Itasdi

Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Praktikum iz merno-akvizicionih sistema

Co-funded by the Erasmus+ Programme of the European Union

Itasdi

Innovative Teaching Approaches in development of Software Designed Instrumentation and its application in real-time systems

Faculty of Technical Sciences

Ss. Cyril and Methodius University Faculty of Electrical Engineering and Information Technologies

Zagreb University of Applied Sciences

School of Electrical Engineering University of Belgrade

Faculty of Physics Warsaw University of Technology

Co-funded by the Erasmus+ Programme of the European Union

Lekcija 14 – Serijska komunikacija sa *Arduinom*

Cilj

Cilj lekcije je da se studenti upoznaju sa:

- Serijskom komunikacijom sa *Arduino* mirkokontrolerom
- Obradom izuzetaka.

Oprema

- Računar sa instaliranim ***Spyder*** softverskim okruženjem.
- Računar sa instaliranim ***Arduino IDE*** softverskim okruženjem.
- ***Arduino UNO R3*** ploča.
- Protobord, kratkospojnici, otpornici, NTC termistor, svetleća dioda, RFID, akcelerometar.

14.1 Slanje podataka sa Arduina ka Python okruženju

Programsko okruženje *Arduino IDE* nije veoma zgodno za prikaz signala, i neku kompleksniju obradu signala. Preko serijske komunikacije *Arduino* mikrokontroler može komunicirati sa različitim programskim jezicima (*LabVIEW*, *MATLAB*, *Python*…). U okviru ovog poglavlja će biti prikazana komunikacija između *Arduino* mikrokontrolera i *Python* programskog jezika.

Zadatak 14.1.1.
Kreirati *Arduino* program koji šalje vreme rada mikrokontrolera u sekundama ka serijskom izlazu.

1. Otvoriti *Arduino IDE* programsko okruženje.
2. Uključiti biblioteku *TimerOne.h* izborom opcije *Sketch»Include Library»Add .ZIP Library*. Ukoliko biblioteka nije instalirana na raučnaru pogledati lekciju 11 koja opisuje dodavanje biblioteke u *Arduino IDE*.
3. Definisati promenljive **int brojac** i **bool stanje** koje imaju početne vrednosti 0 i false, respektivno.
4. U **setup** sekciji koda započeti serijsku komunikaciju inicijalizovati prvi tajmer na 1 s korišćenjem funkcije Timer1.initialize(1000000) i aktivirati prekid za ovaj tajmer korišćenjem funkcije Timer1.attachInterrupt(timerIsr).
5. Kreirati ISR pod nazivom **timerIsr**. Unutar **timerIsr** najpre obezbediti da se pri svakom ulasku u taj deo koda promenljiva **brojac** uveća za 1 i promenljiva **stanje** postavi na true.
6. Unutar glavne petlje pri svakoj iteraciji proveravati vrednost promenljive **stanje**. Samo ukoliko je **stanje** jednako true, na serijskom portu ispisati trenutnu vrednost **brojaca** i vratiti stanje promenljive **stanje** na false.

7. Sačuvati program pod nazivom *vreme.ino*, povezati *Arduino* sa računarom i spustiti kod na ploču. Proveriti rad programa korišćenjem *Serial Monitor*-a. Zatvoriti *Serial Monitor*.

Zadatak 14.1.2.
Kreirati *Python* program koji čita 30 vrednosti sa serijskog porta i ispisuje ih u konzoli.

1. U editoru *Spyder* programskog okruženja otvoriti novu *.py* skriptu (*File»New File*).
2. Funkcije koje služe za serijsku komunikaciju se nalaze u biblioteci *serial*. Dodati biblioteku u program.

NAPOMENA*:* ukoliko biblioteka nije instalirana na računaru, otvoriti Anaconda Command Prompt, ukucati komadu ***pip install pyserial*** i pritisnuti *Enter*. Nakon nekoliko sekundi biblioteka će se instalirati. Osnovne funkcije ove biblioteke su prikazane u Tabeli 14.1.1.

Tabela 14.1.1.

Funkcija	**Objašnjenje**
import serial	Uključivanje biblioteke za serijsku komunikaciju
ser = serial.Serial(port, baudrate)	Otvaranje serijske komunikacije. Argument port označava na koji ulaz je povezan *Arduino*, a argument baudrate označava brzinu serijske komunikacije i mora da se slaže sa brojem koji je definisan u *Arduino IDE*.
ser.read()	Čitanje jednog karaktera sa serijskog ulaza
ser.readline()	Čitanje niza karaktera, do '\n', sa serijskog ulaza
ser.close()	Zatvaranje serijske komunikacije i oslobađanje resursa

3. Otvoriti serijsku komunikaciju. U *Device Manager*-u se može proveriti na kom portu je povezan *Arduino*. Ukoliko je povezan na portu 13, a brzina serijske komunikacije je 9600, otvaranje serijske komunikacije je definisano naredbom: ser = serial.Serial("COM13", 9600).
4. Kreirati promenljivu *i* koja ima vrednost 0.
5. Kreirati *while* petlju u kojoj će se izvršavati kod sve dok vrednost promenljive *i* ne bude veća od 30. Unutar petlje pročitati vrednost sa serijskog ulaza (*ser.readline()*) i ispisati je u konzoli(*print()*). Na kraju svake iteracije petlje uvećati vrednost promenljive *i* za 1.
6. Nakon završetka *while* petlje zatvoriti serijsku komunikaciju.
7. Sačuvati program pod imenom *citanje_serijskog_porta.py*.
8. Pokrenuti program.

Primetiti da se u konzoli ne ispisuje samo vrednost **brojaca** koja se šalje sa *Arduino* mikrokontrolera, već postoje dodatni karakteri. Kako bi vrednost brojača mogla da se iskoristi u programu, potrebno ju je izdvojiti unutar *Python* programa.

Zadatak 14.1.3.
Dopuniti zadatak 14.1.2. tako da ispisuje samo trenutnu vrednost brojača. Kreirati funkciju **izdvajanje** koja vraća celobrojnu vrednost brojača.

1. Nakon čitanja vrednost sa serijskog ulaza, dekodirati tu vrednost pomoću funkcije *decode()*. Nakon dekodiranje se dobija promenljiva **value** tipa *string*.
```
arduinoValue = s.readline()
value = arduinoValue.decode()
```
2. Izvršiti konverziju promenljive **value** u ceo broj (*int()*) i ispisati tu vrednost u konzoli (*print()*).
3. Sačuvati program kao *citanje_serijskog_porta_brojac.py* i pokrenuti ga.

Primetiti da se sada ispisuje samo vrednost brojača, i da se promenljiva **value** pojavila u *Variable Explorer*-u.

14.2 Obrada izuzetaka

Svaki *Python* program se može nasilno prekinuti pritiskom tastera Ctrl+C na tastaturi tokom izvršavanja programa.

Zadatak 14.2.1.
Pokrenuti program *citanje_serijskog_porta.py*. U toku izvršavanja programa nasilno zaustaviti izvršavanje programa (Ctrl+C). Pokrenuti opet program. Primeti da se pojavila greška koja kaže da je traženi port zauzet. Pošto se aplikacija nije izvršila do kraja, serijski port je ostao otvoren. Ukucati u konzoli naredbu *ser.close()*. Kako je bitno izbeći ovakve situacije, potrebno je uvesti rukovanje izuzecima, Sl. 14.2.1.

```
import time

i = 0
try:
    while (i < 1000):
        print(i)
        i += 1
        time.sleep(1)

except KeyboardInterrupt:
    print('Prekid')
```

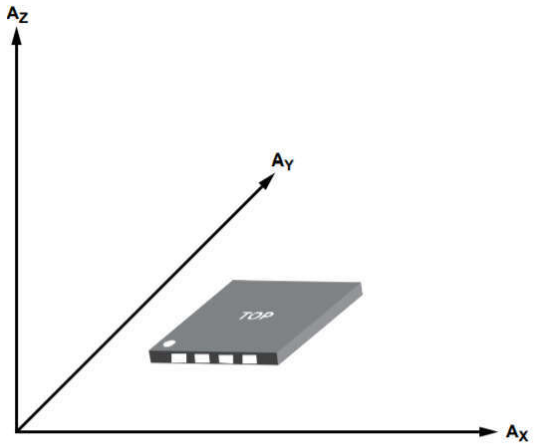
Sl. 14.2.1. Primer rukovanja izuzecima

Program sa Sl. 14.2.1 se izvršava sve dok vrednost brojača *i* ne postane jednaka 1000. U svakoj iteraciji *while* petlje se ispiše trenutna vrednost brojača, uveća za 1, i uvede kašnjenje od 1s. Taj deo koda se nalazi u *try* grani. Ukoliko dođe do nasilnog prekida programa preko tastature, ispisaće se poruka u konzoli da je došlo do prekida.

Zadatak 14.2.2.
Napisati program koji će ispisivati celobrojnu vrednost koja se čita sa serijskog ulaza. U slučaju prekida programa preko tastature treba da se zatvori serijska komunikacija i ispiše poruka korisniku.

1. U editoru *Spyder* programskog okruženja otvoriti novu *.py* skriptu (*File»New File*).
2. Uključiti biblioteku *serial*.
3. Otvoriti serijsku komunikaciju. Obratiti pažnju na definisanje potrebnih parametara – moraju da se slažu sa parametrima iz *Arduino* programa.
4. U *try* grani kreirati *while* petlju čiji je uslov uvek ispunjen (*while True*). Unutar *while* petlje pročitati vrednost koja je stigla na serijski ulaz dekodirati je i konvertovati u celi broj. Ispisati dobijenu celobrojnu vrednost u konzoli.
5. Ukoliko se pojavi izuzetak sa tastature (*except KeyboardInterrupt*) zatvoriti serijsku komunikaciju. Ispisati korisniku poruku da je došlo do prekida programa.
6. Kreirati i treću granu ukoliko se pojavi bilo kakva druga greška (*except*). Program treba da zatvori port i da ispiše korisniku poruku da je došlo do greške.
7. Sačuvati program pod imenom *citanje_i_ispis.py*.
8. Pokrenuti program i prekinuti ga posle nekog vremena preko tastature.

Akcelerometar predstavlja senzor ubrzanja. U najjednostavnijoj izvedbi sadrži prost inercijalni element čiji pomeraj usled ubrzanja daje informaciju o inercijalnoj sili koja na njega deluje. Danas se akcelerometri mogu realizovati u MEMS (MicroElectroMechanical System) tehnologiji i omogućavaju merenje komponenata ubrzanja duž sve tri ose a_x, a_y i a_z. Na Sl. 14.2.2 je prikazan ADXL335 3-osni akcelerometarski čip sa naznačenim osama duž kojih meri ubrzanje. Oko ovog čipa je izgrađen ceo GY-61 akcelerometarski modul.



Sl. 14.2.2 ADXL335 3-osni akcelerometar sa prikazanim mernim osama u odnosu na kućište čipa

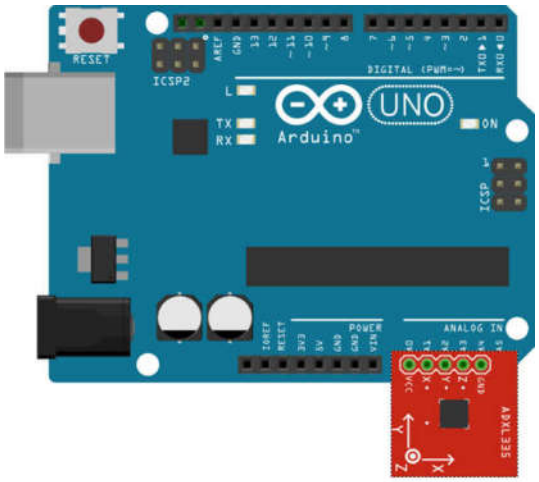
Pri korišćenju akcelerometra je potrebno imati u vidu da će na senzor u svakom trenutku, pored inercijalnog ubrzanja koje je posledica kretanja, delovati i ubrzanje Zemljine teže. Ova činjenica pruža osnov jednom od tipova kalibracije akcelerometara u kome se akcelerometar kontrolisano rotira čime se menja komponenta ubrzanja Zemljine teže koja deluje duž željene ose. Ukoliko poznajemo ugao rotacije, lako možemo povezati ubrzanje koje deluje duž ose sa izlaznim naponom akcelerometra čime se uređaj kalibriše.

Zadatak 14.2.3.
Kreirati *Arduino* program koji čita vrednost x komponente ubrzanja sa GY-61 akcelerometra na svakih 100 ms i šalje je preko serijske komunikacije ka *Python* programu.

1. Otvoriti *Arduino IDE* programsko okruženje.
2. Uključiti biblioteku *TimerOne.h*.
3. Definisati promenljive tipa volatile int **x** i volatile byte **stanje** koje imaju početne vrednosti 0.
4. Definisati konstante VCCPin, xPin, yPin, zPin i GNDPin tipa int koje uzimaju redom vrednosti A0, A1, A2, A3 i A4.
5. U setup sekciji koda započeti serijsku komunikaciju inicijalizovati prvi tajmer na 100 ms korišćenjem funkcije Timer1.initialize(100000) i aktivirati prekid za ovaj tajmer korišćenjem funkcije Timer1.attachInterrupt(timerIsr).
6. Zatim u ostatku setup sekcije podesiti pinove A0 i A4 kao **IZLAZNE** korišćenjem funkcije pinMode() i njihove vrednosti inicijalizovati redom na HIGH i LOW. Ovaj potez pinovima koji su uobičajeno analogni ulazi menja funkciju, te postaju digitalni izlazi. Vrednost digitalnog pina A0 se zatim postavlja na 5V, a digitalnog pina A4 na GND. Razlog pribegavanju ovakvog

vida rešenja leži u olakšavanju povezivanja GY-61 akcelerometra sa *Arduino* pločom.

7. Kreirati ISR pod nazivom **timerIsr**. Unutar **timerIsr** postaviti promenljivu **stanje** na 1 i pročitati vrednost sa analognog ulaza A1 (x osa akcelerometra).
8. Unutar glavne petlje pri svakoj iteraciji proveravati vrednost promenljive **stanje**. Samo ukoliko je **stanje** jednako 1, ispisati vrednost promenljive **x** na serijskom portu i vratiti stanje promenljive **stanje** na 0.
9. Sačuvati program pod nazivom *akcelerometar_timer.ino*.
10. Povezati kolo kao na Sl. 14.2.3 prostim ubacivanjem pinova akcelerometra u odgovarajuće ulaze *Arduino* ploče.



Sl. 14.2.3 Šema povezivanja GY-61 akcelerometra sa *Arduino* pločom

11. Povezati *Arduino* sa računarom i spustiti kod na ploču. Proveriti rad programa korišćenjem *Serial Monitor*-a i *Serial Plotter*-a. Rotirati akcelerometar i posmatrati šta se događa sa signalom x ose akcelerometra. Zatvoriti *Serial Monitor* i *Plotter*.
12. **Isključiti** *Arduino* iz računara.

Zadatak 14.2.4.

Napisati *Python* program koji čita podatke koji mu stižu sa serijskog porta. Zatim skalira tu vrednost u opseg od 0 do 5 V i dodaje je u listu. Kada korisnik prekine izvršavanje programa preko tastature prikazati skaliran signal na grafiku.

1. U editoru *Spyder* programskog okruženja otvoriti novu *.py* skriptu (*File»New File*).
2. Uključiti biblioteke *serial, numpy* i *matplotlib.pyplot*.
3. Otvoriti serijsku komunikaciju. Obratiti pažnju na definisanje potrebnih parametara – moraju da se slažu sa parametrima iz *Arduino* programa.
4. Kreirati promenljivu **lista** i postaviti joj početnu vrednost na [].
5. U *try* grani kreirati *while* petlju čiji je uslov uvek ispunjen (*while True*). Unutar *while* petlje pročitati vrednost koja je stigla na serijski ulaz dekodirati je i konvertovati u celi broj. Celobrojnu vrednost smestiti u promenljivu **vrednost**.

6. Kako *Arduino* nakon A/D konverzije vraća digitalizovane vrednosti u opsegu od 0 – 1023, potrebno ih je skalirati na realan opseg napona od 0 – 5 V. Promenljivu **vrednost** skalirati na zadati opseg i dodati je u listu (funkcija *append*).
7. Ukoliko se pojavi izuzetak sa tastature (*except KeyboardInterrupt*) zatvoriti serijsku komunikaciju. Kreirati promenljivu *fs* = 10. Kreirati vremensku osu pomoću funkcije *arange*. Vremenska osa treba da ima vrednosti od 0 do *len(signal)/fs* sa korakom 1/fs. Prikazati odbirke smeštene u promenljivoj **lista** u zavisnosti od vremena na grafiku.
8. Kreirati i treću granu ukoliko se pojavi bilo kakva druga greška (*except*). Program treba da zatvori port i da ispiše korisniku poruku da je došlo do greške.
9. Sačuvati program pod imenom *citanje_skaliranje.py*.
10. Pokrenuti i testirati program.

14.3 Slanje podataka iz Python okruženja ka Arduinu

Zadatak 14.3.1.
Kreirati *Arduino* program koji čita informacije sa serijskog porta. U zavisnosti da li pročita ON ili OFF treba da uključi, odnosno isključi svetleću diodu.

1. Otvoriti *Arduino IDE* programsko okruženje.
2. Definisati promenljive **inputString** i **operation** tipa String i String respektivno. Podesiti inicijalne vrednosti promenljivih **inputString** i **operation** na prazan string ("").
3. U setup delu programa započeti serijsku komunikaciju sa računarom korišćenjem funkcije Serial.begin(9600) i podesiti pin na kome se nalazi ugrađena svetleća dioda kao izlazni korišćenjem funkcije pinMode(LED_BUILTIN, OUTPUT).
4. Kreirati novu funkciju serialEvent() kao na Sl. 11.1 (lekcija 11).
5. Unutar glavne petlje pri svakoj iteraciji proveravati vrednost promenljive **operation**. Ukoliko je vrednost promenljive jednaka "ON" uključiti diodu, a ako je "OFF" isključiti diodu.
6. Sačuvati program pod nazivom *ON_OFF.ino*.
7. Povezati *Arduino* sa računarom i spustiti kod na ploču.

Zadatak 14.3.2.
Napisati *Python* program koji čita šta korisnik unese preko konzole i šalje pročitanu vrednost preko serijske komunikacije ka *Arduino* mikrokontroleru.

1. U editoru *Spyder* programskog okruženja otvoriti novu *.py* skriptu (*File»New File*).
2. Uključiti biblioteke *serial* i *time*.
3. Otvoriti serijsku komunikaciju. Obratiti pažnju na definisanje potrebnih parametara – moraju da se slažu sa parametrima iz *Arduino* programa.
4. Kreirati *while* petlju čiji je uslov uvek ispunjen (*while True*). Unutar *while* petlje pitati korisnika da unese poruku (funkcija *input*). Korisnik treba da unese ON da bi uključio diodu i OFF da bi isključio diodu. Ukoliko korisnik unese STOP treba zatvoriti serijsku komunikaciju i prekinuti petlju (naredba *break*). Na poruku koju je uneo korisnik dodati karakter za novi red '\n' (spajanje stringova je moguće pomoću operatora +). Preko funkcije *ser.write(poruka.encode())* poslati informaciju ka serijskom portu.
5. Dodati kašnjenje unutar *while* petlje pomoću funkcije *sleep* iz biblioteke *time*. time.sleep(1) – kašnjenje od 1 s
6. Sačuvati program pod imenom *upis.py*.
7. Pokrenuti i testirati program.

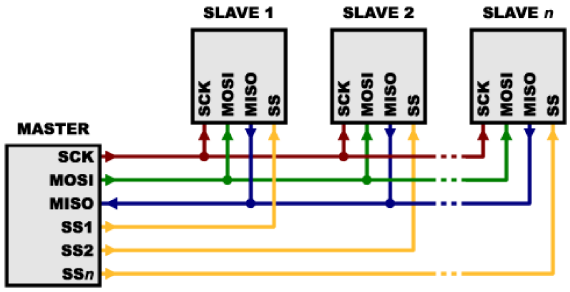
Zadatak 14.3.3 - za samostalan rad
Kreirati aplikaciju koja kontinualno čita vrednost napona sa NTC termistora i ako ta vrednost pređe neki prag koji korisnik unese treba da se uključi svetleća dioda. *Arduino* program treba da čita napona na analognog ulaza i šalje ga ka serijskom portu. Takođe, u zavisnosti od poruke koju primi sa serijskog porta treba da uključi/isključi diodu. *Python* program treba da pita korisnika da unese prag napona.

Zatim, treba da kontinualno (unutar *while* petlje) čita vrednosti sa serijskog porta, skalira dobijenu vrednost i poredi je sa pragom. Ako je vrednost napona veća od praga, treba poslati komandu ON ka *Arduino* mikrokontroleru, a ako je manja onda treba poslati komandu OFF. Program se zaustavlja preko tastature. Obezbediti adekvatno zatvaranje resursa.

14.4 Komunikacija Arduino ploče sa RFID – RC522 modulom

RFID je skraćenica engleskog termina *Radio-Frequency IDentification*, sistema za razmenu podataka bez direktnog kontakta na malim rastojanjima. Ono što ovaj vid komunikacije čini posebno interesantnim je činjenica da ostvarivanje komunikacije ne zahteva dva aktivna uređaja, već je dovoljno da samo jedan uređaj poseduje izvor energije, dok drugi uređaj može biti potpuno pasivan (kartice za plaćanje ili identifikaciju, Bus-Plus i mnogo drugih primera.). Aktivan uređaj svojom antenom emituje RF signale definisane učestanosti. Ovi signali imaju dvostruku ulogu u ostvarivanju RFID komunikacije sa pasivnim uređajem, tj. služe i za komunikaciju, ali i kao izvor energije pasivnoj kartici na kojoj se nalaze podaci. Jedna od prednosti koje RFID pruža leži upravo u tome što kartica na kojoj se nalaze podaci ne zahteva sopstveno baterijsko napajanje, te ne postoji potreba za zamenom baterija, a životni vek kartice može biti veoma dug.

RFID modul koji se koristi u ovoj vežbi **radi na 3.3 V**. Komunikacija između modula i *Arduino* ploče se ostvaruje putem SPI (engl. *Serial Peripheral Interface*) interfejsa. SPI predstava sinhroni (za razliku od klasičnog serijskog porta) serijski protokol za razmenu podataka između mikrokontrolera i perifernih uređaja ili više mikrokontrolera. SPI komunikacija zahteva postojanje jednog *Master* uređaja (obično mikrokontroler, kod nas *Arduino*) i bar jednog (a može i više) *Slave* uređaja (periferije, u ovom poglavlju RFID modul). Sinhronizacija komunikacije se obezbeđuje postojanjem *Clock* signala na zasebnom pinu SCK. Slanje podataka sa *Master* uređaja na *Slave* uređaj se vrši preko konekcije MOSI (*Master Out Slave In*), dok se podaci sa *Slave* uređaja na *Master* uređaj šalju putem MISO (*Master In Slave Out*) konekcije. Pored tri navedene linije (MOSI, MISO i SCK), postoji i četvrta linije *Slave Select* (SS) koja omogućava izbor *Slave* uređaja, tj. signalizira *Slave* uređaju da *Master* sa njim komunicira. Ovo omogućava kontrolu više periferija preko istog SPI porta. Tipičan primer takve konfiguracije je prikazan na Sl. 14.4.1.



Sl. 14.4.1 Povezivanje više *Slave* uređaja na jedan *Master* uređaj putem SPI interfejsa. Slika je preuzeta sa sajta *learn.sparkfun.com*.

Pinovi za SPI komunikaciju se razlikuju u zavisnosti od tipa ploče. *Arduino UNO R3* ploča poseduje pinout prikazan u Tabeli 14.4.1.

Tabela 14.4.1.

Ime linije	Broj pina
MOSI	11
MISO	12
SCK	13

Prednosti SPI interfejsa uključuju brzinu prenosa, jednostavnost prijemnog hardvera, kao i mogućnost slanja podataka na više uređaja preko istog porta, dok se mane ovog vida komunikacije uglavnom odnose na broj potrebnih linija za komunikaciju, posebno u slučaju više *Slave*-ova, i na činjenicu da *Master* mora posredovati svakoj komunikaciji između *Slave*-ova.
Modul koji se koristi za RFID komunikaciju u ovom poglavlju komunicira na učestanosti od 13.56 MHz. Kartica i privezak za ključeve koji dolaze uz modul poseduju sopstveni identifikacioni broj, kao i memoriju od 1024 bajta podeljenih u 16 sektora, koja je dostupna za skladištenje podataka na kartici.

Zadatak 14.4.1.
Kreirati program koji očitava identifikacioni broj sa kartice i ispisuje ga na računaru.

1. Pokrenuti *Arduino IDE* okruženje i u *Sketch* uključiti biblioteku SPI.h koja dolazi zajedno sa okruženjem.
2. Sa linka https://github.com/miguelbalboa/rfid preuzeti biblioteku u formi .ZIP datoteke, a zatim je uključiti u *Sketch*.
3. Korišćenjem #define definisati SS_PIN i RST_PIN kao pinove 10 i 9 respektivno.
4. Kreirati MFRC522 instancu, Sl. 14.4.2. Pinovi MOSI i MISO su automatski podešeni.

```
MFRC522 mfrc522(SS_PIN, RST_PIN);
```

Sl. 14.4.2

5. Unutar setup dela koda obezbediti inicijalizaciju serijske komunikacije kao i u ranijim primerima, kao i inicijalizaciju SPI komunikacije pozivanjem SPI.begin().
6. Inicijalizovati MFRC522 modul pozivanjem mfrc522.PCD_Init().
7. Poslednje što je potrebno uraditi u ovom delu koda je obezbediti slanje poruke korisniku da je inicijalizacija prošla i da je potrebno približi karticu čitaču.
8. Unutar loop dela programa je potrebno najpre proveriti da li je nova kartica prislonjena čitaču i proveriti da li je moguće dobiti njen ID. Ukoliko nije, potrebno je prekinuti datu iteraciju petlje, Sl. 14.4.3.

```
// Da li ima novih kartica?
if ( ! mfrc522.PICC_IsNewCardPresent())
{
  return;
}
// Proverava da li je moguće pročitati ID kartice
if ( ! mfrc522.PICC_ReadCardSerial())
{
  return;
}
```

Sl. 14.4.3

9. Ukoliko je prethodna sekcija koda uspešno izvršena i iteracija nije prekinuta potrebno je ispisati korisniku ID prislonjene kartice u formi „ID kartice je:“, zatim ID kartice i nakon toga preći u novi red. Kako bi se ispisao ID kartice,

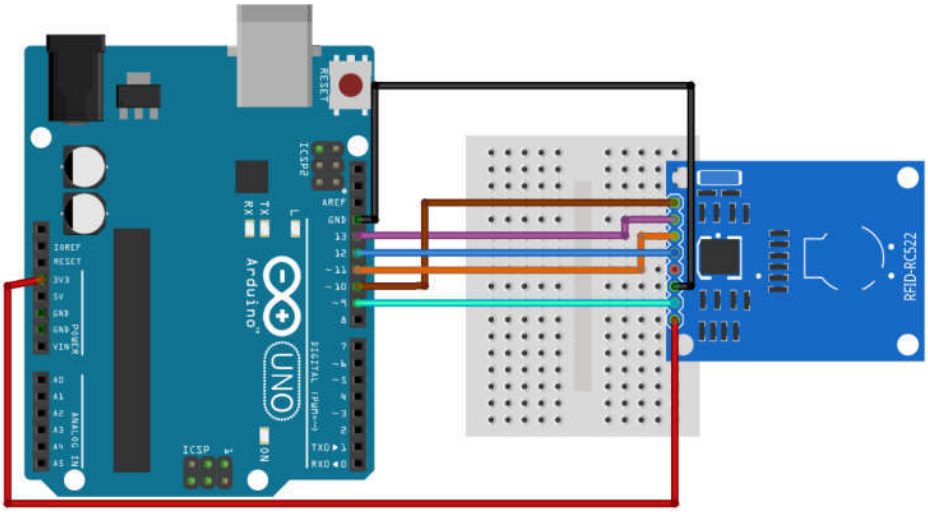
najpre je potrebno kreirati promenljivu String tipa inicijalizovanu kao prazan string koju možemo nazvati **content**.

10. Zatim je potrebno redom pročitati sva četiri bajta identifikacionog broja korišćenjem funkcije mfrc522.uid.uidByte[], pretvoriti ih u String i nadovezati ih korišćenjem funkcije concat, Sl. 4.4.4. Ispisivanje brojeva u heksadecimalnom obliku se obezbeđuje drugim argumentom funkcije String() prikazanog koda. Brojevi su razdvojeni razmakom kako bi se lakše pročitali.

```
for (byte i = 0; i < mfrc522.uid.size; i++)
{
  content.concat(String(mfrc522.uid.uidByte[i], HEX));
  content.concat(" ");
}
```

Sl. 14.4.4

11. Dobijeni String **content** ispisati na računaru.
12. Povezati **SAMO** *Arduino* ploču sa računarom i spustiti kod koji je prethodno napisan i sačuvan. (Ova stavka je bitna pre povezivanja RFID modula sa pločom kako se pre spuštanja koda na ploču RFID modul ne bi spalio usled nekog ranijeg koda koji se već nalazi na ploči).
13. **Isključiti** *Arduino* iz računara i zatim ga povezati sa RFID modulom prema Sl. 14.4.5.



Sl. 14.4.5. Šema prema kojoj je potrebno povezati RFID modul sa *Arduino* pločom

Voditi računa da modul radi na **3.3V** te mu je napajanje potrebno dovesti sa 3.3V pina na *Arduino* ploči! Ukoliko se pogreši pri povezivanju i modul se poveže na 5V postoji realna šansa da će se **spaliti**! Radi provere, u Tabeli 14.4.2 su izlistani nazivi pinova sa odgovarajućim brojevima.

Tabela 14.4.2.

SDA	SCK	MOSI	MISO	IRQ	GND	RST	3.3V
10	13	11	12	N.C.	GND	9	3.3V

14. Povezati *Arduino* sa računarom i otvoriti *Serial Monitor*. Testirati rad programa prinošenjem kartice i/ili priveska čitaču i zapisati njihove identifikacione brojeve na papiru jer će biti korišćeni u narednom zadatku.
15. Sačuvati program kao *citanje_ID_kartice.ino*.
16. **Isključiti** *Arduino* iz računara.

Zadatak 14.4.2.
Kreirati *Python* program koji čita podatke sa serijskog porta. Na serijski port stiže ID kartice. Poveriti da li se pročitan ID poklapa sa predefinisanom vrednošću. Ispisati poruku korisniku da li je došlo do poklapanja.

1. U editoru *Spyder* programskog okruženja otvoriti novu *.py* skriptu (*File»New File*).
2. Uključiti biblioteke *serial*.
3. Otvoriti serijsku komunikaciju. Obratiti pažnju na definisanje potrebnih parametara – moraju da se slažu sa parametrima iz Arduino programa.
4. Kreirati promenljivu **ID** i postaviti joj početnu vrednost na željeni ID kartice (string koji može, a i ne mora da se slaže sa ID vaše kartice).
5. U *try* grani kreirati *while* petlju čiji je uslov uvek ispunjen (*while True*). Unutar *while* petlje pročitati vrednost koja je stigla na serijski ulaz, dekodirati je i smestiti u promenljivu **ID_tren**.
6. Porediti string **ID** sa **ID_tren**. Ukoliko se vrednost poklapaju ispisati korisniku poruku da je operacija uspešna, a ukoliko se ne poklapaju obavestiti ga da operacija nije uspešna.
7. Ukoliko se pojavi izuzetak sa tastature (*except KeyboardInterrupt*) ili bilo kakva druga greška zatvoriti serijsku komunikaciju.
8. Sačuvati program pod imenom *ID_provera_predefinisano.py*.
9. Pokrenuti i testirati program.

14.5 Kreiranje grafičkog interfejsa - multithreading

Kreiranje grafičkog interfejsa podrazumeva da se klasa vezana za aplikaciju kontinualno izvršava sve dok je korisnik ne zatvori. To znači, da izvršavanje koda koji nije definisano komandama na interfejsu nije moguće. Pošto komunikacija sa serijskim portom mora da se izvršava kontinualno, uz izvršavanje aplikacije, potrebno je uvesti niti. Niti omogućavaju paralelno izvršavanje programa.

Funkcije za manipulaciju sa nitima se nalaze u biblioteci *threading*. Nit se kreira pomoću funkcije *Thread*. Kao argument ove funkcije se prosleđuje funkcija koju želimo da izvršavamo paralelno:

```
nit = threading.Thread(funkcija)
```

Prilikom kreiranja niti se kreira objekat klase. Nit se pokreće pomoću metode *start*:

```
nit.start()
```

Prilikom pokretanja niti, izvršava se kod napisan u telu funkcije jednom. Ukoliko je potrebno kontinualni izvršavanje koda, u telu funkcije treba stajati *while* petlja.

Zadatak 14.5.1

Dopuniti dati *Python* kod koji proverava da li se ID očitane kartice slaže sa tekstom koju je korisnik uneo preko aplikacije.

1. U *Spyder* okruženju otvoriti *ID_kartice_GUI.py* skriptu. U skripti se nalazi kod koji definiše izgled grafičkog interfejsa.
2. U glavnom delu programa otvoriti serijsku komunikaciju. Obratiti pažnju na definisanje potrebnih parametara – moraju da se slažu sa parametrima iz *Arduino* programa.
3. Definisati dve funkcije **fcn1** i **fcn2**.
4. U telu **fcn1** kreirati globalnu promenljivu **data**. Pročitati vrednost sa serijskog ulaza, dekodirati je i smestiti u promenljivu **data,** Sl. 14.5.1.

```
def fun1():
    global data
    while(True):
        data = ser.readline().decode()
```

Sl. 14.5.1 Funkcija **fcn1**

5. U telu **fcn2** kreirati aplikaciju, Sl. 14.5.2.

```
def fun2():
    app = QApplication(sys.argv)
    ex = App()
    app.exec_()
```

Sl. 14.5.2 Funkcija **fcn2**

6. U funkciji **initUI** klase **App** kreirati globalnu promenljivu data (*global data)*. U funkciji koja je vezana za pritisak tastera **button_fcn** dodati poređenje promenljive **data** sa tekstom koji je korisnik uneo u polje za unos teksta (**poruka**).

7. Ukoliko se poklapaju ove dve vrednosti ispisati poruku u polje za ispis teksta „OK“. Ukoliko se ne poklapaju ispisati „FAIL“.
8. Nakon otvaranja serijskog porta, i definisanja potrebnih funkcija, kreirati i startovati dve niti.
9. Sačuvati program kao *ID_provera_tekst.py*.

```
t1 = Thread(target=fun1)
t2 = Thread(target=fun2)

t1.start()
t2.start()
```

Sl. 14.5.3 Kreiranje i pokretanje niti

Zadatak 14.5.2. - za samostalan rad

Kreirati *Python* aplikaciju koja proverava da li se ID očitane kartice nalazi u tekstualnoj datoteci. Ukoliko se nalazi ispisati korisniku poruku da mu je ulazak odobren i šalje se poruka ka *Arduino* mikrokontroleru da uključi svetleću diodu (šalje se poruka „ON“). Ukoliko se ne nalazi, korisniku se ispisuje poruka da mu nije odobren ulazak i dioda treba da se treperi sa učestanošću 1 Hz (šalje se poruka „ALARM“). Napisati i odgovarajući *Arduino* kod.