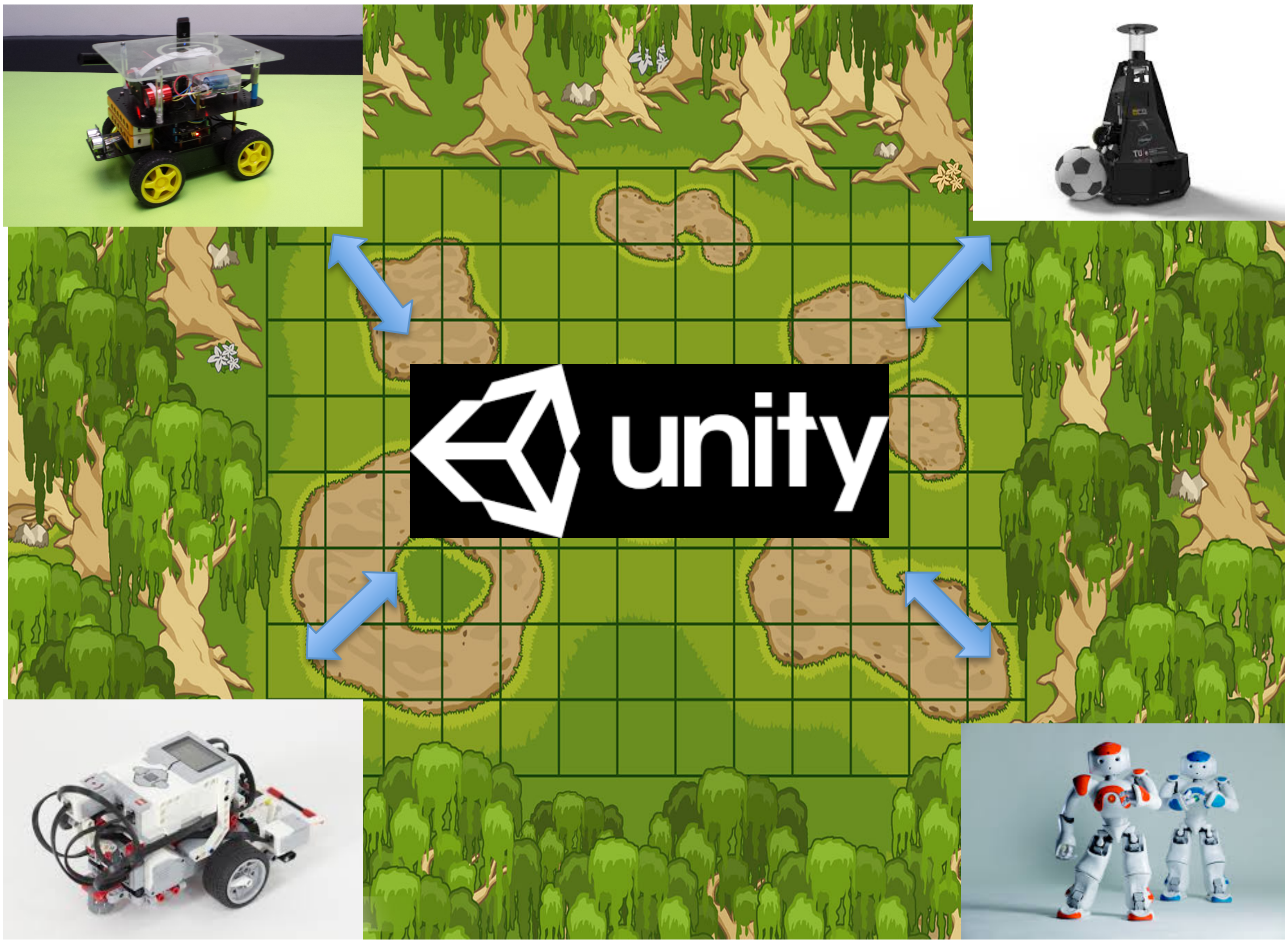
Research: Robot Control from Unity

Rene Bakx, June 2016



Abstract

Standard tools provided by the robot manufacturers work well for simple control of a robot like moving or simple navigation. These standard tools are less suited for giving a robot AI behavior. This requires:

- Knowledge of the robots surroundings or a world representation to be able to plan ahead.
- Tooling which can use this knowledge to give a robot AI behavior.

These ingredients are typically available in games: there is a world representation and tooling available to give game characters AI behavior in this world. Examples of this tooling include inverse kinematics, navigation, pathfinding, behavior design and AI. The research is divided in two parts, of which the first part is described in this report.

1. The first part of the research investigates whether using a game engine for controlling a robot is useful and technically feasible. This research is done during the spring semester 2016 and it shows that using a game engine can be useful and is technical feasible. The latter is proved by building a simple prototype.
2. The second part of the research will focus on giving a robot AI behavior using a game engine. This part of the research will be conducted during the fall semester 2016. It will be documented in a separate report.

Glossary of Terms

AI	Artificial Intelligence
IDE	Integrated Development Environment
OSI	Open Systems Interconnect
ROS	Robot Operating System
SLAM	Simultaneous Localization And Mapping.
TCP	Transmission Control Protocol
TRE	The Robot Engine
UDP	User Datagram Protocol
Unity	Cross Platform game engine developed by Unity Technologies

Table of contents

1. Introduction

1.1. Context

This research is conducted at the ICT & Technology department of the Fontys University of Applied Sciences. At this department students learn about technology in which robotics is an important topic. The research described in this report is about controlling robots and is conducted in the spring semester of 2016.

1.2. Problem description

Controlling a robot is mostly done by using programming tools available for the specific robot. For the Lego Mindstorms robot this is the Lego Mindstorms IDE, for the Nao robot this is Choregraphe. Other robots might run ROS (Robot Operating System) which is middleware containing packages e.g. for motion tracking, face recognition and SLAM (Simultaneous Localization And Mapping). These tools work well for moving a robot along its degrees of freedom, reading sensor data or simple navigation. However they are less suited for giving a robot AI behavior. In this report the term AI means 'Artificial Intelligence' in the context of Game AI. This means it's more the illusion of intelligence and thoughtful action than an actual intelligent reasoning driven behaviour. This requires:

- Knowledge of the robots surroundings or a world representation, to be able to plan ahead.
- Tooling which can use this knowledge to give a robot AI behavior.

These ingredients are typically available in games: there is a world representation and tooling available to give game characters AI behavior in this world. Examples of this tooling include inverse kinematics, navigation, pathfinding, behavior design and AI.

1.3. Goal

This research is part of a bigger research project and it can be considered as being pre-research. The outcome of this pre-research will determine whether or not to continue with the remaining part of the complete research project. The remaining part of the research is on giving robots intelligent behavior or AI (Artificial Intelligence) behavior using a game engine. This will be documented in a separate report.The goal of this research project is to investigate whether using a game engine for controlling a robot has added value and technicallly feasible.

1.4. Constraints

The constraints for this research are listed below.

- The research described in this document is part of a bigger research project. It can be considered as a pre-research. The budget for the complete research is 100 hours. This includes:
    - Writing a research proposal.
    - Doing the pre-research described in this document.
    - Writing a Project Initiation document.
    - Organize and guide student research projects.

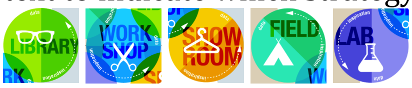
  It excludes:
    - The hours student will spend in the research projects.

  For this pre-research no more than 40 hours can be spent, to leave enough hours for the other activities listed above.

- Because of the very limited time, available or easily obtainable hardware should be used. This will be a previously built robot ([1]) and the Unity game engine.

1.5. Research questions

Before letting students work on a project, it is good that there is confidence that the project is useful ('makes sense') and is technically feasible. When there is experience from the past with similar projects this confidence is already there. However, this project is not a standard project and similar projects are rare if any. Therefore the first two research questions deal with these two axes of confidence: usefulness and feasibility.
Only when both questions can be answered positively, student projects will be defined in which the third question will be answered. Therefore the first two research questions are closed questions. The third question deals with how to use a game engine to add AI behavior to a robot and will be documented in a separate report.
For answering the first two research questions we will use research strategies as defined in [14], backed up by [5]. For clarity we will use the symbols below in the text to indicate which strategy is used.



2. Research question 1

Is control of a robot from Unity useful?
To answer the question whether the project is useful or not ('makes sense'), we first have to define the criteria to determine this. Still the term useful will remain subjective, but we want to have a certain level of confidence that the project will lead to results which have added value.

2.1. Sub-question 1a

What are the criteria for useful control of a robot from Unity?
Using common sense we define the following criteria:

1. Existing research projects, if any, indicate that controlling a robot from Unity is useful.
2. Local game experts indicate that controlling a robot from Unity is useful.
3. The community interested in robotics indicates that controlling a robot from Unity is useful.

2.2. Sub-question 1b

Does our project meet the criteria for useful control of a robot from Unity?
For this question we use the library and field strategies to test the two criteria defined above.

1. 

For the first criterion defined in 2.1 we did some research on available studies and projects. At first we checked official resources like [8] (HBO Kennisbank) and [12] (Scripties van de Nederlandse Universiteiten). Here projects on game engines combined with robotics are available, but with a different goal in mind, for example the simulation of a robot using a game engine.
We broadened the search by using Google and keywords 'robot', 'robotics', 'game engine', 'Unity', 'Unreal' in various combinations. We did find projects using game engine Unity for controlling a physical robot, most notably [4] and [17]. [4] describes a research project to control a lego robot from Unity. This research is done by associate professor Bartneck, C et al. at the University of Canterbury. A quote from this research which uses the term 'The Robot Engine' (TRE): "*This approach is so simple and powerful that we ourselves are surprised that it has not yet already been widely used. We believe that TRE has the potential to bridge the gap between engineers and computer scientists on the one side and psychologists and designers on the other*".
[17] shows on YouTube project where a robotic hand is controlled from Unity.

However, both projects focus on direct control of a robot from Unity, while we want to want to focus on the AI behavior of a robot controlled from Unity. This because AI behavior is difficult to program using tools available on robots while it is well supported in Unity.

2. 

For the second criterion defined in 2.1 some colleagues from ICT & Game Design and Technology were informally asked the question “do you think it makes sense to control a physical robot from Unity with the purpose of giving the robot AI behavior”. The responses were along the following lines:
   - “*Great idea, but response times might be a problem.*”
   - “*Interesting idea, but you will only use a small part of Unity, maybe a tool like Blender is more suitable.*”
   - “*Great, I get questions of students how to connect a Lego Mindstorms robot to Unity. Please let me know when you have results.*”
3. 

For the third criterion defined in 2.1 the showroom strategy is used: a one minute video showing a prototype and an explanation is posted on YouTube ([3]). In the video there is the requests for comments. Since this is very recent, up till now there is one reaction:
   - *“this is cool..... unity is more focused on games right, i mean 3d and other stuff, which are resource intensive. Can we have it running onboard the bot itself?”*

2.3. Conclusion sub-questions

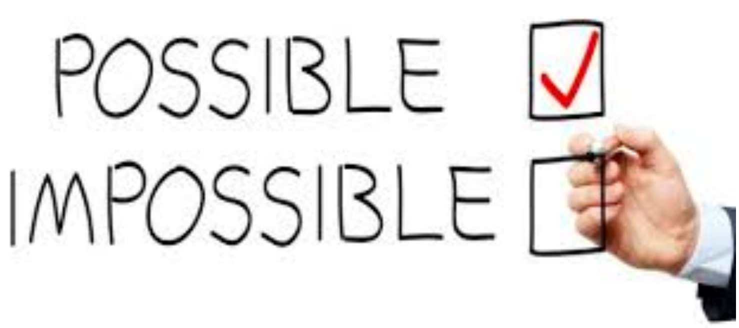
The conclusion of the sub-questions can be summarized as follows:

- There are some research projects done combining Unity and physical robots, but not many. Most of the projects focus of direct control of the robot, and not on the AI behavior.
- Local game experts give encouraging comments. The idea of controlling a robot from Unity is seen as an interesting idea with potential to lead to a useful result.
- The opinion of the community is coming in, but more time is needed to collect data.

2.4. Conclusion research question 1

From the conclusions of the sub-questions follows the conclusion of research question 1: *Both existing research projects as local game experts suggest that controlling a robot from Unity is useful.*

3. Research results research question 2



Is control of a robot from Unity technically feasible?
To answer the question whether the project is technically feasible we first have to define the criteria to determine this. Because the technical feasibility of this project mainly depends on the communication between the robot and Unity, the sub-questions deal with the communication.

3.1. Sub-question 2a

What are the criteria for the communication between a robot and Unity?
For this question we will use our common sense to define the criteria. The criteria for communication can be categorized as follows:

3.1.1. Physical

For the physical connection of the communication we need a wireless connection, so the robot can move freely.

3.1.2. Technology

For the wireless technology we need a technology which is available worldwide and which can deliver high performance. In addition the technology must be available on devices such as robots and laptops.

3.1.3. Performance

Performance can be split up in bandwidth and latency.

3.1.3.1. Bandwidth*:*

Communication between Unity and a robot can take place at multiple levels:

1. At direct control level.
   This means that every degree of freedom of the robot is controlled from Unity. For an humanoid robot this would mean that for example walking and keeping balance is controlled from Unity. For this a large communication bandwidth would be required, for example for the 3-axis accelerometer datastream.
2. At AI level.
   At this level only the desired movement or other action is determined by Unity, like ‘move to the right’ or ‘give the distance to the object straight ahead’.

For this research we focus on communication between Unity and a robot on AI level. The idea is that more primitive operations like keeping balance or

computer vision can better be done on the robot itself, as dedicated and optimized libraries for these operations are often available.
For the communication on AI level the required bandwidth will be limited. For a maximum speed of 1 m/s and a positioning accuracy of 10 cm we need a data transfer every 0.1 s. For obtaining information of the objects around the robot it seems reasonable to use some kind of radar. For a 2D radar image with a 3 meter range and a resolution of 10 cm we need $(2\pi \times 3) / 0.1 \approx 200$ points per 0.1 s. For each point one byte is sufficient to specify the distance. In total this means a bandwidth of 16 kbit/s. For a 3D radar image this will increase to $(\pi \times 3^2) / 0.1^2 \approx 3000$ points per 0.1 s or 240 kbit/s. At a resolution of 1 cm this would increase to appr. 24 Mbit/s.

3.1.3.2. Latency:

If we focus on the AI communication as discussed above, the maximum acceptable latency is about 0.1 s. When this latency is not met, it will limit the maximum allowable speed of the robot.

3.1.4. Protocol

For selecting the communication protocol we define the following criteria.

1. The protocol must not constrain the data to be sent. Therefore we prefer a simple byte stream, as this can be interpreted as any type of data.
2. The protocol must have little overhead, as we want to maximize the performance.
3. The protocol does not have to be 100% reliable. When sensor data or a command is lost it will only lead to a temporary larger deviation between the actual situation and the representation in Unity. At the next information transfer this will be corrected again. This is preferable over a protocol which guarantees correct information transfer at the cost of real time behavior.

3.2. Sub-question 2b

Which technology can be used for the communication which meets the criteria?

For this question we will use the library strategy to check whether existing technologies meet our criteria. Next to this we use the workshop strategy and the lab strategy to actually build a prototype and do measurements. For this we will make use of a robot project maintained on GitHub [1]. This workshop strategy really will help in building confidence that the project is technically feasible and as a side effect it can give a better insight in whether the project is 'useful' or not. We will report on this prototype and measurements in 3.6.
The only wireless communication technology which meets the criteria as specified in 3.1.2 is Wifi. With the Wifi 802.11n standard the practical bandwidth is very dependent of the circumstances and lies in the 15...50 Mbit/s range. The latency of Wifi 802.11n lies in the <100 ms range ([7]). Both bandwidth and latency of Wifi 802.11n meet the criteria as specified in 3.1.3.

3.3. Sub-question 2c

Which protocol can be used for the communication which meets the criteria?

For this question we will use the same strategies as describe in 3.2. As stated in 3.2 we will use Wifi. This wireless protocol operates at OSI layer 1 and 2. [18] shows the OSI model as depicted below including a number of existing protocols per layer.

	Laag	Protocollen/systemen
7	Applicatie	HTTP, SMTP, SNMP, FTP, Telnet, SSH en SCP, NFS, RTSP, RTP, RTCP, X Window System, NFS, LPD, XMPP
6	Presentatie	XML, XDR, ASN.1, SMB, AFP
5	Sessie	TLS, SSL, ISO 8327 / CCITT X.225, RPC, NetBIOS, ASP
4	Transport	TCP, UDP, SCTP, SPX, ATP
3	Netwerk	IP, ICMP, IGMP, X.25, CLNP, ARP, RARP, OSPF, RIP, IPX, DDP
2	Data Link	Ethernet, Token Ring, PPP, HDLC, Frame relay, ISDN, ATM, DSL, SDH, SONET, PDH
1	Fysiek	elektriciteit, radio, laser

Figure 1 OSI model with a number of the protocols from the TCP/IP stack.

Following criterion 1 as stated in 3.1.4 the protocols on layer 5…7 are not suited. This means we will use the Transport layer, which is the minimal layer required for transmitting byte streams. Following criterion 2 and 3 as stated in 3.1.4 we choose for the UDP protocol. This protocol is not 100% reliable but has a better real time behavior. According to [9] UDP is the best choice for multiplayer action games where occasional lag has to be avoided. This scenario resembles Unity connected to multiple robots.

3.4. Sub-question 2d

Which roles do a robot and Unity have with the proposed protocol?

For this question we will use the same strategies as describe in 3.2. According to 3.3 we will use the UDP protocol. This means there is a sever and a client where the client initiates the communication ([11]). Because we want Unity to determine the behavior of the robot, Unity will act as the client and the robot as the server.

3.5. Sub-question 2e

Should the communication between a robot and Unity be synchronous or asynchronous?

For this question we will use the same strategies as describe in 3.2. A game engine like Unity will display the game at a certain frame rate. Every frame Unity will update the state of all game objects. According to [13] updating the game objects happens on the main thread. When the communication between Unity and a robot is delayed this should not influence the frame rate or the communication to other robot connected to Unity. This means that communication between a robot and Unity should be asynchronous. For receiving messages this can be done using a callback method. For sending messages a callback function is not needed. For short messages it is also possible to send it synchronous because the send method will only block until the message is sent, regardless of whether the message arrives or not.

3.6. Conclusions sub-questions

The conclusion of the sub-questions can be summarized as follows:

- The criteria for the communication between a robot and Unity are divided in physical, technology, performance and protocol criteria.
- For the communication we will use Wifi. Wifi 802.11n can deliver sufficient bandwidth for a 3D radar image with a resolution of 10 cm at a distance of 3 m with a frame rate of 10 Hz.
- At a latency of < 0.1 s the maximum speed of the robot will be 1 m/s. When the connection quality decreases then also the maximum allowed speed will decrease because of the increasing latency.
- For the communication protocol we will use UDP. The robot will act as the server and Unity will act as the client initiating the connection.
- Sending and receiving of messages will be asynchronous. Sending short messages can also be done synchronous.
- As explained above the criteria were also tested following the workshop strategy. Existing robot project [1] was used to make an actual connection to Unity using Wifi802.11n and asynchronous UDP socket communication. In Unity a simple scenario was created in which the physical robot has to keep a distance of 50 cm to a box. This behavior is programmed in Unity.
  Measurements on the most critical performance parameter, latency, were done. The latency measured was appr. 0.5 s, but the major part of this 0.5 s can be attributed to a current limitation[1] in robot project [1]. This gives confidence that with optimization the performance criteria can be met. The project is maintained at GitHub [2]. For a YouTube video of the results, see [3].

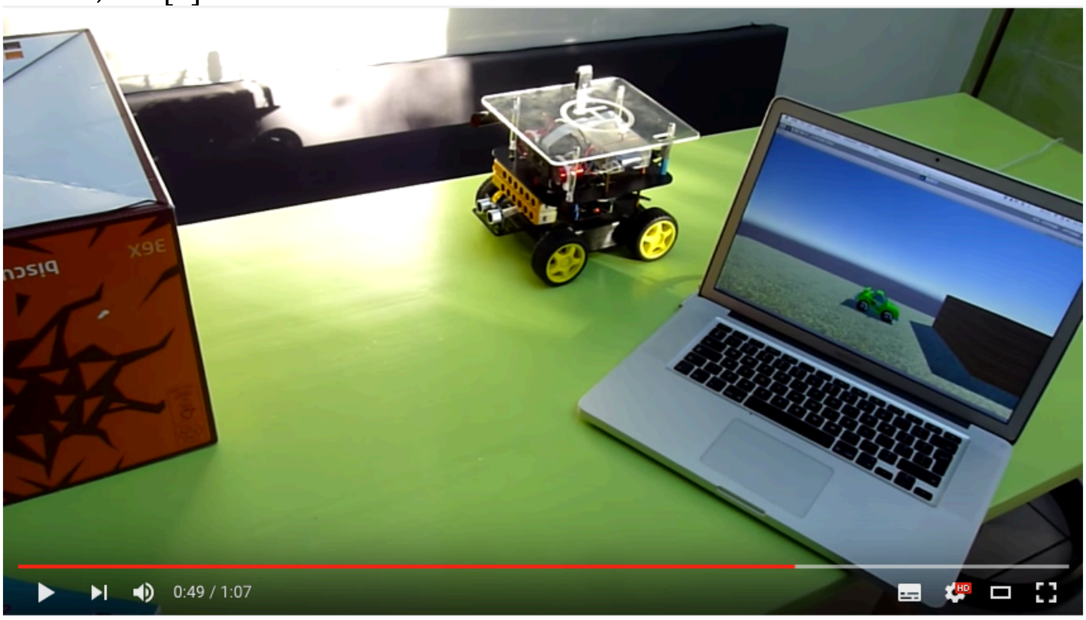


Figure 2. Short demonstration of the prototype on YouTube.

[1] To control the GPIO pins on the Raspberry Pi root permissions are needed. For this a separate Python script was used for every GPIO action which takes about 0.5 s to start up.

3.7. Conclusion research question 2

From the conclusions of the sub-questions follows the conclusion of research question 2: *The control of a robot from Unity is technically feasible*.

4. Bibliography

[1] Bakx, R. (n.d.). DFRobot project. Retrieved May 17, 2016, from https://github.com/rbakx/DFRobot

[2] Bakx, R. (n.d.). UnityRobot project. Retrieved May 17, 2016, from https://github.com/rbakx/UnityRobot

[3] Bakx, R. (2016, May 8). Robot controlled by Unity. Retrieved May 11, 2016, from https://www.youtube.com/watch?v=bVzXpRe1J_o

[4] Bartneck, C., Soucy, M., Fleuret, K., & Sandoval, E. B. (2015). The Robot Engine - Making The Unity 3D Game Engine Work For HRI. *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN2015)*, *RO-MAN2015*, 431-437. doi:10.1109/ROMAN.2015.7333561

[5] Coppens, A., Jacobs, M., Jacobs, T., Niels, R., & Verhoeven, N. (2015, April). PROEVEN VAN ONDERZOEK. Retrieved from http://www.ralphniels.nl/pubs/jacobs-proevenvanonderzoekboek.pdf

[6] Dyachenko, V. (2015, 13 december). C# + .NET: Minimalistic asynchronous UDP example. Retrieved from http://yal.cc/cs-dotnet-asynchronous-udp-example/

[7] Grigorik, I. (2013). *High Performance Browser Networking*. Retrieved from http://chimera.labs.oreilly.com/books/1230000000545/

[8] HBO Kennisbank. (n.d.). Retrieved May 17, 2016, from https://hbo-kennisbank.nl/nl/page/home

[9] Lernö, C. (2014, 1 april). Game servers: UDP vs TCP. Retrieved from https://1024monkeys.wordpress.com/2014/04/01/game-servers-udp-vs-tcp/

[10] Microsoft. (z.j.). Microsoft Developer Network. Retrieved May 8, 2016, from https://msdn.microsoft.com

[11] M Kozierok, C. (2005, September 20). The TCP/IP Guide. Retrieved from http://www.tcpipguide.com/free/t_TCPIPClientEphemeralPortsandClientServerApplicatio.htm

[12] Scripties van de Nederlandse Universiteiten. (n.d.). Retrieved May 17, 2016, from http://scripties.ned.ub.rug.nl/

[13] Threading in Unity. (2011, 26 oktober). Retrieved May 11, 2016, from http://answers.unity3d.com/questions/180243/threading-in-unity.html

[14] Turnhout, K. van, Craenmehr, S., Holwerda, R., Menijn, M., Zwart, J. P., & Bakker, R. E. N. (2013). TRIANGULATIE: EEN BASIS VOOR DE ONDERZOEKSLEERLIJN IN ICT EN MEDIA ONDERWIJS. Retrieved from http://nioc2013.icalab.nl/wp-content/uploads/2013/12/nioc-paper-definitief-KvT.pdf

[15] Unity Technologies. (z.j.). Unity documentation. Retrieved May 8, 2016, from http://docs.unity3d.com/

[16] Unity Technologies. (2015, April 17). Roll a Ball game. Retrieved May 8, 2016, from https://www.youtube.com/watch?v=RFlh8pTf4DU

[17] Voicu, A. (2015, January 16). Unity 3D - Control a robotic hand with Leap Motion and Arduino. Retrieved from https://www.youtube.com/watch?v=68jSTtdhiS0

[18] Wikipedia. (n.d.). TCP/IP. Retrieved May 8, 2016, from https://nl.wikipedia.org/wiki/TCP/IP