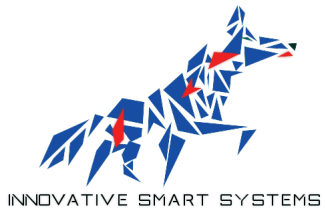
WaTREE

Making Watering Smarter

Authors:
Pierre PRIÉ
Álvaro PASCUAL
Mikita PRAKAPENKA
Axel BAYLE

Tutor:
Jérémie GRISOLIA



January 21, 2019

Abstract

WaTree, Making Watering Smarter

One of the objectives of applications based on the Internet of Things, is to make human life better. Therefore, we have developed WaTree whose goal is to water indoor plants in a smart and autonomous way. This project is divided in two different parts. Firstly, we did a market study and with the results obtained by our plurilingual survey, we were able to design and specify the functionalities of our product. Secondly, we made the technical part, starting by choosing the sensors (ultrasonic, capacitive, photodiode and platinum resistor) to measure plants' conditions, and pumps to water the plants. We implemented the different functions in a Raspberry Pi 3B+ connected to the home's Wi-Fi to send data to the deployed Web Server over oneM2M and recover this data thanks to an Android application developed by us. The first prototype has at least four weeks autonomy and can manage up to three plants fulfilling the market needs. Next steps for this project will be improving the application frontend and optimize the production line.

Keywords : Smart watering, autonomous watering, plants management, indoor watering, IoT, Internet of Things, Raspberry Pi, moisture control , OM2M , IOT , internet of things , custom watering , smart water management system.

Summary

Introduction

WaTree is an innovative smart product, which will make your everyday life smarter and easier. Our team have designed a system in a tree shape, which will keep your plants alive, water them autonomously and being a beautiful object besides all. WaTree have got a smart electronic part containing: soil moisture, light, temperature and water level sensors. Everything being controlled by the Raspberry Pi 3B+ card, which will execute actions depending on user's settings. Those settings will be configured by user through an Android application. The application also allows users to access the current states of the system, by accessing measured data as brightness, temperature and remaining water in the tank.

We were working hard together during the last 3 months, trying to transform our innovative idea to an operational device by the end of the semester.

In this report we will present to you the origin of the project (idea), the market research that has been made, our hardware device and the wireless communication system.

Chapter 1

The project origin

1.1 The problem we want to solve

Nowadays, the transport system is well improved comparing to the past 30 years. Especially, the air transport becomes cheaper, thus the number of plane tourists is significantly increasing. However, we still having much of everyday barriers, which force us to stay at home: as pets or plants for instance. Thus, the major problem that we want to solve is a lack of someone to water your plants. Because it's obvious and repetitive task, we can create small and smart system, which will let you go whatever you want without any worries about the plants.

1.2 Current products on the market

As our project is made in a " startup " mood, we have considered the market study stage of the product like any startup would have done. Thus, after being convinced by our idea we have checked the actual market state thus, our potential competitors.

The smart plant watering market is increasing significantly nowadays. However we didn't find any operational and designed product for the indoor plants. Indeed, there are really successful projects on the autonomous outdoor garden but they doesn't aim the same market. There are also few project for indoor watering but their aim is to take care and growth herbs instead of taking care and watering bigger ornamental plants.

- Véritable, Click & Grow **€ 120 - 150**

  Companies as Véritable (French) and Click & Grow (American) are offering the indoor garden which will autonomously grow small plants, when your put a plant sample inside and fill the water tank.

Figure 1.1: Véritable and Click & Grow indoor gardens

- Continental AWS-10 **€ 50**

  Systems as Continental AWS-10 are efficient in autonomous outside plant watering but are usually not connected and aims another type of customers, those who have a garden for example.

Figure 1.2: Continental AWS-10 system

We could observe a lack of devices, offering the good smart autonomous service for the indoor plants. Through our market study, we will check if customers really feel the needs to have a performing autonomous watering device at home.

1.3 What we provide with WaTree

Our product is a designed system, which could carry up to 3 plants. In each pot, a moisture sensor and a water pipe will be inserted. Globally, a tempereature and a brightness sensors

will provide informations about the environnement. All these components are connected and managed by the small pcb Raspberry Pi 3B+. This pcb will be the central unit of our system thus the smart part. The unit will keep running a main program, which will get moisture, temperature and brightness measurements from the sensors and will autonomously water or not the plants regarding to the plant requirements. Watering will be executed by three pumps, which are connected to the Raspberry and located in the water tank.

The user could monitor and act on the system via an Android application. This application will request the sensors data stored on the server, and so, allows the user to get the system state using just internet connection. Even if the person is far away from home. The application allows as well to set up the Raspberry with the local WiFi connection and configure the system with the right types of plants.

To sum up, we provide a designed device which will carry and water your plants autonomously and an Android application to monitor the system from everywhere.

1.4 Temporal organisation

We scheduled our work on the project from the beginning by making a Gantt Diagram, organizing main development steps. As we explained in the previous parts, this project is a spontaneous student project. So, our first work was to clearly define the project objectives and to translate that objectives into reality by writing a software and hardware architecture for our system. At this point we were able to make a list of all the components needed.

Unfortunately, INSA administration had some organization/communication issues and even if we have transmited them our purchase order at the beginning of October, we only received our order the 30th of November.

Of course, we had to adapt to this situation. That is why, during the waiting for the components delivery, we worked on the market study for our innovative product. During that time, we also review and made some improvements on our software architecture. The final architecture is on the figure 1.3.

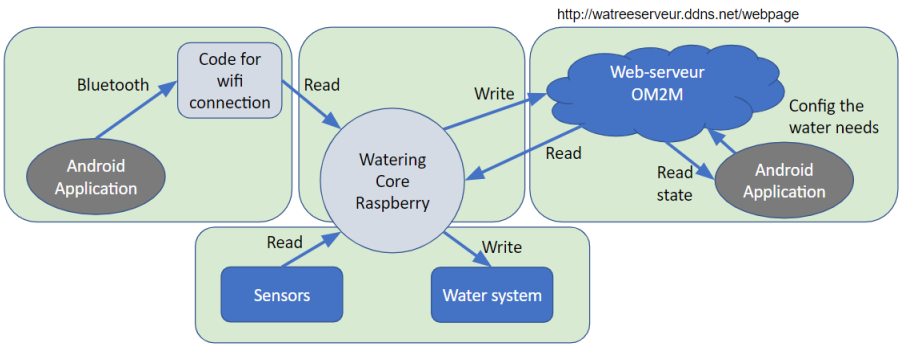


Figure 1.3: The software architecture of the WaTree

After receiving the order, we had to re-plan entirely our time organization because of the

time lost due to the command delay. To do so we divided the work in three parts: device part, android application part and a web server part. To meet the deadline and to be more responsive to issues we have used the Agile method.

One team was first focused on the device part to make the device work on itself, and then start to work on the web server part. At the same time another team was working on the android application part. When these three different parts were done and working, we ended by making them interact to build the final product. Finally, we worked together on the last aspect of the product.

The following picture 1.4 shows the details of our organization in 4 sprints with the Agile method :

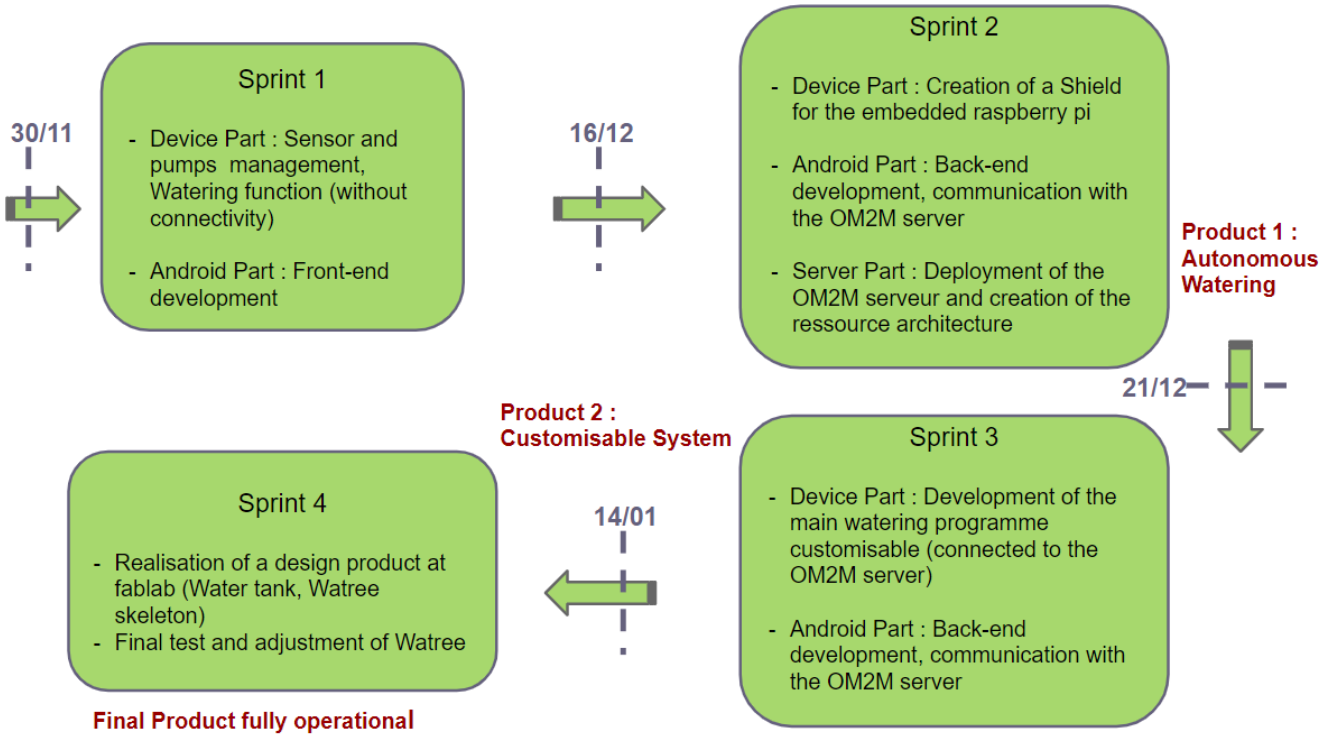


Figure 1.4: Detail of our Agile managment method in 4 sprints

Chapter 2

Our market research

2.1 Method & Goals of the market study

Because of the startup mood of our project and the waiting time for the components delivery we were trying to approve that our idea has a real potential on the market.

First of all, we were analysing the market by looking for existing competitors. This idea was already quite popular in the USA 2 to 4 years ago, and we have some examples of devices which are not existing anymore. However they were offering the same functions as our system in the outdoor. One of those examples was Edyn Garden Sensor Kit

Figure 2.1: Edyn Garden Sensor Kit

This product was well designed but expensive and probably didn't meet its customer.

Thus, we have decided to focus on the indoor field. However we were not sure about the potential of this domain and so the market study was really needed.

To do so, we have created an international survey on 3 languages: Russian, French and Spanish. This survey was well defined to be short, useful for us, interactive and with a good

feedback. We were discussing a lot about the survey structure with the innovation professors to make it smart.

Then we have began the stage of diffusing this survey. It was necessary to choose right diffusion target to get heterogeneous answers. So we were looking for different communities on the social networks mostly, and sharing often this survey to get more and more answers. Another style of market study were requested, as discussions with the plant supermarkets and direct contact with the people in the streets, etc. However we just didn't get enough time to do this, hopefully our survey was quite successful.

2.2 Results & Analysis

Finally the market study, consisting in the online survey brought us these results. We analyse them here.

Figure 2.2: Market study results

We note that we have got 150+ answers from different types of people. 85% are interested by the system, it's proving that our idea could potentially meet its customer on the market.

However, even if people were heterogeneous, we have reached mostly young population, between 22-40 years old. It could means that our survey probably had to be spread a bit better.

The most important question for us was the customers feedback on the cost the system. We have tried to let customers giving us their thoughts instead of giving them the price options. Main result is that they are mostly willing to pay €80 per system, knowing that the cost for us will be around €70. From this we could deduce, that we should sell a lot of these systems for the reasonable price instead of selling few of them for the high price.

This market study wasn't perfect and we have got just few results. However those results are likely the most important for WaTree project, and give us a lot of motivation for the technical part.

Chapter 3

The device

3.1 Choice of the components

Once we had decided the features of the project and the different sensors/actuators needed, we had to find the right components.

The choice of the components was based on two criteria:

- the component must be as cheap as possible to be able to have a final product not expensive. As the market research showed, the customers won't pay a lot for that product and if we want to make profits we need to sell a cheap product with low marges but in high quantity.
- the component must be easy to use with a raspberry, which is the electronic board we choose to use, and meet its specification.

3.1.1 The electronic board

There are a lot of electronic boards on the market that are made for prototyping project like the WaTree, for example Arduino, Raspberry, NXP board ...

We decided to use raspberry because it is more powerful than Arduino and more complex too. Indeed whereas the Arduino are just microcontrollers, Raspberry are real computer with a lot more functionalities. The major one for us was to have a real embedded linux operating system. With the Raspsberry we have access to all the linux command such as Git one's, that were a great advantage for uploading and downloading code from GitHub to the raspberry. Another advantage was the ability to run python code, which is a very famous programming language because of it simplicity and possibilities. In fact, almost everything is codable with python. Finally, we choose the Raspberry 3B+ because that is the last model and the most powerful rapsberry with bluetooth and wifi.

3.1.2 Moisture sensor

There are two main kinds of moisture sensor available on the market:

- the resistive sensors. This kind of sensors are a bit cheaper than the other one, there are based on the measurement of the resistance between two points of the sensor.
- the capacitive sensors. It measures capacity between two electrodes of the sensor. This kind is more precise than the resistive one, furthermore, it is more robust and won't be deteriorated during the measure.

That is why we choose the second type of sensor for our application. After some researches we decided to buy the 426-SEN0193 available on Mouser for almost seven euros.

3.1.3 Water level sensor

For measuring the level of water we had a lot of options.

- Using a strain gauge. That is based on measuring the weight of the tank and deduce the amount of water inside.
- Using a floater. In this case we need a metallic floater and a metallic rail. Depending on the position on the rail , i.e, the resistance or the voltage, the water level can be measure.
- Using a laser sensor. The laser will travel from the top to the bottom of the tank and depending on the time between the sending and the receiving it is easy to calculate the distance with a very high precision.

Finally, and that is what we choose, it is possible to do the same than the laser method with ultrasonic sensor. The advantages of these sensors are their price, and their ease of use. Indeed, the laser is a very thin beam and it is hard to put it to be able to get the echo. Another point is time speed, a laser is a light beam so it is going much faster ($3.10^8 m.s^{-1}$) than the sound($340 m.s^{-1}$) and, by this way, it is harder to develop a software because it has to be very optimise. The HC-SR04 ultrasonic sensor is very popular, so we decided to use it too. On top of that it is quite cheap, only €3.9 on gotronic site.

3.1.4 Pump

The pump were the hardest components to find because of the electronic board choice. In fact we wanted to use the power supply provided by the raspberry to power all the system and, as the Raspberry is a 5V device, the maximum voltage we could use was 5V. This point was the problem, almost all the pumps on the market are 12V power.

After long researches we find the FIT0200 pump on Mouser, for €7.96 that could be powered between 4.5 and 12V. So, we choose this model of pump. But since the raspberry can't deliver a high current we had to use a common emitter circuit, as drawn on the figure 3.1 bellow with a 2N2219A npn transistor. The raspberry is connected to the base pin of the transistor. The common emitter circuit acts as a switch, then, when the base voltage is high the pump is activate and when it get low the pump turn off.

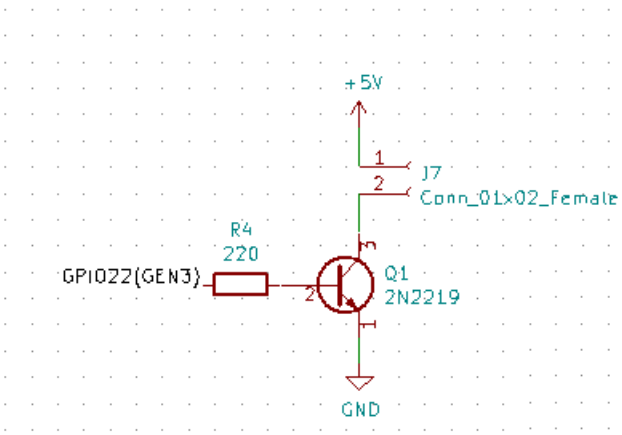


Figure 3.1: The common emitter circuit

3.1.5 Light sensor

There was no big deal and constraint for this sensor. So we choose the facility and took a groove sensor that is cheap and very easy to use. Its only drawback is that it is an analog sensor, just as the moisture sensor, so we had to add an ADC (Analog to Digital Converter). The light sensor we used is the 101020132 from Mouser.

3.1.6 Temperature sensor

This time again, we choose a sensor with groove connector the 101990019 from Lextornic. This one is very interesting because it is waterproof with two meters wire. But it is not a simple sensor which send the value on the GPIO, it uses 1 wire protocol.

This is a protocol of communication such as SPI or I2C, that was not a issue because the Raspberry 3B+ support this protocol on a special GPIO pin.

3.1.7 ADC

Because of the use of analog sensors we had to buy and use ADC to convert the analog value sent by the sensor to a digital one that the raspberry could read.

We found the MCP3008-I/P ADC which provides all we need, indeed there are 8 analog input channels while we were needed only 4 and it could be powered by the raspberry.

3.2 Management of GPIOs

With our several sensors and pumps connected to the raspberry we had to find a solution for the management of the GPIOs. That's why we looked for a python library to do that.

Finally we choose to use PiGPIO because it allows a very easy way to configure the GPIO (input, output . . .) and read their state.

3.2.1 Water level sensor

The water level sensor is a ultrasonic sensor. It has four pins, power supply, ground, echo and trigger. The main pins are the echo and the trigger ones. Indeed, to collect the value we have to send a high level on the trigger pin (for that we use the PiGpio library) then, after 10 microseconds, the pin level is setted to low. Once the trigger phase ends, the echo pin is going to change its value, to high level during a certain amount of time. This time is directly proportional to the distance.

The final step to measure the water level were to compare the distance measure with the references we calibrated, which are the distance when the tank is full and the distance when the tank is empty.

The module we developed for the distance sensor is available on the annexe C.

3.2.2 Temperature sensor

This sensor use a special protocol called 1wire. The GPIO 4 is dedicated for that protocol on the raspberry 3B+.

To read the data of the sensor it is not possible to read the value directly on the pin, we have to read in a specific file and to filter the file to select only the information we want.

The code is available on the annexe D.

3.2.3 Analog sensors

All the analogs sensors are managed the same way. We choose to use a specific library for the ADC, that allowed us to get the values of a sensor by specifying the channel of the ADC connected to the sensor.

Then we convert the value read into the real value. To do that we use function depending on the sensor, for the light one it is just a simple conversion between the data read and the maximum, for the moisture the function is a bit more complex. However this work only if the ADC is connected to the pins dedicated for the SPI protocols on the raspberry.

The code is available on the annexe E.

3.2.4 Pump actuators

The last GPIO we had to handle were the ones connected to pumps. The functionality is exactly the same for the three pumps.

With GPIO library we configure the GPIO in output mode. Then we can set the high level output to turn on the pump and the low level to turn it off.

The code is available on the annexe F.

3.3 The main routine

The main function is an infinite loop that start with the initialization and the set of the references for the water level sensor. A loop is executed each hour and each loop follow the same path.

It starts by getting the value of sensors and sends them to the server. In the particular case of the light sensor it doesn't send the value read but the mean of the 24 last value, that allow to have the mean light during a day.

Once all values are sent on the server, it will get the amount of water in the tank. If the level is above the threshold level then the loop ends.

Otherwise, the program get the different moisture needs on the server. And for each plant it will compare the moisture need and the moisture measured by the sensor. If the difference is higher than 5% it has to water the plant and send the new water level. Right now the watering is just turning on the pump during a certain amount of time that is define in the code and do not depends on criteria.

3.4 Production of a shield for raspberry pi 3B+

At this point we had a working device. We were able to receive data from all our sensors and to control each pumps. However, our electrical mounting was still realized with breadboard which can generated punctual issues like wire disconnection or short circuit.

Therefore, we decided to make our own raspberry shield. For that we used KiCad which is a free open source software. The use of Kicad is divided in two part, the schematics part and the routing part. In the schematics part you draw your electronic diagram, that means how the different components are linked together, which pin is connected to the ground, to the input voltage (3.3V or 5V). For our shield, all our sensors and the pumps are represented by the connectors we used to connect them to the raspberry pi3B+. The picture 3.2 represents our final electronic diagram.

After realizing the electronic diagram and before routing our circuit we must associate a fingerprint to all our component. That means associate a real existing component to all the virtual component in the electronic diagram. Let's take the example of our 4 pins connectors. We specified that the one connected to the CAN will be a basic male connector while the two others are groove connector. This assignment is very important for the routing part in which the shape and the size of the components are considered. To help us select the right real component, 3D views of each of them are disponible with KiCad. For the routing, KiCad already provide us the shape of the shield, we then only have to import our component and link them correctly in the dedicated area. As you can see on the figure 3.3 we realized a two copper layer routing (front in red and back in green) with a ground plan on the front layer.

We had to modify our first routing to meet the reality of the PCB that we can realized at INSA. Indeed, these PCBs are basically two copper layers with an isolated layer in between. That means that the holes we made are not metalized and then if we solder the component on the front layer it may not be connected to the back layer. We also had to consider the shape

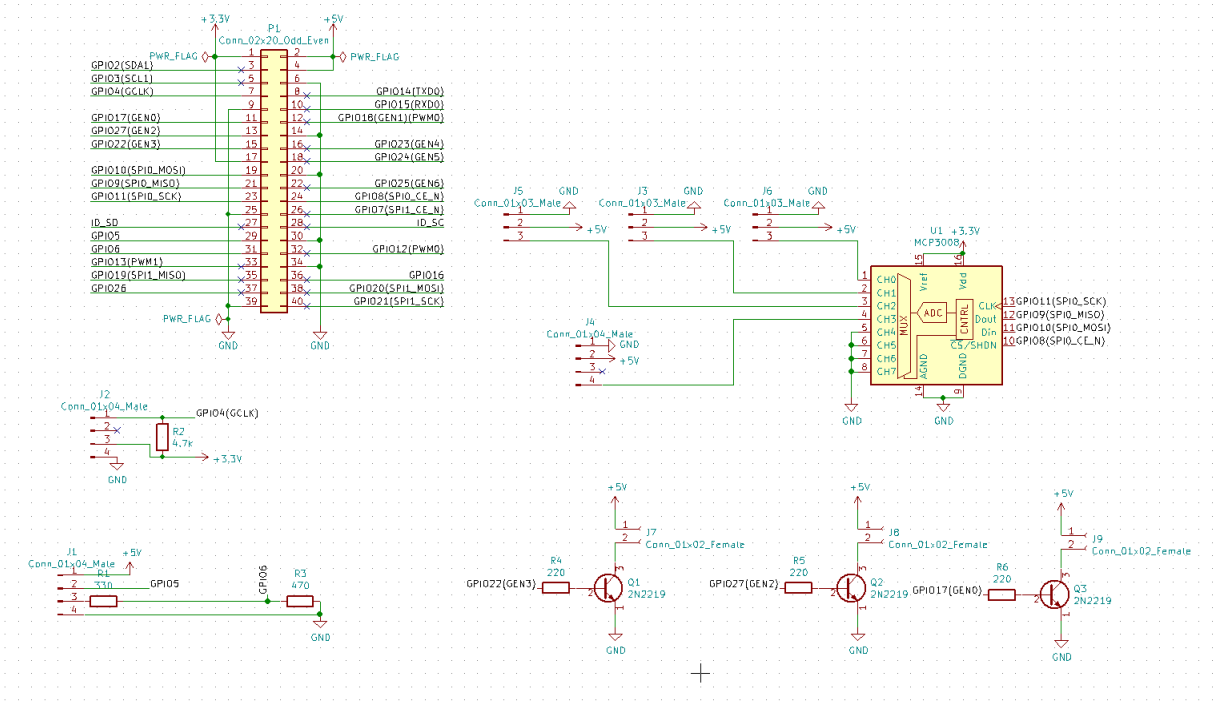


Figure 3.2: Final Electronic diagram of our Watree raspberry shield

of our connectors and their direction. For example, the raspberry connector must be on the back layer (to connect with the raspberry) and because of its plastic encapsulation it can only be soldered from the front layer. For all the other connectors it is the opposite, they must be on the front layer (to be accessible) and soldered from the back layer.

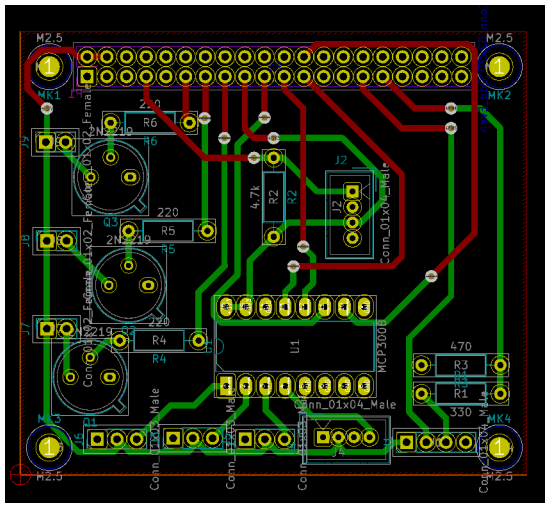


Figure 3.3: Final routing of our Watree raspberry shield

To meet these specifications, we modify our routing to access the raspberry connector

from the front layer (red wire) and all our others component from the back layer (green layer). To switch and connect from one layer the other, we used vias made with iron wire soldered on both sides.

To make the shield, we then printed the routing scheme on transparent sheets to serve as a mask. The blank pcb is then placed in an uv insolation machine with the corresponding masks on the front and back of the blank pcb. So, by puting the PCB after insolation into an acid bath, the acid will attack the copper locally to form the routing pattern. Then all that remains is to drill the holes for the components on the pcb and solder them. This final step is very important and critical and a bad solder can make the pcb unusable. This is why we have been very cautious and checked the validity of our welding as well as the possible short circuit at each step thanks to a multimeter. The figure 3.4 show our final shield mounted on the raspberry.

Figure 3.4: The Watree shield on the raspberry of the embeded system

3.5 Product design

We use the different feedback from the market study to design the first prototype. The aim was to create a all-in-one product that can include the water tank and be the support of the tree plants.

We draw some sketches that we showed to potential customer to find the best one. With their feedbacks we improved it and we finally get the final sketch. Then we modelisated it in 3D, as it showed on figure 3.5 to have a better idea of what it will looks like.

Figure 3.5: 3D model of the first prototype

Once this was validated, we started drawing blueprints. We used the fablab to realize the prototype. The water tank is made on plexiglass and the structure is made on wood.

The hardest part was the joining of the water tank because we had a lot of trouble with water leaks.

The final prototype is showed on the figure 3.6.

3.6 Improvements

The project is a first prototype and, as all prototypes, there are improvements that can be done on both software, hardware and design.

The first improvement can be done on the watering procedure to adapt the amount of water depending on the lack of water and the pot size. One possibility to do that is to add a flowmeter to each pump to measure the amount of water delivered. In addition to those sensors we will had to modify the code, the main routine and the hardware.

Another improvement would be to develop our own card and not only a shield, by doing this we will have a all in one electronic board with only the necessary components, that will allows us to have a cheaper device.

In this prototype there are sleep mode between the loops. That could be a great improvement to put the electronic part in sleep mode and wake up it only the time of the routine and then go to sleep mode for an hour. That will be a power reduction improvement that will lower the use cost of the WaTree.

Finally, on the design part, we could try to improve the design of the product to make it

prettier and find a way to have a better process for the water tank to avoid leaks.

Figure 3.6: Picture of our first implemented prototype

Chapter 4

The Wireless communication

4.1 Wireless functioning of the device

As we mentioned before, we have all our sensors connected to the Raspberry by wire. However, we have developed an important wireless part for our project in order to permit the users the connection to their device. This means that the user would interact with the device using her/his mobile phone no matters where the user would be and moreover, with a network connection to the internet, she/he would be able to monitor and set the device parameters. This is possible thanks to a dedicated Web Server where all the data, parameters and functions needed to run the device would be stored and it will be accessible from our device to manage the watering and sending the data collected by the sensor as well as accessible from out mobile phone. The wireless functioning is based on a network connection to the internet, assuming that almost everyone has a WiFi connecting at their homes and also having a smartphone with the possibility to reach the internet through a data connection or using a WiFi connection, which are deployed all over the world.

4.2 Dedicated Web Server

As we said, we created a web server to store datas like the moisture set for each plant, the mean luminosity over the past 24 hours, the temperature and the remaining autonomy of the water tank. To do so, we deployed the OM2M IoT architecture which is an implementation of the oneM2M standard. The advantage of this technology compares to other one like Thread or Homekit (from Apple) is that it has a standardized interface that allows to communicate with sensors that implement various technology.

OM2M provides a server associated with a database which has a REST architecture, that's means that all datas are considered as resources. You can see on the figure 4.1 the resource tree we deployed for our Watree application. We can see that the "Watree" resource (which is an Application Entity) contains 4 resources (which are Containers): "Temperature", "Niveau_eau", "Plantes_humidite" et "Luminosite". Each of them also contains resources (which are Content Instance), this is on that resources that datas are stored.

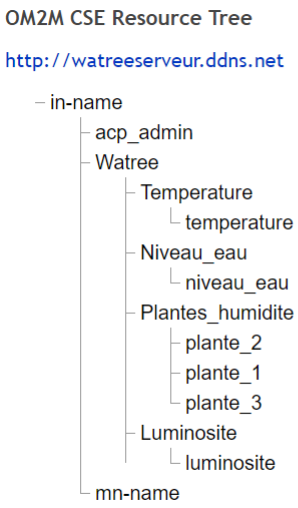


Figure 4.1: Resource tree deployed on the OM2M server for the Watree application

OM2M is based on REST architecture, that means that to interact we will only use 4 different type of HTTP request: POST, GET, PUT and DELETE. In our case, after the deployment of the resource tree we only use POST request to send datas on the server and GET request to recover data from him.

We used a raspberry pi 3B+ to host the server for the Watree application, you can see it on figure4.2. In that way we are sure that the server is always running. To make it accessible from the outside of the local network in which is his deployed, we used the free “no-ip” services which allows to associate a URL to the physical device that host the server (the raspberry here). The URL address of our server is http://watreeserveur.ddns.net.

Figure 4.2: The raspberry pi 3B+ used as a server

4.3 Android application

We have developed an Android application, thanks to the Android Studio IDE, to manage the configuration settings of the different plants and to monitoring the room temperature, brightness and the tank water level. We decided to develop it using three different activities (unlike programming paradigms in which app are launched with a *main()* method, the Android system initiates code with an *Activity instance*), which means that we have three windows (screens), each one with a specific function that permits the user interaction with each function in a friendly way.

We can see the screens aspect in the next figure 4.3 and how the interaction with the different buttons change to another screen and the data requested to the server (if it is the case).

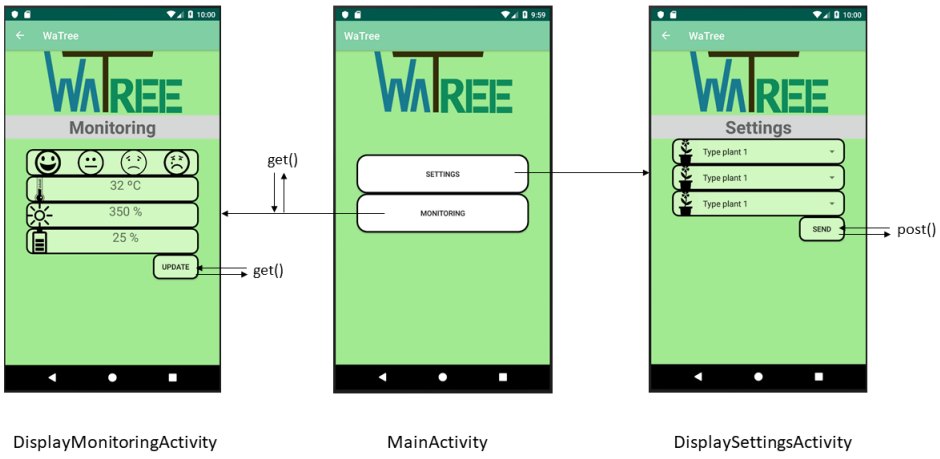


Figure 4.3: Menu, configuration and monitoring part of the application

The *MainActivity* is the one that appears when we open the app. Here we have two buttons, each of them allowing you to change to another screen; to the settings or the monitoring one.

DisplaySettingsActivity is the screen where we can choose what type of plant we have in each pot or saying that we have some place empty so the system will not monitor or water that place. When setting the type of plant at the spinners and clicking at the send button, your app sends this data to the Web Server allowing the system consulting this data and knowing the parameters needed to water your different types of plants. This is made using a post method that creates a new content instance with the amount of humidity that every plant need. Using this type of spinners to choose the type of plant is useful to make scalable the application if in the future we need to increment the number of plants monitored by the system; we will just need to add new spinners and calling the same method post to send the data.

Finally, when you click the monitoring button, a request is made to the server to get the data and show it at the *DisplayMonitoringActivity* where the temperature and luminosity of the room is shown as well as the level of the water tank; this data arrives thanks to a get method that calls the content instant where the data is stored at the server. If we continue at this screen and we want to refresh the data to see if it had change, we can do it clicking at the update button.

4.4 Improvements

This is a first prototype so we have implemented the most important wireless functionalities. However we could made some improvements for the next prototype.

The application could be improved if we add some functionalities such as creating an user account to access the application and made it more secure. This will mean that a web application can be developed to increase the possibilities to the user to access to the data and made changes in the device using her/his user login. Another parameters such as the language of the app could be added.

But the main improvement would be adding a first settings of the device part where we can set all the parameters of our device the first time we use it. This is setting the Raspberry parameters from the application, selecting our WiFi network and password and sending it to the Raspberry allowing it to access our network and the Web Server. A first brainstorming and ideas about this have been made and we decided that the best way of implementing this function is with a bluetooth connection thanks to the bluetooth module available at the Raspberry.

Conclusion

Despite many unpredictable events, we have done our best on this project. Although not all the functionalities we thought about couldn't be implemented, we are very proud of its success. We have learned to manipulate many tools and materials, as well as to better manage our projects over much longer durations than we use to, and finally we have work experience in a business context with its constraints and advantages.

The sharing of knowledge in Hardware design, Software development and network has allowed us to carry this project as far as possible. While we feel that we have wasted time on technical details and timelines, it has allowed us to learn and take a step back from the challenges we faced and the challenges of being an engineer in the world of tomorrow.

We really enjoyed working on this project. It has been an instructor in many capacities and has allowed us to participate in a first project to create an IOT object, on top of that, our approach with market research has put us in the shoes of a start up who wants to understand customers so that it can adapt it solution to the market.

We are confident in the viability of this project and its future prospects, with or without us, the market does exist and more research and analysis on this field could lead to a business plan and a real start up project. And maybe, one day, the release of the WaTree on a crowdfunding platform before being release on the market.

Annexes

Making Watering Smarter

Design and autonomous smart system that will take care of your plants while you are enjoying your life

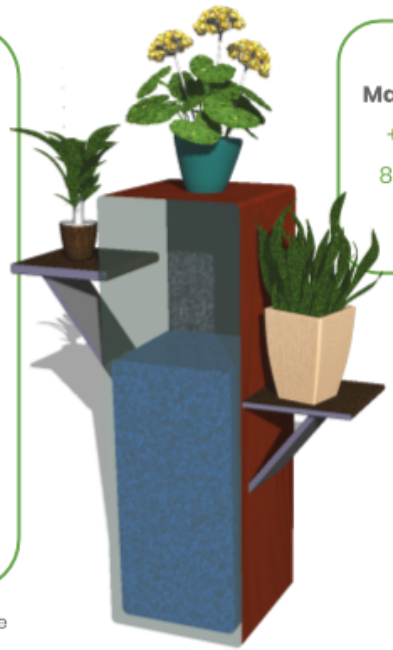
How does it work ?

The System :

- Measure soil moisture, temperature & sunlight
- Water in case of need
- Send data to TTN

The App:

- Track plants' health
- Notify when the water level is to low
- Custom the settings



Where are we today ?

Market Research	Developpment	Roadblocks
+150 answers	Architecture defined	Equipement received late
85% interested people	App almost ready	Time management
	Sensor management	Technical issues

Bayle Axel ; Prakapenka Mikita ; Pascual Alvaro ; Prie Pierre

December 6, 2018

Annexe B: Abstracts

WATREE, Making Watering Smarter

By : Prie Pierre applied physics student specialized in Innovative Smart Systems at Institut Nationale des Sciences Appliquées Toulouse

In the past 10 years, watering system has been improved to be smarter and connected. Unfortunately, this kind of smart indoor watering systems are still not very developed for the home market. Therefore, we decided to create our own indoor smart watering system. We used capacitive soil moisture, sunlight (photodiode) and temperature (platinum resistor) sensors to monitor the health of the plants. We used pumps and plastic pipes for water carrying and an ultrasonic sensor to track the remaining water in the water tank. The embedded system is controlled by a raspberry-pi-3B+ with a python program that we coded. For the connected system, we developed an android application and deployed a server based on oneM2M standard to host the data. We also did market research to see if our product can be successful and find possible improvements. We found that the size and the price of the device was the primary concern of surveyed people. Our first Watree prototype can manage to three plants at the same time with an autonomy of 4 weeks. The application with the server allows to customize the watering settings of each plant and access the current status of the device from all over the world. The proof of concept of our innovative project have been made, but the price and the design of the application are example of possible improvements.

Keywords: smart watering ; indoor watering ; plants management ; moisture control ; IOT ; internet of things ; custom watering ; smart water management system.

WaTree, Making Watering Smarter

By: Bayle Axel, Automatician and Electrornician student, and an Innovative Smart System specilization at INSA Toulouse.

The WaTree is a new innovative smart system for indoor which aim to make watering your plant smarter. The first part of the project was to understand the needs of the market to adapt the WaTree to the customers. With those results we started the conception of both the WaTree, with all the sensors (ultrasonic, capacitive, photodiode and platinum resistor), the pump management, and the android application, both settings and managing interfaces. We finished the development with the webserver and uses of the oneM2M standards to allows the information being available everywhere at any time. At the end, the WaTree provide an operational watering system with 4 weeks of autonomy and more than 5 different types of plants customizable by the application. This project demonstrates that our proof of concept is a promising solution to a real market need. Before launching it we should reduce the costs and make the application aesthetically pleasant.

Keywords : Smart watering, custom needs, indoor watering, plant health management, IoT, Internet of Things, oneM2M.

WaTree, Making Watering Smarter

By: Mikita PRAKAPENKA, Physics and Innovative Smart System student at INSA of Toulouse.

WaTree is a smart device, which will make your life easier and more connected. Starting from an idea of saving plants from death during our holydays period, we have developed this student project in the startup mood from a blank page. The project is composed of two different parts: commercial (market study, product design) and technical (hardware and software). International online surveys were used to bring us constructive results for the commercial part, helping to well design the final device. Technical part was started by the component choice. We have used the moisture sensor for each plant, and temperature, brightness, tank level sensors for whole system. Everything being operated by Raspberry Pi 3B+ electronic board, connected to your local Wi-Fi, sending data to the server and managing pumps to water or not the plants. One M2M is our server used for the data storage. This data is available to the user via an Android Application using an Internet connection from anywhere. The application allows to realize the first setup of the system and then to monitor all performances of it, being sure that the system is doing well for up to 5 weeks of autonomy.

Keywords : Smart watering, Autonomous watering, Indoor watering, IoT, Internet of Things, Raspberry Pi, OM2M.

WaTree, Making Watering Smarter

By: Pascual Álvaro, Computer science, network and Innovative Smart System student at INSA of Toulouse.

One of the objectives of applications based on the Internet of Things, is to make human life better. Therefore, we have developed WaTree whose goal is to water indoor plants in a smart and autonomous way. This project is divided in two different parts. Firstly, we did a market study and with the results obtained by our plurilingual survey, we were able to design and specify the functionalities of our product. Secondly, we made the technical part, starting by choosing the sensors (ultrasonic, capacitive, photodiode and platinum resistor) to measure plants' conditions, and pumps to water the plants. We implemented the different functions in a Raspberry Pi 3B+ connected to the home's Wi-Fi to send data to the deployed Web Server over oneM2M and recover this data thanks to an Android application developed by us. The first prototype has at least four weeks autonomy and can manage up to three plants fulfilling the market needs. Next steps for this project will be improving the application frontend and optimize the production line.

Keywords : Smart watering, autonomous watering, plants management, indoor watering, IoT, Internet of Things, Raspberry Pi, OM2M.

Annexe C : Module for the ultrasonic sensor

```
import pigpio as GPIO
import time
PIN_TRIG = 5
PIN_ECHO = 6

PLEIN = 5
VIDE = 32
def init_distance():
    pi = GPIO.pi()
    pi.set_mode( PIN_ECHO, GPIO.INPUT)
    pi.set_mode( PIN_TRIG, GPIO.OUTPUT)
    pi.write(PIN_TRIG,False)
    time.sleep(0.5)

def distance () :
    pi=GPIO.pi()
    # pulse sur la pin echo
    pi.write(PIN_TRIG,True)
    time.sleep(0.00001)
    pi.write(PIN_TRIG,False)
    started =time.time()
    start=started
    stop = -1
    print(pi.read(PIN_ECHO))
    while ( pi.read(PIN_ECHO)==0):
        start=time.time()
        if (start-started)>15:
            print("abort")
            return -1
    print(pi.read(PIN_ECHO))
    while ( pi.read(PIN_ECHO)==1):
        stop=time.time()
    if stop == -1 :
        print ("Erreur")
        return -1
    else :
        delai=stop-start
        distance= delai*34000/2 # vitesse son / 2 fois distance
        print("Distance : ",distance)
        return distance
```

```
def remplissage_cuve():
    profondeur = distance()
    remplis = ((VIDE - profondeur) / (VIDE - PLEIN)) * 100
    return remplis
```

Annexe D : Module for the temperature sensor

```
# Import Libraries
import os
import glob
import time

# Finds the correct temperature file that holds the temperature data
base_dir = '/sys/bus/w1/devices/'
temperature_folder = glob.glob(base_dir + '28*')[0]
temperature_file = temperature_folder + '/w1_slave'


# A function that reads the sensors data
def read_temp_raw():
    f = open(temperature_file, 'r')  # open the file
    lines = f.readlines()  # return the file data
    f.close()
    return lines


# Convert the value of the sensor into a  celsius temperature
def read_temperature():
    lines = read_temp_raw()
    # While the first line does not contain 'YES', wait for 0.2s
    # and then read the device file again.
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()

    # Look for the position of the '=' in the second line of the
    # device file.
    equals_pos = lines[1].find('t=')

    # If the '=' is found, convert the rest of the line after the
    # '=' into degrees Celsius
    if equals_pos != -1:
        temp_string = lines[1][equals_pos + 2:]
        temp_c = float(temp_string) / 1000.0
        return temp_c
```

Annexe E : Module for analogs sensors

```
#################
# Get ADC Value function
# For MCP 3008 Chip, including 0-7 channels
# CH 0,1,2 ARE HUMIDITY PINS, CH 3 IS THE LIGHT PIN
# Moisture sensor calibration 850 = 0rh 400 = 100 rh
#################
#from dis import _get_name_info


def getADC(channel):
    import time
    import Adafruit_GPIO.SPI as SPI
    import Adafruit_MCP3008

    # Hardware SPI configuration:
    SPI_PORT = 0
    SPI_DEVICE = 0
    mcp = Adafruit_MCP3008.MCP3008(spi=SPI.SpiDev(SPI_PORT,
                                                  SPI_DEVICE))

    # Read all the ADC channel values in a list.
    values = [0] * 8

    for i in range(8):
        # The read_adc function will get the value of the
        #  specified channel (0-7).
        values[i] = mcp.read_adc(i)

        # LECTURE HUMIDITY

        if i == 0 or i==1 or i==2:
            values[i] = ((((1.0 / values[i]) * 400.0) - 0.5)
                         * 200.0)

        # LECTURE LIGHT

        if i == 3:
            values[3] = (values[3] / 1023.0) * 100.0
```

Annexe F : Module for the pump acutator

```
import pigpio as GPIO

PIN_POMPE_1 = 22
PIN_POMPE_2 = 27
PIN_POMPE_3 = 17

pi = GPIO.pi()


def init_pompe():

    pi.set_mode(PIN_POMPE_1, GPIO.OUTPUT)
    pi.set_mode(PIN_POMPE_2, GPIO.OUTPUT)
    pi.set_mode(PIN_POMPE_3, GPIO.OUTPUT)


def gestion_pompe(x, mode):
    if (x == 1):
        tmp = PIN_POMPE_1
    elif (x == 2):
        tmp = PIN_POMPE_2
    elif (x == 3):
        tmp = PIN_POMPE_3
    else:
        print("error")
        return
    pi.write(tmp, mode)
    return 0
```

Annexe G : Main routine code

```
import distance as dist
import getADC as ADC
import temperature as tempe
import pompe as pompe
import time
import watreeserver as serv

DELAY = 3600  # delai entre deux loop en secondes


def init():
    dist.init_distance()
    pompe.init_pompe()


def arrosage():
    it = 0
    lumi_tab = [0] * 24
    while True:

        # lumi[U+FFFD]

        lumi_tab[it] = ADC.getADC(3)
        lumi_mean = lumi_moyenne(lumi_tab)
        # send mean to server
        serv.put_luminosite(lumi_mean)

        # temperature
        temperature = tempe.read_temperature()
        # send temp to server
        serv.put_temperature(temperature)

        # niveau d'eau
        niveau = dist.remplissage_cuve()

        if (niveau > 0):
            # recup des besoins

            #  TO DO : request serveur
            needs = [0] * 3
            needs[0] = float(serv.get_humidite(1))
```

```
                needs[1] = float(serv.get_humidite(2))
                needs[2] = float(serv.get_humidite(3))
                # recup moisture plante
                moisture = [0] * 3
                moisture[0] = ADC.getADC(0)
                moisture[1] = ADC.getADC(1)
                moisture[2] = ADC.getADC(2)

                # comparaison
                for x in range(3):
                    if ((moisture[x] - needs[x]) < -5):
                        pompe.gestion_pompe(x + 1, True)
                        time.sleep(2)
                        pompe.gestion_pompe(x + 1, False)

                # fin while
            it = (it + 1) % 24
            # niveau d'eau
            niveau = dist.remplissage_cuve()
            # send niveau
            serv.put_niveau_eau(niveau)

            # pause between loop
            time.sleep(DELAY)


def lumi_moyenne(lumi_tab):
    sum = 0
    for i in (lumi_tab):
        sum += i
    mean = sum / len(lumi_tab)
    return mean
```