Arduino C programski jezik u mehatronici

Radolović, Patrik

Istarsko veleučilište
Università Istriana
di scienze applicate

Patrik Radolović

ARDUINO C PROGRAMSKI JEZIK U MEHATRONICI

Završni rad

Pula, 2023.

Istarsko veleučilište
Università Istriana
di scienze applicate

Patrik Radolović

ARDUINO C PROGRAMSKI JEZIK U MEHATRONICI

Završni rad

JMBAG: 0233008969, redoviti student

Studijski smjer: Preddiplomski stručni studij Mehatronike

Predmet: Osnove programiranja

Mentor: Marko Turk, dipl. oec., pred.

Pula, rujan 2023.

IZJAVA O SAMOSTALNOSTI IZRADE ZAVRŠNOG RADA

Ovom izjavom potvrđujem da sam završni rad pod nazivom “Arduino C programski jezik u mehatronici” napisao samostalno uz pomoć mentora Marka Turka, pred., primjenjujući znanje stečeno tijekom studiranja te korištenjem stručne literature koja je navedena na kraju rada. Završni rad je napisan u duhu hrvatskog jezika.

Student: *Patrik Radolović*

Potpis: _____

SAŽETAK

Kroz ovaj rad se nastoji olakšati shvaćanje Arduina i C programskog jezika za Arduino, i njihove primjena u mehatronici i automatizaciji. Mehatronika kao interdisciplinarno područje proizlazi iz elektrotehnike i strojarstva kao simbioza koja nastoji automatizirati monotone i repetitivne poslove. Kako bi se olakšala kontrola i mjerenje svih procesa potrebno je i centralno računalo ili mikro upravljač koji bi ih nadgledao, mjerio te reagirao ovisno o zadanoj logici. Kako bi se olakšao pristup, za taj zadatak odabrana je Arduino platforma i povezani Arduino C jezik. Povezavši sve navedeno komponente i sustav Internet stvari moguće je automatizirati svakodnevne uređaje i okolinu. Za realizaciju i prikaz primjena odabrana je jedna od aktualnih tema pametne kuće kao spoj svakodnevice i automatizacije u svrsi olakšavanja življenja.

Ključne riječi: Arduino, C programski jezik, mehatronika, pametna kuća, Internet stvari

SUMMARY

This work aims to facilitate the understanding of Arduino and the C programming language for Arduino and their application in mechatronics and automation. Mechatronics as an interdisciplinary field stems from electrical engineering and mechanical engineering as a symbiosis that seeks to automate monotonous and repetitive tasks. To facilitate the control and measurement of all processes, a central computer or micro-controller is also needed to monitor, measure and react depending on the given logic. To facilitate access, the Arduino platform and the associated Arduino C language were chosen for this task. By connecting all the above components and the Internet of Things system, it is possible to automate everyday devices and the environment. For the realization and display of applications, one of the current topics of the smart house was chosen as a combination of everyday life and automation to make life easier.

Keywords: Arduino, C programming language, mechatronics, smart house, internet of things

Sadržaj

POPIS OZNAKA I KRATICA

OZNAKA	OPIS	JEDINICA
U	Napon	V
I	Struja	A

KRATICA	OPIS
USB	Universal Serial Bus
TTL	Transistor-Transistor Logic
LED	Light Emitting Diode
GND	Ground
ISO	International Organization for Standardization
ANSI	American National Standards Institute
IoT	Internet of Things
SPI	Serial Peripheral Interface Communication
I2C	Inter-Integrated Circuit
LCD	Liquid Crystal Display
DHT	DHT (Digital Humidity and Temperature) Sensor
RFID	Radio-Frequency Identification
UART	Universal Asynchronous Receiver-Transmitter
RX	Receiver
TX	Transmitter
HTML	Hypertext Markup Language

1. Uvod

Kroz razvoj tehničkih znanosti kao i potreba za razvojem samih je bio da se smanji udio ljudskog rada u proizvodnji. Tijekom kontinuirane industrijske revolucije u 20. stoljeću pojavljuje se potreba za uvođenjem električnih motora i sustava i njihovo spajanje s čistim mehaničkim motorima koji su postojali od samih početaka osnovnih vrsta industrije. Tijekom 80-ih godina prošlog stoljeća počinje se upotrebljavati pojam mehatronike kao interdisciplinarnog područja, najčešće u područjima automatizacije. Svaka vrsta automatizacije kao središnji dio mora sadržavati neki oblik centralnog računala za obradu podataka, raznih senzorskih parametara položaja, temperature i ostalih parametara kao i dio sustava namijenjen za kontroliranje aktuatora na temelju ugrađene, željena logike. Naravno to nisu jedini načini korištenja centralnog računala, ali su neki koji su najbliži čovjeku i koje sam čovjek može promatrati. S vremenom su takvi upravljači spali pod neizbježni utjecaj minijaturizacije zbog samog napretka u tehnologiji i potrebe za kompaktnim sustavima. Paralelno s razvojem računala dolazi i do razvoja računalnih jezika kao vrsta komunikacije između čovjeka i računala. Komercijalizacijom industrijskih upravljača dolazi i do njihovog širenja prema amaterskim razinama gdje hobisti i amateri počinju izrađivati svoje izvedbe i interpretacije istih. Tijekom ranih godina 21. stoljeća platforma Arduino zauzima svoje mjesto kao jedna takva izvedba upravljača sada znanih kao mikro upravljači. Kao što i sama riječ govori radi se smanjenoj verziji standardnih upravljača. Da bi se komuniciralo mikroupravljačem i programiralo mikroupravljač bilo je potrebno uspostaviti računalni jezik u tu svrhu. Kako ne bi došlo do komplikacije i potrebe za stvaranjem novog jezika uzeta je kombinacija postojećeg jezika C i njegove nadogradnje C++. Uzimanjem pojedinih funkcija, namjena i pojedinih svojstava dobiva se namjenski jezik Arduino C. Sam jezik koristi i podržani program Arduino IDE u svrsi prevođenja i slanja željenog koda na mikro upravljač. Spojem Arduina i elektroničkih elemenata poput senzora, LED dioda i aktuatora poput istosmjernih motora postiže se osnovna razina nekog automatiziranog sustava koji ovisno o očitanjima senzora izvodi željene radnje. U ovom radu pokušat će se detaljnije pojasniti sam pojam Arduina i osnovnih izvedi same platforme te Arduino C programskog jezika i programiranje tim jezikom. Kasnije se pojašnjuje sam pojam mehatronike i interneta stvari kao vrsta

tehnologije za integriranje postojećih sustava u jedan zajednički, s dvosmjernom komunikacijom. Kao primjer integracije odabran je praktični projekt pametne kuće s namjenom da se pokaže kako spoj pojedinih komponenata može funkcionirati u zatvorenom sustavu na bazi Arduino platforme koja za programiranje koristi Arduino C programski jezik.

2. Arduino

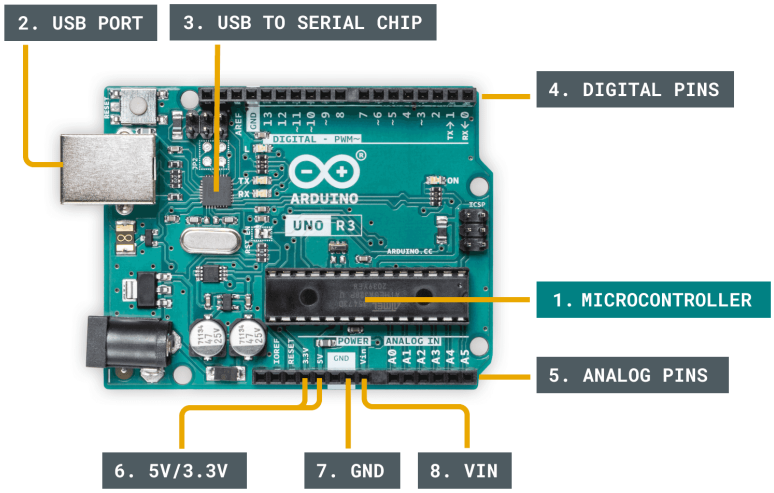
Arduino je otvorena programerska platforma za elektroniku koja je jednostavna za upotrebu i upotrebljava se u edukaciji, izradi prototipova a na koncu može koristiti za sve vrste projekata, automatizacije i kontrole raznih sustava. Neki od tih sustava temelje se na senzorima za očitavanje parametara okoline, kontrolu navodnjavanja ili upravljanje robotima i dronovima. Zbog pristupačnosti i dostupnosti popratne opreme postaje neizostavni dio amaterskih i edukativnih namjena za uvod u područje programiranja i automatizacije. Sastoji se od fizičkog dijela (pločice s mikro upravljačem) i računalnog programa za programiranje i učitavanje željenog koda na fizički dio (Arduino, 2018).
Arduino je na tržište stigao 2005. godine nakon dugogodišnjeg razvoja na institutu za interaktivni dizajn Ivrea u Italiji (Arduino, 2023).

Slika 1: prikaz Arduino logotipa

Izvor: https://commons.wikimedia.org/wiki/File:Arduino_Logo.svg

2.1. Fizički dio

Fizički dio Arduina je skup od nekoliko komponenti koji čine okruženje za sam Atmel mikro upravljač s popratnom memorijom. Na slici 2 mogu se razaznati pojedini dijelovi same pločice mikro upravljače. Za prijenos i komunikaciju s mikro upravljačem služi USB priključak. Zatim imamo čip za pretvorbu podatka iz formata serijske komunikacije USB u jednostavniji serijski format sličniji starijem RS232 ili TTL protokolu. Arduino ne bi bio Arduino bez njegovih ulazno/izlaznih priključaka koji služe za spajanje raznih aktuatora kao što su servo motori, paljenje ili gašenje LED dioda i raznih drugih. Ulazno/izlazne priključke dijelimo na digitalne i analogne, po vrsti i svojstvima. Digitalni nam označavaju upravljanje binarnim vrijednostima 0 ili 1. Primjer upotrebe digitalnih izlaza je jednostavno paljenje i gašenje LED dioda ili upravljanje servo motorom pomoću PWM modulacije. Kod analognih priključaka omogućeno nam je čitanje i pisanje analognom vrijednošću u rezoluciji od 10 bitova, počevši od 0 do 1023. U konačnici se tu još nalaze naponski izlazi od 5V i 3.3V, kao i zajednički negativni iliti GND, tipka za resetiranje, te priključak za napajanje kada Arduino nije povezan putem USB-a (Arduino, 2023).



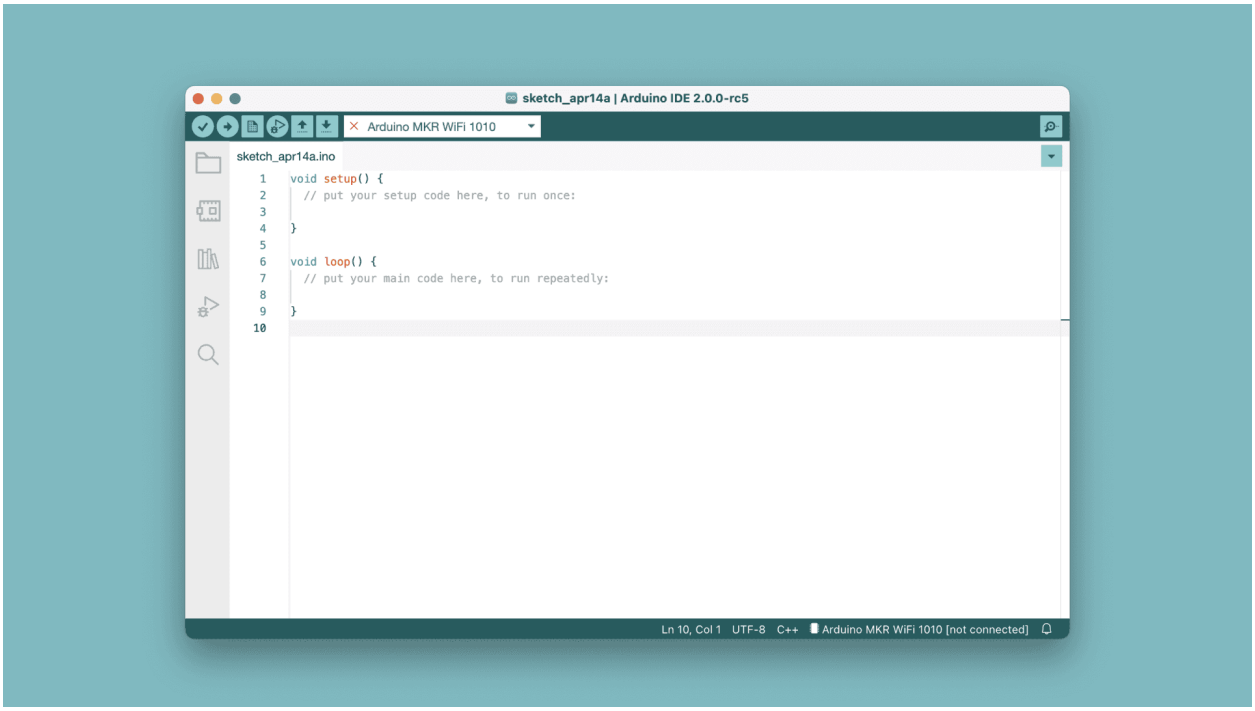
Slika 2: Topologija Arduino pločice

Izvor:

https://docs.arduino.cc/static/d0c28c5bd0894792476c6052dea5fa63/29114/board-anatomy.png

2.2. Arduino IDE

U svrhu pisanja koda, prevođenja i slanja istog na mikro upravljač potreban nam je računalni program Arduino IDE. IDE je skraćenica za Integrated Development Environment, u prijevodu integrirano razvojno okruženje. Program nam omogućuje pisanje željenog koda, njegovo prevođenje i slanje na željeni uređaj. Uz program dolaze osnovni primjeri kodova za razne upotrebe i serijskim monitorom koji služi za dvosmjernu komunikaciju između Arduina i korisnika (Arduino, 2023).



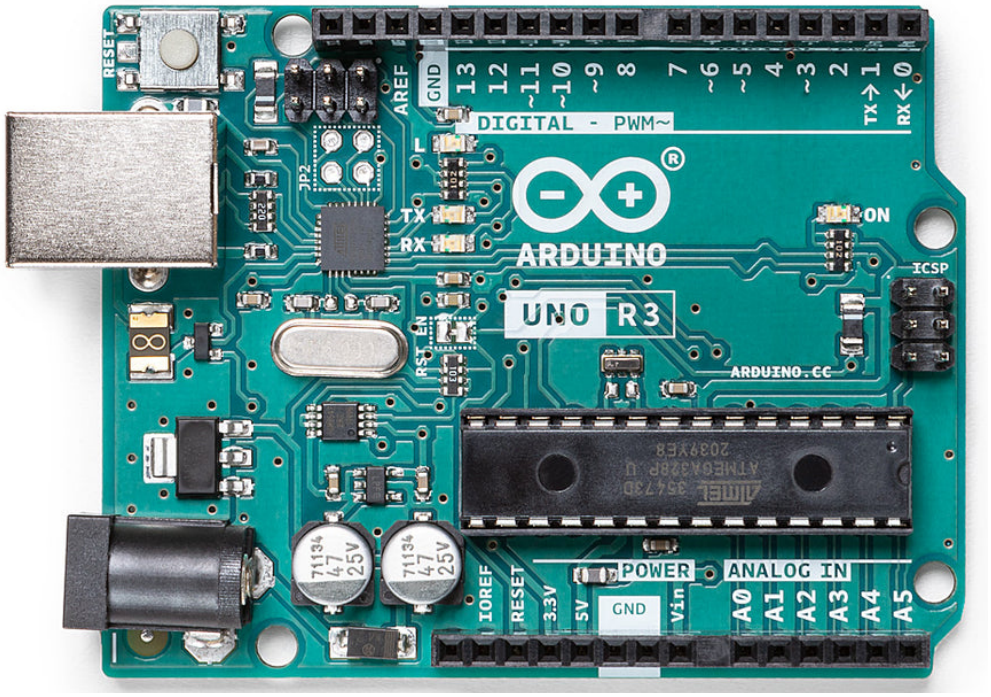
Slika 3: Prikaz Arduino IDE osnovnog okruženja

Izvor: https://docs.arduino.cc/static/26f69dc51d5f05fab1cfa6456f0b2d14/29114/ide-2.png

2.3. Arduino uređaji

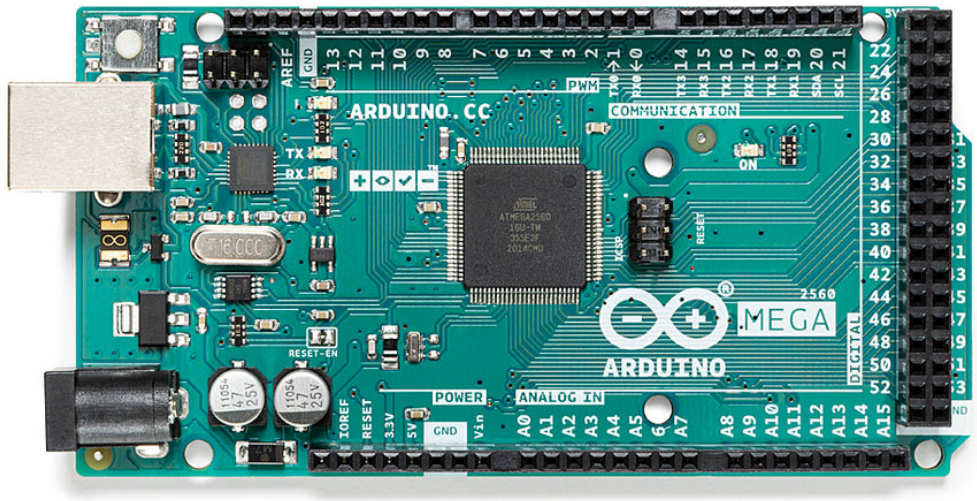
Obitelj Arduino uređaja je dostigla veliku raznovrsnost i namjenu. Glavne razlike su u vrsti mikro upravljača, veličini memorije, broju ulazno/izlaznih priključaka. Jedan od najzastupljenijih je Arduino Uno (Slika 4) kao platforma srednje veličine i dostupnog broja priključaka, prate ga Arduino Mega (Slika 5) i Arduino Nano (Slika 6). Kao što samo ime govori, Arduino Mega je najveći uređaj s najvećim brojem priključaka, većom memorijom

i mikro upravljačem većih mogućnosti. Nano je najmanji i samim time ima najslabije mogućnosti od ostalih (Arduino, 2023).



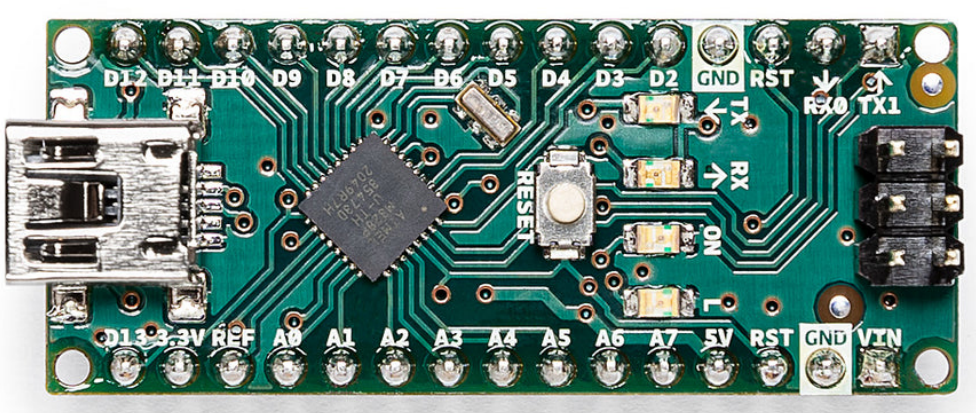
Slika 4: Prikaz Arduino Uno pločice

Izvor: https://store.arduino.cc/products/arduino-uno-rev3



Slika 5: Prikaz Arduino Mega pločice

Izvor: https://store.arduino.cc/products/arduino-mega-2560-rev3



Slika 6: Prikaz Arduino Nano pločice

Izvor: https://store.arduino.cc/products/arduino-nano

Tablica 1: Usporedba različitih izvedba Arduino pločica i njihovih karakteristika

Pločica		Arduino Uno R3	Arduino Nano	Mega2560 Rev3
Mikrokontroler		ATmega328p	ATmega328p	ATmega2560
Vrsta USB-a		USB-B	Mini-B USB	USB-B
Ulazno/izlazni priključci	Digitalni	14	14	54
	Analogni ulaz	6	8	16
	Analogni izlaz	0	0	0
	PWM	6	6	15
Komunikacija	UART	Da	Da	Da
	I2c	Da	Da	Da
	SPI	Da	Da	Da
	CAN	Ne	Ne	Ne
	Bluetooth	Ne	Ne	Ne
	WIFI	Ne	Ne	Ne
Napajanje	U/I napon (V)	5	5	5
	Nominalni ulazni napon	7 - 12	8 - 12	9 - 12
	Struja po U/I priključku (mA)	20	20	20

	Vrsta priključka napajanja	5.5x2.5 priključak(Barrel Jack)	GPIO priključak	5.5x2.5 priključak(Barrel Jack)
	Baterijsko napajanje	Ne	Ne	Ne
Brzina procesora (MHz)		16	16	16
USB u serijski pretvarač		ATmega16U2 16MHz	ATmega16U2 16MHz	ATmega16U2 16MHz
Memorija	Flash (KB)	32	32	256
	SRAM (KB)	2	2	8
	EEPROM (KB)	1	1	4
Dimenzije	Masa (g)	25	5	37
	Širina (mm)	53.4	18	53.3
	Dužina (mm)	68.6	45	101.5

Izvor: https://circuitdigest.com/article/different-types-of-arduino-boards

3. Arduino programski jezici

Arduino platforma se temelji na nekoliko programskih jezika. Programski jezici su C i C++. C jezik je definiran od strane American National Standard Institute (ANSI) 1983. g. Kao standard pojavljuje se tek 1989.g.. Postoje dva naziva za C jezik, ANSI C i ISO C. U samom jeziku nema razlike, jedina razlika je u nazivu. C programski jezik koji se koristi za Arduino platformu je pod verzija C-a jer se razlikuje po nekim elementima. U nastavku ćemo tu verziju C-a nazivati Arduino C kako ga naziva i Purdum iz svoje knjige iz 2015. godine kako ne bi došlo do zabune. Njihova razlika ne utječe na funkcionalnost jezika, jer su oba sposobna izvršavati iste zadatke i funkcije. Jedna od većih razlika između Arduino C i običnog C-a je u samom prevoditelju. Prevoditelj za Arduino C je ustvari Open Source C++ prevoditelj, kao što je GCC (GNU Compiler Collection). Isto tako gotovo sve biblioteke koje se koriste u Arduino su napisane u C++ jeziku. Kod pisanja kodova slobodno se mogu miješati C i C++ programski jezik jer se većina pozadinskih aplikacija Arduina vrti u C++ (Purdum, 2015).

3.1. C programski jezik

C je proceduralni programski jezik opće namjene. Kvalitete koje su jezik C učinile idealnim za takvu funkciju su kompatibilnost s otvorenim kodom, sposobnost rada "blizu“ samog hardvera i njegove performanse. Osnovni elementi C programa su funkcije koje mogu referencirati jedna drugu. U dobro organiziranom programu svaka od funkcija ima određenu svrhu. Funkcije sadrže izjave koje se sekvencijalno izvršavaju kodom, a izjave se mogu kombinirati u blokove. Korisnik može koristiti pripremljene, gotove, funkcijske biblioteke ili napisati vlastite ako nijedna od standardnih funkcija ne odgovara zahtjevima. Kao dodatak standardnoj C biblioteci, dostupne su dodatne specijalne biblioteke, poput grafičkih funkcija. Ipak, korištenjem takvih nestandardnih biblioteka ograničava se prilagodljivost programa na sustave kojima su biblioteke zapravo prilagođene. Svaki C program trebao bi definirati i deklarirati svaku funkciju zasebno, a jednu s posebnim nazivom *main()*. To je početna funkcija koja se izvodi nakon što program započne s radom. Funkcija *main()* je najviša razina programa i može pozivati druge potprograme. Bitno je napomenuti da kompajler zahtijeva da se svaka funkcija prvo definira. Programski kod C-a se sastoji od definicija funkcija, globalnih deklaracija i pravila predprocesiranja. Kod za male programe sastavljen je u jednoj datoteci dok se veći C programi sastoje od više izvornih kodova. Kako se definicije funkcija često temelje na uputama predprocesora i globalnim deklaracijama, izvorne datoteke obično imaju istu strukturu:

1. Upute predprocesora
2. Globalne deklaracije
3. Definicije funkcija

Redoslijed definiranja nije bitan. Definicija same funkcije ne može biti definirana unutar druge, može se samo definirati lokalna varijabla unutar funkcije. C omogućuje kategorizaciju s onoliko izvornih datoteka i datoteka zaglavlja koje su potrebne i koje se mogu zasebno uređivati i sastavljati. Svaki izvorni kod često uključuje povezane zadatke. Ekstenzija datoteke koja se često koristi za označavanje C izvornih datoteka je .c (Prinz, Crawford, 2005).

3.2. C++ programski jezik

C++ je programski jezik koji se koristi za razvoj softvera. To je objektno orijentirani programski jezik što bi značilo da se tijekom programiranja stvaraju objekti koji sadrže varijable i funkcije, ti objekti se mogu nadovezivati jedan na drugog, imaju mogućnost inkapsulacije i povećanja sigurnosti. Isto tako s objektima olakšava se organizacija i stvaranje hijarhije unutar programskog koda. Drugim riječima, naglašava korištenje podataka s jedinstvenim atributima umjesto logike i funkcija. C++ je 1979. izumio Bjarne Stroustrup kako bi proširio C jezik. Razvijen je kako bi programerima pružio mnogo veću fleksibilnost memorije i resursa sustava. C++ se ubrzano proširio kao programski jezik za stvaranje brzih, pouzdanih programa sa širom namjenom. Zbog svoje fleksibilnosti, on savršeno odgovara složenim aplikacijama, sustavnim uređajima i opremi za internet stvari (IoT). C++ programiranje nudi brojne prednosti u vidu programiranja, sigurnosti i pristupa samom jeziku zbog njegove opširnosti i namjene. Kôd se može jednostavno organizirati i kategorizirati budući da je objektno orijentiran i možete reciklirati kod umjesto da ga ponovno pišete ispočetka (Coursera, 2023).

3.3. Usporedba C i C++ jezika

Tablica 2: Usporedba programskih jezika

Parametar	C	C++	Arduino C
Programska paradigma	strukturni ili proceduralni programski jezik.	strukturni kao i objektno orijentirani programski jezik	strukturni kao i objektno orijentirani programski jezik
Povijest	C je razvio znanstvenik Dennis Ritchie 1972. u Bell Laboratories.	C++ je razvio Bjarne Stroustup 1979.	M. Banzi, D. Cuartielles, T. Igoe, G. Martino, D. Mellis 2005.
Pristup	odozgo prema dolje	odozdo prema gore	odozdo prema dolje
Ključne riječi	32 ključne riječi	63 ključne riječi	63 ključne riječi

Vrste podataka	C podržava ugrađene tipove podataka.	C++ podržava i ugrađene i korisnički definirane tipove podataka.	Arduino C podržava i ugrađene i korisnički definirane tipove podataka.
Vrsta ekstenzije datoteke	.c	.cpp	.ino
Datoteke zaglavlja	<stdio.h>	<iostream.h>	<stdio.h>
Dodjela i dealokacija memorije	calloc() i malloc() za dodjelu, a free() za dealokaciju memorije	Koristi se novi operator za dodjelu, operator za brisanje kod oslobađanja	Ovisno o namjerni mogu se koristit karakteristike C i C++
Modifikator pristupa	nije podržan	podržan	podržan
Sigurnost	Nije siguran, lako se manipulira izvana	Povećana sigurnost, nudi enkapsulaciju i skrivanje podataka	Kod se može jedino duplicirat fizičkim kopiranjem mikrokontrolera
Referentna varijabla	Nije podržano	Podržava	Podržava
Preopterećenje funkcija i nadjačavanje funkcija	Ne podržava	Podržava	Podržava
Rukovanje iznimkama	Ne podržava izravno nego samo preko funkcija	Izrazno podržava uz pomoć try-catch bloka	Ne podržava izravno nego samo preko funkcija

Programski odjel	Kod je podijeljen u zasebne blokove kao funkcije	Podjela na klase i objekte	Ovisno o namjerni mogu se koristit karakteristike C i C++
"Inline" funkcija	Ne podržava	Podržava	Podržava
Pogonski tip jezika	Funkcijski pogonjen	Objektivno pogonjen	Ovisno o namjerni mogu se koristit karakteristike C i C++
Kompatibilnost	Može se izvoditi i na C++ kompajleru jer je C temeljni jezik	Može se izvoditi i na C kompajleru jer je C++ uključuje OOP koncept	Izvodi se na C++ kompajleru (GCC)
Podaci i funkcija	Podaci i funkcije su odvojeni zbog proceduralnog jezika	Podaci i funkcija su enkapsulirani zbog objektivno orijentiranog jezika	Ovisno o namjerni mogu se koristit karakteristike C i C++
Ulazna/Izlazna funkcija	scanf() kod ulaza, printf() kod izlaza	cin kod ulaza, cout kod izlaza	scanf() kod ulaza, printf() kod izlaza
Razvoj aplikacija	Prikladan za nisku razinu, npr. Uređivač teksta	Prikladan za visoku razinu, npr. razvoj video igara, pametni telefoni	Prikladan za razvoj tehnologije, automatizacija, robotika itd.
Imenski prostor	Ne podržava, dolazi do kolizije koda	Podržava	Podržava
Korisnici	MySQL, Windows Kernel, Oracle Database itd.	Google Chrome, Torque 3-D game, Microsoft Office itd.	Edukacijske ustanove, amateri, inovatori i ostali

Izvor: https://www.mygreatlearning.com/blog/difference-between-c-and-c/#:~:text=from%20C%20language%3F-,C%20is%20a%20structural%20or%20procedural%20programming%20language%20that%20was,%2C%20Inheritance%2C%20Polymorphism%2C%20etc.

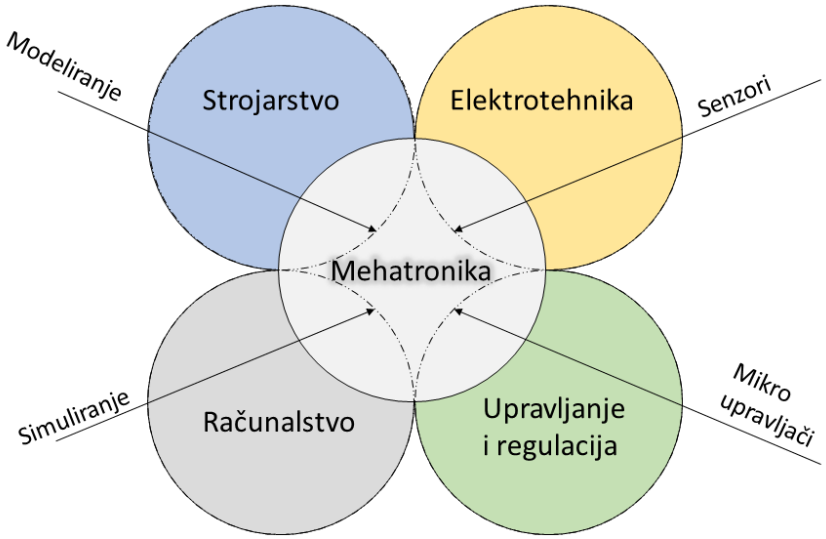
!Tablica je preuzeta jednim dijelom ali sam je ja preveo i nadopunio!

3.4. GCC prevoditelj

GCC se odnosi na "GNU Compiler Collection." GCC je integrativna zbirka prevoditelja za brojne programske jezike. Ti jezici su C, C++, Objective-C, Objective-C++, Fortran, Ada, D i Go. U uobičajenoj upotrebi, izraz GCC ima mnogo tumačenja. Trenutačno službeno značenje je "GNU Compiler Collection," što se odnosi na kompletan skup alata. Izvorno je naziv označavao "GNU C Compiler", a ova je upotreba još uvijek uobičajena kada je fokus na generiranju C koda. Sama srž GCC-ovih optimizatora neovisna je o jeziku, jer je u suštini podijeljen na "back end" i "front end". Podjela je uvedena da svaki dio može može biti korišten različitim programskim jezikom omogućavajući ponovnu uporabu napisanog koda (GCC, 2023).

4. Primjena Arduina u mehatronici

"Mehatronika je interdisciplinarna grana koja kombinira elemente strojarstva, elektroničkog inženjerstva i računalne znanosti za dizajniranje inteligentnih sustava kojima se može upravljati pomoću računala. Mehatronički sustavi često su složeni sustavi koji uključuju mehaničke komponente, poput motora i senzora, kao i elektroničke komponente, kao što su mikroprocesori i senzori. Cilj mehatronike je dizajn i razvoj integriranih sustava koji su pouzdani, učinkoviti i isplativi" (Bolton, 2015., str. 1).



Slika 7: Vennov dijagram s prikazom sastavnih grana mehatronike

Izvor: https://commons.wikimedia.org/wiki/File:MechatronicsDiagram.svg

Kako bi se što lakše integrirani sustavi u mehatroniku potrebno je uklopiti i mikroprocesor, računalo koje će izvoditi zadatke i obradu podataka a u ovom radu za tu namjenu je odabran Arduino. Arduino kao razvojna platforma lako se implementira u sustave automatizacije, kontrole procesa i drugih inačica mehatronike. Omogućuje lako povezivanje sa senzorima i aktuatorima kao i s IoT tehnologijom i funkcijama. Velik broj uzlazno/izlaznih priključaka omogućuje obradu analognih i digitalnih signala što je od ključnog značenja za uspostavu samostalnog mehatroničkog sustava. Primjer jednog sustava je pametna kuća koja koristi i na desetke senzora, elektro motora, releja i drugih elektroničkih komponenti kako bi se ostvarila njena namjena. Arduino u mehatroničkim sustavima u današnje doba predstavlja početnu amatersku razinu na kojoj se lako uči i usavršava prilikom čega se dobiva potrebno početno znanje za razvoj složenijih sustava. Mehatronički sustavi nisu samo ograničeni na elektroniku nego i na strojarske elemente kao što su pneumatski i hidraulički aktuatori koji pomoću elektro-ventila ostvaruju određene zadatke. U sve većem napretku može se očekivati da će sustavi postati sve kompleksniji, a isto tako lakši za korištenje zbog pojave umjetne inteligencije u raznim

integriranim sustavima današnjice kao što su autopiloti u automobilima i zrakoplovima. Isto tako i sami automobili koji su u potpunosti autonomni posjeduju razne senzore i aktuatore koji im omogućuju snalaženje u prostoru i donošenje odluka prema ulaznim podacima.

4.1. Internet stvari (IoT)

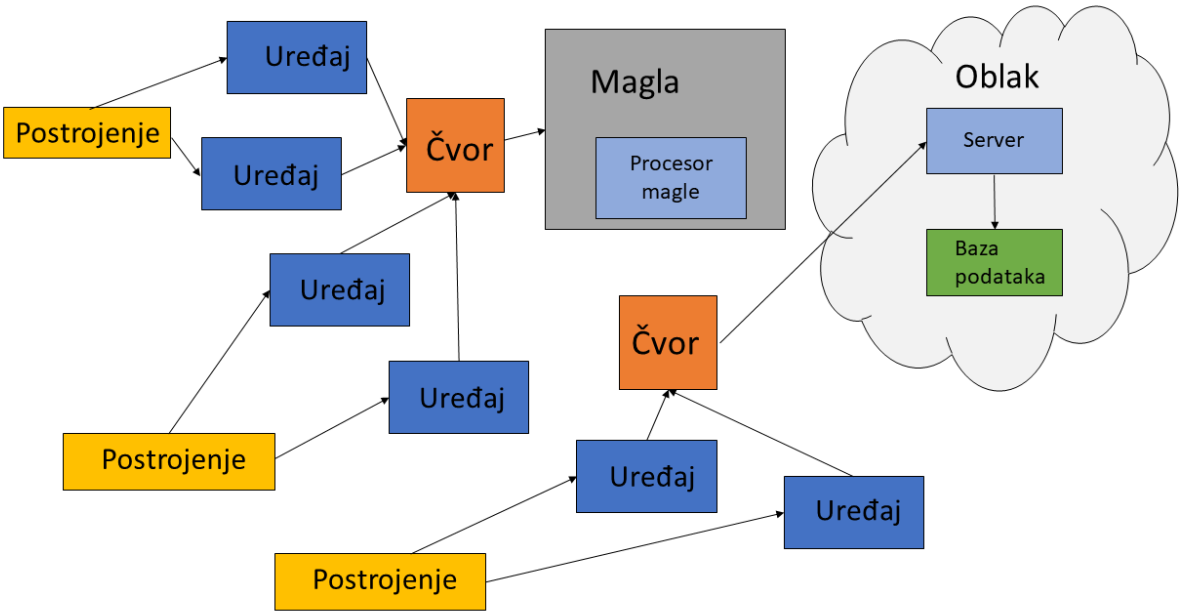
Izraz "Internet of Things" (IoT) uspostavio je Kevin Ashton 1999. dok je istraživao u MIT-ovom Media Centru. Želio je da to predstavlja koncept računala i strojeva opremljenih senzorima koji se povezuju na Internet za prijenos statusa i prihvaćanje uputa (Norris, 2015).

Internet stvari (IoT) postale su uspostavljena medijska tema i poslovni trend. Unatoč svemu, IoT se pojavio kao razvoj novih tehnika s brojnim primjenama. Internet stvari izgrađene su na više već postojećih tehnologija, posebice globalnim podatkovnim sustavima, senzorskim sustavima i računalima. Termin IoT sustav, umjesto Internet of Things, točnije opisuje kako se ovaj sustav koristi. Većina uređaja povezana je zajedno kako bi oblikovala funkcionalna rješenja dok se rjeđe koriste kao ukupni uređaji na svjetskoj mreži. Osim toga, umjesto da bude skupina uređaja s internetom, sustav je prilagođen s jednim ili nekoliko zadataka. IoT uzima u obzir karakteristike složenih procesa. IoT sustav prvenstveno može biti napravljen od senzora, ali može sadržavati i znatan broj aktuatora ovisno o situacijama. Svrha u oba slučaja je analiza signala i informacija kroz vrijeme.

Namjene ovakvih sustava mogu biti od jednostavnih kućanskih aparata i sve do gradskih sustava. IoT u kućnoj primjeni pridonosi štednji tekućih troškova kao i štednji vremena prilikom obavljanja kućanskih poslova a takve kuće mogu poprimit naziv i pametna kuća. Kako bi sustav u gradovima mogao doprinijeti svim građanima postavljeni su ciljevi nadgledanja prometa i pješaka, energetskoj efikasnosti zgrada i postavljanu solarnih farmi kao pasivan izvor proizvodnje energije .

Arhitektura IoT sustava se temelji na raspodjeli i međusobnoj povezanosti uređaja od samog korisnika do magle i baze podataka. Kao što je prikazano na Slici 7 interakcija između postrojenja i uređaja je interakcija čovjeka i ulaznog uređaja u sustav. Sami uređaji nemaju veliku povezanost pa se zatim spajaju na lokalna čvorišta gdje se

informacija prosljeđuju u željenom pravcu. Magla i procesor magle se najčešće mogu pronaći blizu samih čvorišta zbog smanjenja latencije. Oblak i baza podataka služe kao što i samo ime navodi za pohranu podatka kao i kod drugih vrsta tehnologija (Serpanos & Wolf, 2018).



Slika 8: Prikaz arhitekture Interneta stvari

Izvor: Serpanos, D., Wolf, M. (2018). Internet-of-Things (IoT) Systems, str. 8

4.2. Pametna kuća

Pametna kuća je složeni sustav međusobno povezanih uređaja i samog stambenog objekta s ciljem energetske efikasnosti i smanjenja emisija stakleničkih plinova. Kako bi se što lakše opisalo značenje pametne kuće potrebno ju je usporediti s tradicionalnim kućama. Kod tradicionalnih kuća nije moguće pomoću upravljača otvoriti vrata ili ugasiti rasvjetu, pokućni uređaji rade zasebno i nemaju mogućnosti međusobne komunikacije. Kod pametne kuće dolazi do povezivanja uređaja u svrhu održavanja temperature, sigurnosti, automatizacije vrata i prozora, osvijetljena i mnogih drugih. Ova vrsta automatizacije kao rezultat daje smanjenje ljudskih potreba u kućanstvu, smanjenje troškova i veću sigurnost ali sve to je nemoguće postići bez ljudskog utjecaja na projektiranje i programiranje jer uređaji nisu pametni niti nemaju mogućnost međusobnog

povezivanja bez ljudskog faktora. Na koncu, pametna kuća može obavljati na stotine funkcija nakon završetka njene izrade, neke su navedeno u prijašnjem tekstu a mogu se još nadodati pametni uređaji kao što su frižideri, mobilni roboti za usisavanje i pranje podova, vanjska rasvjeta i po mogućnosti elementi obnovljivih izvora, tj. solarni sustavi i drugi.

Slika 9: Dizajn pametne kuće s prikazom pametnih sustava

Izvor: https://www.techtarget.com/iotagenda/definition/smart-home-or-building

5. Programiranje Arduino pametne kuće

5.1. Problemski zadatak

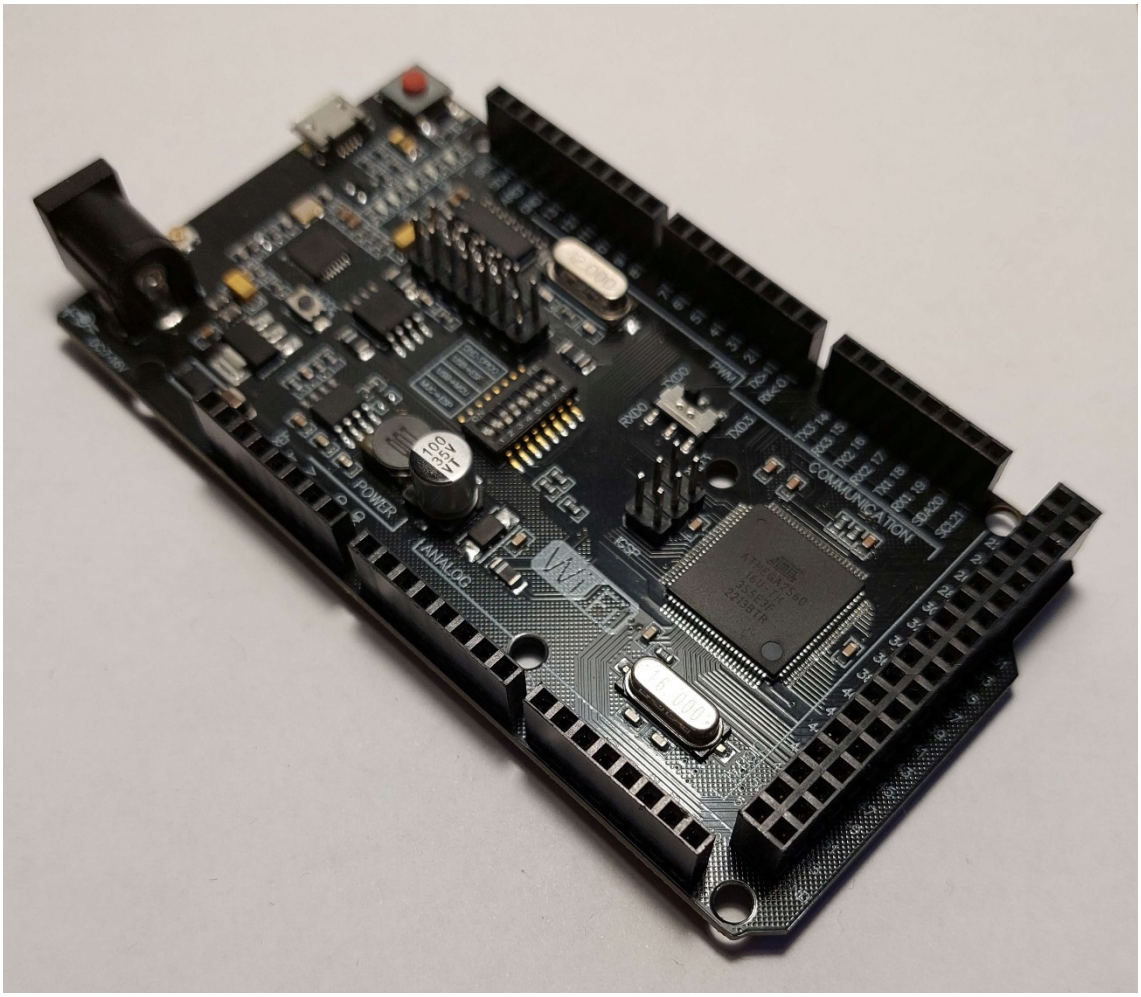
Sve se više pažnje pridaje uštedi energije, emisiji stakleničkih plinova u svakodnevici upućujući na energetsku efikasnost. Tu nastupaju pametne kuće koje pružaju dobra rješenja za predstavljene probleme. Dok rješavaju navedene probleme, preostaju problemi koji su vezani uz same financije. Sustavi koji su orijentirani prema kućnoj automatizaciji mogu postizati visoke cijene, zahtijevati specijalizirane stručne osobe za projektiranje kao i dodatni problem čekanja za izvršenje usluge adaptacije. Pitanje preostaje da li se neka vrsta jednostavne automatizacije može samostalno ostvariti i implementirati kao neki amaterski projekt. Neke od funkcija koje se mogu pronaći u pametnoj kući već su prije navedene dok će se u ovom radu pažnja posvetiti sustavu klimatizacije, sigurnosti i rasvjete.

5.2. Predloženo rješenje

Uz svakodnevni pristup internetu moguće je prikupiti sve potrebne informacije za samostalnu izradu i unaprjeđenje vlastitog doma u pametnu kuću a svaki objekt se može unaprijediti po potrebi i namjeni. Kako bi se sustav projektirao, prvo se mora definirati namjena i funkcije a zatim odrediti i komponente te načini komunikacije elemenata sustava. Za odabrani sustav korištena je Arduino platforma zbog svoje jednostavnosti i pristupačnosti. Isto tako programiranje upravljača i svih funkcija ostvareno je pomoću Arduino C programskog jezika. Funkcije sustava koje će se obraditi su grijanje i hlađenje, otvaranje i zatvaranje prozora, detekcija kiše, upravljanje rasvjetom i sama sigurnost doma. Da bi olakšali pristup sustavu, sve bitne informacije i kontrole nalazit će se na lokalnom poslužitelju na kojeg se povezuje mrežom i putem web tehnologije. Ovom metodom se može pristupiti podacima a također i kontrolirati sam sustav s bilo koje lokacije. Pomoću Arduinovih ulaza i izlaza povezati će se senzori, releji i elektro motor koji će služiti za ranije navedene funkcije u problemskom zadatku. U nastavku se nalazi popis komponenti uključenih u ovom projektu kao i programski kod, te na kraju i fizička realizacija samog projekta.

5.3. Komponente

1. Arduino Mega + ESP8266 - ATmega2560 mikro upravljač temeljen na Arduino Mega platformi koji dolazi s ugrađenim ESP8266 procesorom koji omogućava mrežno povezivanje ethernet protokolom za stvaranje vlastite lokalne mreže. Pruža različite načina rada ovisno o primjeni, kao što je Arduino Mega zasebno, ESP8266 zasebno ili zajednički omogućava komunikaciju putem serijske veze. Karakteristike Mega platforme već su ranije navedene, a ESP se samo nadovezuje na iste.

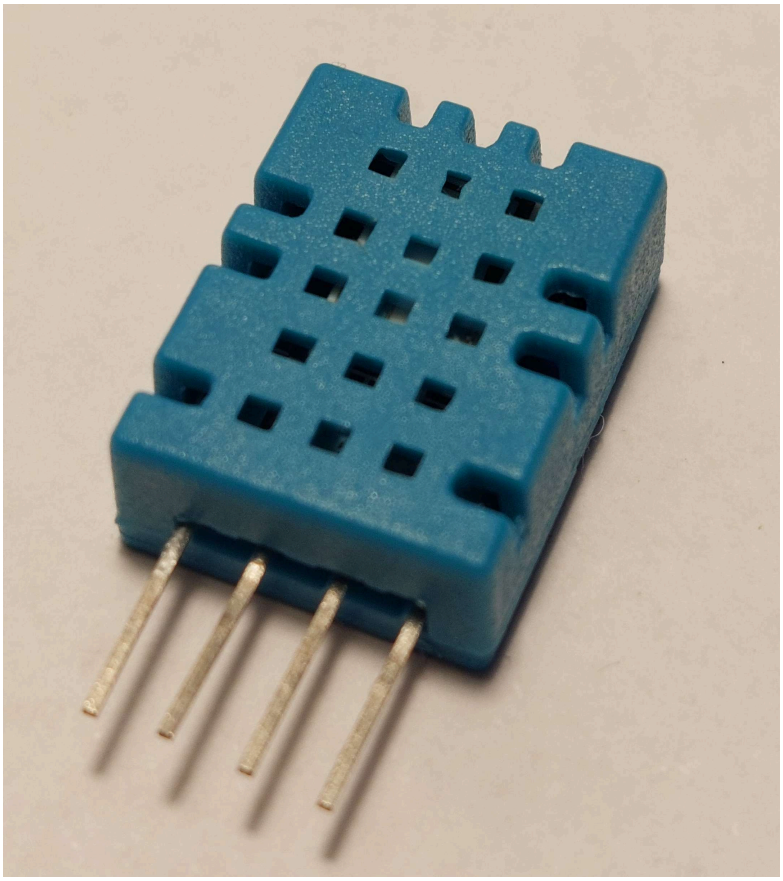


Slika 10: Arduino Mega mikro upravljač s ESP8266

Izvor: Autor

2. DHT11 senzor – senzor koji služi za očitavanje vrijednost temperature zraka i relativne vlažnosti. Razina napona potrebna za napajanje iznosi 3 do 5V.

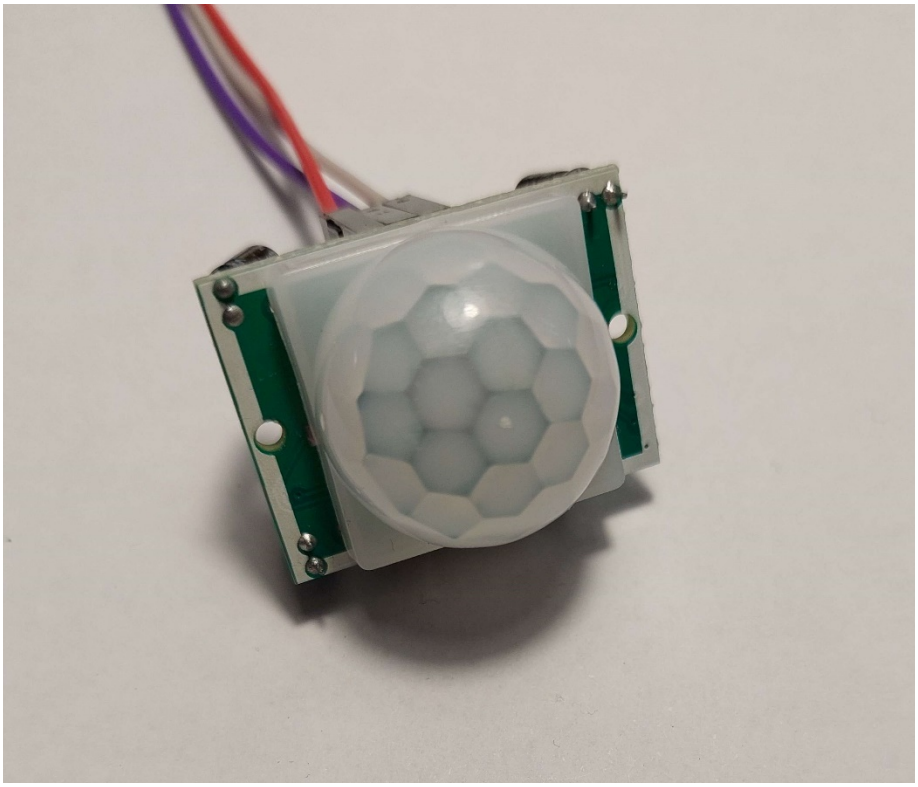
Komponenta je relativno malih dimenzija što omogućava integraciju u složene sustave i ugradnju u mala kućišta (Soldered Electronics, 2023).



Slika 11: DHT11 senzor temperature i relativne vlažnosti zraka

Izvor: Autor

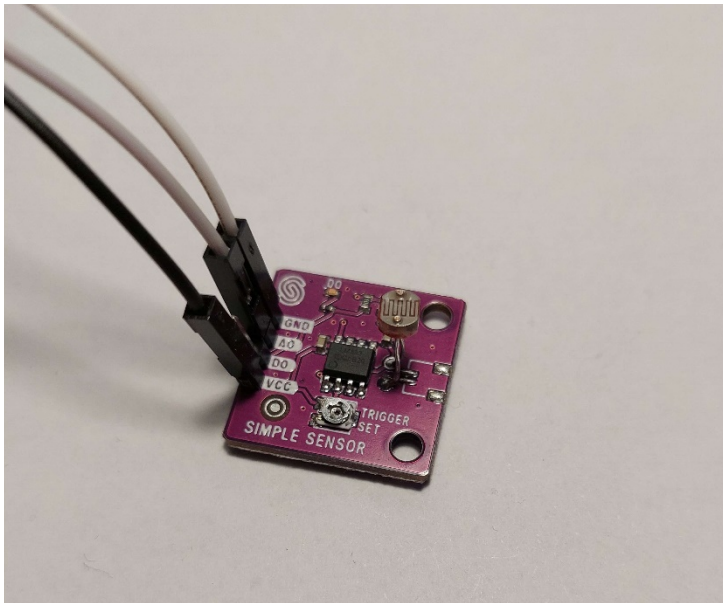
3. PIR senzor – senzor koji se koristi za podsustav rasvjete ili kao sigurnosni element zbog njegove karakteristike da detektira pokret u prostoru. Drugog naziva HC-SR501, ovaj senzor može detektirati pomake do 7m udaljenosti s kutom detekcije od 120 stupnjeva. Pomoću dva potenciometra podešava se osjetljivost senzora kao i dužina trajanje signala kada se senzor upali, tj. detektira pomak. Ulazni napon može biti između 5 i 20V, sa strujom jačine 60 μA (Soldered Electronics, 2023).



Slika 12: PIR senzor za detektiranje pokreta

Izvor: Autor

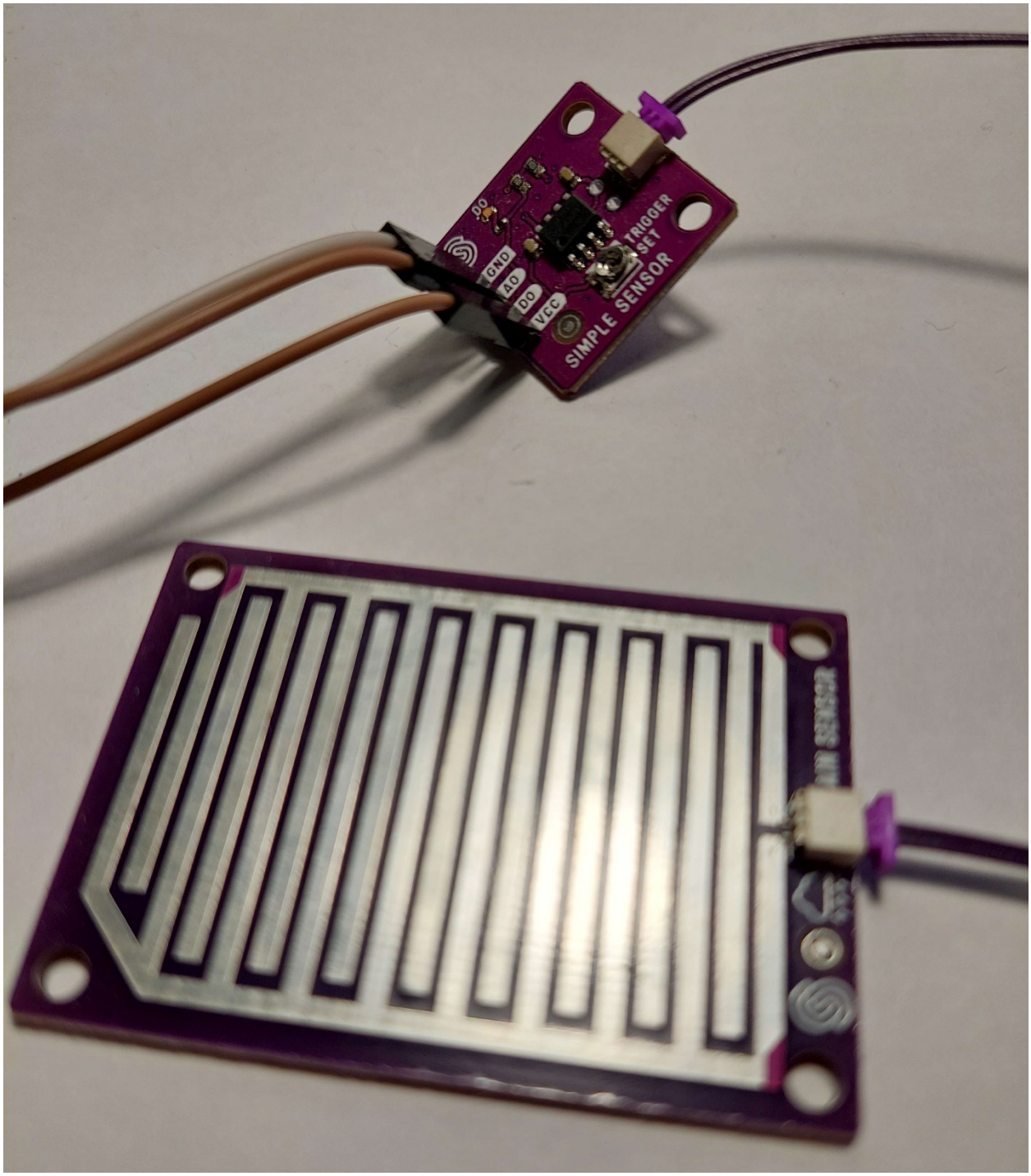
4. Senzor svjetla – ovaj senzor radi na principu svjetlosnog otpornika koji prema razini svjetla koje dobiva mijenja svoj otpor. Izlazni otpor je proporcionalno inverzan što znači da što je veća razina svjetla to je otpor manji, a što je mračnije u okolini, otpor je veći. Isto tako na samoj pločici senzora nalazi se podesivi otpornik koji omogućava podešavanje izlaznog praga na digitalnom izlazu, dok analogni izlaz daje signal u kontinuitetu Koriste se napajanja između 3.3 i 5V (Soldered Electronics, 2023).



Slika 13: Senzor svijetla

Izvor: Autor

5. Senzor kiše - senzor koji se sastoji od dva dijela, od pločice na kojoj se nalaze osjetne linije nikla i pločice. Izlazne vrijednosti mogu biti analogne za kontinuirana mjerenja ili digitalni signal koji se pomoću ugrađenog podesivog otpornika namješta na određenu vrijednost niskog ili visokog stanja. Napon napajanja se nalazi u dosegu 3.3 do 5V (Soldered Electronics, 2023).

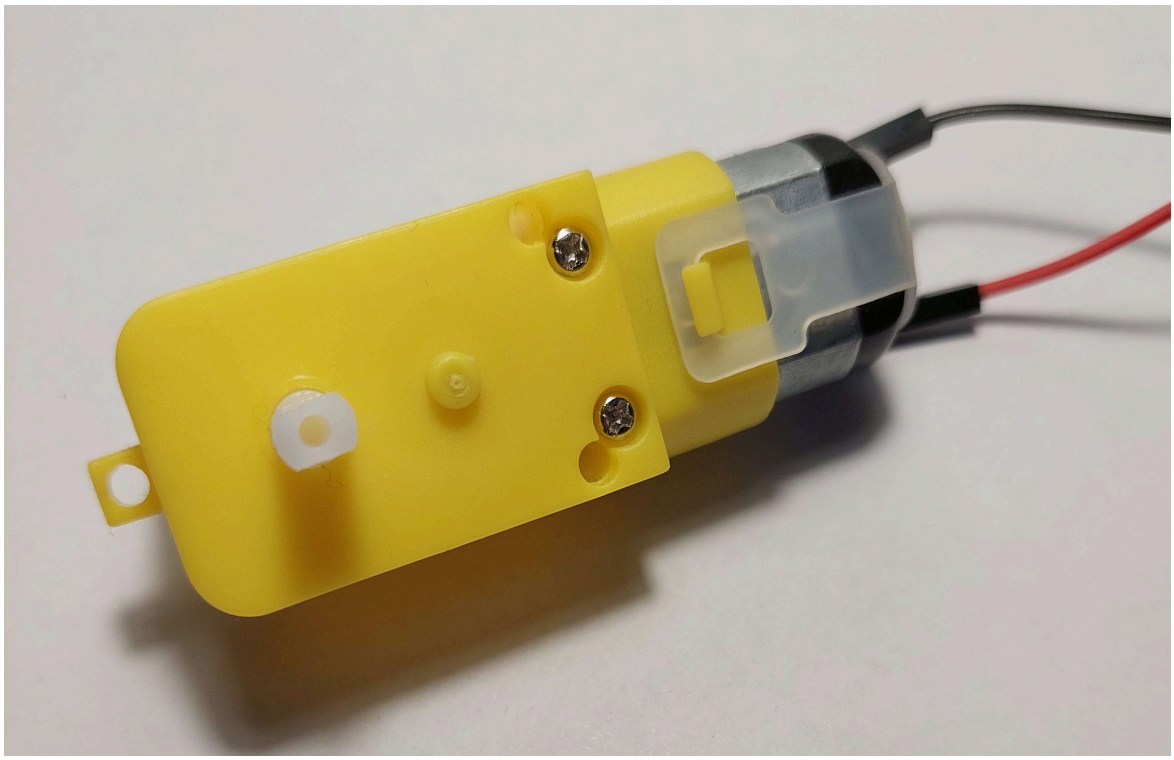


Slika 14: Senzor kiše i niklovana osjetna pločica

Izvor: Autor

6. Istosmjerni motor s prijenosom – kao što samo ime govori ovaj istosmjerni motor opremljen je prijenosom u omjeru 1:48 te se spaja na dolje navedeni L295N

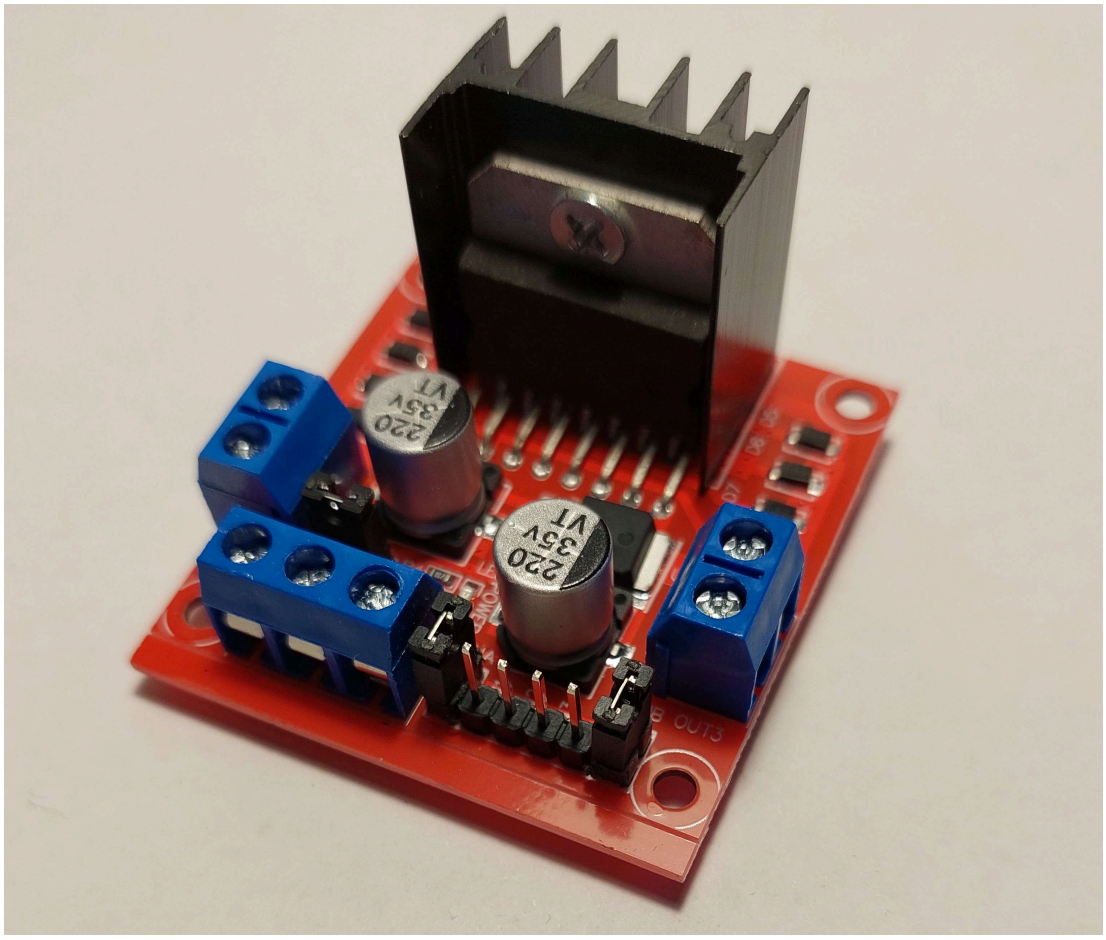
upravljač. Koristi se kod malih robota, mehanizama kod kojih je potrebna neka konstantna vrtnja za rad ili kretanje. Napon potreban za rad koji se dobiva preko upravljača je između 3 i 6V. (DigiKey Electronics, 2023.)



Slika 15: Istosmjerni električni motor

Izvor: Autor

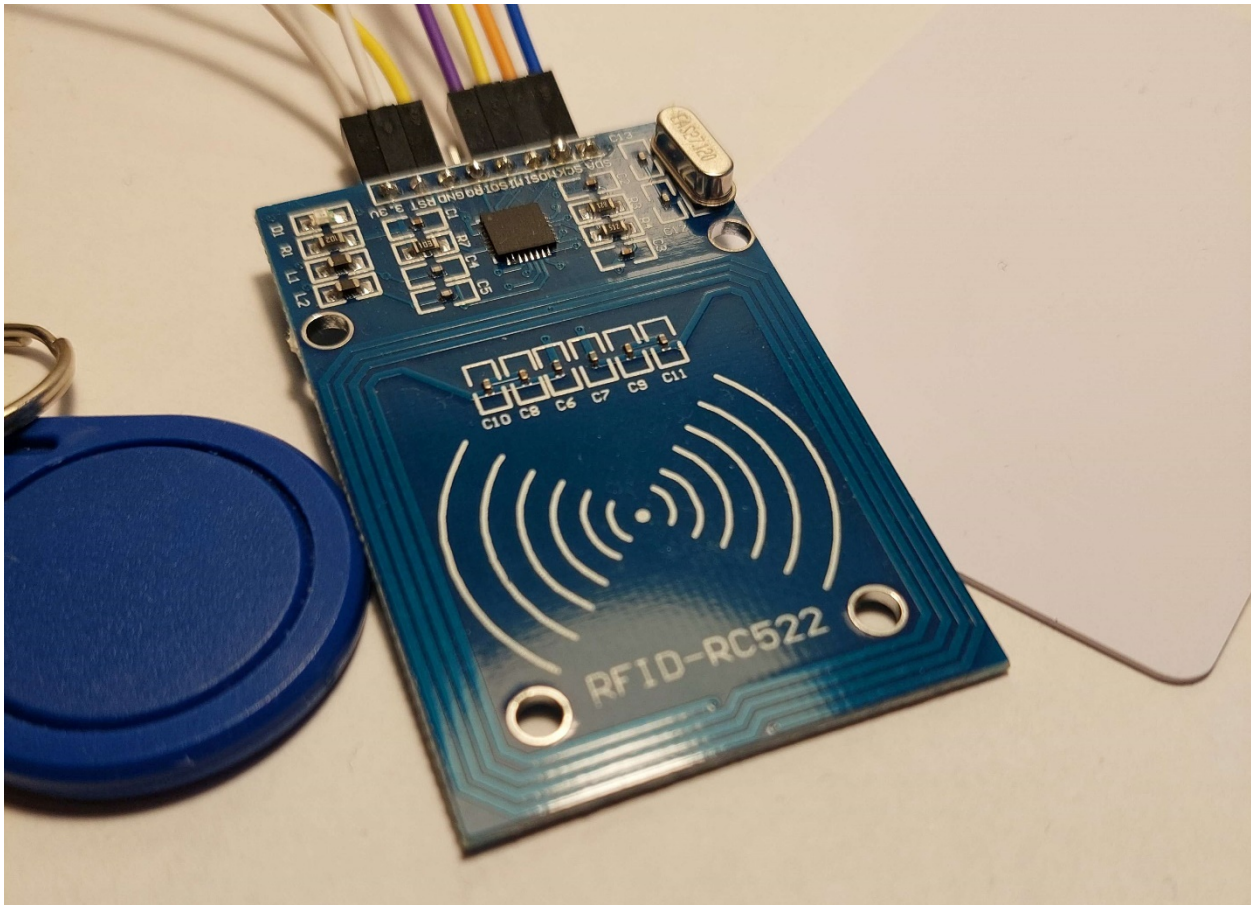
7. L298N upravljač motora – upravljač istosmjernog motora koji pomoću H-mosta kontrolira smjer vrtnje uz pomoć dva ulaza signala, jedan za svaki smjer. Ne stvara opterećenje na sam mikro upravljač zbog potrebe za vanjskim napajanjem od 5 ili 12V ulaznog napona. Daje i mogućnost kontroliranja brzine vrtnje motora pomoću trećeg ulaza signala. Zbog samog načina izvedbe omogućuje kontroliranje dva istosmjerna motora odjedanput (Soldered Electronics, 2023).



Slika 16: L298N upravljač

Izvor: Autor

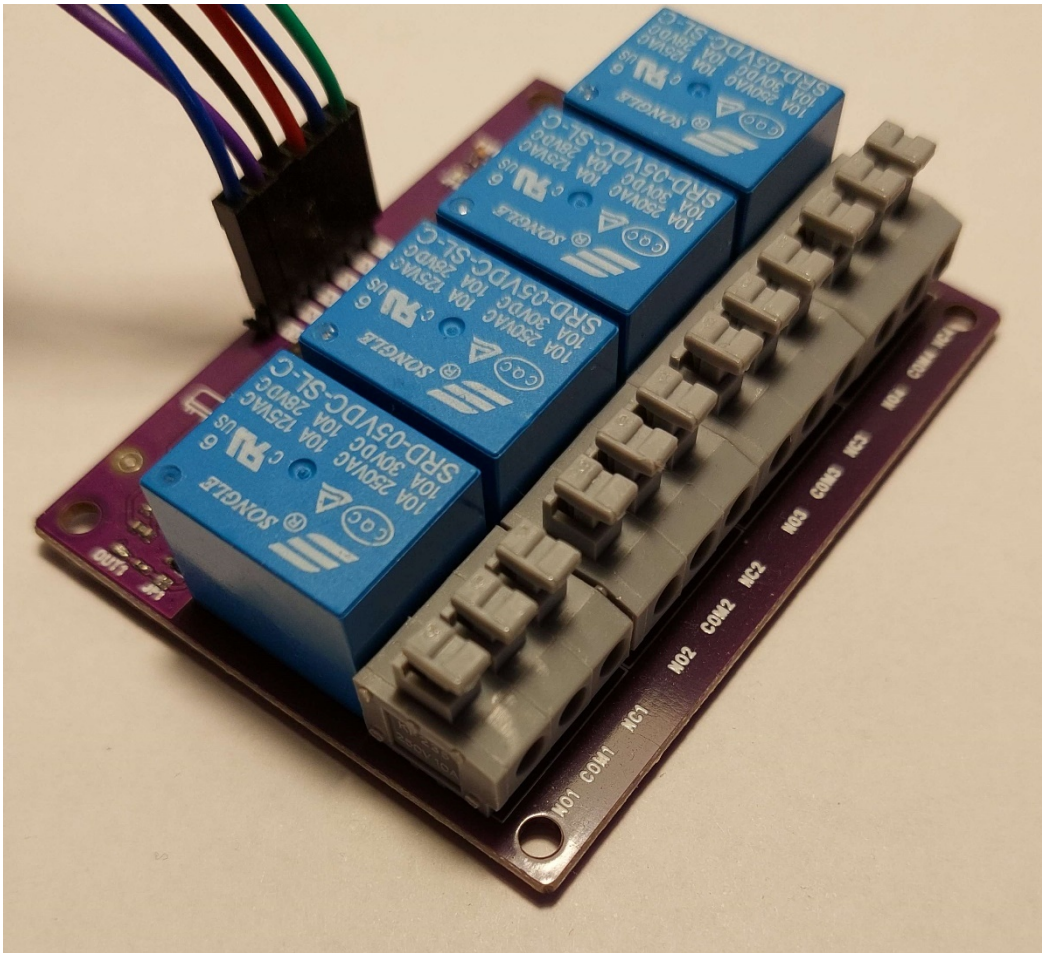
8. RFID čitač – komponenta s osnovnom primjenom za beskontaktnu komunikaciju. Princip rada se može pronaći u svakodnevici, od bankovnih kartica pa sve do otključavanja vrata ili automobila. Pomoću SPI komunikacije dolazi do slanja očitanih informacija na mikro upravljač. Drugi naziv je i MF RC522, doseg očitanja je do 5 cm, a radni napon 3.3V (Soldered Electronics, 2023).



Slika 17: RFID čitač s karticom i tokenom za upotrebu

Izvor: Autor

9. Relejski sklop – relejski sklop omogućuje kontroliranje visokih struja pomoću niskih tj. može se kontrolirati 250V pomoću signala od 5V. Jedan je od čestih elemenata u automatizaciji koji zamjenjuje potrebu za ugradnjom pretvarača napona zbog same funkcije kontrole. Dolazi u različitim konfiguracijama u smislu dostupnih sklopki koje se inače nazivaju kanali. U ovom radu korišten je sklop od 4 kanala za upravljanje, svaki sa sljedećim funkcijama: normalno otvoren izlaz, normalno zatvoren i zajednički priključak. Ovisno o namjeni odabire se jedan od dvoje izlaza (Soldered Electronics, 2023).



Slika 18: Četvero kanalni relejski sklop

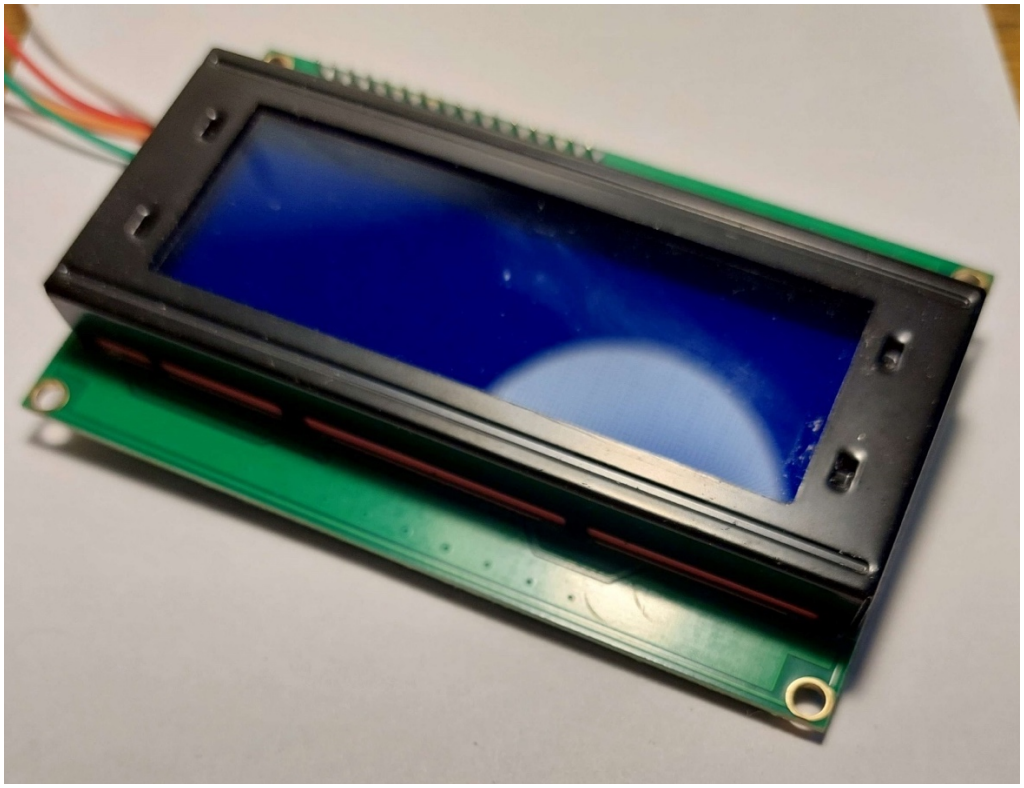
Izvor: Autor

10. Zujalica – jedna od najzastupljenijih komponenti u elektronici koja ima mogućnost proizvodnje zvučnih signala. Princip rada temelji se na piezzo električnom efektu i oscilatoru. Napajanje je 5V (Soldered Electronics, 2023).

Slika 19: Piezzo električna zujalica

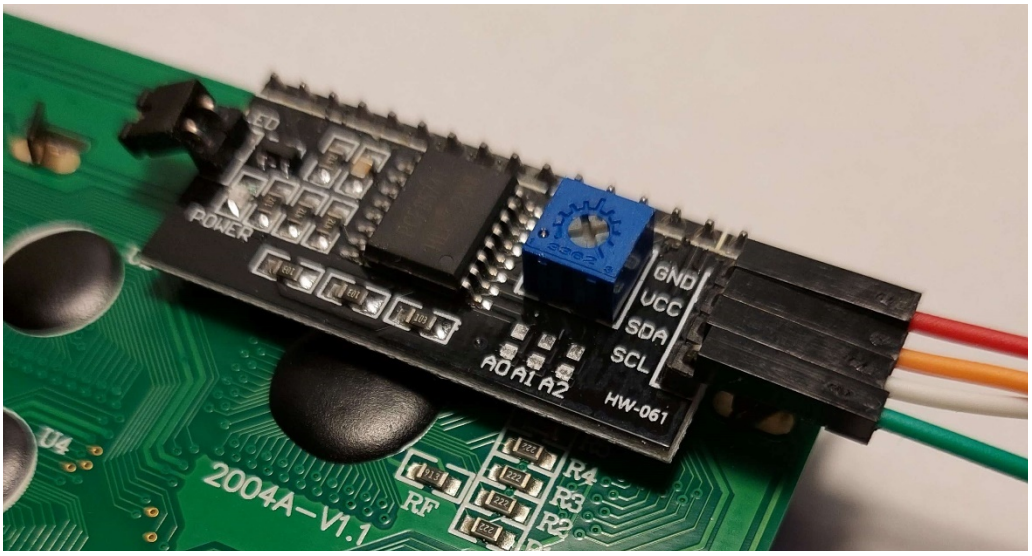
Izvor: Autor

11. LCD zaslon I2C – Jednostavna komponenta koja ima za cilj prikazivanje željenog sadržaja na zaslonu. Korišteni zaslon ima mogućnost prikazivanje 20 znakova u 4 reda, komunikacija s mikro upravljačem se ostvaruje putem I2C adaptera koji omogućava jednostavnu upotrebu i spajanje. Radni napon adaptera i zaslona je 3.3V (Soldered Electronics, 2023).



Slika 20: LCD zaslon 20x4

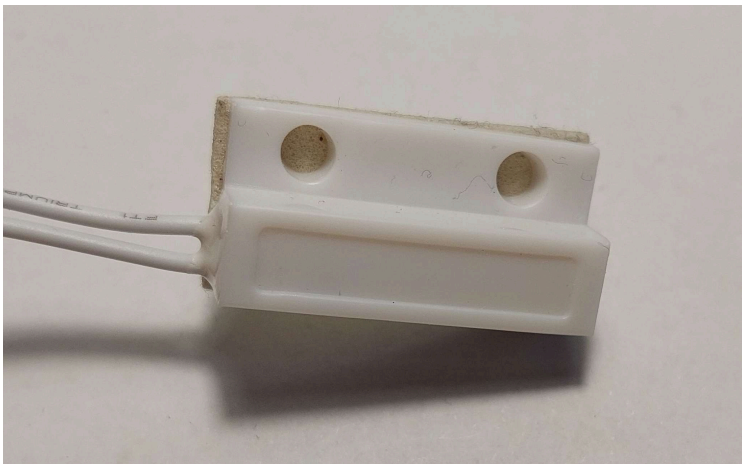
Izvor: Autor



Slika 21: Prikaz I2C adaptera u spoju s LCD zaslonom

Izvor: Autor

12. Magnetski preklopnik – Uz jednostavni princip rada i malih dimenzija, magnetski preklopnik omogućuje jednostavnu upotrebu i integraciju u sustav. Princip rada se bazira na magnetnom privlačenju. U samom preklopniku se nalaze dva razdvojena para kontakata koji se kada se u blizini nađe magnet, približe i zatvore strujni krug te propuštaju napon (Soldered Electronics, 2023).



Slika 22: Magnetski prekidač

Izvor: Autor

13. Tipkalo – jednostavna komponenta s dva para sklopki, koristi se kod jednostavnih pa sve do složenijih projekata. Princip rada se temelji na tome da kada je tipkalo pritisnuto, spoje se dva para kontakta s kojim poteče napon. Ovisno o načinu spajanja može se dobiti i kombinacija da kada je tipkalo otpušteno strujni je krug zatvoren dok se pritiskom onda strujni kruh otvara (Soldered Electronics, 2023).

Slika 23: Prikaz jednostavnog tipkala

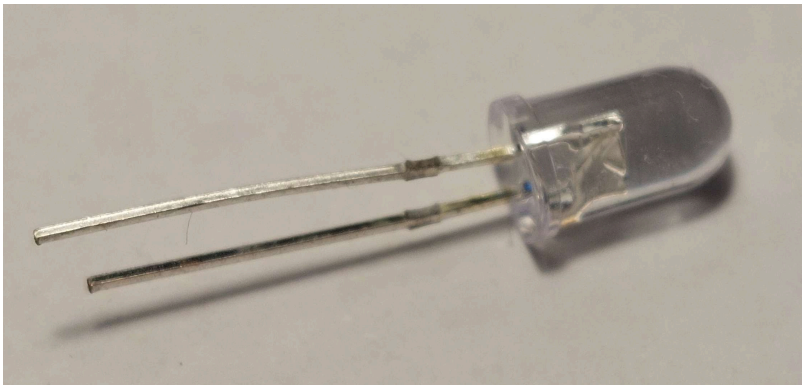
Izvor: Autor

14. Otpornik – Pasivna električna komponenta koja služi za snižavanje razine napona i kako samo ime govori, stvara otpor u strujnom krugu (Soldered Electronics, 2023).

Slika 24: Otpornik

Izvor: Autor

15. LED dioda – vrsta diode koja emitira svijetlo kada kroz nju poteče električni napon (Soldered Electronics, 2023).



Slika 25: bijela LED dioda

Izvor: Autor

5.4. Programski kod

Kako bi sve komponente radile i međusobno komunicirale, potrebno je bilo napisati programski kod te ga učitati na mikro upravljač. Sam kod se može podijeliti na više dijelova, od kojih svaki služi pojedinoj funkciji. Radi se o Arduino C jeziku u kombinaciji s HTML jezikom za izradu i postavljanje strukture internetske stranice. Kako se Arduino platforma sastoji od dva individualna elementa, samog Arduino Mega i ESP8266 mikro upravljača, kod je podijeljen na dva dijela. Analizirat će se samo dijelovi koda zbog jednostavnosti, a potpuni izvorni kod rješenja dan je u Prilogu 1 i Prilogu 2 ovog rada.

5.4.1. Arduino Mega kod

Na početku koda potrebno je dodati sve nužne biblioteke podrške od senzora sve do različitih vrsta komunikacije. Zatim se definira vrsta LCD-a koji se koristi kao i DHT senzora i RFID čitača.

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <Wire.h>
#include <SPI.h>
#include <MFRC522.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>
```

Kako bi Arduino poznavao pametnu karticu koja otključava kuću potrebno ju je zabilježiti u posebnu varijablu koju se naknadno uspoređuje s novo očitanom. U nastavku se definiraju sve potrebne varijable i njihove vrijednosti i stanja.

```
int RAIN_PIN = A1;
int PIR_PIN = 9;
bool isHeating = false;
bool isCooling = false;
String tag_UID = "8A7B29B4";
```

U funkciji *void setup()* definiraju se izlazi i ulazi, početna stanja i inicijaliziramo određene funkcije namijenjene za uspješan rad senzora i pozadinskih procesa.

```
pinMode(RELAY_PIN_HEATING, OUTPUT);
digitalWrite(RELAY_PIN_HEATING, LOW);
pinMode(buttonPin1, INPUT);
Serial.begin(115200);
```

U funkciji *void loop()* postavljaju se osnovni algoritmi programa i željena logika, od samog čitanja ulaza/izlaza do aktiviranja komunikacije i spojenih elemenata.

```
buttonState1 = digitalRead(buttonPin1);
      if(buttonState1 == HIGH ){
            ledState1 = !ledState1;
            digitalWrite(ledPin1, ledState1 ? HIGH : LOW);
            delay(500);
      }
```

Baza programa se okreće oko jedne varijable a to bi bila varijabla koja ukazuje da li je kuća zaključana ili otključana. Na temelju nje se aktiviraju ili deaktiviraju ostale pod-funkcije.Neke od ostalih funkcija su kontrola samih prozora pomoću istosmjernog motora, paljenje i gašenje klimatizacije i sigurnosni sustav u slučaju neovlaštenog ulaza.

```
if (isRaining == true  && topPos == true && botPos == false){
    digitalWrite(motorA1, LOW);
    digitalWrite(motorA2, HIGH);
    isMotorRunning = true;
    wasMotorStopped = false;
    MotorOpen = false;
    MotorClose = true;
  }
if (temperature > desiredTemperature){
```

```
    isHeating = false;
    isCooling = true;
    Serial.print("g");
    digitalWrite(RELAY_PIN_HEATING, LOW);
    digitalWrite(RELAY_PIN_COOLING, HIGH);
  }
```

Programski kod isto podržava i prikaz parametara kuće na LCD zaslonu koji se nalazi i unutar kuće kroz koji se može mijenjati prikaz pomoću dodijeljenog tipkala. Kako bi Arduino Mega komunicirao s ESP-om potrebno je dio koda posvetiti međusobnoj komunikaciji, konkretnije serijska komunikacija tipa UART. UART komunikacija kod Arduina se postavlja spajanjem RX i TX priključaka na njihove inverzne priključke drugog Arduina ili sličnog uređaja koji podržava taj način rada. To bi značilo da se RX1 spaja na TX2, a TX1 na RX2, ali da bi sve to radilo potrebno je spojiti zajednički GND priključak. U ovom slučaju ESP8266 spojen je na RX3 TX3 izlaze Arduino Mege. Slanje podataka kroz komunikacijske kanale ostvarilo se pomoću pokazivača koji odgovaraju namijenjenim stanjima varijabli. Ovaj način je jedan od najosnovnijih ali je zbog svoje jednostavnosti lak za korištenje i implementaciju. Komunikacija je dvosmjerna tako da kod mora sadržavati funkcije za slanje i primanje tj. čitanje pokazivača. U slijedećem primjeru prikazana je obrada dolazne komunikacije.

```
while (Serial.available() > 0) {
   char data = (char)Serial.read();

    if (data == 'A' ){ //led1
      ledState1 = !ledState1;
      digitalWrite(ledPin1, ledState1 ? HIGH : LOW);
    }
    }
```

Kako bi slanje vrijednosti varijable bilo moguće potrebno je slanje bajta po bajt što se može primijetiti u nastavku. Naravno to je samo dio ovog načina komunikacije između Arduino Mega i ESP-a.

```
void intToBytes(byte bytes[], int num, bool smallEndian) {
  unsigned int * len = (unsigned int*) &num;
  if (!smallEndian){
    bytes[0] = (*len) >> 8 & 0xff;
    bytes[1] = (*len) & 0xff;
  }else{
```

```
    bytes[1] = (*len) >> 8 & 0xff;
    bytes[0] = (*len) & 0xff;
  }
}
```

Na samom kraju koda još se pozivaju funkcije namijenjene za očitavanje RFID kartice u svrsi potvrđivanja vlasnika.

5.4.2. ESP kod

Sličan princip i organizacija programskog koda se nalazi i na strani ESP-a. Poziva se biblioteka WIFI koje je posvećena povezivanju i stvaranju lokalne mreže i web poslužitelja za prikaz podataka i kontrolu same kuće.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
```

Bitno je kod pokretanja programa unijeti dobre podatke glede naziva lokalne bežične mreže (SSID) na koju se želite spojiti te njezine pristupne lozinke. Nakon definiranja potrebnih varijabli nalazi se dio gdje se stvara sama internetska stranica u funkciji zvanoj *handleRoot*. U toj funkciji se pomoću HTML koda izrađuje jedna jednostavna web stranica koja prikazuje varijable u obliku „string“.

```
content += "<h2>Light Controls</h2>";
content += "<h3>Light 1</h3>";
content += "<p>State: " + Led1Status + "</p>";
content += "<button onclick=\"toggleLed1()\">Toggle Light
1</button><br><br>";
```

Ista stranica služi i za kontroliranje prozora i klimatizacije same kuće kao i paljenje i gašenje svijetla.

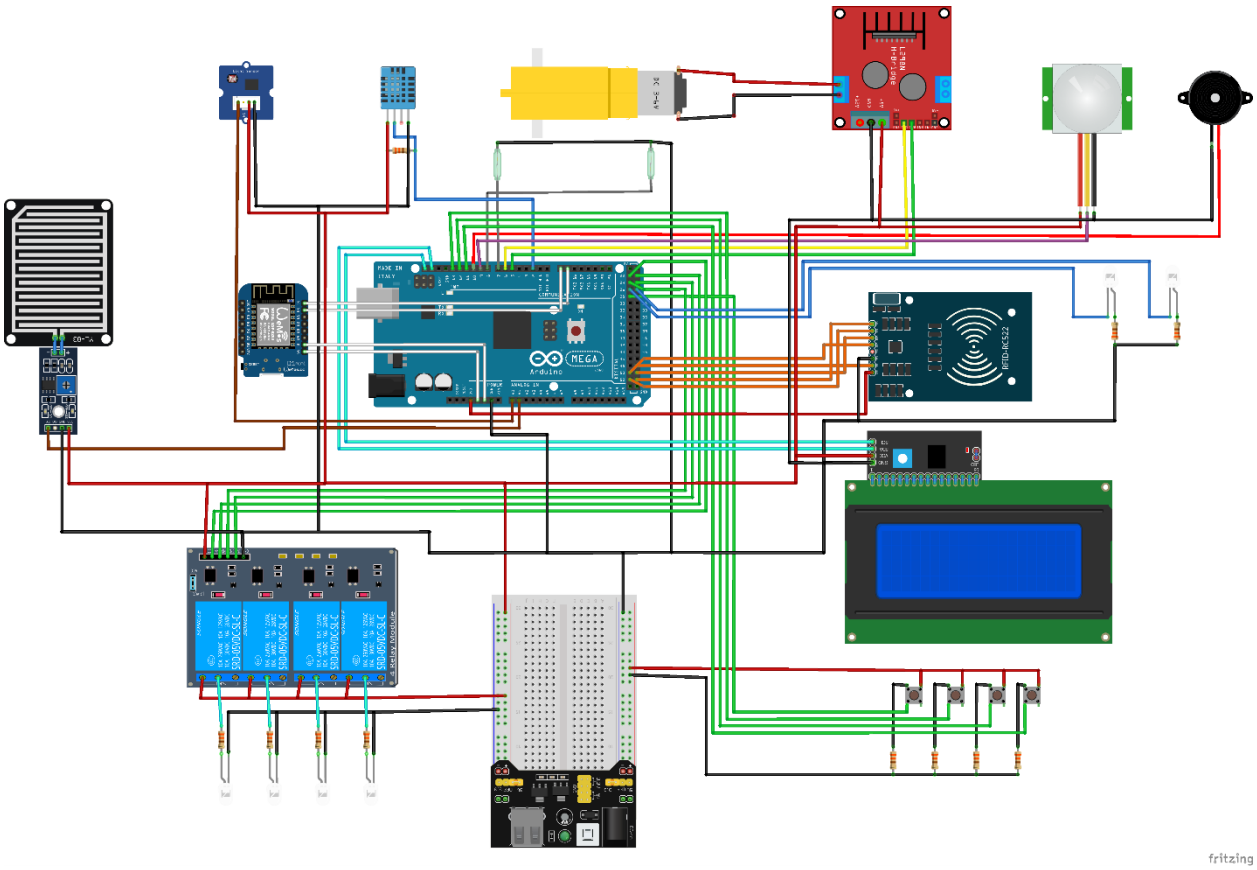
```
void handleMotorControl() {
  if (server.hasArg("motorAction")) {
    String action = server.arg("motorAction");
    if (action == "off") {
      Serial.print("E");
    } else if (action == "open") {
      Serial.print("C");
    } else if (action == "close") {
      Serial.print("D");
```

```
    }
  }
  server.send(200, "text/plain", "");
}
```

Na kraju je kod u potpunosti isti programskom kodu na Arduino Mega strani gdje se uspostavlja komunikacija te primanje i slanje podataka.

5.4.3. Realizacija

S gotovim programskim kodom, započeta je sljedeća faza izrade a to je sama realizacija i implementacija komponenti. Nakon što se u programskom kodu definiralo ulaze i izlaze, prema tom redoslijedu ih se spaja i u stvarnosti.

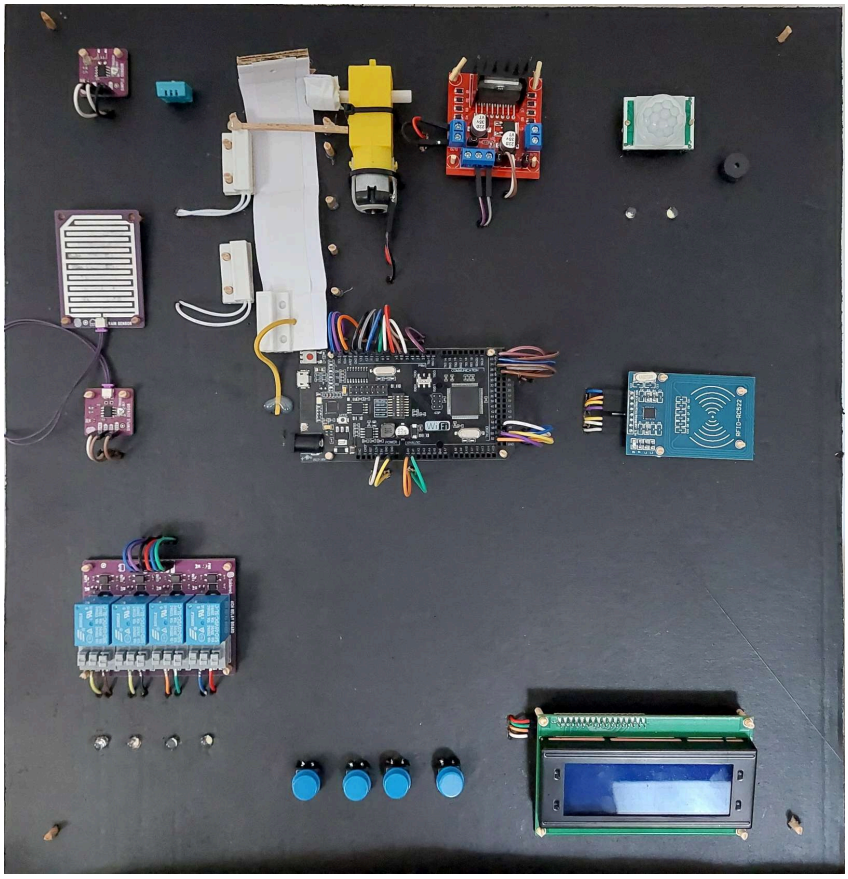


Slika 26: Shema spajanja komponenti na Arduino Mega

Izvor: Autor

Na Slici 26 je prikazana shema spajanja sustava, a izrađena je pomoću programa Fritzing. Senzori koji su navedeni u temi 6.3 raspoređeni su u grupama po namjeni. Na

lijevoj strani nalaze se senzori koji očitavaju stanja i svojstva okoline, dok se na desnoj strani nalaze oni zaduženi za sigurnost same kuće. Kako bi se predočila simulacija sustava grijanja i hlađenja, upotrijebljen je relejski sustav s dvije LED diode te pripadajućim otpornicima od 330Ω kao indikatore trenutne aktivnosti. Ostala dva relejska sklopa služe za paljenje i gašenje LED diode ali u ovom slučaju za simuliranje kućne rasvjete. U donjem desnom dijelu nalazi se LCD zaslon na kojem se prikazuju trenutna stanja kuće kao i vrijednosti zraka u vidu temperature i postotka relativne vlage. Pomoću tipkala mijenja se prikaz stanja na LCD-u. Stanja na LCD-u mogu biti sljedeća: kakvo je vrijeme izvan kuće, koja je pozicija prozora, stanje rada sustava hlađenja i grijanja te da li je kuća zaključana ili ne. Isto tako iskorištena su i dva tipkala za lokalno paljenje i gašenje LED rasvjete i jedan za zaključavanje kuće. RFID senzor s karticom omogućava beskontaktno otključavanje kuće i mogućnosti personaliziranog pristupa za više osoba. Pametnu kuću čini sustav upravljanja automatiziranim prozorima. U ovom rješenju, izrađen je samo automatizirani prozor. Pomoću upravljača i istosmjernog motora kontrolira se podizanje i spuštanje samih prozora, a kad prozor dostigne donju ili gornju graničnu poziciju, aktivira se magnetski senzor koji isključuje motor da ne bi došlo do oštećenja ili loma prozora ili motora. Bitno je napomenuti da je zbog primitivne prirode HTML koda (neraspoloživosti weba) potrebno nakon svake interakcije s web stranicom ponovno učitavanje iste, što je implementirano samim kodom. Prema izrađenoj shemi konstruiran je model koji prikazuje samu strukturu rada i njegovu funkcionalnost. Na Slici 27 prikazan je sam model u radu i njegova konstrukcija od obnovljivog materija.



Slika 27: Funkcionalni izgled modela u radu

Izvor: Autor

5.4.4. Moguća unaprjeđenja

Kako se ovakvi tehnološki sustavi razvijaju velikom brzinom koje prate jači procesori i povećani broj ulazno-izlaznih priključaka moguće je dodavati i veći broj elemenata. Kao prijedlog budućem unaprjeđenju predlaže se dodavanje membranske tipkovnice u svrhu lokalnog zadavanja željene temperature ili otključavanja kuće pomoću korisničke lozinke ili pina. Također se može nadodati veći LCD grafički zaslon gdje se može projektirati izgled kuće kao bitmap datoteka s kojom se može uspostaviti povezanost korisnika i sustava na vizualan način. Moguće i povećanje broja relejskih preklopnika za dodavanje dodatnih elemenata rasvjeta ili uređaja.

6. Zaključak

Ovim radom nastojalo se zbližiti Arduino C programiranje s Arduino platformom i njihovim primjenama u mehatronici u svakodnevnom životu. Pomoću raznih senzora i aktuatora

realizirala se zamisao stvaranja mehatroničkog sustava od početka do kraja, od pisanja vlastitog koda u Arduino C jeziku pa sve do kombiniranja elemenata u jednu cjelinu. Cijeli rad se sastojao od malih cjelina koja su bitne u razumijevanju projektiranja sustava, od samog programiranja C jezicima, preko razvoja mehatronike i interneta stvari, pa sve do samog Arduina. Primjenom Arduina u mehatronici mogu se postići velike stvari u cilju automatizacije, robotike i nadzora raznih parametara. Pametna kuća kao samo jedan on aktualnih primjera automatizacije, spoj je svakodnevnice i moderne tehnologije. Može se očekivati da će napredak u tehnologiji omogućiti lakše programiranje, implementiranje i bržu realizaciju projekata u mehatronici i da će sama Arduino platforma postati naprednija i snažnija.

7. Literatura

Arduino. (n.d.). *What is Arduino?* https://www.arduino.cc/en/Guide/Introduction (20.3.2023).

Arduino Documentation. (n.d.). *Getting Started with Arduino*. https://docs.arduino.cc/learn/starting-guide/getting-started-arduino (20.3.2023).

Arduino Documentation. (n.d.). *UNO R3*. https://docs.arduino.cc/hardware/uno-rev3 (20.3.2023).

Arduino Documentation. (n.d.). *Mega 2560 Rev3*. https://docs.arduino.cc/hardware/mega-2560 (20.3.2023).

Arduino Documentation. (n.d.). *Nano.* https://docs.arduino.cc/hardware/nano (20.3.2023).

Bolton, W. (2015). *Mechatronics: Electronic control systems in mechanical and electrical engineering (6th ed.)*. Pearson Education Limited.

Coursera. (2023). *What Is C++? (And How to Learn It*). https://www.coursera.org/articles/what-is-c-plus-plus (24.3.2023).

GNU Project. (n.d.). *GCC 12.2 manuals*. https://gcc.gnu.org/onlinedocs/12.2.0/ (24.3.2023).

Norris, D. (2015). The Internet of Things: Do-It-Yourself at Home *Projects for Arduino, Raspberry Pi and BeagleBone Black*. McGraw Hill Professional.

Prinz, P., i Crawford, T. (2006). *C in a Nutshell*. O'Reilly Media. https://books.google.hr/books?id=4Mfe4sAMFUYC&pg=PT79&hl=hr&source=gbs_selected_pages&cad=3#v=onepage&q&f=false

Purdum, J. (2015). *Beginning C for Arduino, Second Edition: Learn C Programming for the Arduino (2nd ed.)*. Apress. https://doi.org/10.1007/978-1-4842-0940-0

Serpanos, D., i Wolf, M. (2018). *Internet-of-Things (IoT) Systems: Architectures, Algorithms, Methodologies*. Springer.

Soldered Electronics. (n.d.). *Datasheet for Adafruit product*. (23.8.2023).

Soldered Electronics. *DHT11 temperature and humidity sensor*. https://soldered.com/product/dht11-temperature-and-humidity-sensor/ (23.8.2023).

Soldered Electronics. *Movement sensor HC-SR501*. https://soldered.com/product/movement-sensor-hc-sr501/ (23.8.2023).

Soldered Electronics. *Simple light sensor board*. https://soldered.com/product/simple-light-sensor-board/ (23.8.2023).

Soldered Electronics. *Simple rain sensor*. https://soldered.com/product/simple-rain-sensor/ (23.8.2023).

Soldered Electronics. *H-Bridge Arduino DC Motor Driver Dual L298N*. https://soldered.com/product/dc-motor-driver-dual-h-bridge-l298n/ (23.8.2023).

Soldered Electronics. *RFID reader MFRC-522 with RFID card*. https://soldered.com/product/mfrc-522-rfid-13-56mhz-reader/ (23.8.2023).

Soldered Electronics. *4-channel relay board*. https://soldered.com/product/4-channel-relay-board/ (23.8.2023).

Soldered Electronics. *Aktivni Buzzer 5V*. https://soldered.com/product/active-buzzer-5v/ (23.8.2023).

Soldered Electronics. *LCD display 20x4 I2C white characters on blue background.* https://soldered.com/product/lcd-display-20x4-i2c-white-characters-on-blue-background/ (23.8.2023).

Soldered Electronics. *Magnetic door/window switches*. https://soldered.com/product/magnetic-door-window-switches/ (23.8.2023).

Soldered Electronics. *Pushbutton 12mm*. https://soldered.com/product/pushbutton-12mm/.

Soldered Electronics. *25x 330ohm resistor (for LED diodes).* https://soldered.com/product/25x-330ohm-resistor-for-leds/ (23.8.2023).

Soldered Electronics. *10x crystal white 5mm LED*. https://soldered.com/product/crystal-white-5mm-led-diode/ (23.8.2023).

8. Popis slika

9. Popis tablica

Prilog 1 - Izvorni kod rješenja Arduino Mega

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <Wire.h>
#include <SPI.h>
#include <MFRC522.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 4);

#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

#define SS_PIN 53
#define RST_PIN 49
MFRC522 mfrc522(SS_PIN, RST_PIN);

byte readCard[4];
String tag_UID = "8A7B29B4";
String tagID = "";

int LIGHT_PIN = A0;
int RAIN_PIN = A1;
int PIR_PIN = 9;
```

```
int mag_top = 8;
int mag_bot = 7;

int motorA1 = 5;
int motorA2 = 6;

int lockOnLed = 26;
int lockOffLed = 27;

int buzzer = 10;

int buttonPin1 = 11;
int buttonPin2 = 12;
int buttonPin3 = 28;
int buttonLock = 13;

int RELAY_PIN_HEATING = 22;
int RELAY_PIN_COOLING = 23;
int ledPin1 = 24;
int ledPin2 = 25;

bool isHeating = false;
bool isCooling = false;
bool isRaining = false;
bool isDay = false;
bool isNight = false;
bool ledState1 = false;
bool ledState2 = false;
bool buttonState1 = false;
bool buttonState2 = false;
bool buttonState3 = false;
bool lockbuttonState = false;
bool lockState = false;
bool topPos = false;
bool botPos = false;
bool mov = false;
bool isMotorRunning = false;
bool wasMotorStopped = false;
bool MotorClose = false;
bool MotorOpen = false;
int PIRstate = LOW;

int humidity;
```

```
int temperature;
int desiredTemperature;

int display = 0;
int lastButtonState3 = LOW;
int dc = 0;

void setup() {
  pinMode(RELAY_PIN_HEATING, OUTPUT);
  pinMode(RELAY_PIN_COOLING, OUTPUT);
  digitalWrite(RELAY_PIN_HEATING, LOW);
  digitalWrite(RELAY_PIN_COOLING, LOW);

  pinMode(buttonPin1, INPUT);
  pinMode(buttonPin2, INPUT);
  pinMode(buttonPin3, INPUT);

  pinMode(motorA1, OUTPUT);
  pinMode(motorA2, OUTPUT);
  digitalWrite(motorA1, LOW);
  digitalWrite(motorA2, LOW);

  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  digitalWrite(ledPin1, LOW);
  digitalWrite(ledPin2, LOW);

  pinMode(mag_top, INPUT_PULLUP);
  pinMode(mag_bot, INPUT_PULLUP);

  pinMode(PIR_PIN, INPUT);
  pinMode(RAIN_PIN, INPUT_PULLUP);

  pinMode(buzzer, OUTPUT);

  pinMode(lockOnLed, OUTPUT);
  pinMode(lockOffLed, OUTPUT);
  pinMode(buttonLock, INPUT);

  SPI.begin();
  dht.begin();
  mfrc522.PCD_Init();
  Serial.begin(115200);
```

```
  lcd.init();
  lcd.backlight();
}

void loop(){

  buttonState1 = digitalRead(buttonPin1);
  buttonState2 = digitalRead(buttonPin2);
  buttonState3 = digitalRead(buttonPin3);
  lockbuttonState = digitalRead(buttonLock);

  humidity = dht.readHumidity();
  temperature = dht.readTemperature();

  if(buttonState1 == HIGH ){
    ledState1 = !ledState1;
    digitalWrite(ledPin1, ledState1 ? HIGH : LOW);
    delay(500);
  }

  if(buttonState2 == HIGH ){
    ledState2 = !ledState2;
    digitalWrite(ledPin2, ledState2 ? HIGH : LOW);
    delay(500);
  }

  if(lockbuttonState == HIGH){
    lockState = true;
    delay(500);
  }

  if(lockState == true){
    digitalWrite(lockOnLed, HIGH);
    digitalWrite(lockOffLed, LOW);
    digitalWrite(ledPin1, LOW);
    digitalWrite(ledPin2, LOW);
    digitalWrite(RELAY_PIN_HEATING, LOW);
    digitalWrite(RELAY_PIN_COOLING, LOW);
    ledState1 = false;
    ledState2 = false;
    isHeating = false;
    isCooling = false;
    if (lockState == true && topPos == true){
```

```
    digitalWrite(motorA1, HIGH);
    digitalWrite(motorA2, LOW);
    analogWrite(motorA1, 200);
    isMotorRunning = true;
  if (isMotorRunning == true && topPos == false && botPos == true){
      isMotorRunning = false;
      digitalWrite(motorA1, LOW);
      digitalWrite(motorA2, LOW);
    }
  }
  if(mov == HIGH){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("!LOPOV!");
    lcd.setCursor(0, 1);
    lcd.print("!LOPOV!");
    lcd.setCursor(0, 2);
    lcd.print("!LOPOV!");
    lcd.setCursor(0, 3);
    lcd.print("!LOPOV!");
  }

}else if(lockState == false){
  digitalWrite(lockOnLed, LOW);
  digitalWrite(lockOffLed, HIGH);
  lcd.setCursor(0, 0);
  lcd.print("Temperature: ");
  lcd.print(temperature);
  lcd.print((char)223);
  lcd.print("C");
  lcd.setCursor(0, 1);
  lcd.print("Humidity: ");
  lcd.print(humidity);
  lcd.print("%");
  lcd.setCursor(0, 2);
  lcd.print("--------------------");

  if(buttonState3 == HIGH  && lastButtonState3 == LOW){
    dc++;

    switch (dc) {
      case 1:
        lcd.setCursor(0, 3);
```

```
  lcd.print("                    ");
  lcd.setCursor(0, 3);
  lcd.print("Ambient: ");
  if (isDay == true){
    lcd.print("Day");
  }else{
    lcd.print("Night");
  }
  break;
case 2:
  lcd.setCursor(0, 3);
  lcd.print("                    ");
  lcd.setCursor(0, 3);
  lcd.print("Weather: ");
  if (isRaining == true){
    lcd.print("Raining");
  }else{
    lcd.print("Clear");
  }
  break;
case 3:
  lcd.setCursor(0, 3);
  lcd.print("                    ");
  lcd.setCursor(0, 3);
  lcd.print("Window: ");
  if (topPos == true && botPos == false){
    lcd.print("Up");
  }else if (topPos == false && botPos == true){
    lcd.print("Closed");
  }else if (topPos == false && botPos == false){
    lcd.print("Stopped");
  }
  break;
case 4:
  lcd.setCursor(0, 3);
  lcd.print("                    ");
  lcd.setCursor(0, 3);
  lcd.print("HVAC: ");
  if (isHeating == true && isCooling == false){
    lcd.print("Heating");
  }else if (isHeating == false && isCooling == true){
    lcd.print("Cooling");
  }else if (isHeating == false && isCooling == false){
```

```
          lcd.print("Off");
        }
        break;
      case 5:
        lcd.setCursor(0, 3);
        lcd.print("                    ");
        lcd.setCursor(0, 3);
        lcd.print("Security: ");
        if (lockState == true){
          lcd.print("Locked");
        }else{
          lcd.print("Unlocked");
        }
        break;
      case 6:
        lcd.setCursor(0, 3);
        lcd.print("                    ");
        lcd.setCursor(0, 3);
        dc = 0;
        break;
    }
  }
  lastButtonState3 = buttonState3;
}

while (readID()){
  if (tagID == tag_UID){
    if (lockState == true){
      lockState = false;
    }
  }
}

int rainSensorValue = analogRead(RAIN_PIN);
if (rainSensorValue < 700 ) {
  isRaining = true;
} else {
  isRaining = false;
}

int lightSensorValue = analogRead(LIGHT_PIN);
lightSensorValue = 1023 - lightSensorValue;
if (lightSensorValue > 200) {
```

```
  isDay = true;
  isNight = false;
} else {
  isDay = false;
  isNight = true;
}

float TOP = digitalRead(mag_top);
if (TOP == 0) {
  topPos = true;
} else {
  topPos = false;
}

float BOT = digitalRead(mag_bot);
if (BOT == 0) {
  botPos = true;
} else {
  botPos = false;
}

mov = digitalRead(PIR_PIN);
if (mov == HIGH && lockState == true) {
  for (int i = 0; i < 100; i++) {
    digitalWrite(buzzer, HIGH);
    delay(1);
    digitalWrite(buzzer, LOW);
    delay(1);
  }
  delay(50);
  for (int j = 0; j < 100; j++) {
    digitalWrite(buzzer, HIGH);
    delay(2);
    digitalWrite(buzzer, LOW);
    delay(2);
  }
  delay(500);
  if (PIRstate == LOW) {
    Serial.println("Motion detected!");
    PIRstate = HIGH;
  }
} else {
  delay(500);
```

```
    if (PIRstate == HIGH) {
      Serial.println("Motion stopped!");
      PIRstate = LOW;
      digitalWrite(buzzer, LOW);
    }
  }

  if (desiredTemperature > 0){
    if (temperature > desiredTemperature){
      isHeating = false;
      isCooling = true;
      Serial.print("g");
      digitalWrite(RELAY_PIN_HEATING, LOW);
      digitalWrite(RELAY_PIN_COOLING, HIGH);
    }else if (temperature < desiredTemperature){
      isHeating = true;
      isCooling = false;
      Serial.print("f");
      digitalWrite(RELAY_PIN_HEATING, HIGH);
      digitalWrite(RELAY_PIN_COOLING, LOW);
    }else if (temperature == desiredTemperature){
      isHeating = false;
      isCooling = false;
      Serial.print("h");
      digitalWrite(RELAY_PIN_HEATING, LOW);
      digitalWrite(RELAY_PIN_COOLING, LOW);
    }
  }

  if (isMotorRunning == true && topPos == true && botPos == false && MotorOpen
== true){ //kad se otvore
    isMotorRunning = false;
    digitalWrite(motorA1, LOW);
    digitalWrite(motorA2, LOW);
  }

  if (isMotorRunning == true && topPos == false && botPos == true && MotorClose
== true){ //kad se zatvore
    isMotorRunning = false;
    digitalWrite(motorA1, LOW);
    digitalWrite(motorA2, LOW);
  }
```

```
  if (isDay == true && topPos == false && botPos == true && isRaining == false
&& lockState == false){
      digitalWrite(motorA1, HIGH);
      digitalWrite(motorA2, LOW);
      isMotorRunning = true;
      wasMotorStopped = false;
      MotorOpen = true;
      MotorClose = false;
 }
  if (isNight == true && topPos == true && botPos == false){
    digitalWrite(motorA1, LOW);
    digitalWrite(motorA2, HIGH);
    isMotorRunning = true;
    wasMotorStopped = false;
    MotorOpen = false;
    MotorClose = true;
  }

  if (isRaining == true  && topPos == true && botPos == false){
    digitalWrite(motorA1, LOW);
    digitalWrite(motorA2, HIGH);
    isMotorRunning = true;
    wasMotorStopped = false;
    MotorOpen = false;
    MotorClose = true;
  }

  while (Serial.available() > 0) {
    char data = (char)Serial.read();

      if (data == 'A' ){ //led1
        ledState1 = !ledState1;
        digitalWrite(ledPin1, ledState1 ? HIGH : LOW);
      }

      if (data == 'B' ){ //led2
        ledState2 = !ledState2;
        digitalWrite(ledPin2, ledState2 ? HIGH : LOW);
      }

      if (data == 'C' && topPos == false && botPos == true){ //open
        digitalWrite(motorA1, HIGH);
        digitalWrite(motorA2, LOW);
```

```
        isMotorRunning = true;
        wasMotorStopped = false;
        MotorOpen = true;
        MotorClose = false;
      } else if (data == 'D' && topPos == true && botPos == false) { //close
       digitalWrite(motorA1, LOW);
       digitalWrite(motorA2, HIGH);
       isMotorRunning = true;
       wasMotorStopped = false;
       MotorOpen = false;
       MotorClose = true;
      } else if (data == 'E') { //stop
        if (isMotorRunning) {
          digitalWrite(motorA1, LOW);
          digitalWrite(motorA2, LOW);
          isMotorRunning = false;
          wasMotorStopped = true;
          MotorOpen = false;
          MotorClose = false;
        }
      } else if (data == 'C' && wasMotorStopped == true && topPos == false &&
botPos == false){
          digitalWrite(motorA1, HIGH);
          digitalWrite(motorA2, LOW);
          isMotorRunning = true;
          wasMotorStopped = false;
          MotorOpen = true;
      } else if (data == 'D' && wasMotorStopped == true && topPos == false &&
botPos == false){
        digitalWrite(motorA1, LOW);
        digitalWrite(motorA2, HIGH);
        isMotorRunning = true;
        wasMotorStopped = false;
        MotorClose = true;
      }

        if (data == 'F'){ //heating
          isHeating = true;
          isCooling = false;
          Serial.print("f");
          digitalWrite(RELAY_PIN_HEATING, HIGH);
          digitalWrite(RELAY_PIN_COOLING, LOW);
        }
```

```
        if (data == 'G'){ //cooling
          isHeating = false;
          isCooling = true;
          Serial.print("g");
          digitalWrite(RELAY_PIN_HEATING, LOW);
          digitalWrite(RELAY_PIN_COOLING, HIGH);
        }

        if (data == 'H'){ //stop HC
          isHeating = false;
          isCooling = false;
          Serial.print("h");
          digitalWrite(RELAY_PIN_HEATING, LOW);
          digitalWrite(RELAY_PIN_COOLING, LOW);
        }

        if (data == 'L'){
          sendDHTtemp();
          sendDHThum();
          delay(500);
          sendSensorData();
        }

        if(data == 'X'){
          readDesTemp();
        }
  }
}

void sendSensorData() {
  if (ledState1 == HIGH){
    Serial.print("T");
  }else if (ledState1 == LOW){
    Serial.print("U");
  }

  if (ledState2 == HIGH){
    Serial.print("V");
  }else if (ledState2 == LOW){
    Serial.print("W");
  }
```

```
  if (mov == LOW){
    Serial.print("P");
  }else if (mov == HIGH){
    Serial.print("R");
  }

  if (isDay == true && isNight == false){
    Serial.print("I");
  }else if (isDay == false && isNight == true){
    Serial.print("K");
  }

  if (isRaining == true){
    Serial.print("L");
  }else  if (isRaining == false){
    Serial.print("M");
  }

  if (topPos == true){
    Serial.print("N");
  }else if (botPos == true){
    Serial.print("O");
  }

  if (lockState == true){
    Serial.print("S");
  }else{
    Serial.print("Q");
  }
}

void sendDHTtemp(){
  byte bytes[2];
  intToBytes(bytes, temperature, false);
  Serial.print("Y");
  Serial.write(bytes,2);
}

void sendDHThum(){
  byte bytes[2];
  intToBytes(bytes, humidity, false);
  Serial.print("Z");
  Serial.write(bytes,2);
```

```
}

void readDesTemp(){
  byte bytes[2];
  Serial.readBytes(bytes,2);
  desiredTemperature = bytesToInt(bytes, false);
}

float bytesToInt(byte bytes[], bool smallEndian) {
  if (!smallEndian)
    return (  ( (bytes[0] & 0xFF) << 8) | (bytes[1] & 0xFF) );

  return (  ( (bytes[1] & 0xFF) << 8) | (bytes[0] & 0xFF) );
}

void intToBytes(byte bytes[], int num, bool smallEndian) {

  unsigned int * len = (unsigned int*) &num;

  if (!smallEndian){
    bytes[0] = (*len) >> 8 & 0xff;
    bytes[1] = (*len) & 0xff;
  }else{
    bytes[1] = (*len) >> 8 & 0xff;
    bytes[0] = (*len) & 0xff;
  }
}

boolean readID(){
    if ( ! mfrc522.PICC_IsNewCardPresent())
    {
      return false;
    }
    if ( ! mfrc522.PICC_ReadCardSerial())
    {
      return false;
    }
    tagID = "";
    for ( uint8_t i = 0; i < 4; i++)
    {
      tagID.concat(String(mfrc522.uid.uidByte[i], HEX)); a single String
    }
    tagID.toUpperCase();
```

```
    mfrc522.PICC_HaltA();
    return true;
}
```

Prilog 2 – izvorni kod rješenja ESP8266

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
ESP8266WebServer server(80);

const char* ssid = "";
const char* password = "";

int desiredTemperature;
int temperature;
int humidity;
bool isHeating = false;
bool isCooling = false;
bool weather = false;
bool ambient = false;
bool isRaised_window = false;
bool isLowered_window = false;
bool isMotion_detected = false;
bool isLed1On = false;
bool isLed2On = false;
bool isHouseLocked = false;

void handleRoot() {

  String heatingState = isHeating ? "On" : "Off";
  String coolingState = isCooling ? "On" : "Off";
  String weatherState = weather ? "Raining" : "Not Raining";
  String lightStatus = ambient ? "Day" : "Night";
  String TwinStatus = isRaised_window ? "Yes" : "No";
  String BwinStatus = isLowered_window ? "Yes" : "No";
  String pirStatus = isMotion_detected ? "Yes" : "No";
  String lockStatus = isHouseLocked ? "Yes" : "No";
  String Led1Status = isLed1On ? "On" : "Off";
  String Led2Status = isLed2On ? "On" : "Off";
  String desiredTemp = String(desiredTemperature, 1);
```

```
  String content = "<html><head><title>Smart House Project</title>";
  content += "<meta charset=\"UTF-8\"></head><body>";
  content += "<h1>Smart House Project</h1>";
        content       +=       "<button       onclick=\"refreshData()\">Refresh
Data</button><br><br>";
  content += "<h2>Temperature and Humidity</h2>";
  content += "<p>Temperature: " + String(temperature) + " °C</p>";
  content += "<p>Humidity: " + String(humidity) + " %</p>";
  content += "<h2>Ambient and Weather</h2>";
  content += "<p>Ambient: " + lightStatus + "</p>";
  content += "<p>Weather: " + weatherState + "</p>";
  content += "<h2>Window Status</h2>";
  content += "<p>Raised window: " + TwinStatus + "</p>";
  content += "<p>Lowered window: " + BwinStatus + "</p>";
  content += "<h2>PIR Status</h2>";
  content += "<p>Motion detected: " + pirStatus + "</p>";
  content += "<h2>Heating and Cooling</h2>";
  content += "<p>Heating: " + heatingState + "</p>";
  content += "<p>Cooling: " + coolingState + "</p>";
  content += "<h2>Manual Temperature Controls</h2>";
   content  +=  "<button  onclick=\"turnsistemOff()\">Turn  Off  Heating  or
Cooling</button><br><br>";
     content    +=    "<button    onclick=\"heatingmanual()\">Turn    Heating
on</button><br><br>";
     content    +=    "<button    onclick=\"coolingmanual()\">Turn    Cooling
on</button><br><br>";
  content += "<h2>Desired Temperature</h2>";
   content  +=  "<p>Desired  Temperature:  "  +  String(desiredTemperature)  +  "
°C</p>";
  content += "<p>Input desired temperature: <input type=\"number\" step=\"1\"
id=\"desiredTemp\" value=\"" + desiredTemp + "\"> °C</p>";
     content    +=    "<button    onclick=\"setDesiredTemp()\">Set    Desired
Temp</button><br><br>";
  content += "<h2>Light Controls</h2>";
  content += "<h3>Light 1</h3>";
  content += "<p>State: " + Led1Status + "</p>";
      content     +=     "<button     onclick=\"toggleLed1()\">Toggle     Light
1</button><br><br>";
  content += "<h3>Light 2</h3>";
  content += "<p>State: " + Led2Status + "</p>";
      content     +=     "<button     onclick=\"toggleLed2()\">Toggle     Light
2</button><br><br>";
  content += "<h2>Security Status</h2>";
```

```
  content += "<p>House locked: " + lockStatus + "</p>";
  content += "<h2>Window Motor Controls</h2>";
      content     +=     "<button     onclick=\"turnMotorOff()\">Turn     Motor
Off</button><br><br>";
  content += "<button onclick=\"openMotor()\">Open</button><br><br>";
  content += "<button onclick=\"closeMotor()\">Close</button><br><br>";
  content += "<script>";
  content += "function refreshData() {";
  content += "  var xhr = new XMLHttpRequest();";
  content += "  xhr.open('GET', '/refreshData', true);";
  content += "  xhr.send();";
  content += "  setTimeout(function() { location.reload(); }, 2000);";
  content += "}";
  content += "function setDesiredTemp() {";
          content        +=        "          var        desiredTemp        =
document.getElementById('desiredTemp').value;";
  content += "  var xhr = new XMLHttpRequest();";
  content += "  xhr.open('POST', '/setDesiredTemp', true);";
  content += "  xhr.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');";
  content += "  xhr.send('desiredTemp=' + desiredTemp);";
  content += "  refreshData();";
  content += "}";
  content += "function toggleLed1() {";
  content += "  var xhr = new XMLHttpRequest();";
  content += "  xhr.open('POST', '/ledControl', true);";
  content += "  xhr.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');";
  content += "  xhr.send('ledAction=led1_toggle');";
  content += "  refreshData();";
  content += "}";
  content += "function toggleLed2() {";
  content += "  var xhr = new XMLHttpRequest();";
  content += "  xhr.open('POST', '/ledControl', true);";
  content += "  xhr.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');";
  content += "  xhr.send('ledAction=led2_toggle');";
  content += "  refreshData();";
  content += "}";
  content += "function turnMotorOff() {";
  content += "  var xhr = new XMLHttpRequest();";
  content += "  xhr.open('POST', '/motorControl', true);";
```

```
  content += "  xhr.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');";
  content += "  xhr.send('motorAction=off');";
  content += "}";
  content += "function openMotor() {";
  content += "  var xhr = new XMLHttpRequest();";
  content += "  xhr.open('POST', '/motorControl', true);";
  content += "  xhr.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');";
  content += "  xhr.send('motorAction=open');";
  content += "}";
  content += "function closeMotor() {";
  content += "  var xhr = new XMLHttpRequest();";
  content += "  xhr.open('POST', '/motorControl', true);";
  content += "  xhr.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');";
  content += "  xhr.send('motorAction=close');";
  content += "}";
  content += "function turnsistemOff() {";
  content += "  var xhr = new XMLHttpRequest();";
  content += "  xhr.open('POST', '/manualHC', true);";
  content += "  xhr.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');";
  content += "  xhr.send('manualAction=off');";
  content += "  refreshData();";
  content += "}";
  content += "function heatingmanual() {";
  content += "  var xhr = new XMLHttpRequest();";
  content += "  xhr.open('POST', '/manualHC', true);";
  content += "  xhr.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');";
  content += "  xhr.send('manualAction=heating');";
  content += "  refreshData();";
  content += "}";
  content += "function coolingmanual() {";
  content += "  var xhr = new XMLHttpRequest();";
  content += "  xhr.open('POST', '/manualHC', true);";
  content += "  xhr.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');";
  content += "  xhr.send('manualAction=cooling');";
  content += "  refreshData();";
  content += "}";
  content += "</script>";
```

```
  content += "</body></html>";

  server.send(200, "text/html", content);
}

void handleLedControl() {
  if (server.hasArg("ledAction")) {
    String action = server.arg("ledAction");
    if (action == "led1_toggle") {
      Serial.print("A");
    } else if (action == "led2_toggle") {
      Serial.print("B");
    }
  }
  server.send(200, "text/plain", "");
}

void handleMotorControl() {
  if (server.hasArg("motorAction")) {
    String action = server.arg("motorAction");
    if (action == "off") {
      Serial.print("E");
    } else if (action == "open") {
      Serial.print("C");
    } else if (action == "close") {
      Serial.print("D");
    }
  }
  server.send(200, "text/plain", "");
}

void handleSetDesiredTemp() {
  if (server.hasArg("desiredTemp")) {
    desiredTemperature = server.arg("desiredTemp").toInt();
  }
  server.send(200, "text/plain", "");
  sendDesiredTemp();
}

void handleManualControl() {
  if (server.hasArg("manualAction")) {
    String action = server.arg("manualAction");
    if (action == "off") {
```

```
      Serial.print("H");
    } else if (action == "heating") {
      Serial.print("F");
    } else if (action == "cooling") {
      Serial.print("G");
    }
  }
  server.send(200, "text/plain", "");
}

void handleRefreshData() {
  Serial.print("L");
  server.send(200, "text/plain", "");
}

void setup() {

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

  Serial.println("WiFi connected!");
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  server.on("/", handleRoot);
  server.on("/ledControl", handleLedControl);
  server.on("/motorControl", handleMotorControl);
  server.on("/setDesiredTemp", handleSetDesiredTemp);
  server.on("/refreshData", handleRefreshData);
  server.on("/manualHC", handleManualControl);

  server.begin();
  Serial.begin(115200);
}

void loop() {
  server.handleClient();

  while (Serial.available() > 0) {
    char data = (char)Serial.read();
```

```
if (data == 'I' ){
  ambient = true;
}

if (data == 'K' ){
  ambient = false;
}

if (data == 'L' ){
  weather = true;
}

if (data == 'M' ){
  weather = false;
}

if (data == 'N' ){
  isRaised_window = true;
  isLowered_window = false;
}

if (data == 'O' ){
  isRaised_window = false;
  isLowered_window = true;
}

if (data == 'P' ){
  isMotion_detected = true;
}

if (data == 'R' ){
  isMotion_detected = false;
}

if (data == 'S' ){
  isHouseLocked = true;
}

if (data == 'Q' ){
  isHouseLocked = false;
}
```

```
    if (data == 'T' ){
      isLed1On = true;
    }

    if (data == 'U' ){
      isLed1On = false;
    }

    if (data == 'V' ){
      isLed2On = true;
    }

    if (data == 'W' ){
      isLed2On = false;
    }

    if (data == 'Y'){
      readDHTtemp();
    }

    if (data == 'Z'){
      readDHThum();
    }

    if (data == 'f' ){
      isHeating = true;
      isCooling = false;
    }

    if (data == 'g' ){
      isHeating = false;
      isCooling = true;
    }

    if (data == 'h' ){
      isHeating = false;
      isCooling = false;
    }
  }
}

void readDHTtemp(){
  byte bytes[2];
```

```
  Serial.readBytes(bytes,2);
  temperature = bytesToInt(bytes, false);
}

void readDHThum(){
  byte bytes[2];
  Serial.readBytes(bytes,2);
  humidity = bytesToInt(bytes, false);
}

void sendDesiredTemp() {
  byte bytes[2];
  intToBytes(bytes, desiredTemperature, false);
  Serial.print("X");
  Serial.write(bytes,2);
}

float bytesToInt(byte bytes[], bool smallEndian){
  if (!smallEndian)
    return (  ( (bytes[0] & 0xFF) << 8) | (bytes[1] & 0xFF) );

  return (  ( (bytes[1] & 0xFF) << 8) | (bytes[0] & 0xFF) );
}

void intToBytes(byte bytes[], int num, bool smallEndian){

  unsigned int * len = (unsigned int*) &num;

  if (!smallEndian){
    bytes[0] = (*len) >> 8 & 0xff;
    bytes[1] = (*len) & 0xff;
  }else {
    bytes[1] = (*len) >> 8 & 0xff;
    bytes[0] = (*len) & 0xff;
  }
}
```