

DEGREE PROGRAMME IN ELECTRICAL ENGINEERING

MASTER’S THESIS

A CYBER-PHYSICAL SYSTEM WITH A MIXED REALITY INTERFACE

Author Johan Juntunen

Supervisor Timo Ojala Second Examiner Matti Hämäläinen Technical advisor Vesa Pentikäinen

June 2016

**Juntunen J. (2016) A cyber-physical system with a mixed reality interface.** University of Oulu, Department of Electrical Engineering, Degree Programme in Electrical Engineering. Master’s Thesis, 76 p.

**ABSTRACT**

**This thesis presents a cyber physical system that mirrors and augments real world devices and their functionality in a virtual representation of the real world. The energy characteristics of real world devices are measured and presented in the virtual reality interface in real-time. The virtual reality interface augments the functionality of the real world system with virtual inputs and outputs. The system is designed and implemented using a multi-agent software model. The challenge of keeping real and virtual worlds synchronous and consistent is solved by introducing a synchronisation block.**

**The synchronisation and consistency of the real and virtual worlds were evaluated from a technical perspective. Synchronisation was evaluated with timing measurements. They revealed that the system operated most of the time without breaking timing conditions. Still, there were at times larger delays which showed that the system was not synchronous all the time. Consistency was verified with empirical measurements of visually observable states. The responses of the system to the user actions were mainly consistent, although a malfunction leading to inconsistency between the states of virtual and real devices was found.**

**The results demonstrate that the real world devices and their corresponding virtual representations are controlled synchronously and their visually observable states remain consistent. The thesis shows that it is feasible to augment a cyber-physical system with virtual objects, for example virtual sensors, which operate and interact synchronously with the real world system. The thesis concludes with a discussion on findings and future work.**

**Key words: cyber-physical system, mixed reality, virtual reality, synchronicity, consistency, user interaction, energy measurement.**

**Juntunen J. (2016) Kyberfysikaalinen järjestelmä sekoitetun todellisuuden rajapinnassa.** Oulun yliopisto, sähkötekniikan osasto, sähkötekniikan opintosuunta. Diplomityö, 76 s.

**TIIVISTELMÄ**

**Työssä esitetään kyberfysikaalinen järjestelmä, joka peilaa ja täydentää reaalimaailman laitteet ja niiden toiminnallisuuden reaalimaailman virtuaalisessa esityksessä. Reaalimaailman laitteiden energiatasoja mitataan ja ne esitetään virtuaalitodellisuuden rajapinnassa reaaliaikaisesti. Virtuaaliset sisään- ja ulostulot virtuaalitodellisuuden rajapinnassa täydentävät reaalimaailman järjestelmän toiminnallisuutta. Järjestelmä suunnitellaan ja toteutetaan moniagentti- ohjelmistomallia käyttäen. Reaali- ja virtuaalimaailmojen synkronisuuden ja yhdenmukaisuuden takaamisen tuoma haaste ratkaistaan työssä esitettävän synkronointilohkon avulla.**

**Reaali- ja virtuaalimaailman synkronisuuden ja yhdenmukaisuuden arviointi tehtiin teknisestä näkökulmasta. Synkronointi arvioitiin ajoitusmittausten perusteella. Mittausten mukaan järjestelmä toimi pääsääntöisesti ajoitusvaatimusten puitteissa. Ajoittaisia ajoitusvaatimuksia rikkovia viiveitä kuitenkin esiintyi. Järjestelmän yhdenmukaisuus arvioitiin empiirisesti havainnoimalla järjestelmän silminhavaittavia tiloja. Järjestelmän vaste käyttäjän toimiin oli yhdenmukaista lukuunottamatta toimintahäiriötä, joka tietyissä olosuhteissa johti todellisuuden ja virtuaalimaailman esityksen epäyhdenmukaisuuteen.**

**Tulokset osoittavat, että reaalimaailman laitteiden ja niitä vastaavien virtuaalisten esitysten ohjaus on synkronissa ja silminhavaittavat tilat pysyvät yhdenmukaisina. Työ osoittaa, että on mahdollista täydentää kyberfysikaalista järjestelmää virtuaalisilla objekteilla, esimerkiksi virtuaalisilla sensoreilla, jotka toimivat ja vuorovaikuttavat synkronisesti reaalimaailman järjestelmän kanssa. Työn lopuksi käsitellään esiinnousseita havaintoja ja järjestelmän parannusehdotuksia.**

**Avainsanat: kyberfysikaalinen järjestelmä, sekoitettu todellisuus, virtuaalitodellisuus, synkronisuus, yhdenmukaisuus, käyttäjävuorovaikutus, energian mittaus.**

**TABLE OF CONTENTS**

ABSTRACT TIIVISTELMÄ

TABLE OF CONTENTS ACKNOWLEDGEMENT FOREWORD

LIST OF ABBREVIATIONS AND SYMBOLS

1. [INTRODUCTION 9](#_bookmark0)
2. [THEORY 10](#_bookmark1)
   1. [Cyber-physical Systems 10](#_bookmark2)
   2. [Mixed Reality and Virtuality 10](#_bookmark3)
   3. [Multi-agent Systems 12](#_bookmark5)
   4. [Prior work on using virtual reality in cyber-physical systems 15](#_bookmark6)
3. [DESIGN 18](#_TOC_250004)
   1. [Use Case and Scenarios 18](#_TOC_250003)
   2. [Elicitation of Requirements 19](#_bookmark8)
   3. [System Design 22](#_bookmark9)
      1. [Design patterns and frameworks 22](#_bookmark10)
      2. [Overview of the logical organization 24](#_TOC_250002)
      3. [Subsystem decomposition 24](#_TOC_250001)
      4. [Software and hardware mapping 24](#_TOC_250000)
      5. [Subsystem designs 25](#_bookmark13)
   4. [Interface Design 34](#_bookmark21)
   5. [Analysis of Design 35](#_bookmark22)
4. [IMPLEMENTATION 36](#_bookmark23)
   1. [Cyber-physical system 36](#_bookmark25)
   2. [VirtualWorldModel 43](#_bookmark31)
      1. [VirtualWorldModel 43](#_bookmark33)
      2. [VirtualWorldModelOrganisationRuler 44](#_bookmark35)
      3. [MessageBroker 45](#_bookmark37)
      4. [ServiceMarket 46](#_bookmark39)
      5. [VirtualWorldCommunicator 47](#_bookmark41)
      6. [FederalDelegate 48](#_bookmark44)
      7. [Devices 49](#_bookmark47)
      8. [Interfaces 54](#_bookmark57)
      9. [AsynchronousReceiver 56](#_bookmark62)
   3. [VIRTUALWORLD 57](#_bookmark65)
   4. [SYNCHRONISER 62](#_bookmark75)
5. [EVALUATION 66](#_bookmark79)
6. [DISCUSSION 69](#_bookmark82)
7. [CONCLUSIONS 71](#_bookmark84)
8. [REFERENCES 72](#_bookmark85)

**ACKNOWLEDGEMENT**

This research has been supported by a grant from Tekes – the Finnish Funding Agency for Innovation as part of 'The Naked Approach – Nordic perspective to gadget-free hyperconnected environments' programme.

**FOREWORD**

This thesis was an interesting exploration at the border of real and virtual worlds. It included lots of work but was definitely worth it. It is time to mention those who I think were meaningful and helpful during this process. First and foremost, I give thanks to Jesus Who gave me the strength and understanding needed to complete this thesis.

This master’s thesis was done at VTT Technical Research Centre of Finland in Oulu. I would like to thank VTT and my manager Vesa Pentikäinen for the opportunity for this thesis. I want to thank Mr. Pentikäinen for the continuous support and creative ideas during this thesis. Warm thanks to Samuli Yrjänä who designed the electronics part used in this thesis and had always time for guidance when needed. I would like to thank Petri Myöhänen for the help with virtual environment related issues, and Juho Eskeli and Marko Korkalainen for their support of Bluetooth related questions.

I would like to present my thanks to my supervisor Timo Ojala for the smart comments, constructive criticism and guidance in this thesis. Many thanks to the second examiner Matti Hämäläinen for the feedback. I would like to thank the personnel at University of Oulu. Especially, I am grateful for the support and guidance from the teaching personnel and the study advisors.

I would like to thank my family for their support and encouragement. Many thanks to my friends with whom I have had a good time so many times this far.

Oulu June 1st, 2016

Johan Juntunen

**LIST OF ABBREVIATIONS AND SYMBOLS**

ACMAS Agent Centered Multi-Agent Systems, a set of design principles to design multi-agent systems in terms of mental states of agents [1, p. 1]

AR Augmented Reality

AV Augmented Virtuality

CPS Cyber-Physical Systems

CSCW Computer Supported Collaborative Work GORMAS Guidelines for Organizational Multi-Agent Systems

GPIO General Purpose Input/Output, a generic pin in an integrated circuit which behaviour is programmable

I2C a serial computer bus

IO Input/Output

MRA Mixed reality architecture, an interconnection between offices in mixed reality

O-MaSE Organization-based Multiagent Systems Engineering, a design framework for multi-agent systems

OCMAS Organization Centered Multi-Agent Systems, “A set of principles to design multi-agent systems based on organizations.” [2]

OperA Organizations per Agents, a design framework for multi-agent systems

UART Universal Asynchronous Receiver/Transmitter, a hardware device which transmits and receives data in serial form and does the conversion between serial and parallel data forms

UDP User Datagram Protocol, a connectionless transmission protocol USB Universal Serial Bus

VTT VTT Technical Research Centre of Finland Ltd

# INTRODUCTION

This thesis presents a way of designing and implementing a cyber-physical system so that it is coupled with a virtual world. The virtual world provides a virtual representation of the real world devices and their state, and augments the system with virtual elements that do not have direct real world counterparts. Controllers and controlled elements co-exist in real and virtual realms and control flows across the realms that remain consistent with each other. Thus, jointly, the real and virtual components constitute a cyber-physical system with a mixed reality interface.

Ricci et al. [3, p. 60] argue that it is not easy to create a set of concepts upon which to model open smart environments with human interaction. Still, as an incentive to develop such environments, they state that ambient intelligence in these environments will be an active helper for user in her actions. It is argued the use of virtual world representation should bring some added value to justify its use. Virtual environments give a possibility of broadening the perspective: users can interact with and observe the system both in real world and virtual world. Virtual environments may offer a space, where we can interact, that is isolated from real-world dynamics or communication patterns that impede positive, productive interactions [4, p. 64]. Sometimes it is necessary to bring forth some added value with virtual world; otherwise the user may question the purpose of using it.

In this thesis the focus is on mirroring and augmenting a cyber-physical system with a virtual world so that the elements in the two realms would be synchronous and consistent with each other. These goals for the system’s functionality are evaluated from a purely technical perspective with empirical measurements. Thus, assessing user interaction and experience with the system are outside the scope of the thesis. The system is implemented as a multi-agent software system on the virtual world side which co-exists and communicates with the physical setup existing in the real world.

This thesis is organised as follows. Chapter 2 reviews key literature on cyber- physical systems, mixed and virtual reality, and multi-agent systems. Chapter 3 presents the design of the system. Chapter 4 describes the implementation of the system as a multi-agent software system. Chapter 5 reports the empirical evaluation of the system. Chapter 6 discusses the findings of the work and Chapter 7 concludes the thesis.

# THEORY

The theoretical framework for this thesis comprises of cyber-physical systems, mixed reality and virtuality, and multi-agent systems.

## Cyber-physical Systems

Cyber-physical systems (CPS) consist of computing elements, elements controlled by these computing elements, their communications and the design of how these elements co-exist and operate [3, p. 88], [4, p. 731], [5, p. 363]. The computation has a tight contact with physical processes [6, p. 308]. Cyber-physical systems are used in areas like manufacturing, robotics, transportation, defence, healthcare and smart energy grids [7, p. 37], [8, p. 450].

There is room for theories telling how to design these kind of systems [3, p. 89]. From the design point of view, timing and stability are important CPS- related challenges to be handled.

In CPS physical and computational elements are tightly consolidated and embedded system is extended to computing system [9, p. 184]. CPS combine continuous dynamics of physical space and discrete dynamics of cyber space. Ensuring real-time and predictable communication in CPS within and between these two spaces with different characteristics requires a careful allocation of the communication resource between the components. Non-deterministic delays in communication can lead to instability in control-loops. [10, p. 108] While the open loop architectures have a chance of letting a small error escalate into a system fault, fortunately feedback control provides stability in real physical systems. Thus, in cyber-physical systems all the control loops must be closed so that the system can stand uncertainties, failures etc. (Sha et al., 2009, p. 9)

As can be seen, timing and stability are closely related subjects. A problem related to designing CPS is how “to guarantee the stability and performance in the presence of timing delays induced due to CPS components” (Balasubramaniyan et al., 2016, p. 39). As a guideline, program modelling must address timing and spatial information as the program interacts with real environment. Namely, if timing and concurrency are not taken into account in computing and networking abstractions, the technical progress is delayed, so Lee [11, p. 1]. The physical properties and restrictions related to this environment should be incorporated in the model, too. [12, p. 7]

## Mixed Reality and Virtuality

While the design and operation of CPS alone is challenging due to temporal imperfections in the components of the system [7, p. 37], things are not going any easier when also mixed reality is considered. There are different concepts of reality in computer science. There is the physical world we live in, the reality. On the other end there are artificially created environments, virtual realities. The continuum between these realities is often divided into Reality – Augmented Reality – Mixed Reality – Augmented Virtuality – Virtuality ([Figure 1](#_bookmark4)). Augmented Reality brings computer-generated content to the reality [13, p. 20], [14, p. 1], [15, p. 3]. Augmented Virtuality brings actual data from real world into virtual environment.

Mixed reality covers the continuum from augmented reality (AR) to augmented virtuality (AV). [14, p. 1]

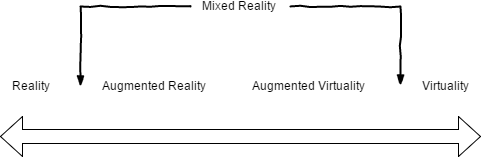


Figure 1. Reality-Virtuality Continuum.

Computer-aided mixed reality has been researched over couple of decades and has reached a firm ground among international research before the millennium [16, p. 5]. It is argued that virtual world extends the capabilities of real devices. Mixed environments enable giving immediate feedback to the user according to her interaction with the environment, which is not possible without the mixed interface between real and virtual environment [16, pp. 10–11]. The feedback considered here should not be mixed with the control feedback. Here the user feedback means an understandable assessment response to a user action. The feedback can be an important factor when a user tries to learn to use a real device. Moreover, if the learning curve of the user is collected and analysed by smart environments, the results from the learning process may help in enhancing the device or the user instructions. Thus it is suggested that mixed reality does not replace the real world but involves it within a broader context by recognising the existence of virtually created content and bringing real and virtual worlds together. If the user can act both in real and virtual worlds, the ways of user involvement in the system are broadened.

Computers are used to support collaboration among people by enabling persons in different locations to come in contact. However, Billinghurst and Kato [17] name two aspects this kind of Computer Supported Collaborative Work (CSCW) lacks: seamlessness and reality enhancement.

Seamlessness is disturbed in switches from different functional workspaces to the others, as user has to endure changes in context and mode of operation. Let us take as an example a group of persons holding a video meeting where everyone is able to see each other’s face through the screens of their communication devices. There is also a presentation view in the video meeting application which allows one person at a time to present their slideshows. A context change happens when a person watches the slideshow and then looks at the video stream showing the participant: those are separate streams of data and thus in different contexts.

Collaboration in mixed reality should enhance the reality: it is not enough to create the same experience as with face-to-face communication in real world[17]. Mixed reality implementations can bring for example graphical and audiovisual experience to the real world, and convey the audio data from reality to virtual world. The investigation by Jung and Hughes [18, p. 22,29] supported the hypothesis that if the visual and information and physical actions correlate, a user will have an

improved sense of telepresence, “an aroused sensation of ‘being together in the same real location’ between users”.

User involvement is affected by how well the user feels being able to communicate with and observe the system. Physical presence is defined as a stronger sense of ‘being there’ in the virtual space, a sensation similar to what individuals experience when being in a specific location in the real world [19, p. 1]. Reaching a physical presence for user is a challenging task in mixed reality systems and this thesis does not try to reach that. This thesis concentrates on designing and implementing a demonstrative mixed reality implementation where a user can trigger real and virtual sensors and observe that triggering these sensors leads to actions that happen simultaneously to the human eye in both domains and the actions can be observed of being visually consistent. The relation between the controller objects executing the actions on the controlled ones is interesting. Control-Display Congruence Continuum presented by Milgram and Colquhoun [20, pp. 14–16] classifies the relation between control and the controlled device by analysing how direct and imminent the effect of control is on controlled object.

Ricci et al. [21, p. 61] explain a mirror world where virtual entities can affect virtual objects, and if an object has a counterpart in real world, it affects also this real object the same way. Bridges et al. [22, p. 7] make a distinction between virtual and mirror world. The former is purely a simulated world whereas the latter is a virtual model of a physical space. The virtual world portion of this thesis is a mirror world of the real world test environment augmented with some additional virtual devices. Thus in this context, the terms are not clearly separated: on the contrary, they are handled as interleaved concepts at the real world – virtual world junction.

Bridges et al. [22, p. 11] also point out the possibility of technology revealing invisible or implicit information with the help of mirror worlds. The measuring the energy in physical world and presenting the measured energy values visually in virtual world demonstrates this possibility in this thesis. Additionally, user can observe the relationship between the illuminance level near the photovoltaic cells and the energy they provide as also the illuminance level is visualised in the virtual environment. The designed cyber-physical system could be also extended to present visually the communication flows between the devices, but this was not included in this thesis.

In this work users can interact with and observe the system both in real world and virtual world. This approach links the thesis subject to the concept of Metaverse. Metaverse is not a simply-defined term, but it is about bringing real and virtual worlds together and fusing their borders. Users can interact with the system whether they are in real or virtual world side of the system. [22, p. 4] User role in virtual world is emphasised and user is provided with extended capabilities [22, p. 6].

## Multi-agent Systems

Software agents have been suggested as building blocks for virtual environments which augment the reality (Croatti and Ricci, 2015, p. 5). Software agent (hereinafter referred to as agent) is “a software entity which functions continuously and autonomously in a particular environment, often inhabited by other agents and processes”. Agents can be defined by different properties of which the following are particularly relevant to this thesis [23, p. 8]:

* reactivity,
* autonomy,
* collaborative behaviour, and
* inferential capability.

A reactive agent needs stimuli to make actions. The opposite of the reactivity is intentionality, according to which agents think by themselves and initiate actions without stimulus. As the real network in this thesis operates reactively, it is a logical choice to make the virtual world act reactively. However, as virtual world is augmented with additional devices, there is a possibility for conflict situations between the states of reality and virtual reality, which need to be taken into account in design.

Autonomy of agent can be considered from different viewpoints. Agent can have autonomy over the way it executes its tasks, while its relations to the surroundings may be defined and restricted by an external entity. Social autonomy emphasizes the independency of agent in deciding how much and when to communicate with the other agents, while absolutely autonomous agent is not even aware of other agents and does anything it can.

Autonomy regarding the design of agent considers the extensibility of the agent system and the autonomy over the interfaces. Extensibility depends on which restrictions the system imposes on the system design, so how much a new agent construct must be modified to confirm to the requirements of the system. Interfacial autonomy considers the amount implementing the interfaces affects the internal design of an agent. In practise the autonomy is a mixture of different types of autonomy. It is pointed out that different agents have different appearances of autonomy relative to different agents. [24, p. 3]

Collaborative behaviour between agents can be discretionary or cooperative. In discretionary collaboration agents have their own goals and they coordinate in tasks which help them reach their individual goals or in also other tasks if they happen to decide so. Similarly, Hayzelden et al. [25, p. 10] make a distinction between self- interested agents and co-operative agents. The former are acting for their own profit and do not consider the global good, while the latter are ready to lose something meaningful to themselves for the sake of the common benefit. In cooperative collaboration the agents share the same goals and the problem solving is distributed among the agents. [26, p. 145] A natural part of problem solving is inferential capability, which means the ability of being able to act on task which requires using a priori knowledge or knowledge fetched from elsewhere along with the information given along with the task [23, p. 8].

From the perspective of agent the individuality reflects its relation to collaboration. Individuality is the degree of agency mentioned by Hayzelden et al. [25, p. 9], where the range of agency goes from weak to strong. The weak agency is used when agents concentrate on task execution and delegation, and when hardware independence is needed. Agents with strong agency use their mental state as the basis for their decisioning. In this thesis the agents have a weak agency, because the objective the agents system tries to fulfil is to enable running coexisting real and virtual worlds synchronically and coherently.

Fidelity means the level how much real and virtual environments are indistinguishable [27, p. 524]. Physical fidelity means how well the mapping between components, objects and properties(looks, physical models, etc.) of the worlds matches. Functional fidelity tells to which extent the the system in simulated

environment acts like the system in physical environment. In this thesis the focus is on the functional fidelity, as the consistency of the system response in physical and virtual worlds is of special interest.

The systems in which tasks are distributed to different agents, also the delivery of resources is considered. In small networks one resource manager might suffice, but scaling up the network leads to a bottleneck situation if only one agent is responsible for resource managing. One solution can be found among multi-agent systems which are distributed, but differ from traditional distributed systems in that they are constructed from intelligent entities, and are itself intelligent [28, p. 1]. Intelligent agent is a computer system which is responsive and social with the environment and proactively acts for its own goals [29, pp. 4–5]. Resource management can be taken care of by intelligent agents.

Agents can request resources from the agents providing the resources by expressing their intentions. The agents can be connected in a distributed manner which helps to convey the intention to be planned and executed at lower levels. [23,

p. 13] These lower levels can be for example isolated groups of agents specialising in executing a single task, like displays showing some graphics. One way to conduct the communication between these isolated groups is through a delegate which delivers the intentions from outside of the group to the agents sharing similar functionalities.

Problem-solving in real world can be mapped into problem-solvingin agent’s model of the real world. Agent tries to solve the problem in its model and maps the solution back to the real world [30, p. 33]. If an agent has a way of solving certain problems, it can be useful to describe the problem with the definitions typical to the context of that agent, rather than trying to modify the problem solving algorithm of the agent to be eligible in the original problem domain.

Problem solving can be distributed to different agents. Collection of the agents forms a multi-agent system [31, p. xiii]. Using Multi-Agent Systems design introduces individual- related terminology, like roles, goals and intentions. It is argued by many people of providing more abstract way than Object-Oriented design when handling many entity control problems is considered. [32, p. 165] Using abstractions removes unnecessary parts and features and helps to concentrate on the most important and repetitive structures. Designing interaction models in cyber- physical systems with the help of multi-agent theory is suggested for example by [8, p. 455].

There are two main sets of design principles for multi-agent systems, agent- and organisation centered. Agent centered multi-agent systems (ACMAS) put the agent in focus. ACMAS systems are designed with respect to agents mental states. According to Shoham [33, pp. 333–334], the mental state of an agent can be divided into three categories:

* belief,
* capability,
* obligation or commitment.

Beliefs can be updated via communication, one can update other agent’s belief, or one can update its own belief about the other agent’s attitude [34, p. 447]. Capability is agent’s understanding of what it is able to do and this understanding affects its decisions. Obligation is like a choice that directs the future actions of agent. A decision is defined as an obligation to oneself. It is noted though, that ACMAS approach brings the problem of unpredictability. Agent-oriented systems produce

emergent phenomena and their interaction lead to results which cannot be predicted. [35, p. 290] This unpredictability is a challenge in system design.

In Organization Centered Multi-Agent Systems (OCMAS) an organisation can define the structure and the goal of the system while not defining the implementation [36, pp. 59–60]. Organisation centered framework is defined by [1, p. 5]:

“1. An organization is constituted of agents (individuals) that manifest a behavior.

1. The overall organization may be partitioned into partitions that may overlap (we will call these partition groups from now on)
2. Behaviors of agents are functionally related to the overall organization activity (concept of role).
3. Agents are engaged into dynamic which may be “typed” using a taxonomy of

roles, tasks or protocols, thus describing a kind of supra-individuality.

1. Types of behaviors are related through relationships between roles, tasks and

protocols.”

Organisations per Agents (OperA) is a design framework for multi-agent organisational structures. It consists of three models: organisational, social and interaction model. Organisational model defines the norms, roles, interactions and ontologies that prevail. Social model maps the roles to the certain agents and define contracts for applying for these roles. Interaction model defines the interaction between the agents. Interaction is crucial to enable the roles to exist in the first place. [37, p. 104] There are other frameworks, like Organization-based Multiagent Systems Engineering (O-MaSE), Tropos and Guidelines for Organizational Multi- Agent Systems (GORMAS), but OperA is claimed of supporting design abstraction more than these ones [38, p. 26].

Communication between agents is important in multi-agent systems. It is argued that the most important feature for agent is the ability to communicate between the agents. Durfee et al. [39, p. 100] described that ability as “a fundamental premise of an agent-based system”. Mental terminology, like beliefs, capabilities, choices and commitments are often advantageous when describing operation of complex systems, thus avoiding the need for understanding the underlying complexity [40, pp. 272– 273]. The beliefs can be updated via communication; one can update other agent’s belief, or one can update its own belief about the other agent’s attitude [34, p. 447]. This kind of abstractioning leaves only the core terms needed for realising the communication functionality and helps to abandon unnecessities and repetitive structures. Repetitive structures are the similarities in different communication implementations which are logically common and should be included in their common parent class.

## Prior work on using virtual reality in cyber-physical systems

Ripolles et al. [41, pp. 349–350] used real world information to model the virtual environment objects visually more realistically. In this thesis the focus is on the functional credibility. The visual aspect is implemented in a level that allows the user to intuitively understand and interact with the system and conceive the mapping between the corresponding real and virtual devices. For more literature about visual consistency and perception, see for example [42]–[44].

Modelling of real and virtual objects is already used in automobile industry. Regenbrecht and Holger used the terms *virtual engineering* and *mixed virtual*

*engineering*. They dismissed the idea of using feedbacks between virtual and real objects, and used instead the software models of the real and virtual worlds [45, pp. 1–2]. On the contrary, this thesis combines real and virtual devices. Ability to control the real devices from virtual world or virtual devices from real world enables taking the good sides from the both worlds: the realistic operation of the real devices and the augmented audiovisual representation capabilities of the virtual world, only to name a few.

In this thesis the real-world environment is mirrored in virtual world. An example of mixed space is Mixed reality architecture (MRA), which is an interconnection between offices in mixed reality [46, p. 62]. It represents real offices in virtual space and the physical positioning of the offices in virtual space affects size of the video stream and volume of the audio received from another office.

Croatti and Ricci [47, p. 9] introduced a conceptual framework for augmented environments. They stated that issues related to augmented worlds are synchronisation of computational augmented layer and physical layer that are superimposed as well as causal consistency of cross domain chains of events. This work approaches these challenges in a simple mixed reality implementation.

Milgram and Kishino [15, pp. 6–7] presented a definition of real and virtual objects in these respective worlds. Real objects “have an actual objective existence”, while the existence of virtual objects is restricted to simulated or effective existence. But when Metaverse is considered, it is logical to think that fusing reality and virtual reality brings more perspectives to the objects. In mixed reality there can be mixed objects. Coutrix and Nigay [48, p. 44] define mixed object as “a set of physical properties linked with a set of digital properties”. It is capable of acquiring or generating physical and/or digital properties, or doing the both.

An example of mixed object is Digital Desk. It is a paper on a table equipped with a camera and a projector. It enables combining physical drawings and digital graphics displayed by the projector. The digital graphics can be added and removed with the help of the camera detecting the physical actions. [49, pp. 11–12] The paper can be thought as a mixed object.

In this work the elements of a cyber-physical system are extended to a broader context as mixed objects. As the earlier-mentioned definition stated, mixed object is described by its linking between physical and digital properties. The concept of mixed object is used in this thesis to link the physical and digital location, looks and functionality of the real and virtual devices. The location of the object in physical location corresponds to the location in digital (virtual) world and vice versa. The looks and the observable visual state of a mixed object is similar in reality and virtual reality.

In this thesis the demonstration of functionality takes the definition of mixed object by Coutrix and Nigay to a more abstract level: not just properties, but the way mixed object acts and functions have linking between physical and virtual world. In this thesis, the functionality in virtual reality is described by means of software agents and the interactions between them in multi-agent system. Functional linking can be concretely described here as follows. The event on object in physical world affects one or many objects in virtual world. For example a physical sensor gets triggered and that triggering causes a virtual display to be put on. This is a simplified example of interaction between mixed objects. The real object and its virtual mirrored representation are not just co-existing; they are a single mixed object in mixed world.

This thesis demonstrates how the introduction of mixed reality can improve the design and use of cyber-physical systems in two ways. Firstly, it is feasible to implement virtual objects, for example virtual sensors, which can operate and interact with real world synchronously. The testing of the virtual implementations of these, perhaps non-existing or not on hand, devices can save time and enable quicker and more agile design processes. Secondly, virtual world gives a possibility to display the monitored energy information in a graphically rich way with less effort than in real world, and the display format can be fairly easily adapted to the user’s need.

# DESIGN

The design process is presented as follows. First the different actors and ways of using the system are recognised. This is followed by the introduction of the requirements. Then the system design is described followed by the establishment of interface design. Lastly, the chosen design is reviewed and discussed.

## Use Case and Scenarios

A use case and example scenarios are presented. Use Case is detailed in Table 1 and illustrated in Figure 2. Primary user is the active user who interacts with the system and triggers input events. Both Primary and Secondary users are able to observe the system response in real and virtual worlds, respectively.

Table 1. Use case details.

|  |  |
| --- | --- |
| **Use case name** | Observe system response |
| **Actors** | Primary user, Secondary user, system |
| **Use case description** | Primary User activates the system and observes the visible state and response of the system in reality. At the same time Secondary User observes the situation in  virtual environment. Goal is to observe the system response. |
| **Preconditions** |  |
| Precondition 1 | Both users are in the room and  Secondary User has accessed the view to the virtual environment. |
| Precondition 2 | The system is powered up and correctly  configured for use. |

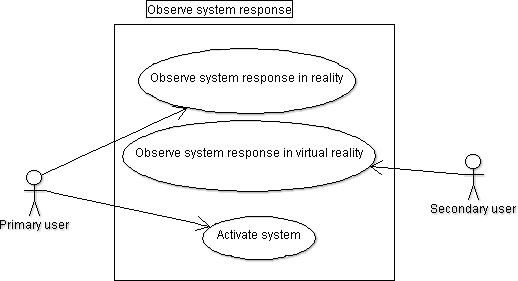


Figure 2. Use case.

One successful and a set of unsuccessful scenarios are presented. The scenarios are presented in [Table 2.](#_bookmark7)

Table 2. Scenarios.

|  |  |
| --- | --- |
| **Successful** | Primary user activates the system. The system responds to the user actions and Primary and Secondary user can compare their experiences. Their experience of the system response is consistent and the dynamics is  synchronous. |
| **Unsuccessful** |  |
| Scenario 1 | Primary user activates the system. The  system halts. |
| Scenario 2 | Primary user activates the system. Observations by Primary and Secondary  user are remarkably different. |
| Scenario 3 | Primary user tries to activate the system.  The system does not register the activation effort. |
| Scenario 4 | Secondary user experiences remarkable  disturbances in the interface to virtual representation. |

## Elicitation of Requirements

The sources [50], [51], [52, p. 677] were used as help when capturing the different requirements.

Table 3. Requirements.

|  |  |
| --- | --- |
| **State/Mode Requirements** |  |
| Power control | User is able to switch on/off the system. |
| **Functional Requirements** |  |
| Control order | Control from reality supersedes control  from virtual reality. |
| Data storing | Runtime data is stored in a non- cumulative way so to allow system to  run long periods of time without the memory running out. |
| Input | Setting a new input value is available all the time in reality and virtual reality. The  system must be operative regardless of the input combination given. |
| Output | Output data is a noticeable response to |

|  |  |
| --- | --- |
|  | the input action, and must be observable  both in reality and virtual reality. |
| Shutdown | No specific actions related to shutdown  are needed. |
| Software control | System handles the synchronisation and consistency of the system response in reality and virtual reality. Conflicting  actions are detected and handled. |
| Startup | During startup the different blocks are given time to settle down and initialise  themselves. |
| Timing | Timing is adjusted so that visually observable actions in reality and virtual reality have a timing difference less than  5ms. |
| Transaction corrections, adjustments and cancellations | System does consistency checks to  ensure that the corresponding states of reality and virtual reality objects match. |
| **External Interface Requirements** |  |
| Connectivity | System allows addition of new real or virtual devices that are able to communicate with the supported  communication methods. |
| Ease of validation | System enables concurrent observation and comparison of the setups in reality and virtual reality. For example energy levels (voltage) are open to be monitored  at a comparable level. |
| Input | System catches input from both setups in reality and virtual world. Activating multiple input resources simultaneously  must be possible |
| Output | System provides output data to reality and virtual reality so that the system  information can be tracked in real-time. |
| Relative timing | Whether the system was activated in reality or virtual reality, the system response happens simultaneously to the  human eye in both realities. |
| **Non-Functional Requirements** |  |
| *Environmental Requirements* |  |
| External stimuli | System handles an external stimulus consistently, if the system reacts to it. Here the consistency means that the corresponding parts of the real and virtual world are in the same state and  the oscillation between states upon change is avoided. |
| System appearance in external | External environment is able to observe |

|  |  |
| --- | --- |
| environment | both the reality and virtual reality operating so consistently that it does not distract the users and does not degrade  the system credibility. |
| *Capacity, current and forecast* | Capacity restrictions stem from the  hardware setup, not by the software setup. |
| *Deployment* | System is deployed in an existing premise at VTT Oulu suitable for this  kind of system setting. |
| *Documentation* | Documentation allows the reproducibility and further development  of the system by a person not familiar with it. |
| *Extensibility* | System is designed to have enough abstraction levels that support  extensibility. |
| *Failure management* | In case of failure the system is restarted  manually. |
| *Failure tolerance* | The system stands minor error situations, like non-critical timing errors and discrepancies in states that should be  equal. |
| *Interoperability* | System provides interfaces to output data  of the system to the outside world. |
| *Modifiability* | System is designed to allow addition of devices, protocols among others like  plugins. |
| *Operability* | System interacts with one active user coherently and operate according to the timing requirements all the time the  system is up. |
| *Performance Requirements* |  |
| Latency | Latency between the activation of the  system and the system response does not exceed 1000ms. |
| *Physical Requirements* | The real world setup is restricted by the  size of the deployment area. |
| *Resource Requirements* | System is designed to enable integration of the computing and embedded part even when embedded part has limited  resources. |
| *Reusability* | System is designed to allow further  development and support extensibility. |
| *Scalability* | Virtual reality enables scaling up at least  to some extent. |
| *Testability* | System allows data extraction to be used in testing to the extent it does not affect  the performance and stability of the |

|  |  |
| --- | --- |
|  | system. |
| *Usability by target user community* | One active user is allowed at a time, amount of passive (observing) users is restricted to the number of people that does fit in the installation premise and stay away from creating stimuli to the  system. |

## System Design

The used design patterns and frameworks are presented. After that the system architecture is presented.

### Design patterns and frameworks

Model-View-Controller (MVC) is a modular design pattern for separating the data, the user interface and the functional logic using these both. The operation of application domain, visual side of the application and the interaction between these two are assigned to different components [53, p. 2].

Organisations per Agents (OperA) is a design methodology for multi-agent systems. The design has three main steps, Organisational model design, Social model design and Interaction model design. [54, p. 53] OperA allows the use of any methodology for designing the agents itself [54, p. 168].

Organisation model design is the design of the multi-agent system from the the perspective of the organisation. It is subdivided into three levels: Coordination Level, Environment Level and Behaviour Level. Coordination Level specifies the structure of the agent society. Environment Level names the roles, global requirements, domain categories and the relations between the categories. Behaviour Level specifies the interaction structure by recognition of the interaction patterns and processes to be used. [54, pp. 169–170]

Social model design describes the roles, which kind of agents ask for which roles and how they gain them [54, p. 77]. Interaction model design defines the concrete interaction scenes and contracts from the point of view of agents [54, p. 89].

Hierarchy framework presented by Dignum [54, pp. 165–167] is divided in top- down direction into two levels: Faciliation layer and Operation layer. Facilitation layer consists of agents responsible for controlling the agent society. The agents at the Operational layer execute the objectives of the society. The hierarchy coordinates the information and resource flow in the society through adjacent levels in top-down direction. The communication links are defined well.

Dignum [54, pp. 162–163] presented Market framework as a way for agents to exchange their services. Market framework comprises three elements: matchmaker, bank and market master. Matchmaker retrieves information about the agents and the services they need or offer. Matchmaker tries to match the agents’ needs and supply. Bank or similar construct can provide the information related to the exchange process. Market master oversees the market protocols.

Mediator is a known term for a certain type of software component. Wiederhold [55, p. 191] describe it as follows: ”A mediator is software module that exploits

encoded knowledge about certain sets or subsets of data to create information for a

higher layer of applications.”.

Federations are groups of agents communicating through delegates [56, pp. 171– 172]. The delegate communicates with other delegates and other objects. If some object wants to communicate with an agent in a federation, they are in contact via the corresponding delegate. When this kind of federal hierarchy is used, agents in the federations can have more independency. They are only communicating with other agents in same federation and with the delegate.

Agent is a body for an independent working unit. actSMART Agent Shell model ([Figure 3](#_bookmark11)) introduced by d’Inverno and Luck [57, pp. 206–207] describes the components agent consists of. The components are Controller, Infostore, Actuator, Sensor, Execution Sequence, Link Management, Control Policies and Agent-Specific Attributes.

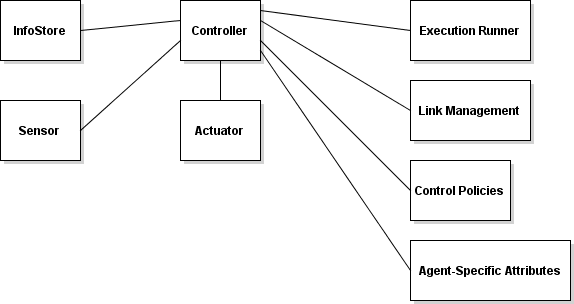


Figure 3. actSMART Agent Shell model.

Controller is a main decision making entity. It uses Sensor and InfoStore elements in its decision-making. Sensor is the interface for Agent to percept its ambience. InfoStore includes the stored information models. By using Sensor and InfoStore to decide its actions, Controller enforces the actions affecting the environment through Actuator. [57, p. 206]

Execution Sequence defines how the aforementioned components execute and in which order. One needs to decide whether they are executed synchronously in a loop or asynchronously in their own threads.

Link Management implements the communication interface. Control Policies are context specific rules according to which agent must perform. Control Policies is a separate component because the independence of rules from the agent architecture is desired. The final component is Agent-Specific Attributes which describe the Agent itself and its properties.

### Overview of the logical organization

The logical components of the system are Reality, Synchronisation block and Virtual reality presented in Figure 4. Reality comprises the hardware part of the system. Synchronisation block and Virtual reality form the software part. Virtual reality creates the virtual representation and Synchronisation block binds Reality and Virtual reality together and is responsible for timing and consistency.

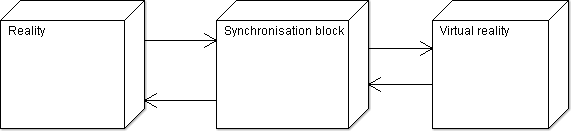


Figure 4. Overview of the logical organisation.

### Subsystem decomposition

System subdivision comprises Cyber-physical system (capitalised on purpose), Synchroniser, VirtualWorldModel and VirtualWorld (Figure 5). Cyber-physical system is the embedded system that is mirrored on the virtual reality. Synchroniser is the module responsible for maintaining correct timing and consistency between reality and virtual reality. VirtualWorldModel is a dynamic object-oriented representation of VirtualWorld and VirtualWorld is an mirrored and augmented virtual representation of Cyber-physical system. The choice of positioning the VirtualWorldModel inside Synchronisation block was strongly affected by the decision of having the control of the functionality of virtual reality objects near Synchroniser.

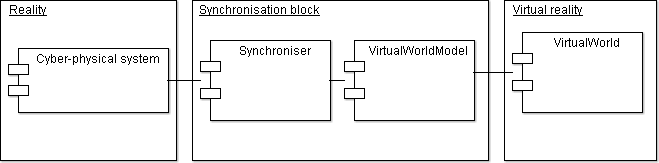


Figure 5. Subsystem decomposition.

### Software and hardware mapping

Software and hardware mapping is presented in [Figure 6](#_bookmark12). Cyber-physical system is instantiated as an embedded system with peripherals. VirtualWorld is running on its own computer, and Synchroniser and VirtualWorldModel run on same pc.

VirtualWorld representation may need a lot of resources, and there is clear interface between it and the model.

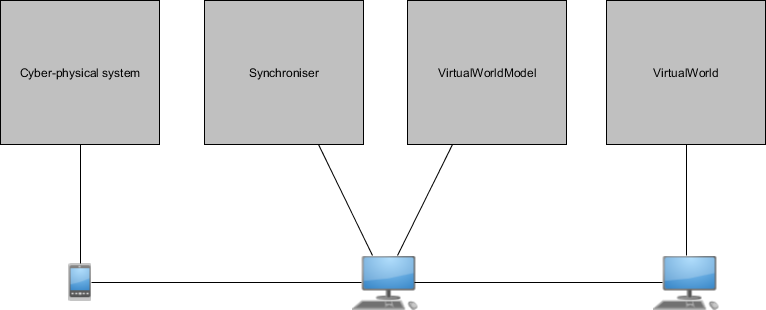


Figure 6. Hardware and software mapping.

The data is stored only on runtime in temporary storages. No kind of access control or security is considered.

### Subsystem designs

As presented earlier, subsystems are Cyber-physical system, Synchroniser, VirtualWorldModel and VirtualWorld.

* + - 1. *Cyber-physical system*

Cyber-physical system (Figure 7) was designed to consist of an input device, some controlled devices, an energy source to be measured, an energy measurement block and a controller. Input device is the interface to receive input stimuli from user in reality. The controller interprets the stimuli and controls the displays to create observable responses.

The main functionalities of Cyber-physical system are to allow user interaction, and to have energy monitoring capabilities. These two aspects, interactive functionality and energy monitoring, were chosen because it is argued they can be brought added value when cyber physical system is merged with mixed reality.

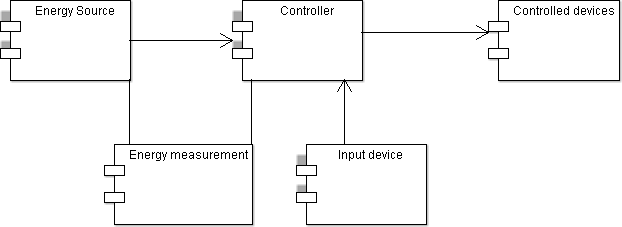


Figure 7. Cyber-physical system decomposition diagram.

* + - 1. *Synchroniser*

The building blocks for Synchroniser are EventSynchroniser, ConsistencyChecker and RealWorldCommunicator as shown in Figure 8. Through EventSynchroniser all events from reality or virtual reality are conveyed, and this block ensures the synchronisation of reality and virtual reality. It EventSynchroniser also recognises and resolves the conflicting control commands. ConsistencyChecker monitors the states of reality and virtual reality and requests VirtualWorldModel to do corrections if discrepancies are found. RealWorldCommunicator is an asynchronous communication block that communicates with Cyber-physical system.

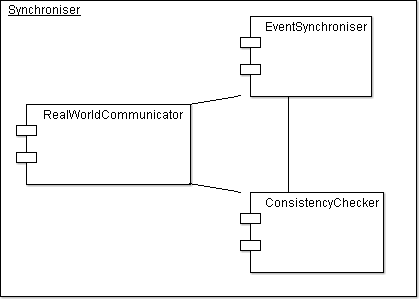


Figure 8. Synchroniser decomposition diagram.

* + - 1. *VirtualWorldModel*

VirtualWorldModel contains the functionality of the virtual devices. It was designed as a society of collaborating agents having some individuality in their operation. The followes OperA multi-agent system framework. The main steps in OperA design are design of Organisation model, Environment model and Behaviour model.

* + - * 1. ***Organisation model design***

In organisation model design the multi-agent system is described from the perspective of the organisation. The design is divided into design of Coordination Level, Environment Level and Behaviour Level.

**Coordination Level**

The structure of VirtualWorldModel is hierarchical which is presented in [Figure 9.](#_bookmark14) It consists of VirtualWorldModel, VirtualWorldModelOrganisationRuler, MessageBroker, ServiceMarket, VirtualWorldCommunicator and Federations.

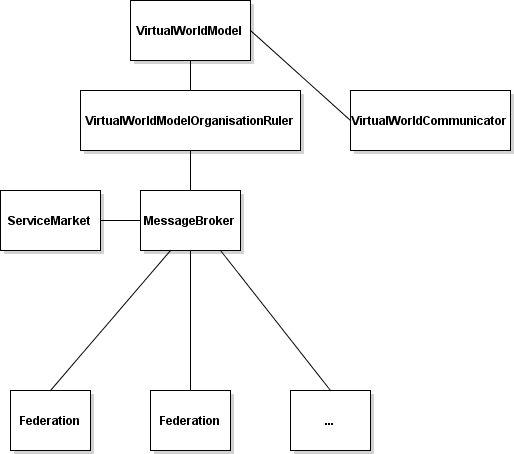


Figure 9. VirtualWorldModel hierarchy.

VirtualWorldModel communicates with Synchroniser and with VirtualWorld through VirtualWorldCommunicator. It organises the external relations of agent society and takes care of the synchronisation actions requested by Synchroniser. VirtualWorldModel lets VirtualWorldModelOrganisationRuler to take care of the agent society itself.

VirtualWorldModelOrganisationRuler is the ruler of the agent domain. It can enforce restrictions, global goals and rules. It collects the synchronisation data from this domain and provides it to the Synchroniser. Thus it is similar to the concept of Mediator explained earlier.

MessageBroker acts as a message delivery actor between Organisation, ServiceMarket and Federations. MessageBroker provides an interface to implement the asynchronous messaging of the system.

ServiceMarket takes care of allocating the services to the subscribers. It is fundamentally same as Market framework. The service providers do not know which subscribers use their service. On the other hand, the subscribers need only to know how much of each service do they have in use. This structure helps avoiding the cumbersome negotiation processes between each service provider-subscriber pairs. The service registration and processing takes place in a single place, thus enabling centralised decision-making. The aspects that are taken into consideration in these decisions are the priority of the subscriber and how to reallocate the services if a higher priority subscriber registers later. The ServiceMarket provides thus a clear advantage of simplifying the resource sharing process.

VirtualWorldCommunicator handles the communication with VirtualWorld. The changes in data of the virtual objects is sent to VirtualWorld to be visualised. Besides, VirtualWorldCommunicator receives input events from VirtualWorld which affect the operation of the VirtualWorldModel.

The environmental consciousness of an agent is limited to a small scope around each agent, and the agents within this social range form a Federation ([Figure 10](#_bookmark15)). Each Federation has a FederalDelegate as its link to the other Federations. The focus in the operation of an agent is on decision-making in this limited scope of consciousness. The division of Agent instances into Federations is done based on the similarity in their functionality.

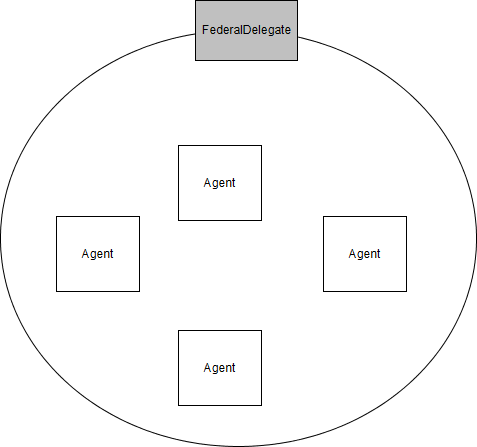


Figure 10. Federation.

FederalDelegate acts as an advocate for a Federation. Agents in a Federation belong densely to the same context. Using the concept of Federations frees the agents to concentrate on acting in their local society while having the channel to the ambient world through a single interface, FederalDelegate. The restrictions and collective messages can be delivered to the logically formed groups through this single interface. Minar et al. [58, p. 288] presented a hypothesis for a design guideline according to which restricting interaction between agents locally to a point simplifies the design and can eliminate failures. Hierarchy framework also emphasizes well- defined communication lines. Thus positive about using Federations is that it limits the amount of information an Agent needs to have about its society and simplifies the communication structure. Definition of agent itself does not belong to OperA. The design of agent is presented after the OperA design.

**Environment Level**

There are two role groups in society: service roles and control roles. ServiceProvider and ServiceSubscriber are service roles. Control roles are Controller and Controlled.

There are two domain categories: agents which have a virtual representation and agents which do not. The agents with virtual representation are further categorised into agents which have and do not have a corresponding real object in reality and agents which do not. The agents with a correspondence have priority over the other agents. The categorisation can only be seen in interactions and negotiations.

**Behaviour Level**

Next the utilised interface patterns and guidelines are presented. Thereafter the

society’s interaction structure is shown.

Lock pattern is a synchronisation pattern used to handle situations in which concurrent accesses to objects are possible but not acceptable. “A lock is an abstraction that allows at most one thread to own it at a time.” [59].

Request-reply is messaging pattern according to which an entity sends a request to another entity. The receiver processes the request and sends a separate response to the sender. This method has uses when the request leads to some processing or causes delay.

Message queueing patterns [60] are used in asynchronous messaging. Message queue pattern is shown in [Figure 11](#_bookmark16) and it describes the asynchronous messaging between sender and receiver. Shared message queue pattern ([Figure 12](#_bookmark17)) is used in this work to deliver messages through a common mediator, Message queueing handler. Broadcast messaging pattern ([Figure 13](#_bookmark18)) enables delivering similar data via same communication channels and is used to distribute for example control commands to similar objects.

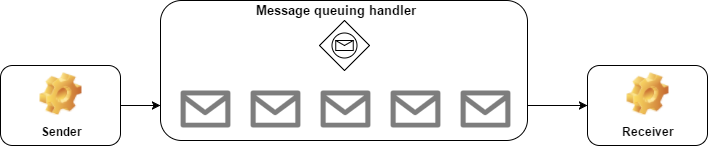


Figure 11. Message queue pattern.

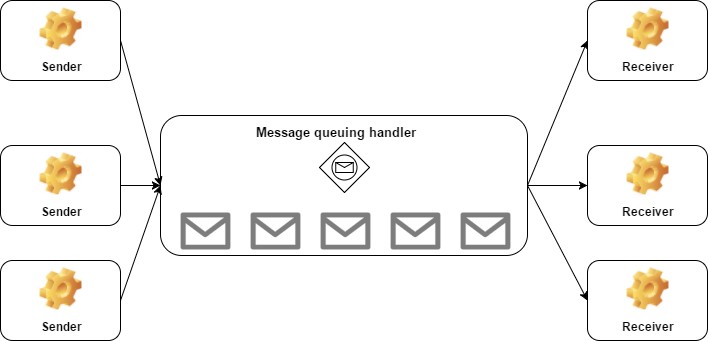


Figure 12. Shared message queue pattern.

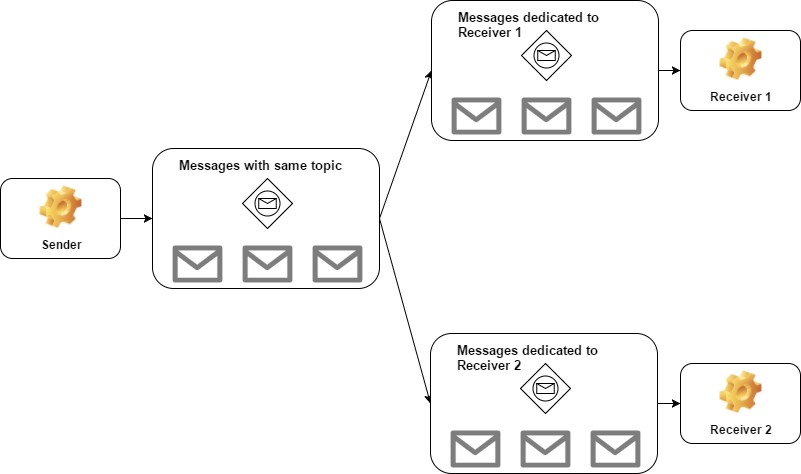


Figure 13. Broadcast messaging pattern.

The interaction structure ([Figure 14](#_bookmark19)) of VirtualWorldModel is hierarchical and fixed. MessageBroker connects VirtualWorldModel, VirtualWorldOrganisationRuler, ServiceMarket, MessageBroker and DelegateAgent instances. Agents reach this communication joint through FederalDelegates. VirtualWorldCommunicator communicates directly with VirtualWorldModel because the input messages from VirtualWorld must be forwarded straightforwardly to Synchroniser.

Messaging between entities next to each other follows Message queue pattern. Shared message queue pattern is applied on MessageBroker to enable centralised conveying of messages in agent society. Broadcast messaging pattern is used on FederalDelegates which can deliver the received messages to the right agents in Federation. Concurrent accesses to the message queues are secured with the use of Lock pattern. Request-Reply pattern is used when registering as ServiceSubscriber in ServiceMarket or when requesting to control some agent.

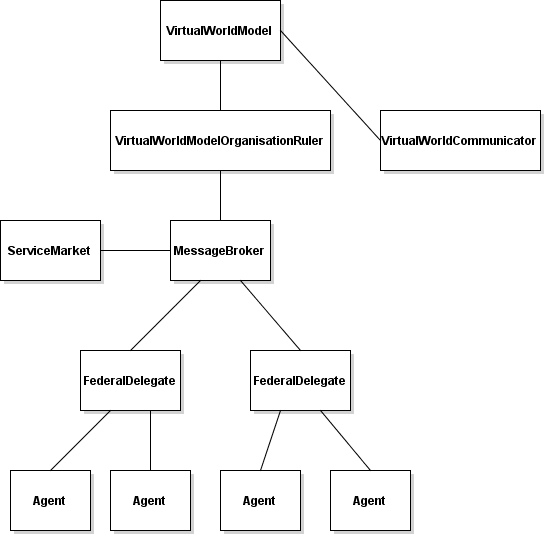


Figure 14. VirtualWorldModel interaction structure.

* + - * 1. ***Social model design***

There are two categories for roles, Service roles and Control roles. The service and control roles are given upon initialisation of the system.

Service roles, ServiceProvider and ServiceSubscriber, define the agent’s behaviour with ServiceMarket. ServiceProvider registers as ServiceProvider at ServiceMarket, tells about the services it offers and updates the statuses of the services. It does not have to do anything else, as ServiceMarket takes care of matchmaking. Similarly, ServiceSubscriber registers as ServiceSubscriber at ServiceMarket and tells the services it needs and services it already has, and notifies ServiceMarket about updates in these details when they change.

Control roles, Controller and Controlled, reflect agent’s relationship to a certain agent. Control role is relationship dependent, and an agent can act as Controller to some agents and Controlled to some agents. Control role is a way for agents to create and acknowledge control relationships.

* + - * 1. ***Interaction model design***

Agents with correspondence in reality have priority compared to other agents when it comes to allocating resources of a service.

Communication follows the message queueing patterns presented by [60]. Concurrency during these messaging processes is enabled and secured by using Lock pattern. Communication consists of structures that allow agents to communicate and convey information in an abstract way. Abstract interfaces enable many different objects to use them and they hide the implementation, so that the agents communicate through simple interfaces. Agents can send messages through these interfaces.

* + - * 1. ***Agent***

Agent is a body for an independent working unit and a basic building block in agent society. The software model in this thesis uses actSMART Agent Shell model introduced by d’Inverno and Luck [57, pp. 206–207] with some modifications. The components of agent in this system design are AgentController, Infostore, SensingElement, Execution Runner, Link Management and Agent-Specific Attributes. The Agent model is shown in [Figure 15.](#_bookmark20) It needs to be mentioned that the referred models present the composition of agent at a high-level and the implementation of the functionality and content of these components were designed and developed in this thesis work.

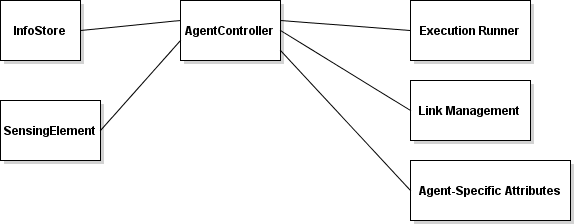


Figure 15. Agent model.

AgentController is a main decision making entity. It uses SensingElement and InfoStore elements in its decision-making. SensingElement is the interface for Agent to percept its ambience. InfoStore includes the stored information models. In this thesis, the information about the services needed, provided and used by agent as well as the information about the agents controlling this agent and agents controlled by this agent is stored here. AgentController enforces the actions which affect the environment.

The system designed is a reactive system which tries to keep the state of VirtualWorld consistent with the state of Cyber-physical system. Execution Runner allows AgentController to enforce the actions resulting from its decisions in a

separate thread. This way AgentController does not need to halt during execution of the actions it decides to make, but can keep being available for planning and decision-making.

Link Management implements the communication interface. Communication is implemented asynchronously in order to allow agents to execute at their own speed. Receiving messages asynchronously was implemented with the help of buffers as described in Figure 5.

Agent-Specific Attributes describe the Agent itself and its properties. Event listening interfaces for the changes in the attribute values are provided.

* + - 1. *VirtualWorld*

VirtualWorld is an artificially created virtual environment that represents the visual side of the virtual world. It mirrors and resembles the physical test environment used in the thesis but has some additional elements. VirtualWorld consists of objects that correspond to the devices of Cyber-physical system in reality, and some additional virtual devices. VirtualWorld shows the visual states of virtual objects and shows changes in those states. These states are consistent and the changes are synchronous with the Cyber-physical system.

VirtualWorld captures the input signals coming from for example a triggered sensor. VirtualWorld does not include any operational logic, and thus it has similarities to the View- component of the Model-View-Controller- pattern used in user interface development. The representation includes the people in the test environment. VirtualWorld is capable of capturing input data resulting from the actions of the active user and producing output data.

## Interface Design

Next the utilised interface patterns and guidelines at the system and subsystem levels are presented. Thereafter, the interface design is presented.

TeaTime is a scalable multiuser collaboration system architecture, which aims at providing a fluent user interface by synchronising I/O events with a common time [61, p. 5]. Nothing else but input and output data events are synchronised.

Observer pattern consists of publishers, which have and maintain some data, and subscribers. Subscribers register at the publisher whose data they want to track. When publisher updates the data, the registered subscribers get notified. Observer pattern is used for example with event handling.

When it comes to synchronisation, this work follows TeaTime approach but only the delivery of input events to actors in reality and virtual reality is synchronised. In this simple system, the time from an actor receiving an input till the time the output being produced is assumed negligible.

Propagating the events follows the Observer pattern. Event propagation is considered for example when change in the state of an input device is notified to the listeners.

There is two-directional communication interface between Cyber-physical system and Synchroniser. The time domains are a little different, as Cyber-physical system is considered operating synchronously here and Synchroniser works on software

domain which is asynchronous in this design. Thus there is a synchronous message handler on embedded side and an asynchronous message handler on software side.

There is no specific API in Synchroniser-VirtualWorldModel interface. Those modules reside in same memory and no specific API is needed. Access to data is provided by gaining a reference to the data. In VirtualWorldModel – VirtualWorld interface, there is a two-directional communication interface with asynchronous messagehandlers on both ends.

## Analysis of Design

The design is evaluated against the requirements. Inspecting the requirements led to a choice of tighter integration between VirtualWorldModel and Synchroniser due to the tight timing requirements. This is why VirtualWorldModel was transferred from logical block VirtualWorld to Synchronisation block. Additionally, requirement of allowing multiple input signals to be received simultaneously from reality and virtual reality made it logical to have functionality of virtual reality close to Synchroniser so that the delay from Synchroniser to VirtualWorldModel is as small as possible. The aim is that the conflict checking in Synchroniser has as up-to-date data as possible.

Allowing one active user at a time makes the approach more simple and clear. The focus of this work can remain on synchronisation and consistency and the technical evaluation can be conducted more straightforwardly.

The chosen asynchronous messaging patterns promote the reusability and extensibility as the operation of the software modules does not halt due to their communication processes.

# IMPLEMENTATION

The system was implemented in ResponseRoom, a research and development environment at Technical Reseach Center of Finland (VTT) in Oulu. The environment is a room equipped with three Kinect motion sensor cameras, three computers and a pair of Oculus Rift glasses. ResponseRoom acts as a test environment for gesture-based lighting control algorithms among other uses. The system deployment is visualised in [Figure 16.](#_bookmark24) Cyber-physical system was deployed on the wall on the left in the image. Synchroniser and VirtualWorldModel were run on computer at the bottom of the image. VirtualWorld was run on a separate computer on the right in the image. Kinect motion sensing cameras at the corners of the room track the user and Oculus Rift glasses provide the vision to the VirtualWorld for the Secondary user mentioned in the use case.

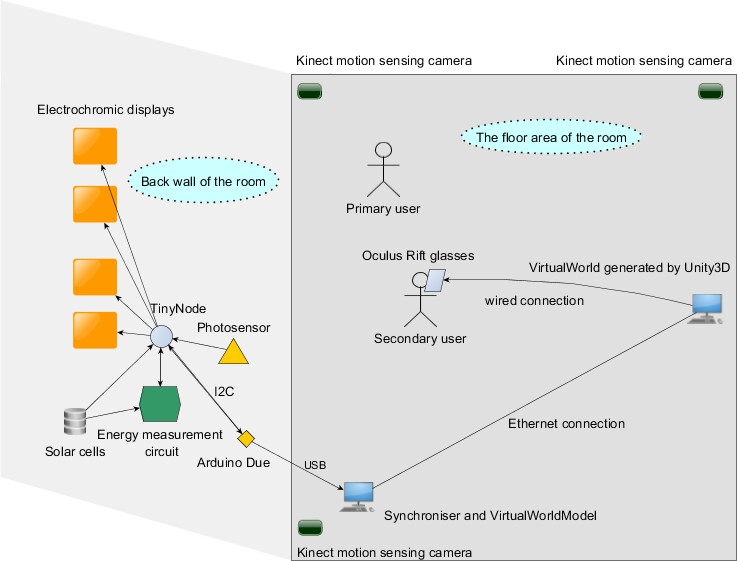


Figure 16. Physical setup of the test environment.

## Cyber-physical system

The real world system is a sensor network ([Figure 17](#_bookmark26)) which comprises a control board, four electrochromic displays, two solar cells and separate power supplies for the energy measurement circuit and microcontroller. The voltage, current and power provided by solar cells and respectively consumed by microcontroller chip is monitored by a measurement circuit.

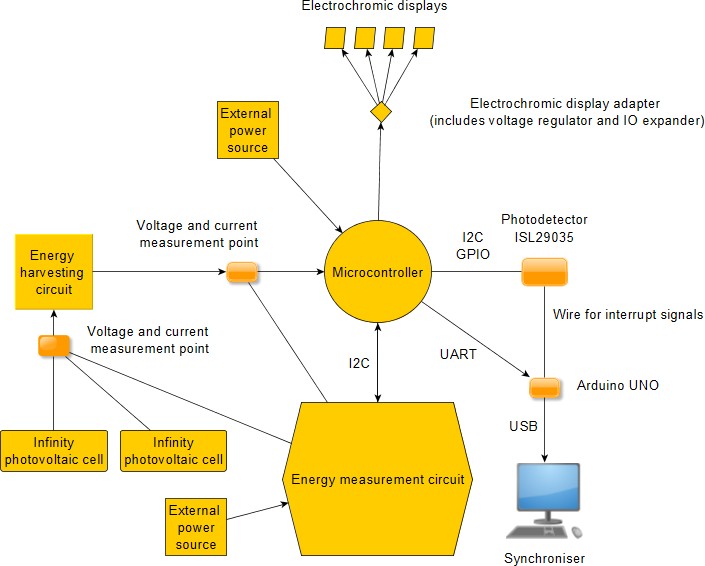


Figure 17. Block diagram of the device network of Cyber-physical system.

The control board includes a microcontroller chip, an energy harvesting unit, an energy measurement circuit and a photosensor. It is illustrated in [Figure 18](#_bookmark27). The board has connections to the electrochromic displays, solar cells and external power supply. It also shared an interrupt wire and a serial channel with an Arduino DUE board.

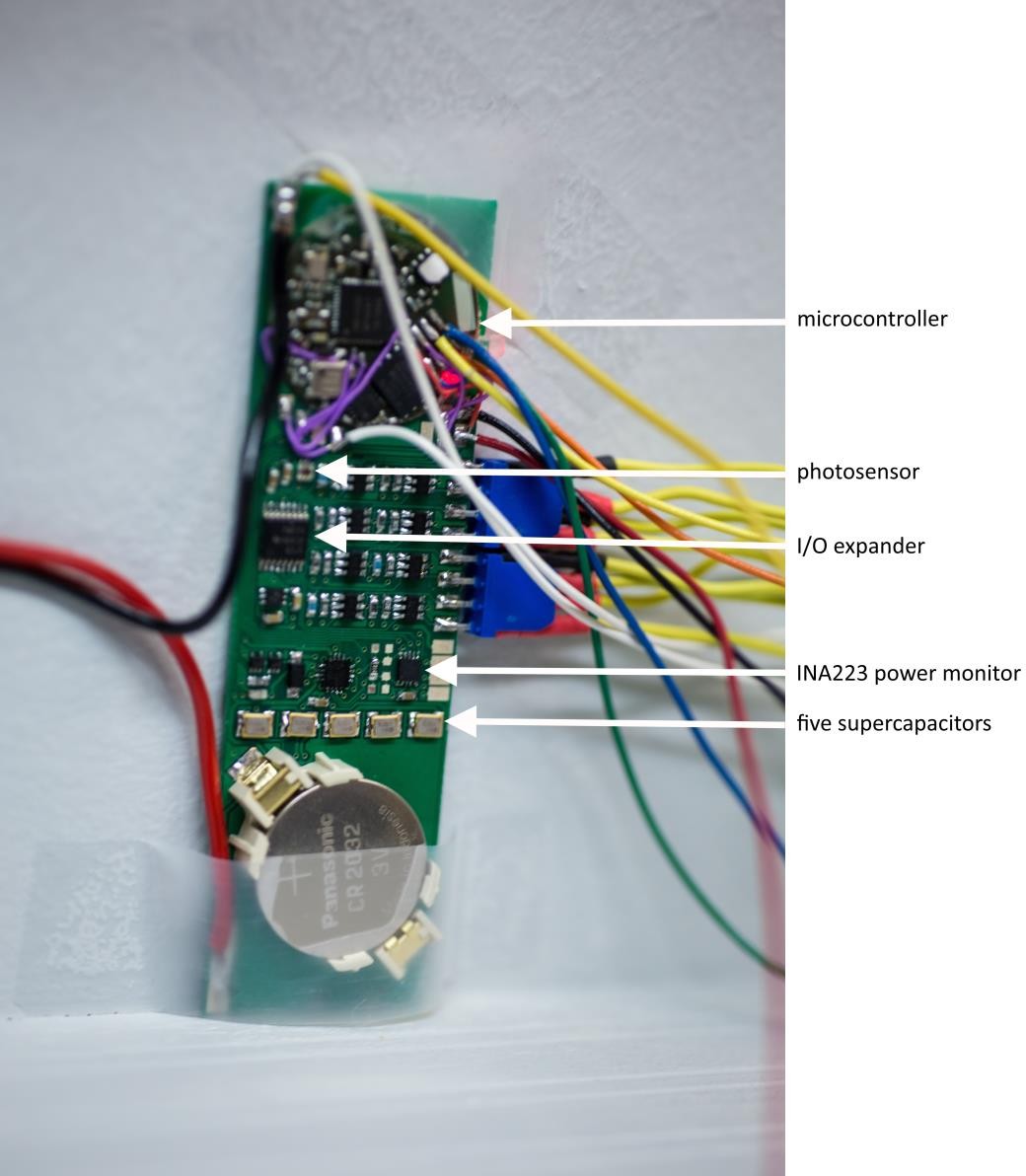


Figure 18. Control board.

The microcontroller chip has been designed by VTT Technical Research Centre of Finland (VTT) and has a size of a coin. It uses a commercial microprocessor and has multiple sensing capabilities along with a Bluetooth radio. General Purpose Input/Outputs (GPIO) along with I2C and Universal Asynchronous Receiver/Transmitter (UART) interfaces equip the chip with a decent connectivity to its environment. The specific description of the microcontroller is elided in order to protect the proprietary rights owned by VTT. The microcontroller listens the interrupt line and drives the displays when an interrupt from the photosensor or from virtual world is detected. It also conducts energy measurements and reports the measured data to the computer.

The energy harvesting unit acts as energy storage and buffer for energy provided by solar cells which transform ambient light into electrical current. The current is stored by energy harvesting circuit to provide an energy source for the microcontroller and the displays ([Figure 19](#_bookmark28)). The energy harvesting circuit is formed by a single diode and supercapacitors. The additional power supply was needed to

run the network, as the solar cells were not able to provide the system with enough power to run constantly. Nevertheless, measuring the energy characteristics of the solar cells provided useful real-time information.

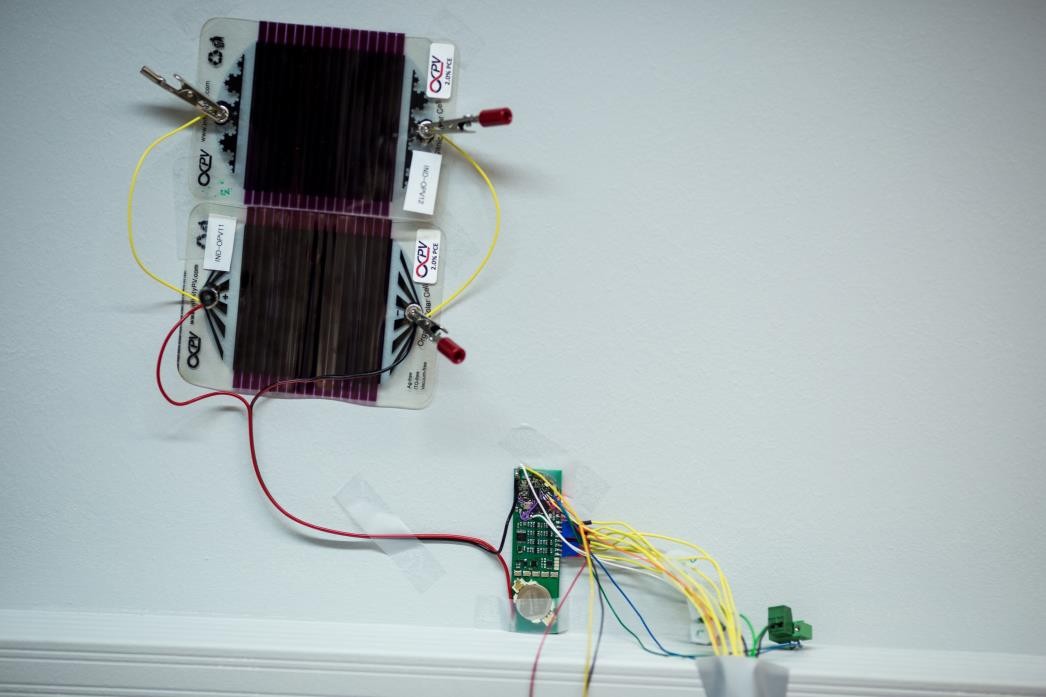


Figure 19. Two solar cells provided energy to the microcontroller circuit.

The measurement circuit consists of multiplexer for the measurement point selection, INA223 power monitor and a coin battery as power supply. The measurement circuit has two measurement points in the circuit: one between the solar cells and the energy harvester, and the other between the harvester and microcontroller chip. This setting enables measuring and comparing the energy provided by different solar cells, the efficiency of the two alternative harvester units, and the energy consumption of the microcontroller and displays. INA223 produces an output voltage that can be used, depending on the mode chosen, to infer the voltage, current, power sourced or power drawn from the measurement point. The microcontroller controls the measurement and changes the measurement point between the two points one after another, and the measured values are read through I2C.

The photosensor used was ISL29035. The values were read through I2C channel and interrupts from photosensor were utilised to read the illuminance values at the interesting events. By placing the photosensor and solarcells closely, the real-time monitoring of the energy of the solar cell and illuminance level provided interesting and useful information to be presented to the user in VirtualWorld.

The displays are electrochromic displays made by Ynvisible ([Figure 20](#_bookmark29)). An electrochromic material can change its colour when its electronic state is changed [62]. Electrochromic displays can be made of electrochromic material to show graphics. A colour state can be activated by applying a voltage difference between the electrodes of the display. The used displays have two different display patterns which are activated by bringing a voltage difference of 1.5V between the electrodes

of the display. Changing the polarity changes the pattern. Controlling four displays would have required a total of eight GPIOs, which was not feasible with this chip, so an Input/Output (IO) expander was used. The pin directions and digital output levels of the IO expander were controlled through an I2C interface. The electrochromic displays were fed by the same external power source used by the microcontroller chip.

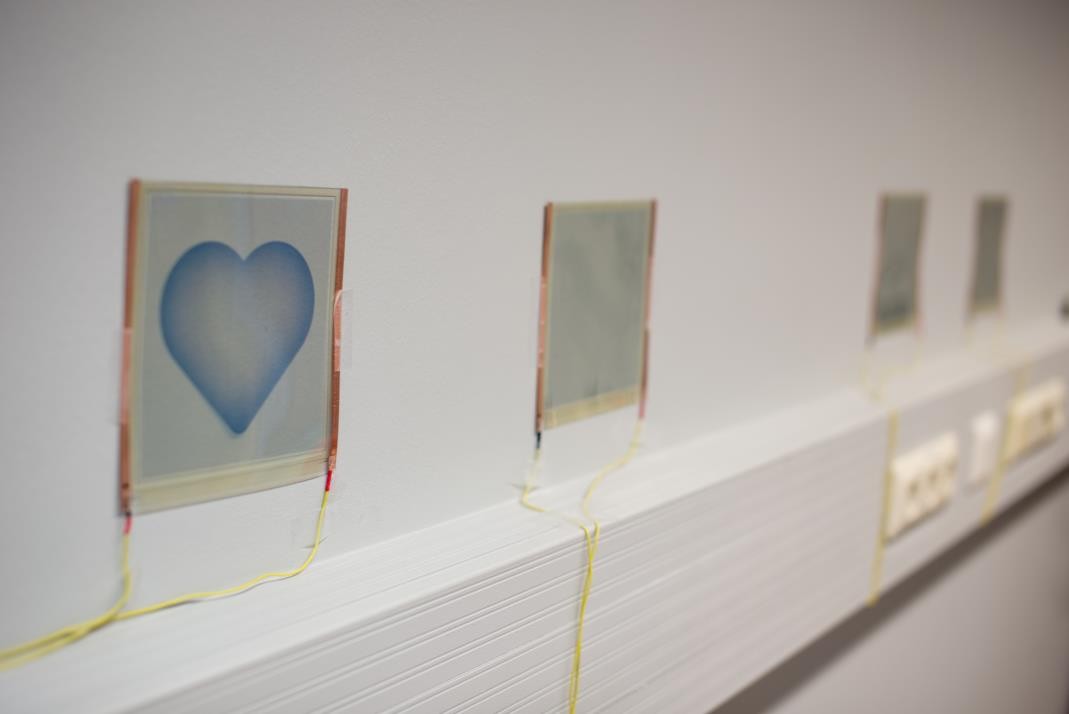


Figure 20. Electrochromic displays.

The interfaces of the control board were co-designed with Samuli Yrjänä. Mr. Yrjänä designed the schematic and layout and assembled it. Designing and programming the software for the microcontroller was done in this thesis work. The software was designed to lead to a low current consumption while running on the microcontroller. The energy measurements to verify the low current consumption were conducted during the development phase.

The information transfer between Cyber-physical system and Synchroniser was done via universal asynchronous receiver/transmitter (UART) and a single wire. The data flowing via UART and wire is conveyed to the Universal Serial Bus (USB) port of the computer by an Arduino DUE board ([Figure 21](#_bookmark30)). UART was chosen for single-direction data transfer and was used to send the energy measurement data to Synchroniser. The wire is a signal line for low-active interrupt signal. When a sensor is triggered in Cyber-physical system or VirtualWorld, the wire is driven low. Both the microcontroller and Synchroniser notice this event, and they act accordingly.

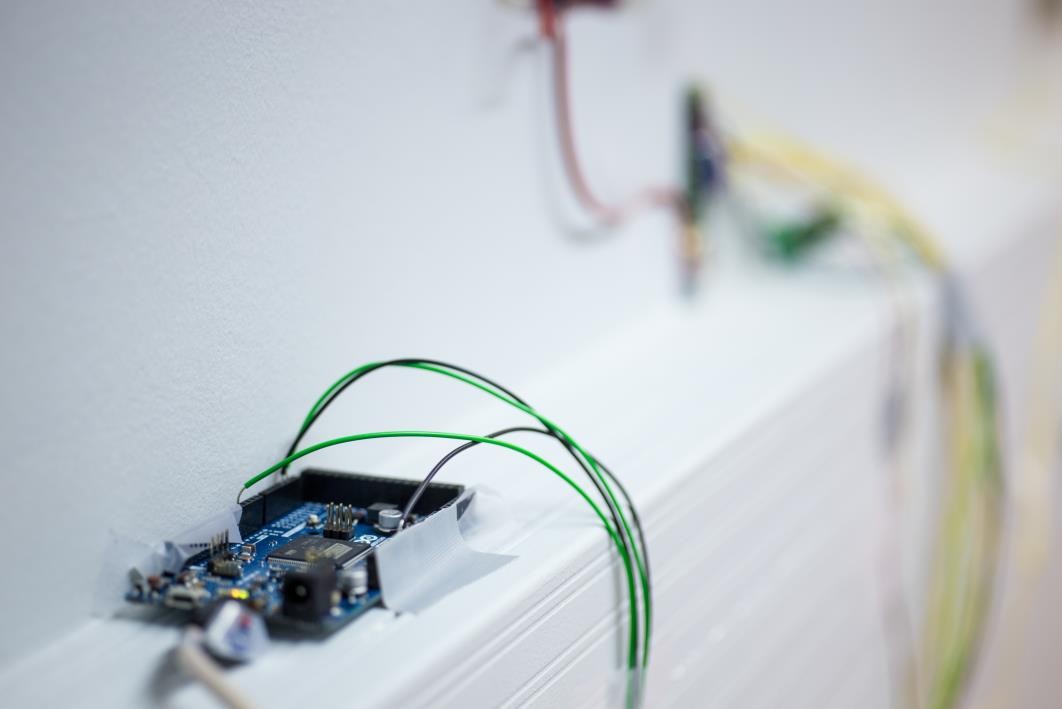


Figure 21. Arduino DUE board was an intermediate interface between

microcontroller circuit and computer.

Using this common wire for event triggering promoted synchronicity and solved the consistency problem related to control signals from different realities trying to control the common resources. The common interrupt signal notifies Cyber-physical system and VirtualWorld synchronically. The control consistency was resolved on first arrived, first served- basis. Whether an interrupt from Cyber-physical system or VirtualWorld pulled the wire low, the first one was noticed and the next interrupt signal was noticed after the current interrupt event had been cleared.

The reason behind this kind of communication structure was the requirement for low power operation. The real network was intended to be supplied by solar cells to at least some extent. The UART on this microcontroller consumed around 1mA current when enabled. This was reported by [63] and the measurements during the thesis work confirmed this behaviour. The current consumption of the microcontroller chip was cut down close to a time average of at most a few hundreds of microamperes. The UART was only used for sending the energy measurement data and illuminance value captured by the photosensor to the Arduino DUE once in a second. Updating the measured energy and illuminance values with this frequency resulted as giving a somewhat reasonable possibility for observing the measured values in virtual world.

The single wire accounts for a current consumption of at most just below 1mA when an interrupt is signalled. This can be accepted, as when the interrupt is cleared, the energy consumption cuts down to below 100 µA on the interrupt line, so the 1mA consumption happens just at times. The interrupt line is held low for at least 100ms and the listeners of the wire read the voltage level often enough to catch the interrupt events.

During the development of the sensor network the current consumption of the measurement controlling system was measured of having an average of around 65

µA. Driving electrochromic displays added the current consumption for around 4 milliamperes per a display. In theory, the average current consumption was low enough to allow the measurement controlling system to be run at least a couple of minutes when photovoltaic cells and preloaded supercapacitors were used. In practise, however, there was a problem with supercapacitors providing the current to the microcontroller, so an external power supply was used to feed the microcontroller chip. Nevertheless, when powered externally to keep the microcontroller chip fully operative, the system measured the energy provided by the photovoltaic cells and the energy drawn by microcontroller from solar cells correctly in real-time.

## VirtualWorldModel

VirtualWorldModel is a dynamic representation of the objects in virtual reality. The object diagram is presented in [Figure 22.](#_bookmark32)



Figure 22. VirtualWorldModel object diagram.

Next the implementation of the classes in VirtualWorldModel are presented. Only the most important attributes and methods sufficient for comprehensive explanation of the functionality of the classes are shown. The classes are presented in hierarchical order starting from top and proceeding down the hierarchical tree.

### VirtualWorldModel

VirtualWorldModel is a class containing the multi-agent system and the organisation ([Figure 23](#_bookmark34)). VirtualWorldModel instance communicates with Synchroniser class instance.

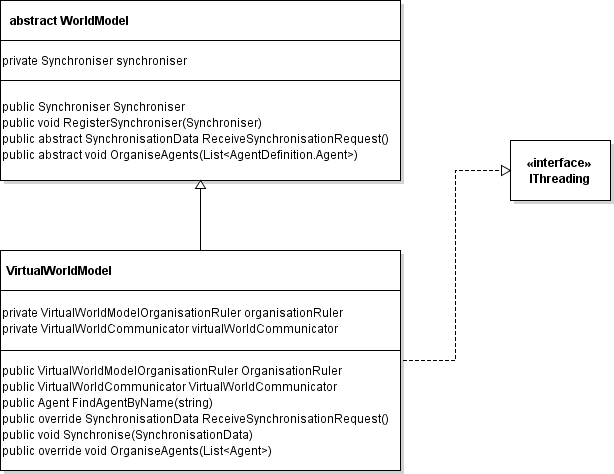


Figure 23. VirtualWorldModel class diagram.

### VirtualWorldModelOrganisationRuler

OrganisationRuler instantiates the VirtualWorldModelOrganisationRuler class ([Figure 24](#_bookmark36)) which contains methods for synchronising the domain if Synchroniser finds anomalies. SendGoal- function is a way of telling an Agent- instance in the organisation to which state it should change itself.

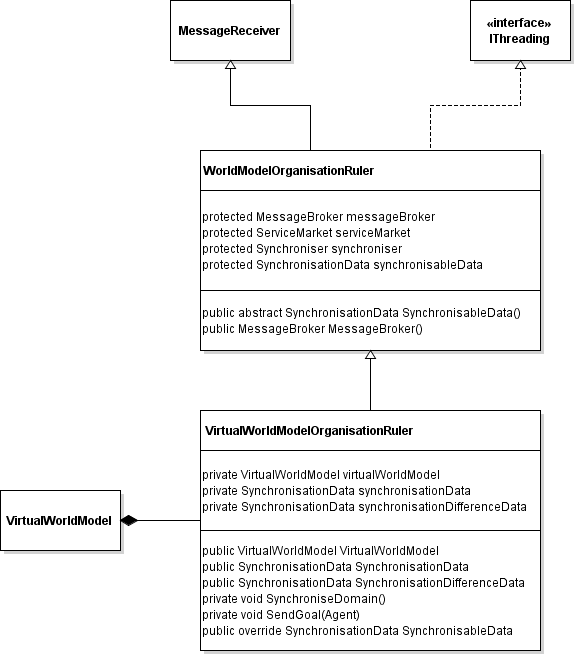


Figure 24. VirtualWorldModelOrganisationRuler class diagram.

### MessageBroker

When an agent or other entity wants to send information to another object, it sends a message to the MessageBroker ([Figure 25](#_bookmark38)) via its FederalDelegate. The details of the receiver are included in the message, so FederalDelegate and MessageBroker know where to forward it. This kind of messaging structure allows deliverying messages to multiple entities, and entities can release themselves from the process of locating the receiver. MessageBroker has references to VirtualWorldModelOrganisationRuler, ServiceMarket and FederalDelegate instances, thus connecting the interaction lines of these objects.

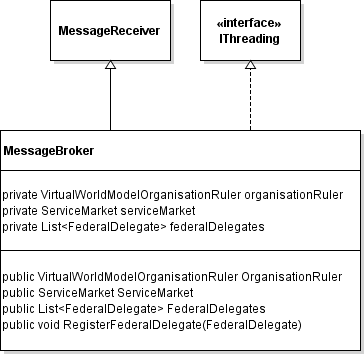


Figure 25. MessageBroker class diagram.

### ServiceMarket

ServiceMarket ([Figure 26](#_bookmark40)) belongs to the communication layer. It is a place for agents to provide and subscribe for services. Each service provider registers at ServiceMarket as service provider and registers also the service provided. Similarly, each service subscriber registers at ServiceMarket as service subscriber and shares the description of service needed. ServiceMarket decides the allocation of services.

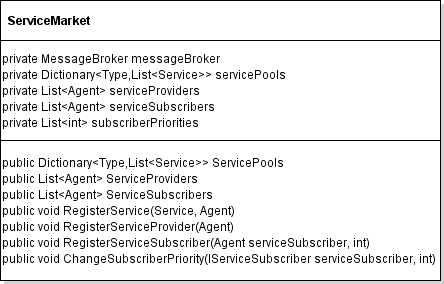


Figure 26. ServiceMarket class.

### VirtualWorldCommunicator

VirtualWorldCommunicator ([Figure 27](#_bookmark42)) communicates with VirtualWorld representation running on Unity. The communication is done using UDP protocol. VirtualWorldCommunicator listens states of the devices in VirtualWorldModel and reports the changes to VirtualWorld. When a sensor gets triggered in VirtualWorld, VirtualWorldCommunicator gets notified about the triggering and forwards the information to OrganisationRuler.

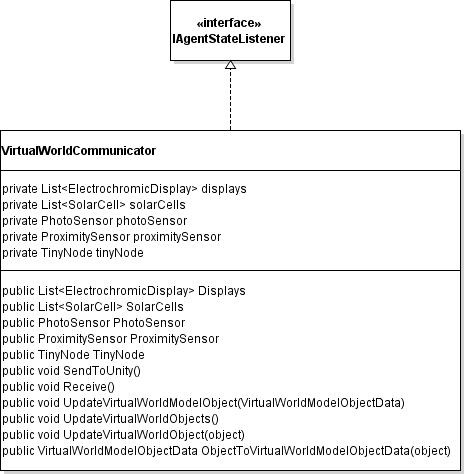


Figure 27. VirtualWorldCommunicator class diagram.

VirtualWorldModelObjectData ([Figure 28](#_bookmark43)) is the data format for the communication between VirtualWorldCommunicator and VirtualWorld. The state and energy characteristics are transferred. It provides Serialise- and Deserialise- methods for data format conversions.

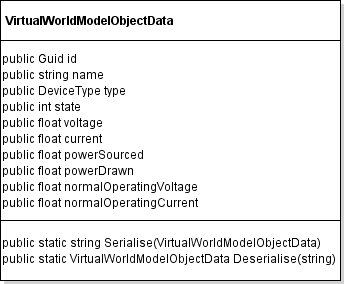


Figure 28. VirtualWorldModelObjectData class.

### FederalDelegate

FederalDelegate ([Figure 29](#_bookmark45)) acts as a representative for Federation and forms a link between MessageBroker instance and Agent instances in the Federation.through references to these instances. FederalDelegate implements IDelegateAgent interface ([Figure 30](#_bookmark46)) which is equipped with method bodies for registering and unregistering Agent instances. In this system electrochromic displays form a Federation, photosensor and proximity sensor form a sensing Federation, and solar cells as well as microcontroller has their own one.

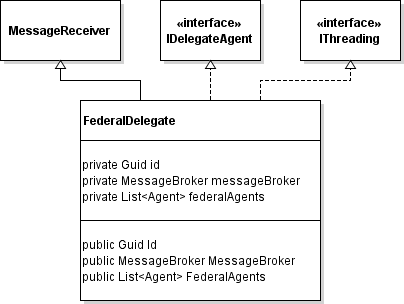


Figure 29. FederalDelegate class diagram.



Figure 30. IDelegateAgent interface.

### Devices

The devices in VirtualWorldModel are MicroController, SolarCell, PhotoSensor, ProximitySensor and ElectrochromicDisplay. Each of them inherit from Agent class which provides the individual thread-based operation capabilities and means of communication. The device diagram is presented in [Figure 31.](#_bookmark48)

MicroController implements IAgentStateListener interface through which MicroController can get notified when the sensor it listens changes its state. SolarCell inherits Agent class also directly, but Photosensor and ProximitySensor have Sensor class as common parent. There is also Display class as an additional abstraction layer between Agent and ElectrochromicDisplay.

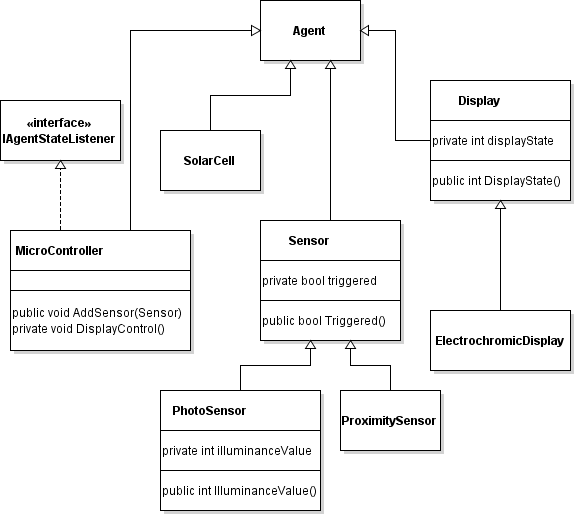


Figure 31. Device class diagram.

* + - 1. *Agent*

Agent class ([Figure 32](#_bookmark49)) is the basic active object in VirtualWorldModel. It contains description of its properties, methods to compare its state with other agents and communicate.

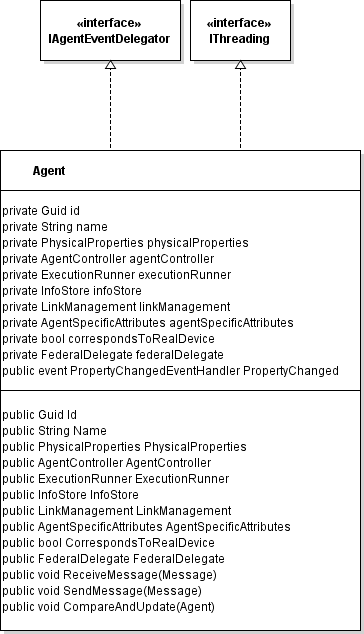


Figure 32. Agent class diagram.

The composition of Agent class is shown in [Figure 33.](#_bookmark50) The inner components of Agent class are AgentController, InfoStore, ExecutionRunner, LinkManagement and AgentSpecificAttributes. SensingElement was included in design, but was omitted in the implementation as sensing is implemented by listening the states of other Agents with Observer pattern. The pattern is formalised by IAgentStateListener and IAgentEventDelegator interfaces.

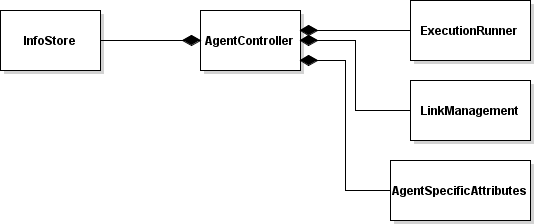


Figure 33. Agent composition diagram.

* + - 1. *AgentController*

AgentController ([Figure 34](#_bookmark51)) is the main decision component of Agent and runs on its own thread. When it makes decisions which lead to actions, it describes the actions as Goal instances, which describe the desired state of an agent. Here the state can refer to the Agent itself or other Agent, depending on the target of the action. The Goal is delivered to the target Agent instance which interprets the Goal and acts accordingly.

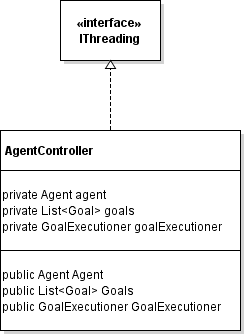


Figure 34. AgentController class diagram.

* + - 1. *InfoStore*

InfoStore ([Figure 35](#_bookmark52)) is a storage for information structures used. In this implementation it comprises an instance of ServiceDependency class ([Figure 36](#_bookmark53)) which contains information about the services the Agent provides, uses or needs.

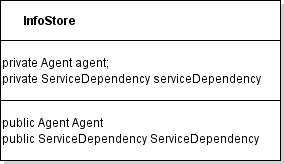


Figure 35. InfoStore class.

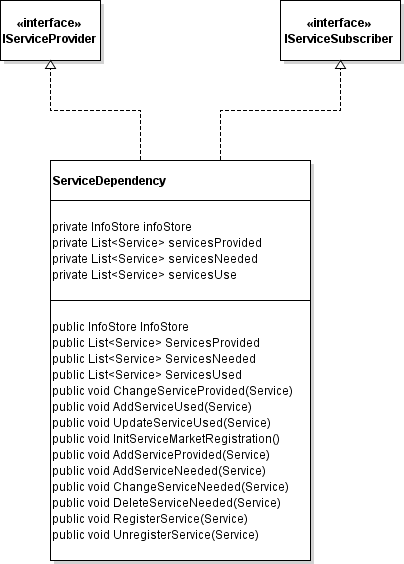


Figure 36. ServiceDependency class diagram.

* + - 1. *ExecutionRunner*

ExectionRunner ([Figure 37](#_bookmark54)) is a component that runs tasks called Executions in own thread. The asynchronous receiving of Executions is built upon abstrac body of ExecutionReceiver.

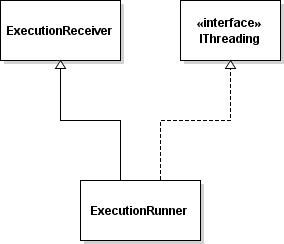


Figure 37. ExecutionRunner class diagram.

* + - 1. *LinkManagement*

LinkManagement ([Figure 38](#_bookmark55)) is a messaging component of Agent. When a message is sent to an Agent, message is sent to the Agent instance itself but Agent routes the messaging via LinkManagement.

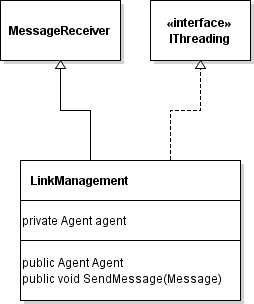


Figure 38. LinkManagement class diagram.

* + - 1. *AgentSpecificAttributes*

AgentSpecificAttributes class shown in [Figure 39](#_bookmark56) present the abstraction of the state of Agent. In this simple implementation only one state used. This state corresponds to for example the visual state of an ElectrochromicDisplay in the child class.

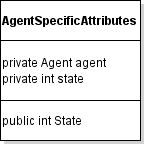


Figure 39. AgentSpecificAttributes class.

### Interfaces

IAgentStateListener ([Figure 40](#_bookmark58)) and IAgentEventDelegator ([Figure 41](#_bookmark59)) are interfaces related to listening the state of an attribute. IAgentStateListener is implemented by Agents wanting to get notified when the listened attribute changes its state. IAgentEventDelegator provides a method for the owner of the listened attribute to notify about the change in attribute.



Figure 40. IAgentStateListener interface.

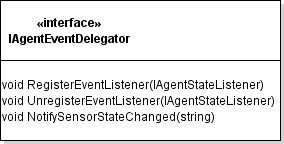


Figure 41. IAgentEventDelegator interface.

IServiceProvider and IServiceSubscriber ([Figure 42](#_bookmark60)) define interfaces for Agent to offer services to and request services from ServiceMarket.

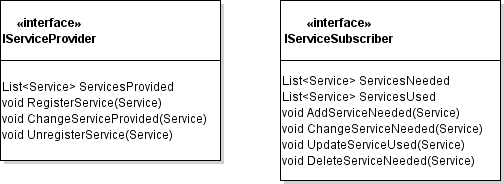


Figure 42. IServiceProvider and IServiceSubscriber interfaces.

IThreading is interface that is shown in [Figure 43](#_bookmark61). It defines method bodies needed to support threading. Run- method is task to be run in thread. ThreadRunnables- list lists the possible tasks for the subcomponents of the implementer of IThreading class. During startup of the VirtualWorldModel, listing the tasks of subcomponents this way, the hierarchical initalisation calls save time as they do not need to fetch the tasks to be run in their own threads.

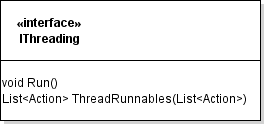


Figure 43. IThreading interface.

### AsynchronousReceiver

AsynchronousReceiver is a base class for MessageReceiver and ExecutionReceiver. This structure is needed for asynchronous operation of the multi-agent system and is shown in [Figure 44.](#_bookmark63)

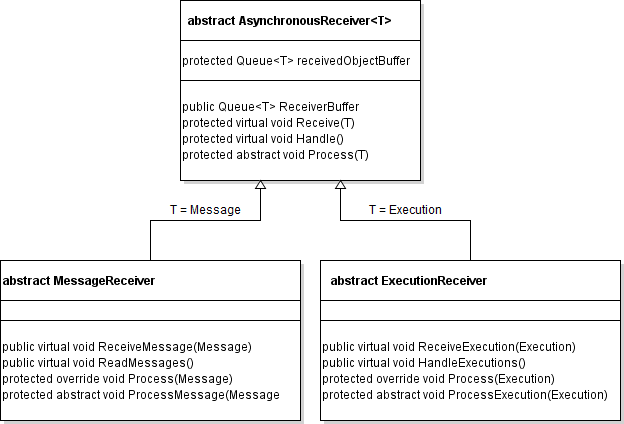


Figure 44. AsynchronousReceiver, MessageReceiver and ExecutionReceiver classes.

MessageReceiver was created in VirtualWorldModel and it produced a reusable and asynchronous communication protocol within the agent domain. Asynchronicity promotes the aspect of independency in operation: not relying on receiving a message brings autonomy [64, p. 271].

MessageReceiver corresponds to the Message queue handler in [Figure 11.](#_bookmark16) MessageReceiver consists of specialised methods for receiving certain information which are routed through a generalised Receive- method to the buffer. The buffer is emptied by Handler- method which directs the information to a general Process- method. Process-method is overridden by the implementers of this MessageReceiver structure. The structure of MessageReceiver is presented in [Figure 45.](#_bookmark64)

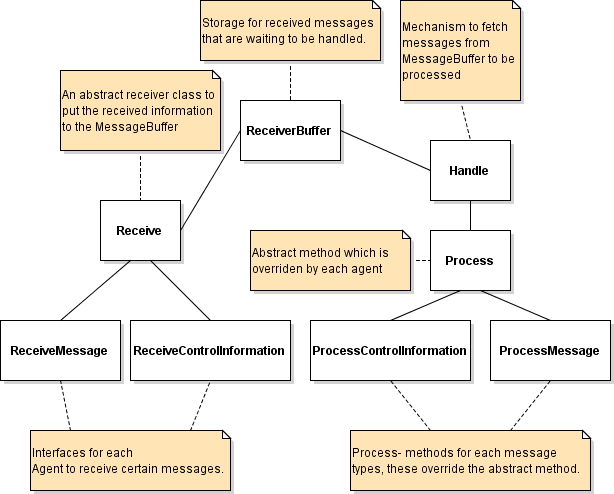


Figure 45. MessageReceiver structure.

ExecutionReceiver provides an asynchronous receiving structure to be implemented in ExecutionRunner. By using that Agent can put executable actions on queue to be executed in a separate thread.

## VIRTUALWORLD

VirtualWorld designed and implemented in this thesis was built upon an existing virtual representation of ResponseRoom. It was created and run on Unity game development platform. The original virtual room includes the tables, computers and fixed ceiling light sources. In this thesis virtual representation was implemented for electrochromic displays, control and energy measurement circuit, two sensors and solar cells. In addition, an indicator for measured illuminance levels was added and representation of control board, solarcells and all of displays included attaching energy meter next to each of them. An energy meter enables showing the actual energy usage of the corresponding device. However, in this thesis the real-time monitoring of energy characteristics for only solar cells and microcontroller was demonstrated. The implementation of virtual electrochromic displays and sensors included adding some functionality. 3D modelling was used to create virtual representations for the electrochromic displays and energy meters. The VirtualWorld setup is shown in [Figure 46](#_bookmark66).

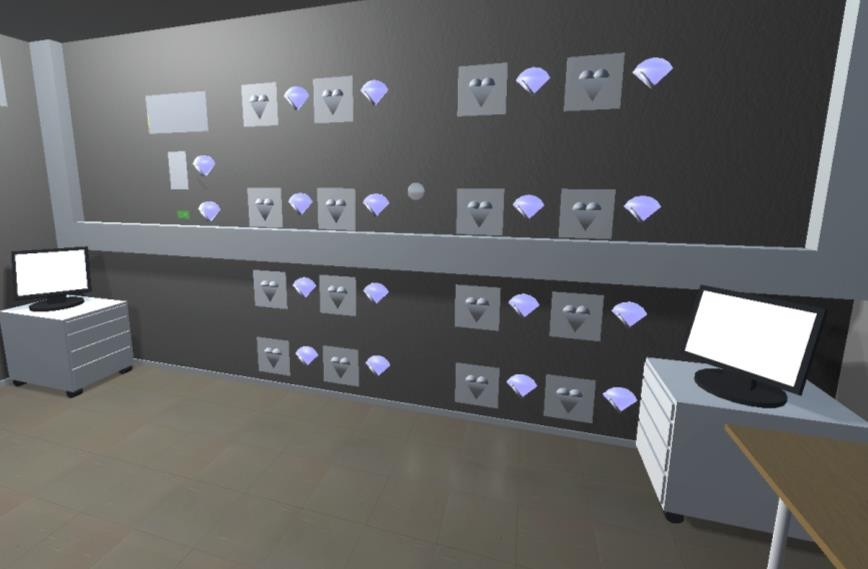


Figure 46. VirtualWorld setup.

There were a total of 16 virtual electrochromic displays ([Figure 47](#_bookmark67)) constructed. They tried to resemble the real electrochromic displays by looks, and four of them were in the same places than their counterparts in real world. Their only functionality was to change their visual state depending on the control orders received from VirtualWorldModel.

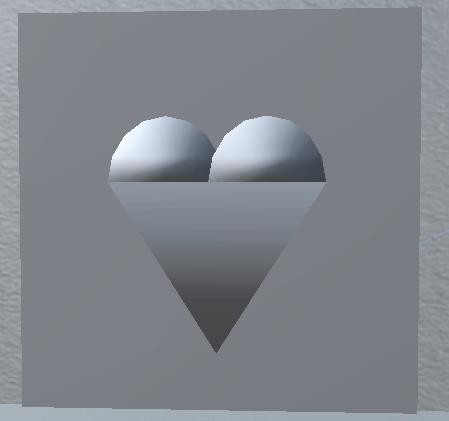


Figure 47. A virtual electrochromic display.

Control and energy measurement circuit was only a visual representation of the real microcontroller. The controlling of electrochromic displays and processing of the sensor values were managed in VirtualWorldModel. The virtual control board is shown in [Figure 48.](#_bookmark68)



Figure 48. The virtual control board.

There were two sensors constructed: photo sensor and proximity sensor. Photo sensor had its counterpart in real world, but it was too complicated to implement the accurate calculation of the illuminance based on virtual world lighting data. Thus the virtual photosensor was fed with the illuminance data measured by the real photosensor and the illumination level was displayed on a virtual display, named LuxIndicator ([Figure 49](#_bookmark69)). The proximity sensor ([Figure 50](#_bookmark70)) existed only in virtual world and it was a demonstration of a successful implementation of a virtual input device which is able to participate in the operation of a cyber physical system while not existing in the physical world.

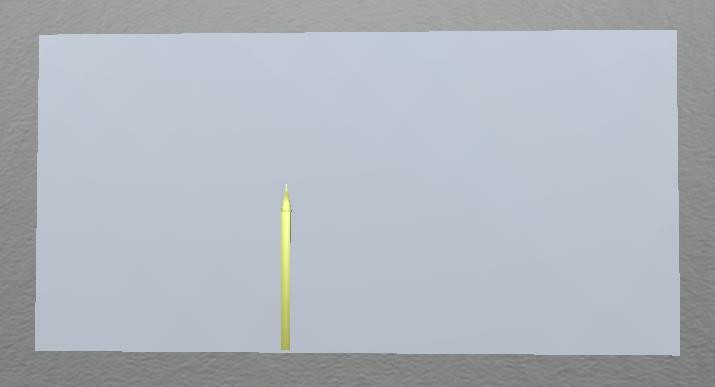


Figure 49. The virtual LuxIndicator.



Figure 50. The virtual proximity sensor.

Solar cells had only a static visual representation in the virtual network in which they were joined together as shown in [Figure 51.](#_bookmark71) It was a blank placeholder, but the energy measurement indicator next to it displayed the energy information related to the cells.



Figure 51. The virtual solar cells were joined together in their virtual representation.

The energy measurement indicator ([Figure 52](#_bookmark72)) looked like a fuel meter. The pointer indicated the energy available to the device relative to its expected energy level. The background screen indicated by its colour whether the monitored energy level was at a low, medium or sufficient level relative to the nominal value. The energy meters related to the solar cells and microcontroller chip displayed the actual energy levels of the corresponding real devices. The energy meters in action are shown in [Figure 53.](#_bookmark73) The upper energy meter represents the energy level of solar cells and the lower one corresponds to the energy level of the microcontroller.

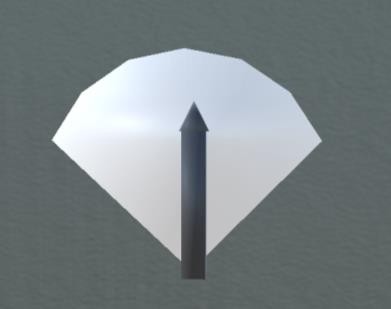


Figure 52. A virtual energy meter.

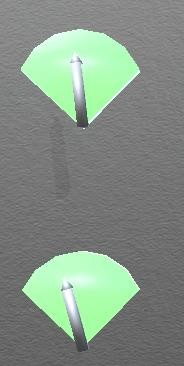


Figure 53. The energy meters in action.

[Figure 54](#_bookmark74) shows the virtual representations of the illuminance meter and the energy meters along with the solar cells and the microcontroller. The figure illustrates that it is feasible to compare the correlation between the measured illuminance value and energy values.

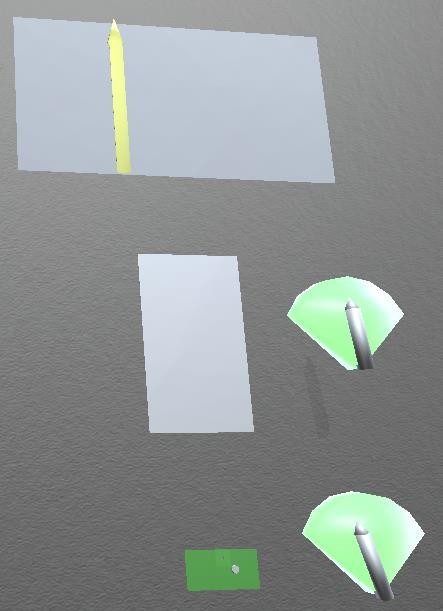


Figure 54. The correlation of illuminance and energy provided by solar cells.

## SYNCHRONISER

Timing diversions are the issues that must be solved, when two different systems produce common inputs to the controllers of the system and the controllers control the elements that are expected to act in a synchronous and consistent way. In this system, when a sensor gets triggered, its triggering is notified to the synchroniser, which conveys the trigger event to the controller in other domain. For example, the low-active interrupt line of the photosensor in Cyber-physical system is listened by the Arduino board which conveys the interrupt events to the VirtualWorldModel. The microcontroller in Cyber-physical system listens to this interrupt line, too, so VirtualWorldModel can inform the microcontroller about the interrupt originating from VirtualWorld by pulling the interrupt line down.

During the development of the system, it was assessed that when the real photosensor triggered an interrupt, the interrupt event was conveyed via VirtualWorldModel to VirtualWorld with such a little delay that the visual events resulting from the interrupt took place both in Real and Virtual Worlds simultaneously to the human eye. Thus no special arrangements regarding to synchronisation was created in that case. The user tests evaluated this decision later.

The **synchronisation problem** was solved in Synchroniser. When the virtual proximitysensor triggers an interrupt event, Cyber-physical system and VirtualWorldModel are notified by Synchroniser about the event. In this case Synchroniser has to send the notification to the VirtualWorldModel some time later than to the Cyber-physical system, so that the synchronicity is maintained. This delay is calculated by regularly measuring the time it takes a piece of information to go from Synchroniser to Arduino DUE board and back (green line in [Figure 55](#_bookmark76)). The same time measurement is done with information transfer between Synchroniser and VirtualWorld (red line in [Figure 55](#_bookmark76)). The difference between these time durations are calculated and divided by two to obtain the time how long the Synchroniser has to delay the start of sending the event notification to the VirtualWorldModel compared to that of event notification to Cyber-physical system.

The delay in the physical world was measured during runtime from Synchroniser just to Arduino DUE board and not to the microcontroller due to the limitations in the runtime communication of Cyber-physical system. This means that the runtime timing measurements were approximate. This choice fulfilled the requirements as can be seen in the Evaluation chapter.

Earlier studies have estimated neurons being able to distinguish the stimuli after 5-25 ms from the neuronal response [65, p. 730]. The analysis of how rapid timing differences human vision catches is not conducted here. The approach taken in this thesis was to ensure that the timing difference between the corresponding actions in Cyber-physical system and VirtualWorld remain below 5 ms.

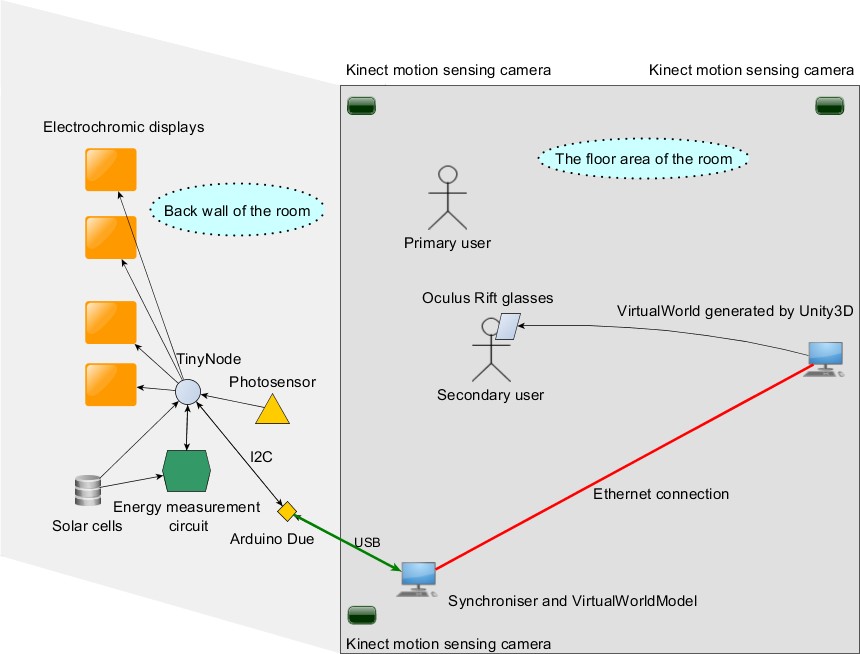


Figure 55. Timing paths for timing delay approximation.

Ensuring the consistency was done by comparing the states of the object representations of physical and virtual world objects. First, VirtualWorldModel is informed about the physical world objects which counterparts in virtual world should be matched. Then on a regular basis Synchroniser asks VirtualWorldModel to compare the states of the corresponding objects and if discrepancies is found, VirtualWorldModel changes the states of the virtual objects to conform the states of physical objects.

The **control order problem** in this system is related to the control actions initiated by interrupt events from photosensor or proximity sensor. If interrupts originating from both the real photosensor and virtual proximity sensor get triggered at the same time, the real photosensor is chosen. If they get triggered at different times, the first interrupt event received gets handled first. This means that every time the controller receives an interrupt it restarts the display flow sequence from the beginning. This choice was made the user interaction point in mind: the responsiveness of the system is maintained compared to the situation in which the user must wait the display flow to end before the system is able to register new sensor interrupt events.

Synchroniser class ([Figure 56](#_bookmark77)) includes references to object representations of physical and virtual objects. However, conforming the states of the virtual objects to the ones of physical objects was assigned to VirtualWorldModel. It is argued the inherent knowledge of the multi-agent system is needed to avoid state changes at the moments critical to the execution of individual agents. The class includes methods for time calibration needed for the synchronisation, event, comparison and update requests.

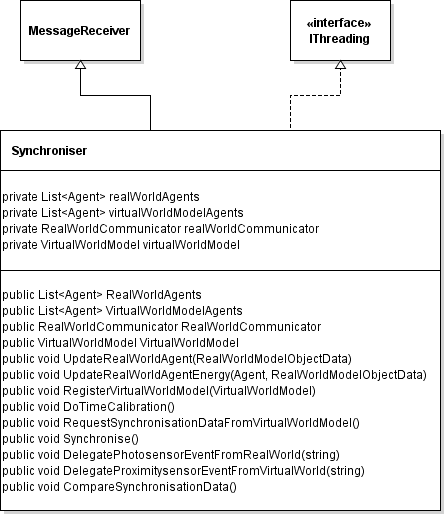


Figure 56. Synchroniser class diagram.

Synchroniser uses User Datagram Protocol (UDP) to communicate with VirtualWorld. The communication is done asynchronously through separate transmit and receive channels. The communication with Cyber-physical system is done through an Arduino Due board connected to a USB port of the computer which runs Synchroniser program. RealWorldCommunicator component ([Figure 57](#_bookmark78)) in Synchroniser is responsible for this communication.

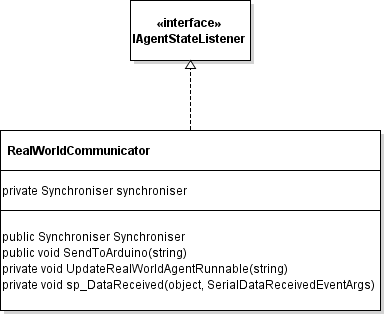


Figure 57. RealWorldCommunicator class diagram.

Synchroniser communicates with Arduino through serial port with a baud rate of 115200. Serial protocol was chosen as the communication channel in the Synchroniser – Cyber-physical system interface because the energy measurement data of the microcontroller circuit comes via serial channel. It was decided it is best to first transfer the measurement data through Arduino to the computer and take care of the data handling there. Synchroniser receives the measurement data from Cyber- physical system and stores the measured energy values of the solarcells and the microcontroller. The limitations in communication with Cyber-physical system is the reason for storing such a limited amount of data.

# EVALUATION

The performance of CPS can be evaluated by measurements, analytical methods or simulation. With complex CPS analytical methods may need complicated mathematical models. Simulation can be a feasible choice if creating a prototype of the system to be measured in different settings requires a lot of effort. [66, p. 265,268] Besides, simulating the software and hardware together is an option but calls for investments [67, p. 13]. Analytical methods are not well-suited for timing evaluation if the software does not define the timing properties formally [67, p. 14].

A technical evaluation consisted of two parts. Firstly, synchronisation was inspected by measuring the timing of the system. The system was simple enough to conduct the timing measurements. Simulation would have required lots of work and would have provided little benefit when the system in this thesis is measurable. The formal analysis was not suitable as the software did not specify the timing formally. Secondly, consistency of the system responses was assessed with empirical measurements of visually observable states.

Synchroniser synchronises the input events in the system by delaying the departure of messages to VirtualWorld. This synchronisation was evaluated by measuring the times for messages to go from Synchroniser to the microcontroller and back in reality (green lines in [Figure 58](#_bookmark80)) and from Synchroniser to VirtualWorld and back in virtual reality (red lines in [Figure 58](#_bookmark80)). The delay times from Synchroniser to these end points were deduced by dividing the times by two and comparing the results. The measurements were conducted by trying to interfere with the operation of the system as little as possible. Three separate runs of measuring the delays in the system were conducted. The few initial long delays were caused by a certain time needed for the system to configure itself and stabilise. When these delays are ignored, the average time from Synchroniser to microcontroller was at the longest

28.400 ms. The time from Synchroniser to VirtualWorld was at most 13.547 ms. The corresponding median values were 4.621 ms to the reality and 1.146 ms to the virtual reality. Thus, the difference between these average values is 14.853 ms and between average medians is 3.475 ms. These can be used to evaluate the delay in system response between the reality and virtual reality.

The median seems more appropriate when evaluating the overall delay due to the temporarily high delay values which raised the average very high. From the inspection of the measured time values, there were found some larger timing discrepancies, which is why the average value does not represent the system delay quite well. However, the high average delay shows that there were at times large delays in Cyber-physical system which may stem from limited communication resources. As conclusion, median value of 3.475 ms describes the system delay and it is lower than the 5 ms which was set as design requirement. High average delay reveals the occasional timing delays in Cyber-physical system.

Consistency evaluation was an empirical evaluation conducted by the system developer interacting with the system and trying to create different situations. Visual states of the displays in reality and virtual reality were monitored. Different kinds of interaction combinations were tried repeatedly. The knowledge of the developer about the system was utilised to find possibly challenging situations for the system. The test took 5 min 30s, and both the reality and virtual reality setups were recorded. Reality was recorded with a video camera and virtual reality with a desktop video

capture program. The videos were analysed afterwards side by side and observing the system responses in physical and virtual worlds and comparing their consistencies.

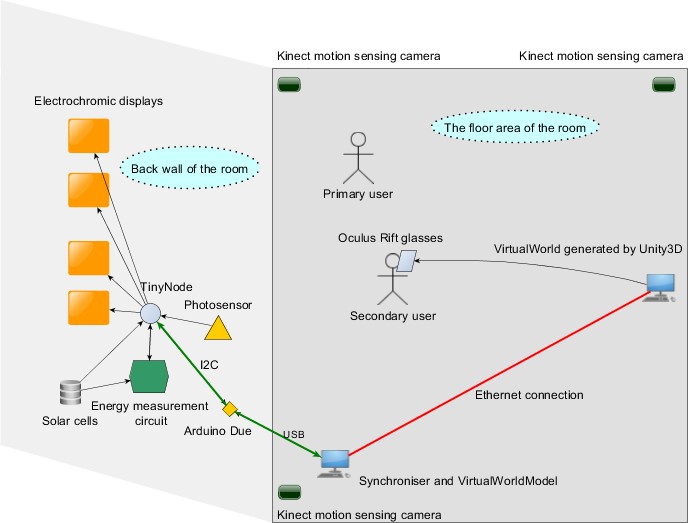


Figure 58. Timing measurement paths.

The system did not crash during consistency evaluation. The responses of the system to the user actions were mainly consistent. Triggering each of the two sensors activated the same display sequence in both domains. An example of a propagating display flow is in [Figure 59](#_bookmark81).However, a malfunction was found. If a vertical display column was active in virtual world when a new sensor triggering activated a new display flow, the new flow was started but the display column was left activated till the flow reached it and reset it. This was the only malfunction found in the test. This malfunction was reproduced to all the display columns by initiating the display flows when the display flow was active. This inconsistency was present only in virtual reality but did not affect the display flows at all in reality or virtual reality.

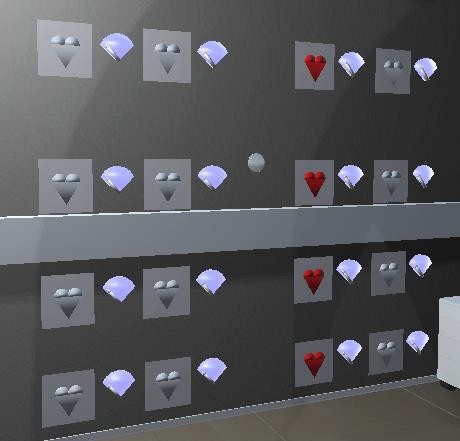


Figure 59. A propagating display flow.

# DISCUSSION

The system was designed to have a clear separation and independency over the specific interface implementations between the system blocks. This enables implementing the system for different physical device networks and promotes the extensibility. This was the impression during the development of the implementation of the system. Only thing that caused some peculiar problems through different blocks was the format of the measured energy data.

The system design defined only abstract Receive/Send- methods which are to be overridden by specific implementations. This is argued of making the same system design reusable with different physical device networks. It is admitted that this does not avoid a possible need of creating additional interface elements at the Cyber- physical system and Synchroniser interface, but at the system level the software blocks, namely Synchroniser, VirtualWorldModel and VirtualWorld, execute independently of the chosen interface. Cyber-physical system network operation may be affected by the chosen communication interface, if the communication uses a remarkably bigger portion of the resources than in this thesis. The system design presented in this thesis does not explicitly define predefined restrictions on the physical networks, as the aim of the system was to enable testing the real networks with a co-existing mirrored and augmented virtual presentation. The implementation introduced some restrictions. Inclusion of a real network to this kind of system should require as little modifications to the real network as possible, so to preserve the characteristics of the original physical network. The asynchronous communication structure was found enabling independent messaging between the blocks. However, the most critical data transfers, like delegating interrupt events, were transferred through synchronous calls.

The limitations in energy and the ways of communication in the physical setup indeed affected the real network operation in this thesis and lead to some simplifications. In order to allow the network to run at least for couple of minutes relying only on the energy harvested from the solar cells, the energy consumption of the microcontroller was cut to as low as possible. This decision resulted as making the UART communication single directional. This way the microcontroller could control the UART according to its own communication needs. Closing the UART when not needed avoided surplus current consumption of 1 mA. For more complicated embedded systems a more sophisticated approach is needed for the sake of handling many interrupt sources. In this demonstration the chosen solution is argued of being simple and effective.

The choice of using the single interrupt line was reasoned by the need to ensure the rapid delegation of interrupt events and the shortage of the other available communication channels. Additionally, from the perspective of the microcontroller, guiding the both interrupt signals to a single interrupt pin of the microcontroller simplified the interrupt handling and helped to solve the **control-order problem** in a simple way, which needed just a little amount of logic in the embedded program code. However, with more complex networks the system design should be refined to give more answers on how to solve the **control-order problem,** when there are multiple interrupt signals from Cyber-physical system and VirtualWorld present.

In this thesis VirtualWorldModel was reactive and mirrored the operation of the real devices at a simple level. VirtualWorldModel was modelled by a society of

intelligent agents on the basis of multiagent methodologies. This approach allows creating active and independently reasoning devices by modelling the as agents having increased amount of autonomy and own intentions.

Despite the focus was not on the visual credibility, the test ambience showed a nice consistency between the real and virtual worlds which can be seen in [Figure 60.](#_bookmark83) This consistency in lighting conditions was thanks to earlier development of the virtual environment in VTT, upon which this thesis was built. The virtual representation could be further refined to show visually the communication flows between the devices. Giving user the possibility of observing the data transfer related information (like the actual data, duration of the transfer and the amounts of energy and communication resources stressed by the data transfer) in VirtualWorld could be helpful in communication analysis and debugging.



Figure 60. The lighting conditions of real and virtual worlds were also consistent.

The linking between the real and virtual functionalities of mixed object does not mean the functionalities have to be exactly same. On the contrary, it is argued that it is useful to utilise the possibilities that virtual reality provides. Some sort of similarity between the functionalities is needed, at least the input and output behaviour should be consistent in mirrored representations. In future the user aspect will be included in evaluation of the system.

# CONCLUSIONS

In cyber-physical system computational elements control devices that have a tight connection to the physical world. Mixed reality brings together the physical world and an artificial imitation of the physical world created by computer.

This thesis presented how to design and implement a cyber-physical system with an interactive mixed reality implementation that combines real and virtual worlds in a way, which allows real and virtual devices in those worlds to co-exist and operate in a synchronous and consistent way. The stimuli that activate the devices can be initiated in both worlds. The real world setup was mirrored and augmented in the virtual world. The synchronicity and consistency were evaluated from the technical perspective.

Cyber-physical system was a physical device network in a test room. It consisted of a microcontroller, four electrochromic displays, a photosensor, two solar cells and an energy harvesting circuit. The microcontroller controlled four electrochromic displays according to the sensor data received from real and virtual sensors. The energy harvesting circuit stored energy from solar cells and provided it for the microcontroller. The energy provided by the solar cells and energy consumed by the microcontroller were measured. These measurements enabled real-time energy monitoring, concurrent comparison of illuminance values with the amounts of energy converted by solar cells and assessment of the efficiency of the energy harvester between the solarcells and microcontroller.

VirtualWorldModel was designed according to OperA multiagent methodology. The independently running entities communicated asynchronously and were thread- driven. A simple service provider/subscriber scheme for resource sharing was implemented.

VirtualWorld was an augmented virtual representation of the physical test room. VirtualWorld gave visual forms to the VirtualWorldModel objects in the mirrored copy of the test room. It was matched with the physical test setup and augmented with additional devices to demonstrate the feasibility of creating a synchronous and consistent upscaled virtual representation of the Cyber-physical system. Displaying the energy and illuminance information in a visual form in real-time brought added value compared to the setup in the physical world.

Synchroniser was designed to take care of timing and to keep the states of the Cyber-physical system and VirtualWorld consistent. It received the energy information regularly from the physical world and conveyed it to be presented in visual form in VirtualWorld.

Technical evaluation was conducted. Synchronisation was evaluated by measuring the timing delays from Synchroniser to physical and virtual worlds. The measurements revealed that system operated most of the time without breaking the timing conditions. Still, there were at times larger delays which showed that the system was not synchronous all the time. Empirical evaluation was used in consistency evaluation. The responses of the system to the user actions were mainly consistent. A malfunction leading to inconsistency between the states of virtual and real displays was found.

# REFERENCES

1. Ferber, J., Gutknecht, O. & Michel, F., “From Agents to Organizations: an Organizational View of Multi-Agent Systems,” in *Agent-Oriented Software Engineering*, 2004, vol. 4, pp. 214–230.
2. “What is Organization Centered Multi-Agent Systems (OCMAS) | IGI Global.” [Online]. Available: http://www.igi- global.com/dictionary/organization-centered-multi-agent-systems- ocmas/21439. [Accessed: 27-May-2016].
3. Wolf, W., “Cyber-physical Systems,” *Computer*, pp. 88–89, 2009.
4. Rajkumar, R. (Raj), Lee, I., Sha, L. & Stankovic, J., “Cyber-physical systems,” in *Proceedings of the 47th Design Automation Conference on*, 2010, p. 731.
5. Lee, E.A., “Cyber Physical Systems: Design Challenges,” in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, 2008, pp. 363–369.
6. Wan, K., Man, K.L. & Hughes, D., “Specification, Analyzing Challenges and Approaches for Cyber-Physical Systems (CPS),” *Eng. Lett.*, vol. 18, no. 3, pp. 308–315, 2010.
7. Balasubramaniyan, S., Srinivasan, S., Buonopane, F., Subathra, B., Vain, J. & Ramaswamy, S., “Design and verification of Cyber-Physical Systems using TrueTime, evolutionary optimization and UPPAAL,” *Microprocess. Microsyst.*, vol. 42, pp. 37–48, May 2016.
8. Fei, H., Yu, L., Athanasios, V.V., Qi, H., Rui, M., Yogendra, P., Ting, Z., Jiang, L., Xin, L. & Neal, N.X., “Robust Cyber–Physical Systems: Concept, models, and implementation,” *Futur. Gener. Comput. Syst.*, vol. 56, pp. 449– 475, 2016.
9. Zhuge, H., “Cyber-Physical Society—The science and engineering for future

society,” *Futur. Gener. Comput. Syst.*, vol. 32, pp. 180–186, 2014.

1. Bo, S., Xingshe, Z. & Mucheol, K., “Mixed scheduling with heterogeneous delay constraints in cyber-physical systems,” *Futur. Gener. Comput. Syst.*, vol. 61, pp. 108–117, 2016.
2. Lee, E.A., “Cyber-Physical Systems - Are Computing Foundations

Adequate?,” 2006.

1. Sha, L., Gopalakrishnan, S., Liu, X. & Wang, Q., “Cyber-Physical Systems: A New Frontier,” in *Machine Learning in Cyber Trust*, J. J. P. Tsai and P. S. Yu, Eds. Boston, MA: Springer US, 2009, pp. 3–13.
2. Pan, Z., Cheok, A.D., Yang, H., Zhu, J. & Shi, J., “Virtual reality and mixed reality for virtual learning environments,” *Comput. Graph.*, vol. 30, no. 1, pp. 20–28, 2006.
3. Tamura, H., Yamamoto, H. & Katayama, A., “Mixed reality: future dreams seen at the border between real and virtual worlds,” *IEEE Comput. Graph. Appl.*, vol. 21, no. 6, pp. 64–70, 2001.
4. Milgram, P. & Kishino, F., “A Taxonomy of Mixed Reality Visual Displays,”

*IEICE Trans. Inf. Syst.*, vol. E77-D, no. 12, pp. 1–15, 1994.

1. Schnabel, M.A., “Framing Mixed Realities,” in *Mixed Reality In Architecture, Design And Construction*, X. Wang and M. A. Schnabel, Eds. Dordrecht:

Springer Netherlands, 2009, pp. 3–11.

1. Billinghurst, M. & Kato, H., “Collaborative Mixed Reality,” in *International Symposium on Mixed Reality*, 1999.
2. Jung, S. & Hughes, C.E., “Pilot Study for Telepresence with 3D-Model in

Mixed,” in *Virtual, Augmented and Mixed Reality*, 2015, vol. 9179, pp. 22–29.

1. Wu, J., Mattingly, E. & Kraemer, P., “Communication in virtual environments: The influence of spatial cues and gender on verbal behavior,” *Comput. Human Behav.*, vol. 52, pp. 59–64, 2015.
2. Paul Milgram, H.C.J., “A Taxonomy of Real and Virtual World Display Integration,” in *Mixed reality: Merging real and virtual worlds*, 1999, pp. 1– 26.
3. Ricci, A., Piunti, M., Tummolini, L. & Castelfranchi, C., “The mirror world: Preparing for mixed-reality living,” *IEEE Pervasive Comput.*, vol. 14, no. 2, pp. 60–63, 2015.
4. Bridges, C., Hummel, J., Hursthouse, J., Moss, R., Society, A.C., Antilla, T., Book, B., Cheng, H., Constable, G., Dyson, E., Farmer, R., Garnett, G., Gruber, E., Keller, J., Marks, R., Poulin, M., Smart, J., Swords, J., White, P., Wilkes, I., Yee, N., Greenberg, J. & Hanke, J., “A Cross-Industry Public Foresight Project,” *Earth*, 2008. [Online]. Available: [http://www.metaverseroadmap.org/MetaverseRoadmapOverview.pdf.](http://www.metaverseroadmap.org/MetaverseRoadmapOverview.pdf) [Accessed: 01-Jun-2016].
5. Bradshaw, J.M., *Software Agents*. American Association for Artificial Intelligence & The MIT Press, 1997.
6. Huhns, M.N. & Singh, M.P., eds., *Readings in Agents*. Morgan Kaufmann Publishers, Inc., 1997.
7. Hayzelden, A.L.G. & Bigham, J., eds., *Software Agents for Future Communication Systems*. Springer Berlin Heidelberg, 1999.
8. Hayes-Roth, B., Brownston, L. & van Gent, R., “Multiagent Collaboration in Directed Improvisation,” in *Readings in Agents*, M. N. Huhns and M. P. Singh, Eds. 1997, pp. 141–147.
9. Stevens, J., Kincaid, P. & Sottilare, R., “Visual modality research in virtual and mixed reality simulation,” *J. Def. Model. Simul. Appl. Methodol. Technol.*, vol. 12, no. 4, pp. 519–537, 2015.
10. Singh, M.P., “Chapter 1,” in *Multiagent Systems - A Theoretical Framework for Intentions, Know-How, and Communications*, Springer, 1994, pp. 1–14.
11. Jennings, N.R. & Wooldridge, M., “Applications of intelligent agents,” in

*Agent technology*, Springer-Verlag New York, Inc., 1998, pp. 3–28.

1. Tecuci, G., *Building Intelligent Agents*. Fairfax: Academic Press, 1998.
2. Shoham, Y. & Leyton-Brown, K., *Multiagent Systems Algorithmic, Game- Theoretic, and Logical Foundations*, 1.1 ed. 2010.
3. Hayzelden, A.L.G., Bigham, J. & Luo, Z., “Multi-Agent Interactions for a Network Management System (Tele-MACS Approach),” *Softw. Agents Futur. Commun. Syst.*, pp. 160–182, 1999.
4. Shoham, Y., “Agent-oriented programming,” in *Readings in Agents*, 1997, pp. 329–349.
5. Haddadi, A., “Towards a Pragmatic Theory of Interactions,” in *Readings in Agents*, 1997, pp. 443–449.
6. Jennings, N.R., “On agent-based software engineering,” *Artif. Intell.*, vol. 117, no. 2, pp. 277–296, 2000.
7. Schmidt Jensen, A. & Villadsen, J., “A comparison of organization-centered and agent-centered multi-agent systems,” *Artificail Intell. Res.*, vol. 2, no. 3, pp. 59–69, 2013.
8. Premm, M. & Kirn, S., “A Multiagent Systems Perspective on Industry 4.0 Supply Networks,” in *Multiagent System Technologies: 13th German Conference*, 2015, p. 291.
9. Garcia, E., Giret, A. & Botti, V., *Regulated Open Multi-Agent Systems (ROMAS): A Multi-Agent Approach for Designing Normative Open Systems*. Springer, 2014.
10. Durfee, E.H., Kiskis, D.L. & Birmingham, W.P., “The agent architecture of the University of Michigan Digital Library,” in *Readings in Agents*, 1997, pp. 98–108.
11. Shoham, Y., “An Overview of Agent-Oriented Programming,” in *Software Agents*, J. M. Bradshaw, Ed. American Association for Artificial Intelligence & The MIT Press, 1997, pp. 271–290.
12. Ripolles, O., Simo, J. & Vivo, R., “Automatic Population of Scenarios with Augmented Virtuality,” in *Computer and Information Sciences II*, 2012, pp. 347–353.
13. Fiorentino, M., de Amicis, R., Monno, G. & Stork, A., “Spacedesign: A Mixed Reality Workspace for Aesthetic Industrial Design,” in *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, 2002, pp. 86–96.
14. Kruijff, E., Swan, J.E. & Feiner, S., “Perceptual issues in augmented reality revisited,” in *2010 IEEE International Symposium on Mixed and Augmented Reality*, 2010, pp. 3–12.
15. David Drascic, P.M., “Perceptual Issues in Augmented Reality,” *SPIE Vol. 2653 Stereosc. Displays Virtual Real. Syst. III*, vol. 2653, pp. 123–134, 1996.
16. Regenbrecht, H. & Schönfelder, R., “Position: Mixed Reality Engineering -

ProQuest,” *Adv. Automob. Eng.*, vol. 4, no. 1, pp. 1–2, 2015.

1. Schnädelbach, H., Galani, A. & Flintham, M., “Embedded Mixed Reality Environments,” in *The Engineering of Mixed Reality Systems*, E. Dubois, P. Gray, and L. Nigay, Eds. London: Springer London, 2010, pp. 57–78.
2. Croatti, A. & Ricci, A., “Programming Abstractions for Augmented Worlds,” 2015. [Online]. Available: [http://soft.vub.ac.be/AGERE15/papers/AGERE\_2015\_paper\_5.pdf.](http://soft.vub.ac.be/AGERE15/papers/AGERE_2015_paper_5.pdf) [Accessed: 21-May-2016].
3. Coutrix, C. & Nigay, L., “Mixed reality: a model of mixed interaction,” in *Proceedings of the working conference on Advanced visual interfaces - AVI ’06*, 2006, p. 43.
4. Coutrix, C. & Nigay, L., “An Integrating Framework for Mixed Systems,” in *The Engineering of Mixed Reality Systems*, E. Dubois, P. Gray, and L. Nigay, Eds. Springer London, 2010, pp. 9–31.
5. Lethbridge, T. & Laganière, R., “4.4 What is a Requirement,” 2005. [Online]. Available: <http://www.win.tue.nl/~mvdbrand/courses/se/0708/slides/College11022008H4>

.pdf. [Accessed: 20-May-2016].

1. Halligan, R., “Types of Requirements,” *Project Performance International*. [Online]. Available: <http://www.ppi-int.com/systems-engineering/types-of-> requirements.php. [Accessed: 20-May-2016].
2. Wang, R.Y., Kon, H.B. & Madnick, S.E., “Data quality requirements analysis and modeling,” in *Proceedings of IEEE 9th International Conference on Data Engineering*, 1993, pp. 670–677.
3. Krasner, G.E. & Pope, S.T., “A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System,” 1988. [Online]. Available: [http://heaveneverywhere.com/stp/PostScript/mvc.pdf.](http://heaveneverywhere.com/stp/PostScript/mvc.pdf) [Accessed: 28-May-2016].
4. Dignum, M.V., “A Model for Organizational Interaction: based on Agents, founded in Logic.” SIKS, 12-Jan-2004.
5. Wiederhold, G., “Mediators in the Architecture of Future Information

Systems,” in *Readings in Agents*, 1997, pp. 185–196.

1. Isern, D., Sánchez, D. & Moreno, A., “Organizational structures supported by agent-oriented methodologies,” *J. Syst. Softw.*, vol. 84, no. 2, pp. 169–184, 2011.
2. D’Inverno, M. & Luck, M., *Understanding Agent Systems*, 2nd ed. Springer- Verlag Berlin Heidelberg New York, 2004.
3. Minar, N., Hultman Kramer, K. & Maes, P., “Cooperating Mobile Agents for Dynamic Network Routing,” *Softw. Agents Futur. Commun. Syst.*, pp. 287– 304, 1999.
4. “Reading 20, Part 1: Synchronization.” [Online]. Available: [http://web.mit.edu/6.005/www/fa14/classes/20-queues-locks/synchronization/.](http://web.mit.edu/6.005/www/fa14/classes/20-queues-locks/synchronization/) [Accessed: 24-May-2016].
5. “Asynchronous Messaging Primer.” [Online]. Available: https://msdn.microsoft.com/en-us/library/dn589781.aspx. [Accessed: 21-May- 2016].
6. Smith, D.A., Kay, A., Raab, A. & Reed, D.P., “Croquet - a collaboration system architecture,” in *First Conference on Creating, Connecting and Collaborating Through Computing, 2003. C5 2003. Proceedings.*, pp. 2–9.
7. Marescaux, M., “Electrochromic displays | Liquid Crystals and Photonics Group.” [Online]. Available: https://lcp.elis.ugent.be/tutorials/tut\_echrom. [Accessed: 08-May-2016].
8. “app\_uart library example with low power mode,” *Nordic Semiconductor ASA*. [Online]. Available: https://github.com/NordicSemiconductor/nrf51- UART-examples/tree/master/ble\_app\_uart\_low\_power. [Accessed: 01-Jun- 2016].
9. Appleby, S. & Steward, S., “Mobile Software Agents for Control in Telecommunications Networks,” *Softw. Agents Futur. Commun. Syst.*, pp. 270–286, 1999.
10. Bodelón, C., Fallah, M. & Reynolds, J.H., “Temporal resolution for the perception of features and conjunctions.,” *J. Neurosci.*, vol. 27, no. 4, pp. 725– 30, 2007.
11. Freitas, A.E.S. & Bezerra, R.M. da S., “Performance Evaluation on Cyber- Physical Systems,” *ICIC Express Lett.*, vol. 10, no. 2, 2016.
12. Lee, E., “Computing foundations and practice for cyber-physical systems: A

preliminary report,” 2007.