Internet of Things Smart Farming Architecture for Agricultural Automation

Adrián Sánchez-Mompó, Heloise Barbier, Won-Jae Yi and Jafar Saniie
Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago IL, U.S.A.

***Abstract*— With the increase of global demand for food and with the adoption of the trends of local sourcing and locally grown produce, the need for gardening and farming automation solutions ranging from industrial to home level has been on the rise. In this paper, we propose an Internet of Things (IoT) farming control system based on the concept of Wireless Sensor and Actuator Networks (WSAN) that provides ideal growing conditions for user-defined crops. This is achieved by utilizing the information provided by a series of sensors monitoring the environmental (temperature, humidity, UV, etc.) and soil (moisture, nutrients, etc.) conditions to control the deployed actuators. To allow a wide range of deployment sizes, we use a two-stage system that combines a series of low-power sensor and actuator nodes with a communication and data processing gateway. The low-power microcontroller reads the data from the sensors and sends them to the data processing gateway, which then calculates the optimal actuator changes required to achieve the desired status. These changes are then feedbacked to the low-power microcontroller which actuates the control devices. The communication between the devices is performed via a bespoke LoRa-based communication protocol optimized for minimal overhead, flexibility, and guaranteed data delivery. This is presented to the user via a website to view the configuration of the system and the observation of the past and present environmental, soil, and actuator status of the system.**

Keywords—Smart Farming, Agricultural Automation, IoT, WSAN, LoRa.

I. Introduction

In the last decade, new trends have appeared focusing on the reduction of greenhouse effect emissions that might accelerate a climate change affecting coastal cities and infrastructures due to the rise of sea level [1], as well as endangered species that can suffer changes in the timing of migration, population sizes and population distributions [2]; these trends are also aimed at the reduction of other gas emissions that have a detrimental effect on the health of the human population [3]. One of the many strategies used to reduce these harmful emissions is local sourcing, which consists of procuring consumables such as food from a small radius of distance. This helps reduce the carbon footprint generated when transporting food from its production location to its consumption place, which represents 30% of the CO_2 emissions derived from food consumption [4]. Local food sourcing has led to the proliferation of home and community farms, including the trend of rooftop gardens in metropolitan areas. In addition, global demand for food has seen an increase with the rise in the global population [5]. With many greenhouses and crop farming facilities still lacking automation, there is a route to increasing productivity, land utilization, and efficiency of existing farms via the introduction of automation measures that lead to better growing conditions [6]. This could help to meet future food demand, in conjunction with other measures, while reducing the potential negative impact on the environment [7].

By utilizing the concepts of Wireless Sensor and Actuator Networks (WSAN) and the Internet of Things (IoT), these farms and greenhouses can be automated, reducing the work hours needed, accelerating the growth of the crops, helping save water, and obtaining higher production yields. Furthermore, by using the LoRa modulation technology [8] (not to be confused with the LoRaWAN architecture [9]) as the base for a custom communications protocol, the system can be flexible in regards to the deployment size to accommodate scenarios ranging from individual farms to industrial-level greenhouses.

Some projects have focused only on irrigation automation [10], and some others have sought total automation of all farming tasks, from seeding to recollection [11], [12]. Our proposal aims to provide time-saving and water-saving automation while maintaining a low-cost approach. We target a highly adaptable system able to work with a different number of sensors and actuators while accommodating different deployment sizes, and providing a wide choice depending on the resources available and the needs of the crops.

II. Smart Farming System Design

The crop farming automation architecture proposed, developed, and demonstrated in this paper has three main parts, as shown in Figure 1. These are: 1) a series of low-power sensor and actuator nodes equipped with LoRa communications and with sensors and/or actuators; 2) a control gateway that communicates with the low-power nodes via LoRa, which makes the decisions for the actuator control, and forwards a copy of the sensor and control data to the database; and 3) a database and a webpage server which provides a user interface that allows users to view and control the state of the system. The connection between the nodes and the gateway is handled via a bespoke LoRa-based communications protocol, designed specifically for this task, which offers the flexibility, scalability, and robustness required for a reliable system operation while minimizing the data and processing overheads.

We have intentionally left the selection of the control algorithm (e.g., actuator control to achieve optimal growing conditions), and hardware (nodes, gateway, sensors, and actuators) undecided, as each application and implementation has different requirements. The algorithm choice could range from a simple solution (e.g., range control, where a value lower than a first threshold activates an actuator and a value higher than a second threshold deactivates it), to more complex solutions that benefit from Proportional-Integral-Derivative (PID) control [13] or Machine Learning control, as used in some control solutions [14], [15].

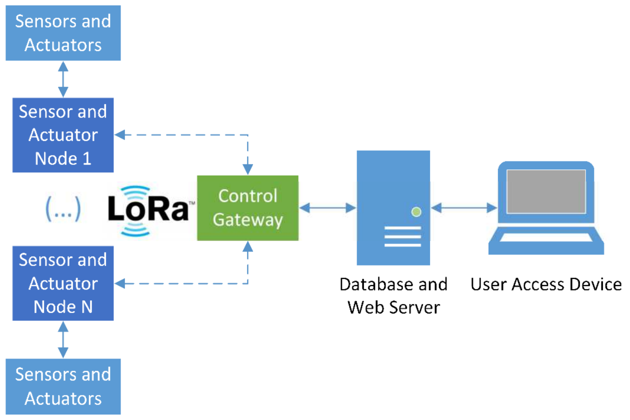


Figure 1. Overview of Smart Farming Connectivity Architecture

The choice of a two-stage system (control gateway and nodes) instead of a single device handling everything or multiple devices directly connected to the internet, has been made: 1) to allow for any size of the deployment, where there may be a need for only a few sensors and actuators or from tens to hundreds of them; 2) to reduce the installation cost in medium to large size deployments by allowing battery or solar operation of the sensor and actuator nodes; 3) to have a centralized decision-making system in case the internet connectivity is unstable or inexistent.

A. Communications Protocol

Farming applications require cost-effective solutions that cover areas with limited access to electricity. Therefore, some of the devices in the systems will need to be battery-powered to keep installation costs down, which will, in turn, require to lower power consumption (shorter processing and transmission times) to obtain a longer battery life. To enable the lowest processing overhead possible while retaining features essential for the optimal function of the system (such as guaranteed delivery and long-range), we built a custom communications protocol on top of the LoRa radio.

The LoRa modulation scheme by Semtech provides a solution for long-range robust communications in scenarios where data rates are not critical [16]. The chirp modulation technique used by LoRa allows for simultaneous transmission in different channels as well as simultaneous co-channel (e.g., same channel) interference-free transmission by the means of different Spreading Factors (SF). It also provides great tolerance to noise, leading to link budgets of up to 156 dB, allowing for long-range communications. All these characteristics make LoRa an effective base physical layer for the communications protocol designed and used in our system design.

As depicted in Figure 2, the sensor and actuator nodes send sensor data to the control gateway, where the control gateway sends back actuator control messages. As the actuator control needs to be performed reliably, an acknowledgment of the receipt is returned by the node to offer the guaranteed delivery.

On the side of the gateway and database communication, the gateway periodically requests the crop information to maintain an updated copy, and when it receives a sensor message or sends an actuator control message, it sends a copy of it to the database, so that the website can show the latest information to the user.

To offer guaranteed delivery of packets on the LoRa communications from the gateway to the sensor and actuator nodes, the protocol includes a blocking acknowledgment mechanism that will not allow processing of any new sensor packets in the gateway until the receipt of the actuator control packet has been acknowledged by the sensor and actuator node.

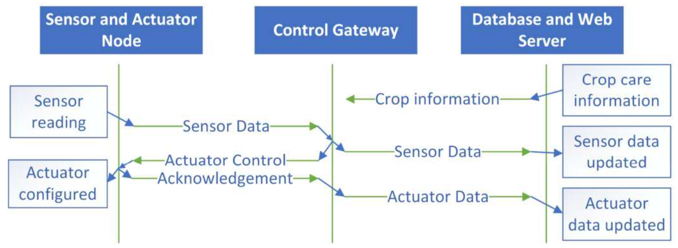


Figure 2. Inter-device Data Flow

However, as the sensor data are tolerant to lost packets given the periodicity with which it is sent, the packets with sensor data transmitted from the sensor and actuator nodes to the gateway are handled by the protocol as the best effort and no assurance is given regarding their delivery and their processing. This is achieved to avoid an excessive number of transmissions in the LoRa channel to reduce collisions and to comply with European regulations [17] regarding the maximum duty cycle in the 868 MHz ISM channel for LoRa (1% of airtime).

The packets sent between the sensor and actuator nodes and the control node only contain data for a sensor or an actuator per message. This design choice enables independent periodic sampling of the sensors to reduce the size of the actuator messages from the Arduino, to reduce the risk of packet collision, and to help to meet the aforementioned European regulations. Moreover, to avoid receiving packets from a node to the gateway in another node, the use of IQ inversion (e.g., LoRa modulation polarity inversion) in one of the channels is highly recommended. In addition, receivers can sometimes detect transmissions from the random noise, causing what is known as phantom packets. We also recommend enabling the use of a Cyclic Redundancy Check (CRC) in the LoRa packet to prevent phantom packets from being processed.

The proposed communications protocol contemplates three types of packets: a sensor data packet, an actuator control packet, and an acknowledgment packet necessary for the guaranteed delivery. For packetizing and depacketizing simplicity and to minimize the packet sizes, all the fields in all three types of packets are 8-bit long.

The structure of the sensor data packet is shown in Figure 3 which consists of three fields: a device ID field with the source device number (as the architecture allows for multiple sensor nodes and actuator nodes connected to a single gateway), a sensor type field with a number indicating the type of sensor being read, and a data field containing the data from the sensor.

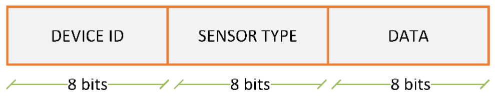


Figure 3. LoRa Sensor Data Packet Structure

The structure of the actuator control packet is shown in Figure 4 which consists of four fields: a device ID field with the destination device (sensor and actuator node) number, a packet number field incremented with every new packet sent by the control gateway, an actuator type field with the actuator that the control gateway wants to change, and a value field that contains the new value for the actuator. The packet number field helps avoid repeating an actuator command when it is resent because the acknowledgment was lost. This is useful when dealing with actuators whose new state depends on both the new command and current state.

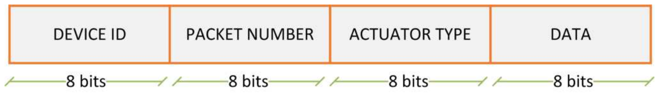


Figure 4. LoRa Actuator Control Packet Structure

Finally, the structure of the acknowledgment packet is shown in Figure 5, which is similar to the sensor data packet (Figure 3), but instead of containing the sensor type and data, it contains the acknowledgment code (0, which is reserved and not used by any sensor), and the packet number that the sensor and actuator node is acknowledging. This way, the control gateway only needs to perform depacketization of one type of packet which, serves two purposes: to send sensor data and to send acknowledgments to control commands.

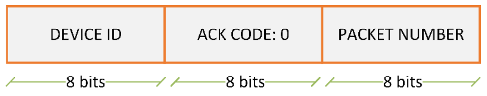


Figure 5. LoRa Acknowledgement Packet Structure

III. System Configuration

The prototype we have built based on the proposed farming automation architecture is composed of three main devices: an Arduino UNO R3 as the low power device for sensor reading and actuator control, a Raspberry Pi 4 Model B as the gateway and system controller, and a computer running a database and web server.

A. *Connectivity*

The Arduino UNO R3 is connected to the Raspberry Pi 4 Model B via LoRa, by using two HopeRF RFM95W modules. The Raspberry Pi 4 Model B is connected to the computer hosting the database and web server via Wi-Fi (through an Access Point). The overall connectivity of the system is shown in Figure 6.

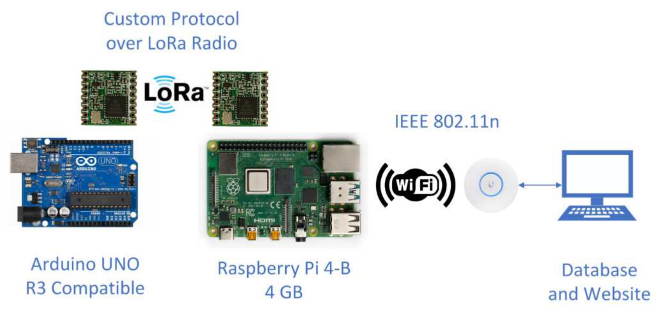


Figure 6. Overview of Prototype System Connectivity

The LoRa radios have been configured, as shown in Table I, to work in the 868 MHz ISM band, with one of the fastest Spreading Factors (SF) available, the fastest Coding Rate (CR), and a large Bandwidth (BW). These parameters are optimal for higher bit rates and lower transmission distances and have been chosen to accommodate operating environments ranging from a small personal farm to an industrial facility where the distance between the different devices is less than 100 meters. The radios have been connected to the Arduino and Raspberry Pi via the power and data pins shown in Table II.

TABLE I. LoRa Radio Configuration Parameters

Parameter	Value
Frequency	868 MHz
Tx Power	17 dBm
Spreading Factor	7
Bandwidth	125 kHz
Coding Rate	4/5
CRC	Yes

TABLE II. Power and Data Requirements for the Devices Connected to the Arduino Node

Name	Operating Voltage	Required Connections
DHT11	5V	Power, one digital pin
Capacitive Soil Moisture Sensor	5V	Power, one analog pin
Vishay VEML6070	5V	Power, I2C
Resistive Rain Sensor	5V	Power, one analog pin
Relays Module	5V	Power, four digital pins
HopeRF RFM95W (LoRa radio)	3.3V	Power, SPI, Reset, Interrupt

The physical transmission speed can be obtained as follows [8]:

$$R_b = SF \frac{CR}{2^{SF}/_{BW}} \quad bps \tag{1}$$

The use of an SF of 7, a CR of 4/5, and a BW of 125,000 Hz provide a physical layer transmission speed of 5,470 bps.

B. *Sensors and Actuators*

For this proof of concept, a series of sensors have been connected to the Arduino to obtain information about the environmental and soil conditions, and a series of actuators have been emulated by connecting LEDs to the actuator control relays. The LEDs show the emulated status of the actuator, providing a better visual understanding of the system. The DHT11 sensor is used to read air humidity and temperature, the VEML6070 to detect the levels of Ultraviolet (UV) radiation, a Printed Circuit Board (PCB)-based resistive rain sensor to detect rain, and a PCB-based capacitive soil moisture sensor to detect the moisture level of the soil. The actuators emulated are a valve to control the irrigation of the soil, a valve to control the moisturization of the air, a heating system to avoid low-temperature freezing, and a motorized cover to avoid excessive UV exposure and to heat escape. The power and data connections used for the sensors and actuator relays connected to the Arduino are shown in Table II.

The Arduino with all the sensors and actuators attached for the prototype can be seen in Figure 7. In this figure, from left to right, we have the LEDs that show the actuator status, the actuators themselves, all the sensors connected in a column, the LoRa communications module, and the Arduino.

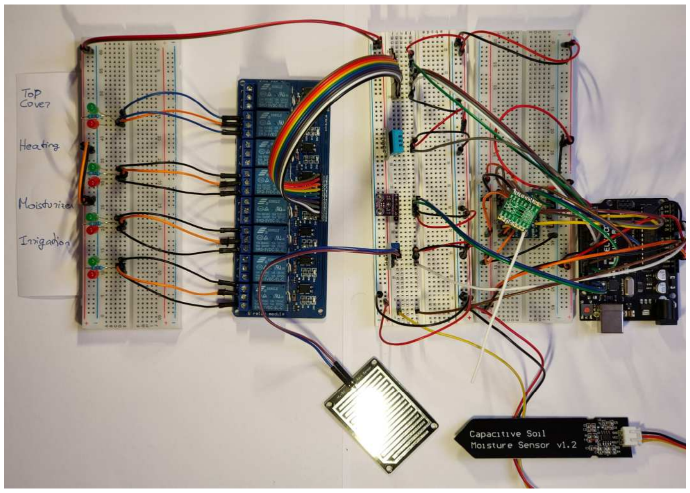


Figure 7. Top View of the Prototype's Arduino Node Assembled with the Communications Module, Sensors, and Actuators

C. Software

To perform all the activities of the system (data collection, communication, actuator control, data storage, and user interaction), we have developed: 1) two main programs, one for the Arduino and one for the Raspberry Pi; 2) a database structure for sensor, actuator and configuration data storage; and 3) a website for user access.

All the code that runs in the Arduino board is written in C++. Two custom libraries have been developed for sensor and actuator control: one for the sampling and calibration of the capacitive soil moisture sensor, and another one for the configuration and testing of the actuator control relays. Multiple manufacturer-provided libraries have been utilized to read the sensors and to interface and communicate via the LoRa radio. A serial interface has been set up via the USB port for debugging purposes.

A program flowchart of the code running on the Arduino is described in Figure 8. On startup, the program configures the sensors, relays, and communications module (RFM95W). This is then followed by an operation loop where sensor data are collected and sent to the Raspberry Pi, and control packets are received and acted upon. The sampling period of each of the sensors is independent and ranges from 5 to 7 seconds in this particular setup.

All the code that runs in the Raspberry Pi board is written in Python (except for external libraries). The Raspberry Pi runs the code that both decides how to control the actuators connected to the Arduino and sends the sensor data and actuator status to the database. The actuator control decisions are made based on the sensor data received from the Arduino and on the crop information retrieved from the database. As shown in the program flowchart in Figure 9, the program starts by configuring the LoRa communications module and the connection to the database and then follows with a program loop. In this loop, the Raspberry Pi obtains periodic updates of the crop information and crop type from the database where the user crop selection is stored. Afterward, it waits until it receives a message from the Arduino containing sensor data to process it. Then, if the algorithm in charge of maintaining optimal conditions for the growth of the crops (in this implementation a threshold-based multi-variable control) determines that the new sensor data requires an actuator change, a packet is sent to the Arduino instructing the changes. As previously mentioned, the delivery of this packet is guaranteed via an acknowledgment mechanism. In this process, the database is also updated with all the sensor data received and with the actuator status changes.

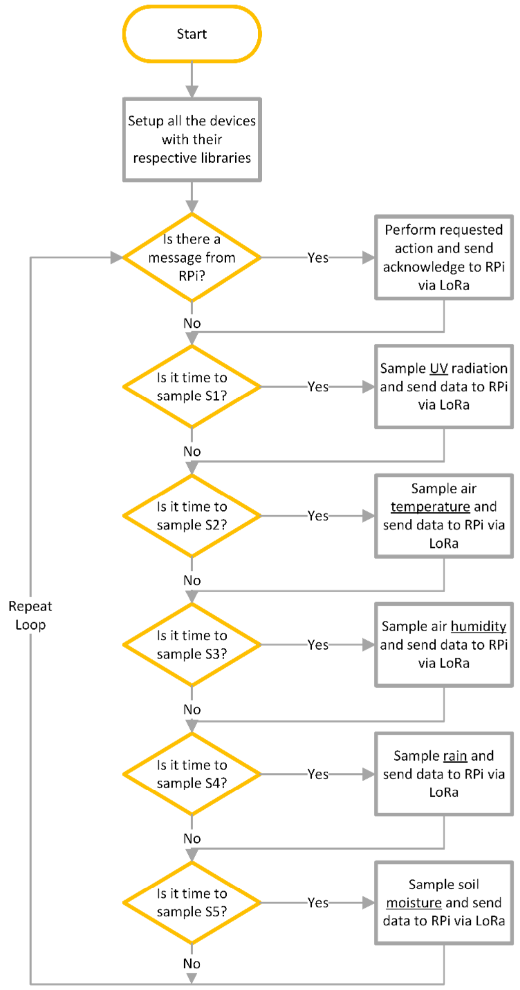

Start
Setup all the devices with their respective libraries
Is there a message from RPi?
Yes
Perform requested action and send acknowledge to RPi via LoRa
No
Is it time to sample S1?
Sample UV radiation and send data to RPi via LoRa
Is it time to sample S2?
Sample air temperature and send data to RPi via LoRa
Repeat Loop
Is it time to sample S3?
Sample air humidity and send data to RPi via LoRa
Is it time to sample S4?
Sample rain and send data to RPi via LoRa
Is it time to sample S5?
Sample soil moisture and send data to RPi via LoRa

Figure 8. Program flowchart of the Prototype's Arduino Node

The database that stores sensor data, actuator status, and crop information is hosted in a MariaDB database engine with a structure of four tables, as can be seen in Figure 10. Each table has been assigned an automatic ID and a series of fields depending on their purpose. The tables are: (a) a sensor data table that stores the readings from the sensors; (b) an actuator data table that stores the actuator status changes instructed by the Raspberry Pi; (c) a crop information table with the ideal soil and environmental conditions for the growth of the crops; and (d) a Raspberry Pi control table that identifies the user-selected crop.

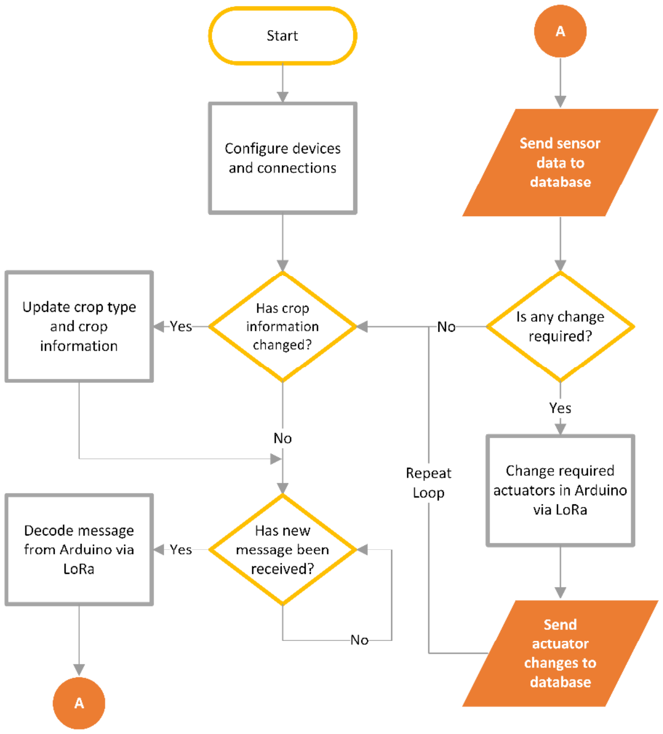
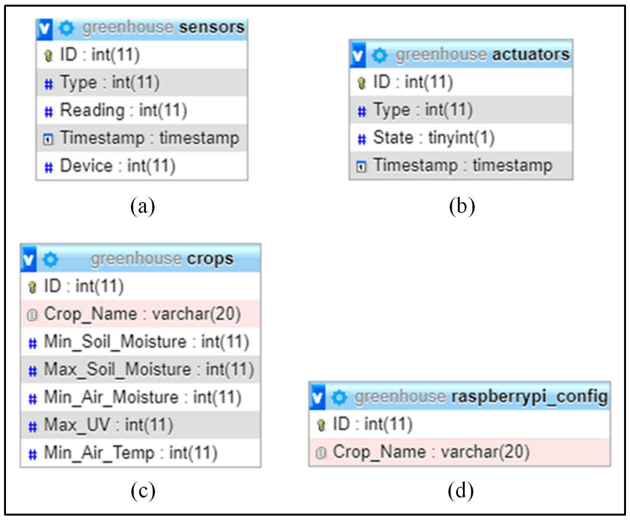

Start
Configure devices and connections
Has crop information changed?
Yes
Update crop type and crop information
No
Has new message been received?
Yes
Decode message from Arduino via LoRa
No
A
A
Send sensor data to database
Is any change required?
No
Yes
Change required actuators in Arduino via LoRa
Send actuator changes to database
Repeat Loop

Figure 9. Program Flowchart of the Prototype's Raspberry Pi Gateway



greenhouse sensors
- ID : int(11)
- Type : int(11)
- Reading : int(11)
- Timestamp : timestamp
- Device : int(11)

(a)

greenhouse actuators
- ID : int(11)
- Type : int(11)
- State : tinyint(1)
- Timestamp : timestamp

(b)

greenhouse crops
- ID : int(11)
- Crop_Name : varchar(20)
- Min_Soil_Moisture : int(11)
- Max_Soil_Moisture : int(11)
- Min_Air_Moisture : int(11)
- Max_UV : int(11)
- Min_Air_Temp : int(11)

(c)

greenhouse raspberrypi_config
- ID : int(11)
- Crop_Name : varchar(20)

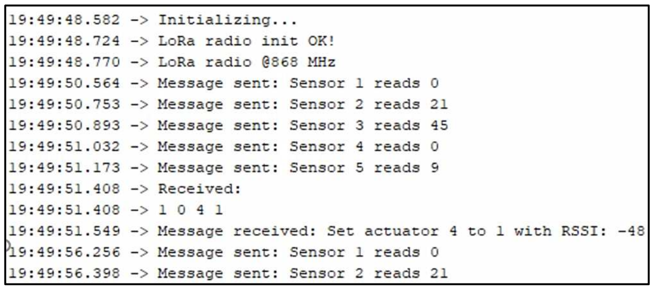
(d)

Figure 10. Prototype System Database Tables, Entries, and Data Types

The last software component of the system is the website provided to the users to both configure the system and check its status. The website code is conformed of HTML for some static content, PHP to allow for dynamically generated content, and CSS to shape the content. The site consists of three web pages: 1) the index with a summary and multiple graphs showing the current environmental and soil conditions, and the status of the actuators; 2) the crop selection page where the user can see and change the currently selected crop and its care characteristics; and 3) the add crop page, where the user may insert a new crop type to the system, with all the related farming information.

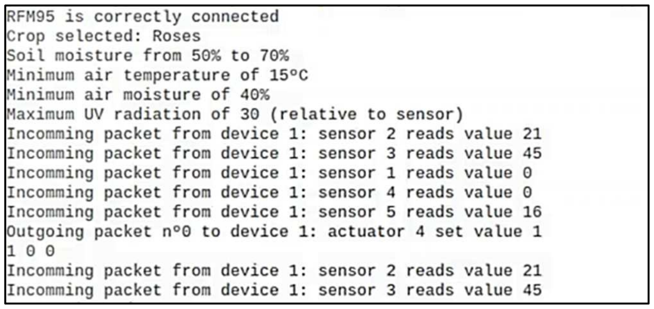
IV. Farming Automation Demonstration

After the assembly of the initial prototype, we proceeded to test its functionality. The testing procedure is divided into five parts: sensor testing, communications testing, controller testing, actuator testing, and website testing. As part of the sensor testing, we calibrated the sensors and obtained sensor readings on the debugging interface of the Arduino, denoted as the messages sent in Figure 11. We followed with the communications testing, where we checked that the sensor information is received by the Raspberry Pi, denoted as the incoming packets in Figure 12. Afterward, during the controller testing, we assessed if the controller was capable of obtaining its configuration from the database (seen on the first lines of Figure 12), sending the sensor data to the database, and sending the actuator control commands to the Arduino and database (seen on the outgoing packet on Figure 12). During the actuator testing, we checked that the control commands were correctly received and executed by the Arduino, denoted as the message received in Figure 11. Finally, during the website testing, the whole system was checked by observing the system response, sensor readings, and actuator changes—in the webpage, the program debugging interfaces, and the relays—, while emulating different environmental and soil conditions in the sensors.



```
19:49:48.582 -> Initializing...
19:49:48.724 -> LoRa radio init OK!
19:49:48.770 -> LoRa radio @868 MHz
19:49:50.564 -> Message sent: Sensor 1 reads 0
19:49:50.753 -> Message sent: Sensor 2 reads 21
19:49:50.893 -> Message sent: Sensor 3 reads 45
19:49:51.032 -> Message sent: Sensor 4 reads 0
19:49:51.173 -> Message sent: Sensor 5 reads 9
19:49:51.408 -> Received:
19:49:51.408 -> 1 0 4 1
19:49:51.549 -> Message received: Set actuator 4 to 1 with RSSI: -48
19:49:56.256 -> Message sent: Sensor 1 reads 0
19:49:56.398 -> Message sent: Sensor 2 reads 21
```

Figure 11. Example of the Arduino Node Serial Debug Logs



```
RFM95 is correctly connected
Crop selected: Roses
Soil moisture from 50% to 70%
Minimum air temperature of 15ºC
Minimum air moisture of 40%
Maximum UV radiation of 30 (relative to sensor)
Incomming packet from device 1: sensor 2 reads value 21
Incomming packet from device 1: sensor 3 reads value 45
Incomming packet from device 1: sensor 1 reads value 0
Incomming packet from device 1: sensor 4 reads value 0
Incomming packet from device 1: sensor 5 reads value 16
Outgoing packet nº0 to device 1: actuator 4 set value 1
1 0 0
Incomming packet from device 1: sensor 2 reads value 21
Incomming packet from device 1: sensor 3 reads value 45
```

Figure 12. Example of the Raspberry Pi Gateway Debug Logs

V. Conclusion

The system based on the proposed farming automation architecture has shown a reliable performance in the testing conducted for the chosen node and gateway configuration (one Arduino as a sensor and actuator node, and one Raspberry Pi as a gateway and system controller). It was able to perform readings of the sensors on the node, communication with the gateway and database, and control of the actuators based on the sensor data and crop care configuration. We intend to expand the prototype to multiple nodes, where the full capabilities of the system can be tested.

The architecture itself can be further improved by adding industry-standard security in the shape of handshaking, cyphering, and hashing techniques. This would lead to Confidentiality, Integrity, and Availability (CIA) of the information and the system. We are also interested in conducting future studies on the different control algorithm choices cited in this paper to analyze their performance on the farming use cases described.

Furthermore, the farming architecture proposed in this paper can be used to build systems with devices (gateway and nodes), sensors, actuators, and control algorithms different from those shown in the prototype. This flexibility is required to allow for the versatility, adaptability, and expandability of the system to cover a wide range of applications and deployment sizes.

References

[1] V. Masson-Delmotte, P. Zhai, H. Pörtner, D. Roberts, J. Skea, P. Shukla and A. Pirani, "Global Warming of 1.5°C.An IPCC Special Report on the impacts of global warming of 1.5°C above pre-industrial levels and related global greenhouse gas emission pathways.," IPCC, 2018.

[2] S. E. Newson, S. Mendes, H. Q. P. Crick, N. K. Dulvy and J. D. Houghton, "Indicators of the impact of climate change on migratory species," Endang Species Res, pp. 101-113, 2009.

[3] A. Laurent and M. Hauschild, "Impacts of NMVOC emissions on human health in European countries for 2000–2010: Use of sector-specific substance profiles," Atmospheric Environment, vol. 85, pp. 247-255, 2014.

[4] J. Zhi and J. Gao, "Carbon Emission of Food Consumption: An Empirical Analysis of China's Residents," in International Conference on Environmental Science and Information Application Technology, Wuhan, 2009.

[5] European Comission, "Global food supply and demand – Consumer trends and trade challenges," EU Agricultural Markets Briefs , 2019.

[6] Food and Agriculture Organization of the United Nations, " The future of food and agriculture – Alternative pathways to 2050," Rome, 2018.

[7] K. Davis, J. Gephart, K. Emery, A. Leach, J. Galloway and P. D'Odorico, "Meeting future food demand with current agricultural resources," Global Environmental Change, vol. 29, pp. 125-132, 2016.

[8] Semtech Corporation, "LoRa™ Modulation Basics," Semtech Corporation, 2015.

[9] LoRa Alliance Technical Committee, LoRaWAN Specification, 1.1 ed., N. Sornin, Ed., 2017.

[10] M. Dursun and S. Ozden, "A wireless application of drip irrigation automation supported by soil moisture sensors," Scientific Research and Essays, vol. 6, no. 7, pp. 1573-1582, 2011.

[11] FarmBot.io, "FarmBot | Open-Source CNC Farming," [Online]. Available: https://farm.bot/. [Accessed 30 01 2021].

[12] R. Eaton and J. Katupitiya, "Autonomous farming: modelling and control of agricultural machinery in a unified framework," International Journal of Intelligent Systems Technologies and Applications.

[13] K. Heong Ang, G. Chong and Y. Li, "PID control system analysis, design, and technology," IEEE Transactions on Control Systems Technology, vol. 13, no. 4, pp. 559-576, 2005.

[14] K. Cheon, J. Kim, M. Hamadache and D. Lee, "On Replacing PID Controller with Deep Learning Controller for DC Motor System," Journal of Automation and Control Engineering, vol. 3, no. 6, 2015.

[15] G. Zhou and J. D. Birdwell, "PID autotuner design using machine learning," in IEEE Symposium on Computer-Aided Control System Design, Napa, 1992.

[16] A. Sánchez-Mompó, "Experimental Analysis of the LPWAN technology LoRa in Smart City and Open Environments," 05 10 2018. [Online]. Available: https://riunet.upv.es/handle/10251/109783. [Accessed 31 01 2021].

[17] European Union - European Telecommunications Standards Institute, "ETSI EN 300 220-1 Final Draft V3.1.1," Europe, 2016.