Projekt Arduino – Interaktywny system z serwomechanizmami i czujnikiem odległości

Opis ogólny:

Projekt został wykonany na platformie **Arduino UNO** i ma na celu stworzenie prostego interaktywnego systemu reagującego na użytkownika. W projekcie wykorzystano ekran LCD z interfejsem I2C, enkoder obrotowy, dwa serwomechanizmy SG90, potencjometr, czujnik odległości HC-SR04 oraz zestaw przycisków.

System posiada **menu sterowane za pomocą enkodera**, które umożliwia użytkownikowi wybór jednej z dwóch funkcji:

1. **Servo Control** – manualne sterowanie kątem dwóch serwomechanizmów przy pomocy potencjometru.
2. **Show Distance** – wyświetlanie odległości od przeszkody oraz dynamiczne sterowanie serwami w zależności od kierunku ruchu.

Schemat podłączenia komponentów:

Zasilanie (rozdzielone przez płytkę stykową):

- **GND z Arduino** → szyna GND (niebieska) na płytce.
- **5V z Arduino** → szyna zasilania (czerwona) na płytce.

Wyświetlacz LCD 16x2 z I2C:

Pin LCD	Pin Arduino UNO
GND	GND
VCC	5V
SDA	A4
SCL	A5

Enkoder obrotowy:

Pin enkodera	Pin Arduino UNO
GND	GND
VCC	5V
CLK	D2
DT	D3

SW (przycisk) D4

📏 ***Czujnik odległości HC-SR04:***

Pin czujnika	Pin Arduino UNO
VCC	5V
GND	GND
TRIG	D5
ECHO	D6

Serwomechanizmy SG90:

Serwo 1:

- Pomarańczowy (sygnał) → D9
- Czerwony (zasilanie) → 5V
- Brązowy (masa) → GND

Serwo 2:

- Pomarańczowy → D10
- Czerwony → 5V
- Brązowy → GND

Potencjometr:

- Lewa nóżka → 5V
- Środkowa → A0
- Prawa nóżka → GND

🔘 ***Przyciski (planowane do rozbudowy):***

- D7, D8, D11, D12, D13 – przygotowane do rozbudowy systemu menu za pomocą przycisków (obecnie nieaktywne).

Działanie systemu:

- **Menu główne** jest wyświetlane na ekranie LCD i obsługiwane za pomocą enkodera obrotowego.
- Po zatwierdzeniu wyboru przyciskiem SW, użytkownik wchodzi do trybu działania.

1. Servo Control:

Użytkownik steruje kątem dwóch serwomechanizmów za pomocą potencjometru – na ekranie wyświetlany jest aktualny kąt.

2. Show Distance:

Na ekranie wyświetlana jest aktualna odległość od obiektu. W tym trybie system analizuje zmianę pozycji obiektu względem czujnika:

- **Gdy obiekt zbliża się** – serwomechanizmy obracają się **zgodnie z ruchem wskazówek zegara**.
- **Gdy obiekt oddala się** – serwomechanizmy obracają się **przeciwnie do ruchu wskazówek zegara**.

Planowane ulepszenia:

- Dodanie obsługi **fizycznych przycisków** do szybkiego przełączania się między trybami bez potrzeby użycia enkodera.
- Implementacja trybu „czuwania” (standby) po dłuższej bezczynności.
- Możliwość regulacji prędkości obrotu serwomechanizmów.
- Ewentualne rozbudowanie o sygnalizację dźwiękową (buzzer) lub wskaźniki LED.

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#include <Servo.h>

// --- LCD ---

LiquidCrystal_I2C lcd(0x27, 16, 2);


// --- Ultrasonic Sensor ---

const int trigPin = 5;

const int echoPin = 6;


// --- Servos ---

Servo servo1;

Servo servo2;
```

```
// --- Potentiometer ---
const int potPin = A0;
// --- Rotary Encoder ---
const int clkPin = 2;
const int dtPin = 3;
const int swPin = 4;
int lastClkState;
int menuIndex = 0;
int totalMenus = 2;

bool actionSelected = false;

void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(clkPin, INPUT_PULLUP);
  pinMode(dtPin, INPUT_PULLUP);
  pinMode(swPin, INPUT_PULLUP);

  servo1.attach(9);
  servo2.attach(10);

  lastClkState = digitalRead(clkPin);

  updateMenu();
}
```

```
void loop() {
  if (!actionSelected) {
    handleEncoder();
    handleButton();
  } else {
    if (menuIndex == 0) {
      int val = analogRead(potPin);
      int angle = map(val, 0, 1023, 0, 180);
      servo1.write(angle);
      servo2.write(angle);

      lcd.setCursor(0, 1);
      lcd.print("Angle: ");
      lcd.print(angle);
      lcd.print("   "); // очищаем остаток
    }

    if (menuIndex == 1) {
      float distance = getDistance();
      lcd.setCursor(0, 1);
      lcd.print("Dist: ");
      lcd.print(distance);
      lcd.print(" cm   ");

      if (distance < 10.0) {
        servo1.write(180);
        servo2.write(180);
      } else {
        servo1.write(0);
        servo2.write(0);
```

```
      }
    }
  }

  delay(100);
}

void updateMenu() {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Menu:");

  lcd.setCursor(0, 1);
  switch (menuIndex) {
    case 0:
      lcd.print("> Servo Control");
      break;
    case 1:
      lcd.print("> Show Distance");
      break;
  }
}

void handleEncoder() {
  int newClkState = digitalRead(clkPin);
  if (newClkState != lastClkState && newClkState == LOW) {
    if (digitalRead(dtPin) != newClkState) {
      menuIndex++;
    } else {
      menuIndex--;
    }
```

```
    if (menuIndex < 0) menuIndex = totalMenus - 1;
    if (menuIndex >= totalMenus) menuIndex = 0;

    updateMenu();
  }
  lastClkState = newClkState;
}

void handleButton() {
  if (digitalRead(swPin) == LOW) {
    delay(200); // debounce
    actionSelected = true;
    lcd.clear();
    if (menuIndex == 0) {
      lcd.setCursor(0, 0);
      lcd.print("Servo Mode Active");
    }
    if (menuIndex == 1) {
      lcd.setCursor(0, 0);
      lcd.print("Distance Mode");
    }
  }
}

float getDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

```
  long duration = pulseIn(echoPin, HIGH);
  float distance = duration * 0.034 / 2;
  return distance;
}
```