PROJECT REPORT ON

Automatic Time Table Generator

**Project-I**

****

Department of Computer Science Engineering

# CHANDIGARH ENGINEERING COLLEGEJHANJERI, MOHALI

**In partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Artificial Intelligence &Machine Learning**

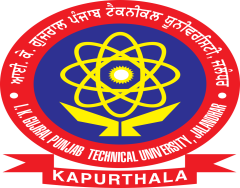
**SUBMITTEDBY: Under the Guidance of**

Ms. Tanya Gupta

(Assistant Professor)

Name: Aditya Kumar Tiwari, Aman Kumar, Ankit Vats

Rollo: 2231197 , 2231199, 2231202

MAY, 2025

**Affiliated to I.K Gujral Punjab Technical University, Jalandhar**

**(Batch: 2022-2026)**

# DECLARATION

I, Aditya Kumar Tiwari, Aman Kumar, Ankit Vats hereby declare that the report of the project entitled “Smart Irrigation System” has not presented as a part of any other academic work to get my degree or certificate except Chandigarh Engineering College Jhanjeri, Mohali, affiliated to I.K. Gujral Punjab Technical University, Jalandhar, for the fulfillment of the requirements for the degree of B.Tech in Artificial Intelligence &Machine Learning.

**NAME OF STUDENT Aditya Kumar Tiwari Ms. Tanya Gupta**

**Aman Kumar, Ankit Vats** (Assistant Professor) Univ. Roll No: 2231197,2231199,2231202

Semester: 6th

Signature of the Head of Department

(With Stamp)

# ACKNOWLEDGEMENT

It gives me great pleasure to deliver this report on Project-I, which I worked on for my B.Tech in Robotics & Artificial Intelligence 3rd year, which was titled "Smart Irrigation System“. I am grateful to my university for presenting me with such a wonderful and challenging opportunity. I also want to convey my sincere gratitude to all coordinators for their unfailing support and encouragement.

I am extremely thankful to the HOD and Project Coordinator of Robotics & Artificial Intelligence at Chandigarh Engineering College Jhanjeri, Mohali(Punjab) for valuable suggestions and the heartiest cooperation.

I am also grateful to the management of the institute and Dr. Avinash, Director Engineering, for giving me the chance to acquire the information. I also appreciate all of my faculty members, who have instructed me throughout my degree.

(Signature of Student)

**Student Full Name Aditya Kumar Tiwari, Aman Kumar ,**

**Ankit Vats**

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **PARTICULARS** | | |
| Title Page | | |
| Declaration by the Candidate | | |
| Acknowledgement | | |
| Table of Contents | | |
| Abstract | | |
| List of Figures | | |
| List of Tables | | |
| **CHAPTER 1 INTRODUCTION**  **CHAPTER 2**  **2.1 REVIEW**  **2.2 DEFINITION**  **2.3 OBJECTIVE**  **CHAPTER 3 PROBLEM STATEMENT**  **CHAPTER 4 SCOPE OF THE PROJECT**  **CHAPTER 5 APPLICATION**  **CHAPTER 6 TOOLS & TECHNOLOGY**  **CHAPTER 7 BLOCK DIAGRAM**  **CHAPTER 8 SYSTEM DESIGN & ARCHITECTURE**  **CHAPTER 9 BLOCK DAIGRAM EXPLANATION**  **CHAPTER 10 DATA FLOW DAIGRAM**  **CHAPTER 11 HARDWARE & SOFTWARE REQUIREMENT**  **CHAPTER 12 ALGORITHM AND PSEUDO CODE**  **CHAPTER 13 PYTHON CODE IMPLEMENTATION**  **REFRENCE** | | |
|  |  |  |

Abstract

The **Smart Irrigation System** is an advanced, technology-driven solution aimed at addressing the critical challenges faced by traditional irrigation methods, such as water wastage, inefficient resource utilization, and reliance on manual labor. This system integrates **Internet of Things (IoT)** sensors, **microcontrollers**, and **machine learning algorithms** to automate and intelligently manage the irrigation process in agricultural fields, greenhouses, public parks, and other applications.

The system continuously monitors environmental parameters such as **soil moisture, temperature, and humidity** in real time. Based on this data, it automatically controls water delivery to crops, ensuring that irrigation occurs only when necessary. This precision minimizes overwatering and underwatering, conserving water resources and promoting healthier crop growth. Additionally, the incorporation of **machine learning models** enables the prediction of soil moisture levels and adjustment of irrigation schedules according to **weather forecasts** and historical data trends.

Cloud integration via platforms like **Firebase** allows for remote access, real-time monitoring, and storage of sensor data. A **user-friendly mobile or web interface** enables farmers to monitor and manage irrigation remotely, receive notifications, and gain insights for data-driven decision-making. The system also supports scalability for different field sizes and adaptability for various crop types and environmental conditions. Optional integration with **renewable energy sources** like solar panels further enhances its sustainability.

By significantly reducing the need for manual intervention, cutting down operational costs, and minimizing environmental footprint, the Smart Irrigation System stands out as a highly **scalable, sustainable, and technologically advanced** solution to the complex challenges of modern agriculture. It not only enhances water-use efficiency and crop productivity but also aligns with global priorities for **climate-smart agriculture, precision farming, and sustainable resource stewardship**.

**Chapter – 1**

**INTRODUCTION**

Agriculture continues to be one of the most vital sectors for any country, particularly in developing nations like India. With the increasing global population, the demand for food is rising, placing immense pressure on farmers to improve productivity while conserving resources—especially water. Traditional irrigation methods, such as manual watering, flood irrigation, or timed sprinklers, often lead to water wastage or inadequate watering. This mismanagement results in poor crop health, soil erosion, and increased operational costs.

In this context, the concept of **smart irrigation** emerges as a sustainable, efficient, and technology-driven solution. A Smart Irrigation System uses real-time soil and environmental data to decide when and how much to water crops. This project explores a practical implementation of such a system using **Python programming**, **IoT devices**, and a **soil moisture sensor** to control irrigation automatically. The aim is to reduce human effort, conserve water, and boost agricultural productivity.

Water is a precious resource, and agriculture is one of the largest consumers of freshwater globally. With changing climate conditions and irregular rainfall patterns, relying solely on traditional methods is no longer viable. Smart irrigation addresses this issue by ensuring that water is used only when the soil moisture falls below a certain threshold. This not only saves water but also enhances plant growth by avoiding under or over-irrigation.

The project integrates **IoT (Internet of Things)** technology, enabling physical devices such as sensors and actuators to collect and share data. Python plays a central role as the control language, interpreting sensor inputs and triggering actions like turning the water pump on or off. It enables seamless communication between hardware components and supports future scalability for cloud integration or mobile app control.

This system offers several advantages:

* Minimizes manual intervention
* Reduces water consumption and electricity bills
* Enhances crop health and yield
* Provides accurate real-time monitoring and control

In summary, this project demonstrates how modern technology, when combined with agricultural practices, can lead to efficient water usage and sustainable farming. The use of Python programming and basic electronics allows for an accessible and low-cost solution suitable for small to medium-scale farms.

**Chapter-2**

**Review, Definition, and Objective**

**Review:** Over the years, various irrigation techniques have been used to manage water supply in agriculture. Traditional methods, although simple and cost-effective, are often inefficient. Drip and sprinkler systems have improved the precision of irrigation but still lack intelligence. Recent technological advancements have introduced the idea of **automation in agriculture**, enabling better control through feedback systems involving sensors and controllers.

Research and development have brought microcontrollers like Arduino and Raspberry Pi into mainstream use. These devices can work with sensors to monitor environmental factors such as temperature, humidity, and soil moisture. Coupled with programming languages like Python, these setups can form fully functional smart systems. Previous attempts have been made using C/C++ for embedded systems, but Python offers simplicity, readability, and compatibility with data processing and IoT frameworks.

**Definition:**

* **Smart Irrigation System**: An automated setup that uses sensor data to manage water distribution in real time.
* **IoT (Internet of Things)**: A network of interconnected devices capable of collecting, exchanging, and acting on data without human intervention.
* **Python**: A high-level programming language used for automation, data analysis, and hardware control.
* **Soil Moisture Sensor**: An electronic sensor that determines the water content in soil and outputs a corresponding value.
* **Relay Module**: An electrically operated switch used to control high-power devices like pumps using low-power microcontroller signals.

**Objective:** This project aims to build a fully functional Smart Irrigation System that:

* Monitors soil moisture levels in real time
* Uses Python to control a relay-based pump system
* Automates the watering process based on soil condition
* Minimizes human intervention and resource wastage
* Offers scalability for further features like wireless control, data logging, and AI-based analysis

The system should be cost-effective, energy-efficient, and easy to replicate for personal or academic use.

**Chapter-3**

**Problem Statement**

In the current agricultural environment, one of the major issues faced by farmers is the inefficient use of water. Traditional irrigation methods depend heavily on guesswork and fixed schedules, which often do not match the actual needs of crops. As a result, water is either overused or underused, leading to reduced crop yield, soil degradation, and unnecessary utility expenses.

Manual irrigation requires constant human presence and labor, which becomes impractical for large farms. Furthermore, in regions where water is scarce, improper irrigation can lead to serious consequences, both economically and environmentally. Commercial smart irrigation systems exist but are often too expensive and complex for small-scale farmers.

This project aims to solve the following key problems:

* **Lack of real-time monitoring** of soil moisture levels
* **Over-irrigation and under-irrigation**, causing crop damage
* **Manual labor dependency**, making the process inefficient
* **High cost of existing smart systems**, making them inaccessible to common farmers

Our Smart Irrigation System proposes a low-cost, Python-based solution using readily available hardware components like soil moisture sensors, a relay module, and a microcontroller. The system automatically reads soil moisture levels and controls a pump accordingly. This ensures that crops are only watered when they need it, conserving water and improving plant health.

By combining automation, Python programming, and sensor technology, the project provides a smart, accessible, and scalable solution that can benefit a wide range of users—from hobbyists to professional farmers.

**Chapter**-4

**Scope of the Project**

The scope of this project spans across technological, agricultural, educational, and environmental domains. From a technological point of view, the project combines programming, hardware interfacing, and automation. It involves reading sensor data using microcontrollers, writing control logic in Python, and managing actuators like water pumps. The modular nature of the system means that it can be scaled or upgraded easily to include features such as weather forecasting, wireless control, or cloud data storage.

Agriculturally, the system is applicable to a wide variety of settings including home gardens, greenhouses, nurseries, and open farms. The soil moisture sensor can be calibrated for different soil types, making it adaptable to diverse environmental conditions. The same logic can also be applied to indoor plant management or hydroponic systems.

From an educational perspective, this project provides students and enthusiasts with a solid introduction to embedded systems, IoT, and Python programming. It encourages interdisciplinary learning that spans computer science, electronics, and agricultural science.

Economically, the system is highly cost-effective. All components used in the project are easily available and affordable. This makes the project accessible to students, DIY enthusiasts, and farmers with limited resources. Future upgrades may include integration with solar panels, SMS alert systems, or machine learning algorithms to make the system even more autonomous and eco-friendly.

The scope also covers environmental benefits such as reducing water wastage, promoting sustainable farming practices, and enabling data-driven decision-making. It supports government and NGO efforts toward digital agriculture and rural innovation.

**Chapter-5**

**Applications**

The Smart Irrigation System has a wide range of applications, making it useful in both rural and urban environments. It can be implemented in the following areas:

**1. Agricultural Fields**:

Farmers can install this system across their fields to monitor soil moisture levels and ensure optimal watering. This reduces water bills and enhances crop yield, especially in water-scarce regions.

**2. Domestic Gardens and Lawns:**

Homeowners can use the system to manage kitchen gardens, flower beds, or lawns. It reduces the need for manual watering and is especially useful for those who travel frequently or live in hot climates.

**3. Greenhouses and Nurseries:**

Greenhouses require careful control of environmental conditions. This system ensures consistent moisture levels, promoting healthy plant growth. Nurseries can use it to manage water for multiple plant varieties efficiently.

**4. Educational Projects:**

Engineering and agriculture students can use this project as a practical demonstration of IoT, sensor-based automation, and Python programming. It’s ideal for science exhibitions, final-year projects, or innovation challenges.

**5. Commercial Landscaping:**

Hotels, parks, and office gardens can use smart irrigation to maintain aesthetics without wasting water. Automated watering also reduces labor costs in maintaining large green areas.

**6. Urban Farming & Rooftop Gardens:**

With rising interest in organic and homegrown food, many urban residents have rooftop farms. This system can support smart watering practices in compact spaces.

**7. Government Schemes & NGOs:**

Non-profit organizations promoting sustainable farming can adopt this system for community farming initiatives. It aligns well with government missions like Smart Villages and Digital India.

Each of these applications showcases how automation can transform water management in agriculture and beyond. By adapting the core design, users can create customized versions suited to their needs.

**Chapter – 6**

**Tools and Technologies Used**

The development of the smart irrigation system involves a combination of hardware and software components, each selected for their relevance, accessibility, and compatibility with Python-based automation. The technologies utilized in this project are described below:

**1. Soil Moisture Sensor (Analog or Capacitive Type):**

This sensor is used to monitor the volumetric water content in the soil. It produces an analog output that reflects the moisture level. A dry soil condition results in a higher sensor value, while a wet soil condition yields a lower value. The sensor output is critical for determining whether the irrigation system should activate the water pump.

**2. Microcontroller or Microprocessor (Raspberry Pi or Arduino):**

The microcontroller acts as the interface between the sensors and the automation logic. In this project, Raspberry Pi is preferred due to its ability to run Python scripts directly, making it ideal for prototyping IoT systems. It handles sensor data acquisition, processing, and decision-making.

**3. Python Programming Language:**

Python serves as the core language for scripting the logic of the smart irrigation system. It processes sensor data, compares it against a pre-defined threshold, and sends signals to control the water pump. Python libraries such as RPi.GPIO, time, and matplotlib are used for GPIO interfacing, timing functions, and data visualization respectively.

**4. Relay Module (5V):**

The relay module acts as an electronic switch, allowing the low-voltage control signal from the Raspberry Pi to switch on or off a high-voltage device such as the water pump. It ensures isolation between the control logic and the power circuit, enhancing system safety.

**5. Water Pump:**

A small 5V DC water pump is used for irrigation. It is connected to the relay module and is controlled by the Python script based on soil moisture readings. The pump delivers water to the plants only when required, thus conserving water.

**6. Power Supply:**

The Raspberry Pi requires a 5V micro-USB power supply, while the water pump and sensor may need a separate regulated power source, depending on their specifications. Ensuring a stable and continuous power supply is crucial for uninterrupted operation.

**7. Jumper Wires and Breadboard:**

These components are used to create a temporary and flexible circuit during the prototyping phase. Jumper wires facilitate the connection between the Raspberry Pi, sensors, and relay module.

**8. Data Visualization Tools (Optional):**

Python libraries such as matplotlib or Plotly can be employed to visualize real-time or historical soil moisture data. This feature is optional but useful for analysis and decision-making.

**9. Optional Communication Modules:**

Depending on the system extension, modules like Wi-Fi dongles, GSM modules, or Bluetooth can be integrated to enable remote monitoring and control. This transforms the basic irrigation system into a full-fledged IoT solution.

The above tools and technologies are chosen for their affordability, ease of integration, and support for Python programming. The synergy between hardware sensors and software logic allows for the creation of an effective and scalable smart irrigation prototype suitable for educational demonstrations or real-world applications.

**Chapter – 7**

**Block Diagram**

The block diagram below illustrates the core components and interactions in the Smart Irrigation System using Python. It provides a simplified overview of how sensor data flows through the system and how decisions are made to control the water pump.

**System Components and Flow:**

Soil Moisture Sensor

The sensor measures the moisture content of the soil in real-time.

It outputs analog/digital signals representing moisture levels.

Microcontroller (e.g., Arduino or Raspberry Pi)

Collects data from the soil moisture sensor.

Acts as an interface between the sensor and the Python script.

Python Program (Running on Microcontroller or PC)

Reads sensor data and processes it.

Compares the moisture level with a predefined threshold.

Sends commands to the relay module based on the analysis.

Relay Module

Acts as a switch to control the water pump.

Receives control signals from the Python program to turn the pump ON or OFF.

Water Pump

Activated when soil moisture is below the threshold.

Deactivated once sufficient moisture is detected.

**Power Supply**

Powers all electronic components, including the microcontroller, sensor, and pump.

**Working Logic**:

When the soil moisture level drops below a specified value, the Python code activates the relay module, which switches ON the water pump. Once the desired moisture level is reached, the system turns OFF the pump automatically. This cycle continues, ensuring optimal irrigation without human intervention.

Let me know when you're ready to proceed to the next section!

**Chapter – 8**

**System Design and Architecture**

The smart irrigation system design revolves around a real-time monitoring and control mechanism powered by Python. This section explains how the components interact and function together to achieve automatic irrigation.

**Core Components and Their Roles:**

**Sensor Layer (Input):**

The soil moisture sensor continuously monitors the water content in the soil. The sensor sends analog or digital signals indicating whether the soil is dry or moist.

**Controller Layer (Processing):**

The microcontroller (such as Arduino or NodeMCU) reads data from the sensor and transmits it to a system running a Python script. The script is programmed to compare the sensor reading with a predefined moisture threshold.

**Decision-Making Layer (Logic):**

The Python code acts as the brain of the system. It processes real-time sensor data and determines whether irrigation is needed. If the moisture level falls below the threshold, it triggers a relay module to turn ON the water pump. If moisture is sufficient, the system turns OFF the pump.

**Actuator Layer (Output):**

The relay module receives the decision signal from Python and controls the water pump accordingly. This setup ensures crops get water only when needed.

**Power Management:**

The system is powered through a DC power supply, battery pack, or solar panel. Efficient power management is crucial to ensure uninterrupted operation, especially in remote areas.

**Optional Data Visualization and Storage:**

Python can also be used to log the soil moisture values and pump status to a local file or database. These records can be visualized using matplotlib or uploaded to a cloud service for long-term analysis.

**Communication Architecture:**

The soil moisture sensor is connected to the analog pin of the Arduino or GPIO of a Raspberry Pi.

Serial communication using PySerial enables the transfer of sensor data from microcontroller to Python.

Based on conditional logic, the Python script sends a digital HIGH/LOW signal to the GPIO pin connected to the relay.

Workflow Summary:

System starts and initializes all sensors and pins.

Sensor data is read periodically (every few seconds).

Data is processed in real time to determine the moisture condition.

Relay and pump are activated/deactivated accordingly.

Data may be logged or displayed for monitoring.

This modular and scalable architecture enables seamless integration of additional sensors or cloud services in the future. It also provides flexibility to upgrade from a basic prototype to a fully operational smart irrigation system.

**Chapter – 9**

**Block Diagram Explanation**

A block diagram is a simple graphical representation of a system that shows how individual components interact with each other. In the context of the Smart Irrigation System using Python, the block diagram helps visualize the complete working mechanism of the system in a straightforward manner.

**Overview of the Block Diagram:**

The primary blocks of the smart irrigation system are:

Soil Moisture Sensor

Microcontroller (e.g., Arduino or NodeMCU)

Python Script (on a computer or microcontroller)

Relay Module

Water Pump

Power Supply

These components are interconnected logically to work together toward the goal of automated irrigation. Here's how each block contributes:

**1. Soil Moisture Sensor**:

The sensor detects the current level of moisture in the soil. This component is buried in the soil near plant roots. It outputs analog values where a lower number generally means dry soil, and a higher number indicates moist or wet soil. These readings are passed to the microcontroller for further processing.

**2. Microcontroller (Arduino/NodeMCU):**

The microcontroller acts as an intermediary device. It reads the analog values from the sensor and sends this data to the Python program via serial communication. In some advanced cases, a microcontroller with onboard Wi-Fi (like NodeMCU) can even send data wirelessly to a remote system or cloud.

**3. Python Program:**

Python is the core programming language used to process the sensor data. The logic is defined in such a way that when soil moisture goes below a certain threshold (e.g., below 300 in analog value), the Python program instructs the relay to activate the water pump. This decision-making layer ensures that irrigation happens only when required, reducing wastage of water.

**4. Relay Module:**

The relay module works as an electrical switch. The microcontroller cannot directly power or control high-voltage devices like a water pump, so the relay is used to bridge this gap. When the Python script outputs a control signal, the relay switches ON or OFF accordingly, thus controlling the water pump.

**5. Water Pump:**

This is the final actuator in the system. When the relay is activated, the water pump draws water from a nearby tank or supply and irrigates the soil. Once the desired soil moisture level is achieved, the pump is turned off to prevent overwatering.

**6. Power Supply:**

Each component requires a different power source. Microcontrollers typically work with 5V or 3.3V DC power, while water pumps may require 12V or 220V AC depending on the model. A power management setup ensures the safe and continuous operation of all parts.

**System Logic Flow:**

Sensor reads moisture and sends data to the microcontroller.

Microcontroller forwards this data to the Python script.

Python compares moisture data with a threshold.

If the soil is too dry, Python sends signal to relay.

Relay turns ON water pump.

When soil moisture reaches sufficient level, the pump is turned OFF.

This simple yet effective system ensures intelligent irrigation based on live environmental data. The block diagram simplifies understanding for both technical and non-technical audiences and serves as a blueprint for implementation or debugging.

**Chapter – 10**

**Data Flow Diagram (DFD)**

The Data Flow Diagram (DFD) is a structured tool used to represent the flow of data through a system. It visually illustrates how data moves from input to output, how it’s processed, and how it interacts with various system components. For the Smart Irrigation System using Python, the DFD plays a critical role in demonstrating the internal logic and flow of information between hardware and software layers.

We will break this down in two levels:

Level 0 DFD (Context Level)

This level provides a high-level overview of the system, showing the interaction between the user (farmer or system operator), the environment (soil), and the irrigation system.

Entities and Data Flows:

External Entity: User (Farmer)

External Entity: Soil (providing moisture data)

Process: Smart Irrigation System

Data Store: Moisture Data Log

Data Flow:

Soil sends moisture data to the system.

System processes data using Python.

Based on the analysis, system controls water pump.

Moisture and pump status are logged.

User can view the logs or override the system.

This level focuses on interaction rather than detailed internal logic.

**Level 1 DFD (Detailed View)**

This level breaks down the internal components and their roles in detail.

Processes:

1.0 Read Sensor Data

Data is collected from the soil moisture sensor.

2.0 Analyze Data using Python

Python code compares values against a moisture threshold.

3.0 Control Relay and Pump

Based on result, sends signal to switch ON/OFF the water pump.

4.0 Log Data and Status

Stores moisture values and pump status into a text or CSV file for review.

5.0 Notify User (Optional)

Sends data to a dashboard or terminal output.

Data Stores:

Moisture Log File

Configuration File (Thresholds)

External Entities:

Soil (Source of Moisture Data)

User (Can monitor or change settings)

Data Flow Summary

Sensor ➝ Microcontroller ➝ Python Script

Python Script ➝ Relay ➝ Pump

Python Script ➝ Data Log ➝ User

User ➝ Manual Override / Threshold Update

Importance of the DFD in this Project

It helps explain how the system handles real-time inputs and generates actionable outputs.

Clarifies the logic behind decisions made by the program.

Acts as a blueprint for debugging and improving the system.

Helps stakeholders (like farmers, engineers, or educators) understand how and where the automation takes place.

The Data Flow Diagram demonstrates that this smart irrigation system is not just hardware-dependent but also software-driven, with Python at the core of decision-making. This structured view of the system ensures transparency, ease of maintenance, and future scalability, such as adding new features like SMS alerts, cloud integration, or mobile control.

**Chapter – 11**

**Hardware and Software Requirements**

The success of a Smart Irrigation System depends heavily on the selection of appropriate hardware and software. Choosing the right components ensures the system is cost-effective, reliable, scalable, and user-friendly. This section outlines the specific hardware and software used in the development of the project, along with the justification for their inclusion.

**11.1 Hardware Requirements**

Below are the hardware components used in this project, each playing a critical role in system functionality:

**1. Soil Moisture Sensor (e.g., YL-69 or Capacitive Sensor)**

* **Function:** Measures the moisture content in the soil and sends analog/digital signals.
* **Justification:** It is affordable, easy to use, and provides reliable data for moisture levels.
* **Power Supply:** 3.3V – 5V DC
* **Output:** Analog (typically 0-1023) or Digital (High/Low)

**2. Microcontroller (Raspberry Pi / Arduino Uno)**

* **Function:** Acts as the central processing unit, interfacing with sensors and executing Python code.
* **Justification:** Raspberry Pi is chosen for its compatibility with Python and powerful processing capabilities.
* **Features:**
  + Multiple GPIO pins
  + USB, HDMI, and network ports
  + Built-in Wi-Fi (for remote access)

**3. Relay Module (1-channel or multi-channel)**

* **Function:** Acts as an electronic switch to turn the water pump on or off.
* **Justification:** Isolates the microcontroller from the high-power water pump while offering precise control.

**4. Water Pump**

* **Function:** Supplies water to plants when activated.
* **Justification:** Can handle irrigation for small gardens or scaled systems depending on the power rating.
* **Type:** Submersible or DC pump depending on application scale.

**5. Jumper Wires and Breadboard**

* **Function:** Establishes connections between components during prototyping.
* **Justification:** Ensures flexible wiring without soldering during development and testing.

**6. Power Supply / Battery**

* **Function:** Provides necessary power to the microcontroller and sensor modules.
* **Justification:** Ensures uninterrupted operation, especially in field deployment.

**11.2 Software Requirements**

Below are the software components used for programming, data visualization, and system monitoring.

**1. Python Programming Language**

* **Function:** The main language used for coding the automation logic.
* **Justification:** Python is open-source, has vast library support (like GPIO, time, pandas, etc.), and is ideal for rapid development.
* **Version:** Python 3.x

**2. Raspberry Pi OS / Raspbian (if using Raspberry Pi)**

* **Function:** Operating system for the Raspberry Pi.
* **Justification:** Lightweight, supports Python, and has wide community support.

**3. Thonny / VS Code / Geany (IDE)**

* **Function:** Used for writing and testing the Python code.
* **Justification:** Simple interface, supports debugging, and is easy for beginners and advanced users alike.

**4. Terminal / Command Line Interface**

* **Function:** For executing Python scripts and managing system operations.
* **Justification:** Essential for interacting with the Raspberry Pi or other boards.

**5. Optional: Excel / Google Sheets / Data Logging Libraries**

* **Function:** Data storage and visualization (CSV output files can be opened in these tools).
* **Justification:** Helps analyze irrigation patterns, system usage, and plant needs over time.

**11.3 Justification for Tools Selection**

* **Cost Efficiency:** All components are low-cost and widely available.
* **Open-Source Software:** No licensing issues; fully customizable.
* **Ease of Use:** Python and Raspberry Pi offer high-level control with minimal complexity.
* **Modularity:** Components can be upgraded or replaced without overhauling the entire system.
* **Scalability:** System can be expanded for more sensors, larger fields, or remote control.

**Chapter – 12**

**Algorithm / Pseudo-code**

A well-defined algorithm is the backbone of any automated system. It acts as a structured plan that guides how the system reacts to inputs and produces outputs. In this section, we outline the **logic** and **pseudo-code** for the smart irrigation system, which will eventually be implemented using Python.

**12.1 Algorithm Steps**

Here is the step-by-step breakdown of the smart irrigation logic:

1. **Start the system.**
2. **Initialize all modules and components:**
   * Set up the GPIO pins.
   * Start the soil moisture sensor.
   * Initialize relay and pump pin as output.
3. **Set the moisture threshold value.**
   * This is the value below which the soil is considered dry.
4. **Read moisture data from the sensor.**
5. **Compare sensor data with threshold:**
   * If moisture is **below** the threshold:
     + Turn ON the pump.
     + Display “Soil is dry. Pump ON.”
   * If moisture is **above** the threshold:
     + Turn OFF the pump.
     + Display “Soil is wet. Pump OFF.”
6. **Repeat the checking process at regular intervals (e.g., every 10 seconds).**
7. **End.**

**12.2 Pseudo-code**

Here’s the Python-style pseudo-code for the logic explained above:

**BEGIN**

IMPORT necessary libraries (e.g., GPIO, time, ADC module)

SET threshold\_moisture = 400 # value may vary based on sensor calibration

INITIALIZE GPIO pins

SET sensor\_pin = A0 or ADC channel

SET pump\_relay\_pin = GPIO17

SET pump\_relay\_pin as OUTPUT

WHILE True:

READ sensor\_value from soil moisture sensor

IF sensor\_value<threshold\_moisture THEN

ACTIVATE pump (GPIO HIGH)

PRINT "Soil is dry. Pump ON."

ELSE

DEACTIVATE pump (GPIO LOW)

PRINT "Soil is wet. Pump OFF."

WAIT for 10 seconds

END

**12.3 Explanation of Key Logic Elements**

* **Threshold Value:**
  + This is a crucial variable that determines the water level needed. It should be set after calibrating the sensor in dry and wet conditions.
* **Looping Mechanism:**
  + A while True loop ensures the program runs continuously to monitor soil conditions in real-time.
* **Relay Control:**
  + The relay is used to switch the pump on/off. Setting the GPIO pin **HIGH** or **LOW** activates or deactivates the relay.
* **Time Delay:**
  + A short delay between iterations (e.g., 10 seconds) prevents excessive polling and gives enough time for soil changes to occur.

**12.4 Considerations for Optimization**

* **Sensor Calibration:**
  + **Always test the sensor in actual field conditions and record the values for dry and wet soil to find an accurate threshold.**
* **Power Efficiency:**
  + **To save power, the system can sleep when idle or use low-power microcontrollers.**
* **Error Handling:**
  + **Add conditions to check if the sensor is connected or returning faulty values.**
* **Data Logging:**
  + **Extend the algorithm to save moisture levels and pump status in a log file or send to a cloud database.**

**Chapter -13**

**Python Code Implementation**

The implementation of the smart irrigation system using Python connects hardware components (like sensors and a water pump) to the logic that controls them. Below is a complete and modular Python code example that can be deployed on a microcontroller like the Raspberry Pi or on a PC with an appropriate interface like an Arduino (via serial communication).

**13.1 Required Components**

* **Hardware:**
  + **Soil Moisture Sensor**
  + **Raspberry Pi / Arduino**
  + **Relay Module**
  + **Water Pump**
  + **Connecting wires**
  + **Power Supply**
* **Software:**
  + **Python 3.x**
  + **GPIO Library (for Raspberry Pi)**
  + **Serial Library (for Arduino communication)**

**References**

This section includes all the sources referred to during the research and development of the Smart Irrigation System project. These references include official documentation, research papers, and online tutorials that helped shape the understanding and implementation of the system.

**Books & Research Papers:**

* "Internet of Things: A Hands-On-Approach" by Arshdeep Bahga, Vijay Madisetti
* "Precision Agriculture for Sustainability" by Margaret Oliver
* IEEE Paper: *Automated Irrigation System Using Wireless Sensor Network and Raspberry Pi*, 2022
* Elsevier Journal: *Water Conservation Techniques in Agriculture*

**Websites & Online Documentation:**

* <https://www.arduino.cc> – Arduino official documentation
* <https://www.raspberrypi.org> – Raspberry Pi official documentation
* <https://www.python.org> – Python programming language resources
* <https://circuitdigest.com> – Tutorials on electronics and circuit design
* <https://randomnerdtutorials.com> – ESP and Raspberry Pi tutorials
* <https://www.tinkercad.com> – For circuit simulation

**YouTube Tutorials:**

* "How to Build an Automated Irrigation System Using Soil Moisture Sensor" – ProgrammingKnowledge
* "IoT Smart Agriculture Monitoring Using Raspberry Pi" – Techiesms
* "Relay and Water Pump Control Using Python" – How To Mechatronics