An Integrated Simulink and IoT Framework for Data Acquisition and Control using Custom Driver Blocks

H. E. Abbassi[1*], K. Lammari[2], F. Bounaama[3], S. Bennaceur[4]

[1]Smart Grids and Renewable Energies Laboratory, Department of Electrical Engineering, Faculty of Technology, University of Tahri Mohamed Bechar, Algeria.

[2]Energetic in Arid Zones Laboratory, Faculty of Technology, University of Tahri Mohamed, Department of Material Science, Bechar, Algeria.

[3]Energetic in Arid Zones Laboratory, Department of Electrical Engineering, Faculty of Technology, University of Tahri Mohamed Bechar, Algeria.

[4]Development of Renewable Energies and their Applications in Sahara AreasZones Laboratory, Faculty of Technology, University of Tahri Mohamed, Department of Material Science, Bechar, Algeria.

ABSTRACT: This study deals with the design and implementation of a fully modularized, embedded system equipped with Internet of Things (IoT) technology, which was developed using the Simulink integrated control. A key contribution of the work is the utilization of only the Simulink environment to devleop object-oriented models without the use of the traditional script-based programming methods. It describes a combination of extensive integration of custom driver blocks through supplementary libraries and S-function algorithm development combined with sensor reading calibration and filtering that further extends the flexibility of the workflow. The performance of the system was validated by creating a real-time data acquisition, on/off control logic, and cloud-based data visualization through the ThingSpeak platform, demonstrated by a case study of an irrigation system in an arid area. The system is versatile, as it is modular and can be adapted to embedded systems in arid, resource-constrained regions, making it suitable for applications in intelligent control and automation in many other applications beyond irrigation.

KEYWORDS: Simulink environment, Embedded Systems, S-Function, Driver blocks, Data acquisition, IoT.

[Received Sep. 19, 2025; Revised Nov 15, 2025; Accepted Dec 15, 2025] Print ISSN: 0189-9546 | Online ISSN: 2437-2110

I. INTRODUCTION

The development of embedded systems has had a significant impact on the fields of automation, data acquisition, monitoring, agriculture, and smart infrastructure. These systems play a crucial role in providing flexible and intelligent control in the industrial and field-deployment environment. As the functional complexity of embedded systems increases, there is a growing demand in development methodologies, which will permit rapid, stable prototyping and stable real-time execution. On the other hand, Model-Based Design (MBD), has emerged as a revolutionary paradigm, providing a single environment of system-level modeling and simulation. (Kshirsagar et al.,2022; Jackson et al.,2006; Nicola et al., 2021; United Electronic Industries, 2025; MathWorks, 2025a). MATLAB/ Simulink exemplifies the MBD paradigm and, unlike other scripting programming languages, enables engineers and researchers to model, simulate, and deploy complex embedded control systems visually. Its graphical tool makes development easier as it connects simulation with real-time implementation, thereby improving traceability and accelerating the development cycle. Its features, facilitated by S-Functions, can be coupled with custom sensors and proprietary hardware, making it involved in a wide variety of complex internet of things (IoT) applications (MathWorks, 2025b; Caseiro et al., 2022). As the trend shifts to model-based design in IoT applications, an increasing number of them are being modeled in Simulink. It supports turnkey integration with hardware for acquisition, control, and deployment to edge and cloud platforms. Its interoperability with IoT cloud platforms, such as ThingSpeak, for real-time analysis and monitoring, contributes to the effective development of connected systems for industrial and research-based applications (MathWorks, 2025c).

The majority of research that has been done on embedded automation systems programming uses scripting languages like C/C++ or Python, where system behaviour is described procedurally, and hardware interactions are written by hand. However, compared to this, our work deals with a more complicated issue of bringing an object-oriented modelling method into the context of the Simulink platform to develop and deploy executable embedded models. During the extensive literature review, these are typically classified into three categories; studies based on pure simulation and no experimental validation; studies that simulate models based on experimental data obtained externally, and a smaller

*Corresponding author: abbassi.hayat@univ-bechar.dz

doi: http://doi.org/10.63746/njtd.v22i5.4034

number of studies that seek to implement everything in Simulink, data acquisition, and deployment.

To highlight how our approach is consistent with the recent Simulink-based development, we begin by reviewing some recent similar works within the agricultural field since our case study belongs to this domain. Al-Omary et al., (2018) presents a cloud-based IoT system of a garden watering system implemented on the Arduino Uno framework with the prototype designed and simulated with MATLAB/Simulink. Their most important contribution is the design of a cloud-based architecture where the ThingSpeak platform interprets sensor data (soil moisture and light) and sends irrigation/light on/off instructions based on cloud-based analysis against preset thresholds. Consequently, the work illustrates irrigation control with the assistance of IoT, but lacks a complete Simulink-based control tasks, advanced signal processing, and custom low-level drivers.

Belkadi et al., (2020) developed sophisticated fuzzy logic control (FLC) system to control the temperature and humidity in the greenhouse with a comprehensive Simulink model of the energy water balance. This control logic was entirely implemented in Simulink for simulation only and then deployed to a Raspberry Pi 3 (using Python programming languages), through which real-time actuation of heating, ventilation, humidification, and dehumidification equipment can be realized. Even though the work incorporates IoT, which was aimed at monitoring, it was more of a high-level FLC design, as opposed to low-level driver modelling, custom Simulink blocks, or embedded hardware-specific implementations. Another work presented by (Laktionov et al., 2019) is a computerized greenhouse microclimate monitoring system based on Arduino (ATMega 2560 with W5100 Ethernet, Simulink Support Package for Arduino, and cloud-based logging on ThingSpeak to emphasize the acquisition and data processing of multiple sensors. The Simulink model manage signal conversion, averaging and threshold -based triggering for actuators such as irrigation, ventilation, lighting and heating. The work was focused on data aggregation and monitoring rather than custom driver creation, embedded algorithmic control, and the entire deployment of Simulink to hardware over standard support-package blocks. Mashhadany et al., (2024) proposed a cloud-based IoT agricultural monitoring system that combine sensors, smart cameras, and intelligent controllers to control the irrigation, detect pests, and environmental factors remotely. The system uses a hybrid fuzzy-PID controller for optimizing water pump operation implemented in MATLAB Simulink 2022b, via ESP8266/ESP32 modules. Although the system uses Simulink to design high-level controllers, it is not based on any custom embedded drivers, S-functions, or low-level modelling of the hardware, instead it uses standard controller blocks and IoT-based data transmission.

Most of the previous studies employ conventional blocks of the Simulink library that do not allow direct control of the algorithmic behaviour. Consequently, they do not pay much attention to sophisticated modelling tools and bottom-level interaction with hardware. This puts a definite gap separating the traditional Simulink blocks from the more advanced features offered by the Simulink Arduino Support Package, which enables user-defined algorithms and integration to the driver level. To address this gap, our contribution introduces advanced Simulink integration that leverages the Arduino support package, custom driver blocks, extended sensor libraries, filtering, map-equation analysis, and real-time visualization via the ThingSpeak application. This offers greater flexibility and algorithmic depth beyond simple block-based device control, making it suitable for embedded control and data acquisition across a wide range of applications.

The rest of the paper is structured as follows: the second section presents the materials and methods, including system architecture, sensor integration technique, and control model design. The third section provides the findings and discussion, emphasizing the efficiency of the proposed approach, and the last part is the conclusion of the paper.

II. MATERIALS AND METHODS

A. Study Area

The experiment was performed in Bechar, Southwest, Algeria. Algeria is among the countries with arid areas, where the desert covers more than 80% of its total territory (Bona et al., 2021). The dry environmental conditions of these areas challenge and concern farmers. Due to water scarcity, farmers rely on groundwater as the primary irrigation source. Bechar is among such common desert-climate cities, which are characterized by a substantial temperature contrast (the highest average maximum temperature is 40°C, and the lowest one is 15°C in January). Traditional agriculture is a challenge with a low rainfall rate, which rarely exceeds 90 mm per year, and frequent sandy winds during the half-season. To survive, it needs to use advanced irrigation methods and use modern farming methods adapted to arid conditions (Weather Atlas, 2024). It involves high-tech irrigation systems and modern agricultural techniques adapted to suit aridity to be viable (Weather Atlas, 2024).

B. Overview

This section demonstrates a case study of an intelligent irrigation system to exemplify our proposed contribution. The system depicts a representative architecture for sophisticated methods, such as embedded S-function development, additional library integration to adopt custom driver blocks for sensors, sensor reading filtering and mapping, and the provision of on/off control logic implementation. Furthermore, it also enables real-time data transmission and visualization via the ThingSpeak platform. The hardware implementation was done using an Arduino (ATMega 2560) microcontroller with a W5100 Ethernet shield. The whole system was designed and tested in a Simulink environment, proving a viable scalable and cost effective IoT solution to smart irrigation

in water-deficient and economically distressed agricultural areas

C. Materials

1) System Architecture

The proposed system has four fundamental structural and functional layers, which are sensor input, processing and control, actuation, and cloud communication, as shown in Figure 1.

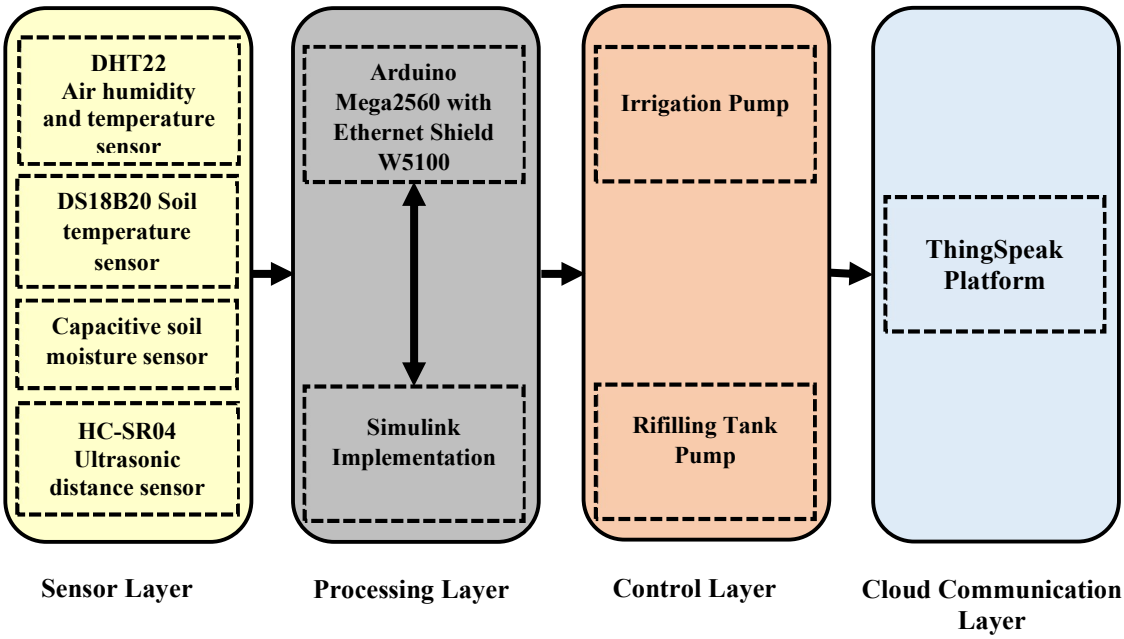


Figure 1 Block Diagram of The Irrigation System

1. **Sensor Layer**: A set of sensors is responsible for collecting environmental data. The sensors used, along with their technical specifications, are presented in Table 1

2. **Processing Layer**: Collecting and processing measurement data was studied using the Simulink® R2020a simulation environment running on a PC connected to an Arduino Mega 2560 microcontroller using serial communication.

3. **Control Layer**: The control layer operates on a threshold-logic model that controls the ON/OFF states of two water pumps: irrigation and tank refilling.

4. **Cloud Communication Layer**: Real-time remote monitoring is enabled via an Ethernet shield W5100 for all collected sensor and control data through the ThingSpeak platform

2) Circuit Diagram

The proposed system's wiring diagram is shown in Figure 2. The process involves an Arduino Uno microcontroller, two 12 V DC power supplies, a 2-channel relay module, and four environmental sensors: a DS18B20 temperature sensor, a DHT22 temperature and humidity sensor, an ultrasonic distance sensor, and a capacitive soil moisture sensor. Three pull-up resistors with the resistance values of 220Ω,110Ω and 1 MΩ to the DS18B20, DHT22, and capacitive soil

Table 1 Characteristics of the Sensors Used

Parameter	Sensor type	Measuring range	Accuracy	References
Soil moisture	Capacitive soil moisture V1.2	0 to 100 VWC (volumetric water content)	±0.08%	(DFRobot, 2024; Schwamback et al., 2023)
Soil temperature	DS18B20	-55 to 125℃	±0.5℃	(Alldatasheet, 2024)
Air temperature	Dht22	- 40 to 80℃	± 0.5℃	(Liu, 2024)
Air humidity	Dht22	0 to 100% RH	± 2 – 5% RH	(Liu, 2024)
Water level	Hc sr04	2 to 400cm	± 3mm	(Morgan, 2014)

moisture sensors, respectively. They were incorporated to improve the precision and reliability of the sensor readings, by eliminating signal interference and ensuring appropriate voltage levels.

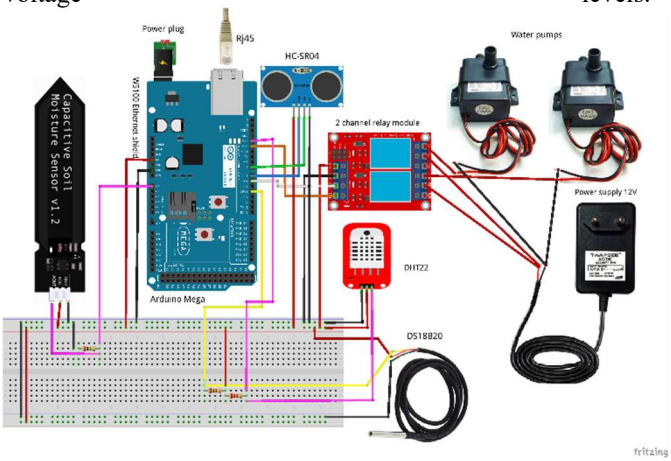


Figure 2 Wiring Diagram of The System

D. Methodology

The proposed system is implemented using a combination of hardware integration and Simulink-based modeling. The entire system presented is based on a central control system consisting of an Arduino (Mega 2560 microcontroller and a W5100 Ethernet shield module for internet access. It consists of a set of sensors, including a soil moisture sensor, a DS18B20 soil temperature sensor, a DHT22 air temperature and humidity sensor, and an HC-SR04 water tank level sensor. To generate code and deploy to hardware, the control logic and simulation model were created in Simulink 2020a, using the Simulink Support Package for Arduino Hardware. These sensors are incorporated with a custom driver block coded using the S-function algorithm to read data from the DS18B20 sensor. In the meantime, additional Arduino libraries were added to support more Simulink blocks for the DHT22 and the ultrasonic sensors, which are not natively supported by the standard support package. For the soil moisture input data, raw digital values were converted into calibrated physical data using the Arduino map function and then were smoothed by a low-pass filter to reduce noise on a high-frequency scale and increase the stability of measurements. This resulted in smoother patterns and more proactive management of irrigation. The system workflow is constructed around collecting primary measurements from sensors. Using pre-established thresholds, the system manages the switching of two water pumps, one dedicated to irrigation and the other for refilling the water tank, thereby providing efficient water management. The Ethernet shield transmits all the collected and processed data to the ThingSpeak IoT platform, allowing remote monitoring with real-time visualization, continuous logging, and trend analysis. The procedure was driven by a modular methodology that performed the following functions as represented in a structural diagram and flowchart, as illustrated in Figures 3 and 4 respectively.

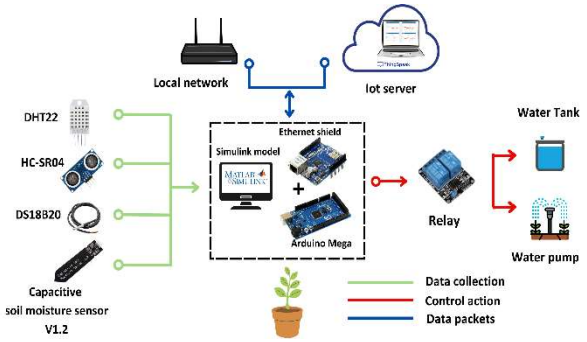


Figure 3 Structural Diagram of the Irrigation System

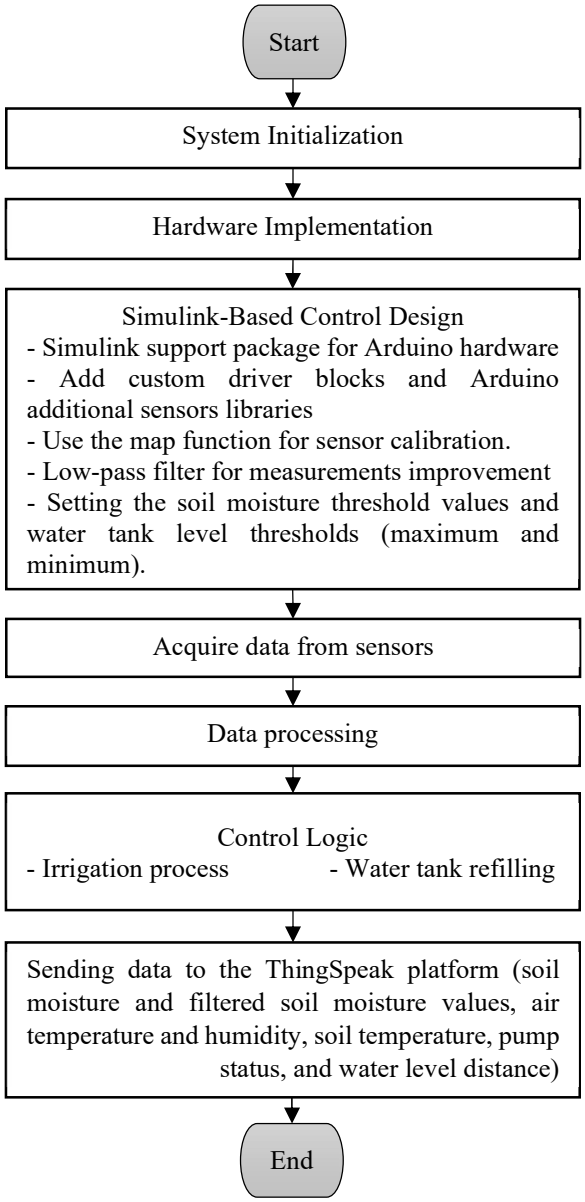


Figure 4 Flow Chart of the Irrigation System

Figure 5 presents the entire hardware design, representing a field-scale, miniature, prototype implementation of the

proposed system, designed for reliability and efficiency in a real-world agricultural environment.

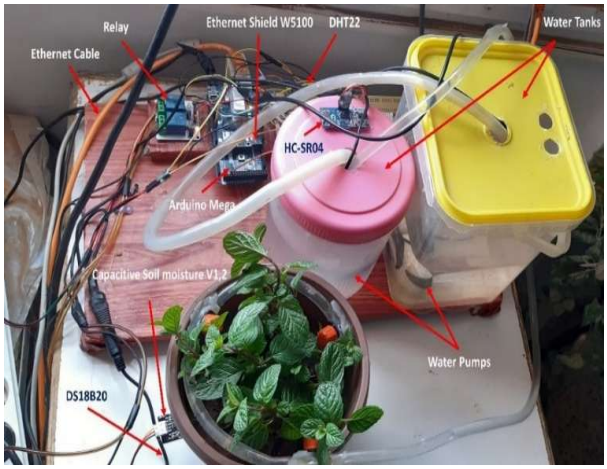


Figure 5 View of Complete Hardware Setup

1) Irrigation Control

The control of irrigation involves three main steps, calibration of the soil moisture sensor readings, data filtering, and the control logic. The following sections expound these steps in a bid to demonstrate the entire process of the irrigation system

i. Calibration and Mapping of Soil Moisture Sensor

The analog readings of the soil moisture sensor are nonlinear and deficient in a physical interpretation. An Arduino map equation was adopted for the calibration and mapping process. The map equation is a linear transformation that remaps an input value from one to another. Given an input value, it computes a proportional position of the input value within the input range, and transforms it into a specified output range. This equation is essential in embedded projects and control systems where sensor values must be changed to useful parameters such as temperature, range, or percentage (Arduino, 2024; Barrozo, 2024; Zafeiriou, 2025). It is mathematically written:

$$out_{\text{map}} = \frac{(x - in_{\text{min}}) \times (out_{\text{max}} - out_{\text{min}})}{(in_{\text{max}} - in_{\text{min}}) + out_{\text{min}}} \quad (1)$$

where:

Out_{map} *is* the mapped output
x is the Input to map
in_{min} is the minimum input to map
in_{max} is the maximum input to map
out_{min} is the minimum mapped output
out_{max} is the maximum mapped output

The analog values obtained by reading the capacitive soil-moisture sensor (usually varying between 0 to 1023) were initially experimentally calibrated with two reference points, one is the minimum wet point obtained when the sensor is immersed in saturated soil, and the other the is the maximum dry point when the sensor is inserted in completely dried soil. The recording measured analog results, as given in Table 2.

Table 2 Sensor readings calibration

Soil condition	Analog readings
Dry	468
Wet	210

Using the map equation, the analogue readings are converted to precise percentage values ranging from 0 to 100% based on the parameter values listed in Table 3. The map equation is included in the Simulink model using a custom equation block, as shown in Figure 6.

Table 3 Map equation parameters

Map parameters	Values
x	0 − 1023
in_{min}	210
in_{max}	468
out_{min}	100
out_{max}	0

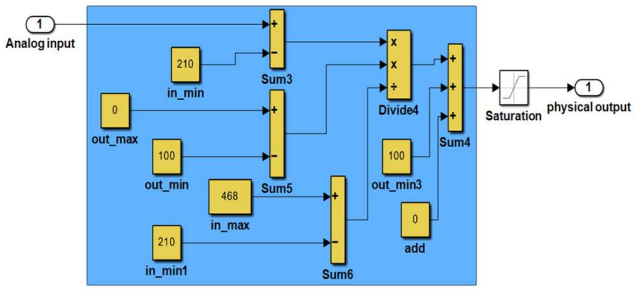


Figure 6 Soil Moisture Mapping Process

ii. Soil moisture filtering

The obtained capacitive soil moisture data are unstable and fluctuate. Under normal operating conditions, a first-order low-pass filter was used to improve measurement consistency and reduce high-frequency noise. A first-order low-pass filter is an electronic circuit that allows low-frequency signals to pass while attenuating those above a defined cutoff frequency. It is widely used in electronic circuits and signal-processing applications. (DigiKey, 2025). Its discrete-time formulation is typically expressed as in Eq. (2) (Helpful, 2024)

$$y(i) = \alpha \times u(i) + (1 - \alpha) \times y(i - 1) \quad (2)$$

Where:
$y(i)$ is Filtered output

α is the smoothing factor
$u(i)$ is the input signal
$y(i-1)$ is the previous filtered output

Where α *is* calculated based on Eq. (3):

$$\alpha = \frac{1}{1+2\pi \times cutoffreq \times \tau_s} \quad (3)$$

Where;

α is the smoothing factor
$cutoffreq$ is the cut-off frequency
$\boldsymbol{\tau_s}$ is the sample time

The low-pass filter coefficients are manually adjusted and fixed. The results are summarized in Table 4 and implemented into Simulink, as shown in Figure 7

Table 4: Adjusted Coefficients of the Low-Pass Filter

Coefficients	Values
Cutofferq	50
Ts	100
$\boldsymbol{\alpha}$	0.0032

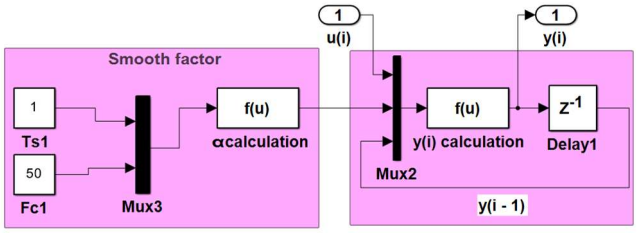


Figure 7 Low-Pass Filter

iii. Control mechanism

The irrigation control mechanism was designed using a threshold-based on/off control logic. Figure 8 presents the implemented ON/OFF controller. The irrigation pump was controlled based on the filtered soil moisture reading. According to whether the comparison with the preset value of 5% is completed, the control signal is alternated between 1 and 0, and the pump is switched on or off accordingly. The binary method of plant management applies in instances where irrigation is regulated to avoid overwatering while maintaining adequate soil moisture. Figure 8 represents the design of the entire irrigation control model implemented in Simulink. Figure 9 presents the corresponding flowchart simultaneously.

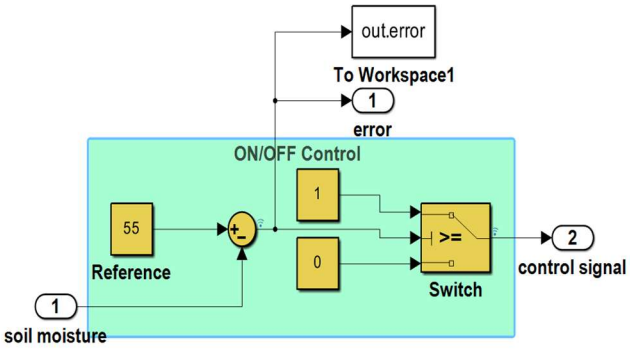


Figure 8 On/Off Control

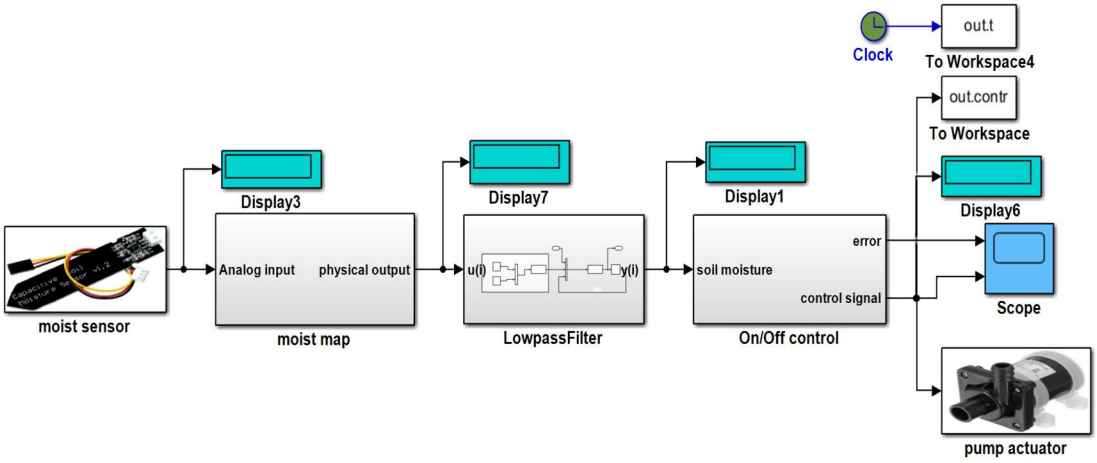


Figure 9: Simulink Model of Irrigation Control

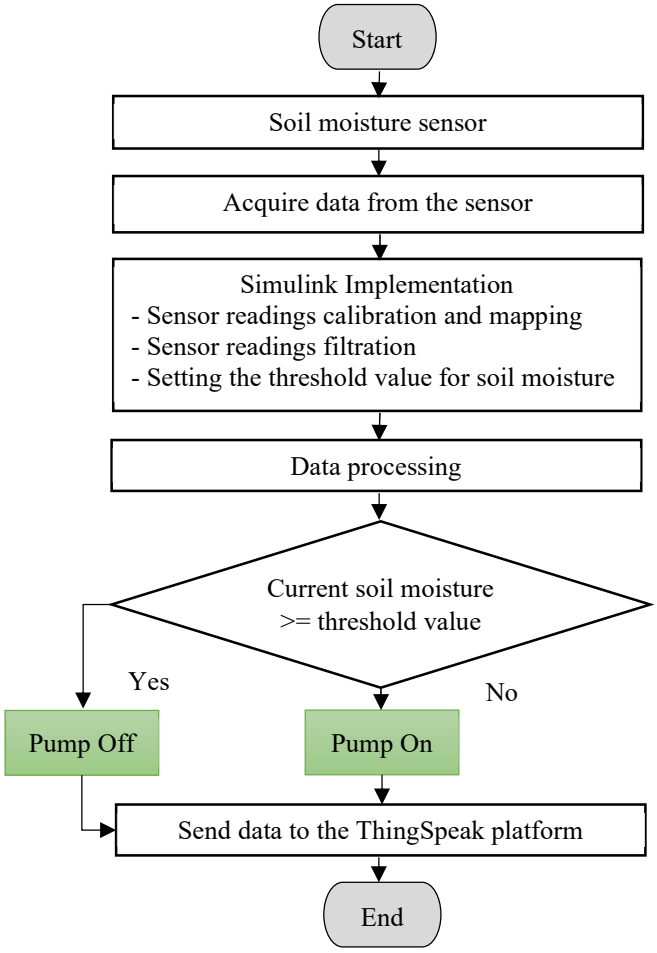


Figure 10 Flowchart of the Irrigation Control

2) Water Level Regulation System

To provide a sufficient water supply for irrigation, a model for controlling water tank level was modeled and implemented to regulate the pump in the Tank. The level of the water tank was measured using a model with an ultrasonic distance sensor block adopted from the Rensselaer Arduino Support Package Library. This package is a Simulink sensor and motor driver library for Arduino, including the ultrasonic distance block (Hurts, 2024). To control the pump, a custom code was generated using a MATLAB function based on current read distance, previous distance, and pump status. This results in the pump being driven by constant comparison of sensor readings with preset limits (7 cm activation and 5 cm deactivation). Figure 11 illustrates the Simulink model structure, while the generated code is presented in Algorithm 1.

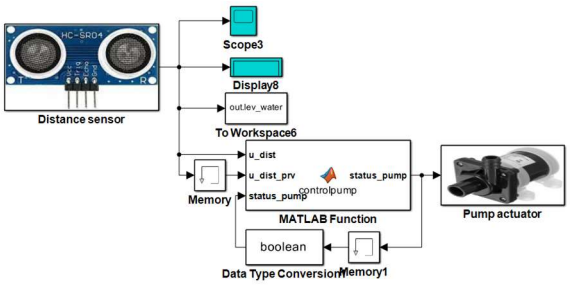


Figure 11 Simulink Model of Water Tank Level Control

Algorithm 1: Pseudocode for Water Level Control

```
function status_pump = controlpump(u_dist,
         u_dist_prv, status_pump)
          u_distOn = 7;
          u_distOff = 5;

    if u_dist == u_distOn && ~status_pump
                    status_pump = true;
```

```
    elseif u_dist <= u_distOff &&
status_pump
                status_pump = false;
          elseif u_dist == u_distOn &&
   ~status_pump && u_dist_prv ~= u_distOn
                status_pump = true;
    end
  end
```

3) Soil Temperature Monitoring

To enhance the accuracy of the system further, a DS18B20 digital temperature sensor was incorporated to monitor soil temperature accurately, as it has a high resolution and one-wire communication capabilities (Alldatasheet, 2024). We integrated it into the Simulink model by developing a custom driver block using S-Function Builder. The latter is a Simulink tool that allows the ability to create a custom driver by inserting C/C++ code via a graphical interface, automatically producing a wrapper that can be used in the model. This method allows direct communication between the DS18B20 sensor and the control model, enabling real-time acquisition of soil temperature data. Creating a custom driver with the S-Function Builder in Simulink involves several steps (Mathworks , 2025d; Tyler, 2020).

i. Include folder creation

After inserting the S-Function Builder block into a model, create an include folder at the following path in the project directory to store header files and supporting code: C:\Users\Client\Documents\Arduino\libraries. The included folder is shown in Figure 12.

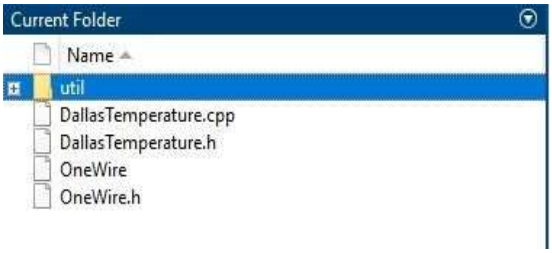


Figure 12 Include Folders

ii. S-function builder construction

The S-function was created using the S-Function Builder, where it was first named and then set up through the block's parameter tabs. The initialization tab was used to configure the S-function's settings, including sample mode, sample time, and other initial conditions. S-function ports and their parameters were specified in the data properties tab. The libraries tab was used to identify any necessary include files, external function declarations, or source files. The Outputs tab was described as the section for writing C/C++ code. Finally, the update tab provides the section that calculates the current state value and propagates it to the next step, ensuring consistency between the generated code and the model's execution. Subsequently, the block program was launched by clicking the build icon. All these steps are represented respectively in Figures 13, 14, 15, 16, 17.

Figure 13 S-function Initialization

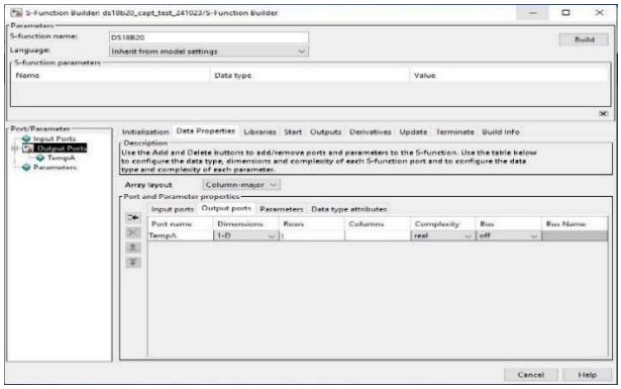


Figure 14 S-function Data Properties

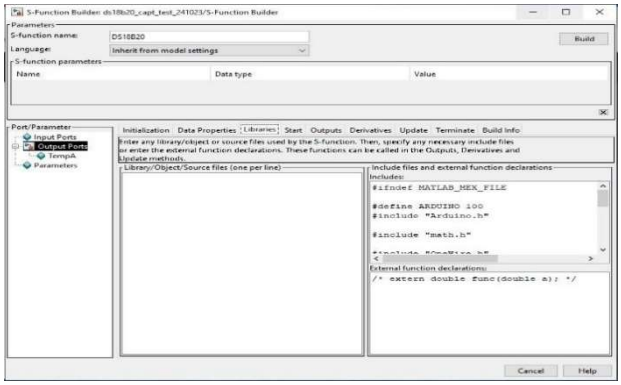


Figure 15 S-function Libraries

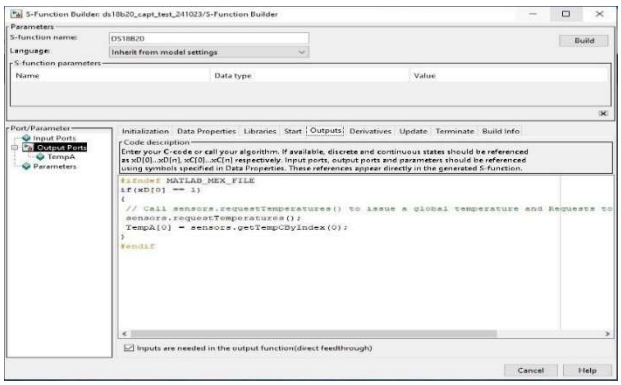


Figure 16 Output Code

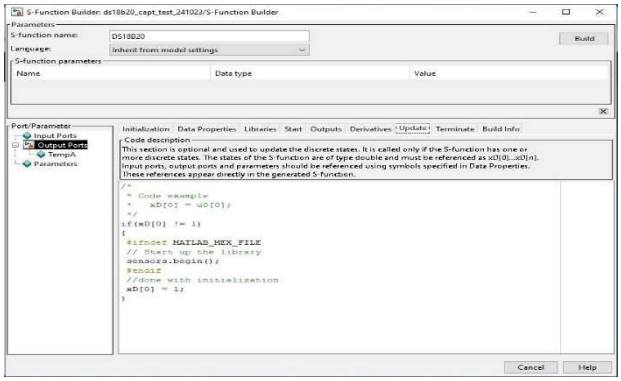


Figure 17 Update Code

After creating the driver, the file "ds189b20pr_wrapper.c" was modified to a C++ extension file "ds189b20_pr_wrapper.cpp", and extern "C" was added before each "void" as illustrated in Figure 18.

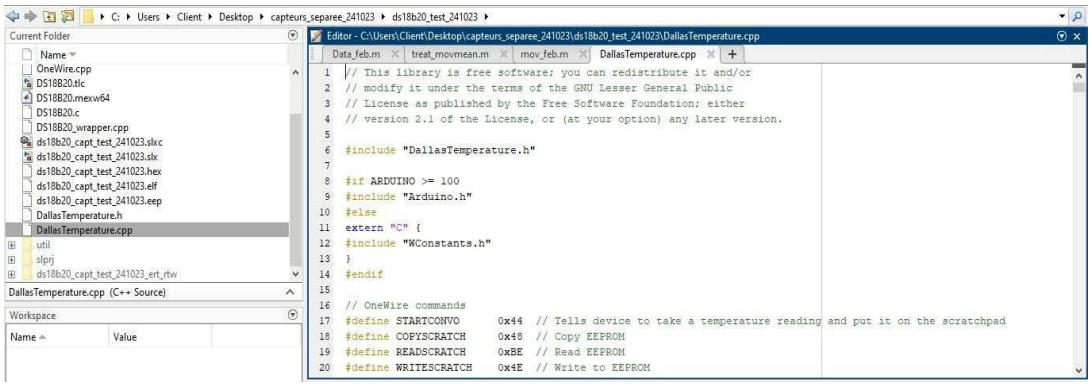


Figure 18 Converting the DS18B20 Wrapper File

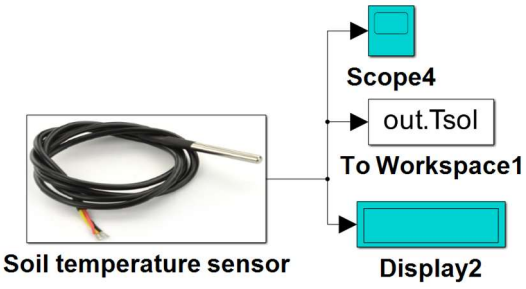


Figure 19 Simulink Model of Soil Temperature Sensor

4) Air Temperature and Humidity Monitoring

To analyse ambient air conditions, an ambient air temperature and humidity sensor (DHT22) was adopted from an additional installed library. This additional Arduino library (DHT, LPS331) includes Simulink blocks for the DHT and LPS331 sensors, simplifying integration into the model's design (Roslovets, 2024). Unlike the standard Arduino blocks, which require additional work to calibrate raw signals and implement protocols, this library provides a ready-to-use solution. Figure 20 presents its implementation in Simulink**.** We installed the Arduino Additional Sensors library by following the steps outlined below:

a. Download the library files from the relevant source (Roslovets, 2024).

b. Unzip the downloaded files to any folder without spaces in the path

c. Run "INSTALL.m"

Implementation of the Simulink soil temperature sensor is shown in Figure 19.

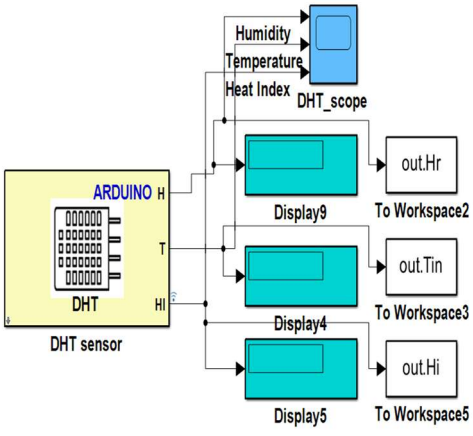


Figure 20 Simulink Model of Air Temperature and Humidity Sensor

5) IoT Application

The designed model incorporates an IoT layer that enables the upload and real-time monitoring of measurement data on the ThingSpeak platform. All these gathered data (soilmoisture, filtered soil moisture, soil temperature, air temperature and humidity, pump status, and water tank level) are sent at regular intervals through a standard component, ThingSpeak Write block, from the Simulink Support Package for Arduino Hardware. The cloud platform stores and presents this type of data in dynamic dashboards and time-series charts. This feature allows users to have access to information wherever they are and gives a real-time record of the performance of the system.

the complete system model in Simulink is illustrated in Figure 21.

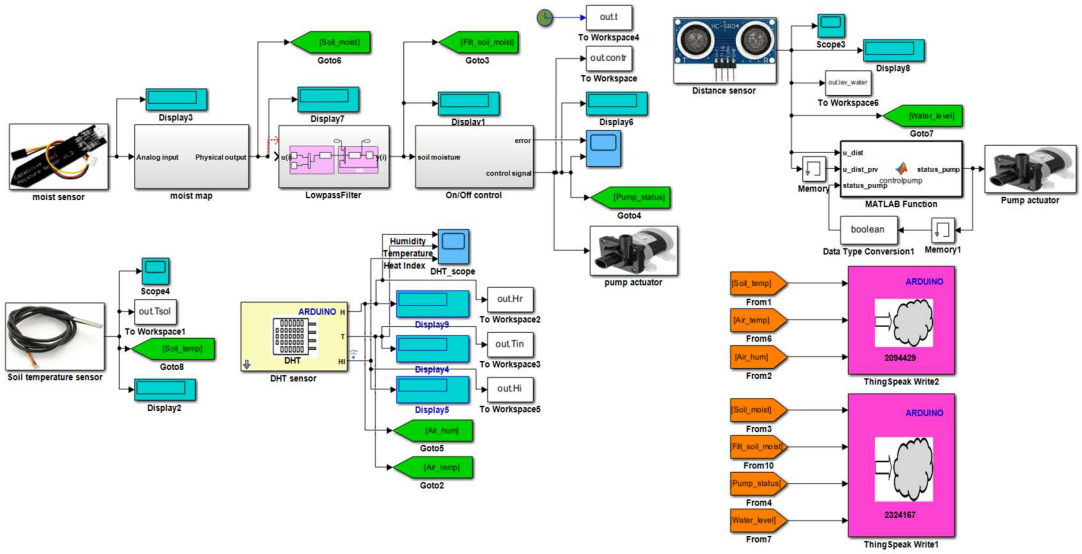


Figure 21 Simulink Model of Soil Temperature Sensor and of Air Temperature and Humidity Sensor

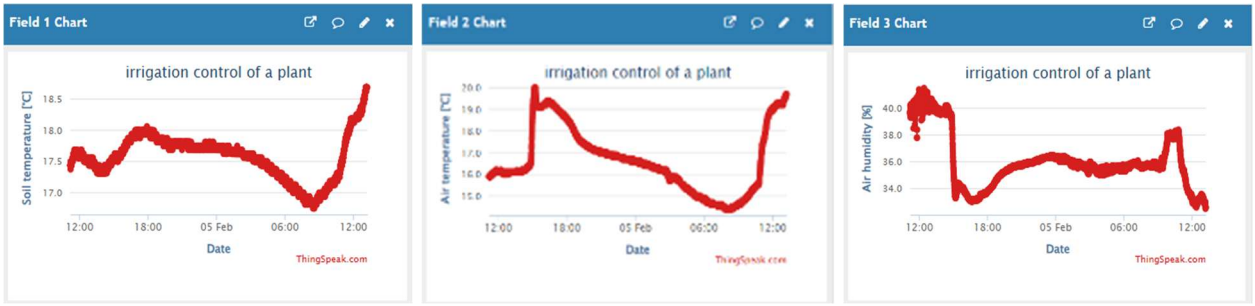
III. RESULTS AND DISCUSSION

The findings of the proposed system are introduced in this section. The results outline how the applied system techniques, such as control, sensing, calibration, filtering, and driver integration, affect the overall performance.

The ENERGARID laboratory at the Tahri Mohamed University of Bechar, Algeria housed a potted plant as a test sample. The main goal of the experiment was to evaluate the functionality of the suggested method. A one-week period was used for data collection in the month of February, when 4459 datasets were obtained on the various types of soil-related and ecological variables. Measurement was uploaded to the ThingSpeak platform every 20 seconds of sample time. The results were displayed as dynamic graphs to be tracked and evaluated.

The Simulink model allows exporting its internal signals into the MATLAB workspace during simulation by converting real-time model outputs into MATLAB variables through the To Workspace block and data acquisition mechanisms. This enables the simulation results to be stored, plotted, and analytically processed for further evaluation and performance validation. The architecture of

Figures 22 (a), (b) and (c) represent the time-series data of soil temperature, air temperature, and air humidity that indicate the daily trends and small environmental deviations during the test time. Temperature and humidity measurements were comparatively smooth and stable during the observation. Air temperature rose during the day and fell during the nights, whereas the reverse was observed for humidity. The temperature of the soil changed more slowly than the temperature of the air, which points to greater thermal stability. These trends indicate the validity of the integrated DHT22 and DS18B20 sensors in real-time measurements of microclimate.

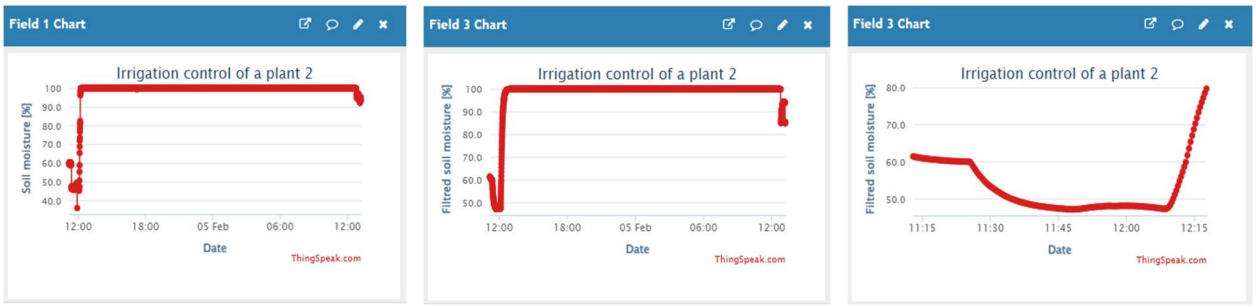


(a) (b) (c)

Figure 22 ThingSpeak Charts of Recorded Data: a) Soil Temperature; b) Air Temperature; c) Air Humidity.

Figure 23 shows the output of the filtering process and its combination with threshold-based irrigation management. Figure 23(a) indicates that unfiltered soil moisture measurements are highly dynamic and noisy and hence can impact pump activation. Once the filtration technique was applied, the measurements became more consistent and smoother, as shown in Figure 23(b), providing a reliable basis for making irrigation decisions. Expanding on this, Figure 23(c) shows soil moisture data by zooming in on it over a given time interval.

A threshold of 55% was set to activate the pump at a point when the soil moisture reaches that threshold, the pump switches on and is off when the desired value is attained. Combining signal-filtering logic with pump-control logic will minimize false trigger events due to signal noise, which will ensure more effective irrigation, less significant wastage of water and energy, and the maintenance of optimum soil-moisture conditions.

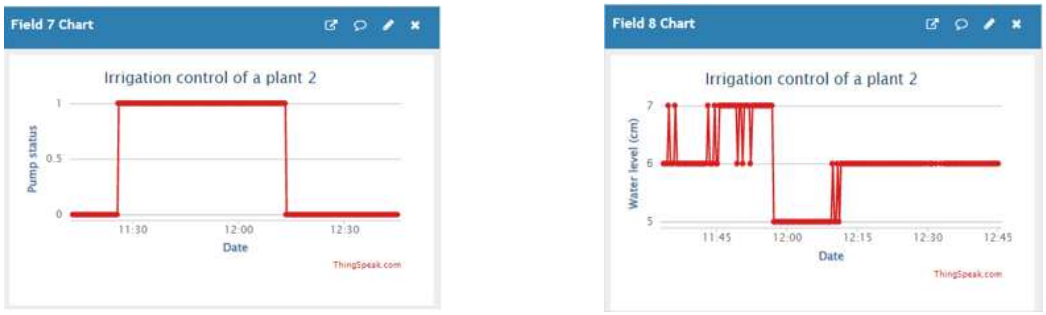


(a) (b) (c)

Figure 23 ThingSpeak Charts of Gathered Soil Moisture Values: a) Gathered Data Before Filtration; b) Filtered Gathered Data; c) Filtered Gathered Data Zoomed in on a Particular Time

Figures 24(a) and 24(b) show the operating condition of the irrigation pump and associated water-level changes in the storage tank. The findings illustrate that there is a distinct correlation between the pump and soil-moisture dynamics: the irrigation pump is activated within the predetermined threshold range, turning on when the soil moisture level falls to the set point, and off once the desired level has been attained again. Simultaneously, the tank-refill system works in a regular way, initiating replenishment exactly when the water level drops below the lower set point of 7 cm Overall,

the findings support the efficacy of threshold-based control in achieving responsive irrigation and steady water management. ThingSpeak enabled real-time monitoring and validation of environmental parameters and control performance. The integration of a Simulink-based control framework with IoT thus offers a scalable, data-driven solution for intelligent irrigation in resource-limited conte



(a) **(b)**

Figure 24 ThingSpeak Charts of Pump Status and Water Tank Level: a) Pump Status; b) Water Tank Level.

IV. CONCLUSION

The purpose of this research was to design and prove a Modular/Simulink based IoT embedded implementation and real time acquisition and control of data system. A case study of intelligent irrigation scenario was used to demonstrate its potential given that the proposed framework was specifically designed in arid regions. The system was simulated using the Simulink environment and programmed to an Arduino Mega using an Ethernet shield. It offers an expandable and programmable sensor integration, processing and actuation. Use of custom driver block development such as S-function algorithm, longer sensor libraries, embedded calibration and filtering algorithms guarantees workflow flexibility. This framework further enables precise on/off control of dual water levels, demonstrating effective water-level management and targeted irrigation for improved efficiency. Being a monitoring layer of the workflow, the system facilitates the transmission of data to the ThingSpeak cloud server for real-time visualization, logging, and analyzing trends. The demonstration was practical in scope and showed the functionality of the system in terms of both control and monitoring, a case that is common in precision irrigation applications.This contribution simplifies sensors integration, data acquisition and real-time cloud computing technology, and as such is applicable to numerous intelligent control and automation uses beyond irrigation.

AUTHOR CONTRIBUTIONS

H. E. Abbassi: writing — original draft, methodology, data curation, software, hardware. **K. Lammari:** Software, conception, experimental design, reviewing, hardware. **F. Bounaama:** Conception and experimental design, investigation, methodology, writing—reviewing and editing, project administration, supervision, Simulink compilation. **S. Bennaceur:** Hardware.

REFERENCES

Al-Omary, A., H. M. AlSabbagh & H. Al-Rizzo (2018). Cloud based IoT for smart garden watering system using Arduino Uno. *Smart Cities Symposium (SCS '18)*, University of Bahrain, 249–254. https://doi.org/10.1049/cp.2018.1401

Alldatasheet (2024). DS18B20 Dallas. Available online at: https://www.alldatasheet.com. Accessed July 20, 2025.

Arduino (2024). map() – Arduino reference. Available online at: https://www.arduino.cc/reference/en/language/functions/math/map/. Accessed July 29, 2025.

Barrozo, J., (2024). How to Use the Arduino map() Function: A Complete Guide. *Flux AI Blog*, 11 July. Available online at: https://www.flux.ai/p/blog/how-to-use-the-arduino-map-function-a-complete-guide. Accessed September 1, 2025.

Belkadi, A., D. Mezghani & A. Mami (2020). Design and implementation of FLC applied to a smart greenhouse. *Engenharia Agrícola*, 40(6), 777–790. https://doi.org/10.1590/1809-4430-Eng.Agric.v40n6p777-790/2020

Bona, E., N. Massa, O. Toumatia, G. Novello, P. Cesaro, V. Todeschini, L. Boatti, F. Mignone, H. Titouah, A. Zitouni, G. Lingua, F. Vuolo & E. Gamalero (2021). Climatic Zone and Soil Properties Determine the Biodiversity of the Soil Bacterial Communities Associated to Native Plants from Desert Areas of North-Central Algeria. *Microorganisms*, 9(7), 1359. https://doi.org/10.3390/microorganisms9071359

Caseiro, L., D. Caires & A. Mendes (2022). Prototyping Power Electronics Systems with Zynq-Based Boards Using Matlab/Simulink—A Complete Methodology. *Electronics*, 11(7), 1130. https://doi.org/10.3390/electronics11071130

DFRobot. (2024). Capacitive soil moisture sensor SKU SEN0193. Available online at: https://wiki.dfrobot.com/Capacitive_Soil_Moisture_Sensor_SKU_SEN0193. Accessed on July 20, 2025.

Helpful (2024). Filters on data. Available online at: https://helpful.knobs-dials.com/index.php/Filters_on_data. Accessed July 29, 2025.

Hurts, J., (2024). Rensselaer Arduino Support Package Library (RASPLib). GitHub. Available online at: https://www.mathworks.com/matlabcentral/fileexchange/62702-rensselaer-arduino-support-package-library-rasplib. Accessed August 25, 2025.

Jackson, D., B. Tannenbaum & W. Jachimczyk (2006). Adoption, impact, and vision of model-based design. *Proceedings of SPIE, the International Society for Optical Engineering*, 6228, 622817. https://doi.org/10.1117/12.665078

Kshirsagar, K., P. Shah & R. Sekhar (2022). Model based design in industrial automation. *2022 6th International Conference on Computing, Communication, Control and Automation (ICCUBEA)*, 1–6. https://doi.org/10.1109/iccubea54992.2022.10010895

Laktionov, I. S., O. V. Vovna, Y. O. Bashkov, A. A. Zori & V. A. Lebediev (2019). Improved Computer-oriented Method for Processing of Measurement Information on Greenhouse Microclimate. *International Journal of Bioautomation*, 23(1), 71–86. https://doi.org/10.7546/ijba.2019.23.1.71-86

Liu, T., (2024). DHT22 (AM2302) digital-output relative humidity & temperature sensor/module. Aosong Electronics Co., Ltd. Available online at: https://www.alldatasheet.com. Accessed July 20, 2025.

Mashhadany, Y. A., H. R. Alsanad, M. A. Al-Askari, S. Algburi & B. A. Taha (2024). Irrigation intelligence—enabling a cloud-based Internet of Things approach for enhanced water management in agriculture. *Environmental Monitoring and Assessment*, 196(5), 438. https://doi.org/10.1007/s10661-024-12606-1

MathWorks (2025a). Build S-Functions Automatically Using S-Function Builder. MATLAB Simulink Documentation. Available online at: https://www.mathworks.com/help/simulink/slref/sfunctionbuilder.html. Accessed August 25, 2025.

MathWorks (2025b). Internet of Things – MATLAB & Simulink for IoT applications. Available online at: https://www.mathworks.com/solutions/internet-of-things.html. Accessed August 25, 2025.

MathWorks (2025c). Model-based design. Available online at: https://www.mathworks.com/solutions/model-based-design.html. Accessed August 25, 2025.

MathWorks (2025d). Simulink Documentation. Available online at: https://www.mathworks.com/help/simulink/index.html. Accessed August 25, 2025.

Morgan, E. J., (2014). HC-SR04 ultrasonic sensor. Alldatasheet. Available online at: https://www.alldatasheet.com. Accessed July 20, 2025.

Nicola, C., M. Nicola, M. Niţu, D. Sacerdoţianu & A. Aciu (2021). Real-Time Sensorless Control of the PMSM based on Genetic Algorithm, Sliding Mode Observer, and SCADA Integration. *Annals of the University of Craiova Electrical Engineering Series*, 45(1), 24–31. https://doi.org/10.52846/aucee.2021.1.04

Roslovets, P., (2024). Arduino additional sensors library (DHT, LPS331). GitHub. Available online at: https://github.com/roslovets/Arduino_Additional_Sensors_Simulink_Library_DHT_LPS331/releases/tag/v1.0.1. Accessed August 25, 2025.

Schwamback, D., M. Persson, R. Berndtsson, L. E. Bertotto, A. N. A. Kobayashi & E. C. Wendland (2023). Automated Low-Cost Soil Moisture Sensors: Trade-Off between Cost and Accuracy. *Sensors*, 23(5), 2451. https://doi.org/10.3390/s23052451

Tyler, N., (2020). How to integrate DS18B20 with Arduino in Simulink / Simulink S-function builder basics-1 temp Probe1. YouTube. Available online at: https://www.youtube.com/watch?v=0OClHIdCyDg. Accessed August 25, 2025.

United Electronic Industries (2025). Model-Based Design Tutorial & Reference Guide – Aerospace DAQ, Test, HIL. UEI. Available online at: https://www.ueidaq.com/model-based-design-tutorial-reference-guide. Accessed August 20, 2025.

Weather Atlas (2024). Yearly & monthly weather – Béchar, Algeria. Available online at: https://www.weather-atlas.com/en/algeria/bechar-climate. Accessed July 20, 2025.

Wilcher, D., (2025). What Is a Low Pass Filter? Understanding an Electronic Filter Circuit. DigiKey Maker.io. Available online at: https://www.digikey.com/en/maker/tutorials/2025/what-is-a-low-pass-filter-understanding-an-electronic-filter-circuit. Accessed September 1, 2025.

Zafeiriou, S., (2025). Arduino map() Function. Steve Zafeiriou. Available online at: https://stevezafeiriou.com/arduino-map-function/. Accessed September 1, 2025.