Izrada modularnog alarmnog sustava

Čavlek, David

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

David Čavlek, 0313026276

Izrada modularnog alarmnog sustava

Završni rad

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

David Čavlek, 0313026276

Izrada modularnog alarmnog sustava

modular alarm system

Završni rad

Mentor:

mr. sc. Željko Knok

Prijava obrane završnog rada

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

PRIJAVA TEME I OBRANE ZAVRŠNOG/DIPLOMSKOG RADA

Stručni prijediplomski studij:

☑ Računarstvo ☐ Održivi razvoj ☐ Menadžment turizma i sporta

Stručni diplomski studij Menadžment turizma i sporta: ☐

Pristupnik: David Čavlek, JMBAG: 0313026276
(ime i prezime)

Kolegij: Baze podataka I
(na kojem se piše rad)

Mentor: mr.sc. Željko Knok
(ime i prezime, zvanje)

Naslov rada: Izrada modularnog alarmnog sustava

Naslov rada na engleskom jeziku: Creation of a modular alarm system

Članovi povjerenstva:
1. mr.sc. Ivan Hegedući, predsjednik
(ime i prezime, zvanje)
2. Nenad Breslauer, v.pred., član
(ime i prezime, zvanje)
3. mr.sc. Željko Knok, mentor
(ime i prezime, zvanje)
4. Marija Miličančuk,v.pred., zamjenski član
(ime i prezime, zvanje)

Broj zadatka: 2023-RAČ-4

Kratki opis zadatka:

Potrebno je izgraditi maketu alarmnog sustava koji koristi Python softver na Raspberry Pi u kombinaciji s nekim elementarnim elektroničkim krugovima. Svi alarmni sustavi imaju dvije osnovne funkcije. Prvo, prate svoju okolinu tražeći promjenu u njoj, poput otvaranja vrata ili prozora ili premještanja osobe po sobi. Drugo, upozoravaju vlasnika ili korisnika na ovu promjenu. Sustav mora koristiti softver za skeniranje za otkrivanje uljeza. Ponašati se kao pas čuvar, hodajući gore-dolje po ogradi u potrazi za uljezom ili alarmom sobom. Mora imati tipku za alarm, koje se može deaktivirati i uključiti. Alati: Za rad koristiti Raspberry Pi razvojno okruženje, Python i odgovarajuće senzore. Literatura: William Pretty, Software, Building a High-Tech Alarm System with Raspberry Pi

Datum: 1.7.2024. Potpis mentora: Željko Knok

Predgovor

Izrada ovog završnog rada bila je zahtjevan, ali veoma poučan proces, koji mi je omogućio produbljivanje znanja o sigurnosnim sistemima i Internet of Things (IoT) tehnologija. Tokom rada susreo sam se s različitim izazovima, ali zahvaljujući podršci kolega, uspješno sam ih prevladao.

Zahvaljujem se svim osobama, ustanovama i tvrtkama koje su mi pružile podršku i pomoć tijekom izrade ovog rada. Bez njihove potpore, ovaj projekt ne bi bio moguć.

Veliku zahvalu dugujem i obrtu „Interijeri Stančin“, koji je izradio maketu kućice na kojoj se nalazi završni rad. Njihov doprinos bio je od neprocjenjive vrijednosti za vizualizaciju i demonstraciju rada sigurnosnog sistema.

Posebnu zahvalnost dugujem svojemu mentoru Željku Knoku na pruženim savjetima, sugestijama i tehničkoj pomoći pri završetku ovog projekta.

Nadam se da će ovaj rad poslužiti kao inspiracija i koristan izvor informacija budućim generacijama studenata koji se žele baviti sigurnosnim tehnologijama i IoT-om. Rad na ovom projektu proširio je moje horizonte i omogućio mi da steknem dragocjena iskustva koja će mi koristiti u budućoj karijeri.

Sažetak

Ovaj završni rad opisuje izradu sigurnosnog sistema baziranog na Raspberry Pi ploči, uz korištenje PIR senzora pokreta, zvučnika i LED dioda. Sistem detektira pokret, aktivira alarm i šalje e-mail obavijesti korisniku. Povezan je s Blynk IoT aplikacijom koja omogućuje daljinsko upravljanje i nadzor. Preko aplikacije korisnik može paliti i gasiti lampicu i uključivati i isključivati alarm. Projekt uključuje detaljne korake instalacije, povezivanja komponenti, konfiguracije e-mail usluge i integracije s Blynk-om. Kroz testiranje i optimizaciju, postignut je stabilan i pouzdan sigurnosni sistem pogodan za kuće, urede i druge objekte.

Ključne riječi: sigurnosni sistem, Raspberry Pi, PIR senzor, Blynk IoT, e-mail obavijesti, daljinsko upravljanje, detekcija pokreta.

Abstract

This thesis describes the development of a security system based on a Raspberry Pi board, utilizing a PIR motion sensor, speaker, and LED light. The system detects motion, activates an alarm, and sends email notifications to the user. It is connected to the Blynk IoT application, allowing for remote control and monitoring. Through the application, the user can turn the light on and off and enable or disable the alarm. The project includes detailed steps for installation, component connection, email service configuration, and integration with Blynk. Through testing and optimization, a stable and reliable security system suitable for homes, offices, and other premises has been achieved.

Keywords: security system, Raspberry Pi, PIR sensor, Blynk IoT, email notifications, remote control, motion detection.

Sadržaj

1. UVOD 1
2. Cilj rada 2
3. Komponente i sredstva 3
3.1 Raspberry Pi 3
3.2 PIR senzor 5
3.3 Zvučnik (Buzzer) 6
3.4 LED dioda 7
3.5 Breadboard (eksperimentalna pločica) 8
3.6 Spojni kablovi 9
3.7 Otpornici od 220 Ohm-a 10
3.8 Opis makete drvene kućice za alarmni sustav 10
4. Instalacija i podešavanje Raspberry Pi 11
4.1 Preuzimanje i instalacija OS-a 11
4.2 Povezivanje komponenti na Raspberry Pi 12
4.3 Provjera GPIO spojeva: 14
5. Konfiguracija e-mail usluge za Raspberry Pi 15
5.1 Kreiranje e-mail adrese 15
5.2 Kreiranje aplikacijske lozinke 16
5.3 Instalacija i konfiguracija SMTP servera na Raspberry Pi 17
6. Povezivanje s Blynk IoT aplikacijom 18
6.1 Registracija na Blynk 19
6.2 Kreiranje novog projekta 19
6.3 Pisanje Python skripte za povezivanje s Blynk-om 20
7. Automatsko pokretanje aplikacije na Raspberry Pi 21
7.1 Koraci za automatsko pokretanje: 22
7.2 Aktivacija alarma: 24
7.3 Detekcija pokreta i alarm: 25
7.4 Kod za BlynkLib.py: 26
7.5 Kod za BlynkTimer.py: 34
8. Zaključak 37
Izjava o autorstvu 38
9. Popis slika 39
10. Popis literature 40
11. Programski kod 41

Popis korištenih kratica

PIR Passive Infrared

LED Light Emitting Diode

IoT Internet of Things

ARM Advanced RISC Machine

NOOBS New Out Of Box Software

USB Universal Serial Bus

HDMI High-Definition Multimedia Interface

OS Operating System

CSI Camera Serial Interface

DSI Display Serial Interface

VCC Voltage Common Collector

GND Ground

OUT Output

1. UVOD

Sigurnosni sistemi postali su neizostavan dio svakodnevnog života, nudeći zaštitu i sigurnost u različitim okruženjima kao što su domovi, uredi i poslovni prostori. Razvoj tehnologije omogućio je stvaranje naprednih, ali pristupačnih rješenja koja su lako dostupna populaciji. Jedno od takvih rješenja je sigurnosni sistem baziran na Raspberry Pi ploči.

Raspberry Pi je računalna ploča koja omogućava razvoj različitih projekata, uključujući sigurnosne sisteme. U kombinaciji s različitim senzorima, Raspberry Pi može služiti kao centralna jedinica za kontrolu i nadzor nad sigurnosnim uređajima.

Ovaj završni rad opisuje proces izrade sigurnosnog sistema koji koristi PIR senzor pokreta za detekciju, zvučnik za alarmiranje i LED diodu za vizualnu indikaciju. Pored toga, sistem je integriran s e-mail uslugom za slanje obavijesti te s Blynk IoT aplikacijom koja omogućava daljinsko upravljanje i nadzor putem mobilne aplikacije. Rad se nalazi u maketi kućice koja omogućava vizualizaciju i praktičnu primjenu razvijenog sistema. Metodologija korištena u ovom radu uključuje istraživanje dostupnih tehnologija, praktičnu implementaciju hardverskih i softverskih komponenti, te testiranje funkcionalnosti sistema. Kroz praktične korake, demonstrirano je kako se različite komponente mogu integrirati kako bi se postigao željeni cilj. Razvijeni sigurnosni sistem ima široku primjenu u različitim okruženjima. Može se koristiti za zaštitu domova, ureda, trgovina i drugih objekata. Integracija s Blynk IoT aplikacijom omogućava korisnicima da daljinski nadziru i upravljaju svojim sigurnosnim sistemom, pružajući dodatni nivo sigurnosti i kontrole.

Ovaj rad bi trebao biti koristan izvor informacija i inspiracija za buduće projekte u sigurnosnim tehnologijama i IoT-a.

2. Cilj rada

Cilj ovog završnog rada je razvoj funkcionalnog i pouzdanog sigurnosnog sistema baziranog na Raspberry Pi ploči. Svrha projekta je istražiti i demonstrirati mogućnosti primjene Raspberry Pi-ja u sigurnosnim aplikacijama, koristeći PIR senzor za detekciju pokreta, zvučnik za zvučne alarme, LED diode za vizualnu indikaciju, te e-mail obavijesti za daljinsku informaciju o sigurnosnim događajima.

Na kraju, rad je smješten u maketu kućice kako bi se omogućila praktična vizualizacija i demonstracija rada sigurnosnog sistema. Cilj je prikazati kako ovakav sistem može biti primjenjiv u stvarnim uvjetima i pružiti sigurnost u raznim okruženjima poput domova, ureda i drugih objekata.

Ovim radom je pokazano kako kombinacija hardverskih i softverskih komponenti može rezultirati naprednim sigurnosnim rješenjem koje je efikasno, pouzdano i lako za korištenje.

3. Komponente i sredstva

3.1 Raspberry Pi

Slika 1 Raspberry Pi 3

Izvor: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

Raspberry Pi je linija malih, pristupačnih i moćnih računalnih ploča razvijenih od strane Raspberry Pi Foundation. Primarno je namijenjen obrazovanju u računalnim znanostima, no zbog svoje fleksibilnosti i sposobnosti, našao je široku primjenu u hobističkim i profesionalnim projektima.

Osnovne karakteristike

Procesor i memorija:

Ploče su opremljene ARM procesorima različitih brzina i broja jezgara, ovisno o modelu (npr. Raspberry Pi 4 ima četverojezgreni procesor).

Memorija varira između modela, s kapacitetima od 512MB do 8GB RAM-a, omogućavajući izvođenje složenijih zadataka.

Pohrana:

Umjesto tvrdog diska, koristi se microSD kartica za pohranu operativnog sistema i podataka.

Noviji modeli podržavaju USB boot, omogućavajući korištenje vanjskih SSD-ova ili tvrdih diskova.

Povezivanje:

GPIO Pinovi: General Purpose Input/Output pinovi omogućuju povezivanje različitih senzora, aktuatora i drugih perifernih uređaja. GPIO pinovi su ključni za projekte koji uključuju interakciju s vanjskim hardverom.

USB Portovi: Omogućuju povezivanje periferija poput tipkovnice, miša, vanjskih diskova itd.

HDMI Port: Povezivanje s monitorima ili televizorima, podržavajući visoku rezoluciju.

Ethernet i Wi-Fi: Integrirani su mrežni portovi za povezivanje na internet i lokalne mreže. Većina novijih modela također ima ugrađen Wi-Fi i Bluetooth.

Operativni sistem:

Raspberry Pi koristi prilagođeni operativni sistem pod nazivom Raspberry Pi OS (ranije poznat kao Raspbian), baziran na Linuxu. Dostupne su i druge distribucije poput Ubuntu-a i različitih specijaliziranih OS-ova za specifične primjene.

OS se instalira na microSD karticu, a dostupni su alati poput NOOBS (New Out Of Box Software) za jednostavniju instalaciju.

Napajanje:

Raspberry Pi koristi micro USB ili USB-C (ovisno o modelu) za napajanje, s naponom od 5V. Preporučuje se korištenje kvalitetnih napajanja kako bi se osigurala stabilnost.

Priključci za kameru i ekran:

CSI (Camera Serial Interface) port za povezivanje kamera omogućuje projekte vezane za video nadzor, snimanje i obradu slike.

DSI (Display Serial Interface) port za povezivanje zaslona omogućuje stvaranje kompaktnih uređaja sa zaslonom, poput tableta ili integriranih kontrolnih ploča.

3.2 PIR senzor

Slika 2 PIR senzor

Izvor: https://www.electronic.ba/bs/arduino-senzori/modul-arduino-pir-senzor-pokreta-hc-sr501

PIR (pasivni infracrveni) senzor je piroelektrični uređaj koji detektira kretanje mjerenjem promjena u infracrvenim razinama koje emitiraju okolni objekti. Ugradnjom Fresnelove leće i kruga detekcije pokreta, modul pruža visoku osjetljivost i nisku razinu šuma. Modul pruža optimizirani krug koji može detektirati kretanje na udaljenosti do 6 metara. Postoje dva utora na senzoru, svaki od posebnog materijala osjetljivog na IR. U nedostatku ičega, dva utora primaju istu količinu IC zračenja. Kada osoba prođe pokraj senzora, presreće ga jedna polovica utora što uzrokuje pozitivnu razliku potencijala između utora. Kada osoba ode pored senzora, presreće ga druga polovica utora što uzrokuje negativnu razliku potencijala između utora. Ova pozitivna i negativna razlika generira puls i tako otkriva kretanje. Senzorski modul ima ugrađen regulator napona od 3,3 V, zaštitnu diodu, podešavanje osjetljivosti i podešavanje vremena odgode. Na modulu postoje tri terminala – uzemljenje, VCC i digitalni izlaz. Napon od 5 V do 12 V može se napajati na VCC pinu, iako je 5 V preporučeno

napajanje. Kada modul otkrije kretanje, izlaz na pinu Digital Out postaje VISOK. Ovo je standardni aktivni visoki signal od 5 V. Digitalni izlazni pin senzora povezan je s GPIO pinovima Raspberry Pi izravno za praćenje signala. Spojen je na GPIO 4. pin Raspberry pi 3. VCC pin modula spojen je na jedan od 5V DC Power pina Pi 3, a uzemljeni pin modula je spojen na jedan od pinova uzemljenja Pi.

3.3 Zvučnik (Buzzer)

Slika 3 Buzzer

Izvor: https://soldered.com/hr/proizvod/buzzer-zvucnik-7mm/

Zvučnik, poznat kao buzzer, je elektronička komponenta koja proizvodi zvučni signal kada je napajan. Postoje dvije glavne vrste: piezoelektrični i elektromehanički. Piezoelektrični buzzeri koriste piezo materijal koji vibrira pod električnim naponom, dok elektromehanički buzzeri koriste elektromagnet i dijafragmu za stvaranje zvuka. Ovi uređaji su jednostavni za instalaciju, imaju nisku potrošnju energije i brzi odziv, što ih čini idealnim za upotrebu u sigurnosnim sistemima, industrijskoj automatizaciji i raznim elektronskim uređajima.

3.4 LED dioda

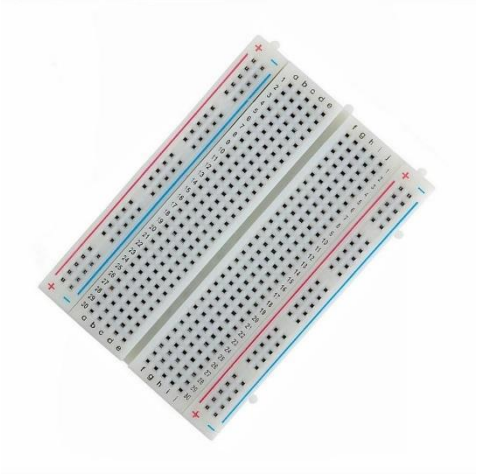
Slika 4 LED dioda

Izvor: https://kronos.hr/dioda-led-i-kucista/8080-led-5-crv-12v.html

LED dioda je poluvodički uređaj koji emitira svjetlost kada kroz njega teče električna struja. LED dioda su energetski učinkovite, dugotrajne i dolaze u različitim bojama i oblicima.. LED dioda se često koristi u rasvjeti, signalizaciji i elektroničkim uređajima zbog svoje pouzdanosti i niske potrošnje energije.

3.5 Breadboard (eksperimentalna pločica)

Slika 5 Breadboard



Izvor: https://www.cytron.io/p-breadboard-16.5x5.5cm-830-holes

Breadboard je alat za izradu i testiranje prototipova elektroničkih sklopova bez potrebe za lemljenjem. Sastoji se od mreže rupa povezanih metalnim kontaktima, omogućujući brzo i jednostavno spajanje komponenti. Središnji dio pločice ima dvije paralelne trake koje se koriste za povezivanje komponenti, dok vanjske trake obično služe kao naponske šine za distribuciju napajanja. Breadboard je idealan za izradu privremenih sklopova, eksperimentiranje i edukaciju u elektronici.

3.6 Spojni kablovi

Slika 6 Male to Female Jumper Wires

Izvor: https://www.indiamart.com/proddetail/breadboard-jumper-wires-male-female-22218795273.html

Slika 7 Female to Female Jumper Wires

Izvor: https://www.indiamart.com/proddetail/breadboard-jumper-wires-male-female-22218795273.html

Muško-ženski spojni kablovi (Male to Female Jumper Wires) imaju pin na jednom kraju i utičnicu na drugom te se koriste za povezivanje GPIO pinova na Raspberry Pi s komponentama na breadboardu. Žensko-ženski spojni kablovi (Female to Female Jumper Wires) imaju utičnice na oba kraja i koriste se za povezivanje komponenti s muškim pinovima, kao što su moduli ili senzori, direktno na Raspberry Pi ili Arduino ploče.

3.7 Otpornici od 220 Ohm-a

Slika 8 Otpornik

Izvor: https://italvideo-split.com/proizvod/otpornik-220-%CF%89-2w/

Otpornik koji se ovdje koristi ima 220 Ohma. Otpornik je pasivna elektronska komponenta koja ograničava protok električne struje u sklopu. Otpornici su obično označeni šarenim prstenovima koji specificiraju njihov otpor, u ovom slučaju 220 Ohma. Otpornik se koristi za zaštitu LED diode, sprječavajući preveliku struju koja bi mogla oštetiti diodu. To osigurava dugotrajan i pouzdan rad LED diode.

3.8 Opis makete drvene kućice za alarmni sustav

Maketa drvene kućice dimenzija 50x40 cm izrađena je pažljivo i precizno kako bi simulirala stvarni prostor koji će biti osiguran alarmnim sustavom. Kućica je izrađena od čvrste drvene ploče koja je obrađena radi estetskog izgleda. Konstrukcija je dovoljno čvrsta da izdrži montažu svih potrebnih elektroničkih komponenti i senzora.

Unutrašnjost kućice je dizajnirana tako da omogućuje jednostavnu instalaciju i pristup komponentama. Maketa ima jedna ulazna vrata koja omogućuju simulaciju ulaska u prostor. Krov nije prisutan kako bi se olakšala montaža i konfiguracija elektroničkih dijelova unutar kućice.

Elektroničke komponente, uključujući Raspberry Pi ploču, PIR senzor, LED diode i zvučnik, bit će smještene unutar makete na način koji simulira stvarni alarmni sustav. Komponente će biti raspoređene tako da optimalno pokrivaju unutarnji prostor makete i omogućuju efikasnu detekciju pokreta i odgovarajuće signalizacije.

Maketa služi kao praktični model za testiranje i demonstraciju funkcionalnosti razvijenog sigurnosnog sustava. Njen dizajn omogućuje lako prilagođavanje i proširenje sustava, pružajući korisnicima uvid u rad alarmnog sustava u kontroliranom okruženju.

Slika 9 Maketa drvene kućice

Izvor: Autor

4. Instalacija i podešavanje Raspberry Pi

4.1 Preuzimanje i instalacija OS-a

Posjeti se web stranica Raspberry Pi (https://www.raspberrypi.com/software/) i preuzme najnovija verzija operativnog sistema Raspberry Pi OS (Raspbian OS).

Priprema microSD kartice:

Stavi se Raspbian OS na SD karticu.

Povezivanje Hardvera:

Ubaci se microSD kartica u Raspberry Pi. Raspberry Pi se poveže na monitor putem HDMI kabla, priključi se tastatura i miš na USB portove. Poveže se napajanje pomoću micro USB-C kabla.

Pokretanje Raspberry Pi:

Kada se poveže napajanje, Raspberry Pi se automatski pokreće. Na ekranu se pojavi boot poruka i zatim se učita desktop okruženje Raspberry Pi OS-a.

Ažuriranje sistema:

Terminal se otvori klikom na ikonu terminala u gornjem lijevom kutu ekrana. Sljedeće naredbe služe za ažuriranje liste paketa i instalaciju dostupnih ažuriranja:

sudo apt-get update

sudo apt-get upgrade

Ove naredbe osigurale su najnovije verzije svih instaliranih paketa i sigurnosnih ažuriranja.

4.2 Povezivanje komponenti na Raspberry Pi

Komponente:

1. Raspberry Pi ploča
2. PIR senzor pokreta
3. Zvučnik (Buzzer)
4. LED dioda
5. Breadboard (eksperimentalna pločica)
6. Spojni kablovi (muško-ženski i žensko-ženski)
7. Otpornik 220 Ohma

Identifikacija pinova na PIR senzoru:

PIR senzor obično ima tri pina: VCC, GND i OUT.

VCC pin se koristi za napajanje senzora.

GND pin je uzemljenje.

OUT pin šalje signal kada se detektira pokret.

Spajanje PIR senzora na Breadboard:

Spojiti VCC pin PIR senzora na 5V pin Raspberry Pi pomoću muško-ženskog spoja.

Spojiti GND pin PIR senzora na GND pin na Raspberry Pi.

Povezivanje OUT pina:

Spojiti OUT pin PIR senzora na jedan od GPIO pinova na Raspberry Pi. Koristiti muško-ženski spojni kabel.

Identifikacija pinova na zvučniku:

Zvučnik obično ima dva pina: pozitivni (VCC) i negativni (GND).

Spajanje zvučnika na Breadboard:

Spojiti VCC pin zvučnika na jedan od GPIO pinova na Raspberry Pi.

Spojiti GND pin zvučnika na GND pin na Raspberry Pi.

Priprema LED žarulje:

LED žarulje imaju dva terminala: anoda (dulji pin) i katoda (kraći pin).

Za sigurnost i kontrolu struje, koristite otpornik od 220Ω u seriji s LED žaruljom.

Spajanje LED žarulje na Breadboard:

Postaviti LED žarulju i otpornik na breadboard tako da je anoda spojena na otpornik.

Povezivanje pinova LED žarulje:

Spojiti slobodni kraj otpornika na jedan od GPIO pinova na Raspberry Pi (preporučeno GPIO22).

Spojiti katodu LED žarulje na GND pin na Raspberry Pi.

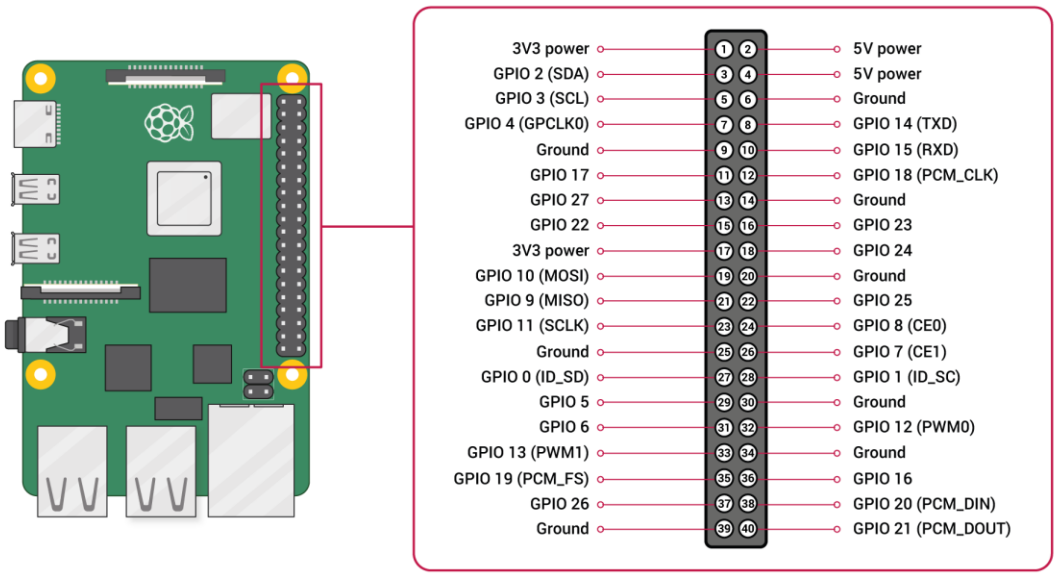
Provjera spojeva:

Provjeriti sve spojeve kako bi se osiguralo da su VCC i GND pinovi svih komponenti pravilno povezani.

4.3 Provjera GPIO spojeva:

Provjeriti spojeve GPIO pinova kako bi se osiguralo da su povezani na odgovarajuće pinove na Raspberry Pi ploči.

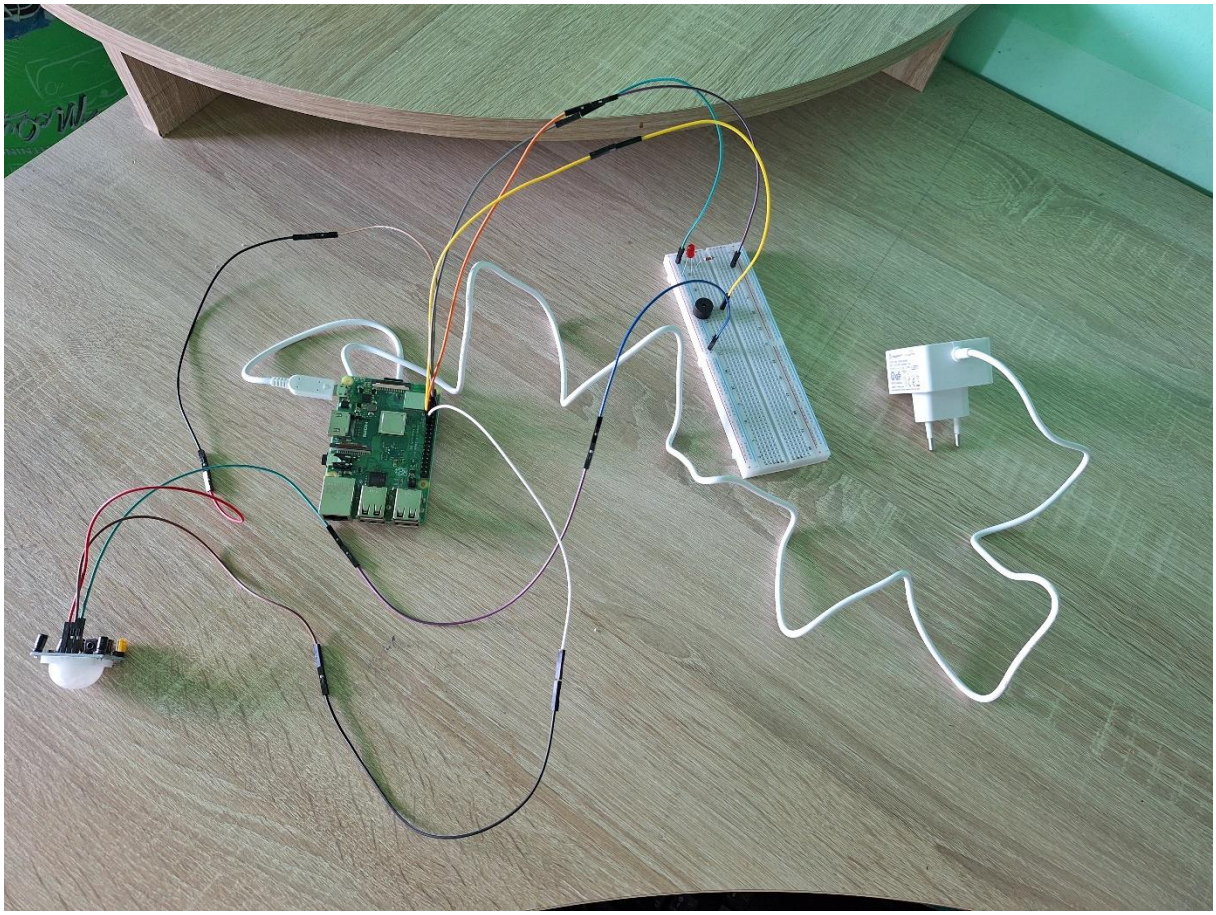
Slika 10 Raspberry Pi pinovi



Izvor: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

Ovako izgleda početna verzija u radu:

Slika 11 Početna verzija



Izvor: Autor

5. Konfiguracija e-mail usluge za Raspberry Pi

5.1 Kreiranje e-mail adrese

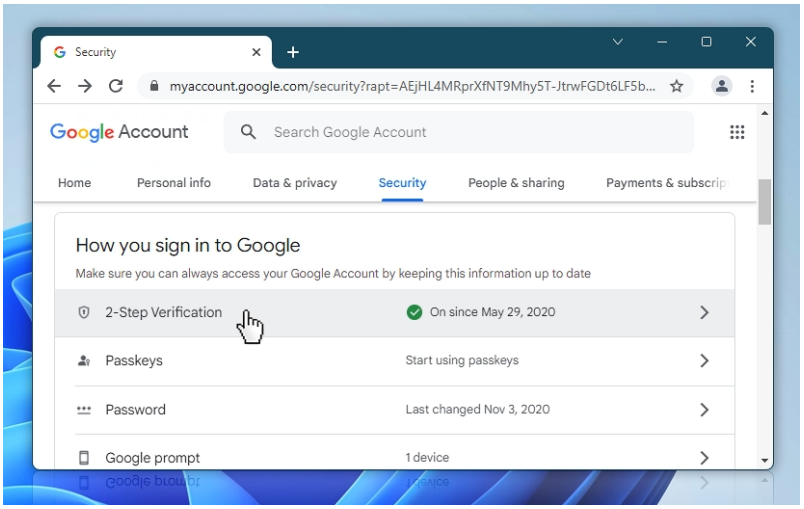
Registracija na Gmail:

Otvoriti web preglednik i posjetiti Gmail.

Kreirati novi Google račun koji će biti korišten za slanje e-mail obavijesti. Unijeti sve potrebne podatke kao što su ime, prezime, željena e-mail adresa i lozinka.

Omogućavanje dvofaktorske autentifikacije:

Slika 12 Omogućavanje dvofaktorske autentifikacije



Izvor: Autor

Prijaviti se na novi Gmail račun.

Posjetiti Google Account Security stranicu i omogućiti dvofaktorsku autentifikaciju.

Ovo su Gmail i lozinka:

rpi.sec2024@gmail.com

"2pw.,rpi2024

5.2 Kreiranje aplikacijske lozinke

(https://www.febooti.com/products/automation-workshop/tutorials/enable-google-app-passwords-for-smtp.html)

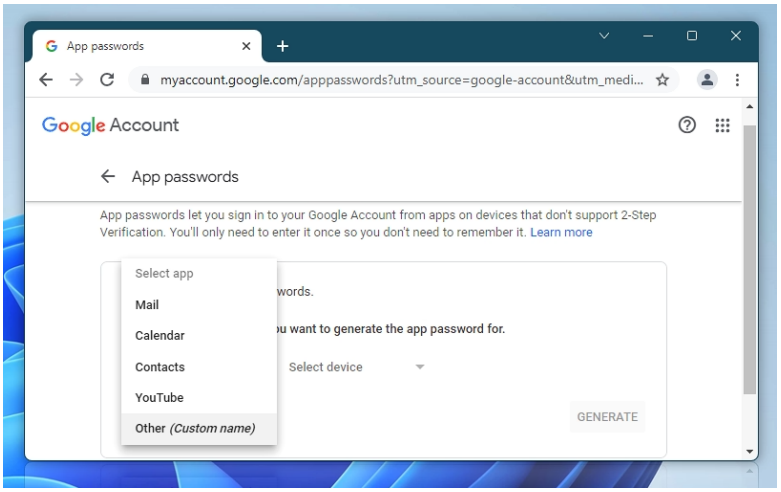
Pristup postavkama sigurnosti:

Nakon omogućavanja dvofaktorske autentifikacije, u odjeljku "Kako se prijaviti na Google" odabrati "App passwords".

Generiranje aplikacijske lozinke:

Kliknuti na "Select app", odabrati "Other (Custom name)" i unesti naziv, npr. "Automation Workshop ".

Slika 13 Unijeti Custom name

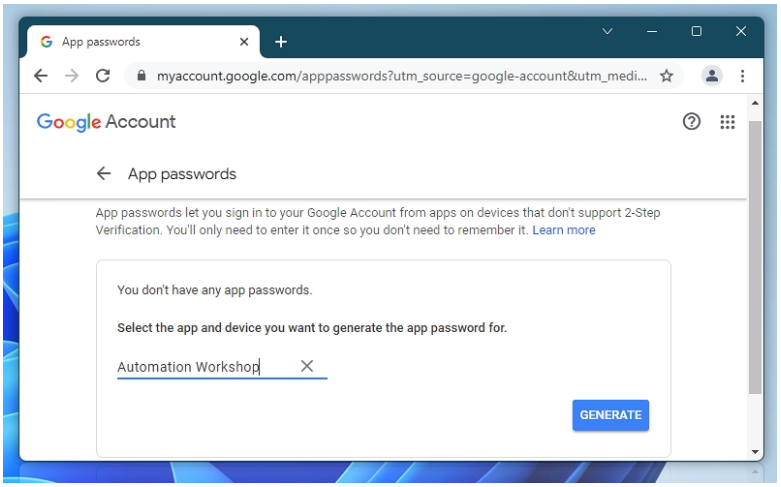


Izvor: Autor

Kliknuti na "Generate" kako bi se dobio 16-znamenkasti kod.

Ovo je taj kod: dccofrzadienyvjx

Slika 14 Generiranje aplikacijske lozinke



Izvor: Autor

5.3 Instalacija i konfiguracija SMTP servera na Raspberry Pi

Smtplib je standardna Python biblioteka za slanje e-mailova putem SMTP-a. Ova biblioteka je već uključena u standardnu Python biblioteku, tako da nije potrebna dodatna instalacija.

Zatim slijedi pisanje Python skripte za slanje e-mailova.

Generirani kod je bilo potrebno upisati u Python kod:

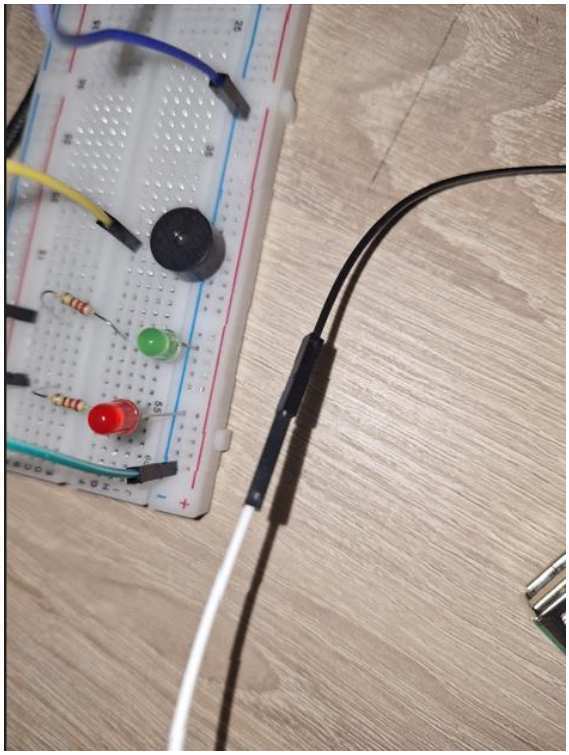
from_email_password = "dccofrzadienyvjx"

Provjeriti pristiglu poštu na adresi david.cavlek5@gmail.com kako bi se vidjelo da je e-mail obavijest uspješno poslana. Ako je e-mail uspješno poslan, treba se vidjeti poruka s naslovom "[WARNING!] Security System - Intruder Alert!".

6. Povezivanje s Blynk IoT aplikacijom

Prije povezivanja s Blynk aplikacijom dodana je jedna zelena LED dioda koja će se moći uključivati i isključivati preko aplikacije. Dioda je povezana s Raspberry pi pinom 4. Još je dodana jedna plava LED dioda na pin 23 koja se uključuje kada se Raspberry pi spoji na server od Blynk Iot aplikacije.

Slika 15 Dodavanje druge LED diode



Izvor: Autor

6.1 Registracija na Blynk

Preuzimanje Aplikacije na mobitelu.

Registracija:

rpi.sec2024@gmail.com

Lozinka: "2pw.,rpi2024

6.2 Kreiranje novog projekta

Kliknuti na ikonu '+' ili 'Create New Project' unutar Blynk aplikacije.

Unijeti naziv projekta "Raspberry Pi Security System“ i odabrati "Raspberry Pi" kao hardversku ploču.

Primanje AUTH_TOKEN-a:

Nakon kreiranja projekta, Blynk će poslati AUTH_TOKEN na e-mail adresu. Ovaj token će se koristiti za povezivanje Raspberry Pi-ja sa Blynk serverima.

Ovo je moj token:

AE6KXE-uyo8szeDQclZEeDnp_pdcH0M3

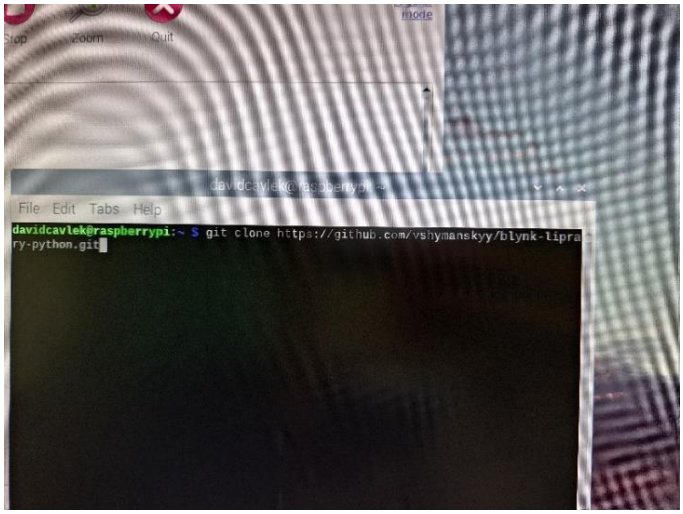
6.3 Pisanje Python skripte za povezivanje s Blynk-om

BLYNK_AUTH_TOKEN = 'AE6KXE-uyo8szeDQclZEeDnp_pdcH0M3'

Ovo je bilo potrebno upisati u Raspberry Pi terminal:

git clone https://github.com/vshymanskyy/blynk-library-python.git

Slika 16 Komanda u terminalu za povezivanje s aplikacijom



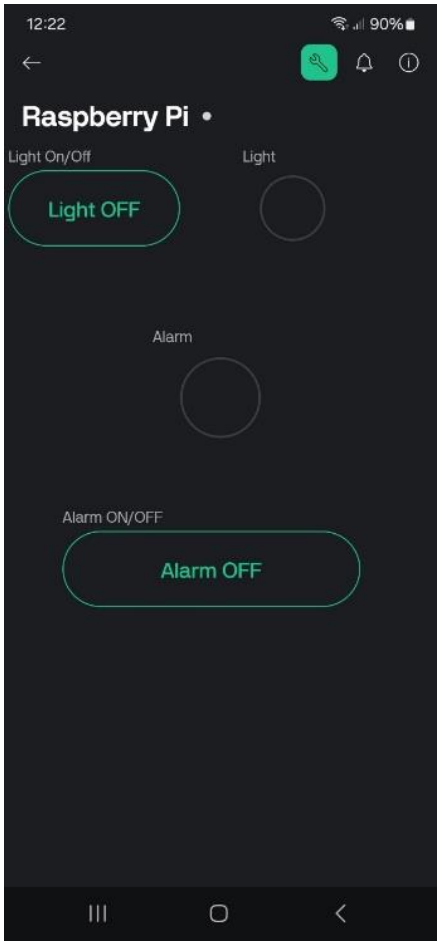
Izvor: Autor

Otvoriti Blynk aplikaciju na mobilnom uređaju. Dodati widgete za kontrolu i Button za V0 virtualni pin, te ih povezati s odgovarajućim funkcijama. Zatim se pritisne gumb u aplikaciji

kako bi se provjerilo uključuje li se i isključuje LED dioda. Zatim se treba dodati još jedan gumb za uključivanje i isključivanje alarma i prilagoditi to u kodu.

Ovako izgleda u aplikaciji:

Slika 17 Screenshot aplikacije



Izvor: Autor

7. Automatsko pokretanje aplikacije na Raspberry Pi

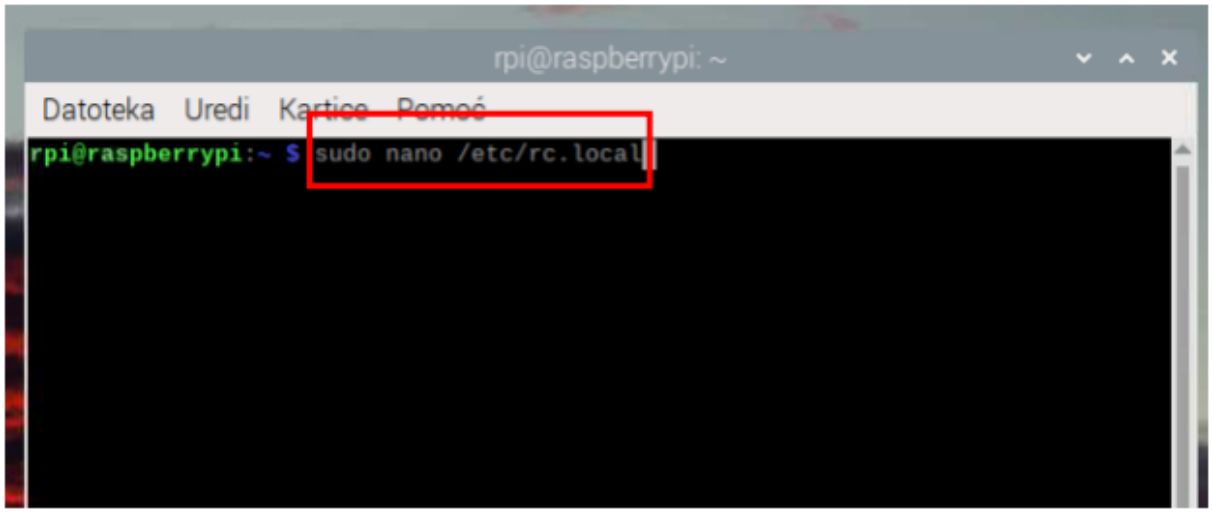
U ovom koraku se objašnjava kako konfigurirati Raspberry Pi da automatski pokrene aplikaciju odmah nakon uključivanja. Ovo je korisno za osiguravanje sigurnosnog sistema kako bi bio stalno aktivan bez potrebe za ručnim pokretanjem aplikacije nakon svakog ponovnog pokretanja uređaja.

7.1 Koraci za automatsko pokretanje:

Na Raspberry Pi otvoriti terminal. U terminal upisati sljedeću naredbu kako bi otvorili rc.local datoteku u nano editoru:

sudo nano /etc/rc.local

Slika 18 Naredba u terminalu za otvaranje rc.local



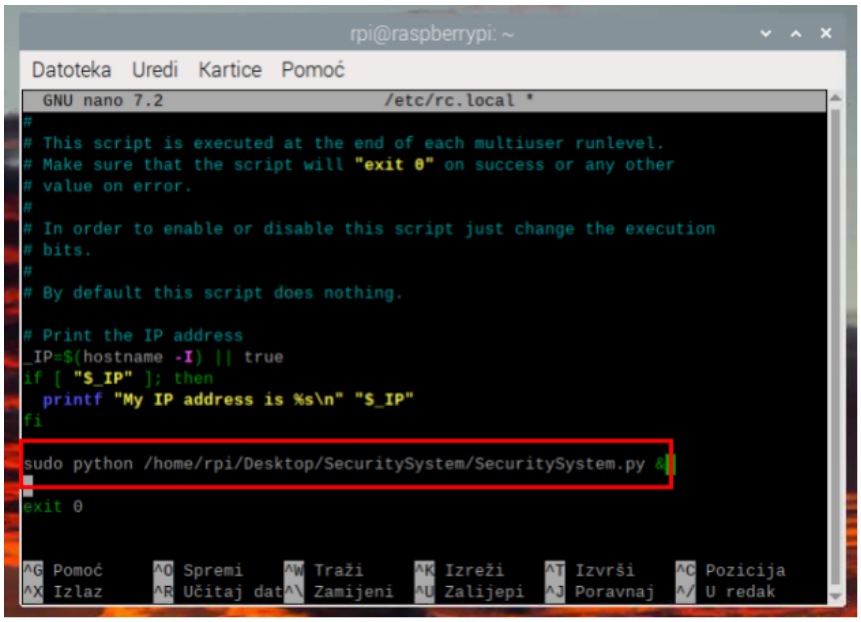
Izvor: Autor

Na kraju datoteke rc.local, odmah prije retka koji sadrži exit 0, dodati sljedeću naredbu:

sudo python /home/rpi/Desktop/SecuritySystem/SecuritySystem.py

Ova naredba će osigurati da se aplikacija SecuritySystem.py pokrene svaki put kada se Raspberry Pi pokrene. Za to je potrebna putanja do datoteke.

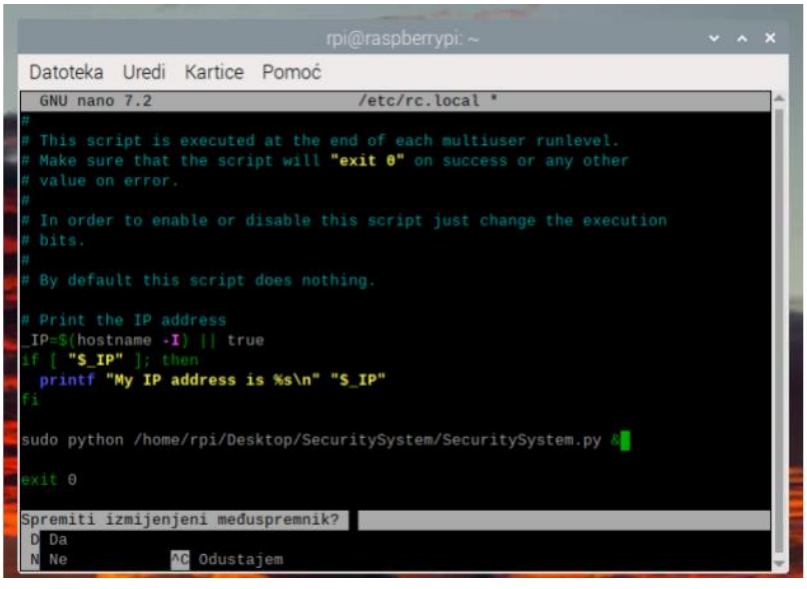
Slika 19 Putanja do datoteke



Izvor: Autor

Nakon dodavanja linije koda, spremiti datoteku pritiskom na CTRL + X, zatim Y za potvrdu i ENTER za spremanje promjena.

Slika 20 Pritisak ENTER za potvrdu

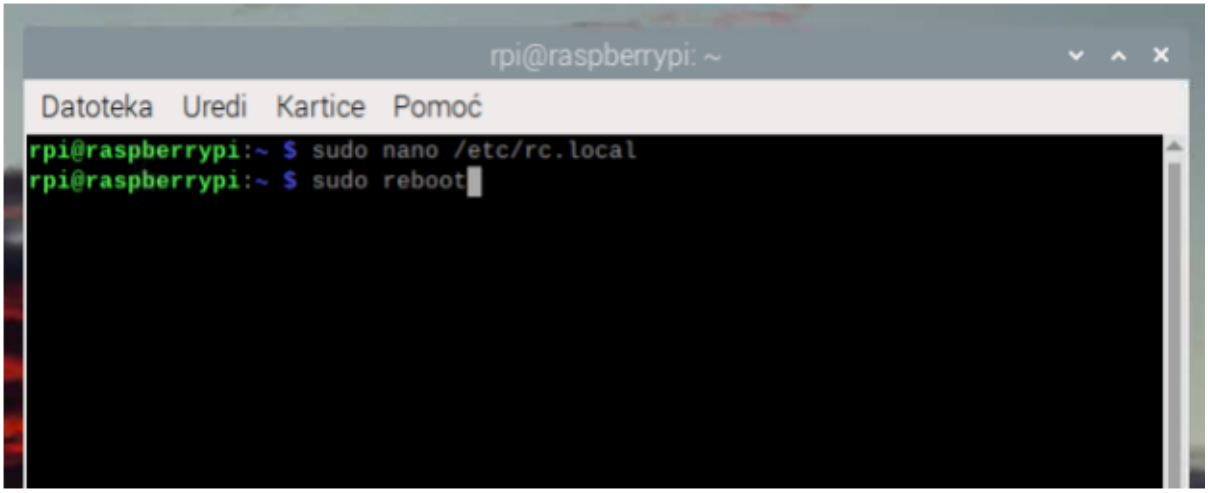


Izvor: Autor

Da bi se primijenile promjene, ponovno treba pokrenuti Raspberry Pi upisom sljedeće naredbe u terminal:

sudo reboot

Slika 21 Naredba za resetiranje



Izvor: Autor

Nakon što se Raspberry Pi ponovno pokrene, aplikacija SecuritySystem.py trebala bi se automatski pokrenuti.

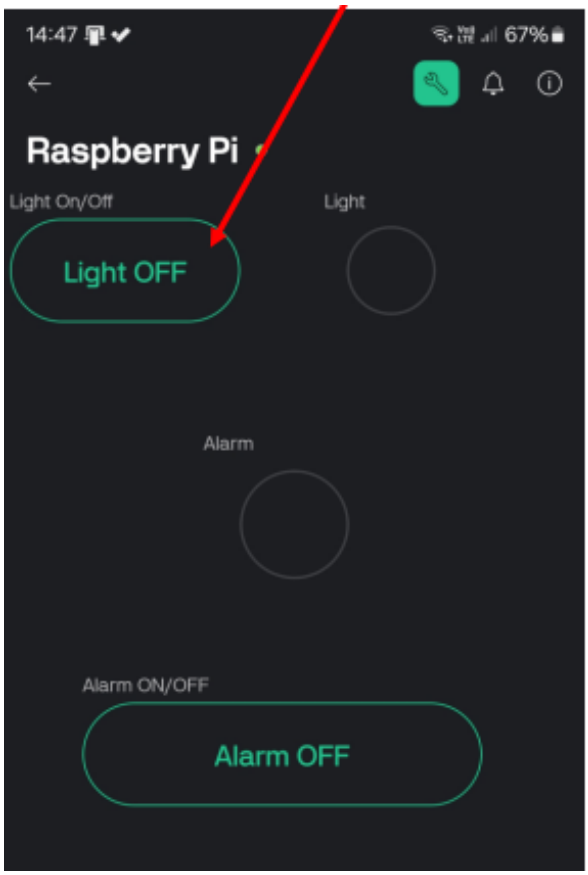
Kontrola LED diode**:**

Prekidačem Light OFF/Light ON u Blynk aplikaciji može se uključivati i isključivati LED dioda. Detekcija pokreta i alarm neće biti aktivni dok ih se ne uključi.

7.2 Aktivacija alarma**:**

Alarm će biti neaktivan dok se ne pritisne prekidač Alarm OFF/Alarm ON u Blynk aplikaciji. Ovo omogućuje aktiviranje funkcije alarma.

Slika 22 Gumb za paljenje i gašenje LED žarulje

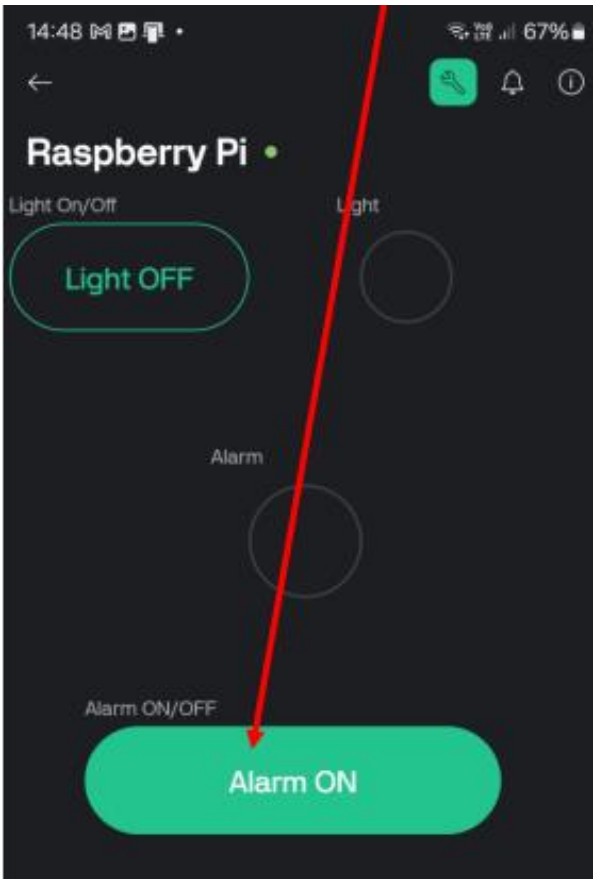


Izvor: Autor

7.3 Detekcija pokreta i alarm:

Kada PIR senzor detektira pokret, alarm se aktivira. Tada se šalje e-mail obavijest, signalizira se crvenom LED diodom na Raspberry Pi-u i u Blynk aplikaciji, te se aktivira zvučnik(buzzer). Trajanje alarma je postavljeno na 15 sekundi. Ako se pritisne tipka Alarm OFF/Alarm ON prije isteka 15 sekundi, alarm će se isključiti.

Slika 23 Gumb za uključivanje i isključivanje alarma



Izvor: Autor

Bilo je potrebno preuzeti dvije Python datoteke kako bi se aplikacija mogla povezati s mojim Raspberry Pi-em. Prva datoteka je BlynkLib.py, a druga BlynkTimer.py.

7.4 Kod za BlynkLib.py:

```
 __version__ = "1.0.0"

import struct
import time
import sys
import os

try:
    import machine
```

```
    gettime = lambda: time.ticks_ms()
    SOCK_TIMEOUT = 0
except ImportError:
    const = lambda x: x
    gettime = lambda: int(time.time() * 1000)
    SOCK_TIMEOUT = 0.05

def dummy(*args):
    pass

MSG_RSP = const(0)
MSG_LOGIN = const(2)
MSG_PING  = const(6)

MSG_TWEET = const(12)
MSG_NOTIFY = const(14)
MSG_BRIDGE = const(15)
MSG_HW_SYNC = const(16)
MSG_INTERNAL = const(17)
MSG_PROPERTY = const(19)
MSG_HW = const(20)
MSG_HW_LOGIN = const(29)
MSG_EVENT_LOG = const(64)

MSG_REDIRECT  = const(41)  # TODO: not implemented
MSG_DBG_PRINT  = const(55) # TODO: not implemented

STA_SUCCESS = const(200)
STA_INVALID_TOKEN = const(9)

DISCONNECTED = const(0)
CONNECTING = const(1)
CONNECTED = const(2)

print("""

    ___  __           __
   / _ )/ /_ _____  / /__
  / _  / / // / _ \\/  '_/
 /____/_/\\_, /_//_/_/\\_\\
```

```
        /___/ for Python v""" + __version__ + " (" + sys.platform +
")\n")

class EventEmitter:
    def __init__(self):
        self._cbks = {}

    def on(self, evt, f=None):
        if f:
            self._cbks[evt] = f
        else:
            def D(f):
                self._cbks[evt] = f
                return f
            return D

    def emit(self, evt, *a, **kv):
        if evt in self._cbks:
            self._cbks[evt](*a, **kv)



class BlynkProtocol(EventEmitter):
    def     __init__(self,     auth,     tmpl_id=None,     fw_ver=None,
heartbeat=50, buffin=1024, log=None):
        EventEmitter.__init__(self)
        self.heartbeat = heartbeat*1000
        self.buffin = buffin
        self.log = log or dummy
        self.auth = auth
        self.tmpl_id = tmpl_id
        self.fw_ver = fw_ver
        self.state = DISCONNECTED
        self.connect()

    def virtual_write(self, pin, *val):
        self._send(MSG_HW, 'vw', pin, *val)

    def send_internal(self, pin, *val):
        self._send(MSG_INTERNAL,  pin, *val)
```

```
    def set_property(self, pin, prop, *val):
        self._send(MSG_PROPERTY, pin, prop, *val)

    def sync_virtual(self, *pins):
        self._send(MSG_HW_SYNC, 'vr', *pins)

    def log_event(self, *val):
        self._send(MSG_EVENT_LOG, *val)

    def _send(self, cmd, *args, **kwargs):
        if 'id' in kwargs:
            id = kwargs.get('id')
        else:
            id = self.msg_id
            self.msg_id += 1
            if self.msg_id > 0xFFFF:
                self.msg_id = 1

        if cmd == MSG_RSP:
            data = b''
            dlen = args[0]
        else:
            data = ('\0'.join(map(str, args))).encode('utf8')
            dlen = len(data)

        self.log('<', cmd, id, '|', *args)
        msg = struct.pack("!BHH", cmd, id, dlen) + data
        self.lastSend = gettime()
        self._write(msg)

    def connect(self):
        if self.state != DISCONNECTED: return
        self.msg_id = 1
        (self.lastRecv, self.lastSend, self.lastPing) = (gettime(),
0, 0)
        self.bin = b""
        self.state = CONNECTING
        self._send(MSG_HW_LOGIN, self.auth)
```

```
    def disconnect(self):
        if self.state == DISCONNECTED: return
        self.bin = b""
        self.state = DISCONNECTED
        self.emit('disconnected')

    def process(self, data=None):
        if not (self.state == CONNECTING or self.state ==
CONNECTED): return
        now = gettime()
        if now - self.lastRecv > self.heartbeat+(self.heartbeat//2):
            return self.disconnect()
        if (now - self.lastPing > self.heartbeat//10 and
            (now - self.lastSend > self.heartbeat or
             now - self.lastRecv > self.heartbeat)):
            self._send(MSG_PING)
            self.lastPing = now

        if data != None and len(data):
            self.bin += data

        while True:
            if len(self.bin) < 5:
                break

            cmd, i, dlen = struct.unpack("!BHH", self.bin[:5])
            if i == 0: return self.disconnect()

            self.lastRecv = now
            if cmd == MSG_RSP:
                self.bin = self.bin[5:]


                self.log('>', cmd, i, '|', dlen)
                if self.state == CONNECTING and i == 1:
                    if dlen == STA_SUCCESS:
                        self.state = CONNECTED
                        dt = now - self.lastSend
```

```
                        info = ['ver', __version__, 'h-beat',
self.heartbeat//1000, 'buff-in', self.buffin, 'dev', sys.platform+'-
py']
                        if self.tmpl_id:
                            info.extend(['tmpl', self.tmpl_id])
                            info.extend(['fw-type', self.tmpl_id])
                        if self.fw_ver:
                            info.extend(['fw', self.fw_ver])
                        self._send(MSG_INTERNAL, *info)
                        try:
                            self.emit('connected', ping=dt)
                        except TypeError:
                            self.emit('connected')
                    else:
                        if dlen == STA_INVALID_TOKEN:
                            self.emit("invalid_auth")
                            print("Invalid auth token")
                        return self.disconnect()
            else:
                if dlen >= self.buffin:
                    print("Cmd too big: ", dlen)
                    return self.disconnect()

                if len(self.bin) < 5+dlen:
                    break

                data = self.bin[5:5+dlen]
                self.bin = self.bin[5+dlen:]

                args = list(map(lambda x: x.decode('utf8'),
data.split(b'\0')))

                self.log('>', cmd, i, '|', ','.join(args))
                if cmd == MSG_PING:
                    self._send(MSG_RSP, STA_SUCCESS, id=i)
                elif cmd == MSG_HW or cmd == MSG_BRIDGE:
                    if args[0] == 'vw':
                        self.emit("V"+args[1], args[2:])
                        self.emit("V*", args[1], args[2:])
                elif cmd == MSG_INTERNAL:
```

```
                    self.emit("internal:"+args[0], args[1:])
                elif cmd == MSG_REDIRECT:
                    self.emit("redirect", args[0], int(args[1]))
                else:
                    print("Unexpected command: ", cmd)
                    return self.disconnect()

import socket

class Blynk(BlynkProtocol):
    def __init__(self, auth, **kwargs):
        self.insecure = kwargs.pop('insecure', False)
        self.server = kwargs.pop('server', 'blynk.cloud')
        self.port = kwargs.pop('port', 80 if self.insecure else 443)
        BlynkProtocol.__init__(self, auth, **kwargs)
        self.on('redirect', self.redirect)

    def redirect(self, server, port):
        self.server = server
        self.port = port
        self.disconnect()
        self.connect()

    def connect(self):
        print('Connecting to %s:%d...' % (self.server, self.port))
        s = socket.socket()
        s.connect(socket.getaddrinfo(self.server, self.port)[0][-1])
        try:
            s.setsockopt(socket.IPPROTO_TCP, socket.TCP_NODELAY, 1)
        except:
            pass
        if self.insecure:
            self.conn = s
        else:
            try:
                import ussl
                ssl_context = ussl
            except ImportError:
                import ssl
```

```
                ssl_context = ssl.create_default_context()
            self.conn           =           ssl_context.wrap_socket(s,
server_hostname=self.server)
        try:
            self.conn.settimeout(SOCK_TIMEOUT)
        except:
            s.settimeout(SOCK_TIMEOUT)
        BlynkProtocol.connect(self)

    def _write(self, data):
        #print('<', data)
        self.conn.write(data)
        # TODO: handle disconnect

    def run(self):
        data = b''
        try:
            data = self.conn.read(self.buffin)
            #print('>', data)
        except KeyboardInterrupt:
            raise
        except socket.timeout:
            # No data received, call process to send ping messages
when needed
            pass
        except: # TODO: handle disconnect
            return
        self.process(data)
```

7.5 Kod za BlynkTimer.py:

```
import time
class BlynkTimer:
    '''Executes functions after a defined period of time'''
    _MAX_TIMERS = 16
    def __init__(self):
        self.timers = []
        self.ids = self._get_unique_id()

    def _get_unique_id(self, current=0):
        '''yields unique id for new timer'''
        numId = current
        while numId < self._MAX_TIMERS:
            yield numId
            numId += 1

    def _add(self, func, **kwargs):
        '''Inits Timer'''
        timerId = next(self.ids)
        timer = Timer(timerId, func, **kwargs)
        self.timers.append(timer)
        return timer

    def _get(self, timerId):
        '''Gets timer by id'''
        timer = [t for t in self.timers if t.id == timerId]
        if len(timer) <= 0:
            return None
        return timer[0]
    def _delete(self, timerId):
        '''Deletes timer'''
        timer = self._get(timerId)
        timer.disable()
        self.timers = [t for t in self.timers if t.id != timerId]
        num_timers = self.get_num_timers()[0]
        self.ids = self._get_unique_id(current=num_timers)
        return timerId
    def get_num_timers(self):
```

```
        '''Returns number of used timer slots'''
        num_timers = len(self.timers)
        return (num_timers, self._MAX_TIMERS)

    def is_enabled(self, timerId):
        '''Returns true if timer is enabled'''
        timer = self._get(timerId)
        return timer.enabled

    def set_interval(self, value, func):
        '''Sets time interval for function'''
        timer = self._add(func)
        timer.set_interval(value)
        return timer.id

    def set_timeout(self, value, func):
        '''Runs function once after timeout'''
        timer = self._add(func, post_run=self._delete)
        timer.set_interval(value)
        return timer.id

    def enable(self, timerId):
        '''Enables timer'''
        timer = self._get(timerId)
        timer.enable()
        return timerId

    def disable(self, timerId):
        '''Disables timer'''
        timer = self._get(timerId)
        timer.disable()
        return timerId

    def run(self):
        '''Polls timers'''
        [t.run() for t in self.timers]

class Timer:
    '''Runs function after specific interval'''
```

```
def __init__(self, id, func, **kwargs):
    self.id = id
    self.func = func
    self.interval = None
    self.start_time = None
    self.enabled = False
    self.on_post_run = kwargs.get('post_run', None)

def _handle_post_run(self):
    '''handles post run events'''
    self.start_time += self.interval
    if self.on_post_run:
        return self.on_post_run(self.id)

def enable(self):
    '''enables Timer'''
    self.enabled = True
    self.start_time = time.time()

def disable(self):
    '''disables timer'''
    self.enabled = False
    self.start_time = None

def set_interval(self, value):
    '''Sets Time Interval for calling function'''
    self.interval = value
    self.enable()

def run(self):
    '''Runs function if interval has passed'''
    if not self.enabled:
        return
    now = time.time()
    if now - self.start_time > self.interval:
        self.func()
        self._handle_post_run()
```

8. Zaključak

Izrada sigurnosnog sistema baziranog na Raspberry Pi ploči pruža priliku za stjecanje dubljeg razumijevanja hardverskih i softverskih dijelova IoT tehnologija. Kroz korake instalacije operativnog sistema, povezivanja komponenti, konfiguracije e-mail usluge i integracije s Blynk IoT aplikacijom, stvara se funkcionalan i pouzdan sigurnosni sistem.

Tijekom rada postoje različiti izazovi, poput pravilne konfiguracije dvofaktorske autentifikacije i kreiranja aplikacijskih lozinki za sigurnu komunikaciju s Gmail SMTP serverom. Također se uči koristiti Blynk aplikaciju za daljinsko upravljanje i nadzor nad sustavom, što značajno poboljšava korisničko iskustvo i praktičnost sustava.

Projekt demonstrira kako se relativno jeftin i kompaktan uređaj poput Raspberry Pi može koristiti za izradu naprednih sigurnosnih rješenja koja su primjenjiva u stvarnim uvjetima. Korištenjem senzora te integracijom s mobilnim aplikacijama stvara se višenamjenski sistem koji može služiti za zaštitu domova, ureda i drugih objekata.

Ovaj rad može poslužiti kao temelj za daljnja istraživanja i unapređenja sigurnosnih sistema baziranih na IoT tehnologijama, te kao inspiracija budućim studentima i profesionalcima u ovom području.

Izjava o autorstvu

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

Bana Josipa Jelačića 22/a, Čakovec

IZJAVA O AUTORSTVU

Završni/diplomski rad isključivo je autorsko djelo studenta 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, internetskih i drugih izvora) bez pravilnog citiranja. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom i nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, DAVID ČAVLEK (ime i prezime studenta) pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog rada pod naslovom IZRADA MODULARNOG ALARMNOG SUSTAVA

te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:

(vlastoručni potpis)

9. Popis slika

Slika 1 Raspberry Pi 3 3
Slika 2 PIR senzor 5
Slika 3 Buzzer 6
Slika 4 LED dioda 7
Slika 5 Breadboard 8
Slika 6 Male to Female Jumper Wires 9
Slika 7 Female to Female Jumper Wires 9
Slika 8 Otpornik 10
Slika 9 Maketa drvene kućice 11
Slika 10 Raspberry Pi pinovi 14
Slika 11 Početna verzija 15
Slika 12 Omogućavanje dvofaktorske autentifikacije 16
Slika 13 Unijeti Custom name 17
Slika 14 Generiranje aplikacijske lozinke 17
Slika 15 Dodavanje druge LED diode 19
Slika 16 Komanda u terminalu za povezivanje s aplikacijom 20
Slika 17 Screenshot aplikacije 21
Slika 18 Naredba u terminalu za otvaranje rc.local 22
Slika 19 Putanja do datoteke 23
Slika 20 Pritisak ENTER za potvrdu 23
Slika 21 Naredba za resetiranje 24
Slika 22 Gumb za paljenje i gašenje LED žarulje 25
Slika 23 Gumb za uključivanje i isključivanje alarma 26

10. Popis literature

1. https://support.google.com/mail/answer/185833?hl=en
2. https://www.febooti.com/products/automation-workshop/tutorials/enable-google-app-passwords-for-smtp.html
3. https://youtu.be/RIbqMIB0p9g
4. https://www.youtube.com/watch?v=RIbqMIB0p9g&ab_channel=SMEDehradun
5. https://www.youtube.com/watch?v=Tw0mG4YtsZk&ab_channel=TechWithTim
6. https://projects.raspberrypi.org/en/projects/physical-computing/11
7. https://www.raspberrypi.com/software/
8. William Pretty, Software_Building a High-Tech Alarm System with Raspberry Pi

11. Programski kod

```
#include the libraries
from gpiozero import MotionSensor, Buzzer, LED
import smtplib
from email.message import EmailMessage
from signal import pause
from time import sleep
import BlynkLib
from BlynkTimer import BlynkTimer

#Blynk IOT credentials
BLYNK_TEMPLATE_ID = 'TMPL4XEpjFreQ'
BLYNK_TEMPLATE_NAME = 'rpi'
BLYNK_AUTH_TOKEN = 'AE6KXE-uyo8szeDQclZEeDnp_pdcH0M3'

#include the PIR sensor pin
pirSensor = MotionSensor(22)
#include the buzzer pin
buzzer = Buzzer(27)
#include the LEDs pins
ledStatus = LED(17)
ledLight = LED(4)

# Initialize Blynk
blynk = BlynkLib.Blynk(BLYNK_AUTH_TOKEN)

#email credentials
from_email_addr = "rpi.sec2024@gmail.com"
from_email_password = "dccofrzadienyvjx"
to_email_addr = "david.cavlek5@gmail.com"
email_subject = "[WARNING!] Security System - Intruder Alert!"
email_body = "Motion Detected!"

#control variable
email_sent = False
alarm_active = False
```

```
#function - motion is detected
def motion_detected():
    global email_sent
    global alarm_active
    if(alarm_active == True):
        if(email_sent == False):
            print("Motion Detected!")
            ledStatus.on()
     #status LED on
            buzzer.beep()
     #alarm buzzer beep
            msg = EmailMessage()
     #create a message object
            msg.set_content(email_body)
     #set the email body
            msg['From'] = from_email_addr                          #set
sender and recipient addresses
            msg['To'] = to_email_addr
            msg['Subject'] = email_subject                         #set
email subject
            server = smtplib.SMTP('smtp.gmail.com', 587)   #connect
to gmail SMTP server and send email
            server.starttls()
            server.login(from_email_addr, from_email_password)
            server.send_message(msg)
            server.quit()
            email_sent = True
            print('email sent')
            blynk.virtual_write(1,1)
     #Blynk alarm status on
            sleep(15)
     #alarm active for 15 seconds before next check
            email_sent = False
            blynk.virtual_write(1,0)
     #Blynk alarm status off

#function - motion is not detected
def motion_not_detected():
    print("Motion Not Detected!")
    ledStatus.off()                          #status LED OFF
    buzzer.off()                       #alarm buzzer off
    sleep(5)
```

```
#function - alarm off
def alarm_off():
    ledStatus.off()                         #status LED OFF
    buzzer.off()                        #alarm buzzer off
    blynk.virtual_write(1,0)          #Blynk alarm status off

#function - LED control through V0 virtual pin
@blynk.on("V0")
def v0_write_handler(value):
    #LED light switch
    if int(value[0]) != 0:
        ledLight.on()
        print("LIGHT ON")
    else:
        ledLight.off()
        print("LIGHT OFF")

#function - Alarm control through V2 virtual pin
@blynk.on("V2")
def v2_write_handler(value):
    global alarm_active
    #Alarm switch
    if int(value[0]) != 0:
        alarm_active = True
        print("ALARM ACTIVE")
    else:
        alarm_active = False
        alarm_off()
        print("ALARM NOT ACTIVE")

#assign a function that runs when motion is detected
pirSensor.when_motion = motion_detected

#assign a function that runs when motion is not detected
pirSensor.when_no_motion = motion_not_detected

#function to sync the data from virtual pins
@blynk.on("connected")
```

```
def blynk_connected():
    print("Raspberry Pi Security System Connected to Blynk IOT")

while True:
    blynk.run()       #Blynk run
```