2019•2020
Faculteit Industriële ingenieurswetenschappen
master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Person tracking with a DMX-control system on FPGA for stage lighting and entertainment industry

PROMOTOR :
dr. ing. Jo VLIEGEN

PROMOTOR :
Dhr. Kristof DANIELS

Simon Aerts, Joost Poelmans
Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

2019•2020
Faculteit Industriële ingenieurswetenschappen
master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Person tracking with a DMX-control system on FPGA for stage lighting and entertainment industry

PROMOTOR :
dr. ing. Jo VLIEGEN

PROMOTOR :
Dhr. Kristof DANIELS

Simon Aerts, Joost Poelmans

Acknowledgements

The last year of our master in industrial engineering Electronics and ICT at UHasselt and KULeuven has been an exciting adventure. Especially since a large part has been spent during the COVID-19 crisis and this required some adjustments. Nevertheless, in this eventful time we still received a lot of help to complete the thesis. Therefore, we would like to express our sincere thanks to those who helped us.

Firstly, we would especially like to thank our promoter Dr. Jo Vliegen for the good advice and always being available to answer our questions. Despite the difficult times to carry out the practical tests, he arranged the minimal needed equipment for us to perform some of the tests at home. These equipment came from the ES&S research group and we would like to thank them for borrowing this material.

We would also like to thank DSV-Rent for providing us the stage lighting equipment. This ensured that we were also able to measure actual systems and to perform a few practical tests from home.

We would also like to express our gratitude to Sofie Puts (girlfriend Simon) for proofreading our thesis and correcting typos.

Finally, we would like to thank our parents for their endless support during our studies and for enabling us to complete this master.

We hope this thesis will appeal to you, the reader, and perhaps provide more information on the subject.

Contents

List of Tables

List of Figures

Abstract

There is a demand for automation in the professional audiovisual industry. One of those things is automating a follow spot which is normally operated manually. Some similar systems have already been released, but this thesis will tackle the problem from a different angle. The approach is to use a field-programmable gate array (FPGA) as a central controller. During the research, support was offered from DSV-Rent and the research group ES&S of KU Leuven.

The thesis works in a step-by-step manner. There is first a literature study that explores which available positioning systems are suitable for use in a follow spot. The study also mentions how the communication and processing of the data will proceed. Secondly, the structure of the system is discussed. It elaborates on the implementation of the code on the FPGA. Next, the overall setup is given. Each component is briefly explained to give an idea of the working principle of the designed system. Finally, the results are presented and discussed. It is shown that the system can interpret location data successively and convert it without introducing excessive delay.

However, the proof-of-concept implementation is still in a development phase and does not yet offer good user-friendly qualities. Additionally, it is not yet possible to make a comparison with existing systems as no tests have been done in the working environment due to the worldwide pandemic. The proof-of-concept implementation that is realised in this thesis is thoroughly tested and shows promising results.

Abstract in Dutch

Er is vraag naar automatisering in de professionele audiovisuele industrie. Een van die dingen is het automatiseren van een volgspot die normaal gesproken handmatig wordt bediend. Gelijkaardige systemen zijn al beschikbaar maar deze masterthesis zal het probleem vanuit een andere hoek benaderen. De aanpak is om een field-programmable gate array (FPGA) als centrale besturing te gebruiken. Tijdens het onderzoek werd ondersteuning geboden door DSV-Rent en de onderzoeksgroep ES&S van de KU Leuven. Eerst is er een literatuurstudie dat positioneringssystemen vergelijkt. Het onderzoek vermeldt ook hoe de communicatie en verwerking van de gegevens zal verlopen. Ten tweede wordt de structuur van het systeem besproken. Het behandelt de implementatie van de code op de FPGA. Vervolgens wordt de proefopstelling doorgenomen. Elk onderdeel wordt kort uitgelegd om een idee te geven van het werkingsprincipe van het ontworpen systeem. Tenslotte worden de resultaten weergegeven en besproken. Er wordt aangetoond dat het systeem locatiegegevens sequentieel kan interpreteren en converteren zonder grote vertragingen op te lopen. De proof-of-concept implementatie bevindt zich echter nog in een ontwikkelingsfase en is daarom niet gebruiksvriendelijk. Bovendien is het nog niet mogelijk om een vergelijking te maken met bestaande systemen omdat er door de wereldwijde pandemie geen praktijktesten mogelijk waren. De gerealiseerde proof-of-concept implementatie in deze masterproef werd grondig getest en laat veelbelovende resultaten zien.

Chapter 1

Introduction

The world is constantly innovating. Many applications have been developed that are still unused in all kinds of branches of industry. This master's thesis will explore the use of a field-programmable gate array (FPGA) in the professional audiovisual (AV) industry. The FPGA offers the flexibility to quickly develop a controller for the AV industry and receives external signals to apply further processing such as tracking. In this study, it is important to demonstrate the benefits of an FPGA. Hopefully there will be more interest in developing applications with FPGAs to support future innovation. In the current chapter, an overview will be given about the situation, a problem definition of this project, as well as the objectives and methods.

1.1 Context

The master's thesis is in cooperation with DSV-Rent located in Alken, Belgium and the research group "Embedded Systems & Security".

DSV-Rent is a company that manages the AV aspects for any type and size of rental project. One of the aspects of AV is lighting. Concerts and performances typically take place in dark environments. Therefore, a followspot is used to accentuate the performers in front of an audience. In most of the cases, these followspots are characterized with a pan, tilt, change size, beam width, and color. Nowadays, the properties of the followspot are manually controlled by the followspot operator. These operators are often positioned near the ceiling of the arena. Recently, the AV industry has been looking for an upgrade. A system where the operator can observe the stage with a camera and remotely control the beam of the followspot. This can be even further improved by using an automatic person tracking system that tracks the performances without the help of an operator.

Embedded Systems & Security is a research group of the KU Leuven. ES&S conducts research on embedded systems with extra focus on the security of digital data applications and efficient implementation. They provide technical support during the project. This includes lending the FPGA, positioning system and measuring equipment. With all this material, there was also the necessary advice to get started.

1.2 Problem statement

For the operation of the positional control and settings of followspots lighting, an operator should be present. An automatic spotlight control process is an innovation that can improve operational reliability and cost savings. Furthermore, the operator can focus more on the effects (i.e. focus, color) of the beam instead of pan and tilt of the followspot. A number of similar systems already exist [1–3], but they are often too expensive or inaccessible for small businesses. That is why there is a need for an affordable and quickly deployable system.

The automated person tracking system designed in this master's thesis can replace the manual pan and tilt control from the followspot operator. The improved system automatically determines the position of the performer with which it calculates the required changes in pan and tilt of the followspot. The communication between the stage light and automated person tracking system is performed using the industry-standard DMX512-A protocol defined by ANSI in [4]. The whole situation of AV should be taken into account, because the system can interfere with external signals. Because, he system will be used on different locations a portable design should be considered. Finally, the system must also guarantee stability and reliability to avoid breakdowns or unwanted surprises during a performance.

1.3 Objectives

The main objective of this master's thesis is to do a feasibility study on a person tracking system on an FPGA in stage lighting and entertainment industry. This can be split into a series of smaller objectives. The first objective is to conduct research into the different tracking systems and determine which are suitable for use in the professional audiovisual industry. The second objective is how to integrate this tracking system into a central system (FPGA). In concrete terms, this means looking at the interface between two systems.

Then there is the third objective where the focus is on how to design the controller. This will go deeper into the necessary logic functions of the FPGA. The penultimate objective is the question of how the central system can control the follow spot. The last objective is to test the obtained product. This phase will establish the properties of the product along with its strengths and weaknesses. The research question can be considered successful if the realized product has the functionality to have a spotlight follow a person. The product should be able to guarantee operational reliability and cost savings for the user. In this way, the subject is a candidate for more development in the future.

To give a concrete meaning to a functional followspot, the following specifications are specified:

1. a person tracking system with $\pm$10 cm precision,
2. an update rate of at least 10Hz to ensure smoothness,
3. easily integrated with other systems without causing hindrance,
4. competitive cost to compete in the market.

1.4 Methods and materials

Initially, a suitable local positioning system will be sought. This system must have an interface that is easy to integrate into an application. This will lead to use of an existing local positioning system (LPS) from Marvelmind Robotics (California, Unites States of America). The LPS includes software that shows the functionality of the product to the user. It also provides a well-documented manual which shows the available instructions of the hardware. Starting from this hardware, a Python example from Marvelmind's GitHub will be used to retrieve raw data from the device. With the appropriate knowledge regarding the Marvelmind's system, data transfer protocols supported by the hardware will be explored. The two options are (1) Serial Peripheral Interface (SPI) and (2) Universal asynchronous receiver-transmitter (UART). The best suited protocol will be selected to then implement it on a FPGA system.

The FPGA itself is a Zynq-7000 SOC on a Xilinx ZedBoard. The FPGA system will eventually direct the stage lighting and be the heart of the system. From this point, work will be done simultaneously on transforming the coordinates into angles and communicating with the followspot. The standard communication protocol commonly used in effects and stage lighting is DMX512-A. More information will be gathered about this in order to be able to implement this in the FPGA's hardware. At the same time, it is investigated how a trigonometric function is calculated in digital logic in order to calculate the pan and tilt angles of the followspot. After this, it is time to deploy a first prototype of the system. It is aimed to have it control the pan and tilt of a stage light by outputting a DMX512-A signal. This is followed by connecting the LPS system, or a system that imitates a LPS, to the FPGA to let the design calculate the angles. The DMX512-A signal at the output will then be controlled autonomously to transmit the correct angles. The inputs and outputs will be adjusted to have a correct transition of the voltage levels. During the design process, simulations in Vivado Design Suite and the use of an oscilloscope or logic analyzer will be used to provide feedback to the design. A final concept visualized in Unity will be used to evaluate the final results. The LPS system can be temporarily replaced by an Arduino to mimic the operation.

1.5 Thesis structure

The thesis has the following structure. First, chapter 1 introduces the subject. Chapter 2 will discuss the studies that have been done beforehand to provide more information on the subject. This provides an overview of the different local positioning systems and which systems may be suitable for person tracking on a stage. An algorithm is also explained that shows how, for example, a calculator can calculate trigonometric functions. The last two parts of chapter 2 briefly explain the operation of UART and DMX512-A. Chapter 3 is a transition from theoretical knowledge to practical implementation. The chosen positioning system is explained and the hardware implementation is informed to the reader. The results are discussed in chapter 4. This chapter also shows feedback in the form of a simulation. Chapter 5 provides the conclusion of the final product. In this chapter self-reflection is done and the future possibilities for the subject are discussed.

Chapter 2

Literature study

It is important to conduct an extensive literature review before continuing with the research itself. Having a good knowledge of the subject and knowing the current available techniques is an important factor in planning and designing an application prototype. A person tracker for a spotlight spans several disciplines. Therefore, each of these subjects is in the literature study. First, positioning will be discussed, with more explanation about the available technologies. This boils down to explaining the advantages and disadvantages of each technology and shows which technology fits well with the design of an automatic follow spot system. Subsequently, the literature study provides more information about the UART protocol, about the DMX512-A standard and about how the AV industry uses DMX512-A to control multiple devices. Finally, a technique to calculate computational angles is also discussed.

2.1 Local positioning systems

In this section, some Local Positioning Systems (LPSs) are discussed together with their field of operations. To provide some more insight into how these systems work, some measuring principles are also discussed, as well as the positioning algorithms that use them [5].

2.1.1 General information

The person tracker is intended to be used on a stage. This is a rather limited area but comes with the requirement that the location needs to be reported quite accurately. Using a satellite base positioning system like a GPS or Galileo, will not be fit for our application because it does not meet our requirement for high precision. Therefore, the focus will be on *indoor positioning systems* which are targeted more towards a limited area to cover but with greater precision.

Using an LPS on a stage has some challenges:

- Multipath effects caused by the reflection of walls and object like stage equipment;
- No Line-Of-Sight (possible blind spot);
- High attenuation and signal scattering originated by high density objects;
- Quick temporal changes due to moving object or people;

- Need for high accuracy, precision and low latency (spotlight mismatch would be noticeable for audience).

Covering a small area with fixed structure also has some advantages:

- Small area to cover means less beacons needed;
- If indoors $\Rightarrow$ minimal weather influence;
- Fixed geometric constraints;
- Usable infrastructure (electricity, Internet, walls for mounting);
- Lower kinematics (the velocity of the tracking objects being generally slow).

2.1.2 Requirements and parameters of an LPS

Each technology that exists has its own unique pros and cons. In order to determine which technology would be a great fit for our project, it would be a great start to look into their specifications. First, it is important to know the requirements of the application in order to define the exact specifications that are to be looked for. Therefore, a list for the most common requirements of an application from a *user's* perspective is provided by [5]:

- Accuracy
- Coverage area
- Cost
- Infrastructure (existing WiFi network, etc)
- Update rate
  - Periodic: specific interval (Hz)
  - Triggered by a user request
  - Triggered by a certain event
- Required output data
- Interface (graphic display, communication protocol, etc)
- Quality of service / Trustworthiness
  - Stability: malfunctions which might lead to exceeding the fault tolerance
  - Availability: continuity in time
- Scalability

When looking from the perspective of the *designer* of the local position system, there are some additional technical specifications to keep in mind:

- Type of technology (RF, optical, magnetic, etc)
- Positioning algorithm used
- Measuring principles
- Wavelength
- Measure types (signal strength, direction, acceleration, etc)

The following sections will elaborate on the most relevant aforementioned specifications.

2.1.3 Measuring principles

This section gives a short overview of different techniques to capture a distance or angular measurement which will be a foundation for the positioning methods discussed later on. The beacons or stationary nodes are usually fixed locations with known coordinates.

A more in-depth description will be given on:

1. Time of Arrival (ToA)
2. Time Difference of Arrival (TDoA)
3. Round Trip Time (RTT)
4. Phase of Arrival (PoA)
5. Angle of Arrival
6. Doppler ranging

Time of Arrival (ToA)

This method is based on knowing the exact time when a signal is sent, when the signal arrives and the speed at which the signal travels. By multiplying the speed with the difference in time, the approximate distance between the beacon and the target can be calculated. The possible location can be displayed as a sphere around the beacon. In order to predict the exact location, an additional two beacons are needed with respective circles so the intersection of these circles narrows down the position to 1 point. The velocity of the signal is often just the speed of light which uses c as a symbol. However, the electromagnetic propagation speed can easily be slowed down by the presence of certain obstacles. An example is the presence of concrete which can more than halve the propagation speed.

$$d = c \times (t_{arrival} - t_{sent}) \tag{2.1}$$

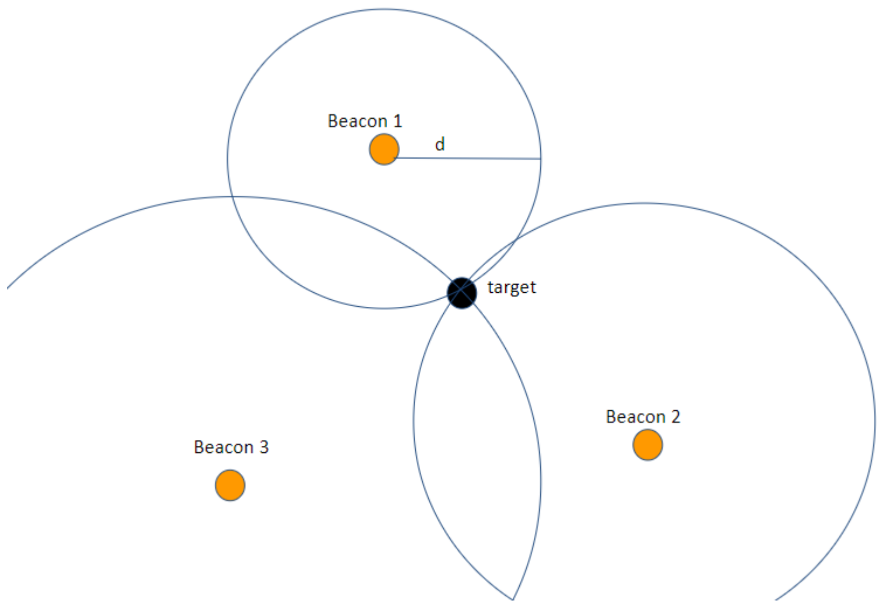


Figure 2.1: Time of arrival principle visualized

Time Difference of Arrival (TDoA)

This method does not need to know when the signal was sent. Only the time of arrivals measured on the beacons is used to calculate the time difference between beacons with $\Delta d = c \times \Delta t$. This time difference between two beacons can then be represented as an hyperbola [6].

$$\Delta d = \sqrt{(x_2 - x)^2 + (y_2 - y)^2} - \sqrt{(x_1 - x)^2 + (y_1 - y)^2} \tag{2.2}$$

With (x_1, y_1) and (x_2, y_2) being the coordinates of 2 beacons. Just like ToA, a total of 3 beacons are required to get a 1-dimensional result. The advantage of TDoA in comparison to ToA, is the fact that the target clock does not have to be accurately synchronised with the beacons.

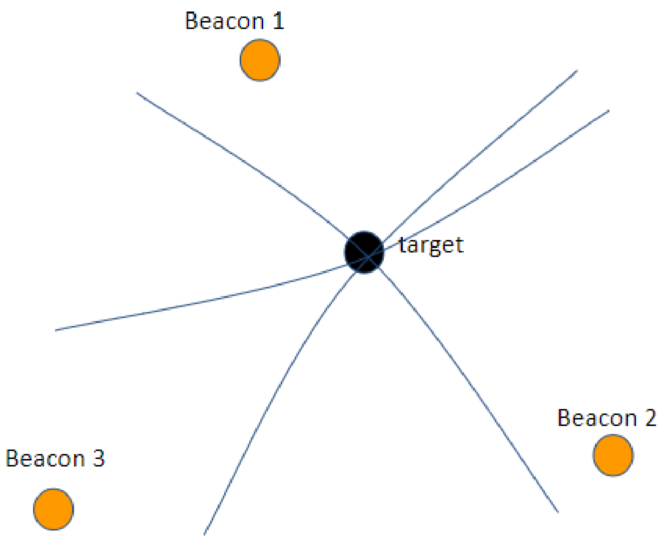


Figure 2.2: Time difference of arrival principle visualized

Round Trip Time (RTT)

Round trip time is basically measuring the time taken by the signal to travel from transmitter to receiver and back. The main advantage of implementing this principle is having no need for synchronization between the devices. No need for synchronization means that there is low complexity and cost when designing hardware that utilises RTT. Having an easy implementation comes at a cost: measurements over multiple devices need to be executed sequentially which causes an increased latency. RTT will usually not be used when the application requires a high update rate.

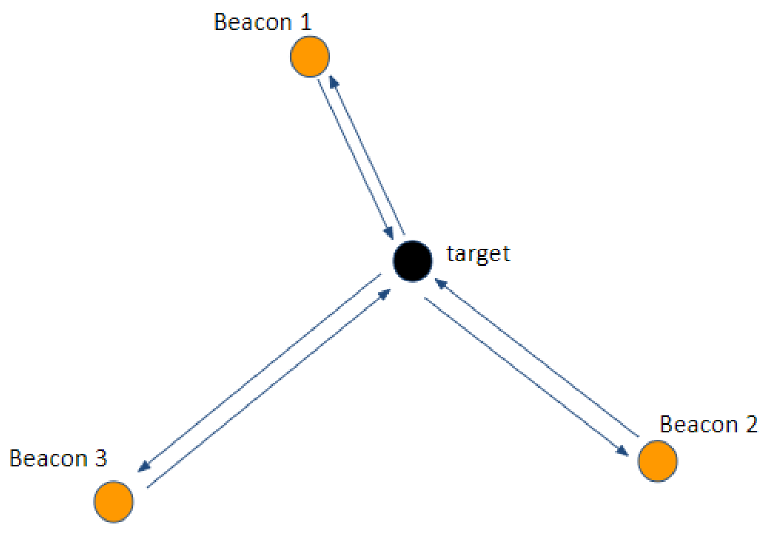


Figure 2.3: Round trip time principle visualized

Phase of Arrival (PoA)

Phase of arrival measures the distance using the received carrier phase. In order to solve the phase wrapping problem, one could use multiple frequencies to evaluate. On the other hand one can choose the wavelength accordingly to the maximum distance. As such, the phase stays between -180 and 180 degrees. Mostly, this is not a valid option when following the European table of frequency allocations [7] regulations.

A simple example is shown in Figure 2.4. It represents a signal sent by the target to a beacon and the phase is measured of both the target and the beacon. This measurement can be plotted on a sine-wave and shows that there is a phase difference between the two signals. Accordingly, the traveled distance can then be derived from the phase difference.

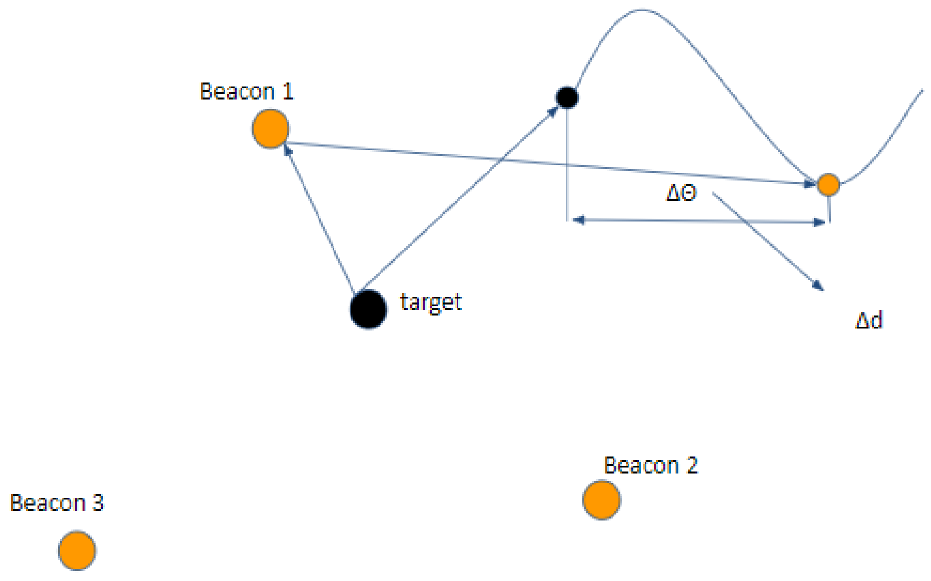


Figure 2.4: Phase of arrival principle visualized

Angle of Arrival

Angle of arrival is used to determine the incidence direction. It has multiple antenna in a grid, and can measure a difference in phase between each of these antenna which provides information to calculate the angle theta. More details are available in the article of Wielandt and De Strycker [8, p. 7].

Doppler ranging

This principle is only used to determine the speed between receiver and transmitter. When transmitting a signal while travelling at a certain speed, waves are "compressed" or "decompressed" and a receiver might perceive this as a signal with a higher or lower frequency, respectively. This property can be used at the receiver to calculate the speed difference with which the transmitter is travelling. If the receiver is stationary, this results in the speed of the transmitter.

2.1.4 Positioning methods

This section gives a short overview of different techniques for positioning. A more in-depth description will be given on:

1. Cell of Origin (CoO)
2. Centroid Determination
3. Multilateration
4. Polar Point Method
5. Fingerprinting
6. Dead Reckoning

Cell of Origin (CoO)

Cell of origin defines the locations of a mobile device merely on its presence in a certain area. Multiple cells can be available to a device, but the one with the strongest signal is the one which will be assigned to the device's location. A common example is a phone cell tower that can help with identifying the location of a phone. The nature of this method also indicates that the technique is not really intended to be accurate. Therefore, it is mainly used when only a rough estimate is needed.

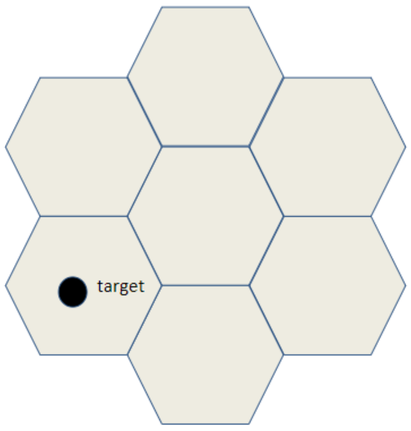


Figure 2.5: Cell of origin principle visualized

Centroid Determination

A mobile station receives signals from multiple beacons. Each signal has its own Received Signal Strength Indicator (RSSI). It is possible to calculate a weighted centroid location with the RSSI used as a weight in the calculation. An example is shown in Figure 2.6. There are two orange beacons and one black tracking device. The signal strength of the second beacon will be stronger because it is much closer to the target. This will lead to having more impact of the X2 coordinate in equation (2.3) so the location of the mobile device will be located closer to beacon 2.

$$X_{target} = \frac{RSSI_1 \times X_1 + RSSI_2 \times X_2}{2} \tag{2.3}$$

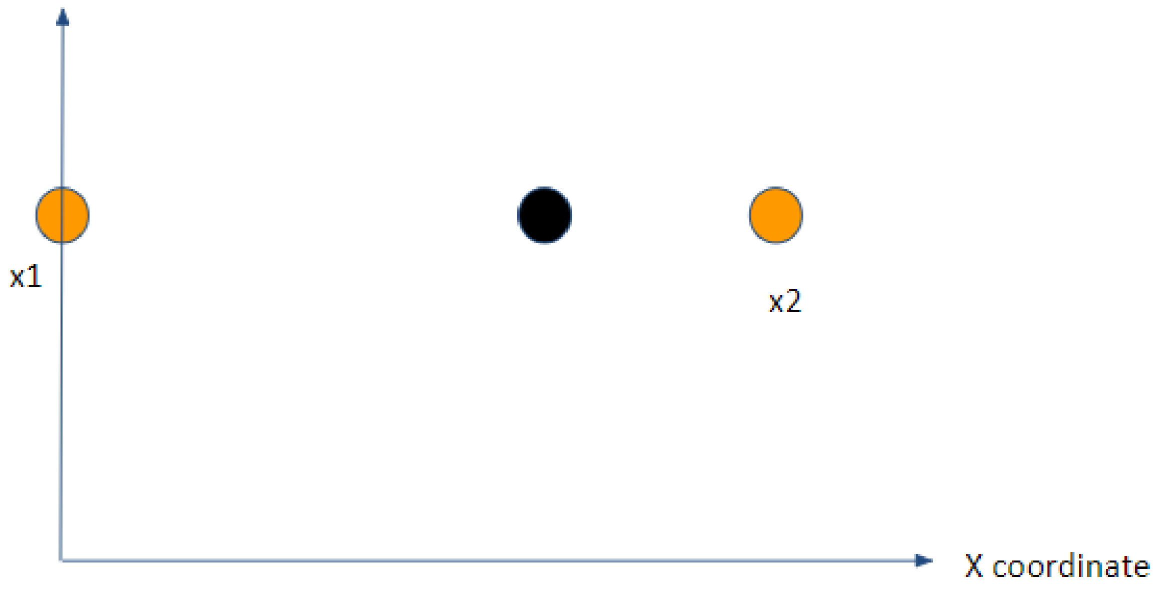


Figure 2.6: Centroid determination method example

Multilateration

Multilateration is the determination of the position of an object by using multiple distance measurements coming from different nodes. These distance measurements can be techniques like

ToA, TDoA and RTT which are briefly explained in Section 2.1.3.

Polar Point Method

Only one station is needed. This station delivers a distance measurement and an angle measurement. The location is then defined by these 2 parameters.

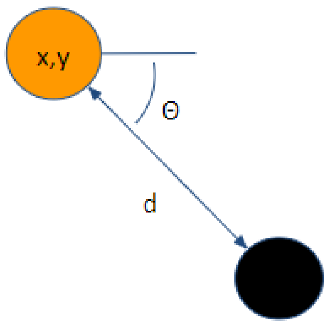


Figure 2.7: Polar point method

Fingerprinting

It relies on the recording of a parameter that changes with distance. Usually, the signal strength is used from several nodes, beacons or access points. Firstly, the target must go around and store the measurements of signal strength with known location coordinates. Consequently, a measurement with location (x,y) with a given RSSI of each node will be stored. This will be performed for multiple locations until sufficiently data is present. This data will be stored in a database which will be used when the systems comes “online”. Once the system comes online and the mobile devices want to know its location, it will measure the RSSI of each node and compare it to the database. The best match will be used as location coordinate of the mobile device.

This technique is often used with Wi-Fi. One of the main disadvantages is that a change in the environment can cause a complete different fingerprinting map. In order to restore this, one must update the database again.

Dead Reckoning

Using a previously know position and measuring the velocity over time of the mobile station, dead reckoning is able to give an estimation of a position. The main problem of this method is that the previously given location will be used. No location measurement is perfect with the consequence that that every new location is built upon the previous made errors. Therefore, deviations of positions are cumulative and may cause a substantial error after few iterations. Dead Reckoning is mostly combined with absolute positioning methods and a Kalman filter. An example of dead reckoning can be the situation of a car driving into a tunnel and losing its GPS signal. The Inertial Measurement Unit (IMU) can be used to predict the location knowing the car’s initial position and the speed it is driving at.

2.1.5 Local positioning technologies

Positioning systems always use a type of medium to determine a position. The most common ones are electromagnetic waves. However, there is also a different approach like using sound waves. When trying to determine the fitting positioning technology, it is recommended to compare the performance parameters of the technology with the user application. Figure 2.8 shows several technologies where the 2 most commonly important requirements are plotted on the x and y graph respectively accuracy and coverage.

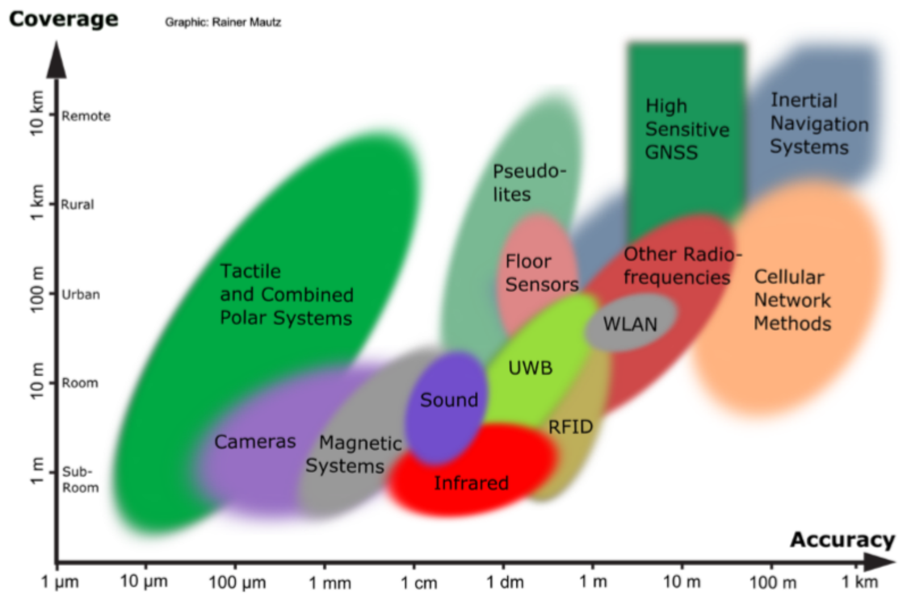


Figure 2.8: Overview of indoor technologies in dependence on accuracy and coverage

Source: [5]

Starting from this, there is already a good indication of what types of technologies need to be addressed further when keeping our application in mind. Therefore, a small summary is provided of possible candidates. It is important to keep in mind that many of these technologies are able to implement a wide variety of positioning methods and even use a combination of multiple methods. Combining methods gives options to play with the strengths of each method and try to reduce the disadvantages that comes with each and every positioning method.

Wireless Local Area Network

A Wireless Local Area Network (WLAN) or also known as Wi-Fi is a common infrastructure these days. It is very common in a lot of devices and makes it a cheap technology to set up a positioning system. One of the major hurdles is the high signal attenuation. This can lead to having to use a high amount of access-points to compensate for the signal loss. The positioning methods often use the RSSI of an access point with the most common positioning method being the fingerprinting method as stated in [9], [10]. However, the accuracy is only in the order of meters.

Bluetooth

Bluetooth has similar benefits as Wi-Fi. It is common in many devices and provides a low cost solution. However, the main disadvantage is the shorter range of Bluetooth. This has the outcome that even more beacons are needed for the same coverage area in comparison with WLAN. Nonetheless, it is more power efficient. It gives the option to operate beacons with a battery as opposed to Wi-Fi. This is especially the case with Bluetooth Low Energy (BLE) which can operate months with a single coin-cell battery. One of the most common methods is the use of cell of origin. A final remark is that the accuracy is only good enough to detect the presence of the target at room level.

Ulta-WideBand

Ultra-WideBand (UWB) is a short-range technology with a high bandwidth. A signal is declared as a UWB signal when the bandwidth occupies more than 500MHz or 20% of the carrier frequency. There are additional specifications which a UWB signal needs to comply with in order to maintain its legal state when unlicensed. These specifications are described by the European Communications Committee (ECC) [11] to avoid interference with other radio signals. This states for example that the signal is restricted between 6.0 GHz and 8.5 GHz. UWB has a better penetrability through walls and has less problems regarding a multipath. It results in a lower demand of transmitters in comparison with a technology like WLAN which suffers from high signal attenuation. UWB is an important technology because the precision is substantially better than most. Indoor applications like for example robotic movement inside a storage area will often need high accuracy. UWB is the desired technology because most implementations provide an accuracy below a meter down to several centimeters. There are also techniques like *passive UWB localisation* which uses the retrieved scattered waves which has the possibility to detect scattered waves reflecting of a person.

It is important to point out that the range is limited to 100m for safety by the ECC. Additionally, when trying to consider the costs, this system is not using any existing or common cheap infrastructure like Bluetooth or WLAN. This can be an extra unnecessary cost when low accuracy is needed for the application.

Sound

Sound is another interesting technology to use when trying to locate a position. It uses mechanical waves as a medium and propagates at lower speed than electromagnetic waves. Most positioning systems use ultrasonic waves. However, there are some systems that use the audible frequency range. It is mainly used indoor because there are environmental influences which can be problematic. The first major problem can be temperature. Temperature influences the speed of sound which can lead to performance degradation when the system only uses the standard speed of sound which is 344 m/s. This change in speed can be estimated with equation (2.4).

$$v_{\text{air}} = (331.3 + 0.606 \cdot T) \quad \text{m/s} \tag{2.4}$$

Another detrimental effect is wind. Here, there is also Doppler shifting of the frequency that gives inconsistencies. The system's coverage area is only between 10 to 30 meters. However, it is possible to get centimeter accuracy. This means that it competes with UWB in terms of high precision and relatively large coverage area applications. The advantages of using sound is the fact that a time calculation becomes easy to execute. It can use a time difference of arrival approach when broadcasting an RF and ultrasound signal at the same time. The receiver can listen to these two signals and extract the distance by using the time difference of arrival with a simple equation:

$$\Delta t = \frac{distance}{v_{air}} - \frac{distance}{v_{RF}} \tag{2.5}$$

The propagation speed of light is fast in comparison with sound that the second term in previous equation can be omitted. A last important note is the rather slow update rate of a sound positioning system. This value can vary between 10Hz and 20Hz but decreases as the distance between beacon and target increases.

2.1.6 Summary

Table 2.1: Overview of the technologies

	Accuracy	Common measuring principals	range	update rate	suitability
WLAN	-	fingerprinting	-	++	-
Bluetooth	−	CoO	−	++	-
UWB	+	T(D)oA / RTT	++	++	++
Ultrasound	++	TDoA	+	-	+

With a small overview of types of technologies and which methods, principles, advantages and disadvantages, it is now possible to understand what type of local positioning system should suit a certain application. The question remains which system on the market will suffice. It is important to look at the requirements and identify the available options. The application of a person tracker for stage lighting will have a heavy focus on accuracy and update rate to ensure smoothness. This section indicated that UWB and sound are two possible contenders.

2.2 CORDIC

In order to retrieve the pan and tilt from the coordinates, it is necessary to implement a way to calculate trigonometric functions. An ideal solution is to use CORDIC which stands for COordinate Rotation DIgital Computer. This method is relatively easy to understand and implement in Verilog [12], [13]. It is based on the principle of vector rotation and increasing the accuracy with each iteration as long as it is implemented in memory beforehand. First, the algorithm for calculating the sine and cosine functions is explained. Thereafter, a brief explanation is given to adapt this to calculate the arc tangent of an angle.

A basic vector representation is shown in Figure 2.9. To calculate vector V_n which is V_0 rotated over θ, the following equations can be used:

$$\begin{aligned} X_n &= X_0 cos(\theta) - Y_0 sin(\theta) \\ Y_n &= Y_0 cos(\theta) + X_0 sin(\theta) \end{aligned} \tag{2.6}$$

One can also rewrite these equations as:

$$\begin{aligned} X_n &= cos(\theta)[X_0 - Y_0 tan(\theta)] \\ Y_n &= cos(\theta)[Y_0 + X_0 tan(\theta)] \end{aligned} \tag{2.7}$$

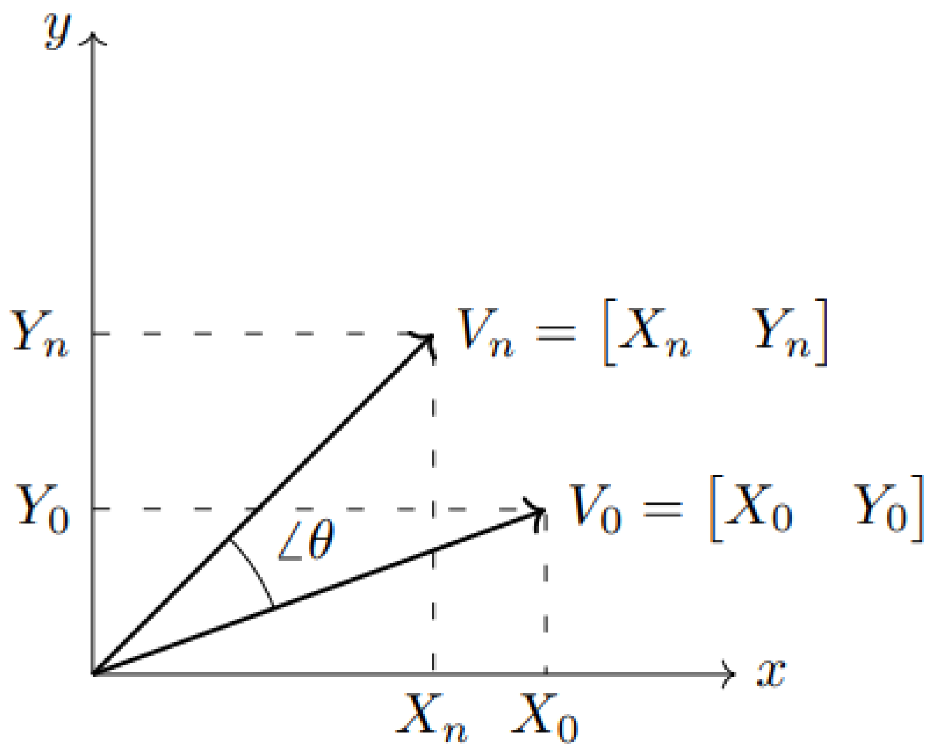


Figure 2.9: Rotation of vectors

Now in the CORDIC algorithm there will be restrictions to the angle so that $tan(\theta) = 2^{-i}$. Examples of the angles are given in Table 2.2.

Table 2.2: Example of tangent angles for the CORDIC algorithm.

i	$tan(\theta) = 2^{-i}$	θ
0	1	45°
1	1/2	26.565°
2	1/4	14.036°
3	1/8	7.125°
4	1/16	3.576°
...	...	...

Rewriting the cosine as a function of tangent and using the angle restriction will give us the following equation:

$$cos(\theta) = \frac{1}{\sqrt{\frac{1}{cos(\theta)^2}}} = \frac{1}{\sqrt{\frac{cos(\theta)^2+sin(\theta)^2}{cos(\theta)^2}}} = \frac{1}{\sqrt{1+tan(\theta)^2}} = \frac{1}{\sqrt{1+tan^2(tan^{-1}(2^{-i}))}} = \frac{1}{\sqrt{1+(2^{-2i})}} = K_i$$

The value of K_i is only determined by i. K_i is therefore independent of the rotation, direction of the rotation, X and Y of each iteration. K_i replaces $cos(\theta)$ but does not account for the possible negative value of the cosine function ($cos(\theta)$ lays between -1 and 1). Therefore, a parameter d_i will complete the replacement of the cosine function. With the introduction of K_i together with the parameter d_i to control the direct of the vector rotation, it is possible to rewrite the vector equation without a trigonometric function as displayed at (2.8).

$$
\begin{aligned}
X_n &= K_i[X_0 - Y_0 d_i 2^{-i}] \\
Y_n &= K_i[Y_0 + X_0 d_i 2^{-i}] \; with \; d_i = \pm 1
\end{aligned}
\tag{2.8}
$$

Note that the restrictions on θ have been introduced specifically so that in the calculation of X_n and Y_n, a bit shift to the right is used instead of calculating the tangent. X_n and Y_n will later on represent the new coordinates for each iteration step (or rotation step). The question remains now, how does an angle other than the ones listed in the Table 2.2 get calculated ? With equation 2.8, it is possible to do iterative operations that essentially perform a series of successively smaller rotations that will converge to the correct angle. Accuracy is dependent on how many iterations are done. Each iteration uses a value that is stored in the Lookup table (LUT) since the row number of the lookup Table corresponds to the current iteration number. Every additional iteration increases the convergence to the actual angle and thus improves accuracy. The amount of entries in the Table is chosen by the required precision of the application. However, that is not the only thing to keep in mind. The bit shift right in equation 2.8 means that there is no point in doing more iterations than the bit width of X and Y. So when doing Niterations it is important to choose a LUT with a length longer than at least the iteration count and a bit width for X and Y which is also at least as long.

Table 2.3 shows the iterative step towards a desired angle of 20 degrees. With each step there is a calculation of X and Y according to Equation 2.8. The rotation direction or d_i is evaluated by whether the current angle is greater or smaller than 0.

Table 2.3: CORDIC example of calculating 20°.

i	$tan(\theta) = 2^{-i}$	$\theta = atan(2^{-i})$	Current angle	Rotation
0	1	45°	20°	−45°
1	1/2	26.565°	−25°	26.565°
2	1/4	14.036°	1.565°	−14.036°
3	1/8	7.125°	−12.471°	7.125°
4	1/16	3.576°	−5.346°	3.576°
...	...	...	...	...
19	...	...	−0.000195°	...

It is important to realize that the factor K_i is the same for each step because the tangent is the same for both the positive and the negative angle. Another important feature of K_i is the occurrence of convergence after several iterations seen at Equation 2.9. This allows the ability to easily remove the K factor from the algorithm later on by applying the inverse of K at some step in the process. K will be approximately 0.6073 when the iteration count N is at least seven.

$$
K_i = \prod_{i=0}^{N} \left(\frac{1}{\sqrt{1 + (2^{-2i})}} \right) \tag{2.9}
$$

$$
\Rightarrow K_i = (0.7071) \times (0.8944) \times (0.9701) \times (0.9923) \times (9981) \times (0.9995) \times (0.9999) \times (1) \times (1) ... = 0.6073
$$

With the basics explained, it is possible to give a summary of the algorithm:

$$\begin{aligned}
X_{i+1} &= [X_i - Y_i d_i 2^{-i}] \\
Y_{i+1} &= [Y_i + X_i d_i 2^{-i}] \\
X_{final} &= K \times X_N \\
Y_{final} &= K \times Y_N \\
Angle_{i+1} &= Angle_i - d_i tan^{-1}(2^{-i})
\end{aligned} \tag{2.10}$$

$$with\, N \geq 7\, in\, order\, for\, K = 0.6073\, and\, d_i = \pm 1$$

This algorithm basically computes a vector rotation without any trigonometrical functions and can therefore be represented as equation 2.6. If X is equal to 1 and Y is equal to 0, equation 2.6 becomes:

$$\begin{aligned}
X_n &= cos(\theta) \\
Y_n &= sin(\theta)
\end{aligned} \tag{2.11}$$

A final remark is that only an angle between $-90° \leq \theta \leq -90°$ can be calculated with this algorithm. A pre-rotation is required in order to calculate different angles. This is something that will be pointed out later on in the Verilog implementation on the FPGA in section 3.1.3.

The CORDIC method can be adapted to calculate inverse trigonometric functions, hyperbolic functions, logarithms and more as mentioned in [14]. In this project it was necessary to process the coordinates coming from a positioning system and convert it to an angle. To calculate the pan and tilt from the spotlight, the arctangent of X&Y and X&Z is taken, respectively. The CORDIC method that has been previously explained will need some adjustment to change it to being able to calculate the cosine and sine of an angle to calculate the arctangent when X and Y are given as input. The algorithm will look like Equation 2.12. Instead of using the angle as an input and calculating X and Y, this method will use the X and Y as an input and calculate an angle. The vector will slowly converge by fluctuating around zero degrees which is done by using the sign of the Y vector. The algorithm is keeping track of the change in angle with each iteration and will represent the arctan of X and Y. Unlike the previous method of calculating the sine and cosine, it is not necessary to compensate for the factor K because K does not influence the outcome of the angle.

$$\begin{aligned}
X_{i+1} &= [X_i - Y_i d_i 2^{-i}] \\
Y_{i+1} &= [Y_i + X_i d_i 2^{-i}] \\
Angle_{i+1} &= Angle_i - d_i tan^{-1}(2^{-i})
\end{aligned} \tag{2.12}$$

$$with\, d_i = 1\, when\, Y\, is\, negative\, and\, d_i = -1\, when\, Y\, is\, Positive$$

2.3 UART

The Universal Asynchronous Receiver and Transmitter (UART) uses a data protocol to send data over a serial line, as illustrated in Figure 2.10. It contains a transmitter and a receiver.

The transmitter is a shift register that loads data in parallel. This is followed by shifting the bits one by one at a specific frequency. The receiver shifts data bit by bit in. The serial line is logic 1 when there is no communication happening (IDLE state). The communication is sent into frames. Each frame contains a start bit, data bits, stop bits and an optional parity bit. The start bit is always logic 0. The data bits can contain 6, 7 or 8 bits and these bits can logic 0 or logic 1 depending of which data is being sent. The number of stop bits can be 1,1.5 or 2. These bits are always logic 1. Error detection can be done with the use of the parity bit.

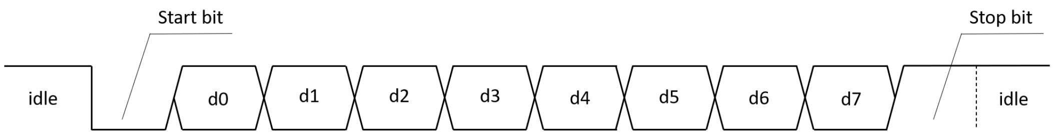


Figure 2.10: Example of UART frame with 8 data bits, no parity bit and 1 stop bit

The transmitter and receiver must agree on the following parameters before the transmission can happen: the clock frequency, the amount of data bits and stop bits, and if a parity bit is present. Otherwise, the receiver will not know at which and what kind of frame is sent by the transmitter.

2.4 DMX512-A

Digital Multiplexed (DMX) 512-A is the current standard [15] that describes the transmission between controllers and controlled lighting equipment and accessories. It is the successor of the DMX512 standard (without A). The remainder of this thesis refers to DMX512-A as DMX.

2.4.1 DMX data protocol

The DMX data protocol uses the UART data protocol with 8 bits (or a byte) as data bits, 2 stop bits, no parity bit and 250000 baud rate as basis. Therefore a slot (another name for a frame in the DMX standard) contains a data value between 0-255. These slots are always send into a packet of multiple slots, also called a universe. Each universe can contain up to 513 slots. Typically, the first slot contains 0x00 (NULL START) as data, as this is the default value. Other values on the first slot are for further expansions of the standard. The secondary slot is the first slot that contains actual data. A SPACE and mark after break (MAB) are used to announce a new universe. The timing of each frame is given in the tables below. Something to mention is that an error of 4% is allowed on each bit.

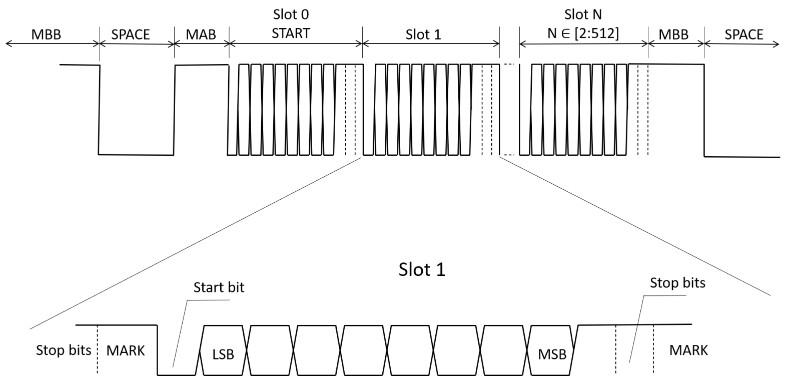


Figure 2.11: DMX data protocol

The specific timing of the transmitter and receiver the DMX signals are given in Table 2.4 and Table 2.5.

Table 2.4: Timing diagram values for transmitting a DMX signal

Source: [15]

Description	Min	Typical	Max	Unit
Bit rate	245	250	255	kbit/s
Bit time	3.92	4	4.08	μs
Minimum update time for 513 slots	-	22.7	-	ms
Maximum Refresh Rate for 513 slots		44		updates/s
”SPACE” for BREAK	92	176	-	μs
“MARK” After BREAK (MAB)	12	-	- < 1.00	μs s
“MARK” between slots (MBS)	0	-	<1.00	s
“MARK” Before BREAK (MBB)	0	-	<1.00	s
DMX packet or universe	1204	-	- < 1.00	μs s

Table 2.5: Timing diagram values for receiving a DMX signal

Source: [15]

Description	Min	Typical	Max	Unit
Bit rate	245	250	255	kbit/s
Bit time	3.92	4	4.08	μs
Minimum update time for 513 slots	-	22.7	-	ms
Maximum Refresh Rate for 513 slots		44		updates/s
”SPACE” for BREAK	88	176	-	μs
“MARK” After BREAK (MAB)	8	-	- < 1.00	μs s
“MARK” between slots (MBS)	0	-	<1.00	s
“MARK” Before BREAK (MBB)	0	-	<1.00	s
DMX packet or universe	1196	-	- < 1.00	μs s

2.4.2 Communication

DMX implements a simplex serial communication system. One DMX controller can send the DMX signal up to 32 devices. All the devices that are in the same ring, receive the same slots. The slot out of which they use data is set on the device itself. Therefore multiple devices can get data out of the same slots or they can all use individual slots.

There are multiple varieties XLR cables. These cables are used in the professional audio, video and lighting industry. The standard DMX recommends to use XLR5 connectors and cables to connect devices with each other. XLR5 cables contain 1 wire for the *ground*. DMX universes are sent over the *data 1* wires. This is done with the use of the RS485 protocol. The 2 wires for *data 2* of the XLR5 cables are used in enhanced version of DMX. Therefore, the industry also uses XLR3 (XLR5 without data 2 wires) connectors to save money.

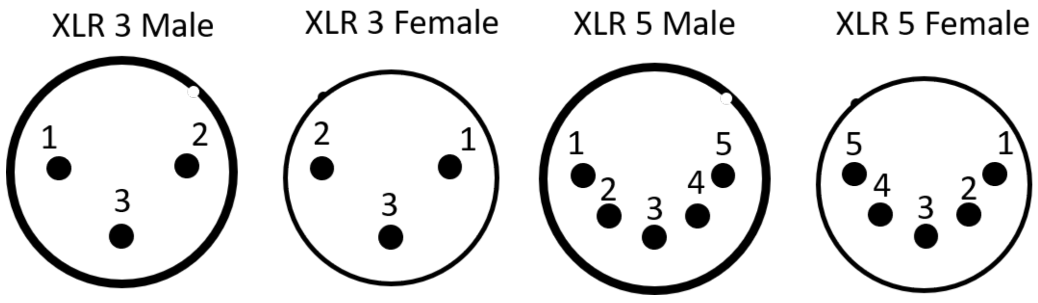


Figure 2.12: The front view of XLR connectors

Table 2.6: Properties XLR cables

pin assignment	name	Limit Volt with Common	Comment
1	Ground	0 V	
2	data 1-	$0 < v < +6$ VDC	
3	data 1+	$0 < v < +6$ VDC	
4	data 2-	$0 < v < +6$ VDC	only enchanced function
5	data 2+	$0 < v < +6$ VDC	only enchanced function

2.5 Results of the literature study

The literature study provides information to choose an appropriate positioning system. It is suggested that UWB and ultrasound are possible candidates when an accurate positioning system is required. Looking at existing systems, it is noticeable that many stage tracking systems are based on UWB (e.g. Zactrack and Follow-me TraXYZ [1], [2]). UWB offers the necessary accuracy and is at the same time a robust system with a reasonably large coverage area.

The CORDIC algorithm study shows how to calculate trigonometric functions in a step-by-step manner with only a simple lookup Table and an iterative procedure. This will be applied in a hardware description language (HDL) to implement it on an FPGA. In addition to the CORDIC implementation on the FPGA, there will also be a module that will use UART and DMX. UART is a well-known protocol and therefore, it is easy to start from an existing implementation. For DMX, a new implementation will be designed.

2.6 Conclusion

With a small overview of different LPS-type technologies and methods, principles, advantages and disadvantages, it is now possible to understand which type of LPS should suit a certain application. The question remains which system on the market will suffice. It is important to look at the requirements and identify what the available options are. The application of a person tracker for stage lighting will have a heavy focus on accuracy and update rate to ensure smoothness. With regard to CORDIC, UART and DMX, sufficient insight has now been created to start with the Verilog implementation.

Chapter 3

Methods

Following chapter describes the design of the system. It is a step by step explanation of how the system is created. Each section follows on the previous section and adds parts to the system to get the final result. The first section *Parts* describes the 5 major parts of the system. There is an in depth explanation of what the function of these parts are, how these parts work and what the input and output of these parts are. Section two *Communication* describes how the communication between these parts is done and how this communication is implemented. Next, the section *Simulation* explains how a simulation design + environment is created to test the system. The last section describes the measuring equipment.

3.1 Parts

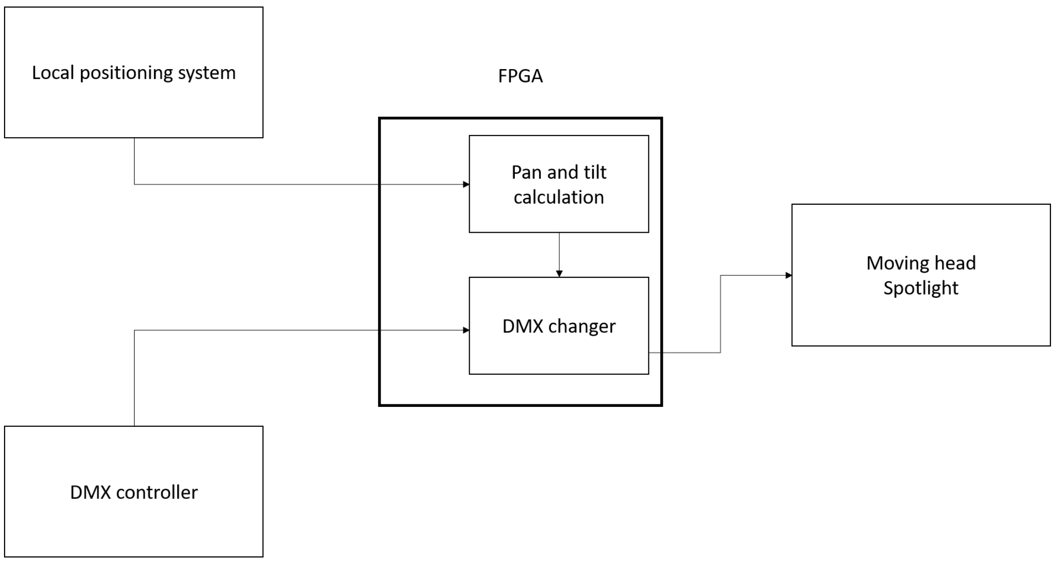


Figure 3.1: Block diagram of the implementation

There are 5 individual parts that all have their own major role in the system. These are:

- The local positioning system that measures the coordinates of the artist and send these to the pan and tilt calculation;
- DMX controller that is controlled by the operator to change data on the DMX slots;

- The pan and tilt calculation of the coordinates on the FPGA;
- DMX changer that adds the pan and tilt to the DMX signal;
- The moving head spotlight that follows the artist on the stage.

3.1.1 Field-programmable gate array

An FPGA is an circuit which is designed to be configurable after manufacturing. This configuration is done by an hardware description language (HDL). HDL is also used to program an application specific integrated circuit (ASIC) which is used in applications where it is specifically designed for as it is not programmable. An FPGA is chosen in this thesis as it provides an easy way to develop, test and iterate over each new design. The specific device used in this master thesis is the ZedBoard Zynq-7000 ARM / FPGA SoC Development Board [16]. It has the ability to configure programmable logic (PL) and utilize a processing system (PS) namely an ARM processor. However, for the person tracking application, everything is implemented in PL. Future work could use the PS to design a more flexible application using software. More information about the board can be found at [17].

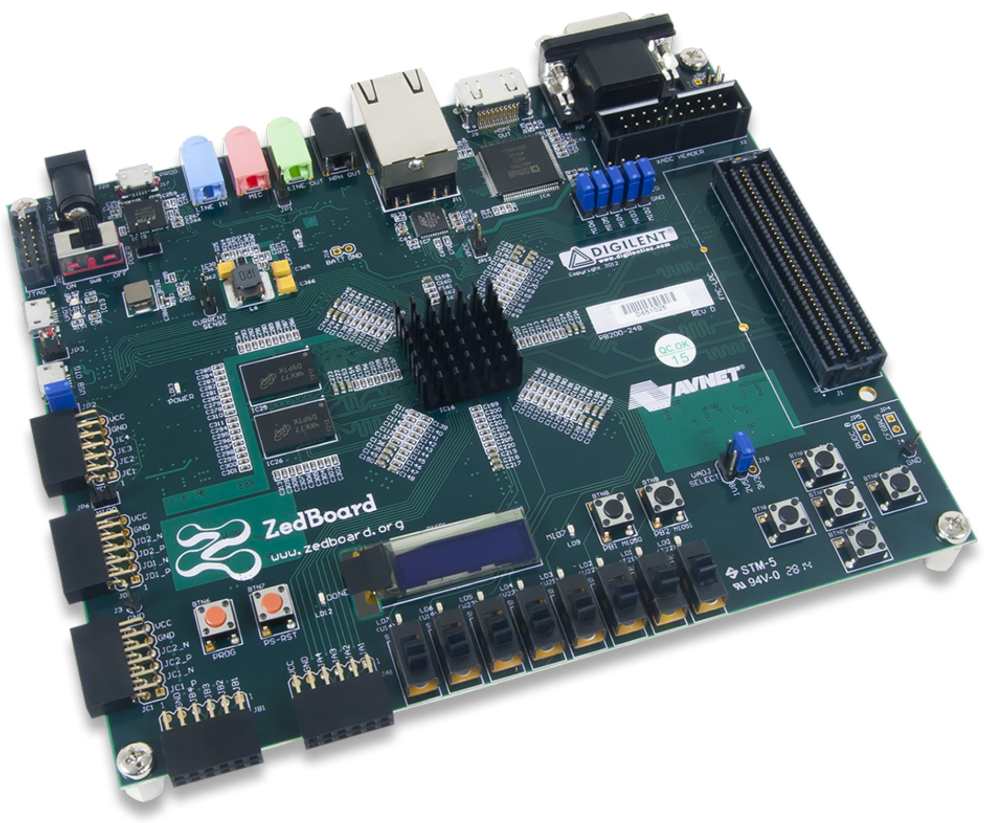


Figure 3.2: Zynq-7000 ARM/FPGA SoC Development Board

Source: [17]

The FPGA in this thesis is configured in Verilog which is one of the most popular HDLs. The other variant is VHDL. The designs in the FPGA consists of a DMX, pan and tilt via CORDIC, and UART implementation using a 100MHz clock. Of course, a number of inputs and outputs must be addressed to exchange the UART and DMX signals. To do this, the JA Digilint Pmod header (2x6) is currently being used. These general purpose headers consist of 12 female pins, that are placed in 2 rows and 6 columns. It has 2 pins that provide a Vcc of 3.3 volt and 2 pins that are used as a ground. The rest of the pins can be used as input and output. More information about all connectors and specifications can be found in the data sheet of the zedboard at [16].

I/O pin assignment

Pin JA1 is currently used to receive UART serial communication. Pin JA9 sends out the modified DMX signal. Pin JA2 can serve as the input signal for the DMX controller. The input signal is then used to control the output signal. An input signal can also be generated internally on the FPGA if there is no existing DMX signal to use. Finally, button BTND on the board is used as a reset for the design.

3.1.2 Local positioning system

The literature study provides a baseline to start looking at which requirements and technologies are needed for a stage light tracking system. The LPS should try to comply with the objectives that are discussed in Section 1.3. It is decided to start with a turnkey based positioning system as a design of a new system would require a lot of resources and would probably not provide the finishing touches as the commonly used systems on the market. The study also found that ultra-wideband or ultrasound are the technologies that offer sufficient accuracy. UWB is the technology used in some existing stage light trackers such as Zactrack and Follow-me TraXYZ [1], [2]. Therefore, initially an existing UWB system is sought that offered easy integration for the FPGA. Then, there turned out to be an available system in research group "Embedded Systems & Security". However it was a system based on ultrasound and is designed by a company called *Marvelmind Robotics* [18].

Their devices are split into two architectures, namely inverse architecture (IA) and non-inverse architecture (NIA). The IA means that the stationary beacons emit ultrasound and the mobile beacons receive it. The NIA will then have mobile beacons that emit ultrasound and stationary beacons that receive the signal. There are some major differences between these architectures. Most importantly, the NIA systems generally have slightly better accuracy and can use between 1 to 4 mobile beacons. IA systems, however, have the option to add more mobile beacons. Additional information between the two architectures can be found in the architecture comparison sheet available at [19].

The available system of the research institution is a NIA starter set system shown in Figure 3.3. The starter set has four stationary beacons, one mobile beacon and one modem that supports up to 250 beacons. This is certainly sufficient since the focus is initially on using only one mobile beacon. The device is advertised as a ± 2 cm indoor navigation system. It is often used for autonomous vehicles, robots and drones. The beacons transmit both ultrasound and radio frequencies (RF). The RF are in the license-free band of 433MHz or 868Mhz. As already discussed in the literature study, the system determines the position of the mobile beacon based on the propagation speed of sound. The Time-of-Flight (ToF) or also called the Time of Arrival (ToA) is measured with respect to each beacon and a location is calculated by multilateration. It is important that for valid 3D tracking, there must be at least 3 stationary beacons in line of sight of the mobile beacon. Furthermore, it is recommended that the distance between the two nearest beacons should not exceed 30 meters.

The coverage area with the starter set configuration can be up to 1000 m^2. The update rate can vary from 0.05Hz to 25Hz. This can be adjusted manually, but the maximum speed varies depending on the distance between mobile and stationary beacon. A smaller distance gives a shorter delay and therefore, it gives a higher update rate in exchange. For the NIA architecture, multiple beacons would have a huge detrimental effect on the update rate as it works via a time division multiple access (TDMA) method. In short, the update rate can be determined by multiplying $\frac{1}{n}$ with the update rate where n represents the number of mobile beacons. The beacons have a LiPo battery of 1000 mAh. The vendor claims that the stationary beacons with an update rate of 16Hz can be operational for up to 72 hours. For the mobile beacons with 8Hz, this is 12 hours.

Marvelmind interface

Figure 3.3: Marvelmind starter set HW v4.9-NIA

In terms of interface, 2 protocols are available: 1) SPI, and 2) UART, which are both available via pins on the beacon board or router. Virtual UART through USB is also an option. UART through the physical pins is the interface on which this project is based. SPI has an higher data throughput but it is not necessary as a UART connection of the supported 500kbps suffices. SPI has a chip enable signal which is not necessary because communication does not need to be multiplexed between different devices. UART is relatively easy to implement and allows to quickly set up a test rig. Hence, UART as communication protocol between the marvelmind and FPGA, is chosen.

The datasheet about the marvelmind interfaces [20] has been consulted to gain insight into how the communication proceeds. This datasheet contains a table that provides an insightful overview of the format of a streaming packet. Based on this, a HDL design is made to decode the packets.

As seen in Table 3.1, the first packet to arrive needs to be 0xFF then 0x47 needs to arrive on the FPGA and the rest of the data packet depends on the type of data packet. Multiple packages are available. There is the option to receive the hedgehog (mobile beacon) coordinates in centimeters or in millimeter resolution. It is also possible to receive the locations of all beacons. In addition to receiving coordinates, there is the option to receive data from the inertial sensors, telemetry data and more. In order to realize a first practical version, it is chosen to work with data code

Table 3.1: Streaming packet format

Source: [20]

Offset	Size (bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x47
2	2	uint16_t	Code of data in packet	See detail
4	1	uint8_t	Number of bytes of data transmitting	N
5	N	N bytes	Payload data according to code of data field	
5+N	2	uint16_t	CRC-16 (see appendix)	

0x0001 that represents a packet with centimeter resolution. Table 3.2 gives a more detailed table of the packet which will be received on the FPGA.

Table 3.2: Streaming packet format for hedgehog coordinates in cm (code of data 0x0001)

Source: [20]

Offset	Size(bytes)	Type	Description	Value
0	1	uint8_t	Destination address	0xff
1	1	uint8_t	Type of packet	0x47
2	2	uint16_t	Code of data in packet	0x0001
4	1	uint8_t	Number of bytes of data transmitting	0x10
5	4	uint32_t	Timestamp: internal time of beacon ultrasound emission, in milliseconds from the moment of the latest wakeup event. See note.	
9	2	uint16_t	Coordinate X of beacon, cm	
11	2	uint16_t	Coordinate Y of beacon, cm	
13	2	uint16_t	Coordinate Z, height of beacon, cm	
15	1	uint8_t	Byte of flags: Bit 0: 1 - coordinates unavailable. Data from fields X,Y,Z should not be used. Bit 1: timestamp units indicator (see note) Bit 2: 1 - user button is pushed (V5.23+) Bit 3: 1 - data are available for uploading to user device, see section 2 (V5.34+) Bit 4: 1 - want to download data from user device, see section 2 (V5.34+) Bit 5: 1 – second user button is pushed (V5.74+) Bit 6: 1 – data for another hedgehog (not same one that sending this packet) Bit 7: – reserved (0)	
16	1	uint8_t	Address of hedgehog	
17	2	uint16_t	Bit 0. . . 11: orientation of hedgehogs pair in XY plane, decidegrees (0. . . 3600) Bit 12: 1 – coordinates are given for center of beacons pair; 0 – coordinates for specified beacon Bit 13. . . 15: reserved (0)	
19	2	uint16_t	Time passed from ultrasound emission to current time, milliseconds (V5.88+)	
21	2	uint16_t	CRC-16 (see appendix)	

Marvelmind Verilog

Starting from the dataset, a design has been created to decode the 0x0001 package. An overview of the inputs and outputs is given in Figure 3.5. The DV signal indicates to the marvelmind module when a byte is ready to read in the UART module. The UART module is briefly discussed in Section 3.2.1. After the full packet has been parsed, the "updated" output signal goes high and indicates that the X, Y and Z coordinates are available at the output.

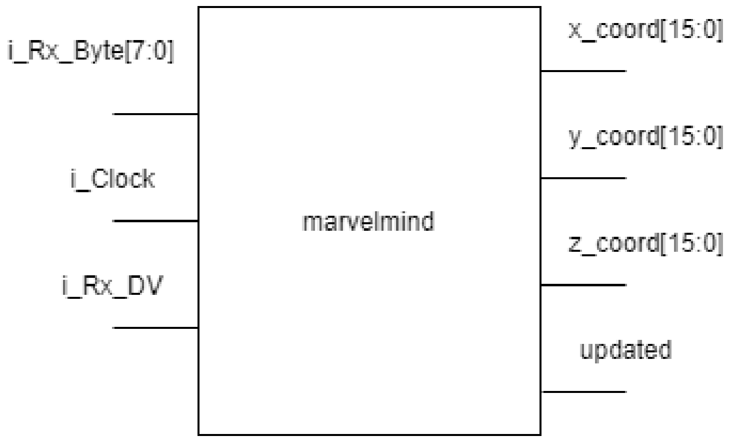


Figure 3.4: Marvelmind module block representation

The functions of the Verilog code are provided schematically in Figure 3.5. The marvelmind module keeps every correct incoming byte in a serial_buffer. The serial_buffer_offset keeps the number of the current incoming byte. If no correct bytes are received, no packet_received will be signaled and the buffer will start again from zero. Note that there is no implementation of cyclic redundancy check (CRC).

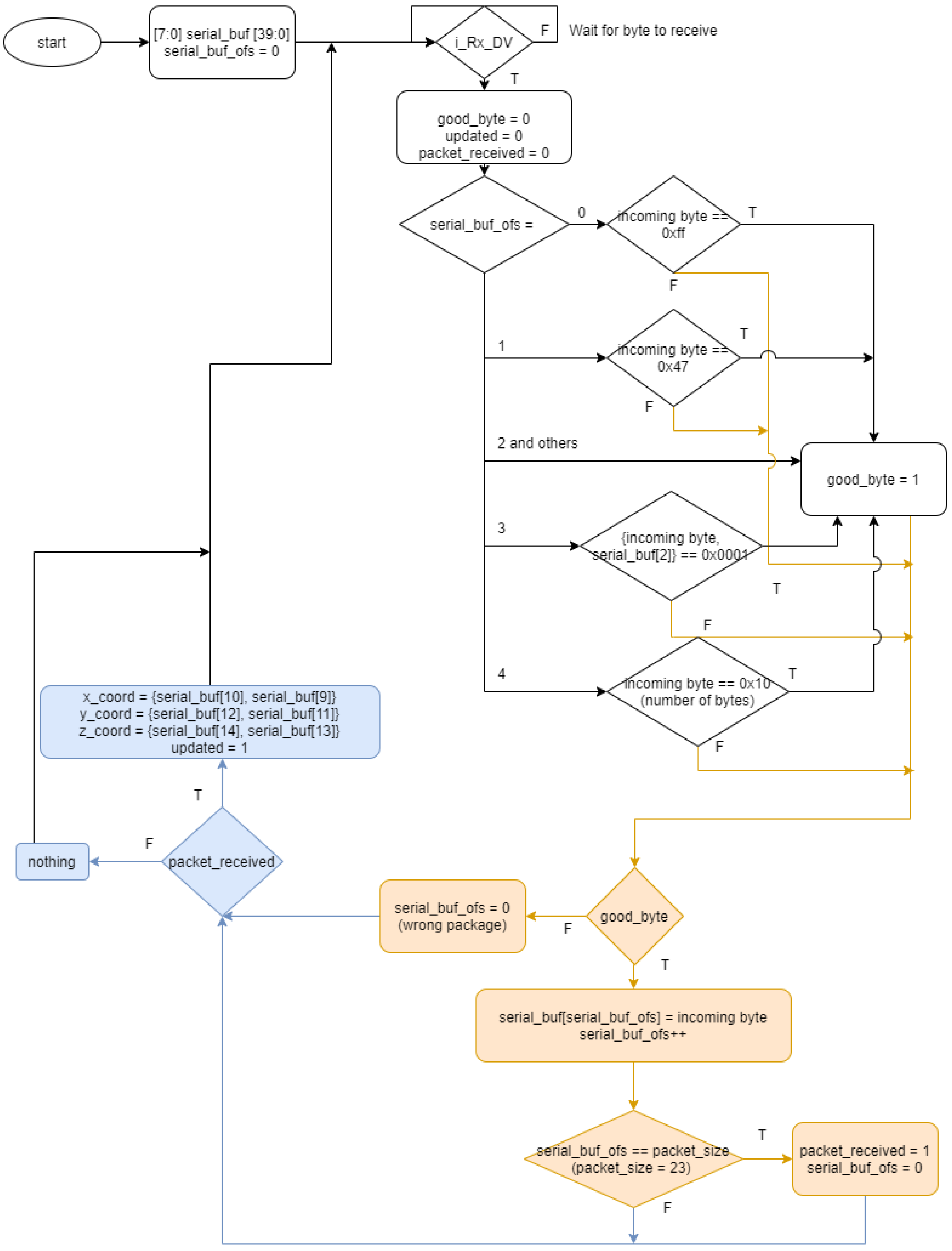


Figure 3.5: Marvelmind module decision chart

3.1.3 Pan and tilt calculation

The pan and tilt module converts the coordinates from the marvelmind module to two corners that can be used to control the spotlight. The first step in creating this module is to implement CORDIC. This will be done as explained as explained in Section 2.2.

CORDIC

The CORDIC module is based on an existing module from Kirk Weedman [12]. However, some adjustments are made to convert this to an arc tangent function by applying the equations of (2.12). The module can be divided into 3 parts. The first part is the instantiation of the lookup table, which stores the arc tangent values of 2^{-i}. The table has 31 values and thus goes to an i of 30. This means that the maximum of iteration that can be performed is 31. A choice has been made to use the coordinates in the form of a 16 bit signed value. This means that only 16 iterations are used, this is enough accuracy for the angles.

The second part is the pre-rotation of the corners. If an angle is not in the domain of $-90° \leq \theta \leq 90°$, the algorithm will not be able to calculate it. For angles that lie in the 2nd and 3rd quadrant of the coordinate system, a rotation must take place to the 4th and 1st quadrant, respectively. These conversions give the same tangent because the trigonometry shows that a 180 degree shift has no influence on the tangent. This can also be found in the Verilog code where the characters of the x and y coordinate are evaluated and rotated appropriately. For example, an angle in the 2nd quadrant will be converted to x = -x and y = -y and the current angle will be incremented by 180 degrees.

The third part is to run the algorithm itself. This has already been explained and implements the algorithm of (2.12) in Verilog. A repetition of this is given again in (3.1).

$$\begin{aligned} X_{i+1} &= [X_i - Y_i d_i 2^{-i}] \\ Y_{i+1} &= [Y_i + X_i d_i 2^{-i}] \\ Angle_{i+1} &= Angle_i - d_i tan^{-1}(2^{-i}) \end{aligned} \tag{3.1}$$

$$with\ d_i = 1\ when\ Y\ is\ negative\ and\ d_i = -1\ when\ Y\ is\ Positive$$

The current module implements the 16 iterations in a pipeline. This offers the possibility to obtain a high throughput. However, this is not necessary for the current application. This is still susceptible to possible adjustments to make the design more compact.

Pan and tilt

The inputs and outputs of the tilt_pan module are given in 3.6. A clock comes in along with x, y, z and an enable signal. The enable signal is to indicate when a set of new coordinates are ready. This signal will come from the marvelmind module. The output signal "ready" will go low during the process of calculating the angles. When this gets high again, it will indicate that the angles are ready at the output and that a set of new coordinates can be entered again.

With the implementation of CORDIC, it is possible to pan and tilt the calculate of each set of coordinates. There are a few methods to approach this. A pan angle is first calculated by means

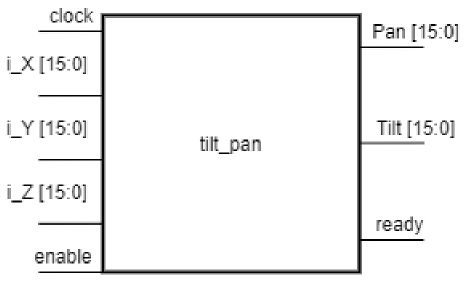


Figure 3.6: Pan and tilt module block representation

of the x and y coordinates.

$$Pan = arctan(y_1/x_1) \tag{3.2}$$

Then, the Euclidean distance d from the target in the XY ground plane is calculated.

$$d = \sqrt{x^2 + y^2} \tag{3.3}$$

This defines a vertical two-dimensional plane along with the z coordinate. The tilt angle is calculated by taking the CORDIC arc tangent of the z coordinate with respect to the distance equation (3.3).

$$Tilt = arctan(z_1/d) \tag{3.4}$$

The working principle of the pan and tilt is shown in figure 3.7.

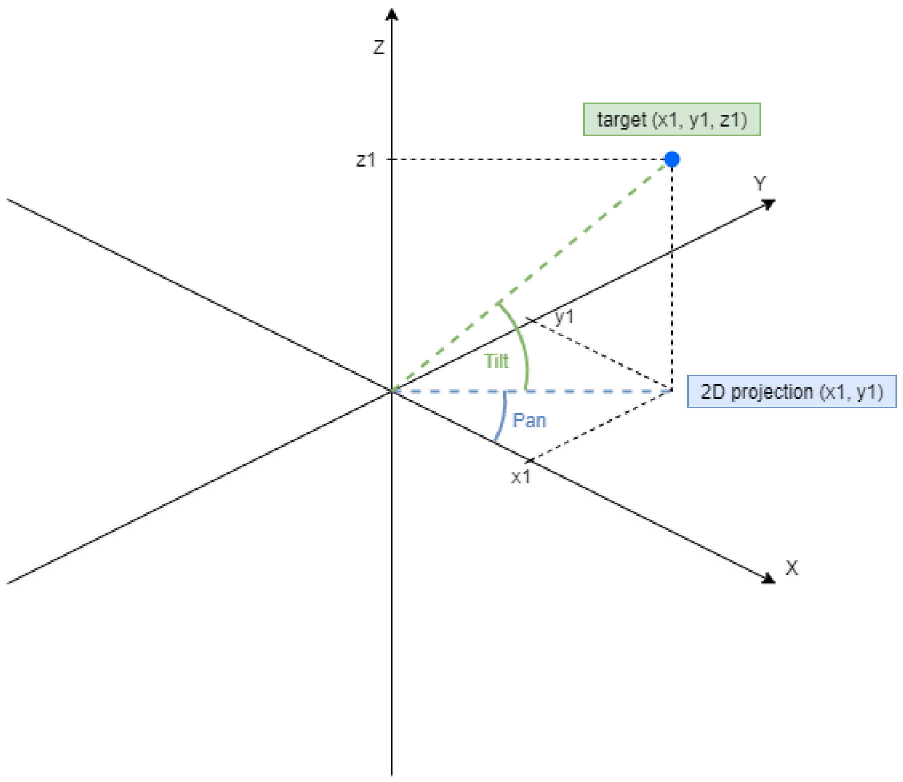


Figure 3.7: Working principle of calculating the 3D pan and tilt

There is one more important note about the implementation of the Euclidean distance. It is not possible to immediately use a square root in Verilog. Nevertheless, there is a way to calculate square roots anyway by using a variant of CORDIC. Basic knowledge has been obtained by adapting and designing a CORDIC module, but in order to work in a time-efficient manner, a CORDIC IP (= preconfigured function) module from Vivado Design Suite has been used. This IP module can then be constructed as required for the pan and tilt calculation. With the current implementation, the output of the CORDIC square root function has a 16 bit unsigned number. However, the CORDIC arc tangent module must receive a 16 bit signed number. To achieve this conversion the number that serves as input for the arc tangent, the first bit of this number becomes 0 and the rest of the 15 bits [14: 0] are the bits of the square root function's output.

This means that the Euclidean distance may be a maximum of 2^{15} in order to display a correct tilt. To calculate a limit of the x and y coordinates, they become equivalent in equation (3.3). This can be written mathematically as:

$$\begin{aligned} 2^{15} = 32768 &= \sqrt{x^2 + x^2} \\ x &= \frac{32768}{\sqrt{2}} \\ x &\approx \pm 23170 \end{aligned} \tag{3.5}$$

Here, it is concluded that the distances should not exceed 23 170. This is more than sufficient since this means a maximum distance of 231 m because the resolution is in centimeters. In order to have a simple limit on this during the test phase, it is ensured that the distances do not exceed 14 bits (2^{14} = 16 384).

Conversion of pan and tilt angle

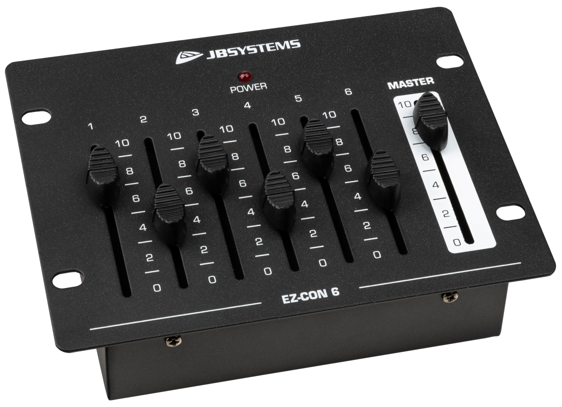
The last step in the tilt_pan module ensures a conversion between the calculated angle and the angle expected by the spotlight. In section 3.1.6 is a more detailed description provided of how a spotlight is controlled. The bottom line is that each type of rotating spotlight defines its own maximum rotation. The pan and tilt are calculated so that when all 16 bits are high, a full angle of 360 degrees is obtained. This means that the 360 degrees has a resolution of 65 536. The smallest possible step is then 0.005 degrees.

Most spotlights on the market often map these 16 bit to an angle different from 360 degrees. For example, there appear to be many spotlights that will map the pan at 540 degrees. If this is to be taken into account in the module then 360 degrees must be mapped over not 65536 possible angles, but at $65536 \times \frac{2}{3} = 43690$ possible angles. This is done by a read only memory (ROM). This ROM file is generated by a Python script that takes 2/3 of the incoming number to compress the 360 degrees to a resolution of 43 690. The ROM file is included when uploading the bitstream from Vivado to the FPGA to initialize the ROM memory. In a practical setup, first the settings of the moving head spotlight will be investigated. The ROM file is then generated depending on the spotlight settings. The current implementation simply converts both pan and tilt to 540 degrees. In the future there will be an adjustment depending on the available spotlight.

3.1.4 DMX controller

The DMX controller is the part of the system that generates the DMX signal. It is manually controlled by a spotlight operator to control the data on the slots. Each controller has its own properties (i.e. the amount of slots that be configured, XLR3/XLR5 connector and timing of the SPACE,MAB, ...) because these can vary between a minimum and maximum value.

The "JBSYSTEMS ez-con6" (see Figure 3.8a) is used as DMX controller as this is given by DSV-Rent. It contains a XLR3 connector to connect to other devices. Figure 3.8b displays the signal sent by the Ez-con6. The ez-con6 sends 33 slots (1 start slot and 32 data slots) but only the first 6 slots can be configured. This is done with the help of 7 sliders. One slider for each slot and 1 master slider that scales all slots, i.e. when the master slider is completely closed the range is 0-0 and when completely open 0-255. The individual ranges are scaled linearly). The rest of the slot contains 0x00 as data.



(a) Frontview of the DMX controller

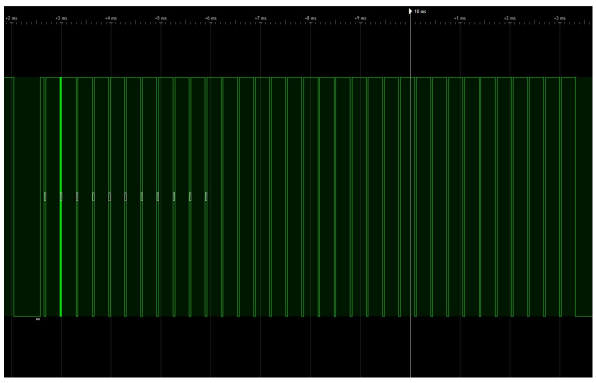

(b) Ez-con6 output DMX signal

Figure 3.8: JBSystems Ez-con6

3.1.5 DMX changer

The purpose of the DMX changer is to take the signal created by the DMX controller and pasted the pan and tilt onto specific slots without interfering with the values of other slots. Pasting the data onto an already created DMX signal is preferred because then the spotlight operator can then still use the controller to control values of the other slots for other effects (like the spotlight colour of the spotlight). There are multiple ways to implement this. The following two options are considered: 1) measure, edit and create a new signal, and 2) paste values onto the slots of an already existing DMX signal.

Measure, edit and create a new DMX signal

A DMX reader is designed to obtain all the values of the DMX Controller signal. The data is then saved into a buffer. Once the universe is done, the pan and tilt are saved in specific places in the buffer. Thereafter, a completely new DMX signal is created from the buffer data. Then, this signal is sent to the moving head for configuring the pan and tilt. This method requires to read a DMX signal and also to create a new DMX signal. This new DMX signal must be in sync with DMX controller otherwise data is lost. This can be solved by taking a specific point on the DMX controller and always start sending on that specific point. The end of SPACE could be such

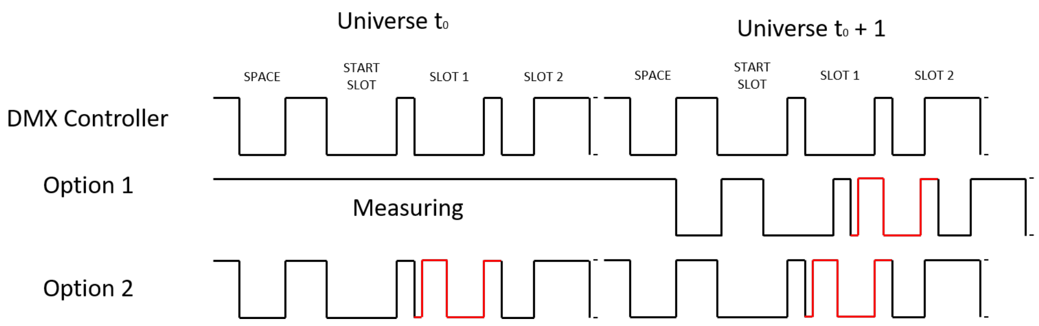


Figure 3.9: Visualization of first two slots by different options

point. The amount of slots sent by the DMX controller must also be measured. The minimum values of the timings are taken. If the minimum timings are taken and the same amount of slots are send as the input DMX signal then the time span of the output DMX signal is always less then the input. Therefore, the output signal is always completely send and no data is lost by sending a new universe before the previous universe is done.

Paste onto slots of an already existing DMX signal

The other option is to just paste new values on the slots of the DMX signal. This can be seen as a mux that had the option to choose between the controller DMX slot or a custom-made slot. The advantages of this option are that synchronization of the universe is not important anymore but instead synchronization of the slot is important. However, if the start bit and the position of the slot in the universe can be detected, values can be added into a specific slot. Another advantage is that the values are added on the DMX signal of the DMX controller, therefore the only delay present is due to measuring and transmitting the signal. A disadvantage is that a value can be pasted on the SPACE. This option can only detect a SPACE when the SPACE is done. Therefore, it can see the first part of the SPACE as slot and therefore a value can appear if the DMX changer is configured to paste a value on that slot. The amount of slots must be known upfront to prevent pasting data on the SPACE. A bit can also have an error. For this reason, a slot created by the DMX changer must have less bit time span then the DMX controller bit time span.

The second option is chosen because this is easier to implement. This design did not need a DMX reader. It instead required a slot counter and slot start detector. The synchronization could also happen after each slot instead of each universe which results in less problems with synchronization.

Design in Verilog

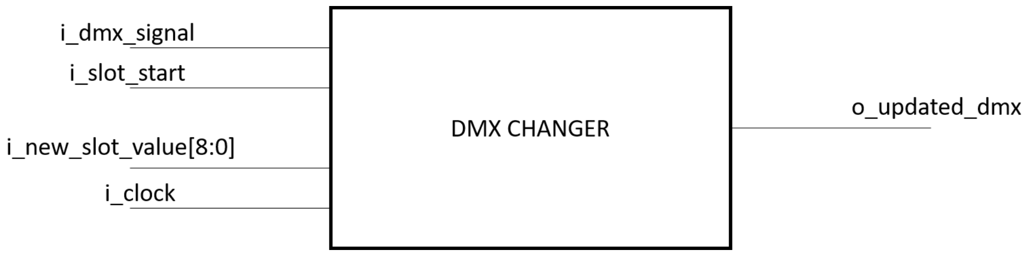


Figure 3.10: DMX changer module block representation

Figure 3.10 represents the block module for the DMX changer. The inputs of this module are: the current DMX signal, slot start, the new value for the current slot, current slot which is on the DMX slot and the clock signal. The output is an updated DMX signal with values pasted on the slots which need new values.

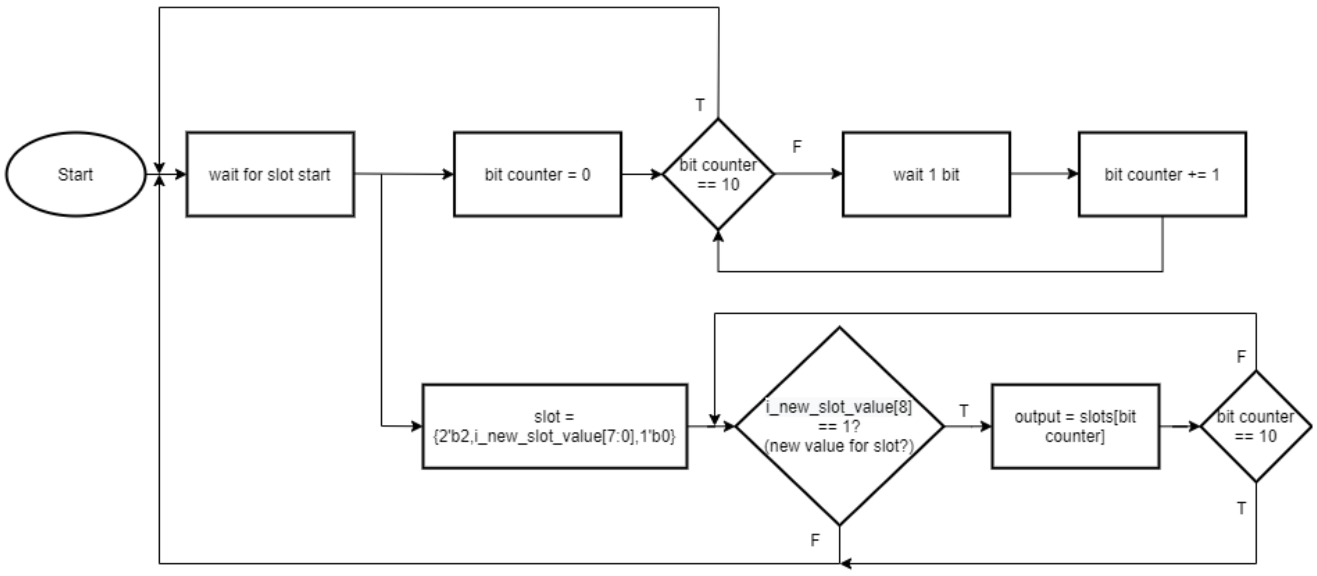

Start
wait for slot start
bit counter = 0
bit counter == 10
T
F
wait 1 bit
bit counter += 1
slot = {2'b2,i_new_slot_value[7:0],1'b0}
i_new_slot_value[8] == 1? (new value for slot?)
T
F
output = slots[bit counter]
bit counter == 10
F
T

Figure 3.11: DMX changer program flow

Figure 3.11 shows the flow of the module made in Verilog. The module starts two processes once i_slot_start is high. The first process controls the bit counter. This bit counter decides which bit to take from the new slot. After the start, the bit counter is set to zero. Then, the process checks if the bit counter is equal to 10, this means that the last stop bit is being displayed. If the bit counter is 10, then it starts waiting for a new slot start. If the bit counter is not equal to 10, then it waits one bit. Next, the process increments the bit counter and go back checking if the bit counter is equal to 10. This is repeated until bit counter is 10. While the first process is controlling the bit counter, the second process decides what to show. It starts by checking if their is a new value to add onto the slot. It does this by checking the most significant bit of i_new_slot_value_. If this bit is 0, nothing changes on the input DMX signal and the updated DMX signal is the unmodified input DMX signal. Thereafter, it goes back checking for start slots. If the most significant bit of i_new_slot_value_is 1, then it creates a new slot with the 8 least significant bits. These bits represent the new data for slots. The new slot contains following bits: The two most significant bits are 1 representing the stop bits. The next 8 bits represent the data bits. Finally, the least significant bit is a zero bit, representing the start bit of the next slot. After creating the slot, it takes the correct value out of the slot with the help of the bit counter. Thereafter, it checks if the bit counter is equal to 10. If the bit counter is not equal to 10 then it goes back to take the correct value for the updated DMX. If the bit counter is 10, then it assigns the input DMX to output DMX. Finally, the second process goes back to wait for a slot start as process one.

3.1.6 Moving-head spotlight

There are several spotlights with a rotating head in 2 directions, namely pan and tilt. Communication with the spotlight is done via DMX. DMX has 512 channels over which data transfer is done. Such a channel then has 8 bits to communicate, this is a value from 0 to 255. A channel then often corresponds to a specific instruction that the lamp can execute. All these specifications can be found in the datasheet of the particular device. The ADJ Vizi BSW 300 can be taken as an example [21]. The data sheet indicates that a maximum of 20 channels can be used to control the device. Channel 1 corresponds to the pan movement. Channel 2 controls the fine movement. For tilt this is the same on channels 3 and 4. Then, the other 16 channels are still used to control the color wheel, shutter settings, dimmer, etc. Based on the data sheet, the first channel can be determined as the 8 most significant bits and the second channel represents the 8 least significant bits of the spotlight. These 16 bits can achieve a maximum pan rotation of 540 degrees. For tilt rotation, there is a maximum of 240 degrees according to the data sheet. This is also the reason for the conversion of the 16 bits at 360 degrees into the desired angle in the pan and tilt module. The remainding channels should enter and exit the FPGA unchanged.

3.2 Communication

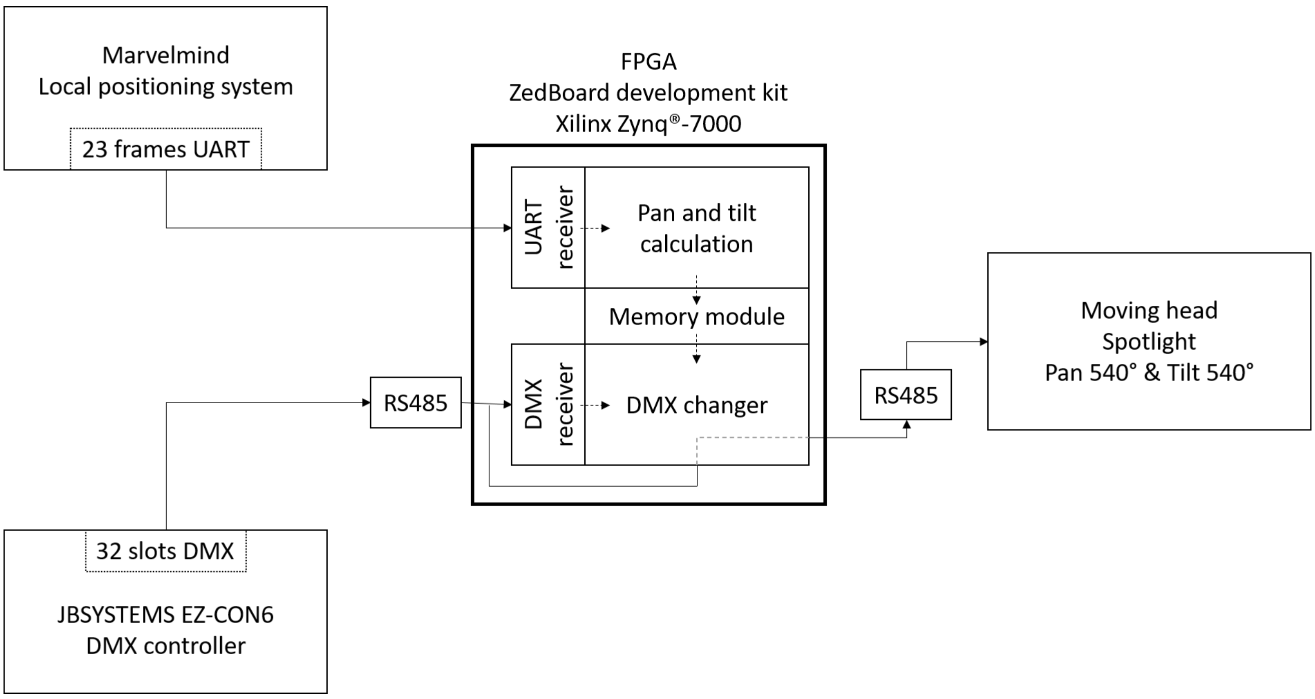


Figure 3.12: Communication between parts

Previous chapter explained how each of these parts worked. This section goes more in depth of how these parts communicate with each other. More particular the UART communication between the marvelminds and the FPGA. The DMX communication reader on the FPGA for data input for the DMX changer. Followed by, the memory module to communicate data from the pan and tilt module to the DMX changer. Lastly, the RS485 protocol to go from a 1 wire data signal to a DMX signal is explained.

3.2.1 UART

UART is a well-known protocol of which many examples of HDL implementation are available online. UART is also available as an Intellectual Property (IP). IP is a preconfigured logic function that can be used. For Vivado Design Suite, there are 2 UART IP's available, namely AXI UART LITE and AXI UART 16550. However, public HDL code from Nandland [22] is used because of the clear and simple explanation of its Verilog design.

The UART module is designed to receive 8 data bits with one start bit and no parity bit. The module has 2 input and 2 output signals as shown in Figure 3.13. An input signal i_clock and i _RX _Serial for this received data. The output signal o_Rx_DV shows when a correct bit has been received (DV = data valid) by becoming high for a duration of 1 clock period. The received bit is then ready to be read in parallel on o_Rx_Byte [7:0].

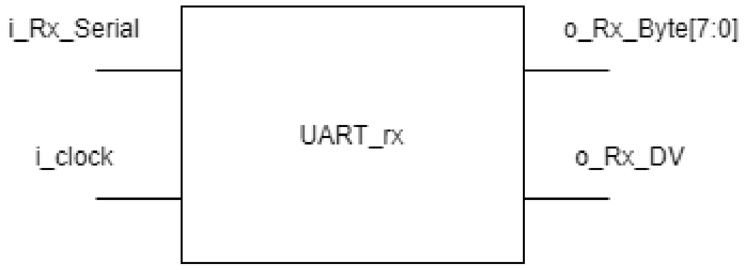


Figure 3.13: UART block representation

The module also provides the ability to work with different baud rates and clock rates by passing a parameter to the module, namely CLKS_PER_BIT. The parameter is essentially to calculate how many clock signals are required to sample 1 UART bit. This parameter can be calculated depending on the frequency of the clock and the baud rate of the UART signal.

$$\text{CLKS\_PER\_BIT} = \frac{\text{clock frequency}}{\text{baud rate UART}} \tag{3.6}$$

In short, the module will read in the signal by means of a state machine. There are 5 states: idle, start, data, stop and cleanup. The idle state continuously detects whether a start bit has been detected and then switches to the start state. The start state will take a sample in half of the start bit to confirm the bit's validity. Next, it is switched to the data state and each of the 8 bits is sampled. As the penultimate state, 1 bit period will be waited to represent the stop bit and the DV signal indicating that a byte is ready will be raised. The cleanup state will reset the DV signal to low and bring the idle state back up. The cycle continues.

3.2.2 Memory module

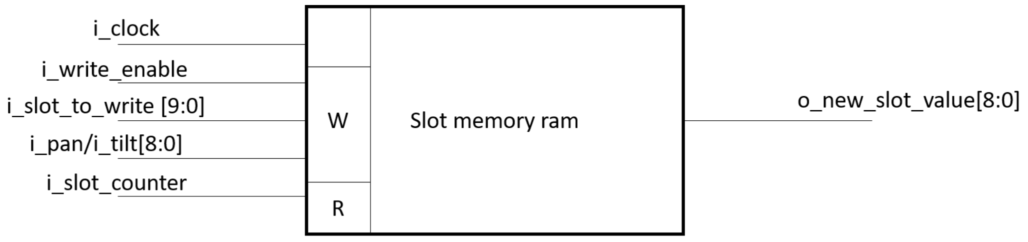


Figure 3.14: Memory module block representation

The communication between pan and tilt calculation and DMX changer is done with a simple dual port Ram module from the FPGA. This module contains two ports. One port for writing data and one port for reading data as can be seen in Figure 3.14 The pan and tilt is connected to the writing port and the DMX changer is connected to the reading port. It has a depth of 513. Each depth represents 1 slot (1 START slot + 512 data slots). A slot has a width of 9 bits. The 8 least significant bits are the data bits. The most significant bit is used to let the DMX changer know if the value needs to be added onto the DMX signal. The DMX changer adds the data onto DMX signal if this is bit is 1. Otherwise, it takes the input DMX signal, even if there is data onto the other bits.

Pan and tilt saver

The angles are sent to the slot memory ram at the start of the first data slot (slot 1) if the pan and tilt are ready. Otherwise, it has to wait for the pan and tilt to be ready to sent the values. Waiting for pan and tilt to be ready, can be done because the amount of clock cycli needed to calculate the pan and tilt is always less than the amount of clock cycli needed for the start bit to pass. This procedure is added at the final top level module to assign the pan and tilt values to the memory module in a fitting manner. It uses 3 signals for this: a write enable, 8 data bits and the designated slot.

The angles are written by assigning the appropriate 8 bits to each predetermined slot. In the current implementation bits [15:8] of the pan angle are placed into slot 1. Bits [7:0] for fine tuning of the pan angle are placed in slot 2. The same is done for the tilt angle on slot 3 and 4.

3.2.3 DMX receiver

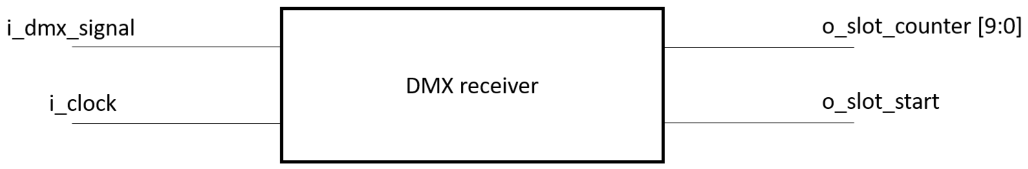


Figure 3.15: DMX changer module block representation

The DMX receiver measures the incoming DMX signal for slot starts and the current slot. It communicates when a slot starts to the DMX changer. The current slot is communicated to the memory file.

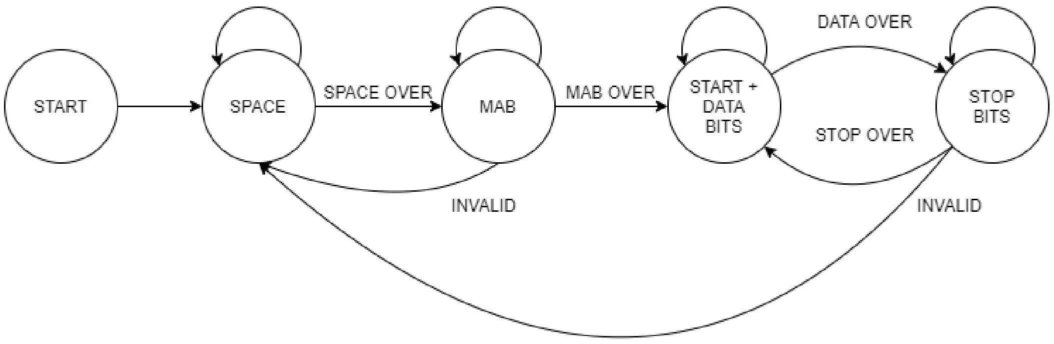


Figure 3.16: State machine of the receiver module

Figure 3.16 displays the implementation of the receiver module as a state machine. The DMX receiver starts and enters the SPACE state. Here, it waits until the SPACE is over. It does this by creating a counter that checks if the input DMX signal is zero for the minimum SPACE time duration (see table 2.5 for minimum duration of the DMX signal). If this is correct, it goes from SPACE to MAB. The MAB OVER is the same as the SPACE OVER but instead of waiting for the minimum SPACE TIME, it checks for the minimum MAB TIME. A DMX signal is 1 for the MAB and therefore the module checks if the input DMX signal is 1 during the minimum MAB TIME. Thereafter, the module enters the START + DATA BITS state if the signal is 1 for the minmum MAB Time. Otherwise, the module enters the SPACE state because of an INVALID.

The data bits can be 0/1 and therefore the module waits until the minimum START + DATA BITS duration is passed. Then, it enters the STOP BITS state. In the STOP BITS state, it checks if the input DMX gets 1 before the maximum START + DATA BITS duration is passed. If the input DMX signal does not get 1 in time, then it means that stop bits are not coming. This results into an INVALID. In this situation, the module goes back into the SPACE state to repeat the process. If the stop bits get 1 in time, then it checks for the minimum STOP TIME duration. The module enters the SPACE if the duration is less then the minimum STOP TIME duration. If the duration of the STOP BITS are more then the minimum STOP BITS duration, it goes back to START + DATA BITS to measure the next slot. While going back to START + DATA BITS state, it sets the output slot start to 1 and increments the output slot counter. The output slot counter is reset to 0 when the module enters the SPACE state from the STOP BITS state.

3.2.4 RS485

RS-485 uses two signal wires to send binary data over a cable interface. It uses 2 complementary voltage levels to act as an differential signal that is immune to electrical noise. EIA-485 defines the voltage levels to be between ± 1.5V and ± 5V [23]. Because this specification is within the domain of the DMX, it is possible to use a MAX485 chip [24] to control DMX. The MAX485 allows to transfer a signal from the FPGA to a XLR cable which requires a differential signal and vice versa.

3.3 Simulation

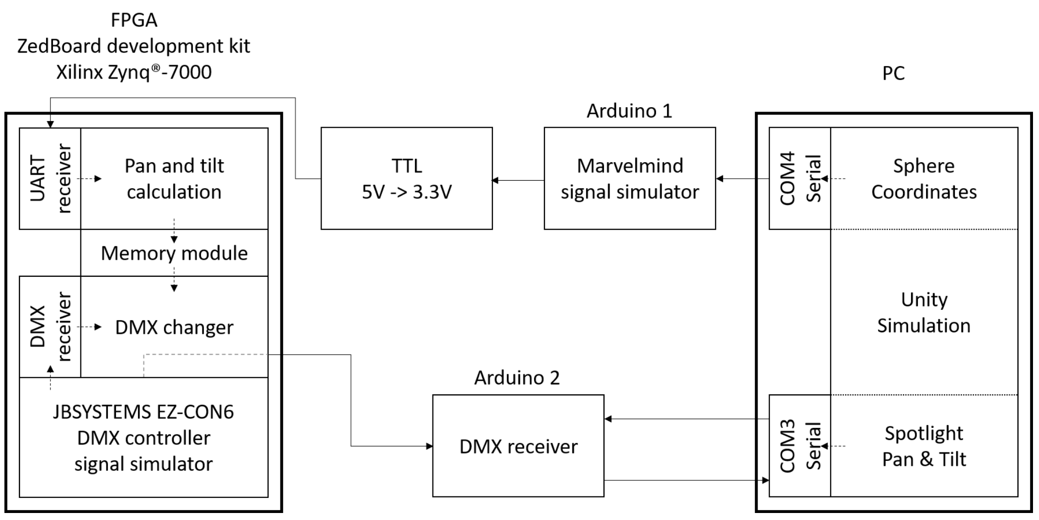


Figure 3.17: Block diagram of the simulation

A simulation is made to test the design on the FPGA. Unity is used to simulate the person and moving head spotlight. The marvelminds are replaced by an Arduino that simulates the interface of the marvelminds. This Arduino receives coordinates from the Unity simulation. A Transistor–Transistor Logic (TTL) is added to connect the Arduino with the FPGA. A new module is created on the FPGA to simulate DMX signal from the DMX controller. A second Arduino is added to measure the DMX signal received from the FPGA. This Arduino sends the pan and tilt values measured on the updated DMX signal to Unity. Unity will then adjust the moving head spotlight in the simulation environment.

3.3.1 Unity

Unity is a real-time 3D development platform. It is best known for being a tool to develop games but with some creativity it can also be used as simulation for a person and moving head. Unity personal 2019.3.12f1 version is used because this is a free version for individual use. Unity works with scenes. Many objects, e.g. 3D objects and light sources, can be added to such scene. These objects can be used to simulate a person and moving head spotlight. A sphere is used as person, a spotlight light source is used as moving head spotlight. The sphere is controlled with the use of the arrow keys, the SPACE to move up and the Z key to move down. This simulates the walking behavior of a person. The spotlight is placed on the origin of the scene. By doing this the coordinates is always relative to spotlight. Unity also has C# implemented as scripting language. This allows sending and receiving data in Unity from the Arduinos with the use of a C# serial library. Unity has a main thread where everything graphic is updated. In this thread, the pan and tilt is updated. Unity does not allow to access 3D objects in other thread outside the main thread. Therefore, the coordinates are saved into global variables. A new thread is created to handle interaction with the Arduino. First, this thread sends a character to Arduino 2 with the use of UART over COM4. As such, Arduino 2 knows that it is ready to receive data. Next, it listens on serial COM4 for the pan and tilt sent by the Arduino 2 and save this values in global variables. Finally, it sends the coordinates of the sphere to Arduino 1 with the use of UART over COM3. This thread is always running in background when simulating the environment.

3.3.2 Arduino 1

Arduino 1 is used to send coordinates to the FPGA instead of the Marvelminds. It receives the coordinates from Unity. By using these coordinates, it simulates the same packet as a Marvelminds generates. The FPGA is connected on the Digital 1 pin/TX pin and the PC is connected with the USB connector. The Arduino receives data from the PC over the serial and sends data to the FPGA. Both devices uses the same serial port of the Arduino. Therefore, the PC receives the data sent to the FPGA. This is solved by discarding the buffer on COM4 in Unity. The baudrate of the Marvelminds is 500 000. Therefore, the PC and Arduino also communicate on a baudrate of 500 000.

The coordinates send by the Marvelminds are `int_16` (see table 3.2). Therefore value ranging between -32768 and 32767 can be send as coordinates of the marvelmind packet. The two least significant bytes are taken of numbers with more then two bytes. The numbers are sent as character from the pc to the Arduino. A character comma is used to announce the end of a number. Characters other then numbers or minus signs are not used. The Arduino simulates a marvelmind packet each time it receives three numbers. The frames of the marvelmind packet with a standard value or coordinates are sent with a value. The other frames, like the address of hedgehog, contain no useful information for the UART receiver and are therefore send as zeros frames. The values from the first, second and third number are taken for respectively x,y, and z coordinates. Figure 3.18 describes the function of the Arduino as a flowchart.

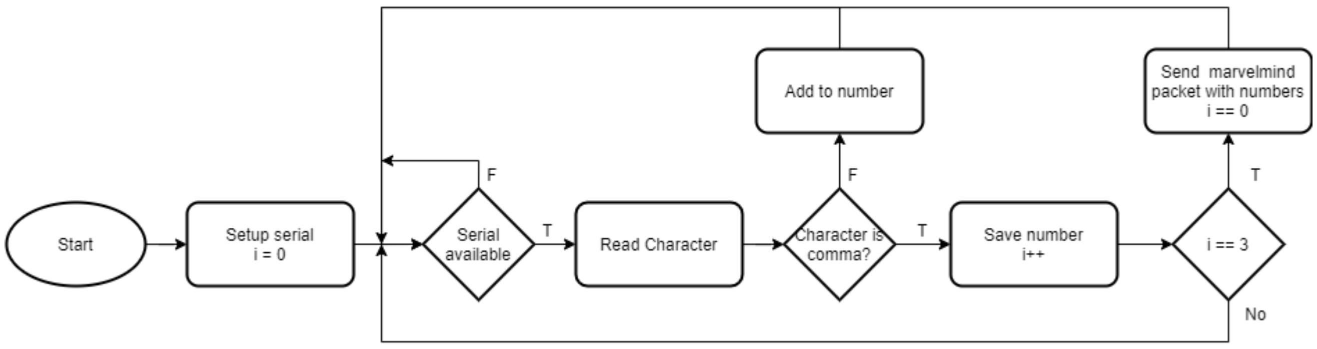


Figure 3.18: Flowchart Arduino 1 program

3.3.3 Arduino 2

Arduino 2 measures the DMX signal transmitted by the FPGA. The DMX signal is connected onto port 3. The PC is connected onto the USB Connector of the Arduino. A baudrate of 500000 is used to communicate between the PC and Arduino. The measurement of the DMX happens with a DMX library. This library implements a measure function that tries to read the first n slots of universe. The function returns a true if it could read the first n slots, otherwise it returns false. The first n slots are saved into variables. Only the first 6 slots are measured and saved into variables. Finally, the Arduino sends the values of the slots to the PC once it receives a character onto the serial bus.

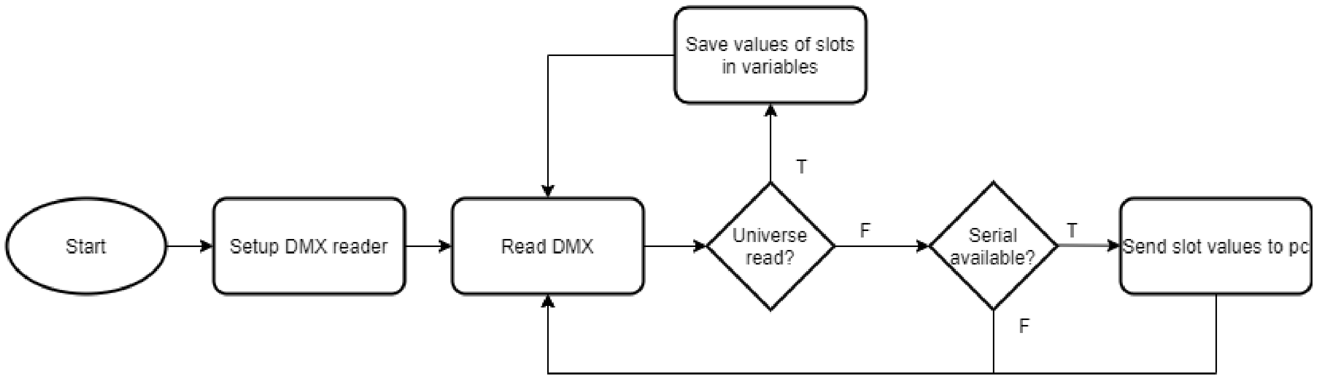


Figure 3.19: Flowchart Arduino 2 program

3.3.4 TTL

When the digital port of the Arduino uno is set high, a voltage of 5V DC is send over the cable. However, this voltage level is too high for the FPGA to receive. Therefore, a transistor transistor logic (TLL) is added to transform the 5V DC into 3.3V DC. A resistor of 1kΩ is added in series at the 3.3V output to reduce overshoot.

3.3.5 Ez-con6 simulation

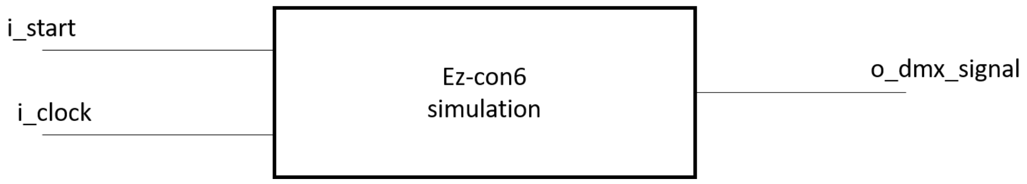


Figure 3.20: Ez-con6 module block representation

The JBSystem EZ-CON6 is integrated onto the FPGA for the simulation. Therefore a design is created on the FPGA that simulates JBSystem EZ-CON6. Figure 3.20 represents the EZ-con6 module. It has 2 inputs: an input start to start emitting the signal and clock signal. It outputs the DMX signal. This signal is then used for the DMX changer and DMX receiver as DMX signal input.

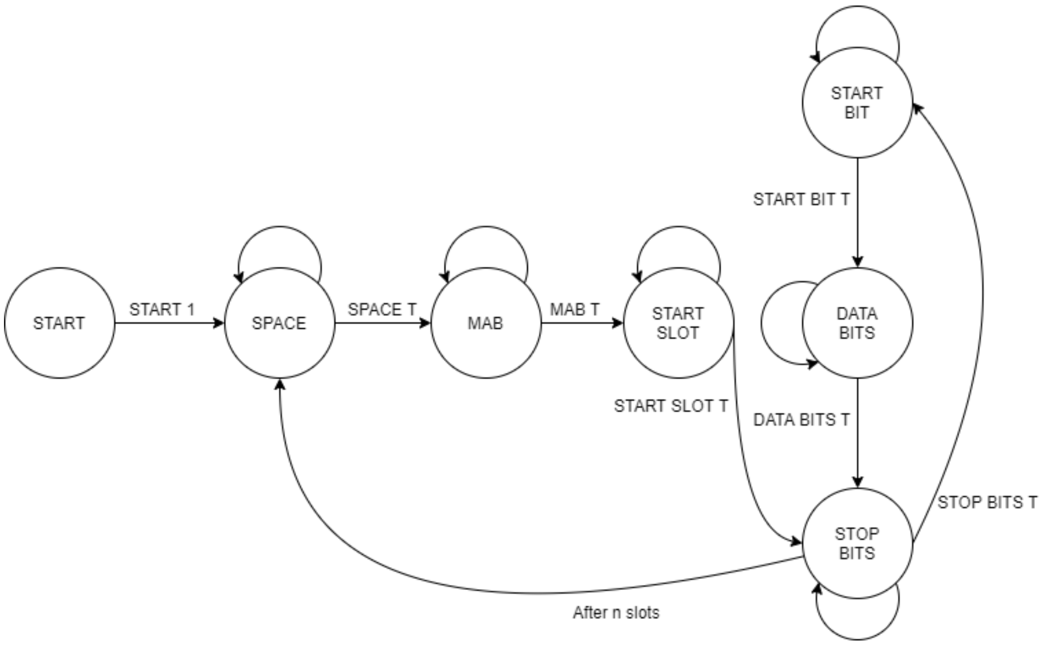


Figure 3.21: State machine implementation Ez-con6

Figure 3.21 describes the state flow of the module. It starts in START state, where it waits for start to become 1. After that it moves the SPACE state. After given time SPACE T it goes the MAB state where it stays until MAB T is passed. The module then enters the START SLOT state, the start bit of the start slot is included in the timing START SLOT T. Then it enters the STOP BITS state. The STOP BITS state wait for STOP BITS T time before going the START BIT. The STOP BITS T contains the time it takes for stop bits + time of the MBS. It stays in the START BIT for START BIT T. After that it enters the DATA BITS state, where it stays for DATA BITS T. Finally, it enters STOP BITS again.

The loop from START BIT to STOP BITS is repeated n times. Each time representing 1 slot. Therefore, N can maximum be 512, because this is the maximum amount of slots in a universe. If it reaches n + 1 it returns back to the SPACE state, where it repeats the whole cycle again. Going to the SPACE state can be seen as starting a new universe.

Table 3.3: Ez-con6 real vs. parameters taken for simulating

properties	Ez-con6	parameters simulation	output value
NULL START SLOT	True	True	/
DATA SLOT	32	32	/
SPACE	525 - 537μs	530μs	0
MAB	67.8 – 70.8μs	69.8μs	1
START SLOT	±35.960μs	36μs	0
START BIT	±3.90μs	4μs	0
DATA BITS	±31.70μs	32μs	0
STOP BITS + MBS	282-288μs	288μs	1
UNIVERSE	±11.3s	±11.3s	/

Table 3.3 displays the timings of the Ez-con6 and the timings taken for the simulation.The timing of SPACE, MAB and STOP BITS + MBS varies between 2 values by the real signal.Therefore, a value is chosen between ranges of the real SPACE and MAB for the simulated signal SPACE and MAB. For the STOP + MBS a timing of 288 μs is taken because this is the most occurring value by the real STOP + MBS. A 36 μs START + DATA BITS is taken for the simulated signal. This makes each bit 4 μs long, which is easier to work with. This is also recommended value by the standard (see table 2.4. All of the timing values taken for the simulated signal adhere to the DMX standard (see table 2.4).Table 3.3 also displays the output value for each state. A simple DMX signal with the right timing is wanted because the DMX changer will eventually write new values onto slots. Therefore, all the data bits are taken 0. This result into universe with all slot containing zero.

3.4 Measurement

Two tools are used to measure and check all the generated signals. The first one is a IKA logic analyzer SQ200 [25]. The second one is the Tektronix MSO4102B-L [26].

3.4.1 IKA logic analyzer SQ200

The IKA logic analyzer SQ200 can measure up to frequencies of 200 MHz. It can trigger on multiple voltage levels including 3,3V and 5V, which are the voltage levels of the FPGA and Arduino. The IKA logic analyser SQ200 can also measure 4 signals at the same time. ScanaStudio is the software tool used to interfere with IKA logic analyzer SQ200.

Figure 3.22: IKA logic analyzer SQ200

Source: [25]

3.4.2 Tektronix MSO4102B-L

This is an oscilloscope of the Tektronix MSO4000 series. It has two analog channels and a bandwidth of 1GHz. In contrast to the logic analyzer, analog signals can be measured. It is also more accurate with the ability to measure 5GS/s when a single channel is used. More info can be found in the datasheet of [27].

Chapter 4

Results

In this chapter the results of Chapter 3 are discussed. First, the design on the FPGA is tackled, showing the behavioral simulations of Vivado of the different modules. These simulations also briefly discuss what data is present on the I/O pins and what the module has processed. On top of that an overview of the module's use of the resources of the FPGA is included.

In addition to the behavioral simulation of the design, the results of the test setup are also discussed. This test setup combines the FPGA with Unity to visually present the person tracking system in simulation. This section will explain the operation of the two Arduinos, Unity and the communication between the devices. Finally, there is a small discussion that summarizes the achieved results and looks at the possibilities for the future.

4.1 FPGA design

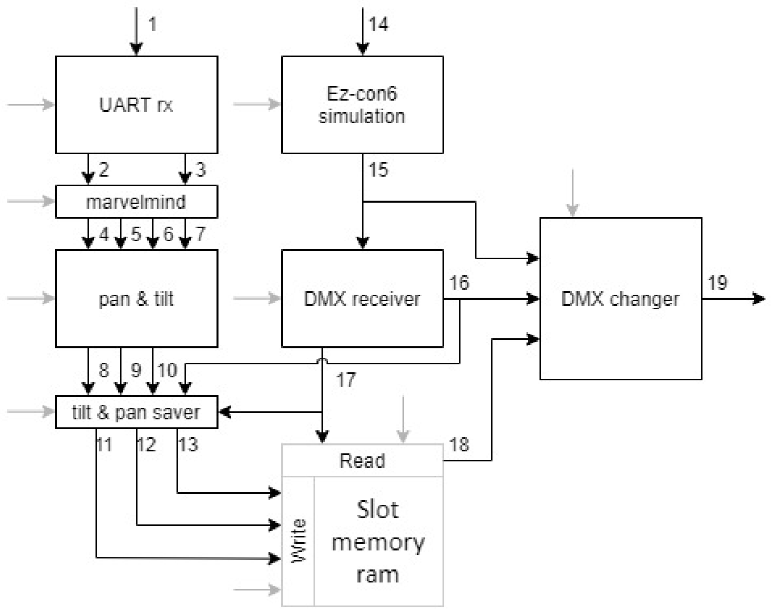


Figure 4.1: Final design FPGA design

Figure 4.1 describes the final result of the FPGA design for simulation. It shows how each part was connected with each other. The gray arrows represents the clock signal. The following sections go more in depth in the results of each module. The number between brackets after the signals displayed on the Figures match to the numbers on Figure 4.1.

4.1.1 UART rx

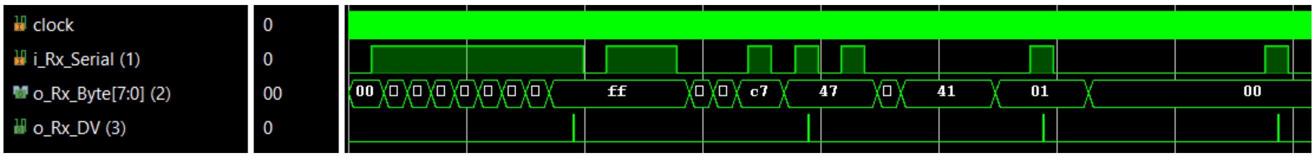


Figure 4.2: UART simulation

Figure 4.2 displays how outside serial data (1) entered the module and was processed to be accessed as a byte (2). If the byte was valid then the data valid signal (3) was high for 1 clock.

4.1.2 Marvelmind module

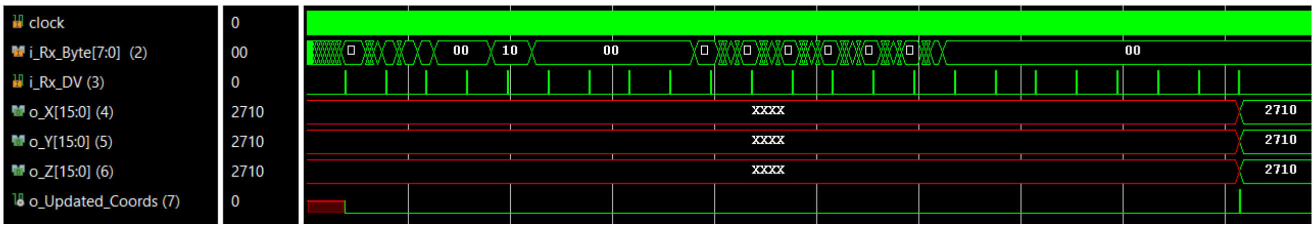


Figure 4.3: Marvelmind module simulation

The Marvelmind decoder module as seen on Figure 4.3 shows the processing of the UART module's bytes (2). The expected Marvelmind streaming packet size was 23 bytes. This can be found in the figure by counting the 23 pulses of DV (3). At the end of the packet stream, the coordinates were ready at the output (4-6). This was indicated by a high "updated" signal (7).

4.1.3 Tilt and pan

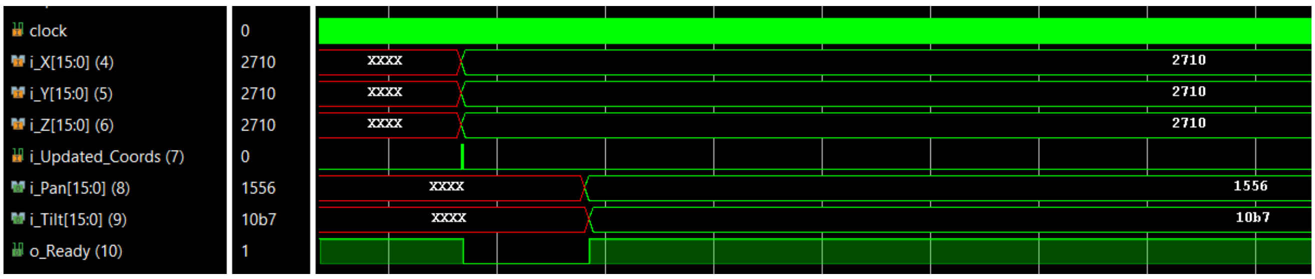


Figure 4.4: Tilt and pan simulation

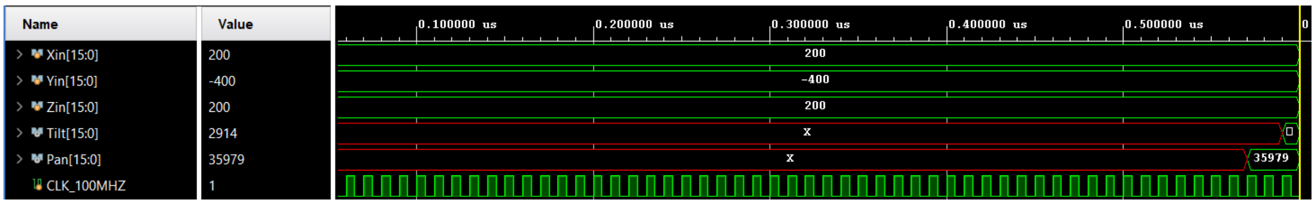
The input coordinates (4-6) were used as soon as updated was high (7). The calculation of the angles needed 58 clock cycles. After this, a ready signal (10) was used to indicate that the pan and tilt (8-9) were ready to be used. This ready signal was low when the module was busy with performing the calculation.

CORDIC

Calculating the pan and tilt angles was one of the most important tasks that the design had to perform. This module must return a reliable angle that did not deviate too much from the actual angle. Therefore, it was extensively tested. A test bench was made for this in Verilog. This test bench read a data_file_.dat file which contained the x, y and z coordinates. The corresponding calculated angles were stored afterwards in an output_data.txt file. Based on these results, it was concluded that the the angles were calculated correctly. In order to have a real visual confirmation, a simulation was designed. However, there was one important note that follows from testing. Small values only need a few bits. For example, a decimal value of 2 was going to have a binary value of 0000 0000 0000 0010. The CORDIC algorithm works by shifting right the intermediate results **i** times as shown in (3.1). **i** corresponds to the current iteration step.

Small numbers such as 0000 0000 0000 0010 had the disadvantage that multiple shifts had no effect because after a few shifts it was already zero. The angle that ultimately corresponded to this, had therefore a large deviation. This was not too big of a problem, because there was always work in centimeter resolution and the distances will practically always be relatively large because the spotlight is mounted above person height.To show a practical representation in simulation, the example of figure 4.16 (values: 200, -400,200) is given as input for the pan and tilt module.

The first column in Table 4.1 shows the obtained 16 bit values for the pan angle of the test bench in Figure 4.5. As previously explained, this angle had to be decreased $\frac{2}{3}$ times to adapt for a spotlight with 540 degrees of freedom over 16 bits. To convert this back to 360 degrees, in the 3rd column a multiplication of $\frac{3}{2}$ was performed. This was then divided by $2^{16} = 65536$ to map the angle between 0 and 1. Finally, this was multiplied by 360 degrees to show an intuitive result. The last 2 columns return the angles when coordinates were entered directly in an arc tangent function on a calculator.



Name	Value
Xin[15:0]	200
Yin[15:0]	-400
Zin[15:0]	200
Tilt[15:0]	2914
Pan[15:0]	35979
CLK_100MHZ	1

Figure 4.5: Screenshot test bench simulation 200, -400, 200

Table 4.1: Table of CORDIC pan and tilt result of 200, -400, 200

16 bit Pan (540°)	16 bit Tilt (540°)	$\frac{Pan \times \frac{3}{2}}{65536} \times 360$	$\frac{Tilt \times \frac{3}{2}}{65536} \times 360$	$atan(\frac{y}{x})$	$atan(\frac{z}{\sqrt{x^2+y^2}})$
35979	2914	296.46°	24.01°	$-63.43°$ or 296.57°	24.09°

4.1.4 EZ-con6 simulation

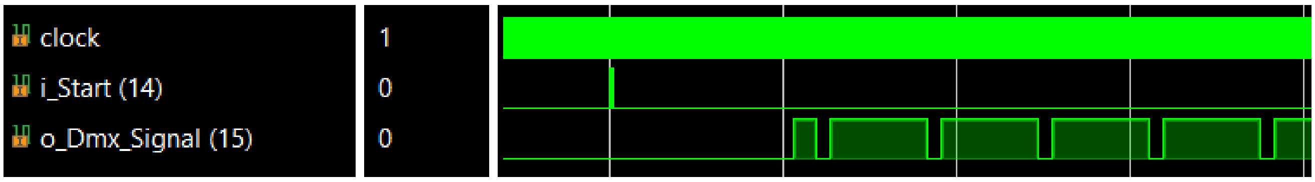


Figure 4.6: DMX receiver Ez-con6

Figure 4.6 illustrates the Ez-con6 simulation. The start signal (14) was controlled by a button on the FPGA. If this button was pressed, the signal turned high and the module started sending the DMX signal (15). The time between the start high and the first high of the DMX signal was the SPACE. Next, the MAB is shown with thereafter the null start slot and data slots with MBS.

Simulated vs. real

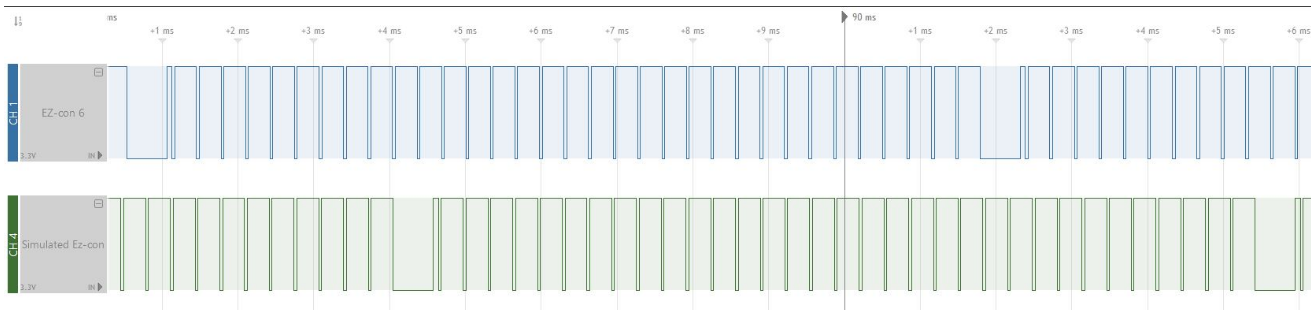


Figure 4.7: Output input arduino 2

Figure 4.7 illustrates a DMX signal generated by the Ez-con6 and FPGA. The blue signal was the DMX signal generated by the Ez-con6. The green signal was the DMX signal generated on the FPGA. Taking the timings as discussed in 3.1.5 resulted therefore into 2 fairly similar signals.

4.1.5 DMX receiver

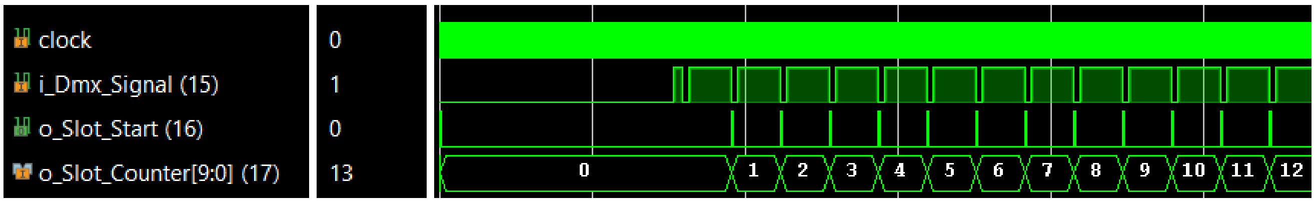


Figure 4.8: DMX receiver simulation

Figure 4.8 shows the simulation of the DMX receiver. The slot start (15) gave a pulse for each data slot start of the input DMX signal. The slot counter (17) was also simultaneous updated with with each data slot start.

4.1.6 Slot memory ram

The slot memory has a read and write part. Writing was in the order of clock cycle while reading was in the order of data slots. Therefore, both are shown separately.

Write

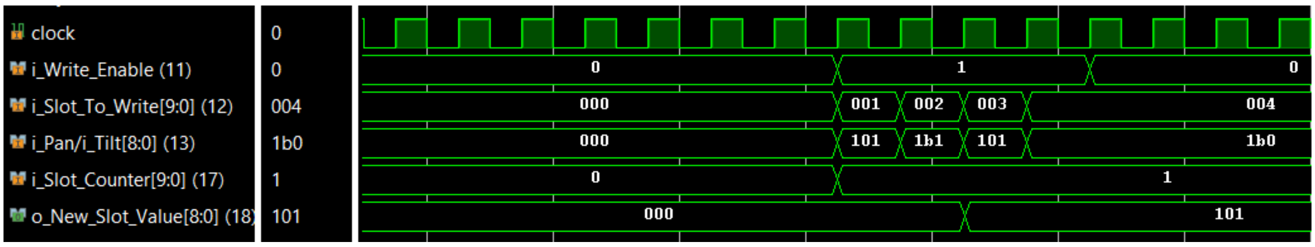

clock 0
i_Write_Enable (11) 0
i_Slot_To_Write[9:0] (12) 004
i_Pan/i_Tilt[8:0] (13) 1b0
i_Slot_Counter[9:0] (17) 1
o_New_Slot_Value[8:0] (18) 101
0
1
0
000
001
002
003
004
000
101
1b1
101
1b0
0
1
000
101

Figure 4.9: Slot memory write simulation

Figure 4.9 shows the writing part of the slot memory. The slots (12) were written with the pan and tilt values (13) once the write enable (11) was set to high. It took 2 clock cycli to write and then read the same value out of that slot.

Read

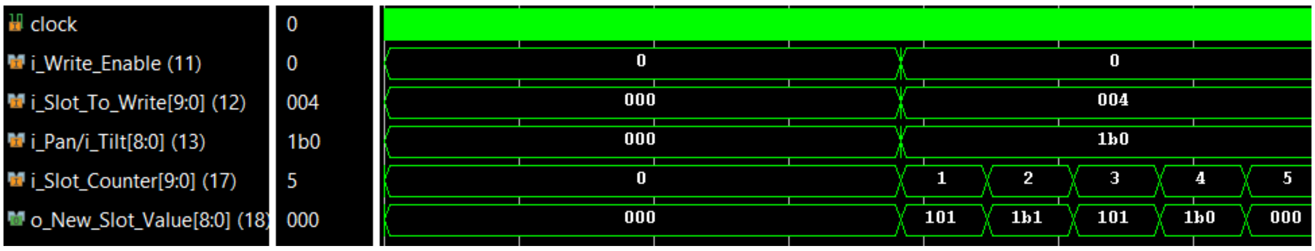

clock 0
i_Write_Enable (11) 0
i_Slot_To_Write[9:0] (12) 004
i_Pan/i_Tilt[8:0] (13) 1b0
i_Slot_Counter[9:0] (17) 5
o_New_Slot_Value[8:0] (18) 000
0
0
000
004
000
1b0
0
1
2
3
4
5
000
101
1b1
101
1b0
000

Figure 4.10: Slot memory read simulation

Figure 4.10 displays the reading part of the slot memory. The new values for the slots (18) were loaded for each slot (17). These values contained 9 bits. The first bit was 1 for slots with a new value to indicate that there was a new value. The other bits were the new value. Slots that had no new data had 0x000 as new value (first bit was 0 to indicate that there is no new data).

Pan and tilt saver

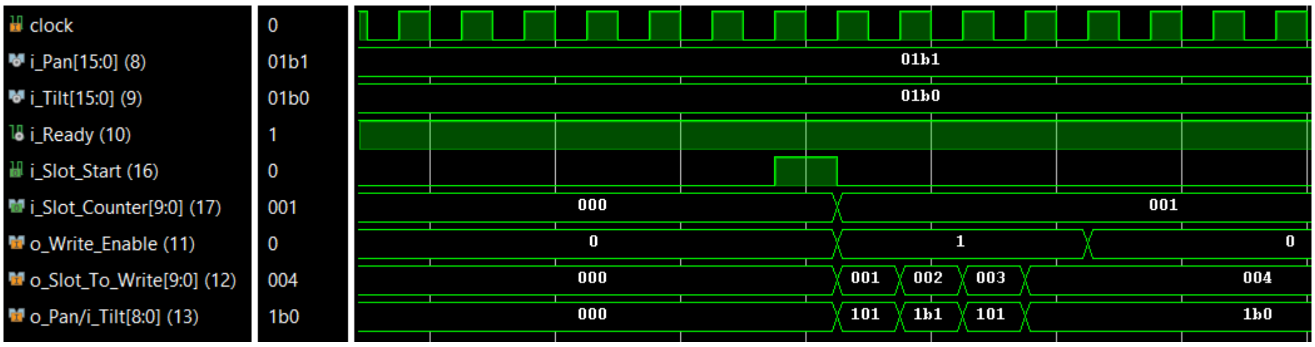


Figure 4.11: Pan and tilt saver simulation

The pan and tilt saver is displayed in Figure 4.11. If a slot started (16), this slot was the first slot data (17) and the pan and tilt was ready (10) then it turned the write enable high (11) for the slot memory. Simultaneously, the pan and tilt saver module generated the values (13) for each slot out of the pan (8) and tilt (9) received values of the pan and tilt calculator, with the corresponding slot to write too (12).

4.1.7 DMX changer

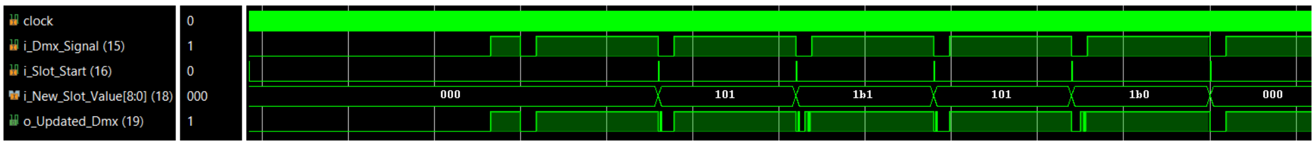


Figure 4.12: DMX changer simulation

Figure 4.12 shows the DMX changer part. It received the generated DMX signal (16) from the EZ-con6 part. If the slot start was high, it saved the new values (18) onto the updated DMX signal (19) when the values needed to be pasted.

Original and updated

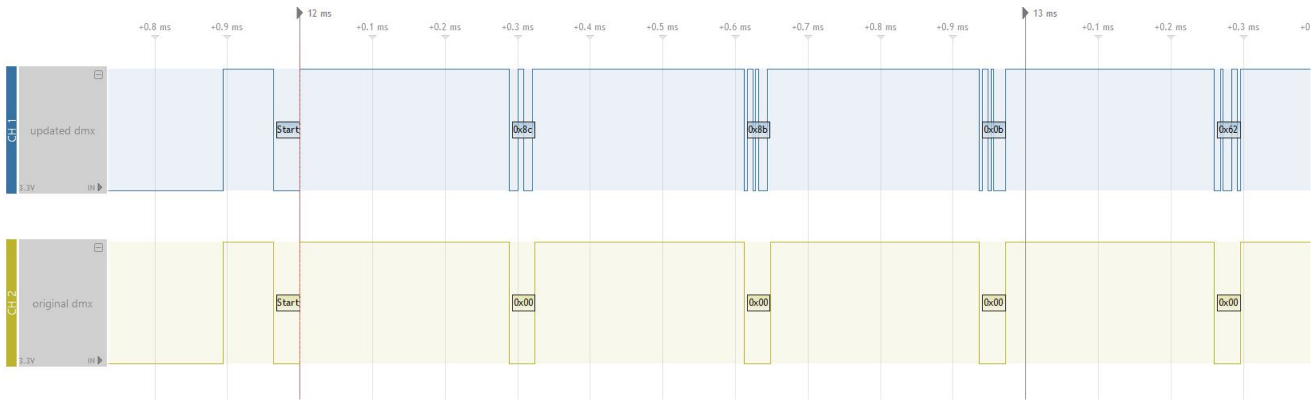


Figure 4.13: First 5 slots of DMX original and DMX updated with pan and tilt

Figure 4.13 shows the original DMX compared to the updated DMX signal. Both were the same except for the data the slots. The updated DMX signal contained the pan and tilt values for coordinates 200, -400, 200 as seen in 4.1.3.

4.1.8 Resources used

The developed FPGA design used slices, Ram and DSPs component. Table 4.2 displays the total resources of the zed-board, the total resources used by the design and the resources used by each component. The design used a low amount of the resources of the total resources on the Zedboard. 21,8 % by the RAM tile was used the most. Most of the resources were used for the pan and tilt module. This module had the most slices and was the only module with DSPs because of the calculation of the angles. The pan and tilt also used a RAM to convert 540° to 360° as seen in 4.1.3. Therefore, it used the most RAM tiles.The slot memory ram was the other module with RAM tiles. The other modules all had fairly low use of slices. Summing up all the individual slices resulted into a higher total resources as the total resource design seen in Table 4.2. This was because some slices are used for multiple parts. The total resources design slices was the correct value to use.

Table 4.2: Slices, Block RAM and DSP resources of the current design

Component	Slices	RAM Tile	DSPs
Total resources Zedboard	13300	140	220
Total resources design	609 (4,6 %)	30.5 (21,8 %)	2 (0,9 %)
DMX Controller	38	0	0
Marvelmind	60	0	0
Slot memory ram	1	0,5	0
DMX receiver + sender	40	0	0
Pan and tilt	476	30	2
UART rx	15	0	0

4.2 Simulation

The following section shows the results of the simulation. It starts with setup used for simulating the system. Next, the results of the communication between the used Arduinos and Unity are shown. This is followed by the results of the spotlight following the person in Unity. The section ends with a video of the test simulation.

4.2.1 Setup

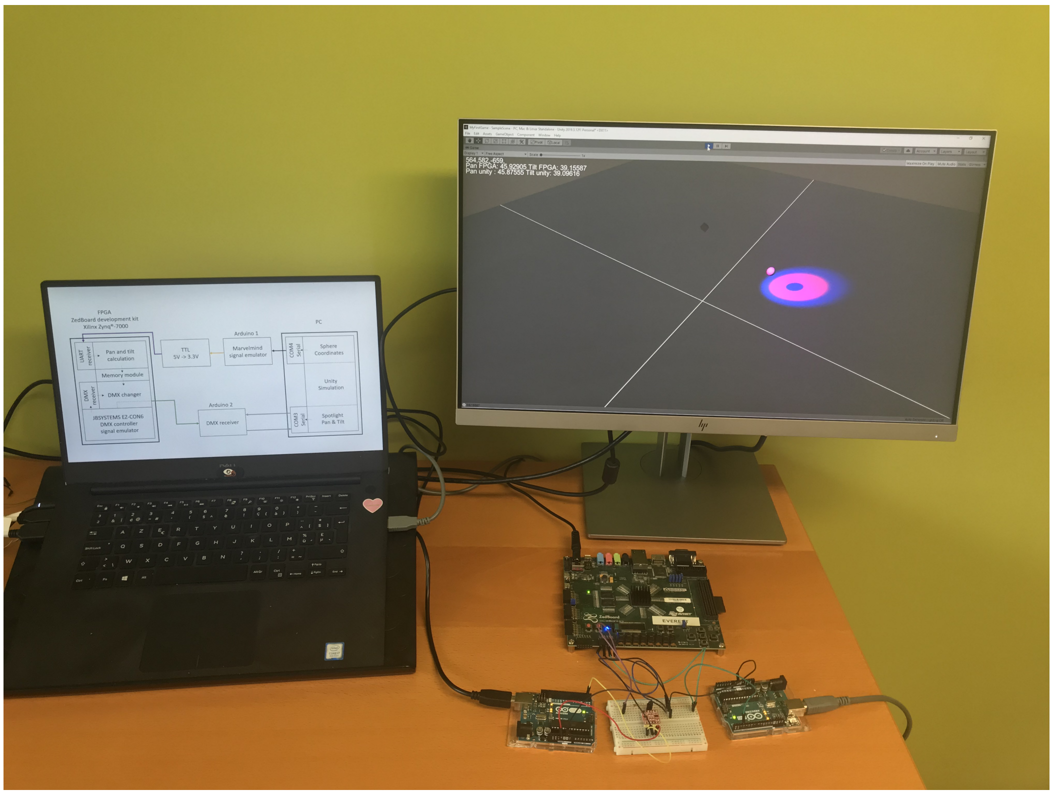


Figure 4.14: Simulating setup

Figure 4.14 displays the simulating system described in 3.3 in practice. The left screen showed the design of the system. The right screen showed the Unity environment with the FPGA controlled spotlight (red spotlight) and the ideal spotlight (blue spotlight). The sphere represented the person that needed to be followed. The device closest to the right screen was the ZedBoard FPGA. Arduino 1 was the right Arduino and Arduino 2 was the left Arduino. The TTL was positioned between the two Arduinos. The Arduinos were connected to the PC with a USB 2.0 CABLE TYPE A/B and to the FPGA and TTL with jumper wires.

4.2.2 Arduino 1

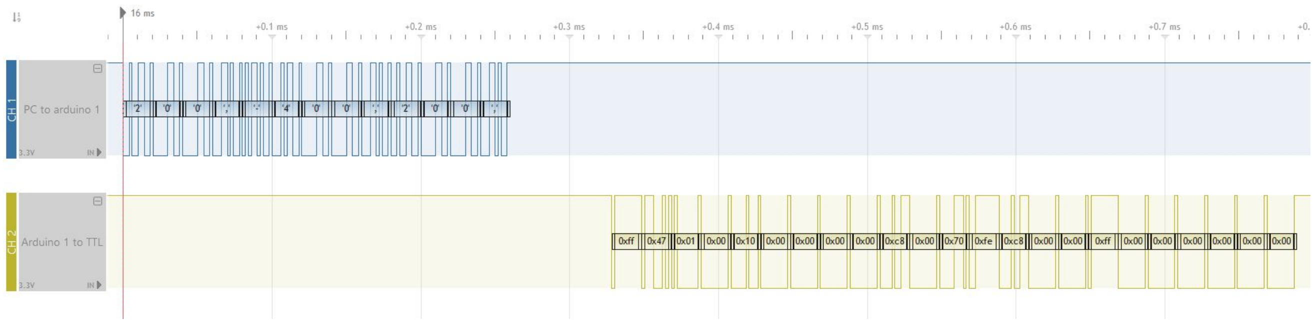

16 ms
+0.1 ms
+0.2 ms
+0.3 ms
+0.4 ms
+0.5 ms
+0.6 ms
+0.7 ms
CH 1
PC to arduino 1
CH 2
Arduino 1 to TTL
0xff 0x47 0x01 0x00 0x10 0x00 0x00 0x00 0x00 0xc8 0x00 0x70 0xfe 0xc8 0x00 0x00 0xff 0x00 0x00 0x00 0x00 0x00 0x00

Figure 4.15: Output input Arduino 1

Figure 4.15 illustrates the income and output signal of Arduino 1. The blue signal was send by Unity to the Arduino. The yellow signal was send by the Arduino to the TTL. The Arduino started sending the Marvelmind packet once it received 3 numbers.

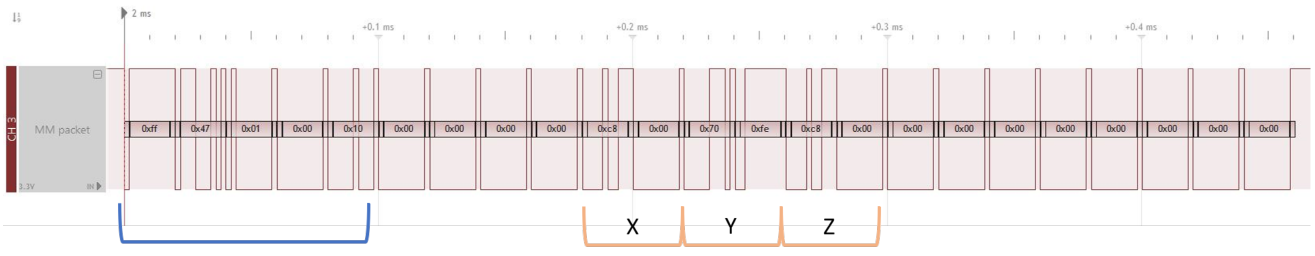

2 ms
+0.1 ms
+0.2 ms
+0.3 ms
+0.4 ms
CH 3
MM packet
0xff 0x47 0x01 0x00 0x10 0x00 0x00 0x00 0x00 0xc8 0x00 0x70 0xfe 0xc8 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
X
Y
Z

Figure 4.16: Marvelmind packet for numbers 200,-400,200 dec

Figure 4.16 displays the Marvelmind packet sent by Arduino 1. The frames between the blue bracket on Figure 4.16 contain the information over the simulated Marvelmind packet.Therefore, they were not zero (see 3.2). The frames between the orange brackets represented the coordinates of the Marvelmind packet (x is 200 dec or 0x00C8 hex, y is -400 dec or 0xFE70 and z is 200 dec or hex 0x00C8). The others frames were all zero frames as described in 3.3.2.

4.2.3 Arduino 2

Figure 4.17 shows the input and output of Arduino 2. The blue signal was the DMX signal generated by the FPGA. The first 4 data slots of the DMX signal are shown in Figure 4.18. The yellow signal was the signal from the Unity to Arduino 2 and the red signal was the signal from Arduino 2 to the Unity. This Arduino sent the data to the Unity once it received a character. In this case, a ”0” was send, but this could be any character. The values sent to the Unity were the values from the first 6 slots, as shown in Figure 4.17, Figure 4.18, and Figure 4.19.

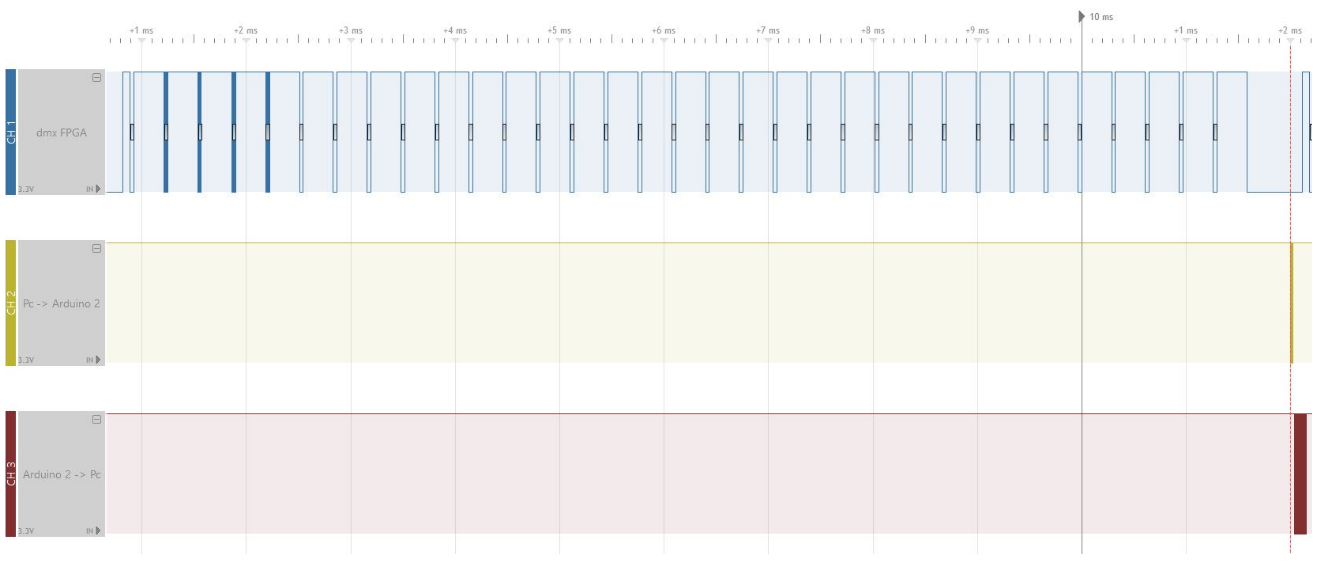


Figure 4.17: Output input Arduino 2 with DMX

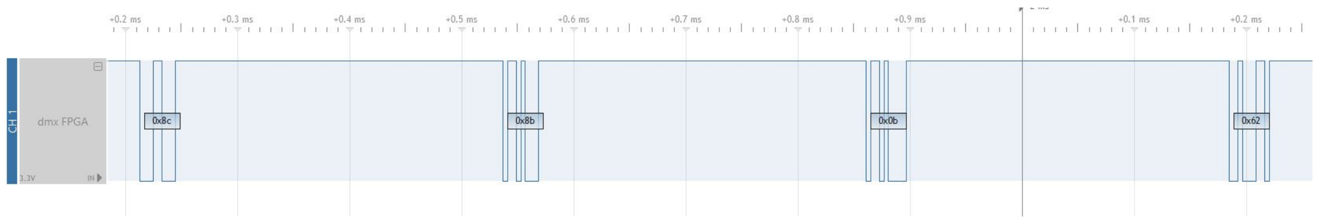


Figure 4.18: First 4 slots of the DMX signal

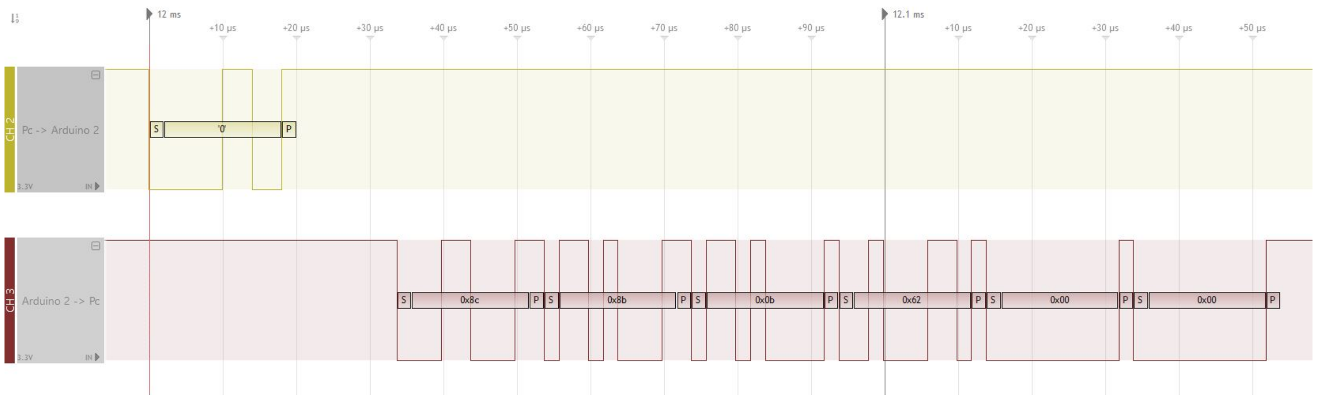


Figure 4.19: Output input Arduino 2 without DMX

4.2.4 Unity environment

Figure 4.20 shows the Unity project for simulating the person and the moving head spotlight. The items in the red box were objects added to the scene. These were 3D objects like a sphere, the ground and lines, light sources like the spotlight FPGA, spotlight real and directional light and cameras to get a view while running the simulation. The blue box displayed the scripts that were used to handle the communication between Unity and both Arduinos, adjusting properties of the 3D objects and light sources and movement of the sphere and camera.

The sphere represented the person that needed to be tracked. Coordinates of the center of the sphere were send to Arduino 1 as position of the person. A ground plane and wall (not visible on 4.22) were added to see the spotlights. The red spotlight was controlled by the DMX controllor on the FPGA. The blue spotlight was the ideal spotlight controlled by Unity. The cube represented

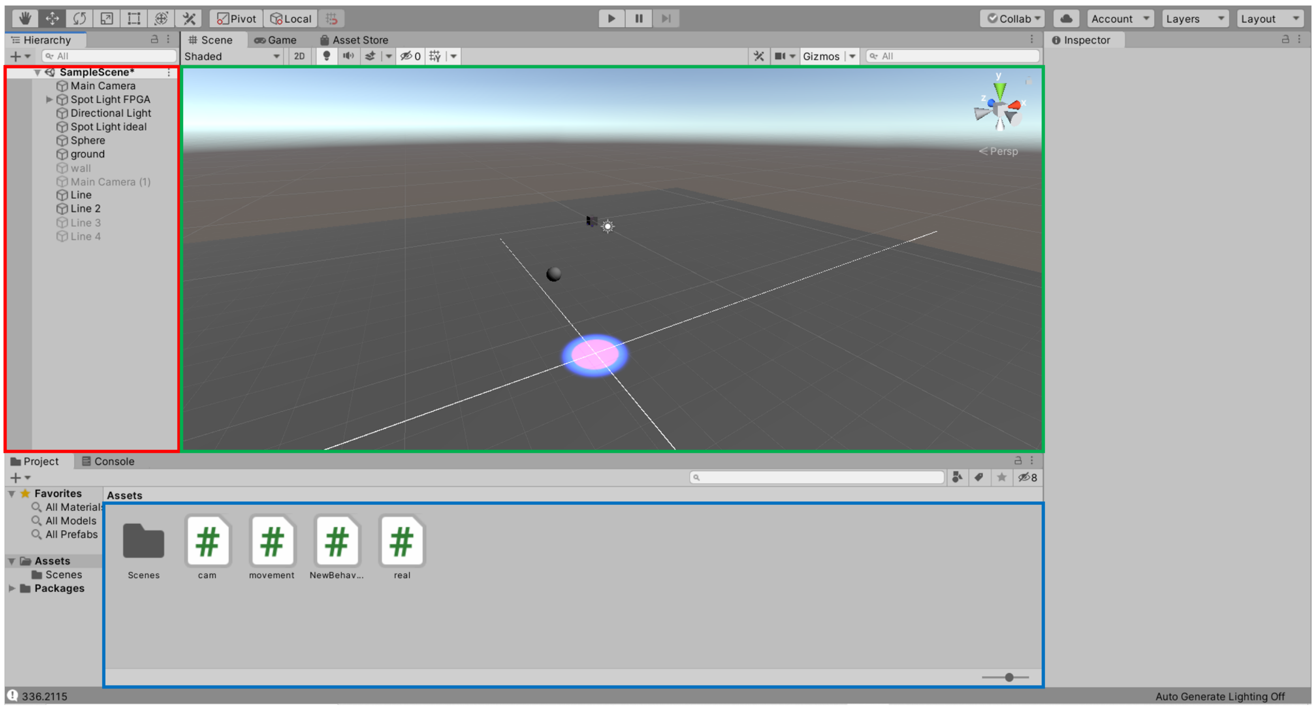


Figure 4.20: Unity 3D simulation environment

the red spotlight and turned when the DMX controlled spotlights turns. The lines were added to visualize the transmission from negative to positive coordinates. There were no 3D objects added to visualise the ideal spotlight. Lastly, a directional light was added to get a light effect on places where there was no spotlight.

4.2.5 Communication

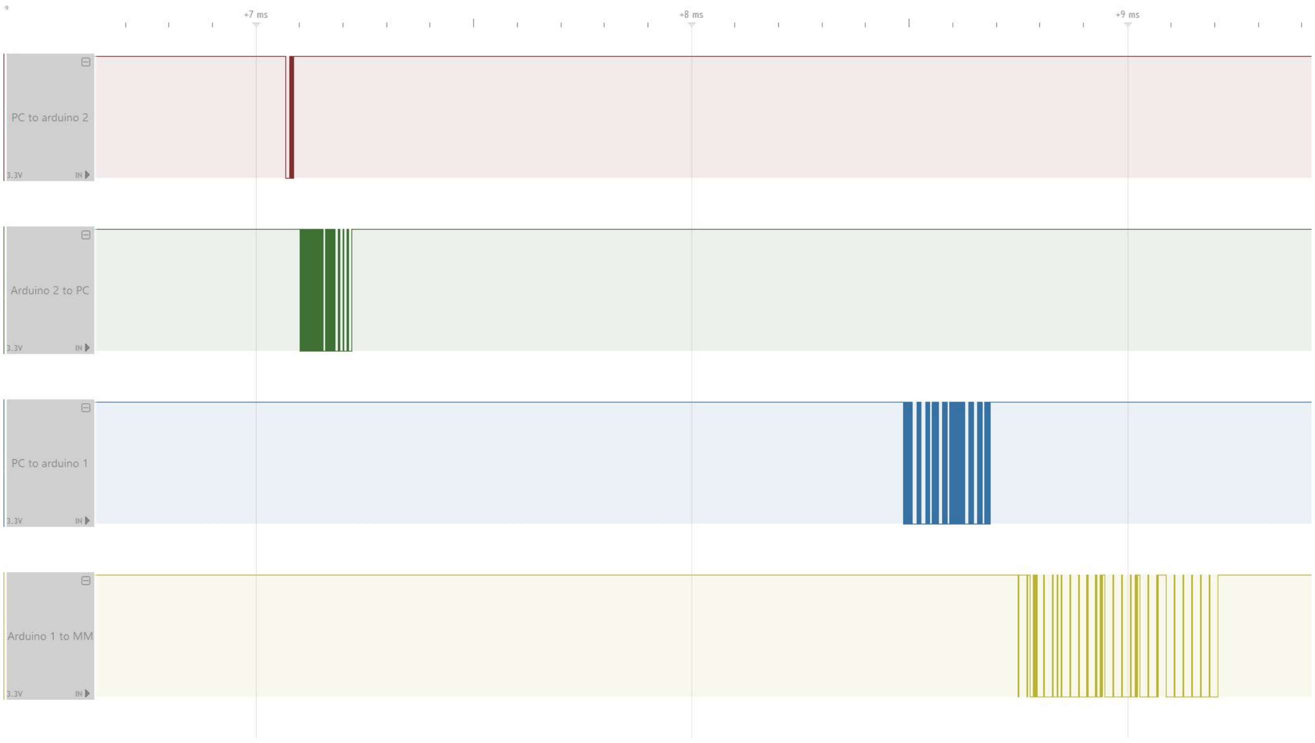
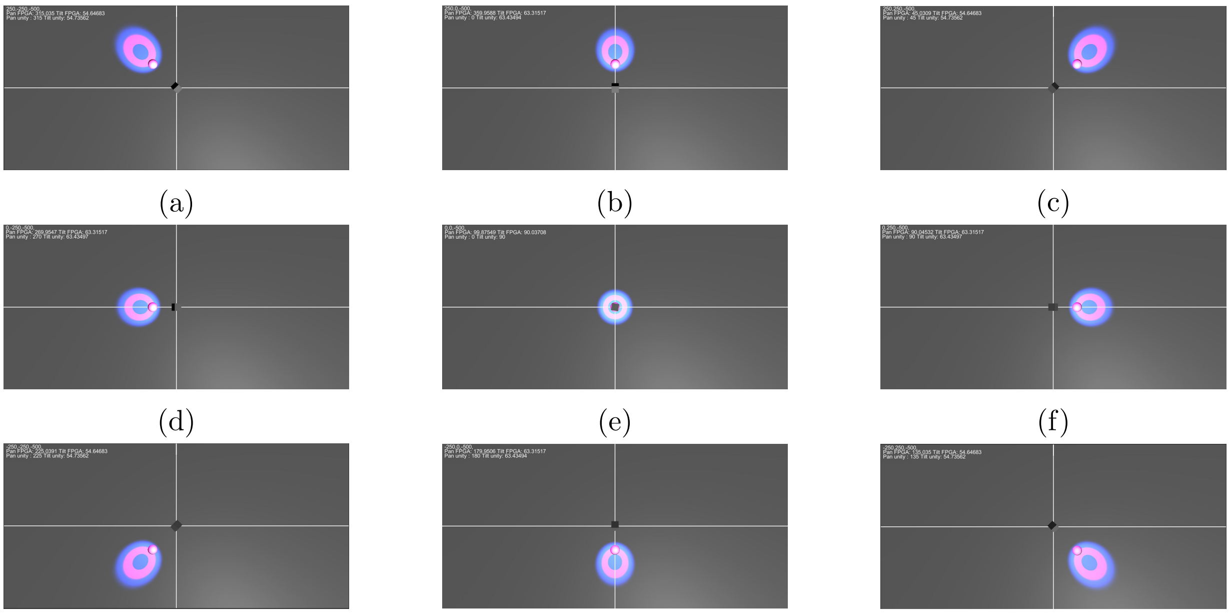


Figure 4.21: Output input Arduino 2

Figure 4.21 shows the communication between Unity and both Arduinos. The handling of how the Unity communicated with the Arduinos happened as expected. First, a character was send to Arduino 2 (red signal). Then, Arduino 2 responded with the values of the slots (green signal). Next, the Unity sent the coordinates of the sphere to Arduino 1 (blue signal). The response of the Arduino 1 was also shown (yellow signal), even though it was not used by Unity, to display that Arduino 1 sent the Marvelmind packet when it received the coordinates.

4.2.6 Visualisation

Pan



```
250,-250,-500,
Pan FPGA: 315.035 Tilt FPGA: 54.64683
Pan unity : 315 Tilt unity: 54.73562
```

(a)

```
250,0,-500,
Pan FPGA: 359.9588 Tilt FPGA: 63.31517
Pan unity : 0 Tilt unity: 63.43494
```

(b)

```
250,250,-500,
Pan FPGA: 45.0309 Tilt FPGA: 54.64683
Pan unity : 45 Tilt unity: 54.73562
```

(c)

```
0,-250,-500,
Pan FPGA: 269.9547 Tilt FPGA: 63.31517
Pan unity : 270 Tilt unity: 63.43497
```

(d)

```
0,0,-500,
Pan FPGA: 99.87549 Tilt FPGA: 90.03708
Pan unity : 0 Tilt unity: 90
```

(e)

```
0,250,-500,
Pan FPGA: 90.04532 Tilt FPGA: 63.31517
Pan unity : 90 Tilt unity: 63.43497
```

(f)

```
-250,-250,-500,
Pan FPGA: 225.0391 Tilt FPGA: 54.64683
Pan unity : 225 Tilt unity: 54.73562
```

(g)

```
-250,0,-500,
Pan FPGA: 179.9506 Tilt FPGA: 63.31517
Pan unity : 180 Tilt unity: 63.43494
```

(h)

```
-250,250,-500,
Pan FPGA: 135.035 Tilt FPGA: 54.64683
Pan unity : 135 Tilt unity: 54.73562
```

(i)

Figure 4.22: Effect of coordinates on the pan of the DMX controller

Table 4.3: Extra information subfigures of figure 4.22

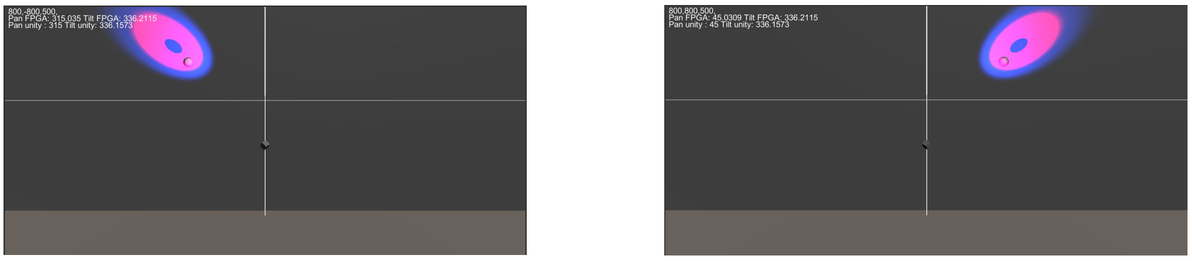
label	X	Y	Z	Unity pan °	Unity tilt °	fpga pan °	fpga tilt °
a	250	-250	-500	315	54.74	315.04	54.65
b	250	0	-500	0	63.43	359.96	63.31
c	250	250	-500	45	54.74	45.03	54.65
d	0	-250	-500	270	63.43	269.95	63.32
e	0	0	-500	0	90	99.88	90.04
f	0	250	-500	90	63.43	90.05	63.32
g	-250	-250	-500	225	54.74	225.04	54.65
h	-250	0	-500	180	63.43	179.95	63.31
i	-250	250	-500	135	54.74	135.04	54.65

Figure 4.22f shows the FPGA spotlight (red) and the ideal spotlight (blue). The horizontal line presented the y-axis and the vertical line presented the x-axis. Table 4.3 displays the coordinates and the pan and tilt calculated by Unity and the FPGA. For the z coordinates, negative numbers

were used to display the effect of the X en Y coordinates. The FPGA spotlight was always inside the ideal spotlight. This can also be seen in Table 4.3 as the value of the pan and tilt by the FPGA and Unity showed similar results except for e. The pan calculated by the FPGA was 99.88° off the value of pan calculated by Unity. However, this was not a problem because the lamp was shining in the extension of the pan axis. Therefore, rotating pan would have effected the direction to which the lamp was shining. This is also shown in 4.22e where the FPGA controlled spotlight was still in the middle of the Unity spotlight.

The pan went from 360° to 0° when it crossed the upper vertical line. This is visualized in Figures 4.22a - 4.22c. The calculated pan by the FPGA in Figure 4.22a is 315°. In Figure 4.22b, this was 359.96 °. When entering the first quadrant (see figure 4.22c, the pan started counting from 0° which resulted in a FPGA pan of 45.03 ° and not 405.03°.

Tilt


```
800,-800,500,
Pan FPGA: 315.035 Tilt FPGA: 336.2115
Pan unity : 315 Tilt unity: 336.1573
```
```
800,800,500,
Pan FPGA: 45.0309 Tilt FPGA: 336.2115
Pan unity : 45 Tilt unity: 336.1573
```

(a) (b)

Figure 4.23: Effect of coordinates on the tilt of the DMX controller

Table 4.4: Extra information subfigures of figure 4.23

label	X	Y	Z	Unity pan °	Unity tilt °	fpga pan °	fpga tilt °
a	800	-800	500	315	336.16	315.04	336.21
a	800	800	500	45	336.16	45.03	336.21

Figure 4.23 already showed the spotlights for multiple coordinates with negative z values. Figure 4.22f shows the spotlights for 2 coordinates with a positive z as coordinate. A wall was added into 3D scene to get a better view of the lights. The horizontal line illustrated the transition between the negative and positive z coordinates. The vertical line illustrated the transition between the negative and positive y coordinates. The pan and tilt of the FPGA were similar to the pan and tilt of the Unity (see table 4.23). The tilt was more than 180° because the lamp was looking upwards.

4.2.7 Video

A video (youtube: https://youtu.be/MqpbinhQAzs) was created to demonstrate the DMX controller on the FPGA.The same setup was used as in 4.2.1. There was some delay between the FPGA controlled spotlight and the sphere. This was due to the delay between sending and reading the coordinates in Unity and the accuracy of 2 digits after the comma (moving the sphere less then 0.01 resulted into sending the same coordinates).The FPGA controlled spotlight was practically synchronised with Unity's blue spotlight which represents the ideal spotlight. The controlled spotlight stopped following the sphere once the cable between the FPGA and DMX receiver was disconnected. The controlled spotlight continued to follow the sphere once this cable was reconnected.

4.3 Discussion

The current implementation performed proper tracking. A useful aspect during the development phase was the possibility to make adjustments to the hardware. For example, instead of 512 data slots, it was possible to broadcast only 32 data slots with a minor adjustment in HDL. A number of such adjustments took place in the development phase of the first prototype, which accentuated the benefits of the flexibility of the FPGA.

From Table 4.2 it could be concluded that BRAM used the most resources of the FPGA in proportion to the other modules. This originated from the ROM in Section 3.1.3 that provided the conversion of the angle to a correct spotlight angle. For every angle that had to be converted to its own specific angle, a ROM conversion table was needed. Only a single ROM was used in the test setup. In real life, two instances of this ROM are often required as 540° for pan and 270° for tilt are common. Taking this into account, it was possible to search for the cheapest FPGA on which the current implementation could run. The XC7A50T can be found in the datasheet of the 7 Series FPGAs [28], which met the BRAM specification to control a spotlight with different pan and tilt. This chip had a cost of approximately 50 euros excluding VAT according to the following website [29].

This was a very rough estimate. However, more ways to convert angles should be explored in the future. There are many possible adjustments that can improve the design and are not yet been looked at. The initial goal of the first test set-up was to achieve correct operation.However, with the current successful simulation, the next question rises: How can multiple spotlights be added to the person tracking system? A first idea is to use a RAM memory where an offset of each spotlight will be given with respect to the origin of the coordinate system. Then, the pan and tilt of each location can be calculated sequentially and inserted into the correct DMX512 channel. There are also other ways such as using a Marvelmind streaming packet that requests locations from the beacons whereby the beacons are associated with their associated spotlight.

Since the FPGA design can be adjusted at any time, there will also be the possibility to easily add other functionalities of the spotlight. An important function of a spotlight is to adjust the focus. It might be interesting that as the distance increases, the light beam becomes more focused. Similar applications can be devised to take advantage of other functions such as color selection and dimming.

Finally, a correct comparison with existing tracking systems in the lighting industry could not yet be performed. No tests were been taken place in a practical work environment due to COVID-19 restrictions. Practical tests are important to make a decision on how well the Marvelminds perform in a potentially difficult environment. There may be possible interference or other phenomena that might indicate that the Marvelminds trackers are not suitable for the application. The Marvelminds work on ultra-sound and therefore may also be affected by a lot of wind, so the question is whether the system can be used outdoor. It is recommended to switch to a different tracking device because the current devices are not waterproof. Marvelmind offers an

IP67 model [30] for this, but this waterproof model has an impact on the price. In addition, it is difficult to estimate a cost, since the number of beacons to be installed could not be correctly determined. Therefore, this requires further testing to determine whether a tracking system with FPGAs and Marvelmind trackers can gain market share in the spotlight tracking market.

Chapter 5

Conclusion and further work

This thesis developed a basic person tracking system with the aim of using it in the entertainment and lighting industry. Although there were no practical tests with a spotlight, it was possible to mimic the operation via simulation. Because no concrete tests have taken place in the working environment, it is difficult to make decisions regarding the financial and the economic aspects of the person tracker. Therefore,it is important that future work realizes a practical comparison with existing systems in order to gain insight into the potential of the current setup. As demonstrated in this thesis, the introduction of an FPGA as a central controller in the system offers many advantages. This way, new functions can easily be added without losing performance and without purchasing new specialized hardware. From the developer perspective, this has tremendous advantages. Different systems can be developed on the same hardware and depending on the application, the right bitstream can be uploaded.

In the long term, this will also yield potential benefits for the installation companies. As such, the latest technologies can be integrated into the installations with a single long-term purchase. There is still the cost of developing and maintaining the HDL code. However, this development can proceed very efficiently. If the integration of FPGAs eventually increases in popularity in the entertainment and stage lighting industry, a company may be formed that is solely engaged in developing code at the request of stage installation companies. In a manner of speaking, a company can then rent out FPGAs and offer customized solutions at a relatively low cost through their expertise and high demand for similar setups. However, these are only speculations and give an idea behind the vision of a possible new market.

Bibliography

[1] "Automated follow system: Zactrack lighting technologies gmbh: Wien." [Online]. Available: https://www.zactrack.com/

[2] "Follow-me traxyz." [Online]. Available: https://follow-me.nu/product/traxyz-extension

[3] "Pozyx." [Online]. Available: https://www.pozyx.io/

[4] *American National Standard ANSI E1.11 – 2008 (R2018) Entertainment Technology—USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories*, ESTA, 2018. [Online]. Available: https://tsp.esta.org/tsp/documents/docs/ANSI-ESTA_E1-11_2008R2018.pdf

[5] R. Mautz, "Indoor positioning technologies," 2012.

[6] B. O'Keefe, "Finding location with time of arrival and time difference of arrival techniques," *ECE Senior Capstone Project*, 2017.

[7] *THE EUROPEAN TABLE OF FREQUENCY ALLOCATIONS AND APPLICATIONS IN THE FREQUENCY RANGE 8.3 kHz to 3000 GHz (ECA TABLE)*, Electronic Communications Committee (ECC) within the European Conference of Postal and Telecommunications Administrations (CEPT), 2019. [Online]. Available: https://www.efis.dk/sitecontent.jsp?sitecontent=ecatable

[8] S. Wielandt and L. D. Strycker, "Indoor multipath assisted angle of arrival localization," *Sensors*, vol. 17, no. 11, p. 2522, 2017.

[9] Z. Farid, R. Nordin, and M. Ismail, "Recent advances in wireless indoor localization techniques and system," *Journal of Computer Networks and Communications*, vol. 2013, 2013.

[10] S. Xia, Y. Liu, G. Yuan, M. Zhu, and Z. Wang, "Indoor fingerprint positioning based on wi-fi: An overview," *ISPRS International Journal of Geo-Information*, vol. 6, no. 5, p. 135, 2017.

[11] *A Summary of Worldwide Telecommunications Regulations governing the use of Ultra-Wideband radio*, Decawave Limited, 2015. [Online]. Available: https://www.decawave.com/sites/default/files/apr001_uwb_worldwide_regulations_summaryrev1.2.pdf

[12] K. Weedman, "Cordic design and simulation using verilog." [Online]. Available: http://www.hdlexpress.com/Verilog/VT.html

[13] A. A. B. Raj, *FPGA-based embedded system developer's guide.* CRC Press, 2018.

[14] A. Sultan, “Cordic: How hand calculators calculate,” *The College Mathematics Journal*, vol. 40, no. 2, pp. 87–92, 2009. [Online]. Available: https://doi.org/10.1080/07468342.2009.11922342

[15] A. N. Standard, *USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories*, 2018.

[16] *ZedBoard Hardware User’s Guide*, Zedboard, 2014. [Online]. Available: http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf

[17] “Zedboard zynq-7000 arm/fpga soc development board.” [Online]. Available: https://www.xilinx.com/products/boards-and-kits/1-elhabt.html

[18] “Marvelmind robotics,” May 2020. [Online]. Available: https://marvelmind.com/

[19] “Architectures comparison,” Nov 2018. [Online]. Available: https://marvelmind.com/pics/architectures_comparision.pdf

[20] “Hardware interface and protocol of data exchange with mobile beacon via usb, uart and spi interfaces,” Aug 2019. [Online]. Available: https://marvelmind.com/pics/marvelmind_beacon_interfaces.pdf

[21] *Vizi BSW 300 User Instructions*, ADJ Products, 2017. [Online]. Available: https://cdb.s3-us-west-1.amazonaws.com/ItemRelatedFiles/10525/vizi_bsw_300.pdf

[22] “Uart, serial port, rs-232 interface.” [Online]. Available: https://www.nandland.com/vhdl/modules/module-uart-serial-port-rs232.html

[23] “Rs-485 (eia/tia-485) differential data tr - maxim integrated.” [Online]. Available: https://www.maximintegrated.com/en/design/technical-documents/tutorials/7/736.html

[24] *MAX481/MAX483/MAX485/ MAX487–MAX491/MAX1487*, Maxim Integrated Products, 2014. [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf

[25] I. Logic. [Online]. Available: https://www.ikalogic.com/sq-logic-analyzer-pattern-generator/

[26] [Online]. Available: https://www.tek.com/oscilloscope/mso4000-dpo4000

[27] *MSO4000B, DPO4000B Series Datasheet*, Tektronix, 2013. [Online]. Available: https://datasheetspdf.com/pdf-file/799310/Tektronix/MSO4102B-L/1

[28] *7 Series FPGAs Data Sheet: Overview*, Xilinx, 2018. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

[29] “Xc7a50t-1ftg256c.” [Online]. Available: https://www.digikey.be/product-detail/nl/xilinx-inc/XC7A50T-1FTG256C/122-1916-ND/5039080

[30] “Beacon mini-rx-ip67,” Oct 2019. [Online]. Available: https://marvelmind.com/product/mini-rx-ip67/