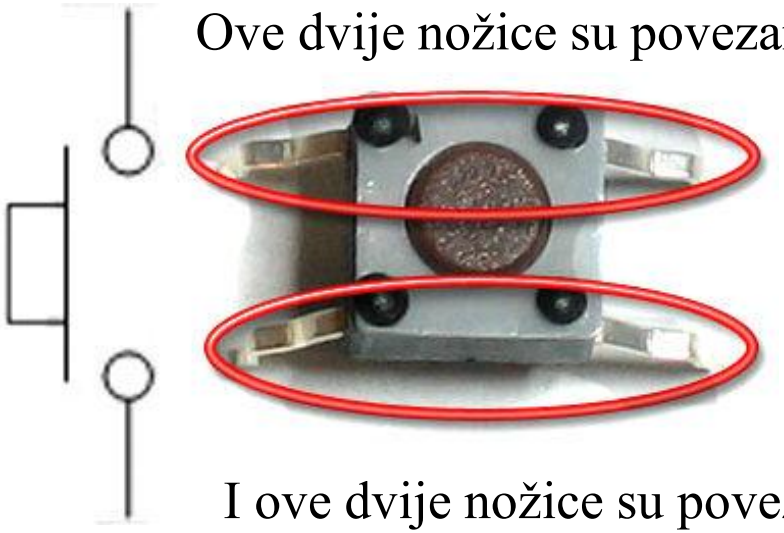
Dizajniranje jednostavnih kolaborativnih rješenja u sistemima sa programabilnim platformama

Dodajmo senzore -dugme

- Dugme (engl. Push button) – Ima dva stanja: pritisnuto (`HIGH`) i otpušteno (`LOW`) i ostvaruje kratak spoj između nožica kada je pritisnuto.
- Očitava mu se vrijednost funkcijom `digitalRead().`
- Dugme zapravo ima dvije konekcije – sa dvije moguće vrijednosti napona.
- Kada dugme nije pritisnuto, gornje nožice su na istom potencijalu i donje nožice su na istom potencijalu, ali ne postoji međusobna veza.



- Kada se dugme pritisne, sve četiri nožice su na istom potencijalu.

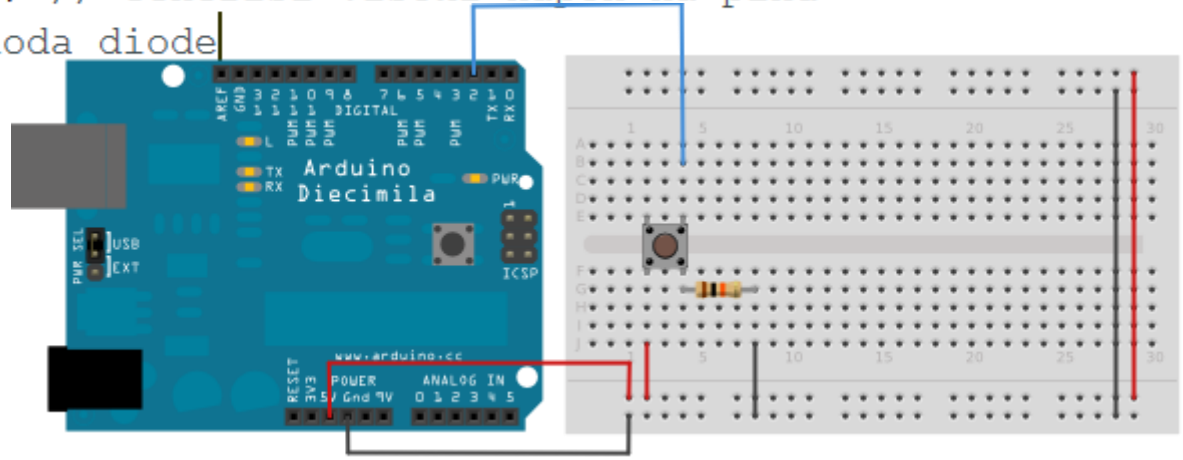
https://www.shallowsky.com/arduino/class/button.html

Potrebna sintaksa

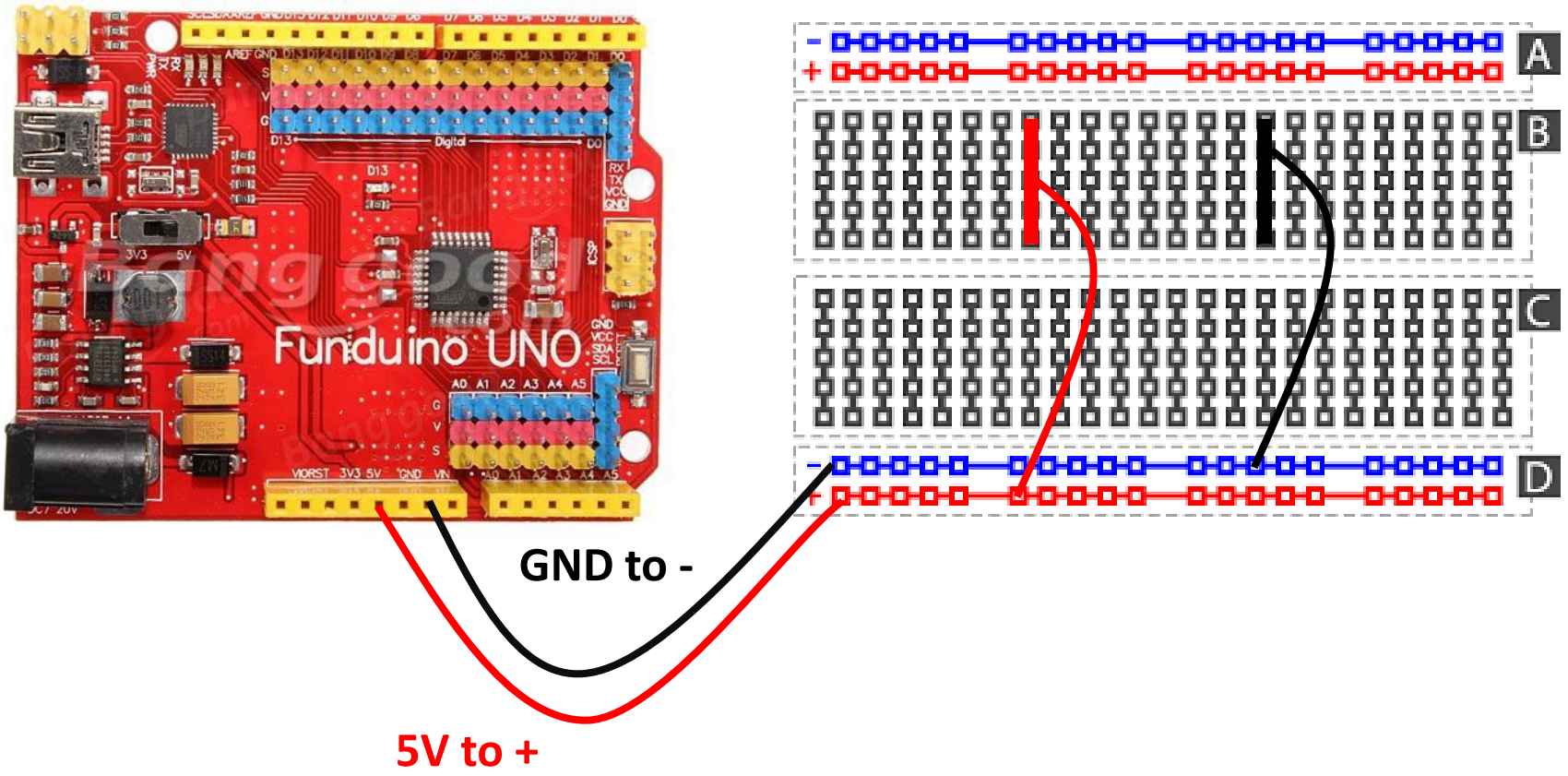
- `int digitalRead(pin)` – provjerava da li postoji napon na specificiranom pinu dugmeta i vraća vrijednost `HIGH/LOW` u zavisnosti od rezultata provjere.
- Omogućava osluškivanje okruženja, prijem i skladištenje informacija o njegovom ponašanju, i djelovanje na osnovu njih.
- `File > New` za otvaranje novog fajla za editovanje – pisanje programa
- Globalne promjenljive i deklaracija su iste kao u u programskom jeziku C.
- Selekcije imaju istu sintaksu kao u programskom jeziku C.
- Petlje imaju istu sintaksu kao u programskom jeziku C.

Povežimo diodu sa dugmetom

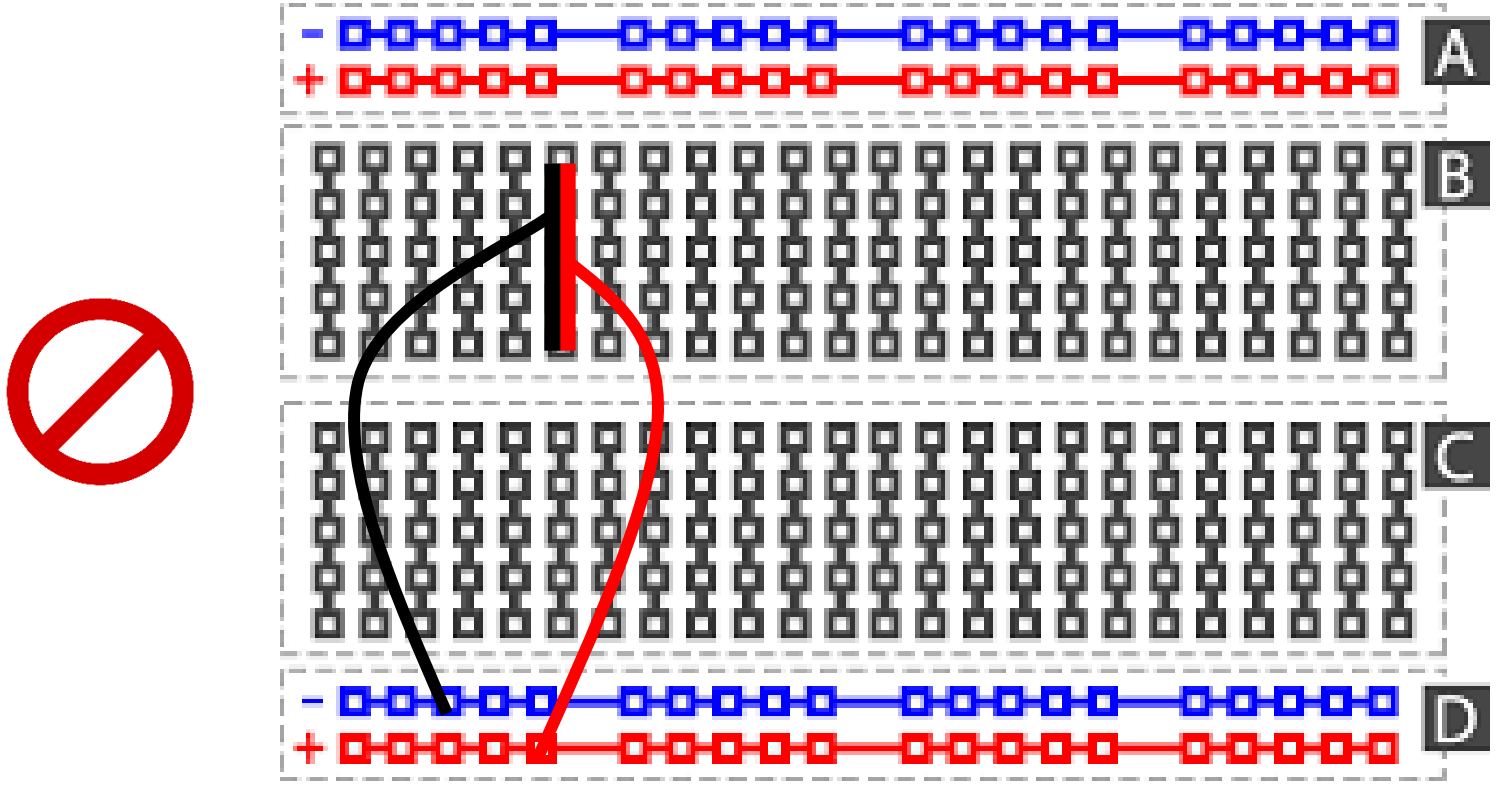
```
/*pritisak na dugme uključuje diodu*/
const int LED = 13; // pin za LED
const int BUTTON = 2; // pin za dugme
int val = 0; // početno stanje  pina BUTTON (HIGH/LOW)

void setup() {
  pinMode(LED, OUTPUT); // LED je izlazni pin
  pinMode(BUTTON, INPUT); // BUTTON je uzlazni pin
}

void loop(){
  val = digitalRead(BUTTON); // čitaj vrijednost ulaznog pina i pamti je

  // Provjeri da li je dugme pritisnuto (HIGH)
  if (val == HIGH) {
    digitalWrite(LED, HIGH); // Generiši visoki napon na pinu
    // za koji je vezana anoda diode
  }
  else {
    digitalWrite(LED, LOW);
  }
}
```



Napajanje



Ne spajati + i – provodnike



Pretvorimo dugme u prekidač

```
/*uključi diodu nakon pritiska na dugme i neka
ostane uključena do narednog pritiska*/
const int LED = 13; // pin za LED
const int BUTTON = 2; // ulazni pin za dugme
int val = 0; // početno stanje  pina BUTTON (HIGH/LOW)
int state = 0; // 0 = LED iskljucena / 1 = LED ukljucena

void setup() {
  pinMode(LED, OUTPUT); // LED je izlazni pin
  pinMode(BUTTON, INPUT); // BUTTON je uzlazni pin
}

void loop(){
  val = digitalRead(BUTTON); // čitaj vrijednost ulaznog pina i pamti je
  // Provjeri da li je dugme pritisnuto (HIGH)
  // i promijeni stanje diode ukoliko jeste
  // isključi je ukoliko je uključena i uključi ukoliko je isključena
  if (val == HIGH) {
    state = 1 - state; //1-0=1 i 1-1=0
  }
  if (state == 1){
    digitalWrite(LED, HIGH); // uključi LED
  }
  else {
    digitalWrite(LED, LOW);
  }
}
```

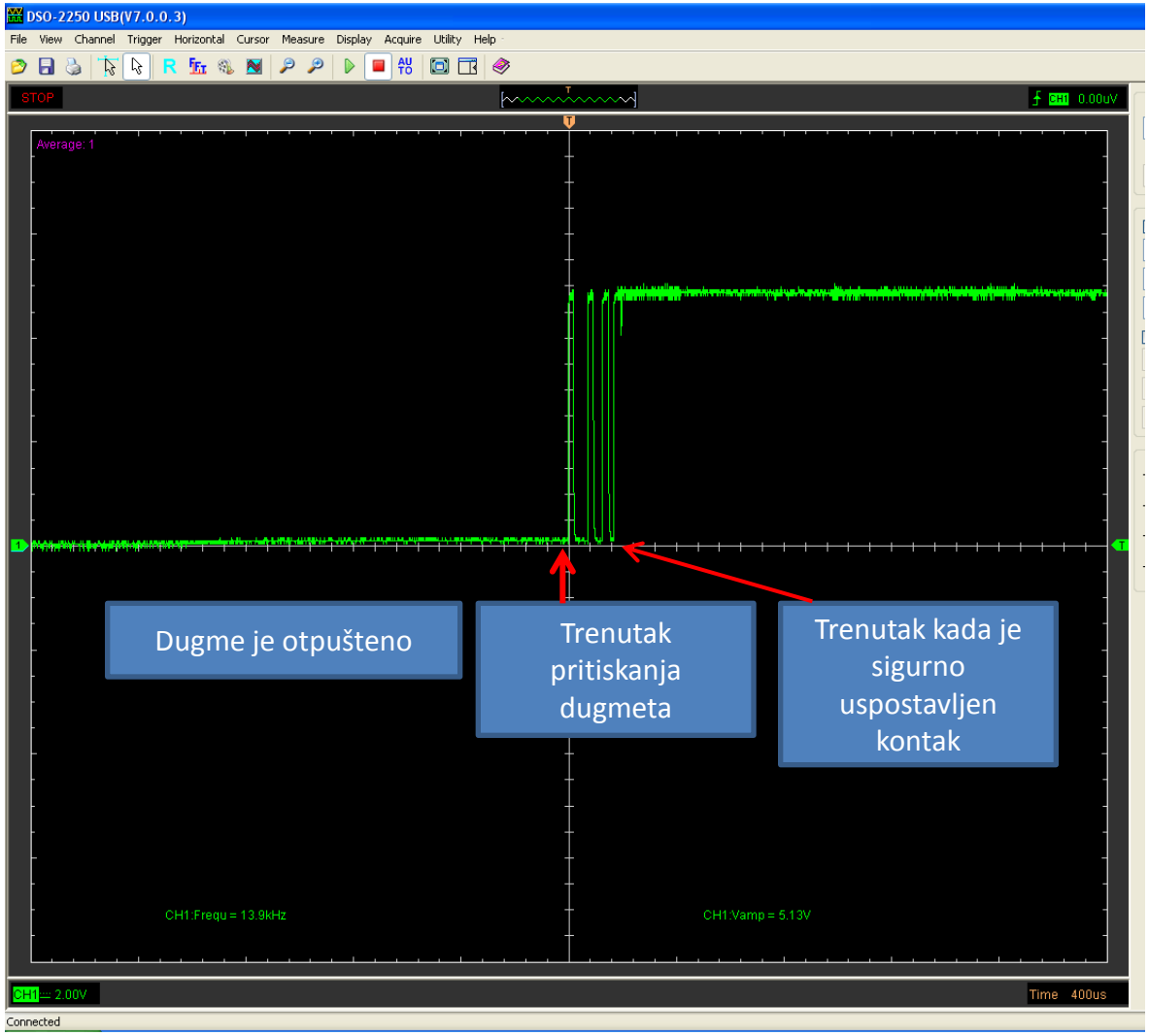
Testirajmo kod

- Primjećujemo da se svijetlo mijenja tako brzo da ne možemo pouzdano reći da li je dioda uključena ili isključena.
- Arduino je veoma brz. Izvršava sopstvene interne instrukcije brzinom od 16 miliona instrukcija po sekundi.
- Ovo znači da za vrijeme dok je pritisnuto dugme, Arduino možda čita stanje dugmeta (val = `HIGH`) nekoliko hiljada puta i shodno tome mijenja vrijednost promjenjive `state` (0,1,0,1,0,1). Dakle, dioda se velikom brzinom uključuje i isključuje.
- Da bismo pretvorili dugme u prekidač treba da odredimo tačan trenutak kada je dugme pritisnuto ili otpušteno, odnosno promjenljiva `val` promijenila vrijednost i da samo u tom trenutku mijenjamo stanje promjenljive `state`.
- Čuvajmo prethodnu vrijednost promjenljive `val` u pomoćnoj promjenljivoj `val_old` i mijenjajmo stanje diode samo kada se pritisne dugme (`val = HIGH` i `val_old = LOW`)

```
/*uključi diodu nakon pritiska na dugme i neka
ostane uključena do narednog pritiska*/
const int LED = 13; // pin za LED
const int BUTTON = 2; // ulazni pin za dugme
int val = 0; // početno stanje  pina BUTTON (HIGH/LOW)
int state = 0; // 0 = LED iskljucena / 1 = LED ukljucena

void setup() {
  pinMode(LED, OUTPUT); // LED je izlazni pin
  pinMode(BUTTON, INPUT); // BUTTON je uzlazni pin
}

void loop(){
  val = digitalRead(BUTTON); // čitaj vrijednost ulaznog pina i pamti je
  // Provjeri da li je dugme pritisnuto (HIGH)
  // i promijeni stanje diode ukoliko jeste
  // isključi je ukoliko je uključena i uključi ukoliko je isključena
  if (val == HIGH) {
    state = 1 - state; //1-0=1 i 1-1=0
  }
  if (state == 1){
    digitalWrite(LED, HIGH); // uključi LED
  }
  else {
    digitalWrite(LED, LOW);
  }
}
```

Testirajmo kod

- Dugme ima u sebi mehanički prekidač koji nije savršen, posebno kada se dugme ne pritisne do kraja, pa se generišu neki lažni signali koji se nazivaju odbijanje (engl. bouncing). Odnodno potrebno je neko vrijeme da se uspostavi siguran kontakt – stalna HIGH vrijednost očitavanja.



https://www.allaboutcircuits.com/uploads/articles/jc_osc_1a.png

Testirajmo kod

- Kada dođe do ove pojave, Arduino, neko vrijeme nakon pritiskanja dugmeta, vidi veoma brzu sekvencu signala za uključivanje i isključivanje, iako se ne djeluje na dugme.
- Postoji mnogo tehnika koje su razvijene da bi se odradio debouncing.
- Obično je dovoljno da se doda kašnjenje, dovoljno dugo da bi se sljedeće očitavanje vršilo nakon stabilizacije napona na vrijednost HIGH.
- Pokazuje se da je dovoljno kašnjenje od 10 do 50 ms.
- Ukoliko u konkretnom primjeru, i dalje bude detektovano blinkanje diode, povećati ovu vrijednost.

```
/*uključi diodu nakon pritiska na dugme i neka
ostane uključena do narednog pritiska.
Odrađen debouncing sa kašnjenjem*/

const int LED = 13; // pin za LED
const int BUTTON = 7; // ulazni pin za dugme

int val = 0; // stanje  pina BUTTON (HIGH/LOW)
int state = 0; // 0 = LED iskljucena / 1 = LED ukljucena
int val_old = 0; // čuva prethodnu vrijednost promjenljive val

void setup() {
  pinMode(LED, OUTPUT); // LED je izlazni pin
  pinMode(BUTTON, INPUT); // BUTTON je uzlazni pin
}
```

```
void loop(){
  val = digitalRead(BUTTON); // čitaj vrijednost ulaznog pina i pamti je

  // Provjeri da li je dugme upravo (u ovoj iteraciji) pritisnuto
  // i promijeni stanje diode ukoliko jeste
  if (val == HIGH && val_old == LOW) {
    state = 1 - state; //1-0=1 i 1-1=0
    delay(10); // dodaj kašnjenje kako bi se prije sljedeće
    // iteracije sigurno prešlo u dio sa uspostavljenim
    // kontaktom - sigurno vrijednots val = HIGH
  }
  val_old = val; //ovo je stara vrijednost za sljedeću iteraciju
  if (state == 1){
    digitalWrite(LED, HIGH); // uključi LED
  }
  else {
    digitalWrite(LED, LOW);
  }
}
```

Podsjetimo se prvog primjera sa prethodnih predavanja

```
#define LED_PIN 13
void setup()
{
  pinMode(LED_PIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_PIN, HIGH);
  delay(20);
  digitalWrite(LED_PIN, LOW);
  delay(20);
}
```

```
#define LED_PIN 13
void setup()
{
  pinMode(LED_PIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_PIN, HIGH);
  delay(10);
  digitalWrite(LED_PIN, LOW);
  delay(30);
}
```

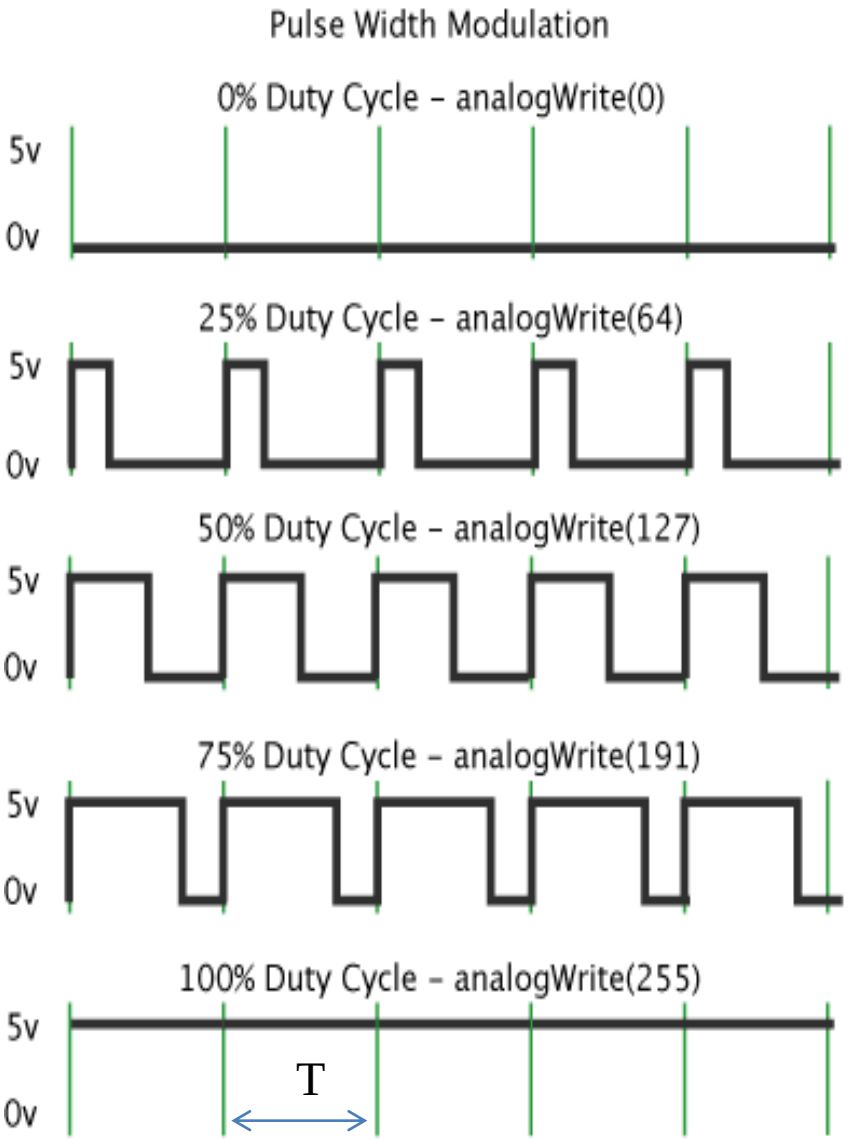
Smanjivanjem vrijednosti za kašnjenje, povećavamo frekvenciju uključivanja i isključivanja dioda. U jednom trenutku, zbog ograničene mogučnosti obrade slika (40 slika u sekundi), naše oko neće detektovati blinkanje diode. Posljedica je da se dobija prigušena svjetlost diode u mjeri koja odgovara odnosu kašnjenja nakon paljenja i gašenja diode.
Lijevo je prigušena na 50%, desno na 25 %.

PWM (Pulse Width Modulation)

- Način postizanja efekta prigušene svjetlosti u primjeru diode koja blinka nije uvijek prikladan.
- Ukoliko se želi pročitati senzor ili poslati podaci na serijski port, desiti će se da LED blinka dok čeka da se završi sa čitanjem senzora.
- Srećom, Arduino ploča ima analogne pinove **3, 5, 6, 9, 10 i 11**, koji se mogu kontrolisati instrukcijom `analogWrite(pin,PWM).`
- Nakon izvršavanja ove funkcije `pin` će generisati povorku pravougaonih impulsa (5V i 0V), u kojoj je trajanje visokog nivoa napona (5V – dioda je uključena) definisano argumentom `PWM`.
- Ova povorka impulsa se smjenjuje do sljedećeg poziva funkcije koja ima zadati pin kao argument.

PWM

- `analogWrite(PIN,PWM)` očekuje broj od 0 do 255 kao argument, pri čemu 255 daje punu osvetljenost (konstantan napon od 5V) i 0 daje isključeno (konstantan napon od 0V).
- Napon na zadatom pinu i dalje može da ima samo dvije vrijednosti, mi sada zadajemo njihovo trajanje!
- U jednoj periodi T imamo jedan put napon 5V i jedan put napon 0V.
- Odnos *PWM/255*T* je trajanje napona od 5 V u jednoj periodi *T*.
- Frekvencija kojom se smjenuju povorke je 490 Hz, odnosno 950 Hz(1/s) za pin 5 i 6 kod ArduinoUno.
- Interesantan video
- https://www.youtube.com/watch?v=GQV0ElK0Nv4



frekvencija = 1/T

https://www.arduino.cc/en/Tutorial/PWM

```
// Postepeno prigušivati i pojačavati
// osvjetljenost LED diod

const int LED = 9; // pin za LED
int i = 0; // Brojač kojim ćemo mijenjati
// širinu impulsa

void setup() {
 pinMode(LED, OUTPUT); // LED je output
}

void loop(){
  for (i = 0; i < 255; i++) { // petlja od 0 do 254
    analogWrite(LED, i); // postavljaj osvjetljaj LED diode
    delay(10); // Čekaj 10ms jer se analogWrite trenutno
    // izvršava i ne bi se vidjele promjene
 }
  for (i = 255; i > 0; i--) { // petlja od 255 do 1 (priguši)
    analogWrite(LED, i); // postavljaj osvjetljaj LED diode
    delay(10); // čekaj 10ms
 }
}
```

PWM

- Sada ćemo napraviti primjer u kojem pritisak na dugme i njegovo otpuštanje uključuje i isključuje diodu diodu, a duže zadržavanje pritisnutog dugmeta mijenja osvjetljaj diode.
- Koristimo taster i diodu

```
const int LED = 9; // pin za LED
const int BUTTON = 7; // input pin za taster
int val = 0; // čuva vrijednost input pin-a
int old_val = 0; // čuva prethodnu vrijednost za val
int state = 0; // 0 = LED off  1 = LED on
int brightness = 128; // Čuva vrijednost osvjetljaja
unsigned long startTime = 0; // trenutak pritiska tastera

void setup() {
  pinMode(LED, OUTPUT); // LED je output
  pinMode(BUTTON, INPUT); // BUTTON je input
}
void loop() {
  val = digitalRead(BUTTON); // čita i skladišti stanje tastera
  //provjerava da li je taster pritisnut
  if ((val == HIGH) && (old_val == LOW)) {
    state = 1 - state;
    startTime = millis(); // millis() Arduinov sat
  // vraća koliko je ms prošlo od kada je
  // ploča posljedni put restartovana
  //ovdje pamti kada je dugme pritisnuto
  delay(10);
  }
```

```
  // provjerava da li je dgme bilo duže pritisnuto
  if ((val == HIGH) && (old_val == HIGH)) {
    // ako je bilo pritisnuto duže od 500ms.
    if (state == 1 && (millis() - startTime) > 500) {
      brightness++; // povećaj osvjetljaj za 1
      delay(10); // kasni da se ne bi osvjetljaj
      //prebrzo povećavao
        if (brightness > 255) { // 255 je max za osvjetljaj
        brightness = 0; // ako se dobije vrijednost veća od 255
        // resetujemo osvjetljaj na 0
      }
    }
  }
  old_val = val; // val je sada stara vrijednost
  if (state == 1) {
    analogWrite(LED, brightness); // postavi LED na zadati osvjetljaj
  }
  else {
    analogWrite(LED, 0); // LED OFF
  }
}
```

Analogni ulaz

- Da bi se čitali analogni senzori – čija se vrijednost kontinualno mijenja, trebaju nam analogni pinovi.

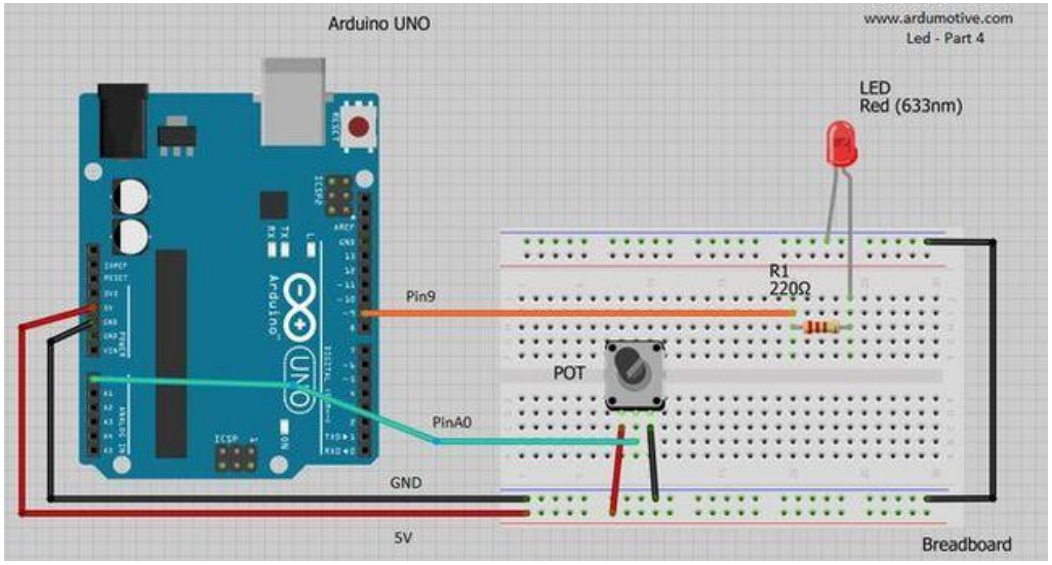
- Analogni pinovi nam omogućavaju da odredimo tačnu vrijednost primijenjenog napona uz pomoć `analogRead()` funkcije
- Ova funkcija vraća broj između 0 i 1023, koji predstavlja napone između 0 i 5 V.
- Na primer, ako postoji napon od 2,5 V primenjen na pin broj 0, analogRead (0) vraća 512.

Efekat prigušene svjetlosti

```
// Podesi osvjetljaj diode na vrijednost zadatu
// analognim ulazom
const int LED = 9; // pin za LED
int val = 0; // vrijednost očitana sa senzora

void setup() {
  pinMode(LED, OUTPUT); // LED je OUTPUT
  // Analogni pinovi su sutomatski postavljeni
  // kao INPUT
}
void loop() {
  val = analogRead(0); // čitaj vrijednost sa senzora
  analogWrite(LED, val/4); // postavi LED na vrijednost
  // definisanu senzorom
  // analogWrite očekuje vrijednosti 0-255
  // zato dijelimo sa 4. analogRead daje maksimu
  // 1023
  delay(10); // zaustavi program za zadato kašnjenje
}
```

Potenciometar

- Potenciometar je otpornik promjenljive otpornosti. Otpornost mu se mijenja okretanjem jednostavnog dugmeta i očitava se kao analogna vrijednost.
- Svi potenciometri imaju tri pina:
  - Spoljašnji pinovi se koriste za povezivanje izvora napajanja (Vref i gnd).
  - Srednji pin (izlazni - output) daje promjenljivu vrijednosti otpora definisanu okretanjem dugmeta.



Kada se dugme okrene do kraja na jednu stranu, na izlazni pin potenciometra se šalje 0V, kada se okrene na drugu stranu, šalje se 5V. AnalogRead(0) u tom slučaju očitava 1023.

Naredba `map(value, 0, 1023, 0, 255);` vrši preskaliranje vrijednosti promjenljive value u željeni opseg.

```
// Šalje monitoru vrijednost očitanu sa analognog
// ulaza 0

const int SENSOR = 0;
int val = 0; // vrijednost sa senzora
void setup() {
  Serial.begin(9600); // otvori serijski port za slanje
  // 9600 bits per second
}
void loop() {
  val = analogRead(SENSOR); // čitaj sa senzora
  Serial.println(val); // štampaj vrijednost
  delay(100); // čekaj 100ms

}
```