Smart Irrigation Robot

2019 - 2023

Project Code: 14

Internal Advisor: Mr **Azhar** Mushtaq
Mr Waseem Ahmad

Project Manager: Mr **Fahad Maqbool**

Project Team:

Rida Tahir (Leader) **BITF19M038**
Anam Shahzadi **BITF19M009**
Areej Fatima **BITF19M027**

Submission Date: 15 June, 2023

Document Information

Category	Information
Customer	CS & IT, UOS
Project	Smart irrigation robot
Document	Requirement Specifications
Document Version	1.0
Identifier	PGBH01-2020-RS
Status	Draft
Author(s)	Rida Tahir **Leader** Areej Fatima Anam Shehzadi
Approver(s)	PM
Issue Date	
Document Location	fyp
Distribution	1. Advisor 2. PM 3. Project Office

Definition of Terms, Acronyms and Abbreviations

This section should provide the definitions of all terms, acronyms, and abbreviations required to interpret the terms used in the document properly.

Term	Description
ASP	Active Server Pages
RS	Requirements Specifications

Table of Contents

Chapter 1 PROJECT PROPOSAL

Chapter 2 SRS

1. Abstract

The rapid advancement of robotics and automation technologies has opened up new possibilities for improving agricultural practices. This abstract presents a smart irrigation robot, a novel solution designed to enhance the efficiency and sustainability of water management in agriculture. By leveraging a combination of sensors, actuators, and intelligent algorithms, the robot autonomously monitors and optimizes irrigation processes, reducing water waste and maximizing crop yields

The smart irrigation robot utilizes a network of soil moisture sensors to collect real-time data on soil moisture levels, enabling precise and targeted irrigation. Through an integrated irrigation system, the robot can accurately deliver water to specific areas of the field based on the detected moisture levels, ensuring that crops receive optimal hydration without overwatering. Additionally, the robot employs weather monitoring sensors to consider external factors such as rainfall and humidity, adjusting irrigation schedules accordingly

To optimize water distribution, the smart irrigation robot employs machine learning algorithms and data analytics. By continuously analyzing the collected data, the robot can learn and adapt its irrigation patterns to suit the specific needs of different crops and soil types. The integration of machine learning also enables the robot to make predictions and recommendations for future irrigation cycles, improving water efficiency and reducing manual intervention

The smart irrigation robot offers several advantages over traditional irrigation methods. By automating the irrigation process, it minimizes human errors and inconsistencies, ensuring uniform water distribution throughout the field. Moreover, the robot operates autonomously, reducing labor requirements and freeing up farmers' time for other essential tasks. By conserving water resources and minimizing unnecessary water usage, the robot contributes to the overall sustainability of agricultural practices

2. Background and Justification

Background:

Irrigation is a vital component of agriculture, enabling farmers to provide crops with the necessary water for optimal growth and yield. However, traditional irrigation practices often suffer from inefficiencies, leading to water wastage, increased costs, and environmental concerns. With the advancement of technology, there is a growing need for innovative solutions that can enhance the efficiency and sustainability of agricultural water management.

Justification:

- **Water Conservation**:
  Water scarcity is a pressing global issue, and agriculture is one of the largest consumers of freshwater resources. Traditional irrigation methods often result in overwatering or uneven distribution, leading to water wastage. By implementing a smart irrigation robot, water usage can be optimized by precisely delivering water where and when it is needed, reducing overall water consumption and conserving this valuable resource.

- **Improved Crop Yields:**
  Proper irrigation is crucial for maximizing crop productivity. Under or over-irrigation can have detrimental effects on plant health and yield. The smart irrigation robot utilizes real-time soil moisture data and weather information to deliver the right amount of water to crops, ensuring optimal hydration. This targeted approach promotes healthier plant growth, minimizes stress, and increases crop yields.
- **Precision Farming**:
  The integration of robotics and automation in agriculture, such as smart irrigation robots, enables precision farming. By employing sensors and intelligent algorithms, these robots can monitor and analyze various environmental factors, including soil moisture, temperature, and weather conditions. This data-driven approach allows farmers to make informed decisions and take proactive measures to ensure crop health and productivity.
- **Labor Efficiency:**
  Manual irrigation methods often require significant labor inputs, with farmers spending considerable time and effort in irrigation activities. By deploying smart irrigation robots, the labor-intensive process of manually watering crops can be reduced or eliminated. This allows farmers to allocate their time and resources more effectively, focusing on other essential farming tasks, such as pest control, crop monitoring, and overall farm management
- **Environmental Impact**:
  Smart irrigation robots contribute to sustainable agriculture practices by minimizing the environmental impact associated with excessive water usage and chemical runoff. By optimizing water delivery and reducing the need for excessive fertilizers, pesticides, and herbicides, these robots help mitigate water pollution, soil degradation, and negative impacts on surrounding ecosystems
- **Cost Reduction:**
  Traditional irrigation methods can be costly due to the expenses associated with water usage, energy consumption, and labor. Smart irrigation robots once implemented, can provide cost savings by optimizing water usage, reducing energy requirements, and minimizing labor inputs. These cost reductions can contribute to the economic viability and long-term sustainability of farming operations

3. Project Methodology

The objective of this project is to provide a combination of manual supervision and partial automation and is similar to manual set up in most respect but it reduces the labor involved in terms of irrigation design is simple, easy to install, micro controller based circuit to monitor and record the values of temperature, soil moisture that are continuously modified and controlled in order optimize them to achieve maximum plant growth and yield. Also the use of easily available components reduces the manufacturing and the design is quite flexible as the software can be changed any time. It can thus be made to the specific requirements of the user. This makes the proposed system to be an economical, portable and a low maintenance solution for greenhouse applications, especially in rural areas and for small scale agriculture.

1) Identify requirements and objectives:

- Define the specific requirements and objectives of the smart irrigation robot project, considering factors such as the target crops, field size, environmental conditions, and water management goals
- Determine the desired features and capabilities of the robot, including sensors, actuators, data analytics, and automation

2) Research and Design:

- Conduct a comprehensive review of existing smart irrigation systems, robotic technologies, and related research papers to gather insights and identify potential design approaches
- Design the architecture and components of the smart irrigation robot, considering factors such as sensor integration, communication systems, power management, and the robot's physical structure

3) **Sensor integration**:

- Select and integrate appropriate sensors for collecting data on soil moisture, temperature, humidity, and weather conditions
- Implement sensor calibration procedures to ensure accurate and reliable data collection
- Develop algorithms for sensor data fusion and interpretation.

4) Actuator and Automation System:

- Determine the appropriate actuators, such as valves or pumps, for delivering water to the crops
- Develop an automation system that enables the robot to control and adjust irrigation processes based on the sensor data and predefined irrigation schedules
- Implement algorithms for precise water delivery, considering factors such as crop water requirements, soil moisture levels, and weather conditions

5) Data Analytics and Machine Learning

- Design and implement data analytics algorithms to process and analyze the collected sensor data
- Develop machine learning models to predict crop water needs, optimize irrigation schedules, and improve water distribution accuracy over time.
- Incorporate algorithms for anomaly detection and fault diagnosis to ensure the system's robustness and reliability

6) Prototype Development:

- Build a functional prototype of the smart irrigation robot based on the designed architecture and components
- Test and validate the prototype in controlled environments, such as laboratory or small-scale field trials
- Collect performance data and fine-tune the system parameters based on the test results

7) Field Testing and Optimization:

- Deploy the smart irrigation robot in real-world field conditions
- Monitor and evaluate the robot's performance, including water delivery accuracy, energy efficiency, and overall system reliability

8) **Documentation and Knowledge Transfer**:

- Document the design, implementation, and testing processes, creating technical documentation and user manuals for future reference
- Share the knowledge and experience gained from the project through research papers, conference presentations, and open-source contributions

4. Project Scope

- The project aim is to detect the dryness in soil using sensors and provide water to the plants appropriately.
- This project helps to maintain the plants quite easily.
- In this project we are detecting soil moisture and need for Irrigation.
- Increase in productivity
- reduced water consumption
- Safe
- No manpower required
- Reduce soil erosion and nutrient leaching
- Require smaller water sources
- Water Conservation
- Real Time data give
- Lowered Operation's Costs
- Efficient and saves time
- Reduce soil erosion

5. High level Project Plan

i. Project initiation phase

- Define project scope, objectives, and success criteria.
- Identify key stakeholders and establish communication channels.
- Allocate project resources, including budget, team members, and required equipment

ii. Requirement Gathering and Analysis Phase

- Conduct a thorough analysis of water management requirements, environmental factors, and specific needs of the target crops
- Determine the desired features and functionalities of the smart irrigation robot
- Document the requirements in a comprehensive specification document

iii. Design and Development Phase

- Develop the overall architecture and system design for the smart irrigation robot
- Select and integrate the necessary sensors, actuators, and communication systems
- Implement algorithms for data collection, processing, and irrigation control
- Build a prototype of the smart irrigation robot and perform initial testing and validation

iv. Testing and Optimization Phase

- Evaluate the performance of the robot in terms of water delivery accuracy, energy efficiency, and system reliability
- Identify and address any issues or shortcomings through iterative refinement and optimization

v. Field Testing and Validation Phase

- Deploy the smart irrigation robot in real-world agricultural fields
- Monitor and evaluate the robot's performance in different farming conditions, considering factors such as field size, soil composition, and crop variability
- Gather feedback from farmers and agronomists to assess user satisfaction and identify areas for improvement

vi. Documentation and Reporting Phase

- Prepare comprehensive documentation, including technical specifications, user manuals, and system operation guidelines
- Compile project reports summarizing the development process, challenges faced, and lessons learned
- Present project findings and outcomes to key stakeholders, industry experts, and relevant forums

vii. Deployment and Maintenance Phase

- Plan and execute the deployment strategy for widespread adoption of the smart irrigation robot
- Provide training and support to farmers and users for efficient operation and maintenance of the system
- Establish a maintenance and support framework to ensure continuous functionality and timely troubleshooting

viii. Monitoring and Evaluation Phase

- Continuously monitor the performance of the smart irrigation robot in the field, gathering data on water savings, crop yield improvements, and environmental impact
- Analyze the collected data and conduct periodic evaluations to measure the effectiveness of the system
- Incorporate feedback from users and stakeholders for further optimization and future enhancements

CHAPTER 2
SRS & FS

1. Introduction

The IoT and artificial intelligence technologies are the most viable alternative solution for traditional yield prediction and irrigation because of the remote accessibility, interoperability, cognitive capability, etc. These technologies aid the farmers in monitoring agricultural parameters remotely from embedded devices, drastically reducing human efforts. It makes farming much efficient and a smarter

1.1 Purpose of Document

The purpose of a plant watering robot using Arduino is to automate the process of watering plants. The robot is designed to detect when the soil is dry and then water the plant accordingly. The Arduino is used as the brain of the robot, which controls the various sensors and actuators.
The plant watering robot consists of a water tank, a pump, a moisture sensor, and an Arduino board. The moisture sensor is used to detect the moisture level of the soil and triggers the pump to water the plant when the soil becomes too dry. The water tank provides a steady supply of water for the robot to use.
The robot can be programmed to water the plants at specific times, and it can also be set to water the plants for a specific duration. This makes it possible to ensure that the plants are watered at the right time and in the right amount.
The advantages of using a plant watering robot are that it can save time and reduce water usage while promoting plant health. It is an eco-friendly way to water plants, and it can be used in various settings, such as homes, gardens, and greenhouses.

1.2 Project Overview

Our proposed system uses a robot with an Arduino UNO microcontroller. This robot is powered by Arduino microcontroller it consists of robotic chassis, wheels, pipe, water pump, Electronic Circuit, , and ultrasonic sensor. This robot can be operated in an autonomous mode in which robot will move automatically and it will detect obstacle using ultrasonic sensors. In this automatic plant watering system is being used by keeping the water pump on. This robot is made up of high quality material, it has motors with rough terrain wheel which can take full 360-degree rotation from both sides as it moves in front and backward direction.
Alogirthm
Smart irrigation robot built using **Decision tree** algorithm which is machine learning algorithm that trains system on a part of collected data. The system is **self-learning neural network**, which depend on reading of detector of soil wetness. **KNN** (k-nearest neighbor) algorithm also deployed.

1.3 Scope

Home gardening: A plant watering robot can be a useful tool for home gardeners who want to automate their watering process. The robot can be programmed to water plants at specific intervals, ensuring that plants receive the optimal amount of water.

Agriculture: In large-scale agriculture, a plant watering robot can save time and labor costs by automating the watering process. The robot can be programmed to water crops at specific times and in specific quantities, ensuring that plants receive the necessary amount of water.

Greenhouses: A plant watering robot can be used in greenhouses to maintain optimal moisture levels for plants. The robot can be programmed to water plants based on factors such as temperature and humidity, ensuring that plants receive the right amount of water.

Education: A plant watering robot can be used as an educational tool to teach students about robotics, programming, and plant biology. It can be used as a hands-on project to engage students in STEM (science, technology, engineering, and math) learning.

Accessibility: A plant watering robot can be especially helpful for individuals with physical disabilities or limited mobility who may find it challenging to water plants manually.

- No manpower required
- Reduce soil erosion and nutrient leaching
- Require smaller water sources
- Real Time data give
- Lowered Operation's Costs

2. Overall System Description

The **anticipated users** of the plant watering robot using Arduino could include individuals who want to automate their gardening process, people who have difficulty with mobility or access to their plants, or individuals who want to reduce the amount of time they spend on plant care. These users may range from hobbyists to professional gardeners.

There are known **constraints** to consider when developing the plant watering robot using Arduino. For example, the robot will need to be able to accurately measure and dispense water while minimizing water waste. The robot will also need to be energy efficient and have a long battery life or a reliable power source. Additionally, the robot should be designed to be user-friendly, easy to set up, and customizable for different plant types and environments.

Assumptions for the plant watering robot using Arduino include that the user has access to a smartphone or computer to control the robot and that the plants being watered are in containers or pots rather than planted directly in the ground. The robot will also need to be weather-resistant if used outdoors and able to withstand various environmental conditions, such as heat, humidity, and moisture.

Dependencies for the plant watering robot using Arduino include the availability of compatible sensors and components for measuring soil moisture, water levels, and other relevant data. The robot may also depend on a stable internet connection for remote control or data logging. Additionally, the development of the robot may require expertise in programming, electronics, and engineering.

Here is a list of things a plant watering robot using Arduino will do:

Measure soil moisture levels: The robot will use sensors to measure the moisture levels in the soil, which will determine whether or not the plants need to be watered.

Dispense water: If the soil moisture levels are below the threshold, the robot will turn on the water pump and dispense water to the plants.

Monitor water levels: The robot will monitor the water levels in the reservoir or supply and alert the user when it's time to refill.

Provide automation: The robot will automate the plant watering process, reducing the need for manual watering and ensuring that plants receive the optimal amount of water.

Here is a list of things a plant watering robot using Arduino will not do:

Prune or care for plants: The robot is designed only to water plants and will not prune or care for them in other ways.

Plant seeds: The robot is not designed to plant seeds, and it cannot replace the initial planting process.

Adjust lighting or temperature: The robot does not have the ability to adjust lighting or temperature, which can also affect plant growth.

Move around: The robot is stationary and cannot move around to water plants in different locations.

2.1 User characteristics

Based on the functionality of a plant watering robot using Arduino, the following user classes can be anticipated:

Homeowners/Gardeners: These users are likely to be the primary user class for the plant watering robot using Arduino. They will use the robot to automate the process of watering their plants at home or in their gardens.

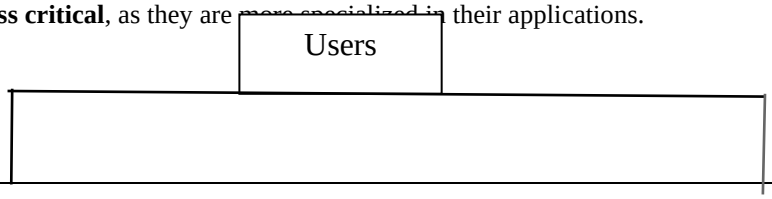
Agricultural Farmers: Farmers who have small farms or nurseries can also use this robot to automate watering their crops. This would allow them to reduce labor costs and increase efficiency.

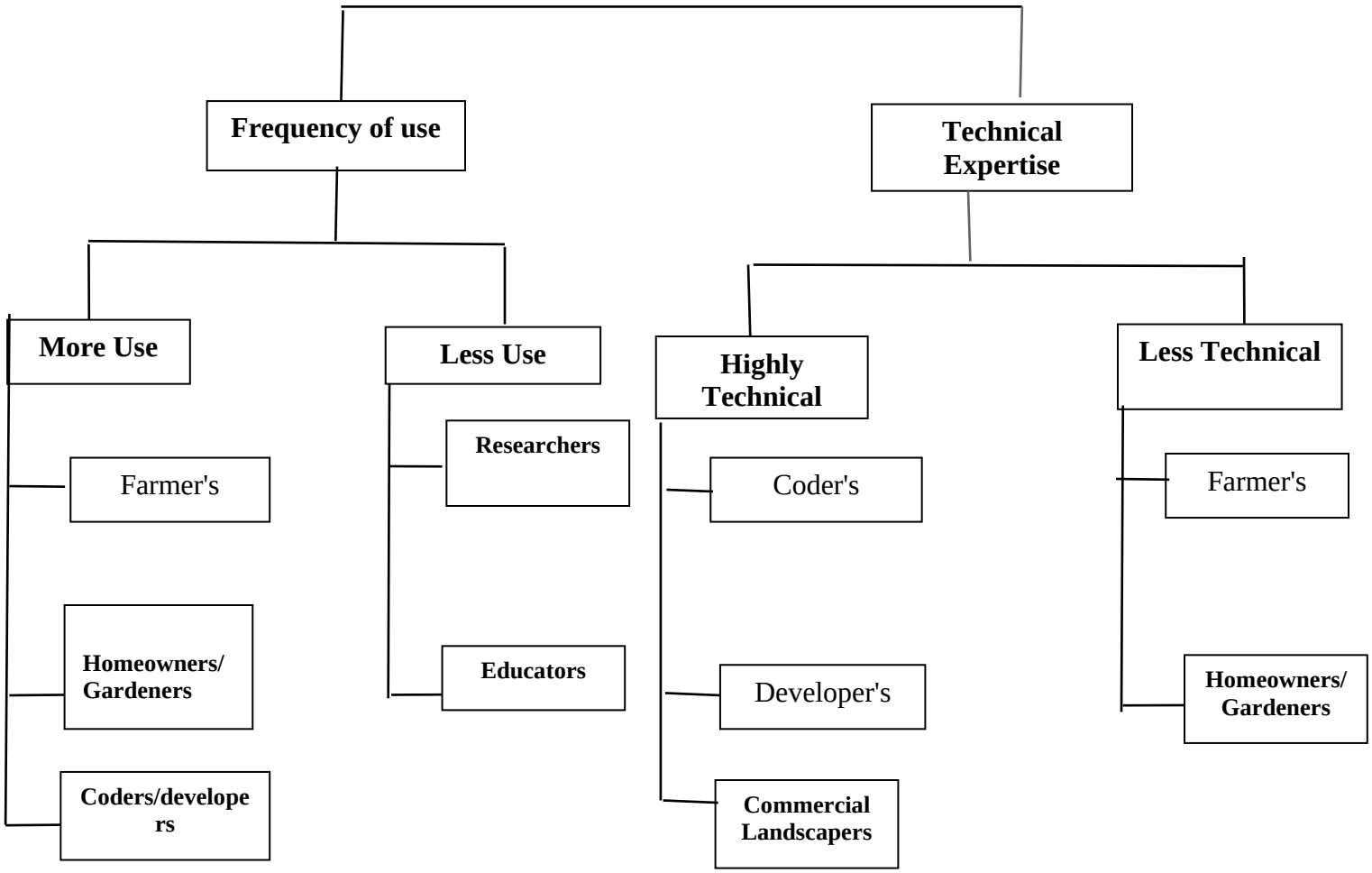
Researchers: Researchers in plant sciences or robotics could use the robot to study and analyze plant growth and irrigation systems.

Educators: The plant watering robot using Arduino can be used by educators to teach students about robotics, programming, and plant care.

Commercial Landscapers: Landscapers who manage large properties could use the robot to automate watering for a more efficient and cost-effective operation.

Out of the above user classes, **homeowners/gardeners** and **agricultural farmers** are the **most critical** ones as they are the primary users who would benefit the most from the plant watering robot using Arduino. The other user classes are **less critical**, as they are more specialized in their applications.

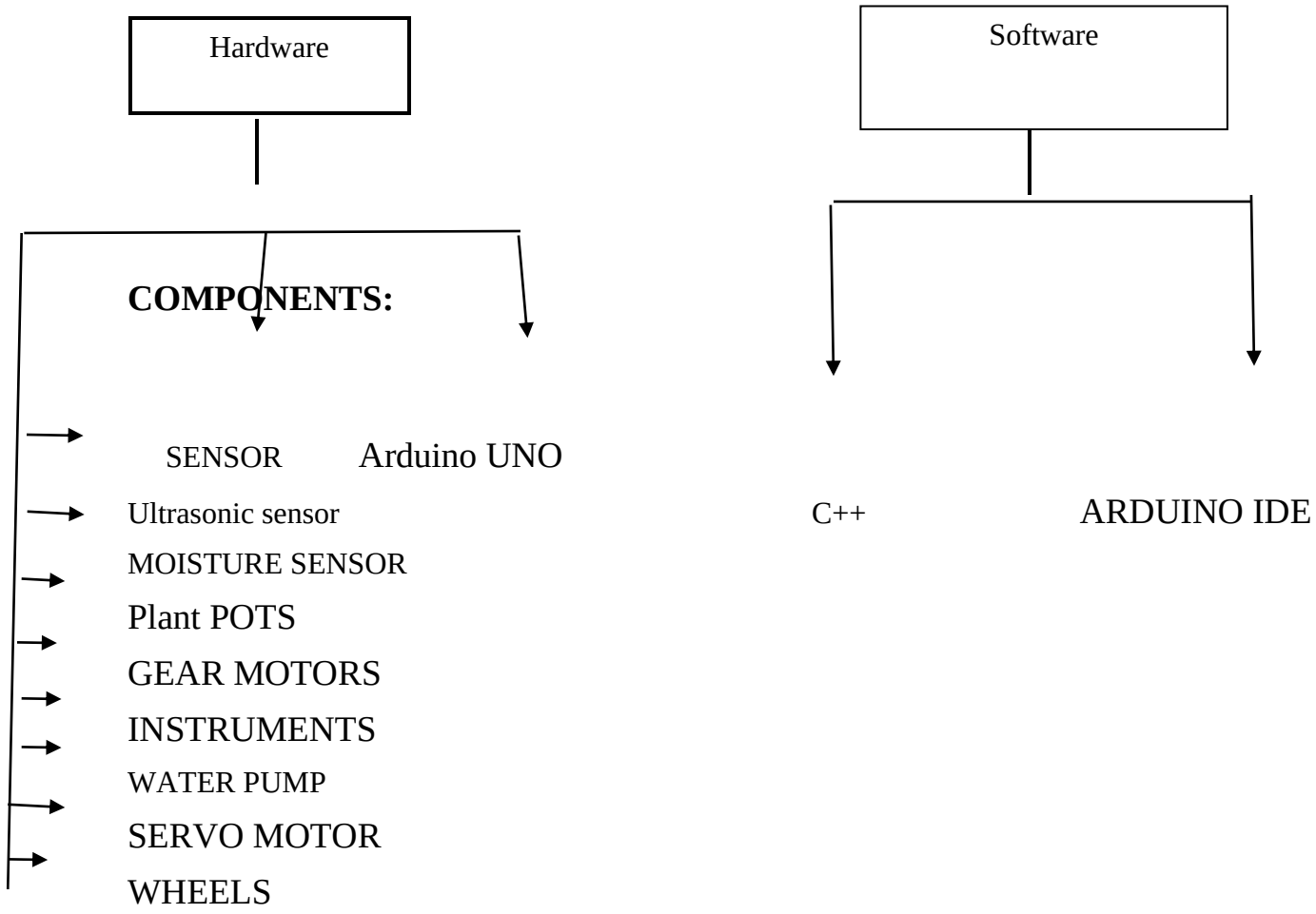




2.2 Operating environment

The plant watering robot using Arduino software will operate in an indoor or outdoor environment, depending on the location of the plants. The hardware platform will include an Arduino microcontroller, a water pump, a moisture sensor, and a water container to deliver water to the plants.

The operating system for the plant watering robot will be the Arduino software environment, which is based on the C++ programming language. This software will be used to write the code that controls the robot's actions, such as turning on and off the water pump **and movement of robot** based on the readings from the moisture sensor.



2.3 System constraints

Limitations of this project is

1. The system can only be used via internet connection.
2. The system can be used with the help of batteries on the field where AC current is not available

6. Software constraints

If you are developing a plant watering robot using Arduino, you may need to consider the following software constraints:

Motor Control: The robot will require motors to move around and control the watering mechanism. We use Arduino software that can control the motors and provide the necessary torque to move the robot around.

Sensor Integration: The robot should be able to sense the moisture level of the soil and take appropriate action based on the readings. We use C++ coding that can read sensor data and respond accordingly.

Power Management: The robot will need to be powered by batteries, so we have sufficient batteries that can bear the load of the robot monitor the battery level and take appropriate action.

Calibration: The robot is calibrated to ensure accurate readings from the sensors and precise movements from the motors. The robot can calibrate and fine-tune its parameters.

7. Hardware constraints

For plant watering robot using Arduino, we need to consider the following hardware constraints:

Power Supply: The robot will require a power source, for that we choose the appropriate batteries of 5v or power supply to provide sufficient power for the robot to function.

Motor Selection: The robot will require motors to move around and control the watering mechanism. We choose the appropriate motors that are based on the robot's size, weight, and power requirements.

Sensor Selection: The robot should be able to sense the moisture level of the soil and take appropriate action based on the readings. So, the sensors are based on their accuracy, reliability, and power requirements.

Watering Mechanism: The robot will need to have a mechanism to water the plants. We choose the appropriate mechanism water container that based on the plant's size and watering requirements.

Chassis and Wheels: The robot will require a chassis and wheels to move around. The appropriate chassis and wheels are added based on the robot's size and weight.

Arduino Board: We choose Arduino UNO board which based on the robot's requirements. Some boards have more input/output pins, more memory, or faster processors.

Connectors and Cables: Appropriate connectors and cables to connect the various components of the robot together.

Enclosure: The robot will require an enclosure to protect the electronics and components from the environment. So, we enclosure the robot's size and the environment it will be operating in.

8. Cultural constraints (includes language etc.)

Cultural beliefs: Some cultures may have different beliefs about how plants should be watered or cared for. For example, some cultures believe that plants should only be watered in the morning or evening. You may need to consider these beliefs when designing the robot's watering schedule.

Environmental factors: Different regions may have different environmental factors that can impact plant growth and watering needs. For example, some regions may have more rainfall than others, and the robot may need to adjust its watering schedule accordingly.

Design preferences: Different cultures may have different design preferences or aesthetics. For example, some cultures may prefer more ornate or decorative designs, while others may prefer more minimalist designs.

User expectations: User expectations may vary based on cultural factors. For example, some cultures may expect the robot to be more autonomous and require less user input, while others may prefer more control over the robot's actions.

Legal constraints

Legal constraints generally refer to an outdated and insufficient legal framework that works against irrigation operations.

.

Environmental regulations: The robot should comply with environmental regulations related to water usage and waste disposal. For example, it should not use excessive water, and any waste generated by the robot should be disposed of properly.

Liability: You should consider the potential liability issues related to the robot's operation. For example, if the robot malfunctions and causes damage to the plants or property, who is liable for the damages?

Privacy: If the robot collects data about the plants or the environment, you should ensure that the data is collected and used in compliance with privacy regulations.

Import/export regulations: If you plan to sell the robot internationally, you should ensure that it complies with import/export regulations related to electronics and robotics.

- **Environmental constraints**

Moisture and humidity: The robot will be operating in a moist environment, so you will need to ensure that the electronics and other components are protected from moisture and humidity.

Temperature: The robot should be able to operate in a wide range of temperatures, from cold nights to hot days, depending on the environment it will be operating in.

Terrain: The robot will need to be able to navigate over different types of terrain, such as grass, rocks, or soil.

Water quality: The quality of the water in the environment may affect the operation of the robot. For example, if the water is highly acidic, it may corrode the robot's components.

Power supply: The availability of a reliable power supply in the environment may affect the design of the robot. For example, if the robot will be operating in a remote location without access to electricity, you may need to use solar power or other alternative power sources.

User constraints

Age: If the project is being developed for children, you may need to consider the age range of the target audience. The user interface and controls may need to be simplified or made more intuitive to accommodate younger users.

Cognitive abilities: Users with cognitive disabilities or limitations may have difficulty understanding complex instructions. You may need to simplify the interface and provide more visual cues to accommodate different cognitive abilities.

Familiarity with technology: Users with limited familiarity with technology may have difficulty using complex controls. You may need to provide more guidance or instructional materials to help users get started with the robot.

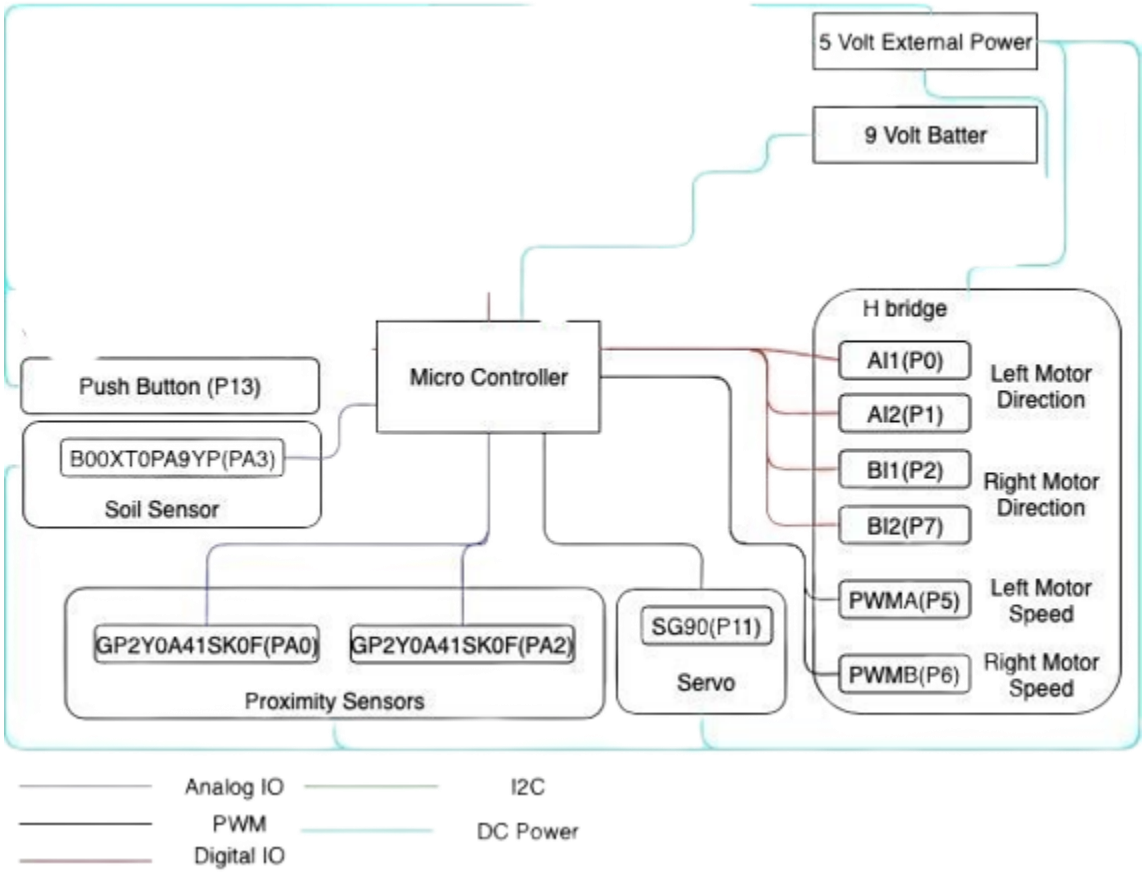
PROFESSIONAL AND ETHICAL CONSTRAINTS

Professionally talking the real time implementation of the project to the field requires expertise in the electrical and its related subsequent areas such as IOT , Embedded and a bit of programming in order to program the microcontroller to control the entire smart irrigation system.
Ethically talking anything done with good and humane intention and gesture and which can be used for humanely purposes for making the tasks easily accomplished is consider- erred Ethically Affirmative in in its very consideration.

Technical constraints

Include the poor state of irrigation infrastructure, the lack of infrastructure maintenance, inadequate skills and knowledge of utilizing irrigation systems, and poor irrigation management and service delivery.

3. External Interface Requirements



3.1 Hardware Interfaces

The plant watering robot using Arduino has several interfaces between its software and hardware components. These interfaces include the following:

Sensor Interface: The sensor interface is used to communicate between the software and hardware components of the robot. It supports different types of sensors such as moisture sensors, temperature sensors, and humidity sensors. The data from these sensors is transmitted to the software, which uses it to determine when the plants need watering.

Actuator Interface: The actuator interface is used to control the hardware components of the robot. It supports different types of actuators such as pumps, motors, and valves. The software sends commands to the actuator interface, which then controls the actuators to water the plants or perform other actions as needed.

In the loop function, the Blynk library is run

The program in the Arduino reads the moisture value from the sensor every 20 seconds. If the value reaches the threshold value, the program does the following three things:

- It moves the servo motor , along with the water pipe fixed on it, toward the potted plant, whose moisture level is less than the predetermined/ threshold level.
- It starts the motor pump to supply water to the plant for a fixed period of time and then stops the water pump.
- It brings back the servo motor to its initial position.

3.2 Communications Interfaces

Physical Buttons: Basic plant watering robots may feature physical buttons interface directly on the robot itself. Users can navigate through menus and adjust settings by interacting with these physical interfaces.

Sensor Integration: Plant watering robots equipped with various sensors, such as moisture sensors or light sensors, may communicate their data through specific interfaces. This can be done through analog or digital interfaces, allowing the robot to relay information about soil moisture levels or environmental conditions.

4. Functional Requirements

Plant Setup: The customer needs to set up the robot for their plant by providing relevant information such as plant species, age, watering schedule, and soil moisture level. The robot must have an easy-to-use interface to input this information and store it for future reference.

Soil Moisture Monitoring: The robot needs to monitor the soil moisture level of the plant(s) and notify the customer when it falls below a certain threshold. The robot must have a moisture sensor and an algorithm to determine the optimal moisture level for the plant(s) based on their species and age.

Watering: The robot needs to water the plant(s) when the soil moisture level falls below the threshold. The robot must have a water delivery system, such as a pump or a valve, that can dispense the right amount of water based on the plant(s)' watering schedule and the moisture level.

Remote Access: The customer needs to access the robot and its data remotely, using a web or mobile application. The robot must have a network connection, such as Wi-Fi or Bluetooth, and an API or web service that can expose its functionality and data to the customer's application.

Data Storage: The robot needs to store the relevant data related to the plant(s), such as species, age, watering schedule, and moisture level. The robot must have a database or storage system that can store and retrieve this data efficiently and securely.

Security: The robot needs to ensure the security and privacy of the customer's data and communication channels. The robot must have encryption and authentication mechanisms to prevent unauthorized access and ensure data confidentiality.

5. Non-functional Requirements

5.1 Performance Requirements

The following are the key performance characteristics that should be considered:

Speed: The robot must be able to perform its tasks, such as soil moisture monitoring and watering, in a timely manner, without causing any delay or disruption to the plant care process. The speed of the robot's software and hardware should be optimized to minimize any latency or response time.

Precision: The robot must be able to measure and control the soil moisture level and watering volume with high precision, to ensure optimal plant health and avoid overwatering or underwatering. The precision of the robot's sensors, actuators, and control algorithms should be optimized to achieve this goal.

Capacity: The robot must be able to handle multiple plants, each with different watering schedules and moisture levels, simultaneously. The capacity of the robot's database, storage, and processing power should be optimized to accommodate this requirement.

Safety: The robot must operate safely, without causing any harm to the plants or the users. The safety of the robot's hardware, software, and communication channels should be optimized to prevent any accidents or failures.

Reliability: The robot must be reliable, and its software and hardware should be designed to minimize the risk of failure or malfunction. The reliability of the robot's components, sensors, actuators, and communication channels should be optimized to ensure that the plant care process is not disrupted.

5.2 Safety Requirements

Electrical Safety: The robot's electrical components, including the power supply, wiring, and connectors, must be designed and installed to prevent electric shock or fire hazards.

Mechanical Safety: The robot's mechanical components, including the chassis, motors, and moving parts, must be designed and installed to prevent physical harm to the plants or users.

Fire Safety: The robot's components and enclosure must be designed and installed to prevent any fire hazards, such as short circuits or overheating.

Data Privacy: The robot's communication channels and data storage must be secured to prevent any unauthorized access or data breaches.

5.3 Security Requirements

Authentication and Authorization: The robot's software and hardware must include authentication and authorization mechanisms to ensure that only authorized users can access the robot's data or control its operations.

Data Privacy and Protection: The robot's communication channels, data storage, and processing must be secured to prevent any unauthorized access, data breaches, or data loss.

System Integrity: The robot's software and hardware must be designed and installed to prevent any malicious attacks, such as viruses or malware, that could compromise the integrity of the system. The robot must comply with relevant system integrity standards, such as ISO/IEC 27001, and use secure coding and testing practices.

Policy and Compliance: The robot must also provide a clear and transparent privacy policy to inform users about the data collection and usage practices.

5.4 User Documentation

To produce this project, you'll need some tools, some material, and the code from this project

Software Setup

The program is written in Arduino IDE in c++ programming language. The code is well-commented and easy to understand. Compile the irrigation code and upload it to the microcontroller, using Arduino IDE version 1.

Once the arduino is loaded with the program it's time to wire it all up. Follow the diagram and wire everything up accordingly. Do not plug the arduino in to power at this point. Make sure everything is situated before letting it loose on your plants. You will need to attach a hose to the pump which will deliver water to your plants. I used a fish air hose to do this.The sensor will calibrate by itself once it is kept in the soil and the threshold value will be shown on the serial monitor in Arduino. Serial debugging is available in this program.

Code

Upload the attached sketch to your arduino. Once the arduino is loaded with the program it's time to wire it all up. Follow the diagram and wire everything up accordingly. Do not plug the arduino in to power at this point(important point)

First, you have to make to install these libraries in arduino IDE

```
 #include <AFMotor.h>
#include<Servo.h>
```

We have to define the triggered pin and echo pin for ultrasonic sensor 1 and ultrasonic sensor 2 triggered pin> is used to trigger the ultrasonic sound pulses. Echo pin produces a pulse when the reflected signal is received. echo pin>Echo pin is an **Output pin**. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor

```
const int trigPin = A0;                 //Defines the trig pin of ultrasonic sensor 1st
const int echoPin = A1;                  //Defines the echo pin of ultrasonic sensor 1st
const int trigPin1=A2;                  //Defines the trig pin of ultrasonic sensor 2nd
const int echoPin1=A3;                   //Defines the echo pin of ultrasonic sensor 2nd
```

Defines the pin of moisture sensor pin and store value given by moisture sensor

```
int mpin = A4;
int mout;
```
Defines the frequency which will be given to motor 1 and motor 2

```
AF_DCMotor motor1(1, MOTOR12_1KHZ);
AF_DCMotor motor2(2, MOTOR12_1KHZ);
AF_DCMotor motor4(4, MOTOR12_1KHZ);
```
Starts serial communication with the arduino and PC and /Define the attached pin for servo motor 1

```
void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  m1.attach(10);
  m2.attach(9);
```
To set the particular speed of motor 1 and motor 2

```
 motor1.setSpeed(255);
  motor2.setSpeed(255);
```
Formula that set the distance of robot goes from one plant to next one

```
distance= duration*0.034/2;
```
Make sure everything is situated before letting it loose on your plants. You will need to attach a hose to the pump which will deliver water to your plants. If you haven't plugged in the pump go ahead and do so. Finally connect the arduino to your chosen power source. I'm using a simple 5V charger and a USB cable.

Tools

The tools are very simple, I used for this project:

- Battery holders
- mounting
- A flat screwdriver
- Plastic rod

If you have them you can also add some wire strippers, but they are not indispensable.

Material

Here is a list of the products used to build the system.

To control the system:

- Arduino UNO (2100 pkr)
- Servo SG90(640)
- L293D Sheild(500pkr)

- 100 RPM Geared motor and wheels(880pkr)
- Prototyping jumper wires(140pkr)
- Wooden box
- Plastic container

For the autonomy in energy:

- LI_ON Battery(280pkr)
- Charging module(TP4056)(120pkr)

For the water tank:

- 12V DC pump(350pkr)

The sensors:

- Ultrasonic sensor(380pkr)
- Moisture sensor(180pkr)

For a total of 10k. That's not cheap! But keep in mind that it's still cheaper than a pre-built system, and with a lot more capabilities! Also, some parts are only for prototyping, and we purchased many components in groups of several pieces for other projects, you don't need 3 NodeMCU boards, nor 6 relays for this project

Power Block The L293D Sheild can drive 4 DC motors and 2 stepper or Servo motors at the same time. Each channel of this module has the maximum current of 1.2A and doesn't work if the voltage is more than 25v or less than 4.5v.the supply voltage and reduces it to a constant 5V making it suitable to run the Arduino & Soil Moisture Sensor.

Moisture Sensor The sensor feeds an analog value to the Arduino. The threshold level of moisture is calibrated by the user depending on the type of plant used.

GUIDELINES:

Before powering the circuit on, you need to keep in mind the following macro definitions in the code:

- Changing the angle of rotation of the servo horn toward the first plant and second plant. The default values are 70 degrees and 145 degrees.
- Changing the watering time according to the size of the plant. The default values are five seconds and eight seconds.
- Changing the threshold value according to your need. The default value is 600.

The water pipe is connected to the servo motor which rotates according to the requirement.

- If there are two crops A & B. and if A has less amount of moisture then the servo motor rotates toward crop A.
- Starts the watering and when it will fill up it will rotate towards PLANT B. this is one more benefit of this project.

Place the flower plants where the pipe from the servo motor can easily reach them. When the moisture level dips below 600, the servo rotates at an angle of 70 degrees. That is after the servo motor horn moves 70 degrees toward the first pot, the motor pump will be on for five seconds and then stop automatically. Then, the servo returns to its original position. Similarly, if you are using a second sensor, the servo motor horn will move to 145 degrees to the second biggest pot, and the motor pump will be on for eight seconds and then stop automatically. The servo returns to its original position.

User Manual:.
This user manual contain a detailed guide about the project with help user to setup the whole project very easily. Here is the file of user manual
user manual.docx

6. Assumptions and Dependencies

Environmental factors: The performance and reliability of the plant watering robot using Arduino may be affected by environmental factors such as temperature, humidity, and soil conditions. If these factors are not accounted for in the requirements, the system may not function properly.

Availability of components: The availability of components such as sensors and other hardware may affect the development and deployment of the plant watering robot using Arduino. If certain components are not available or are delayed, it may impact the system's performance or delay its development.

Changes in technology: Changes in technology could impact the plant watering robot using Arduino's requirements. For example, new sensors or other hardware components may become available that could improve the system's performance or functionality.

User behavior: The behavior of the end-users could affect the performance and reliability of the plant watering robot using Arduino. For example, if users do not properly maintain the system, it may not function properly.

Changes in regulations: Changes in regulations could impact the requirements for the plant watering robot using Arduino. For example, if new safety or privacy regulations are introduced, the system may need to be modified to comply with these regulations.

The plant watering robot using Arduino may have dependencies on external factors such as:

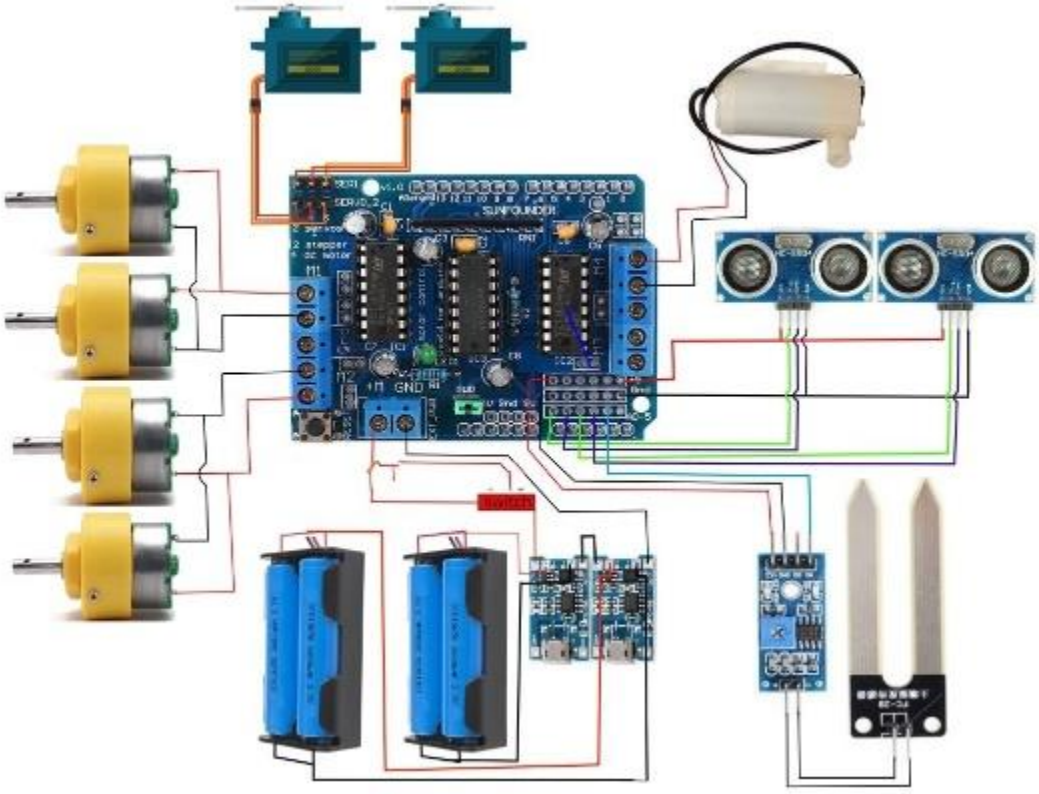
Availability of hardware components: The plant watering robot using Arduino may be dependent on the availability of certain hardware components that are being developed by third-party vendors. The system's development and deployment may be delayed if these components are not available or are delayed.

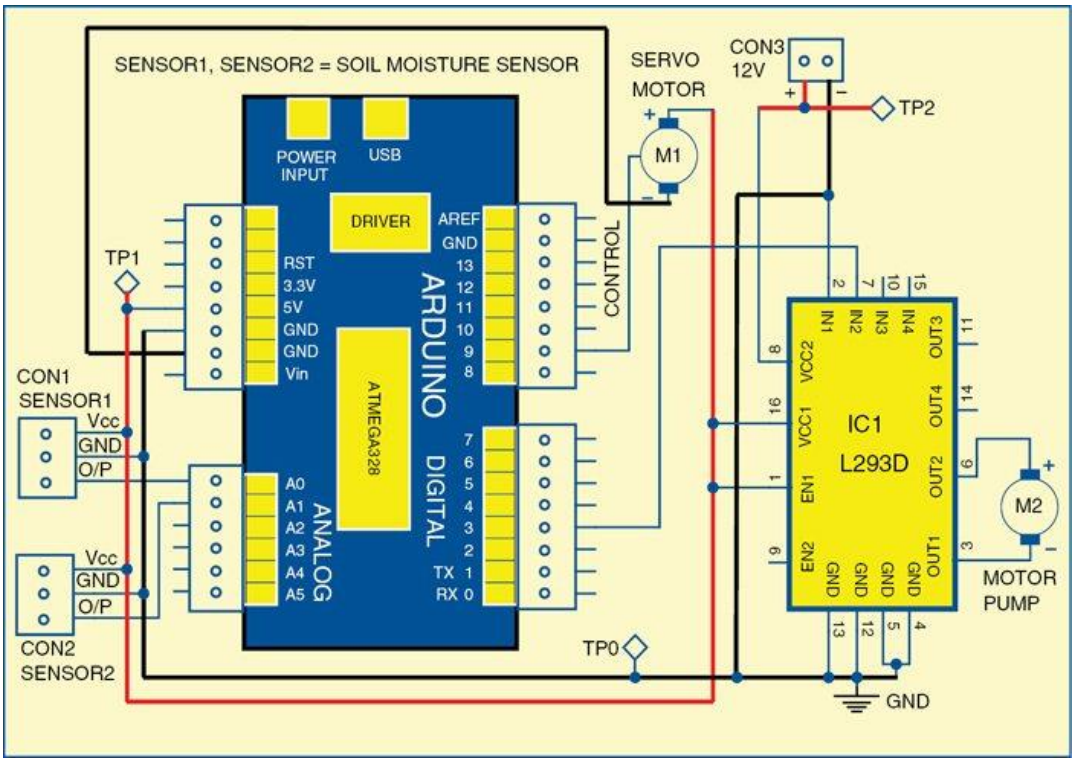
Availability of software components: The plant watering robot using Arduino may also be dependent on the availability of certain software components such as libraries or APIs that are being developed by third-party vendors.

Integration with other systems: The plant watering robot using Arduino may need to be integrated with other systems, such as a cloud-based irrigation management system or a weather forecast service. The system's performance and functionality may be impacted if these integrations are not completed properly or are delayed.

Regulatory compliance: The plant watering robot using Arduino may be dependent on meeting certain regulatory compliance requirements. For example, the system may need to comply with safety or privacy regulations, which could impact its design and development.

7. System Architecture



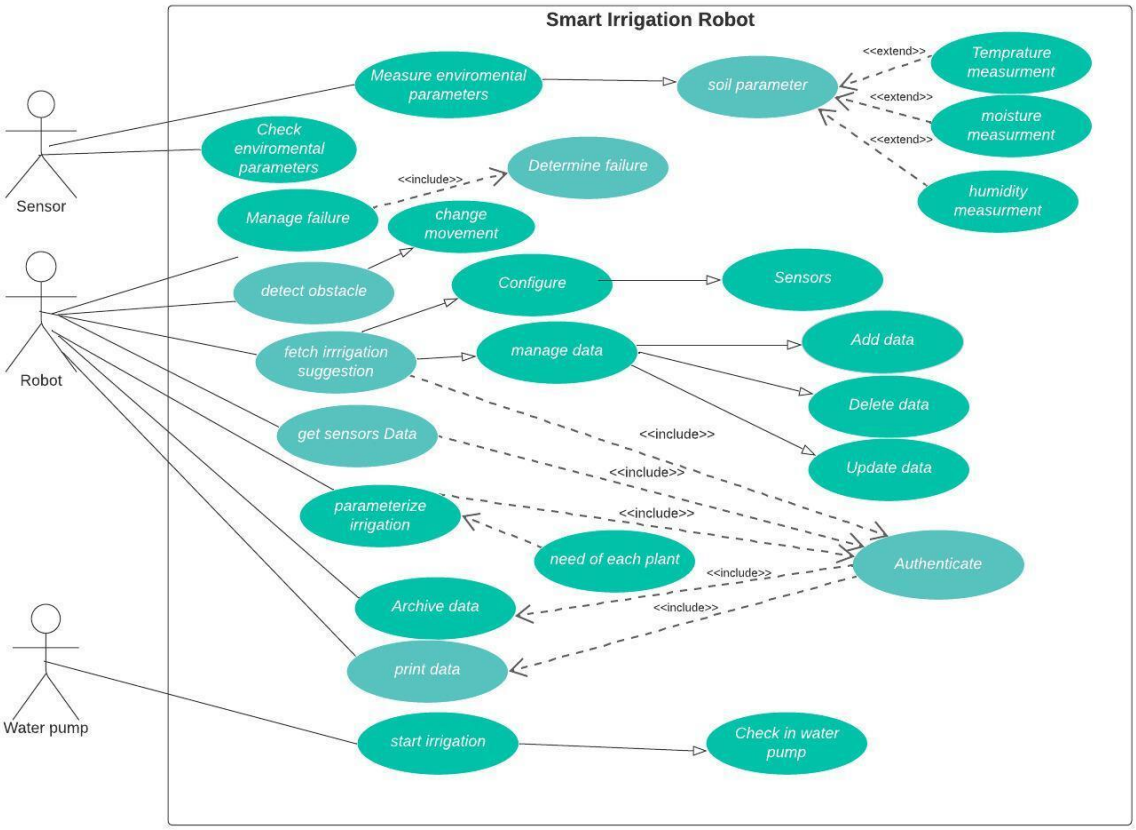


The plant watering robot using Arduino consists of several hardware components such as a water pump, moisture sensors, and a microcontroller. The microcontroller, which is an Arduino board, is responsible for controlling the operation of the hardware components based on the system's software.

The data flow in the system starts with the moisture sensors, which collect data on the soil moisture levels. This data is then sent to the microcontroller, which processes the data and sends control signals to the water pump to turn it on or off based on the system's rules. The microcontroller also sends the data to the data storage component for logging and analysis.

8. Use Cases

8.1 UseCase Diag



8.2 Use Case Description

< UC-001: **ACTORS**>	
Actors: 3 Actors incude in use case one is ROBOT and the second is WATER PUMP and 3rd one is SENSORS	
Feature:	Smart plant watering robot
Use case Id:	UC-001
Pre-conditions:	Assumptions required before executing the use case of a plant watering robot are: 1. The plant watering robot is designed to work for indoor plants only. 2. The robot is equipped with a water tank with a sufficient amount of water, as required for watering plants. 3. The robot has a set of sensors that can detect when a plant needs

	watering based on soil moisture levels. 4. A specific watering schedule has been set up for each plant based on its watering needs. 5. The robot has been programmed to move around the room without obstacles and return to its charging station when the battery is low or at the end of its workday. 6. The charging station is set up in a convenient location for the robot and within range of the robot's sensors and communication systems. 7. The robot is equipped with a mechanism that can control the amount of water it dispenses while watering the plants. 8. The user has provided the robot with access to the relevant areas of the indoor environment, including the plant pots and any other required resources. 9. The robot has a reliable and strong internet connection to communicate with the user and receive updates, if applicable.

Scenarios

Step#	Action	Software Reaction
1.	User - the person who purchases and uses the plant watering robot, as well as monitors its performance and interacts with it as needed.	1. Creating the watering schedule 2. Monitoring robot performance 3. Providing feedback 4. Adjusting the robot's movement 5. Configuring the robot setting
2.	Robot - the main actor responsible for carrying out the plant watering use case	

Alternate Scenarios.

1. If the moisture level in the soil is high, the robot could skip watering the plant and move on to the next plant in the queue.
2. If the robot encounters an obstacle while navigating through the garden, it could detect the obstacle and change the course accordingly to avoid it.
3. The robot could have a feature to automatically adjust the quantity of water depending on the type of plant being watered.
4. It could have a feature to detect and analyze weather forecasts to adapt its watering schedule to optimize plant growth and conserve water resources.
5. In case the robot senses that there is an issue with the watering system, it could alert the user about the issue to get it resolved promptly.

6. The user could have the option to communicate with the robot through a mobile app or personal voice assistant to monitor and control the robot's activities.
7. After watering all the plants, the robot could provide the user with a detailed report of which plants were watered, when, and how much water was used to provide valuable insights into the garden's health**.**

Post Conditions

Step#	Description

	Here are some conditions that can be expected at the completion of the use case in plant watering robot: **1.** All plants in the garden would have been watered correctly based on their moisture level and watering needs. **2.** The robot would have navigated through the garden seamlessly, without getting stuck or damaging any plants. **3.** The amount of water used by the robot should be optimal, ensuring plant health and water conservation. **4.** Any obstacles in the garden would have been avoided or appropriately handled by the robot. **5.** The user would have received accurate feedback and reports concerning the watering process via an app or voice assistant. **6.** Overall, the garden should be healthier, and plant growth should have improved, leading to more yield or a better-looking garden.
Use Case Cross referenced	<Related use cases, which use or are used by this use case>

Concurrency and Response

Give an estimate of the following

To determine the number of concurrent users and expected response time for a plant watering robot, we need to consider several factors:

1. The type of communication protocol used by the robot (e.g. Bluetooth, Wi-Fi, cellular, etc.)
2. The processing power and memory capacity of the robot
3. The complexity of the watering schedule and the number of plants being watered
4. The distance between the robot and the user's device

- ★ Assuming that the robot uses a Wi-Fi protocol and has adequate processing power and memory, the number of concurrent users should not be an issue as the robot can be controlled by multiple users simultaneously.
- ★ In terms of response time, the expected time would depend on the complexity of the schedule and number of plants. For example, if the watering schedule is simple with only a few plants, the response time could be almost instantaneous. However, if the schedule is complex and the robot is watering a large number of plants, the response time could be slower.
- ★ In general, for a plant watering robot, users would expect a response time of a few seconds to a minute. However, if the robot encounters any issues, such as a plant accidentally being over-watered or under-watered, the response time should be much quicker to avoid any further damage.

9. Graphical User Interfaces

Give a detailed account of user interfaces included in this project.

<UI 01>	
Interface Id.	UC-001

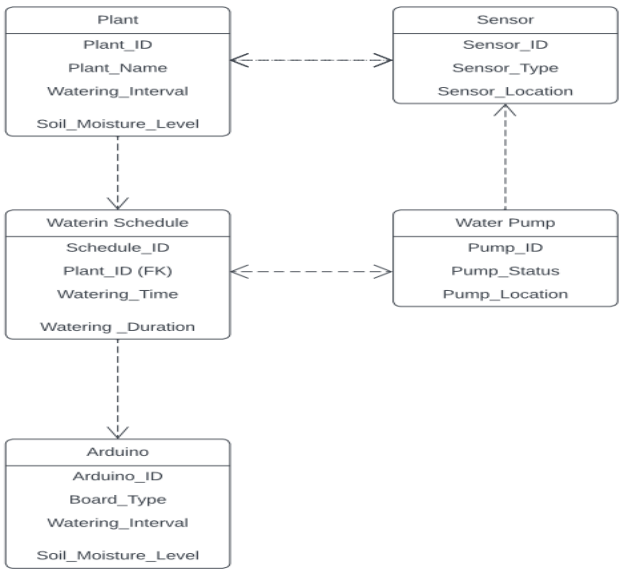
Use case Reference	
Snapshot	
There is no graphical user inface ot smart irrigation robot as it is self working robot which doesn't require any interface	
Data dictionary reference	
Label	**Data dictionary identifier**
	1. Plant ID - a unique identifier assigned to each plant in the system
	Water Level - the current amount of water in the plant's soil
	3. Watering Schedule - the frequency and duration of watering for each plant
	4. Soil Type - the specific type of soil in which each plant is planted
	Plant Species - the specific species of each plant in the system
	Watering History - a record of when each plant was last watered and how much water was applied
	Sensor Data - data collected from any sensors used in the system, such as soil moisture sensors or temperature sensors
	User ID - a unique identifier assigned to each user of the system
	User Preferences - user-specific settings, such as notifications or preferred watering hours
	Error Logs - a record of any errors or issues that occur in the system, including error codes and descriptions.

10. High Level Design

10.1 ER Diagram

DBMS ER diagram (UML notation)

Areej | May 28, 2023



10.2 Data Dictionary

The convention recommended for writing the data dictionary is as follows.

< Data 1>	
Name	Plant
Alias	None
Where-used/how-used	This entity represents the plant's that need watering.It is used as an input to the watering process.
CONTENT DESCRIPTION	The plant entity can have attributes such as Plant_ID (unique identifier), Plant_Name, Watering_Interval, Soil_Moisture_Level etc.
Supplementary information	None

< Data 2>	
Name	Sensor

Alias	None
Where-used/how-used	This entity represents the sensors used to detect soil moisture levels.
CONTENT DESCRIPTION	The sensor entity can have attributes such as Sensor_ID (unique identifier), Sensor_Type, Sensor_Location etc.
Supplementary information	None

< Data 3>	
Name	Watering Schedule
Alias	None
Where-used/how-used	This entity represents the schedule for watering plants.
CONTENT DESCRIPTION	The watering schedule entity can have attributes such as Schedule_ID (unique identifier), Plant_ID (foreign key referencing the plant entity), Watering_Time, Watering_Duration etc.
Supplementary information	None
< Data 4>	
Name	Water Pump
Alias	None
Where-used/how-used	This entity represents the water pump used for watering plants.
CONTENT DESCRIPTION	The water pump entity can have attributes such as Pump_ID (unique identifier), Pump_Status, Pump_Location etc.
Supplementary information	None

< Data 5>	
Name	Arduino
Alias	None
Where-used/how-used	This entity represents the Arduino Microcontroller board used to control the watering process.

CONTENT DESCRIPTION	The Arduino entity can have attributes such as Arduino_ID (unique identifier), Board_Type, Firmware_Version etc.
Supplementary information	None

The notation to develop content description is given below:

Data construct	**Notation**	**Meaning**
	=	is composed of
Sequence	+	and
Selection	[|]	either-or
Repetition	$\{\}^n$	n repetitions of
	( )	optional data
	* … *	delimits comments

11. Requirements Traceability Matrix

Sr. #	Feature	Use case ID	UI ID	Priority	Build Number	Use Case Cross reference (Related Use Cases)
	Move robot to desire location	001	UI 001	high	001	Actor(Robot)
	Monitor Soil moisture level	002	UI 002	high	002	Actor (Sensors)
	Detect obstacles and avoid collision	003	UI 003	high	003	Actor(Robot)
	Recharge battery	004	UI 004	high	004	Actor(Robot)
	Set watering Schedule and amount	005	UI 005	low	005	Actor(water pump)

The columns carry the following meaning:

- Feature: Lists system features based on which use cases are built.
- Use Case ID: Write the ID of the use case for easy lookup
- UI ID: Write the user interface ID for this use case.
- Priority: Give an appropriate rating to each use case according to its priority
- Build Number: Write the reference number to which this feature belongs.
- Use Case Cross Ref: Write the related use cases separated with commas.

12. Risk Analysis

Some possible risks associated with a plant watering robot include:

1. **Physical injury**: The robot could accidentally bump into someone or something, causing injury or damage.

2. **Water damage**: The robot could malfunction and overwater or spill water onto surfaces, causing damage or creating a slip hazard.

3. **Electrical shock**: The robot uses electricity to power its functions, and a malfunction or damage to the electrical components could lead to electrical shock.

4. **Plant damage**: The robot might not accurately detect the moisture levels in the soil, causing under or over watering, which could damage or kill plants.

- **Risk Identification**.

  1. **Technical malfunction:** There is a risk of technical malfunction, such as malfunctioning of sensors, faults in programming or hardware failure, that could lead to the robot not watering the plants correctly or not working at all.

  2. **Power supply disruption**: There is a risk of power supply disruption, such as power outages or battery exhaustion, that could cause the plant watering robot to stop working abruptly.

  3. **Plant damage**: There is a risk of plant damage if the robot malfunctions or if the sensors and programming are not set up correctly. The watering system could also cause over-watering or under-watering of plants, leading to their deterioration.

  4. **Environmental conditions**: Environmental conditions, such as extreme temperatures and humidity, could affect the performance of the robot and potentially cause malfunctions or damage to the robot.

  5**. Inadequate maintenance**: If the robot is not maintained properly, there is a risk of wear and tear that could cause malfunctions or breakdowns.

  6. **Security breaches**: There is a risk of security breaches, such as hacking or unauthorized access to the system, which could compromise the performance of the robot or lead to the theft of valuable plant data.

  7. **Compatibility issues:** There is a risk of compatibility issues if the robot is not compatible with other systems or devices, which could lead to communication issues or malfunctioning of the robot.

  8. **Cost of maintenance**: The cost of maintaining a plant watering robot could be high, which could lead to financial risks if the maintenance expenses are not factored into the budget.

- ## Risk Drivers

  1. **Complexity of technology** - The technology used to create plant watering robots is complex and can be difficult to understand, which increases the risk of error or malfunctions.

  2. **Dependence on sensors** - The robot relies heavily on sensors to determine water needs and navigate around objects, which can malfunction or provide inaccurate data.

  3. **Unforeseen environmental variables** - The robot may not be equipped to handle all environmental variables such as humidity or temperature changes, leading to inaccuracies in watering the plants.

  4. **User error** - Users may not fully understand how to properly program or operate the robot, leading to errors that can damage plants or cause physical harm.

5. **Cybersecurity threats** - As more plant watering robots are connected to the internet, they become vulnerable to cybersecurity attacks which can cause the robot to malfunction or compromise user data.

6**. Battery or power issues** - The robot's battery or power source may fail, leading to operational issues and potential safety hazards.

7. **Limited monitoring capabilities** - The robot may not have the ability to monitor all plants in a large area, leading to under or over-watering in certain areas.

8. **Insufficient testing and maintenance** - Lack of regular testing and maintenance could contribute to malfunctions and safety hazards.

- **Risk Mitigation Plan**

To mitigate risks associated with plant watering robots, the following risk mitigation plan can be implemented:

1. **Conduct a thorough risk assessment**: Identify all potential risks involved in the use of plant watering robots and assess their severity.

2. **Design for safety**: Implement safety features such as sensors to detect water and soil moisture levels, and collision avoidance technology to prevent damage to plants or the robot itself.

3. **Regular maintenance and calibration:** Regularly maintain and calibrate the robot to ensure it operates optimally and accurately delivers the right amount of water to plants.

4. **User training**: Provide clear and concise user manuals and training to ensure that users understand how to properly operate and maintain the robot.

5. **Test in controlled environments:** Test the robot in a controlled environment to ensure it works as intended and identify any potential issues before deployment.

6. **Limit autonomous operation:** Limit the autonomous operation of the robot to specific areas and tasks to reduce the risk of damage or harm.

7. **Emergency stop feature**: Implement an emergency stop feature that can be activated by the user in case of malfunctions or emergencies.

8. **Regular monitoring**: Regularly monitor the robot's performance, and identify and address any problems as soon as they arise.

13. Cost Estimation Sheet

1.	**Software development cost**	nil
2.	**Packaged software**	Arduino IDE version 1
3.	**Hardware**	15k
4.	**Network**	
5.	**Client**	Farmers, students, researchers
6.	**Misc.**	
		Total cost =

CHAPTER 3 TEST PLAN

I. Introduction

Smart Classroom is the system that is being tested its testing consist of three modules which are **Smart lights and Smart fans** which are motion sensing.

Fingerprint door lock which aimed to reduce the consumption of energy and to reduce the disturbance during class. It may include a fingerprint scanner that scan and verify and for automatic lights and fans the main thing is PIR sensor.

a. Purpose of Document

The purpose of this document is to specify all the requirements of Smart Classroom in detail. The document identifies all the software as well as hardware requirements for the Smart classroom being developed. It also gives an overview of functional and non-functional requirements, user characteristics, use cases, intended audience and other constraints. Moreover, the document defines domain-specific as well as project-specific terms that will be used each in its own particular context, hence for the entirety of the project

Intended Audience

1. Developers
2. Project Adviser
3. Evaluation panel
4. Users (Teachers and students)

b. Project Overview

Smart classroom which aim to replace the conventional system. Project has cover two main modules of smart classroom in which the first one is biometric door-lock and second one is smart light and fans.

To function, the user needs to match the fingerprint , which will scan the fingerprint and verify fingerprint in less than 1 second through hand wave gesture(Module 1).

In module 2, when there will be nobody the lights and fan will automatically switch off. First of all it will help in reducing the disturbance in the classroom as teacher will enter the class and scans his fingerprint the door will automatically closed and when there will be no body in the class room the lights and fans will automatically switch off.

II. Scope of Testing

★ **Functional testing:**

There are no syntax errors in this code, but there are some potential logical issues

- In the ultra_read function, the ultra_time variable is passed as a reference but is not being used or updated within the function
- The flag1 and flag2 variables are being used to keep track of whether a person is detected by the two ultrasonic sensors, but they are not being reset to 0 when a person is no longer detected.

2.Security testing

The system must be able to be operated by a person who is allowed to operate it.
Temperature should be maintained in order to avoid burning of wires and sensors.
The fingerprint module should be dust free, so that it can scans properly.

3.usability testing

The LCD display is not being cleared before updating the display in the loop function, which could cause overlapping text if the display is not wide enough to accommodate the new text.
The message "fan is" is always displayed on the second line of the LCD, even if the fan is off. You might want to move this message inside the "if" statement that checks the status of the fan.

14. Performance testing

The distance threshold of 90 centimeters for detecting a person seems arbitrary and may not be appropriate for all situations.

15. Regression testing

Pin conflicts: The code uses pins 2 and 3 for the software serial connection and pin 8 for the relay. If these pins are already being used by other devices or functions in the hardware, conflicts may arise.

Timing issues: The code uses delay() functions to control timing between different operations. If the delays are not set appropriately, the program may not function as intended.

III. Test Plan Strategy

Unit Testing

Definition:

Unit testing is a type of testing that focuses on the individual components, or units, of a software system. When it comes to a smart classroom project, there are several different components that could be tested individually. Here are some examples of unit testing that could be performed:

Participants: Anam Shahzadi, Rida Tahir, Areej Fatima

Methodology:

Unit testing in the context of a smart classroom project would involve testing individual pieces of the software or hardware that make up the overall system. The following is a proposed methodology for conducting unit testing in a smart classroom project:

*1. **Testing the user interface:** This test would focus on the components of the user interface, such as buttons, text boxes, and menu items. It would ensure that each component is working properly and that user input is properly processed.*

*2. **Testing the connectivity**: This test would focus on the components that allow the smart classroom to connect to various devices, such as projectors, speakers, and other audio/visual equipment. It would ensure that the data and signals sent and received are accurate and timely.*

*3. **Testing the data storage**: This test would focus on the components that allow the smart classroom to store and retrieve data, such as student attendance records and lesson plans. It would ensure that data is properly stored and retrieved,*

Integration Testing

Definition:

a type of software testing in which the different units, modules or components of a software application are tested as a combined entity.

Participant Rida Tahir, Anam Shahzadi, Areej Fatima

Methodology:

1. Data Flow: The system's data flow should be verified. This means checking whether the data is properly flowing from one component to another without any loss or data corruption.

2. Compatibility: Compatibility testing should be carried out to ensure that all the hardware and software components of the smart irrigation system are compatible with each other.

3. Communication: Communication testing should be done to verify that the communication between the hardware and software components of the smart irrigation system is working properly. This includes testing that the system is able to receive data from sensors and that it is sending signals to control the irrigation system.

4. End-to-end functionality: Integration testing should include end-to-end testing, which checks whether the smart irrigation system is capable of completing the intended task. For example, the system should be tested to verify that it can detect when a plant needs watering and that it is capable of providing the appropriate amount of water.

5. Error Handling: Integration testing should also verify that the system can handle errors gracefully. This means testing how the system responds when components fail or data errors occur.

6. Performance: Integration testing should include testing to ensure that the system performs optimally under different conditions. For example, testing the system's performance under different soil moisture levels and environmental conditions.

Memory limitations: Depending on the memory available on the hardware, the program may run out of memory if too many variables or functions are used. This can cause crashes or unexpected behavior

3.System Testing

System testing in a smart classroom project is a type of testing that verifies the functionality and performance of the entire smart classroom system. This testing is performed on a fully integrated system to ensure that it is working correctly and meeting all the requirements.

Participants
Anam Shahzadi, Rida Tahir, Areej Fatima

Methodology:
The methodology for system testing of a smart classroom project would involve the following steps:

1. Functional testing: to ensure that all the features and functionalities of the smart classroom system are working as per the requirements.

2. Performance testing: to check if the smart classroom system is able to handle a certain level of load without compromising on its performance.

3. Compatibility testing: to ensure that the system is working correctly in all the targeted environments like different browsers, operating systems, and devices.

4. Security testing: to verify the system's ability to protect user data and prevent unauthorized access.

The ultimate goal of system testing in a smart classroom project is to ensure that the system is fully functional, reliable, and performs as expected. This testing phase helps to identify defects or issues early in the development process, which can save time and money in the long run.

Overall, a well-defined testing methodology is essential to ensure that the smart classroom system is developed and delivered with the highest quality possible.

IV. Test Environment

To create a smart classroom test environment, the following hardware and software configurations are typically required:

Hardware:

- A smart board or interactive whiteboard
- A projector or display monitor
- A computer or laptop
- Fingerprint lock
- Fan
- Light
- Wi-Fi or wired internet connectivity

Software:

- An operating system (Windows, MacOS, Linux)
- Arduino IDE (for software testing)
- Mantis Bug tracker(For tracking and making reports of bugs and errors that occur during testing of the project)

V. Schedule

Testing Activities	Start Date	End Date
Functionality Testing	20-04-2023	20-04-2023
System Testing	21-04-2023	21-04-2023
Integration Testing	24-04-2023	25-04-2023
Performance Testing	20-04-2023	21-04-2023
Security Testing	27-04-2023	28-04-2023
Usability Testing	20-04-2023	21-04-2023
Interoperability Testing	21-04-23	21-04-2023

VI. Control Activities

In the context of a smart classroom using Arduino, control activities might include monitoring the performance of sensors and other devices, controlling the flow of data, and managing user access to the system.

We held meeting for the evaluation of testing strategy of smart classroom.For this purpose each Members contributes equally.

No of members	Name
1	Rida Tahir
2	Anam Shahzadi
3	Areej Fatima

Meetings:

We divide our work in different days according to this we held different meeting:

1. **First meeting** was held on **19-04-2023** we detected code errors, install libraries, check how much time it take for installation of libraries.
2. **Second meeting** was held on **20-04-2023**, in this meeting Functionality, usability and performance testing was done.
3. **Third meeting** was held on **21-04-2023**, in this Performance, system and interoperability testing was done.
4. **Fourth meeting** was held on **24-04-23**, in this meeting Integrating testing is done.
5. **Fifth meeting** was held on **27-04-23**, we did work on Security testing.
6. In the **Sixth meeting** that was held on **28-04-2023** we checked the overall working of Smart classroom and modules that are used in smart classroom project like fan modules , fingerprint door lock and light module.

Each member equally took part in these testing
In the context of a smart classroom using Arduino, control activities include

- Schedule meeting
- Planning
- Choose the right participants
- Detailed meeting with individual
- Perform Testing
- Prepare report
- Track and report the bugs\errors

Number of Activity	To do	Have done
1	Schedule meeting	We schedule a meeting with project team and enquired about the details of their project.
2	Planning	We plan what and how testing being done? What type of strategy applied to complete the testing?
3	Choose the right participants	Group leader(Rida) assign each module a module for perform testing
4	Detailed meeting with individual	Each individual schedule a meeting to ask about the details of perspective module
5	Perform Testing	First, each member test the code using Arduino IDE software then test the hardware of assigned module
6	Prepare report	Each member make report of assigned module all pros and cons of modules are added in it.
7	Track and report the bugs\errors	We track the bug and error using the mantis bug tracker then make a detailed report of it

VII. Functions to be Tested

Here is an example of how the requirements for testing a smart classroom using Arduino can be listed and described:

Smart lights

- Test the functionality of the Arduino-controlled lights, ensuring that they can be turned on/off automatically based on the time of day and occupancy status of the room.
- Verify that the system responds correctly to user inputs, such as adjusting the brightness or turning on/off specific lights using a mobile app or web interface.

Smart Fans

- Test the functionality of the Arduino-controlled fan, ensuring that they can be turned on/off automatically based on the time of day and occupancy status of the room.
- Verify that the system responds correctly to user inputs, such as adjusting the turning on/off specific fans using a mobile app or web interface.

Automated Fingerprint Door-lock

- Fingerprint recognition accuracy
- False acceptance rate
- User capacity
- Speed of recognition
- Durability and resilience
- Battery life and power management
- User interface and ease of use

VIII. Functions not to be Tested

Arduino Library Functions:

These are built-in functions provided by the Arduino framework. They are extensively tested and have a low risk of failure. These functions do not need to be tested as they have been tested by the Arduino community.

Hardware Drivers:

These are also low-risk functions that interface directly with the hardware components. Hardware drivers are usually pre-tested by the manufacturers, and the likelihood of a defect is relatively low.

Reusable Functions:

If a function has been tested previously and has demonstrated 100% satisfaction, it is not necessary to test it again. These functions can be reused across multiple projects and are considered low-risk.

Simple Functions:

Functions that perform simple tasks, such as reading data from a sensor or writing data to an output pin, may not need to be tested as they have a low risk of failure.

User Interface Functions:

Functions that are purely cosmetic and do not have any effect on the functionality of the system may not need to be tested. Examples of these functions include changing the background color of the user interface or displaying a message on the screen.

IX. Test Case Design and Description

Test Case ID:	*TC0001*	**QA Test Engineer**	*Rida Tahir*
Test case Version:	SCS-TC-001	**Reviewed By**	*Rida Tahir*
Test Date:	*20-4-2023*	**Use Case Reference(s)**	*Fingerprint door lock*
Revision History	*no*		
Objective	*Fingerprint module must follow the login and matching process.* When fingerprint login each enter two times, will two the second entry image is processed, and the composite template is stored in the module. When the fingerprint is matched through the fingerprint sensor the image to be verified entered and processed then matched with the fingerprint template		
Product/Ver/Module	**Product**. Fingerprint door-lock **Vesion**. No **Modules**. Fingerprint sensor, power supply, memory management, Led indicator, user interface, locking mechanism		
Environment:	**Hardware** For the implementation of this project we will be using a fingerprint sensor, arduino, breadboard, solenoid door lock, LCD 16x2. **Software** Arduino IDE		
Assumptions:	Stable Power Supply Authorized users only Consistent fingerprint replacement No hardware or software defects Fingerprint recognition accuracy		
Pre-Requisite:	*Infrastructure to build properly* Matching and login process have to worked properly		

Step No.	**1.1.1.1.1 Execution description**	**Procedure result**
1 2 3 4 5	Login process Matching process Memory managment Hardware (RFID ,sensor) Interface	*Arduino IDE software in used for the software testing of fingerprint door lock code is being tested properly it is error free and working as per described*

Comments

There are errors in infrastructure which has to corrected asap

	Passed Failed Not Executed

Test Case ID:	TC002	**QA Test Engineer**	Anam Shahzadi
Test case Version:	001	**Reviewed By**	*Rida Tahir*
Test Date:	20/04/2023	**Use Case Reference(s)**	Fan module
Revision History	properly functioning		

Objective	To check the Fan module Functionality
Product/Ver/Module	Product: Smart classroom Version: Nil Module: Fan Module The Fan module test case ensures that the Fans will be on until the person is in the room .
Environment:	**Hardware**: - Microcontroller (Arduino) - Ultra sonic sensors hc-sr04*2 - jumper wires - channel relay - LED display -100R resistor - 4.7K resistor - 1K resistor - Bulb holder - 220v LED bulb - 5v 22amp power adapter **Software**: - Programming software (such as Arduino IDE or Python) - Code libraries or modules for controlling the fans and sensors - Communication software (such as Bluetooth or Wi-Fi) - Mobile or desktop application for remote control and monitoring of the fan module
Assumptions:	1. The fan module detect when the classroom is unoccupied and turn off automatically. 2. Reliable power supply
Pre-Requisite:	1. Internet connectivity: The light module must be connected to the internet to function properly. 2. Fans automatically on and off after enter or exit the room. 3.Infrastructure build properly

Step No.	Execution description	Procedure result
1. 2. 3. 4.	Entry process Exit process Hardware(Aurdino or Ultra sonic sensors) Interface	The Fans accurately detects and recognizes persons and automatically on when person enter the room and off when person leave the room. . The system responds appropriately when the person is detected. Arduino IDE software in used for the software testing of fan module code is being tested properly.

Comments: There was no issue while performing this test, and also was easy to use this functionality.

	Passed failed not executed		
Test Case ID:	*TC0003* *Light Module*	**QA Test Engineer**	Areej Fatima
Test case Version:	SCS-TC-003	**Reviewed By**	*Rida Tahir*
Test Date:	*Date 23-04-2023*	**Use Case Reference(s)**	Light module
Revision History	*Created the initial version of the use case, describing the process of student existance and unavailability in a smart classroom system.*		
Objective	*The objective of the sensor testing is to validate the accuracy, reliability, and functionality of the sensors used in the smart classroom system, including the ultrasonic sensor. The testing aims to ensure that the sensors correctly detect and measure the required parameters such as student presence and student unavailability in the class, enabling the system to make informed decisions for turning on or off the light.*		
Product/Ver/Module	**Product**. *Light module* **Vesion**. No **Modules**. light sensor, resistor, LED indicator, user interface		
Environment:	**Hardware** Arduino Ultra sonic sensors Breadboard Relay LED bulb Jumper wires Bulb holder Power adapter Resistor **Software** Arduino IDE Programming software (such as Arduino IDE) - Code libraries or modules for controlling the light and sensors - Communication software (such as Bluetooth or Wi-Fi) - Mobile/desktop application for remote control and monitoring light module		

Assumptions:	***1)*** *To avoid overflow of current resistors should be used so that the LED lights don't get burnt out. The resistor was used so that the LED lights would remain in a good position and they don't get burnt out.* ***2)*** *Ultra sonic sensor's detect the light module and turn on or off the light when student or teacher enter or leave the classroom.* ***3)*** *Internet connectivity for proper functioning of light.* **4)** Reliable power supply
Pre-Requisite:	**1)** Basic knowledge of Arduino **2)** Arduino Board **3)** Light sensor **4)** LEDs/ Relay Module **5)** Circuit Design **6)** Arduino IDE **7)** Programming **8)** Testing **9)** Precautions to prevent short circuits, overheating, or electrical hazards.

Step No.	Execution description	Procedure result
1 2 3 4 5	Existence Absence Hardware (arduino, ultra sonic sensor's) Interface	*With the help of ultra sonic sensors and arduino light module will automatically turn on and off depands upon the presence and absence of any person in the classroom. Arduino IDE software in used for the software testing of light module testing code is being tested properly it is error free and working as per described*

Comments

No errors in their infrastructure and working successfully

	Passed Failed Not Executed

16. Traceability Matrix

Sr. no.	Requirement ID	Use Case ID	GUI ID	Test case ID	Status	Date Completed
1	RQ001	SCS-TC-003 (Fingerprint door lock)	Nil	TC001	Failed	20-04-2023
2						
3	RQ002 SCS-RQ-003	Fan Module SCS-UC-003 (Light Module)	Nil Nil	TC002 SCS-TC-003	Passed Passed	21-04-2023 23-04-2023

17. Major Deliverables

1. Test Plan Strategy
2. Schedule
3. Control Activities
4. Test Case Design and Description

18. Risks and Assumptions

The smart classroom system using Arduino has been thoroughly designed and planned, taking into account the necessary hardware components, software tools, and development environments. The Arduino programming language and associated libraries are appropriate for developing the required functionality for the smart classroom system.

The smart classroom system has been designed to be user-friendly and easy to use for both teachers and students. The system has been thoroughly tested, including unit testing and integration testing, to ensure that all components work together as expected.

The smart classroom system is compatible with existing infrastructure, including network connectivity and power supply. The system has been designed with security in mind and includes appropriate measures to protect against unauthorized access or data breaches.

Risks

- Technical challenges may arise during the development and implementation of the smart classroom system, which are hardware compatibility issues or software bugs.
- The system may not be scalable or adaptable to changing requirements, resulting in the need for significant redesign or reimplementation.
- The system may not meet the needs and expectations of all stakeholders, resulting in a lack of adoption or usage.
- The system may not be reliable or robust, leading to downtime or data loss.
- Security vulnerabilities may be present in the system, leading to unauthorized access or data breaches.
- The smart classroom system may require significant resources and expertise to maintain and support over time, including hardware and software updates and troubleshooting.
- The cost of implementing the smart classroom system using Arduino may be higher than expected, resulting in budget overruns or delays in implementation.

Exit Criteria

Exit Criteria for Smart Classroom using Arduino:

- All the functions identified in the requirement document should be present and working as expected.
- All high priority problems reported during the testing phase should be resolved and no high priority problems should be open.
- There should be no more than three medium priority problems open.
- All the components of the smart classroom system, including the Arduino board, sensors, and other hardware components, should be functioning properly.
- The smart classroom system should be able to perform all the desired actions and functionalities as specified in the requirement document.
- The system should be stable and reliable, and it should be able to operate without any major issues or errors.

- The system should meet the performance and scalability requirements as specified in the requirement document.
- The system should be thoroughly tested and validated to ensure that it meets all the quality and functionality requirements.
- The test results should be reviewed and approved by the stakeholders before the system is considered ready for deployment.
- A final test report should be prepared that documents the testing approach, test results, and any issues encountered during the testing phase.

19. References

Ref. No.	Document Title	Date of Release/ Publication	Document Source
PGBH01-2023-RS	Git hub project details	May 14, 2023 – May 21, 2023	https://github.com/digiworld01/Smart_Irrigation_Robot
PGBH02-2023-RS	An electronic irrigation system using IoT and neural networks	Vol. 9, No. 4, December 2021	**https://pdfs.semanticscholar.org/7025/6dd367fa704536070799536b798b115100b9.pdf?_ga=2.155354555.1201499504.1673364410-872327970.1673364410**
PGBH03-2023-RS	Intelligent IoT Based Automated Irrigation System	SSN 0973-4562 Volume 12, Number 18 (2017)	https://www.ripublication.com/ijaer17/ijaerv12n18_33.pdf
PGBH04-2023-RS	Plant autonomous mobile robot	JULY 2012	https://www.researchgate.net/publication/260005632_Plant_Watering_Autonomous_Mobile_Robot
Ref. No.	Document Title	Date of Release/ Publication	Document Source
PGBH01-2003-RS	Git hub project details	May 14, 2023 – May 21, 2023	https://github.com/Almasfati/smart-classroom
PGBH02-2003-RS	Mantis bug tracker report	14-05-2023 to 21-05-2023	https://mantis.klickwash.net/my_view_page.php
PGBH03-2003-RS	Door Lock System Using Fingerprint	15 February 202	file:///C:/Users/PC/Downloads/042.pdf

20. Appendices

Appendix A

Arduino Board: The heart of the system, controlling the interactions between various components.
Arduino IDE The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.
Arduino Code: Include the complete code or a snippet of the Arduino sketch used to program the plant watering robot. You can provide a detailed explanation of the code, highlighting the different functions and their purpose.
Actuator Specifications: Provide specifications and relevant details for the actuators used in the robot, such as the water pump or solenoid valve. Include information about voltage ratings, current requirements, and any necessary wiring instructions.
Assembly Instructions: If the construction of the robot involves any specific steps or techniques, include a detailed assembly guide or instructions. This can include images or diagrams to assist in the assembly process.
Arduino Board: The heart of the system, controlling the interactions between various components.
Arduino IDE The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

APPENDIX C

Circuit Diagram: Present a circuit diagram of the electronic components used in the plant watering robot. This diagram should illustrate the connections between the Arduino board, sensors, actuators, power supply, and any other relevant components. Include labels for each component and clear wiring instructions.

Component List: Present a comprehensive list of all the components used in the plant watering robot. Include details such as part numbers, quantities, and sources. This list can serve as a reference for anyone looking to replicate or modify the robot.

Calibration Procedures: If the robot requires calibration of sensors or actuators, provide step-by-step instructions on how to perform the calibration. Include details on any necessary tools or equipment and the expected outcomes of the calibration process.

Appendix D

Data Logging and Analytics: Arduino can collect data from various sensors and systems in the smart classroom. This data is to be logged and analyzed to gain insights into usage patterns, energy efficiency, and classroom optimization
Data Logging and Analysis: If the robot collects and logs data during its operation, include information on how to access and analyze the data. Provide details on the data format, storage location, and any tools or software required for data analysis.

Appendix F

Fingerprint door lock: Arduino integrate with security devices such as door locks, surveillance cameras, and intrusion detection sensors. It monitor and secure the classroom by providing access control and real-time monitoring.

APPENDIX P

Power Supply Information: If the robot uses batteries or a specific power supply, include details about the power requirements, voltage levels, and charging or power management procedures.

APPENDIX T

Troubleshooting Guide: Provide a troubleshooting guide that includes common issues that may arise during the operation of the plant watering robot and their possible solutions. Include any error codes, warning signs, or debugging techniques that can help identify and resolve problems.

Appendix S

Sensors: Various sensors be integrated into the smart classroom, such as light sensors, fan and fingerprint door lock sensors. These sensors enable automation and data collection for monitoring and optimizing the classroom environment.

Smart Lighting: Arduino control the classroom lighting based on ambient light conditions or occupancy. It adjust the brightness and color temperature of the lights for optimal learning conditions.

Smart Fan: Arduino control the classroom fan based on ambient fan conditions or occupancy. It can adjust the regulator for optimal learning conditions.

Safety Considerations: Include a section on safety considerations specific to the plant watering robot. Highlight any potential hazards, safety precautions, and guidelines for safe operation, especially if the robot interacts with water or electricity.

Sensor Data Sheets: Include data sheets or technical information for the sensors used in the robot. This can include specifications, pin configurations, interfacing details, and any calibration or setup instructions specific to each sensor