JU MJEŠOVITA SREDNJA ELEKTROTEHNIČKA ŠKOLA TUZLA

ŠKOLSKA GODINA 2019./2020.

MATURSKI RAD

Predmet: Digitalna Tehnika

Tema: Sat sa servo motorima

MENTOR:
Zulfić Edin dipl. ing, el. teh
Mandžukić Asmir dipl. ing, el. teh

UČENIK:
Elmin Softić
Razred: 4T2

TUZLA, april, 2020.god.

Sadržaj

1. Uvod

Ideja za ovim projektnim potiče od želje da upravljamo sa što više servo motora jednim mikrokontrolerom. Razmislili smo o idejama šta da napravimo pa smo smislili da napravimo jednostavni sat koji bi „digitalno“ ispisao cifre tako da bi izgledale kao da su ispisane sa 7-segmentnim displejom. Ti segmenti bi, zapravo, bili servo motori spojeni na jedan mikrokontroler.

Svaka cifra se sastoji od 7 segmenata. Sat mora imati minimalno 4 cifre, dvije za sate, a dvije za minute. To bi značilo da u ovom projektu upravljamo 28 servo motora.

U ovom radu ćemo opisati kompletan proces izrade sata, projektovanje dijelova, način printanja, te opisati elektronske komponente potrebne za realizaciju sata te princip djelovanja samog sata. Kao prilog ćemo dostaviti još i program koji smo napisali da bi ovaj projekt proradio.

Detaljno ćemo analizirati korake kako smo povezali sve dijelove, sastavili motore i isprintali plastične nosače pomoću 3d printera.

2. Specifikacija materijala i opreme

Najprije ćemo navesti i opisati sve dijelove koje smo koristili za izradu ovog rada. Ovo radimo kako bi se bolje upoznali sa projektom i lakše pratili korake koji slijede poslije ovog.

Spisak dijelova:

- Arduino UNO (klon)
- 4 PCA9685 16-Channel Servo Driver-a
- 28 9g Servo motora
- Nosači motora
- Plastične „L“ profil lajsne
- Drvena ploča
- DS1307 RTC modul
- Napojna jedinica od 5V 2A za napajanje motora

Plastične lajsne ćemo koristiti kao segmente brojeva. Samo ćemo ih isjeći na potrebnu dužinu i zalijepiti na nosače.

Drvena ploča nije ništa drugo nego velika, drvena daska na koju ćemo montirati sve ostale dijelove.

Pomenutu ploču i lajsne smatramo da nema potrebe da opisujemo detaljnije.

2.1 Arduino UNO

Arduino UNO je otvorena platforma za razvijanje projekata bazirana na softveru i hardveru koji je lahko koristiti. Sama Arduino platforma koristi ATMEL-ove mikrokontrolere u kao bazu svega.

Ove ploče mogu da čitaju unose kao što su svijetlo na senzoru, pritisak dugmeta ili nivo vode u nekoj posudi i da te pročitane podatke koriste kako bi obavili nešto na izlazu (kao što je aktiviranje motora, paljenje LED i slično).

Odlučili smo da koristimo Arduino UNO jer on podržava I2C protokol komunikacije PCA9685 drajvera, ima dovoljno ulaznih i izlaznih pinova, te dovoljno memorije da podrži naš program.

UNO ima 14 digitalnih I/O[1] pinova(od kojih 6 ima mogućnost PWM izlaza) i 6 analognih I/O pinova. Možemo ga programirati pomoću Arduino IDE.

Čip koji se nalazi na UNO ploči je ATmega328 koji ima unaprijed programiran bootloader kako bi mi, kao korisnik, mogli da lakše programiramo svoj kod na njega. ATmega328 je 8-bitni AVR mikročip.

Ima 32 KB flash memorije od koje se 0.5 KB koristi za bootloader, 2 KB SRAM, te 1 KB EEPROM. Njegova brzina rada je 16 MHz, ali se ona može smanjiti ukoliko postoji potreba.

Napon na kojem radi je 5V, dok sama ploča može primiti 7 do 20 V ulaznog napona što omogućava široku primjenu.

Arduino UNO ima mogućnost UART, I2C, SPI i USB komunikacije.

Slika 1 - Arduino UNO

[1] I/O – Input/Output – Ulaz/Izlaz

2.2 PCA9685 16-Channel Servo Driver

Imajući u vidu da želimo upravljati 28 servo motora jednim Arduinom, to jednostavno ne možemo. Prvo što Arduino ima samo 6 PWM pinova, a nama treba 28, a drugo što sam Arduino UNO ne bi mogao da podrži upravljanje svim motorima.

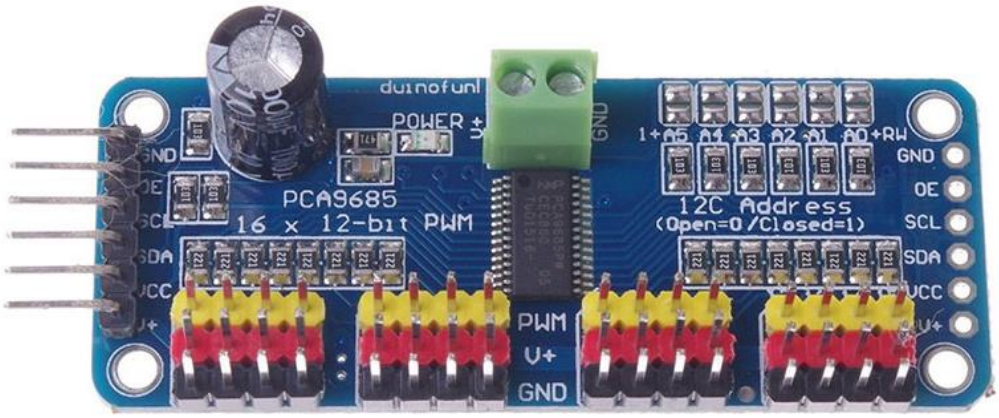
Da bi olakšali sebi izradu projekta, na tržištu smo pronašli PCA9685 drajver koji će da upravlja motorima. Arduino ćemo koristiti da damo naredbe drajveru, a drajver će da upravlja motorima.

Ovaj drajver može da pokreće 16 PWM izlaza. Kontrolisan je I2C protokolom, te se može napajati vanjskim izvorom napajanja. Njegova rezolucija je 12 bita za svaki izlaz.

To znači da za servo motore ima rezoluciju od 4us kada koristimo 60Hz frekvenciju ažuriranja signala.

PCA9685 ima programabilnu I2C adresu pomoću koje možemo povezati do 62 drajvera na jednu I2C sabirnicu. Ukupno, to bi bilo 992 PWM izlaza što je previše za bilo koji projekt.

Mi ćemo koristiti samo 4 drajvera kako bi olakšali upravljanje ciframa. Naravno, možemo i smanjiti potrebu drajvera tako što povežemo dvije cifre na jedan drajver, ali onda moramo proširiti naš program da bi kompenzovali manjak drajvera.



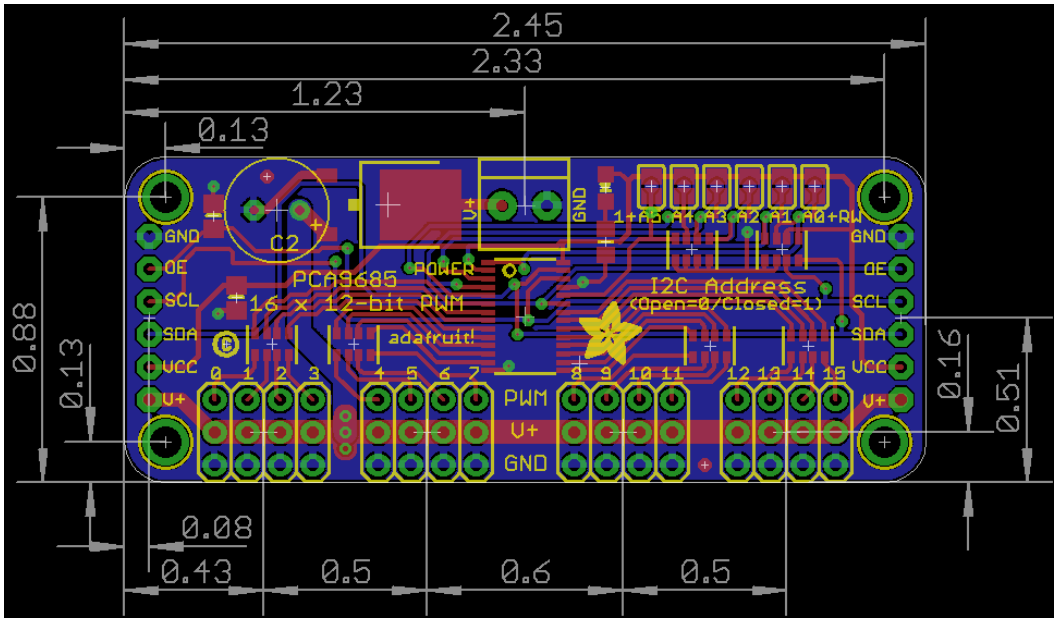
Slika 2 - PCA9685 driver

Drajver ima GND i VCC pinove za spajanje zajedničkog pina GND koji se mora spojiti sa Arduinom i za napajanje logičkog napona. VCC se također spaja na Arduino na njegov 5V izlaz.

„V+“ je opcionalni pin koji služi za napajanje motora. Ne mora se spojiti, ali onda moramo na zeleni terminal blok dovesti odgovarajući napon od 5VDC do 6VDC da bi motori radili. Moguće je koristiti i 12VDC na ovom napajanju, ali samo ako servo motori zahtijevaju toliki napon. Tada moramo paziti da ne povežemo Arduino na taj pin inače ćemo spržiti mikrokontroler.

Također imamo SCL i SDA pinove, te OE pin. OE (*Output Enable*) pin je opcionalni pin kojeg ne moramo koristiti. Služi za omogućavanje rada izlaza drajvera. Kada dovedemo logičku nulu, svi izlazni pinovi su omogućeni, a kada dovedemo logičku jedinicu, onda su onemogućeni. Bez spajanja na OE pin, on je uvijek u stanju logičke nule.

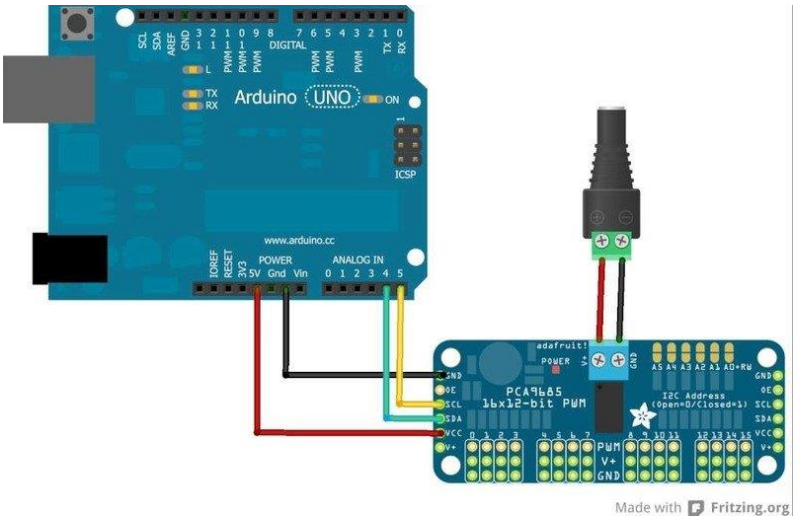
O SCL i SDA pinovima ćemo detaljno pisati kasnije.



Slika 3 - PCB šema drajvera

Drajver spajamo na Arduino na sljedeći način:

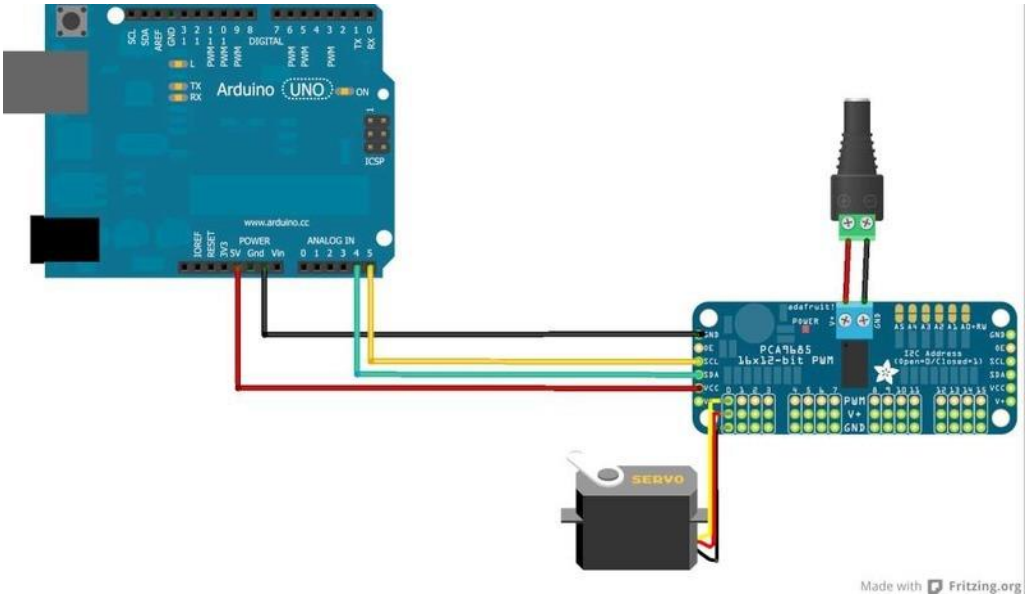
- +5V —> VCC
- GND —> GND
- SDA —> SDA
- SCL —> SCL
- Na terminal blok ćemo dovesti napajanje za motore.



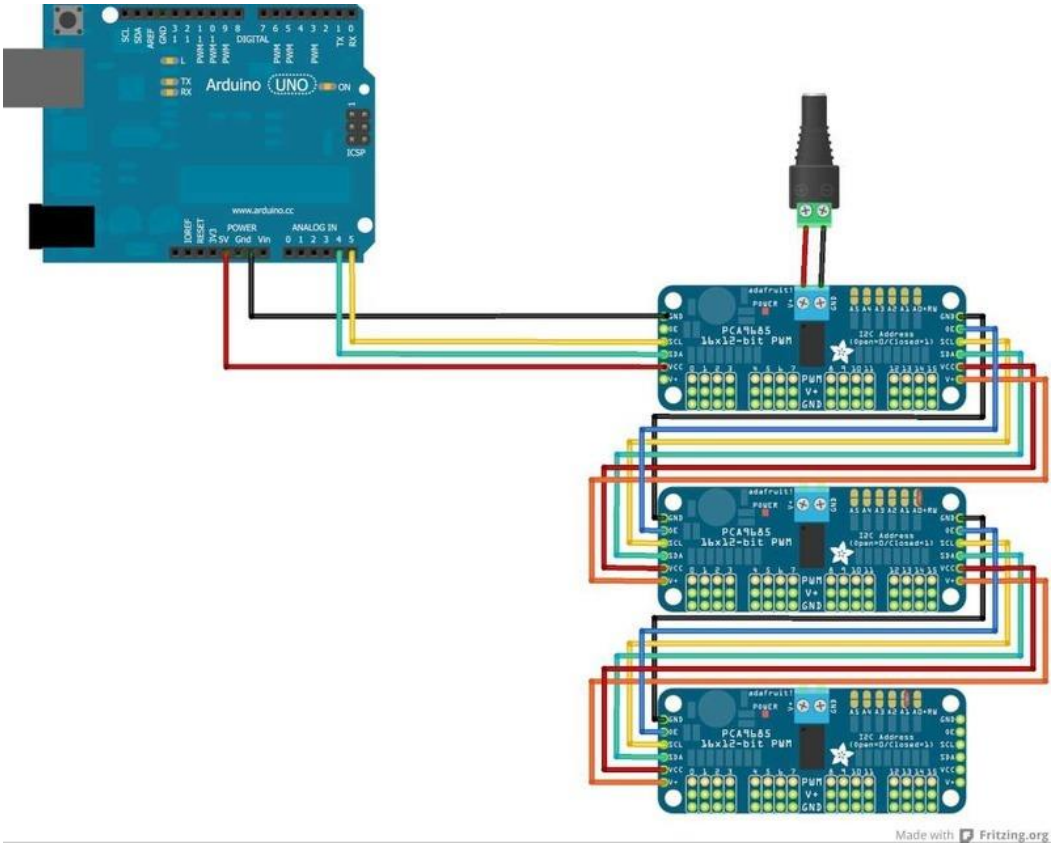
Slika 4 - Povezivanje drajvera

Servo motore ćemo spojiti na izlaze na dnu drajvera.

Ono što je dobra stvar sa ovim drajverom je to što se mogu serijski povezati jer svi koriste istu I2C sabirnicu.



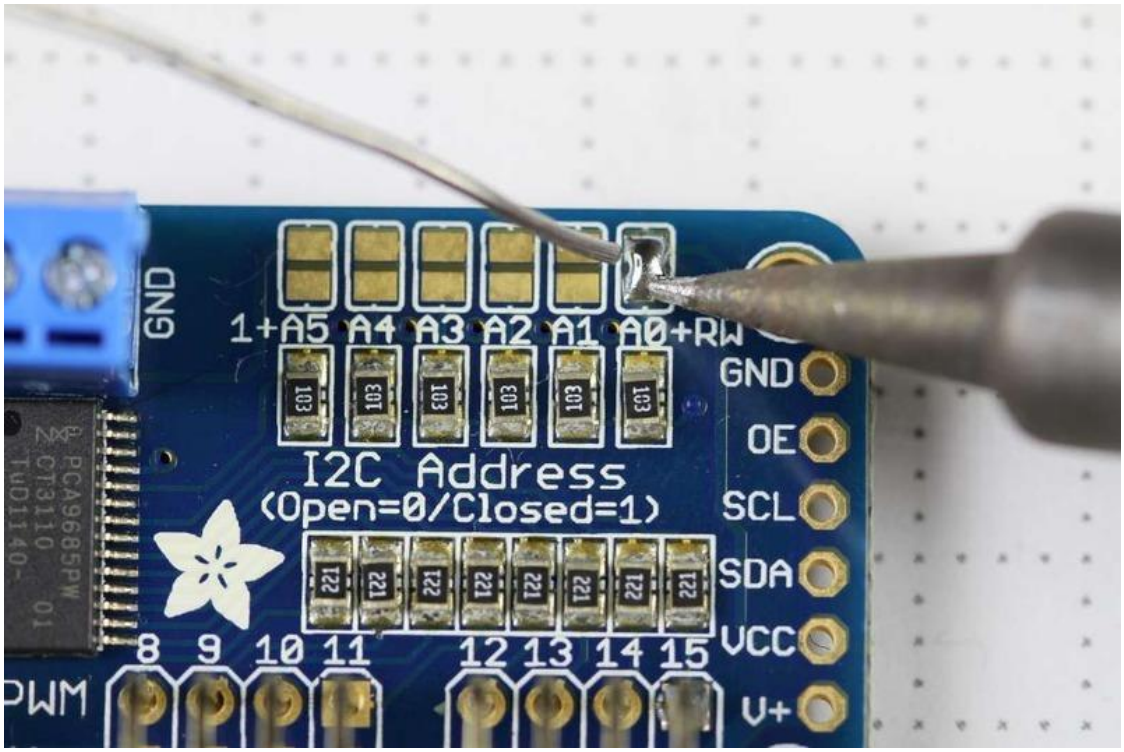
Slika 5 - Povezivanje servo motora na drajver



Slika 6 - Serijsko povezivanje drajvera

Pošto uređaji na I2C sabirnici moraju imati svoju unikatnu adresu, moramo „programirati“ neku na naš drajver. To ćemo uraditi tako što ćemo kratko spojiti binarni broj koji želimo da bude naša adresa.

Osnovna adresa je 0x40, a na nju dodajemo našu vrijednost koju odredimo. Na primjer: kratko spojimo A0 na drajveru. To je broj 1 u binarnom brojnom sistemu. Na 0x40 dodajemo 1 pa će biti 0x41.

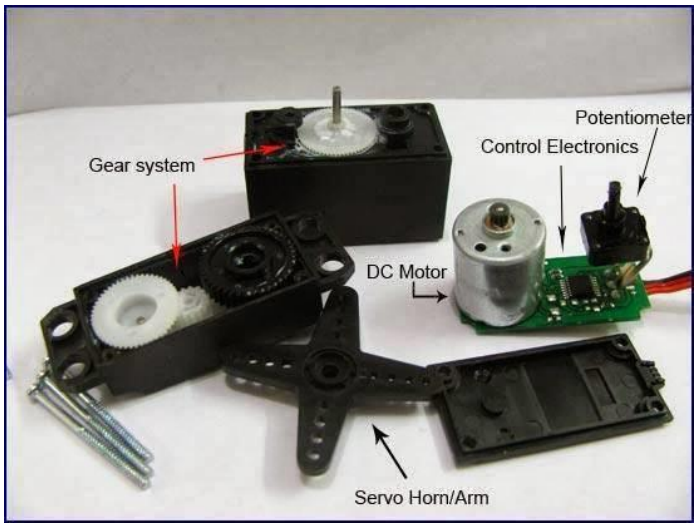


Slika 7 - Programiranje adrese drajvera

2.3 Servo motori

Servo motor je rotacioni aktuator ili linearni aktuator koji omogućuje preciznu kontrolu ugaone ili linearne pozicije, brzine i akceleracije. Sastoji se od potenciometra, motora, nekoliko zupčanika i nešto elektronike.

Motorom se upravlja pomoću PWM signalom koji elektronika unutar motora pretvara u mehanički pokret. PWM signal se dovodi na ulaz signala u servo motor, a signal dolazi od nekog upravljačkog elementa, kao što je Arduino UNO u našem slučaju.

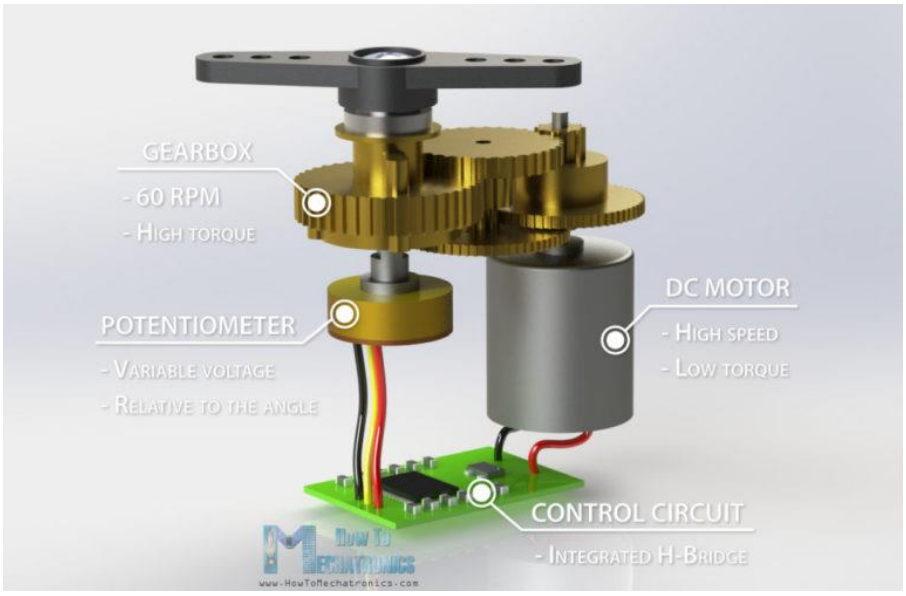


Slika 8 - Dijelovi u servo motoru

Servo motori se, po građi, najčešće okreću od 0º do 180º, neki do 210º, a neki imaju „otključane“ mehanizme tako da se mogu okretati kontinualno. Stepen koliko se motor okreće zavisi od dužine pulsa PWM signala. Napaja se sa 3.5V do 6V.

Svoju rotaciju servo motor graniči potenciometrom tako što okretanje pretvara u otpor, a taj otpor kasnije koristi kako bi zaustavio motor na određenom uglu.

Na slici ispod opet vidimo unutarnje dijelove servo motora.



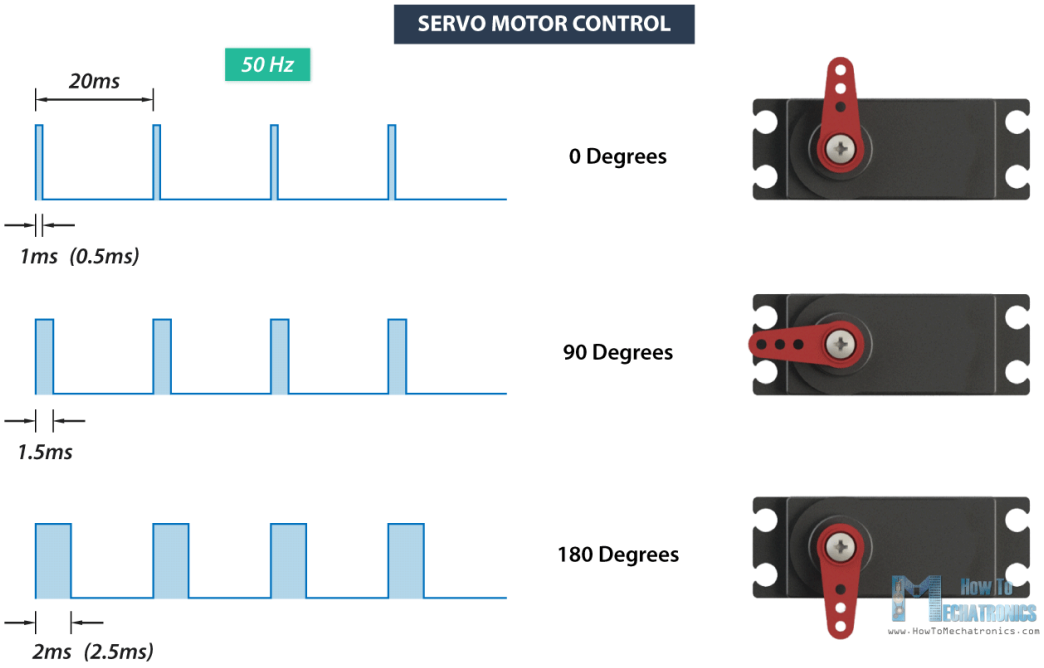
Slika 9 - Dijelovi servo motora

Potenciometar je spojen na zadnji zupčanik tako da kada se okreće motor, okreće se i potenciometar stvarajući napon koji je proporcionalan apsolutnom okretnom uglu osovine motora. U sklopu koji upravlja motorom ima integrisani H-most koji omogućava motoru da se okreće u oba smjera dok razlika dolazećeg signala i signala sa potenciometra ne bude 0.

Motor je upravljan slanjem serije elektronskih pulseva različitih širina. Svaki puls bi trebao biti doveden na signalni ulaz motora svakih 20ms.

Širina pulsa određuje ugao na koji će se motor okrenuti. Generalno motori se okreću do 180º jer imaju fizičku blokadu da se okreću dalje, tako da pulsevi od 1ms odgovaraju uglu od 0º, 1.5ms odgovara 90º, a 2ms odgovora 180º.

Neki proizvođači promijene ove vrijednosti tako da mogu biti malo veće ili manje.



Slika 10 - Zavisnost rotacije motora od širine PWM pulseva

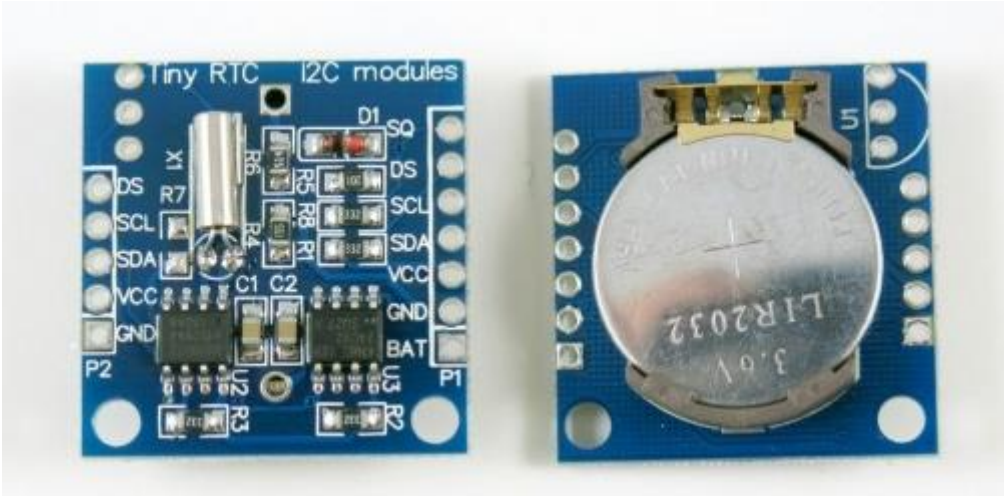
2.4 DS1307 RTC modul

DS1307 RTC modul je modul koji radi na I2C sabirnici. To je sat koji prati sekunde, minute, sate, dane, mjesece, godine, u suštini datum. Može da radi u formatu 24 sata ili 12 sati sa pamćenjem AM/PM indikatora. Automatski prati mjesece sa manje od 31 dana, uključujući korekcije za prijestupne godine.

Podaci su sačuvani BCD notacijom. Sam sat još ima i 56 bajti NV SRAM da može izvršiti sve kalkulacije.

Ovaj modul radi na malim strujama kako bi trošio što manje struje. Kako ne bi izgubili zapisano vrijeme naš modul ima i EEPROM koji proširuje memoriju modula, a pored toga ima i bateriju koja omogućava brojanje vremena uslijed nestanka napajanja preko glavnih pinova. Sam DS1307 ima sklop za automatsko prebacivanje napajanja na baterijski pogon.

Troši manje od 500nA i načinu rada napajanja sa baterije dok i dalje oscilator radi.



Slika 11 - DS1307 RTC modul

Važno je napomenuti da se prvobitno postavljanje vremena i datuma ovog uređaja može postaviti pokretanjem nekoliko linija koda Arduina:

```
#include "RTClib.h"

RTC_DS1307 rtc;

void setup() {
  // sljedeca linija postavlja vrijeme na vrijeme kompajliranja
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

  // Sljedeca linija bi postavila datum I vrijeme na specificnu vrijednost
  // Recimo: Januar 21, 2014 3am
  rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
}
```

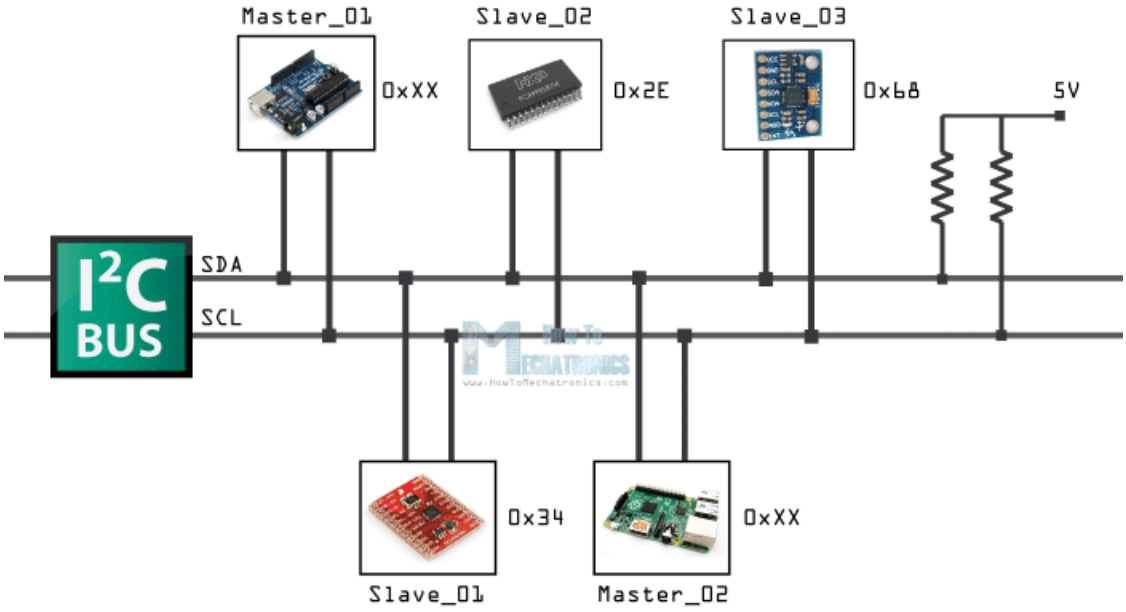
3. I2C

I2C komunikaciona sabirnica je popularna i široko primijenjena zbog svoje jednostavnošću implementacije u projekte i korištenja.

Jednostavna implementacija dolazi s činjenicom da je za rad I2C sabirnice potrebno samo 2 žice za komunikaciju između 112 uređaja kada se koristi 7 bita za adresiranje, a 1008 uređaja ako se koristi 10 bita za adresiranje.

Svaki uređaj ima svoju unikatnu adresu koju „master“ uređaj, onaj koji daje naredbe, može pročitati te adrese i odlučiti na koji spojeni uređaj želi poslati naredbu.

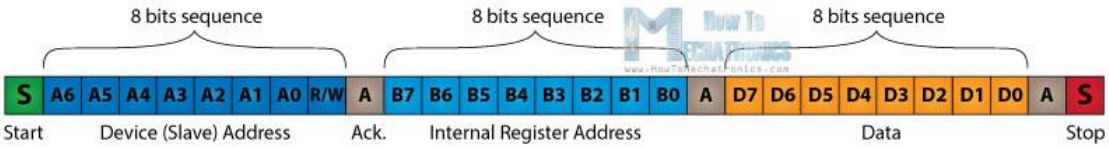
Kod I2C sabirnice postoje 2 linije, *Serial Clock* (SCL) i *Serial Data* (SDA). SCL predstavlja clock signal koji sinhronizira sve uređaje na sabirnici, a SDA je linija preko koje se prenose podaci kao što su naredbe.



Slika 12 - Primjer jedne mreže uređaja

Podaci se šalju sekvencama po 8 bita. Nakon specijalne sekvence za početak komunikacije prate adresa na koju se šalje podatak, a zatim i sam podatak koji se šalje.

Kada se informacije prenesu šalje se specijalni stop signal koji uređaju koji prima podatke govori da je komunikacija sa njim završena.



Slika 13 - Struktura komunikacije

4. Izrada projekta

U ovom dijelu ćemo opisati proces izrade projekta.

4.1 Nosači i servo motori

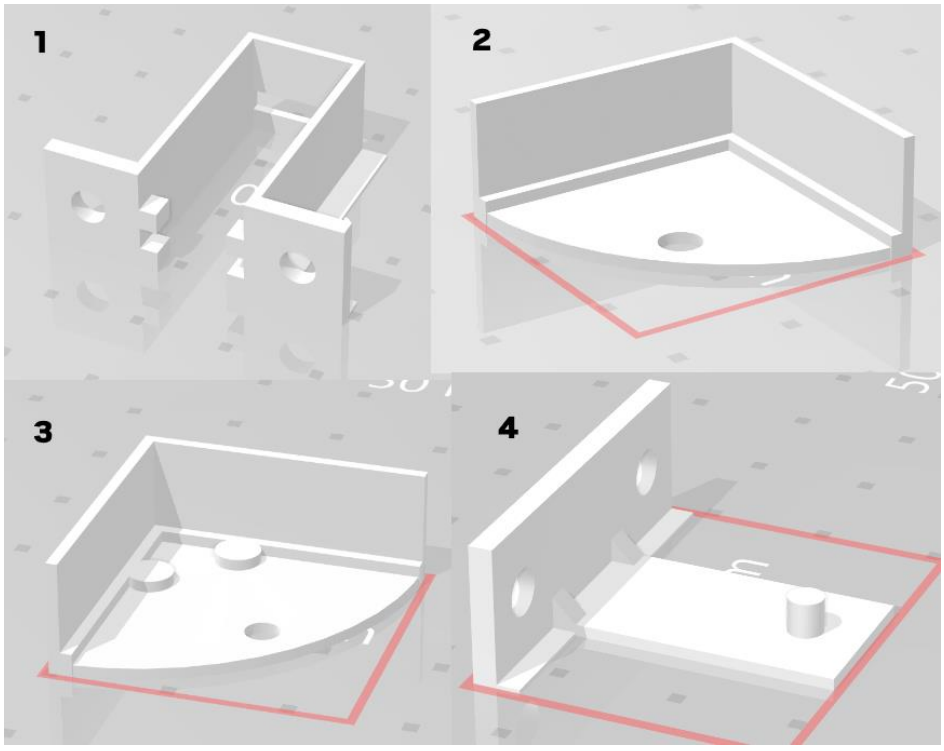
Kako bi izgradili projekt, najprije smo isprintali nosače motora na našem 3D printeru. Da bi testirali printer i modele isprintali smo nekoliko crnih nosača, ali smo kasnije koristili bijelu ABS plastiku.

Ovaj proces je trajao najduže jer je ukupno bilo potrebno isprintati 112 dijelova. Naravno, pojavi se negdje dio koji nije dobro isprintan, pa se mora ponovo printati. Ovo je oduzelo još vremena.

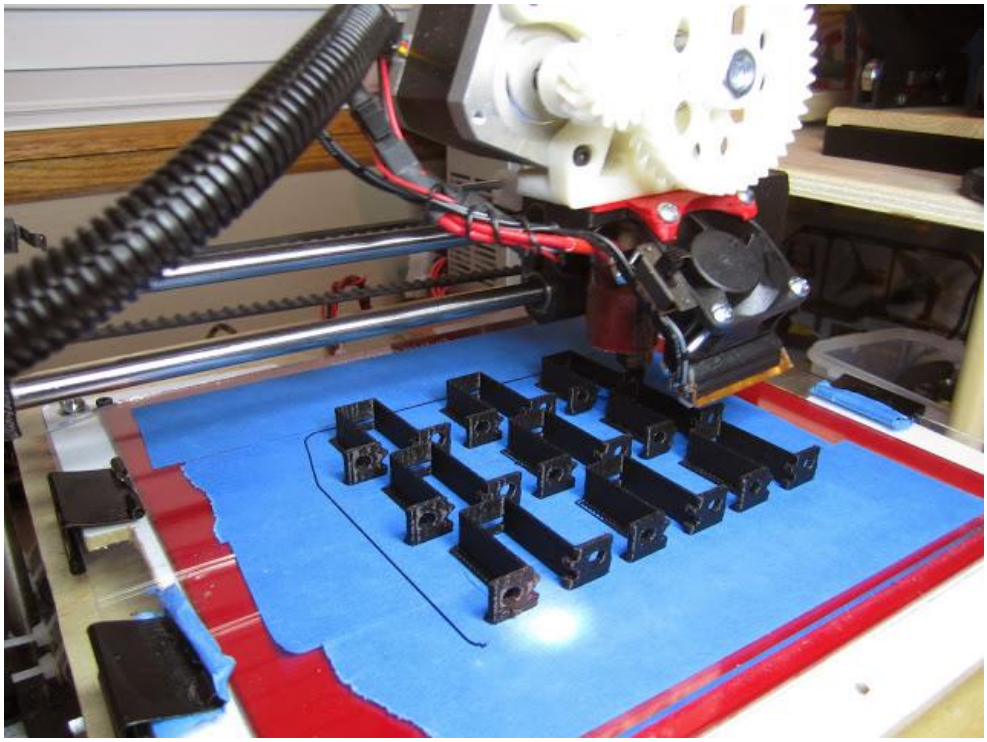
Ako uzmemo da je za svaki dio bilo potrebno oko 12 minuta da se isprinta, sa loše isprintanim dijelovima (ponovno printanje), ukupno je printer radio ~25 sati.

Na slici ispod vidimo modele koje smo printali.

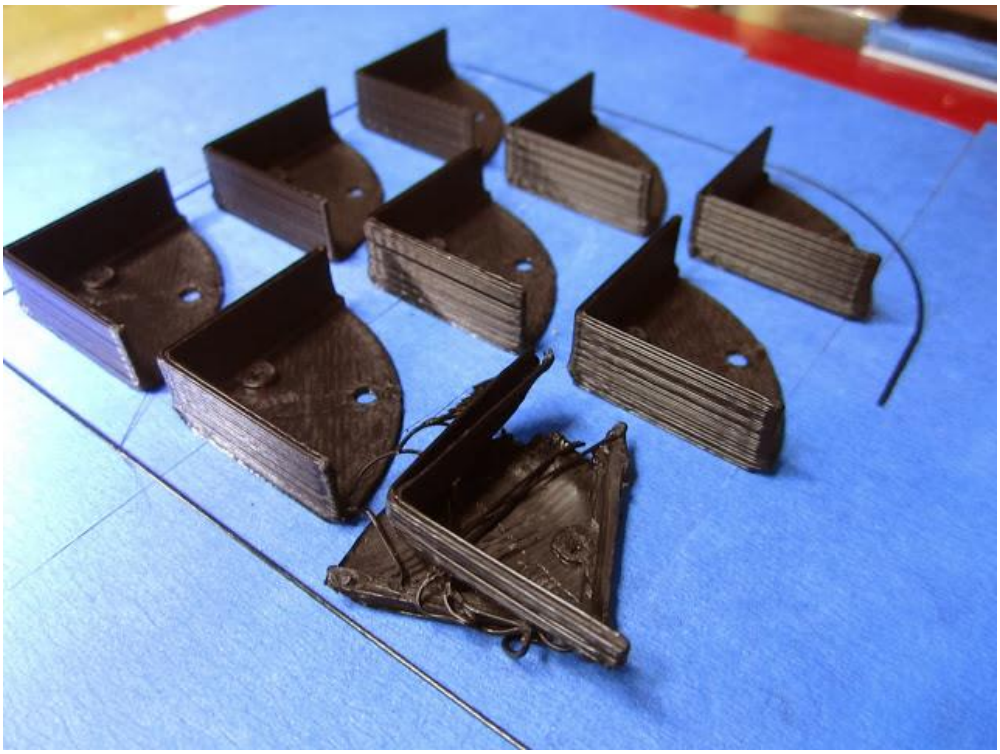
1. Nosač servo motora
2. Jedan od držača segmenta
3. Držač segmenta koji se montira na servo motor
4. Nosač držača 2



Slika 14 - Modeli koje smo printali



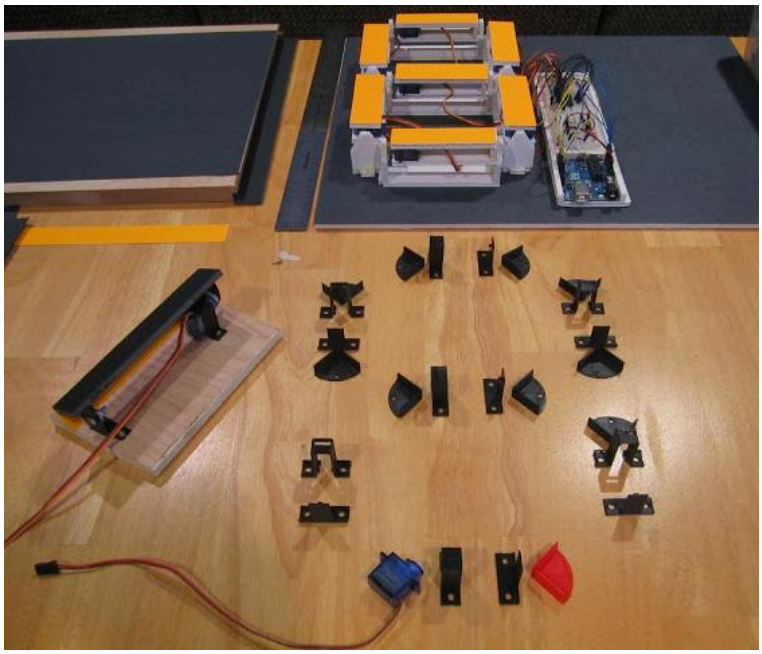
Slika 15 - Prvo printanje dijelova



Slika 16 - Prikaz dijela koji nije dobro isprintan

Na slici iznad vidimo primjer loše printanog dijela. Ovakve dijelove, očito, ne možemo koristiti, te ih moramo ponovo printati.

Isprintane dijelove smo sastavili sa servo motorima tako što smo servo motore stavili u njihove nosače, a na okretni dio smo zavrnuli nosače segmenata.



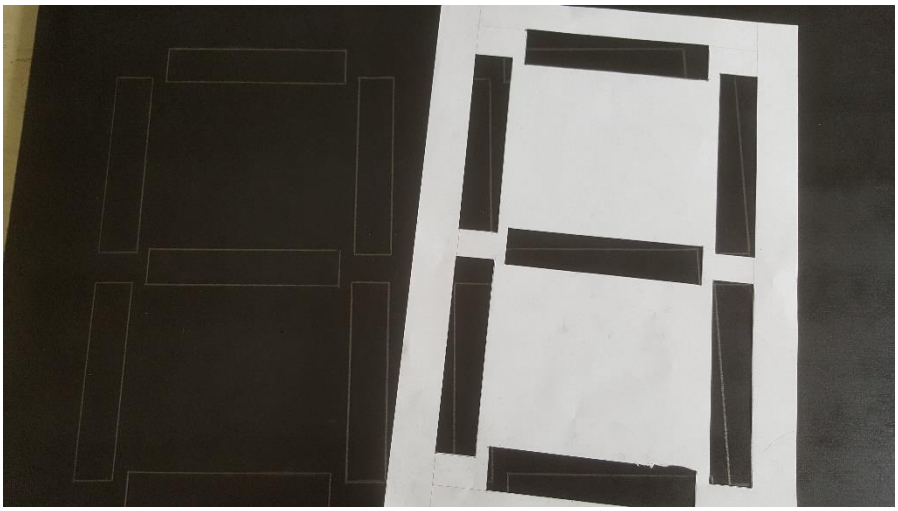
Slika 17 - Isprintani dijelovi poredani u željenoj formaciji

Slika 18 - Montiranje servo motora na nosače

4.2 Priprema drvene ploče i montiranje motora

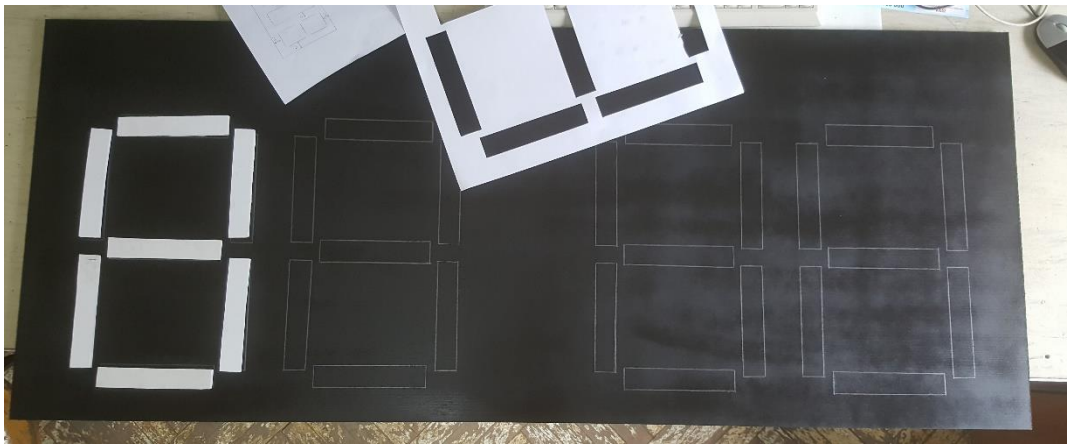
Nakon pripreme i montiranja servo motora na nosače vrijeme bilo je da ih postavimo na drvenu ploču. No, prije nego što smo to odradili, drvenu ploču smo ofarbali crnom bojom kako bi se lakše isticale cifre.

Bilo je potrebno na ploču montirati nosače i servo motore tako da formiraju segmente kao na 7-segmentnom displeju. Da bi olakšali ovaj posao napravili smo šablon po kojem smo nacrtali segmente na ploču grafitnom olovkom.



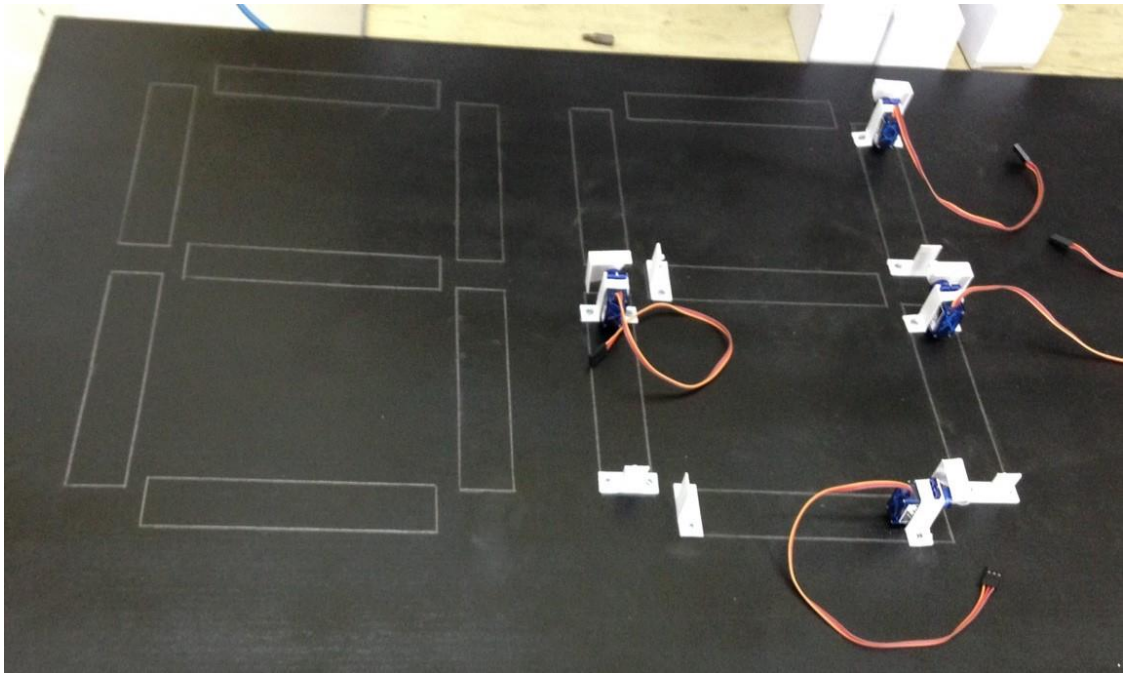
Slika 19 - Šablon i nacrtana cifra

Nakon što smo nacrtali sve 4 cifre mogli smo dobiti predstavu kako naš konačni rezultat treba da izgleda. Da bi lakše pokazali izgled jedne cifre poredali smo 7 dijelova forex ploče. Kasnije će to zapravo biti dijelovi lajsne.

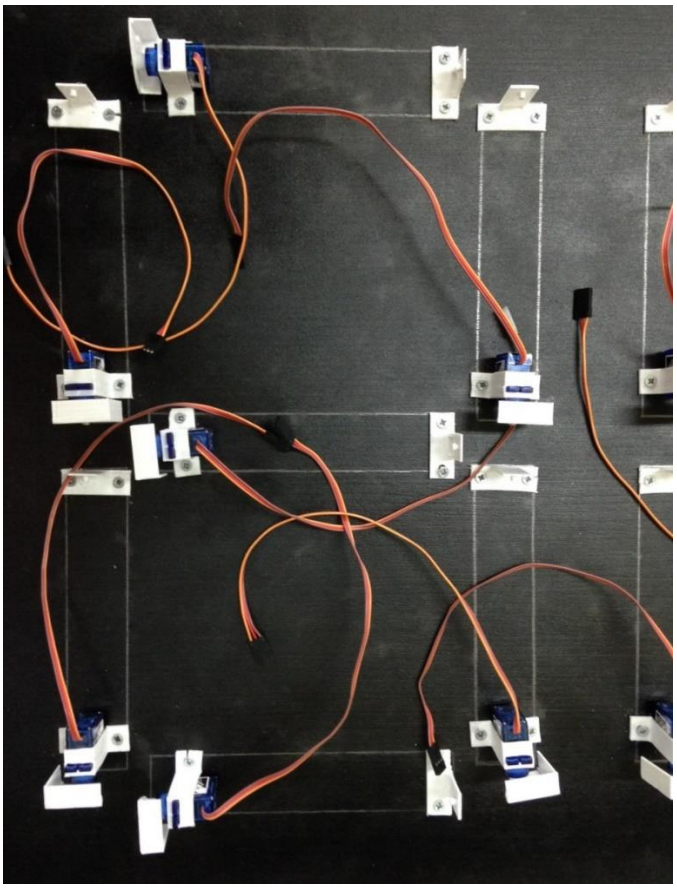


Slika 20 - Nacrtane sve cifre

Bilo je vrijeme da montiramo sve nosače i servo motore na nacrtana mjesta na ploči. Svaki segment bi imao po jedan nosač i jedan servo motor. Te dvije komponente našeg projekta bi bile na oba kraja svakog segmenta.

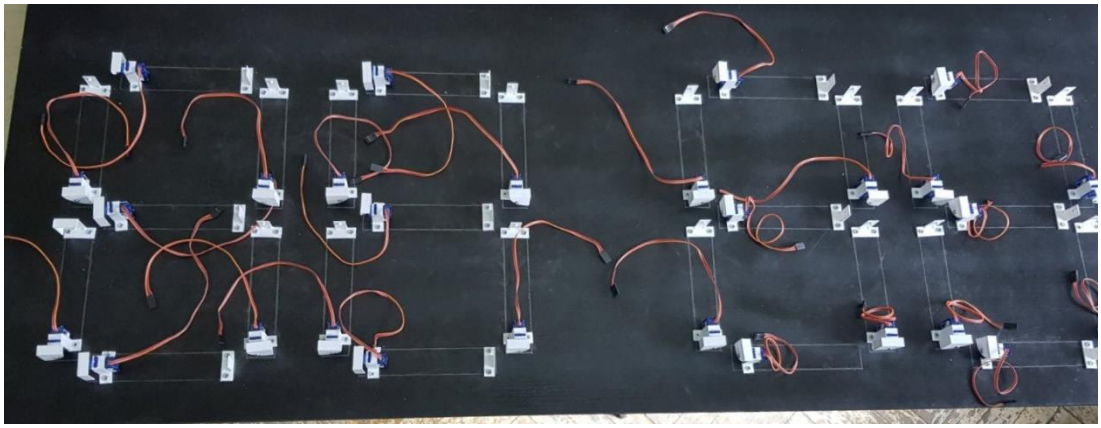


Slika 21 - Proces postavljanja nosača i motora



Slika 22 - Jedna od "gotovih" cifri

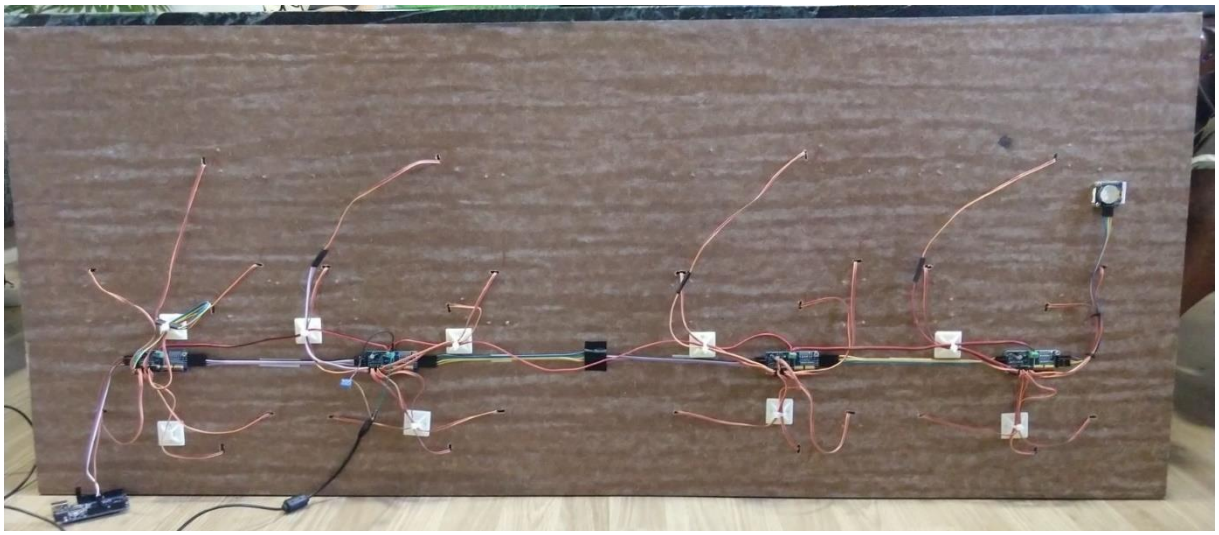
Na slijedećoj slici vidimo kako izgleda naš sat sa svim montiranim nosačima i servo motorima. Preostaje samo da napravimo rupe za žice motora i da postavimo dijelove forex ploče na svaki od motora i nosača, i prednja strana našeg sata bi bila gotova.



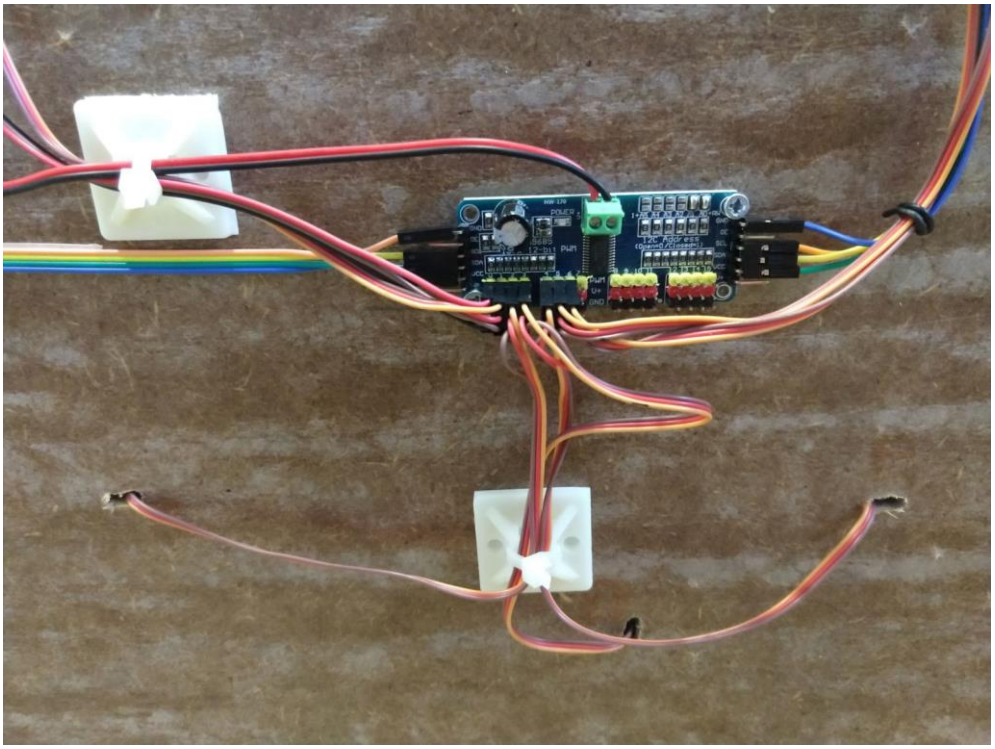
Slika 23 - Svi nosači i motori montirani

4.3 Montiranje elektronike i segmenata

Na poleđini ploče smo montirali sve komponente, povezali ih i organizovali žice, a na prednjoj strani smo montirali lajsne koje smo prethodno isjekli na potrebnu mjeru kako bi služile kao segmenti brojeva. Također smo dodali dva komada plastike kao tačke između brojeva sata i brojeva minuta.

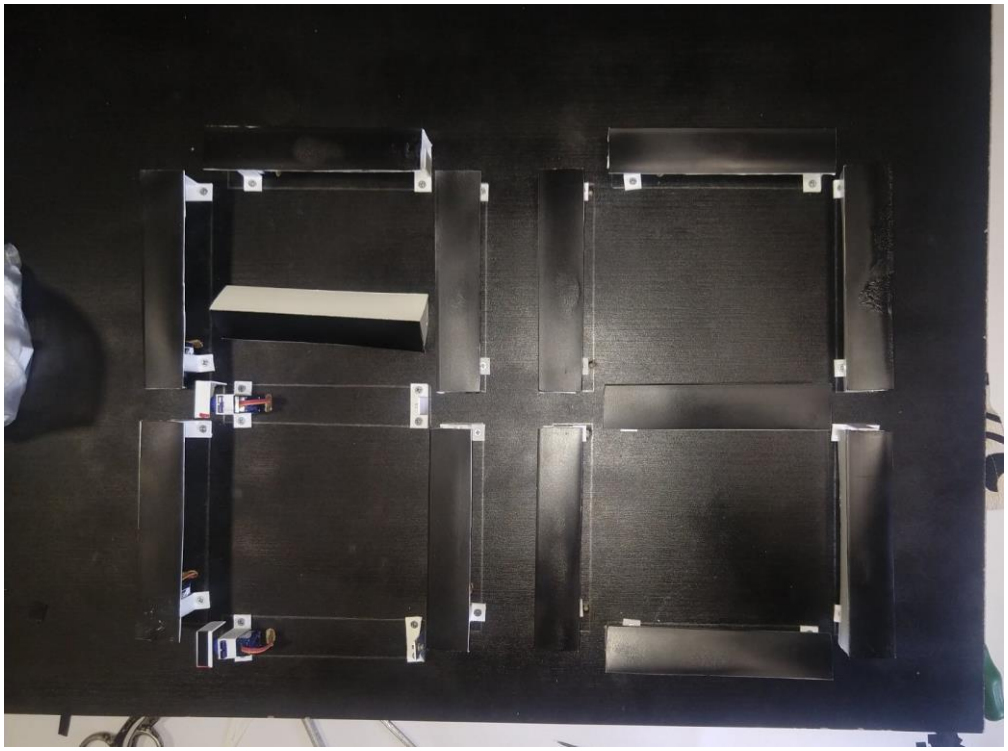


Slika 24 - Spajajne uređaja



Slika 25 - Detaljniji prikaz jednog drajvera

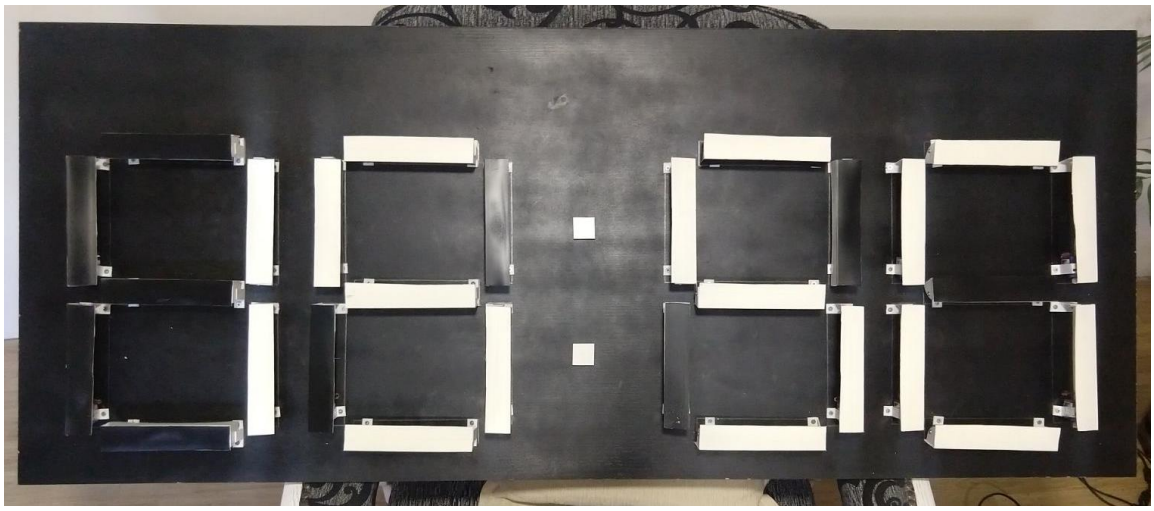
Na slici 25 vidimo detaljniji prikaz spojenih motora na jedan drajver, a na slici 24 se vidi serijsko povezivanje drajvera.



Slika 26 - Postavljanje segmenata na motore

Na slici 26 vidimo postavljanje segmenata na motore. Njih smo samo zalijepili super ljepilom na nosače kako bi osigurali da ne spadnu.

Slika 27 predstavlja konačan sat.



Slika 27 - 15:50 Vrijeme kada je projekt konačno zaživio

5. Program

Program ovog rada je veoma jednostavan. U ovom dijelu ćemo ga detaljno opisati.

5.1 Deklaracije i definisanje

Najprije je bilo potrebno uvesti biblioteke koje program treba da koristi:

- Adafruit_PWMServoDriver.h
- RTClib.h

```
#include <Adafruit_PWMServoDriver.h>
#include "RTClib.h"

RTC_DS1307 rtc;

Adafruit_PWMServoDriver pwm0 = Adafruit_PWMServoDriver(0x40);
Adafruit_PWMServoDriver pwm1 = Adafruit_PWMServoDriver(0x41);
Adafruit_PWMServoDriver pwm2 = Adafruit_PWMServoDriver(0x42);
Adafruit_PWMServoDriver pwm3 = Adafruit_PWMServoDriver(0x43);

#define SERVOMIN 150  //This is the 'minimum' pulse length count
#define SERVOMAX 600  //This is the 'maximum' pulse length count
#define USMIN 600     //This is the rounded 'minimum' microsecond length
#define USMAX 2400    //This is the rounded 'maximum' microsecond length
#define SERVO_FREQ 50 //Analog servos run at ~50 Hz updates

#define HD 3 //Sati Desetice
#define HJ 2 //Sati Jedinice
#define MD 1 //Minute Desetice
#define MJ 0 //Minute Jedinice

//Uglovi motora za ukljucen i iskljucen segment
#define ON 0
#define OFF 90

//Definisani segmenti u kao slova zbog lakseg rada
#define A 0
#define B 1
#define C 2
#define D 3
#define E 4
#define F 5
#define G 6
```

5.2 setup() funkcija

Zatim smo deklarisali RTC koji koristimo i 4 PWM drajvera, te im dodijelili adrese, a nakon toga smo definisali neke vrijednosti koje će nam olakšati pisanje daljnjeg programa.

Svaki Arduion program ima setup() funkciju i loop() funkciju. setup() funkcija se izvršava samo jednom, pri paljenju Arduina, a loop() funkcija se konstantno obavlja i ponavlja.

U setup() funkciji smo uključili serijsku komunikaciju kako ni mogli prepoznati da li je program uspješno počeo da radi, ili ima neka greška. Odredili smo baudrate da bude 9600.

Provjeravamo da li je RTC uspješno pokrenut i da li postoji komunikacija sa njim, a nakon toga smo svaki drajver zasebno podesili sa odgovarajućim podacima.

```
void setup() {
  Serial.begin(9600);
  Serial.println("Setup start...");

  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    while (1);
  }

 if (! rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
  }

  pwm0.begin();
  pwm0.setOscillatorFrequency(27000000);
  pwm0.setPWMFreq(SERVO_FREQ);  // Analog servos run at ~50 Hz updates
  delay(10);

  pwm1.begin();
  pwm1.setOscillatorFrequency(27000000);
  pwm1.setPWMFreq(SERVO_FREQ);  // Analog servos run at ~50 Hz updates
  delay(10);

  pwm2.begin();
  pwm2.setOscillatorFrequency(27000000);
  pwm2.setPWMFreq(SERVO_FREQ);  // Analog servos run at ~50 Hz updates
  delay(10);

  pwm3.begin();
  pwm3.setOscillatorFrequency(27000000);
  pwm3.setPWMFreq(SERVO_FREQ);  // Analog servos run at ~50 Hz updates
  delay(10);

  Serial.println("Setup done!");
}
```

5.3 Funkcija za postavljanje ugla servo motora

Kako bi olakšali upravljanje motorima napisali smo funkciju kojoj proslijedimo koji segment želimo da okrenemo, a funkcija dalje radi to sama.

U suštini, prosljeđujemo podatak na kojem drajveru (tj. koju od 4 cifre), koji motor da se okrene, i za koji stepen.

```
void setServoDegree(uint8_t seg, uint8_t n, uint8_t degree) {
   int pulsedeg = map(degree, 0, 180, SERVOMIN, SERVOMAX);
        if(seg == 0)
        {
          pwm0.setPWM(n, 0, pulsedeg );
        }
        else if(seg == 1)
        {
          pwm1.setPWM(n, 0, pulsedeg );
        }
        else if(seg == 2)
        {
          pwm2.setPWM(n, 0, pulsedeg );
        }
         else if(seg == 3)
        {
          pwm3.setPWM(n, 0, pulsedeg );
        }
}
```

5.4 Funkcija za ispisivanje broja

Kako bi znatno smanjili broj linija koda programa, napravili smo još jednu funkciju koja automatski postavlja primljeni broj na određeno mjesto.

Koristili smo *switch* naredbu koja provjerava koji broj je proslijeđen, a zatim postavlja svaki od segmenata po pravilu izgleda cifre.

```
void displayNumber(uint8_t segm, uint8_t num){
  //display on X segment Y number
  switch(num){
    case 0:
      setServoDegree(segm, A, ON);
      setServoDegree(segm, B, ON);
      setServoDegree(segm, C, ON);
      setServoDegree(segm, D, ON);
      setServoDegree(segm, E, ON);
      setServoDegree(segm, F, ON);
      setServoDegree(segm, G, OFF);
      break;

    case 1:
      setServoDegree(segm, A, OFF);
      setServoDegree(segm, B, ON);
      setServoDegree(segm, C, ON);
      setServoDegree(segm, D, OFF);
      setServoDegree(segm, E, OFF);
      setServoDegree(segm, F, OFF);
      setServoDegree(segm, G, OFF);
      break;

      .
      .
      .

    case 8:
      setServoDegree(segm, A, ON);
      setServoDegree(segm, B, ON);
      setServoDegree(segm, C, ON);
      setServoDegree(segm, D, ON);
      setServoDegree(segm, E, ON);
      setServoDegree(segm, F, ON);
      setServoDegree(segm, G, ON);
      break;

    case 9:
      setServoDegree(segm, A, ON);
      setServoDegree(segm, B, ON);
      setServoDegree(segm, C, ON);
      setServoDegree(segm, D, ON);
      setServoDegree(segm, E, OFF);
      setServoDegree(segm, F, ON);
      setServoDegree(segm, G, ON);
      break;
  }
}
```

Radi prostora prikazali smo samo prva i zadnja dva dijela koda u *switch* naredbi.

5.5 loop() funkcija

U loop() funkciji smo pročitali vrijeme iz RTCa i izračunali koji brojevi se trebaju pokazati na kojem mjestu na našem satu.

Koristili smo prethodno napisane funkcije.

Također smo morali deklarisati neke varijable u koje bi smjestili proračunate cifre koje želimo da se ispisu na našem satu. Njih smo napisali odmah iznad funkcije.

```
int satDesetice, satJedinice, minDesetice, minJedinice;

void loop() {
  DateTime now = rtc.now();

  satDesetice = now.hour() / 10;
  satJedinice = now.hour() % 10;
  minDesetice = now.minute() / 10;
  minJedinice = now.minute() % 10;

  displayNumber(MJ, minJedinice);
  delay(100);
  displayNumber(MD, minDesetice);
  delay(100);
  displayNumber(HJ, satJedinice);
  delay(100);
  displayNumber(HD, satDesetice);
  delay(100);
}
```

6 Zaključak

Iz ovog projekta smo mnogo naučili. Od detaljnog rada servo motora do I2C sabirnice i protokola kojim uređaji komuniciraju.

Napisali smo program koji upravlja motorima i stekli iskustvo u radu sa raznim komponentama. Od drajvera PWM signala do RTC sata i samog Arduina naučili smo kako ih povezati, reći im šta i kako da rade i na koji način ih napajati.

Sat je konačno napravljen, a bilo je i vrijeme. Preostaje samo da ga stavimo na neko mjesto na zidu.

7 Literatura

1. https://www.arduino.cc
2. https://en.wikipedia.org/wiki/Arduino_Uno
3. https://learn.adafruit.com/16-channel-pwm-servo-driver?view=all
4. https://en.wikipedia.org/wiki/Servomotor
5. https://circuitdigest.com/article/servo-motor-basics
6. https://e-radionica.com/hr/blog/2016/06/23/kkm-servo-motor-tower-pro-sg90/
7. https://www.electronics-lab.com/project/ds1307-rtc-module/
8. https://create.arduino.cc/projecthub/electropeak/interfacing-ds1307-rtc-module-with-arduino-make-a-reminder-08cb61
9. https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/
10. https://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/
11. http://archive.fabacademy.org/archives/2017/fablabcept/students/219/final-project.html
12. Google Images