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.

'*ecø*' is a product than can be commercialized both as a physical system and a digital solution for the client. As a physical system it will be able to monitor all the ambiental factors that could affect a plant's growth and development, e.g. humidity, temperature, luminosity, soil quality etc.

As a digital solution, it comes with visualization software that will be able to show and display information in a clear and concise way, as well as in an intuitive manner. The project promoters also hope to be able to motivate a big community of collaborators so that the product improves substantially over time.

This is not all though, for the '*ecø*' also comes with Virtual Assistant integration, with which the user would be able to interact easily and seamlessly.

When it comes to future functions or ones in early stages of development, the biggest one so far has been the incorporation of sensors and mechanisms capable of detecting the pressence of bugs and insects that could pose a threat to the plant. There are multiple other initiatives and lines of development and improvement for the product within the company, but those are in very early stages and developing further into those lines of work in this document would not be appropriate, given the volatile nature of initiatives in such early stages.

Discarded Ideas

The project has suffered many changes since its intial conception and ideation. Some of these changes and modifications have been so great and have influenced the project such a big deal that they earn a mention in the following list, just by their very nature as disruptive and polarizing within the developing team:

- **Detection of plagues**
  An initial idea was that the system should detect different insects and monitor for known plagues that attack plants and crops. Conceptually insect detection seems simple but its implementation isn't. Some initial

ideas entailed the used of infrared radiation, but this idea turned out to be completely inefficient.
This possible feature has not been completely discarded just yet. There is a type of sensor called *PIR*, which could be configured in such a way that detect small-sized animals. The development of this idea is currently experimenting some issues.

- **Development of own microcontroller**
  This would have been extremely useful for the product itself, given that the development of a microcontroller specialized for the task would have optimized the way in which the product was designed. After much thought, this idea was discarded, given the short timeframe this product has for development and the ambitious nature of the initiative.

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

There are many different kinds of sensors on the market. Many of these are extremely capable and can be obtained very easily. When deciding on the sensors, the team always looked for the most reliable and available on the market. Here are some of the sensors the project will use:

- **DHT22**
  It is a basic and low-cost digital temperature and humidity sensor. It measures the surrounding air thanks to a capacity humiditor sensor and a thermistor, and spits out digital signal on the data pin. It's extremely simple to use, given that its transmissions are already digital, but one can only make readings each two seconds. Extreme speed, or lack thereof, is not a highly important factor when it comes to plant humidity and temperature measuring, so the only real downside would not be an actual downside.
  Just like any avid and shrewd reader would have realized by now, this sensor will be used to measure exterior temperature and humidity.
- **LDR**
  A *Light Dependent Resistor* or a photo resistor is a device whose resistivity is a function of the incident electromagnetic radiation. These are light sensitive devices also called photo conductors, photo conductive cells or simply photocells.
  The ability to detect light is a must in the soon-to-be deliverable product, therefore the use of this sensor was an easy and obvious solution.
- **YL-69**

According to the product description the manufacturer provides: *"The YL-69 is a simple sensor that can be used to detect soil moisture/relative humidity within the soil. The module is able to detect when the soil is too dry or wet. Great for use with automatic plant watering systems"*.
The need for a sensor like this, especially considering it's insignificant price in comparison to the price of the rest of the setup and components. Terefore, this will be the main triggering mechanism to notify the plant or plants need to be watered.

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
  It's an open-source project that has a very large community behind it. *Arduino IDE* comes with a text editor for writing code, a message area, a console, and other common functionalities of full-fledged IDEs. The programming languages it admits are C and C++, but with some special rules when it comes to structuring the code. The IDE also comes with plenty of built-in libraries for common tasks like robot control or I/O mechanisms.
  This is the option the team ended up choosing, not only for its simplicity and usefulness, but also because it can interact directly with the *ESP32* chip that the project will use as a main component.

Development Methodology

One of the key factors when it comes to a project, especially the creation of a new one, is the *Development Methodology.* These methodologies are a means by which a product can be created in an organized, somewhat predictable and planned manner. A *Development Methodology* is nothing more than the process of planning, creating, testing and then deploying a project. And the different types of methodologies only alter the way in which those components interact with each other.

The specific methodology used for this project is *Lean w/ Kanban.* Here are some of the main reasons that made the group choose this methodology:

1. **Waste reduction**. Helps keep project simple and clean. Everything that is not absolutely necessary is promptly eliminated, be it garbage code, unecessary meetings, obsolete hardware, etc.

2. **Knowledge improvement**. This is one of the best aspects of *Lean.* Even if every member has a defined function within the team and their tasks are clearly stated, any other member can lend a hand and help if they feel the need or have the time. This helps keep the team agile and on their feet.

3. **Delayed decision-making**. Within *Lean* decisions are made as the requirements are being met, as well as their focus starts to change. The project starts building itself in a progressive and adaptative manner because it hasn't followed a strictly defined idea since its inception. Instead, it has been allowed to grow and morph into it needed to become.

4. **Quick delivery**. Given the minimalism of the methodology and its short iterations of work, the authors of this document believe a further explanation of this specific aspect would be an insult to the reader.

5. **Team-oriented**. Every single member is part of the decision-making process, as well as the definition of their own work. This methodology works best in teams that extremely motivated because the different members usually divide the work amongst them without the need of actual meetings and specific, written-down planification.

6. **Focus on the objective**. Thanks to many of the factors mentionded above, the project does not lose sight of the objective. One could say the *motto* of the team would be *"Think big, act small, err frequently and fix quickly"*.

When combining *Lean* with *Kanban*[1] controlling the assignment of different tasks and jobs of the different team members was extremely easy and intuitive. This combination is the key component to the development of the prototype of the product, since it increases visibility of the project and enhances control of the direction in which the development steers towards.

Lean Kanban is a methodology used by various startups, just like IMVU, which is a company created by Eric Ries. He created a methodology called *Lean Startup*, which as 'improved' version of the methodology this project uses, designed specifically to adapt to the way a startup works.

The development cycles in *Lean Kanban* start with the creation of an idea, which is then assessed and developed in order to analyze if its addition to the overall project would be of any real value. With this process, the worse case scenario is still obtaining knowledge about the creation and implementation of that specific idea, even if it does not end up being part of the project. This is the way some of the most important parts of '*ecø*' were created, like the *Amazon Alexa* virtual-assistant integration or the detection of factors like humidity and light levels.

[1] Kanban is a visualization methodology and mechanism, not an actual defined development process.

Implementation of the metodology

In order to apply the methodology in an effective manner, the team has used the project collaboration tools of *GitHub Projects*, which officially integrated as part of the software of University of Alcalá thanks to the *GitHub Campus Program.*

The team met weekly to plan their activities and tasks for the following week and kept in touch with the mentioned tools, also able to track changes in the status of each task and maintain the visibility of the overall project.

Every single modification and all the planification has been recorded thanks to *GitHub Projects* and a link to the respository will be provided with this document.

Application Architecture

Business Model

Development Plan

Risk Assessment

Contingency Plan

Overview of the Project