Web Simulation of a Thymio Robot

Quentin Flückiger (flucq1@bfh.ch)

October 17, 2019

Declaration of Authorship

Management Summary

Contents

1 Introduction **5**

2 Environment **5**
2.1 Thymio 5
2.1.1 What is Thymio 5
2.1.2 How does it works 6
2.1.3 The different programming languages 7
2.1.4 What already exist 9
2.2 TypeScript 9
2.3 Three JS 9

3 Conclusion and future work **10**

4 Appendices **10**
4.1 Requirements Documentation 10
4.1.1 Vision 10
4.1.2 Objectives 10
4.1.3 System Boundaries 10
4.1.4 Requirements 10
4.2 Virtual Machine Configuration 10
4.3 Version control 10
4.4 Meetings 10

5 Problems encountered **11**

1 Introduction

2 Environment

2.1 Thymio

2.1.1 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 acknowledgment 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 holt 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.

Figure 1: 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.

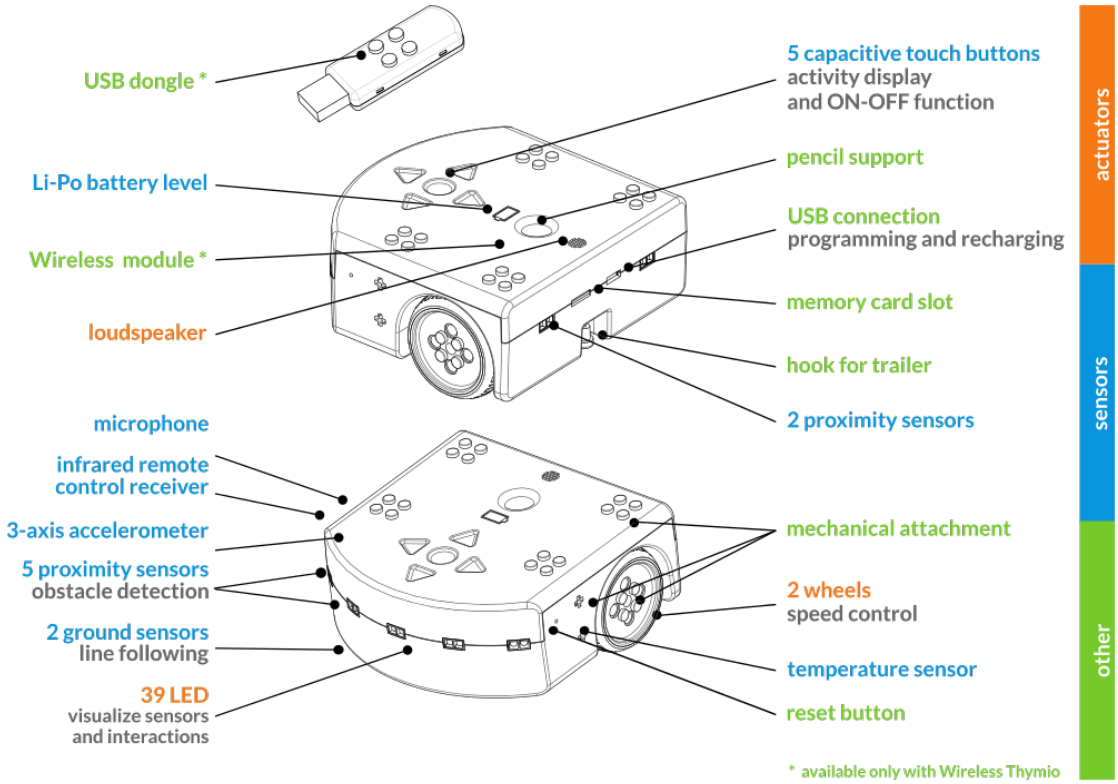

Figure 2: Thymio II sensor and actuator

2.1.2 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.

2.1.3 The different programming languages

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.

To begin creating a program follow the first steps described in the "How does it works" section. 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

Figure 3: Thymio VPL Event and Action blocks

By clicking the button with a student as icon we enable the advanced mode that gives us more possibilities for multiple blocks and as well the ability to assign multiple action to one event.

Blockly

Aseba

Scratch

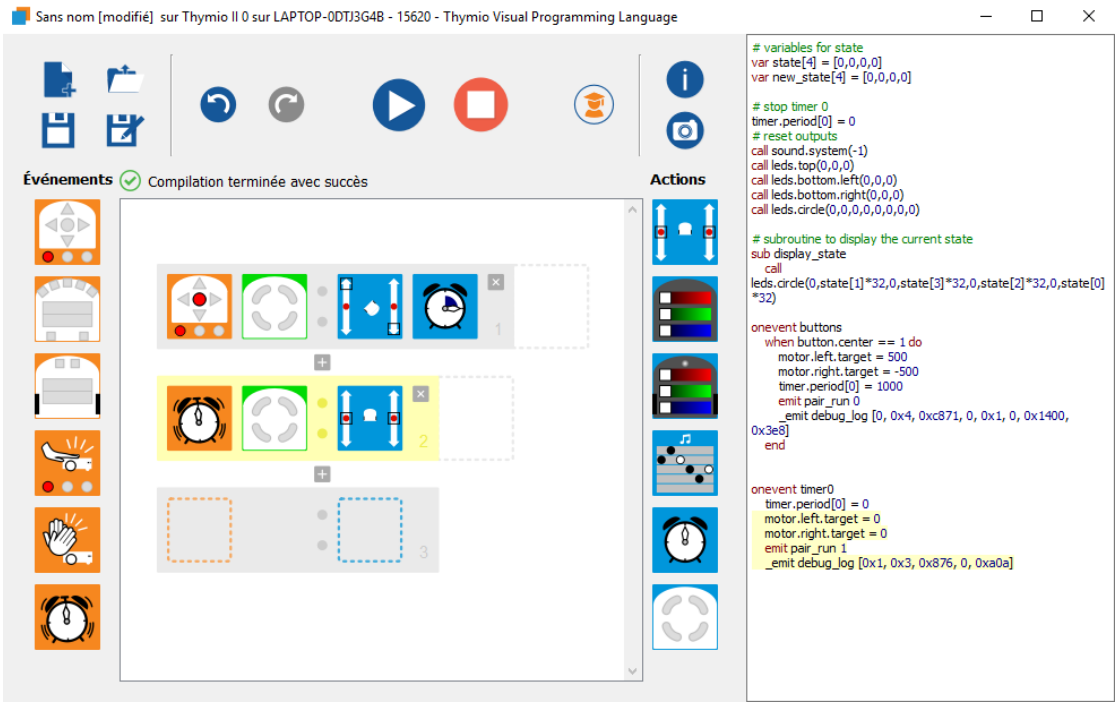

Figure 4: Thymio VPL Window

2.1.4 What already exist

WeBots / Aseba logiciel

2.2 TypeScript

2.3 Three JS

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**

3 Conclusion and future work

4 Appendices

4.1 Requirements Documentation

4.1.1 Vision

4.1.2 Objectives

4.1.3 System Boundaries

4.1.4 Requirements

4.2 Virtual Machine Configuration

Step by step Guide on how to set it up and use it.

4.3 Version control

4.4 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

List of Figures

1 From left to right, ”Monsieur Patate”, Thymio, Thymio II 5
2 Thymio II sensor and actuator 6
3 Thymio VPL Event and Action blocks 8
4 Thymio VPL Window 9

List of Tables

5 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