|  |  |  |  |
| --- | --- | --- | --- |
| Assignment 1 | | Project Summary | |
| Course | | Practical Robotics and Smart Things - 2021 | |
| GitHub Repository | | https://github.com/nikola400/IDI-System | |
|  | | | |
| Project author | | | |
| № | Name | | Faculty Number |
| 1 | Nikola Petrov | | 62199 |

|  |  |
| --- | --- |
| Project name | Intelligent Drip Irrigation System - IDI System |

|  |
| --- |
| 1. Short project description (Business needs and system features) |
| The *Internet of Things (IoT)* has made our homes and electrical appliances smarter, interconnected and easier to use. The proposed project develops a system, providing a smart way to irrigate the plants in your garden - *Intelligent Drip Irrigation System (IDI System).*  The system is planned to use a well with inconsistent water level - that is why we need another water container.  The hardware implementation is based on two ESP32 microcontrollers, equipped with following sensors:   * *BME280 - used to measure the temperature in the air;* * [*Capacitive Soil Moisture Sensor*](https://www.aliexpress.com/item/1005001621780703.html?spm=a2g0o.productlist.0.0.6b503e84soSZlS&aem_p4p_detail=202106271422566219378799742080014644563) *- used to measure the soil moisture;*   *Next scope:*   * *(Still in search) Sensor for recognizing the level of the water in the well;* * *(Still in search) Sensor for recognizing the level of the water in the water container;*   IDI System actuators include:   * [Water valve](https://www.aliexpress.com/item/4000208125696.html?spm=a2g0o.productlist.0.0.3fc15aa7hFwZVH&aem_p4p_detail=202106271207368712269062280013862216) to start and stop the water from the container to the system;   Next scope:   * A pump for the well - used to fill the container with water from the well. Controlled with the second ESP32 and a dedicated relay for that pump; * A relay to control the pump;   The IDI System has the following main modes of operation:   * *Default mode - Gathering environment information e.g. temperature, soil moisture, weather forecast, current time to calculate if watering is needed.* * *Watering mode - when the water valve is unlocked and drop irrigation is running.*   There is also a (Grafana or something else) dashboard, where admin user could see sensor’s measurements, the level of the water in the container and the level of the water in the well. The System will push alert notifications if there is critical water level in the well.  In the next scope of the system, it is planned to have another water source as a backup to the well.  There will be only two main user roles (actor in UML):   * *Regular User* – can login into Grafana, browse dashboards and receive notifications from the system * Administrator - this user is also able to start and stop the whole system and can manage whether the system should use weather forecast.   There is a Web Server for easy access of the system metrics. The web server is also used by user to configure the System(for now only plant type is configurable, but the irrigation hours and the use of weather forecast would be configurable soon.)  The available endpoints of the ESP32 Web Server are:   * GET / - gives information about all of the rest endpoints * GET /test - just a test endpoint to see that the Web Server is up * GET /temp - gives information about air temperature * GET /hum - gives information about air humidity * GET /alt - gives information about approx. altitude * GET /soilMoisture - gives information about soilMoisture * GET /metrics - gives information about metrics from all sensors * GET /plant - gives information about plant configuration of the system * POST /setPlant - upsert(insert or update) plant configuration of the system   Important : Default plant configuration is set for TOMATO |

|  |  |  |
| --- | --- | --- |
| 1. Main Use Cases / Scenarios | | |
| **Use case name** | **Brief Descriptions** | **Actors Involved** |
| * 1. **Login to (Grafana) service** | The *User* would be previously registered in the Grafana. They should be able to login to Grafana service via loginName and password. | Regular Users, Admins |
| * 1. **Browse information via dashboard** | *Users should be able to* browse the information that is gathered by the sensors e.g. air temperature, soil moisture, water level, etc. | Regular Users, Admins |
| * 1. **Receive notifications** | *Users should receive warning notifications if the level of the water is low. There will be separate notifications for the water level of the well and the water level of the container.* | Regular Users, Admins |
| * 1. **Manage Whether Forecast** | *Administrators* *should be able to set interactively via REST requests to the IDI System whether the system should or should not use a forecast to calculate when watering is needed.* | *Admins* |
| * 1. **Manage System** | *Administrators* can start and stop the IDI System when needed via REST request to the system. | *Admins* |

|  |  |
| --- | --- |
| 1. Main Dashboards | |
| **Dashboard name** | **Brief Descriptions** |
| * 1. **Measurements** | Presents all of the sensor’s measurements that are gathered by the system. |
| * 1. **Water level** | Presents what is the current level of water in both the well and the container in different ways. (e.g. with quantity measurement in litres, in percents, with hidrograms) |
| * 1. **General Dashboard** | A dashboard that presents the sensor's measurements and water levels together in a simple way. |
| * 1. **Detailed Dashboard** | Shows different detailed panels with the available information. (e.g. the average air temperature for the week/month, the average precipitation for week/month/year, a plot with temperature/soil moisture values, etc.) |

|  |  |  |
| --- | --- | --- |
| 1. API Resources (REST/SSE/WebSocket Backend) | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Users** | GET *User Data* for all users, and POST new *User Data*. Available only for *Administrators*. | */api/users* |
| * 1. **User** | GET, PUT, DELETE *User Data* for *User* with specified *userId*. Available only for Administrators | */api/users/{userId}* |
| * 1. **Login** | POST *User Credentials* (loginName and password) and receive a valid *Security Token* to use in subsequent API requests. | */api/login* |
| * 1. **Logout** | POST a logout request for ending the active session with *OKTS,* and invalidating the issued *Security Token*. | */api/logout* |
| * 1. **Configurations** | GET *Configurations of the IDI System.* Available only for administrators. | */api/config* |
| * 1. **Configurations** | POST *Configuration of the IDI System. (if the system is on or off and and if the weather forecast is on or off)* Available only for administrators. | */api/config?configParam=value* |
| * 1. **Push Alert Notifications** | GET alert notifications that were pushed to the users. (all notifications will be kept in the database with some retention period. | */api/alerts* |

|  |  |  |
| --- | --- | --- |
| 1. Web Server API Resources | | |
| **View name** | **Brief Descriptions** | **URI** |
| * 1. **Root** | GET information about the available endpoints in the Web Server | */esp32/* |
| * 1. **Temperature** | GET current temperature | */esp32/temp* |
| * 1. **Humidity** | POST *current humidity* | */api/hum* |
| * 1. **Altitude** | POST current approx. altitude | */api/alt* |
| * 1. **Soil Moisture** | GET current soil moisture | */api/soilMoisture* |
| * 1. **Metrics** | POST *current state of all of the available metrics* | */api/metrics* |
| * 1. **Plant Configuration** | GET plant configurations. This includes name of the plant, Plant minimal soil moisture and Plant maximum soil moisture | */api/plant* |
| **5.8. Plant Configuration** | POST plant configuration. Update plant configuration | */esp32/setPlant* |