IT AND BUSINESS COLLEGE

Department: Computer Science

Project title: Automated watering system

Team name: Team-2

Members:

Ulanbekov Ramazan - 238715015

Turarov Zhandar - 238715022

Toktomambetov Emir - 238715005

Zamirov Anvar - 238715007

Sultanov Askar - 238715012

Instructor: Nurbekov Mirlan

Group: SCA-23A

Date: 17.12.25

Table of contents

List of figures

List of tables

Abstract

We implemented an Arduino-based automatic plant watering system that measures soil moisture and activates a water pump only when plants need water. The system addresses the problem of manual watering, aiming to save water and reduce maintenance. It uses a soil moisture sensor connected to an Arduino UNO; when the sensor reading indicates dry soil, the Arduino drives a 5 V pump via a relay, and it stops watering once moisture is sufficient. A Bluetooth module provides optional manual on/off control via a smartphone and also shows values of humidity. In testing, the system reliably maintained soil moisture at the set threshold, demonstrating effective automated irrigation. Key results include consistently triggering the pump when soil readings exceeded the threshold and successfully controlling the pump remotely by phone. This project contributes to home-automation and smart-garden applications by offering a low-cost, standalone irrigation solution that can conserve water and assist busy users.

1.Introduction & Background

Freshwater is needed for crop and energy production, industrial fabrication, as well as for human and ecosystem needs. According to the AQUASTAT database (AQUASTAT, 2016), 69% of the total extracted freshwater was used by the agriculture sector, whereas 19% was used by the industrial sector and the rest was used by the domestic segment. Therefore, water was considered a critical resource for the agriculture sector to ensure future global food security.

However, the continued increase in demand for water by domestic and industrial sectors and greater concerns for environmental quality created a challenge for every country: to reduce farm water consumption while sustaining fresh food requirements. Consequently, there was an urgent need to create strategies based on science and technology for the sustainable use of water. Industrialists and researchers were working to build efficient and economical automatic systems to control water usage in order to reduce waste.

Irrigation is the artificial application of water to land for agricultural production. The water requirement of the soil depended on soil properties such as moisture and temperature. Effective irrigation could influence the entire growth process, and automation in irrigation systems using modern technology could provide better irrigation management. In general, most irrigation systems were manually operated. These traditional techniques could be replaced with automated irrigation techniques to use water more efficiently and effectively.

Traditionally, farmers had to be present in their fields to carry out the irrigation process. Nevertheless, in the context of this project, farmers often needed to manage their agricultural activities along with other occupations. A sensor-based automated irrigation system provided a promising solution for farmers where the presence of a farmer in the field was not compulsory during the irrigation process.

Arduino was a flexible, programmable hardware platform designed to control a circuit logically. The central component of the Arduino interface board was an integrated circuit chip that could be programmed using the C++ language. This microcontroller was an AVR type, which was produced by Atmel. The device could read input, process a program, and produce various outputs based on project requirements.

In this chapter, the development of an automated irrigation system based on an Arduino microcontroller was presented. In this system, a soil moisture sensor was used to detect and monitor the soil humidity for the plant. Based on the soil moisture level, the system was designed to automatically activate the water pump to irrigate the plant when the soil was too dry and turn off the pump when the soil was sufficiently wet.

1.1 History of different automated watering options

In previous works, considering automated watering techniques, it was found that Arduino-based sensors had been utilized for plant watering systems (Devika et al.) and automated irrigation systems.

An Arduino-based automatic plant watering system was proposed by Devika et al. (2014) (watch Figure 1). In this work, the authors developed a system where an Arduino microcontroller was used to control two functional components: moisture sensors and a motor/water pump to automatically water the plant. The function of the moisture sensor was to sense the level of moisture in the soil, while the water pump supplied water to the plants.

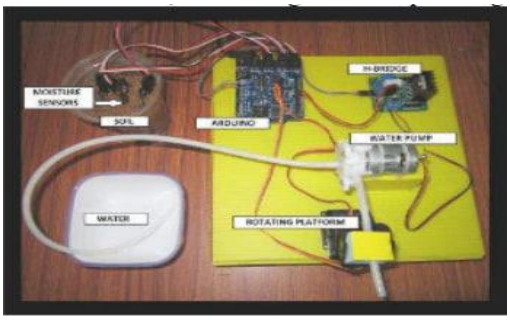


Figure 1. Option proposed by Devika et. al

Bansod et al. present a similar design using multiple moisture sensors and a submersible pump under Arduino. These and other projects typically use a microcontroller and a relay to drive a pump. They are also used humidity as value, but values was different for different soil in the code.

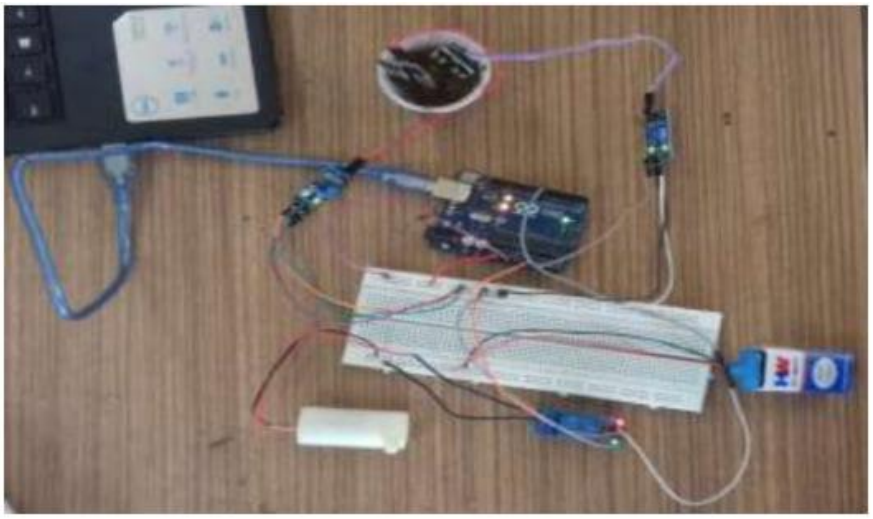


Figure 2. Option proposed by Bansod et al.

1.2 Types of irrigation systems

1.2.1 Drip irrigation system

Drip irrigation systems delivered water directly to the root zone of plants through pipes and emitters (as shown in figure 3). Water flowed slowly and continuously, which reduced evaporation and surface runoff. This system was widely applied in agriculture, greenhouses, and small gardens.
The system worked by supplying pressurized water through a network of tubes. Small openings released water in controlled amounts near each plant. In automated setups, timers or controllers regulated the watering duration and frequency.

Figure 3. Drip irrigation system

1.2.2 Sprinkler irrigation system

Sprinkler irrigation systems distributed water over plants in the form of fine droplets, simulating natural rainfall (as illustrated in figure 4). These systems were commonly used for lawns, parks, and large agricultural fields.
The system operated by pumping water through pipes to sprinkler heads. When activated, the sprinkler heads sprayed water evenly over a defined area. Automation was achieved using timers or central controllers that scheduled irrigation cycles.

Figure 4. Sprinkler irrigation system

1.2.3 Sensor based automatic irrigation system

Sensor-based irrigation systems adjusted watering based on real-time soil and environmental conditions (as presented in figure 5s). These systems were considered advanced forms of automatic irrigation and were often used in smart agriculture and home automation.
The system functioned by collecting data from soil moisture sensors. A controller processed the sensor readings and activated or deactivated water pumps or valves when predefined thresholds were reached. This method ensured that watering occurred only when necessary.

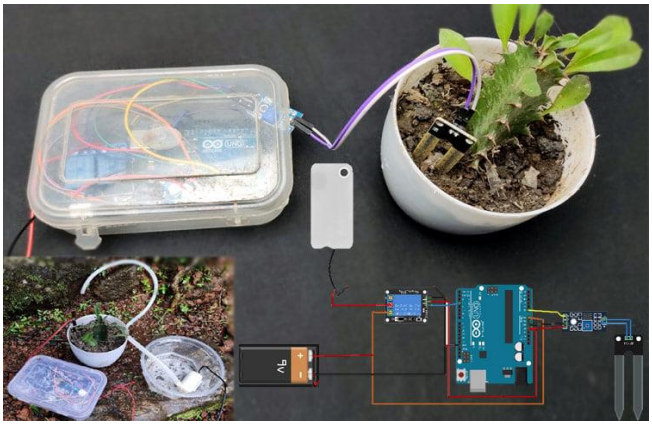


Figure 5. Sensor based automatic irrigation system

2.Problem statement and objectives

Many plant owners forget to water regularly or over-water, leading to plant stress or wasted resources. Our project's problem statement is to automate plant irrigation based on soil moisture to ensure plants receive water only when needed. The primary objectives are:

1) Automatic irrigation: use a soil moisture sensor and microcontroller to water plants without human intervention when soil is dry.

2) Manual override: implement wireless (Bluetooth) control so a user can remotely turn watering on or off via a smartphone if desired.

3) Reliability and efficiency: ensure the system is responsive and uses water efficiently, avoiding unnecessary watering (conserving water) and preventing plant under- or over-watering.

4) Low cost and simplicity: employ common, inexpensive hardware (Arduino Uno, relay, inexpensive pump/sensor) and straightforward software to keep the design accessible.

Functional requirements derived from these objectives include: the system must continuously read soil moisture, activate the pump when moisture falls below a threshold, deactivate the pump when moisture is sufficient, and respond to Bluetooth commands from a mobile app. Non-functional constraints include maintaining safe voltage levels (we use 5 V logic and motor supply), limiting overall cost (Arduino UNO and modules total under ~$20), satisfying typical Bluetooth range (≈5–10 m indoors), and fitting within a small enclosure or breadboard. Mechanical constraints are minimal (the system fits beside a plant pot), but safety is crucial: all electronics run at low voltage, and no high-voltage components are used. The pump is powered from a separate low-voltage supply to avoid driving high currents through the Arduino.

3.System requirements and constrains

3.1 Functional requirements

The system must monitor soil moisture in real time and switch the water pump on or off based on the moisture reading. It must also allow a user to manually trigger or stop irrigation via a Bluetooth-connected app. Specifically: reading the analog sensor input at least once per second, updating the pump state immediately upon threshold crossing, and sending/receiving simple commands over Bluetooth for “water on” or “water off.” The system shall indicate its status (e.g. by an LED or serial output) for debugging, and log moisture levels for tuning.

3.2 Constrains

3.2.1 Non-functional requirements

The wireless control should work reliably over ~5 m with low latency (Bluetooth classic, e.g. HC-05 module, typically has <100 ms lag). Power constraints include using only 5 V DC (Arduino and sensor from USB or battery, pump from 5 V supply) to eliminate high-voltage risks. The relay must switch at most 5 V and handle the pump’s current (~200 mA). The design’s cost and size are constrained by the requirement for hobbyist feasibility (we estimated total component cost ≈$25).

3.2.2 Mechanical constraints

Mechanical constraints are negligible since components are small (breadboard-sized) and the pump is external. Environmental constraints include preventing soil moisture from short-circuiting electronics (all wiring was kept above the soil level).

3.2.3 Safety

Safety constraints: maximum supply voltages are 5 V (logic) and ~5 V for the pump; we ensured a common ground between Arduino and pump circuit.

4. System design

The system works by taking input and calculating the moisture in the soil and when the value exceeds 440 it starts watering, you can also see the diagram in figure 6:

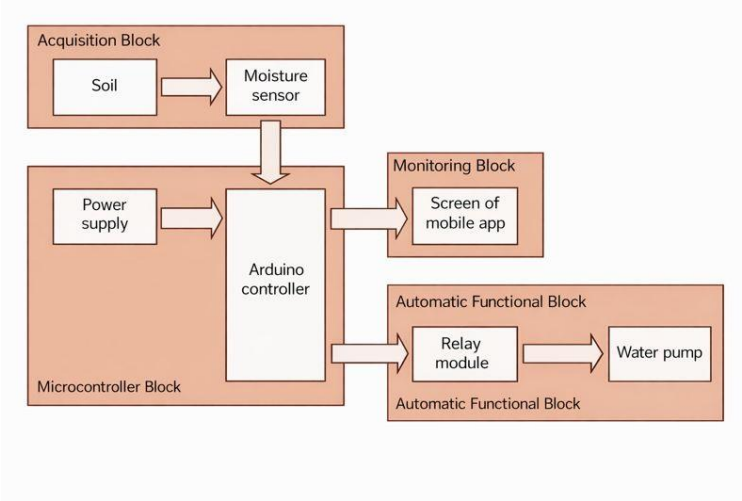


Figure 6. Automatic irrigation system diagram

4.1 Hardware design

Below you can see a table where the components are listed, in which you can also see their figure numbers are, so you can also see what they look like:

Component	**Purpose**	**Image Reference**
Arduino Uno R3	Central microcontroller and processing unit	**Figure 7**
Soil moisture sensor	Measures volumetric water content by electrical resistance or capacitance. Provides a voltage to A0 proportional to moisture.	**Figure 8**
HC-05 Bluetooth module	Wireless communication for mobile app	**Figure 9**
Single relay channel module	Acts as an electronic switch. Arduino digital output drives its input; the relay common/NO terminals switch the pump's supply.	**Figure 10**
Water pump (5V submersible)	Delivers water when powered. Requires ~200 mA. Placed in a reservoir or bucket, connected via a plastic hose to the plant.	**Figure 11**
Power supply (power bank)	Provides stable power for the motor and electronics	**Figure 12**

Table 1. Components we used

Figure 7. Arduino UNO R3 Microcontroller

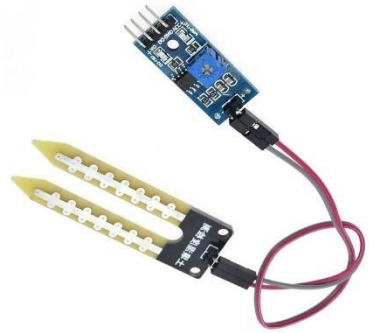


Figure 8. Soil moisture sensor

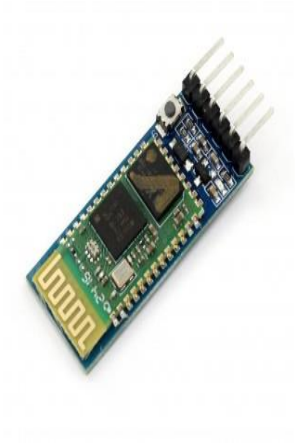


Figure 9. HC-05 Bluetooth Communication Module

Figure 10. Single relay channel module

Figure 11. Water pump (5V submersible)

Figure 12. Power supply (power bank that we used)

4.2 Construction materials

Materials: The physical model was developed after the structure was assembled, and it was made from LEGO pieces. The structure resembles a farm, with a hose extending from it that can water the soil, like a garden.

4.3 Software design

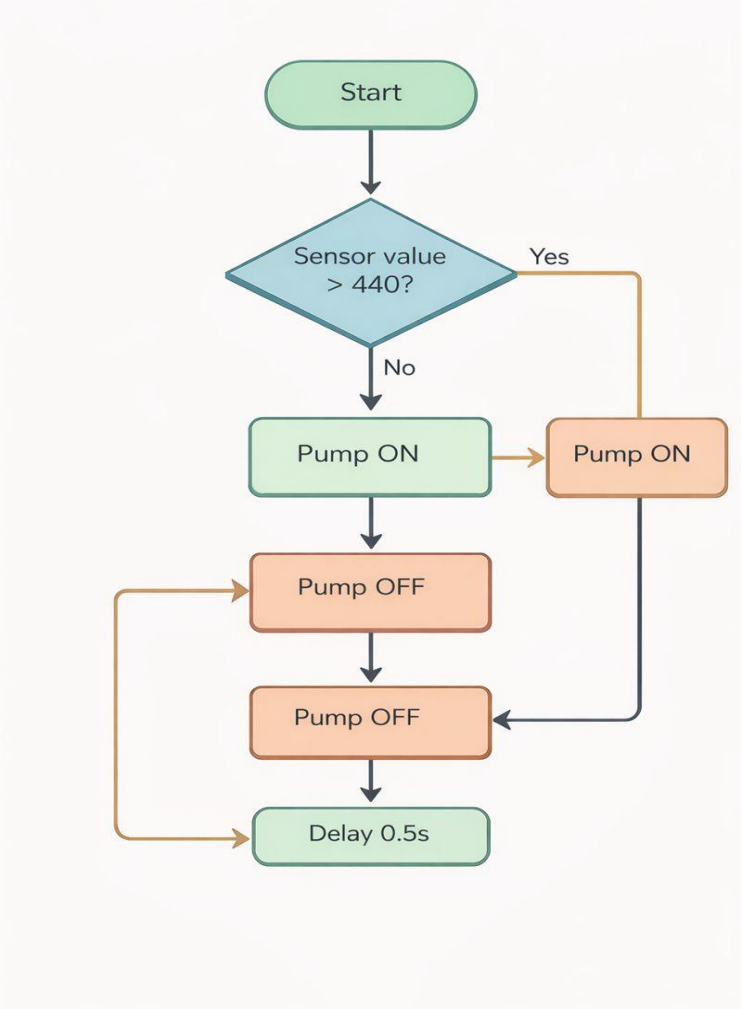


Figure 13. Flowchart for automating watering

The software logic of the Arduino automated watering system was implemented using Arduino C/C++ in the Arduino ide. The program followed a simple cyclic control structure, where sensor data was continuously read, evaluated, and used to control the water pump.

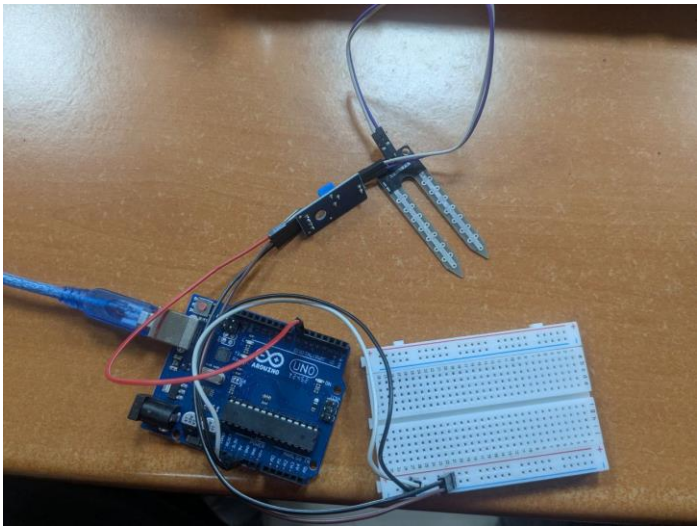
The program started with system initialization. After power was applied, the Arduino board entered the setup phase, where the serial communication was configured at 9600 baud for monitoring purposes. During this stage, the relay control pin was configured as an output, and the soil moisture sensor pin was prepared as an analog input. The pump was set to the off state by default to ensure safe system startup.

After initialization, the program entered the main loop, which repeated continuously while the system was powered. In each loop iteration, the Arduino read the soil moisture level using the analogRead function. The sensor returned a value between 0 and 1023, which represented the electrical conductivity of the soil and indirectly indicated its moisture level.

The measured sensor value was then printed to the serial monitor. This step allowed real time observation of soil conditions during testing and calibration. The serial output was used to verify correct sensor operation and to select an appropriate moisture threshold.
The decision block compared the measured sensor value with the predefined threshold value of 440. This comparison represented the core logic of the control system. When the sensor value was greater than 440, the soil was considered dry, and the system activated the watering process. When the sensor value was equal to or lower than 440, the soil was considered sufficiently moist, and watering was stopped.
Based on the decision result, the Arduino controlled the relay module. If the soil was dry, the relay control pin was set to HIGH, which activated the relay and powered the water pump. If the soil was not dry, the relay control pin was set to LOW, which disconnected power from the pump and stopped water flow.
After controlling the pump, the program introduced a fixed delay of 0.5 seconds. This delay stabilized system behavior, reduced unnecessary switching of the relay, and limited excessive sensor readings. After the delay, the program looped back to the moisture reading step and repeated the entire process.
This flow ensured automatic regulation of soil moisture. The pump was activated only when dryness was detected and was turned off once sufficient moisture was reached. The continuous loop structure allowed the system to respond to changing soil conditions without user intervention.

5.Prototypes versions and progress

5.1 Prototype 0000

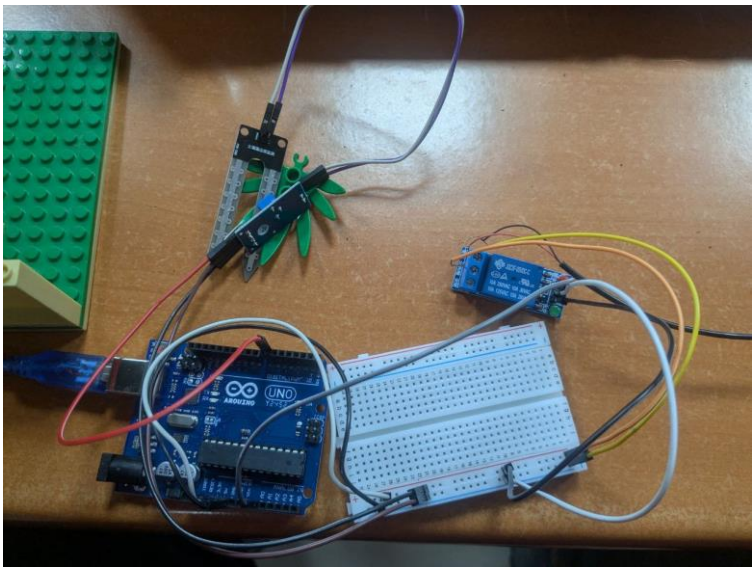


Figures 14. Prototype 0000

Purpose: The purpose of this version was to connect a soil moisture sensor to an Arduino and also test its operation with the Arduino.

Changes: In this prototype, an Arduino was connected to the sensor. Initial motion tests showed that the Arduino was working and the sensor was also being sensed.

5.2 Prototype 0001



Figures 15. Prototype 0001

Purpose: The purpose of this prototype was to connect a single relay channel to our Arduino system with a sensor in order to receive signals from the sensor.

Changes: The Arduino system expanded and the relay, through its built-in LED, showed green, which meant that it was able to detect contact with the sensor.

5.3 Prototype 0002

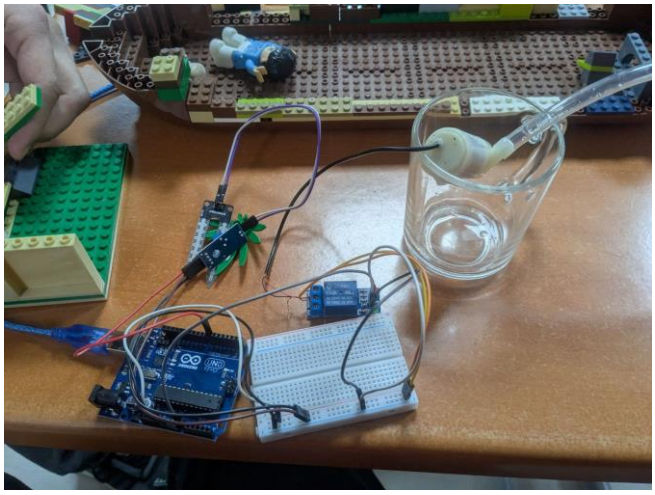


Figure 16. Prototype 0002

Purpose: The purpose of this prototype was to connect a pump to a relay, and also a mini hose to the pump, so that when a signal is given, it would draw water from a glass and water the plant.

Changes: In the end, everything worked, where the relay received a signal from the sensor, and thanks to it, it transmitted it to the pump and it started watering.

5.4 Prototype 0003

```
int soilPin = 7;
int relayPin = 6;

int soilState = 0;
```

Figure 17. Prototype 0003

This code represented where the hardware itself was located, it was the first place where the sensor was connected to D7(figure 17), but it did not work correctly, based only on the 1/0 principle, that is, dry or wet, which was not automated.

5.5 Prototype 0004 -

```
int soilPin = A0;
int relayPin = 6;

int threshold = 440;

bool autoMode = true;
```

Figures 18. Prototype 0004

In this part of the code, D7 was changed to A0(figure 18), thanks to this analogue it could automatically calculate the soil moisture and, based on its value, supply water or not.

5.6 Prototype 0005

```
if (soilValue > threshold) {
    digitalWrite(relayPin, LOW);
} else {
    digitalWrite(relayPin, HIGH);
}
```

Figure 19. Prototype 0005

In this prototype, we specified that water would be supplied when the value exceeded 500, as shown in figure 19. However, it worked completely the opposite way. The system only started watering when it was wet, and did not water when it was dry.

5.7 Prototype 0007

```
if (autoMode) {
  if (soilValue > threshold) {
    digitalWrite(relayPin, HIGH);
  } else {
    digitalWrite(relayPin, LOW);
  }
}
```

Figure 20. Prototype 0006

5.1 Prototype 0008

In this prototype in figure 20, we swapped HIGH and LOW, and it worked. The humidity value was also lowered to 440. This code has a bug fixed, and now it waters when the soil is dry and stops when it's wet.

```
bluetooth.print("SOIL:");
bluetooth.print(soilValue);

bluetooth.print(";STATUS:");
bluetooth.print(soilValue > threshold ? "DRY" : "WET");

bluetooth.print(";MODE:");
bluetooth.println(autoMode ? "AUTO" : "MANUAL");
Serial.println(soilValue);
```

Figure 21. Prototype 0007

In the prototype shown in Figure 21, system status information is now transmitted to a mobile app for monitoring. Sensor readings are also logged for debugging purposes, and a fixed latency has stabilized the system's operation.

5.1 Prototype 0009

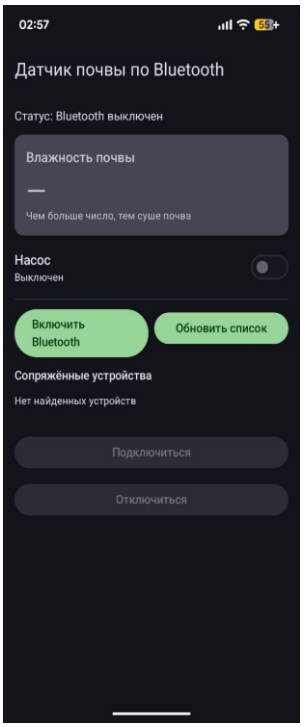


Figure 22. Prototype 0008

A prototype mobile app for wireless monitoring and control of an Arduino-based automated irrigation system was developed. The app communicated with the Arduino controller via Bluetooth and served as the user interface for the system.

The app's main screen displayed the current Bluetooth connection status. This indicator informed the user whether the system was connected or disconnected from the hardware module. A refresh button was added to scan for available Bluetooth devices and update the device list.

The app displayed the current soil moisture reading received from the Arduino. A brief explanation was provided that higher numerical values indicate drier soil conditions. This helped users correctly interpret the sensor data without technical knowledge.

Manual pump control was implemented using a toggle switch. When activated, the switch sent commands to turn the water pump on or off. This feature allowed the user to disable automatic mode when manual watering was required.

Buttons for enabling Bluetooth and controlling paired devices were included to simplify the connection process. The interface was designed with clear labels and a minimal number of elements to ensure ease of use and quick interaction.

This prototype demonstrated successful real-time communication between the mobile app and the Arduino system. It confirmed that soil data monitoring and pump control can be performed remotely using a simple and user-friendly interface. We also selected it as the primary mobile app for this project.

6. Implementation

```
#include <SoftwareSerial.h>

SoftwareSerial bluetooth(2, 3);
```

Figure. 23

Bluetooth communication was initialized to enable wireless data exchange with a mobile application.

```
int soilPin = A0;
int relayPin = 6;

int threshold = 440;

bool autoMode = true;
```

Figure. 24

Sensor input, relay output, moisture threshold, and control mode were defined for system operation.

```
void setup() {
  Serial.begin(9600);
  bluetooth.begin(9600);

  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);
}
```

Figure. 25

Serial communication was configured, the relay pin was set as output, and the pump was turned off at startup.

```
if (bluetooth.available()) {
  String command = bluetooth.readStringUntil('\n');
  command.trim();
```

Figure. 26

Incoming Bluetooth commands were read and prepared for processing.

```
if (command == "AUTO") {
  autoMode = true;
}
else if (command == "ON") {
  autoMode = false;
  digitalWrite(relayPin, HIGH);
}
else if (command == "OFF") {
  autoMode = false;
  digitalWrite(relayPin, LOW);
}
```

Figure. 25

Control mode was switched and the pump state was updated based on user commands.

```
int soilValue = analogRead(soilPin);

if (autoMode) {
  if (soilValue > threshold) {
    digitalWrite(relayPin, HIGH);
  } else {
    digitalWrite(relayPin, LOW);
  }
}
```

Figure. 26

Soil moisture data was collected from the sensor as an analog value.

Automatic watering was activated or stopped based on sensor readings and threshold comparison.

```
bluetooth.print("SOIL:");
bluetooth.print(soilValue);

bluetooth.print(";STATUS:");
bluetooth.print(soilValue > threshold ? "DRY" : "WET");

bluetooth.print(";MODE:");
bluetooth.println(autoMode ? "AUTO" : "MANUAL");
```

Figure. 27

System status information was sent to the mobile application for monitoring.

```
Serial.println(soilValue);

delay(500);
```

Figure. 28

Sensor values were logged for debugging, and a fixed delay stabilized system behavior.

7. Discussion

7.1 Work that went well

The components of our system performed reliably and exceeded our expectations. Firstly, the watering system worked on the first try, and it actually watered. Secondly, for this project, we used a relatively old pot of soil and left it there, and to our surprise, greenery began to sprout.

Step-by-step unit testing significantly simplified development. Work began with testing control of a single water pump, followed by adding a humidity sensor, timer, and other components to the system. This allowed us to isolate potential issues and avoid complex debugging of the finished device. Consolidating the ground lines of the Arduino, sensors, and power module improved the stability of readings and prevented malfunctions. After switching the system to fully autonomous battery power, it operated reliably and maintained the set watering cycle without intervention.

7.2 Problems we faces

Fortunately, there were very few problems. The first was that the initial design watered when the soil was wet and did not water when it was dry. This was easily resolved (see the prototypes). Another problem was that it mistook air for soil and started watering, but the solution was the same as the first problem.

7.3 Final project

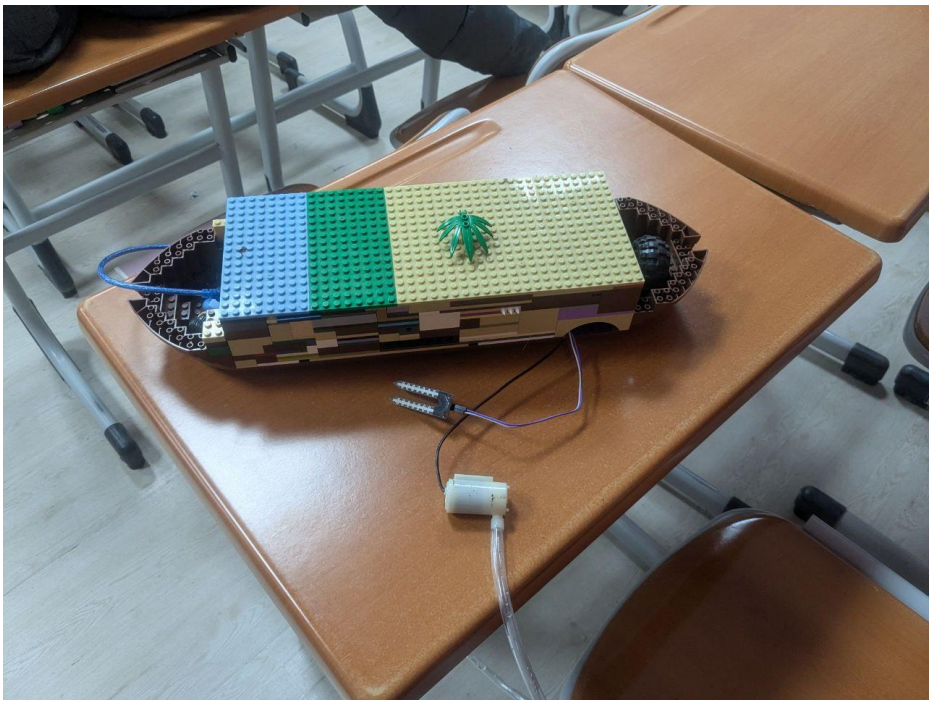


Figure 29. Final project

After all the prototyping, studying and combining our system with our LEGO design, the project was completed and continued to function correctly within the design.

8. Conclusion and future works

We have built and tested an Arduino automated watering system that successfully waters a plant based on soil moisture. The final system detects dry soil, activates a pump via a relay, and then stops when moisture is restored. It also supports manual on/off control through a Bluetooth-connected smartphone. This design simplifies plant care and conserves water, aligning with goals of sustainable home gardening and automation
In summary, the project solved the core problem: timely, automated irrigation.
For future work, we suggest several enhancements:

1) Add sensors: Integrate additional sensors (e.g. soil temperature, humidity) and use more advanced moisture sensors (capacitive) to improve reliability.

2) Water reservoir monitoring: Include a float switch or optical level sensor to prevent dry-run of the pump.

3) Networking: Upgrade to an IoT-capable microcontroller (e.g. ESP32) to enable Wi-Fi/cloud control and monitoring (data logging on platforms like ThingSpeak).

4) Multiple zones: Extend the design to multiple plants by adding more relays and valves, controlled by the same or an expanded microcontroller.

5) Power optimization: Explore low-power modes or solar power for outdoor use

6) User interface: Develop a dedicated mobile app or webpage to set thresholds, view sensor data, and schedule watering (instead of just raw commands).

Overall, the project demonstrates that even a simple Arduino setup can implement a functional smart irrigation system. With the improvements above, it could evolve into a more robust product.

9. References

[1] Devika, S. V., Khamuruddeen, S., Khamurunnisa, S., Thota, J. and Shaik, K. (2014) *Arduino based automatic plant watering system*. International Journal of Advanced Research in Computer Science and Software Engineering, 4(10), pp. 449-456.

[2] Bansod, S., Jaiswal, R., Sargam, P. and Sawant, S. (2020) *Arduino based water irrigation system*. Proceedings of the International Conference on Emerging Trends in Engineering and Technology. IEEE.

[3] Banzi, M. and Shiloh, M. (2014) *Getting started with Arduino*. 3rd edn. Sebastopol: O'Reilly Media.

[4] Monk, S. (2016) *Programming Arduino: getting started with sketches*. 2nd edn. New York: McGraw-Hill Education.

[5] Kim, Y., Evans, R. G. and Iversen, W. M. (2008) 'Remote sensing and control of an irrigation system using a distributed wireless sensor network', *IEEE Transactions on Instrumentation and Measurement*, 57(7), pp. 1379-1387.

[6] Gutiérrez, J., Villa-Medina, J. F., Nieto-Garibay, A. and Porta-Gándara, M. A. (2014) 'Automated irrigation system using a wireless sensor network and gprs module', *IEEE Transactions on Instrumentation and Measurement*, 63(1), pp. 166-176.

[7] Jones, H. G. (2004) 'Irrigation scheduling: advantages and pitfalls of plant-based methods', *Journal of Experimental Botany*, 55(407), pp. 2427-2436.

10.Appendices

The following resources provide direct access to the project's digital materials, including the full source code and a demonstration video.

Resource	Description	Value / Image Reference
GitHub repository	Full Arduino C/C++ source code, mobile app and circuit diagram.	https://github.com/ANITS60106/Arduino-automated-watering-system
YouTube video	Demonstration of the final prototype in operation.	https://www.youtube.com/watch?v=1ssp8XlErzk

Table 2. Appendices