Detekcija olujnog nevremena



Milena Đukić 40/2020 Mentor: Aleksandar Peulić

Kristina Stojković 83/2020

Pavle Oprić 69/2020

05.07.2024. Kragujevac. Prirodno-matematički fakultet

Sadržaj

[Koraci u prikupljanju I obradi podataka 3](#_Toc171082767)

[1. Uvod u Radar Meteoreologiju 3](#_Toc171082768)

[2. Prikupljanje Podataka 3](#_Toc171082769)

[3. Obrada Signala 3](#_Toc171082770)

[Reflektivnost (Z) i Intenzitet Padavina (R) 4](#_Toc171082771)

[1. Reflektivnost (Z) 4](#_Toc171082772)

[2. Z-R Odnos 4](#_Toc171082773)

[3. Računanje Intenziteta Padavina 4](#_Toc171082774)

[4. Intenzitet Padavina 4](#_Toc171082775)

[Detekcija olujnog nevremena u python-u 5](#_Toc171082776)

[Inicijalizacija okruzenja 5](#_Toc171082777)

[Instalacija pysteps pomoću conda/mamba paket menadžera(preporučeno) 5](#_Toc171082778)

[Instalacija pysteps sa mamba paket menadžerom u novom okruženju 5](#_Toc171082779)

[Biblioteke 6](#_Toc171082780)

[1. pysteps 6](#_Toc171082781)

[2. matplotlib 6](#_Toc171082782)

[3. serial 6](#_Toc171082783)

[4. datetime 6](#_Toc171082784)

[5. numpy 7](#_Toc171082785)

[6. time 7](#_Toc171082786)

[Proces Detekcije Nevremena 8](#_Toc171082787)

[1. Učitavanje Podataka 8](#_Toc171082788)

[2. Konverzija Reflektivnosti u Intenzitet Padavina 8](#_Toc171082789)

[3. Detekcija Oluje 8](#_Toc171082790)

[4. Vizualizacija 8](#_Toc171082791)

[5.Slanje Upozorenja 9](#_Toc171082792)

[Simulacija serijske komunikacije između STM32 mikrokontrolera i Python aplikacije 10](#_Toc171082793)

[1. Korišćenje Virtual Serial Port Driver za simulaciju komunikacije 10](#_Toc171082794)

[2. Python biblioteka za serijsku komunikaciju 10](#_Toc171082795)

[3. STM32 mikrokontroler i UART1 konfiguracija 10](#_Toc171082796)

[4. Funkcionalnost Aktiviranja Alarma 10](#_Toc171082797)

[5. Konfiguracije na STM32 11](#_Toc171082798)

[Proteus šema 13](#_Toc171082799)

[Demo projekta 16](#_Toc171082800)

# Koraci u prikupljanju I obradi podataka

## 1. Uvod u Radar Meteoreologiju

Radar meteorologija koristi radare za detekciju i praćenje padavina. Radarski sistemi emituju elektromagnetne talase koji se odbijaju od padavina (kiša, sneg, grad) i vraćaju nazad ka radaru. Vreme potrebno za povratak talasa i jačina povratnog signala omogućavaju određivanje položaja i intenziteta padavina.

## 2. Prikupljanje Podataka

Radar emituje impulze elektromagnetnog zračenja u atmosferu. Kada ti impulzi naiđu na padavine, deo energije se rasprši nazad ka radaru. Ovaj povratni signal se detektuje i koristi za kreiranje slike reflektivnosti.

## 3. Obrada Signala

Povratni signal se pretvara u digitalne podatke koji se dalje obrađuju kako bi se dobile informacije o intenzitetu padavina. Proces obrade uključuje nekoliko koraka:

* **Filtriranje šuma**: Uklanjanje nepoželjnih signala koji nisu vezani za padavine.
* **Kalibracija**: Podešavanje podataka kako bi se dobile tačne vrednosti reflektivnosti.
* **Konverzija**: Pretvaranje digitalnih signala u vrednosti reflektivnosti (dBZ).

Primeri snimaka radara:

A white circle with pink spots

Description automatically generated A white circle with pink spots

Description automatically generated A white circle with pink spots

Description automatically generated

# Reflektivnost (Z) i Intenzitet Padavina (R)

## 1. Reflektivnost (Z)

Reflektivnost (Z) je mera jačine povratnog radarskog signala. Izražava se u decibelima (dBZ) i predstavlja logaritamski odnos između detektovane jačine signala i referentne jačine. Veća vrednost dBZ ukazuje na jače padavine.

## 2. Z-R Odnos

Intenzitet padavina (R) se može izračunati iz reflektivnosti pomoću empirijskog Z-R odnosa. Ovaj odnos povezuje reflektivnost sa intenzitetom padavina na sledeći način:

R=a\*(Z)^b

gde su:

* R - Intenzitet padavina (mm/h)
* Z - Reflektivnost (mm^6/m^3)
* a i b - Empirijski konstante koje zavise od vrste padavina

U meteorološkoj praksi, vrednosti za a i b se obično određuju za specifične vrste padavina. Na primer, za kišu se često koriste vrednosti a=200 i b=1.6.

## 3. Računanje Intenziteta Padavina

Proces računanja intenziteta padavina iz reflektivnosti uključuje sledeće korake:

* **Prikupljanje podataka o reflektivnosti**: Radar prikuplja povratne signale i pretvara ih u vrednosti reflektivnosti.
* **Primena Z-R odnosa**: Vrednosti reflektivnosti se koriste u Z-R formuli za izračunavanje intenziteta padavina.

## 4. Intenzitet Padavina

Intenzitet padavina (R) predstavlja brzinu padavina, tj. količinu padavina koja pada na određenu površinu tokom određenog vremena. Izražava se u milimetrima po satu (mm/h). Veći intenzitet padavina ukazuje na jače padavine.

# Detekcija olujnog nevremena u python-u

## Inicijalizacija okruzenja

### Instalacija pysteps pomoću conda/mamba paket menadžera(preporučeno)

Conda je open-source sistem za upravljanje paketima i okruženjima koji radi na Windows, macOS i Linux operativnim sistemima. Mamba je zamena za conda koja nudi bolje performanse i pouzdanija rešenja za upravljanje okruženjima. Mamba brzo instalira, pokreće i ažurira pakete i njihove zavisnosti. Takođe omogućava lako kreiranje, čuvanje, učitavanje ili prebacivanje između različitih okruženja na vašem lokalnom računaru.

Od verzije 1.0, pysteps je dostupan na conda-forge, zajednički vođenom repozitorijumu paketa za conda pakete.

### Instalacija pysteps sa mamba paket menadžerom u novom okruženju

1. **Kreiranje novog okruženja sa mamba**

Prvo, pokrenite terminal (ili Command Prompt na Windows-u) i unesite sledeće komande za kreiranje i aktiviranje novog Python okruženja nazvanog ‘pysteps’ koristeći Python verziju 3.10:

mamba create -n pysteps python=3.10

mamba activate pysteps

Ova komanda će kreirati i aktivirati novo Python okruženje nazvano ‘pysteps’ koje koristi Python verziju 3.10.

1. **Dodavanje conda-forge kanala**

Sledeći korak je dodavanje conda-forge kanala gde se nalazi pysteps paket. U terminalu unesite sledeću komandu:

conda config --env --prepend channels conda-forge

Ova komanda dodaje conda-forge kanal kao izvor paketa za trenutno aktivno okruženje.

1. **Postavljanje prioriteta kanala**

Da biste postavili conda-forge kanal kao prioritetni, unesite sledeću komandu:

conda config --env --set channel\_priority strict

Ovaj korak nije strogo neophodan, ali se preporučuje jer conda-forge i default conda kanali nisu 100% kompatibilni. Postavljanjem conda-forge kao prioritetnog kanala osiguravate da će se paketi preuzimati sa ovog kanala.

1. **Instalacija pysteps paketa**

Na kraju, da biste instalirali pysteps i sve njegove zavisnosti, unesite sledeću komandu:

mamba install pysteps

Ova komanda će preuzeti i instalirati pysteps paket zajedno sa svim zavisnostima koje su potrebne za njegovo funkcionisanje.

## Biblioteke

### 1. pysteps

pysteps je Python biblioteka namenjena za stohastičku prognozu padavina na osnovu vremenskih serija radara reflektivnosti. Biblioteka nudi alate za unos i obradu radar podataka, detekciju oluja, praćenje i prognozu padavina.

* **Modul io**: Koristi se za unos i obradu podataka.
* **Modul feature**: Pruža funkcionalnosti za detekciju karakteristika, kao što su olujne ćelije.
* **Modul tracking**: Koristi se za praćenje oluja kroz vremenske serije podataka.
* **Modul utils**: Sadrži korisne funkcije, uključujući konverziju reflektivnosti u intenzitet padavina.
* **Modul visualization**: Omogućava vizualizaciju padavina i praćenje oluja.

### 2. matplotlib

matplotlib je popularna Python biblioteka za izradu grafikona. U ovom projektu koristi se za vizualizaciju podataka o padavinama i olujama.

### 3. serial

serial biblioteka omogućava komunikaciju sa serijskim portovima. Koristi se za slanje signala upozorenja putem serijskog porta.

### 4. datetime

datetime modul pruža klase za manipulaciju datumima i vremenom. Koristi se za upravljanje vremenskim oznakama u podacima.

### 5. numpy

numpy je fundamentalna Python biblioteka za naučne proračune i obradu velikih multidimenzionalnih nizova i matrica. Koristi se za manipulaciju i analizu radarskih podataka.

### 6. time

time modul pruža funkcije za rad sa vremenom. U ovom projektu koristi se za pauze između slanja serijskih signala kako bi se osiguralo pravilno vreme čekanja.

**Sve navedene biblioteke je potrebno instalirati(ukoliko nedostaju).**

## Proces Detekcije Nevremena

### 1. Učitavanje Podataka

Prvi korak u procesu je učitavanje radar podataka. Radar podaci se obično čuvaju u arhivama i sadrže informacije o reflektivnosti padavina u različitim vremenskim trenucima.

date = datetime.strptime("201607112100", "%Y%m%d%H%M")

data\_source = rcparams.data\_sources["mch"]

root\_path = data\_source["root\_path"]

path\_fmt = data\_source["path\_fmt"]

fn\_pattern = data\_source["fn\_pattern"]

fn\_ext = data\_source["fn\_ext"]

importer\_name = data\_source["importer"]

importer\_kwargs = data\_source["importer\_kwargs"]

timestep = data\_source["timestep"]

fns = io.archive.find\_by\_date(

date, root\_path, path\_fmt, fn\_pattern, fn\_ext, timestep, num\_next\_files=20

)

importer = io.get\_method(importer\_name, "importer")

R, \_, metadata = io.read\_timeseries(fns, importer, \*\*importer\_kwargs)

### 2. Konverzija Reflektivnosti u Intenzitet Padavina

Reflektivnost (Z) se može konvertovati u intenzitet padavina (R) pomoću Z-R odnosa. Ovaj odnos je definisan kao R = a \* (Z)^b , gde su a i b konstante.

a = 200 # Z-R relationship constant

b = 1.6 # Z-R relationship exponent

Z, metadata = to\_reflectivity(R, metadata)

precip\_intensity = a \* (Z \*\* b)

### 3. Detekcija Oluje

Detekcija oluja se vrši pomoću funkcija iz pysteps.feature modula. Ove funkcije identifikuju olujne ćelije na osnovu reflektivnosti.

input\_image = Z[2, :, :].copy()

timestamp = timelist[2]

cells\_id, labels = tstorm\_detect.detection(input\_image, time=timestamp)

### 4. Vizualizacija

Podaci o padavinama i identifikovane olujne ćelije se vizualizuju pomoću matplotlib biblioteke.

plot\_precip\_field(Z[2, :, :], geodata=metadata, units=metadata["unit"])

plt.xlabel("Swiss easting [m]")

plt.ylabel("Swiss northing [m]")

plot\_cart\_contour(cells\_id.cont, geodata=metadata)

plt.show()

### 5.Slanje Upozorenja

Za slanje upozorenja, prvo je neophodno inicijalizovati serijsku vezu kojom će se vršiti komunikacija između python skripte I mikrokontrolera:  
  
ser = serial.Serial('COM2', 9600, timeout=1)

Ova linija kreira objekat ser koji predstavlja otvorenu serijsku vezu sa uređajem na serijskom portu 'COM2' na brzini prenosa od 9600 bps (bitova po sekundi), sa timeout-om od 1 sekunde za čitanje. Komunikacija se vrši slanjem I primanjem podataka preko navedenog serijskog porta.

Ako je maksimalna reflektivnost veća od određene vrednosti (u ovom slučaju 25 dBZ, što predstavlja ozbiljniju oluju), šalje se upozorenje putem serijskog porta.

max\_dbz = np.nanmax(Z)

if max\_dbz > 25:

for \_ in range(10):

ser.write(b'T')

tm.sleep(0.5)

# Simulacija serijske komunikacije između STM32 mikrokontrolera i Python aplikacije

## 1. Korišćenje Virtual Serial Port Driver za simulaciju komunikacije

Iz razloga što nismo imali potrebne fizičke komponente za direktnu serijsku komunikaciju, koristili smo Virtual Serial Port Driver kako bismo simulirali interakciju između STM32 mikrokontrolera i Python aplikacije. Ovaj alat omogućio nam je kreiranje virtualnih COM portova - tacnije , COM1 i COM2. Svaki od ovih portova simulira fizički serijski port na računaru, omogućavajući nam da uspostavimo virtualni kanal za razmenu podataka između mikrokontrolera i računarske aplikacije (Python-a).

## 2. Python biblioteka za serijsku komunikaciju

U Python-u smo koristili biblioteku serial za upravljanje serijskom komunikacijom sa STM32 mikrokontrolerom preko virtualnog porta COM2. Ova biblioteka omogucava jednostavno slanje i primanje podataka preko serijskog porta. Konfigurisali smo serial.Serial objekat sa parametrima koji uključuju COM port, brzinu prenosa (baud rate) - 9600, i vreme čekanja (timeout) od 1 sekunde za čitanje operacije.

## 

## 3. STM32 mikrokontroler i UART1 konfiguracija

U simulacijskom okruženju kao što je Proteus, konfigurisali smo UART1 na STM32 mikrokontroleru za TX (slanje) i RX (primanje) operacije. UART1 je serijski interfejs koji nam je omogućio direktnu komunikaciju između mikrokontrolera i Python aplikacije putem serijskog porta. Potrebno je bilo konfigurisati port - COM1 i brzinu prenosa (baud rate) - 9600.

## 4. Funkcionalnost Aktiviranja Alarma

Implementirana je funkcionalnost za aktiviranje alarma na STM32 mikrokontroleru kada se primi određeni podatak preko UART-a. Ova funkcionalnost se sastoji iz dva glavna dela:

1. Python skripta šalje karakter 'T' preko serijskog porta kada maksimalna vrednost reflektivnosti (dBZ) prelazi prag od 25.

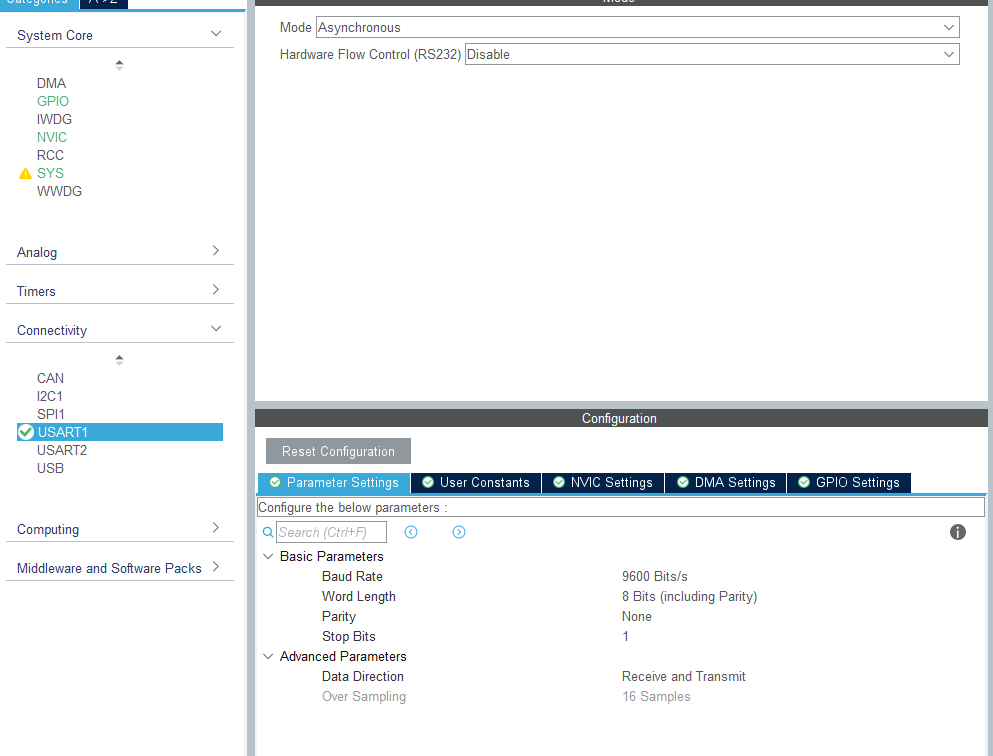
En bild som visar text, skärmbild, Teckensnitt

Automatiskt genererad beskrivning

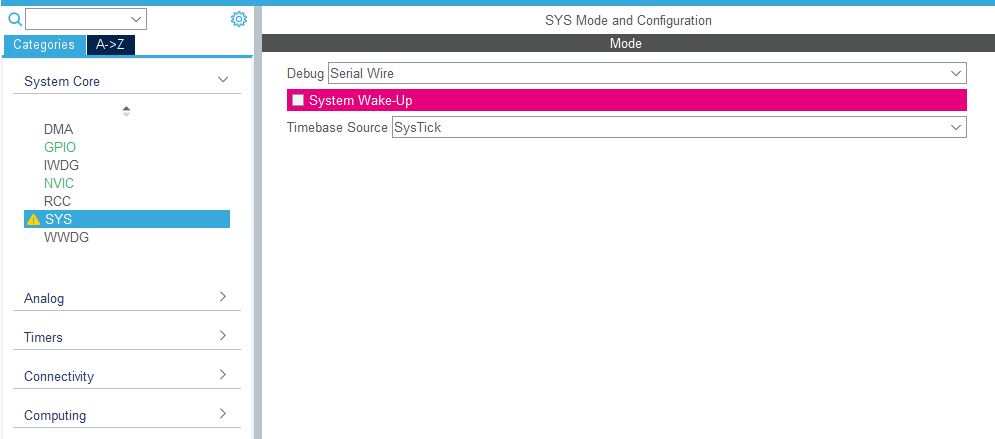
1. Kada STM32 mikrokontroler primi karakter 'T' preko UART1, uključiće alarm.

## 5. Konfiguracije na STM32

Konfiguracija UART-a za mikrokontroler je obavljena na sledeći način:



Za pravilno funkcionisanje alarma System Core (SYS) je postavljen na Serial Wire.



Krajnja kofiguracija pinova izgleda ovako.

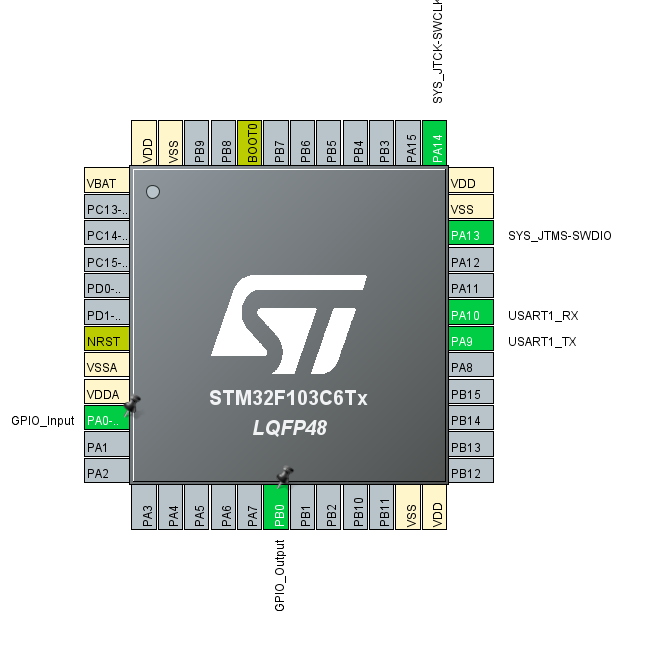
PA0-GPIO\_input

PBO-GPIO\_Output

PA9-USART1\_TX

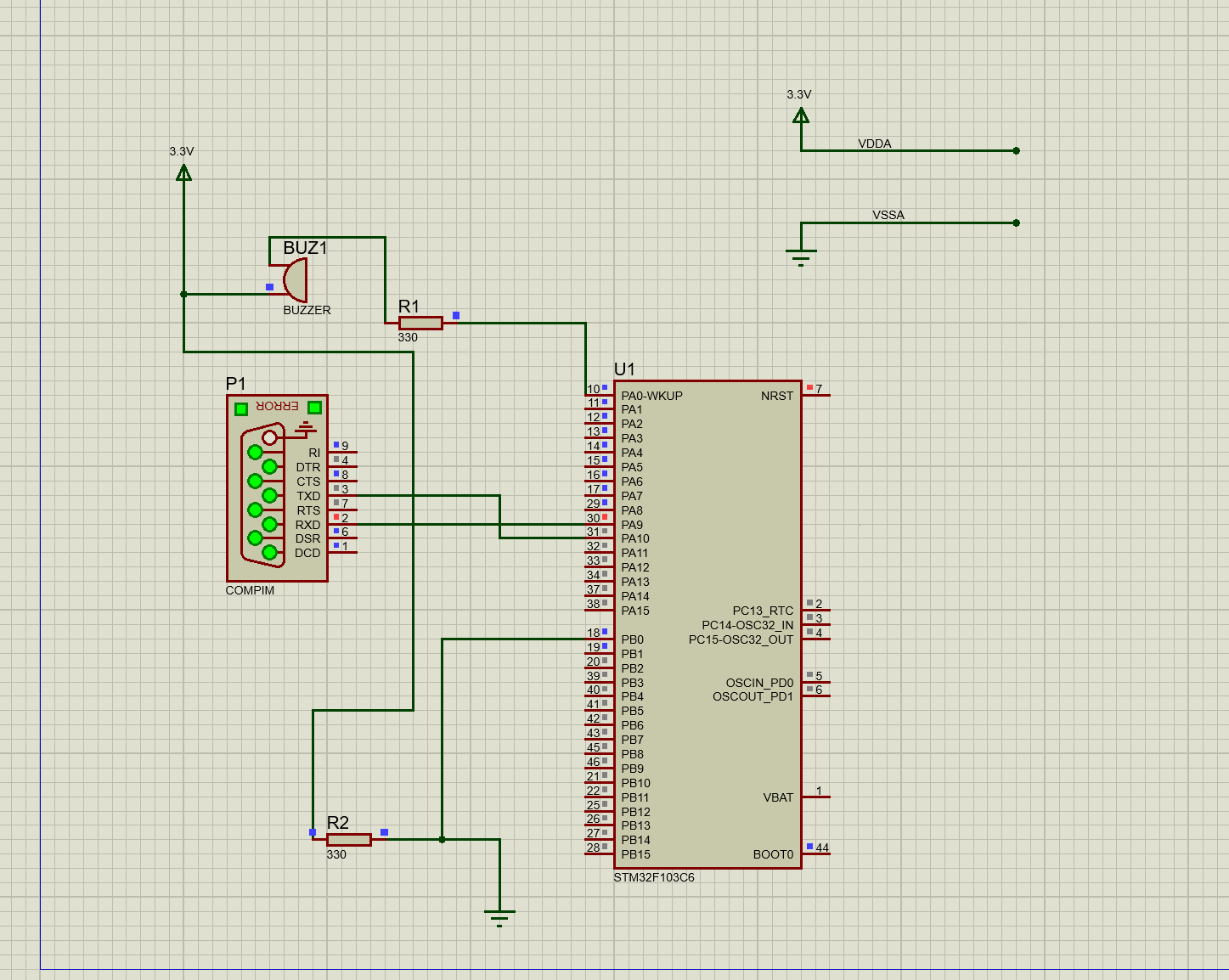
PA10-USART1\_RX

PA13 I PA14 se automatski inicijalizuju kao posledica prethodnih kofiguracija.

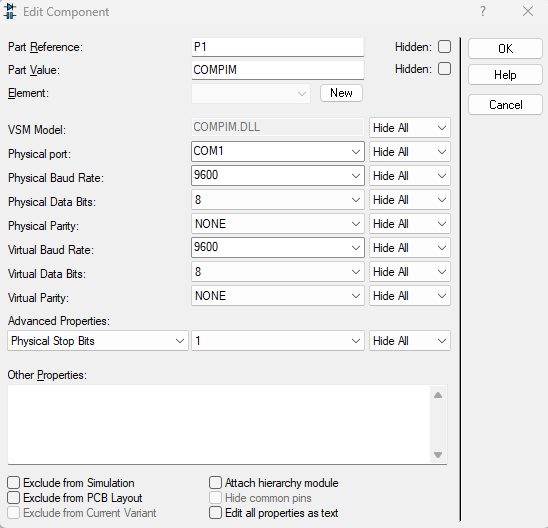


# Proteus šema

U Proteusu, šema za simulaciju projekta se sastoji od STM32F103C6 mikrokontrolera, COMPIM modula, dva otpornika i alarma. STM32F103C6 mikrokontroler je povezan sa COMPIM modulom, omogućavajući komunikaciju sa računarom preko serijskog porta. Dva otpornika su uključena u šemu radi pravilnog funkcionisanja i zaštite komponenti, dok alarm služi kao indikator događaja ili stanja sistema.

COMPIM (Computer Peripheral Interface Module) je modul u Proteus softveru za simulaciju, koji omogućava simulaciju serijskih komunikacija preko COM porta računara. Korišćenjem COMPIM modula, inženjeri mogu testirati serijsku komunikaciju između simuliranih uređaja i stvarnih aplikacija na računaru, što olakšava razvoj i debugging serijskih komunikacionih protokola.

COMPIM je podešen na port “COM1” I Baud Rate je postavljen na 9600. Alarm je ostao na standardnjoj konfiguraciji.



# Demo projekta

1. Aktivirati pysteps okruzenje

A screenshot of a computer

Description automatically generated

2. Otvoriti proteus šemu I učitati hex fajl u mikrokontroler.

A screenshot of a computer

Description automatically generated

3. Napraviti odgovarajuće uparene virtualne portove na lokalnoj mašini (primer u Virtual Serial Port Driver softveru)

A screenshot of a computer

Description automatically generated

4. Pokrenuti simulaciju u proteus-u

A computer screen shot of a computer

Description automatically generated

5. U aktiviranom pysteps okruženju, pokrenuti python skriptu za detekciju

A screenshot of a computer

Description automatically generated

6. Pojaviće se vizuelni prikaz obrađenih snimaka radara I skripta će se zablokirati.

A screenshot of a computer screen

Description automatically generated

7. Nakon što se prikaz isključi, skripta će nastaviti sa radom. Obračunaće se maksimalna refleksivnost I ukoliko ta vrednost prelazi određenu granicu, signal upozorenja biće poslat. Primanje I obradu signala sada možemo videti u simulaciji koju smo pokrenuli u proteusu. Alarm će se oglasiti(**Napomena: alarm se ne čuje na nekim verzijama proteus-a**). Vizuelni indikator predstavlja treptanje svetla na RX pinu(pinu primanja) COMPIM interfejsa.

A computer screen shot of a circuit board

Description automatically generated