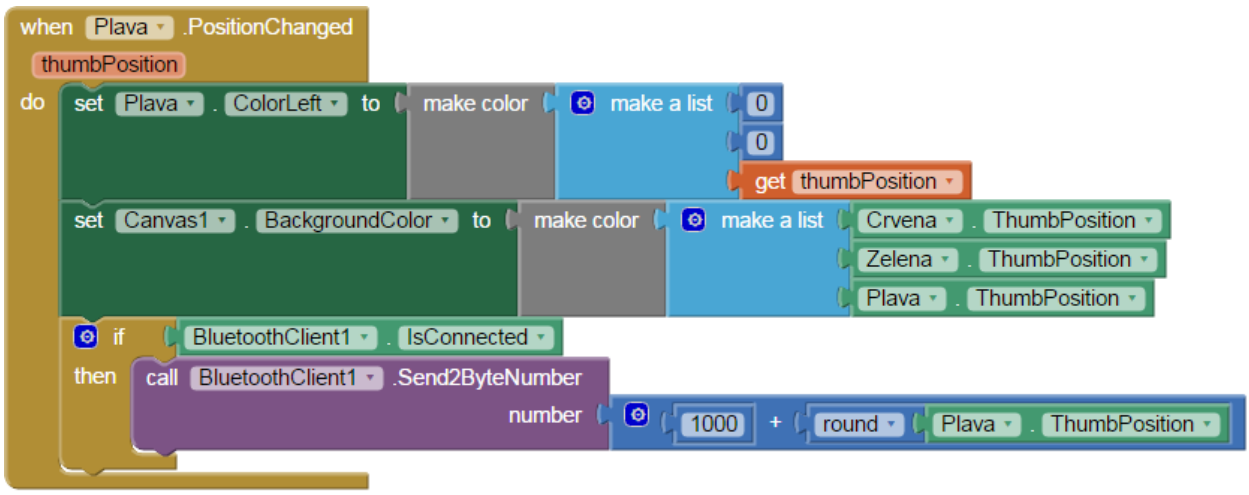
intenziteta svake od triju glavnih boja dobije se prikaz trenutne boje na LED traci. Nakon definiranja klizača i platna, potrebno je poslati vrijednosti na *Arduino*, tj. *Bluetooth* modul. Slanje vrijednosti odvija se preko bloka i funkcije *call BluetoothClient1.Send2ByteNumber*. U nastavku bloka potrebno je odrediti koji će se brojevi slati za određivanje vrijednosti pojedine boje. Prema tome, kod crvene boje se šalje *3000 + round Crvena.ThumbPosition*, tj. 3000 - 3255, kod zelene boje *2000 + round Zelena.ThumbPosition*, tj. 2000 - 2255 dok se kod plave boje šalje *1000 + round Plava.ThumbPosition*, tj. 1000 - 1255. Svaka boja ima svoju tisućicu iz razloga da se izbjegne kolizija podataka.



Slika 34. Definiranje klizača za crvenu boju

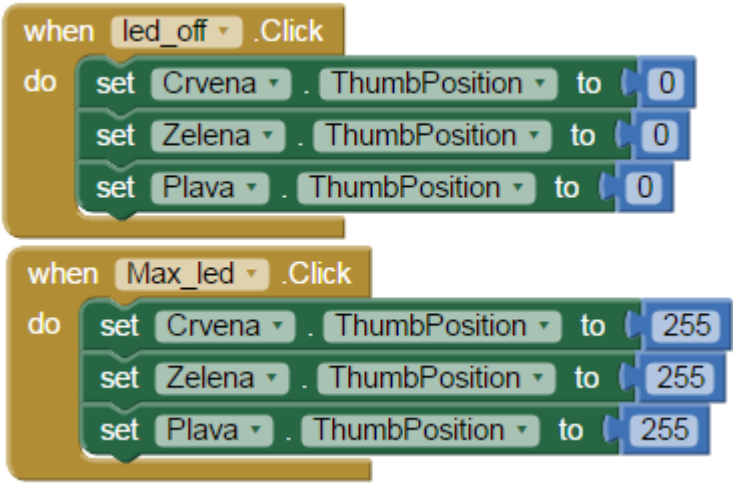


Slika 35. Definiranje klizača za zelenu boju



Slika 36. Definiranje klizača za plavu boju

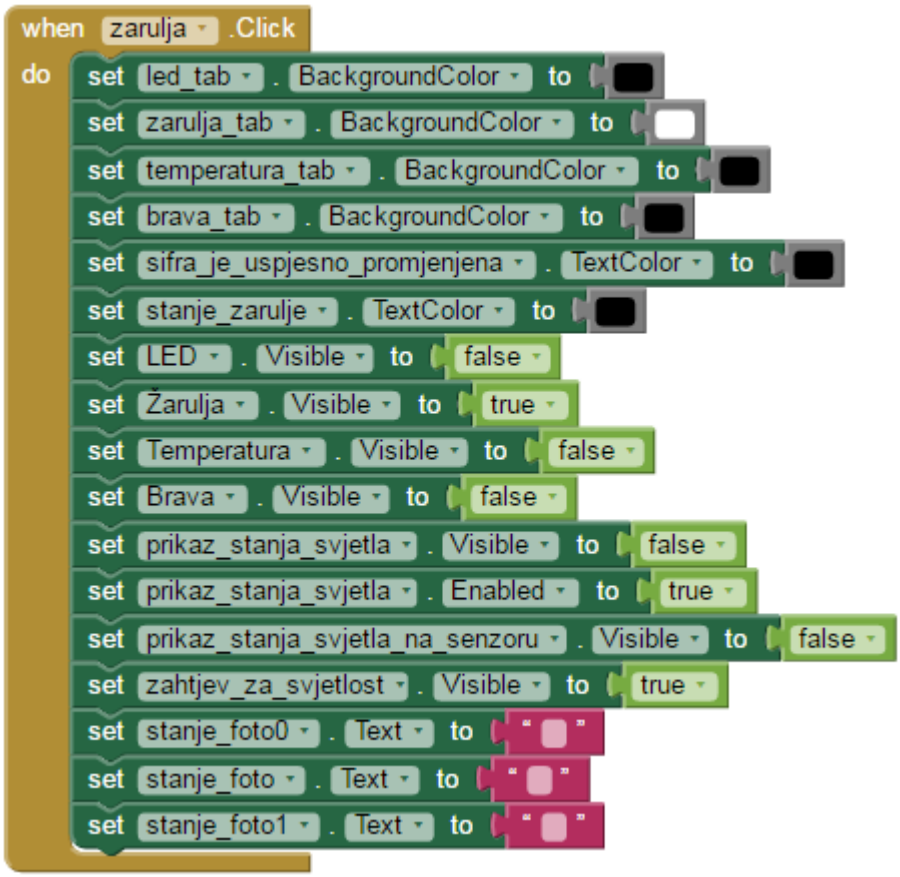
Uz klizače, na kartici na upravljanje LED trakom nalaze se dvije tipke za brže namještanje klizača. Tipka "OFF" postavlja klizače na minimalnu vrijednost (0), odnosno gasi LED traku. Tipka "MAX" postavlja klizače na maksimalnu vrijednost (255), te se time dobije najjača svijetlost LED trake.



Slika 37. Definiranje tipki "OFF" i "MAX"

5.3.2. Relejni modul (žarulja)

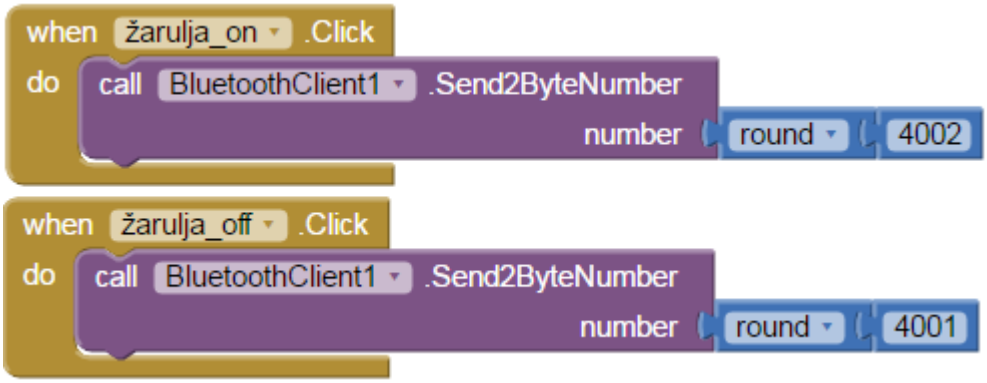
Definiranje blokova za upravljanje relejnim modulom također započinje promjenom boje pozadine kartice (*eng. tab*), zatim se uključuje vidljivost komponenata potrebnih za kontrolu žarulje i isključuje se vidljivost ostalih komponenata. Uz definiranja vidljivosti pojedinih komponenata potrebno je izbrisati vrijednosti koje se nalaze na fotootporniku, odnosno vrijednosti na tekstualnim oznakama (*eng. label*).



Slika 38. Definiranje funkcija pritiskom na karticu za upravljanje žaruljom i fotootpornikom

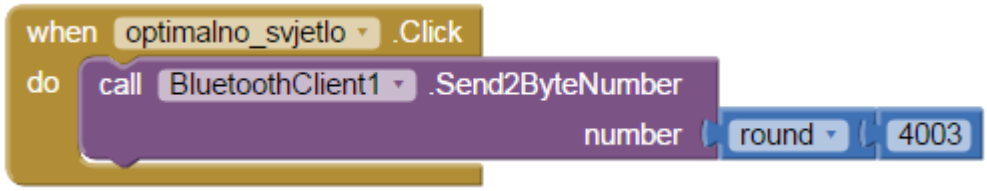
Kartica za upravljanje žaruljom je podijeljena na tri dijela. U dijelu RUČNO UPRAVLJANJE su dvije tipke za uključivanje (tipka "Žarulja ON") i isključivanje (tipka "Žarulja OFF") žarulje. Potrebno je poslati vrijednosti na *Arduino*, tj. *Bluetooth* modul. Slanje vrijednosti odvija se preko bloka i funkcije *call BluetoothClient1.Send2ByteNumber*. U nastavku bloka potrebno je odrediti koji će se brojevi slati za pojedine akcije. Tako se kod pritiska tipke za

uključivanje ("Žarulja ON"), na *Arduino* šalje broj 4002. Pritiskom tipke za isključivanje ("Žarulja OFF"), na *Arduino* se šalje broj 4001.



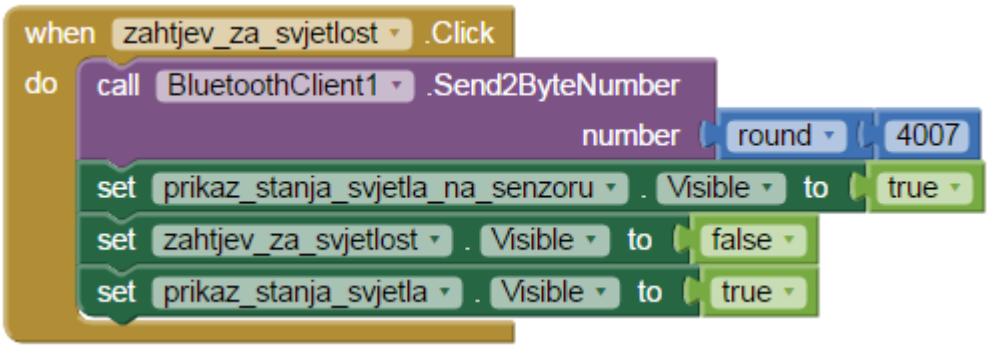
Slika 39. Definiranje tipki za uključivanje i isključivanje žarulje

Kod dijela OPTIMALNO SVJETLO, ukoliko korisnik pritisne na tipku "OPTIMALNO SVJETLO" tada se provjerava stanje na fotootporniku te ako je premalo svijetlosti u okolini, relej uključi žarulju, u suprotnom je isključi. Slanje vrijednosti se također odvija preko bloka i funkcije *call BluetoothClient1.Send2ByteNumber*. Pritiskom tipke "OPTIMALNO SVJETLO" na *Arduino* se šalje broj 4003.



Slika 40. Definiranje tipke "OPTIMALNO SVJETLO"

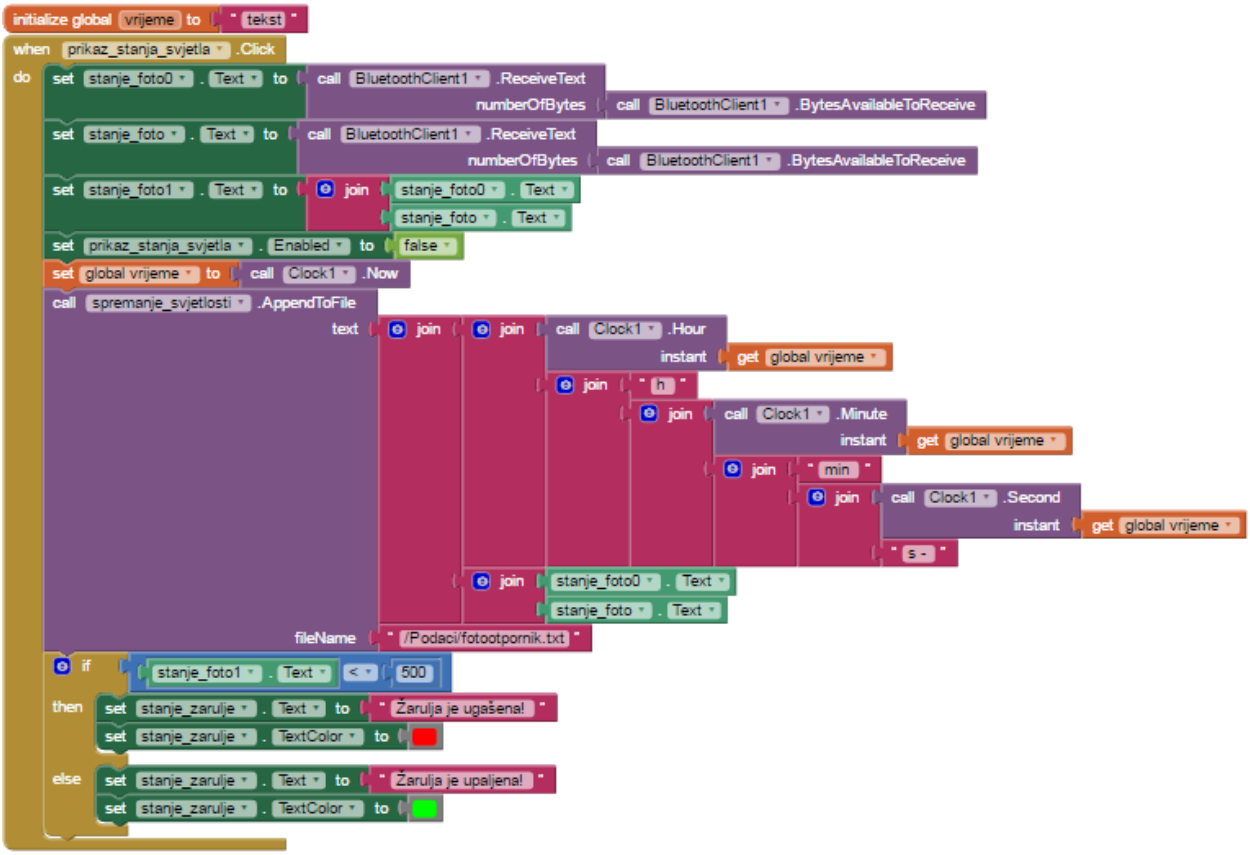
U dijelu PRIKAZ STANJA NA SENZORU ukoliko korisnik pritisne "Zahtjev za prikaz stanja na senzoru" (*slika* 41.), na *Arduino* se šalje zahtjev za ispisivanje trenutnog stanja na fotootporniku, tj. broj 4007. Također se definiraju vidljivosti pojedinih komponenata te se prijašnja tipka zamjenjuje tipkom "Pritisnite za prikaz stanja na senzoru".



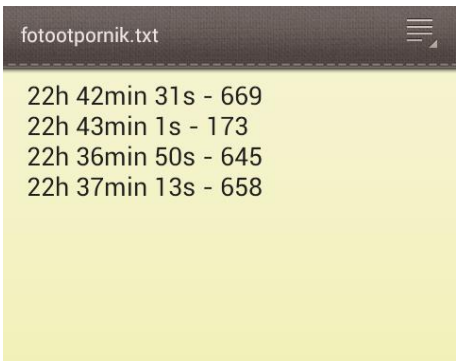
Slika 41. Definiranje tipke "Zahtjev za prikaz stanja na senzoru"

Ukoliko korisnik pritisne novu tipku, tj. " Pritisnite za prikaz stanja na senzoru", ispod tipke se ispiše stanje na senzoru te se ujedno sprema u tekstualnu datoteku na mobilnom uređaju i tipka se ne može ponovo pritisnuti (*set prikaz_stanja_svjetla.Enabled to false*) tako dugo dok se barem jedanput ne promjeni kartica (*eng. tab*), nakon čega se korisniku ponovo vraća tipka za slanje zahtjeva za ispisivanje stanja na fotootporniku. Da bi korisnik mogao vidjeti stanje na fotootporniku, potrebno je učitati vrijednost koja se je pritiskom na prijašnju tipku (*Slika 41.)* ispisala u *Arduino Serial Monitor*. Kako bi se osiguralo sigurno i točno učitavanje vrijednosti najbolje je proces napraviti dva puta za stanja foto i foto0 (*set stanje_foto0.Text to call BluetoothClient1.ReceiveText*) te nakon toga vrijednosti spojiti u jedan tekst (*join stanje_foto0.Text || stanje foto.Text*). Nadalje slijedi spremanje podataka na mobilni uređaj. Da bi se podaci spremali ovisno o vremenu u kojem su učitani, potrebno je ponajprije dodati komponentu "Sat" (*eng. Clock*) i definirati globalnu varijablu vrijeme. (*initialize global vrijeme to "tekst"*). Zatim je potrebno na globalnu varijablu spremiti trenutno vrijeme (*set global vrijeme to call Clock1.Now*). Nakon toga slijedi definiranje teksta koji će se spremiti i lokacije na koju će se spremiti, što se definira u nastavku bloka *call spremanje_svjetlosti.AppendToFile*. U nastavku "text" dijela iz globalne varijable se uzimaju podaci o tome koliko je sati, minuta i sekundi te se spremaju zajedno s prethodno učitanim vrijednostima na fotootporniku. Lokacija na koju će se podaci spremiti određuje se u nastavku "fileName" dijela. Konkretno u ovom slučaju podaci se spremaju na lokaciju /Podaci/fotootpornik.txt, tj. prva kosa crta (/) određuje spremanje na SD karticu, zatim slijedi ime datoteke (Podaci) te naziv tekstualne datoteke u koju će se spremiti vrijednost (fotootpornik). Zatim je još potrebno definirati povratnu informaciju o stanju releja, tj. žarulje. Definiranje stanja releja se izvršava provjerom jakosti svijetla u okolini, tj. vrijednosti na fotootporniku u neposrednoj blizini žarulje. Ako je vrijednost veća od 500

pretpostavlja se da je žarulja upaljena i ispisuje se tekst zelene boje "Žarulja je upaljena!". Ukoliko je vrijednost manja od 500 ispisuje se tekst crvene boje "Žarulja je ugašena!"



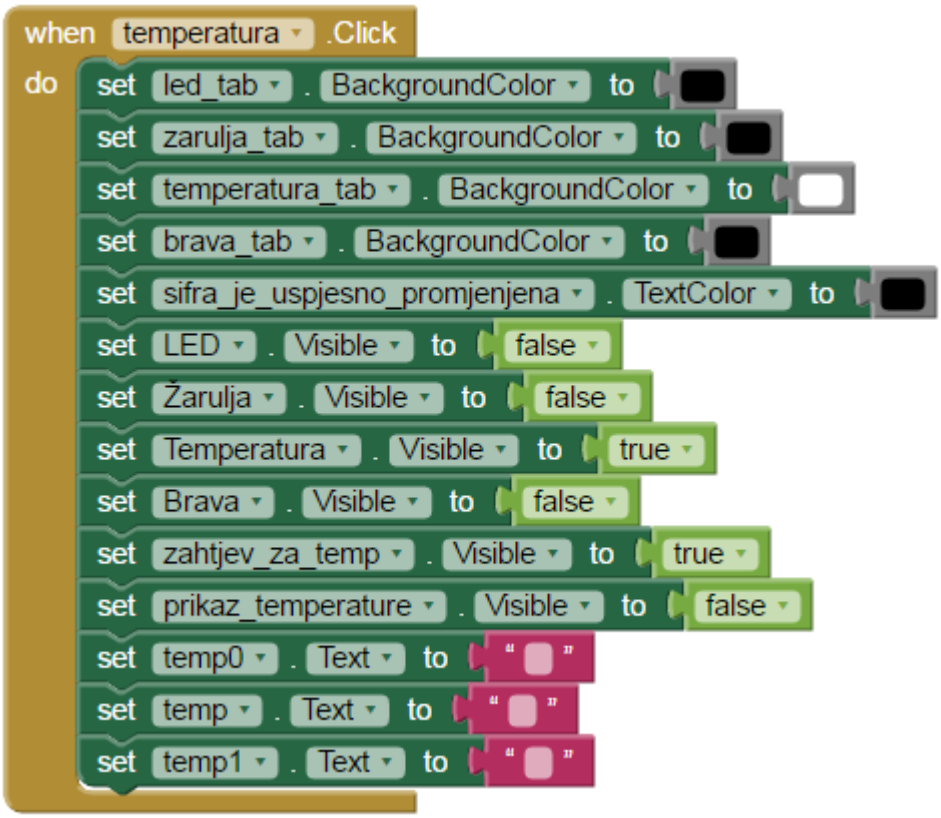
Slika 42. Očitavanje vrijednosti na fotootporniku i spremanje na mobilni uređaj



Slika 43. Izgled spremljenih podataka o jakosti svjetla

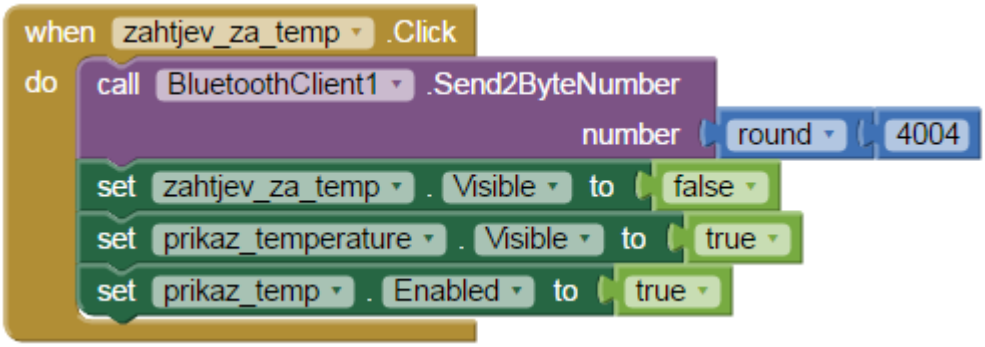
5.3.3. Temperatura

Definiranje blokova za očitavanje vrijednosti na temperaturnom senzoru započinje promjenom boje pozadine kartice (*eng. tab*), zatim se uključuje vidljivost komponenata potrebnih za slanje zahtjeva za očitavanje vrijednosti i postupak očitavanja vrijednosti dok se isključuje vidljivost ostalih komponenata. Uz definiranja vidljivosti pojedinih komponenata potrebno je izbrisati vrijednosti koje se nalaze na temperaturnom senzoru, odnosno vrijednosti na tekstualnim oznakama (*eng. label*).



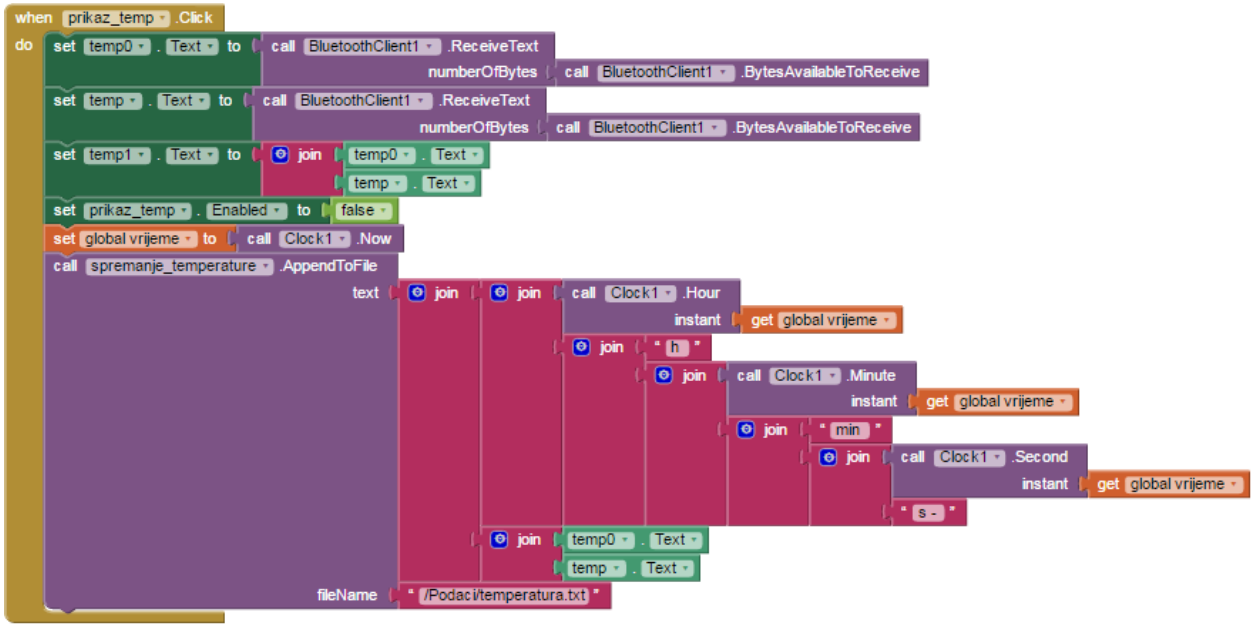
Slika 44. Definiranje funkcija pritiskom na karticu za očitavanje vrijednosti na temperaturnom senzoru

Ako korisnik pritisne "Slanje zahtjeva za temperaturu" (*slika 44.*), na *Arduino* se šalje zahtjev za ispisivanje trenutnog stanja na temperaturnom senzoru, odnosno broj 4004. Također se definiraju vidljivosti i dostupnosti pojedinih komponenata te se prijašnja tipka zamjenjuje tipkom "Pritisnite za prikaz temperature: "



Slika 45. Definiranje tipke "Slanje zahtjeva za temperaturu"

Ukoliko korisnik pritisne tipku " Pritisnite za prikaz temperature: ", ispod tipke se ispiše stanje na senzoru te se ujedno sprema u tekstualnu datoteku na mobilnom uređaju i tipka se ne može ponovo pritisnuti (*set prikaz_temp.Enabled to false*) tako dugo dok se barem jedanput ne promjeni kartica (*eng. tab*). Nakon toga se korisniku ponovo vraća tipka za slanje zahtjeva za ispisivanje stanja na temperaturnom senzoru. Da bi korisnik mogao vidjeti stanje na temperaturnom senzoru, potrebno je učitati vrijednost koja se je pritiskom na prijašnju tipku (Slika 45.) ispisala u *Arduino Serial Monitor*. Kako bi se osiguralo sigurno i točno učitavanje vrijednosti najbolje je proces napraviti dva puta za stanja temp i temp0 (*set stanje_temp0.Text to call BluetoothClient1.ReceiveText*) i (*set stanje_temp.Text to call BluetoothClient1.ReceiveText*) te nakon toga vrijednosti spojiti u jedan tekst (*join stanje_temp0.Text || stanje temp.Text*). Zatim slijedi spremanje podataka na mobilni uređaj. Podaci o temperaturi se također spremaju ovisno o vremenu u kojem su učitani, budući da već od ranije postoji globalna varijabla "vrijeme" nije potrebno definirati novu varijablu već se ista može koristiti za sva mjesta gdje je potrebno učitavanje vremena u aplikaciji. Potrebno je samo na globalnu varijablu spremiti trenutno vrijeme (*set global vrijeme to call Clock1.Now*). Nakon toga slijedi definiranje teksta koji će se spremiti i lokacije na koju će se spremiti, što se definira u nastavku bloka *call spremanje_temperature.AppendToFile*. U "text" dijelu iz globalne varijable se također kao i kod svjetlosti uzimaju podaci o tome koliko je sati, minuta i sekundi nakon čega se spajaju zajedno s prethodno učitanim vrijednostima na temperaturnom senzoru. Lokacija na koju će se podaci spremiti određuje se u nastavku "fileName" dijela. U ovom radu podaci se spremaju na lokaciju /Podaci/temperatura.txt, tj. prva kosa crta (/) određuje spremanje na SD karticu, zatim slijedi ime datoteke (Podaci) te naziv tekstualne datoteke u koju će se spremiti vrijednost (temperatura).



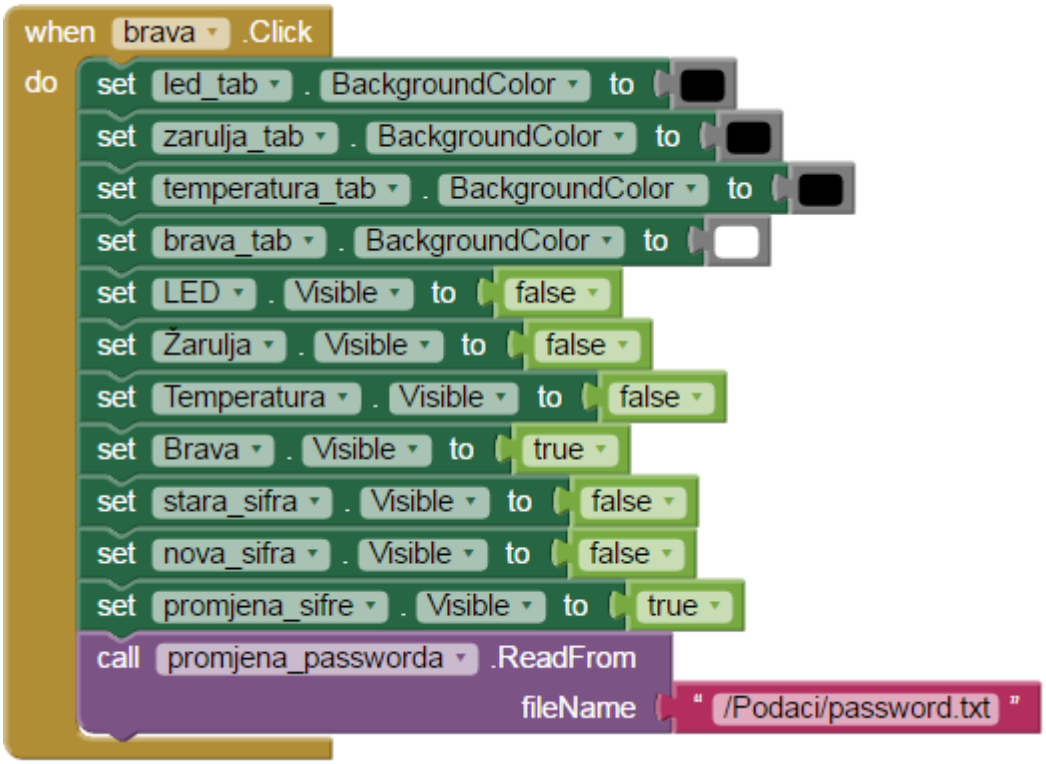
Slika 46. Očitavanje vrijednosti na temperaturnom senzoru i spremanje na mobilni uređaj

Slika 47. Izgled spremljenih podataka o temperaturi

5.3.4. Brava

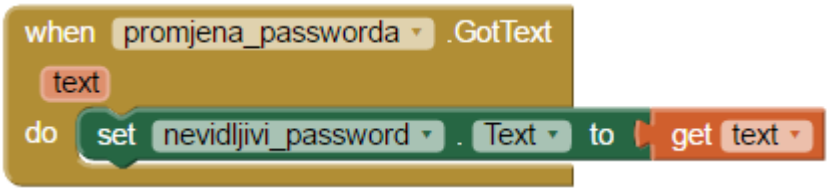
Definiranje blokova za upravljanje bravom započinje promjenom boje pozadine kartice (eng. tab), zatim se uključuje sigurnosna provjera korisnika mobilnog uređaja. Kako bi se sigurnosna provjera mogla uspješno izvršiti i kako bi korisnik ujedno imao mogućnost promjene lozinke, potrebno je učitati lozinku koja se trenutno nalazi u spremljenoj tekstualnoj datoteci na mobilnom uređaju (*call promjena_passworda.ReadFrom fileName "/Podaci/password.txt"*). Ukoliko korisnik odluči promijeniti lozinku, nova lozinka se sprema u istu datoteku ali o tome

više u nastavku ovog rada. Uz sigurnosnu provjeru mijenjaju se postavke vidljivosti pojedinih komponenata.



Slika 48. Definiranje funkcija pritiskom na karticu za upravljanje bravom

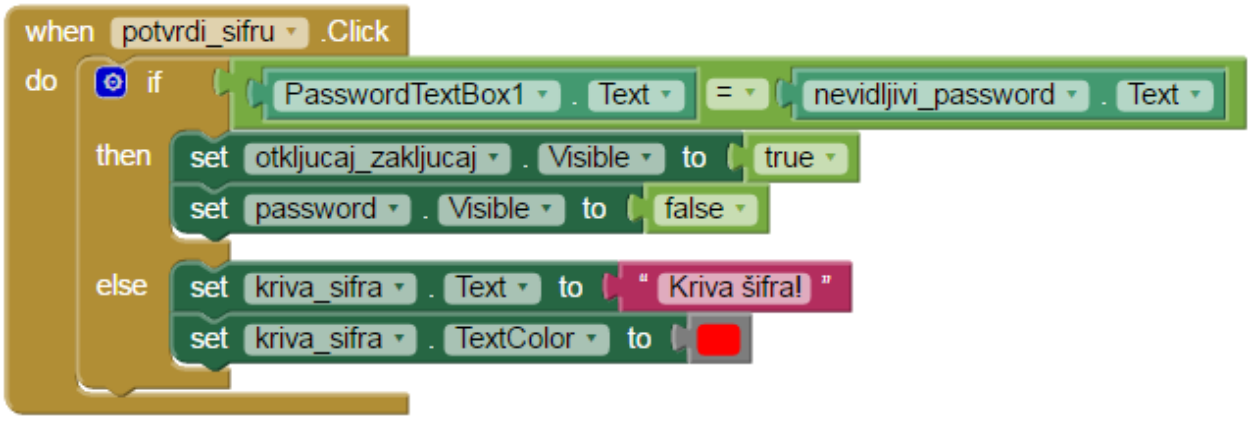
Nakon što se učita lozinka iz tekstualne datoteke, potrebno ju je staviti u aplikaciju. U ovom radu to je realizirano na način da se lozinka učita u tekstualnu oznaku (*set nevidljivi_password.Text to get text*) ali ona je nevidljiva za sve korisnike.



Slika 49. Učitavanje lozinke iz mobilnog uređaja u aplikaciju

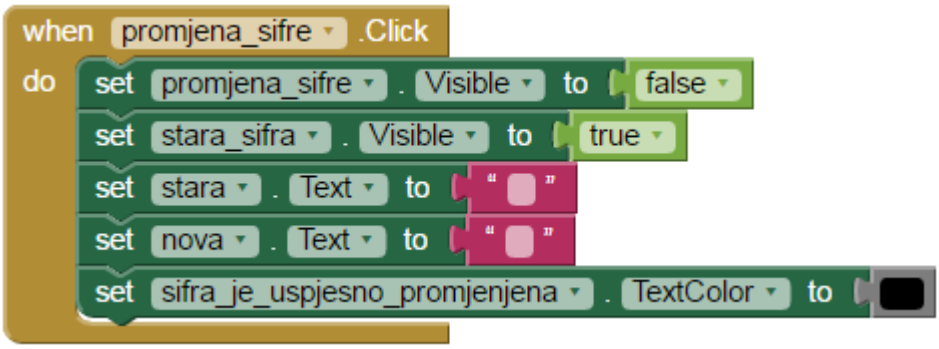
Da bi se pojavile tipke za upravljanje bravom, potrebno je upisati točnu lozinku u tekstualni okvir za lozinku i pritisnuti tipku "POTVRDITE ŠIFRU". Nakon čega se izvršava provjera točnosti upisane lozinke (*if PasswordTextBox1.Text = nevidljivi_password.Text*). Ako je upisana

lozinka točna uključuje se vidljivost tipki za upravljanje bravom, a ukoliko upisana lozinka nije točna korisnik dobije povratnu informaciju da je upisana lozinka pogrešna. (*set kriva_sifra.Text to "Kriva šifra!"*)



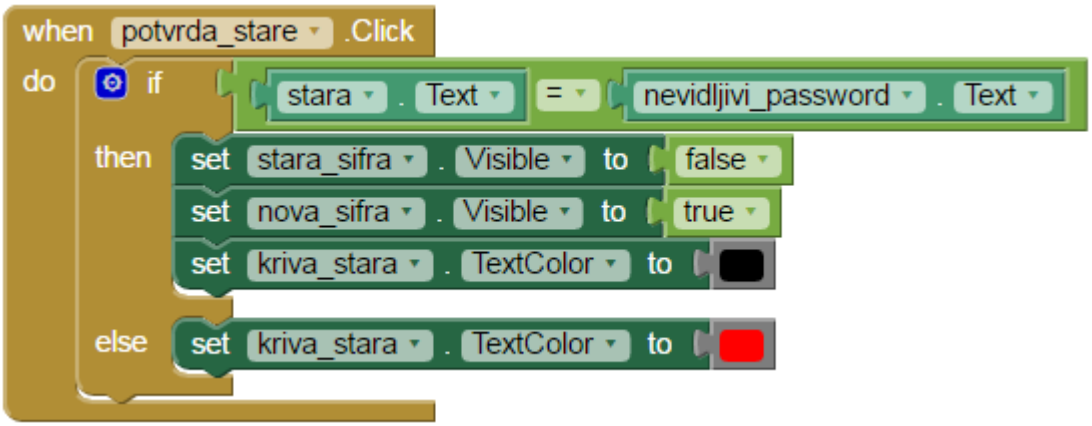
Slika 50. Definiranje tipke za potvrdu upisane lozinke

Nakon što se upiše i potvrdi točna lozinka, uključuju se vidljivosti za tipke "OTKLJUČAJ", "ZAKLJUČAJ" i "PROMJENA ŠIFRE". Ukoliko korisnik pritisne "PROMJENA ŠIFRE" mijenjaju se postavke o vidljivostima pojedinih komponenata, pojavljuje se tekstni okvir za upis stare lozinke zbog provjere identiteta korisnika koji želi promijeniti lozinku i prazne se tekstni okviri za lozinke.



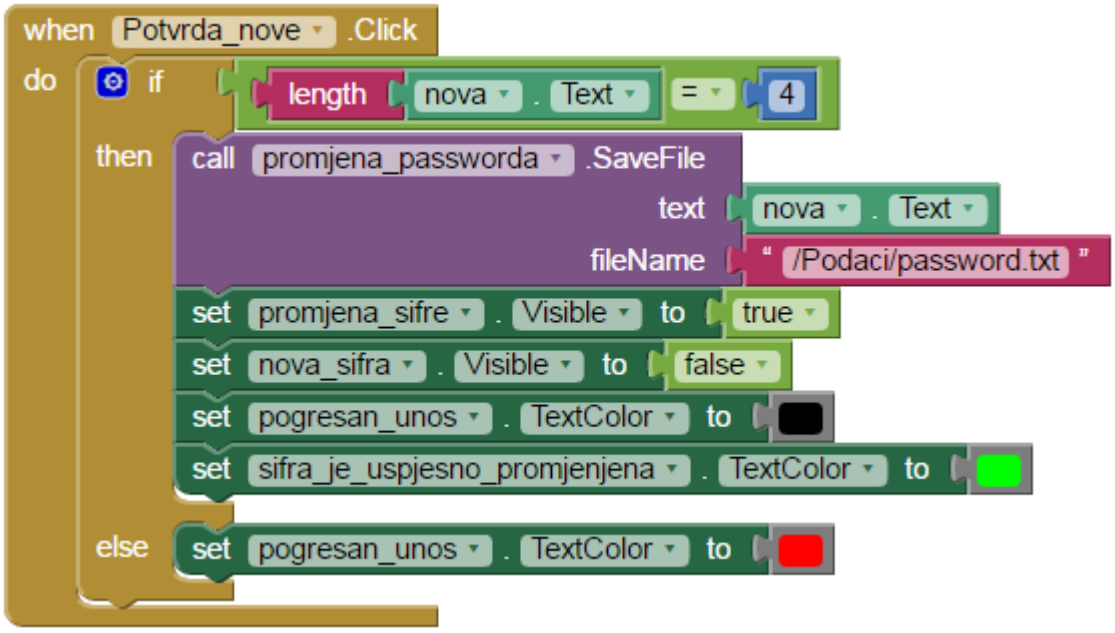
Slika 51. Definiranje tipke za promjenu lozinke

Nakon toga je potrebno ponovo upisati točnu staru lozinku, a proces se odvija na isti način kao i prije samog početka upravljanja bravom. Jedina razlika je u tome što se potvrdom stare lozinke u procesu promjene lozinke, pojavljuje tekstni okvir za upis nove lozinke.



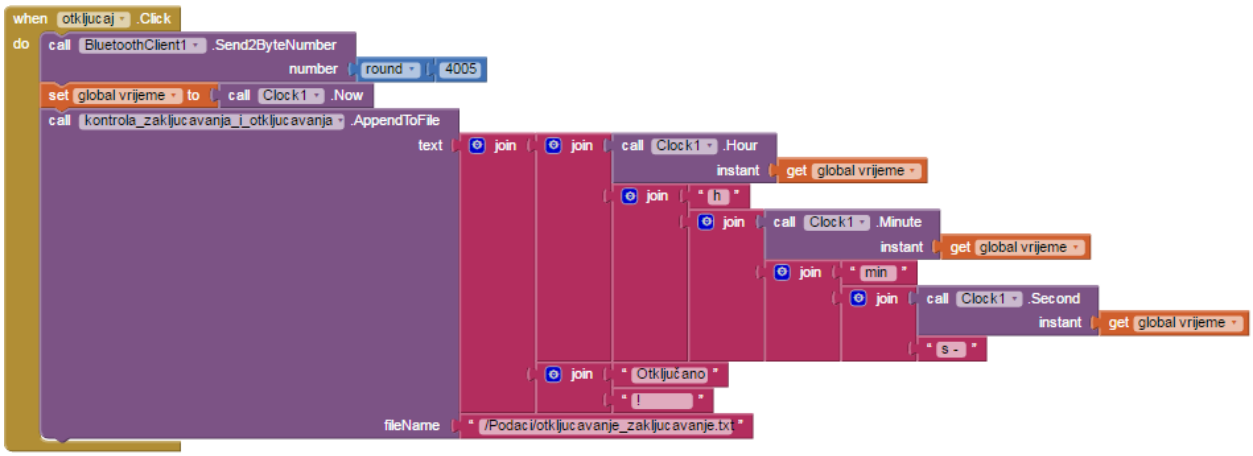
Slika 52. Potvrda stare lozinke u procesu promjene lozinke

Kod upisa nove lozinke potrebno je paziti da se upiše četveroznamenkasta lozinka. Ako korisnik upiše manje od četiri znamenke ili više od četiri znamenke, na ekranu se pojavljuje povratna informacija o pogrešnom unosu. Ukoliko se upiše četveroznamenkasta lozinka, lozinka se sprema na mobilni uređaj na mjesto gdje je bila spremljena stara lozinka (*call promjena_passworda.SaveFile*) te se u nastavku "text" doda blok kojim se definira koji tekst će se spremiti u datoteku (*nova.Text*). Nakon toga, u nastavku "fileName" upiše se ista lokacija i naziv datoteke stare lozinke kako bi se stara tekstualna datoteka zamijenila novom. Ujedno se dobije povratna informacija o uspješnoj promjeni lozinke.

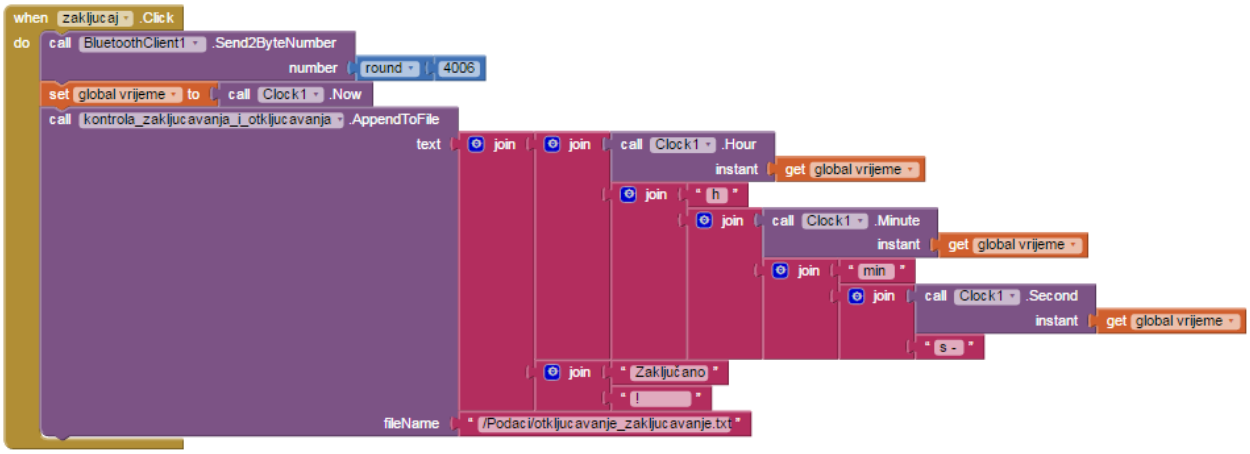


Slika 53. Potvrda nove lozinke u procesu promjene lozinke

Preostale su još tipke "OTKLJUČAJ" i "ZAKLJUČAJ" za upravljanje bravom. Ako korisnik pritisne tipku "OTKLJUČAJ", na *Arduino* se šalje zahtjev za uključivanje zelene LED diode (otključavanje brave), odnosno broj 4005 (*slika 54.*). Ukoliko korisnik pritisne tipku "ZAKLJUČAJ", na *Arduino* se šalje zahtjev za uključivanje crvene LED diode (zaključavanje brave), tj. broj 4006 (*slika 55.*). Zatim slijedi spremanje podataka na mobilni uređaj. Podaci o stanju brave se također spremaju ovisno o vremenu u kojem su tipke pritisnute. Budući da već od ranije postoji globalna varijabla "vrijeme", nije potrebno definirati novu varijablu već se ista može ponovo koristiti. Potrebno je samo u globalnu varijablu spremiti trenutno vrijeme (*set global vrijeme to call Clock1.Now*). Nakon toga slijedi definiranje teksta koji će se spremiti i lokacije na koju će se spremiti, što se definira u nastavku bloka *call kontrola_zakljucavanja_i_otkljucavanja.AppendToFile*. U "text" dijelu iz globalne varijable se također kao i kod prijašnjih slučajeva spremanja uzimaju podaci o tome koliko je sati, minuta i sekundi nakon čega se spajaju zajedno sa stanjem brave. Lokacija na koju će se podaci spremiti određuje se u nastavku "fileName" dijela. Podaci se spremaju na lokaciju /Podaci/otkljucavanje_zakljucavanje.txt, tj. prva kosa crta (/) određuje spremanje na SD karticu, zatim slijedi ime datoteke (Podaci) te naziv tekstualne datoteke u koju će se spremiti stanje brave (otkljucavanje_zakljucavanje).



Slika 54. Definiranje tipke "OTKLJUČAJ"



Slika 55. Definiranje tipke "ZAKLJUČAJ"

Slika 56. Izgled spremljenih podataka o stanju brave

6. Zaključak

U današnje vrijeme je nemoguće zamisliti život bez moderne tehnologije. Naglim razvojem postala je sve zastupljenija i jeftinija. Udaljeno upravljanje elektroničkim uređajima danas više ne mora biti luksuz, već se pomoću *Android* uređaja relativno jeftino može modernizirati vlastiti dom. Temelj ovog rada je *Arduino Uno* platforma koja svojom cijenom, jednostavnom izradom prototipova i otvorenošću koda uvelike pomaže kod jeftine modernizacije vlastitog doma. U ovom radu uređajima se upravlja pomoću *Arduino Uno* platforme koja dobiva naredbe od korisnika preko *Bluetooth modula* i aplikacije dizajnirane u *App Inventoru*. U *App Inventoru* korisnici mogu brzo testirati te popravljati aplikacije, dovoljno je samo jednom pokrenuti testiranje i svaka promjena biti će vidljiva u testnoj aplikaciji na mobilnom uređaju. *App Inventor* je prilagođen svim korisnicima koji nemaju dovoljno znanja oko programiranja u JAVA programskome jeziku tako da se "čisto" programiranje zamjenjuje blokovima naredbi. Nažalost *App Inventor* je u *Beta* fazi testiranja tako da još uvijek ne radi sve kako bi trebalo. Konkretno u ovom slučaju izrade rada veliki problem bila je *Bluetooth* veza prilikom promjene ekrana (*eng. screen).* Prilikom svake promjene ekrana *Bluetooth* bi se veza morala ponovo uspostavljati, a to nije nimalo praktično. Iako *App Inventor* ima broje nedostatke, utjecaj tih nedostataka je moguće smanjiti smislenim korištenjem blokovskog programiranja.

U radu je uspješno prikazana realizacija upravljanja rasvjetom i električnom bravom korištenjem mobilnog uređaja i *Bluetooth* komunikacije. Rasvjeta kojom se upravlja, realizirana je RGB LED trakom i električnom žaruljom. Kontrola RGB LED trakom realizirana je upotrebom RGB Shielda zato što mikrokontroler *Arduino* razvojnog sustava na svojim pinovima ne može dati dovoljnu veliku struju za napajanje LED trake. Također je uspješno realizirana upotreba fotootpornika i temperaturnog senzora. Fotootpornik je iskorišten u dvije svrhe, da na zahtjev korisnika, a na temelju mjerenja osvjetljenja okoline, omogući donošenje odluke, da li bi bilo dobro uključiti žarulju ili je isključiti te da aplikaciji šalje podatke o trenutnoj jakosti osvjetljenja okoline. Korisnik može upravljati bravom koja je zaštićena četveroznamenkastom znakovnom lozinkom uz mogućnost mijenjanja lozinke. Relativno dobra stvar u sustavu je pohranjivanje podataka o stanjima na fotootporniku i temperaturnom senzoru u tekstualnu datoteku na mobilnom uređaju. Također, podaci o otključavanju i zaključavanju spremaju se u tekstualnu datoteku ovisno o vremenu u kojem je određena radnja napravljena. Najvažnija je realizacija mogućnosti udaljene kontrole *Arduino* sustava, tj. njegovih izlaza (kojima se dalje može upravljati raznim uređajima) i mogućnost dobivanja podataka sa njega samog. Realizacija sklopovskog dijela rada nije toliko zahtjevna kao izrada same aplikacije te uspostavljanje

uspješne komunikacije između mobilnog uređaja i *Arduino* sustava. Pravilnim spajanjem svih navedenih komponenata te korištenjem potrebne programske logike realizirana je početna ideja ovog rada.

7. Literatura

Slike

[1] https://www.arduino.cc/en/uploads/Main/ArduinoUno_R3_Front_450px.jpg (Dostupno 21.6.2016)

[2] http://www.velleman.co.uk/contents/media/ka01b.jpg (Dostupno 21.6.2016)

[3] http://arduinolearning.com/wp-content/uploads/2016/05/hc-06.jpg (Dostupno 21.6.2016)

[4] http://storage.googleapis.com/wzukusers/user-17563472/images/56a7e041983158qhMBqv/30led-Strip.jpg (Dostupno 28.6.2016)

[5] https://tushev.org/images/electronics/arduino/ds18x20/DS18B20.jpg (Dostupno 29.6.2016)

[6] http://www.electroschematics.com/wp-content/uploads/2014/11/arduino-uno-r3-schematic.png (Dostupno 30.6.2016)

INTERNETSKE JEDINICE

[7] https://www.arduino.cc/en/Main/ArduinoBoardUno (Dostupno 21.6.2016)

[8] http://www.apogeekits.com/PDF_Files/manual-ka01.pdf (Dostupno 21.6.2016.)

[9] http://www.velleman.co.uk/contents/en-uk/p578_ka01.html (Dostupno 21.6.2016.)

[10] https://developer.android.com/guide/topics/connectivity/bluetooth.html (Dostupno 23.6.2016.)

[11] https://hr.wikipedia.org/wiki/Android_(operacijski_sustav) (Dostupno 28.6.2016.)

[12] http://www.edukacentar.hr/Blog/Uvod-u-razvoj-android-aplikacija (Dostupno 28.6.2016.)

[13] http://www.flexfireleds.com/pages/Comparison-between-3528-LEDs-and-5050-LEDs.html (Dostupno 28.6.2016.)

[14] http://www.ledshop.hr/blog/11_Kako-pravilno-spajati-LED-traku-.html (Dostupno 28.6.2016.)

[15] https://hr.wikipedia.org/wiki/Fotootpornik (Dostupno 29.6.2016.)

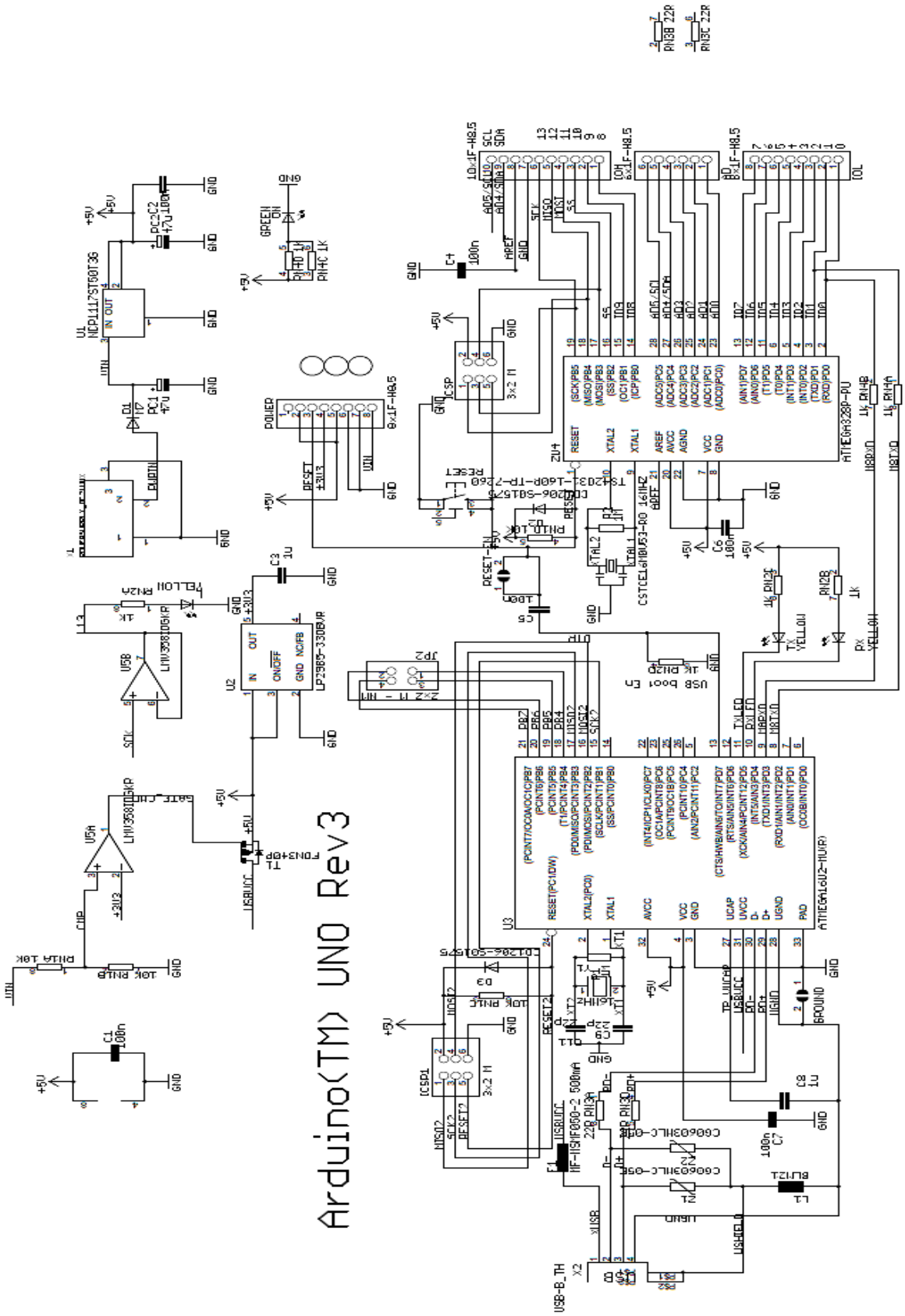
[16] http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf (Dostupno 29.6.2016.)

[17] http://appinventor.mit.edu/explore/about-us.html (Dostupno 30.6.2016.)

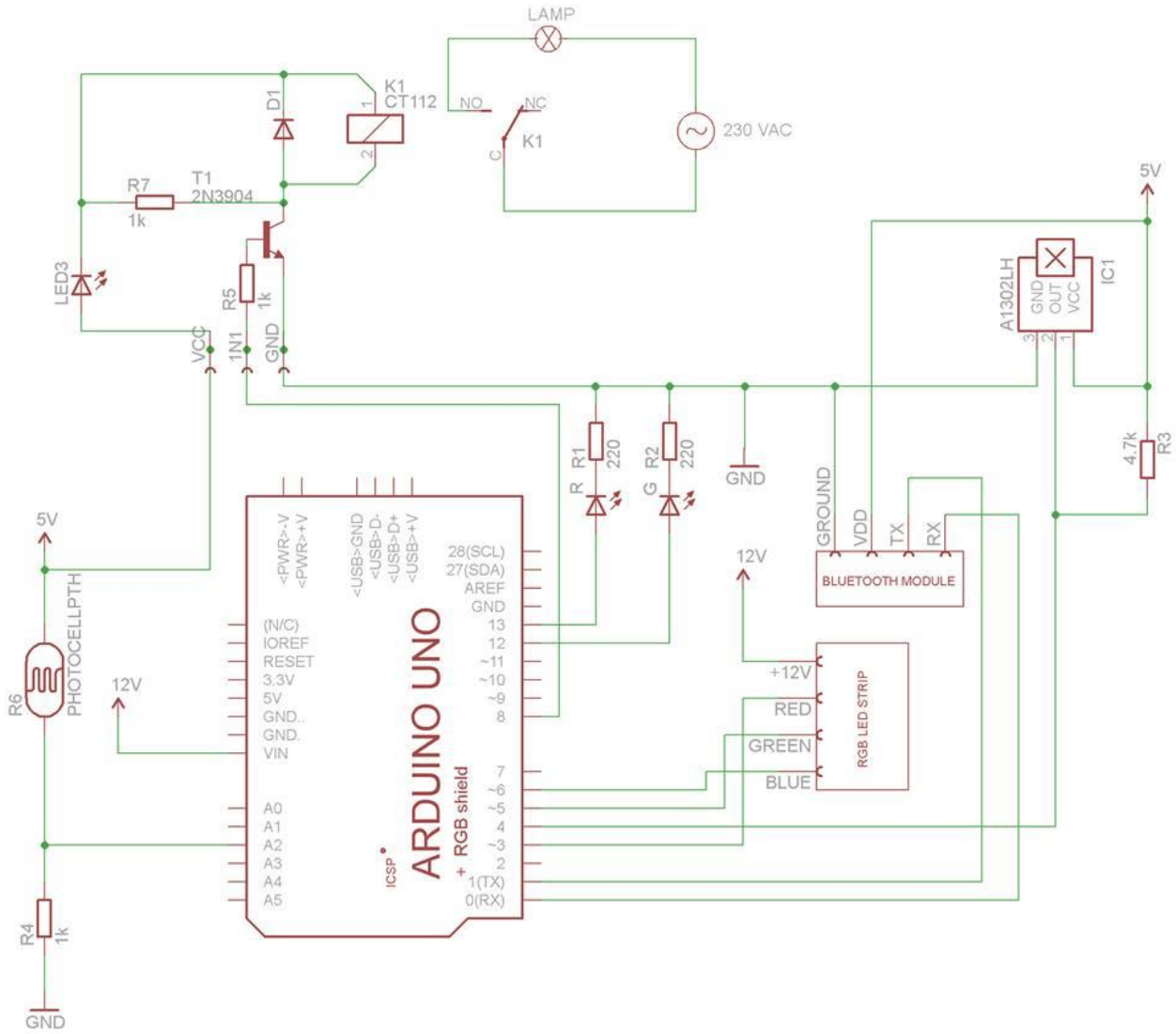
[18] https://en.wikipedia.org/wiki/App_Inventor_for_Android (Dostupno 30.6.2016.)

8. Prilozi

PRILOG 1 : ELEKTRONIČKA SHEMA ARDUINO UNO (REV 3) [6]



PRILOG 2 : SHEMA POVEZIVANJA MIKROKONTROLERA S ELEKTRONIČKIM ELEMENTIMA



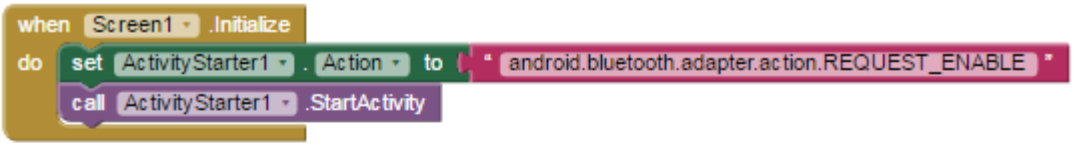
PRILOG 3 : PROGRAMSKI KOD ARDUINO

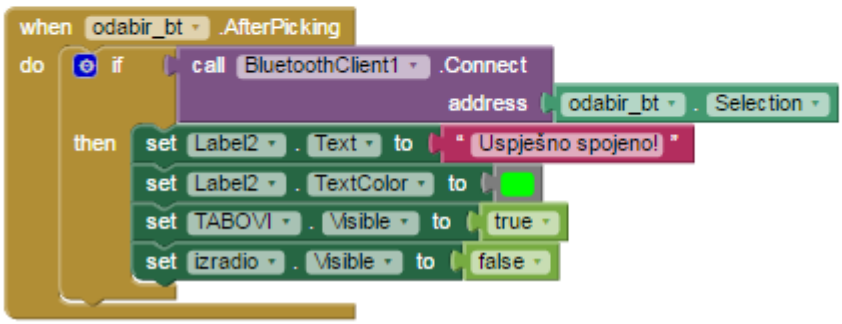
```
// Završni rad //
// Danijel Korunić //

#include <OneWire.h>
#include <DallasTemperature.h>
#include <HardwareSerialRS485.h>
#include <MessageReader.h>
#define temPin 4
OneWire ourWire(temPin);
DallasTemperature sensors(&ourWire);

int bluetoothTx = 0;
int bluetoothRx = 1;
int foto=2;
int svjetlost;

void setup()
{
 pinMode(3,OUTPUT); // Crvena LED
 pinMode(5,OUTPUT); // Zelena LED
 pinMode(6,OUTPUT); // Plava LED
 pinMode(12,OUTPUT); // Brava ON
 pinMode(13,OUTPUT); // Brava OFF
 pinMode(8,OUTPUT); // Žarulja - relej

 digitalWrite(3,LOW);
 digitalWrite(5,LOW);
 digitalWrite(6,LOW);
 digitalWrite(12,HIGH); // otkljucano po defaultu
 digitalWrite(13,LOW);
 digitalWrite(8,LOW);

  // temp senzor počinje čitati
  sensors.begin();
```

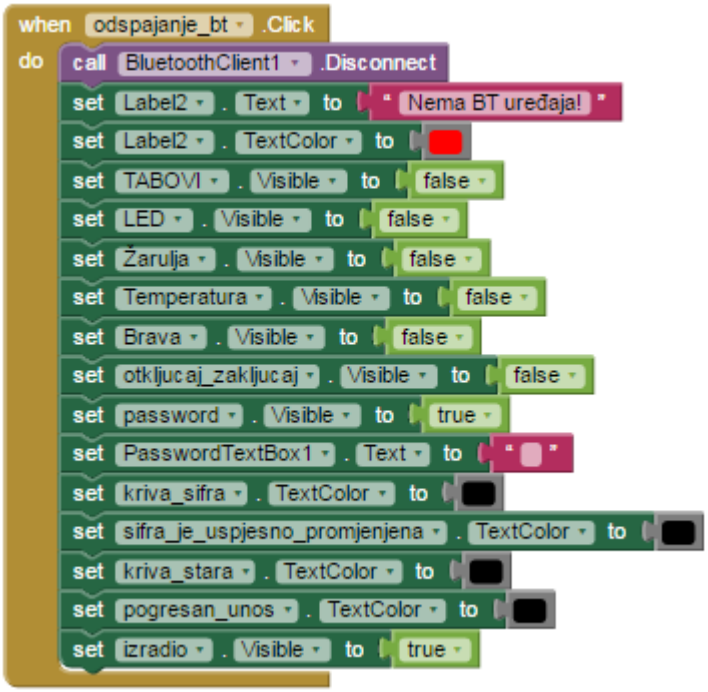
```
  //početak serijske komunikacije
  Serial.begin(9600);
}

void loop()
{
      // FOTOOTPORNIK
      svjetlost = analogRead(foto);
      delay(10);

  //Čitanje informacija s bluetooth modula
  if(Serial.available()>= 2 )
      {
    int BTnaredba1 = Serial.read();
    int BTnaredba2 = Serial.read();
    int BTnaredba = (BTnaredba2 *256) + BTnaredba1;

    // LEDICE
    if (BTnaredba >= 1000 && BTnaredba <1255)     {
    int plava = BTnaredba;
    plava = map(plava, 1000,1255,0,255);
    analogWrite(6,plava);
    delay(10);
                                                  }

    if (BTnaredba >=2000 && BTnaredba <2255)      {
      int zelena = BTnaredba;
      zelena = map(zelena,2000,2255,0,255);
      analogWrite(5,zelena);
      delay(10);
                                                  }

     if (BTnaredba >=3000 && BTnaredba < 3255)     {
      int crvena = BTnaredba;
      crvena = map(crvena, 3000, 3255,0,255);
      analogWrite(3,crvena);
      delay(10);
                                                  }
```

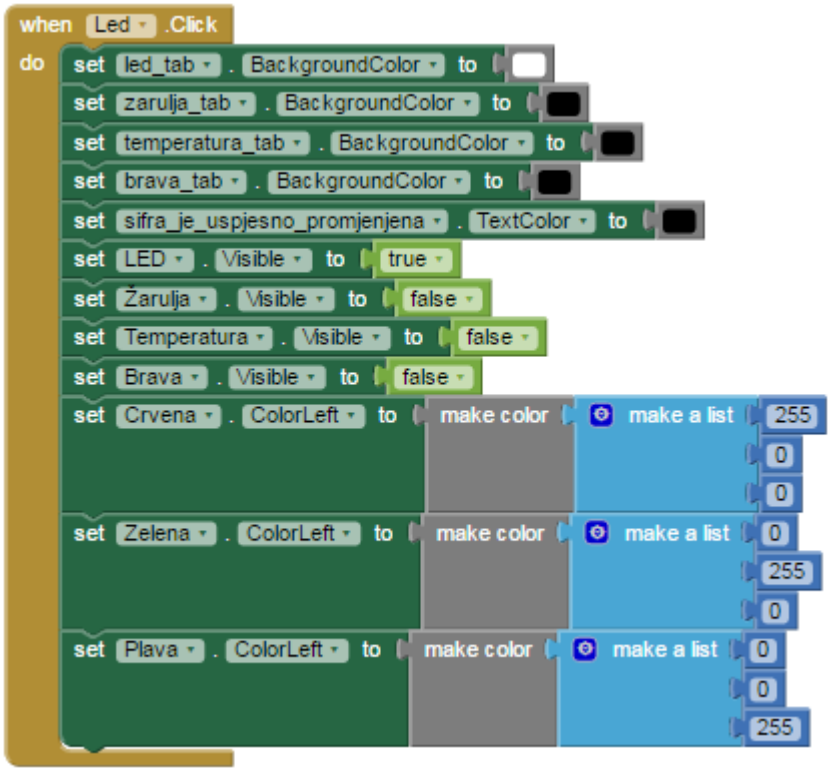
```
      // FOTOOTPORNIK (optimalno svjetlo)
      if (BTnaredba == 4003 && svjetlost < 200 && svjetlost > 0 )
      {digitalWrite(8,HIGH);}

      if (BTnaredba == 4003 && svjetlost > 200)
      {digitalWrite(8,LOW);}

       if (BTnaredba == 4007)
      {Serial.println(svjetlost);}

      // ŽARULJA
      if (BTnaredba==4002)       {
        svjetlost=5100;
        digitalWrite(8,HIGH);    }

      if (BTnaredba==4001)       {
        svjetlost=5300;
        digitalWrite(8,LOW);     }

      // TEMPERATURA
      if (BTnaredba == 4004)     {
      sensors.requestTemperatures();
      Serial.println(sensors.getTempCByIndex(0));
                                 }

      // BRAVA
      if (BTnaredba == 4005)     {
      digitalWrite(12,HIGH);
      digitalWrite(13,LOW);
                                 }

      if (BTnaredba == 4006)     {
      digitalWrite(12,LOW);
      digitalWrite(13,HIGH);
                                 }
     }
}
```

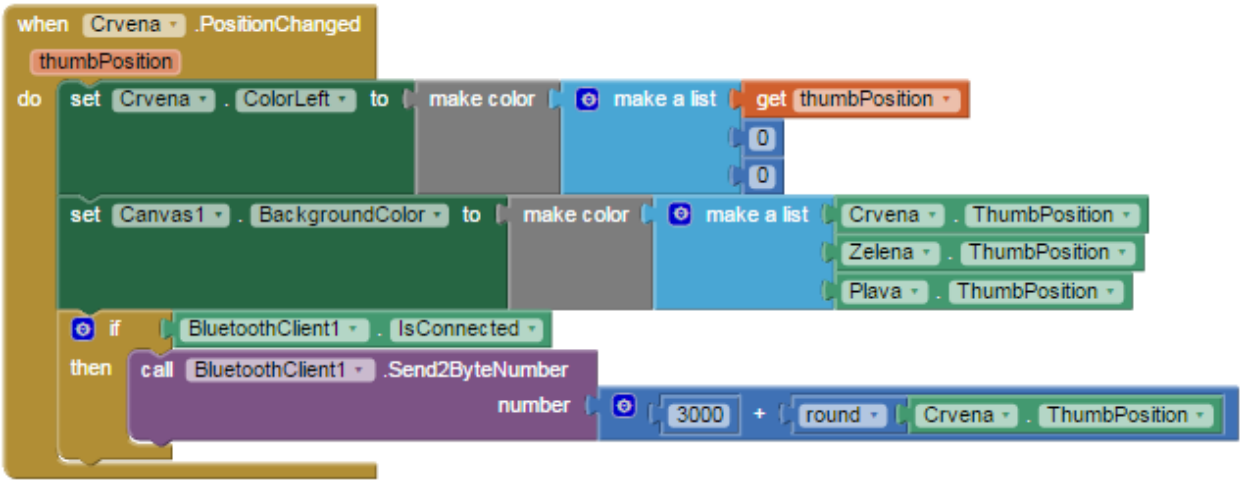
PRILOG 4 : PROGRAMSKI KOD - APP INVENTOR

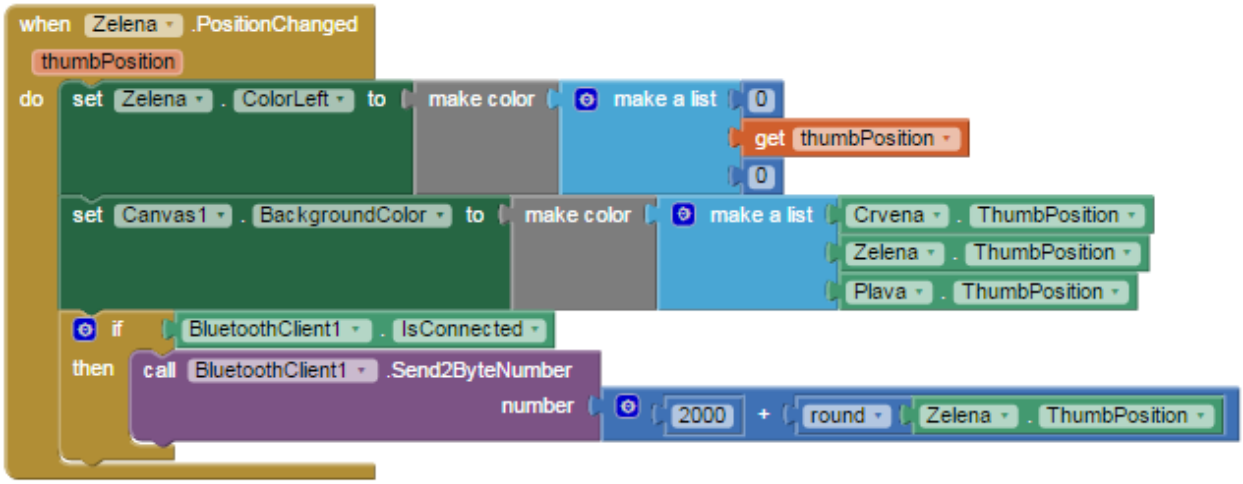


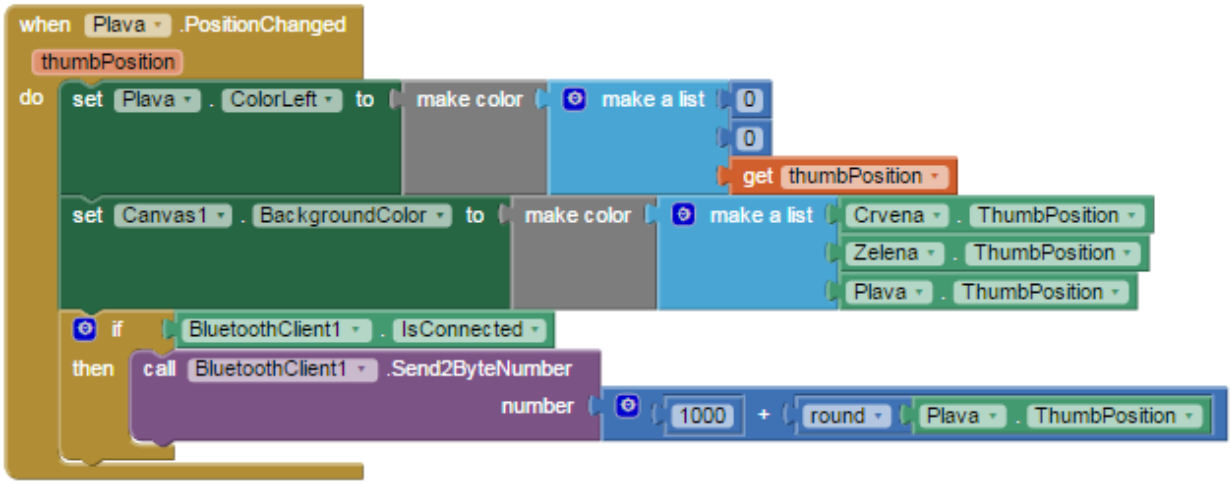


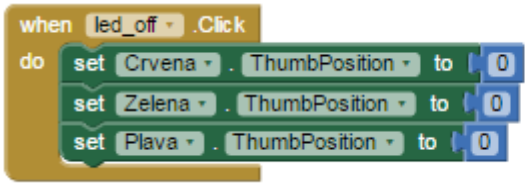


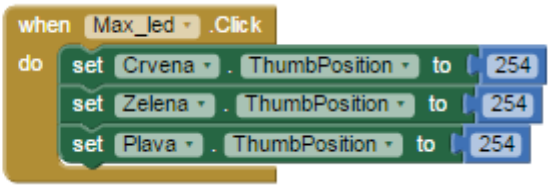

when Led .Click
do
set led_tab . BackgroundColor to
set zarulja_tab . BackgroundColor to
set temperatura_tab . BackgroundColor to
set brava_tab . BackgroundColor to
set sifra_je_uspjesno_promjenjena . TextColor to
set LED . Visible to true
set Žarulja . Visible to false
set Temperatura . Visible to false
set Brava . Visible to false
set Crvena . ColorLeft to make color make a list 255 0 0
set Zelena . ColorLeft to make color make a list 0 255 0
set Plava . ColorLeft to make color make a list 0 0 255

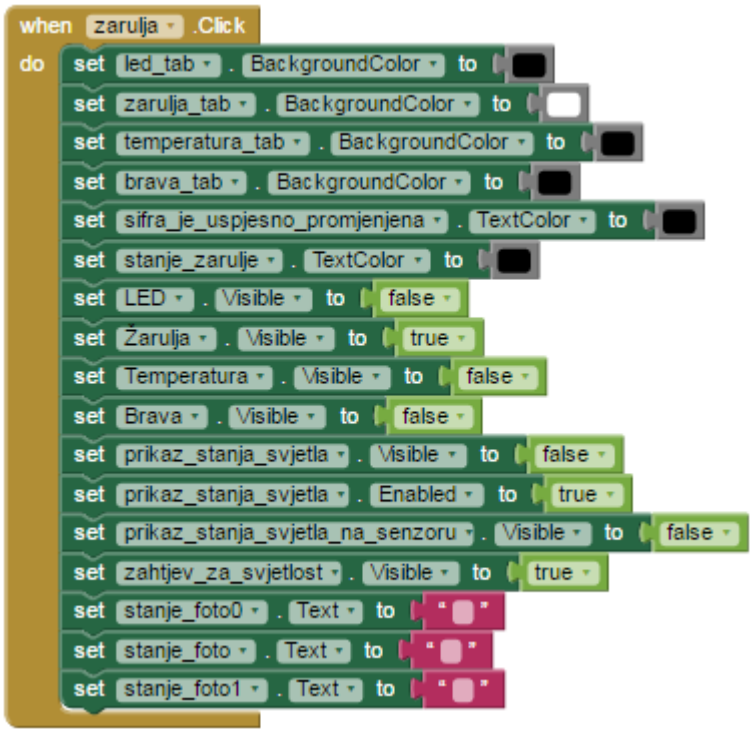

when Crvena .PositionChanged
thumbPosition
do
set Crvena . ColorLeft to make color make a list get thumbPosition 0 0
set Canvas1 . BackgroundColor to make color make a list Crvena . ThumbPosition Zelena . ThumbPosition Plava . ThumbPosition
if BluetoothClient1 . IsConnected
then call BluetoothClient1 .Send2ByteNumber
number 3000 + round Crvena . ThumbPosition

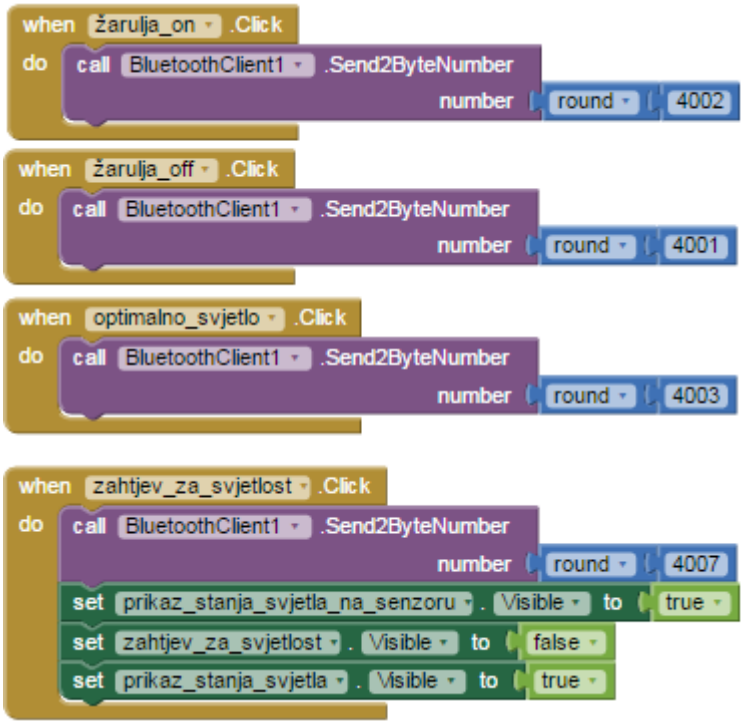

when Zelena .PositionChanged
thumbPosition
do set Zelena . ColorLeft to make color make a list 0
get thumbPosition
0
set Canvas1 . BackgroundColor to make color make a list Crvena . ThumbPosition
Zelena . ThumbPosition
Plava . ThumbPosition
if BluetoothClient1 . IsConnected
then call BluetoothClient1 .Send2ByteNumber
number 2000 + round Zelena . ThumbPosition

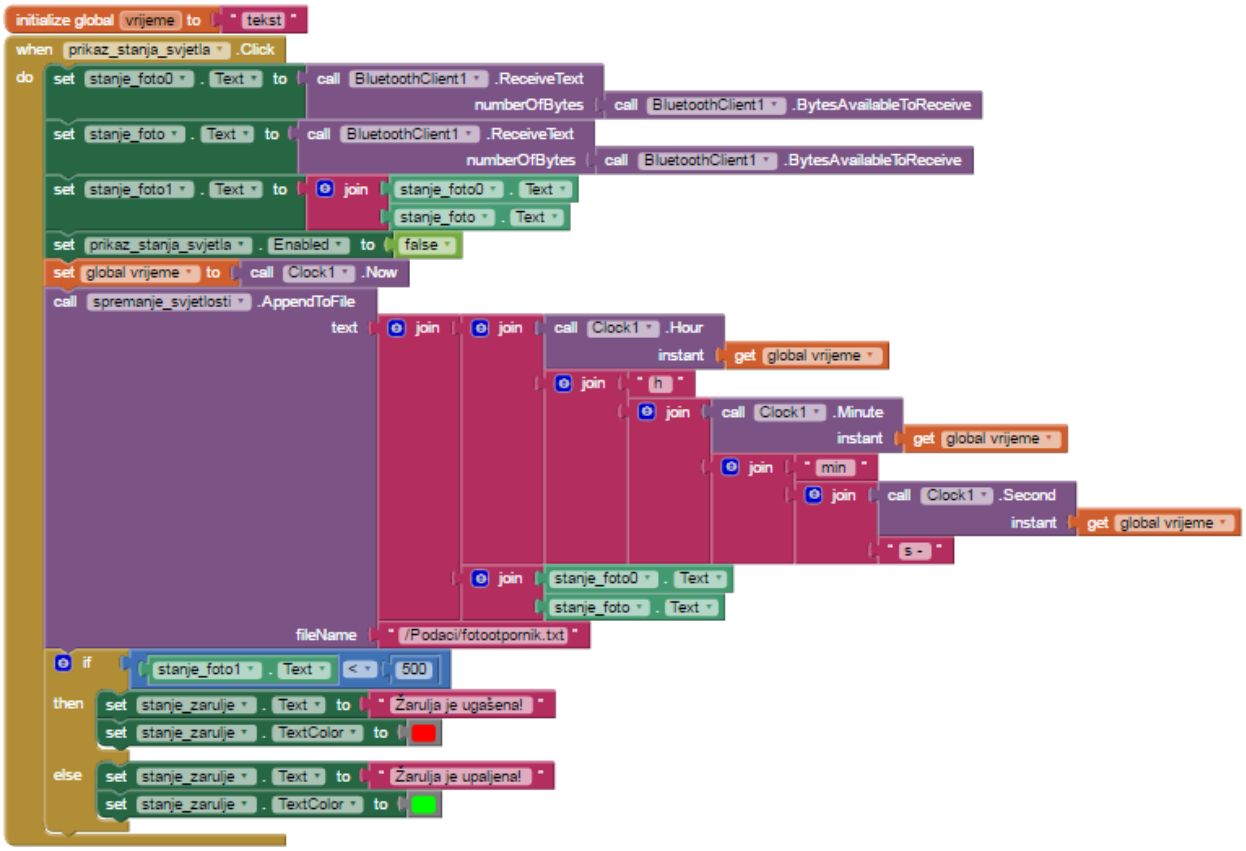

when Plava .PositionChanged
thumbPosition
do set Plava . ColorLeft to make color make a list 0
0
get thumbPosition
set Canvas1 . BackgroundColor to make color make a list Crvena . ThumbPosition
Zelena . ThumbPosition
Plava . ThumbPosition
if BluetoothClient1 . IsConnected
then call BluetoothClient1 .Send2ByteNumber
number 1000 + round Plava . ThumbPosition

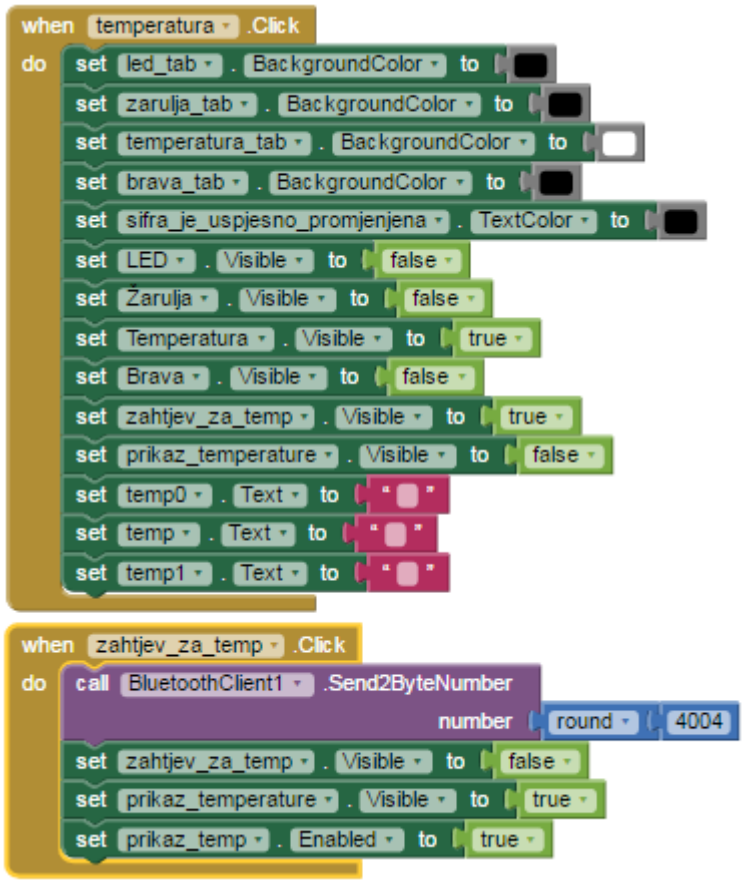

when led_off .Click
do set Crvena . ThumbPosition to 0
set Zelena . ThumbPosition to 0
set Plava . ThumbPosition to 0

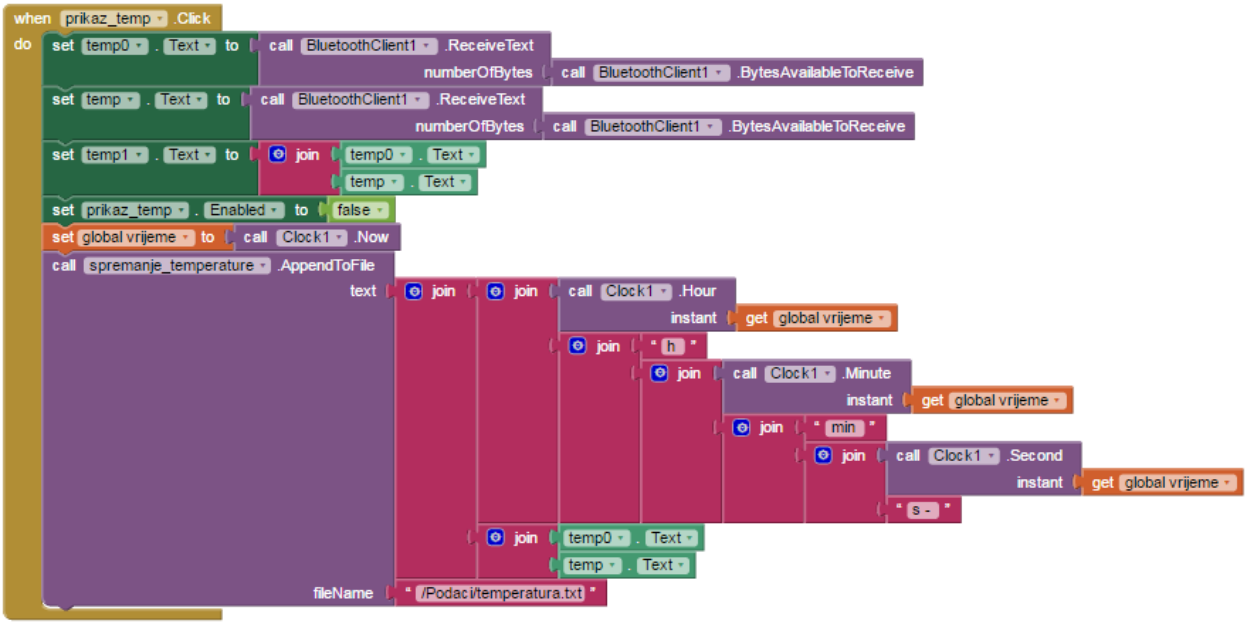

when Max_led .Click
do set Crvena . ThumbPosition to 254
set Zelena . ThumbPosition to 254
set Plava . ThumbPosition to 254

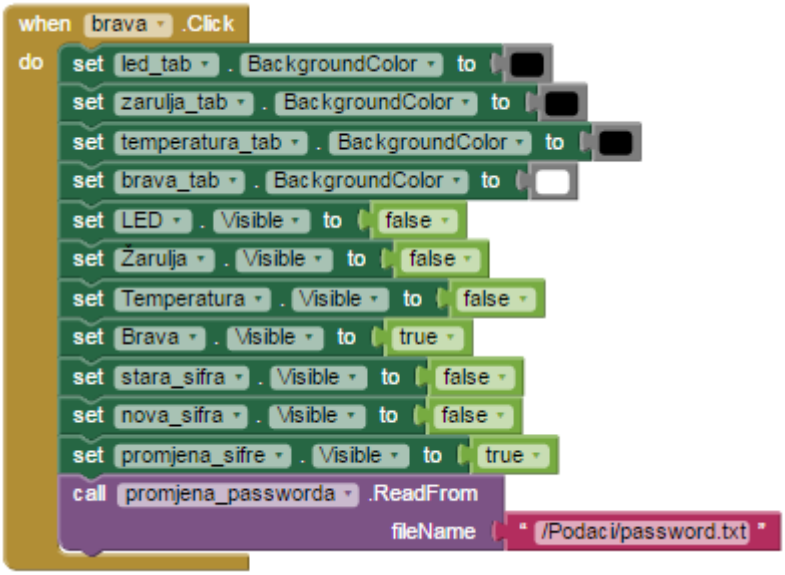

when zarulja .Click
do set led_tab . BackgroundColor to
set zarulja_tab . BackgroundColor to
set temperatura_tab . BackgroundColor to
set brava_tab . BackgroundColor to
set sifra_je_uspjesno_promjenjena . TextColor to
set stanje_zarulje . TextColor to
set LED . Visible to false
set Žarulja . Visible to true
set Temperatura . Visible to false
set Brava . Visible to false
set prikaz_stanja_svjetla . Visible to false
set prikaz_stanja_svjetla . Enabled to true
set prikaz_stanja_svjetla_na_senzoru . Visible to false
set zahtjev_za_svjetlost . Visible to true
set stanje_foto0 . Text to
set stanje_foto . Text to
set stanje_foto1 . Text to

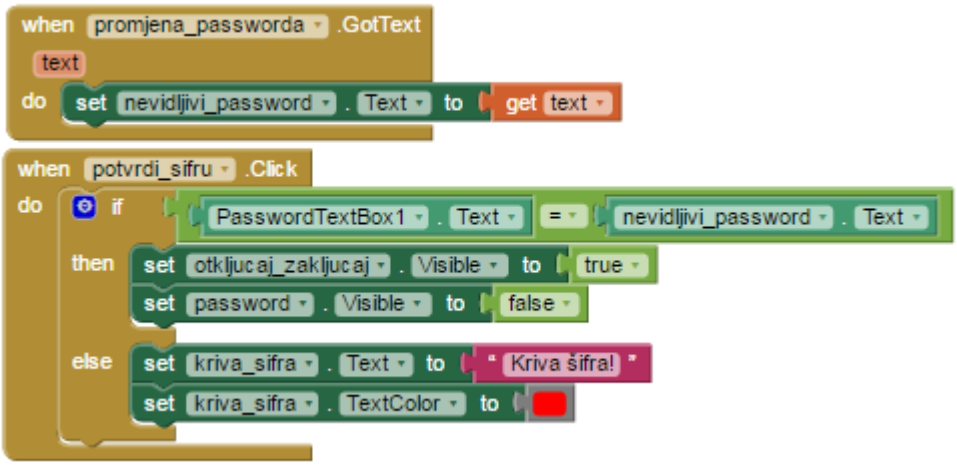

when žarulja_on .Click
do call BluetoothClient1 .Send2ByteNumber
number round 4002
when žarulja_off .Click
do call BluetoothClient1 .Send2ByteNumber
number round 4001
when optimalno_svjetlo .Click
do call BluetoothClient1 .Send2ByteNumber
number round 4003
when zahtjev_za_svjetlost .Click
do call BluetoothClient1 .Send2ByteNumber
number round 4007
set prikaz_stanja_svjetla_na_senzoru . Visible to true
set zahtjev_za_svjetlost . Visible to false
set prikaz_stanja_svjetla . Visible to true

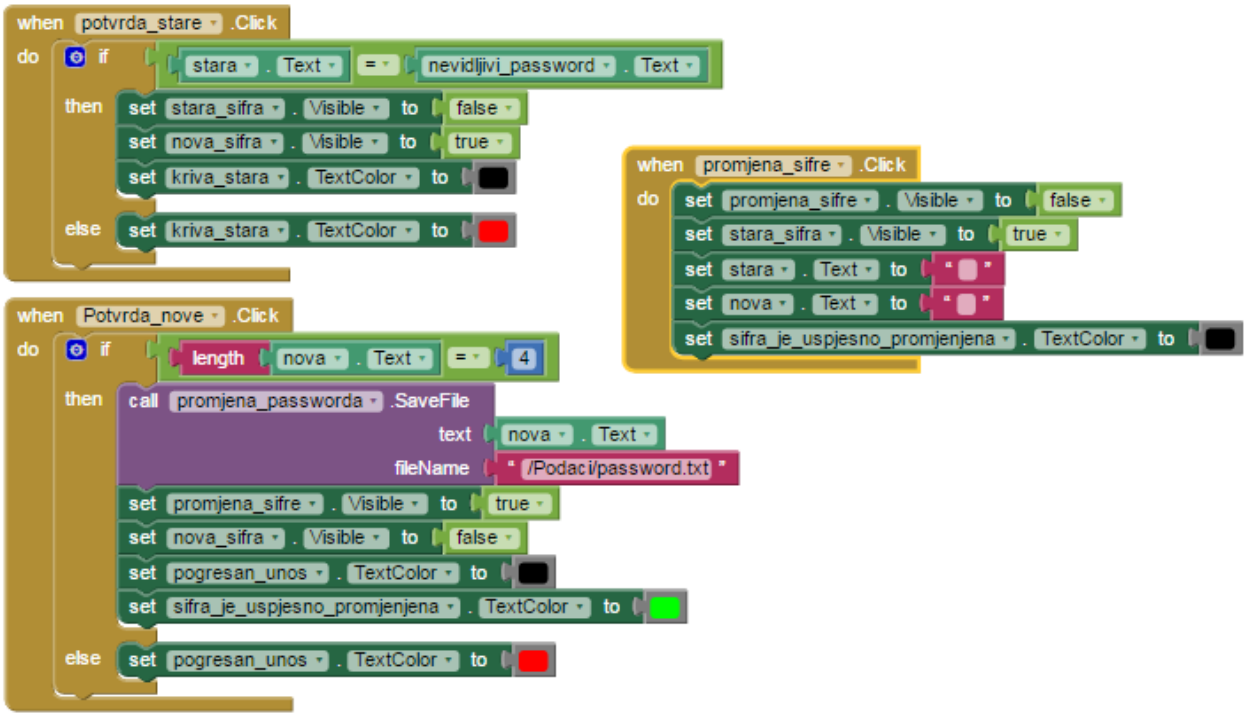

initialize global vrijeme to " tekst "
when prikaz_stanja_svjetla .Click
do set stanje_foto0 . Text to call BluetoothClient1 .ReceiveText
numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
set stanje_foto . Text to call BluetoothClient1 .ReceiveText
numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
set stanje_foto1 . Text to join stanje_foto0 . Text
stanje_foto . Text
set prikaz_stanja_svjetla . Enabled to false
set global vrijeme to call Clock1 .Now
call spremanje_svjetlosti .AppendToFile
text join join call Clock1 .Hour
instant get global vrijeme
join " h "
join call Clock1 .Minute
instant get global vrijeme
join " min "
join call Clock1 .Second
instant get global vrijeme
" s - "
join stanje_foto0 . Text
stanje_foto . Text
fileName " /Podaci/fotootpornik.txt "
if stanje_foto1 . Text < 500
then set stanje_zarulje . Text to " Žarulja je ugašena! "
set stanje_zarulje . TextColor to
else set stanje_zarulje . Text to " Žarulja je upaljena! "
set stanje_zarulje . TextColor to

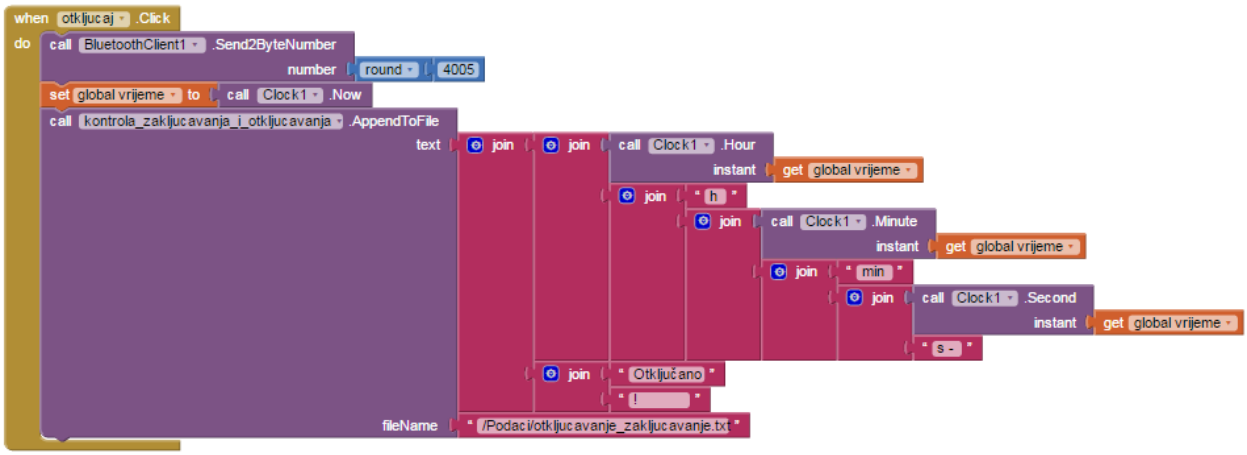

when temperatura .Click
do set led_tab . BackgroundColor to
set zarulja_tab . BackgroundColor to
set temperatura_tab . BackgroundColor to
set brava_tab . BackgroundColor to
set sifra_je_uspjesno_promjenjena . TextColor to
set LED . Visible to false
set Žarulja . Visible to false
set Temperatura . Visible to true
set Brava . Visible to false
set zahtjev_za_temp . Visible to true
set prikaz_temperature . Visible to false
set temp0 . Text to " "
set temp . Text to " "
set temp1 . Text to " "
when zahtjev_za_temp .Click
do call BluetoothClient1 .Send2ByteNumber
number round 4004
set zahtjev_za_temp . Visible to false
set prikaz_temperature . Visible to true
set prikaz_temp . Enabled to true

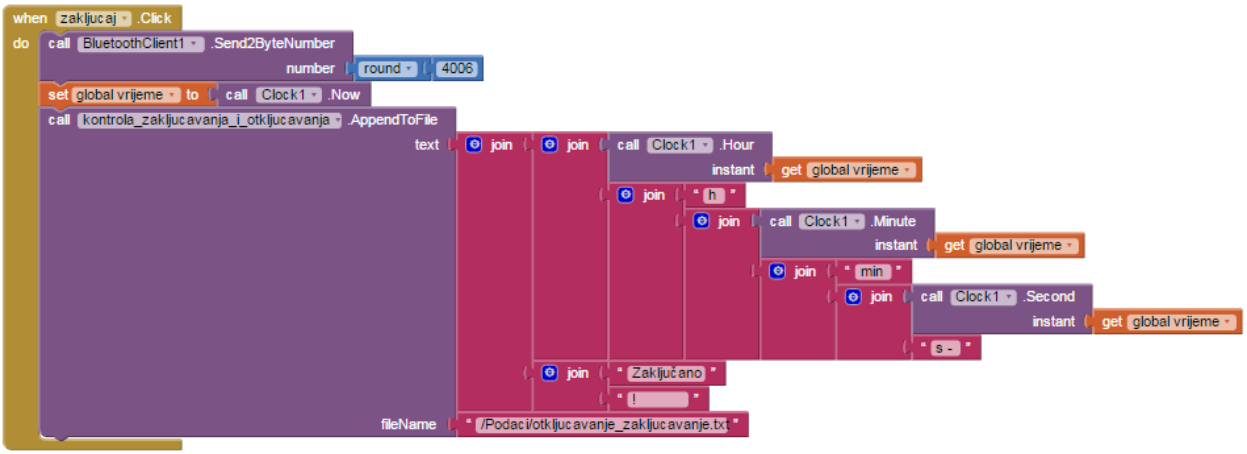

when prikaz_temp .Click
do set temp0 . Text to call BluetoothClient1 .ReceiveText
numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
set temp . Text to call BluetoothClient1 .ReceiveText
numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
set temp1 . Text to join temp0 . Text
temp . Text
set prikaz_temp . Enabled to false
set global vrijeme to call Clock1 .Now
call spremanje_temperature .AppendToFile
text join join call Clock1 .Hour
instant get global vrijeme
join " h "
join call Clock1 .Minute
instant get global vrijeme
join " min "
join call Clock1 .Second
instant get global vrijeme
" s - "
join temp0 . Text
temp . Text
fileName " /Podaci/temperatura.txt "


when brava .Click
do set led_tab . BackgroundColor to
set zarulja_tab . BackgroundColor to
set temperatura_tab . BackgroundColor to
set brava_tab . BackgroundColor to
set LED . Visible to false
set Žarulja . Visible to false
set Temperatura . Visible to false
set Brava . Visible to true
set stara_sifra . Visible to false
set nova_sifra . Visible to false
set promjena_sifre . Visible to true
call promjena_passworda .ReadFrom
fileName " /Podaci/password.txt "


when promjena_passworda .GotText
text
do set nevidljivi_password . Text to get text
when potvrdi_sifru .Click
do if PasswordTextBox1 . Text = nevidljivi_password . Text
then set otkljucaj_zakljucaj . Visible to true
set password . Visible to false
else set kriva_sifra . Text to " Kriva šifra! "
set kriva_sifra . TextColor to


when potvrda_stare .Click
do if stara . Text = nevidljivi_password . Text
then set stara_sifra . Visible to false
set nova_sifra . Visible to true
set kriva_stara . TextColor to
else set kriva_stara . TextColor to
when promjena_sifre .Click
do set promjena_sifre . Visible to false
set stara_sifra . Visible to true
set stara . Text to
set nova . Text to
set sifra_je_uspjesno_promjenjena . TextColor to
when Potvrda_nove .Click
do if length nova . Text = 4
then call promjena_passworda .SaveFile
text nova . Text
fileName /Podaci/password.txt
set promjena_sifre . Visible to true
set nova_sifra . Visible to false
set pogresan_unos . TextColor to
set sifra_je_uspjesno_promjenjena . TextColor to
else set pogresan_unos . TextColor to


when otkljucaj .Click
do call BluetoothClient1 .Send2ByteNumber
number round 4005
set global vrijeme to call Clock1 .Now
call kontrola_zakljucavanja_i_otkljucavanja .AppendToFile
text join join call Clock1 .Hour
instant get global vrijeme
join h
join call Clock1 .Minute
instant get global vrijeme
join min
join call Clock1 .Second
instant get global vrijeme
s -
join Otključano
!
fileName /Podaci/otkljucavanje_zakljucavanje.txt


when zakljucaj .Click
do call BluetoothClient1 .Send2ByteNumber
number round 4006
set global vrijeme to call Clock1 .Now
call kontrola_zakljucavanja_i_otkljucavanja .AppendToFile
text join join call Clock1 .Hour
instant get global vrijeme
join h
join call Clock1 .Minute
instant get global vrijeme
join min
join call Clock1 .Second
instant get global vrijeme
s -
join Zaključano
!
fileName /Podaci/otkljucavanje_zakljucavanje.txt

UNIVERSITY NORTH

Sveučilište Sjever

VŽ KC

MMI

SVEUČILIŠTE SJEVER

IZJAVA O AUTORSTVU

I
SUGLASNOST ZA JAVNU OBJAVU

Završni/diplomski rad isključivo je autorsko djelo studenta koji je isti izradio te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, izvora s interneta, i drugih izvora) bez navođenja izvora i autora navedenih radova. Svi dijelovi tuđih radova moraju biti pravilno navedeni i citirani. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom, odnosno nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, Danijel Korunić pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor završnog rada pod naslovom Upravljanje rasvjetom i električnom bravom korištenjem mobilnih uređaja i Bluetooth komunikacije te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Danijel Korunić

Danijel Korunić

Sukladno Zakonu o znanstvenoj djelatnost i visokom obrazovanju završne/diplomske radove sveučilišta su dužna trajno objaviti na javnoj internetskoj bazi sveučilišne knjižnice u sastavu sveučilišta te kopirati u javnu internetsku bazu završnih/diplomskih radova Nacionalne i sveučilišne knjižnice. Završni radovi istovrsnih umjetničkih studija koji se realiziraju kroz umjetnička ostvarenja objavljuju se na odgovarajući način.

Ja, Danijel Korunić neopozivo izjavljujem da sam suglasan s javnom objavom završnog rada pod naslovom Upravljanje rasvjetom i električnom bravom korištenjem mobilnih uređaja i Bluetooth komunikacije čiji sam autor.

Danijel Korunić

Danijel Korunić