תרגיל בית מעשי C:

ארדואינו מפעילים וחיישנים

|  |  |  |  |
| --- | --- | --- | --- |
| **קבוצה 6** | | | |
| עידו חסאווי | ענבר ארבל | עוז בר-לב | גפן בן-שושן |
| 318963675 | 319009718 | 208659508 | 318287562 |



תוכן עניינים:

[מטרת המערכת: 3](#_Toc186118390)

[משימת הבקרה: 3](#_Toc186118391)

[שרטוט מודל Tinkercad: 3](#_Toc186118392)

[טבלת רכיבים: 4](#_Toc186118393)

[תיאור הבדיקות שבוצעו: 4](#_Toc186118394)

[חומרה: 4](#_Toc186118395)

[לוגיקה: 5](#_Toc186118396)

[תיאור תוצאות הבדיקות: 5](#_Toc186118397)

[חומרה: 5](#_Toc186118398)

[לוגיקה: 6](#_Toc186118399)

[מסקנות: 6](#_Toc186118400)

[נספחים: 8](#_Toc186118401)

[נספח א: 8](#_Toc186118402)

[נספח ב: 9](#_Toc186118403)

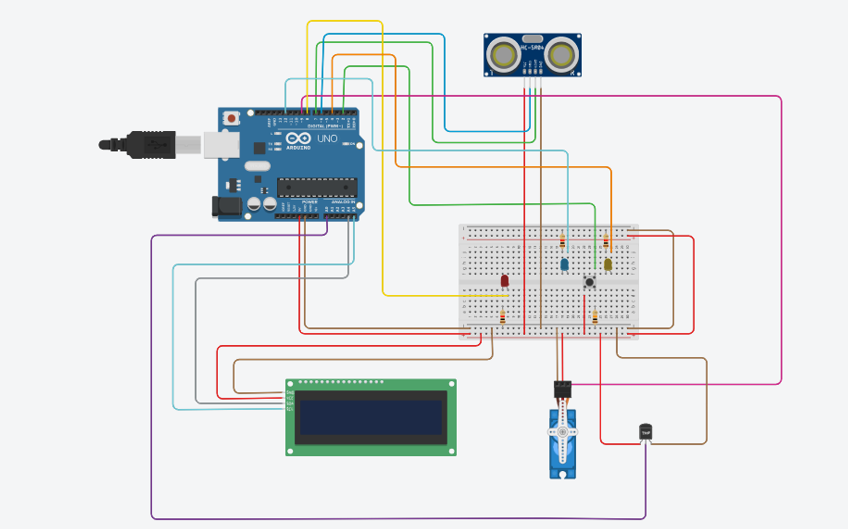
# מטרת המערכת:

מטרת המערכת היא לפתח מנגנון אוטומטי לחלוקת מגבונים לחים הכולל אפשרות לבחירת טמפרטורה. המערכת מזהה משתמש באמצעות חיישן אולטרסוני, מבצעת שליטה בחיישן טמפרטורה ובמנוע סרוו על מנת לחלק מגבונים בהתאם לבחירת המשתמש (טמפרטורה רגילה, טמפרטורה נמוכה, או גבוהה). כמו כן, המערכת כוללת מנגנון לניטור מגבונים שנותרו והודעה על חוסר.

## משימת הבקרה:

* זיהוי משתמש במרחק מתאים באמצעות חיישן אולטרסוני.
* קריאת חיישן טמפרטורה לבקרה על לחות המגבונים
* שימוש במסך LCD לשיפור חווית המשתמש ומתן מידע על מצב המערכת
* הפעלת מנוע סרוו לחילוק מגבונים בהתאם לזיהוי משתמש.
* מתן חיווי באמצעות נורות LED על סטטוס המערכת.

## שרטוט מודל Tinkercad:



# טבלת רכיבים:

|  |  |  |  |
| --- | --- | --- | --- |
| **רכיב** | **מספר פין** | **תצורה** | **מטרה** |
| חיישן טמפרטורה | 0A | INPUT | קריאת טמפרטורה מהסביבה |
| חיישן אולטרסוני | 7,6 | OUTPUT/INPUT | זיהוי משתמש (זיהוי בקשה להוצאת מגבון) |
| מנוע סרוו | 9 | OUTPUT | חלוקת מגבונים |
| מסך LCD | SDA,SCL | OUTPUT | הצגת סטטוס המערכת |
| נורה LED אדום | 8 | OUTPUT | התרעה על גמר מלאי המגבונים |
| נורה LED צהוב וכחול | 12,4 | OUTPUT | מתאר חימום של המגבונים על פי הטמפרטורה המבוקשת |
| כפתור | 2 | INPUT | בחירת טמפרטורה |

# תיאור הבדיקות שבוצעו:

## חומרה:

בבדיקה זו בחנו את דיוקו של חיישן הטווח. נבצע שתי בדיקות במטרה לוודא דיוק החיישן הראשונה השוואה לחיישן נוסף ובדיקה נוספת על ידי השווה למדידה ידנית שבצענו בעזרת סרגל

שלבי הבדיקה:

* נבחר 5 מרחקים קבועים 5 ס"מ, 10 ס"מ, 15 ס"מ, 20 ס"מ, ו-25 ס"מ
* לכל מרחק נבצע:
  + 10 קריאות מחיישן הנבדק
  + 10 קריאות מחיישן נוסף זהה
  + מדידה ידנית באמצעות סרגל כאמת מידה
* נבצע חישוב ממוצעים של הנתונים שהתקבלו
* בדיקת ההשערה
  + השערת האפס : אין הבדל מובהק בין החיישנים והבדיקה הידנית
  + השערה חלופית: קיים הבדל מובהק בין הבדיקות

H0: μe = Manual testing

H1: μe ≠ Manual testing

* + המבחן יבוצע ברמת מובהקות של

## לוגיקה:

מטרת הבדיקה הינה לוודא כי המערכת עובדת לפי התכנון

שלבי הבדיקה:

1. שינוי דרישת טמפרטורה על ידי לחיצה על כפתור
2. הפעלת המנורה המתאימה (צהוב ל15 מעלות וירוק ל30 מעלות) בהתאם לדרישה וטמפ'
3. זיהוי לקוח במרחק בקטן מ5 ס"מ
4. הפעלת הסרוו להוצאת מגבון
5. לאחר בקשה של 4 מגבונים הדלקה של נורה אדומה המסמלת גמר מלאי

# תיאור תוצאות הבדיקות:

## חומרה:

|  |  |  |  |
| --- | --- | --- | --- |
| **מרחק אמיתי**  **)מדידה ידנית)**  **ס"מ** | **חיישן נבדק ממוצע ס"מ** | **חיישן נוסף ממוצע**  **ס"מ** | **p-value** |
| 5 | 5.1 | 5.2 | 0.67 |
| 10 | 10.3 | 10.2 | 0.74 |
| 15 | 15.1 | 15.0 | 0.81 |
| 20 | 20.2 | 20.1 | 0.65 |
| 25 | 25.3 | 25.4 | 0.71 |

* ערכי ה -p בכל המרחקים גבוהים מ-0.05, ולכן אין הבדל מובהק בין תוצאות החיישן הנבדק, החיישן הנוסף, והמדידה הידנית.
* תוצאות אלו מאשרות את השערת האפס: החיישן מדויק ועומד בסטנדרט הנדרש.

## לוגיקה:

|  |  |  |
| --- | --- | --- |
| שלב הבדיקה | הבדיקה | תוצאות |
| שינוי דרישת טמפרטורה על ידי לחיצה על כפתור | לאחר כל לחיצה בדיקה שהתצוגה משתנה במסך ה LCD לתצוגה הרצויה | הכפתור עובד בתצורה תקינה |
| הפעלת המנורה המתאימה (צהוב ל15 מעלות וירוק ל30 מעלות) בהתאם לדרישה וטמפ' | התאמת הטמפ' ע"י קירור או חימום הסביבה במטרה לבדוק הדלקות נורה מתאימה בהתאם לדרישות המשתמש | כאשר המשתמש בחר ב -15 מעלות והסביבה הייתה מתחת לטמפ' זו הפעלה נורה צהובה ובהתאמה גם הנורה הירוקה  חיישן טמפרטורה ונורות תקינים |
| זיהוי לקוח במרחק בקטן מ5 ס"מ | דימוי לקוח הרוצה לקבל מגבון והדפסה של המרחק למערכת ברגע הזיהוי | ע"פ בדיקת החומרה החיישן תקין ובנוסך הודפסו נתונים רק מתחת ל 5 ס"מ משמע המערכת עובדת כראוי |
| הפעלת הסרוו להוצאת מגבון | הסרוו מופעל רק לאחר זיהוי לקוח | הסרוו עובד על פי הנדרש |
| לאחר בקשה של 4 מגבונים הדלקה של נורה אדומה המסמלת גמר מלאי | בגמר מלאי המגבונים נבדוק כי הסרוו לא מסתובב בזיהוי בקשת לקוח חדשה ומנורה אדומה נדלקת | המערכת עובדת כמתוכנן |

* בנוסף לבדיקות המוצגות לאחר כל פעולה מודפסים בקונסול המערכת תיאורים המספקים אינדיקציה על מצב המערכת והתקינות שלה

# מסקנות:

המערכת מתפקדת כראוי ובהתאם לתכנון, ממשק המשתמש קל ונוח לתפעול והבנה. המערכת מזהה משתמשים במדויק באמצעות חיישן אולטרסוניק, מתאימה את טמפרטורת המגבונים לפי בחירת המשתמש, ומספקת חיווי ברור על מלאי המגבונים באמצעות נורת לד. עם זאת, ניכר כי ניתן לשפר את המערכת על ידי שילוב ממשק משתמש מתקדם, התאמה לתנאי סביבה משתנים, וחיווי מדויק יותר לכמות המגבונים שנותרו. באופן כללי, לדעתנו, הפרויקט מממש את מטרותיו

**שדרוג אפשרי:**

1. הוספת אפשרות לבחירה מדויקת יותר של טמפרטורה (לדוגמה, בעזרת פוטנציומטר).
2. שדרוג הממשק הגרפי על מסך הLCD מול המשתמש
3. יכולת שמירה על טמפרטורה של המגבונים (קירור וחימום) לטמפרטורה הרצויה

# נספחים:

## נספח א:

**הוראות שימוש במערכת הארדואינו קבוצה 6:**

**מטרת המערכת:**

מטרת המערכת היא לפתח מנגנון אוטומטי לחלוקת מגבונים לחים הכולל אפשרות לבחירת טמפרטורה.

**הוראות שימוש:**

1. על ידי לחיצה על כפתור ניתן לבחור את טמפרטורת המגבונים מבין שלושת האופציות: טמפ' חדר - דיפולט, 15 מעלות- לחיצה אחת , 30 מעלות-2 לחיצות. הבחירה הרצויה תוצג במסך.
2. יש להצמיד את היד אל מול חיישן הטווח במרחק של עד 5 ס"מ בכדי להגיש בקשה למערכת של שליפת מגבון .
3. המערכת תזהה את הבקשה ותוציא מגבון ללקוח .
4. במידה הטמפרטורה של המגבונים נמוכה מהטמפרטורה הרצויה של הלקוח המערכת תדמה חימום על המגבונים על ידי הדלקה נורה צהובה (15 מעלות) או ירוקה (30 מעלות)

**דגשים נוספים:**

1. בתחילת הפעלת המערכת תבחר כברירת מחדל האופציה של שמירת המגבונים בטמפרטורת החדר (על ידי לחיצת כפתור אוטומטית).
2. הדלקות נורה אדומה מסמנת על גמר מלאי המגבונים ובכך לא ניתן לבקש למשוך מגבון נוסף.
3. על מנת לדמות את המערכת , בחרנו שבחבילת מגבונים יהיו 4 מגבונים בלבד ובסיומם המערכת תגיע לגמר.
4. לאחר הגעת המערכת לגמר יש להטעין מחדש מגבונים במערכת ולאתחל אותה.

## נספח ב:

**תיעוד קוד:**

#include <LiquidCrystal\_I2C.h>

#include <Wire.h>

#include <Servo.h>

#include <OneWire.h>

#include <DallasTemperature.h>

// DS18B20 - Temperature Sensor

#define ONE\_WIRE\_BUS A0 // DS18B20 data pin connected to A0

OneWire oneWire(ONE\_WIRE\_BUS);

DallasTemperature sensors(&oneWire);

float temperature;

// LEDs

const int ledPin = 8; // Empty wipe LED

const int low\_temp = 4; // LED to simulate hitting the device

const int high\_temp = 12; // LED to simulate hitting the device

int counterwipe = 0; // Counter for used wipes

// Servo motor

Servo myServo;

int position = 0;

// LCD

LiquidCrystal\_I2C lcd(0x27, 16, 2); // Set I2C address (adjust as needed)

// Ultrasonic

long duration; // Ultrasonic duration

float distance; // Calculated distance

const int triggerPin = 6; // Ultrasonic sensor trigger pin

const int echoPin = 7; // Ultrasonic sensor echo pin

// Button

const int buttonPin = 2; // Button pin

int buttonState = 0; // Current button state

int lastButtonState = HIGH; // Previous button state (HIGH = not pressed)

int temperatureState = -1; // 0 = No Temp, 1 = Low Temp, 2 = High Temp

bool printTempState = false; // Flag to ensure one-time print per button press

void setup() {

// LEDs

pinMode(ledPin, OUTPUT);

pinMode(low\_temp, OUTPUT);

pinMode(high\_temp, OUTPUT);

// Servo

myServo.attach(9); // Servo connected to pin 9

myServo.write(0); // Ensure servo starts in the neutral position

delay(925);

myServo.write(90);

// Initialize LCD

lcd.init();

lcd.backlight();

lcd.print("choose temperature");

// Configure pins

pinMode(triggerPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(buttonPin, INPUT\_PULLUP); // Button with internal pull-up resistor

// Initialize serial communication

Serial.begin(9600);

Serial.println("System Initialized.");

}

void loop() {

// Read temperature from DS18B20

sensors.requestTemperatures(); // Send the command to get temperatures

temperature = sensors.getTempCByIndex(0); // Read the first sensor's temperature

// Button Handling

buttonState = digitalRead(buttonPin);

if (buttonState == LOW && lastButtonState == HIGH) {

Serial.println("Button Pressed!");

// Update temperature state

temperatureState++;

if (temperatureState > 2) {

temperatureState = 0;

}

printTempState = true; // Enable print flag

delay(100); // Debounce delay

}

if (printTempState) {

Serial.println("The temp is:");

Serial.println(temperature);

// Display temperature state on LCD

lcd.clear(); // Clear LCD before printing new state

if (temperatureState == 0) {

lcd.print("Room Temp");

Serial.println("Room Temperature Selected");

} else if (temperatureState == 1) {

lcd.print("Low Temp: 15 C");

Serial.println("Low Temperature Selected");

if (temperature < 15) {

digitalWrite(low\_temp, HIGH);

delay(5000);

digitalWrite(low\_temp, LOW);

delay(200);

Serial.println("Low Temp: Small action.");

}

} else if (temperatureState == 2) {

lcd.print("High Temp: 30 C");

Serial.println("High Temperature Selected");

if (temperature < 30) {

digitalWrite(high\_temp, HIGH);

delay(5000);

digitalWrite(high\_temp, LOW);

delay(200);

Serial.println("High Temp: Strong action.");

}

}

printTempState = false; // Reset flag

}

lastButtonState = buttonState; // Update button state

// Ultrasonic Sensor

digitalWrite(triggerPin, LOW);

delayMicroseconds(2);

digitalWrite(triggerPin, HIGH);

delayMicroseconds(10);

digitalWrite(triggerPin, LOW);

// Measure echo duration

duration = pulseIn(echoPin, HIGH);

// Calculate distance (cm)

distance = (duration \* 0.0343) / 2;

if (distance < 5 && counterwipe < 4) {

Serial.println("Client detected!");

Serial.print("Distance: ");

Serial.println(distance);

// Servo action to simulate dispensing the wipe

myServo.write(0); // Dispense wipe

delay(925); // Wait for the servo to move to the desired position

myServo.write(90); // Return to neutral position

delay(200); // Wait for servo to return to neutral position

counterwipe++;

Serial.print("Wipe used: ");

Serial.println(counterwipe);

}

// If all wipes are used, turn on the empty wipe LED

if (counterwipe >= 4) {

digitalWrite(ledPin, HIGH); // Out of wipes

}

delay(100); // Avoid overlapping measurements

}