Smart Garden

Analyze, design and implement a project prototype that connects a WEB Server based on RPi with other actuators with the goal of creating an autonomous system.

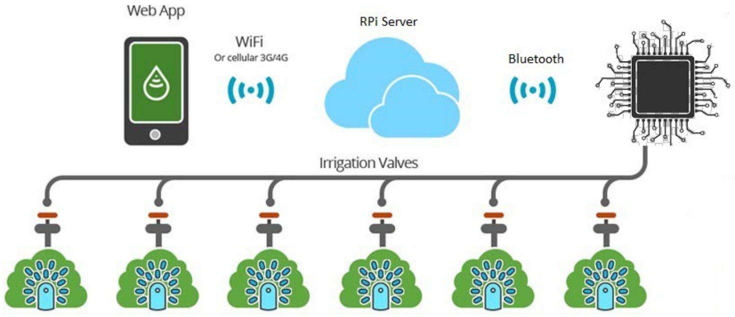
Studenti:

Olaru Mihai-Alexandru 1405B

Tiganescu Dragos 1405B

Axinte Silviu-Costin 1405B

Project theme

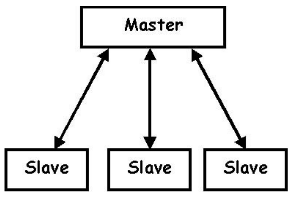


This project aims to automate some of the work which is done in the garden, and also to keep its owner informed about the status of the soil and weather.

The goal is not only a remote control of the tasks done in the garden, but, in lack of input from the owner, the system should be able to take care of the garden on its own.

Project requirements

The system will have two principal modules: the master and the slaves.



The master

This module will be the server hosted on **RPi**. The server's task is to monitor the slaves and to process their information.

- Update the information from the slaves on the WEB Page
- Send commands to the slave (for now the command is to water the garden)

The slaves

Its main task is to monitor the garden through the sensors and give the information collected to the server. Actions will take place only when the server decides to.

Note: This design has been chosen for its ability to be scalable. (we can add as many slaves as we need)

Necessary Hardware

For the chosen design we will use the following items:

- One Raspberry Pi Zero W board
- One XMC 1100 Infineon board
- Humidity sensors
- Temperature sensors
- Distance sensors

- Bluetooth modules
- Water pump
- Water tank
- Battery

Other implementations

“**Click and Grow”** is a company that is trying to supply the same service. This is the link to their webpage. They offer this service as an indoor solution for plant growth, what we try to offer is an outdoor solution, for the garden and the surroundings of the house (and even inside the house) .

Resources management

Project

The project versioning will be dealt with in Bitbucket using Git.
https://bitbucket.org/givemeafork/smart_garden/src/master/

Team

Every member of the team took one of the big parts of the project like this:

1) Olaru Mihai-Alexandru

   a) Set up the server
   b) Handle the communication between the modules.

2) Tiganescu Dragos

   a) Work with Mihai on getting the server working
   b) Develop the Web page with all its features

3) Axinte Silviu-Costin

   a) Develop the functionality of the slave module

Project functionality

The slave

As we said before, the main task of a slave is to monitor the garden and give the feedback to the server.
If the server decides to wet the garden, the slave shall be able to accomplish that task.

Monitoring the garden

The slave uses **humidity** and **temperature** sensors to monitor the garden.
The slave will also be in charge of monitoring the water level from the water tank. We will monitor the water level using a distance sensor.
These sensors will be powered with 3.3 V and they will give analog signal as feedback for their environment.

Actuators

The slave uses a **water pump** to wet the garden. We will power the pump with a battery. The pump has only the alimentation terminals.
To control this actuator we will also need a **motor driver**. This driver will be powered from the battery, and has as input 3 pins (enable, and two pins for the control of the motor).
For the enable input, the xmc board will give a pwm signal (to control the speed of the pump), and for the control inputs we will use 2 pins that will set the pump on.

The xmc1100 board

This is the hearth of the slave. It processes all the data retrieved from the sensors and commands the water pump to wet the garden when the server decides to.
The communication with the server is done using Bluetooth. The communication is explained below.

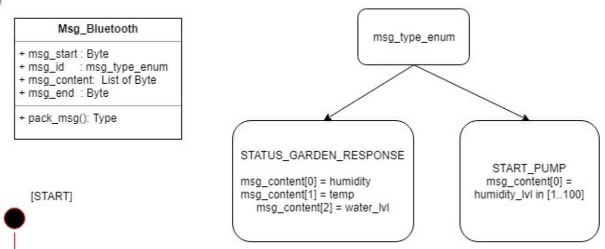
Communication

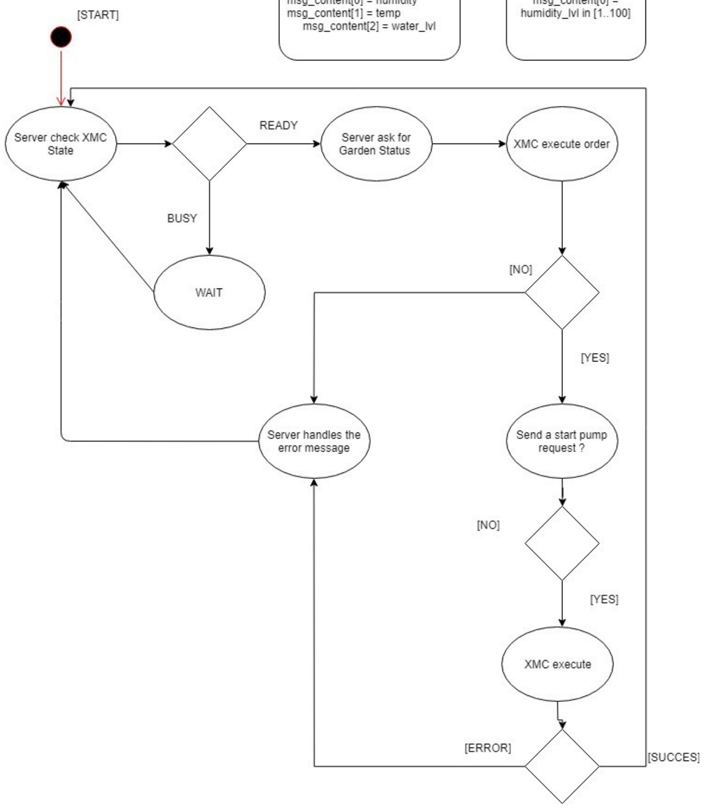
The communication with the server is done using Bluetooth.
The server will asynchronously check the status of the garden and if needed will instruct the slave to wet the garden. The communication is explained in the following image.

Communication between XMC and Server

-> Message Structure

-> Message Flow


Msg_Bluetooth
+ msg_start : Byte
+ msg_id : msg_type_enum
+ msg_content: List of Byte
+ msg_end : Byte
+ pack_msg(): Type
msg_type_enum
STATUS_GARDEN_RESPONSE
msg_content[0] = humidity
msg_content[1] = temp
msg_content[2] = water_lvl
START_PUMP
msg_content[0] = humidity_lvl in [1..100]


[START]
Server check XMC State
READY
Server ask for Garden Status
XMC execute order
BUSY
WAIT
[NO]
[YES]
Server handles the error message
Send a start pump request ?
[NO]
[YES]
XMC execute
[ERROR]
[SUCCES]