Expertise
and insight
for the future

Teemu Koskenneva

Reality enhanced digital boardgame

Metropolia University of Applied Sciences

Bachelor of Engineering

Information and communication technology

Bachelor's Thesis

4 December 2020

Author Title Number of Pages Date	Teemu Koskenneva Reality enhanced digital boardgame 37 pages 4 December 2020
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Smart Systems
Instructors	Joseph Hotchkiss, Senior Lecturer

As the era of physical media is slowly coming to an end, oddities such as boardgames aided by audiovisual media, including VHS tapes and DVDs, have already disappeared from the market. The goal of this thesis was to design a modern version of these types of boardgames, however with the twist of having most of the game take place inside a video game, while the physical boardgame would act as a support for the game.

To make the boardgame more of a spectacle, instead of normal static game pieces, the pawns were represented by robots that replicated events from the video game as they happened. A Zumo robotic platform manufactured by Pololu was used for the pawns' bodies, as they were equipped with tank treads and infrared sensors, both of which were used to follow the guide lines on the printed gameboard. To act as the Zumo's brain a PSoC 5lp development board was chosen, for its ability to move the motors with pulse width modulation, to receive signals from the game through a Wi-Fi module, and to read and process data coming from them.

The game was made to be a simple demonstration of how a game could interact with the Zumo, by communicating through the MQTT light weight message protocol. Therefore, the game was designed to be a simple slow pace turn based strategy game that is easy to build upon in the future.

After some setbacks, the thesis had to be scaled down to a purely theoretical work, as facilities and resources were unavailable during the Covid-19 pandemic. Because of this the thesis was written keeping in mind that its contents could be replicated by using alternative devices and still be able to apply the basic concepts to them.

To compensate for the lack of a practical portion in the project, a lot of the time reserved for it was instead used to design future features and discuss their impact on the whole experience, making the thesis an adequate blueprint for a project with a lot of adaptability and a suggestion for the future.

Keywords	digital, boardgame, line following

Tekijä Otsikko Sivumäärä Aika	Teemu Koskenneva Reality enhanced digital boardgame 37 sivua 4.12.2020
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Älykkäät järjestelmät
Ohjaajat	Lehtori Joseph Hotchkiss

Lopputyö pyrkii esittämään teoreettisen mallin modernisoidulle versiolle audiovisuaalisilla medioilla parannelluille lautapeleille, jotka rikastuttivat ja helpottivat lautapeleillä pelaamista VHS-kasettien ja myöhemmin DVD-levyjen avulla. Tarkoituksena on esittää teoreettinen työ, joka on helppo ymmärtää, soveltaa ja toteuttaa.

Tämä moderni versio pyrkii vaihtamaan lautapelin ja audiovisuaalisen median roolit, jolloin lautapelin tarkoitus on toimia videopeliä rikastavana elementtinä. Käytännössä tämä toteutetaan käyttämällä tulostettua paperia pelilautana ja robotteja pelinappuloina, jotka pyrkivät kopioimaan videopelin sisäisiä tapahtumia, joita parannetaan pelilaudalla ääni- ja valoefekteillä.

Nappulat hyödyntävät Pololun Zumo -alustojen sensoreita ja moottoreita kentällä liikkumiseen ja aistimiseen. Näiden alustojen aivoina sen sijaan toimivat PSoC 5lp -kehityslevyt, jotka ottavat vastaan pelin tietoja sekä sensorien lukemia ja lähettävät niiden perusteella käskyjä Zumolle.

Työssä käytettävät peruskonseptit, kuten pulssinleveysmodulaatio (PWM) tai se, kuinka aktiivinen infrapunasensori toimii sekä projektissa käytetyt laitteet valittiin, sillä periaatteella, että kaikki tarvittavat resurssit löytyisivät Metropolia Ammattikorkeakoulun tiloista. Pyrittiin myös siihen, että projekti olisi myös helppo toteuttaa vastaavilla vaihtoehtoisilla laitteilla.

Tämä ajatusmaailma jaettiin myös videopeliosion kanssa, mikä pelin kehitystyön lisäksi keskittyy valtaosin osoittamaan koodauksen ajatusmaailmaa vuokaavioiden avulla. Tämän tarkoituksena on helpottaa menetelmien soveltamista koodauskielestä riippumatta.

Avainsanat	

Contents

List of Abbreviations

AI	Artificial intelligence
HP	Hit points / Health points
TBS	Turn based strategy
SMG	Sub machine gun
PWM	Pulse width modulation
MQTT	Message queuing telemetry transport

1 Introduction

As humanity is slowly transitioning away from physical media, a peculiar novelty that has existed for decades, has suddenly vanished from the market, becoming another forgotten casualty in the evolution of entertainment.

This forgotten novelty is the art of adding flavor to a boardgame by bundling it with a VHS tape or a DVD. During the VHS era the role of the tapes was usually to set the mood by showing actors reenacting scenes from the game or explaining the rules with added showmanship.

At the start of the start of the 21st century, this type of multimedia entertainment took a major leap forward with the rise of the DVDs. This new technology allowed the audiovisual media portion to come forth from the backgrounds and really become part of the game. The DVDs allowed a limited amount of intractability by selecting options from the DVDs menu that matched with the current gameboard event. These interactions could trigger randomized events that could affect the gameboard or just add some flare to the experience with images and sounds.

However, as the presence of the physical media is slowly starting to fade, so will this already somewhat obscure oddity fade in the history. To breathe some new air into this entertainment concept in this day and age, it would be crucial to continue the trend of moving the game away from the board and into the screen. Switching the roles of the screen and board will allow to take full advantage of a videogame's easy methods to teach the game, allowing more varied gameplay and the ability to easily update the game, while also having a real-world representation of the game world's events.

Due to circumstances, such as Covid-19 pandemic and the closing of the Leppävaara campus. There is a lack of workspaces and other resources, which caused the project to be scaled down to a purely theoretical work. However, this will not reduce the amount of research and consideration put into the physical aspects of the project.

To make the gameboard a real part of the videogame experience, simple robots will be used as the gameboard's pawns. These robots comprised of Zumo robot platforms made

by Pololu which have motors and infrared sensors that make them perfect for line following. This ability will be vital as the gameboard itself will be a large printed paper with guiding lines helping the robots to reliably navigate from one game square to another.

To be able to control these robots a Cypress manufactured PSoC 5lp development board will be used as its brain. The board will interpret the readings coming from the infrared sensors and will control the Zumo's motors with pulse width modulation signals, to move itself towards its destination on the gameboard.

The last piece of the gameboard puzzle is a Wi-Fi module that lets the PSoC to communicate with the video game inside the computer, by using a light weight messaging protocol called MQTT. This protocol allows a reliable and energy efficient way to send data between devices, which is perfect for battery powered devices such as the Zumo.

As the project's main focus will be the hardware, the accompanying videogame will just be a simple proof of concept demonstration, which aims to show ways how the robots and the video game can interact with each other. The game is a turn-based strategy game similar to chess, as this kind of game would be the easiest to understand and effective way to demonstrate the robot's ability to move around the board and possibly replicate events from the game with lights and sounds.

2 Theoretical Background

At the start of the project QR codes were considered to be used as the center of corners and intersections that are marked with a red square in the prototype illustration of the board seen in Figure 1 below. The purpose of these was to help game pieces to confirm their whereabouts and orientation on the board by reading the QR codes, which would help the pieces notice if they have gone the wrong way or if there are any discrepancies between the commands and their current situation on the board. The QR codes would also reduce the artificial intelligence (AI) requirements of the Zumo to interpret line shapes, as the codes could tell the piece what kind of section it is on top of. However, as the Zumo platform lacks a QR reader but contains everything else, it would be more efficient to use it as it is and find a way to navigate the lines without the QR codes.

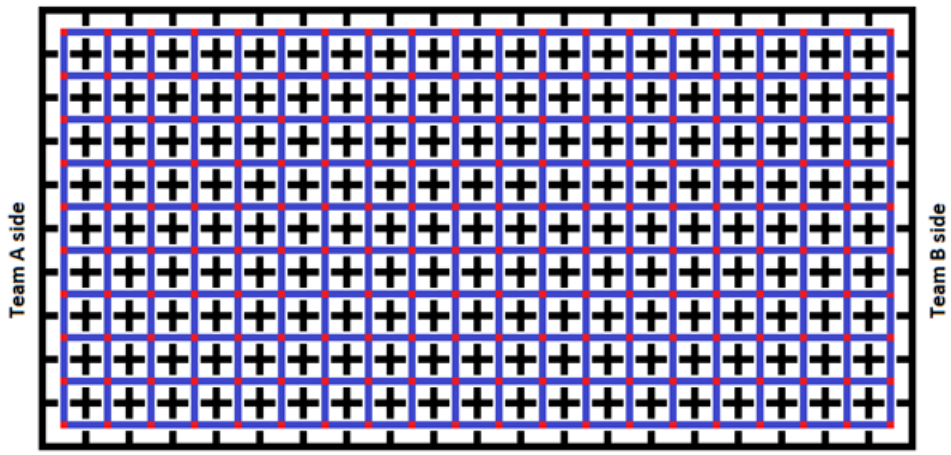

Figure 1. A prototype illustration of the gameboard.

Therefore, as the Zumo was already planned to follow the lines using reflection sensors, navigating the corners and intersections was done with sensor interpretation and logic. To make it easier for the robots to navigate, the gameboard uses a grid layout as seen in Figure 1 above and will work similarly to a chessboard. The black squares in Figure 1 above represent zones that are like the square spaces in a chessboard and follow the same coordinate naming sense. Therefore, the top left zone was named A1 and the zone next to it was called A2 and the one below A1 was B1.

As seen in Figure 1 above and more clearly in Figure 2 below, each zone has a guideline represented in blue that the robot will use to navigate its way out of each zone. The center of each zone is represented as a red square, which also marks every crossroad

or corner in the guideline. These highlighted sections will rely on the robot's logic to successfully change lanes or turn without losing the precious guideline.

Figure 2. Closer look at a single corner zone on the gameboard.

The Zumo has six reflection sensors in front of it [1; 2], divided into two pairs for the left and right side. The reflection sensors work by emitting an infrared light below the Zumo and then measure how much of that light is reflected back. When the sensor is on top of a darker surface, less light is reflected back to the sensor [3], which is why the field is made dominantly white to help the black guiding lines to stand out more for the sensors. The line tracking happens by having the Zumo try and keep the black line between the two innermost sensors. If the light reflecting back to either sensor decreases, that side is getting closer to the line and the Zumo should change its direction slightly to keep the line at the center.

Using these reflection readings, it is somewhat easy to keep track when the unit leaves and arrives at a zone. Because the guidelines leading from zone to zone are straight, it can be deduced that the only situations where the outermost sensors receive a decreased amount of reflected light is when the robot arrives at an intersection. In other words, the center of a zone. In theory this would work as follows. The unit starts at the top left corner of the field facing the bottom of the field, and it is requested to move one zone down and two to the right. In this situation, the unit would start to go forward keeping the line between the two innermost sensors until two left sensors notice a black line. Then the unit would start to turn left ignoring the outermost sensors until there is a line between the two innermost sensors. After this it would continue going forward, stopping only after all the sensors go simultaneously over a black line twice, as this would indicate the act of going over two four-way crossroads. After this the unit has reached its destination.

Mapping

There are different ways to share the map between the game and the robots. One way of doing it could be to have the map inside both the robots and the PC. If done like this, both sides would have a map as a list and they could keep track of where the robots are while keeping each other informed of their own positions. This way it is less likely that the robots will be lost on the map or at least this makes it more likely to catch desynchronizations between both parties. The downside to this is that either the maps have to be static in size or there needs to be more communication at the start of the match where each robot gets the layout of the map and other important information about the upcoming gameboard.

It also feels wasteful to have basically the same information duplicated onto each robot. However, this could reduce the amount of data needed to be sent after the initial set up as the game could just send in the coordinates on the map to the robot and it would do the rest in theory. This could cause some problems later if pathfinding has to be implemented because that could be really resource intensive and therefore draining the battery life of the robots needlessly.

The pathfinding could be done on the computer, which could then sent the data to the robot's as individual directions, or if the robot's themselves got a replica of the map, the path to the desired location could be sent to a robot in the form of waypoint coordinates, meaning that the robot would get the coordinates to the waypoints one by one until it arrives at its destination. These waypoints would be located at the end of each vertical or horizontal vector that starts from the robot's current location and ends at the center of a zone.

3 Game Theory

In the proof of concept demo, only the hit points (HP) [4; 5], damage, armor and range attributes will be used on the player characters to make the game as simple as possible while still showing some rudimentary mechanics of the game, such as damage calculations and hit detection. These attributes work as follows: HP is a numerical value that shows the condition of the individual or component it is associated with. When this numerical value reaches zero the component or individual is considered destroyed or dead in the game. Some games use other terms such as fainted or fled the scene, but the main point is that the target is unusable for the rest of the match or game depending on the rules.

Damage is an integer value that is closely tied to HP because it represents a numerical value to any damage that the target is subjected to. For instance, if a bullet or a sword hits its intended target, the HP of the target would decrease by the predetermined amount of damage that the bullet or sword has been set to inflict.

Armor is also an integer value that works similarly to HP but depending on the mechanics it can work a bit differently from game to game. For example, some games use armor as an extra HP reserve that depletes first but cannot regenerate over time unlike HP. Others flip this around and have the armor be an extra HP reserve that does not regenerate but still protects the main HP value that can regenerate, and some use it as a fixed value that does not permanently change during a match and is instead deducted from damage during damage calculations. In this case the received damage is firstly deducted from the armor value, after which the remainder is deducted from HP (if it is over 0). During the testing phase, the third version of armor mechanics will be used, but it might be replaced with other options or even a mixture of the option later in the project's lifespan.

Range is the final attribute used in the testing phase, and instead of one integer, it is comprised of two, a minimum and a maximum range. These integer values are used to determine if an attack from an equipment or weapon is in range to hit the target. For example, a weapon with the minimum range of one and a maximum range of five can hit a target that is in between one and five game spaces from the weapon's user. If the target is closer than the minimum space to the user (in this case right next to it), the user cannot perform the attack with the desired weapon (think of something like trying to shoot someone while the barrel of the gun goes past the targets head). On the other hand, if the target is further away than the maximum value (in this case six spaces away from

the weapon user), then the attack is either automatically considered a miss or its chances of hitting the target are drastically reduced.

As seen in table 1 below, there are a few more attributes planned to be added into the game on a later date to add complexity and more strategic possibilities.

Table 1. A table of possible attributes and how they could work.

Attribute	**Explanation**
HP (Hit Points) (will be in the demo build)	Integer value that shows the condition of the subject it is attached to.
Damage (will be in the demo build)	Integer representation of the damage the target subject received from an attack or effect.
Armor (will be in the demo build)	Integer value that represents the amount of protection an equipment offers. This integer value is used to reduce or negate the amount of damage the subject receives.
Range (will be in the demo build)	Array of two integer values indicating how far and close to a target the equipment it is assigned to can be used.
Ammo (will be in the demo build)	Array of two integer values showing how much ammunition is left and how much is the maximum capacity. The values are used to show how many times the equipment can be used before reloading.
Speed (planned for later)	Integer value that represents how much a player's pawn can move during its turn. This value is usually negatively impacted by the increase of the weight attribute.
Accuracy (planned for later)	Integer value that tells the average change of hitting the target with the equipment it is attached to. Usually shown as success percentage and the value can be affected by outside factors in combat.
Weight (planned for later)	Integer value that shows the equipment's theoretical weight. This value is used to limit the amount of equipment a player can take to a battle.
Damage type (planned for later)	Value that shows in what form the damage is dealt to the target (e.g. piercing, cutting, blunt force, heat). This value increases the equipment options as there are different types of armors that are effective against certain types of attacks.

The game will fall under the genre of turn based strategy games (TBS). This genre of games typically functions similarly to card or board games such as Uno or chess where

each player takes a turn to do their move in the game. [6.] The project's game part functions a lot like chess in the sense that the game will use a board made out of spaces and each player will take turns to assign actions (such as move or attack) to their pawns.

For the purpose of this project the board will be standardized to a certain size (depicted as how many zones wide and long it is) to help with the testing of the robot and to prove the concept, but later it would be ideal if the board could be made as long and wide as the players themselves want it to be. Later the board will also include passive and active obstacles, and objectives to add flavor to the gameplay. Passive obstacles are objects such as buildings or terrain that don't do anything and the only purpose of which is to obscure players' movements by only being there. Active obstacles on the other hand are objects or terrain that cause damage, status conditions or both. For example, a swampy area would slow down a pawn's movement, or fog would reduce a weapon's range between the weapon and target. Objectives are objects or areas that are usually used with different game modes and add something to the game flow. An example of this would be the game mode 'capture the flag', where the objective would be to capture the flag object on the board and bring it back to predetermined space on the board. [7.] The objectives could also be objects that do not actively affect the game but which could be used to help the player to achieve victory. Examples of this could be health or repair kits on the board that could restore the players' HP or a power up that increases one of the player's attributes past its current limit for a limited time like speed or armor boost.

Game modes are usually predetermined rules for the match and are used to offer the players different kinds of gameplay experiences. Examples of classic game modes would be a deathmatch where the objective is to defeat as many enemies as possible either in a set time, the amount of lives or the number of kills in a free for all battle, or a team deathmatch which functions like a normal deathmatch but the players are divided into teams. There are also a lot of game modes that use objectives to vary the gameplay. Classic examples of this would be capturing the flag where the objective is to take possession of the flag somewhere on the game area and carry it to a pre-determined area which then rewards the player with a point [7], or playing king of the hill where players get points for the amount of time they spend alone inside a certain area. Games often also make their own game modes or give old ones a new twist. Examples of this are the halo franchises swat which is a deathmatch game mode but the players are restricted to only using certain weapons, and skull hunters where whenever a player is killed with a headshot, they would drop a skull. The objective of this game mode is to collect these skulls and deliver them to certain drop off areas on the game map. If the player is killed

while carrying skulls, they would drop them on the field for others to pick them up. During the testing phase the game will only support a game mode called deathmatch where two players fight against each other until one is defeated. However hopefully later on the line, more varied game modes could be added.

Example Loadouts

Loadouts are pre-selected sets of equipment that are either made by users or by the game developers before a game match [8]. They are usually used to make it easier for a player to switch or adjust their equipment quickly to match the situation. For example, in a combat-oriented game the player could have loadouts made a certain playstyle or an enemy type in mind. Loadouts are common in first person shooter (FPS) games like halo reach where the player can create their own game modes with premade loadouts of their choosing consisting of two weapons and an armor ability skill [9]. Battlefield: Bad Company 2, on the other hand, lets the player create a loadout for each class that is specific for that player, and it is used in most of the game modes but the amount of weapons and skills one can assign for each loadout is limited by the class and its level, so most of the weapons and skills are limited to be used in that specific class.

To showcase what kind of loadouts players could make in the game later in the development cycle, three example loadouts were created to show what kind of equipment players could choose while following one of the three different example playstyles. Each loadout is limited to only be able to carry a total of 10 weight units and each equipment shows signs of simple balancing and how it could be done in the final game.

Sniper

The example loadout sniper, as seen in table 2, has dedicated most of its weight capacity for a railgun weapon that allows it to cause a good amount of damage from a distance. To balance this weapon its ammunition count is limited to one, forcing the user to reload after each shot. This makes the player have to choose between moving or reloading the weapon after use, which would normally make it useless, but because the railgun has such a long range it can be used multiple times before the enemy can get too close to the user.

Table 2. Possible loadout for a pawn that plays according to the sniper architype.

Architype	**Loadout**	
Sniper	Name	Attributes
	long range railgun	damage (4-7) damage type (piercing) range (2-7) ammo (1/1) weight (6)
	molten iron magnum	damage (1-3) damage type (blunt, heat) range (0-2) ammo (6/6) weight (2)
	chain knife	damage (1-2) damage type (cutting) range (0-1) ammo (-/-) weight (1)
	light armor coating	armor (2) weight (1)

As a sidearm the sniper uses a magnum that does not weigh much but can be useful for finishing off an opponent who has gotten too close for the railgun, as its minimum range is two, meaning that it cannot hit an opponent that is in two tile radius of the user.

As a last line of defense, the sniper is equipped with a knife that weights one unit and does not cause much damage, but unlike the other two weapons the knife does not need to be reloaded after a certain amount of use, making it a reliable damage source when all other weapons are out of ammo.

Because the weapons take nine points out of the sniper's ten-point weight capacity, this limits the sniper's selection of armor, making it the weakest part of this loadout. Because of its dedication to damage output in the expense of defense, the pawn that would use this loadout could be seen falling into the architype of glass canon. This architype is known for having a massive damage output but being extremely easy to defeat by causing minimal damage. [10.]

Close Quarters Battle

The close quarters battle example loadout, or CQB, specializes in short range middle to high damage exchanges and its equipment reflects this fact, as can be seen in table 3 below. The CQB splits its weight capacity more evenly between its equipment. In addition, as the name implies, this loadout's weaponry is dedicated to close and personal combat, with its shotgun and sub machine gun (SMG) combination. The CQB's primary weapon, the shotgun is designed to deal a good amount of damage but it is limited by its short range. It also demonstrates a possible damage type, electric, that could have a small change, for example, to cause the enemy to stun and loose a part or the whole next turn.

The SMG, on the other hand, has the possible damage type, liquid, that could have the possibility to synergize with the shotgun by having a chance to increase the change of getting a stunned status condition from electric damage. It also demonstrates a risk and reward mechanic in its way of attack. As can be seen in table 3 below, the SMG's damage has been written as (0-2*5). This means that when used the SMG will do damage five times in a row and each time it can cause zero, one or two occurrences of damage to the target, meaning that in the worst-case scenario it can deal a total of zero occurrences of damage to the target, but in the best case, it can deal a total of ten occurrences of damage. In this case it could be stronger than the shotgun, which can only deal at best six occurrences of damage, but the shotgun is guaranteed to deal at least three if used in range.

Table 3. CQB architypes possible loadout.

Architype	**Loadout**	
CQB (Close Quarters Battle)	Name	Attribute
	Needle shock shotgun	damage (3-6) damage type (piercing, electric) range (1-3) ammo (8/8) weight (4)
	Water gel bull-pup sub machine gun	damage (0-2*5) damage type (blunt, liquid) range (0-2) ammo (5*5/25) weight (3)
	Medium thermal plate armor	armor (5, heat, cutting) weight (3)

The CQB uses an armor that protects a lot more than the previous example's armor. This is to allow the pawn that uses this loadout to be able to receive more damage and therefore last longer so it could either get close enough to deal damage or to just survive longer in an engagement.

The CQB's armor shows a possible simple way how armors could be balanced. As seen in table 3 above the sniper's armor gives two armor points for one weight unit. Compared to the CQB's armor which gives five armor points for three weight units, it would seem like the sniper's armor would be superior as it would give more armor points if scaled to needing the same amount of weight capacity. With this it would seem that the sniper's armor has set the standard for armors to be two armor points for each weight unit needed, but in the case of CQB's armor the missing armor point is replaced with more protection against heat and cutting damage types. In the game this could be used to balance the armors by taking away one armor point that protects from all damage types and it would be split to protect against two specific damage types with added effectiveness. For example, instead of negating one damage like a normal armor would do, a specific damage type armor could negate the damage from 1.25 to 2 more effectively.

Tank

The tank loadout is named after a classic architype, common in games. A tank dedicates as much as possible into anything providing protection from damage sources. This dedication to defense usually is to allow it to fulfill its main purpose which is to tank incoming damage, meaning that the tank will actively try to interfere with any damage that is meant for the tank's team members, therefore receiving damage meant for others. [11.]

Because the tank architype relies on others to deal damage to the enemy, this loadout would not be very practical in the game until it is updated to allow multiple pawns on the field.

Following the example of tank architypes, as can be seen in the loadout table 4 below, over half of this loadout's weight capacity is used for the armor, which has the most armor points compared to armors found in the other loadouts.

Table 4. Possible equipment loadout for a tank architype.

Architype	**Loadout**	
Tank	Name	Attribute
	Oscillating great plasma sword	damage (3-5) damage type (cutting, heat) range (0-2) ammo (-/-) weight (4)
	The big boy fortress plate armor	armor (12) weight (6)

The remaining four weight capacity is used for the only weapon this loadout has, a great sword that shows that even melee weapons can cause a good amount of damage from a distance [12]. However, depending on how the movements are handled in the game, the great sword can become useless against weapons that have better range, which could easily lead into two negative outcomes. Either the pawn using the tank loadout becomes an easy target that cannot fight back or the match will turn into a stalemate where the tank pawn cannot reach the opponent and the opponent cannot get through the tank's armor, making both sides unable to win.

Adding Features and Discussion

During the initial phases the game will only have three different weapons and armor types. Due to lack of variety in pre-match choices the game will most likely become similar to a game of rock paper scissors, as only the precombat choices will have any real effect in winning the match. For example, if the players can pick whichever weapon and armor in the game, they would most likely pick the strongest of each which would end in a symmetrical gameplay situation that will cause the game to go on until one of the players makes an obvious mistake. Each player would try to avoid each other's attacks while trying to successfully make an attack which in theory would be okay, but in this situation, it could become an endless game of cat and mouse. Another example on how the balancing would end up in a stalemate would be to give a numerical value to each weapon and not allowing the players to have a combined value over a certain amount (for example the weakest weapon would have a value of one and the strongest three, and the allowed maximum would be four). By adding this one value the strategically possible combinations increase from one to three (weak weapon heavy armor, heavy weapon weak armor or medium weapon medium armor), making this a literal rock paper scissors situation, but the problem here is that if the opponent does not make an obvious mistake during the match, then the outcome of the entire match is decided while selecting the weapons and armor. There are two different ways to increase the amount of strategy a player can use during a match. The first way is to add more attributes to the game, like accuracy to the weapons or weight limit to the equipment. This helps to differentiate the choices from each other and can make sure that each option has its strengths and weaknesses in each situation. The second one is to add more options to pick from before the match. This makes the pre-match choices harder to predict and encourages the players to give more thought on what to choose.

These two options also affect the complexity of the game and balance each other out in a sense. While just adding more weapons and equipment may give the player more options in itself it also will end up devolving the game back into rock paper scissors (which feeds the gameplay mentality of trying to one-up each other endlessly) if not restricted by anything. This restriction can come in the form of new attributes like weight which would limit the amount and combination of equipment that could be taken into the match, therefore making it less predictable which equipment's the opposing player chose. Attributes could also be used to allow certain type of synergies to form between different choices.

Adding attributes and new equipment methods also complement each other well in the sense that the increased number of attributes gives more variety in equipment and thus allows for both unique and viable parts of strategy or loadout. For example, a game where the equipment has only one or two attributes, but hundreds of different kinds of unique equipment, would have to have one or two pieces of equipment that are objectively better in the game. Therefore, they would end up being as good as the only equipment, as every player typically wants to win. This means they will most likely choose the best equipment, as will everyone else, therefore evening the playing field. On the other hand, if there were only five or ten pieces of equipment and hundreds of attributes, the equipment would just end up filling in the generic equipment roles (almost every first person shooter has to have the pistol, the shotgun, the assault rifle and the over powered super gun to the kin of a rocket launcher) with little to no overlapping or motivation to try and break the most common playstyles.

It would be preferable if a feature could be added that would bring the players back to the game and give them a reward for investing their time. The most common example in modern games, of this kind of feature, are the different kinds of unlockable content that can be accessed when a player reaches a certain goal. The unlockable content varies from usable game items, cosmetic items or even achievements (e.g. virtual medals that show that the player has achieved something in the game, for example beating the game on the hardest difficulty level).

Another way to get players to return to the game is to get them emotionally attached, for example having an engaging story where the player connects with the characters. These connections will bring them back to see how their favorites will end up or even just to spend more time with them. (For example, this is very common in the dating simulation and visual novel genre.) Another example would be to give the player a customizable character that will grow and develop during the game with the player. [13.] An example of this is in the game X-Com Enemy Unknown in which the player is given three random soldiers. It is not uncommon to get attached to these virtual characters while the player watches over them. Through each fight, the player decides their skills and equipment and maybe even sees them fall in combat.

However, one of the most effective ways to add replay ability to the game while also increasing its entertainment value is to add micro-randomizations to certain aspects of the game. These randomizations affect minor aspects of the constant gameplay such as enemies dropping more valuable items in rare instances or weapons not having a set

constant damage output, but instead an array of possible amount of damage between two points. [14.] This guarantees a bit of excitement whenever the weapon is used as it does not always end up as one would assume. However, there is still a fine line between small random variables that make the game more enjoyable, as it is still somewhat expectable, and randomizing things to the point that it becomes too unpredictable after which the player can start to feel cheated and discouraged from playing again.

There was also discussion of adding features that would make it easier to find opponents for a match in situations where finding an additional player locally is difficult. The more coding and testing intensive route could be to add an AI opponent to the game. This AI could act as a training partner for the player or just a way to play the game even alone, but for the AI to be any use for the player requires that it would be complex enough to offer a challenge. Otherwise, nothing will be learned and no fun will be had.

Adding an online multiplayer mode could also be a less coding intensive way to find an opponent. The turn-based nature of the game will make it easier to add the capability to play online, as there is no real reason to try to synchronize each player experience in real-time. This also opens the door to an online tournament where the two contestants connect to the same gameboard in a third-party location that is then streamed to the viewers.

Development Cycle

In theory the game will go through three stages while in development. The first stage would be described as barebones, because during this stage the game contains just the bare minimum of gameplay elements such as hit points, movements, menus, and methods of inflicting damage to opponent.

The second stage will be called a “tofu” stage. The name comes from the idea that during this stage there are a lot of more specific gameplay elements such as possible customization options, game modes, and character attributes. However, everything will be named simply and without any kind of lore or setting in mind. Therefore, the pawns will be considered to be just square colorless blobs with no identifying characteristics that would make them seem like robots, tanks or something else. During this stage there will also be a lot of testing and tweaking on the main gameplay formula like how many pawns will be used by a single player, what parts of the game are fun and what could be improved etc.

Lastly the final stage is simply called the game stage. At this point the final touches will be added. This includes lore, new options for the players, new features like maps and modes etc. During this stage the idea of what the robot pawns represent will be finalized.

Gameplay Considerations

There is a lot to consider while designing a game. For example, what will make it fun, what the objective is, and what will bring people back. One worry at this stage of the development considering the turn-based slow tempo of the game is would it be fun to only have one pawn for each player. [6.] This is because while going through well-known turn-based games, no one unit game comes to mind. Games like X-Com Enemy Unknown starts the player being able to control a squad of three units and games like Civilization 5 start the player with two units. However, in Civilization the player will most likely sacrifice one of those units to build a city in the first few turns of the game leaving them with just one. Before you can build another unit, the only thing the player can do is to reveal more of the map using this lonesome unit and set a city development and research project on the first turn and then wait for around 4-10 turns for your possible other unit to be made. These first turns are quite boring in the sense that the only thing you can do is move your unit a few tiles and then pass the turn over.

Because the unit's actions will be the main way of interacting with the game, there is a chance that the gameplay will be too slow paced or uneventful with just two pawns on the field. Three ways are taken into consideration on how to prevent this problem. The first one is to simply add more pawns onto the board, but this will cause problems like having to sync all the robots with the computer at the same time, and it is important to ask what will happen if one of the robots is blocking a chokepoint on the map. The second way would be to change the gameplay from turn-based into real-time, but this could cause the game to become disorganized, as keeping track of the robot's movements could prove challenging. Furthermore, the third option is just to add other gameplay elements to spice up the gameplay. An example of this would be Civilization 5 where moving the pawns is just a part of the game. There are also diplomacy options and decisions on what upgrades to research and build. Also, X-com breaks the gameplay by centering the whole game inside a command base where the player can build and research new upgrades for the units.

4 Methods and Materials

The physical components of the project consist of a Zumo robot, a printed paper field and a computer as the controller. The Zumo robot uses many different kinds of methods and concepts to work.

Zumo

Zumo is a robotic base for the Arduino development board, which has been fitted with a custom shield developed at Metropolia University of Applied Sciences, which allows it to be used with the PSoC development board in this project.

The Zumo harbors inside multiple different sensors and a pair of motors that can be used with the development board attached to it like seen in the block diagram, Figure 3, below. These sensors include a row of reflectance sensors, a gyroscope and an accelerometer. In addition, there is an added pair of ultrasonic sensors added to the Zumo bases that can be used for the pawn robots. [1.]

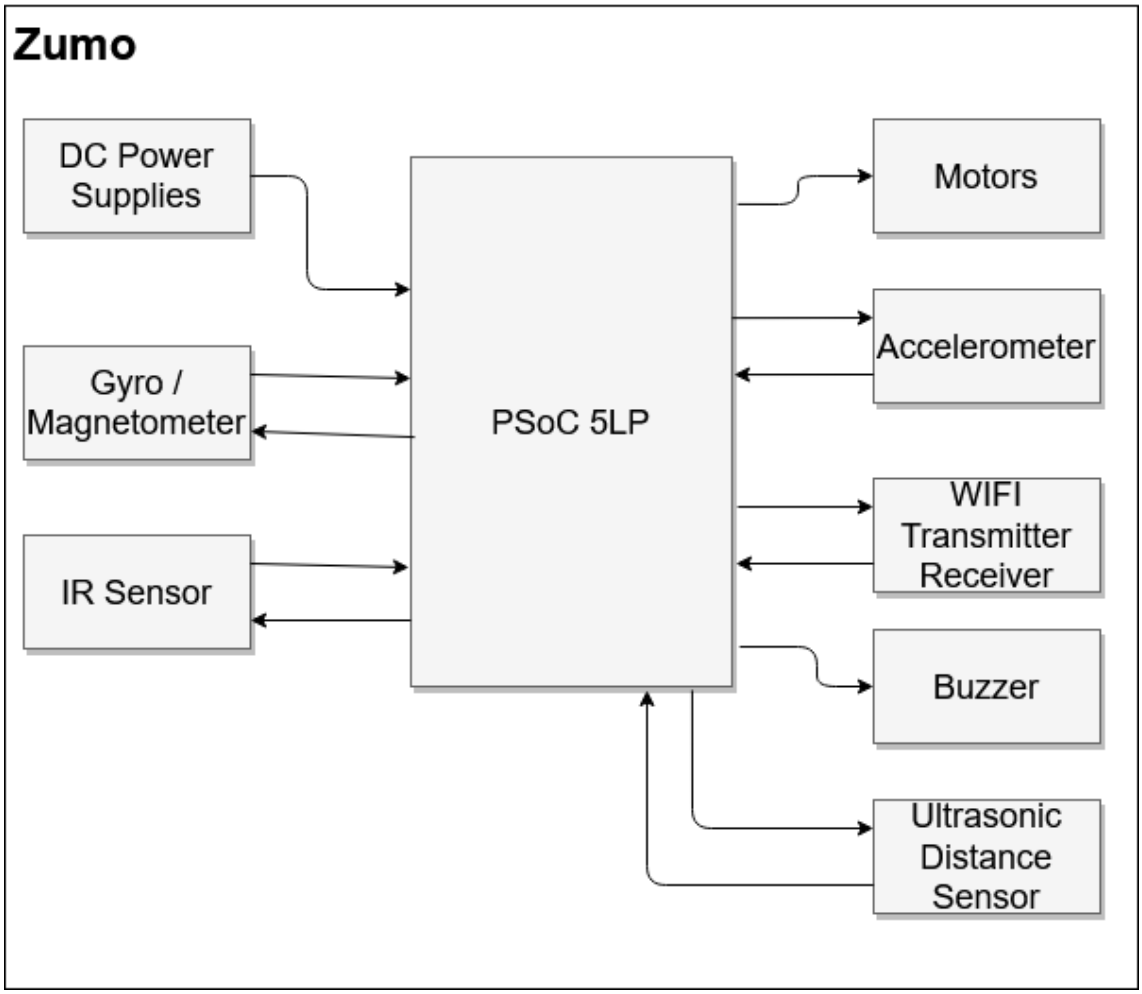

Figure 3. Block diagram showcasing the interactions between the PSoC 5LP development board and the Zumo.

For now, only the Zumo's treads and reflectance sensors were used for the movements and line following, but there could be some use for the remaining sensors. The gyroscope could be used to aid in the navigation of the gameboard by confirming which way the Zumo is pointing at and helping it to make more accurate turns in corners and intersections. The ultrasonic sensor could be used to detect obstacles and the distance from them, if real life representations of the game world's obstacle buildings, for example, were to be added to the gameboard. This could also help to confirm the location of the pawns on the field by referencing the digital game map to see if the obstacle it detects should be there or not, but in the end, this would mostly be adding a feature to the robot

just for the sake of using the ultrasonic sensors. And to add a little bit flare to the performance, it would be entertaining to use the Zumo's build in a buzzer to create some rudimentary sounds that could be played during proper circumstances for sound effects.

IR Sensor

IRs or infrared sensors are components used for multiple purposes such as temperature and distance measurement, remote control or, in the case of the project, line following. These sensors measure incoming electromagnetic radiation in the infrared wavelength and can fall into two categories, passive and active. The difference between the two is that passive sensors only wait and measure the radiation around them while active sensors are usually paired with IR transmitters that expel radiation which is then deflected by the target to the IR sensor, as can be seen in Figure 4. [3.]

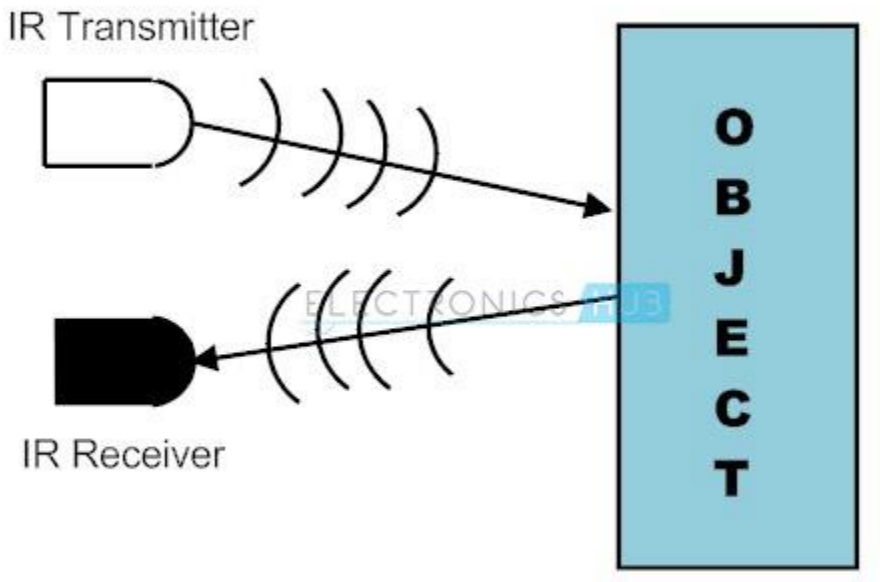

Figure 4. Representation of how active IR sensors operate by bouncing radiation from the target to the sensor. Copied from [3.]

The Zumo uses a line of six active IR sensors to follow the line by comparing the readings from each sensors and possible deviations between them [1; 2], as the dark color of the line absorbs more of the sent radiation than the white of the background, meaning that

the more an active sensor is on top of the line the smaller the reading is that it receives from the deflection. [3.]

H-bridge

An H-bridge is a circuit that allows the switching of current directions for whatever is connected to it. Inside the Zumo they are used to switch the direction of the motors on the treads. The H-bridge works by having four switches usually depicted in sets of two, where one set has two switches connected to each other while the component connected to the bridge is connected between these two sets as seen in Figure 5 below. In the picture the positive side switches are called A1 and B1, and the negative side switches are A2 and B2.

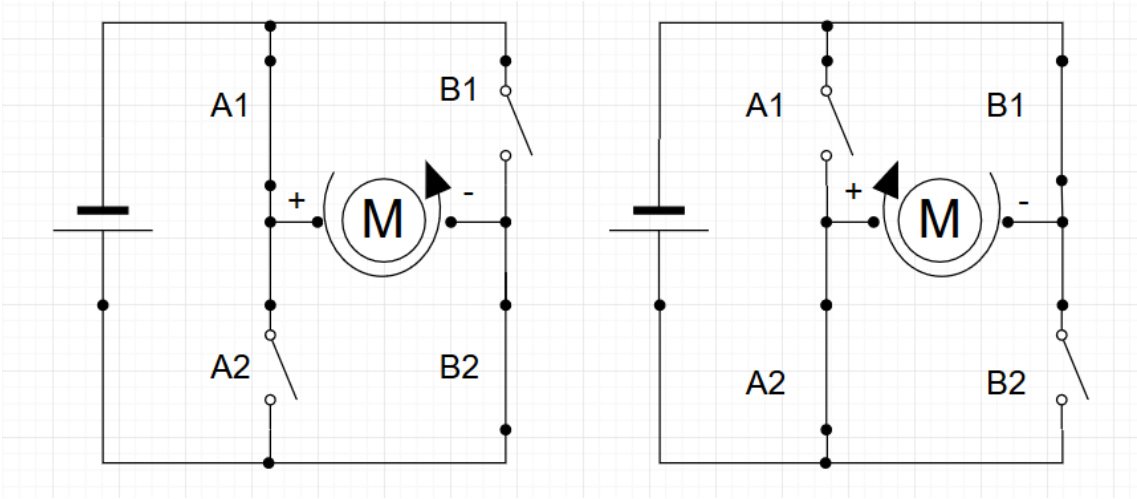

Figure 5. A simplistic H-bridge connected to an electric motor.

As can be observed in Figure 5, by closing the specific switches while keeping the two others open, it is possible to affect how the electricity flows into the motor, thereby changing its rotatory direction. [15; 16.] The closed switches are always diagonal because the closure of both switches in a set will cause a short-circuit, which can cause damage to the electronic components. [17.]

Because it is impractical and too slow to have someone manually flip the switches to make the motor reverse its rotation direction, in the real thing all the switches will be replaced with transistors that will disconnect the circuit when little to no voltage (below 0.6) is applied to them and completes it when the voltage passes a certain threshold.

[18.] This makes it possible to change the direction of the motor by using a microcontroller to apply the voltage to an H-bridge circuit or even to create a H-bridge inside the microcontroller itself digitally.

PWM (Pulse Width Modulation)

Pulse width modulation is a method that is used to limit a component's power consumption. It works by rapidly switching the power on and off, creating a graph that looks similar to what can be seen in Figure 6 below. The amount of power going through is determined by the collective amount of time the power is allowed to go through. The amount of power going through is usually measured in cycle percentage, where a cycle is the part of the PWM signal that gets looped over and over, and the percentage is the amount of time the power is on compared to the time when its off. For example, 01010101 would have a cycle of 50% and 01100000 would have 25%. [19.]

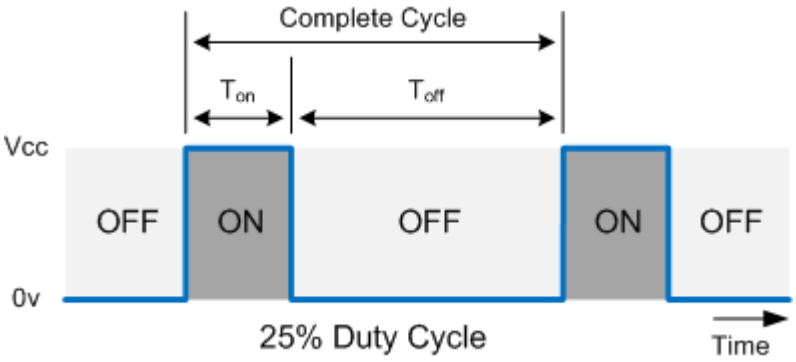

Figure 6. A sample of different parts of a PWM wave. Copied from [20.]

PWM pulses are measured in the millisecond (ms) scale and the frequency of the signal can be calculated by dividing one by the signal's period. A PWM signal's period is the time it takes for the signal to start repeating itself, usually measured from peak to peak or drop to drop. For example, if the period is 10 ms long, the frequency would be 100 Hz as it repeats once every 10ms (1/10ms=0,1). The duty cycle of a PWM signal is calculated by dividing the pulse width with the period and then multiplied with 100%. A pulse width is the time that the pulse stays high before returning back down. For example, in the Figure 6 above the pulse stays high for one millisecond. In this case, the duty cycle would be calculated as follows: (pulse width / period) * 100%. This would result in a duty cycle of 10% as the pulse remains high for a tenth of the period's length.

MQTT

MQTT is a light communication protocol used to send messages between the Zumo and the computer using a Wi-Fi connection. It uses a system where an author creates data that is then sent to a distributor, which then distributes it to the subscribers that have requested the distributed data. [21.]

As seen in Figure 6 below, the game inside the computer will be the author in this case. The MQTT broker program will work as the distributor sending the data forward to the Zumo which is the subscriber that will then receive and use data sent from the game.

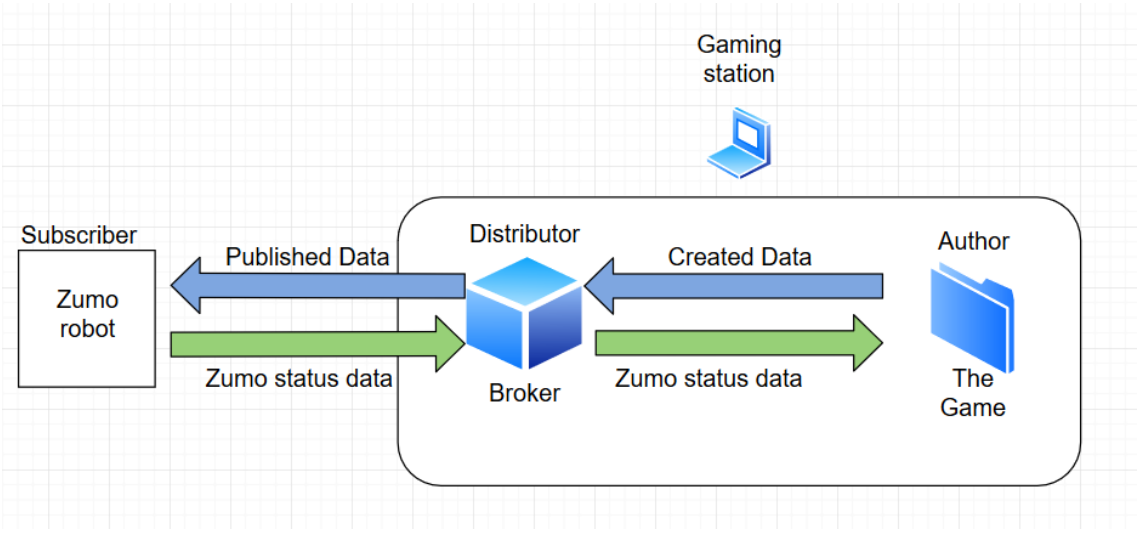

Figure 7. How the MQTT is used to connect different parts of the project.

But like in the real world, this author subscriber relationship does not have to be one sided, and in this case, it is highly useful to have the Zumo be able to send data back to the game on multiple occasions. [21.]

PID

A proportional–integral–derivative (PID) is a reactive algorithm controller that is commonly used with line following robots. PID is used to correct the unavoidable drifting that is caused by the small differences in motors speeds. This speed difference is present even within motors of the same model and manufacturer which can be caused by normal wear and tear or the fact that no product is perfect. To counter this drifting, the PID algorithm checks if either of the central reflection sensors are away from the line and how long it has been since the last check. If one of the sensors is not on the line, then the algorithm calculates how strongly it has to correct the robot's course by altering the motor speeds. This alteration in speed is dependent on how far away from the line the sensor is and how quickly this change occurred. For example, a course correcting turn for a small deviation is a lot softer than for a drastic one in the same time period. This is to decrease the possibility of losing the guideline between checks.

Each part of the controller can be used alone as a simpler program or combined with another part to create a more complicated and precise closed loop controller as can be seen in Figure 8 below.

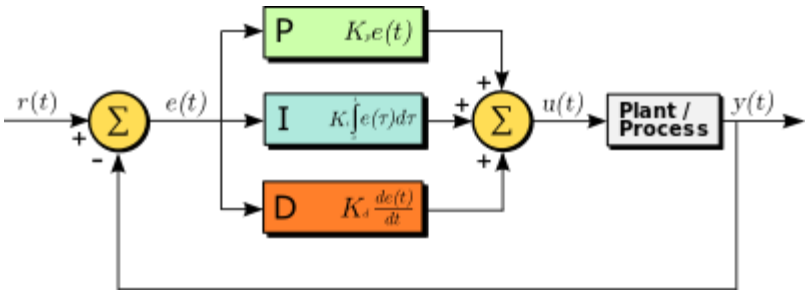

Figure 8. Visualized PID controllers' formulas working together. Copied from [22.]

Like the term closed loop implies, the controller loops around while constantly trying to refine the modifier in an effort to minimize the current error.

The P or proportional controller as its name implies tries to get rid of the error by trying to correct it by responding proportionally. This can be seen as a wave graph where the Zumo's deviation from the line can be seen as a positive or negative value depending on which side of the line the Zumo is deviating to. The P controller will react to the line's

amplitude. The larger the amplitude is, the harder the P controller will try to correct it. [22.] Depending on the frequency of the readings and the speed of the change, the corrective actions of the P controller can often result in an osculating error line as it tends to only take into account the current situation [23], therefore often overshooting with its corrections.

The I or Integral controller, on the other hand, reacts to the graph error line's surface area and focuses on fixing the error by adding to the controller modifier in small steps [5]. Depending on which side has accumulated more error, the I controller will try to correct to the opposite direction by a pre-determined amount and will slowly arrive around the zero-error mark with minimal oscillation [22; 23]. However, compared to the P controller, this process will take a lot of time and the Zumo would spend more time on one of the wrong sides of the line.

The PI controller is one of the most common combinations of these three controller types and it combines the P controller's speed to the I controller's accuracy by using the I controller as a limiter for the P controller, reducing the P controllers' oscillation. A real-life example of this could be a ball that goes back and forth inside a bowl, where the P controller would be the ball and the I controller would be the angle of the bowl's curvature. At the start the bowl's curvature is insignificant and the bowl can effortlessly roll back and forth across the middle line. However, as the curvature of the bowl becomes steeper, the ball will start to lose more and more of its momentum, shortening the distance it travels from the center line after each passing. This continues until the ball loses all of its momentum and ends up just sitting in the center of the bowl.

The D or derivative controller tries to predict the next error reading by taking into account the previous and current errors [22]. This prediction is then used to adjust the modifier in an effort to try preventing over shooting the zero line while still being able to make more significant advances towards the zero line.

Cypress PSoC Programmable System on Chip

The Psoc is a prototyping microcontroller board that is very flexible, easy to use and very power efficient making it a good choice for embedded systems just like the zumo robots used as the pawns on the gameboard.

The specific board in use is PSoC 5lp which houses an ARM cortex-M3 microcontroller and a universal digital block (UDB) that can be configured to create different kinds of digital and analog peripherals and functions such as PWM and I2C. These blocks can even be combined to create a more procession heavy 16, 24 or 32-bit versions of the functions. [24S5.] The UDB also allows the changing of output pin properties allowing them to be configured to work for example as PWM pins if needed. This makes the PSoC board very flexible in the sense that it allows users to take away or convert more pins into important tasks like motor control if needed.

5 Implementation

A large amount of the project revolves around the game's menu system as it is the only part that allows a two-way communication between the project and the user. Therefore, a lot of the implementation time was spent on making sure it works properly.

Game

The whole game program runs through a menu function that gives the user input options, which return values that determine which subfunction needs to be activated. For example, at the start of the program it would display the user a starting menu that could follow the flowchart seen in Figure 9 below, by giving the user three options to choose from. These could be to start a new game session, create a new loadout or to simply exit the game.

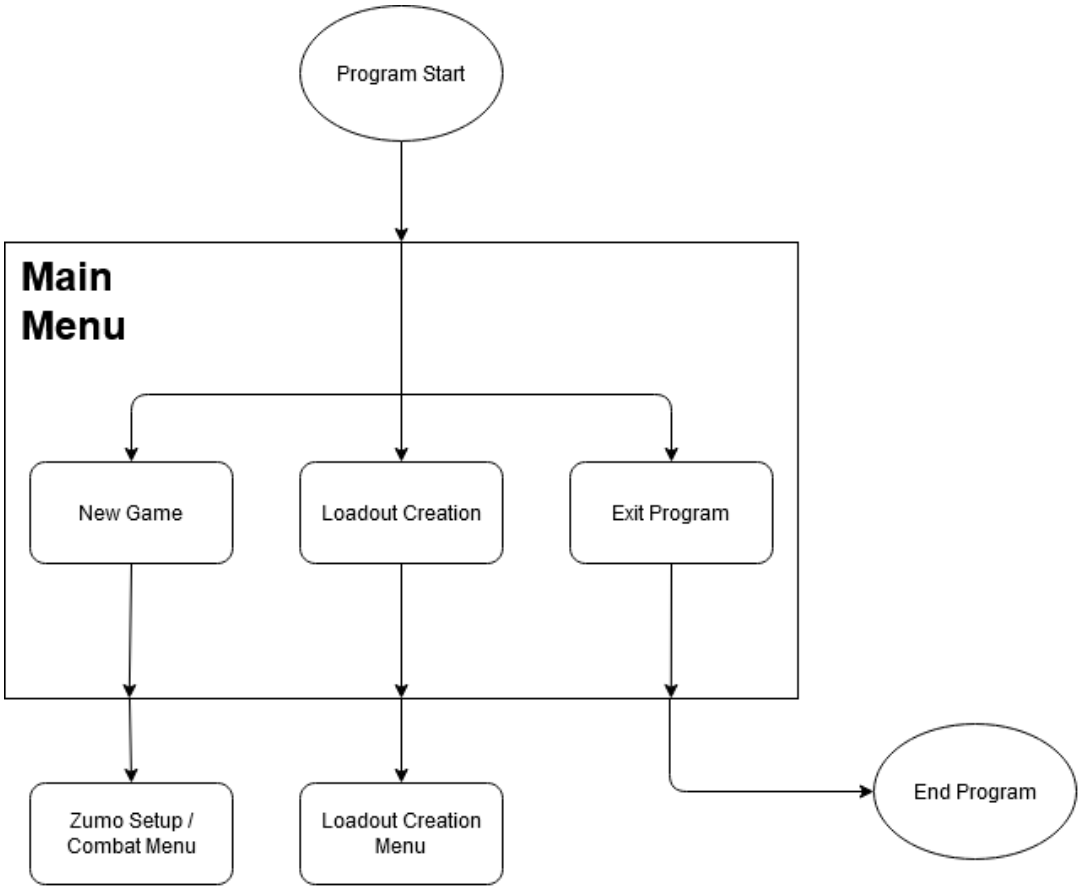

Figure 9. Main menu flowchart.

If the start of the game session was selected, the program would initiate the Zumo setup by asking the user to set up the Zumos as seen in the flowchart, Figure 10, below. After this the menu function would go down the path to a point where the user has to select a pre-made loadout either found in the game or made by a user from the start up menu. The loadouts will then be copied to a list of participants for the session and each loadout will be assigned a Zumo to act as a physical representative on the gameboard.

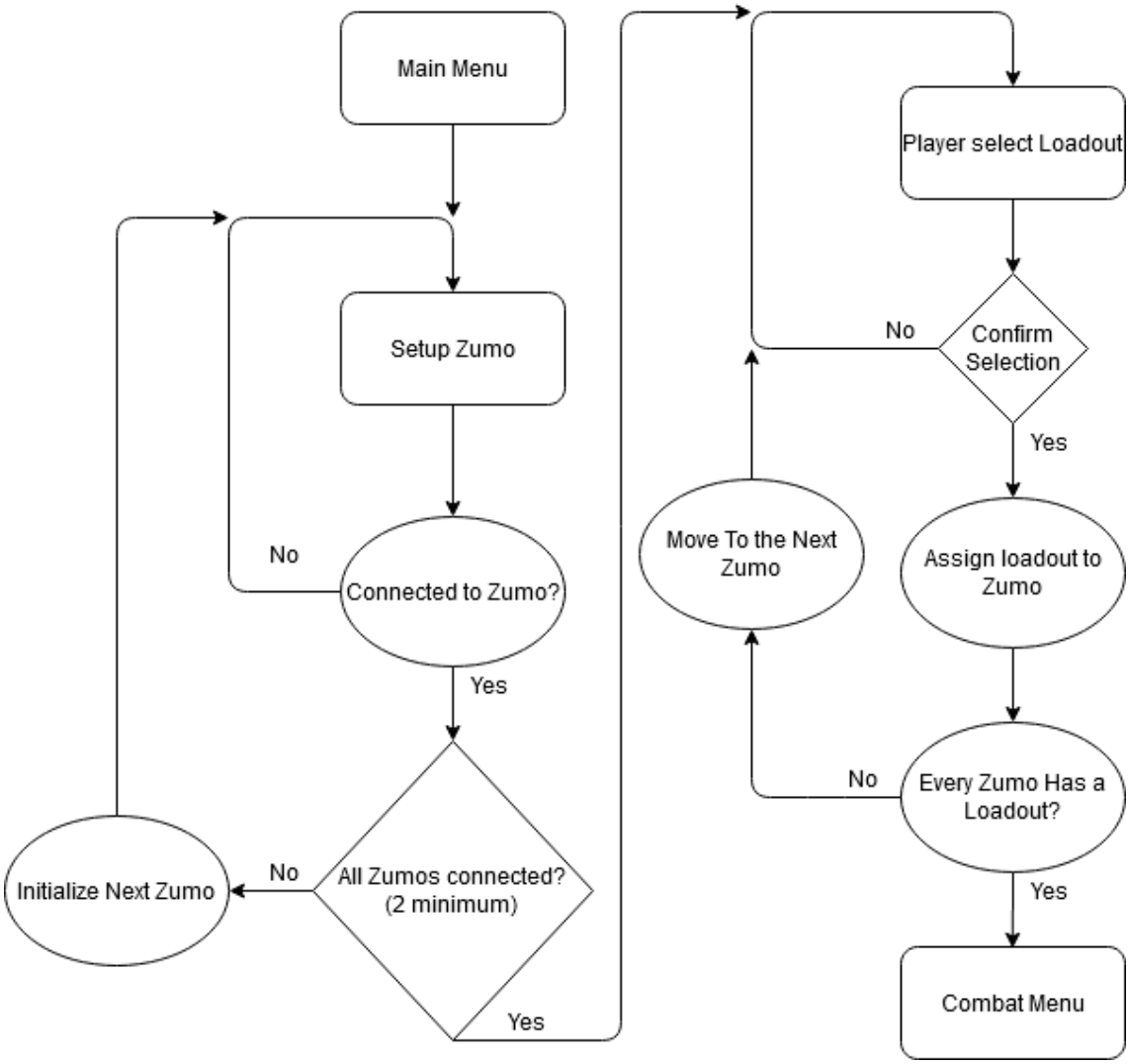

Figure 10. Flowchart of the Zumos setup between main menu and combat menu.

The menu function will then move to the actual game loop where one at a time each user is asked to decide a set of actions for the in-game character and the Zumo to complete.

A typical user turn could follow the flowchart in Figure 11 below where it starts by the program asking whether the user wants to move or do an action. If movement is selected, the menu function will go into the movement section, where the user can plot out where the unit should move to. If this action is confirmed, the menu will then send the instructions to the communication function. After this, the menu function would loop back to the previous section, set the turn to be over and go back to the start of the game loop, give the role of active player to the next player in line and then start all over again.

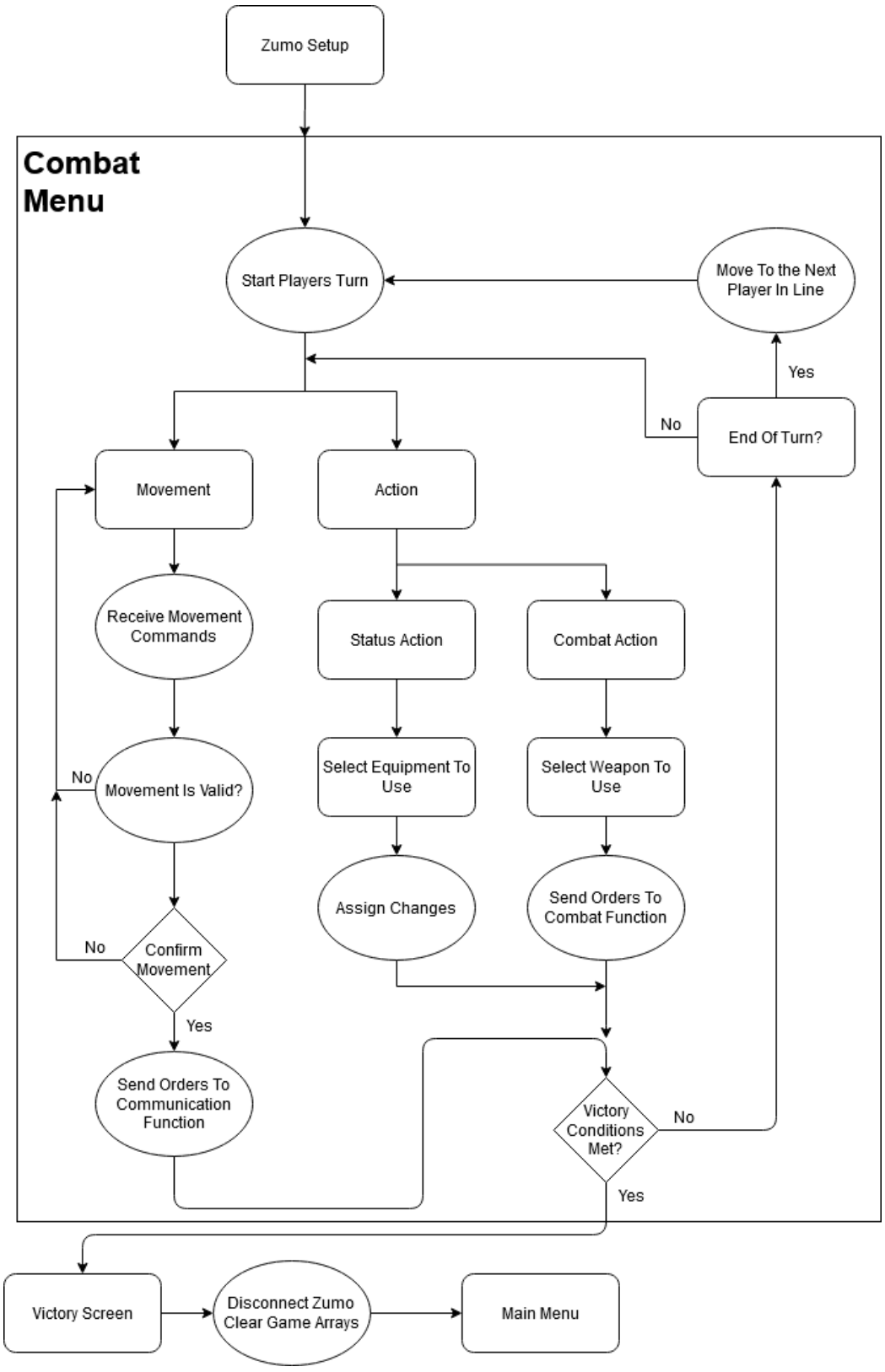

Figure 11. Flowchart of the basic idea behind how the combat menu system will be handled inside the game.

If the “action” option is selected, the menu function will go to the action section of the function and display a menu screen with the possible actions that can be performed. The actions that can be selected can either inflict modifiers to one or both of the pawns, such as lowering or rising the unit’s attributes (such as HP, armor and speed), add modifiers to the field for example in the form of mines that could cause damage to a unit entering the affected tile, or initiate an attack which will then call for the combat function.

This game menu loop will only end if one of the conditions is met, such as a pawn’s HP going equal or under zero, one of the players forfeiting the match or a special objective being completed depended on the selected game mode.

The combat function’s job is to calculate and enact the result of the battle fought between the combatants after each engagement. When one of the players declares an attack, the combat function is called. The function needs the involved parties’ array index number as parameters, so it can compare their equipment and other variables to calculate how much damage will be deducted from the defender’s hit points with hard math and random number generation.

The communication function works independently from the rest of the program because it needs to send ping messages to the Zumo robots to make sure the connection won’t go into sleep mode. After receiving the ping message, the Zumo will reply with a pong, which ensures that the robot is still connected to the network. Other parts of the game program can activate the communication function by giving it a char command string, which it will then send to the Zumos instead of the periodic ping message as any message is sufficient in keeping the line awake.

Zumo

As the Zumo will work as a puppet for the game, most of its functionality will revolve around receiving commands and following them and the line accordingly. Because of this, when first started, it will go into a standby mode, waiting for a new gaming session to be started, which will prompt the game to send initialization data to the Zumo. After receiving the data and confirming with the game that everything is ready, the Zumo will start executing its programing as can be seen in Figure 12 below. As seen in Figure 12, after receiving the initialization information the Zumo will start to wait for its turn while periodically sending a ping to the game in an effort to keep the connection alive. When the Zumo receives a message indicating its turn has started followed by instructions, it will then either start following the line in front of it or turn according to the instructions and then follow the line depending on if the coordinate to move to is in front of the Zumo or not.

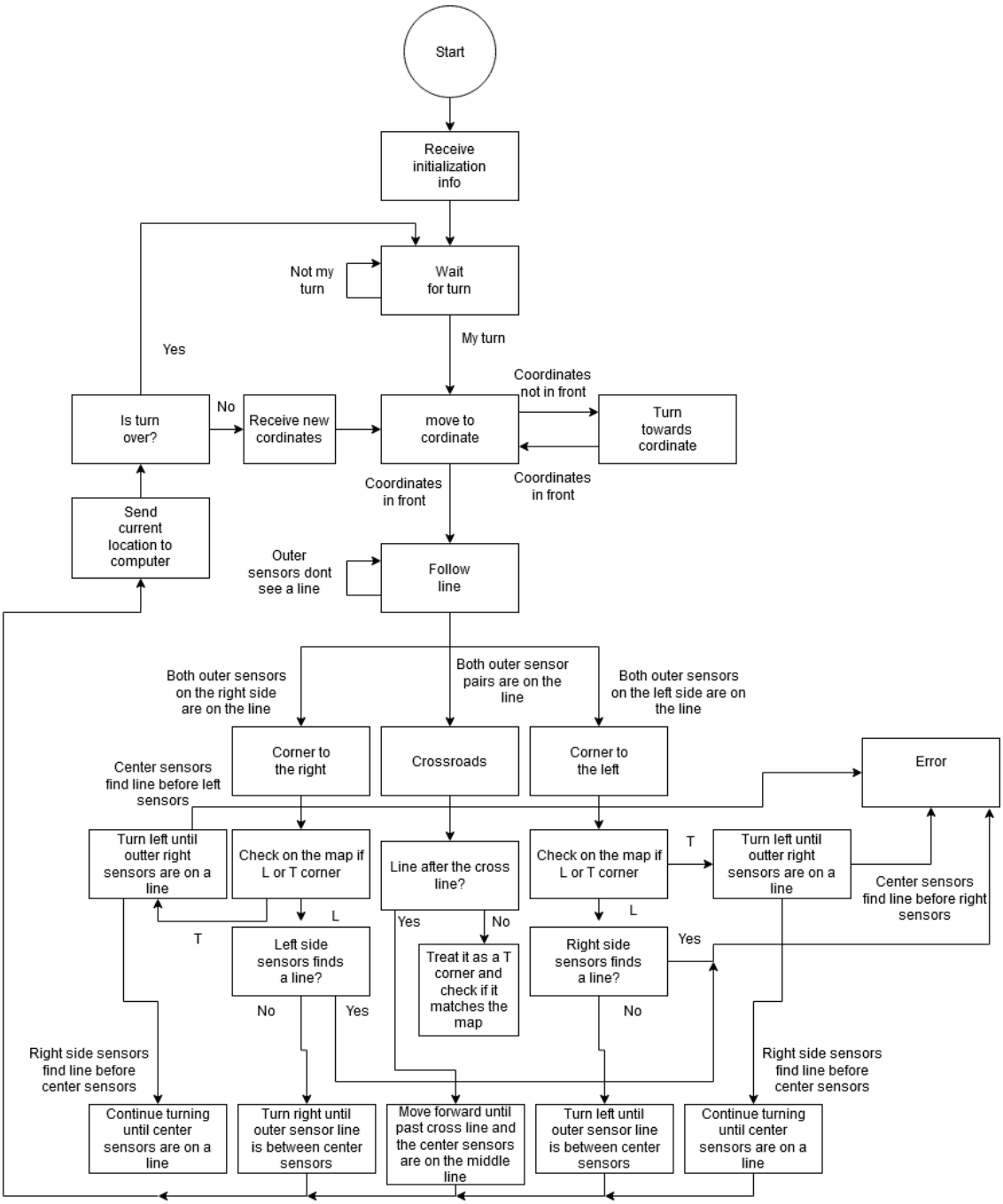

Figure 12. Simplistic logic flowchart of how the Zumo will work on the gameboard.

While the Zumo is following the line, it can determine what kind of a section is in front of it, depending on which sensors give a reading and whether they imply there to be a line or not. Three examples of this can be seen in Figure 12 above, which are when the middle and left sensors give a line reading indicating that there is a corner turning to the left in front of the Zumo. This can also mean that the corner is part of a T cross on the

right side of the gameboard from the Zumo's point of view. If the Zumo got the readings from its center- and right-side IR sensors, then it would indicate a corner in the line turning to the right, which would imply a mirrored situation in comparison to the left corner. The third option is when all the sensors give a line reading indicating there to be a crossroad in front of the Zumo. This could also be another T cross going from the center of the gameboard towards the outer lines, if there is no center line after going through the horizontal line.

After arriving to the corner or inter-section of the line, the Zumo will check if it matches the expected section of the gameboard on the initialization data. If the section does not match the initialization data, the Zumo will stop and send an error message to the computer as it has lost its way at some point of its travel.

If the section matches the data, the Zumo will inform the computer of its arrival and go into waiting until the next step of the instructions arrive, after which it will loop back to the start of the Zumo's turn while checking for inconsistencies with the data, until the computer informs that the turn has ended, after which the Zumo will go into waiting until its next turn or the end of the game.

6 Conclusion

As the project's main goal was to create the next evolutionary step for the now almost extinct VHS and DVD boardgame genre. The primary focus was to create an easy to replicate system that could be used as a base for multiple different kinds of games in the future.

To achieve this a lot of concepts studied at Metropolia University of Applied Sciences were combined, such as controlling the amount of electricity going to a robot's motors with pulse width modulation, and using an active infrared sensor to monitor guide lines found on the gameboard. By combining these beginner level concepts with technology found at the campus, such as the Zumo robot platform and the PSoC 5lp development board, it was easy to create a line following robot to be used as the pawns.

Then adding a Wi-Fi module to this easy to use robot allowed the usage of a light weight messaging protocol called MQTT, which would allow the communication between the robot and a custom video game running on a computer.

With this the hardware side was completed, which was followed with a simple game program that can show how the boardgame and video game could interact with each other.

Sadly, during the project, a major setback in the form of closing and demolition of the Leppävaara campus, followed by the outbreak of the Covid-19 pandemic, prevented the use of dedicated development space and resources. This forced the scope of the whole project to be scaled down from a full practical project, to a fully theoretical work instead.

To compensate the lack of practical work, time was used to hone the game side by adding possible ways to improve it in the future and discussing how these changes could impact the experience.

This time was also used to take a closer look at the individual aspects of the hardware side and by doing so learning some simple but valuable things, such as the benefit of not wasting time on implementing and tuning a PID controller when a simpler P controller would work just as well, and how the PSoC uses universal digital blocks to create digital logic gates and functions.

As the project is a victim of circumstances and inadequate time management, it would not be a stretch to say that a lot can be improved, be it by finding a suitable place outside of campus for testing at least some of the hardware, or creating a small demo of the game even if there was not a safe and practical way to connect it to the Zumo.

However, even though the project did not allow the demonstration of skills learned while studying, it heavily encouraged to focus on theoretical matters, such as finding more about individual subjects. Also, the lack of a physical component forced the project to be looked at from different perspectives, in an effort to foresee possible problems that the project could face in the future. Concluding in an intense session of figuring out new ways to possibly improve the project, while also finding ways to prevent possible problems caused by the additions.

Taking all of this into account, the project is a failure when compared to the original intentions. Moreover, the whole process went worse than what was hoped for. However, it was an educational experience which allowed for more diverse learning situations than what was expected. Because of this the project can be seen as an eye-opening learning experience. The project can also be seen as an educational tool to show that even though failure is always near, there are still things to be salvaged and learned from it. Therefore, the project is a failure in comparison to the original goals, but an adequate learning experience like it was always intended to be.

References

1 Pololu, (2019). Pololu Zumo 32U4 Robot User's Guide. [Online] Available: https://www.pololu.com/docs/pdf/0J63/zumo_32u4_robot.pdf [Accessed 9 October. 2020].

2 Pololu, (2019). Pololu Zumo Shield for Arduino User's Guide. [Online] Available: https://www.pololu.com/docs/pdf/0J57/zumo_shield_for_arduino.pdf [Accessed 9 October. 2020].

3 Electronics Hub, (2015). IR SENSOR. [Online] Available: https://www.electronicshub.org/ir-sensor/ [Accessed 30 July. 2020].

4 Computer Hope, (2019). What Is HP?. [Online] Available: https://www.computerhope.com/jargon/h/hp.htm [Accessed 9 October. 2020].

5 Macgregor Jody, (2018). The History of Hit Points. [Online] Available: https://www.pcgamer.com/the-history-of-hit-points/ [Accessed 14 October. 2020].

6 Johnson Soren, (2009). Analysis: Turn-Based Versus Real-Time. [Online] Available: /view/news/116864/Analysis_TurnBased_Versus_RealTime.php [Accessed 9 October. 2020].

7 Computer Hope, (2018). What Is CTF (Capture the Flag)?. [Online] Available: https://www.computerhope.com/jargon/c/ctf.htm [Accessed 9 October. 2020].

8 Wiktionary.org, (2020). Loadout. [Online] Available: https://en.wiktionary.org/wiki/loadout#English [Accessed 9 October. 2020].

9 Technopedia, (2011). What Is First Person Shooter (FPS)?. [Online] Available: http://www.techopedia.com/definition/241/first-person-shooter-fps [Accessed 14 October. 2020].

10 Wiktionary.org, (2020). Glass Cannon. [Online] Available: https://en.wiktionary.org/wiki/glass_cannon [Accessed 14 October. 2020].

11 Computer Hope, (2019). What Is a Tank?. [Online] Available: https://www.computerhope.com/jargon/t/tank.htm [Accessed 9 October. 2020].

12 Computer Hope, (2019). What Is Melee?. [Online] Available: https://www.computerhope.com/jargon/m/melee.htm [Accessed 9 October. 2020].

13 Teng Ching-I, (2010). Customization, Immersion Satisfaction, and Online Gamer Loyalty. Computers in Human Behavior. 2010 November 1;26(6):1547–54.

14 Bycer Josh, (2018). A Study into Replayability -- Defining Variance. [Online] Available: https://www.gamasutra.com/blogs/JoshBycer/20180521/318337/A_Study_Into_Replayability__Defining_Variance.php [Accessed 28 July. 2020].

15 Northwestern Mechatronics Wiki, (2009). Driving a High Current DC Motor Using an H-bridge. [Online] Available: http://hades.mech.northwestern.edu/index.php/Driving_a_high_current_DC_Motor_using_an_H-bridge [Accessed 4 September. 2020].

16 Loflin Lewis, (2018). H-Bridge Motor Control Using Power MOSFETS. [Online] Available: http://www.bristolwatch.com/ele/h_bridge.htm [Accessed 4 September. 2020].

17 Modular Circuits, (2011). H-Bridges – the Basics. [Online] Available: https://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/ [Accessed 10 March. 2020].

18 learn.sparkfun.com, (2014). Transistors. [Online] Available: https://learn.sparkfun.com/tutorials/transistors/applications-i-switches [Accessed 27 February. 2020].

19 Kompulsa. Introduction To PWM: How Pulse Width Modulation Works. [Online] Available: https://www.kompulsa.com/introduction-pwm-pulse-width-modulation-works/ [Accessed 14 October. 2020].

20 OnionDocs. Using the PWM Expansion. [Online] Available: https://docs.onion.io/omega2-docs/using-pwm-expansion.html [Accessed 15 October. 2020].

21 OASIS, (2019). MQTT Version 5.0. [Online] Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html [Accessed 10 June. 2019].

22 ElProCus - Electronic Projects for Engineering Students, (2013). The Working Principle of a PID Controller for Beginners. [Online] Available: https://www.elprocus.com/the-working-of-a-pid-controller/ [Accessed 28 July. 2020].

23 PID National Instruments, (2020). Theory Explained. [Online] Available: https://www.ni.com/fi-fi/innovations/white-papers/06/pid-theory-explained.html [Accessed 4 September. 2020].

24 Cypress.com, (2014). Universal Digital Block (UDB) Editor Guide. [Online] Available: https://www.cypress.com/file/139386/download [Accessed 8 October. 2019].