

Optimaliseren van Blockly4Arduino voor verbeterde leerprocessen

Robin Meurisse

Promotoren: prof. dr. ir. Sofie Van Hoecke, dr. Benny Malengier
Begeleider: dr. Maria-Cristina Ciocci (Ingegno)

Masterproef ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: elektronica-ICT

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. Koen De Bosschere

Vakgroep Materialen, Textiel en Chemische Proceskunde
Voorzitter: prof. dr. Paul Kiekens

Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2017-2018



Optimaliseren van Blockly4Arduino voor verbeterde leerprocessen

Robin Meurisse

Promotoren: prof. dr. ir. Sofie Van Hoecke, dr. Benny Malengier
Begeleider: dr. Maria-Cristina Ciocci (Ingegno)

Masterproef ingediend tot het behalen van de academische graad van
Master of Science in de industriële wetenschappen: elektronica-ICT

Vakgroep Elektronica en Informatiesystemen
Voorzitter: prof. dr. ir. Koen De Bosschere

Vakgroep Materialen, Textiel en Chemische Proceskunde
Voorzitter: prof. dr. Paul Kiekens

Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2017-2018



Voorwoord

Voor u ligt de scriptie 'Optimaliseren van Blockly4Arduino voor verbeterde leerprocessen'. Deze scriptie is een onderzoek naar de mogelijke vormen van optimalisatie alsook een studie over de mogelijke manieren van implementatie om het platform te verbeteren.

Dit werk is geschreven in kader van het behalen van het diploma Industrieel Ingenieur Elektronica-ICT: afstudeerrichting ICT, aan de Universiteit Gent, campus Kortrijk. Van 25 september 2017 tot 1 juni 2018 is er onderzoek en implementatie verricht om de gestelde onderzoeks vragen te behandelen en te beschrijven, zoals te lezen is in deze scriptie.

Bij deze wil ik graag mijn begeleiders Sofie Van Hoecke, Benny Malengier en Maria-Cristina Ciocci bedanken voor de goede begeleiding en ondersteuning tijdens deze masterproef. Daarnaast wil ik alle respondenten bedanken die mij hebben geholpen bij het verkrijgen van info voor mijn onderzoek.

Daarnaast bedank ik ook zeker de Universiteit Gent voor de opleiding die ik daar genoten heb en de mogelijkheden die ik gekregen heb.

Ten slotte wil ik mijn familie en vrienden bedanken voor hun morele en financiële steun tijdens de opleiding en het masterproef traject.

Robin Meurisse, juni 2018

Toelating tot bruikleen

“De auteur geeft de toelating deze scriptie voor consultatie beschikbaar te stellen en delen van de scriptie te kopiëren voor persoonlijk gebruik.

Elk ander gebruik valt onder de beperkingen van het auteursrecht, in het bijzonder met betrekking tot de verplichting de bron uitdrukkelijk te vermelden bij het aanhalen van resultaten uit deze scriptie.”

Robin Meurisse, juni 2018

Optimaliseren van Blockly4Arduino voor verbeterde leerprocessen

door

Robin MEURISSE

Scriptie ingediend tot het behalen van de academische graad van
MASTER OF SCIENCE IN DE INDUSTRIËLE WETENSCHAPPEN: ELEKTRONICA-ICT
Academiejaar 2017–2018

Promotoren: prof. dr. ir. Sofie VAN HOECKE, dr. Benny MALENGIER

Scriptiebegeleiders: dr. Maria-Cristina CIOCCI

Faculteit Ingenieurswetenschappen

Universiteit Gent

Vakgroep Elektronica en Informatiesystemen

Voorzitter: prof. dr. ir. Koen DE BOSSCHERE

Samenvatting

In deze scriptie wordt gepoogd antwoord te krijgen op de vraag hoe men het leerproces via een blokgebaseerde omgeving beter kan faciliteren. In het onderzoek dat wordt uiteengezet, worden verschillende opties onder de loep genomen en een oplossing voorgesteld om uit te werken en te integreren. Dit houdt hoofdzakelijk in dat de stappen die ondernomen moeten worden om een programma op een aangesloten hardware device te programmeren, zoveel mogelijk te automatiseren. Het onderzoek en de implementatie worden toegepast op het Blockly4Arduino-platform. Dit blokgebaseerde platform is namelijk ontworpen voor STEM en focus op het aanleren van programmeertechnieken. Blockly4Arduino wordt momenteel ingezet in workshops om de jeugd warm te maken voor technologie en hen te begeleiden in het aanleren van programmeren en aansturen van hardware. Dankzij de relevantie van het blokgebaseerde platform, is het de ideale gelegenheid om het onderzoek en implementatie toe te passen om dit platform te optimaliseren. Daarnaast worden andere aspecten van het platform onderzocht zoals het huidige assortiment aan blokken en het gebruik van tutorials. Voor beide zaken worden aanvullende opties besproken die het leerproces kunnen vereenvoudigen.

Trefwoorden

Blockly4Arduino; Optimalisatie; Compiler; Blokgebaseerd programmeren; Arduino software;

Optimizing of Blockly4Arduino to improve learning processes

Robin Meurisse

Supervisor(s): Sofie Van Hoecke, Benny Malengier, Maria-Cristina Ciocci

Abstract— This article explores the possibilities of how the block-based platform Blockly4Arduino, can be further optimized. The conducted research looks for optimization towards the back-end of the platform but also towards the current list of available blocks. After the research, a few solutions will be tested and implemented based on the results. The conducted research will aid the workflow to code and program an Arduino device from the browser with the Blockly4Arduino-platform.

Keywords— Block-based programming, Codebender.cc, Node.js-server, Arduino.cc, Chrome extension

I. INTRODUCTION

Due to the rising number of jobs and the need for technical profiles, there is a need to motivate the youth to choose for mathematics or science. That's the reason why the Flemish government launched a campaign to stimulate the youth to choose for a technical, technological, scientific or mathematical education (STEM). This thesis only focuses on one part of the broad range of subject that STEM offers: ICT. Workshops are being hosted that allow children (age 9 to 14) to program their first game with Scratch or to program a physical device like an Arduino. This is the part where Blockly4Arduino comes in. The platform offers a block-based interface to build and program code destined for an Arduino. However, the platform is only able to create code but not to compile it, the Arduino code needs to be compiled to a binary file so that the Arduino board can be programmed. At the moment, the users need to install other software to compile and upload the code.

In this abstract, we will first research the possibilities to add a compiler to the platform so that the users don't need to install extra software. Secondly, we will research the current list of used blocks and add some new ones to extend the possible programs a user can make with Blockly4Arduino. After the research, the best solutions will be implemented.

II. RESEARCH

A. Block-based vs Text-based programming

A lot of children have bad scores when they need to combine problem-solving with algorithmic thinking. The STEM promotion states that it is necessary to teach these skills from a young age. But a lot of teachers have trouble to efficiently teach these skills, this is where we can ask the question if the current method of teaching is good enough. A study [1] about analytic thinking, problem-solving and independence of children (age 10-14) finds that the students can master these skills through programming. The authors refer here to the block-based programming-environment Scratch, where the users can practise their problem-solving capabilities to build a program. Another study [2] focuses on using block-based environments to teach students logical thinking and problem-solving. The only

progress that was noticed was the improvement of problem-solving but not logical thinking. A second test the study did was researching the correlation of fun and programming in a block-based environment. The results were that the fun-factor was a strong motivator was to learn and program new things. The more fun the students had, the faster they were able to process new techniques and make progress.

In this paragraph, block-based programming will be further discussed and compared to text-based programming. From a recent investigation [3], these are some advantages block-based has over text-based:

1. Easy to read blocks;
2. No strange punctuations in code or syntax;
3. Shape and layout of the blocks is a huge help. Blocks can only be linked together if they have the right shape;
4. No occurrences of syntax-errors which means the users can focus on creating the algorithm;
5. Blocks are easier to handle than code, drag-and-drop is easier than writing;
6. Commands or functions don't need to be remembered. Users have a list of available blocks they can choose from.

A few disadvantages that block-based has in comparison to text-based:

1. Less powerful, the options are limited;
2. The bigger the program, the harder it is to see logic;
3. No understanding of a specific language. The blocks generate the code for you;
4. The use and architecture of the blocks limits the creativity of the user.

From the mentioned advantages and disadvantages can be learned that the block-based environment is perfect for users that don't have much experience. These users will create smaller programs, have more fun doing so with the blocks and can focus on creating programs due to not having to worry about syntax or errors. But the options are limited with the provided blocks, so for more experienced users it is strongly advised to make the step to text-based.

B. Google Blockly and Blockly4Arduino

Google Blockly is an open-source project designed by Google and comes with a wide range of predefined blocks. Blockly is not a programming language but rather a web-based programming-environment that generates code through blocks for JavaScript, Python and many more. Blockly4Arduino (figure 1) is built on top of Google Blockly and generates Arduino code. This environment was created to teach students problem-solving and programming skills through working with Arduino

boards. A lot of new blocks have been added to the platform to aid the students in programming their devices.

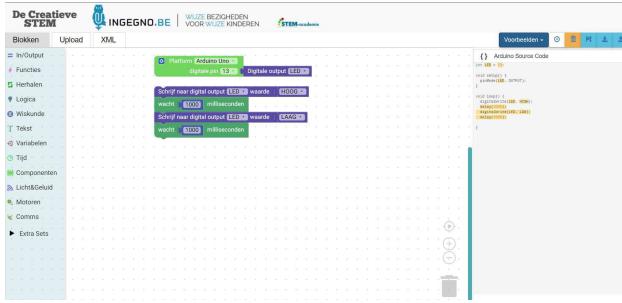


Fig. 1. Blockly4Arduino: Block-based-environment to generate Arduino code

C. Arduino code compiler

The Blockly4Arduino-environment is only able to generate Arduino code but not to compile or upload it to the device. The generated code needs to be compiled to transform the code to machine programmable language. Since the platform doesn't compile the code, there is a need for 3rd party software like the Arduino IDE or to a paying webservice like Codebender.cc. This also means that the users have a need to install and setup the extra software if they want to program their devices. This creates an extra step in the workshops and lectures before users can start programming. When the setup is complete, the users need to copy the generated code from the platform and paste it in the extra software to compile and upload their code. This workflow can be thoroughly optimised and therefore it could come in handy that Blockly4Arduino runs its own compiler. At this point, a research was performed to search the best solutions to implement an Arduino compiler. Out of the conducted research about possible compilers, there were 2 good options: implementation through Codebender or through the Arduino IDE. In section 'Implementation', the implementation-step of the compilers goes further into detail.

D. Adding extra blocks to Blockly4Arduino

Another investigation researched other block-based platforms and the list of blocks they offered. The goal of this investigation was to compute a list of useful blocks that the platform doesn't have yet. Out of the following list, there is one remark: for some future blocks it's already possible with the current blocks to program the functionality but the addition of a simpler block can aid inexperienced users. A next step could be to let the users try and make the same functionality but without the added blocks, to add some more complexity. The following list contains the most important blocks that aren't available yet:

1. Display blocks (e.g. LCD or OLED);
2. Detection blocks (e.g. Motion sensor, distance sensor, sound sensor, etc.);
3. Control blocks (e.g. adding a joystick);
4. Interrupts, schedule function,

E. Interaction and tutorials

From recent research [4] there are a lot of students that don't follow the tutorial step by step but rather look through the im-

ages instead of reading the text. When working with tutorials, we can conclude that it is necessary to add some form of visual context to make the steps clear and have a higher chance that the students can recreate a program on their own. Beside that, it has also been stated that the use of images creates another advantage: fun-factor. The students find it much more enjoyable to go through a list of pictures than to read a paragraph about how they should do it. In a previous section the fun-factor was already mentioned and how it increases the motivation of the students. The Blockly4Arduino-platform already offers some great visual tutorials, but we can still ask the question if there are ways to make the platform itself more interactive. These are some ways to do it:

1. Light up the next block needed for the program;
2. Display only the blocks that are necessary to build the program (don't display the full list of blocks);
3. Simulation of the code.



Fig. 2. Google Blockly Games example: maze

In figure 2 we see an example of Google Blockly Games where the platform only displays the blocks necessary to build a program and where they simulate the output. However, this platform increases in difficulty with each exercise which implements another form of interaction: difficulty. Since the Blockly4Arduino-platform focusses on physical computing, it is not necessary to implement simulation because the goal is to program a hardware device.

To help improve Blockly4Arduino, the following things can be done:

1. Adding difficulty through different type of blocks (e.g. one simple block or split into various blocks to code the same thing);
2. Listing of only the blocks that are needed for the exercise;
3. Figures in the blocks that describe the functionality of a block.

III. IMPLEMENTATION

A. Adding new blocks to Blockly4Arduino

In the previous section, we discussed the type of blocks that could still be added to the platform. Here the added blocks will be described but not into detail (see the main thesis for the full explanation and figures). A first block that was implemented was a button-block that uses the internal pull-up resistor of the Arduino board. This means no resistor has to be connected and the code changes a little bit. A second type that was added was to control a 7-segment display. To drive the display, 3 blocks had to be added: a control block to initiate to which pins each

segment was connected, a write number block to write a number between 0-9 and a single segment block that will turn on a single segment to create your own sequence. A third type was the implementation of blocks for an OLED display. To program an OLED display, IC communication is needed. There are in total 4 blocks for the OLED: an initialise block to choose the screen resolution, a font block to choose size of the font, a cursor block to set the starting pixel (figure 3) and to write a string at that position and lastly a write to display block that sends the configured text to the display.



Fig. 3. OLED block to write a string at the given position (cursor)

The last type was to read the distance value of a supersonic distance sensor. This sensor contains 2 pins, an Echo and a Trigger pin. The Trigger pin will emit a supersonic soundwave while the Echo pin will wait for the signal to come back (figure 4).

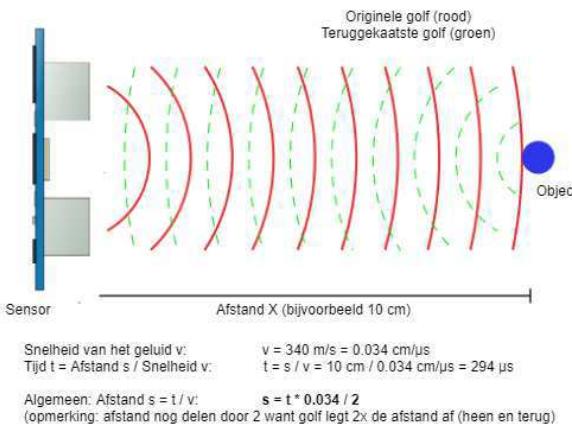


Fig. 4. Calculate the distance with the a supersonic sensor

A usability improvement that was done to the Blockly4Arduino-platform was a copy-button (figure 5). This is not a block but an added button to the toolbar that will copy the generated code to the clipboard. This allows for easy transfer between the platform and extra software with the guarantee that the user copied the whole thing.



Fig. 5. Added Copy-button to the toolbar

B. Optimisation workflow: Codebender Compiler

Codebender's architecture is displayed in the figure 6. This architecture consists of a webpage that contains a text-editor to

write code. The domain also contains a compiler to transform the code to machine language (.hex-file). Through a browser extension, it is possible to acces this webpage and to compile the code, the extension is also capable of flashing the connected hardware. This setup of Codebender is actually the setup we want to have for the Blockly4Arduino-platform because it enables the users of the platform to code, compile and upload their programs from within the webpage.

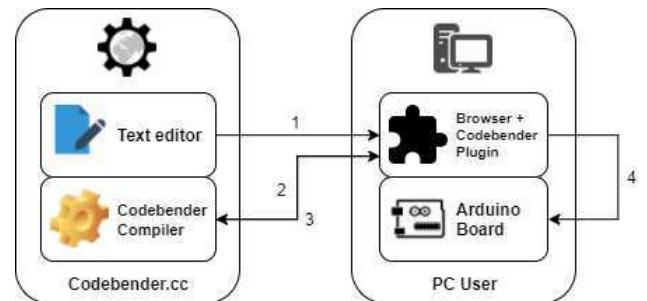


Fig. 6. Codebender.cc architecture

The Codebender projects are open-source available on Github and consist of 3 main projects: Compilerflasher, Compiler and Builder.

The first project: Codebender Compilerflasher, consists of a JavaScript-script that you include into your webpage and will make the connection to the Codebender extension to compile and upload the code. Codebender has updated in the meantime their website and extension which means the 4 year old script cannot be used anymore. The second project: Codebender Compiler, is a Symfony (PHP) project that creates a webserver where you can send a HTTP POST-request to and the server will compile the requested code and return a hex-file (figure 7).

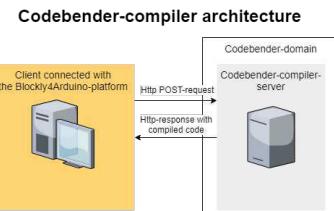


Fig. 7. Codebender Compiler server architecture

During the testphase, there were 2 problems with the project: libraries were not compiled or fetched so you had to send the complete library code with the request. This meant a library-management system had to be build on top of the compiler. A second problem was the Cross-domain requests which raises security errors in the modern webbrowsers. This problem could be solved with the addition of cross-domain related headers.

The third and last project: Codbender Builder (figure 8), this project is again a Symfony-project but with the ability to fetch requested libraries. On his turn, the Builder will send the code to the Codebender Compiler who will compile the whole sketch. This project could be the solution to the library management problem we had with the Codebender Compiler project. However, we ran into several problems. Due to outdated packages

and it being an old Symfony-project, it was hard to use the latest technologies and also the libraries didn't get fetched correctly.

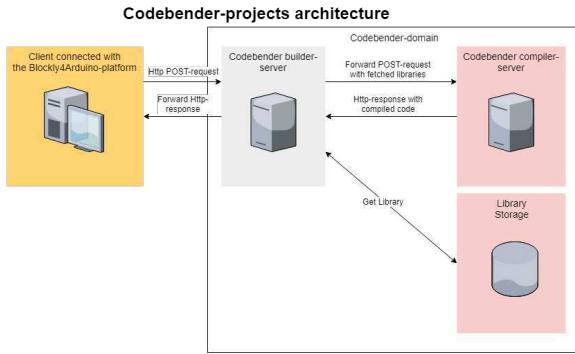


Fig. 8. Codebender Builder that communicates with Codebender Compiler to compile the code and a storage place were the libraries are stored

Conclusion: it would require a lot of time to find the fault and to upgrade/rewrite the projects with the latest technologies. Here the question was asked if there wasn't another solution available that could better fit our needs with an active developer team.

C. Optimisation workflow: arduino-builder

The Arduino IDE is the official environment to code and program Arduino code. Arduino also has an online tool called Arduino Create where users can code, compile and upload their code from their browser using an extension. Now both the IDE and the website use the same tool in the back-end: Arduino-builder. This tool is capable of building whole projects to compiling sketches and returning a hex-file. The Arduino-builder tool follows 4 steps to build a project (figure 9). The pre-processor will merge all files to one main sketch that will be compiled. The gcc-compiler will then compile the sketch to send it to the assembler. The assembler will transform the compiled code to machine language and lastly the linker will merge the libraries together to finish a hex-file.

The Arduino-builder is a command-line tool which means it can be executed through bash, but hence we prefer a webserver to compile the sketches. The next step was to build a webserver, the choice was made to build a server with Node.js because of the non-blocking code (call-back-functions), speed and performance, lightweight, concurrent request handling and native JavaScript and JSON. The flowchart of the build Node.js-server is explained in figure 10. First a website sends a HTTP POST-request to the server where the server will check the data. Then a temporary folder is created to store the output files of the compile process. The next step will be to execute the command-line tool with the requested code and boardtype and returns a hex-file. Lastly the temporary files will be deleted to save space. If any of the steps go wrong, an error message is send back, otherwise the returned hex-file is send back to the website.

D. Chrome extension

The big advantage that an extension offers is the access to the serial ports of the client pc. Since a couple of years, it was possible to do this with a few lines of JavaScript in a website script but obviously due to security reasons, this is no longer possible.

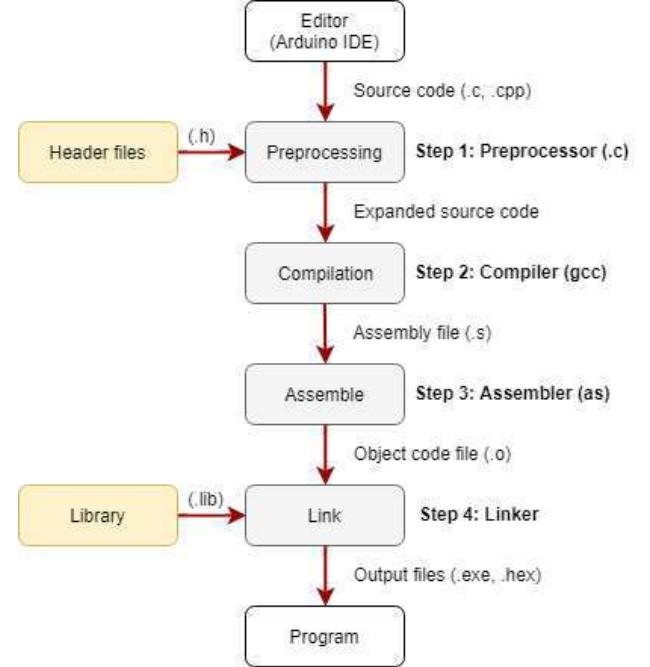


Fig. 9. Compilation steps of the Arduino-builder tool

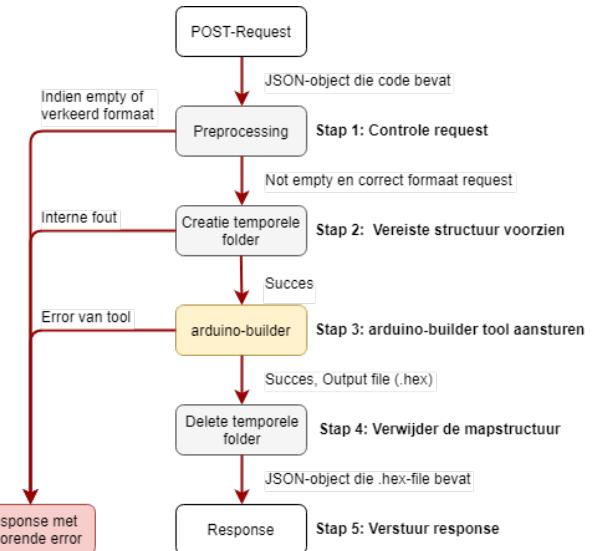


Fig. 10. Flowchart of the steps the Node.js-server will take to compile a request

What is possible however, is to use an extension with permissions to use the serial devices so that a browser can access the devices but not the website. This means that an extension needs to be built and communication between a webpage and the extension needs to be set up. First off, the extension is build with Google Chrome due to various API's available to access serial devices. Otherwise, a Node.js tool called AVRDUDE is used to flash connected hardware devices. The advantage this tool has over the popular AVRDUDE is the ability to run it with Node.js and to use it in an extension. In figure 11 the architecture of the Chrome extension is showed. The webpage will run a content-script that can communicate with a background-script run in the extension. That way, the extension has access to the Serial API

the Chrome browser offers to flash the hardware devices.

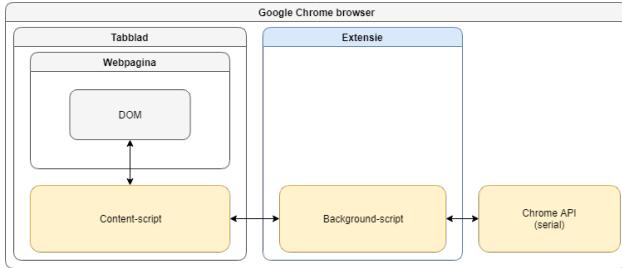


Fig. 11. Chrome extension architecture

E. Result of optimisation workflow

The results of previous sections is shown in figure 12. Here the Blockly4Arduino-platform will first send a HTTP POST-request to the Node.js server who will run the Arduino-builder tool to compile the code and return a hex-file. When Blockly4Arduino receives the hex-file, the hex-file is then send to the extension who will use AVRIGIRL to flash the connected hardware.

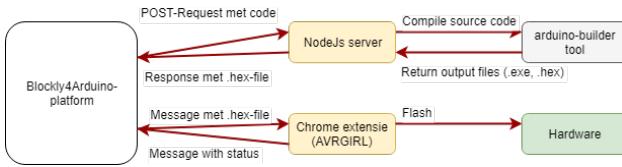


Fig. 12. Final architecture of the Blockly4Arduino-platform

Followingn figure 13 shows the implementation of added code to communicatie with the Node.js server and to compile the code. This can be done by selecting the type of board and the serial port and to verify or upload it.



Fig. 13. Final result: Compile code and flash the device through the extension from the Blockly4Arduino-platform

IV. EVALUATION

A. Codebender Compiler vs. Arduino-builder tool

For evaluation, both compilers were tested on speed and performance (figure 14). In the figure we can see that the Codebender Compiler is quite faster for more requests than the Node.js server who uses the Arduino-builder tool. Here we have to remark that the Arduino-builder tool performs extra steps to compile a complete project were the Codebender Compiler only is able to compile sketches and doesn't take into account libraries

or other files or certain checks. Overall, the Arduino-builder tool is the better solution because it still gets frequent updates and is more future proof than the outdated Codebender Compiler. It has not yet been optimised for remote compilation as the Codebender toolchain has over the last years.

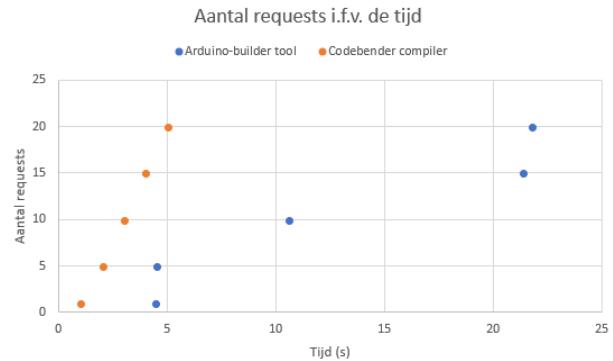


Fig. 14. Performance of both compilers tested

B. Evaluation of Blockly4Arduino

During the workshops, a questionnaire was given to a group of children who could answer some questions about Blockly4Arduino (for the questionnaires, see the master thesis). The findings were that they enjoyed using the Blockly4Arduino-platform but would rather not use the extra software (like the Arduino IDE). So, adding the functionality was quite useful. The Copy-code button that was added was received positively and is used a lot. The only questions that were not really answered positively was about saving or uploading programs. For some users it was not clear how to do it without help. Another point was how good the users could follow the tutorials, most users answered quite neutral on the subject. During examination of a workshop, it was quite clear the children were distracted fast and it was more a race to finish the project first than really doing the project good which led to steps being skipped and programs not working.

V. CONCLUSION

We conducted a study about how to increase the functionality of the Blockly4Arduino-platform by first researching why the platform was used and how. Following was a study about which blocks could be added and how the workflow could be optimised. During the implementation, some new blocks were added to program different kinds of hardware like: buttons, 7-segment displays, OLED displays and supersonic distance sensors. The main part of the implementation however was the creation of a compiler server, so the platform could compile the code itself. There were 2 steps: first the compiler of Codebender and second the Arduino-builder tool. Because the Codebender projects were outdated and the functionality wasn't working like it should be, the option was to implement another compiler. The Arduino IDE and Arduino Create use the Arduino-builder tool on the back-end to build their projects and to compile their sketches. A Node.js-server was then created to control and execute the Arduino-builder tool. The last step in the implemen-

tation was to create a Chrome extension because with an extension, the platform could have access to the serial devices to program a connected Hardware device. The chosen flasher-tool was AVRGIHL. The evaluation section tested both compilers on speed but overall the Arduino-builder compiler was chosen, even though it is slower. From a questionnaire about user experience on the Blockly4Arduino-platform, we only got good results about the functioning and working of the platform and it's added features. The only downside was saving and uploading stored programs, these steps were not so clear. This could be investigated in future work. Another thing that could be investigated is the load the hard drive gets because with every request, the Arduino-builder tool will create and write some files between each step. The server will delete these files to limit the space used but this could wear down the lifespan of the hard drive.

REFERENCES

- [1] Giordano, D., Maiorana, F., *Teaching algorithms: Visual language vs flowchart vs textual language*, 2015 IEEE Global Engineering Education Conference (EDUCON), pp. 499-504, Tallinn, Estonia, Mrt. 2015.
- [2] Kalelioglu, F., Gibahar, Y., *The effects of teaching programming via scratch on problem solving skills: A discussion from learners' perspective*, Informatics in Education, pp. 33-50, Vilnius, Lithuania, Jan. 2014.
- [3] Tamilias, A., Karvounidis, T., Garofalaki, T., Kallergis, Z., Blocks for Arduino in the Students Educational Process, 2017 IEEE Global Engineering Education Conference (EDUCON), pp. 910–915, Athens, Greece, Apr. 2017.
- [4] Fraser, N., *TTen things we learned from Blockly*, 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond), pp. 49–50, Atlanta, USA, Oct. 2015.

Inhoudsopgave

Voorwoord	iv
Toelating tot bruikleen	v
Overzicht	vi
Extended abstract	vii
Inhoudsopgave	xiii
1 Probleem -en doelstelling	1
2 Onderzoek	4
2.1 Blokgebaseerd programmeren	4
2.1.1 Google Blockly en Blockly4Arduino omgeving	8
2.2 Compiler voor Arduino-code	9
2.3 Creatie van extra blokken voor Blockly4Arduino	13
2.4 Interactie en tutorials voor Blockly4Arduino	14
2.4.1 Wat bestaat er al?	15
2.4.2 Hoe te implementeren?	18
2.4.3 Toegepast op Blockly4Arduino	20
2.5 Blockly4Arduino varianten	20
3 Implementatie	23
3.1 Uitbreiden assortiment aan blokken	23
3.1.1 Drukknop met interne pull-up weerstand	23
3.1.2 7-segment display	24
3.1.3 OLED display	26
3.1.4 Supersonische afstandssensor HC-SR04	28
3.1.5 Kopieer-knop	31
3.2 Optimalisatie workflow	32
3.2.1 Introductie naar integratie compiler	32

3.2.2	Implementatie via Codebender	32
3.2.3	Implementatie via Arduino.cc	44
3.2.4	Google Chrome extensie	54
3.2.5	Eindresultaat implementatie voor verbetering workflow	59
4	Evaluatie	61
4.1	Verschillen tussen de uitgewerkte oplossingen	61
4.2	Evaluatie toegevoegde blokken	62
5	Besluit en toekomstperspectieven	65
A	Vragenlijst 7-segment component	68
B	Vragenlijst Gebruikservaring	70
C	Handleiding voor het opzetten van de Compiler server en Chrome extensie	72
Bibliografie		80
Lijst van figuren		83
Lijst van tabellen		86

Hoofdstuk 1

Probleem -en doelstelling

Door het dalende aantal leerlingen die afstuderen in een wetenschappelijke richting en de toenemende jobs in wiskunde, exacte wetenschappen, techniek en technologie, is er nood aan stimulans. Daarom lanceerde de Vlaamse regering samen met het onderwijs een campagne om meer leerlingen voor een STEM-richting te laten kiezen. STEM is een internationaal letterwoord dat staat voor een waaier aan technologische, technische, exact-wetenschappelijke en wiskundige opleidingen en beroepen.

Specifiek voor STEM is het belangrijk om al op jonge leeftijd kennis te maken met praktijkvoorbeelden en toepassingsmogelijkheden van wetenschappen. Naast de onderwijsactiviteiten op school worden ook buitenschools workshops gehouden om kinderen van alle leeftijden en achtergronden, spelenderwijs te onderwijzen in techniek, wetenschap en technologie.

De STEM-opleidingen bevatten een heel brede waaier aan onderwerpen maar deze scriptie zal focussen op een onderdeel, namelijk techniek en programmeren. Onder deze tak worden workshops georganiseerd op verschillende manieren, lopende van het programmeren van spelletjes in Scratch of het ontwerpen van figuren om te 3D-printen tot het programmeren en aansturen van hardware (physical computing). Ingegno in samenwerking met De Creatieve STEM vzw, bieden een waaier van deze workshops aan om kinderen (9 tot 14 jaar oud) enthousiast te maken over technologie.

Bepaalde workshops van Ingegno draaien rond physical computing en de deelnemers programmeren Arduino boards om zo hardware aan te sturen. Het gebruik van Arduino vereist de installatie van software alsook bibliotheken om aan de slag te kunnen. Mits de kinderen hun eigen materiaal meebrengen moet dit tijdens de sessie worden opgezet ten koste van

de tijd om het programma te leren. Daarnaast is de Arduino IDE text-gebaseerd en is het niet ideaal om kinderen (zonder ervaring) erop los te laten en al meteen resultaat te verwachten. Daarom hebben de oprichters van Ingegno een omgeving gebouwd die gebaseerd is op Google Blockly. Met deze omgeving moeten de kinderen een programma bouwen aan de hand van blokken in plaats van tekst waardoor er tijdens de workshops kan gefocust worden op het uitwerken van een oplossing en de moeilijkheid niet ligt bij het gebruik van het programma, de syntax of het al dan niet kunnen snel typen. Het bouwen van een programma aan de hand van blokken genereert de code achterliggend zodat de kinderen enkel maar de gegenereerde code hoeven te kopiëren en plakken.

Eén probleem blijft echter bestaan: via de website kan enkel de code gegenereerd worden. De code moet dan nog geupload worden naar de Arduino, de Arduino IDE of een 3th party website is nodig om de code te kunnen uploaden naar de Arduino. Dit zorgt ervoor dat de kinderen meer dan één applicatie nodig hebben om de opdrachten te kunnen uitvoeren. Het vereenvoudigen van deze laatste stap zou het verlopen van de workshops en uitwerken van de oefeningen vlotter maken.

In deze scriptie wordt ingegaan op het optimaliseren van de Blockly4Arduino omgeving om het leerproces vlotter te doen verlopen. Hierbij wordt de onderstaande onderzoeksraag gesteld.

De gestelde onderzoeksraag: Onderzoek (en implementeer) een manier om in de Blockly4Arduino-omgeving de gegeneerde code te kunnen compileren en uploaden naar een Arduino board.

Naast de centrale onderzoeksraag, worden nog enkele deelvragen gesteld omtrent de Blockly4Arduino-omgeving.

1. Onderzoek het kunnen van de huidige versie en implementeer nieuwe en gebruiksvriendelijke functionaliteit door bv. het toevoegen van extra blokken.
2. Onderzoek het gebruik van tutorials en moeilijkheidsgraden om de omgeving te kunnen verbeteren zodat kinderen van thuis uit aan de slag kunnen. Daarbij is het nuttig om verschillende graden te kunnen voorzien in de oefeningen zodat gebruikers van alle leeftijden en niveaus aan de slag kunnen.

In eerste instantie werd een literatuuronderzoek verricht om de gestelde onderzoeks-vragen, beschreven in Hoofdstuk 2. Op basis van het uitgevoerde onderzoek werd verder gebouwd om de implementatie stap voor stap uit te werken in Hoofdstuk 3. De implementatie bestaat uit het opzetten van een code-compiler alsook het uitbreiden van het assortiment aan blokken op het Blockly4Arduino-platform. Na de implementatie werd ook een evaluatie uitgevoerd op de gemaakte blokken en een test van de opgezette compiler, beschreven in Hoofdstuk 4. Ten slotte wordt afgesloten met de conclusie en in de appendix van deze scriptie zijn de enquêtes terug te vinden alsook de documentatie voor het installeren en opzetten van de gemaakte software en server.

Hoofdstuk 2

Onderzoek

Dit hoofdstuk handelt enerzijds over de mogelijke uitbreidingen en opmerkingen aan de huidige structuur van Blockly4Arduino. Anderzijds worden de oplossingen voorgesteld per onderzoeksraag, gebaseerd op onderzoek, bestaande literatuur en architecturen.

2.1 Blokgebaseerd programmeren

Heel veel jongeren scoren slecht als het komt op de combinatie van probleem-oplossend denken en algoritmisch denken. De hervormingen van de laatste jaren maken duidelijk dat vandaag het kunnen programmeren als een vaardigheid gezien wordt die van jongens af aan moet aangeleerd worden. Ook veel lesgevers hebben moeilijkheden om programmeren en logica aan te leren. Vandaar dat deze problemen de vraag oproepen of de manier van les geven wel geschikt is. Daarnaast kan ook de vraag gesteld worden: indien het al moeilijk is in het hoger onderwijs, wat is dan de juiste strategie om les te geven in het lager -en middelbaar onderwijs? Onderzoeken (Kumar, 2014; Kalelioglu, 2012) [1], [2] verwijzen naar de programmeeromgeving Scratch voor het benadrukken van vaardigheden zoals kritisch denken, probleemoplossend vermogen en zelfstandigheid. Zij stelden vast dat wanneer leerlingen gebruik maken van een programmeertaal zoals Scratch, ze al deze vaardigheden verwerven omdat ze al snel problemen tegenkomen die ze moeten oplossen om verder te kunnen met het programma.

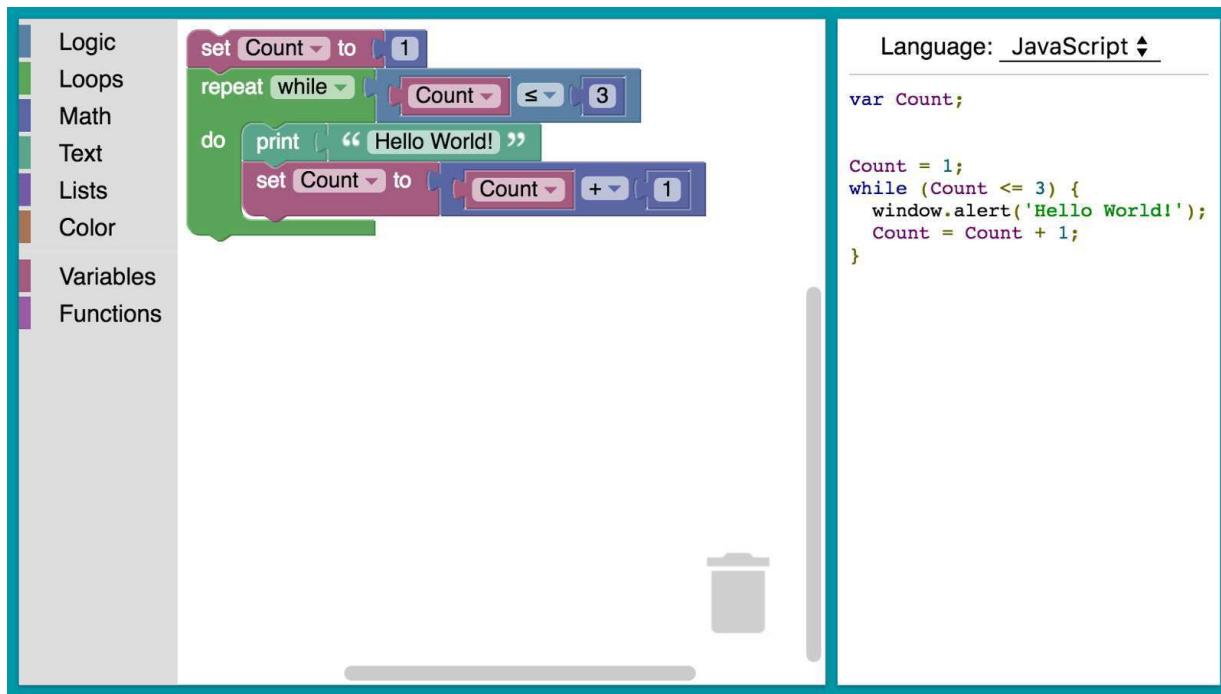
De volgende scripties (Davis, 2013; Shih, 2017; Ciocci, 2013; wyffels, 2014) [3], [4], [5], [6] hebben de efficiëntie onderzocht van het gebruik van blokgebaseerd programmeren in onderwijsomgevingen om kinderen probleem-oplossend denken aan te leren. In eerste instantie

werd het effect van visueel programmeren onderzocht op het probleem-oplossend denken en logisch redeneervermogen. Er bleek vooral vooruitgang te zijn bij probleem-oplossend denken maar niet bij logisch redeneervermogen. De conclusie was dat een blokgebaseerde omgeving nuttig is om probleem-oplossend denken aan te leren. Een tweede vaststelling was het verband tussen het aanleren van technieken via een blokgebaseerde omgeving en de mate waarin lagere schoolkinderen het leuk vinden om te gebruiken. Hieruit bleek dat beide zaken op elkaar inwerken, hoe leuker de leerlingen het vonden, hoe sneller ze de technieken aanleerden en omgekeerd. Het is dus belangrijk bij het ontwerpen van een omgeving om rekening te houden met de emotionele toestand en cognitieve vaardigheden van de gebruiker. Algemeen gezien kan gesteld worden dat programmeren het probleem-oplossend vermogen kan versterken bij leerlingen van alle leeftijden. De nood komt om een educatief platform te ontwikkelen waar deze vaardigheden kunnen aangeleerd worden. Aan de andere kant moeten we ook rekening houden dat het voorzien van een platform niet genoeg zal zijn om efficiënt probleem-oplossend denken aan te leren en op andere manieren ingezet zal moeten worden voor het logisch en algoritmisch denken.

Blokgebaseerde programmeeromgevingen zijn aan een sterke opmars bezig als een manier van lesgeven aan individuen zonder enige voorkennis in programmeren. In deze omgevingen, worden blokken gebruikt om de instructies te representeren waarmee de gebruikers deze blokken kunnen verbinden om een programma te bekomen (figuur 2.1).

Uit recente onderzoeken (Giordano, 2015; Tamlias, 2017; Ferrari, 2016) [7], [8] en [9] blijkt dat blokgebaseerd programmeren zijn voordelen heeft ten opzichte van tekstgebaseerd programmeren waardoor de voorkeur uitgaat naar een blokgebaseerde omgeving bij nieuwelingen. Het grote voordeel is dat blokgebaseerd programmeren eenvoudiger is om onderstaande redenen.

1. Blokken zijn eenvoudiger te lezen. Gebruikers identificeren de omschrijving als nuttig en kunnen de functionaliteit afleiden van de omschrijving;
2. Het gebrek aan onverstaanbare punctuatie in coderegels;
3. De vorm en layout van de blokken is een grote hulp. De hoofdreden hier is dat de gebruiker geen blokken kan verkeerd plaatsen omwille van de vorm. Via de vormen kunnen ze ook afleiden hoe de sequentie moet opgebouwd worden;
4. Syntax errors komen niet voor waardoor de gebruiker kan focussen op het creëren van het algoritme;



Figuur 2.1: Google Blockly als voorbeeld van een blokgebaseerde programmeeromgeving die achterliggend code genereert in een specifieke taal (hier JavaScript).

5. Blokken zijn eenvoudiger om mee om te gaan. Drag-and-drop commando's zijn in gebruik makkelijker dan typen;
6. Blokken als geheugensteun. De commando's of coderegels moeten niet onthouden worden zoals in text-gebaseerd programmeren. De gebruikers kunnen alle blokken overlopen in een gestructureerde manier wat de nieuwelingen in programmeren een hulpsteun biedt.

Dankzij bovenstaande redenen, wordt blokgebaseerd programmeren aanzien als een perfecte tool voor beginners. Maar waarom enkel voor beginners? Uit onderzoek blijkt dat blokgebaseerd programmeren ook enkele nadelen heeft ten opzichte van tekstgebaseerd programmeren.

1. Minder flexibel. Tekst-gebaseerde programmeertalen zijn krachtiger omdat ze tot meer in staat zijn;
2. Binnen een groot programma kunnen blokken moeilijk te onderhouden zijn. Hoe groter of complexer de code, hoe meer tijd het ook in beslag zal nemen om het programma te implementeren in een blokgebaseerde omgeving;

3. Niet authentiek. Via de blokken schrijft de gebruiker zelf geen code in een specifieke taal, de code wordt gegenereerd door de blokken;
4. Structuur van de blokken beperkt de creativiteit van de programmeur.

Nu de voor -en nadelen van blokgebaseerd programmeren behandeld zijn, wordt er gekeken wat de voorwaarden zijn om van een blokgebaseerde omgeving te kunnen spreken (Tamilias, 2015) [8]. Wat definieert dus een blokgebaseerde programmeertaal.

1. Standaardisering: de blokken moeten voldoen aan een gestandaardiseerd model dat attributen, variabelen, functies en klassen kan bevatten;
2. Onafhankelijkheid: de blokken moeten compileerbaar zijn zonder het bestaan van andere blokken;
3. Combineerbaarheid: de blokken moeten duidelijk de interne acties tonen met interfaces naar externe interacties en toegang tot aangrenzende blokken;
4. Inzetbaar: de blokken moeten kunnen functioneren als een op zichzelf staande entiteit;
5. Documentatie: de blokken moeten een gedocumenteerde bibliotheek bevatten voor de functies of klassen;
6. Testbaar: er moet een zekerheid zijn over de blokken zodat ze voldoen aan de gebruikers wensen en noden.

Vele blokgebaseerde programmeeromgevingen voldoen aan bovenstaande requirements maar zijn daarom niet hetzelfde. Na onderzoek over verschillende gerelateerde omgevingen, blijken er toch heel wat verschillen en gelijkenissen te bestaan. Toch blijft er nog heel wat ruimte over om een unieke omgeving te creëren waarin gebruikers kunnen programmeren. Later in de studie worden meer omgevingen en varianten besproken met de voor -en nadelen, zie Sectie 2.5

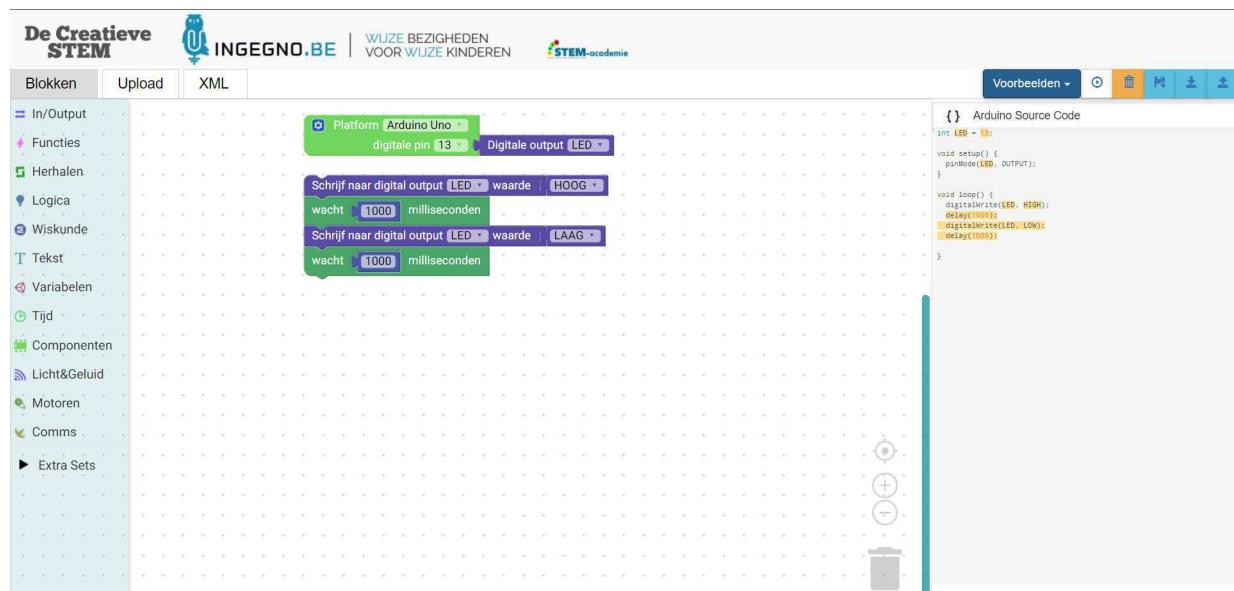
De onderstaande lijst geeft enkele belangrijke gelijkenissen terug.

1. Doelgroep: vele omgevingen focussen op een jong publiek maar vooral op nieuwelingen qua programmeren;

2. Structuur: allemaal reflecteren ze de syntax en structuur van bestaande programmeertalen;
3. Situering: de omgevingen situeren zich in een multimedia context met een focus op culturele relevantie. Bovendien integreren gebruikers kunst, muziek en interactie in hun projecten die leiden tot de creatie van games, verhalen en applicaties.

2.1.1 Google Blockly en Blockly4Arduino omgeving

Blockly is een open source project ontworpen door Google en komt met een groot aanbod aan voorgedefinieerde blokken zoals controle blokken, wiskundige functies, enz.. Blockly is echter geen programmeertaal maar een web-based visuele programmeeromgeving die code genereert voor specifieke programmeertalen zoals JavaScript, Python en meer. Gebruikers van het platform kunnen blokken kiezen en connecteren om een algoritmische oplossing te bekomen die code genereert, deze code stelt dan de oplossing voor in die specifieke programmeertaal. Google voorziet ook een API waarmee developers aan de slag kunnen om nieuwe blokken te maken of andere code te genereren (Celic, 2015; Liang, 2016; Ashrov, 2015) [10], [11], [12]. Hierdoor zijn er veel varianten van Blockly te vinden op het internet. De dubbele weergave van zowel blokkencode als tekst-gebaseerde code is een zeer interessante en unieke omgeving die geschikt is voor een educatieve omgeving.



Figuur 2.2: Blockly4Arduino: blokgebaseerde programmeeromgeving die achterliggend Arduino code genereert.

Blockly4Arduino is een blokgebaseerde programmeeromgeving die afstamt van ArduBlockly die op zijn beurt afstamt van Google Blockly. Deze web gebaseerde omgeving laat toe om via blokken een programma te schrijven die specifiek Arduino code genereert. Deze web omgeving kan gebruikt worden in elke moderne browser wat de barrière om toe te treden drastisch verlaagt. Om de omgeving simpel en gebruiksvriendelijk te houden, zijn enkel de hoofdkenmerken van Google Blockly en object-georiënteerd programmeren overgenomen. Daarnaast zijn er veel nieuwe blokken toegevoegd om het aansturen van hardware te vereenvoudigen via Arduino code.

2.2 Compiler voor Arduino-code

De Blockly4Arduino omgeving is een web applicatie die een gebruiker in staat stelt om via blokken een programma te maken die wordt omgezet naar Arduino code. Eenmaal de Arduino code gegenereerd is, moet deze ook nog op het Arduino bordje geplaatst worden. De Arduino wordt gekoppeld aan de gebruikers' computer waardoor de web applicatie niet rechtstreeks toegang kan krijgen tot deze Arduino. Hier komt de eerste onderzoeksvergadering sprake: hoe kunnen we de koppeling met de Arduino optimaliseren. De koppeling van Blockly4Arduino met een aangesloten Arduino kan op 2 manier gebeuren:

1. Via lokaal te installeren software, zoals de Arduino IDE;
2. Via een browser extensie die toegang vraagt tot de seriële poorten, zoals Codebender.cc (Amaxilatis, 2014) [13].

De Arduino IDE is een software-omgeving die lokaal op de computer geïnstalleerd moet worden. Via deze software wordt de koppeling gemaakt met een Arduino board en is het mogelijk om de code in de text-editor te builden en te uploaden naar het bordje. Ook een seriële monitor is beschikbaar om de output te kunnen weergeven. Het gebruik van deze software vereist een lokale installatie en bij gebruik van bibliotheken is het niet zo gebruiksvriendelijk om deze te importeren. Daarnaast moeten gebruikers van Blockly4Arduino de gegenereerde code nog kopiëren en plakken in de text-editor van de Arduino IDE wat voor een extra stap zorgt. Bijgevolg is deze software minder geschikt om snel op te zetten in een educatieve omgeving bestaande uit korte workshops en zijn er extra stappen nodig om het gemaakte programma op het Arduino bordje te plaatsen. Een alternatief is het gebruik van browser extensies zoals Codebender.cc. Deze extensie laat toe om via de browser

toegang te krijgen tot de seriële poorten van de computer waardoor het mogelijk is om te connecteren met een aangesloten Arduino board. De extensie is ook in staat om code te builden en te flashen naar het verbonden bord. Hier moeten de gebruikers ook opnieuw de gegenereerde code van Blockly4Arduino kopiëren en plakken naar de text-editor van de Codebender.cc website. Niettemin dient er geen lokale software geïnstalleerd te worden buiten het toevoegen van een extensie in de webbrowser.

Aangezien er naast de Blockly4Arduino webapplicatie nog een 3th party applicatie nodig is om de code te kunnen testen op een Arduino board, zou het beter en eenvoudiger zijn dat Blockly4Arduino als stand-alone applicatie kan functioneren. Daarom worden verschillende alternatieven en mogelijke oplossingen onderzocht om het proces te optimaliseren (Tabel 2.1). Mogelijke opties zijn:

1. Codebender, een website waar de gebruiker Arduino code kan schrijven, compileren en uploaden naar een aangesloten Arduino board;
2. Arduino IDE, een cross-platform programmeeromgeving ontworpen om Arduino code te schrijven, compileren en te uploaden naar Arduino boards;
3. Arduino Create, de online versie van de Arduino IDE;
4. PlatformIO, een cross-platform IDE met de mogelijkheid om Arduino code te schrijven, compileren en uploaden;
5. MakeFile, een bestand waarin gedefinieerd wordt welke bestanden gecompileerd moeten worden en hoe ze gelinkt moeten worden om dan te compileren via een C-compiler zoals avr-gcc;
6. Scons, een command-line tool die het mogelijk maakt om bestanden te compileren en te uploaden vanaf de command line;
7. Visuino, een grafische programmeeromgeving voor Arduino die achterliggend de Arduino IDE gebruikt om te compileren;
8. Command-line tool zoals avr-gcc, maakt het mogelijk om Arduino code te compileren.

In Tabel 2.1 staan de onderzochte compilers in relatie met gewenste voorwaarden. Na onderzoek is het duidelijk Codebender.cc aan de meeste voorwaarden voldoet, gevolgd door het populaire Arduino IDE platform. Het grote voordeel dat Codebender heeft is de

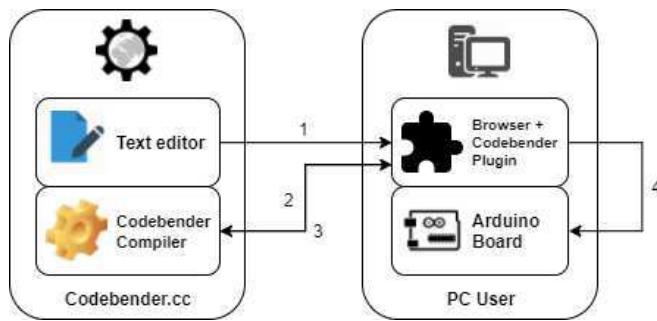
Tabel 2.1: Vergelijking tussen mogelijke compilers op basis van een aantal karakteristieken.

Compilers	Codebender	Arduino Create	Arduino IDE	PlatformIO	MakeFile	Scons	Visuino	cmd tool
Open Source	X	-	X	X	X	X	X	-
Offline mogelijk	-	-	X	X	X	X	X	X
Chromebook compatibel	X	X	X	-	-	-	-	-
Geen gebruikers-account vereist	X	-	X	X	X	X	X	X
Geen software te installeren	X	-	-	-	-	-	-	-
Gebruiksvriendelijk	X	X	X	-	-	-	X	-

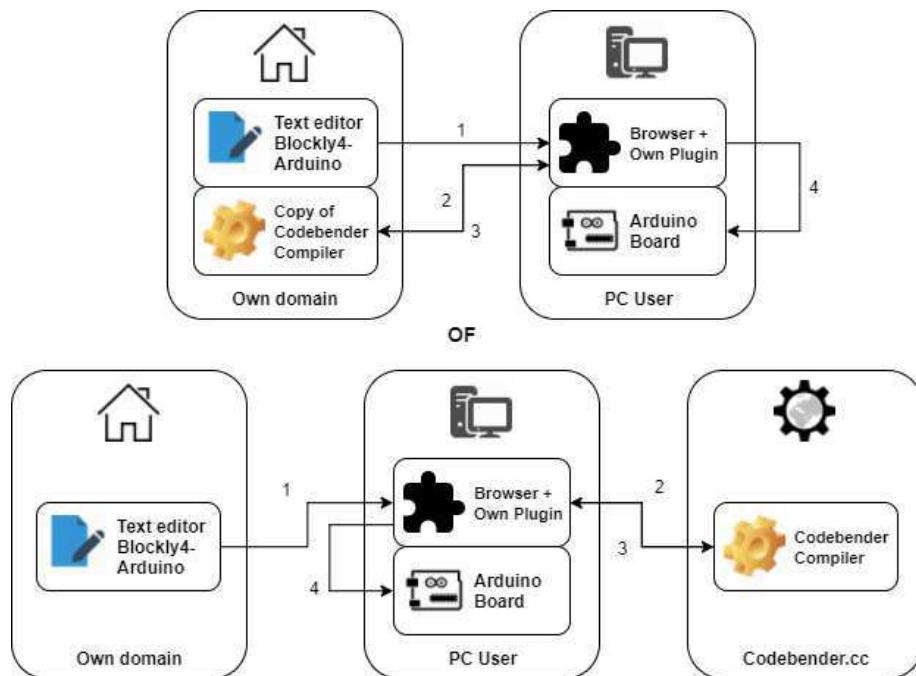
open-source code die ter beschikking staat op GitHub. Hierdoor is het mogelijk om de technologie van Codebender te integreren in een eigen project om een stand-alone versie te creëren. De koppeling van Blockly4Arduino met de open-source code van Codebender is uiterst geschikt om de webapplicatie uit te breiden zodat gebruikers via een browser extensie hun programma kunnen uploaden naar een Arduino bordje. Hierbij is er geen extra software nodig die geïnstalleerd moet worden en het opzetten van de workshops gemakkelijker loopt. Ten slotte is deze oplossing ideaal om ook workshops te kunnen geven op Chromebooks, deze computers komen steeds frequenter voor in educatieve omgevingen waardoor het installeren van software ongewenst is.

Opmerking: voor ChromeOS (Chromebooks) zijn er onder andere plugins zoals Arduino Create om code op een Arduino board te plaatsen maar deze extensie werkt niet op andere Operating Systems. Als ultiem doel is het beter om voor een universele oplossing te gaan zoals het integreren van Codebender dan voor ieder besturingssysteem een andere oplossing te moeten zoeken.

In figuur 2.3 is de huidige structuur te zien die Codebender gebruikt om code te compileren en te uploaden naar een Arduino board. In de text-editor worden de code-regels ingevoerd, daarna wordt op compileren geklikt waardoor connectie gemaakt wordt met hun browser-plugin. Deze plugin zal de code laten compilen door hun compiler (staat op hun domein). Na compilatie wordt de gecompileerde code naar het Arduino Board geflasht, dit is mogelijk omdat de browser plugin toegang heeft tot de seriële poorten van de computer.



Figuur 2.3: Codebender.cc: Compiler structuur waarbij de gebruiker code kan maken op het platform en via een extensie deze compileren en uploaden naar een aangesloten Arduino board.



Figuur 2.4: Compiler structuur aangepast voor Blockly4Arduino. Ofwel op het domein van Blockly4Arduino hosten ofwel gebruik maken van de diensten van Codebender.

Mits de originele code van de Codebender compiler open-source is, is het mogelijk om de compiler te integreren in een eigen project. Dit kan op twee manieren, zoals te zien is in figuur 2.4. Een eerste manier is om zelf de compiler op te zetten binnen het eigen domein en een eigen plugin te ontwikkelen. Op die manier is het mogelijk om dezelfde structuur te bekomen als in figuur 2.3. Een tweede mogelijkheid is om de compiler op het codebender domein te gebruiken zodat men zelf deze niet moet opzetten. Er zal echter ook wel een

eigen plugin moeten geschreven worden om de code van het eigen domein met de compiler te linken. De plugin zal dan ook de code uploaden naar het board. De precieze werking en implementatie komt in een volgend hoofdstuk aan bod.

2.3 Creatie van extra blokken voor Blockly4Arduino

Een tweede onderzoeksvraag is het toevoegen van extra blokken en architecturen om de functionaliteit van de bestaande Blockly4Arduino omgeving te verhogen. Extra functionaliteit betekent dat er een grotere verscheidenheid aan programma's kan gerealiseerd worden via het combineren van blokken.

Deze sectie is praktijkgericht en legt de nadruk op het implementeren van nieuwe blokken. Mits het platform al operationeel is, is het eenvoudig om nieuwe blokken toe te voegen. Er moet enkel nagedacht worden over de volgende zaken:

1. Functionaliteit;
2. Layout;
3. Code generatie.

Tijdens onderzoek naar varianten (Pasternak, 2017) [14] van het platform kwamen er ook verschillen naar boven omtrent het aanbod in blokken. Zo heeft het Blockly4Arduino platform een hele reeks blokken ter beschikking maar zijn er nog uitbreidingen mogelijk om meer functionaliteit aan te bieden. Bij het onderzoek hadden varianten volgende blokken voorzien die Blockly4Arduino (nog) niet heeft. Opmerking: sommige zaken zijn reeds mogelijk te programmeren met de bestaande blokken, maar functionaliteit bestaat niet als dedicated blok.

1. Display blokken;
 - (a) LCD display;
 - (b) OLED display.
2. Detectie blokken;
 - (a) bewegingssensor (bv. PIR Motion sensor);
 - (b) Afstand sensor (bv.HC-SR04 Distance sensor);

- (c) Geluid sensor.
- 3. Aansturing blokken;
 - (a) Joystick.
- 4. Overige.
 - (a) Interrupts

Indien bovenstaande lijst aan blokken geïmplementeerd zou worden in Blockly4Arduino, zijn meer toepassingen mogelijk op een intuïtieve manier. Naast deze items kunnen additioneel nog blokken toegevoegd worden om complexere toepassingen en structuren mee uit te werken, hierbij wordt aan de volgende elementen gedacht.

- 1. Lijst en Array;
- 2. For-each loop;
- 3. Klassen en overerving.

Om klassen te implementeren in Blockly4Arduino zou er een grondige studie moeten gedaan worden naar de visualisatie en hoe dit een project kan vooruithelpen. Omdat het doelpubliek beginnende programmeurs zijn, worden er zelden tot nooit complexere structuren gebruikt en lijkt het gebruik en implementatie van klassen dan ook overbodig. Via lijsten en arrays is het wel mogelijk om objecten of strings op te slaan en bij te houden. Dit is handig wanneer er oefeningen gemaakt worden waarmee een display wordt aangestuurd of via serieel text wordt verzonden. De foreach loop is een extra lus die kan gebruikt worden in oefeningen om over een lijst te itereren.

2.4 Interactie en tutorials voor Blockly4Arduino

Een laatste onderzoeksraag handelt over het interactiever maken van het platform. Het doel van dit onderzoek is om het platform te verduidelijken en hulp of hints te voorzien waar nodig zodat leerlingen van thuis uit extra oefeningen kunnen maken zonder de hulp van een coach of leerkracht. Deze sectie is puur gericht op literatuur en werd niet verder uitgewerkt in het hoofdstuk implementatie wegens out of scope.

2.4.1 Wat bestaat er al?

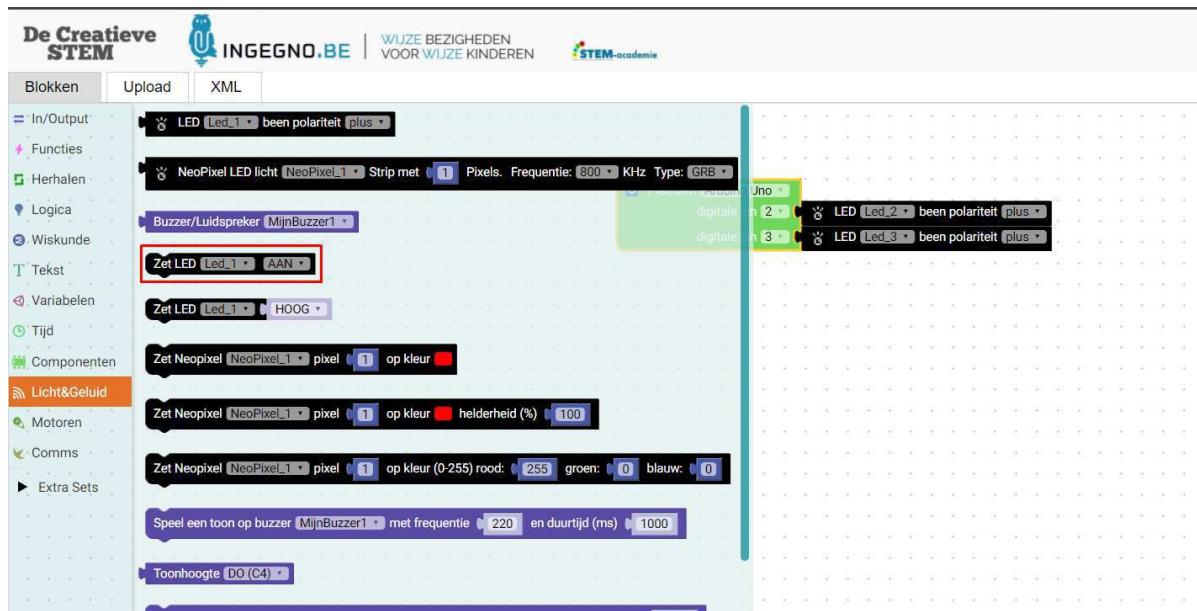
Uit onderzoeken [15], [16], [17], [18] en ondervindingen blijkt dat kinderen de tekst of tutorial niet volledig lezen en voornamelijk zich baseren op de beschikbare afbeeldingen. Indien een tutorial voorzien wordt waarin alle stappen tekstuueel staan uitgelegd, is de kans nog altijd groot dat er kinderen zijn die er niet in slagen om het probleem zelfstandig op te lossen. Het is dus voordeliger om bij hints en tutorials te werken met visualisatie in plaats van tekst. Anderzijds heeft het visueel werken nog een belangrijk voordeel dat kan teruggekoppeld worden naar 2.1, namelijk de fun factor: hoe leuker ze iets vinden, hoe gemotiveerde gebruikers zijn om de oefeningen op te lossen. Gebruikers vinden het veel leuker om aanwijzingen te krijgen via een vorm van visualisatie dan een blok tekst te moeten lezen.

Nu het nuttig blijkt om interactie en visualisatie te voorzien, wordt er gekeken naar de beste manier om dit te implementeren. Er werden verschillende varianten van Blockly4Arduino onderzocht. Hieruit bleken de volgende twee methodes telkens terug te keren.

1. Stap voor stap de nodige blokken uitleggen;
2. Enkel de blokken die nodig zijn om het programma op te lossen zijn beschikbaar i.p.v. het volledige gamma;
3. Simulatie van de gemaakte code.

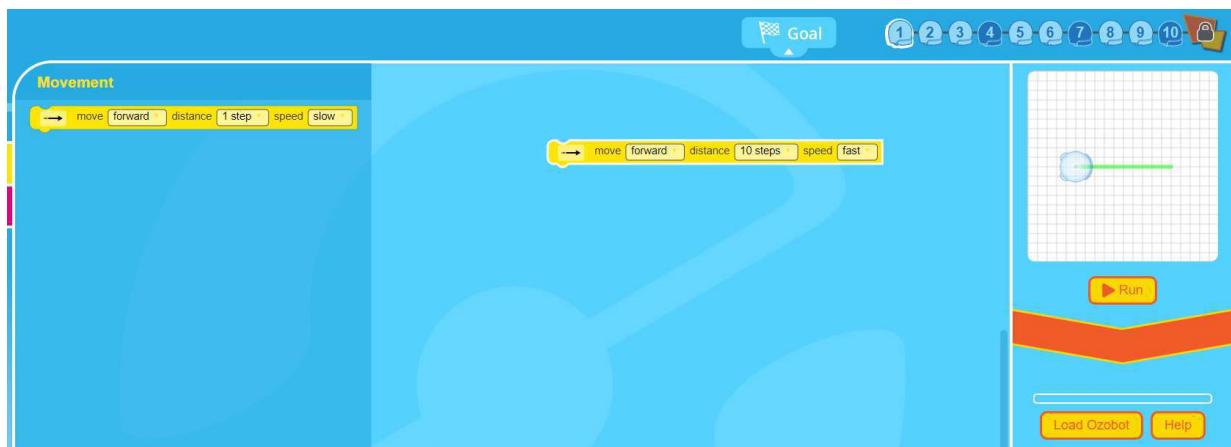
Een voorbeeld van de eerste manier is te vinden in figuur 2.5. Hierbij worden ontbrekende blokken uitgelicht om de gebruiker te helpen bij het maken van een programma. Het voordeel van deze methode is dat de gebruiker in het volledige assortiment aan blokken moet zoeken en al kennis maakt de andere blokken. De blokken blijven uitgelicht totdat het gewenste programma gemaakt is.

Een voorbeeld van de tweede manier kan geïmplementeerd worden zoals in figuur 2.6. Deze implementatie (van Ozoblockly) zorgt ervoor dat je enkel de blokken kan gebruiken die voor de oplossing zorgen. Hierbij is er telkens een venster met output te zien waarin de gebruiker het programma ziet werken. De oplossing wordt aanvaard wanneer het doel bereikt wordt. Meerdere oplossingen zijn dus mogelijk! Hoe minder blokken de gebruiker nodig heeft, hoe meer punten hij/zij kan krijgen op die oefening. Door het puntensysteem blijft de motivatie hoog en wordt er gezocht naar betere oplossingen indien men vaststelt



Figuur 2.5: Blockly4Arduino: Nodige blokken uitlichten uit het assortiment.

dat er niet hoog gescoord is. Het grote voordeel van deze methode is dat er spelenderwijs aangeleerd wordt hoe er met de blokken kan gewerkt worden.



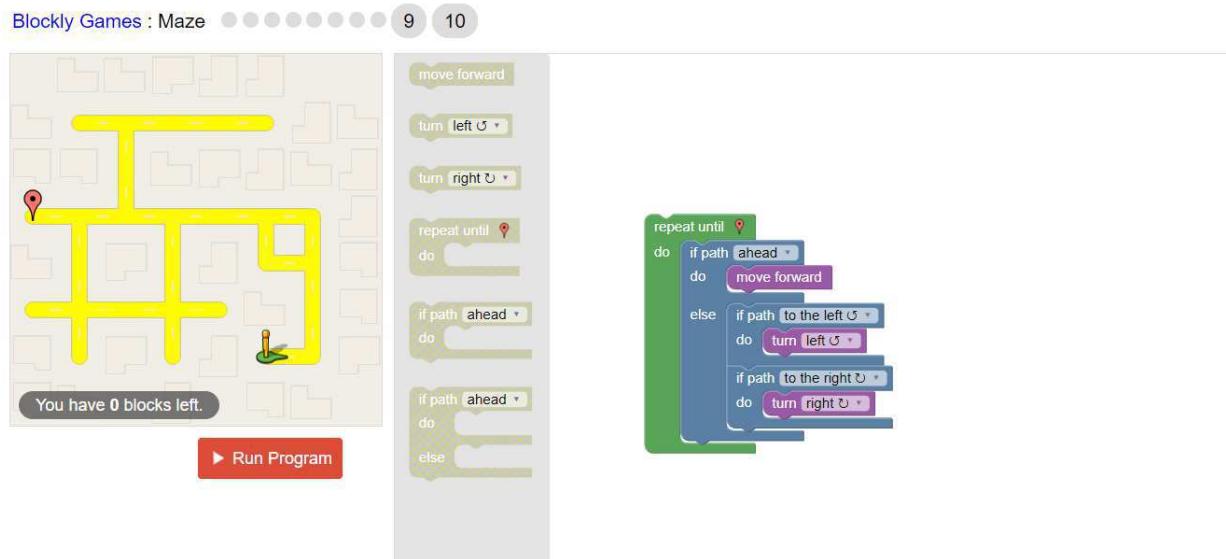
Figuur 2.6: Ozoblockly Games.

Naast Ozoblockly (figuur 2.6¹) maakt ook Google Blockly gebruik van deze methode, deze is echter wel iets uitgebreider en heeft nog een extra voordeel. In figuur 2.7² is een voorbeeld te zien van een gebruiker die door een doolhof moet manoeuvreren aan de hand van de

¹<https://ozoblockly.com/editor>

²<https://blockly-games.appspot.com/>

voorziene blokken. Hierbij wordt niet enkel kennis gemaakt met de te gebruiken blokken maar moet men ook in staat zijn om probleem-oplossend te denken.



Figuur 2.7: Blockly Game Maze, blokgebaseerde oefening.



Figuur 2.8: Blockly Game Pond, tekstgebaseerde oefening.

Als derde figuur 2.8 is een oefening afgebeeld die de gebruiker eerst moet maken aan de hand van blokken en daarna wordt dezelfde oefening gevraagd maar nu tekst-gebaseerd. Er zijn weliswaar andere waarden en parameters zodat de gebruiker nog altijd moet nadenken

maar hier wordt meteen de overgang gemaakt tussen blokgebaseerd en tekstgebaseerd programmeren. De manier waarop Google Blockly de overgang maakt tussen blokken en tekst is dus door bijna exact dezelfde oefeningen tekstueel opnieuw te maken.

Naast bovenstaande mogelijkheden is er ook de mogelijkheid om de blokken zo veel mogelijk te laten representeren naar een programmeertaal door de tekst in de blokken aan te passen zoals:

1. Dezelfde functie of variabele namen te gebruiken zoals in een echte programmeertaal (bv. if-else, for, int);
2. Kleine letters te gebruiken zoals in de echte syntax;
3. (Optioneel) Gebruik van haakjes bij de juiste structuren.

Via deze opties kan de drempel tussen blok -en tekst-gebaseerd programmeren ook verkleind worden.

2.4.2 Hoe te implementeren?

Uit vorige paragraaf werd duidelijk dat er voornamelijk 2 manieren zijn om hulp en hints te geven in een blokgebaseerde omgeving. Deze twee manieren hebben elk hun voordelen maar een combinatie van beide zou het meest gewenste resultaat opleveren. Om zowel enkel de nodige blokken te tonen, deze stap voor stap uit te lichten en een simulatie-omgeving te voorzien, kan en zal de omgeving er volledig anders uitzien per oefening. Zo kan de keuze gemaakt worden om enkel de nodige blokken weer te geven of de gebruiker toegang te geven tot het volledig assortiment. Hierbij zou er ook kunnen gekozen worden of er stap voor stap hints verschijnen of blokken worden uitgelicht of niet. Door met bovenstaand vermelde opties te gaan spelen, kunnen er heel eenvoudig ook moeilijkheidsgraden gecreëerd worden.

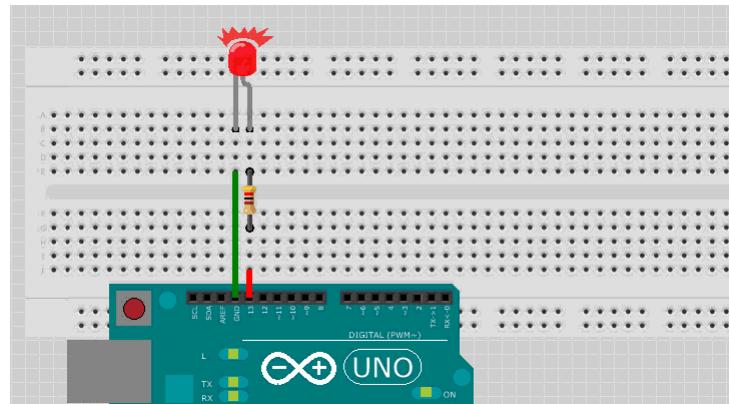
1. Eenvoudig:
 - (a) Enkel de nodige blokken beschikbaar;
 - (b) Stap voor stap worden er blokken uitgelicht of verschijnen er berichtjes met hints.
2. Middelmatig:
 - (a) Enkel de nodige blokken beschikbaar;

- (b) Geen uitgelichte blokken, gebruiker moet zelf de nodige stappen vinden.

3. Geavanceerd:

- (a) Volledig assortiment aan blokken beschikbaar;
 (b) Geen uitgelichte blokken, gebruiker moet zelf de nodige stappen vinden.

Zoals eerder vermeld maakt Google Blockly Games gebruik van oefeningen die volledig tekst-gebaseerd zijn. Hierbij heeft de gebruiker geen toegang tot de blokken en moet alles getypt worden. Dit kan een extra parameter zijn om meer moeilijkheidsgraden te maken of de moeilijkheidsgraad te verhogen.



Figuur 2.9: Voorbeeld van simulatie venster voor oefening: blink led met Arduino.

Om de werking van het programma te tonen, kan ook een output venster geïmplementeerd worden. Dit zou het eenvoudigst geschreven worden in Javascript waarbij een Arduino board getoond wordt met bijhorend verbonden sensoren, afhankelijk van de oefening. Bij de oefening om bijvoorbeeld een led te laten blinken (figuur 2.9), wordt een Arduino board getoond met een verbonden led en indien de code juist is, zal de led oplichten. Hierdoor hoeft de gebruiker niet telkens een Arduino board bij te hebben en de code te uploaden om te testen. Hier is er echter wel een nadeel dat het moeilijk zal zijn om dit zo flexibel mogelijk te maken. Hoogstwaarschijnlijk zal de output maar kunnen gebruikt worden bij voorgedefinieerde oefeningen. Indien de gebruiker een blok of sensor gebruikt die niet voorzien is om te animeren in de output, zal de output al niet kunnen werken. Om een output window te maken dat voor elk programma kan werken, is veel extra tijd en middelen nodig. Voor tutorials en voorgedefinieerde oefeningen echter, is dit zeker een mogelijkheid om te integreren.

2.4.3 Toegepast op Blockly4Arduino

We kunnen ook de vraag stellen of het toevoegen van een simulatie wel wenselijk is. Indien de oefeningen opgelost en geverifieerd worden met behulp van gesimuleerde output, verdwijnt er ook een heel belangrijke eigenschap van het platform: physical computing. Het leren bouwen van de circuits is een deel van het leerproces die andere vaardigheden traint dan enkel het algoritmisch denken. Het is net deze eigenschap die het platform onderscheidt van andere blokgebaseerde platformen. Algemeen zou de toevoeging nuttig zijn wanneer gebruikers van thuis uit oefeningen willen maken zonder dat ze over de hardware beschikken. De tijd en resources nodig om dit te implementeren zijn een grote overwegende factor alsook het feit dat de nadruk in de workshops op physical computing ligt en geen gebruik zal gemaakt worden van deze toepassing. Integratie is dus optioneel indien gewenst en zou geen prioriteit mogen krijgen over andere zaken zoals de optimalisatie van de workflow of breder gamma aan blokken.

2.5 Blockly4Arduino varianten

Google Blockly vormt de basis van vele andere platformen die ook een blokgebaseerde omgeving aanbieden. Via sommige platformen is het mogelijk om 2D en 3D spelletjes mee te ontwikkelen, met andere kan aan beeldverwerking worden gedaan, er zijn er ook die specifiek code omzetten om hardware aan te sturen, etc.. In deze sectie worden een aantal varianten onderzocht op basis van gebruiksvriendelijkheid, bruikbaarheid, layout en moeilijkheid. Met gebruiksvriendelijk wordt bedoeld hoe eenvoudig het is om de blokken te gebruiken en met de omgeving in het algemeen te werken. Hierbij is de layout ook belangrijk of de verschillende types blokken duidelijk afgelijnd zijn van elkaar en hoe duidelijk de omschrijving en tekst van een blok is. Bruikbaarheid staat in verband met de omgeving, is er de mogelijkheid om code te exporteren of importeren en op welke manier gebeurt dit. Uiteindelijk wordt ook nog de moeilijkheid in kaart gebracht om blokken bedoeld voor beginners tegenover blokken voor gevorderden uit te zetten en zo niveauverschil proberen te creëren. Het uiteindelijke doel van deze sectie is om de blokken zoveel mogelijk te kunnen optimaliseren naar de gebruikers toe.

In Tabel 2.2 zijn verschillende varianten opgeliist en geklasseerd volgens gebruiksvriendelijkheid, bruikbaarheid, layout en moeilijkheidsgraad. De categorieën zijn beoordeeld van - tot XXX waarbij - staat voor niet goed, X voor goed, XX zeer goed en XXX is uitstekend.

Tabel 2.2: Blockly Varianten

Variant	Gebruiksvriendelijk	Layout	Bruikbaarheid	Moeilijkheid
Scratch	XX	XX	X	X
Alice	-	-	XX	XX
Blockly	XX	XX	XXX	X
Blocklyduino	XX	X	XXX	X
Ozoblockly	XXX	XX	X	X / XX / XXX
Micro:bit	XX	XX	X	X
Open Roberta Lab	XX	XX	X	X
Robo builder	XX	XX	X	X
VubbiScript	XX	XX	XX	X

De onderzochte varianten zijn:

1. Scratch, een blokgebaseerde programmeeromgeving geschikt voor het maken van visualisaties zoals interactieve verhalen, animaties, spellen, muziek en kunst;
2. Alice, een blokgebaseerde programmeeromgeving geschikt voor het maken van simpele spellen in 3D;
3. Blockly, een blokgebaseerde programmeeromgeving die achterliggend code genereert voor een specifieke taal (e.g. JavaScript, Arduino code, ...);
4. Ozoblockly, een blokgebaseerde programmeeromgeving specifiek ontworpen voor het aansturen van de Ozobot;
5. Open Roberta Lab, een blokgebaseerde programmeeromgeving die het mogelijk maakt om motoren en sensoren te configureren en aan te sturen van een robot;
6. VubbiScript, een blokgebaseerde programmeeromgeving gebaseerd op de Unity3D engine om spellen mee te maken.

Nu de varianten gerangschikt zijn volgens bepaalde waarden is het heel eenvoudig om per categorie te gaan kijken naar wat deze varianten goed doen en wat we er van kunnen leren. Ozoblockly is heel gebruiksvriendelijk omdat er verschillende niveaus voorzien zijn met bijhorende oefeningen en tutorials, alles is duidelijk en de ozobot zelf is eenvoudig te programmeren. Daarnaast is ook een simulatievenster voorzien waar de gebruiker de werking van het programma kan bestuderen zonder de hardware te hoeven aansturen.

De layout van de meeste platformen is sterk gelijkend op elkaar waardoor het moeilijk is om op vlak van design betere of slechtere blokken te kunnen aanduiden. Blocklyduino heb ik echter iets lager ingeschat omdat deze veel afbeeldingen gebruikt in blokken die te klein en onduidelijk overkomen. Duidelijke figuren of iconen zijn een grote hulp bij het gebruik van sensorblokken. Bruikbaarheid is gequoteerd op vlak van exporteerbaarheid en aansturing. Een platform dat gemaakt is voor maar 1 stuk hardware aan te sturen is niet zo nuttig of bruikbaar. Het mag dan wel een uitstekend platform zijn, het is niet meteen inzetbaar in andere gebieden. Vandaar dat Blockly en Blocklyduino hoog scoren omdat er naar verschillende talen en hardware kan geschreven worden. Als laatste parameter is de moeilijkheidsgraad onderzocht, dit werd getoetst door te kijken voor welk doelpubliek het platform ontwikkeld is en welk doelpubliek het platform gebruikt. Ook de mogelijkheid om op verschillende moeilijkheidsgraden te werken is een groot pluspunt. Alice is een moeilijker platform omdat er meer blokken nodig zijn om hetzelfde te verkrijgen wat het moeilijker maakt om een programma te maken. Ozoblockly daarentegen biedt de mogelijkheid om op verschillende niveaus te werken en is ontwikkeld voor meerdere doelgroepen te laten kennismaken met hun platform.

Uit Tabel 2.2 kunnen er dus enkele conclusies getrokken worden om Blockly4Arduino te optimaliseren.

1. Moeilijkheidsgraden voorzien;
2. Blokken onderverdelen per moeilijkheidsgraad of fijnere categoriën;
3. Simulatievenster integreren (voor bepaalde oefeningen);
4. Duidelijke figuren en iconen ter verduidelijking van blokken.

Hoofdstuk 3

Implementatie

In dit hoofdstuk worden de onderzochte zaken uit vorig hoofdstuk uitgewerkt met de daarbij horende stappen om tot het resultaat te komen. De zaken die worden uitgewerkt zijn:

1. Uitbreiden van het assortiment aan blokken;
2. Optimalisatie van de workflow voor het Blockly4Arduino-platform.

3.1 Uitbreiden assortiment aan blokken

Deze sectie handelt over de toegevoegde blokken aan het Blockly4Arduino-platform. In het vorige hoofdstuk werden varianten van het platform en frequente programmeerstructuren onderzocht om uiteindelijk tot een lijst te komen met nuttige blokken die kunnen toegevoegd worden. Deze blokken zullen een specifieke werking hebben en het bouwen van een programma eenvoudiger maken. Sommige zaken waren reeds mogelijk om uit te werken met de bestaande blokken maar vereisten een complexe structuur, de nieuwe blokken zorgen voor een vereenvoudiging.

3.1.1 Drukknop met interne pull-up weerstand

Arduino boards bevatten reeds interne weerstanden aan de digitale pinnen om de interne circuits te beschermen tegen kortsluiting. Deze hardware kan geprogrammeerd worden zodat de interne weerstanden gebruikt worden als pull-up weerstanden. Dit betekent dat er

geen externe weerstand moet gebruikt worden. Deze techniek wordt toegepast op het gebruik van een drukknop. Mits de code voor het programmeren anders is: interne weerstand moet als pull-up ingesteld worden, kan het reeds voorziene blok in het Blockly4Arduino-platform niet gebruikt worden. Vandaar de kleine aanpassing om voor deze optie een initialisatie-blok te voorzien (figuur 3.1).

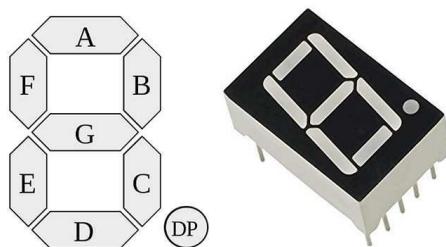


Figuur 3.1: Toevoeging van paars initialisatie-blok om externe drukknop met interne pull-up weerstand te gebruiken.

Met bovenstaand blok wordt de interne pull-up geactiveerd, hiervoor moet een andere parameter meegegeven worden aan de gebruikte functie: `pinMode("pinNummer", INPUT)` wordt `pinMode("pinNummer", INPUT_PULLUP)`.

3.1.2 7-segment display

Een 7-segment display (figuur 3.2) is een component dat bestaat uit 7 lichtgevende segmenten die 1 of meerdere LED-lichtjes bevatten. Sommige displays bevatten een extra segment als punt-representatie. Het gebruik van deze component wordt voornamelijk gebruikt bij het afbeelden van getallen of bepaalde letters.

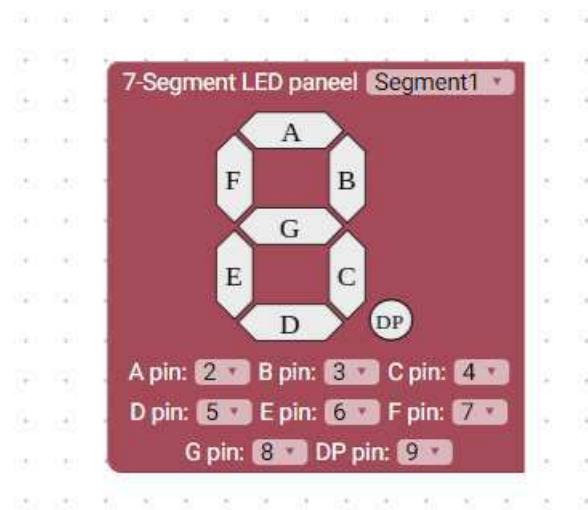


Figuur 3.2: 7-Segment component met aanduiding van de verschillende segmenten en punt.

Bij zo'n 7-segment display zijn doorgaans de anodes onderling verbonden, hier spreekt men van common anode. Om alle segmenten aan te sturen, hebben we genoeg met 8 digitale input pinnen en 1 output pin (Ground).

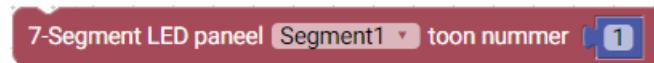
Mits de Blockly4Arduino-omgeving al een blok bevat om een digitale pin aan te sturen, zou het display aangestuurd kunnen worden met de bestaande blokken. Om hierop verder te werken of een oefening te maken met een teller, zou de code veel te lang en complex worden waar optimalisatie mogelijk is.

Zoals eerder vermeld, zijn er dus maximaal 8 digitale pinnen nodig om een display aan te sturen. Hierbij is een centrale blok ontworpen om in te stellen welk segment met welke digitale pin verbonden wordt (figuur 3.3).



Figuur 3.3: 7-segment centrale blok om alle segmenten aan een digitale pin te koppelen.

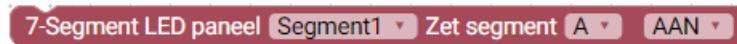
Nu de gewenste pinnen geconfigureerd zijn, moet het display nog aangestuurd worden. Hiervoor is een tweede blok ontworpen dat toelaat om een cijfer op het display te tonen (figuur 3.4). Het blok vraagt de naam van het 7-segment display die we eerder gedefinieerd hebben in het centrale blok. Daarna wordt er meegegeven welk cijfer er getoond moet worden.



Figuur 3.4: 7-segment blok dat achterliggend de code genereert om het meegegeven cijfer te tonen.

Een derde en laatste blok omtrent de 7-segment component bestaat uit een blok dat een enkel segment kan aansturen (3.5). De beperking bij vorig blok is dat er voorlopig enkel

nummers kunnen getoond worden. Met dit laatste blok is het dus mogelijk om andere sequenties te maken.



Figuur 3.5: 7-segment blok dat achterliggend een enkel segment zal aansturen.

3.1.3 OLED display

OLED of Organic Light Emitting Diode, is een halfgeleider lichtbron waarbij de organische materialen licht uitstralen wanneer er stroom doorgaat. Dit zorgt ervoor dat ze geen achtergrondverlichting nodig hebben zoals bij LCD-schermen. OLED displays hebben net daarom 2 grote voordelen: ze zijn dunner en energiezuiniger.

Deze displays worden aangestuurd via de I²C-bus. Mits het Blockly4Arduino-platform nog geen bestaande blokken bevat om over I²C te communiceren, zijn er 2 mogelijkheden:

1. Nieuwe blokken introduceren om I²C communicatie op te zetten;
2. Achterliggend gebruik maken van een bibliotheek om sneller en eenvoudiger OLED displays aan te sturen.

Voordelen van een Arduino-bibliotheek voor I²C communicatie:

1. Eenvoudiger op te bouwen code dankzij de voorziene functies van de library;
2. Bruikbaar voor meerdere types OLED displays zonder dat de Arduino-code zal wijzigen.

Nadelen van een bibliotheek:

1. Afhankelijk van een externe bibliotheek waarbij de code of functionaliteit vast ligt;
2. Extra bibliotheek die gecompileerd moet worden door de compiler.

Na het bestuderen van enkele Arduino-bibliotheken omtrent I²C, is beslist dat de weg via een bibliotheek beter geschikt was om te gebruiken. De hoofdreden hiervoor zijnde dat de bibliotheek al functies bevatte om verschillende types en groottes van OLED displays aan te sturen waardoor er standaard blokken kunnen ontworpen worden zonder dat de code achterliggend veel zal veranderen. De gekozen bibliotheek voor I²C-communicatie met de OLED displays is: U8G2-library. Deze bibliotheek laat toe om verschillende types en groottes van OLED displays aan te sturen.

Design van de OLED blokken

Om het OLED display aan te sturen worden 4 blokken gebruikt. Achterliggend wordt een include gedaan van de U8G2-library om eenvoudiger het display aan te sturen over I²C . Het doel is om met onderstaande blokken tekst te kunnen tonen op het display.

De volgende blokken zijn gebouwd en getest via een monochromatisch OLED display met een resolutie van 128x32 pixels. Het gaat over een monochromatisch display wat wil zeggen dat er maar 1 kleurtoon gebruikt wordt. Een blok voorzien om tekst -of achtergrondkleur te kiezen is bijgevolg nog niet geïmplementeerd wegens het niet kunnen testen van de werking.

Een eerste blok (figuur 3.6) initialiseert het display voor een bepaalde resolutie. De library kan een heel arsenal aan displays aansturen maar vereist is om aan te geven over welke resolutie het gaat. Met dit blok kunnen er later op eenvoudige wijze extra resoluties worden toegevoegd.

OLED Initialiseer OLED1 met resolutie 128x32

Figuur 3.6: OLED blok om de resolutie van het aan te sturen display in te stellen in de code.

Het tweede blok (figuur 3.7) laat toe om een pixelgrootte te selecteren voor de tekst zodat er wat kan gespeeld worden met de inhoud die op het scherm moet komen. Indien er 1 enkele variabele op het scherm getoond moet worden, kan dit met een groter font. Indien men meerdere variabelen onder elkaar wil weergeven, zal dit moeilijker zijn op een 32px hoog scherm dus moet de grootte van de tekens verkleind kunnen worden.



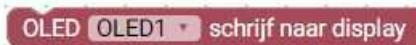
Figuur 3.7: OLED blok om de font-grootte in te stellen van de tekst die op het display zal verschijnen.

Het print block (figuur 3.8) zorgt ervoor dat je kan instellen waar de cursor moet staan om de tekst te schrijven. De cursor bepaalt de startpositie vanaf waar tekst zal geschreven worden. Vervolgens kan je ook al de gewenste tekst/variabele meegeven om af te beelden. Dit blok kan meerdere keren gebruikt worden om ervoor te zorgen dat er meerdere waarden onder of naast elkaar kunnen geschreven worden.



Figuur 3.8: OLED blok om vanaf een bepaalde positie de meegegeven waarde op het display te tonen.

Het laatste block zal al de ingestelde waarden van de vorige blokken naar het scherm schrijven.



Figuur 3.9: OLED blok om de ingestelde waarden te schrijven naar het display.

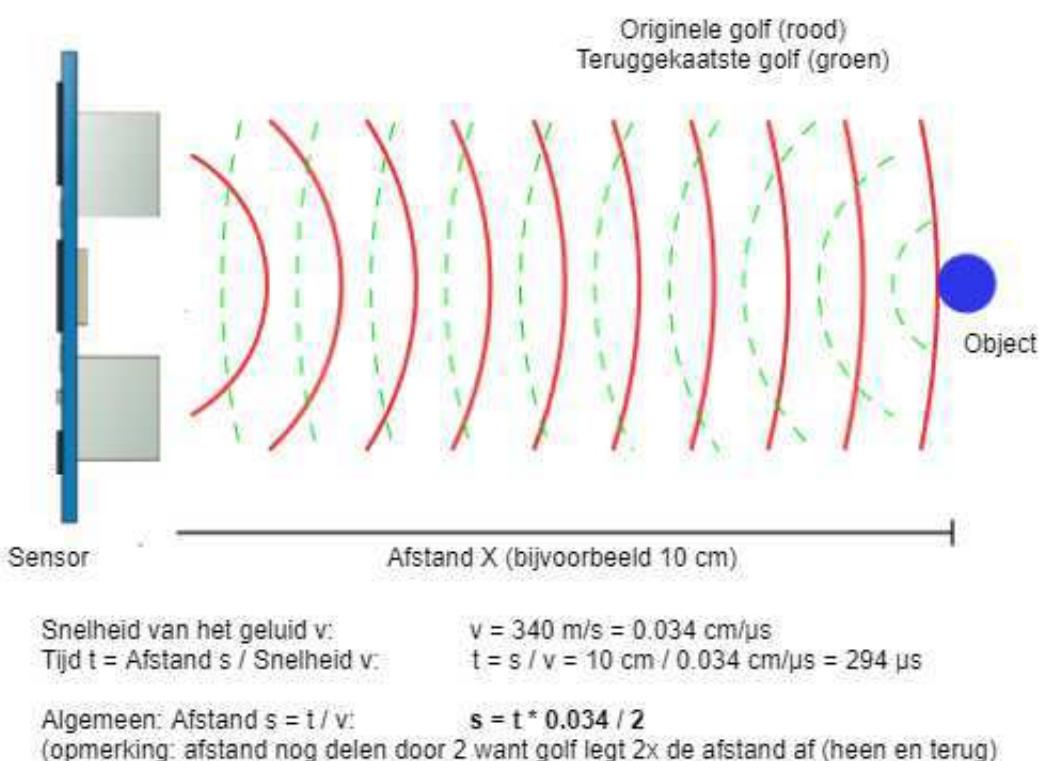
Met bovenstaande blokken is het mogelijk om waarden te tonen op een OLED display en hierbij in te stellen waar precies de tekst moet komen. Ook het instellen van de tekstgrootte is te regelen.

3.1.4 Supersonische afstandssensor HC-SR04

Deze subsectie handelt over het implementeren van blokken om de afstand op te meten aan de hand van een supersonische sensor (component HC-SR04).

Werking supersonische sensor

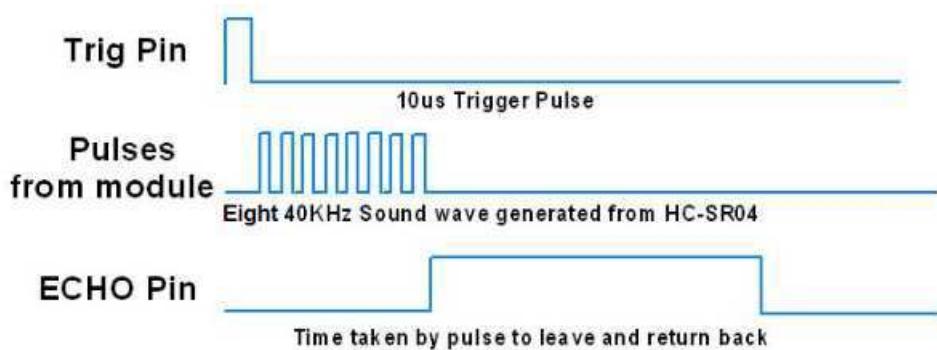
De HC-SR04-component is een supersonische sensor die ultrasone golven uitzendt aan 40 kHz die zich voortplanten door de lucht. Als er een object zich in het pad van de golf bevindt, zal de golf terugkaatsen naar de module. Aangezien de snelheid van geluid gekend is, kan de afstand tot het object berekend worden. Opmerking: de afstand moet gedeeld worden door 2 want de geluidsgolf legt 2 keer die afstand af, vertrekken en terugkaatsen. Onderstaande figuur verduidelijkt hoe de afstand wordt berekend (figuur 3.10).



Figuur 3.10: Formule om de afstand te berekenen aan de hand van de tijd en de snelheid van geluidsgolven.

Om de HC-SR04-module aan te sturen zijn er 4 pinnen voorzien: VCC, GND, Echo en Trigger. De Trigger -en Echo-pinnen zullen instaan voor het verzenden en ontvangen van de supersonische golven. Eerst wordt de Trigger-pin voor een korte periode ($10 \mu\text{s}$) hoog gebracht waardoor 8 cyclische sonische golven worden uitgezonden (zie figuur 3.11)¹. De Echo-pin zal de verzonden golven terug opvangen en geeft de tijd terug in microseconden.

¹<https://sites.google.com/site/summerfuelrobots/arduino-sensor-tutorials/eren-module>



Figuur 3.11: Timing diagram van de Trigger -en Echopinnen

Design blokken voor de supersonische afstandssensor

Om de sonische sensor aan te sturen, zijn er 2 nieuwe blokken voorzien: een configuratie-blok en een blok om de huidige afstand te verkrijgen.

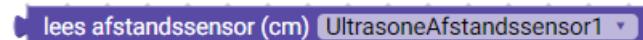
In het configuratie-blok (figuur 3.12), kan je de Echo -en Triggerpinnen, de naam van de sensor en de maximale wachttijd definiëren. Indien de maximale wachttijd niet wordt meegegeven en geen object wordt gedetecteerd, zal de sensor 0s teruggeven. Dit heeft een grote invloed op de werking want de afstand zou plotseling 0cm worden indien niets wordt gedetecteerd. Standaard staat deze parameter ingesteld op 1 seconde, dit betekent dat de maximaal detecteerbare afstand $s_{max} = 1000\mu s * 0.034\text{cm}/\mu s / 2 = 17\text{cm}$. De maximale wachttijd is dus een parameter die toelaat om de maximaal detecteerbare afstand in te stellen.



Figuur 3.12: Sensor blok om een supersonische sensor te configureren, stelt de Echo -en Trigger-pinnen in alsook de maximale wachttijd.

Het read-blok (figuur 3.13), bestaat enkel uit een dropdown-list waar je de naam van de afstandssensor kan selecteren. Dit blok zendt en ontvangt de supersonische signalen en

berekent de afstand. Vervolgens wordt de berekende afstand geretourneerd.



Figuur 3.13: Sensor blok om de afstand te retourneren van de aangegeven sonische sensor.

3.1.5 Kopieer-knop

Bij het gebruik van het Blockly4Arduino-platform, wordt de code geschreven aan de hand van blokken. Achterliggend wordt de Arduino-code gegenereerd, deze code moet eerst gecompileerd worden tot een binair -of hexadecimaal bestandsformaat want het is met dit bestandsformaat dat de hardware wordt geprogrammeerd. Om de code die het platform genereert om te zetten naar een hexadecimaal bestandsformaat, wordt de code gekopieerd van het platform en geplakt in Codebender of de Arduino IDE om te compileren. Tijdens de Blockly4Arduino-workshops werd vastgesteld dat de compiler soms fouten teruggaf die te wijten waren aan het fout kopiëren en niet aan de code. Wanneer code werd gekopieerd, kwam het voor dat niet de volledige code werd geselecteerd vooraleer te kopiëren.



Figuur 3.14: Werkbalk van het Blockly4Arduino-platform met toegevoegde kopieer-knop.

Om bovenstaande ergernis te vermijden en een vlottere overgang te creëren van het platform naar de compiler, werd een kopieer-knop toegevoegd (figuur 3.14). De kopieer-knop doet volgende stappen voor de gebruiker: selecteert de volledige code en kopieert deze daarna naar het klembord. Hiermee worden fouten vermeden en wordt de gebruiker gespaard van het selecteren van de code en kopiëren via CTRL+C.

Uit navraag blijkt dat de toevoeging van de kopieer-knop al veel wordt gebruikt en een handige toevoeging is voor het platform. Zie het Evaluatie hoofdstuk (hoofdstuk 4).

3.2 Optimalisatie workflow

In vorig hoofdstuk zijn er verschillende varianten en mogelijkheden onderzocht om de workflow te verbeteren voor de gebruikers tijdens workshops. Uit het gevoerde onderzoek bleek dat de architectuur van Codebender optimaal bleek te zijn om te integreren.

Gebruikers van het Blockly4Arduino-platform kunnen het platform voorlopig enkel gebruiken om code te maken door middel van blokken. De nadruk in de workshops ligt echter op physical computing waarbij ook hardware moet aangestuurd worden. Hiervoor is een extra applicatie of software nodig en moet er regelmatig geswitcht worden tussen de applicaties. Anderzijds is het opzetten van extra software tijdrovend in de korte workshops waardoor minder gefocust kan worden op het aanleren van technieken.

3.2.1 Introductie naar integratie compiler

De workflow verbeteren van het Blockly4Arduino-platform kan door het integreren van een code-compiler en code-flasher. Dit betekent dat er geen externe software of applicaties nodig zijn om de hardware aan te sturen. Hierdoor kan de focus verder op physical computing gelegd worden.

Om volgende secties beter te begrijpen is het belangrijk dat eerst het onderscheid gemaakt wordt tussen Arduino -en C-code (Badamasi, 2014) [19], [20]. Dit verschil helpt mee om de redeneringen te volgen die gemaakt worden of hoe componenten werken. een Arduinoschets verschilt van een standaard C-programma doordat het een main mist (geleverd door de Arduino-core), functie-prototypes niet verplicht zijn en bibliotheken opnemen automatisch is (enkel een `#include` nodig). Het compileren van C en Arduino code resulteert in binaire data (machine taal). Deze data kan ook worden voorgesteld door middel van ASCII-tekens en wordt opgeslagen in een Intel HEX-bestand.

3.2.2 Implementatie via Codebender

In de literatuurstudie werd onderzocht op welke manieren de code kon gecompileerd worden. Hierbij werden verschillende methodes en bestaande compilers onderzocht (zie tabel 2.1). Uit het eerste onderzoek bleek dat de meest geschikte weg, de weg was via Codebender. Codebender.cc is een online platform waar gebruikers code kunnen schrijven en

vanuit het platform zelf al hun hardware programmeren. Het Codebender-platform werkt op onderstaande manier.



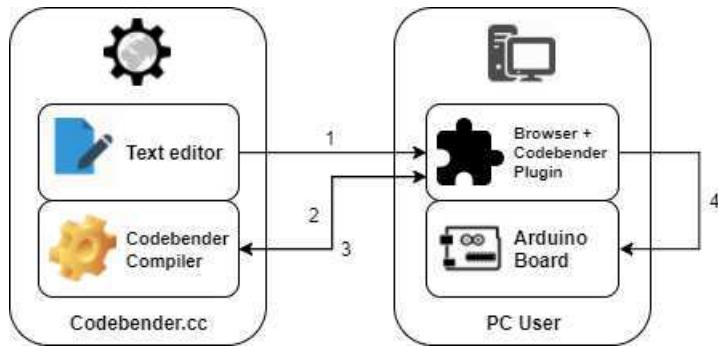
Figuur 3.15: Codebender-platform: User-interface om programma's te schrijven, te verifiëren en te uploaden.

In figuur 3.15 is de grafische interface te zien van Codebender. Hier zijn 5 zaken van belang van hoe Codebender de code zal compileren en uploaden naar de hardware.

1. Tekst-editor waar de code wordt geschreven;
2. Selectie uit een aantal hardware boards om te programmeren;
3. Selectie uit een lijst van poorten waarop hardware aangesloten is;
4. Verifieer-knop: compiler controleert of de code geen fouten bevat;
5. Upload-knop: programmeert de aangesloten hardware op de gekozen poort.

Achterliggend wordt de code gecompileerd en geüpload volgens onderstaande architectuur (figuur 3.16). Het Codebender-platform werkt voor het compileren en uploaden van de code via een browser extensie beschikbaar voor Google Chrome en Firefox. Deze extensie staat in voor het ophalen van de geschreven code en deze te verzenden via een HTTP POST-request naar de compiler die zich binnen het domein van Codebender bevindt. De compiler zal vervolgens de code compileren en omzetten naar een binair of HEX formaat en dit terugsturen in het response-bericht. Eenmaal de extensie de response heeft ontvangen, zal de extensie de hardware programmeren.

Er waren 2 hoofdredenen waarom uit het onderzoek bleek dat Codebender het meest geschikt was:



Figuur 3.16: Codebender.cc architectuur

1. Om de functionaliteit te voorzien voor het Blockly4Arduino-platform wensen we een gelijkaardige structuur te bekomen;
2. De Codebender source code is beschikbaar op Github onder een open source licentie.

De source code van Codebender bestaat uit verschillende projecten opgebouwd in Symfony (PHP-omgeving), deze projecten omvatten onder andere:

1. Compilerflasher;
2. Compiler;
3. Builder.

Codebender: compilerflasher

Het eerste onderdeel van de Codebender repository op Github bestaat uit een JavaScript-bestand genaamd compilerflasher. Dit bestand bevat JavaScript-code die de verbinding maakt tussen de eerste versie van de Codebender-website en hun extensie. Compilerflasher stond in voor de volgende zaken:

1. Nagaan of de webbrowser-extensie geïnstalleerd is;
2. Een lijst met alle hardware boards die te compileren zijn afhalen van de server en deze weergeven;
3. Via de extensie de beschikbare poorten opvragen en oplijsten;
4. Een POST-request uitvoeren naar hun domein met de gegeven code om deze te laten compileren;

5. Via de extensie de gecompileerde code laten flashen naar het geselecteerde board om dit te programmeren.

Indien dit script geïntegreerd wordt in Blockly4Arduino, zou het theoretisch al mogelijk zijn om hardware te programmeren vanuit de browser via het Blockly4Arduino-platform. Het compileren gebeurt dan aan de kant van Codebender en via hun beschikbare extensie zou de code geflasht kunnen worden. Tabel 3.1 geeft de voor -en nadelen terug van het implementeren van het compilerflasher-script. De voordelen sommen vooral de functionaliteiten op die nodig zijn om de architectuur te voorzien binnen het Blockly4Arduino-platform. Anderzijds komen er heel wat nadelen naar boven waarom deze weg via compilerflasher niet geschikt is, namelijk: afhankelijkheid. Voor zowel het compileren als flashen van de code ben je volledig afhankelijk van de diensten van Codebender. Indien er server-sided iets verandert bij Codebender, is de kans groot dat deze weg niet meer werkt. Ook als hun diensten falen kan er zelf geen back-up ingeschakeld worden om de functionaliteit te laten doorgaan. Daarenboven kunnen er zelf geen updates of nieuwe types hardware worden toegevoegd.

Tabel 3.1: Schematisch overzicht van de mogelijkheden met het compilerflasher-script met bijhorende voor -en nadelen.

Opties	Voordelen	Nadelen
Koppeling Codebender extensie	Code beschikbaar om de extensie aan te sturen; Weinig tot geen aanpassingen nodig.	
Compileren op hun server	Zelf geen compiler op te zetten; Enkel gebruik maken van hun diensten; Zekerheid dat compilatie correct verloopt.	Geen controle over compiler; Geen controle over nieuwe update;s Volledig afhankelijk van Codebender.
Extensie	Mogelijkheid om snel en eenvoudig code te compileren en te flashen;	Geen eigen controle over de extensie; Volledig afhankelijk van de mogelijkheden die de extensie biedt; Enkel mogelijk om de gedefinieerde hardware te programmeren, nieuwe types toevoegen niet mogelijk.

Uitwerking met compilerflasher

Om de functionaliteiten en de werking van het Codebender-platform beter te leren kennen, is er een testpagina opgezet om de compiler en extensie aan te sturen. Hierbij bleek dat het compilerflasher-script al verouderd was ten opzichte van de huidige website van Codebender en hun extensie. Het script was wel nog in staat om POST-requests uit te voeren waardoor de code kon geverifieerd worden. Mits de extensie al vernieuwd was en dit script verouderd, kon er geen verbinding gemaakt worden met de extensie. Hierdoor was het niet mogelijk om aangesloten hardware te vinden of de hardware te flashen.

Conclusie compilerflasher

Het compilerflasher-script was een manier waarop Codebender de verbinding maakte tussen hun webpagina en hun extensie. Mits dit script open-source beschikbaar is, is het vanzelfsprekend dat andere developers ook hun server beginnen aan te spreken. Codebender heeft ondertussen een update van hun website en extensie doorgevoerd waardoor het aanspreken van hun diensten niet meer van toepassing is.

Ondanks de mogelijkheden die het script moest bieden, zijn er te grote nadelen aan verbonden. Denk hierbij aan de afhankelijkheid en er zijn weinig tot geen aanpassingen mogelijk. Indien de diensten van Codebender voor een bepaalde tijd niet beschikbaar zijn, zou het Blockly4Arduino-platform bijgevolg ook geen code meer kunnen compileren en flashen.

Codebender: Compiler

Een tweede project dat beschikbaar was op Codebenders Github repository is het project genaamd Compiler. Codebender Compiler is een Symfony-project, Symfony is een PHP-framework voor web development. Dit project bouwt een webserver op waar requests worden ontvangen en achterliggend wordt in Python de code gecompileerd. Met dit project is het mogelijk om de compiler van Codebender zelf te hosten. Zelf deze compiler hosten heeft zeker en vast zijn voordelen, zie tabel 3.2.

Uit vorige paragraaf bleek dat het voordelig was om zelf een compiler op te kunnen zetten zodat het beheer in eigen handen was. Hierdoor kunnen er eenvoudiger aanpassingen doorgevoerd worden of kan je de compiler op een andere manier integreren in het project. Zoals in het onderzoek uiteen werd gezet, zou de uiteindelijke structuur er moeten uitzien zoals in figuur 3.17. Deze figuur omschrijft 2 oplossingen: zelf de compiler opzetten wat het meest optimale zou zijn of de compiler gebruiken aan de kant van Codebender. De

Tabel 3.2: Voor -en nadelen bij het gebruik van de Compiler van Codebender.

<i>Opties</i>	<i>Voordelen</i>	<i>Nadelen</i>
Compiler zelf hosten	Niet afhankelijk van Codebender; Mogelijkheid om zaken aan te passen, te verbeteren; Aanpassen naar gelang van gewenste architectuur.	Updates zelf moeten doorvoeren; Nieuwe hardware of andere versies zullen niet gecompileerd kunnen worden; Server hosten brengt een bepaalde kost met zich mee.

compiler gebruiken aan de kant van Codebender zou gebruikt kunnen worden als back-up maar idealiter wordt de Compiler zelf opgezet om onafhankelijk te zijn.

Uitwerking met Codebender Compiler

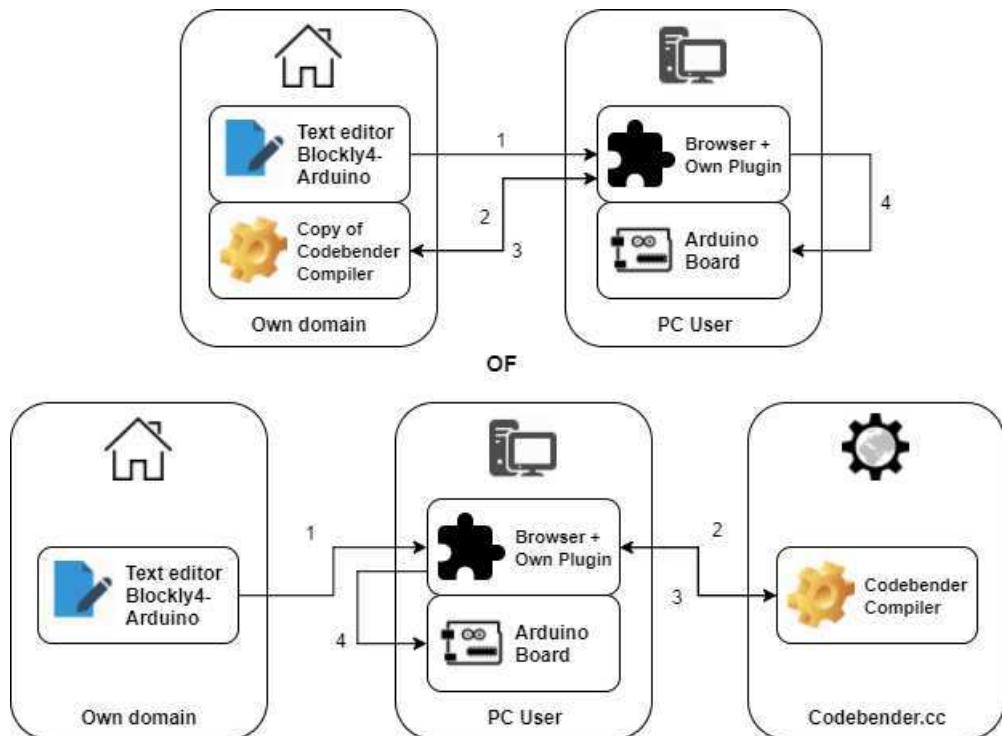
Om de Compiler te integreren, wordt het bestaande Symfony-project opgezet om de server te hosten. Eenmaal de server is opgezet en de nodige packages geïnstalleerd zijn, bekomen we de architectuur die te zien is in figuur 3.18.

De server is zodanig opgebouwd dat deze HTTP-requests verwacht met JSON-content volgens een bepaalde structuur (zie figuur 3.19).

Figuur 3.19 geeft een HTTP POST-request weer waarin data wordt meegegeven in JSON-formaat. De meegegeven data bestaat uit de programmacode, bibliotheken, compileerversie en het type hardware waarnaar gecompileerd wordt. Om de requests te sturen, werd een simpele server opgezet met een webpagina die een POST-request stuurde naar het IP-adres van de opgezette compiler. Enkel bij het sturen van een request met de data in het juiste formaat, werd correct gecompileerd. De compiler stuurde daarna een response terug die de gecompileerde code bevatte in HEX-formaat (zie ook figuur 3.19).

Nu de compiler-server functioneel was, werd er verder getest met verschillende scripts en hardware-boards. Hierbij kwamen al snel enkele zaken aan het licht die de werking tussen de webpagina en de server verhinderden.

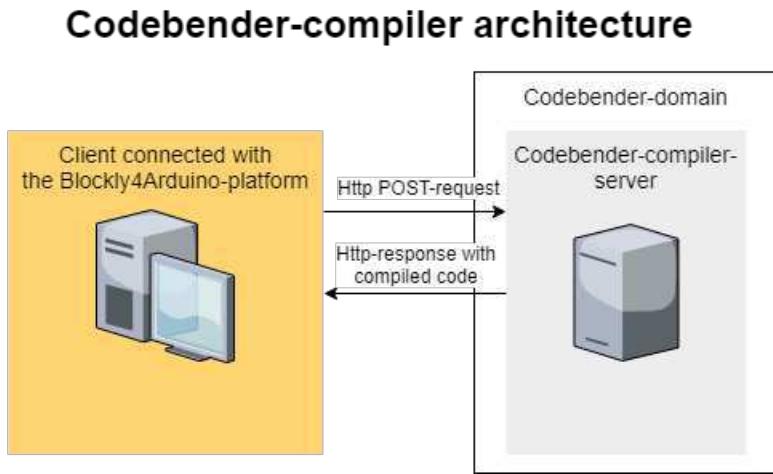
1. Cross-domain error (e.g. CORS) in browser;
2. Oude Symfony-structuur en packages;
3. Libraries meegeven was moeilijk.



Figuur 3.17: Codebender: Compiler-structuur na integratie of Compiler op het domein van Codebender gebruiken.

De eerste problemen deden zich voor bij het testen van de requests gestuurd vanop een webpagina. De POST-methoden kunnen eenvoudiger getest worden met programma's zoals Curl of Postman en deze verliepen zoals verwacht. Echter als een request verstuurd wordt vanop een webpagina, heeft de webbrowser nog enkele veiligheidsmaatregelen die een grote rol spelen. Zo werd op de **Cross-Origin Resource Sharing** error gestoten. Webbrowser zoals Google Chrome en Firefox verwachten bepaalde headers in de request -en response-berichten om ervoor te zorgen dat niet iedereen toegang heeft tot domeinen die niet hetzelfde zijn als het domein van waar het bericht wordt verstuurd. Deze beveiliging zorgde ervoor dat het Compiler-project moest worden aangepast om de respectievelijke headers terug te sturen in de response. De structuur van dit project aanpassen was net iets moeilijker dan verwacht en brengt ons bij het tweede probleem.

Het tweede probleem was dat het ging om een oud Symfony-project met oude packages. De versie van Symfony was 2.3 terwijl de huidige versie 4.0 is. De nodige packages die werden gebruikt voor het project bestonden ook uit oude versies. Het project updaten naar nieuwe versies was niet zo eenvoudig omdat sommige packages niet compatibel waren met versies van andere. Bijgevolg een hele zoektocht om deze toch juist te krijgen. Uiteindelijk



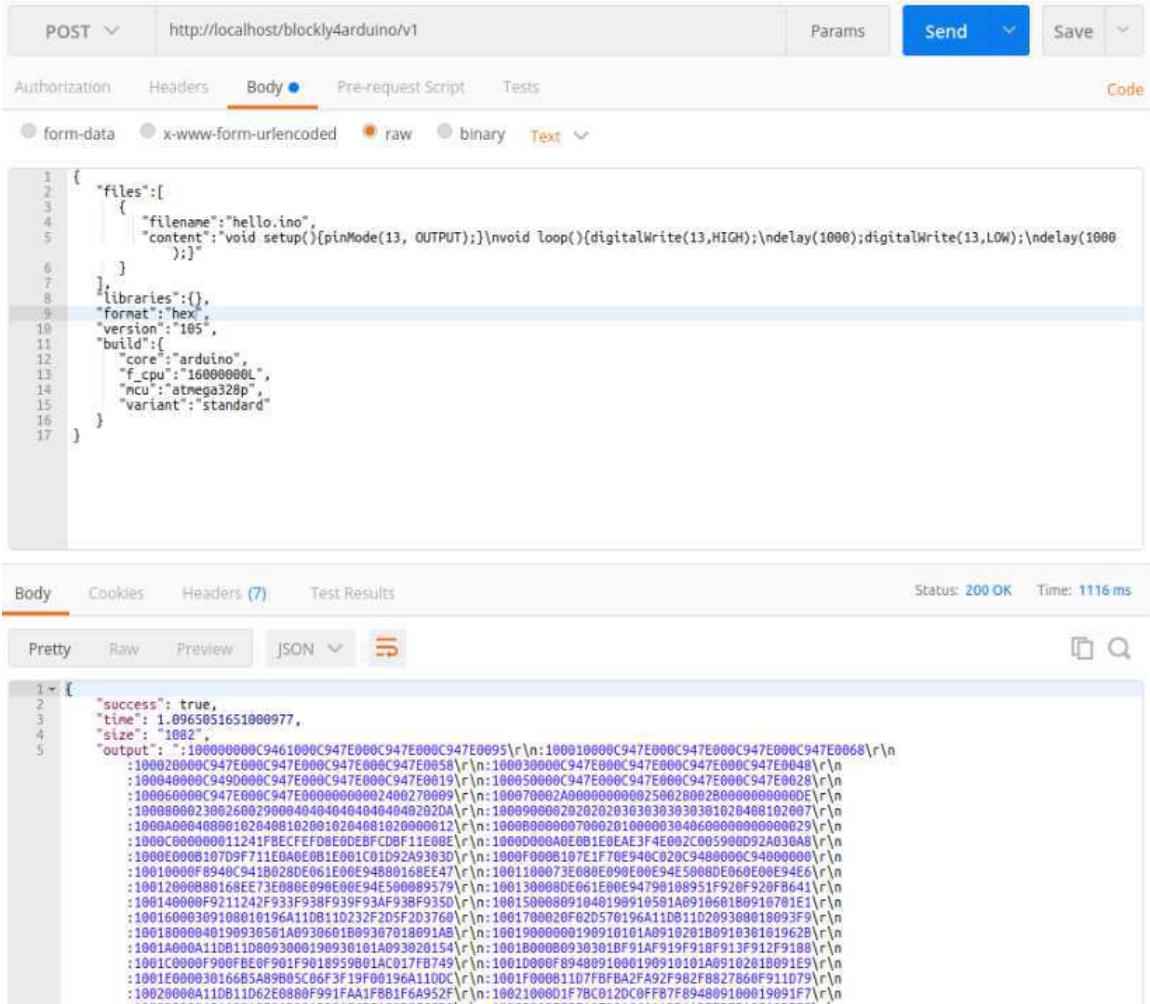
Figuur 3.18: Codebender: Architectuur van de compiler. De compiler ontvangt HTTP-requests en zal de gekregen data compileren om uiteindelijk data in HEX-formaat terug te sturen.

is de update toch geslaagd en zijn de CORS-headers toegevoegd aan de respons. Hierna kon er verder getest worden en dit werd gedaan door verschillende scripts te compileren. Hier kwam het laatste probleem naar boven.

Het derde probleem omvatte het ophalen en compileren van bibliotheken. De compiler beschikte over een beperkt aantal bibliotheken die standaard gecompileerd konden worden. Indien de code een include had van een bepaalde bibliotheek, en deze bibliotheek was niet een van de standaard-bibliotheken, dan moest de volledige code in de bibliotheek meegegeven worden in de JSON-parameter: library. Dit betekende dat er een database zou moeten aangelegd worden van de nodige bibliotheken waarbij deze opgehaald en meegegeven worden indien de code een include van deze bibliotheken bevatte.

Conclusie Codebender Compiler

Het Symfony-project genaamd Compiler biedt de mogelijkheid om de compiler van Codebender zelf op te zetten. De voordelen zijn voornamelijk dat er controle is over de compiler en dat er aanpassingen mogelijk zijn. Nadelen van dit project kwamen vooral ter sprake tijdens het uitwerken en testen van de compiler. Hierbij werd duidelijk dat de structuur en het project duidelijk verouderd zijn, zo kan de compiler maar Arduino-code compileren tot versie 105 terwijl de laatste versie al 106 is. Ook het toevoegen of onderhouden van packages wordt moeilijk omdat deze verder blijven evolueren maar het project mee upda-



The screenshot shows a POST request in Postman. The URL is `http://localhost/blockly4arduino/v1`. The request body is a JSON object:

```

1  {
2     "files": [
3         {
4             "filename": "hello.ino",
5             "content": "void setup(){pinMode(13, OUTPUT);}void loop(){digitalWrite(13,HIGH);\\ndelay(1000);digitalWrite(13,LOW);\\ndelay(1000);}"
6         }
7     ],
8     "libraries": {},
9     "format": "hex",
10    "version": "105",
11    "build": {
12        "core": "arduino",
13        "f_cpu": "16000000L",
14        "mcu": "atmega328p",
15        "variant": "standard"
16    }
17 }

```

The response status is 200 OK, and the time taken is 1116 ms. The response body is a JSON object containing the output of the compilation:

```

1  {
2     "success": true,
3     "time": 1.0965051651000977,
4     "size": 1082,
5     "output": "... (large binary blob of assembly code) ..."

```

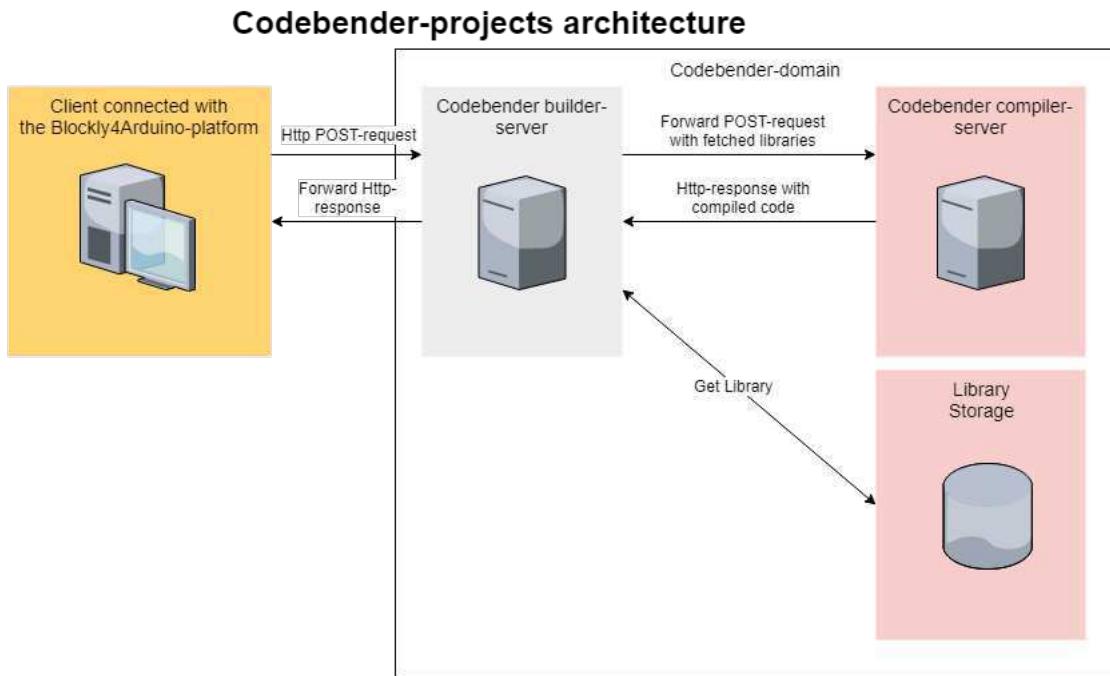
Figuur 3.19: Codebender: POST-request naar de Compiler met meegegeven data in JSON-formaat.

ten is moeilijk door de vele conflicten. Ten laatste is er een inefficiënte manier voorzien om bibliotheken mee te geven om te compileren.

Codebender: Builder

Het derde en laatste project dat beschikbaar is op de Github repository van Codebender gaat over Codebender Builder. Dit is wederzijds een Symfony-project maar heeft een ander doel: de bibliotheken voorzien voor de Codebender Compiler. Dit project zou al meteen een probleem oplossen die we met het Compiler-project hadden. Mede dankzij de slecht voorziene documentatie, was het gebruik of nut van dit project niet duidelijk totdat

het project werd opgezet. Het Builder-project staat in om bibliotheken te detecteren in de scripts en de bijhorende bibliotheek mee te geven met de compiler zodat ieder script gecompileerd kan worden. De architectuur van de verschillende Codebender-projecten ziet er uit als volgt (figuur 3.20).



Figuur 3.20: Architectuur Codebender componenten

De structuur van de verschillende projecten is opgebouwd volgens figuur 3.20, waarbij de Builder-server de centrale eenheid is. Deze ontvangt de POST-request waarbij er wordt onderzocht of de code extra libraries nodig heeft om te kunnen compileren. Indien dit zo is, worden de corresponderende bibliotheken opgehaald bij een gekoppelde opslagplaats en toegevoegd aan het request-bericht. Vervolgens zal de Builder zelf dit request doorsturen naar de Compiler waarbij deze nu de code kan compileren met meegegeven bibliotheken. Eenmaal de compilatie is uitgevoerd, wordt een request-bericht opgesteld van de Compiler en verzonden via de Builder terug naar de Client.

Bij het bespreken van de Compiler werd een architectuur voorgesteld (figuur 3.18) waarbij we een architectuur voorstellen met maar 2 componenten: de client en de Compiler. Omdat de Codebender-projecten los van elkaar opgezet kunnen worden, was het niet duidelijk tijdens de eerste fase dat de Builder-component nodig was. Daarenboven kunnen de Builder en Compiler servers rechtstreeks aangesproken worden door de client via HTTP-requests.

Deze architectuur is omslachtig door het gebruik van 2 servers terwijl het eenvoudiger zou geweest zijn om deze componenten te verbinden tot 1 geheel in plaats van alles te splitsen.

Tabel 3.3: Voor -en nadelen bij het gebruik van Builder-server van Codebender.

Opties	Voordelen	Nadelen
Builder zelf hosten	Bibliotheken worden opgehaald en meegegeven met de Compiler; Mogelijkheid om zaken aan te passen, te verbeteren; Aanpassen naar gelang van gewenste architectuur.	2 server op te zetten (Builder en Compiler); Updates zelf moeten doorvoeren; Nieuwe hardware of andere versies zullen niet gecompileerd kunnen worden; Server hosten brengt een bepaalde kost met zich mee.

Ten slotte worden de voor -en nadelen opgeliist voor het integreren van het Builder-project. Hierbij is het grote voordeel dat de bibliotheken afgehandeld zullen worden en er zelf geen library-management-systeem moet gebouwd worden. Anderzijds is het grote nadeel dat updates naar nieuwe versies en compilatie van nieuwe hardware, zelf uitgevoerd moet worden.

Uitwerking met Codebender Builder

Deze sectie handelt over het opzetten van de Symfony-project om zelf de Codebender Builder te kunnen hosten. Er werd reeds vermeld dat het voordelig zou zijn om zelf deze services aan te bieden waardoor de architectuur van Codebender wordt overgenomen en geïmplementeerd in het Blockly4Arduino-platform. Nu de Compiler werkende is, hoeft enkel nog de Builder-server opgezet te worden om de architectuur af te werken.

Bij het opzetten van de architectuur werden al snel dezelfde problemen vastgesteld als bij het Compiler project: verouderde architectuur en packages die niet meer compatibel waren met recente technologieën zoals PHP7 of recente besturingssystemen zoals Ubuntu. Vandaar bestond de eerste stap uit het updaten en upgraden van het project met de dependencies om de server te kunnen starten.

Bij het starten van de Builder-server, was het belangrijk om 2 koppelingen vast te leggen: deze naar de Compiler en deze naar een locatie waar de extra libraries zijn opgeslagen. Voor de extra bibliotheken volstond het om deze op te slaan onder een lokale folder of het gebruik van een file-server.

De volgende fase bestaat uit het testen van de werking. Om de Builder-server aan te spreken, wordt er opnieuw gebruik gemaakt van HTTP POST-requests die JSON-data bevatten. Hier is echter op te merken dat de structuur van de JSON-data niet dezelfde is als de JSON-data die we naar de Compiler stuurden.

Echter bij het verzenden van een request naar de Builder-server, werd niet de output teruggestuurd die verwacht werd. Hierbij werden de bibliotheken niet juist opgehaald of meegegeven waardoor de Compiler dus de code niet kon compileren. Mits er amper tot geen documentatie beschikbaar was over dit project, is de enige oplossing om het PHP-project trachten te debuggen, te reverse engineeren of alles stap voor stap te controleren en te verbeteren. Op de server werd bij elk request een error-bericht weergegeven die geen nuttige informatie bevatte om te achterhalen waar het probleem zich bevond. Om wille van de verouderde Symfony-projecten was de kans ook groot dat de fout te wijten was aan een package-mismatch waardoor sommige functies niet meer correct functioneerden.

Conclusie Codebender Builder

Het library-probleem dat de Compiler-server had, kon in theorie worden opgelost met de Builder-server. Deze Builder zou ervoor zorgen dat meegegeven bibliotheken werden gedetecteerd en opgehaald om mee te geven aan de Compiler zodat de code correct gecompileerd kon worden. In praktijk kwamen er enige problemen naar boven omtrent de benodigde packages die outdated waren alsook de effectieve werking van de Builder die niet zijn werking deed. Om deze problemen op te lossen zou er veel tijd moeten gespendeerd worden om deze architectuur functioneel te maken alsook het feit dat de projecten verouderd waren. Dit wil zeggen dat bepaalde packages niet meer toegankelijk zijn en alles geüpdate moet worden. Daarnaast zou de volledige architectuur eigenlijk best herschreven worden, de reden hiervoor is dat naar de toekomst toe het niet zeker is hoelang deze oude projecten nog bruikbaar zijn. Bij het onderzoek bleek dat de deze architectuur hetgeen was dat gezocht werd om de architectuur van Blockly4Arduino uit te breiden. Nu in de praktijk de projecten zijn onderzocht en getest, bleken deze niet zo geschikt te zijn. Vervolgens moet nu de vraag gesteld worden of er verder tijd wordt gestoken in het aanpassen van de Codebender architectuur of er een andere oplossing kan gemaakt worden die beter future-proof is dan de implementatie via Codebender.

3.2.3 Implementatie via Arduino.cc

In vorige sectie: Implementatie via Codebender, werd getracht de architectuur uit te breiden van het Blockly4Arduino-platform door een compiler-server op te zetten van Codebender die de code compileert tot HEX-formaat. De volledige werking was echter niet functioneel mits problemen met het Codebender-Builder-project. Ook werd vastgesteld dat de structuur en de code niet meer geüpdatet worden en dit zou kunnen problemen geven naar de toekomst toe. Hierbij werd de vraag gesteld of het nog nuttig was om het Codebender-project verder uit te werken en of het niet beter zou zijn om een andere implementatie te voorzien.

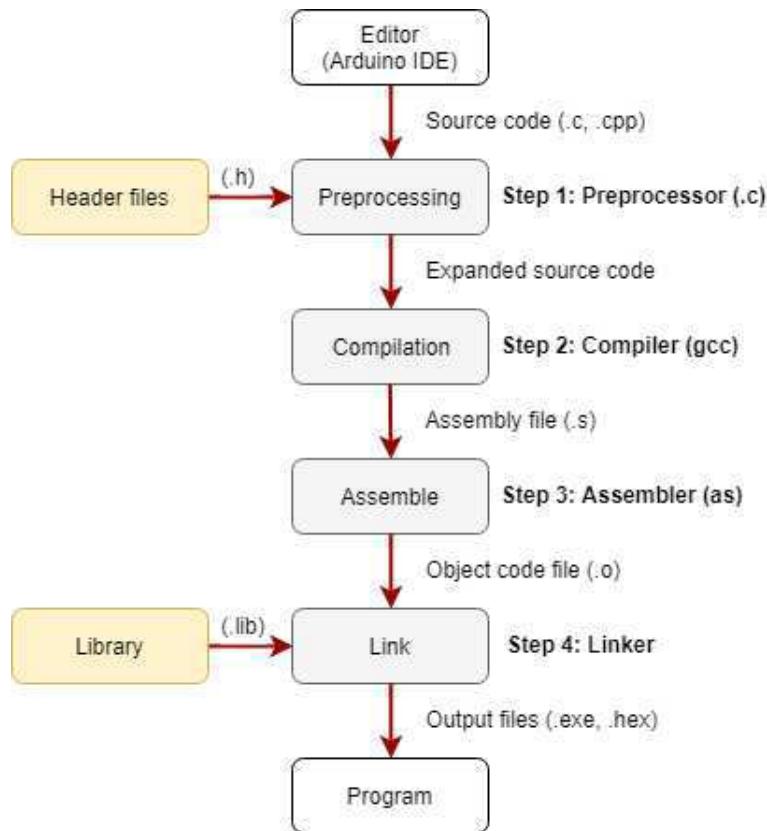
Hierbij werd beslist om een andere implementatie te zoeken, eerst en vooral werd er terug geblikt op het onderzoek en meer bepaald de mogelijke oplossingen. Uit tabel 2.1 bleek Codebender de eerste optie te zijn maar niet de enige. Zo is het ook mogelijk om een compiler op te zetten via de officiële Arduino-software.

Het Arduino.cc-platform heeft 2 mogelijke manieren waarop gebruikers code kunnen schrijven en uploaden naar hun Arduino boards: de Arduino IDE en Arduino Create.

De Arduino IDE of **Integrated Development Environment**, is een grafische omgeving geprogrammeerd in Java. De gebruikers die met de Arduino software werken, kunnen programma's schrijven om te uploaden naar hun Arduino board in C of C++. Deze software kan worden geïnstalleerd op verschillende platformen zoals Linux, Mac OS, Windows en recentelijk ook Chrome OS. De gebruikers van Arduino zijn hoofdzakelijk ontwikkelaars, designers of hobbyisten. Arduino maakt gebruik van Github en Git als versie-controlesysteem om ontwikkelaars te laten samenwerken aan de software. Contributies aan de software komen enerzijds van het Arduino software development team en anderzijds van de open-source community. De Arduino IDE voert sinds versie 1.6.1 niet meer zelf de compilatie uit, achterliggend wordt een programma aangesproken die de compilatie doet, namelijk arduino-builder. Eenmaal het project gecompileerd is, wordt een tweede tool aangesproken om het board te programmeren: AVRDUDE.

Figuur 3.21 stelt het stappenplan² voor van hoe een project wordt gecompileerd met de Arduino IDE. De Arduino IDE spreekt achterliggend de arduino-builder tool aan, deze tool onderneemt een aantal stappen om een project te compileren, aangeduid in het grijs.

²<https://github.com/arduino/Arduino/wiki/Build-Process>



Figuur 3.21: Arduino-builder: Stappenplan van het compileer-proces met de avr gcc-compiler.

Samengevat ziet de structuur er uit als volgt:

1. Arduino IDE:
 - (a) Arduino-builder:
 - i. Preprocessing;
 - ii. Compile;
 - iii. Assemble;
 - iv. Link.
 - (b) AVRDUDE.

De preprocessor voert enkele transformaties uit op de main sketch-file vooraleer deze wordt doorgegeven naar de avr-gcc compiler. Alle project-bestanden worden namelijk samengevoegd tot 1 main sketch file (C++). Daarna worden alle `#include[header].h`-regels toegevoegd aan de sketch. Deze header-files bevatten alle nodige definities voor de Arduino

core. Daarna worden alle functies gezocht in de main file en worden er declaraties voorzien per functie.

Sketches worden gecompileerd door avr-gcc en avr-g++ afhankelijk van het geselecteerde hardware board. De compiler heeft toegang nodig tot enkele folders: project-folder, bibliotheek-folder, Arduino core-folder en de hardware-folder. Voordat alle .c of .cpp-files gecompileerd worden, wordt een poging gedaan om een voorgaand object-file (.o) te hergebruiken om het build proces te versnellen. Wanneer een sketch wordt gecompileerd, wordt dit gebuild in een temporary folder.

De assembler heeft als taak om de code om te zetten naar machine taal (.o), afhankelijk van de geselecteerde hardware.

Ten slotte zorgt de linker voor het samenvoegen van alle object files alsook het toevoegen van bibliotheken om uiteindelijk een uitvoerbaar bestand (.exe/.elf) of HEX-file te bekomen. Enkel het gedeelte dat gebruikt wordt uit de bibliotheken wordt toegevoegd om de grootte van de HEX-file te beperken.

De HEX-file is de finale output van de compilatie. Om de hardware nu ook nog te programmeren, wordt een tweede tool gebruikt door de Arduino IDE: AVRDUDE. AVRDUDE is een bootloader die in staat is om het flash geheugen van Arduino processoren te programmeren via serial of USB. Deze bootloader implementeert een subset van het STK500 8bit protocol dat ontworpen is om AVR chips te programmeren.

Naast de Arduino IDE bestaat er ook een online web-editor om projecten te maken en te uploaden vanuit de browser, deze noemt Arduino Create. Arduino Create bevat een text-editor waarin de gebruiker code kan schrijven om de hardware aan te sturen. Net zoals bij Codebender.cc is er een verificatie -en upload-knop voorzien om de code vanuit een online omgeving te compileren en te flashen. Achterliggend gebruikt Arduino Create dezelfde tool als de Arduino IDE om de compilatie uit te voeren: arduino-builder. In voorgaande paragrafen werd uitgelegd hoe de compilatie-stap ineen zat. Om de hardware te programmeren maakt Arduino Create gebruik van een extensie om toegang te krijgen tot de USB-poorten (meer hierover in sectie 3.2.4).

Ten slotte wordt de open-source Github projecten aangekaart. Aangezien het Arduino-platform ontworpen is door en voor developers, is er continue integratie van nieuwe technologieën en blijft de gebruiker up-to-date met zijn/haar projecten. Er werd eerder al vermeld dat Arduino werkt met Github en Git waardoor het arduino-builder project beschikbaar is

om te gebruiken en aan te passen. In volgende sectie maken we een uiteenzetting van hoe de arduino-builder gebruikt kan worden om de sketchen te compileren die gemaakt worden op het Blockly4Arduino-platform.

Implementatie arduino-builder

De tool die wordt gebruikt door de Arduino IDE en Arduino Create om projecten te builden en code te compileren, is beschikbaar op Github als open-source zodat ontwikkelaars en hobbyisten kunnen helpen om het Arduino-platform uit te breiden en te verbeteren. In deze sectie zal uiteengezet worden hoe dit project kan geïntegreerd worden in het Blockly4Arduino-platform om sketchen te compileren.

Tabel 3.4: Voor -en nadelen bij het gebruik van de arduino-builder tool.

Opties	Voordelen	Nadelen
arduino-builder gebruiken	Volledige projecten builden; Tijdens compilatie worden de nodige libraries opgehaald en meegegeven; De niet standaard bibliotheken zijn eenvoudig toe te voegen via de Arduino IDE; Er verschijnen regelmatig nieuwe updates; Cross-platform (Windows, Linux en Mac OS).	Executable die per client moet geïnstalleerd en uitgevoerd worden; Creatie temporary files tijdens compilatie-proces.

In figuur 3.21 werden de stappen al beschreven die de arduino-builder-tool gebruikt om een project te builden. Daarnaast werd ook uitgelegd dat de Arduino IDE en Arduino Create achterliggend gebruik maken van deze tool. Vervolgens wordt zelf uitgewerkt hoe de tool kan aangestuurd worden om een project te builden en een HEX-file te bekomen. Eerst en vooral worden de voor -en nadelen opgesomd in tabel 3.4 bij het gebruik van de arduino-builder tool. Deze tool biedt vele mogelijkheden maar heeft het nadeel dat de executable lokaal uitgevoerd moet worden en remote niet beschikbaar is. Dit nadeel is eigen aan de huidige versie van de Arduino Software maar dit betekent niet dat er geen oplossing kan voor gevonden worden.

Het uiteindelijke doel is dat het Blockly4Arduino-platform zelf sketches (programma's) kan compileren, indien de arduino-builder tool gebruikt wordt, moet de architectuur aangepast

worden zodat het platform in staat is om de tool aan te spreken. Om de arduino-builder te integreren worden 2 implementatie-stappen voorzien:

1. Het aansturen en het gebruik van de tool;
2. Server bouwen die achterliggend de tool zal aanspreken.

Aansturen en gebruik arduino-builder

Eerder werd al vermeld wat de arduino-builder doet om een sketch te compileren en om te zetten naar een binary file maar niet hoe deze tool aangestuurd wordt. Deze sectie zal dieper ingaan op het concreet aansturen van de software en de vereisten om deze tool te kunnen aansturen.

De Arduino Software maakt gebruik van verschillende modules om een overzicht te creëren in duizenden lijnen code. Door het gebruik van de modules, is het mogelijk om een duidelijke weergave te krijgen in een complexe software. Tabel 3.5 geeft de verschillende modules weer met bijhorende rollen.

Tabel 3.5: Hoofd modules met bijhorende rolverdeling

Module	Rol
app	Map die de Grafische User-Interface (GUI) bevat.
arduino-core	Map die de hoofdonderdelen van de IDE bevat (libraries en source folders).
build	Deze map is bedoelt voor het builden en aanpassen van de IDE.
hardware	Alle nodige hardware-componenten (AVR, bootloader, ...) die gebruikt en ondersteund worden door Arduino.
libraries	Standaard bibliotheken die gebruikt worden door Arduino tijdens compilatie, zijn hier opgeslagen.

Aangezien de Arduino Software rekent op deze modules, rekent de arduino-builder ook op deze modules. Dit wil zeggen dat bij het aansturen van de tool, de paden naar de verschillende modules meegegeven moeten worden. Tijdens het lezen van de documentatie over de arduino-builder tool, kwamen een aantal interessante zaken naar boven:

1. Software is gebouwd met Google Go-code;
2. Tool is enkel beschikbaar als command-line tool;

3. Arduino-builder is in staat om de Arduino Hardware specificaties te verhandelen, het correct aansturen van gcc en het produceren van gecompileerde sketches.

Bij bovenstaande zaken wordt het duidelijk dat we de tool kunnen aanspreken via command line of een bash-script. Dit zal handig zijn voor het verdere verloop van de implementatie omdat we een applicatie kunnen bouwen die een bash-script zal uitvoeren. Ten tweede is het belangrijk om te noteren dat het gaat over een Google Go-project. Dit wil zeggen dat lokaal de go-module moet geïnstalleerd zijn om de tool te kunnen uitvoeren.

Vermits de aansturing command-line gebeurt, wordt eerst een bash-script geschreven die deze tool zal aanspreken. Bij het bash-script moeten de paden meegegeven worden van de onderliggende modules (zie Listing 3.1).

Codefragment 3.1: Bash-script to compile a script using the arduino-builder tool

```
#!/bin/bash

LOCALARDUINOPATH=PATH_TO_YOUR_ARDUINO_IDE # e.g. ./Arduino
BUILDERPATH=PATH_TO_YOUR_LOCAL_ARDUINO FOLDER # e.g. ./arduino-1.8.5

SKETCHNAME=$1                                # first system parameter is
                                              sketchname
BOARDNAME=$2                                 # second system parameter is
                                              boardname e.g. 'arduino:avr:uno'

LOCALBUILDPTH=PATH_TO_TEMP_DIRECTORY          # e.g. ./temp
LOCALSKETCHPATH=PATH_TO_THE_DIRECTORY_OF_SKETCH # e.g. ./Documents

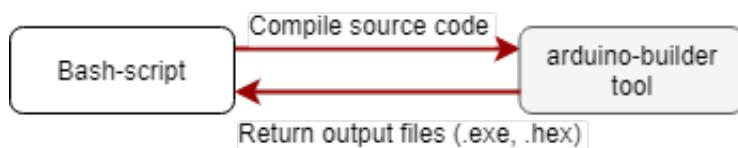
mkdir $LOCALBUILDPTH                          # Make a subfolder in the temporary
                                              directory to store the output files

$BUILDERPATH/arduino-builder -compile -hardware $BUILDERPATH/hardware
  -build-path $LOCALBUILDPTH -tools $BUILDERPATH/hardware/tools -tools
  $BUILDERPATH/tools-builder -libraries $BUILDERPATH/libraries -libraries
  $LOCALARDUINOPATH/libraries -fqbn $BOARDNAME
  $LOCALSKETCHPATH/$SKETCHNAME'.ino'
```

Indien gebruik wordt gemaakt van Arduino IDE, zal de IDE de correcte verwijzing naar de paden van de modules verzorgen. Echter als de tool rechtstreeks wordt aangesproken zoals

in het script (Listing 3.1), moeten deze paden worden meegegeven. Naast de vereiste modules, moeten nog een paar andere zaken worden meegegeven. De eerste parameters betreft *-compile*, *-dumb-pref* of *-preprocess*. De optie compile zal een sketch compileren en een binary file als output genereren, dumb-pref zal alle gebruikte voorkeuren oplijsten en ten slotte zal de optie preprocess de samengevoegde sketch afprinten (tijdens preprocessing worden alle sketches en bestanden samengevoegd, zie 3.21). De optie die gebruikt wordt is natuurlijk *-compile* omdat we sketches willen laten compileren door de arduino-builder tool. Een andere parameter die wordt meegegeven is: *-fqbn*. Deze parameter staat voor **Full Qualified Board Name** en vraagt voor welk type Arduino board de compilatie moet gebeuren. Een laatste parameter die naast de modules wordt meegegeven is locatie naar de sketch-file (.ino).

Met het script en de juiste parameters, is het nu mogelijk om via bash de tool aan te sturen, dit is nogmaals geïllustreerd in figuur 3.22.



Figuur 3.22: Aansturing arduino-builder tool via Bash

Server voor aansturing arduino-builder

In vorige sectie werd besloten dat er twee stappen nodig waren om de implementatie op te zetten via de arduino-builder. De eerste stap bestond uit het aansturen van de tool en de tweede stap uit het bouwen van een server die achterliggend de aansturing kon verzorgen. Mits in vorige sectie het succesvol verlopen is om via een Bash-script de tool aan te sturen, is het ook mogelijk om via een andere weg de tool aan te spreken. In deze sectie wordt uiteengezet hoe we tool kunnen aanspreken rekening houdende met het Blockly4Arduino-platform.

Het doel is om op het Blockly4Arduino-platform te compileren of een server te hebben die compileert en de gewenste output teruggeeft. Er werd gekozen om een HTTP-server op te zetten in Node.js die achterliggend de arduino-builder tool zal aansturen bij een inkomende POST-request. Node.js is een runtime environment die de mogelijkheid biedt om JavaScript uit te voeren aan de server-zijde. Daarnaast is de engine gebouwd op

dezelfde V8 JavaScript engine die Google Chrome gebruikt. De reden waarom Node.js gekozen werd boven een andere omgeving is in tabel 3.6 te vinden (Chitra, 2017) [21]. Hierbij werd duidelijk dat Node.js betere voordelen biedt voor de server die gebouwt moet worden om de volgende hoofdredenen:

1. Snelheid;
2. Non-blocking code;
3. Concurrent request handling.

Het eerste puntje spreekt voor zich dat snelheid een belangrijke factor vormt. De gebruikers van het platform verwachten wel enige vertraging maar hoe sneller de requests kunnen behandelt worden, hoe beter. Een tweede belangrijke factor is de niet-blokkerende code, door het gebruik van callback functies is het heel onwaarschijnlijk dat de server zal vastlopen of crashen. Een nadeel hierbij is wel dat de vele callbacks de code onoverzichtelijk of complex kunnen maken. Ten derde en de meest belangrijke factor is de concurrent request handling, via Node.js kunnen requests gelijktijdig uitgevoerd worden waardoor de delay al sterk wordt beperkt.

Tabel 3.6: Node.js vs Symfony

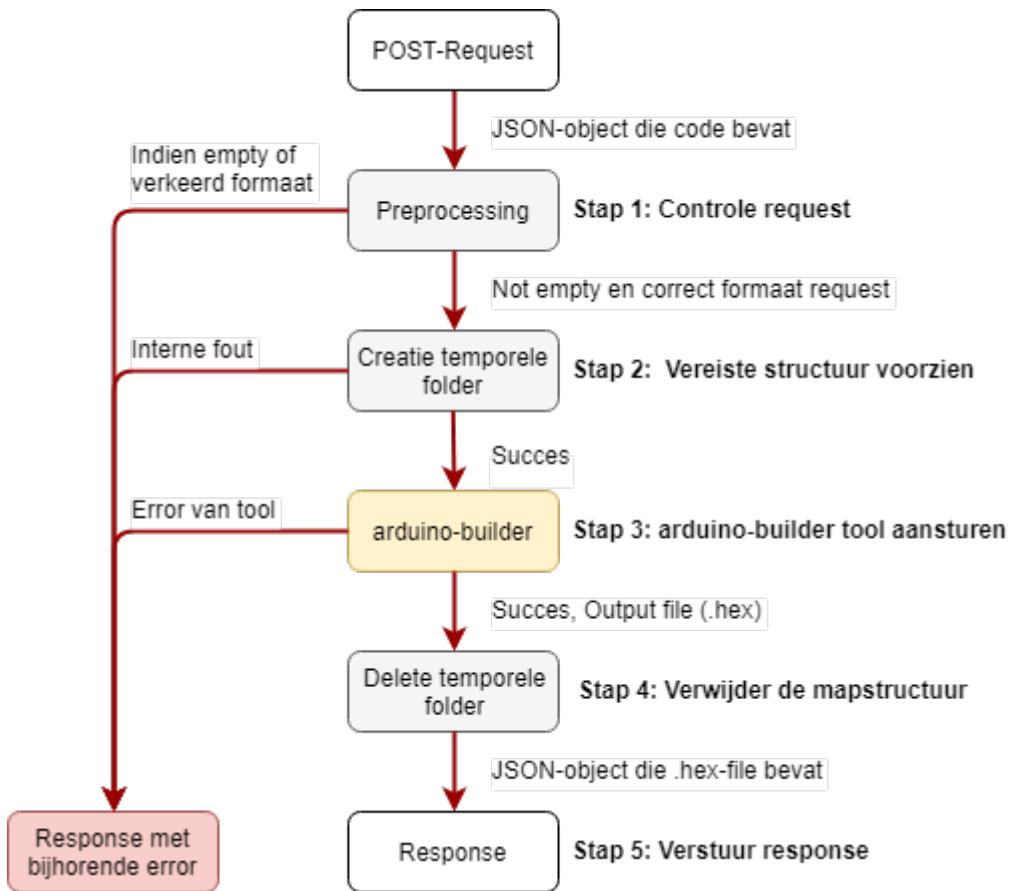
<i>Opties</i>	<i>Node.js</i>	<i>Symfony</i>
Voordelen	JavaScript + JSON; Non-blocking code d.m.v. callbacks; Snelheid en performantie; Concurrent request handling; Heel groot aanbod van gratis tools via npm; Code sharing en hergebruik; Asynchroon; Event-gebaseerd; Easy to learn; Lightweight.	PHP; Grottere code-basis; Connectie met SQL databases; Groot aanbod van tools via composer; Compatibel met alle hosting systemen; Eenvoudig te deployen.
Nadelen	Moeite met heavy-computing; SQL databases aanspreken is iets moeilijker; Callback-functies kunnen voor teveel geneste lussen zorgen.	MVC niet zo eenvoudig op te zetten; Grote overhead; Langere request-processing chain; Requests worden 1 per 1 afgehandeld.

Nu de technologie vastligt waarmee de server wordt opgezet, kan de uiteindelijke implementatie beginnen. Eerst en vooral wordt een HTTP-server opgezet waarmee verbinding kan gemaakt worden via een webbrowser door naar `http://localhost` te surfen. Figuur 3.23 geeft de interne werking weer van de geschreven server in Node.js. De stappen lopen vanaf het moment dat een request binnentkomt tot het moment dat de antwoord wordt verzonden.

Het POST-request bevat een JSON-object met daarin de code die op het Blockly4Arduino-platform genereert is. De server zal uit het request-bericht deze code halen om te laten pre-processen (stap 1). Deze eerste stap zal nagaan of het request-bericht de juiste JSON-structuur bevat en of deze data niet null of empty is. Indien deze velden correct zijn wordt overgegaan naar stap 2 of als de data niet in het verwachte formaat staat, wordt een error-response teruggestuurd. Stap 2 omvat de juiste structuur voorzien, tijdens het aansturen van de arduino-builder tool werd het duidelijk dat er verschillende stappen werden doorlopen (ook reeds besproken). Deze stappen genereren elk een aantal bestanden waardoor een locatie moet voorzien zijn om deze bestanden naar toe te schrijven, standaard komt dit in de *temp*-folder van het systeem terecht. Per project wordt een temporary folder aangemaakt waarin de arduino-builder tool zijn bestanden zal plaatsen. Bij het correct aanmaken van een temporary folder zal overgegaan worden naar stap 3. Deze stap is het aansturen van de tool via een bash-command (zie het bash-script in Listing 3.1). Bij het uitvoeren van het bash-command, zijn er 3 callbacks: Out, Error en Close. Statusberichten van de tool worden geplaatst op Out en dan zijn er 2 opties: ofwel wordt er een Error-event geactiveerd waardoor zeker is dat er een error is opgetreden ofwel wordt een Close-event geactiveerd waardoor zeker is dat de tool succesvol heeft gecompileerd. Indien een error-event zich voordoet, wordt een response-bericht opgesteld met het error-bericht. Anderzijds wordt verdergegaan naar stap 4 en 5. Deze stappen wissen de temporary folder om geheugen te besparen en er wordt een JSON-object aangemaakt waarin de HEX-file terechtkomt. Dit JSON-object wordt vervolgens teruggezonden in een response-bericht naar het Blockly4Arduino-platform.

Nu de server operationeel is, kunnen er naar andere aspecten gekeken worden zoals het aantal requests die verwerkt kunnen worden, de response-tijd en veiligheid. Voor de snelheid en aantal requests wordt verwezen naar hoofdstuk 4, sectie 4.1.

Qua veiligheid kunnen er al snel aanpassingen gebeuren om deze server een stuk veiliger te maken.



Figuur 3.23: Stappenplan werking Node.js-Server, van request tot response

1. Binnenkomende requests enkel toestaan indien ze gestuurd worden vanuit een bepaald domein;
2. HTTP-server upgraden naar HTTPS d.m.v. public/private key + certificaat;
3. CORS-headers toevoegen.

Indien de binnenkomende requests geblokkeerd worden indien ze niet vanaf een bepaald domein worden verstuurd, kan iemand met kwade bedoelingen al geen request sturen. Een tweede mogelijk zou zijn om de berichten te encrypteren door te upgraden naar HTTPS maar hierbij kan de vraag gesteld worden of dit überhaupt nuttig is. De verzonden berichten bevatten geen user-data en indien ze onderschept worden is dit niet erg. Ten slotte is er de optie om CORS-headers toe te voegen waardoor cross-domain berichten zenden en ontvangen wordt uitgesloten. Dit kan gelinkt worden aan het eerste item.

3.2.4 Google Chrome extensie

In vorige secties werden verschillende compilers onderzocht en geïmplementeerd. Uiteindelijk werd een werkende implementatie bekomen via de arduino-builder tool. Via HTTP-request -en response berichten is het nu mogelijk dat een website Arduino code kan compileren. Wat echter nog niet mogelijk is, is het aansturen van de hardware zelf vanuit de browser. Om wille van veiligheidsredenen is het niet meer mogelijk om met enkele lijnen JavaScript, toegang te krijgen tot de USB-poorten van de client. In deze sectie wordt dieper in gegaan op het ontwikkelen van een oplossing om toch toegang te krijgen tot de seriële -en USB-poorten van de client.

Chrome API

Officieel is er medegedeeld dat Google Chrome Apps discontinued³ worden. Dit wil zeggen dat applicaties niet meer stand-alone kunnen draaien. De developers van Chrome raden dan ook aan om de applicaties om te bouwen naar extensies. Verschuiving naar extensies betekent dat de technologie enkel beschikbaar is voor gebruikers die Google Chrome gebruiken. Google Chrome is cross-platform wel beschikbaar voor zowel Windows, linux, Mac OS en Chrome OS.

De eerste stap bij het zoeken naar een oplossing voor de USB-poorten, is opnieuw bestaande technologieën gaan onderzoeken om uiteindelijk een beslissing te maken op hetgeen het beste past als oplossing. Tijdens het onderzoeken van het Codebender.cc en het Aruino Create platform, werd duidelijk dat deze platformen gebruik maakten van een browser extensie.

Om de scope van technologieën te verkleinen, wordt rekening gehouden met onderstaande vragen.

1. Welke Arduino boards moeten ondersteund worden? A: Zo veel mogelijk, te starten met de meest gebruikte;
2. Gebruiken alle Arduino boards dezelfde USB interface? A: Ja;
3. Gebruiken alle Arduino boards hetzelfde communicatie-protocol? A: Nee;

³<https://developer.chrome.com/apps/migration>

4. Bestaan er al packages die ons kunnen helpen met het uitbouwen van onze applicatie?
A: Ja.

Op basis van bovenstaande vragen werden volgende opties opgesteld:

1. Browser Serial API of WebUSB API;
2. ATMEL AVR STK500_v1 en STK500_v2 Protocollen;
3. Verder bouwen op bestaande technologieën (e.g. AVRDUDE);

Serial API & WebUSB API zijn JavaScript API's toegankelijk in de meeste browsers (cross-platform), die de mogelijkheid bieden om verbinding te maken met aangesloten hardware. Deze API's geven manufacturers van hardware de kans om hun drivers te beschrijven en aan te sturen vanuit een browser. Het doel van deze API's is meer om data uit te wisselen tussen browser en de hardware dan echt de mogelijkheid om de hardware te programmeren/flashen. E.g. drukknop indrukken op Arduino verzend een string die dan via de API op de webpagina getoond wordt. Na een kleine test bleek het niet te lukken om verbinding te maken met een aangesloten Arduino board. In de officiële documentatie van de API's is dit niet geüpdatet maar er is een catch met het gebruik van Serial -en WebUSB API. Doordat manufacturers hun drivers blootstellen brengt dit grote veiligheidsrisico's met zich mee waardoor de recente versies van browsers deze API's niet meer ondersteunen. Zo kan er snel en eenvoudig malware geïnjecteerd worden op zowel de computer als de hardware. Deze manier kan dus geschrapt worden.

In de gestelde vragen kwam aan bod of alle Arduino boards met hetzelfde communicatie-protocol aan te sturen zijn. Origineel zijn er specifiek protocollen ontworpen om de ATMEL AVR chips te flashen, aangezien de microchips compatibel zijn met de AVR chips kunnen deze protocollen gebruikt worden voor het flashen van Arduino boards. Het originele protocol heet STK500 en is ondertussen vervangen door STK500_v2. Er bestaan andere versies van het STK-protocol maar deze 500 versie volstaat voor de Arduino boards. Daarnaast is er een handige JavaScript API gebouwd die achterliggend gebruikt maakt van dit protocol om via deze API hardware te kunnen flashen. Tools zoals de alom bekende AVRDUDE zijn gebouwd gebruik makende van deze API. Hieronder zijn de verschillende versies opgeliist van het protocol die nodig zijn om de volledige range van Arduino boards aan te sturen.

1. AVR061 / STK500 v1: (ATMega328 chips);

2. AVR068 / STK500 v2: (e.g. Arduino Mega, ...);
3. AVR109 (e.g. Arduino Leanardo, ...).

Met bovenstaande protocollen en API's is het nu mogelijk om hardware te flashen. Dit kan gebruikt in het Blockly4Arduino verhaal doordat een hele range aan Arduino boards geflasht kunnen worden met HEX-files. Vervolgens zijn er 2 mogelijke manieren om verder te gaan:

1. Eigen implementatie van het STK500 protocol;
2. Bestaande tools gebruiken (e.g. tool zoals AVRDUDE).

De meest voor de hand liggende werking zou deze zijn via de AVRDUDE tool. AVRDUDE staat voor **AVRDownloader/UploaDEr** en is een command-line tool die de mogelijkheid biedt om op verschillende platformen hardware te flashen. Deze technologie bestaat al een eindje en wordt ook gebruikt door de Arduino IDE om de hardware te flashen. Er is echter nog opzoekingswerk gedaan of er geen andere mogelijkheden waren, hierbij kwam AVRGIRL naar boven. AVRGIRL staat voor **AVRGeneal ISP Programming tool** en is gebouwd om dezelfde functionaliteit te bieden als AVRDUDE maar geschreven in JavaScript en Node.js. In tabel 3.7 zijn de voor -en nadelen opgesomt bij het gebruik van AVRGIRL.

Tabel 3.7: Voor -en nadelen van de AVRGIRL tool om Arduino boards te flashen.

<i>Opties</i>	<i>Voordelen</i>	<i>Nadelen</i>
Flashen via AVRGIRL	Gebruik van de STK500 protocollen; Open-source en beschikbaarheid op GitHub; Node.js equivalent van AVRDUDE; Simpele implementatie voorzien van een Google Chrome extensie; Ondersteunt een brede range aan Arduino boards en ATMEL chips.	Andere hardware niet compatibel.
Google Chrome Extensie	Maak achterliggend gebruik van de AVRGIRL tool; Toegang tot USB-poorten van client mits toelating.	Beperkt tot Google Chrome browser.

In bovenstaande tabel werd ook een Google Chrome extensie vermeld. Deze AVRGIRL tool heeft namelijk de documentatie voorzien om een Chrome extensie te bouwen die achterliggende de tool gebruikt. Er is namelijk 1 goede reden waarom de implementatie een extensie vereist: toegang tot de USB-poorten. In bovenstaande paragrafen werd vermeld dat het vroeger mogelijk was om met JavaScript-code hardware aan te sturen vanuit de browser. Om veiligheidsredenen is hier vanaf gestapt en de functionaliteit verschoven naar browser extensies. Door het gebruik van een extensie, wordt de gebruiker geïnformeerd over mogelijke veiligheidsrisico's en moet de gebruiker toegang verlenen aan de extensie om de USB-poorten aan te sturen. Er is meer controle en webpagina's kunnen achterliggend de hardware niet meer raadplagen om malware-injectie te veroorzaken of privacy te schenden. Om hardware aan te sturen vanuit een webpagina is dus 1 extra stap nodig: het gebruik van een extensie.

Een Chrome extensie is een klein stukje software die de functionaliteit van de browser kan verbeteren of uitbreiden. Een extensie bestaat uit een aantal onderdelen:

1. Manifest-file, definieert:

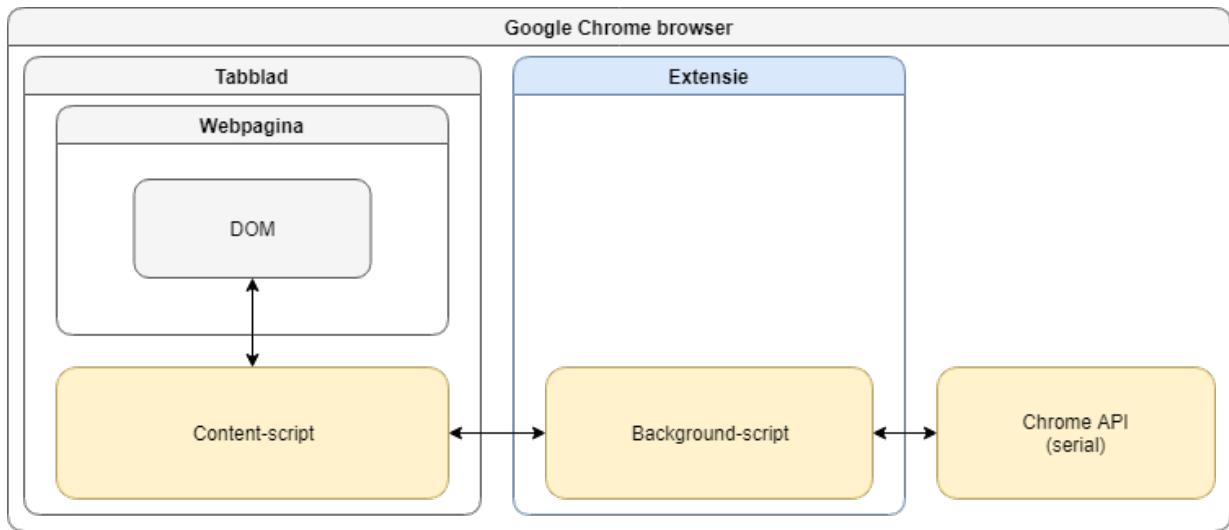
- (a) Naam;
- (b) Versie;
- (c) Permissies;
- (d) Background-script;
- (e) Iconen.

2. Background-script;

3. Resource files.

De manifest-file bevat alle informatie zodat de browser weet wat tot de extensie behoort. Een background-script is een niet-zichtbare pagina die de main-logic van de extensie bevat. Daarnaast wordt de Chrome.runtime API gebruikt door de extensie. Deze API haalt de background-pagina op, geeft informatie over de manifest-file terug en luistert en antwoordt op events in de extensie. Figuur 3.24 geeft een overzicht weer van de architectuur die bekomen wordt door het toevoegen van een Chrome extensie.

In bovenstaande figuur is de architectuur beschreven waarop de extensie wordt aangesproken en waarop de extensie op zijn beurt toegang verkrijgt tot de USB-poorten via de



Figuur 3.24: Architectuur Chrome extensie. Een content-script van de webpagina is in staat om te communiceren met het background-script van de extensie. Dit background-script kan op zijn beurt de API's aanspreken die beschikbaar zijn in de browser.

Chrome API. Naast de HTML van de webpagina is ook een content-script aanwezig geschreven in JavaScript. Dit script is in staat om verbinding te maken met het background-script van de extensie indien het correcte ID is meegegeven. Het background-script bevat de code om via AVR GIRL aangesloten hardware op te lijsten en te flashen. Het background-script is wel in staat om dit te doen in tegenstelling tot het content-script omdat een Chrome extensie wel nog toegang heeft tot alle Chrome.* API's.

Google Chrome bevat 2 API's voor het versturen van berichten met onderstaand type:

1. Simpel one-time requests;
2. Long-lived connections.

De geschreven extensie maakt gebruik van beide types. De one-time requests worden verstuurd vanuit content-script om na te gaan of de gebruiker de gewenste extensie heeft toegevoegd aan de Chrome browser. Het tweede type wordt gebruikt om long-lived connections op te zetten want het flashen van de hardware gebeurt niet onmiddelijk. Om geen time-out te verkrijgen wordt een long-lived connection gebruikt.

Naar beveiliging toe, kan er in de manifest-file gedefinieerd worden welke domeinen toegang hebben tot de extensie. Hierdoor is het momenteel enkel mogelijk om vanuit Blockly4Arduino

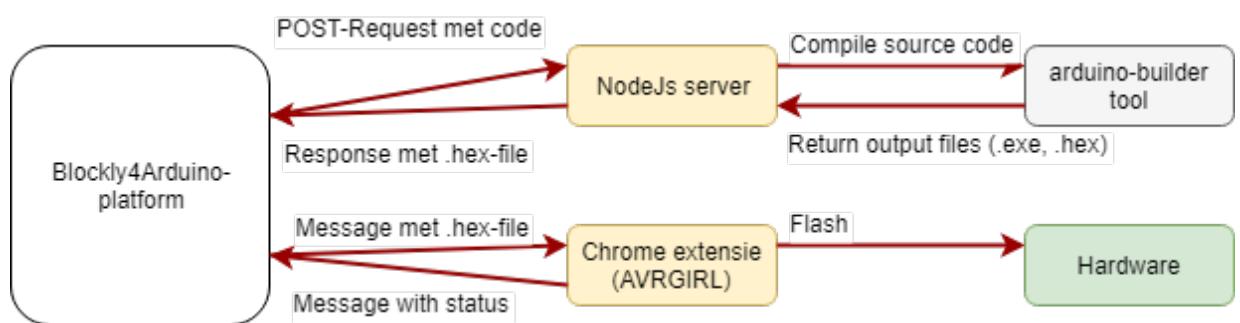
de extensie te gebruiken want andere domeinen zijn opgegeven. Deze stap is toegevoegd om niet iedereen toegang te verlenen.

Conclusie extensie

Via een extensie is het toch mogelijk om toegang te verkrijgen tot de aangesloten hardware via de Chrome browser. De geschreven extensie maakt gebruik van AVRGIRL, die staat in staat om alle types Arduino boards te flashen. Natuurlijk moet ook nog de connectie gelegd worden tussen het Blockly4Arduino-platform en de extensie en dit gebeurt aan de hand van short-lived connections om de beschikbaarheid te testen en het versturen van de data voor het flashen van de hardware, gebeurt via long-lived connections.

3.2.5 Eindresultaat implementatie voor verbetering workflow

In vorige secties werden verschillende compilers en technologieën onderzocht en geïmplementeerd om tot een finale oplossing te komen. Deze oplossing bestaat uit het gebruik van de command-line tool: arduino-builder, die de mogelijkheid biedt om sketches te compileren. Voor deze tool is er een Node.js-server gebouwd waardoor code naar deze server kan gezonden worden, de server zal achterliggend de code compileren en het resulterende HEX-file terugsturen. Nu het mogelijk is om de code te compileren, werd via een Google Chrome extensie de functionaliteit gecreëerd om hardware te flashen met AVRGIRL. De finale architectuur voor de optimalisatie van de workflow ziet er als volgt uit (figuur 3.25).



Figuur 3.25: Finale architectuur optimalisatie workflow

Via het Blockly4Arduino-platform is het nu mogelijk om de gemaakte code via de blokken te compileren. Hiervoor moet de gebruiker dus geen andere software installeren of een

andere tool gebruiken. De compilatie geeft de code terug in machinetaal (HEX) waardoor de Arduino boards geprogrammeerd kunnen worden. Deze laatste stap maakt gebruik van een extensie om de hardware te flashen waardoor er toch van de gebruiker vereist is om een Chrome extensie te installeren. Met 1 druk op de knop, wordt de hardware nu geprogrammeerd vanuit het Blockly4Arduino-platform (figuur 3.26).



Figuur 3.26: Blockly4Arduino met toegevoegde functionaliteit om code te verifiëren en te uploaden via de extensie

Hoofdstuk 4

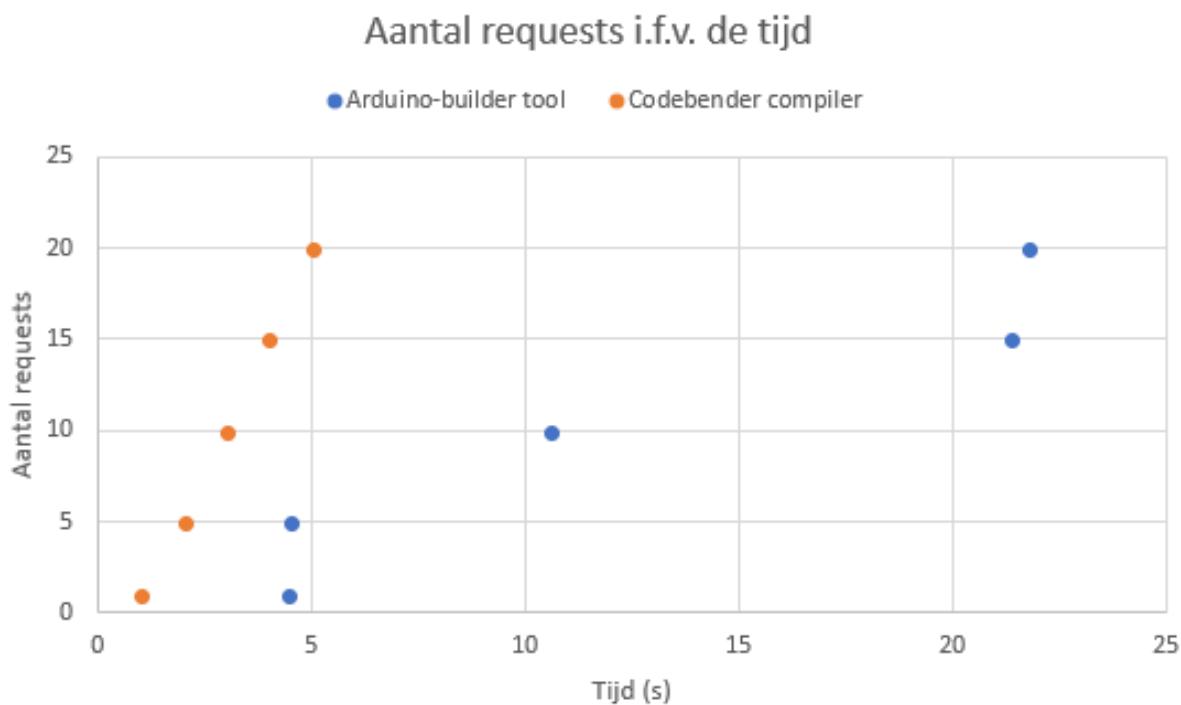
Evaluatie

Dit hoofdstuk zal enerzijds een vergelijking maken tussen de opgezette compilers qua capaciteit en snelheid. Anderzijds zal de feedback over de toegevoegde functionaliteit geanalyseerd worden. Deze feedback is bekomen door een groep studenten een vragenlijst te laten invullen tijdens een workshop van Ingegno.

4.1 Verschillen tussen de uitgewerkte oplossingen

In deze sectie worden de uitgewerkte compilers met elkaar vergeleken in functie van capaciteit (aantal requests) en snelheid.

Een eerste test bestaat uit een script dat een aantal requests naar de compilers zal sturen. Hierbij wordt gekeken hoelang het duurt om alle requests terug af te handelen. De resultaten zijn bekomen door iedere meting 5 maal uit te voeren en het gemiddelde te nemen (zie figuur 4.1). Op de figuur is te zien dat Codebender compiler-server sneller de requests zal afhandelen. Ook is af te leiden dat er qua compilatie-tijd geen verschil is tussen 1 of 5 requests bij de arduino-builder server om wille van multithreading. Echter bij het verhogen van het aantal requests is duidelijk te zien dat de executietijd van de arduino-builder server in sprongen verhoogt. Wel is op te merken dat arduino-builder meerdere stappen volgt waarbij bestanden gecreëerd worden wat tijd in beslag neemt, Codebender compiler werkt zonder tussenstappen wat de compilatie-tijd aanzienlijk versnelt. Een tweede opmerking is dat de Codebender compiler enkel actief is en zonder de builder-component. De arduino-builder server is wel in staat om volledige projecten te verwerken waardoor er extra stappen en tijd nodig zijn.



Figuur 4.1: Aantal requests in functie van de tijd voor de opgezette compilers: Arduino-builder en Codebender Compiler.

Een tweede test bestond uit het versturen van grote hoeveelheden requests (e.g. 500) over een korte tijdsperiode en nagaan of de compilers dit konden verwerken. Voor beide compilers was dit het geval maar de Codebender compiler-server deed er aanzienlijk minder lang over (zie ook hierboven). Het is moeilijk in te schatten of de PHP vs. Node.js-servers hier mee te maken hebben want de manier waarop de compiler zelf werkt is fundamenteel anders waardoor er grote verschillen zijn qua tijd. Om echt de snelheid tussen PHP en Node.js te testen zou het arduino-builder project moeten omgevormd worden naar een PHP-server.

Onderstaande tabel zet de voor -en nadelen van beide compilers nog eens op een rij (tabel 4.1). De verschillen tussen Node.js en Symfony zijn in tabel 3.6 besproken en uitgezet.

4.2 Evaluatie toegevoegde blokken

Naast het testen van de compilers is er ook onderzoek gedaan of de toegevoegde blokken nuttig waren in gebruik en duidelijk van layout. Hiervoor werden 2 enquêtes opgesteld

Tabel 4.1: Codebender compiler-server vs. arduino-builder server

<i>Opties</i>	<i>arduino-builder (Node.js)</i>	<i>Codebender compiler (Symfony)</i>
Eigenschappen	JavaScript; Easy to learn; Lightweight; Arduino-builder tool kan volledige projecten builden; Kan veel requests verwerken.	PHP; Codebender compiler is sneller (zonder Codebender builder getest); Kan veel requests verwerken.

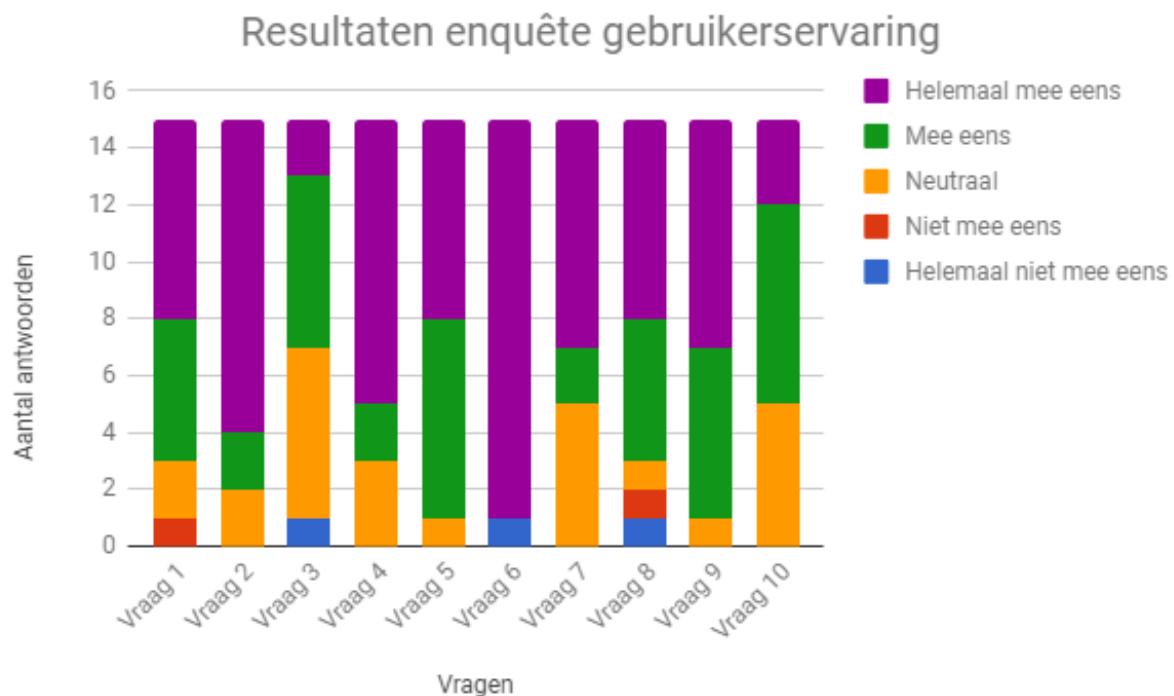
(zie Appendix A, B) die peilden naar de opinie van de Blockly4Arduino-gebruikers. De gestelde vragen handelden over het gebruik van het Blockly4Arduino-platform zelf tot het gebruik van de nieuwe blokken.

De vragenlijsten zijn gegeven tijdens Ingegno-workshops waar een twee groepen van 8 en 7 kinderen hebben aan deelgenomen. De leeftijden van deze kinderen varieerde tussen 8 en 14 jaar. Hieronder zijn de resultaten weergegeven op de enquête in verband met de gebruikerservaring (figuur 4.2). De enquête in verband met de toegevoegde blokken is nog niet gevraagd omdat er nog geen workshops hebben plaatsgevonden die gebruik maakten van de nieuwe blokken.

Verdere opmerkingen door de gebruikers:

1. Scratch vind ik gemakkelijker;
2. Ik vind Arduino tof;
3. Ik wil meer toffe projecten maken.

Uit de resultaten zijn er een aantal zaken af te leiden. De meeste gebruikers antwoorden dat ze het Blockly4Arduino-platform eenvoudig vonden om mee te werken maar dat ze wel een aantal dingen moesten leren vooraleer ze met het platform aan de slag konden gaan (vragen 1-2). Een tweede zaak is de nood aan extra software om de code te compileren en te uploaden. Hieromtrent zijn de gebruikers het eensgezind dat het toffer zou zijn indien die extra software niet meer nodig was (vragen 3-4). Een derde onderwerp ging over het kopiëren van de gemaakte code naar de extra software en hoe eenvoudig ze dit vonden (vragen 5-6). Er werd namelijk een extra knop toegevoegd die automatisch de gemaakte code naar het klembord van de gebruiker kopieerde zodat er geen fouten meer gemaakt



Figuur 4.2: Resultaten enquête over de gebruikservaring, afgenomen door 15 gebruikers tijdens workshops van Ingegno. De vragen zelf zijn te vinden in Appendix B.

konden worden. Hierop werd alweer positief geantwoord. Een vierde onderwerp ging over het opslaan van gemaakte programma's en het uploaden van gesavede programma's naar het platform (vragen 7-8). Hierbij werd duidelijk dat het opslaan geen probleem vormt maar het inladen van een opgeslagen programma voor sommige niet zo eenvoudig is. Dit is een puntje waarmee nog rekening kan gehouden worden naar de toekomst toe. Het laatste onderwerp handelde over hoe duidelijk de blokken hun functie beschreven en in welke mate de gebruikers in staat waren om de tutorials zelfstandig te volgen (vragen 9-10). De blokken werden algemeen als duidelijk beschouwd maar bij het zelfstandig volgen van de tutorials stelden de meeste een neutrale houding op. Na het meevolgen van een workshop is wel op te merken dat de kinderen snel afgeleid zijn en ze te snel naar de volgende stap gaan zonder alle voorgaande stappen afgewerkt te hebben. Bij het zelfstandig kunnen volgen van een tutorial speelt de leeftijd waarschijnlijk de grootste rol waarom ze nog niet allemaal zonder hulp een stappenplan kunnen volgen.

Hoofdstuk 5

Besluit en toekomstperspectieven

In de literatuurscriptie werd onderzocht hoe het leerproces van techniek en technologie kan geoptimaliseerd worden voor het Blockly4Arduino platform. Hierbij werd vastgesteld dat de stappen die ondernomen moeten worden om een programma te schrijven, kunnen worden ingekort of geautomatiseerd. De stap om de code te compileren en vervolgens te uploaden naar het Arduino board kan namelijk volledig geautomatiseerd worden. Daarvoor werd een studie gedaan naar mogelijke technologieën en beschikbare compilers. Het resultaat was dat de technologie van Codebender uitermate geschikt is om te integreren in het Blockly4Arduino platform. De open-source code die beschikbaar is maakt de technologie ideaal om te integreren (een verduidelijkend schema van de structuur met uitleg is voorzien bij figuur 2.4). Eenmaal de compiler en plugin functioneel zijn, hoeven ze enkel nog gekoppeld te worden met het Blockly4Arduino platform.

Een tweede onderzoeksvraag handelde over het uitbreiden van de functionaliteit van het platform. Hiervoor is er onderzoek verricht naar bestaande gelijkaardige platformen om gelijkenissen, verschillen, nieuwe en betere functies te vinden. Vertrekkend van deze resultaten (zie sectie 2.3), kunnen er sneller nieuwe blokken en structuren worden toegevoegd die relevant zijn voor het platform en het aanleren van programmeren, technologie en probleem-oplossend denken.

De derde onderzoeksvraag ging over het optimaliseren van tutorials en hints. De besluiten rond dit stuk geven aan wat het beste werkt. Zo vermijdt men best het overvloedig gebruik van tekst bij het aanbieden van hints en blijken figuren effectiever te zijn. Op vlak van tutorials zijn er Blockly varianten die met verschillende moeilijkheidsgraden werken en hierbij een visuele output voorzien om de gebruiker te helpen. Het implementeren van

moeilijkheidsgraden is een stap in de goede richting om oefeningen aan te bieden voor meerdere leeftijden en doelgroepen.

In de scriptie is reeds aangehaald waarom blok-gebaseerd programmeren zijn voordelen heeft voor beginnende programmeurs. Naar de toekomst toe is het ook belangrijk om de overstap van blok-gebaseerd naar tekst-gebaseerd programmeren te integreren. Een goede manier van werken is door dezelfde oefening op beide manieren te vragen, eerst blok-gebaseerd zodat de gebruiker het algoritme onder de knie heeft, daarna tekst-gebaseerd zodat er kan gefocust worden op het schrijven van de coderegels en syntax.

Tijdens het hoofdstuk van de implementatie werden er 2 zaken uitgewerkt: het uitbreiden van het assortiment aan blokken en het uitbreiden van de functionaliteit van het platform door het integreren van een compiler.

Het assortiment aan blokken werd uitgebreid met blokken voor een drukknop met interne weerstand, een 7-segment display aan te sturen, de huidige waarde van een supersonische sensor verkrijgen en om tekst naar een OLED display te schrijven. Deze blokken werden geselecteerd om toe te voegen na onderzoek van andere platformen en hun assortiment aan blokken. Naast het toevoegen van blokken, werd nog een extra knop toegevoegd aan de werkbalk waarmee gebruikers sneller en zonder fouten de gemaakte code konden kopiëren naar het klembord.

Bij het uitbreiden van de functionaliteit werd de focus gelegd op het koppelen van het Blockly4Arduino-platform met een code-compiler. Mits de beschikbare compilers werken via te installeren software of een betalende webpagina, werd geopteerd om zelf een compiler op te zetten. Hierbij werd eerst een opstelling uitgewerkt met behulp van open-source projecten van Codebender. Deze opstelling bleek niet volledig functioneel te zijn om wille van libraries die moeilijk tot niet werden meegegeven aan de compiler. Vervolgens werd een tweede opstelling gemaakt met de open-source projecten van Arduino zelf. De Arduino software is zodanig opgebouwd dat deze achterliggend gebruikt van een command-line tool om projecten te builden en sketches te compileren. Hierbij was het mogelijk om de compiler zelf aan te sturen en een server te bouwen die achterliggend deze tool aanspreekt. De opgezette server is gebouwd in Node.js en laat nu toe om vanaf een webpagina code te sturen via HTTP POST-berichten en de gecompileerde code terug te krijgen in hex-formaat. Eenmaal de compiler opgezet was, was er echter nog 1 probleem en dat was het programmeren van de Arduino boards vanuit de browser. Om deze stap uit te werken, is gebruik gemaakt van een Google Chrome extensie die wel de toegang kan krijgen tot de

seriële poorten van de gebruikers. Achterliggend maakt de extensie gebruik van AVR GIRL, een JavaScript package, om de Arduino boards te kunnen programmeren volgens bepaalde communicatie-protocollen.

Bij de evaluatie werden 2 zaken onderzocht: de snelheid en load van de opgezette compilers en de verwerking van een enquête omtrent gebruikerservaring bij het Blockly4Arduino-platform. Uit de enquête bleek dat de gebruikers het platform eenvoudig vinden om mee te werken, de blokken duidelijk hun werking of functie omschrijven en het toffer zou zijn indien er geen externe software of compiler nodig is. Een onderwerp waar de gebruikers het iets minder over eens waren is het opslaan van hun code en het uploaden van opgeslagen code, dit bleek niet zo gemakkelijk te zien en zou verder kunnen onderzocht worden in Future work.

Future work

Naar de toekomst toe kunnen er nog enkele zaken onderzocht of verder uitgebouwd worden. Een eerste uitbreiding kan gemaakt worden bij de compiler gebouwd met de arduino-tool. De werking van de arduino-tool is zodanig opgebouwd dat verschillende stappen doorlopen worden die allemaal output-files genereren. Deze bestanden worden naar een temporele map geschreven waarbij de server deze bestanden terug wist eenmaal de .hex-file bekomen wordt. Per request worden dus enkele bestanden aangemaakt en gewist wat er voor kan zorgen dat de harde schijf sneller zal verslijten. Ook al gaat het maar om enkele kilobytes, het effect op lange termijn zou verder onderzocht kunnen worden.

Bijlage A

Vragenlijst 7-segment component

Vragenlijst omtrent 7-segment blok

De volgende vragen gaan rond uw gebruikservaring met het Blockly4Arduino-platform. Gelieve in te vullen in welke mate u akkoord gaat met onderstaande uitspraken op een schaal van "Helemaal niet mee eens" tot "Helemaal mee eens".

- | | Niet
mee eens | Mee
eens |
|---|--|-------------|
| 1. Ik vind de beschikbare blokken om een 7-segment display aan te sturen duidelijk. | <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="3"/> <input type="button" value="4"/> <input type="button" value="5"/> | |
| 2. Ik kan zonder hulp een 7-segment display aansturen. | <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="3"/> <input type="button" value="4"/> <input type="button" value="5"/> | |
| 3. Het programma bouwen voor zo'n display aan te sturen kan heel snel dankzij de beschikbare blokken. | <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="3"/> <input type="button" value="4"/> <input type="button" value="5"/> | |

Via de led-blokken is het ook mogelijk om een 7-segment display aan te sturen.

- | | Niet
mee eens | Mee
eens |
|--|--|-------------|
| 4. Via de led-blokken slaag ik er ook in om het 7-segment display aan te sturen. | <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="3"/> <input type="button" value="4"/> <input type="button" value="5"/> | |
| 5. Met de led-blokken kan ik het programma sneller bouwen dan met de 7-segment blokken. | <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="3"/> <input type="button" value="4"/> <input type="button" value="5"/> | |
| 6. Ik vind de led-blokken eenvoudiger om het programma te bouwen dan met de 7-segment blokken. | <input type="button" value="1"/> <input type="button" value="2"/> <input type="button" value="3"/> <input type="button" value="4"/> <input type="button" value="5"/> | |

Verdere opmerkingen of verbeteringen:

Bijlage B

Vragenlijst Gebruikserving

Vragenlijst omtrent gebruikservaring

De volgende vragen gaan rond uw gebruikservaring met het Blockly4Arduino-platform. Gelieve in te vullen in welke mate u akkoord gaat met onderstaande uitspraken op een schaal van "Helemaal niet mee eens" tot "Helemaal mee eens".

	Niet mee eens	Mee eens
1. Ik vind de Blockly4Arduino-website eenvoudig om mee te werken.	1 2 3 4 5	
2. Ik moest eerst een aantal dingen leren voordat ik met het platform aan de slag kon.	1 2 3 4 5	
3. Ik vind de extra software(Arduino IDE/Codebender) om de hardware te programmeren gemakkelijk.	1 2 3 4 5	
4. Ik zou het tof vinden als de extra software niet meer nodig is om de hardware te programmeren.	1 2 3 4 5	
5. Code kopiëren van de website naar de Arduino IDE/Codebender is simpel.	1 2 3 4 5	
6. Ik maak veel gebruik van de groene knop "Code kopiëren".	1 2 3 4 5	
7. Gemaakte programma's opslaan is gemakkelijk.	1 2 3 4 5	
8. Opgeslagen programma's uploaden gebeurt snel en eenvoudig.	1 2 3 4 5	
9. De blokken zijn handig en duidelijk om een programma te bouwen.	1 2 3 4 5	
10. De beschikbare voorbeelden en tutorials zijn duidelijk. Ik kan zonder hulp deze tutorials volgen en maken.	1 2 3 4 5	

Verdere opmerkingen of verbeteringen:

Bijlage C

Handleiding voor het opzetten van de Compiler server en Chrome extensie

Compile server

Project to serve back-end of Blockly4Arduino.

Compile your sketches or build your project and return .hex-file of requested code.

This project uses the arduino-builder tool used by the Arduino IDE.

Requirements

- Linux or Ubuntu OS (Host or VM);
- Node.js installed (<https://nodejs.org/en/download/>);
- npm installed (if not installed with Node.js);
- Latest version of the Arduino IDE installed (<https://www.arduino.cc/en/Main/Software>).

Setting up the server

- Download or clone following GitHub-directory: <https://github.com/RMeurisse/ArduinoBuilder>;
- `cd` to downloaded directory;
- Run `(sudo) npm install` in the current directory. This will install all dependencies specified in the manifest.json-file;
- Change the file ‘config.js’ to specify your local paths and variables:
 - Local ‘./arduino’-directory: this directory will contain the extra libraries you downloaded with the Arduino IDE;
 - Local ‘./arduino-1.8.5’-directory: this directory contains all the files used by the arduino-builder tool;
 - Local ‘temporary’-directory: this is the directory where the arduino-builder tool will store its temporary files, this can be a directory you created or you can use the ‘/tmp’-directory of Linux/Ubuntu.
- Run `(sudo) node server.js`;
- Go to: `http://localhost:{Port}/`. With {Port} being the port specified in the ‘config.js’-file;
- Write your code in the textbox and verify your code or upload it directly to a connected Arduino board;
- (To automatically start the server at start-up, see next section).

Automatic start server at start-up of the system

- In the downloaded directory, find the file ‘blocklyserver.service’;
- Edit the following line in the ‘blocklyserver.service’-file to refer to the location of the server.js-file:

```
ExecStart=[PATH TO SERVER.JS FILE]
```

- This service-file will execute the server at startup or restart the server automatically if it crashes;
- Before continuing: make sure the first line of the server.js-file contains the following: `#!/usr/bin/env node`. This command will make sure the system knows that file needs to be executed with node. Also make sure the server.js-file is executable by doing the following: `(sudo) chmod +x [PATH TO SERVER.JS FILE]`;
- Copy or move this file to the startup-directory. For linux: `(sudo) cp [PATH TO SERVICE FILE] /etc/systemd/system/`;
- Enable the service: `systemctl enable blocklyserver.service`;
- Start the service: `systemctl start blocklyserver.service`;
- Verify it's running: `systemctl status blocklyserver.service`;
- If there are problems, you can check the errors in the log: `less /var/log/syslog` or messages with `journalctl -u blocklyserver.service`.

Optional: The above steps will run the service as root. If you want to run the server from a specific user only, follow the steps below;

- Change the following lines to specify the user:

```
User=[USERNAME]
```

```
Group=[YOUR GROUP]
```

- After making changes to the service file, reload it: `systemctl daemon-reload`;
- Restart the service: `systemctl restart blocklyserver.service`;
- Check status with: `systemctl status blocklyserver.service`.

Development

If you want to integrate the given index.html page into your own website.

POST request to server

Your website should send a HTTP POST-request with JSON-data of the following format:

```
{
  'data': 'void setup(){} void loop(){}',
```

```
'boardName': 'arduino:avr:uno'  
}
```

The 'data'-property should contain your program (text written in the textbox) and the 'boardName'-property should contain the name of the board you want to compile code for (see 'Accepted board-types' below for the name).

Response from the server

When sumbitting a request, the server will return a response message containing JSON-data of the following format:

```
{  
  'hex': '...',  
  'out': '...',  
  'err': '...'  
}
```

The 'hex'-property contains the compiled program in hex-format, ready to be uploaded onto the board.

The 'out'-property contains all the console messages of the compiler while the 'err'-property contains the error messages if there was an error, otherwise this will be empty.

Accepted board-types/names

The following table displays the board string parameters for the POST-request to the server. The last column is the name for the board that the extension requires. The board string is different for the builder from the extension.

Only the standard board strings are given for the arduino-builder. If you want to add other boards, use the Arduino IDE to include them. When you verify a program, the console outputs the commands used, search for the option `-fqbn [arduino:avr:....]`. This option is the one you will have to include in your website.

Programmer	Arduino-builder Board String	Extension Board String
Arduino Uno	arduino:avr:uno	uno
Arduino Mega 1280	arduino:avr:mega:cpu=atmega1280	mega
Arduino Mega 2560	arduino:avr:mega:cpu=atmega2560	mega
Arduino ADK	arduino:avr:megaAdk	adk

Programmer	Arduino-builder Board String	Extension Board String
Arduino Leonardo	arduino:avr:leonardo	leonardo
Arduino Micro	arduino:avr:micro	micro
Arduino Nano 328	arduino:avr:nano:cpu=atmega328	nano
Arduino Lilypad USB	arduino:avr:lilypad-usb	lilypad-usb
Arduino Duemillanove	arduino:avr:diecimila:cpu=atmega168	duemilanove168
Arduino Yun	arduino:avr:yun	yun
Arduino Esplora	arduino:avr:esplora	esplora
RedBearLab Blend Micro	``	blend-micro
Tiny Circuits Tinyduino	``	tinyduino
SparkFun Pro Micro	arduino:avr:pro:cpu=16MHzatmega328	sf-pro-micro
Qtechknow Qduino	``	qduino
Pinoccio Scout	``	pinoccio
Femtoduino IMUduino	``	imuduino
Adafruit Feather 32u4 Basic Proto	``	feather
Arduboy	``	arduboy
Adafruit Circuit Playground	arduino:avr:circuitplay32u4cat	circuit-playground-classic

Functions used by the webpage

The webpage uses several functions to compile the given code and flash the hardware through the extension.

```
/**  
 * Function checkExtension will poll if the extension is installed or not.  
 * The function uses short-lived connections with the extension to do this (chrome.runtime.sendMessage()).  
 * Steps:  
 * 1. Send message to extension with option 'check';  
 * 2. Extension will respond if installed, otherwise chrome.runtime.sendMessage() will re
```

```
turn false.  
*/  
function checkExtension()
```

```
/**  
 * Function populatePorts will add connected devices to the select box.  
 * Steps:  
 * 1. Send message to extension with option 'ports';  
 * 2. Extension will retrieve all ports that have hardware connected;  
 * 3. Extension sends message back containing JSON-data with the list of ports;  
 * 4. This function will populate the drop-down list with the returned ports.  
 */  
function populatePorts()
```

```
/**  
 * Function handleSubmit() will load a hex-file from the local PC  
 * and post it to the extension to flash the device.  
 * Steps:  
 * 1. Open filesystem so the user can select a .hex-file;  
 * 2. Make JSON-Object containing the contents of the selected file, the Arduino board na  
me and the selected port;  
 * 3. Send the object to the extension with a port-object (port.postMessage() is a long-l  
ived connection because it can take some time);  
 * 4. Extension will respond if the flashing was succesful.  
 */  
function handleSubmit()
```

```
/**  
 * Function verifytxt() will return whether or not the code is compilable.  
 * This function will NOT flash the device.  
 * Steps:  
 * 1. Makes a HTTP POST-request to the specified compiler with the code from the textbox;  
 * 2. Compiler sends a message back with success or error;  
 * 3. Display the returned error.  
 */  
function verifytxt()
```

```
/**  
 * Function uploadtxt() will flash the code in the textbox to the connected device  
 * Steps:  
 * 1. HTTP POST-request to compile the code and to receive the hex-file;  
 * 2. Send received hex-file to the extension;  
 * 3. Extension will flash a connected device;  
 * 4. Extension will return success if flashing was succesful.  
 */  
function uploadtxt()
```

Blockly4Arduino Chrome Extension

Extension will use AVR-GIRL-Arduino to flash hex-files to Arduino boards from the Chrome-browser.

Requirements

- Google Chrome browser is installed;
- Install the following tool globally: `npm install -g browserify`.

Development

To develop this further or make changes:

1. Edit `background.js` (not `background.bundle.js`);
2. Always run `browserify background.js -o background.bundle.js` each time you change `background.js` or `lib/flash.js`, and then manually reload the chrome app;
3. When you reload the chrome app, you'll also need to refresh `index.html` so the page can reconnect to the reloaded chrome app.

Run the extension with the Compiler server of Blockly4Arduino

To run the demo with the Compiler server repository:

1. Fork and clone or download this repository (<https://github.com/RMeurisse/Arduino-Builder-chrome-extension>);
2. Run `npm install` within the newly cloned local repo;
3. Run `browserify background.js -o background.bundle.js`;
4. In your Chrome browser, visit `chrome://extensions`;
5. Click `Load unpacked extension`;
6. Navigate to this cloned repo, and click `Select`;
7. Click `Launch` when you see the extension appear at the top of the extensions list;
8. Copy the `id` value of the extension, and update `var extensionid = '...';` line in `index.html` of the Arduino-Builder repo;
9. See the Arduino-Builder repo to run the nodejs-server;
10. Plug in an Arduino, and write some code in the textbox. Finally hit the 'upload'-button.

Bibliografie

- [1] Deepak Kumar. Digital playgrounds for early computing education. *ACM Inroads*, 5(1):20–21, March 2014.
- [2] Kalelioglu, F., GĂ¶lbahar, Y., *The effects of teaching programming via scratch on problem solving skills: A discussion from learners' perspective*, Informatics in Education, pp. 33-50, Vilnius, Lithuania, Jan. 2014.
- [3] Davis, M., *Computer coding lessons expanding for K-12 students*, pp. 28,30–31, Texas, USA, 2013.
- [4] Wen-Chung Shih. Mining Learners' Behavioral Sequential Patterns in a Blockly Visual Programming Educational Game. In *2017 INTERNATIONAL CONFERENCE ON INDUSTRIAL ENGINEERING, MANAGEMENT SCIENCE AND APPLICATION (ICIMSA 2017)*, pages 77–78. IEEE; IEEE SEOUL SECT; iCaste; Korea Ind Secur Forum; KGU; Univ Sci & Technol Beijing; Seoul Metropolitan Govt; King Mongkuts Univ Technol Thonburi; KCSA; River Publishers, 2017. International Conference on Industrial Engineering, Management Science and Application (ICIMSA), Seoul, SOUTH KOREA, JUN 13-15, 2017.
- [5] Cesar Vandevelde, Jelle Saldien, Maria-Cristina Ciocci, and Bram Vanderborght. Overview of technologies for building robots in the classroom. In *International Conference on Robotics in Education, Proceedings*, pages 122–130, 2013.
- [6] Francis wyffels, Bern Martens, and Stefan Lemmens. Starting from scratch: experimenting with computer science in flemish secondary education. In Carsten Schulte, Michael E. Caspersen, and Judith Gal-Ezer, editors, *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, pages 12–15. ACM, 2014.
- [7] Giordano, D., Maiorana, F., *Teaching algorithms: Visual language vs flowchart vs textual language*, 2015 IEEE Global Engineering Education Conference (EDUCON), pp. 499-504, Tallinn, Estonia, Mrt. 2015.

- [8] Tamilias, A., Karvounidis, T., Garofalaki, T., Kallergis, Z., *Blocks for Arduino in the Students Educational Process*, 2017 IEEE Global Engineering Education Conference (EDUCON), pp. 910–915, Athens, Greece, Apr. 2017.
- [9] Ferrari, A., Poggi, A., Tomaiuolo, M., *Object oriented puzzle programming*, Parma, Italy, Jun. 2016.
- [10] Culic, I., Radovici, A., Vasilescu, L., *Auto-generating Google Blockly Visual Programming Elements for Peripherla Hardware*, 2015 14th Roedunet International Conference - Networking in Education and Research (RoEduNet NER), pp. 94–98, Craiova, Romania, Sep. 2015.
- [11] Tyng-Yeu Liang, Hao-Ting Peng, and Hung-Fu Li. A Block-Oriented C Programming Environment. In *PROCEEDINGS OF 2016 INTERNATIONAL CONFERENCE ON APPLIED SYSTEM INNOVATION (ICASI)*. Taiwanese Inst Knowledge Innovat; Inst Elect & Elect Engineers; Sci Educ Ctr; Natl Formosa Univ; Chia Nan Univ Pharmacy & Sci; Kun Shan Univ; S Taiwan Univ Sci & Technol; St. Johns Univ; Natl United Univ; Univ Ryukyus; Kongju Natl Univ; Hannam Univ; Jimei Univ; iVAN Technol Enterprise Grp; IEEE Tainan Sect; Fujian Informat Ind Assoc; Fuzhou Cross Strait Ind Design Creat Park, 2016. International Conference on Applied System Innovation (IEEE ICASI), Fuzhou Univ, Okinawa, JAPAN, MAY 28-JUN 01, 2016.
- [12] Adiel Ashrov, Assaf Marron, Gera Weiss, and Guy Wiener. A use-case for behavioral programming: An architecture in Java Script and Blockly for interactive applications with cross-cutting scenarios. *SCIENCE OF COMPUTER PROGRAMMING*, 98(2, SI):268–292, FEB 1 2015.
- [13] V. Amaxilatis, D. Georgitzikis. *Using Codebender and Arduino in Science and Education*, pages 119–134. Springer International Publishing, Cham, 2014.
- [14] Erik Pasternak, Rachel Fenichel, and Andrew N. Marshall. Tips for Creating a Block Language with Blockly. In Turbak, F and Gray, J and Kelleher, C and Sherman, M, editor, *2017 IEEE BLOCKS AND BEYOND WORKSHOP (B&B)*, pages 21–24. IEEE, 2017. IEEE Blocks and Beyond Workshop (B&B), Raleigh, NC, OCT 09-10, 2017.
- [15] Fraser, N., *Ten things we learned from Blockly*, 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond), pp. 49–50, Atlanta, USA, Oct. 2015.

- [16] Irene Sheridan, Deirdre Goggin, and Linda O'Sullivan. EXPLORATION OF LEARNING GAINED THROUGH CODERDOJO CODING ACTIVITIES. In Chova, LG and Martinez, AL and Torres, IC, editor, *INTED2016: 10TH INTERNATIONAL TECHNOLOGY, EDUCATION AND DEVELOPMENT CONFERENCE*, INTED Proceedings, pages 6541–6548, 2016. 10th International Technology, Education and Development Conference (INTED), Valencia, SPAIN, MAR 07-09, 2016.
- [17] Federico Garcia. Cactus: automated tutorial course generation for software applications. 2000.
- [18] Harms, J., Gray, S., Cosgrove, D., Kelleher, C., *Automatically generating tutorials to enable middle school children to learn programming independently*, ACM International Conference Proceeding Series, pp. 1–4, Hudson, USA, Jun. 2013.
- [19] Jack Purdum. Beginning C for Arduino, Second Edition. pages 0–388. Apress, 2015.
- [20] Badamasi, Y., *The working principles of an Arduino*, 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), pp. 1–4, Abuja, Nigeria, 2014.
- [21] Lakshmi Prasanna Chitra and Ravikanth Satapathy. Performance Comparison and Evaluation Of Node.Js And Traditional Web Server (IIS). In *2017 INTERNATIONAL CONFERENCE ON ALGORITHMS, METHODOLOGY, MODELS AND APPLICATIONS IN EMERGING TECHNOLOGIES (ICAMMAET)*, 2017. International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), Chennai, INDIA, FEB 16-18, 2017.

Lijst van figuren

2.1	Google Blockly als voorbeeld van een blokgebaseerde programmeeromgeving die achterliggend code genereert in een specifieke taal (hier JavaScript).	6
2.2	Blockly4Arduino: blokgebaseerde programmeeromgeving die achterliggend Arduino code genereert.	8
2.3	Codebender.cc: Compiler structuur waarbij de gebruiker code kan maken op het platform en via een extensie deze compileren en uploaden naar een aangesloten Arduino board.	12
2.4	Compiler structuur aangepast voor Blockly4Arduino. Ofwel op het domein van Blockly4Arduino hosten ofwel gebruik maken van de diensten van Codebender.	12
2.5	Blockly4Arduino: Nodige blokken uitlichten uit het assortiment.	16
2.6	Ozoblockly Games.	16
2.7	Blockly Game Maze, blokgebaseerde oefening.	17
2.8	Blockly Game Pond, tekstgebaseerde oefening.	17
2.9	Voorbeeld van simulatie venster voor oefening: blink led met Arduino.	19
3.1	Toevoeging van paars initialisatie-blok om externe drukknop met interne pull-up weerstand te gebruiken.	24
3.2	7-Segment component met aanduiding van de verschillende segmenten en punt.	24
3.3	7-segment centrale blok om alle segmenten aan een digitale pin te koppelen.	25
3.4	7-segment blok dat achterliggend de code genereert om het meegegeven cijfer te tonen.	25
3.5	7-segment blok dat achterliggend een enkel segment zal aansturen.	26
3.6	OLED blok om de resolutie van het aan te sturen display in te stellen in de code.	27

3.7	OLED blok om de font-grootte in te stellen van de tekst die op het display zal verschijnen.	28
3.8	OLED blok om vanaf een bepaalde positie de meegegeven waarde op het display te tonen.	28
3.9	OLED blok om de ingestelde waarden te schrijven naar het display.	28
3.10	Formule om de afstand te berekenen aan de hand van de tijd en de snelheid van geluidsgolven.	29
3.11	Timing diagram van de Trigger -en Echopinnen	30
3.12	Sensor blok om een supersonische sensor te configureren, stelt de Echo -en Trigger-pinnen in alsook de maximale wachttijd.	30
3.13	Sensor blok om de afstand te retourneren van de aangegeven sonische sensor. .	31
3.14	Werkbalk van het Blockly4Arduino-platform met toegevoegde kopieer-knop. .	31
3.15	Codebender-platform: User-interface om programma's te schrijven, te verifiëren en te uploaden.	33
3.16	Codebender.cc architectuur	34
3.17	Codebender: Compiler-structuur na integratie of Compiler op het domein van Codebender gebruiken.	38
3.18	Codebender: Architectuur van de compiler. De compiler ontvangt HTTP-requests en zal de gekregen data compileren om uiteindelijk data in HEX-formaat terug te sturen.	39
3.19	Codebender: POST-request naar de Compiler met meegegeven data in JSON-formaat.	40
3.20	Architectuur Codebender componenten	41
3.21	Arduino-builder: Stappenplan van het compileer-proces met de avr gcc-compiler.	45
3.22	Aansturing arduino-builder tool via Bash	50
3.23	Stappenplan werking Node.js-Server, van request tot response	53
3.24	Architectuur Chrome extensie. Een content-script van de webpagina is in staat om te communiceren met het background-script van de extensie. Dit background-script kan op zijn beurt de API's aanspreken die beschikbaar zijn in de browser.	58
3.25	Finale architectuur optimalisatie workflow	59
3.26	Blockly4Arduino met toegevoegde functionaliteit om code te verifiëren en te uploaden via de extensie	60

4.1	Aantal requests in functie van de tijd voor de opgezette compilers: Arduino-builder en Codebender Compiler.	62
4.2	Resultaten enquête over de gebruikservaring, afgenomen door 15 gebruikers tijdens workshops van Ingegno. De vragen zelf zijn te vinden in Appendix B.	64

Lijst van tabellen

2.1	Vergelijking tussen mogelijke compilers op basis van een aantal karakteristieken.	11
2.2	Blockly Varianten	21
3.1	Schematisch overzicht van de mogelijkheden met het compilerflasher-script met bijhorende voor -en nadelen.	35
3.2	Voor -en nadelen bij het gebruik van de Compiler van Codebender.	37
3.3	Voor -en nadelen bij het gebruik van Builder-server van Codebender.	42
3.4	Voor -en nadelen bij het gebruik van de arduino-builder tool.	47
3.5	Hoofd modules met bijhorende rolverdeling	48
3.6	Node.js vs Symfony	51
3.7	Voor -en nadelen van de AVRGIRL tool om Arduino boards te flashen.	56
4.1	Codebender compiler-server vs. arduino-builder server	63

