Univerza v Ljubljani
Fakulteta za strojništvo

University of Ljubljana
Faculty of Mechanical Engineering

Rok Oražem

Automated Garden Irrigation System

Seminar paper for the course Mechatronic Actuators

Ljubljana, 2026

Contents

1. Introduction

1.1 Background and Motivation

Modern agriculture and home gardening face increasing challenges in optimizing natural resources. Manual watering of large garden beds not only causes uneven moisture distribution but also leads to excessive and unnecessary consumption of drinking water from the mains. By utilizing affordable and soft-programmable microcontrollers, such as the Arduino platform, it is possible to develop closed-loop mechatronic systems. These systems enable precise irrigation based on real environmental parameters, while supporting ecologically oriented approaches, such as rainwater harvesting.

1.2 Project Objectives

The main objective of this project work is the implementation of an automated garden irrigation system. The target garden area measures 8×3 m, representing a total of 24 m^2 of surface area, primarily dedicated to growing lettuce.

The system must meet the following technical and functional criteria:

- **Sustainable source:** The system must operate exclusively by harvesting rainwater from an IBC collection tank with a capacity of 1000 liters.
- **Optimal irrigation method:** Use of micro-spray technology that ensures wetting of the foliage and the shallow root system.
- **Autonomy and control:** Fully autonomous operation of the microcontroller without user intervention, guided by moisture sensors and real-time data.
- **System safety:** Integration of protective filters against nozzle clogging and implementation of an electronic safety mechanism against dry running of the pump.

2. Hydraulic System Design and Calculations

2.1 Irrigation Area Analysis

Most of the irrigation area is used for growing lettuce, which has specific moisture requirements. Due to its shallow root system, which extends into the top 10 to 15 cm of the soil, it reacts quickly to drought stress. A lack of moisture causes tissue lignification, a bitter taste, and premature bolting (going to seed). For optimal development, an overhead micro-sprayer method was selected. In

addition to moistening the soil, these sprayers lower the microclimate temperature around the leaves during the peak of summer heat.

To cover the dimensions of 8×3 m, Gardena $180°$ spray nozzles from the Micro-Drip series were chosen. The nozzles operate in a fan-shaped arc with a nominal radius range of $r = 2.75$ m to 3.0 m at appropriate pressure. They are installed along both longer sides in an arc pattern for the best coverage.

(a) Garden

(b) Spray nozzle

2.2 Hydraulic Calculation of Consumption and Flow Parameters

One Gardena spray nozzle consumes $Q_{\text{nozzle}} = 100$ liters of water per hour (l/h) at the nominal operating pressure $p_{\text{op}} = 1.5$ bar. For the entire 8 m length of the garden, we mount $n = 3$ nozzles. The total volume flow rate of the system Q_{total} is calculated using the equation:

$$Q_{\text{total}} = n \cdot Q_{\text{nozzle}} = 3 \cdot 100 \,\frac{\text{l}}{\text{h}} = 300 \,\frac{\text{l}}{\text{h}} \tag{2.1}$$

Converted into international and practical gardening units:

$$Q_{\text{total}} = \frac{300}{60} \,\frac{\text{l}}{\text{min}} = 5 \,\text{l/min} \tag{2.2}$$

2.3 Determination of Pressure Head and Pump Selection

To ensure proper droplet dispersion, we must consider the pressure drop along the pipeline, the resistance in the fittings, and the pressure drop across the fine mesh filter, which is necessary due to particles in the rainwater. The total required pump pressure p_{pump} is determined as:

$$p_{\text{pump}} = p_{\text{op}} + \Delta p_{\text{friction}} + \Delta p_{\text{filter}} + \rho g \Delta z \tag{2.3}$$

Where Δz is the geodetic height (the difference between the bottom of the tank and the nozzle level, approx. 1.5 m), ρ is the density of water ($1000\,\text{kg/m}^3$), and g is the gravitational acceleration ($9.81\,\text{m/s}^2$). The predicted pressure drops are $\Delta p_{\text{friction}} \approx 0.2$ bar and $\Delta p_{\text{filter}} \approx 0.3$ bar.

$$p_{\text{pump}} = 1.5\,\text{bar} + 0.2\,\text{bar} + 0.3\,\text{bar} + 0.15\,\text{bar} = 2.15\,\text{bar} \tag{2.4}$$

Since manufacturers of submersible pumps express pressure in terms of maximum delivery head (H_{max}), we use the relation $10\,\text{m}\ \text{H}_2\text{O} \approx 1\,\text{bar}$:

$$H_{\text{max}} \geq 21.5\,\text{m} \tag{2.5}$$

The ideal choice would be a surface pump with a delivery head of $H_{\text{max}} = 25\,\text{m}$ ($2.5\,\text{bar}$), but its price is approximately €80. Since I already had a Parkside submersible pump available at home with a slightly lower delivery head, I compensated for this shortcoming by using more nozzles.

3. Detailed Description and Specification of Hardware Components

For the successful realization of the mechatronic system, the selection of industrially stable components capable of operating under conditions of increased humidity and outdoor temperature fluctuations is crucial.

3.1 Arduino Uno Microcontroller

The core of the electronic control is the Arduino Uno microcontroller board, based on the 8-bit ATmega328P integrated circuit. The main technical characteristics of the controller are:

- **Clock speed:** $16\,\text{MHz}$, which enables fast processing of sensor data and real-time execution of safety loops.
- **Memory:** $32\,\text{KB}$ of Flash memory (of which approximately $0.5\,\text{KB}$ is occupied by the bootloader), $2\,\text{KB}$ of SRAM for variables, and $1\,\text{KB}$ of EEPROM for permanent storage of calibration constants.
- **Power supply:** The operating voltage of the microcontroller is $5\,\text{V}$. The enclosure power is supplied via an external AC/DC adapter connected to the built-in DC barrel jack or to the V_{in} pin with a voltage of $7 - 12\,\text{V}$, while the onboard regulator steps down the voltage to a stable $5\,\text{V}$.
- **Input/Output pins:** The board offers 14 digital pins (6 of which support pulse-width modulation - PWM) and 6 analog inputs with a built-in 10-bit analog-to-digital converter (ADC).

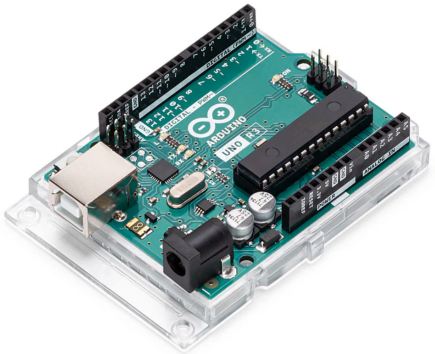

Figure 3.1: Arduino Uno

3.2 Capacitive Soil Moisture Sensor v1.2

Unlike classic, low-cost resistive sensors with two exposed metal prongs, the selected capacitive sensor version 1.2 is not subject to galvanic corrosion.

- **Operating principle:** The sensor measures soil capacitance, which changes according to the dielectric permittivity of the surroundings. Since water has a significantly higher relative permittivity ($\varepsilon_r \approx 80$) than dry soil ($\varepsilon_r \approx 3-5$), the capacitance of the circuit changes linearly with the moisture level.
- **Circuitry:** A timer circuit with a permanent high-frequency oscillator and a TL0725 operational amplifier are integrated on the sensor board. This generates an analog voltage signal ranging between 1.5 V (completely wet) and 3.0 V (completely dry).
- **Connection:** The sensor requires three wires for operation: V_{CC} (5 V), GND (ground), and Aout (analog output).

Figure 3.2: Capacitive soil moisture sensor v1.2

3.3 RTC DS1302 Real-Time Clock Module

Since Arduino lacks an internal mechanism to maintain accurate calendar time during a power outage, the system includes a DS1302 real-time clock (RTC) module.

- **Accuracy and design:** This is a classic and cost-effective integrated circuit for time tracking, operating in conjunction with an external 32.768 kHz crystal oscillator. Since it lacks built-in temperature compensation, minor time deviations (a few minutes per month) may occur depending on outdoor temperature fluctuations.
- **Communication:** It communicates with the Arduino via a simple 3-wire serial interface. Instead of the I^2C bus, it uses three arbitrary dedicated digital lines: RST/CE (reset/chip enable), DAT (data), and CLK (clock). This allows greater flexibility in choosing free digital pins on the controller.
- **Backup power:** The module contains a slot for a lithium coin cell battery (e.g., CR2032), which ensures continuous clock operation and retention of 31 bytes of internal static RAM for up to 10 years without external power.

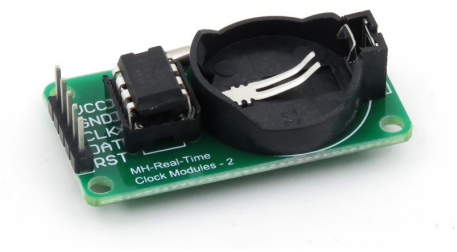

Figure 3.3: RTC DS1302 real-time clock module

3.4 Electromechanical Relay Module (5V / 230V)

Switching of the pump's power circuit is performed via a single-channel relay module.

- **Specifications:** The relay (manufacturer Songle, model SRD-05VDC-SL-C) allows switching of AC voltages up to $250\,\mathrm{V}$ AC and currents up to $10\,\mathrm{A}$, which is sufficient for starting the induction motor of the selected pump.
- **Optical isolation:** The module includes an optocoupler (optoisolator) that galvanically isolates the control section (Arduino) from the power section. This prevents the transfer of voltage spikes and electromagnetic interference (noise), generated during the switching of the inductive load (pump motor), back to the processor.
- **Control:** The relay is triggered by a digital signal (HIGH/LOW) from digital pin D10.

Figure 3.4: Relay module

3.5 Parkside PTPK 400 Submersible Clean Water Pump

To pump rainwater from the collection IBC tank and supply the irrigation network, a Parkside PTPK 400 submersible clean water pump is used. The pump acts as the main executive (actuator) element of the mechatronic system, which is indirectly switched on by the relay module via a digital signal.

The main technical and functional characteristics of the pump are:

- **Nominal power:** The pump is equipped with an energy-efficient $400\,\mathrm{W}$ motor operating at a standard mains voltage of $230\,\mathrm{V}/50\,\mathrm{Hz}$. The power provides the stable torque required for continuous long-term operation during watering cycles.
- **Flow capacity:** The maximum volume flow rate of the pump is $Q_{\mathrm{max}} = 10000\,\mathrm{l/h}$, which exceeds the total nominal consumption of the spray nozzles with a large margin and allows rapid filling of the pipeline upon startup.

- **Delivery head and pressure:** The maximum delivery head (H_{max}) is $7\,\text{m}$, which corresponds to a nominal pressure of $0.7\,\text{bar}$. The pump allows submersion down to an operating depth of $7\,\text{m}$ and shallow suction of clean water up to a remaining level of approx. $5\,\text{mm}$ above the bottom of the tank.
- **System protection and housing:** The device features an integrated adjustable float switch for dry-running protection, which cuts off power if the water level is too low. The robust housing is made of impact-resistant plastic with a high protection rating of IPX8, ensuring permanent watertightness and safe operation in a submerged environment.

Figure 3.5: Parkside PTPK 400 submersible pump

3.6 Filters and Pressure Regulation (Gardena Master Unit 1000)

Collected rainwater from the roof contains fine particles (dust, pollen, microalgae). Without proper filtration, the fan-shaped openings of the spray nozzles with a diameter below 1 mm would clog within a few days.

A Gardena Master Unit 1000 master unit is integrated at the beginning of the distribution network, performing two functions:

1. **Fine filtration:** It features a built-in stainless steel mesh with a density of $120\,\text{mesh}$, which retains all particles larger than $0.125\,\text{mm}$.
2. **Pressure reduction:** It dynamically reduces the inlet pressure to a constant 1.5 bar in the case of a stronger pump. This prevents hydraulic shocks during pump startup from causing the ejection of quick connectors on the $13\,\text{mm}$ diameter connecting pipe.

Figure 3.6: Gardena Master Unit 1000 pressure regulating unit

3.7 Alphanumeric 16x2 LCD Display with I2C Interface

To monitor key system parameters and interact with the system, an alphanumeric LCD display with a resolution of 16x2 characters is integrated. The display acts as a simple human-machine interface (HMI), providing the user in the field with an immediate overview of the current system status.

The main technical and functional characteristics of the display and communication module are:

- **Display capability:** It allows the display of alphanumeric characters in two lines (16 characters per line). It is based on the standard industrial Hitachi HD44780 controller.
- **I2C communication module:** The interface with the PCF8574 integrated circuit converts the parallel signal into the serial I^2C protocol, reducing the number of required control pins from 6 to just 2 signal lines.
- **Connection and addressing:** It connects to the hardware I^2C pins SDA (A4) and SCL (A5). Software control is performed via the `LiquidCrystal_I2C` library at address $0\text{x}27$ or $0\text{x}3F$.
- **Backlight and contrast:** It includes a software-adjustable LED backlight and a built-in potentiometer (trimmer) for manual adjustment of the display contrast.
- **Power supply and consumption:** It operates at a voltage of $5\,\text{V}$ directly from the Arduino. Current consumption with the backlight turned on is low (below $25\,\text{mA}$), putting minimal load on the system power supply.

Figure 3.7: 16x2 LCD display

3.8 System Power Supply (5 V USB Power Supply)

For stable and reliable operation of the microcontroller and all peripheral components (LCD display, capacitive sensors, RTC module, and the control part of the relay), a power supply via a mobile phone charging adapter is used in the system.

3.9 Component Table

Component	Arduino Pin	Signal Type
Capacitive moisture sensor 1	A0	Analog input ($0 - 5$ V)
Capacitive moisture sensor 2	A1	Analog input ($0 - 5$ V)
RTC DS1302 (RST)	D8	Digital output
RTC DS1302 (DAT)	D7	Digital input/output
RTC DS1302 (CLK)	D6	Digital output
Relay module	D10	Digital output
16x2 LCD display (SDA)	SDA	I^2C input/output
16x2 LCD display (SCL)	SCL	I^2C input/output
Setting button 1 (toggle)	D2	Digital input (interrupt)
Setting button 2 (up)	D3	Digital input
Setting button 3 (down)	D4	Digital input

Table 3.1: Pin layout and hardware connection configuration of the Arduino Uno microcontroller.

4. Software Algorithm and Measurement Calibration

4.1 Calibration and Processing of Analog Values

Since the capacitive sensor returns a raw voltage value via a 10-bit ADC converter (value range $0-1023$), a physical calibration was performed. The measurements provided the boundary conditions recorded in Table 4.1.

Soil condition	Sensor 1	Sensor 2	Calculated moisture (%)
Completely dry soil (in air)	460	454	0%
Saturated soil (mud after rain)	221	215	100%

Table 4.1: Calibration data of the analog-to-digital converter for the capacitive sensor v1.2.

In the code, the critical triggering threshold is set to a value of 40 %. Due to the occurrence of electromagnetic noise on longer signal cables (3 m length to the sensors), the algorithm does not use individual readings but instead performs "software smoothing using a moving average" from 10 consecutive samples at a 500 ms interval. Due to the size of the area, two sensors are used, the average of which represents the final current moisture value.

4.2 Control Program Logic

The program is designed as a finite state machine using a `switch - case` structure. It is divided into 3 main states, which have their own substates:

- States 0 and 1 (Standby and measurement): Initializes the LCD display output and periodically reads values from the two analog moisture sensors. The signals are filtered (average of 10 measurements) and scaled to a range of $0-99\,\%$. The algorithm tracks real time via the RTC module and, upon transitioning to a new day, calculates the running 7-day average moisture. If the current time matches the set watering time and the moisture is below the critical threshold ($40\,\%$), the system switches to the watering state.
- State 2 (Watering): Turns on the pump relay. The watering duration is monitored non-blocking using the `millis()` function. After the set interval expires, the pump turns off, the day of the last watering is recorded, and the system returns to standby.
- States 3, 4, and 5 (Settings menu): Turns off the pump and switches the LCD to configuration mode. The user can interactively change the watering start hour (0–23 h) and the pump operation duration (in steps of 30 seconds) using the "up" and "down" buttons.

Asynchronous management with a hardware interrupt (`TipkaPreklop`): The switching between the main screen and the configuration menus is performed via an external hardware interrupt on digital pin 2 (`attachInterrupt`). When button 1 is pressed, the software reacts instantly and cyclically changes the states of the state machine via the `cnt` counter. A software switch debounce protection with a time window of $200\,$ms is built into the function, preventing false multiple triggers.

The `zapisDneva` function: An auxiliary function that converts the numerical day value from the RTC module (0–6) into a string (e.g., "MON", "TUE") for display on the screen.

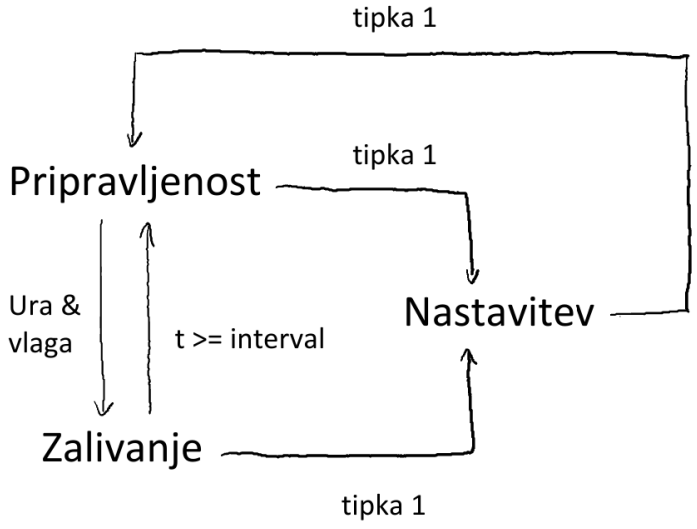

Figure 4.1: State transition diagram

4.3 Implemented Program Code in C++

The complete control code, ready for compilation in the Arduino IDE environment, is shown below:

```
#include <RtcDS1302.h>
#include <ThreeWire.h>
#include <LiquidCrystal_I2C.h>

int tipka_preklop = 2;
int tipka_nastavitev_gor = 3;
int tipka_nastavitev_dol = 4;
bool pritisnjeno = 0;
int rele = 10;
int senzor_vlage1 = A0;
int senzor_vlage2 = A1;

int cas_zalivanja = 5;
int interval_zalivanja = 60;

int vlaga = 0;
int vlaga_array[] = {0,0,0,0,0,0,0};
int povp_vlaga = 0;

volatile int stanje = 0;
volatile unsigned long zadnjiCasPrekinitve = 0;
const unsigned long casUmirjanja = 200;

unsigned long zacetek;
int dan;
String str_dan = "/";

LiquidCrystal_I2C lcd(0x27,16,2);
ThreeWire myWire(7,6,8);
RtcDS1302<ThreeWire> Rtc(myWire);

void setup() {

pinMode(tipka_preklop, INPUT);
pinMode(tipka_nastavitev_gor, INPUT);
pinMode(tipka_nastavitev_dol, INPUT);
pinMode(rele, OUTPUT);

attachInterrupt(digitalPinToInterrupt(tipka_preklop), TipkaPreklop, RISING);

Rtc.Begin();
//RtcDateTime currentTime = RtcDateTime(__DATE__ , __TIME__);
//Rtc.SetDateTime(currentTime);

lcd.init();
lcd.backlight();
```

```
RtcDateTime now = Rtc.GetDateTime();
dan = now.DayOfWeek();
}

void loop() {

RtcDateTime now = Rtc.GetDateTime();

switch(stanje){
  case 0: //standby state (display)
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Watered: ");
    lcd.print(str_dan);
    lcd.setCursor(0, 1);
    lcd.print("MOI:  % AVG:  %");

    stanje = 1;
    break;

  case 1: //standby state
    vlaga = 0;
    for(int i = 0; i < 10; i++){
      int val = analogRead(senzor_vlage1);
      val = constrain(val, 220, 455);
      val = map(val, 455, 220, 0, 99);
      vlaga += val;
      delay(50);
    }

    for(int i = 0; i < 10; i++){
      int val = analogRead(senzor_vlage2);
      val = constrain(val, 220, 455);
      val = map(val, 455, 220, 0, 99);
      vlaga += val;
      delay(50);
    }
    vlaga /= 20;

    if(now.DayOfWeek() != dan){
      static int n = 0;
      if(n == 7){
        for(int i = 0; i < 7; i++){
          vlaga_array[i] = 0;
        }
        n = 0;
      }

      vlaga_array[n] = vlaga;
      povp_vlaga = 0;
      for(int i = 0; i <= n; i++){
```

```
      povp_vlaga += vlaga_array[i];
    }
    povp_vlaga /= (n+1);
    n++;

    dan = now.DayOfWeek();
  }

  lcd.setCursor(4, 1);
  lcd.print("  ");
  lcd.setCursor(4, 1);
  lcd.print(vlaga);
  lcd.setCursor(12, 1);
  lcd.print("  ");
  lcd.setCursor(12, 1);
  lcd.print(povp_vlaga);

  if(now.Hour() == cas_zalivanja && vlaga < 40){
    digitalWrite(rele, HIGH);
    zacetek = millis();
    stanje = 2;
  }
  break;

case 2: //watering state
  if((millis() - zacetek) >= (interval_zalivanja * 1000)){
    digitalWrite(rele, LOW);
    str_dan = zapisDneva(dan);
    stanje = 0;
  }
  break;

case 3: //settings state (display)
  digitalWrite(rele, LOW);

  lcd.clear();
  lcd.setCursor(3,0);
  lcd.print("Watering");
  lcd.setCursor(1,1);
  lcd.print("at:");
  lcd.print(cas_zalivanja);
  lcd.print("h");
  lcd.setCursor(8,1);
  lcd.print("time:");
  lcd.print(interval_zalivanja);
  lcd.print("s");

  stanje = 4;
  break;

case 4: //settings state (watering time)
```

```
  if(!pritisnjeno && digitalRead(tipka_nastavitev_gor)){
    pritisnjeno = 1;
    cas_zalivanja++;
    if(cas_zalivanja > 23) cas_zalivanja = 0;

    lcd.setCursor(4,1);
    lcd.print("   ");
    lcd.setCursor(4,1);
    lcd.print(cas_zalivanja);
    lcd.print("h");
  }

  if(!pritisnjeno && digitalRead(tipka_nastavitev_dol)){
    pritisnjeno = 1;
    cas_zalivanja--;
    if(cas_zalivanja < 0) cas_zalivanja = 23;

    lcd.setCursor(4,1);
    lcd.print("   ");
    lcd.setCursor(4,1);
    lcd.print(cas_zalivanja);
    lcd.print("h");
  }

  if(!digitalRead(tipka_nastavitev_gor) && !digitalRead(tipka_nastavitev_dol))
  pritisnjeno = 0;

  break;

case 5: //settings state (watering interval)
  if(!pritisnjeno && digitalRead(tipka_nastavitev_gor) && interval_zalivanja < 990){
    pritisnjeno = 1;
    interval_zalivanja += 30;

    lcd.setCursor(12,1);
    lcd.print("    ");
    lcd.setCursor(12,1);
    lcd.print(interval_zalivanja);
    lcd.print("s");
  }

  if(!pritisnjeno && digitalRead(tipka_nastavitev_dol) && interval_zalivanja > 0){
    pritisnjeno = 1;
    interval_zalivanja -= 30;

    lcd.setCursor(12,1);
    lcd.print("    ");
    lcd.setCursor(12,1);
    lcd.print(interval_zalivanja);
    lcd.print("s");
  }
```

```
    if(!digitalRead(tipka_nastavitev_gor) && !digitalRead(tipka_nastavitev_dol))
    pritisnjeno = 0;

    break;

  default: break;
}

delay(100);
}


void TipkaPreklop(){
  static int cnt = 0;
  unsigned long trenutenCas = millis();

  if (trenutenCas - zadnjiCasPrekinitve > casUmirjanja) { //due to Switch Bounce effect
    if(cnt == 0){
      stanje = 3;
      cnt++;
    }
    else if(cnt == 1){
      stanje = 5;
      cnt++;
    }
    else if(cnt == 2){
      stanje = 0;
      cnt = 0;
    }

  zadnjiCasPrekinitve = trenutenCas;
  }
}

String zapisDneva(int stevilo){
  switch(stevilo){
    case 0: return "SUN";
    case 1: return "MON";
    case 2: return "TUE";
    case 3: return "WED";
    case 4: return "THU";
    case 5: return "FRI";
    case 6: return "SAT";
    default: return "err";
  }
}
```

5. Conclusion

Within the scope of this project work, a working prototype of an automated garden irrigation system based on the Arduino Uno platform was successfully realized. The developed mechatronic system fully meets the set goals, as it enables completely autonomous and sustainable management of a 24 m^2 area using exclusively harvested rainwater from an IBC storage tank.

High operational reliability was achieved through the integration of capacitive moisture sensors, a fine mesh filter, and software smoothing of measurements, which eliminated issues with corrosion, noise, and nozzle clogging. The control algorithm, designed as a finite state machine with non-blocking time loops (`millis()`), ensures deterministic task execution. Additionally, the introduction of an external hardware interrupt on pin D2 and an I^{2}C LCD display allows the user to easily and instantly adjust watering parameters directly in the field.

As a meaningful upgrade to the system in the future, I foresee the integration of a Wi-Fi module (e.g., ESP32) for IoT network connectivity, which would enable remote control and system monitoring via a mobile application, as well as adjusting irrigation based on online weather forecasts.

Figure 5.1: Electronic enclosure of the product