ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

UNIVERSITY OF WEST ATTICA

Εξαμηνιαία Εργασία Σχεδίασης Συστημάτων με Μικροελεγκτές 2024 – 2025

6ο Εξάμηνο Σπουδών, Μηχανικών Βιομηχανικής Σχεδίασης Και Παραγωγής

Παντελής - Μάριος Ντζιος 20389216

ΑΥΤΟΜΑΤΙΣΜΟΣ ΘΕΡΜΟΚΗΠΙΟΥ ΜΕ ARDUINO UNO R3

Περιεχόμενα

---

1. Εισαγωγή

Η παρούσα εργασία έχει ως στόχο τη δημιουργία ενός αυτόματου συστήματος ελέγχου για ένα μικρό θερμοκήπιο, χρησιμοποιώντας την πλακέτα Arduino. Σκοπός είναι να διατηρούνται καλές συνθήκες μέσα στο θερμοκήπιο, έτσι ώστε τα φυτά να μεγαλώνουν σωστά, όπως θα γινόταν σε ένα "έξυπνο" περιβάλλον.

Το σύστημα παρακολουθεί συνεχώς δύο βασικά στοιχεία του περιβάλλοντος: τη θερμοκρασία και το φως. Η θερμοκρασία μετριέται με έναν ειδικό αισθητήρα(tmp36), ενώ η φωτεινότητα με έναν φωτοευαίσθητο αισθητήρα (LDR). Οι μετρήσεις εμφανίζονται σε πραγματικό χρόνο σε μια μικρή οθόνη (LCD), ώστε ο χρήστης να βλέπει τι συμβαίνει.

Επίσης, ο χρήστης μπορεί να επιλέξει ποια θερμοκρασία θέλει να διατηρείται μέσα στο θερμοκήπιο, πατώντας κουμπιά. Αν χρειαστεί, ένας ανεμιστήρας ενεργοποιείται

αυτόματα για να ρίξει τη θερμοκρασία. Όλα αυτά τα μέρη συνεργάζονται, κάνοντας το σύστημα ολοκληρωμένο και αποτελεσματικό για τον έλεγχο του περιβάλλοντος μέσα στο θερμοκήπιο.

2. Εκφώνηση Εργασίας

Σκοπός της εργασίας είναι η κατασκευή και επίδειξη σε λειτουργία μιας διάταξης αυτοματισμού θερμοκηπίου μέσω μικροελεγκτή. Ο αυτοματισμός διαβάζει την φωτεινότητα και τη θερμοκρασία του περιβάλλοντος και αναλόγως ελέγχει τον ανεμιστήρα αερισμού και τον μηχανισμό σκίασης του θερμοκηπίου.

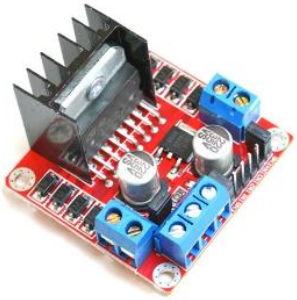
3. Εξοπλισμός

- Arduino Uno R3

Εικόνα 1. Arduino uno R3 grobotronics.gr

Το Arduino Uno αποτελεί την βασική πλακέτα της τεχνολογίας Arduino και προτείνεται για να ξεκινήσετε την εκμάθηση με τα ηλεκτρονικά και τον προγραμματισμό.

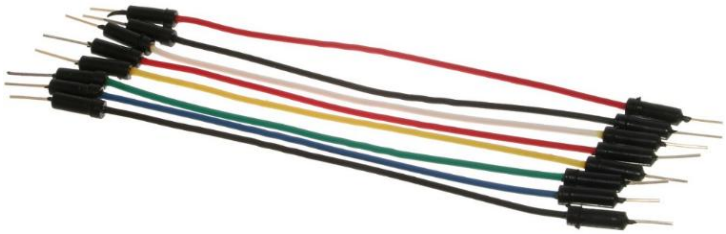
- Motor Driver



Εικόνα 2. Dual Motor Driver Module L298N grobotronics.gr

Το L298N είναι ένα διπλό H-Bridge ολοκληρωμένο κύκλωμα, σχεδιασμένο για τον έλεγχο κινητήρων DC και βηματικών μοτέρ. Μπορεί να ελέγχει την ταχύτητα και την κατεύθυνση περιστροφής δύο μοτέρ ανεξάρτητα, χρησιμοποιώντας PWM (Pulse Width Modulation).

- Jumper Wires



Εικόνα 3. jumper wires Grobotronics.gr

Ένα Jumper wire είναι ένα ηλεκτρικό καλώδιο, ή μια ομάδα από αυτά σε ένα καλώδιο, με έναν σύνδεσμο ή έναν πείρο σε κάθε άκρο (ή μερικές φορές χωρίς αυτά - απλά "κασσιτερωμένο"), το οποίο συνήθως χρησιμοποιείται για τη διασύνδεση των εξαρτημάτων μιας breadboard ή άλλου πρωτότυπου ή κυκλώματος δοκιμής, εσωτερικά ή με άλλο εξοπλισμό ή εξαρτήματα, χωρίς συγκόλληση.

- Hobby Motor 5V DC with gear

Εικόνα 4. Hobby Motor 5V Grobotronics.gr

Αυτός ο μικρός κινητήρας συνεχούς ρεύματος είναι ο ίδιος κινητήρας που χρησιμοποιείται στον κινητήρα συνεχούς ρεύματος TT (DG01D) και μπορεί να χρησιμοποιηθεί για επισκευές. Το μικρό λευκό γρανάζι είναι τοποθετημένο υπό πίεση, μπορεί να αφαιρεθεί με το χέρι και να αντικατασταθεί με το αρχικό, εάν τυχαίνει να είναι διαφορετικό.

- Servo Micro

Εικόνα 5. Servo Micro Grobotronics.gr

Οι κινητήρες servo είναι μικρές συσκευές που έχουν έναν εξωτερικό άξονα Αυτός ο άξονας μπορεί να μετακινηθεί σε διάφορες θέσεις αν αποσταλεί στον servo ένα κατάλληλο σήμα ελέγχου (PWM - Pulse Width Modulation). Το σήμα που του αποστέλλεται καθορίζει τη γωνιακή θέση του άξονα, η οποία συνήθως κυμαίνεται από 0° έως 180°, αν και υπάρχουν και servo με μεγαλύτερο εύρος.

- LCD Display

Εικόνα 6. Display 16x2 Character LCD - 3.3V Blue Grobotronics.gr

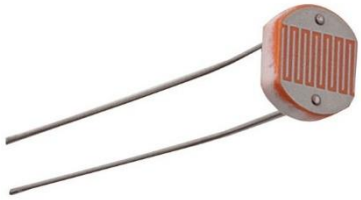
Η οθόνη **16x2 Character LCD - 3.3V Blue** είναι μια μικρή οθόνη υγρών κρυστάλλων που μπορεί να εμφανίσει έως και 2 γραμμές με 16 χαρακτήρες η καθεμία. Λειτουργεί στα 3.3V και διαθέτει μπλε φόντο με λευκούς χαρακτήρες για καθαρή ανάγνωση. Είναι ιδανική για προβολή απλών πληροφοριών όπως μενού, αισθητήρες ή μηνύματα σε μικροελεγκτές όπως Arduino.

- Buttons

Εικόνα 7. Button Grobotronics.gr

Τα κουμπιά στο Arduino λειτουργούν ως διακόπτες εισόδου, επιτρέποντας στον χρήστη να δίνει εντολές ή να ενεργοποιεί λειτουργίες με το πάτημα ενός πλήκτρου. Συνδέονται εύκολα σε ψηφιακές εισόδους για ανίχνευση πατήματος.

- Φωτοαντίσταση RLDR



Εικόνα 8. LDR Φωτοαντισταση Grobotronics.gr

Η φωτοαντίσταση είναι μια μεταβλητή αντίσταση η τιμή της οποίας αλλάζει ανάλογα με το φως που πέφτει πάνω σε αυτή.

- Αισθητήριο θερμοκρασίας

Εικόνα 9. TMP36 Αισθητήριο θερμοκρασίας Grobotronics.gr

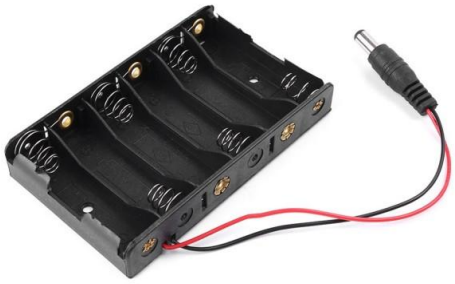
Αισθητήρας Θερμοκρασίας TMP36 με αναλογική έξοδο. Το εύρος θερμοκρασίας όπου μετράει είναι από -40°C έως +125°C με ακρίβεια ±2°C. Η τροφοδοσία κυμαίνεται μεταξύ 2.7V έως 5.5V DC. Συμβατός με τις περισσότερες αναπτυξιακές πλακέτες, όπως το Arduino.

- Φωτοεκπομποί δίοδοι LED

Εικόνα 10. Red LED Grobotronics.gr

Τα LED χρησιμοποιούνται για οπτική ένδειξη καταστάσεων ή συμβάντων και ελέγχονται μέσω ψηφιακών εξόδων. Ανάβουν ή σβήνουν στέλνοντας αντίστοιχα HIGH ή LOW σήματα.

- Battery supply



Εικόνα 11. Μπαταριοθήκη Grobotronics.gr

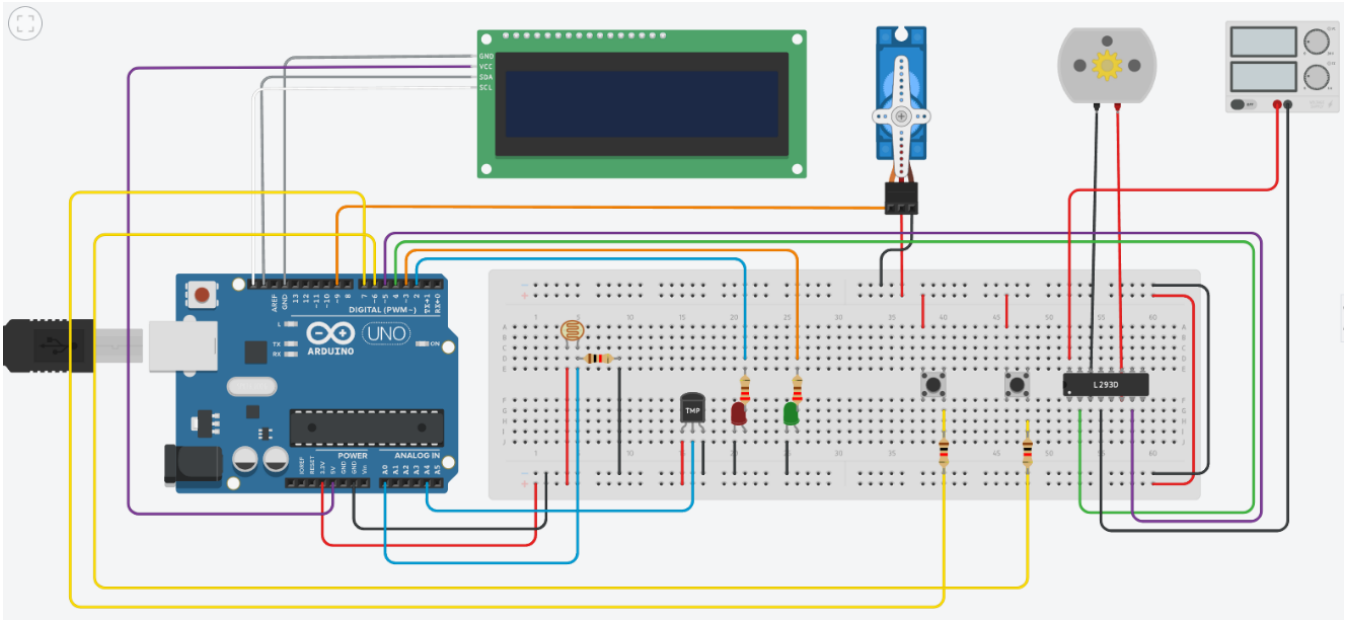
Βάση Πλαστική για Μπαταρίες ΑΑ. Μπορούν να τοποθετηθούν 6 μπαταρίες και διαθέτει υποδοχή για 5.5 x 2.1.

4. Κύκλωμα Μικροελεγκτή

Το κύκλωμα του μικροελεγκτή όπως φαίνεται στη παρακάτω φωτογραφία, αποτελεί το κεντρικό σύστημα ελέγχου του θερμοκηπίου, ενσωματώνοντας όλους τους απαραίτητους αισθητήρες για την αυτόνομη λειτουργία του.

- **Μικροελεγκτής Arduino:** Το Arduino λειτουργεί ως ο "εγκέφαλος" του συστήματος, αναλαμβάνοντας την επεξεργασία των δεδομένων από τους αισθητήρες και την αποστολή εντολών στους ενεργοποιητές. Στο κύκλωμα, διακρίνονται οι εξής βασικές συνδέσεις:
- **Αισθητήρας Θερμοκρασίας (TMP36):** Ο αισθητήρας είναι συνδεδεμένος στην αναλογική είσοδο A2 του Arduino και χρησιμοποιείται για τη συνεχή μέτρηση της θερμοκρασίας του περιβάλλοντος εντός του θερμοκηπίου.
- **Αισθητήρας Φωτεινότητας (LDR):** Ο φωτοευαίσθητος αντιστάτης (LDR) συνδέεται στην αναλογική είσοδο A0 του Arduino και χρησιμοποιείται για τη μέτρηση των επιπέδων φωτεινότητας, προκειμένου να ελεγχθεί το σύστημα σκίασης.
- **Οθόνη LCD (I2C):** Η οθόνη Liquid Crystal Display συνδέεται μέσω του πρωτοκόλλου I2C (στους ακροδέκτες SDA και SCL του Arduino) και χρησιμοποιείται για την προβολή σε πραγματικό χρόνο των μετρήσεων θερμοκρασίας και φωτεινότητας, καθώς και μηνυμάτων του συστήματος.
- **Κουμπιά Ελέγχου (SW1 & SW2):** Δύο ψηφιακά κουμπιά (συνδεδεμένα στα Pins 6 και 7) επιτρέπουν στον χρήστη να αλληλεπιδρά με το σύστημα, κυρίως για τη ρύθμιση της επιθυμητής θερμοκρασίας στόχου.
- **Servo Motor:** Το servo motor (συνδεδεμένο στο Pin 9) αποτελεί τον μηχανισμό σκίασης του θερμοκηπίου. Ελέγχεται από το Arduino για να προσαρμόζει τη σκίαση αναλόγως των επιπέδων φωτεινότητας.
- **DC Motor (Ανεμιστήρας) & L298N Motor Driver:** Ο ανεμιστήρας του συστήματος αερισμού είναι ένας κινητήρας συνεχούς ρεύματος (DC Motor), ο οποίος ελέγχεται από τον οδηγό κινητήρων L298N. Οι ακροδέκτες IN1 και IN2 του driver είναι συνδεδεμένοι στα ψηφιακά Pins 4 και 5 του Arduino αντίστοιχα, επιτρέποντας την ενεργοποίηση/απενεργοποίηση του ανεμιστήρα. Δεδομένου ότι τα pins ENA και ENB του L298N είναι βραχυκυκλωμένα, ο κινητήρας λειτουργεί μόνο σε πλήρη ταχύτητα.
- **LEDs Ενδείξεων (LED1 & LED2):** Δύο φωτοεκπομποί δίοδοι (LED1 στο Pin 2 και LED2 στο Pin 3) παρέχουν οπτική ένδειξη της λειτουργίας του ανεμιστήρα (LED1) και του μηχανισμού σκίασης (LED2) αντίστοιχα.
- **Τροφοδοσία:** Μια εξωτερική συστοιχία μπαταριών (9V) παρέχει την απαραίτητη τάση λειτουργίας τόσο στο Arduino όσο και στον οδηγό κινητήρων L298N, διασφαλίζοντας την αυτονομία του συστήματος.

Το ολοκληρωμένο αυτό κύκλωμα επιτρέπει την συνεργασία όλων των επιμέρους στοιχείων, ώστε το σύστημα του θερμοκηπίου να παρακολουθεί, να επεξεργάζεται δεδομένα και να ρυθμίζει αυτόνομα τις περιβαλλοντικές συνθήκες σύμφωνα με τις προκαθορισμένες παραμέτρους.



Σχέδιο 1. Αποτύπωση Ηλεκτρονικού Κυκλώματος Θερμοκηπίου TinkerKad.

5. Ανάλυση Λογισμικού Μέρους

5.1 Δομή του Προγράμματος

Η βασική του δομή ακολουθεί το πρότυπο προγραμματισμού του Arduino (setup() και loop()), εμπλουτισμένο με τη χρήση εξωτερικών βιβλιοθηκών και ειδικά σχεδιασμένων custom συναρτήσεων.

Βασικές Δομές και Οργάνωση Προγράμματος:
Το πρόγραμμα αξιοποιεί τις ακόλουθες κύριες δομές, οι οποίες συνεργάζονται εντός ενός συστήματος καταστάσεων (nProgramState) για τη διαχείριση της ροής εκτέλεσης:

- **Global Variables:** Στην αρχή του κώδικα, δηλώνονται όλες οι παγκόσμιες μεταβλητές που αφορούν τα pins, τα αντικείμενα των βιβλιοθηκών (LCD, Servo), τους αισθητήρες, τις σταθερές βαθμονόμησης και τις μεταβλητές κατάστασης του προγράμματος.
- **setup() Συνάρτηση:** Εκτελείται μία φορά κατά την έναρξη του Arduino. Περιλαμβάνει την αρχικοποίηση της Serial Communication, τον ορισμό όλων των pins (για κουμπιά, αισθητήρες, LEDs, servo, motor driver) ως εισόδους/εξόδους, την αρχικοποίηση της LCD και του Servo. Στη συνέχεια, θέτει την αρχική κατάσταση του προγράμματος σε **"Λειτουργία Προγραμματισμού Θερμοκρασίας"** και καλεί τη συνάρτηση readTemperatureSetting() για την έναρξη αυτής της φάσης.
- **loop() Συνάρτηση:** Εκτελείται συνεχώς μετά την ολοκλήρωση του setup(). Περιέχει ένα switch statement που ελέγχει την τρέχουσα κατάσταση του προγράμματος (nProgramState), επιτρέποντας την εκτέλεση διαφορετικών λειτουργιών ανάλογα με τη φάση:
    - **case 0 (Programming Mode):** Διαχειρίζεται την είσοδο και την παραμονή στην κατάσταση προγραμματισμού της θερμοκρασίας. Εμφανίζει μήνυμα "Control: OFF" και καλεί ξανά τη

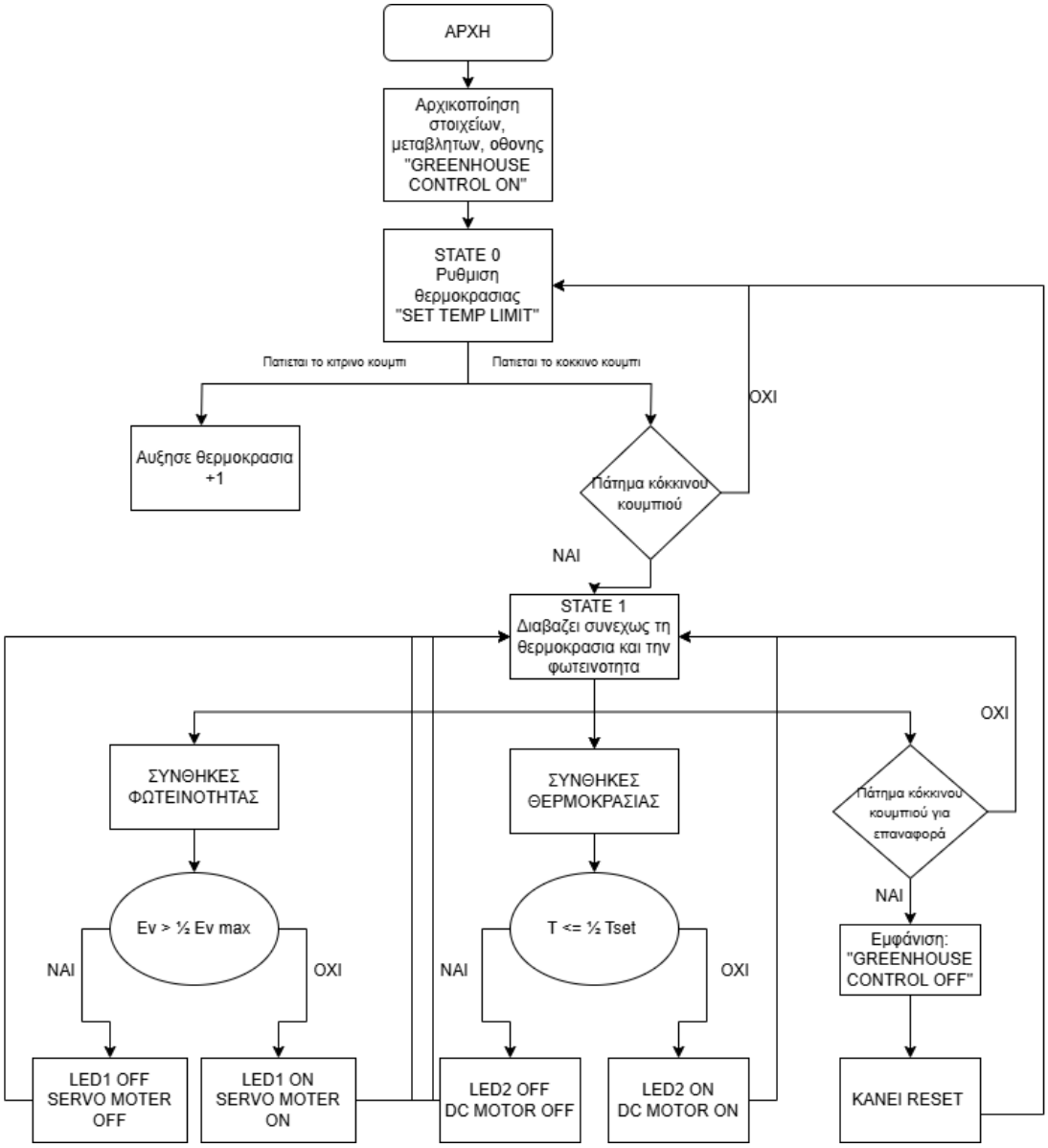
readTemperatureSetting(), η οποία επιτρέπει στον χρήστη να ρυθμίσει την επιθυμητή θερμοκρασία-στόχο μέσω των κουμπιών. Σε αυτή τη φάση, οι ενεργοποιητές (ανεμιστήρας, LED1) απενεργοποιούνται.

- **case 1 (Normal Operation Mode):** Αυτή είναι η κύρια φάση λειτουργίας του συστήματος, όπου πραγματοποιείται συνεχής παρακολούθηση και έλεγχος.
  - Ελέγχει αν το κόκκινο κουμπί πατήθηκε, για πιθανή μετάβαση πίσω στην κατάσταση προγραμματισμού.
  - **Ανάγνωση Αισθητήρων:** Διαβάζει συνεχώς τις τιμές από τους αισθητήρες θερμοκρασίας (readTemperatureC()) και φωτεινότητας (readLightSensorLux()).
  - **Εμφάνιση Δεδομένων:** Εμφανίζει τις τρέχουσες μετρήσεις (Φωτεινότητα Lux, Θερμοκρασία °C) στην οθόνη LCD.
  - **Έλεγχος Σκίασης:** Υλοποιεί τη λογική ελέγχου του servo motor (βάσει φωτεινότητας και ορίου Ev_half), προσαρμόζοντας τη θέση του servo για σκίαση και ελέγχοντας το LED2.
  - **Έλεγχος Αερισμού:** Υλοποιεί τη λογική ελέγχου του DC motor (ανεμιστήρα) και του LED1 (βάσει θερμοκρασίας και ορίου T_set_half), ενεργοποιώντας ή απενεργοποιώντας τον ανεμιστήρα σε πλήρη ταχύτητα (δεδομένης της βραχυκυκλωμένης σύνδεσης ENA/ENB του L298N driver).
  - Περιλαμβάνει μια μικρή καθυστέρηση (delay(800)) πριν την επόμενη επανάληψη, διασφαλίζοντας την πολυπλεξία στην επεξεργασία.

- **Βιβλιοθήκες (Libraries):**
  - **LiquidCrystal_I2C.h**: Παρέχει συναρτήσεις για τον έλεγχο της I2C LCD οθόνης (π.χ., init(), print(), setCursor(), clear(), backlight()).
  - **Servo.h**: Επιτρέπει τον έλεγχο των servo motors (π.χ., attach(), write()).
- **Custom Συναρτήσεις (Custom Functions):**
  - **readLightSensorLux()**: Διαβάζει το LDR και μετατρέπει την αναλογική τιμή σε Lux, βασιζόμενη σε βαθμονόμηση (πλήρες σκοτάδι → 0 Lux, πλήρες φως → MAX_LUX).
  - **readTemperatureC(int pin)**: Διαβάζει τον αισθητήρα θερμοκρασίας (TMP36) από συγκεκριμένο pin, εκτυπώνει την αναλογική τιμή στον Serial Monitor και τη μετατρέπει σε βαθμούς Κελσίου με προσαρμοσμένο συντελεστή.
  - **scrollText(int row, String message, int delayTime, int lcdColumns)**: Δημιουργεί ένα οριζόντιο εφέ κύλισης κειμένου στην οθόνη LCD.
  - **readTemperatureSetting()**: Διαχειρίζεται ολόκληρη τη διαδικασία ρύθμισης της θερμοκρασίας-στόχου από τον χρήστη μέσω των κουμπιών, συμπεριλαμβανομένης της εμφάνισης στην LCD και της μετάβασης σε κατάσταση κανονικής λειτουργίας.

Αυτή η δομή επιτρέπει την αποτελεσματική διαχείριση των διαφόρων λειτουργιών του θερμοκηπίου, εξασφαλίζοντας την ταυτόχρονη παρακολούθηση και απόκριση στις περιβαλλοντικές συνθήκες.

5.2 Διάγραμμα Ροής



5.3 Λογισμικό (Software)

/********** Greenhouse Control System
 * Purpose
 * This program implements an automated greenhouse control system using an Arduino
 * microcontroller. It continuously monitors temperature and luminance, displays
 * these values on an LCD, and controls a shading mechanism (servo) and a ventilation
 * fan (DC motor via L298N driver) based on environmental conditions and user settings.

```
 *
 * Hardware
 * - Arduino microcontroller (e.g., Uno)
 * - I2C LCD Display (0x27 address, 16x2 characters) connected via SDA/SCL pins
 * - Yellow Pushbutton (SW1) connected to Digital Pin 6 (for temperature increment)
 * - Red Pushbutton (SW2) connected to Digital Pin 7 (for temperature
confirmation/mode switch)
 * - Temperature Sensor (TMP36) connected to Analog Pin A2
 * - Light Dependent Resistor (LDR) connected to Analog Pin A0 (with pull-down/up
resistor)
 * - Servo Motor (for shading) connected to Digital Pin 9
 * - DC Motor (for ventilation fan) connected to L298N Motor Driver
 * - L298N IN1 connected to Digital Pin 4
 * - L298N IN2 connected to Digital Pin 5
 * - L298N ENA/ENB pins are shorted (motor operates at full speed only)
 * - L298N powered by an external 9V battery pack (e.g., 6xAA batteries)
 * - LED1 connected to Digital Pin 2 (indicates fan operation)
 * - LED2 connected to Digital Pin 3 (indicates shading operation)
 *
 * Software
 * - Uses Arduino standard libraries: LiquidCrystal_I2C.h for LCD, Servo.h for servo
control.
 * - Employs a state machine approach for program flow (Programming Mode /
Normal Operation).
 * - Custom functions are used for sensor reading, text scrolling, and temperature
setting.
 * - Motor control is implemented via digitalWrite for on/off control due to shorted
L298N enable pins.
 *
 ********************/

*(Υπογραμμίζω τα σχόλια να είναι πιο ευκολοδιάβαστο)*

#include <LiquidCrystal_I2C.h> // Include library for I2C LCD displays
#include <Servo.h>             // Include library for controlling servo motors

// Declare LCD screen dimensions
int lcdColumns = 16; // Number of columns on the LCD screen
int lcdRows = 2;     // Number of rows on the LCD screen

// Declare program state variable
int programState; // Stores the current operating state (0: programming, 1: running)

// Declare pushbutton pin variables
int yellowButtonPin = 6; // Digital pin for the yellow button (SW1)
int redButtonPin = 7;    // Digital pin for the red button (SW2)

// Declare Servo motor object and pin variables
Servo mainServo;           // Create a Servo object to control a servo motor
int servoControlPin = 9;   // Digital pin connected to the servo motor's signal wire
```

```
int servoPosition;          // Variable to store the current position of the servo motor

// Declare motor driver input pins for L298N (no PWM speed control as ENA/ENB are shorted)
int motorInput1Pin = 11; // Digital pin connected to IN1 of the L298N motor driver
int motorInput2Pin = 5; // Digital pin connected to IN2 of the L298N motor driver

// Declare LED pin variables
int led1Pin = 2; // Digital pin for LED1 (fan status indicator)
int led2Pin = 3; // Digital pin for LED2 (shading status indicator)

// Define sensor pins using preprocessor macros
#define tempSensorPin A2 // Analog pin for the temperature sensor
#define ldrSensorPin A0  // Analog pin for the Light Dependent Resistor (LDR)

// Declare temperature target variable
float targetTemperature = 1; // Variable to store the user-set target temperature for fan control

// Initialize LCD object (I2C address, columns, rows)
LiquidCrystal_I2C lcd(0x27, 16, 2); // Create LCD object with I2C address 0x27, 16 columns, 2 rows
String welcomeMessage = "GreenHouse Program"; // String to hold the scrolling welcome message

// Define LDR calibration constants
const int LDR_ANALOG_MAX_READING = 1023; // Maximum analog reading from LDR in complete darkness (or fully covered)
const float MAX_LUX_CALIBRATED = 200.0;  // Maximum Lux value corresponding to the brightest light reading

// Declare environmental thresholds for shading control
float luminanceMax = MAX_LUX_CALIBRATED; // Maximum luminance, used for scaling
float luminanceHalfThreshold = luminanceMax / 2.0; // Half of max luminance, threshold for shading

// Declare threshold for fan control
float temperatureSetHalf; // Half of the user-set target temperature (targetTemperature / 2.0)

float readLightSensorLux() {
  int ldrAnalogValue = analogRead(ldrSensorPin); // Read the analog value from the LDR sensor

  // Map the analog value (0-1023) to a Lux range (0-MAX_LUX_CALIBRATED)
  // This mapping assumes:
  //   - Analog value 0 (brightest light) corresponds to MAX_LUX_CALIBRATED (max Lux)
  //   - Analog value 1023 (darkest light) corresponds to 0 Lux
```

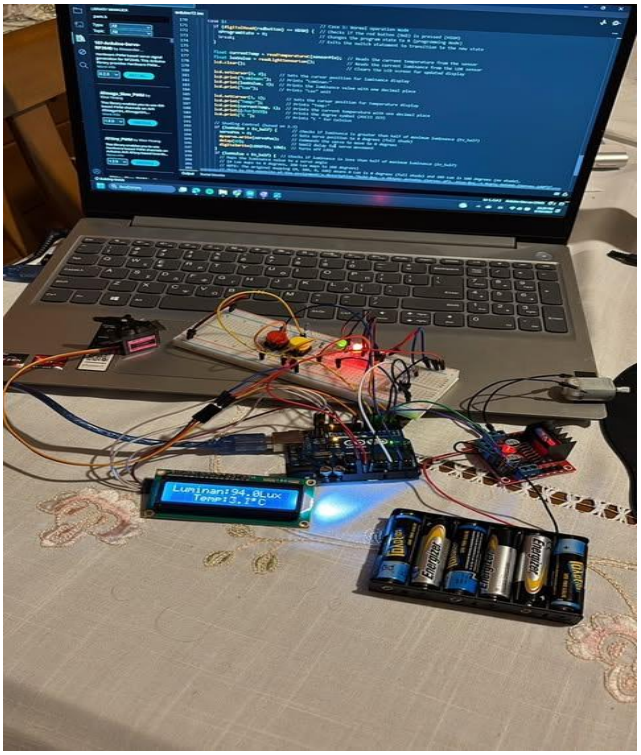
```
  // The 'map' function performs a linear re-scaling.
  float lux = map(ldrAnalogValue, 0, LDR_ANALOG_MAX_READING,
MAX_LUX_CALIBRATED, 0);

  // Constrain the Lux value to be within valid boundaries (0 to
MAX_LUX_CALIBRATED)
  if (lux < 0) {
    lux = 0;
  }
  if (lux > MAX_LUX_CALIBRATED) {
    lux = MAX_LUX_CALIBRATED;
  }
  return lux; // Return the calculated Lux value
}

float readTemperatureC(int pin) {
  int analogValue = analogRead(pin); // Read the analog value from the specified
sensor pin
  Serial.print("Raw Analog Value (Temp Sensor): "); // Print a label to the Serial
Monitor
  Serial.println(analogValue);                    // Print the raw analog value to the Serial
Monitor

  // Convert the analog value to Celsius using a custom calibration factor.
  // This factor (0.036977) was previously determined to adjust the sensor's
  // reading to match expected real-world temperatures (e.g., 622 -> 23 °C).
  float temperatureC = analogValue * 0.036977;

  Serial.print("Calculated Temp (Adjusted): "); // Print a label to the Serial Monitor
  Serial.println(temperatureC, 1);              // Print the calculated temperature (with 1
decimal place)
                                  // to the Serial Monitor.
  return temperatureC; // Return the calculated temperature in Celsius
}

void scrollText(int row, String message, int delayTime, int lcdColumns) {
  for (int i = 0; i < lcdColumns; i++) { // Loop to add leading spaces to the message for
initial off-screen position
    message = " " + message;             // Adds a space to the beginning of the message
  }
  message = message + " "; // Add a trailing space to allow the entire message to scroll
off-screen
  for (int pos = 0; pos < message.length(); pos++) { // Loop through each character
position to simulate scrolling
    lcd.setCursor(0, row); // Set the cursor to the beginning of the specified row
    lcd.print(message.substring(pos, pos + lcdColumns)); // Print a segment of the
message that fits the LCD width
    delay(delayTime); // Pause for the scrolling animation effect
  }
}
```

```
void readTemperatureSetting() {
  // Display initial welcome messages on LCD
  lcd.setCursor(3, 0);  // Set cursor position for "GreenHouse"
  lcd.print("GreenHouse"); // Print "GreenHouse"
  lcd.setCursor(5, 1);  // Set cursor position for "Control"
  lcd.print("Control"); // Print "Control"
  delay(3000);          // Pause for 3 seconds

  // Clear LCD and display "Set Temp Limit" and scrolling welcome message
  lcd.clear();          // Clear the LCD screen
  lcd.setCursor(0, 1);  // Set cursor position for "Set Temp Limit"
  lcd.print("Set Temp Limit"); // Print "Set Temp Limit"
  lcd.setCursor(0, 0);  // Set cursor position for scrolling message
  scrollText(0, welcomeMessage, 250, lcdColumns); // Call function to scroll the
welcome message
  delay(2000);          // Pause for 2 seconds

  // Clear LCD and display instructions for temperature adjustment
  lcd.clear();          // Clear the LCD screen
  lcd.setCursor(0, 0);  // Set cursor position for instruction line 1
  lcd.print("Press SW1/SW2 to"); // Print instruction
  lcd.setCursor(0, 1);  // Set cursor position for instruction line 2
  lcd.print("+/Enter Temp:"); // Print instruction
  lcd.setCursor(14, 1); // Set cursor position for target temperature value
  lcd.print(targetTemperature); // Print the current target temperature

  // Loop to allow user to adjust target temperature
  while (digitalRead(redButtonPin) != HIGH) { // Continue looping until the red
button (SW2/Enter) is pressed
    if (digitalRead(yellowButtonPin) == HIGH) { // If the yellow button (SW1/+) is
pressed
      targetTemperature = targetTemperature + 1; // Increment target temperature by 1
      // Check if target temperature exceeds upper limit (60 degrees) and reset if needed
      if (targetTemperature == 61) {
        targetTemperature = 1; // Reset target temperature to 1 if it exceeds 60
      }
      delay(500); // Debounce delay for the button press

      // Update LCD display with new target temperature
      lcd.setCursor(0, 0); // Reset cursor position
      lcd.print("Press SW1/SW2 to"); // Re-print instruction
      lcd.setCursor(14, 1); // Reset cursor position for value
      lcd.print(targetTemperature); // Print the updated target temperature
      lcd.setCursor(0, 1); // Reset cursor position
      lcd.print("+/Enter Temp:"); // Re-print instruction
    }
  }

  // Display "Greenhouse Control: ON" message after temperature setting is confirmed
```

```
  lcd.clear();          // Clear the LCD screen
  lcd.setCursor(3, 0);  // Set cursor position
  lcd.print("Greenhouse"); // Print "Greenhouse"
  lcd.setCursor(2, 1);  // Set cursor position
  lcd.print("Control: ON"); // Print "Control: ON"
  delay(3000);          // Pause for 3 seconds

  programState = 1; // Set program state to 1 (Normal Operation Mode)
}

// setup function: This function runs once when the Arduino board powers up or
resets.
void setup() {
  Serial.begin(9600);          // Initialize serial communication at 9600 bits per second
  pinMode(yellowButtonPin, INPUT); // Configure the yellow button pin as an input
  pinMode(redButtonPin, INPUT);    // Configure the red button pin as an input
  pinMode(ldrSensorPin, INPUT);    // Configure the LDR sensor pin as an input
  pinMode(tempSensorPin, INPUT);   // Configure the temperature sensor pin as an
input

  mainServo.attach(servoControlPin); // Attach the Servo object to its control pin
(enables servo control)
  lcd.init();                // Initialize the LCD module
  lcd.clear();               // Clear any existing content on the LCD screen
  lcd.backlight();           // Turn on the backlight of the LCD display

  pinMode(led2Pin, OUTPUT);     // Configure LED2 pin as an output (for shading
status)
  pinMode(led1Pin, OUTPUT);     // Configure LED1 pin as an output (for fan status)

  // Configure the L298N motor driver input pins as outputs
  pinMode(motorInput1Pin, OUTPUT);    // Configure motorInput1Pin as an output
  pinMode(motorInput2Pin, OUTPUT);    // Configure motorInput2Pin as an output
  // Note: No pinMode for motorEnablePin is needed as ENA/ENB are shorted on the
L298N driver.

  mainServo.write(180);          // Set initial servo position to 180 degrees (no shade)
  delay(15);                 // Short delay to allow the servo to reach its initial position

  programState = 0;          // Set the initial program state to 0 (temperature
programming mode)
  readTemperatureSetting();     // Call the function to begin the temperature setting
process
}

// loop function: This function runs repeatedly after the setup() function completes.
void loop() {
  switch (programState) { // Evaluate the current program state
    case 0: // Case 0: Programming mode (setting target temperature)
      lcd.clear();          // Clear the LCD screen
```

```
    lcd.setCursor(3, 0);  // Set cursor position
    lcd.print("Greenhouse"); // Print "Greenhouse"
    lcd.setCursor(2, 1);  // Set cursor position
    lcd.print("Control: OFF"); // Print "Control: OFF"

    // Ensure the DC motor and LED1 are turned off when exiting normal
operation/entering programming mode
    digitalWrite(motorInput1Pin, LOW); // Set motorInput1Pin to LOW to stop the
motor
    digitalWrite(motorInput2Pin, LOW); // Set motorInput2Pin to LOW to stop the
motor
    digitalWrite(led1Pin, LOW);        // Turn off LED1

    delay(3000);              // Pause for 3 seconds
    readTemperatureSetting(); // Call the function to re-enter the temperature setting
process
    break;                    // Exit the switch statement
  case 1: // Case 1: Normal operation mode
    if (digitalRead(redButtonPin) == HIGH) { // Check if the red button (SW2) is
pressed (HIGH)
      programState = 0; // Change the program state to 0 (programming mode)
      break;           // Exit the switch statement to transition to the new state
    }

    // Read sensor values
    float currentTemperature = readTemperatureC(tempSensorPin); // Read the current
temperature from the sensor
    float luxValue = readLightSensorLux();                    // Read the current
luminance from the LDR sensor
    lcd.clear(); // Clear the LCD screen for updated display

    // Display sensor values on LCD
    lcd.setCursor(0, 0); // Set cursor position for luminance display
    lcd.print("Luminan:");  // Print "Luminan:"
    lcd.print(luxValue, 1); // Print the luminance value with one decimal place
    lcd.print("Lux");       // Print "Lux" unit

    lcd.setCursor(3, 1); // Set cursor position for temperature display
    lcd.print("Temp:");    // Print "Temp:"
    lcd.print(currentTemperature, 1); // Print the current temperature with one decimal
place
    lcd.print((char)223);  // Print the degree symbol (ASCII 223)
    lcd.print("C ");       // Print "C " for Celsius

    // Shading Control (based on Section 2.3 of the assignment)
    if (luxValue > luminanceHalfThreshold) { // If luminance is greater than half of
maximum (bright conditions)
      servoPosition = 0;          // Set servo position to 0 degrees (full shade)
      mainServo.write(servoPosition); // Command the servo to move to 0 degrees
      delay(50);                  // Small delay for servo movement
```

```
      digitalWrite(led2Pin, LOW);  // Turn off LED2 (as per provided code, if full
shade is not "operation")
                                   // NOTE: Based on previous discussion, if full shade is
considered "operation",
                                   // then LED2 should be HIGH here. Reconfirm assignment's
exact wording.
    }
    if (luxValue < luminanceHalfThreshold) { // If luminance is less than half of
maximum (less bright conditions)
      // Current implementation is kept as per provided code, but be aware of this
behavior.
      servoPosition = map(luxValue, 0, 200, 0, 180);
      mainServo.write(servoPosition); // Command the servo to move to the calculated
position
      delay(50);                   // Small delay for servo movement
      digitalWrite(led2Pin, HIGH);    // Turn on LED2 (during analogous shading
operation)
    }

    // Airing System Control (based on Section 2.4 of the assignment)
    temperatureSetHalf = targetTemperature / 2.0; // Calculate half of the user-set
target temperature

    if (currentTemperature > temperatureSetHalf) { // If current temperature is greater
than the set threshold
      // Activate the DC motor at full speed (since ENA/ENB are shorted, variable
speed is not possible)
      // The assignment asks for "variable angular speed", but with shorted ENA/ENB
this is not achievable
      // with the current L298N connection. The motor will operate at full speed.
      digitalWrite(motorInput1Pin, HIGH); // Set IN1 to HIGH for one direction of
rotation
      digitalWrite(motorInput2Pin, LOW);  // Set IN2 to LOW for the chosen direction
      digitalWrite(led1Pin, HIGH);        // Turn on LED1 to indicate fan operation
    } else { // If current temperature is less than or equal to the set threshold
      // Deactivate the DC motor
      digitalWrite(motorInput1Pin, LOW); // Set IN1 to LOW to stop the motor
      digitalWrite(motorInput2Pin, LOW); // Set IN2 to LOW to stop the motor
      digitalWrite(led1Pin, LOW);        // Turn off LED1
    }

    delay(800); // Pause for 800 milliseconds before the next loop iteration
    break;      // Exit the switch statement
  }
}
```



Εικόνα 12. Στιγμιότυπο από την υλοποίηση της εργασίας

6. Συμπεράσματα

Ολοκληρώνοντας αυτή την εργασία, είμαι στην ευχάριστη θέση να πω ότι το αυτοματοποιημένο σύστημα ελέγχου θερμοκηπίου με το Arduino λειτουργεί επιτυχώς! Κατάφερα να παρακολουθώ συνέχεια τη θερμοκρασία και τη φωτεινότητα, βλέποντας όλες τις πληροφορίες ζωντανά στην οθόνη LCD.
Πιο συγκεκριμένα, το σύστημα αποδείχθηκε πολύ αποτελεσματικό σε δύο βασικούς τομείς:

- **Σκίαση:** Το μοτεράκι (servo) για τη σκίαση, χάρη στον αισθητήρα φωτεινότητας (LDR), προσαρμόζει αυτόματα τη σκιά, και το LED2 μας δείχνει πότε λειτουργεί.
- **Αερισμός:** Ο ανεμιστήρας, που ελέγχεται από τον οδηγό L298N και αντιδρά στην ένδειξη του αισθητήρα θερμοκρασίας (TMP36), ενεργοποιείται αυτόματα όταν ζεσταίνει πολύ, με το LED1 να ανάβει για να το υποδείξει.

Ήταν πολύ ενδιαφέρον να βλέπω πώς το σύστημα ανταποκρίνεται στις συνθήκες και η δυνατότητα να ρυθμίζω εγώ την επιθυμητή θερμοκρασία με τα κουμπιά πρόσθεσε μια ωραία πινελιά ελέγχου. Παρόλο που ο ανεμιστήρας λειτουργεί σε πλήρη ταχύτητα λόγω του τρόπου σύνδεσης του L298N, η διαδικασία υλοποίησης αυτής της εργασίας ήταν ιδιαίτερα απαιτητική ειδικά καθώς εργάστηκα μόνος και οι αρχικές μου γνώσεις στον προγραμματισμό ήταν περιορισμένες. Ωστόσο, μέσα από αυτή τη διαδικασία, έμαθα πάρα πολλά για την αλληλεπίδραση υλικού και λογισμικού, και κατάφερα να εφαρμόσω τις γνώσεις μου σε ένα πρακτικό project. Τέλος, πιστεύω πως αυτό το μικρό θερμοκήπιο αποτελεί ένα λειτουργικό παράδειγμα για το πώς η τεχνολογία μπορεί να βοηθήσει στην έξυπνη γεωργία.

7. Βιβλιογραφία

[1] Arduino. "Arduino Official Documentation." Available: https://www.arduino.cc/reference/en/. [Accessed: [Current Date, e.g., 11-Jun-2025]].
[2] S. Monk, *Programming Arduino: Getting Started with Sketches*, 2nd ed. McGraw-Hill, 2016.
[3] Adafruit. "Adafruit LCD I2C Backpack." Available: https://learn.adafruit.com/i2c-lcd-backpack. [Accessed: [Current Date, e.g., 11-Jun-2025]].
[4] Arduino. "Servo Library." Available: https://www.arduino.cc/reference/en/libraries/servo/. [Accessed: [Current Date, e.g., 11-Jun-2025]].
[5] Analog Devices. "TMP36/TMP37 Low Voltage, Temperature Sensors Datasheet." Available: https://www.analog.com/media/en/technical-documentation/data-sheets/tmp35_36_37.pdf. [Accessed: [Current Date, e.g., 11-Jun-2025]].
[6] STMicroelectronics. "L298 Dual Full-Bridge Driver Datasheet." Available: https://www.st.com/resource/en/datasheet/l298.pdf. [Accessed: [Current Date, e.g., 11-Jun-2025]].
[7] R. Balogh, “Motor Driver L298N – A Practical Guide for Robotics Applications,” *Robotics Journal*, vol. 12, no. 4, pp. 55-67, 2013.
[8] SparkFun Electronics. "Button Hookup Guide." Available: https://learn.sparkfun.com/tutorials/button-hookup-guide. [Accessed: [Current Date, e.g., 11-Jun-2025]].
[19] robobill.gr. "Breadboard - Τι είναι και πώς λειτουργεί;" Available: https://robobill.gr/breadboard-ti-einai-kai-pws-peitourgei/. [Accessed: [Current Date, e.g., 11-Jun-2025]].
[10] Wikipedia. "Jump wire." Available: https://en.wikipedia.org/wiki/Jump_wire. [Accessed: [Current Date, e.g., 11-Jun-2025]].