

***Modern University for Technology & Information***

**Faculty Of Engineering**

**Control Engineering Department**

***SIMACAS***

**A premium project submitted to Control Engineering Department in partial fulfilment of the requirements for the degree of Bachelor of Science in Control Engineering**

**SUPERVISED BY:**

DR. MOHAMED FAWZY ABD EL HAMED

**PROVIDED BY:**

**AHMED MOHAMED SOBHY**

**ABD ELRAHMAN ALHAMMADY**

**MOSTAFA AHMED ELSHEIKH**

**ACKNOWLEDGEMENT**

Table of Contents

[Executive Summary 5](#_Toc168340442)

[Chapter 1: Introduction 6](#_Toc168340443)

[Project Aim 6](#_Toc168340444)

[Summarized Methodology 8](#_Toc168340445)

[Quick Walkthrough 10](#_Toc168340446)

[Chapter 2: Literature Review 11](#_Toc168340447)

[Chapter 3: Methodology 20](#_Toc168340448)

[System Overview 20](#_Toc168340449)

[Mechanical Design 23](#_Toc168340450)

[Intro 23](#_Toc168340451)

[Aluminum Frame 23](#_Toc168340452)

[Profile Traversal Mechanism 23](#_Toc168340453)

[Water Distribution System 25](#_Toc168340454)

[Electronics 26](#_Toc168340455)

[Intro 26](#_Toc168340456)

[Axis Control 26](#_Toc168340457)

[Water Distribution Circuit 26](#_Toc168340458)

[Sensor Monitoring 26](#_Toc168340459)

[Programming 27](#_Toc168340460)

[Intro 27](#_Toc168340461)

[Axis Control 27](#_Toc168340462)

[Sensor Monitoring 27](#_Toc168340463)

[Main Board 27](#_Toc168340464)

[Chapter 4: Conclusion 28](#_Toc168340465)

[References 29](#_Toc168340466)

[Figure 1: Aluminum Profile Frame 3D model 18](#_Toc166723775)

# Executive Summary

# Chapter 1: Introduction

## Project Aim

Irrigation stands as a cornerstone of human civilization, pivotal to its survival and growth throughout history. From ancient times to modern days, the evolution of irrigation has mirrored the progress of civilization itself. Particularly since the Industrial Revolution, and more recently the Data Revolution, advancements in technology have transformed traditional practices. Modern machinery and automation have streamlined many processes that were once labor-intensive, achieving levels of efficiency and precision that were previously unattainable. Despite these advancements, the field of irrigation continues to offer opportunities for innovation, especially in specialized applications.

This project targets one such niche: the development of a small-scale, temporary, automated irrigation system designed for highly controlled environments. These systems are particularly beneficial for specialized agricultural research or the cultivation of rare and exotic plants under precise conditions. The primary goal of this project is to engineer a sophisticated system capable of meticulously monitoring and controlling every aspect of a plant’s environment.

Key features of this innovative system include:

* Comprehensive Monitoring: Utilizing an array of sensors, the system will continuously monitor critical parameters such as temperature, humidity, water levels, and CO2 concentrations. This data-rich approach ensures that all environmental factors affecting plant growth are observed and analyzed.
* Precise Irrigation and Fertilization: The core of the system’s functionality lies in its ability to dispense water and nutrients with pinpoint accuracy. Through the integration of advanced robotics, both the quantity and timing of water and fertilizer application can be precisely controlled, tailored to the plant's specific needs at any growth stage.
* Intelligent Control System: At the heart of the operation is a main Microcontroller Unit (MCU). This MCU is tasked with interpreting sensor data and managing output actions. It is programmed with a sophisticated 'model predictive control' algorithm, which optimizes conditions for maximum plant growth and yield.
* Adaptive Algorithms: The system’s adaptive capabilities allow for real-time adjustments based on immediate environmental feedback, enhancing the efficacy and responsiveness of the irrigation strategy.

By focusing on these advanced technological solutions, this project aims to push the boundaries of what is possible in controlled agricultural systems. It promises not only to enhance the precision of plant cultivation but also to contribute valuable insights into the optimal conditions for plant growth, offering potential applications in both research settings and rare plant cultivation.

Central to achieving this precision is the integration of a model predictive control (MPC) algorithm within the system's main Microcontroller Unit (MCU). The MPC algorithm is crucial for several reasons:

* Anticipatory Adjustments: Unlike conventional control systems that react to changes, MPC predicts future conditions and makes proactive adjustments. This predictive capability allows for the optimization of irrigation and fertilization schedules based on forecasted environmental and plant needs.
* Optimal Resource Utilization: By accurately forecasting future states, the MPC algorithm ensures optimal use of water and nutrients, which is essential for conserving resources while maximizing plant health and productivity.
* Enhanced Growth and Yield: The algorithm continuously refines its predictions and control outputs, adapting to the plant’s growth stages and environmental fluctuations. This adaptability is key to enhancing plant growth rates and maximizing yield.
* Precision and Control: The system's ability to precisely control the delivery of water and nutrients directly addresses the specific needs of each plant, tailored to its particular growth conditions. This level of control is particularly advantageous in research settings or for cultivating plants that require specific care.

The proposed irrigation system not only focuses on the practical aspects of plant cultivation but also embodies a cutting-edge approach to agricultural technology. Through the application of the MPC algorithm, this project aims to set new standards in the precision farming industry, providing insights that could influence broader agricultural practices.

By leveraging such advanced control strategies, this project not only aims to enhance the efficiency and effectiveness of irrigation practices but also to contribute to the broader field of agricultural science, offering scalable solutions that could be adapted for various controlled-environment agriculture applications.

## Summarized Methodology

To achieve an efficient and controlled environment for plant development, a novel irrigation system was designed and developed. The system integrates advanced automation and precision control technologies, aimed at optimizing plant growth conditions.

Structural Framework:

* Expandable Aluminum Frame: A robust and expandable aluminum frame is suspended above the plants. This frame provides a durable and stable platform for the robotic components.
* V-Wheel Ball Bearings and Stepper Motors**:** Small robots traverse the aluminum frame using V-wheel ball bearings driven by stepper motors, ensuring smooth and precise movement.

Robotic Components:

* 3D Printed Robot Frame**:** The robot frame is 3D printed using PLA+ material, which is known for its high reliability and durability in outdoor environments. This material choice also lowers manufacturing costs and allows for complex geometries that would be difficult to achieve with traditional manufacturing methods.
* Dual Sprinkler System**:** Each robot is equipped with two 3D printed sprinklers, connected to separate tubes—one for water and the other for low viscosity fertilizer and essential chemicals. The sprinklers are designed for easy exchange and redesign, enhancing the system's adaptability. Various sprinkler designs cater to different irrigation needs, such as far radial dispersal or direct downward dispersal, increasing the versatility of the system.

Control Circuitry:

* Custom PCB and Microcontroller**:** A specialized circuit was designed, and a PCB was fabricated to ensure maximal reliability and efficiency. The circuit includes a powerful microcontroller, a radio transceiver, an ultrasonic sensor, stepper motor drivers, and other complementary components.
* Wireless Communication and Closed-Loop Control**:** The microcontroller receives commands wirelessly from the main microcontroller running the Model Predictive Control (MPC) algorithm. It then executes these commands by signaling the stepper motor drivers. The ultrasonic sensor on the robot frame measures the distance from the origin point, enabling closed-loop control. This allows for precise movement adjustments based on real-time measurements.

Fluid Control Network:

* Solenoid Valves**:** A network of solenoid valves precisely controls the flow of liquids from the water and fertilizer tanks to the desired robot and sprinkler. These valves are monitored and controlled by the main microcontroller.

Main Control System:

* Main Microcontroller Board**:** The main microcontroller wirelessly manages the traversing robots and solenoid valves. It is also connected to a myriad of sensors monitoring various plant conditions.
* MPC Algorithm Implementation**:** The MPC algorithm on the main microcontroller analyzes sensor data, performs intricate calculations, and determines the optimal control actions. This ensures precise irrigation and nutrient delivery tailored to the plants' needs, maximizing growth and yield.

This integrated system leverages modern manufacturing techniques, advanced control algorithms, and real-time monitoring to create an optimized environment for plant growth.

## Quick Walkthrough

Steps to use the project and quick results

# Chapter 2: Literature Review

[1] The practise of farming has endured significant transformation as technology advances every day. The constraints of area and nonlinear nature of climatic conditions, polyhouse kind of concepts are increasing, which is helpful in production of flowers, vegetables and fruits. The proposed work discusses such an automated irrigation system that highlights the optimum solution for the efficient use of water and electricity for agricultural purposes. Field survey and literature shows that the existing systems are available with two solutions, one is timer-based and another one is moisture-based automization. Moreover, the timer-based system has demerits like being semi-automated i.e., timer needs to be changed manually according to climate. Similarly, in moisture-based systems, reliability is the issue. Therefore, the main objectives of the proposed work are to overcome the demerits of the present systems by integrating both the systems, to develop a fully automated irrigation system, to manage the use of water, electricity, and to add a remote controlling system. The report includes algorithm for the integration of moisture and timer-based system which provides the optimum efficiency on the water use and the use of solenoidal valve.

[2] The article "Improving water-efficient irrigation: Prospects and difficulties of innovative practices" by Levidow et al. (2014) discusses the challenges and potential of innovative irrigation practices aimed at enhancing water efficiency. The authors emphasize that while technological advancements in irrigation can offer significant benefits, such as economic advantages and reduced environmental impacts, there are several barriers to their effective implementation.

Key challenges include the lack of adequate knowledge among farmers regarding water usage, irrigation applications, and crop yield responses to different water management practices. The article highlights the importance of a knowledge-exchange system that involves all stakeholders, including farmers, extension services, and water management organizations, to promote better water management practices.

The study presents two case studies from Europe that illustrate these challenges and opportunities. In both cases, despite the availability of advanced irrigation technologies, farmers faced difficulties in achieving optimal water efficiency due to inadequate knowledge and support systems. The authors suggest that a continuous knowledge-exchange system is necessary to ensure that stakeholders can share responsibilities and achieve greater water efficiency across the entire water-supply chain.

[3] The paper "Model Predictive Control for Closed-Loop Irrigation" by Lozoya et al. (2014) explores the use of model predictive control (MPC) in irrigation systems to enhance water efficiency. The primary objective is to optimize the amount of water used by accurately controlling soil moisture levels based on real-time climatic factors and soil conditions. The authors present a detailed model that incorporates soil moisture control and climatic variables, demonstrating that MPC can significantly improve water use efficiency compared to traditional time-based and basic soil moisture control irrigation systems.

Key highlights of the paper include:

Hydrological Balance Model: The paper outlines a model based on the hydrological balance that describes water storage changes in relation to irrigation, rainfall, evapotranspiration, and other factors.

Evapotranspiration and Soil Moisture: It discusses the critical role of evapotranspiration and soil moisture in irrigation control, explaining how these factors influence water demand and availability.

MPC Implementation: The authors propose an MPC strategy that predicts future irrigation needs and adjusts water application accordingly, minimizing water consumption while maintaining optimal soil moisture for crop health.

Simulation and Validation: Through simulations, the MPC-based system is compared with traditional irrigation methods, showing superior performance in reducing water usage and maintaining soil moisture levels.

Practical Considerations: The paper addresses the computational demands of MPC and suggests that advancements in embedded systems make the implementation feasible for large-scale agricultural applications.

The study concludes that MPC offers a promising approach to improving irrigation efficiency, which is crucial for sustainable water management in agriculture.

[4] The paper "Model Predictive Control for Real-Time Irrigation Scheduling" by Khusro Saleem et al. (2013) discusses the application of Model Predictive Control (MPC) to optimize irrigation scheduling, thereby improving agricultural productivity by better matching water supply to crop demand. The authors highlight that traditional irrigation scheduling often relies on heuristic rules, which can lead to over-watering, wasting water, and reducing crop yields.

Key points from the paper include:

MPC Framework: The authors propose an MPC framework that incorporates real-time data, such as soil moisture and climatic conditions, to make more accurate irrigation decisions. The MPC algorithm optimizes watering schedules within operational constraints like water availability and maximum or minimum water application limits.

Water Balance Model: The foundation of the proposed system is a water balance model that accounts for various water flows into and out of the soil column, including irrigation, precipitation, evapotranspiration, runoff, and deep percolation. This model helps in predicting the soil moisture deficit and planning irrigation events accordingly.

Optimization and Control: The paper details how the MPC controller uses predictive models to minimize deviations from desired soil moisture levels, ensuring efficient water use. The approach accounts for uncertainties in climate inputs and operational constraints, providing a systematic method for real-time irrigation scheduling.

Simulation and Results: Using measured evapotranspiration and precipitation data, the authors demonstrate that the MPC approach leads to improved set-point tracking and adherence to constraints compared to traditional heuristic methods. The results show that MPC can significantly reduce over-watering and under-watering issues.

[5] The paper "An Overview of Smart Irrigation Management for Improving Water Productivity under Climate Change in Drylands" by Ahmed et al. (2023) reviews various smart irrigation techniques aimed at enhancing water use efficiency (WUE) in dryland regions, which face significant challenges due to water scarcity and climate change. The authors discuss the limitations of traditional irrigation scheduling methods, which often lead to over- or under-irrigation, and highlight the potential of smart irrigation systems to address these issues.

Key points from the paper include:

Smart Irrigation Systems: These systems use a combination of soil, climate, and plant data to optimize irrigation schedules. Technologies such as artificial intelligence (AI), deep learning, variable rate irrigation (VRI), and unmanned aerial vehicles (UAVs) are integral to these systems, enabling precise water application based on real-time data.

Benefits of Smart Irrigation: Smart irrigation systems can significantly improve WUE by ensuring that water is applied in the right amounts, at the right times, and in the right places. This leads to increased crop yields, reduced water usage, and better resource management.

Model Predictive Control (MPC): MPC is highlighted as a key technology in smart irrigation, capable of managing irrigation schedules based on predictive models that consider future climatic and soil conditions. This approach helps in maintaining optimal soil moisture levels, thereby enhancing crop growth and productivity.

Challenges and Recommendations: Despite the advantages, the implementation of smart irrigation systems faces challenges such as high costs, complexity, and the need for extensive training for farmers. The authors recommend further research and development to make these technologies more affordable and user-friendly, as well as greater support from governments and institutions to promote widespread adoption.

Case Studies and Applications: The paper provides examples of successful smart irrigation implementations in various regions, demonstrating the effectiveness of these technologies in improving agricultural productivity and sustainability in drylands.

[6] The paper "Computation of Control Law for State Transfer Problem in Efficient Way for a Single Input" by Rajani Metri and Bhooshan Rajpathak (2022) presents a novel method for solving the state transfer problem (STP) in linear time-invariant (LTI) systems with a single input. The proposed method focuses on reducing the computational complexity typically associated with solving the STP by eliminating the need to compute the controllability Gramian matrix and the state transition matrix, which are traditionally required in conventional methods.

Key points from the paper include:

Control Law Computation: The authors introduce an efficient technique for computing the control law for STP without calculating the controllability Gramian and state transition matrix. This method relies on direct computations involving mode vectors and simplifies the process significantly.

Mathematical Efficiency: The proposed method minimizes the use of complex mathematical operations such as matrix inverses and determinant calculations, which are often computationally intensive and prone to rounding errors.

Special Cases Handling: The method is particularly advantageous for systems with real and distinct roots, repeated roots, and complex eigenvalues, demonstrating flexibility across various types of LTI systems.

Theoretical and Experimental Validation: The paper includes theoretical proofs and experimental results to validate the proposed method. The experimental setup involves a Hardware-In-Loop (HIL) system to demonstrate the practical applicability of the control law in an industrial context.

Advantages: The method provides a computationally efficient way to achieve state transfer in finite time with minimum control energy, making it suitable for real-time applications in industrial automation, electric drives, and other control systems.

[7] The paper "Design and Evaluation of Mobile Drip Irrigation System" by Khairy et al. (2016) explores the development and performance assessment of a Mobile Drip Irrigation System (MDIS). This system combines the efficiency of surface drip irrigation with the flexibility and economic benefits of center pivot and lateral move irrigation systems. The research emphasizes the system's ability to improve application efficiency through precise water and nutrient delivery, reducing water losses due to wind drift and evaporation.

Key points from the paper include:

MDIS Design: The MDIS utilizes classic drip irrigation materials, integrating on-line drip hoses with center pivot or lateral move systems. The design aims to optimize wetting patterns and ensure efficient water distribution.

Efficiency and Performance: The study found that the MDIS achieved application efficiencies higher than 82%. The system's efficiency is attributed to the slow, methodical release of water directly to the soil, promoting optimal plant growth.

System Components and Setup: The MDIS comprises several components, including driven wheels, a water supply pipe, drip tubes, and pressure compensating drippers. The system's speed and the distance between drip tubes were varied to assess performance under different conditions.

Wetting Front Advance: The research measured the wetting front advance in horizontal, vertical, and diagonal directions in loamy sand soil. Results showed that the wetting front advance increased with higher discharge rates and system speeds, indicating effective water infiltration and distribution.

Calibration and Evaluation: The MDIS was calibrated at different operating pressures and evaluated for parameters such as the coefficient of variation, emission uniformity, and flow variation. The study confirmed that the drip tubes used were fully pressure compensating and performed well under various pressures.

Application in Drylands: The paper suggests that the MDIS technology can significantly improve irrigation efficiency in dryland regions, where water conservation is crucial. The system's ability to maintain dry wheel tracks and reduce soil compaction is particularly beneficial for these areas.

[8] The paper "Irrigation Control Based on Model Predictive Control (MPC): Formulation of Theory and Validation Using Weather Forecast Data and AQUACROP Model" by Delgoda et al. (2016) presents a robust model predictive control (MPC) framework for optimizing irrigation schedules. The main goal is to minimize both root zone soil moisture deficit (RZSMD) and irrigation amount, especially under limited water supply conditions. The authors investigate the integration of direct measurements into the MPC system and introduce two robust MPC techniques—Certainty Equivalence control (CE) and Disturbance Affine Feedback control (DA)—to handle uncertainties in weather forecasts.

Key points from the paper include:

MPC Framework: The proposed framework aims to maintain optimal soil moisture levels by adjusting irrigation amounts based on predictive models that account for future weather conditions and soil moisture data. This approach is proactive, unlike traditional methods that react to soil moisture deficits after they occur.

Robust MPC Techniques:

Certainty Equivalence Control (CE): Assumes perfect knowledge of future disturbances and uses deterministic forecasts to optimize irrigation.

Disturbance Affine Feedback Control (DA): Accounts for uncertainty in weather forecasts by using a set of possible disturbances, leading to more conservative but reliable irrigation schedules.

System Model: The study uses a simplified water balance model to represent the actual physical system, focusing on the main dynamics affecting soil moisture. The model is validated using AQUACROP simulations and weather data from Shepparton, Victoria, Australia.

Feasibility and Stability: The paper discusses the importance of ensuring feasibility and stability (Input to State Stability, ISS) in MPC-based irrigation systems. The authors propose removing state constraints to enhance feasibility under water-limited conditions and demonstrate the system's ability to recover from extreme rainfall events.

[9] The paper "Smart Irrigation Systems: Overview" by Gamal et al. (2023) provides a comprehensive review of smart irrigation systems aimed at enhancing water efficiency in agriculture. The authors discuss the integration of advanced technologies such as wireless communication systems, Internet of Things (IoT), smart sensing, and energy harvesting to improve irrigation scheduling and overall water management.

Key points from the paper include:

Smart Irrigation Technologies: The paper highlights the critical role of smart technologies in irrigation, including IoT devices for real-time monitoring, AI for predictive analytics, and wireless sensor networks (WSNs) for data collection and communication. These technologies facilitate precise control over water usage, optimizing irrigation schedules based on real-time data.

Components of Smart Irrigation:

Real-time Irrigation Scheduling: Using real-time data from soil and climate sensors to adjust irrigation schedules dynamically.

IoT and Wireless Communication: Leveraging IoT devices and WSNs to collect and transmit data efficiently, enabling remote monitoring and control of irrigation systems.

Smart Sensing: Implementing advanced sensors to monitor soil moisture, weather conditions, and plant health, ensuring accurate data collection for better decision-making.

Energy Harvesting: Utilizing renewable energy sources, such as solar panels, to power irrigation systems, making them more sustainable and reducing operational costs.

Challenges and Opportunities: The paper discusses various challenges in implementing smart irrigation systems, including high costs, complexity, and the need for extensive training for farmers. It also explores opportunities for improving water use efficiency and agricultural productivity through smart irrigation.

[10] The paper "Automatic Irrigation Systems for Efficient Usage of Water using Embedded Control Systems" by Vijendra Babu et al. (2020) explores the development and implementation of an automatic irrigation system aimed at optimizing water usage in agriculture through embedded control systems. The paper addresses the need for efficient water management in the context of increasing water scarcity and the rising demand for food production.

Key points from the paper include:

Automation and Embedded Control Systems: The authors propose the use of embedded control systems to automate irrigation, replacing the traditional manual methods. The system utilizes various sensors to monitor environmental parameters such as soil moisture, humidity, and temperature, which inform the irrigation process.

System Design: The proposed system includes an Arduino Uno development board that interfaces with multiple sensors (ultrasonic, light ambient, humidity, temperature, and soil moisture sensors). These sensors provide real-time data to control irrigation equipment such as solenoid valves and water sprinklers, ensuring precise water application.

Operational Efficiency: The automatic irrigation system is designed to operate based on real-time soil moisture levels, turning on the water supply only when necessary. This approach prevents over-watering, reduces water waste, and enhances crop yield and quality.

Prototype Implementation: A prototype of the system was developed and tested. The results demonstrated effective utilization of water resources and significant labor savings. The system also helped in preventing soil erosion and nutrient runoff by avoiding excessive water flow.

Scalability: The paper discusses the scalability of the proposed system for larger fields by increasing the number of sensors and control units. This adaptability makes it suitable for various agricultural settings.

# Chapter 3: Methodology

## System Overview

The objectives of this project are varied and highly specific, necessitating a comprehensive and detailed methodology. This project's design prioritizes versatility, maintainability, and, most importantly, the ability for the entire system to be installed and removed quickly and easily with minimal equipment and expertise. This feature offers exceptional advantages for businesses seeking temporary irrigation solutions, both on large and small scales, and for individuals needing to set up a complete irrigation system rapidly.

One of the primary objectives is to achieve extreme control over the environment where the plants are grown. This includes precise regulation of light color and intensity, water and fertilizer flow rates, and the timing of irrigation. Additionally, an extensive array of sensors monitors every conceivable aspect of the plants' conditions. The sensor data is fed into a specially developed Model Predictive Control (MPC) algorithm, which automates all outputs to optimize yield, plant quality, or any other user-specified goals.

Having outlined the high-level objectives, we now turn to the implementation of the solutions.

To achieve a highly maintainable and easily installed irrigation system, we had to deviate from the common practice of burying water tubes underground. Although this method is cost-effective in the long run, it is labor-intensive to install due to the required digging, and even more so to remove, as it can potentially disrupt the soil layer where the plants grow. This makes it unsuitable for a system designed for easy installation and removal. Our solution involves the construction of a framework of aluminum profiles, where upright rods support an array of elevated aluminum profile rods. These profiles act as rails along which a specially designed robot traverses using stepper motors and V-wheel-shaped bearings that grip the aluminum profile grooves, allowing it to cover the entire length of the system. This design minimizes ground disturbance as no digging or tube laying is required. The actual sprinklers irrigate the plants from above, traveling across the fields on the elevated rails, reaching any desired location.

To achieve the extraordinary level of control promised by this system, we meticulously designed a specialized robot capable of precisely traversing the elevated aluminum profiles to position itself exactly over the areas needing irrigation. The robot is equipped with two types of sprinklers: one suitable for wide-area irrigation and another for targeting smaller areas with a more concentrated water flow. The robot frame and sprinklers are made from 3D-printable PLA+ material, which provides ample tensile strength and the numerous advantages of 3D printing, such as the ability to create complex geometries that would be extremely difficult to produce using traditional subtractive methods.

The installation process begins with setting up the aluminum profile structure. Upright rods are anchored into the ground, supporting horizontal aluminum profiles that form the rail system. This design ensures that the system is both sturdy and easily dismantled. The robot, designed for seamless integration with these profiles, can be swiftly installed and calibrated. Once set up, the robot moves along the rails, providing precise irrigation based on sensor feedback and the MPC algorithm's directives.

To ensure maintainability, all components are designed for easy access and replacement. The use of 3D-printed parts allows for quick manufacturing of custom components, ensuring that any damaged or worn parts can be rapidly replaced. This modularity extends to the sensors and control units, which are mounted in accessible locations and connected via quick-release fittings.

The extraordinary control over the irrigation environment is achieved through an array of sensors that monitor soil moisture, ambient temperature, humidity, light intensity, and other critical parameters. These sensors provide real-time data to the MPC algorithm, which processes the information and adjusts the irrigation parameters accordingly. This level of control ensures optimal growing conditions, enhancing plant health and yield.

In summary, this project's methodology centers on creating a highly versatile, maintainable, and easily installable irrigation system. By leveraging advanced robotics, sensor technology, and innovative design, the project achieves exceptional control over plant growth environments while ensuring ease of use and flexibility for various applications.

## Mechanical Design

### Intro

The mechanical design of this project is a critical component, integral to realizing the project's objectives. One of the primary challenges in the mechanical aspect is ensuring ease of installation and removal, a feature essential for the system's practicality and versatility in various research settings.

#### Challenges and Innovative Solutions:

1. **Ease of Installation and Removal:**
   * **Challenge:** Designing a system that can be easily set up and dismantled without requiring extensive labor or technical expertise.
   * **Innovative Solution:** To address this, we decided to relocate the water sprinklers from an underground setup to a hanging configuration above the plants. This approach eliminates the complexities associated with underground installations and allows for a more straightforward and rapid deployment.
2. **Development of a Hanging Structure:**
   * **Challenge:** Creating a structure that is rigid, reliable, cost-effective, and easy to assemble and disassemble.
   * **Innovative Solution:** Aluminum profiles were chosen for constructing the hanging structure. These profiles are renowned for their ease of installation, as they can be quickly bolted together and unbolted for removal. Aluminum’s inherent properties ensure consistent rigidity and exceptional straightness, unlike steel bars, which can sometimes be bent or curved. This makes aluminum an ideal material for maintaining structural integrity and ensuring precise alignment of components.

By leveraging aluminum profiles, the system benefits from:

* **Ease of Handling:** Lightweight yet robust, aluminum profiles can be managed and maneuvered without heavy machinery, facilitating quicker setup.
* **Cost-Effectiveness:** Aluminum is relatively inexpensive compared to other metals, making it an economical choice for large-scale or repeated installations.
* **Durability and Reliability:** Resistant to corrosion and maintaining structural integrity over time, aluminum ensures a long-lasting framework for the irrigation system.

The specific details of the aluminum frame structure and its assembly will be discussed in the subsequent sections.

Building upon the foundation of aluminum profiles laid above the plants, another significant mechanical challenge arises with the precise and efficient delivery of water and fertilizer. Directly hanging sprinklers every 10 to 30 cm would not only be cost-inefficient but also compromise the accuracy of nutrient delivery. To address this, we developed a unique aluminum profile traversal robot.

The primary goal of this traversal robot is to ensure precise movement along the aluminum profiles using high-precision stepper motors. Equipped with custom electronics, the robot can communicate wirelessly, enhancing its operational flexibility. A dedicated battery powers the robot, ensuring uninterrupted performance over extended periods.

#### Design and Functionality of the Traversal Robot

1. **Accurate Traversal:** The robot is engineered to traverse the length of the aluminum profiles with exceptional precision, facilitated by precision stepper motors. These motors enable the robot to position itself accurately along the profile, ensuring that water and fertilizer are dispensed exactly where needed.
2. **Custom Electronics and Wireless Communication:** The onboard electronics are custom-designed to manage the robot's operations and ensure seamless wireless communication with the central control unit. This wireless capability allows for real-time adjustments and monitoring, optimizing the irrigation and fertilization processes.
3. **Efficient Power Management:** The robot is powered by a rechargeable battery, ensuring that it can operate autonomously without constant need for external power sources. This design choice enhances the system's flexibility and ease of use, especially in remote or large-scale setups.
4. **Cost Efficiency and Precision:** Instead of deploying multiple sprinklers along the entire length of each aluminum profile, the traversal robot carries only two sprinklers. As the robot moves, it positions these sprinklers precisely, reducing the number of sprinklers needed and thereby significantly cutting costs. This approach not only economizes on the number of sprinklers but also ensures that water and fertilizer are delivered with high accuracy, enhancing the effectiveness of the irrigation system.
5. **Operational Advantages:** Each aluminum profile line, regardless of its length, is traversed by a single robot. This configuration means that the cost of sprinklers is no longer directly proportional to the length of the profile, leading to substantial cost savings. Additionally, the precision and repeatability of the robot’s movements ensure that the water and fertilizer are applied exactly where required, improving resource efficiency and plant health.

In Summary, the aluminum profile traversal robot represents a significant innovation in the design of automated irrigation systems. By reducing the number of sprinklers needed and enhancing the precision of water and nutrient delivery, the robot not only lowers costs but also increases the efficiency and effectiveness of the system. This innovative approach ensures that resources are used optimally, making the system both economically and environmentally sustainable.

In the subsequent sections, we will delve into the detailed design, assembly, and operational testing of the traversal robot, demonstrating its critical role in achieving the project's objectives.

The final challenge in the mechanical design of the automated irrigation system was developing an efficient water distribution system. As previously mentioned, each profile traversal robot is equipped with two sprinklers: one for water and the other for fertilizer. The water distribution system's primary task is to deliver these fluids to the appropriate robot at the correct time.

**Challenges and Innovative Solutions:**

1. **Electronic Control of Water and Fertilizer Flow:**
   * **Challenge:** Precisely controlling the flow of water and fertilizer to each robot without manual intervention.
   * **Solution:** The implementation of solenoid valves, powered by relay boards, addresses this challenge effectively. Solenoid valves act as electronic switches for liquids, allowing precise control over the flow of water and fertilizer. These valves are operated by microcontroller signals, which toggle the valve between open (HIGH) and closed (LOW) states.

Each robot in the system is connected to two solenoid valves: one for water and one for fertilizer. This configuration allows for accurate, remote-controlled dispensation of water and fertilizer, ensuring that each robot receives the necessary fluids at the precise moment needed.

1. **Connecting Multiple Solenoid Valves to a Single Source:**
   * **Challenge:** Efficiently connecting multiple solenoid valves to a single water or fertilizer tank.
   * **Solution:** To solve this, a custom-designed 3D-printed part was developed. This specialized component features a single inlet connected to the water or fertilizer tank and multiple outlets corresponding to the number of robots in the system.

This part acts as a manifold, distributing the fluid from one central tank to all the connected robots. For example, the water tank connects to one side of the manifold, while the other side has several outlets, each leading to a solenoid valve on a different robot. The same design is used for the fertilizer tank.

#### The Key Features of the Water Distribution System:

1. **Precision Control:**
   * Solenoid valves enable precise control over the flow of water and fertilizer. By using microcontroller signals to operate the valves, the system can manage the timing and quantity of fluid dispensation accurately.
2. **Remote Operation:**
   * The entire process is automated and controlled remotely, eliminating the need for human intervention. This remote control capability enhances the system's efficiency and reliability.
3. **Custom Manifold Design:**
   * The 3D-printed manifold is a critical innovation that simplifies the connection between the tanks and multiple robots. This part ensures that fluids are distributed evenly and reliably to each robot, regardless of the system's scale.
4. **Scalability:**
   * The custom manifold design allows for scalability. By adding more outlets, the system can easily accommodate additional robots, making it adaptable to various research and cultivation needs.
5. **Cost-Effectiveness:**
   * By utilizing solenoid valves and custom manifolds, the system minimizes the number of components required for fluid distribution. This not only reduces costs but also simplifies maintenance and increases the overall reliability of the system.

In Summary, the water distribution system is a crucial component of the automated irrigation system, ensuring that water and fertilizer are delivered precisely and efficiently. Through the innovative use of solenoid valves and custom 3D-printed manifolds, the system achieves high precision and reliability while remaining cost-effective and scalable. These advancements contribute significantly to the system's ability to provide optimal growing conditions for plants, thereby enhancing both research and cultivation outcomes.

In the following sections, we will explore the detailed design, implementation, and testing of the water distribution system, demonstrating its effectiveness in supporting the project's overall objectives.

### Aluminum Frame

The aluminum profile frame is a critical component of the automated irrigation system, providing essential support and stability. The frame is constructed entirely from 2040 profiles, which have a cross-section of 20mm x 40mm. This specific dimension was selected based on mechanical stress calculations, confirming its suitability for the application's requirements.

The design of the aluminum profile frame involves several key elements. The frame consists of multiple parallel lengths of 2040 profiles, which form the primary horizontal structure. These parallel profiles are connected at either end by longer profiles, ensuring the frame acts as a cohesive unit and providing stability.

Along the length of the first and last parallel profiles, vertical profiles of 1.5 meters are placed at 2-meter intervals. These vertical profiles serve as pillars, supporting the entire structure and maintaining its integrity under operational loads. This arrangement satisfies both the strength and functional requirements of the system.

To provide a clearer understanding of the aluminum profile frame, the following visual aids are included:

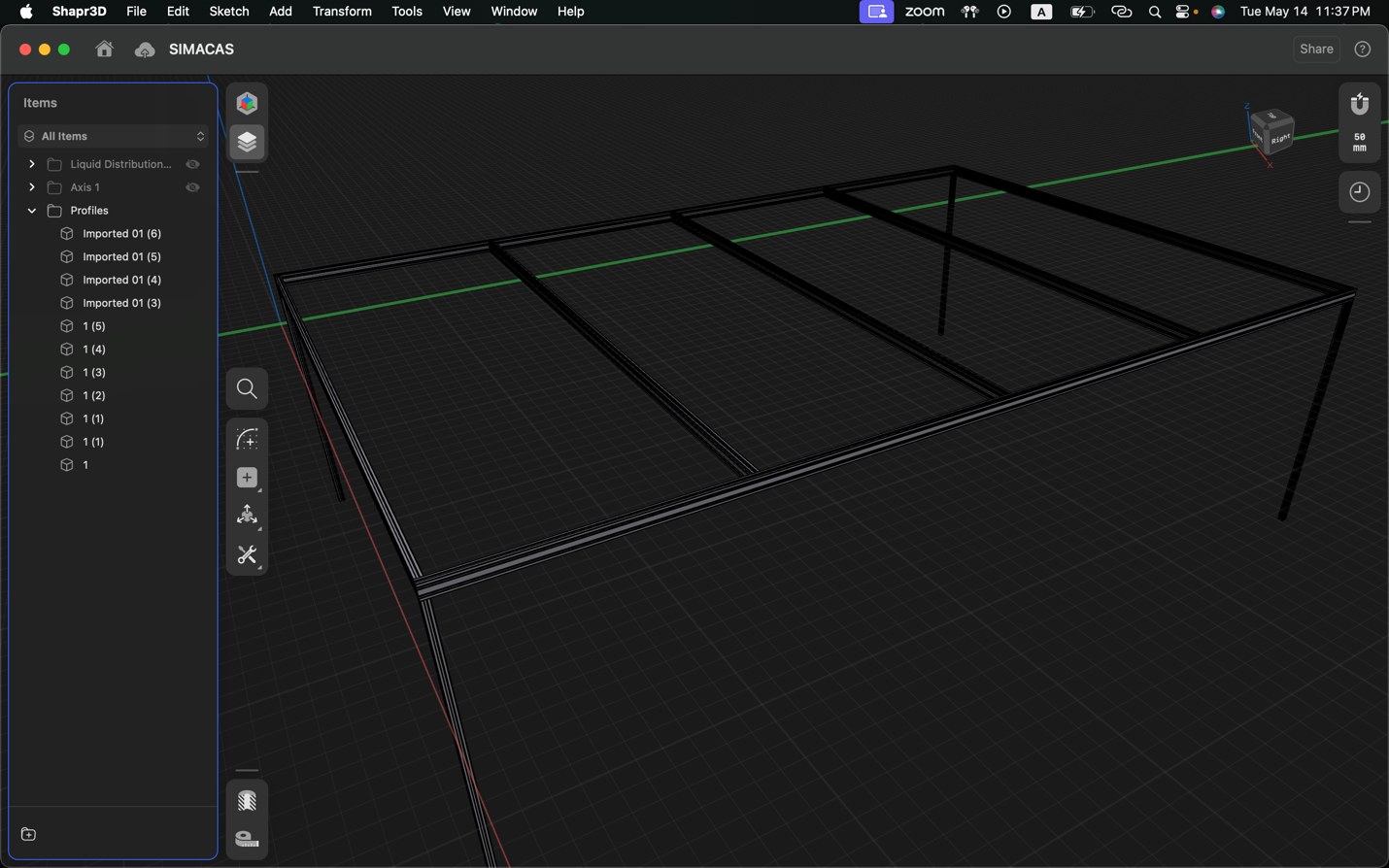
* 

Figure 1: •This image offers a detailed view of the frame's design, illustrating how the profiles are arranged and connected.

* **[Placeholder for Real-Life Photo]:** This photograph shows the actual assembled frame, highlighting its robustness and the precision of its construction.

The use of 2040 profiles offers several advantages. Firstly, the profiles provide an optimal balance of strength and weight, making them easy to handle and assemble. This feature is particularly beneficial for systems that need to be frequently moved or reconfigured.

In terms of durability and stability, the 2040 profiles ensure the frame can withstand the stresses of regular operation. Aluminum’s resistance to corrosion contributes to the longevity of the structure, even in humid or wet environments. Additionally, using standard-sized aluminum profiles helps keep costs manageable, as they are readily available and economical to source.

The modular nature of aluminum profiles allows for easy modifications and scalability. This flexibility enables the addition of extra profiles to extend or reinforce the frame, adapting to various sizes and requirements of different setups.

In Conclusion, The aluminum profile frame is a meticulously designed structure that provides the necessary support and stability for the automated irrigation system. By using 2040 profiles, the frame meets the mechanical strength requirements while offering ease of assembly, durability, and cost-effectiveness. The visual aids included in this section further illustrate the design and construction of the frame, showcasing its practical application in the project.

### Profile Traversal Mechanism

The Robot Traversal Mechanism was one of the most mechanically challenging aspects of the project, necessitating a unique design tailored specifically for this application. Given the need for intricate parts, 3D printing emerged as the ideal manufacturing solution. It allows for the creation of extremely complex geometries at an affordable price and offers faster manufacturing times compared to traditional subtractive methods.

The traversal robot is designed with two main halves. One half houses the batteries, custom PCB, and one of the two stepper motors used to drive the system. The other half contains the second stepper motor and the two sprinklers. These two halves are connected by four 5mm bolts running through V-wheel ball bearings. These bearings are crucial as they keep the system snugly in place while allowing free movement across the profile line with minimal friction.

A special 3D-printed coupler was designed for the stepper motor shafts. These couplers have a small screw hole to secure the coupler to the motor shaft and a groove where the bare V-wheel nylon shell (without the inner ball bearing) is hot-glued onto the 3D-printed coupler. Essentially, this coupler connects the motor shaft to a V-wheel shell, which grips the aluminum profile's V-groove and propels the system.

To provide a clearer understanding of the traversal robot, the following visual aids are included:

A computer screen shot of a machine

Description automatically generated

A computer screen shot of a machine

Description automatically generated

**< another 3d model photo of the middle part >**

This image shows the detailed 3D model of the robot, illustrating the design and component layout.

* **[Placeholder for Real-Life Photo]:** This photograph captures the assembled traversal robot, highlighting its key components and overall structure.

The 3D printing approach ensures that the robot can be manufactured quickly and affordably while maintaining the necessary precision and functionality. This innovative design enables the robot to traverse the aluminum profiles with high accuracy, ensuring precise water and fertilizer delivery.

Overall, the traversal robot's design reflects a balance between mechanical complexity and practical functionality, showcasing the benefits of 3D printing in developing specialized components for advanced agricultural systems.

### Water Distribution System

The water distribution system, while the least mechanically challenging aspect of the project, was nonetheless crucial for its success. The primary challenge involved developing the 3D-printed part that connects the water tank to the multiple traversal robots within the system. This component needed to ensure a reliable, water-tight connection to distribute water and fertilizer effectively.

After completing the 3D model, printing it, and testing, the right water-tight fittings and tubing had to be sourced. Once these components were acquired, the assembly of the entire system followed a specific sequence:

First, the water tank and the fertilizer tank were each connected to their respective 3D-printed 1-to-many adapters. These adapters serve as manifolds, allowing a single input from the tank to be split into multiple outputs, each leading to a different traversal robot.

Next, each outlet of these adapters was connected to a solenoid valve. These valves are normally closed and will only allow the flow of water or fertilizer when activated by a 12V signal. The signal is driven by a mechanical relay, which is itself powered by an optocoupler. This setup isolates the control signal, ensuring reliable operation controlled by the Main MCU, which will be discussed in detail in the electronics and programming chapters.

From each solenoid valve, long water tubes are used to deliver the fluids to the sprinklers mounted on the traversal robots. This configuration ensures that each robot receives the precise amount of water or fertilizer needed, delivered exactly where it is required.

To provide a clearer understanding of the water distribution system, the following visual aids are included:

* **[Placeholder for 3D Model Screenshot of a Solenoid Valve]:**
* **[Placeholder for 3D Model Screenshot of a 1-to-Many Water Adapter]:**
* **[Placeholder for 3D Model Screenshot of the Entire System]:**
* **[Placeholder for Real-Life Photo of a Solenoid Valve]:**
* **[Placeholder for Real-Life Photo of a 1-to-Many Water Adapter]:**
* **[Placeholder for Real-Life Photo of the Entire System]:**

These images illustrate both the conceptual design and the real-life implementation of the system, highlighting the integration of each component.

The careful design and assembly of the water distribution system ensure efficient and precise delivery of water and fertilizer, contributing significantly to the overall effectiveness and reliability of the automated irrigation system. The combination of 3D-printed parts, solenoid valves, and custom tubing solutions underscores the innovative approach taken to address the challenges of controlled environment agriculture.

## Electronics

### Introduction

The electronic design of the automated irrigation system is essential for ensuring precise control and monitoring of various parameters. The system requires three distinct circuits to be developed: the Axis Control Circuit for the traversal robot, a Sensor Interfacing Circuit, and a Main MCU Circuit. The Main MCU Circuit houses the Main MCU, which controls the other circuits and the water distribution system. Designing PCBs for each circuit is crucial as it ensures compact, reliable, and organized electronic configurations that enhance the system's overall performance and maintainability.

### Axis Control Circuit

The Axis Control Circuit is designed to receive commands wirelessly from the Main MCU board via an nRF24 radio transceiver module. It executes these commands by sending the appropriate signals to two stepper motor drivers, each driving its respective stepper motor. This circuit is also connected to an ultrasonic sensor that measures the distance from an absolute zero point, providing closed-loop control and significantly increasing position accuracy. All calculations and PID control are performed locally by a powerful STC MCU.

#### Key components of the Axis Control Circuit include:

* **Stepper Motors and Drivers:** Each stepper motor driver receives control signals from the STC MCU, enabling precise movement of the traversal robot.
* **Ultrasonic Sensor:** This sensor provides real-time position feedback, ensuring accurate traversal along the aluminum profiles.
* **STC MCU:** A powerful microcontroller that handles local computations and control tasks, including PID control for enhanced position accuracy.
* **nRF24 Radio Transceiver Module:** Enables wireless communication with the Main MCU, receiving commands and sending status updates.

#### Visual Documentation for Axis Control Circuit

* **[Placeholder for Schematic of Axis Control System]:** Detailed diagram showing the connections between the stepper motors, motor drivers, ultrasonic sensor, STC MCU, and nRF24 module.
* **[Placeholder for PCB Design of Axis Control Circuit]:** Screenshot of the PCB layout designed for the Axis Control Circuit.
* **[Placeholder for Real-Life Photo of Axis Control PCB]:** Photo of the fabricated and assembled PCB.

### Sensor Interfacing Circuit

The Sensor Interfacing Circuit features a local STC MCU that interfaces with a variety of sensors, including temperature, humidity, soil moisture, CO2, and light sensors. This circuit collects environmental data and transmits it to the Main MCU board using a one-wire communication protocol.

#### Key components and functions of the Sensor Interfacing Circuit include:

* **Temperature and Humidity Sensors:** Monitor ambient conditions around the plants.
* **Soil Moisture Sensors:** Measure the moisture content in the soil, critical for irrigation control.
* **CO2 and Light Sensors:** Provide data on CO2 levels and light intensity, essential for optimizing plant growth.
* **STC MCU:** Collects sensor data and manages communication with the Main MCU.
* **One-Wire Communication Protocol:** Ensures efficient transmission of sensor readings to the Main MCU.

#### Visual Documentation for Sensor Interfacing Circuit

* **[Placeholder for Schematic of Sensor Interfacing Circuit]:** Diagram illustrating the connections between the sensors, STC MCU, and the one-wire communication interface.
* **[Placeholder for PCB Design of Sensor Interfacing Circuit]:** Screenshot of the PCB layout designed for the Sensor Interfacing Circuit.
* **[Placeholder for Real-Life Photo of Sensor Interfacing PCB]:** Photo of the fabricated and assembled PCB.

### Main MCU Circuit

The Main MCU Circuit, powered by a Raspberry Pi Pico W running MicroPython, serves as the central control unit. It receives sensor readings via the one-wire communication protocol, sends positioning commands wirelessly through the nRF24 radio transceiver module, and controls the solenoid valves via individual signals to each optocoupler that manages the relays. Additionally, the Main MCU provides PWM signals to a light strip covering the irrigation area, controlling the lighting intensity.

#### Key responsibilities and components of the Main MCU Circuit include:

* **Raspberry Pi Pico W:** Acts as the Main MCU, coordinating all control and monitoring functions.
* **nRF24 Radio Transceiver Module:** Facilitates wireless communication with the Axis Control Circuit.
* **Optocouplers and Relays:** Control the solenoid valves, managing water and fertilizer flow.
* **PWM Control for Light Strip:** Adjusts lighting intensity along the irrigation area.
* **Wi-Fi Module:** Hosts a simple website allowing users to monitor and control the entire system remotely.

#### Visual Documentation for Main MCU Circuit

* **[Placeholder for Schematic of Main MCU Circuit]:** Diagram showing the connections between the Raspberry Pi Pico W, nRF24 module, optocouplers, relays, solenoid valves, and the light strip.
* **[Placeholder for PCB Design of Main MCU Circuit]:** Screenshot of the PCB layout designed for the Main MCU Circuit.
* **[Placeholder for Real-Life Photo of Main MCU PCB]:** Photo of the fabricated and assembled PCB.

### Importance of PCB Design

Designing PCBs for each circuit is fundamental to the project's success. PCBs offer a compact and reliable way to organize and connect electronic components. They ensure consistent performance by reducing the risk of loose connections and improving the overall durability of the system. Moreover, PCBs simplify troubleshooting and maintenance, making the system more user-friendly and scalable. The precision of PCB design also enhances the efficiency of signal transmission and power distribution, which is critical for the accurate operation of the automated irrigation system.

### Conclusion

The electronics chapter provides a comprehensive overview of the components and systems that enable precise control and monitoring of the automated irrigation system. The integration of stepper motors, solenoid valves, sensors, and sophisticated programming ensures that the system operates efficiently and effectively, meeting the project's goals of precision agriculture. The visual aids included further illustrate the intricate design and functionality of the electronic components, highlighting their critical role in the success of the system.

## Programming

### Intro

The programming of the automated irrigation system is integral to its functionality and efficiency. This chapter delves into the detailed programming efforts behind the Axis Control Board, Sensor Monitoring Board, and the Main Board, culminating in the implementation of a sophisticated Model Predictive Control (MPC) Algorithm.

1. **Axis Control Board STC MCU Programming:**
   * The Axis Control Board's STC MCU is responsible for executing commands received wirelessly from the Main MCU. It controls the stepper motors to ensure precise movement of the traversal robot. The programming includes handling wireless communication, motor control, and position feedback using an ultrasonic sensor. Advanced PID control algorithms are implemented locally to enhance position accuracy.
2. **Sensor Monitoring STC MCU Programming:**
   * The Sensor Monitoring Board's STC MCU interfaces with various environmental sensors, including temperature, humidity, soil moisture, CO2, and light sensors. This section covers the programming needed to collect data from these sensors and transmit the readings to the Main MCU via a one-wire communication protocol. The programming ensures efficient data acquisition and reliable communication.
3. **Main Board MicroPython Programming:**
   * The Main Board, powered by a Raspberry Pi Pico W running MicroPython, serves as the central hub for controlling the entire system. This section details the programming required to receive sensor data, send positioning commands to the Axis Control Board, and control the solenoid valves and light strip. The Main Board also hosts a web interface, allowing users to monitor and control the system remotely via its built-in Wi-Fi module.
4. **MPC Control Algorithm:**
   * The implementation of the Model Predictive Control (MPC) algorithm is the highlight of the system's programming. The MPC algorithm optimizes irrigation schedules based on real-time sensor data and predictive modeling. This section explains how the algorithm anticipates future conditions and makes proactive adjustments to maximize plant growth and resource efficiency. The integration of the MPC algorithm within the Main Board's programming ensures precise and adaptive control of the irrigation system.

Each of these sections provides an in-depth look at the programming techniques and challenges addressed throughout the development of the automated irrigation system. The following sections will elaborate on the specific code structures, logic, and algorithms used to bring this innovative project to life, highlighting the meticulous work and ingenuity involved in its creation.

### Axis Control Board STC MCU Programming

#### Main Objective and Logic

The main objective of the Axis Control Board STC MCU is to execute precise movement commands for the traversal robot, received wirelessly from the Main MCU. To achieve this, the board must accurately control the stepper motors, ensuring the robot moves to the correct position along the aluminum profiles. This involves handling wireless communication, motor control, and position feedback using an ultrasonic sensor. Advanced PID control algorithms are implemented locally to maintain high positioning accuracy and achieve closed-loop control.

The Axis Control Board utilizes the STC8H8K64U MCU, a powerful microcontroller running at 35MHz. This MCU supports up to 64KB of code and 8KB of RAM. It features five timers, including one 32-bit timer and two 16-bit timers, multiple hardware UARTs, and multiple SPI and I2C interfaces. This combination of features makes it an excellent fit for the application. Programming is done using embedded C with the open-source uni-stc library for low-level hardware drivers. The program is flashed using a USB-to-TTL converter and the CLI tool stcgal, which is versatile, reliable, and easy to use, thanks to the MCU's in-built bootloader.

#### PID Control Algorithm

A key feature of the Axis Control Board is the implementation of a PID (Proportional-Integral-Derivative) control algorithm. The PID controller continuously calculates an error value as the difference between a desired setpoint (target position) and the measured process variable (current position). It applies corrections based on proportional, integral, and derivative terms to minimize this error. The algorithm ensures precise control over the robot's position by dynamically adjusting the motor commands, enabling absolute value control.

#### Components and Their Functions

1. **nrf24l01.c:**
   * This module handles wireless communication between the Axis Control Board and the Main MCU. It uses the nRF24L01 radio transceiver module to send and receive commands and status updates.
   * **Functionality:** Initializes the nRF24L01 module, manages data transmission and reception, and handles communication protocols to ensure reliable wireless communication.
2. **project-defs.h:**
   * This header file contains project-specific definitions, constants, and macros used throughout the code. It centralizes key parameters, making the code more maintainable and easier to configure.
   * **Functionality:** Defines hardware pin mappings, system parameters, and configuration settings.
3. **protocol.c:**
   * This module defines the communication protocol used for data exchange between the Axis Control Board and the Main MCU. It ensures that commands and status messages are correctly formatted and interpreted.
   * **Functionality:** Implements functions for encoding and decoding messages, managing communication states, and handling protocol-specific tasks.
4. **stepper\_motor.c:**
   * This module controls the stepper motors, generating the appropriate signals to drive them based on commands received from the Main MCU. It includes functions for motor initialization, step control, and speed adjustment.
   * **Functionality:** Interfaces with motor drivers, executes movement commands, and provides functions for precise motor control.
5. **terminal.c:**
   * This module provides a terminal interface for debugging and manual control. It allows developers to interact with the Axis Control Board, sending commands and receiving feedback through a serial interface.
   * **Functionality:** Implements a command-line interface, processes user inputs, and outputs system status and debug information.
6. **ultrasonic.c:**
   * This module interfaces with the ultrasonic sensor to measure the distance from an absolute zero point. It provides accurate position feedback necessary for closed-loop control.
   * **Functionality:** Initializes the ultrasonic sensor, triggers measurements, and processes sensor data to determine the current position of the traversal robot.
7. **main.c:**
   * This is the main application code that integrates all the modules and orchestrates the overall operation of the Axis Control Board. It initializes the system, manages the control loop, and coordinates communication and motor control tasks.
   * **Functionality:** Sets up hardware components, initializes modules, implements the main control loop, processes received commands, and applies the PID control algorithm for precise positioning.

#### Summary

The Axis Control Board STC MCU is a sophisticated system designed to ensure precise movement of the traversal robot. Through the implementation of a PID control algorithm, wireless communication, and integration of various hardware components, the board achieves accurate and reliable control over the robot's positioning. The following sections will provide detailed code structures, logic explanations, and implementation details for each of the components listed above.

### Sensor Monitoring STC MCU Programming

#### Main Objective and Logic

The primary objective of the Sensor Monitoring STC MCU is to interface with various environmental sensors and collect real-time data, which is then transmitted to the Main MCU. The board uses a one-wire communication protocol to send aggregated sensor readings efficiently. The logic involves initializing each sensor, regularly polling them for data, and formatting the data for transmission to ensure seamless integration with the Main MCU.

The Sensor Monitoring board utilizes the STC8H8K64U MCU, a powerful microcontroller running at 35MHz. This MCU supports up to 64KB of code and 8KB of RAM. It features five timers, including one 32-bit timer and two 16-bit timers, multiple hardware UARTs, and multiple SPI and I2C interfaces. This combination of features makes it an excellent fit for the application. Programming is done using embedded C with the open-source uni-stc library for low-level hardware drivers. The program is flashed using a USB-to-TTL converter and the CLI tool stcgal, which is versatile, reliable, and easy to use, thanks to the MCU's in-built bootloader.

#### Components and Their Functions

1. **project-def.h:**
   * This header file includes project-specific definitions, constants, and macros used across the sensor monitoring codebase. It centralizes key parameters for easier maintenance and configuration.
   * **Functionality:** Defines sensor pin mappings, system parameters, and configuration settings.
2. **one-wire.c:**
   * This module handles the one-wire communication protocol used to transmit sensor data to the Main MCU. It ensures efficient and reliable data transfer over a single communication line.
   * **Functionality:** Implements functions for initializing the one-wire bus, sending and receiving data packets, and managing communication errors.
3. **temperature.c:**
   * This module interfaces with temperature sensors, reading ambient temperature values necessary for controlling the irrigation environment.
   * **Functionality:** Initializes the temperature sensor, reads temperature data, and processes it for transmission.
4. **humidity.c:**
   * This module handles the humidity sensors, collecting data on ambient humidity levels, which is crucial for maintaining optimal plant growth conditions.
   * **Functionality:** Initializes the humidity sensor, reads humidity data, and processes it for transmission.
5. **light\_intensity.c:**
   * This module interfaces with light sensors, measuring the intensity of light in the environment, which is essential for plant photosynthesis.
   * **Functionality:** Initializes the light sensor, reads light intensity data, and processes it for transmission.
6. **co2.c:**
   * This module handles CO2 sensors, providing data on carbon dioxide levels, an important factor for plant health and growth.
   * **Functionality:** Initializes the CO2 sensor, reads CO2 concentration data, and processes it for transmission.
7. **water\_level.c:**
   * This module interfaces with soil moisture sensors, measuring the moisture content in the soil to ensure appropriate irrigation levels.
   * **Functionality:** Initializes the soil moisture sensor, reads moisture data, and processes it for transmission.
8. **main.c:**
   * This is the main application code that integrates all sensor modules and coordinates their operation. It initializes the system, manages the sensor polling schedule, and handles the transmission of aggregated data to the Main MCU.
   * **Functionality:** Sets up hardware components, initializes sensor modules, implements the main control loop, collects data from sensors, and uses the one-wire protocol to send data to the Main MCU.

#### Summary

The Sensor Monitoring STC MCU is designed to efficiently collect and transmit environmental data from various sensors. Through the use of a one-wire communication protocol, the board ensures reliable data transfer to the Main MCU. Each sensor-specific module is responsible for initializing its respective sensor, reading data, and preparing it for transmission. The following sections will provide detailed code structures, logic explanations, and implementation details for each of the components listed above.

### Main Board MicroPython Programming

The Main MCU for this project is the Raspberry Pi Pico W, featuring the powerful RP2040 MCU. This microcontroller unit has two cores running at 240MHz, making it capable of handling heavy processing tasks compared to other MCUs in its class. Its processing power and versatility make it an excellent choice for serving as the Main MCU in this project.

#### Python File Components and Their Functionality

1. **main.py:**
   * **Functionality:** The main entry point for the program. It initializes all necessary components, starts the web server, and handles the main control loop.
   * **Linkage:** Imports and initializes modules like one-wire.py, nrf24.py, and water\_distribution.py. It also sets up the web server to host the HTML interface.
2. **one-wire.py:**
   * **Functionality:** Implements the one-wire communication protocol to interface with the sensor monitoring board.
   * **Linkage:** Provides a OneWire class that is used by main.py to read sensor data and transmit it to the Main MCU.
3. **nrf24.py:**
   * **Functionality:** Manages the SPI communication with the nRF24 module, sending commands to the Axis Control Board.
   * **Linkage:** Provides an NRF24 class that main.py uses to send positioning commands to the traversal robot.
4. **water\_distribution.py:**
   * **Functionality:** Controls the GPIO pins connected to the solenoid valves, managing water and fertilizer distribution.
   * **Linkage:** Provides a WaterDistribution class used by main.py to control the solenoid valves based on commands received from the web interface.
5. **web\_server.py:**
   * **Functionality:** Hosts the HTML interface and handles HTTP GET and POST requests from the client.
   * **Linkage:** Uses classes from nrf24.py and water\_distribution.py to process inputs from the web interface and control the system accordingly.
6. **index.html:**
   * **Functionality:** The web interface for the project, titled "SIMACAS". It includes vertical sliders for controlling the position of each traversal robot, toggle buttons for each solenoid valve, and a text area displaying all sensor readings.
   * **Linkage:** Served by web\_server.py and interacts with main.py through HTTP requests.

#### Detailed Explanation of Each File

##### **one-wire.py**

**Overview:** Implements the one-wire communication protocol to interface with the sensor monitoring board.

* **Class: OneWire**
  + **Attributes:**
    - pin: GPIO pin used for one-wire communication.
  + **Methods:**
    - \_\_init\_\_(self, pin): Initializes the one-wire communication on the specified pin.
    - reset(self): Resets the one-wire bus.
    - write\_byte(self, data): Writes a byte of data to the one-wire bus.
    - read\_byte(self): Reads a byte of data from the one-wire bus.
    - read\_sensors(self): Reads sensor data and returns it in a structured format.

##### **nrf24.py**

**Overview:** Manages SPI communication with the nRF24 module to send commands to the Axis Control Board.

* **Class: NRF24**
  + **Attributes:**
    - spi: SPI interface.
    - csn\_pin: Chip Select Not pin.
    - ce\_pin: Chip Enable pin.
  + **Methods:**
    - \_\_init\_\_(self, spi, csn\_pin, ce\_pin): Initializes the nRF24 module with the given SPI interface and pins.
    - send\_command(self, command): Sends a command to the Axis Control Board.
    - set\_position(self, position): Sends a command to set the traversal robot to a specific position.
    - receive\_status(self): Receives status updates from the Axis Control Board.

##### **water\_distribution.py**

**Overview:** Controls the GPIO pins connected to the solenoid valves, managing water and fertilizer distribution.

* **Class: WaterDistribution**
  + **Attributes:**
    - valve\_pins: List of GPIO pins connected to the solenoid valves.
  + **Methods:**
    - \_\_init\_\_(self, valve\_pins): Initializes the water distribution system with the specified valve pins.
    - open\_valve(self, valve\_index): Opens the specified solenoid valve.
    - close\_valve(self, valve\_index): Closes the specified solenoid valve.
    - toggle\_valve(self, valve\_index): Toggles the specified solenoid valve.

##### **web\_server.py**

**Overview:** Hosts the HTML interface and handles HTTP GET and POST requests from the client.

* **Class: WebServer**
  + **Attributes:**
    - server: Web server instance.
    - nrf24: Instance of the NRF24 class.
    - water\_distribution: Instance of the WaterDistribution class.
  + **Methods:**
    - \_\_init\_\_(self, nrf24, water\_distribution): Initializes the web server with instances of NRF24 and WaterDistribution.
    - start(self): Starts the web server.
    - handle\_request(self, request): Handles incoming HTTP requests, parses GET and POST data, and calls appropriate methods from nrf24.py and water\_distribution.py.

##### **index.html**

**Overview:** The web interface titled "SIMACAS". It includes controls for the system.

* **Structure:**
  + **Title:** "SIMACAS"
  + **Elements:**
    - Vertical sliders for each traversal robot to control their absolute positions.
    - Toggle buttons for each solenoid valve.
    - Text area displaying all sensor readings.

< screenshot of html website >

##### **main.py**

**Overview:** The main entry point for the program. Initializes components and starts the web server.

* **Functionality:**
  + Initializes instances of OneWire, NRF24, and WaterDistribution.
  + Sets up the web server using WebServer.
  + Implements the main control loop to manage system operations.

#### Summary

The Main Board's programming leverages the powerful Raspberry Pi Pico W and its RP2040 MCU to coordinate the entire automated irrigation system. The structured approach using modular Python files ensures robust, maintainable, and scalable code. Each file has a specific role, from handling sensor data to controlling the traversal robot and managing the web interface. The object-oriented programming (OOP) paradigm is used extensively to encapsulate functionality and maintain clear separation of concerns, enhancing both code readability and system reliability.

### Main Control Algorithm – Model Predictive Control (MPC)

#### History of MPC

Model Predictive Control (MPC) has a rich history that traces back to the early developments in control theory and optimization in the late 20th century. It has evolved from basic concepts of optimal control to a sophisticated control strategy widely used in various industries today.

The foundations of MPC lie in optimal control theory, which emerged in the 1960s and 1970s. Optimal control aims to find control laws for dynamic systems that optimize a certain performance criterion. Early works by pioneers such as Rudolf Kalman and Lev Pontryagin laid the groundwork for optimal control. Kalman's contributions to linear quadratic regulators (LQR) and the Kalman filter were particularly influential.

The term Model Predictive Control itself was coined in the late 1970s. During this period, the concept of receding horizon control (a precursor to MPC) began to take shape. Receding horizon control involves solving an optimization problem over a finite horizon and implementing only the first part of the solution before recalculating the optimization for the next horizon. This approach allows for continuous updating and adjustment based on new information.

In the 1980s, MPC started to gain significant attention in the process industries, particularly in chemical engineering. The development of dynamic matrix control (DMC) by Cutler and Ramaker in 1979 was a major milestone. DMC used a linear model to predict future plant outputs and optimize control moves. This was one of the first practical implementations of MPC in industry.

The 1990s saw substantial advances in MPC algorithms and their applications. Researchers began developing more sophisticated models and optimization techniques, allowing MPC to handle a broader range of control problems. Key developments included the use of quadratic programming (QP) for solving the optimization problems and the incorporation of constraints on inputs and outputs, which are crucial for practical applications.

During this period, MPC also started to be applied in various fields beyond chemical engineering, including aerospace, automotive, and energy systems. The flexibility and robustness of MPC made it an attractive choice for complex control tasks.

In the 2000s and beyond, the integration of MPC with modern computational techniques and advances in hardware significantly expanded its capabilities. The rise of powerful processors and real-time computing allowed MPC to be applied in faster and more complex systems. Researchers explored the use of nonlinear models, hybrid systems, and robust MPC to handle uncertainties and disturbances more effectively.

The development of explicit MPC, which pre-computes the control law offline and stores it as a piecewise affine function, made MPC feasible for applications with strict real-time requirements. Additionally, advancements in machine learning and data-driven modeling have further enhanced the predictive accuracy and adaptability of MPC.

Today, MPC is a well-established control strategy with applications in a wide range of industries, from robotics and autonomous vehicles to energy management and smart grids. Current research focuses on integrating MPC with machine learning, distributed and decentralized MPC, and leveraging cloud computing and the Internet of Things (IoT) for large-scale and networked control systems.

The future of MPC looks promising, with ongoing efforts to make it more accessible and easier to implement through the development of user-friendly software tools and libraries. The continuous improvement in computational power and algorithms will likely drive further innovations, enabling MPC to tackle even more complex and dynamic control challenges.

In Summary, The history of Model Predictive Control (MPC) reflects its evolution from early concepts in optimal control to a versatile and powerful control strategy widely used across various industries. With its ability to handle multivariable control problems, constraints, and uncertainties, MPC has become an essential tool for modern control systems. The integration with advanced computational techniques and ongoing research efforts ensures that MPC will continue to play a crucial role in the future of control engineering.

#### Detailed Explanation of MPC

Model Predictive Control (MPC) is an advanced control strategy that uses an explicit model of the process to predict the future behavior of the system and optimize control moves. This section provides an in-depth explanation of how the MPC algorithm works, including the mathematical equations involved.

##### Basic Principles of MPC

The core idea behind MPC is to solve an optimization problem at each control step. The optimization problem aims to minimize a cost function over a finite prediction horizon while satisfying system constraints. The solution involves predicting the future states of the system, calculating the optimal control inputs, and applying the first control input in the sequence. This process is repeated at every time step with updated system information, leading to a moving or receding horizon approach.

##### Components of MPC

1. **Prediction Model:**
   * A mathematical model representing the system dynamics is used to predict future states. The model can be linear or nonlinear, depending on the system.
   * For a linear time-invariant system, the state-space representation is commonly used:

x(k+1)=Ax(k)+Bu(k)x(k+1) = A x(k) + B u(k)x(k+1)=Ax(k)+Bu(k)

y(k)=Cx(k)+Du(k)y(k) = C x(k) + D u(k)y(k)=Cx(k)+Du(k)

where:

* + - x(k)x(k)x(k) is the state vector at time step kkk
    - u(k)u(k)u(k) is the control input vector
    - y(k)y(k)y(k) is the output vector
    - A,B,C,DA, B, C, DA,B,C,D are matrices defining the system dynamics

1. **Prediction Horizon (N):**
   * The length of the future time window over which predictions are made. A longer horizon generally provides better performance but increases computational complexity.
2. **Control Horizon (M):**
   * The number of future control moves to be optimized. Typically, M≤NM \leq NM≤N.
3. **Cost Function:**
   * The objective function that quantifies the performance of the control actions. A common choice is the quadratic cost function:

J=∑i=0N−1[(y(k+i)−r(k+i))TQ(y(k+i)−r(k+i))+Δu(k+i)TRΔu(k+i)]J = \sum\_{i=0}^{N-1} \left[ (y(k+i) - r(k+i))^T Q (y(k+i) - r(k+i)) + \Delta u(k+i)^T R \Delta u(k+i) \right]J=i=0∑N−1​[(y(k+i)−r(k+i))TQ(y(k+i)−r(k+i))+Δu(k+i)TRΔu(k+i)]

where:

* + - y(k+i)y(k+i)y(k+i) is the predicted output at time step k+ik+ik+i
    - r(k+i)r(k+i)r(k+i) is the reference trajectory at time step k+ik+ik+i
    - Δu(k+i)\Delta u(k+i)Δu(k+i) is the change in control input u(k+i)−u(k+i−1)u(k+i) - u(k+i-1)u(k+i)−u(k+i−1)
    - QQQ and RRR are weighting matrices

1. **Constraints:**
   * Constraints on the states and control inputs, which can include physical limitations and safety requirements:

xmin⁡≤x(k+i)≤xmax⁡x\_{\min} \leq x(k+i) \leq x\_{\max}xmin​≤x(k+i)≤xmax​

umin⁡≤u(k+i)≤umax⁡u\_{\min} \leq u(k+i) \leq u\_{\max}umin​≤u(k+i)≤umax​

##### MPC Optimization Problem

The MPC algorithm involves solving the following optimization problem at each time step kkk:

1. **Predict Future States:**
   * Using the prediction model, estimate the future states and outputs over the prediction horizon NNN.
2. **Formulate the Cost Function:**
   * Define the cost function JJJ to minimize the difference between the predicted outputs and the reference trajectory, as well as the magnitude of control input changes.
3. **Incorporate Constraints:**
   * Include constraints on states and control inputs in the optimization problem.
4. **Solve the Optimization Problem:**
   * Find the sequence of control inputs {u(k),u(k+1),…,u(k+M−1)}\{u(k), u(k+1), \ldots, u(k+M-1)\}{u(k),u(k+1),…,u(k+M−1)} that minimizes the cost function while satisfying the constraints.
5. **Apply the Control Input:**
   * Implement the first control input u(k)u(k)u(k) from the optimized sequence.
6. **Update the System:**
   * At the next time step k+1k+1k+1, update the state information and repeat the process.

Mathematically, the optimization problem can be expressed as:

min⁡{u(k),…,u(k+M−1)}J\min\_{\{u(k), \ldots, u(k+M-1)\}} J{u(k),…,u(k+M−1)}min​J

subject to:

x(k+i+1)=Ax(k+i)+Bu(k+i),i=0,…,N−1x(k+i+1) = A x(k+i) + B u(k+i), \quad i = 0, \ldots, N-1x(k+i+1)=Ax(k+i)+Bu(k+i),i=0,…,N−1

y(k+i)=Cx(k+i)+Du(k+i),i=0,…,N−1y(k+i) = C x(k+i) + D u(k+i), \quad i = 0, \ldots, N-1y(k+i)=Cx(k+i)+Du(k+i),i=0,…,N−1

xmin⁡≤x(k+i)≤xmax⁡,i=1,…,Nx\_{\min} \leq x(k+i) \leq x\_{\max}, \quad i = 1, \ldots, Nxmin​≤x(k+i)≤xmax​,i=1,…,N

umin⁡≤u(k+i)≤umax⁡,i=0,…,M−1u\_{\min} \leq u(k+i) \leq u\_{\max}, \quad i = 0, \ldots, M-1umin​≤u(k+i)≤umax​,i=0,…,M−1

##### Example of MPC Implementation

Consider a simple linear system with state-space representation:

x(k+1)=[1T01]x(k)+[0T]u(k)x(k+1) = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x(k) + \begin{bmatrix} 0 \\ T \end{bmatrix} u(k)x(k+1)=[10​T1​]x(k)+[0T​]u(k)

y(k)=[10]x(k)y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} x(k)y(k)=[1​0​]x(k)

where TTT is the sampling time.

For a prediction horizon NNN and control horizon MMM:

1. **Prediction Model:**
   * Predict future states using the state-space equations.
2. **Cost Function:**
   * Define the cost function to minimize deviations from the reference and control effort:

J=∑i=0N−1[(y(k+i)−r(k+i))2+λ(Δu(k+i))2]J = \sum\_{i=0}^{N-1} \left[ (y(k+i) - r(k+i))^2 + \lambda (\Delta u(k+i))^2 \right]J=i=0∑N−1​[(y(k+i)−r(k+i))2+λ(Δu(k+i))2]

where λ\lambdaλ is a tuning parameter for control effort.

1. **Constraints:**
   * Include constraints on states and inputs, such as:

−10≤x(k+i)≤10-10 \leq x(k+i) \leq 10−10≤x(k+i)≤10

−2≤u(k+i)≤2-2 \leq u(k+i) \leq 2−2≤u(k+i)≤2

1. **Optimization:**
   * Solve the quadratic programming (QP) problem to find the optimal control inputs.

##### Solving the QP Problem

The QP problem can be solved using numerical optimization algorithms. Popular methods include:

* **Interior Point Methods:** Efficient for large-scale QP problems.
* **Active Set Methods:** Suitable for smaller problems and real-time applications.
* **Sequential Quadratic Programming (SQP):** Combines QP with nonlinear programming techniques.

##### Summary

Model Predictive Control (MPC) is a powerful control strategy that leverages model-based predictions and optimization to achieve optimal control performance. By solving a finite-horizon optimization problem at each time step, MPC can handle multivariable systems with constraints, making it suitable for a wide range of applications. The detailed understanding of MPC involves the formulation of the prediction model, cost function, and constraints, followed by solving the optimization problem to determine the optimal control actions.

#### Proposed MPC Model for the Automated Irrigation System

The proposed Model Predictive Control (MPC) model for the automated irrigation system takes into account the critical environmental parameters (temperature, humidity, CO2, and light intensity) and aims to optimize the system outputs (water flow rate, water positioning, and light intensity). This model uses the sensor readings as inputs to predict future states and control the irrigation and lighting systems effectively.

##### System Inputs and Outputs

* **Inputs (Measured by Sensors):**
  + T(k)T(k)T(k): Temperature at time step kkk
  + H(k)H(k)H(k): Humidity at time step kkk
  + CO2(k)CO2(k)CO2(k): CO2 concentration at time step kkk
  + L(k)L(k)L(k): Light intensity at time step kkk
* **Outputs (Controlled Variables):**
  + Qw(k)Q\_w(k)Qw​(k): Water flow rate at time step kkk
  + Pw(k)P\_w(k)Pw​(k): Water positioning at time step kkk
  + Lc(k)L\_c(k)Lc​(k): Light intensity control at time step kkk

##### State-Space Representation

To model the system in a state-space form, we define the state vector x(k)x(k)x(k), control input vector u(k)u(k)u(k), and output vector y(k)y(k)y(k) as follows:

* **State Vector:**

x(k)=[T(k)H(k)CO2(k)L(k)]x(k) = \begin{bmatrix} T(k) \\ H(k) \\ CO2(k) \\ L(k) \end{bmatrix}x(k)=⎣⎡​T(k)H(k)CO2(k)L(k)​⎦⎤​

* **Control Input Vector:**

u(k)=[Qw(k)Pw(k)Lc(k)]u(k) = \begin{bmatrix} Q\_w(k) \\ P\_w(k) \\ L\_c(k) \end{bmatrix}u(k)=⎣⎡​Qw​(k)Pw​(k)Lc​(k)​⎦⎤​

* **Output Vector:**

y(k)=[T(k)H(k)CO2(k)L(k)]y(k) = \begin{bmatrix} T(k) \\ H(k) \\ CO2(k) \\ L(k) \end{bmatrix}y(k)=⎣⎡​T(k)H(k)CO2(k)L(k)​⎦⎤​

The system dynamics can be represented using the state-space equations:

x(k+1)=Ax(k)+Bu(k)+Ed(k)x(k+1) = A x(k) + B u(k) + E d(k)x(k+1)=Ax(k)+Bu(k)+Ed(k)

y(k)=Cx(k)+Du(k)y(k) = C x(k) + D u(k)y(k)=Cx(k)+Du(k)

where:

* AAA is the state transition matrix
* BBB is the control input matrix
* EEE is the disturbance matrix
* CCC is the output matrix
* DDD is the feedthrough matrix
* d(k)d(k)d(k) represents external disturbances (e.g., environmental variations)

##### State-Space Matrices

For simplicity, assume a linear relationship between the states and control inputs. The matrices can be approximated as follows:

A=[1000010000100001],B=[β11β12β13β21β22β23β31β32β33β41β42β43],E=[ϵ1ϵ2ϵ3ϵ4]A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \beta\_{11} & \beta\_{12} & \beta\_{13} \\ \beta\_{21} & \beta\_{22} & \beta\_{23} \\ \beta\_{31} & \beta\_{32} & \beta\_{33} \\ \beta\_{41} & \beta\_{42} & \beta\_{43} \end{bmatrix}, \quad E = \begin{bmatrix} \epsilon\_{1} \\ \epsilon\_{2} \\ \epsilon\_{3} \\ \epsilon\_{4} \end{bmatrix}A=⎣⎡​1000​0100​0010​0001​⎦⎤​,B=⎣⎡​β11​β21​β31​β41​​β12​β22​β32​β42​​β13​β23​β33​β43​​⎦⎤​,E=⎣⎡​ϵ1​ϵ2​ϵ3​ϵ4​​⎦⎤​

C=I4×4,D=04×3C = I\_{4 \times 4}, \quad D = 0\_{4 \times 3}C=I4×4​,D=04×3​

where βij\beta\_{ij}βij​ are coefficients that define how the control inputs affect the states, and ϵi\epsilon\_{i}ϵi​ represent the effects of external disturbances on the states.

##### Cost Function

The MPC algorithm aims to minimize a quadratic cost function over a prediction horizon NNN:

J=∑i=0N−1[(y(k+i)−r(k+i))TQ(y(k+i)−r(k+i))+u(k+i)TRu(k+i)]J = \sum\_{i=0}^{N-1} \left[ (y(k+i) - r(k+i))^T Q (y(k+i) - r(k+i)) + u(k+i)^T R u(k+i) \right]J=i=0∑N−1​[(y(k+i)−r(k+i))TQ(y(k+i)−r(k+i))+u(k+i)TRu(k+i)]

where:

* y(k+i)y(k+i)y(k+i) is the predicted output vector at future time step k+ik+ik+i
* r(k+i)r(k+i)r(k+i) is the reference trajectory for the output vector
* QQQ is a positive semi-definite matrix that weights the tracking error
* RRR is a positive definite matrix that weights the control effort

##### Constraints

The system is subject to physical and safety constraints on the states and control inputs:

xmin⁡≤x(k+i)≤xmax⁡,i=1,…,Nx\_{\min} \leq x(k+i) \leq x\_{\max}, \quad i = 1, \ldots, Nxmin​≤x(k+i)≤xmax​,i=1,…,N

umin⁡≤u(k+i)≤umax⁡,i=0,…,M−1u\_{\min} \leq u(k+i) \leq u\_{\max}, \quad i = 0, \ldots, M-1umin​≤u(k+i)≤umax​,i=0,…,M−1

##### Optimization Problem

At each time step kkk, the MPC optimization problem can be formulated as:

min⁡{u(k),…,u(k+M−1)}J\min\_{\{u(k), \ldots, u(k+M-1)\}} J{u(k),…,u(k+M−1)}min​J

subject to:

x(k+i+1)=Ax(k+i)+Bu(k+i)+Ed(k+i),i=0,…,N−1x(k+i+1) = A x(k+i) + B u(k+i) + E d(k+i), \quad i = 0, \ldots, N-1x(k+i+1)=Ax(k+i)+Bu(k+i)+Ed(k+i),i=0,…,N−1

y(k+i)=Cx(k+i)+Du(k+i),i=0,…,N−1y(k+i) = C x(k+i) + D u(k+i), \quad i = 0, \ldots, N-1y(k+i)=Cx(k+i)+Du(k+i),i=0,…,N−1

xmin⁡≤x(k+i)≤xmax⁡,i=1,…,Nx\_{\min} \leq x(k+i) \leq x\_{\max}, \quad i = 1, \ldots, Nxmin​≤x(k+i)≤xmax​,i=1,…,N

umin⁡≤u(k+i)≤umax⁡,i=0,…,M−1u\_{\min} \leq u(k+i) \leq u\_{\max}, \quad i = 0, \ldots, M-1umin​≤u(k+i)≤umax​,i=0,…,M−1

##### Summary

The proposed MPC model for the automated irrigation system integrates sensor readings (temperature, humidity, CO2, and light intensity) to optimize the control outputs (water flow rate, water positioning, and light intensity). By solving a finite-horizon optimization problem at each control step, the MPC algorithm ensures that the system operates efficiently and effectively, maintaining optimal growing conditions while adhering to physical and safety constraints. This model provides a robust framework for implementing precise and adaptive control in the irrigation system.

#### MPC Algorithm Implementation in the Discrete Domain

Implementing a Model Predictive Control (MPC) algorithm in the discrete domain involves several systematic steps, from system modeling to real-time implementation. Here’s a detailed guide:

##### 1. System Identification and Modeling

**Objective:** Develop a mathematical model that accurately describes the system dynamics.

1. **Data Collection:**
   * Collect input-output data from the system through experiments or simulations.
2. **Model Selection:**
   * Choose an appropriate model structure (e.g., state-space model, transfer function, or ARX model).
   * For state-space models, define the state vector x(k)x(k)x(k), control input u(k)u(k)u(k), and output y(k)y(k)y(k).
3. **Parameter Estimation:**
   * Use system identification techniques to estimate the model parameters.
   * Ensure the model captures the essential dynamics and is of appropriate complexity.
4. **Model Validation:**
   * Validate the model using a separate set of data to ensure it accurately represents the system.

##### 2. Discretization of the Continuous-Time Model

**Objective:** Convert the continuous-time model to a discrete-time model for digital implementation.

1. **Continuous-Time State-Space Model:**

x˙(t)=Acx(t)+Bcu(t)\dot{x}(t) = A\_c x(t) + B\_c u(t)x˙(t)=Ac​x(t)+Bc​u(t)

y(t)=Cx(t)+Du(t)y(t) = C x(t) + D u(t)y(t)=Cx(t)+Du(t)

1. **Discretize the Model:**
   * Use a sampling period TsT\_sTs​ to convert the model to discrete-time.

x(k+1)=Ax(k)+Bu(k)x(k+1) = A x(k) + B u(k)x(k+1)=Ax(k)+Bu(k)

y(k)=Cx(k)+Du(k)y(k) = C x(k) + D u(k)y(k)=Cx(k)+Du(k)

* + Where A=eAcTsA = e^{A\_c T\_s}A=eAc​Ts​ and B=(∫0TseAcτdτ)BcB = \left( \int\_0^{T\_s} e^{A\_c \tau} d\tau \right) B\_cB=(∫0Ts​​eAc​τdτ)Bc​.

##### 3. Define the Prediction Horizon and Control Horizon

**Objective:** Determine the future time steps over which predictions and optimizations are made.

1. **Prediction Horizon NNN:**
   * The length of the future period over which the system's behavior is predicted.
2. **Control Horizon MMM:**
   * The number of future control inputs to be optimized, typically M≤NM \leq NM≤N.

##### 4. Formulate the Cost Function

**Objective:** Define an objective function that quantifies the performance of the control actions.

1. **Quadratic Cost Function:**

J=∑i=0N−1[(y(k+i)−r(k+i))TQ(y(k+i)−r(k+i))+u(k+i)TRu(k+i)]J = \sum\_{i=0}^{N-1} \left[ (y(k+i) - r(k+i))^T Q (y(k+i) - r(k+i)) + u(k+i)^T R u(k+i) \right]J=i=0∑N−1​[(y(k+i)−r(k+i))TQ(y(k+i)−r(k+i))+u(k+i)TRu(k+i)]

* + Where r(k+i)r(k+i)r(k+i) is the reference trajectory, QQQ is the weight matrix for tracking errors, and RRR is the weight matrix for control efforts.

##### 5. Incorporate Constraints

**Objective:** Ensure the control actions and system states stay within feasible and safe limits.

1. **State Constraints:**

xmin⁡≤x(k+i)≤xmax⁡,i=1,…,Nx\_{\min} \leq x(k+i) \leq x\_{\max}, \quad i = 1, \ldots, Nxmin​≤x(k+i)≤xmax​,i=1,…,N

1. **Input Constraints:**

umin⁡≤u(k+i)≤umax⁡,i=0,…,M−1u\_{\min} \leq u(k+i) \leq u\_{\max}, \quad i = 0, \ldots, M-1umin​≤u(k+i)≤umax​,i=0,…,M−1

##### 6. Solve the Optimization Problem

**Objective:** Find the sequence of control inputs that minimize the cost function while satisfying constraints.

1. **Optimization Problem:**

min⁡{u(k),…,u(k+M−1)}J\min\_{\{u(k), \ldots, u(k+M-1)\}} J{u(k),…,u(k+M−1)}min​J

1. **Numerical Solvers:**
   * Use numerical optimization algorithms (e.g., quadratic programming) to solve the problem.

##### 7. Implement the Control Law

**Objective:** Apply the first control input from the optimized sequence and update the system state.

1. **Apply Control Input:**
   * Implement u(k)u(k)u(k), the first element of the optimized control sequence.
2. **Update State Estimate:**
   * Use state observers or estimators (e.g., Kalman filter) to update the state vector x(k)x(k)x(k) based on the new measurements.

##### 8. Receding Horizon Implementation

**Objective:** Continuously update the control inputs by repeating the optimization at each time step.

1. **Measure System States:**
   * Obtain new sensor readings and update the state vector x(k)x(k)x(k).
2. **Repeat Optimization:**
   * Re-solve the optimization problem with the updated state vector and apply the new control input.
3. **Implement Control Input:**
   * Apply the first control input from the new optimized sequence and continue the process.

##### Example of Implementation Steps in Pseudo-Code:

# Initialize system model parameters

A, B, C, D = get\_discrete\_model\_matrices()

x = initial\_state()

N, M = define\_horizons()

# Weight matrices for the cost function

Q = define\_state\_weight\_matrix()

R = define\_input\_weight\_matrix()

# Constraints

x\_min, x\_max = define\_state\_constraints()

u\_min, u\_max = define\_input\_constraints()

while control\_loop\_is\_active():

# Measure current state

x = measure\_current\_state()

# Define the optimization problem

def cost\_function(u\_seq):

J = 0

x\_pred = x

for i in range(N):

x\_pred = A @ x\_pred + B @ u\_seq[i]

y\_pred = C @ x\_pred + D @ u\_seq[i]

J += (y\_pred - r[i]).T @ Q @ (y\_pred - r[i]) + u\_seq[i].T @ R @ u\_seq[i]

return J

# Solve the optimization problem

u\_opt = solve\_optimization\_problem(cost\_function, constraints=(x\_min, x\_max, u\_min, u\_max))

# Apply the first control input

apply\_control\_input(u\_opt[0])

# Update state (for next iteration)

x = update\_state\_estimate(x, u\_opt[0])

##### Summary

Implementing an MPC algorithm in the discrete domain involves several steps, including system identification, model discretization, defining prediction and control horizons, formulating the cost function, incorporating constraints, solving the optimization problem, and implementing the control law in a receding horizon manner. By following these steps, MPC can achieve optimal control performance, ensuring the system operates efficiently and within safety limits.

#### Proposed Model Implementation as Equations in Discrete time Domain

The following section details the discrete-time implementation of the Model Predictive Control (MPC) algorithm for the automated irrigation system. This includes the formulation of the state-space model, cost function, constraints, and the optimization problem.

##### System Model

**State-Space Representation:**

The state-space model for our system is given by:

x(k+1)=Ax(k)+Bu(k)+Ed(k)x(k+1) = A x(k) + B u(k) + E d(k)x(k+1)=Ax(k)+Bu(k)+Ed(k)

y(k)=Cx(k)+Du(k)y(k) = C x(k) + D u(k)y(k)=Cx(k)+Du(k)

where:

* x(k)=[T(k)H(k)CO2(k)L(k)]x(k) = \begin{bmatrix} T(k) \\ H(k) \\ CO2(k) \\ L(k) \end{bmatrix}x(k)=⎣⎡​T(k)H(k)CO2(k)L(k)​⎦⎤​ is the state vector at time step kkk
* u(k)=[Qw(k)Pw(k)Lc(k)]u(k) = \begin{bmatrix} Q\_w(k) \\ P\_w(k) \\ L\_c(k) \end{bmatrix}u(k)=⎣⎡​Qw​(k)Pw​(k)Lc​(k)​⎦⎤​ is the control input vector
* y(k)=[T(k)H(k)CO2(k)L(k)]y(k) = \begin{bmatrix} T(k) \\ H(k) \\ CO2(k) \\ L(k) \end{bmatrix}y(k)=⎣⎡​T(k)H(k)CO2(k)L(k)​⎦⎤​ is the output vector
* AAA, BBB, EEE, CCC, and DDD are matrices defining the system dynamics

**Discrete-Time Matrices:**

Assume the continuous-time system has been discretized with a sampling time TsT\_sTs​. The discrete-time matrices AAA, BBB, EEE, CCC, and DDD are:

A=[1000010000100001],B=[β11β12β13β21β22β23β31β32β33β41β42β43],E=[ϵ1ϵ2ϵ3ϵ4],C=I4×4,D=04×3A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \beta\_{11} & \beta\_{12} & \beta\_{13} \\ \beta\_{21} & \beta\_{22} & \beta\_{23} \\ \beta\_{31} & \beta\_{32} & \beta\_{33} \\ \beta\_{41} & \beta\_{42} & \beta\_{43} \end{bmatrix}, \quad E = \begin{bmatrix} \epsilon\_{1} \\ \epsilon\_{2} \\ \epsilon\_{3} \\ \epsilon\_{4} \end{bmatrix}, \quad C = I\_{4 \times 4}, \quad D = 0\_{4 \times 3}A=⎣⎡​1000​0100​0010​0001​⎦⎤​,B=⎣⎡​β11​β21​β31​β41​​β12​β22​β32​β42​​β13​β23​β33​β43​​⎦⎤​,E=⎣⎡​ϵ1​ϵ2​ϵ3​ϵ4​​⎦⎤​,C=I4×4​,D=04×3​

##### Cost Function

**Quadratic Cost Function:**

The cost function JJJ to be minimized is:

J=∑i=0N−1[(y(k+i)−r(k+i))TQ(y(k+i)−r(k+i))+u(k+i)TRu(k+i)]J = \sum\_{i=0}^{N-1} \left[ (y(k+i) - r(k+i))^T Q (y(k+i) - r(k+i)) + u(k+i)^T R u(k+i) \right]J=i=0∑N−1​[(y(k+i)−r(k+i))TQ(y(k+i)−r(k+i))+u(k+i)TRu(k+i)]

where:

* r(k+i)r(k+i)r(k+i) is the reference trajectory for the outputs
* QQQ is the weight matrix for the tracking error
* RRR is the weight matrix for the control effort

##### Constraints

**State and Input Constraints:**

xmin⁡≤x(k+i)≤xmax⁡,i=1,…,Nx\_{\min} \leq x(k+i) \leq x\_{\max}, \quad i = 1, \ldots, Nxmin​≤x(k+i)≤xmax​,i=1,…,N

umin⁡≤u(k+i)≤umax⁡,i=0,…,M−1u\_{\min} \leq u(k+i) \leq u\_{\max}, \quad i = 0, \ldots, M-1umin​≤u(k+i)≤umax​,i=0,…,M−1

##### Optimization Problem

**Formulation:**

At each time step kkk, the optimization problem is to find the sequence of control inputs {u(k),u(k+1),…,u(k+M−1)}\{u(k), u(k+1), \ldots, u(k+M-1)\}{u(k),u(k+1),…,u(k+M−1)} that minimizes the cost function JJJ subject to the system dynamics and constraints.

**Optimization Problem:**

min⁡{u(k),…,u(k+M−1)}J\min\_{\{u(k), \ldots, u(k+M-1)\}} J{u(k),…,u(k+M−1)}min​J

subject to:

x(k+i+1)=Ax(k+i)+Bu(k+i)+Ed(k+i),i=0,…,N−1x(k+i+1) = A x(k+i) + B u(k+i) + E d(k+i), \quad i = 0, \ldots, N-1x(k+i+1)=Ax(k+i)+Bu(k+i)+Ed(k+i),i=0,…,N−1

y(k+i)=Cx(k+i)+Du(k+i),i=0,…,N−1y(k+i) = C x(k+i) + D u(k+i), \quad i = 0, \ldots, N-1y(k+i)=Cx(k+i)+Du(k+i),i=0,…,N−1

xmin⁡≤x(k+i)≤xmax⁡,i=1,…,Nx\_{\min} \leq x(k+i) \leq x\_{\max}, \quad i = 1, \ldots, Nxmin​≤x(k+i)≤xmax​,i=1,…,N

umin⁡≤u(k+i)≤umax⁡,i=0,…,M−1u\_{\min} \leq u(k+i) \leq u\_{\max}, \quad i = 0, \ldots, M-1umin​≤u(k+i)≤umax​,i=0,…,M−1

#### Proposed Model Implementation in MicroPython

Below is the detailed implementation of the MPC algorithm in Python for the proposed system. This implementation involves defining the system model, setting up the optimization problem, and running the MPC in a control loop. The implementation assumes that the necessary libraries for numerical optimization (e.g., scipy.optimize) are available.

##### Step 1: Define the System Model

First, define the system model using the state-space representation.

import numpy as np

from scipy.optimize import minimize

# System parameters

A = np.eye(4) # State transition matrix (assuming identity for simplicity)

B = np.array([[0.1, 0.0, 0.0],

[0.0, 0.1, 0.0],

[0.0, 0.0, 0.1],

[0.1, 0.1, 0.1]]) # Control input matrix

C = np.eye(4) # Output matrix

D = np.zeros((4, 3)) # Feedthrough matrix

# Initial state

x0 = np.array([20.0, 50.0, 400.0, 300.0]) # Example initial values for T, H, CO2, L

# Prediction and control horizons

N = 10 # Prediction horizon

M = 3 # Control horizon

# Weight matrices

Q = np.eye(4) # State error weight matrix

R = np.eye(3) # Control effort weight matrix

# Constraints

x\_min = np.array([15.0, 30.0, 300.0, 200.0]) # Minimum state constraints

x\_max = np.array([30.0, 70.0, 600.0, 800.0]) # Maximum state constraints

u\_min = np.array([0.0, 0.0, 0.0]) # Minimum control input constraints

u\_max = np.array([10.0, 10.0, 100.0]) # Maximum control input constraints

##### Step 2: Define the Cost Function

The cost function is defined to minimize the deviation from the reference trajectory and the control effort.

def cost\_function(u\_flat, x0, N, Q, R, A, B, C, r):

u = u\_flat.reshape(N, -1) # Reshape control sequence

x = np.copy(x0)

cost = 0.0

for i in range(N):

x = A @ x + B @ u[i]

y = C @ x

cost += (y - r[i]).T @ Q @ (y - r[i]) + u[i].T @ R @ u[i]

return cost

# Reference trajectory

r = np.array([[25.0, 50.0, 400.0, 500.0]] \* N) # Example reference for T, H, CO2, L

##### Step 3: Solve the Optimization Problem

Use a numerical optimizer to solve the control problem at each time step.

def solve\_mpc(x0, r, N, M, Q, R, A, B, C, x\_min, x\_max, u\_min, u\_max):

u0 = np.zeros((N, B.shape[1])) # Initial guess for control inputs

# Constraints for the optimizer

bounds = [(u\_min[i], u\_max[i]) for \_ in range(N) for i in range(B.shape[1])]

# Objective function for the optimizer

objective = lambda u\_flat: cost\_function(u\_flat, x0, N, Q, R, A, B, C, r)

# Solve the optimization problem

result = minimize(objective, u0.flatten(), bounds=bounds)

# Extract the optimal control sequence

u\_opt = result.x.reshape(N, -1)

return u\_opt[0] # Return only the first control input

# Example usage

u\_opt = solve\_mpc(x0, r, N, M, Q, R, A, B, C, x\_min, x\_max, u\_min, u\_max)

print("Optimal control input:", u\_opt)

##### Step 4: Implement the Control Loop

Run the MPC in a control loop, updating the state at each step.

def mpc\_control\_loop(x0, r, N, M, Q, R, A, B, C, x\_min, x\_max, u\_min, u\_max, steps):

x = np.copy(x0)

control\_inputs = []

states = [x0]

for \_ in range(steps):

u\_opt = solve\_mpc(x, r, N, M, Q, R, A, B, C, x\_min, x\_max, u\_min, u\_max)

x = A @ x + B @ u\_opt # Update state with the optimal control input

control\_inputs.append(u\_opt)

states.append(x)

# Here you would also update the actual system (e.g., hardware control)

return states, control\_inputs

# Run the control loop for a specified number of steps

steps = 20

states, control\_inputs = mpc\_control\_loop(x0, r, N, M, Q, R, A, B, C, x\_min, x\_max, u\_min, u\_max, steps)

print("States:", states)

print("Control inputs:", control\_inputs)

##### Summary of Implementation Steps

1. **Define the System Model:**
   * Specify the state-space representation with matrices A,B,C,DA, B, C, DA,B,C,D.
   * Set initial state values and horizons for prediction and control.
2. **Define the Cost Function:**
   * Create a cost function to minimize the tracking error and control effort.
3. **Solve the Optimization Problem:**
   * Use a numerical optimizer (e.g., scipy.optimize.minimize) to find the optimal control inputs subject to constraints.
4. **Implement the Control Loop:**
   * Run the MPC algorithm in a control loop, updating the state at each step and applying the optimal control input.

This implementation demonstrates a basic setup for an MPC algorithm in Python, tailored for the proposed automated irrigation system. Further refinements may include more accurate modeling of the system dynamics, handling disturbances, and integrating with actual hardware interfaces for real-time control.

# Chapter 4: Conclusion

# References

[1] A. A. Desai, R. A. Metri, S. R. Patil, A. A. Nagargoje and D. S. Desai, "Automated Irrigation System for Efficient and Portable Farming," 2023 International Conference on Power, Instrumentation, Control and Computing (PICC), Thrissur, India, 2023, pp. 1-6, doi: 10.1109/PICC57976.2023.10142376.

[2] Levidow, L., Zaccaria, D., Maia, R., Vivas, E., Todorovic, M., & Scardigno, A. (2014). Improving water-efficient irrigation: Prospects and difficulties of innovative practices. Agricultural Water Management, 146, 84-94. doi:10.1016/j.agwat.2014.07.012

[3] Lozoya, C., Mendoza, C., Mejía, L., Quintana, J., Mendoza, G., Bustillos, M., Arras, O., & Solís, L. (2014). Model predictive control for closed-loop irrigation. Proceedings of the 19th World Congress The International Federation of Automatic Control, Cape Town, South Africa, August 24-29, 2014. doi:10.3182/20140824-6-ZA-1003.01920

[4] Saleem, K., Delgoda, D., Ooi, S. K., Dassanayake, K. B., Liu, L., Halgamuge, M. N., & Malano, H. (2013). Model predictive control for real-time irrigation scheduling. Retrieved from [ResearchGate](https://www.researchgate.net/publication/260418442).

[5] Ahmed, Z., Gui, D., Murtaza, G., Yunfei, L., & Ali, S. (2023). An Overview of Smart Irrigation Management for Improving Water Productivity under Climate Change in Drylands. Agronomy, 13(8), 2113. https://doi.org/10.3390/agronomy13082113

[6] Metri, R. A., & Rajpathak, B. A. (2022). Computation of Control Law for State Transfer Problem in Efficient Way for a Single Input. IEEE Transactions on Industry Applications. doi:10.1109/TIA.2022.3159502

[7] Khairy, M. F. A., Elmesery, A. A. M., Zabady, F. E., & Fayed, M. H. (2016). Design and Evaluation of Mobile Drip Irrigation System. Misr Journal of Agricultural Engineering, 33(4), 1329-1350.

[8] Delgoda, D., Malano, H., Saleem, S. K., & Halgamuge, M. N. (2016). Irrigation Control Based on Model Predictive Control (MPC): Formulation of Theory and Validation Using Weather Forecast Data and AQUACROP Model. Environmental Modelling & Software, 78, 40-53. doi:10.1016/j.envsoft.2015.12.012

[9] Gamal, Y., Soltan, A., Said, L. A., Madian, A. H., & Radwan, A. G. (2023). Smart Irrigation Systems: Overview. IEEE Access. doi:10.1109/ACCESS.2023.3251655

[10] Babu, D. V., Shijin, K. V., Sreejith, N. S., Anjana, V. S., & Karthikeyan, C. (2020). Automatic Irrigation Systems for Efficient Usage of Water using Embedded Control Systems. IOP Conference Series: Materials Science and Engineering, 993, 012077. doi:10.1088/1757-899X/993/1/012077