Elektronička kocka

Požarko, Denis

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij, Automatika

ELEKTRONIČKA KOCKA

Završni rad

Denis Požarko

Osijek, 2023.

SADRŽAJ

1. UVOD

Digitalizacija je sve prisutna u svakodnevnom životu ljudi. Sve više i više vidimo "pametnih" stvari koje nikad nismo mogli zamisliti prije 20 godina. Počevši od nosivih uređaja poput digitalnih satova, pa do digitalnih zrcala i frižidera pa nakraju i pametnih klupa i kućanskih zvona. Teško je zamisliti što će doći sljedeće. Želja i maštovitost ljudi za digitalizacijom nema granice.

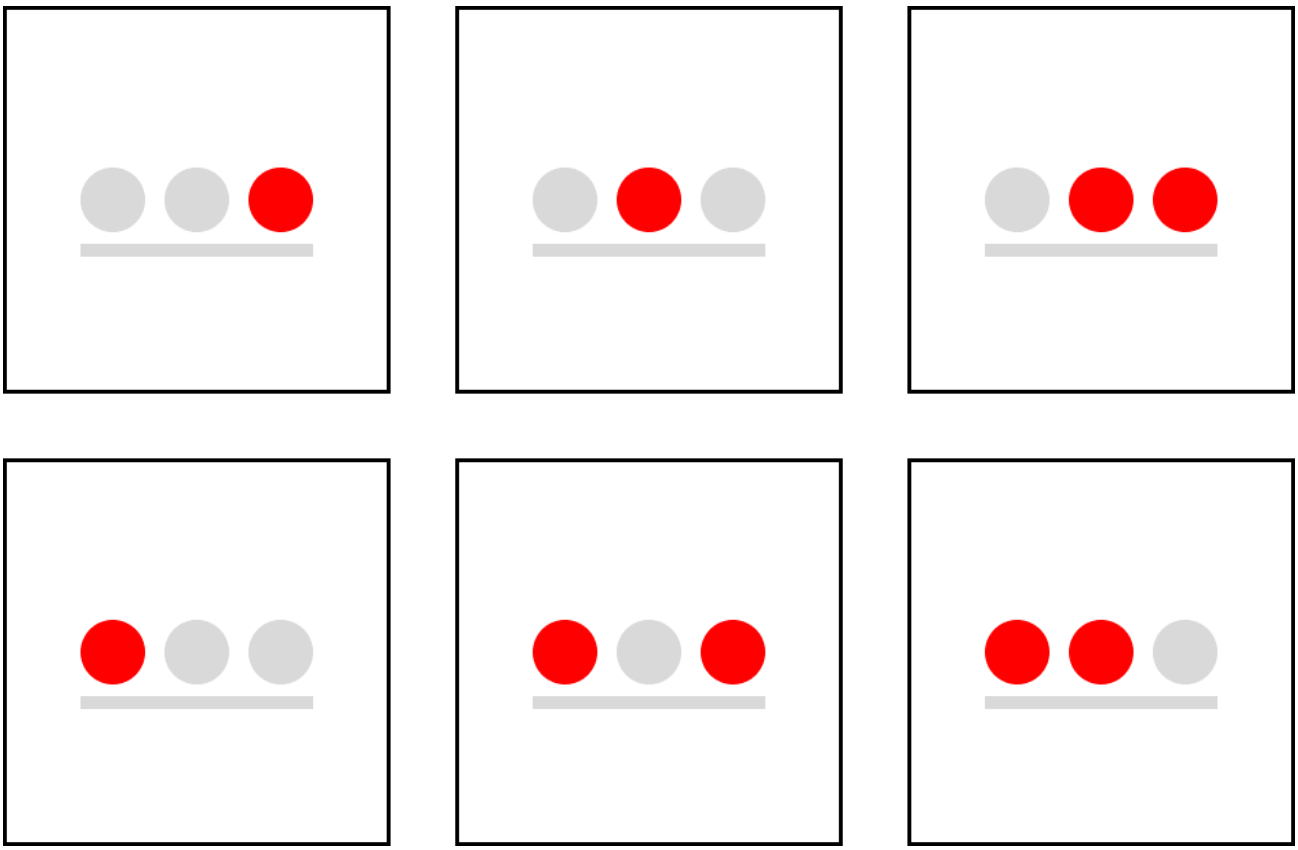
Kao ideju za prvi korak digitalizacije društvenih igara najbolje je početi od najosnovnijeg predmeta koji se nalazi u skoro svakoj društvenoj igri.. kockom! Naime kocka se temelji na ugrađenim mikroupravljačem Arduino Nano-u na koji je spojen senzor pokreta. Pri podražaju tog senzora tj. pri detekciji pokreta kocka će ispisati nasumice izabran (eng. *random*) broj od 1 do 6 pomoću svjetlećih dioda u binarnom obliku. Gdje upaljena dioda tj. dioda koja u tom trenutku svijetli predstavlja 1 dok ugašena predstavlja 0. Tako bi npr. broj 1 bio prikazan kao (0,0,1), broj 2 (0,1,0) itd.

1.1. Zadatak završnog rada

Zadatak završnog rada je razvoj i izrada elektroničke kocke temeljene na mikroupravljaču.

2. KOMPONENTE

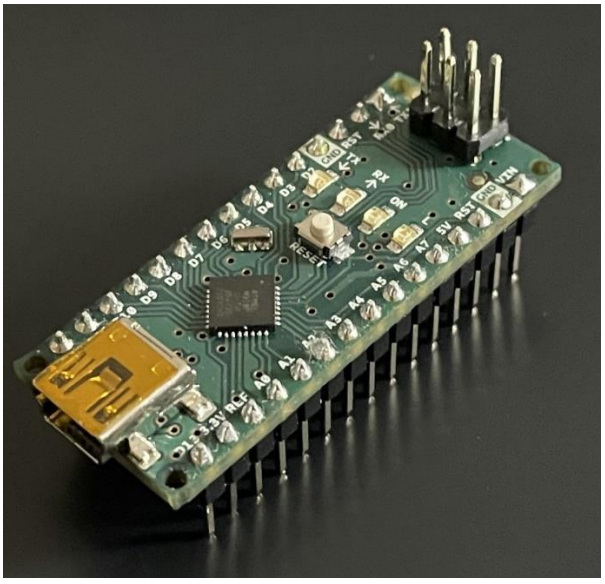
Elektronička kocka zamišljena je da se pri uključivanju Arduino Nano razvojne pločice senzor akcelerometar resetira na 0 kako bi se postavile prave početne vrijednosti te se sustav postavlja za vrtnju glavne petlje. Naime unutar glavne petlje senzor sprema trenutne vrijednosti koje se uspoređuju sa vrijednostima iz prošlog ciklusa petlje sa određenim odstupanjem. Ako se vrijednosti razlikuju pokreće se animacija za očitavanje koja traje minimalno 4 sekunde te se resetira sve dok se kocka ne prestane kretati. Tada se generira nasumice odabran (eng. *random*) broj koji se u binarnom obliku ispisuje pomoću svjetlećih dioda (Sl. 2.1.). Prilikom ispisa nasumice odabranog broja postavljaju se nove početne vrijednosti koje će se vrednovati sa trenutnima prilikom pomaka.



Slika 2.1. Prikaz ispisa broja u binarnom obliku pomoću svjetlećih dioda

2.1. Arduino Nano

Arduino Nano je vrlo mala tiskana pločica zasnovana na ATmega328P mikroupravljaču. Dimenzije mu iznose svega 45x18 mm što ga čini vrlo atraktivnim za integraciju u volumenom manje projekte. Nano se može napajati putem Mini-B USB konekcije, ne reguliranim vanjskim napajanjem od 6-20V (nožica 30) ili reguliranim vanjskim napajanjem (nožica 27). Njegov mozak tj. mikroupravljač ATmega328 sadrži 32KB brze memorije (eng. *flash memory*) od koje je 2KB rezervirano za *bootloader*. Također ima i 2KB SRAM-a i 1KB EEPROM-a.



Slika 2.2. Arduino Nano

2.1.1. Ulaz i izlaz

„Arduino Nano ima 14 digitalnih nožica (eng. pin) koje se mogu koristiti za input ili output i rade na naponu od 5 volta. Svaka nožica može pružiti ili primiti maksimalno 40mA i ima unutarnji pull-up otpornik (isključen u zadanim postavkama) od 20-50 kOhm-a. [1]“

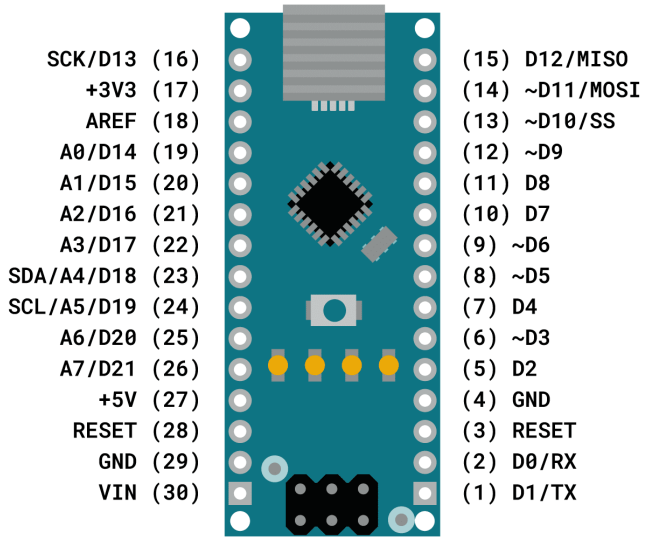
Osim tih neke od nožica imaju specijalizirane funkcije:

- Serijski: nožice 0 (RX) i 1 (TX) koji se koriste za primanje (RX) i prijenos (TX) TTL serijskih podataka. Ove nožice su spojene na odgovarajuće nožice FTDI USB-to-TTL serijskog čipa

- Vanjski prekidi (eng. *external interrupts*): nožice 2 i 3. Ovi prekidi mogu se pokrenuti tijekom niske vrijednosti, tijekom uspona ili pada vrijednosti ili promjene u vrijednosti
- PWM (pulsno-širinska modulacija eng. pulse-width modulation): nožice 3, 5, 6, 9, 10 i 11 koji osiguravaju PWM izlaz
- SPI: nožice 10 (SS), 11(MOSI), 12(MISO), 13 (SCK) koji služe sa SPI komunikaciju
- LED: nožica 13 koji služi za svjetleću diodu ugrađenu u Nano

„Nano također ima i 8 analognih nožica od kojih svaka pruža 10 bita rezolucije tj. 1024 različite vrijednosti. Rade na rasponu do 5 volti ali je moguće i promijeniti gornju granicu raspona pomoću ugrađene funkcije. Analogne nožice 6 i 7 se također mogu koristiti i kao digitalni. [1]“

Osim toga nožice A4 (SDA) i A5 (SCL) služe za I2C (TWI) komunikaciju koristeći *Wire* biblioteku.



Slika 2.3. Arduino Nano pinout [11]

2.1.2. Komunikacija

Arduino Nano ima nekoliko vrsta komunikacije koja služi za komunikaciju sa računalom i drugim mikroupravljačima. ATmega328 pruža UART TLL (5V) serijsku komunikaciju preko nožica 0 (RX) i 1 (TX). „Komunikacija se ostvaruje sa FTDI FT232RL čipom preko USB i FTDI upravljačkih

programa (eng. *drivers*) uključenih u Arduino softver, pružaju virtualni komunikacijski priključak (eng. *COM port*) softveru na računalu. Arduino softver sadržava serijski monitor koji omogućuje interakciju između korisnika i Arduino koristeći tekstualne podatke. Prilikom prijenosa podatak RX i TX svjetleće diode su indikator prijenosa podataka između FTDI čipa i USB spojenog na računalo. [8],, Za serijsku komunikaciju preko digitalni nožica koristi se *SoftwareSerial* paket te nema indikator prijenosa kao pri komunikaciji putem USB. Atmega328 također podržava I2C i SPI komunikaciju koja je podržana od strane Arduino softvera i olakšana sa Wire paketom.

2.1.3. Programiranje

„Arduino Nano se najčešće programira sa Arduino softverom (Arduino IDE). Programiranje je osigurano sa prethodno ugrađenim *bootloader*-om koji omogućava učitavanja novog koda bez korištenja dodatnog programatora. Kod se piše C ili C++ programskim jezikom uz dodatne specijalne metode i funkcije. Arduino program naziva se skica (eng. *sketch*) i sprema se u .ino datoteke te se obrađuje i prevodi u strojni jezik. [2]“

Skica se većinom sastoji od 2 glavne metode, *Setup()* i *Loop()* te također može sadržavati i ostale metode napisane od strane autora. *Setup()* se uvijek se uvijek pokrene samo jednom i to pri pokretanju mikroupravljača. Time se u toj metodi najčešće postavljaju neke početne vrijednosti, inicijaliziraju se određene komponente na određenim nožicama i sl. Dakle, mikroupravljač se priprema za daljnje izvršavanje glavnog dijela programa.

Loop() metoda je glavni dio programa koja započinje sa svojim radom nakon izvršavanja *Setup()* funkcije. Ona se neprestano vrti sve dok je mikroupravljač uključen na napajanje.

Vrlo često ljudi pišu cijeli glavni program unutar metode *Loop()*. Time ona jako brzo postane zatrpana različitim ponavljajućim funkcijama i varijablama koje se mogu raščlaniti i razvrstati u nove metode. Time bi se razvrstala funkcionalnost koda što bi ga napravilo puno čitljivijim.

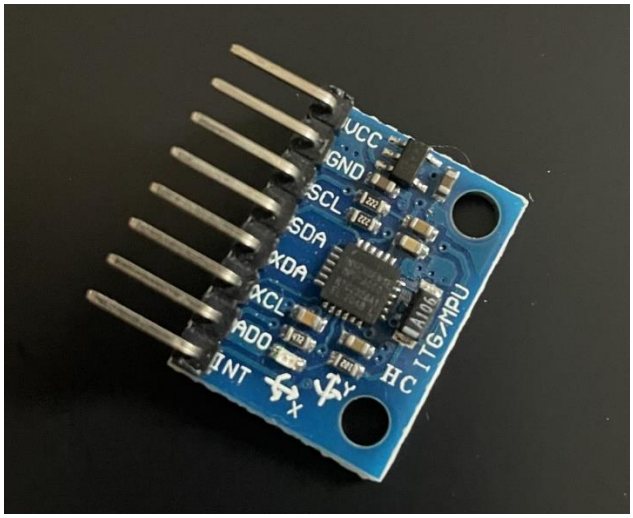
2.1.4. Automatsko resetiranje programa

Program na Arduino Nano-u se automatski resetira sa strane softvera na računalu na koje je Nano spojen prilikom očitavanja novog koda. Tako izbjegava potrebu za pritiskanjem tipke za resetiranje prije svakog očitavanja.

Jedna od linija za kontrolu toka hardvera (DTR) od FT232RL čipa je spojena na liniju za resetiranje ATmega328 preko kondenzatora od 100 nF. Kada je ta linija potvrđena tj. nisko postavljena, linija za resetiranje padne dovoljno nisko da resetira čip što omogućava učitavanje koda jednostavnim pritiskom tipke za učitavanje u Arduino okruženju. Što znači da *bootloader* može imati kratku pauzu, budući da se nisko postavljanje DTR-a može dobro koordinirati sa početkom učitavanja. Ova postavka ima i druge implikacije. Npr. kada je Nano spojen na računalo sa Max OS X ili Linuxom, resetira se svaki put kada se uspostavi veza iz softvera pomoću USB-a. Tijekom sljedeće pola sekunde, *bootloader* radi. Iako je isprogramiran da ignorira pogrešne podatke, presretnuti će prvih nekoliko bajtova podataka prilikom uspostavljanja veze. Ako skica koja radi na Nanu primi jednokratnu konfiguraciju ili druge podatke kada se prvi put pokrene, treba provjeriti dali softver sa kojim komunicira čeka sekundu nakon uspostavljanja veze i prije slanja ovih podataka..

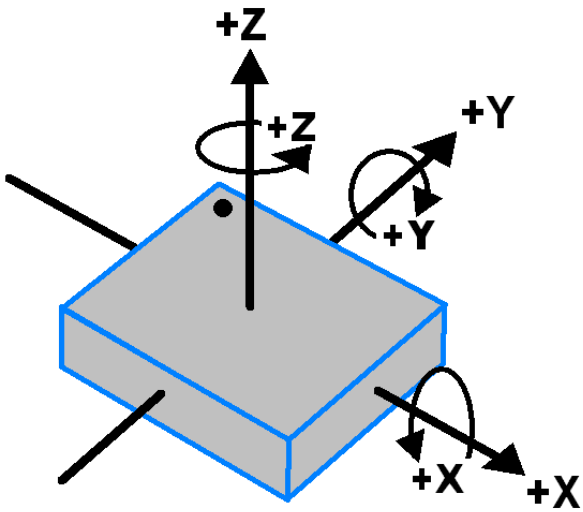
2.2. JOY-IT MPU6050

„MPU6050 je senzor kretanja koji ima 3 osi žiroskopa i 3 osi ubrzanja tj. 6 stupnjeva slobode. To mu omogućuje da mjeri do 16384 LBS/g i 131 LBS/dps. Komunikaciju sa mikrokontrolerom ostvaruje preko SCL i SDA nožica koristeći I2C konekciju. XCL i XDA nožice služe za proširenje senzora sa nekim drugim senzorom (npr. magnetometar) preko I2C komunikacije. AD0 se koristi za promjernu adrese senzora između vrijednost 0x68 i 0x69. Ostale dvije nožice su GND koji se koristi za uzemljenja te VCC preko kojeg se realizira napajanje senzora sa iznosom od 5V. [4]“



Slika 2.4. JOY-IT MPU6050

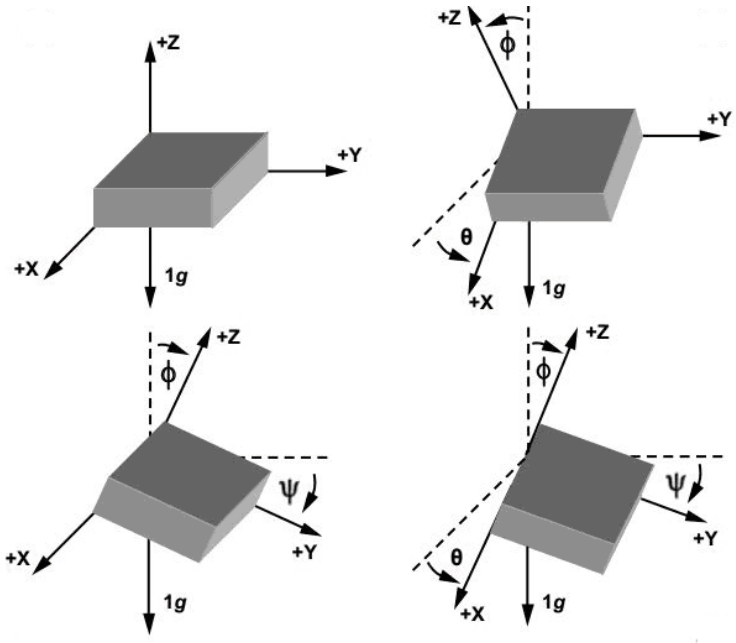
MPU6050 se sastoji od 3-osnog žiroskopa i 3-osnog akcelerometra s Micro Electro Mechanical System (MEMS) tehnologijom.



Slika 2.5. Rotacije žiroskopa duž osi [5]

„Žiroskop je uređaj koji može detektirati brzinu rotacije duž osi X, Y, Z (Sl. 2.5.). Kada su žiroskopi zakrenu oko bilo koje osi, Coriolisov efekt uzrokuje vibraciju koje je detektirana od strane MEM-a unutar MPU6050. Rezultirajući signal se pojačava, demodulira i filtrira kako bi proizveo napon koji je proporcionalan kutnoj brzini i koji se onda digitalizira koristeći 16-bitni ADC (eng. *analog-to-digital converter*) za uzorkovanje svake osi. Puni raspon je ±250, ±500, ±1000, ±2000. Kutnu brzinu svake osi mjeri u stupnjevima po jedinici sekunde. [5]“

Akcelerometar se koristi za detekciju kuta nagiba ili ugiba na X, Y, i Z osi. U stanju mirovanja detektira samo gravitacijsku silu no prilikom kretanja detektira i sve ostale sile koje djeluju na njegove osi.



Slika 2.6. Djelovanje sila na različite kutove akcelerometra [5]

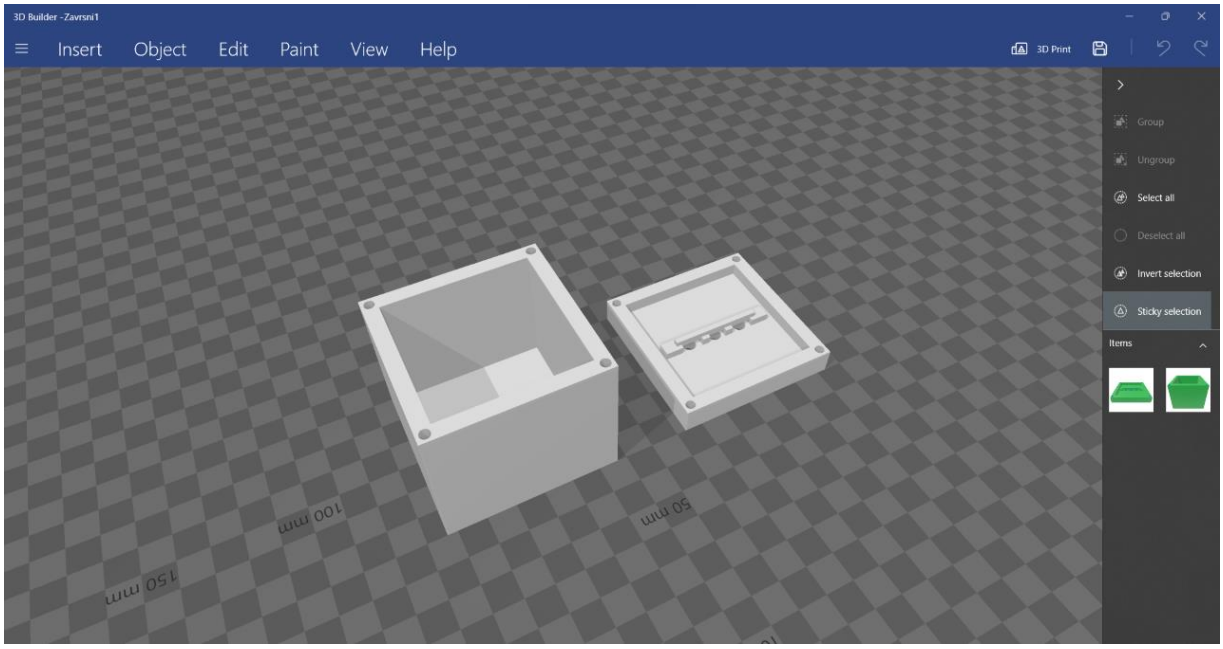
„Ubrzanje duž neke osi pomiče pokretnu masu čiji pomak debalansira diferencijalni kondenzator što rezultira sa izlaznom veličinom. Amplituda izlazne veličine proporcionalna je ubrzanju. Tu amplitudu 16-bitni ADC pretvara u digitalizirani izlaz. Raspon ubrzanja je ±2g, ±4g, ±8g, ±16g gdje g predstavlja gravitacijsku silu. Kada je akcelerometar postavljen na ravnu površinu iznositi će 0g na X i Y osi te +1g na Z osi. [5]“

Izračunavanje algoritma za obradu pokreta izvodi DMP (eng. *Digital Motion Processor*). Uzima podatke iz žiroskopa, akcelerometra i dodatnog vanjskog senzora poput magnetometra i procesira ih. Na taj način senzor uklanja trošenje procesorske snage miktrokontrolera za obradu podataka pokreta. Rezultati obrade čitaju se iz DPM registara.

MPU6050 također ima ugrađeni i senzor temperature čiji je izlaz digitalizira ADC. Podatci senzora temperature mogu se pročitati iz registra podataka senzora.

2.3. Kućište

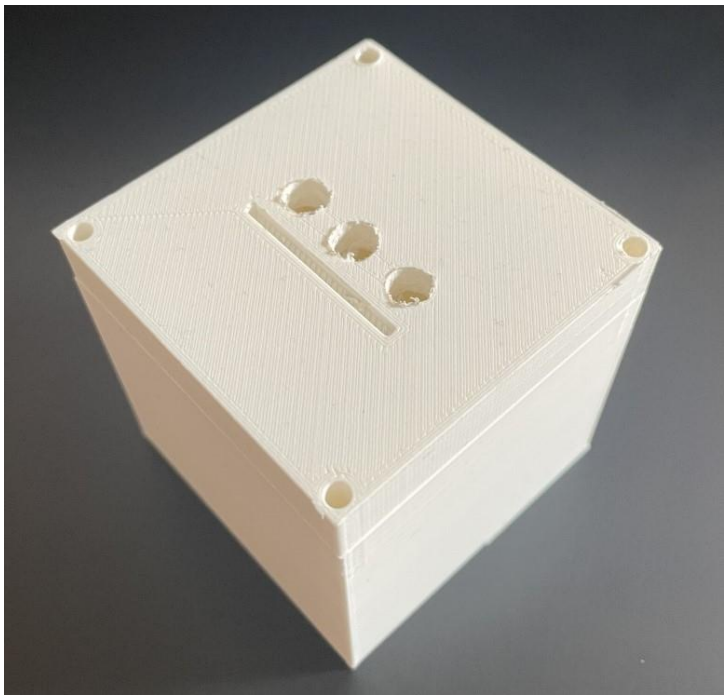
Kućište je dizajnirano unutar Microsoft-ovog programa 3D builder. 3D builder je jedan od najjednostavniji programa za modeliranje jer pokriva samo nekakve osnovne funkcionalnosti 3D modeliranja te je korisničko sučelje (eng. *user interface*) jako jednostavno. Sve funkcionalnosti se nalaze odmah unutar izbornika neke od kojih su, Umetni (eng. *Insert*) za dodavanje osnovnih objekata poput kocki, cilindra, piramide i sl., Objekt (eng. *Object*) pomoću kojeg se kopira, briše, udvostručava, mjeri odabrani objekt. Uredi (eng. *Edit*) je izbornik koji se najviše koristi unutar programa, on služi za oblikovanje, spajanje, oduzimanje dijelova objekta sa drugim objektima. Boja (eng. *Paint*) izbornik služi za bojanje i teksturiranje objekata dok Pregled (eng. *View*) izbornik služi za namještanje prikaza scene.



Slika 2.7. Scena unutar 3D buildera.

Što se tiče same kocke ona se sastoji od dva dijela koji zajedno imaju dimenzije 60x60x60mm a debljina zidova im je 5mm što samim time znači da je unutrašnjost kocke 50x50x50mm. Donji dio kocke je totalno šupalj dok gornji ima rupice za diode te potporu na koju se oslanjaju.

Slika 2.8. Isprintana kocka – otvorena



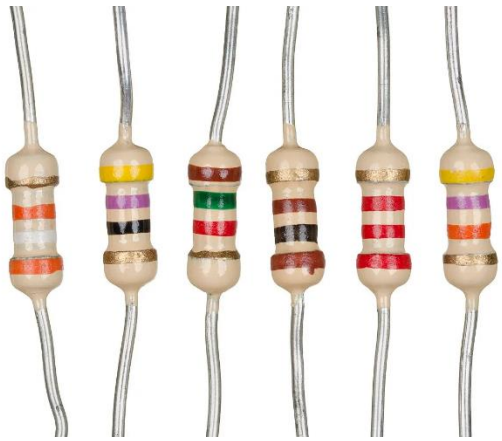
Slika 2.9. Isprintana kocka – zatvorena

2.4. Ostale komponente

„Svjetleća dioda ili LED (eng. *led-emitting diode*) je poluvodički izvor svjetlosti koji emitira svjetlost kada kroz nju teče struja. Elektroni u poluvodiču prelaze u elektronske rupe oslobađajući energiju tj. svjetlost. [7]“ Koriste se u ekranima, svjetlosnim rješenjima ili predstavljaju pokaznike npr. ako stroj radi ili ne radi.

Slika 2.7. Svjetleća dioda

„Otpornik je pasivna električna komponenta koja se koristi za smanjenje protoka struje, podešavanje razine signala, za podjelu napona i sl. Funkcija otpornika je određena njegovim otporom. Uobičajeni komercijalni otpornici proizvode se u rasponu od 9 redova veličine koje su bojom naznačene na komponenti. Najčešće ih možemo pronaći u električnim mrežama i krugovima te su sveprisutni u električnim uređajima. [9]„



Slika 2.8. Otpornici različitih vrijednosti otpora

3. HARDVERSKA IZVEDBA PROJEKTA

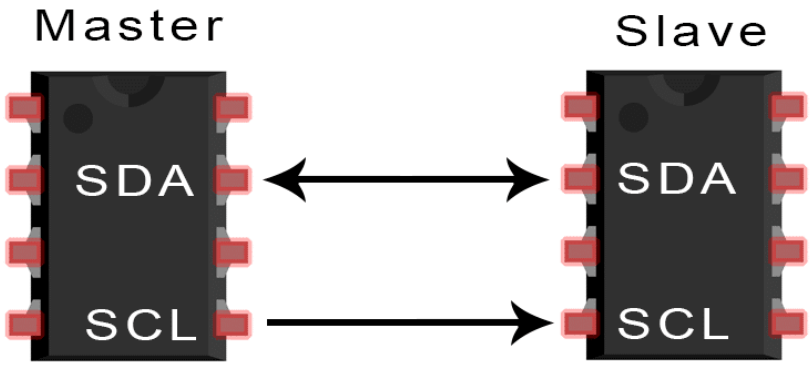
Svi dijelovi nalaze se unutar kućišta, tj. kocke te su time nevidljivi korisnicima. Senzor i Arduino Nano povezani su preko SDA i SCL nožica te komuniciraju serijski koristeći I2C protokol. Arduino je također povezan sa 3 svjetleće diode preko otpornika.

Izbor otpornika za spoj svjetlećih dioda određen je formulama (1-1) i (1-2)

$$R_s = \frac{v_s - v_F}{I_F} = \frac{5v - 1{,}8v}{12mA} = \frac{3{,}2}{12 \times 10^{-3}} = 266{,}67 \qquad (1-1)$$

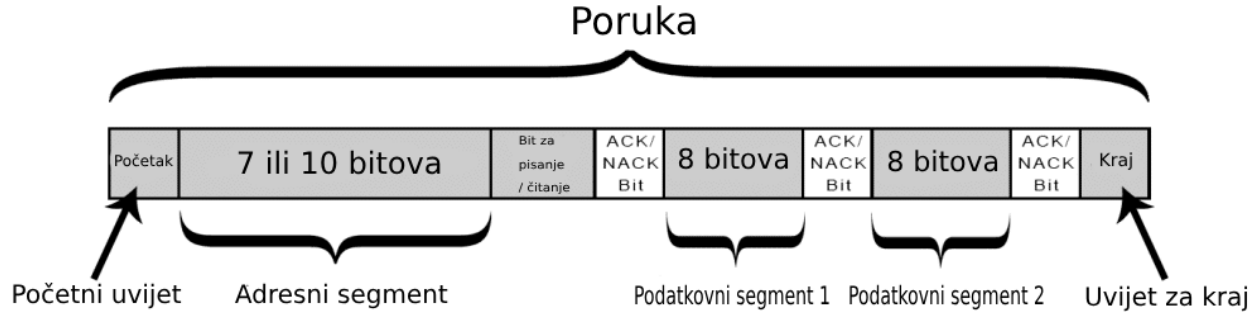
$$R_s = \frac{v_s - v_F}{I_F} = \frac{5v - 1{,}8v}{20mA} = \frac{3{,}2}{20 \times 10^{-3}} = 160 \qquad (1-2)$$

Gdje je R_s – otpor otpornika, v_s – napon na digitalnoj nožici, v_F – pad napona na diodi, I_F – optimalna amperaža na kojoj dioda radi. Za crvenu diodu maksimalna struja koja može kontinuirano prolaziti kroz nju bez oštećenja iznosi 20mA. Prema tome odabran je raspon struje koji je odgovara diodi. Budući da Nano na digitalnim nožicama daje napon od 5V a crvena svjetleća dioda ima pad napona od prilike 1.8V. Razlikom istih dobije se napon od 3.2V i kada to se on podjeli sa poželjnim amperažama na kojima dioda radi dobije se raspon otpora od 160ohma do 266.67ohma. Otpornici unutar tih veličina su u optimalnom rasponu, koristeći otpornik višeg otpora smanjuje svjetlinu dok se otpornici manjeg otpora ne preporučuju kako ne bi došlo do oštećena svjetleće diode. Stoga je odabran otpornik od 220ohma.



Slika 3.1. Jednostavni prikaz I2C komunikacije [6]

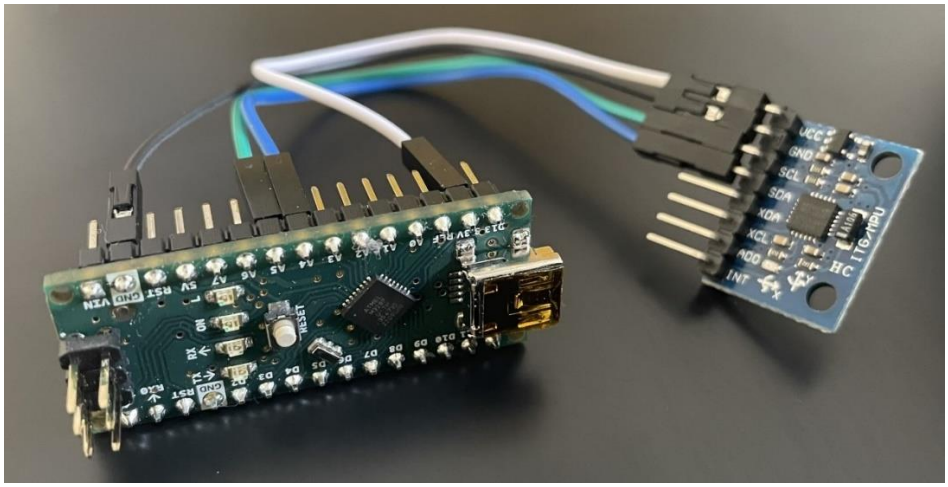
I2C je serijski komunikacijski protokol koji komunicira putem SDA (eng. *Serial Data*) i SCL (eng. *Serial Clock*) sabirnica. Radi na principu razmjenjivanja poruka između master-a i slave-a gdje master (koji može imati više slave-ova) kontrolira tok komunikacije. Svaka poruka se sastoji od više segmenata a to su: stanje za početak komunikacije, adresa, bit za pisanje/čitanje, podatci, ACK/NACK bit i stanje za kraj komunikacije. ACK (eng. *acknowledge*) i NACK(eng. *no-acknowledge*) bitovi su bitovi potvrde koji primatelj poruke vraća pošiljatelju da potvrdi dali je segment poruke ispravno primljen. Izlaz svih bitova poruke je sinkroniziran sa uzorkovanjem bitova pomoću signala takta kojeg master i slave dijele a master kontrolira.



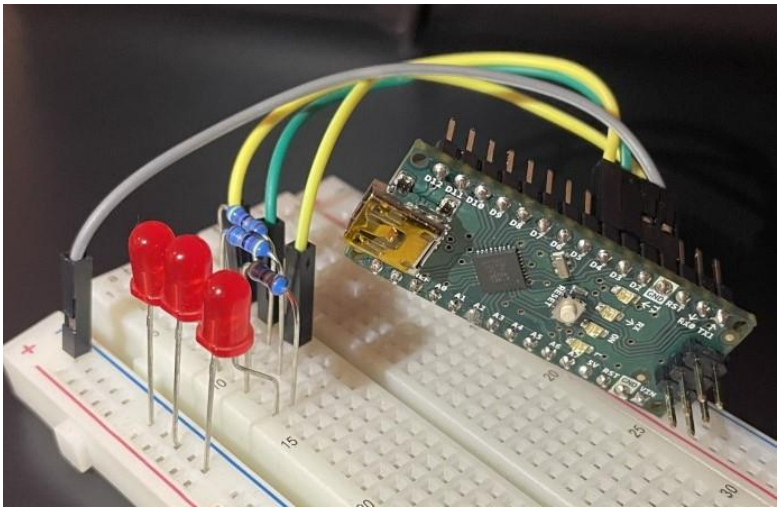
Slika 3.2. Grafički prikaz segmenata poruke [6]

„Funkcionira na principu da master pošalje poruku svim slave-ovima koji onda uspoređuju primljenu adresu iz poruke sa svojom adresom. Ako se adresa poklapa, slave pošalje nazad masteru ACK bit. Zadnji bit u toj adresi određuje dali master želi poslati podatke slave-u ili primiti podatke od njega. Nakon što master primit ACK bit od slave-a on počinje slati/primati prvi segment podataka. Svaki segment podatka je dugačak 8 bita i započinje sa najvažnijim bitom. Na kraju svakog segmenta podataka šalje se ACK ili NACK bit te na kraju kada su svi segmenti razmijenjeni šalje se stanje za kraj komunikacije. [6]“

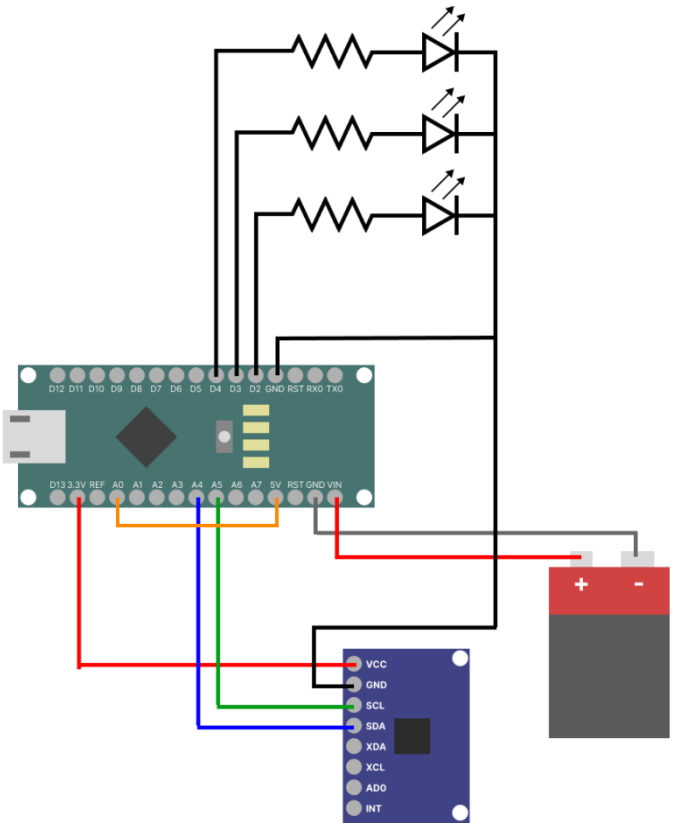
Dakle Arduino je povezan sa senzorom koristeći serijsku komunikaciju. Sa svakom svjetlećom diodom je pojedinačno spojen preko otpornika te napaja se baterijom od 9 volti preko VIN pina. Također postoji spoj A0 pina sa pinom za napajanje 5V. Taj spoj isključivo služi za inicijalizaciju generatora nasumice odabranih brojeva (eng. *random*) tako što sa nepovezanom 5V pinu pojavljuje električni šum nastao iz njegove okoline.



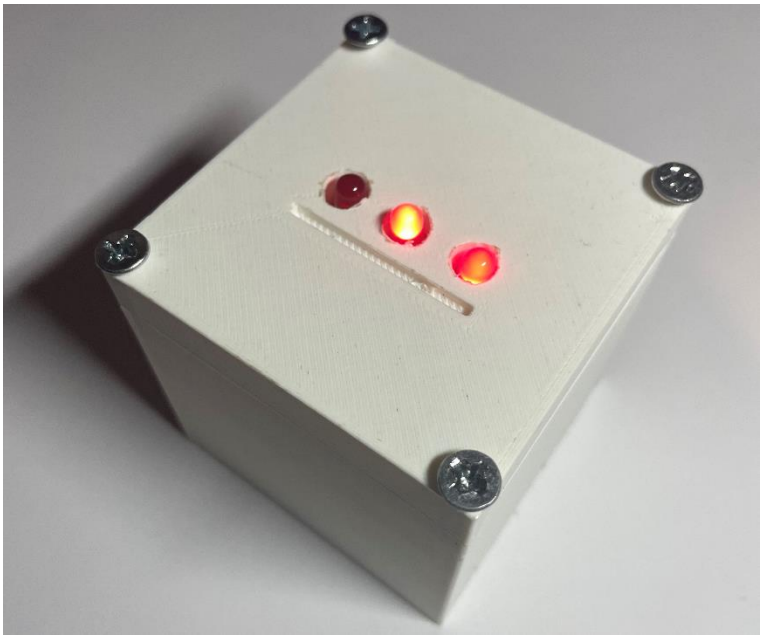
Slika 3.3. Spoj mikroupravljača Arduino Nano i MPU6050 senzora



Slika 3.4. Spoj mikroupravljača Arduino Nano i svjetlećih dioda



Slika 3.5. Nacrt kompletnog spoja



Slika 3.4. Elektronička kocka

4. SOFTVERSKA IZVEDBA PROJEKTA

Na samom početku programa referenciran je Wire.h [3] paket koji služi za I2C komunikaciju te su definirane određene varijable koje će se koristi kroz program. U *Setup()* metodi senzor se resetira kako bi mogao primiti nove početna vrijednosti tako što se u njegov registar na adresi 0x6B postavlja 0, tj. 0x00. Potom se inicijaliziraju svjetleće diode te se vrti petlja za postavljanje varijable generatora nasumičnih brojeva 64 puta. Prilikom svake iteracije petlje izvodi se XOR (bitovski ekskluzivni ILI) operacija nad trenutnoj vrijednosti *seed* varijable i rezultatom zbrajanja analognog očitanja električnog šuma pina A0, trenutnim milisekundama te trenutnim mikrosekundama. Iteracija se odvija svakih 50 milisekundi zbog naredbe *delay(50)* koja osigurava promjenu vrijednosti očitanja šuma na A0 pinu. Te na samom kraju se postavlja generator pomoću navedene varijable.

```
#include "Wire.h"

const int MPU = 0x68; //Adresa MPU6050 senzora
float AccX, AccY, AccZ; //Varijable za spremanje trenutne vrijednosti akcelerometra
float oldAccX, oldAccY, oldAccZ; //Varijable za spremanje početne vrijednosti akcelerometra

bool isSaveOld = true; //Varijabla koja omogućava spremanje početnih vrijednosti
bool isActivity = true; //Varijabla za provjeru aktivnosti
int randNumber; //Varijabla za spremanje random broja

const int fLED = 4; //prvi LED
const int sLED = 3; //drugi LED
const int tLED = 2; //treći LED

float elapsedTime, currentTime, previousTime; //Varijable ya mjerenje vremena

void setup() {
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission(true);

  pinMode(fLED, OUTPUT);
  digitalWrite(fLED, LOW);
  pinMode(sLED, OUTPUT);
  digitalWrite(sLED, LOW);
  pinMode(tLED, OUTPUT);
  digitalWrite(tLED, LOW);

  long seed = 0;
  for(int i = 0; i<64; i++){
    seed ^= analogRead(A0) + millis() + micros();
    delay(50);
  }

  randomSeed(seed);
}
```

Slika 4.1. Definicija varijabli, pripremanje senzora za rad i postavljanje svjetlećih dioda

Unutar *Loop()* funkcije započinje se čitanje podataka sa akcelerometra. Kako bi se podatci iščitali iz registara potrebno je poznavati njihove adrese, tj. iščitati ih iz podatkovnog lista (eng. *datasheet*). Naime, za svaku os podatci su spremljeni u dva bajta ili dva registra. „Zato moramo odabrati čitanje iz 6 registara započevši sa prvim u redu, registru na adresi 0x3B. [10]“

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT[7:0]							

Slika 4.2. Tablica adresa registara akcelerometra [10]

```
void loop(){
  Serial.begin(9600);
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true);
  readAcc(isSaveOld); //Početna vrijednost = true

  timeCalculator();
  delay(50);
}
```

Slika 4.3. Loop metoda

Zatim se pozivaju metode za čitanje iz registara i računanje vremena. U slučaju prvog ciklusa petlje metoda za čitanje iz registara spremi početne vrijednosti u varijable namijenjene za njih te postavlja vrijednost varijable za spremanje početnih vrijednosti na FALSE. „Pročitane vrijednosti se prije spremanja dijele sa senzitivnošću, koja ima zadanu vrijednost od 16384, kako bi dobili očekivanu vrijednost. [10]„

AFS_SEL	Full Scale Range	LSB Sensitivity
0	±2*g*	16384 LSB/*g*
1	±4*g*	8192 LSB/*g*
2	±8*g*	4096 LSB/*g*
3	±16*g*	2048 LSB/*g*

Slika 4.4. Tablica postavki akcelerometra [10]

Za sve ostale cikluse petlje trenutne vrijednosti sprema u varijable namijenjene za njih. Tada provjerava odstupanje tih vrijednosti unutar metoda *didXXXXChange()*. Prilikom odstupanja za 1g u slučaju da je varijabla za aktivnost, *isActivity*, postavljena na FALSE, sprema se trenutno vrijeme odstupanja i omogućava se nanovo čitanje početnih vrijednosti postavljanjem varijable *isSaveOld* na TRUE te se zatim *isActivity* postavlja na TRUE. Budući da se varijabla *isActivity* prilikom svakog

ispisa nasumice odabranog broja postavlja na FALSE. Početne varijable će se spremati nakon završetka svakog ciklusa.

```
void readAcc(bool saveOld){
  if(saveOld){
    oldAccX = (Wire.read() << 8 | Wire.read()) / 16384.0;
    oldAccY = (Wire.read() << 8 | Wire.read()) / 16384.0;
    oldAccZ = (Wire.read() << 8 | Wire.read()) / 16384.0;
    isSaveOld = false;
  }
  else if(!saveOld){
    AccX = (Wire.read() << 8 | Wire.read()) / 16384.0;
    AccY = (Wire.read() << 8 | Wire.read()) / 16384.0;
    AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0;
    if(didAccXChange() || didAccYChange() || didAccZChange()){
      if(!isActivity){
        previousTime = millis();
        isSaveOld = true;
      }
      isActivity = true;
    }
  }
}

bool didAccXChange(){
  if(oldAccX + 1 < AccX || oldAccX - 1 > AccX){
    return true;
  }
  return false;
}

bool didAccYChange(){
  if(oldAccY + 1 < AccY || oldAccY - 1 > AccY){
    return true;
  }
  return false;
}

bool didAccZChange(){
  if(oldAccZ + 1 < AccZ || oldAccZ - 1 > AccZ){
    return true;
  }
  return false;
}
```

Slika 4.5. Metoda čitanja podataka iz registara i funkcije odstupanja

Druga metoda unutar *Loop()* metode je metoda za mjerenje vremena. Ona sprema trenutno vrijeme. To trenutno vrijeme oduzima se sa vremenom odstupanja akcelerometra i sprema ga u varijablu proteklog vremena koja se sprema u sekundama podjelom sa 1000. Ta varijabla proteklog

vremena služi za resetiranje vremena zato što se kocka može kotrljati nekoliko sekundi tijekom koje bi si se u varijablu vremena odstupanja spremalo vrijeme u tom trenutku. Dakle bez tog resetiranja vremena rezultat bi se proizveo prije prestanka kotrljanja. Zatim se provjerava jeli prošlo više od 4 sekunde nakon posljednjeg odstupanja te ako je poziva se metoda za ispis nasumice odabranog broja na kocki, ako nije vrti se animacija za očitavanje.

```
void timeCalculator(){
  currentTime = millis();
  elapsedTime = (currentTime - previousTime)/1000;
  if(elapsedTime > 4 && isActivity){
    printRandomNumber();
  }else if(isActivity){
    loadingAnimation();
  }
}
```

Slika 4.6. Metoda mjerenja vremena

Animacija za očitavanje predstavlja jednosmjerno paljenje i gašenje svjetlećih dioda.

```
void loadingAnimation(){
  digitalWrite(fLED, HIGH);
  delay(100);
  digitalWrite(sLED, HIGH);
  delay(100);
  digitalWrite(tLED, HIGH);
  digitalWrite(fLED, LOW);
  delay(100);
  digitalWrite(sLED, LOW);
  delay(100);
  digitalWrite(tLED, LOW);
}
```

Slika 4.7. Metoda animacije

Metoda za ispis nasumice odabranog broja na kocki proizvodi nasumice odabran broj od 1 do 6 kojeg prosljeđuje na funkciju za ispis pomoću svjetlećih dioda. Te na kraju postavlja aktivnost na *FALSE* te time završava ciklus.

```
void printRandomNumber(){
  if(isActivity){
    randNumber = random(1,7);
    printNum(randNumber);
    isActivity=false;
  }
}
```

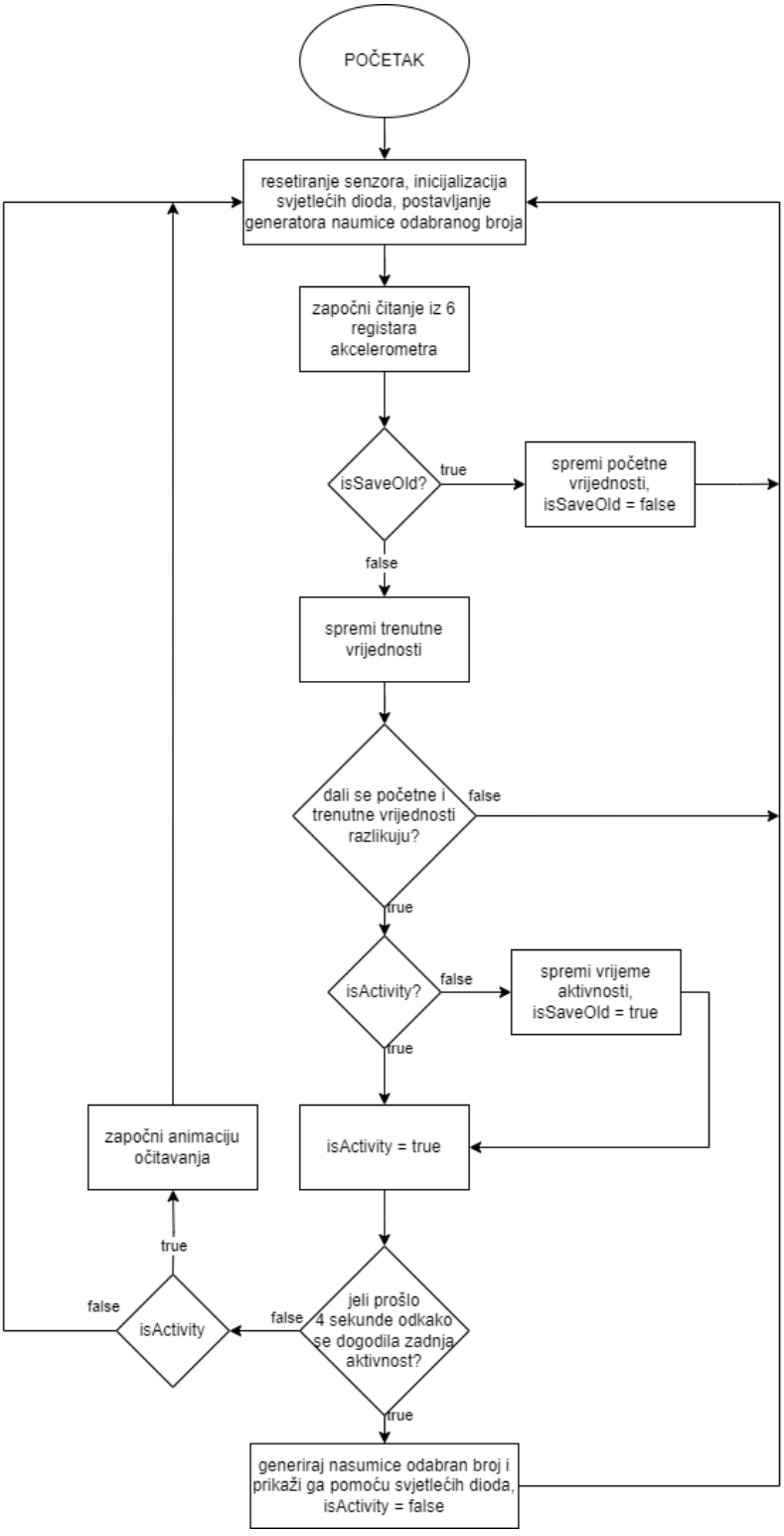
Slika 4.8. Metoda za ispis nasumice odabranog broja

Na samom kraju ispis broja pomoću svjetlećih dioda odrađuje se u binarnom obliku pomoću *switch-case* naredbe.

```
void printNum(int genNum){
  switch(genNum){
    case 1:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, LOW);
      break;
    case 2:
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, LOW);
      break;
    case 3:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, LOW);
      break;
    case 4:
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, HIGH);
      break;
    case 5:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, HIGH);
      break;
    case 6:
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, HIGH);
      break;
  }
}
```

Slika 4.9. Metoda za ispis nasumice odabranog broja pomoću svjetlećih dioda

Ukratko, nakon postavljanja mikroupravljača za rad unutar petlje se pri svakom ciklusu čitaju podatci iz registara akcelerometra. U svakom ciklusu kada je varijabla za spremanje početnih vrijednosti postavljena na TRUE, one se spremaju te se ista varijabla na FALSE. U suprotnom, spremaju se trenutne vrijednosti koje se uz odstupanje vrednuju sa prijašnje spremljenim početnim vrijednostima. U slučaju da se trenutna vrijednost dovoljno razlikuje od početne vrijednosti i varijabla koja ukazuje na aktivnost je postavljena na FALSE, sprema se vrijeme aktivnosti i omogućuje se čitanje novih početnih vrijednosti u sljedećem ciklusu postavljenjem varijable na TRUE, također se varijabla za aktivnost postavlja na TRUE. Nakon toga vrijeme aktivnosti se oduzima sa trenutnim vremenom rezultirajući u varijabli koja odbrojava koliko je prošlo sekundi od zadnje aktivnosti akcelerometra. Dok ta varijabla ne iznosi više od 4 sekunde, u petlji se obrađuje metoda za animaciju učitavanja. Kada prođe 4 sekunde, generira se nasumice odabran broj koji se ispisuje u binarnom obliku pomoću svjetlećih dioda. Varijabla za aktivnost se postavlja na FALSE te time omogućuje nanovo spremanje početnih vrijednosti koje će se vrednovati u novom ciklusu.



Slika 4.10. Prikaz toka algoritma

5. TESTIRANJE

Testiranje je izvedeno na otvorenoj kocki povezanoj na laptop preko USB-a radi korištenja serijskog monitora na Arduino IDE platformi. Na početku programa definirane su 7 novih varijabli od kojih 6, (test1, test2, …, test6), služe za zbrajanje pojave pojedinačnih brojeva od 1 do 6 te jedna varijabla, testCounter, koja sprema trenutni broj izvršavanja programa unutar glavne petlje. Varijabla testCounter povećava se za 1 prilikom svakog izvršavanja funkcije *printRandomNumber()* te prilikom 200. ciklusa ispisuje broj ponavljanja ostalih varijabli. Ostale varijable uvećavaju se za 1 prilikom ispisa njihovog određenog broja unutar *printNum()* funkcije.

```
#include "Wire.h"

const int MPU = 0x68;
float AccX, AccY, AccZ;
float oldAccX, oldAccY, oldAccZ;

bool firstAcc = true;
bool isActivity = true;
int randNumber;

int testCounter = 0;
int test1 = 0;
int test2 = 0;
int test3 = 0;
int test4 = 0;
int test5 = 0;
int test6 = 0;

const int fLED = 4;
const int sLED = 3;
const int tLED = 2;

float elapsedTime, currentTime, previousTime;
```

Slika 5.1. Definiranje testnih varijabli

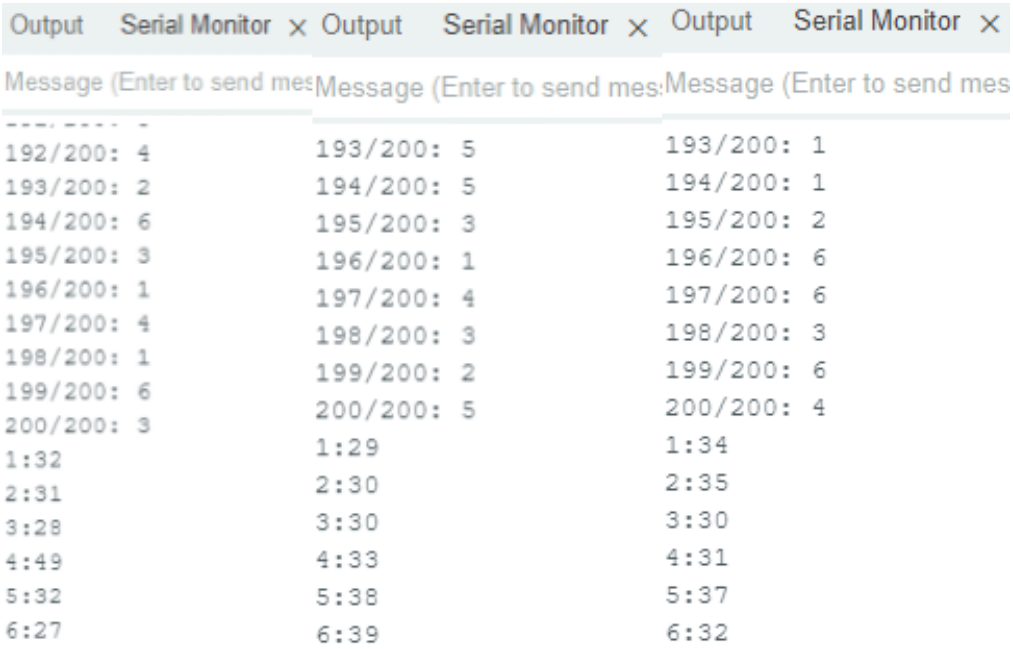
```
void printRandomNumber(){
  if(isActivity){
    randNumber = random(1,7);
    Serial.print(testCounter);Serial.print("/200: ");Serial.println(randNumber);
    testCounter++;
    if(testCounter == 200){
      Serial.print("1:");Serial.println(test1);
      Serial.print("2:");Serial.println(test2);
      Serial.print("3:");Serial.println(test3);
      Serial.print("4:");Serial.println(test4);
      Serial.print("5:");Serial.println(test5);
      Serial.print("6:");Serial.println(test6);
    }
    printNum(randNumber);
    isActivity=false;
  }
}
```

Slika 5.1. Uvećanje testCounter varijable te ispis pojedinačnih testnih varijabli

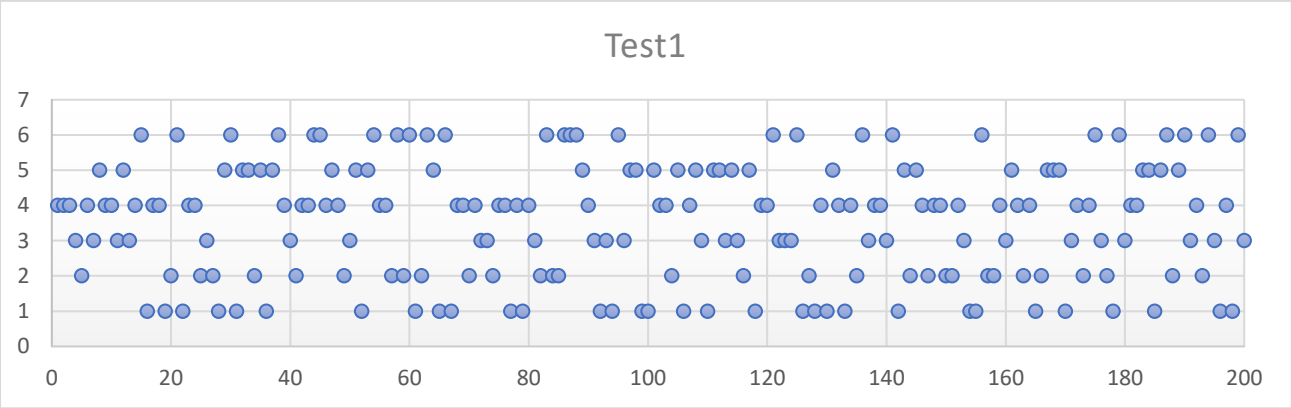
```
void printNum(int genNum){
  switch(genNum){
    case 1:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, LOW);
      test1++;
      break;
    case 2:
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, LOW);
      test2++;
      break;
    case 3:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, LOW);
      test3++;
      break;
    case 4:
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, HIGH);
      test4++;
      break;
    case 5:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, HIGH);
      test5++;
      break;
    case 6:
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, HIGH);
      test6++;
      break;
  }
}
```

Slika 5.3. Uvećanje pojedinačnih testnih varijabli

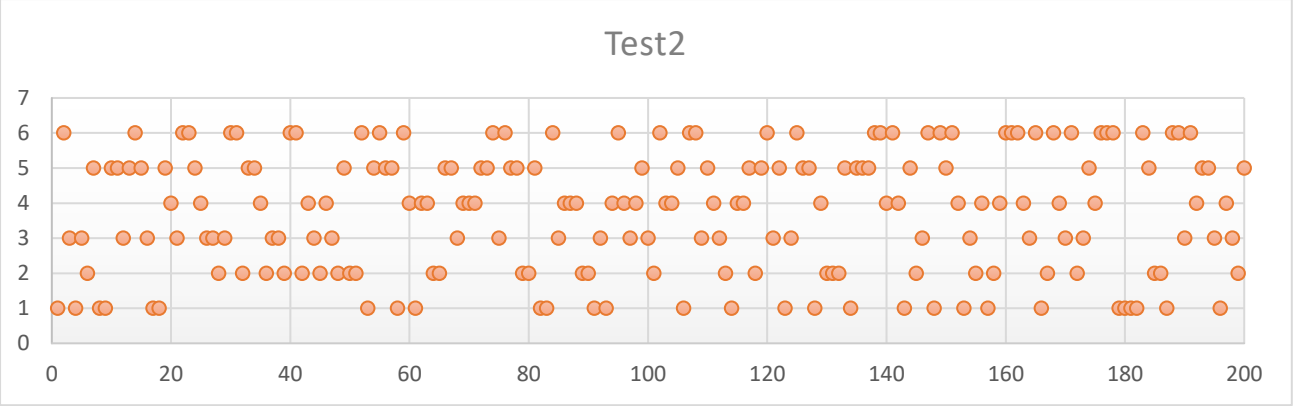
Testiranje se odvilo 3 puta. Konačni rezultati ukazuju da će se prilikom svakog pokretanja kocke postaviti drugačiji uvjeti generatora nasumice izabranih brojeva te se time postiglo ne predvidivo očitanje broja na kocki (Sl. 5.4.). U idealnom slučaju učestalost pojavljivanja svakog broja bio bi jednak tj. 1/6 = 0.16666667 no nažalost prava/istinita nasumičnost ne postoji.



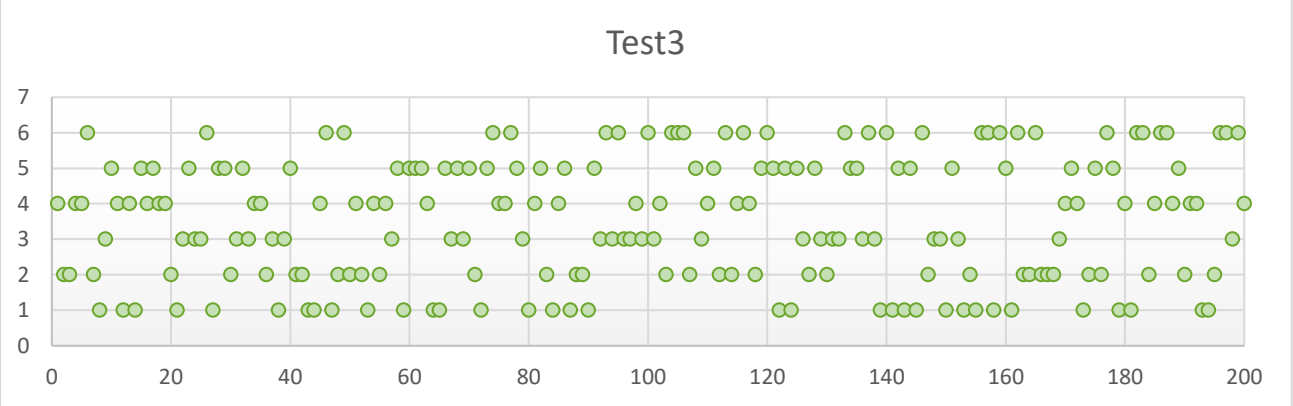
Slika 5.4. Rezultati testiranja



Slika 5.5. Grafički prikaz prvog testiranja



Slika 5.6. Grafički prikaz drugog testiranja



Slika 5.7. Grafički prikaz drugog testiranja

6. ZAKLJUČAK

U ovom završnom radu razvijena je elektronička kocka koja prilikom fizičkog podražaja ispisuje nasumice odabran broj u binarnome obliku preko svjetlećih dioda. Unutar teorijskog dijela opisane su osnovne funkcionalnosti Arduino Nano-a te Arduino platforme. Objašnjen je rad senzora MPU6050 i njegovih funkcionalnosti tj. rad akcelerometra i žiroskopa. U praktičnog dijela rada opisano je kako je on izveden, kako su komponente fizički spojene i na koji način komuniciraju te kako sustav radi kao jedna cjelina. Programski dio daje još dublji pogled na način rada elektroničke kocke i komunikacije između svake komponente. Te je nakraju testiranjem dokazana nepredvidljivost ispisa kocke.

Za razliku od običnih kocki, ova kocka je u potpunosti nepredvidiva jer se krajnji rezultat ne može namjestiti. Tj. korištenjem običnih kocki moguće je izvježbati različite tehnike bacanja kako bi se povećala šansa za poželjnim rezultatom. U slučaju elektroničke kocke, tehnika bacanja je u potpunosti zanemariva jer krajnju odluku donosi mikrokontroler sa nasumice izabranim brojem. Tijekom testiranja broj 1 se sveukupno pojavio s vjerojatnošću od 0.15833 ili 95 puta, broj 2 s vjerojatnošću od 0.16 ili 96 puta, broj 3 s vjerojatnošću od 0.14667 ili 88 puta, broj 4 s vjerojatnošću od 0.18833 ili 113, broj 5 s vjerojatnošću od 0.17833 ili 107 puta, broj 6 s vjerojatnošću od 0.16333 ili 98 puta.

LITERATURA

[1] Arduino 2022, Arduino Nano, (pristupljeno 25.06.2022)
https://store.arduino.cc/products/arduino-nano

[2] Wikipedia 2022, Arduino, (pristupljeno 25.06.2022.)
https://en.wikipedia.org/wiki/Arduino

[3] Arduino 2022, Wire, (pristupljeno 25.06.2022.)
https://www.arduino.cc/reference/en/language/functions/communication/wire/

[4] JOY-IT 2022, MPU6050, (pristupljeno 26.06.2022.)
https://joy-it.net/en/products/SEN-MPU6050

[5] ElectronicWings 2022, MPU6050 (Gyroscope + Accelerometer + Temperature) Sensor Module, (pristupljeno 26.06.2022)
https://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module

[6] BASICS OF I2C COMMUNICATION PROTOCOL, (pristupljeno 26.06.2022)
https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

[7] ElectronicsTutorials 2022, The Light Emitting Diode, (pristupljeno 27.06.2022.)
https://www.electronics-tutorials.ws/diode/diode_8.html

[8] FTDI chip 2022, FT232RL, (pristupljeno 28.06.2022.)
https://ftdichip.com/products/ft232rl/

[9] Wikipedia 2022, Resistor, (pristupljeno 30.06.2022.)
https://en.wikipedia.org/wiki/Resistor

[10] InvenSense, MPU-6000 and MPU-6050 Register Map and Descriptions Revision 4.2, 2013 (pristupljeno 30.06.2022.)
https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf

[11] Arduino Nano Board Guide (Pintout, Specifications, Comparison), (pristupljeno 31.06.2022.)
https://www.pinterest.com/pin/642818546815065475/

SAŽETAK

U ovom radu izrađena ja elektronička kocka temeljena na Arduino Nano mikroupravljaču. Uz Arduino Nano koristio se i senzor pokreta MPU6050 koji sadrži akcelerometar, žiroskop i termometar. Obrazloženo je sklopovsko rješenje te je opisana njihova I2C serijska komunikacije i *Wire.h* biblioteka potrebna za ostvarivanje te komunikacije. Osim toga sažeta je izrada 3D isprintanog kućišta. Programsko rješenje je izvedeno u obliku stalnog provjeravanja promjene vrijednosti akcelerometra. Pri detekciji pokreta u akcelerometru poziva se petlja za učitavanje koja prestaje prilikom prestanka kretanja kocke. Prilikom prestanka kretnje se ispisuje nasumice odabran broj u binarnom obliku na svjetlećim diodama. Testiranje se obavilo pomoću serijskog monitora na Arduino platformi gdje se kocka bacala 200 puta. Test je obavljen sveukupno 3 puta te je svaki prilikom 200 bacanja dao različite rezultate što ukazuje na nepredvidivost kocke.

Ključne riječi: Arduino, elektronička kocka, MPU6050, senzor pokreta, akcelerometar

ABSTRACT

Title: Electronic Dice

In this thesis, an electronic dice based on the Arduino Nano microcontroller is developed. In addition to Arduino Nano, the MPU6050 motion sensor is also used, which contains an accelerometer, gyroscope and thermometer. The circuit solution is explained together with the I2C serial communication and the Wire.h library which is necessary for implementing that communication. Additionally, the creation of 3D-printed casing is succinctly described. The software solution is realized through a continuous monitoring of the accelerometer's value changes. Upon detecting motion in the accelerometer, a loop for loading is initiated, which halts upon the dice's cessation of movement. When the movement stops, a randomly selected number is displayed in binary form on the LED's. Testing was conducted using the serial monitor on Arduino platform, with dice being rolled 200 times. The test was repeated a total of 3 times, and each set of 200 rolls yielded different results, indicating the unpredictability of the dice.

Keywords: Arduino, electronic dice, MPU6050, motion sensor, accelerometer

ŽIVOTOPIS

Denis Požarko rođen je 25.09.1999. u Vukovaru. Završio je Tehničku školu Nikole Tesle, Vukovar gdje je stekao zvanje tehničar za računalstvo. Nakon srednje upisuje Elektrotehnički fakultet u Osijeku, smjer Automatika. Posjeduje znanje i iskustvo sa Pythonom, C#, C, C++, Golang, HTML, CSS i Typescript programskim jezicima. Uz dosadašnje obrazovanje stekao je iskustvo sa programima Matlab, AutoCAD. Aktivno se služi engleskim jezikom u govoru i pismu.

PRILOZOG A: Arduino skica za elektroničku kocku

```
#include "Wire.h"

const int MPU = 0x68; //Adresa MPU6050 senzora
float AccX, AccY, AccZ; //Varijable za spremanje trenutne vrijednosti akcelerometra
float oldAccX, oldAccY, oldAccZ; //Varijable za spremanje početne vrijednosti akcelerometra

bool isSaveOld = true; //Varijabla koja omogućava spremanje početnih vrijednosti
bool isActivity = true; //Varijabla za provjeru aktivnosti
int randNumber; //Varijabla za spremanje random broja

const int fLED = 4; //prvi LED
const int sLED = 3; //drugi LED
const int tLED = 2; //treće LED

float elapsedTime, currentTime, previousTime; //Varijable ya mjerenje vremena

void setup() {
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission(true);

  pinMode(fLED, OUTPUT);
  digitalWrite(fLED, LOW);
  pinMode(sLED, OUTPUT);
  digitalWrite(sLED, LOW);
  pinMode(tLED, OUTPUT);
  digitalWrite(tLED, LOW);

  long seed = 0;
  for(int i = 0; i<64; i++){
    seed ^= analogRead(A0) + millis() + micros();
    delay(50);
  }

  randomSeed(seed);
}

void loop(){
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
```

```
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true);
  readAcc(isSaveOld); //Početna vrijednost = true

  timeCalculator();
  delay(50);
}


void readAcc(bool saveOld){
  if(saveOld){
    oldAccX = (Wire.read() << 8 | Wire.read()) / 16384.0;
    oldAccY = (Wire.read() << 8 | Wire.read()) / 16384.0;
    oldAccZ = (Wire.read() << 8 | Wire.read()) / 16384.0;
    isSaveOld = false;
  }
  else if(!saveOld){
    AccX = (Wire.read() << 8 | Wire.read()) / 16384.0;
    AccY = (Wire.read() << 8 | Wire.read()) / 16384.0;
    AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0;
    if(didAccXChange() || didAccYChange() || didAccZChange()){
      if(!isActivity){
        previousTime = millis();
        isSaveOld = true;
      }
      isActivity = true;
    }
  }
}

bool didAccXChange(){
  if(oldAccX + 1 < AccX || oldAccX - 1 > AccX){
    return true;
  }
  return false;
}

bool didAccYChange(){
  if(oldAccY + 1 < AccY || oldAccY - 1 > AccY){
    return true;
  }
  return false;
}

bool didAccZChange(){
```

```
  if(oldAccZ + 1 < AccZ || oldAccZ - 1 > AccZ){
    return true;
  }
  return false;
}

void timeCalculator(){
  currentTime = millis();
  elapsedTime = (currentTime - previousTime)/1000;
  if(elapsedTime > 4 && isActivity){
    printRandomNumber();
  }else if(isActivity){
    loadingAnimation();
  }
}

void loadingAnimation(){
  digitalWrite(fLED,HIGH);
  delay(100);
  digitalWrite(sLED,HIGH);
  delay(100);
  digitalWrite(tLED,HIGH);
  digitalWrite(fLED,LOW);
  delay(100);
  digitalWrite(sLED,LOW);
  delay(100);
  digitalWrite(tLED,LOW);
}

void printRandomNumber(){
  if(isActivity){
    randNumber = random(1,7);
    printNum(randNumber);
    isActivity=false;
  }
}

void printNum(int genNum){
  switch(genNum){
    case 1:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, LOW);
      break;
    case 2:
```

```
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, LOW);
      break;
    case 3:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, LOW);
      break;
    case 4:
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, HIGH);
      break;
    case 5:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, HIGH);
      break;
    case 6:
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, HIGH);
      break;
  }
}
```

PRILOZOG B: Arduino skica za testiranje elektroničke kocke

```
#include "Wire.h"

const int MPU = 0x68; //Adresa MPU6050 senzora
float AccX, AccY, AccZ; //Varijable za spremanje trenutne vrijednosti akcelerometra
float oldAccX, oldAccY, oldAccZ; //Varijable za spremanje početne vrijednosti akcelerometra

bool isSaveOld = true; //Varijabla koja omogućava spremanje početnih vrijednosti
bool isActivity = true; //Varijabla za provjeru aktivnosti
int randNumber; //Varijabla za spremanje random broja

int testCounter = 0;
int test1 = 0;
int test2 = 0;
int test3 = 0;
int test4 = 0;
int test5 = 0;
int test6 = 0;

const int fLED = 4; //prvi LED
const int sLED = 3; //drugi LED
const int tLED = 2; //treće LED

float elapsedTime, currentTime, previousTime; //Varijable ya mjerenje vremena

void setup() {
  Serial.begin(9600);
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
  Wire.write(0x00);
  Wire.endTransmission(true);

  pinMode(fLED, OUTPUT);
  digitalWrite(fLED, LOW);
  pinMode(sLED, OUTPUT);
  digitalWrite(sLED, LOW);
  pinMode(tLED, OUTPUT);
  digitalWrite(tLED, LOW);

  long seed = 0;
  for(int i = 0; i<64; i++){
    seed ^= analogRead(A0) + millis() + micros();
```

```
    delay(50);
  }
  Serial.print("Seed: ");Serial.println(seed);
  randomSeed(seed);
}

void loop(){
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(MPU, 6, true);
  readAcc(isSaveOld); //Početna vrijednost = true

  timeCalculator();
  delay(50);
}


void readAcc(bool saveOld){
  if(saveOld){
    oldAccX = (Wire.read() << 8 | Wire.read()) / 16384.0;
    oldAccY = (Wire.read() << 8 | Wire.read()) / 16384.0;
    oldAccZ = (Wire.read() << 8 | Wire.read()) / 16384.0;
    isSaveOld = false;
  }
  else if(!saveOld){
    AccX = (Wire.read() << 8 | Wire.read()) / 16384.0;
    AccY = (Wire.read() << 8 | Wire.read()) / 16384.0;
    AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0;
    if(didAccXChange() || didAccYChange() || didAccZChange()){
      if(!isActivity){
        previousTime = millis();
        isSaveOld = true;
      }
      isActivity = true;
    }
  }
}

bool didAccXChange(){
  if(oldAccX + 1 < AccX || oldAccX - 1 > AccX){
    return true;
  }
  return false;
}
```

```
bool didAccYChange(){
  if(oldAccY + 1 < AccY || oldAccY - 1 > AccY){
    return true;
  }
  return false;
}

bool didAccZChange(){
  if(oldAccZ + 1 < AccZ || oldAccZ - 1 > AccZ){
    return true;
  }
  return false;
}

void timeCalculator(){
  currentTime = millis();
  elapsedTime = (currentTime - previousTime)/1000;
  if(elapsedTime > 4 && isActivity){
    printRandomNumber();
  }else if(isActivity){
    loadingAnimation();
  }
}

void loadingAnimation(){
  digitalWrite(fLED,HIGH);
  delay(100);
  digitalWrite(sLED,HIGH);
  delay(100);
  digitalWrite(tLED,HIGH);
  digitalWrite(fLED,LOW);
  delay(100);
  digitalWrite(sLED,LOW);
  delay(100);
  digitalWrite(tLED,LOW);
}

void printRandomNumber(){
  if(isActivity){
    randNumber = random(1,7);
    Serial.print(testCounter+1);Serial.print("/200: ");Serial.println(randNumber);
    testCounter++;
    if(testCounter == 200){
      Serial.print("1:");Serial.println(test1);
```

```
      Serial.print("2:");Serial.println(test2);
      Serial.print("3:");Serial.println(test3);
      Serial.print("4:");Serial.println(test4);
      Serial.print("5:");Serial.println(test5);
      Serial.print("6:");Serial.println(test6);
    }

    printNum(randNumber);
    isActivity=false;
  }
}

void printNum(int genNum){
  switch(genNum){
    case 1:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, LOW);
      test1++;
      break;
    case 2:
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, LOW);
      test2++;
      break;
    case 3:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, HIGH);
      digitalWrite(tLED, LOW);
      test3++;
      break;
    case 4:
      digitalWrite(fLED, LOW);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, HIGH);
      test4++;
      break;
    case 5:
      digitalWrite(fLED, HIGH);
      digitalWrite(sLED, LOW);
      digitalWrite(tLED, HIGH);
      test5++;
      break;
    case 6:
```

```
    digitalWrite(fLED, LOW);
    digitalWrite(sLED, HIGH);
    digitalWrite(tLED, HIGH);
    test6++;
    break;
  }
}
```