ecø, a Rainforest Product

Pablo Acereda Gracia, David Emanuel Craciunesuc, Pablo Martínes García, and Laura Pérez Medeiro

Introduction

This project will consist of an intelligent system with capabilities to control abiotic factors that affect the wellbeing of plants, with the objective of improviing their health and quality of life.

Factors such as humidity and temperature, as well as the levels of light these receive, will be monitorized and analyzed frequently in order to verify and ensure optimal quality of life for the plants.

Thanks to these measurements, a series of alarms and different warning states will be implemented in order to better control the status of the plants and alert the users of their current situation.

Users will be able to easily install the system themselves, and interact and control it via a virtual assistant such as *Amazon Alexa*.

Context

Current situation of the presented problem

The market has seen its fair share of intelligent systems created to aid with plant irrigation control, and there are those that even come with a mobile app interface, such as Blossom, PlantLink or Edyn. Some of the newcomers to jump aboard the backyard-sprinkler train are the so-called 'intelligent flowerpots', like the one *Xiaomi* has recently started to sell, capable of watering the plants automatically depending on the humidity of the soil.

There are other projects like *GR0* that are capable of suggesting what plant species one should buy by analyzing the quality and type of soil one uses.

Nevertheless, none of these offers use any kind of virtual-assistant integration or any well-designed user experience, for that matter. *That* will be the main difference our project will have. Not only will the user control the system through a virtual assistant, the design and user experience is planned to be exceptional and extremely easy and intuitive.

End-of-project prediction

Once the course finishes, our intention is to have created a device capable of auomatically keeping alive crops or plants with interactions through *Amazon Alexa*.

In order to achieve that, our device will be able to give accurate weather predictions, alert of possible plagues that might attack the crops and monitor the optimal conditions for the plants themselves.

Target audience

This project aims to aid the gardening aficionados that do not have a great amount of time at their disposal to take care of their plants optimally. It also aims to assist farmers that need help with the care of their crops and seek to minimize the effect of unexpected and external factors to their produce.

Project Scope

Just as previously mentioned, the main objective of this technological endeavor is the creation of a good-enough prototype in order to shot to multiple possible future investors so that they help to commercialize and empower the creation of the product itself through their monetary support. In short and plain English, the objective of the product is to attract suitable investors that, with their contributions, would help the product grow and transform into a marketable commercial system. This product, '*ecø*', is currently the only product the startup *Raninforest* has, therefore this project is of vital importance to the company.

ø' is a product than can be commercialized both as a physical system and as

Discarded Ideas

Technology to Use

This section contains the different options that were considered for the project. The different decisions about the different used technology within ght project took into account the experience of the various members of the group, as well as the monetary cost of the different hardware elements, their general availability and the trust the very manufacturer inspired.

Controllers

- **Arduino**
  Arduino is an open-source electronics platform that is based on easy-to-use software and hardware. The *Arduino* boards themselves are controlled by sending a set of instructions to a microcontroller on the board. To do so, one must use the *Arduino* programming language, which is based on *Wiring*, and the **Arduino Software (IDE)**, which is based on *Processing*. These boards were our first choice because they are extremely easy to use. The programming language is extremely easy to pick up if one already knows *C*, and the *Arduino* community offers a wide array of free-to-use resources that improve the quality and reduce the effort of any project trying to use *Arduino*.

Even with all it's benefits, *Arduino* ended up being discarded as a possible option, given that the absolutely cheapest of boards would still cost around $20.

- **Raspberry Pi**
  According to the official *Raspberry Pi* webpage: *"Raspberry Pi is a low cost, credit-card sized computer that plugs into a monitor or a TV, and uses standard keyboard and mouse. (. . .) It's capable of doing what you'd normally expect a regular computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, word processing and playing games".*
  Just like a regular computer, once would be able to program it to do whatever they'd want it to do. This would have been ideal for the project itself, given the simplicity of its use and the gigantic array of languages that are compatible with it. In the end, though, it was discarded because the computer itself was too large in comparison to the rest of the elements of the project.
- **ESP32**
  The *ESP32* is a series of low-cost, low-power system on a chip **SoC** microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. This was the option the team ended up choosing for the project, given that the *ESP32* was specifically designed for **wearable electronics and IoT applications.**
  One can also program on it using the *Arduino IDE*, which was a huge advantage. Most of the team already knew how to use and control *Arduino*, so not having to learn a skil specifically for the microcontroller plus all the benefits of the *Arduino* community made the group decide on the *ESP32* as an option.

Sensors

- DHT22
- DSB18B20:
- LDR:
- YL-69:

Programming Languages

As previously mentioned, the chosen microcontroller is the *ESP32*. Therefore, the options when it comes to programming languages have been:

- MicroPython
  MicroPython an open source *Python* programming language interpreter that is capable of running on small embedded development boards. With the adaptability build into *MicroPython*, one can write clean and simple

Python code to control hardware directly instead of having to use complex low-level languages.
Even if a reduced version, *MicroPython* still supports most of *Python*'s syntax and implements most of its inner mechanism. Given all these features and advantages, it was a quick initial choice for the project.
These are some of the features that set it apart from other embedded systems:

  - **Interactive REPL** This feature allows execution of code without the need of any compilation or uploading time, which is perfect for systems with a high level of experimentation.
  - **Extensive software library** *MicroPython* already comes with libraries built in to support common tasks, like JSON data parsing, regular expression handling or even network socket programming.
  - **Extensibility** Advanced users ma be able to mix *MicroPython* with extensible low-level C/C++ functions in order to further optimize their code and make the execution faster when it really matters.

  MicroPython has some very useful features. Unfortunately, it also comes with its downsides:

  - Slower code and higher memory needs when compared to C/C++.
  - Complicated microcontroller initialization process.
  - Limited functionality for some key libraries.

- FreeRTOS
  FreeRTOS is a real-time operating system made specifically to run on embedded systems. It is especially good because it natively provides core real time scheduling, inter-task communication, timing and syncrhonisation primitives. This transalates into a more accurately controlled kernel and a system that is able to execute tasks exactly when they have to be executed, a *deterministic* system.
  In *FreeRTOS*, applications can be assigned in a static manner while objects themselves can be assigned dynamically with different memory-assignment memory schemes. This system was quickly discarded. Even with the great deal of documentation it counts, the complexity of *FreeRTOS* and the learning curve it would entail made it practiacally impossible in the provided timeframe.

- Mongoose OS
  Mongoose OS is an Internet of Things Firmware Development Framework under Apache License Version 2.0.
  Mongoose OS is a firmware development framework specifically designed for development of IoT products. It is highly compatible with a wide array of microcontrollers, just like the *ESP32*, and it's objective is to fill 'the noticeable for embedded software developers' between firmware created for prototyping and bare-metal microcontrollers.

It comes with an integrated web server and supports interaction with both private and public clouds like AWS IoT or Mosquitto.

- Arduino IDE:

Software Development Methodologies

Application Architecture

Business Model

Development Plan

Risk Assessment

Contingency Plan

Overview of the Project