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 11 – Serijski port. Korišćenje prekida. Primena senzora i aktuatora.

Cilj

Cilj vežbe je da studente:

- upozna sa prednostima i ograničenjima standardnih metoda čitanja sa serijskog porta.
- uvede u principe primene prekida (eng. *interrupt*) uz naglašavanje značaja ovog tipa programiranja u situacijama kada je neophodno obezbediti brzu reakciju mikrokontrolera na spoljašnje događaje ili prosto precizno kontrolisati frekvenciju odabiranja pri akviziciji.
- uputi u uobičajenu problematiku povezivanja spoljašnjih senzora i aktuatora na *Arduino* ploču.

Oprema

- Računar sa instaliranim ***Arduino IDE*** softverskim okruženjem.
- ***Arduino UNO R3*** ploča.
- Protobord, kratkospojnici, otpornici (220 Ω, 10 KΩ), NTC termistor, svetleća dioda, kapacitivni senzor dodira, četvorocifreni sedmosegmentni displej sa TM1637 čipom, HC-SR04 ultrazvučni modul, pasivni *piezo buzzer*, *Servo* motor.

NAPOMENA: Najnoviju verziju *Arduino IDE* okruženja, kao i dokumentaciju vezanu za platformu je moguće naći na zvaničnom *Arduino* sajtu: https://www.arduino.cc/

11.1 Slanje poruka Arduino ploči preko serijskog porta

Ukoliko je potrebno kontrolisati izvršavanje programa na *Arduino* ploči putem računara, tj. omogućiti korisniku interakciju sa programom slanjem definisanih poruka, koristiće se skup funkcija za prijem i čitanje poruka poslatih *Arduino* ploči. Kako poruke obično dolaze na ploču povremeno i u nedefinisanim vremenskim intervalima (kada korisnik odluči da ih pošalje), potrebno je obezbediti da se čitanje sa porta izvršava samo onda kada postoji poruka koju je potrebno pročitati. U tom slučaju je *Arduino* slobodan da nesmetano obavlja druge zadatke između poruka. Kroz naredni primer će biti objašnjeni ključni koncepti u realizaciji programa koji omogućava korisniku da slanjem poruke „*start*“ ili „*stop*“ započne, odnosno zaustavi akviziciju analognog napona sa učestanošću od 1 Hz. Indikacija da *Arduino* čeka na start akvizicije će biti uključivanje i isključivanje ugrađene svetleće diode na ploči sa učestanošću od 1 Hz. Na ovaj način će biti ilustrovana sposobnost programa da

nesmetano izvršava druge zadatke (analognu akviziciju ili uključivanje svetleće diode) u periodu između poruka.

3. Pokrenuti *Arduino IDE* okruženje i kreirati promenljive **inputString**, **operation** i **voltage** tipa String, String i int respektivno. Podesiti inicijalne vrednosti promenljivih **inputString** i **operation** na prazan string ("") i promenljive **voltage** na 0.
4. 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).
5. Za sada ostaviti prazanu loop funkciju i kreirati novu funkciju, naziva serialEvent(). Ova funkcija ne vraća ništa kao rezultat, a biće pozvana automatski svaki put kada se detektuju dolazeći podaci. U telu funkcije serialEvent je potrebno upisati kod koji će omogućiti čitanje dostupnih podataka na serijskom portu. Kod koji je potrebno kreirati unutar funkcije serialEvent je prikazan na Sl. 11.1.1.

```
void serialEvent() {
  while (Serial.available()) {

    char inChar = (char)Serial.read();
    if (inChar == '\n') {
      operation = inputString;
      inputString = "";
      Serial.println(operation);
    }
    else {
      inputString += inChar;
    }


  }
}
```

Sl. 11.1.1. Funkcija serialEvent()

Ovaj kod omoućava čitanje karaktera sa serijskog porta sve dok postoje karakteri u baferu koje je moguće pročitati. Nakon što je karakter pročitan proverava se da li je "*newline*" karakter. Ukoliko nije, dodaje se na postojeći **inputString**. Ukoilko je ipak poslati karakter "*newline*", tj. "\n", smatra se da je cela poruka poslata i vrednost iz promenljive **inputString** se upisuje u promenljivu **operation** koja definiše trenutni zadatak programa ("start" ili "stop"). Radi provere rada programa se podaci iz promenljive **operation** šalju na računar gde se ispisuju, a promenljiva **inputString** se vraća na prazan string kako bi bila spremna za čitanje sledeće poruke.

Serial.read(): funkcija za čitanje iz bafera sa serijskog porta.
Serial.available(): funkcija koja vraća broj bajtova (karaktera) koji su pristigli i nalaze se u baferu na serijskom portu. Maksimalan broj nepročitanih karaktera u baferu je 64.

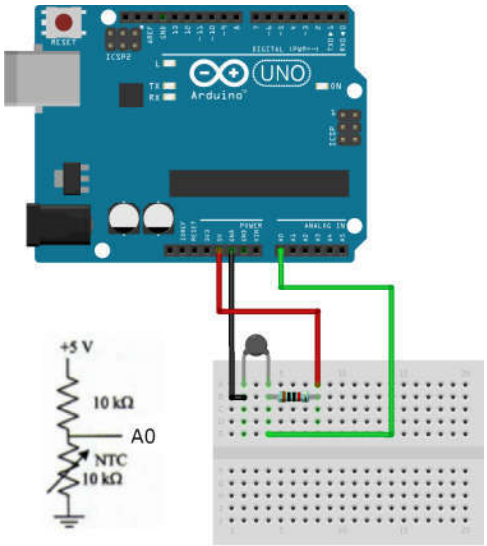
---

6. Unutar loop funkcije je potrebno obezbediti izvršavanje različitih zadataka u zavisnosti od trenutne vrednosti promenljive **operation**. Ukoliko je u promenljivu **operation** upisano "start", potrebno je da program sa učestanošću od 1 Hz izvršava akviziciju analognog napona i slanje dobijene vrednosti ka računaru. Ukoliko je u promenljivu **operation** upisano "stop", potrebno je obezbediti da se sa učestanošću od 1 Hz uključuje i isključuje svetleća dioda ugrađena na *Arduino* ploči. Ukoliko vrednost promenljive **operation** nije ni "start" ni "stop", korsniku se prijavljuje greška i u promenljivu **operation** se upisuje "stop". Kod koji unutar loop sekcije obebeđuje željenu funkcionalnost je prikazan na Sl. 11.1.2.

```
void loop() {

  if (operation == "start") {
    voltage = analogRead(A0);
    Serial.println(voltage);
    delay(1000);
  }

  else if (operation == "stop") {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(500);
    digitalWrite(LED_BUILTIN, LOW);
    delay(500);
  }

  else {
    Serial.println("Greska pri unosu! Program se vraca u stanje cekanja.");
    operation = "stop";
    delay(1000);
  }
}
```

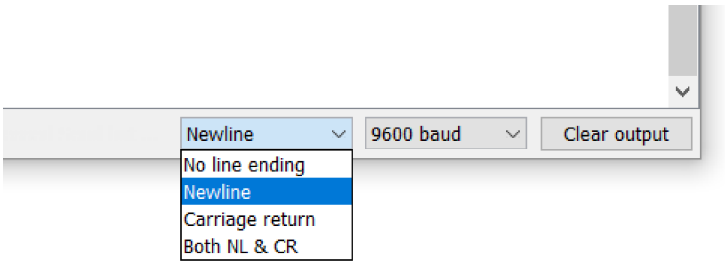
Sl. 11.1.2. Kod unutar loop funkcije programa

7. Povezati kolo prema šemi sa S. 11.1.3. Povezati *Arduino* sa računarom, sačuvati program kao *SerialEvent_start_stop.ino* i spustiti ga na ploču.



Sl. 11.1.3. Povezivanje naponskog razdelnika sa NTC termistorom na *Arduino*

8. Testirati rad programa u slučaju izbora različitih opcija iz padajućeg menija na Sl. 11.1.4.
9. **Isključiti** *Arduino* iz računara.



Sl. 11.1.4. Definisanje karaktera koji še šalju na kraju poruke

Zadatak 11.1.1. – za samostalni rad

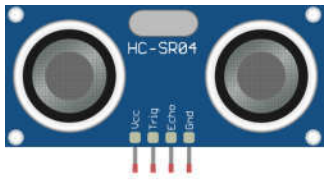
Realizovati program koji omogućava korisniku da slanjem vrednosti od 0 do 255 preko serijskog porta podešava faktor ispune impulsno širinski modulisanih pulseva koji kontrolišu svetleću diodu povezanu u kolo prema Sl. 10.5.2. Za konverziju iz *string*-a u *int* koristiti sekciju koda sa Sl. 11.1.5.

```
if (inChar >= '0' && inChar <= '9') // is inChar a number?
{
  inputString = inputString * 10 + inChar - '0'; // yes, accumulate the value
}
```

Sl. 11.1.5. Konverzija iz stringa

11.2 Ultrazvučni senzor rastojanja

Modul HC-SR04 koji predstavlja ultrazvučni senzor rastojanja se sastoji iz dva osnovna elementa, emitera ultrazvuka koji generiše ultrazvučne talase frekvencije 40 KHz i detektora ultrazvuka koji detektuje ultrazvučne talase koji se reflektuju od prepreke i vraćaju ka modulu. Na taj način je, poznajući brzinu zvuka u vazduhu, jednostavno moguće doći do informacije o udaljenosti prepreke od modula ukoliko se izmeri vreme proteklo od generisanja ultrazvučnog talasa do njegovog povratka na detektor. *Pinout* modula HC-SR04 je prikazan na Sl. 11.2.1.



Sl. 11.2.1. *Pinout* HC-SR04 modula

Dok je funkcija pinova Vcc i Gnd jasna (pinovi za napajanje modula), funkcija pinova Trig i Echo mora biti dodatno objašnjena. Naime, Trig pin služi za generaciju ultrazvučnih pulseva. Ukoliko se na Trig pin dovede napon od 5 V u trajanju od 10 µs doći će do generacije 8 ultrazvučnih pulseva, dok Echo pin pri detekciji ultrazvuka menja vrednost na LOW.

1. Otvoriti novi program i kreirati promenljive **trajanje** i **rastojanje** tipa long, i int respektivno.
2. Kreirati konstante **trigPin** i **echoPin** tipa int koje će označavati pinove 9 i 10 *Arduino* ploče respektivno.
3. U setup sekciji koda definisati **trigPin** digitalni pin kao izlazni, a **echoPin** digitalni pin kao ulazni. Započeti serijsku komunikaciju sa računarom.
4. U loop sekciji koda pri svakom ulasku najpre obezbediti da je stanje pina **trigPin** podešeno na LOW. Zatim zaustaviti izvršavanje programa na 2 µs korišćenjem funkcije delayMicroseconds(2).
5. Sledeći korak loop sekcije zahteva postavljanje **trigPin** digitalnog pina u stanje HIGH na 10 µs kako bi se generisali ultrazvučni pulsevi. Ovaj zadatak je moguće realizovati uzastopnim korišćenjem funkcija digitalWrite() i delayMicroseconds() kao na Sl. 11.2.2.

```
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

Sl. 11.2.2. Kod koji obezbeđuje slanje napona od 5 V u trajanju od 10 µs na trigPin

6. Vreme koje je proteklo od generisanja ultrazvučnih impulsa do njihove detekcije na **echoPin** pinu nakon refleksije je moguće izmeriti korišćenjem funkcije pulseIn(echoPin, HIGH).

---

delayMicroseconds(µs): zaustavlja izvršavanje programa za definisani broj mikrosekundi.
pulseIn(pin, value): za pin koji predstavlja prvi argument funckije meri vreme u mikrosekundama koje taj pin provede HIGH ili LOW stanju, gde se željeno stanje definiše drugim argumentom funkcije - "value". Funkcija kao rezultat vraća izmereni broj mikrosekundi. Po potrebi je moguće dodati i treći argument funkcije "timeout" koji definiše koliko dugo će se čekati na završetak trajanja impulsa.

Naime, pri generisanju ultrazvučnih impulsa senzor automatski podešava stanje **echoPin** pina na HIGH. Kada detektor ultrazvuka detektuje reflektovani ultrazvučni talas, stanje **echoPin** pina se vraća na LOW. Rezultat funkcije pulseIn(echoPin, HIGH) je potrebno upisati u prethodno definisanu promenljivu **trajanje**.

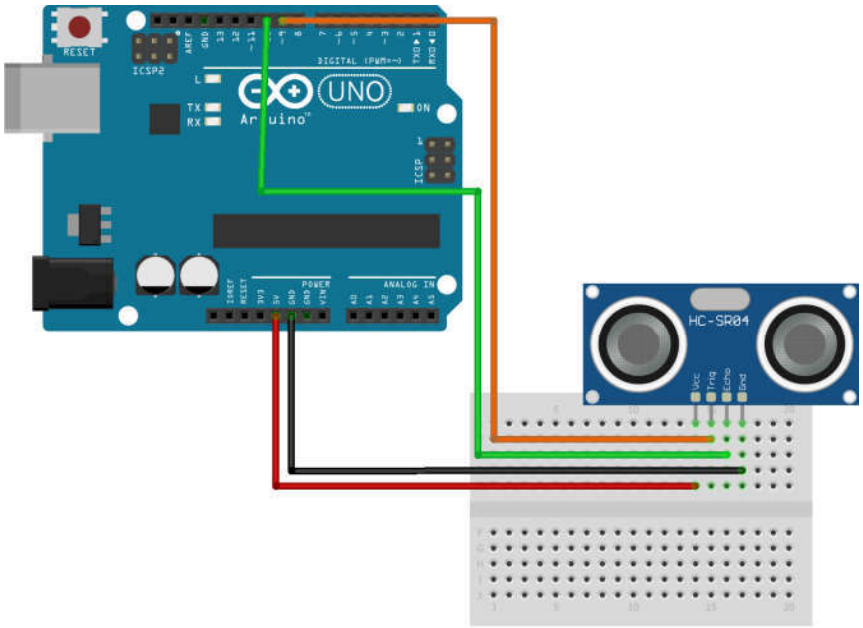
7. Pretpostavljajući da je brzina zvuka u vazduhu za trenutne parametre vazduha približno 340 m/s napisati jednačinu koja na osnovu vrednosti promenljive **trajanje** dolazi do rastojanja izraženog u centimetrima i rezultat jednačine upisati u promenljivu **rastojanje**.

NAPOMENA: Obratiti pažnju da je vreme u μs koje je proteklo od generisanja pulseva do njihovog povratka do modula nakon refleksije o prepreku zapravo vreme koje potrebno ultrazvučnom talasu da pređe **dva** rastojanja od modula do prepreke!

8. Ispisati rezultat za rastojanje izraženo u centimetrima na računaru u formi prikazanoj na Sl. 11.2.3. korišćenjem funkcija Serial.print() i Serial.println().

Sl. 11.2.3. Forma u kojoj je potrebno ispisati rezultate

9. Obezbezbediti da se merenje izvršava dva puta u sekundi dodavanjem funkcije delay(500).
10. Povezati HC-SR04 modul sa *Arduino* pločom prema Sl. 11.2.4.



Sl. 11.2.4. Šema prema kojoj je potrebno povezati *Arduino* i HC-SR04 modul

11. Sačuvati program kao *HC-SR04.ino*, povezati *Arduino* sa računarom i spustiti kod na ploču.
12. Otvoriti *Serial Monitor* i testirati rad programa postavljanjem prepreke ispred ultrazvučnog senzora rastojanja. Do koje maksimalne udaljenosti senzor pokazuje adekvatno očitavanje? Gde sve nalaze primene senzori koji koriste isti ili sličan princip merenja?
13. **Isključiti** *Arduino* iz računara.

Zadatak 11.2.1. – za samostalni rad

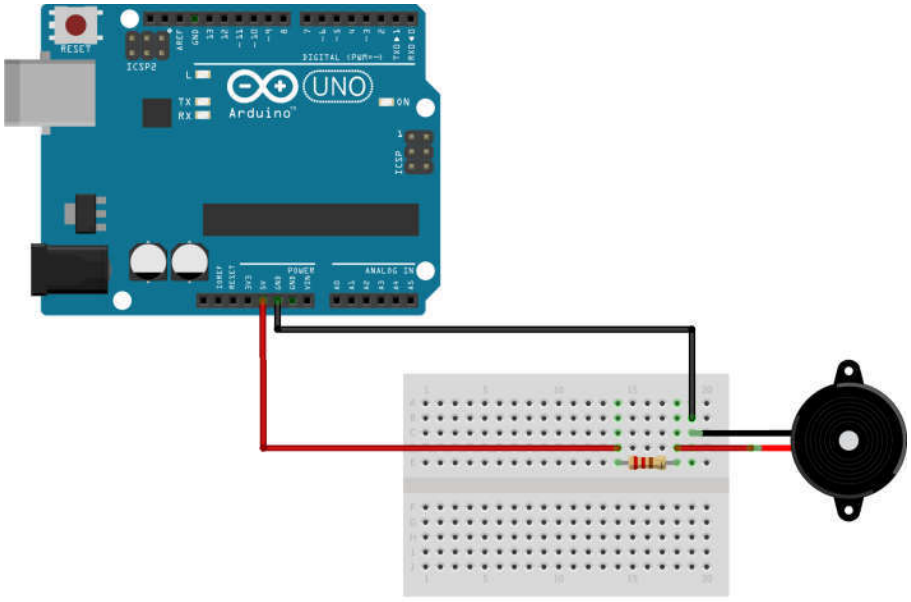
Sačuvati kopiju prethodnog programa pod novim imenom i modifikovati je tako da se vrednosti izmerenog rastojanja prikazuju na četvorocifrenom sedmosegmentnom displeju. Displej povezati prema Sl. 10.6.3, a **trigPin** promeniti na pin 11.

11.3 Arduino i zvučna signalizacija

Pored svetlosne signalizacije koja je već obrađena u okviru prethodnih poglavlja, projektima je često potrebno dodati i neki vid zvučne signalizacije. U najprostijem slučaju, od interesa je korišćenje *piezo buzzer*-a, komponente koja u sebi sadrži materijal sa piezoelektričnim svojstvima. Piezoelektrični efekat materijala rezultuje deformacijom materijala kada se on nađe u električnom polju. Ovaj efekat je poznat kao inverzni piezoelektrični efekat i omogućava generaciju zvuka i ultrazvuka usled periodičnih deformacija materijala koje su posledica promenljivog napona na *buzzer*-u. Direktni piezoelektrični efekat predstavlja generisanje naelektrisanja usled deformacije materijala i on se može koristiti pri detekciji ultrazvuka kao npr. kod modula HC-SR04.

U praksi je moguće naići na dve vrste *piezo buzzer*-a: aktivne i pasivne. Aktivni u sebi sadrže eletroniku koja automatski generiše periodičnu pobudu, te pobuđuje piezoelektrični materijal. Za njihov rad je stoga dovoljno priključiti ih na jednosmerni napon. Pasivni *piezo buzzer* zahteva da se pobuđuje periodičnim signalom kako bi se generisao zvuk.

1. Povezati *piezo buzzer* u šemu sa Sl. 11.3.1.



Sl. 11.3.1. Šema za povezivanje *piezo buzzer*-a sa *Arduino* pločom

NAPOMENA 1: Voditi računa o polarizaciji *piezo buzzer*-a. Kraj komponente koji je potrebno spojiti na pozitivan napon je često označen dužim metalnim kontaktom i/ili plusom na kućištu same komponente.

NAPOMENA 2: *Piezo buzzer* se obično povezuje na digitalni izlaz mikrokontrolera. U zavisnosti od same komponente, može se desiti da *piezo buzzer* poseduje veoma malu otpornost, te u slučaju direktnog povezivanja na digitalni izlaz povlači veću struju od maksimalne izlazne struje na digitalnom pinu *Arduino* ploče. Iz tog razloga, kako se *Arduino* ne bi spalio, poželjno je *piezo buzzer* povezivati na red sa otpornikom otpornosti 100-400 Ω, te na taj način ograničiti struju koju komponenta vuče.

2. Uključiti *Arduino* u računar i na osnovu ponašanja utvrditi da li *piezo buzzer* koji se koristi u zadatku spada u aktivne ili pasivne!
3. **Isključiti** *Arduino* iz računara.

U narednom zadatku će program iz sekcije 11.2. biti modifikovan tako da se u zavisnosti od izmerenog rastojanja od prepreke generišu različiti tonovi na *piezo buzzer*-u.

1. Otvoriti program koji je napisan u sekciji 11.2. i sačuvati njegovu kopiju pod novim nazivom. U kopiji programa dodati novu konstantu **tonePin** tipa int. Ovaj pin će biti korišćen za kontrolu *piezo buzzer*-a.
2. Na samom kraju loop sekcije, pre pozivanja delay(500) funkcije, potrebno je dodati sekciju koda koja će obezbediti da *piezo buzzer* svira: ton frekvencije 100 Hz ukoliko se prepreka nalazi na rastojanju između 15 i 20 cm, ton frekvencije 250 Hz ukoliko se prepreka nalazi na rastojanju između 10 i 15 cm, i ton frekvencije 400 Hz ukoliko se prepreka nalazi na rastojanju manjem od 10 cm. Generisanje pulseva podesive frekvencije je u *Arduino IDE* okruženju omogućeno korišćenjem funkcije tone(). Deo koda koji je potrebno dodati je prikazan na Sl. 11.3.2.
3. Otvoriti *Serial Monitor* i testirati rad kreiranog sistema menjanjem rastojanja prepreke od HC-SR04 modula.
4. **Isključiti** *Arduino* iz računara i ukloniti HC-SR04 modul sa šeme, a *piezo buzzer* ostaviti povezan.

```
if (rastojanje <= 20 && rastojanje > 15) {
  tone(tonePin, 100);
}
else if (rastojanje <= 15 && rastojanje > 10) {
  tone(tonePin, 250);
}
else if (rastojanje <= 10) {
  tone(tonePin, 400);
}
else {
  noTone(tonePin);
}
```

Sl. 11.3.2. Sekcija koda koja obezbeđuje sviranje različitih tonova na *piezo buzzer*-u u zavisnosti od rastojanja prepreke

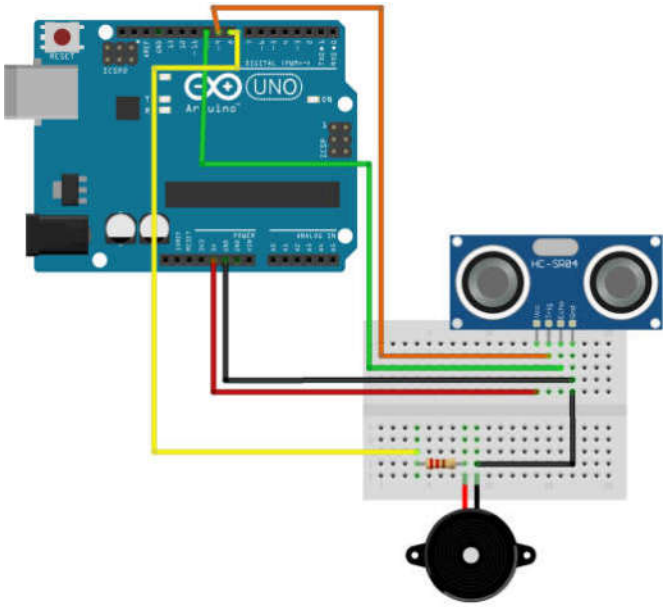
---

tone(pin, frequency, duration): generiše povorku četvrtki definisane frekvencije izražene u Hz na željenom pinu. Poslednji argunemt funkcije nije obavezan, a definiše trajanje tona u milisekundama.
noTone(pin): prekida generisanje povorke četvrtki ukoliko ton nije već prekinut u slučaju isteka definisanog trajanja (treći argument funkcije tone()).

Napomene:
- Funkcija tone() onemogućava korišćenje funkcije analogWrite() na pinovima 3 i 11.
- Nije moguće koristiti funkciju tone() za frekvencije manje od 31 Hz.
- Istovremeno generisanje različitih tonova na vše pinova nije moguće, već je potrebno pozvati funkciju noTone() pre poziva funkcije tone () na sledećem pinu.

---

5. Sačuvati program, povezati šemu prema Sl. 11.3.3, povezati *Arduino* sa računarom i spustiti kod na ploču.



Sl. 11.3.3. Šema prema kojoj se povezuju *buzzer* i HC-SR04 modul sa *Arduino* pločom

6. Otvoriti primer *toneMelody* koji se nalazi u sekciji *File»Examples»02.Digital* i sačuvati ga kao novi *Sketch* pod nazivom *toneMelody_test.ino*. U okviru ovog primera se poziva "pitches.h" koja sadrži definisane konstante za frekvencije nota od B0 do D#8.
7. Proučiti prikazani program.
8. Povezati *Arduino* sa računarom, spustiti kod na ploču i testirati rad sistema.
9. **Isključiti** *Arduino* iz računara.

Zadatak 11.3.1. – za samostalni rad

U dokumentu SuperMario.txt se nalaze note i trajanje nota za melodiju iz igrice Super Mario. Modifikovati kopiju prethodnog programa tako da *piezo buzzer* svira ovu melodiju **repetitivno**.

11.4 Korišćenje prekida

Nekada je od presudne važnosti detektovati određeni događaj i bez odlaganja izvršiti zadatu sekciju koda nevezano od toga gde se trenutno nalazimo u programu. Primer za to bi bilo praćenje važnih signala sa senzora na čije promene je potrebno brzo odreagovati ili pak izvršavanje određenih zadataka u tačno definisanim vremenskim intervalima. Nekada prosto nije poželjno opterećivati procesor konstantnim proveravanjem vrednosti na nekom digitalnom ulazu repetitivnim korišćenjem funkcije digitalRead() u loop sekciji programa, već je poželjno rešiti to na drugi način, a osloboditi procesor kako bi mogao da izvršava druge zadatke unutar petlje. Rešnje svih prethodno navedenih problema je moguće korišćenjem prekida (eng. *interrupt*). Prekid obezbeđuje da se na određeni definisani događaj odreaguje brzo i izvrši se određena sekcija koda koja se nalazi unutar željene funkcije koja se naziva *Interrupt Service Routine (ISR)*. Pri detekciji posmatranog događaja procesor prekida ono što je u tom trenutku radio, izvršava kod unutar ISR i nakon toga se vraća prethodno započetim zadacima u glavnom delu koda.

U zadatku koji sledi će biti realziovan program koji će obavljati istu funkciju kao i program sa Sl. 10.2.3, tj. funkciju uključivanja ugrađene svetleće diode sve dok je kapacitivni senzor dodira dodirnut. Međutim, ovaj program će koristiti prekid za praćenje stanja kapacitivnog senzora dodira, te će procesor biti slobodan da radi druge zadatke unutar loop dela koda!

1. Otvoriti novi *Sketch* pritiskom na *File»New* i sačuvati ga pod nazivom *TouchLED_interrupt.ino*. U programu najpre definisati konstantu tipa int pod nazivom **touchPin** i njenu vrednost podesiti na 2 (ovo će biti pin na koji će biti povezan izlaz kapacitivnog senzora dodira). Zatim kreirati promenljivu volatile byte pod nazivom **state** i inicijalizovati je na LOW stanje.
2. U setup sekciji koda inicijalizovati LED_BUILTIN pin kao izlazni, i **touchPin** kao INPUT_PULLUP. Zatim pozvati funkciju sa argumentima attachInterrupt(digitalPinToInterrupt(touchPin), blink, CHANGE).

---

attachInterrupt(digitalPinToInterrupt(pin), ISR, mode): funkcija koja uključuje prekid. Prvi argument ove funkcije je broj, 0 ili 1 koji predstavlja broj prekida, a ne broj pina čije se stanje posmatra! Za *Arduino UNO R3* ploču brojevi 0 i 1 se odnose na pinove 2 i 3 respektivno. Ipak, nije preporučljivo koristiti vrednosti 0 i 1 direktno u funkciji, već je bolje kosrititi funkciju digitalPinToInterrupt(pin) koja kao argument prima željeni pin (2 ili 3 za UNO) i prevodi ga u odgovarajući broj prekida. Drugi argument funkcije attachInterrupt() predstavlja naziv funkcije koja će se izvršiti kada do prekida dođe. Treći i poslednji argument funkcije predstavlja mod koji definiše kakva akcija na željenom pinu će dovesti do prekida. Dozvoljeni modovi su: LOW, CHANGE, RISING i FALLING. U slučaju moda CHANGE koji se koristi u zadatku, potrebno je inicijalizovati *Interrupt* pin kao INPUT_PULLUP.
detachInterrupt(digitalPinToInterrupt(pin)): isključuje željeni prekid.

---

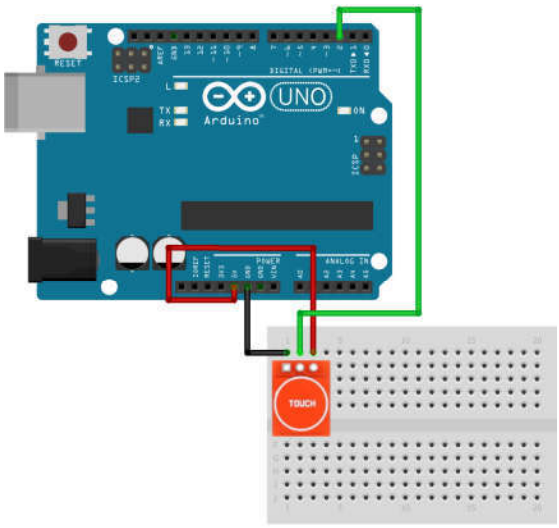
3. Glavnu petlju ostaviti praznu, a definisati novu funkciju naziva **blink** koja ne prima argumente i ne vraća rezultate - void blink(). Ova funkcija će biti ISR i izvršavaće se svaki put kada dođe do prekida.
4. U telu ove funkcije obezbediti da se pri svakom ulasku u ISR **blink** vrednost promenljive **state** menja - state = !state, a zatim upisati novu vrednost promenljive **state** na digitalnom izlazu ugrađne svetleće diode korišćenjem funkcije digitalWrite().

Konačan izgled programa je prikazan na Sl. 11.4.1. Obratiti pažnju da unutar loop dela koda ne postoje instrukcije, tj. da je sada u loop delu koda moguće izvršavati druge stvari po potrebi!

5. Sačuvati promene u programu.
6. Povezati kolo prema Sl. 11.4.2, povezati *Arduino* sa računarom i spustiti kod na ploču.
7. Testirati rad programa dodirivanjem kapacitivnog senzora dodira i posmatranjem reakcije ugrađene svetleće diode na ploči. Ukoliko je sve urađeno prema instrrukcijama, ne bi trebalo da bude razlike u radu programa sa Sl. 10.2.3. i programa sa Sl. 11.4.1. uz komentar da novi program izvršava promenu stanja ugrađene svetleće diode samo kada dođe do promene stanja na pinu 2 *Arduino* ploče, a nema ništa u loop sekciji koda!
8. **Isključiti** *Arduino* iz računara, i rastaviti kolo sa protoborda.

```
const int touchPin = 2;
volatile byte state = LOW;

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(touchPin, INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(touchPin), blink, CHANGE);
}

void loop() {
}

void blink() {
  state = !state;
  digitalWrite(LED_BUILTIN, state);
}
```

Sl. 11.4.1. Program za čitanje sa kapacitivnog senzora dodira i uključivanje svetleće diode korišćenjem prekida



Sl. 11.4.2. Šema povezivanja kapacitivnog senzora dodira sa *Arduino* pločom u slučaju korišćenja prekida

NAPOMENA: Pri pisanju ISR funkcija treba poštovati sledećih nekoliko veoma važniih pravila:

- ISR mora biti veoma kratkog trajanja! Pri prekidu se prekida izvršavanje glavne petlje što ne bi trebalo dugo da traje!
- ISR nema ni ulazne ni izlazne promenljive.
- Unutar ISR ne koristiti funkcije delay(), Serial.println(), Serial.read().
- Promenljive koje se koriste za razmenu podataka između ISR i glavnog programa obavezno deklarisati kao volatile kako bi kompajler znao da ove promenljive mogu promeniti vrednost u svakom trenutku i kako ih ne bi eliminisao kao "nekorišćene" radi čuvanja memorije.

Zadatak 11.4.1. – za samostalni rad

Realizovati program koji omogućava trigerovanu akviziciju na *Arduino* ploči korišćenjem prekida. Potrebno je da akvzicija počne kada se detektuje silazna ivica na pinu 2 *Arduino* ploče. Za detekciju silazne ivice podesiti stanje pina 2 kao INPUT. Silaznu ivicu generisati prebacivanjem kraja kratkospojnika sa 5 V pina na GND.

11.5 Biblioteka TimerOne *za kontrolu prvog hardverskog tajmera*

Mikrokontroler na *Arduino* ploči poseduje tri hardverska tajmera, *Timer0*, *Timer1* i *Timer2*. Tajmeri su odgovorini za neke od ključnih funkcionalnosti koje su korišćene u programima koji su prethodno demonstrirani. Tako je, na primer, osmobitni tajmer *Timer0* direktno odgovoran za rad funkcija za merenje vremena delay(), millis() (funkcija koja vraća trenutno vreme u miliskundama od početka izvršavanja programa) i micros() (funkcija koja vraća trenutno vreme u mikrosekundama od početka izvršavanja programa sa korakom od 4 μs). Ovaj tajmer je povezan na pinove 5 i 6 *Arduino UNO R3* ploče i takođe je odgovoran za PWM na ovim pinovima. Osmobitni tajmer *Timer2* obezbeđuje precizno podešavanje frekvencije pri korišćenju funkcije tone() i odgovoran je za PWM na pinovima 3 i 11 *Arduino UNO R3* ploče. Tajmer sa najvećom rezolucijom od čak šesnaest bita se je *Timer1* i on je odgovoran za PWM na pinovima 9 i 10 *Arduino UNO R3* ploče. *Timer1* se koristi unutar biblioteke za upravljanje *Servo* motorom, a inače je slobodan za posebne primene. Biblioteka koja olakšava korišćenje ovog tajmera je TimerOne biblioteka koju je moguće skinuti sa sledećeg linka: https://github.com/PaulStoffregen/TimerOne. Ova biblioteka omogućava dve ključne funkcionalnosti:

- potpunu kontrolu nad generacijom PWM signala na pinovima 9 i 10 sa velikom rezolucijom faktora ispune i frekvencije; i
- periodično generisanje prekida, svaki put kada tajmer izbroji određeni broj mikrosekundi. Ova funckionalnost je posebno važna kada je potrebno veoma precizno definisati frekvenciju odabiranja pri analognoj akviziciji.

Program koji će biti obrađen treba da obezbedi akviziciju sa precizno definisanom frekvencijom odabiranja korišćenjem prekida.

1. Otvoriti novi program i u njega uključiti biblioteku *TimerOne.h* izborom opcije *Sketch»Include Library»Add .ZIP Library*.
2. Deklarisati dve promenljive volatile int **voltage** i volatile byte **state** i inicijalizovati ih na 0.
3. U setup sekciji koda započeti serijsku komunikaciju, inicijalizovati prvi tajmer na 100 ms korišćenjem funkcije Timer1.initialize(1000000) i aktivirati prekid za ovaj tajmer korišćenjem funkcije Timer1.attachInterrupt(timerIsr).

Timer1.initialize(value): inicijalizuje prvi hardverski tajmer. Argument funkcije je perioda tajmera izražena u mikrosekundama.
Timer1.attachInterrupt(function, period): poziva ISR čiji je naziv prvi argument funkcije na određeni broj mikrosekundi definisan drugim argumentum funkcije. Ukoliko se drugi argument funkcije izostavi, ISR se poziva na osnovu argumenta funkcije Timer1.initialize().

---

4. Zatim kreirati ISR pod nazivom **timerIsr**. Unutar **timerIsr** najpre obezbediti da pri svakom ulasku u taj deo koda promenljiva **state** uzme vrednost 1 čime se označava da je ISR izvršena. Zatim pročitati vrednost na analognom ulazu A0 i upisati je u promenljivu **voltage**.

5. Unutar glavne petlje pri svakoj iteraciji proveravati vrednost promenljive **state**. Samo ukoliko je **state** jednako 1, na serijskom portu ispisati pročitanu vrednost sa A/D konvertora i vratiti stanje promenljive **state** na 0.

NAPOMENA: Na ovaj način se obezbeđuje da se čitanje analognog napona izvršava u definisanim vremenskim inetrvalima, dok se ispisivanje vrednosti izvršava u pauzi između čitanja samo onda kada postoji pročitana vrednost. Ovako definisan kod, prikazan na Sl. 11.5.1, poseduje značajno precizniju frekvenciju odabiranja od koda koji koristi funkciju delay() unutar glavne petlje, jer tada vreme između dva uzorkovanja zavisi od vremena izvršavanja svih ostalih funkcija u glavnoj petlji! Ipak, perioda sa kojom se poziva ISR ne sme biti isuvše mala kako bi bili sigurni da će kod unutar ISR stići da se izvrši.

```
#include <TimerOne.h>

volatile int voltage = 0;
volatile byte state = 0;

void setup()
{
  Serial.begin(9600);
  Timer1.initialize(1000000);
  Timer1.attachInterrupt(timerIsr);
}

void loop()
{
  if(state){
    Serial.println(voltage);
    state = 0;
    }
}

void timerIsr()
{
    state = 1;
    voltage = analogRead(A0);
}
```

Sl. 11.5.1. Akvizicija sa precizno definisanom frekvencijom odabiranja

6. Sačuvati program kao *Timer_Acq.ino*, povezati *Arduino* sa računarom i testirati program. Rezultate čitanja analognog napona analizirati korišćenjem *Serial Monitor*-a. Spustiti kod na ploču nekoliko puta sa različitim verdnostima za periodu sa kojom se izvršava čitanje sa A/D konvertora.
7. **Isključiti** *Arduino* iz računara.

11.6 Upravljanje Servo motorom

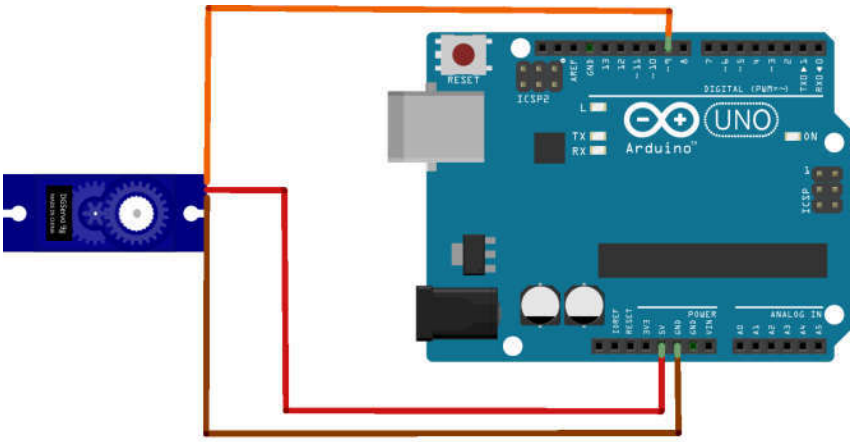
Servo motori su motori kod kojih je moguće precizno kontrolisati poziciju osovine slanjem PWM signala sa *Arduino* ploče. Precizna kontrola pozicije je posledica povratne sprege koju motor poseduje. Pulsevi koji služe za kontrolu *Servo* motora ne traju dugo, obično od oko 0.5 do oko 2 ms. Definisani opseg trajanja za konkretnu komponentu se stoga skalira na opseg uglova od 0 do 180 stepeni. Pored pina za kontrolu pozicije često žute ili narandžaste boje, *Servo* motor poseduje i pinove za napajanje Vcc i Gnd, najčešće crvene i crne(ili braon) boje respektivno. Jednostavna kontrola *Servo* motora je omogućena korišćenjem funkcija iz *Servo.h* biblioteke koja dolazi već inkorporirana u *Arduino IDE*.

1. Otvoriti primer *Sweep* izborom opcije *File»Examples»Servo* analizirati strukturu programa.

Servo myservo: kreira *Servo* objekat naziva **myservo** koji omogućava kontrolu *Servo* motora.
myservo.attach(pin, min, max): prvi argument ove funkcije predstavlja pin za kontrolu na koji je *Servo* povezan. Preostala dva argumenta nisu obavezna, a predstavljaju minimalno i maksimalno trajanje poslatih pulseva koji odgovaraju rotaciji od 0 i 180 stepeni respektivno. Ukoliko se ovi argumenti izostave, šalju se standardne vrednosti od 544 µs trajanja za 0 stepeni i 2400 µs trajanja za 180 stepeni.
myservo.write(angle): funkcija koja služi za kontrolu ugla *Servo* motora. Argument funkcije je ugao izražen u stepenima koji funkcija prevodi u odgovarajuće trajanje poslatog impulsa.

---

2. Povezati *Servo* na *Arduino* prema šemi sa Sl. 11.6.1.
3. Povezati *Arduino* sa računarom i spustiti kod na ploču.



Sl. 11.6.1. Šema povezivanja *Servo* motora sa *Arduino* pločom

4. **Isključiti** *Arduino* iz računara, ali šemu ostaviti sastavljenu.
5. Otvoriti novi program u kome će prema principu rada programa iz *Zadatka 11.1.1.* biti realizovan program koji će omogućiti kontrolu pozicije *Servo* motora slanjem poruka sa računara. Najpre u program uključiti biblioteku *Servo.h* i definisati promenljive **inputAngle** i **angle** tipa int koje će služiti za čuvanje trenutno pročitane vrednosti i krajnje pročitane vrednosti respektivno, Sl. 11.6.2. Zatim kreirati *Servo* objekat naziva **myservo**.

```
#include <Servo.h>

int inputAngle = 0;
int angle = 0;

Servo myservo;
```

Sl. 11.6.2.

6. U okviru setup sekcije programa započeti serijsku komunikaciju sa računarom i definisati pin 9 kao pin za kontrolu *Servo* motora, Sl. 11.6.3.

```
void setup() {

  Serial.begin(9600);
  myservo.attach(9);


}
```

Sl. 11.6.3.

7. Glavnu petlju ostaviti praznu, a u serialEvent() funkciji obezbediti čitanje nadolazećih karaktera sa serijskog porta i njihovu konverziju u brojeve od 0 do 9, Sl. 11.6.4.
8. Nakon što je čitanje završeno, obezbediti slanje vrenosti ugla *Servo* motoru i ispisivanje iste vrednosti na računaru radi provere.

```
void loop() {

}


void serialEvent() {
  while (Serial.available()) {

    char inChar = (char)Serial.read();
    if (inChar == '\n') {
      angle = inputAngle;
      inputAngle = 0;
      Serial.println(angle);
      myservo.write(angle);
    }
    else if (inChar >= '0' && inChar <= '9') // is inChar a number?
    {
      inputAngle = inputAngle * 10 + inChar - '0'; // yes, accumulate the value
    }

  }
}
```

Sl. 11.6.4.

9. Sačuvati program kao *servoSerial.ino*, povezati *Arduino* na računar i spustiti kod na ploču.
10. Korišćenjem *Serial Monitor*-a testirati rad programa.
11. **Isključiti** *Arduino* iz računara i rastaviti šemu.

NAPOMENA: Korišćenje biblioteke *Servo.h* onemogućava PWM na pinovima 9 i 10 *Arduino* ploče!