תרגיל בית מעשי C: ארדואינו מפעילים וחיישנים

עידו טדיאן 208082685 | קורל מרדכי 206979718

דולב ראובני 207629015 | עדי לוי 319126298

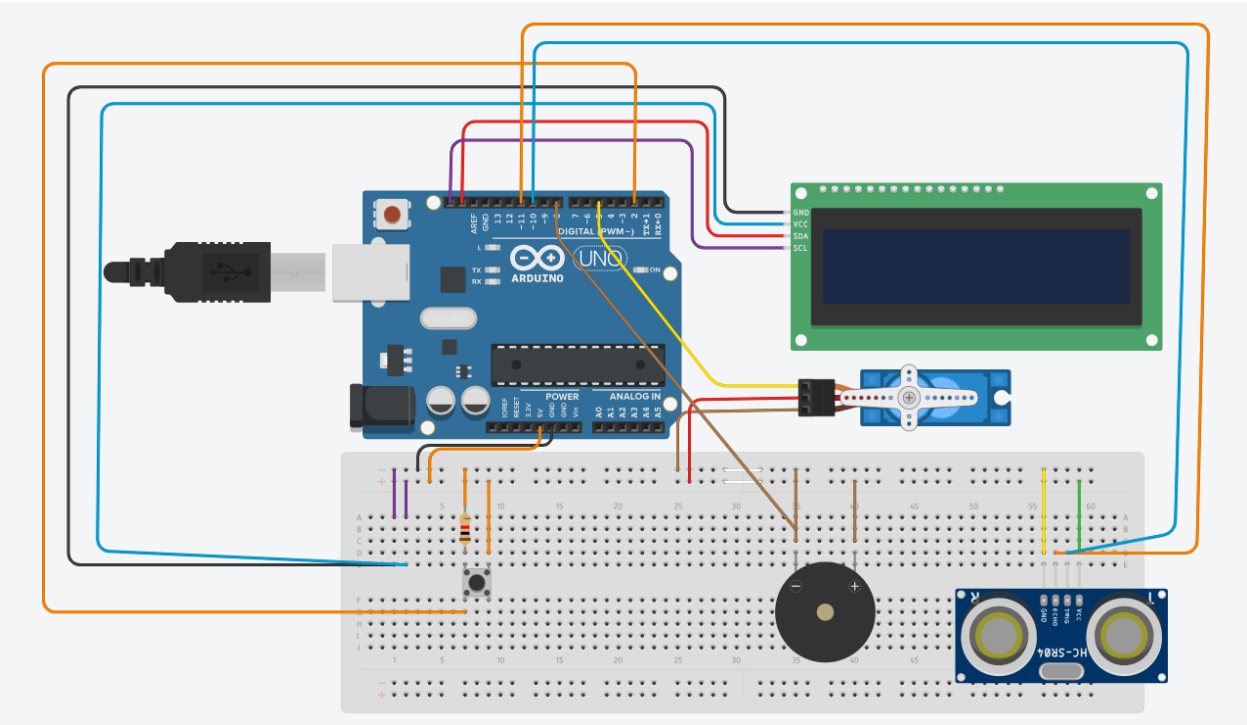
1. המערכת מדמה מערכת מעקב לספירת שכיבות סמיכה עבור מתאמנים. מטרת המערכת היא לוודא כי כל שכיבת סמיכה מתבצעת באופן תקין, כלומר ירידה מלאה של לפחות 10 ס"מ מהקרקע. המערכת מבקשת מהמשתמש לבצע שכיבות סמיכה עד שהספירה תגיע ל-10. בכל פעם שהמשתמש מבצע שכיבת סמיכה בצורה נכונה (כלומר, ירידה של לפחות 10 ס"מ), המערכת מעדכנת את מספר השכיבות שנעשו ומציגה את המספר הנוכחי על מסך ה-LCD. בנוסף, המערכת משמיעה צפצוף כחיווי לביצוע שכיבת הסמיכה בהצלחה. כאשר המשתמש מגיע ליעד של 10 שכיבות סמיכה, המערכת מציגה על מסך ה-LCD הודעת סיום ("Great Job!") מנגנת מנגינת סיום, ומבצעת שני סיבובים של דגל אס"א בן-גוריון.

2. **טבלת רכיבים:**

סוג רכיב	פרמטר	חיבור	תצורת הרכיב	הפעלה (מה מטרת השימוש ברכיב).
מסך LCD	VCC	5V	Output	מסך תצוגה לצורכי חיווי: התחלה, ספירת שכיבות הסמיכה והודעת סיום.
	GND	GND		
	SDA	SDA		
	SCL	SCL		
Active Buzzer (Piezo)	GND	GND	Output	השמעת צליל כמתן משוב לביצוע שכיבת סמיכה כראוי והשמעת מנגינה בהגעה ליעד.
	I/O	8		
	VCC	5V		
Ultrasonic	VCC	5V	Input	חיישן אקטיבי למדידת מרחק האדם מהקרקע בעת שכיבת הסמיכה.
	Trig	10~		
	Echo	11~		
	GND	GND		
Servo	Power	5V	Output	מתן משוב חיובי עבור סיום החזרות על ידי סיבוב דגל.
	GND	GND		
	Signal	5~		
Button	VCC	5V	Input	כפתור התחלת המערכת.
	Signal	2		
	GND	GND (with resistor)		

- בערכה שבידינו לא קיים רכיב Piezo, לכן השתמשנו בActive Buzzer, על כן יש אי הלימה בין הטבלה ובין השרטוט שביצענו בTinkercad.

שרטוט המערכת (סימולטור Tinkercad):



3. לצורך בדיקת תקינות ודיוק פעול חיישן ה-Ultrasonic בו ביצענו שימוש במערכת שפיתחנו, ביצענו 10 הרצות. בכל הרצה, קירבנו את היד לחיישן מספר פעמים (לצורך הדמיה של שכיבת סמיכה), כאשר היעד בכל הרצה הוא 10. תיעדנו את מספר "שכיבות הסמיכה" שנקלטו על ידי המערכת (היעד) לעומת מספר "שכיבות הסמיכה" שבוצעו בפועל. לצורך ניתוח התוצאות וקביעת מובהקות ההבדלים בין ביצועי המערכת למצב בפועל, השתמשנו במבחן t. מבחן זה מאפשר לנו להעריך את רמת הדיוק והאמינות של החיישן ולקבוע האם קיים הבדל מובהק סטטיסטית בין מספר הזיהויים שביצעה המערכת לבין מספר קירובי היד המדמים שכיבות סמיכה שנעשו בפועל.

פירוט ההרצות:

מס' הרצה	מס' שכיבות סמיכה שהמערכת קלטה	מס' שכיבות סמיכה שבוצעו בפועל	Error (שכיבות סמיכה)
1	10	10	2
2	10	10	0
3	10	11	1
4	10	11	1
5	10	10	0
6	10	10	0
7	10	12	2
8	10	10	0
9	10	10	0
10	10	10	1

4. באמצעות כלי ה-Data Analysis בתוכנת Excel, ביצענו מבחן t לשתי דגימות בלתי תלויות עם הנחת שוויון שונויות, כפי שראינו במעבדה 8.

t-Test: Two-Sample Assuming Equal Variances		
	Variable 1	*Variable 2*
Mean	10	10.4
Variance	0	0.488888889
Observations	10	10
Pooled Variance	0.244444444	
Hypothesized Mean Difference	0	
df	18	
t Stat	-1.809068067	
P(T<=t) one-tail	0.043587063	
t Critical one-tail	1.734063607	
P(T<=t) two-tail	0.087174127	
t Critical two-tail	2.10092204	

לפי תוצאות המבחן, התקבל עבור המבחן הדו-צדדי ערך P-Value הגדול מרמת המובהקות המקובלת $\alpha = 5\% < 8.7\% = P\_Value$. לכן, לא נדחה את השערת ה0 (H_0) ונאמר כי אין הבדל מובהק בין מס' שכיבות הסמיכה שנספרו לבין מס' שכיבות הסמיכה בפועל (קירובי היד לחיישן). קיימת סטיה של 4% (ממוצע של 10.4 עבור שכיבות הסמיכה בפועל לעומת 10 עבור ספירת החיישן), המעידה על רמת דיוק של 96%.

5. בעבודה זו פיתחנו מערכת עם בקר ה*Arduino UNO* ורכיבי חומרה שונים: חיישן ה*Ultrasonic* בו השתמשנו למדידת מרחק, *Buzzer* למתן משוב קולי ומסך *LCD* להצגת מידע בזמן אמת בנוגע לתהליך הנמדד (הודעות התחלה וסיום והצגת מידע בנוגע למס' שכיבות הסמיכה שנעשו עד כה). התנסנו בהרכבה פיזית של המערכת, חיבור הרכיבים וכתיבת קוד שמימש את הלוגיקה שהגדרנו – החל מזיהוי תנועה, ספירה ומתן משוב למשתמש. כמו כן, ביצענו מבחן סטטיסטי כדי לבדוק את תקינות פעולת רכיב ה-*Ultrasonic*. תוצאות המבחן הראו כי החיישן פועל באופן תקין עם רמת דיוק של כ-96%.

לשדרוג המערכת והרחבת יכולותיה, נציע שני שדרוגים מרכזיים. ראשית, פיתוח מודול אימון מתקדם הכולל רמות קושי מדורגות, יעדי אימון משתנים וסבבים המוגבלים בזמן או במספר חזרות. שדרוג זה יתאים את המערכת למגוון רחב של מתאמנים ברמות כושר שונות, ויהפוך אותה מכלי ספירה בסיסי למערכת אימון אדפטיבית ומאתגרת. שנית, לשיפור דיוק הספירה נציע הטמעת מערכת זיהוי כפולה באמצעות הוספת חיישן משלים שיפעל במקביל לחיישן הקיים. אימות כפול זה יגדיל משמעותית את אמינות הזיהוי, יצמצם כמעט לחלוטין את הסיכוי לפספוסים, ויבטיח ספירה מדויקת המשקפת נאמנה את ביצועי המתאמן. שילוב שני השדרוגים יעניק למערכת גמישות תפעולית ודיוק מקצועי, ויהפוך אותה לפתרון אימון מקיף ואמין.

נספח א' – קוד המערכת

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h> // Include Servo library

// Set the LCD address to 0x27 for a 16 char, 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Define the pin for the button
const int buttonPin = 2;

// Define ultrasonic sensor pins
const int trigPin = 10;
const int echoPin = 11;

// Define buzzer pin
const int buzzerPin = 8;

// Define servo pin
const int servoPin = 5; // The pin connected to the MG995 servo

// Create a Servo object
Servo myServo;

```

```
// Define variables
int pushUpCount = 0;
long duration;
int distance;
int lastDistance = 100; // Variable to store previous distance
bool isCounting = false; // To track when counting should start

// Servo variables
int angle = 0;
bool Direction = true;
unsigned long rotationStartTime = 0; // Store time when rotation starts
bool rotationInProgress = false; // Flag to indicate whether the rotation should still be happening

void setup() {
  // Initialize the LCD
  lcd.init();
  lcd.backlight();

  // Initialize Serial Monitor
  Serial.begin(9600);

  // Set button pin as input
  pinMode(buttonPin, INPUT);

```

```
  // Set ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Set buzzer pin as output
  pinMode(buzzerPin, OUTPUT);

  // Attach the servo to the specified pin
  myServo.attach(servoPin);
}

void loop() {
  // Read the button state
  int buttonState = digitalRead(buttonPin);

  // If the button is pressed (assuming active HIGH)
  if (buttonState == HIGH) {
    // Show "Start Training" for 5 seconds
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Start Training");
    Serial.println("Training started...");
    delay(5000); // Wait for 5 seconds

```

```
    // Reset push-up count and start training
    pushUpCount = 0;
    lastDistance = 100; // Set lastDistance to something higher than 10 cm to start
    isCounting = true;

    // Now start measuring the distance
    while (isCounting) {
      // Trigger the ultrasonic sensor
      digitalWrite(trigPin, LOW);
      delayMicroseconds(2);
      digitalWrite(trigPin, HIGH);
      delayMicroseconds(10);
      digitalWrite(trigPin, LOW);

      // Measure the time it takes for the echo to return
      duration = pulseIn(echoPin, HIGH);

      // Calculate the distance in cm
      distance = duration * 0.0344 / 2;

      // If the distance is less than 10 cm and lastDistance was greater than 10 cm
      if (distance < 10 && lastDistance >= 10) {
        // Increment push-up count
        pushUpCount++;

        // Print to Serial Monitor
        Serial.print("Push-up done! Current count: ");
        Serial.println(pushUpCount);

        // Display push-up count on LCD
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Push-ups Count: ");
        lcd.setCursor(0, 1);
        lcd.print(pushUpCount);

        // Play beep sound when counter goes up
        tone(buzzerPin, 1000, 200); // Beep with 1000 Hz for 200 milliseconds

        delay(1000); // Wait for 1 second (debounce)

        // When reaching 10 push-ups, display "Great job!"
        if (pushUpCount >= 10) {
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Great job!");
          Serial.println("Training complete. Great job!");

          // Play success music (a simple melody)
          successMusic();

          // Start rotating the servo (rotate back and forth)
          startRotation();

          isCounting = false; // Stop the training after 10 push-ups
        }
      }

      // Update the last distance for comparison
      lastDistance = distance;

      delay(100); // Small delay to avoid sensor noise
    }
  }

  // If rotation is in progress, update the servo movement
  if (rotationInProgress) {
    rotateServo();
  }
}
```

```
// Function to play a success music when "Great job!" is shown
void successMusic() {
  int melody[] = { 262, 294, 330, 349, 392, 440, 494, 523 }; // Frequencies (C4 to C5)
  int noteDuration = 300; // Duration of each note

  // Play a simple ascending melody
  for (int i = 0; i < 8; i++) {
    tone(buzzerPin, melody[i], noteDuration); // Play each note
    delay(noteDuration);
  }
  noTone(buzzerPin); // Stop the buzzer after the melody
}

// Function to start the servo rotation
void startRotation() {
  rotationStartTime = millis(); // Store the current time
  rotationInProgress = true;    // Enable rotation
}

// Function to rotate the servo back and forth for 5 seconds
void rotateServo() {
  unsigned long currentMillis = millis();

  // Check if 5 seconds have passed since rotation started
  if (currentMillis - rotationStartTime < 5000) {
    // If still within 5 seconds, keep rotating the servo
    if (Direction) {
      angle++;
      if (angle == 180) {
        Direction = false;
      }
    } else {
      angle--;
      if (angle == 0) {
        Direction = true;
      }
    }

    myServo.write(angle);
    delay(15); // Smooth rotation
  } else {
    // After 5 seconds, stop rotation and detach the servo
    rotationInProgress = false; // Stop the rotation
    myServo.write(angle); // Make sure the servo holds its last position
    myServo.detach(); // Detach the servo to stop the rotation
  }

}
```