Using Open Source for Rapid Prototyping of Proprietary Systems

A thesis on the fusion of a rapid prototyping platform and a proprietary system

Andreas Drotth
Soded Alatia
Abstract

Having a successful innovation process is crucial for any competitive enterprise. Being able to rapidly prototype and validate new concepts can reduce the time and cost from the specification-phase to a final product. This thesis investigates how open source hardware and software can accelerate the idea validation of proprietary systems.

The development of a microcontroller shield and open-source-based software is described and how this artifact can take advantage of the open source community, thus making innovation more flexible and concept validation quicker. This thesis followed a five-stage development process, and is evaluated with two controlled experiments.

The resulting artifact was shown through these controlled experiments that it was able to improve the idea validation of proprietary systems. The result of the controlled experiments showed that a new concept could be validated faster.
Acknowledgements

We would like to thank everyone involved with the project for making this thesis possible, including the companies Verisure and Mikrodust. Special mention goes to Björn Strandmark at Mikrodust for all the support, feedback and expertise in the field of embedded systems. A big thanks to our supervisor Ulrik Eklund for the great support, feedback and discussions. Finally, we would like to express our gratitude to our close friends, in particular Theodor Lindquist and Alf Stenbrunn for the invaluable feedback, support and helping us with the formation of this thesis paper.
6 Discussion
6.1 Choice of Method ............................................. 22
6.2 Analysis of Result ............................................. 22
6.3 Comparison with Related Work ............................. 23

7 Conclusion ..................................................... 24
7.1 Findings ....................................................... 24
7.2 Contributions ................................................ 25
7.3 Future Work .................................................. 25

A List of Requirements ......................................... 28
B List of Implemented Library Functions (SHRPE.h) .... 30
C Library Example Usage - Native ............................ 32
D Library Example Usage - Stream ............................ 34
E Getting Started Guide ....................................... 36
F Transcript of Questions and Answers ...................... 38
Abbreviations and definitions

**Verisure:** The company Verisure Innovation AB.

**Mikrodist:** The company Mikrodist AB.

**Open Source Hardware:** Hardware and technology designed under the Open Design culture movement. Examples on open source manufacturers are the Arduino platform and Raspberry Pi platform.

**OSD:** Open Source Development.

**Library:** A set of resources and functions, used to develop software.

**Embedded System:** A computer system with dedicated or few functions within a larger system.

**Artifact:** A manufactured component with the corresponding software.

**Shield:** A printed circuit expansion board (PCB), usually mounted to a microcontroller via its pin headers.

**RF module:** A radio frequency communication module manufactured by Mikrodist.

**CRC:** Cyclic Redundancy Check. Error-detecting code that adds a check value to data. The length of the CRC comes in various forms but common is CRC16 and CRC32 (16 bit and 32 bit respectively).

**IoT:** Internet of Things. A term for how machines, devices, vehicles, household appliances, clothes and other objects are outfitted with sensors and processors, with the ability to connect to networks and share collected data. The data can be used for analysis and control of systems.

**Sniffer:** A program that captures the communication over the air (radio sniffer)

**Advantages & Benefits:** To quickly distinguish between the two terms: A car with a bigger tank has an advantage over another car (with the exact same specifications except a smaller tank). The benefit of this is that the car with the bigger tank can drive for a longer period.

**Drawback:** A disadvantage. Less favorable condition or circumstance.

**NDA:** Non Disclosure Agreement. It is a matter of confidentiality agreement between the parties involved.

**Infographic:** Graphic visual representations of information, data or knowledge intended to present information quickly and clearly.

**JSON:** A text format that is easy to read by humans, and easily interpreted by computers.

**API:** Application Programming Interface. Usually a linked library.

**SW:** Software

**HW:** Hardware
1 Introduction

According to an infographic made by Cisco [1], there are billions of embedded systems all over the world, and the number of interconnected devices is growing steadily. In the same infographic [1], it is shown that there is a shift in the market where more products are becoming connected to the Internet since the introduction of the Internet of Things-era. Rothwell [2] states that in an industrial-technological change, the manufacturing companies are forced to seek greater flexibility and efficiency in responding to market changes.

This chapter explains the background of this thesis, its research aim, and limitations.

1.1 Background

Rothwell [2] underlines the importance of an innovation process that is efficient in terms of time and cost and presents a historical overview of five different process models. The “Fugle innovation process model” made by Preez et al. [3] in 2006 is an extension of Rothwell’s work and presented as the seventh generation innovation process model (see section 3.2 for a detailed description). The same authors [3] show that one of the key components to an efficient innovation process is the concept feasibility stage. Additionally, the authors [3] argue that the “concept feasibility” stage is about investigating how feasible a concept is which can be investigated using methods including, but not limited to, prototyping. This thesis is based on the “fugle model” made by Du Preez et al. [3] and focuses on the concept feasibility stage.

Rapid prototyping has emerged to be a well-used method to create physical components quickly because it enables faster concept validation [4]. Furthermore, open source has been a success and well established to be used as a form of rapid prototyping method, which according to Snyder et al. [5] and the authors Weiss, Younker and Pearce et al. [6, 7, 8] enables the reuse of existing blocks of code.

Figure 1 shows a visualization of the background description above and is intended to clarify how the investigated methods in the two lower layers fit well in a broader context. The efficient innovation process is the problem domain area that Rothwell has highlighted. Moreover, the concept feasibility was mentioned by Du Preez et al. as one of the key components to an efficient innovation process. These two layers are the focus of this thesis. The bottom two layers are the investigated methods for finding a solution to the top two layers and is done by solving a specific problem at a company called Verisure (see section 1.1.1).

Both Du Preez et al. [3] and Merkel et al. [4] states that rapid prototyping is a way of validating ideas, and consequently a way of testing how feasible a concept is. To further narrow down the investigation of this thesis, use of open source was chosen as a method for rapid prototyping.
1.1.1 Verisure Innovation AB

Verisure is a smart-alarm security company with the focus on innovation and selling home alarm systems to consumers [9]. Verisure Innovation Centre is where new products and services are developed, and where new ideas are tested. Much like the thinking of Internet of Things, the alarm systems are connected to a bigger system to enable the ability to share information and data between products, which gives more possibilities for new product ideas. To achieve this connection, Verisure’s products utilize an integrated radio communication module (RF module) manufactured by Mikrodust AB, which is further explained in 1.1.2. There are proprietary components in Verisure’s products and is what makes Verisure’s situation interesting for investigation.

Verisure wants to improve on their innovation process since their demonstrations and evaluations of new product ideas are mostly conceptualized-based. This is due to the fact that hardware for a new product idea has to be constructed before any real evaluation can be applied.

Verisure requests an artifact based on an open source platform. The artifact should consist of a shield with the RF module from Mikrodust. A list of requests from Verisure is described in section 5.1.

1.1.2 Mikrodust AB

Mikrodust is a company which specializes in low energy radio communication and is a subcontractor to Verisure. They develop and manufacture microcontroller-based radio modules, one of which is the integrated module used within Verisures products. This RF module is well-established in Verisures ecosystem of products and the software of the RF module is proprietary.
1.2 Research Aim and Question

Rothwell [2] states that having an efficient innovation process is preferable to an organization to meet the market changes. The authors Du Preez et al [3] describes in their "fugle" model that concept feasibility stage is one of the key stages for a successful innovation process. This stage is all about "fail smart and fast" and getting learning experience fast. Provided that the remaining stages of the "fugle-model" are fulfilled, it would be cost-effective to have an efficient prototyping phase [3].

The aim of this thesis is to examine how an open source hardware platform can accelerate the idea validation of proprietary systems. The research question below was defined to achieve the aim of this thesis.

- How can the use of open source hardware accelerate the idea validation of proprietary systems?

1.3 Limitations

Only one method for rapid prototyping (use of open source) was investigated for validating ideas. Specifically, only the usage of one open source platform was investigated.

An artifact consisting of a shield for a development board has been manufactured and was limited to an Arduino UNO. According to Arduino’s website [10], the Arduino UNO is the most common development board and the amount of information/support that exists simplified the development of the software. Although the shield should, in theory, be compatible with other development boards of the Arduino family with the same architecture, the compatibility has not been tested and thus, it is not guaranteed to work properly.

1.3.1 Licensing Consideration

Licensing properties are taken into consideration because the Arduino platform is open source. On Arduino’s homepage the following is stated about the LGPL-licence, which is a free software license published by the free software foundation [11]: "Using the Arduino core and libraries for the firmware of a commercial product does not require you to release the source code for the firmware. The LGPL-license does, however, require you to make available object files that allow for the relinking of the firmware against updated versions of the Arduino core and libraries. Any modifications to the core and libraries must be released under the LGPL-license" [12].

The code on the RF module was developed entirely using proprietary code and shall not be publicly available. However the electronic schematics of the shield as well as the developed library for the Arduino UNO development board has been released as open source.
2 Theoretical Background

In this chapter, the four layers of Figure 1 are presented, i.e. efficient innovation process, concept feasibility, rapid prototyping and open source.

2.1 Efficient Innovation Process

Du Preez et al. [3] describes the importance of innovation in the following way: "Innovation is widely recognized by industry and academics as an essential competitive enabler for any enterprise that wants to remain competitive and survive and grow".

Moreover the research of Du Preez et al. shows that although most enterprises fully realize the importance of innovation and continue spending more on it, it is rather the management process that is the key to success (i.e gaining profit from the innovation).

In 2006, Du Preez et al. [3] presented the seventh-generation innovation process model which is an extension of the previous models that were originally presented by Rothwell [2]. It is called the "Fugle model" and focuses on open innovation and network models.

Below is a description of the seven models described by both Rothwell [2] and Du Preez et al. [3] and Table 1 shows a visual overview of the models:

<table>
<thead>
<tr>
<th>Generation</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st (1930's)</td>
<td>Technology Push.</td>
</tr>
<tr>
<td>2nd (1960's)</td>
<td>Market Pull.</td>
</tr>
<tr>
<td>3rd (1970's)</td>
<td>Coupling Model.</td>
</tr>
<tr>
<td>4th (1980's)</td>
<td>Interactive Model.</td>
</tr>
<tr>
<td>5th (1990's)</td>
<td>Network Model.</td>
</tr>
<tr>
<td>7th (now)</td>
<td>Extended Innovation Network.</td>
</tr>
</tbody>
</table>

Table 1: Overview of the different models

The **first and second generation models** were presented in the 1930’s and 1960’s and are explained as either being driven by the needs of the market, or by advances in technology and science.

The **third generation model**, which was presented in the 1970’s, is a unified (coupling) model that recognizes both the marked needs and technology capabilities within the frame of the model. Although it is not linear since feedback loops exist and is operated iteratively, the workflow is essentially sequential. However, the third generation model does not address refinements and optimizations after the deployment stage and focuses mostly on the concept phase of the product innovation.

To improve upon the linear models, the **fourth generation model** was presented in the 1980’s where an interactive approach was developed. The innovation process was considered as parallel activities across organizational departments. This model has led to an increase in both horizontal strategic alliances within the organization’s departments as
well as strategic vertical relationships between other organizations. The development of cross-functional integration within firms to gain greater potential from real-time information processing is emphasized in the model. A limitation of this interactive model is that it stops prior to the implementation stage. Similarly to the third generation model, continuous improvement after the launch of the new product should be part of the innovation process, which is not supported in this model.

A new **fifth generation model** was presented in 1990’s to define the complexity of the innovation process. This model is called the **network model** and is characterized by the influence and effective communication of the external environment. It focuses on establishing a link between the key role-players. The innovation has its center in the organization. The process is affected by internal factors (i.e. organizational structure, cooperate strategies) and external factors (i.e. national infrastructure and regulations). This model is mostly a closed network model where most of the new ideas are generated from within the organization and in secrecy.

During the beginning of the 2000’s a **sixth generation model** was presented and is called the **open innovation model**. This model expands on the fifth model and focuses not only on the internal idea generation and development, but effectively uses external ideas and paths to market. The most obvious benefit of using an open innovation model is that the amount of idea generation exceeds the fifth generation model in which the idea generation was focused internally (i.e. within the organization). This new sixth generation model also enables companies to find new strategic tools and growth by being more open and collaborative.

The **seventh generation model** is already identified even though the sixth generation model is relatively new. It is called the "Fugle model" and can be described as an **extended innovation network**. It combines the network and open models to form an integrated innovation network. According to Du Preez et al. [3] the Fugle model "tries to integrate various best practice activities and concepts identified in the literature and therefore presents a consolidated view of the innovation process". Furthermore, the model is described as "centered on a generic innovation process that combines the convergent innovation front-end or funnel (identification and evaluation) with the divergent deployment and exploitation of the innovation" [3].

### 2.2 Concept Feasibility

Concept feasibility is, according to the Fugle model, only one of multiple stages in an innovation process. This stage follows naturally after having generated ideas and refined them into workable concepts. The concept feasibility stage is all about collecting further information and examining its feasibility by prototyping and modeling. The idea is to "fail fast and smart" and learning by doing. It is better and more cost-effective to iterate through this stage multiple times rather than during the deployment stage [3].

Concept Feasibility is one of the key steps in the innovation process and can be the difference between having a proof-of-concept idea, rather than just a conceptual framework. The result of this stage decides which concept the company will finance [3].
2.3 Rapid Prototyping

Buchenreider [13] describes Rapid prototyping as an "engineering action in which a technical system is derived from the specification and operand in a real-life environment for evaluation, testing, refinement, and operation". It is understood to be a method used for idea validation and thus a way of testing how feasible a concept is (see figure 1). This method has often been used to create physical components quickly and has since evolved to be used in other areas such as software development. There are many forms of rapid prototyping software including, simulation (example of simulation: [14]), code generation (example of code generation [15]) and re-use blocks of code (example of reusability [16]). Lately, rapid prototyping has been used in hardware as well, i.e. re-use the design of integrated circuits and other electronic components [7].

2.4 Open Source

The opensource.com website describes that "Open source projects, products, or initiatives are those that embrace and celebrate open exchange, collaborative participation, rapid prototyping, transparency, meritocracy, and community development" [17].

Although open source is widely recognized as "something that can be modified because its design is publicly accessible" [17], research shows it has an imprecise meaning [18]. Gacek et al. [18] found two common denominators (the projects adherence to open source development and that developers are also users) in 80 different open source projects of different sizes, but they concluded that there were multiple other varying characteristics to describe an open source project.

Open source originates from the software development but has since evolved to be included in, but not limited to, hardware. Two well known Open Source Hardware platforms are Arduino [12] and Raspberry Pi [19].
Related Work

In this chapter, papers with similar work or results are presented.

The main tool used for searching research papers and journals was a search tool called "Summon", which is accessible for students of Malmo University. Summon is similar to Google Scholar in which the search tools enable the user to search through multiple databases simultaneously. However, the main difference between the two searching services is that Summon allows filtering the result to only allow 'peer-reviewed articles'.

Six databases with either high reputation or large amounts of content in the field of computer science were chosen. These were: SPRINGER, ACM Digital Lib, Science Direct, IEEE, Web of Science and Scopus. Only English-written and peer-reviewed articles were selected. Moreover, only papers with full-text access were chosen.

Two research papers were selected using Google scholar by searching for the title of the research papers in quotation marks.

Two compilations were created to document all the findings of the research papers. The first included data of the research papers such as key search terms, title, date of search, database and a link to the article. The other included an overview of the different combinations of key search terms in the six databases mentioned above and the amount of results for each of these combinations. To filter the most related research papers, only papers from the search results fewer than 20 were selected. Section 3.1 - 3.8 presents the research papers that were chosen.

<table>
<thead>
<tr>
<th>Keyword Combination</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Open source hardware&quot;</td>
<td>183</td>
</tr>
<tr>
<td>&quot;proprietary hardware&quot;</td>
<td>483</td>
</tr>
<tr>
<td>&quot;Open source hardware&quot; + &quot;proprietary hardware&quot;</td>
<td>5</td>
</tr>
<tr>
<td>&quot;open source hardware&quot; + &quot;benefits&quot;</td>
<td>77</td>
</tr>
<tr>
<td>&quot;open source hardware ecosystem&quot;</td>
<td>0</td>
</tr>
<tr>
<td>&quot;open source hardware&quot; + &quot;ecosystem&quot;</td>
<td>14</td>
</tr>
<tr>
<td>&quot;open source hardware&quot; + &quot;proprietary&quot;</td>
<td>52</td>
</tr>
<tr>
<td>&quot;proprietary hardware&quot; + &quot;open source&quot;</td>
<td>75</td>
</tr>
<tr>
<td>&quot;innovation process&quot; + &quot;rapid prototyping&quot;</td>
<td>218</td>
</tr>
<tr>
<td>&quot;innovation process&quot; + &quot;rapid prototyping&quot; + &quot;open source&quot;</td>
<td>16</td>
</tr>
<tr>
<td>&quot;innovation process&quot; + &quot;rapid prototyping&quot; + &quot;open source hardware&quot;</td>
<td>2</td>
</tr>
<tr>
<td>&quot;open source&quot; + &quot;closed source&quot;</td>
<td>746</td>
</tr>
<tr>
<td>&quot;open source hardware&quot; + &quot;closed source hardware&quot;</td>
<td>0</td>
</tr>
<tr>
<td>&quot;open source hardware&quot; + &quot;closed source&quot;</td>
<td>10</td>
</tr>
<tr>
<td>&quot;open source&quot; + &quot;closed source hardware&quot;</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2: Overview of search results against keyword combinations
3.1 Towards the Fifth-Generation Innovation Process

Rothwell [2] presented in 1994 a literature review of the evolution of innovation process model and, at that time it was the fifth-generation which was the latest model. Rothwell stated that companies are leading towards the adoption of "sophisticated electronic toolkit in their design and development activities to enhance developmental flexibility". The investigation of the rapid prototyping method in this thesis will contribute to the statement of Rothwell.

3.2 An Innovation Process Model for Improving Innovation Capability

Du Preez et al. [3] present a literature review of the evolution of the innovation process model. The paper is based on Rothwell’s contributions and additional literature that was identified after Rothwell’s paper was published. An integrated process model of the latest developments is consolidated, which is presented as the "Fugle model" which is the seventh-generation innovation process model (see figure 2). Unlike most innovation process models that focus mostly on the identification and filtering of new ideas, the Fugle model also takes the implementation, commercialization and exploitation into consideration. It also utilizes the best practice activities and concepts from the literature. The authors underline that an organization must be strong in every phase of the Fugle model in order to succeed from idea to product.

The fugle model (Figure 2) has been simplified and only include what is relevant to this thesis. The original model is more complex and contains the external network as well.

![Image](image.png)

Figure 2: *The Fugle model, with flow from left to right. The marked stage is where the focus of this thesis lays.*

3.3 Rapid prototyping of embedded HW/SW systems

Buchenrieder [13] describes the importance of prototyping embedded systems. The paper provides arguments to why having a working prototyping phase may very well drastically reduce the time from specification to a final product. The author also explains how prototyping can assist in all the steps in the engineering process and identifies key areas for research. Buchenreider’s paper contributes to the theory of the problem domain area.
3.4 Accelerating the innovation process with the help of OS prototyping HW

The paper was written by Merkel et al. in 2012 [4] provides good knowledge about how open source hardware is used mainly in scientific projects. The paper also investigates how such open source hardware can benefit the prototyping stage within companies as well, by using already existing hardware as complementary equipment during the innovation process. Furthermore, the paper investigates and compares several different open-source manufacturers, both software, and hardware, showing the benefits versus the limitations of the different hardware.

3.5 The Aspects of Choosing Open Source Versus Closed Source

The paper of Khanjani et al. [20] focuses on the open-source development in general, versus the more traditional closed source development that is more common within a company’s development cycle. The paper discusses the many different benefits that open source can bring, but also provides information regarding the problems of using open source for development.

3.6 Beyond Open Source Software: Solving Common Library Problems Using the Open Source Hardware Arduino Platform

Younker et al. [7] collects information about Arduino and gives a walk-through on what Arduino actually is. The paper provides plenty of knowledge about the benefits of the brand and the ecosystem, but does also look into some of the disadvantages and why Arduino is useful in some areas but not others. The collection of references is also useful. The authors implemented two technical solutions to validate their statement of quickly and inexpensively prototyping custom hardware solutions for common library problems. The artifacts were manufactured using only an Arduino development board and various sensors from the open source hardware environment. The authors’ way of working on these solutions provides insight on the usage of the Arduino open source hardware platform.

3.7 An Open Source(d) Controller

Sarkinen [21] provides useful information about open source hardware and software in general, but the paper focuses on how the usage of open source affects a business model. The usage of open hardware and software can both bring benefits and disadvantages, even liabilities that may come under certain licenses which are common amongst open source HW and SW. The paper examines not only the availability of support and community around different known open source manufacturers but also a thorough investigation of the legal aspects. Examples of consequences are provided with solutions on how to fully use the potential while following legal obligations.
3.8 Agility Counts in Developing Small-Size Software

The paper of Wang and Lingfeng in 2007 [22] focuses entirely on agile development. During the implementation phase, there is a need for effective working and developing, and agile development is a good practice for better results. The paper compares several popular different agile methodologies to a conclusion describing which one is best for what scenario and circumstance. Furthermore, the paper examines why agile development is working well together with open source software development.
4 Method

The prototype along with the corresponding software was developed using systems development as a research methodology. This way of working is proposed by Nunamaker and Chen and uses a five-stage development process [23]. These five stages describe a systematic approach to constructing and evaluating a system. Nunamaker and Chef emphasize that the stages should be done in order but if necessary one could revise earlier stages and change previous design choices, see figure 3.

Section 4.1 - 4.5 describes the different stages in figure 3 and presents the work that was done in this thesis in the respective stages.

4.1 Construction of a Conceptual Framework

This stage is about stating an interesting and meaningful research question, investigating the system requirements and studying previous work that has been done in the same problem domain.

The broad research area was narrowed down to a specific problem domain which is illustrated in figure 1 in section 1.1. From the beginning, a decision was made to build an artifact (prototyping kit) for Verisure because of their need of an improved innovation process (see section 5.1). A literature study was performed, focusing on the benefits of open source, how other similar work within the domain of research has been done, as well as how an efficient innovation process is achieved. During this stage, meetings were held
biweekly with representatives from Verisure and Mikrodust. The senior product specialist at Verisure informed about the intentions of improving the innovation process, and some of the goals the project should achieve. A complete list of requirement specification could be obtained from these biweekly meetings. See appendix A which could be derived from Verisure’s requests listed in section 5.1. The result of this stage led to the information provided in section 1 - section 3.

4.2 Develop a Systems Architecture

The second stage is about developing a system architecture, identifying the tasks and defining the sub-tasks of the solution proposal as well as defining the interrelationships among them.

During this stage, meetings were held weekly with only representatives from Mikrodust to finalize the system architecture. An existing reference application used in Mikrodust’s operating system was given to study the existing code that could, and should, be re-used to program the module. A communication protocol (UART) was decided for the communication between the module and the Arduino. Figure 4 is the result of this stage.

![Figure 4: Overview of the solution approach broken down into subtasks.](image)

4.3 Analyze and Design the System

The third stage is about developing alternative solutions and choosing one solution. At this stage, the functions of the system should have clear definitions. Ultimately, the chosen design should be used as a blueprint for how to build the system in the next stage.

Since Verisure requested a working prototype as fast as possible, no alternative solutions were developed. However, during the weekly meetings with Mikrodust, the analysis of the design was performed simultaneously while developing the system architecture. The system architecture and the design choice were developed and analyzed iteratively.

An analysis was needed to determine what existing code could, and should, be re-used. Mikrodust has developed an operating system that is being run on the RF module that handles basic functionality such as inputs and outputs. Additionally, the operating system is capable of executing user-defined applications that have been developed for this
operating system. Moreover, Mikrodust has in co-operation with Verisure developed radio communication API’s that connects to and transmits data to the Verisure gateway.

The result of this work is presented in section 5.1.

4.4 Build The System

This stage is about using the information obtained from the previous stages in order to build the system.

During the fourth stage, the actual artifact was built. The hardware was built by Mikrodust and a second revision was manufactured after feedback from the developers. However, the software was developed by the authors of this thesis. The software application of the RF module was implemented upon the underlying operating system and radio communication API.

While developing the software, an agile way of working was used, using elements of a Scrum approach such as having a prioritized backlog and demos with the scrum master. This approach was decided together with Mikrodust, with them being the scrum master and Verisure the product owner. The resulting artifact is further described in section 5.2.

4.5 Observe and Evaluate the System

The fifth and final stage is about evaluating the system using the requirements which were decided in the first stage. Then, the system should be observed in the ‘real world environment’ using, for example, case studies. The research question should then be answered using the information gained from the observation of the system.

During the last stage, the artifact was evaluated in two ways. The artifact was first evaluated against the requirements (see section 5.3.1) to ensure the artifact fulfills Verisure’s goals by creating two example Arduino applications (see Appendix C and D). A proprietary software from Mikrodust (radio sniffer) was used to capture radio communication to confirm that the data sent and received was correct. The radio sniffer enabled testing of the developed code. The artifact was then evaluated by observation in a business environment, via two controlled experiments. The developers from Verisure participated in these controlled experiments (see the sections 5.3.2 and 5.3.3 for detailed descriptions).
5 Results

The existing prototyping process at Verisure was analyzed. At Verisure, there are existing products, such as domestic house alarm systems, smoke detectors, etc. A majority of these products are meant to be connected to gateways as various entities, forming a network of entities that communicate with each other. This is achieved by utilizing the integrated radio frequency module (RF-module) from Mikrodust to communicate with the gateways and having all entities communicate using the same radio protocol.

Verisure needed a solution that improved their innovation process to test the feasibility of new concepts and ideas. It was complicated to verify or evaluate a new idea for a product, due to the hardware of the products had to be constructed and assembled before any software could be tested. A prototyping environment or a prototyping kit was requested to simplify the experimentation of new software for product ideas, without having to construct new hardware.

5.1 System Overview

Below are the requests from Verisure:

- The artifact must act within the Verisure network similar to existing products. This means that the artifact must not interfere with other products within the same network. This request allows new product-ideas and concepts to be developed and tested in a correct environment (i.e. within a network alongside existing products). Therefore, the artifact must include the RF module from Mikrodust, to communicate using the same protocol as the other entities.

- New software should be possible to test without constructing new hardware. The artifact must be capable of varying tasks, which allows the developer to test ideas with different purposes. Therefore, the artifact must be developed to be as generic and flexible as possible.

- Verisure has requested the use of popular open source hardware, to gain the access to third-party constructed hardware. Verisure’s suggestion is using the Arduino family of microcontrollers.

- Verisure also requested that the artifact should be based around a shield, a circuit expansion board, that would be mounted on the Arduino microcontroller. This shield should include the RF module from Mikrodust.

As seen below in figure 5, the artifact should act as a stand-alone entity within a Verisure network, working alongside existing products. The complete list of requirements in appendix A are based on the aforementioned requests.
5.2 Artifact

The constructed artifact is a prototyping kit, consisting of an Arduino shield and an Arduino library. The Arduino library consists of an API of functions for the Arduino IDE and is the main software for the developer to use. The Arduino shield is manufactured to fit an Arduino UNO development board. Both the hardware and the software part of the prototyping kit are described below, and are illustrated in figure 6 and 7 respectively.

Figure 6: Overview of the artifacts hardware parts. The RF module is integrated onto the shield, and the shield is used by mounting it into the Arduino development board.
As seen in figure 7, the software on the RF module consists of two parts, the underlying OS and the developed application. On the right side, there is the Arduino application run by the Arduino UNO. The developers write their own code, that utilizes the developed library for Arduino and, if desired, other libraries that are wanted for their application.

5.2.1 Microcontroller Shield

The hardware is in the form of a shield, constructed to be mounted on a development board, specifically the Arduino UNO. Choosing this particular development board was not only a part of the requests by Verisure (see appendix A) but also due to this particular board’s popularity [4]. The Arduino UNO development board offers a wide range of compatible third-party hardware and software that can be used for prototyping.

Figure 7: Overview of the artifacts software parts, showing the relation between the different software parts.

Figure 8: An overview of the manufactured shield.
The shield is an expansion circuit board, constructed around Mikrodust’s RF module as the main part (see the RF module encapsulated on the shield in figure 8). Other integrated parts of the shield are:

- LEDs. Three, each in a different color (red, green and blue).
- Antenna connector. Needed for radio transmission.
- RESET button. Although the Arduino board has a reset button of its own, it will be mostly covered by the shield and less accessible. Therefore, an additional, more reachable, reset button on the shield board has been added.
- UART connector and jumper (front). Communication between the shield and the Arduino CPU is via UART. UART is also used when flashing the Arduino board. The jumper enables the developer to momentarily break the connection between the Arduino’s UART and the shield. This is only done when uploading new code the Arduino board.
- UART connector (rear). Having an extra connector on the same UART line gives the developer an ability to "listen" on the communication. This extra connector is used for debugging purposes.
- DEBUG interface. Used to both program and directly communicate with the module. The interface is a JTAG-connector.

Third-party hardware compatible for Arduino can be directly connected to the shield since all pin headers located on the Arduino board are accessible via the shield.

### 5.2.2 Shield Module Software

The application that is being run on the RF module was developed with the help of Mikrodust and is proprietary (the code is solely owned by Mikrodust). Since a non-disclosure agreement has been signed by us and Mikrodust, only a brief explanation of what the code does is given below, and no proprietary parts can be described.

The code was developed by re-using large blocks of code from an existing reference program developed by Mikrodust. It was then modified to work as a master-slave protocol where the Arduino microcontroller is the master and the shield is the slave. The shield listens for requests from the Arduino microcontroller and transmits the request to the Verisure gateway using radio communication. The shield also listens for requests from the gateway and transmits the request to Arduino.
5.2.3 Arduino External Library

The API available to the developer is an external library designed for the Arduino IDE [24]. This library follows the design standard and guidelines set by Arduino [25, 26, 27]. The external library is released under the LGPL [12] license, making it open source and freely editable by anyone. This is an integral part of the solution.

The library provides the means to operate and control the shield. A full list of functions, including detailed explanation, is described in appendix B.

Several of these functions follow a standard that is present within the Arduino community, to further ease the ability to use other libraries on top or together with this library. This enables the developer to reuse existing blocks of code from other Arduino-sketches without the need to worry about the API being different from other libraries API’s.

- Functions like `begin()` and `loop()` are the main parts of almost any code designed for the Arduino IDE, and this library as well. These functions provide and run the necessary code to start up a library.

- Data communication functions are based on the standard Arduino Stream interface [28], namely `available()`, `read()`, `peek()`, `flush()`, `write()`. These functions, both names and purpose, resembles that of any Stream interface and therefore provide more ease for the developer if switching to another library used for communication. Having a standard of function names for a different kind of communication leads to the fact that less code has to be modified when switching libraries. Ethernet [29], GSM [30] and WiFi [31] are examples of the standard libraries for the Arduino IDE regarding communication.

The communication between the shield module and the Arduino is, as mentioned in section 5.2.1, done via UART. A protocol has been implemented that controls the flow of how data is being sent and received. The Arduino acts as a master and the shield acts as a slave. UART communication does not typically have this structure, but this architecture was chosen in case I2C is implemented in the future.

The data packets sent and received are framed to ensure that the data is not corrupted during transmission. The framing of the packets is done with a third-party external library (see [32]) that is open source and is re-used as part of our Arduino library. The framing library takes advantage of the standard library for the UART communication from the Arduino IDE, Serial [33]. The actual communication uses the default configuration for the Arduino UNO, which is sending a byte at the time. The framing library provides these key components:

- One start byte and one stop byte.
- CRC16 (Cyclic redundancy check of 16 bits).
- Byte stuffing on all data bytes, including the CRC checkbytes.

In order to prevent corrupt data packages being sent between the Arduino and the shield, the third-party open source library [32] was used to check for errors in the data.
5.2.4 Getting Started Guide

A getting started guide has been written for the developers that want to use the manufactured artifact, which can be seen in Appendix E.

![Figure 9: The shield is mounted on an Arduino UNO. The cables seen are used for debugging purposes. The blue LED is indicating power.](image)

5.3 Verification and Validation

The table in 5.3.1 shows how the requirements in Appendix A are met. The artifact is then also validated with two controlled experiments as described in section 5.3.2 and 5.3.3.

The first controlled experiment involved a team of 20 developers within Verisure. This was done halfway through the project with the aim to collect feedback from the developers in order to refine and add more functionality. The second controlled experiment involved only one senior developer at the end of the project.

5.3.1 Verification of Requirements

Based on the requests from Verisure in section 5.1, a complete list of requirements was created and can be found in appendix A. In this section, the results are shown whether these requirements were fulfilled or not. A proprietary software from Mikrodust (radio sniffer) was used to capture radio communication to confirm that the payload was correct. The sniffer enabled testing of the developed code.
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Fulfilled</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The solution should be a shield</td>
<td>Yes</td>
<td>The solution is a shield with corresponding software as requested by Verisure.</td>
</tr>
<tr>
<td>2. Benefit of an open source hardware ecosystem.</td>
<td>Yes</td>
<td>The solution is able to benefit from the Arduino open source hardware ecosystem. Developers using the shield has full flexibility to make use of various third-party hardware.</td>
</tr>
<tr>
<td>3. Build upon proprietary hardware.</td>
<td>Yes</td>
<td>The shield is constructed around a proprietary module.</td>
</tr>
<tr>
<td>4. Compatible with Arduino Uno.</td>
<td>Yes</td>
<td>The main target development board is the Arduino UNO. Additionally, the shield is compatible with an Arduino MEGA.</td>
</tr>
<tr>
<td>5. Programmable with the Arduino IDE.</td>
<td>Yes</td>
<td>The target development environment is the Arduino IDE and the developed library is designed for that particular software. A guide for how to get started with the artifact is provided.</td>
</tr>
<tr>
<td>6. Should consist of Mikrodusts RF Module.</td>
<td>Yes</td>
<td>As mentioned in requirement 3, the shield is built around the RF module, making the shield an extension to the module.</td>
</tr>
<tr>
<td>7. Should build upon the existing underlying radio module OS.</td>
<td>Yes</td>
<td>The underlying OS is the foundation for the new firmware on the module. The firmware takes advantage of the underlying OS to perform RF communication tasks.</td>
</tr>
<tr>
<td>8. Make use of the existing radio communication API.</td>
<td>Yes</td>
<td>Just as with the underlying OS, the firmware takes advantage of the radio communication API.</td>
</tr>
<tr>
<td>9. Developers should be able to test ideas with limited Verisure specific knowledge.</td>
<td>Yes</td>
<td>The design of the library makes it possible to use without specific knowledge about how Verisure’s system works. The functions follow a standard interface commonly seen within libraries in the Arduino community, which allows for a quicker understanding of how to use this library.</td>
</tr>
<tr>
<td>10. Data sent and received should have a generic form/data structure.</td>
<td>Yes</td>
<td>The library contains two functions <code>uploadObject()</code> and <code>downloadObject()</code>. With these functions, a developer can send and/or receive 1-40 bytes of data. The data is framed with the same headers as any other type of message sent within the network.</td>
</tr>
<tr>
<td>11-18. Arduino library should implement fundamental Verisure functions.</td>
<td>Yes</td>
<td>Verisure requested the most fundamental functions. All of the functions have successfully been implemented, which can be seen in the two example apps in appendix C and appendix D.</td>
</tr>
</tbody>
</table>
5.3.2 Validation: Developer Hackathon

The first controlled experiment was made during a two-day hackathon of which five hours were spent using the artifact. The total amount of developers at the hackathon was 20 but the amount of developers who worked with the artifact was only 10. The other developers were working on a solution to collect the data being uploaded from the Arduino shield to Verisure’s back-end system. Since only five shields were manufactured at this point of time, the ten developers were divided into five groups. Each group was instructed to come up with new ideas and then use the artifact to validate those ideas. It should be noted that the development of the artifact was still in progress and the only function that was supported at this point of time was the `uploadObject()` function. There were two objectives of the first controlled experiment:

- Collect feedback from the developers to refine the artifact and add new functionality.
- Get a statement from the developers what their impression of the artifact was and if it would help them trying out new ideas faster than before.

The feedback was received in the form of emails on two different occasions. The first email was received at the end of the hackathon from the hackathon coordinator, highlighting existing bugs and expressing gratitude for the creation of the artifact. The email also contained suggestions for new functionality.

The second email clarified that the coordinator of the hackathon had gathered the feedback from the rest of the hackathon developers, and the answers are the coordinator’s impression based on the collected feedback. The questions and answers were written in Swedish. An English translation of the transcript can be found in Appendix F.

5.3.3 Validation: Senior Developer Evaluation

The second controlled experiment was done by one senior developer. The artifact had been refined and new functionality had been added. The project leader then created the classic "hello world!"-program using the manufactured shield and the corresponding library API. This was a demonstration that data could be sent and received from the back-end system of Verisure.

Another two examples were written by the senior developer to be demonstrated in an open invitation to all the developers within Verisure. The internal presentation at Verisure occurred on September 3rd, 2015. The first of these two examples was called a "doorbell example". This example consisted of a button that was configured as an input in the software written in the Arduino IDE. When the button was pressed, a doorbell chime sound would go off using an existing Verisure product that was placed somewhere else in the conference room. The second example was called a "generic button". In this example, whenever a button was pressed, a formatted string in JSON would be uploaded to the Verisure back-end system. It would then turn on and off a lamp via a smart-plug by Verisure. These two examples showed the capabilities of the artifact and were written in just one day by the senior developer.

One day after the presentation, the project leader sent feedback in an email stating that the overall presentation, including the result and demonstration, had been highly appreciated. Many in the audience thought "the proposed solution was great" and the project leader is planning on including the artifact in Verisure’s production environment.
6 Discussion

6.1 Choice of Method

The artifact was developed using 'systems development' as a research methodology. This way of working is proposed by Nunamaker and Chen and uses a five-stage development process [23]. These five stages describe a systematical approach to constructing and evaluating a system and is according to Nunamaker and Chen an appropriate way of working when creating an artifact. An alternative method is the ‘design-science’ in information systems research which is a seven-stage guideline and according to [34], an effective way of designing a technical artifact. There are similarities between the methods such as the illumination of creating a purposeful artifact for a specific problem domain and highlighting the importance of evaluating the artifact. The two main reasons to why the 'systems development' method by Nunamaker and Chen [23] was chosen are that the method

- gives more detailed steps in creating the artifact such as developing an architecture and analyzing a system design.
- enables the possibility to revise different stages iteratively if necessary.

6.2 Analysis of Result

The aim of this thesis was to investigate how open source hardware can accelerate the idea validation of proprietary systems. The result of this thesis work is an artifact developed for a company called Verisure. The artifact consists of proprietary components attached to an Arduino shield and the corresponding software consists of both proprietary code and open source software. The artifact acts as an entity within the Verisure network working alongside existing products.

The artifact is a working solution to Verisure’s problem, and it shows advantages to help developers at Verisure try out new product concepts and validate the ideas. Through two controlled experiments, we have observed the use of the artifact and how easy it is to experiment with and test how feasible an idea is. By taking advantage of the Arduino open source platform, the developers can use third-party hardware for Arduino and reuse open source software. This has not been possible previously as new hardware had to be constructed for each idea, which is costly in terms of time and money. The goal was to bridge between Verisure’s proprietary, but necessary hardware, and available open source hardware.

Other companies with the same problem as Verisure will not be able to apply the artifact to their working environment and systems because it is deeply integrated into Verisure’s network. This is, of course, a drawback with our solution. We believe it is difficult to implement a generic solution that works for multiple companies since the aim of the artifact is for 'proprietary use'. However, we argue that the same approach (see section 4) to develop the artifact could be applicable for other companies with the same aim.

All of the software that has been developed for the Arduino platform has been released as open source. Since the artifact is intended for internal use only this is not going to be a problem. If Verisure plans on selling the artifact for profit, a thorough investigation of what licensing the artifact should be released under is needed.
6.3 Comparison with Related Work

In section 3, some of the related work is presented. The papers 'Towards the Fifth-Generation Innovation Process' and 'An Innovation Process Model for Improving Innovation Capability' by Rothwell [2] and Du Preez et al. [3] describe the underlying stages for having a successful innovation process. Both papers identify the 'idea validation' stage as one of the key stages for an effective innovation process. The paper 'Rapid Prototyping of Embedded HW/SW Systems' by Buchenrieder [13] also highlights the importance of how rapid prototyping of embedded systems can drastically reduce the time from specification to a final product. By evaluating the artifact through two controlled experiments, we have shown how the use of open source hardware and software as a method for rapid prototyping can decrease the idea validation to just one day.

The paper 'Accelerating the innovation process with the help of OS prototyping HW' by Merkel et al. [4], is the most similar work to our thesis and thus the most interesting paper to compare with. The paper investigates the same research area as we do. They describe on how open source hardware can benefit an entire innovation process while we only draw conclusions based on how open source hardware and software is beneficial when testing an idea for its feasibility. We have provided an artifact of both proprietary hardware and open source hardware/software as a combined solution, whereas the paper of Merkel et al. only presents the existing open source hardware platform as a way of prototyping.

Our work is an extension of the work of Merkel et al. [4] since we extend the rapid prototyping capabilities to be included when applied to proprietary systems by taking advantage of open source.
7 Conclusion

By using open source it is possible to rapidly and inexpensively create prototypes (see section 3.5 and 3.6). This way of validating ideas has increased in popularity and is described by Merkel et al. [4]. Idea validation is, according to Rothwell [2], one of the key stages in an efficient innovation process. This thesis investigated the 'rapid prototyping' method for validating ideas.

To further narrow down the specific problem that was investigated, the focus was on how companies that use, develop and sell proprietary systems, can utilize open source hardware platforms as a method for validating ideas and thus finding out if a concept is feasible. The "Fugle model" is presented in section 3.2 to provide a context for the investigation in this thesis.

The manufactured artifact is only intended to be used by the company Verisure since the artifact interacts with the company’s proprietary systems. However, other companies that wish to accelerate the idea validation of their proprietary systems could use the same approach as described in section 4.

This thesis is based on the 'Fugle model' (see Figure 2), which is made by Du Preez et al. [3], by investigating the rapid prototyping method in the 'concept feasibility' stage.

7.1 Findings

An artifact was developed for Verisure that combines an open source hardware platform with proprietary systems to enhance the prototyping capabilities. By using the Arduino open source hardware platform, our results show that Verisure (a company that manufactures proprietary systems) are able to take advantage of the whole platform, including both electronic components and the associated libraries.

The results show that validating new ideas using the artifact can be done without the need to construct new hardware and have a working example in just one day. The constructed artifact enables the developer to experiment with third-party hardware for the Arduino platform. The artifact works as an entity within a network of other existing products from Verisure.

Using controlled experiments (see section 5.3.2 and 5.3.3) to evaluate the developed artifact, it is shown that the idea of rapidly prototyping a proprietary system using open source is possible. The combined impression from the developers that participated in the first controlled experiment was that the artifact would help the developers validate ideas quicker. The coordinator of the hackathon says that, although the developers in the product software team have not yet used the artifact to develop and test new ideas (apart from the hackathon), his judgment was that the artifact will make it easier for them.

The results also show that the overall impression from the audience of the presentation in the second controlled experiment was that the artifact was much appreciated and that there are plans to include the artifact in the production environment.
7.2 Contributions

The objective of this thesis was to enable rapid prototyping for companies with proprietary systems by combining their products with open source hardware. Based on the advantages of open source hardware that the authors Weiss and Rajala et al. states [6, 35], this thesis contributes to the problem domain area of the rapid prototyping method by extending these advantages to be applicable in proprietary systems as well. This thesis also contributes to the statement of Du Preez et al. [3] that further research must be done on the underlying stages of the “fugle model” by investigating one of the best practices.

7.3 Future Work

Future research lies in performing a case study at Verisure to observe if the artifact has made it easier for rapid prototyping. More work could be done to compare and identify the best open source platform. Moreover, only one technique was investigated for validating ideas and other techniques such as simulating, modeling and code generation should be investigated as well. Is it possible to take advantage of multiple platforms at the same time, or combining different techniques for the ultimate way of validating ideas?
References


## A List of Requirements

<table>
<thead>
<tr>
<th>Number</th>
<th>Requirement</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The solution should be in the form of a shield to be applied to a microcontroller.</td>
<td>Requirement from Verisure. A shield is a common way of applying additional hardware when programming with a microcontroller.</td>
</tr>
<tr>
<td>2</td>
<td>The shield should be able to benefit of an open source hardware ecosystem.</td>
<td>Requirement for this thesis. The technical solution should be able to benefit from the open source community.</td>
</tr>
<tr>
<td>3</td>
<td>The shield should build upon proprietary hardware.</td>
<td>To answer the research question, the hardware that benefits from open source must be proprietary/closed source.</td>
</tr>
<tr>
<td>4</td>
<td>The shield should be compatible with Arduino Uno.</td>
<td>Verisure wants to build on existing work if possible when prototyping, including a range of hardware modules such as sensors and actuators. Arduino is the desired platform according to Verisure.</td>
</tr>
<tr>
<td>5</td>
<td>A developer should be able to program with the Arduino IDE.</td>
<td>Arduinos own IDE is the desired development environment, both requested by Verisure, as well as being popular amongst the Arduino community.</td>
</tr>
<tr>
<td>6</td>
<td>The shield should consist of Mikrodusts RF Module.</td>
<td>Mikrodusts RF Module is already used in Verisures products and is the target when prototyping ideas.</td>
</tr>
<tr>
<td>7</td>
<td>The RF module firmware application should build upon the existing underlying OS.</td>
<td>The OS is already present in the module firmware and supplies basic functionality from the MCU.</td>
</tr>
<tr>
<td>8</td>
<td>The RF module firmware application should make use of the existing radio communication API.</td>
<td>The radio communication API implements security-based RF protocol that is necessary in Verisure products.</td>
</tr>
<tr>
<td>9</td>
<td>Developers should be able to prototype ideas with limited or no Verisure domain specific knowledge.</td>
<td>Verisure wants external developers (developers not working for Verisure) to be able use this technical solution.</td>
</tr>
<tr>
<td>10</td>
<td>The data sent and received by the shield should have a generic form/data structure.</td>
<td>The gateways and network at Verisure should not care what kind of data is sent, and just pass it by no matter the content.</td>
</tr>
<tr>
<td>11</td>
<td>The Arduino library should implement the most fundamental Verisure product functions.</td>
<td>When testing/prototyping an idea, the resulting code should be as close to an end product as possible, meanwhile giving the developer the freedom needed when programming.</td>
</tr>
<tr>
<td>12</td>
<td>The library should contain an uploadObject function.</td>
<td>Derived from requirement 11. Generic function to send data from to the gateway connected to the shield.</td>
</tr>
<tr>
<td>13</td>
<td>The library should contain a getNextMsg function.</td>
<td>Derived from requirement 11. Function to request the next message from the module.</td>
</tr>
<tr>
<td></td>
<td>The library should contain a reset-Shield function.</td>
<td>Derived from requirement 11. The user must be able to reset the shield via software, not only via the hardware reset button.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>15</td>
<td>The library should contain a downloadObject function.</td>
<td>Derived from requirement 11. Function to download the next incoming object from the module.</td>
</tr>
<tr>
<td>16</td>
<td>The library should handle a MsgAckSync response.</td>
<td>Derived from requirement 11. Acknowledgement sent from the gateway that the message sent has arrived.</td>
</tr>
<tr>
<td>17</td>
<td>The library should handle a MsgAck response.</td>
<td>Derived from requirement 11. Same as SR15, only asynchronous.</td>
</tr>
<tr>
<td>18</td>
<td>The library should handle a sendNextMsg response.</td>
<td>Derived from requirement 11. Response from the shield containing data, after the user has requested getNextMsg.</td>
</tr>
</tbody>
</table>
B. List of Implemented Library Functions (SHRPE.h)

```c
/* Initiate the shield */
int begin();

/* Handle shield events
* Shall be called by app in loop-function */
void loop();

/* Get the shield state
* return: the state */
shrpe_state_t getState();

/* Send an UploadObject message.
* The shield only supports one outgoing message at a time.
* It is possible to send in any state.
* If the message is not acknowledged the send is considered failed
* and the shield tries to re-connect automatically.
* param buffer: byte array that contains the data to send
* param size: number of bytes to send
* return: number of bytes sent or error (<0) */
int sendUploadObject(const uint8_t *buffer, size_t size);

/* Receive an UploadObject acknowledgment.
* param buffer: byte array that will contain the received data after the call
* param length: the length of the array
* return: number of bytes received or error (<0) */
int receiveUploadObjectAck(uint8_t *buffer, size_t length);

/* Receive a DownloadObject message, if any pending
* param buffer: byte array that will contain the received data after the call
* param length: the length of the array
* return: number of bytes received or error (<0) */
int receiveDownloadObject(uint8_t *buffer, size_t length);

/* Set contacts
* param contacts: bit array containing 8 contacts */
```
int setContacts(uint8_t contacts);

virtual int available();

virtual int read();

virtual int peek();

virtual void flush();

virtual size_t write(uint8_t byte);

virtual size_t write(const uint8_t *buffer, size_t size);

virtual int flushWriteBuffer();
C Library Example Usage - Native

```c
#include <Shrpe.h>
#include <SoftwareSerial.h>

Shrpe shrpe;
SoftwareSerial mySerial(10, 11); // RX, TX
shrpe_state_t currentState;
uint8_t loopCounter = 0;
bool isSending = false;

uint8_t upl_obj[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                     11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
                     21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
                     31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41};
uint8_t receiveBuffer[38];

void setup() {
  // init shield and serial port
  int state = shrpe.begin();
  mySerial.begin(115200);
  mySerial.println("Starting SHRPE demo example");
  mySerial.print("Current state: ");
  mySerial.println(state);

  // get current state
  currentState = shrpe.getState();
}

void loop() {
  // this loop executes approx once a second.
  // let shield lib execute
  shrpe.loop();

  // check shield state
  shrpe_state_t newState;
  newState = shrpe.getState();
  if (newState != currentState) {
    mySerial.print("New shield state: ");
    mySerial.println(newState);
    currentState = newState;
  }

  uint8_t ack_res[2];
  size_t buf_len = 2;
  int ack_len;
  if (isSending) {
    ack_len = shrpe.receiveUploadObjectAck(ack_res, buf_len);
    if (ack_len == 1) {
      mySerial.print("ack received?: ");
      mySerial.println(ack_res[0]);
      if (ack_res[0] == 1) {
        mySerial.print("ack result: ");
        mySerial.println(ack_res[1]);
        if (ack_res[1] == 0) {
          isSending = false;
        }
      }
    }
  }
}```
static uint8_t contacts = 0;

// alternate between 0 and 2
contacts = (contacts ^ 0x02);

int result = shrpe.setContacts(contacts);
mySerial.print("setContacts result: ");
mySerial.println(result);

// setContacts after an ack has been received
# Library Example Usage - Stream

```cpp
#include <Shrpe.h>
#include <SoftwareSerial.h>

Shrpe shrpe;
SoftwareSerial mySerial(10, 11); // RX, TX
shrpe_state_t currentState;
uint8_t loopCounter = 0;
bool isSending = false;

uint8_t upl_obj[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
                      11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
                      21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
                      31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41};
uint8_t receiveBuffer[38];
uint8_t oneByte = 5;

void setup() {
  // init shield and serialport
  shrpe.begin();
  mySerial.begin(115200);
  mySerial.println("Starting SHRPE demo example");

  // get current state
  currentState = shrpe.getState();
}

void loop() {
  // this loop executes approx once a second.

  // let shield lib execute
  shrpe.loop();

  // check shield state
  shrpe_state_t newState;
  newState = shrpe.getState();
  if (newState != currentState) {
    Serial.print("New shield state: ");
    Serial.println(newState);
    currentState = newState;
  }

  // time to send?
  int resultWrite;
  if (loopCounter % 5 == 0) {
    // resultWrite = shrpe.write(oneByte);
    resultWrite = shrpe.write(upl_obj, 41);
    if (resultWrite == sizeof(upl_obj) || 40) {
      int flushResult = shrpe.flushWriteBuffer();
      if (flushResult < 0) {
        mySerial.print("failed to flushWriteBuffer(), result: ");
        mySerial.println(flushResult);
      }
    }
    mySerial.print("wrote with result: ");
    mySerial.println(resultWrite);
  }

  // download an object if any
}```
int result;
while (shrpe.available()) {
    mySerial.print(shrpe.read());
    if (shrpe.available() >= 1) {
        mySerial.print(" ", ");
    } else {
        mySerial.println();
        mySerial.println();
    }
} // shrpe.flush();
if (result > 0) {
    // if object received
    mySerial.print("\nReceived download object of length: ");
    mySerial.println(result);
} delay(1000);
E Getting Started Guide
Arduino SHRPE Shield

Shield layout

Figure E.1: Overview of the shrpe shield and labels.

Jumper3 in figure E.1 should be removed when flashing the Arduino board. The other two jumpers should always be attached and are used for the UART communication. This design was chosen for future modifications of the communication protocols.

There are three LEDs on the shield indicating power (blue), connection status (green) and radio communication (red). As long as the shrpe shield is connected to the power source the blue LED is turned on. Similarly, the green LED remains turned on once a successful login has been established. The red LED serves two functions:

1. Indicates each hello-message that is sent before login successful.
2. Shows a successful radio- uploadObject transmission.

The red LED flashes at 200ms intervals continuously.

Using the Shield With Older Boards
The shield is guaranteed to work on Arduino UNO rev3. Additionally, it has been tested
to work with an Arduino MEGA board as well.

Installation Guide

Arduino SHRPE library source

The Arduino RF Shield library can be found on github:

https://github.com/Drotth/SHRPE-Arduino

Setting up the Library

1. Download the repository to a location of your choice.

2. Copy the entire repo (folder named “SHRPE-Arduino”) to your workspaces folder for custom libraries. By Arduino default (in windows), this is usually located under “My Documents/Arduino/libraries”.

3. Start your Ardino IDE and import the library to your sketch under “Sketch->Include Library->shrpe”. SHRPE-Arduino is seen furthest down in the list under “Contributed libraries”.

Native App Example

See Appendix C.

Stream App Example

See Appendix D.

Module source

The source code is proprietary and belongs to Mikrodust AB.

Newer firmware versions will be released as a FOTAimg. To update the shield, follow the steps below.

FOTA Upgrade

This section is proprietary for Verisure and can not be described publicly.
F Transcript of Questions and Answers

(Translated from original text in Swedish)

1. How long was the workshop?
   - 5h

2. How many projects did you manage to do?
   - 1-2 per group (some groups managed to do one, others two)

3. What is your opinion on how many projects that were successful?
   - The majority succeeded with their projects. However, they got to choose a project within what was supported, and the hardware that we had with us.

4. What was your general opinion at that time, were there a need of a lot of changes in both HW and SW or did it feel easy to work with?
   - Now I did not work with this, but studied what the other groups did, but my understanding was that it was pretty easy to use what was available. It was mainly support for downstream communications / commands that was the limitation.

5. How do you think this shield has helped you to develop and test new ideas, if you compare to how it was before?
   - To date, the Product Software team (which was the team that had the workshop) has not used this solution to develop and test new ideas, but my judgement based on what I saw at the workshop, is that it will help to make it easy when you want to test new ideas. If there exist hardware that can be used for exactly such an idea that you want to test (e.g. a good PIR whether to test something with motion detection), then I judge that the shrpe-shields will be of great use.