Università degli Studi di Torino
DIPARTIMENTO DI INFORMATICA
Corso di Laurea Triennale in Informatica

Academic Year 2022/2023

Design and Development of the Digital Twin of a Greenhouse

candidate
Eduard Antonovic Occhipinti
947847

supervisor
Prof. Ferruccio Damiani

co-supervisors
Prof. Einar Broch Johnsen

Rudolf Schlatte

Eduard Kamburjan

Aknowlegements

Special thanks to Chinmayi Bp for always being available to help us figure out the electronics and Gianluca Barmina and Marco Amato with whom I worked together on the project

Declaration of Originality

"Dichiaro di essere responsabile del contenuto dell'elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d'autore. Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata."

Abstract

We will see how digital twins can be used and applied in a range of scenarios, we'll introduce the language 'SMOL', created specifically for this purpose, and talk about the work of me and my collegues in the process of designing and implementing a digital twin of a greenhouse.

Contents

Introduction

The following project was realized as an internship project during my Erasmus semester abroad in Oslo, Norway. The project was realized in collaboration with the University of Oslo and the Sirius Research Center.

The project consists in the creation of a digital twin of a greenhouse and the optimization of the environmental conditions for maximum growth. Using SMOL we can predict when to water the plants and how much water to use.

Learning Outcome

- Understanding the concept of digital twins
- Learning the syntax and semantics of SMOL
- Learning the basics of the FMI standard
- Learning basic electronics to connect sensors and actuators to a Raspberry Pi

By modelling a small scale model of a digital twin we were able to build the basis for the analysis of a larger scale problem. Other than the competences listed above, it was also important being able to work in team by dividing tasks and efficiently managing the codebases of the various subprojects (by setting up build tools and CI pipelines in the best way possible, for example).

Electronic prototyping was also a big part of the project. We had to learn how to connect sensors and actuators to a Raspberry Pi (which required the use of breadboards, analog-to-digital converters and relays for some of them) and how to translate the signal in a way that could be used as data.

Results

A functioning prototype was built with the potential to be easily scaled both in terms of size and complexity. Adding new shelves with new plants is as easy as adding a new component to the model and connecting the sensors and actuators to the Raspberry Pi. The model is easily exentable to add new components and functionalities such as actuators for the control of the temperature and the humidity or separate pumps dedicated to the dosage of fertilizer. Sensors are also easily added (a PH sensor would prove useful for certain plants) both in terms of the asset model and in terms of code, each sensor/actuator is represented as a class and they are all being read indipendently from each other.

Tools and Technologies

The following is a general oveview of the tools and technologies used in the project.

InfluxDB

InfluxDB is a powerful open-source time-series database that is used to store the data collected by the *data collectors*. It is designed to handle high volume and high frequency time-stamped data. Being a NoSQL database, it is schemaless and it allows for a flexible data model.

It serves as the main data storage for the greenhouse.

It's organized in *buckets* that are used to store the *measurements*. Each *measurement* is composed of:

- *measurement name*
- *tag set*
  - Used as keys to index the data
- *field set*
  - Used to store the actual data
- *timestamp*

For example the following query will return the last moisture measurement, given a measurement range of 30 days, from the bucket of name `greenhouse`:

```
from(bucket: "greenhouse")
  |> range(start: -30d)
  |> filter(fn: (r) => r["_measurement"] == "ast:pot")
  |> filter(fn: (r) => r["_field"] == "moisture")
  |> filter(fn: (r) => r["plant_id"] == %1)
  |> keep(columns: ["_value"])
  |> last()
```

Chronograf is included in the InfluxDB 2.0 package, it is the tool that we chose to visualize the data stored in the database and to create dashboards.

Figure 1: An example of the visualization tools offered by the Chronograf Dashboard

Developing a library to interface with the sensors

When working with the Raspberry Pi 4 the obvious choice for a programming language is Python, it is the most widely used language for the Raspberry Pi and it has a lot of support and libraries available.

The goal was to make it extremely modular to be able to add new sensors and actuators with ease.

OWL

OWL is a knowledge representation language that is used to describe the `asset model` of the greenhouse. It is used to create a formal description of the greenhouse's physical structure and the relationships between the different components.

The following code example models a class 'Basilicum' and links information about the ideal moisture in the asset model:

```
###  http://www.semanticweb.org/gianl/ontologies/2023/1/
sirius-greenhouse#Basilicum
ast:Basilicum rdf:type owl:Class ;
  rdfs:subClassOf ast:Plant ,
    [ rdf:type owl:Restriction ;
        owl:onProperty ast:hasIdealMoisture ;
          owl:hasValue "70.0"
    ] .
```

SMOL Language

`SMOL` is an OO programming language in its early developement stages, it allows us to:

- Interact with the `InfluxDB` and read data from the database, directly without the need of a third party libraries
- Read and query the knowledge graph, mapping the data to objects in the heap
- Map the program state to a knowledge graph by means of semantic lifting, the program state can be then queried to extract information about the state of the system
- Represent and run simulation and interact with `modelica`

It will be treated in more details in its dedicated section.

Digital Twins

NASA's definition of digital twin
"An integrated multiphysics, multiscale, probabilistic simulation of a veichle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its flying twin. It is ultra-realistic, and may consider one or more important and interdependant veichle systems"

A digital twin is a live replica of a Pysical System and is connected to it in real time, application. Digital Twins are meant to understand and control assets in nature, industry or society at large, they are meant to adapt as the underlying assets evolve with time. [1]

Applications

Digital Twins are already exensively used in a wide range of fields, ranging from power generation equipment - like large engines, power generation turbines - to establish timeframes for regularly scheduled maintenance, to the health industry where they can be used profile patients and help tracking a variety of health indicators. [2]

SMOL

SMOL (Semantic Micro Object Language) is an imperative, object-oriented language with integrated semantic state access. It can be used served as a framework for creating digital twins. The interpreter can be used to examine the state of the system with SPARQL, SHACL and OWL queries.

Co-Simulation

SMOL uses *Functional Mock-Up Objects* (FMOs) as a programming layer to encapsulate simulators compliant with the FMI standard into object oriented structures [1]

The project is in its early stages of developement, during our internship one of our objectives was to demonstrate the capabilities of the language and help with its developement by being the first users.

SMOL

SMOL (Semantic Micro Object Language) is an imperative, object-oriented language with integrated semantic state access. It can be used served as a framework for creating digital twins. The interpreter can be used to examine the state of the system with SPARQL, SHACL and OWL queries.

Co-Simulation

SMOL uses *Functional Mock-Up Objects* (FMOs) as a programming layer to encapsulate simulators compliant with the FMI standard into object oriented structures [1]

The project is in its early stages of developement, during our internship one of our objectives was to demonstrate the capabilities of the language and help with its developement by being the first users.

Software Components

The following is an overview of the software components it was necessary to write to achieve our goals.

Data Collector Program

To collect data from the sensors connected to the greenhouse we wrote a python program that retrieves the data and uploads them to InfluxDB.

The repo can be found at https://github.com/Nessuno/greenhouse-data-collector. What follows is an overview of the program

Greenhouse Asset Model

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri.

SMOL Twinning program

The `SMOL` program is run periodically by the server and is responsible for creating the digital twin and running the FMI simulators. It achieves this in the following steps:

1. It reads the `asset model` from the `OWL` file
2. It generates `SMOL` objects from the asset model individuals
3. For each asset object it retrieves the sensor data associated with that specific asset from the database
4. After retrieving the data it performs the semantic lifting of the program state, creating a knowledge grappph that represents the state of the assets in the greenhouse

The Greenhouse

The specific greenhouse we're working on has the following characteristics:

- It is divided in two shelves
- Each shelf is composed by two groups of plants
- Each group of plants is watered by a single water pump
- Each group of plants is composed by two plants
- Each plant is associated with a pot

Assets - Sensors

The following sensors are used to monitor the environmental conditions of the greenhouse and the plants:

Greenhouse

- 1 webcam used to measure the light level, can be replaced with a light sensor that would also provide an accurate lux measurement

Shelves

- 1 `DHT22` sensor used to measure the temperature and humidity

Pots

- 1 capacitive soil moisture sensor used to measure the moisture of the soil

Plants

- 1 `Raspberry Pi Camera Module v2 NoIR` used to take pictures of the plants and measure their growth by calculating the `NDVI`

The Role of each Raspberry Pi

There are in total 5 Raspberry Pi 4 used in this project. The division in roles and the usage of the same hardware makes it very scalable and easy to replicate. As follows:

- 1 Raspberry Pi 4 is used as a server, it hosts the digital twin and the FMI simulators, the server is also used to host the database in which all the data is stored and accessed.
- 3 Raspberry Pi 4 are used as clients, they are connected to the sensors and the actuators and are responsible for sending the data to the server.
- 1 Raspberry Pi 4 is used as a router and serves to connect clients and server wirelessly.

The Server The host runs an `InfluxDB` instance that holds the data retried from the clients (*data collectors*) and a `Java` program that periodically runs the `SMOL Twinning program` which is responsible for creating the digital twin and running the FMI simulators.

The Clients We also refer to them as *data collectors*

The Router The Raspberry was configured with `hostapd` and `dnsmasq` to act as a router and provide a wireless network for the clients to connect to. The local network is used to access the client via `SSH` and to send data to the server via `HTTP` requests.

The Digital Twin

Semantic Lifting

References

[1] Eduard Kamburjan, and Rudolf Schlatte, “The SMOL language.” https://smolang.org/

[2] IBM, “What is a digital twin?.” https://www.ibm.com/topics/what-is-a-digital-twin