

# Smart Irrigation System with Sensor Fault Detection using Scilab

**Sanyam Bhavsar**

Int. M.Tech Student in Data Science and Computation VIT Bhopal University

IoT and Data Analysis / Agriculture Automation / Sensor Fault Detection

23-05-2026

## Abstract

Water management is a very important part of modern agriculture, and smart irrigation systems can help use water more efficiently. This project focuses on developing a Scilab-based smart irrigation system that uses sensor readings such as soil moisture, temperature, and humidity to decide whether irrigation should be turned ON or OFF. The main aim is to simulate how automated irrigation can improve decision-making compared to manual monitoring.

Along with irrigation control, this project also includes sensor fault detection. In real systems, sensors may sometimes give incorrect readings because of malfunction, noise, or abnormal changes in values. Such faulty readings can lead to wrong irrigation decisions. To solve this, fault detection methods such as checking out-of-range values, sudden jumps, and inconsistent readings are used in the simulation.

In the improved version of this case study, redundant sensors are added for soil moisture and temperature. Two sensors are simulated for the same physical quantity, and the difference between their readings is used as a residual. If this residual crosses a threshold, the system detects a possible sensor fault. Faulty readings are corrected before they are used for irrigation control. The complete system is modeled in Scilab using simulated sensor data over time, and the results are shown using graphs of corrected soil moisture, residuals, temperature, humidity, and irrigation status.

## 1. Introduction

Water management has become an increasingly important issue in agriculture, especially in situations where water resources are limited and efficient irrigation is necessary for healthy crop growth. In many traditional irrigation systems, watering is done either manually or according to fixed time intervals. Although this method is simple, it does not always reflect the actual condition of the soil. As a result, water may be supplied even when it is not needed, or the soil may remain dry when irrigation is required.

Smart irrigation systems offer a more effective solution by using sensor data to monitor field conditions. Soil moisture is one of the most important parameters in such systems because it gives a direct indication of whether the soil has enough water. Environmental parameters such as temperature and humidity can also provide useful information about surrounding conditions.

However, the reliability of such systems depends on the accuracy of the sensors. In practical situations, sensor readings may become faulty because of noise, temporary malfunction, calibration error, or unrealistic changes in values. If such readings are used directly, the irrigation decision may become incorrect. Therefore, fault detection becomes an important part of a reliable smart irrigation system.

In the earlier version of this project, range checking and sudden-jump checking were used to detect abnormal sensor values. In the improved version, redundant sensors are also added for soil moisture and temperature. This means that two sensors are used to measure the same quantity. If both sensors are healthy, their readings should remain close. If the difference between them becomes large, the system can identify that a fault may be present.

This project is developed in Scilab using simulated sensor data. The system checks the validity of sensor readings, identifies faulty values, corrects them using simple logic, and then uses the corrected data to make irrigation decisions. This makes the project both practical and technically meaningful.

## 2. Problem Statement

The main goal of irrigation is to provide water to the soil in the right amount and at the right time. In conventional systems, irrigation decisions are often not based on actual field conditions, which can lead to inefficient water usage. A smart irrigation system can improve this process by using sensor readings to monitor soil and environmental conditions continuously.

The main challenge, however, is that sensor data cannot always be assumed to be correct. A moisture sensor may show a negative value, a temperature sensor may suddenly show an unrealistically high reading, or a sensor may produce an abrupt jump that does not represent real conditions. If the system uses such faulty readings directly, it may turn irrigation ON or OFF incorrectly.

This case study addresses both of these issues together. First, it models a smart irrigation system using soil moisture, temperature, and humidity data. Second, it introduces a fault detection mechanism to identify abnormal sensor readings. The improved implementation also includes redundant soil moisture and temperature sensors. These redundant sensors are used to calculate residuals, which help detect inconsistent readings between sensors measuring the same quantity.

Once faulty values are detected, they are corrected before being used in irrigation control. The entire system is implemented in Scilab so that the behavior can be analyzed using graphs.

The main objectives of the project are:

1. To simulate soil moisture, temperature, and humidity data in Scilab.
2. To create redundant sensor readings for soil moisture and temperature.
3. To inject artificial faults into sensor data.
4. To detect faults using range check, sudden-jump check, and residual-based check.
5. To correct faulty values before irrigation decision-making.
6. To decide irrigation ON/OFF using corrected soil moisture.
7. To show the behavior of the system using Scilab graphs.

### 3. Basic concepts related to the topic

#### **a) Smart Irrigation**

A smart irrigation system is an automated system that decides when irrigation is needed by using input from sensors. Unlike manual irrigation, it does not depend only

on human judgment or fixed schedules. Instead, it responds to sensor data and helps improve water efficiency.

In this project, the irrigation decision is based mainly on corrected soil moisture. If the corrected soil moisture is less than the selected threshold, irrigation is turned ON. If the corrected soil moisture is greater than or equal to the threshold, irrigation is kept OFF.

The basic logic is:

**If corrected soil moisture < threshold, irrigation = ON**

**If corrected soil moisture  $\geq$  threshold, irrigation = OFF**

In this project, the moisture threshold is taken as 40%.

### **b) Soil Moisture**

Soil moisture represents the amount of water available in the soil. It is the most important input in this project because the irrigation decision is mainly based on this value.

In the simulation, soil moisture is represented in percentage form. A valid soil moisture value should be between 0% and 100%. Any value below 0 or above 100 is considered invalid and is marked as a fault.

### **c) Temperature and Humidity**

Temperature and humidity are additional environmental parameters included in this project. Although they are not directly used to turn irrigation ON or OFF in the current model, they are part of the sensor set and make the simulation more realistic.

Temperature and humidity are also checked for abnormal values. This helps demonstrate how multiple sensor readings can be monitored and validated in a smart system.

The valid ranges used in the project are:

**Soil moisture: 0% to 100%**

**Temperature: 5°C to 50°C**

**Humidity: 10% to 100%**

### **d) Sensor Fault Detection**

In any sensor-based system, it is important to verify whether the readings are reasonable. Sensor fault detection is the process of identifying abnormal or incorrect values before they affect system operation.

In this project, three checks are used.

### **1. Range Check**

A sensor value is marked faulty if it lies outside its valid range.

Examples:

**Soil moisture = -12%**

**Soil moisture = 135%**

**Temperature = 82°C**

**Humidity = -8%**

These values are not realistic, so they are considered faulty.

### **2. Sudden-Jump Check**

A reading is also marked faulty if it changes too sharply compared to the previous reading. This helps detect unrealistic spikes.

The jump is calculated as:

**Jump = |current value - previous value|**

If the jump is greater than the allowed limit, the value is marked as faulty.

### **3. Residual-Based Check**

This is the new feature added in the improved code.

Residual-based checking uses redundant sensors. Two sensors are used to measure the same quantity. If both sensors are working properly, their readings should be close. If the difference between the two readings becomes too large, one of the sensors may be faulty.

For soil moisture:

**Moisture residual = Moisture Sensor 1 - Moisture Sensor 2**

For temperature:

**Temperature residual = Temperature Sensor 1 - Temperature Sensor 2**

If the residual crosses the threshold, a fault is detected.

### **e) Redundant Sensors**

Redundant sensors are used to improve reliability. In this project, two soil moisture sensors and two temperature sensors are simulated.

For example:

**Moisture Sensor 1 = 44%**

**Moisture Sensor 2 = 45%**

The values are close, so the sensors are considered normal.

But if:

**Moisture Sensor 1 = 44%**

**Moisture Sensor 2 = 90%**

then one of the sensors may be faulty.

This is useful because the system does not depend on only one sensor reading.

Instead, it compares two readings and detects inconsistency.

#### **f) Fault Correction**

Once a faulty reading is detected, it should not be used directly. In this project, faulty values are corrected before the irrigation decision.

The correction method is:

<b>Condition</b>	<b>Correction Method</b>
Both redundant sensors are healthy	Use average of both sensors
Sensor 1 is faulty	Use Sensor 2 value
Sensor 2 is faulty	Use Sensor 1 value
Both sensors are faulty	Use previous corrected value
Humidity sensor is faulty	Use previous corrected humidity value

#### **g) Threshold-Based Decision Making**

Threshold-based logic is one of the simplest forms of control used in automation systems. In this project, the corrected soil moisture value is compared with a threshold to determine irrigation status.

This makes the decision process transparent, easy to understand, and suitable for implementation in Scilab.

#### **h) Time-Series Simulation**

The sensor readings in this project are generated over multiple time steps. Since the values change with time, the outputs can be treated as time-series data. This allows the behavior of the system to be visualized clearly using plots such as moisture vs time, residual vs time, and irrigation status vs time.

#### 4. What is Adopted, Modified, and Not Implemented

##### 4.1 What is Adopted from the Research Papers

Concept from Research Papers	Implemented in This Project
Soil-moisture-based irrigation control	Yes. Irrigation depends on corrected soil moisture.
Pump ON/OFF decision	Yes. Pump action is represented by irrigation status 1 or 0.
Monitoring soil and environmental parameters	Yes. Soil moisture, temperature, and humidity are simulated.
Sensor fault detection	Yes. Faults are detected before irrigation decision.
Direct redundancy	Yes. Two moisture sensors and two temperature sensors are simulated.
Residual generation	Yes. Residual is calculated as Sensor 1 - Sensor 2.
Threshold-based fault detection	Yes. Fault is detected when residual crosses threshold.

## 4.2 What is Modified

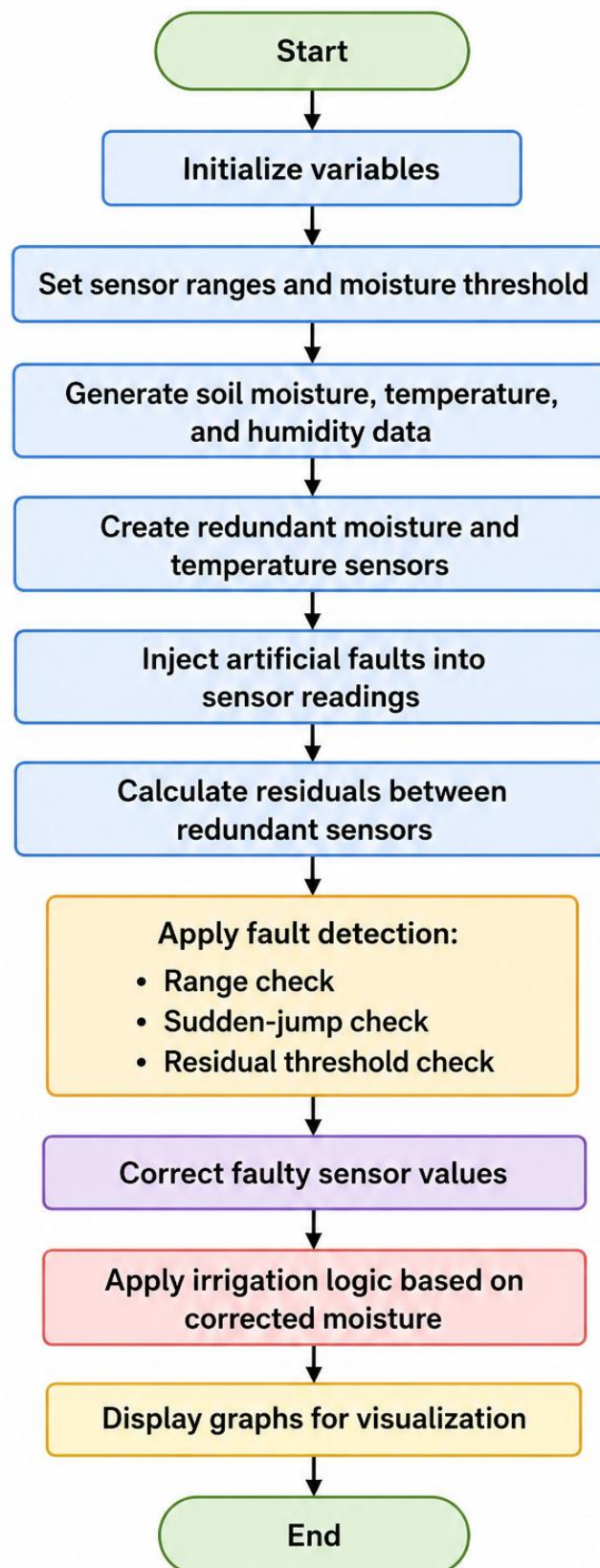
Research Paper/System Concept	Modification in This Project
Hardware irrigation system	Converted into Scilab simulation.
Real sensor data	Replaced with simulated sensor readings.
Arduino, relay, and pump	Represented using Scilab logic and irrigation status.
Complete parity-space method	Simplified into residual-based comparison.

## 4.3 What is Not Implemented

Not Implemented	Reason
Physical hardware	Project is simulation-based.
Real sensors and pump	Only simulated values are used.
GSM/IoT/mobile app	Outside project scope.
Real wireless sensor network	Replaced by simulated sensor arrays.
Full parity-space FDI	Only simplified residual method is implemented.



## 5. Flowchart



## 6. Software/Hardware used

**Operating System:** Windows 11

**Software:** Scilab 2026.0.1

**Toolbox Used:** No external toolbox used

**Hardware:** Personal computer/laptop with standard processing capability

No physical irrigation hardware is used in this case study. The Arduino, sensors, pump, and wireless sensor nodes are represented only through Scilab simulation logic.

## 7. Procedure of execution

- I. Open Scilab and clear the command window, workspace, and graphics windows.
- II. Define the total number of simulation steps and the threshold values used for irrigation and fault detection.
- III. Generate simulated values of soil moisture, temperature, and humidity over time.
- IV. Create redundant sensor readings for soil moisture and temperature.
- V. Insert abnormal values manually to simulate faulty sensor behavior.
- VI. Store the original faulty sensor readings for comparison.
- VII. Calculate residuals between redundant moisture and temperature sensors.
- VIII. Apply range-based checks to identify values that lie outside valid limits.
- IX. Apply sudden-change detection to identify unrealistic jumps in the sensor signals.
- X. Apply residual-based detection to identify inconsistent readings between redundant sensors.
- XI. Mark abnormal values as faults.
- XII. Correct faulty values using healthy sensor readings or previous corrected values.
- XIII. Compare the corrected soil moisture value with the irrigation threshold.
- XIV. Set irrigation status to ON if the soil is too dry, otherwise keep it OFF.
- XV. Plot graphs of corrected soil moisture, residuals, temperature, humidity, and irrigation status.
- XVI. Observe the graphs and printed summary to interpret the system performance.

## 8. Result

The simulation generated sensor readings for soil moisture, temperature, and humidity over a fixed time period. To test the fault detection logic, some artificial faults were added to the sensor values. These included invalid moisture readings, values above the valid range, unrealistic temperature values, abnormal humidity readings, and sudden changes in sensor data.

The system was able to detect these faulty readings using three methods: range checking, sudden-jump checking, and residual-based checking between redundant sensors. After a fault was detected, the faulty value was corrected before it was used for irrigation control. Because of this, the irrigation decision was not affected by incorrect sensor readings.

The soil moisture graph shows how the moisture level changes with time. The irrigation status graph shows when the system turns irrigation ON and OFF. Irrigation is turned ON when the corrected soil moisture falls below the threshold, and it is turned OFF when the moisture level becomes sufficient.

The residual graphs show the difference between the redundant sensors. During normal operation, this difference remains within the threshold. When a fault is introduced, the residual increases and crosses the threshold, which helps in identifying sensor inconsistency.

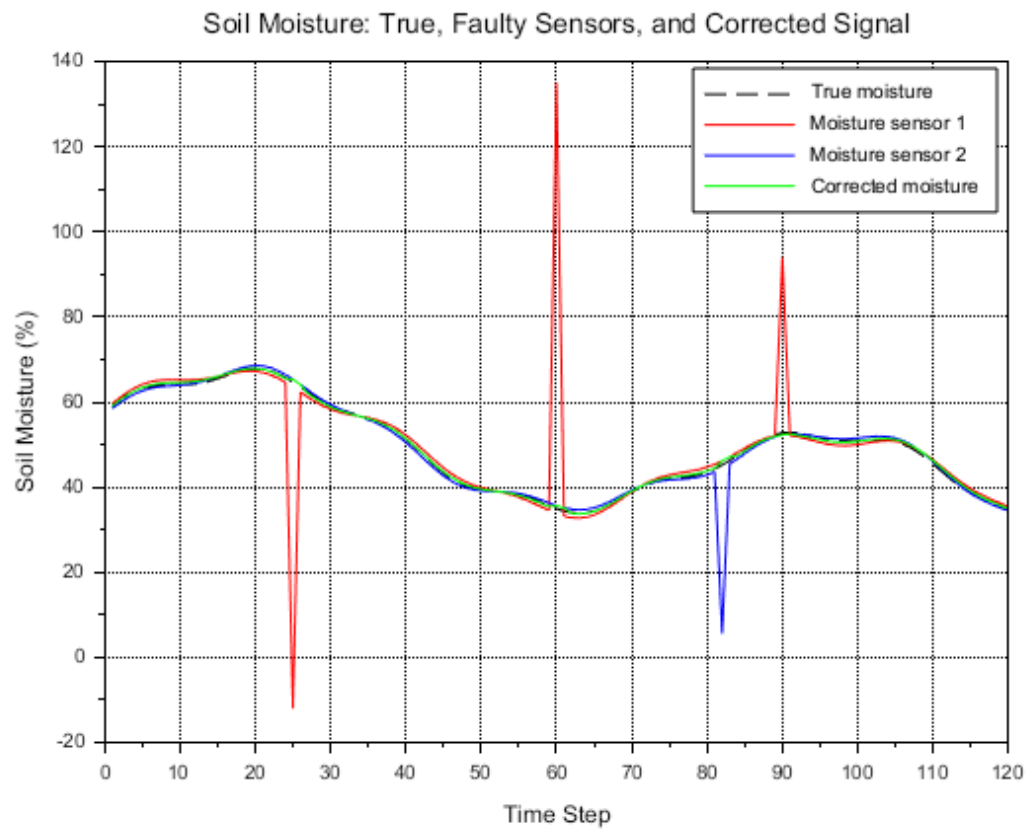
Overall, the results show that the project successfully simulates a smart irrigation system with improved sensor fault detection. The system is able to handle faulty readings and make more reliable irrigation decisions using corrected sensor values.

.

### 8.1 Soil Moisture Result

The soil moisture graph shows true soil moisture, two simulated moisture sensor readings, and corrected soil moisture. Faults such as out-of-range values and sudden jumps were inserted into the moisture readings.

The corrected soil moisture curve is more reliable because faulty readings are corrected before irrigation control. This shows that the system does not directly depend on raw faulty data.



**Figure 1:** Soil moisture readings and corrected soil moisture versus time.

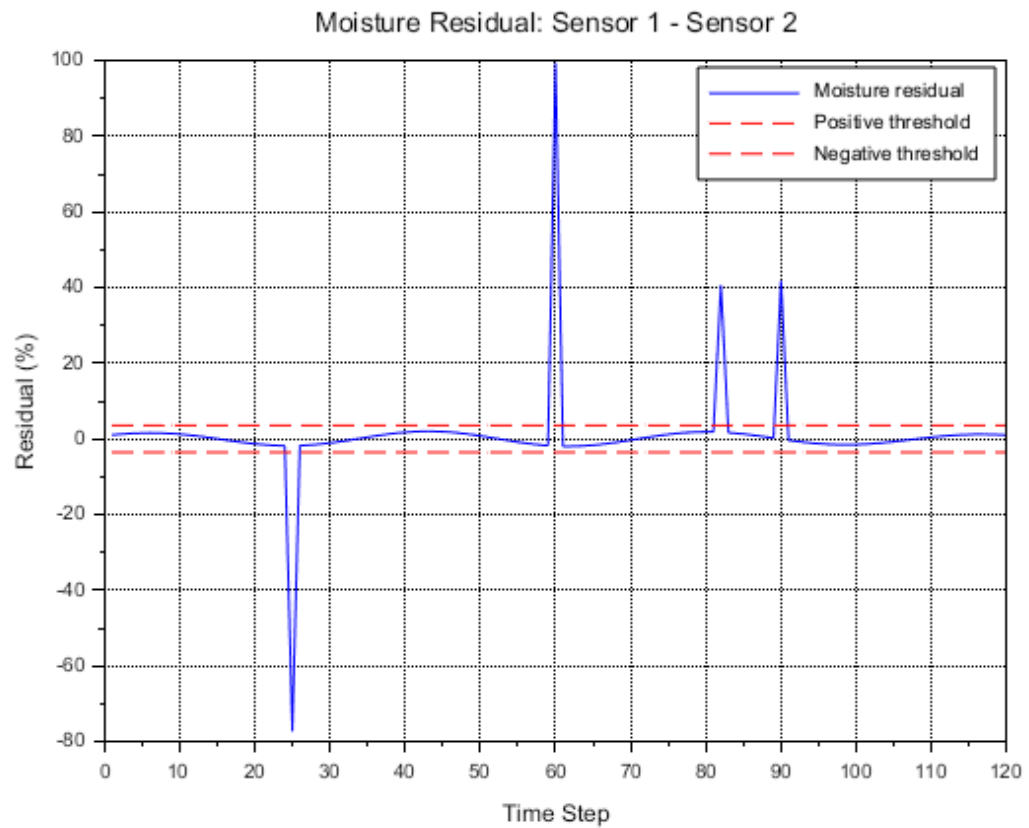
## 8.2 Moisture Residual Result

The moisture residual graph shows the difference between moisture sensor 1 and moisture sensor 2.

**Moisture residual = Moisture Sensor 1 - Moisture Sensor 2**

When both sensors are working normally, the residual remains within the threshold.

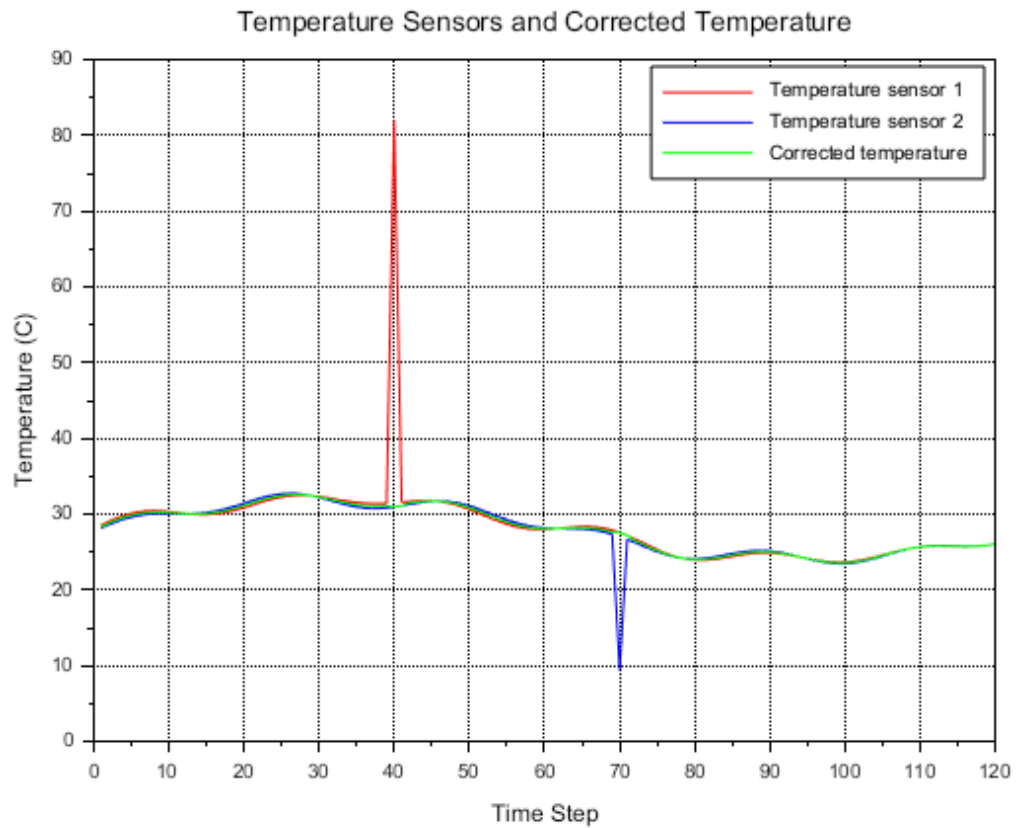
When a moisture sensor fault occurs, the residual crosses the threshold.



**Figure 2:** Moisture residual versus time.

### 8.3 Temperature Result

The temperature graph shows the readings of two temperature sensors and the corrected temperature value. If one sensor gives an abnormal reading, the system detects it and uses the healthier sensor value or previous corrected value.



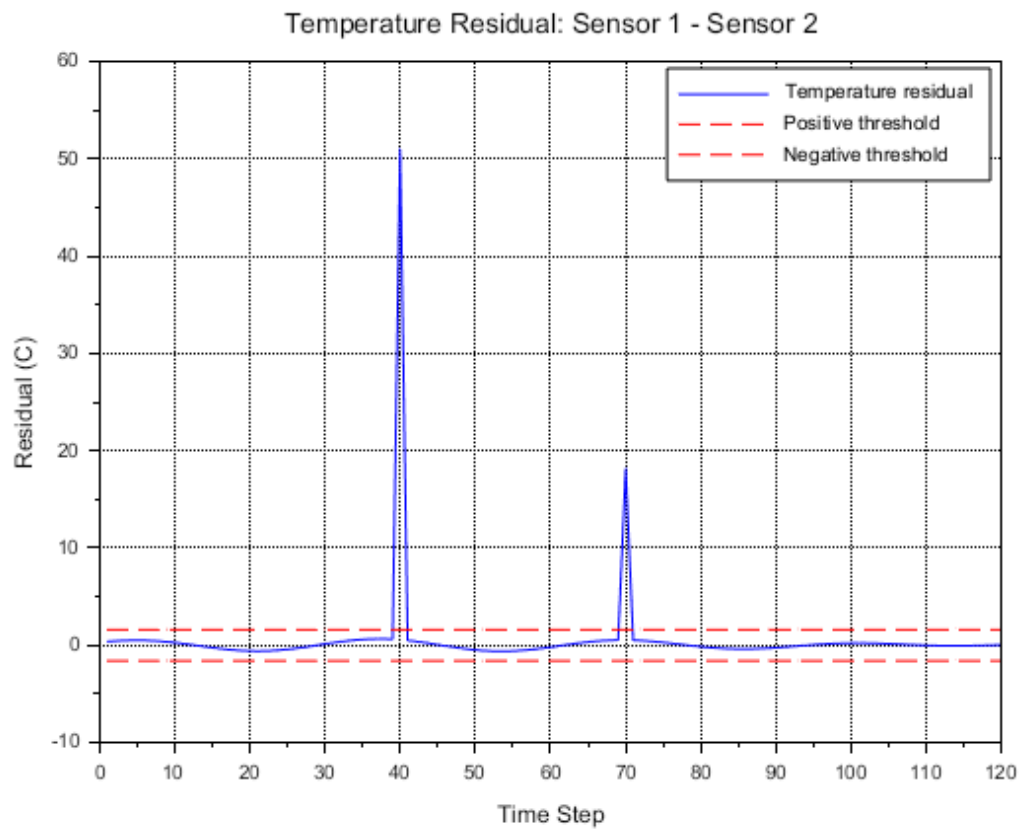
**Figure 3:** Temperature sensor readings and corrected temperature versus time.

#### 8.4 Temperature Residual Result

The temperature residual graph shows the difference between two temperature sensors.

Temperature residual = Temperature Sensor 1 - Temperature Sensor 2

A large residual indicates that one of the temperature sensors may be faulty.

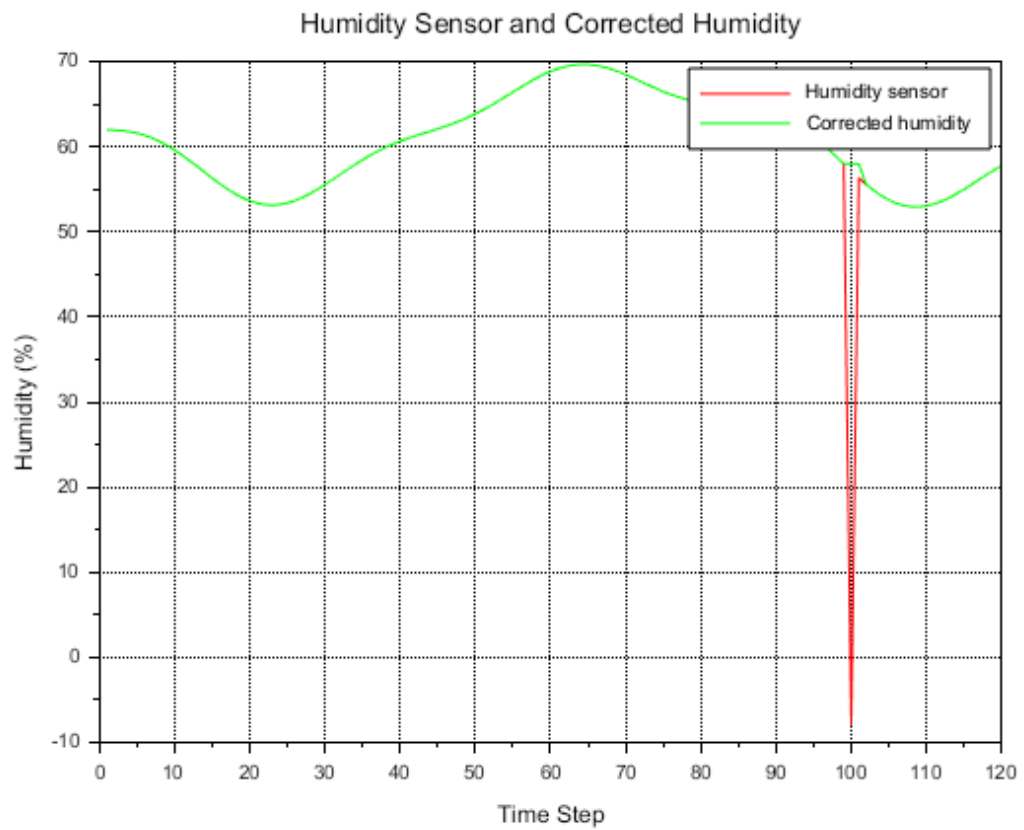


**Figure 4:** Temperature residual versus time.

### 8.5 Humidity Result

Humidity is simulated using one sensor. Since there is no redundant humidity sensor in this project, residual checking is not applied to humidity. Instead, range checking and sudden-jump checking are used.

If humidity is faulty, the previous corrected humidity value is used.



**Figure 5:** Humidity reading and corrected humidity versus time.



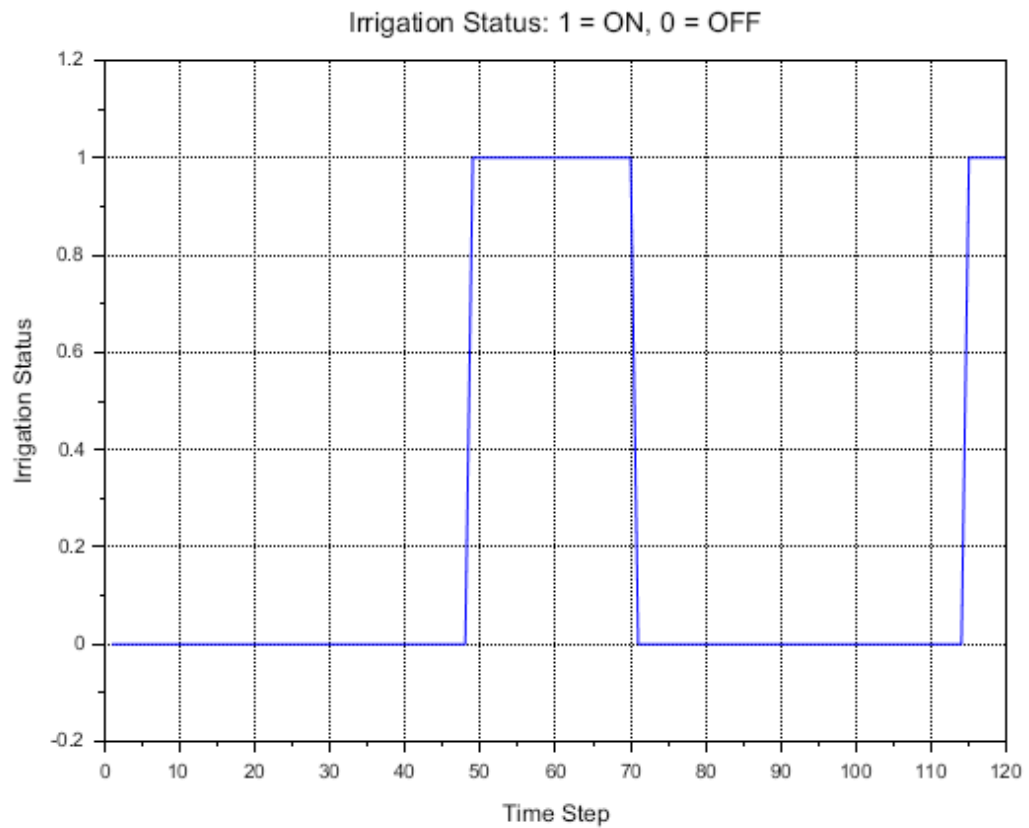
## 8.6 Irrigation Status Result

The irrigation status graph shows when the irrigation system is ON or OFF.

**1 = Irrigation ON**

**0 = Irrigation OFF**

The result confirms that irrigation is turned ON when corrected soil moisture falls below the threshold and turned OFF when the soil moisture is sufficient.



**Figure 6:** Irrigation status versus time.

## 8.7 Result Comparison with Reference Papers

Parameter	Expected from Research Paper	Observed in Scilab Simulation
Irrigation control	Pump turns ON when soil moisture is low and OFF when moisture is sufficient.	Irrigation status becomes 1 when corrected moisture is below 40% and 0 otherwise.
Sensor monitoring	Soil moisture, temperature, and humidity are monitored.	These parameters are simulated using Scilab-generated data.
Redundant sensors	Two sensors can measure the same quantity for fault detection.	Two moisture sensors and two temperature sensors are used.
Residual-based detection	Fault is detected when residual deviates from normal value.	Fault is detected when Sensor 1 - Sensor 2 crosses the threshold.
Implementation type	Research papers use hardware/laboratory setup.	This project uses software simulation only.

## References

Automatic Irrigation System using Soil Moisture Sensor ResearchGate:

[https://www.researchgate.net/publication/362701179\\_Automatic\\_Irrigation\\_System\\_using\\_Soil\\_Moisture\\_Sensor](https://www.researchgate.net/publication/362701179_Automatic_Irrigation_System_using_Soil_Moisture_Sensor)

Sensor fault detection and isolation for smart irrigation using parity space approach Pdf:

<https://pdfs.semanticscholar.org/240a/bf4b9733b62074dd2a54bcbcb89190f7a179.pdf>