```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Servo.h>

//---------------------
// הגדרות חיבורים
//---------------------
#define LED_PIN        3    // מנורה אדומה
#define TRIG_PIN      10    // חיישן אולטרה-סוניק: TRIG
#define ECHO_PIN      13    // חיישן אולטרה-סוניק: ECHO
#define SERVO_PIN     11    // מנוע סרבו
#define BUZZER_PIN     5    // באזר פסיבי
#define BUTTON_PIN     2    // כפתור הפעלה
#define LCD_ADDRESS  0x27   // כתובת I2C של ה-LCD
#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,       // לחצו "Start", מציג "Welcome"
  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;     // מצב הכפתור בלולאה הקודמת (INPUT_PULLUP)

// זמנים מבוססי millis():
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;     // זמן תחילת HOLD (7 שניות)
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
};
// משך כל תו במילישניות (כיוון שמנגינה קצרה, כל תו 500ms):
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

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

//---------------------
// פונקציה למדידת מרחק (בסנטימטרים) HC-SR04 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;
  }
}

//---------------------
// פונקציה לעדכון מנינגינת ה־BUZZER
//---------------------
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();

  // עדכון מנגינה אם אנו במצב opening
  updateMelody(now);

  // קריאת מצב הכפתור (INPUT_PULLUP: LOW = לחוץ)
  bool buttonState = digitalRead(BUTTON_PIN);
  if (lastButtonState == HIGH && buttonState == LOW) {
    if (state == OFF) {
      enterState(STARTING);
```

```
  } else {
    entryCount = 0;
    enterState(OFF);
  }
}
lastButtonState = buttonState;

// ניהול מצבי ה-State Machine
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;
    }
    // הזזה הדרגתית של הסרבו מ־0° עד 90°
    if (servoAngle < 90 && now >= nextServoMoveTime) {
      servoAngle++;
      servo.write(servoAngle);
      nextServoMoveTime += SERVO_STEP_MS;
      Serial.print(F("Servo angle (up) → "));
      Serial.println(servoAngle);
    }
    // מעבר לשלב ההחזקה כשמגיעים ל־90°
    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;
    }
    // הורדה הדרגתית של הסרבו מ־180° ל־91°
    if (servoAngle > 91 && now >= nextServoMoveTime) {
      servoAngle--;
      servo.write(servoAngle);
      nextServoMoveTime += SERVO_STEP_MS;
      Serial.print(F("Servo angle (down) → "));
      Serial.println(servoAngle);
    }
    // ברגע שמגיעים ל־91°, עצירה ל־90° וחזרה ל־IDLE
```

```
      else if (servoAngle <= 91) {
        servo.write(90);
        enterState(IDLE);
      }
      break;
  }
}
```