Web Simulation of a Thymio Robot

Quentin Flückiger (flucq1@bfh.ch)

November 12, 2019

Declaration of Authorship

Management Summary

Contents

1 Introduction

2 Environment

3 Thymio

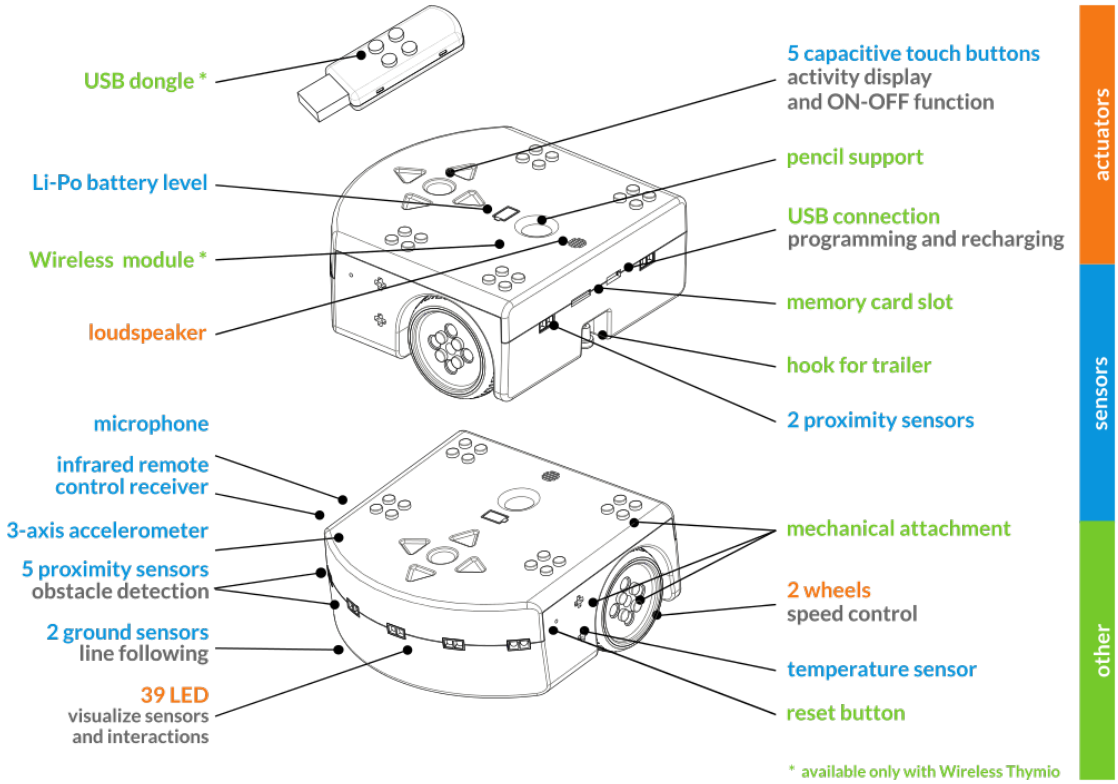
3.1 What is Thymio

Thymio is an educational robot that aims at improving early education (starting in primary school) in STEM (Science, Technology, Engineering and Mathematics), computational thinking, base computer science and researching the acknowledgement by kids of robots in their learning environment. The project also had technical aims, such as how to provide hardware modularity, fast reaction time amid perception and action, clear internal communication bus in a user-friendly way and streamline development for group robot, this includes direct changes to the robots' programs and parallel debugging wirelessly, transparently and cheaply.

The Thymio project is based on a collaboration between the MOBOTS group from the Swiss Federal Institute of Technology in Lausanne (EPFL) and the Lausanne Arts School (ECAL). MOBOTS being the Miniature Mobile Robots Group, they are mainly focused around system design for small robots of the kind. It started with a strange-looking pile of components, that were assembled on any kind of support and hold the name of "Monsieur Patate" (Sir Potato), most likely due to its appearance, that saw life during the first workshop between the two contributors. After what the first "Thymio" was developed, it was a four-block robot that could be self-assembled, but not self-programmed as it was coming with pre-programmed behaviours. It was used as a user study to gather feedback from clients to know what features needed to be implemented on the Thymio II.

From left to right, "Monsieur Patate", Thymio, Thymio II

The result is a robot with a complex and complete set of sensors and actuators. The National Centre for Competence in Research (NCCR) Robotics research program supported the development of the robot whereas Mobsya, a non-profit organization that creates a robot, software, and educational activities to broaden young people's mind about technology and science, oversees the production, distribution, and communication of said robot. Every step of the Thymio project is open-source and has a non-profit aim to enhance the quality of it with the user's project and research, and reduce the cost and augment the lifetime for educational platforms and materials.



Thymio II sensor and actuator

3.2 How does it works

As seen in the figure above there exist two Thymio models, Thymio and Wireless Thymio. The difference between them lies in the ability of the second one to be programmed wirelessly, as its name suggests. To begin the creation of a program for the robot there exist two possibilities. The first one, and the most common one for the public is done by using the software Aseba and a connected Thymio. In this case, the robot needs to be plugged in via USB cable or USB dongle (possible only if it is the Wireless Thymio) and powered on. Then the software can be used to connect to said robot and start to program in one of the four different programming languages, that are: VPL, Blockly, Aseba, and Scratch. Once the program is ready and sent to the robot it will be available to play.

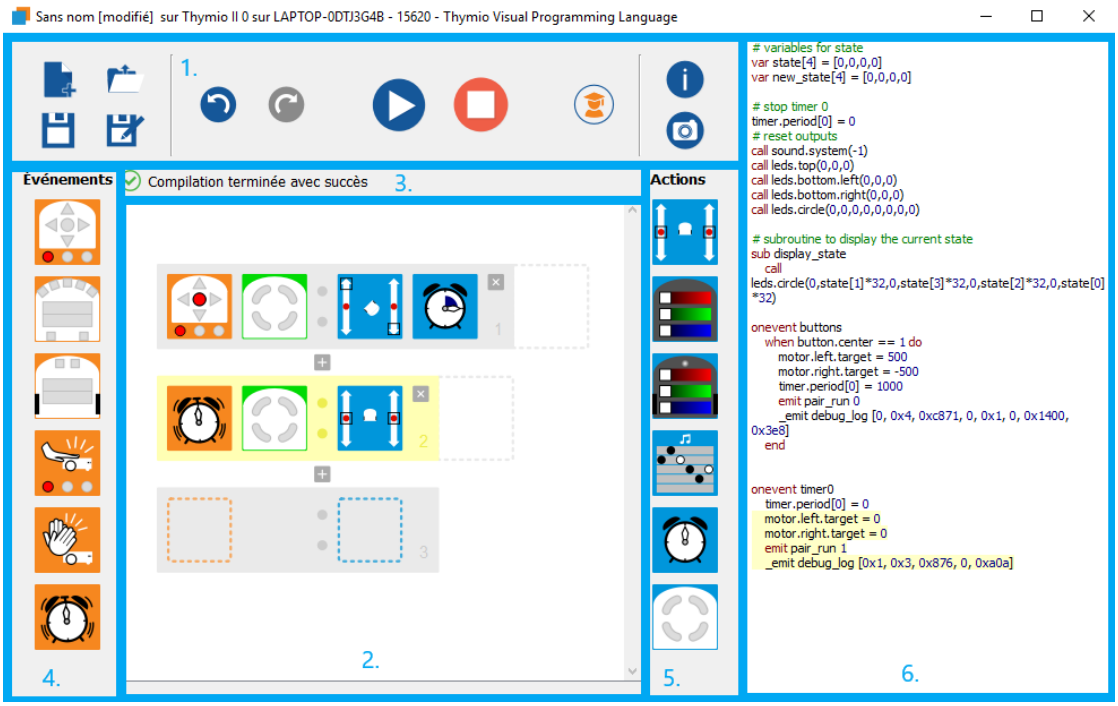
The second option is to use the work-in-progress Thymio Suite version. This software doesn't require a Thymio robot to be connected physically (or wirelessly) at all times as it has its own simulator built-in. The four said languages are still available, and one need to be chosen. After what comes the choice of connecting a physical Thymio or starting a simulation to emulate the programmed behaviour.

3.3 The different programming languages

3.3.1 VPL

One of the four different possibilities to program the Thymio is by using the visual programming language, or VPL, developed by the creator of Aseba. A visual programming language is an abstraction of the more common way to program. It is based on the manipulation of program elements graphically that can be manipulated following some spatial grammar to create a program. VPLs are based on a set of entities and relations, whereas most of the time entities are represented by boxes, or other graphical objects, and relations by simple arrows. They can be categorized into icon-based, form-based and diagram-based languages depending on the extent of visual expression inside of it. The use of visual programming languages can be found in multiple areas, such as the game engine "Unreal Engine 4" where their system of Blueprints is created upon a node-based VPL, or "Microsoft SQL Server Integration Services". This abstraction allows easier access for neophytes, for example using graphic elements such as blocks, forms, diagrams, and others reduce drastically, if not eliminate, the syntactic errors made by the user.

In the case of the VPL developed by Aseba's team, and the one we are mostly interested in, we have a programming language based on two types of blocks: Event blocks and Action blocks. From those two are built the seventeen, respectively eleven event blocks and six action blocks, entities. One of the main goals of VPL for Thymio was to let people who cannot yet read the ability to start programming and discover this world.



Thymio VPL Event and Action blocks

To begin creating a program follow the first steps described in the section 3.2 at the page 6. Once the VPL option has been chosen and the Thymio Visual Programming Language window appears we are ready to go. The window is split into six different regions with each of their purposes.

1. A tool bar
2. A programming window
3. Console messages
4. The event blocks
5. The action blocks
6. The program translated into AESL

Event blocks


Buttons are touched.
Red buttons are active.


Horizontal sensors detect an object.
White = an object is detected.
Black = No object is detected.


(Advanced mode) As above, but the slides can be used to set the thresholds.


Ground sensors detect light or dark.
White = a lot of reflected light is detected.
Black = little reflected light is detected.


(Advanced mode) As above, but the slides can be used to set the thresholds.


The robot has been tapped.


(Advanced mode) The robot has been tapped.


(Advanced mode) The pitch (forwards and backwards) of the robot is within the red segment.


(Advanced mode) The roll (left and right) of the robot is within the red segment.


The robot detects a loud noise.


(Advanced mode) The timer has counted down to zero.

Action blocks


Set the power of the left and right motors.
Move a slider up (forward)
or down (backwards).


Set the colour of the top of the robot.
Move the sliders to mix red, green and blue.


Set the colour of the bottom of the robot.
Move the sliders to mix red, green and blue.

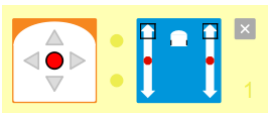

Play music.
Click on a bar to set a note.
White notes are longer than black notes.
Click on a note to change white ↔ black.
Click again to silence this note.


(Advanced mode) Start a timer in the range of 0–4 seconds.
Click on the clock face to set the time.


(Advanced mode) Set the current state.
Grey = do not change the value.
White = set to 0.
Yellow = set to 1.

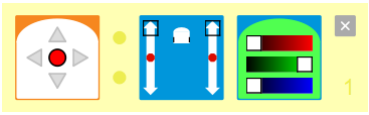
Thymio VPL Window

At first, the programming window will be empty of blocks, containing just a placeholder with empty slots. This placeholder is the base of every Thymio VPL program, it contains exactly one event block and one or more action block. This means that whenever the event of the event block happens then the set of actions added to this placeholder will occur at the same time. For example, with the following pair:



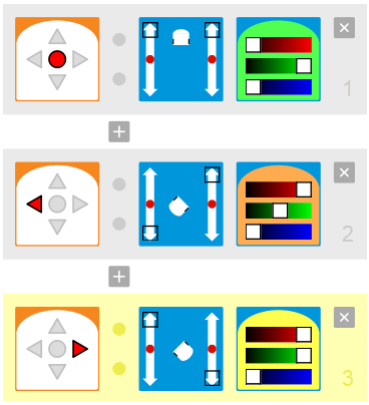
Event and one Action relation

Both wheels are powered to the maximum when the middle button is pressed. But more than one action can be attributed to one event, to do so simply drag another action block onto the previous pair, notice that the same block cannot be used twice for the same event. Here we turned the lights on top and set them to a complete green:



Event and multiple Actions relation

The maximum amount of action blocks we can add to an event is four, but we can add as many event blocks to our program as we want. Let us add two more event blocks to allow the robot to turn:



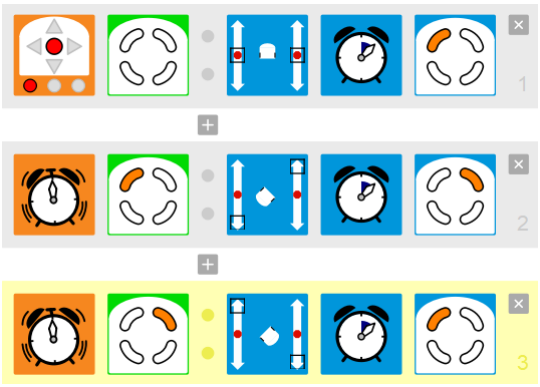
Event and Actions relations

Now we have a basic behavior, go straight with green lights when the middle button is pushed, turn left on itself with orange lights when the left button is pushed, and at last turn right on itself with yellow lights when the right button is pushed.

By clicking the button with a student as an icon we enable the advanced mode that gives us more possibilities for multiple blocks. It raises the amount of action block from four to six as well.

Let us refactor a bit the program from before, we will change the program by making the robot look left then right and starting over again using timers. To help us develop a more interesting program we have now access to a condition, a four led light on top of the robot, using this and the timer we can behave depending on the state of the robot. For example, hereafter the middle button was pressed a timer will start and after a short amount of time, it will light one particular led. Afterward, the event “timer elapsed” will be triggered but which pair should the program execute, turning right or turning left? Hence comes the use of the condition as we will execute the part of the program

that corresponds to the state of the condition light. In this example, it will go back and forth between the two pairs:



Advanced program

3.3.2 Blockly 4 Thymio

The second possibility is to use Blockly4Thymio which is an environment based on Blockly. Blockly was released in May 2012 and was initially a replacement for Open-Blocks for the MIT App Inventor. It is an open-source client-side library that allows its users to easily add a block-based visual programming language to an application or website. Blockly is not in itself a programming language but rather used to create one. Its design makes it flexible and it can support a large set of features. As it is a visual programming language, we find the same advantages as the first possibility, such for example applying programming principles with no regard towards syntactic error. Blockly is among the growing and most used visual programming environments because of a few important features. First, it can export the code generated with the blocks to one of the five following programming languages, as a built-in feature, JavaScript, Lua, Dart, Python, and PHP, and can be enhanced for any textual programming languages. The block pool can be expanded from its base pool or even reduced depending on the needs. The blocks are not restrained to only basic tasks and can implement sophisticated programming tasks. And it has been translated in over forty languages, and as well right-to-left versions.

Blockly includes a set of pre-defined blocks that can be used to develop with more ease the wanted application. They are arranged into eight families:

Logic: Blocks with Boolean definition, equality check, and conditions.

Loop: Blocks for loops.

Math: Blocks for numbers, arithmetic operation, a few basic math functions (for example cos, sin, square root) and some mathematical constant (Pi).

Text: Blocks to create text and text operations.

Lists: Blocks to create lists and standard list operation (length, get the value).

Color: Blocks with a color definition.

Variables: Blocks to create variables, and to set/get their values.

Functions: Blocks to create functions, with return value or not, and to call existing function.

Each block holds a pre-assigned shape, thus restraining its usage to certain situations as a "hidden" way to control the syntax. Their shapes are defined by the different connections with other blocks, both external and internal, while external blocks describe what happens after or before, the internals describe what happens during or what are the arguments, logic. Following is a basic variable block with three external connectors, and a math block with the value of one, with one connector, that is assigned to the `Count` variable (the blocks need to be assembled).

Variable block

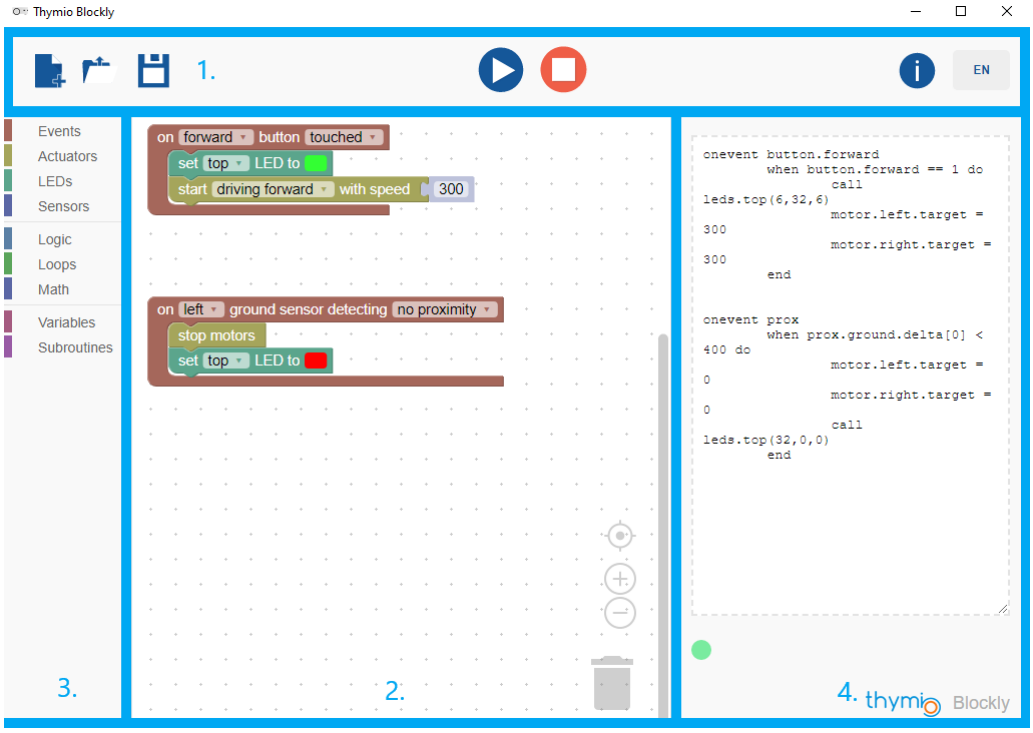
Using the same logic as above we created a `Limit` variable with the value of 5 to demonstrate the next example. The block used is from the logic family and test whether the `Count` variable is smaller or equal to the `Limit` as internal blocks. It can then be added to a loop, a function or other statements that needs logic.

Logic block

Added to the base that Blockly is for Blockly4Thymio, is a compilator that interpret and adapt the Blockly code directly into Aseba language, and an Aseba Framework. Let us once again follow the steps described in the "How does it works" section in order to start blockly-ing a little program with Blockly4Thymio. Note that it is possible to open the Thymio Blockly environment without going through the Thymio suite, and without any Thymio II connected (physically or simulated). To do so open the location of Thymio, the downloaded not the installed, and select thymio_blockly, and then index. The environment window that opens after choosing the Blockly option is split into four parts.

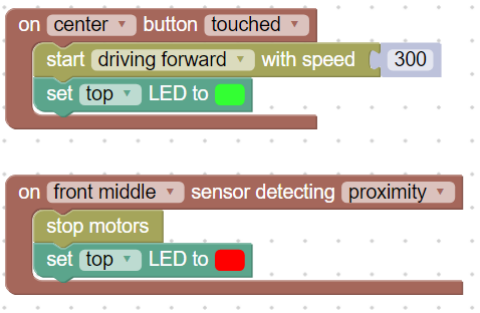
1. A tool bar
2. A programming window
3. The category of blocks

4. The program translated into AESL



Thymio Blockly window

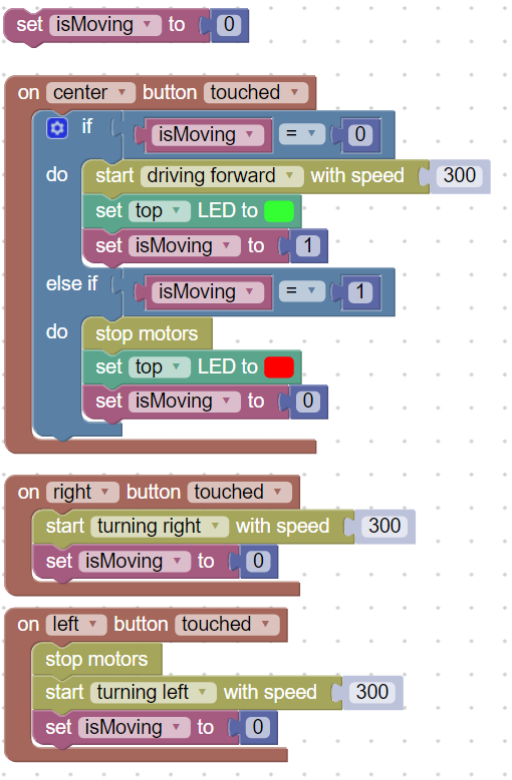
The following figure demonstrates a simple program, once run the program listens to two different events. When the center button is pressed and when the front middle proximity sensor detects a wall. The first one will activate the two motors at the same speed, as to drive forward, and light the top LED to green. Whereas the second will stop the motors and turn the LED to red.



Basic program

Here we set a variable to act as a control if Thymio is moving or not. We then use this information into a test when we click the middle button, and we either move forward or

stop according to the result. We added two other events for the right and left buttons that are responsible to turn the robot.



More complex program

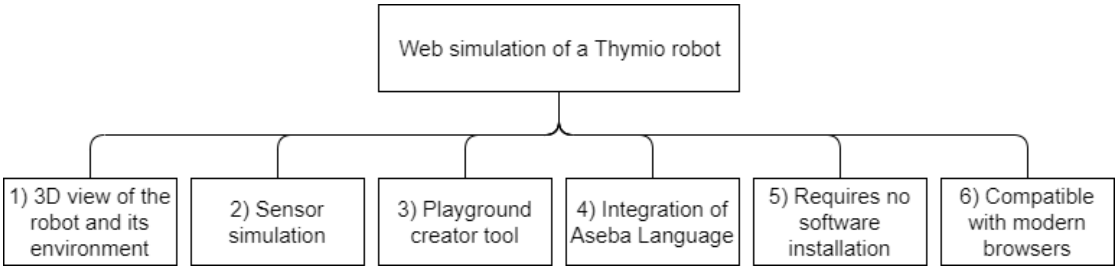
3.3.3 Aseba

3.3.4 Scratch

4 Requirements Documentation

4.1 Vision

4.2 Goals



4.3 System Boundaries

4.4 Requirements

4.5 Risk Analysis

Hack via .aesl as we execute the code if some code with malicious is fed we still execute it. What happens if part of the project aren't implemented ? How to bypass blockade? Security? No financial risks.

4.6 Stakeholder Descriptions

Product Owner Flückiger Quentin flucq1@bfh.ch *Interests*:

- The product owner wants to satisfy the customer.

Development Team Flückiger Quentin flucq1@bfh.ch *Interests*:

- The development team wants to develop a usefull application for the customer.

4.7 User Stories

Users User Stories
As a user, I want to uplaod an .aesl file, so that I can witness the simulated behaviour.
Description:
The user wants to upload an .aesl file to see the programmed behaviour simulated. Success:

- The simulation works.

Failure:

- The .aesl file doesn't contain a program.
- The .aesl file contains behaviour not included in the simulator.

As a user, I want to create a simple testing environment, so that I can diversify the experiences.
Description:
The user wants to create home made playground with a simple playground creation tool.
Success:

- The playground was successfully created and saved.

Failure:

- The created playground isn't saved properly.
- The user encounters trouble while creating the playground, be it meshes creation or placement.

As a user, I want to use the application without having to install anything, so that the application can be accessed easily.
Description:
The user wants to access and use the application without installing anything. Success:

- The use can start the application directly in his browser.

Failure:

- The webserver isn't accessible.

4.8 Use Cases Model

Use case diagram

5 What already exist

WeBots / Aseba logiciel

6 Three JS

7 Architecture

Domain class diagrams, Domain model, Package diagram, Sequence diagram, System sequence diagram

Write about the mvc, the interpreter pattern (?if used), problems because the three languages write files differently. All with the .aesl extension but not the same content format.

8 Usefull latex commands (to be deleted later on)

```
function generateBox(color, width, height, depth){

  var geometry = new THREE.BoxGeometry( width, height, depth );
  var material = new THREE.MeshPhongMaterial( { color : color} );
  var box = new THREE.Mesh( geometry, material );

  return box;
}
```

For that we read the article book of **Jerald:2015:VBH:2792790** which is very interesting, much more than the article **Diniz:2017:UGO:3100317.3100324**

9 Conclusion and future work

10 Appendices

10.1 Product Backlog

ID	Story Name	Story / Task Description	Priority	Est. Effort [h]	Update Effort [h]	Actual Effort[h]	Status
1	Create Documentation	Develop and write the documentation	High	108	52		In-Progress
2	Set up the Environment	Setting up and configuring the development environment	High	12	12	12	Done
3	Basic Learning	Learning and training of the different technology used later on	High	44	58	58	Done
4	Develop Playgrounds	Create playgrounds and function to generate meshes	High	40	48	48	Done
5	Update Documentation	Update existing documentation	High				In-Progress
6	Architecture Implementation	Refactor the existing code into the designed architecture	High	40	48		In-Progress
7	Web Deployement	Deploy the application on a webserver	High	16	12	12	Done
8	Basic UI	Implement a basic UI	Low	8	4		In-Progress
9	Behaviour Pipeline	Create pipeline to take .aesl file and translate/compile it into behaviour in javascript for the Thymio II	High				To Do
10	Phyisics Implementation	Implementation of Collisions for ThreeJS Meshes	High				To Do

11	Enhanced UI	Enhancement of the current UI	Low				To Do
12	Customizable Playgrounds	Implementation of a playground creator for users	High				To Do
13	Update Documentation	Update existing documentation and elaborate use cases and diagrams	High				To Do
14	Enhanced Behaviour Pipeline	Implement more sensors, action and event for Thymio II	High				To Do
15	Finish Documentation	Finish existing documentation	High				To Do
16	Prepare Defense	Prepare the defense	High				To Do
			Total				

10.2 Sprint Backlog

10.2.1 First Sprint

2019-09-16 until 2019-10-07

ID	Story Name	Story / Task Description	Priority	Est. Effort [h]	Update Effort [h]	Actual Effort[h]	Status
1	Create Documentation	Develop and write the documentation	High	16	16	16	Done
1.1	Template and Content	Choose a latex template and modify the content structure	High	8	8	8	Done
1.2	About Thymio	What is thymio and how does it work.	High	8	8	8	Done
2	Set up the Environment	Setting up and configuring the development environment	High	12	12	12	Done
2.1	GitHub	Create the project in GitHub and the Git environment	High	4	4	4	Done
2.2	Tools	Install Thymio Suite, NodeJS and download ThreeJS	High	8	8	8	Done

10.2.2 Second Sprint

2019-10-07 until 2019-10-28

ID	Story Name	Story / Task Description	Priority	Est. Effort [h]	Update Effort [h]	Actual Effort[h]	Status
3	Basic Learning	Learning and training of the different technology used later on	High	44	58	58	Done
3.1	ThreeJS	Read documentation and examples, and practice	High	16	20	20	Done
3.2	JavaScript	Update and deepen knowledge	High	8	8	8	Done
3.3	Thymio languages	Learning and using VPL, Blockly, Aseba and Scratch	Medium	24	30	30	Done
4	Develop Playgrounds	Create playgrounds and function to generate meshes	High	40	48	48	Done
4.1	Two Default Playgrounds	Generating two defalut playground to be choosen for the simulator	Medium	12	12	12	Done
4.2	Thymio Model	Create or load Thymio model	Medium	4	4	4	Done
4.3	Mesh Generation	Create function to generate meshes for the playgrounds	High	24	32	32	Done

10.2.3 Third Sprint

2019-10-28 until 2019-11-15

ID	Story Name	Story / Task Description	Priority	Est. Effort [h]	Update Effort [h]	Actual Effort[h]	Status
5	Update Documentation	Update existing documentation	High				In Progress

5.1	Four supported languages	Descibe and initiate to VPL, Blockly, Aseba and Scratch	High	32	24		In Progress
5.2	Backlogs	Create the Sprint and Project backlog	High	12	8	16	Done
5.3	Architecture	Create DCD, DM, PD, SD, SSD and proposition of architecture	High	20	4		In Progress
5.4	User Stories	Formulate the User Stories	Medium	8			To Do
5.5	Risk Analysis	Create risk analysis	High	12			To Do
6	Architecture Implementation	Refactor the existing code into the designed architecture	High	40	48		In Progress
6.1	Refactor Code	Refactor existing code into MVC Pattern	High	20	44	44	Done
6.3	Unit Testing	Write the JavaScript tests	High	12			To Do
6.4	JSDoc	Write the JavaScript-Doc	High	8	4	.	In-Progress
7	Web Deployement	Deploy the application on a webserver	High	16	12	12	Done
7.1	Virtual Machine Setup	Set up the Virtual machine	High	8	4	4	Done
7.2	WebServer	Create WebServer and publish it on bfh network	High	8	8	8	Done
8	Basic UI	Implement a basic UI	Low	8	4		In Progress
8.1	Pages UI	Create three pages UI, one for each of the following index, simulation and creation pages	Low	8	4		In Progress

10.2.4 Fourth Sprint

2019-11-15 until 2019-12-09

ID	Story Name	Story / Task Description	Priority	Est. Effort [h]	Update Effort [h]	Actual Effort[h]	Status

9	Behaviour Pipeline	Create pipeline to take .aesl file and translate/compile it into behaviour in javascript for the Thymio II	High				To Do
10	Phyisics Implementation	Implementation of Collisions for ThreeJS Meshes	High				To Do

10.2.5 Fifth Sprint

2019-12-09 until 2019-12-30

ID	Story Name	Story / Task Description	Priority	Est. Effort [h]	Update Effort [h]	Actual Effort[h]	Status
11	Enhanced UI	Enhancement of the current UI	Low				To Do
12	Customizable Playgrounds	Implementation of a playground creator for users	High				To Do

10.2.6 Sixth Sprint

2019-12-30 until 2020-01-18

ID	Story Name	Story / Task Description	Priority	Est. Effort [h]	Update Effort [h]	Actual Effort[h]	Status
13	Update Documentation	Update existing documentation and elaborate use cases and diagrams	High				To Do
14	Enhanced Behaviour Pipeline	Implement more sensors, action and event for Thymio II	High				To Do
15	Finish Documentation	Finish existing documentation	High				To Do
15.1	Create Video	Create the video file	High				To Do

15.2	Prepare Presentation Day	Create the poster and the presentation for the Presentation Day	High				To Do
15.3	Write in the Book	Write the page for the Book	High	8	4		To Do
15.4	Finish Writting Documentation	Terminate the writting part of the documentation	High				To Do
15.5	Prepare to Submit	Check spelling mistake, check images, print it, put the project on a USB stick	High				To Do

10.2.7 Seventh Sprint

2020-01-18 until 2020-01-end

ID	Story Name	Story / Task Description	Priority	Est. Effort [h]	Update Effort [h]	Actual Effort[h]	Status
16	Prepare Defense	Prepare the defense	High				To Do

10.3 Gantt Diagram

10.4 Version control

10.5 Configuration

User information
On demand.

Access the Windows Virtual Machine

	Linux WM -ssh	Windows VM - rdp
Windows	Putty	Remote Desktop Connection
Linux	terminal	Remmina
MacOS	terminal	Microsoft Remote Desktop

See link bellow for more information and links. (It requires to be inside the bfh networkd to access it) `https://intranet.bfh.ch/TI/fr/Studium/Bachelor/Informatik/Tools/VMsHowto/Pages/default.aspx?k=vm`

First Configuration of IIS manager

We followed the steps in this tutorial video in order to configure IIS. `https://www.youtube.com/watch?v=rPRLe7QeVHM`
Then we had to open the port in order to access it from within the LAN.

1. Open the windows Firewall, click on Inbound Rules and New Rule. This will open the New Inbound Rule Wizard.
2. Select the desired type, Port, click next.
3. Choose TCP and specify the port used, here 80, click next.
4. Select Allow connection, click next.
5. Select all three profile options, click next.
6. Add a Name and a description to this rule, click finish.

With this specified the website is now accessible from within the LAN at the following adress : http://147.87.116.44/Code/HTML

Additional setup

It was needed to create a web.config file and add a few file extension so that the .mtl and .obj would still be able to load. Otherwise we encountered an error of the type "Failed to load resource: the server responded with a status of 404 (Not Found)." The text that needed to be added to the web.config file is the following :

```
<?xml version="1.0" encoding="UTF-8"?>
  <configuration>
    <system.webServer>
         <staticContent>
           <remove fileExtension=".mtl" />
           <mimeMap fileExtension=".mtl" mimeType="text/plain" />
           <mimeMap fileExtension=".obj" mimeType="application/octet-stream" />
         </staticContent>
    </system.webServer>
</configuration>
```

10.6 Meetings

Date	Content
17.09.2019	**Kick Off meeting** - Documentation/Management - Technology to use : ThreeJS and Typescript - Setting up the goals
24.09.2019	**Second meeting** - Documentation language : English - Thymio model - Base talk about riks management
08.10.2019	**Third meeting** Workplace - Discussion on the choice of Windows as the Virtual Machine - Create a configuration file with the information of the VM - And an architecture proposal
15.10.2019	**Fourth meeting** - Which shapes and meshes should the user be able to create for his own custom playground - Problems with webserver, has to be accessible from outside the vm, so maybe switching from windosw to linux - Talk about the problem of thymio suite, that is the software allows the user to create programs only if a physical or a simulated one is plugged in
25.10.2019	**Fifth meeting** - Discussed using a Finite state machine to handle the events, but it may be too rigid so a non-deterministic finite state machine was the possible solution we came with - First little talk about the meeting with the expert, report

List of Figures

11 Problems encountered

- javascript not refreshing properly due to cache -> disable cache
- 3d Model not loaded on the webserver -> first tried to change the directory, then mixed two solution. Had to create a web.config file and add file extension for .mtl and .obj. https://stackoverflow.com/questions/41245938/web-server-cannot-find-mtl-file https://stackoverflow.com/questions/16097580/three-js-loading-obj-error-in-azure-but-not-locally
- shadow not rendering on plane of all playgrounds

- javascript file not found on server, net::ERR_ABORTED 404 (Not Found) => first solution (working partially) was to add a IIS_IUSRS.
- Thymio Blockly has trouble loading saved files. Using the software I wasn't able to load any .aesl file previously created with it, but I could load them if I used the index.html one.