NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

MARIA EDUARDA PAIVA CONTINS

BSc in Computer Science and Engineering

TANGIBLE SENSOR-DRIVEN APPROACH TO ADAPTABLE COGNITIVE EXERCISES

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon

January, 2025

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

TANGIBLE SENSOR-DRIVEN APPROACH TO ADAPTABLE COGNITIVE EXERCISES

MARIA EDUARDA PAIVA CONTINS

BSc in Computer Science and Engineering

Advisers: Carmen Pires Morgado
Assistant Professor, NOVA School of Science and Technology
Rui Pedro da Silva Nóbrega
Assistant Professor, NOVA School of Science and Technology

Examination Committee

Chair: Hervé Miguel Cordeiro Paulino
Associate Professor, NOVA School of Science and Technology

Rapporteur: João Alfredo Fazendeiro Fernandes Dias
Associate Professor, IADE - Faculty of Design, Technology and Communication

Adviser: Carmen Pires Morgado
Assistant Professor, NOVA School of Science and Technology

Co-adviser: Rui Pedro da Silva Nóbrega
Assistant Professor, NOVA School of Science and Technology

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
January, 2025

Tangible Sensor-Driven Approach To Adaptable Cognitive Exercises

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.7) [31].

Acknowledgements

I would like to begin by expressing my deepest gratitude to my adviser, Professor Carmen Morgado, as well as my co-adviser, Professor Rui Nóbrega. Their invaluable guidance, insight, and help throughout the course of this dissertation were pivotal in its completion. Nova School of Science and Technology has been a place of growth and learning, and I am especially grateful to the Department of Computer Science for providing me with the resources and knowledge necessary to pursue my research interests.

I am also deeply thankful to my family for their unwavering belief in me, even when the path seemed uncertain. Knowing that I have made them proud means more than the completion of this work itself. The opportunity to pursue an education has been a gift, and I hope to continue to make the most of it in the years ahead.

To my partner, whose steady presence and support have been a constant source of strength, thank you. Without it, this dissertation would have been far more challenging. Here's to many more shared milestones in the future.

I am fortunate to have friends who have shared in this journey with me, motivating me to push forward and expand my horizons. Their company has made this process richer. I look forward to continuing our friendships and watching them succeed in their own pursuits.

And finally, an unexpected thank-you goes to my cats, who, in their own way, ensured I took breaks — whether needed or not — often by physically stopping me by sitting on my keyboard.

Abstract

This dissertation presents the research and development of a modular Tangible User Interface (TUI) system designed to enhance cognitive skills through interactive serious games. The proposed TUI system leverages sensorized physical objects to interact with games that target cognitive exercises such as memory, attention, and problem-solving. Unlike traditional cognitive rehabilitation methods, which are often expensive and rely on repetitive exercises, the TUI-based system offers a more engaging and adaptable approach.

A key focus of this work is the modular nature of the TUI. The system's flexibility allows for physical modifications, such as changing the types or numbers of sensors used, to accommodate different difficulty levels or even support cooperative gameplay. Furthermore, the TUI system is designed to be easily extensible, enabling new games to be developed with minimal changes to the existing setup, making it highly adaptable to various therapeutic and educational settings.

To validate the effectiveness of this approach, the TUI system was tested with two distinct serious games, each developed to demonstrate the modularity and adaptability of the interface. The empirical studies conducted involved a diverse group of participants to provide insights into the usability, user satisfaction, and overall effectiveness of the system.

The results from this research aim to demonstrate the potential of modular TUIs in creating customizable and engaging tools for cognitive development, providing a valuable complement to traditional methods of cognitive rehabilitation and education.

Keywords: Human-Compute Interaction, Tangible User Interfaces, Cognitive Rehabilitation, Cognitive exercise, Serious games

Resumo

Esta dissertação apresenta o desenvolvimento de um sistema modular de TUI projetado para melhorar as competências cognitivas através de jogos sérios interativos. O sistema TUI proposto utiliza objetos físicos sensorizados para interagir com jogos que visam exercícios cognitivos, como a memória, a atenção e a resolução de problemas. Ao contrário dos métodos tradicionais de reabilitação cognitiva, que costumam ser dispendiosos, exigir sessões presenciais e depender de exercícios repetitivos, o sistema baseado em TUI oferece uma abordagem mais envolvente e adaptável.

Um dos principais focos desta tese é a natureza modular da TUI. A flexibilidade do sistema permite alterações físicas, como a mudança dos tipos ou do número de sensores utilizados, de modo a acomodar diferentes níveis de dificuldade ou até mesmo suportar jogos cooperativos. Além disso, o sistema TUI foi desenvolvido para ser facilmente extensível, possibilitando o desenvolvimento de novos jogos com mínimas alterações na configuração existente, tornando-o altamente adaptável a vários contextos terapêuticos e educativos.

Para validar a eficácia desta abordagem, o sistema TUI foi testado com dois jogos sérios distintos, cada um desenvolvido para demonstrar a modularidade e a adaptabilidade da interface. Os estudos empíricos realizados envolveram um grupo diversificado de participantes, representando. Estes testes de utilizador, forneceram informações sobre a usabilidade, satisfação do utilizador e a eficácia geral do sistema.

Os resultados desta investigação visam demonstrar o potencial das TUIs modulares na criação de ferramentas personalizáveis e envolventes para o desenvolvimento cognitivo, proporcionando um complemento valioso aos métodos tradicionais de reabilitação cognitiva e educação.

Palavras-chave: Interação Pessoa-Computador, Interfaces Tangíveis, Reabilitação Cognitiva, Exercício Cognitivo, Jogos Sérios

Contents

List of Figures

Listings

Acronyms

ADHD Attention Deficit Hyperactivity Disorder *(p. 2)*
ASD Autism Spectrum Disorder *(p. 2)*
ATT Attribute Profile *(p. 50)*

BLE Bluetooth Low Energy *(p. 33)*

GATT Generic Attribute Profile *(p. 50)*
GUI Graphical User Interface *(pp. 2, 13)*

HCI Human Computer Interaction *(pp. 1, 6)*

PSSUQ Post-Study System Usability Questionnaire *(p. 19)*

RFID Radio-Frequency Identification *(pp. 31, 39)*

SPD Sensory Processing Disorder *(p. 2)*
SUMS System Usability MetricS *(p. 19)*
SUS System Usability Scale *(p. 19)*

TUI Tangible User Interface *(pp. iii, iv, 1, 2, 26, 77)*

UUID Universally Unique Identifier *(p. 50)*

1 Introduction

In an era marked by pervasive technology, the link between computers and the well-being of individuals has long become a topic of interest in Human Computer Interaction (HCI). As technology seeps into all aspects of our lives, the interest in exploring its potential in health and education has grown, and the field of HCI has been at the forefront of studying the impact of these kinds of technology.

Investing into novel approaches that harness this potential to improve the quality of life in individuals in areas such as rehabilitation, education, overall health and well-being is at a crucial juncture.

The potential of technology to enhance cognitive health and capabilities is a particularly promising area within this field. By examining the interplay between serious games and the TUI, we aim to explore this intersection by introducing a platform designed to promote the cognitive well-being of individuals, both for personal use and in rehabilitation or educational settings.

Our focus is cognitive exercises that promote the development of cognitive skills including memory, attention and problem-solving.

In this chapter, we present context and motivation for the work developed, followed by the objectives of the work, and finally, an outline of the document.

1.1 Context And Motivation

The use of technology in fields such as education, entertainment, and healthcare has become increasingly common. HCI is a design-oriented field that studies the interaction between humans and computers, and how to design these interaction interfaces.

Serious games [43] are a category of digital applications, in particular games, that have other purposes beyond those of entertainment, presenting a unique blend between elements of gaming and educational or therapeutic elements. Serious games can be applied to a broad spectrum of fields such as education, simulation, cognitive therapy and rehabilitation or physical therapy [42]. The term closely relates to concepts like e-learning, edutainment, game-base learning or exergames.

The use of serious games also has been shown to have positive effects when combined with several forms of therapy, as oftentimes, traditional methods can be repetitive and monotonous, leading to users having a hard time adhering to the prescribed exercises [23], this way, by introducing a gamified and interactive approach, users show better engagement and motivation.

To shape interactions between users and computers, we most commonly think of the use of a keyboard and mouse, or a touchscreen. These are defined as a Graphical User Interface (GUI), but these interfaces can take different shapes, such as a TUI, which is a distinct type of interface that have garnered attention in the HCI field.

These are defined as interfaces that leverage physical objects to interact with digital information and have been shown to have positive impact when compared to GUIs, as they use the intuitive interaction with material objects, utilizing their affordances to guide the type of interaction naturally.

Considering how pervasive technology has become in our lives, and how digital devices and games are so familiar to most of the population, there are reasons to explore alternative modalities to traditional forms of physical and cognitive therapy by leveraging the potential of serious games and TUIs.

Serious games for cognitive therapy and Rehabilitation are seen many times in the form of games that help improve attributes like perception, knowledge acquisition, skill acquisition, social and soft skills [10].

Certain demographics can especially benefit from this type of technology, such as children, neurodivergent people, the elderly, and people with disabilities (physical, cognitive).

Studies were also conducted to understand the effects of serious games on the elderly [47], looking into the possible improvements serious games can have in aspects like strength, health-related knowledge, self-efficacy, social interactions and communication skills.

Research on serious games for children with Autism Spectrum Disorder (ASD) [49] normally focus on communication skills and social behavior, imaginative skills, sensory integration and learning. Many children with ASD also have Sensory Processing Disorder (SPD), meaning they exhibit "hyper- and hypo-sensitivities in multiple domains" [32], with senses like vision, smell, auditory, and touch. Sensory integration games seem to be the most neglected subgenre of serious games for people with ASD, and TUIs show great potential here. For the designs of these games for children with ASD, it is important to provide customization options for parents or caretakers to better adjust the game to the child's needs.

Furthermore, individuals with Attention Deficit Hyperactivity Disorder (ADHD), specifically children, have also shown positive reactions and outcomes when using serious games as a form of therapy [50, 21], aiding with executive function, improving daily life skills, time management and promoting social interaction. The most characteristic symptoms of ADHD are inattention, and difficulty with concentration, so an engaging

game and a new way to interact with it can be a powerful tool to help cope with these symptoms.

In most cases, children's attitude towards learning is positively impacted by the use of games. While the content learned and its effectiveness may not differ significantly from traditional teaching methods, the engagement provided by the game aspect can be a powerful motivator. The same can be said for adults, as the use of serious games has been shown to have positive results in simulation training in work related settings [10], or with elderly people [47] in physical therapy and cognitive rehabilitation, as previously presented.

Using TUIs in combination with serious games can provide a more engaging and interactive experience, as the tangible objects bring a natural and intuitive interaction that uses our innate ability to interact with physical objects to help us understand and manipulate digital information. TUIs have helped in the development of cognitive skills, as fine motor skills, hand-eye coordination, and spatial awareness.

The use of TUIs had seen special impact in: Education: In the form of interactive learning tools. Museums and exhibits: interactive displays where visitors can manipulate physical objects to learn more about the exhibit. Healthcare: Rehabilitation tools for cognitive and motor skill development. Gaming and Entertainment: Augmented games and interactive installations that provide a more engaging experience.

1.2 Objectives

The goal of this work is to develop a platform featuring a set of sensorized physical objects that users interact with, alongside compatible use-case games that run on a computer. These sensorized objects serve as the input/output (I/O) interface for the games, tracking user actions through integrated sensors and providing feedback via actuators and the graphical user interface.

The use-case games are designed to offer exercises that stimulate cognitive skills. These games need to be easily configurable by the players, their caregivers, or other third parties supporting the player. They offer varying levels of difficulty using the same set of sensorized objects that form the TUI. The TUI itself is modular and can be physically adapted to suit the specific requirements of different serious games.

This work aims to develop a platform that can be easily extended to accommodate new games, settings, and scenarios, making it suitable for a variety of contexts, such as therapy, education, and health. The platform should be able to cater to a wide range of users while maintaining a consistent interaction paradigm, physical interface, and communication protocols.

The specific objectives of this work are to:

- Develop a platform to help improve and rehabilitate cognitive skills in diverse users, with potential applications in therapy, education, and health.

- Create a TUI with an embedded system of sensors and actuators to facilitate and enhance interaction with the digital components of the platform.
- Design a flexible and adaptable physical interface by coordinating modular components of the TUI.
- Implement a middleware layer that bridges the gap between the physical interface and the digital component, establish robust communication protocols that enable the interaction between the physical interface and the digital components.
- Provide a framework for easily adding new games and scenarios to the platform, enabling future expansions and adaptability. And for validation, develop two use-case games that stimulate cognitive skills, to serve as initial examples.
- Offer guidelines for implementing and using the platform in various settings and scenarios, including therapeutic, educational, and rehabilitative contexts.

1.3 Contributions

During this research, we designed and built a set of sensorized physical objects that interact with compatible digital games, using cards as the main physical objects for interaction. The platform is modular and adaptable, allowing for easy configuration and extension to various contexts.

This platform brings the following contributions:

- Creating a three layer platform that integrates sensorized physical objects, digital games, and a middleware layer connects the two.
- Design of two use-case games that stimulate cognitive skills, with customizable difficulty levels for diverse users.
- Provision of a framework for adding new games and scenarios to facilitate future adaptability and expansion.
- Comprehensive evaluation of the platform with a diverse group of users to assess its effectiveness and usability.

1.4 Document Outline

This document is organized in six chapters, after this introduction, in Chapter 2, we present a state of the art in the field of HCI, serious games, TUIs, and other related topics.

In Chapter 3, we present the platform developed and describe the architecture, methodology, hardware and software components, communication protocols, use case games and evaluation methods.

Chapter 4 focuses on the implementation of the platform, detailing the development of the hardware and software components as well as the use case games, and we modeled the communication protocols that allow the interaction between the TUI and the games.

In Chapter 5, we present the evaluation of the platform, metrics used, questionnaires and interviews, and a discussion of the results.

Finally, in Chapter 6, we draw conclusions from the work developed, reflect on the results obtained and present future work and possible future improvements to the platform.

2

Related Work

This chapter reviews the state of the art in the areas of study that are related to the subject in this document, overviewing relevant work that can serve as a basis of knowledge for the initial development of this project. To start, topics such as Human Computer Interaction (HCI), User Experience, and User Interface are defined and analyzed, with special emphasis on tangible and pervasive user interfaces. Then, the concepts of gamification and serious Games are presented, paying particular attention to the use of these concepts in the context of education and cognitive rehabilitation or exercise. And finally, the use of sensors and actuators is reviewed, with special attention to the use of these devices in the context of the development of TUIs and using *sensorized* objects as I/O interfaces.

2.1 Serious Games

Serious games have become a popular topic since they were first introduced in the 1980s [1]. Its definition is not unified, but it is generally accepted that a serious game is a game with a purpose other than pure entertainment, used normally for training, simulation, and education [42]. Particularly in education, much of the research leverages the popularity of video games to create engaging and motivating learning experiences [28]. Along with education, serious games are also used in the context of cognitive rehabilitation and exercise, motor control and coordination as discussed later in this section. Neatly tied with serious games is the concept of gamification, which is defined as the 'process of enhancing services with (motivational) affordances in order to invoke gameful experiences and further behavioral outcomes [24].

2.1.1 Motor Control And Therapy

In this section, different uses of serious games in the context of motor control and coordination are presented and discussed. For this, works that utilizes serious games that promote interaction with users through the use of body movements and gestures are specially relevant. These games are typically designed to promote physical exercise or support rehabilitation efforts, being that sometimes the repetitive and monotonous nature

Figure 2.1: An example exercise from TANGO:H for motor exercises [23]

of these type of exercises can make them hard to follow through, and serious games and in particular exergames can enhance patient adherence to rehabilitation programs [23].

For instance, in the works of Schönauer et al. [40] focuses on the use of serious games for rehabilitation of patients suffering from chronic pain. a system that enables patients to train motor skills through serious games that monitors body movements and biosignals.

C.S. González et al. [23, 22] presents a serious game, TANGO:H (Tangible Goals: Health), Figure 2.1 that is to be used to design active games to help with the rehabilitation of hospitalized children. This platform allows medical professionals to create personalized games to cater to the rehabilitation of different pathologies, and the characteristics of each user group.

The game is divided into two modes: Game, and designer. The game mode focuses on the interaction with the patient and having them perform the exercises Previously defined, and in designer mode, the medical professionals/specialists can lay out the exercise plan and objectives to be obtained by the patient when playing the game. The game can be defined to different types of exercise: physical, cognitive, and a hybrid of both. The physical exercises promote the stimulation of the users joints and muscles to perform specific movements. Figures are presented on a screen by the designer, with different colors and body parts that the user must utilize to successfully complete the task, Figure 2.2.

During gameplay, the user's 3D movements are mapped onto a 2D plane. The position of their joints is compared to predefined goals to evaluate task completion. If the position of all joints are satisfied, the task is considered complete, and the user can move on to the next task.

TANGO:H also provides a designer application that allows medical professionals and specialists to create personalized game sessions. The designer application provides a graphical user interface, where the specialist can define the type of exercise to be performed, shown in Figure 2.3. It can define the type of exercises: physical, cognitive, or a hybrid of both, and what tasks the user must perform to complete the exercise, along

Figure 2.2: Users of TANGO:H in the "Acaman" Acquired Brain Damage Day Center [23]

Figure 2.3: TANGO:H designer interface used to modify game modes/difficulty [23]

with the level of difficulty.

Additionally, the designer application supports a recommender system, that takes into account user's success rate in the previous exercises to recommend the next exercise, if the user has a high success rate, the next exercise will be more difficult, and if users have a low success rate, the next exercise will be easier. To help the specialist supervising the patient, the game also provides feedback on the user's performance in speed, equilibrium, coordination, and reflexes. This feedback to then better adapt the exercises to the user's needs.

To help fight the range of impairments that can be caused by stroke, Burke et al. [12], with the research group at the University of Ulster, propose several systems for serious games that promote upper limb rehabilitation following a stroke through the integration of 3D virtual environments and sensor and camera based motion tracking. Specifically, the use of the Nintendo Wii technology, webcams, and virtual reality.

Earlier, Burke et al. [12] presents a system using magnetic sensor-based VR equipment was used be able to track the movement of the user's arms and hands when performing

Figure 2.4: Orange catching game: players move a physical basket, that controls the virtual one, to catch the falling oranges on screen [12]

simple tasks such as reaching, moving, and releasing real objects. On top of this, game-like exercises were developed, one of which consisted of a task that required users to reach and grasp falling virtual objects (oranges), using a physical basket to catch them, this object is tracked and mapped to a virtual basket in the game, as can be seen in Figure 2.4.

To allow for a more personalized rehabilitation program, the system allows the therapist to define some parameters of the game, such as the speed of the falling objects, or individual size of the objects.

The other game-like exercise developed was an adaptive version of Whack-a-mole, but with mice, Figure 2.5, that aims to improve the user's reaction time, accuracy, and speed of the user's upper limb movement, along with the user's ability to perform visual discrimination tasks and selective attention. The game has two levels of difficulty, one where only one mouse appears and the played tries to hit it with a virtual hammer that is mapped to the user's hand position, and in level two, dogs may appear, and the user must avoid hitting them.

Similarly to the orange catching game, the specialist can define some parameters of the game, to specialize the game to the user's capabilities. And in addition, the system can also dynamically adapt the game's difficulty to the user's performance.

2.1.2 Cognitive Training And Rehabilitation

In this section, contributions to the topic of serious games that promote cognitive skills such as reasoning, working memory, sustained attention, visual recognition, planning and organization, will be presented.

In the published work of Seo Jung Yun et al. [48], a serious game to be used in early stages of dementia to address the impact of cognitive changes is presented. A VR interface system was developed with the purpose of training attention, memory, and executive

Figure 2.5: Whack a Mouse game: players try to hit creatures on screen [12]

Figure 2.6: Cooking and harvest game: players interact with virtual objects (food, tools...) to complete a recipe [48]

function in the elderly. The enriched environment focuses on real life mundane tasks such as cooking, as familiar imagery can stimulate recollection of memories in the elderly.

Through a HTC Vive head-mounted display, and a hand pose estimation in a virtual environment. Users are immersed in a harvest and cook game, Figure 2.6, set in a country house rural scenery. The harvest game emphasizes selective attention and working memory, where users must gather ingredients before time runs out; failing this, the game highlights the correct items.

The cooking game is also focused on working memory and sustained attention, but also intends to train executive function as well as spacial and procedure memory. In this game, users must follow a recipe that is verbally given to them. The recipe includes multiple steps, some of which may require ingredients harvested earlier. Similarly to the harvest game, if users do not complete the recipe in time, the game highlight the next step in the recipe by showing an arrow pointing at the next ingredients needed.

Both game modes can be played in two different difficulty levels, that are set by an external operator supervising.

Alejandra Ornelas Barajas et al. [7] introduced a serious game for children with autism spectrum disorder (ASD) that combines a GUI and a TUI system. The tangible interface

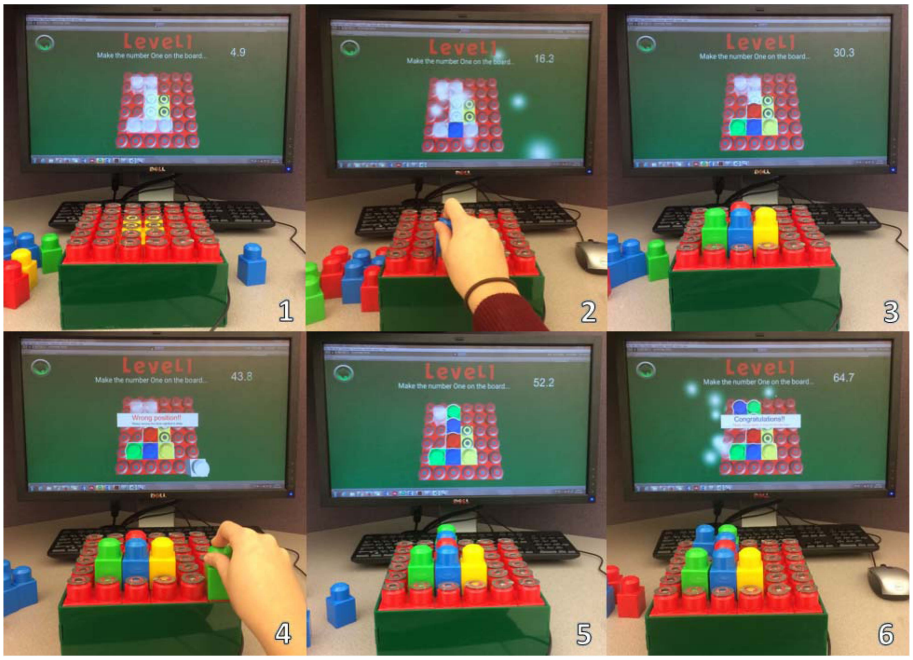

Figure 2.7: Example of Mini-games and feedback [44]

is built from Lego blocks with embedded electronic modules. This system was developed as a therapeutic tool to support the development of autistic children's social and cognitive skills, with different levels of difficulty.

The goal for the user is to replicate a model shown on screen, on the provided TUI, Figure 2.7. During the game, users are given real-time feedback and guidance to progress through the GUI, an example of this is when a block is placed in the right position and the corresponding block on the screen lights up, and the transparency of the block is reduced, Figure 2.7. Additionally, the system also provides instructive feedback when the user is placing the blocks in the wrong position.

The game was set to play in teams, and the system was designed to be used in a controlled setting, where user's progress was monitored, and adapt the game's difficulty to the user's capabilities. One teams played the game without the GUI, receiving cards with the model to replicate, and the other team played with the GUI, and at the end of the game they switched roles.

Social interaction, solitary play, collaborative play, engagement in other activities and performance were measured and compared between the non-computer and computer conditions, and concluded that the computer condition was more effective in promoting social interaction and collaborative play as well as increases in performance.

2.1.3 Education

Some literature has been published on the use of serious games in the context of education.

Terrenghi et al. [44] introduced the Learning Cube, the design and development of this learning appliance is motivated by research that indicates that physical interaction

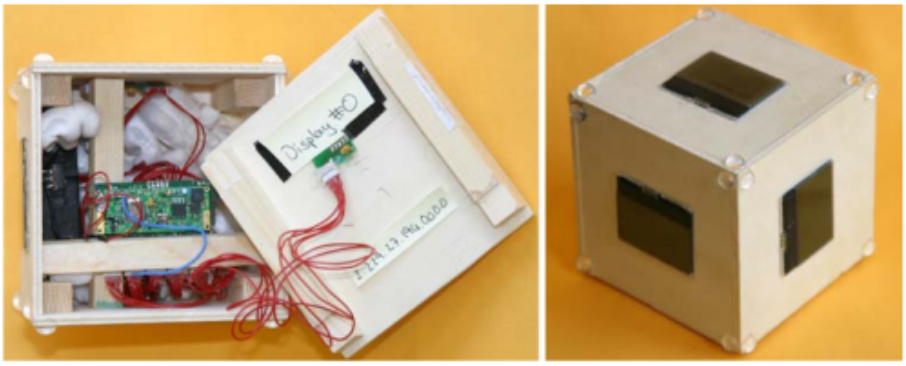

Figure 2.8: The Learning Cube [44]

with the world helps solidify children's learning and representational mappings [36] [37].

A cube was used as both a game interface and as a learning platform. The Learning Cube is a digitally augmented physical cube enriched with displays on all sides, and an embedded speaker inside, Figure 2.8. The main functionality of the Learning Cube is to provide multiple-choice options, shown on the built-in displays, and can be used to create a variety of applications such as: recognition of the same picture, word-picture association, vocabulary trainer, and matching multiple different 2D views.

With the vocabulary trainer activity, the word to be translated in shown on the currently top facing LCD, and the remaining LCDs will provide the possible translations. To select the correct option, to select it, the user shakes the cube as gesture input.

Another noteworthy use of the characteristics of a physical cube for learning is the exploitation of its shape for matching 2D views of an object. On the top facing side, a top view of an object will be shown, the remaining faces will show other views, but include a wrong or false one, which users must detect and select.

Sánchez et al. [39] presents a mobile serious game for eight grade students with the purpose of developing problem-solving and collaborative skills was presented. The game, in Figure 2.9, is divided into two modes, *MSG Evolution* and *BuinZoo&Museum*, both with the end goal of helping the players learn about species evolution. MSG Evolution consists of a 3D model of an environment where the players must try to keep the species alive against the environmental variables, and BuinZoo&Museum is a trivia game that guide a visit to a museum and allows the player to work on concepts related to the evolution of several species. In the visit to the museum, it is explained how fossilization happens, and then a challenge is proposed where players must explain the morphological, physiological and/or behavioral adaptations that took place in a species evolution, through a multiple choice quiz.

2.2 Human-Computer Interaction

The term "Human-Computer Interaction" (HCI) first became popular in the 1980s [15], with roots in the interactions of manual tasks where workers interact with machines. More

Figure 2.9: MSG Evolution [39]

recently the HCI field has been focused on computer science and system design. HCI encompasses the design, implementation, and evaluation of user systems.

HCI and interfaces are intrinsically related, the term "TUIs" (TUIs) were presented in 1997, by Ishii and Ullmer [26] as user interfaces that "augment the real physical world by coupling digital information to everyday physical objects and environments. TUIs have been gaining popularity within computing, and reflects an emphasis on physicality and environment in interaction design [33]. These are often compared to the more common GUIs, and research suggests that TUIs may have advantages over GUIs as it leverages the affordances of tangible objects, as Fitzmaurice et al. [20] presents.

2.2.1 Tangible And Pervasive User Interfaces

Building on the work of Terrenghi et al. [44], the cube's shape offers extensive handling and manipulation options, allowing for a variety of physical interactions to serve as user input. Previous work that makes use of these characteristics such as work conducted by Zhou et al. [51], where in hand with augmented reality, a tangible and intuitive interface that makes use of a foldable 3D cube to help with story telling is used, as seen in Figure 2.10.

The cube is used as a tangible interface for story telling, where each face of the cube represents a different part of the story, and the cube can be folded and unfolded to represent the current part of the story being told.

Likewise, Ryoicchi Watanabe et al. [46] utilizes the affordances of a cube to create a novel tangible interface, the ActiveCube. This cube allows users to construct and interact with a 3D environment using physical cubes, as seen in Figure 2.11. These connected cubes are used as bidirectional user interfaces, wand their position and orientation are tracked in real-time. Each cube is equipped with input and output devices, these provide the user with a variety of interaction methods.

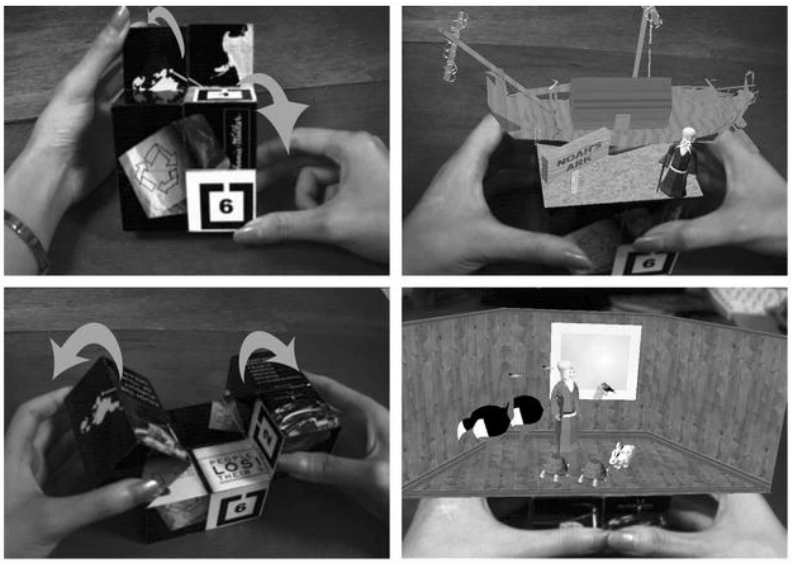

Figure 2.10: Foldable story telling 3D cube [51]

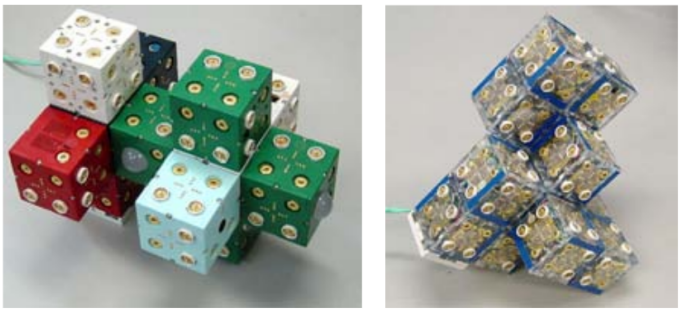

Figure 2.11: ActiveCube network [46]

Figure 2.12: ActiveCube network being modeled into a loop structure [46]

A network of cubes forms a 3D environment, as demonstrated in Figure 2.12, where cubes are arranged into a loop configuration. These networks can be expanded by incorporating additional cubes into the existing structure. Each cube is uniquely identified, using this ID, the system can determine the connection status of the cube, and the position and orientation of that cube in relation to the other cubes in the network.

Furthermore, a user can model a plane like shape with the cubes and the computer can recognize this shape, and will then display a virtual plane that is similar to the shape that has been modeled. Then users can interact with the virtual plane using the I/O

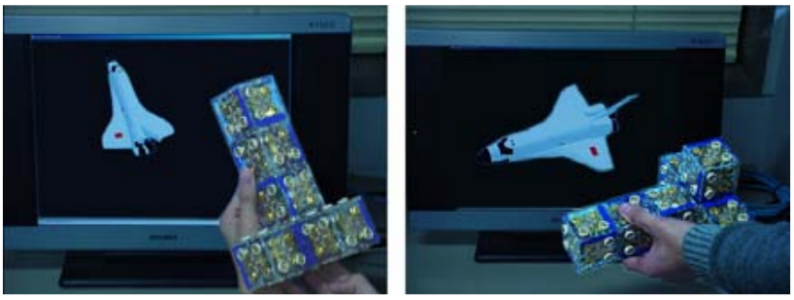

Figure 2.13: Interaction using a gyroscopic sensor cube [46]

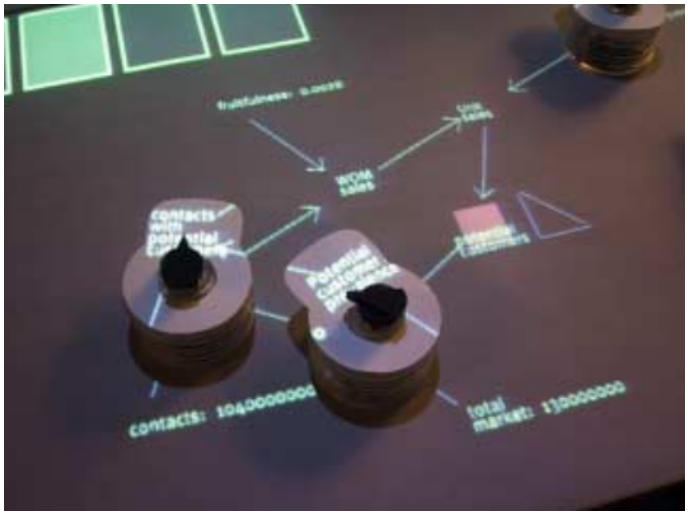

Figure 2.14: Sensetable interactive surface and pucks with modifiers/dials [35]

capabilities of the physical blocks, such as the height of the plane. In Figures 2.13, a cube with a gyroscopic sensor is used to control the inclination and orientation of a virtual object. The user may receive feedback from the system when, per example, the virtual object's height dropped too fast, this is done by the cube vibrating, a light turning on, or a sound being played.

In 2001, *Sensetable* was presented by Ishii et al. [35], consisting of a tabletop workspace for display and input of digital information, as seen in Figure 2.14. The tabletop workspace utilizes multiple low latency wireless objects being that can be manipulated users to interact with the system, which in addition, have dials or modifier tokens in tracked objects to map physical changes in the application.

To achieve this, a pair of modified Wacom tablets are used, these are placed next to each other to create a wider interactive surface. The mice used to interact with these tablets each have a serial number to identify them when switching surfaces. Information would be projected onto the surface of the table, and the user would interact with the system by manipulating the objects on the table surface using the pucks, and it is also possible to interact with an external display as seen in Figure 2.15.

Sensetable was used to create a variety of applications, one of them was a tool for teaching students about chemical reactions, as seen in Figure 2.14. The user can use the

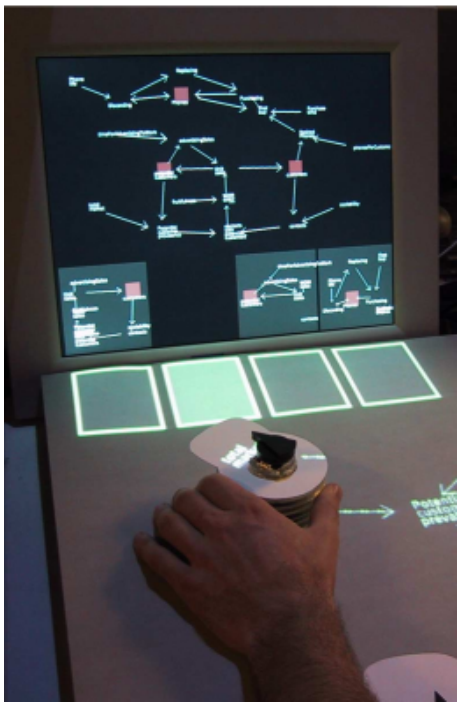

Figure 2.15: Sensetable interaction with an external display [35]

pucks to represent atoms or molecules, and then move them around in the interactive workspace. This is used to simulate chemical reactions, where users can bring the atoms and/or molecules needed for a particular reaction to occur. Here, the aforementioned modifier tokens can be used to change the electrical charge of the atom or molecule.

In 2007 Sergi Jordà et al. [27] presented the Reactable, Figure 2.16, a tangible and interactive luminous tabletop interface that allows musicians to create music by manipulating physical objects sitting on top it. Each object represents a component of a classic modular synthesizer, with dedicated function for generation, control and modification of sound. The result of these changes is immediately reflected on the tabletop, these projections convey information about the behavior of the objects, as parameter values and configuration, and lines connecting the objects.

To track the puck and user's fingers, the Reactable uses an IR camera that sits below the table. This way the system is not affected by any occlusions that may occur. This way, it is also possible to hide the camera and projector from the users, and allows to easily track many objects at the same time with and low latency, and independently form each other, which was a worry in the aforementioned work of Ishii et al. [35], that was circumvented differently.

The pucks provide different type of interactions that change different parameters, and are shaped differently depending on their function, for example a square puck would be a generator, a round puck would act as a controller, and a pentagonal puck would be a mixer, and so forth. The different parameter changing interactions are spinning the puck and dragging the finger around the perimeter of the object. These actions change

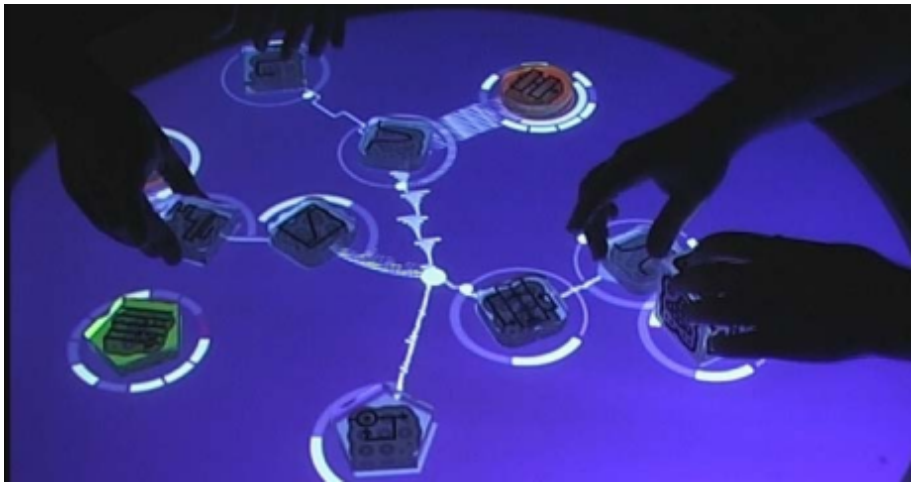

Figure 2.16: Interaction with Reactable [27]

different parameters depending on the type of puck. The different types of pucks can in turn interact with each other by being moved into proximity or contact with other puck, allowing for the creation of more complex soundscapes.

In a system with so much interaction between objects, and so many parameters to be changed, it is paramount to have very descriptive visual feedback to help monitor the object's state, and the Reactable achieves this by using a graphic synthesizer. The objects have an aura around them that is projected onto the table, that represent a parameter of that puck, and lines connecting the objects represent the real waveforms of the audio being produced, as well as using shapes that are informative of some type of interaction or parameter. Control lines are used to represent the density and intensity of the values being transported. In the case of oscillators and metronomes, or objects that vibrate at visible rates are enriched with a heartbeat-like animation.

2.2.2 Design Of Interfaces For Children

When designing interfaces for children, some extra considerations must be taken into account, as the cognitive and physical development of children is different from that of adults, and even vary greatly between different age groups. For instance, children have a shorter attention span, and may be more reluctant to follow instructions. With this in mind it is important to design interfaces that are intuitive and easy, and that are able to keep the child's attention for longer periods of time.

Several guidelines have been proposed to design interfaces for children both for TUIs and Graphical User Interfaces, like in mobile learning applications. These are guidelines on elements like navigation, text, image, content, color, audio, audio, input/output support and feedback. These guidelines were validated in Latiff et al. [29] by Human-Computer Interaction researchers, Early Childhood Education researchers.

Navigation should be natural and intuitive, located at the top or bottom of the screen and provide full control of the navigation mechanism, avoiding complex navigation

structures. The application should, when possible, avoid scrolling. Allowing to undo and redo options, and the user's reading level should be taken into account when designing menus and instructions.

Text should be clear and easy to read, with appropriate size, font, and contrast. Single story letters are normally preferred as they are what children are taught to read first. Font size change should be provided to allow for different reading levels, and visual problems.

As for images and icons, they play a crucial role in the design of interfaces for children, as they are the main facilitators of the child's understanding of the application. Icons should be visually meaningful, and graphics should be of cartoon style, with bright colors, and simple shapes. Providing choice of characters by the children is also a good practice, as it allows for a more personalized experience. Giving these characters a personality, ability to engage with the child, and express emotions is also important.

Content should use simple language. It is useful to provide introductory content: learning objectives, reinforcement of the content, games, and quizzes. When necessary, the application should provide tutorials before activities. Storytelling and characters teaching the content also have positive impact. Additionally, different difficulty levels should be provided, and have child's progress monitored, and achievements displayed to motivate the child. Finally, between learning activities, the application should provide a break, and allow the child to rest before initiating a new challenge.

Feedback is crucial in the design of interfaces for children to solidify knowledge and validate the child's performance. Feedback should be immediate, and provide positive reinforcement, and when possible, through characters and animations. To prevent frustration and demotivation, when dealing with errors, the application should provide hints and allow for redo options when a mistake is made.

Sonia Chiasson and Carl Gutwin [13] add some more principles: Interfaces should be strongly visual, as well as track the child's explorations of the game, it is important for children to remember previous interactions. They also suggest that children enjoy tangible interfaces, and that that direct manipulation aids in the exploration and discovery process. Finally, reward systems help to motivate children.

2.2.3 HCI Interface Evaluation

Evaluation techniques are used to assess the usability of a system in terms of how easy it is for users to learn and use the system, its efficiency, and user satisfaction.

There are several methods to evaluate the usability of a system, Figure 2.17, and a common one is user testing, a form of user-based evaluation [8], where representative users are asked to perform tasks and this interaction is observed, noting whether the user is able to complete the task, and how long it takes to do so, and note where they have difficulties with the interface. These tests should have defined goals and objectives, and have the user's experience can be gauges through the use of questionnaires, interviews, and observations.

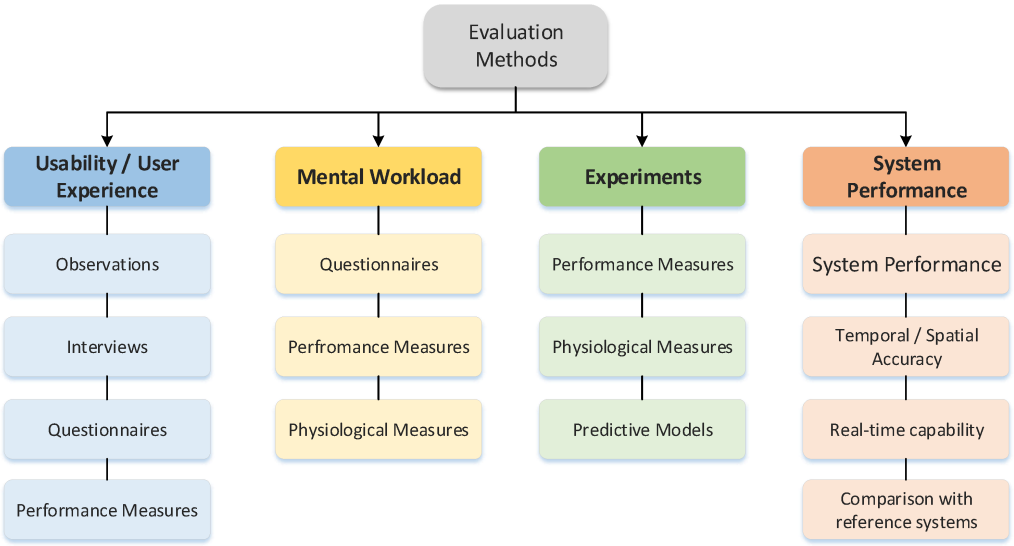

Figure 2.17: Evaluation Methods [6]

A popular questionnaire used to evaluate the usability of a system is the System Usability Scale (SUS) [11], used as a quick and reliable tool to provide a measure of user's subject perception the usability of an interface. It is a 10 item questionnaire that uses a 5-point Likert scale ranging from 'strongly disagree' to 'strongly agree', and after summing up the results, calculations can be made to determine a representative score (from 0 to 100) for overall usability of the system. The odd-numbered items have a positive tone; the tone of the even-numbered items is negative.

The Post-Study System Usability Questionnaire (PSSUQ) [30] gauges the user's reaction to the system being tested, similar to SUS. However, while SUS focuses on perceived usability and learnability, PSSUQ emphasizes perceived satisfaction. It employs a 7-point Likert scale, ranging from 'strongly disagree' to 'strongly agree', with an additional 'not applicable' option. The questionnaire is based upon System Usability MetricS (SUMS), and through rounds of improvements, the most used version of the questionnaire was crafted, containing 16 items, and produces four different scores: overall score, system quality, information quality, and interface quality.

When evaluating the usability of a system, as presented in Section 2.1 [34], it is important to consider the user's background and experience, including sociodemographic factors and previous experience with similar systems.

During user interaction with the system, observations on performance (including scores, mistakes, and time taken to complete tasks), effort, immersion, and presence can provide valuable insights into the user's experience. And finally, through the use of questionnaires, measure the user's satisfaction in terms of usability, learnability, and satisfaction.

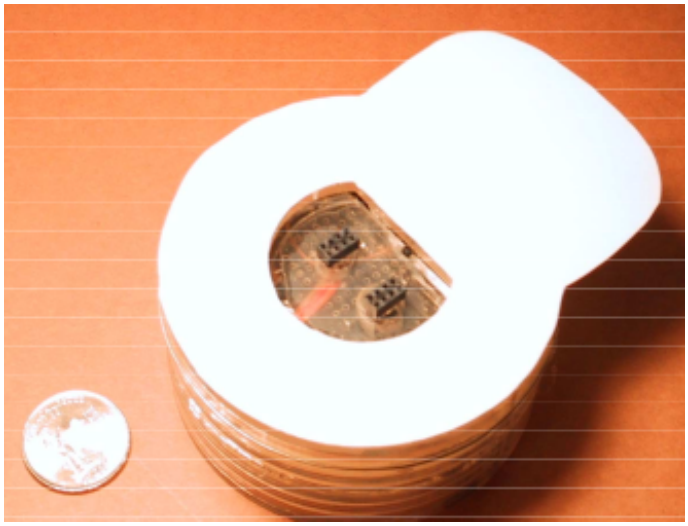

Figure 2.18: A Sensetable puck, with a socket for attaching a dial or modifier. A US quarter is shown for scale [51]

2.3 Ubiquitous Systems

Ubiquitous computing [25], also referred to as pervasive computing, refers to the idea of embedded computation and network communication into our environments and everyday objects, enabling them to form a network of interconnected devices. At times, to support this form of computing, it is necessary to use sensors, actuators, and microcontrollers to allow sending and receiving of data from the environment. A sensor is a device that detects or measures a physical quantity, and in the context of this work sensors with electrical outputs are the most suitable ones. The physical quantity being measured can be from different types of energy such as radiant, mechanical, gravitational, electrical, thermal, and magnetic. An actuator converts an electrical signal to some action, many times mechanical.

2.3.1 Sensors And Actuators In HCI

Recalling the work of Ishii et al. [35] evaluated in Section 2.2, and the pucks used to interact with the system, the sensing technology used as the table could only detect two pucks at a time, and to circumvent this issue, a duty cycle system was implemented where each puck was turned on one third of the time, but the pucks were also modified to include a circuit to sense when the puck is being touched, and then turn in on forcefully. To achieve this, capacitance sensor, Figure 2.19, that monitors an antenna wire wrapped around the circumference of the puck, Figure 2.18.

The microprocessor inside the puck detects a capacitance above a certain threshold, indicating that the puck is being touched. This solution allows the system to track these objects at a latency equal to the one of an unmodified Wacom tablet, while the rest, the ones being switched on and off through the duty cycle, are updated with higher latency,

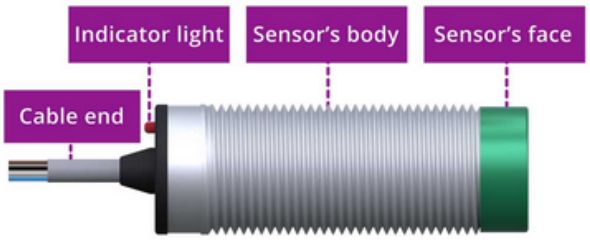

Figure 2.19: Capacitance sensor (https://www.realpars.com/blog/capacitive-sensor)

of less than one second. Capacitance sensors, or capacitive sensors [1] are electronic devices capable of detecting presence or proximity of solid or liquid targets, even without direct contact, and can be used for example for part detection and count in workstations or conveyors.

Burke et al. [12] also used sensors in their earlier work based on magnetic sensor-based VR equipment. In the Orange catching game presented in section Serious Games, the basket the user uses to catch the oranges has magnetic sensors attached to it, to allow the system to track its position. Similarly, in the Whack a Mouse game, the hammer used to hit the mice is tracked using a magnetic sensor that Block with LED matrix is attached to user's hand.

These magnetic sensor-based VR equipments use magnetic trackers [41], which are sensors that use magnetic fields, from a static transmitter, to determine the real-time positions of a receiver that is attached to the object being tracked. The transmitter generates a low-frequency magnetic field to be detected by the receivers, and that signal is then filtered and amplified to be sent to the computer for processing and calculation of the object's x, y and z coordinates.

The ActiveCube presented in Section 2.2 also uses sensors and actuators to allow for a variety of interactions with the user, having each cube is equipped with a sensor or an actuator as can be seen in Figure 2.20.

The ActiveCube uses sensors like gyroscopes, ultrasonic sensors, tactile sensors, sound sensors, and infrared sensors. They also have some cubes containing actuators such as vibration motors, LEDs, buzzers to emit sound, a fan, and a LED matrix (Figure 2.21).

Some examples of the aforementioned sensors being used are presented in Figure 2.13, where a cube with a gyroscopic sensor is used to control the inclination of a virtual spaceship, and the cube with the ultrasonic sensor is used to control the height of the objects in relation to a surface and the speed of the integrated propeller can also be taken into account, Figure 2.21. The direction of the spaceship can also vary depending on the inclination of the network of cubes.

[1] https://www.realpars.com/blog/capacitive-sensor, last accessed 01/2024

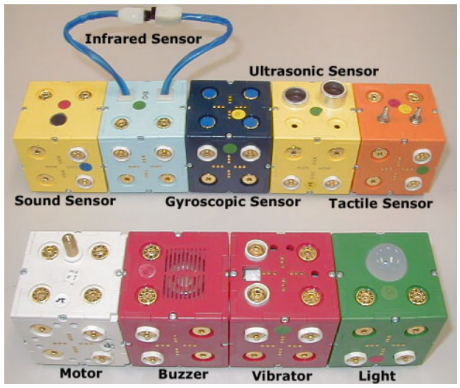

Figure 2.20: ActiveCube with input/output sensors [46]

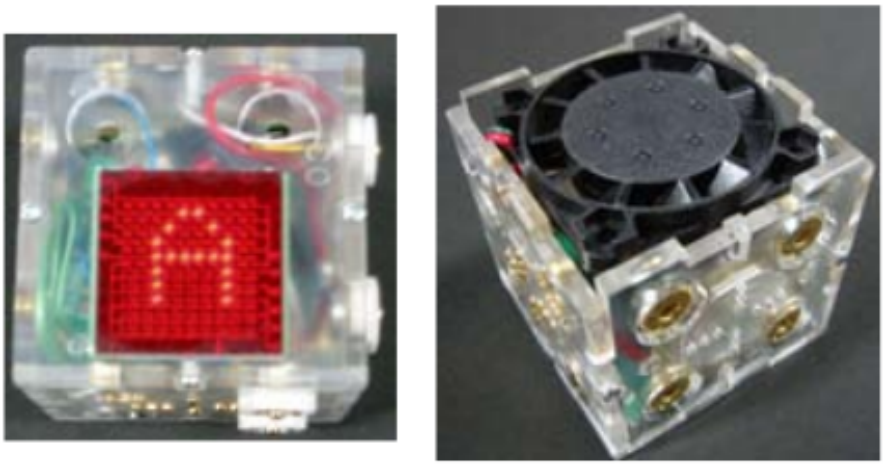

Figure 2.21: Block with LED matrix, and with propeller/fan [46]

Gyroscopic sensors [2] and function as devices that measure angular velocity, and are used to measure the inclination of an object, the velocity it is moving at, the direction, and if it is turning, and how quickly.

An ultrasonic sensor [3] is a device can measure the distance to an object, or presence of an object, using ultrasonic sound waves by sending a pulse of sound, and then listening for the echo when the sound bounces and converting it into a digital signal that is then used to calculate the distance to the object.

Alerts can be sent to users using the integrated actuators, such as the vibration motor, LEDs, and the buzzer. These were used to warn the user about certain issues, like when the virtual object's height drops too fast. The matrix of LEDs was used to display similar information to the user, by per example, displaying an SOS or DANGER message. In other contexts, these actuators can be used to provide different kinds of feedback such as the completion of a task, or an indicator of some kind of event, or system state.

[2] accelerometers: https://www.circuitbread.com/ee-faq/how-do-accelerometers-and-gyroscopes-work, last accessed: 01/2024

[3] Ultrasonic sensors: https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor, last accessed: 01/2024

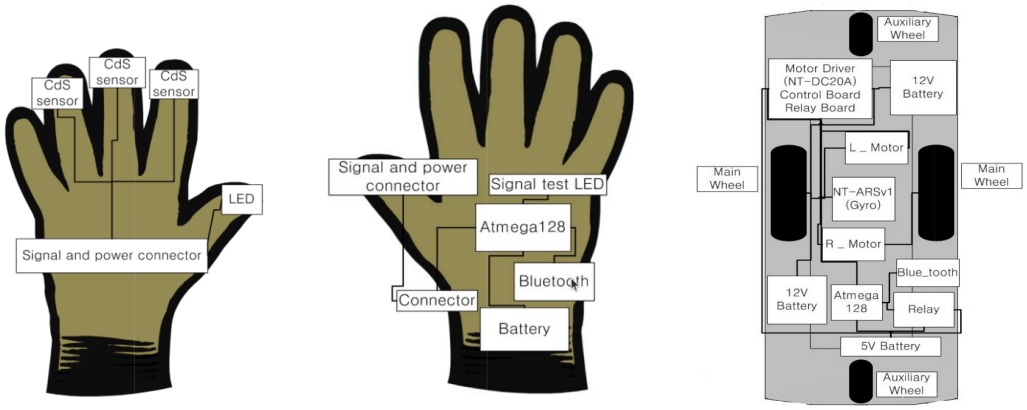

Figure 2.22: Glove and board hardware architecture [38]

2.3.2 Communication

Communication between the different components of a system can be done in a variety of ways, and the choice of communication method depends on the requirements of the system, and the hardware being used. For the work being developed, wireless communication is preferred, as it is less intrusive to the user. This section will go over some wireless protocols like WiFi and Bluetooth, and Zigbee.

Bluetooth is a wireless protocol designed for short range communication, connecting mobile and fixed over a distance of around 10 meters. It uses frequency hopping, in the 2.4GHz band, to avoid interference from other devices. Bluetooth is better suited for low power, and low cost wireless connection, and excels at short range communication of smaller amounts of data between devices, like with audio, data transfer and control. Bluetooth can have up to seven simultaneously connected devices. Bluetooth 5.0, the current version, can achieve data rate of around 2Mbps while this rate is increased to 24Mbps in Bluetooth Low-Energy [14], but these levels can be lower because of obstacles, distance and interference.

Jaemyung Ryu et al. [38] presents a system where a board robot is controlled wirelessly through a sensing glove. Figure 2.22 shows the design of the glove, it is equipped with sensors and an Atmega128 microcontroller with a Bluetooth module, which is powered by a 5V power supply. The board's direction is controlled by the user's finger movements through the sensing glove, that translates analog signals to digital signals, and then sends them to the board robot through the Bluetooth module. The board, Figure 2.22 is also equipped with sensors and actuators, batteries, and an Atmega128 microcontroller, with a Bluetooth module.

WiFi provides high speed wireless connection over long range distances using radio waves at 2.4GHz and 5GHz bands. WiFi has much higher data transfer rate and speed than Bluetooth. It is better used in portable access and ad hoc networking.

For instance, in the VR system created implemented in the Work of Seo Jung Yun et al. [48], the components communicate wirelessly using WiFi and Bluetooth protocols, as

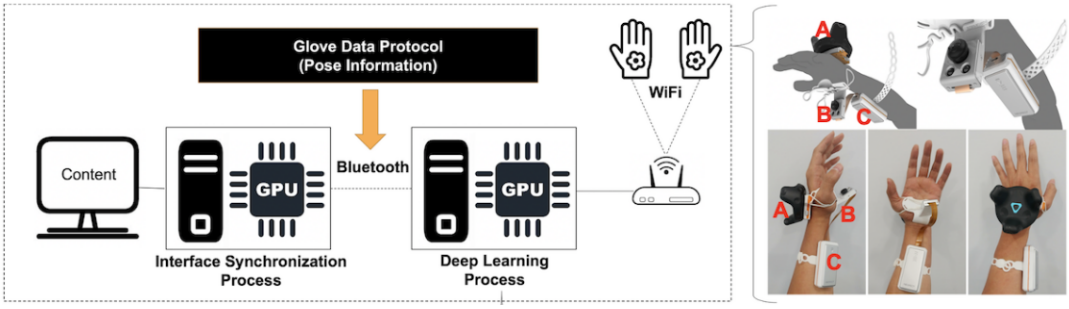

Figure 2.23: Communication between the different components of the VR system [48]

seen in Figure 2.23. Two hand motion tracking modules send real-time data of the hand and finger positions to the computer system using WiFi communication, where hand position and finger joint angles are calculated through a deep learning process, this data is then is sent to the interface synchronization interface process in the same computer system for the VR game through a Bluetooth connection.

Finally, Zigbee similarly to Bluetooth works in short range communication, and has low power consumption. Zigbee is a wireless protocol that accounts for very low data rates, up to 250 kbps, operating at 2.4GHz as well. It is better utilized in larger networks, the maximum devices connected depends on the hub or coordinator, ranging from thirty-five to more than one hundred devices. Zigbee is commonly used in IoT devices, home automation, sensor networks, and industrial control. The communication is normally periodic or intermittent. A drawback of Zigbee is that many devices do not offer supports for it.

2.4 Summary

In conclusion, research in Human-Computer Interaction has evolved to explore innovative methods for enhancing user interactions with digital systems. This literature review underscores the importance of considering user needs, preferences, and behaviors in the design of interactive systems. Interfaces that captivate and adapt to user requirements can significantly improve the user experience, particularly in serious games with sensitive contexts, such as therapeutic settings.

With this in mind, external control of the game's difficulty level becomes a key factor in the design of these serious games. The emergence of TUIs has introduced innovative ways of interacting with digital content, in engaging and natural ways, and this can bring a new dimension to serious games for rehabilitation and cognitive training. The integration of ubiquitous systems in interfaces alike can help with the design of a TUI with I/O capabilities, utilizing a various sensors and actuators to read user input and provide feedback. This feedback is incredibly important in the case of tangible interfaces, as the state of the system of the digital content must match and react to user input with as little latency as possible as to not break immersion.

To evaluate the effectiveness of interactive systems, researchers emphasize the importance of conducting user studies. Questionnaire-based tests, and case studies can be conducted to evaluate the effectiveness of the interface. Heterogeneous groups of users should be considered, and the system should be evaluated in different contexts, to ensure the system is robust and can be used in a variety of different settings.

3

Tangible Sensor-Driven Approach To Adaptable Cognitive Exercises

We propose a platform that promotes cognitive training, rehabilitation, and education by integrating serious games with the engaging capabilities of TUIs. It consists of sensorized physical objects that serve as a way to interact with digital serious games. These sensor-driven objects provide an intuitive and adaptable way for users to engage in cognitive exercises, making the training process more motivating and effective.

In this chapter, we will discuss the following:

- **System Requirements and User Needs:** An overview of the platform's development requirements, addressing the needs of various user demographics and potential use cases.
- **Development Methodology:** A detailed explanation of the incremental and iterative approach used in designing the TUI, developing the games, and creating the Middleware.
- **Proposed Solution and System Architecture:** A comprehensive description of the system architecture, highlighting how each component contributes to the overall adaptability, versatility, and usability of the platform.

3.1 Overview

Our platform is designed to address common limitations in existing serious games and TUIs, particularly their lack of versatility and adaptability. Many current solutions are closely tied to specific games or exercises, which can lead to repetitive usage and, eventually, decreased motivation among users. To mitigate these issues, our platform is built to be modular, configurable, and capable of supporting a wide range of scenarios, exercises, and user demographics, such as children with learning disabilities, adults with cognitive impairments, the elderly, and individuals recovering from strokes or brain injuries.

By developing a flexible and modular system that combines serious games with TUIs (TUIs), this work aims to create a valuable tool for cognitive training and rehabilitation that is both engaging and adaptable to a wide range of users and contexts. The TUI developed in this platform consists of sensorized playing cards that are read by a device with card slots; when cards are placed into these slots, the system interprets them and uses the input to influence the game's logic, making the interaction intuitive and dynamic. Cards pose a versatile and engaging way to interact with the system, as they can represent different symbols, colors, numbers, or letters, and the game logic can be easily adapted to the requirements of specific activities.

The adaptability of the platform ensures that it can evolve, incorporating new methodologies and exercises to continually enhance user engagement and effectiveness. By providing detailed guidelines for future developers and practitioners, this work lays the groundwork for a versatile framework that can be extended and customized for various cognitive rehabilitation, training, and educational programs. The system's open-ended design encourages collaboration across disciplines, that address the diverse needs of users, from children with learning disabilities to adults undergoing cognitive rehabilitation, thereby maximizing the impact and reach of the platform.

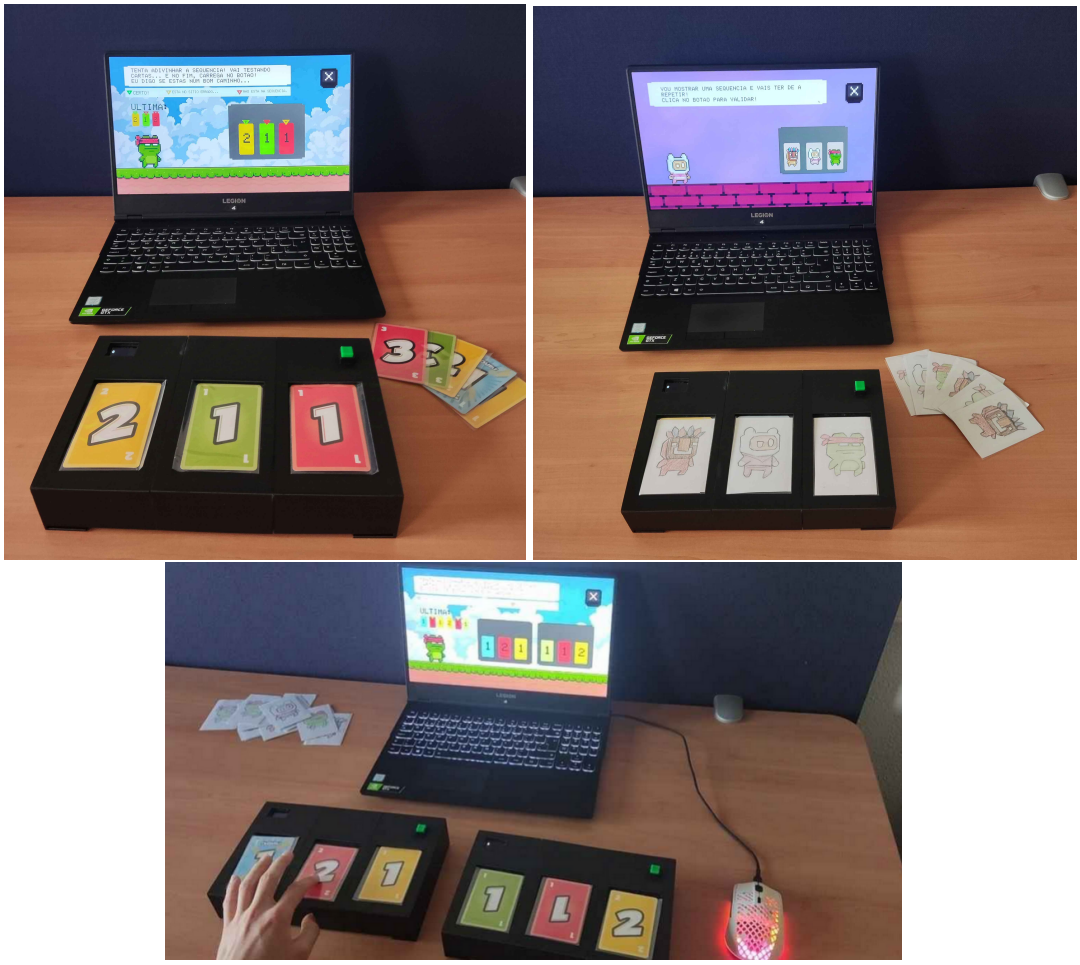

Figure 3.1: TUI being used in use case games

The system is composed of three main components: the TUI, the Middleware, and the game component. The TUI is the physical interface that users interact with, the Middleware acts as a bridge between the TUI and the game, and the game component provides the digital environment where the cognitive exercises take place. Figure 3.1 illustrates the interface and game components.

The key features of the developed platform include:

- **TUI Design:** At the core of the platform is a modular TUI composed of sensorized objects, specifically designed as playing cards that are read by a device with card slots. When cards are placed into these slots, the system interprets them and uses the input to influence the game's logic, making interaction intuitive and dynamic. The TUI can be easily customized by adding or removing modules and adapting the sensors to fit the needs of different exercises and games. This modularity provides flexibility in configuring the system to various difficulty levels, modes, or even cooperative interactions.
- **Middleware and Communication Protocols:** The platform includes a Middleware layer that acts as a bridge between the physical interface and the digital components (serious games). This layer handles the communication protocols that enable seamless interaction, ensuring a consistent and responsive user experience.
- **Adaptable Serious Games:** Two initial use-case games were developed to demonstrate the system's capabilities. The game side is designed to be configurable to support various difficulty levels and types of cognitive exercises. The games serve as examples of how the platform can be used to create diverse activities.
- **Extensibility and Versatility:** The platform is designed to be easily extended, allowing new games and scenarios to be added with minimal effort. The versatility of the card-based TUI system supports a wide range of applications, enabling developers to craft new game mechanics, cognitive exercises, or learning modules that utilize the same set of sensorized cards mechanism and the logic they embody. New modules or components can be added without altering the underlying system, allowing for scalable implementations that cater to individual or group activities, cooperative or competitive modes, and varying levels of difficulty. This flexibility reduces development time for new applications and ensures a consistent and familiar interaction experience for users.

3.2 User Requirements

It is important to define the three different types of users that will interact with the system: the **end-users**, the **practitioners/educators**, and the **developers**.

The **end-users** are the individuals who will be using the system for cognitive training, rehabilitation, or educational purposes. These users may have different cognitive abilities,

age groups, and backgrounds, so the system must be adaptable and engaging to them. The end-users will interact with the TUI, play the games, and receive feedback on their performance, so this interaction with the TUI must be intuitive and engaging. The tangible objects must be easy to handle, and it should be clear how to interact with the game. The platform should provide different levels of difficulty, feedback, and guidance. It must be an interface that does not have a steep learning curve, and that can be used by children and adults alike, while not being too simplistic for adults or too complex for children.

The **practitioners/educators** are the professionals who that may be using the system to guide the end-users in their cognitive training, rehabilitation, or educational activities. These users will be responsible for setting up the system and configuring the interface. The system must be easy to set up and configure for different users. The physical interface needs modular components that can be easily set up and used in different configurations.

Lastly, the **developers** are the individuals who will be extending the system, adding new games, exercises, or features. Adding new games or exercises should be easy, and require minimal changes, have reusable components, and make sure these users have the tools to extend the system in a way that is consistent with the rest of the system, without needing to have a deep understanding of the system's architecture or the way the hardware works, communicates and is configured.

3.3 System Requirements

For the system to be able to adapt to different contexts and demographics, it must be highly configurable and flexible, meaning that the tangible interface could be flexible enough to have a different number of sensors when needed, or different types of sensors, and the game should be able to interpret the data from these sensors without needing to be changed.

The behaviour of the tangible interface should be agnostic to the game or exercise being played, meaning that the game should be able to interpret the data from the sensors and provide feedback accordingly, without needing to change the interface or the sensors themselves, even when different games or exercises are being played.

The game side must be able to adapt to different levels of difficulty, and different demographics, provide real-time feedback, and the communication between the objects and the game should have relatively low latency.

The physical objects should be easy to set up and use, and have easy configuration on the game side. They need to be physically separated from the game device, therefore, the communication between the objects and the game must be wireless.

In terms of scalability, the platform developed must be able to extend for larger groups or multiple users with minimal changes to the system, and no changes to how the physical objects are used. Additionally, it should be easy to potentially integrate this TUI with other games or exercises in the future.

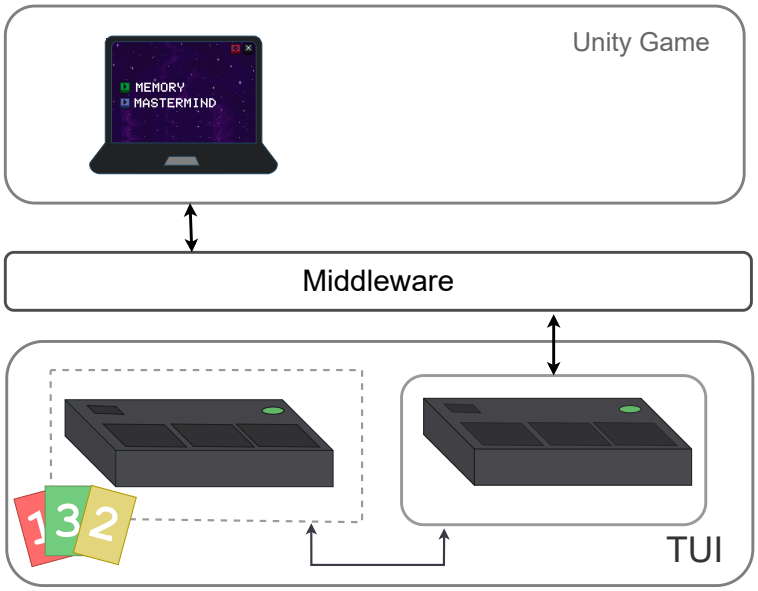

Figure 3.2: System components

3.4 Methodology

This research is based on an iterative design process, where the system is developed incrementally, and the design is continuously refined throughout the iterative process. The initial focus was on the design of the TUI, while having in mind the possible use case modes that could be implemented to test the effectiveness of that interface, using this information to design the system architecture.

The implementation started with the design of the physical objects, and the sensors and actuators that would be used, and then moving on to the Middleware that would bridge the TUI and the game, and slowly integrating the game component.

To validate the proposed solution, we conducted user-centered testing. The results of this testing were used to evaluate the usability, effectiveness, and user satisfaction of the system, as well and validating the adaptability and scalability of the system when creating new games or exercises.

3.5 Proposed Solution

We propose a three layer system, composed of three main components: the TUI, the Middleware, and the game component. Their interaction is illustrated in Figure 3.2. The user interacts with the TUI using cards that are read by modules with card slots. The cards used in the system will have different colors and symbols. The modules are boxes with three card slots, for the cards being played, a button for interaction, and a small OLED screen for feedback.

The number of modules used can vary depending on the game or exercise being played, but they are ultimately independent of each other and the user can set them up in any

order.

The choice to have the cards as the main component of the system was done to provide a simple and intuitive way for the user to interact with the games, that could be easily adapted to different semantic meanings and used in therapeutic and educational settings that require different types of inputs from the ones modeled in this work. Additionally, the same set of cards can have different meanings depending on the game being played, and the game logic can be easily adapted to the requirements of specific activities. The cards can be used to represent different symbols, colors, numbers, or letters, and this translation is done at the game level, not at the physical object level, through a configuration file that maps the card ID to a specific meaning.

To test the system we developed use case games that would help evaluate both the usability and effectiveness of TUIs, as well as the adaptability and scalability of the system. The use case games will require the sensor data from the modules to be interpreted them according to the game logic. They were designed to provide different types of cognitive exercises, such as memory and problem-solving, and proving that the steps taken to make different levels of difficulty and different types of exercises, are linear and easy to follow. These games are a memory game, and a puzzle game akin to the classic Mastermind game, as they work different cognitive skills. To play, the user, therapist, or educator, will have to set up the modules that will be used, and then, the user will have instructions on how to play the game.

Game interactions are the result of having users place cards on the modules' slots, and pressing the button to submit answers when needed. These use case games were designed using the Unity game engine.

The game maintains a communication line with the Middleware. This layer is running on the same device as the game. The Middleware layer bridges the communication between the physical objects and the game. Data sent received from the TUI's sensors is forwarded to the games.

The microcontroller is responsible for reading the sensors outputs, interpreting the data, coordinate the communication between the modules (when more than one is being used), and sending the data to the Middleware. The modules are equipped with Radio-Frequency Identification (RFID) sensors, that are used to read the cards unique ID, and a button that is used to interact with the game.

The same goes for the tangible interface, if the game is changed, the microcontroller should be able to forward the data in the same way as before.

The game must be able to run independently of the sensors used in the objects, so that if the devices (sensors, actuators, or microcontroller) are changed, modifications to the system should be minimal, only new configurations of the semantics of the information being sent to the game should be needed, and the game should be able to interpret the data from the new sensors as if they were the old ones, or differently if a new game is being created. This assumes the developer has the knowledge to adapt the microcontroller firmware, as we can not anticipate possible sensor changes.

If the game is changed to a new one, a new set of rules and interactions have to be implemented, but the physical objects and middleware, including all communication links, should be able to be used in the same way.

During the state-of-the-art research, we studied and compared different tangible interfaces, and serious games, and we believe that this approach brings a novel and adaptable system that can easily be extended to different contexts and demographics. Even though the system is designed with cognitive rehabilitation and exercise in mind, it can be valuable across various therapeutic and educational settings.

Such scenarios include:

- **Speech and language development**: Children could use the system to practice vocabulary, sentence structure, and pronunciation by using the cards to form sentences and words. With cards that have letters or words, the child could then press the button to hear the sentence or word being spoken, or have to write the word based on images shown on the screen. This activity would be aimed at improving vocabulary, sentence structure, pronunciation or articulation. This can also be used similarly for new language learners.

- **Social Communication skills**: Children could use the cards to role-play conversations or social situations, where a situation is presented on the screen, and the child must use the cards to respond to the situation, by practicing turn-taking, asking questions, or expressing feelings. This could help with improving pragmatic language skills such as initiating and maintaining conversations, understanding and using non-verbal communication, and understanding social cues. This could be especially useful for children with social communication difficulties such as those with autism spectrum disorder.

- **Math and logic skills**: Having a game where the cards have numbers or mathematical operations. The child could be asked to solve puzzles or equations by arranging the objects in the correct order, complete patterns, or solve puzzles. This can help introduce mathematical concepts in a fun and engaging way, and help with problem-solving skills.

- **Creativity and imagination**: Children could use the cards, representing characters, objects, settings, or actions, to create stories, role-play, or create their own story. This could help with creativity, imagination, and storytelling skills. The game could provide prompts or situations that the child must respond to, or the child could create their own stories or move the story along.

To allow this system to be extended to these scenarios, a way to facilitate the creation of new games or exercises must be added, like a way to easily add new decks of cards, or new configurations of new sensor outputs and their semantics, and mechanisms to simplify TUI configuration, and communication between the game components.

There is a configuration panel that allows the user to receive data from sensors and map it to different symbols or colors, to later be interpreted by the game. This step creates a JSON file that can later be used in new games or new TUI configurations for the same game.

3.6 System Components

The three components, TUI, Middleware, and the game component, Figure 3.2, are designed to communicate with each other to allow the user to interact with the game through the manipulation of the physical objects.

The TUI is composed of a set of sensorized objects that can be used to interact with the game. These objects, or modules, are composed of three card slots, sensors and actuators, and a microcontroller responsible for reading the sensors outputs, interpreting the data and processing it.

For the purpose of this work, we will use up to 2 modules, meaning up to 6 card slots, but the system should be able to be used with more modules if needed.

The communication between multiple modules is done through ESP-NOW [17]. This protocol allows for communication between microcontrollers with low latency and low power consumption. One of the modules will be the leader, elected upon TUI start-up, and this module will be responsible for forwarding the data from the all the other modules, and its own data.

The Middleware is responsible for interpreting the data from the sensors. For this layer to receive this data, it acts as a Bluetooth Low Energy client [16] that connects to the modules, and reads the data that was handled by the microcontroller on the modules, that runs a Bluetooth Low Energy (BLE) server.

This middleware runs on the same device as the game being played and connects to the game through a socket, and to the TUI via BLE.

BLE [4] was decided over other possible or similar protocols like:

- Wi-Fi [19]: would either require the tangible interface to be connected to a network, which may not be possible, as some networks, do not allow direct communication between devices in it due to security policies. Another option could be creating a separate Wi-Fi network to have both the sensorized interface and the game connect to, this however, adds a new configuration step, which adds complexity, as it assumes knowledge on the user end on how to set up a new network, which in its simplest form would be a Wi-Fi hotspot using a smartphone. This would require an additional device. Alternatively, the sensorized objects could host a Wi-Fi access point for the game device to connect to; but, this would prevent the computer running the game from connecting to the internet while the game is active.

Figure 3.3: Physical Mastermind game

- Bluetooth classic [16]: in terms of data transfer rates, latency, or range, both these Bluetooth would be usable in our platform, as communication is needed with seconds of interval normally, and it is meant to be used in proximity with the other device, but as it is a battery powered device, BLE stood out as a better option. Additionally, BLE's pairing and Generic Attributes [4] data structure mechanisms facilitated the connection and data transfer organization.

The game component has a virtual companion that guides the user through different games, and provides feedback on the user's actions. It is at the game layer that the data is effectively translated and interpreted in accordance to the game logic.

Delegating data translation from the sensors to the game logic enhances the system's adaptability and flexibility, as the same form of input, a card ID, or a button press can be interpreted in different ways depending on the game being played. That way no changes to the physical objects are needed when changing the game.

3.7 Use Case Games

We have chosen to implement a simple memory game, and a puzzle game, akin to the classic Mastermind game, seen in Figure 3.3, as they work different cognitive skills. Both these games could be used in cognitive rehabilitation scenarios or as cognitive stimulation exercises to be used in different contexts.

The games we chose to implement and design were chosen as they are simple to learn, and can be used to test the effectiveness of the system, and gather feedback from users on the usability and effectiveness of this form of interaction. These were designed to better

suit a younger demographic, as the system would be more engaging and motivating for children, although it is still suited for older demographics.

The games were created is a 2D environment, with a main menu where users can choose between the different game modes. After picking a game mode, the user may also define the difficulty mode. Before gameplay, the user may be prompted to activate the desired modules by placing cards on them, one at a time, in order.

Mastermind

The player must guess a sequence of cards by placing them on the card slots, and when ready to submit, press the submit button. The graphical interface will then show feedback: green if a card's position is correct, yellow if it is in the sequence but in the wrong position, and red if not in the sequence at all.

Game difficulty modes:

- Easy: Requires only one module (3 card slots).
- Medium: Requires 2 modules (6 card slots).

Memory

A sequence will be presented on screen and then the player will try to replicate it using the card slots.

Game difficulty modes:

- Easy: Requires only one module (3 card slots).
- Medium: Requires two modules (6 card slots), and the sequence is shown for the same amount of time as the easy mode.

A flow of interaction with a game like Mastermind would be as follows:

The game is started, and the Mastermind game mode is chosen from the main menu via the game interface. The user may be prompted to configure the modules, if they are not already configured.

Then, with the game started, the user will be asked to guess a sequence of cards. As the user places the cards on the slots, the game provides feedback in real-time, by having a representation of the cards on the screen, and changing the color of the cards to represent the actual state of the TUI.

When the user is ready to submit the sequence, they press the submit button, and the game will provide feedback on the user's guess, in the form of small arrows in different colors that represent the correctness of the user's guess: green for correct color and position, yellow for correct color but wrong position, and gray for incorrect color.

The last sequence and its feedback will be represented on screen so that the user can make another guess. Once the user guesses the correct sequence, the game will show a

panel congratulating the user, and the user can then choose to play again, change the game mode, or go back to the main menu.

3.8 Summary

In this chapter, we presented a platform designed to promote cognitive by combining the benefits of serious games with the engaging capabilities of TUIs, followed by a discussion of the requirements, methodology, proposed solution, and system architecture.

The platform is designed to be adaptable, versatile, and user-friendly, supporting a wide range of scenarios, exercises, and user demographics. The system's modular design allows for easy customization, extension, and integration, making it a valuable tool for cognitive training and rehabilitation in various settings.

Following this chapter, we will detail the implementation of the different components of the system, based on the requirements and guidelines discussed.

4

Implementation

This chapter details the design process and implementation details of how we developed the solution proposed in the previous chapter.

We present the TUI's hardware, such as sensors, actuators and microcontroller firmware, the module design, the software components of our system like the Middleware layer, the communication protocols, and the games developed to test the system.

As previously mentioned, the system is divided into three main components: the TUI, the Middleware, and the Unity games. These components are interconnected through different communication protocols, as shown in Figure 4.1, and each component's implementation is detailed in the following sections.

4.1 Tangible Sensor-Driven Interface For Cognitive Exercises

To achieve the final prototype of the TUI, we incrementally built and tested the hardware and software components.

The main hardware components are the modules that contain the sensors, actuators, and microcontrollers, that are responsible for reading the data from the cards and user input, sending it over to the Middleware.

The software components are the firmware running on the microcontrollers, the Middleware, that connects the hardware components to the Unity games, and the Unity games themselves.

4.1.1 Design

The design of the TUI was done gradually, starting with a single module, and then adding more modules as the system was tested and working. The modules were designed to be portable and easy to set up, so that they could be used in different environments and by different users.

The first iteration of the design was a wooden stand with four acrylic tubes, and colorful spheres that could be placed inside the tubes. To read the color of the spheres,

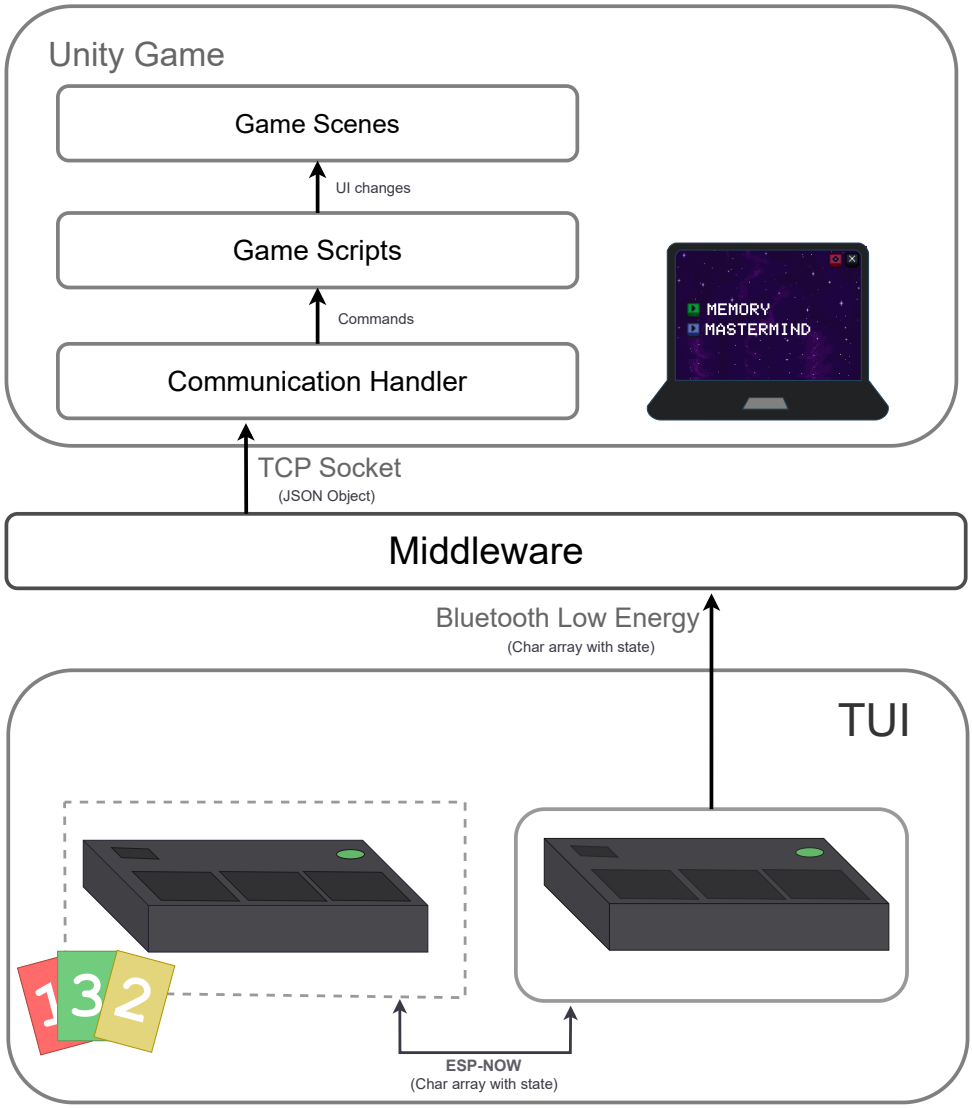

Figure 4.1: System Architecture

RGB sensors were used, and to detect the presence of the spheres we'd use proximity sensors, and a microcontroller would handle the data and communication.

This design impacted flexibility significantly, as it was primarily suited for younger children. Consequently, older children, teenagers, or adults may find this less appealing due to the limited range of games that could be played.

The final design were small boxes with slots for cards, and a button for submitting a solution, as shown in Figure 4.2. This solution was the one that most closely resembled a card game, and was the most flexible in terms of possible games that could be played with it, the possibility to change the cards used in the games, and use more than one module at the same time when needed.

While testing the system, a few changes were made to the design, such as the addition of an LCD to provide feedback to the user, especially during the configuration step, and

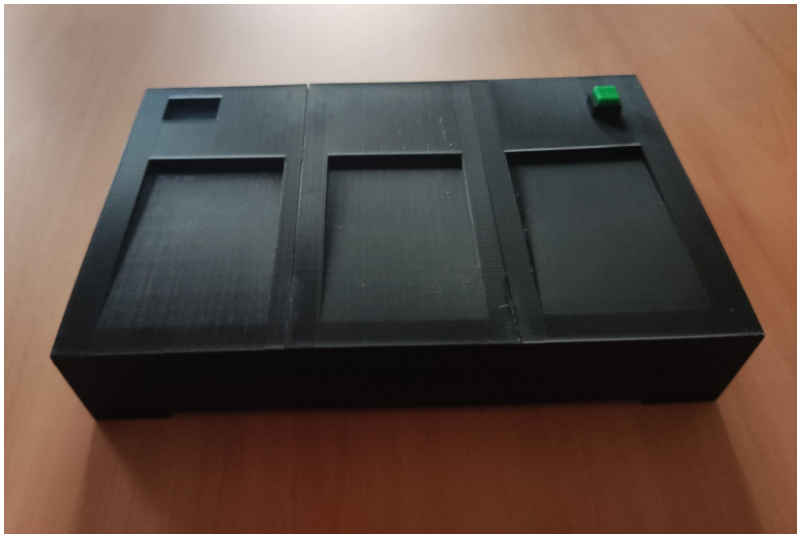

Figure 4.2: Card reader module

the addition of a button to submit a solution when games required it.

The elements were added at different stages of the design process, as the system was tested and new requirements were identified. These initial prototype iterations were important to understand the limitations of the hardware like power consumption, and the sensor accuracy, and the software, like the communication protocols used, and the data processing.

Once the system was tested and working, the final prototype was built, with three 3D printed cases for the modules, as well as adding a power supply to make the modules portable.

4.1.2 Tangible User Interface

In this section we detail the hardware components that are used in the TUI: the sensors used to read the data from the cards, the actuators used to provide feedback to the user, how the cards were modified to be read by the sensors, and the microcontrollers used to handle the data and communication.

4.1.2.1 Microcontroller

Each module, as seen in Figure 4.3, needs a microcontroller to handle the data from the sensors and actuators, and to communicate with the Middleware. The microcontroller used in the modules is the ESP32, as they are more accessible than other microcontrollers like the Arduino, and have a good balance between cost and features. The ESP32 is also compatible with the MFRC522 RFID readers, and has vast support from the community, which makes it easier to work with.

ESP32s also have imbedded support for BLE [4], which is used to communicate with the Middleware, and ESP-NOW [3], which is used to communicate between the modules.

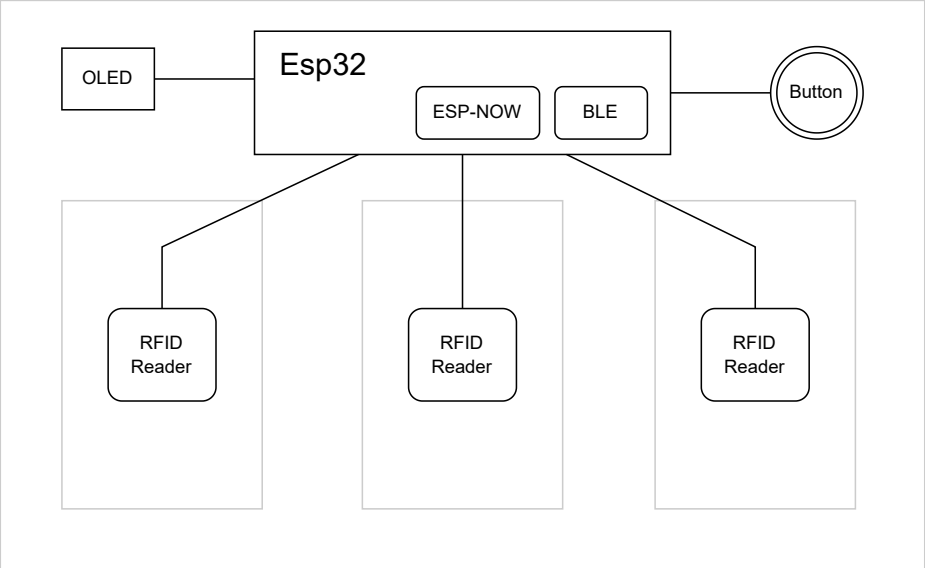

Figure 4.3: Module components

During the development of the system, the ESP32s were powered by a USB cable connected to a computer, and later a power bank was used to make the modules portable.

The microcontroller were flashed with the firmware developed using the Arduino IDE, and the ESP32 libraries for BLE and ESP-NOW and the MFMFRC522 library [5] for the RFID readers.

This microcontroller runs a firmware that is divided in two main components, *setup* and *loop*. *setup* is responsible for initializing the sensors, actuators, and communication protocols, and the loop does the data processing and connection to the Middleware.

4.1.2.2 Sensors and Actuators

To achieve the goal of the TUI, we needed sensors to read the data from the cards, and actuators to provide feedback to the user, a diagram of the components can be seen in Figure 4.3. Firstly we needed cards that could be read by the sensors, so we equipped a deck of cards with RFID tags. Simultaneously, we used RFID readers to read the data from the cards.

To help with feedback and configuration, the modules are equipped with a small OLED screen. The screens are capable of displaying text, numbers, and symbols. The one used is a monochrome 128x64px OLED with ESP32 integration via an Arduino library [2], and it communicates with the ESP32 using the I2C protocol [18], which is widely used in systems with multiple devices. It uses a two wire system where one is dedicated to transmit data (Serial Data Line), and one that synchronizes the transmission (Serial Clock Line).

The RFID readers used are the MFRC522. The MFRC522 is compatible with the ESP32

microcontroller and is capable of reading the unique ID of RFID tags at a distance of a few centimeters. The MFRC522 communicates with the ESP32 using the SPI protocol. This communication is preferred over I2C in cases like ours as it offers higher speeds and better transfer rates, since our RFID system requires frequent and fast data exchange.

The MFRC522 has a built-in antenna that is used to communicate with the RFID tags. The MFRC522 is capable of reading and writing data to the RFID tags, but in our system, we only use it to read the unique ID of the RFID tags.

Many of the possible games that could be played with the TUI required a button to submit a solution, so a simple push-button switch was added to the modules. The button is connected to the ESP32 using a digital input pin.

The modules were incrementally built and tested, starting with a single module, with one reader, then adding more readers, and testing for accuracy and reliability, especially when multiple readers were used in the same module. Having multiple readers in the same module, interference could become an issue, especially when the readers are close to each other, and a card hovers between them, but even in this case, the readers were able to read the unique ID of the RFID tags consistently, as it was defined that one unique ID would be read by only one reader, and the data read was accurate as the last read ID was the one that was sent to the Middleware.

It was also important to be able to detect when a card was removed from the slot, and since the RFID readers are not necessarily prepared for this, a mechanism of polling was implemented to detect when a card was removed from a slot.

The MFMFRC522 library has a function, *PICC_WakeupA*, that sends a Request For Answer to the card, and if a card is present, the card will send a response and the function will return true. If it returns false, we clear that space in the state array, and send this new state to the Middleware.

This introduced a new problem, as the RFID would sometimes read a card removal from slight movements, such as, when placing a card in a slot, or when a card was almost placed but then picked back up, or when a card was read by a reader in another module when being placed in a different slot.

To tackle this problem, we designed a solution that effectively identifies the removal of an RFID card from a reader by leveraging a robust polling mechanism combined with state management and signal stability checks. The key to its effectiveness lies in its use of consecutive detections to confirm the presence of a card, ensuring that temporary signal losses or environmental noise do not cause false-positive removals.

Reliable Detection through Consecutive Confirmations: A significant aspect of this solution is the use of a detection counter, 'detections', which is incremented each time a card is successfully detected by the reader. To ensure reliable identification of a card's presence, the system requires a minimum number of consecutive detections—in this case, 15 detections, with an interval of 20 milliseconds between them. This threshold accounts for potential environmental noise, brief interruptions in the RFID signal, or minor

movements of the card that might cause it to momentarily fall out of range. By setting this threshold, the system avoids falsely assuming that a card has been removed due to transient conditions, providing more reliable detection.

Identifying Card Removal: If the card fails to be detected within any polling cycle (i.e., PICC_WakeupA does not detect the card), the function immediately exits. This behavior indicates that the card is no longer present. When the card has been detected in the required number of consecutive polling cycles (15 in this case), the system then considers the card definitively present.

State Management for Card Presence: The system maintains an internal state variable to keep track of whether a card was previously detected on the reader. If the internal state shows that a card was present and the card is no longer detected (after failing to meet the consecutive detection threshold), the state is updated to reflect the removal. This state-clearing mechanism is a crucial step in the solution because it marks the exact moment of card removal, enabling the system to react accordingly.

Handling the Removal Event: Once the removal of a card is identified, the solution performs several actions: it updates the internal state to reflect the removal, clears the corresponding slot in the state array, and updates another internal variable, 'lastChange-WasRemoval[reader]', to indicate that the most recent state change was a removal, this flag is set to 'true' to mark that the most recent state change was indeed a removal, allowing for better logging and decision-making in future interactions.

Ensuring a Robust End-State: After handling a removal or detection event, the system uses the PICC_HaltA() command to halt the RFID communication session. This halting is crucial to avoid leaving the card reader in an active communication state, which could cause erroneous readings or system instability.

These process runs only when the lastChangeWasRemoval[reader] flag is set to false, meaning it avoids running the polling mechanism when a card has been removed last instead of placed. This mechanism was tested and proved to be reliable, and the system was able to detect when a card was removed from a slot, and when a card was placed in a slot, and the data sent to the Middleware was accurate and reliable.

4.1.3 Microcontroller Firmware

The microcontrollers, ESP32s, run firmware responsible for handling the data from the sensors and actuators, and for communicating with the Middleware. At *setup*, the microcontroller:

- Initializes SPI communication. SPI is a synchronous serial communication protocol that allows for high-speed data transfer between the ESP32 and the RFID readers.
- Initializes the RFID readers. The RFID readers are initialized using the MFMFRC522 library, which is a library that provides an API for interacting with the RFID readers.
- Initializes the ESP-NOW communication. ESP-NOW is a communication protocol that allows for one-to-many and many-to-many communication between ESP devices.
- Runs the leader election process, explained in detail in the next Section 4.1.6, to select a leader among the ESP32 modules to handle the communication with the Middleware.

The microcontroller is constantly checking the state of the RFID readers, and the button, and if a change is detected, the data is prepared to be sent over to the Middleware. The modules state is read and saved in variables, serialized into a char array, and sent over to the Middleware using BLE.

The firmware manages various state and control variables to facilitate the coordination of the distributed system of ESP32 modules that use RFID readers and ESP-NOW communication for leader election and data synchronization. Each module maintains a set of variables to keep track of the module's current state and control its behavior.

Some variables are used to store information about the leader election process, such as the leader's MAC address, while others help manage the button state, keeping track of last button state and a debounce counter to prevent long button presses from being registered as separate events. The firmware also keeps track of the last actions on each RFID reader, like the last card read, the last card removed, and the last button state.

4.1.4 Module To Module Communication

When designing a network of ESP modules, it is essential to establish an efficient communication protocol that can handle the data flow between the modules and the Middleware. The middleware layer is a program running on the same machine as the Unity game, and is implemented in Python.

We considered two primary approaches for communication between the ESP32 modules and the Middleware:

Electing a Leader for Communication Through BLE and ESP-NOW:

In this approach, seen above in Figure 4.4, one ESP32 module is dynamically elected as the leader, as described in Section 4.1.6. The non-leader ESP32 modules communicate their data to the leader using ESP-NOW - a lightweight, low-power, and low-latency wireless communication protocol optimized for short-range communication between ESP32 devices.

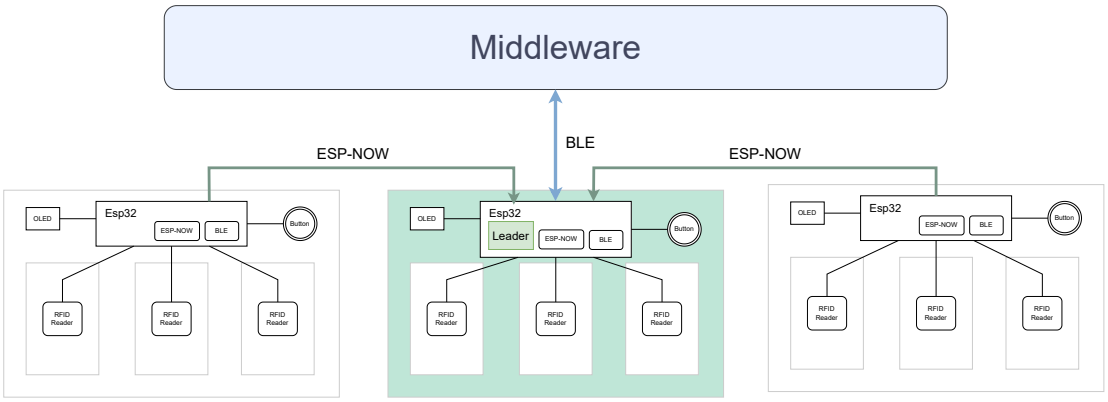

Figure 4.4: Elected Leader for Communication Through BLE and ESP-NOW

The leader module then receives the data from all other modules and sends it to the Middleware via BLE.

This approach offers several key advantages. By restricting BLE communication to a single leader, the network minimizes BLE traffic congestion, which is particularly important given BLE's limited bandwidth and number of simultaneous connections.

Additionally, ESP-NOW is more power-efficient than BLE, reducing overall power consumption for non-leader modules, which is crucial for battery-powered devices. This *setup* also enhances scalability; as the number of ESP32 modules increases, the BLE traffic (number of devices) remains constant, while only the ESP-NOW communication overhead increases. ESP-NOW also provides low-latency data transfer, and operates without the need for a Wi-Fi infrastructure, it supports multiple peers, and though it limits packet size to 250 bytes, it remains efficient for sensor data transmission.

Furthermore, aggregating and filtering data at the leader module allows for reduced transmission volume and bandwidth usage when communicating with the Middleware.

However, this approach does introduce some complexity, such as implementing a robust leader election mechanism and handling the potential for a single point of failure if the leader module goes offline. This can be mitigated by the leader election process, which can allow for the automatic selection of a new leader if the current leader fails. This is useful but in terms of the game logic, it can be problematic, as the game is expecting a set number of modules, and if one fails, the game would not work as expected, even if the leader election process runs again and a new leader is elected.

Nonetheless, the advantages of reduced BLE congestion, improved scalability, and lower power consumption make this the more suitable choice for our system.

Direct Communication of All ESP32s to the Middleware via BLE:

An alternative approach involves each ESP32 module independently communicating with the Python application using BLE. This eliminates the need for a leader election process and allows direct transmission of data, potentially reducing latency.

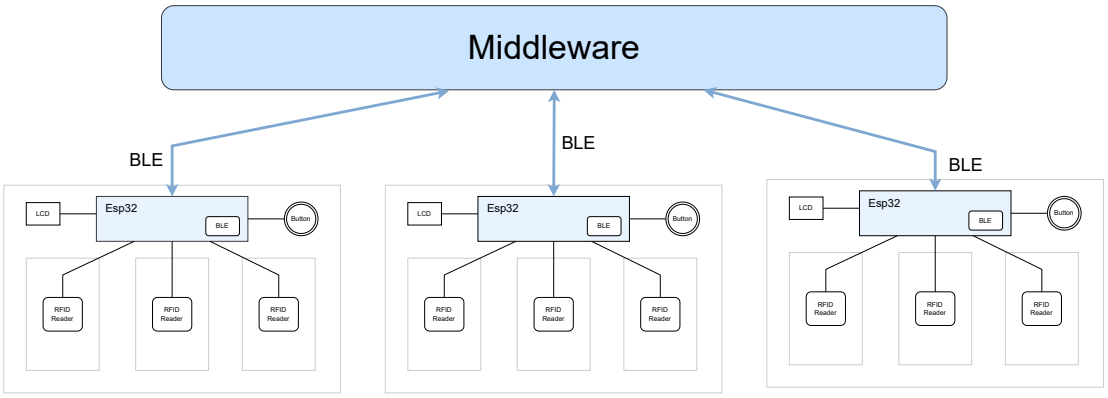

Figure 4.5: Communication of All ESP32s to the Middleware via BLE

Additionally, this approach avoids a single point of failure; if any ESP32 module fails, the remaining modules continue to communicate directly with the Python application.

Despite these benefits, this approach poses significant challenges in terms of scalability and power consumption. BLE is not designed to handle numerous simultaneous connections efficiently, and as the number of modules increases, the network can quickly become congested. This can result in increased latency, dropped connections, and unreliable data transmission.

Managing multiple BLE connections also introduces complexity in the Middleware application, making it less maintainable and more prone to errors.

Given the project's requirements for scalability, low power consumption, and minimizing BLE traffic congestion, the first approach, electing a leader module for centralized BLE communication with ESP-NOW communication between modules, was chosen. This method provides a scalable solution that can efficiently handle a larger number of modules while conserving power and maintaining reliable communication with the Python application.

4.1.5 Message Format

Each module needs to be identified uniquely, so that the Middleware can know which module is sending the data when receiving it. This is done by assigning a unique ID to each module, and sending this ID along with the data.

To design the communication between modules, a message structs were defined, with the following structures:

The *leader_Message* is used to send messages related to the leader election process:

- msgType: The type of message, used to distinguish between different types of messages (leader_Message, state_Message).
- type: The type of message, used to distinguish between different types of leader election messages:

```
// Leader Election Message struct
typedef struct leader_Message {
    uint8_t msgType;
    int type;
    int id;
} leader_Message;

// State Message struct
typedef struct state_Message {
    uint8_t msgType;
    int id;
    int nr_readers;
    char module[MODULE_SIZE];
} state_Message;
```

Listing 4.1: Leader Election and State Message Structs

  - type 0: *Leader election request message*, sent by the checking ESP32 to check if there is already a leader.
  - type 1: *Leader election response message*, sent by the leader ESP32 to inform that it is the leader.
  - type 2: *Leader election confirmation message*, sent by the checking ESP32 to inform that it has assumed leadership.

- id: The unique ID of the ESP32 module.

The *state Message* struct is used to send the state of the RFID readers and the button to the leader:

- msgType: The type of message, used to distinguish between different types of messages (leader_Message, state_Message).
- id: The unique ID of the ESP32 module.
- nr_readers: The number of RFID readers in the module.
- module: The state of the RFID readers and the button, serialized into a char array.
- MODULE_SIZE: The size of the char array.

On ESP-NOW *setup*, the ESP32s register a callback function that is called when a message is received, and one that is called when a message is sent and shows the status of message delivery. After having these functions set up, the ESP32s are ready to send and receive messages and run the leader election process. The communication using ESP-NOW uses the devices MAC address as the destination address, or a broadcast address.

4.1.6 Leader Election Protocol

As already mentioned, in our system, utilizing multiple ESP32 microcontrollers requires a leader election process to ensure flexibility and reliability, especially in dynamic environments where devices may join or leave the network.

By implementing a leader election mechanism, the system remains resilient regardless of the boot order or the number of ESP32 devices, as the election ensures that only one device will manage the communication with the Middleware. This dynamic assignment also eliminates the need for a specific module to host a previously designated leader, enhancing fault tolerance and system adaptability during *setup*, reboots, or failures.

The leader election process is initiated when an ESP32 boots, and it follows a set of rules to determine whether the ESP32 should assume leadership or wait for a leader to be elected:

No Leader Has Been Previously Elected:
If no leader has been elected prior to the current boot, the ESP32 follows these steps:

1. The ESP32 sends a broadcast message to all other ESP32 modules in the network, containing a *leader request message*.
2. It waits for a response from other ESP32 modules for 2 seconds, expecting a *leader response message*.
3. If no response is received within the specified time frame, the ESP32 assumes leadership. It then broadcasts a *leader confirmation message* to all other ESP32 modules in the network. The other modules, upon receiving this message, save the leader's MAC address as a peer.

Leader Has Been Previously Elected:
If a leader has already been elected, the booting ESP32 follows this procedure:

1. The booting ESP32 sends a broadcast message to all other ESP32 modules in the network, containing a *leader request message*.
2. It waits for a response from the currently elected leader ESP32 for 2 seconds, expecting a *leader response message*.
3. Upon receiving a *leader response message*, the booting ESP32 saves the leader's MAC address as a peer and does not attempt to assume leadership.

In Figure 4.6, the leader election process is illustrated when a new ESP32 boots, showing the steps taken when no leader has been previously elected, and when a leader has already been established. It is important to point out that the leader election process will many times be run in parallel, as the ESP32s can be booted at the same time, and the leader election process is run by all ESP32s that are booting. An example of parallel election processes is shown in Figure 4.7.

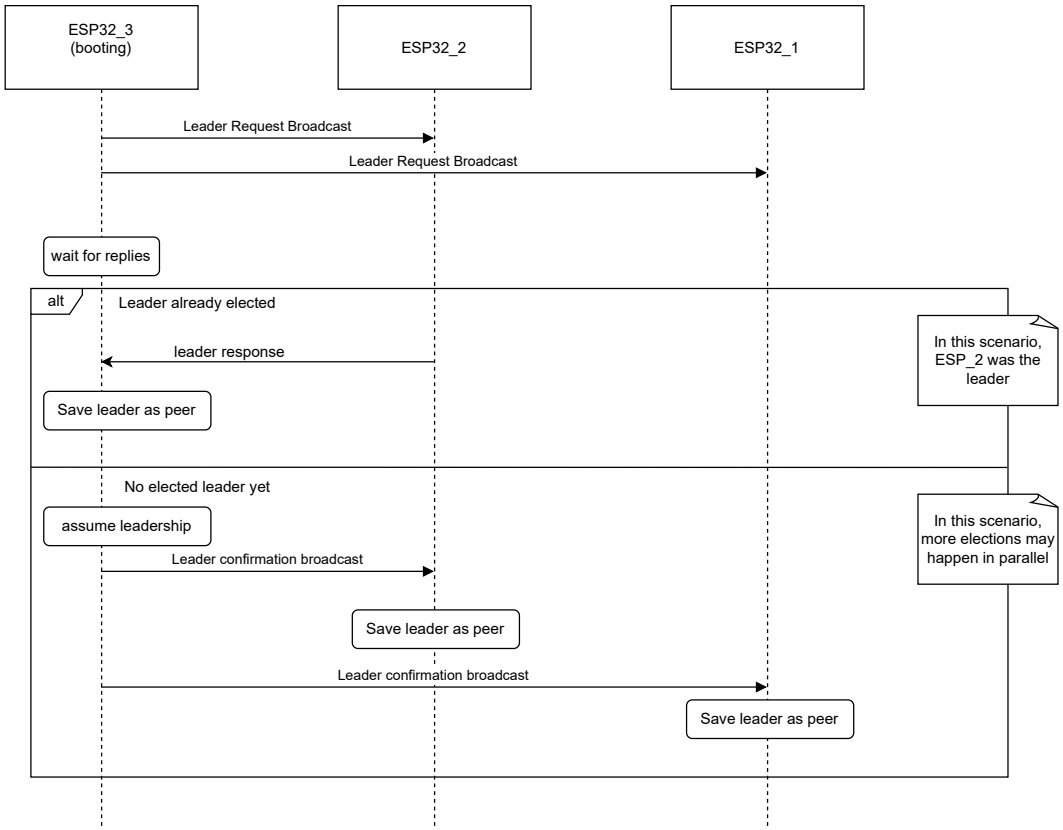

Figure 4.6: Leader election when a new ESP32 boots

The leader election process also addresses scenarios where the leader fails, or a message fails to be delivered:

- If an ESP32 module fails to deliver a message, it will attempt to resend it up to three times. If the message fails to be delivered after three attempts, the ESP32 assumes that the leader has failed.

- In this case, the module runs the leader election process again. If it does not receive a response from any other ESP32 modules, it assumes leadership.

This failure-handling mechanism is critical for maintaining system stability in cases where the leader ESP32 malfunctions, or runs out of battery. However, this behavior impacts the game logic and user experience, as the game expects a fixed number of modules after configuration. If one module fails, the game may not function as expected, even if a new leader is elected. If that occurs, the module that failed may need to be restarted, or the set of modules may need to be reconfigured.

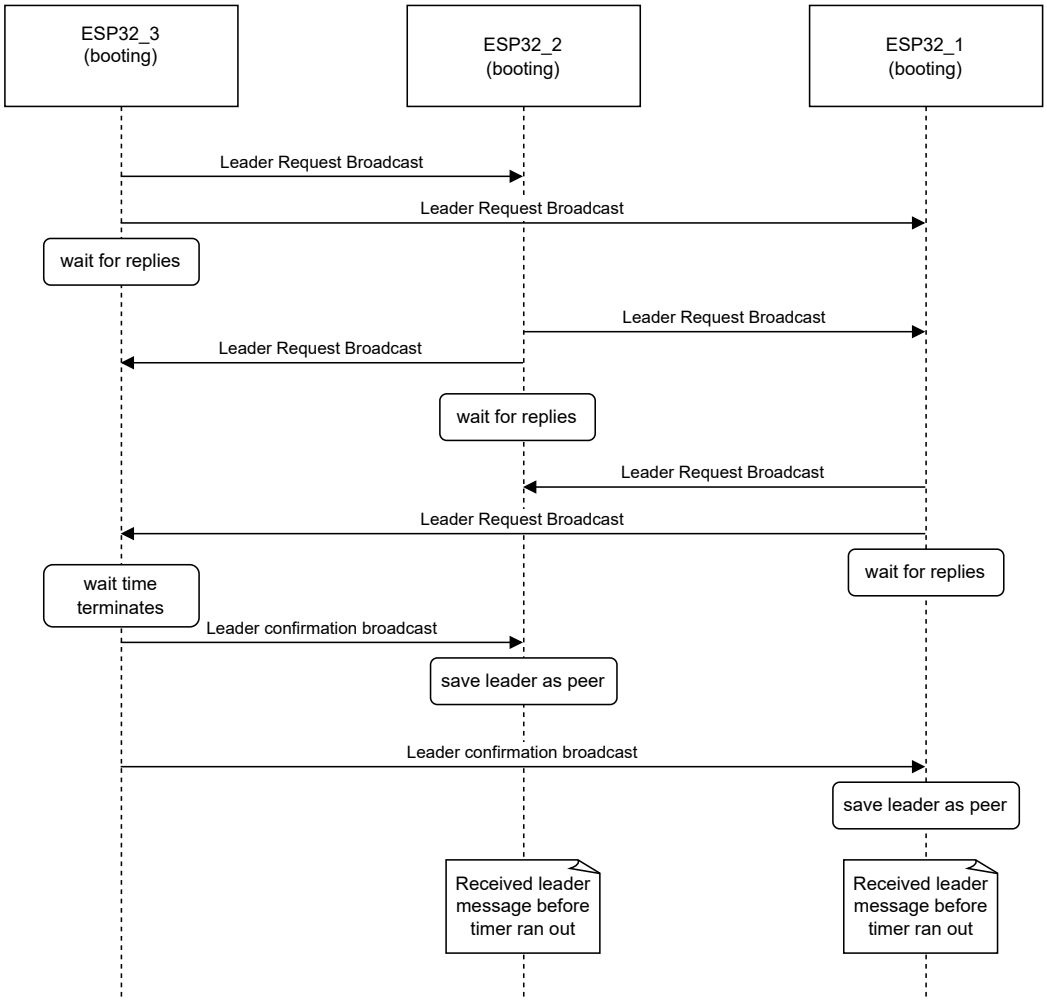

Figure 4.7: Leader election with three ESP32 booting at the same time.

4.2 Middleware

The Middleware is a software layer that connects the hardware components to the Unity games, running on the same machine, responsible for handling the data received from the ESP32s, and sending it to the Unity games to be interpreted and used in the game logic. It makes sure that the data is sent only once, limiting the number of messages sent to the Unity game, and making sure that the data is sent in the correct format.

4.2.1 Microcontroller-to-Middleware Communication

Once the leader has been elected, the ESP32s are ready to send the state of the RFID readers and the button to the Middleware. This communication is done using BLE, and the data is sent in a tuple format, where the first element is the ESP32 ID, the second is an array state of the readers, and the last element represents whether the button was pressed

or not.

The Middleware is a software layer, programmed using Python, that uses the Bleak library [9], a Python library that provides an API for interacting with BLE devices. The Generic Attribute Profile (GATT) in BLE is a client-server protocol, where the ESP32 acts as the server and the Middleware acts as the client.

The elected ESP32 advertises itself as a BLE device, and the Middleware scans for BLE devices and connects to the ESP32 using the ESP32's device name or MAC address.

GATT uses the Attribute Profile (ATT) to transfer data between the server and the client. The protocol is based on the concept of services and characteristics. A service is a collection of characteristics, and a characteristic is a piece of data that can be read, written, or notified. The server defines the services and characteristics, and the client reads, writes, or subscribes to the characteristics.

In our system, we use one service with two characteristic:

- One for sending data from the ESP32 to the Middleware, with reading permissions only.
- One for receiving data from the Middleware, with writing permissions only.

The ESP32 sends data to the Middleware by writing to the characteristic that is used for sending data, and in the Middleware, that characteristic is read, checked if the data is valid and different from the last data received, and if so, it is sent over to the Unity game.

The services and characteristics have Universally Unique Identifier (UUID) defined in the ESP32 firmware, and the Middleware uses this UUID to connect to the ESP32 and read and write data. A UUID is a 16-bit or 128-bit number (when used in custom services) that uniquely identifies a service or characteristic in a BLE device.

It keeps a connection open to the ESP32, this connection is established when the Middleware is started, and an instance of the BLE client device is created, if an error occurs, the Middleware will attempt to reconnect to the ESP32.

The Middleware uses the Bleak library to interact with the BLE devices, and the library provides an API for scanning for devices, connecting to devices, reading and writing data to characteristics, and subscribing to notifications.

The Middleware is responsible for handling the data received from the ESP32s, and processing it to be sent to the Unity game. This layer translates the data received from the ESP32s into a format that is easier to parse and use in the game logic.

The data received from the ESP32s is sent in char arrays structured as follows:

```
char data[_SIZE] = [id,nr_readers,state_0,...,state_n, button];
// n = nr_readers
```

Listing 4.2: Data sent by TUI

It then parses this data into a JSON object with the following structure:

```
{
    "id": 1,
    "nr_sensors": 3,
    "sensor_states": ["50 40 350 123", "", "30 20 100 321"],
    "button_state": 0
}
```

Listing 4.3: JSON message sent to game

This JSON object is then sent to the Unity game to be translated into a command that is used in the game logic.

4.2.2 Middleware-to-Game Communication

As these two components are running on the same machine, the communication between them makes use of the loopback address, and the data is sent using a TCP connection, when a connection is established, the Middleware sends the data received from the ESP32 to the Unity game.

This connection is kept open while the Middleware is running, and if an error occurs, the Middleware will attempt to reconnect to the Unity game. If unable to connect, the Middleware will keep trying to connect to the Unity game, and after 10 attempts, it will stop running and inform the user that it was unable to connect to the Unity game.

4.3 Game Side

Unity is a widely used game engine that enables developers to create 2D and 3D games, as well as interactive experiences. The core concepts of Unity revolve around *GameObjects* and scripting.

GameObjects are the fundamental elements in Unity, representing all entities in a scene such as characters, props, cameras, and lights. Each Game Object can have various components attached to it that define its behavior and appearance.

In addition to this, developers can attach custom scripts to *GameObjects* to add unique functionalities. These use C# as the scripting language to define the behaviors and interactions of *GameObjects*. Scripts are written as classes that inherit from a base class that offers a set of methods for handling different aspects of Game Object behavior. Normally they have the *Start()* method that is called once that object is initialized, and the *Update()* method that is called every frame to update the object's state.

Games in Unity are built using scenes, which are individual levels or environments that contain all the elements of the game. The first scene is the main menu, where the user can start the game, configure the modules, and exit the game. This is the where the script that handles the communication with the Middleware is initialized.

```
public struct Command
{
    public int id;
    public string[] positionStates;
    public int buttonState;
}
```

Listing 4.4: TUI command structure

```
void startCommMiddleware()
{
    Debug.Log("Starting server");
    ThreadStart ts = new ThreadStart(GetData);
    thread = new Thread(ts);
    thread.\textit{Start()};
}

void GetData()
{
    while (true) {
        try {
            server = new TcpListener(IPAddress.Any, connectionPort);
            server.\textit{Start()};
            Debug.Log("Server started, waiting for client...");
            client = server.AcceptTcpClient();
            running = true;
            Debug.Log("Client connected.");
            while (running) { Connection(); }
        }
        catch (SocketException e) { // Handle socket exception }
        finally { // Handle server stop }

        Thread.Sleep(1000); // Wait 1 second before trying to reconnect
    }
}
```

Listing 4.5: Game-Middleware connections and communication

4.3.1 Communication Handler

The communication handler is a *C#* script that is attached to a *GameObject* that is initialized when the game is started. This script is responsible for creating a TCP client [45] that connects to the Middleware, and for receiving the data sent by the Middleware.

This class then parses the data received, a JSON object, and translates it into an Object of the class *Command*, that is then added to a message queue. The message queue stores the commands received from the Middleware, and the game scripts reads from this queue to update the game state.

On the *Start()* method, the script creates a TCP client that connects to the Middleware:

The object follows the Singleton pattern, which is pattern used to ensure that only one instance of the object is created throughout the game, and that the object is accessible from any other script in the game. This is important as the object is responsible for receiving the data sent by the Middleware, and most other scripts in the game need to access this data

```
    {
        "id": 1,
        "nr_sensors": 3,
        "sensor_states": ["50 40 350 123", "", "30 20 100 321"],
        "button_state": 0
    }
```

Listing 4.6: Message received from Middleware

```
if (!string.IsNullOrEmpty(dataReceived)) {
    try {
        // Parse the JSON object
        var jsonData = JsonSerializer.Deserialize<JsonElement>(dataReceived);
        // Read the ESP32 ID
        int ESP32Id = jsonData.GetProperty("id").GetInt32();
        // Read the number of sensors
        int numReaders = jsonData.GetProperty("nr_sensors").GetInt32();
        // Read the sensor states
        JsonElement sensorStatesArray = jsonData.GetProperty("sensor_states");
        string[] readerStates = new string[numReaders];
        for (int i = 0; i < numReaders; i++) {
            readerStates[i] = sensorStatesArray[i].GetString();
        }
        // Read the button state
        int buttonState = jsonData.GetProperty("button_state").GetInt32();
        // Create a new command with the parsed data
        // Enqueue the command for processing
    } catch (JsonException ex) { // Handle JSON parsing errors
    } catch (Exception ex) { // Handle general errors }
} else { // Handle empty data }
```

Listing 4.7: Command parsing

to update the game state. Finally, when the game is closed, the script closes the connection to the Middleware.

Threads are used to handle the communication with the Middleware, as the communication is done in a separate thread from the main thread that runs the game logic, to avoid blocking the game loop.

4.3.2 Game Sensor Data handling

It is important that the data is sent in a format that is easy to parse and use in the game logic. The sensor data is translated into a semantic representation that the game can understand, and this data needs to be flexible enough to be used in different games, and with different configurations of the modules. Data sent by the modules needs to be sent formatted in a way that is flexible in both the number of sensors, the number of modules, and the type of data being read.

The data received is sent in the form of a JSON object, with the following structure:

Upon receiving the data, the communication handler parses the JSON object and creates a Command object with the data received.

The data is then parsed and translated into a Command object, that is then added to a message queue. The message queue is stores the commands received from the Middleware, and the game logic reads from this queue to update the game state.

4.3.3 Card deck creation

The use of different card decks is necessary for the creation of different games, as the cards are the main interaction point between the user and the system.

Knowing an easy way to create and manage the card decks is important, we developed a tool to simplify the process of creating and managing the card decks, and to allow the game developer to easily create new card decks or crating variations of existing card decks.

This translation mechanism can be used to create any type of translation from different types of sensor outputs to the game logic data and not just the RFID readers.

The card deck creation tool is a graphical interface that uses Unity's UI elements to allow the user to create and manage the card decks, shown in Figure 4.8. Just like any other component in the games created, an instance on the Command Handler class to receive the data from the Middleware.

This tool helps create a JSON file that contains the mapping between sensor readings and the data that the game logic can understand. The configuration scene requests a name for the card deck, and then prompts the user to indicate the color and symbol of the subsequent cards. The user is guided through the process, and asked to tap the card on the one of the module's card slots. It is asked of the user to tap all the cards of that type, and then the process repeats for the next card type, until all the cards are created.

The dedicated script manages the creation and modification of card decks stored in JSON format. It defines two serializable classes: *dataTranslation*, which holds the properties of a sensor reading (color and number), and *dataTranslationEntry*, which pairs a card with a key for identification. These entries are stored in a list within the *dataTranslationList* class.

We attempt to create new JSON file representing an empty deck, named according to the user's input. It first checks if a file with that name already exists to prevent overwriting.

The another method allows adding cards to an existing deck. It checks if the deck JSON file exists and if the key for the new card is unique. If both conditions are met, it adds the card and updates the JSON file.

Figure 4.8: Deck configuration panel

4.3.4 Data Translation

The data translation system is a crucial component in the game logic, responsible for converting the raw sensor data into meaningful game states. This translation process involves mapping the sensor data to specific game elements, and it is essential for ensuring that the game logic can interpret and respond to the sensor data correctly.

In this context, data translation refers to converting sensor data into meaningful semantic representations by using external JSON files that map sensor states to game attributes. These JSON files contain key-value pairs that define the mapping between sensor states and game attributes, such as colors, numbers, or text. This technique allows dynamic updates of game assets and separates the logic from the visual components, improving flexibility and scalability.

Translation System: The core of this system is the mapping of unique state keys to specific attributes (such as color and number) stored in a JSON file. This structure facilitates dynamic handling of game elements without hardcoding their properties. Each entry pairs a unique key with an object containing attributes (e.g., color, number, text). A collection of these entries is maintained and accessed at runtime, enabling the system to handle multiple state changes.

Loading the Translator: The method for loading translations involves reading a JSON file that holds key-value pairs, where each key represents a state and the value holds the corresponding attributes. The contents are then parsed into a dictionary, allowing for fast lookups when the game state changes.

Processing Commands and Translating States: During runtime, when the game receives input, it uses the previously loaded dictionary to translate these states into game data, which is then applied to the corresponding game elements. The system retrieves the corresponding attributes (e.g., color, text) from the dictionary, and uses it to update UI elements such as colors, text, or icons, to reflect the current state.

Example Flow:

- A state or command is received with a key (e.g., '53 432 12 82', representing the ID of the RFID tag).
- The system looks up the key in the dictionary and retrieves the associated attributes (e.g., blue color, number 7).
- The visual representation in the game is updated to match the retrieved attributes.

This generalized approach to data translation allows for a clear separation between data and visuals, making it easy to update or change elements by modifying the JSON

Figure 4.9: Interface Configuration Panel

file rather than the game code. This method also improves maintainability and supports rapid iteration and scaling of game content.

4.3.5 Interface Configuration

In modular and dynamically configurable systems, such as TUIs in our example, there is often a requirement to manage a variety of interconnected components, each of which may have different states or functionalities. In our system, modules represent the hardware devices specified before that need to be dynamically activated, ordered, and configured based on real-time input or user interactions.

With our system, each module is a separate entity representing a distinct virtual element that needs to be managed and displayed in the Unity environment. When multiple modules are uses, they need to be activated, configured, read from or removed in a specific order. This ordering needs to be maintained and easily accessible for further operations. The way in which modules are activated should be independent of the number of modules present in the system, and allow for easy expansion or reduction of the number of modules.

To address these challenges, a graphical configuration interface was developed to provide a user-friendly and intuitive way to manage the activation, configuration, and removal of modules in the Unity environment, as presented in Figure 4.9. The interface allows users to interact with the system in real-time, providing immediate feedback and enabling dynamic configuration changes.

The user is asked to configure the modules before starting the game, and the interface guides the user through the process. They are asked to place a card in any on the module's card slots, going in order from 1 to N, where N is the number of modules, from the left most module to the right most module.

A digital representation of the modules is shown on the screen, to help the user understand the configuration process. This configuration panel is prepared to handle up to 3 modules, but can be easily expanded to handle more modules.

This game script works closely with other components like the communication handler script and a static class that keeps the defined configuration after the configuration process is finished. The static class is used to store the configuration of the modules, and is accessible from any other script in the game. It is used to keep track of the order in which the modules were configured, and the state of each module, how many modules are active, and the configuration process, whether it is concluded or not.

The script attached to a *GameObject* that is initialized when the interface configuration scene is loaded. It is responsible for:

Receiving and Processing Commands: It listens for incoming commands from the external communication handler and processes them to determine which module should be activated.

Dynamic Activation of Modules: It activates the appropriate module *GameObjects* based on the commands received and updates the configuration state file accordingly.

This configuration data is then used by the game logic to determine which modules are active, and in which order they were configured. As mentioned before, Unity games are built using scenes, and normally, state from one scene is not carried over to the next, but by using a static class, the configuration data is kept throughout the game. The configuration script relies on several important components to manage the configuration process effectively:

Modules array: An array of *GameObjects* representing the graphical representation of physical modules in the scene.

Order management: The order array is used to keep track of the order in which modules are activated. This array 3nsures that modules are activated in a specific sequence dictated by the user commands.

Communication handler instance: This class integrates an instance the communication handler script to receive commands from the Middleware and process them accordingly.

While the script is running, it listens for incoming commands from the communication handler, once it detects that the command queue is not empty, it processes the command, reads the ID of the module that sent the command, and activates the corresponding module in the scene. The script then updates the order array with the ID of the module

that was activated, and the state of the module is set to active. This process is repeated until all modules are activated, and the configuration process is concluded.

Once the configuration process is finished, the script saves the configuration data to the static class responsible for keeping the configuration data of the modules.

4.4 Use Case Games

To validate the framework and the TUI, we developed two games designed to test the system's capabilities and limitations. These games were selected to demonstrate that the interaction between the hardware and software components is both reliable and accurate, while also being intuitive and user-friendly.

The games are adaptations of classics: Memory and Mastermind. These choices were made due to their simplicity, broad appeal, and ease of understanding. The Memory game, commonly associated with children, and Mastermind, a classic strategy game that can be appealing to adults, were chosen to target a wide range of users.

It was essential to verify that the same interface could function effectively across different games, regardless of the unique logic and rules each game requires. This reinforces the system's flexibility and adaptability, proving how the same hardware can be used to support multiple games.

Furthermore, the number of modules used in each game can be adjusted, influencing the game's difficulty and complexity, and demonstrating how this variability enhances the user experience. Figure 4.10 shows the difficulty menu panel for both games.

The decision to implement different decks of cards for each game was aimed at highlighting the system's versatility in terms of user interaction. Each game was designed to accept different card types. This flexibility in design ensures that the system can support diverse gameplay experiences.

A major focus was placed on ensuring that the game development process would be streamlined for developers, making the system easy to use and understand. Our objective was to simplify the development process and make it easy to integrate the hardware components with new game logic. To achieve this, we simplified and abstracted the communication between the Middleware and the game, configuration of the modules,

Figure 4.10: Game modes

and the translation of the data received from the Middleware. This allows developers to focus on game design rather than complex integration.

Communication between the Middleware and the game is encapsulated in a single script, designed to be easily understood and reusable across any scene that utilizes the TUI.

The previously mentioned configuration interface plays a crucial role in the development process as well. It allows for module configuration across different games, reducing the burden on the developer by automating steps that would otherwise require manual configuration.

When more than one module is used in a game, the order of the modules becomes important, as the game logic must handle incoming data accordingly. To achieve this, the system accesses configuration data and the order in which the modules were initialized, using a static class that stores the configuration settings.

To correctly interpret the data received from the hardware, a translation mechanism was defined to map the raw input into meaningful data for the game logic. This translation system is based on a JSON file that stores the mapping between the raw data and the game elements.

4.4.1 Memory Game

We designed a Memory game using the TUI, as shown in Figure 4.11. The game is played on a physical board where the player is challenged to recreate a sequence of symbols that are temporarily shown on-screen. The game leverages physical RFID-enabled cards, each corresponding to a different visual symbol and the reader system. It begins by showing the player a randomly generated sequence of three symbols.

Figure 4.11: Memory game

Random Sequence Generation: This sequence is generated by picking N numbers between 1 and 3, where N is the number of modules configured in the system, and each number corresponds to a symbol, one of three characters that are used in the game. These numbers have enumerator values that are used to map the numbers to the symbols, and are used to create the sequence. This sequence is displayed to the player for a few seconds before it is hidden.

Player Input via RFID Cards: Players interact with the game by placing RFID-enabled cards on RFID readers integrated into the board. Each card has an RFID tag that represents a specific symbol. When a card is placed on a reader, the RFID reader system detects it and communicates the information to Unity using the Command Handler class.

Game Logic and Processing: The core game logic is handled by the *MemoryEasy* class, which listens for commands from the command handler script. When a command is received:

- The game checks which cards/RFID tag are on the readers.
- Each ID read is then matched to the corresponding symbol in the dictionary that was created from the JSON file.
- The corresponding *GameObjects* are updated to display the correct symbols on the screen.

Visual Updates and Feedback: The game updates the visual representation of the cards on the modules by enabling or disabling different child *GameObjects* that represent symbols. This is done directly in Unity's main thread to ensure smooth visual updates without lag or delay.

Sequence Verification: When the player presses the confirm button, the game checks if the sequence of symbols created by the player matches the randomly generated sequence. The following outcomes are possible:

- If the sequences match, the player wins the round a 'winner panel' game object is enabled, congratulating the player, and giving the player the option to play again, change the difficulty, or exit the game.
- If the sequences do not match, the player loses the round, and is given the options to see the sequence again, or to keep going where they left off.

Creating Level Variations: The way the game is designed allows for easy level variations, by changing the number of modules, the number of symbols, and the game logic. The game can be easily expanded to include more modules, therefore requiring the player

to remember a longer sequence of symbols which is how we chose to change the difficulty level in this case.

In this case the difficulty could also be increased through other means such as reducing the time the sequence is shown to the player, or by having the symbols appear one at a time, and the player having to remember the sequence as it is shown, in the same order that it is shown. This way, the game logic could be changed in terms of rules only, and the hardware *setup* would not need to be changed.

We opted to keep the game simple, and to focus on the interaction between the hardware and software, to show that the system is reliable and accurate, and that the game logic can be easily implemented and changed. In this harder mode, the user must instead remember a sequence of six symbols, and the game is won when the player correctly recreates the sequence.

The changes needed to create a different game mode based on the easy mode were:

- Random Sequence Validation: The game generates a random sequence of six symbols. With some additional restriction if some symbol is repeated too many times, the sequence is regenerated.

- Expansion of Game Modules and Sequence: The medium difficulty introduces the concept of handling multiple modules. The game is split into two modules, each responsible for a subset of the cards. The first module handles the first three cards (positions 0 to 2), and the second module (secondModuleId) handles the last three cards (positions 3 to 5). The game logic is adapted to process commands for each module independently. Depending on which module sends a command, the game must determine which part of the card array to update. This is managed by dynamically calculating the start and end indices of the card subset that each module is responsible for.

Everything else remains the same: communication, configuration, and game logic.

4.4.2 Mastermind Game

We designed a Mastermind game, seen in Figure 4.12. The game is played on a digital interface where the player attempts to recreate a sequence of symbols. The game uses visual cards to represent different symbols, and players interact with these cards via a command input system to make their guesses.

Random Sequence Generation: The game randomly selects a sequence of colors and numbers from a predefined list defined in a JSON file. The sequence length and complexity vary depending on the difficulty level. For instance, in the 'easy' mode, the sequence comprises three cards, while in 'medium' mode, it includes six cards. Each card represents a combination of a color and a number, which are mapped using enumerations from JSON configuration files. The sequence is generated at the start of the scene, after the JSON file

with the sequences is read, and the game is ready to start. Then once submitted, the game checks if the player's guess matches the generated sequence, and provides feedback based on the accuracy of the guess: a green indicator for a correct guess, a yellow indicator for when a card exists in the sequence but not in that position, and a red indicator for an incorrect guess.

Player Input via Command System: Players interact with the game by selecting cards that represent their guesses for the sequence. Each selection triggers a command through the Command Handler class, which manages the game's input system. Commands contain the card positions and states, which are processed by the game logic. Which are then processed by the game logic to determine if the player's guess matches the generated sequence.

Game Logic and Processing: When a command is received, the positions and states are read:

- The game checks which cards are selected and their corresponding positions.
- The selected card IDs are matched to the correct symbols using a dictionary created from the JSON translation files.
- The corresponding *GameObjects* in Unity are updated to reflect the current selection, displaying the appropriate color and number.

Visual Updates and Feedback: The game provides immediate visual feedback based on the player's input, meaning the cards placed on the tangible interface, and this is done by updating the visual representation of the cards to display the correct colors and

Figure 4.12: Mastermind game

numbers corresponding to the selected positions in real time.

Sequence Verification: When the player submits their guess, the game checks the sequence of colors and numbers against the target sequence. The outcomes are as follows:

- If the player's sequence matches the generated sequence, a 'winner panel' is displayed, indicating the player's victory and offering options to play again, change the difficulty, or exit the game.
- If the sequences do not match, the player is notified of their incorrect guesses and feedback is provided to help them improve their next guess. Indicators are shown to denote whether a player's guess is correct (green), partially correct (yellow), or incorrect (red).

Creating Level Variations: The game is designed to allow for easy adjustments to its difficulty levels by varying the number of cards and modules. For example, the easy mode involves a single module with three cards, while the medium mode introduces an additional module, expanding the sequence to six cards. This modular approach allows for future expansions or variations in gameplay.

- Random Sequence Validation: For medium difficulty, the game generates a random sequence of six symbols. If a symbol repeats too frequently, the sequence is regenerated to ensure diversity. This validation helps maintain the challenge level for players without introducing excessive repetition.

- Expansion of Game Modules and Sequence Handling: The medium difficulty level introduces the concept of handling multiple card modules. The game is divided into two modules, each responsible for managing a subset of the three cards. The first module handles cards in positions 1, 2 and 3, while the second module handles cards in positions 3, 4 and 5.

The game logic stays the same but dynamically calculates the offset to update the correct part of the card array, depending on which module sends a command.

4.5 Game Development Guidelines

The games created follow a set of guidelines that were established to ensure that the system is user-friendly for future developers.

Games created this using this framework should follow a design similar to the one shown in Figure 4.13. *CommHandler* is a script that handles the communication between

the Middleware and the game, and should be attached to a *GameObject* in the first scene of the game.

The Configuration Scene can be used completely, and runs a script that is responsible for handling the configuration of the modules. This script writes to a static class that keeps the configuration data throughout the game, *SharedData*, that should be placed in the *assets* folder of the application. Data stored in this class is accessible from any other script in the game, and is used to keep track of the order in which the modules were configured, and the number of modules.

Game logic in the game scenes is individual and should be created according to the game's rules and logic. It should listen for commands from the *CommHandler* script, and update the game state accordingly. To consult the configuration data, the game logic should access the *SharedData* class.

To help understand problems that may arise during development of new games, error reporting in the form of logs is essential. The system logs errors, warnings, and information messages, to help the developer understand what is happening in the game, specifically related to the communication between the Middleware and the game.

The use of cards as the main interaction form may be extended to other types of actions and interactions besides the ones proposed in this work, and extended to different game types. To help create a new deck for a game being developed, the deck creation tool can be used to create a new JSON file that maps the sensor outputs to the game semantics.

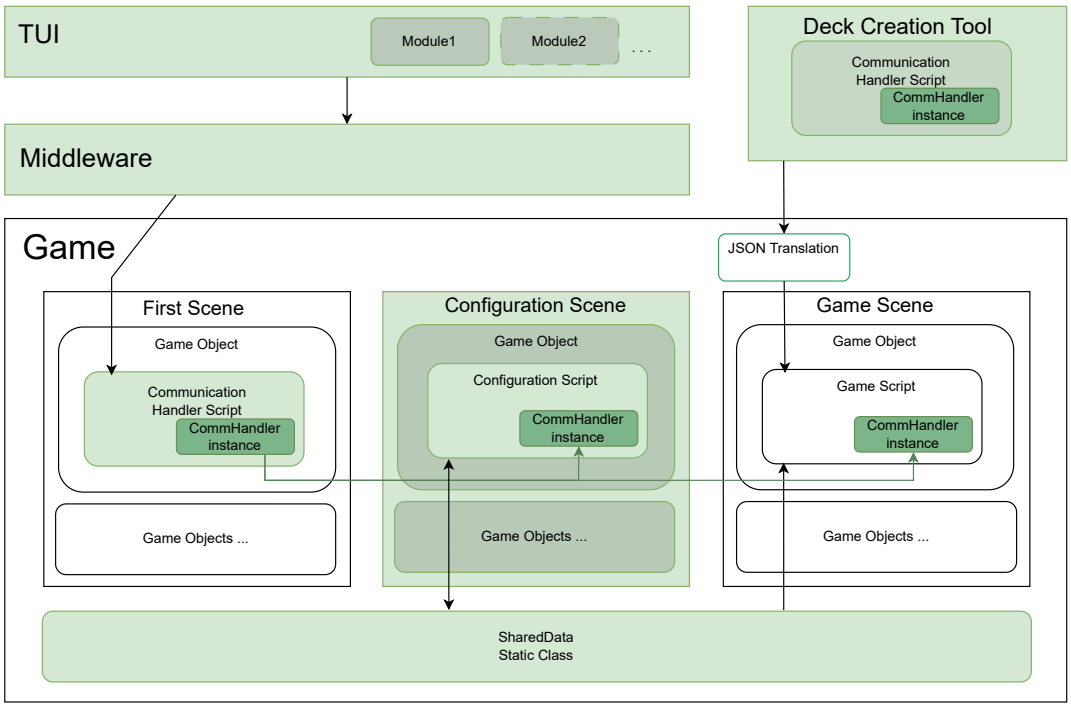

Figure 4.13: Game design guidelines

```
char data[MAX_SIZE] = [id,nr_readers,state_0,...,state_n, button];
```

Listing 4.8: Data format of TUI messages

The concept of a 'deck' is not limited to traditional cards; it can be extended to encompass various types of sensor inputs. The actual physical objects would also need to be adapted, and the microcontroller code would need to be updated to match the new sensors. When this is done, the developer should do the same to add process of adding a new card deck, using the deck creation tool, JSON file that maps the new sensor outputs to relevant game semantics. Instead of passing cards by the RFID reader, other sensor interaction needs to be replicated. The Middleware will be able to handle the new data, with no changes to its behavior, as long as the data sent to the Middleware is sent in the same format:

4.6 Summary

We presented the system architecture, which consists of three main components: the ESP32 firmware, the Middleware, and the Unity game. The ESP32 firmware is responsible for handling the hardware components, such as RFID readers and buttons, and for sending the data to the Middleware. The Middleware is a software layer that connects the hardware components to the Unity games, responsible for handling the data received from the ESP32s, and sending it to the Unity games to be interpreted and used in the game logic. The Unity use case games created are responsible for handling the game logic and user interface, and for displaying the game to the user.

The way the game is designed allows for easy level variations, by changing the number of modules, the number of symbols, and the difficulty of the game. Cards are used as the main interaction form providing easy ways to adapt the system to different types of games and interactions, and the deck creation tool can be used to create new card decks for new games. This allows for easy expansion of the game, and for the creation of new games using the same hardware components. Still, changing the sensory components of the game is also possible, in case developers want to create a new game using new interaction models.

In next chapter, we test the usability of the system and games created, evaluating the system's performance and reliability. Additionally, we validate the platform's flexibility and scalability, and how it can be used to create new games and experiences.

5

Validation and evaluation

After the system was implemented and tested, we followed through with the evaluation, testing its performance and validating its effectiveness. This chapter outlines the evaluation methodology, the experimental setup, and the results obtained during the evaluation process.

The first part of the evaluation focuses on assessing the TUI, the games developed, and their interaction. Evaluating the developed work is crucial to validate the usability of the interface, the system's performance, and the users' interest in the proposed solution, particularly the use of card-based games for interacting with cognitive training exercises.

To assess the usability of the TUI, a user study was conducted with a group of participants who interacted with both the interface and the games developed. Additionally, the system's performance was evaluated based on the perceived consistency and reliability of its operations, as well as the performance of the games from the perspective of the testers.

The second part of the evaluation is dedicated to validating the framework that was developed. A key objective of this process is to measure the framework's usefulness in facilitating the creation of new games. To achieve this, we assess the framework's flexibility and simplicity in game development, as well as its configurability and capacity for integrating new sensors into the system.

Throughout the chapter, we provide clarification on the metrics used, describe the experimental setup and methodology, and present the results obtained. Alongside the results we provide that explores their significance and implementations.

5.1 Evaluation Criteria

The evaluation of the system will focus on both end-users and therapists/educators. It will be based on criteria and metrics such as:

- Usability: The ease of use of the TUI, the games, and the framework.
- Communication and Reliability: The consistency and reliability of the system, as well as the performance of the games.

- Adaptability to different game modes: The ability of the system to adapt to different game modes and different difficulty levels.
- Concept variability: Validation of the concept of using card-based games to create new cognitive training games.

The evaluation of the system at the developer level, when creating new games using the developed framework will be based on criteria and metrics such as:

- Learning curve: How hard is it to learn how to use the framework and create new games.
- Customizability: How easy is it to extend the framework to meet the requirements of different game contexts.
- Multiplayer support: How easy is it to create multiplayer games using the framework.
- Error Reporting: How easy is it to debug and fix errors in the games created.

5.2 Experimental Setup

We conducted the testing in a room where the participants could sit and play the games. The TUI was placed on the table in front of the users, and the games were displayed on a computer screen.

Participants were asked to follow a set of instructions regarding the games they were going to play. As they played the games, we collected data on the time spent playing, the errors made, and feedback given by the participants.

The games tested were presented in Chapter 4, and the participants were asked to play them at different difficulty levels. The testing scenario was designed incrementally, starting with the easiest games and increasing the difficulty as participants progressed until they played all games at all difficulty levels.

Sessions were designed to last around fifteen minutes each. Participants were given a script to follow, with instructions on how to play the games, and a sequence of games to play:

- Answer the pre-questionnaire.
- Task 1: Play the Memory game in easy mode.
- Task 2: Play the Memory game in hard mode.
- Task 3: Play the Mastermind game in easy mode.
- Task 4: Play the Mastermind game in hard mode.

This test script was designed to maximize the number of configurations the participants had to perform before playing the games, in order to test the ease of configuration of the TUI. Between each task, a new configuration was needed, and the participants were asked to configure the games themselves.

5.3 Participants And Background Data

The evaluation of the system was conducted in a controlled environment, with a group of 15 participants, aged between 12 and 48 years old, Table 5.2, and almost equally distributed in gender, Table 5.1. Education levels can be seen in Table 5.3. From the background data collected, 80%, were students. Participants were asked to first fill a pre-questionnaire that collected background data, their experience with cognitive training games, and their expectations regarding the games they were going to play.

Table 5.1: Gender distribution

Gender	Participants (N = 15)
Male	53.3%
Female	46.7%

Table 5.2: Age distribution

Age	Participants (N = 15)
12-18	53.3%
18-35	26.6%
35-48	20.1%

Table 5.3: Level of education

Education	Participants (N = 15)
Elementary	26.7%
High School	40%
Bachelor or Higher	33.3%

Participants were asked to rate their familiarity with technology on a scale from 1 (not comfortable) to 5 (very comfortable). The results showed that 86.6% of participants considered themselves to have a good level of experience with technology, while 13.4% rated their experience as moderate to low. Following this, they were asked about the anticipated difficulty of the games they were about to play, with 60% expecting the games to be easy. These findings are summarized in Table 5.4, and Figures 5.1 and 5.2.

Table 5.4: Experience with technology and expectations regarding the games

Questions	Median	Q1	Q3	Mean	Std.Dev.
1. I am comfortable with electronic devices such as computers and mobile phones.	5	4	5	4.33	0.90
2. I think the interface and game will be easy to use.	4	3	4	3.67	1.05

Comparing the responses to *'I am comfortable with electronic devices such as computers and mobile phones'* to the ones to *'I think the interface and game will be easy to use'* we can see that there is a relation between them, as both graphs have a similar distribution of responses,

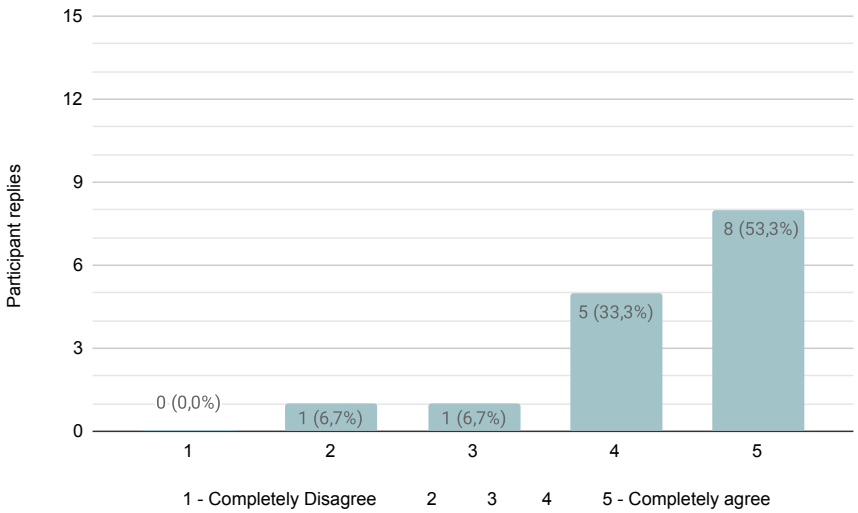

Figure 5.1: Graph showing participant responses to the statement: *'I am comfortable with electronic devices such as computers and mobile phones'*, with 1 representing *'Completely Disagree'* and 5 representing *'Completely Agree'*

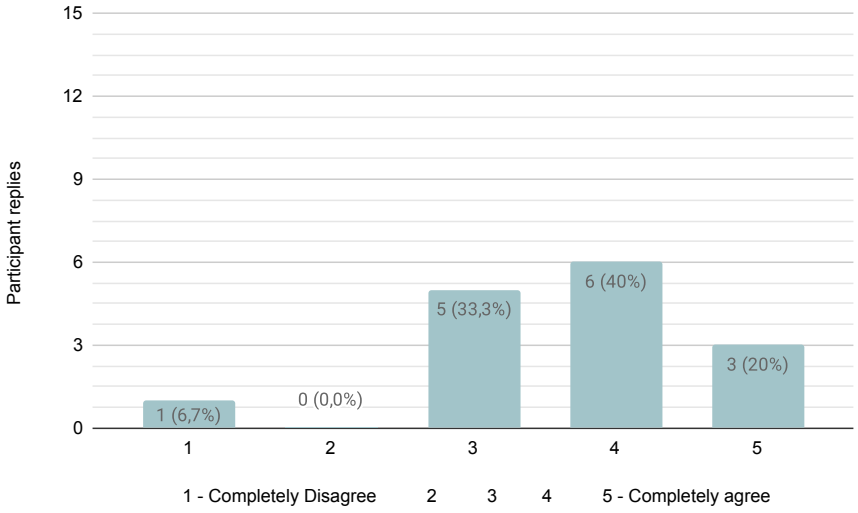

Figure 5.2: Graph showing participant responses to the statement: *'I think the interface and game will be easy to use'*, with 1 representing *'Completely Disagree'* and 5 representing *'Completely Agree'*

with most participants considering themselves comfortable with electronic devices, and most participants expecting the interface and games to be easy to use.

The users were also asked if they had ever played or knew any cognitive training/therapeutic games. As seen in Table 5.5, 46.7% of the participants responded positively, and gave examples of games they had played, such as online math games (like *Cool Math Games* [1]), duolingo, Kahoot, and *educaplay* [2].

[1] https://www.coolmathgames.com/, last accessed:09/2024
[2] https://www.educaplay.com/, last accessed:09/2024

Table 5.5: Participants and their experience with cognitive training/therapeutic games

Response	Participants (N = 15)
Yes	46.7%
No	53.3%

Only 20% of the participants replied that they already knew the concept of TUIs, Table 5.6. As examples of devices that users believed to be TUIs, they mentioned the Nintendo Switch, the Wii, and VR devices. This data was collected to understand the participants' familiarity with TUIs and see if the lack of experience would impact the user experience with the games and the TUI.

Table 5.6: Participants and their experience with TUIs

Response	Participants (N = 15)
Yes	20%
No	80%

Finally, participants were given a list of usability attributes and asked to pick the ones they considered most important in a game, Figure 5.3. The most selected attributes were: ease of use (of the TUI) and varying levels of difficulty. This data was collected to understand the participants' expectations regarding games similar to those they were about to play, allowing us to evaluate our platform against these expectations.

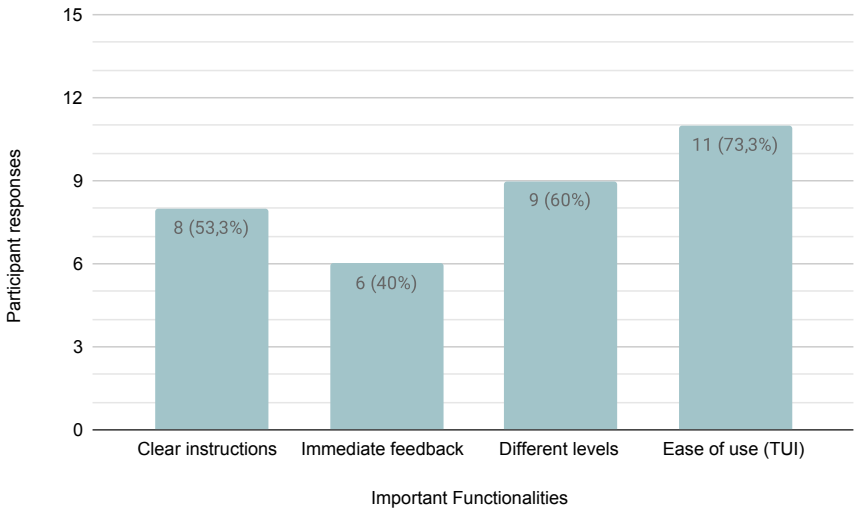

Figure 5.3: Graph showing participant responses to the statement: *'Which of the following attributes do you consider most important platforms such as this one?'*

5.4 Questionnaire Results

At the end of the session, participants were asked to fill a post-questionnaire that collected data that would help us understand the usability of the TUI. Through the use of the SUS questionnaire [11], and some questions tailored around the interaction model of the TUI, we were able to understand the usability of the system and the games developed.

Table 5.7: Experience with technology

Questions	Median	Q1	Q3	Mean	Std.Dev.
1. It was easy to understand how to use the cards.	5	5	5	4.87	0.35
2. It was easy to understand how to use the submit button.	4	3	4	4.07	0.96
3. It was easy to first understand how to configure the games.	3	2	5	3.27	1.16
4. Configuration of the games became easier after the first game.	5	5	5	4.4	1.06
5. It was easy to navigate through the games.	5	5	5	5	0
6. It was easy to change the difficulty level of the games.	5	5	5	5	0

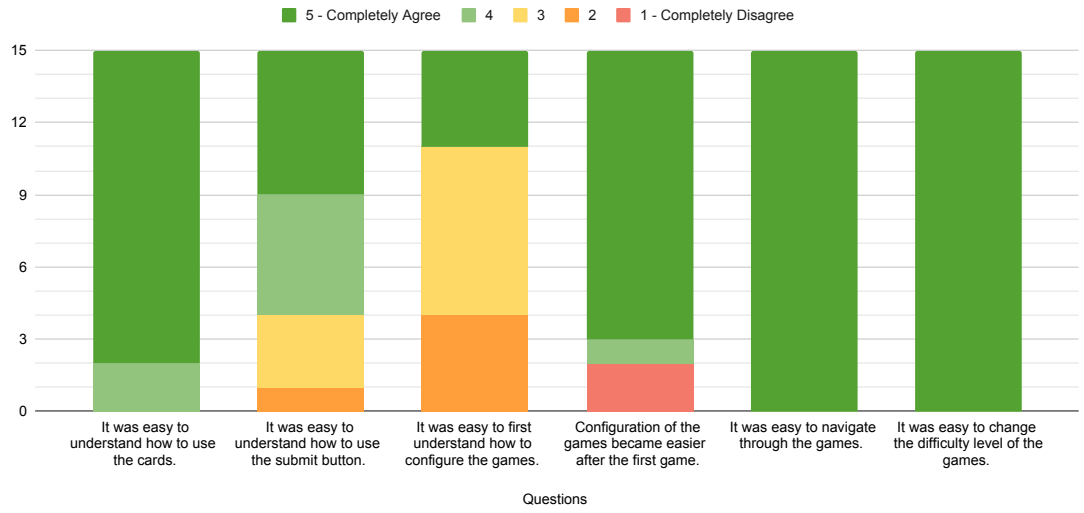

Figure 5.4: Graph showing participant responses to the post-questionnaire

The results of the post-questionnaire can be found in Table 5.7 and Figure 5.4.

Overall, participants found both the TUI and the games accessible, with most able to interact with the games without significant issues. Users quickly understood how to use the cards to complete the games, and the navigation through the games was considered easy by all participants. The interaction method was considered intuitive, and all participants were able to independently complete the test guide without any issues.

The results showed that independently of the participants' experience with technology, age, or scholar level, system was easy to use, and the participants had a positive experience with the games and the TUI. Additionally, players that had no previous contact with TUIs were able to understand how to use the cards and the modules without any issues.

One issue raised is related to the configuration of the games. Question 3, *'It was easy to first understand how to configure the games'* had a median of 3, which was the lowest median of all questions. The following question, *'Configuration of the games became easier after the first game'* does however show that most participants had less trouble configuring the games after the first game. The second-lowest median was for question 2, *'It was easy to understand how to use the submit button'*, which had a median of 4, and a mean of 4.07.

Finally, participants were asked to give feedback and suggestions for improvement of the games and the TUI. It was notable that the initial configuration of the games was somewhat confusing for some participants, as reflected in the post-questionnaire, but after the first configuration was done, all were able to configure the without any issues.

Participants reported that the issue was related to the way the configuration steps were presented rather than the actual difficulty of the configuration. Meaning that the text used to guide the participants through the configuration steps was not clear enough, and some suggested that the configuration steps should be presented in a more visual way. After understanding the configuration steps, participants were able to configure the games without any issues. The last question of the pre-questionnaire, Table 5.3, *'Which of the following attributes do you consider most important platforms such as this one'*, showed that many users considered clear instructions to be an important attribute in a platform like the one they were going to use. And the feedback given by the participants was in line with this, as most of the issues were related to the clarity of the instructions given to the participants.

The first difficulty levels were considered easy by most participants, and the Mastermind game was considered the most engaging game by all participants. Some users asked to play the games again, and some asked for more games to play.

A mistake sometimes made by participants was looking for a submit button on the actual graphical interface of the games, instead of using the modules. This was expected, as most participants are more used to digital interfaces. When asked about how they felt the system could be improved to make it more clear that the button on the TUI was the submit button, most participants suggested that the button should be more visually appealing, and perhaps have a label that says 'submit'.

Overall, most of the issues were more related to the way the games presented information to the participants, and not with the games themselves. The TUI was considered easy to use, and the card based games were considered engaging and an interesting way to create new cognitive training games.

The game interface got a SUS score of 87.5 out of 100, which helps us understand that the interface is easy to use and that the participants were satisfied with it.

5.5 Framework Validation

The developed games validated the framework's design in practical applications. The previous chapter, specifically Section 4.4, detailed the implementation of various use case games, illustrating how different games were built following the design guidelines outlined in Section 4.5. These guidelines demonstrate how to leverage the provided code artifacts and components to efficiently develop new games.

This validation showed that the framework was able to support:

Context Flexibility:
The primary objective of designing multiple games was to validate the framework's ability to support different cognitive training activities using card-based interactions. By employing a unified TUI, the system accommodated a variety of cognitive exercises without significant reconfiguration. The flexible nature of the framework allowed for easy adaptation across games.

Handling Different Difficulty Levels:
A key factor in validating the framework was its adaptability to varying levels of difficulty. Each of the developed games introduced different levels of complexity, which could involve a varying number of card slots, different numbers of cards, or unique rule sets. The framework's ability to handle these changes without structural modifications highlighted its robustness in managing different gameplay mechanics.

Modular and Developer-Friendly Design:
It was important to simplify the development process, allowing developers to concentrate on game logic without needing an in-depth understanding of the framework's underlying architecture. This streamlined development process ensured that game logic could be implemented and modified independently of the system's communication layers, reducing complexity for game developers.

Reusability and Scalability:
A key design goal was ensuring that existing code artifacts and components could be reused across multiple games. This was achieved by maintaining a modular structure that promoted scalability, allowing for the easy creation of new games with minimal overhead. The framework's scalability made it adaptable to future expansions, and updates to the system could be made without disrupting existing games.

Sensor Integration:
The framework's design allowed for the integration of new sensors and devices, providing flexibility for future enhancements. This feature ensures that the system can be extended to support additional sensors or input devices, enabling the creation of new game mechanics

and interactions.

Supporting Multiple Card Decks:
Another important element of the framework's evaluation was testing how easily different card decks, as well as other sensor inputs, could be created and integrated. The system's architecture allowed for the straightforward creation and configuration of card decks, as well as other sensor readings, in the form of JSON files with semantic translations. This method ensures that developers can quickly adapt and configure games for diverse gameplay experiences, whether by modifying card interactions or incorporating new sensor data, without the need to change the underlying framework.

Beyond these core features, several key aspects of the framework were evaluated to ensure it meets the needs of developers, especially regarding learning curve, customizability, multiplayer support, and error reporting.

One of the strengths of the framework is its relatively short learning curve. The encapsulation of communication and interaction with the TUI remove the need for developers to have a deep understanding of the system's architecture. We created configuration scripts and scenes that encompass the whole configuration process. This can be used in its entirety. These configurations steps populate configuration data that can be accessed in any game or game mode, allowing for the creation of new game modes that require different configurations.

Customizability was key focus during the development of the framework. The system supports the creation of a wide range of game genres and contexts, providing flexibility in gameplay mechanics, card deck designs, and difficulty levels. The use of configuration scripts allows for the quick adaptation of games to different settings, and the framework's modular design ensures that game components, such as card decks, rules, and modules, can be easily customized. Developers can extend or modify existing components, or introduce new game types, with no disruption to the system's architecture. As a result, the framework provides a flexible platform for developers to create unique and diverse game experiences.

The framework's scalability extends to local multiplayer support as well. While it was initially tested with single-player games, it can be extended to accommodate multiplayer game modes without significant structural changes. The modular design of the TUI's hardware allows for the addition of more cards and card slots to support multiplayer interactions. Developers can implement multiplayer mechanics by building on the existing communication and interaction framework in place, focusing on game logic and multiplayer-specific interactions without needing to modify the core elements.

Error reporting and debugging within the framework are well-supported through detailed logs and error-handling mechanisms. The framework provides clear feedback during game development and runtime, allowing developers to quickly identify and fix errors.

The case where the developers want to change the interaction mode of the games, by the integration of new sensors and devices, does not require changes to the existing game or Middleware. This because, the integration of new forms of sensor data is compatible with the existing translation tool in place to convert card ID data into game data.

5.6 Discussion

5.6.1 Usability And Performance Evaluation

The evaluation process demonstrated that the system and framework performed well across a variety of metrics set out in the evaluation criteria. The participants' feedback indicated that the TUI was generally easy to use. Even participants with limited experience with technology or TUIs were able to grasp the card-based interaction method with little difficulty.

The SUS score of 87.5 further supports the system's high usability, affirming its potential for broader application in therapeutic and educational settings.

However, as previously mentioned, the configuration step posed some challenges, particularly in terms of how instructions were communicated. Several participants expressed confusion during the configuration phase, suggesting that more visual cues or clearer instructions would improve the user experience. But once the initial configuration was completed, participants had no issues configuring the games. The same could be applied to the submit button, which some participants found less noticeable than expected.

During testing, the system handled changes in gameplay complexity and physical card setups without structural modifications. This adaptability makes it ideal for a variety of cognitive training scenarios, supporting the creation of games with diverse mechanics. When participants used more than one module, the system's responsiveness and reliability remained consistent, with no reported inconsistencies, or performance issues.

5.6.2 Developer Experience And Framework Evaluation

The framework's ability to adapt to different game modes, difficulty levels, and configurations was a key strength highlighted during the evaluation. From the developer's perspective, the framework demonstrated a short learning curve, customizable game logic, and modular components, making it easy to create new games.

The configuration scripts provided an efficient way to set up new game modes and integrate new sensors or interaction types.

Once the base game structure was set in place (Middleware, Communication Handler, configuration tools...) it was easy to build different games. These, at their core, used the same structure of communication and configuration, leaving only the game logic to be implemented. This separation of concerns removes the need to deeply take into consideration the underlying architecture.

Error reporting was another strong point of the framework. Clear logs and feedback mechanisms during development allowed for efficient debugging, ensuring that errors could be quickly identified and addressed.

As previously mentioned, the framework allows for the creation of multiplayer games, the addition of more cards and card slots to support multiplayer interactions, and the integration of new sensors and devices, without changes to the remaining architectural layers.

5.7 Summary

In summary, the evaluation demonstrated that the system meets its core goals of providing an engaging, adaptable, and user-friendly platform for cognitive training. Participants found the games engaging and the TUI easy to use independently of their experience with technology, age or scholar levels. Feedback from participants indicated a positive experience, with suggestions for minor improvements to enhance usability such as more visual cues and clearer instructions at the use case game level.

The framework was able to support the creation of different games, with different difficulty levels, and different card decks, showcasing its flexibility and scalability, even at the hardware level, providing a solid foundation for future expansions and updates.

The evaluation process provided valuable insights into the usability and performance of the system, as well as the effectiveness of the TUI in creating new cognitive training games, which suggest that it holds significant potential for further development and deployment in real-world applications.

In the final chapter, we will present the conclusions of this research, as well as the contributions and future work that can be done to improve the system and the framework.

6

Conclusion

This dissertation has introduced a framework designed to streamline the creation of cognitive training games using a card-based Tangible User Interface (TUI). The development of two distinct games, Memory and Mastermind, served as validation for the framework, demonstrating its flexibility, scalability, and adaptability across different difficulty levels, card decks, and gameplay mechanics.

By leveraging developer-friendly architecture, the framework significantly reduces the learning curve required to design and implement new games, Developers are able to focus on game logic rather than complex system integration, and the reuse of code artifacts ensures efficient development across multiple game modes.

The framework's customizable nature allows developers to design games that target specific cognitive skills by adjusting game parameters, card designs, and difficulty levels. This is facilitated by the framework's modular design, and the choice to have cards as the primary game element. Cards are highly versatile components that are easily adaptable to different gameplay mechanics, as presented in Chapter 3. Cards are also a familiar that people of all ages and backgrounds know how to interact with. This makes card-based games accessible and easy to learn for a wide range of users, including demographics that may have very different needs and abilities. The physical nature of the cards also provides a tactile and visual element that can be used to enhance the gameplay experience, making the games more engaging and immersive.

The way the platform is designed also allows for easy expansion of the number of card reading modules to be used in the different games. Having a self-managing hardware setup allows for the system to be easily scaled to accommodate more players or more complex games. The Middleware layer is responsible for managing the communication between the different modules, ensuring that the game logic is not affected by the number of modules connected, while establishing and maintaining a connection with the game client and the TUI modules.

Providing flexibility of hardware used was also a key design decision, as it allows for it to be adapted to different sensor data to be received on the game side and translated into game semantics, the deck creation tool that allows for the creation of new card Decks, also

works for this scenario as it simply reads an input and creates a semantic translation of the data. Additionally, the possibility to have more than one card reading module allows for the system to be easily scaled to accommodate local multiplayer games, with very similar game logic to the single player games created in this thesis.

The adaptability of platform positions it as a possibly valuable tool in cognitive rehabilitation settings and education. Games created with this framework can be tailored to meet the needs of individual users, allowing for personalized training programs that target specific cognitive skills.

The current implementation of the framework is limited to a single player, and the games created are relatively simple in terms of gameplay mechanics, and despite the framework being designed to allow local multiplayer games, multiplayer in different machines would require changes to the current architecture.

Due to the choice of having Bluetooth Low Energy as the communication protocol and ESP-NOW as the communication protocol between the card reading modules, the hardware is limited to microcontrollers of the ESP32 family.

The framework was designed to be sensor-agnostic, but changing the sensors used will still require changes to the TUI microcontroller's firmware, as the only scenario offered in this thesis was the use of RFID cards.

The use of a Bluetooth protocol can be limiting in terms of compatibility with different devices, as not all devices have Bluetooth capabilities. Most portable devices have Bluetooth capabilities, but some desktop computers may not have Bluetooth capabilities, and the user would need to have an external Bluetooth dongle to be able to play the games.

6.1 Future Work

The framework has a lot of potential for future work, as it can be expanded in many ways. One of the most immediate future work would be to create more robust documentation support for development. This documentation would include a more detailed explanation of the game's architecture, the reusable code components and how to use them, and how to create new games using the framework.

To help guide local multiplayer development, an example of a local multiplayer game would be a good addition to the platform, as it would show how to create a game that uses more than one card reading module and how to manage commands received from different modules into different user input commands using only game logic.

It would also be important to provide support for non-local multiplayer games, as this would allow for the creation of games that can be played by people in different devices, which can be useful for games where you may want players not to see each other's cards, games that require a larger playing area than a single device can provide, or other scenarios where players need to be separated. For this to be possible, one option is to make changes at the level of the firmware using ESP-NOW to coordinate data from the card

reading modules of different players, while keeping the rest of the platform components independent of each other.

Creating more sensor firmware codes to allow some more ready-to-use sensor configurations to be used with the framework if necessary would also be a good addition to the framework.

Additionally, still at the hardware level, transforming the current communication protocol behavior and set up into a library that can be used by future developers to create new games that require changes to the sensor data being collected. This is a step to make encapsulate that portion of the firmware and make it more reusable and easier to understand, without the need to understand the whole firmware code.

To make this platform more appealing to educators and therapists, it would be relevant to create a way to track the progress of the users, and to provide feedback on the user's performance, so that these professionals can have a better understanding of the user's progress and adapt the training program accordingly. This could be done in the form of an external application that would connect to the game client and receive data from user sessions.

Bibliography

[1] C. C. Abt. *Serious games*. University press of America, 1987 (cit. on p. 6).

[2] Arduino. *Adafruit SSD1306 library documentation*. 2024. URL: https://www.arduino.cc/reference/en/libraries/adafruit-ssd1306/ (visited on 2024-09-24) (cit. on p. 40).

[3] Arduino. *ESP-NOW library documentation*. 2024. URL: https://docs.arduino.cc/tutorials/nano-esp32/esp-now/ (visited on 2024-09-24) (cit. on p. 39).

[4] Arduino. *ESP32 BLE library documentation*. 2024. URL: https://www.arduino.cc/reference/en/libraries/esp32-ble-arduino/ (visited on 2024-09-24) (cit. on pp. 33, 34, 39).

[5] Arduino. *MFRC522 library documentation*. 2024. URL: https://www.arduino.cc/reference/en/libraries/mfrc522/ (visited on 2024-09-24) (cit. on p. 40).

[6] D. Bachmann, F. Weichert, and G. Rinkenauer. "Review of Three-Dimensional Human-Computer Interaction with Focus on the Leap Motion Controller". In: *Sensors* 18.7 (2018). ISSN: 1424-8220. DOI: 10.3390/s18072194. URL: https://www.mdpi.com/1424-8220/18/7/2194 (cit. on p. 19).

[7] A. O. Barajas, H. Al Osman, and S. Shirmohammadi. "A Serious Game for children with Autism Spectrum Disorder as a tool for play therapy". In: *2017 IEEE 5th International Conference on Serious Games and Applications for Health (SeGAH)*. IEEE. 2017, pp. 1–7 (cit. on p. 10).

[8] J. C. Bastien. "Usability testing: a review of some methodological and technical aspects of the method". In: *International journal of medical informatics* 79.4 (2010), e18–e23 (cit. on p. 18).

[9] BLEAK. *BLEAK library documentation*. 2024. URL: https://bleak.readthedocs.io/en/latest/ (visited on 2024-09-24) (cit. on p. 50).

[10] E. A. Boyle et al. "An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games". In: *Computers & Education* 94 (2016), pp. 178–192 (cit. on pp. 2, 3).

[11] J. Brooke. "SUS: a retrospective". In: *Journal of usability studies* 8.2 (2013), pp. 29–40 (cit. on pp. 19, 71).

[12] J. W. Burke et al. "Serious games for upper limb rehabilitation following stroke". In: *2009 Conference in Games and Virtual Worlds for Serious Applications*. IEEE. 2009, pp. 103–110 (cit. on pp. 8–10, 21).

[13] S. Chiasson and C. Gutwin. "Design principles for children's technology". In: *interfaces* 7.28 (2005), pp. 1–9 (cit. on p. 18).

[14] F. J. Dian, A. Yousefi, and S. Lim. "A practical study on Bluetooth Low Energy (BLE) throughput". In: *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. 2018, pp. 768–771. DOI: 10.1109/IEMCON.2018.8614763 (cit. on p. 23).

[15] A. Dix. *Human-computer interaction*. Pearson Education, 2003 (cit. on p. 12).

[16] Espressif. *Bluetooth library documentation*. 2024. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/bluetooth/index.html?highlight=bluetooth (visited on 2024-09-24) (cit. on pp. 33, 34).

[17] Espressif. *ESP-NOW library documentation*. 2024. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_now.html?highlight=esp%20now (visited on 2024-09-24) (cit. on p. 33).

[18] Espressif. *I2C library documentation*. 2024. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/i2c.html (visited on 2024-09-24) (cit. on p. 40).

[19] Espressif. *Wi-Fi library documentation*. 2024. URL: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/network/esp_wifi.html (visited on 2024-09-24) (cit. on p. 33).

[20] G. W. Fitzmaurice, H. Ishii, and W. A. Buxton. "Bricks: laying the foundations for graspable user interfaces". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1995, pp. 442–449 (cit. on p. 13).

[21] P. García-Redondo et al. "Serious Games and Their Effect Improving Attention in Students with Learning Disabilities". In: *International Journal of Environmental Research and Public Health* 16.14 (2019). ISSN: 1660-4601. DOI: 10.3390/ijerph16142480. URL: https://www.mdpi.com/1660-4601/16/14/2480 (cit. on p. 2).

[22] C. S. González et al. "TANGO: H: creating active educational games for hospitalized children". In: *Management Intelligent Systems: Second International Symposium*. Springer. 2013, pp. 135–142 (cit. on p. 7).

[23] C. S. González-González et al. "Serious games for rehabilitation: Gestural interaction in personalized gamified exercises through a recommender system". In: *Journal of biomedical informatics* 97 (2019), p. 103266 (cit. on pp. 2, 7, 8).

[24] J. Hamari, J. Koivisto, and H. Sarsa. "Does Gamification Work? – A Literature Review of Empirical Studies on Gamification". In: *2014 47th Hawaii International Conference on System Sciences*. 2014, pp. 3025–3034. DOI: 10.1109/HICSS.2014.377 (cit. on p. 6).

[25] M. A. Iqbal et al. *Enabling the internet of things: fundamentals, design and applications*. John Wiley & Sons, 2020 (cit. on p. 20).

[26] H. Ishii and B. Ullmer. "Tangible bits: towards seamless interfaces between people, bits and atoms". In: *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*. 1997, pp. 234–241 (cit. on p. 13).

[27] S. Jordà et al. "The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces". In: *Proceedings of the 1st international conference on Tangible and embedded interaction*. 2007, pp. 139–146 (cit. on pp. 16, 17).

[28] F. Laamarti, M. Eid, and A. E. Saddik. "An overview of serious games". In: *International Journal of Computer Games Technology* 2014 (2014), pp. 11–11 (cit. on p. 6).

[29] H. S. A. Latiff, R. Razali, and F. F. Ismail. "User interface design guidelines for children mobile learning applications". In: *International Journal of Recent Technology and Engineering (IJRTE)* 8.3 (2019), pp. 3311–3319 (cit. on p. 17).

[30] J. R. Lewis. "Psychometric evaluation of the post-study system usability questionnaire: The PSSUQ". In: *Proceedings of the human factors society annual meeting*. Vol. 36. 16. Sage Publications Sage CA: Los Angeles, CA. 1992, pp. 1259–1260 (cit. on p. 19).

[31] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[32] E. J. Marco et al. "Sensory processing in autism: a review of neurophysiologic findings". In: *Pediatric research* 69.8 (2011), pp. 48–54 (cit. on p. 2).

[33] P. Marshall. "Do tangible interfaces enhance learning?" In: *Proceedings of the 1st international conference on Tangible and embedded interaction*. 2007, pp. 163–170 (cit. on p. 13).

[34] I. Mayer. "Towards a comprehensive methodology for the research and evaluation of serious games". In: *Procedia Computer Science* 15 (2012), pp. 233–247 (cit. on p. 19).

[35] J. Patten et al. "Sensetable: A Wireless Object Tracking Platform for Tangible User Interfaces". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '01. Seattle, Washington, USA: Association for Computing Machinery, 2001, pp. 253–260. ISBN: 1581133278. DOI: 10.1145/365024.365112. URL: https://doi.org/10.1145/365024.365112 (cit. on pp. 15, 16, 20).

[36] J. Piaget. "How children form mathematical concepts". In: *Scientific American* 189.5 (1953), pp. 74–79 (cit. on p. 12).

[37] J. J. Rieser, A. E. Garing, and M. F. Young. "Imagery, action, and young children's spatial orientation: It's not being there that counts, it's what one has in mind". In: *Child Development* 65.5 (1994), pp. 1262–1278 (cit. on p. 12).

[38] J. Ryu et al. "Wireless control of a board robot using a sensing glove". In: *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE. 2014, pp. 423–428 (cit. on p. 23).

[39] J. Sánchez and R. Olivares. "Problem solving and collaboration using mobile serious games". In: *Computers & Education* 57.3 (2011), pp. 1943–1952 (cit. on pp. 12, 13).

[40] C. Schönauer et al. "Chronic pain rehabilitation with a serious game using multimodal input". In: *2011 International Conference on Virtual Rehabilitation*. IEEE. 2011, pp. 1–8 (cit. on p. 7).

[41] G. Stamou et al. "4.11 - 2D and 3D Motion Tracking in Digital Video". In: *Handbook of Image and Video Processing (Second Edition)*. Ed. by A. BOVIK. Second Edition. Communications, Networking and Multimedia. Burlington: Academic Press, 2005, pp. 491–XVIII. ISBN: 978-0-12-119792-6. DOI: https://doi.org/10.1016/B978-012119792-6/50093-0. URL: https://www.sciencedirect.com/science/article/pii/B9780121197926500930 (cit. on p. 21).

[42] T. Susi, M. Johannesson, and P. Backlund. *Serious Games : An Overview*. Tech. rep. HS-IKI-TR-07-001. University of Skövde, School of Humanities and Informatics, 2007, p. 28 (cit. on pp. 1, 6).

[43] T. Susi, M. Johannesson, and P. Backlund. "Serious games: An overview". In: (2007) (cit. on p. 1).

[44] L. Terrenghi et al. "A cube to learn: a tangible user interface for the design of a learning appliance". In: *Personal and Ubiquitous Computing* 10 (2006), pp. 153–158 (cit. on pp. 11–13).

[45] Unity. *Unity Transport library documentation*. 2024. URL: https://docs.unity3d.com/Packages/com.unity.transport@2.0/manual/client-server-simple.html (visited on 2024-09-24) (cit. on p. 52).

[46] R. Watanabe et al. "The soul of ActiveCube: implementing a flexible, multimodal, three-dimensional spatial tangible interface". In: *Computers in Entertainment (CIE)* 2.4 (2004), pp. 15–15 (cit. on pp. 13–15, 22).

[47] J. Wiemeyer and A. Kliem. "Serious games in prevention and rehabilitation—a new panacea for elderly people?" In: *European Review of Aging and Physical Activity* 9.1 (2012), pp. 41–50 (cit. on pp. 2, 3).

[48] S. J. Yun et al. "Cognitive training using fully immersive, enriched environment virtual reality for patients with mild cognitive impairment and mild dementia: Feasibility and usability study". In: *JMIR Serious Games* 8.4 (2020), e18127 (cit. on pp. 9, 10, 23, 24).

[49] H. M. Zakari, M. Ma, and D. Simmons. "A review of serious games for children with autism spectrum disorders (ASD)". In: *Serious Games Development and Applications: 5th International Conference, SGDA 2014, Berlin, Germany, October 9-10, 2014. Proceedings 5*. Springer. 2014, pp. 93–106 (cit. on p. 2).

[50] Y. Zheng et al. "A review on serious games for ADHD". In: *arXiv preprint arXiv:2105.02970* (2021) (cit. on p. 2).

[51] Z. Zhou, A. D. Cheok, and J. Pan. "3D story cube: an interactive tangible user interface for storytelling with 3D graphics and audio". In: *Personal and Ubiquitous Computing* 8 (2004), pp. 374–376 (cit. on pp. 13, 14, 20).