

תרגיל בית מעשי 3 – ארדואינו מפעילים וחיישנים

מגישים: יובל כהן, יותם מולכו, עוז ריגר הכהן, שקד מורג

11 קבוצה

תאריך: 5.6.25

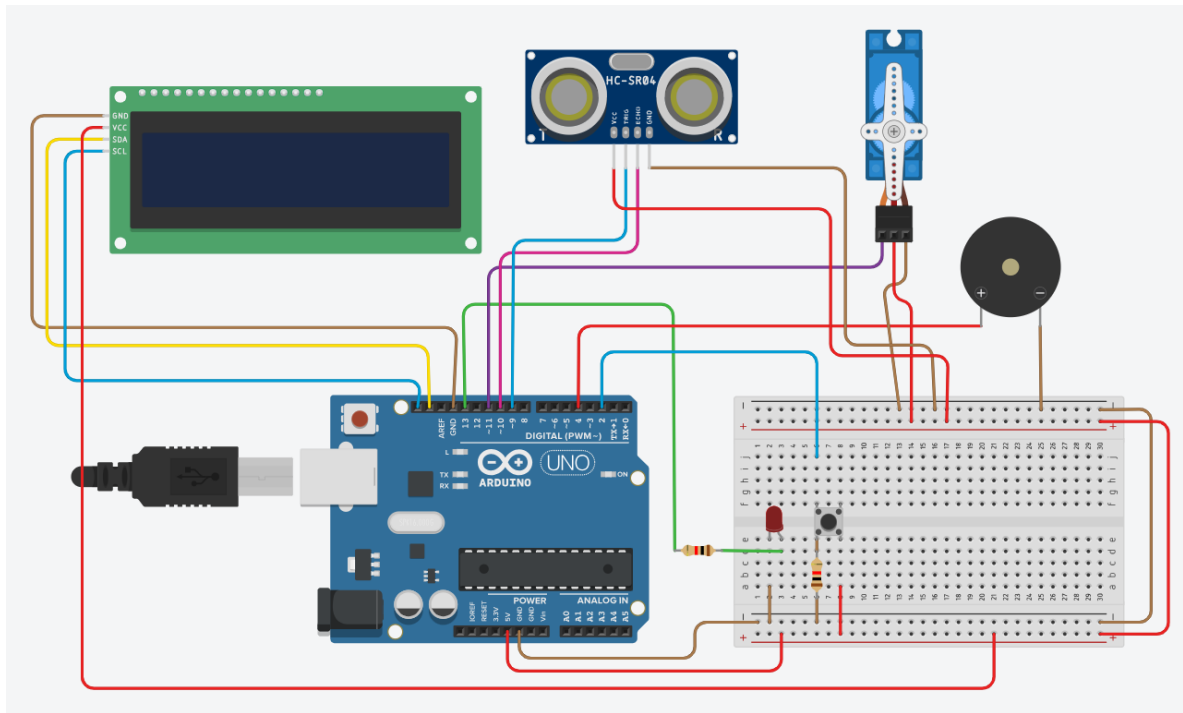
1. תיאור מטרת המערכת ומשימת הבקר:

מטרת מערכת הארדואינו היא להוות בקר אוטומטי שמייעל את שגרת האימונים ומשפר את ביצועי הספורטאים. בכל בוקר, מאמן הקבוצה מגיע למתחם ולוחץ על כפתור ההדלקה של המערכת בשביל להתחיל את עבודת הבקר. בעת הלחיצה על כפתור ההדלקה תהיה הודעת "Welcome" במסך ה LCD. במערכת קיימת מנורה אדומה - כל עוד המערכת פועלת, המנורה נשארת דולקת. אחרת, המנורה כבויה. כל מתאמן שמגיע למתחם האימון צריך לעבור בשער ייעודי. בשער זה מותקנים חיישני תנועה, מנוע סרבו, ובאזר של הבקר. תפקיד חיישן התנועה הוא לזהות תנועה של המתאמנים במרחק של 5 ס"מ ועד 150 ס"מ מהשער. ברגע שחיישן התנועה מזהה תנועה, מנוע הסרבו יתחיל לפעול ויפתח את השער. המנוע יפתח את השער עד לזווית של 90 מעלות, יבצע השהייה של 7 שניות על מנת שהמתאמן יעבור בשער, ולאחר 7 שניות השער ייסגר במלואו. משלב הפתיחה של השער ועד הסגירה שלו - המנורה האדומה תהבהב. בנוסף, ברגע שמנוע הסרבו פותח את השער, הבאזר יתחיל לנגן את המנגינה של השיר " Ode to Joy" במשך 7 שניות על מנת לתת מוטיבציה למתאמן. המערכת תתאפס ותהיה מוכנה להכניס מתאמן חדש רק לאחר שהשער נסגר, מנוע הסרבו סיים לפעול, הבאזר הפסיק לנגן והמנורה האדומה דולקת. מסך ה LCD יהיה בידי המאמן, במסך יופיע מספר המתאמנים שנכנסו למתחם האימונים ומספר זה יתעדכן בלייב.

הנחות יסוד:

- דלת הכניסה תפתח רק כאשר תזהה חיישן תנועה פעיל בטווח תקין, דלת זו תישאר פתוחה למשך 7 שניות (זמן גדול מהנדרש לכניסה) ובסיום הדלת תיסגר. אנו מניחים שהמתאמן מספיק להיכנס בזמן זה.
- רק אדם אחד יכול להיכנס בכל פעימה של פתיחת דלת.

2. תיאור התכן והשרטוט מתוך Thinkercad :



טבלת רכיבים (לפי הסימולציה):

שם הפין	חיבור	תצורה	תיאור
Servo_pin	D11	Output	מנוע סרבו - מבצע תנועה סיבובית
US_Trig	D9	Output	אות שיגור של חיישן מדידת המרחק ותנועה - אולטראסוניק
US_Echo	D10	Input	אות קליטה של חיישן מדידת המרחק ותנועה - אולטראסוניק
LCD	SDA-SDA, SCL-SCL	Output	מסך LCD לתצוגה
Buzzer	D4	Output	באזר להשמעת סאונד
Start_button	D2	Input	כפתור הדלקה של המערכת

3. בדיקות של המערכת:

1. מטרת הבדיקה היא לוודא כי חיישן התנועה (Ultrasonic) מזהה עצמים בטווח הפעולה הנדרש של 5–150 ס"מ בצורה מדויקת, עקבית ואמינה. מאחר שהפעלת המערכת מבוססת על זיהוי מדויק של מתאמן המתקרב לשער, כל סטייה משמעותית במדידת המרחק עלולה להוביל לתפקוד שגוי של המערכת – כגון פתיחה מוקדמת או מיותרת של השער, עיכוב בלתי רצוי בפתיחה, או לחלופין אי־פתיחה של השער כאשר יש בכך צורך. לכן קיימת חשיבות קריטית לאימות רמת הדיוק של החיישן, וזאת באמצעות בדיקת השוואה לערך ייחוס מדוד.

לצורך הבדיקה בחרנו לבדוק את החיישן במרחק קבוע של 20 ס"מ, שהוא ערך מייצג שנמצא בטווח הפעולה המרכזי של המערכת. הונח עצם מדוד במרחק מדויק של 20 ס"מ מהחיישן. בוצעו 20 מדידות של הקריאה שהחיישן סיפק. לכל קריאה חושב ההפרש (סטייה) מהערך הרצוי. הנתונים הוזנו לתוכנת R לביצוע ניתוח סטטיסטי (t-test חד־מדגמי).

נגדיר כי:

השערת האפס (H_0): ממוצע המרחקים שנמדדו על ידי החיישן שווה ל-20 ס"מ

$$H_0: \mu = 20$$

השערה חלופית (H_1): ממוצע המרחקים שנמדדו שונה מ-20 ס"מ (קיימת סטייה מובהקת).

$$H_1: \mu \neq 20$$

4. תיאור תוצאות הבדיקה:

תוצאות המדידה:

מספר מדידה	מרחק (שנמדד) ס"מ
1	20.2
2	20.4
3	20.0
4	20.02
5	19.8
6	19.9
7	20.0
8	20.1
9	20.0
10	19.7
11	20.1
12	20.0
13	19.97
14	20.0
15	20.03
16	20.05
17	20.0
18	19.6
19	19.8
20	20.2

תוצאות המבחן הסטטיסטי:

בוצע ניתוח סטטיסטי באמצעות תוכנת R התקבל ערך p של 0.872, בעוד שרמת המובהקות שנקבעה היא $\alpha = 0.05$. מכיוון ש- $p < 0.05$, לא נדחה את השערת האפס. לפיכך, ניתן להסיק כי אין הבדל מובהק בין ממוצע הקריאות של החיפוש לבין הערך הרצוי של 20 ס"מ. לכן, ניתן להסיק שחיפוש התנועה פועל בדיוק מספק, ועומד בדרישות המערכת. המערכת תוכל להסתמך עליו לצורך זיהוי מתאמנים בכניסה, ללא חשש לטעויות משמעותיות, זאת מאחר וגם הבדיקה נעשתה עם מקדם ביטחון.

2. לצד בדיקת הדיוק של חיפוש התנועה, בוצעה גם בדיקת לוגיקה שמטרתה לוודא שהמערכת פועלת על פי התנאים הלוגיים שתוכננו. המערכת אמורה לפתוח את השער אך ורק כאשר מתקיימים שני תנאים במקביל: זיהוי תנועה בטווח (5–150 ס"מ). בנוסף, המערכת אמורה להתאפס לאחר סיום תהליך הכניסה

– כלומר: השער נסגר, המנורה הירוקה חוזרת לדלוק באופן רציף, הבאזר מפסיק לפעול, והמערכת מוכנה לזיהוי מתאמן חדש.

לצורך הבדיקה בוצעו 6 תרחישים שונים, המתארים מצבים אפשריים בזמן אמת. כל תרחיש בוצע פעמיים.

מספר הבדיקה	מצב המערכת	תגובת המערכת בפועל	האם תקין?
1	זוהתה תנועה בלבד, ללא תג	השער לא נפתח, אין תגובה מהמערכת	תקין
2	הונח תג בלבד, ללא תנועה	השער לא נפתח, אין תגובה מהמערכת	תקין
3	זוהתה תנועה וזיהוי תג RFID	השער נפתח, מנורה מהבהבת, הבאזר פועל	תקין
4	לאחר 7 שניות	השער נסגר, הבאזר הפסיק, המנורה חוזרת לדלוק קבוע	תקין
5	ניסיון כניסה חוזר לפני סגירת השער	אין תגובה – המערכת לא מזהה משתמש נוסף	כן
6	ניסיון כניסה נוסף לאחר סגירה מלאה	המערכת מזהה שוב – התהליך חוזר(כניסה נוספת)	כן

לאורך כל הבדיקות, המערכת פעלה בהתאם ללוגיקת התכנון. היא הגיבה רק כאשר התקיימו שני התנאים (תנועה + תג), והתאפסה בצורה תקינה לאחר כל מעבר. הבדיקה הלוגית מאשרת שהבקר מתפקד באופן מהימן ועקבי בתנאי עבודה שונים.

שתי הבדיקות (הכמותית והלוגית) מראות כי המערכת מתפקדת באופן תקין, מדויק ומהימן. המערכת עומדת בדרישות שתוכננו – הן מבחינת זיהוי פיזי של מתאמן, והן מבחינת תגובה לוגית לפעולה, וניתן להסתמך עליה בסביבת אימון אמיתית.

5. מסקנות מהעבודה:

למדנו המון מהעבודה על הבקר Arduino. ההתנסות המעשית גרמה לנו לחקור ולהבין איך מחברים כראוי את החישניים והעניקה לנו תחושה שאפשר לבצע עם המערכת המון דברים.


אנחנו חושבים שיש לשדרג את המערכת עם הוספה של מערכת RFID אשר תוצב ליד שער הכניסה. עם הוספת מערכת זו, בכל כניסת מתאמן, עליו יהיה להצמיד תג זיהוי לחיישן ובכך נוכל להגיד למאמן איזה מתאמן נכנס, באיזה שעה ואפילו להציג לו נתוני הגעה.

6.נספחים:

תיעוד מאפליקציית R:

```
1 # Sample data
2 data <- c(20.2, 20.4, 20, 20.02, 19.8, 19.9, 20, 20.1, 20, 19.7,
3           20.1, 20, 19.97, 20, 20.03, 20.05, 20, 19.6, 19.8, 20.2)
4
5 # Hypothesized mean
6 hypothesized_mean <- 20
7
8 # Perform one-sample t-test
9 result <- t.test(data, mu = hypothesized_mean)
10
11 # Print the p-value
12 cat("P-value:", result$p.value, "\n")
13 |
```

13:1 (Top Level) ↕

 R 4.4.2 · C:/Users/yuval/Downloads/ ↗

```
> # Sample data
> data <- c(20.2, 20.4, 20, 20.02, 19.8, 19.9, 20, 20.1, 20, 19.7,
+           20.1, 20, 19.97, 20, 20.03, 20.05, 20, 19.6, 19.8, 20.2)
>
> # Hypothesized mean
> hypothesized_mean <- 20
>
> # Perform one-sample t-test
> result <- t.test(data, mu = hypothesized_mean)
>
> # Print the p-value
> cat("P-value:", result$p.value, "\n")
P-value: 0.8721955
```

```

1 # Sample data
2 data <- c(20.2, 20.4, 20, 20.02, 19.8, 19.9, 20, 20.1, 20, 19.7,
3           20.1, 20, 19.97, 20, 20.03, 20.05, 20, 19.6, 19.8, 20.2)
4
5 # Hypothesized mean
6 hypothesized_mean <- 20
7
8 # Perform one-sample t-test
9 result <- t.test(data, mu = hypothesized_mean)
10
11 # Print the p-value
12 cat("P-value:", result$p.value, "\n")

```

```

> # Sample data
> data <- c(20.2, 20.4, 20, 20.02, 19.8, 19.9, 20, 20.1, 20, 19.7,
+           20.1, 20, 19.97, 20, 20.03, 20.05, 20, 19.6, 19.8, 20.2)
>
> # Hypothesized mean
> hypothesized_mean <- 20
>
> # Perform one-sample t-test
> result <- t.test(data, mu = hypothesized_mean)
>
> # Print the p-value
> cat("P-value:", result$p.value, "\n")
P-value: 0.8721955

```


הוראות למפעיל:

1. פתח את תוכנת Arduino IDE והדבק את הקוד המצורף בתחתית העמוד.
2. חבר את בקר ה Arduino למחשב בעזרת חיבור USB.
3. לטעינת הקוד בבקר לחץ על כפתור "וי" - Verify ואז על "חץ ימינה" - Upload.
4. לתחילת פעולת הבקר לחץ על כפתור הלחיצה.
5. שים לב שמסך ה LCD דלוק ומעביר לך מסרים.
6. בשביל לדמות כניסת מתאמן, העבר את היד במרחק של כ 15 ס"מ מחיישן התנועה.
7. כעת המערכת תפתח את השער, תנגן מנגינה ותציג הודעה על מסך ה LCD.
8. שים לב שבסיום פעולת פתיחת השער, מונה כמות המתאמנים שנכנסו למתחם גדל.
9. בשביל לדמות עוד כניסת מתאמן, העבר את היד במרחק של כ 15 ס"מ מחיישן התנועה.
10. בשביל לסגור את המערכת - לחץ על כפתור הלחיצה.

תיעוד קוד:

```
#include <Wire.h>

#include <LiquidCrystal_I2C.h>

#include <Servo.h>

//-----

// הגדרות חיבורים

//-----

#define LED_PIN      3      // מנורה אדומה

#define TRIG_PIN     10     // TRIG : חיישן אולטרה-סוניק

#define ECHO_PIN     13     // ECHO : חיישן אולטרה-סוניק

#define SERVO_PIN    11     // מנוע סרבו

#define BUZZER_PIN   5      // באזר פסיבי

#define BUTTON_PIN   2      // כפתור הפעלה

#define LCD_ADDRESS  0x27   // LCD-של ה I2C כתובת

#define LCD_COLUMNS  16

#define LCD_ROWS     2

//-----

// ספריות ואובייקטים

//-----

LiquidCrystal_I2C lcd(LCD_ADDRESS, LCD_COLUMNS, LCD_ROWS);

Servo      servo;

//-----

// מצב המכונה (State Machine)
```

```

//-----

enum State {

    OFF,           // המערכת כבויה

    STARTING,      // "Welcome" מציג, "Start" לחצו

    IDLE,          // מחכה לזיהוי תנועה

    OPENING_MOVE,  // המנוע נע מ-0° ל-90°+ מנגינה מתחילה

    OPENING_HOLD,  // המנוע ב-90° למשך 7 שניות, מנורה מהבהבת, מנגינה נמשכת

    CLOSING_MOVE,  // המנוע נע מ-180° ל-91°, מנורה מהבהבת

};

State state = OFF;

//-----

// משתנים לניהול המצב

//-----

int entryCount      = 0;           // מונה כניסות

bool lastButtonState = HIGH;       // (INPUT_PULLUP) מצב הכפתור בלולאה הקודמת

// millis(): זמנים מבוססי

unsigned long stateStartTime = 0;   // זמן כניסה לכל מצב

unsigned long nextBlinkTime   = 0;   // מועד הבא להבהבת מנורה

unsigned long nextServoMoveTime = 0; // מועד הבא להזיז את הסרבו ב 1°-

unsigned long holdStartTime   = 0;   // (7 שניות) HOLD זמן תחילת

unsigned long nextNoteTime    = 0;   // מועד הבא לעבור לתו הבא

bool ledBlinkState = LOW;          // מצב הבהוב מנורה

int servoAngle    = 0;             // זווית נוכחית של הסרבו

```

```

//-----

// "Ode to Joy" - מנגינה
//-----

// (תדרי התווים (הערכה של התווים העיקריים במנגינה :

const int melody[] = {

    330, 330, 349, 392, 392, 349, 330, 294,

    262, 262, 294, 330, 330, 294, 294

};

// (500ms משך כל תו במילישניות (כיוון שמנגינה קצרה, כל תו :

const unsigned long noteDurations[] = {

    500, 500, 500, 500, 500, 500, 500, 500,

    500, 500, 500, 500, 500, 500, 500

};

const int melodyLength = sizeof(melody) / sizeof(melody[0]);

int melodyIndex = 0;           // אינדקס התו הנוכחי במנגינה

// (b) (millis-פרקי זמן):

const unsigned long START_DELAY_MS    = 1000;    // STARTING שניה במצב 1
const unsigned long BLINK_INTERVAL    = 500;     // 500ms הבהוב מנורה כל
const unsigned long SERVO_STEP_MS     = 15;      // כל 15 מ"ש מזיזים 1° לסרבו
const unsigned long OPEN_HOLD_MS      = 7000;    // שניות פתוח 7

//-----

// HC-SR04 פונקציה למדידת מרחק (בסנטימטרים) מחיישן
//-----

```

```

float readUltrasonicCM() {

    digitalWrite(TRIG_PIN, LOW);

    delayMicroseconds(2);

    digitalWrite(TRIG_PIN, HIGH);

    delayMicroseconds(10);

    digitalWrite(TRIG_PIN, LOW);


    unsigned long duration = pulseIn(ECHO_PIN, HIGH, 30000);

    if (duration == 0) {

        return 999.0; // Timeout → אין אובייקט קרוב
    }

    float distanceCM = (duration * 0.0343) / 2.0;

    return distanceCM;
}


//-----
// setup()
//-----

void setup() {

    pinMode(LED_PIN,    OUTPUT);

    pinMode(TRIG_PIN,    OUTPUT);

    pinMode(ECHO_PIN,    INPUT);

    pinMode(SERVO_PIN,    OUTPUT);

    pinMode(BUZZER_PIN, OUTPUT);

    pinMode(BUTTON_PIN, INPUT_PULLUP);


    lcd.init();

```

```

    lcd.backlight();

    digitalWrite(LED_PIN, LOW);

    noTone(BUZZER_PIN);

    servo.detach();

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(F("Press to Start"));

    Serial.begin(9600);

    Serial.println(F("System booting → State = OFF"));
}

//-----
// מעבר בין מצבים: פונקציית עזר
//-----

void enterState(State newState) {

    unsigned long now = millis();

    state = newState;

    stateStartTime = now;

    switch (newState) {

        case OFF:

            servo.detach();

            digitalWrite(LED_PIN, LOW);

            noTone(BUZZER_PIN);

            lcd.clear();

```

```

    lcd.setCursor(0, 0);

    lcd.print(F("Press to Start"));

    Serial.println(F("State → OFF"));

    break;

case STARTING:

    digitalWrite(LED_PIN, HIGH);

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(F("Welcome"));

    Serial.println(F("State → STARTING"));

    break;

case IDLE:

    servo.detach();

    noTone(BUZZER_PIN);

    digitalWrite(LED_PIN, HIGH);

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(F("Athletes:"));

    lcd.setCursor(0, 1);

    lcd.print(entryCount);

    Serial.println(F("State → IDLE"));

    break;

case OPENING_MOVE:

    // הסרבו נע מ-0° ל-90°, מנורה מהבהבת, מנגינה מתחילה

```

```

servo.attach(SERVO_PIN);

servoAngle = 0;

servo.write(0);

// התחלת מנגינת "Ode to Joy"
melodyIndex = 0;
nextNoteTime = now;

tone(BUZZER_PIN, melody[melodyIndex]);

digitalWrite(LED_PIN, LOW);

ledBlinkState = LOW;

nextBlinkTime = now + BLINK_INTERVAL;

nextServoMoveTime = now + SERVO_STEP_MS;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print(F("Opening Gate"));

Serial.println(F("State → OPENING_MOVE"));

break;

case OPENING_HOLD:

    // הסרבו ב-90°, מנורה מהבהבת, מנגינה נמשכת עד תומה
    servoAngle = 90;

    servo.write(90);

    holdStartTime = now;

    digitalWrite(LED_PIN, LOW);

    ledBlinkState = LOW;

    nextBlinkTime = now + BLINK_INTERVAL;

    lcd.clear();

    lcd.setCursor(0, 0);

```



```

    lcd.print(F("Holding the door"));

    Serial.println(F("State → OPENING_HOLD"));

    break;

case CLOSING_MOVE:

    // הסרבו נע מ-180° ל-91°, מנורה מהבהבת, מנגינה לא מנוגנת

    servo.attach(SERVO_PIN);

    servoAngle = 180;

    servo.write(180);

    digitalWrite(LED_PIN, LOW);

    ledBlinkState = LOW;

    nextBlinkTime = now + BLINK_INTERVAL;

    nextServoMoveTime = now + SERVO_STEP_MS;

    lcd.clear();

    lcd.setCursor(0, 0);

    lcd.print(F("Closing Gate"));

    Serial.println(F("State → CLOSING_MOVE"));

    break;

}

}

//-----

// BUZZER-פונקציה לעדכון מנגינת ה

//-----

void updateMelody(unsigned long now) {

    // כל עוד אנחנו בסטייט פתיחה או החזקה, ממשיכים לנגן

```

```

    if ((state == OPENING_MOVE || state == OPENING_HOLD) && melodyIndex <
melodyLength) {

        if (now >= nextNoteTime) {

            // עוברים לתו הבא

            melodyIndex++;

            if (melodyIndex < melodyLength) {

                tone(BUZZER_PIN, melody[melodyIndex]);

                nextNoteTime = now + noteDurations[melodyIndex];

            } else {

                noTone(BUZZER_PIN); // סיימנו את המנגינה

            }

        }

    }

}

```

```

//-----

```

```

// loop()

```

```

//-----

```

```

void loop() {

```

```

    unsigned long now = millis();

```

```

    // opening עדכון מנגינה אם אנו במצב

```

```

    updateMelody(now);

```

```

    // (לחוץ = LOW INPUT_PULLUP: קריאת מצב הכפתור)

```

```

    bool buttonState = digitalRead(BUTTON_PIN);

```

```

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

```

```

    if (state == OFF) {

        enterState(STARTING);

    } else {

        entryCount = 0;

        enterState(OFF);

    }

}

lastButtonState = buttonState;

// State Machine -ניהול מצבי ה
switch (state) {

    case OFF:

        // הכל כבוי

        break;

    case STARTING:

        if (now - stateStartTime >= START_DELAY_MS) {

            enterState(IDLE);

        }

        break;

    case IDLE: {

        float distance = readUltrasonicCM();

        Serial.print(F("Measured: "));

        Serial.print(distance);

        Serial.println(F(" cm"));

        if (distance >= 15.0 && distance <= 150.0) {

```

```

    entryCount++;

    enterState (OPENING_MOVE) ;

}

break;

}

case OPENING_MOVE:

    // הבהוב מנורה

    if (now >= nextBlinkTime) {

        ledBlinkState = !ledBlinkState;

        digitalWrite(LED_PIN, ledBlinkState);

        nextBlinkTime += BLINK_INTERVAL;

    }

    // 90° הזזה הדרגתית של הסרבו מ-0° עד

    if (servoAngle < 90 && now >= nextServoMoveTime) {

        servoAngle++;

        servo.write(servoAngle);

        nextServoMoveTime += SERVO_STEP_MS;

        Serial.print(F("Servo angle (up) → "));

        Serial.println(servoAngle);

    }

    // 90° מעבר לשלב ההחזקה כשמגיעים ל

    if (servoAngle >= 90) {

        enterState (OPENING_HOLD) ;

    }

    break;

```

```

case OPENING_HOLD:

    // בהבהוב מנורה

    if (now >= nextBlinkTime) {

        ledBlinkState = !ledBlinkState;

        digitalWrite(LED_PIN, ledBlinkState);

        nextBlinkTime += BLINK_INTERVAL;

    }

    // לאחר 7 שניות, מעבר לסגירה

    if (now - holdStartTime >= OPEN_HOLD_MS) {

        enterState(CLOSING_MOVE);

    }

    break;

case CLOSING_MOVE:

    // הבהוב מנורה

    if (now >= nextBlinkTime) {

        ledBlinkState = !ledBlinkState;

        digitalWrite(LED_PIN, ledBlinkState);

        nextBlinkTime += BLINK_INTERVAL;

    }

    // 91° הורדה הדרגתית של הסרבו מ-180° ל

    if (servoAngle > 91 && now >= nextServoMoveTime) {

        servoAngle--;

        servo.write(servoAngle);

        nextServoMoveTime += SERVO_STEP_MS;

        Serial.print(F("Servo angle (down) → "));

        Serial.println(servoAngle);
    }

```

```
}  
  
// IDLE-ברגע שמגיעים ל-91°, עצירה ל-90° וחזרה ל  
  
else if (servoAngle <= 91) {  
  
    servo.write(90);  
  
    enterState(IDLE);  
  
}  
  
break;  
  
}  
  
}
```