ESCUELA POLITÉCNICA SUPERIOR DE MONDRAGON UNIBERTSITATEA
MONDRAGON UNIBERTSITATEKO GOI ESKOLA POLITEKNIKOA
MONDRAGON UNIVERSITY FACULTY OF ENGINEERING

Trabajo presentado para la obtención del título de
Titulua eskuratzeko lana
Final degree project for taking the degree of

GRADO EN INGENIERÍA EN INFORMÁTICA
INFORMATIKAKO INGENIARITZA GRADUA
DEGREE IN COMPUTER ENGINEERING

Título del Trabajo *Lanaren izenburua* Project Topic

RUNTIME VERIFICATION FOR SPATIO-TEMPORAL PROPERTIES OVER IOT NETWORKS

Autor *Egilea* Author OIHANA GARCIA ANACABE

Curso *Ikasturtea* Year 2021/2022

Título del Trabajo *Lanaren izenburua* Project Topic

RUNTIME VERIFICATION FOR SPATIO-TEMPORAL PROPERTIES OVER IOT NETWORKS

Nombre y apellidos del autor
Egilearen izen-abizenak
Author's name and surnames
GARCIA ANACABE, OIHANA

Nombre y apellidos del/los director/es del trabajo
Zuzendariaren/zuzendarien izen-abizenak
Project director's name and surnames
EZIO BARTOCCI
ILLARRAMENDI, MIREN

Lugar donde se realiza el trabajo
Lana egin deneko lekua
Company where the project is being developed
TU WIEN

Curso académico
Ikasturtea
Academic year
2021/2022

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Laburpena

(Laburpen amaieran ipini dokumentuaren amaierarantz informazio gehiago dagoela euskaraz *Erreferentzia bat sartu atal horretara)

Resumen

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Table of contents

1. Introduction

This chapter is the introduction to the Bachelor's Degree Final Project "Runtime verification for spatio-temporal properties overt IoT networks". In this section, the concepts involved in the project are defined. Additionally, the project definition, scope of the project, planification and the product specification and requirements are explained.

1.1. Problem definition

IoT (**Internet of Things**) is the area of computer science that collects the challenges of connecting millions of smart devices and sensors and making them accessible via the internet. This field is growing fast. The forecast is that the number of connected devices by 2030 will be 25.44 billion worldwide [1]. These devices are already part of several fields (e.g., e-health services, smart cities, e-farm, and intelligent transportation systems (ITS)), being a big part of the digitalization of society to build a smart world.

Among the systems that can exploit an IoT infrastructure, a noteworthy category is **Cyber Physical Systems** (CPS), where physical systems are monitored and/or controlled by a computational core. They interact with physical processes through sensors and actuators. The increasing numbers of IoT devices and intelligent systems made CPS influence society. They can be found in different sectors such as self-driving cars, home equipment and medical devices [2] [3]. The following definition is the most famous one for the term "Cyber Physical Systems":

> *"Cyber-Physical Systems are engineering, physical and biological systems whose operations are integrated, monitored, and/or controlled by a computational core. Components are networked at every scale. Computing is deeply embedded into every physical component, possibly even into materials. The computational core is an embedded system, usually demands real-time response, and is most often distributed. The behaviour of a cyber-physical system is a fully-integrated hybridisation of computational (logical) and physical action."*
>
> *(Helen Gill, US National Science Foundation) [4]*

Monitoring is an activity related to the wider category of **Runtime Verification** (RV), which purpose is to observe information from a system while it is operating and analyse the behaviour to detect if it satisfies or violates certain properties. Monitoring the status of a CPS at runtime can give precise information to ensure reliability, safety, robustness and security [5] [6].

This project focuses precisely on the challenges when doing monitoring on CPS over IoT and provides an implementation of a service to monitor data collected by sensors at runtime. It is closely related to some aspects of Helen Gill's definition. The IoT devices are in the physical part where they are spatially distributed and networked. The data will be collected, both across space and time. One main task of the project is to connect the sensors with the monitor so they can

share information (i.e., networking). Finally, this data will be sent to MoonLight to monitor everything in real-time.

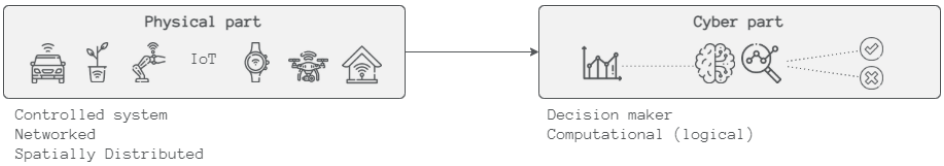


Figure 1-A Project outline

1.2. Objectives

The objective of this project is to feed MoonLight (the provided monitor) with live data. Moonlight previously has been tested with a Matlab interface since this tool has several CPS models and analysis tools available [7]. With that being said, this is the first time for the monitor to receive real data, so the purpose of this project is to help MoonLight be able to monitor different use cases and see how it works with actual real time data.

Therefore, the main objective is to implement a middleware. This middleware will enable the monitor to communicate with different applications in a distributed network. To achieve this, the middleware not only will provide some services but also will be capable expand by adding more services.

To implement this, several and diverse goals must be met. On the one hand, the monitoring of spatio-temporal properties will be carried out using logic-based specification languages, so the language used, STREL, must be comprehended in order to implement it on the use-cases defined in the project. On the other hand, the communication of the sensors with the monitor has to be established, to achieve this, two types of protocols will be used, MQTT and Bluetooth Low Energy.

The sensors are spatially distributed collecting the surrounding data. Using the Internet access, the communication between the physical world and the middleware must be enabled. With this connection, the middleware must be able to monitor the systems behaviour at runtime. Runtime monitoring analyzes the current state of the physical part to detect if it satisfies the specified properties.

Finally, the middleware has to adapt to the different use cases. The middleware is a software which offers and communicates different services. The services that must be present are the monitor, the data collector service and the dashboard. Furthermore, it should be easy to add more services and use cases, making it flexible and accept different upcoming data will add more value to the framework.

1.3. Project phases

The project duration is of eight months, from November 2021 to June 2022. To achieve the objectives, the project has been divided into some tasks and scheduled to manage the work. The development has been divided into the next phases:

1.3.1. Introduction to the project and department

The project is held in Technische Universität Wien (TU Wien), Austria's largest research and educational institution in the field of technology and natural sciences. The group is the Cyber-Physical System department. This phase has consisted of learning the subjects involved in the project and the tools employed during the development. The principal tool used was Moonlight, therefore, most of the time was dedicated to this, to learn the concepts to comprehend it (i.e., CPS and STREL) and how to use it.

1.3.2. Product development

After studying the project bases and making some trial examples with the moonlight framework, the product development started. Starting from the middleware, continuing with the networking and the sensors, and finishing with the layout/framework, the development was carried out during the entire process.

A planification was scheduled to make sure that the objectives were achieved before the deadline. There was a beta release planned for the beginning of April. The beta release consisted in a product that was close in look, feel and function to the final version of the project. This beta release happened one week after the planned date, however, the product had more fancy stuff. The product not only had the main functions of the final product done (i.e., all the flow, beginning in the sensors until the moonlight monitor) but also it was able to fully support SOA (Service Oriented Architecture). The services implemented in the Middleware were the one in charge with the MQTT connections and the Online Moonlight service, able to monitor the input data. Besides, the Data Bus did …

In the resting month HTPP, … wre implemented in the middleware. In terms of hardware, the Thingy52 and the ESP were able to do … (Previously, I had some demo programs with the Thingy)

Some objectives of the project had changed or added from the beginning to the end. During the development of the project, the supervisors proposed new ideas, like the use of the dashboard. At the beginning the objective was to develop the way from the sensors to the monitor, but finally, adding another step, the middleware became bi-directional. It was not only able to take the data and monitor it, but also to get the results and communicate them to the client through the dashboard. To add this and more services to the middleware easily, it has a SOA architecture (Service Oriented Architecture).

1.3.3. Other tasks

- **Tests**: Software testing has a high importance in the process of developing and evaluating a product. It has several benefits like precenting bugs and reducing development costs. Tests

were used for several purposes, to validate that the units perform as expected, to verify the functions work well together and, in some cases, to do a TDD (Test Driven Development) where the test cases were developed before the software. Lastly, with the continuous integration, the tests where automated every time the commits were pushed to the repository.

- **Documentation**: During the entire project the details of the product were recorded in this document.
- **Meetings and more**: Every week I had a meeting with the supervisors to keep a track of the project, make some questions and do or ask for suggestions. Every now and then I had other extra meetings if needed. Besides, in the first months, I was given the opportunity to attend the IoT master's course, that helped me with the sensors and the MQTT protocol.

1.3.4. Gantt Chart

To organize the development, a Gantt chart was created at the beginning of the project, __ is the summarized version of the planification. Some tasks took longer than expected due to the changes in the product and other unexpected issues, nevertheless, the project was successfully completed. As time passed, the track of the project was being recorded, in the chart of ___ appears the detailed actual development of the whole project.

GANTT CHART TXIKITO

1.4. Product requirements

This thesis consists largely in developing a software project, due to this, the main resources used are software, nevertheless, some hardware devices were used too. Down below there is a list of the requirements needed during this project divided into two groups, software and hardware:

1.4.1. Software

Moonlight[1]

This is the principal resource, a lightweight Java-tool monitor, it can monitor temporal, spatial and spatio-temporal properties of distributed complex systems, like Cyber-Physical Systems. It has two different monitoring approaches, offline and online. Finally, supports the specification of properties written with the Reach and Escape Logic (STREL).

GitHub Actions and SonarCloud

GitHub Actions is a continuous integration and continuous delivery platform that allows for automating the build, test and deployment pipeline. This project is in a GitHub repository, so is possible to run a workflow. Every time a push is done, the Middleware is built, tested and analysed in SonarCloud.

[1] GitHub - MoonLightSuite/MoonLight: a light-weight framework for runtime monitoring.

SonarCloud automatically analyzes branches, the code quality and code security. By knowing the bugs, code smells, security hotspots and coverage and the fast feedback it was possible to maintain the code cleaner.

Programming languages

Nearly all the project is developed in Java. As previously mentioned, Moonlight is implemented as a Java program and so is the developed Middleware. The version used is Java 17, the last Long Term Support release.

C is used to develop the program of the devices of this project. A language that can be considered one of the most widely used programming languages in IoT.

Gradle

Gradle is a build automation tool. It is open-source and it is focused on flexibility and performance. Its features include the management of the dependencies.

Zephyr

The Zephyr Project is a scalable open-source real-time operating system (RTOS) supporting multiple hardware architectures (more than 350 boards), between them Thingy52 can be found. The Zephyr projects are CMake-based. CMake is an open-source, cross-platform family of tools designed to build, test and package software.

Integrated Development Environment (IDE)

First, for the middleware development, IntelliJ IDEA was used. This tool provides code completion, static code analysis and refactoring. This makes the development workflow experience smoother.

Then, Visual Studio Code was for the Thingys. This IDE provides just the tools a developer needs for a quick code-build-debug.

Finally, Arduino IDE was also used. This makes it easy to write code and upload it to the board, in this case to the ESP32.

1.4.2. Hardware

Thingy52

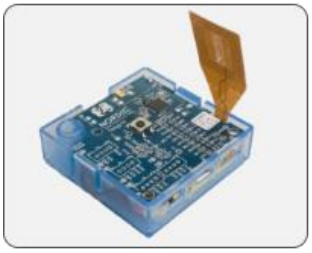
The Nordic Thingy52 board is an easy-to-use prototyping platform. It is designed for helping to build IoT demos, with power optimization and several sensors. It is based on the nRF52832 Bluetooth 5 system on chip (SoC). In this project, the Bluetooth Low Energy (BLE) and environmental sensors such as temperature, humidity and air quality (CO2 and TVOC) are used.

Segger J-Link

SEGGER J-Links are the most widely used debuggers on the market thanks to its many supported CPUs and compatibility with the popular environments. The J-Link EDU Mini is a reduced version to allow students and educational facilities to debug different devices. USB-powered, can communicate at high speed.

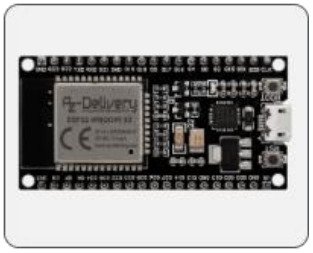
ESP32

ESP32 is low-cost, ultra-low power consumption system on a chip microcontroller. ESP32 can interface with other systems to provide Wi-Fi and Bluetooth functionality through its SPI/SDIO or I2C/UART interfaces. In this project case, uses Bluetooth to communicate with the Thingy sensors and the Wi-Fi to publish it in the MQTT broker.



Nordic Thingy:52

Segger J-Link EDU mini



ESP32

Figure 1-B Hardware

2. State of the art

In the state of the art chapter, the technologies used in the project are analysed to study the current level of development by searching and reading the previous researches done.

2.1. Cyber Physical Systems

CPS plays an important role in Industry 4.0. The concept of Fourth Industrial Revolution was introduced in 2010 by the German government. This is an interpretation of Klaus Schwab, the executive chairman of the Word Economic Forum (WEF): *"Now a Fourth Industrial Revolution is building on the Third, the digital revolution that has been occurring since the middle of the last century. It is characterized by a fusion of technologies that is blurring the lines between the physical, digital, and biological spheres."*[8]. The mechanical systems by their own are not relevant anymore, that is why CPS is widely applied in various fields. By providing the physical objects with computing and communication capabilities, they can turn into intelligent objects and surpass the previous systems. CPS can learn from the physical world and in some cases even interact with the world, turning environments into Smart Environments [9] [10]. In conclusion, CPS are deeply influencing the society and reshaping the perception and interaction with the physical world [3].

2.2. Monitors

CPSs are susceptible to faults so there is a need to detect the status of the systems. Monitoring is an activity to observe systems behaviour, and enabling the runtime verification (RV) techniques, monitors can track systems while they are operating.

As previously mentioned, the monitor used in this project is **Moonlight**. Moonlight is a software monitor solution, in the next table some software monitors are presented:

Table 2-A Software monitors for RV

Monitor	Specification Languages	Programming Language
Moonlight	STREL (Spatio-Temporal Reach and Escape Logic)	Java
RV-Monitor	LTL (Lineal Temporal Logic)	Java
jSSTL	SSTL (Signal Spatio-Temporal Logic)	Java
Kieker	-	Java

- Moonlight: This monitor is a java-tool for monitoring properties of distributed complex systems.

- RV-MONITOR: This monitor enables the runtime verification, in this case, monitors are integrated into the original system to check its behaviour during execution [11].
- jSSTL: Is a java-based tool to monitor trajectories coming from simulations or from measurements of real systems. But it is an offline monitoring, so it does not support RV [12].
- Kieker: This framework is able to monitor at runtime. It allows developers to replace or add framework components [11].

Moonlight and its interfaces are still under development; however, it is sufficiently qualified to apply it in real environments, and therefore in this project.

Comparing the monitors previously mentioned, Moonlight is the most suitable one for this project. In the first place, it can be used for distributed systems, which means that it can connect with various components. Afterwards, it supports online monitoring, and the state of the system can be observed at runtime. Finally, with the STREL specification language, it can monitor the system both across space and time.

In conclusion, Moonlight has the desired features to monitor spatio-temporal properties of distributed systems. The other monitors have some interesting attributes, but they lack of some elements that are necessary for this project (e.g., jSSTL does not support online monitoring).

2.3. Spatio-Temporal Logics

The monitors should include a language to specify the requirements needed to monitor the system efficiently. Signal Temporal Logic (STL) is widely used for analyzing programs in CPSs. However, STL is not expressive enough for real world cases, because they do not only require temporal information but also spatial needs. Most specification languages and tools available for CPS supports only the monitoring of temporal properties (e.g., MTL, STL and TRE). Thereby, there are some existing spatial extensions of STL (e.g., SSTL, SpaTeL and STREL) [13].

In the following table there is a comparison of the specification coverage on 1000 quantitatively-specified real city requirements between STL, SSTL, STREL and SaSTL:

Table 2-B Number of specification coverage on 1000 requirements [13]

	SaSTL	**STREL**	**SSTL**	**STL**
Number of covered requirements	950	431	431	184
Covered key elements	Temporal & Spatial Range & Aggregation, Counting, Percentage	Temporal& Spatial Range	Temporal& Spatial Range	Temporal

As we can see in the Table 2-B Number of specification coverage on 1000 requirements the specification language with the most coverage is SaSTL, nevertheless, STREL, the language used with Moonlight, is expressive enough to cover the most important requirements.

In spite of that, STREL is more convenient for the following reasons:

- STREL can handle online monitoring, this solution is preferable because it permits to take immediate action during execution and is computationally cheaper [14].
- STREL can handle mobile or dynamic CPS. In SSTL the topology is assumed to be static, it is impossible to monitor nodes changing locations [15].

In Appendix A **STREL** there is more information about STREL and how to express the spatio-temporal behaviours in this specification language.

2.4. Middleware

A distributed system is composed of several hardware and software elements which must be integrated, the element that facilitates the management of such environments is the middleware. For example, IoT requires integrating and working with data and different algorithms distributed in other applications; as well as operating in real time with a wide variety of processes.

Distributed systems are not only applicable to computers and physical components, but also a logical dimension. A distributed system can be defined as a set of independent processes or applications that interact with each other. The middleware is in charge of enabling the communication and data management of the distributed applications [16].

The middleware developed for this project communicates different services with each other in real time; these services can be found locally or distributed.

2.5. Communication protocols

A communication protocol is required to exchange messages between computing systems. These are the communication standards used in the project:

2.5.1. MQTT

2.5.2. Bluetooth Low Energy

2.5.3. Wi-Fi

3. Development

3.1. Monitor

Moonlight online approach

Monitoring tools for CPS are generally agnostic of the spatial configuration of the entities such as sensors and computational units.[Moonlight document]

In this section a novel online (out-of-order) monitoring algorithm for STREL is presented. Differently from the standard offline approach, where all the data is available at the beginning of the execution, online monitoring is performed incrementally, when a new piece of data is available. In this case, the uncertainty related to the absence of information must be taken into account. For that aim, the machinery of imprecise signals can be exploited to represent the uncertainty, where the result of the monitoring process, whether it is a satisfaction or a robustness signal, is refined as soon as new updates of the input arrive [Online Monitoring of Spatio-Temporal Properties for Imprecise Signals]

we introduced an online monitoring algorithm for STREL that exploits imprecise signals that can be refined by updates arriving in any order, and that can monitor updates on different locations in parallel. We implemented the proposed methodology in the Moonlight monitoring tool.

Rather than replicate a piece of hardware or a piece of software, another approach to error detection is dynamic or runtime verification. Dynamic or runtime verification is 26 2.5. Runtime Verification, Enforcement & Adaptation performed during the execution of software, at runtime, and dynamically checks its behaviour. Runtime Verification (RV) consist of extracting information from a running system and using it to detect, and possibly react to, observed behaviors satisfying or violating certain properties [DGH+16]. Runtime verification avoids the complexity of traditional formal verification techniques (like model checking and theorem proving) by analyzing only one or a few execution traces and working directly with the actual system. (MIREN)

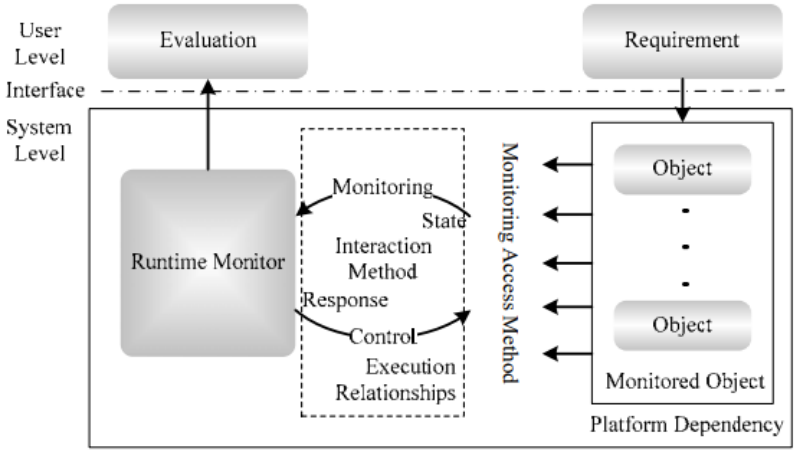


Figure 1. Software Runtime Monitoring Model

Figura: A Generic Software Monitoring Model and Features Analysis

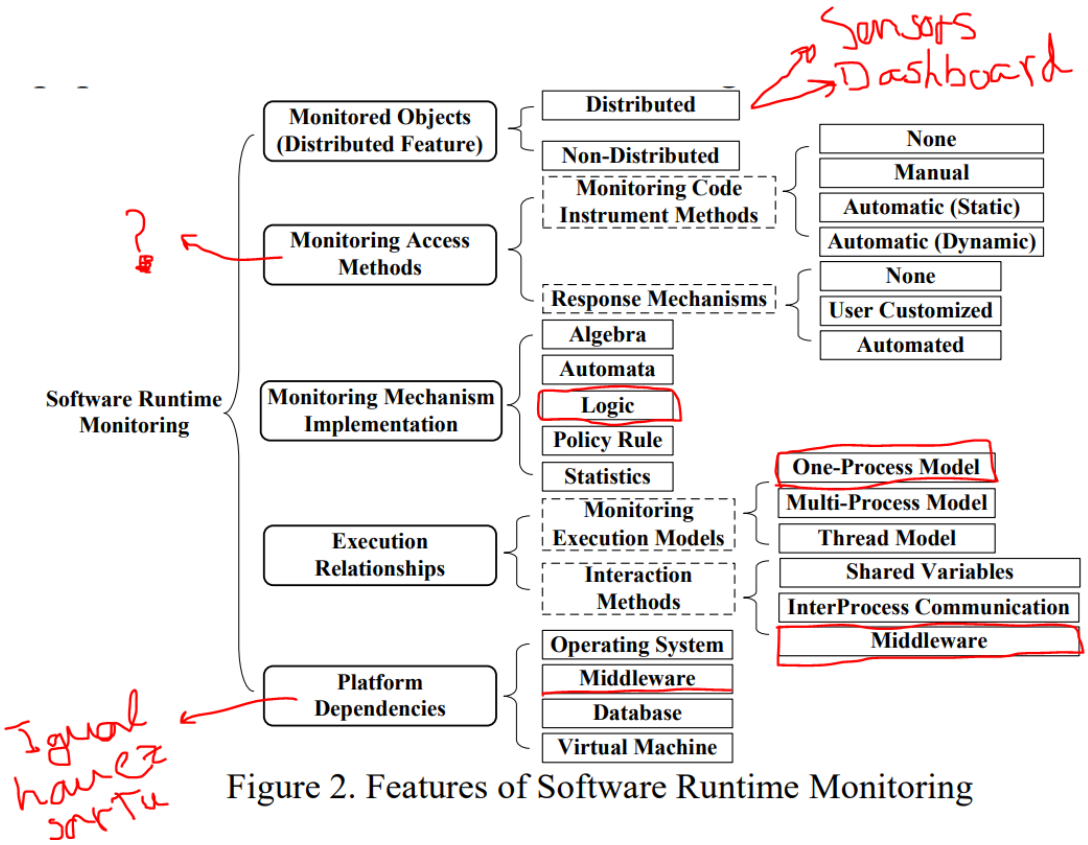
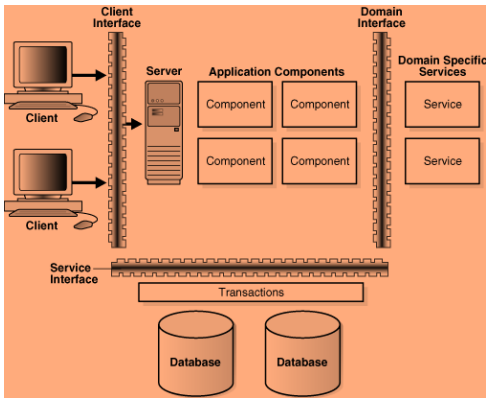


Figure 2. Features of Software Runtime Monitoring

A Generic Software Monitoring Model and Features Analysis

3.2. Middleware



Egin horrelako zerbait nire adibidea erabiliz

https://docs.oracle.com/cd/E21764_01/core.1111/e10103/intro.htm#ASCON110

The CPS must be able to overcome the system uncertainty, scalable and tolerant to threats

3.3. Services SOA

Luckily, the middleware I was developing was as general as possible and the tests done had a good coverage, so to change the architecture of the software hadn't take so long and I was able to extend the project.

3.4. Design pattern

Builder → https://refactoring.guru/

3.5. Buffer

Collecting binary data bits into groups that can then be operated on as a unit,

automatic buffering.

It helps devices to manipulate data before sending or receiving.

3.6. Robustness

Error handling

Maintainability

3.7. Kodea egiteko sistemak / Pattern

Again "Visitor Pattern Considered Pointless - Use Pattern Switches Instead"-I buruz hitz egin eta Java 17. Again Duplikazioen eta polymorfismoari buruz hitz egin

3.8. Physical system

3.8.1. Thingy

Kconfig Json importatu ahal izateko → zephyrrena

CMakeList

Prj.conf → sensoreak enable egin ahal izateko

https://github.com/google/eddystone/blob/master/protocol-specification.md

3.8.2. ESP 01

3.9. MQTT

DR1 Lightweight communication methods

DR2 Interoperability.

DR3 Non-blocking event propagation. Events may arrive at unknown rates

DR4 Scalability(??)

Edge-based Runtime Verification for the Internet of Things

3.10. BLE

3.11. WIFI

4. Problems and solutions

How to manage the time and the buffer

Overriding problems, how to save them

Dealing with missing values / Imprecise Signals

1. The very first values

   Discard it directly

2. The missing values during the program

   Time Chain splitter

Coordinate the sensors

Moonlight is prepared to monitor starting from the time 0

Time table to coordinate the sensors

Global reference time is one of the important architecture criteria in CPS components. The global reference time helps to ensure the real-time communication performance are achived. Global reference time is the basis CPS component to confirm the communication between physical and cyber world will work properly. CPS able to conduct a synchronous or asynchronous interaction with the physical world. Cyber-Physical System (CPS) State of the Art

Duplicated values from sensors -Maybe DELETE-

Other minor problems

- MoonlightRecord had some problems: Escape + online monitor + MoonlightRecord = infinite loop

  The null values didn't throw errors, wrong error handling.

  I reported this issues -> Ennio created Tuple.class, that had everything MoonlightRecord was offering but with these errors fixed: I just used Tuple from that moment on.

- I was having problems with Windows + Zephyr project -> I used Linux for the coding of the sensors

  for the Thingy sensors, I used a Linux virtual machine. IntelliJ IDEA does not support C/C++ officially and it requires quite a lot memory, that's why I used Visual Studio Code.

5. Use cases

5.1. Office use case

5.2. Wiener linien use case (?¿?¿?¿?¿?¿?¿)

6. Economic memory

The majority of the cost of a software project is in long-term maintenance. [clean code liburua]

Intellij IDEA: unibertsitateko kontuari esker dohainik

7. Conclusions and future lines

This is thechnical

7.1. Conclusions

a. Reflexiones técnicas: relacionadas con los objetivos del proyecto b. Reflexión sobre las implicaciones sociales, de salud y seguridad, medioambientales, económicas e industriales c. Reflexión sobre la aplicación de conocimientos relativos a cuestiones económicas, organizativos de gestión (gestión del riesgo y del cambio) en el contexto industrial y comercial.

7.2. Future lines

“Smart Home Automation System Using on IoT” dokumentuan rosas dagoenari begirada bat bota /!\

Legal aspects: General Data Protection Regulation (GDPR):

Error processing: Maybe there are things that are assumed that can fail and they are not handled, for example, the times of the sensors are always ascendant.

8. Personal evaluation of the experience(?) and the project

Proiektua egiten nola sentitu naizen aipatu

Esperientziari dagokionez: A) Unibertsitatea: nola sentitu naizen, IoTko kurtsoak, astero egiten diren hitzaldietara joaten utzi, liburua utzi irakurteko… B) Beste herrialde batera joan: Leku berriak ezagutu, bertoko kulturatik ikasi, bakarrik bizitzea eta independentzia.

9. Sarrera, ondorioak eta etorkizuneko ildoak

Atal honetan sarrera, ondorioak eta etorkizuneko ildoak atalen laburpen bat egingo da euskaraz.

9.1. Sarrera

9.2. Ondorioak

9.3. Etorkizuneko ildoak

10. Appendix A

STREL

Titulua aldatzerako orduan kontuz! Formatua galdu gabe/!\ Aurkibidean polit ikusteko

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Tabla A1 …

11. Appendix B

MQTT

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

12. Appendix C

REST

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

13. Appendix D

Data bus

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

14. Appendix E

Gantt chart

15. Bibliography

[1] T. Insights, «Statista,» December 2020. [En línea]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[2] L. Nenzi, E. Bartocci, L. Bortolussi, M. Loreti y E. Visconti, «Monitoring Spatio-Temporal Properties (Invited Tutorial),» de *Runtime Verification*, Springer International Publishing, 2020.

[3] D. Ratasich, F. Khalid, F. Geissler, R. Grosu, M. Shafique y E. Bartocci, «A Roadmap Toward the Resilient Internet of Things for Cyber-Physical Systems,» *IEEE Access,* vol. 7, pp. 13260-13283, 2019.

[4] H. Gill, US National Science Foundation, 2006.

[5] C. Tsigkanos, M. M. Bersani, P. A. Frangoudis y S. Dustdar, «Edge-based Runtime Verification for the Internet of Things,» *IEEE Transactions on Services Computing,* 2021.

[6] M. Illarramendi, L. Etxeberria, X. Elkorobarrutia, J. Perez, F. Larrinaga y G. Sagardui, «MDE based IoT Service to enhance the safety of controllers at runtime,» 2019.

[7] E. Bartocci, L. Bortolussi, M. Loreti, L. Nenzi y S. Silvetti, «MoonLight: A Lightweight Tool for Monitoring Spatio-Temporal Properties,» *ArXiv,* vol. abs/2104.14333, 2020.

[8] K. Schwab, «The Fourth Industrial Revolution: what it means, how to respond,» World Economic Forum, 2016. [En línea]. Available: https://www.weforum.org/agenda/2016/01/the-fourth-industrial-revolution-what-it-means-and-how-to-respond/.

[9] J. Jamaludin y J. M. Rohani, «Cyber-Physical System (CPS): State of the Art,» *2018 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube),* 2018.

[10] G. Garatu, «Que son los Sistemas Ciber-Físicos (CPS) en la nueva Industria 4.0,» 2017. [En línea]. Available: https://grupogaratu.com/que-son-sistemas-ciber-fisicos-cps/.

[11] M. Illarramendi, «Runtime Observable and Adaptable UML State Machine-Based Software Components Generation and Verification: Models@Run.Time Approach,» 2019.

[12] L. Nenzi, L. Bortolussi y M. Loreti, «jSSTL - A Tool to Monitor Spatio-Temporal Properties,» 2017.

[13] M. Ma, E. Bartocci, E. Lifland, J. Stankovic y L. Feng, «SaSTL: Spatial Aggregation Signal Temporal Logic for Runtime Monitoring in Smart Cities,» 2020.

[14] E. Visconti, E. Bartocci, M. Loreti y L. Nenzi, «Online Monitoring of Spatio-Temporal Properties for Imprecise Signals,» 2021.

[15] E. Bartocci, L. Bortolussi, M. Loreti y L. Nenzi, «Monitoring mobile and spatially distributed cyber-physical systems,» 2017.

[16] A. Rivas, A. Pérez García Osvaldo y J. Hernández-Palancar, «Estado del arte sobre aplicaciones del middleware ROS,» 2016.