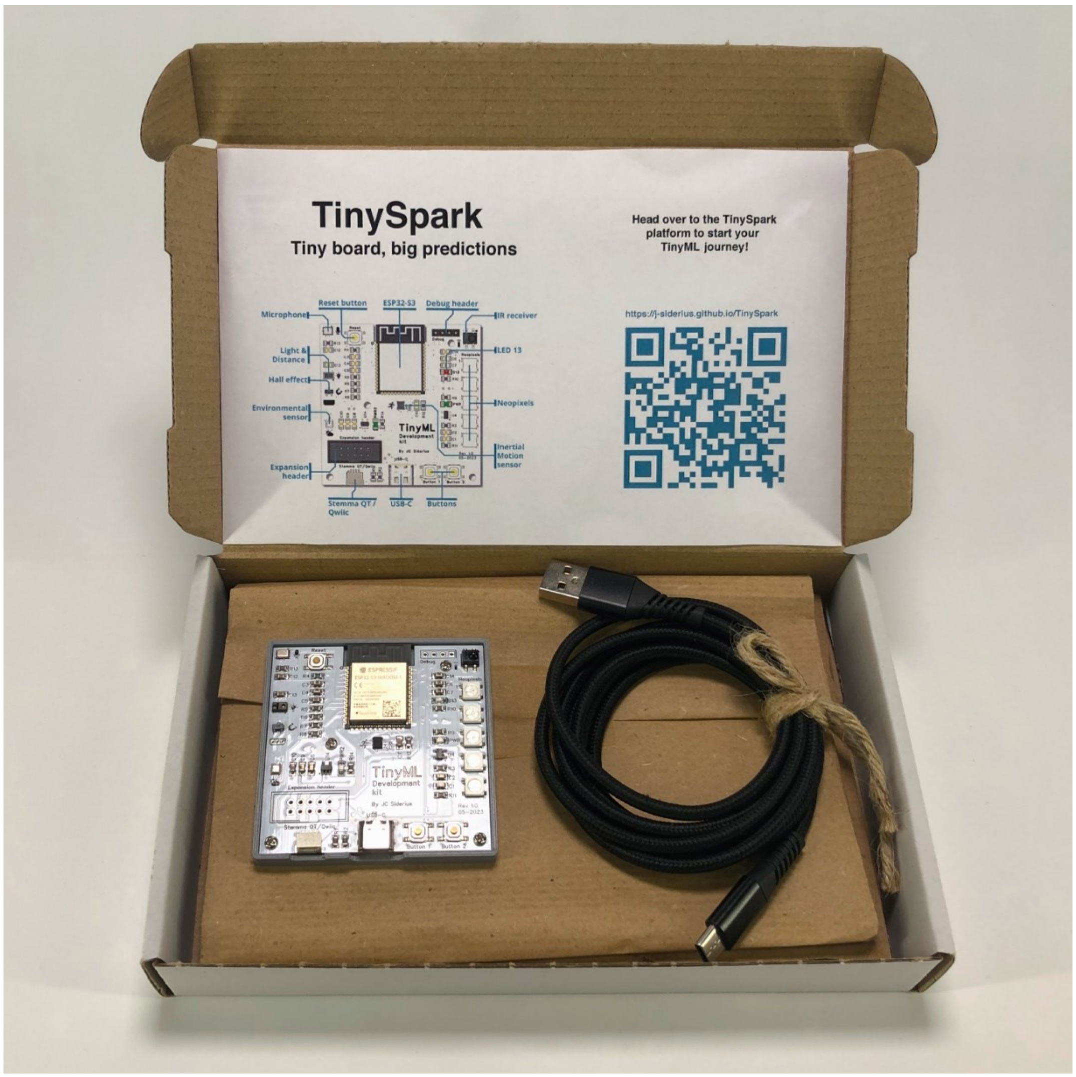
Teaching (Tiny)ML using tangible educational methods



Jannick Siderius
S2100282

Creative Technology bachelor thesis
Semester 2022-2

Supervisor: M. Gerhold
Critical Observer: M. Gómez Maureira

Acknowledgement

I am very grateful to my two supervisors, Marcus Gerhold and Maro Gómez Maureira, for their invaluable guidance, advice, and efforts in arranging various aspects of my thesis. Their expertise and support have been very helpful in shaping the outcome of my research. I would also like to extend my appreciation to my parents and friends, whose support and availability as rubber duckies during programming and writing, as well as their proofreading assistance have been incredible. I am thankful to the testers who participated in my user testing. Their willingness to dedicate time and provide valuable insights and feedback has significantly enriched the quality of my research. Last, I would like to thank the SIL-EE laboratory of the University of Twente, for granting me access to their circuit board assembly facilities.

Abstract

Tiny Machine Learning is the process of integrating machine learning algorithms into resource-constrained edge devices. The technology is already embedded into applications from smart home speakers to industrial pipeline leak detection and will continue to make a growing impact on businesses, hobbyists and industries alike. Currently, there is little engaging educational material available on Tiny Machine Learning for students. With the help of a tangible educational kit and by project-based learning materials this research aims to fill in this void.

In order to achieve engaging and interesting learning experiences, research was conducted to establish effective learning methods as well as gain background knowledge about Tiny Machine Learning. The resulting findings helped shape the design requirements for the realisation of the end product, an educational kit called TinySpark, that teaches Tiny Machine Learning with the help of a custom development board and interactive online platform.

The educational kit was evaluated through a user experience test, which was followed by a semi-structured interview. Test participants were enthusiastic about TinySpark and noted that their engagement and interest in the topic had grown. According to some participants, the kit could be easily expanded by adding more modules and project material in the future. Overall, the user experience testing was a success, as participants gained knowledge on complex concepts and could autonomously deploy Tiny Machine Learning models to the development board.

In conclusion, educational kits proved very engaging and useful in teaching Tiny Machine Learning to users. The development board and interactive online platform enhanced their comprehension and knowledge. By applying teaching methods like these, it is possible to effectively prepare students for a future filled with Tiny Machine Learning applications.

Contents

Glossary

AI

Artificial Intelligence is the overarching term for software that can perform (seemingly) intelligent decisions using its inbuilt algorithms. AI is both used for pre-programmed intelligent behaviour, as well as learned intelligence e.g. through machine learning algorithms.

Development Board

A circuit board containing a microcontroller chip as well as supporting hardware to enable engineers or users to interact with the chip easily. Most commonly used for prototyping and in hobby projects, e.g. the Arduino Uno development board.

Edge Device

A device which is located at the edge of a network, in this context, an edge device is considered any appliance which is meant to be used by an end user. E.g. smart speaker, fitness tracker, autonomous car, digital dog collar.

IMU

An Inertial Measurement Unit is a mechanoelectrical sensor which can measure forces in different directions. Most commonly, these sensors are used to detect acceleration and rotation in various axes.

Machine Learning

The act of letting computers 'learn' certain tasks without explicitly telling the program what to do. Machine learning uses algorithms to analyse a dataset and mathematically calculate how to correctly predict outcomes based on certain inputs.

SDK

A Software Development Kit is a specialised piece of software that can be used to write and compile code for specific hardware. It often includes all necessary configurations for this hardware and can help developers when writing software functions.

Tangible learning

The way of learning concepts using physical objects, e.g. understanding physical properties through playful interaction.

TinyML

A branch of machine learning that focuses on optimising algorithms to be able to run on edge devices and perform using little power and processing power.

1 Introduction

Tiny Machine Learning (TinyML) powers detection in a wide range of applications from smart home speakers to pipeline leak prevention. This relatively new field within computer science enables engineers to integrate powerful Artificial Intelligence (AI) into so-called edge devices. These appliances are low-power, low-cost and often do not require an internet connection to function. Combined with sensors, edge devices are ideally suited for a multitude of functions and they enable designers and engineers to explore new avenues for the integration of this technology.

Presently, only two approaches to integrate TinyML into projects and devices exist. The first, 'bare-metal' approach, requires intrinsic knowledge of complex mathematical and computer science topics. This method relies on fundamentally programmed AI, which calls for an understanding of low-level programming languages such as C, as well as mathematical knowledge of linear algebra and statistics [1]. The second approach relies heavily on cloud services, offering TinyML through a 'Platform as a Service' (PaaS) [2]. Users upload their data and let the platform take care of teaching the algorithm and programming the AI. A general approach for learning TinyML is lacking, and existing development and learning systems are complex and scarce.

The research presented aims to solve this problem by designing a self-supported educational kit (TinySpark) for learning and implementing TinyML into projects and devices. Machine learning projects in this kit could for example be used to detect human presence in rooms or to analyse vibration patterns to identify maintenance problems. The educational kit will be aimed at college and university-level students, as they are the designers and engineers of the future. Teaching them new technologies early on is important to ensure proper adoption and usage. The kit will be designed for students who have an application-level understanding [3] of programming concepts.

The objective of this research was to *determine if educational kits and project-based learning systems can teach students TinyML effectively (Research Question)*. To properly answer this question, two additional subjects needed to be investigated. First, teaching methods were explored to find *which methods can be used to enable successful self-supported learning of complex topics such as TinyML (Sub Research Question 1)*. In addition, a self-supported educational kit for learning and implementing TinyML was produced. Second, the educational kit was analysed to see if it was suitable *for teaching complex topics in a comprehensible and engaging way to university students (Sub Research Question 2)*. Combined, these investigations provide valuable insights for answering the initial research question.

The research is structured as follows: In chapter 2, background literature and the state of the art on educational kits and TinyML are presented. In chapter 3, the methodology for answering the research goals is given. Chapter 4 discusses the preliminary ideation that took place to develop an educational kit around TinyML, and at the end of the chapter, requirements for the product are discussed. In chapter 5, the realisation of the educational kit will be demonstrated. In chapter 6, the product is evaluated through user experience testing. The results of the user test will be discussed in chapter 7. Finally, this research is concluded in chapter 8, where findings are summarised and future work recommendations will be given.

2 Background

To make well-informed decisions during the rest of the research and product development, it is important to examine existing research and products. This chapter comprises expert opinion(s) on the subject, comprehensive literature research, explores popular educational methods and investigates the influence of adding tangibility to the learning process. After that, the origins of machine learning and the importance of TinyML are investigated. Lastly, an important insight into the current state of the art is gained. Here, the currently available development boards used for TinyML, available online learning material, programming languages as well as online learning platform systems are compared.

Educational Methods

There has been growing interest in different educational methods besides the 'traditional' teacher-centred learning. Tangible and self-supported learning have emerged as alternative methods to engage pupils and students with varying curricula and subjects. Tangible learning describes the act of understanding topics or functions through the use of physical structures, such as model representations or even highly technical devices [4]. Horn et al. [5] show that tangible learning can facilitate the learning process by increasing engagement and understanding. Students face increasingly complex concepts and problems, which could become easier to understand by incorporating tangible teaching into their learning process.

Successful learning and teaching methods

To evaluate educational methods, it is important to establish what factors contribute to successful learning and teaching methods. There are many different aspects that influence learning; not only the teaching environment and style, but also less obvious effects such as peers in the classroom and framing of learning materials.

Teaching in a way which enables exploration of the subject matter at hand is one of the key principles of successful teaching. Inquisitive behaviour is a strong motive for learning new things [6]. Matthews et al. [7] support this by stating that 'curiosity driven learning is very important for development'. Similarly, Resnick and Silverman [8] implement a system of 'low floors and wide walls' in their educational recommendation, supporting a wide range of exploration opportunities. Students who are allowed to participate in learning activities together, collaborating on assignments, problem solving and participating in discussions about the material, seem to gain a deeper understanding in the subject matter. The collaborative experiences foster the formation of different perspectives and teach important skills such as negotiation, tolerance and listening [6]. Matthews et al. [7] also observed that students witnessing other students think about and act on inputs stimulates the exploration of new connections.

Besides an open and inquisitive approach, learning should be tailored to the specific context it is presented in. Information should be fitting to the intended target audience, be it for a certain grade or scholarly environment. Bekker et al. [9] state that teaching materials have to be age appropriate, because if they are too easy, students lose interest. On the other hand, if materials do not fall within students' realm of imagination, they might be perceived as too hard to grasp [10]. Furthermore, by targeting specific topics in context, for example animals in their natural habitat, much more student engagement can be achieved according to Stanton et al. [11]. The incorporation of context can be extended to include support for multiple learning styles. M. Resnick and B. Silverman [8] describe this as encouraging 'many paths', in which the students are let free to choose which learning path they want to seek out. Traditionally, teacher-centred teaching methods may not always be the best choice for course materials. That is why Bekker et al. [9] suggest that deeper understanding and applied knowledge of subject matter can often be heightened using alternative teaching methods such as tangible learning. Playful learning is another opportunity to change teaching styles: it can enable students to participate in technologies that they may not be familiar with, or do not have the domain knowledge for to fully understand [7].

Teaching materials not only need to be tailored to students' specific context, but the material should also carefully consider where to apply abstraction of concepts. By employing so-called 'black-boxes' [8], teachers can purposefully obscure facts or knowledge to explain concepts without focussing on unnecessary details. While this can lead to greater participation even for students without intrinsic domain knowledge according to Matthews et al. [7], Resnick and Silverman [8] point out that designers should be careful to not create unnecessary confusion by leaving out critical components. Experimentation and exploration are strong motivators for learning [6]. Additionally, 'curiosity driven learning is very important for development' according to Matthews et al. [7].

As seen in this part of the review, there are many aspects that influence the effectiveness of teaching and learning methods. Experimentation and exploration prove fairly effective in teaching new concepts, while collaborating with peers can increase learning as well as improve understanding of a topic. Furthermore, providing material that is well-tailored to the student population and that enables multiple paths of learning can contribute to engaging teaching that reaches a broad audience. Lastly, educators should be mindful to not obscure (parts of) concepts that might prove important to garner a good understanding of the topic. By using these pointers, educators may deliver effective and engaging learning experiences to their students.

Effects of tangible learning

Successful learning and teaching methods involve many different aspects of education. Tangible learning, in which physical commodities that support the teaching topic or context are used alongside other educational methods, could have a profound impact on students. It is important to explore in what way tangible interactions influence the students' learning process.

Tangible artefacts can have a great impact on the collaboration between students in the classroom. Stanton et al. [12] note that especially large tangibles slow down interactions and make student experiences more deliberate. The researchers however find that one downside of this approach is that it can make individual interactions and decisions more difficult. Zuckerman et al. [13] refute this, stating that collaboration actually has a positive impact on learning as it can lead to discussion. This can lead to greater understanding of concepts and improve classroom cohesion. The finding is confirmed by research from S. Somyürek [14] and Bekker et al. [9].

Tangible learning methods also provide ways to discover difficult topics in a participatory and hands-on way. Price et al. [6] find that being able to handle objects and interact with them enables creativity, forms knowledge and stimulates awareness. Furthermore, physically interacting with tangible objects can immerse students into the learning process and raise awareness of concepts [15]. P. Marshall [12] is more careful in this regard, stating that in 2007, frameworks for testing and studying tangible learning effects on students are lacking and need to be investigated further. In a literature review conducted in 2020, Matthews et al. [7] however find that there is (now) ample evidence to support tangible learning as a viable way to build effective theoretical knowledge.

In science and technology subjects, there still exists a gap between gender participation: girls are less likely to follow courses in these areas. While testing tangible programming interfaces, Horn et al. [15] found that tangible interaction methods were not only more engaging and prolonging interest, but they also attracted more traditionally underserved groups of participants. For example, girls were even more likely to try and interact with tangible interfaces than boys. Tangible learning seems to be less daunting and it invites people to try something new, even if they are not familiar with its workings. In addition to this, M. S. Horn [16] finds in a later study that tangible objects 'lure' participants into interacting with them, stating that experiences that include hands-on interaction are more attractive in general.

Although it might not be immediately obvious, tangibles in education were more popular than ever during COVID-19. Many teachers have been wondering how they could teach certain disciplines of academia remotely. For example, laboratory practice was made almost impossible due to stay-at-home regulations, even though many in higher education consider it an essential part of scientific education. Even before it became a mandatory practice, J. P. Oliver and F. Haim [17] already proved in 2009 that at-home laboratory experiments can be successfully applied. Their research even shows that take-home labs led to a better understanding of subject matter, as well as higher acceptance and motivation, which in turn resulted in higher grades. Additionally, students acquired more academic skills and developed a more responsible attitude towards laboratory work in general. This is corroborated in a more general educational study by Jones et al. [14], who not only found improved achievements, but also a more investigative mindset which led to important individual reflection.

As seen above, tangibles can impact students and their learning experience in many different ways. Not only do lessons augmented with physicalities improve the collaboration between students, but they can also lead to a more positive classroom environment. This added dimension can lead to a better understanding of tough concepts and can improve even the engagement of students who otherwise might not have been interested in learning certain subjects, one good example being female students exploring science. Additionally, many different types of educational activities such as lab work can benefit from added tangible interaction, as improvements in grades and level of understanding show. When analysing all effects of tangible learning, a positive influence on students' abilities to learn can be found.

Comparing teaching methods

After analysing factors that influence how effective education can be for students, as well as the benefits of tangible learning methods, it is appropriate to also look at a direct comparison between more traditional, teacher-centred education and tangible learning methods. By analysing both methods side-by-side a better decision can be made when choosing an educational model.

First, tangible learning methods can lead to ineffective education if the material is implemented poorly. Bekker et al. [13] as well as Dickerson et al. [18] find that tangible learning is only useful if teachers fully understand the underlying materials and subject. When this is the case, teachers can facilitate learning through tangible methods successfully. Additionally, teachers with more experience in their subject already use tangible methods more often in their lessons than less experienced teachers [14]. Experienced teachers were furthermore observed to be more confident in lessons with opportunities for open inquiry from students. Second, teacher-centred lessons also seem to be less optimal for teaching time, as more time is spent on explaining concepts and materials than answering questions, discussing with students and evaluating results [17]. Students seem to appreciate this more open approach to lessons: S. Somyürek [14] states that students found tangible problem-solving activities more useful and expedient than traditional education. Not only that, but Dickerson et al. [18] also find that tangible learning can empower populations of students that may not otherwise be able to approach certain subject matter.

By directly comparing different teaching methods, a clear effect of tangible learning on students can be observed. The most important discovery concerns teacher knowledge: it becomes clear that curriculum material needs to be fully comprehended by teachers to implement tangible methods. Although there might exist some caveats about the adoption of tangible education, there are definitive indicators that implementing them benefits students.

This literature review aimed to ascertain whether tangible educational methods can benefit students more than traditional, teacher-centred learning methods. Through investigation of the aspects of teaching that influence successful learning, it was observed that methods including an open and inquisitive approach, tailored materials as well as collaborative learning benefit students most.

Tangible learning methods were also evaluated and found to have a profound effect on students' ability to understand arduous topics, but also to help spark interest among underserved student groups for specific courses. Further, in direct comparisons between educational methods, benefits of implementing tangible materials into learning have been shown.

In conclusion, tangible educational methods seem to benefit students, although it is important to consider when, how and why they are implemented, to ensure that they are deployed effectively.

(Tiny) Machine Learning

The earliest mention of machines being able to learn using mathematics is in the 1943, in the paper 'A logical calculus of the ideas immanent in nervous activity' by McCulloch and Pitts [19]. The authors propose a system which can mathematically describe the neural activities in the brain, leading to the first description of artificial neurons. In 1950, Alan Turing introduced a new technique for testing if a machine can think, commonly known as the 'Turing Test', by checking if answers from said machine are good enough to convince a human interviewer they are talking to another human [20]. Rosenblatt [21] was the first researcher to implement the concept of neurons into physical hardware, by building the Mark I Perceptron, a machine the size of a closet that was supposed to recognize images taken by the camera mounted to its front panel. His attempts had limited success. Two years later, in 1959, Samuel et al. [22] coined the term machine learning, showing that a computer was able to learn to play checkers by training itself only on simple rules it was given.

Plunging further into the historical development of machine learning, it is clear to see that major advancements were made in the field: in 1967 the introduction of the 'Nearest Neighbour' algorithm to solve the travelling salesman problem [23], in 1979 the Stanford Cart [24] which was able to autonomously navigate an obstacle course without human intervention, after that came NETTalk in 1986, a system developed by Sejnowski and Rosenberg [25] that could learn to speak words in a similar way to babies, then came DeepBlue, which beat chess champion Garry Kasparov in 1989 [26], leaping forward, in 2011 IBMs Watson computer successfully competed in Jeopardy!, a popular American quiz show [27] and later in 2015, Google's AlphaGO managed to beat the world champion in Go (an ancient Chinese board game long thought to be too difficult to master for a computer) [28].

Around 2016, big companies like Google and Meta (formerly Facebook) became increasingly interested in the use of machine learning and began developing their own frameworks to simplify the implementation of ML algorithms. This resulted in two popular open-source tools; TensorFlow [29] and PyTorch [30], which enable researchers and developers to easily build and deploy complex machine learning models for various applications. These frameworks also facilitated the advancement of deep learning, a subset of machine learning that uses multiple layers of artificial neural networks to learn from data. With the widespread adoption of smartphone technology came mobile machine learning, which enabled applications on mobile devices to learn from data they generate. Currently, many smartphone manufacturers, as well as smartphone operating systems, use machine learning to recognise speech commands and images as well as generate recommendations for its users [31]. This historical overview shows that machine learning was and still is a fast-evolving field that has revolutionised numerous domains and continues to shape our modern world.

With the increasing popularity of machine learning and its applications in various domains, there has been a growing interest in extending its capabilities to tiny devices, hence the emergence of TinyML. Introduced in 2016 by Han et al. [32], Tiny Machine Learning or then rather 'Deep Compression' of existing machine learning algorithms was primarily focussed on reducing the power consumption of the memory used to store prediction models. While memory might not be the main power factor in modern microprocessors anymore, optimising machine learning models to run on the tiny devices is still a very significant topic.

Considering that an estimated 28 billion microcontrollers were shipped in 2020 alone [33], and smart appliances such as voice assistants [34], gait analysing fitness trackers [35] and many more are increasingly becoming popular, interest in TinyML is growing. The push towards using more Edge Computing devices and running machine learning inference on them is well summarised by Jeff Bier [36] using the acronym BLERP. Bier [36] concludes that Bandwidth, Latency, Economics, Reliability and Privacy are the main aspects that interest companies and developers to employ machine learning techniques on the edge.

Bandwidth is important because edge devices often collect vast amounts of data from their embedded sensors, however they rarely have the transmission capability or even the power to send all data to the cloud for analysis. Furthermore, data uploading into the cloud seems universal, however when looking at large data volumes and commercial use, there are often bandwidth limits in place, which can become problematic when ingesting for example video data. Added to this, edge devices might not even have a connection to the internet, for example, when they are deployed in remote regions.

Latency is another key factor, as even connections with a very high bandwidth can have significant delays in transmission and reception of data. For some applications of edge machine learning, for example autonomous driving, a faster response is needed to resolve critical decisions.

Economics influence machine learning enormously. Not only does bandwidth access to data centres cost a lot, large servers that are needed to run machine learning algorithms are also expensive and require vast amounts of energy to operate. Most hardware used in TinyML applications is low-cost and very available, making it ideal to deploy at a large scale.

Reliability ties back to devices needing to be connected to the internet to function. If controllers are running critical tasks, it should become clear that a dependence on internet connectivity is impractical. If machine learning models can run locally on the device itself however, the risk of failures and intermittency in service is reduced.

Privacy is another key factor in TinyML. Since prediction (and sometimes even training) of the machine learning algorithm is performed locally, there exists no need for constant data uploading and sharing to external storage or computing nodes. Because of this, devices can function completely independently, even when outside services might not be available. This can give not only peace of mind to privacy-conscious persons, but also to companies and institutions whose data might be a valuable resource.

Early research in TinyML was mostly focussed on implementing PC-based machine learning algorithms on microcontrollers using compression and consolidation functions. So called pruning of machine learning nodes and features often required complex algorithms, like the Bonsai algorithm as introduced by Kumar et al. [37]. Compression techniques were later introduced, using algorithms like the Huffman Coding [38] to store complex features using less memory. In more recent years, Google's Tensorflow has gotten an entire software branch dedicated to running on resource constrained devices, called Tensorflow Lite Micro [39]. Additionally, STMicroelectronics [40], the manufacturer of the popular STM32 line of microprocessors has added X-CUBE-AI to their integrated development environment (IDE), to add support for many different machine learning libraries to its chips [41].

State of the Art

TinyML Development Boards

Below in Table 1 is an overview of development boards currently marketed as 'TinyML ready.' The boards were found by evaluating the 'Development board' section of the major electronics retailers Digikey[1], RS[2] and Sparkfun[3] as well as the Arduino[4] online shop. The selection was based on product descriptions and -titles, specifically focussing on mentions of 'TinyML,' 'Edge Machine Learning,' 'TensorFlow compatible' and 'AI.' The Target Audience was deduced from the product description and marketing information. Table 1 was generated in the spring of 2023.

Product	**Programming environment**	**Chip & Memory**	**Sensors**	**Target Audience**	**Cost**
Arduino Nano 33 BLE Sense	Arduino IDE, MicroPython, Edge Impulse SDK	nRF52840@ 64MHz, 1MB ROM, 256KB RAM	9-axis IMU, Humidity, Pressure, Temperature, Microphone, Gesture	Hobby, School, Semi-professional	€ 35
Arduino Nano RP2040 Connect	Arduino IDE, MicroPython, Mbed	RP2040 @ 133MHZ + 133MHZ, 448KB ROM, 264KB RAM, 16MB Flash	6-axis IMU, Microphone, Temperature	Hobby, School, Semi-professional	€ 26
Arduino Nicla Sense Me	Arduino IDE	nRF52832 @ 64MHz, 512KB ROM, 64KB RAM, 2MB Flash	9-axis IMU, Humidity, Temperature, Pressure, CO2	Semi-professional	€ 69
Arduino Nicla Voice	Arduino IDE	NDP120 + nRF52832 @ 64MHz, 512KB ROM, 64KB RAM, 16MB Flash	9-axis IMU, Microphone	Semi-professional	€ 69

1 https://www.digikey.nl/en/products/category/development-boards-kits-programmers/33 (accessed Apr. 14, 2023)
2 https://nl.rs-online.com/web/c/raspberry-pi-arduino-development-tools /development-tools-single-board-computers/microcontroller-development-tools/ (accessed Apr. 14, 2023)
3 https://www.sparkfun.com/categories/393 (accessed Apr. 14, 2023)
4 https://store.arduino.cc/collections/boards (accessed Apr. 14, 2023)

Product	Programming environment	Chip & Memory	Sensors	Target Audience	Cost
Arduino Nicla Vision	Arduino IDE, OpenMV	STM32H747AII6 @ 480MHz + 240MHz, 2MB ROM, 1MB RAM, 16MB Flash	6-axis IMU, Camera, Microphone	Semi-professional	€ 99
Arduino Portenta H7 + Vision shield	Arduino IDE, MicroPython, Mbed, OpenMV	STM32H747XI @ 480MHz + 240MHz, 2MB ROM, 1MB RAM	Camera	Professional	€ 130
SeeedStudio Wio Terminal	Arduino IDE, MicroPython, CodeCraft	ATSAMD51P19 @ 120MHz, 4MB ROM, 192KB RAM	3-axis IMU, Microphone, Illumination	Hobby, School, Semi-professional	€ 40
M5Stack Core2 ESP32	Arduino IDE, MicroPython, FreeRTOS, UIFlow	ESP32 @ 240MHz + 240MHz, 8MB ROM, 512KB RAM, 8MB PSRAM	6-axis IMU, Microphone	Hobby, School, Semi-professional	€ 50
Himax WE-I Plus	Edge Impulse SDK	HX6537-A @ 400MHz, 2MB ROM, 2MB RAM	3-axis IMU, Microphone, Camera	Professional	€ 70
Arducam Pico4ML	Arducam toolchain	RP2040 @ 133MHZ + 133MHZ, 2MB ROM, 264KB RAM	9-axis IMU, Microphone, Camera	Hobby, School, Semi-professional	€ 25
SparkFun Edge	Ambiq Micro SDK toolchain	Apollo3 Blue @ 48MHz, 1MB ROM, 384KB RAM	3-axis IMU, Microphone, Camera	Semi-professional, Professional	€ 27
OpenMV Cam H7	MicroPython, OpenMV	STM32H743VI @ 480MHz, 2MB ROM, 1MB RAM	Camera	Semi-professional, Professional	€ 80
Sipeed Mo Sense	TinyMaix, FreeRTOS	BL702 @ 144MHz, 192KB ROM, 132KB RAM, 512KB Flash	6-axis IMU, Microphone	Hobby, Semi-professional	€ 10

Product	Programming environment	Chip & Memory	Sensors	Target Audience	Cost
Syntiant TinyML	Arduino IDE, Edge Impulse SDK	NDP101 + ATSAMD21G18 @ 48MHz, 256KB ROM, 32KB RAM, 2MB Flash	6-axis IMU, Microphone	Semi-professional	€ 35
STM32 B-L4S5I-IOT01A Discovery	STM32 Cube, Edge Impulse SDK	STM32L4S5VIT6 @ 120MHZ, 2MB ROM, 640KB RAM, 8MB Flash	9-axis IMU, Humidity, Pressure, Temperature, Microphone, Gesture	Hobby, School, Semi-professional	€ 50
Espressif ESP32-S3-Box	ESP-IDF, FreeRTOS, MicroPython	ESP32-S3 @ 240MHz + 240MHz, 8MB ROM, 512KB RAM, 8MB PSRAM	6-axis IMU, Microphone	Hobby, Semi-professional	€ 45
Espressif ESP-S3-EYE	Arduino IDE, ESP-IDF, FreeRTOS	ESP32-S3 @ 240MHz + 240MHz, 8MB ROM, 512KB RAM, 8MB PSRAM	3-axis IMU, Microphone, Camera	Semi-professional	€ 55
Espressif ESP-S3-Korvo-2	ESP-IDF, FreeRTOS	ESP32-S3 @ 240MHz + 240MHz, 8MB ROM, 512KB RAM	Microphone, Camera	Semi-professional, Professional	€ 90
Nordic Semi Thingy:53	Edge Impulse SDK, nRF connect SDK	nRF5340 @ 128MHz, 1MB ROM, 512KB RAM	9-axis IMU, Humidity, Temperature, Pressure, CO2, Colour, Microphone	Professional	€ 60
Nordic Semi Thingy:91	Edge Impulse SDK, nRF connect SDK	nRF52840 @ 64MHz, 1MB ROM, 256KB RAM	3-axis IMU, Humidity, Temperature, Pressure, CO2, Colour	Professional	€ 110

Product	Programming environment	Chip & Memory	Sensors	Target Audience	Cost
Adafruit EdgeBadge	CircuitPython	ATSAMD51J19 @ 120MHz, 512KB ROM, 192KB RAM, 2MB Flash	3-axis IMU, Microphone, Illumination	Hobby, School	€ 35

Table 1: Development boards marketed as 'TinyML' ready.

From the comparison Table 1, some interesting findings regarding the specifications of development boards aimed at TinyML can be found.

First, many of the development kits support the Arduino IDE [42] and Edge Impulse SDK [43], although quite a few chips can also only be programmed by using proprietary software development kits (SDKs). This can have several (dis)advantages. By using a proprietary SDK, companies retain full control over how software is written for their specific chip, efficiency improvements can be implemented as well as specialised functionality of chips can be fully utilised. However, because these SDKs are often closed source, they do not adapt to specific users' needs as well as more generalised programming environments such as the Arduino IDE [42], which has a lot of community support.

Second, many development boards feature 32-bit microcontrollers that have more processing power than most popular chips [44] such as the ATmega328P (8-bit, 16MHz), PIC16F877A (8-bit, 20MHz), STM32F103 (32-bit, 72MHz) or the ESP8266 (32-bit, 80MHz). Additionally, many 'TinyML ready' development boards contain more storage (ROM) and working memory (RAM) than the popular chips. Given the need for the storage of inference models, as well as available memory during prediction, this makes sense.

Furthermore, boards made for TinyML applications typically feature a selection of sensors on-board, so that developers do not need to connect external hardware to get started with programming. The most common sensors include an Inertial Measurement Unit (IMUs, specifically 6- to 9-axis), a Microphone, environmental sensors like Temperature or Humidity and a Camera. Last, the average price of 'TinyML ready' development boards is around €60.

Tiny Machine Learning Educational Kits

Even though TinyML is not a completely new field anymore, and multiple microcontroller manufacturers have created development boards as seen in the previous sections, learning kits for TinyML are still scarce. Although all board developers include code examples with their product, not many prepare a full educational experience around their device. The availability of educational material for TinyML development boards was established by inspecting the product descriptions and company websites of all board manufacturers from Table 1. At the moment of writing, there are three manufacturers who do provide learning materials to some extent:

1. Arduino[5]: The Arduino Tiny Machine Learning Kit can be used in conjunction with two different EdX / HarvardX online courses, aimed at introducing students to TinyML and using ML prediction on the Arduino Nano 33 BLE Sense. The courses focus on keyword spotting, visual wake words, anomaly detection and gesture recognition [45]. The course uses the online platform Edge Impulse [2] for data capture, model training and deployment. The usage of the camera module as well as the environmental sensors on the development board included in the kit are not explained. In the Arduino blog [46] there is another guide on implementing TinyML using TensorFlow Lite Micro, but this only concerns two use cases and is more of a tutorial.
2. Seeedstudio[6]: TinkerGen, the education division of Seeedstudio has developed a TinyML course using the Wio Terminal development board. The course covers two types of gesture recognition, audio detection, a people counter and a smart weather station project. The model training and deployment uses the online platform Edge Impulse [2]. The manual for the course contains some inaccuracies regarding the capabilities of the target device, as well as missing source code for some examples. Seeedstudio also has tutorials available for their own software development suite CodeCraft, in which basic TinyML models can be trained and deployed.
3. ArduCam[7]: The ArduCam Pico4ML board is accompanied with several tutorials. These cover different types of image recognition, wake word detection and gesture recognition. The tutorials use pre-trained models and the online platform Edge Impulse [2].

From this analysis, it becomes apparent that most development boards that can be used for TinyML applications lack comprehensible educational materials following a curriculum that builds up step-by-step. Most tutorials primarily focus on using an (online) platform like Edge Impulse [2], as well as deploying pre-trained machine learning models to the development board. This limits the learning effectiveness of the material, since users lack the possibility to produce applications of their own. Additionally, most learning materials available are bound to specific development boards or (online) code platforms. This segmentation could hold back users in their learning process, since they only explore one specific facet of TinyML, instead of learning about the concept as a whole.

[5] https://store.arduino.cc/products/arduino-tiny-machine-learning-kit (accessed Apr. 14, 2023)
[6]https://files.seeedstudio.com/wiki/Wio-Terminal-TinyML/No-code_Programming_to_Get_Started_with_TinyML.pdf (accessed Apr. 14, 2023)
[7] https://www.arducam.com/product/arducam-pico4ml-tinyml-dev-kit-rp2040- board-w-qvga-camera-lcd-screen-onboard-audio-b0330/ (accessed Apr. 14, 2023)

Product Research

In the process of designing the product two more highly relevant background topics appeared that need to be discussed: programming languages and online platforms. These more compact research topics were used as reference during the Realisation in Chapter 5.

Development Board Programming Languages

As discovered in the comparison of TinyML development boards, the Arduino IDE [42] and the Edge Impulse SDK [43] are often used to program development boards. Because there are many programming languages available for microcontrollers, it is worthwhile to investigate the differences between them. The aim of this comparison is to have a clear overview of the available programming languages on popular development boards.

C / C++

The C and C++ programming languages are the most well-supported on microcontrollers. Before the advent of popular SDKs, all microcontroller code had to be written in these languages, since it is easily converted into machine code (the actual processor instructions used by microcontrollers). Like the Arduino language, C and C++ code needs to be compiled and then uploaded as a binary file to the development board. The programming languages are used in many platforms such as STM32 and Espressif microcontroller ecosystems.

Arduino[8]

The Arduino programming language might be the most used embedded programming language in hobby and semi-professional applications. The language, based on C++, is used across many different development boards and relishes widespread support and integration of sensor libraries. Code written in Arduino needs to be compiled and then uploaded as a binary file to the development board. The programming language is also integrated into many SDKs such as the Edge Impulse SDK [47] and is used on many platforms, including the Arduino microcontroller ecosystem.

MicroPython[9]

The MicroPython programming language is relatively young (2014) and is based on the popular Python 3 language. MicroPython code is interpreted, so it does not need to be compiled before uploading to the development board. This also enables users to see actual code they uploaded to a board at a later date, since the plain-text program is saved to the microcontroller. MicroPython is integrated into some SDKs such as the Edge Impulse SDK [47] and the OpenMV SDK [48]. It is also used on many platforms such as the Espressif and Raspberry Pi Pico microcontrollers.

[8] https://www.arduino.cc/reference/en/
[9] https://micropython.org/

CircuitPython[10]

CircuitPython is a branch of the MicroPython, made by Adafruit Ltd. Subsequently, the language shares many similarities with MicroPython. The company wanted to make programming modern development boards even easier, and implemented easy code editing as well as a plethora of sensor libraries that were previously not available in MicroPython. The programming language supports many platforms such as modern Arduino, Espressif and STM32 microcontrollers.

FreeRTOS[11]

The last programming language is FreeRTOS, which is a C-based, speciality language focussed on real-time applications. Because of this, FreeRTOS lends itself well to the requirements of (Tiny) machine learning. The language is compiled, so it uploads the compile program as a binary file to the microcontroller. The programming language is integrated into Espressifs development environment and can be used on many platforms such as Espressif and STM32 microcontrollers.

Online Platforms

There are many different online platforms available for the publishing of (educational) materials. Some of the popular platforms are discussed below, comparing their ease-of use, unique features, accessibility and price.

Google Classroom[12]

Google Classroom offers a simple and user-friendly interface, making it easy for teachers and students to navigate and engage in virtual classrooms. It integrates well with other Google tools and the service is free for educational institutions, which makes it an accessible option for educators and students. The software platform offers a complete learning environment and student management system.

Skillshare, Coursera and Udemy[13]

Many for-profit online learning platforms like Skillshare, Coursera and Udemy offer paid video courses taught by industry as well as college and university professors from around the world. The platforms boast extensive opportunities for studying assorted topics. Services are mostly paid, with some one-time buy options as well as subscription models. Teachers do not pay to set up a course on these platforms.

---

[10] https://circuitpython.org/
[11] https://www.freertos.org/
[12] https://edu.google.com/workspace-for-education/classroom/
[13] https://www.skillshare.com/en/, https://www.coursera.org/, https://www.udemy.com/

Personal Website

This option gives the most freedom in terms of possibilities and features, but since it would be required to program and research the website from scratch, this approach might suffer from a high starting cost in terms of development time. Apart from hosting costs, the teacher is completely free to choose any payment model they prefer.

GitHub Pages[14]

GitHub Pages is an accessible platform for hosting static websites and web projects. It has seamless integration with Git, so developers can effortlessly apply version control to their projects and collaborate with others. GitHub Pages supports various frameworks and technologies, such as the site builder Jekyll. While some functionality might need to be developed, the extensive community support makes GitHub Pages a popular choice among developers. The Pages environment gives a free domain and storage where projects can be presented.

EdX[15]

EdX is an online learning platform which provides a wide range of courses from prestigious universities and institutions. It has interactive learning features, including discussion fora and quizzes, which enhance student engagement and knowledge retention. The service is free, but the platform can be upgraded to include more advanced features such as class management and administration.

[14] https://pages.github.com/
[15] https://www.edx.org/

3 Method

TinyML can be quite a complex topic, nevertheless it has many interesting and useful applications. It was also found that tangible learning methods such as kit-based learning can be a powerful tool for instruction. From the state-of-the-art research in Chapter 2, it is obvious that the current options for studying TinyML are lacking. To address this shortcoming in the field and try to find a solution, a new kit-based learning system was designed (see Product), subsequently tested by students and then analysed (Evaluation) to find out, if this modern approach facilitates a better and deeper learning about and understanding of TinyML.

Product

The design process of the product was led by the Waterfall design framework [49], a sequential methodology that is split into several key phases.

- Requirements: all necessary requirements are collected, including information on the background of the research, functional- and non-functional requirements, learning goals, scope, cost and timelines. The requirements and learning goals are defined in the MVP development process in Chapter 4.
- Design: solutions for the requirements proposed in the first phase are designed. These include hardware design, content outlines, user flows and the overall design of the product.
- Implementation: the (technical) implementation of the research takes place. This includes the software development, content writing, requirement implementation and product finalisation. The design and implementation phases are described in Chapter 5.
- Verification: the developed product is verified by user experience testing. Additionally, all requirements are evaluated. If major unexpected errors are found, it may be necessary to revise the design stage of the product, to comply with the requirements set in the first phase. This is generally very expensive and laborious to resolve. The verification process is discussed in Chapter 6.
- Finalisation: the product is ready and the research can be finalised. In accordance with the GP-track, this includes writing the GP thesis and preparing the GP defence. Additionally, the requirements are evaluated and the Research Questions posed are answered in the conclusion, in Chapter 8.

The Waterfall design framework was chosen because of its proven effectiveness in time planning [50]. Additionally, due to the structured nature of the framework, progress can be easily analysed and requirements are known upfront, reducing the variability in the implementation phase. This structure can be seen in Figure 1.

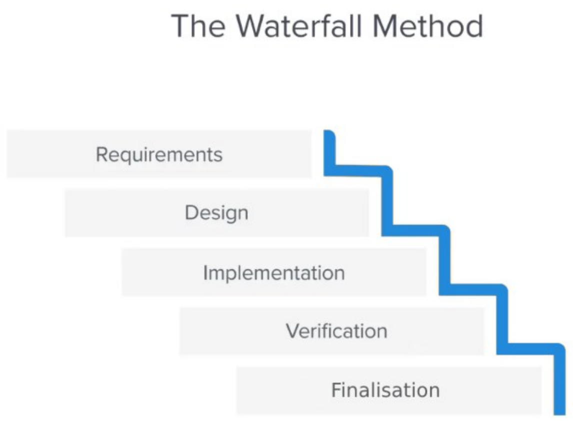


Figure 1: The waterfall method illustrated. Source: adapted from https://business.adobe.com/blog/basics/waterfall

Evaluation

To evaluate the effectiveness of the product, qualitative research was conducted. This method was chosen over quantitative research because of limited resources to construct the product. The research was conducted with participants from the target group, students in higher education, who used the product to learn TinyML concepts and applications. The test participants were given a product to take home and try out and interviews were conducted after using it. These interviews were semi-structured to evaluate participants' (learning) experience. Participants were asked about their baseline experience in programming and educational experiences. In the interview, participants were questioned on their experience with the product, the way of instruction, the supporting materials, any positive or negative occurrences as well as their level of knowledge on TinyML. After concluding the interviews, they were evaluated and key participant observations were identified. The results were analysed to identify the impression of the product, accomplished learning outcomes and overall valuation of the product.

The final evaluation of the product was conducted by using the Honeycomb UX framework [51], a seven-faceted model that helps assess products (see Figure 2 for a visual representation). While the framework was originally developed for analysing websites, it is now used for a wider range of assessments as well as product evaluation. Through its approach, the Honeycomb model ensures that products are not only valued on their usability, but also on other critical needs. By evaluating different approaches, it is easy to improve the product in small sections. The framework facets are:

- Useful: A product should be useful, where possible creating innovative solutions to requirements.
- Desirable: a product should be desirable, either for its value or looks.
- Accessible: products and designs should respect everyone, even if they want to use them with a disability.
- Credible: users of the product should be able to trust the information they receive.

- Findable: the product should be easy to navigate, and information is accessible and logically structured.
- Usable: designs should remain usable, meaning that they should not sacrifice usability for beauty for example.
- Valuable: the product should deliver value to its user, through gained knowledge, or by de-mystifying concepts.

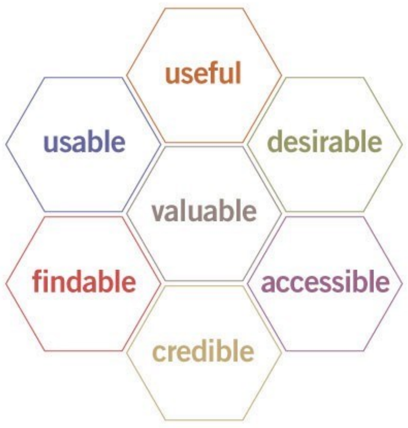


Figure 2: The honeycomb UX framework illustrated. Source: https://semanticstudios.com/wp-content/uploads/2004/06/honeycomb.jpg

4 MVP Development

The analysis of relevant literature and state of the art in Chapter 2 identified an important research gap. Although there seems to be an increasing interest in TinyML and multiple companies have released machine learning ready development boards, the instruction on implementing TinyML is lacking. While most companies include some examples to point new users of their development boards in the right direction to getting machine learning inference to run on the boards, underlying knowledge is missing. This could be solved through the design of a hands-on course in Tiny Machine Learning, which demystifies some of the common black-box principles of machine learning in general, while also teaching students to implement TinyML successfully for a wide variety of real-world problems.

This chapter explores the development of a Minimum Viable Product (MVP) and its key components, including a stakeholder analysis, the definition of learning goals and requirements, and the introduction of the MVP itself. In line with the Waterfall methodology in Chapter 3, it is important to lay the foundation before moving forward with the subsequent stages of the development process. The sections in this chapter provide guidance and structure to the design process later in the research.

Stakeholder Identification

To develop a targeted and mindful product to the proposed problem, it is important to identify the different stakeholders that affect the design process and product usage. In Table 2 below, the different stakeholders are identified and described.

Stakeholder	Description of stakeholder	Expectations of product
Students	Primary users of the product. Effectiveness of product will have an immediate impact on students' knowledge of the topic.	Easy to understand, interesting, fun and engaging.
Industry	Needs engineers / students who have relevant skills in the field. Effectiveness of product has an impact on the availability of well-trained staff.	Informative, good preparation for the real world.
Teachers	Can implement the product to teach relevant skills. Effectiveness of the product has an impact on the quality of teaching.	Easy to implement into existing curricula, informative, high quality.

Society	This is an indirect stakeholder. Since products might be used in the real world (through industry or otherwise), users of the product will possibly affect this stakeholder.	Safe, ethical, unbiased.

Table 2: Stakeholders, their description and the expectation of the proposed product.

Learning Goals

By defining learning goals, it becomes clearer for both the designer and the users what an educational kit on (Tiny)ML might achieve. Learning goals can be used to guide the structuring of learning materials and can give a good overview of concepts to be covered. The learning goals are formulated as follows.

After the completion of the educational kit on (Tiny)ML, the user:

1. Understands the basic functionality of a neuron in a neural network.
2. Understands the basic network structure of a neural network, including the interconnectivity of neurons.
3. Understands the training algorithm used to tune a simple neural network.
4. Understands the inner workings of a neural network.
5. Can model the mathematics needed for calculating the prediction and training of a neural network.
6. Can implement a neural network in a programming language.
7. Can deploy a simple neural network to an edge device.
8. Can develop and deploy a neural network based on sensor input(s) independently.

Requirements

After the identification of stakeholders and learning goals, it is possible to determine design or product requirements. These can be split into two different categories: functional- and non-functional requirements. Functional requirements should describe what the product should do in a measurable way, while non-functional requirements define in what way the product should solve the problem [52].

Functional requirements

1. The content on TinyML that is included in the product is up to date.
2. The content on TinyML that is included in the product is accurate.
3. The content of the learning material covers the described Learning Goals.
4. Users of the product can design TinyML applications on their own after learning techniques using the product.
5. The product uses a mainstream programming language such as Python, Java or C++.
6. The product provides all necessary components to test and build various TinyML applications.
7. The instructions in the product are distributed in either a digital repository or in print.

8. The hardware included in the product follows all electrical safety guidelines.
9. The software used in the product is in line with the industry standards for software in TinyML.
10. The product links presented concepts to real life applications of TinyML.
11. The algorithms used in the product should not rely fully on platforms such as Edge Impulse [43].
12. The product should be affordable for the intended target audience. The current target price is below the average development board price as found in the Background Research - Chapter 2, €60.

Non-functional requirements

13. The product is suitable for students with an application level [3] knowledge of programming.
14. The product is fun and engaging to use.
15. The information presented in the contents of the product is divided into manageable sections.
16. Applications that are made by using (components of) the product are safe and ethical.
17. Machine learning is explained in a way which demystifies the common black-box principles.
18. The product supports multiple learning styles.
19. The presentation of the product is attractive and aesthetic.

MoSCoW Analysis

To prioritise the preliminary requirements drafted in the previous section and aid the selection of a concept, the requirements are ordered according to the MoSCoW methodology [53]. This method orders requirements according to four categories:

- **Must have**: the fundamental requirements, without them the product will not be successful and might not be safe to use.
- **Should have**: the product would benefit from implementation of these requirements, however it will work without them and be safe to use.
- **Could have:** these requirements would be nice to implement, however their impact on the final product is minimal.
- **Won't have**: while these requirements might be valid, due to the current objective of development or design, they will not be implemented in the product.

Must have	Should have	Could have	Won't have
1, 2, 3, 4, 7, 12, 13, 16, 18	5, 8, 9, 11, 14, 15	6, 10, 17, 19	-

Table 3: MoSCoW division of product requirements.

The division of the requirements, which can be seen in Table 3, was established as follows:

- Requirements 1 and 2 were categorised as Must have, because the field of TinyML is complex and quickly evolving, making outdated or inaccurate information detrimental to the product goals.
- Requirement 3 was categorised as Must have, because the Learning Goals are specifically designed to ensure comprehensive understanding and proficiency in key concepts.
- Requirement 4 was categorised as Must have, because empowering users to independently design TinyML applications is crucial for fostering creativity, innovation, and autonomy within the technology field.
- Requirement 5 was categorised as Should have, because the product and its subsequent users would benefit from having a mainstream programming language. However it would be possible to teach or learn another programming language to successfully use the product.
- Requirement 6 was categorised as Could have, because while providing all necessary components to test and build various TinyML applications is desirable, it may be subject to resource or time constraints during the implementation.
- Requirements 7 and 18 were categorised as Must have, because clear and accessible instructions are essential for users to effectively use the product and overcome potential obstacles or challenges. By providing instructions in either a digital repository or in print, the product ensures that users have multiple options to access the necessary guidance, catering to different preferences and situations.
- Requirement 8 was categorised as Should have, since ensuring the safety and preventing hazards is of foremost importance. Due to the nature of low-power electronics however, the importance is recognised and encouraged but not strictly necessary at this stage of the product.
- Requirements 9 and 11 were categorised as Should have, since the adherence to industry standards and algorithms (through platforms like Edge Impulse) is desirable, however it is important to remain flexible and compatible with a wide range of systems.
- Requirement 10 was categorised as Could have, because establishing connections between the presented concepts and real-life applications of TinyML is advantageous for enhancing practical understanding and fostering tangible use cases. However, while it is acknowledged, it may not be an essential aspect for the immediate functionality or success of the product.
- Requirement 12 was categorised as Must have, since the affordability of the product would determine the interest of the target group. As such, the price-point could determine the overall success of the product.
- Requirement 13 was categorised as Must have, since the intended target audience are specifically students with some programming experience.
- Requirement 14 was categorised as Should have, since a fun and engaging product could lead to more interest in and subsequently more success for the product.
- Requirement 15 was categorised as Should have, because structure can enhance the overall accessibility, comprehension and ease of learning for users, however it is not considered a crucial part of the product.

- Requirement 16 was categorised as Must have, because ensuring that the product is safe and ethical has the utmost importance for user well-being, legal compliance, and underscores the fundamental responsibility of machine learning developers.
- Requirement 17 was categorised as Could have, since the demystification of the common 'black-box' of machine learning might help some users conceptually, however, it does not impact the product much.
- Requirement 19 was categorised as Could have, because the product does not represent a final, marketable product and therefore does not have to rely upon aesthetics to sell.

Minimum Viable Product

After determining the stakeholders, learning goals and requirements, it is possible to define a Minimum Viable Product (MVP) according to the must-have requirements. The MVP is an implementation of the product which covers all necessary parameters to validate the effectiveness of the product [54]. In the MVP description, the requirements are be noted by their number, for example requirement 1 would be noted as (#1).

The MVP in this research is an educational kit on TinyML. The kit contains a microcontroller development board as well as a (digital) publication with educational material on TinyML. The microcontroller is custom design, since the price of conventional development boards capable of handling TinyML tasks is high for the intended target group (#12). Additionally, the construction of educational materials is easier if the designer has an intrinsic knowledge of the hardware.

The educational material was structured accordingly:
- Introduction to the field of TinyML; including current applications (#1)
- Machine learning basics; inputs and outputs, perceptrons (#17)
- Machine learning basics 2; networks, prediction (#17)
- Machine learning intermediate; training, network structures (#17)
- Machine learning advanced; classification types, optimization (#17)
- TinyML libraries; Tensorflow Lite Micro
- TinyML platforms: Edge Impulse
- Applications; real life example applications (#1).

These contents could be included as chapters in a book or digital publication (#7). All chapters should be accompanied by examples and projects that users can explore themselves (#4). The users should be able to program examples using the knowledge gained from the material and their prior programming experience (#3). To ensure the quality of materials, all content should be based on leading and current literature and software (#1, 2). The material should additionally contain insets on responsible and respectful Machine Learning development, as well as ethical guidelines (#15). A rendition of the proposed TinyML educational kit can be seen in Figure 3.

Figure 3: Rendition of the proposed TinyML educational kit.

5 Realisation

To properly evaluate the effects of tangible learning systems, combined with the requirements from Chapter 4, it was decided to design an educational kit, called TinySpark. The kit includes a development board and online platform for instructions, as described in previous chapters. As could be seen in the background research, there are already numerous development boards marketed as TinyML capable, as well as platforms that teach TinyML. However as discussed in Chapter 4, there are some valid points of improvement in terms of price, sensor integration, level of required pre-knowledge and non-proprietary of platforms.

Development Board

Price and sensor integration could be influenced during the design of the development board. Through careful component selection, the component price can be kept low, and the sensors that are integrated into the development board determine its capabilities.

After careful recall of the discovered requirements in the comparison of TinyML ready development boards, the main microcontroller chip was chosen. The requirements call for a chip that has a fast clock frequency as well as ample storage capacity. Additionally, it is beneficial to have many interfacing options such as Inter-Integrated Circuit (I2C), USB, Serial Peripheral Interface (SPI), Analog as well as Digital interfaces. Due to the ongoing chip shortage [55], it was also important to choose a chip that is widely available.

The decision was made to select a ESP32-S3 chip from Espressif [56]. The chip features two fast processing cores, high-speed storage options and broad selection of peripheral interfaces. In addition to this, the chip features a multitude of expansion options for future development, including Wi-Fi and Bluetooth capabilities, and AI acceleration using the manufacturers' own software platform. Moreover, Espressifs chips are regarded as versatile and very cost-effective. The features of the ESP32-S3 are recognised in the researchers' own positive experience with this chip architecture.

For the selection of sensors, the criteria were similar; common interfacing options such as I2C, as well as availability at a reasonable price. Because of the plethora of sensor options available, sample projects that should be executable using the development board were collected in the list below. Then, the selection was made to include as many sample projects as possible with available and cost–effective sensors.

The following sample projects were considered:

- Logic gates using physical inputs and outputs
- Vibration and Fall detection using motion detection
- Gesture detection using motion detection
- Wake word detection using sound decoding
- Weather prediction using environmental factor analysis
- Game intelligence for snake using logic
- Morse-code decoding using timing analysis
- Plant health monitoring using environmental factor analysis
- Room occupancy detection using environmental and ambient factor analysis
- Driving behaviour analysis using motion detection

- Smart lighting control using ambient factor analysis
- Food quality analysis using environmental factor analysis

The following sensors were selected to be included on the development board:

- LSM6DS3TR-C inertial motion sensor
- ICS-43434 microphone
- APDS-9930 light and distance sensor
- BME-280 environmental sensor
- AH-49E hall effect sensor (magnetic)
- H638T-TR2 infrared receiver

Additionally, some additional inputs and outputs were added, such as two user-programmable input buttons, one output LED and five addressable RGB LEDs. The connectivity to the development board was managed by the ESP32-S3s internal USB peripheral, and it was attached using an USB-C connector due to its widespread adoption. There were two more external connectors added, one for connecting Stemma QT and Qwiic sensors[16], and one for connecting generic sensors using standard digital and analog peripherals. Lastly, some passive components such as voltage regulators, capacitors and resistors were chosen. A full breakdown of components including the final product cost can be found in Appendix I. The final electronics schematics can be found below in Figure 4 and as a larger image in Appendix II.

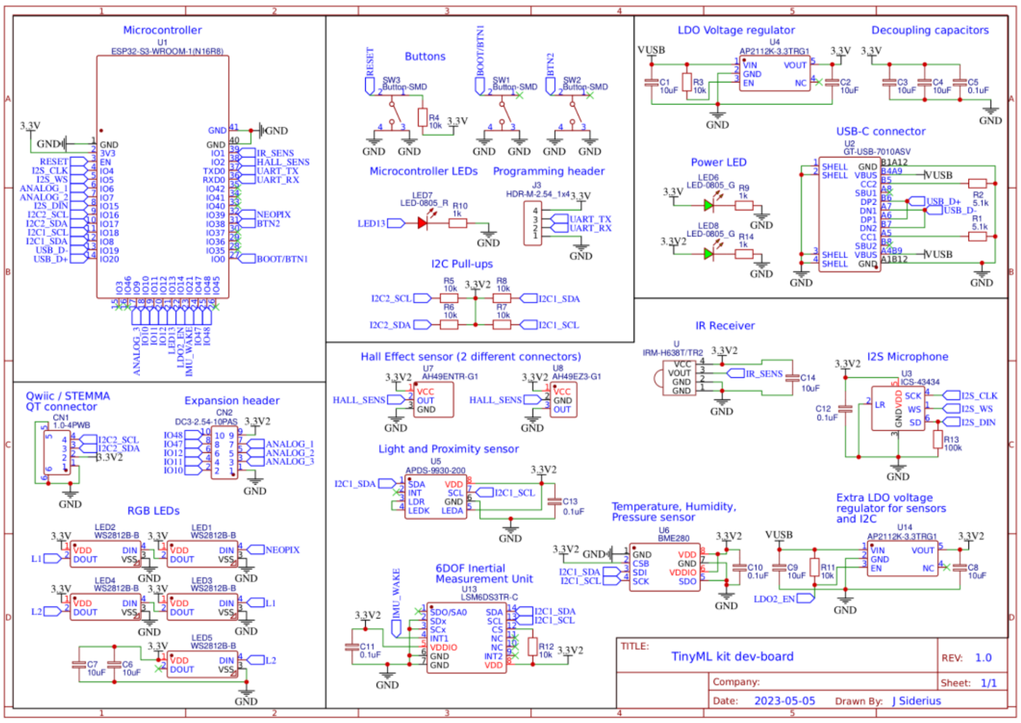


Figure 4: Development board electronics schematic.

[16] Stemma QT and Qwiic are connection standards developed by Adafruit Ltd. and Sparkfun respectively. They interface I2C connections through a proprietary connector, which is also implemented on the development board.

After selection of components, a printed circuit board (PCB) was designed. For this, considerations on the size and layout were explored in the PCB design software. Amongst others, it was considered how the development kit would be used during learning. Because all sensors and connectors are on-board of the PCB, the form factor was chosen to be a square, with all inputs and outputs positioned around the edges, and the microcontroller chip at the top (see Figure 5). All components are placed on the topside of the PCB, mostly in rows, to make assembly easier. The circuit board was complemented with explanatory text, as well as symbols depicting each sensors' main capability (e.g. a magnet symbol for the hall effect sensor).

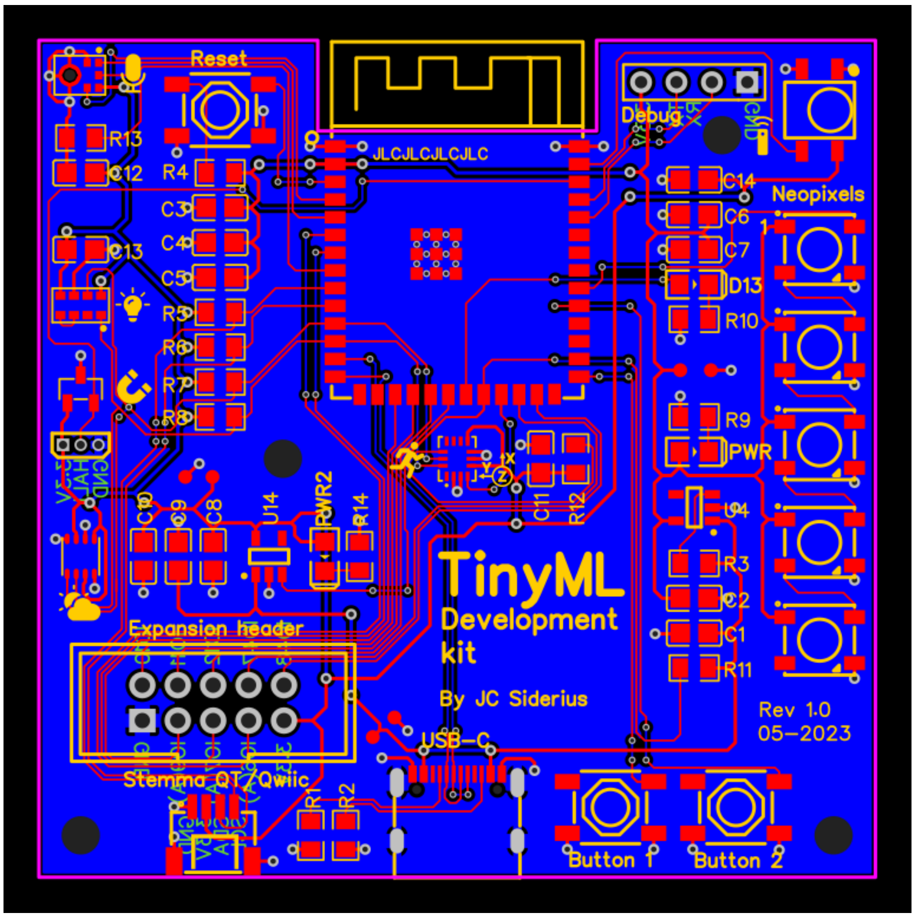


Figure 5: Development board, PCB layout.

After finalising the design of the circuit boards, they were put into production and the required components were ordered. Later, the PCBs were assembled by hand (due to the low volume), at home and in the SIL-EE lab at the University of Twente [57]. The circuit boards were pasted with soldering paste, after which all components were placed using tweezers. Lastly, the soldering paste was molten using a reflow oven. Some complications occurred with the soldering of the USB-C connectors, which have several small-pitched pins. After manual rework using a soldering station, all development boards were technically functional. Lastly, a 3D-printed base was made for each development board, to electrically isolate the underside of the circuit board, as well as to give the development board some bulk and stop it from moving around. See Figures 6 and 7 below for an impression of the finished development board.

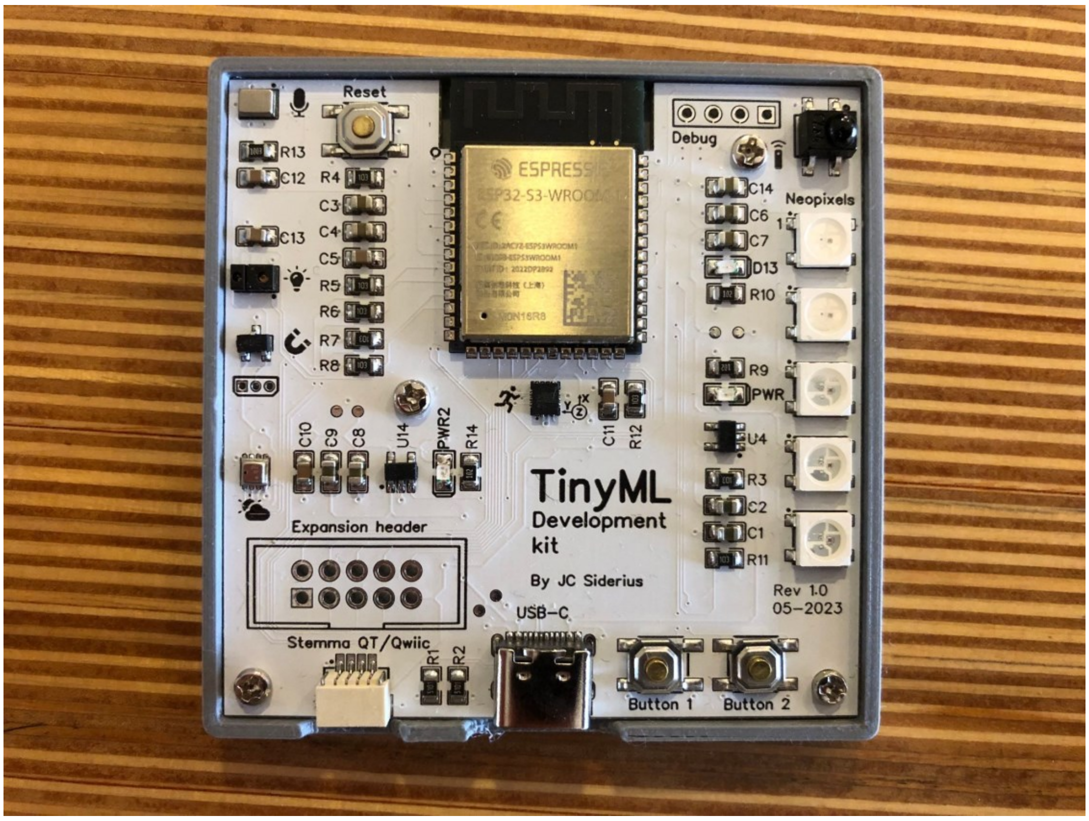


Figure 6: The finished development board, including the 3D printed base.

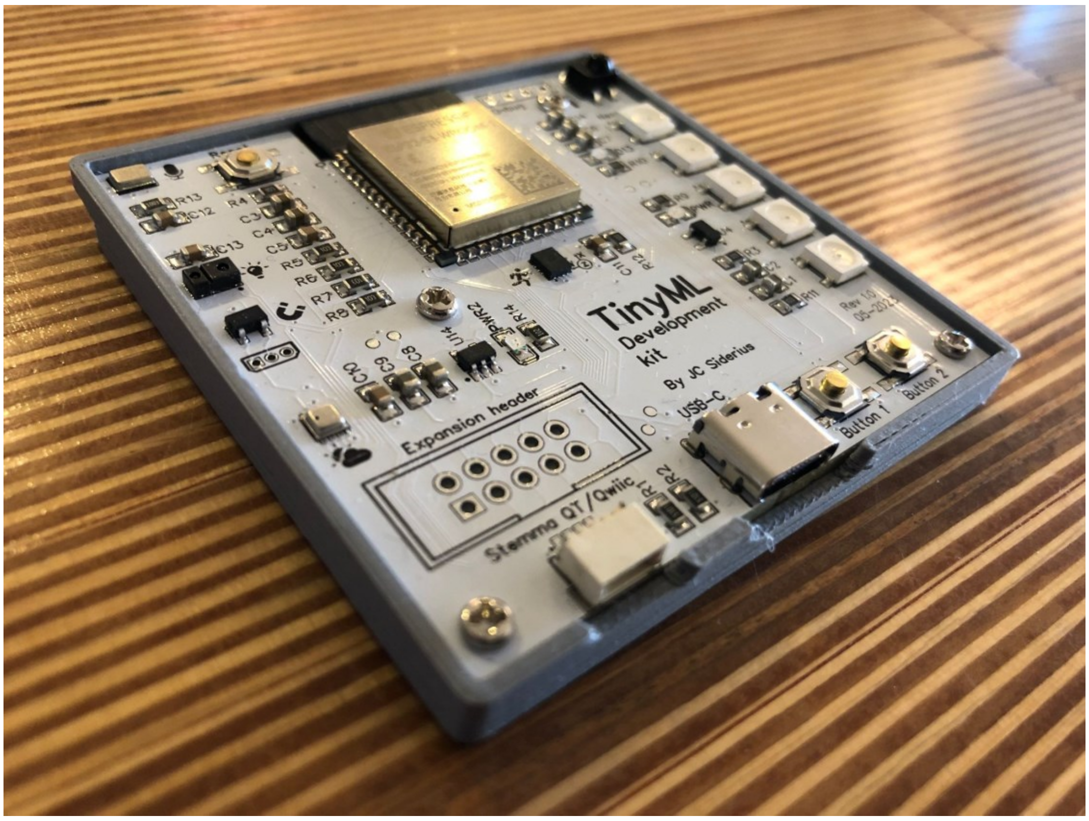


Figure 7: Detail of the finished development board.

The last element to be finished for the development board was the software. As discussed in Chapter 4 - MPV, it was important for the user to not require proprietary software to program the development board. After evaluation of the different available programming languages introduced in Chapter 2, CircuitPython was chosen for its ease of use and open ecosystem. The language offers good support for the on-board sensors on the designed development board and makes programming very easy for the end user through the USB functionality. Additionally, the Python language is used as a basis for CircuitPython, meaning that anyone with some experience with regular Python can quickly gain an understanding of the programming language.

To use the CircuitPython language, it was necessary to build firmware for the development board. The programming language requires the definition of all inputs and outputs available on the board, as well as setting the correct chip version and build settings such as processor frequency. It was also possible to include pre-made software libraries into the firmware. This enables easier development, since no external inclusions are needed to write software when the libraries are already internally stored. The custom firmware enables users to easily interface with all components on the development board and makes explaining of its functions clear. After the configuration was done, the firmware had to be built inside of a Linux virtual computer. The Adafruit build guide [58] was of major help. However, the Linux system as well as the code needed to be altered to make sure the firmware was compiled correctly. The firmware then needed to be installed on the development boards. This required some additional steps to properly configure the microcontrollers storage and registers. After this however, users could easily upload their own code to the development board, through the built-in USB-C connection.

For all but one sensor, software libraries were already pre-programmed. However, for the APDS9930 distance and light sensor, no CircuitPython-specific library was available. Due to this, a software library was developed for this sensor [59], including all functions and settings from the sensor. The library was written with the help of existing libraries from Adafruit as well as the datasheet for the respective sensor.

To evaluate all functionality, as well as provide reference code for the online platform, example code for every sensor, input and output was written. All five produced development boards were successfully tested and ready for evaluation by the test participants.

Online Platform

After a brief evaluation of all options for the delivery of learning materials and instructions, such as paper-based, a book or online, an online platform was chosen. Having material online means that additions and edits can be easily implemented where needed, code can be tested directly online and concepts can be explained using interactive visualisations. In addition to this, material is also conveniently accessible and can be adapted to different languages or teaching methods easily.

There is a plethora of available online platforms for hosting information or learning material. A selection of platforms was analysed in the Background Research in Chapter 2, after which the final platform was chosen. This was GitHub Pages [60] in combination with the static website template system MkDocs, and the graphic layer Material. The combination was picked for its simple setup, modern look and feel, as well as range of possibilities for integration with code plugins, mathematical notation systems and interactive elements. The integration with Github was furthermore especially useful, since all standard version control systems could now be used for easy synchronisation and error tracing.

Initially, the platform was set up as a placeholder, including a page which contained all sorts of useful pre-made structures, code-blocks, images and notations. This page was extensively used throughout the development of the online platform as a quick reference to the various options of the chosen platform.

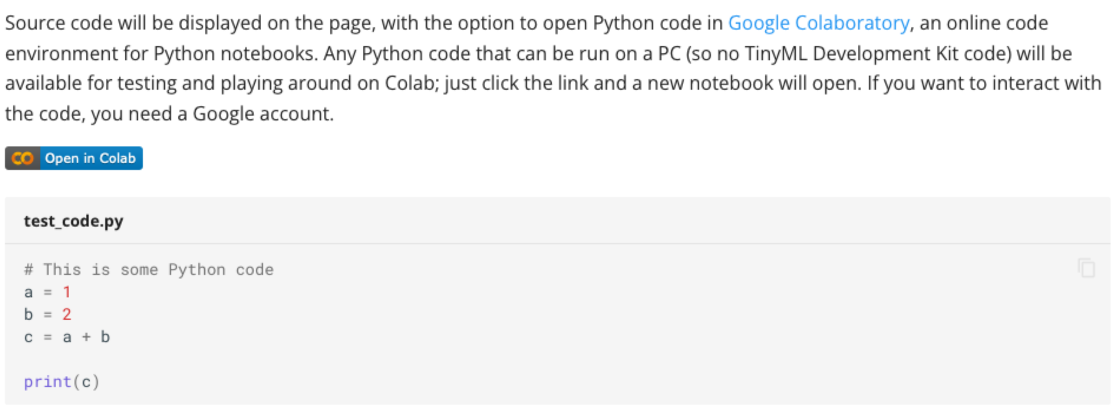
Then, the chapter structure of the learning material was selected, based on research on TinyML conducted in Chapter 2 and the requirements from Chapter 4. This structure can be seen in Table 4 below. Please note however that it was changed over the course of writing the educational material due to time constraints, this change is discussed in Chapter 7.

Chapter 1 - Introduction to neurons
Chapter 2 - Introduction to networks
Chapter 3 - Introduction to training
Chapter 4 - Larger models and input shaping
Chapter 5 - Optimization, compression and more projects

Table 4: Initial chapter structure of the online material.

To explain the underlying mathematics of various topics, a plugin was used to properly display mathematical formulas on the online platform. The MathJax library enables the use of LaTeX mathematical notation to render formatted formulas in every browser. Additionally, interactive elements were coded using the P5.js programming language. The P5.js framework allows easy graphical program making, using standard website elements such as sliders and frames.

While composing the learning material, it became apparent that it would be beneficial to include code right into the online platform. This would make for easy lookup of functions and enables users to quickly copy or change code. To facilitate this, all Python code used in the learning material is on Google Colaboratory, an online coding platform that can run Python programs directly in the browser. All CircuitPython code is hosted alongside the educational material on Github, since it cannot be run directly in the browser. The code is displayed inside of code blocks in the online platform, this is shown in Figure 8.


Source code will be displayed on the page, with the option to open Python code in Google Colaboratory, an online code environment for Python notebooks. Any Python code that can be run on a PC (so no TinyML Development Kit code) will be available for testing and playing around on Colab; just click the link and a new notebook will open. If you want to interact with the code, you need a Google account.

test_code.py

```
# This is some Python code
a = 1
b = 2
c = a + b

print(c)
```

Figure 8: Python code from Google Colaboratory displayed on the online platform.

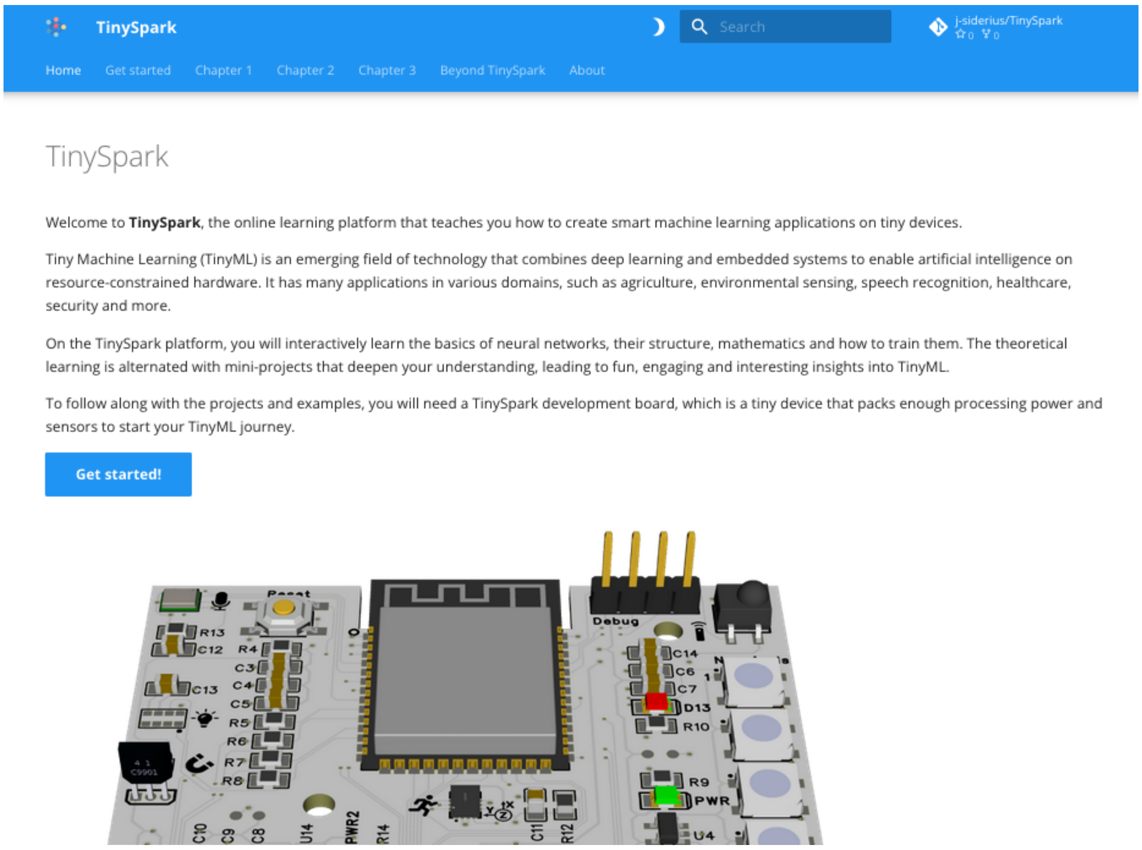
For each of the chapters of the online platform, an engaging example as well as a mini project was composed to keep the material interesting and engaging. This also ensures that the user can utilise the development board for their own applications, because they would gain familiarity with it during the learning process.

Due to time constraints, the chapter structure of the learning material was truncated at chapter 3. Although the following chapters would have contained interesting and relevant information, it was decided in consultation with the thesis supervisors to limit the amount of writing. This decision made it possible to focus more on the three main chapters. The content from chapters 4 and 5 was not lost however, as it was partially added to the recommended readings at the end of the online material. Additionally, possible further project ideas were also added, to stimulate users to implement some more applications on their own.

A precursory section of the online platform focuses on explaining the development board itself. Although it is assumed that users of the online platform have some prior experience with programming and development boards / embedded programming such as Arduino, the development board has some interesting and complex sensors and functions which need to be explained. The section was called 'Get started' hinting to the user that this would be the first point of contact with the material and development board. The section does not only include pointers and relevant information on the sensors of the development board, but also on programming and the workings of the online platform.

Lastly, the media on the online platform, including graphics and pictures, were unified in their styling according to a common colour scheme, to match the modern look and feel of the platform. Where possible, interactive elements such as neural network simulations were added to clarify concepts of the learning material. After the last chapter, a section was added that contained links to further learning material, machine learning frameworks and more mini-project ideas.

The online platform is currently available at https://web.archive.org/web/20230624094219/https://j-siderius.github.io/TinySpark/ (Archived June 2023). Some impressions of the TinySpark platform can be found in Figures 9, 10 and 11.



TinySpark

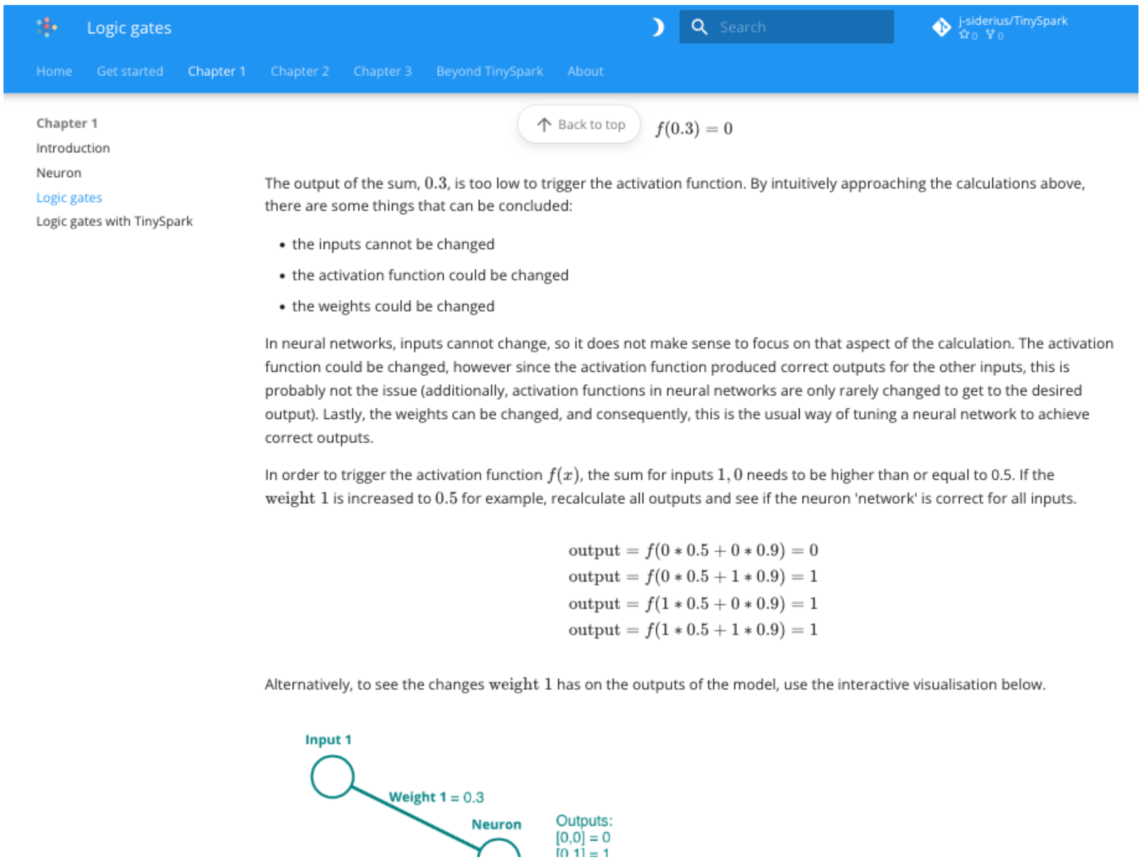
Welcome to **TinySpark**, the online learning platform that teaches you how to create smart machine learning applications on tiny devices.

Tiny Machine Learning (TinyML) is an emerging field of technology that combines deep learning and embedded systems to enable artificial intelligence on resource-constrained hardware. It has many applications in various domains, such as agriculture, environmental sensing, speech recognition, healthcare, security and more.

On the TinySpark platform, you will interactively learn the basics of neural networks, their structure, mathematics and how to train them. The theoretical learning is alternated with mini-projects that deepen your understanding, leading to fun, engaging and interesting insights into TinyML.

To follow along with the projects and examples, you will need a TinySpark development board, which is a tiny device that packs enough processing power and sensors to start your TinyML journey.

Figure 9: Landing page of the online platform.



$f(0.3) = 0$

The output of the sum, 0.3, is too low to trigger the activation function. By intuitively approaching the calculations above, there are some things that can be concluded:

- the inputs cannot be changed
- the activation function could be changed
- the weights could be changed

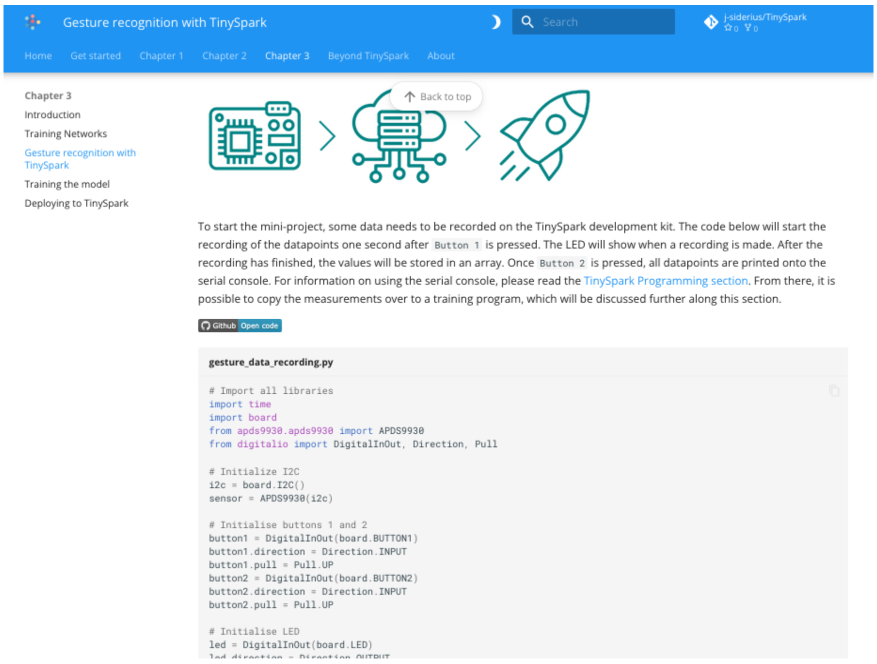
In neural networks, inputs cannot change, so it does not make sense to focus on that aspect of the calculation. The activation function could be changed, however since the activation function produced correct outputs for the other inputs, this is probably not the issue (additionally, activation functions in neural networks are only rarely changed to get to the desired output). Lastly, the weights can be changed, and consequently, this is the usual way of tuning a neural network to achieve correct outputs.

In order to trigger the activation function $f(x)$, the sum for inputs $1, 0$ needs to be higher than or equal to 0.5. If the $\text{weight } 1$ is increased to 0.5 for example, recalculate all outputs and see if the neuron 'network' is correct for all inputs.

$$\begin{aligned}\text{output} &= f(0 * 0.5 + 0 * 0.9) = 0\\ \text{output} &= f(0 * 0.5 + 1 * 0.9) = 1\\ \text{output} &= f(1 * 0.5 + 0 * 0.9) = 1\\ \text{output} &= f(1 * 0.5 + 1 * 0.9) = 1\end{aligned}$$

Alternatively, to see the changes $\text{weight } 1$ has on the outputs of the model, use the interactive visualisation below.

Figure 10: Excerpt from the first chapter of the online platform.



To start the mini-project, some data needs to be recorded on the TinySpark development kit. The code below will start the recording of the datapoints one second after `Button 1` is pressed. The LED will show when a recording is made. After the recording has finished, the values will be stored in an array. Once `Button 2` is pressed, all datapoints are printed onto the serial console. For information on using the serial console, please read the TinySpark Programming section. From there, it is possible to copy the measurements over to a training program, which will be discussed further along this section.

gesture_data_recording.py

```python
# Import all libraries
import time
import board
from apds9930.apds9930 import APDS9930
from digitalio import DigitalInOut, Direction, Pull

# Initialize I2C
i2c = board.I2C()
sensor = APDS9930(i2c)

# Initialise buttons 1 and 2
button1 = DigitalInOut(board.BUTTON1)
button1.direction = Direction.INPUT
button1.pull = Pull.UP
button2 = DigitalInOut(board.BUTTON2)
button2.direction = Direction.INPUT
button2.pull = Pull.UP

# Initialise LED
led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT
```

Figure 11: Example project and code on the online platform.

Educational Kit - TinySpark

Subsequently, the educational kit TinySpark was assembled. It comprises the development board, a USB-C cable and a quick-start guide (see Figure 12 below). The quick start guide gives an overview of the development board, including the sensors and connectors. Furthermore it links to the online learning platform that was presented in the previous section. The kit was packaged in a cardboard box and packing paper, to keep the packaging small and easy to open.

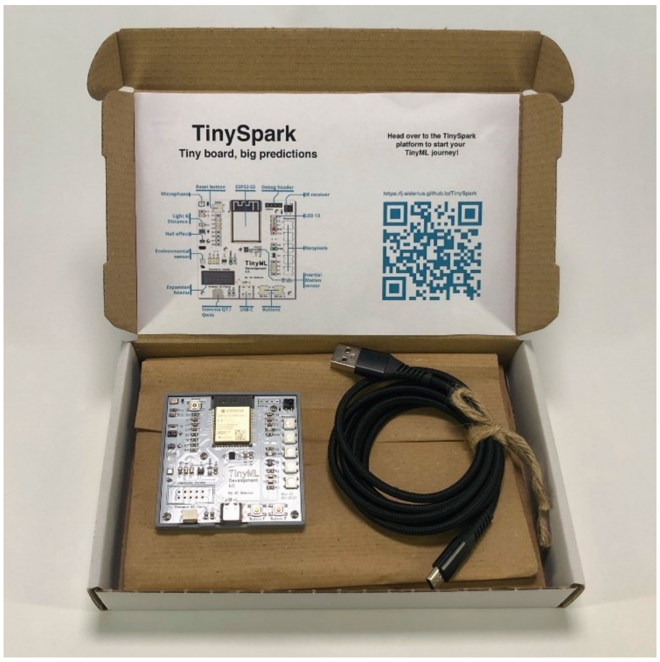

TinySpark
Tiny board, big predictions
Head over to the TinySpark platform to start your TinyML journey!

Figure 12: The educational kit TinySpark and its contents.

Testing

To evaluate the product, it was decided to perform qualitative user experience testing using a small pool of test participants. The testing procedure was designed to get a realistic impression of participants' interaction with the development board as well as the online learning platform.

Any user testing requires approval from the University of Twente Ethics Committee. To get a positive approval, all facets of ethics related to the user testing, such as personal data, recording methods, testing activities, burdens and risks, needed to be evaluated. Subsequently, a testing plan was drawn up and user information brochures as well as consent forms were created (see Appendices III and IV).

Because of the limited number of educational kits as well as available testing time, the number of participants evaluating the product would be restricted. In qualitative research participant selection is purposeful. In this product testing, only participants with some experience in programming were chosen to obtain valuable insight into possible improvements of the kit and (tangible) learning methods. However, the participants were not selected on their knowledge of (Tiny) Machine Learning.

Participants in the user experience testing received an educational kit and were asked to perform tasks involving the learning materials and development board. All needed information could be found on the online platform.

After the actual testing phase, the participants were invited to a semi-structured interview to analyse their experiences with the TinySpark kit. The interview questions can be found in Appendix V. First, an overview of the background knowledge of the participants on relevant topics was established. Then, the participants responded to questions about the development board, the online platform and the educational kit. Lastly, the participants were asked about the overall user experience, including improvements to the material or development board.

The interview questions were structured to maximise insight into the research questions posed in this research. Questions on the educational material incorporated in the online platform try to establish if the participants were able to comprehend and learn about complex topics as (Tiny)Machine learning. The participants were asked if the development board which applies hands-on learning had added value. Moreover, information was gathered on the effectiveness of adding tangibility and project-based learning. Furthermore, the questions about user experience gave insight into the level of engagement of the participants, and information on (dis-)advantages of the educational kit in comparison to literature or other online courses about TinyML.

6 Evaluation

In this chapter, the effectiveness and usability of the educational kit TinySpark that was developed during this research was thoroughly examined and assessed. The primary objective was to verify if the product covers the requirements and learning goals introduced in Chapter 4 - MVP Development. Additionally, the product will be assessed according to the Honeycomb UX evaluation framework laid out in Chapter 3 - Method. This chapter provides an in-depth analysis of the user testing outcome and presents the key findings derived from their experiences.

User testing

The interview after the user experience testing was split into multiple sections, as can be seen in Appendix V.

The initial impressions of the educational kit TinySpark as well as the response to the online learning platform were very positive. Several participants mentioned that the development board and packaging looked very professional and well-designed. Participants enjoyed the easy navigation around the online platform, as well as the clear descriptions. Overall, the initial impression was that the educational kit would be "a very useful and complete proof of concept to further develop" (Participant 2).

The learning material was evaluated next. The chapter explaining the concept of neurons was regarded as simple or basic, but effective. One participant said that "the material was very comprehensible, better than some [university] courses" (Participant 3). The second chapter, explaining the basics of networks and structures, was perceived as more difficult. Some participants did not understand the example presented here, however, the mathematics and theory were explained well. The interactive visualisation was insightful to play with, yet "impossible to solve [by hand], although I guess that is the point of machine learning" (Participant 3). The third and final chapter was perceived as fun and interesting, because of its more elaborate theory and examples. The material was "intuitive" and gave participants the opportunity to "experiment with programming my own network" (Participants 1 and 2).

Next, the participants were asked if they thought the content of the learning material was complete or should be complemented, several suggestions were made. One participant found the example in the second chapter of the material to be "somewhat confusing" (Participant 1). Another participant thought that adding "another chapter, with a more challenging project" (Participant 2) could make the experience even more attractive.

Additionally, participants were asked about advantages and disadvantages of the online learning platform. All participants mentioned that the interactive, online experience added to their "motivation to learn and progress through the material." Nevertheless, one suggestion was to change the code platform from Google Colaboratory to an embedded system, because "switching to an external code platform is cumbersome and breaks continuity" (Participant 1).

As defined in one of the requirements, participants were also asked about the impact of the black-box that surrounds machine learning and the apparent 'lifting' of that box in the learning material. Most participants found the chosen approach to "explain the complexities of a neural network better [than closed approaches]" (Participant 2) and one even said that the online material opened the way to a “faster transition into more complex systems for machine learning” (Participant 3).

The development board that accompanied and aided the learning materials was also assessed. Most participants found the board an asset to the experience, with the added benefit that “all sensors are on-board and ready to use” (Participant 2). One participant, however, mentioned that the development board was “not integrated and used sufficiently, with only few projects” (Participant 1), although at the same time, the participant mentioned that it was “fun to use”. When asked for their experience with the development board, all participants commended the ease of installation using CircuitPython, one stating “I wish all products were this easy to connect, just plug it in and it works” (Participant 3). On another note, two participants did experience a sensor inconsistency or failure when using the distance sensor and environmental sensor, respectively.

Last, the overall user experience was evaluated. Most participants found the experience to be “motivating” and “great” (Participant 1), with an emphasis on the “strength [of combining] theory and practice” (Participant 3). The online platform was “clean and modern” (Participant 2), which supported learning and made the experience “much better than [traditional] lectures” (Participant 3). The self-paced character of the material is also attributed to its engaging nature. Finally, most participants said that additional learning material on more complex topics as well as more mini-projects would “extend and enhance” the product.

Honeycomb UX framework

The user experience test was not only qualitatively analysed, but also with the help of the Honeycomb UX [51] framework that was introduced in Chapter 3 - Method. All seven aspects of the framework are discussed through the synthesis of interview answers from participants. Additionally, the analysis of the research gaps that were identified in Chapter 2 - Background is considered.

The product should be useful.

According to several user test participants, the educational kit offers relevant and practical knowledge regarding machine learning and TinyML. Additionally, the content was found to be engaging and aligned well with the defined learning goals. The modern approach to learning using an online platform which included interactivity added to this.

The product should be desirable.

The educational kit adds value to the learning experience concerning machine learning and TinyML. It included interactive visualisations on the online platform and the product presents theoretical concepts in accessible quantities according to participants. Furthermore, several participants mentioned that the educational kit was very appealing, both aesthetically and content wise. The kit motivated all participants to want to learn more.

The product should be accessible.

The educational kit presents the learning material in multiple ways, enabling various learning styles, which the participants appreciated. The physical accessibility of the online platform however leaves some room for adaptation. The pages are, for example, not particularly suitable for colour blind people.

The product should be credible.

The kit provides accurate and up-to-date information on machine learning and TinyML. Participants also recognised that there was a multitude of externally linked information included, which gave them the opportunity to explore topics further and find relevant additional information.

The product should be findable.

The development board included in the educational kit is accompanied by an extensive online guide. Most participants found this to be very helpful, however, some pointed out that not all sensors and peripherals of the development board were detailed enough. The navigation options as well as the search functionality available on the online platform were appreciated by all participants. Lastly, the chapter structure of the learning materials was found to be logical and well structured.

The product should be usable.

As mentioned in the UX trait above, all participants appreciated the ease of navigation of the online platform. Additionally, all participants were pleased with the connectivity and programming interface of the development board. The information about the development board available on the online platform however, was not always easily found by all participants.

The product should be valuable.

All participants recognised the added value of practical applications that the development board offered. Additionally, most participants responded positively to the interactive elements present on the online platform. One participant commented that, with some minor additional material, the product could be used as an educational tool.

The analysis of the Honeycomb UX framework indicates a positive overall user experience. The kit was found to be useful, offering relevant and practical knowledge while aligning well with the defined learning goals. Its desirability was evident, as participants expressed enthusiasm and eagerness to learn more, appreciating the appealing aesthetics and consistent presentation. Overall, the evaluation highlights the strengths and areas for improvement, providing valuable insights for further enhancing the user experience of the educational kit for learning TinyML. The discussion of areas of improvement can be found in Chapter 7- Discussion and Recommendations.

Learning Goals

The learning goals were evaluated by synthesising interview answers from the user testing participants. During the interviews, specific questions regarding the three main topics of learning were asked, and additional observations are derived from the interview answers. The observations are noted in the Notes section of the evaluation Table 5.

Learning goal	Notes	Achieved
1. The user understands the basic functionality of a neuron in a neural network.	All participants mentioned that the functionality of a neuron was properly explained and that the accompanying mathematics were understandable.	Yes
2. The user understands the basic network structure of a neural network, including the interconnectivity of neurons.	Several participants mentioned that the principles of the network structure were explained well and that the accompanying mathematics were understandable.	Yes
3. The user understands the training algorithm used to tune a simple neural network.	Several participants mentioned that the training algorithm was made very clear by inclusion of the mini-projects, and that the accompanying mathematics were understandable.	Yes
4. The user understands the inner workings of a neural network.	All participants mentioned that their understanding of neural networks increased, and that they could explain the structures within neural networks.	Yes
5. The user can model the mathematics needed for calculating the prediction and training of a neural network.	All participants mentioned that their understanding of neural networks increased, and that they could explain the mathematics needed for predictions within neural networks.	Yes
6. The user can implement a neural network in a programming language.	All participants implemented multiple (small) neural networks using the provided code examples in the Python programming language.	Yes

7. The user can deploy a simple neural network to an edge device.	All participants deployed multiple (small) neural networks to the educational kit development board.	Yes
8. The user can develop and deploy a neural network based on sensor input(s) independently.	Untested.	No

Table 5: Learning goals evaluation using the user testing interviews.

The evaluation of the learning goals demonstrates that all but one of the learning goals were adequately met in the educational material. Participants expressed a clear comprehension of the main theoretical concepts and were able to implement neural networks using the Python programming language. In addition, they successfully deployed neural networks to the educational kit development board, highlighting practical application on an edge device. While the independent development and deployment of a neural network based on sensor input remains untested due to time constraints, the evaluation highlights the educational kit's effectiveness in delivering comprehensive knowledge and practical skills in the field of TinyML.

Requirements

The evaluation of the functional requirements was performed after the product was fully finished and actively being tested by the user experience test participants. The testing of functional requirements was performed by either observing or measuring the product, or by evaluating the material on the online platform. The testing method is described in the Notes section of the evaluation Table 6. The requirements are ordered according to the MoSCoW method as stated in MVP Development in Chapter 4.

Requirement	**Notes**	**Achieved**
MoSCoW: Must have		
1. The content on TinyML that is included in the product is up-to-date.	Compared to recent literature the material is up-to-date.	Yes
2. The content on TinyML that is included in the product is accurate.	Compared to popular literature, the material is accurate.	Yes
3. The content of the learning material covers the described learning goals.	See the learning goal evaluation above for more details.	Yes
4. Users of the product can design TinyML applications on their own after learning techniques using the product.	The development board includes more on-board sensors and expansion options.	Yes

7. The instructions in the product are distributed in either a digital repository or in print.	The materials are available online.	Yes
12. The product should be affordable for the intended target audience. The target price is below the average development board price as found in the Background Research - Chapter 2, €60.	The total (material) cost of the educational kit is €21.15. (See Appendix I)	Yes
MoSCoW: Should have		
5. The product uses a mainstream programming language such as Python, Java or C++.	The product is programmed in Python and CircuitPython.	Yes
8. The hardware included in the product follows all electrical safety guidelines.	All hardware components were integrated using their provided and relevant design considerations. However, the development board was not subjected to an official electrical safety inspection.	Partially
9. The software used in the product is in line with the industry standards for software in TinyML.	The educational kit uses Python, which is one of the main languages in machine learning.	Partially
11. The algorithms used in the product should not rely fully on platforms such as Edge Impulse.	The algorithms used in the educational kit are completely coded from scratch, using only built-in Python libraries.	Yes
MoSCoW: Could have		
6. The product provides all necessary components to test and build various TinyML applications.	The development board contains many on-board sensors and has expansion options.	Yes
10. The product links presented concepts to real life applications of TinyML.	All mini-projects included in the learning material represent a real-life application of TinyML.	Yes

Table 6: Functional requirement evaluation according to the MoSCoW method.

The analysis of functional requirements shows that most of the requirements for the product have been achieved. While some requirements were partially met, the evaluation highlights the educational kit's strong alignment with functional needs, positioning it as a valuable resource for learning TinyML.

The non-functional requirements were evaluated through the synthesis of interview answers from user experience test participants. The observations are described in the Notes section of the evaluation Table 7. The requirements are again ordered according to the MoSCoW method as stated in MVP Development in Chapter 4.

Requirement	Notes	Achieved
MoSCoW: Must have		
13. The product is suitable for students with an application level knowledge of programming.	Several participants mentioned that the required programming knowledge was very minimal and accessible.	Yes
16. Applications that are made by using (components of) the product are safe and ethical.	Participants mentioned the inherent safety of the development board, with its built-in error handling. Ethics of the product were not discussed by the participants.	Partially
18. The product supports multiple learning styles.	Several participants mentioned that they liked the variety in explanation and experiments, for example the complex mathematics were supported by interactive visualisations.	Yes
MoSCoW: Should have		
14. The product is fun and engaging to use.	All participants mentioned it was fun to use the educational kit, and that they were engaged in the content.	Yes
15. The information presented in the contents of the product is divided into manageable parts.	Several participants mentioned that the division into chapters and sections helped oversee the different aspects of the learning material.	Yes

MoSCoW: Could have		
17. Machine learning is explained in a way which demystifies the common black-box principles.	All participants mentioned that it is best to learn Machine Learning without a black-box, which this material supports.	Yes
19. The presentation of the product is attractive and aesthetic.	All participants mentioned the clean and professional look of both the development board and online platform.	Yes

Table 7: Non-functional requirement evaluation according to the MoSCoW method.

From the evaluation of the non-functional requirements, it becomes clear that most of the drafted requirements were partially or fully implemented into the product. While some requirements were partially achieved, overall, the evaluation demonstrates the educational kit's effectiveness in addressing non-functional aspects and delivering an engaging and comprehensive learning experience.

7 Discussion and Recommendations

During the background research, design of the product, as well as the user testing, some problems were identified. Many of these were resolved during the design process and user experience testing. Nevertheless, it is important to recognize the boundaries and limitations of this research.

Background Research

Tangible educational methods were one of the explored topics of the literature review. It is important to note that the background research identified many different tangible methods, however, not all could be included in the background research due to the scope of work. The selection was based on the most relevant published work, meaning that less universally adopted, but valid and meaningful methods were disregarded.

Another facet of the background research was TinyML development boards. Due to the vastness of the electronics industry, it was only possible to index boards featured on popular marketplaces such as Adafruit and Digikey. However, this could mean that development boards built by less well-known companies and only sold in specialty stores might have been excluded from the overview. It is also important to note that development boards only tell a story of convenience, as there are many hundreds (or even thousands) of additional microcontroller chips available on the market. These chips might be purpose built for applications including machine learning, but due to their specific target audience or device, they are not broadly and conveniently available on development boards.

Lastly, in the evaluation of educational material available for TinyML, the exploration was limited to easily accessible and free teaching materials. Although general principles and techniques are covered, it is possible that there is more material available about this topic that might be enclosed in paid courses or even university and college courses. These were not included because of cost and time constraints.

Product - Educational Kit TinySpark

As discussed before, most of the design decisions regarding component selection for the development board were made based on availability and partly guided by the preference of the researcher. There were two points of attention for the selection of on-board sensors. The chosen microphone type (Inter Integrated Sound, I2S) was later found to not be compatible with the chosen programming language, CircuitPython. While there is active development on the integration of this type of microphone, there was no working integration at the moment of firmware compilation. Secondly, the chosen light and distance sensor had an inadequate performance during the measurement of distance. The sensor is originally designed for use in hand detection, in devices such as soap dispensers. Due to this, the distance sensing abilities are not as accurate as expected.

Next, the chosen online platform has some compromises. While the modern look and feel of the website certainly contribute to legibility and understanding, it can also decrease accessibility for people with disabilities. The main colour scheme as well as some diagrams are less suitable for people with colour blindness or reduced vision.

The coding platform itself, Google Colaboratory, reduces accessibility as a Google account is needed to evaluate code. In addition to this, the user also needs to leave the online platform to go to the Colaboratory code page to evaluate and change code.

Due to time constraints, two chapters in the learning material were excluded in the final product. The respective chapters would have covered *larger models and input shaping* and *Optimization and compression,* respectively. The first topic would have enabled users of the educational material to expand their gained theoretical knowledge into more extensive and capable prediction models. The second topic could give users the tools to employ more extensive machine learning models on their development board, through optimisation. Although these are both interesting and relevant subjects that need to be explored in future research, the core knowledge on (Tiny) machine learning was still adequately covered in the first three chapters.

The last observation concerns the use of teaching methods incorporated in the online learning platform. There was a lot of background research conducted on the most effective teaching methods in Chapter 2. During the writing of the learning material, the most important pointers of the findings were certainly included. However, all material could have been checked against the most important findings again, to ensure compliance with these guidelines.

Testing

The UX testing measured and assessed user satisfaction, user interaction and ease of use of the TinySpark development board as well as the online learning platform. The testing was conducted with a small number of participants. While this limits the number of responses gained from the user test, as one would get from quantitative testing, the qualitative approach meant that the results were expansive and insightful. Furthermore, due to the free and unsupervised exploration of the product that was given to the test participants, they were able to gather detailed observations on all aspects of the product. While the real-world settings where the product was evaluated were variable and somewhat uncontrolled, the observations were more natural and comprehensive. The semi-structured interviews allowed for more flexibility in participants' answers and comments.

Future recommendations

There are four elements that are interesting to consider for further research and testing. They concern extended background research, product alterations, learning material and testing development.

In the background research, tangible learning was one of the main pillars. One potential oversight was the inclusion of mainly popular teaching methods. In future research, it would be interesting to include different educational approaches, such as cooperative learning. Another valuable source of information that could not be included in this background research due to time constraints is the well-regarded book *Mindstorms* by Seymour Papert [61]. The author worked extensively on researching "learning theories" and has received praise for the inclusion of novel technologies into his teaching models.

The development board could also be altered. One of the chosen sensors, the microphone, could not be used fully due to software incompatibility. In addition, the included distance sensor was perceived as somewhat hard to work with. With more elaborate component selection and testing, a new revision of the development board could be produced. This could improve the overall quality of the experience users have with the educational kit.

As already mentioned in this chapter, some sections of the online learning material were excluded. Although the proposed topics were not of highest importance to gain a basic understanding of (Tiny) machine learning, they would have added valuable information. The whole learning experience might become more rounded, since important concepts for the expansion of machine learning models are discussed here. Additionally, the information contained in these chapters could enable users to explore more interesting applications using the development board.

Finally, the user experience testing of the current educational kit TinySpark could be extended. Even if the number of participants of the current user experience test is relatively small, valuable insights into their experiences and behaviour and the usability of the product have been gained. If the user experience testing would be expanded and the number of participants increased, quantitative data could be analysed to further support the findings and in-depth insight of this user experience test. Additionally, it would be recommendable to choose participants from different backgrounds for testing. For example, it would be good to include people without extensive programming knowledge. By doing this, the evaluation would be conducted on many different levels of understanding and pre-existing knowledge, which could further expose possible unaccounted facets of the learning material and development board.

8 Conclusion

TinyML is an interesting and fast-growing field of AI. The technology offers many useful opportunities for numerous industry sectors as well as consumer products. As shown in the State-of-the-Art research, there exist very little learning materials covering TinyML. In a society that is adopting AI to an increasingly impactful extent, it should be obvious that the engineers of the future must have enough knowledge on the topic, to make better decisions.

Research into educational methods has shown that adding tangible aspects to learning materials enriches the obtained knowledge and increases engagement. Project based learning was also identified as being effective in gaining deep understanding of subject matter and developing problem-solving skills. By incorporating tangible and project-based learning in educational methods, the learning process empowers students to master complex topics like TinyML (Sub Research Question 1).

To test the hypothesis that tangible learning would benefit students, an educational kit was developed and produced. The TinySpark kit and accompanying online platform provide easy opportunities for learning and interaction with real world contexts. The educational kit offers great interactivity and user experience testing shows that the kit is a valuable addition to the learning material.

The analysis of educational kits aimed at teaching complex topics to university students revealed that the TinySpark kit not only met all the necessary requirements, but also proved to be highly engaging, professional, user-friendly, comprehensible, and provided an excellent foundation for learning TinyML. The positive outcomes from the user experience test validate the suitability of these educational kits for effectively transferring knowledge and facilitating comprehension of complex topics. With their engaging and accessible nature, these kits hold significant potential for enhancing the teaching and learning experience in the field of TinyML (Sub Research Question 2).

The overall conclusion is that this research demonstrates that educational kits and project-based learning are highly effective in teaching students TinyML (Research Question). The exploration of these approaches revealed multiple ways in which they can contribute to the learning process, including enhancing engagement, promoting hands-on experience, fostering problem-solving skills, and facilitating a deeper understanding of TinyML concepts.

The effectiveness of educational kits for teaching TinyML holds significant relevance within the field of AI. Society is increasingly adopting AI technologies, prompting a heated debate about AI as many people fear the role it may play in our future. Therefore it will be of utmost importance to understand this new technology well. By showing the potential of educational kits and project-based learning, this research offers valuable insights and practical solutions for educators as well as the industry, enabling them to enhance the learning experience and equip students with the necessary competencies in the rapidly evolving field of TinyML.

It is evident that TinyML and related technologies are not going to slow down or disappear. Thus, as educators, researchers, and policymakers, it is crucial to incorporate innovative approaches in educational materials regarding these technologies. By doing so, it is possible to empower the current and future generations of learners to drive innovation even further and make a positive contribution to our future.

References

[1] 'Machine learning education', *TensorFlow*. https://www.tensorflow.org/resources/learn-ml (accessed Mar. 17, 2023).

[2] 'The developer-first edge ML platform', *Edge Impulse*. https://www.edgeimpulse.com/product (accessed Mar. 17, 2023).

[3] P. Armstrong, 'Bloom's Taxonomy', *Vanderbilt University*, 2010. https://cft.vanderbilt.edu/guides-sub-pages/blooms-taxonomy/ (accessed Mar. 17, 2023).

[4] K. Peppler, *The SAGE Encyclopedia of Out-of-School Learning*, 2 vols. Thousand Oaks,, California, 2017. doi: 10.4135/9781483385198.

[5] M. S. Horn, R. J. Crouser, and M. U. Bers, 'Tangible interaction and learning: the case for a hybrid approach', *Pers. Ubiquitous Comput.*, vol. 16, no. 4, pp. 379–389, Apr. 2012, doi: 10.1007/s00779-011-0404-2.

[6] S. Price, Y. Rogers, M. Scaife, D. Stanton, and H. Neale, 'Using "tangibles" to promote novel forms of playful learning', *Interact. Comput.*, vol. 15, no. 2, pp. 169–185, Apr. 2003, doi: 10.1016/S0953-5438(03)00006-7.

[7] S. Matthews, S. Viller, and M. A. Boden, '"... and we are the creators!" Technologies as Creative Material', in *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, in TEI '20. New York, NY, USA: Association for Computing Machinery, Feb. 2020, pp. 511–518. doi: 10.1145/3374920.3374980.

[8] M. Resnick and B. Silverman, 'Some reflections on designing construction kits for kids', in *Proceedings of the 2005 conference on Interaction design and children*, Boulder Colorado: ACM, Jun. 2005, pp. 117–122. doi: 10.1145/1109540.1109556.

[9] T. Bekker, S. Bakker, I. Douma, J. Van Der Poel, and K. Scheltenaar, 'Teaching children digital literacy through design-based learning with digital toolkits in schools', *Int. J. Child-Comput. Interact.*, vol. 5, pp. 29–38, Sep. 2015, doi: 10.1016/j.ijcci.2015.12.001.

[10] G. Jones, L. Robertson, G. E. Gardner, S. Dotger, and M. R. Blanchard, 'Differential Use of Elementary Science Kits', *Int. J. Sci. Educ.*, vol. 34, no. 15, pp. 2371–2391, Oct. 2012, doi: 10.1080/09500693.2011.602755.

[11] D. Stanton *et al.*, 'Classroom collaboration in the design of tangible interfaces for storytelling', in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Seattle Washington USA: ACM, Mar. 2001, pp. 482–489. doi: 10.1145/365024.365322.

[12] P. Marshall, 'Do tangible interfaces enhance learning?', in *Proceedings of the 1st international conference on Tangible and embedded interaction*, Baton Rouge Louisiana: ACM, Feb. 2007, pp. 163–170. doi: 10.1145/1226969.1227004.

[13] O. Zuckerman, S. Arida, and M. Resnick, 'Extending tangible interfaces for education: digital montessori-inspired manipulatives', in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Portland Oregon USA: ACM, Apr. 2005, pp. 859–868. doi: 10.1145/1054972.1055093.

[14] S. Somyürek, 'An effective educational tool: construction kits for fun and meaningful learning', *Int. J. Technol. Des. Educ.*, vol. 25, no. 1, pp. 25–41, Feb. 2015, doi: 10.1007/s10798-014-9272-1.

[15] M. S. Horn, E. T. Solovey, R. J. Crouser, and R. J. K. Jacob, 'Comparing the use of tangible and graphical programming languages for informal science education', in *Proceedings of the*

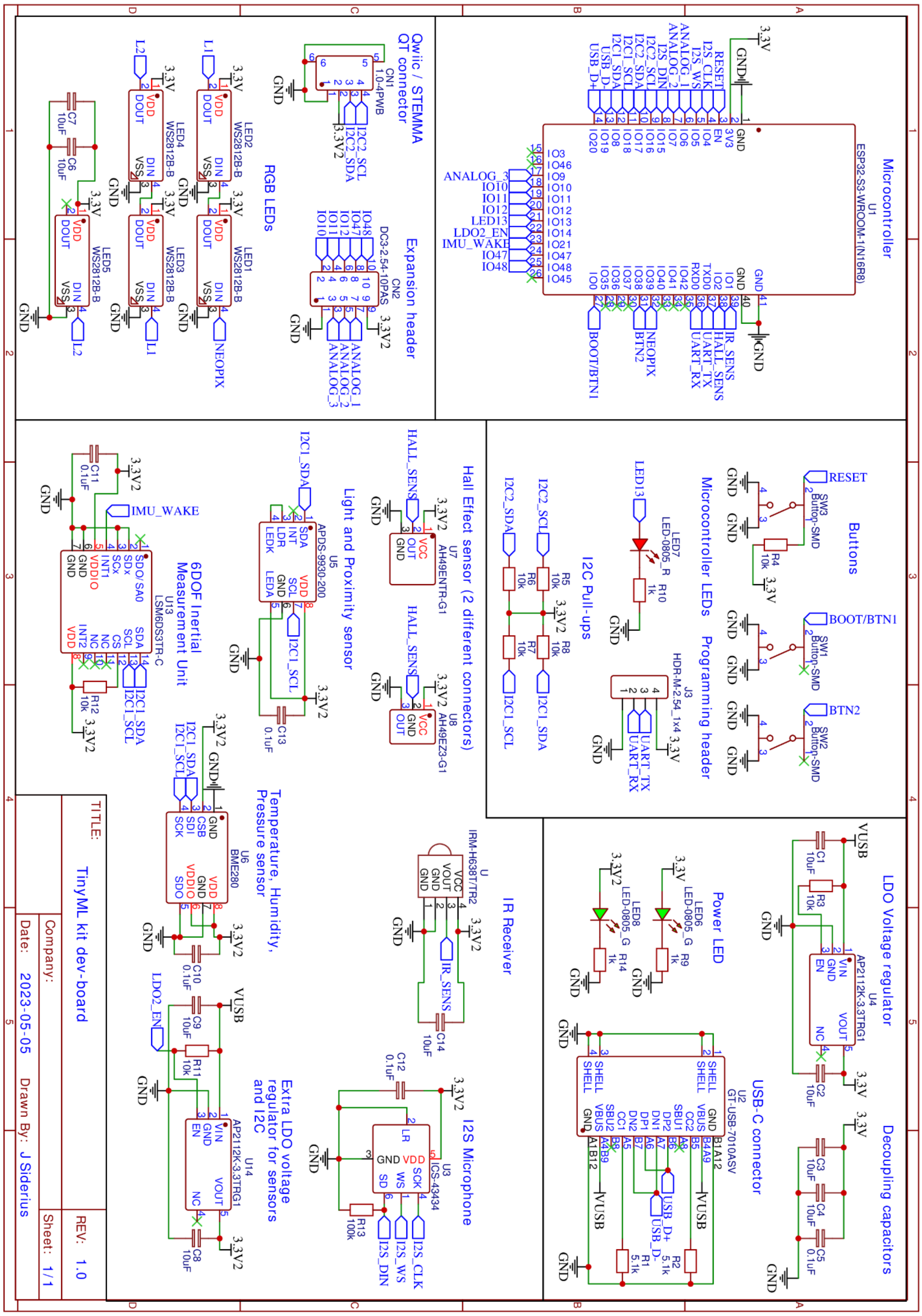
SIGCHI Conference on Human Factors in Computing Systems, Boston MA USA: ACM, Apr. 2009, pp. 975–984. doi: 10.1145/1518701.1518851.
[16] M. S. Horn, 'Tangible Interaction and Cultural Forms: Supporting Learning in Informal Environments', *J. Learn. Sci.*, vol. 27, no. 4, pp. 632–665, Oct. 2018, doi: 10.1080/10508406.2018.1468259.
[17] J. P. Oliver and F. Haim, 'Lab at Home: Hardware Kits for a Digital Design Lab', *IEEE Trans. Educ.*, vol. 52, no. 1, pp. 46–51, Feb. 2009, doi: 10.1109/TE.2008.917191.
[18] D. Dickerson, M. Clark, K. Dawkins, and C. Horne, 'Using science kits to construct content understandings in elementary schools', *J. Elem. Sci. Educ.*, vol. 18, no. 1, pp. 43–56, Oct. 2006, doi: 10.1007/BF03170653.
[19] W. S. McCulloch and W. Pitts, 'A logical calculus of the ideas immanent in nervous activity', *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943, doi: 10.1007/BF02478259.
[20] A. M. Turing, 'Computing Machinery and Intelligence', *Mind*, vol. LIX, no. 236, pp. 433–460, Oct. 1950, doi: 10.1093/mind/LIX.236.433.
[21] F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. in Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957. [Online]. Available: https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf
[22] A. L. Samuel, 'Some Studies in Machine Learning Using the Game of Checkers', *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, Jul. 1959, doi: 10.1147/rd.33.0210.
[23] T. Cover and P. Hart, 'Nearest neighbour pattern classification', *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, doi: 10.1109/TIT.1967.1053964.
[24] H. P. Moravec, 'Obstacle avoidance and navigation in the real world by a seeing robot rover', 1980. Accessed: Apr. 14, 2023. [Online]. Available: https://www.ri.cmu.edu/pub_files/pub4/moravec_hans_1980_1/moravec_hans_1980_1.pdf
[25] T. Sejnowski and C. R. Rosenberg, 'NETtalk: a parallel network that learns to read aloud', 1988. Accessed: Apr. 14, 2023. [Online]. Available: https://www.semanticscholar.org/paper/NETtalk%3A-a-parallel-network-that-learns-to-read-Sejnowski-Rosenberg/406033f22b6a671b94bcbdfaf63070b7ce6f3e48
[26] B. Weber, 'Swift and Slashing, Computer Topples Kasparov', *The New York Times*, May 12, 1997. Accessed: Apr. 14, 2023. [Online]. Available: https://www.nytimes.com/1997/05/12/nyregion/swift-and-slashing-computer-topples-kasparov.html
[27] M. Hale, 'Actors and Their Roles for $300, HAL? HAL!', *The New York Times*, Feb. 08, 2011. Accessed: Apr. 10, 2023. [Online]. Available: https://www.nytimes.com/2011/02/09/arts/television/09nova.html
[28] 'AlphaGo: Mastering the ancient game of Go with Machine Learning', Jan. 27, 2016. https://ai.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html (accessed Apr. 10, 2023).
[29] 'TensorFlow Core', *TensorFlow*. https://www.tensorflow.org/overview (accessed Apr. 14, 2023).
[30] 'PyTorch'. https://www.pytorch.org (accessed Apr. 14, 2023).
[31] 'On-Device Machine Learning', *Google Developers*. https://developers.google.com/learn/topics/on-device-ml (accessed Apr. 14, 2023).

[32] S. Han, H. Mao, and W. J. Dally, ‘Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding’, arXiv, Feb. 2016. doi: 10.48550/arXiv.1510.00149.
[33] ‘Global Microcontroller Market Size, Share & Trends Analysis Report 2021-2028 - ResearchAndMarkets.com’, Oct. 13, 2021. https://www.businesswire.com/news/home/20211013005793/en/Global-Microcontroller-Market-Size-Share-Trends-Analysis-Report-2021-2028---ResearchAndMarkets.com (accessed Apr. 14, 2023).
[34] ‘Amazon Echo Dot 5’, *Amazon*. https://www.amazon.com/dp/B09B8V1LZ3/ (accessed Apr. 14, 2023).
[35] ‘Fitness Trackers | Shop Fitbit’. https://www.fitbit.com/global/nl/products/trackers (accessed Apr. 14, 2023).
[36] J. Bier, ‘What’s Driving AI and Vision to the Edge’, *EE Times Asia*, Sep. 08, 2020. https://www.eetasia.com/whats-driving-ai-and-vision-to-the-edge/ (accessed Apr. 14, 2023).
[37] A. Kumar, ‘Resource-efficient Machine Learning in 2 KB RAM for the Internet of Things’, 2017, [Online]. Available: https://www.microsoft.com/en-us/research/uploads/prod/2017/06/kumar17.pdf
[38] D. A. Huffman, ‘A Method for the Construction of Minimum-Redundancy Codes’, *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952, doi: 10.1109/JRPROC.1952.273898.
[39] ‘TensorFlow Lite for Microcontrollers - Experiments with Google’. https://experiments.withgoogle.com/collection/tfliteformicrocontrollers (accessed Apr. 11, 2023).
[40] ‘STM32 Microcontrollers (MCUs) - STMicroelectronics’. https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html (accessed Apr. 14, 2023).
[41] ‘X-CUBE-AI - AI expansion pack for STM32CubeMX - STMicroelectronics’. https://www.st.com/en/embedded-software/x-cube-ai.html (accessed Apr. 14, 2023).
[42] ‘Arduino IDE software’, *Arduino*. https://www.arduino.cc/en/software (accessed Apr. 14, 2023).
[43] ‘Edge Impulse DSP and Inferencing SDK’. Edge Impulse, Apr. 12, 2023. Accessed: Apr. 14, 2023. [Online]. Available: https://github.com/edgeimpulse/inferencing-sdk-cpp
[44] E. Odunlade, ‘Top 10 Popular Microcontrollers Among Makers’, *Electronics-Lab.com*, Jun. 19, 2020. https://www.electronics-lab.com/top-10-popular-microcontrollers-among-makers/ (accessed Apr. 14, 2023).
[45] ‘Applications of TinyML | edX’. https://www.edx.org/course/applications-of-tinyml (accessed Apr. 14, 2023).
[46] S. Mistry and D. Pajak, ‘Get Started With Machine Learning on Arduino | Arduino Documentation | Arduino Documentation’, *Arduino*. https://docs.arduino.cc/tutorials/nano-33-ble-sense/get-started-with-machine-learning (accessed Apr. 14, 2023).
[47] ‘Edge Impulse libraries’. https://docs.edgeimpulse.com/docs/deployment/running-your-impulse-locally (accessed Jun. 23, 2023).
[48] ‘OpenMV MicroPython language reference’. https://docs.openmv.io/reference/index.html (accessed Jun. 23, 2023).
[49] Adobe Communications Team, ‘Waterfall Methodology: Project Management’, *Adobe*, Mar. 18, 2022. https://business.adobe.com/blog/basics/waterfall (accessed Apr. 16, 2023).

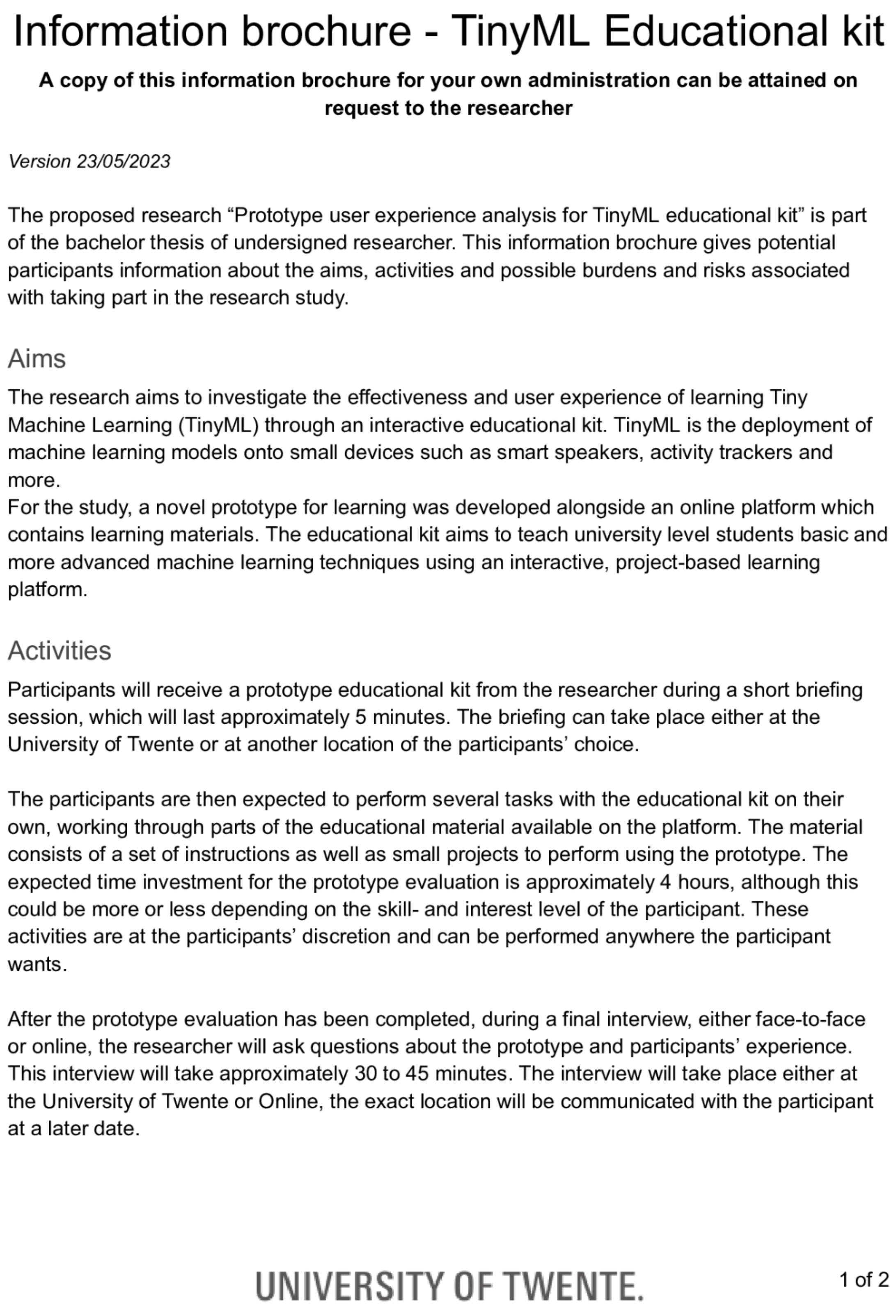
[50] Monday.com Team, ‘Waterfall Methodology’, *Monday.com*, Jan. 01, 2023. https://monday.com/blog/project-management/waterfall-methodology/ (accessed Apr. 17, 2023).
[51] P. Morville, ‘User Experience Design - Honeycomb UX framework’, *Semantic Studios*, Jun. 21, 2004. https://semanticstudios.com/user_experience_design/ (accessed Apr. 16, 2023).
[52] ‘A Guide to Functional Requirements’, *Nuclino*. https://www.nuclino.com/articles/functional-requirements (accessed Apr. 17, 2023).
[53] *DSDM Project Framework*. DSDM Consortium, 2014. Accessed: Apr. 17, 2023. [Online]. Available: https://www.agilebusiness.org/dsdm-project-framework.html
[54] E. Ries, ‘Minimum Viable Product: a guide’, *Startup Lessons Learned*, Aug. 03, 2009. https://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html (accessed Apr. 17, 2023).
[55] S. Deshpande, ‘Supply chain issues and autos: When will the chip shortage end?’, Apr. 18, 2023. https://www.jpmorgan.com/insights/current-events/supply-chain/supply-chain-chip-shortage (accessed Jun. 24, 2023).
[56] ‘ESP32-S3 Wi-Fi & Bluetooth 5 (LE) MCU | Espressif Systems’. https://www.espressif.com/en/products/socs/esp32-s3 (accessed Jun. 23, 2023).
[57] ‘SIL-EE lab information’, *University of Twente*. https://sil-ee.wiki.utwente.nl/ (accessed Jun. 24, 2023).
[58] D. Halbert, ‘Building CircuitPython | Adafruit Learning System’, *Adafruit*, Jun. 02, 2023. https://learn.adafruit.com/building-circuitpython/introduction (accessed Jun. 24, 2023).
[59] J. Siderius, ‘CircuitPython APDS9930 software library’. Jun. 07, 2023. Accessed: Jun. 24, 2023. [Online]. Available: https://github.com/j-siderius/CircuitPython_APDS9930
[60] ‘GitHub Pages’, *GitHub Pages*. https://pages.github.com/ (accessed Jun. 24, 2023).
[61] S. Papert, *Mindstorms: children, computers, and powerful ideas*, 2nd ed. New York: Basic Books, 1993.

Appendices

Appendix I - Development board schematic


Microcontroller
U1 ESP32-S3-WROOM-1(N16R8)
Qwiic / STEMMA QT connector
Expansion header
RGB LEDs
Buttons
Microcontroller LEDs
Programming header
I2C Pull-ups
Hall Effect sensor (2 different connectors)
Light and Proximity sensor
6DOF Inertial Measurement Unit
LDO Voltage regulator
Decoupling capacitors
USB-C connector
Power LED
IR Receiver
I2S Microphone
Temperature, Humidity, Pressure sensor
Extra LDO voltage regulator for sensors and I2C
TITLE: TinyML kit dev-board
REV: 1.0
Company:
Date: 2023-05-05
Drawn By: J Siderius
Sheet: 1/1

Appendix II - Information brochure



Information brochure - TinyML Educational kit

A copy of this information brochure for your own administration can be attained on request to the researcher

Version 23/05/2023

The proposed research “Prototype user experience analysis for TinyML educational kit” is part of the bachelor thesis of undersigned researcher. This information brochure gives potential participants information about the aims, activities and possible burdens and risks associated with taking part in the research study.

Aims

The research aims to investigate the effectiveness and user experience of learning Tiny Machine Learning (TinyML) through an interactive educational kit. TinyML is the deployment of machine learning models onto small devices such as smart speakers, activity trackers and more.
For the study, a novel prototype for learning was developed alongside an online platform which contains learning materials. The educational kit aims to teach university level students basic and more advanced machine learning techniques using an interactive, project-based learning platform.

Activities

Participants will receive a prototype educational kit from the researcher during a short briefing session, which will last approximately 5 minutes. The briefing can take place either at the University of Twente or at another location of the participants’ choice.

The participants are then expected to perform several tasks with the educational kit on their own, working through parts of the educational material available on the platform. The material consists of a set of instructions as well as small projects to perform using the prototype. The expected time investment for the prototype evaluation is approximately 4 hours, although this could be more or less depending on the skill- and interest level of the participant. These activities are at the participants’ discretion and can be performed anywhere the participant wants.

After the prototype evaluation has been completed, during a final interview, either face-to-face or online, the researcher will ask questions about the prototype and participants’ experience. This interview will take approximately 30 to 45 minutes. The interview will take place either at the University of Twente or Online, the exact location will be communicated with the participant at a later date.

UNIVERSITY OF TWENTE.

1 of 2

Burdens and Risks

The expected burdens of the study on the participants include time investment to complete the tasks using the prototype, as well as time for the final interview. No monetary burden will be placed upon participants, the prototype educational kit will be provided free of charge for the duration of the study, and the educational material is freely available online.

There are no expected risks associated with the research study. During the study, participants are not expected to encounter any dangerous, upsetting or otherwise disturbing situations or content.

Expectations

Participants will be expected to perform tasks in the educational material using the prototype educational kit seriously. The participant is expected to answer questions about their experience evaluating the prototype truthfully during the final interview (this of course excluded any questions that the participant might not want to answer). Lastly, participants are expected to treat the prototype with care during their use and return it in the same condition to the researcher in which they received it.

Consent

Once a participant decides to participate in the research study, they will be asked to fully read, understand and sign a consent form provided by the researcher. It is important to note that participants can decide to withdraw from the research at any time without explanation or justification. This will not incur any burdens onto the participant, and will have no negative impact on them whatsoever.

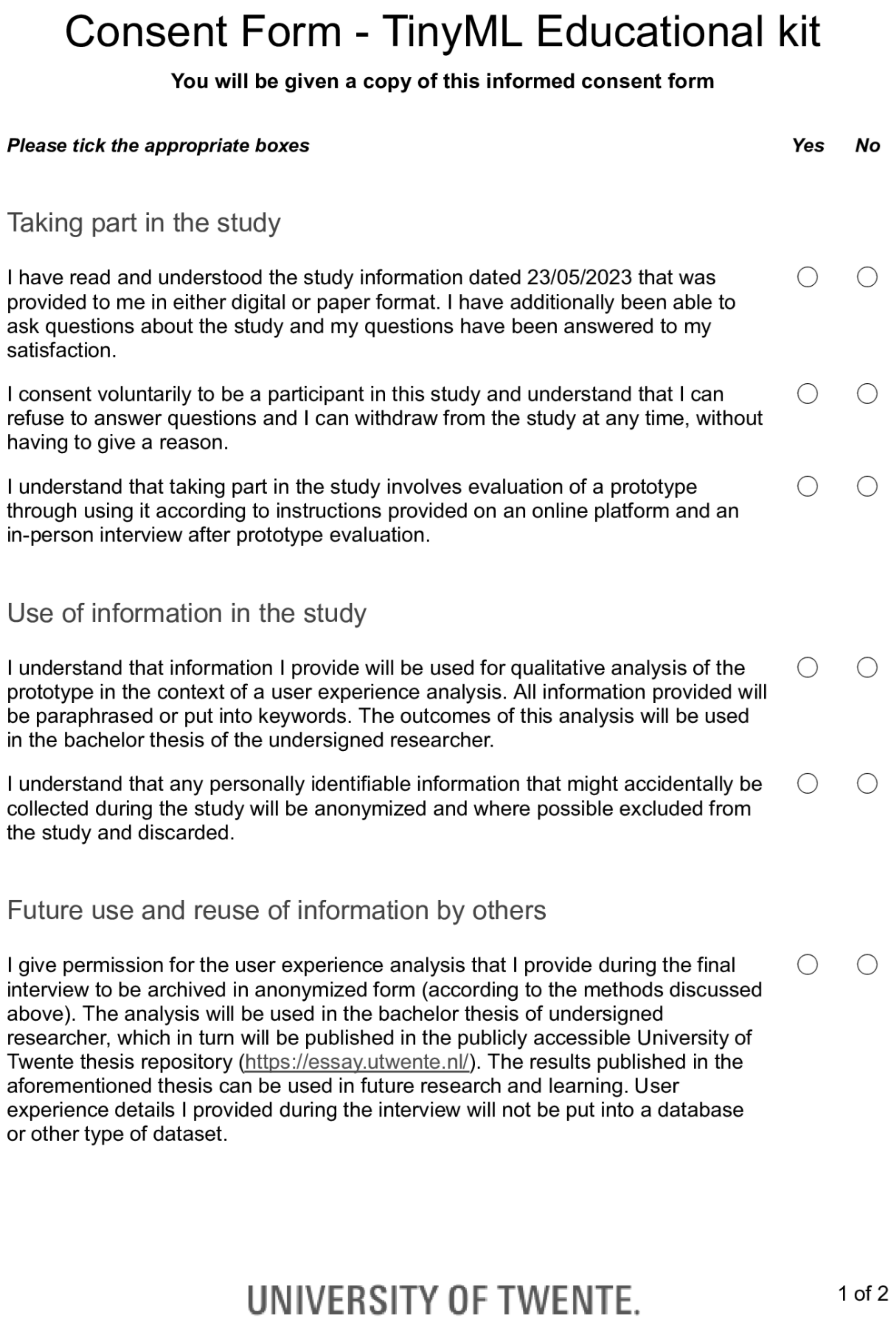
Contact

If you have further questions about the study, please contact the researcher Jannick Siderius at [redacted].

If you want to discuss matters related to the study, but not talk directly to the researcher, please contact the study supervisor Marcus Gerhold at [redacted].

If you have questions about your rights as a research participant, or wish to obtain information, ask questions, or discuss any concerns about this study with someone other than the researcher(s), please contact the Secretary of the Ethics Committee Information & Computer Science at [redacted].

Appendix III - Consent form



Consent Form - TinyML Educational kit

You will be given a copy of this informed consent form

Please tick the appropriate boxes	***Yes***	***No***
Taking part in the study		
I have read and understood the study information dated 23/05/2023 that was provided to me in either digital or paper format. I have additionally been able to ask questions about the study and my questions have been answered to my satisfaction.	○	○
I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	○	○
I understand that taking part in the study involves evaluation of a prototype through using it according to instructions provided on an online platform and an in-person interview after prototype evaluation.	○	○
Use of information in the study		
I understand that information I provide will be used for qualitative analysis of the prototype in the context of a user experience analysis. All information provided will be paraphrased or put into keywords. The outcomes of this analysis will be used in the bachelor thesis of the undersigned researcher.	○	○
I understand that any personally identifiable information that might accidentally be collected during the study will be anonymized and where possible excluded from the study and discarded.	○	○
Future use and reuse of information by others		
I give permission for the user experience analysis that I provide during the final interview to be archived in anonymized form (according to the methods discussed above). The analysis will be used in the bachelor thesis of undersigned researcher, which in turn will be published in the publicly accessible University of Twente thesis repository (https://essay.utwente.nl/). The results published in the aforementioned thesis can be used in future research and learning. User experience details I provided during the interview will not be put into a database or other type of dataset.	○	○

UNIVERSITY OF TWENTE.

1 of 2

Withdrawal of consent

I understand that I can withdraw from the research at any time without explanation or justification	○	○

Signatures

Name of the Participant	Signature	Date

I have accurately informed the potential participant about the nature and expectations of this study and, to the best of my ability, ensured that the participant understands to what they are freely consenting.

Name of the Researcher	Signature	Date

Contact

If, during the course of the research, you have further questions about the study, please contact the researcher Jannick Siderius at [redacted].

If you want to discuss matters related to the study, but not talk directly to the researcher, please contact the study supervisor Marcus Gerhold at [redacted].

If you have questions about your rights as a research participant, or wish to obtain information, ask questions, or discuss any concerns about this study with someone other than the researcher(s), please contact the Secretary of the Ethics Committee Information & Computer Science at [redacted].

UNIVERSITY OF TWENTE.

2 of 2

Appendix IV - Interview questions

User Experience interview questions

TinyML educational kit testing

Welcome to the interview, thank you for participating in this research.
The data that is collected during this interview will be written or typed out, in a paraphrased manner (so no exact sentences).
The purpose of the interview is to gather information about the participants' background, prototype, learning methods as well as participants' general experience in using the kit.

Main questions will be in normal type, *follow up questions will be in italic.*

Background

What programming experience did you have when starting this test? *Can you give an example of the experience you have?*
Did you have any experience with machine learning / TinyML before starting this test? *Can you give an example of the experience you have?*
What arduino experience did you have when starting this test? *Can you give an example of the experience you have?*

TinyML kit

Can you describe your initial thoughts and impressions when starting to use the educational kit? *Can you describe both the development kit as well as the online platform?*
Can you describe your experience of interacting with the online learning platform? *Describe the interaction with the website (e.g. navigation, aesthetic), not yet the content.*

Learning material

Can you explain what you learned in the first chapter on the basics of neurons? *Can you give an example of knowledge you retained or found especially interesting?*
Can you explain what you learned in the second chapter on the basics of networks? *Can you give an example of knowledge you retained or found especially interesting?*
Can you explain what you learned in the third chapter on backpropagation / training? *Can you give an example of knowledge you retained or found especially interesting?*
Was there any important content missing from the learning material in your opinion? *This could be minor additions such as more examples or more extensive explanations, or major additions such as another chapter, missing context information etc.*
Can you describe if lifting the 'black box' on machine learning contributed to your learning experience? *Normally, machine learning is taught in a very abstract way, here, the researcher*

tried to make learning more understandable by providing examples as well as comprehensive explanations.
Were there any advantages or disadvantages in using an online learning platform like the one presented in the test? *Can you give concrete examples of advantages and disadvantages?*
Was the experience comprehensible? *Please describe why it was or wasn't e.g. because of understanding, writing style, examples given, platform used etc.*

Development board

Did the development board add value to the learning process? *Can you give examples where the development board helped or hindered you?*
How did you experience using the development board? *Can you give actual usage examples from your experience?*
How did you find the information on using the development board that was present on the online platform? *Was there any information missing, incomplete, unclear etc.?*
Were there any features of the development board that you liked or disliked? *This can include on-board sensors, connection options, aesthetics etc.*

Experience

Could you describe the overall user experience of your test? *This can include how the kit was composed, what the learning material entailed, how the projects were introduced etc.*
Was the experience engaging? *Please describe why it was or wasn't.*
What are the advantages and disadvantages of this type of learning experience? *This can include evaluation of the kit as a whole, the online platform, the examples, the writing style etc.*
If you have learned other skills (e.g. programming language, microcontroller, video editing etc.) using an online learning platform, can you compare your experience from that learning experience with the one you tested? *The comparison can include platform comparison (e.g. features), example usage, writing styles, content delivery etc.*
What changes would you make to the educational kit if you could? *This could be changes to the development board, the kit, the online platform, the examples etc.*

Are there any other relevant points related to the evaluation of the educational kit we missed during this interview? *This could be (major) problems you occurred, comments on the content, structure etc.*

Thank you very much for participating in this research.

Appendix V - Educational Kit cost breakdown

Unit	**Price**
Circuit board (*unit price/qty 10*)	€ 2.84
ESP32-S3-WROOM-1 N16R8 *(unit price/qty 5)*	€ 4.71
LSM6DS3TR-C (*unit price/qty 10*)	€ 1.34
ICS-43434 (*unit price/qty 5*)	€ 1.98
APDS-9930 (*unit price/qty 10*)	€ 0.87
BME-280 (*unit price/qty 10*)	€ 3.07
AH-49E (*unit price/qty 50*)	€ 0.10
H638T-TR2 (*unit price/qty 10*)	€ 0.83
Neopixels (WS2812B) (*unit price/qty 100*)	€ 0.19
Voltage regulators (*unit price/qty 50*)	€ 0.17
USB-C connector (*unit price/qty 10*)	€ 0.21
Stemma QT / Qwiic connector *(unit price/qty 20)*	€ 0.15
Expansion header (*unit price/qty 10*)	€ 0.19
LEDs (*unit price/qty 100*)	€ 0.10
Buttons (*unit price/qty 50*)	€ 0.13
Passive components (Resistors and Capacitors) *(unit price/qty 1000)*	€ 0.34
3D printed case	€ 0.25
Mounting hardware (*unit price/qty 100*)	€ 0.07
Soldering consumables	€ 0.50
USB-C cable	€ 1.99
Packaging and Printing *(unit price/qty 10)*	€ 1.12
Total/Educational Kit	**€ 21.15**

All components were purchased in the quantities as described. Shipping, handling and VAT are included in the price. Assembly costs such as machine time, wear and tear, labour time are not included in the price. All prices are accurate as of June 2023.