Framework and Development Process for IoT Data Gathering

Mika Saari, Petri Rantanen, Sami Hyrynsalmi and David Hästbacka

Abstract The Internet of Things (IoT) is a growing area in everyday life. New applications under the umbrella term IoT are being developed continually. This development has raised the need for framework definitions for different purposes. This research introduces a special software/hardware framework for data gathering systems to be used in IoT related systems. The purpose of the research is to show the usability of a certain software/hardware combination in prototype development. The software/hardware framework has been developed during several research projects by following the same prototype development process. This is proposed as a descriptive model for the prototyping process. The main contribution of this research is the framework itself. The framework consists of a model of the system with selected components. The placement of the sensor network is also presented. The purpose of the framework is to guide and assist the construction of data gathering prototypes. Furthermore, the advantages of the framework are to support re-usability, portability, and interchangeability. This research introduces the framework, its main components, and their interconnections. In addition, the prototype development process used is presented.

Mika Saari
Tampere University, Computing Sciences, Pori, Finland e-mail: mika.saari@tuni.fi

Petri Rantanen
Tampere University, Computing Sciences, Pori, Finland e-mail: petri.rantanen@tuni.fi

Sami Hyrynsalmi
LUT University, Department of Software Engineering, Lahti, Finland e-mail: sami.hyrynsalmi@lut.fi

David Hästbacka
Tampere University, Computing Sciences, Hervanta Campus, Finland e-mail: david.hastbacka@tuni.fi

1 Introduction

The Internet of Things (IoT) is a growing area in everyday life. New applications under the umbrella term IoT are being developed continually. The IoT paradigm is the integration of several technologies and communications solutions [1]. This development has raised the need for framework definitions for different purposes. For example, the draft of the IEEE standard [2] defines an architectural framework for the Internet of Things (IoT).

This article introduces a special software/hardware (SW/HW) framework for data gathering systems to be used in IoT related systems. The research question can be stated as follows: How to generalize the prototyping of IoT data gathering in a framework of required software and hardware components?

The research question was formulated during previous data gathering prototype system development projects. The main purpose of these prototypes is to gather data, for example environmental data such as temperature, humidity, or carbon dioxide levels (Fig. 1). The aim was to focus on the reproducibility of components within the development process. This study presents guidelines for selecting the required software and hardware components. The purpose of the SW/HW framework is to guide and assist when constructing data gathering prototypes. Furthermore, the advantages of the framework are that it supports re-usability, portability, and interchangeability.

This study is part of the research related to the Internet of Things (IoT) carried out by the Software Engineering and Intelligent Systems (SEIntS) group at Tampere University, Pori. The SW/HW framework has been developed during several research projects. These projects have contained multiple iteration rounds. Many of these rounds have produced a research article, whose main target was to describe the working prototype. The first prototype system was introduced by Saari et al. [3] in 2015. That research introduced the initial idea of a framework and a working implementation from it. Research on reducing energy consumption [4] presented the advantages of rapid prototyping with off-the-shelf devices and open source software.

The main idea of prototype development has been to start with off-the-shelf devices and open source software. These key software and hardware components are then modified in the desired direction and usually a working prototype system is produced.

The main result from our research is the framework itself. This has the ability to act as a guiding principle when developing new prototypes for gathering data. This framework also aims to represent the development of software and hardware usage in data gathering systems; in particular the evolution in the usage of both software and hardware is considered in different parts of the system. The framework could be used as a model when planning new data gathering prototypes for sensor networks.

The second finding made during the research is a descriptive model for the prototyping process (DMPP). This is a model of prototype development practices that have been applied in several research projects between university and enterprises (mostly small and medium-sized enterprises (SMEs)) in Finland. The purpose of the

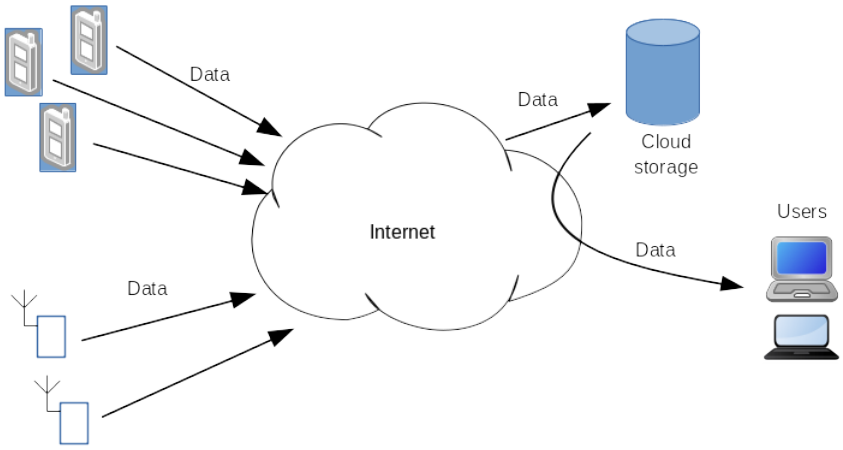

Fig. 1 The overall architecture of a data gathering IoT prototype system.

model is to introduce how academic research can conduct prototype development with regional enterprises.

The structure of this paper is as follows: In Section II, we review the related research about prototyping, the development process, and related frameworks. In Section III, we introduce an implementation of university-enterprise collaboration in prototype development described by means of process modeling notation. Section IV introduces the SW/HW framework for IoT data gathering. Section V continues by describing the validation and testing of the framework. Section VI includes a discussion and suggestions for future research on the topic and finally, Section VI summarizes the study.

2 Related research

The importance of prototyping embedded SW/HW systems was introduced by [5]. The reason for this was because system differences had increased, and the product relied mainly on variations in software and system features. In addition, involving users in the specification process is important because more and more customers expect solutions and services tailored exactly to their particular needs. In a more recent study, [6] developed a working prototype using off-the-shelf components. This also showed that a lengthy product development life cycle is not required when using a rapid prototyping process.

Rapid prototyping could be presented as a circle (Fig. 2). Rapid prototyping includes three stages: making a prototype, reviewing the result, and refining and iterating [7]. We used the idea of rapid prototyping in our projects. The working prototype solution in the context of IoT requirements is as follows: hardware to run

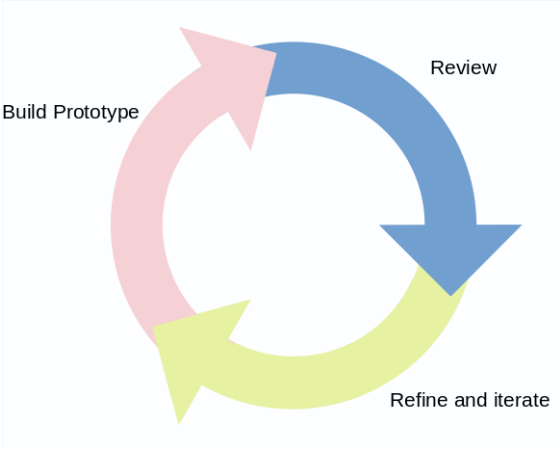

Fig. 2 The circle of the rapid prototyping process. Adapted from [7]

the software; software to collect, save, and transmit the data; the right technologies for the use cases to make things easier for both developer and user. [4]

2.1 Development process of the prototypes

The IoT prototyping process can be viewed from two perspectives: a software development process and an embedded hardware development process. Furthermore, the authors have researched the IoT data gathering prototype development process by collecting data from several prototyping processes.

A development process from the area of software development is suitable for this study. The developed prototypes and the presented SW/HW framework include a lot of software development. The process model could be descriptive or prescriptive. A prescriptive model tells how the process should be performed, whereas a descriptive model tells how a process is performed in a particular environment. The third option, a proscriptive model, also describes the activities that could be done [8]. The descriptive process model (DPM) [9] [10] introduces an eight step approach for producing a process model. These steps are divided into two phases: the set-up phase and the execution phase. The eight step approach was used when the descriptive model for the prototyping process (DMPP) was developed. The DMPP is presented in Section 3.

The book "Introduction to Embedded Systems - A Cyber-Physical Systems Approach" by Lee and Sashia is based on the idea that designing and implementing an embedded system consists of three major parts of the process: modeling, design, and analysis [11]. The modeling phase specifies what a system does by defining the system model and the set of requirements. The artifacts, such as a combination of software and hardware components, are produced in the design phase. An artifact

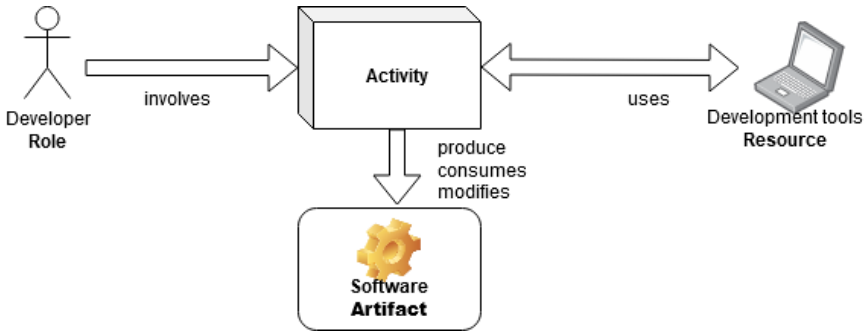

Fig. 3 Example of basic concepts related to the prototype development process. Adapted from [18]

is a working system and it describes how the system works. In the analysis phase, information on the system is obtained to specify why the system works as it works. [11]

2.2 The framework for prototyping

The SW/HW framework idea is not a new issue in the research field. For example, earlier studies [12], [13], [14] have addressed the framework subject from a real-time system perspective. In these studies, the design was at micro-controller level, whereas our prototypes use off-the shelf single-board micro-controllers. For example, Srivastava and Brodersen handled board level module generation, system software generation, and hardware-software integration in a unified framework [14]. Their study mentions a rapid prototyping method, but it was not explained further.

IoT architecture consists of several components, which can be divided into layers as follows: sensing layer, networking layer, service layer, and interface layer [15], [16]. The SW/HW framework is focused on the sensing and service layers. The networking layer exists but is not the focus of our research. The interface layer is described to the user in the SW/HW framework but is excluded from the study.

A wireless sensor network (WSN) can be used in various application areas. A WSN includes sensor nodes, which consist of sensing, data processing, and communicating components. A sensor network is composed of a large number of sensor nodes, which send data to the data storage. Since sensor nodes have data processing ability, the uploaded data can be either raw or pre-processed [17].

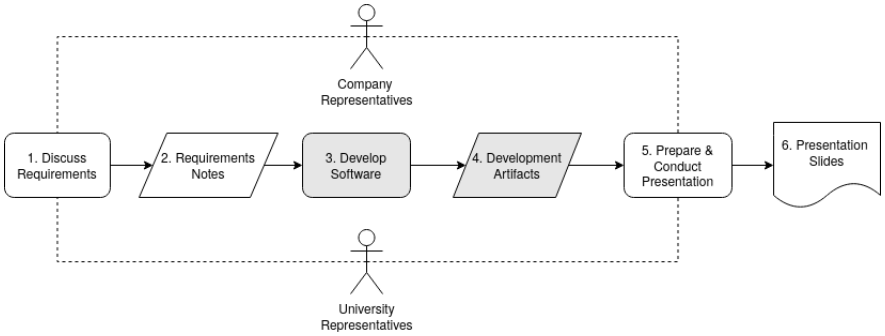

Fig. 4 Process model for prototype development. Adapted from [18].

3 Descriptive model for the prototyping process (DMPP)

We introduced our descriptive software process model for IoT prototyping in [18]. The purpose of this section is to present how the selected process model has supported the development of the framework.

The DMPP [18] could be extended to include hardware, because the model itself does not limit the type of artifact. Therefore, when the process is mentioned in this study, it generally means every kind of artifact, i.e., software or hardware, made in the prototype development process. The focus is on modeling the prototyping process in a research context, but its use in general is not restricted.

The DMPP was developed using the aforementioned descriptive process model (DPM) approach [10]. The basic concepts related to processes are role, activity, resource, and artifact (Fig. 3). The example is illustrated by the developer (role) involved in software developing (activity) using a programming tool (resource). The activity produces some software (artifact) used in the prototype system. The process data for the model is collected through interviews with the developers involved in the four different prototype development processes. These four prototype development projects and their outcomes are reported in several studies [19], [20], [21], [22]. Common to all of the studies are that they present developed IoT prototype systems that gather data. Although the software and hardware components in the prototypes vary, overall they can be used to model the prototype development process.

Fig. 4 presents the developed DMPP [18]. The model includes six steps. These steps support or use the SW/HW framework in the following ways:

1. The first step starts from the requirements definition, a collaborative discussion between the developers and the client. The client defines what kind of data would be useful. The developer group starts to define the hardware and overall architecture of system and how the data will be collected by the software. The selected hardware mostly determines the software environment and tools used.
2. The outcome of the discussion is the first artifacts: for example, the prototype system requirements within the discussion notes. The developer group constructs the first architecture model of the component interconnections.

3. The third step is the software/hardware prototype development made by the research group including the project manager and software/hardware developers. The clients' representatives are involved in the development process in the role of instructor. In this step the SW/HW framework is used as the guideline for selecting the components for the developed prototype.
4. The fourth step introduces the working prototype artifact. It contains the developed software and hardware components. Also, the interconnections of the components are tested. The testing process overall is usually only the functional testing of the prototype system. Additionally, the gathered data are inspected and if possible, compared to the expected results.
5. The fifth step includes preparing the outcome of the development process. The SW/HW framework can be complemented if necessary.
6. The sixth step is to publish the results, for example, the prototype system, the collected data, and the analysis of the project.

The process model in Fig. 4 is a simplified presentation of the prototype development process and therefore it does not mention common procedures such as iterations, testing, and customer testing.

Iterations are an efficient way to test and develop an idea. The first working prototype is made as simple as possible with basic components. For example, the hardware could be chosen at first only for testing the idea. If the idea works, the hardware is changed for more suitable hardware in the next iteration round. A good example of this is the application where we tested the use case [23]: Is it possible to take a photo in selected GPS coordinates automatically and send this photo to the cloud storage? This idea was tested with the Android application in a smartphone and the idea was found to be a workable solution. The smartphone had some limitations with the automation: The developed photographing application had to be started, it was not possible to be aware of the application crashing, and the possibility of remote control was not easily implemented. The second iteration round to solve the same use case was carried out with the following hardware: Raspberry Pi, camera, battery, GPS sensor, WiFi, and 4G modem. This time the Raspberry Pi OS made it possible to implement the automated operations and remote control with Linux tools.

4 SW/HW Framework for IoT data gathering

In this section we introduce the SW/HW framework for an IoT data gathering system. The framework consists of several hardware and software components. The purpose and advantage of the framework is to support re-usability, portability, and interchangeability. Another purpose of the framework is to guide and assist the construction of data gathering prototypes.

The definition of the SW/HW framework has been made in several academic research projects where the focus has been the collection of data using self-made

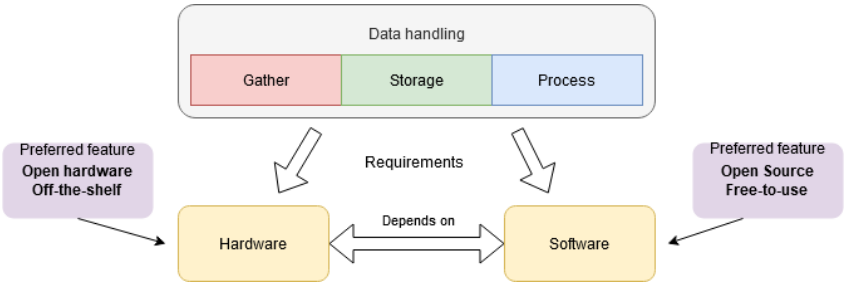

Fig. 5 Relationships of the SW/HW framework.

prototype systems. Fig. 5 illustrates the framework and its relationships. The previously presented model in Section 3 has connections to the framework and its development. DMPP step 1 gives guidelines for data handling. The client tells the developers what kind of data are useful. In step 3 the developers make the decisions on what kinds of software and hardware components are needed to fulfill the clients' data gathering expectations.

Furthermore, Figure 5 clarifies the interconnections of the framework components. These interconnections guide the selection of software and hardware components when constructing a prototype. The framework relies on off-the-shelf devices, because this speeds up development by minimizing hardware design and implementation [24]. The information that is to be collected determines the collection and structure used by the software and hardware components. Optional features are also used when selecting the components. In hardware, open hardware and off-the-shelf devices are preferable, because they are reasonably priced, quickly available, and have community support. The software components should have similar features: open source and community support. Please note that these criteria are not preferred when implementing the final application for production use.

The first question for the constructor of an IoT data gathering system when implementing the prototype is what information or data to collect. The answer to the question should be clear and it should also be the motivator of constructing the system. The next three questions are presented after the decision to collect data:

- How to gather data?
- How to store data?
- How to process data?

For an end user or client the availability of the data is important. Therefore the preferable place to store and process the data is in some cloud storage or server. Cloud storage could be some old Linux server or fully optimized commercial cloud computing service. However, before the data are in cloud storage, they have to be collected and stored temporarily in a sensor device or gateway device. Furthermore, the first data processing and storing should be managed by the sensor device or gateway device. For example, if the network has a communication problem, there is the possibility of losing data unless the data are forwarded rather than being stored

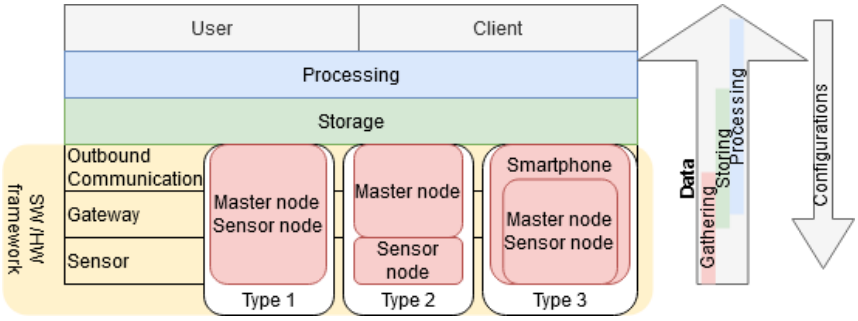

Fig. 6 Layer diagram of IoT data gathering and how the SW/HW framework is placed in it.

temporarily. The SW/HW framework focuses on data storage and processing in the sensor or gateway layers.

Fig. 6 shows the overall layer architecture of the IoT data gathering prototype system in a sensor network. The presented SW/HW framework is located in the lower part, containing the sensors, gateways, and preferable means of communication to operate with cloud storage. A few explanations of the Fig. are given below:

- The user and client layer utilizes the collected data.
- A processing layer is needed in most use cases. The data are processed in the way the clients require. The user can use the raw, unprocessed data.
- The storage layer collects and saves the data. The purpose of this level is to ensure data retention.
- The outbound communication layer belongs to the SW/HW framework. Its purpose is to offer a suitable data transfer method.
- Data gathering has three different hardware constructions: Type 1 - master node-sensor node combination; Type 2 - several sensor nodes collect the data in one master node; Type 3 - Fully operational sensor device - a smartphone collects the data.
- The data flow from the sensors to the client or user. The data could be temporarily stored on the gateway level. The data processing could be also done on then gateway level if it is necessary and possible.
- Configurations and monitoring are enabled to ensure faultless operation and for testing purposes.

The division of the master node - sensor node into three different types of construction supports the versatility of the SW/HW framework. The main idea is that these three types make it possible to collect a wide range of data.

Table 1 The main features of three different types of data gathering constructions

	Type 1	**Type 2**	**Type 3**
Basic construction	SBC with sensor(s)	SBC master node and group of sensor nodes with sensors	Smart phone
Data gathering	No limitations – suitable for large data chunks such as photos	Suitable for low data transfer – SBC limitations	Device sensors – no hardware modifications
Data processing	SBC limitations	Mean value calculus, visualizing	Mean value calculus
Data storage	No limitations temporary storage	No limitations database storage	No limitations temporary storage

4.1 Hardware of the SW/HW Framework

The data gathering hardware can be divided on a higher level into three type of constructions, as can be seen in Fig. 6. The framework uses off-the-shelf hardware and devices. This limitation accelerates prototype development as at least partially tested devices can be used. The hardware can be categorized in two parts:

- Sensor node - Sensor hardware consisting of a combination of sensors and control device.
- Master node - data gathering and storing device that has the capability of collecting, storing, and processing data.

The division into two parts is enough for hardware when compared to the three types of data gathering devices in Table 1. On the hardware side, types 1 and 3 are embedded together - the sensors and processing capabilities are in one device. In type 2 the master node can control several sensor nodes.

The **sensor node** collects the data with sensors. The data could be simple data, such as temperature and humidity. On the other hand, the data could be more complex, such as photos. The hardware is selected according to the data requirement. The separation of sensor node is made because of type 2 where the idea is to use several sensor nodes with one master node. In type 2 construction sensors are connected to a single-board micro-controller, such as Arduino or similar, which can handle a lot of simple data. Simple data are numerical values. Type 2 sensor nodes are simple, low-cost devices. These are connected to the master node and the amount of transferred data should be in bytes or kilobytes. The preferred communication methods include Bluetooth, ZigBee, and LoRa for short distance, low rates, and low power consumption [25], [26]. Types 1 and 3 are similar to each other; both are physically one entity: Type 1 consists of sensors and an SBC such as Raspberry Pi. The collected data could be complex and may need processing power, for example, photos. Type 3 is smartphone based solutions. A typical smartphone has several sensors, e.g., gyroscope, accelerometer, and ambient light sensor [27]. For example, Android phones could handle a lot of simple data from their own sensors, or complex data such as photos from the phone's own camera.

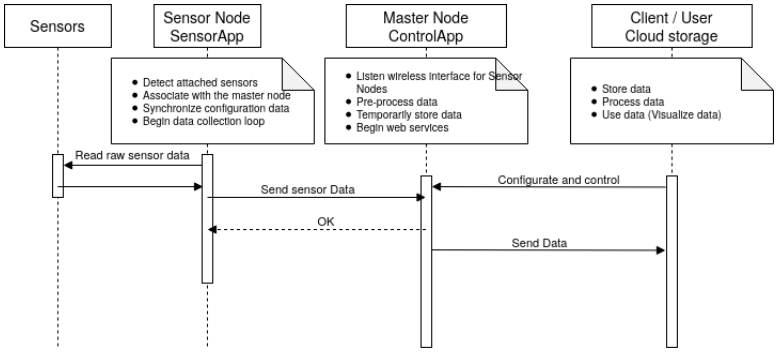

Fig. 7 Diagram of software component interconnections in the SW/HW framework.

The **master node** collects data from the sensor nodes. The master node has the capability to pre-process or temporarily store data if needed. The master node has a communication channel for a larger data transfer rate and long distance, for example, 3G / 4G / 5G, or WiFi are suitable. Depending on the master node's communication channel, remote control and configuration are possible. For example, with the Raspberry Pi the remote configuration is easily constructed with a suitable communication channel and Linux OS tools. The idea of a master node in type 3 smartphone based solutions is implemented with a self-made application, which handles data collection, storage, and processing. The application limitations come from the phone's OS and the fact that hardware changes or modifications have not been made.

This SW/HW framework relies on communication to the public Internet. The collected data is transferred via the Internet to the data storage devices. These could be cloud servers with a database or dedicated open source Linux servers for saving data. There are several database models for storing sensor data and each of these has a special use case where they are best. The SW/HW framework can be applied to all of these techniques.

4.2 Software of the SW/HW Framework

The hardware of the SW/HW framework also requires software. The software used is mostly open source. In this way the selected software is community tested and the source code is freely available. Open source software is also free to use. Therefore, several software combinations can be used for testing purposes without extra costs.

The software components are divided into three parts:

- Sensor software - gets sensor data from sensors
- Data gathering and pre-processing software
- Data storage software

Regarding how these three parts work together, the diagram in Fig. 7 is a guideline for dividing software components between the master node and sensor node on the abstract level. This kind of approach supports the modular development and interchangeability of components. The diagram also illustrates the input from the sensors and the output to the cloud.

The list of software components starts from the sensor node with sensor software and ends with data storage software for the master node. The sensing and sensor software typically have low-level programming with C++, Python, or Perl scripts, which are dedicated to do a few tasks, for example reading sensor data, reviewing the data, and storing the data. Sensor devices, such as Arduino with sensors, are typically a micro-controller board that runs a dedicated program. Types 1 and 2 in Fig. 6 represent this kind of approach.

The Raspberry Pi based SBCs can perform both sensing and data gathering. In type 1 for example, the sensors could be connected to the Raspberry Pi and the sensor software reads the values from the sensors. If the construction is type 2, the gathering software handles the data collection from sensor nodes. The data gathering and preprocessing software are more complex and usually type 2 SBC devices are equipped with the full Linux operating system (OS). In the data storing and preprocessing phases, the SW/HW framework utilizes pre-made software and libraries. For example, Raspberry Pi could offer the gathered data to the Internet with a server application. Preprocessing in this scenario could be image recognition with image recognition software and library.

The gathered data are stored in the cloud server - this is the assumption of the SW/HW framework. Temporarily, the data could be saved to the master node using a suitable database. If the data meet the definition of a time series: "A sequence of numbers collected at regular intervals over a period of time" then a good choice is a time series database [28]. For example, the open-source time series database InfLuxDB is suitable for SBC hardware and is widely used in IoT solutions [29].

The other suitable database model for data storage is a relational model. The collected data could be stored locally in the sensor device, for example Raspberry Pi with Linux OS, MariaDB database, and a RESTful API combination. The RESTful API (Application Programming Interface) method allows remote control or management of a device over the network.

Smartphones equipped with the Android OS have been tested with this SW/HW framework. The Android OS has a software development kit (SDK), which enables the wide use of smartphone capabilities. For example, the SDK enables phone camera usage [23]. The SDK also enables usage of the smartphone's accelerometer sensor [30].

Data storage on the mobile phone is enabled by the OS. The SDK provides the capabilities to use files and databases for data storage. In terms of the SW/HW framework, the user should be able to use the data. The SDK also enables data transfer to cloud services.

Cloud storage for data is a better choice than local storage. Cloud storage could be, for example, a Linux server or maintained commercial cloud service such as

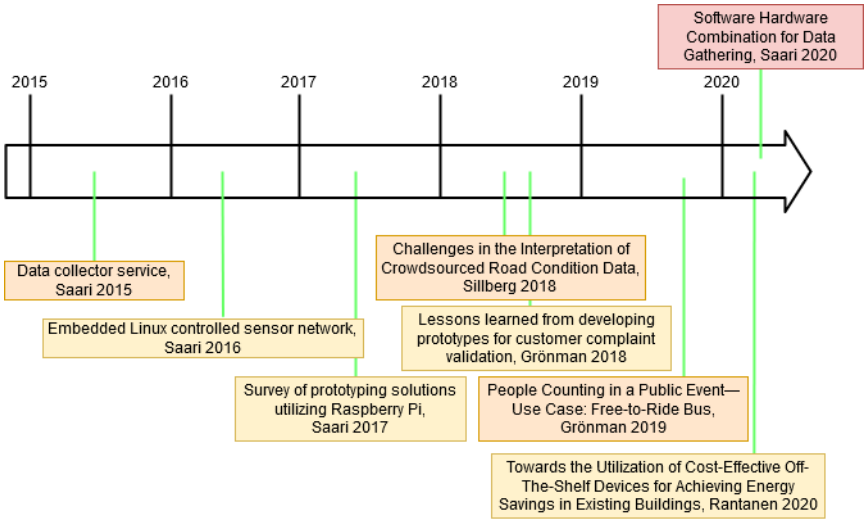

Fig. 8 Prototypes and validation in the SW/HW framework timeline.

Google Firebase. Both of these have more capabilities to store a larger amount of data than the local database in Raspberry Pi.

5 Validating the SW/HW Framework by prototyping

The SW/HW framework has been developed and tested during several research projects. The majority of IoT data gathering prototype systems and their findings have been reported in different studies [3], [31], [32], [30], [23], [22], [33]. The timeline of studies is presented in Fig. 8. The timeline also includes the first release of the SW/HW framework study [34]. These projects have contained multiple iteration rounds. Each iteration has produced a working prototype system. This section gives an overview of the systems and lists the main findings during the development of the system. The systems are divided into the previously presented types 1, 2, and 3.

5.1 Type 1 and type 2 with SBC related prototype systems

Types 1 and 2 are SBC and Linux OS based data gathering prototype systems. Type 1 usually contains one sensor connected directly to the SBC.

- A data collector service [3] was the first implementation. The SBC was a Beaglebone Black which gathered temperature, humidity, and brightness data. The data were stored in the SBC and there was a service to offer the data to users. The SBCs

were connected with an Ethernet connection. The main goal of the study was to experiment how well a cost-efficient SBC could be used to gather sensory data, and how this data could be provided to the client over the public Internet. This goal was reached successfully, and the designed system was tested and found to work as planned [3]. The study proved that fully working data gathering prototypes can be developed with off-the-shelf devices and open source tools.

- The ability to use off-the-shelf devices prompted us to find out how others have used these devices in academic research. A survey of prototyping was made to find out the benefits and limitations of Raspberry Pi [32]. Also, we searched for the testing methods of these prototypes. The study showed that the Raspberry Pi is a widely used device in research implementations of different kinds. Some testing methods of prototype systems were found: software testing, software performance testing, and validation of data tests [32]. This study clarified the operating environment for the SW/HW framework.
- The third [23] and fourth [22] studies were similar to each other. Both prototypes presented in the studies were based on a Raspberry Pi and camera combination in a vehicular environment. The power supply was a battery backup and both used a 3G/4G communication channel. The data, photos, were transferred to the cloud storage where the data processing was handled. The focus of the studies was to analyze and process the data on a cloud server but the functionality of the SBC based prototype system was also ensured. Both of the systems were located in a client's environment for several weeks to collect data. The long testing time clearly showed that the SW/HW framework needs a configuration channel from user to device. In these systems, the SSH service and terminal were used as a remote channel.

5.2 Type 2 with SBC related prototype systems

The type 2 configuration has been tested in three real-world projects. The common construction of type 2 data gathering prototypes is that one master node SBC works with several sensor nodes. The communication is one-way from sensor nodes to the master nodes. The master node with a group of sensor nodes was tested in several research cases:

- The first was [31] where one SBC master node, an Intel Galileo Gen 2 development board, communicated using wireless XBee technology with several sensor nodes, on an Arduino development board. The collected data were environmental. The targets of the study were threefold: to test the model, to determine how well cost-efficient SBCs could be used to gather sensory data from several sensor nodes, and how to deliver this data to clients over the public Internet [31]. The study showed that SW/SW frameworks need a separate sensor node - master node architecture. This is useful when several sensors are required in a small area.
- The second use case [33] presented a wireless sensor system for monitoring indoor living or working conditions. The study expanded the range of sensors by

using wireless LoRa technology in communication between the master node and sensor nodes. In the main construction, the master node Raspberry Pi received environmental data from several Sodaq sensor nodes. The data were stored in the master node for analysis and processing. The primary purpose of the tests was to validate the sensor system construction. Based on the experiments, we found that LoRa was a good choice for sensor applications within concrete buildings [33].

- The third use case used commercial, but open hardware and open source, RuuviTag sensor nodes [35]. These cost-efficient sensors collected temperature, humidity, pressure, and motion information. The data were sent using Bluetooth communication to the Raspberry Pi master node. The data were stored on an InfluxDB database and visualized with Grafana visualizing software. One of the RuuviTag experiments is documented in a study [36]. This prototype system is quickly configurable, because RuuviTags do not need a configuration; only the Raspberry Pi needs a setup. These prototype systems showed that even though the gathered data were small in quantity, the Raspberry Pi limited the visualization. The second issue raised was a limitation in the amount of memory operations with a Raspberry Pi memory card. Therefore the prototype system experiment showed that data should be transferred and stored on a cloud server. From the perspective of the SW/HW framework, this prototype system showed the usability of modular development and the fact that the sensor nodes and master node are interchangeable.

5.3 Type 3: Smartphone related prototype systems

The smartphone is an excellent WSN sensor node. It has a working hardware package: power source, communication skills, and sensor devices. It also has a suitable OS, which allows the wide usage of the hardware. Our first ideas of using the smartphone as a sensor were presented in [37]. The main question presented was "How to utilize mobile technology to supply disaster information to both mobile terminals and desktop computers?" [37].

In the recent type 3 prototype systems, we used Android smartphones. These two documented data gathering prototypes are next discussed from the perspective of the SW/HW framework.

- The first data gathering prototype implementation was presented in study [30], which utilized data collected by smartphone sensors, such as an accelerometer and GPS, to detect variations in road surface conditions. The use of a smartphone was preferred because the data were obtained from a group of users driving on roads in western Finland and most of these people owned a smartphone that was good enough for prototyping purposes. The data gathering device was therefore an off-the-shelf Android smartphone without any hardware modifications. The software was a combination of user interface application and background service. The results of the study were that the basic programming task of creating a simple application for tracking the user's location and gathering data from the basic sen-

sors embedded in a mobile device is a straightforward process. Minor difficulties arose because of variations in the phones' basic software and hardware depending on the manufacturer. In relation to the SW/HW framework, the prototype showed the ability to use a smartphone as a WSN sensor node.

- The second data gathering prototype was a solution for idea testing. The question asked was: Is it possible to take a photo if the phone is in a selected GPS position? Study [23] presents the working prototype solution for this problem. For the SW/HW framework, this prototype again showed the ability to use a smartphone as a WSN sensor node. The program code utilized an implementation of the haversine formula, which determines the great circle distance between two locations and is relatively simple to implement yet accurate enough for our use cases. The application can be installed on any reasonably new Android device and takes advantage of the built-in sensors and camera of the device.[23]

6 Discussion

This study aims to resolve the research question: How to generalize the prototyping of IoT data gathering in a framework of required software and hardware components?

To answer the question, we made a reasonable number of data gathering prototypes and reported on them in academic research papers. Thus, our earlier research answered the question. The studies highlighted several aspects in constructing a SW/HW framework. The framework describes the main findings from developed prototypes. The framework brings out three different approaches for different use cases. The research papers presented how prototyping development can be made cost efficiently. This was enabled by using off-the-shelf embedded devices such as smartphones and Raspberry Pi SBCs. The devices have the type of OS that can be modified for sensor usage.

Furthermore, the prototype highlighted the knowledge we have about the process of prototyping. The model can be used as guidance when designing a new prototyping project together with a customer who wants to obtain information about some target environment.

The development of the SW/HW Framework raised several new ideas for research topics. These topics are briefly discussed here.

Each of the prototypes discussed has sensor software: software that reads the sensor, a temperature sensor for example, changes the value form bytes to an integer with a reasonable formula, and sends or stores the value somewhere. The study mentions the sensor software several times, but its construction has not been discussed in detail. This low-level program is coded using C/C++, Python, Java, or a similar programming language. It should be noted that at this level the initial data processing could be done, e.g., the mean value of accelerometer sensor values within one second. Is it possible to get improved performance without data loss? How should this low-level software be programmed? These would be suitable questions for future study.

The topics of user and user experience are beyond the scope of this study. Our prototypes were developed due to the needs of some project partner. The prototypes were tested with use case testing and once the customer had received a reasonable answer to a certain need, the development was stopped (except for one example [21] where there was a long piloting period in a real usage environment). The project outputs and prototypes are freely exploitable by the project partners.

The sensor prototypes produce a large amount of sensor data. Data processing and data mining are important issues, which this study leaves for future research, as it is such an extensive area. The issues of data visualization have been handled in some studies [30]. In addition, sensor data will become more usable if merged with other publicly available data. This kind of data could be weather data or map data [38], [21].

Performance problems have not been extensively addressed in this study, but when using the SW/HW framework they have to be taken into account. In data gathering construction type 2, the amount of sensor nodes is limited, but no exact limit can be set. The limit is changed by the range, communication channel, data to be transferred and so on. The data processing can also affect the performance problems. For example, in this study photos are often mentioned as difficult regarding performance, especially motion detection. For example, photos should be transferred to a cloud server for processing.

The SW/HW framework does not set the quality criteria for components, but how can the selection of high-quality components be ensured? "The hardware quality depends on the price" is one claim, which in most situations makes sense. The second level for selection is "good enough". The SW/HW framework does not set these kinds of selection criteria for software or hardware components and therefore these decisions are left to the framework user.

The vulnerability of data is worth considering. Is the data critical and what happens if we cannot obtain the data? Is it possible to manipulate the data and what would the consequences be in that case? For example, what happens if somebody changes the data. This SW/HW framework does not take a position on the matter, but these are significant issues. Furthermore, security issues are important for IoT devices. Security vulnerabilities and attacks on IoT systems have been covered extensively by [39]. The SW/HW framework does not pay attention to security except for the communication channel. This concern was raised in [31] and the proposed, more secure, communication technology LoRa has been discussed by [40].

7 Summary

This paper introduced the software/hardware framework and a descriptive model for the prototyping process. The framework was developed during several research projects by following the same prototype development process. The model presented the process for constructing and testing a data gathering prototype with six

steps, starting from discussion of requirements and ending with the presentation of collected data. The main aspects of these steps were presented briefly.

A sensor network consists of several layers, from data gathering devices to data users. The framework is placed in the data gathering layer. The three types of data gathering constructions were presented by introducing the software components, the hardware components, and their interconnections.

Research findings: The model and the framework were validated by presenting several previous research projects and studies.

Acknowledgements This work is part of the KIEMI ("Vähemmällä Enemmän – Kohti Kiinteistöjen Energiaminimiä", or "Less is More: Towards Energy Minimum of Properties" in English) project and has been funded by the European Regional Development Fund and the Regional Council of Satakunta.

References

1. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.
2. Ieee approved draft standard for an architectural framework for the internet of things (iot), 2019.
3. Mika Saari, Pekka Sillberg, Petri Rantanen, Jari Soini, and Haruka Fukai. Data collector service - practical approach with embedded linux. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, International convention on information and communication technology, electronics and microelectronics, pages 1037–1041. IEEE, 2015.
4. Mika Saari, Pekka Sillberg, Jere Grönman, Markku Kuusisto, Petri Rantanen, Hannu Jaakkola, and Jaak Henno. Reducing energy consumption with iot prototyping. *Acta Polytechnica Hungarica*, 16(9, SI):73–91, 2019.
5. Klaus Buchenrieder. Rapid prototyping of embedded hardware/software systems. In *Proceedings. Ninth International Workshop on Rapid System Prototyping (Cat. No.98TB100237)*, volume 5, pages 2–3. IEEE Comput. Soc, 2000.
6. C P Kruger, A M Abu-Mahfouz, and G P Hancke. Rapid prototyping of a wireless sensor network gateway for the internet of things using off-the-shelf components. In *2015 IEEE International Conference on Industrial Technology (ICIT)*, pages 1926–1931, 2015.
7. Nick Babich. What is rapid prototyping?, 2019. Accessed: 2020-11-16.
8. Chris Jensen and Walt Scacchi. Discovering, modeling, and re-enacting open source software development processes: a case study. In *New Trends in Software Process Modeling*, number February 2006 in Series on Software Engineering and Knowledge Engineering, pages 1–20. World Scientific Publishing Company, 2006.
9. Ulrike Becker, Dirk Hamann, and Martin Verlage. Descriptive Modeling of Software Processes. *IESE-Report No. 047.97/E*, 1997.
10. Ulrike Becker-Kornstaedt and Richard Webby. A comprehensive schema Integrating Software Proces Modeling and Software Measurement. *IESE-Report No. 047.99/E*, 1999.
11. Edward Ashford Lee and Sanjit A. Seshia. *Introduction to Embedded Systems. A Cyber-Physical Systems Approach. Second Edition*, volume 195. 2017.
12. C. Kreiner, C. Steger, E. Teiniker, and R. Weiss. A HW/SW codesign framework based on distributed DSP virtual machines. *Proceedings - Euromicro Symposium on Digital Systems Design: Architectures, Methods and Tools, DSD 2001*, pages 212–219, 2001.

13. D. Saha, R.S. Mitra, and A. Basu. Hardware software partitioning using genetic algorithm. In *Proceedings Tenth International Conference on VLSI Design*, pages 155–160. IEEE Comput. Soc. Press, 1997.
14. M.B. Srivastava and R.W. Brodersen. Rapid-prototyping of hardware and software in a unified framework. In *1991 IEEE International Conference on Computer-Aided Design Digest of Technical Papers*, pages 152–155. IEEE Comput. Soc. Press, 1991.
15. Li Da Xu, Wu He, and Shancang Li. Internet of Things in Industries: A Survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, 2014.
16. Alex Vakaloudis and Christian O'Leary. A framework for rapid integration of IoT Systems with industrial environments. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 601–605. IEEE, 2019.
17. IF Akyildiz, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.
18. Mika Saari, Jari Soini, Jere Grönman, Petri Rantanen, Timo Mäkinen, and Pekka Sillberg. Modeling the software prototyping process in a research context. In M. Tropmann-Frick, B. Thalheim, H. Jaakkola, Y. Kiyoki, and N. Yoshida, editors, *Information Modelling and Knowledge Bases XXXII*, volume 333, pages 107–118. IOS Press, 2021.
19. Jere Grönman, Petri Rantanen, Mika Saari, Pekka Sillberg, and Juha Vihervaara. Low-cost ultrasound measurement system for accurate detection of container utilization rate. In *2018 41th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2018.
20. Jari Soini, Pekka Sillberg, and Petri Rantanen. Prototype system for improving manually collected data quality. In Zoran Budimac and Tihana Galinac Grbac, editors, *Proceedings of the 3rd Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, SQAMIA 2014, September 19-22, 2014, Lovran, Croatia*, Ceur workshop proceedings, pages 99–106. M. Jeusfeld c/o Redaktion Sun SITE, 2014.
21. Jari Soini, Markku Kuusisto, Petri Rantanen, Mika Saari, and Pekka Sillberg. A Study on an Evolution of a Data Collection System for Knowledge Representation. In A. Dahanayake, J. Huiskonen, and Y. Kiyoki, editors, *Information Modelling and Knowledge Bases XXXI*, volume 321, pages 161 – 174. IOS Press, 2019.
22. Jere Grönman, Pekka Sillberg, Petri Rantanen, and Mika Saari. People Counting in a Public Event—Use Case: Free-to-Ride Bus. In *2019 42th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2019.
23. Jere Grönman, Petri Rantanen, Mika Saari, Pekka Sillberg, and Hannu Jaakkola. Lessons learned from developing prototypes for customer complaint validation. In *Proceedings of the SQAMIA 2018: 7th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*, volume 2217, pages 27–30. CEUR Workshop Proceedings, 2018.
24. Fuewen Frank Liou. *Rapid Prototyping and Engineering Applications*. Taylor & Francis, CRC Press, Boca Raton, second edition, feb 2019.
25. Marco Centenaro, Lorenzo Vangelista, Andrea Zanella, and Michele Zorzi. Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios. *IEEE Wireless Communications*, 23(5):60–67, oct 2016.
26. Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara. Low Power Wide Area Networks : An Overview. *IEEE Communications Surveys & Tutorials*, 19(2):855–873, 2017.
27. Kodrat Iman Satoto, Eko Didik Widianto, and Sumardi. Environmental Health Monitoring with Smartphone Application. In *2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, pages 281–286. IEEE, 2018.
28. Dmitry Namiot. Time series databases. *Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL2015)*, 1536:132–137, 2015.
29. Andreas Bader, Oliver Kopp, and Michael Falkenthal. Survey and comparison of open source time series databases. *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)*, 266:249–268, 2017.
30. Pekka Sillberg, Jere Gronman, Petri Rantanen, Mika Saari, and Markku Kuusisto. Challenges in the Interpretation of Crowdsourced Road Condition Data. In *2018 International Conference on Intelligent Systems (IS)*, pages 215–221. IEEE, 2018.

31. M. Saari, A. M. Baharudin, P. Sillberg, P. Rantanen, and J. Soini. Embedded Linux controlled sensor network. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1185–1189. IEEE, 2016.
32. Mika Saari, Ahmad Muzaffar bin Baharudin, and Sami Hyrynsalmi. Survey of prototyping solutions utilizing Raspberry Pi. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 991–994. IEEE, 2017.
33. *Towards the utilization of cost-effective off-the-shelf devices for achieving energy savings in existing buildings*. IEEE, 2020.
34. Mika Saari, Petri Rantanen, and Sami Hyrynsalmi. Software hardware combination for data gathering. In *Proceedings of 2020 IEEE 10th International Conference on Intelligent Systems (IS2020)*, 2020.
35. Ruuvitag technical specifications, https://ruuvi.com/files/ruuvitag-tech-spec-2019-7.pdf, 2019. Accessed: 2020-11-23.
36. Mika Saari, Jere Grönman, Jari Soini, Petri Rantanen, and Timo Mäkinen. Experimenting with Means to Store and Monitor IoT based Measurement Results for Energy Saving. In *2020 43th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. IEEE, 2020.
37. Pekka Sillberg, Petri Rantanen, Mika Saari, Jari Leppäniemi, Jari Soini, and Hannu Jaakkola. Towards an IP-based alert message delivery system. In J Landgren and S Jul, editors, *ISCRAM 2009 - 6th International Conference on Information Systems for Crisis Response and Management: Boundary Spanning Initiatives and New Perspectives*, number June 2015, page 8 p. Information Systems for Crisis Response and Management, ISCRAM, 2009.
38. Pekka Sillberg, Mika Saari, Jere Grönman, Petri Rantanen, and Markku Kuusisto. Interpretation, Modeling, and Visualization of Crowdsourced Road Condition Data. In R Goncalves, V Sgurev, V Jotsov, and J Kacpzyk, editors, *Intelligent Systems: Theory, Research and Innovation in Applications*, Intelligent Systems: Theory, Research and Innovation in Applications, pages 99–119. Springer, 2020.
39. Sachin Babar, Antonietta Stango, Neeli Prasad, Jaydip Sen, and Ramjee Prasad. Proposed embedded security framework for Internet of Things (IoT). In *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, pages 1–5. IEEE, 2011.
40. M. Saari, A. Muzaffar bin Baharudin, P. Sillberg, S. Hyrynsalmi, and W. Yan. LoRa — A survey of recent research trends. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0872–0877. IEEE, 2018.