Mini-Turetă
Laser

Vrăbete Alexandru Nicolae

Cuprins

Mini-Turetă Laser

Acest proiect vizează dezvoltarea unui dispozitiv care să controleze direcția de propagare a unui fascicul laser. Acesta va permite ajustarea precisă a poziției fasciculului în toate direcțiile (sus-jos, stânga-dreapta). Pentru a realiza acest lucru, vom utiliza două servomotoare montate perpendicular, care vor asigura mișcarea laserului.

Principalele obiective ale proiectului sunt următoarele:

1. **Controlul turetei**

   Implementarea unui sistem de control manual pentru fiecare servomotor, utilizând un potențiometru (joystick). Acest lucru ne va permite să ajustam poziția fasciculului laser.

2. **Monitorizarea traiectoriei**

   Microcontroler-ul va fi responsabil de monitorizarea traiectoriei fasciculului. Coordonatele x și y ale fasciculului vor fi afișate pe un ecran LCD, oferind o vizualizare în timp real a poziției acestuia.

   Cu ajutorul potențiometrului (joystick) transmitem o valoare către servomotoare, acestea având valori cuprinse între 0-90 grade

3. **Ținta fasciculului laser**

   Pentru realizarea țintei laser, am folosit un singur punct de detecție, acesta fiind realizat dintr-un fotorezistor. Atunci când fotorezistorul depășește o anumită valoare prestabilită de noi, algoritmul va aprinde dioda led de culoare verde care va admite faptul că ținta a fost atinsă cu succes.

Resurse Hardware

În cadrul proiectului, am utilizat următoarele resurse hardware:

1. **Servomotoare**: Am integrat două servomotoare pentru a asigura mișcarea fasciculului în direcțiile dorite (sus-jos, stânga-dreapta).

2. **Modul diodă-laser:** Acesta un dispozitiv care emite un fascicul de lumină laser prin intermediul unei diode laser.

3. **Modul Joystick**: Acest modul permite controlul manual al fiecărui servomotor. Utilizatorul poate ajusta poziția fasciculului utilizând joystick-ul.

4. **LCD I2C**: Am adăugat un ecran LCD cu interfață I2C pentru a afișa coordonatele x și y ale fasciculului în timp real. Acesta oferă o vizualizare convenabilă a traiectoriei.

5. **ESP32**: Pentru a detecta fasciculul laser, am folosit un microcontroler ESP32, la acesta am conectat un fotorezistor și o diodă led. Ținta este independentă față de microcontroller-ul pe care l-am folosit pentru dezvoltarea turetei-laser.

Resurse software

Am realizat două coduri folosind mediul de dezvoltare Arduino: unul pentru turetă laser, iar celălalt pentru țintă.

Librării folosite:

Servo.h: Aceasta librărie am folosit-o pentru a controla cele două servomotoare.

LiquidCrystal_I2C.h: Aceasta librărie am folosit-o pentru controlul display-ului LCD și afișarea unghiurilor servomotoarelor. Mediul de dezvoltare se bazează limbajul de programare C.

Funcționalități:

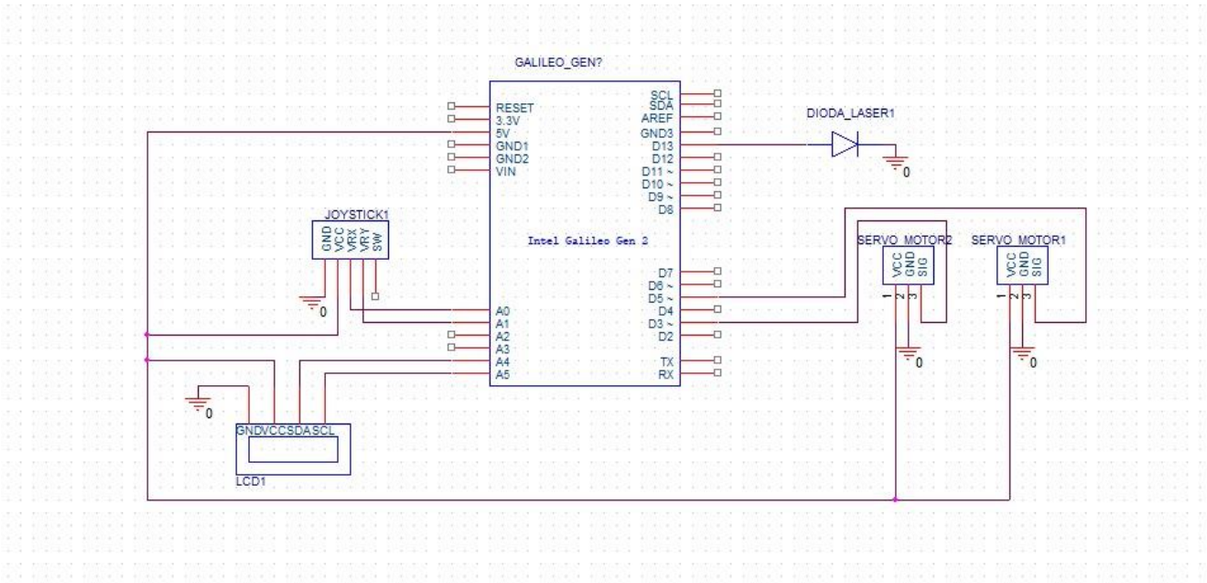
Codul ne ajută să controlăm cele două servomotoare (myservo1 și myservo2) folosind semnale PWM (Pulse Width Modulation) pe pinii 3 și 5.

Valorile pentru controlul servomotoarelor sunt citite de la potențiometrul conectat la pinii analogici A0 pentru coordonata x și A1 pentru coordonata y .

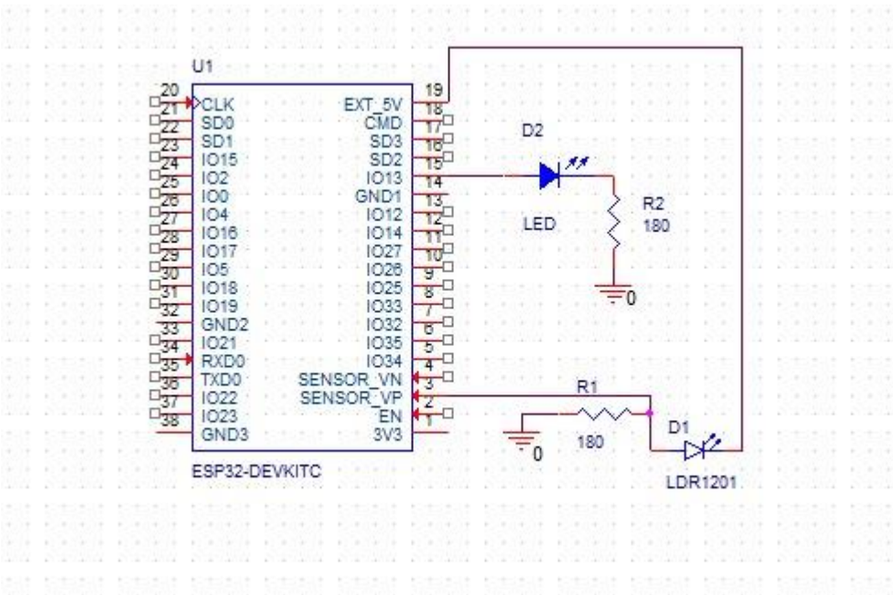
Valorile citite de la potențiometru sunt apoi mapate la unghiurile minime și maxime ale servomotoarelor și sunt folosite pentru a controla poziția acestora. Unghiurile actuale ale servomotoarelor sunt afișate display-ul LCD.

Implementare hardware

Schema electrică a microcontroller-ului Intel Galileo Gen 2 și a turetei



Schema electrică a țintei



Principiul de funcționare al turetei

Placa Intel Galileo Gen 2 este utilizată ca unitatea centrală de control a întregului sistem. Aceasta primește intrări de la potențiometru (joystick) și controlează servomotoarele cât și dioda laser. Placa este alimentată prin pinul VIN, iar cele două pinuri de masă (GND1 și GND2) sunt conectate la masa comună a circuitului.

Potențiometrul (joystick-ul) este conectat la intrările analogice A0 și A1 ale plăcii Galileo. Acestea permit citirea poziției potențiometrului pe axele X și Y.

Datele de la potențiometru sunt utilizate pentru a controla poziția servomotoarelor, prin intermediul semnalelor PWM (Pulse Width Modulation) generate de placa Galileo.

Servomotoarele sunt conectate la pinii digitali D2 și D3 al plăcii Galileo. Acești pini sunt aleși pentru a controla servomotoarele deoarece aceștia pot genera semnale PWM (Pulse Width Modulation) necesare pentru controlul poziției servomotoarelor.

Servomotoarele primesc semnale PWM (Pulse Width Modulation) de la placa Galileo, care controlează poziția lor, aceasta fiind determinată de poziția potențiometrului (joystick-ului).

Dioda laser este conectată la pinul digital D13 al plăcii Galileo. Aceasta este controlată digital, putând fi pornită și oprită prin setarea pinului D13 în stare HIGH (pornit) sau LOW (oprit).Pinul D13 este un pin special care, pe multe plăci compatibile cu Arduino, include un LED intern conectat pentru debugging. În cazul nostru, acest pin este folosit pentru a controla dioda laser.

Modulul LCD este conectat la plăcuță folosind un protocol de comunicație I2C (SDA și SCL). Pinii SDA "serial data pin" și SCL "serial data clock" sunt folosiți pentru comunicația serială cu LCD-ul, permițând afișarea de informații, cum ar fi poziția potențiometrului sau starea servomotoarelor și a diodei laser.

Principiul de funcționare al țintei

Principiul de funcționare al acestui circuit se bazează pe utilizarea unui microcontroler ESP32 pentru a controla un LED (D2) în funcție de iluminarea detectată de un senzor fotoresistor (LDR - Light Dependent Resistor, D1).

LDR-ul (Light Dependent Resistor) este un senzor care își schimbă rezistența în funcție de cantitatea de lumină care cade pe el. Când iluminarea este puternică, rezistența LDR-ului scade, iar când iluminarea este scăzută, rezistența crește.
LDR-ul este conectat într-un divizor de tensiune împreună cu rezistența R1 (180 ohmi). Aceasta permite obținerea unei tensiuni variabile la punctul de conexiune dintre R1 și LDR, în funcție de nivelul de iluminare.

Microcontrolerul ESP32 preia tensiunea variabilă de la divizorul de tensiune (R1 și LDR) prin pinul SENSOR_VP. Aceasta tensiune este proporțională cu nivelul de lumină detectat de LDR.
În funcție de valoarea tensiunii citite, ESP32 va decide dacă să aprindă sau să stingă LED-ul (D2).

LED-ul este conectat în serie cu rezistența R2 (180 ohmi) pentru a limita curentul care trece prin el.
ESP32 controlează LED-ul prin intermediul unui pin de ieșire digital. Atunci când ESP32 activează acest pin (setându-l la un nivel logic "HIGH"), curentul va circula prin circuit și LED-ul va lumina.
Când ESP32 dezactivează pinul (setându-l la un nivel logic "LOW"), curentul nu va mai circula și LED-ul se va stinge.

Codul implementat pe Intel Galileo Gen 2

```
#include <Servo.h>

#include <LiquidCrystal_I2C.h>

const uint8_t I2C_ADDRESS = 0x27;

const uint8_t LCD_CHAR = 16;

const uint8_t LCD_LINE = 2;

char *TITLE_ANGLE1 = "X: ";

char *TITLE_ANGLE2 = "Y: ";

LiquidCrystal_I2C lcd(I2C_ADDRESS, LCD_CHAR, LCD_LINE);

Servo myservo1;       // servo 1

Servo myservo2;      // servo 2

int pot1pin = A0;     // X

int pot2pin = A1;   // Y

unsigned int servo1Pin = 3; // servo 1 pin

unsigned int servo2Pin = 5; // servo 2 pin

unsigned int servo1AngleMin = 0; // servo 1 min angle

unsigned int servo1AngleMax = 90; // servo 1 max angle

unsigned int servo2AngleMin = 0; // servo 2 min angle

unsigned int servo2AngleMax = 90; // servo 2 max angle

unsigned int servo1Val = 0; // variable to read potentiometer 1

unsigned int servo2Val = 0; // variable to read potentiometer 2

unsigned int angle1 = 0; // current angle for servo 1
 unsigned int angle2 = 0; // current angle for servo 2
```

Am început prin includerea librăriei **Servo.h** și **LiquidCrystal_I2C.h** pentru controlarea servomotoarelor și a ecranului LCD. În continuare, constantele I2C_ADDRESS, LCD_CHAR și LCD_LINE definesc adresa I2C și dimensiunile afișajului LCD, urmând să definim și pointerii de caractere pentru afișarea titlurilor.

Obiectul „lcd” este o instanță a clasei **LiquidCrystal_I2C** care facilitează comunicarea cu afișajul LCD I2C. Prin intermediul acestei instanțe, controlam afișarea textului pe ecranul LCD, setând poziția cursorului, afișând mesaje și titluri pentru unghiurile curente ale servomotoarelor.

Cele două instanțe myservo1 și myservo2 ale clasei Servo sunt definite pentru controlul servomotoarelor, urmând să definim pinii analogici si digitali necesari potențiometrului și servomotoarelor.

Am realizat maparea servomotoarelor pentru un interval de mișcare de la 0 până la 90 de grade. Algoritmul nostru presupune că valoarea inițială a unghiului este definită ca zero grade. După ce codul este încărcat pe placă și sistemul este pornit, servomotoarele vor fi automat calibrate să se poziționeze la 45 de grade fiecare, pe baza valorii citite de la potențiometru.

```
void setup() {
  pinMode(13, OUTPUT);


  myservo1.attach(servo1Pin);  // attach servo 1 to its pin
  myservo2.attach(servo2Pin);  // attach servo 2 to its pin


  lcd.init();
  lcd.clear();
  lcd.backlight();
  lcd.print("Valori X si Y:");
  lcd.setCursor(0, 1);
  lcd.print("X: ");
  lcd.setCursor(8, 1);
  lcd.print("Y: ");
  delay(2000);
}
```

Începem prin setarea pinului digital 13 ca ieșire (OUTPUT), deoarece acesta ne furnizeaza curent și va fi utilizat pentru controlul diodei laser . Următoarele două linii de cod atașează servomotoarele la pinii specifici (servo1Pin și servo2Pin). Atașarea servomotoarelor la pini permite controlul lor prin modificarea unghiului.

Funcția de lcd.init inițializează afișajul LCD, in timp ce lcd.clear curăță ecranul LCD, eliminând orice text afișat anterior iar lcd.backlight activează iluminarea fundalului (backlight) a afișajului LCD.

Functia lcd.print afișează textul “Valori X si Y:” pe prima linie a afișajului în timp ce lcd.setCursor(0, 1) mută cursorul pe a doua linie a afișajului, iar lcd.print("X: ") și lcd.print("Y: ") afișează textul "X: " și "Y: " pe a doua linie, la pozițiile corespunzătoare.

Delay(2000) așteaptă 2000 de milisecunde (2 secunde) înainte de a continua cu următoarea instrucțiune.

```
void loop() {
  digitalWrite(13, HIGH);
  servo1Val = analogRead(pot1pin); // read potentiometer 1 value
  servo2Val = analogRead(pot2pin); // read potentiometer 2 value
  sendServos(servo1Val, servo2Val); // send servo values
}
```

Linia digitalWrite(13, HIGH) setează pinul digital 13 la nivel logic “HIGH” pentru activarea diodei laser.

Următoarele două linii citesc valorile analogice de la potențiometru, acestea vor fi în intervalul 0-1023, reprezentând tensiunea aplicată la potențiometru. Ulterior valorile vor fi citite prin funcția sendServos(servo1Val,

servo2Val).

Când valoarea inițială a unghiului este diferită de noua valoare preluată de la potențiometru, valoarea preluată devine noua valoare inițială care va fi transmisă către afișajul LCD. În același timp, atunci când apare o nouă valoare, valoarea anterioară va fi ștearsă de pe ecranul LCD și va fi afișată valoarea nouă. Această buclă funcționează atât timp cât unghiurile sunt diferite, iar coordonatele respective sunt afișate în grade.

```
void sendServos(int value1, int value2) {
  unsigned int newAngle1 = map(value1, 0, 1023, servo1AngleMin, servo1AngleMax);
  unsigned int newAngle2 = map(value2, 0, 1023, servo2AngleMin, servo2AngleMax);

  if (angle1 != newAngle1) {
    myservo1.write(newAngle1);
    lcdDisplay(newAngle1, angle2); // display servo 1 angle
    angle1 = newAngle1;
  }

  if (angle2 != newAngle2) {
    myservo2.write(newAngle2);
    lcdDisplay(angle1, newAngle2); // display servo 2 angle
    angle2 = newAngle2;
  }
}
```

Începem prin definirea funcției sendServos aceasta avand responsabilitatea de a actualiza pozițiile servomotoarelor în funcție de valorile citite de la potențiometru.

Următoarele două linii folosesc funcția map pentru a converti valorile analogice citite de la potențiometru în unghiuri pentru servomotoare, în intervalul specificat de unghiuri minime și maxime.

Instrucțiunile if verifică dacă unghiurile noi calculate (newAngle1 și newAngle2) sunt diferite de unghiurile curente (angle1 și angle2). Dacă există o diferență, poziția corespunzătoare a servomotoarelor este actualizată folosind myservo1.write(newAngle1) și myservo2.write(newAngle2).

De asemenea, funcția lcdDisplay este apelată pentru a afișa unghiurile actualizate ale celor două servomotoare pe ecranul LCD.

```
void lcdDisplay(int angle1, int angle2) {
  clearCharacters();
  lcd.setCursor((unsigned)strlen(TITLE_ANGLE1), 1);
  lcd.print(angle1); // display angle 1
```

```
  lcd.print((char)223);


  lcd.setCursor((unsigned)strlen(TITLE_ANGLE1) + String(angle1).length() + 2, 1);
  lcd.print(TITLE_ANGLE2);
  lcd.print(angle2); // display angle 2
  lcd.print((char)223);
}
```

Funcția lcdDisplay are rolul de a afișa unghiurile curente ale servourilor, iar clearCharacters are rolul de a șterge orice caractere afișate anterior pe ecranul LCD.

lcd.setCursor((unsigned)strlen(TITLE_ANGLE1), 1) identic cu lcd.setCursor((unsigned)strlen(TITLE_ANGLE1) + String(angle1).length() + 2, 1) ne ajută pentru a muta cursorul la poziția corespunzătoare pentru afișarea unghiului 1, respectiv 2, urmând să se afișeaze valoarea unghiului pentru unghiul 1 si 2. Pentru afișarea simbolului ° (grad) am folosit lcd.print((char)223) din biblioteca specifică Arduino pentru LCD.

```
void clearCharacters() {
  for (int i = (unsigned)strlen(TITLE_ANGLE1)  - 1; i <= LCD_CHAR - 1; i++) {
    lcd.setCursor(i, 1);
    lcd.write(254);
  }
}
```

Funcția clearCharacters are rolul de a șterge caracterele afișate anterior pe ecranul LCD. Se parcurge un interval de la (unsigned)strlen(TITLE_ANGLE1) - 1 până la LCD_CHAR - 1.

Pentru fiecare poziție, se setează cursorul pe linia 1 și coloana corespunzătoare.

Pentru fiecare index în intervalul specificat, se setează cursorul LCD la acea poziție și se trimite comanda specială 254 pentru a șterge caracterul de pe ecranul LCD de la acea poziție.

Codul implementat pe modulul ESP32

```
const int led = 13;          // led pin
const int sensor_pin = 36;  // sensor pin
int sensor;                          // sensor reading
const int threshold = 1000;  // threshold to turn LED on


void setup(){  // setup code that only runs once


  pinMode(led, OUTPUT);  // set LED pin as output
  Serial.begin(9600);    // initialize serial communication
```

```
}


void loop(){   // code that loops forever
  sensor = analogRead(sensor_pin);   // read sensor value
  delay(500);
  Serial.println(sensor);                 // print sensor value
  if(sensor>threshold){  // if sensor reading is less than threshold
    digitalWrite(led,HIGH);  // turn LED on
  }
  else{  // else, if sensor reading is greater than threshold
    digitalWrite(led,LOW);    // turn LED off
  }
}
```

Am început prin definirea pinilor care sunt conectați atât la LED cât și la senzorul de lumină. Variabila int sensor stochează valoarea citită de la senzor.

În continuare am setat pinul pentru ieșire, urmând să inițializăm comunicația serială la o viteză de 9600 biți transferați în fiecare secundă. Prin sintaxa sensor = analogRead(sensor_pin) citim valoarea de la senzorul de lumină și o stochăm în variabila sensor.

După ce am stocat valoarea senzorului, introducem o întârziere de 500 de milisecunde pentru a evita citirea repetată și rapidă a senzorului. Apoi, folosim Serial.println(sensor) pentru a afișa valoarea senzorului în monitorul serial, ceea ce ne permite să vedem valoarea actuală a senzorului în timp real.

În final, folosim o structură de control if-else pentru a verifica dacă valoarea senzorului este mai mare decât pragul stabilit. Dacă este, aprindem LED-ul folosind digitalWrite(led,HIGH). Dacă valoarea senzorului este mai mică sau egală cu pragul, stingem LED-ul folosind digitalWrite(led,LOW).

Acest proces se repetă la nesfârșit în bucla loop(), permițând LED-ului să se aprindă și să se stingă în funcție de nivelul de lumină detectat de senzor.

Concluzii

Proiectul a evidențiat importanța unei proiectări atente și a testării riguroase în implementarea unui sistem de control al direcției unui fascicul laser. Integrarea corectă a componentelor hardware și software a fost esențială pentru funcționarea corespunzătoare a sistemului, necesitând o înțelegere detaliată a interacțiunilor între acestea.

Depășirea obstacolelor tehnice, cum ar fi interferența semnalului și gestionarea nivelurilor de lumină ambientală, a necesitat abordări creative și soluții adaptabile.

Am întâmpinat discrepanțe între valorile citite de senzori și poziția reală a servomotoarelor, ducând la erori în poziționarea fasciculului laser și la o monitorizare inexactă a traiectoriei. Am realizat teste și depanări continue pentru a identifica și remedia erorile apărute în poziționarea fasciculului laser și a monitorizării traiectoriei.

Dezvoltarea acestui proiect a adus cu sine oportunitatea de a dobândi competențe avansate în programare embedded, inginerie electronică și control servo.

Finalizarea și succesul proiectului au fost rezultatul unei colaborări eficiente între membrii echipei și a unei abordări metodice în fața problemelor întâmpinate pe parcursul dezvoltării.

Bibliografie

Documente disponibile exclusiv online:

https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

https://epe.utcluj.ro/SMP/SMP_TEMA_2.pdf

https://www.intel.com/content/dam/support/us/en/documents/processors/embedded-processors/galileo_boarduserguide_330237_002.pdf