

POROČILO »ZOPp«

Simon LIPOVŠEK

Razred: E3.C

Šolsko leto 2021/2022

Datum: 12.10.2021

Kazalo:

[1. Blok shema 3](#_Toc2035743466)

[2. Delovanje 3](#_Toc269920423)

[3. Simulacija 5](#_Toc1793698164)

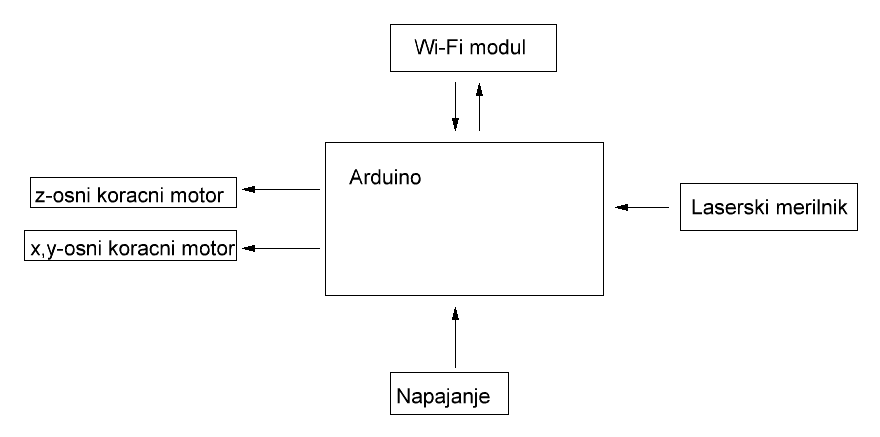
[4. Vezalna shema 5](#_Toc1565402601)

[5. Kosovnica 6](#_Toc1075627297)

[6. Kritični material 6](#_Toc1972370893)

[7. Komentar 7](#_Toc2036379909)

# Blok shema



# Delovanje

Delovanje se začne ko poženemo python program na računalniku. Python program pošlje signal preko interneta na Wi-Fi modul. Informacije za začetek merjenja prebere Arduino in začne meriti. Najprej postavi senzor in koračne motorje na začetno pozicijo. Laserski merilnik tako 5-krat izmeri razdaljo saj se včasih lahko zgodijo napake in s tem pridobimo bolj natančen rezultat. Podatke o zmerjenih razdaljah in kordinatnih pozicijah senzorjev arduino pošlje preko Wi-Fi modula računalniku. Arduino že med pošiljanjem s pomočjo horizontalnega koračnega motorja zavrti podlago za 1°. Merjenje razdalje inpo pošiljanje le te skupaj s koordinatam pošilja dokler ne naredi popolni krog. Takrat se senzor premakne za 1mm na z osi in vse se spet ponovi. Računalnik s pomočjo kotnih funkcij izračuna natančne koordinate in razdalje od središča plošče. Pri končanem merjenju Arduino pošlje informacijo o zaključki in to sproži pretvarjanje in shranjevanje zbranih točk v 3D model.

1. **ARDUINO PLOŠČA**

Arduino je odprto-kodni programski jezik z izvorom v programskem jeziku C. Arduino plošča je naprava na kateri lahko programiraš čip ATMEGA32, lahko pa uporabljaš celotno ploščo ter njene izhodne pine za povezovanje z zunanjimi komponentami. Plošča vsebuje tudi flash pomnilnik in komunikacijski procesor za povezavo z računalnikom. Plošča nam omogoča programiranje čipa z USB konektor. Preko USB vmesnika lahko Arduino ploščo napajamo, programiramo in z njo komuniciramo. Arduino plošča podpira analogne , digitalne vhodne/izhodne napetosti in PWM. Analogni signali so shranjeni kot 10 bitne informacije in obratno ima izhodna analogna zapis 1024 stanj. Omogoča tudi različne komunikacijske protokole z napravami, kot sta: \_\_, I2C.

**Uporaba v projektu**

Arduino plošča bo zbirala podatke iz različnih komponent in podajala ukaze drugim napravam. Podatke bo pošiljala na računalnik preko wi-fi modula

1. Wi-Fi MODUL

Modul uporablja 2,4 GHz frekvenčni pas. Količina prenosa in nalaganja se giblje med 250 kb/s in 2Mb/s. Modul lahko uporabimo kot zasebno omrežje. Ta omrežje je ločen od omrežja in so vsi podatki poslani glavnemu sprejemniku. Druga možnost je pa da ga povežemo na internet. Preko interneta se modul lahko obnaša kot sprejemnik in oddajnik. Modulu se na internetu določi IP naslov preko katerega se naprave lahko povezujejo nanj.

**Uporaba v projektu**

Modul je uporabljen kot komunikator med Arduinom in računalnikom. Arduino prejme signale iz računalnika za začetek skeniranja. Pošlje pa informacije laserskega senzorja saj je računanje 3D modela zanj prezahtevno in je le to opravljeno v računalniku in sicer s pomočjo Python programa.

1. LASERSKI MERILNIK

Laserski senzor s pomočjo laserja izmeri oddaljenost predmeta. Oddaljenost izračuna z časovno razliko oddanega signala in sprejetja odboja tega signala.

**Uporaba v projektu**

Naloga laserskega merilnika je da izmeri oddaljenost od senzorja do predmeta položenega na vrteči plošči. Meritev se opravi na vsake 2mm zunanjega obsega vrteče podlage.

1. KORAČNI MOTOR

Koračni motor je elektromehanski del, ki pretvori električni signal v mehansko rotacijo grede. Vsak pomik osi za točno določen kot se izvede ob električnem pulzu. Koračni motor torej pretvarja električne pulze v rotacijo njegove grede.

**Uporaba v projektu**

V projektu uporabljam dva koračna motorja. Prvi koračni motor vrti podlago in spreminja x in y pozicijo objekta. Drugi koračni motor pa premika Z pozicijo laserskega merilnika. Poleg razdalje zmerjene z laserskim merilnikom se pošljejo tudi x, y, z koordinate te meritve.

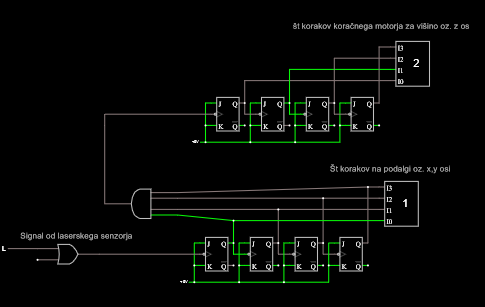
1. NAPAJANJE

Za napajanje uporabljamo napajalni kabel in adapter. Napajalni adapter je vključen v vtičnico. Znotraj adapterja je električno vezje, ki pretvori 230V izmenično napetost v 12V enosmerno napetost.

**Uporaba v projektu**

Za napajanje je uporabljen 12V kabel. S tim direktno napajamo motorje in regulator napetosti. Regulator napetosti zniža 12V v 5V in s tem napaja Arduino ploščo in druge 5V module.

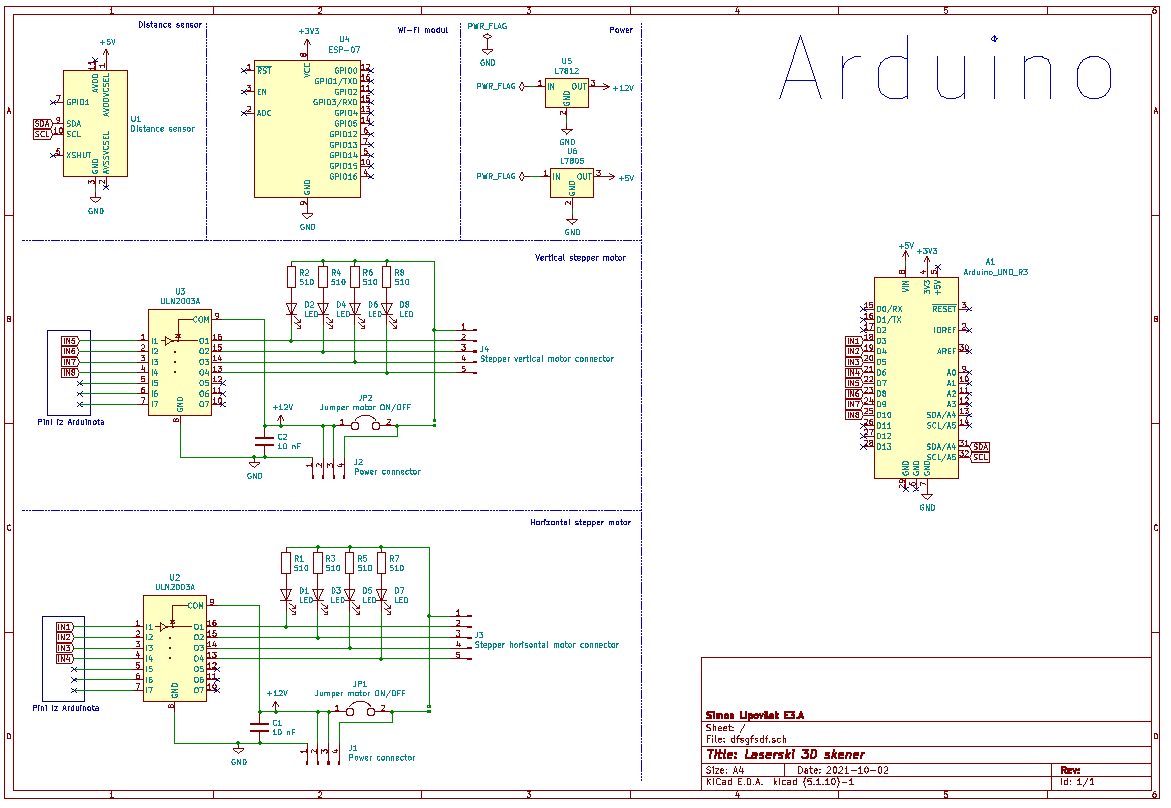
# Simulacija



Simulacija je bila izvedena s programom Circuit simulator. Delujoče vezje je na linku: <https://tinyurl.com/yg5t72ye>.

Simulacija prikazuje delovanje servomotorjev znotraj programa. Na levi strani simulacije lahko vidimo ali vrata s katerimi sporočimo vezju da je merilnik razdalje zaključil merjenje in čaka da spremeni pozicijo. Signal merilnika razdalje premakne koračni motor na x,y osi. Kot s katerim je koračni motor premaknjen na x,y osi je prikazan s številko na spodnjem desnem zaslonu. Ko je x,y osni koračni motor naredil cel krog, pošlje signal vertikalnemu koračnemu motorju. Ta resetira zaslon na x,y koračnem motorju in izpiše svoj zavrten kot. Kot s katerim se je vertikalni motor zavrtel je prikazan na zgornjem desnem zaslonu.

# Vezalna shema



Koračna motorja in laserski merilnik ne bodo neposredno pritrjene na vezje ampak bodo pri spajkani preko pinov na vezje.

# PCB

Pri vezavi so bila podnožja nekaterih komponent zamenjana z podnožjem njim najbolj podobnim. sss

**Primer vezave v KiCad programu:**

Vezave prvega sloja: Pogled vezav drugega sloja:

Diagram

Description automatically generatedDiagram, schematic

Description automatically generated

**3D predstava vezja v KiCad programu**

3d pogled z ptičje perspektive: 3D pogled iz hrbtne strani:

Diagram, schematic

Description automatically generatedDiagram

Description automatically generated

# Kosovnica

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Element** | **Vrednost** | **Raster** | **Kos** | **Opomba** |
| **Upor** | 510 | 7,5 | 8 |  |
| **LED** |  |  | 8 |  |
| **ULN2003A** |  |  | 2 |  |
| **Jumper** |  |  | 2 |  |
| **Napajalni konektorji** |  |  | 70 | Uporaba za pripenjanje arduina in drugih odmaknjenih komponet |
| **Kondenzator** | 10 nF |  | 2 |  |
| **L7812** | 12V |  | 1 |  |
| **L7805** | 5V |  | 1 |  |
| **Wi-Fi modul** |  |  | 1 |  |
| **Laserski senzor** |  |  | 1 |  |
| **Arduino uno** |  |  | 1 |  |

# Kritični material

Za projekt je potrebno imeti: laserski merilnik, 2 koračna motorja, Arduino. Za upravljanje motorjev je potrebni uravnalnik napetosti saj je potrebno dvigniti na 12V.

# Koda

Koda se deli na dva dela, prvi del se izvede v arduinu med tem ko drugi del pa na računalniku. S tem prihranimo na času pri samem skeniranju in pridobimo lepše in bolj elegantne predstave podatkov oz. objekta.

1. **Koda za Arduino**

**#include <Arduino.h> //Povemo prgramu da naj to kompila za arduiono če kompilamo v arduino IDE jo je potrebno izbrisati**

**#include <Wire.h> //Uvozimo knjižnico, ki vsebuje navodila za komunikacijo preko I2C-ja**

**#include <Adafruit\_VL6180X.h> //Uvozimo knjižnico, ki vsebuje navodila za dešifracijo podatkov pridobljenih iz senzorja**

**#include <Stepper.h> //Uvozimo knjižnico, ki vsebuje funkcijo za pogovor z koračnim motorjem**

**int numberofstep = 48\*64; //Definirmo število korakov, ki jih lahko koračni motor naredi**

**Stepper vertikalniMotor(numberofstep,5,7,6,4); //Definiramo motor ter pine na katere je priključen**

**Stepper krozniMotor(numberofstep,9,11,10,8); //Definiramo drug motor ter pine na katere je definiran**

**Adafruit\_VL6180X vl = Adafruit\_VL6180X(); //Odpremo knjižnico za dešifriranje podatkov iz merilnika razdalj**

**int y = 0; //Nastavimo pozicijo krožnega motorja**

**int z = 1; //Nastavimo pozicijo vertikalnega motorja**

**void setup() { //Začetne nastavitve, ki se zaženejo samo enkrat**

**Serial.begin(115200); //Aktiviramo serijsko povezo**

**while (!Serial) { //Počakamo in pogledamo če je komunikacija z računalnikom vzpostavljena**

**delay(1); //Počakamo stotinko**

**}**

**Serial.println("Adafruit VL6180x test!"); //Preko serijskega porta povemo, da bomo testirali senzor**

**if (! vl.begin()) { //Tvorimo if stavek z rezultatom o odvisnosti senzorja za razdalje**

**Serial.println("Failed to find sensor"); //Če nismo uspeli najti senzorja za razdalje in nam to pove**

**while (1); // Ko funkcija drži jo preskočimo**

**}**

**Serial.println("Sensor found!"); //Če smo prišli do tega dela kode nam pove da smo našli senzor in da lahko nadaljujemo**

**Serial.println("dodatna vrstica"); //To je pomožna vrstica za usklajevanje z Python programu**

**krozniMotor.setSpeed(9); //Nastavitev hitrosti krožnega motorja**

**vertikalniMotor.setSpeed(9); //Nastavitev hitrosti verikalnega motorja**

**}**

**void loop() { //Začetek glavne zanke**

**for(z=0;z<2048;z++){ //For zanka ki premakne potor iz zčetne pozicije do najvišje pozicije na vertikalni osi**

**vertikalniMotor.step(1); //Funkcija, ki premakne motor za prej nastavljeno eno stopnjo**

**for(y=0;y<2048;y++){ //For zanka, ki zavrti kororačni motor za cel krog**

**krozniMotor.step(1); //Funkcija katera zavrti krožni motor za prej določeno enoto**

**z = z; //Z spremenljivki dodelimo novo vrednos**

**Serial.print("visina : "), Serial.print(z); //Izpišemom višino preko serijskega izhoda**

**Serial.print(" ,vrtljaj : "),Serial.print(y); //Izpišemo vrtljaj preko serijskega izhoda**

**/\* konec motorčka za y os**

**\*/**

**uint8\_t range = vl.readRange(); //Preberemo razdaljo iz senzorja, če lahko, če dobi nič izpusti**

**uint8\_t status = vl.readRangeStatus(); //Prebememo status senzorja (Če je znotraj kateri error)**

**if (status == VL6180X\_ERROR\_NONE) { //Pogledamo če senzor je senzor dobil napake**

**Serial.print(", Range: "); Serial.println(range); //Na serijska vrate so izpissani podatki meritev**

**}**

**// Če se zgodi napaka, jo izpiše na uporabniku prijazen način!**

**if ((status >= VL6180X\_ERROR\_SYSERR\_1) && (status <= VL6180X\_ERROR\_SYSERR\_5)) { //če je bil kakšen error bomo pregledal če je ta specifična**

**Serial.println("System error"); //Če je bila ta napaka nam pove da je nekaj narobe s programom**

**}**

**else if (status == VL6180X\_ERROR\_ECEFAIL) { //Pogleda če je ta specifična napaka zaznana**

**Serial.println("ECE failure"); //nam pove da senzor ne dela**

**}**

**else if (status == VL6180X\_ERROR\_NOCONVERGE) { //Pregleda če se pojavi ta specifična napaka**

**Serial.println("No convergence"); //Nam pove da ne najde programa za dešifriranje podatkov**

**}**

**else if (status == VL6180X\_ERROR\_RANGEIGNORE) { //Pregleda če se pojavi ta specifična napaka**

**Serial.println("Ignoring range"); //Pove da ni ničesar pred senzorjem pred senzorjem oz. da je preveč oddaljeno**

**}**

**else if (status == VL6180X\_ERROR\_SNR) { //Pregleda če se pojavi ta specifična napaka**

**Serial.println("Signal/Noise error"); //Pove da je v podatkih preveč šuma in da ne more razbrati le teh**

**}**

**else if (status == VL6180X\_ERROR\_RAWUFLOW) { //Pregleda če se pojavi ta specifična napaka**

**Serial.println("Raw reading underflow"); //Laser se ne odbije nazaj v senzor**

**}**

**else if (status == VL6180X\_ERROR\_RAWOFLOW) { //Pregleda če se pojavi ta specifična napaka**

**Serial.println("Raw reading overflow"); //Preveč laserja se odbije nazaj**

**}**

**else if (status == VL6180X\_ERROR\_RANGEUFLOW) { //Pregleda če se pojavi ta specifična napaka**

**Serial.println("Range reading underflow"); //Podatek ki ga je hotel laser pretvoriti v številko je premejhen da bi ga poslal**

**}**

**else if (status == VL6180X\_ERROR\_RANGEOFLOW) { //Pregleda če se pojavi ta specifična napaka**

**Serial.println("Range reading overflow"); //Podatke, ki ga je hotel program poslati kot številko je prevelik da bi ga lahko poslal (Npr laser rabi predolgo časa da se odbije nazaj (Amplituda laserja je zanesljiva ampak čas je prevelik))**

**}**

**delay(20); //Počakamo 20 stotink če hočemo**

**}**

**}**

**}**

1. **Koda za Python program**

import math #Uvozimo modul z matematičnimi funkcijami

import serial #Uvozimo modul, ki nam omogoča branje serijskega vmesnika na računalniku

import matplotlib.pyplot as plt #Modul, ki nam prikaže podatke

ser = serial.Serial('COM3', 9600) #Initilaiziramo komunikacijskiu kanal na com portu

i = 0 #Neka spremenljivka

coordinates = [] #Pripraimo prostor za shranjevanje kordinat

X = [] #Pripravimo prostor za sharanjevanje kordinat x osi

Y = [] #Pripravimo prostor za sharanjevanje kordinat y osi

Z = [] #Pripravimo prostor za sharanjevanje kordinat z osi

obrat = 200 #Definiram velikost obrata na motorju

def plotting(x, y, z): #Funkcija ki vzame podatke ter sestavi 3d model

fig = plt.figure() #Pripravimo vrst 3d prikazovalnika

ax = plt.axes(projection="3d") #Povemo da bomo uporabili

ax.plot3D(x, y, z) #Povemo na kero os plottamo katere podatke

ax.set\_xlabel('X Axes') #Poimenujemo osi

ax.set\_ylabel('Y Axes') #Poimenujemo osi

ax.set\_zlabel('Z Axes') #Poimenujemo osi

plt.show() #prikažemo graf

while (i < 1 \* obrat): #Ko je i manjše kot število obratov

try: # Če je vse vredu poizkusimo

bingo = str(ser.readline()) # preberemo vrstico, ki je bla poslano na sersijki vhod

board = bingo.split(':') #Ločimo vrstice glede na dvopičja

measured\_height = board[1].split(',') #Loči besedilo glede na vejice

measured\_height = int(measured\_height[0]) #Iz številke, ki je zapisana kot beseda jo spremenimo v številka

measured\_spin = board[2].split(',') #Loči besedilo glede na vejice

measured\_spin = measured\_spin[0] #Iz številke, ki je zapisana kot beseda jo spremenimo v številka

measured\_distance = board[3].split(",") #Loči besedilo glede na vejice

measured\_distance = measured\_distance[0] #iz številke, ki je zapisana kot beseda jo spremenimo v številka

calculated\_angle = int(measured\_spin) \* (360 / 2048) #Izračunamo kot za katerega se motor zavrti

calculated\_distance = 10 - int(measured\_distance) #Pretvori v pravilne enote!!!!!

calculated\_width = math.sin(calculated\_angle) \* calculated\_distance #izračunamo x kordinante

X.append(calculated\_distance) #Dodamo x kordinato v prej narejeno tabelo

Y.append(calculated\_width) #Dodamo y kordinato v prej narejeno tabelo

Z.append(measured\_height) #Dodamo z kordinato v prej narejeno tabelo

coordinates.extend([calculated\_distance, calculated\_width, measured\_height]) #Premaknemo kordinate v spremewnljivko kordinta

print(coordinates) #Izpišemo kordinate

i = i + 1 #povečamo i (i predstavlja število meritev, ki jih bo senzor naredil)

print(i) #Izpišemo na kateri meritvi smo

except: #Ko prekinemo zanko oz. zanki zmanjka podatkov

plotting(X, Y, Z) #Aktiviramo funkcijo za prikazovanje pik v 3d kordinatnem objektu

break #Prekinemo in končamo

# Komentar

Predvidevam da nekatere naprave ter njihov način povezovanja mogoče ne bo uresničen po začetnem pričakovanju tega dokumenta.