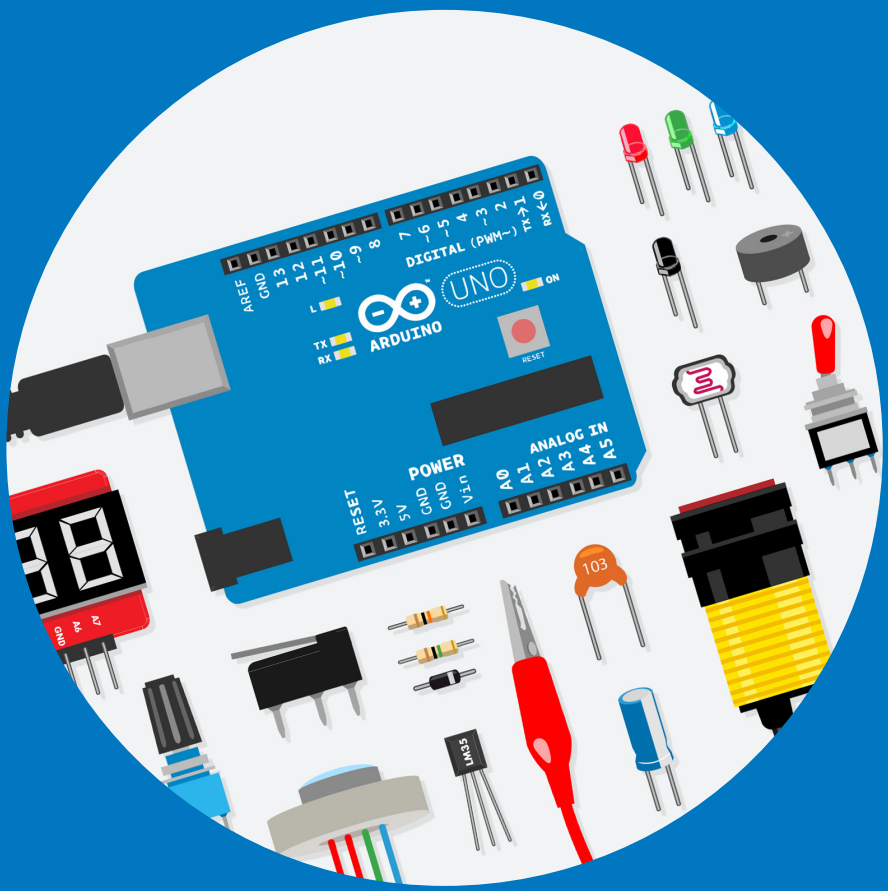
Department of Electrical Engineering and Automation

Lowering barriers on embedded system design

Turning innovations into prototypes

Kimmo Karvinen



Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 68/2019

Lowering barriers on embedded system design

Turning innovations into prototypes

Kimmo Karvinen

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Electrical Engineering, at a public examination held at the lecture hall AS1 of the school on 17 May 2019 at 12:00.

Aalto University
School of Electrical Engineering
Department of Electrical Engineering and Automation

Supervising professor
Professor Ville Kyrki, Aalto University, Finland

Preliminary examiners
Prof. Peter Jamieson, Miami University, USA
Dr. Jennifer Cross, Tufts University, USA

Opponent
Dr. Martin Edin Grimheden, KTH, Sweden

Aalto University publication series
DOCTORAL DISSERTATIONS 68/2019

ISBN 978-952-60-8506-7 (printed)
ISBN 978-952-60-8507-4 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-8507-4

Unigrafia Oy
Helsinki 2019

Finland

Printed matter
4041-0619

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Kimmo Karvinen

Name of the doctoral dissertation
Lowering barriers on embedded system design

Publisher School of Electrical Engineering

Unit Department of Electrical Engineering and Automation

Series Aalto University publication series DOCTORAL DISSERTATIONS 68/2019

Field of research Automation Technology

Manuscript submitted 28 February 2019 **Date of the defence** 17 May 2019

Permission for public defence granted (date) 2 April 2019 **Language** English

☐ **Monograph** ☒ **Article dissertation** ☐ **Essay dissertation**

Abstract

Embedded system development platforms have made designing prototypes and devices possible outside the engineering domain. While interdisciplinary community and maker movement have strongly adopted breakout physical computing toolkits, such as Arduino, it is not only hobbyists utilizing them.

Easy accessibility toolkits are used in various research and scientific projects and in engineering education. Using embedded system development platforms have become a widespread practice of both engineers and non-engineers alike, covering almost every field. While more accessible tools save beginners from many low-level problems, embedded system design still requires a set of skills which may seem overwhelming for novices. Getting started can be burdensome especially for non-engineers as embedded systems are a mixture of both software and hardware with challenging subareas, such as programming.

This dissertation explores which tools and processes would lower the threshold of designing embedded systems, enabling non-engineers and novice engineers to turn their innovations into working prototypes in such way that the workflow allows experimentational prototypes to evolve into deployable embedded systems.

As a result, a method is presented, including a framework for selecting novice friendly sensors, a minimalistic approach for teaching robot prototyping, an IoT rapid prototyping laboratory setup and a case study employing the same tools for developing satellite subsystems as those that are applicable for multidisciplinary novice use. In the case workshops, all student groups successfully completed the given prototyping tasks. In addition to tangible results, reception of the subject, including typically burdensome areas such as programming, was very positive.

Keywords Sensors, embedded systems, Arduino, engineering education, learning barriers, microcontrollers, free software, Internet of Things (IoT)

ISBN (printed) 978-952-60-8506-7 **ISBN (pdf)** 978-952-60-8507-4

ISSN (printed) 1799-4934 **ISSN (pdf)** 1799-4942

Location of publisher Helsinki **Location of printing** Helsinki **Year** 2019

Pages 112 **urn** http://urn.fi/URN:ISBN:978-952-60-8507-4

Tekijä
Kimmo Karvinen

Väitöskirjan nimi
Kynnyksen madaltaminen sulautettujen järjestelmien suunnittelussa

Julkaisija Sähkötekniikan korkeakoulu

Yksikkö Sähkötekniikan ja automaation laitos

Sarja Aalto University publication series DOCTORAL DISSERTATIONS 68/2019

Tutkimusala Automaatiotekniikka

Käsikirjoituksen pvm 28.02.2019 **Väitöspäivä** 17.05.2019

Väittelyluvan myöntämispäivä 02.04.2019 **Kieli** Englanti

☐ **Monografia** ☒ **Artikkeliväitöskirja** ☐ **Esseeväitöskirja**

Tiivistelmä

Sulautettujen järjestelmien kehitysalustat ovat mahdollistaneet prototyyppien ja laitteiden suunnittelun myös muiden kuin insinöörien toimesta. Vaikka monialainen yhteisö ja Maker-liike ovat vahvasti omaksuneet kehitysalustojen, kuten Arduinon, käytön, ei niiden hyödyntäminen rajoitu harrastajiin.

Helppokäyttöisiä työkaluja käytetään lukuisissa tutkimus- ja tiedeprojekteissa, sekä insinöörien opetuksessa. Sulautettujen järjestelmien kehitysalustojen käytöstä onkin tullut laajasti levinnyt käytäntö myös muilla kuin teknisillä aloilla. Vaikka helppokäyttöisemmät työkalut säästävät käyttäjät monilta matalan tason ongelmilta, sulautettujen järjestelmien suunnittelu vaatii laajan yhdistelmän erilaisia taitoja, joiden opettelu voi vaikuttaa liian haastavalta aloittelijoille. Alkuun pääsy voi olla erityisen vaikeaa muille kuin insinööreille, koska sulautetut järjestelmät ovat yhdistelmä ohjelmistoa ja laitteistoa, sekä niiden haastavia osa-alueita, kuten ohjelmointia.

Tämän väitöskirjan aiheena on tutkia, mitkä työkalut ja prosessit laskisivat kynnystä suunnitella sulautettuja järjestelmiä siten, että muidenkin kuin insinöörien olisi mahdollista kehittää innovaatioistaan toimivia prototyyppejä. Olennaista on, että työnkulku mahdollistaa kokeellisten prototyyppien kehittymisen käyttökelpoisiksi sulautetuiksi järjestelmiksi.

Tuloksena esitetään metodi, joka sisältää tavan valita aloittelijaystävällisiä sensoreita, minimalistisen lähestymistavan robotiikan prototyyppien rakentamisen opettamiseen, nopeat IoT-prototyypit mahdollistavan prosessin, sekä case-esimerkin, jossa käytetään samoja monialaiseen aloittelijakäyttöön sopivia työkaluja satelliitin sensorin kehittämiseen. Case-työpajoissa kaikki opiskelijaryhmät rakensivat onnistuneesti prototyypit toimeksiannon mukaisesti. Onnistuneiden projektien lisäksi perinteisesti haastavana pidetyn aihepiirin vastaanotto oli erittäin positiivinen.

Avainsanat Sensorit, sulautetut järjestelmät, Arduino, mikrokontrollerit, avoin lähdekoodi, IoT

ISBN (painettu) 978-952-60-8506-7 **ISBN (pdf)** 978-952-60-8507-4

ISSN (painettu) 1799-4934 **ISSN (pdf)** 1799-4942

Julkaisupaikka Helsinki **Painopaikka** Helsinki **Vuosi** 2019

Sivumäärä 112 **urn** http://urn.fi/URN:ISBN:978-952-60-8507-4

Acknowledgements

The author wishes to express his thanks to

Aalto University, Core Factory, Jennifer Cross, Haaga-Helia University of Applied Sciences, Ahmad Ibrahim, Juho Jakka, Peter Jamieson, Jarmo Karvinen, Jatta Karvinen, Kirsti Keltikangas, Tomi Knuutila, Kari Koskinen, Hannu Leppinen, Maker Media, O'Reilly Media, Cem Ozan Asma, Piisku, Jaan Praks, Zenon Pudlowski, Readme.fi, Will Sillitoe, Tuomas Tikka, University of Art and Design Linz, University of Lapland, Ville Valtokari, Andreas Zingerle, Ahmed F. Zobaa

For their overwhelming support, the author wishes to express his special thanks to

Ville Kyrki, Tero Karvinen, and Jussi Suomela

And above all to my fiancée Marianna Väre whose endless support made this thesis possible.

Helsinki, 28 February 2019
Kimmo Ilmari Karvinen

Contents

List of Abbreviations

ADCS	Attitude determination and control system
AJAX	Asynchronous JavaScript and XML
API	Application programming interface
CORS	Cross-Origin Resource Sharing
COTS	Commercial off-the-shelf
DIY	Do-it-yourself
EEE	Electrical, electronic, and electromechanical
FOSS	Free and open-source software
GUI	Graphical user interface
HIL	Hardware-in-loop
I2C	Inter-Integrated Circuit
IC	Integrated circuit
IDE	Integrated development environment
IFTTT	If This Then That
IoT	Internet of Things
LAMP	Linux, Apache, MySQL, PHP
LDR	Light-dependent resistor
NAT	Network address translation
QFP	Quad Flat Package
R&D	Research and development
RiE	Robotics in Education
RTC	Real-time clock
SPI	Serial Peripheral Interface

List of Publications

This doctoral dissertation consists of a summary and of the following publications which are referred to in the text by their numerals.

I Karvinen, K. 2018. Choosing Novice Friendly Sensors. International Journal of Electrical Engineering Education (IJEEE). First published September 18 2018. https://doi.org/10.1177/0020720918800821

II Karvinen, K. 2016. Teaching robot rapid prototyping for non-engineers - a minimalistic approach. World Transactions on Engineering and Technology Education (WTE&TE), volume 14, issue 3, pages 341-346.

III Karvinen, K. & Karvinen, T. 2018. IoT Rapid Prototyping Laboratory Setup. International Journal of Engineering Education (IJEE), volume 34, issue 1, pages 263-272.

IV Karvinen, K., Tikka, T., & Praks, J. 2015. Using hobby prototyping boards and commercial-off-the-shelf (COTS) components for developing low-cost, fast-delivery satellite subsystems. Journal of Small Satellites (JoSS), volume 4, issue 1, pages 301-314.

1. Introduction

Embedded system development platforms have made designing prototypes and devices possible outside the engineering domain. While interdisciplinary community and the maker movement have strongly adopted prototyping toolkits, such as Arduino [1], it is not only hobbyists utilizing them. Easy accessibility toolkits are used in various research and scientific projects [2] and in engineering education [3]. Using embedded system development platforms has become a widespread practice of both engineers and non-engineers alike, covering almost every field [4].

Which advantages could be provided by non-engineers participating in designing and developing embedded systems? Would non-engineers provide additional value in this field compared to engineers who have dedicated their studies to mastering technical aspects of the process? While lacking technical skills, non-engineers can be experts in their own domain. Even though engineers have an excellent understanding of their own area of expertise, it is likely that they do not have more than a superficial knowledge of other areas, such as medicine, biology or user interface design. Even the best possible R&D (research and development) organization cannot possess expertise beyond its main field of interests or to assume that it does not need input from outsiders [5]. Therefore, having more interdisciplinary involvement from different areas of expertise can bring out innovations and approaches that would not otherwise be possible [6]. Diverse groups participating in different technology projects also form communities that support other users and products on the market [7], [8]. An ideal goal should be to utilize knowledge from experts from varied areas to bring out new innovations and to support existing projects without compromising the quality of the outcome or teaching skills that are not necessary for the process.

This thesis focuses on lowering barriers on embedded system design, making it more straightforward to turn innovations into prototypes. Accessible tools and workflows make getting started easier and produce more instantaneous results. On the other hand, advanced users benefit from effective workflow and short feedback loop. In the scope of this thesis, novices are defined as students who lack an embedded system and programming skills. Novice students can be undergraduate non-engineers or novice engineers.

In order to gather an embedded system contribution outside the engineer domain, non-engineers should be able to participate in the prototyping process. Building prototypes is a functional way to test ideas and communicate these to others. At the same time, it provides a hands-on practical application of

engineering and a possibility for project-oriented design-based learning [9]. The prototyping process can expose design weaknesses, specify technical requirements and ensure that a device is fit for its designed purpose. Making a prototype in an early stage of the development life cycle also allows one to entirely change the approach or even discard an infeasible design. Successful prototypes can evolve into products or become parts of other prototypes or products. Publishing open source projects makes findings available to others beyond the original development group.

While accessible tools save beginners from many low-level problems, embedded system design still requires a set of skills which may seem overwhelming for novices [10], [11], [12], [13]. First of all, there are various prototyping kits with different features and limitations for novices to choose from [14]. In addition, there is a variety of electronic retailers selling large selection of components [15], [16]. Proprietary toolkits force the user to design prototypes based on components made available by manufacturer [10].Using a non-proprietary development platforms allows for an ample range of further choices to be made when selecting components to fit the desired outcomes. When the development platform has been selected, the combination of software and hardware including difficult sub-domains, such as programming [4], can create an insurmountable learning barrier. While programming is a central part of learning embedded systems, mastering it can take about 10 years [17]. Hence to properly learn programming before getting started with embedded systems is not a viable option for most people. Learning programming is also generally considered to be difficult, which shows in the high drop-out rates from programming courses [18]. If novices aim to build prototypes for commercial purposes or for real-life use, choosing tools and components that fit the purpose can be a challenge on its own. Many novice friendly toolkits are not designed for serious prototyping limiting their use beyond learning [14].

The challenges that this research recognizes and focuses on are:

- Choosing suitable hardware and software from an abundant selection [15], [16], [14].
- Embedded system programming challenges [19], [20].
- Defining necessary skills and workflow that enables prototyping based on students' own ideas [20], [10].
- Choosing tools and processes that are suitable for both novice use and more serious prototyping [14].

The main research question of this thesis is:

> Which tools and processes lower the threshold of designing embedded systems, enabling non-engineers and novice engineers to turn their innovations into working prototypes? The workflow should allow experimental prototypes to evolve to deployable embedded systems.

1.1 From first prototypes to deployable embedded systems

There are several approaches available that aim to make prototyping and programming easier and less intimidating than with breakout toolkits combined with syntax revealing IDE (Integrated development environment). Many approaches aimed at novices, such as programmable bricks and modular blocks [11], [14], provide an easy process but fixed functionality limits development possibilities and the potential for real-life solutions. Programming environments aimed at novices often hide the actual code and provide a graphical user interface (GUI) instead. This allows students to learn programmatic constructs and the logic of the program before learning syntax [21]. The down side of GUI is that it prevents users from learning to read and write the syntax. As programming and understanding syntax are a central part of learning embedded systems, these solutions have a reduced usability for serious prototyping and for creating a foundation for further development. Without learning syntax most of the free community resources, such program and circuit examples, can not be utilized. The ability to understand syntax can also be applied to all major programming environments and it is indispensable for a user to be able to perform debugging.

Any tool or development environment is a trade-off, arguably, learning visual programming tools first could make learning syntax easier [22]. An easier start can also enhance motivation and self-efficacy [22]. On the other hand, modern development tools are quite straightforward to use and using syntax from the beginning saves students from learning multiple tools and workflows.

In addition, proprietary systems limit the range of possible components, forcing the user to design prototypes based on options made available by the manufacturer [10]. These limitations narrow down possibilities in using learned skills for commercial and industrial prototypes. The maker movement is an example of how people from different fields are not primally learning embedded systems to improve their automation and electrical engineering skills, but in order to be able to build prototypes and devices based on their ideas [12].

Arduino was chosen as the development platform in the scope of this thesis, as it is free and open-source software (FOSS) with open hardware. In addition, it is widely available and has a strong online community. In engineering education, Arduino is also used for its accessibility, adaptability and compatibility [23], [4], [24]. As Arduino uses an ATmega microcontroller found also in many commercial products, the prototype can evolve from the first tests into a final product.

Figure 1 shows an example of development that started as an Arduino prototype and evolved to a space instrument. The final version of the sun sensor used in Aalto-1 and Aalto-2 satellites uses the same program, microcontroller and sensor as the prototype developed with Arduino. The miniaturized version includes Atmega328P in Quad Flat Package (QFP) and leaves out supporting electronics that are not necessary for the final model. The sun sensor prototyping project is presented in Chapter 5.

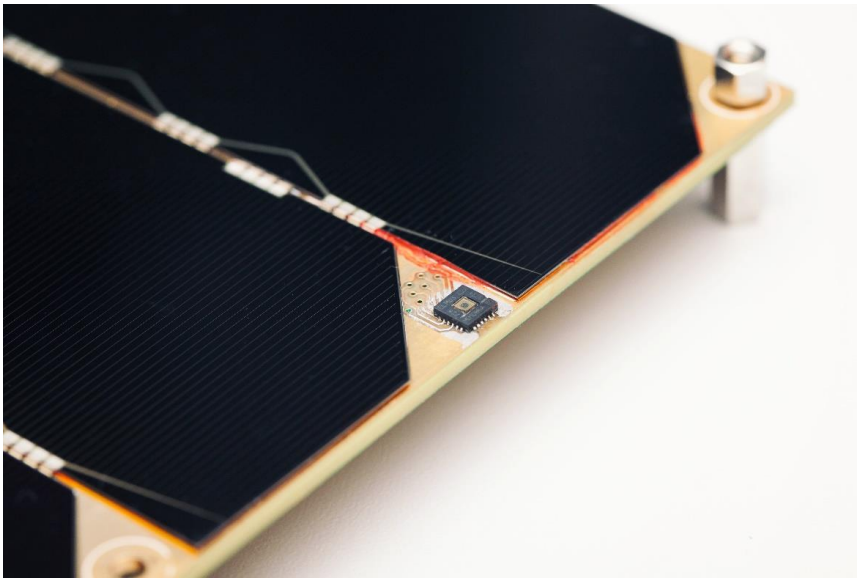


Figure 1. Final version of the sun sensor used in Aalto-1 and Aalto-2 satellites.

1.2 Related research

There is ample research available regarding programming and embedded system learning challenges [11], [12], [25], [26], [27], [28] and utilizing Arduino for teaching and prototyping [3], [23], [4], [24]. Programming and embedded system challenges are a combination of software and hardware issues, such as syntax problems [25], [26], [27] [28], problems understanding the relationship of virtual and physical domains [11], [12] and problems in locating errors [12]. Most research tends to agree that programming is a remarkably difficult skill to learn [25], [26], [27], [28] and building working embedded systems is complicated for novices [11], [12].

There are several advantages that research highlights about Arduino that explains its popularity and why it was chosen as the development platform in the scope of this thesis. Open source hardware and software is often emphasized and differentiates it from proprietary options [3], [23], [4]. A large online community provides content, support and reference implementations for Arduino users [3], [24]. Importantly, both teachers and students often see it is a competent solution for embedded system design [3], [4], [24].

As Patiño O.A et al. point out, the Arduino workshop results also include less obvious outcomes, such as development of code understanding and system integration skills [29]. In research by Jamieson, P., the final project results of the Arduino course are more creative and better than in previous years [3]. A popular hobbyist magazine MAKE: summarizes reasons why the platform has become a tool of choice for non-engineers, such as hobbyists and artists, in the article “Why the Arduino Won and Why It’s Here to Stay” [30]. The impact is

even compared to the early days of personal computers [30]. The main reason provided for Arduino's success is that it allows people who lack electronics or microcontrollers skills to realize their ideas [25]. Arduino's strength is that that it provides a relatively simple way to connect the necessary tasks together to build a vast number of different devices and prototypes [25].

Arduino is also used as a development platform in numerous research and scientific projects [2], [31]. However, there is no systematic research providing a method aimed at multidisciplinary embedded system prototyping, that would allow novices to design and build prototypes that can evolve from experimental prototypes to deployable embedded systems. Presented research aims to fill this gap and describe a workflow that lowers barriers and supports development beyond learning and practising embedded systems. The basis from which this work builds upon is the possibility to enable multidisciplinary students to use the same accessible toolkit for diverse purposes, from simple beginner learning to innovative device prototypes and all the way to serious devices. Aim is not to define a process that emphasizes learning technical skills, such as programming or teaching physical computing, but to present and test a workflow that focuses on the skillset that enables effective and versatile prototyping. These aspects are covered in chapters 3 and 4 describing embedded systems workshops. Chapter 5 presents a use case where the same prototyping toolkit is utilized for serious prototyping.

There is a vast selection of components available for different needs and designs. For novices this can be a challenge, as they lack the knowledge to make choices that match their skill level. Research prominently describes courses using LEGO MINDSTORMS with proprietary components [32], Arduino accompanied with a more unrestricted component selection [3], as well as various more experimental easy accessibility prototyping systems, such as Bloctopus [39]. However, currently there is no common best practice for choosing sensors for teaching novice embedded system design. Chapter 2 presents a framework for selecting sensors from any manufacturer for multidisciplinary novice groups leaving the application field for the framework to be defined by teachers, students and course objectives.

1.3 Research methods

By using a constructive research method and inductive reasoning, a combination of tools and workflows is proposed. The suitability of the proposed method is validated by qualitative research instantiated by pilot workshops and a case study.

Two pilot workshops were arranged to test and validate the methods. In the first case workshop, learning robot prototyping was tested in a two-day-course at the University of Art and Design in Linz (n=9).

Teaching IoT rapid prototyping was piloted in a four-day-workshop at the University of Lapland with art students (n=19) (Figure 2). After the pilot workshop, the experiment was repeated with another group (n=27) at Haaga-Helia

University of Applied Sciences. The workshop in Haaga-Helia was not arranged by the author of this thesis.

In both workshops, Lintz and University of Lapland, there were preliminary and feedback questionnaires collected. Data for assessment was collected by paper questionnaires and observation was made by the workshop teacher. The questionnaires included topics that were used for matching teaching to the student groups skill level. From the perspective of this thesis and the related articles, there were two key results analyzed for both workshops.

Firstly, in the preliminary questionnaire, the learners were asked to evaluate their own programming skill-level on scale 1-5, 5 being the most positive. The same evaluation was asked to be made again in the feedback questionnaire to be able to see if the evaluation had changed during the course: “How would you evaluate your programming skills?” (scale 1-5).

Secondly, in the feedback questionnaire, the learners were asked to rate the workshop on the same scale 1-5. This result was analyzed to indicate whether the workflow and processes were well-received by the target audience, given the complex nature of the topic and the short timeframe of the workshops: “How would you rate this workshop”? (scale 1-5).

A case study using Arduino and a commercial-off-the-shelf component for developing low-cost and fast-delivery satellite sub-systems is presented in chapter 5. This chapter studies if the same tools, that are suitable for multidisciplinary novice use, are sufficient for prototyping and developing functional final products that can be deployed in the field. Even though, the operation of the sun sensor presented in the use case is relatively simple, it could not be prototyped by most toolkits aimed for novices. The requirement specifications for satellite parts add additional challenges for component selection and the programming environment.

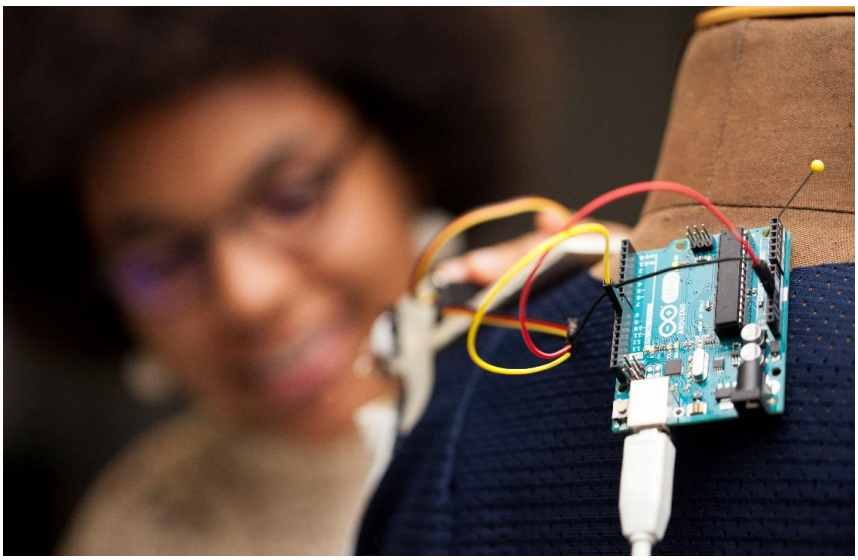


Figure 2. Student prototyping a shirt with sleeves that can be raised and lowered with a cell phone user interface. Teaching IoT rapid prototyping was piloted in a four-day-workshop at the University of Lapland with art students.

1.4 Maker movement and end users

Research on maker movement and end users presents challenges, possibilities and topics related to this thesis. At the start of the 21st century, the price of the prototyping equipment, such as 3D printers and laser cutters, became considerably less expensive, enabling prototypes to be created rapidly and without traditional production processes [33]. MAKE Magazine, Maker Faire and fab labs provided ideas, an environment and tools for creative fabrication [33]. Affordable hardware, open software, digital fabrication and sharing results brought a possibility for a growing group of everyday people to innovate and create [8], [34]. The maker movement is formed of people who want to learn new skills and technologies to be able to use their creativity for designing, building and crafting [34]. Makers also desire to be more than consumers of the products they use [35].

The distinct features of the maker movement are focused on physical objects instead of digital ones and a do-it-with-others mindset instead of just do-it-yourself (DIY) [34], [36].

Development made by end users blurs the separation between designers and consumers [18]. As work practises and requirements change over time, computational systems are never complete but must constantly evolve to meet the demands of the users [18]. To flexibly meet the changing demands, users themselves should be able to modify the systems to suit their needs [37]. End user development is not straightforward, and a lack of adequate programming skills is one of the major issues that users face [37]. Despite the technical challenges, most programs today are not written by professional programmers but by people with expertise in other domains [38]. While the maker movement is driven by the expression of creativity, the main motivation for the end user development is typically a need to make applications more suitable for end users themselves [20].

Makers and participating end users create new products and support existing products by their contribution [8]. Both are largely dependent on accessible tools and community support. While methods presented in this thesis focus on generally lowering barriers on embedded system design to turn innovations into prototypes, they are also usable for providing a foundation for makers and end users in order to participate in embedded system projects.

1.5 Contributions

Key contributions of the thesis are as follows.

Publication I, Choosing novice friendly sensors

Choosing novice friendly sensors can make a substantial difference for novices when using breakout physical computing toolkits. In this paper a framework for selecting sensors for novice use is presented and accessibility obstacles are defined.

- Framework for selecting sensors for a novice use.
- Four properties for evaluating sensor suitability for a novice use.

Publication II, Teaching robot rapid prototyping for non-engineers - a minimalistic approach

A method of teaching non-engineers robot rapid prototyping is proposed. A minimalistic skill-set is defined, aiming to enable students to design and build their own basic robot rapid-prototypes.

- Minimalistic teaching approach, lowering barriers for non-engineers engaging in robot prototyping projects and supporting utilisation of interdisciplinary expertise.
- Case study indicating that basic embedded systems skills can be summarized in a package that can be presented in one day.

Publication III, IoT Rapid Prototyping Laboratory Setup

A setup for rapid IoT prototyping in a classroom, identifying necessary skills and combining these into a workshop that allows students to turn their ideas into prototypes during a four-day-workshop.

- Approach that enables novice students to turn their ideas into working IoT prototypes during a four-day-workshop.
- Defining necessary basic skills for learning and prototyping IoT.
- Open source setup for IoT prototyping utilizing a development board and a computer.

Publication IV, Using hobby prototyping boards and commercial-off-the-shelf components for developing low-cost and fast-delivery satellite subsystems

The paper presents a use case where the same development platform used in the novice workshops described in this thesis is utilized for serious prototyping. The focus is on the Aalto-1 Sun Sensor prototype development process.

- Process for developing low-cost, fast-delivery satellite subsystems using commercial off-the-shelf (COTS) components.
- Presenting a use case: Aalto-1 Sun Sensor Subsystem Prototype Development.
    - The sun sensor is one of the smallest and cheapest satellite sun sensors available.
    - Use case provides evidence that despite the strong DIY background, hobby prototyping board Arduino can be used for serious prototyping.

1.6 Structure of the thesis

This thesis consists of three parts: the introduction, four chapters introducing the publications and the published articles. Chapter 2 presents a framework for selecting sensors and four properties for evaluating sensor suitability for a novice user. Selective sensor exposure can make learning embedded system easier for the novices, without hiding the program syntax or using proprietary components. Six learning barriers in end-user programming by Ko et al. is adopted for evaluating obstacles with sensors [27]. Four properties for evaluating sensors and ways to overcome those obstacles are presented. The framework provided in the article can be used for component selection for workshops or for estimating sensor suitability for external groups in projects such as described in chapter 5.

The following chapter 3 presents a minimalistic approach for teaching robot rapid prototyping for non-engineers. Findings can be applied also to teaching other than robot related embedded system basics. The presented method can be used for introductory embedded system education and as a primer for more advanced workshops, such as the IoT-workshop described in chapter 3.

Chapter 4 introduces an IoT rapid prototyping laboratory setup that enables novice students to turn their ideas into working IoT prototypes during a four-day-workshop. Approach enables a fast prototyping cycle, using a common and well-established development board and a computer. The entire process is based on free software and open source tools. The article provides a method for more advanced and specialized embedded system prototyping, while keeping the process accessible for novices and non-engineers.

Chapter 5 shows how a hobby prototyping board can be a viable option for serious prototyping and how external groups can bring value to a technical project. In the case example Arduino and commercial-off-the-shelf (COTS) components are used for developing low-cost and fast-delivery satellite sub-systems. The use case supports the concept that the same tools, that are utilized in novice prototyping in earlier chapters, are also suitable for serious development projects.

Results are concluded and summarized in chapter 6.

2. Choosing novice friendly sensors

Breakout physical computing toolkits, such as Arduino, enable users to connect sensors produced by various manufacturers with diverse properties, such as varying protocol and connection type. Implementation difficulty can fluctuate considerably between the sensors while novices are often unable to estimate whether components are suited to their skill level. Beginning to learn embedded systems with unsuitable sensors can have a negative effect on the user's motivation and self-efficacy. Typically, solutions that aim to provide easy component implementation use parts that are specially designed for easy accessibility, such as LEGO MINDSTORMS, Topobo or Bloctopus [14], [39], [40]. The approach presented in Publication I provides a framework that allows one to choose novice friendly sensors from a wider selection, when someone other than a novice chooses and provides the sensor selection.

On prototyping workshops or courses, sensor assortment is not typically selected by students. Instead, users choose components from an available selection to match their desired embedded system functionality. By pre-selecting novice friendly sensors several prototyping challenges can be reduced in advance.

Publication I has two key contributions:

1. A framework for selecting sensors for a novice use
2. Presenting four properties for evaluating sensor suitability for a novice use

Novices are defined as non-engineers or novice engineers who lack embedded system and programming skills. Arduino is chosen as the example development platform. Free and open-source software (FOSS) with open hardware along with broad compatibility, easy setup, low cost and comprehensive ecosystem make it suitable for an educational setting [4], [31].

The main target audience for making sensor selection based on findings presented in this chapter are university and polytechnic level non-engineers and novice engineers.

2.1 Novice challenges and learning barriers

Even though using embedded systems has become a multidisciplinary practice [4], novices face considerable challenges while learning a new area combining software and hardware [11], [12]. Programming, which is a central part of prototyping process, is particularly burdensome and challenging [25], [26], [41], [42].

Ko et al. present Six Learning Barriers in End-User Programming: design, selection, coordination, use, understanding, and information [27]. In Publication I barriers are adopted for using sensors as follows:

1) Design Barriers
The user does not know what he/she wants the embedded system to do.

2) Selection Barriers
The user does not know which tool, code or sensor to use.

3) Coordination Barriers
The user does not know how to make the selected tools, codes or sensors work together.

4) Use Barriers
The user does not know how to use the selected tools, codes or sensors correctly.

5) Understanding Barriers
The user thought that he/she could use the selected tools, codes or sensors, but they do not work as expected.

6) Information Barriers
The user has an idea why the sensor does not work but does not know how to verify it.

2.2 Evaluating sensor suitability for novice use

Publication I outlines that the suitability of sensors for a novice's use can be evaluated based on four properties:

1) Protocol complexity
2) Connection type and component size
3) Understandable real-life phenomena measured
4) Documentation

Learning barriers in relation to sensor properties are shown in Table 1. Sensor protocol complexity and documentation are the key factors affecting implementation challenges.

Table 1 Sensor properties and learning barriers

	Design Barriers	Selection Barriers	Coordination Barriers	Use Barriers	Understanding Barriers	Information Barriers
Protocol complexity			•	•	•	•
Connection type and component size			•	•		
Understandable real-life phenomena measured	•	•				
Documentation	•	•	•	•	•	•

2.2.1 Protocol complexity

Protocol complexity defines the sensor program's accessibility. It affects coordination, use, understanding and information barriers. If the program is too complex compared to a user's skills, the development is difficult while keeping the syntax intact. If the sensor is paired with a reference implementation, it is possible to achieve the basic functionality even with complicated protocol. However, debugging and extending functionality is challenging if the program is only partly understood. Protocol complexity can be divided further into four properties as follows:

- Definition strictness
- Amount of code needed for basic execution
- Need for a library
- Need for advanced coding concepts

A novice friendly sensor should have strict protocol definitions, relatively short amount of code needed for basic execution, and it should not use libraries or advanced coding concepts.

2.2.2 Connection type and component size

Connection type and component size affects coordination and use barriers. To avoid learning additional skills in the beginning, a sensor for novices should be connectable by using breadboard or pin headers. This condition leaves out many industrial sensors that are connected by surface mounting. Breakout components are easy to solder but still demand learning extra steps, adding one more possible point of failure.

A goal or a desired outcome of a workshop or a project may require using components that demand soldering or surface mounting, as with the Aalto-1 satellite project (Figure 3). In that case, appropriate support or guidance should be provided considering the skill level of the group building the prototype.

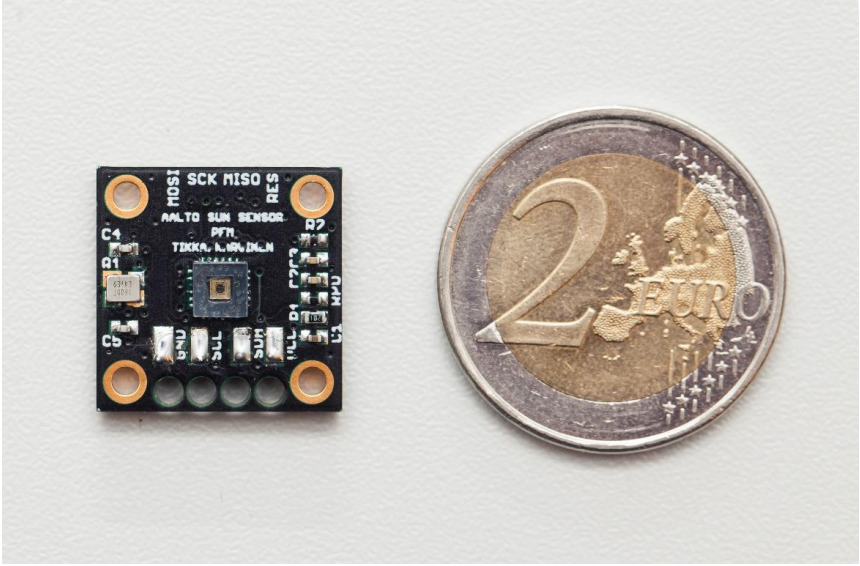


Figure 3. Surface-mount sensors used in Aalto-1 satellite project could be connected to Arduino by using breadboard or pin headers.

2.2.3 Understandable real-life phenomena measured

Understanding what a sensor measures influences on design and selection barriers. The inability to understand what a sensor measures and what it is used for renders it useless for a novice user.

Novices should be able to select the sensor according to the real-life phenomena it measures instead of how it is measured. With limited knowledge, a novice cannot select a sensor based on how it works, but on what it measures. For example, a proximity sensor can operate by using an infrared transmitter and receiver, however, for a novice it is important that it reacts to objects close to it. Dividing sensors by their real-life phenomena measured also helps to assemble a comprehensive sensor assortment. A one-sided assortment would force users to make their design based on available components instead of their own innovation [10].

2.2.4 Documentation

Documentation impacts on all learning barriers as shown in Table 2. Hence the quality of the documentation can make a crucial difference between the success and failure of the sensor use.

Table 2 Documentation and learning barriers

	Design Barriers	Selection Barriers	Coordination Barriers	Use Barriers	Understanding Barriers	Information Barriers
Target of measurement	•	•				
Purpose of sensor	•	•				
Platform specific reference implementation			•	•	•	•
Troubleshooting section					•	•

Documenting the target of the measurement and the purpose of the sensor solves issues related to design barriers and selection barriers. Without them it is difficult for a user to select a sensor to match their desired functionality.

Coordination barriers and use barriers can be lowered by including a platform-specific reference implementation. A properly commented reference implementation makes further development easier and more straightforward.

If the sensor does not work as expected and debugging is not successful, the reference implementation also provides a baseline for a user to return. Together with a troubleshooting section it helps to lower understanding barriers and information barriers.

2.3 Discussion

In respect of the thesis research question on which tools and processes lower the threshold of designing embedded systems, enabling non-engineers and novice engineers to turn their innovations into working prototypes, publication I presents a process for selecting sensors that lowers several barriers that non-engineers and novice engineers face when getting started with embedded systems. The presented approach provides a method for selecting sensors from any manufacturer for teaching novice embedded system design. Currently there is no common best practice for selecting sensors for such use.

When aiming to lower the threshold of designing embedded systems the sensors that are used with selected prototyping platform can have a considerable effect on overall difficulty. Predefined novice friendly sensor selection can lower embedded system learning barriers. Sensor protocol complexity and documentation are the most important properties affecting implementation challenges.

A perfect sensor for a novice use would have strict protocol definitions, a basic execution that could be undertaken with a relatively short code, it would not need a library, code would not include advanced coding concepts, it could be connected by using breadboard or pin headers, the real-life phenomena it

measures would support the task in hand and it would be understandable for the user. Documentation should explain the purpose of the sensor, the target of measurement, and it should include platform specific reference implementation as well as a troubleshooting section.

Ko et al. expects a growing demand for lowering barriers towards programming as more and more jobs require programming skills, and millions of end-user programmers face difficulties they need to overcome [27]. Instead of end-users, the approach described in Publication I focuses on lowering barriers for novices who are designing their first prototypes, preferably based on their own innovation.

The usability of the framework is limited for more advanced students and especially for experts. In particular, selection, coordination and use barriers are not significant issues for experts [27]. They face understanding and information barriers caused by more complicated problems and projects [27]. While debugging tools and software architecture have a key role in overcoming these challenges [27], well-made documentation can save experts from unnecessary work and reduce misunderstanding.

The presented approach focuses on selecting sensors which would support the task in hand and which would be suitable for novice use. In his thesis, Sadler J. presents barriers to novice electronics prototyping, including issues that affect the sensor usability but fall out of the scope of the Publication I [11]. As tools are designed for different audiences, their suitability for a novice can vary [11]. Sensor cost does not directly affect novice usability, but it can form a financial barrier [11]. The presented framework takes into account that sensor selection needs to be suitable for the desired outcome of the prototype or the workshop. However, a surprising need for a specific sensor can cause a significant time barrier [11]. Also, even if the sensor would be suitable for the designed functionality, the toolkit can prevent a user from achieving the desired outcome, for example, by lacking sufficient computational power [11].

The findings of Publication I are usable with breakout toolkits that do not have a graphical programming interface or predefined component selection. Toolkit features, such as lack of an analog-to-digital converter, may have an effect on sensor implementation difficulty.

3. Teaching robot rapid prototyping for non-engineers - a minimalistic approach

In Publication II a method for teaching robot rapid prototyping is proposed. Traditionally, teaching graduate and undergraduate robotics is not focused on robot design and prototyping except in engineer education [43]. Robots are used for example in teaching computer science [44], [45], artificial intelligence [46] and programming [47]. One of the reasons for using robotics in education (RiE) is to make subjects more interesting and to increase motivation [48].

The aim of Publication II is to provide a minimalistic skill-set that enables students to build simple robot prototypes based on their own designs. The development platform used is free and open-source software with open hardware. Proprietary components are not needed, making the approach suitable for various environments.

The method was studied in a pilot workshop at the University of Art and Design in Linz with 2nd and 3rd year students in the department of Timebased and Interactive Media (n=9). Most students had limited knowledge about embedded systems, robots and programming. Some had participated in basic processing lectures and tested Arduino, while some had almost no experience in these domains.

The main goal was to familiarize students with embedded system basics, enabling them to build a simple robot prototype without predefined functionality. The timeframe was short, two days, so one of the main interests was to see if turning designs into working prototypes would be too overwhelming for the student group with no engineer background, or if that goal could be obtained in the given timeframe with no drop-outs. Moreover, objectives included finding out how the topic was received overall and if self-evaluations of the students' own programming skills changed during the course.

To support achieving the goals, a basic skill-set was determined. The workshop included teaching the necessary embedded system skills, designing and building functional robots without following ready-made instructions or predefined functionality.

The first day was used for teaching the basic embedded system and robot design skills and the second one for building a robot prototype in groups of two. The robot behavior and sensors used were defined by the student groups, instead of following ready-made instructions. Support and guidance were provided through the process by the teacher. All groups successfully completed the

task, indicating that basic embedded system skills can be learned during one day.

Publication II makes two key contributions:

1. A minimalistic teaching approach, lowering barriers for non-engineers engaging in robot prototyping projects and supporting utilization of interdisciplinary expertise.
2. A case study supporting the concept that basic embedded system skills can be learned during one day.

3.1 Method and development platform

In this context basic embedded system skills are defined as a skill-set that enables students to build simple embedded systems based on functionality they have designed. Key skills for novice Arduino users are summarized in Table 3.

The central concept is to understand how input, data processing and output form all embedded systems, robots included as shown in Figure 4. Simple embedded systems can consist of a single sensor, a microcontroller and an output component while complex embedded systems have more advanced processing and more inputs and outputs. Often outputs are routed back to inputs, forming feedback loops controlling the behavior of the embedded system. During the workshop students tested input, data processing and output to comprehend how they work individually and how they connect to each other.

In scope of this thesis a robot is defined as an embedded system with an actuated mechanism programmable in two or more axes and with a degree of autonomy [19]. To qualify as a robot, the device has to be able to move and/or manipulate the environment, and it has to be autonomous enough to complete its tasks based on sensor input without human interference [19].

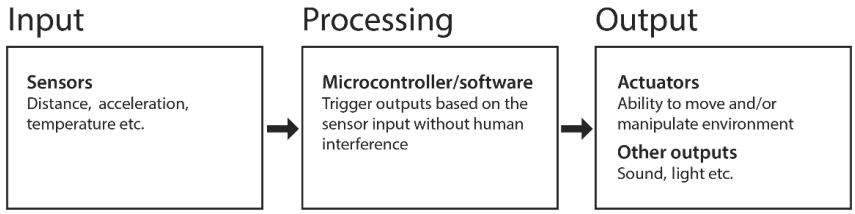


Figure 4. Input, data processing and output.

The learning process is divided into manageable steps as shown in Figure 5. Students complete each step by building it with the support of a ready-made reference implementation and circuit diagram. Programming theory is not taught before this part, but the most important program structures and parts of the syntax are learned as part of the reference implementations.

The first step is a “Hello World” program that verifies that the development environment and the development platform are working. This is also a point

where students can return if their embedded system does not work as expected and the problem cannot be solved by debugging. In the development platform used in the workshop, Arduino Uno, the standard "Hello World" involves compiling and uploading a program that makes an on-board LED connected to pin 13 blink.

Each step is then tested with a preselected input or output component. After the input and output components have been individually tested, they are combined into a simple embedded system. In the workshop example an IR sensor was combined with a piezo speaker. When something comes near the sensor, the speaker alarms.

Hobby servo motors were used as actuators. Instead of using typical limited rotation servos, continuous rotation models were chosen to be used as motors for wheels. It is not necessary for students to understand how the square wave pulse makes the servos move, as long as they learn how servo position is controlled by changing the pulse length.

Moving two servos is almost as straightforward as moving just one. When students can move servos, an input component is added. Together with a pre-built platform, these create a simple wheeled mobile robot platform that can react according to sensor readings. The pre-built platform used in the workshop did not include any electronics. It solely worked as a simple chassis for mounting wheels, a breadboard and Arduino. Compared to providing a ready-made robot platform this approach enables students to individually test each part and to get familiar with their programs before starting the development of their own devices. Reference implementations do not use libraries but reveal the whole syntax to students.

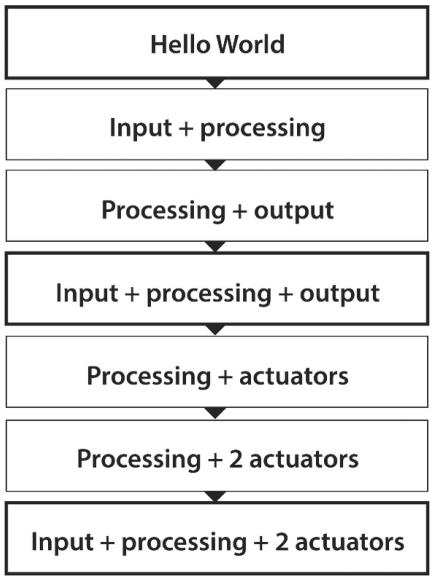


Figure 5. Sample setup steps.

Arduino was used as a development platform solving many low-level issues of prototyping. Utilizing a development platform allows for the development of embedded systems without comprehensive understanding about electronics or programming.

The key skills for novice Arduino users can be divided into basic embedded system structure and operation, basic IDE skills, connecting sensors and outputs and utilizing the online community resources as shown in Table 3. After understanding the concept of input, data processing and output the user needs to be able to utilize IDE for connecting hardware to IDE, compiling, uploading and using a serial monitor. This allows uploading and testing programs that do not demand extra input or output components. A serial monitor provides a central tool for debugging and for understanding communication between the computer and Arduino. In order to successfully connect sensors and output components, it is necessary to understand how to use digital, analogue and power pins on Arduino. Arduino has a strong online community that provides help, programming examples and tutorials. By combining and editing programs students can utilize the online community resources to further develop their designs and ideas. To make use of community resources users need to be able to edit syntax, combine programs and perform basic debugging.

Table 3
Key skills for novice Arduino users

Basic embedded system structure and operation	Basic IDE skills	Connecting sensors and outputs	Utilizing the online community resources
Input	Connecting hardware to IDE	Using digital pins	Editing syntax
Processing	Compiling	Using analogue pins	Combining programs
Output	Uploading	Using power pins	Basic debugging
	Using serial monitor		

3.2 Programming challenges and solutions

Programming is an indispensable skill for embedded system and robot prototyping. This is a major challenge as programming has a vast amount of different things to learn, it is commonly considered difficult and programming courses have high drop-out rates [41]. Learning to be a good programmer demands a substantial amount of time and effort. The path from novice to becoming an expert programmer takes about ten years [17]. Hence in a short workshop and especially with a non-technical audience, it is important to focus on basic concepts and structures that support understanding the examples. Findings of Lahtinen et al. cited in the article “Choosing novice friendly sensors” helps to separate advanced and more basic programming concepts [28], [49].

In our case workshop, control structures and syntax that were not included in the reference implementations were taught based on the needs of the students' own robot designs. Programming and debugging issues can be diminished in

advance by providing students with novice friendly sensors, as described in chapter 2.

Before starting to make their own programs, students should be able to create a mental model of the program [41]. It does not need to be technical but rather explain in small steps what should happen. Ability to make a mental model for solving a problem and then executing it with the syntax is an essential part of a programming task [42]. Creating a block diagram, as shown in Figure 6, helps to split the program in parts and to understand what kind of functions are needed in order to achieve the desired behavior. Students should be encouraged to think of programming as a series of small challenges and then build and test one step at a time.

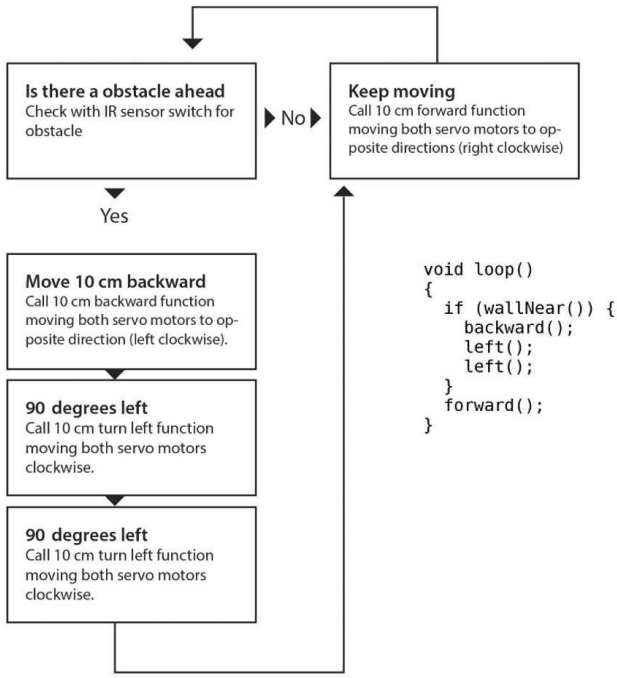


Figure 6. Wall avoiding behavior block diagram and code.

Coding style can make a major difference in how easy the code is to understand and to alter. Figure 7 shows two ways of writing the main program. Using simple blocking conditional statements and function calls keeps the program both readable and reflects the block diagram structure.

Wall avoiding robot code #1

```
void loop()
{
  if (wallNear()) {
    backward();
    left();
    left();
  }
  forward();
}
```

Wall avoiding robot code #2

```
void loop() {
  switchState = digitalRead(sensorPin);
  if(switchState == LOW) {
        for (int i=0; i<15; i++) {
            pulseServo(servoLeftPin, 1200);
            pulseServo(servoRightPin, 1800);
        }
        for (int i=0; i<30; i++) {
            pulseServo(servoLeftPin, 1200);
            pulseServo(servoRightPin, 1200);
        }
  } else {
        for (int i=0; i<15; i++) {
            pulseServo(servoLeftPin, 1800);
            pulseServo(servoRightPin, 1200);
        }
  }
  delay(10);
}
```

Figure 7. Two ways of writing main program for wall avoiding robot.

3.3 Case workshop assessment

The case study was focused on following if the students could use the theory and examples to design and build simple robots. Also, it seemed likely, that if the building day would not have been successful, it would have had an impact on students' self-efficacy and would have demotivated learning embedded systems further.

The case workshop indicated that the skills needed for designing and building basic robots and embedded systems can be summarized in a package that can be presented in one day. At the Linz workshop, every group successfully designed behavior and built their robots in one day, after only one day of learning theory and embedded system basics. The workshop agenda is presented in Table 4.

TABLE 4
WORKSHOP AGENDA

Day 1 Technology, theory and personal innovation	**Day 2** Utilizing and publishing results
• Preliminary knowledge review • Introduction • Key skills for novice Arduino users • Building the Sample setup steps together • Overview of available sensor • Forming groups of two • Assembling the pre-built robot platform • Designing group's robot behavior and component requirements	• Designing group's robot behavior and component requirements • Building the group project • Group project presentations • Feedback questionnaire

Both days were eight hours in length, including lunch break and shorter breaks between lectures and building. The first day consisted of an introduction

to embedded systems, presenting key skills shown in Table 3, building the sample setup steps shown in Figure 5 together, taking an overview of available sensors, dividing students into groups of two, assembling the pre-built robot platform, giving an assignment for the next day and starting the planning of the robot behavior for the next day. In each phase it was confirmed by the teacher that all students had successfully executed the given assignments.

At the beginning of the day two, student groups defined the behavior, inputs and outputs for the robot. Selection of about 40 sensors was provided and students were encouraged to design a straightforward reaction for the robot according to sensor input. During the second day, support for design, building and debugging was provided. While the first day was focused on learning by following directions and reference implementations, during the second day students applied the skills learned without pre-made instructions.

Problems were not solved for the students, but they were advised to build in small steps, deal with one problem at a time and use code examples to isolate the problem. When Arduino did not seem to work as expected, a "Hello World" program was compiled to make sure that the development environment and the development platform were still working. On several occasions, this helped to quickly find random problems, such as issues with the USB-connection and short circuits. Likely, if support would not have been available at this point, new and confusing problems could have led to frustrating and ineffectual debugging attempts. The most prominent slowdowns during the first day were caused by the Arduinos losing connection with the computers. The main reason for this seemed to be that some computers stopped communicating with the connected device if it drew too much power via USB port. This seemed to occur especially when using servo motors and did not happen on majority of the computers. The final robot prototypes included, for example, flame following (Figure 8), proximity sensing and color changing robots.

Measuring whether the workshop goals were obtained were done by three ways. Firstly, there were no drop-outs during the course indicating that the presented workflow was suitable for the audience. Secondly, all groups were able to produce devices based on their own design. Lastly, data was collected from the students by a preliminary and a feedback questionnaire to determine how they felt about the course workflow and their own programming skills.

A preliminary and a feedback questionnaire were used to measure change in students' self-evaluation of their programming skills and to allow students to rate the course. The average rating for the workshop was 4.4 on a scale from 1 to 5, 5 being the most positive (n=8). Self-evaluations of programming skills raised from an average of 2.3 to 2.9. Because of the small sample size and the values based on self-evaluation, conclusions about the change in the actual programming skills should not be made. Rather this can be interpreted so that during the workshop students programming self-efficacy did not drop despite the technical context. All self-evaluations of programming skills either raised or stayed the same. Moreover, due to the small sample size, calculating statistical significance was not reasonable.

Related research reports similarly very positive student reception on using Arduino as a development platform [50] [3]. In the case workshop, students did not appear to be overwhelmed by the new skills and technology in contrast to some related research [50]. Arguably, practices used in the case workshop, such as understanding the concept of input, data processing and output, utilizing sample setup steps and building in small steps, supported learning without frustrating students by unsustainable amount of information. While programming courses traditionally have high drop-out rates [41], the case workshop had no drop-outs, despite the non-technical audience.

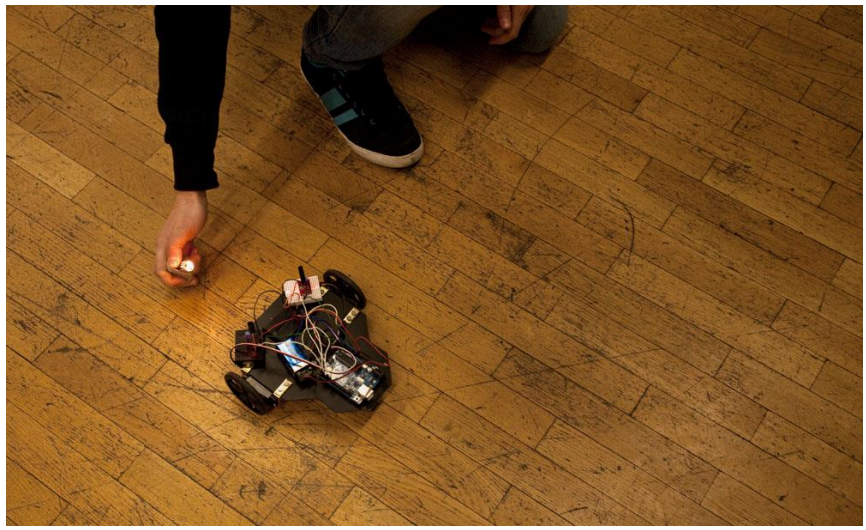


Figure 8. Learning robot prototyping in two days was tested in a robot workshop at the University of Art and Design in Linz. Here a student tests a flame following behavior.

3.4 Discussion

In respect of the thesis research question on which tools and processes lower the threshold of designing embedded systems, enabling non-engineers and novice engineers to turn their innovations into working prototypes, publication II describes a process, tools and key skills for teaching embedded systems and robotics during a two-day-workshop.

Based on successful group projects, the presented workflow is a suitable way of engaging novices in basic embedded system design. Despite the short timeframe students were able to define the functionality of their projects instead of following ready-made instructions. The practices that helped novices to deal with embedded system challenges can be summarized as follows:

- Understanding the concept of input, data processing and output
- Utilizing sample setup steps

- Learning the key skills for novice Arduino users
- Starting with “Hello World”
- Creating a mental model of the desired functionally
- Utilizing the online community resources
- Using coding style that keeps the program readable and reflects the mental model structure.
- Building in small steps
- Dealing with one problem at the time
- Compiling “Hello World” when embedded system or program does not seem to work as expected

These practices were determined in order to obtain the goals set for the workshop. It cannot be concluded that the list is complete or that some of the practices could not be left out and still achieve the desired outcomes. However, by using these practices, all goals of the workshop were met.

Even though the group projects were successful in the case workshop, many of the students needed repetitive support from the teacher in the building phase, such as advice with debugging. With a larger non-technical novice group, more time or teachers may be necessary. Also, without the pre-built chassis for the wheeled robot, the time frame would have been inadequate.

While one day for learning the basics may seem like an unrealistically short amount of time, though more likely with a more technically orientated audience, tangible results could be achieved with an even shorter introduction. In research by Jamieson and Herdtner, they describe how Arduino is introduced to electrical and computer engineering students over two 55 minute lectures [51]. After the introduction, students utilize online community resources to build the assigned projects [51].

Botelho et al. describes a pedagogical methodology for RiE that aims to help students to create a working prototype without following ready-made instructions [48]. This approach has some similarities with the process that starts with a mental model described above. Steps suggested by Botelho et al. are virtual sketch, functional sketch, concrete sketch, prototype construction and presentation [48]. These approaches could complement each other by splitting the mental model planning into virtual model and functional model. Prototype presentation including evolution process would be a natural way of summarizing the findings by different groups.

LEGO MINDSTORMS is a popular option for teaching novices robot prototyping and embedded system basics, such as in research by Kim et al. [32] or Bilotta et al. [52]. Similarly, as in the minimalistic approach described above, MINDSTORMS enables students to build prototypes without comprehensive knowledge about embedded systems [32]. The main weaknesses of MINDSTORMS are a proprietary component selection and a GUI-based programming environment. The component selection not only limits the possible outcomes but can also make prototypes less reliable [32]. The downside of utilizing GUI is that students do not learn traditional programming skills that can be applied to all major programming environments. MINDSTORMS has a syntax-based

programming option instead of GUI [53] but students may end up learning both approaches as in the course described by Kim et al. [32]. While LEGO MINDSTORMS and similar kits have drawbacks compared to the presented approach, they provide an appropriate solution for building mechanical structures, such as a chassis for a robot. For future courses, 3D printing could be used for mechanical parts. Publishing a free printable version of the chassis would also allow a straightforward use of the presented method. One viable option for a platform could be combining Arduino's functionality with commercially available Parallax Boe-Bot chassis described by Balogh R. [54].

MINDSTORMS also provides a less demanding teaching platform than the presented solution. Using Arduino combined with non-proprietary component selection requires a teacher to be more proficient with embedded system than with a fully productized MINDSTORMS package. Despite the approachability and popularity of the LEGO MINDSTORMS, presumably learning more traditional embedded systems would allow easier participation in serious development projects and produce more diverse results.

Programming is the most prominent obstacle in learning robot rapid prototyping. Platform specific reference implementations help novices to get working results that can be then altered, mixed and developed further. Creating a mental model of the program can help to define desired behavior and to divide it into smaller pieces. This also helps to see the program as a combination of small, separate challenges and then deal with one of them at the time. Suitable programming style supports the mental model and keeps the program readable.

Findings in chapter 2, “Choosing novice friendly sensors” provide guidelines on how to choose suitable sensor selection for students. The next chapter describes a novice friendly method for more advanced and specialized embedded system prototyping.

4. IoT Rapid Prototyping Laboratory Setup

Successful IoT prototyping requires a diverse skill set, including embedded systems knowledge combined with using network and servers. With suitable workflow and a development environment, necessary basic skills that allow turning ideas into working prototypes can be taught in four days to novice engineers and even to students who are not technically oriented. IoT is a particularly interesting prototyping area as IoT development is expected to be an important leap in the ICT sector, influencing a vast amount of different fields, such as smart cities, environmental monitoring, health-care, and security [55], [56].

This approach allows students to use and learn the new technology in a meaningful context. Prototyping, design and project oriented approaches have successfully been used in engineering education [9], [57].

The presented approach enables a fast prototyping cycle, using a common and well-established development board and a computer. Arduino Uno is used for prototyping and a Python program running on the same computer handles the needed Internet communications. Related research shows that Arduino is used for purpose of engineering education for its accessibility, adaptability and compatibility [2], [23], [4], [58].

It is common to add extra components to Arduino to enable wireless connectivity, such as using Wi-Fi Shield or ESP8266 [59]. This raises the difficulty level as it demands more complex setup with a wireless connection. The presented method makes it possible to learn the needed basic skills in manageable steps, allowing students to focus on the actual prototype instead of struggling with the wireless and Internet communication problems.

After the prototyping phase, a device can be easily ported to an inexpensive and small ESP8266 based microcontroller. Compared to developing IoT prototypes directly with ESP8266, the setup presented is considerably faster. The whole process is based on free software tools which provides a possibility to utilize prototypes commercially, without a risk of a third party changing or discontinuing services.

We arranged a four-day-workshop at the University of Lapland with art students from diverse backgrounds and varying levels of technical skills (n=19). The students were from various departments, such as art education, industrial design and interior & textile design. Regarding technical skills, a majority reported that they had not received a single credit from programming course before. The main goal was to teach embedded system and IoT basics allowing students to design and build their own IoT-devices. The course was divided into

teaching basic skills for the whole class and to supporting building projects by two teachers. Similarly, as in the Lintz workshop, the defined timeframe, four days, was short given the complexity of the topic. The aim was to see if it was possible for the target group to build functional IoT-devices in the given time without ready-made and pre-defined outcomes. One of the goals was also to observe whether the drop-out rate could be kept at zero despite the technical topic and non-engineer audience. Moreover, an aim was to find out how the topic was received overall and if self-evaluations of the students' own programming skills changed during the course.

All teams successfully built a working IoT prototype based on their own ideas. The experiment was later repeated with another group (n=27) at a university of applied sciences, getting similar results. Results indicate that this method is effective for learning IoT prototyping skills during a 4-day-workshop.

Lessons learned in chapter 2 "Choosing novice friendly sensors" and chapter 3. "Teaching robot rapid prototyping for non-engineers - a minimalistic approach" are useful for teaching basics of embedded systems without cloud connection as well as for choosing suitable component selection for education.

Publication III makes three key contributions:

1. Approach that enables novice students to turn their ideas into working IoT prototypes during a four-day-workshop.
2. Defining necessary basic skills for learning and prototyping IoT.
3. Open source setup for IoT prototyping utilizing a development board and a computer.

4.1 Teaching IoT prototyping

Supporting the goal of the workshop, basic skills for learning and prototyping IoT are defined as a skill-set that enables students to build IoT device prototypes based on their own designs. As the area is broad, the aim is to leave out everything that is not necessary to attain this goal. Figure 10 summarizes these skills.

Like all embedded systems, [60] IoT devices consist of input, processing and output. Unlike traditional embedded systems, IoT devices can have input, output or both connected to cloud as shown in Figure 9. In addition to cloud connections, an IoT device can have local sensors and outputs, but without connection to cloud, the system would not qualify as an IoT device.

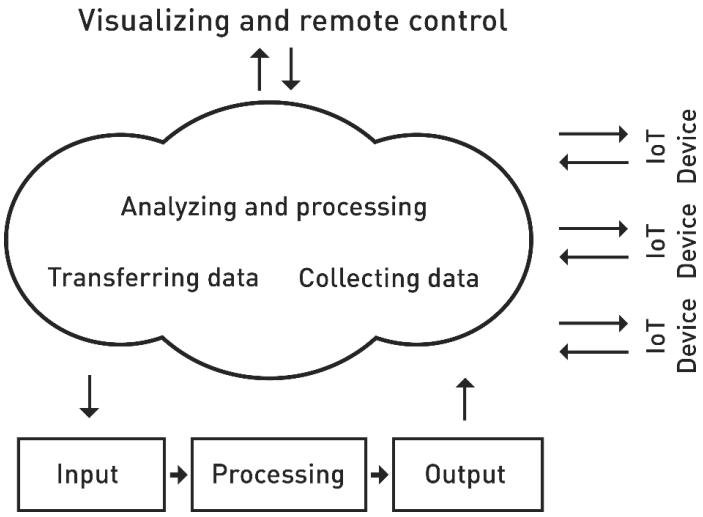


Figure 9. Structure of an IoT device.

IoT device structure can be divided into “embedded system”, “embedded system and cloud communication” and “cloud deployment and development”. Figure 10 presents this subdivision including the most central skills needed in building a basic embedded system and in creating a foundation for understanding how IoT devices operate. Before moving to cloud communication students need to have a basic understanding of how embedded systems work. It is necessary to understand how input, processing and output work separately and together. In-depth understanding of programming is not necessary, but students must be familiar with basic program structure and syntax. Without any command of these skills, utilizing reference implementations and online community resources would be challenging. IDE's built-in serial monitor is needed for debugging as well as for understanding serial communication when communicating with cloud. Teaching basic embedded system skills is described in detail in chapter 3: “Teaching robot rapid prototyping for non-engineers - a minimalistic approach”.

After understanding the basics of input, processing and output, it is effortless to develop the embedded system into an IoT device with cloud communication. Writing to cloud is not significantly different from using any traditional output: instead of outputting data to Arduino’s pins, a value is written to the serial port. Python proxy program then transfers data to the server. To read from the cloud, data is received by writing request to the serial port, instead of reading it from a local sensor.

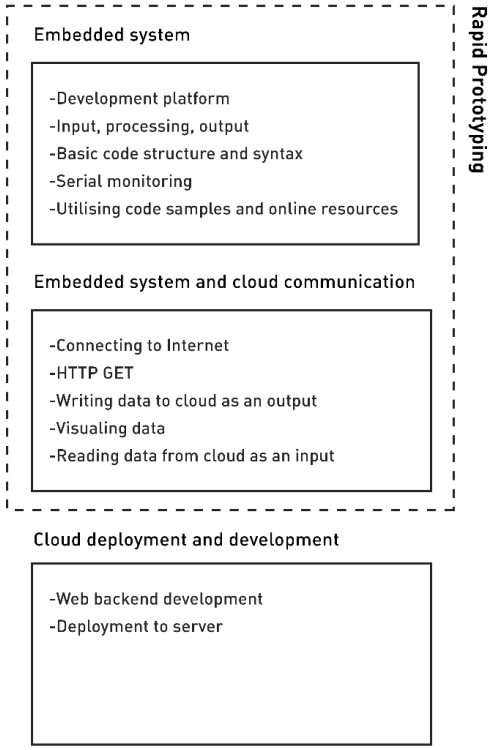


Figure 10. IoT prototype building basics.

With the presented setup, IoT prototyping is possible without server programming, as the internet communication is limited to passing floating point numbers. Students only need to understand how the communication works between the cloud and the embedded system. On day four there was an optional cloud deployment and development learning session for advanced students.

For utilizing the setup, a computer with Arduino IDE and Python 3 installed is required. All software required is free and open source. Necessary programs can be installed to Linux, Windows or OS X making it suitable for almost any environment. The agenda of the workshop is presented in Table 5.

TABLE 5
WORKSHOP AGENDA

Day 1 Technology, theory and personal innovation	**Day 2** Utilizing and publishing results	**Day 3** Debugging and cloud services	**Day 4** Demo day
• Preliminary knowledge review • Introduction to IoT • Designing embedded system • Designing IoT device • Successful prototyping process • Group project plan • Presenting project plan	• Serious results with accessible tools • Documenting projects • Publishing • Building the group project	• Common problems and solutions • For advanced students: cloud service deployment and operation • Building the group project	• Finalizing the prototype • Building the group project • Group project presentations • Feedback

4.2 Setup for prototyping

The prototype is built with Arduino Uno which is connected to a desktop computer running Arduino IDE as shown in Figure 11. Only one cable is needed for connecting Arduino to a computer, while all data is written to a serial port and read from the serial port by a Python program run on the computer. As the setup is wired and limits the number of needed physical components, there is a reduced change for user errors. Implementation components are shown in Table 6.

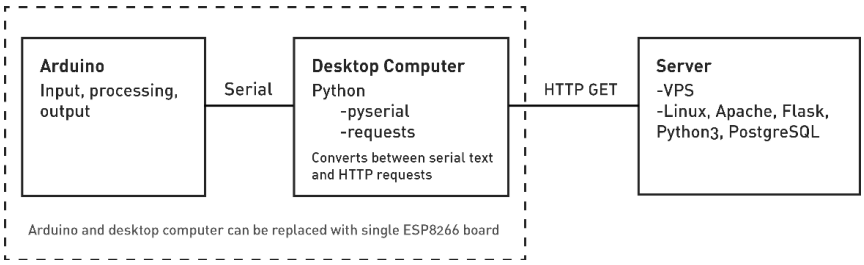


Figure 11. Setup for prototyping.

When developing prototypes, students only write code in Arduino IDE. The program is compiled and uploaded with a single click, making the development cycle fast and straightforward. This also allows students to focus on one programming language and environment. Editing the Python proxy or the server end is not mandatory for successful prototyping. Python Proxy does however provide an additional and useful tool for debugging as all requests are shown on it.

Table 6
Components chosen for implementation

Hardware	Arduino Uno	Desktop computer	Virtual Private Server
OS	No OS	Linux/Windows/Mac	Linux
Software	Student's code	BotBookEspy	http://one.api.botbook.com, Apache, Flask, PostgreSQL
Language	C++	Python 3	Python 3, SQL

Arduino can either send a value to a server or ask for a value from a server via the Python proxy. Server API (application programming interface) supports three types of GET requests: storing a data point, returning value for last data point and returning all stored data points, as shown in Table 7. An API key is used for authenticating the requests. When asking for data, Python proxy only forwards the first line of HTTP response body to Arduino containing the data point value in a string form. From the user's point of view this is not noticeably different from reading a value from a local sensor.

Table 7
API endpoints provided by BotBookAPI v 1.0.0

Path	Feature
/add/<addkey>/?x=	Store a data point [currentTime, x]
/last/<viewkey>	Return x value for last data point
/json/<viewkey>/index.json	Return all stored data points

The server handling GET requests is configured to be used as a backend for a web page. A static web page on an external server can perform AJAX (asynchronous JavaScript and XML) requests, enabling students to develop web and mobile interfaces for their devices. Students can utilize any web tools and formerly learned skills to build web interfaces or even cell phone apps by using mobile application development frameworks, such as Apache Cordova [61]. This feature was applied by students both in creating control and visualizing web interfaces, as shown in Figure 12. Importantly, web tools are constantly evolving, and they are also widely used outside the engineering domain. This allows students to utilize formerly learned skills to elaborate their IoT-device.

Figure 12. Student made, visual web interface reacting to a number of Bluetooth devices present in another space. This is an example of utilizing common web tools for creating an aesthetically pleasing presentation of the IoT-device measurements.

The server code is based on free tools stack (Linux, Apache, PostgreSQL, mod_wsgi, Python 3, Flask) and was provided for students. This allows

developing the system further and keeping the control of the IoT solution without third party commercial operators. During the workshop, one advanced student also created his own backend with similar architecture utilizing LAMP (Linux, Apache, MySQL, PHP) stack.

4.3 Standalone operation and miniaturization with ESP8266

Internet access is programmed into separate functions, which allows a user to replace them and connect to the internet without a computer or Python proxy. With minor program changes, the prototype can be developed to a wireless standalone device using, for example, Arduino WiFi Shield or ESP8266. Taking advantage of this workflow, IoT prototypes can be developed reliably with a robust wired setup.

The ESP8266 is a small low-cost Wi-Fi capable microprocessor, designed by Espressif systems [62]. It is possible to miniaturize the whole IoT device into a single ESP8266 board. In addition to an accessible and reliable setup, using presented workflow also enables a faster prototyping cycle compared to developing with ESP8266 from the start.

ESP-12E and ESP-01 were tested with sample code to find out how fast program uploads they would enable. The sample program had 9 lines of code. Using Arduino IDE with ESP-12E, the upload took about 30 seconds including time to compile the program. With ESP-01 the upload time was approximately 34 seconds. Upload to Arduino Uno took under 4 seconds. While the difference is not remarkable for a single upload it clearly has an effect on the prototyping speed as the program is changed and uploaded repeatedly during the development.

4.4 Case workshop assessment

There were no drop-outs during the course indicating that the presented workflow was suitable for the target audience. The most important result to measure however was the student groups' ability to design and build working IoT prototypes. In the case workshop, all student groups were able to design and build a working prototype based on the assignment. Final prototypes included, for example, a shirt with sleeves controllable by a cell phone, imitation candle that lights up when a real candle is lit up in a different location and a felt owl that reacts to the number of Bluetooth devices present in another space (Figure 13).

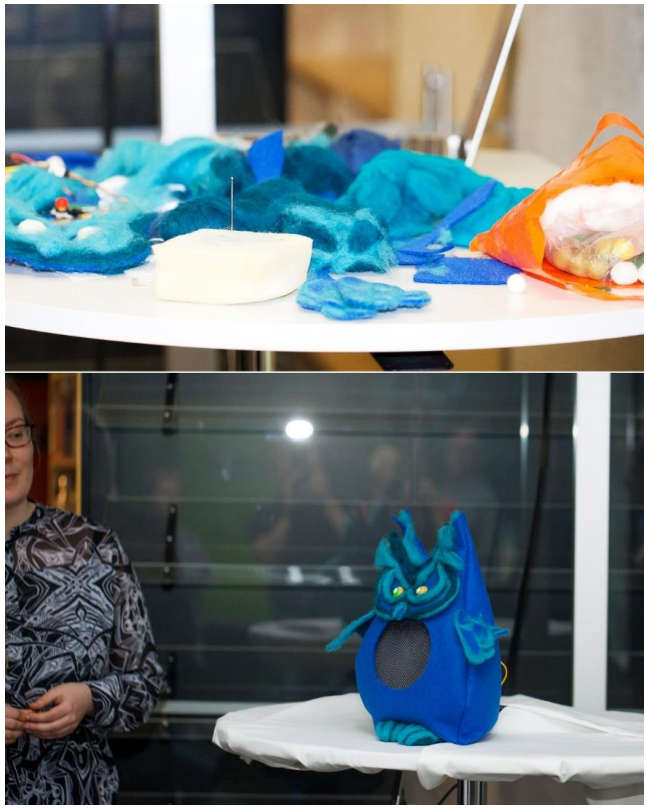


Figure 13. Student group turning felt into an animatronic IoT owl.

The most notable challenge during the workshop was a lack of technical, and especially programming, skills. Some students had virtually no experience in those areas. On the other hand, most groups managed to utilize skills learned from their own domains, such as designing clothes, and combining them to IoT prototypes. Students also started fluently innovating concepts when asked to plan an IoT device based on their own idea.

To see how the workshop was received overall and if self-evaluations of the students' own programming skills changed during the course, data was collected from the students by a preliminary and a feedback questionnaire. In the preliminary questionnaire students reported that they are interested in learning IoT, for example, to combine embedded systems with their art, to make interactive pieces, building home automation, to learn prototyping tools and to combine electronics to textiles. An example of the most challenging topic in IoT that students mentioned was programming and connecting devices to Internet. Seven of the students also reflected that their main challenge was the total lack of knowledge and experience on the subject.

Despite the technical topic, the reception and evaluation of the workshop was positive. On a scale of 1 to 5 (5 being the most positive), students scored the course 4.3 on average. All participants would also recommend the course to other students. In the preliminary knowledge review, over half of the students evaluated their programming skills with the lowest available number 1 while the

average rating was 1.5 (n=19). In the feedback, the average was raised to 2.5 and no one selected the lowest rating. The result was statistically significant at $p < .05$ calculated with Wilcoxon Signed-Rank test. Again, this more likely indicates change in self-efficacy than in actual objective programming skills. It is noteworthy that every self-evaluation rating either raised or stayed the same.

In the feedback questionnaire students indicated that the single most challenging topic in IoT for them was, for example, programming generally, combining programs, understanding server-side, understanding syntax, python and debugging. Related research also brings out that students with limited technical skills can be overwhelmed by the Arduino even when they have an overall positive impression of the platform [50]. In the case workshop, guidance by teacher accompanied with practises presented in chapter 3 helped to keep the students from frustrating with the embedded system and IoT challenges.

According to feedback the most interesting things during the workshop were, for example, problem solving, individual support, Arduino in general, getting things to work despite the lack of prior experience and realizing new possibilities. Examples of the most complex things at the workshop that students listed included, combining a device with a website, learning a completely new domain, programming challenges, understanding causalities, the fast pace of teaching, choosing the right components and server operation.

Based on feedback and successful projects, the chosen approach was appropriate for the diverse group. However, the pace of the workshop including many new skills and techniques for most of the students, render it relatively demanding for novices. If the group would have been less motivated or there would have been less individual support this could have caused some students to give up.

After the pilot workshop, the course was repeated with another group (n=27) in Haaga-Helia University of Applied Sciences by Tero Karvinen, with similar results. All groups successfully built a working IoT prototype during the workshop. Differences in educational background showed in prototyping results between the two workshops. Art students focused more on aesthetic factors while ICT students developed more advanced technical solutions. In Haaga-Helia University, students rated the workshop 4.8 on average on a scale of 1 to 5. Change in self-evaluation on programming skills was not measured for this course.

4.5 Discussion

Regarding the thesis research question, publication III presents tools, process, open source setup and a skillset that enables non-engineers and novice engineers to turn their innovations to IoT-devices prototypes.

The teaching method with the setup combining Arduino, Python Proxy and desktop computer for IoT prototyping was successful. Students were able to design and build working prototypes based on their own ideas. The presented approach extended the traditional embedded system prototyping process by adding a possibility to communicate with the cloud, without raising the threshold too high for novice use.

The practices used in the workshop can be summarized as follows:

- Using a common and well-established development board and a computer
- Using free software and open source tools
- Teaching the basic program structure and syntax
- Using wired setup for prototyping
- Using process that does not demand server programming
- Understanding the concept of input, processing and output
- Utilizing code examples and Internet resources
- Utilizing text output of the Python proxy program and Arduino IDE's serial monitor for debugging
- Using GET to send and receive data

Wired platform effectively eliminated random errors which could be caused by wireless protocols and components. The workflow did not require server programming which allowed students to focus on developing their prototypes with Arduino. Non-engineers and novices could benefit from preliminary embedded system workshop, as most issues derived from the lack of basic technical skills. Writing a working syntax is a major novice challenge. Even though the approach leaves all unnecessary technical details out, the combination of programming, embedded systems, new concepts and techniques can become overwhelming for a novice student. Personal support from a teacher or teachers can be invaluable to solve the issues before they become insurmountable.

Using GET to send and receive data was easy to understand and advancing from embedded system to IoT device was not a prominent challenge. Several groups also utilized the possibility to use IoT server as the backend for web page interfaces.

The prototyping cycle is considerably faster when prototyping is done with Arduino instead of ESP8266. When tested the upload time from Arduino IDE to ESP-12E was 750% of the upload time to Arduino. Setup allows the prototyping with the wired setup, and with minor program changes, the prototype can be changed to a wireless standalone device. Time frame did not allow students to further develop their prototypes into standalone ESP8266 devices, leaving that phase to be tested on future courses.

Combining student-made devices with available IoT-devices and platforms could make prototypes more versatile and allow students to focus on designing the missing parts of the IoT-ecosystems. Several applications can be connected to the setup described above simply by sending and receiving GET-requests. One of the student groups utilized this feature for timing device events by loading the HTTP API from the cell phone calendar event. With a suitable setup, it would be possible, for example, to enable a student made prototype to control Philips Hue lights or react to a click of a Flic button.

There are various solutions available for controlling IoT devices from separate ecosystems. IFTTT (If This Then That) is a popular web service that connects other web services and IoT devices to each other [63], [64]. IFTTT demands

permission for the services it controls, including the personal data associated with the accounts [64]. This may raise privacy concerns in some users making it an unsuitable solution.

Li et al. propose the mobile programming system Epidosite that uses smartphone as a hub for IoT automation [65]. Epidosite connects different devices by manipulating their mobile apps, aiming to provide a solution mainly for smart home automation [65]. Programming automation is done by the user demonstrating the desired set of actions which the system records [65]. Combining a smartphone hub with automation applications, such as Tasker [66] or Automagic [67], would allow creating automated workflows utilizing a diverse selection of IoT devices combined with cell phone or tablet functionalities. Local hub could also be built so that the user does not need to give all application permissions to a single third party service provider.

The next chapter (chapter 5: “Using hobby prototyping boards and commercial-off-the-shelf components for developing low-cost and fast-delivery satellite sub-systems”) shows how the Arduino prototype can develop from an early prototype into a space ready device.

5. Using hobby prototyping boards and commercial-off-the-shelf components for developing low-cost and fast-delivery satellite sub-systems

The use of small satellites has increased remarkably in the past years and the community participating in the process is growing [68]. The lower cost has enabled satellite and subsystem development for new groups such as start-up companies and universities [69]. The traditional space industry has also started to use faster, better and cheaper (FBC) approaches to increase the number of research missions without raising costs [70].

Publication IV proposes a development process that allows participants with multidisciplinary backgrounds to contribute to satellite projects. Development is undertaken by combining commercial off the shelf components (COTS) with readily available open source hobby development board (Arduino).

The use case of the Aalto-1 sun sensor prototype development using this process is presented. Requirements for the sensor are defined by an internal development group while the prototyping is executed by an external group. The use case shows how an Arduino prototype can evolve from the early tests into a space ready component.

Publication IV makes two key contributions:

1. Demonstrating a process for developing low-cost, fast-delivery satellite subsystems using commercial off-the-shelf (COTS) components.
2. Presenting use Case: Aalto-1 Sun Sensor Subsystem Prototype Development.
    - Sun sensor is one of the smallest and cheapest satellite sun sensors available.
    - Use case shows that despite the DIY background, hobby prototyping board Arduino can be used for serious prototyping.

The research question defined that “The workflow should allow experimental prototypes to evolve to deployable embedded systems.” In this regard the most interesting aspect of the publication IV is whether the same accessible tools that are suitable for novice prototyping, are usable for developing functional final

products. As presented earlier, basic Arduino and embedded systems skills can be learned in a relatively short amount of time. While being accessible, Arduino can be directly used as a part of commercial and research hardware [31]. This narrows the gap between early prototypes and deployable embedded systems.

Satellite subsystem prototyping also provides an interesting use case as the requirements of a space instrument are more demanding than in most casual embedded systems. These requirements rule out most toolkits aimed for learning that include a proprietary component selection.

5.1 Small satellite subsystem Development Process

Research and public information of the small satellite projects and about the actual practices used in development and testing is limited. Publication IV contributes to this field by describing both the organization and the technical solution of the development project.

The organizational structure used in Aalto-1 nanosatellite project is shown in Figure 14. The internal group is formed of space technology experts responsible for project planning, specifications, management, and the final implementation. External groups contribute to individual subprojects and can be formed of interdisciplinary participants. This structure allows for the bringing in of outside expertise and ideas into the project, while controlling the quality of the outcome.

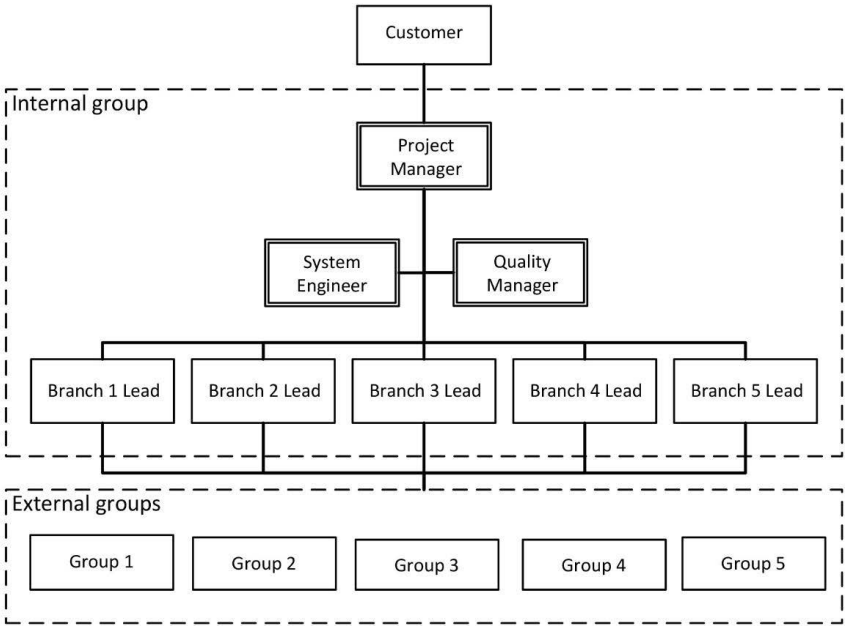


Figure 14. Small satellite project organizational structure.

The internal group defines requirement specifications for the subsystems, as shown in Table 8. The external group then proposes suitable components,

designs a prototype and carries out prototype verification. The final design and final verification are performed by the internal group.

Table 8
Sun sensor subsystem key requirements defined by the internal group.

Requirement	Value
Mass	30 g
Power	30 mW
Dimensions	6 mm * 6 mm * 6 mm (external)
Interface	I2C
Performance	5 deg accuracy (1σ) in 90 deg FOV
Environmental durability	Standard / Analysis

Given requirement specifications control how innovative the outcome can be. Stricter requirements increase prototype suitability for the main project, but at the same time they diminish the possibility for new innovations and unconventional approaches.

5.2 Open-source hobby development platform and sun sensor subsystem prototype development

Using a hobby development board, such as Arduino, lowers the threshold of participating in a satellite project as it solves many of the low-level requirements, such as connecting input and outputs. This is especially valuable when the external group members are not embedded system professionals.

After the prototyping phase, it is usually necessary to optimize the component price, mass and volume. As Arduino uses the ATmega microcontroller, it is possible to use the same program, microcontroller and sensor in the final product. Often support electronics used in the development platform can be left out, making the device considerably lighter and smaller.

Requirement specifications in Aalto-1 prevented the use of commercially available sun sensors, as the external envelope of the component had to fit in 6 mm * 6 mm * 6 mm space.

The external group proposed using Elmos E910.86 Integrated Solar Angle Sensor [71] combined with Arduino. The approach was validated by the internal group before moving on to the prototyping phase.

Elmos E910.86 is a tiny surface mounting sensor that can not be connected to Arduino by using breadboard or pin headers. As it also demands adding resistors and capacitors, a custom protoboard was manufactured, as shown in Figure 15.

E910.86 uses Serial Peripheral Interface (SPI) to communicate with Arduino. As noted in the article “Choosing novice friendly sensors” [49] SPI is a loosely defined protocol making its implementation relatively demanding without suitable documentation. Writing the SPI interface for Arduino based on its datasheet took over one week of work. For an external group with less experience in programming, this would likely have been an insurmountable challenge. This also emphasizes how sensor selections have an effect on the level of the difficulty of a project. For comparison, an unrelated SPI component, a HMC5983 magnetometer from Honeywell, was tested by the external group. By using platform-specific reference implementation, it was working in less than 10 minutes.

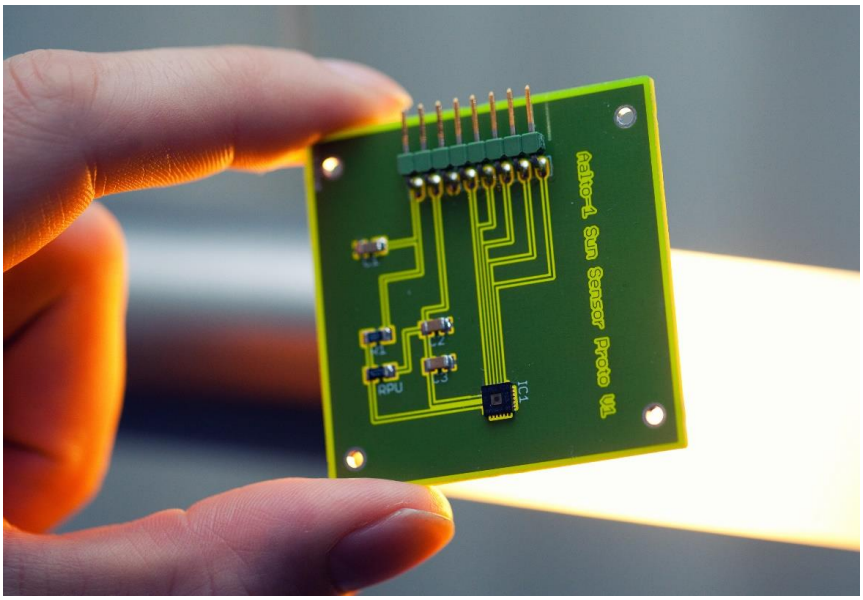


Figure 15. Custom protoboard for Elmos E910.86 Integrated Solar Angle Sensor.

In the flight-ready satellite, the onboard attitude determination and control system (ADCS) will ask for a sun vector from the sun sensors. As the ADCS was not yet available during the prototyping process, one Arduino was built to mimic ADCS to allow early testing for the sun sensor, as shown in Figure 16. Arduino was programmed to communicate with two different protocols to support the sensor’s SPI and the I2C used by ADCS.

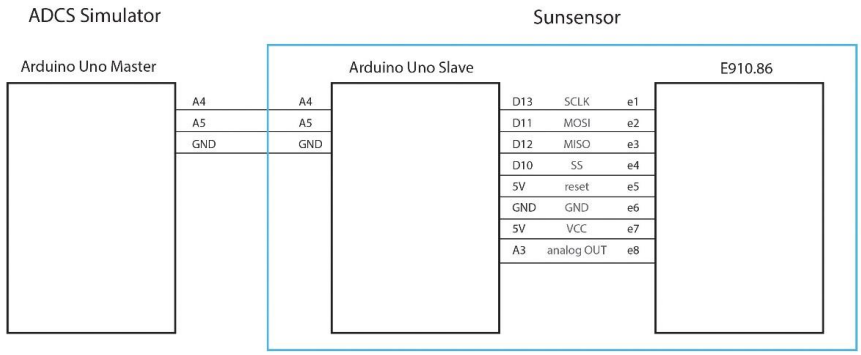


Figure 16. Sun sensor subsystem connected to an ADCS simulator.

After the prototype fulfilled the given requirement specifications tested by the external group, an environmental test campaign was conducted by the internal group. The sensor was successfully thermal-cycled while being operational for one week from −70 to +100°C.

In radiation testing the sensors stopped working after being exposed to a total dose of 11 krad. As this was not adequate for the mission, a cover class was added to protect the sensor. After the modification, the sun sensor was successfully re-tested for accuracy and the final version was designed, as shown in Figure 17.

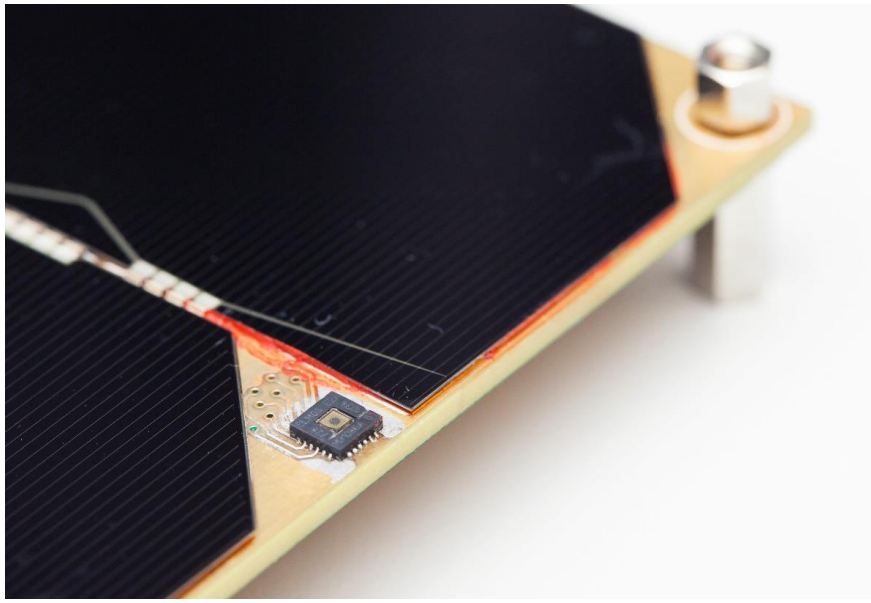


Figure 17. Final version of the sun sensor.

5.3 Discussion

The development process and Arduino as a development platform were suitable for the satellite sensor subsystem development. The final space-instrument was implemented on the same hardware as Arduino Uno and utilized the program from the prototyping phase. This also made early testing possible and validated the program and the component choices. As this case example only presents the building of one subsystem, the results cannot be generalized.

The case shows that despite the strong hobby and DIY background, Arduino is a viable option for serious prototype development, allowing prototypes to evolve to deployable embedded systems. Arduino has also been successfully utilized by other groups for accessible satellite development, such as described in research by Geeroms et al. and Holman et al. [72], [73]. The prototype described could not have been developed with available proprietary kits as it required using components that fit to requirement specifications. Also, the software development needed to communicate with the SPI sensor would not have been possible with commonly available programming environments that have graphical user interface instead of syntax. This should not be interpreted as meaning that proprietary kits or tools with graphical user interface would not be suitable for any serious prototyping. Rather it shows that tools described in earlier chapters are flexible enough to suit real-life prototyping needs.

Related research describes Arduino as a very student-friendly development platform for small satellite development, being sufficiently easy for beginners and versatile enough for more advanced users [74]. In the case project, the requirement specification for the sun sensor demanded using components that raised the required skill level for the prototype. With a less technically oriented group this could have led to failure with given requirement specifications. The experience level and need for guidance of the external group should be estimated in each subproject. The challenges caused by the surface mounting SPI sensors are consistent with the findings presented in chapter 2 “Choosing novice friendly sensors”.

The final subsystem has an external envelope of only 6 mm * 6 mm * 1 mm for integrating its sensor part, allowing it effortlessly to fit to solar panels and with it being one of the smallest and inexpensive satellite sun sensors available at the time of publishing. The satellite sun sensor prototype tutorial was published online along with the source codes to enable other groups to utilize and further develop the sensor [75].

6. Conclusions

This thesis makes two key claims:

- With the presented tools and processes, the threshold of designing embedded system prototypes can be lowered, enabling non-engineers and novice engineers to turn their innovations into working prototypes.
- Presented tools and processes create a foundation that enables innovation to develop from the first experimental prototypes to deployable embedded systems.

  The key components presented in the previous chapters are:

  - Utilizing non-proprietary open-source software with open hardware.
  - Using hardware that is suitable for both prototyping and production.
  - Utilizing the online community resources.
  - Choosing novice friendly sensors.
  - Learning programming with syntax revealing IDE.
  - Utilizing a presented minimalistic approach for learning basic embedded systems to achieve instant tangible results.
  - Developing IoT-prototypes with a presented wired setup that does not demand server programming or using multiple programming tools.
  - Utilizing an expert internal group for validating contributions of the interdisciplinary external group.

Choosing suitable sensors for a novice use helps to prevent various issues that beginners face when getting started with embedded systems, as is discussed in chapter 2. Guidelines for selecting novice friendly sensors and to overcome novice learning barriers are defined.

In chapter 3 a minimalistic approach for learning basic embedded systems and robot prototyping is presented. With the workflow presented, the basics can be learned in one day. The use case shows that even with limited knowledge and skillset, novices can design and build simple devices based on their own design.

IoT prototyping demands tools and skillset beyond basic embedded system prototyping. Chapter 4 introduces an approach that enables novice students to turn their ideas into working IoT prototypes during a four-day-workshop.

Chapter 5 presents a process for developing low-cost, fast-delivery satellite subsystems using commercial off- the- shelf (COTS) components combined to Arduino platform. The use case shows how Arduino prototypes can evolve from early tests into ready products.

In the case workshops, all student groups successfully completed given prototyping tasks. In contrast to the complexity of the area, pilot workshops show that with suitable tools and workflow, learning to build basic embedded system prototypes can be surprisingly fast and easy. In addition to tangible results, reception of the subject, including typically burdensome areas such as programming, was very positive. Students' self-evaluation of their programming skills also raised during the workshops.

Student self-evaluation on their programming skills likely tells more about self-efficacy than objective skill-level. The skill-level of the group and educational background can also have a substantial impact on this. In a non-technical group, a mediocre programmer can seem advanced compared to the beginners. On the other hand, objectively advanced students can be modest about their knowledge and skills.

Most importantly, the feedback reveals that students found the technical subject useful and felt that their programming skills had increased. This positively affects students' self-efficacy and creates a foundation for learning more in the area that is traditionally considered difficult.

Presented processes offer a different approach to the multidisciplinary embedded systems teaching challenges compared to the other solutions described in related research. Often easy accessibility components with limited functionality, such as LEGO MINDSTORMS, are offered for non-engineer novice prototyping, [32]. Even though these types of systems allow novice rapid prototyping, arguably some very basic embedded system skills, such as using protocols and connecting non-proprietary components, are not learned by using them. Proprietary component selection can also hamper even a basic prototyping process [32].

In contrast, platforms and tools used in the previous chapters are free and open-source software (FOSS) with open hardware. As proprietary tools or components are not required, the approach is suitable for various environments and for further development.

The strength of the dedicated easy accessibility prototyping kits is that they can be used to create tangible results very quickly, as shown in the research by J. Sadler, using modular sensor system Bloctopus [11].

When learning programming, some processes hide the syntax from novices and a graphical user interface (GUI) is provided instead. Block-based programming languages like Scratch and Blockly are popular examples of this approach [76], [77]. While hiding code can make code less intimidating for some groups, it prevents users from learning some of the most central skills needed to develop more advanced prototypes. The ability to read and write syntax is invaluable as

it can be applied in different environments with different tools, and also allows for the development of early prototypes into ready products. In most environments it is not possible, for example, to edit code examples, combine programs, take advantage of community resources and to perform debugging without some level of programming skills. By including the syntax in the novice learning, combined with a setup that allows instant tangible results, it can help to lower sociological barriers on programming [25] and to make it a natural part of the design process.

In their research, Jamieson and Herdtner bring out a valid question on how embedded system student projects that utilize open source projects should be assessed [51]. This issue is emphasized especially with electrical engineers who should understand the inner workings of the embedded systems instead of just combining and editing existing open source resources. Jamieson and Herdtner suggest some solutions [51] that could be useful when assessing results created by the processes presented in this thesis. In particular, the approach where students document the code used, combined, added and designed [51], would presumably be well suited for multidisciplinary courses. It would make the assessment of the projects easier as well as clarify the process for students themselves.

6.1 Application and future research

The findings are largely usable with other breakout toolkits that do not have a graphical programming interface or a proprietary component selection. The methods and findings of this thesis should be adapted to the target groups skill level. In particular, students' programming experience should be considered. With novice students, the focus should be on learning embedded systems basics and on turning their ideas into simple prototypes. On the other hand, advanced students can benefit from the presented workflows and tools, that allow rapid prototyping by focusing on design instead of technical challenges.

The workshops and the case study were executed at a university level. Findings from "Choosing Novice Friendly Sensors" and "Teaching robot rapid prototyping for non-engineers - a minimalistic approach" could also be particularly useful at a significantly lower educational level, such as in high schools.

Future research and workshops could investigate how the presented method is suited for diverse audiences and what kind of benefits could be achieved by varying the goals, timeframe and educational background. Longer workshops would allow prototyping more advanced and finalized devices. However, the short duration of the case workshops prevented undertaking a follow-up to learn if the students would have been able to elaborate their knowledge and skills beyond embedded system basics.

Chapter 4 introduced an approach for a single more advanced and specialized embedded system area. Similar workshops could be designed for other interesting areas, such as swarm intelligence or embedded systems utilizing neural networks. Workshops suitable for novices would help to bring interdisciplinary contributions to the new areas.

As our case study was limited to building one satellite subsystem, it cannot confirm the advantages of using external groups for increasing innovation and knowledge. To test this, several interdisciplinary groups could be used to simultaneously provide different solutions for the same requirements or problem. Using loose requirement specifications would allow the possibility for more innovative and surprising approaches. To utilize interdisciplinary external groups, one solution especially related to an academic setting would be to arrange a workshop for the target groups and focus on building prototypes within requirement specifications. This would allow provision for technical support for the external group and, on the other hand, enabling collecting possible new innovations and insight for the R&D internal group.

References

[1] C. Severance. “Massimo Banzi: Building Arduino.” *Computer*, vol. 47, pp. 11-12, Jan. 2014.

[2] A. Soriano, L. Marin, M. Valles, A. Valera and P. Albertos. “Low Cost Platform for Automatic Control Education Based on Open Hardware.” *IFAC Proceedings Volumes*, vol. 43, pp. 9044-9050, Dec. 2014.

[3] P. Jamieson. “Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat?” in *Proc. FECS*, 2010, pp. 289-294.

[4] C. Parikh. “Introducing Arduino Platform to Sophomore's using an apt recipe,” in *Proceedings of the 2014 ASEE North-Central Section Conference*, 2014, pp. 1-8.

[5] E. K. Huizingh. “Open innovation: State of the art and future perspectives.” *Technovation*, vol. 31, pp. 2-9, Jan. 2011.

[6] H. Chesbrough, W. Vanhaverbeke and J. West. *Open Innovation A New Paradigm*. Oxford, UK: Oxford University Press, 2006, pp. 0-19.

[7] L. Zhao and S. Elbaum. “Quality assurance under the open source development model.” *Journal of Systems and Software*, vol. 66, pp. 65-75, Apr. 2003.

[8] C. Anderson. *Makers: The New Industrial Revolution*. USA: Crown Business, 2012, pp. 99-118.

[9] S. Chandrasekaran, A. Stojcevski, G. Littlefair, and M. Joordens. “Project-oriented design-based learning: aligning students' views with industry needs.” *International Journal of Engineering Education*, vol. 29, pp. 1109-1118, Mar. 2013.

[10] A. Bonarini, M. Matteucci, M. Migliavacca and D. Rizzi. “R2P: An open source hardware and software modular approach to robot prototyping.” *Robotics and Autonomous Systems*, vol. 62, pp. 1073-1084, Jul. 2014.

[11] J. A. Sadler. “The Anatomy of Creative Computing: Enabling Novices to Proto-type Smart Devices.” Doctoral dissertation, Stanford University, USA, 2016.

[12] T. Booth. “Making Progress: Barriers to Success in End-User Developers' Physical Prototyping,” in *Visual Languages and Human-Centric Computing (VL/HCC), 2015 IEEE Symposium*, 2015, pp. 299-300.

[13] T. Booth and S. Stumpf. “End-user experiences of visual and textual programming environments for Arduino,” in *International Symposium on End User Development*, 2013, pp. 25-39.

[14] P. Blikstein. “Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future,” in *Proceedings of the 12th international conference on interaction design and children*, 2013, pp. 173-182.

[15] ”Adafruit.” Internet: https://www.adafruit.com/, [February 26, 2019].

[16] ”Sparkfun.” Internet: http://www.sparkfun.com/, [February 26, 2019].

[17] L. E. Winslow. “Programming pedagogy - a psychological overview.” *ACM SIGCSE Bulletin*, vol. 28, pp. 17-22, Sep. 1996.

[18] G. Fischer. "End-user development and meta-design: Foundations for cultures of participation," in *International Symposium on End User Development*, Springer, Mar. 2009, pp. 3-14.

[19] “ISO 8373:2012 (en), Robots and Robotic Devices – Vocabulary.“ Internet: https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-2:v1:en, [January 20, 2018].

[20] A. Repenning and A. Ioannidou. “What makes end-user development tick? 13 design guidelines,” in *End user development*, Springer, 2006, pp. 51-85.

[21] D. Malan, and H. Leitner. "Scratch for budding computer scientists." *ACM Sigcse Bulletin*, vol. 39, pp. 223-227, Jun. 2007.

[22] M. Armoni, O. Meerbaum-Salant, and M. Ben-Ari. "From scratch to “real” programming." *ACM Transactions on Computing Education (TOCE),* vol. 14, pp. 25, Feb. 2015.

[23] A. Araujo, D. Portugal, M. S. Couceiro, and R. P. Rocha. “Integrating Arduino-based educational mobile robots in ROS.” *Journal of Intelligent & Robotic Systems*, vol. 77, pp. 281-298, Feb. 2015.

[24] P. Hertzog and A. Swart. “Arduino-Enabling engineering students to obtain academic success in a design-based module”, in *Global Engineering Education Conference (EDUCON)*, IEEE, 2016, pp. 66-73.

[25] C. Kelleher and R. Pausch. “Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers.” *ACM Computing Surveys (CSUR)*, vol. 37, pp. 83-137, Jun. 2005.

[26] J. F. Pane and B. A. Myers. “Studying the language and structure in non-programmers' solutions to programming problems.” *International Journal of Human-Computer Studies*, vol. 54, pp. 237-264, Feb. 2001.

[27] A. J. Ko, B. A. Myers and H. H. Aung. “Six Learning Barriers in End-User Programming Systems,” in *Visual Languages and Human Centric Computing, 2004 IEEE Symposium*, 2004, pp. 199-206.

[28] E. Lahtinen, K. Ala-Mutka and H-M Järvinen. ”A study of the difficulties of novice programmers.” *Acm Sigcse Bulletin*, vol. 37, pp. 14-18, Jun. 2005.

[29] O. A. Patiño, S. Contreras-Ortiz and J. C. Martínez-Santos. “Evolution of Microcontroller’s Course under the Influence of Arduino,” in *Proc. 14th LACCEI Int. Multi-Conf. Eng., Edu., Technol.*, 2016, pp. 1-7.

[30] P. Torrone. “Why the Arduino won and why it’s here to stay.” *Make: Technology on your time.* Internet: https://makezine.com/2011/02/10/why-the-arduino-won-and-why-its-here-to-stay/, [February 26, 2019].

[31] J. M. Pearce. “Building research equipment with free, open-source hardware.” *Science*, vol 337, pp. 1303-1304, Sep. 2012.

[32] S. H. Kim and J. W. Jeon. "Introduction for freshmen to embedded systems using LEGO Mindstorms." *IEEE Transactions on Education*, vol. 52, pp. 99-108, Feb. 2009.

[33] P. Blikstein. "Digital fabrication and 'making'in education: The democratization of invention." *FabLabs: Of machines, makers and inventors*, vol. 4, pp. 1-21, 2013.

[34] E. R. Halverson and K. Sheridan. "The maker movement in education." *Harvard Educational Review*, vol. 84, pp. 495-504, Dec. 2014.

[35] D. Dougherty. "The maker movement." *Innovations: Technology, Governance, Globalization*, vol. 7, pp. 11-14, Jul. 2012.

[36] K. Peppler and S. Bender. "Maker movement spreads innovation one project at a time." *Phi Delta Kappan*, vol. 95, pp. 22-27, Nov. 2013.

[37] H. Lieberman, F. Paternò, M. Klann and V. Wulf. "End-user development: An emerging paradigm," in *End user development*, Springer, 2006, pp. 1-8.

[38] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrance, H. Lieberman, B. Myers and M. B. Rosson. "The state of the art in end-user software engineering." *ACM Computing Surveys (CSUR)*, vol. 43, pp. 21, Apr. 2011.

[39] J. Sadler, K. Durfee, L. Shluzas and P. Blikstein. "Bloctopus: a novice modular sensor system for playful prototyping," in *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction,* ACM, 2015, pp. 347-354.

[40] H. S. Raffle, A. J. Parkes and H. Ishii. "Topobo: a constructive assembly system with kinetic memory," in *Proceedings of the SIGCHI conference on Human factors in computing systems,* ACM, 2004, pp. 647-354.

[41] A. Robins, J. Rountree and N. Rountree. "Learning and teaching programming: A review and discussion." *Computer science education*, vol. 13, pp. 137-172, Jun. 2003.

[42] J. F. Pane and B. A. Myers. "Studying the language and structure in non-programmers' solutions to programming problems." *International Journal of Human-Computer Studies*, vol. 54, pp. 237-264, Feb. 2001.

[43] D. Alimisis, M. Moro, J. Arlegui, A. Pina, S. Frangou and K. Papanikolaou. "Robotics & constructivism in education: The TERECoP project." *EuroLogo*, vol. 40, pp. 19-24, Aug. 2007.

[44] B. S. Fagin, L. D. Merkle and T. W. Eggers. 'Teaching Computer Science with Robotics Using Ada/Mindstorms 2.0," in *Proceedings of the 2001 annual ACM SIGAda international conference on Ada*, 2001, pp. 73-78.

[45] B. Fagin. and L. Merkle. "Measuring the effectiveness of robots in teaching computer science." *ACM SIGCSE Bulletin*, vol. 35, pp. 307-311, Feb. 2003.

[46] F. Klassner. "A Case Study of LEGO Mindstorms TM Suitability for Artificial Intelligence and Robotics Courses at the College Level." *ACM SIGCSE Bulletin*, vol. 34, pp. 8-12, Feb. 2002.

[47] P. B. Lawhead. "A road map for teaching introductory programming using LEGO© mindstorms robots." *ACM SIGCSE Bulletin*, vol. 35, pp. 191-201, Jun. 2002.

[48] S. S. Botelho, L. G. Braz, and R. N. Rodrigues. "Exploring creativity and sociability with an accessible educational robotic kit," in *Proceedings of the 3rd International Conference on Robotics in Education (RiE 2012)*, 2012, pp. 55-60.

[49] K. Karvinen. "Choosing Novice Friendly Sensors." First Published September 18. https://doi.org/10.1177/0020720918800821

[50] J. Sarik and I. Kymissis. "Lab kits using the Arduino prototyping platform," in *2010 IEEE Frontiers in Education Conference (FIE)*, 2010, pp. T3C-1.

[51] P. Jamieson and J. Herdtner. "More missing the Boat—Arduino, Raspberry Pi, and small prototyping boards and engineering education needs them." *Frontiers in Education Conference (FIE)*, IEEE, 2015, pp. 1-6.

[52] E. Bilotta, L. Gabriele, R. Servidio and A. TAvernise. "Edutainment robotics as learning tool," in *Transactions on edutainment III*, Springer, 2009, pp. 25-35.

[53] F. Klassner. "A Case Study of LEGO Mindstorms TM Suitability for Artificial Intelligence and Robotics Courses at the College Level." *ACM SIGCSE Bulletin*, vol. 34, pp. 8-12, Feb. 2002.

[54] R. Balogh. "Educational robotic platform based on Arduino," in *Proceedings of the 1st international conference on Robotics in Education, RiE2010*, 2010, pp. 119-122.

[55] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. "Internet of things: Vision, applications and research challenges." *Ad hoc networks*, vol. 10, pp. 1497-1516, Sep. 2012.

[56] S. Madakam, R. Ramaswamy and S. Tripathi. "Internet of Things (IoT): A literature review." *Journal of Computer and Communications*, vol. 3, pp. 164-173, May 2015.

[57] A. M. Agogino, S. L. Beckman, C. Castaños, J. Kramer, C. Roschuni, and M. Yang. "Design Practitioners' Perspectives on Methods for Ideation and Prototyping." *International Journal of Engineering Education*, vol. 32, pp. 1428-1437, Jan. 2016.

[58] P. Hertzog and A. Swart. "Arduino—Enabling engineering students to obtain academic success in a design-based module," in *Global Engineering Education Conference (EDUCON),* IEEE, 2016, pp. 66-73.

[59] T. Sarkar and N. Das. "Exploring Web of Things with embedded devices." *International Journal of Advanced Networking and Applications*, vol. 7, pp. 2719-2723, Nov. 2015.

[60] P. Laplante and S. Ovaska. *Real Time System Design and Analysis*. USA: Wiley-IEEE Press, 2011, pp. 3-4.

[61] "Apache Cordova." Internet: https://cordova.apache.org/, [January 20, 2018].

[62] K. K. Patel, J. Patoliya and H. Patel. "Low cost home automation with ESP8266 and lightweight protocol MQTT." *Transactions on Engineering and Sciences*, vol. 3, pp. 14-19, Dec. 2015.

[63] B. Ur, M. Pak Yong Ho, S. Brawner, J. Lee, S. Mennicken, N. Picard, D. Schultze and M. L. Littman. "Trigger-action programming in the wild: An

analysis of 200,000 IFTTT recipes," *in Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, ACM*, 2016, pp. 3227-3231.

[64] S. Ovadia. "Automate the internet with "if this then that" (IFTTT)." *Behavioral & social sciences librarian*, vol. 33, pp. 208-211, Oct. 2014.

[65] T. J. J. Li, Y. Li, F. Chen and B. A. Myers. "Programming IoT Devices by Demonstration Using Mobile Apps," in *International Symposium on End User Development,* Springer, 2017, pp. 3-17.

[66] "Tasker for Android." Internet: http://tasker.dinglisch.net/, [February 25, 2018].

[67] "Automagic." Internet: http://automagic4android.com/en/, [February 25, 2018].

[68] M. Swartwout. "The First One Hundred CubeSats: A Statistical Look." *Journal of Small Satellites*, vol. 2, pp. 213-233, Dec. 2013.

[69] K. Woellert, P. Ehrenfreund, A. J. Ricco and H. Hertzweld. "Cubesats: Cost-effective science and technology platforms for emerging and developing nations." *Advances in Space Research*, vol. 47, pp. 663-684, Feb. 2011.

[70] L. J. Paxton. ""Faster, better, and cheaper" at NASA: Lessons learned in managing and accepting risk." *Acta Astronautica*, vol. 61, pp. 954–963, Nov. 2007.

[71] "Integrated Solar Angle Sensor E910.86." Internet: www.mouser.com/ds/2/594/910_86-224506.pdf, [January 20, 2018].

[72] D. Geeroms, S. Bertho, M. De Roeve, R. Lempens, M. Ordies and J. Prooth. "ARDUSAT, an Arduino-based cubesat providing students with the opportunity to create their own satellite experiment and collect real-world space data," *ESA Publications Division C/O ESTEC*, 2015.

[73] W. T. Holman, B. D. Sierawski, R. Reed, R. A. Weller, and A. L. Sternberg. "The small satellite (cubesat) program as a pedagogical framework for the undergraduate ee curriculum," *Age*, vol. 24, pp. 1, 2014.

[74] D. Geeroms, S. Bertho, M. De Roeve, R. Lempens, M. Ordies and J. Prooth. "ARDUSAT, an Arduino-based Cubesat providing students with the opportunity to create their own satellite experiment and collect real-world space data." ESA Publications Division C/O ESTEC, 2015

[75] K. Karvinen and T. Karvinen. "Satellite Sun Sensor Prototype Tutorial." Internet: http://botbook.com/satellite/, September 4, 2013 [January 20, 2018].

[76] S. Papadakis, M. Kalogiannakis, V. Orfanakis and N. Zaranis. "Novice programming environments. Scratch & app inventor: a first comparison," in *Proceedings of the 2014 Workshop on Interaction Design in Educational Environments, ACM*, 2014, pp. 1.

[77] Y. Xia. "Using Blockly to Create Simple Sensor & Actuator Based Applications on the SensibleThings Platform." Bachelor thesis, Mid Sweden University, Sweden, 2014.

Embedded system development platforms have made designing prototypes and devices possible outside the engineering domain. While interdisciplinary community and maker movement have strongly adopted breakout physical computing toolkits, such as Arduino, it is not only hobbyists utilizing them. Easy accessibility toolkits are used in various research and scientific projects and in engineering education. Using embedded system development platforms have become a widespread practice of both engineers and non-engineers alike, covering almost every field.

This dissertation explores which tools and processes would lower the threshold of designing embedded systems, enabling non-engineers and novice engineers to turn their innovations into working prototypes in such way that the workflow allows experimentational prototypes to evolve into deployable embedded systems.

Aalto-DD 68/2019

ISBN 978-952-60-8506-7 (printed)
ISBN 978-952-60-8507-4 (pdf)
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Electrical Engineering
Department of Electrical Engineering and Automation
www.aalto.fi