ICM Summer of Code Day 2: PWM, serijski protokoli. Vanjske jedinice

Fokus današnje radionice biti će korištenje raznih Arduino pinova u modu **izlaza**. Podsjetimo se nekih od glavnih naučenih funkcija s prethodne radionice:

```
void setup(); // funkcija koja se pokreće prilikom paljenja Arduina
void loop();  // funkcija koja se pokreće po završetku setup-a,
              // a po završetku se ponovno pokreće - postiže se efekt
             // programske petlje
```

Na prethodne dvije funkcije ujedno i nemamo utjecaj: one će biti izvršene bez obzira koristili ih mi ili ne, ali ništa ne spriječava programera da u neku od funkcija (ili obje) ostavi praznom.

Neke od korištenih funkcija za serijsku komunikaciju:

```
Serial.begin(baud);   // uspostavlja serijsku vezu s argumentom
brzine
                     // Arduino podržava brzine do 2Mbps

Serial.print(varijabla_proizvoljnog_tipa);
Serial.println(varijabla_proizvoljnog_tipa); // funkcije za slanje
                                             // podataka preko serije:
                                             // println ujedno šalje i
                                             // novi red

Serial.available() // vraća broj nepročitanih bajtova (znakova) iz
                  // serijskog stoga (buffera, spremnika)

Serial.read()       // čita jedan znak sa serijskog buffera
Serial.readString() // čita sve znakove na bufferu u obliku Stringa
```

Funkcija za inicijalizaciju arduino pinova:

```
pinMode(broj_pina, mod);
```

Gdje *mod* može biti:
`INPUT:` za ulaz u modu visoke impendacije - očekuju se smetnje i problemi ukoliko se pin nalazi u situaciji da nije spojen na nikakav izlaz - preporuča se konstrukcija pullup ili pulldown-a s otpornikom

`INPUT_PULLUP:` za ulaz u modu internog pullup sklopa (Arduino nema ugrađeni interni pulldown), kojeg preferiramo koristiti radi jednostavnosti strujnog kruga, ali uz trik da je pin prilikom nespajanja u visokoj (`HIGH`) razini (što i sama riječ pull**up** naglašava)
`OUTPUT:` za ulaz u modu izlaza.
Neke od funkcija za manipuliranje ili čitanje s pinova:

```
digitalWrite(pin, state) // izlaz iz pina na 0V (state=LOW), 5V
(HIGH)
digitalRead(pin) // čitanje sa digitalnog pina

analogRead(pin) // čitanje sa analognog pina
analogWrite(pin, value) // zapis analogne vrijednosti (0, 5V i
između,
                        // naznačene vrijednostima value=0 do 255)
                        // na DIGITALNI pin (3, 5, 6, 9, 10, 11)
```

Na velikom dijelu današnje radionice bavit ćemo se analognim pinovima pa će gornja zavrzlama postati jasnija - voditelj radionice će detaljno objasniti što se i zašto događa.

Neke od funkcija ili korisnih koncepata za manipuliranje vremenom:

```
delay(t); // zaustavlja program na t-milisekundi (1000ms = 1s)
delayMicroseconds(t); // zaustavlja program na t-mikrosekundi
                      // (1000000us=1s). Oprez, valjano za
                      vrijednosti
                      // do nekoliko tisuća mikrosekundi, ne više
while(1); // čest trik kako zaustaviti program
```

Svojstva (sintaksa) ostalih koncepata u C-u (if, for, while) nisu specifične za Arduino nego za programski jezik C. Od polaznika se očekuje da njihovo načelno korištenje poznaju, a detalje je moguće raspraviti s voditeljem radionice.

Analogni pinovi, analogne vrijednosti:

Kao i na prethodnoj radionici, uvod u navedene koncepte biti će obrađen kroz zadatke, a teorijsku osnovu pojasniti će voditelj radionice.

Zadatak 1: Konstruirati naponsko djelilo koristeći dva otpornika (jednaka ili različita) spojena serijski, jednu stranu povezati na 5V, drugu na GND, a vezu između dva otpornika povežite na proizvoljan analogan pin. U razmaku od 10ms periodički ispisivati (npr. `Serial.println(vrijednost)`) očitanu vrijednost pomoću funkcije `analogRead(pin)`.

Vrijednost *pin*-a je integer kojeg možete upisivati u obliku konstante: `int pin = A1;` (ili A0, A2, A3, A4, A5...)

Ukoliko na radionici budete imali potenciometar / trimer (imamo ih u jako ograničenim količinama), pokušajte umjesto naponskog djelila koristit njega. Ispis umjesto Serial Monitorom prikažite Serial Plotterom (Tools - Serial Plotter). Raspravite s voditeljem radionice kako rade (fizički) ti sklopovi. Napon možete izmjeriti i dostupnim multimetrom.

Zadatak 2: LED-icu povežite na jedan od pinova koji podržavaju "analogan" izlaz- Kontrolirajte intenzitet LED-ice na način da:

a) Koristite funkciju `digitalWrite()`, a intenzitetom upravljajte učestalošću paljenja - npr imajte jednu *for* petlju u kojoj ćete 33% (ili konfigurabilno) vremena LED-icu držati upaljenom, a 66% vremena ugašenom. U nastavku je prikazan gore navedeni primjer gdje svaki znak niza označava jednu iteraciju petlje, znak **-** predstavlja ugašenu LED-icu, a **O** upaljenu: **O--O--O--O--O--O--O--O--O--O--**.... Napomena: uvjerite se da **ne radite** nešto kao **OOOOOOOOOO---** ispisom stanja preko serijskog monitora - stanja moraju biti "isprepletena" da bi efekt bio što bolji. Možete između iteracija dodavati i *delay* proizvoljne duljine

b) Koristite funkciju `analogWrite()`.

Zadatak 3: Koristeći funkcije `Serial.readString()` i `ime_mojeg_stringa.toInt()`, `map()` –njihovo korištenje možete naći na internetu (dovoljno je upisati npr. "Arduino map function" u google), preko serijskog monitora zadajte postotak intenziteta LED-ice spojene na nekom pinu. Nijednu od navedenih funkcija ne morate koristiti da bi ostvarili funkcionalnst, ali je to poželjno jer će vama olakšati stvari - alternativno riješite zadatak na način na koji znate, a kasnije raspravite s voditeljem radionice kako bi to napravili na drugi (ne nužno i bolji) način.

Zadatak 4: Koristite potenciometar kontrolu intenziteta LED-ice. Kontrolu ostvarite

a) neprogramski (neovisno o programu kojeg ste uploadali na Arduino)
b) Programski

Ukoliko još nemate potenciometar, preskočite zadatak. U slučaju programske implementacije, vrijednosti potenciometra ispisujte na Serial Monitor (ili plotter) - vrlo je bitno imati naviku koristiti ga za dijagnozu napisanih programa, jer ne postoji drugi mehanizam kako bi utvrdili što i zašto ne radi (osim ponekad ako ste fizički nešto krivo spojili……).

Serijski protokoli:

Serijski protokol je skup pravila za komunikaciju: kako jedan uređaj komunicira s drugim. Na svu sreću, gotovo nikad ne moramo brinuti o pravilima komunikacije - obično se rad svodi na to da je potrebno znati pravila povezivanja žica na uređaj, i to je to. Programski dio u većini slučajeva ostavljamo ugrađenim bibliotekama iz kojih samo pozivamo par funkcija.

UART

UART (engl. Universal Asynchronous Receiver [and] Transmitter) je danas vrlo raširen serijski protokol, i najnapredniji od onih koje ćemo obraditi u sklopu radionica. Samo ime naglašava da je samim protokolom moguće komunicirati u oba smjera - koristili ste UART za komunikaciju između računala i Arduina. UART sam po sebi za komunikaciju zahtijeva dvije podatkovne žice - Rx (engl. Receiver) i Tx (engl. Transmitter). Kako bi dva uređaja mogli komunicirati, potrebno je isprepleteno povezati njihove Rx i Tx veze (Rx prvog uređaja na Tx drugog uređaja, i Tx prvog uređaja na Tx drugog uređaja). Ovo je intuitivno jer možete zamisliti da podatak s jednog uređaja na Tx vezi (Transmitter: hrv. Odašiljač) putuje prema Rx vezi cijljnog uređaja na kojem on prima podatak. Također, nije potrebno voditi o brigu o sinkronizaciji slanja i primanja - svaki uređaj može primati i odašiljati istovremeno, čak i ako istog trenutka ne pročitamo podatak. Nepročitani podaci spremaju se u međuspremnik (engl. Serial Buffer) koji se prazni čitanjem podataka. Svaki uređaj posjeduje buffer određene veličine - Arduino ima buffer veličine 68B (Bajta, znakova, 1 znak - char = 1B). Arduino digitalni pinovi 0 i 1 su direktno spojeni na hardverski sklop Arduina za komunikaciju UART-om - to je isti onaj kojem pristupamo USB-om.

Zadatak 5. Povežite Rx i Tx veze vašeg Arduina, i Arduina vašek kolege ili kolegice do vas. Kao Arduino program uploadajte program koji u `setup()` i `loop()` nema ništa. Otvorite Arduino Serial Monitor na oba računala i probajte nešto poslati. Jednu od dvije povezane žice maknite. Što sad opažate? Napomena: kada povezujemo dva uređaja nekim signalom, potrebno je da imaju jednaku referentnu nulu - GND. Stoga, iako su samo dvije podatkovne žice, potrebno je dodati i treću koja nema “pametnu” ulogu, ali je uvijek preporučljivo spojiti je - u većini slučajeva vam bez nje neće funkcionirati.

Zadatak 6. Napravite program kojim će Arduino komunicirati s vašim susjedskim Arduinom. Napravite sustav u kojem Arduino pali LED diodu na jednu sekundu po primitku poruke **1** (kao integer). Po gašenju LED diode pošaljite preko serije podatak **1**. susjednom uređaju. Kad napravite ovaj zadatak, javite voditelju radionice da raspravite o tome.

Ostali serijski protokoli biti će obrađeni tek u teoriji, kako nismo u mogućnosti osigurati dovoljan broj modula. Od svih protokola koje ćemo obraditi UART je najmoćniji (komunikacija u oba smjera, teoretski neograničene brzine komunciranje - Arduino 2Mb/s). Jedna od mana jest da svaki par uređaja mora imat sklop za UART - Arduino ima samo jedan, tako da ne može

komunicirati s 2 uređaja istovremeno preko UART-a (može, ali ograničeno i uz nepraktična rješenja).

I²C

I²C (I2C, engl. Inter Integrated Circuit) je protokol koji omogućava komunikaciju velikog broja uređaja na spojenih na istoj žici. Uređaji imaju SDA (Data) i SCL (Clock) veze te se spajaju jedno s drugim (ne unakrsno kao kod UART-a). Uređaji su u mogućnosti komunicirati preko iste žice istovremeno jer svaki uređaj ima svoju adresu (koju ili specificirate ili znate od svakog uređaja - zato je često teže spojiti više **identičnih** uređaja odjednom, ali postoje metode da se to riješi). Mana I2C protokola je što je obično jedan uređaj taji koji prikuplja podatke (engl. Master) a ostali mu uslužuju svoje informacije (engl. Slave) te vrlo spor prijenos informacija u odnosu na druge vrste protokola.

SPI

SPI (engl. Serial Peripheral Interface) je protokol vrlo sličan I2C-u, uz veće brizne prijenosa podataka i ponešto drugačiji sustav adresiranja. Veze su MISO (engl. Master In Slave Out), MOSI (engl. Master Out, Slave In), SCL (Clock) i CS (Chip select). Uređaji se spajaju jedan na jedan (a ne unakrsno kao kod UART-a), a Chip Select je veza koja mora biti posebna za svaki uređaj, i služi kako bi javili konkretnom uređaju da počne s radom. Zahtijeva nešto više veza od I2C (tri veze se dijele + CS za svaki uređaj posebno), ali zato manje od UART-a i ne zahtjeva poseban sklop po vezi s uređajem.

Sa I2C, SPI sabirnicama ćete se implementacijski upoznati po potrebi, ovisno o vanjskoj jedinici koju budete dobili za vježbu.

Vanjske jedinice:

Kako na vježbama nemamo resurse da bi svima podijelili jednake vanjske jedinice, zadatke ćete dobiti individualno od strane voditelja radionice (npr. implementirati parking senzor pomoću HC-SR04 senzora, pokrenuti servo motor, napraviti igru pomoću digitalnog zaslona, prikazati temperaturu očitanjem s digitalnog ili analognog senzora.