Schoolnite 2

Učenik: Mihael Orak

Mentor: Roman Rubčić

OŠ Iver, Sesvetski Kraljevec

Opis projekta

Schoolnite 2 je igra za dva igrača. Cilj igre je obraniti Zemlju od napada vanzemaljaca. Svaki igrač gađa svog vanzemaljca i tko prvi skupi deset pogodaka pobjeđuje.

Da bi krenuli s gađanjem i nakon svaka tri ispucana metka potrebno je napuniti pištolj. To se radi tako da točno odgovorimo na jedno ponuđeno pitanje.

Pitanja se na početku igre učitavaju na pištolj preko MKR-1000 s web servera. Na serveru imamo popis tema, koji se može lako proširivati. Na primjer teme: Zbrajanje do 10, Množenje i dijeljenje do 100, Engleski poznavanje boja, Priroda i društvo - obitelj, Glavni gradovi država svijeta,... Kad se odabere određena tema, pitanja iz te teme se učitavaju na pištolj. Svaki pištolj odnosno igrač može imati svoju temu, prilagođeno uzrastu ili pak prema gradivu koje se želi vježbati. Pištolji su napravljeni tako da prilikom gađanja ispuštaju zvuk, zavibriraju, te laserom gađaju metu.

Poveznice na filmove:

Schoolnite2
https://vimeo.com/426798212

Schoolnite2_3Dprint
https://vimeo.com/426800652

Schoolnite2_Meta
https://vimeo.com/426800763

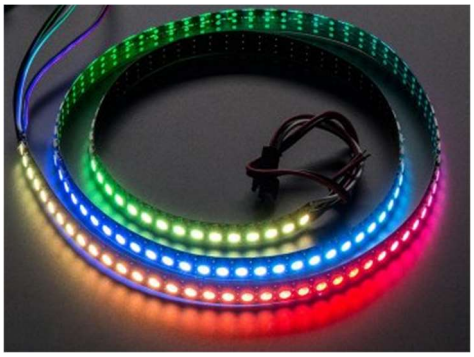
Schoolnite2_Vanzemaljci
https://vimeo.com/426800791

Potrebni dijelovi

Meta

- Arduino MKR-1000
- LCD Display
- Tipkala 3kom
- Laser receiver 2 kom

- Stepdown 12V na 5V
- adapter 12V
- utičnica za adapter 12V
- Led traka s mogućnošću adresiranja dioda (NeoPixel LED Strip)

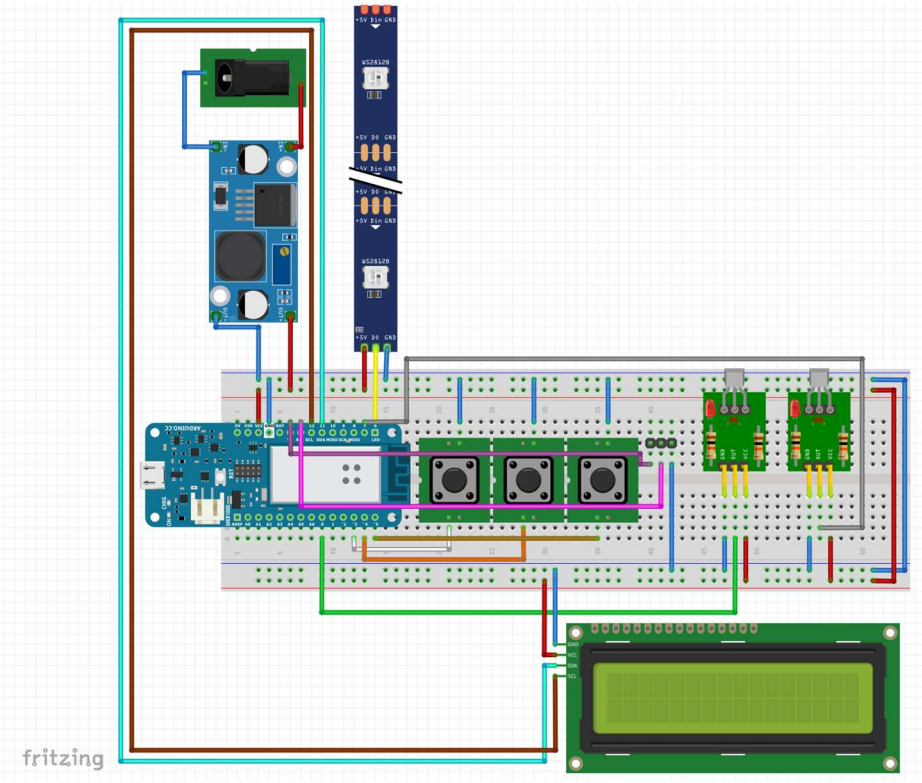


Pištolj

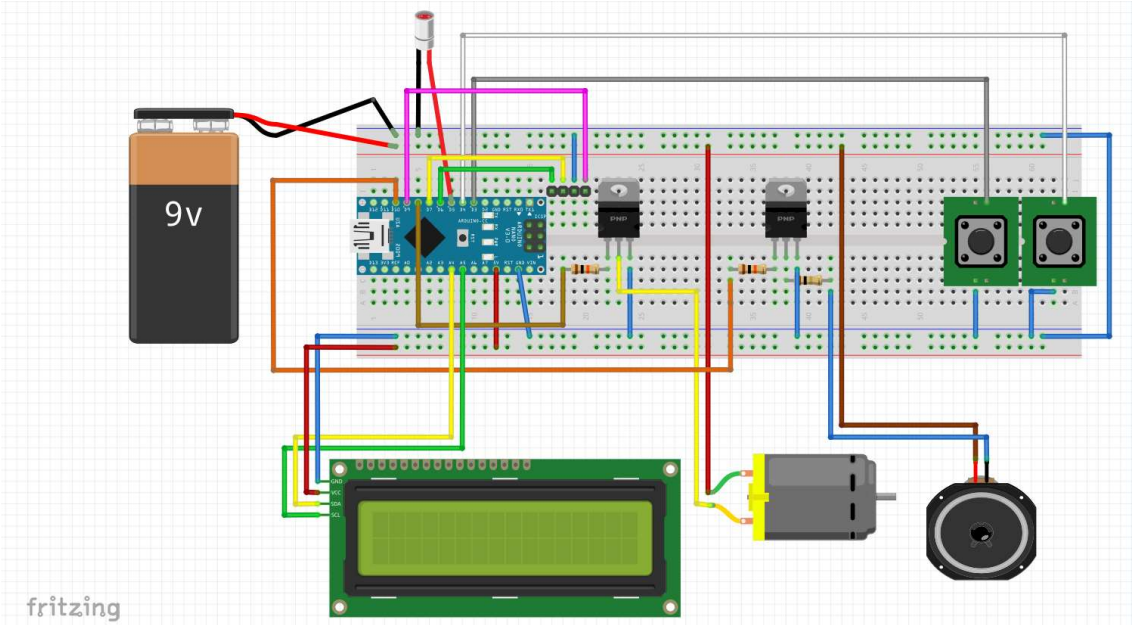
- Adruino nano
- LCD Display
- Tranzistor TIP-120 2kom
- Tipkala 2kom
- Vibracijski motor
- Zvučnik
- Otpornik 100 ohma 2 kom
- Otpornik 2 ohma
- Laser
- 9V baterija

Fritzing shema

Meta s vanzemaljcima

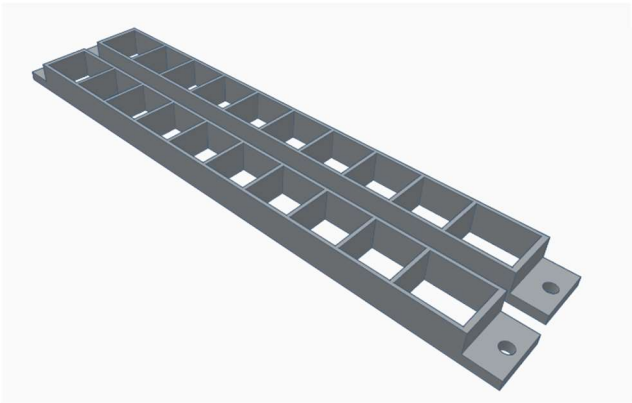


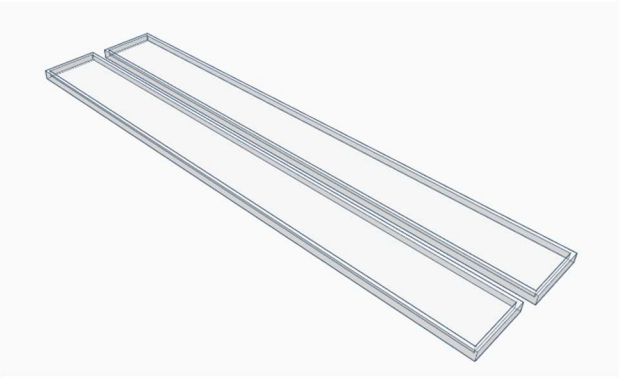
Pištolj

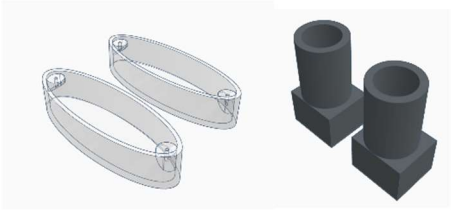


3D printani dijelovi

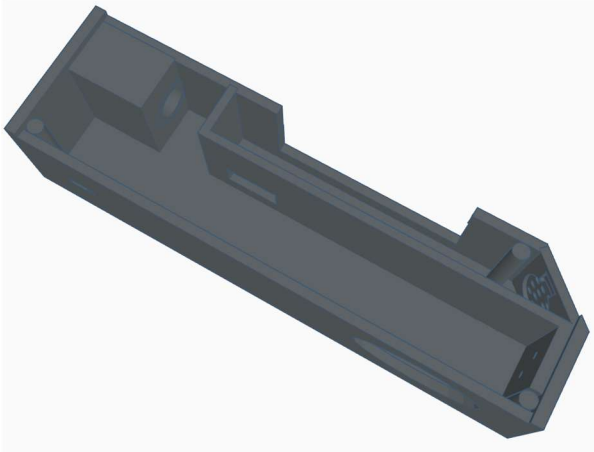
Na meti

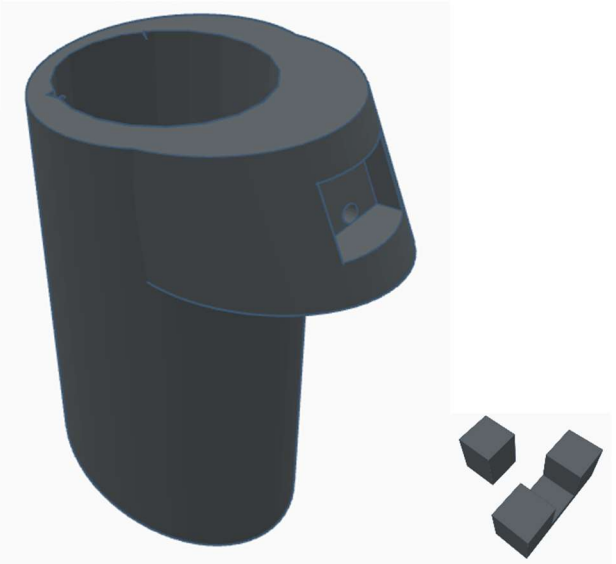


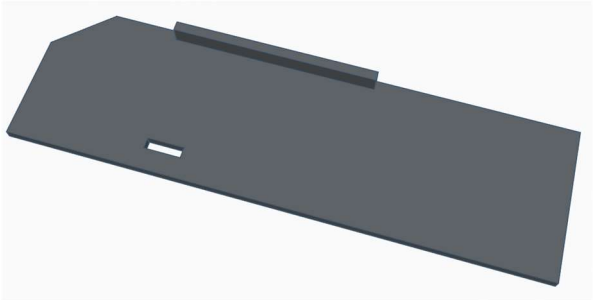




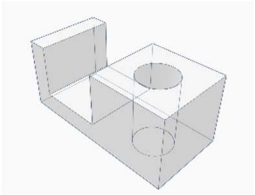
Pištolj





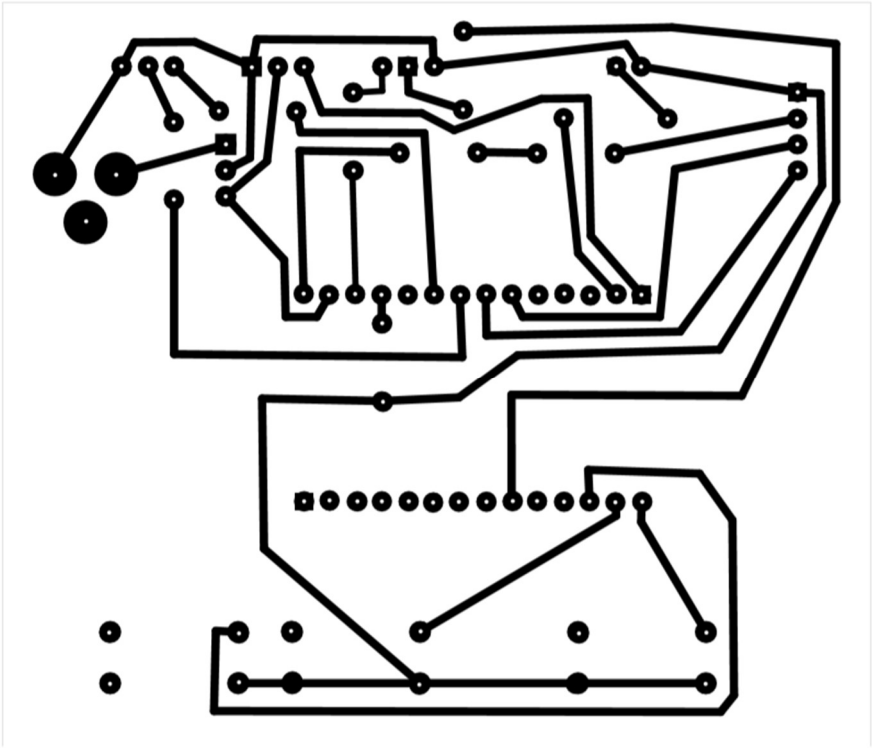


Držači žica

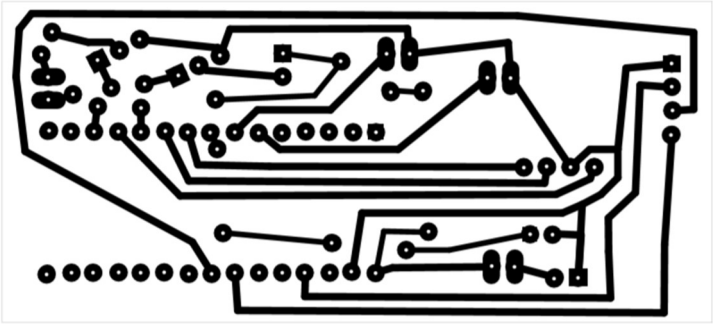


Tiskane pločice

Meta



Pištolj



PROGRAM

Meta s vanzemaljcima

```
#include <WiFi101.h>
#include "arduino_secrets.h"
#include <LiquidCrystal_I2C.h>
#include <FastLED.h>

#define MAX_BROJ_IGRACA 4
#define MAX_DULJINA_TEMA_I_ADR 65
#define MAX_DULJINA_PIT_ODG 65
#define BROJ_TEMA 20   // maksimalni broj objasnjenja
#define BROJ_PITANJA 10   // maksimalni broj pitanja
#define BUTTON_LEFT 3
#define BUTTON_RIGHT 4
#define BUTTON_SELECT 5
#define LED_STRIP_PIN 6

#define LASER_CITAC_PIN_1 0
#define LASER_CITAC_PIN_2 7
#define BODOVI_POTREBNI_ZA_POBJEDU 10
#define LED_STRIP_BROJ 24

WiFiClient client;
LiquidCrystal_I2C lcd(0x27,16,2);
CRGB led_strip[LED_STRIP_BROJ];


const char ssid[] = "TPLOM_2";        //ime rutera
const char pass[] = "DE241ZFYWA251K127";    // šifra rutera
char pis_kod_start[] = "*#pis1";
char pis_kod_end[] =   "*#pis1*";

char popis_tema[BROJ_TEMA][MAX_DULJINA_TEMA_I_ADR];
char naziv_teme[33];  // x objasnjenje s četiri odgovora maksimalno 30 znak s upitnikom na početku
char adresa_teme[7] ;  // rjesenje za x objasnjenja
char pitanja[BROJ_PITANJA][MAX_DULJINA_PIT_ODG];

char adresa_teme_igrac[MAX_BROJ_IGRACA][7] ;
int pis_bodovi[MAX_BROJ_IGRACA];
char tmp_1[5];

int x;
int broj_tema = 0;
int broj_pitanja = 0;
int odabrana_tema = 0;
bool odabrani_mod = true; // true -> mod ucitavanja pitanja, false -> mod igre
bool vanzemaljac_puca = true;

int bodovi_igrac_1 = 0;
int bodovi_igrac_2 = 0;
int citac1;
int citac2;
int pobjednik = 0;
int trajanje_eksplozije = 3000; // trajanje je u milisekundama
int metak_brzina_vanzemaljci = 30; // brzina mijenjanja je u milisekundama

int vanzemaljac_metak_polozaj;

int status = WL_IDLE_STATUS;

char server[] = "mihior.atwebpages.com";    //adresa moje stranice

unsigned long pocetak_pozadina;
unsigned long pocetak_explozija;

int rgb_eksplozija[3]={255, 0, 0};
int rgb_pucanje[3]={255, 165, 0};
int rgb_igrac_1[3]={0, 0, 255};
int rgb_igrac_2[3]={0, 255, 0};

void Pozadina()
{

  if (vanzemaljac_puca and millis() - pocetak_pozadina >= metak_brzina_vanzemaljci)
  {
```

```
    led_strip[vanzemaljac_metak_polozaj + 2] = CRGB(0, 0, 0);
    led_strip[vanzemaljac_metak_polozaj + 12 + 2] = CRGB(0, 0, 0);

    vanzemaljac_metak_polozaj -= 1;

    if (vanzemaljac_metak_polozaj >= 0)
    {
      led_strip[vanzemaljac_metak_polozaj] = CRGB(rgb_pucanje[0], rgb_pucanje[1], rgb_pucanje[2]);
      led_strip[vanzemaljac_metak_polozaj + 12] = CRGB(rgb_pucanje[0], rgb_pucanje[1], rgb_pucanje[2]);
    }
    FastLED.show();
    pocetak_pozadina = millis();
    if (vanzemaljac_metak_polozaj <= -3)
    {
      vanzemaljac_metak_polozaj = 10;
      vanzemaljac_puca = false;
      pocetak_pozadina += 1000;
    }

  }//if (pocetak_pozadina - millis() >= metak_brzina_vanzemaljci)

  if (!vanzemaljac_puca and millis() >= pocetak_pozadina)
  {
    vanzemaljac_puca = true;
  }

}
void Eksplozija(int igrac)
{
  int blicanje = 500;
  bool eksplozija_ukljucena = true;

  lcd.setCursor(0,0);
  lcd.print("Pobijednik je:");
  lcd.setCursor(0,1);
  lcd.print("Igrac: " + pobjednik);

  pocetak_explozija = millis();

  while (millis() - pocetak_explozija < trajanje_eksplozije) //vanzemaljac eksplodira
  {
    if (!eksplozija_ukljucena)
    {
      led_strip[12 * pobjednik - 2] = CRGB(0, 0, 0);
      led_strip[12 * pobjednik - 1] = CRGB(0, 0, 0);
    }
    else
    {
      led_strip[12 * pobjednik - 2] = CRGB(rgb_eksplozija[0], rgb_eksplozija[1], rgb_eksplozija[2]);
      led_strip[12 * pobjednik - 1] = CRGB(rgb_eksplozija[0], rgb_eksplozija[1], rgb_eksplozija[2]);
    }
    delay(blicanje);
    FastLED.show();
    if (blicanje > 50)
    {
      blicanje /= 1.2;
    }

    eksplozija_ukljucena = !eksplozija_ukljucena;
  } // while (pocetak - millis() < trajanje_eksplozije)

} //void Eksplozija(int igrac)

void PokreniIgru()
{
  boolean omoguci1 = true;
  boolean omoguci2 = true;

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Igraj");
  while (pobjednik == 0)
  {
    citac1 = digitalRead(LASER_CITAC_PIN_1);
    citac2 = digitalRead(LASER_CITAC_PIN_2);
    if (!omoguci1 and citac1 == LOW) {
      omoguci1 = true;
    }
    if (!omoguci2 and citac2 == LOW) {
      omoguci2 = true;
    }
```

```
    if (omoguci1 and citac1 == HIGH)
    {
      omoguci1 = false;
      bodovi_igrac_1 += 1;
      led_strip[bodovi_igrac_1 - 1] = CRGB(rgb_igrac_1[0], rgb_igrac_1[1], rgb_igrac_1[2]);
      FastLED.show();
      Serial.println("1");
      if (bodovi_igrac_1 >= 10){
        pobjednik = 1;
      }
    }
    if (omoguci2 and citac2 == HIGH)
    {
      omoguci2 = false;
      bodovi_igrac_2 += 1;
      led_strip[bodovi_igrac_2 - 1 + 12] = CRGB(rgb_igrac_2[0], rgb_igrac_2[1], rgb_igrac_2[2]);
      FastLED.show();
      Serial.println("2");
      if (bodovi_igrac_2 >= 10){
        pobjednik = 2;
      }
    }

  }//while (pobjednik == 0)
  Eksplozija(pobjednik);
} // void PokreniIgru()

void UcitajPitanja()
{
  boolean promjena = true;

  while (!pritisnut(BUTTON_SELECT)){         // ponavljaj dok ne pritisne tipku SELECT
    if (promjena) {
      Serial.println(naziv_teme);
      Serial.println(popis_tema[odabrana_tema]);
      sscanf(popis_tema[odabrana_tema], "%[^;];%[^;];", adresa_teme, naziv_teme);  // temp_string
razdvoji po znaku ";" i spremi u adresa_teme i naziv_teme
      Serial.println(naziv_teme);

      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print(naziv_teme);
      if (strlen(naziv_teme) > 16) {
        lcd.setCursor(0,1);
        lcd.print(naziv_teme + 16);  // ispiši u novi red drugi dio naziva (od 16 znaka na dalje)
      }
      promjena = false;
    }
    if (pritisnut(BUTTON_LEFT)){
      odabrana_tema--;
      promjena = true;
    } else if (pritisnut(BUTTON_RIGHT)){
      odabrana_tema++;
      promjena = true;
    }
    if(odabrana_tema < 0){
      odabrana_tema = broj_tema - 1;
    }
    else if (odabrana_tema > broj_tema - 1){
      odabrana_tema = 0;
    }
  }//while(true)

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Citanje s serve.");

  strcpy(adresa_teme_igrac[0], adresa_teme);

  broj_pitanja = ProcitajStranicuSaWebServera(adresa_teme_igrac[0], BROJ_PITANJA);
  Serial.println(broj_pitanja);
  Serial.println(pitanja[0]);

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Slanje podataka...");

  Serial.println(pis_kod_start);
  Serial1.write(pis_kod_start, MAX_DULJINA_TEMA_I_ADR);
  delay(400);
```

```
  for(x = 0; x < broj_pitanja; x++){
    Serial.println(pitanja[x]);
    Serial1.write(pitanja[x], MAX_DULJINA_TEMA_I_ADR);
    delay(400);
  }
  Serial.println(pis_kod_end);
  Serial1.write(pis_kod_end, MAX_DULJINA_TEMA_I_ADR);
  delay(400);
}//void UcitajPitanja()


int ProcitajStranicuSaWebServera(String WebStranica, int MaxBrRedaka)
{
  String temp_string="";
  char temp_string2[32*5+10]="";
//  char tmp_rjesenje[5]="";
  char znak;
  int trenutni_red = 0;
  char pamti = false;
  int znak_br = 0;
  bool spojenNaServer = false;

  while (!spojenNaServer)
  {
    Serial.println("\nStarting connection to server...");
    //lcd.setCursor(1,0);
    lcd.print("Spajanje...");
    printWiFiStatus();

    if (client.connect(server, 80))
    {
      Serial.println("connected to server");
      Serial.println(WebStranica);
      //client.println("GET /P00000.html HTTP/1.1");
      //printf(temp_string, "GET /%s.html HTTP/1.1", WebStranica);
      temp_string = "GET /" + WebStranica + ".txt HTTP/1.1";
      client.println(temp_string);
      client.println("Host: mihior.atwebpages.com");
      client.println("Connection: close");
      client.println();
      spojenNaServer = true;
      delay(1000);
    }
    else
    {
      Serial.println("Spajanje na web server nije uspjelo");
      client.stop();
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Greska!");
      delay(1000);
      //while(true);
    }
  }//while (!spojenNaServer)

  while (!client.available())  //čekaj dok klijent ne posatne dostupan
  {
    Serial.println("Klijent nije dostupan!");
  }

/*
  for (int i=0; i<32*5+10; i++) { // pobriši varijablu da ne ostanu stati podaci
    temp_string2[i] = ' ';
  }  */
  while (client.available())
  {
    znak = client.read();
//    Serial.print(znak);
    if (znak == '$')
    {
      pamti = true;
      znak = client.read();   // procitaj slijedeci znak
    }
    if (znak == '#')
    {
      pamti = false;
      break;    // napusti petlju while
    }
    if (pamti)
    {
```

```
      if (znak == '\n'){
        temp_string2[znak_br] = '\0';
        if (WebStranica == String("P00000")) {
          //sscanf(temp_string2, "%[^;];%[^;];", adresa_teme[trenutni_red], naziv_teme[trenutni_red]);
// temp_string razdvoji po znaku ";" i spremi u adresa_teme i naziv_teme
          strcpy(popis_tema[trenutni_red], temp_string2);
        }
        else
        {
          //sscanf(temp_string2, "%[^;];%[^;];%[^;];%[^;];%[^;];%[^;];", pitanje[trenutni_red][0],
pitanje[trenutni_red][1], pitanje[trenutni_red][2], pitanje[trenutni_red][3], pitanje[trenutni_red][4],
tmp_rjesenje);  // temp_string razdvoji po znaku ";" i spremi u pitanja i rjesenja
          //rjesenje[trenutni_red] = atoi(tmp_rjesenje);    // pretvori iz niza znakova u int
          strcpy(pitanja[trenutni_red], temp_string2);
        }
        znak_br=0;
        trenutni_red++;
        /*
        for (int i=0; i<32*5+10; i++) { // pobriši varijablu da ne ostanu stati podaci
          temp_string2[i] = ' ';
        }  */
        if (trenutni_red >= MaxBrRedaka){  // ako više nema mjesta za pamtiti teme
          pamti = false;   // prestani pamtiti
        }
      }
      else
      {
        temp_string2[znak_br] = znak;
        znak_br++;
      }
    } // if (pamti)
  } // while (client.available())
  client.stop();
  Serial.println("Odspajanje sa servera");
  return(trenutni_red); // vrati rezultat funkcije broj procitanih redova
}//ProcitajStranicuSaWebServera(String WebStranica, int MaxBrRedaka)

void printWiFiStatus() {
  // print the SSID of the network you're attached to:
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:
  IPAddress ip = WiFi.localIP();
  Serial.print("IP Address: ");
  Serial.println(ip);

  // print the received signal strength:
  long rssi = WiFi.RSSI();
  Serial.print("signal strength (RSSI):");
  Serial.print(rssi);
  Serial.println(" dBm");
}

// provjeri da li je pritisnut neki gumb i čekaj dok se ne otpusti
bool pritisnut(int gumb){
  if(digitalRead(gumb) == LOW){
    delay(200);
    while (digitalRead(gumb) == LOW){
    }
    return(true);
  }
  return(false);
}
```

Pištolj

```
#include <SPI.h>
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>

#define MAX_DULJINA_PIT_ODG 65
#define MAX_BROJ_PITANJA 10
#define POTREBNO_ODGOVORA 1
#define DOBIVENI_METCI 3

#define BUTTON_SHOOT 3
#define BUTTON_PICK 4
#define LASER 5
#define SOFT_SERIAL_RX 6
#define SOFT_SERIAL_TX 7
#define MOTOR 8
#define PIN_SPOJEN 9
#define BUZZER 10

#define LASER_KOD 80
#define UKUPNO_TRAJANJE_LASERA 100
#define ODSTUPANJE 1


byte metakZnak[] = {
  B11111,
  B11111,
  B11111,
  B11111,
  B11111,
  B11111,
  B11111,
  B11111
};

LiquidCrystal_I2C lcd(0x27,20, 4);
SoftwareSerial SoftSerial(SOFT_SERIAL_RX, SOFT_SERIAL_TX); // SoftwareSerial(RxPin, TxPin)

int ukupnoPitanja = 0;
int metci = 0;
int prosli_metci = 0;
int ucitaoPitanja = 0;
int pitanjeOdgovara = 0;
int x, z;
int button_pick;
int button_shoot;

char pitanja[MAX_BROJ_PITANJA][MAX_DULJINA_PIT_ODG];
char pitanje[5][33];
int rjesenje;

  char primljeno[MAX_DULJINA_PIT_ODG];
  bool pamti = false;

void setup()
{
  lcd.init();
  lcd.backlight();
  pinMode(BUTTON_SHOOT, INPUT_PULLUP);
  pinMode(BUTTON_PICK, INPUT_PULLUP);
  pinMode(PIN_SPOJEN, INPUT_PULLUP);
  pinMode(LASER, OUTPUT);
  pinMode(MOTOR, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  lcd.createChar(0, metakZnak);

  Serial.begin(9600);
  SoftSerial.begin(9600);

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Ucitajte pitanja!");

}

void loop()
{
  //Serial.println(digitalRead(PIN_SPOJEN));
```

```
  if(digitalRead(PIN_SPOJEN) == LOW){
    delay(1000);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Spojen");
    lcd.setCursor(0,1);
    lcd.print("Ucitavanje");

    metci = 0;
    pitanjeOdgovara = 0;

    ucitajPitanja();
    lcd.clear();
    while(digitalRead(PIN_SPOJEN) == LOW){
      lcd.setCursor(0,0);
      lcd.print("odspojite");
      lcd.setCursor(0,1);
      lcd.print("zicu");
    delay(1000);
    }
    ucitaoPitanja = 1;
    //u podprogramu gore ispiše na kraju da je učitao
    }

  if(ucitaoPitanja == 1){
    button_pick = digitalRead(BUTTON_PICK);
    button_shoot = digitalRead(BUTTON_SHOOT);
    if(button_shoot == LOW and button_pick == LOW){
          lcd.clear();
          while(button_shoot == LOW or button_pick == LOW){
            button_pick = digitalRead(BUTTON_PICK);
            button_shoot = digitalRead(BUTTON_SHOOT);
            lcd.setCursor(0,0);
            lcd.print("Odpusti");
            lcd.setCursor(0,1);
            lcd.print("Tipke");
          }
          metci = 0;
          pitanjeOdgovara = 0;
    }
    if (metci == 0){
      odgovoriNaPitanja();
      lcd.clear();
    }
    else{
      ispisiMetke();
      lcd.setCursor(0,1);
      lcd.print(metci);

      if(digitalRead(BUTTON_SHOOT) == LOW){
        metci--;
        ispisiMetke();
        for(z = 0; z < ODSTUPANJE; z++){
          digitalWrite(LASER, HIGH);
          delay(LASER_KOD);
          digitalWrite(LASER, LOW);
          delay(UKUPNO_TRAJANJE_LASERA - LASER_KOD);
        }
        //tone(BUZZER, 250, 250);
        digitalWrite(MOTOR, HIGH);
        noise(500, 150);
        //digitalWrite(MOTOR, HIGH);
        //delay(100);
        digitalWrite(MOTOR, LOW);
        delay(250);
        //možda se cuje pucanj pistolja
      }
      while(digitalRead(BUTTON_SHOOT) == LOW){
      }
    }
  }
}


void ucitajPitanja(){
  char primljeno[MAX_DULJINA_PIT_ODG];
  bool pamti = false;

    Serial.println("radim");
  ukupnoPitanja = 0;
  while(true){
```

```
    primljeno[0] = '\0';
    if (SoftSerial.available()) // when data is available in serial buffer
      {
      SoftSerial.readBytes(primljeno, MAX_DULJINA_PIT_ODG); //Read the serial data and store in var
      Serial.println(primljeno);
      if (strcmp(primljeno, "*#pis1") == 0){
        pamti = true;
        } else if (strcmp(primljeno, "*#pis1*") == 0){
          pamti = false;
          ukupnoPitanja--;  // popravi broj pitanja, jer je previše za 1
          break;
        } else if (pamti){
          strcpy(pitanja[ukupnoPitanja], primljeno);
          ukupnoPitanja++;
        }
      }
  }


  pitanjeOdgovara = 0;  // nakon učitavanja kreni od prvog pitanja
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("!Ucitano!");
}

void odgovoriNaPitanja(){
  char tmp_rjesenje[5]="";
  int pitanjeIliOdgovor = 0;
  int odgovoriTocni = 0;

  while(odgovoriTocni < POTREBNO_ODGOVORA){
    sscanf(pitanja[pitanjeOdgovara], "%[^;];%[^;];%[^;];%[^;];%[^;];%[^;];", pitanje[0], pitanje[1],
pitanje[2], pitanje[3], pitanje[4], tmp_rjesenje);  // temp_string razdvoji po znaku ";" i spremi u
pitanja i rjesenje
    rjesenje = atoi(tmp_rjesenje);
    lcd.clear();
    while (! (pritisnut(BUTTON_SHOOT) and pitanjeIliOdgovor!=0) ){         // ponavljaj dok ne pritisne
tipku BUTTON_SHOOT, ali ne na pitanju, nego na odgovorima
      lcd.setCursor(0,0);
      lcd.print(pitanje[pitanjeIliOdgovor]);
      if (strlen(pitanje[pitanjeIliOdgovor]) > 16) {
        lcd.setCursor(0,1);
        lcd.print(pitanje[pitanjeIliOdgovor] + 16);  // ispiši u novi red drugi dio naziva (od 16 znaka
na dalje)
      }

      if(pritisnut(BUTTON_PICK)){
        pitanjeIliOdgovor++;
        lcd.clear();
        if (pitanjeIliOdgovor > 4){
          pitanjeIliOdgovor = 0;
        }
      }
    }
    if (pitanjeIliOdgovor == rjesenje){
      odgovoriTocni++;
    } else {
      //TODO: ispisi netocno !!!!!!!!!!!!!!!!
    }
    pitanjeOdgovara++;
    pitanjeIliOdgovor = 0;
    if (pitanjeOdgovara > ukupnoPitanja){
      pitanjeOdgovara = 0;
    }
  }
  metci = DOBIVENI_METCI;
}

// provjeri da li je pritisnut neki gumb i čekaj dok se ne otpusti
bool pritisnut(int gumb){
  if(digitalRead(gumb) == LOW){
    delay(200);
    while (digitalRead(gumb) == LOW){
    }
    return(true);
  }
  return(false);
}

void ispisiMetke(){
  if (metci != prosli_metci){
```

```
    prosli_metci = metci;
    lcd.clear();
    lcd.setCursor(0,0);
    for(x = 0; x < metci; x++){
      lcd.write(0);
      lcd.print(" ");
    }
  }
}

void noise(int freq, int duration)  {
  int low = freq - 500;
  int high = freq + 500;
  unsigned long time = millis();
  while(millis() - time <= duration)  {
    tone(BUZZER, random(low, high));
  }
  noTone(BUZZER);
}
```