Development and implementation of a smart greenhouse

Donadio, Maximiliano C.[1]; Garcia-Martinez, Nicolas[1]; Vivas, Luis[1]; Britos, Paola V. [1]

[1] Universidad Nacional de Rio Negro – Sede Atlántica – Río Negro - Argentina
maxdonadio@hotmail.com; {ngarciam, lvivas, pbritos}@unrn.edu.ar

Abstract. A smart greenhouse was developed at Laboratorio de Informatica Aplicada with the department of Agronomic Engineering at the University of Rio Negro, it must provide real-time measurements of parameters such as humidity, temperature and luminosity; it also must allow manual and automatic control for actuators such as heaters, sprinklers and fans based on user input. In order to fulfill the aforementioned requirements the following actions were performed: (a) Design and implementation of a webpage to communicate with and control the greenhouse and, (b) Development: at first a model using Object Oriented Programming was implemented in an Arduino Mega board equipped with an ethernet shield; posteriorly, given that Arduino could not fulfill the necessary tasks, it was decided to develop a second prototype using a Raspberry Pi 2 Model B+ board along with completely new software programmed in Python 3.

Keywords: Python, Actuators, Sensors, Robotics, Greenhouse.

1. Introduction

The need for a smart greenhouse arose from the difficulties concerning research that were presented by the department of Agronomical Engineering at the university, monitoring a greenhouse in real time can be a difficult task which may complicate research and crop production; which is the reason an effective and efficient tool was required.

At Laboratorio de Informática Aplicada (LIA) a project to develop a smart greenhouse was started, several requirements were established, among them are temperature, humidity and luminosity control in real time, collecting and storing data for later analysis; as well as activating or deactivating different actuators manually and automatically, changes in actuators such as the aforementioned must be documented as well to provide more information for future research, it is in this way that the user could observe not only the greenhouse's status at any moment in time but also know which factors were present at said moment. [1] [2] [3].

In order to build the first functional prototype, it was decided to utilize an Arduino MEGA board equipped with an ethernet shield, DHT22 sensor, LDR sensor modules with GL55 photoresistors, two 5 relay modules, an electric heater, a water pump and diverse electronic components to provide the greenhouse with energy. The software was designed using Object Oriented Programming (POO), which results efficient to perform the measurements and changes needed, as well as acting as an ethernet client

and server simultaneously in order to send and receive GET and POST requests to communicate with a webpage from which the user can manually control the greenhouse.

The present article is structured in the following way: Section 2 explains how the software was developed, followed by verification and validation of said software in Section 3, finally, Section 4 presents conclusions and future research.

2. Software development

Initially, the software was developed in an Arduino board using C as programming language. However, due to the need of measuring temperature, humidity and luminosity in real time and communicating with the webserver at the same time, it was imperative to utilize parallelism which Arduino cannot support due to architecture limitations. It was for this that for the second prototype, the board was replaced by a Raspberry Pi 2 Model B+ using Python 3 as programming language along with Raspbian as operative system since it is the manufacturer's own OS and is much more efficient in the interpretation and execution of the board's ARM processor. The sensor utilized in both prototypes were DHT22 for temperature and humidity and LDR modules with GL55 photoresistors for luminosity, the actuators did not change between the two prototypes. Details on the implementation of the aforementioned can be found in the following sub-sections.

2.1 Modules

The software design contains 7 modules along with the main module, a bash script, and a module called which was designed and implemented by the author "APIWebServer", details on their implementation and functionality are featured below:

a) Actuador: Contains methods to activate, deactivate, check the actuator's parameters and status, just receives order and does not do anything else on its own than initializing
b) Sensor: Checks and measures values for LDR y DHT22 sensors, as well as confirming if the established limits have been breached, the user can modify the sensor's properties using the webpage. The measurements from the DHT22 sensors are obtained using the module "MyPyDHT" [4], whereas measurements for the LDR sensors are collected in an analog manner since no working libraries to obtain digital measurements for LDR sensors were found.
c) Estación: Defines methods to initialize all of the greenhouse's data and obtains data for all of the sensors, actuators and the greenhouse's own data from the webserver which then utilizes to initialize every sensor and actuator. Methods to search for sensors and actuators using their respective ID were implemented as well.

d) Control: The most important module in the greenhouse, contains an instance of “WebController” which collects measurements periodically, as well as receiving commands to send them to the server and update the parameters for the actuators, sensors and for the greenhouse itself, it is also responsible of receiving and interpreting the user’s commands. Contains a subclass called “SchedPeriodico” which utilizes the “sched” module [5] in a recursive manner so as to queue functions and procedures to be executed every certain interval which at the time is set by the server. This allows automatic sensor and actuator control, which can be close to real-time control since the interval for measurement recollection can be smaller than 1 second.

e) WebServer: Implements the webserver for the greenhouse, it was made using a framework called “Flask” [6]. This class handles all of the POST and GET requests that the server sends. Code example can be found below:

```
1.self.app.add_url_rule ('/<path: path>','manejarReq’,
    view_func=self.manejarReq, methods= ['POST', 'GET'])
```

The sentence indicated that the class will handle POST and GET requests via a method called “manejarReq”. It will listen for requests in every path and every request will be handled by sending a message with certain parameters to an instance of “APIWebServer”.

f) WebController: This class is responsible for the communication between the greenhouse and the webserver and acts as the web client for the greenhouse , it contains an instance of “Control”, as well as methods to obtain the greenhouse’s self IP for which a bash script is utilized, the script is executed using a python module called “subprocess” [7]. In order to perform all web-related activities it utilizes http.client [8] and Requests [9].

g) ProcessController: Defines an object known as “processController” which has only one method that utilizes a module known as “threading” [10], in order to start 2 threads, one that is in charge of all activities regarding the greenhouse’s measurement recollection, actuator control and web client, and another one that allows the greenhouse to act as a webserver in of itself and receives and executes all the commands the user sends from the webpage. Most of the contents of this module can be seen below:

```
1. import threading
2.
3. def inicializarProcesos (self, estacion, servidor):
4.         estacion.inicializar()
5. threadControl=threading.Thread(target=estacion.controlar)
6. threadServer=threading.Thread(target=servidor.runServer)
7.         threadControl.start()
8.         threadServer.start()
```

h) Bash script: Returns the IP for the greenhouse using 4 commands and pipes

```
1.#!/bin/bash
2.ifconfig eth0 | grep "inet " | cut -c 13-28 | tr -d ' n\n'
```

At the moment of writing this paper, it will only succeed in the task if the greenhouse is connected to internet via Ethernet, for future research the script will be improved in order to make it more versatile and integral.

i) APIWebServer: Contains an instance of Control and possesses only 1 method which is in charge of receiving and interpreting orders from the webserver.

j) Main: The main module, declares one instance of WebController, Control, APIWebServer, Estacion and WebServer, followed by the execution of the only method in ProcessController to initiate functions.

2.2 Webserver and webpage

The webserver as well as the webpage are currently hosted at the University's servers, which counts with remote access. The webpage is made in HTML5 and PHP, which allows it to send and receive requests of the types POST (Requests that send information) and GET (Requests that solicitate information) both using JSON format. Said page, has the following functions implemented:

- Displaying sensor data, actuator status, pause the automatic control in the greenhouse, as well as changing parameters in both actuators and sensors and creating new ones as well.
- Information Storage: Data collected by the sensors, as well as any changes in actuators, are both saved in database for every greenhouse in a network. This allows the administration of several greenhouses at once and documents information for multiple greenhouses simultaneously, therefore being able to function no matter the amount or scale of the greenhouses. The information is displayed in the "Reportes" section, which shows a graph with a progression of the temperature of the greenhouse along time, in the "Mediciones" section, the raw data for every measurement can be seen in any period of time. This is achieved by extracting and processing data from the database at the time the user enters the webpage.

3. Verification and validation

Verification and validation for each of the features that the webpage currently has are detailed below:

a) Addition of new sensors and actuators: In order to validate this feature, 3 sensors and 3 actuators were created in the webpage, it was consecutively verified that by connecting the sensors and actuators to their respective pins, activity remained normal with the newly added actuators and sensors. This test validated emission, reception and storage of data as well as adding new

sensors, actuators, the only limitation being the number of pins available on the board (Fig. 1 y 2).

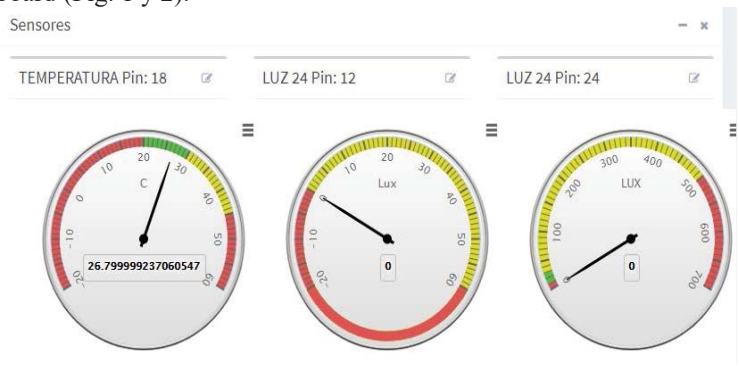

Fig.1. Luminosity and temperature sensors

```
ACCION:CONSULTAR_SENSORES
ARGUMENTOS: {'estacion_id': '2'}
**************************************************

********************************
{'19': 0, '16': 0, '20': 0, '18': 26.799999237060547}
********************************
```

Fig. 2. Reaction to the request of sensor data

b) Ping from the webpage to the greenhouse in order to verify connection status was successful (Fig. 3 y 4).

```
**************************************************
ACCION:PING
ARGUMENTOS: {'estacion_id': '2'}
**************************************************

OK
10.114.80.51 - - [12/Apr/2019 14:53:52] "GET /{"accion":"PING","parametros":{"es
tacion_id":"2"}} HTTP/1.0" 200 -
#################LIMITE#################
```

Fig. 3. The greenhouse's reaction to the connection test

Fig. 4. Webpage receiving confirmation from the greenhouse, thus displaying a success message.

c) Updating parameters for sensors and actuators: By modifying the parameters in the previously added, the commands were executed and were successful, only having to reconnect said components to their new respective pins to remain normal activity. This did not affect performance in any way. (Fig. 5 y 6).

```
10.114.80.51 - - [12/Apr/2019 17:13:06] "GET /{"accion":"ACTUALIZAR_SENSOR","par
ametros":{"pos":null,"pin":"18","limite_superior":"25","actuador_maximo_pos":"2"
,"limite_inferior":"20","actuador_minimo_pos":"2","alerta_maximo":"30","alerta_m
inimo":"20","tipo":"T","lib":"DHT22","id":"18","activo":1,"nuevo":"0"}} HTTP/1.0
" 200 -
```

Fig. 5. Sensor update received by the greenhouse

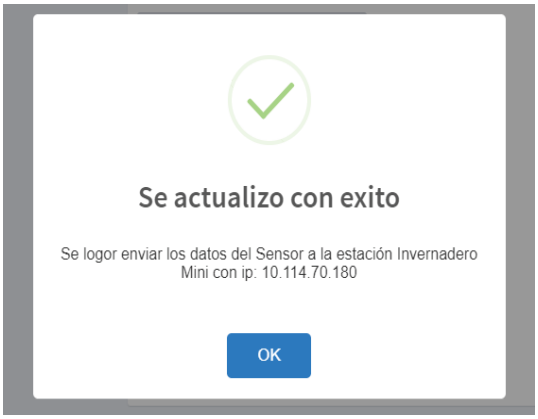

Fig. 6. Success message from the webpage

d) Manual actuator control and automatic control stoppage: This was validated by activating and deactivating each actuator 3 times which was successful. Posteriorly, automatic control was paused which resulted in the greenhouse only taking commands from the user while still performing the expected sensor measurements, therefore being successful (Fig. 7, 8 y 9).

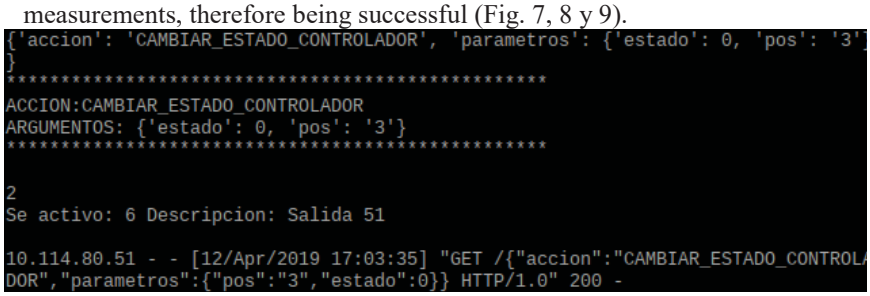

Fig. 7. Reaction to actuator activation and deactivation received

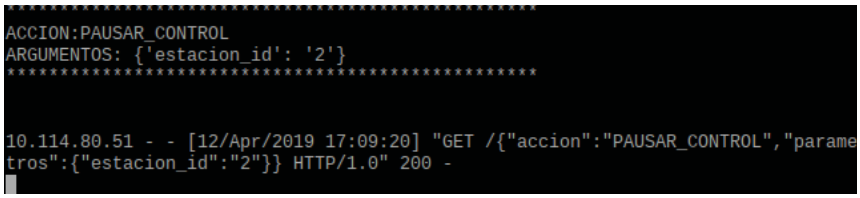

Fig. 8. Automatic function stoppage received

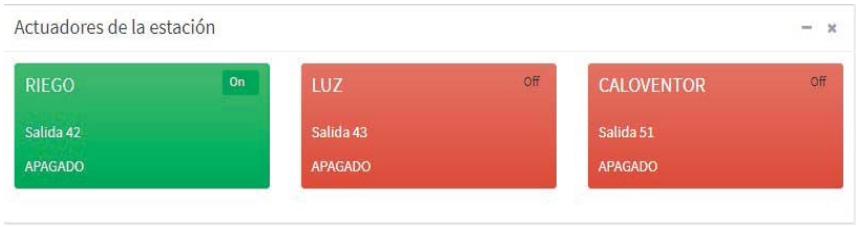

Fig. 9. Actuator control from the webpage

e) Automatic control in case of temperature/luminosity limit breaches: This test was performed by covering the LDR sensors and exhaling near the DHT sensors, as well as positioning a heat source near them. It was verified that in cases were opposite limits were breached in a rapid and consecutive manner (Less than 2 seconds in between) actuators will not cease function, however in case the intervals are greater or equal to 2 seconds, it could be observed that the corresponding actuators for each sensor were activated even in cases where opposite limits were breached consecutively, in this last case doing so by deactivating the currently functioning actuator and activating the correct actuator.

Concerning performance, communication was considered acceptable based on the expert's opinion and response and reaction times were not greater than the intervals defined in the greenhouse's programming.

4. Conclusions and future research

In order to validate the results several hardware and software test were performed over several prototypes. As a result, it was proven that the greenhouse will turn useful for future research in the field of Agronomics.

It is believed that research in domotics in cooperation with agronomics could result in better ways to administrate crops, optimize their growth and development and improving the quality of future research in the field of botany as well. Smart greenhouses such as the one detailed in this paper can be escalated and improved to be later commercialized and possess an impact commercially as well as scientifically.

As future research, it is expected to utilize PID algorithms in order to optimize actuator control for the greenhouse, as well as using MQTT for all real-time communication-related activities. Furthermore, various types of sensors and actuators will be added, such as current sensors, humidity sensors for ground usage, sprinklers and several improvements in terms of hardware used for influencing the environment inside the greenhouse. Concerning the webpage, Angular will be utilized as the main framework due to the efficiency and wide variety of functions that it provides [11] as well as a custom architecture designed around the established requirements(Fig. 10), which will be adapted to any necessary changes.

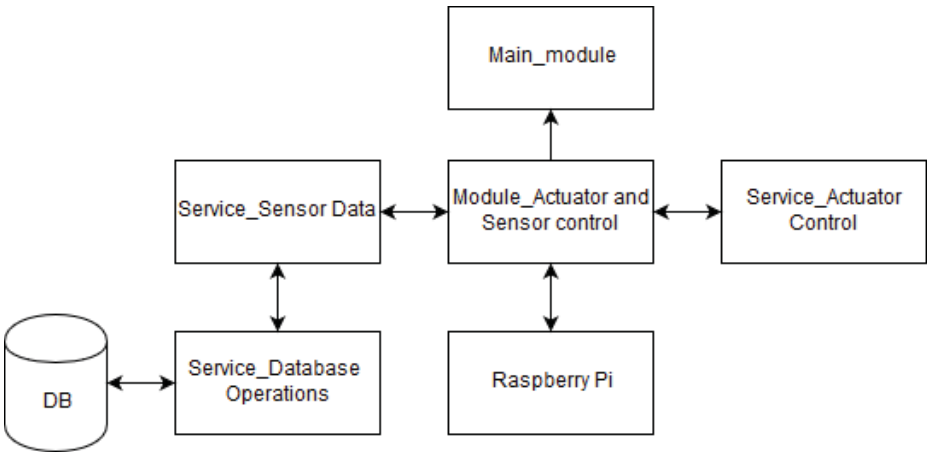

Fig.10. Architecture for the newly designed webpage.

Posteriorly, Data Science algorithms and processes will be utilized [11] for smart data analysis.

Bibliografía

[1] K. Meah, J. Forsyth y J. Moscola, «A Smart Sensor Network for an Automated Urban Greenhouse,» 2019.

[2] R. B. Salikhov y A. A.Zainitdinova, «System of Monitoring and Remote Control of Microclimate in Greenhouses,» 2019.

[3] A. Potapovs, A. Avotins, P. Apse-Apsitis, M. Gorobetz y P. Ceirs, «Continuous Crop Weight Measurement Sensor Calibration Algorithm for Industrial Greenhouse,» 2019.

[4] S. Vettor, «freedom27/MyPyDHT/A Python 3 library for Raspberry Pi to interact with the Humidity & Temperature sensors DHT11, DHT22 and AM2302,» Abril 2019. [En línea]. Available: https://github.com/freedom27/MyPyDHT.

[5] G. van Rossum, «The Python Standard Library,» 2019. [En línea]. Available: https://docs.python.org/3/library/sched.html.

[6] A. Ronacher, «Flask-web development, one drop at a time,» 2019. [En línea]. Available: http://flask.pocoo.org/.

[7] G. van Rossum, «The Python Standard Library,» 2019. [En línea]. Available: https://docs.python.org/3/library/subprocess.html.

[8] G. van Rossum, «The Python Standard Library,» 2019. [En línea]. Available: https://docs.python.org/3/library/http.client.html.

[9] K. Reitz, «Requests: HTTP for Humans,» 2019. [En línea]. Available: http://docs.python-requests.org/en/master/.

[10] G. van Rossum, «The Python Standard Library,» 2019. [En línea]. Available: https://docs.python.org/3/library/threading.html.

[11] P. Britos, *Tesis Doctoral Procesos de Explotación Basados en Sistemas Inteligentes,* La Plata, Buenos Aires: UNLP, 2008.

[12] G. van Rossum, «Python 3.0,» 2019. [En línea]. Available: https://www.python.org/download/releases/3.0/.

[13] E. Upton, «Raspberry Pi 2 Model B+,» 2019. [En línea]. Available: https://www.raspberrypi.org/blog/price-cut-raspberry-pi-model-b-now-only-25/.

[14] M. Banzi, «Arduino,» 2019. [En línea]. Available: https://store.arduino.cc/usa/mega-2560-r3.