Revista sítio novo
Instituto Federal do Tocantins

Embedded server to automatic control of wireless transducer network for irrigation based on Internet of things

Tiago da Silva Almeida [1],
Lucas Beraldo Roledo [2],
Rafael Lima de Carvalho [3] e
Warley Gramacho da Silva [4]

Data de submissão: 1º/10/2019. Data de aprovação: 18/12/2019.

Abstract – The growth of the Internet of Things, as well as the optimization needs for agriculture, have opened many opportunities for projects with the integration of these two areas. This paper shows the development of a solution to monitoring crops using the Internet of Things. The sensors and actuators for this precision agriculture design are able to exchange data with an embedded server, which runs on an Intel Edison device. Therefore, an ad-hoc wireless sensor network is established among the devices, using the MQTT protocol. As a result, the monitoring and control of a watering crop project were possible using the proposed solution. In addition, the embedded server allowed the creation of a web interface for the farmer, which can interact with the whole system throughout a web browser.
Keywords: Embedded server. Internet of things. Precision agriculture. Wireless sensor network.

Servidor incorporado para controle automático da rede de transdutores sem fio para irrigação com base na Internet das coisas

Resumo – O crescimento da Internet das Coisas, assim como as necessidades de otimização para fins agrícolas, abriu muitas oportunidades para projetos que integram as duas áreas. Este artigo relata o desenvolvimento de uma solução para o monitoramento de colheitas com a utilização da Internet das Coisas. Os sensores e atuadores para esse desenho de agricultura de precisão foram capazes de trocar dados com um servidor incorporado, que é processado em um dispositivo Intel Edison. Assim, uma rede sem fio ad-hoc fica estabelecida entre os dispositivos, utilizando um protocolo MQTT. Como resultado, o monitoramento e controle de uma colheita irrigada foi possível com a utilização da solução proposta. Além disso, o servidor incorporado permitiu a criação de uma rede de interfaces para o fazendeiro, que pode interagir com todo o sistema por meio de um browser da web.
Palavras-chave: Servidor incorporado. Internet das coisas. Agricultura de precisão. Sensor de rede sem fio.

Introduction

We are currently experiencing a phase of the popularization of projects involving some kind of electronic automation. This is possible due to the low price for integration of circuits with a single encapsulation, also called System-on-Chip or just SoC, bringing simplicity to the

[1] Assistant Professor in Department of Computer Science, Universidade Federal do Tocantins, Palmas, Tocantins, Brazil. tiagoalmeida@uft.edu.br
[2] Graduated in Computer Science, Universidade Federal do Tocantins, Palmas, Tocantins, Brazil. beraldo@uft.edu.br
[3] Adjunct Professor in Department of Computer Science, Universidade Federal do Tocantins, Palmas, Tocantins, Brazil. rafael.lima@uft.edu.br
[4] Adjunct Professor in Department of Computer Science, Universidade Federal do Tocantins, Palmas, Tocantins, Brazil. wgramacho@uft.edu.br

construction of automated systems. In addition, a new area of research has emerged, the Internet of Things (IoT), along with many web development facilities.

According to Lee and Lee (2015), IoT is an emerging market that will generate US$ 14.4 trillion. From an industrial perspective, four industrial sectors generate more than half of this value. These four sectors include the production sector at 27%; retail trade by 11%; information services and finance and insurance by 9%. Other sectors such as wholesaling, health and education fall behind with variations between 1% and 7% (LEE & LEE, 2015). These data show a big growth and popularization of the IoT area.

This rapid popularization also addresses important issues that need to be considered, as well: a) *network overhead* – because of the large number of connected devices; b) *data security* – it is necessary to evaluate if the web services allow the data to be exchanged avoiding the possibilities of data theft; and c) *upgrade* – the speed with which these systems evolve is very large, and more effective and automated ways of upgrading them are needed (KANSAKAR & MUNIR, 2018).

As evidenced by Kansakar and Munir (2018) and Al-Qaseemi *et al*. (2016), another growing problem due to popularization is the lack of standardization of IoT-based systems. The lack of standardization is a problem both with logical systems as physical architectures. Therefore, solutions offered by different brands will not be able to interact with each other.

In the case of residential automation, a term that is increasingly associated with the media as a synonym for modernity, we have air conditioning systems that spend a lot of energy and could have schedules to turn-on or turn-off. This way, its power could be adjusted according to the current temperature to increase the systems' energy efficiency (AL-ALI *et al*., 2017).

According to Guang, Logenthiran and Abidi (2017), IoT is much more than smart homes or connected applications. The focus is on their scalability that allows their implementation in much larger scenarios, such as smart cities. Moreover, according to Sharif, Li and Sharif (2018) and Zhou *et al.* (2016), IoT has enabled real possibilities of connected traffic lights, traffic management, and redirection through connected traffic lights, unified camera system, data processing to monitor pedestrian traffic, etc.

Precision Agriculture can be characterized as the temporal and spatial presence of equipment and sensors that assist in monitoring the field in the presence of fertilizers, pesticides or irrigation (LOZOYA, AGUILAR and MENDOZA, 2016; ABOUZAR, MICHELSON and HAMDI, 2016; ZHOU *et al.*, 2016). Through it, we have the ability to accurately control the use of resources and obtain detailed information about the state of the environment, actions which, in addition to reducing environmental degradation, also increase the producer's profit either through increased production or through precision in the use of inputs.

Within the scope of Precision Agriculture, there are designed technologies that allow the process of data collection and automated production. In recent years, there has been a growth in the development of microcontroller designs with the goal of offering hardware services in mobile applications, whether they are on websites or smartphone applications. The term "Internet of Things" has been rapidly introduced, in which "things" represent any kind of physical object (not software), that can be controlled through the Internet.

This growth can be explained by two key factors: a) the creation of a large number of frameworks for web and mobile application development, such as: Angular.JS, Node.JS and Bootstrap.JS , significantly increasing the ease and productivity of projects; b) the creation of a large number of microcontrollers platforms with open plug-and-play and design modules, such as Arduino, Raspberry Pi, Photon and Intel-based ones. The areas of application for projects involving Internet of Things are wide, including engineering, medicine, industry among others. In this project, we focus on agriculture because there is a great demand for food all over the world.

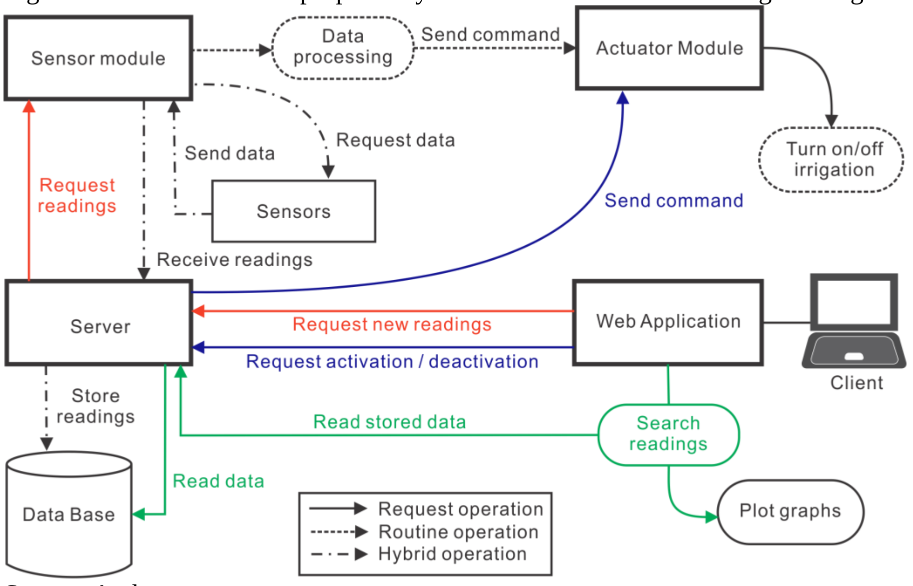
With the popular and emerging Internet of Things, more and more devices are being developed with the ability to interact with each other through the Internet, favoring the implementation and enhancing the resources of these systems in this type of application. For example, it is ideally possible for these systems to communicate directly with climate services, product suppliers and many other organizations that encompass the agricultural production chain. This is why it is possible to see the growing number of jobs linking IoT devices to the development of precision agriculture systems.

Given this scenario, this paper presents an application server running on a SoC platform developed through the MEAN stack (composed of the MongoDB, Express.JS, Angular.JS and Node.JS frameworks, respectively) (TUNG, TOAN and LEE, 2017; CHANIOTIS, KYRIAKOU and TSELIKAS, 2015). The server is responsible for centralizing the sensor module information in a plantation for automatic irrigation control. The modules are responsible for the sensing and performance for irrigation. As the most varied crops need irrigation and have different water demands, the proposed solution uses irrigation profiles for different crops, at different seasons.

Methodology

This section provides a holistic view of the automatic irrigation system in order to have a better understanding of its operation. The project includes a sensor module and an actuator module. There are a brief discuss the operation of each module and how they interact with each other, visually represented in Figure 1.

Figure 1 – Overview of the proposed system with the interaction among the irrigation modules.



Source: Authors.

The *Sensor Module* is responsible for collecting and processing the data exchange throughout the system. It includes temperature sensors, air humidity, soil moisture, soil temperature, leaf moisture and luminosity. Thus, all sensors connected to the node read the data at a scheduled interval, and the obtained information is simultaneously processed to identify the need for irrigation, sent to the server via the Internet, and stored. It is important to highlight that the user can request the *Sensor Module* to do the reading of a specific node value, at any time. In this case, the information is sent to the server only for client consultation, without interfering with the routine of the actuators.

The *Actuator Module,* upon receiving the instructions from the *Sensor Module*, will be responsible for turning the irrigators on and off in the field. This instruction only occurs after processing the data collected in the *Sensor Module*, where the current soil moisture is used to determine the need for water in the plantation, allowing the management and optimization of water resources. It is worth noting that the user can also interact with the *Actuator Module* through the web application when sending requests to turn on or off the modules.

The *Server Module* is the center of the system, being responsible for the data persistence and for establishing the connection between the client and the modules in the field. Upon receiving the data from the sensor nodes, the information is stored in the server database. The server must then be able to return the user's queries and forward any user request.

In the end, there is a web application that allows the user to interact with the system via an internet browser. From there, the client can consult the sensor readings, register irrigation profiles for different crops or seasons, register the modules in the field and access them to perform requests such as re-reading, module activation/deactivation and remote configuration.

The idea of an automatic irrigation system brings with it the idea of a system that helps the user by making some trivial decisions, with no need of user presence. In the case of planting, this means the minimization of the resource is a natural and expected consequence with the usage of this tool. Therefore, the monitoring of soil health takes a lot of time if it is done manually. The implementation of this system allows automated monitoring, with a deeper level of detail in relation to the size of the covered area. Thus, soil maintenance becomes less costly for the producer.

The system consists of sensor/actuator modules, which are made up from microcontroller prototyping, plates with sensors capable of reading useful environmental information around the planting. These modules are a part of a WSN architecture, which includes the sensor nodes and a centralized server (which works as a base station) (SHOUYI *et al.*, 2013). This means that, the user can interact with the nodes using this web server, which has the ability to forward the messages to the proper nodes as well as to take automatic decisions.

Among the possible circuits, we chose the ESP8266 because of the following characteristics: a) minimized physical size; b) built-in Wireless 802.11 features; c) coupled standard antenna; d) low power consumption. Moreover, this last feature can be further optimized using routines and functions preprogrammed in the circuit itself.

A fundamental aspect of the IoT devices is the connectivity feature. The connection ability also depends on network protocols. The employed protocol used in the proposed system is the MQTT (Message Queuing Telemetry Transport). The main reason of such a choice is that MQTT is a lightweight and flexible asynchronous messaging protocol that allows the implementation in hardware of highly restricted device and in networks of limited bandwidth and high latency.

Unlike HTTP (Hypertext Transfer Protocol), which is a one-to-one protocol, MQTT facilitates and reduces the cost of transmitting a message to all devices on the network (common use case in IoT applications) through its model of publication and signature. In this model, two entities are defined: a message broker (the server that receives the published messages and routes them to interested subscribers) and the client (representing both devices and applications).

In the server side, the programming tools include the MEAN stack, which naturally supports the MVC (Model-View-Controller) standard. The server application was programmed using Node.JS + Express.JS, while the client was programmed using the Angular.JS framework.

Because the server will be responsible for mediating the interactions between the web application operated by the user and the sensor/actuator modules in the field, it is necessary to ensure communication and the correct flow of information between them. We define the communication between the server and web application through the HTTP protocol. HTTP allows communication between client and server through a request-response type pattern (called Requests and Texts, respectively). The main methods of sending information in this pattern are HTTP GET and HTTP POST.

The following separation and nomenclature were adopted in order to facilitate the organization and identification of the modules: the μClimate, responsible for reading the data of the environment and decision-making in relation to irrigation and μIrrigation that, as the name itself, will perform the activation/deactivation of the hydraulic system, allowing the precise irrigation of the area.

Developed prototype

The proposed system allows the modularity of the nodes, so that in a section of the plantation (defined by the reach of the sensors), according to the specific need of the area, we can associate to the nodes the pertinent modules the control need for that region.

As we delve deeper into message exchanges in the network, we have the following scenario: each sensor node is responsible for: a) detecting whether a region needs irrigation; b) sending commands directly to the assigned actuator; c) publishing the sensor data via MQTT (enabling the user to have access to the information). Upon this situation, we then have a direct hierarchy relationship in which the server is at the top of the chain of commands. Therefore, the server shall know all the propagated commands (through messages) to the sensor modules and subsequently to the actuator modules associated with them.

The relation of employed sensors and their corresponding parameters are described in Table 1.

Table 1 – Type and respective parameters of each employed sensor.

Sensor Type	Detection Time	Current	Voltage	Precision	Sensing range
Air Temperature	< 5s	200 µA to 500 mA	3 to 5V	2 ºC	0º to 50ºC
Air Humidity	< 5s	200 µA to 500 mA	3 to 5V	±5.0% UR	20% to 90% UR
Soil Temperature	750ms	< 1.5mA	3 to 5.5V	±5 ºC	-55ºC to +125ºC
Soil Humidity	-	< 0.4mA	3 to 5.5V	-	-
LDR Luminosity	120ms	< 75mA	3 to 5V	-	10 to 1000 LUX
Leaf Humidity	-	< 100mA	3.3 to 5V	-	-
Water Flow	-	< 15mA	3.5 to 24V	± 10 %	1 to 30L/min

Source: made by the authors.

The rain sensor or leaf humidity is a sensor whose test piece is sheet-shaped and measures the amount of water that is retained in a leaf after a rain.

Actuators are necessary at that moment when some intervention must be done in the soil to control soil moisture. The actuators used to execute the project are:

- Solenoid Valve;
- Relays.

The user specifies the reading interval; the µClimate module regularly publishes its measure in the topics described in Table 2. In addition, this module can be configured from the list of commands contained in Table 3. These data are exchanged between the two modules µClimate and µIrrigation to allow automatic control of the system. That is, according to the values of the sensors, it is possible to start the irrigation process. This allows the autonomy of the system eliminating the need for human intervention, in such repetitive tasks.

Table 2 – List of topics published by the µClimate module to the application server.

Topic	Message
iam/module/muclimate/[modulename]/reading/datetime	Instant datetime
iam/module/muclimate/[modulename]/reading/luminance	value[0-1023]
iam/module/muclimate/[modulename]/reading/airhumidity	value[%]
iam/module/muclimate/[modulename]/reading/airtemp	value[0 a 50]
iam/module/muclimate/[modulename]/reading/groundtemp	value[-55 a +125]
iam/module/muclimate/[modulename]/reading/groundtemp	value[0-1023]
iam/module/muclimate/[modulename]/reading/leafhumidity	value[0-1023]

Source: made by the authors.

Table 3 – Command list sent from µClimate module to the corresponding µIrrigation module.

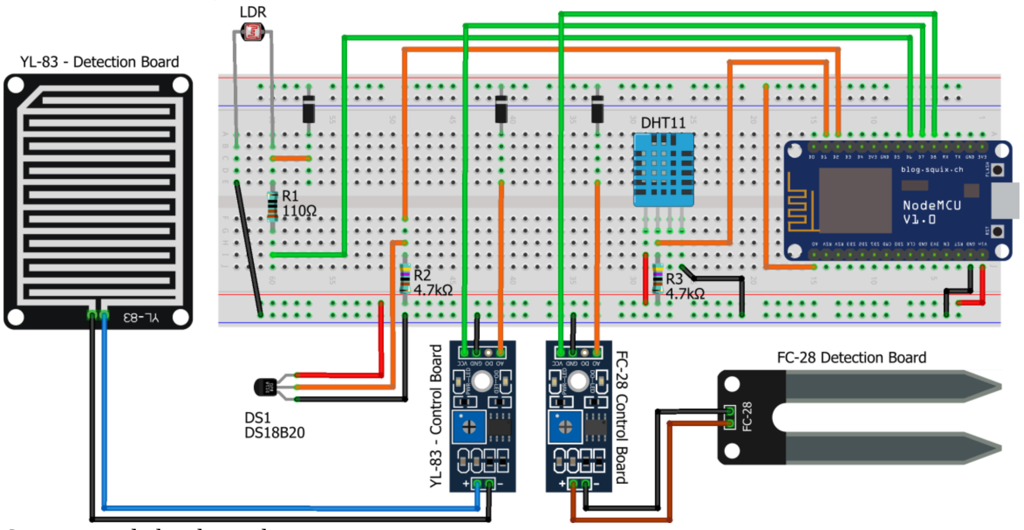
Topic	Message
iam/modules/microclimate/[modulename]/profile/readingsintervals	duration of irrigation in seconds (default: 10)
iam/modules/microclimate/[modulename]/profile/irrigationinterval	duration of irrigation in seconds (default: 10)
iam/modules/microclimate/[modulename]/profile/initialtrigger	value[0-1023] (default: 800)
iam/modules/microclimate/[modulename]/profile/finaltrigger	value[0-1023] (default: 1000)

Source: made by the authors.

The reading values of the luminance sensors, soil moisture and leaf humidity expressed in the Table 2 which vary between 0 and 1023 represents the reading of the analog sensors. The ESP8266 A/D converter has a resolution of 10 bits, hence $2^{10} = 1024$ possible read values. Since the reading of these sensors is based on resistance, the original value read by it represents the value inversely. For example, in the light sensor when there is a high incidence of light on the sensor the reading values are small and when there is little light values they are large, so we implemented a mapping of the values that take the data ranging from 0 to 1023. Then, we converted to a variation of 1023 to 0. The sensors for earth temperature, air temperature and air humidity are digital and therefore their values will vary according to the limitations of each sensor.

The physical implementation (Figure 2) was made in such a way as to make optimum use of the pins available on the ESP8266, freeing the D0 pin, which is used to drive the relay in the µIrrigation module (Figure 3). This measure was taken so that the two connection diagrams were complementary, allowing the merge of the two and resulting in the node µClimate and µIrrigation making the customization of the project.

Figure 2 – Diagram of connections between the components of the µClimate module. All sensors are connected to the same module.



Source: made by the authors.

Throughout the implementation, we found that the ESP8266 has only one pin that is capable of doing analog readings, making it necessary to use a multiplexer so that it is possible to connect several analog sensors to this analog pin, which on the board is marked A0. In the absence of a multiplexer IC, a multiplexer circuit was implemented with the use of diodes.

The multiplexer circuit, which can further analyzed in Figure 2, consists of alternating the activation of the sensors through the green connections and collecting their data through the orange connections. To this end, we connect the pins GND of all sensors to the GND pin in the ESP8266, then we connect the VCC pin of each sensor to a different digital pin in ESP8266 (in this case the pins are D6, D7, and D8), then connecting the data outputs of all the sensors to the pin A0. After that procedure, we can make the activation of the sensors by alternately sending HIGH signal on the digital pins according to the analog sensors connected to them, according to the diagram, pins D6, D7, and D8 activate the Light sensors, Leaf Humidity and Soil Humidity sensors, respectively. Finally, to avoid leakage of current in undesired directions in the circuit, diodes were used at the output of each of the sensors.

The µIrrigation module is the field actuator. The µIrrigation is connecting to a µClimate module of a region and, within the programmed routine; it is going to receive from that module the instruction of how to operate. Thus, when a new read is processed, the µClimate assigned to this module will send a power on/off command according to the region's water requirement and the module µIrrigation will activate/deactivate the hydraulic system.

First, the sensor data is stored locally for use. Compounded by a SoC ESP8266 connected to a relay, it connects to the MQTT broker and awaits instructions, which are detailed in Table 4.

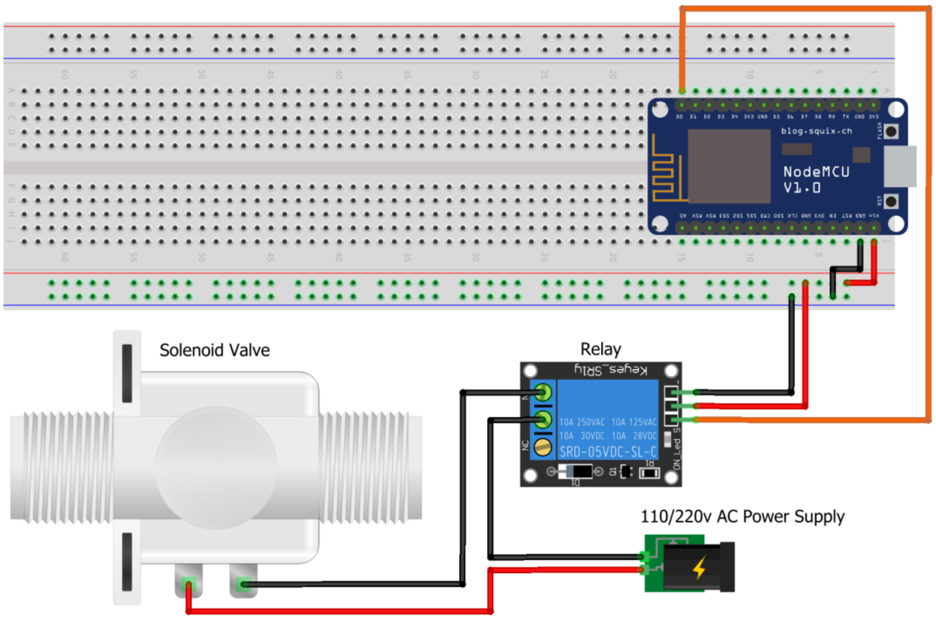
Table 4 – List of commands accepted by µIrrigation module.

Topic	Message
iam/module/microirrigation/[modulоname]/doirrigate	Turn on/off
iam/module/microirrigation/[modulename]/irrigationinterval	Duration of irrigation in seconds (default: 10)

Source: made by the authors.

The layout of your physical assembly is illustrated in Figure 3. The pin D0, by default, is the activation pin of the relay for solenoid valve actuation. The solenoid valve has external 110 / 220V AC power.

Figure 3 – Diagram of µIrrigation module connections. The solenoid valve to turn on/off irrigation requires an external power supply in the prototype. Different kind of valve can be used to require the same power supply as the microcontroller.



Source: made by the authors.

The Table 5 represents the data model created to store the reading data in the database, defining the pattern where the sensor information is saved on the server. The Table 6 represents the data model created to store user-registered culture profiles, serving as a configuration pattern for routine variables of the field sensing modules. Finally, the Table 7 represents the data model of the registered modules, giving the server reference as to the use of the modules in the field.

Table 5 – The abstract data model to store sensor readings collection in the database. This collection is associated with crop profiles.

```
//Document for the reading
{
_id: <ObjectID> //ID for the object
idLeitura: <Integer> //auto-increment reading id
sensor: <String>, //Name of the Sensor Module
groundTemp: <Integer>, //Ground temperature around the corresponding sensor
groundHumidity: <Integer>, //Ground humidity value around the corresponding
sensor
luminance: <Integer>, //Luminance degree at the sensor's position
leafHumidity: <Integer>, //Leaf humidity value read by the sensor
airTemp: <Number>, //Air temperature around the corresponding sensor
airHumidity: <Number>, //Air humidity read by the sensor
readingDatetime: <Datetime> //Reading Datetime
}
```

Source: made by the authors.

Table 6 – Abstract data model to store crop profiles collection in database.

```
//Document for the Profile
{
_id: <ObjectID> //ID assigned automatically in order to identify the
document
name: <String>, //Sensor Module Name
currentProfile: <String>, //Current profile in use by the module
location: <String>, //Coordinates to the module's physical location
status: <String> // Is the module active or inactive?
}
```

Source: made by the authors.

Table 7 – The abstract data model to store actuator step collection in the database. This collection is associated with crop profiles.

```
//Document for the Profile
{
_id: <ObjectID> //ID assigned automatically in order to identify the object
idProfile: <Numero> //ID assigned automatically in order to identify the
associated Profile name: <String>, //Profile name for a specific
crop/season
readingInterval: <Integer>, //time interval in minutes to execute the
scheduled reading
irrigationInterval: <Integer>, //time interval in minutes to check for
irrigation need.
irrigationTrigger: <Integer>, // minimal threshold value used to trigger
the irrigation actuator
acceptableHumidity: <Numero> //minimal threshold value to deactivate the
irrigation actuator
}
```

Source: made by the authors.

It is worth mentioning that these are generic models designed for the scope of the desired solution, and their attributes can easily be modified, allowing new attributes to be inserted according to the specific planting needs and availability of new types of sensors.

The web application is developed in the simplest possible way, prioritizing access to the system resources and its functionalities for the solution proposed, not being presented as a final product. Therefore, we start with a kind of menu with the main features of the server: See Readings, Manage Profiles and Manage Modules. For read-only queries, the application offers a page that lists all saved readings in the server database, offering the option to access each of these readings to view its detailed information and the option to generate graphs based on the registered data.

Given the need to show off information from readings, we implemented a Google Charts module called Angular Google Charts, available for installation by NPM of Node.JS, making it easy to observe information by generating graphs with the server data. With Google Charts, you can create interactive charts of different types like area charts, periods, bars, and more for browsers and mobile devices.

For this solution, we prefer to use a dynamic area chart that displays the values of the readings based on the sensor type. There are many possibilities of representing the data of the server, so as a generic example we decided to display the average of the readings throughout the day on a certain date chosen by the user, as seen in Figure 4, 5 and 6.

For profile management, we have a page similar to readings, which lists the profiles registered in the system with a link to your information, and the option to register a new profile. Thus, according to necessity, the user can add several profiles of different cultures and use them later in the configuration of new sensors modules integrated to the system.

For module management, we also created a table listing pages of the registered modules and the new module register option. This allows to, whenever a new sensor module is inserted into the sensor network, it is added to the system with a profile of cultivation. Therefore, the user has access to the information of each module registered, being able to manage its actuation status and the current profile. When accessing the link of a specific module, the user is able to interact with it remotely, thus being able to request a reading of this sensor at any time, send a request for the module to turn on/off (based on its status). Most importantly, whenever the module profile changes, you can reconfigure its operation variables without the need to reprogram it manually by publishing the new profile information to the topic that the device subscribes through the MQTT protocol.

In short, the developed application allows the client to access and view the server data and interact with the devices at the field. However, the connection to the CloudMQTT broker exists only on the server side, and it is necessary to pass on the server the requests made by the user so that they can be published in their respective topics. For this solution, we decided to implement the Socket.IO library on the server.

Socket.IO enables real-time, event-based, two-way communication that works across platforms, browsers, and devices, with a focus on reliability and speed, and is widely used in instant messaging applications. It consists of two parts: a server that integrates the HTTP server Node.JS and a client library that loads on the browser side. This library can be obtained through the Node.JS Package Manager, the NPM.

Results and discussion

For server testing purposes, as a solution for an automatic irrigation system, the overall system operation was simulated using the Intel Edison application server and an ESP8266 sensor/actuator module, with inserted sensors into a vase with a portion of black earth, both connected on the internet in a controlled environment. The sensor module includes temperature and soil moisture, temperature and humidity environment, light and leaf humidity.

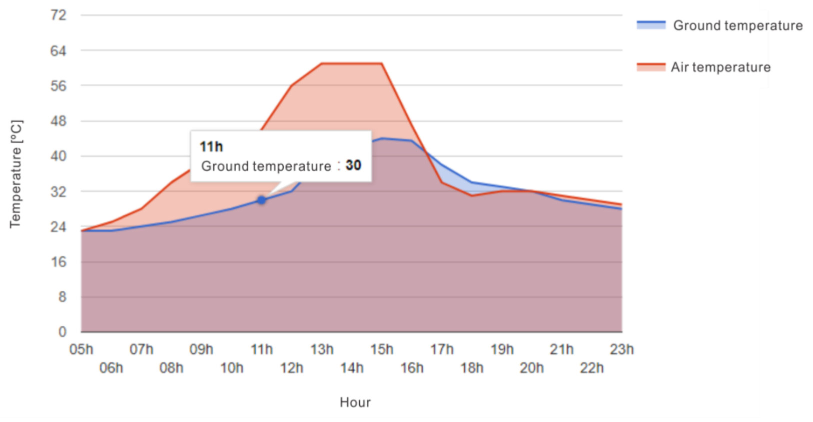
Initially, the interactions between the server requests and the module in the field were tested. In this regard, access to the Intel Edison application via the browser was done and the user functions that interact with the module (request read, request activation/deactivation and reconfigure) were performed, passing requests (and pertinent data) in real time to the server through of the socket. According to the request, the server is able to publish the data in a specific CloudMQTT topic and from it, the ESP8266 module can respond according to the signed topic. It was possible to check if the message flow between server and module was working correctly because when accessing CloudMQTT in the browser, we have access to the history of messages obtained by the broker in each topic.

Another important aspect of the automatic irrigation system is the persistence of readings. For this purpose, the ESP8266 module was allowed to execute its data collection routine and send it to the server on November 1st, 2017 (started at 5:00 p.m. and finished at 11:30 p.m.). In this process the inverse of user requests is happening: it is now the ESP module that publishes messages in the CloudMQTT read topics which passes this data to our subscribing server. Thus, the sensor data are inserted into its corresponding attribute of a new Read-type object until all values are filled and then the read is saved to the database.

An important discussion regarding the reliability of data transmission emerges in this issue. Since the tests always took place in a controlled environment, there was no case of a fall in the connection and loss of information, but it is well known that if it is a precision agriculture system, there may be cases of use in rural areas where the signal is precarious. Thus, there is interference in the WiFi signal, which raises the opportunity to deepen the subject in future studies on network protocols and solutions in cases of connection failure more frequent.

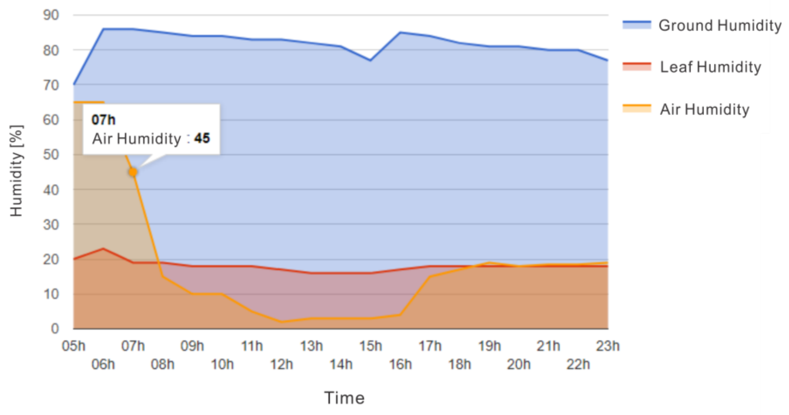
It is important to highlight the server role in monitoring and managing the sensor modules. In a real use case in which sensor modules can be inserted and scattered in distant agricultural fields, the server ability to manipulate and even configure the modules operating profile remotely by the browser is of great relevance. In addition, visualizing the data from the readings through graphics allows a better understanding of the modules’ behavior in a given region. In Figures 4 and 5 we see, respectively, a general view of the variation of temperature and humidity in the region of the sensor module. In Figure 6, we present the light readings in a controlled environment when the data collection.

Figure 4 – Readings graph of ground and air temperature obtained by the server application.



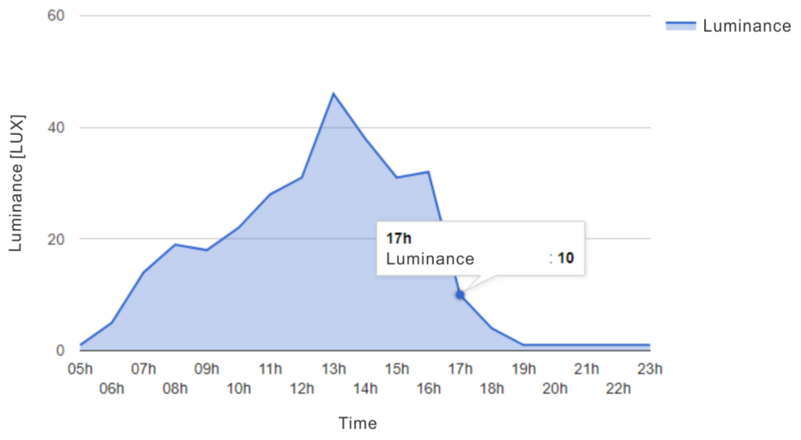
Source: made by the authors.

Figure 5 – Readings graph of ground, air and leaf humidity obtained by the server application.



Source: made by the authors.

Figure 6 – Luminosity graph obtained by the server application.



Source: made by the authors.

We observed that Intel Edison responded well to tests performed on the server. In applications that demand thousands of concurrent connections and requests, it is possible to hit the processing bottleneck quickly. However, since these types of farming systems are usually private (accessed by only one person or team that monitors the plantation) we believe that the device's hardware is sufficient to meet the requirements. Therefore, it dismisses the investment in robust equipment of high financial expense and evidencing the cost versus benefit of embedding the application in a device SoC, as done with Intel Edison. In addition, the high flexibility of the server developed with the MEAN stack allows the easy insertion of new modules in the sensor wireless network, ensuring that the system is scalable in cases of agricultural expansion.

Conclusion

The advance in the construction of embedded systems and the creation of more efficient web services allowed and leveraged the growth of the Internet of Things area. One of the main applications of IoT technology is precision agriculture. This branch aims at the development of state-of-the-art equipment for the efficient management of agribusiness, regardless of which segment it is in. This is the focus of this paper.

Therefore, a solution was developed for precision agriculture for the water control of a given crop. The water control used in irrigation has been chosen to be one of the most serious and fundamental problems of modern agriculture. The proposed system was divided into three modules: one for sensing, one for irrigation and the web server.

In the sensing module, several types of sensors were used, which would not all be effectively needed for automatic irrigation control, however, the generated data provides more information about the climate for decision making by the producer.

At the same time, all the information is sent to the Internet, which facilitates information retrieval for later reuse, sharing, and treatment. After sending to the broker on the internet, the data should be sent to an application server for the effective use. From the server, it is possible to develop mobile and/or hybrid applications for data manipulation.

The tests were carried out in a controlled environment without the presence of plants. However, their functioning could be observed and demonstrated to be very adequate, which allows concluding that the proposed system model can be used without configuration difficulties in quite different cultures and on different scales.

However, we emphasize that due to errors in the temperature and humidity sensor (DHT11), it would be prudent to use sensors that are more robust in future endeavors. The change of soil moisture sensor would also be prudent. It is a resistive sensor and the oxidation of the material is inevitable, at most delayed. Therefore, with these additions and changes the solution costs would be higher, which initial is quite low. A major development concern was keeping project low costs.

References

ABOUZAR, P., MICHELSON, D. G. and HAMDI, M. **RSSI-based distributed self-localization for wireless sensor networks used in precision agriculture**, IEEE Transactions on Wireless Communications, vol. 15, no. 10, p. 6638–6650, 2016.

AL-ALI, A. R. *et al*. **A smart home energy management system using IoT and big data analytics approach**, IEEE Transactions on Consumer Electronics, vol. 63, no. 4, p. 426–434, 2017.

AL-QASEEMI, S. A. *et al*. **IoT architecture challenges and issues: Lack of standardization**, 2016, San Francisco, United States. Future Technologies Conference (FTC), p. 731–738, 2016.

CHANIOTIS, I. K., KYRIAKOU, K. D. and TSELIKAS, N. D. **Is Node.JS a viable option for building modern web applications? A performance evaluation study**, Computing, vol. 97, no. 10, p. 1023–1044, 2015.

GUANG, N. L. L., LOGENTHIRAN, T. and ABIDI, K. **Application of internet of things (IoT) for home energy management**, 2017, Bangalore, India. IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), p. 1–6, 2017.

KANSAKAR, P. and MUNIR, A. **Selecting microarchitecture configuration of processors for internet of things (IoT)**, IEEE Transactions on Emerging Topics in Computing, p. 1–12, 2018.

LEE, I. and LEE, K., **The internet of things (IoT): Applications, investments, and challenges for enterprises**, Business Horizons, vol. 58, no. 4, p. 431 – 440, 2015.

LOZOYA, C., AGUILAR, A. and MENDOZA, C. **Service oriented design approach for a precision agriculture datalogger**, IEEE Latin America Transactions, vol. 14, no. 4, p. 1683–1688, 2016.

SHARIF, A., LI, J. P. and SHARIF, M. I. **Internet of things network cognition and traffic management system**, Cluster Computing, Jan., 2018, p. 1– 9, 2018.

SHOUYI, Y. *et al.* **Design of wireless multi-media sensor network for precision agriculture**, China Communications, vol. 10, no. 2, p. 71–88, 2013.

TUNG, D. M., TOAN, N. V. and LEE, J. **Exploring the current consumption of an Intel Edison module for IoT applications**, 2017, Torino, Italy, IEEE International Instrumentation and Measurement Technology Conference (I2MTC), p. 1–6, 2017.

ZHOU, L. *et al.* **ROSCC: An efficient remote sensing observation-sharing method based on cloud computing for soil moisture mapping in precision agriculture**, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 9, no. 12, p. 5588–5598, 2016.