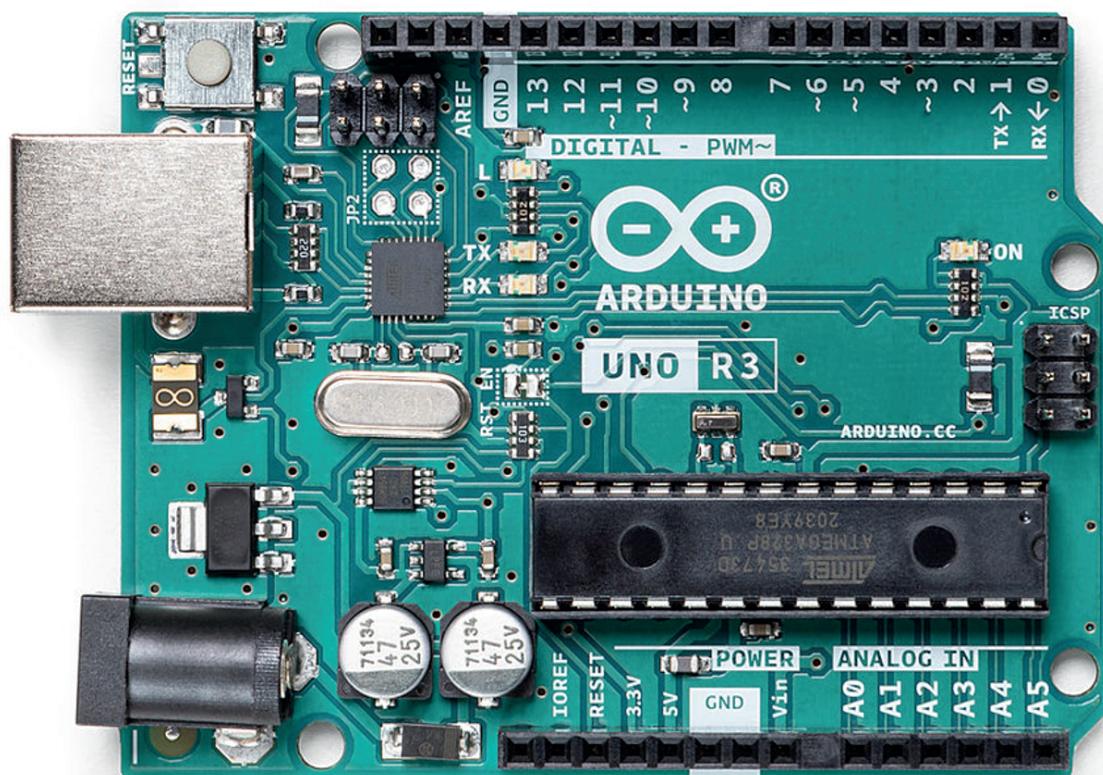




ARDUINO



Tonko Kovačević

Tonko Kovačević

ARDUINO



Projekt je sufinancirala Europska unija iz Europskog socijalnog fonda.
Sadržaj publikacije isključiva je odgovornost Obrtne tehničke škole Split



EduSplit Obrtna tehnička škola
Regionalni centar kompetentnosti Split

Tonko Kovačević

ARDUINO

Priručnik za program obrazovanja Arduino.

Split, 2023.

Autor: Tonko Kovačević, prof. str. st.

Urednik: (za ALGEBRA d.o.o.)

Naslov: **Arduino**

Recenzent: Tonči Kozina, dipl. ing. el.

Lektorica: Blaženka Bračun, prof.

Grafičko oblikovanje: ALGEBRA d.o.o.

Nakladnik: Obrtna tehnička škola Split

Odgovorna osoba: ravnatelj Milivoj Kalebić

Za nakladnika: ALGEBRA d.o.o.

Više informacija:

Obrtna tehnička škola Split

Plančićeva 21

21000 Split

e-pošta: ured@ss-obrtna-tehnicka-st.skole.hr

mrežna adresa: edusplit.eu

ISBN: 978-953-8537-05-9

Regionalni centar kompetentnosti Obrtne tehničke škole

Split, 2023.

Obrtnička tehnička škola, Plančićeva 21, 21000 Split, OIB: 43651407703, nositelj je isključivog prava iskorištavanja ovog autorskog djela, prostorno, vremenski i sadržajno neograničeno, a koje pravo obuhvaća imovinska prava autora i to osobito, ali ne isključivo, pravo reproduciranja (pravo umnožavanja), pravo distribuiranja (pravo stavljanja u promet), pravo priopćavanja autorskog djela javnosti te pravo prerade. Pojedina imovinska autorska prava treća osoba može steći isključivo na temelju pisane suglasnosti Obrtničke tehničke škole.

Sadržaj

1. ARDUINO PLATFORMA	9
1.1. Arduino arhitektura	10
1.1.1. Arhitektura Aduino R3 pločice	10
1.1.2. Instalacija programske podrške	12
1.2.2. Kontrola pauza u izvođenju programa.....	16
1.2.3. Kontrola programa s funkcijama za mjerenje vremena	17
1.2.4. Unos podataka putem serijske komunikacije.....	19
1.2.5. Rad s EEPROM memorijom.....	21
2. PRIMJENA SENZORA	23
2.1. MJERENJE OSVJETLJENJA	24
2.2. MJERENJE UDALJENOSTI	26
2.3. MJERENJE TEMPERATURE	30
2.3.1. Senzor DS18B20	30
2.3.2. NTC termistor	35
2.3.3. Mjerenje temperature i vlažnosti pomoću DHT22 senzora	39
2.3.4. Senzor BME280 I2C	43
2.3.5. Senzor BME280 SPI	48
2.4. MJERENJE RAZINE TEKUĆINE	52
3. PRIMJENA ULAZNO-IZLAZNIH JEDINICA	55
3.1. KONTROLA ULAZNO-IZLAZNIH PRIKLJUČAKA	56
3.2. PRIMJENA MEMBRANSKE TIPKOVNICE	58
3.3. ARDUINO ANALOGNO-DIGITALNA PRETVORBA	62
3.4. PRIMJENA LCD ZASLONA	64
3.4.1. LCD zaslon.....	64
3.4.2. Aplikacija za ispis posebnih znakova na LCD zaslon	68
3.4.3. Primjena I2C komunikacije za povezivanje LCD zaslona	71
4. ARDUINO PROJEKTI	77
4.1. TERMOSTAT ZA UPRAVLJANJE SUSTAVOM GRIJANJA ILI HLAĐENJA.....	78
4.1.1. Projektni zadatak	78
4.1.2. Dizajn projekta	78
4.1.3. Izvedba projekta.....	82
4.2. APLIKACIJA ZA UPRAVLJANJE KORAČNIM MOTOROM	86
4.2.1. Projektni zadatak	86
4.2.2. Dizajn projekta	86
4.2.3. Aplikacija za upravljanje koračnim motorom	91
Popis elemenata korištenih u sadržaju	94
Popis literature	96

1

POGLAVLJE

ARDUINO PLATFORMA

Nakon ovog poglavlja moći ćete:

- objasniti arhitekturu Arduino R3 pločice
- opisati način razvoja programa u Arduino IDE razvojnoj okolini
- izraditi, prevesti, učitati i izvršiti Arduino program
- razumjeti primjene funkcija za rad s vremenom i EEPROM memorijom.

1.1. ARDUINO ARHITEKTURA

1.1.1. Arhitektura Aduino R3 pločice

Arduino je tehnologija otvorenog koda čija je platforma zasnovana na hardveru i softveru koji je fleksibilan i jednostavan za korištenje. Ova platforma namijenjena je za učenike, studente, inženjere, umjetnike i sve zainteresirane koji žele stvarati interaktivne objekte ili okruženje, a naročito je pogodna za razvoj aplikacija Interneta stvari IoT (eng. *Internet of Things*). Arduino platforma u stalnom je razvoju i do danas postoji više od dvadeset Arduino mikrokontrolerskih pločica za različitu namjenu (Arduino, 2023). Osnovne skupine Arduino pločica su:

- **Arduino NANO:** različite izvedbe s podrškom Wi-Fi, *Bluetooth* i IoT tehnologije
- **Arduino MKR:** različite izvedbe s podrškom Wi-Fi, LoRAWAN, GSM i LTE tehnologije
- **Arduino UNO:** različite izvedbe, a posljednja je inačica Arduino Uno R4
- **Arduino MEGA:** različite izvedbe za projekte koji zahtijevaju veću računalnu snagu i veći broj digitalnih ulaza/izlaza (54, od čega je 15 PWM) i analognih ulaza (16).

Osnovni model koji se koristi u ovom priručniku jest Arduino Uno R3 (slika 1-1). Za programiranje Arduina koristi se prilagođeni C++ programski jezik i razvojna okolina Arduino IDE. Karakteristike Arduino Uno R3 platforme su (tablica 1-1):

1. mikrokontroler:

- **model:** ATMEL ATmega328P
- **arhitektura:** 8-bitna
- **kapacitet programske memorije:** 32 kB *flash*-memorije

2. memorija:

- **programska memorija:** 32 kB
- **podatkovna memorija (SRAM):** 2 kB
- **EEPROM:** 1 kB

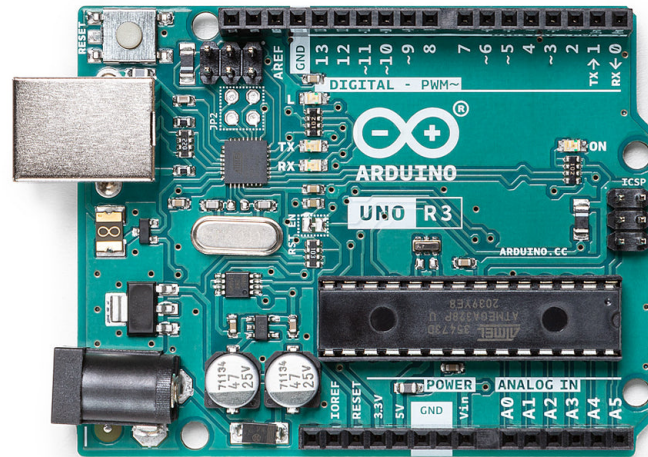
3. ulazi/izlazi:

- **digitalni ulazi/izlazi:** 14 digitalnih pinova koji se mogu konfigurirati kao ulazi ili izlazi, a od čega je 6 PWM
- **analogni ulazi:** 6 analognih ulaza za mjerenje analognih signala

4. ostale karakteristike:

- **frekvencija:** 16 MHz keramički oscilator koji osigurava osnovni takt mikrokontrolera
- **USB priključak:** omogućuje povezivanje s računalom i programiranje mikrokontrolera
- **napajanje:** ima priključak za napajanje

- **ICSP sučelje:** ICSP (eng. *In-Circuit Serial Programming*) sučelje omogućuje programiranje mikrokontrolera bez vađenja iz sklopa
- **tipka za reset:** omogućuje ručno resetiranje mikrokontrolera.



Slika 1.1: Arduino Uno R3 (izvor: <https://www.arduino.cc/>)

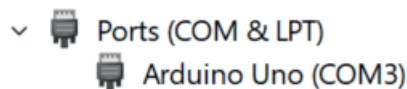
Tablica 11: Tehničke karakteristike Arduina Uno R3

Parametar	Vrijednost
mikrokontroler	ATmega328
radni napon	5 V
ulazni napon (preporučeno)	7 – 12 V
ulazni napon (ograničenje)	6 – 20 V
digitalni I/O pinovi	14 (od čega 6 omogućava PWM izlaz)
analogni input pinovi	6
DC struja za I/O pin	40 mA
DC struja za 3.3 V pin	50 mA
flash-memorija	32 KB (ATmega328), od čega se 0,5 KB koristi pri učitavanju i podizanju sustava
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
brzina	16 MHz

1.1.2. Instalacija programske podrške

Za korištenje Arduino IDE (Arduino razvojne okoline) potrebno je slijediti sljedeću proceduru:

1. **Spajanje na računalo:** Arduino pločice povezuju se **s računalom** USB kabelom. Za Arduino Uno R3 pločicu koristi se standardni USB kabel (A konektor na B konektor). Arduino Uno R3 pločica ima automatsko napajanje preko USB kabela. Na samoj Arduino pločici postoji i dodatni konektor za eksterno napajanje. Mogući je ulazni napon od 6 V do 20 V. Preporučeni je napon od 7 V do 12 V. Nakon spajanja pločice na računalo zelena LED dioda (PWR) stalno svijetli.
2. **Arduino softver:** upravljački softver olakšava pisanje Arduino koda i njegovo upisivanje u Arduino pločicu. Postoje posebne verzije softvera za *Windows*, *Mac OS X* i *Linux*. Trenutno posljednja verzija za *Windows* jest Arduino IDE 2.2.1. Upravljački softver može se pronaći na službenoj Arduinovoj stranici koristeći poveznicu <http://arduino.cc/en/Main/Software>.
3. **Instalacija pogonskog programa:** nakon instalacije Arduino IDE 2.2.1 softvera potrebno je dodati upravljački program (eng. *driver*) na sljedeći način:
 - u *Windows*ima je potrebno u **Start Menu** otvoriti **Device Manager**. **Device Manager** može se otvoriti i kroz odabir **Control Panela**, zatim **System and Security** pa **System**, i na kraju **Device Manager**
 - pod opcijom **Ports (COM & LPT)** treba se nalaziti prikaz portova kao na sljedećoj slici:



Slika 1.2. Oznake portova

- ako priključci **COM & LPT** nisu vidljivi, potrebno ih je potražiti pod **Other Devices** i **Unknown Device**
- desnim klikom na **COM** priključak pridružen Arduino pločici odabere se opcija **Update Driver Software**
- odabere se opcija **Browse my computer for Driver software**
- na kraju se pronađe pogonski program nazvan **arduino.inf** koji je lociran u **Drivers** mapi unutar prethodno skinutog Arduino softvera. Klikom na tu datoteku *Windows* će dovršiti instalaciju programa.

4. **Pokretanje programa:** program se pokreće klikom na ikonu na slici 1-3.

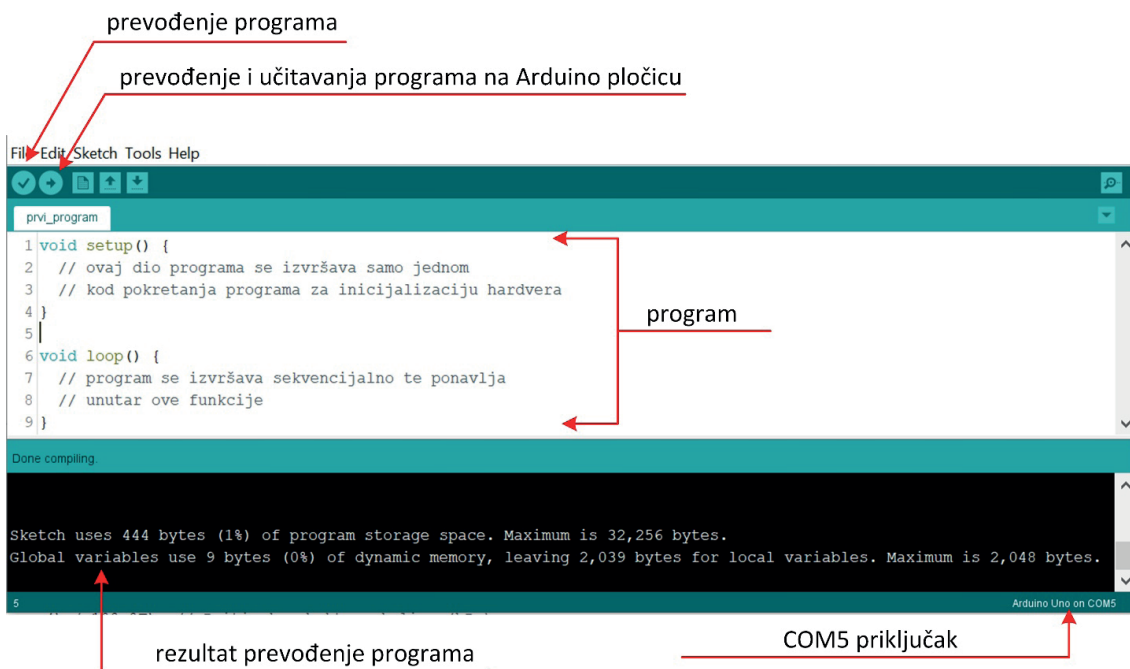


Slika 1.3: Ikona Arduino IDE (izvor: „Autorski rad“)

4 RAZVOJ PROGRAMA

Izrada programa

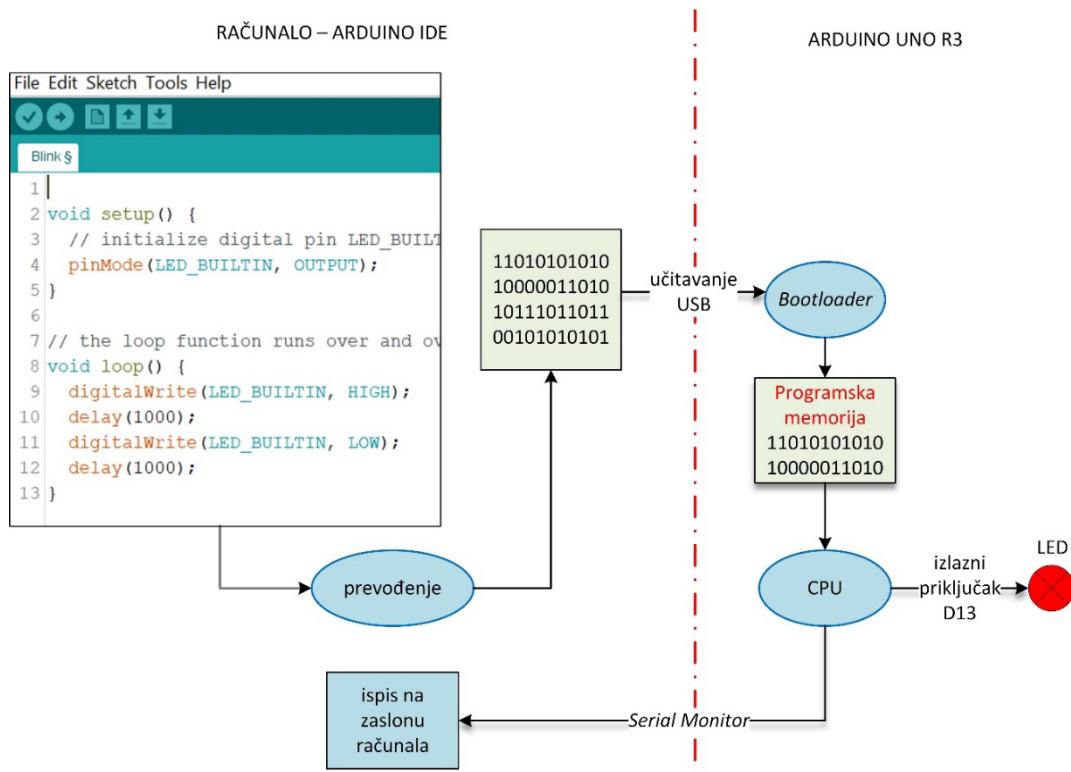
Na slici 1-4 prikazan je osnovni Arduino program koji se sastoji od samo dviju funkcija, *setup()* i *loop()*. Ove dvije funkcije obavezne su i mora ih sadržavati svaki Arduino program. Ne smije se mijenjati naziv funkcija.



Slika 1.4: Osnovni Arduino program (izvor: „Autorski rad“)

Unutar *setup()* funkcije piše se kôd koji se odnosi na inicijalizaciju hardvera i ovaj se kôd izvršava samo jednom, pokretanjem programa. Tip podataka *void* ispred *setup()* funkcije označava da funkcija izvršava naredbe unutar sebe, ali kao rezultat izvršavanja ne vraća programu nikakvu vrijednost. Sav programski kôd unutar jedne cjeline, bloka naredbi ili funkcije, u C++ jeziku nalazi se unutar vitičastih zagrada `{}`. Oznaka `//` označava komentar koji se piše u jednoj liniji kôda i njega računalo ignorira, a služi programerima za lakše razu-

mijevanje programa. Ako se komentar proteže na više linija kôda, potrebno je na početku svake linije navesti oznaku komentara `//` ili na početku komentara navesti oznaku `/*` i napisati komentar u više linija te na kraju komentara navesti oznaku `*/`. Kôd unutar `loop()` funkcije sekvencionalno se izvršava i ciklički ponavlja. Kada izvršavanje dođe do kraja funkcije, ponovno se vraća na početak i tako neprekidno izvršava. Nakon što je program napisan, potrebno ga je prevesti (eng. *compile*) klikom na ikonu prikazanu na slici 1-4. Ako je prevođenje bilo uspješno, u Arduino IDE sučelju pojavljuje se poruka **Done compiling**. U prozor ispod ove poruke ispisuju se podaci o upotrijebljenim i raspoloživim Arduino resursima. Posljednji korak predstavlja odabir odgovarajućeg komunikacijskog priključka (eng. *COM port*) i učitavanje (eng. *upload*) programa na Arduino pločicu klikom na ikonu prikazanu na slici 1-4. Na slici 1-5 prikazan je cjelokupan proces razvoja programa i njegova izvođenja na Arduino pločici.



Slika 1.5: Razvoj i izvođenje Arduino programa (izvor: „Autorski rad“)

U programskom kôdu 1-1 dan je program koji kontrolira paljenje i gašenje ugrađene LED diode na Arduino pločicu s detaljnim komentarima. Za ispis poruka na zaslonu računala potrebno je uključiti opciju u izborniku **Tools->Serial Monitor** ili pritiskom na kombinaciju tipki **Ctrl+Shift+M**. U **Serial Monitoru** potrebno je postaviti komunikacijsku brzinu na 9600 bauda koja je postavljena unutar `setup()` funkcije, slika 1-6.

Programski kôd 11: Program za kontrolu LED diode

```
/* Program za kontrolu LED diode ugrađene
 * na Arduino pločicu. Dioda se pali i gasi svake sekunde!
 */
void setup() {
  //Inicijalizacija komunikacije za ispis na zaslonu računala
  Serial.begin(9600);
  //Inicijalizacija digitalnog priključka LED_BUILTIN kao izlaznog
  pinMode(LED_BUILTIN, OUTPUT);
}
// loop funkcija se izvršava bez prekida
void loop() {
  //upali LED diodu
  digitalWrite(LED_BUILTIN, HIGH);
  //Ispiši na zaslon računala
  Serial.println("LED dioda upaljena");
  //čekaj jednu sekundu
  delay(1000);
  //ugasi LED diodu
  digitalWrite(LED_BUILTIN, LOW);
  //Ispiši na zaslon računala
  Serial.println("LED dioda ugašena");
  //čekaj jednu sekundu
  delay(1000);
}
```



Slika 1.6: Ispis na zaslonu računala (izvor: „Autorski rad“)

1.2.2. Kontrola pauza u izvođenju programa

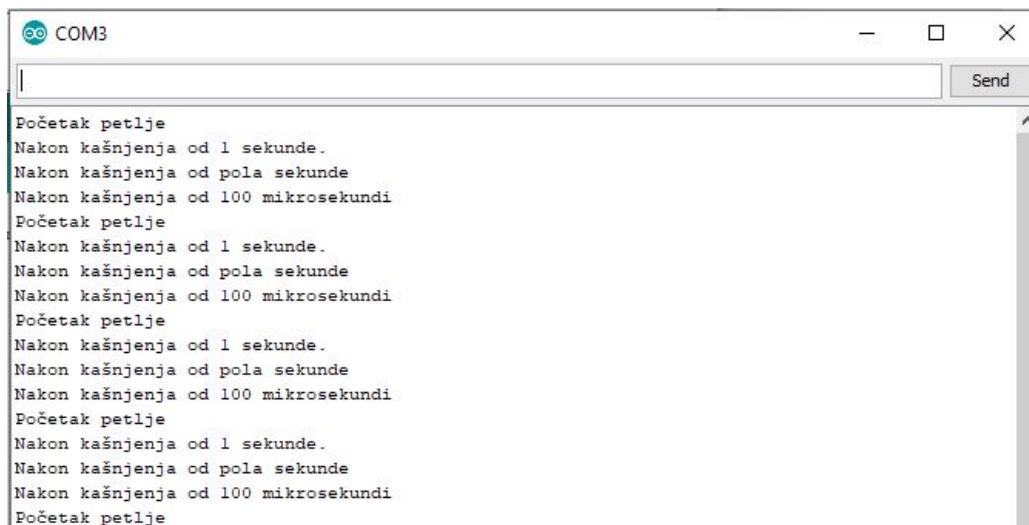
Funkcije `delay()` i `delayMicroseconds()` u Arduino programskom okruženju koriste se za stvaranje pauza u izvođenju programa, programski kôd 1-2.

Programski kôd 12: Kontrola pauza u izvođenju programa

```
// Kontrola pauza u izvođenju programa
void setup() {
  // Inicijalizacija serijske komunikacije s brzinom od 9600 bauda
  Serial.begin(9600);
}

void loop() {
  // Prikaz poruke na serijskom monitoru
  Serial.println("Početak petlje");
  // Pauza od 1000 milisekundi (1 sekunda)
  delay(1000);
  // Prikaz poruke na serijskom monitoru
  Serial.println("Nakon kašnjenja od 1 sekunde.");
  // Pauza od 500 milisekundi
  delay(500);
  // Prikaz poruke na serijskom monitoru
  Serial.println("Nakon kašnjenja od pola sekunde");
  // Pauza od 100 mikrosekundi
  delayMicroseconds(100);
  // Prikaz poruke na serijskom monitoru
  Serial.println("Nakon kašnjenja od 100 mikrosekundi");
}
```

Nakon učitavanja i izvođenja programa na zaslonu računala ispisuju se poruke prikazane na slici 1-7. U ovom primjeru koristi se funkcija `delay()` za stvaranje pauze u trajanju od 1 sekunde (1000 milisekundi) među porukama na serijskom monitoru. Zatim slijedi još jedna pauza od 500 milisekundi. Nakon toga koristi se funkcija `delayMicroseconds()` za vrlo kratku pauzu od 100 mikrosekundi. Bitno je napomenuti da funkcija `delayMicroseconds()` nije dostupna na svim Arduino pločicama, posebice onima koje koriste ATmega8 kao mikrokontroler. Uglavnom se koristi na pločicama koje koriste ATmega168, ATmega328, ATmega1280, ATmega2560 i slične mikrokontrolere.



```
COM3
Početak petlje
Nakon kašnjenja od 1 sekunde.
Nakon kašnjenja od pola sekunde
Nakon kašnjenja od 100 mikrosekundi
Početak petlje
Nakon kašnjenja od 1 sekunde.
Nakon kašnjenja od pola sekunde
Nakon kašnjenja od 100 mikrosekundi
Početak petlje
Nakon kašnjenja od 1 sekunde.
Nakon kašnjenja od pola sekunde
Nakon kašnjenja od 100 mikrosekundi
Početak petlje
Nakon kašnjenja od 1 sekunde.
Nakon kašnjenja od pola sekunde
Nakon kašnjenja od 100 mikrosekundi
Početak petlje
```

Slika 1.7: Ispis pauza na zaslonu računala (izvor: „Autorski rad“)

1.2.3. Kontrola programa s funkcijama za mjerenje vremena

Korištenje funkcija `millis()` i `micros()` umjesto `delay()` i `delayMicroseconds()` na Arduino platformi često se preporučuje iz nekoliko važnih razloga:

1. Nemogućnost izvođenja drugih zadataka tijekom pauze:

- funkcije `delay()` i `delayMicroseconds()` zaustavljaju izvođenje programa tijekom trajanja pauze. Ako je potrebno obaviti drugi zadatak, na primjer, čitanje vrijednosti sa senzora ili ostvariti komunikaciju s drugim uređajem, korištenje `delay()` može biti problematično jer blokira izvođenje drugih dijelova programa.

2. Pouzdanije mjerenje vremena:

- funkcije `millis()` i `micros()` omogućuju preciznije mjerenje vremena u odnosu na funkcije `delay()` i `delayMicroseconds()`. To je posebno važno ako se žele izbjeći pogreške koje se mogu pojaviti kada se koriste pauze u programu.

3. Omogućuje izvođenje više zadataka i paralelno izvršavanje programa:

- funkcije `millis()` i `micros()` omogućuju da se prati proteklo vrijeme bez zaustavljanja izvođenja ostatka kôda. Ovo je ključno za implementaciju programa koji omogućavaju izvođenje više zadataka (engl. *Multitasking*) ili paralelnog izvršavanja više zadataka istovremeno.

4. Upravljanje događajima:

- korištenjem funkcija `millis()` i `micros()` mogu se jednostavno implementirati sustavi za upravljanje događajima po određenom rasporedu, npr. izvršavanje određenog kôda svakih 60 sekundi.

Primjer korištenja ovih funkcija dan je u programskom kôdu 1-3. Ovaj primjer pokazuje kako stvarati vremenske intervale u programu, mjeriti vrijeme među događajima i upravljati događajima na temelju proteklog vremena. U ovom primjeru funkcija *millis()* koristi se za mjerenje proteklog vremena u milisekundama, dok se funkcija *micros()* koristi za mjerenje vremena u mikrosekundama. Program provjerava je li prošao određeni vremenski interval primjenom ovih dviju funkcija i ispisuje poruku na zaslon računala. Ovo je osnovni primjer kako se funkcije *millis()* i *micros()* koriste za vremensko praćenje i upravljanje događajima u Arduino kôdu.

Programski kôd 13: Primjena funkcija *millis()* i *micros()*

```
// Primjer koji koristi funkcije millis() i micros()
unsigned long previousMillis = 0;
const long intervalMillis = 1000; // Interval u milisekundama (1 sekun-
da)
unsigned long previousMicros = 0;
const long intervalMicros = 500000; // Interval u mikrosekundama (0.5
sekundi)

void setup() {
  // Inicijalizacija serijske komunikacije s brzinom od 9600 bauda
  Serial.begin(9600);
}

void loop() {
  // Mjerenje proteklog vremena u milisekundama
  unsigned long currentMillis = millis();

  // Provjera je li prošao interval u milisekundama
  if (currentMillis - previousMillis >= intervalMillis) {
    Serial.println("Prošla je jedna sekunda u millis()");
    previousMillis = currentMillis; // Ažuriranje vremena
  }

  // Mjerenje proteklog vremena u mikrosekundama
  unsigned long currentMicros = micros();

  // Provjera je li prošao interval u mikrosekundama
  if (currentMicros - previousMicros >= intervalMicros) {
    Serial.println("Prošlo je pola sekunde u micros()");
    previousMicros = currentMicros; // Ažuriranje vremena
  }
}
```

1.2.4. Unos podataka putem serijske komunikacije

Za povezivanje Arduino pločice i računala može se koristiti **Serial Monitor** koji omogućava komunikaciju između Arduino mikrokontrolera i računala putem serijskog priključka. To je vrlo koristan alat za otkrivanje grešaka u programu (engl. *Debug*), za praćenje i unos podataka u Arduino iz računala. U programskom kôdu 1-4 dan je primjer aplikacija za unos i ispis podataka putem **Serial Monitora**, slika 1-8.

Programski kôd 14: Unos i ispis podataka putem Serial Monitora

```
/*
 * Unos znakovnog, cjelobrojnog i realnog podatka te
 * unos znakovnog niza putem serijske komunikacije
 */
void setup() {
  Serial.begin(9600);
}

void loop() {
  // Unos znaka
  Serial.print("Unesi znak: ");
  while (!Serial.available()) {
    // čekaj na unos
  }
  char charInput = Serial.read();
  Serial.print("Uneseni znak: ");
  Serial.println(charInput);

  // Unos cjelobrojnog podatka
  Serial.print("Unesi cjelobrojni podatak: ");
  while (!Serial.available()) {
    // čekaj na unos
  }
  int intInput = Serial.parseInt();
  Serial.print("Uneseni cjelobrojni podatak: ");
  Serial.println(intInput);

  // Unos realnog podatka
  Serial.print("Unesi realni podatak: ");
  while (!Serial.available()) {
    // čekaj na unos
  }
  // Ispis realnog podatka s 4 decimale
  float floatInput = Serial.parseFloat();
```

```

Serial.print("Uneseni realni podatak: ");
Serial.println(floatInput, 4); mjesta

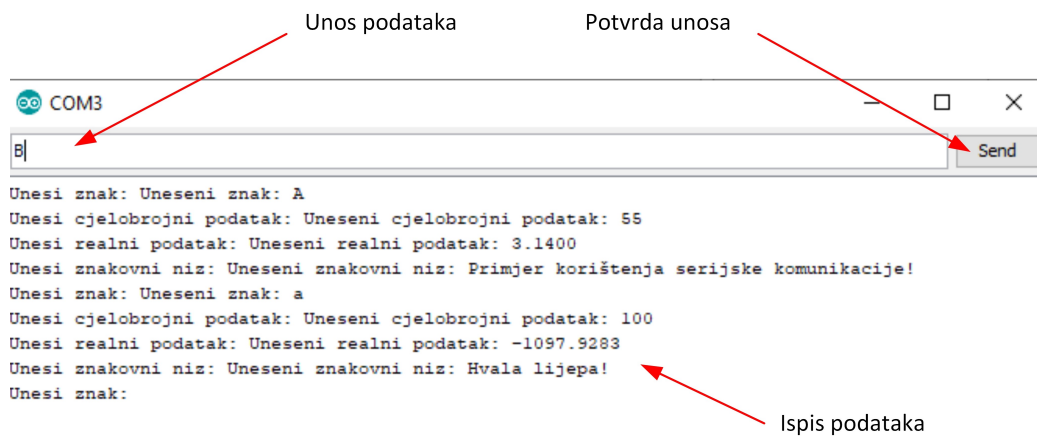
// Unos znakovnog niza
Serial.print("Unesi znakovni niz: ");
while (!Serial.available()) {
  // čekaj na unos
}
String stringInput = Serial.readString();
Serial.print("Uneseni znakovni niz: ");
Serial.println(stringInput);

// Pričekaj malo
delay(1000);
}

```

Za unos podataka koriste se sljedeće funkcije:

- *Serial.available()* – koristi se za provjeru koliko bajtova (znakova) ima u serijskom prijemnom spremniku (engl. *Buffer*) koji čekaju na čitanje, vraća vrijednost veću od 0 ako ima podataka za čitanje, a 0 ako nema podataka za čitanje. Nakon čitanja podatka s jednom od sljedećih funkcija, isti se briše iz spremnika pa sljedeće pozivanje funkcije *Serial.available()* daje kao rezultat manji broj raspoloživih podataka za čitanje.
- *Serial.read()* – koristi se za čitanje jednog bajta iz prijemnog spremnika.
- *Serial.parseInt()* – koristi se za čitanje cijelog broja (eng. *Integer*) iz serijskog prijemnog spremnika.
- *Serial.parseFloat()* – koristi se za čitanje realnog broja (engl. *Floating-Point Number*) iz serijskog prijemnog spremnika.
- *Serial.readString()* – koristi se za čitanje niza znakova (engl. *String*) iz serijskog spremnika.



Slika 1.8: Korištenje Serial Monitora za unos i ispis (izvor: „Autorski rad“)

1.2.5. Rad s EEPROM memorijom

EEPROM (engl. *Electrically Erasable Programmable Read-Only Memory*) na Arduino platformi omogućava pisanje i čitanje podataka koji se čuvaju i nakon isključivanja napajanja. Kapacitet memorije Arduino Uno R3 platforme jest 1024 bajta. Za rad s ovom memorijom potrebno je uključiti biblioteku **EEPROM.h** koja podržava sljedeće funkcije:

- `EEPROM.read(adresa)` – čita jedan bajt podataka iz memorije s navedene adrese
- `EEPROM.write(adresa, vrijednost)` – zapisuje jedan bajt podataka na navedenu adresu u memoriji
- `EEPROM.update(adresa, vrijednost)` – zapisuje jedan bajt podataka na navedenu adresu u memoriji samo ako se razlikuje od prethodno zapisane vrijednosti na toj adresi u EEPROM memoriji
- `EEPROM.get(adresa, podaci)` – čita bilo koji tip podataka iz memorije s navedene adrese
- `EEPROM.put(adresa, podaci)` – zapisuje podatke bilo kojeg tipa na navedenu adresu u memoriji
- `EEPROM[adresa]` – ovaj operator omogućava direktno čitanje ili zapisivanje u memoriju na navedenu adresu, EEPROM identifikator metoda je za rad s EEPROM memorijom kao s podatkovnim poljem
- `EEPROM.length()` – ova funkcija vraća veličinu EEPROM memorije u bajtovima.

U programskom kôdu 1-5 dan je primjer korištenja EEPROM memorije, a na slici 1-9 primjer ispisa podataka iz EEPROM memorije.

Programski kôd 15: Rad s EEPROM memorijom

```
//Rad s EEPROM memorijom
#include <EEPROM.h>

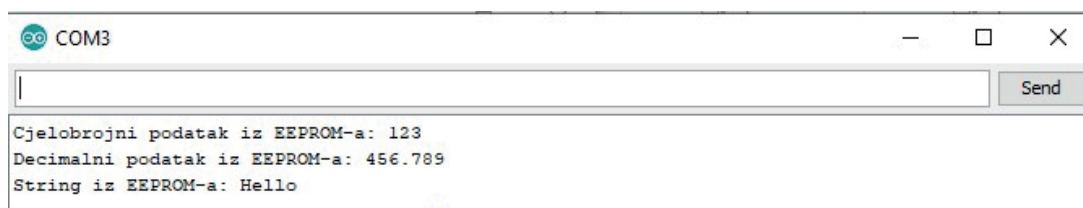
void setup() {
  Serial.begin(9600);
  // Pisanje cjelobrojnog broja u EEPROM na adresu 0
  int intValue = 123;
  EEPROM.put(0, intValue);

  // Čitanje cjelobrojnog broja iz EEPROM s adrese 0
  int readIntValue;
  EEPROM.get(0, readIntValue);
  Serial.print("Cjelobrojni podatak iz EEPROM-a: ");
```

```
Serial.println(readIntValue);
// Pisanje decimalnog broja (float) u EEPROM na adresu 4
// poslije cjelobrojnog broja koji zauzima 4 bajta
float floatValue = 456.789;
EEPROM.put(sizeof(int), floatValue);
// Čitanje decimalnog broja (float) iz EEPROM s adrese 4
float readFloatValue;
EEPROM.get(sizeof(int), readFloatValue);
Serial.print("Decimalni podatak iz EEPROM-a: ");
// Ispisujemo s tri decimalna mjesta
Serial.println(readFloatValue, 3);
// Pisanje niza znakova (string) u EEPROM na adresu 8
// poslije decimalnog broja koji zauzima 4 bajta
String stringValue = "Hello";
EEPROM.put(sizeof(int) + sizeof(float), stringValue);

// Čitanje niza znakova (stringa) iz EEPROM s adrese 8
String readStringValue;
EEPROM.get(sizeof(int) + sizeof(float), readStringValue);
Serial.print("String iz EEPROM-a: ");
Serial.println(readStringValue);
}

void loop() {
// Loop ne radi ništa u ovom primjeru
}
```



Slika 1.9: Ispis podataka iz EEPROM memorije (izvor: „Autorski rad“)

2

POGLAVLJE

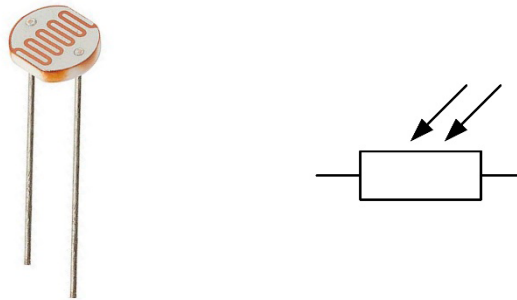
PRIMJENA SENZORA

Nakon ovog poglavlja moći ćete:

- opisati fizikalna načela rada senzora
- primijeniti senzor i odgovarajuće komunikacijsko sučelje
- izraditi aplikaciju za rad sa sensorima.

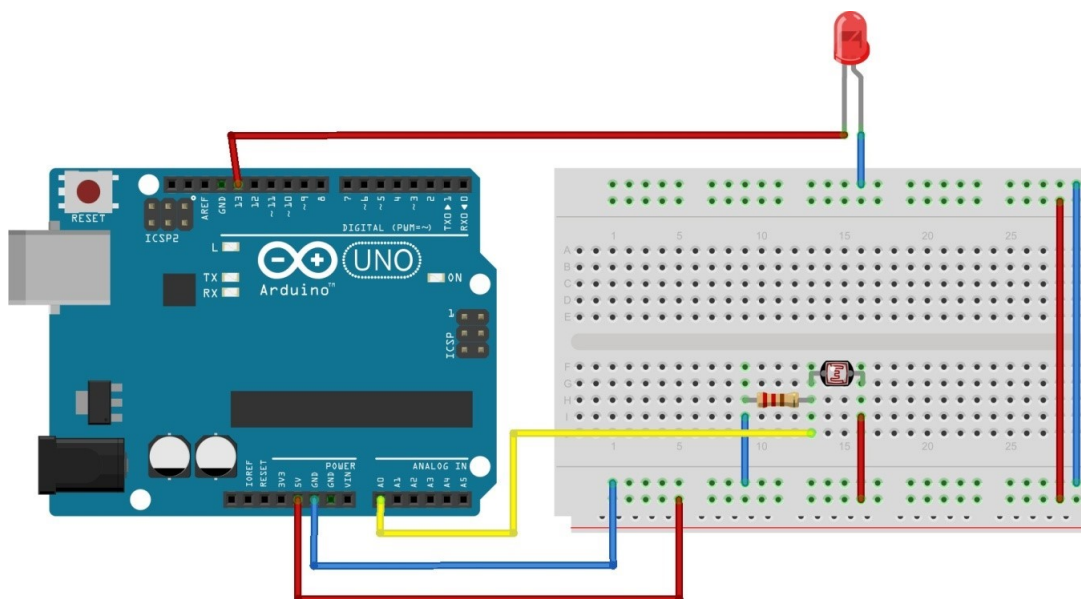
2.1. MJERENJE OSVJETLJENJA

Za mjerenje osvjetljenja u ovoj aplikaciji koristi se fotootpornik LDR (eng. *photoresistor* ili *light dependent resistor*). Ovaj otpornik izrađuje se od poluvodiča s velikim električnim otporom, a njegov se električni otpor smanjuje povećanjem intenziteta ulazne svjetlosti. Na slici 2-1 prikazan je primjer izvedbe i simbol fotootpornika.



Slika 2.1: Primjer izvedbe i simbol fotootpornika (izvor: „Autorski rad“)

Ovisno o osvjetljenju, fotootpornik pali i gasi LED diodu, slika 2-2. Otpornik od 10 k Ω i LDR dva su serijski spojena otpora koji se zbrajaju i tvore zajednički otpor, što znači da promjenjivi otpor LDR-a smanjuje ukupni otpor, ovisno o svjetlu koje obasjava LDR na najmanje 10 k Ω . Istovremeno, padom otpora rastu struja i napon u krugu. U tablici 2-1 dana je ovisnost otpora, struje i napona o osvjetljenju i izabranom otporu.



Slika 2.2: Shema spajanja svjetlosnog prekidača (izvor: „Autorski rad“)

Tablica 21: Ovisnost otpora, struje i napona o osvjetljenju i izabranom otporu

Vrsta svjetla	Inenzitet Lux	Otpor LDR kΩ	Ukupan otpor kΩ	Struja mA	Napon V
mračan hodnik	0,1	600	610	0,008 mA	0.1
noć s mjesecom	1	70	80	0,07 mA	0.6
tamna soba	10	10	20	0,25 mA	2.5
mračan oblačan dan	100	1,5	11,5	0,43 mA	4.3
oblačan dan	1000	0,3	10,3	0,5 mA	5

Programski kôd 21: Aplikacija svjetlosni prekidač

```

/* SVJETLOSNI PREKIDAČ
Ovaj program pali i gasi LED diodu u ovisnosti od postignutog
uvjeta svjetline koja pada na fotootpornik LDR.
Krug:
* LED dioda spojena na pin 13
* LDR spojen na analogni pin A0 */
// postavljanje LDR i LED pinova
int LDR_Pin = A0;
int izlaz_LED = 13;
void setup(){
  // inicijalizacija komunikacije za prikaz na zaslonu računala
  Serial.begin(9600);
  // pin 13 definiran kao izlazni
  pinMode(izlaz_LED, OUTPUT); //Definiranje varijable izlaz_LED kao izlaza
}
void loop()
{
  //čitanje osvjetljenja s analognog pina A0
  int LDRstanje = analogRead(LDR_Pin);
  // ispis osvjetljenja na zaslonu računala
  Serial.print("Trenutna osvjetljenost = ");
  Serial.println(LDRstanje);
  // kašnjenje od 10 milisekundi radi stabilnosti programa
  delay(10);
  // uvjet paljenja LED diode
  if(LDRstanje >= 900)
  {

```

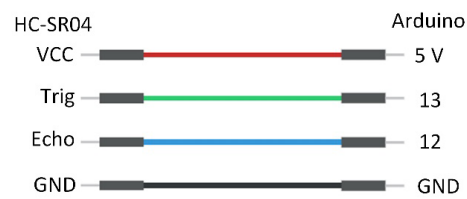
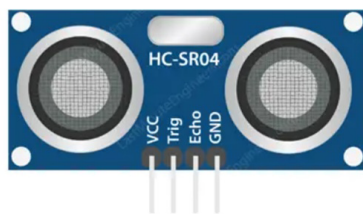
```

digitalWrite(izlaz_LED, HIGH);
}
else
{
digitalWrite(izlaz_LED, LOW);
}
}

```

2.2. MJERENJE UDALJENOSTI

Za mjerenje udaljenosti koristit će se senzor HC-SR04 prikazan na slici 2-3, a karakteristike su dane u tablici 2-2.

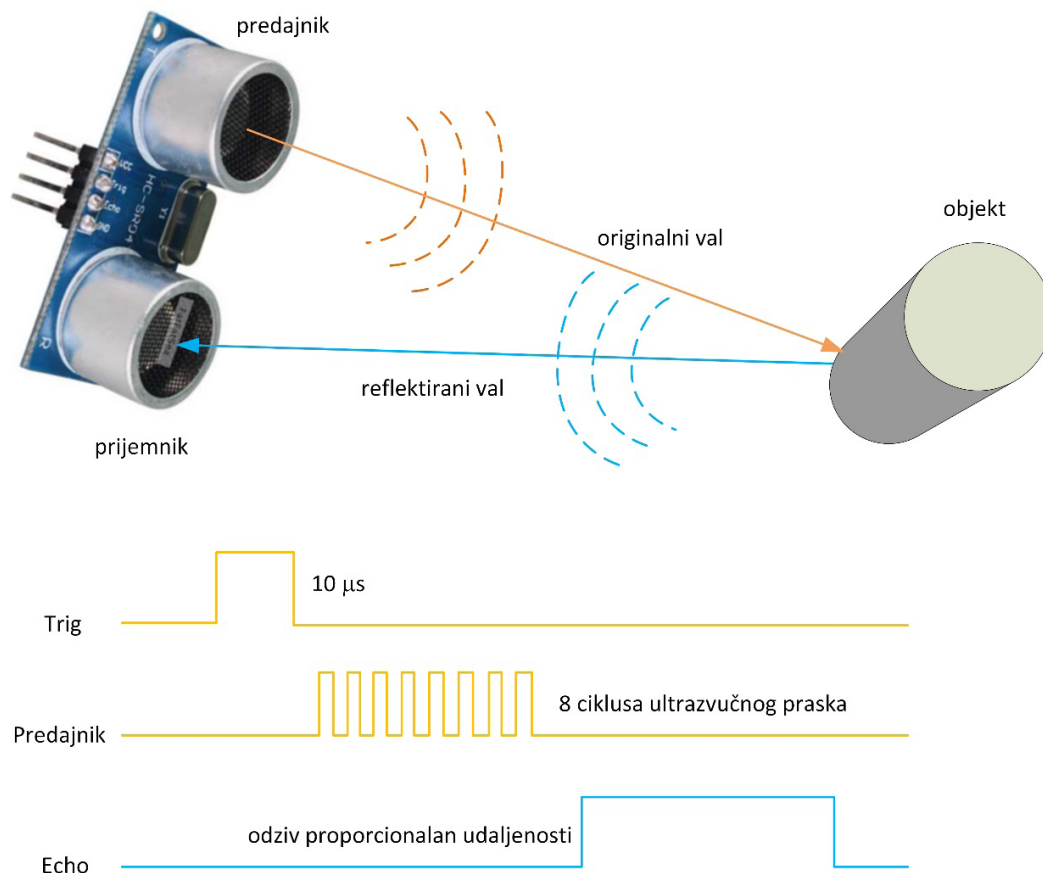


Slika 2.3: Senzor HC-SR04 (izvor: „Autorski rad“)

Tablica 22: Karakteristike senzora HC-SR04

napon	5 V DC
struja	15 mA
frekvencija	40 kHz
najmanja udaljenost	2 cm
najveća udaljenost	400 cm
točnost	3 mm
kut	< 15 °
dimenzije	45 x 20 x 15 mm

Senzor emitira ultrazvuk na frekvenciji od 40 kHz koji putuje zrakom i ako se na putu nalazi objekt ili prepreka, signal će se odbiti natrag do senzora, slika 2-4.



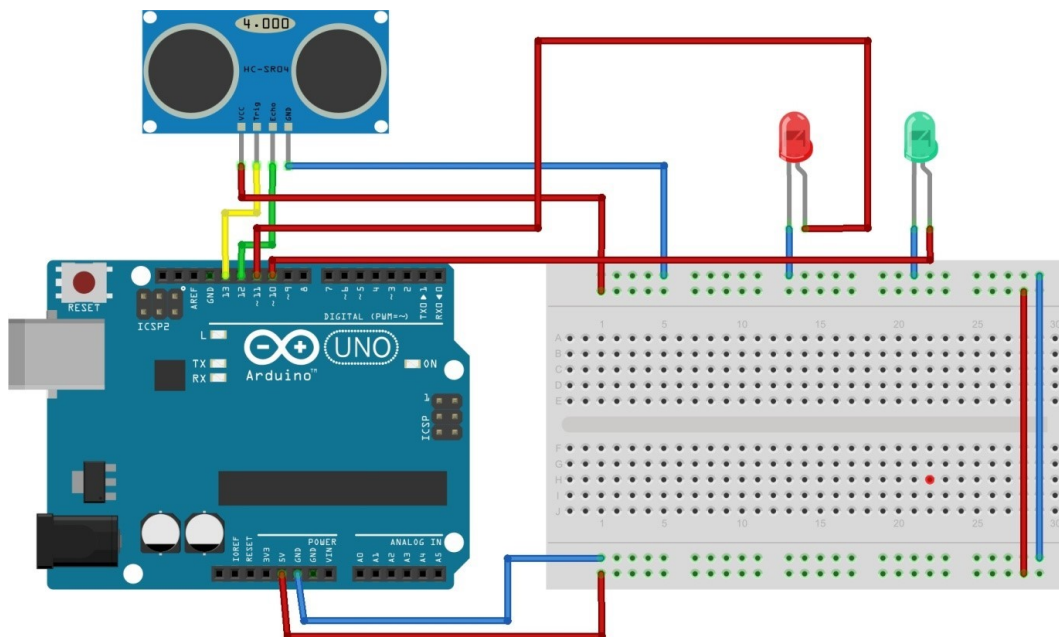
Slika 2.4: Princip rada ultrazvučnog senzora (izvor: „Autorski rad“)

Uzimajući u obzir vrijeme putovanja i brzinu zvuka, jednostavno je izračunati udaljenost. Okidač (eng. *trigger*) koristi se za pokretanje ultrazvučnih impulsa. Postavljanjem ovog priključka na **HIGH** za vrijeme od 10 µs senzor inicira ultrazvučni prasak. Kao odgovor, senzor odašilje ultrazvučni niz od osam impulsa na frekvenciji od 40 kHz. Ovaj uzorak od 8 impulsa posebno je dizajniran tako da prijamnik može razlikovati odašlane impulse od ambijentalnog ultrazvučnog šuma. Kako ovih osam ultrazvučnih impulsa putuje kroz zrak dalje od odašiljača, **Echo** stanje postavi se na **HIGH** za prijem reflektiranog signala i kada se primi, **Echo** signal očitava vrijeme trajanja **t** ovog impulsa u mikrosekundama. Ako se reflektirani signal ne primi nakon 38 ms, **Echo** stanje postavi se na **LOW**. Prema tome, impuls od 38 ms ukazuje da nema prepreka u dometu senzora. Uzimajući u obzir brzinu zvuka u zraku od 340 m/s, udaljenost **d** jednostavno se može izračunati preko jednadžbe:

$$d = t [\mu s] \cdot 0,034 \left[\frac{cm}{\mu s} \right].$$

Jednadžba 21: Izračun udaljenosti objekta

Shema spajanja senzora dana je na slici 2-5, a programski kôd 2-2 predstavlja aplikaciju za mjerenje udaljenosti.



Slika 2.5: Shema spajanja senzora HC-SR04 (izvor: „Autorski rad“)

Programski kôd 22: Aplikacija za mjerenje udaljenosti

```

/* SENZOR UDALJENOSTI HC-SR04
   Pali se crvena LED dioda ako je predmet bliže od 10 centimetara,
   a ako je predmet dalje od 10 centimetara,
   gasi se crvena i pali se zelena LED dioda.
   Ako predmet nije u doseg od 50 centimetara, obje su LED diode
   ugašene,
   na Serial Monitoru ispisuje se "Nema predmeta u doseg od 50 centime-
   tara". */
//definicija priključaka
#define trigPin 13
#define echoPin 12
#define led 11
#define led2 10

void setup() {
  //inicijalizacija komunikacije za prikaz na zaslonu računala
  Serial.begin (9600);
  //definiranje izlaza i ulaza na priključcima
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(led, OUTPUT);

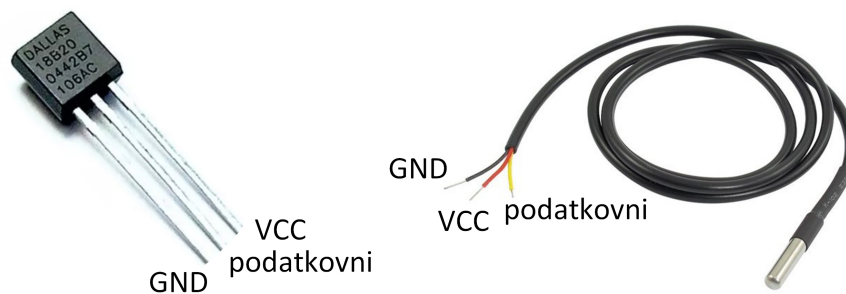
```

```
pinMode(led2, OUTPUT);
}
void loop() {
  // deklaracija lokalnih varijabli udaljenost d i vrijeme t
  long t;
  long d;
  // početno postavljanje Trigera na nisku razinu
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // postavljanje Trigera na visoku razinu za vrijeme od 10 mikrosekundi
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  // postavljanje Trigera na nisku razinu
  digitalWrite(trigPin, LOW);
  // očitavanje vremena Echo-priključka u HIGH stanju u mikrosekundama
  t = pulseIn(echoPin, HIGH);
  //izračunavanje udaljenosti u centimetrima
  d = (t/2) * 0,034;
  // Uvjet za paljenje LED dioda
  if (udaljenost < 10) {
    // Kad se ispune uvjeti, crvena LED se pali, a zelena LED se gasi.
    digitalWrite(led,HIGH);
    digitalWrite(led2,LOW);
  }
  // Pali se zelena, a gasi crvena za udaljenost između 10 i 50 cm.
  else {
    digitalWrite(led,LOW);
    digitalWrite(led2,HIGH);
  }
  // Za udaljenost veću od 50 cm i manju ili jednaku 0 gase se obje LED
  diode.
  if (udaljenost >= 50 || udaljenost <= 0){
    digitalWrite(led,LOW);
    digitalWrite(led2,LOW);
    // ispis na zaslonu računala
    Serial.println("Nema predmeta u doseg od 50 centimetara");
  }
  else {
    // ispis udaljenosti na zaslonu računala u cm
    Serial.print(udaljenost);
    Serial.println(" cm");
  }
  delay(500);
}
```

2.3. MJERENJE TEMPERATURE

2.3.1. Senzor DS18B20

U raznim sustavima često se javlja potreba za mjerenjem temperature. Jedan od načina primjena je komercijalno dostupnog i jeftinog senzora DS18B20. Na slici 2-6 prikazan je senzor DS18B20 u TO-92 kućištu i njegova vodonepropusna izvedba.



Slika 2.6: Senzor DS18B20 u kućištu TO-92 i njegova vodonepropusna izvedba (izvor: „Autorski rad“)

DS18B20 jednožilni je (eng. *one-wire*) temperaturni senzor koji proizvodi tvrtka *Dallas Semiconductor*, koju je preuzeo *Maxim Integrated* (DS18B20 DALLAS, 2023). Budući da je to jednožilni uređaj, potreban mu je samo jedan digitalni priključak za komunikaciju s mikrokontrolerom. Senzor je dostupan u dvama oblicima. Jedan dolazi u pakiranju TO-92, a drugi dolazi u obliku vodootporne sonde koja se uglavnom koristi kada se mjeri temperatura pod vodom, ispod zemlje ili u uvjetima izloženosti vanjskim vremenskim utjecajima. U ovoj vježbi koristi se vodonepropusni model. Senzor temperature DS18B20 precizan je i ne zahtijeva nikakve vanjske komponente za rad. Ima temperaturni raspon rada od $-55\text{ }^{\circ}\text{C}$ do $+125\text{ }^{\circ}\text{C}$ i točnost od $\pm 0,5\text{ }^{\circ}\text{C}$. Razlučivost senzora temperature može se postaviti na 9, 10, 11 ili 12 bita. Zadana razlučivost pri uključivanju senzora jest 12-bitna, uz točnost od $0,0625\text{ }^{\circ}\text{C}$. Senzor radi na naponu od 3 V do 5,5 V i troši samo 1 mA tijekom aktivnog stanja, a u stanju mirovanja oko $1\text{ }\mu\text{A}$, tablica 2-3.

Tablica 23: Karakteristike DS18B20 senzora

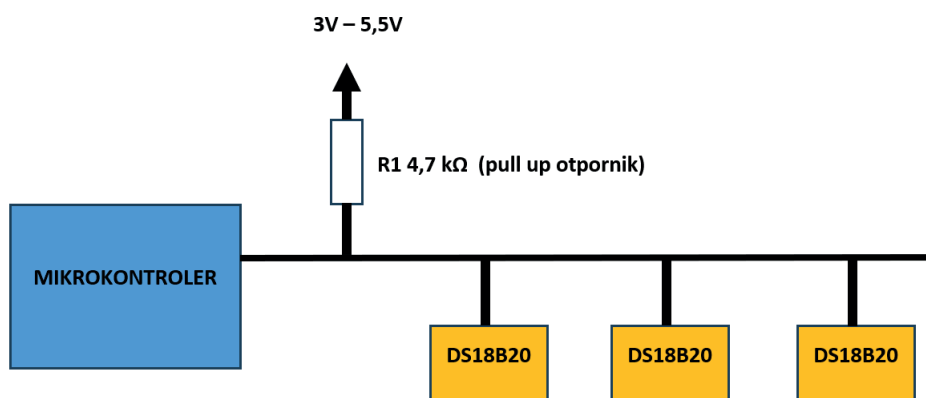
DS18B20 senzor	Tehničke karakteristike
napajanje	3 V – 5,5 V
potrošnja struje	1 mA
temperaturni opseg	$-55\text{ }^{\circ}\text{C}$ – $125\text{ }^{\circ}\text{C}$
točnost	$\pm 0,5\text{ }^{\circ}\text{C}$
rezolucija	9 – 12 bitova (ovisno o odabiru)
vrijeme pretvorbe	< 750 ms

DS18B20 senzor radi na načelu izravnog pretvaranja temperature u digitalnu vrijednost. Njegova je glavna značajka mogućnost mijenjanja broja bitova u skladu s potrebnom rezolucijom. Raspored priključaka na DS18B20 senzoru dan je u tablici 2-4.

Tablica 24: Raspored priključaka DS18B20 senzora

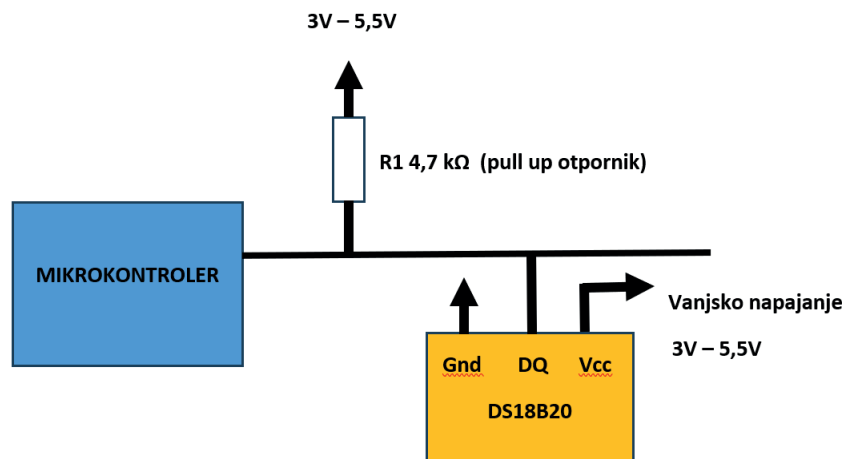
DS18B20 priključak	Značenje
VCC	napajanje senzora u opsegu od 3,3 V do 5 V
podatkovni	podatkovni priključak koji se spaja na digitalni priključak na Arduino
GND	spoj na masu

Jedna je od bitnih značajki jednožilne sabirnice da se više senzora može spojiti na istu sabirnicu. Na istu sabirnicu može se povezati više senzora budući da je svaki od DS18B20 senzora unaprijed programiran s jedinstvenim 64-bitnim serijskim kôdom. Ova značajka može biti izuzetno korisna kada je potrebno izvršiti mjerenje temperature na velikom području, slika 2-7.



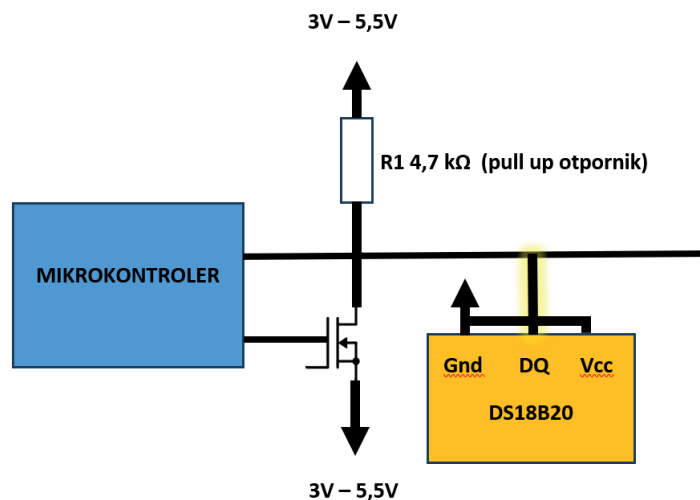
Slika 2.7: Povezivanje više DS18B20 senzora na istu sabirnicu (izvor: „Autorski rad“)

Postoje mogućnosti vanjskog i parazitnog napajanja DS18B20 senzora temperature. Kod metode vanjskog napajanja senzor se napaja iz baterijskog ili naponskog izvora, a ova metoda primjenjiva je za temperature ispod 100 °C. Prednost je ove metode što nema dodatnog opterećenja na podiznom (engl. *Pull-up*) otporniku koji se koristi na sabirnici i osigurava veliku točnost mjerenja, slika 2-8.



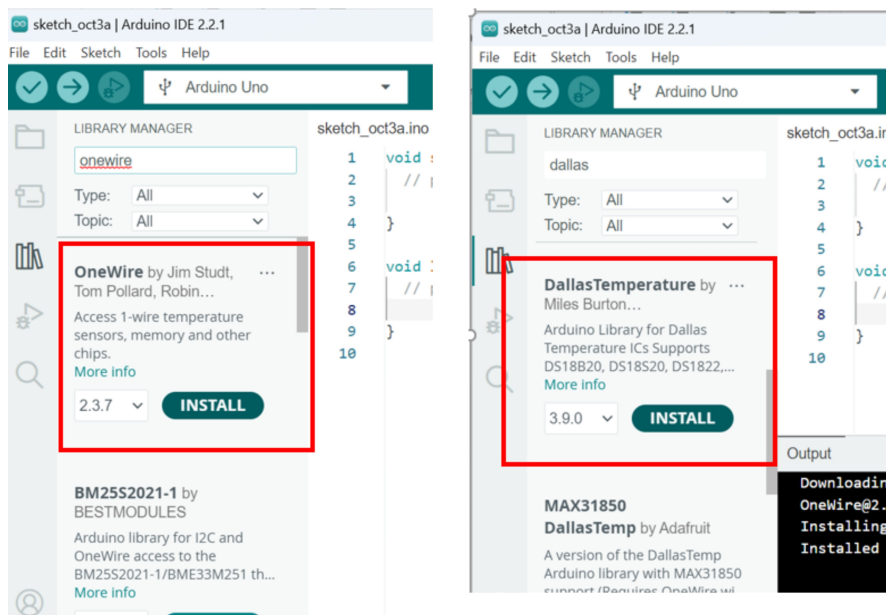
Slika 2.8: Vanjsko napajanje DS18B20 senzora (izvor: „Autorski rad“)

Kod metode parazitnog napajanja ne trebamo posebno napajanje i koristi se za temperature veće od 100 °C. Pored podiznog otpornika u ovoj metodi potrebno je koristiti MOSFET tranzistor, slika 2-9.



Slika 2.9: Parazitno napajanje DS18B20 senzora (izvor: „Autorski rad“)

Schema povezivanja komponenti za mjerenje temperature pomoću senzora DS18B20 dana je na slici 2-10.



Slika 2.12: Odabir OneWire.h i DallasTemperature.h biblioteka (izvor: „Autorski rad“)

Biblioteka *OneWire.h* služi za rad s jednožilnom sabirnicom, dok biblioteka *DallasTemperature.h* služi za rad sa senzorom DS18B20. U programskom kôdu 2-3 dana je cjelovita aplikacija za čitanje podataka sa senzora i njihov ispis na zaslonu računala.

Programski kôd 23: Aplikacija za mjerenje temperature pomoću senzora DS18B20

```

/*****
Mjerenje temperature pomoću senzora DS18B20
*****/
//Učitavanje biblioteka
#include <OneWire.h>
#include <DallasTemperature.h>
// Podatkovni priključak na koji je spojen senzor DS18B29
const int oneWireBus = 2;
// Postavljanje OneWire reference za omogućavanje jednožilne komunikacije.
OneWire oneWire(oneWireBus);
// Prosljeđivanje adrese senzora DallasTemperature biblioteci
DallasTemperature sensors(&oneWire);

void setup() {
  // Pokretanje serijske komunikacije za Serial Monitor
  Serial.begin(9600);
  // Pokretanje DS18B20 senzora

```

```
sensors.begin();
}
void loop() {
  //Zahtjev za očitanjem temperature sa senzora
  sensors.requestTemperatures();
  // deklaracija varijable kojoj pridružujemo vrijednost temperature
  float temperaturaC = sensors.getTempCByIndex(0);
  //Ispisivanje vrijednosti varijable temperaturaC na zaslonu računala
  Serial.print(temperaturaC);
  //Ispisivanje znaka za Celzije
  Serial.println(" °C");
  // Kašnjenje 5 sekundi
  delay(5000);
}
```

Unutar funkcije *loop()* poziva se funkcija *requestTemperatures()* koja upućuje zahtjev svim sensorima na sabirnici da izvrše mjerenje temperature. Zatim se poziva funkcija *getTempCByIndex(deviceIndex)*, gdje je *deviceIndex* adresa senzora na sabirnici koja je u razmatranom programu 0 jer postoji samo jedan senzor na sabirnici. Ova funkcija očitava temperaturu sa senzora. U biblioteci *DallasTemperature.h* postoje i druge funkcije za rad sa senzorom, kao što su:

- *setResolution()* – postavlja razlučivost internog analogno-digitalnog pretvarača (ADC) na 9, 10, 11 ili 12 bita, što odgovara točnošću očitavanja od 0,5 °C, 0,25 °C, 0,125 °C i 0,0625 °C
- *bool getWaitForConversion()* – vraća *boolean* vrijednost za utvrđivanje je li mjerenje temperature dovršeno ili nije
- *setHighAlarmTemp()* i *setLowAlarmTemp()* – konfiguriraju unutarnje alarme za visoku i nisku temperaturu uređaja u stupnjevima Celzija. Prihvatljivi temperaturni raspon je od -55 do 125 °C
- *bool hasAlarm()* – funkcija vraća **TRUE** kada temperatura prijeđe bilo visoku ili nisku postavljenu vrijednost alarma.

2.3.2. NTC termistor

Termistori su promjenjivi otpornici koji mijenjaju svoj otpor s promjenom temperature. Oni se dijele prema načinu na koji električni otpor reagira na promjene temperature. Kod termistora s negativnim temperaturnim koeficijentom (NTC) električni otpor smanjuje se s porastom temperature. Temperaturno područje rada NTC termistora jest od -50 °C do 150 °C. Kod termistora s pozitivnim temperaturnim koeficijentom (PTC) električni otpor raste s porastom temperature. Temperaturno područje rada PTC termistora jest od -50 °C do 220 °C. U ovom primjeru koristit će se NTC termistor od 100 kΩ na referentnoj temperaturi od

25 °C. Termistor se povezuje s Arduino pločicom prema shemi prikazanoj na slici 2-13. Za čitanje podataka s termistora koristi se analogni priključak A0 na Arduino pločici. Arduino čita napon na analognom ulazu s naponskog djelitelja prema jednadžbi 2-2:

$$U_{ul} = U \frac{R_1}{R_1 + R_2},$$

Jednadžba 22: Ulazni napon

gdje je:

U_{ul} - napon na analognom ulazu A0

U - napon napajanja $U = 5\text{ V}$

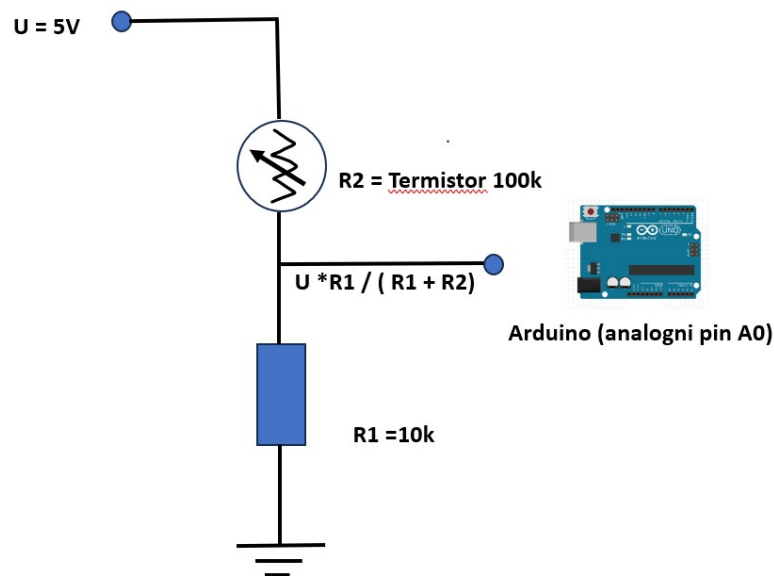
R_1 - pomoćni otpornik od 10 kW

R_2 - otpor termistora na temperaturi T.

Otpor NTC termistora na danoj temperaturi T sada se lako izračuna primjenom jednadžbe:

$$R_2 = R_1 \left(\frac{U}{U_{ul}} - 1 \right).$$

Jednadžba 23: Računanje otpora termistora



Slika 2.13: Povezivanje Arduina i NTC termistora (izvor: „Autorski rad“)

Nakon što se izračuna otpor termistora na danoj temperaturi, potrebno ga je pretvoriti u temperaturu. Za ovu pretvorbu koristi se Steinhart-Hartova jednađba (*Steinhart-Hart equation*, 2023) prema kojoj se izračunava otpor poluvodiča pri različitim temperaturama:

$$\frac{1}{T} = A + B \cdot \ln R + C(\ln R)^3,$$

Jednađba 24: Steinhart-Hartova jednađba

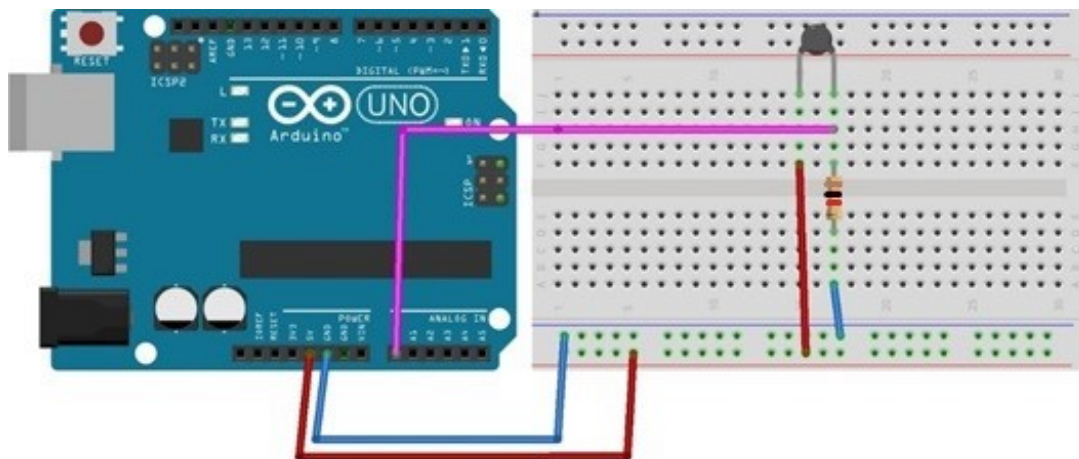
gdje je:

T – temperatura u kelvinima

R – otpor na temperaturi **T** u ohmima

A, B, C – Steinhart-Hartovi koeficijenti koji ovise o vrsti i modelu termistora i temperaturnom području.

Shema povezivanja za Arduino aplikaciju dana je na slici 2-14.



Slika 2.14: Cjelovita shema povezivanja Arduina i 100 k NTC termistora (izvor: „Autorski rad“)

U programskom kôdu 2-4 dana je aplikacija za mjerenje temperature pomoću termistora, a na slici 2-15 ispis mjerenja na zaslon računala.

Programski kôd 24: Aplikacija za mjerenje temperature pomoću termistora

```

/*
  Termistor 100 k
  */
int ThermistorPin = 0;
int Vo;
float R1 = 10000;
float logR2, R2, T, Tc, Tf;
float c1 = 1.009249522e-03, c2 = 2.378405444e-04, c3 = 2.019202697e-07;

void setup() {
  Serial.begin(9600);
}
void loop() {

  Vo = analogRead(ThermistorPin);
  R2 = R1 * (1023.0 / (float)Vo - 1.0);
  logR2 = log(R2);
  T = (1.0 / (c1 + c2*logR2 + c3*logR2*logR2*logR2));
  Tc = T - 273.15;
  Tf = (Tc * 9.0) / 5.0 + 32.0;

  Serial.print("Temperatura: ");
  Serial.print(Tf);
  Serial.print(" F; ");
  Serial.print(Tc);
  Serial.println(" °C");

  delay(500);
}

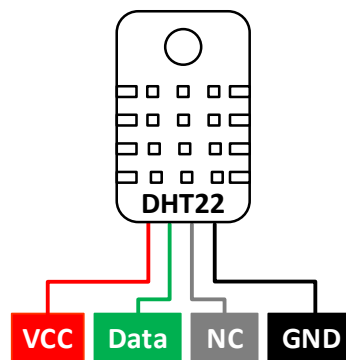
```

The screenshot shows a serial terminal window titled 'COM3'. The window contains a list of temperature readings. Each line displays the temperature in Fahrenheit followed by the temperature in Celsius, separated by a semicolon. The readings are: 80.15 F; 26.75 °C (repeated 5 times), 90.82 F; 32.68 °C (repeated 5 times).

Slika 2.15: Ispis očitavanja NTC termistora (izvor: „Autorski rad“)

2.3.3. Mjerenje temperature i vlažnosti pomoću DHT22 senzora

Za mjerenje temperature i vlažnosti koristit će se *DHT22* digitalni senzor prikazan na slici, slika 2-16 (*Aosong Electronics Co., Ltd, DHT22, 2023*) Ovaj senzor može mjeriti temperaturu u rasponu od $-40\text{ }^{\circ}\text{C}$ do $125\text{ }^{\circ}\text{C}$, s točnošću od $\pm 0,5\text{ }^{\circ}\text{C}$, a rezolucija mjerenja iznosi $\pm 0,1\text{ }^{\circ}\text{C}$. Relativnu vlagu mjeri u rasponu od 0 % do 100 %, s točnošću od $\pm 0,2\text{ }%$, a rezolucija mjerenja vlage iznosi 0,1 %. Napon napajanja kreće se u rasponu od 3,3 V do 6 V, a tipično se koristi napajanje od 5 V. *DHT22* ima ugrađene mehanizme otpornosti na smetnje kako bi osigurao pouzdane i precizne podatke čak i u okruženjima s elektromagnetskim smetnjama. Podaci sa senzora šalju se putem jednosmjerne serijske komunikacije, a kašnjenje u očitavanju podataka iznosi oko 2 sekunde, koliko je potrebno da se izmjere temperatura i vlaga te pošalju na Arduino pločicu.

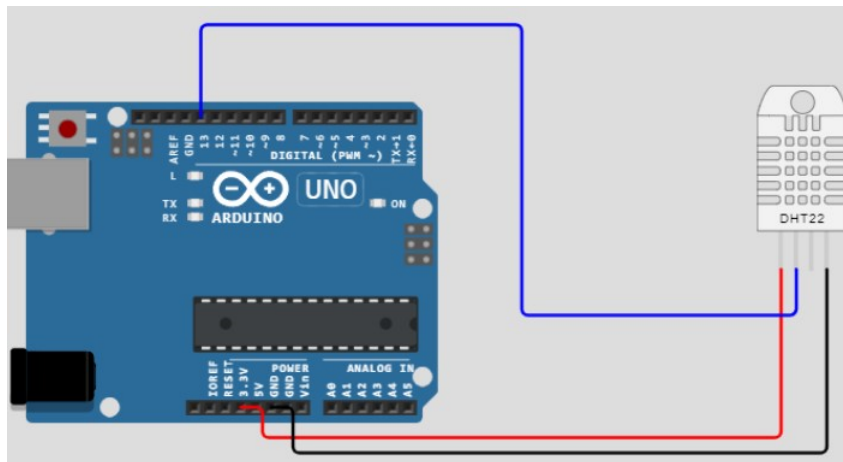


Slika 2.16: Senzor DHT22 (izvor: „Autorski rad“)

Slika 2-16 prikazuje raspored i značenje izvoda senzora DHT22:

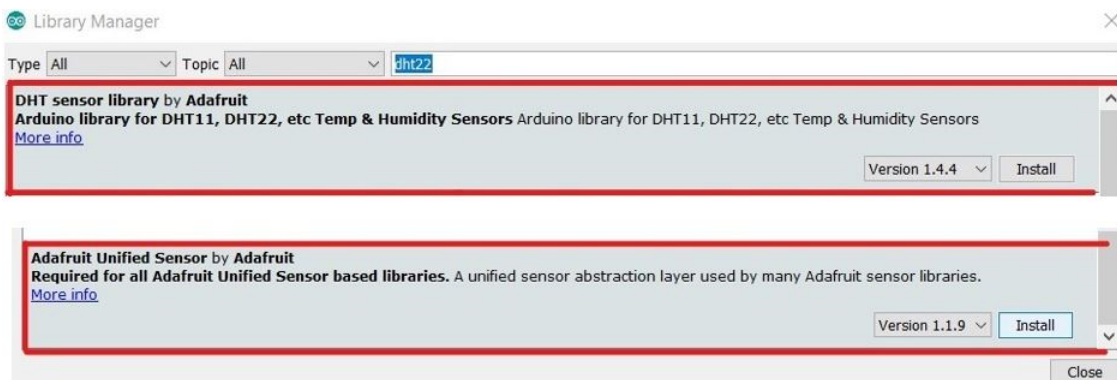
- **VCC** – izvod preko kojeg se senzor napaja (od 3,3 V do 6 V)
- **Data** – komunikacijski izvod koji daje digitalni izlaz, serijski šalje izmjerene podatke o temperaturi i vlazi
- **NC** – izvod koji se ne koristi
- **GND** – izvod za masu/uzemljenje.

Senzor se povezuje s Arduino pločicom prema shemi na slici 2-17. Izvod senzora za slanje podataka spojen je na digitalni izvod 13 na Arduino pločici.



Slika 2.17: Povezivanje Arduina i senzora DHT22 (izvor: „Autorski rad“)

Za rad sa senzorom DHT22 potrebno je instalirati biblioteku za njegovu podršku. U ovom primjeru koristi se *Adafruit Unified Sensor library*. Biblioteka se dodaje na sljedeći način: u Arduino sučelju odabere se **Sketch > Include Library > Manage Libraries**. Za podršku korištenog senzora DHT22 potrebno je instalirati sljedeće dvije biblioteke prikazane na slici 2-18.

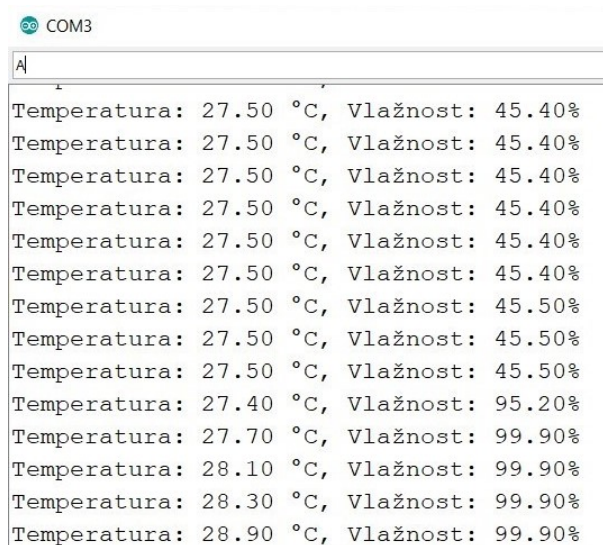


Slika 2.18: Potrebne biblioteke za rad sa senzorom DHT22 (izvor: „Autorski rad“)

Programski kôd 2-1 daje primjer korištenja senzora DHT22 za mjerenje temperature i vlažnosti te prikaz podataka na zaslonu računala. Slika 2-19 prikazuje ispis podataka na zaslon računala.

Programski kôd 25: Mjerenje pomoću DHT22 senzora

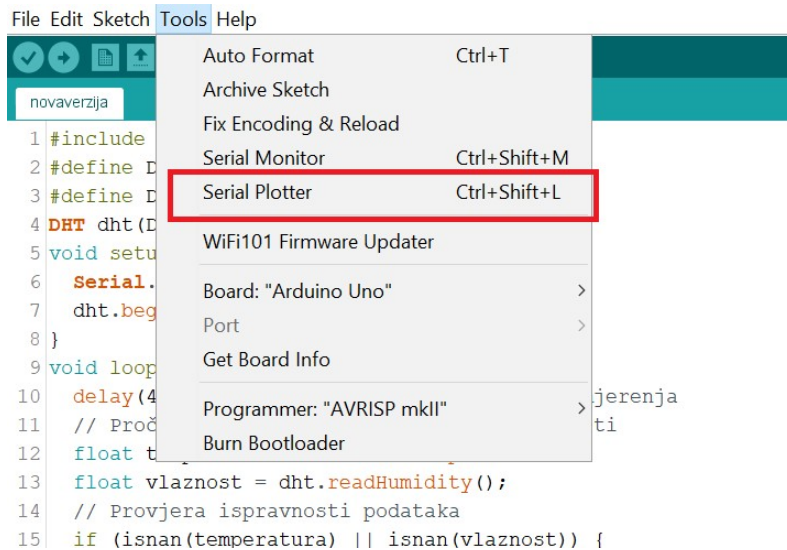
```
#include "DHT.h"
#define DHT_PIN 13
#define DHT_TYPE DHT22
DHT dht(DHT_PIN, DHT_TYPE);
void setup() {
  Serial.begin(9600);
  dht.begin();
}
void loop() {
  delay(4000); // čekanje 4 sekunde između mjerenja
  // Pročitaj podatke o temperaturi i vlažnosti
  float temperatura = dht.readTemperature();
  float vlaznost = dht.readHumidity();
  // Provjera ispravnosti podataka
  if (isnan(temperatura) || isnan(vlaznost)) {
    Serial.println("Nemoguće dobiti podatke sa senzora!");
  } else {
    // Ispisivanje prikupljenih podataka na Serial monitoru
    Serial.print("Temperatura: ");
    Serial.print(temperatura);
    Serial.print(" °C, Vlažnost: ");
    Serial.print(vlaznost);
    Serial.println(" %");
  }
}
```



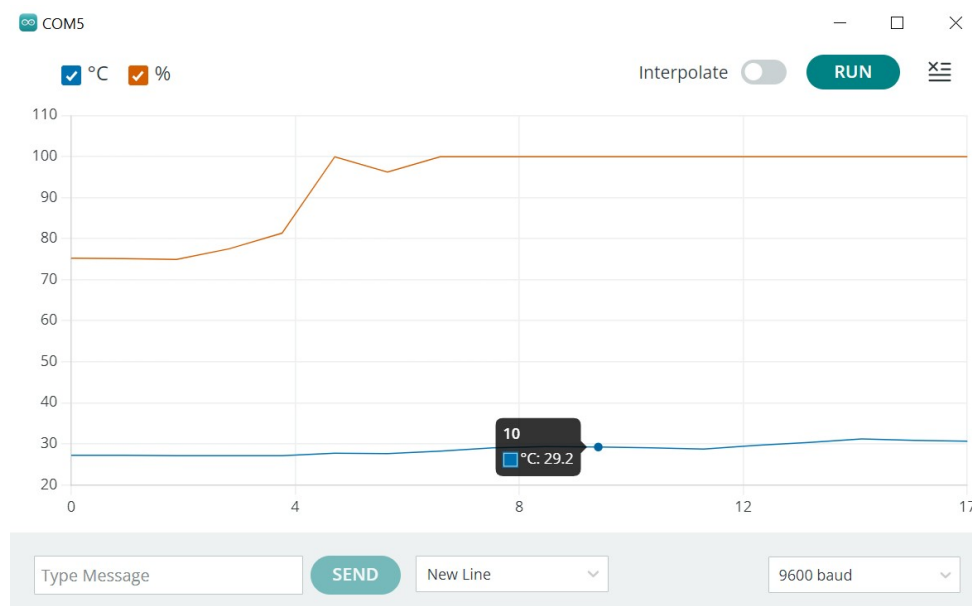
```
COM3
A
Temperatura: 27.50 °C, Vlažnost: 45.40%
Temperatura: 27.50 °C, Vlažnost: 45.40%
Temperatura: 27.50 °C, Vlažnost: 45.40%
Temperatura: 27.50 °C, Vlažnost: 45.40%
Temperatura: 27.50 °C, Vlažnost: 45.40%
Temperatura: 27.50 °C, Vlažnost: 45.40%
Temperatura: 27.50 °C, Vlažnost: 45.50%
Temperatura: 27.50 °C, Vlažnost: 45.50%
Temperatura: 27.50 °C, Vlažnost: 45.50%
Temperatura: 27.40 °C, Vlažnost: 95.20%
Temperatura: 27.70 °C, Vlažnost: 99.90%
Temperatura: 28.10 °C, Vlažnost: 99.90%
Temperatura: 28.30 °C, Vlažnost: 99.90%
Temperatura: 28.90 °C, Vlažnost: 99.90%
```

Slika 2.19: Ispis podataka sa senzora DH22

U Arduino IDE sučelju postoji mogućnost grafičkog prikaza podataka pomoću funkcije **Serial Plotter**. Prvo se odabere **Tools** u izborniku, a zatim **Serial Plotter** ili kombinacijom tipki **Ctrl+Shift+L**, slika 2-20, a grafički prikaz podataka na slici 2-21.



Slika 2.20: Odabir funkcije za grafički prikaz podataka (izvor: „Autorski rad“)



Slika 2.21: Grafički prikaz podataka sa senzora DHT22 (izvor: „Autorski rad“)

2.3.4. Senzor BME280 I2C

BME280 senzor je tlaka, temperature i vlažnosti zraka (Bosch, Humidity sensor BME280, 2023.) te je popularan u mnogim primjenama uključujući meteorološke stanice, nadzor okoliša, GPS uređaje i druge, slika 2-22.

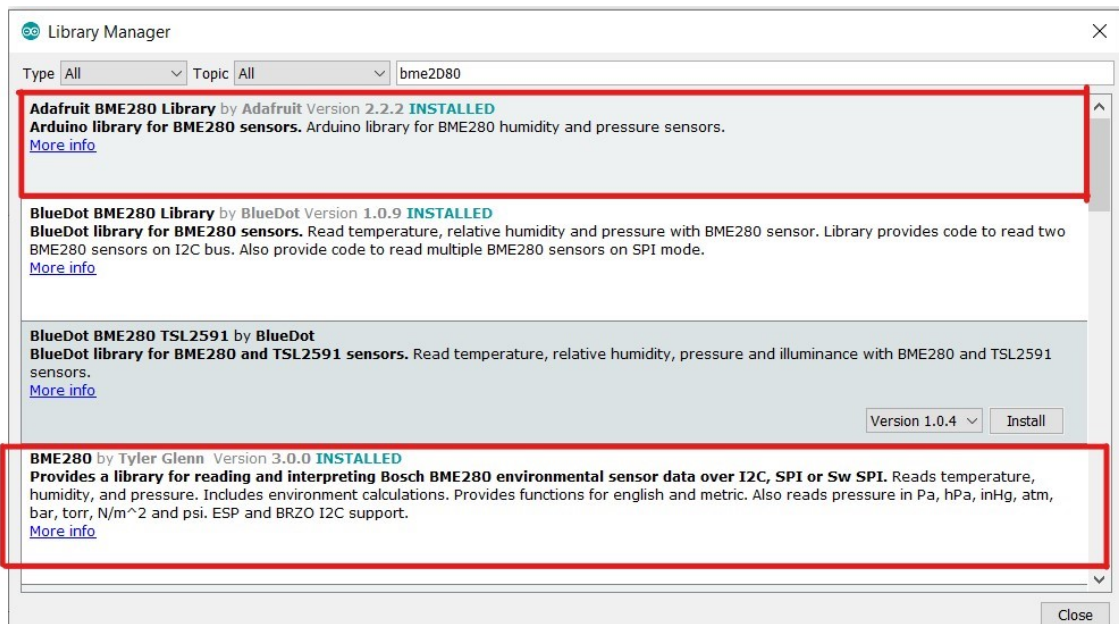


Slika 2.22: Senzor BME280 (izvor: „Autorski rad“)

Značajke BME280 senzora su:

- **Mjerenje tlaka:** senzor mjeri atmosferski tlak. Preciznim mjerenjem tlaka omogućuje se proračun nadmorske visine i detekcija promjena u vremenskim uvjetima. Raspon mjerenja tlaka obično je od 300 hPa do 1100 hPa.
- **Mjerenje temperature:** senzor također mjeri temperaturu okoline. Raspon mjerenja temperature jest od $-40\text{ }^{\circ}\text{C}$ do $+85\text{ }^{\circ}\text{C}$.
- **Mjerenje vlažnosti zraka:** mjeri relativnu vlažnost zraka u rasponu od 0 % do 100 %. Ova funkcija korisna je u aplikacijama gdje je praćenje vlažnosti važno, kao što su unutarnji klimatski sustavi, vremenske stanice i kontrola okoliša.
- **Visoka preciznost:** senzor vrlo precizno mjeri vrijednosti tlaka, temperature i vlage.
- **Velika brzina očitavanja vrijednosti:** senzor može brzo izmjeriti tlak i temperaturu, što ga čini pogodnim za aplikacije koje zahtijevaju veliku brzinu uzorkovanja.
- **Kalibracija:** senzor obično dolazi s unaprijed kalibriranim parametrima. To znači da nije potrebno dodatno kalibrirati senzor nakon što ga spojimo.
- **Niska potrošnja energije:** senzor ima nisku potrošnju energije pa se može baterijski napajati.
- **Digitalno sučelje:** senzor komunicira pomoću serijske komunikacije I2C (eng. *Inter-Integrated Circuit*) i SPI (eng. *Serial Peripheral Interface*) što ga čini kompatibilnim s raznim mikrokontrolerima i mikroprocesorima.

Prvo je potrebno instalirati BME280 biblioteku kako bi se senzor mogao koristiti u Arduino IDE okruženju. To se radi na sljedeći način: odabere se **Sketch > Include Library > Manage Libraries**. Nakon toga se u tražilicu upiše BME280 i instaliraju se sljedeće dvije biblioteke prikazane na slici 2-23.



Slika 2.23: Biblioteke za podršku BME280 (izvor: „Autorski rad“)

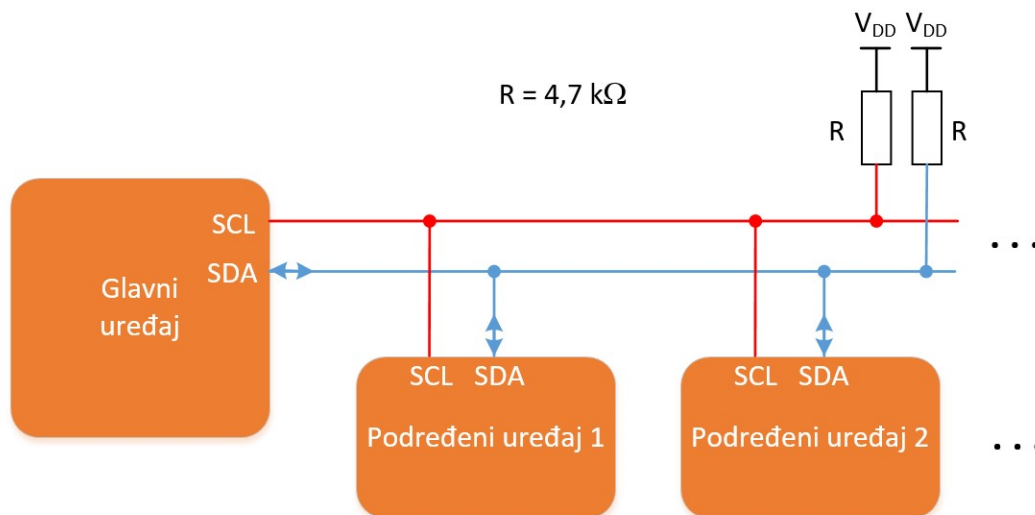
Jedan od načina povezivanja senzora BME280 s Arduino pločicom primjena je I2C komunikacije. I2C serijski je komunikacijski protokol koji omogućava razmjenu podataka među različitim elektroničkim komponentama, kao što su mikrokontroleri, senzori, memorije i drugi uređaji. I2C razvio je *Philips* (sada *NXP Semiconductors*) i postao je široko korišten u elektronici zbog svoje jednostavnosti i pouzdanosti (UM10204, *I2C-bus specification and user manual*, 2021.). Karakteristike I2C protokola su:

- **Dvije signalne linije za komunikaciju:** SDA (eng. *Serial Data Line*) i SCL (eng. *Serial Clock Line*). SCL linija generira taktove koji sinkroniziraju komunikaciju među uređajima, a SDA linija služi za prijenos podataka.
- **Glavni – podređeni arhitektura:** u I2C komunikaciji jedan je uređaj glavni (eng. *master*), koji kontrolira komunikaciju, dok su ostali uređaji podređeni (eng. *slave*), koji odgovaraju na naredbe glavnog uređaja.
- **Adresiranje uređaja:** svaki uređaj na I2C mreži ima jedinstvenu adresu koja je identifikacija tog uređaja. Glavni uređaj koristi adresu kako bi komunicirao s određenim podređenim uređajem.
- **Prijenos podataka:** podaci se prenose serijski bit po bit, a niz od osam bitova čini jedan bajt.

I2C komunikacija koristi dva podizna (eng. *pull-up*) otpornika kako bi osigurala pouzdanu i stabilnu komunikaciju među uređajima na sabirnici, slika 2-24. Podizni otpornici spojeni su između SDA linije i napajanja V_{DD} te između SCL linije i napajanja V_{DD} . Zbog ovoga je na liniji stanje logičke jedinice kada uređaji ne komuniciraju. Svaki uređaj može postaviti svoju liniju na nisku razinu (logička nula), ali je ne može aktivno podići na visoku razinu (logička

jedinica). Vrijednosti podiznih otpornika kreću se u rasponu od 2.2 k Ω do 10 k Ω i služe za pasivno podizanje linija prema napajanju. Kada uređaji počnu komunicirati, svaki uređaj može povući SDA liniju na nisku razinu kako bi prenio logičku nulu, a kada se linija oslobodi, podizni otpornici ponovno podižu napon na visoku razinu, čime je omogućena sigurna i pouzdana komunikacija.

U slučaju kada je mnogo uređaja spojeno na sabirnicu, kapacitivno opterećenje može biti značajno, stoga podizni otpornici omogućavaju održavanje brzine i kvalitete komunikacije bez obzira na kapacitivno opterećenje. I2C protokol najčešće se koristi u aplikacijama gdje je potrebna komunikacija između više uređaja unutar istog sustava, kao što su senzori temperature, vlage, svjetla, EEPROM memorije i drugi. U tablici 2-5 dan je raspored povezivanja izvoda između BME280 senzora i Arduino pločice.



Slika 2.24: I2C sabirnica (izvor: „Autorski rad“)

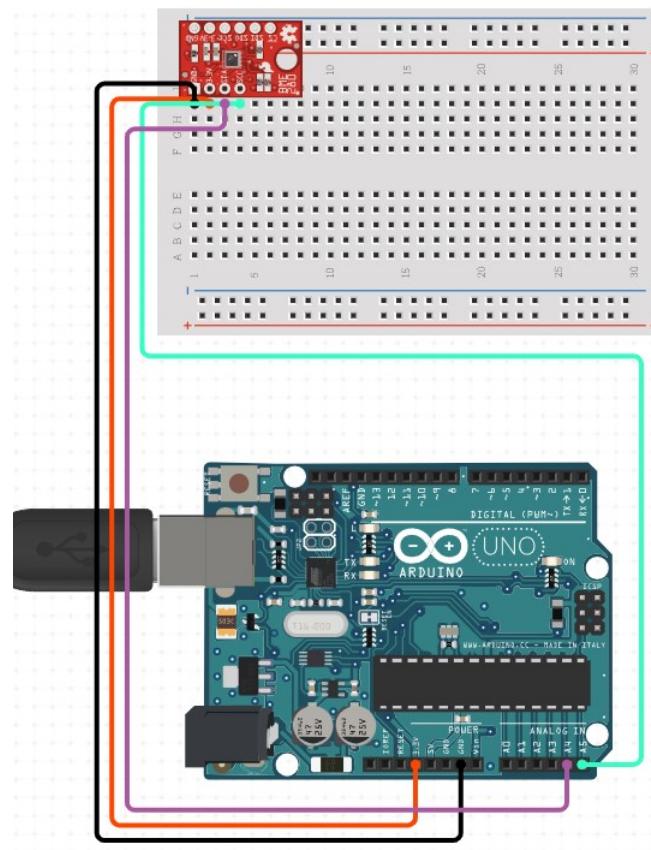
Tablica 25: Povezivanje senzora BME280 i ESP32 pločice putem I2C komunikacije

BME280	Arduino
VCC	3,3 V
GND	GND
SCL	A5
SDA	A4

Primjer mjerenja temperature, vlažnosti i tlaka pomoću BME280 senzora i I2C komunikacije dan je u programskom kôdu 2-6 s detaljnim komentarima, a na slici 2-25 dana je shema povezivanja senzora s Arduino pločicom.

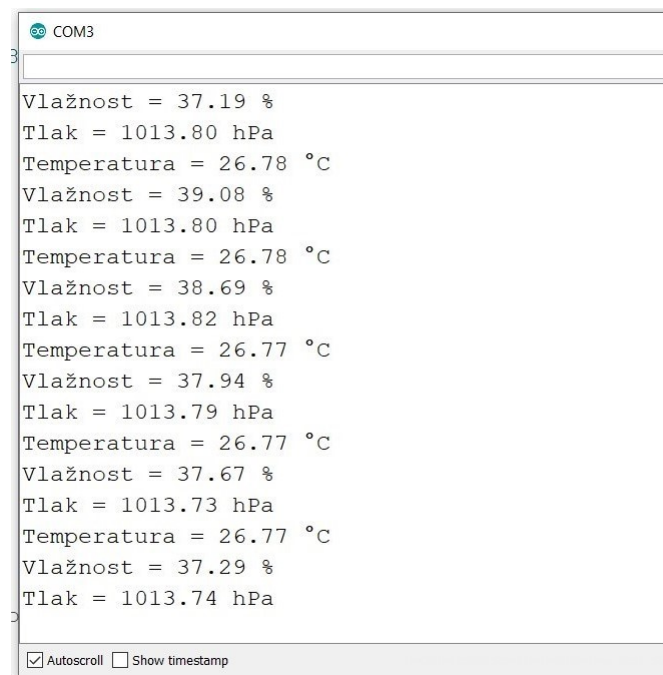
Programski kôd 26: Primjena BME280 senzora i I2C komunikacije

```
#include <Wire.h> //podrška za I2C komunikaciju
#include <Adafruit_Sensor.h> //podrška za rad sa sensorima
#include <Adafruit_BME280.h> //podrška za BME280 senzor
Adafruit_BME280 bme; //kreiranje objekta za BME280 senzor
void setup() {
  Serial.begin(9600);
  Wire.begin(); // inicijalizacija I2C komunikacije
  if (!bme.begin(0x76)) { //inicijalizacija i adresiranje BME280 senzora
    Serial.println("Nije moguće pronaći BME280 senzor, provjerite veze ili
    adresu.");
    while (1);
  }
}
void loop() {
  //čitanje i prikaz temperature
  Serial.print(F("Temperatura = "));
  Serial.print(bme.readTemperature());
  Serial.println(" °C");
  //čitanje i prikaz vlažnosti
  Serial.print(F("Vlažnost = "));
  Serial.print(bme.readHumidity());
  Serial.println(" %");
  //čitanje i prikaz tlaka
  Serial.print(F("Tlak = "));
  Serial.print(bme.readPressure() / 100.0F); // Pritisak u hektopaskalima
  (hPa)
  Serial.println(" hPa");
  // Pauza od 5 sekundi među očitavanjima
  delay(5000);
}
```



Slika 2.25: Povezivanje Arduina i BME280 putem I2C (izvor: „Autorski rad“)

Ovaj kôd koristi Adafruit BME280 biblioteku koja olakšava komunikaciju s BME280 senzorom preko I2C sabirnice. Prvo se inicijalizira serijska komunikacija i I2C sabirnica. Zatim se inicijalizira BME280 senzor, a ako senzor nije pronađen, ispisuje se poruka korisniku. U funkciji *loop()* program čita temperaturu, vlažnost i tlak sa senzora i ispisuje ih na zaslonu računala svakih pet sekundi. Vrijednosti su ispisane u stupnjevima Celzija za temperaturu, postocima za vlažnost i hektopaskalima (hPa) za tlak, slika 2-26.



```

COM3
Vlažnost = 37.19 %
Tlak = 1013.80 hPa
Temperatura = 26.78 °C
Vlažnost = 39.08 %
Tlak = 1013.80 hPa
Temperatura = 26.78 °C
Vlažnost = 38.69 %
Tlak = 1013.82 hPa
Temperatura = 26.77 °C
Vlažnost = 37.94 %
Tlak = 1013.79 hPa
Temperatura = 26.77 °C
Vlažnost = 37.67 %
Tlak = 1013.73 hPa
Temperatura = 26.77 °C
Vlažnost = 37.29 %
Tlak = 1013.74 hPa
Autoscroll Show timestamp

```

Slika 2.26: Ispis podataka BME280 I2C

2.3.5. Senzor BME280 SPI

Drugi način povezivanja senzora BME280 i Arduino pločice jest putem SPI komunikacije. SPI komunikacijski je protokol koji se koristi za komunikaciju među različitim elektroničkim komponentama u uređajima i sustavima (M. Grusin, *Serial Peripheral Interface (SPI)*, 2023.). Ovo je sinkroni serijski protokol koji koristi četiri osnovne signalne linije za komunikaciju među uređajima.

Arhitektura SPI komunikacije:

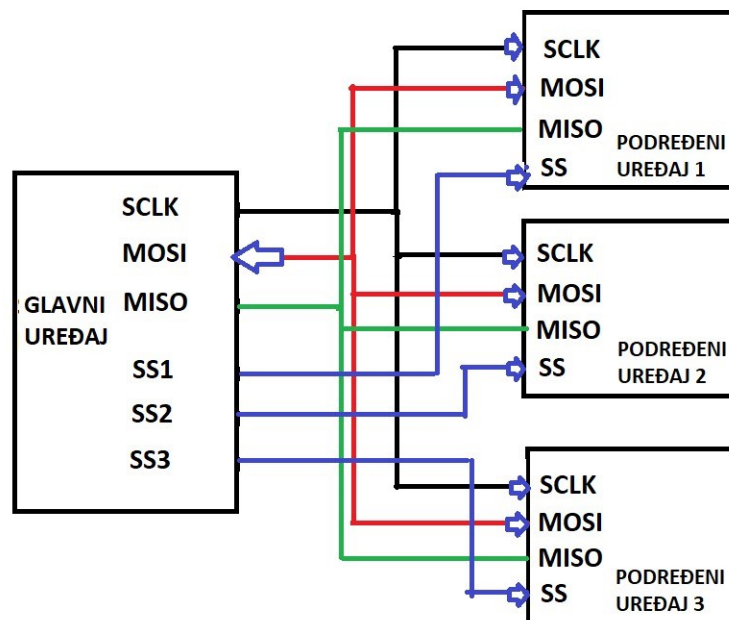
- **SCLK (eng. Serial Clock):** ova linija generira taktove i sinkronizira komunikaciju među uređajima.
- **MISO (eng. Master In Slave Out):** ova linija služi za prijenos podataka od podređenog prema glavnom uređaju.
- **MOSI (eng. Master Out Slave In):** ova linija prenosi podatke od glavnog uređaja prema podređenim uređajima.
- **SS/CS (eng. Slave Select/Chip Select):** ova se linija koristi za odabir podređenog uređaja s kojim se želi komunicirati.

Osnovne značajke SPI protokola su:

- **Potpuna dvosmjerna (eng. full-duplex) komunikacija:** SPI omogućava potpunu dvosmjernu komunikaciju, a to znači da glavni i podređeni uređaji mogu istovremeno primiti i slati podatke.

- **Glavni-podređeni konfiguracija:** u SPI komunikaciji postoji samo jedan glavni uređaj koji kontrolira komunikaciju, dok može biti više podređenih uređaja koji odgovaraju na naredbe.
- **Nema adresiranja uređaja:** u SPI protokolu nema jedinstvenih adresa za uređaje, kao što je to slučaj kod I2C komunikacije. Umjesto toga koristi se fizička linija SS/CS za odabir određenog podređenog uređaja za komunikaciju.
- **Brza komunikacija:** podaci se mogu prenositi brzinama do 60 Mb/s među integriranim krugovima na elektroničkim pločicama.

SPI često se koristi u aplikacijama gdje je potrebna brza i pouzdana serijska komunikacija, kao što je komunikacija sa sensorima, memorijskim uređajima i u mnogim drugim elektroničkim uređajima. Na slici 2-27 prikazan je način povezivanja uređaja putem SPI komunikacije.

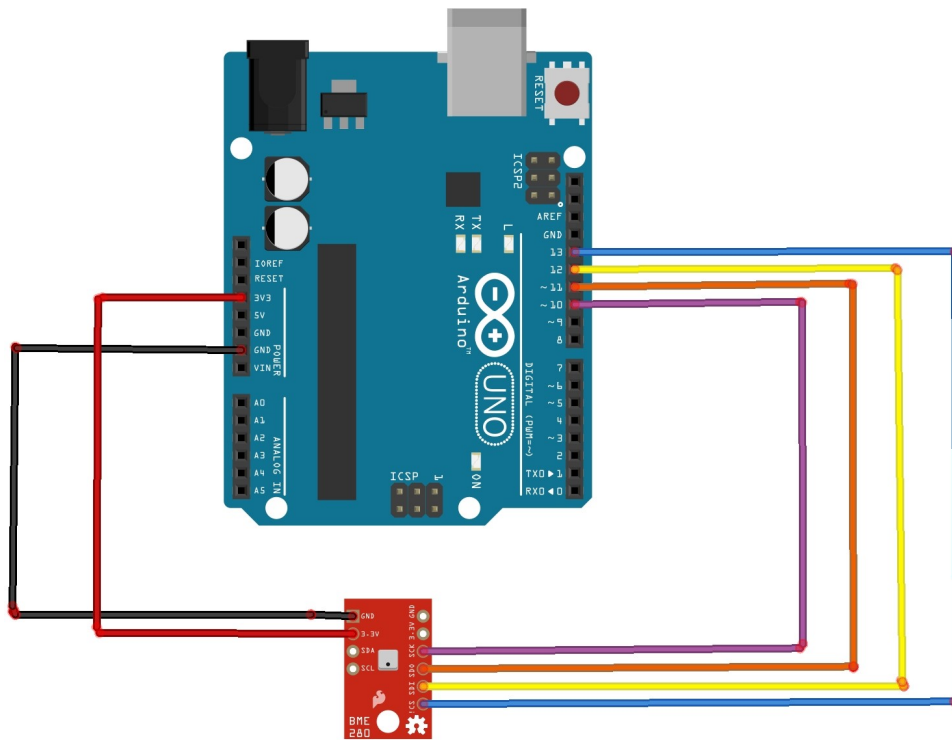


Slika 2.27: Povezivanje uređaja putem SPI komunikacije (izvor: „Autorski rad“)

Kako bi BME280 senzor spojili preko SPI komunikacije s Arduino pločicom, potrebno je povezati pinove na način prikazan u tablici 2-7, a shema povezivanja prikazana je na slici 2-28.

BME280	Arduino
3,3 V	3,3 V
GND	GND
SCK (SPI Clock)	D 10
SDO (MISO)	D 11
SDI (MOSI)	D 12
CS (Chip Select)	D 13

Slika 2.28: Povezivanje senzora BME280 i Arduina putem SPI komunikacije



Slika 2.29: Shema povezivanja senzora BME280 i Arduino pločice putem SPI komunikacije (izvor: „Autorski rad“)


Primjena BME280 senzora i SPI komunikacije dana je u programskom kôdu 2-3.

Programski kôd 27: Programski kôd 2 7: Primjena BME280 senzora i SPI komunikacije

```
#include <SPI.h> //podrška za SPI komunikaciju
#include <Adafruit_Sensor.h> //podrška za rad sa senzorima
#include <Adafruit_BME280.h> //podrška za BME280 senzor
// definicija izvoda
#define BME_SCK 10 //SPI Clock
#define BME_MISO 11 //SPI MISO (Master In Slave Out)
#define BME_MOSI 12 //SPI MOSI (Master Out Slave In)
#define BME_CS 13 //SPI Chip Select
//definicija objekta za rad sa senzorom
Adafruit_BME280 bme(BME_CS, BME_MOSI, BME_MISO, BME_SCK);
void setup() {
  Serial.begin(9600);
}
void loop() {
  //čitanje i prikaz temperature
  Serial.print(F("Temperatura = "));
  Serial.print(bme.readTemperature());
  Serial.println(" °C");
}
```

```
//čitanje i prikaz temperature
Serial.print(F("Vlažnost = "));
Serial.print(bme.readHumidity());
Serial.println(" %");
//čitanje i prikaz temperature
Serial.print(F("Tlak = "));
Serial.print(bme.readPressure() / 100.0F); // Tlak u hektopaskalima
(hPa)
Serial.println(" hPa");
// Pauza od 5 sekundi među očitavanjima
delay(5000); // Pauza od 5 sekundi među očitavanjima
}
```

Ispis podataka s BME280 senzora koji koristi SPI komunikaciju za povezivanje s Arduino pločicom dan je na slici 2-30.



The screenshot shows a serial monitor window titled 'COM3' with a 'Send' button. The output text is as follows:

```
Temperatura: 23.91 °C
Vlažnost: 30.60 %
Tlak: 1019.44 hPa
Temperatura: 23.91 °C
Vlažnost: 30.57 %
Tlak: 1019.45 hPa
Temperatura: 23.93 °C
Vlažnost: 30.49 %
Tlak: 1019.43 hPa
Temperatura: 23.94 °C
Vlažnost: 30.47 %
Tlak: 1019.47 hPa
Temperatura: 23.95 °C
Vlažnost: 30.46 %
Tlak: 1019.43 hPa
```

At the bottom of the window, there are controls: Autoscroll, Show timestamp, a 'Newline' dropdown menu, '9600 baud' dropdown menu, and a 'Clear output' button.

Slika 2.30: Ispis podataka sa senzora BME280 putem SPI komunikacije (izvor: „Autorski rad“)

2.4. MJERENJE RAZINE TEKUĆINE

Za mjerenje razine tekućine koristi se plastični plutajući senzor koji radi kao prekidač. Ovaj se senzor postavlja okomito na tekućinu čiji razinu pratimo. Radi na jednostavnoj i pouzdanoj osnovi, koristi se plutajući plovak koji se diže i spušta s razinom tekućine. Plovak je pričvršćen na mehanizam prekidača koji djeluje na temelju magnetskog polja kada razina tekućine dosegne određenu točku. Neki od uobičajenih načina primjene ovog senzora nivoa tekućine su:

- kontrola pumpi
- indikacija razine vode u spremniku
- alarmiranje nivoa
- uparivanje s drugim uređajima za praćenje i kontrolu.

Na slici 2-31 prikazan je plutajući plovak, a u tablici 2-6 dane su tehničke karakteristike plovka.



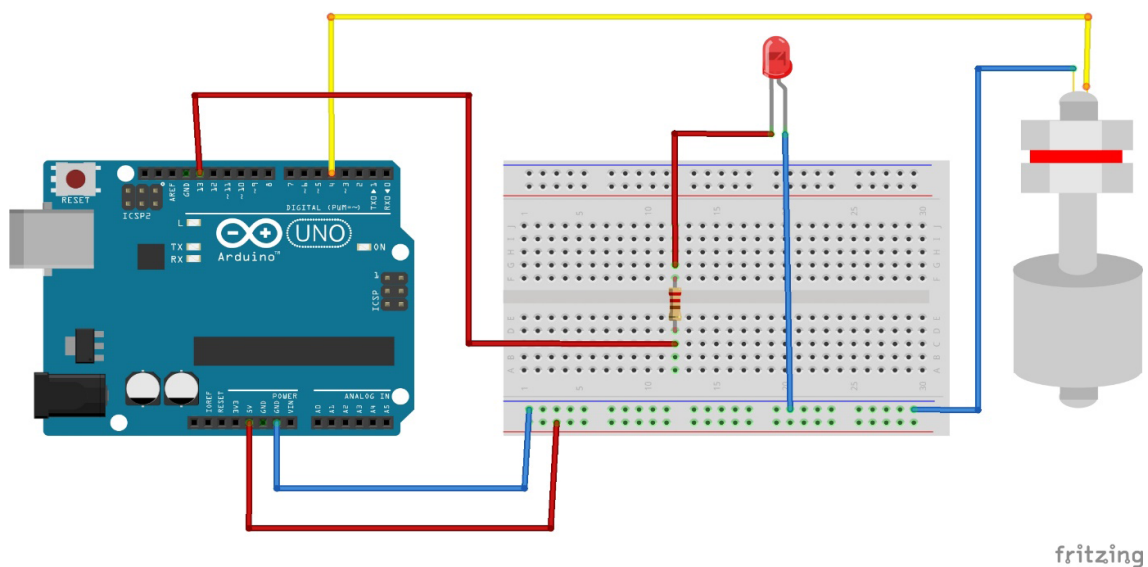
Slika 2.31: Plutajući plovak (izvor: „Autorski rad“)

Tablica 26: Tehničke karakteristike plutajućeg plovka

Plutajući plovak	Tehnički podaci
najveći iznos snage kontakta	10 W
najveći iznos napona prekidanja	220V DC/AC
najveći iznos struje prekidanja	1,5 A
najveći iznos probojnog napona	300 V DC/AC

Plutajući plovak	Tehnički podaci
najveći iznos struje	3 A
najveći iznos kontaktnog otpora	100 mΩ
najveći iznos temperature	-10 / +85 °C
promjer navoja	9,5 mm / 0,374"
veličina kućišta prekidača	23,3 x 57,7 mm / 0,9" x 2,27"
duljina kabela	36 cm / 14,2"
neto težina	70 g

Na slici 2-32 dana je shema povezivanja plutajućeg plovka koji je povezan na priključak 4 na Arduino pločici, a signalizacijska LED dioda na priključak 13. Aplikacija za mjerenje razine tekućine dana je u programskom kôdu 2-7.



Slika 2.32: Shema povezivanja plutajućeg plovka i Arduina (izvor: „Autorski rad“)

Programski kôd 28: Programski kod za očitavanje razine s plutajućeg plovka

```

/*
  Plastični plutajući prekidač, senzor razine tekućine
  */

#define PLOVAK 4 // broj pina za plovak
#define LED 13 // broj pina za LED diodu

void setup()

```

```
{
  // inicijalizacija LED pina kao izlaza
  pinMode(LED, OUTPUT);
  // Inicijalizacija pina za plovak kao ulaza
  pinMode(PLOVAK, INPUT_PULLUP); // Unutarnji Arduino otpornik 10k
}
void loop()
{
  if(digitalRead(PLOVAK) == LOW)
  {
    // upali (LED on)
    digitalWrite(LED, HIGH);
    Serial.println("U vodi");
  }
  else
  {
    // ugasi (LED off)
    digitalWrite(LED, LOW);
    Serial.println("Izvan vode");
  }
}
```

Kada na priključku "PLOVAK" ima napona, on se nalazi u visokom stanju (stanje 1), a to označava da je razina vode ispod predviđene granice. Na zaslonu računala ispisuje se poruka "Izvan vode" i LED dioda ugašena je. Kada na priključku "PLOVAK" nema napona, on se nalazi u niskom stanju (stanje 0), a to označava da je razina vode visoka i da je dosegla predviđenu razinu. Na zaslonu računala ispisuje se poruka "U vodi" i LED dioda svijetli.

3

POGLAVLJE

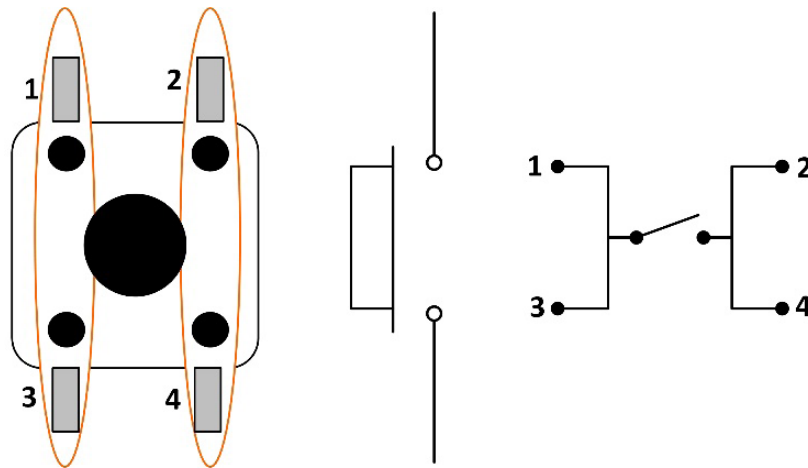
PRIMJENA ULAZNO- IZLAZNIH JEDINICA

Nakon ovog poglavlja moći ćete:

- objasniti načine primjene ulaznih jedinica
- opisati način rada LCD zaslona
- izraditi programe za kontrolu ulazno-izlaznih uređaja: prekidača, tipkovnice i LCD zaslona.

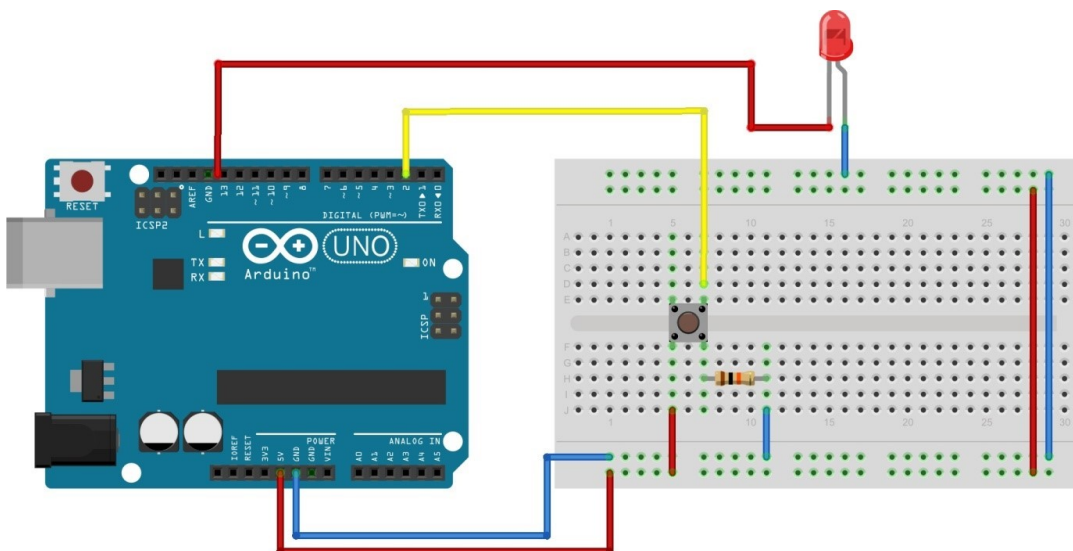
3.1. KONTROLA ULAZNO-IZLAZNIH PRIKLJUČAKA

Sljedeći primjer prikazuje primjenu ulazno-izlaznih priključaka na Arduino pločici. U primjeru se kao ulazna jedinica koristi mehanički prekidač (eng. *push button*) i LED dioda kao izlazni signalizacijski element. Mehanički prekidač ili tipkalo mala je komponenta koja pomoću pritiska otvara ili zatvara električni krug, slika 3-1. Kada je pritisnuta, električni se kontakti unutar prekidača zatvaraju i omogućavaju protok struje kroz električni krug.



Slika 3.0.1: Mehanički prekidač (izvor: „Autorski rad“)

Shema povezivanja Arduino pločice, mehaničkog prekidača i LED diode prikazana je na slici 3-2, a programski kôd u 3-1.



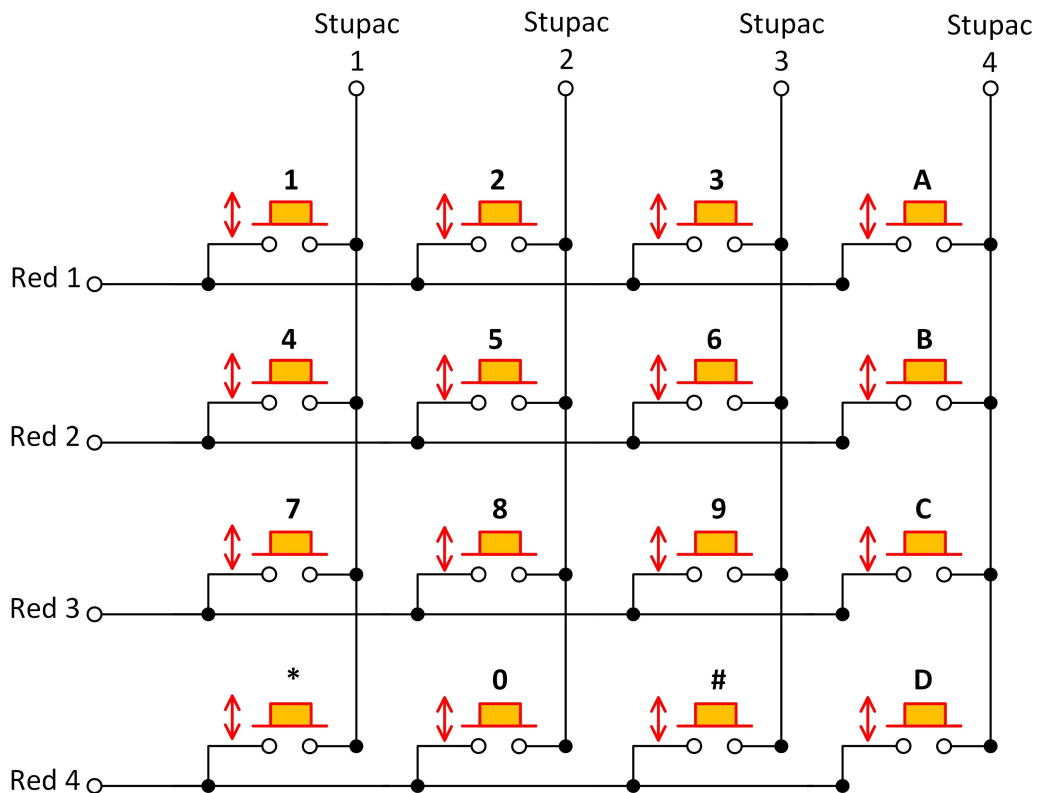
Slika 3.2: Shema povezivanja mikrokontrolera, prekidača i LED diode (izvor: „Autorski rad“)

Programski kôd 3.1: Program za kontrolu ulazno-izlaznih priključaka

```
/* Pali i gasi LED diodu koja je spojena na digitalni pin 13,
u trenutku kad se pritisne tipkalo koje je spojeno na pin 2.
Krug:
* LED dioda spojena s pina 13 na masu
* Tipkalo spojeno na pin 2 i napon od +5 V
* 10 kW otpornik spojen između pina 2 i mase */
// postavljanje pina tipkala
const int buttonPin = 2;
// postavljanje pina od LED diode
const int ledPin = 13;
// varijabla za čitanje statusa tipke
int buttonState = 0;
void setup() {
  // postavljanje LED pina kao izlaznog:
  pinMode(ledPin, OUTPUT);
  // postavljanje pina od tipkala kao ulaznog:
  pinMode(buttonPin, INPUT);
}
void loop(){
  // čitanje stanja na pinu od tipkala
  buttonState = digitalRead(buttonPin);
  // provjera je li tipka pritisnuta, ako jest, onda je stanje HIGH
  if (buttonState == HIGH) {
    // upali LED diodu
    digitalWrite(ledPin, HIGH);
  }
  else {
    // ugasi LED diodu
    digitalWrite(ledPin, LOW);
  }
}
```

3.2. PRIMJENA MEMBRANSKE TIPKOVNICE

Tipkovnica služi kao ulazna jedinica za unos podataka za različite aplikacije. U ovom projektu koristi se 4 x 4 tipkovnica koja se spaja s Arduinoom. Ispod svake tipke nalazi se membranski prekidač. Električna shema tipkovnice prikazana je na slici 3-3.



Slika 3.3: Električna shema tipkovnice (izvor: „Autorski rad“)

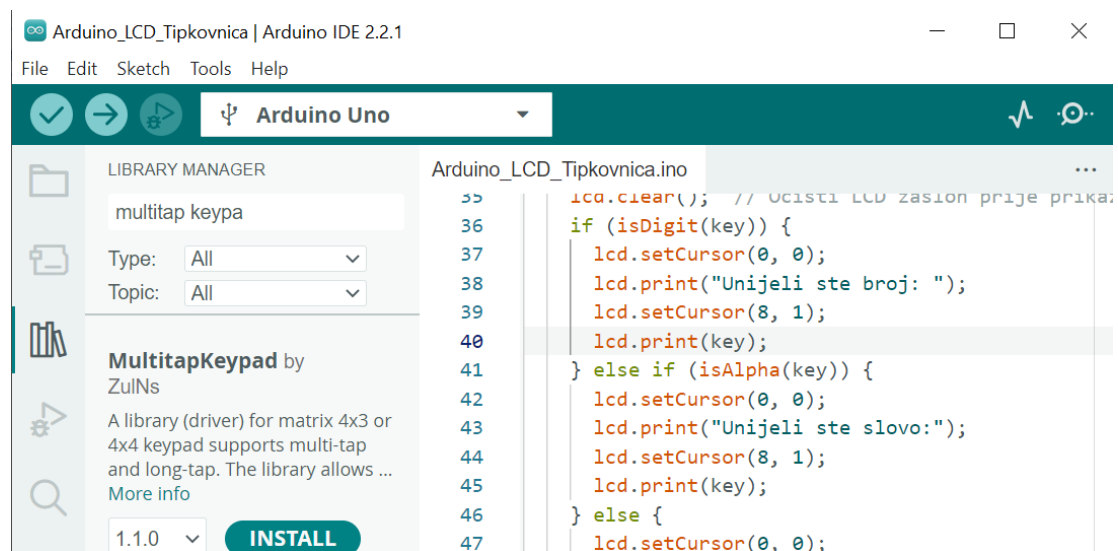
Svaki prekidač u retku povezan je s drugim prekidačima u istom retku žicom ispod podloge. Svaki prekidač u stupcu spojen je na isti način, jedna strana prekidača spojena je žicom na sve ostale prekidače u tom stupcu. Pritiskom na tipku zatvara se prekidač između stupca i retka te dopušta da struja teče među njihovim priključcima. Svaki redak i stupac spojeni su na jedan Arduino ulazni priključak, što znači da je potrebno ukupno 8 podatkovnih priključaka za 4 x 4 tipkovnicu.

Tehničke karakteristike tipkovnice 4 x 4 su:

- **veličina pločice:** 76 x 76 x 0,8 mm
- **dužina kabela:** 85 mm (uključujući konektor)
- **konektor:** Dupont s 8 priključaka, razmak među priključcima od 0,254 mm
- **način montaže:** samoljepljiva traka

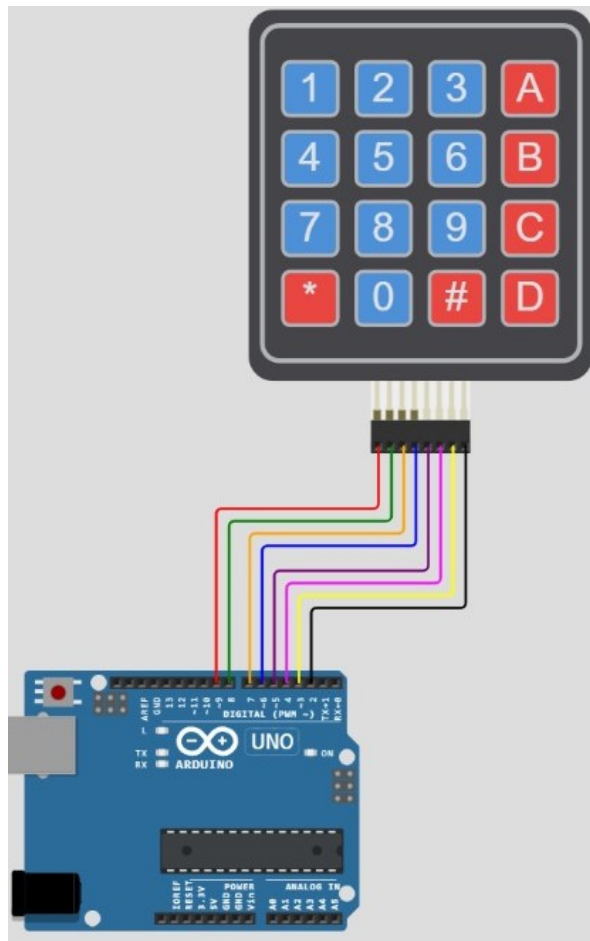
- **najveći iznos DC napona i struje:** 35 V, 100 mA
- **specifikacija izolacije:** 100 MΩ na 100 V
- **električna čvrstoća izolacije:** 250 V (efektivni napon), frekvencija 60Hz uz 1 minutu opterećenja
- **oscilacije prilikom pritiska ili puštanja tipke:** ≤5 ms
- **očekivano trajanje:** 1 milijun zatvaranja
- **težina:** 7,5 g.

Za korištenje tipkovnice prvo je potrebno instalirati biblioteku *Keypad.h*, slika 3.4.



Slika 3.4: Instalacija biblioteke za podršku tipkovnice (izvor: „Autorski rad“)

Na slici 3-5 prikazana je shema povezivanja tipkovnice i Arduino pločice. U programskom kôdu 3-2 dan je primjer aplikacije za učitavanje znakova s tipkovnice, a na slici 3-6 ispis unesenih znakova na zaslonu računala.



Slika 3.5: Shema povezivanja tipkovnice i Arduino pločice (izvor: „Autorski rad“)

Programski kôd 3.2: Program za unošenje znakova s tipkovnice

```

/*
Primjer korištenja tipkovnice za unos podataka
*/
// Biblioteka za podršku tipkovnice
#include <Keypad.h>
// Definiranje broja redova i stupaca
const int numRows = 4;
const int numCols = 4;
// Postavljanje vrijednosti za svaku tipku
char keymap[numRows][numCols] = {
    {'1','2','3','A'},
    {'4','5','6','B'},
    {'7','8','9','C'},
    {'*','0','#','D'}
};

```

```
// poveži redove na pinove R1 na 9, R2 na 8, R3 na 7 i R4 na 6
byte rowPins[numRows] = {9, 8, 7, 6};
// poveži stupce na pinove C1 na 5, C2 na 4, C3 na 3 i C4 na 2
byte colPins[numCols] = {5, 4, 3, 2};
// definicija podatkovnog objekta keypad
Keypad keypad = Keypad(makeKeymap(keymap), rowPins, colPins, numRows,
numCols);

void setup() {
  Serial.begin(9600);
}

void loop() {
  // Kada se tipka pritisne, spremi u varijablu key
  char key = keypad.getKey();
  // ispisivanje što je pritisnuto
  if (key) {
    if (isDigit(key)) {
      Serial.print("Unijeli ste broj: ");
      Serial.println(key);
    } else if (isAlpha(key)) {
      Serial.print("Unijeli ste slovo: ");
      Serial.println(key);
    } else {
      Serial.print("Unijeli ste znak: ");
      Serial.println(key);
    }
  }
}}
```



```
COM3
Unijeli ste broj: 1
Unijeli ste slovo: A
Unijeli ste slovo: A
Unijeli ste slovo: B
Unijeli ste znak: #
Unijeli ste broj: 0
```

Slika 3.6: Ispis znakova unesenih preko tipkovnice (izvor: „Autorski rad“)

3.3. ARDUINO ANALOGNO-DIGITALNA PRETVORBA

Arduino koristi analogno-digitalni pretvornik (engl. *Analog-to-Digital Converter*) za pretvaranje analognih signala u digitalne. Osnovna načela rada Arduino AD pretvornika su:

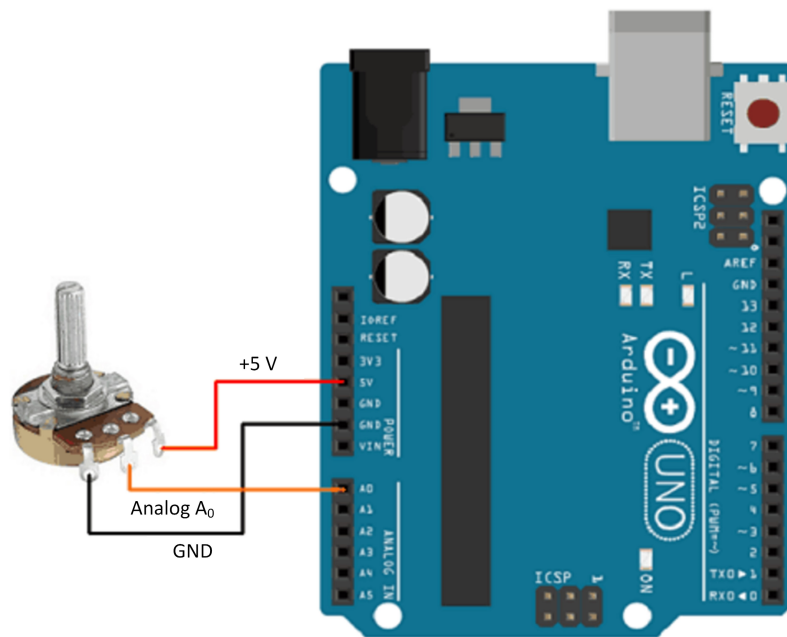
- **Referentni napon:** AD pretvornik pretvara analogni signal u digitalni tako što ga uspoređuje s referentnim naponom. Arduino ima unaprijed postavljen referentni napon od 5 V, što znači da će AD pretvornik mjeriti analogni signal u odnosu na ovaj napon.
- **Razlučivost (engl. Resolution):** razlučivost AD pretvornika određuje koliko je moguće diskretnih vrijednosti signala dobiti kao rezultat pretvorbe. Arduino koristi 10-bitni AD pretvornik, što znači da je moguće proizvesti 2^{10} , odnosno 1024 različitih digitalnih vrijednosti.
- **Analogni ulazi:** Arduino Uno ima šest analognih ulaza (A0–A5) koji su povezani na AD pretvornik.
- Čitanje **vrijednosti:** kada korisnik zatraži čitanje analognog signala s određenog analognog ulaza (npr. A0), mikrokontroler koristi AD pretvornik za uzorkovanje napona na tom ulazu u odnosu na referentni napon.
- **Digitalizacija:** AD pretvornik zatim digitalizira uzorkovani analogni signal u digitalni oblik pomoću svoje razlučivosti.
- **Prikaz vrijednosti:** primjerice, ako AD pretvornik prikazuje digitalnu vrijednost 512, to bi značilo da je analogni signal oko polovice vrijednosti između 0 V i 5 V.

Očitana digitalna vrijednost s AD pretvornika može se pretvoriti u napon, primjerice, ako Arduino očitava digitalnu vrijednost 400, tada je napon moguće izračunati pomoću jednadžbe:

$$U_{ul}(V) = 400 \frac{5 V}{1023} = 1,96 V.$$

Jednadžba 31: Računanje napona na AD pretvorniku

Slijedi primjer korištenja AD pretvornika koji očitava vrijednost napona s 10 k Ω potencijometra, slika 3-7.



Slika 3.7: Shema povezivanja Arduina i potencijometra (izvor: „Autorski rad“)

Funkcija `analogRead(analogni_priključak)` čita digitalnu vrijednost koja je dovedena na analogni priključak, programski kôd 3-3.

Programski kôd 3.3: Primjena AD pretvorbe

```

/*
Primjena AD pretvorbe
*/
int sensorPin = A0; // analogni ulaz za potencijometar
int digitalV = 0; // pohrana očitane vrijednosti s analognog ulaza
float analogV = 0.00; //varijabla za prikaz napona

void setup() {
  Serial.begin(9600);
}

void loop() {
  // čitanje analogne vrijednosti
  digitalV = analogRead(sensorPin);
  Serial.print("digitalna vrijednost = ");
  //ispis digitalne vrijednosti na zaslon
  Serial.print(digitalV);
  //pretvorba digitalne vrijednosti u napon

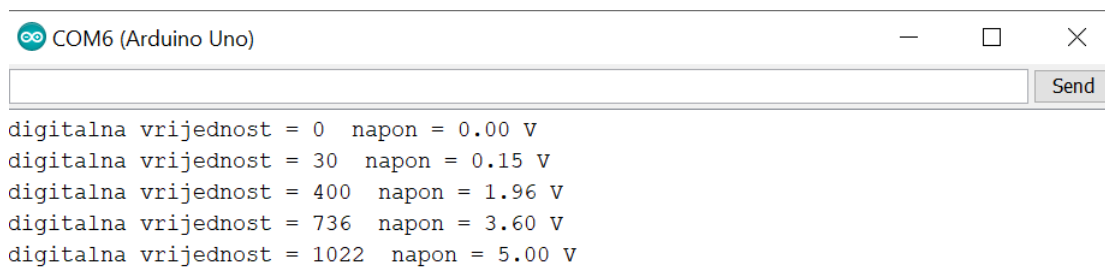
```

```

analogV = (digitalV * 5.00)/1023.00;
Serial.print(" napon = ");
Serial.print(analogV);
Serial.println(" V");
delay(1000);
}

```

Slika 3-8 prikazuje ispis na zaslon računala vrijednosti očitanih s potenciometra i rezultat AD pretvorbe.



```

digitalna vrijednost = 0 napon = 0.00 V
digitalna vrijednost = 30 napon = 0.15 V
digitalna vrijednost = 400 napon = 1.96 V
digitalna vrijednost = 736 napon = 3.60 V
digitalna vrijednost = 1022 napon = 5.00 V

```

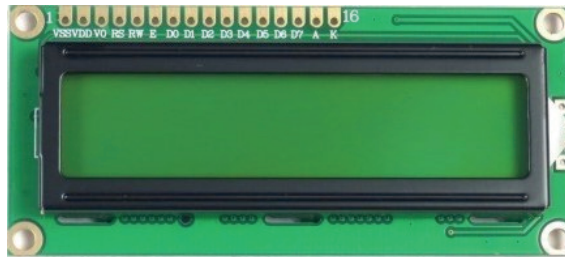
Slika 3.8: Ispis na zaslon računala očitanih vrijednosti s potenciometra i AD pretvorba (izvor: „Autorski rad“)

3.4. PRIMJENA LCD ZASLONA

3.4.1. LCD zaslon

LCD (engl. *Liquid Crystal Display*) zaslon je s tekućim kristalima koji služe za prikaz znakova, slika 3-9. Kada kroz tekući kristal teče struja, oni postaju neprozirni i blokiraju pozadinsko osvjetljenje koje se nalazi iza zaslona. Kao rezultat toga, područje kroz koje teče struja bit će tamnije od ostatka zaslona. Aktiviranjem sloja tekućih kristala u određenim pikselima mogu se generirati različiti znakovi. Ako se pažljivo pogleda, mogu se uočiti sićušni pravokutnici za svaki znak na zaslonu. Svaki od ovih pravokutnika predstavlja matricu veličine 5 x 8 piksela, slika 3-10. LCD zasloni sa znakovima dostupni su u raznim veličinama i bojama:

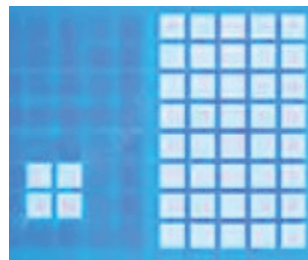
- izvedba sa 16 stupaca i jednim retkom (16 × 1)
- izvedba sa 16 stupaca i dva retka (16 × 2)
- izvedba sa 16 stupaca i četiri retka (16 × 4)
- izvedba s 20 stupaca i četiri retka (20 × 4).



U ovom priručniku koristit će se LCD zaslon sa 16 stupaca i dva retka (16 × 2), tablica 3-1.



Slika 3.9: Prikaz LCD 16 x 2 zaslona (izvor: „Autorski rad“)



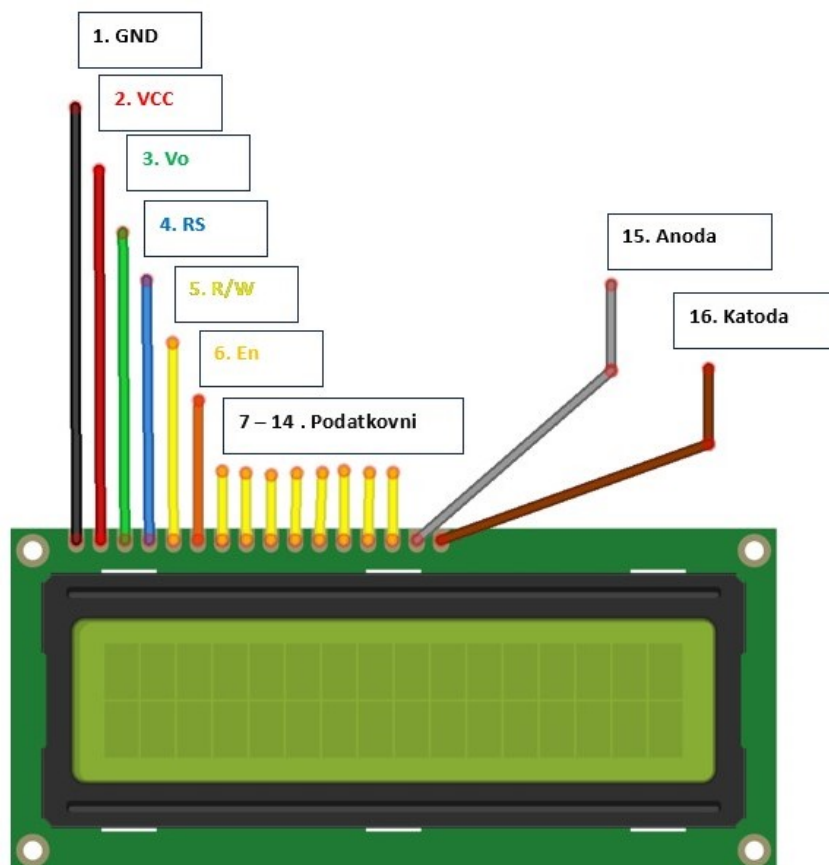
Slika 3.10: Matrica piksela reda 5 x 8 (izvor: „Autorski rad“)

Tablica 31: Tehnička specifikacija LCD zaslona

LCD priključak	Značenje
GND	priključak za uzemljenje, odnosno masu
VCC	priključak za napajanje, 5 V
Vo (LCD kontrast)	priključak za kontrolu kontrasta LCD-a, koristeći naponski djelitelj, može se precizno namjestiti kontrast
RS (engl. <i>Register Select</i>)	priključak se koristi za odvajanje naredbi od podataka, kao što je postavljanje pokazivača na određenu lokaciju, brisanje zaslona sl., RS priključak postavljen je na <i>LOW</i> kada se šalju naredbe te na <i>HIGH</i> kada se šalju podatci na LCD

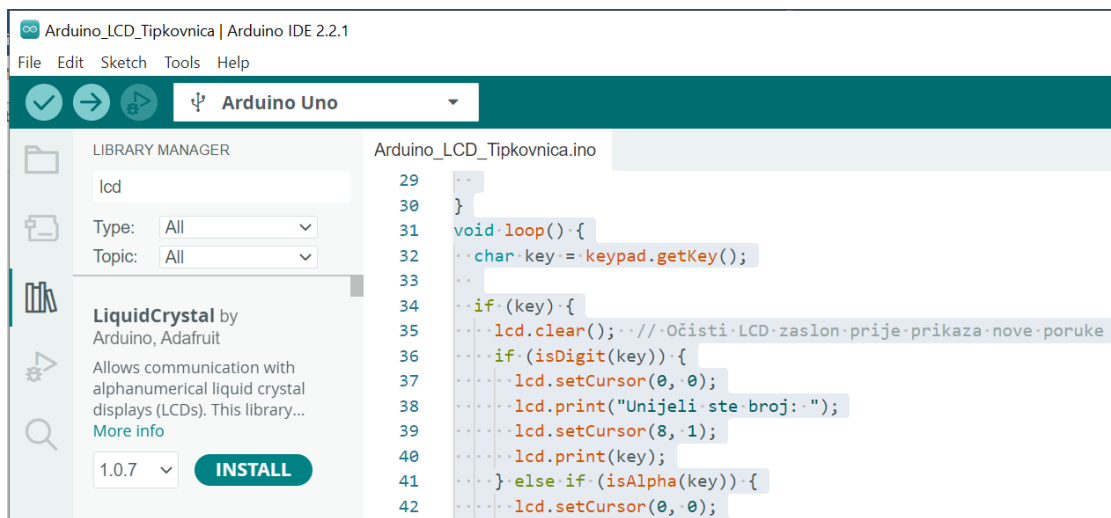
R/W (engl. <i>Read/Write</i>)	priključak omogućuje čitanje podataka s LCD zaslona ili pisanje podataka na LCD zaslon; u primjerima u ovom priručniku LCD koristi se samo kao izlazni uređaj i ovaj je priključak postavljen na <i>LOW</i> , što znači da je u načinu rada za pisanje podataka
E (engl. <i>Enable</i>)	priključak se koristi za omogućavanje prikaza: kada je ovaj priključak postavljen na <i>LOW</i> , LCD ignorira aktivnost na R/W, RS i linijama sabirnice podataka, a kada je postavljen na <i>HIGH</i> , tada LCD obrađuje dolazne podatke
D0-D7 (engl. <i>Data Bus</i>)	priključci od D0 do D7 služe za prijenos 8-bitnih podataka koji se šalju na zaslon, npr. za prikaz velikog slova A na zaslonu ovi se priključci postavljaju na 0100 0001 (prema ASCII kôdu 65)
A-K (engl. <i>Anode & Cathode</i>)	ovi priključci koriste se za kontrolu pozadinskog osvjetljenja LCD zaslona

Na slici 3-11 prikazan je raspored i značenje priključaka LCD zaslona.

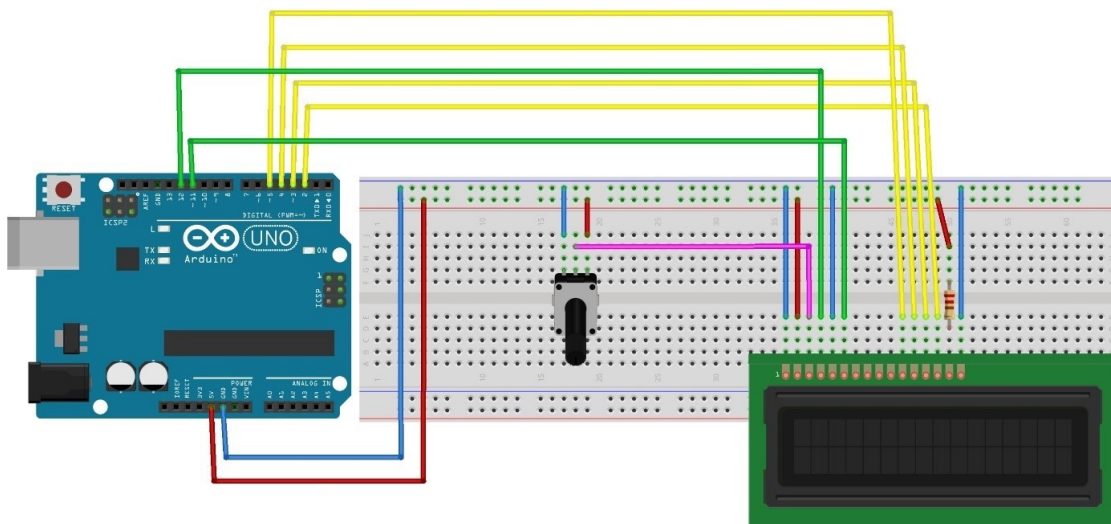


Slika 3.11: Raspored i značenje priključaka LCD zaslona (izvor: „Autorski rad“)

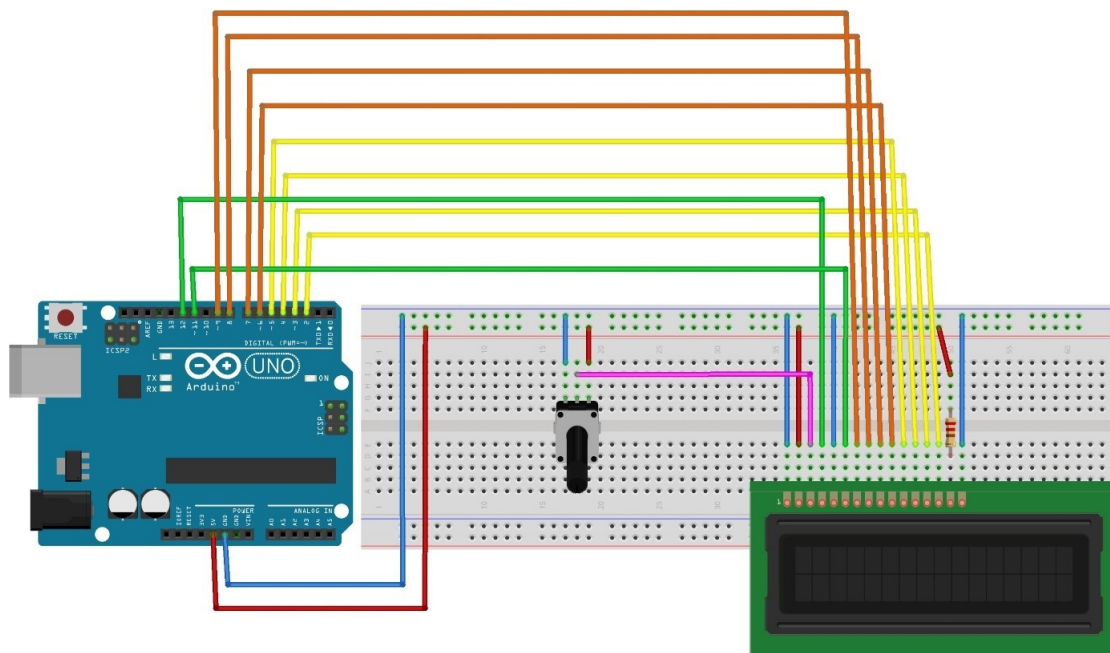
Neki LCD zasloni uključuju otpornik za ograničavanje struje za pozadinsko osvjetljenje ovisno o proizvođaču. Ovaj otpornik nalazi se na stražnjoj strani LCD zaslona u blizini priključka broj 15. Ako LCD ne sadrži ovaj otpornik, potrebno je dodati otpornik od $220\ \Omega$. Za pravilno rukovanje LCD zaslonom potrebno je pogledati tehničku dokumentaciju proizvođača. Za fino podešavanje kontrasta zaslona potrebno je spojiti potencijometar od $10\ k\Omega$. Jedan izvod potencijometra na napajanje od 5 V, drugi izvod na uzemljenje, a srednji izvod s potencijometra na priključak broj 3 LCD zaslona. Prije korištenja LCD zaslona potrebno je instalirati biblioteku *LiquidCrystal.h*, slika 3-12. Postoje dva načina spajanja LCD zaslona, koristeći 4 podatkovna priključka (slika 3-13) ili 8 podatkovnih priključaka (slika 3-14).



Slika 3.12: Instalacija biblioteke *LiquidCrystal.h* (izvor: „Autorski rad“)



Slika 3.13: Spajanje LCD zaslona s 4 podatkovna priključka (izvor: „Autorski rad“)



Slika 3.14: Spajanje LCD zaslona s 8 podatkovnih priključaka (izvor: „Autorski rad“)

Kada se koristi 8 podatkovnih priključaka, u jednom trenutku prenosi se 8 bitova (jedan bajt) podataka, a kada se koriste 4 podatkovna priključka, u jednom trenutku prenose se 4 bita (engl. *Nibble*) podataka. Koji će se način rada od ovih dvaju koristiti, ovisi o zahtjevima aplikacije. Kada je potrebna veća brzina prijenosa, koristi se 8-bitni način rada, a u slučaju da je potrebno koristiti manje priključaka zbog povezivanja drugih uređaja, koristi se 4-bitni način rada.

3.4.2. Aplikacija za ispis posebnih znakova na LCD zaslon

Za rad s LCD zaslonom potrebno je uključiti *LiquidCrystal.h* biblioteku. Bitne funkcije koje se koriste su:

- *lcd.home()* – postavlja kursor u gornji lijevi kut LCD-a bez brisanja zaslona
- *lcd.blink()* – prikazuje treptajući blok od 5×8 piksela na mjestu na kojem će biti upisan sljedeći znak
- *lcd.noBlink()* – isključuje treptanje LCD kursora
- *lcd.cursor()* – prikazuje donju crtu na mjestu na kojem će biti upisan sljedeći znak
- *lcd.noCursor()* – funkcija skriva LCD pokazivač
- *lcd.scrollDisplayRight()* – pomiče sadržaj zaslona jedno mjesto udesno, a ako se želi da se tekst neprekidno pomiče, mora se koristiti ova funkcija unutar *for* petlje.

Svi LCD zasloni s upravljačkim programom Hitachi HD44780 imaju dvije vrste memorije: CGROM (ROM memorija) i CGRAM (RAM memorija) za generiranje znakova. CGROM stalna je memorija koja pohranjuje font koji se pojavljuje na LCD zaslonu. Na primjer, kada se pošalje naredba za ispis slova **A**, potrebno je poznavati koji se pikseli trebaju uključiti za ispis slova **A**. Ovi podaci pohranjeni su u CGROM memoriji. CGRAM je privremena memorija za pohranu korisnički definiranih znakova. Ova memorija ima kapacitet od 64 bajta. Za LCD zaslon od 5×8 piksela moguće je definirati samo 8 korisnički definiranih znakova koji se mogu pohraniti u CGRAM, dok je za LCD zaslon od 5×10 piksela moguće pohraniti samo 4 korisnički definirana znaka.

Ukoliko klasični zadani font nije dovoljan, utoliko je moguće izraditi vlastite prilagođene znakove i simbole. Oni su korisni kada je potrebno prikazati znak koji nije u standardnom ASCII skupu znakova. Kako se svaki simbol na zaslonu sastoji od matrice 5×8 piksela, potrebno je prilagođeni znak definirati unutar ove 5×8 matrice. Za to se koristi funkcija `createChar()`. Kako bi se koristila ova funkcija, prvo je potrebno stvoriti niz od 8 bajtova. Svaki bajt u nizu odgovara retku u matrici 5×8 . Unutar bajta binarne znamenke 0 i 1 označavaju koji pikseli u nizu trebaju biti isključeni, a koji uključeni. Svi ovi korisnički definirani znakovi pohranjeni su u CGRAM memoriju LCD zaslona. Programski kôd 3-4 prikazuje primjer kreiranja četiriju posebnih znakova: srce, zvono, zvučnik i znak za zaključano, slika 3-15. Za povezivanje LCD zaslona i Arduino pločice koristi se shema prikazana na slici 3-13.

Programski kôd 34: Ispis posebnih znakova na LCD zaslon

```
/*
LCD POSEBNI ZNAKOVI
Program stvara i ispisuje četiri posebna znaka na 16 x 2 LCD zaslon.
To su znakovi: srce, zvono, zvučnik i zaključano. */
// uključivanje LiquidCrystal.h biblioteke
// LCD biblioteka: https://www.ardumotive.com/i2clcden.html
#include <LiquidCrystal.h>
// dodjeljivanje pinova za komunikaciju s LCD zaslonom.
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// kodovi za posebno stvorene znakove
// srce
byte Srce[8] = {
0b00000,
0b01010,
0b11111,
0b11111,
0b01110,
0b00100,
```

```
0b00000,
0b00000};
// zvono
byte Zvono[8] = {
0b00100,
0b01110,
0b01110,
0b01110,
0b11111,
0b00000,
0b00100,
0b00000};
// zvučnik
byte Zvucnik[8] = {
0b00001,
0b00011,
0b01111,
0b01111,
0b01111,
0b00011,
0b00001,
0b00000};
// zaključeno
byte Zakljucano[8] = {
0b01110,
0b10001,
0b10001,
0b11111,
0b11011,
0b11011,
0b11111,
0b00000};

void setup()
{
    // inicijalizacija LCD zaslona
    lcd.begin(16, 2);
    // stvaranje novih znakova
    lcd.createChar(0, Srce);
    lcd.createChar(1, Zvono);
    lcd.createChar(2, Zvucnik);
    lcd.createChar(3, Zakljucano);
    // isprazni LCD sučelje
    lcd.clear();
}
```

```
// Ispiši poruku na LCD
lcd.print("Poseban znak");
}

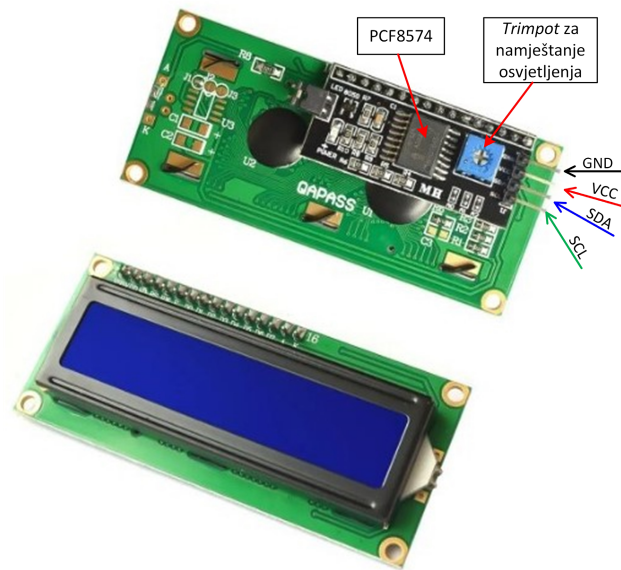
// ispisivanje svih znakova
void loop()
{
  // postavljanje kursora i prvog znaka
  lcd.setCursor(0, 1);
  lcd.write(byte(0));
  // postavljanje kursora i drugog znaka
  lcd.setCursor(2, 1);
  lcd.write(byte(1));
  // postavljanje kursora i trećeg znaka
  lcd.setCursor(4, 1);
  lcd.write(byte(2));
  // postavljanje kursora i prvog znaka
  lcd.setCursor(6, 1);
  lcd.write(byte(3));
}
```



Slika 3.15: Ispis posebnih znakova na LCD zaslon (izvor: „Autorski rad“)

3.4.3. Primjena I2C komunikacije za povezivanje LCD zaslona

Iz ranijeg primjera povezivanja LCD zaslona s Arduino pločicom može se primijetiti da zahtijeva mnogo Arduino priključaka. Čak i u 4-bitnom načinu, Arduino zahtijeva sedam veza – polovicu dostupnih digitalnih I/O priključaka na Arduino pločici. Rješenje je koristiti I2C LCD zaslon koji koristi samo dva analogna ulazna priključka i ovi se priključci mogu dijeliti s drugim I2C uređajima. Tipični I2C LCD zaslon sastoji se od HD44780 znakovnog LCD zaslona i I2C LCD adaptera. Adapter za I2C komunikaciju sastoji se od PCF8574 integriranog kruga, *trimpota* za namještanje osvjetljenja zaslona i ulazno-izlaznih priključaka, a prikazan je na slici 3-16. *Trimpot* je vrsta potenciometra, odnosno promjenjivog otpornika koji se koristi za fino namještanje električnog otpora u elektroničkim krugovima.



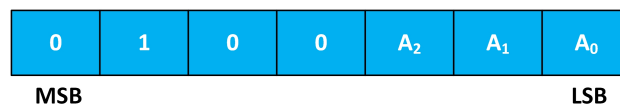
Slika 3.16: LCD zaslon s I2C adapterom (izvor: „Autorski rad“)

Za povezivanje više uređaja na istoj I2C sabirnici potrebno je postaviti različitu adresu za svaki uređaj kako bi se izbjegli sukobi s drugim uređajima na I2C sabirnici. U tu svrhu, I2C adapter dolazi s trima mjestima za lemljenje (A_0 , A_1 i A_2), a adresa se postavlja kada se lemi-licom zatvori veza (engl. *Jumper*) na određenom mjestu. Važno je napomenuti da nekoliko tvrtki, uključujući *Texas Instruments* i *NXP Semiconductors*, proizvodi isti PCF8574 čip, a I2C adresa LCD zaslona ovisi o proizvođaču čipa. Na slici 3-17 prikazan je način postavljanja adrese za *Texas Instruments* čip. Unaprijed postavljena (engl. *Default*) I2C adresa jest $0x27$.

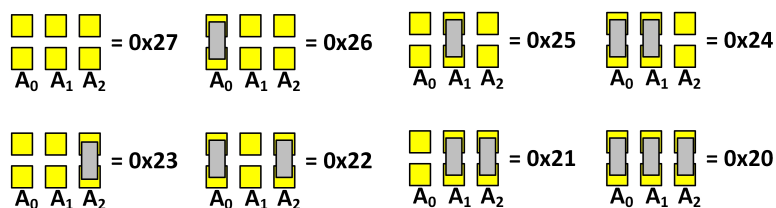
Postavljanje adrese na I2C adapteru



7-bitni I2C registar



Postavljanje različitih I2C adresa

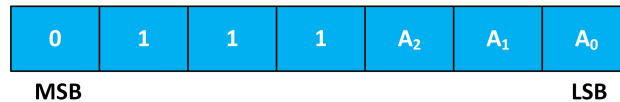
Slika 3.17: Postavljanje adrese za *Texas Instruments* čip (izvor: „Autorski rad“)

Na slici 3-18 prikazan je način postavljanja adrese za NXP Semiconductors čip. Unaprijed postavljena I2C adresa jest 0x3F.

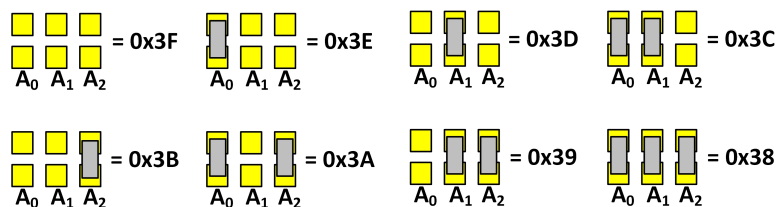
Postavljanje adrese na I2C adapteru



7-bitni I2C registar

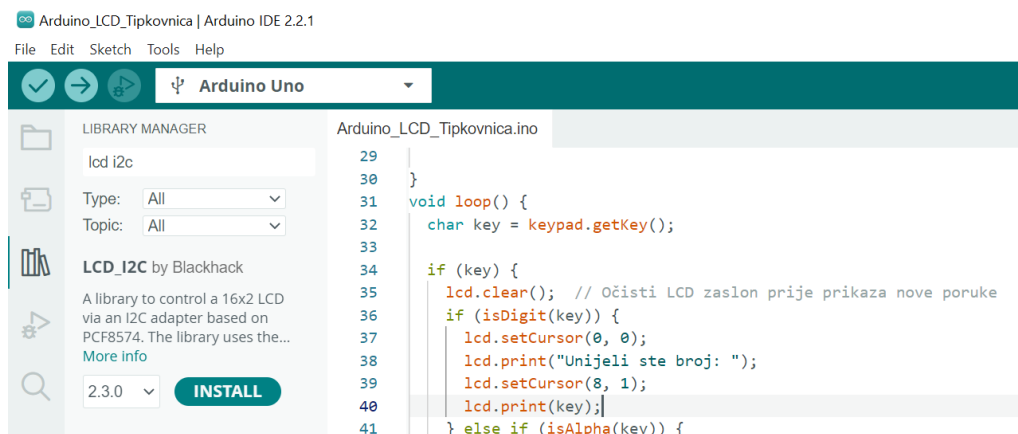


Postavljanje različitih I2C adresa

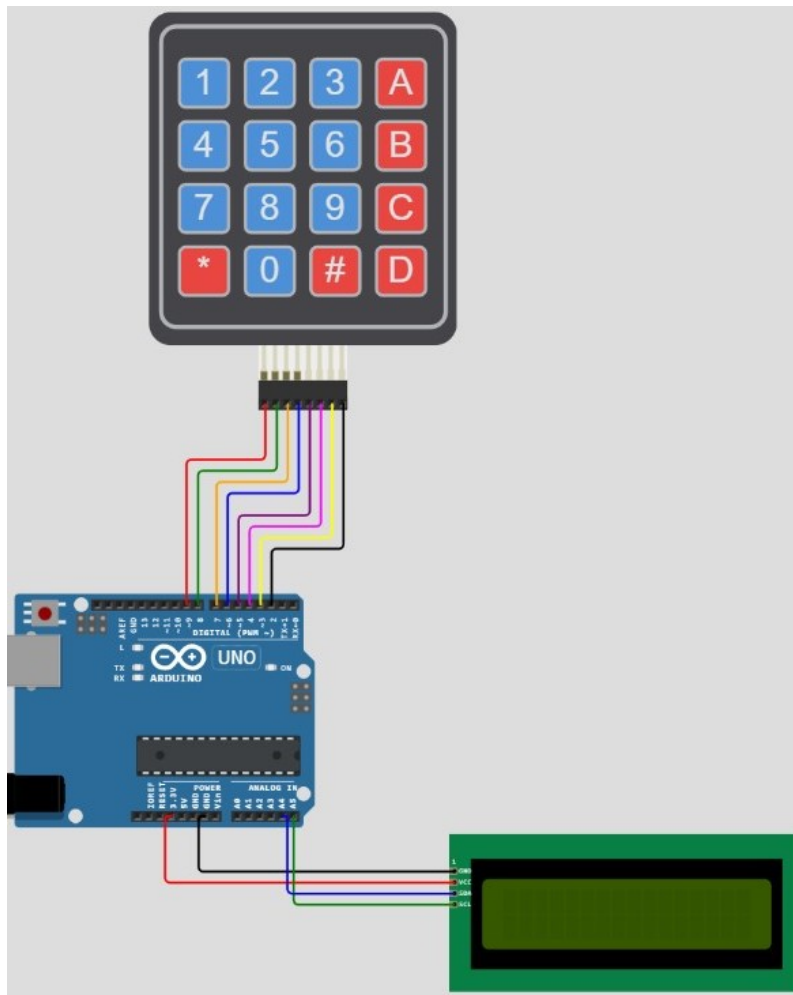


Slika 3.18: Postavljanje adrese za NXP Semiconductors čip (izvor: „Autorski rad“)

Za korištenje LCD zaslona s I2C komunikacijom potrebno je instalirati biblioteku *LiquidCrystal_I2C.h* (slika 3-19). Na slici 3-20 prikazana je shema povezivanja LCD I2C zaslona, tipkovnice i Arduino pločice, a programski kod 3-5 daje primjer aplikacije za prikaz znakova na zaslonu unesenih s tipkovnice. Prikaz ispisa na LCD I2C zaslonu dan je na slici 3-21.



Slika 3.19: Instalacija biblioteke *LiquidCrystal_I2C.h* (izvor: „Autorski rad“)



Slika 3.20: Shema povezivanja LCD I2C zaslona, tipkovnice i Arduina (izvor: „Autorski rad“)

Programski kôd 35: Unos znakova s tipkovnice i ispis na LCD I2C zaslonu

```

/*
Program za unos znakova s tipkovnice i
ispis na LCD I2C zaslonu.
*/
// Uključivanje potrebnih biblioteka za podršku žične komunikacije
// LCD I2C zaslona i tipkovnice
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
// Definiranje broja redova i stupaca tipkovnice
const int numRows = 4;
const int numCols = 4;
// Postavljanje vrijednosti za svaku tipku
char keymap[numRows][numCols] = {

```

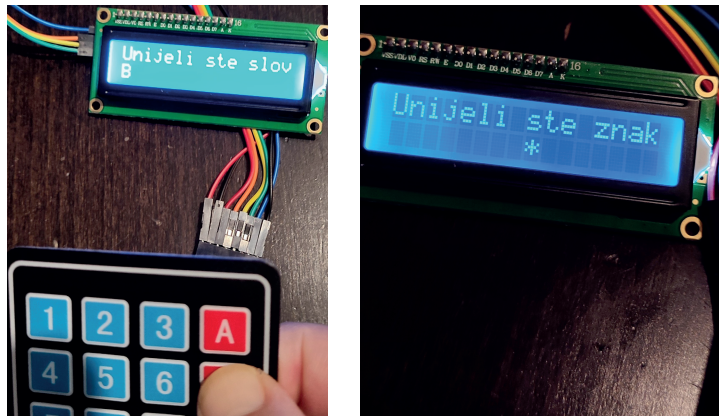
```
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};
// poveži redove na pinove R1 na 9, R2 na 8, R3 na 7 i R4 na 6
byte rowPins[numRows] = {9, 8, 7, 6};
// poveži stupce na pinove C1 na 5, C2 na 4, C3 na 3 i C4 na 2
byte colPins[numCols] = {5, 4, 3, 2};
// definicija podatkovnog objekta keypad
Keypad keypad = Keypad(makeKeymap(keymap), rowPins, colPins, numRows,
numCols);
// Inicijalizacija LCD zaslona, 0x27 je default adresa
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  Serial.begin(9600);

  lcd.init();           // Inicijalizacija LCD-a
  lcd.backlight();     // Uključivanje pozadinskog osvjetljenja
  lcd.print("Pritisnite tipku!");
}

void loop() {
  //čitanje koja tipka je pritisnuta
  char key = keypad.getKey();
  // Ispis na LCD zaslon
  if (key) {
    lcd.clear(); // Očisti LCD zaslon prije prikaza nove poruke
    if (isDigit(key)) {
      lcd.setCursor(0, 0);
      lcd.print("Unijeli ste broj: ");
      lcd.setCursor(8, 1);
      lcd.print(key);
    } else if (isAlpha(key)) {
      lcd.setCursor(0, 0);
      lcd.print("Unijeli ste slovo:");
      lcd.setCursor(8, 1);
      lcd.print(key);
    } else {
      lcd.setCursor(0, 0);
      lcd.print("Unijeli ste znak: ");
      lcd.setCursor(8, 1);
      lcd.print(key);
    }
  }
}
```

```
}  
delay(2000); // Pričekaj kratko kako bi poruka bila vidljiva  
lcd.clear(); // Očisti LCD zaslon za sljedeću poruku  
}  
}
```



Slika 3.21: Prikaz unesenih znakova na LCD I2C zaslonu (izvor: „Autorski rad“)

4

POGLAVLJE

ARDUINO PROJEKTI

Nakon ovog poglavlja moći ćete:

- osmisлити Arduino projekt za automatizaciju procesa
- objasniti model sustava automatizacije procesa
- izraditi električnu shemu sustava automatizacije
- izraditi aplikaciju za sustav automatizacije procesa.

4.1. TERMOSTAT ZA UPRAVLJANJE SUSTAVOM GRIJANJA ILI HLAĐENJA

4.1.1. Projektni zadatak

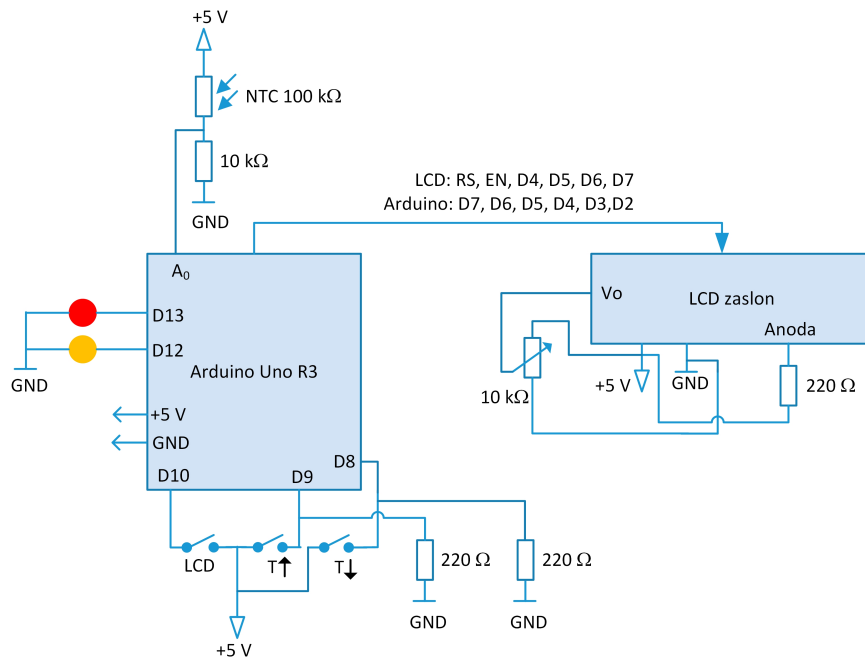
Potrebno je izraditi termostat za upravljanje sustavom grijanja ili hlađenja. Termostat je osmišljen na način da ima dvije tipke kojima korisnik može zadati radnu temperaturu. Jedna je od tipki za odabir više temperature, a druga je tipka za odabir niže temperature. Trenutna izmjerena temperatura uspoređuje se sa zadanom radnom temperaturom. Ako je izmjerena temperatura manja od zadane radne temperature, pali se crvena LED dioda koja označava da se uključilo grijanje. Isključeni grijač predstavljen je svijetljenjem žute LED diode. Za očitavanje temperature koristi se NTC termistor od 100 k Ω i serijski otpornik od 10 k Ω . Na LCD zaslonu ispisuju se potrebni podaci te se zaslon može paliti i gasiti pritiskom na tipku. Ukoliko se želi sačuvati radna temperatura kao zadana vrijednost, utoliko ju je potrebno pohraniti u EEPROM.

4.1.2. Dizajn projekta

Za realizaciju projekta potrebne su sljedeće komponente:

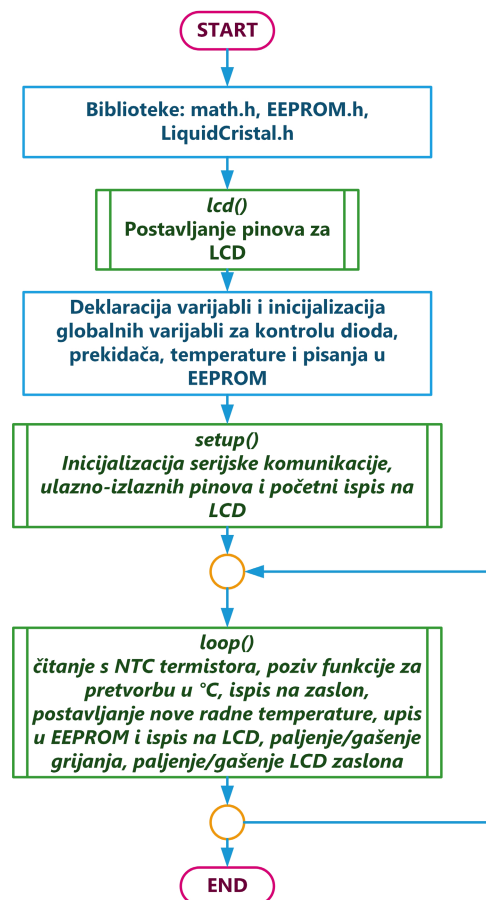
- Arduino Uno pločica
- LCD zaslon
- NTC termistor 100 k Ω
- žuta i crvena LED dioda
- potenciometar od 10 k Ω
- tri prekidača
- tri otpornika od 220 Ω
- otpornik od 10 k Ω
- razvojna pločica (engl. *Breadboard*)
- žice za povezivanje.

Sve ove komponente ranije su opisane u priručniku, kao i način njihovog povezivanja s Arduino pločicom. Za realizaciju sustava termostata komponente je potrebno povezati s Arduino pločicom na način prikazan na slici 4-1.



Slika 4.1: Povezivanje komponenti sustava termostata (izvor: „Autorski rad“)

Na slici 4-2 prikazan je blok-dijagram glavnog programa za realizaciju sustava termostata *Termostat.ino*.

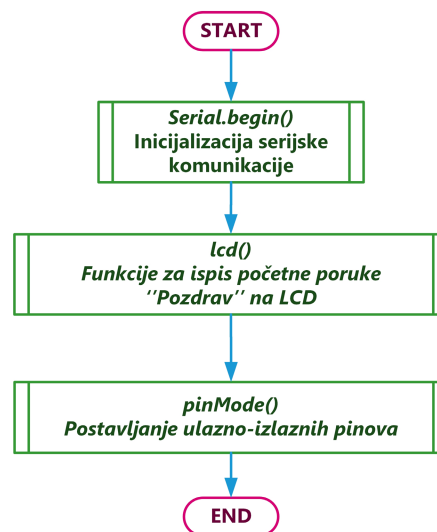


Slika 4.2: Blok-dijagram programa *Termostat.ino* (izvor: „Autorski rad“)

Za realizaciju projekta potrebno je koristiti sljedeće biblioteke:

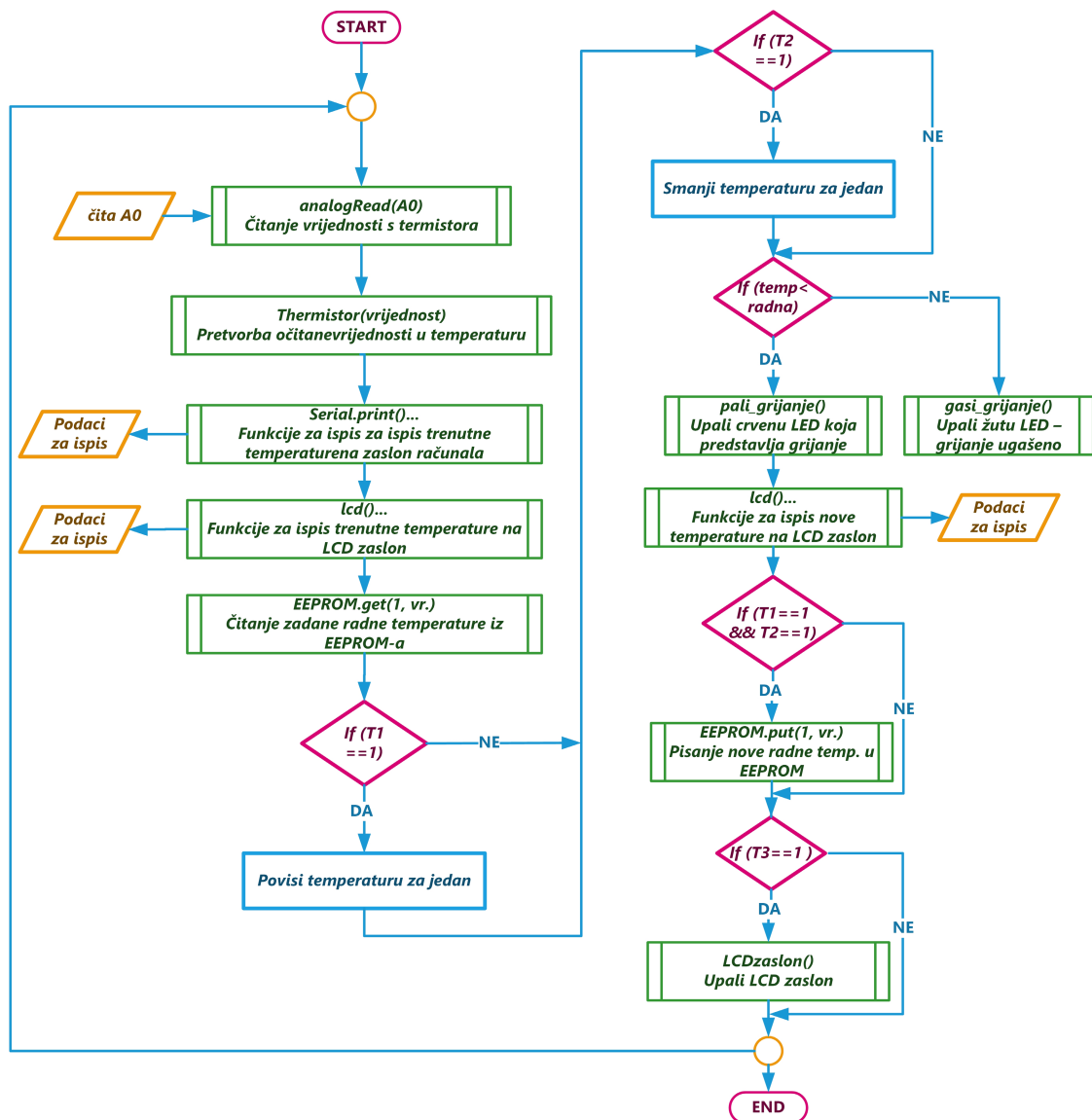
- **math.h** – biblioteka za podršku matematičkih funkcija, koristit će se *log()* funkcija za pretvorbu očitane analogne vrijednosti s termistora u temperaturu
- **EEPROM.h** – biblioteka za podršku funkcija za rad s EEPROM memorijom kako bi se pohranila zadana radna temperatura
- **LiquidCrystal.h** – biblioteka za podršku funkcija za rad s LCD zaslonom.

Unutar funkcije *setup()* potrebno je izvršiti inicijalizaciju hardvera i ispis početne poruke prema blok-dijagramu na slici 4-3.



Slika 4.3: Blok dijagram *setup()* funkcije (izvor: „Autorski rad“)

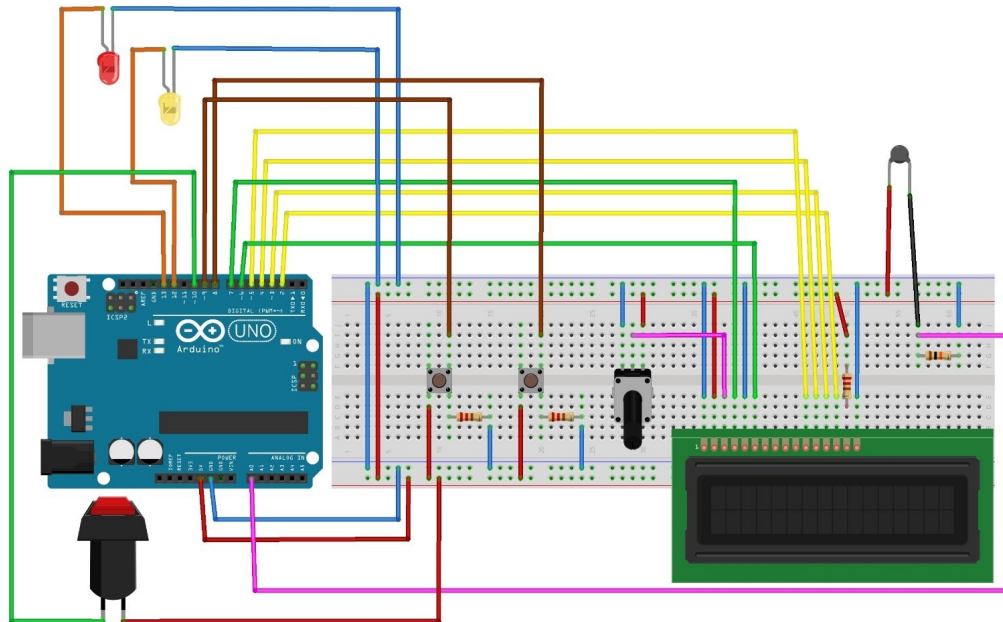
Blok dijagram programa koji se sekvencijalno izvodi i ciklički ponavlja dan je unutar *loop()* funkcije i prikazan je na slici 4-4. Pored toga, u programu je potrebno definirati dvije korisničke funkcije koje se pozivaju unutar *loop()* funkcije. Prva je funkcija *Thermistor()* koja, kao ulazni parametar, prima vrijednost očitane s analognog ulaza i pretvara je u temperaturu koju vraća kao izlaznu vrijednost. Druga je funkcija *LCDsvjetlo()* koja pali LCD zaslon kada korisnik pritisne tipku za uključivanje. Funkcija *LCDispis()* ispisuje podatke o trenutnoj temperaturi i namještenoj radnoj temperaturi na LCD zaslon.



Slika 4.4: Blok-dijagram loop() funkcije (izvor: „Autorski rad“)

4.1.3. Izvedba projekta

Na slici 4-5 dana je cjelovita električna shema sustava termostata.



Slika 4.5: Električna shema sustava termostata (izvor: „Autorski rad“)

Aplikacija za upravljanje sustavom grijanja dana je u programskom kôdu 4-1 s detaljnim komentarima.

Programski kôd 41: Aplikacija za upravljanje sustavom grijanja

```
// Uključivanje potrebnih biblioteka
#include <math.h> // biblioteka za rad s log funkcijom
#include <EEPROM.h> // biblioteka za rad s EEPROM
#include <LiquidCrystal.h> // LCD biblioteka
// pinovi za spajanje LCD zaslona 7,6,5,4,3,2
LiquidCrystal lcd(7,6,5,4,3,2);

float settemp; // varijabla za namještanje temperature
int paljenje =13; // crvena LED na pinu 13
int paljenje_fan =12; // žuta LED na pinu 12
int temp_vise = 9; // tipka za povećavanje temperature na pinu 9
int temp_nize = 8; // tipka za povećavanje temperature na pinu 8
int signal_svjetla = 10; // tipka za paljenje svjetla
int stanje_svjetla = 11; // stanje svjetla
int onoff = 0; // 0 = off, 1 = on
int8_t Echeck=0;
```

```
// Funkcija za izračun temperature, vraća očitane temperaturu
double Thermistor(int ADpretvorba) {
  double Temp;
  Temp = log(10000.0*((1024.0/ADpretvorba-1)));
  Temp = 1/(0.001129148 +(0.000234125+(0.0000000876741*Temp*Temp))*Temp);
  Temp = Temp - 273.15;
  return Temp;
}

// Funkcija za paljenje i gašenje LCD zaslona
void LCDsvjetlo()
{
  if (onoff==0)
  {
    onoff=1;
    digitalWrite(11, HIGH);
    Serial.println("LCD svijetli");
  }
  else
  if (onoff==1)
  {
    onoff=0;
    digitalWrite(11, LOW);
    Serial.println("LCD ugašen");
  }
}

void setup() {
  // Inicijalizacija serijske komunikacije
  Serial.begin(9600);
  // inicijalizacija LCD zaslona i namještanje broja stupaca i redaka
  lcd.begin(16, 2);
  // namještanje pokazivača na stupac 0, redak 0
  lcd.setCursor(0,0);
  // prilikom prvog pokretanja ispisuje se Pozdrav u trajanju od 1
  sekunde
  lcd.print("Pozdrav!");
  delay(1000);
  //očisti LCD zaslon
  lcd.clear();
  // Postavljanje izlaznih i ulaznih pinova
  pinMode(paljenje, OUTPUT);
  pinMode(paljenje_fan, OUTPUT);
  pinMode(signal_svjetla, INPUT);
}
```

```

    pinMode(stanje_svjetla, OUTPUT);

}

void loop() {
    int val;
    double temp;
    // čitanje vrijednosti s analognog ulaza
    val=analogRead(0);
    // pročitana vrijednost se šalje u funkciju za pretvorbu u temperaturu
    temp=Thermistor(val);
    // Ispis na zaslonu računala
    Serial.print("Temperature = ");
    Serial.print(temp);
    Serial.println(" C");
    delay(100);
//-----
    // Ispis na LCD zaslonu
    lcd.setCursor (0,0); // namještanje pokazivača na stupac 0, linija 0
    lcd.print ("Temp.  ");
    lcd.print (temp); // Ispis trenutne temperature u C
    lcd.print ('C');
//-----
    // čitanje namještene temperature iz EEPROM-a
    EEPROM.get(1, settemp);
    delay (250); // kašnjenje zbog stabilnosti programa
    // Ako je pritisnuta tipka za povećanje temperature
    if
        (digitalRead(temp_vise)== HIGH )
    {
        settemp ++ ; // uvećaj za jedan
        Echeck =1;
    }
    // Ako je pritisnuta tipka za smanjenje temperature
    if
        (digitalRead (temp_nize) == HIGH)
    {
        (settemp --); // umanjiti za jedan
        Echeck=1;
    }
    // ako je stvarna temperatura manja od zadane temperature
    if (temp < settemp)
    {
        digitalWrite (paljenje, HIGH); // upali grijanje – crvena LED
    }
}

```

```
        digitalWrite (paljenje_fan, LOW); //ugasi žutu LED kada grijanje
radi
    }
    else // u suprotnom ugasi LED diodu
    {
        digitalWrite (paljenje,LOW); // ugasi grijanje – crvena LED
        digitalWrite (paljenje_fan, HIGH); //upali žutu LED kada grijanje ne
radi
    }

//-----
// namještanje pokazivača na stupac 0, linija 1
    lcd.setCursor (0,1);
// Ispis Zadano prije ispisa brojke zadane temperature
    lcd.print ("Zadano ");
// ispis nove zadane radne temperature
    lcd.print (settemp);
// ispis znaka C (Celzij)
    lcd.print ('C');
// Ispis nove zadane radne temperature na serijskom monitoru
    Serial.println(settemp);
// Spremi podatke o radnoj temperaturi ako je bilo promjena u EEPROM
memoriju
//Želiš zaspisati novu radnu u EEPROM?
    if (Echeck ==1){
// Ako su pritisnut tipke T1 i T2 istovremeno zapiši novu radnu
// temperaturu u EEPROM
    if (digitalRead(temp_vise)== HIGH && digitalRead(temp_nize)== HIGH){
        EEPROM.put (1,settemp);
        Echeck = 0;
    }
}

if(digitalRead(signal_svjetla) == HIGH)
{
    LCDsvjetlo();
}
Echeck =0;
delay (250); // kašnjenje 250 milisekundi
}
```

Nakon prevođenja programa i učitavanja u Arduino pločicu potrebno je izvršiti testiranje programa i utvrditi radi li prema zahtjevima projektnog zadatka.

4.2. APLIKACIJA ZA UPRAVLJANJE KORAČNIM MOTOROM

4.2.1. Projektni zadatak

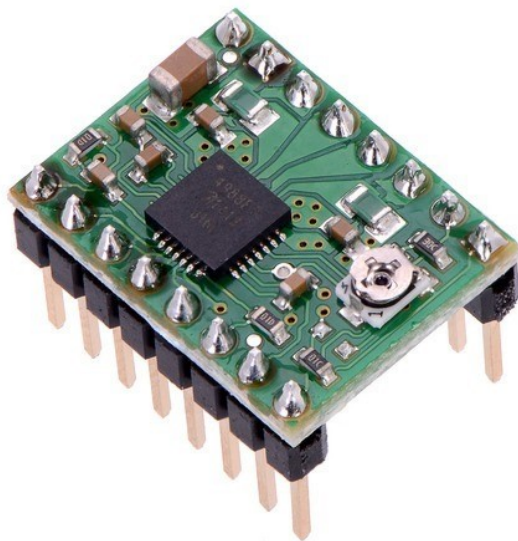
Izraditi aplikaciju za upravljanje koračnim motorom NEMA17 pomoću A4988 čipa. Smjer vrtnje motora određuje se pomoću odgovarajuće tipke, odnosno pritiskom na tipku za lijevo ili tipku za desno. Brzina vrtnje motora namješta se pomoću potenciometra.

4.2.2. Dizajn projekta

Za realizaciju projekta potrebne su sljedeće komponente:

- Arduino Uno R3 pločica
- NEMA17 koračni motor
- A4988 čipa za upravljanje motorom
- potenciometar 10 kW
- dva tipkala (prekidača)
- dva otpornika od 220 W
- žice za povezivanje
- razvojna pločica.

Slijedi opis komponenti koje još nisu korištene u ovom priručniku. Za upravljanje koračnim motorom koristit će se A4988 čip, slika 4-6. Ovaj čip namijenjen je za upravljanje bipolarnim koračnim motorima (A4988, 2023.).



Slika 4.6: A4988 čip za upravljanje koračnim motorom (Izvor: <https://www.pololu.com/product/1182>)

A4988 čip moguće je ugoditi za upravljanje pet različitih veličina pokreta, odnosno koraka prema sljedećim pravilima:

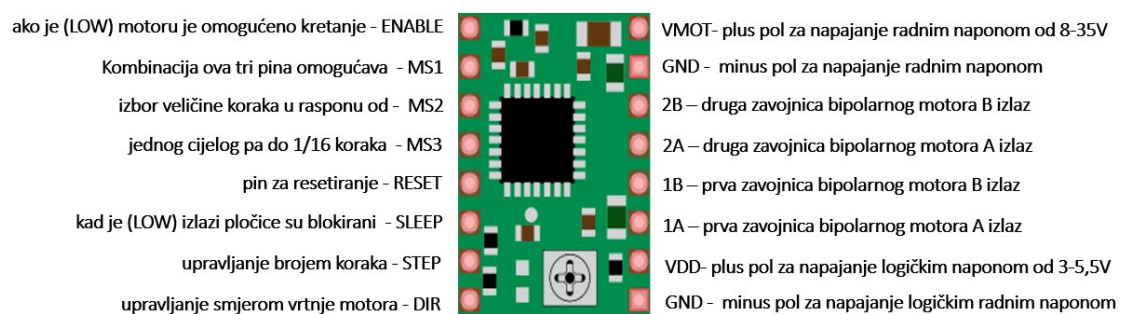
- **puni korak** – 200 koraka za jedan okret motora
- **pola koraka** – 400 koraka za jedan okret motora
- **četvrtina koraka** – 800 koraka za jedan okret motora
- **osmina koraka** – 1600 koraka za jedan okret motora
- **šesnaestina koraka** – 3200 koraka za jedan okret motora.

Upravljačka pločica A4988 koristi naponske razine od 3 V do 5,5 V, a najveći je iznos struje 2 A po jednoj fazi uz hlađenje pločice, dok najveći iznos struje iznosi 1 A bez hlađenja. Pločica se napaja naponom od 8 V do 35 V, tablica 4-1.

Tablica 41: Karakteristike A4988 čipa

Karakteristike A4988 čipa	Vrijednost
najmanji iznos logičke naponske razine	3 V
najveći iznos logičke naponske razine	5,5 V
stalna struja po fazi	1 A
najveći iznos struje po fazi (uz hlađenje)	2 A
najmanji iznos radnog napona	8 V
najveći iznos radnog napona	35 V

Na slici 4-7 dan je raspored priključaka A4988 upravljačke pločice.



Slika 4.7: Raspored priključaka A4988 upravljačke pločice (izvor: „Autorski rad“)

Koračni motor spaja se na priključke 1A, 1B, 2A i 2B na način da se jedna zavojnica koračnog motora spoji na 1A i 1B priključke, a druga zavojnica koračnog motora na 2A i 2B priključke. Arduino se spaja na *STEP* priključak za određivanje broja koraka i *DIR* priključak za odabir smjera vrtnje. Napajanje se spaja na priključke *VMOT* i *GND*. Preporučljivo je paralelno spojiti kondenzator radi zaštite A4988 pločice od vršnih napona. Priključke *SLEEP* i *RESET* potrebno je međusobno spojiti kako bi se omogućio rad pločice. U slučaju da se koristi priključak *SLEEP*, na njega se dovodi logička 0 i pločica se nalazi u stanju mirovanja te tako smanjuje potrošnju energije kada motor ne radi. Postavljanjem *RESET* priključka u stanje logičke 0, A4988 pločica postavlja se u početno definirano stanje. Direktnim spajanjem priključaka *RESET* i *SLEEP* dovodi se logička 1 i tako omogućava rad pločice. Priključak *ENABLE* služi za moguće isključivanje pločice i ako se postavi u stanje logičke 1, izlazi pločice bit će blokirani, a time je pločica isključena za upotrebu. Priključci MS1, MS2 i MS3 koriste se za odabir broja koraka motora. Ako se želi koristiti puni korak, tada ovi priključci ostaju odspojeni. U slučaju da se želi mijenjati veličina koraka, potrebno je na priključke spojiti logička stanja prema tablici 4-2.

Tablica 42: Postavljanje broja koraka motora

MS1	MS2	MS3	VELIČINA KORAKA	BROJ KORAKA U JEDNOM OKRETU
LOW	LOW	LOW	Puni korak	200
HIGH	LOW	LOW	1/2 koraka	400
LOW	HIGH	LOW	1/4 koraka	800
HIGH	HIGH	LOW	1/8 koraka	1600
HIGH	HIGH	HIGH	1/16 koraka	3200

Koračni motori u svojoj su osnovi pretvornici električne energije u odgovarajući mehanički pomak. Prilikom uzbude namotaja rotor se pokreće u željenom smjeru za predviđeni kut. Smjer vrtnje mijenja se promjenom impulsnog slijeda (okretanjem polariteta), a brzina rotacije promjenom frekvencije impulsa. Prijedni kut ovisi o broju koraka, odnosno o broju primljenih impulsa. Postoji nekoliko načina podjele koračnih motora. Jedan je od načina razlikovanja prema konstrukciji namotaja i napajanju. Na taj se način razlikuju bipolarni i unipolarni koračni motori. U ovom projektu koristit će se koračni motor prema *NEMA17* (*NEMA17 stepper motor, 2023.*) standardu kojega je donijelo američko udruženje proizvođača elektronike (engl. *National Electrical Manufacturers Association*). S tim standardom označen je promjer prednjeg kućišta motora, a broj 17 označava veličinu od 1,7" (engl. *Inch*), slika 4-8. Tehničke karakteristike *NEMA17* motora dane su u tablici 4-3.

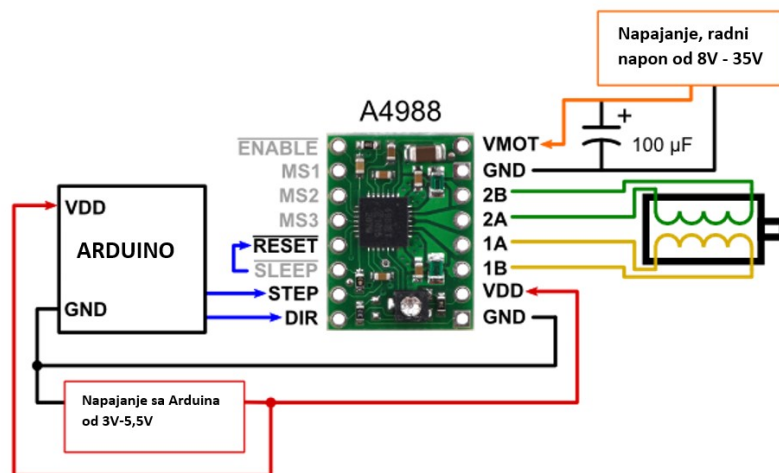


Slika 4.8: NEMA17 koračni motor (Izvor: <https://soldered.com/hr/proizvod/nema17-stepper-motor/>)

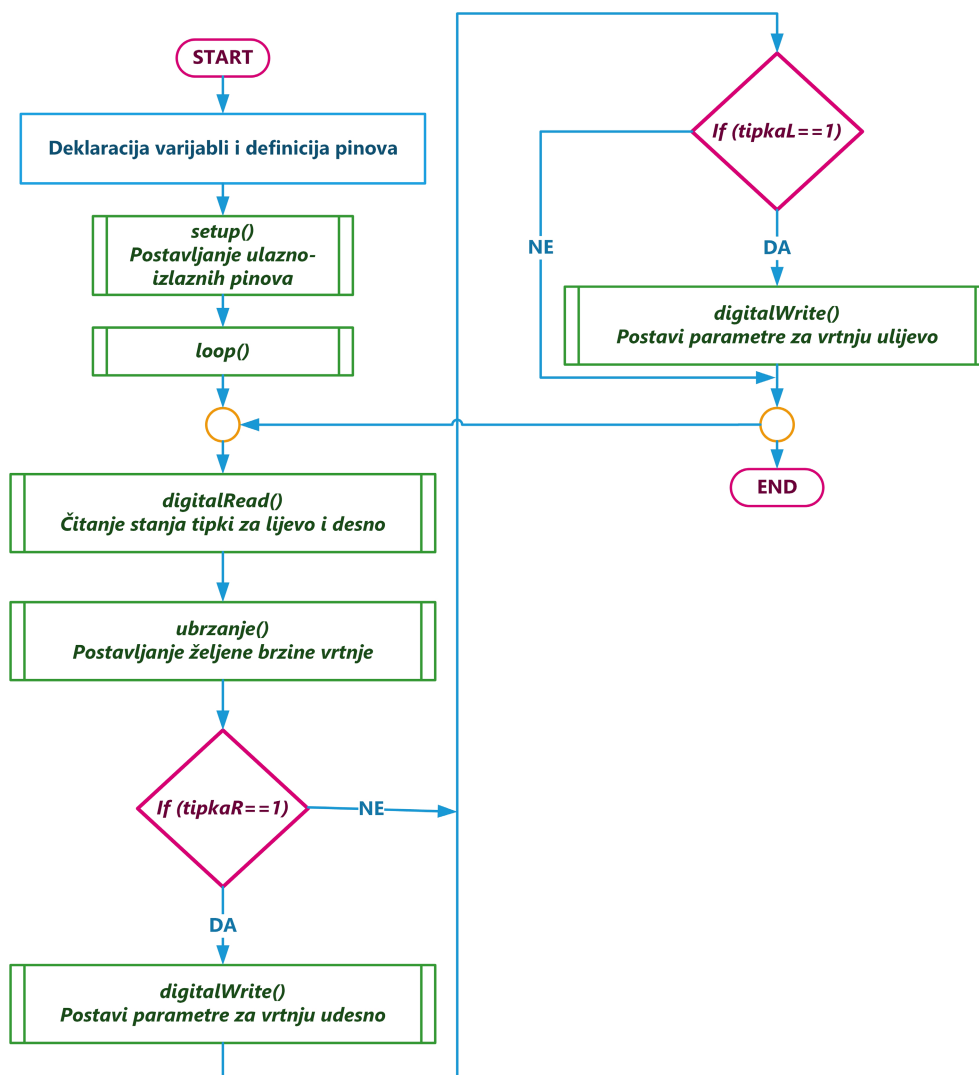
Tablica 43: Karakteristike NEMA17 koračnog motora

Model 17HS4417	Vrijednosti
kut koraka	1,8°
dužina motora	40 mm
broj žica	4
napon	2,55 V
struja	1,7 A
fazni otpor	1,5 Ω
fazna indukcija	2,8 mH
težina motora	288 g
moment	60 oz-in ili 0,423 Nm

Prilikom spajanja na vanjsko napajanje potrebno je paralelno spojiti kondenzator od najmanje 47 μF zbog zaštite A4988 upravljačke pločice od vršnih napona, a način povezivanja s Arduinoom i koračnim motorom prikazan je na slici 4-9. Dijagram toka programa za upravljanje motorom prikazan je na slici 4-10. Pored standardnih funkcija, koje se koriste u programu, potrebno je definirati korisničku funkciju *ubrzanje()*. Ova funkcija nema ulaznih parametara, već samo čita vrijednost potencijometra s analognog ulaza i prema tome namješta brzinu vrtnje motora koju vraća kao izlazni parametar.



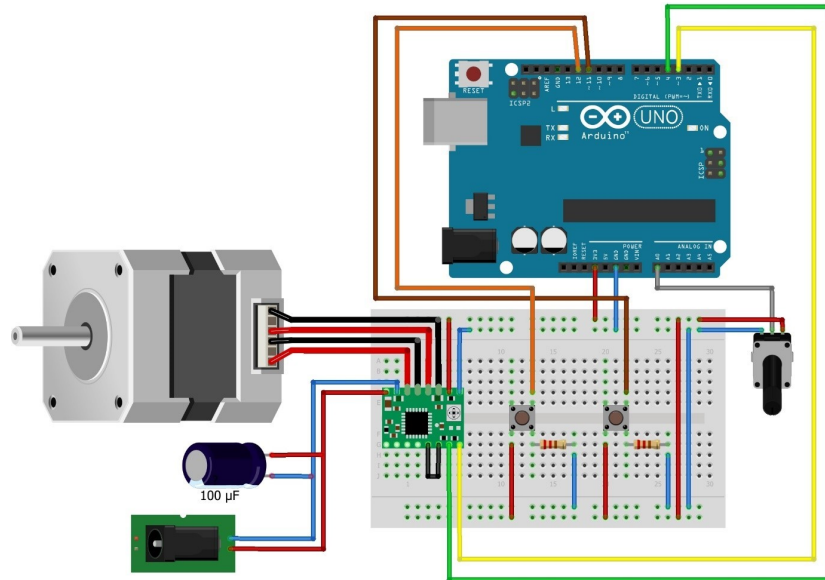
Slika 4.9: Povezivanje A4988 upravljačke pločice, Arduina i koračnog motora (izvor: „Autorski rad“)



Slika 4.10: Dijagram toka programa za upravljanje motorom (izvor: „Autorski rad“)

4.2.3. Aplikacija za upravljanje koračnim motorom

Na slici 4-11 prikazana je cjelovita električna shema povezivanja sustava za upravljanje koračnim motorom NEMA17.



Slika 4.11: Električna shema sustava za upravljanje koračnim motorom (izvor: „Autorski rad“)

U programskom kôdu 4-2 dana je aplikacija za upravljanje koračnim motorom prema projektnom zadatku. Nakon prevođenja programa i učitavanja u Arduino Uno R3 pločicu potrebno ga je testirati i vidjeti ponaša li se program u skladu s projektnim zadatkom.

Programski kôd 42: Aplikacija za upravljanje koračnim motorom

```

/*
Upravljanje koračnim motorom NEMA17 pomoću A4988 čipa
Pritiskom na lijevu tipku (tipka 1) motor se okreće u lijevu stranu,
pritiskom na desnu tipku (tipka 2) okreće se u desnu stranu.
Promjenom položaja potenciometra mijenja se brzina vrtnje motora.
Krug
* A4988 VCC na Arduino 3,3v, GND na Arduino GND
* A4988 STEP na Arduino pin 3, DIR na Arduino pin 4
* A4988 RESET I SLEEP spojiti međusobno.
* A4988 1A, 1B, 2A, 2B, spojiti koračni motor.
* A4988 VMOT i GND spojiti napajanje motora (8–35V) + kondenzator 100 µF
* Tipka 1, na pin 11 Arduino, Tipka 2, na pin 12 Arduino.
* Potenciometar signal na Arduino A0,
*/

```

```
//definicija varijabli i pinova
const int stepPin = 3;
const int dirPin = 4;
const int buttonPinL = 11;
const int buttonPinR = 12;
int buttonStateL = 0;
int buttonStateR = 0;
// parametri funkcije za ubrzanje
int delay_namjestanje, delay_mapiranje;

void setup() {
  // definiranje ulazno-izlaznih pinova
  pinMode(buttonPinL, INPUT);
  pinMode(buttonPinR, INPUT);

  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}

void loop() {
  // čitanje stanja tipki
  buttonStateL = digitalRead(buttonPinL);
  buttonStateR = digitalRead(buttonPinR);
  // Pridružuje delay_mapiranje varijabli vrijednosti dobivene iz funkcije
  ubrzanje()
  // izrađuje prilagođeno kašnjenje, ovisno o potencijometru, o kojem ovisi
  brzina motora.
  delay_mapiranje = ubrzanje();
  // ako je pritisnuta tipka za desno
  if(buttonStateR == HIGH){
    digitalWrite(dirPin, LOW);
    digitalWrite(stepPin, HIGH);
    delay(delay_mapiranje);
    digitalWrite(stepPin, LOW);
    delay(delay_mapiranje);
  }
  // ako je pritisnuta tipka za lijevo
  if(buttonStateL == HIGH){
    digitalWrite(dirPin, HIGH);
    digitalWrite(stepPin, HIGH);
    delay(delay_mapiranje);
    digitalWrite(stepPin, LOW);
    delay(delay_mapiranje);
  }
}
```

```
// Funkcija za namještanje brzine
int ubrzanje()
{
  // Čitanje vrijednosti potencijometra
  int delay_namjestanje = analogRead(A0);
  // Pretvaranje pročitanih vrijednosti s potencijometra od 0 do 1023
  // u željene vrijednosti delaya između 1 i 30
  int nova_vrijednost = map(delay_namjestanje, 0, 1023, 1,30);
  return nova_vrijednost;
}
```

Popis elemenata korištenih u sadržaju

Slika 1.1: Arduino Uno R3 (izvor: https://www.arduino.cc/)	11
Slika 1.2: Oznake portova	12
Slika 1.3: Ikona Arduino IDE (izvor: „Autorski rad“)	13
Slika 1.4: Osnovni Arduino program (izvor: „Autorski rad“)	13
Slika 1.5: Razvoj i izvođenje Arduino programa (izvor: „Autorski rad“)	14
Slika 1.6: Ispis na zaslonu računala (izvor: „Autorski rad“)	15
Slika 1.7: Ispis pauza na zaslonu računala (izvor: „Autorski rad“)	17
Slika 1.8: Korištenje Serial Monitora za unos i ispis (izvor: „Autorski rad“)	20
Slika 1.9: Ispis podataka iz EEPROM memorije (izvor: „Autorski rad“)	22
Slika 2.1: Primjer izvedbe i simbol fotootpornika (izvor: „Autorski rad“)	24
Slika 2.2: Shema spajanja svjetlosnog prekidača (izvor: „Autorski rad“)	24
Slika 2.3: Senzor HC-SR04 (izvor: „Autorski rad“)	26
Slika 2.4: Princip rada ultrazvučnog senzora (izvor: „Autorski rad“)	27
Slika 2.5: Shema spajanja senzora HC-SR04 (izvor: „Autorski rad“)	28
Slika 2.6: Senzor DS18B20 u kućištu TO-92 i njegova vodonepropusna izvedba (izvor: „Autorski rad“)	30
Slika 2.7: Povezivanje više DS18B20 senzora na istu sabirnicu (izvor: „Autorski rad“)	31
Slika 2.8: Vanjsko napajanje DS18B20 senzora (izvor: „Autorski rad“)	32
Slika 2.9: Parazitno napajanje DS18B20 senzora (izvor: „Autorski rad“)	32
Slika 2.10: Shema povezivanja senzora DS18B20 i Arduina (izvor: „Autorski rad“)	33
Slika 2.11: Uključivanje potrebnih biblioteka (izvor: „Autorski rad“)	33
Slika 2.12: Odabir OneWire.h i DallasTemperature.h biblioteka (izvor: „Autorski rad“)	34
Slika 2.13: Povezivanje Arduina i NTC termistora (izvor: „Autorski rad“)	36
Slika 2.14: Cjelovita shema povezivanja Arduina i 100 k NTC termistora (izvor: „Autorski rad“)	37
Slika 2.15: Ispis očitavanja NTC termistora (izvor: „Autorski rad“)	38
Slika 2.16: Senzor DHT22 (izvor: „Autorski rad“)	39
Slika 2.17: Povezivanje Arduina i senzora DHT22 (izvor: „Autorski rad“)	40
Slika 2.18: Potrebne biblioteke za rad sa senzorom DHT22 (izvor: „Autorski rad“)	40
Slika 2.19: Ispis podataka sa senzora DH22	41
Slika 2.20: Odabir funkcije za grafički prikaz podataka (izvor: „Autorski rad“)	42
Slika 2.21: Grafički prikaz podataka sa senzora DHT22 (izvor: „Autorski rad“)	42
Slika 2.22: Senzor BME280 (izvor: „Autorski rad“)	43
Slika 2.23: Biblioteke za podršku BME280 (izvor: „Autorski rad“)	44
Slika 2.24: I2C sabirnica (izvor: „Autorski rad“)	45
Slika 2.25: Povezivanje Arduina i BME280 putem I2C (izvor: „Autorski rad“)	47
Slika 2.26: Ispis podataka BME280 I2C	48
Slika 2.27: Povezivanje uređaja putem SPI komunikacije (izvor: „Autorski rad“)	49
Slika 2.28: Povezivanje senzora BME280 i Arduina putem SPI komunikacije	49
Slika 2.29: Shema povezivanja senzora BME280 i Arduino pločice putem SPI komunikacije (izvor: „Autorski rad“)	50
Slika 2.30: Ispis podataka sa senzora BME280 putem SPI komunikacije (izvor: „Autorski rad“)	51
Slika 2.31: Plutajući plovak (izvor: „Autorski rad“)	52

Slika 2.32: Shema povezivanja plutajućeg plovka i Arduina (izvor: „Autorski rad“)	53
Slika 3.0.1: Mehanički prekidač (izvor: „Autorski rad“)	56
Slika 3.2: Shema povezivanja mikrokontrolera, prekidača i LED diode (izvor: „Autorski rad“)	56
Slika 3.3: Električna shema tipkovnice (izvor: „Autorski rad“)	58
Slika 3.4: Instalacija biblioteke za podršku tipkovnice (izvor: „Autorski rad“)	59
Slika 3.5: Shema povezivanja tipkovnice i Arduino pločice (izvor: „Autorski rad“)	60
Slika 3.6: Ispis znakova unesenih preko tipkovnice (izvor: „Autorski rad“)	61
Slika 3.7: Shema povezivanja Arduina i potencijometra (izvor: „Autorski rad“)	63
Slika 3.8: Ispis na zaslon računala očitanih vrijednosti s potencijometra i AD pretvorba (izvor: „Autorski rad“)	64
Slika 3.9: Prikaz LCD 16 x 2 zaslona (izvor: „Autorski rad“)	65
Slika 3.10: Matrica piksela reda 5 x 8 (izvor: „Autorski rad“)	65
Slika 3.11: Raspored i značenje priključaka LCD zaslona (izvor: „Autorski rad“)	66
Slika 3.12: Instalacija biblioteke LiquidCrystal.h (izvor: „Autorski rad“)	67
Slika 3.13: Spajanje LCD zaslona s 4 podatkovna priključka (izvor: „Autorski rad“)	67
Slika 3.14: Spajanje LCD zaslona s 8 podatkovnih priključaka (izvor: „Autorski rad“)	68
Slika 3.15: Ispis posebnih znakova na LCD zaslon (izvor: „Autorski rad“)	71
Slika 3.16: LCD zaslon s I2C adapterom (izvor: „Autorski rad“)	72
Slika 3.17: Postavljanje adrese za Texas Instruments čip (izvor: „Autorski rad“)	72
Slika 3.18: Postavljanje adrese za NXP Semiconductors čip (izvor: „Autorski rad“)	73
Slika 3.19: Instalacija biblioteke LiquidCrystal_I2C.h (izvor: „Autorski rad“)	73
Slika 3.20: Shema povezivanja LCD I2C zaslona, tipkovnice i Arduina (izvor: „Autorski rad“)	74
Slika 3.21: Prikaz unesenih znakova na LCD I2C zaslonu (izvor: „Autorski rad“)	76
Slika 4.1: Povezivanje komponenti sustava termostata (izvor: „Autorski rad“)	79
Slika 4.2: Blok-dijagram programa Termostat.ino (izvor: „Autorski rad“)	79
Slika 4.3: Blok dijagram setup() funkcije (izvor: „Autorski rad“)	80
Slika 4.4: Blok-dijagram loop() funkcije (izvor: „Autorski rad“)	81
Slika 4.6: A4988 čip za upravljanje koračnim motorom (Izvor: https://www.pololu.com/product/1182)	86
Slika 4.7: Raspored priključaka A4988 upravljačke pločice (izvor: „Autorski rad“)	87
Slika 4.8: NEMA17 koračni motor (Izvor: https://soldered.com/hr/proizvod/nema17-stepper-motor/)	89
Slika 4.9: Povezivanje A4988 upravljačke pločice, Arduina i koračnog motora (izvor: „Autorski rad“)	90
Slika 4.10: Dijagram toka programa za upravljanje motorom (izvor: „Autorski rad“)	90
Slika 4.11: Električna shema sustava za upravljanje koračnim motorom (izvor: „Autorski rad“)	91

Tablica 11: Tehničke karakteristike Arduina Uno R3	11
Tablica 21: Ovisnost otpora, struje i napona o osvjetljenju i izabranom otporu	25
Tablica 22: Karakteristike senzora HC-SR04	26
Tablica 23: Karakteristike DS18B20 senzora	30
Tablica 24: Raspored priključaka DS18B20 senzora	31
Tablica 25: Povezivanje senzora BME280 i ESP32 pločice putem I2C komunikacije	45
Tablica 26: Tehničke karakteristike plutajućeg plovka	52
Tablica 31: Tehnička specifikacija LCD zaslona	65
Tablica 41: Karakteristike A4988 čipa	87
Tablica 42: Postavljanje broja koraka motora	88
Tablica 43: Karakteristike NEMA17 koračnog motora	89
Jednadžba 21: Izračun udaljenosti objekta	27
Jednadžba 22: Ulazni napon	36
Jednadžba 23: Računanje otpora termistora	36
Jednadžba 24: Steinhart–Hartova jednadžba	37
Jednadžba 31: Računanje napona na AD pretvorniku	62

Popis literature

- 1 A4988, Stepper Motor Driver Carrier (2023).
<https://www.pololu.com/product/1182> (pristupljeno: 7. 9. 2023.).
- 2 Aosong Electronics Co.,Ltd, DHT22 (2023).
<https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf>
(pristupljeno: 15. 5. 2023.).
- 3 Arduino (2023).
<https://www.arduino.cc/> (pristupljeno: 15. 5. 2023.).
- 4 Bosch, Humidity sensor BME280 (2023).
<https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/>
(pristupljeno: 7. 9. 2023.).
- 5 Grusin, M. (2023). Serial Peripheral Interface (SPI).
<https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all> (pristupljeno: 10. 5. 2023.).
- 6 NEMA17 stepper motor (2023).
<https://soldered.com/hr/proizvod/nema17-stepper-motor/> (pristupljeno: 11. 10. 2023.).
- 7 Steinhart-Hart equation (2023).
https://en.wikipedia.org/wiki/Steinhart%E2%80%93Hart_equation (pristupljeno: 9. 8. 2023.).
- 8 UM10204, I²C-bus specification and user manual (2021).
<https://www.nxp.com/docs/en/user-guide/UM10204.pdf> (pristupljeno: 10. 5. 2023.).



EduSplit Obrtna tehnička škola
Regionalni centar kompetentnosti Split



Projekt je sufinancirala Europska unija iz Europskog socijalnog fonda.

Za više informacija o EU fondovima molimo pogledajte web-stranicu
Ministarstva regionalnoga razvoja i fondova Europske unije.
www.strukturnifondovi.hr