Model pametne kuće

Bugarin, Mario

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

MODEL PAMETNE KUĆE

Rijeka, Studeni 2022.

Mario Bugarin

0069079710

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij elektrotehnike

Diplomski rad

MODEL PAMETNE KUĆE

Mentor: prof. dr. sc. Dario Matika

Rijeka, Studeni 2022.

Mario Bugarin

0069079710

Orginal Zadatka

IZJAVA

Sukladno članku 9. te 1. stavkom Pravilnika o diplomskom radu, diplomskom ispitu i završetku diplomskih sveučilišnih studija Tehničkog fakulteta u Rijeci od 29. lipnja 2011., izjavljujem da sam ovaj diplomski rad naziva „Model pametne kuće“ izradio samostalno.
Diplomski rad izrađen je u periodu od 20.03.2022 do 01.10.2022.

Rijeka, Studeni 2022.

Mario Bugarin

ZAHVALA

Zahvaljujem se obitelji, djevojci i prijateljima na podršci i pomoći tokom svih pet godina studiranja.

Također se zahvaljujem svome mentoru prof. dr. sc. Dariu Matiki na korisnim savjetima i pomoći u izradi ovog diplomskog rada.

Sadržaj

1. UVOD

Pojam automatizacije najbolje opisuje korištenje raznovrsne tehnologije za obavljanje zadataka u cilju smanjenja ljudskoga rada. Automatizacija uključuje korištenje sustava upravljanja i različite opreme kao što su strojevi, telekomunikacijske mreže, procesi u tvornicama, upravljanje i stabilizacija brodova i zrakoplova, autonomna vozila, pametne kuće i drugi. Primjena automatizacije je veoma raznolika i opširna, od jednostavnih automatskih garažnih vrata kod kuće do velikih industrijskih upravljačkih SCADA (eng. *supervisory control and data acquisition*) sustava s tisućama mjerenja ulaznih i izlaznih kontrolnih signala. Prema kompleksnosti upravljanja, upravljački sustavi mogu se kretati od jednostavne on-off kontrole do kompleksnih algoritama visoke razine sa velikim brojem varijabli i korištenja umjetne inteligencije. Automatizacija omogućuje daljinsko upravljanje i nadzor procesa, smanjuje udio ljudskog rada, opažanja i odlučivanja te služi kao nastavak procesa mehanizacije.

Sve raširenija primjena automatizacije nalazi se u sustavima automatizacije kuće. Napretkom tehnologije, pojam „pametna kuća“ nastavlja se razvijati te se povećava broj njezinih aplikacija. Pametna kuća sastoji se od više pametnih uređaja koji zajednički čine sustav koji osigurava automatiziranost prostora. Sustav upravlja i nadzire kućne aparate i sustave kao što su klima, rasvjeta, grijanje, sustavi za zabavu te uređaje bijele tehnike. Također može uključivati sustave sigurnosti doma kao što su kontrola pristupa i alarmni sustavi.

U novije vrijeme ključni pokretač transformacije sustava pametnih kuća je tehnologija internet stvari (eng. *Internet of Things* - IoT). IoT nudi funkcije povezivanja klasičnih uređaja sa internetom u cilju dijeljenja podataka ili dobivanja instrukcija što je omogućilo razvoj industrije pametnih kuća. IoT donosi internetsku povezanost, obradu podataka i analitiku u svijet fizičkih objekata. Za korisnike to znači interakciju s globalnom informacijskom mrežom u stvarnom vremenu, gdje god se nalazili.

U ovome radu biti će predstavljen pojam i ideja pametne kuće te IoT tehnologija, kratak uvod u umjetnu inteligenciju i njezina implementacija kod pametnih kuća. Na kraju će biti opisan izrađeni model pametne kuće sa njegovim hardverskim i softverskim komponentama, njegove funkcije i značajke te primjena umjetne inteligencije na sustav.

2. PAMETNA KUĆA

Pametna kuća odnosi se na kuću koja koristi mobilne uređaje povezane s internetom kako bi omogućila daljinski nadzor te pametno upravljanje uređajima i sustavima. Internet omogućuje povezivanje i komunikaciju uređaja u pametnoj kući. Omogućava korisniku kontrolu osvjetljenja, regulaciju temperature, upravljanje sigurnosnim sustavima te kontrolu drugih funkcija putem daljinske kontrole.

2.1. Način rada pametne kuće

Uređaji pametne kuće međusobno su povezani i može im se pristupiti putem jedne središnje točke, npr. putem pametnog telefona, tableta, prijenosnog računala ili panela na dodir. Brave na vratima, televizori, termostati, kućni monitori, kamere, svjetla, pa čak i uređaj kao što je hladnjak mogu se kontrolirati putem jednog sustava kućne automatizacije. Sustav se instalira na mobilni ili drugi umreženi uređaj. [1]

Pametni kućanski uređaji dolaze sa funkcijama samoučenja kako bi mogli naučiti navike i rasporede korisnika i izvršiti prilagodbe prema potrebi. Pametne kuće s kontrolom rasvjete omogućuju vlasnicima da smanje potrošnju električne energije. Neki sustavi kućne automatizacije upozoravaju korisnike kuća ako se otkrije bilo kakvo kretanje u kući dok su odsutni, dok drugi imaju sposobnost da automatski nazovu nadređene službe u slučajevima požara ili provale. Nakon povezivanja, pametni sigurnosni sustavi i pametni uređaji dio su IoT tehnologije. [1]

Sustavi koji tvore pametnu kuću mogu se povezivati žičano ili bežično. Bežične sustave lakše je instalirati te je njihova implementacija jeftinija i jednostavnija. Žičani sustavi smatraju se pouzdanijima i obično ih je teže „hakirati“. Implementacija ovih sustava je puno skuplja u odnosu na bežične.

Najpopularniji sustav automatizacije kuće temelji se na centraliziranom modelu. Takav model omogućava upravljanje iz centralnog procesorskog sustava svim ostalim uređajima. Baza ovakvog sustava je uređaj koji predstavlja mrežni pristupnik na kojemu je instaliran upravljački program te on služi kao centralna jedinica upravljanja. Tom upravljačkom programu moguće je pristupiti pomoću pametnog telefona, tableta ili računala. Pametni uređaji se žičano ili bežično povezuju na centralnu jedinicu putem nekog od protokola kao što su *WiFi*, *ZigBee*, *Z-Wave*, *BLE* i *Matter*.

2.2. Prednosti i nedostaci pametnih kuća

- Prednosti

Praktičnost: Pametne kuće su praktičnije od konvencionalnih. Svi sustavi mogu se automatizirati i integrirati kako bi zadovoljili specifične potrebe korisnika, bilo da se radi o pojačavanju termostata prije nego što korisnik dođe kući navečer ili sinkronizaciji medijskih uređaja u raznim prostorijama.

Ušteda energije: Pametne kuće ostvaruju uštedu energije. Moguće je programirati kontrole klime i rasvjete prema želji korisnika. Pametni sustavi prskalica i senzora moguće je integrirati kako bi se smanjilo rasipanje vode te je moguće alarmiranje korisnika u slučaju curenja vode.

Sigurnost: Postoji mnoštvo unutarnjih i vanjskih sigurnosnih rješenja koja se mogu nadzirati na daljinu, putem zaštitarske tvrtke ili lokalne policije i vatrogasca. Pametne kuće imaju napredne kućne sigurnosne sustave s raznim kamerama, senzorima pokreta i drugim naprednim značajkama.

Pristupačnost: Pametne kuće mogu ponuditi odlično rješenje za osobe s invaliditetom ili starije osobe budući da mogu biti opremljene značajkama pristupačnosti kao što su sustavi glasovnih naredbi koji mogu zaključati vanjska vrata, kontrolirati svjetla i čak upravljati računalom.

Povećana vrijednost doma: kuća sa pametnim značajkama može postići višu tržišnu vrijednost. Pametna sigurnost, inteligentna kontrola temperature, pametni uređaji za zabavu i uređaji bijele tehnike mogu drastično povećati cijenu kuće

Poticaji osiguranja: Neki osiguravatelji smanjuju svoje premije za pametne kuće pošto pametni sustavi mogu minimizirati nastalu štetu.

- Nedostaci

Početni troškovi: Pametna kućna tehnologija zahtijeva veća početna ulaganja od obične kućne tehnologije.

Prenaponi struje: Potrebno je osigurati da je dom adekvatno zaštićen od prenapona struje.

Kompatibilnost: Industrija pametne tehnologije je konkurentna, a na tržištu postoji mnogo opcija. Postoji nekoliko proizvođača pametne kućne tehnologije i nisu svi njihovi proizvodi kompatibilni. Važno je istražiti i odlučiti koji ekosustav najbolje odgovara potrebama korisnika.

Ranjivost na prekide rada s internetom: Ako korisnikov internet padne, većina pametnih sustava i uređaja ima neki oblik sigurnosne kopije koja im omogućuje nastavak rada. Međutim, korisnik se može izložiti riziku da izgubi neke od naprednijih funkcija njegovih pametnih uređaja dok se veza ne uspostavi.

2.3. Internet stvari

Izraz Internet stvari (IoT) općenito opisuje tehnologije u kojima senzori, aktuatori i svakodnevni uređaji dobivaju računalnu moć i mrežnu povezivost te im to omogućuje da generiraju, razmjenjuju i troše podatke te međusobno komuniciraju uz minimalnu ljudsku intervenciju. Međutim, ne postoji jedinstvena, univerzalna definicija Internet stvari. Ideja povezivanje objekta međusobno i s internetom nije nova, no spoj nekoliko tehnoloških i tržišnih trendova omogućuje jeftino i jednostavno međusobno povezivanje većeg broja manjih uređaja. [2]

Mrežna povezanost postala je jeftina, brza i sveprisutna te čini gotovo sve „povezanim“ dok je *Internet Protocol* (IP) komunikacijski protokol postao dominantni svjetski standard za mrežno povezivanje te pruža dobro razrađenu platformu alata i softvera koji se mogu jeftino i jednostavno implementirati u širok raspon uređaja. Razvitkom računala, dolazi do veće snage procesora po nižim cijenama i uz manju potrošnju energije. Napredak u proizvodnji omogućuje ugradnju vrhunske komunikacijske i računalne tehnologije u veoma male objekte što je potaknulo napredak jeftinih i malih uređaja za senzoriranje. Napredak u analizi podataka, brzo povećanje računalne snage, pohrane podataka i usluga u oblaku te novi algoritmi omogućuju prikupljanje, korelaciju i analizu golemih količina podataka. Računalstvo u oblaku iskorištava računalne resurse koji su umreženi i udaljeni za upravljanje, pohranu i obradu podataka te omogućuje distribuiranim i malim uređajima komunikaciju sa snažnim pozadinskim kontrolnim i analitičkim mogućnostima. [2]

Iot ima i svoju lošu stranu. IoT se može promatrati kao svijet nadzora, kršenja privatnosti i vezanosti potrošača. Dolazi se do problema kao što je hakiranje osobnih automobila spojenih na internet, nadzor koji proizlazi iz značajki prepoznavanja glasa u pametnim televizorima i strahovi vezani uz privatnost koji proizlaze iz potencijalne zlouporabe IoT podataka. [2]

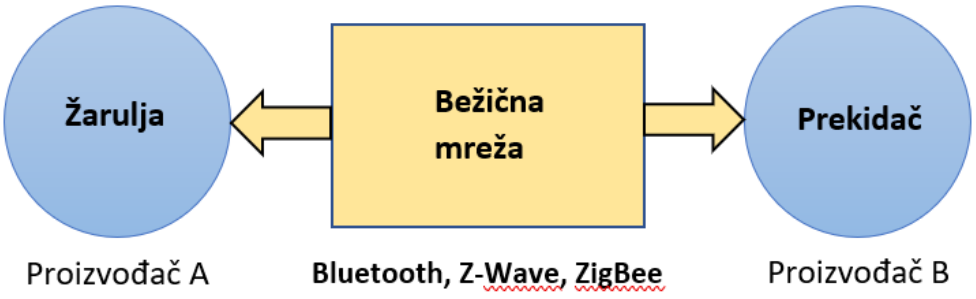
2.4. IoT komunikacijski modeli

- Komunikacija uređaj-uređaj

Komunikacijski model uređaj-uređaj (eng. *Device to Device Communication*) predstavlja direktnu komunikaciju dva ili više međusobno povezanih uređaja bez pomoći posrednika, tj. aplikacijskog poslužitelja. Najčešće se komunikacija kod ovih uređaja provodi pomoću interneta i IP mreža. Međutim, ovi uređaji često koriste komunikacijske protokole kao što su *Z-Wave, Bluetooth* ili *ZigBee* za uspostavljanje direktne komunikacije između uređaja, kao što je prikazano na slici 2.1. [3]

Ovaj komunikacijski model omogućuje uređajima koji koriste određeni komunikacijski protokol razmjenu podataka i komunikaciju kako bi ostvarili svoju funkciju. Ovaj komunikacijski model najviše se koristi upravo kod aplikacija automatizacije kuće koje obično koriste male podatkovne pakete za komunikaciju između uređaja s niskim zahtjevima za prijenos podataka. [3]

Često komunikacijski protokoli između uređaja nisu kompatibilni, što korisnika tjera da odabere obitelj uređaja koji koriste zajednički protokol. Na primjer, obitelj uređaja koji koriste *Z-Wave* protokol nije izvorno kompatibilna sa *ZigBee* obitelji uređaja. [3]



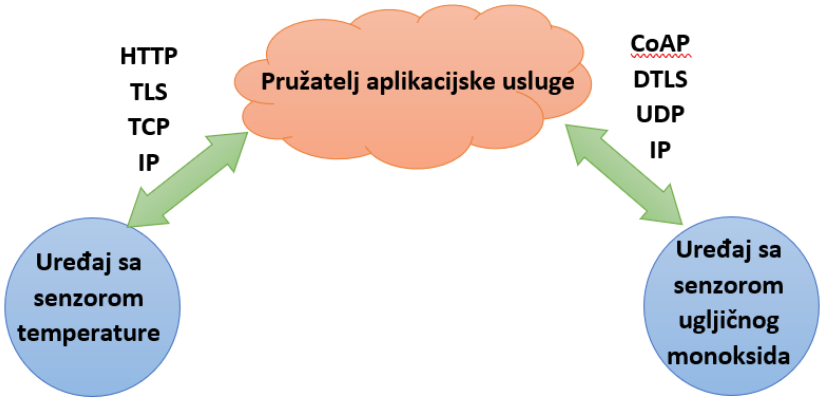
Slika 2.1. *Komunikacijski model uređaj-uređaj*

- Komunikacija uređaj-oblak

U modelu komunikacije između uređaja i oblaka (eng. *Device to Cloud Communication*), IoT uređaj povezuje se direktno s internetskom uslugom u oblaku koji se ponaša poput pružatelja aplikacijske usluge za kontrolu prometa poruka i razmjenu podataka. Ovaj komunikacijski model

koristi prednosti postojećih komunikacijskih mehanizama poput Wi-Fi veza ili tradicionalnih žičanih Ethernet veza za uspostavljanje veze između određenog uređaja i IP mreže, koja se u konačnici povezuje s uslugom u oblaku. [3]

Međutim, izazovi interoperabilnosti mogu nastati kada se pokušavaju integrirati uređaji različitih proizvođača. Često su uređaj i usluga u oblaku od istog dobavljača. Ako su vlasnički podaci protokoli koji se koriste između uređaja i usluge u oblaku, vlasnik ili korisnik uređaja mogu biti vezani za određenu uslugu u oblaku, ograničavajući ili sprječavajući korištenje alternativnih pružatelja usluga. [3]

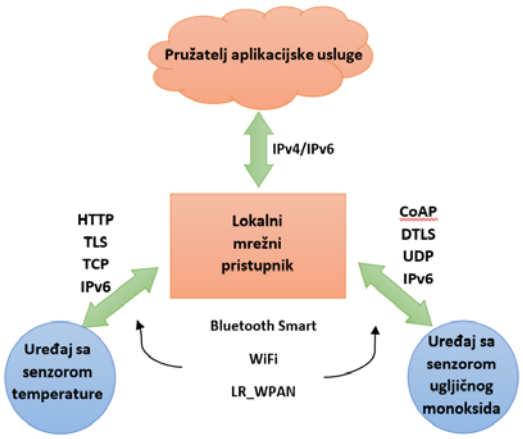


Slika 2.2. *Komunikacijski model uređaj-oblak*

- Model uređaj-mrežni pristupnik

Aplikacijski softver instaliran na lokalnom pristupnom uređaju, ponaša se kao veza između uređaja i usluge u oblaku te pruža funkcije kao što je prijevod podataka ili protokola te povećava sigurnost sustava. [3]

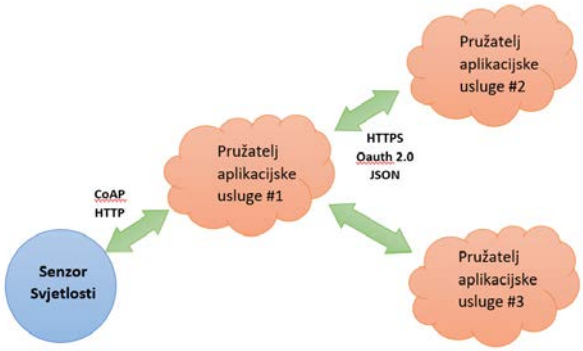
Drugi oblik modela, uređaj-mrežni pristupnik (eng. *Device to Gateway Model*), je pojava "hub" uređaja u aplikacijama kod pametnih kuća. To su uređaji koji služe za povezivanje IoT uređaja sa lokalnom mrežom i omogućuje pristup uslugama na oblaku, ali također mogu povezivati uređaje sa različitim protokolima. Ovaj se komunikacijski model često koristi za integraciju novih pametnih uređaja u postojeći sustav s uređajima koji nisu izvorno kompatibilni s njima. Loša strana ovog modela je da nužan razvoj softvera i sustava pristupnika aplikacijskog sloja dodaje složenost i cijenu cjelokupnom sustavu. [3]



Slika 2.3. *Komunikacijski model uređaj-mrežni pristupnik*

- Model pozadinskog dijeljenja podataka

Pozadinski model dijeljenja podataka (eng. *Back-End Data-Sharing Model*) odnosi se na komunikacijski model koji korisnicima omogućuje analizu i izvoz podataka pametnih uređaja iz usluge u oblaku u kombinaciji s podacima iz drugih izvora. Ovaj je pristup proširenje komunikacijskog modela između uređaja i oblaka. Pozadinska arhitektura dijeljenja omogućuje prikupljanje i analizu podataka prikupljenih iz pojedinačnih tokova podataka IoT uređaja. [3]



Slika 2.4. *Komunikacijski model pozadinskog dijeljenja podataka*

2.5. Često korišteni komunikacijski protokoli kod pametnih kuća

- WiFi

WiFi (eng. *Wireless Fidelity*) nudi najlakši i vjerojatno najrobusniji komunikacijski protokol kod automatizacije kuća radi svoje univerzalnosti i učestale upotrebe. WiFi predstavlja standardnu lokalnu mrežu kod većina kuća što pojeftinjuje i olakšava implementaciju pametnih kućnih uređaja temeljenih na WiFi-ju. Za aplikacije koje zahtijevaju veliku brzinu prijenosa podataka, WiFi predstavlja veoma prikladan izbor, dok njegova arhitektura bazirana na IP-u čini implementaciju aplikacija temeljenih na IoT-u relativno lakšom i jednostavnom u usporedbi s drugim protokolima. Sve ove značajke dolaze sa cijenom koja se očitava u visokoj potrošnji energije, kratkom dometu i visokom osjetljivošću na smetnje. [4]

- Zigbee

Veoma popularan komunikacijski protokol za izradu pametne kuće. Zigbee je otvoren i fleksibilan komunikacijski protokol (topologija isprepletene mreže) razvijen na frekvencijskom pojasu od 2,4 GHz. Radi svoje male snage i potrošnje veoma je prikladan kod pametnih kuća kojima upravljaju uređaji na baterije. Zigbee uređaji ne temelje se na IP-u te kao takvi zahtijevaju mrežni pristupnik za povezivanje s internetom. Zigbee nudi nisku brzinu prijenosa te postoji opasnost od smetnji uslijed interferencije na frekvencijskom pojasu od 2,4 GHz. [4]

- Z-Wave

Z-Wave je široko korišteni komunikacijski protokol usmjeren na malu potrošnju energije koji je razvio Z-wave savez. Radi u mesh mrežnoj arhitekturi sličnoj kao i Zigbee, no u pojasu od 908 MHz. Nudi nisku latenciju i niske smetnje, ali ima znatno manju brzinu prijenosa u usporedbi s drugim protokolima. [4]

- BLE

Razvijen od strane posebne interesne skupine za Bluetooth, BLE (eng. *Bluetooth Low Energy*) je dorada klasičnog Bluetooth protokola sa nižom potrošnjom energije te se zato naziva Bluetooth Smart. Prednost ovog protokola je smanjena potrošnja energije pri prijenosu podataka i u stanju mirovanja te jednostavna implementacija i niska latencija. Nedostatak mu je mali domet. [4]

- Matter

Razvijen od strane Connectivity Standard Alliance (prethodno Zigbee savez) i podržan od strane velikih proizvođača rješenja za pametnu kuću uključujući; Apple, Amazon i Google. To je protokol temeljen na IP-u koji je dizajniran za rješavanje izazova interoperabilnosti rješenja za pametnu kuću, pružajući okvir koji omogućuje komunikaciju između uređaja, aplikacija i usluga u oblaku. Matterova arhitektura pruža jedinstveni protokol za stvaranje eko sustava pametne kuće koji podržava različite protokole. [4]

3. UMJETNA INTELIGENCIJA

Pojam umjetna inteligencija (UI, prema eng. akronimu AI, od *Artificial Intelligence*) odnosi se na dio računalne znanosti (informatike) koji se bavi stvaranjem sustava koji obavljaju zadatke za koje je potreban nekakav oblik inteligencije. Neke funkcije UI sustava uključuju planiranje, učenje, rasuđivanje, rješavanje problema i donošenje odluka. Inteligentni sustavi su tehnološki napredni sustavi koji percipiraju i reagiraju na svijet oko sebe te uče iz iskustva te se mogu prilagođavati. Funkcije inteligentnoga sustava jesu: komunikacija s čovjekom ili s drugim inteligentnim sustavima, zaključivanje, planiranje te prikupljanje i obrada informacija i znanja. [5]

Postoje 3 vrste inteligencije: slaba UI, jaka UI te umjetna super inteligencija. Slaba UI jedina je vrsta umjetne inteligencije koja je do danas uspješno realizirana. Slaba umjetna inteligencija dizajnirana je za obavljanje pojedinačnih zadataka, npr. prepoznavanje lica, prepoznavanje govora/glasovni asistenti, vožnja automobila ili pretraživanje interneta. Jaka UI predstavlja koncept stroja s općom inteligencijom koji oponaša ljudsku inteligenciju i/ili ponašanje, sa sposobnošću učenja i primjene svoje inteligencije za rješavanje bilo kojeg problema. Jaka UI može razmišljati, razumjeti i djelovati na način koji se ne razlikuje od ljudskog u bilo kojoj situaciji. Umjetna super inteligencija je hipotetska umjetna inteligencija koja ne samo da oponaša ili razumije ljudsku inteligenciju i ponašanje već sustavi ili strojevi postaju svjesni sebe i nadilaze kapacitet ljudske inteligencije i sposobnosti. [6]

Sustavi temeljeni na umjetnoj inteligenciji brzo se razvijaju u smislu primjene, prilagodbe, brzine obrade i mogućnosti. Strojevi sve više postaju sposobni preuzeti manje rutinske zadatke. Dok ljudska inteligencija zapravo 'donosi' savršenu odluku u odgovarajuće vrijeme, UI se samo bavi 'odabirom' prave odluke u odgovarajućem trenutku. UI nedostaje kreativnosti u odluci koju ljudi mogu donijeti. Različite domene kao što su filozofija, računalna znanost, matematika, statistika, biologija, fizika, sociologija, psihologija i mnoge druge spojile su se kako bi potaknule interdisciplinarnu prirodu umjetne inteligencije. [7]

3.1. Grane umjetne inteligencije

Postoji širok skup tehnika koje dolaze iz domene umjetne inteligencije kao što su pristranost, lingvistika, planiranje, vizija, robotska automatizacija procesa, znanost odlučivanja, obrada prirodnog jezika, itd. U nastavku je opisano šest glavnih grana umjetne inteligencije.

3.1.1. Strojno učenje

Jedno od najzahtjevnijih tehnika umjetne inteligencije. Strojno učenje (eng. *Machine learning*) je potpodručje umjetne inteligencije koje omogućuje računalima učenja bez eksplicitnog programiranja. Metoda strojnog učenja koristi se za obavljanje složenih zadataka na način koji je sličan načinu na koji ljudi rješavaju probleme. [8]

Strojno učenje počinje s podacima: brojevima, fotografijama ili tekstom, kao što su bankovne transakcije, slike, zapisi o popravcima, vremenski niz podataka iz senzora ili izvješća o prodaji. Podaci se prikupljaju i pripremaju za korištenje kao podaci za obuku ili informacije na kojima će se model strojnog učenja trenirati. Što više podataka postoji, to je program bolji. [8]

Zatim programeri odabiru model strojnog učenja koji će koristiti, dostavljaju podatke i puštaju računalni model da se sam osposobi za pronalaženje obrazaca ili predviđanja. S vremenom ljudski programer također može prilagoditi model, uključujući promjenu njegovih parametara, kako bi ga pogurao prema točnijim rezultatima. [8]

Neki se podaci izvlače iz podataka za treniranje koji se koriste kao podaci za testiranje, čime se testira koliko je točan model strojnog učenja kada mu se prikažu novi podaci. Rezultat je model koji se može koristiti u budućnosti s različitim skupovima podataka. [8]

Postoje tri potkategorije strojnog učenja:

Modeli učenja sa učiteljem (eng. *supervised learning*) obučavaju se s označenim skupovima podataka, koji omogućuju modelima da uče i postaju točniji tijekom vremena. Učitelj posjeduje znanje o okolini u obliku parova ulaznih i izlaznih podataka. Tijekom učenja dolazi do pogrešaka koje predstavljaju razliku između optimalnog i dobivenog odziva za neki ulazni vektor. Tokom učenja parametri mreže se mijenjaju pod utjecajem ulaznih vektora i signala pogreške. Taj proces se ponavlja sve dok model ne nauči imitirati učitelja. Na primjer, algoritam se trenira sa slikama

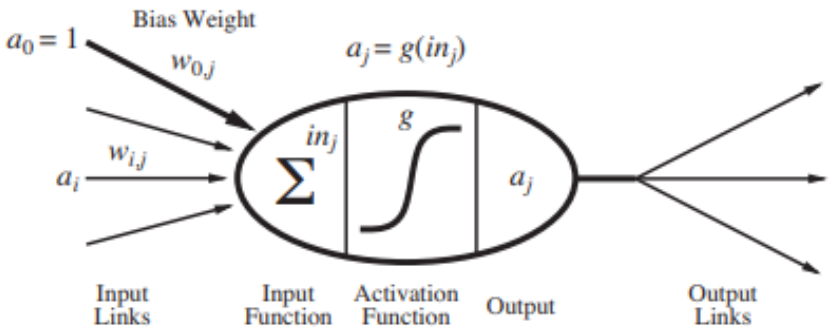
broda u nekakvom okruženju. Programeri označuju dijelove slike koje predstavljaju brod, a stroj traži načine da sam identificira slike broda. Učenje sa učiteljem najčešći je tip strojnog učenja koji se danas koristi. [8]

U modelima učenja bez učitelja (eng. *unsupervised learning*), program traži uzorke u neoznačenim podacima. Model učenja bez učitelja može pronaći obrasce ili trendove koje ljudi izričito ne traže. Na primjer, program strojnog učenja bez učitelja mogao bi pregledati podatke o online prodaji i identificirati različite vrste klijenata koji kupuju. [8]

Ojačano učenje (eng. *reinforcement learning*) trenira strojeve putem metode pokušaja i pogrešaka da poduzmu najbolju akciju uspostavljanjem sustava nagrađivanja. Učitelj ne daje povratnu informaciju u obliku veličine pogreške nego samo govori koliko je određeni korak u učenju dobar. Ojačano učenje može osposobiti modele za igranje igrica ili osposobiti autonomna vozila za vožnju govoreći stroju kada je donio ispravne odluke, što mu pomaže da s vremenom nauči koje radnje treba poduzimati. [8]

3.1.2. Neuronske mreže

Potaknuti hipotezom da se mentalna aktivnost prvenstveno sastoji od elektrokemijske aktivnosti u mrežama moždanih stanica koje se nazivaju neuroni, neki od najranijih radova umjetne inteligencije usmjereni su na stvaranje umjetnih neuronskih mreža. Slika 3.1 prikazuje jednostavan matematički model neurona koji su osmislili MCCulloch i Pits (1943.). Grubo govoreći, neuron se „aktivira“ kada linearna kombinacija njegovih ulaza premaši neki (tvrdi ili meki) prag, tj. implementira linearni klasifikator. Neuronska mreža je skup međusobno povezanih neurona te su njezina svojstva određena njezinom topologijom i svojstvima „neurona“. [9]



Slika 3.1. *Matematički model neurona*

Neuronske mreže sastoje se od skupa neurona koji sadrže otežane ulaze, sumator, aktivacijsku funkciju, prag i izlaz. Veza a_i služi za širenje izlaznog signala iz neurona *i* prema neuronu *j*. Svaka veza također ima pridruženu numeričku težinu $w_{i,j}$, koja određuje snagu i predznak veze. Baš kao u modelima linearne regresije, svaka jedinica ima jedan lažni ulaz $a_0 = 1$ s pridruženom težinom $w_{0,j}$. Svaki neuron najprije računa sumu otežanih ulaza prema formuli (3.1):

$$in_j = \sum_{i=0}^{n} w_{i,j} a_i \quad (3.1),$$

gdje je:

a_i veza između neurona *i* i neurona *j*,

$w_{i,j}$ težina ulaza a_i,

n broj veza,

in_j suma otežanih ulaza.

Zatim primjenjuje aktivacijsku funkciju g na prethodnu sumu tako da se dobije izlaz (3.2):

$$a_j = g(in_j) = g\left(\sum_{i=0}^{n} \omega_{i,j} a_i\right) \quad (3.2),$$

gdje je:

a_j izlaz iz neurona j,

g() aktivacijska funkcija.

Aktivacijska funkcija *g* obično je ili tipa prag, te se tada neuron naziva perceptron, ili logistička funkcija te se u tom slučaju neuron naziva sigmoidalan perceptron. Obje ove nelinearne aktivacijske funkcije osiguravaju važno svojstvo da cijela neuronska mreža može predstavljati nelinearnu funkciju. [9]

Postoje dva fundamentalna načina povezivanja neurona u mrežu. Jednosmjerne mreže bez povratne veze (eng. *feed-forward network*) imaju vezu u samo jednom smjeru, tj. ona formira usmjereni aciklički graf. Svaki neuron prima ulaz od "uzvodnih" neurona i isporučuje izlaz "nizvodnim" neuronima, tj. nema petlji. Jednosmjerna mreža predstavlja funkciju svog trenutnog ulaza, dakle ne postoje unutarnja stanja osim težina. Mreža s povratnom vezom (eng. *recurrent network*) vraća svoje izlaze natrag na vlastite ulaze. To znači da mreža u kombinaciji sa

elementima za kašnjenje tvori dinamički sustav koji može doseći stabilno stanje ili pokazivati oscilacije ili čak kaotično ponašanje. Štoviše, odgovor mreže na određeni ulaz ovisi o njegovom početnom stanju, koje može ovisiti o prethodnim ulazima. Mreža ima sposobnost kratkoročnog pamćenja. [9]

3.1.3. Robotika

Roboti su fizički agenti koji obavljaju zadatke manipulirajući fizičkim svijetom. Da bi to učinili, opremljeni su efektorima kao što su noge, kotači, zglobovi i hvataljke. Efektori imaju samo jednu svrhu: djelovati fizičkim silama na okolinu. Roboti su također opremljeni senzorima koji im omogućuju da percipiraju svoje okruženje. Današnja robotika koristi raznolik skup senzora, uključujući kamere i lasere za mjerenje okoliša, te žiroskope i akcelerometre za mjerenje kretanja robota.

Umjetna inteligencija može pomoći robotu u obavljanju mnogih zadataka, od uspješnog snalaženja u okolini, do identificiranja objekata oko robota ili pomaganja ljudima s raznim zadacima kao što su kretanje kroz prostor ili operacije uz pomoć robota.

U nastavku su navedene neke od aplikacija umjetne inteligencije i strojnog učenja u robotici.

UI i tehnologije računalnog vida mogu pomoći robotima da identificiraju i prepoznaju objekte s kojima se susreću, pomoći u odabiru detalja u objektima i pomoći u navigaciji i izbjegavanju prepreka. [10]

Manipulacija i hvatanje potpomognuto umjetnom inteligencijom se dugo smatralo teškim zadatkom za robote. UI se koristi za pomoć robotima u hvatanju predmeta. Uz pomoć umjetne inteligencije, robot može ispružiti ruku i uhvatiti predmet bez potrebe za ljudskim upravljačem. [10]

Kroz poboljšane mogućnosti strojnog učenja, roboti dobivaju veću autonomiju, smanjujući potrebu ljudi za planiranjem i upravljanjem navigacijskim stazama i tokovima procesa. Strojno učenje i umjetna inteligencija pomažu robotu da analizira svoje okruženje i pomažu u vođenju njegovog kretanja, što robotu omogućuje izbjegavanje prepreka ili, u slučaju softverskih procesa, automatsko manevriranje oko iznimaka procesa ili uskih grla u toku. [10]

Da bi roboti imali određenu razinu autonomije, često moraju biti u stanju razumjeti svijet oko sebe. To razumijevanje dolazi od prepoznavanja omogućenog UI-om i obrade prirodnog jezika. Strojno učenje pokazalo je značajnu sposobnost pomoći strojevima da razumiju podatke i identificiraju obrasce kako bi mogli djelovati prema potrebi. [10]

3.1.4. Ekspertni sustavi

Ekspertni sustav odnosi se na računalni sustav koji oponaša inteligenciju donošenja odluka ljudskog stručnjaka. To provodi izvođenjem znanja iz svoje baze znanja implementacijom rezoniranja i pravila uvida u smislu korisničkih upita. Učinkovitost ekspertnog sustava u potpunosti se oslanja na ekspertovo znanje akumulirano u bazi znanja. Što je više informacija prikupljeno u njemu, sustav više povećava svoju učinkovitost. Na primjer, ekspertni sustav daje prijedloge za pravopis i pogreške u Google tražilici. Ekspertni sustavi izgrađeni su da se bave složenim problemima putem rasuđivanja kroz tijela stručnosti, posebno izražena pravilima "if-then" umjesto tradicionalnog programa kodiranja. Ključne značajke ekspertnih sustava uključuju izuzetno brz odziv, pouzdanost, razumljivost i visoku izvedbu. [11]

3.1.5. Neizrazita logika

Neizrazita logika (eng. *fuzzy logic*) je tehnika koja predstavlja i modificira nesigurne informacije mjerenjem stupnja do kojeg je hipoteza točna. Neizrazita logika također se koristi za razmišljanje o prirodno nesigurnim konceptima. Neizrazita logika je prikladna i fleksibilna za implementaciju tehnika strojnog učenja i pomoć u logičkom oponašanju ljudske misli. [11]

Neizrazito upravljanje je metodologija za konstruiranje sustava upravljanja u kojima je preslikavanje između ulaznih i izlaznih parametara predstavljeno neizrazitim pravilima. Neizrazita kontrola je vrlo uspješna u komercijalnim proizvodima kao što su automatski mjenjači, video kamere i električni brijači. Ove aplikacije su uspješne jer imaju male baze pravila, nemaju lančane zaključke i podesive parametre koji se mogu prilagoditi kako bi se poboljšala izvedba sustava. [9]

Neizrazita logika temelji se na teoriji neizrazitih skupova. Kod neizrazitih skupova nema jasnih granica između istine i laži te između pripadnosti nekog elementa nekom skupu. Karakteristična funkcija može poprimiti bilo koju vrijednost iz intervala [0,1]. [9]

Neizrazit skup se definira jednadžbom:

$$F = \{x, \mu_F(x)\} \quad (3.3),$$

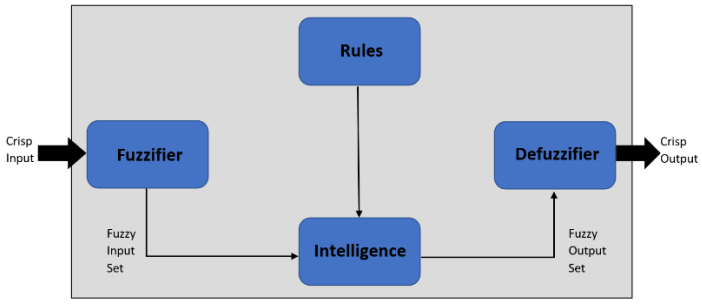
gdje je:

F neizrazit skup, definiran skupom uređenih parova elemenata *x* i stupnja pripadnosti $\mu_F(x)$

x element čiji se stupanj pripadnosti skupu utvrđuje

$\mu_F(x)$ funkcija pripadnosti skupu, poprima vrijednosti iz intervala [0,1]

Na slici 3.2 prikazan je fuzzy logički sustav. Modul fuzzifikacije (eng. *Fuzzyfication module*) transformira skup ulaznih podataka (eng. *crisp input*) u fuzzy skup podataka. Baza podataka (eng. *Knowledge base*) sadrži if-then pravila koje pišu eksperti, tj. programeri. Sustav zaključivanja (eng. *Inference Engine*) simulira proces zaključivanja kao kod ljudi primjenom „if-then" pravila na fuzzy skupu podataka. Modul defuzifikacije (eng. *Defuzzification module*) transformira fuzzy skup podataka u izlazni skup podataka (eng. *crisp output*). [9]



Slika 3.2. *Fuzzy logički sustav*

3.1.6. Obrada prirodnog jezika

Obrada prirodnog jezika (eng. *Natural Language Processing*-NLP) je dio računalne znanosti i umjetne inteligencije koji može pomoći u komunikaciji između računala i čovjeka prirodnim jezikom. To je tehnika računalne obrade ljudskog jezika. Omogućuje računalu čitanje i razumijevanje podataka oponašanjem ljudskog prirodnog jezika. [11]

Implementacija NLP-a daje razne prednosti kao što su;

- poboljšava točnost i učinkovitost dokumenata,
- ima mogućnost izrade automatiziranog sažetka zadanog teksta,
- koristi se kod osobnih asistenta kao što je Alexa,

- organizacijama omogućuje implementaciju chatbota za korisničku podršku,
- olakšava analizu raspoloženja.

Neke od NLP aplikacija su prijevod teksta, analiza osjećaja i prepoznavanje govora, tumačenje recenzija kupaca i poboljšanje njihovog iskustva te ostali. [11]

3.2. Primjena umjetne inteligencije kod pametnih kuća

Posljednjih godina umjetna inteligencija postala je važan dio pametnih kuća diljem svijeta. Tehnologija umjetne inteligencije dodala je dodatnu razinu sigurnosti, a istovremeno omogućila ljudima da uživaju u komfornijem načinu života. Zbog povećane potražnje za pametnim kućama, alati za kućnu automatizaciju sada su pristupačniji, a pametni kućanski uređaji sve „pametniji“. [12]

- UI za sigurnost doma

Jedna od najpopularnijih upotreba umjetne inteligencije pri projektiranju domova je njezina sposobnost poboljšanja sigurnosti doma. Tehnologija je ugrađena u uređaje opremljene raznim značajkama, uključujući prepoznavanje lica i analiza prijetnji. Sustav može prepoznati objekte i lica te poslati obavijesti o osobama koje stoje na ulaznim vratima. Prepoznavanje lica omogućuje naprednim AI uređajima da prepoznaju lica članova obitelji, prijatelja, pa čak i kućnih ljubimaca. Štoviše, umjetna inteligencija također je korištena za izradu pametnih brava koje se aktiviraju putem mobilnih uređaja. [12]

Dimni alarmi s umjetnom inteligencijom već su dostupni na tržištu. Opremljeni pametnim značajkama, mogu govoriti, upozoravati, pa čak i razmišljati sami, a sve kako bi spriječili požare i curenje ugljičnog monoksida. Pametni dimni alarmi komuniciraju s vlasnicima putem mobilnih aplikacija instaliranih na njihovim pametnim telefonima. Mogu slati različite obavijesti o razini baterije, dimu, curenju ugljičnog monoksida, pa čak i odrediti lokacije na kojima bi moglo biti požara ili dima. [12]

- UI za automatizirane aktivnosti u kućanstvu

Pametne kuće mogu pomoći ljudima da uštede vrijeme i trud i usredotoče se na stvari koje su im važne. Uređaji s umjetnom inteligencijom mogu se pobrinuti za pokretanje robotskog usisavača, oprati i osušiti odjeću, prilagoditi unutarnju temperaturu prema preferencijama ljudi prisutnih u kući, puštati glazbu, podešavati temperature automobila i automatski paliti i gasiti svjetla. [12]

- UI za pametne kuhinje

Pametne kuhinje pomažu ljudima da izbjegnu nezgode povezane s predugim ili nedovoljno kuhanjem hrane. Pametne pećnice i štednjaci sada su opremljeni AI sustavima koji mogu procijeniti temperaturu hrane koja se kuha i smanjiti je ako se pokaže da je visoka. Štoviše, aplikacije mogu obavijestiti korisnike kada je hranu potrebno izvaditi iz pećnice ili štednjaka. Može čak pokrenuti proces predgrijavanja prije nego što korisnik dođe kući. [12]

- UI za skladištenje električne energije

Umjetna inteligencija ima ulogu u distribuiranoj proizvodnji energije. To je zato što integrira automatske sustave za pohranu energije s distribuiranim izvorima energije za pohranjivanje i korištenje električne energije. Štoviše, zbog sposobnosti tehnologije da pohranjuje električnu energiju u sustave za pohranu energije, pametne kuće mogu imati koristi od energije čak i ako dođe do nestanka struje. Drugim riječima, zahvaljujući pohrani energije koju omogućuje UI tehnologija, pametni će se domovi lakše osamostaliti od električne mreže i uštedjeti na računima za struju. [12]

- UI za glasovnu kontrolu

Alexa, Siri i Google Assistant su virtualni glasovni asistenti s omogućenom umjetnom inteligencijom koji se koriste za upravljanje raznim sustavima pametnog doma putem glasovnih naredbi. Svaki put kada jedan od glasovnih asistenta s UI-om napravi pogrešku, naučit će iz toga. Glasovni asistent uzima podatke povezane s upitom, uči iz pogreške i pokušava dati povoljan odgovor. Sve se na vrijeme pohranjuje, analizira i poboljšava. Glasovni asistent oslanja se na podatke i strojno učenje kako bi pomogao u kontroli pametnog doma. [12]

4. KORIŠTENI SOFTVER

4.1. Home Assistant

Za potrebe izrade ovog zadatka korišten je program Home Assistant (HA). HA je kao Python aplikaciju započeo Paulus Schoutsen u rujnu 2013. i prvi put je javno objavio na internet servisu GitHub-u u studenom 2013. Home Assistant je univerzalni softver za kućnu automatizaciju, besplatan je, otvorenog koda te je dizajniran da bude centralni sustav upravljanja u pametnoj kući. HA omogućuje kontrolu i pristup pametnim kućnim uređajima na lokalnoj mreži. Po želji, omogućuje dizajniranje kućne automatizacije bez potrebe za povezivanjem s uslugama na oblaku tako da se kućni uređaji ne moraju izlagati internetu gdje ih se može pratiti ili ugroziti njihovo korištenje. HA pruža radnje i skripte temeljene na pravilima za stvaranje automatizacije, raspoređivanje zadataka, obavijesti i glasovnu kontrolu, vrijeme i upravljanje uvjetima događaja. Također pruža funkcije za izravne akcije i radnje na zahtjev. Radi popularnosti programskog jezika Python i velike baze korisnika, sve više programera doprinosi razvijanju ovog softvera.

HA omogućuje integraciju postojećih uređaja. Trenutačno HA podržava više od 1800 integracija iz širokog spektra kategorija. Njegovo svojstvo interoperabilnosti može povezivati više uređaja različitih proizvođača, npr. može se kontrolirati Philips Hue Zigbee svjetla putem Z-Wave zidnog prekidača. Home Assistant sposoban je komunicirati sa svakom platformom i identificira ih kao svjetla. Neke od spomenutih integracija koje se koriste u ovome radu navedene su u nastavku.

Home Assistant ne samo da integrira i kontrolira uređaje nego i omogućuje izradu automatizacija. Daje mogućnost paljenja i gašenja žarulje koja podržava Wi-Fi pomoću Zigbee žarulje. Može poslati obavijest korisniku laptopa ako je baterija njegovog Android telefona pri kraju. Može prigušiti IKEA žarulje dok korisnik gleda film. Potonji primjer pokazuje da HA može biti korisniji od programa zatvorenog koda jer se ne integrira samo s fizičkim uređajima, već i s brojnim aplikacijama i uslugama.

4.2. ESPHome

Jedno od integracija koje se koriste u HA je ESPHome integracija. ESPHome je alat u vlasništvu tvrtke Nabu Casa s kojim se može kreirati prilagođeni firmware za lako dostupne i jeftine ESP8266 i ESP32 mikročipove. Nisu potrebne nikakve programerske vještine da bi se započelo s ESPHome-om. Postoji službena integracija ugrađena u Home Assistant, a ESPHome uređaji, također poznati kao čvorovi, na mreži automatski bivaju otkriveni. ESPHome je alat koji čita konfiguracijsku datoteku pisanu u YAML-u[1] i stvara prilagođenu binarnu datoteku firmwarea. Svaki čvor, to jest ono što se naziva pojedinačnim ESPHome uređajem, ima svoju individualnu konfiguracijsku datoteku. [13]

ESPHome podržava velik broj senzora i aktuatora koji se mogu konfigurirati sa samo nekoliko redaka YAML-a. Jedina stvar na koju je potrebno obratiti pažnju je na koji je pin senzor ili aktuator spojen. Ne samo da se podaci sa senzora mogu prikupljati, ESPHome također može proslijediti podatke priključenim komponentama. ESPHome također može pokretati različite zaslone, u rasponu od točkastih matrica do zaslona e-papira. [13]

Slika 4.1. *Konfiguracijske datoteke čvorova u integraciji ESPHome*

4.3. Samba share

Program Samba share pruža usluge dijeljenja i ispisa datoteka za računala na mreži. Koristi protokol *Server Message Block* i *Common Internet File System* (SMB/CIFS), tako da su usluge stvorene pokretanjem Sambe dostupne na Linuxu, macOS-u i Windows klijentima.

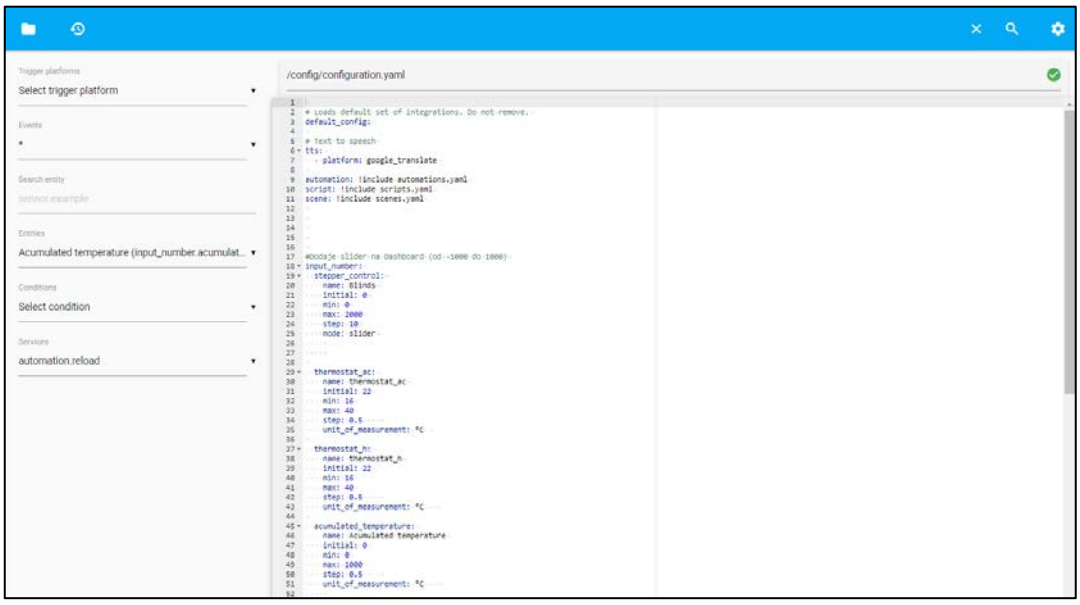
[1] YAML je jezik za serijalizaciju podataka koji se često koristi za stvaranje konfiguracijskih datoteka s bilo kojim programskim jezikom.

4.4. File Editor

Uređivač datoteka (*eng. File Editor*), prije poznat kao konfigurator (eng. *configurator*), mala je web-aplikacija kojoj se pristupa pomoću Home Assistant-a te koja nudi preglednik datoteka i uređivač teksta za izmjenu datoteka na računalu na kojem je pokrenut Uređivač datoteka i Home Assistant. Pokreće ga Ace Editor, koji podržava označavanje sintakse za različite kodne/označne jezike. YAML datoteke automatski se provjeravaju radi pogrešaka u sintaksi tijekom uređivanja. [14]

Značajke integracije:

- web-uređivač za izmjenu datoteka s isticanjem sintakse i YAML lintingom[2],
- prijenos i preuzimanje datoteka,
- popisi s dostupnim entitetima, okidačima, događajima, uvjetima i uslugama,
- ponovno pokretanje Home Assistant-a izravno jednim pritiskom na gumb i ponovno učitavanje grupa, automatizacija, itd.,
- izravne veze na dokumentaciju i ikone Home Assistant-a,
- izvršavanje naredbe ljuske unutar spremnika dodataka,
- postavke uređivača spremaju se u preglednik. [14]



Slika 4.2. *Sučelje integracije File Editor*

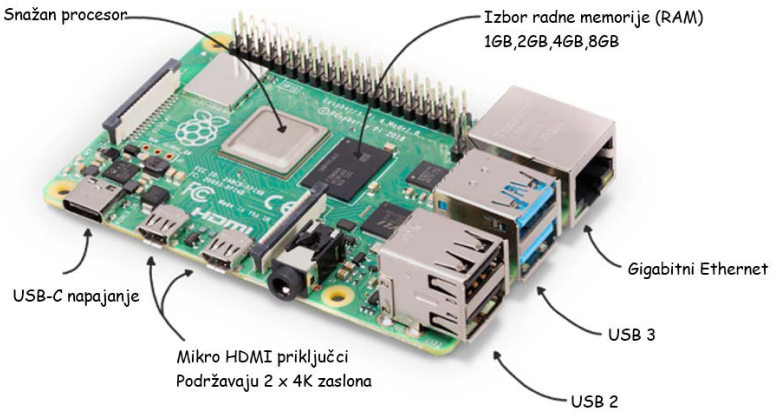
[2] Linting je automatizirana provjera izvornog koda za programske i stilske pogreške.

5. KORIŠTENI HARDVER

5.1. Raspberry Pi

Raspberry Pi je ime za seriju cjenovno pristupačnih mikroračunala koje je razvila tvrtka Raspberry Pi Foundation u Ujedinjenom Kraljevstvu. Cilj razvijanja Raspberry Pi mikroračunala je promicanje podučavanja osnova informatike, programiranja i robotike. Raspberry Pi je vrlo jeftino računalo koje pokreće Linux, ali također nudi skup GPIO (eng. *general purpose input output*) pinova, što omogućuje kontrolu elektroničkih komponenti za fizičko računalstvo i istraživanje Interneta stvari (IoT). Raspberry Pi se radi svoje niske cijene, modularnosti i dizajna otvorenog koda (eng. *open-source*) široko koristi u mnogim područjima. Omogućuje programiranje u programskim jezicima kao što su Python i Scratch. Obično ga koriste kompjuterski i elektronički hobisti pri izradi raznih projekata. [15]

Raspberry pi 4 Model B je najnoviji i najmoćniji model mikroračunala marke Raspberry. Raspberry Pi 4 Model B ima nadograđeni CPU i GPU. CPU ima nešto veću brzinu takta (1,5 GHz) i ima noviju, učinkovitiju arhitekturu od one koja se nalazi u starijim modelima. Isto vrijedi i za GPU, koji može reproducirati 4K/60FPS H.265-kodirani video na 2 zaslona. Dolazi u verzijama od 1 do 8 GB RAM-a te pruža mogućnosti poput Bluetooth 5,0 / BLE i USB 3.0 povezivanja te True Gigabit Ethernet komunikacije.



Slika 5.1. *Raspberry Pi 4 Model B*

5.2. ESP8266

ESP8266 WiFi modul je samostalni SOC (*System on chip*) s integriranim TCP/IP protokolom koji može svakom mikrokontroleru dati pristup zadanoj WiFi mreži. Wi-Fi modul ESP8266 se uglavnom koristi za razvoj krajnjih IoT aplikacija. Naziva se samostalnim bežičnim primopredajnikom te je dostupan po vrlo niskoj cijeni. Koristi se za omogućavanje internetske veze raznim aplikacijama ugrađenih sustava. Može raditi ili kao podređena ili kao samostalna aplikacija. Ako ESP8266 Wi-Fi radi kao podređeni uređaj za host mikrokontrolera, tada se može koristiti kao Wi-Fi adapter za bilo koju vrstu mikrokontrolera koristeći UART (eng. *universal asynchronous receiver-transmitter*) ili SPI (eng. *serial peripheral interface*). Ako se modul koristi kao samostalna aplikacija, tada osigurava funkcije mikrokontrolera i Wi-Fi mreže. Skup AT naredbi potreban je mikrokontroleru za komunikaciju s ESP8266 Wi-Fi modulom. Stoga je razvijen sa softverom za AT naredbe kako bi omogućio Arduino Wi-Fi funkcionalnosti, a također omogućuje učitavanje različitog softvera za dizajn vlastite aplikacije na memoriji i procesoru modula. [16]

Wi-Fi modul ESP8266 dolazi s ROM-om za pokretanje od 64 KB, RAM-om za korisničke podatke od 80 KB i RAM-om za upute od 32 KB. Može podržati 802.11 b/g/n Wi-Fi mrežu na 2,4 GHz zajedno sa značajkama I2C, SPI, I2C sučelja s DMA i 10-bitnim ADC-om. Povezivanje ovog modula s mikrokontrolerom može se jednostavno izvršiti putem serijskog priključka. Vanjski pretvarač napona potreban je samo ako radni napon prelazi 3,6 volta. Najviše se koristi u robotici i IoT aplikacijama zbog niske cijene i kompaktne veličine. Postoje različite vrste ESP modula koje su dizajnirali različiti proizvođači koji se razlikuju po broju GPIO pinova. [16]

U ovome radu koristi se NodeMCU DEVKIT 1.0 (prikazan na slici 5.2) kojeg pokreće procesor ESP8266.

Slika 5.2. *Razvojna pločica NodeMCU DEVKIT 1.0*

5.3. Senzor temperature i vlage DHT 11

Jednostavan, spor i jeftin, no dobar za osnovno bilježenje podataka. Senzor se sastoji od dva djela, kapacitivnog senzora vlage i termistora. Unutar senzora se nalazi i jednostavan čip koji vrši analogno-digitalnu pretvorbu i emitira digitalni signal s temperaturom i vlagom koji se kasnije čita pomoću mikrokontrolera.

Specifikacije senzora DHT11:

- vrlo niska cijena,
- napajanje: 3 do 5V,
- maksimalna potrošnja struje: 2,5mA,
- raspon mjerenja vlage: 20-80% vlažnosti s 5% točnosti,
- raspon mjerenja temperature: 0-50°C s ±2°C točnost,
- najveća brzina uzrokovanja: 1Hz,
- veličina tijela: 15,5 mm x 12 mm x 5,5 mm.

Slika 5.3. *Senzor temperature i vlage DHT 11*

5.4. Senzor pokreta HC-SR501

HC-SR501 je inačica PIR (*eng. Passive Infrared*) senzora koji služi za detektiranje pokreta tijela čovjeka ili životinje na temelju infracrvenog zračenja topline. Ovaj senzor može se pronaći u većini modernih sigurnosnih sustava, automatskih prekidača za svjetlo, otvarača garažnih vrata i sličnih aplikacija kamo se na temelju pokreta aktivira akcija ili događaj.

Specifikacije senzora HC-SR501:

- napon: 4.8V - 20V,
- struja: 50µA,
- kut i udaljenost detekcije: 120°, 7m,
- logički izlaz: 3.3V,
- dimenzije: 32mm x 24mm.

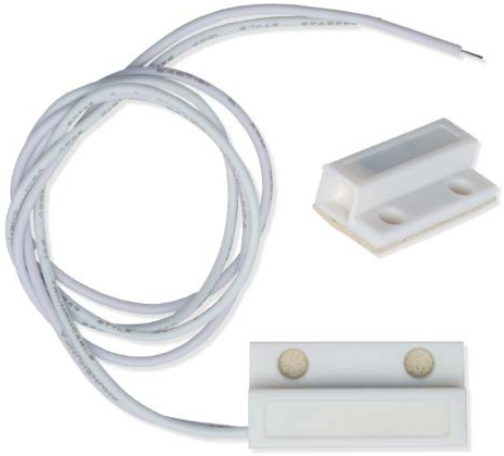
Slika 5.4. *PIR senzor HC-SR501*

5.5. Reed relej s magnetom

Ovaj prekidač za vrata ima ugrađen reed relej koji se može uključiti magnetom. Reed relej uspostavlja kontakt (zatvorena veza) ako se magnet približi. Ovaj set se sastoji od dva djela: reed releja sa žicama i magneta. Oba su ugrađena u plastično kućište koje se može zalijepiti ili zašarafiti na postolje.

Tehnički podaci:

- maksimalni napon: 110VDC,
- maksimalna struja: 100mA,
- životni vijek: 100 milijuna puta prebacivanja,
- udaljenost prebacivanja: 18 mm ± 6 mm,
- dimenzije: 27 x 14 x 10 mm.



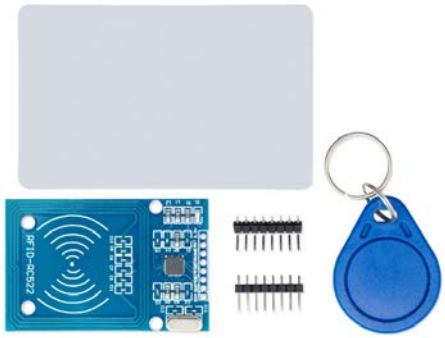
Slika 5.5. *Reed relej s magnetom*

5.6. RFID čitač RC522

RC522 RFID modul jedna je od najjeftinijih RFID (*eng. Radio frequency identification*) opcija. Dolazi s RFID karticom i RFID privjeskom s 1 KB memorije. Modul RFID čitača RC522 dizajniran je za stvaranje elektromagnetskog polja od 13,56 MHz i komunikaciju s RFID oznakama. Čitač može komunicirati s mikrokontrolerom preko 4-pinskog SPI-a s maksimalnom brzinom prijenosa podataka od 10 Mbps. Također podržava komunikaciju preko I2C i UART protokola.

Specifikacije:

- frekvencijski raspon: 13,56 MHz ISM pojas,
- host sučelje: SPI / I2C / UART,
- radni napon napajanja: 2,5 V do 3,3 V,
- maksimalna radna struja: 13-26mA,
- minimalna struja (isključivanje): 10μA.

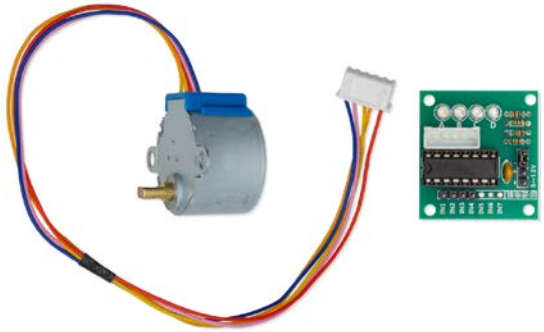


Slika 5.6. *RFID čitač RC522 sa karticom i privjeskom*

5.7. Koračni motor 28BYJ-48 sa driverom ULN2003

28BYJ-48 je peterožilni unipolarni koračni motor koji radi na naponu od 5V. Motor isporučuje okretni moment od 34,3 mN*m pri brzini od oko 15 okretaja u minuti. Daje dobar okretni moment i u stanju mirovanja koji se održava sve dok se motor napaja strujom. Prema tablici s podacima, kada motor 28BYJ-48 radi u punom koraku, svaki korak odgovara rotaciji od 11,25°. To znači da postoje 32 koraka po okretaju. Motor ima set reduktora 1/64 te mu je radna struja 240 mA.

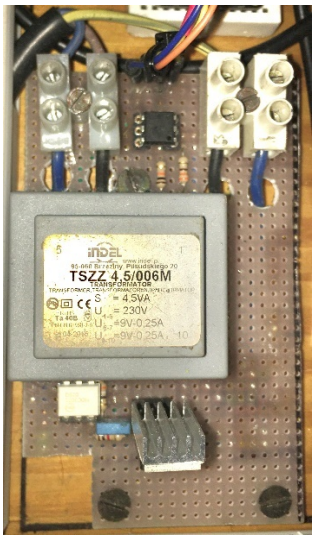
Za upravljanje motorom potreban je upravljački sklop ULN2003. ULN2003 sastoji se od niza od sedam Darlingtonovih parova tranzistora, od kojih svaki par može pokretati opterećenje do 500 mA i 50 V. Četiri od sedam parova koriste se na ovoj ploči.



Slika 5.7. *Koračni motor 28BYJ-48 sa driverom ULN2003*

5.8. Izmjenični regulator napona

Izmjenični regulator napona služi za kontrolu jačine svjetlosti žarulje. Radi na principu rezanja napona i struje napajanja žarulje. Sastoji se od detektora prolaska napona kroz nulu i strujnog kruga za upravljanje kutom okidanja trijaka. Kada sklop primijeti prolazak napona kroz nulu ovisno o željenoj jačini svjetlosti okida se trijak i napaja žarulja. U posljednje vrijeme ovaj regulator postao je često korištena solucija za sustave pametne kuće. Na primjer, kada je potrebno glatko promijeniti svjetlinu svjetla, žarulja se polako pali ili gasi, stvarajući ugodnu atmosferu.



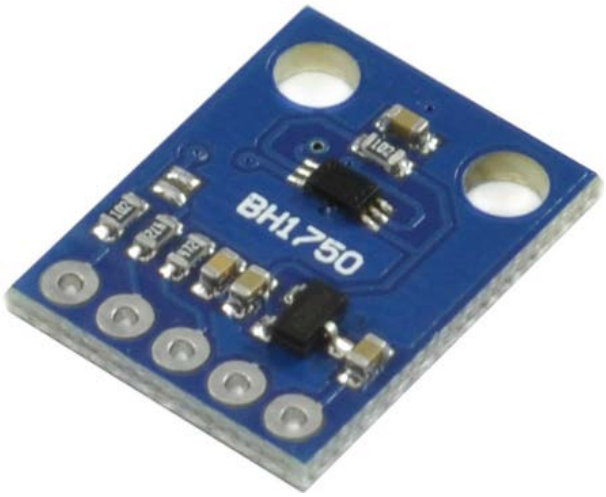
Slika 5.8. *Izmjenični regulator napona*

5.9. Senzor svjetlosti BH1750

BH1750 je digitalni senzor ambijentalnog svjetla koji se obično koristi u mobilnim telefonima za upravljanje svjetlinom zaslona na temelju osvjetljenja okoline. Ovaj senzor može precizno mjeriti LUX vrijednost svjetlosti od 0 do 65535lx.

Specifikacije:

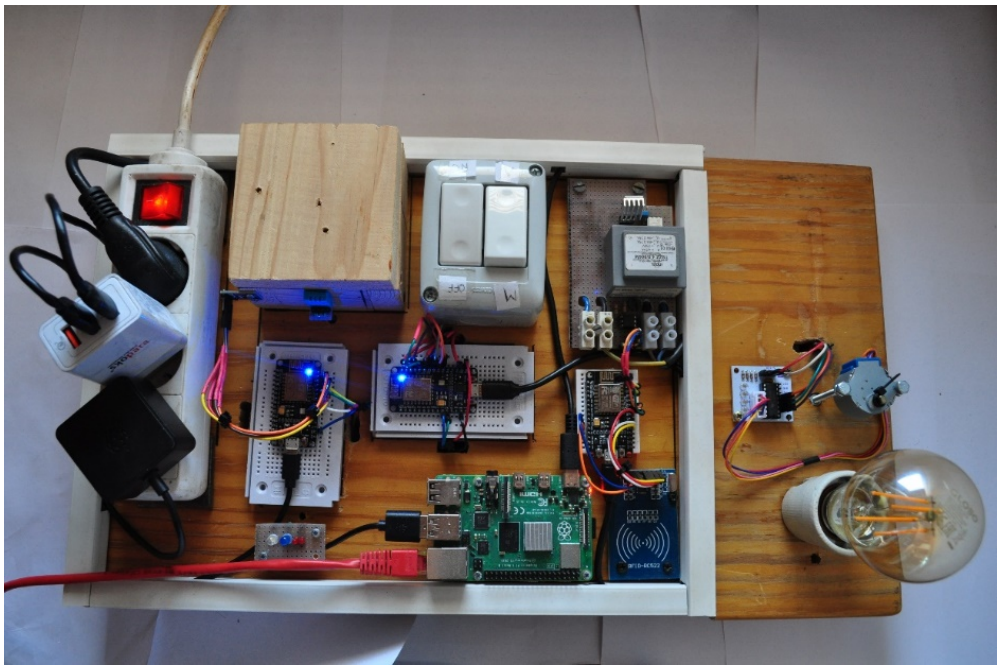
- napajanje: 2,4V - 3,6V (obično 3V),
- potrošnja struje: 0.12mA,
- mjerni raspon: 1-65535lx,
- komunikacija: I2C sabirnica,
- točnost: +/-20%,
- ugrađeni A/D pretvarač za pretvaranje analognog osvjetljenja u digitalne podatke,



Slika 5.9. *Senzor svjetlosti BH1750*

6. IZRADA I POVEZIVANJE HARDVERSKOG DJELA

Postolje ovog modela izrađeno je od daske veličine 25x46 cm na koju su postavljeni sljedeći elementi: grlo sa žaruljom, koračni motor 28BYJ-48 sa driverom ULN2003, izmjenični regulator napona, RC522 RFID čitač, tri WiFi modula NodeMCU DEVKIT 1.0 ESP8266 nazvani ESP0, ESP1, ESP2, Raspberry pi 4 Model B, izmjenični i serijski prekidač, senzor pokreta HC-SR501, reed relej s magnetom, senzor temperature i vlage DHT 11, senzor svjetlosti BH1750, 3 LED diode (crvena, plava, bijela), produžni kabel sa tri utora, napajanje za Raspberry, napajanje za WiFi module te žice i kablovi koji služe za spajanje spomenutih dijelova.

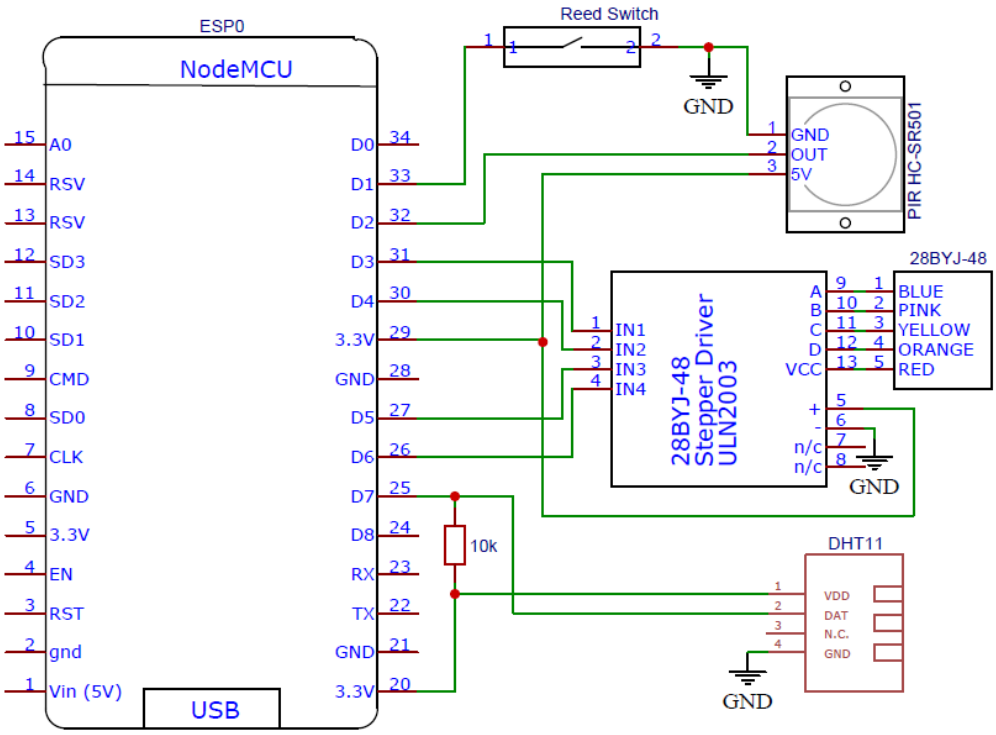


Slika 6.1. *Izgled gotovog modela pametne kuće*

- Povezivanje komponenti s WiFi modulom ESP0

Na slici 6.1. prikazana je shema spajanja WiFi modula ESP0 sa sljedećim komponentama:

- senzor pokreta HC-SR501,
- koračni motor 28BYJ-48 sa driverom ULN2003,
- senzor temperature i vlage DHT 11,
- reed relej s magnetom.

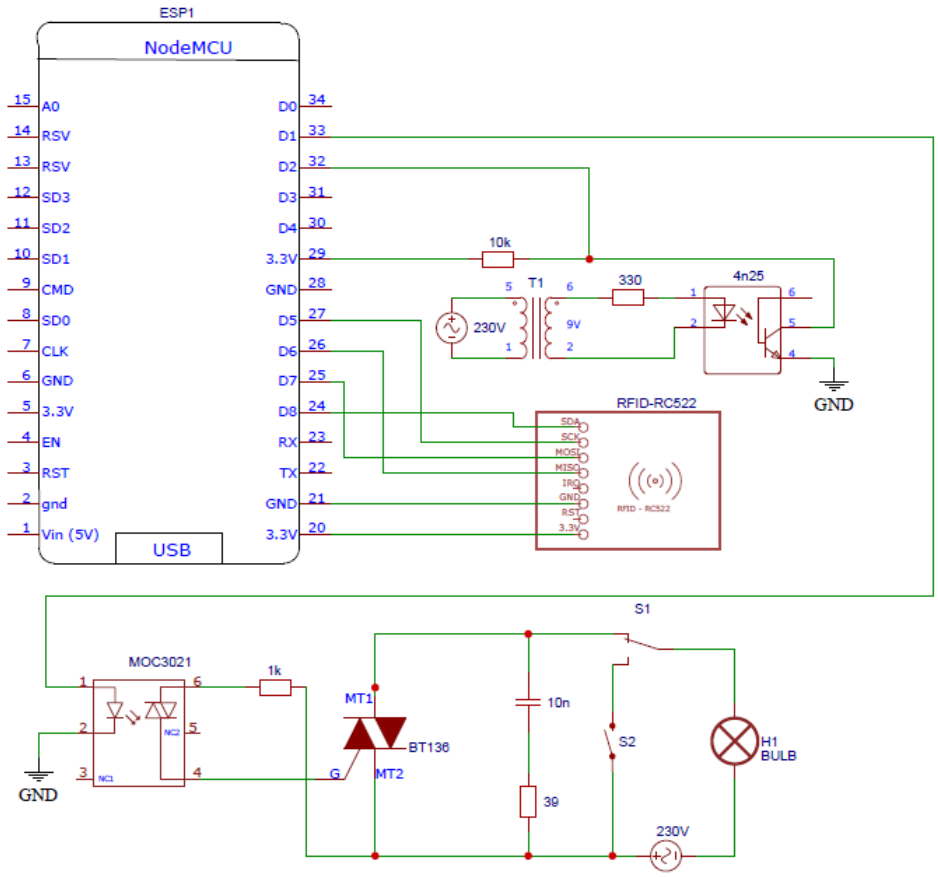


Slika 6.2. *Shema spajanja WiFi modula ESP0 sa komponentama*

- Povezivanje komponenti s WiFi modulom ESP1

Na slici 6.2. prikazana je shema spajanja WiFi modula ESP1 sa sljedećim komponentama:

- RFID čitač RC522,
- izmjenični regulator napona.

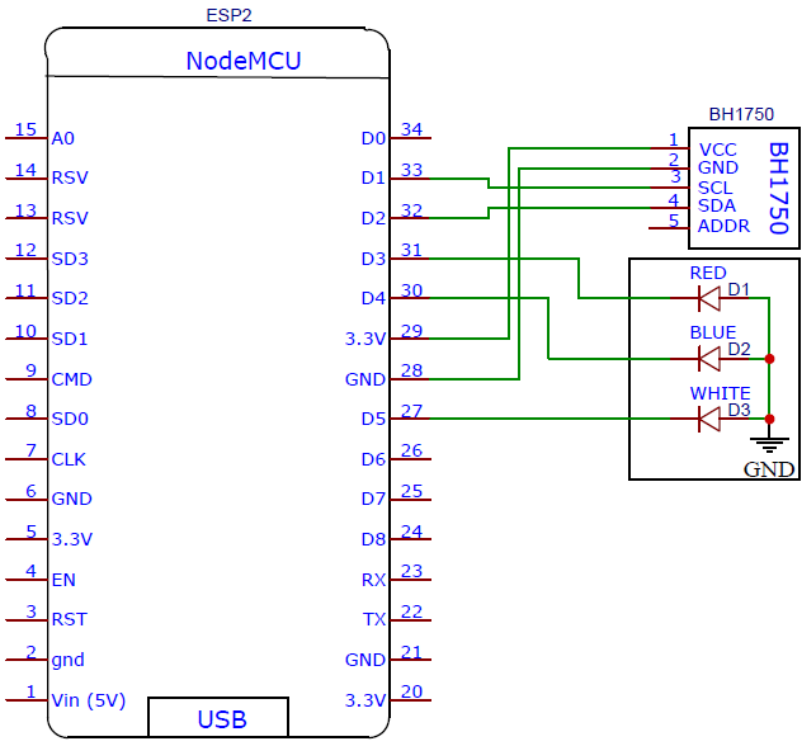


Slika 6.3. *Shema spajanja WiFi modula ESP1 sa komponentama*

- Povezivanje komponenti s WiFi modulom ESP2

Na slici 6.3. prikazana je shema spajanja WiFi modula ESP2 sa sljedećim komponentama:

- Senzor svjetlosti BH1750,
- 3 LED diode.



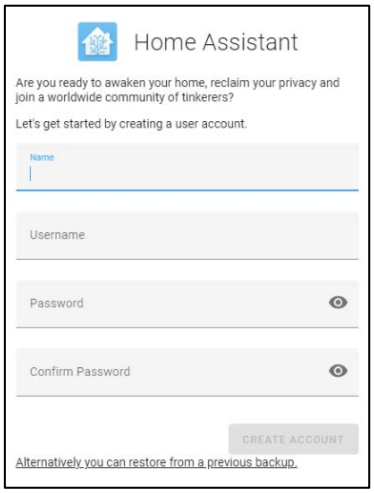
Slika 6.4. *Shema spajanja WiFi modula ESP2 sa komponentama*

7. REALIZACIJA SOFTVERSKOG DJELA

Prvi korak ka realizaciji softverskog djela ovoga rada je instaliranje programa Home Assistant na mikrokontroler Raspberry Pi 4 Model B. Ovaj mikrokontroler nema sekundarnu unutarnju memoriju te mu je potrebna vanjska. Za to se koristi mikro SD kartica kapaciteta 64GB. Pomoću laptopa i alata Raspberry Pi Imager na Raspberry Pi se instalira operacijski sustav (OS) na jednostavan način. Ovaj program preuzima odabrani OS, dopušta njegovu konfiguraciju, formatira SD karticu i zatim upisuje OS na nju. Na slici 7.1. prikazan je alat Raspberry Pi Imager.

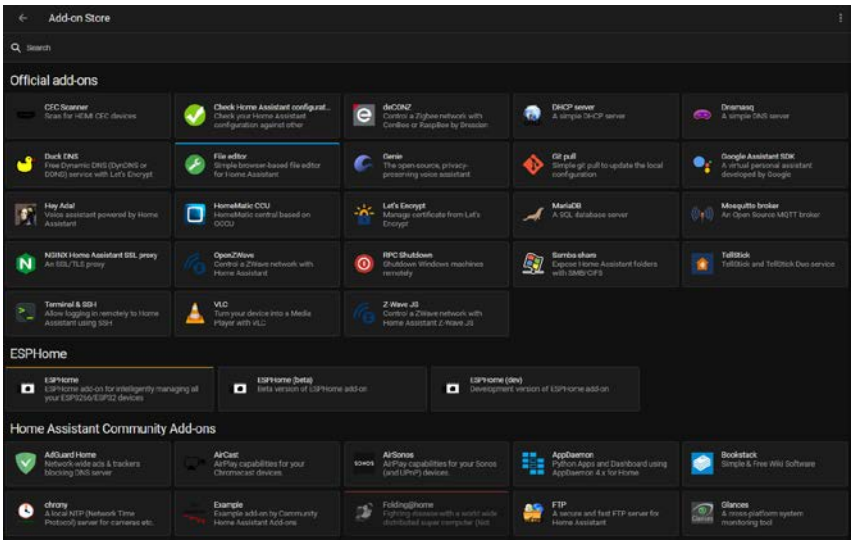
Slika 7.1. *Programski alat Raspberry Pi Imager*

Nakon instalacije OS-a, SD kartica se umeće u Raspberry. Raspberry se uključuje i povezuje ethernet kabelom sa računalom ili ruterom. Upisivanjem lokalne IP adrese u internet preglednik računala otvara se sučelje programa Home Assistant te započinje inicijalizacija aplikacije. Zatim se stvara korisnički račun kao što je prikazano na slici 7.2.



Slika 7.2. *Stvaranje korisničkog računa u Home Assistant-u*

Na sljedećem ekranu HA će prikazati sve uređaje koji su otkriveni na lokalnoj mreži. Nakon stvaranja korisničkog računa Home Assistant je spreman za rad te korisnika dovodi u web sučelje na kojem su prepoznati uređaji i entiteti. Važan korak je dodavanje integracija. U trgovini integracija (Slika 7.3.) dodavaju se i instaliraju prije spomenute integracije, ESPHome, File Explorer i Samba share. Integracija ESPHome je potrebna za upravljanje WiFi modulima i definiranje senzora spojenih na module dok File Explorer nudi uvid u YAML skripte koje se mogu uređivati i mijenjati. Samba share služi za pristup datotekama pomoću programa na računalu.



Slika 7.3. *Trgovina integracija*

U YAML datoteke u integraciji ESPHome upisuju se naredbe kojima definiramo tip i ime mikrokontrolera, postavke povezivanja sa lokalnom mrežom te definiramo ulaze i izlaze u mikrokontroler. Ulazi i izlazi mogu biti senzori, aktuatori, binarni sezori, imaginarne sklopke i drugi.

Sa prvim mikrokontrolerom, ESP0, spojeni su senzor pokreta HC-SR501, koračni motor 28BYJ-48 sa driverom ULN2003, senzor temperature i vlage DHT 11 te reed relej s magnetom. U nastavku je prikazan programski kod (YAML datoteka) sa objašnjenjima ovog mikrokontrolera.

```
esphome:
  name: esp0         # Naziv mikrokontrolera

esp8266:
  board: nodemcuv2   # Tip mikrokontrolera


logger:              # Omogućavanje logera
```

```
# Enable Home Assistant API
api:                                                    # Omogućavanje API sučelja
  encryption:
    key: "Le00xy7hS0zPICTjbt9oQeEHBe/TkB5T8Qgr08wjIX4="    # Kod za enkripciju

  services:                                             # Popis korisnički definiranih usluga
    - service: control_stepper                          # Ime usluge Home Assistanta
      variables:                                        # Popis varijabli
        target: int                                     # Definiranje tipa varijable target
      then:
        - logger.log: "Speed updated"                   # Upisivanje u loger
        - stepper.set_target:                           # Postavljanje pozicije
            id: my_stepper                              # Id entiteta
            target: !lambda 'return target;'            # Vrijednost pozicije

ota:                                                    # ota (Over The Air) komponenta
  password: "fb9a33f038809e1804586230637f9bd2"          # Lozinka koja se koristi za ažuriranja

wifi:                                                   # Omogućavanje WiFi povezivanja
  ssid: "MW40V_4E48"                                    # Ime mreže
  password: "xxxx"                                      # Lozinka za povezivanje

  manual_ip:                                            # Omogućavanje manualne IP adrese
    static_ip: 192.168.1.134                            # IP adresa mikrokontrolera
    gateway: 192.168.1.1                                # IP adresa rutera
    subnet: 255.255.255.0                               # subnet mask adresa

  # Enable fallback hotspot (captive portal) in case wifi connection fails
  ap:                                           # Omogućavanje načina rada pristupne točke na čvoru
    ssid: "Esp0 Fallback Hotspot"               # Ime pristupne točke
    password: "FG3aVVJ3Mp5I"                    # Lozinka za povezivanje

captive_portal:                                 # Omogućavanje captive portal komponente

sensor:                                         # Vrsta entiteta
  - platform: dht                               # Vrsta platforme
    pin: D7                                     # Broj pina na mikrokontroleru
    temperature:
      name: "Temperature"                       # Naziv senzora temperature
    humidity:
      name: "Humidity"                          # Naziv senzora vlage
    update_interval: 15s                        # Interval za ažuriranje senzora

binary_sensor:                                  # Vrsta entiteta
  - platform: gpio                              # Vrsta platforme
    pin:
      number: GPIO05                            # Broj pina na mikrokontroleru
      mode:                                     # Svojstva pina
        input: true                             # Definiranje pina kao ulazni
        pullup: true                            # Omogućavanje pull up otpornika
      inverted: true                            # Invertiranje ulaza
    name: senzor_vrata                          # Naziv senzora

  - platform: gpio                              # Vrsta platforme
    pin: GPIO04                                 # Broj pina na mikrokontroleru
    name: "PIR Sensor"                          # Naziv senzora
    device_class: motion                        # Definiranje klase senzora
```

```
stepper:                                # Vrsta entiteta
  - platform: uln2003                   # Vrsta platforme
    sleep_when_done: true               # Mod bez napajanja kada se ne koristi
    id: my_stepper                      # id senzora
    pin_a: D3                           # Broj pina na mikrokontroleru
    pin_b: D4                           # Broj pina na mikrokontroleru
    pin_c: D5                           # Broj pina na mikrokontroleru
    pin_d: D6                           # Broj pina na mikrokontroleru
    max_speed: 250 steps/s              # Definiranje maksimalne brzine

    # Optional:
    acceleration: inf                   # Vrijednost akceleracije
    deceleration: inf                   # Vrijednost deceleracije
```

Sa drugim mikrokontrolerom, ESP1, spojeni su RFID čitač RC522 i izmjenični regulator napona. U nastavku je prikazan programski kod (YAML datoteka) ovog mikrokontrolera sa objašnjenjima.

```
esphome:
  name: esp1                                            # Naziv mikrokontrolera

esp8266:
  board: nodemcuv2                                      # Tip mikrokontrolera

# Enable logging
logger:                                                 # Omogućavanje logera

# Enable Home Assistant API
api:                                                    # Omogućavanje API sučelja
  encryption:
    key: "ZqHJnE+MzSief54s+TW8k3E+v1LICX55SeU0sP7TOvM=" # Kod za enkripciju

ota:                                                    # ota (Over The Air) komponenta
  password: "1fb43556198fb13b0c21b4358b7b8329"           # Lozinka koja se koristi za ažuriranja

wifi:                                                   # Omogućavanje WiFi povezivanja
  ssid: "MW40V_4E48"                                    # Ime pristupne točke
  password: "xxxx"                                      # Lozinka za povezivanje

manual_ip:                                              # Omogućavanje manualne IP adrese
    static_ip: 192.168.1.100                            # IP adresa mikrokontrolera
    gateway: 192.168.1.1                                # IP adresa mrežnog pristupnika
    subnet: 255.255.255.0                               # subnet mask adresa


  # Enable fallback hotspot (captive portal) in case wifi connection fails
  ap:                                                   # Omogućavanje načina rada pristupne točke na čvoru
      ssid: "Esp1 Fallback Hotspot"                     # Ime pristupne točke
      password: "fPdZpNcByKXV"                          # Lozinka za povezivanje

captive_portal:                                         # Omogućavanje captive portal komponente

spi:                                                    # Definiranje SPI protokola
  clk_pin: GPIO14                                       # Definiranje CLK pina
  mosi_pin: GPIO13                                      # Definiranje MOSI pina
  miso_pin: GPIO12                                      # Definiranje MISO pina
```

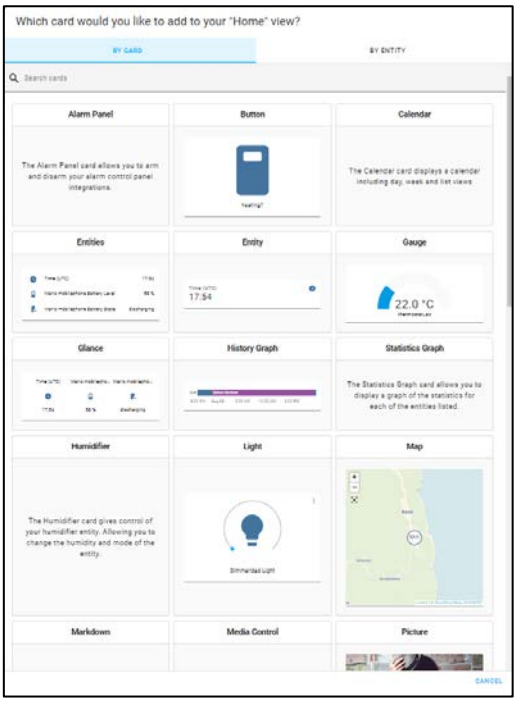
```
rc522_spi:                              # Vrsta entiteta
  cs_pin: GPIO15                        # Definiranje CS pina
  update_interval: 0.1s                 # Interval ažuriranja

binary_sensor:                          # Vrsta entiteta
  - platform: rc522                     # Vrsta platforme
    uid: AC-45-DC-38                    # Jedinstveni identifikator
    name: "Blue NFC Tag"                # Ime taga

  - platform: rc522                     # Vrsta platforme
    uid: 2A-71-24-A5                    # Jedinstveni identifikator
    name: "White NFC Tag"               # Ime taga

   output:
  - platform: ac_dimmer                 # Vrsta platforme
    id: dimmer1                         # id senzora
    gate_pin: GPIO05                    # Definiranje gate pina (upravljanje trijakom)
    zero_cross_pin:
      number: GPIO04                    # Definiranje zero_croos pina (detekcija nul napona)
      mode:
        input: true                     # Definiranje pina kao ulaz
      inverted: yes                     # Invertiranje ulaza
    min_power: 0.1                      # Definiranje minimalne snage
    max_power: 0.105                    # Definiranje maksimalne snage


light:
  - platform: monochromatic             # Vrsta platforme
    output: dimmer1                     # Definiranje izlaza
    name: Dimmerized Light              # Naziv entiteta
```

Sa trećim mikrokontrolerom, ESP1, spojeni su senzor svjetlosti BH1750 i 3 LED diode. U nastavku je prikazan programski kod (YAML datoteka) sa objašnjenjima ovog mikrokontrolera.

```
esphome:
  name: esp2        # Naziv mikrokontrolera
esp8266:
  board: nodemcuv2  # Tip mikrokontrolera
# Enable logging
logger:             # Omogućavanje logera
# Enable Home Assistant API

api:                                                    # Omogućavanje API sučelja
  encryption:
    key: "Jjr5yVZoKX3SUlWytEPqCqLLU/iTsKpBafTu3HmtCzo="    # Kod za enkripciju

ota:                                                    # ota (Over The Air) komponenta
  password: "0c5c3970ee64eee2b64ca75468955ced"          # Lozinka koja se koristi za ažuriranja

wifi:                                                   # Omogućavanje WiFi povezivanja
  ssid: "MW40V_4E48"                                    # Ime mreže
  password: "xxxx"                                      # Lozinka za povezivanje

manual_ip:                                              # Omogućavanje manualne IP adrese
    static_ip: 192.168.1.124                            # IP adresa mikrokontrolera
```

```
    gateway: 192.168.1.1                          # IP adresa rutera
    subnet: 255.255.255.0                         # subnet adresa


  # Enable fallback hotspot (captive portal) in case wifi connection fails
  ap:                                           # Omogućavanje načina rada pristupne točke na čvoru
    ssid: "Esp2 Fallback Hotspot"               # Ime pristupne točke
    password: "pBTGAijJLfij"                    # Lozinka za povezivanje

captive_portal:                                 # Omogućavanje captive portal komponente


switch:                                         # Vrsta entiteta
  - platform: gpio                              # Vrsta platforme
    pin: 0                                      # Broj pina na mikrokontroleru
    name: "heating1"                            # Naziv entiteta

  - platform: gpio                              # Vrsta platforme
    pin: 02                                     # Broj pina na mikrokontroleru
    name: "cooling1"                            # Naziv entiteta

  - platform: gpio                              # Vrsta platforme
    pin: 14                                     # Broj pina na mikrokontroleru
    name: "dehumidifier"                        # Naziv entiteta

i2c:
  sda: D2                                       # Definiranje SDA pina
  scl: D1                                       # Definiranje SCL pina
  scan: True

sensor:                                         # Vrsta entiteta
  - platform: bh1750                            # Vrsta platforme
    name: "BH1750 Illuminance"                  # Naziv senzora
    address: 0x23                               # I2C adresa senzora
    update_interval: 1s                         # Interval ažuriranja
    filters:
      - calibrate_linear:                       # Filter sa kalibraciju
        - 0.0 -> 0.0                            # Mapiranje vrijednosti
        - 1500 -> 100                           # Mapiranje vrijednosti
```

Nakon konfiguriranja ESP-ova u integraciji ESPHome dodavaju se senzori i aktuatori na nadzorne ploče u Home Assistant-u. Entiteti se dodavaju tako da se na određenoj nadzornoj ploči izvrši slijed akcija: **Open dashboard menu > Edit Dashboard > ADD CARD** te se zatim bira entitet ili kartica kojom se konfigurira izgled entiteta (Slika 7.4.).



Slika 7.4. *Dodavanje entiteta na nadzornu ploču*

7.1. Automatiziranje Home Assistant-a

Unutar Home Assistant-a nalaze se sve informacije o senzorima i servisima koji su integrirani unutar programa. Te informacije su dostupne korisniku na nadzornim pločama i mogu se koristiti kao okidači za pokretanje automatizacija. Automatizacije u Home Assistant-u omogućavaju automatsko pokretanje određene radnje na temelju druge radnje. To mogu biti očitanja ili promjena stanja senzora, određeno vrijeme, dostizanje određene vrijednosti entiteta te drugi. Blueprints automatizacije su unaprijed napravljene automatizacije od strane zajednice programera koje se mogu jednostavno dodati unutar programa Home Assistant.

Sve automatizacije su napravljene od tri glavna elementa: okidača, uvjeta i akcije. Okidači (eng. *trigger*) opisuju događaje koji bi trebali pokrenuti automatizaciju. Kada bilo koji od okidača automatizacije postane istinit (okidač se aktivira), Home Assistant će potvrditi uvjete, ako postoje, i pozvati akciju. Moguće je odrediti više okidača za jednu automatizaciju. Drugi dio automatizacije je uvjet (eng. *condition*). Uvjeti su izborni dio pravila automatizacije i mogu se koristiti za sprječavanje radnje da se dogodi kada se ne aktiviraju. Kada uvjet poprimi stanje „false“ automatizacija će se prestati izvršavati. Najvažniji dio automatizacije je akcija (eng. *action*). Akcija će se izvršiti kada se okidač pokrene i svi uvjeti budu ispunjeni. U nastavku je prikazan jedan jednostavan primjer automatizacije. Okidač ove automatizacije je promjena stanja entiteta:

„input_button.turn_on_light“ iz stanja „off“ u stanje „on“. Uvjet ove automatizacije je entitet: „switch.cooling1“ u stanju „off“, tj. treba biti isključena klimatizacija. Ispunjavanjem uvjeta i okidanjem okidača izvršava se akcija koja pali svjetlo sa svjetlinom 100, tj. entitet „light.dimerized_light“ prelazi u stanje „on“.

```
alias: Turn on light                                #ime automatizacije
description: ""
mode: single
trigger:                                            #okidač automatizacije
  - platform: state
    entity_id:
      - input_button.turn_on_light
    from: "off"
    to: "on"
condition:                                          #uvjet automatizacije
  - condition: device
    type: is_off
    device_id: 6404f5c9a7eaad05cfb9a78dda6492c4
    entity_id: switch.cooling1
    domain: switch
action:                                             #akcija automatizacije
  - type: turn_on
    device_id: f3ea0229be091a90be0dc50b7bc8bd2f
    entity_id: light.dimmerized_light
    domain: light
    brightness_pct: 100
```

7.1.1. Uređivač automatizacije

Uređivač automatizacije (eng. *Automation Editor)* jednostavan je način stvaranja i uređivanja automatizacija iz korisničkog sučelja. Nije potrebno pisati programski kod. Jednostavnim klikom na ikone odabiru se okidači, uvjeti i akcije koje program sam prevodi u programski jezik, tj. dodaje automatizaciju u YAML datoteku. Automatizacije u Uređivaču automatizacije stvaraju se tako da se ide u **Settings>Automation & Scenes>CREATE AUTOMATION>Start with an empty automation** te se zatim odabiru željene postavke.

Osim automatizacija, u sklopu programa Home Assistant postoji i komponenta za pisanje skripti (eng. *Scripts*). Komponenta skripte omogućuje korisnicima određivanje slijeda radnji koje će izvršiti Home Assistant kada je uključen. Komponenta skripte će stvoriti entitet za svaku skriptu i omogućiti da se njima upravlja putem usluga. Glavna razlika između skripti i automatizacija je da skriptama nisu potrebni okidači da bi se izvodile.

7.2. Template

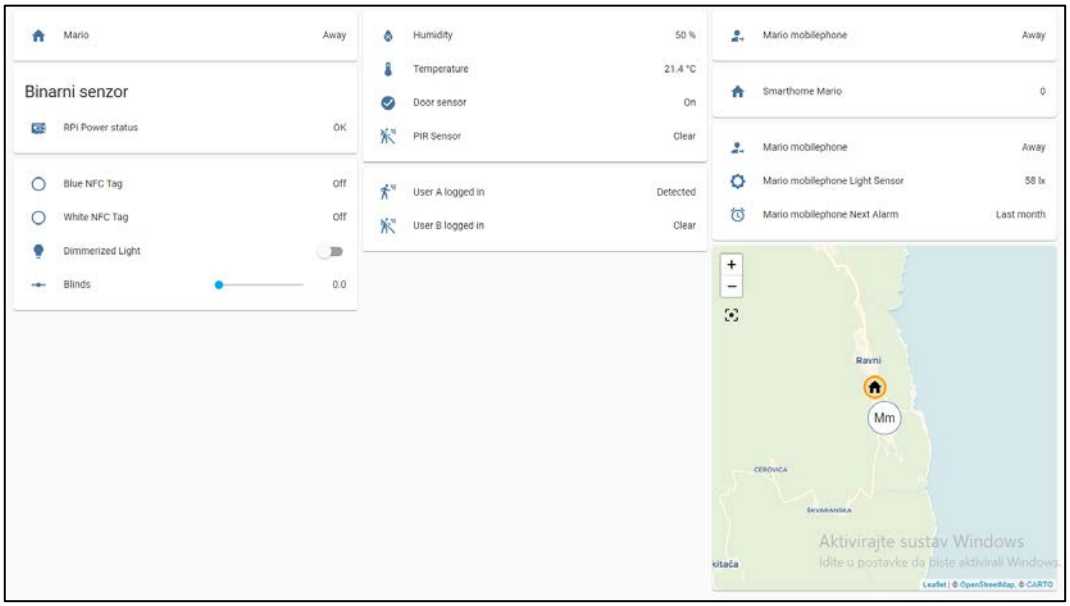
Integracija predložak (eng. *Template*) omogućuje stvaranje entiteta koji svoje vrijednosti generira iz vrijednosti ili nekog svojstva drugog entiteta. To se postiže određivanjem predložaka za svojstva entiteta, poput imena ili stanja. Npr. moguće je izraditi template senzor koji sprema srednju vrijednost očitanja dvaju senzora kao što je prikazano ispod.

```
template:
  - sensor:
      - name: "Average temperature"
        unit_of_measurement: "°C"
        state: >
          {% set bedroom = states('sensor.bedroom_temperature') | float %}
          {% set kitchen = states('sensor.kitchen_temperature') | float %}

          {{ ((bedroom + kitchen) / 2) | round(1, default=0) }}
```

7.3. Kontrolne ploče u Home Assistantu

- Kontrolna ploča Pregled

Kontrolna ploča Pregled sadrži sve osnovne informacije o stanjima senzora i aktuatora te prikazuje kartu gdje se nalazi pametna kuća i pametni telefon korisnika. Ova kontrolna ploča služi za provjera ispravnosti entiteta i testiranje.



Slika 7.5. *Kontrolna ploča Pregled*

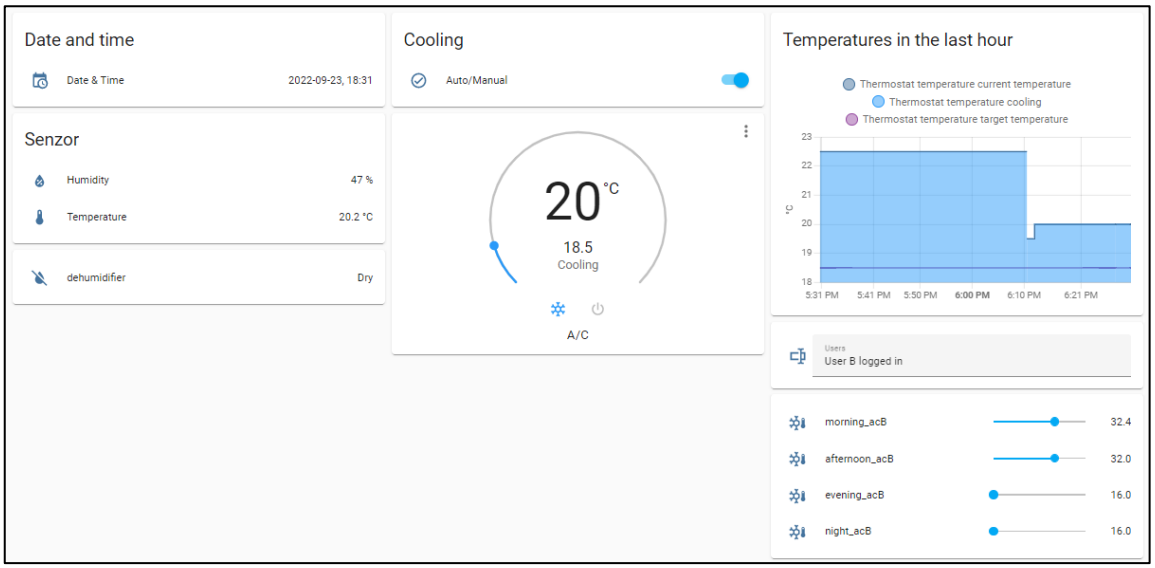
- Kontrolna ploča Klimatizacija

Kontrolna ploča klimatizacije (A/C) služi za upravljanje uređajem za klimatizaciju i odvlaživačem zraka. Na ovoj ploči nalaze se datum i vrijeme, vrijednosti temperature i vlage u kući, stanje odvlaživača zraka, tipka za prebacivanje iz automatskog u manualni način rada i obrnuto, termostat sa stvarnom i željenom temperaturom, graf koji prikazuje željenu i stvarnu temperaturu u posljednjih sat vremena, okvir koji prikazuje koji je korisnik prijavljen te na kraju vrijednosti temperatura po dijelovima dana (jutro, poslijepodne, večer i noć).

Klimatizacijom se može upravljati automatski i manualno. Korisnik se prijavljuje u sustav pametne kuće prislanjanjem svoje kartice na RFID čitač. U automatskom načinu rada klimatizacija se pali i podešava ovisno o korisniku i djelu dana. Svaki korisnik može si sam podesiti temperature po dijelovima dana. Npr. ako je zadana temperatura korisnika A ujutro 24°C, u 8:00h pali se klimatizacija sa tom temperaturom. Prijavom korisnika temperatura se automatski podešava ovisno o djelu dana. U manualnom modu temperatura klimatizacije ne ovisi o korisniku i djelu dana nego samo o termostatu kojeg se podešava.

Sustav može pamtiti temperature te ih prilagoditi korisniku. Ukoliko korisnik u manualnom načinu rada u nekom djelu dana pet puta promjeni temperaturu, sustav uzima srednju vrijednost tih pet temperatura te je postavlja kao zadanu u tom djelu dana u automatskom načinu rada. Na taj način sustav personalizira klimatizaciju prema korisniku.

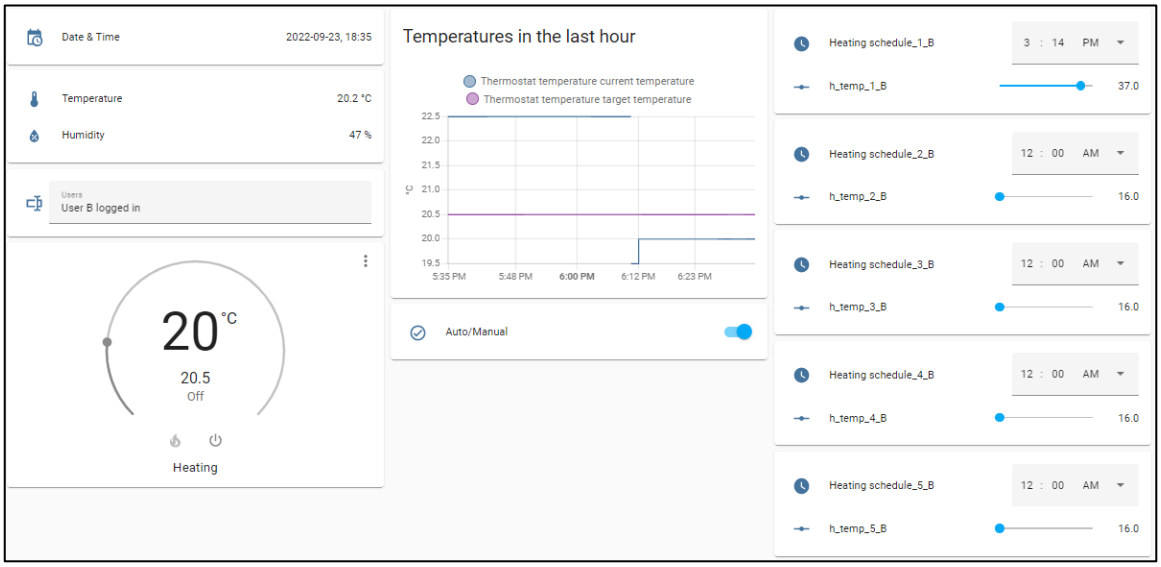
Radi veće efikasnosti i manjih troškova, otvaranjem vrata klimatizacija se gasi nakon tri sekunde te ju je nemoguće ponovno upaliti dok se vrata ne zatvore. Termostat klimatizacije ima histerezu od jednog stupnja. U slučaju da je zadana temperatura 24°C, kuća se hladi na temperaturu od 23°C te se zatim gasi. To sprječava konstantno paljenje i gašenje klimatizacije. Na istom principu radi odvlaživač zraka. Pali se na vrijednosti vlage 42% a gasi na 38%.



Slika 7.6. *Kontrolna ploča Klimatizacija*

- Kontrolna ploča Grijanje

Nadzorna ploča grijanja služi za upravljanje grijanjem pomoću termostata. Na ovoj kontrolnoj ploči nalaze se datum i vrijeme, vrijednosti temperature i vlage u kući, tipka za prebacivanje iz automatskog u manualni način rada i obrnuto, termostat sa stvarnom i željenom temperaturom, graf koji prikazuje željenu i stvarnu temperaturu u posljednjih sat vremena te na kraju raspored grijanja. Rasporedom grijanja korisnik si može sam izraditi automatski raspored grijanja, tj. koja temperatura će u određeno vrijeme biti zadana u kući.



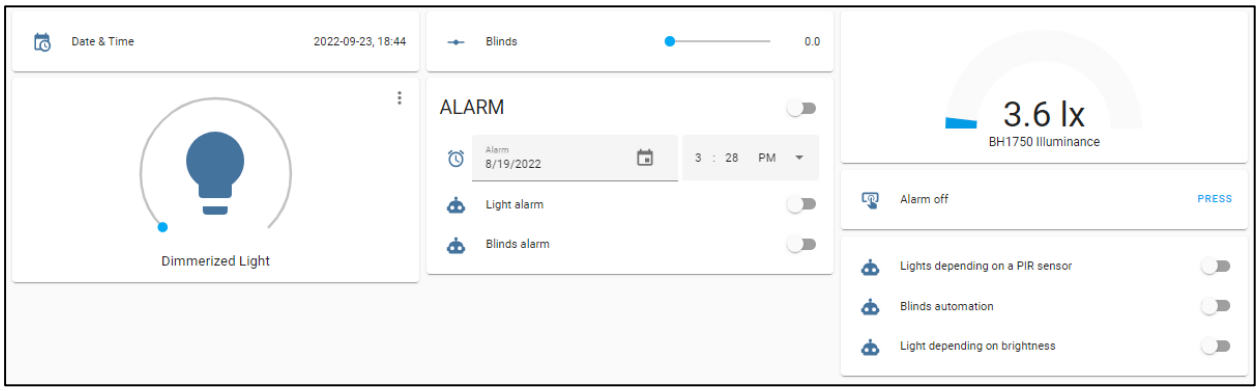
Slika 7.7. *Kontrolna ploča Grijanje*

- Kontrolna ploča Rasvjeta

Upravljanje rasvjetom izvodi se pomoću ove kontrolne ploča koja sadrži datum i vrijeme, entitet za paljenje i prigušivanje svjetla, klizač za otvaranja i zatvaranja roleta, entitet alarm, vrijednost očitanja senzora svjetlosti te automatizacije koje je moguće paliti i gasiti.

Moguće je kontrolirati razinu svjetlosti te paliti i gasiti svjetlo, otvarati i zatvarati rolete te ih postavljati u željeni položaj. Entitet alarm služi za buđenje korisnika ujutro. Postavi se vrijeme i datum alarma. U vrijeme pokretanja alarma svjetlo se počinje paliti. Tranzicija od stanja mraka do pune svjetlosti traje 20 sekundi (ova postavka se može mijenjati). To služi da se korisnik probudi opuštenije i sa manje stresa. Nakon 20 sekundi svjetlo je uključeno. Takva ista automatizacija se može izvesti sa roletama.

Na ovoj kartici postoje tri automatizacije. Prva automatizacija, „Light depending on brightness“ služi prigušivanju svjetla žarulje ovisno o vanjskoj svjetlosti koju senzor očitava. Na istom principu radi automatizacija „Blinds automation“ koja preslikava razinu svjetlosti u kući u razinu otvorenosti roleta. Zadnja automatizacija, „Lights depending on a PIR sensor“ pali svjetlo kada senzor detektira kretanje u prostoru ispred njega te ga gasi nakon 20 sekundi.



Slika 7.8. *Kontrolna ploča Rasvjeta*

- Kontrolna ploča Alarmi

Na kontrolnoj ploči alarma nalaze se 4 vrste alarma te tipka „alarm check“.
„ALARM intruders“ služi za detektiranje uljeza u kući nakon postavljanja alarma. Kada korisnik izlazi iz kuće uključuje alarm. Kada se vraća, nakon otvaranja vrata (očitanja reed releja) mora u

roku od 5 sekundi potvrditi karticom svoj identitet ili pritisnuti tipku „ALARM check“. U slučaju da ne potvrdi, pali se alarm te dolazi obavijest o provalniku na pametni telefon korisnika.

„ALARM – Lights on while user is away“ služi obavještavanju korisnika ukoliko je pustio upaljeno svjetlo i napustio kuću. Zatim nudi mogućnost gašenja svjetla.

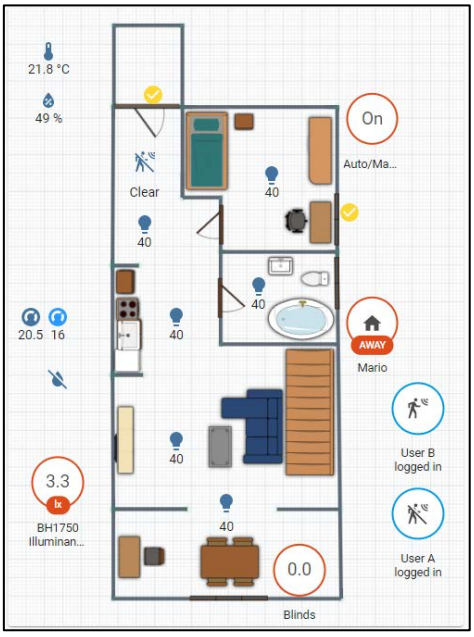
„ALARM - Blinds closing“- obavještava korisnika da dolazi vrijeme sa oborinama te mu nudi mogućnost zatvaranja roleta.

„ALARM – motion while user away“ se uključuje u slučaju da se detektira kretanje u kući dok je korisnik udaljen od kuće.

Slika 7.9. *Kontrolna ploča Alarmi*

- Kontrolna ploča Nadzor

Slika na ovoj kontrolnoj ploči nam omogućuje upravljanje svim entitetima i automatizacijama sa jednog mjesta, uz vizualnu pregled senzora i aktuatora u stvarnom vremenu.



Slika 7.10. *Kontrolna ploča Nadzor*

7.4. Mobilna aplikacija Home Assistant

Za upravljanje pametnom kućom pomoću pametnog telefona potrebno je instalirati aplikaciju *Home Assistant* koju skidamo putem *Google Play-a.* Kod prvog otvaranja aplikacije potrebno je upisati IP adresu na kojoj se nalazi Home Assistant. Zatim se upisuju korisničko ime i lozinka te je aplikacija spremna za korištenje. Korisničko sučelje je u načelu isto kao i na laptopu no neke funkcije su prilagođene manjoj veličini ekrana. Ukoliko je potrebno upravljati pametnom kućom „iz daljine“ dodava se integracija DuckDNS u HA. Ova integracija je besplatna dinamička DNS usluga koja omogućuje usmjeravanje osobne poddomene pod domenu duckdns.org, tj. aplikaciji se može pristupiti preko URL-a ukoliko pametni telefon i hub uređaj nisu povezani na istu lokalnu mrežu.

Slika 7.11. *Izgled Home Assistant sučenja na pametnom telefonu*

8. PRIMJENA UMJETNE INTELIGENCIJE NA PROJEKTNI ZADATAK

Za učinkovito upravljanje energijom u kući važno je predviđanje unutarnje temperature tog prostora. Točno predviđanje unutarnje temperature kuće ne samo da doprinosi poboljšanim uvjetima toplinske udobnosti, već također ima ulogu u očuvanju energije grijanja i hlađenja te služi kao dodatan korak ka potpunoj automatizaciji doma. Jedan od velikih potrošača električne energije u kući je i svjetlo. Točno predviđanje energetskih zahtjeva svjetlosti u kućanstvu moglo bi pružiti korisne informacije za donošenje odluka o proizvodnji i kupnji energije te bi se svjetlo moglo automatski paliti i gasiti.

8.1. Primjena umjetnih neuronskih mreža

U ovom primjeru koristi se umjetna neuronska mreža za predikciju vrijednosti temperature u kući, tj. stvaranje regresijskog modela temperature na temelju očitanja senzora. To je vrsta dinamičkog filtriranja, u kojem se prošle vrijednosti jedne ili više vremenskih serija podataka koriste za predviđanje budućih vrijednosti. Model je stvoren za korisnika A te se prema tome može personalizirati temperatura prema korisniku.

Prvi korak je stvaranje tablica (.csv datoteka) u koje će se podaci upisivati. Tablicu za sakupljanje temperatura, *temp_col_A.csv* u Home Assistantu predstavljamo kao entitet *filenotify* dok tablicu za sakupljanje vremenskih uzoraka, *time.csv* predstavljamo kao entitet *filenotify1*.

Temperature i vrijeme se zapisuju pomoću automatizacije „*Sensor data collecting (Setting temperature_user A)*“ čiji je programski kod opisan u nastavku. Svakom ručnom promjenom temperature korisnika A u tablice se zapisuju zadana temperatura i vrijeme promjene temperature.

```
alias: Sensor data collecting (Setting temperature_user A)
description: ""
trigger:                                    #okidač automatizacije – promjena stanja atributa
  - platform: state                         #temperature tj. identiteta climate.cooling
    entity_id:
      - climate.cooling
    attribute: temperature
condition:                                  #uvjet automatizacije – prisutnost korisnika A
  - condition: state
    entity_id: input_boolean.presence_user_a
    state: "on"
action:                                     #akcija automatizacije – upisivanje vrijednosti u
  - service: notify.filenotify1             #tablice temp_col_A i time
    data:
```

```
      message: "{{now().hour * 3600 + now().minute*60 + now().second}}" #pretvaranje u sekunde
  - service: notify.filenotify
    data:
      message: "{{states.climate.cooling.attributes.temperature}}"
mode: single
```

Nakon zapisivanja vrijednosti temperature i vremena u tablice potrebno je iste učitati u *Spyder-u*, programu za programiranje u python programskom jeziku. Otvaranje tablica vrši se sa sljedećim programskim kodom:

```
#otvaranje datoteke time.csv
with open(r'\\192.168.1.138\config\time.csv') as file_name:
    time = np.loadtxt(file_name, delimiter=",")             #upisivanje vrijednosti iz datoteke
                                                            #u varijablu time
print(time)

#otvaranje datoteke temp_col_a
with open(r'\\192.168.1.138\config\temp_col_A.csv') as file_name1:
    temp_col_A = np.loadtxt(file_name1, delimiter=",")      #upisivanje vrijednosti iz datoteke
                                                            #u varijablu temp_col_A
print(temp_col_A)
```

Pošto se ovaj model ne koristi konstantno i služi samo kao prezentacija pametne kuće, nije moguće dobiti točne i konstantne podatke željene temperature i vremena pa se te vrijednosti za ovaj rad odabiru nasumičnim funkcijama. Vrijeme u sekundama se dijeli na 400 dijelova, tj. simulira se uzimanje uzorka svakih 216 sekundi. Vrijednosti temperatura imaju isti broj uzoraka kao i vrijeme, no uzorci su podijeljeni na 10 dijelova u kojima postoje maksimalna i minimalna temperatura između kojih se uzimaju nasumične vrijednosti.

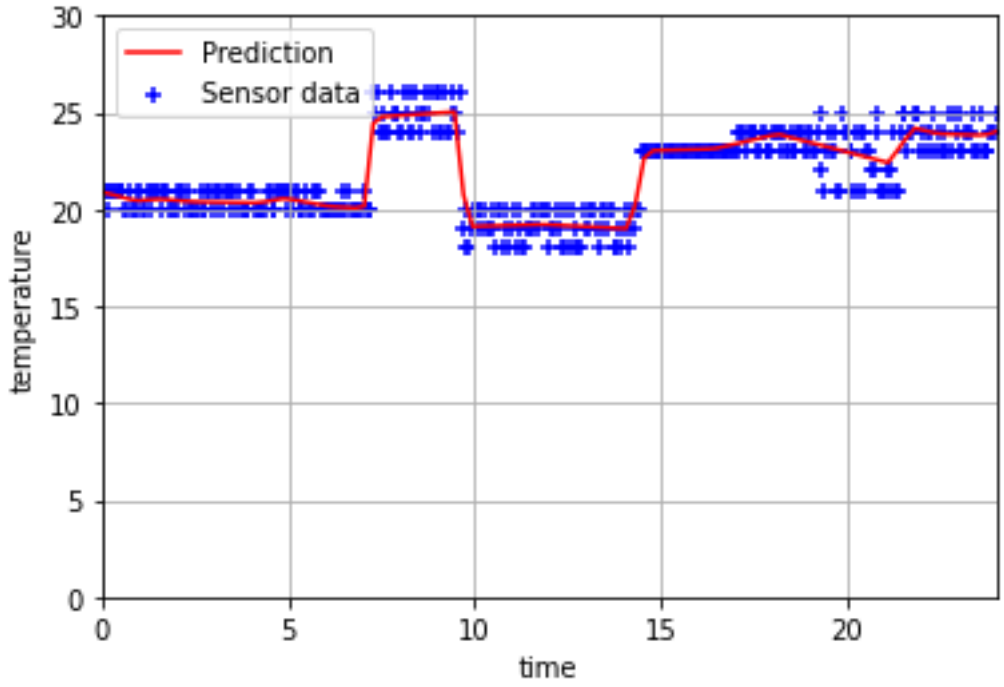
Nakon predprocesiranja podataka poželjno je podatke normirati. Vremenski podaci se dijele sa 86400 (broj sekunda u danu) a zadane temperature sa 26 (maksimalna temperatura). Za stvaranje modela koristi se *MLPRegressor* tj. višeslojna mreža sa svojstvom regresora. Funkcija *GridSearchCV()* služi za odabir najoptimalnijih hiperparametra između nekoliko prije odabranih s obzirom na podatke za treniranje. Nakon odabira hiperparametara mreže stvara se model neuronske mreže koji ima mogućnost predviđanja temperatura ovisno o vremenu.

```
import numpy as np                          #uvoz biblioteka
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPRegressor
import random
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
```

```
random.seed(0)                              #inicijalizacija generatora nasumičnih brojeva

time = []                                   #stvaranje liste time
z = 216                                     #stvaranje varijable z

for i in range(0, 400):                     #iteracija po i
    time.append(z)                          #dodavanje vrijednosti varijable z u listu time
    z = z + 216                             #zbrajanje varijable z sa 216
time = np.array(time)                       #pretvaranje liste u polje


temp = []                                   #stvaranje liste temp

for i in range(0, 40):                      #iteracija po i
    temp.append(random.randint(20, 22))     #ubacivanje nasumičnih brojeva u listu

for i in range(40, 80):
    temp.append(random.randint(24, 26))

for i in range(80, 120):
    temp.append(random.randint(20, 21))

for i in range(120, 160):
    temp.append(random.randint(18, 20))

for i in range(160, 200):
    temp.append(random.randint(18, 20))

for i in range(200, 240):
    temp.append(random.randint(24, 25))

for i in range(240, 280):
    temp.append(random.randint(23, 23))

for i in range(280, 320):
    temp.append(random.randint(23, 24))

for i in range(320, 360):
    temp.append(random.randint(21, 25))

for i in range(360, 401):
    temp.append(random.randint(23, 25))


temp = np.array(temp)                       #pretvaranje liste u polje

temp = temp / 26                            #normiranje vrijednosti

time = time / 86400
```

```
n = len(time)                                   #spremanje dužine polja time u varijablu n

time = np.reshape(time, [n, 1])                 #mijenjanje oblika polja u (50, 1)

estimator = MLPRegressor()                      #definiranje estimatora kao MLPRegressor()


param_grid = {                                   #definiranje mogućih hiperparamtara mreže
    'hidden_layer_sizes': [(50, 50, 50), (50, 100, 50), (100, 1)],
    'activation': ['relu', 'tanh', 'logistic'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant', 'adaptive'],
    'solver': ['adam']}

#traženje najoptimalnijih hiperparametra
gsc = GridSearchCV(estimator, param_grid, cv = 5, scoring = 'neg_mean_squared_error', verbose = 0,
n_jobs = -1)
grid_result = gsc.fit(time, temp)
best_params = grid_result.best_params_
gsc.fit(time, temp)

#stvaranje modela neuronske mreže sa najoptimalnijim parametrima
model = MLPRegressor(hidden_layer_sizes = best_params["hidden_layer_sizes"],
                     activation = best_params["activation"],
                     solver = best_params["solver"],
                     max_iter = 500000,
                     n_iter_no_change = 20000)


model.fit(time, temp)                       #treniranje modela

n_test = 100                                #broj testnih podataka
x_test = np.linspace(0, 1, n_test)          #polje sa testnim podatcima
x_test = np.reshape(x_test, [n_test, 1])    #mijenjanje oblika polja u (n_test, 1)
predictions = model.predict(x_test)  #ubacivanje testnih vrijednosti u model neuronske mreze

#crtanje grafa prikupljenih i predviđenih podataka

plt.figure()
plt.plot(x_test * 86400 / 3600, predictions * 26, color = 'r', label = 'Prediction')
plt.scatter(time * 86400 / 3600, temp * 26, color = 'b', marker = '+', label = 'Sensor data')
plt.xlim((0, 24))
plt.ylim((0, 30))
plt.xlabel('time')
plt.ylabel('temperature')
plt.grid()
plt.legend()
plt.show()
```

Na slici 8.1. prikazane su vrijednosti temperatura u jednom danu. Sa markerom '+' inače su prikazana očitanja senzora, no u ovom slučaju to su nasumične vrijednosti koje prikazuju zadane temperature za jedan radni dan. Pomoću grafa može se zaključiti ponašanje korisnika, povećavanje temperature ujutro kada se probudi i popodne kada stigne sa posla. Niže temperature su zadane preko noći i ujutro dok je korisnik na poslu. Crvenom bojom prikazano je predviđanje temperature pomoću modela neuronske mreže. U model se ubacuje 100 testnih podataka, tj. vremenskih uzoraka za koje mreža predviđa željenu temperaturu.



Slika 8.1. *Graf podataka sa senzora i graf predviđenih temperatura*

Nakon predviđanja podataka, potrebno je podatke prilagoditi i vratiti u program HA. To se radi tako da se predviđene temperature zaokružuju na cijeli broj ili na decimalni sa decimalom 0.5.

```
def roundToHalf(array):                    #funkcija za zaokruživanje
   return np.around(array * 2.0) / 2.0

predictions_ = roundToHalf(predictions*25) #zaokruživanje vrijednosti i spremanje u
                                            #varijablu preditions_
```

Integracijom *csvwriter* upisujemo vrijednosti iz polja u datoteku *predictions_.csv* prema sljedećem programskom kodu:

```
filename = r'\\192.168.1.138\config\predictions.csv'
with open(filename, 'w', newline="") as file:          #otvaranje datoteke
    csvwriter = csv.writer(file)                       #stvaranje cswriter objekta
    csvwriter.writerow(predictions_)                   #upisivanje u datoteku
```

U HA-u izrađuje se skripta koja zadaje temperaturu hlađenja tako da uzima vrijednost temperature iz datoteke *predictions_.csv* ovisno o vremenu.

```
alias: AI temperature set
sequence:
  - service: climate.set_temperature
    data:
      temperature: "{{ states.sensor.time_pr.state.split(\",\")[now().hour] }}"
    target:
      entity_id: climate.cooling
mode: single
```

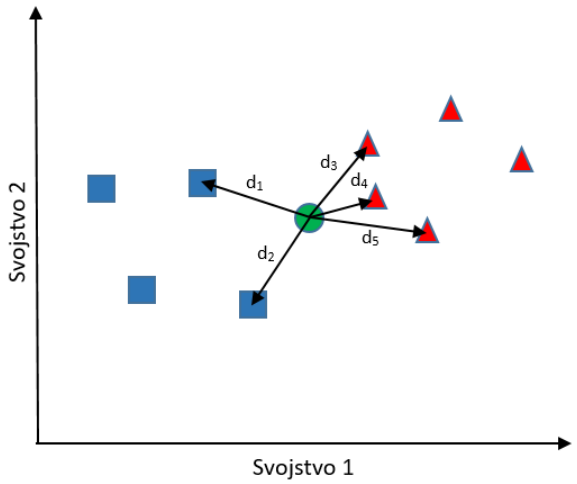
Sada je moguće stvoriti automatizacija u HA koja namješta temperaturu svakih sat vremena. Automatizacija se pokreće na svaki puni sat te pokreće prethodno definiranu skriptu.

```
alias: AI set temperature
description: ""
trigger:
  - platform: time
    at: "00:00:00"
  - platform: time
    at: "01:00:00"
  - platform: time
    at: "02:00:00"
  .
  .
  .
  - platform: time
   at: "21:00:00"
  - platform: time
    at: "22:00:00"
  - platform: time
    at: "23:00:00"

condition: []
action:
  - service: script.turn_on
    data: {}
    target:
      entity_id: script.1662296611102
mode: single
```

8.2. Primjena algoritma K-Najbližih susjeda

Algoritam K-Najbližih susjeda (*eng. K-Nearest Neighbor Algorithm* – KNN) metoda je strojnog učenja koja se smatra vrlo jednostavnom za implementaciju. KNN metoda je vrlo intuitivna metoda koja klasificira objekt na temelju podataka o učenju koji su najbliži objektu. Objekt je klasificiran prema podacima o najbližem susjedu, pri čemu se objektu dodjeljuje najčešća klasa među njegovih najbližih k susjeda (k je pozitivan cijeli broj, obično mali). Ako je k = 1, tada se objekt dodjeljuje samo najbližoj susjednoj klasi. [17]

Problem predviđanja ponašanja korisnika može se preslikati u sličnosti na temelju klasifikacije. Korisnički povijesni podaci i testni podaci mapirani su u vektorski skup. Svaki vektor predstavlja N dimenziju za svaku naviku korisnika. Zatim se za donošenje odluka izračunavaju metrike sličnosti kao što je euklidska udaljenost. [17]



Slika 8.2. *Distribucija podataka k-NN algoritma*

Slika 8.2 prikazuje distribuciju podataka k-NN algoritma koja ima dvije klase (trokuti i kvadrati) i dva svojstva (svojstvo 1 i svojstvo 2). Podaci koji već imaju definirane klase služe kao podaci za treniranje. Prema slici 8.2 odabrano je $k_{susjeda}$= 5 te će algoritam tražiti 5 podatka za treniranje koji su najbliži testnom podatku (zeleni krug). [17]

Udaljenost između podataka testa i podataka treninga može se izračunati mjerenjem udaljenosti između točaka koje predstavljaju podatke testa i svih točaka koje predstavljaju podatke treninga pomoću formule Euklidske udaljenosti:

$$d_i = \sqrt{(x_i - x_t)^2 + (y_i - y_t)^2} \quad (3.3),$$

gdje je:

d_i - udaljenost kordinata testnog podatka i kordinata i-tog podatka za trening,

x_i - x kordinata i-te točke koja predstavlja podatak za trening (svojstvo 1 podatka),

y_i - y kordinata i-te točke koja predstavlja podatak za trening (svojstvo 2 podatka),

x_t - x kordinata točke koja predstavlja testni podatak (svojstvo 1 podatka),

y_t - y kordinata točke koja predstavlja testni podatak (svojstvo 2 podatka).

Udaljenosti se sortiraju i najbliži susjedi se određuju na temelju broja najbližih udaljenosti tj. susjeda. Većinska kategorija od najbližeg susjeda koristi se kao rezultat klasifikacije testnih podataka. Na slici 8.2. zeleni krug će se klasificirati kao klasa trokuta. [17]

U ovom primjeru predviđati će se vremenski intervali po danima u tjednu u kojima je svjetlo uključeno ili isključeno. Za izradu modela najprije je potrebno prikupiti i pretprocesirati podatke, tj. prilagoditi ih tako da se mogu programski obraditi. Svaki dan u tjednu predstavlja jedan cijeli broj, tako da npr. ponedjeljak odgovara broju 1, utorak broju 2 itd. Isto tako svakom vremenskom intervalu pridružuje se cijeli broj pa interval od 8:00h do 8:59h predstavlja broj 8.

Za izradu modela najprije je potrebno prikupiti podatke iz sustava. Za to se koristi automatizacija „*Sensor data collecting (Turning ON/OFF light by user A)*“ koja svakih pet minuta provjerava stanje svjetla te ga upisuje u tablicu zajedno sa danom u tjednu i vremenom. Svaki parametar (stanje, dan u tjednu, vrijeme) upisuje se u posebnu tablicu. Kao i u prijašnjem primjeru, u HA-u tablice *day.csv* (dani u tjednu), *hour.csv* (vremenski intervali u danu) i *ON_OFF.csv* (stanje svjetla) predstavljamo redom kao *filenotify2, filenotify3, filenotify4.*

```
alias: Sensor data collecting (Turning ON/OFF light by user A)
description: ""
trigger:                                    #okidač automatizacije – vremenski obrazac (5 min)
  - platform: time_pattern
    minutes: "5"
condition:                                  #uvjet automatizacije – prijavljen korisnik A
  - condition: state
    entity_id: input_boolean.presence_user_a
    state: "on"
action:                                     #akcija automatizacije (if-then)
  - if:                                     #if
      - condition: device                   #entitet light.dimmerized_light ima stanje on
        type: is_on
        device_id: f3ea0229be091a90be0dc50b7bc8bd2f
```

```
        entity_id: light.dimmerized_light
        domain: light
    then:                                   #then
      - service: notify.filenotify4
        data:
          message: "{{1}}"                  #upisivanje u datoteku filenotify4 „1“
    else:                                   #else
      - service: notify.filenotify4
        data:
          message: "{{0}}"                  #upisivanje u datoteku filenotify4 „0“

  - service: notify.filenotify2
    data:
      message: "{{now().isoweekday()}}"     #upisivanje u datoteku filenotify2 dan u tjednu
  - service: notify.filenotify3
    data:
      message: "{{now().hour}}"             #upisivanje u datoteku filenotify3 vrijeme
mode: single
```

Nakon zapisivanja u tablice potrebno je iste učitati u *Spyder-u*. Otvaranje tablica vrši se sa sljedećim programskim kodom:

```
with open(r'\\192.168.1.138\config\day.csv') as file_name:      #otvaranje datoteke time.csv
day = np.loadtxt(file_name, delimiter = ",")                    #upisivanje vrijednosti iz datoteke u
                                                                #varijablu time

with open(r'\\192.168.1.138\config\hour.csv') as file_name1:    #otvaranje datoteke temp_col_A
hour = np.loadtxt(file_name1, delimiter = ",")                  #upisivanje vrijednosti iz datoteke u
                                                                #varijablu temp_col_A

with open(r'\\192.168.1.138\config\ON_OFF.csv') as file_name2: #otvaranje datoteke temp_col_A
state = np.loadtxt(file_name2, delimiter = ",")                 #upisivanje vrijednosti iz datoteke u
                                                                #varijablu temp_col_A
```

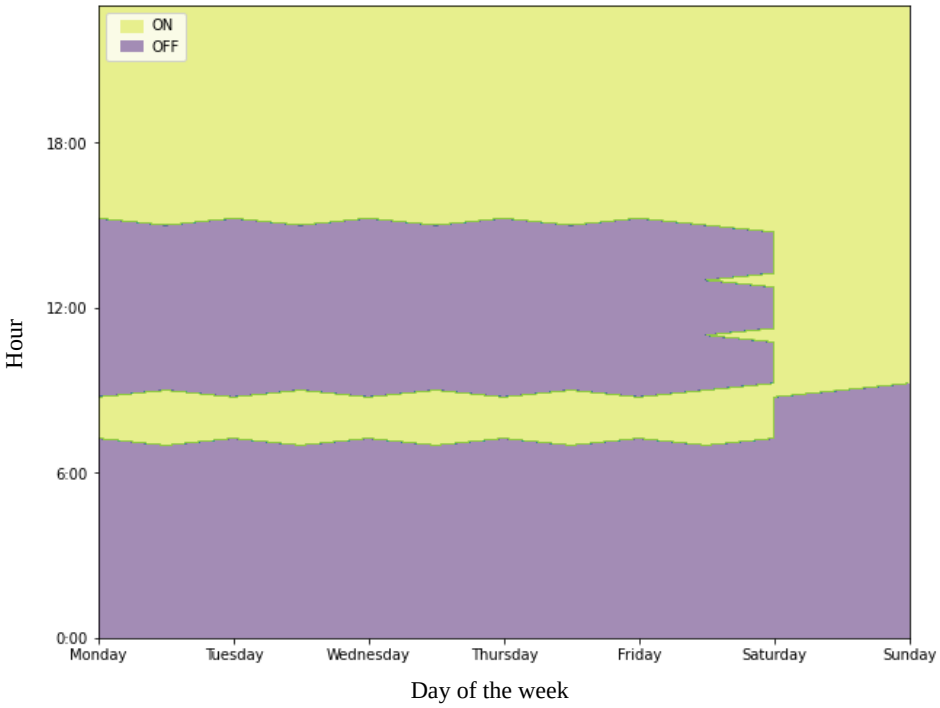
I u ovome primjeru sakupljeni podaci ne bi bili dovoljno točni i njihova količina bi bila veoma mala pa se iz tog razloga podaci simuliraju tako da se može demonstrirati rad algoritma za korištenje ubuduće.

Ubacivanjem podataka u program *Spider* stečeni su uvjeti za izradu algoritma k najbližih susjeda. Programski kod započinje ispunjavanjem lista sa podacima o danu, vremenu i stanju svjetla. Zatim se stvara model algoritma k najbližih susjeda kojemu se odabire parametar *n_neighbours* tj. broj susjeda. Na kraju koda, za predviđanje upotrebe svjetla, gradi se mreža od 1920000 točaka koja se ubacuje u model te se na grafu iscrtava područje uporabe svjetla.

```
import numpy as np                      #uvoz biblioteka i alata
import matplotlib.pyplot as plt
from sklearn import svm
import random

#Monday                                 #stvaranje listi za svaki dan pojedinačno
day1=[1,1,1,1,1,1,1,1,1,1,1]
hour1=[0,2,4,6,8,10,12,14,16,18,20,22]
state1=[0,0,0,0,1,0,0,0,1,1,1,1]

#Tuesday
day2=[2,2,2,2,2,2,2,2,2,2,2]
hour2=[0,2,4,6,8,10,12,14,16,18,20,22]
state2=[0,0,0,0,1,0,0,0,1,1,1,1]

#Wednesday
day3=[3,3,3,3,3,3,3,3,3,3,3]
hour3=[0,2,4,6,8,10,12,14,16,18,20,22]
state3=[0,0,0,0,1,0,0,0,1,1,1,1]

#Thursday
day4=[4,4,4,4,4,4,4,4,4,4,4]
hour4=[0,2,4,6,8,10,12,14,16,18,20,22]
state4=[0,0,0,0,1,0,0,0,1,1,1,1]

#Friday
day5=[5,5,5,5,5,5,5,5,5,5,5]
hour5=[0,2,4,6,8,10,12,14,16,18,20,22]
state5=[0,0,0,0,1,0,0,0,1,1,1,1]

#Saturday
day6=[6,6,6,6,6,6,6,6,6,6,6]
hour6=[0,2,4,6,8,10,12,14,16,18,20,22]
state6=[0,0,0,0,1,1,1,1,1,1,1,1]

#Sunday
day7=[7,7,7,7,7,7,7,7,7,7,7]
hour7=[0,2,4,6,8,10,12,14,16,18,20,22]
state7=[0,0,0,0,0,1,1,1,1,1,1,1]

day=[]                                  #spajanje listi
day.extend(day1)
day.extend(day2)
day.extend(day3)
day.extend(day4)
day.extend(day5)
day.extend(day6)
day.extend(day7)

hour=[]
hour.extend(hour1)
hour.extend(hour2)
hour.extend(hour3)
hour.extend(hour4)
hour.extend(hour5)
hour.extend(hour6)
hour.extend(hour7)
```

```
state=[]
state.extend(state1)
state.extend(state2)
state.extend(state3)
state.extend(state4)
state.extend(state5)
state.extend(state6)
state.extend(state7)


x_input=np.array(list(zip(day,hour)))                #stvaranje matrice svojstava
y_input=np.array(state)                              #stvaranje matrice klasa

from sklearn.neighbors import KNeighborsClassifier   #uvoz biblioteke

knn = KNeighborsClassifier(n_neighbors=2)            #stvaranje modela algoritma K-NN
knn.fit(x_input, y_input)                            #unos podatka za treniranje


xx, yy = np.meshgrid(np.arange(1, 7, 0.01), np.arange(0, 23, 0.01) #stvaranje mreže za predikciju

Z = knn.predict(np.c_[xx.ravel(), yy.ravel())        #unos podataka u model
Z = Z.reshape(xx.shape)                              #promjena oblika matrice

ticks_x = [1, 2, 3, 4, 5, 6, 7]                      #definiranje oznaka grafa
labels_x = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

ticks_y = [0, 6, 12, 18]
labels_y = ["0:00", "6:00", "12:00", "18:00"]

plt.figure(figsize = (10, 8))                        #crtanje grafa
cs=plt.contourf(xx, yy, Z, alpha=0.5)
labels = ['ON', 'OFF']
for i in range(len(labels)):
    cs.collections[i].set_label(labels[i])
plt.legend(loc='upper left')
plt.xticks(ticks_x, labels_x)
plt.yticks(ticks_y, labels_y)
plt.show()
```

Slika 8.3. prikazuje graf predviđanja stanja svjetla u jednom tjednu. Vidljivo je da korisnik pametne kuće ima upaljeno svjetlo ujutro prije posla te popodne kada je doma. Vikendom se korisnik kasnije budi te svjetlo biva kasnije uključeno. Egzaktan način odabira najboljeg broja k najbližih susjeda ne postoji te se on dobiva empirijski, tj. pomoću testiranja i na temelju pokušaja i pogrešaka.



Slika 8.3. *Predviđanje stanja svjetla pomoću K-NN algoritma*

8.2. Primjena genetskog algoritma

Genetski algoritmi (GA) su algoritmi koji se temelje na pretraživanju i temelje se na konceptima prirodne selekcije i genetike. GA su podskup puno veće grane računalstva poznate kao evolucijsko računalstvo. U GA postoji skup ili populacija mogućih rješenja danog problema. Ta rješenja zatim prolaze kroz rekombinaciju i mutaciju (kao u prirodnoj genetici), proizvodeći novu djecu, a proces se ponavlja kroz različite generacije. Svakoj jedinki (ili kandidatskom rješenju) dodijeljena je funkcija podobnosti (temeljena na vrijednosti objektivne funkcije), a bolje sposobne jedinke imaju veću šansu za parenje i daju više "sposobnijih" jedinki. To je u skladu s Darwinovom teorijom o "opstanku najjačih". [17]

Genetski algoritam koristi tri glavne vrste pravila u svakom koraku za stvaranje sljedeće generacije iz trenutne populacije:

- Pravila odabira odabiru jedinke, zvane roditelji, koji doprinose populaciji u sljedećoj generaciji. Odabir je općenito stohastički i može ovisiti o rezultatima pojedinaca.
- Pravila križanja spajaju dva roditelja kako bi formirali djecu za sljedeću generaciju.
- Pravila mutacije primjenjuju nasumične promjene na pojedinačne roditelje kako bi se formirala djeca.

Slika 8.4. *Dijagram toka genetskog algoritma*

Genetski algoritam sastoji se od 4 glavna elementa:

- Početna populacija: stvaranje nasumične populacije kandidata rješenja,
- Funkcija podobnosti: daje ocjenu dobrote (kvalitete) pojedinog rješenja,
- Genetski operatori: mijenjaju oblik rješenja sa ciljem povećanja funkcije pogodnosti (križanje i mutacija),
- Uvjet zaustavljanja: služi zaustavljanju algoritma ukoliko je ispunjen prethodno zadani uvjet.

Umjetna inteligencija se u ovom primjeru koristi za pamćenje navika paljenja svjetla uporabom genetskog algoritma. Ulazni podaci za algoritam prikupljaju se paljenjem svjetla. U tablice *day.csv*, *hour.csv*, *blinds.csv* i *light_sensor.csv* upisuju se vrijednosti dana, sata, razine otvorenosti roleta i razina svjetlosti izvan kuće. Cilj je za određena stanja navedenih entiteta odrediti kada će se automatski paliti svjetlo. Nadalje, u *Python* programu stvara se tablica koja sadrži prethodno sakupljene podatke u obliku da svaki red predstavlja vrijednosti entiteta u trenutku paljenja svjetla. U tablici se traže redovi koji se često ponavljaju ili su veoma slični te oni predstavljaju najčešće navike korisnika.

Podaci se sakupljaju svakim paljenjem svjetla pomoću sljedeće automatizacije:

```
alias: Sensor Data collecting(GA_Recognizing habits_AI)
description: ""
trigger:                                                    #okidač automatizacije
  - platform: device
    type: turned_on
    device_id: f3ea0229be091a90be0dc50b7bc8bd2f
    entity_id: light.dimmerized_light
    domain: light
condition:                                                  #uvjet automatizacije
  - condition: state
    entity_id: input_boolean.presence_user_a
    state: "on"
action:                                                     #akcija automatizacije
  - service: notify.filenotify5
    data:
      message: "{{now().hour}}"
  - service: notify.filenotify6
    data:
      message: "{{now().isoweekday()}}"
  - service: notify.filenotify7
    data:
      message: "{{states.sensor.bh1750_illuminance.state  | round(0)}}"
  - service: notify.filenotify8
    data:
      message: "{{states.input_number.blinds.state  | round(0)}}"
mode: single
```

Da bi se mogli koristiti prethodno sakupljenim podacima, potrebno je tablice učitati u *Spider-u* pomoću sljedećeg koda:

```
#otvaranje datoteke GA_day.csv i upisivanje u polje time
with open(r'\\192.168.1.138\config\GA_day.csv') as file_name:
    day = np.loadtxt(file_name, delimiter=",")

#otvaranje datoteke GA_hour.csv i upisivanje u polje hour
with open(r'\\192.168.1.138\config\GA_hour.csv') as file_name1:
    hour = np.loadtxt(file_name1, delimiter=",")

#otvaranje datoteke GA_Blinds.csv i upisivanje u polje blinds
with open(r'\\192.168.1.138\config\GA_Blinds.csv') as file_name2:
    blinds = np.loadtxt(file_name2, delimiter=",")

#otvaranje datoteke GA_Light_Sensor.csv i upisivanje u polje light_sensor
with open(r'\\192.168.1.138\config\GA_Light_Sensor.csv') as file_name2:
    light_sensor = np.loadtxt(file_name2, delimiter=",")
data=[]
data.append(day)
data.append(hour)
data.append(blinds)
data.append(light_sensor)
data=np.array(data)
data=data.T
```

I u ovom primjeru očitanja senzora su nasumično odabrana radi nemogućnosti konstantnog sakupljanja podataka.

Ovaj algoritam se sastoji od 4 osnovna elementa: funkcije *generate_parent*, funkcije *get_fitness*, funkcije *mutate* i *glavnog algoritma*. Funkcija *generate_parent* stvara polje sa 4 ćelije koje predstavlja roditelja tj. prvog kandidata rješenja. Funkcija *get_fitness* određuje vrijednost funkcije pogodnosti prema rješavanju problema, tj. koliko dobro roditelj odgovara rješenju. Funkcija *mutate* primjenjuje genetski operator mutacije koji uzima jedan genotip i nasumično ga mijenja.

Glavni algoritam sadrži petlju koja obavlja sljedeće radnje:

1. Stvaranje prvog kandidata rješenja,
2. Izračunavanje njegove funkcije pogodnosti,
3. Petlja koja modificira kandidata rješenja pomoću funkcije mutate sve dok nije zadovoljen uvjet zaustavljanja,
4. Kandidat rješenja se uspoređuje sa svim ostalim poljima navika i traži se njegovo preklapanje sa drugim elementima matrice. Ovisno o broju istih elemenata kandidat rješenja se sprema u tablicu *n_matching*,
5. Petlja se vraća na početak.

```
import csv                          #uvoz potrebnih biblioteka i funkcija
import numpy as np
import matplotlib.pyplot as plt
import random

def solve(lis):                     #funkcija za uklanjanje duplih elementa
    seen = set()
    for x in lis:
        if tuple(x) not in seen:
            yield x
            seen.add(tuple(x))

geneset1 = [i for i in range(1,8)]   #definiranje seta podataka za dan
geneset2 = [i for i in range(0,24)]  #definiranje seta podataka za sat
geneset3 = [i for i in range(0,101)] #definiranje seta podataka za razinu otvorenosti roleta
geneset4 = [i for i in range(0,26)]  #definiranje seta podataka za razinu svjetlosti

n=50
data=np.array([[0] * 4] * n)       #definiranje polja veličine 4xn

#popunjavanje polja nasumičnim vrijednostima
```

```
for i in range(0,n):
    data[i,0]=random.sample(geneset1, 1)[0]
    data[i,1]=random.sample(geneset2, 2)[0]
    data[i,2]=random.sample(geneset3, 1)[0]
    data[i,3]=random.sample(geneset4, 1)[0]

#dodjeljivanje istih vrijednosti redovima 14,30,45
data[14]=[3,19,27,21]
data[30]=[3,19,27,21]
data[45]=[3,19,27,21]

#definiranje funkcije generate_parent
def generate_parent():
    genes = []
    genes.extend(random.sample(geneset1, 1))

    genes.extend(random.sample(geneset2, 1))

    genes.extend(random.sample(geneset3, 1))

    genes.extend(random.sample(geneset4, 1))

    return genes

#definiranje funkcije get_fitness
def get_fitness(guess,j):
    fit = sum(1 for expected, actual in zip(data[j], guess) if expected == actual)
    return fit

#definiranje funkcije mutate
def mutate(parent):
    index = random.randrange(0, len(parent))
    child_genes = list(parent)
    if(index==0):
        new_gene, alternate = random.sample(geneset1,2)
        child_genes[index] = alternate if new_gene == child_genes[index] else new_gene
    if(index==1):
        new_gene, alternate = random.sample(geneset2,2)
        child_genes[index] = alternate if new_gene == child_genes[index] else new_gene
    if(index==2):
        new_gene, alternate = random.sample(geneset3,2)
        child_genes[index] = alternate if new_gene == child_genes[index] else new_gene
    if(index==3):
        new_gene, alternate = random.sample(geneset4,2)
        child_genes[index] = alternate if new_gene == child_genes[index] else new_gene
    child_genes=np.array(child_genes)
    return child_genes

#definiranje praznih polja

n_matching=[]

n_matching2=[]

n_matching3=[]

n_matching4=[]

#glavni algoritam
```

```
for i in range(len(data)):
    best_parent=generate_parent()
    best_fitness=get_fitness(best_parent,i)
    while(True):
        child = mutate(best_parent)
        child_fitness = get_fitness(child,i)
        # print(str(child))
        # print(str(child_fitness))
        if(best_fitness >= child_fitness):
            continue
        # display(child)
        if(child_fitness >= len(best_parent)):
            print('')
            # print('Element number:' +str(i))
            print('child_fitness: '+str(child_fitness))
            break

        best_fitness = child_fitness

        best_parent = child
    for k in range(len(data)):
        if(k != i):
            # print('')
            # print(k)
            # print('fitness with',str(k),'. column:' +str(get_fitness(child,k)))
            if(get_fitness(child,k)==1):
                n_matching2.append(data[k])
            if(get_fitness(child,k)==2):
                n_matching3.append(data[k])
            if(get_fitness(child,k)==3):
                n_matching.append(data[k])
            if(get_fitness(child,k)==4):
                print('fitness with',str(k),'. column:' +str(get_fitness(child,k)))
                n_matching4.append(data[k])

#brisanje duplih elemenata iz polja n_matching4 i upisivanje u polje n_matching

n_matching4=tuple(solve(n_matching4))

n_matching=np.array(n_matching4)
```

Najčešće navike se spremaju u datoteku Light_routine prema sljedećem programskom kodu:

```
filename = r'\\192.168.1.138\config\Light_routine.csv'
with open(filename, 'w', newline="") as file:
    csvwriter = csv.writer(file)
    csvwriter.writerow(n_matching)
```

Automatizacija na temelju rješenja genetskog algoritma se radi tako da kao okidač automatizacije služi predviđeno vrijeme. U to vrijeme automatski se pali svjetlo i rolete se postavljaju u željeni položaj.

```
alias: GA
description: ""
trigger:
  - platform: state
    entity_id:
      - input_number.ga_pom
    from: "{{ states.sensor.Light_routine.state.split(\" \")[2] | int }}"
    to: "{{ states.sensor.Light_routine.state.split(\" \")[2] | int }}"
condition: []
action:
  - type: turn_on
    device_id: f3ea0229be091a90be0dc50b7bc8bd2f
    entity_id: light.dimmerized_light
    domain: light
  - service: input_number.set_value
    data:
      value: "{{ states.sensor.Light_routine.state.split(\" \")[3]  }}"
    target:
      entity_id: input_number.blinds
mode: single
```

Točnost ovih obrađenih algoritama može se povećati većim brojem podataka za treniranje, tj. dužim praćenjem navika korisnika u manjim intervalima. Jedan od problema leži u tome da korisnici tj. ljudi mijenjaju svoje navike ovisno o načinu života u nekom periodu te zato algoritam treba učestalo hraniti novim podacima. Preciznost algoritma biti će bolja što je kaotičnost načina života korisnika manja. Npr. stariji ljudi u mirovini imaju predvidljivije ponašanje i navike koje je lakše modelirati. Ovi rezultati su prediktivni rezultati koji nisu nužno točni prema stvarnim uvjetima, zbog različitog ponašanja korisnika.

ZAKLJUČAK

U ovom radu predstavljen je kratak uvod u pojam pametnih kuća, njihove prednosti i nedostaci, važnost IoT tehnologija te komunikacijski modeli i protokoli za razmjenu informacija. U budućnosti će se koncept pametnih kuća sve više koristiti i biti jedan od osnovnih elementa kod planiranja i izgradnje kuća no sa njime se treba koristiti sa velikim oprezom jer pametne kuće imaju i svoje nedostatke, kako u hardverskom smislu tako i softverskom. Dolazi se do problema većih početnih financijskih ulaganja, rizika od zakazivanja opreme i krađe podataka putem interneta, no razvojem tehnologije i povećanjem sigurnosti ti će se problemi minimizirati ili ukloniti.

U trećem poglavlju opisana je umjetna inteligencija te primjeri njezina korištenja kod pametnih kuća. Korištenje UI-e ima velike prednosti za izradu komfornijeg i sigurnijeg doma te je postala ključan element u automatizaciji doma. Povećava se autonomija i sigurnost kuća, te se može smanjiti potrošnja električne energije planiranjem potrošnje korištenjem umjetne inteligencije.

Za potrebe ovoga rada izrađen je fizički model pametne kuće koji se sastoji od hardverskog i softverskog djela. Hardverski dio sastoji se od mikrokontrolera, senzora, aktuatora i svjetla montiranih na drvenu površinu. Glavni zadatak u softverskom djelu modela ima program Home Assistant koji je instaliran na mikrokontroler Raspberrry pi. Izradom modela dolazi se do zaključka da se sa relativnom malim financijskim sredstvima može automatizirati svoj dom no potrebno je puno vremena, istraživanja i isprobavanja. Nedostatak HA-a je u tome što je to program otvorenog koda te proizvođač ne može garantirati besprijekoran rad programa i omogućavati potpunu programsku podršku. Dobiva se dojam nedovršenosti programa. Dijelovi Hardverske opreme kao što su senzori ili aktuatori nemaju svoja kučišta te u lošijim atmosferskim uvjetima ne daju dovoljno točna i brza očitanja. Senzore je potrebno samostalno ožičiti što može predstavljati problem slabih kontakta ili kratkog spoja. Zaključak je da se na ovaj način može automatizirati dom samo u privatne svrhe sa ciljem učenja, no ne i u komercijalne svrhe. Kod novih kuća potrebna je ugradnja profesionalne hardverske i softverske opreme no takvu opremu prati i puno veća cijena.

Zadnji dio ovoga rada opisuje primjere korištenja umjetne inteligencije u pametnoj kući. Na neki način, pametna kuća se ni ne može nazvati „pametna“ ukoliko nema nekakav oblik umjetne inteligencije ugrađen u sebi. Problem kod ovog pristupa automatizaciji kuće je velika količina podataka i potreba za brzim procesiranjem istih što u većini slučaja glavni „hub“ uređaj kao što je mikrokontroler Raspberry pi ne može izvršiti. U to svrhu se preporučuju modeli komunikacije koji

omogućuju izvoz podataka na oblak sa kojeg može jače računalo preuzeti, analizirati i procesirati te vratiti podatke. Ako se puno vremena posveti analizirajući podatke te isprobavajući različite algoritme može se doći do veoma preciznih prognoza navika korisnika te je moguće optimizirati sustav pametne kuće.

Ovaj zadatak je uspješno izrađen uz pomoć znanja prikupljenog tokom studiranja, literature na internetu te naravno uz podršku i pomoć mog mentora. Rad može služiti kao osnova za buduće projekte u cilju izrade pametne kuće studentima i hobistima te je hardverski i softverski dio dovoljno razjašnjen te napisan jasno i razumljivo. Umjetna inteligencija je jedno veoma zanimljivo polje računalne znanosti te se sve više koristi u svim domenama ljudskih djelatnosti radi velike efikasnosti, brzine i alternativnom načinu rješavanju problema.

LITERATURA

[1] Hayes, A.: „Smart Home“, s Interneta, https://www.investopedia.com/terms/s/smarthome.asp, 14. rujna 2022.

[2] Singhania V.: „The Internet of Things: An Overview Understanding the Issues and Challenges of a More Connected World“, s Interneta, https://www.academia.edu/, listopad 2015.

[3] Iqra, Z.; Abu R.; Javed A.: „Internet of Things“, s Interneta, https://www.ijedr.org/papers/IJEDR1603172.pdf, rujan 2016.

[4] Odunlade E.: „What makes a Smart Home smart? A guide to protocols and applications“, s Interneta, https://www.wevolver.com/article/what-makes-a-smart-home-smart-a-guide-to-protocols-and-applications, 11. siječnja 2022.

[5] https://www.enciklopedija.hr/natuknica.aspx?id=63150, s Interneta, Hrvatska enciklopedija, mrežno izdanje. Leksikografski zavod, pristupljeno 22. rujan 2022.

[6] Escott E.: „What are the 3 types of AI?“, s Interneta, https://codebots.com/artificial-intelligence/the-3-types-of-ai-is-the-third-even-possible, 24. listopad 2017.

[7] Ghosh A.; Chakraborty D.; Law A.: “Artificial intelligence in Internet of things“, s Interneta, https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/trit.2018.1008, 14. studenti 2018.

[8] Brown S.; „Machine learning, explained“, s Interneta, https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained, 21. travanj 2021.

[9] Russel J. S.; Norvig P.: „Artificial Intelligence: A Modern Approach“, Prentice Hall, Upper Saddle River, 2010.

[10] Walch K.: „Application of AI in robotics boosts enterprise potential“, s Interneta, https://www.techtarget.com/searchenterpriseai/feature/Application-of-AI-in-robotics-boosts-enterprise-potential, 29. rujan 2020.

[11] Tyagi N.: „6 Major Branches of Artificial Intelligence (AI)“, s Interneta, https://www.analyticssteps.com/blogs/6-major-branches-artificial-intelligence-ai, 24. travanj 2021.

[12] https://canterbury.ai/using-ai-for-smart-homes/, s Interneta, 18. listopad 2021.

[13] Colman A.: „What is ESPHome, and what can it do?“, s Interneta, https://home-assistant-guide.com/2021/08/25/what-is-esphome-and-what-can-it-do, s Interneta, 25. kolovoz 2021.

[14] https://github.com/home-assistant/addons/blob/master/configurator/README.md, s Interneta, 11. srpanj 2020.

[15] https://opensource.com/resources/raspberry-pi, s Interneta.
[16] https://www.elprocus.com/esp8266-wi-fi-module/, s Interneta.
[17] Nugroho T.; Nasrun M.; Setianingsih C.: „Smart Lamp Control Based on User Behavior For Two Lamps Using K-Nearest Neighbour“, IEEE Xplore, 2019.
[18] https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_introduction.htm#, s Interneta.

Izvori slika

[1] Izvor: Autor
[2] Russel J. S.; Norvig P.: „Artificial Intelligence: A Modern Approach“, Prentice Hall, Upper Saddle River, 2010.
[3] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/, s Interneta.
[4] https://www.chipoteka.hr/, s Interneta.
[5] https://e-radionica.com/hr/, s Interneta.

POPIS SLIKA

SAŽETAK I KLJUČNE RIJEČI

U ovom diplomskom radu opisan je pojam pametne kuće i navedene su njene prednosti i nedostaci. Dan je kratak uvod u IoT tehnologije te su spomenuti komunikacijski modeli i protokoli kod pametnih kuća. Teorijski je obrađen kratak uvod u umjetnu inteligenciju te je opisano šest glavnih grana od kojih se sastoji. Izrađen je fizički model pametne kuće te su opisani svi njegovi softverski i hardverski dijelovi te način rada. U zadnjem djelu rada primijenjena je umjetna inteligencija na projektni zadatak te su isprobana tri različita algoritma umjetne inteligencije koji su pisani u python programskom jeziku.

Ključne riječi: pametna kuća, internet stvari, umjetna inteligencija, Home Assistant, python

In this thesis, the concept of a smart house is described and its advantages and disadvantages are listed. A brief introduction to IoT technology was given, and communication models and protocols for smart homes are mentioned. A short introduction to artificial intelligence is theoretically treated and the six main branches of which it consists are described. A physical model of the smart house was created and all its software and hardware parts and the mode of operation are described. In the last part of the work, artificial intelligence was applied to the project task and three different artificial intelligence algorithms were tested, which were written in the Python programming language.

Keywords: smart house, Internet of Things, artificial intelligence, Home Assistant, python