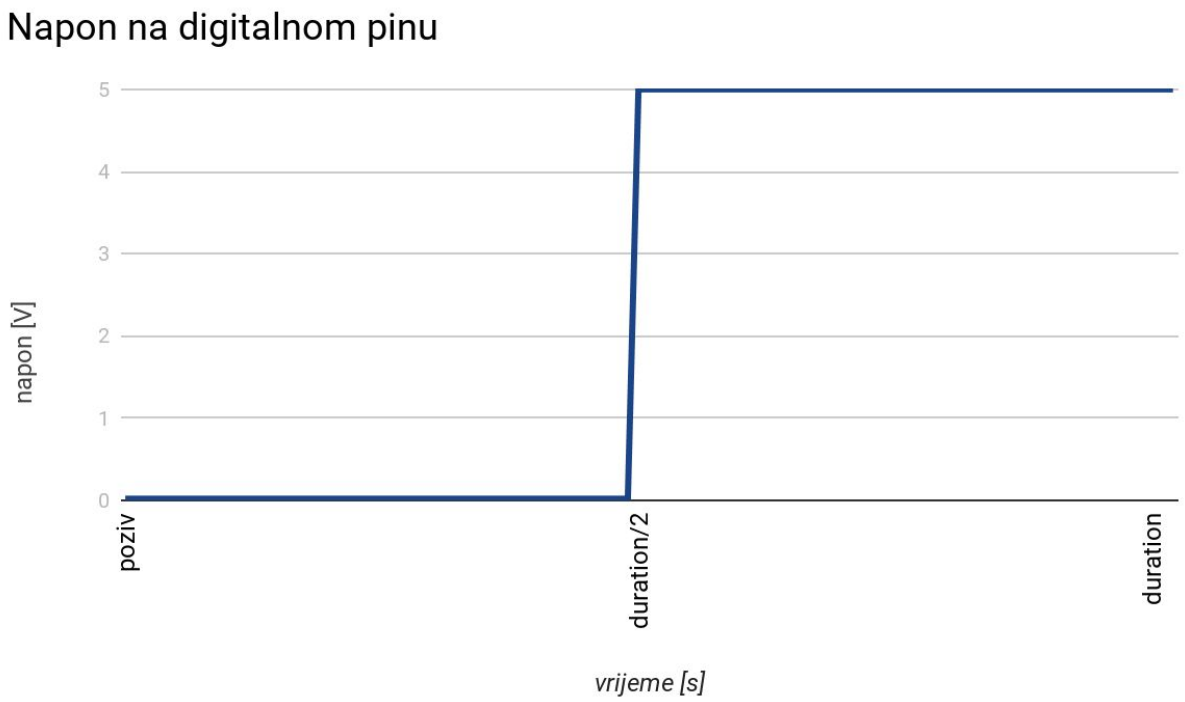
Funkcije 4: vanjske jedinice

Zadano je *5* zadataka koje je poželjno rješiti primjenom *funkcija*, koje su detaljnije obrađene na prethodnim radionicama. Svaki zadatak sastoji se od nekoliko, nekad manje očitih, problema koje pokušajte rješiti traženjem dokumentacije na internetu ili jednostavno pitajte. Rješenja zadataka pohranjujte u poseban direktorij koji ćete napraviti u svrhu ove vježbe, a svaki zadatak neka bude u imenovan u obliku *zadatak_x.ino*. Slobodno koristite funkcije sa zadnjih radionica koje ste napravili sami, a možete koristiti i funkciju koja služi za učitavanje podataka sa serije koje je unio korisnik. Izvorni kod funkcije dostupan je na linku: pastebin.com/rph1FEk4.

Zadatak 1: Implementiraj funkciju zadanog prototipa:

```
void singleWave(int pin, float duration);
```

koja pali i gasi LED diodu na pinu `pin` na način na koji je opisan na slici:



Vrijeme u varijabli `duration` izraženo je u **milisekundama**. Za postizanje ovog koristi funkciju *delay()*. Kako je ulazni argument u funkciju *delay()* bezpredznačni integer, a ne *float* s kojim radimo u funkciji, poželjno je castati. Isprobaj svoju funkciju višestrukim pozivom s parametrom `duration` u iznosu od dvije sekunde.

Zadatak 2: implementiraj funkciju prototipa:

```
void playTone(float frequencyHz, float duration, int pin);
```

Koja uzastopno zove funkciju *singleWave* dok god ne prođe vrijeme *duration*, izraženo u milisekundama. Argument *frequencyHz* potrebno je modificirati na način predstavlja broj puta u sekundi koliko bi se funkcija *singleWave* trebala pozivati, bez obzira što je broj poziva određen argumentom *duration*. Primjer:

```
playTone(200, 10000, 13);
```

Ovaj poziv funkcije znači da će se funkcija *singleWave* pozivati s učestalošću (**frekvencijom** :) od dvjesto puta u sekundi, te da će biti pozvana ukupno oko 2000 puta. Matematičkije:

$t_{ukupno} = 10000ms = 10s$ (tokom deset sekundi)

$f = 200Hz = 200s^{-1}$ (frekvencijom od 200 poziva u sekundi)

$n_{ukupno} = ?$ (tražimo ukupan broj poziva)

$f = \frac{1}{T_{period}} = \frac{n_{uk}}{t_{ukupno}} \quad \Rightarrow \quad n_{ukupno} = f * t_{ukupno}$

$n_{ukupno} = 200s^{-1} * 10s = 2000$

Očekivano je da će broj poziva varirati za do 5%, kako ne možemo utjecati na duljinu poziva potprograma, frekvenciju oscilatora; sklopa koji definira kada Arduino izvodi koju operaciju, u ulozi generatora takta (engl. *clock*, od kuda potječe i možda vam poznatiji termin *overclock*). Spoji jednu nogu *piezo* zvučnika na pin 13, a drugu na GND od Arduina. Što primjećuješ? Ekperimentiraj s parametrima funkcije!

Zadatak 3: Na svom stolu si dobio ultrazvučni senzor tipa HC-SR04. Prouči njegovu funkcionalnot iz njegovog tvorničkog datasheeta dostupnog na linku: https://goo.gl/aJzJPE. Prokomentiraj s voditeljem radionice kako se radi s ovim senzorom.

Implementiraj funkciju sa slijedećim prototipom:

```
unsigned long ultrasonicTime(int triggerPin, int dataPin);
```

Koja vraća broj mikrosekundi trajanja visoke razine povratnog signala sa senzora. Funkcija može mjerenje ponoviti nekoliko puta i vratiti srednju vrijednost mjerenja ne bi li rezultat bio što bolji. Ako radite ponavljanje mjerenja, nemojte više od deset puta. Ispišite vrijednost na Arduino Serial monitoru! Kako se pročitana vrijednost mijenja ovisno o udaljenosti predmeta od senzora?
Hint: pogledaj što radi ugrađena funkcija pulseIn()

Zadatak 4: Implementirajte funkciju prototipa:

```
unsigned long distance_cm();
```

Koja vraća udaljenost objekta detektiranog ultrazvučnim senzorom. Vrijednost neka vraća u centimetrima. U funkciji pretpostavite da je *trigger* pin spojen na pin 13, a *echo* na pin 12. Ovu funkciju implementirajte tako da ona koristi ranije napisanu funkciju *ultrasonicTime*, s tim da podatke iz te funkcije morate obraditi (u ovom slučaju samo pomnožiti ili podijeliti konstantom koju sami odredite ne bi li imali što preciznije mjerenje).

Zadatak 5: ukoliko do sada niste, implementirajte do kraja funkciju za prikaz broja na 7 segmentnom displayu. U nastavku je zadatak iz prošle radionice. Niz *segmenti* dostupan je na linku: https://pastebin.com/C6AhwJW5

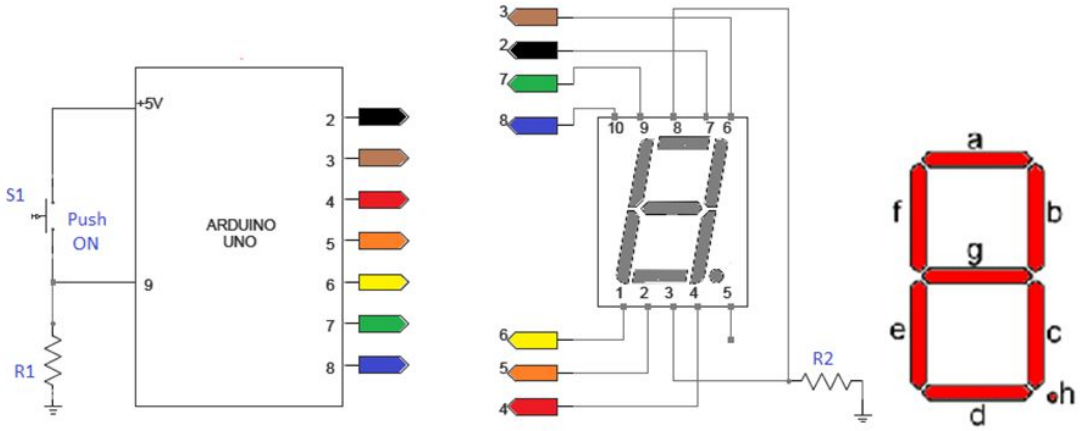
Prouči kako funkcionira 7-segmentni display. Pokušaj upaliti njegove segmente samo spajanjem na izlaze GND i 5V. Implementiraj funkciju zadanog prototipa

```
void setNumber7Segment(int number);
```

koja prikazuje broj zadan argumentom
Također, u nastavku je zadan i niz binarnih brojeva koji označavaju koji segmenti se pale za koju brojku. Npr. ukoliko želimo prikazati broj 3, u nizu *segmenti[3]* nalazi se binaran zapis `B01001111` koji označava koji segmenti moraju biti upaljeni, počevši od najnižeg bita (skroz desnog): segment A mora biti upaljen, segment B mora biti upaljen, segment C mora biti upaljen, segment D mora biti upaljen te segment G mora biti upaljen.

```
byte segmenti[] = { B00111111,  B00000110,  B01011011,  B01001111,
B01100110,B01101101,  B01111101,  B00000111,  B01111111,  B01101111 };
```

Display pospajaj kao na skici (zasad nije potrebno implementirati i funkciju gumba):



Primjeti da za implementaciju ovog zadatka možeš koristiti *bitwise* operacije iz pretprošle vježbe. Za bržu implementaciju, poželjno je iskoristiti funkciju *toggleLED* s prethodne radionice. Po završetku implementacije, u glavnom programu implementiraj brojač sekundi koji se reserita na nulu nakon 9. Sekunde. Za ovo koristi operaciju *modulo* (%).

Zadatak 6: Ukoliko si sve zadatke do sad uspješno rješio, pokreni sljedeći kod (dostupan i na linku: pastebin.com/qTiM9rJV):

```
void setup() {
    pinMode(13, OUTPUT);
    pinMode(10, OUTPUT);
    digitalWrite(13, LOW);
    pinMode(12, INPUT);
}

void loop() {
    unsigned long measuredDistance = distance_cm();
    int distance_dm = (int) ((measuredDistance / 10) % 10);
    setNumber7Segment(distance_dm);
    playTone(min(10, 100 - measuredDistance), 1000, 10);
}
```

U programu je moguće mijenjati neke vrijednosti, ali je prikazana ideja.
Što bi trebao raditi ovaj program? Ukoliko radi kao što očekuješ, čestitam na uspješnoj implementaciji parking senzora :)

Zadatak 5 (za brze, s prošle radionice): Pomoću dva gumba i LED matrice implementiraj jednostavnu “zmijicu”. Zmijica se smije “zabijati” u sebe, a prolazak kroz zidove ne treba biti definiran. Akciju pritiskom na dvaju gumba potrebno je implemenitrati prekidnim rutinama, prouči ih (engl. *interrupt*). Ukoliko si upoznat s objektom paradigmom, implementiraj zmijicu kao objekt. Interna struktura koju možeš koristiti za pohranu stanja zmijice su dva niza fiksne duljine, svaki niz koristeći kao jednu dimenziju zmijice, a svaki element kao jedan “segment” zmijice.