Wireless Sensor-based Agricultural Monitoring System

ALEXANDROS ZOGRAFOS

KTH Information and Communication Technology

Degree project in
Communication Systems
Second level, 30.0 HEC
Stockholm, Sweden

Wireless Sensor-based Agricultural Monitoring System

Alexandros Zografos

2014-03-25

Master's Thesis

Examiner and academic adviser
Professor Gerald Q. Maguire Jr.

School of Information and Communication Technology (ICT)
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract

Today energy resources are becoming scarcer and therefore more valuable. In conjunction with the population growth over last century, the need for finding new, more efficient, and sustainable methods of agricultural cultivation and food production has become more critical. To facilitate this process, we are designing, building, and evaluating a system for precision agriculture which provides farmers with useful data about the soil, the water supply, and the general condition of their fields in a user friendly, easily accessible manner. Our system aims to make cultivation and irrigation more efficient as the farmer is able to make better informed decisions and thus save time and resources.

The diversity of location and climatic effects upon agricultural cultivation, along with other environmental parameters over time makes the farmer's decision-making process more complicated and requires additional empirical knowledge. Applying wireless sensor networks for monitoring environmental parameters and combining this information with a user-customized web service may enable farmers to exploit their knowledge in an efficient way in order to extract the best results from their agricultural cultivation. The system can scale based on each farmer's demands and the resulting ensemble of collected information may represent a valuable resource for future use, in addition to its use for real-time decision making. The design of the precision agriculture system contains a prototype solution regarding the sensor platform and a customizable service that can be utilized in different ways and by several entities.

Keywords: IEEE 802.15.4, precision agriculture, wireless sensor networks, ZigBee

Sammanfattning

Idag när energiresurser blir allt knappare och knappare blir de även mer värdefulla. I samband med befolkningstillväxten under förra århundradet har behovet av att hitta nya, mer effektiva och hållbara metoder inom jordbruket och livsmedelsproduktion blivit av allt större vikt. . För att underlätta denna process har vi designat, byggt och utvärderat ett system för precisionsjordbruk som ger bönder mer användbara data om jorden, vattenförsörjning och det allmänna tillståndet i sina områden på ett användarvänligt och lättillgängligt sätt. Vårt system syftar till att göra odling och bevattning effektivare då bonden kan fatta bättre underbyggda beslut och därmed spara tid och resurser.

Mångfalden av läget och jordbrukets klimatpåverkan, tillsammans med andra miljöparametrar över tiden gör bondens beslutsprocess mer komplicerad än tidigare och kräver ytterligare empirisk kunskap. Att tillämpa trådlösa sensornätverk för övervakning av dessa parametrar och att presentera? denna information med en användarvänlig skräddarsydd webbtjänst kan göra det möjligt för jordbrukare att utnyttja på ett effektivt sätt nåde bästa resultaten från sitt jordbruk. Systemet kan skala utifrån varje bondes krav och den insamlade data kan utgöra en värdefull resurs för ett framtida jordbruk, utöver dess användning för dagens bondes beslut. Utformningen av systemet för precisionsjordbruk innehåller en prototyplösning avseende sensorplattformen och en anpassningsbar tjänst som kan användas på olika sätt och av flera enheter.

Nyckelord: IEEE 802.15.4, precisionsjordbruk, trådlösa sensornätverk, ZigBee

Acknowledgements

I would like to express my deepest appreciation to all those who provided me the possibility to complete this thesis. A special gratitude I give to my academic advisor and examiner at KTH, Prof. Gerald Q. "Chip" Maguire Jr. whose contribution in stimulating suggestions and encouragement helped me to coordinate my project especially in designing and writing this thesis. His ideas, experience and advice led me to extract the optimal results of the project and helped me to become a better engineer in general.

Furthermore, I would like to acknowledge with much appreciation the crucial role of the Ericsson team I worked within and gave me the permission to use the required equipment and materials to complete my tasks. A special thanks goes to my supervisor Maxim Teslenko who-with patience-helped me to understand and follow parts of the project that were completely new to me and provided to me knowledge that I was lacking.

Last but not least, many thanks go to Athanasios Karapantelakis who stood by me during the whole time of the project and helped me to develop my skills by investing his full effort in guiding me through the process and trying to make me better engineer. His help was meaningful and made me a better public presenter and a more communicative person in working environments while boosting my confidence and my analytical skills.

I would like to express my special gratitude to the whole Global Services Research team in Ericsson for the great opportunity and the time they offered me. They made me feel as a part of the team and listened to my opinions and ideas and we all cooperated in a very effective way. This thesis includes work that was carried out by the team and I tried to combine and present the outcome in a way that would be considered as optimal.

Special thanks to my family, in particular my parents and my brother for their unconditional affection, moral support and lovely inspirations during the period of my study, and all through my life. I would not be the person who I am now without their support.

Table of contents

List of Figures

List of Tables

List of acronyms and abbreviations

6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
AMEE	Avoiding Mass Extinctions Engine
AMON	AMEE Monitoring Object Notation
CoAP	Constrained Application Protocol
DC	Direct current
ER	Edge Router
FFD	Full-function Devices
GML	Geographic Markup Language
GND	Ground
GPIO	General-purpose input/output
HTTP	Hypertext Transfer Protocol
ID	identifier
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
ISP	In-System Programming
JSON	Javascript Object Notation
LCD	Liquid Crystal Display
LED	Light-emitting diode
MAC	Media access control
M2M	Machine to machine
OS	Operating System
OSI	Open Systems Interconnection
PAN	Personal Area Network
PCB	Printed Circuit Board
REST	Representational State Transfer
RFD	Reduced-function Devices
RGB	Red-Green-Blue
RPL	Routing Protocol for Low-Power and Lossy Networks
SensorML	Sensor Model Language
SICS	Swedish Institute of Computer Science
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TI	Texas Instruments
TTL	Transistor–transistor logic
UART	Universal serial receiver/transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus

VIN	Input Voltage
WSN	Wireless Sensor Network
XML	Extensible Markup Language

1 Introduction

This chapter introduces the general area of research and describes the purpose of this Master's thesis project. The introduction to the scientific area is followed by a description of the problems that set the goals for this project. The chapter ends with a description of the structure of this thesis.

1.1 General introduction to the area

The contemporary world is in a transition stage where problems concerning global issues, such as global warming and alternative energy sources, are combined with new challenges demanding immediate solutions. Society's focus has shifted from economic growth to sustainable development, where environmental, social, and economic aspects are considered together, rather than separately. Policies that promote sustainability in all sectors of the economy (manufacturing, agriculture, and services) are now considered as a part of good governance. Problems such as climate change, population growth, and poverty (especially hunger), occur in a context of a gradual depletion of natural resources and the fear of diminishing coal energy reserves. These are some of the global issues that are thought to require multidisciplinary approaches in order to be addressed successfully.

In this Master's project we focus on agricultural production and cultivation. This overall process has a significant role in fulfilling the basic human need for food. The production, preparation, packaging, distribution, etc. of food also generates a lot of income. The aim of this Master's thesis project is to exploit modern technologies and tools to improve monitoring and management of crops, in order to improve the efficiency and sustainability of farming and food production. To this end, we have designed a system for precision agriculture, which relies on a wireless sensor network combined with a service to provide individual farmers with access to data that they find useful. The system utilizes wireless sensor nodes that collect and transmit data about the quality of the water supply, the soil, and other parameters in an agricultural field. While such sensor-based systems have been investigated earlier, one of the key innovations to be explored in this Master's thesis project is the combination of these sensors systems with a service-driven business model to increase their ease of use and to amplify the gains that can be realized via an *integrated* system. The goal is to give a farmer a more complete picture of the current and historic crop status in order to foster better informed decision making. It is expected that such decisions will benefit *both* farming and irrigation by saving time and resources.

Factors such as the diversity of conditions which vary depending on location (for example weather, presence of insects, and disease) combined with the inability to predict the future characteristics of the environment during the different seasons over time complicate the decision making process and require specialized knowledge. This project is an attempt to bring some of these micro-environmental sources of information into the decision making process of farmers.

1.2 Problem definition

The process of utilizing technology in farming and cultivation requires deep knowledge of agricultural processes, biology, chemistry, and empirical knowledge. There are many parameters which must be taken into consideration and investigated in depth when designing a system that should improve cultivation procedures by making the whole process more effective and sustainable.

In order to design and build a precision agriculture system that can be widely used by many users and applied in different contexts, many questions need to be addressed. Some of these questions are:

- Is it feasible to design a system that will accommodate every possible scenario in an agricultural context and do so for all possible users?
- Is automation in agriculture really useful and in what part or parts of the cultivation process (e.g. seed planting, growing, harvesting, selling) can it be applied?
- What is the cost of the cultivation process and how can this cost be reduced by automating one or more parts of this process?

- What is the most costly component of this process that could be reduced? How and how much could this cost be reduced?
- Are geographic parameters such as location, altitude, solar exposure, ground and air moisture, ground and air temperature, mineral content of the soil, the (micro-) climate, or the season, sufficient to make a significant difference in the way that a crop is cultivated?
- What are the sensitivities of the crop that should be taken care of when cultivating?
- What types of plants are to be planted and how long will this crop be planted in this location? What is the planned rotation of crops? What are the plans for applying fertilizer to this location? What is the level of the farmer's empirical knowledge?
- Are there any abnormalities regarding the location, season of the year, previous crops in a specific field, or a combination of all these aspects which need to be considered as part of an informed decision making procedure by the farmer?

Today, these open questions cannot be answered with confidence even by experts. Agricultural science is a multidisciplinary field and all of the above aspects need to be taken into account when making decisions about cultivation of a field on a farm. Furthermore, research in agricultural science is strongly related to local areas. Climate and soil properties vary from one place to another and from time to time. Climate change and transformation of the plants and soil occur as time passes, thus making successful and sustainable cultivation a tough process for someone who does not know the specific aspects of the locality and how the process needs to evolve over time in this specific geographical and microclimatic area.

1.3 Goals

Considering the above problems, designing a system to improve the state of agriculture that can be used in multiple contexts is a challenging task and is too complex of a problem to address in such a broad perspective. However, it may be possible to develop a working solution that can be applied *in specific settings*. This thesis describes a monitoring system that collects data using a wireless sensor network, and then relays this data through a gateway to a (cloud based) server. At the server the data are stored and analyzed in order to provide the user[*] with useful statistics and alerts as input to this user's decision making process. As mentioned above, experience and knowledge from experts should be utilized when designing the monitoring process. This knowledge is valuable as it would support the decision making process for the farmer. Ideally, the system's configuration should serve a specific group of farmers (i.e. fields and crop specific) and thus, be more effective. The system should allow the users themselves to select what types of sensors will be used and to specify the expected range of measured values from these sensors. It is important that the system allows the user to configure the range of the expected values, thus the user can configure a system to ensure appropriate inputs to the decision making process. The system should be capable of supporting different types of sensors (for example sensors measuring temperature, humidity, light, electric conductivity of water, etc.) and these measurements should be correlated with the help of agricultural experts to produce meaningful results. In the solution proposed in this thesis, users will use a web interface to login and configure the system's parameters; hence they can generate their custom selection of alerts[†]. This web interface will also provide access to statistical data calculated from the dataset of values previously collected by the sensors.

The system should be scalable, to accommodate large amounts of data from different fields and sensor types, large numbers of users, and to utilize technologies based on data mining and machine learning to extract patterns that can be utilized to make more informed decisions in the future. Users will be able to access the sensor data in order to manually or programmatically find similarities and exploit differences in the data mining process and the resulting crop yields in order to develop and adopt new ways of cultivation (both for themselves and potentially conveying this knowledge to others so that they too might improve their cultivation). The knowledge from the aggregated data collected

[*] We will assume in this thesis that the user of this data is a farmer.
[†] We define an "alert" to be any deviation of sensor values outside of the range of expected values – as configured by the user.

from sensors, together with expert feedback can be used to maximize the value of the data collected using sensor measurements.

1.4 Methodology

This research will follow the *inductive paradigm* [23]. The goal is to investigate the previously implemented systems and to find the most suitable technologies that can be applied to focus our research and to build a suitable and valuable system. Since it is not possible to make a hypothesis from the beginning and then justify this hypothesis at the end, the *deductive method* is not applicable.

Since this study will examine wireless sensor network architectures and applications in the agricultural sector - a qualitative method will be used [23]. This method will give us a better understanding of why and how the process should be designed. More specifically, the work can be split into the four following parts:

1. Literature study and design & conduct a survey,
2. Design of a prototype solution,
3. Implementation of the prototype, and
4. Evaluation of the resulting prototype.

The literature study provides the background information that is necessary for understanding the feasibility of the design and the previously implemented solutions. The survey utilizes questionnaires and interviews with farmers and agricultural scientists to understand the actual needs and problems. The design of the prototype is based on the prior research and the knowledge gained by interviews, thus giving an optimal architecture and evaluating the best-suited protocols for this application. The implementation follows the rules and guidelines specified during the design. The result of this design should be a prototype that could be adapted to address potential changes in requirements due to the addition of new requirements or deeper interpretation of the needs that arise. Regarding the evaluation of the system, a testing procedure should consider the end-to-end performance in order to find out if there are any problems concerning the system's functionality or robustness. Additionally, a test case scenario should be utilized to attract people whom are interested in or worked with similar applications in order to solicit ideas for future work or to further develop and evaluate the proposed system.

1.5 Structure of the thesis

Following this generic introduction in this chapter, Chapter 2 describes the background of this project and to lay the foundation required to understand the rest of this thesis. Chapter 3 describes the implementation and Chapter 4 describes the analysis of the design, the web interface, and the prototype. The thesis concludes in Chapter 5 with some conclusions, some suggestions for future work, and some reflections on social, economic, and ethical issues that are relevant to this thesis project.

2 Background

The purpose of this chapter is to introduce to the reader the technologies that are examined and used in this project. Furthermore, some existing implementations and related projects are presented as they were found interesting and helpful during the design process leading to the system proposed in this thesis.

2.1 Precision Agriculture

The term “precision agriculture” refers to a contemporary approach of applying new technologies utilizing sensors in order to optimize the agricultural cultivation processes. Precision agriculture-based system design principles are increasingly used in research projects and commercial products to provide solutions for crop status monitoring, water supply regulation for irrigation, fertilizer management, pest control, and automated harvesting. Such systems benefit the complete process by reducing cost through automation and saving time. Another significant feature that precision agriculture provides to farmers is the ability to prevent hazardous incidents and to proactively monitor their crops and the local environmental conditions. The effectiveness of precision agriculture is based on the analysis of accurate sets of measurements in soft real-time. Parameters such as the soil condition and humidity are aggregated and analyzed, in order to extract useful information that a farmer can use as a recommendation or guidance; or even to apply fully automated procedures to the crop cultivation process chain.[1]

2.2 Wireless sensor platforms

There are currently lots of sensor platforms and every day, new projects appear that introduce additional new sensor platforms. The variety and rapid evolution of sensor platforms makes the choice of an appropriate platform a complicated process. There is a wide range of solutions that cover all the needs of the potential users. Some of these solutions that may serve the purpose of our project are described in the following subsections.

2.2.1 Herjulf Wireless Multi-Function custom platform

The Herjulf Wireless Multi-Function custom platform [2] is a custom printed circuit board (PCB) that was designed in order to provide an easy-to-use platform. The microcontroller is an Atmel ATMega128rfa1. This microcontroller is widely used and several operating systems, including Contiki (see section 2.3.3) and TinyOS[*], have been ported to it. There is an integrated IEEE 802.15.4 transceiver on the board along with a high-performance built-in antenna that can achieve a range of 300 m in line-of-sight conditions when the output power is 3mW. The hardware specifications are [2]:

- Low-power voltage regulator for wide input voltage range
- Direct current (DC) power input (bypassing the voltage regulator) at a maximum of 3.6 V
- Hi-resolution 12-bit temperature sensor (which also provides a unique 64 bit ID)
- 2 analog inputs
- 1 analog input to voltage regulator
- 1 general purpose input/output (GPIO)/pulse pin with jumper selectable pull-up
- 1 programmable power pin V_{cc}
- 1 GPIO with connector for 1-wire bus
- 2 (Red/Yellow) light emitting diodes (LED) for monitoring and debugging
- Programmable via AVR 6-pin In-System Programming (ISP) or via Universal Serial Bus (USB)/serial bootloader
- Optional on-board humidity sensor
- Optional on-board comparator, typically a 0805[†] or 1206[*] phototransistor

[*] www.tinyos.net
[†] Equivalent to a 2012 metric sized part (i.e., 2.0 mm × 1.25 mm package)

Regarding the power options of the platform, it can utilize either external power sources or batteries. The external sources can be:

- USB power by using the 6-pin header with a USB-TTL cable
- Regulated input power ranging from 3.5-18 Volts DC (The regulated input power can be used with a 6/12 Volt lead acid or 4.2 V lithium-ion battery.)

Through the direct current power input, using the V_{cc} and ground (GND) pins different kinds of batteries can be used with the requirement that the voltage should not be more than 3.6 Volts in order not to avoid problems for the PCB. Coin Cell batteries, AA, AAA, and different types of batteries, such as alkaline, lithium, or NiMH are suitable solutions.

2.2.2 Texas Instruments' 6LoWPAN Sub1GHz Evaluation kit

The Texas Instruments (TI) 6LoWPAN Sub1 GHz Evaluation kit [3] can be used for wireless sensor networks utilizing the 868/915 MHz band and consists of a complete set of different types of hardware. There is a 6LoWPAN compatible Edge Router (ER) based on TI's OMAP-L138 processor and CC1180EM for wireless communication. The ER's operating system is Sensinode Ltd.'s NanoRouter™ 2.0 and the sensor nodes run Sensinode Ltd.'s NanoRouter™ 2.0 lite[†].

In the kit there are four additional boards: two CC1180DB and two EM430F5137RF900. These boards are used as sensor devices. A user application runs on the ER. Communication with the nodes occurs through a Wireless Network Processor (WNP) which is connected via universal asynchronous receiver/transmitter (UART) with the host processor. The Sensinode Nodeview 2.0 software is used to control the network and to test the devices.

Some of the main features of this system are:

- Low memory footprint, small stack size;
- The data rates that can be achieved are 50, 100, 150, and 200kbits/s;
- Neighbor Discovery with ICMPv6; and
- Peer-to-peer communication and self-healing mesh networking.

This evaluation kit is a very convenient and useful platform since it can provides a complete wireless sensor network. With the addition of proprietary software the user can have a web based application for visual interaction with the sensors.

2.2.3 Freescale's Freedom Development Platform: FRDM-KL25Z

The Freescale Freedom Development Platform FRDM-KL25Z [4] is a development platform with a simple but sophisticated design. It is intended to provide rapid prototyping for microcontroller-based applications. It is a low-cost solution which embeds a lot of useful features. The microcontroller is based on the ARM Cortex-M0+ core, specifically the Kinetis L Series KL1x (KL14/15) and KL2x (KL24/25). The maximum operating frequency is 48 MHz and the flash memory has a size of 128 KB. There is also a full speed USB controller and analog and digital peripherals. There is an RGB LED, capacitive touch slider, and a 3-axis accelerometer. The pin layout of the board is compatible with the Arduino's layout, thus offering compatibility with many expansion boards.

The board can be powered from several sources, either using regulated inputs or directly. The USB and V_{IN} input power sources are regulated on board with a 3.3 V linear regulator. There is also the option of using a coin cell battery or the 3.3 V pin (with the requirement that the input voltage range is 1.71-3.6 V in order for the board to function properly).

2.3 Previous work in wireless sensor networking

This section is a background study of previous systems and tools which can be used for building a precision agriculture system operating in a resource constrained environment. Specifically, this section

---

[*] Equivalent to a 3216 metric sized part (i.e., 3.2 mm × 1.6 mm package)

[†] Further details of these devices can be found at http://www.sensinode.com/EN/products/software.html .

describes previous research work on precision agriculture systems (sections 2.3.1 and 2.3.2), software supporting precision agriculture systems operating in constrained environments (section 2.3.3), application-layer network protocols used for modeling the sensor data (section 2.3.4), and a network protocol stack specifically designed to work in resource-constrained environments - thus perfectly matching requirements for our system (section 2.3.5).

2.3.1 A wireless low-cost irrigation system using ZigBee Technology

The system described by Zhou, et al. [5] consists of a portable controller, a wireless sensor node, a weather station, and several wireless actuators. The sensor node collects temperature and air humidity parameters in its area-of-coverage (referred to as a “section”). A nearby weather station monitors the meteorological information (such as rainfall). All the sensory data are wirelessly sent to the portable controller. The portable controller transmits control commands wirelessly to the actuator nodes. The actuator nodes are used to control a pump and electromagnetic valves. Both sensor nodes and actuator nodes serve as end devices in a wireless sensor network (WSN). This WSN is based on a star network topology. The portable controller acts as a ZigBee coordinator to build and maintain the wireless sensor network. When the coordinator starts, it creates a personal area network (PAN) and allows the end devices (sensor and actuator nodes) to join the network. After the WSN forms, the portable controller receives sensor data and displays them on a liquid crystal display (LCD). The portable controller also manages the irrigation system. Because the ZigBee WSN is self-forming and self-healing, this *ad hoc* irrigation system allows user to dynamically increase or decrease the number of end devices (sensor nodes and actuator nodes) to meet their current requirements. If routers are used, then a mesh irrigation network connecting several sections can be configured for remote intelligent irrigation. Expert irrigation strategies can be applied by connecting the portable controller to a computer through a serial interface. Manual operation is also possible by the users pressing buttons on the portable controller.

2.3.2 Energy efficient data transmission in automatic irrigation system using wireless sensor networks

The system described by Sudha, Valarmathi, and Babu [6] utilizes a number of nodes to automate and optimize the irrigation process for multiple fields. Each node has a unique address that is assigned by the base station. Each node consists of a temperature sensor and a moisture sensor that measures the soil moisture. The need for irrigation is decided by a microcontroller based upon on the measured temperature and moisture values. Irrigation is regulated by using solenoid valves to control the mix of water and fertilizer and to control a pump.

The wireless link protocol is a time division multiple access (TDMA)-based scheme. This scheme seeks to minimize the energy consumption of each node while transmitting data from the nodes. A base station assigns an address to each node when the node is first powered on. After receiving an address the node switches to an idle state during which it can only receive frames from the base station. In this mode, the nodes require low energy because the TDMA scheme provides a collision-free communication environment and thus there is no need for content for access to the channel. When the base station broadcasts an address to the network, the node with the corresponding address replies with its sensed results and then the node returns to idle mode. This design offers robustness since if a node fails, then only the results of this node’s area are not transmitted.

Apart from this direct technique, the sensors are divided into clusters and the head of this cluster is responsible for accumulating and transmitting data. The choice of the cluster head is based on the remaining energy level of the nodes. The node with the highest energy level, becomes head since the transmission of the data require more energy than being in idle mode and just receiving. This method is also based on a TDMA scheme that offers collision-free communication and low energy consumption.

2.3.3 Contiki Operating System

Contiki[*] is an open source operating system (OS). There is an active community that continues to develop Contiki. The latest version is 2.6. Contiki runs on a range of platforms (including both TI and Atmel microcontrollers)[7].

Contiki supports IP connectivity. Contiki OS offers out-of the box support for TCP/IP and UDP/IP protocol stacks and can also supports application layer protocols, such as the Hypertext Transfer Protocol (HTTP) and the HTTP-derived Constrained Application Protocol (CoAP). COAP is designed to be used in resource-constrained environments. Contiki supports both IP version 4 (IPv4) and IP version 6 (IPv6). Furthermore, Contiki provides a 6LowPAN implementation to support transmission of IPv6 packets over IEEE 802.15.4 networks. As the memory of most sensor systems is small, some platforms can encounter problems when using Contiki's full IPv6 stack [8].

2.3.4 Models for sensor data transmission and storage

This section presents a number of protocols proposed for modeling sensor data and efficiently transmitting this data for processing in resource constrained environments.

2.3.4.1 Sensor Model Language (SensorML)

The OpenGIS Sensor Model Language Encoding Standard (SensorML) is an Extensible Markup Language (XML) representation of data provided by sensors. SensorML is used to facilitate the storage and sending/receiving of sensor data. Different types of sensors can be supported by the SensorML format, ranging from simple types of data (such as temperature) to more complex types of data. The encoding of the components and the processes of the SensorML data forms a Geographic Markup Language (GML) format using its own schema [9].

Some of the purposes that SensorML could be suitable for are:

- Ease of processing and analysis of measured sensor data,
- Correlation of measured data and the corresponding geolocation, and
- Compatibility of data format and accessibility.

SensorML is an extensible format since it is based on XML. For this reason SensorML can be used as a framework for any process regarding sensor data, storage, processing, or using this data in different components of systems that require communication and transmission of the data [10].

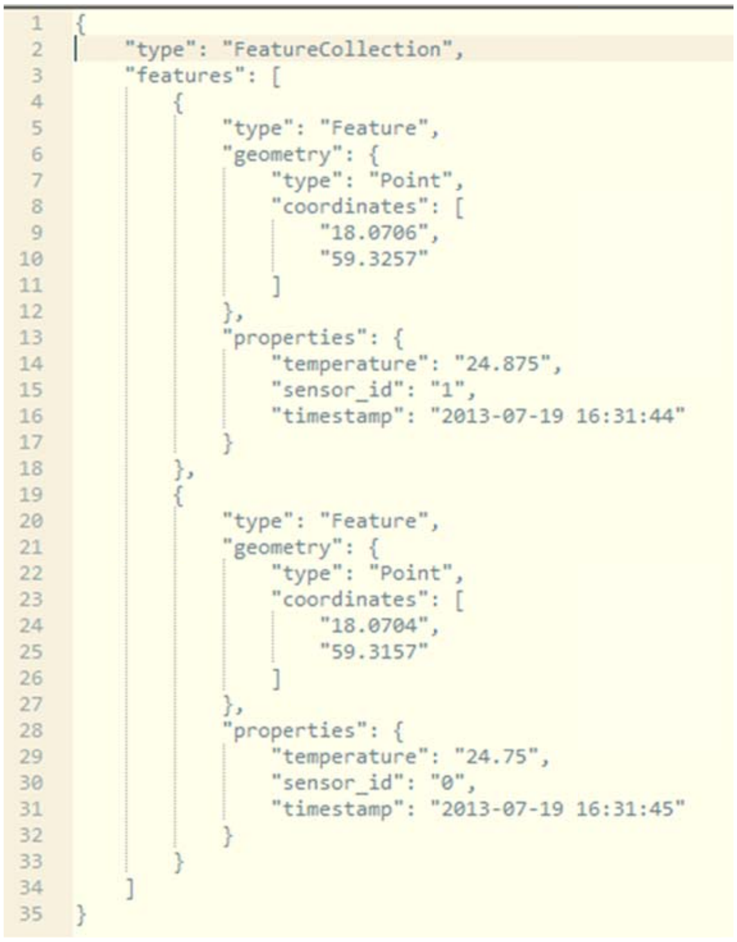
2.3.4.2 GeoJSON / AMON

GeoJSON is a structured format for encoding geographical data. It is a specific type of Javascript Object Notation (JSON). A variety of usage scenarios can be supported by the different types of data that GeoJSON supports. According to the GeoJSON specification, objects forming geometrical structures can be grouped under a FeatureCollection (for example Points, Polygons, Multipolygons, Linestring, MultiLineString, MultiPoints, and GeometryCollection). Every complete GeoJSON structure is a valid JSON object and contains the description of elements. Every FeatureCollection contains arrays of features that describe the corresponding element in which the type, the name, and some properties are the main parts of the structure [11].

The GeoJSON format is based on JSON and therefore is characterized modern and better suited for transferring data than XML models due to the smaller overhead that is required (i.e., the encoding is less verbose). Another advantage of GeoJSON when it comes to usage with object-oriented programming languages is that it stores the data in JSON in arrays and records, while XML stores the data in trees [35].

Figure 2-1 illustrates a typical example of the use of JSON data, to describe points on a map along with some properties for each point, in this case a temperature, sensor_id, and timestamp.

[*] http://www.contiki-os.org/



```
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [
                    "18.0706",
                    "59.3257"
                ]
            },
            "properties": {
                "temperature": "24.875",
                "sensor_id": "1",
                "timestamp": "2013-07-19 16:31:44"
            }
        },
        {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [
                    "18.0704",
                    "59.3157"
                ]
            },
            "properties": {
                "temperature": "24.75",
                "sensor_id": "0",
                "timestamp": "2013-07-19 16:31:45"
            }
        }
    ]
}
```

Figure 2-1: GeoJSON example

Avoiding Mass Extinctions Engine (AMEE)* Monitoring Object Notation (AMON) [13] is an open-source, JSON-based data format alternative to GeoJSON. Unlike GeoJSON, AMON is meant to model a wide range of data from metering/monitoring devices. This means that it is an extensible format and its open source nature allows contribution for addition or modification of the standard core [12].

2.3.5 Protocols for WSN

In the following paragraphs, the most common protocols for communication between sensor devices are presented. The protocols are described and compared in order to find the protocol most suitable for the prototype system to be realized in this thesis project.

2.3.5.1 CoAP (RESTful communication)

Constrained Application Protocol (COAP) is designed for devices operating in constrained environments, i.e., where resources are limited. CoAP is a RESTful protocol [45] specialized for machine to machine (M2M) applications and works in both synchronous and asynchronous modes, depending on the architecture.

COAP is datagram protocol utilizing UDP and has optional support for reliability. This reliability can be achieved by adding an option which marks the CoAP header as “Confirmable” and thus requests that a positive acknowledgement be sent by the receiver. The simplicity of COAP results in low overhead and low complexity parsing procedures [14,15].

* https://www.amee.com/

2.3.5.2 6LoWPAN

Low-power wireless personal area networks (LoWPANs) are networks that consist of low power, short range, low bitrate, and low cost devices (such as wireless sensors). These devices typically have limited memory and computational resources. Such devices can be utilized in networks where power is limited and high throughput is not the main objective. The general characteristics of the network regarding the lower layers of the network stack are based on the IEEE 802.15.4 standards. The physical layer of IEEE 802.15.4 uses frequencies of 868 MHz, 915 MHz, and 2.4 GHz with data rates of 20 kbps, 40 kbps, and 250 kbps respectively. In general, the devices are powered by a battery since the power requirements are low and usually the devices spend most of their time in sleep mode in order to reduce their energy consumption to low levels. During these sleep periods, the sensors node cannot be accessed. Other reasons that may prevent access to a sensor include that the sensor's exact physical location is frequently unknown, low battery power level, low signal strength, or physical tampering with the device. The low power and low cost of these devices and networks potentially enables very large numbers of sensors to be deployed in the future.

Taking all of these limitations into consideration, a self-organizing network infrastructure is needed. Since the deployment of such networks in the future may involve very large numbers of devices the size of the address space should be correspondingly large. Additionally, the sensors may be connected directly to the internet and other systems, thus making IP addressing an ideal solution. The approach that combines all the above characteristics and covers the needs is the IPv6, a solution that has great potential to be the dominant player in LoWPANs in the future, enabling the dream of an internet-of-things (IoT) [16].

2.3.5.3 IEEE 802.15.4

The main use of IEEE 802.15.4 *(Media Access Control Layer and Physical Layer)* [17,18] is for communication networks with low power and low throughput requirements. The main objectives of the IEEE 802.15.4 protocol are reliability of transferring data, simplicity of the installation process, reasonable battery life, low cost, and the flexibility of the protocol. Several network topologies (star, mesh, or tree) can be used. The more complex topologies, such as a mesh topology, require more sophisticated node functionality. For this reason, devices that comply with IEEE 802.15.4 can be divided into two categories: full-function devices (FFD) and reduced-function devices (RFD). RFDs are used in applications with simple topologies and requirements and can only associate with one FFD. RFDs can only be leaves of a tree since they cannot associate with a device other than the network coordinator. FFDs can work as coordinators in networks and can be associated with more than one RFD and FFDs. FDDs provide synchronization to the other devices of the network.

Figure 2-2 illustrates the protocol stack of the IEEE 802.15.4 standard. This protocol stack is based on the OSI seven layer model and it corresponds to the bottom two layers: the physical (PHY) and the medium access control (MAC) (the lower part of the OSI link layer).

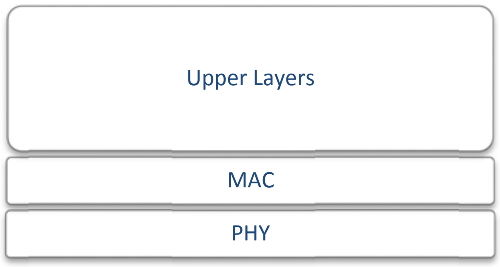


Figure 2-2: IEEE 802.15.4 Stack

The physical layer can utilize one of 27 different channels depending on the location of the device, according to the following categorization:

- 1 channel (Europe): 868.0 - 868.6 MHz
- 10 channels (USA): 902.0-928.0 MHz
- 16 channels (Worldwide): 2.40-2.48 GHz

2.3.5.4 Zigbee

Zigbee® is a proprietary protocol stack that provides a network layer and the framework for an application layer over the PHY and MAC layers of IEEE 802.15.4. As described in the previous subsection, in IEEE 802.15.4 networks at least one FFD is required to act as the network coordinator. The RFDs are generally battery-powered, while the FFDs are usually line-powered. RFDs search for available networks, transmit their data according to their applications, receive data from the network coordinator, and enter sleep mode for lower power consumption. RFDs can only talk to FFDs and more specifically, only the one FFD to which they have associated as their network coordinator. The FFDs can serve as network coordinators, routers, or end devices; with the Zigbee terms for these types of devices: ZigBee coordinator, ZigBee router, and ZigBee end device (respectively).

The ZigBee network layer supports star, tree, and mesh topologies. In a star topology, the ZigBee coordinator initiates and maintains the network, and all the end devices communicate directly with this ZigBee coordinator. In mesh and tree topologies, the ZigBee coordinator is responsible for creating the network and for choosing certain key network parameters, but ZigBee routers can be used to extend the network. In tree networks, Zigbee routers transfer data and control messages over the network using hierarchical routing. [19]

2.3.5.5 Comparison of 6LoWPAN, IEEE 802.15.4, and ZigBee

Table 2-1 presents a comparison between the above protocols. Note that all three protocols can operate in the same frequency bands and that all three can support 128 bit AES encryption.

Table 2-1: Protocol comparison (adapted from the comparison on the Dresden Elektronik (ZigBee Alliance member) website [20])

Feature	IEEE 802.15.4 MAC	6LoWPAN	ZigBee Pro
Possible network topologies	Star	Tree, mesh	Mesh
Multi-hop capable	No	Yes	Yes
Over-The-Air-Update	No	No	in ZigBee Smart Energy standardized
IPv6 compatible	No	Yes	No
Automated channel change	No	No	Yes
Maximal number of nodes	Implementation dependent	> 64000	> 64000 nodes per network
Stand-by modi	Have to be implemented	Available	depending on implementation
End-to-end encryption	None	IPSec scheduled	Application link keys
Preferred field of application	Point-to-point connections, networks with small number of nodes	WPANs with IP connectivity	Smart energy und home automation applications

2.3.6 Telecombretagne's IPv6 stack

Telecombretagne has implemented two IPv6 stacks for Arduino. Both of these are based upon the Contiki network stack, but one of them (*pico IPv6)* is even further optimized for small size to fit on the Arduino Uno. The details of each of these stacks are given in the following paragraphs.

2.3.6.1 Arduino µIPv6 Stack

Arduino[*] is a platform for embedded hardware and software prototyping. An IPv6 stack for Arduinos seems to be necessary given a transition to the IoT with Ericsson envisioning 50 billion connected devices by the year 2020[36].

The Arduino µIPv6 stack is an open source implementation of the IPv6 stack for the Arduino MEGA and Xbee, based on the Contiki network stack. This stack implements the low-power protocols 6LoWPAN, RPL, and CoAP. This stack can be freely used for both commercial and non-commercial use. The current interface that is provided is the XBee Series 1 and can be used in combination with IEEE 802.15.4, IEEE 802.3 (Ethernet), and IEEE 802.11 (Wi-Fi) MAC Layers using the appropriate shields or interfaces [21]. The µIPv6stack is organized as shown in Figure 2-3.

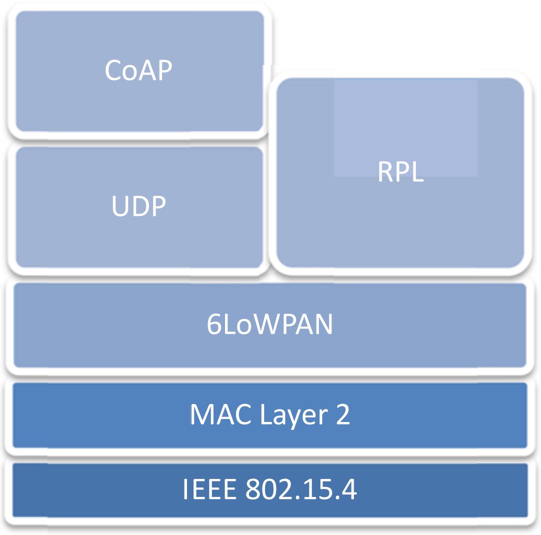


Figure 2-3: Arduino µIPv6 Stack

2.3.6.2 Arduino pico IPv6 stack

In order to use the IPv6 stack on Arduino UNO which only has 2kB of RAM, a different version of the stack was implemented. The pico IPv6 stack (pIPv6) is a more lightweight version of the µIPv6 stack. This lighter-weight stack has reduced RPL and CoAP functionalities in order to fit into the available resources of the Arduino UNO. Additionally, this stack is compatible with all Arduino boards and can run on top of the same lower layer protocols as µIPv6. This new pIPv6 stack is open source and the source code can be found on github. The stack is further described in [22]. The stack is organized as shown in Figure 2-4.

[*] www.arduino.cc

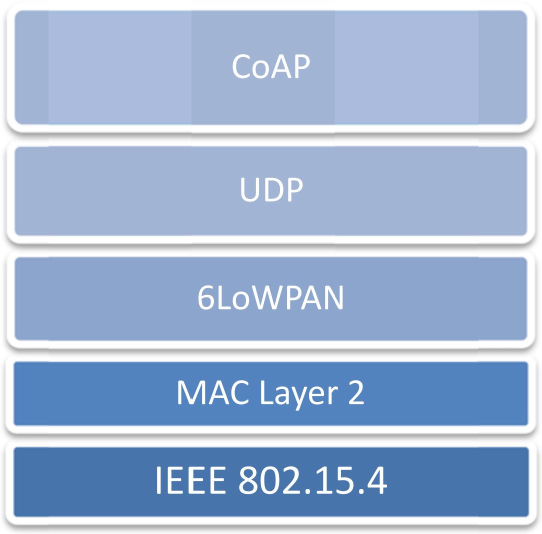


Figure 2-4: Arduino pIPv6 Stack

2.3.7 Overview of technologies for resource constrained environments

In this section we have presented technologies (network protocols, data representation formats, and relevant prior systems) used for monitoring and measuring specific values in an environment, such as the agriculture monitoring prototype system we propose to develop. The knowledge of existing technologies presented in this section, is meant to function as reference for choosing the appropriate subset to meet the requirements set for our prototype system (see Chapter 3).

3 Implementation

In this chapter we describe the technologies, the equipment, and the architecture that have been used to develop our model system. The implementation of the designed system is described. Additionally, the development process is described with details of the form of the different versions of the prototype that were created along with the changes that took place during the development process.

In the scope of both the desire to realize a prototype for a system to support precision agriculture and driven by visions of the internet of things, a wireless sensor-based monitoring system was designed and built. This system was designed based upon consideration of a number of fundamental constraints regarding the resource constrained environments that it would likely be applied in. Furthermore, such a system should be developed while carefully considering some fundamental design constraints:

- Cost-effective scalability such that the system could be applied in small private backyards, fields, or even on large industrial farms.
- IPv6 network addressing for the sensors that will provide a future-proof architecture with more than 50 billion devices connected by 2020 [36].
- Adaptability on the support of different sensors because of using the 1-wire protocol.
- Modularity and use of standardized network protocols allows the use of third-party sensors.
- End-to-end data security.
- Use of existing practices and equipment from earlier projects within Ericsson [24].

At this point it should be mentioned that the work that is presented in this Master's Thesis is a result of cooperation within the team in Ericsson Global Services Research. Some parts are referenced and the corresponding work is described in this document.

3.1 Architecture Overview

Figure 3-1 presents the architecture of the proposed system.

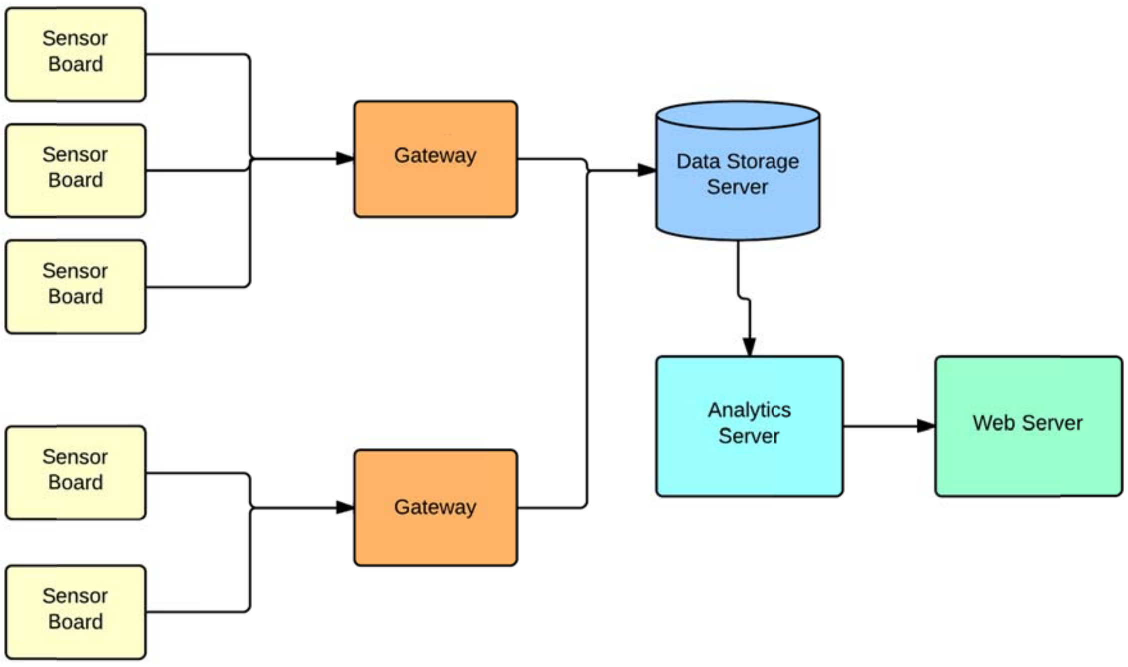


Figure 3-1: Architecture diagram

Each sensor board senses and collects data from the environment using the attached sensors and then transmits these data to a gateway. Each gateway receives data from the sensor boards that are within range. Multiple gateways can be used in order to cover an area; whether it is a small private farm or a large commercial farm. Each user can use their own gateways or share gateways that cover the desired area. Each gateway relays the data that it receives to a data storage server where these data are stored into a data base. The stored data are subsequently used by an analytics server which analyzes this data in order to present it via a web server as part of a webpage.

There is scalability with respect to the number of the sensor boards that can be used since each board has an IPv6 address. In order to cover large areas, due to range limitations, multiple gateways will need to be used. While for small neighboring fields a single gateway might have sufficient range to receive the sensed data. Note that each gateway has an IPv6 address for both of its interfaces*. On the server side, the data storage server and the analytics server could share the same hardware for lower cost. In order to increase the flexibility and capacity when demands increases, a cloud-based solution could be used to store and process in increased amount of data.

In the following sections of this chapter, each component of the aforementioned system is described in detail.

3.2 Sensor Boards

Our system utilizes sensor boards which consist of a microcontroller connected to one or more sensors, along with a ZigBee transceiver, transmitting data which are then retrieved at a gateway. This microcontroller reads data from the sensors and transmits the sensor's output to the gateway, running our own custom software. This software was designed and implemented taking into account resource utilization and memory usage, using a stripped down version of the Contiki OS UDP/IPv6 stack. The IPv6 address can be preconfigured the before the microcontroller is powered on. Communication with the sensors takes place with a 1-Wire bus [43], which gives us the ability to utilize any 1-Wire bus capable sensor on the sensor board. In our case we are using temperature measurements with a Maxim Integrated Products, Inc. DS18S20 sensor †[42].

Communication between the sensor board and the gateway is done through ZigBee transceivers. In our prototype, Digi International Inc. Digi XBee® ZigBee chips are used, providing a range of 90 m, but it is possible to use their ZigBee PRO trasceivers to increase the range to up to 1.6 km [41]. These chips can also support data encryption, using a symmetric-key algorithm (AES). Because of the use of symmetric key encryption, the cryptographic key is common to the sensor board and the gateway and is hardcoded *a priori* into the sensor board.

3.2.1 First prototype: Arduino Duemillanove + Zigbee + Raspberry Pi

The first board that was used as prototype for the sensor board was the Arduino Duemilanove, with an Atmel ATMega 168 microcontroller. This board was used in an earlier project within Ericsson in a crab farming setting in China [24]. For the communication with the gateway, a Digi Zigbee S1 chip [41] was used by both the sensor board and the gateway. In the first prototype no security was implemented, thus all the communication was unencrypted. This solution was considered acceptable since this realization was only intended for use as a development prototype.

As an early prototype, limitations came up and solutions had to be found. The first limitation was the code memory size of this type of Arduino. This processor's flash memory capacity was 9 KB, this limited the size of the firmware that could be used, hence limiting the functionality of this prototype. The main problem was that the code size of the Contiki OS UDP/IPv6 stack was more than 30 KB [21]; hence this UDP/IPv6 stack could not be used. The solution was to create a stripped down version of this stack by removing unneeded functionality, resulting in an IPv6 stack implemented as an Arduino sketch that was smaller than 9 KB.

* We assume that one interface is an IEEE 802.15.4 interface and the other is an Ethernet or wide area network interface. For details see section 3.5.

† Details of this sensor can be found at http://datasheets.maximintegrated.com/en/ds/DS18S20.pdf

3.2.2 Second prototype: Arduino Uno + Zigbee + Raspberry Pi

For the second prototype (shown in Figure 3-2), the board that was used was a similar Arduino board, but one with greater flash memory capacity. The Arduino version that was used for this second platform was the UNO which is based on an Atmel ATMega328 microcontroller. This microcontroller is more widely-used and supported. Additionally, with this microcontroller it was possible to run the Contiki OS, since this OS had already been ported to this type of microcontroller. The ATMega328's flash memory is 32 KB that offers greater freedom to write more complex firmware than was possible in the first prototype (where only the most basic functionality could be realized). On this second prototype, AES encryption was utilized for the communication with the gateway. This encryption was available by using the embedded functionality of XBee module. Since, the memory capabilities of this prototype are better, the IPv6 library from Telecombretagne [21] was used in order to utilize this characteristic.

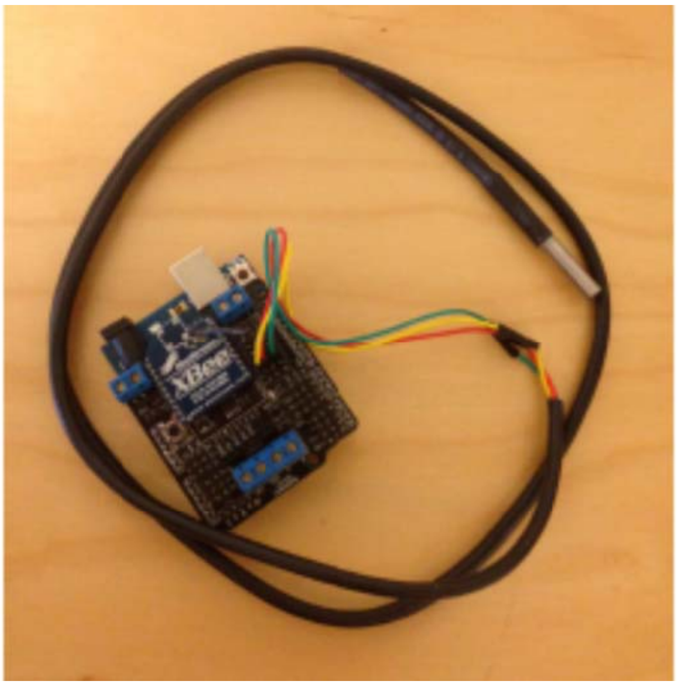


Figure 3-2: Arduino UNO with XBee shield and 1-wire temperature sensor

3.2.3 Third prototype: Custom platform + Zigbee + RaspberryPi

As a next step, our design was prototyped using a breadboard (shown in Figure 3-3). Taking cost into account, the cost could be decreased if we used a custom design of the sensor board rather than purchasing an Arduino device. In terms of energy demands, the power consumption of the system was decreased (from that of using the Arduino + Contiki) since we implemented a power-efficient "sleep mode".

In the future a compact product could be created by combing the selected integrated circuits of our prototype design a 3D-printed case. In large volumes the cases could be injection molded; further decreasing the system's cost.

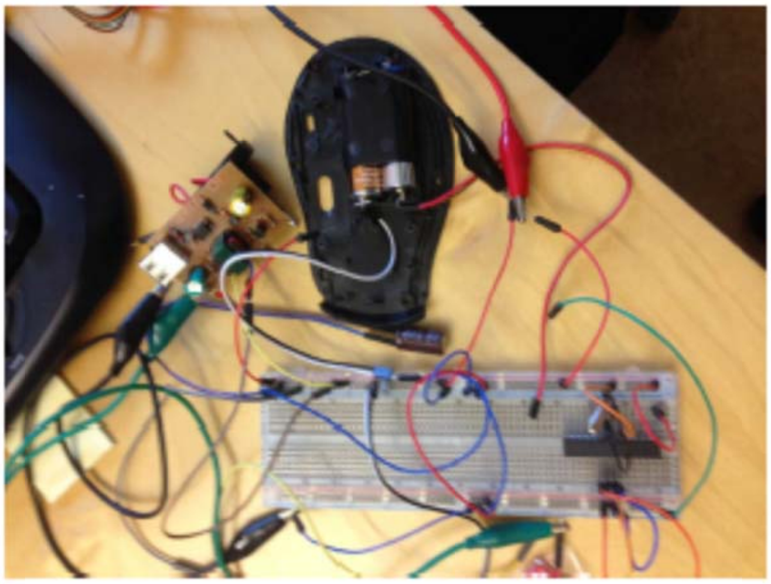


Figure 3-3: Custom platform with breadboard, powered by 2 AA batteries

3.3 Power Management

In this section we consider various aspect of power management. The section begins with a discussion of power sources, since the type of power source will have a large impact on what types of power management can be implemented. This is followed by a discussion of how power consumption might be decreased further.

3.3.1 Power Sources

Considering the cost of a sensor network, the number of nodes, and the expected operating time of the nodes, manual maintenance should not be mandatory. As the sensors are located in a field and there is generally no access to the power grid; thus, there is a constrained choice of power supplies. In this section we consider a number of efficient solutions for supplying power to the sensor board. This discussion takes into account a number of alternative power sources and the power consumption of the software and hardware components chosen for the prototype sensor board. Note that in this discussion we consider only the third prototype.

3.3.1.1 Power Sources

A number of alternative regarding power sources are listed in Table 3-1. We do not consider any of these alternatives as optimal, since no single alternative provides an optimal solution. In any case we have to minimize the power consumption of the sensor board. In any case, we should not selected sensors that would deplete the power source's energy within a few days, as a farmer should not need to maintain the sensor board that often. In fact, the system should be able to operate for at least the duration of a growing season for the field where the sensor nodes are placed.

Table 3-1: Potential power sources

Renewable	Solar, Wind, Wave
Non-renewable	Batteries
Main advantages of renewable versus non-renewable sources	Extended time of autonomous (maintenance free) operation Potentially much higher ratio of (Total output energy over life time of power supply)/(weight of power supply)
Main disadvantages of renewable versus non-renewable sources	More difficult to deploy Adds complexity and extra points of failure Typically higher initial and total cost Bulky system Heavy dependency on environment conditions

3.3.1.2 Batteries

The most common power supply that we consider as a reference is batteries. Batteries are cost effective and rechargeable batteries could be used. However, we do not think that rechargeable batteries will be used in the typical case.

The drawbacks of a rechargeable battery are:

- Higher cost;
- Lower capacity for one discharge cycle compared to non-rechargeable batteries;
- In a low power application there is no total cost savings due to ability to recharge the battery, as the discharge cycles are so long that batteries will last longer than the sensor nodes; and
- The batteries will need to be replaced because they have become too old before they have become overused (i.e., they cannot be recharged).

Table 3-2 contains a comparison of the most typical battery elements. From this table we can see that if the size of the battery is the major limitation, then the CR2032 is a good choice since it is the most popular choice among coin cell type batteries. On the other hand, the CR2032 is not efficient and relatively expensive. D batteries have the lowest cost for a given amount of energy, but they are bulky. The best tradeoff regarding size and energy is the ubiquitous AA battery.

Table 3-2: Typical characteristics of most common battery elements

Name (IEC designation)	Typical Capacity (mAh)	Voltage,V	Energy (mWh)	Price (SEK[*])	Energy/price (mWh/SEK)
AAA (R03)	1200 (alkaline)	1.5	1800	4	450
AA (R6)	2700 (alkaline)	1.5	4050	4	1012
C (R14)	8000 (alkaline)	1.5	12000	15	800
D (R20)	12000 (alkaline)	1.5	18000	15	1200
9V Transistor (CRV9)	565 (alkaline)	9	5085	29	175
CR2032	240 (Lithium-MnO_2) 3V	3	720	12	60

[*] Price is the lowest price per battery unit in value packs found at Class Ohlsson website - http://www.clasohlson.com/se/b/El/Batterier-laddare/Alkaliska-batterier

3.3.2 Further improvement on hardware design

In order to extend the battery life-time of the custom sensor platform, an external power regulator was used. As can be seen in Figure 3-4, the final result on the size of the platform is significant. The ATMega328 chip was used on the breadboard along with the XBee S1 module. The new addition is the Microchip MCP1252 power regulator [44]. This prototype is used with 2 AA batteries to allow the platform to operate for approximately two years while sending data with a frequency of twice per day. The schematics and power consumption calculations can be found in Appendix A and Appendix B. This is achieved because MCP1252 power regulator has the shutdown pin (SHDH) which allows it to be turned completely off and thus isolating the output circuit from the power supply.

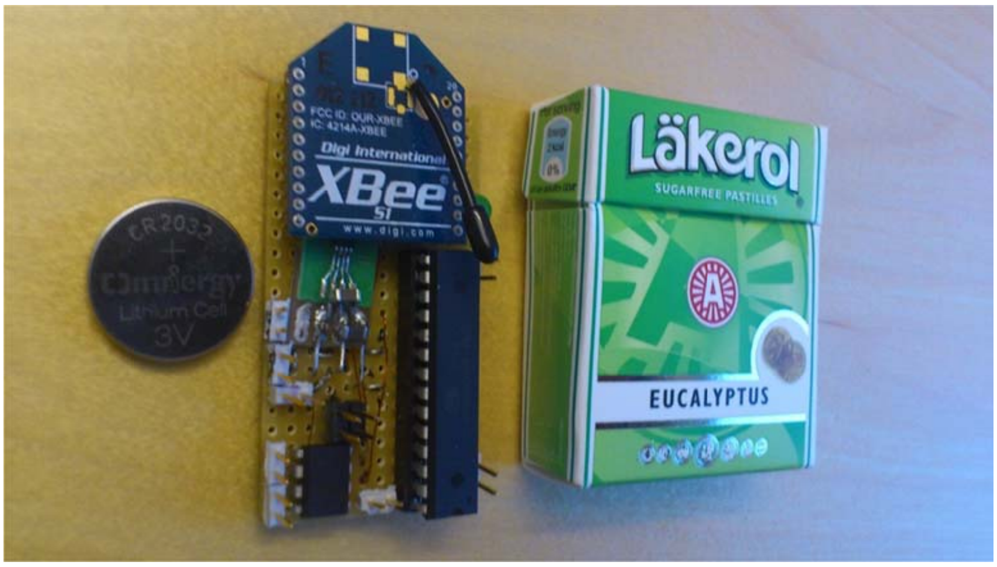


Figure 3-4: Physical size of the improved custom sensor platform

3.4 Sensor Hardware

This section includes a description of the protocol that is used for sensor modules on the platform along with two connection options.

3.4.1 1-wire bus system

The sensors were used for this project use a 1-wire bus interface. This is one of the most common interfaces for communication between sensors and Arduino motherboards; hence there is lots of example code [38]. This protocol is well-known because it provides an easy to use, low-cost solution that cooperates effectively with Arduino and many other microcontroller based platforms. There are freely available libraries that can be used in order to utilize these sensors. With a single line, the sensors communicate *and* get power. Additionally, multiple sensors using the same protocol can be connected to the same bus called by Maxim Integrated (formerly Dallas Semiconductor) a "Microlan" [39].

Regarding the connection to Arduino, the library that was used initially on the Arduino to communicate with sensor via its input/output pins, required 3 wires to be connected. Because 1-wire devices are powered through the same wire as they receive and send data, this wire has to be connected to V_{cc} through a pull-up resistor. Thus in order to use 1-wire devices they should be connected by three wires to an Arduino board: data, ground, and V_{in}.

Figure 3-5 shows the connection of Arduino and to the DS18S20 sensor. The DS18S20 temperature sensor's data pin is connected to pin 2 of Arduino for input. Any device that uses this 1-wire bus could be connected in the place of the temperature sensor.

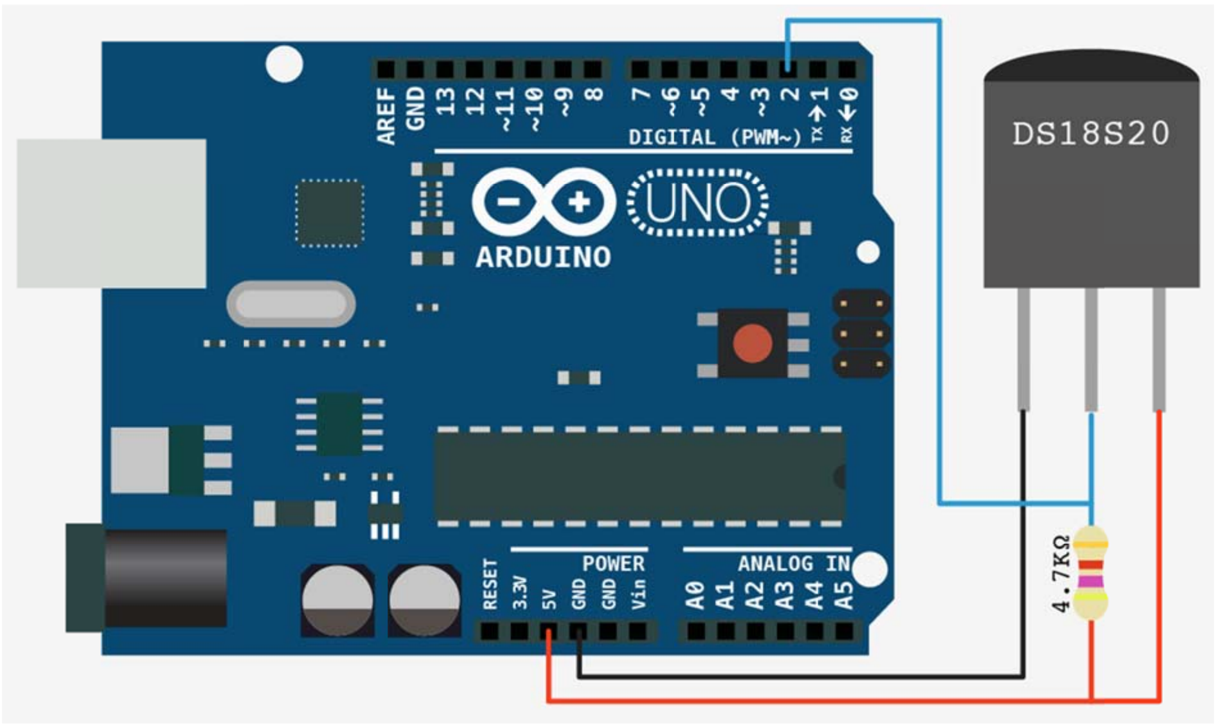


Figure 3-5: Temperature sensor DS18S20 hooked-up on Arduino Uno (Figure shared under: Attribution-Share Alike 3.0 Unported) [40]

3.4.2 Alternative way of connecting 1-wire sensors

Figure 3-6 shows the connection of the sensor to an Arduino board without using the third wire and the pull-up resistor. A custom library enables the Arduino to be in mode where pins are internally connected to on-chip to V_{in} via a pull-up resistor.

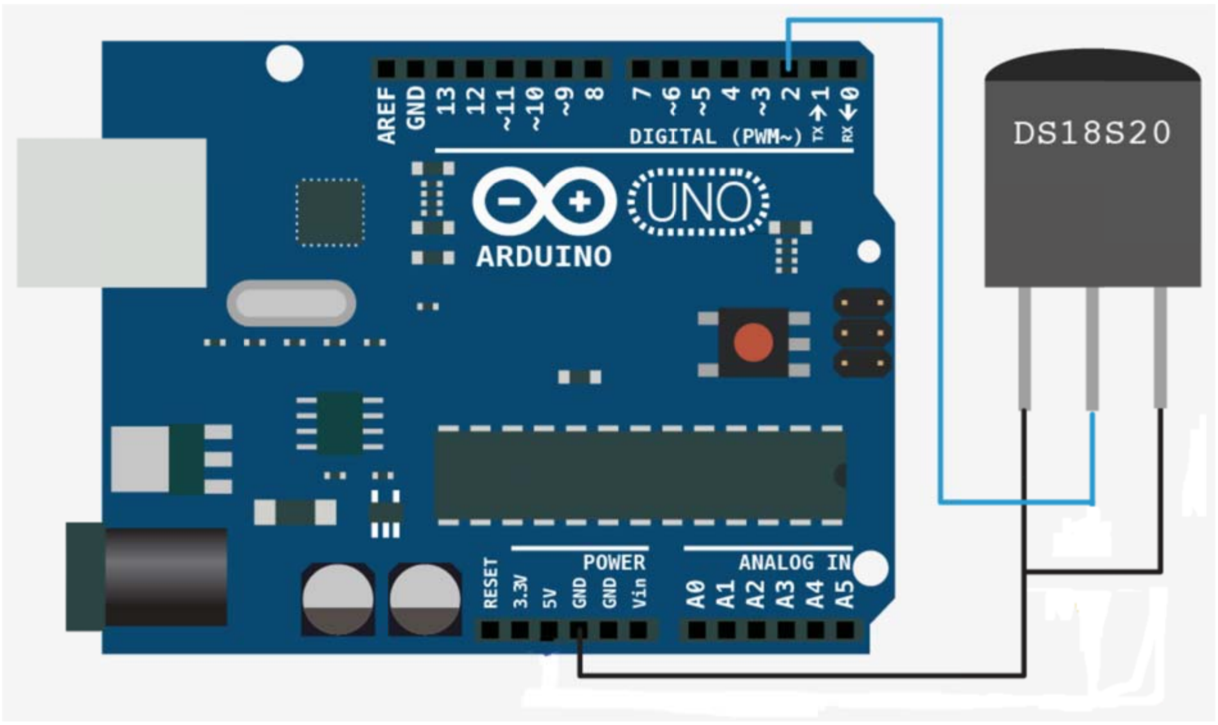


Figure 3-6: Temperature sensor DS18S20 hooked-up on Arduino Uno without using the pull-up resistor (Figure shared under: Attribution-Share Alike 3.0 Unported) [40]

This type of connection was used and tested and finally was not adopted since there was a downfall in stability of the system. There were some false measurements that were disappearing when the third cable was plugged.

3.5 Gateway

The motherboard used for the gateway in our system prototype is a Raspberry Pi. The Raspberry Pi is a low-cost, economy-sized, energy-efficient computer. We utilized this board as a gateway to receive packets from the sensor nodes and forward the collected data to the server. This board runs an open-source and light OS based on Debian, called Raspbian [25]. For the communication with the sensors, an XBee Series 1 transceiver (the same transceiver as used by the sensors) was used. One of the benefits of using a linux* computer as a gateway is that it was easy to write a program to receive packets from the sensor nodes and forward them to server.

For this project a Java program was written to:

- Receive the packets via a serial port connected to the XBee transceiver using the RXTX API [37],
- Decompose the message and retrieve the IPv6 address of the source and the destination, and
- Send the packet to the destination server.

When de-capsulation of the message takes place, the destination address is identified as IPv6 or IPv4 (based upon the form of the IPv6 address). The gateway relays the packet to the server at the specified destination address. The destination address, which is the server's, is hardcoded in sensor firmware.

Figure 3-7 shows the packets being received by gateway and the path that they follow until they are sent to the server. A Digi XBee S1 transceiver is connected to the serial port of the Raspberry Pi computer and receives incoming packets from sensors. The "Packet processor" is the program that receives the packets from the serial port (where the XBee S1 is connected) and translates these packets to UDP/IPv4 (or IPv6) packets to be transmitted over an Ethernet. Note that in our tests, the XBee S1 chip deals with fragmented packets; so that the packets received via the serial port have already been reassembled. The packets are placed into the outgoing IP packet queue of the OS and are subsequently sent by the built-in Ethernet Network Interface Controller (NIC) to the Storage Server.

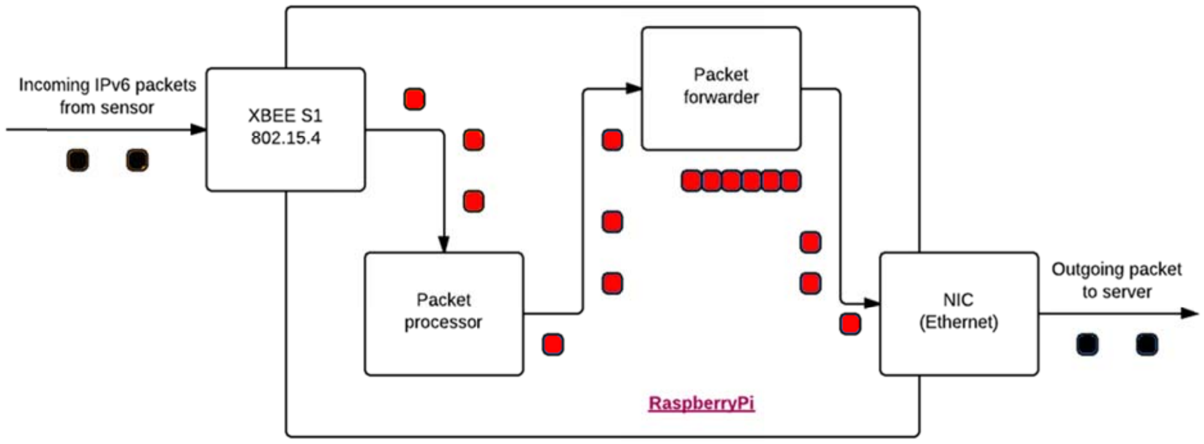


Figure 3-7: Packet Processing and Forwarding on Gateway

Regarding the authenticity of the messages that pass through the gateway, there is no security check whether the messages received from the sensors are malicious of not. However, we know that when we use the AES encryption of the traffic over the XBee link that the packets can *only* come from a source that knows the correct symmetric key as otherwise the receiver would not be able to decode

* Raspbian is a linux distribution.

the packet into a valid plain text IPv6 packet. The packet processing designed for this project *assumed* that the sensors are *not* rogue sensor nodes, i.e., they do not send fake data. Checking the authenticity of packets in terms of their claimed source and destination is planned for a future version of the system.

Figure 3-8 presents the prototype gateway as it is used in our tests of the system. The XBee S1 chip is connected through a serial port to the Raspberry Pi. The XBee S1 chip is assumed to be able to receive all of the packets from sensors that are within range. This prototype gateway uses the Raspberry Pi's embedded 5V miniUSB port for power (connected via the black cable on the left in the figure). The Ethernet network interface is connected to a gray Ethernet cable via a RJ45 jack as illustrated in Figure 3-8.

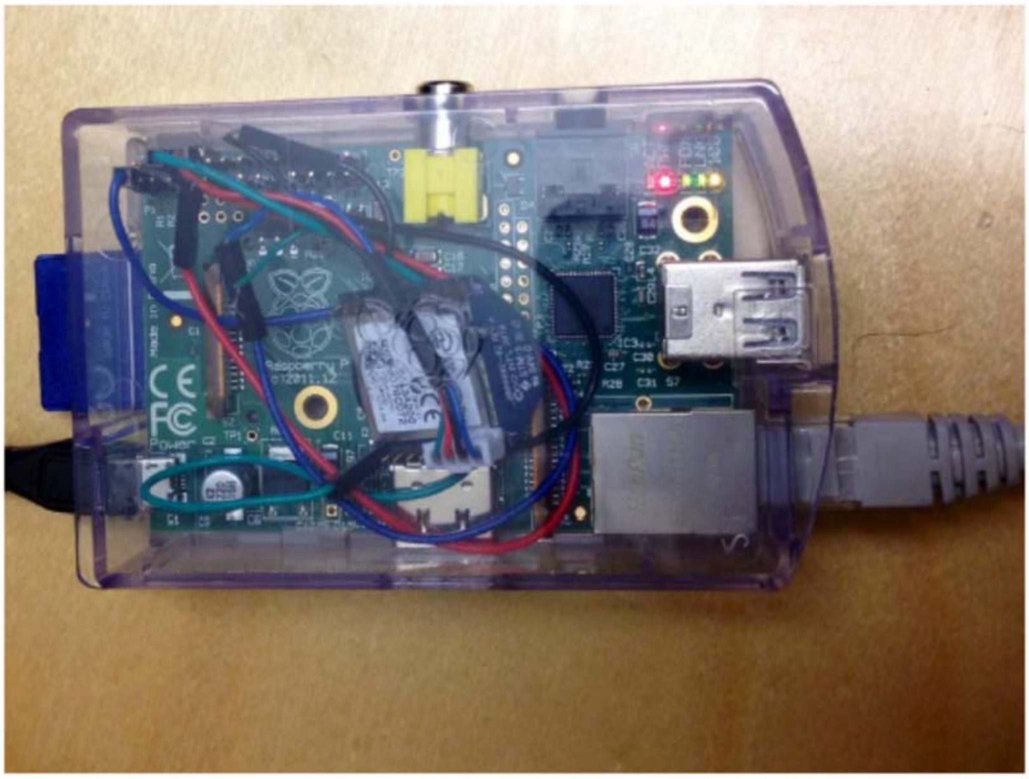


Figure 3-8: Prototype Gateway built using a Raspberry Pi with a XBee S1 tranceiver connected via a serial port

3.6 Data Storage Server

At the beginning of the design process, the storage requirements were low. A simple system that could run a database was thought to be sufficient. The initial implementation utilized a Raspberry Pi running the Raspbian OS [25] to implement a database and process the test queries. Such a system was sufficient when only a few sensors were submitting data. However, to provide better scalability (in terms of storage and performance) a Cloud based system is a more viable solution. Cloud computing and storage services are very convenient for applications that use sensor data due to huge amount of data that can be generated and because of the potentially large numbers of queries that such a collection of data could receive. There are already cloud services that offer device and data management, such as the "Device Cloud" system that Etherios Inc. [30] offers. For this project, a cloud solution was intentionally avoided and a Raspberry Pi was used as storage server because of network, cost, and time limitations. The system was developed within Ericsson's corporate network setting and permission would be needed to access the Ericsson Cloud using IPv6. It was though that getting this access could be time-consuming and was unnecessary for the planned evaluation of the prototype.

For the initial testing the web server and analytics server both reside on the same physical machine as the storage server. To scaling the system up each of these types of servers would be realized by a logical pool of servers running in a cloud. Such a cloud based implementation offers very high scalability as has been demonstrated by others as described below.

Regarding the database that is used for the project, a MySQL [32] system was implemented. This database was selected because MySQL runs on the Raspberry Pi and MySQL is one of the most common database management systems. Furthermore, as described by Ruslan Mukhammadov[31], MySQL is a suitable solution in terms of read performance when large amounts of data are retrieved from the database, while writes to the database are much less frequent. This solution is efficient when it comes to data analytics, as huge amounts of data have to be retrieved from the database when historical records or large amounts of contemporary data are combined for comparisons.

3.7 Web Server and User Interface

In order to implement real-time monitoring of the sensor data via a Web interface that users could use to access these data, we designed a web service that presents the sensors on a map along with the data. An OpenStreetmap [33] map with a leaflet [34] library is used to present the data via the website. Both OpenStreetmap and leaflet are open source solutions, helping to keep the system cost low. For the data input into the map the GeoJSON [11] format is used with the help of a php script that retrieves the data from the database and encodes the data into the GeoJSON format. The encoded data are then presented via the map. AJAX technologies are used for refreshing parts of the page with new data *without* refreshing the whole page. The web server that is used is the well-known Apache HTTP Server [37]. The web user interface is an easy-to-use platform in which users have the ability to add sensors, fields of different types, and monitor real-time the data from these sensors. Two different types of users can use the interface: users and administrators. In the section 4.2, the process of generating web pages and presenting the data is described in detail, along with screenshots of the web interface.

3.8 Security

The communication between the sensors and the gateway is secured by AES using the built-in functionality of XBee module. The gateway-server connection can be secured or unsecured. There are two main options for securing this communication:

- IPsec with AES in CBC mode and use of PKCS5 padding [26,27,28] and
- SSL tunneling with Diffie-Hellman (DH) authentication, RC4 encryption, and MD5 hashing [29].

When using SSL a properly signed certificate has to be purchased from an certificate authority or a self-signed certificate generated and signed by the user's certificate authority. The certificate authority's public key certificate must be accessible to both the server and the gateway.

For testing purposes unencrypted communication can be used in order to find out if the system is functionally working and if the packets are correctly received by the server. After this initial testing either IPsec or SSL tunneling should be used between the gateway and the data storage server. Similarly either IPsec or SSL tunneling should be used between the data storage server and the analytics server and again between the analytics server and the web server. SSL tunneling should be used between the web server and the user's web browser. This means that the user's web browser should also have a copy of the certificate authority's public key certificate that is used for the user's web browser's certificate and the web server's certificate. Note that the certificates used between the web server and the user's browsers can use a different certificate authority that is used for securing the communication between web server & analytics server, between the analytics server & data server, and between the gateway & data server. It is even possible to use a public key certificate scheme in conjunction with securing the communication between the sensor nodes and the gateway. In the future one might even consider securing the communication between the sensor node and the data server using certificates while using preconfigured keys with the sensors nodes and the gateway. Details of these alternatives security schemes should be consider in future work.

3.9 Cost

A summary of the cost of the system hardware is described in this section. The goal was to keep the cost low in order to make the system easier to deploy for users. Therefore the software and tools for server are open source.

Regarding the hardware, the cost* is described below:

For the Gateway:

- 1 Raspberry Pi Type B and 8GB SD card : **$45**
- 1 Micro USB Power Supply Adapter, 5V, 2A : **$9**
- 1 Raspberry Pi Type B Case : **$6.50**
- 1 XBee 1mW Wire Antenna - Series 1 (802.15.4) : **$23**

Summary: **$83.50**

For the Sensor Platform:

- 1 Arduino Uno board : **$28.50**
- 1 DS18S20 High-Precision 1-Wire Digital Thermometer : **$3**
- 1 MCP1252 power regulator : **$1.60**
- 1 XBee 1mW Wire Antenna - Series 1 (802.15.4) : **$23**
- 2 x AA batteries: $**1.20**

Summary: $**57.30**

The gateway is a low-cost solution that includes a RaspberryPi and an XBee S1 module. The sensor platform includes an Arduino Uno, the power regulator, the XBee module, the battery module and one temperature sensor as a reference, more sensors can be added considering the needs.

The final prototype that was developed does not require the usage of the Arduino Uno board, but only the microcontroller unit which costs $3.7. So the price summary becomes: $32.5 for the sensor platform.

* The cost is calculated based on the prices found in : https://www.sparkfun.com/products/8665 , http://uk.rs-online.com/web/ and http://www.digikey.com/product-search/en. See also Table 3-2.

4 Analysis

This chapter presents the purpose of designing the proposed precision agriculture system, the limitations that were faced during the implement, and the evaluation of the system as implemented by the prototype. The decisions that lead to the choices that have been made during the design and implementation of the project are presented in order to give the reader both a storyline of the development of the system and so that these decisions are made explicit (as these decisions might be made differently by others in the future).

4.1 Design purpose and limitations

The main goal of designing a precision agriculture system is to introduce additional information and communication technology into cultivation processes in order to enable more effective agriculture, while simultaneous reducing the impact of this cultivation on the environment (i.e., improving the sustainability of agriculture). Designing such a system is not an easy job, since a generic implementation that would work in most of the cases is something that cannot be easily realized.

Following discussions with agricultural scientists and farmers, the common outcome in most cases was that it is very difficult to have a generic set of rules that are applicable everywhere. There are distinct aspects that have to be considered for each specific setting. Location of the field, distance from water source(s), inclination, quality of the soil, pest infections, and historic statistics are some of the aspects that have to be considered when a cultivation plan is designed and followed. The diversity of local conditions is a major problem in decision making in each case. There could even be differences in a specific place year by year causing variation in the crops that are cultivated. Even within a range of several meters, differences in conditions can be observed that cause heterogeneity in cultivation and the resulting production. The knowledge source to recognize and address these specifics is the users themselves as they apply their cultivation methods over a period of years in custom ways in order to achieve the best results. One of the goals of the system's design is to allow these users to pass this knowledge on to the system in such a way that the system can enrich its knowledge base. This transfer of knowledge can occur by customizing the system for the specific needs that each user has. By allowing the user to add custom rules for each specific field (more specifically for each part of the field) and for every sensor that is utilized, the knowledge is transferred from the user to the knowledge base of the system[*]. After using the system for a period of time, historical data can also be accessed in order to retrieve information about past cultivation schemes in both specific settings and in a diverse set of settings. Correlating old cultivation schemes with the corresponding yields can lead to better decision making for future cultivations.

With respect to the sensor platform part of the system, an older prototype platform was initially used as a base. Our first hardware sensor prototype was inherited from Ericsson labs' crab farming project in China. However, the design of the database and web server part of our project started from scratch. Because the time window for this project was limited we sought an initial base and inspiration from other projects. As noted earlier we tried to use the Contiki OS on this first sensor prototype, but found that we were not able to use the Contiki OS because the open-source operating system had not been ported in ATMega168 microcontroller and the size of the IP stack code was too large for this target platform. After spending more than 3 weeks of trying to use Contiki on this prototype, we decided to move forward with a second prototype based upon the Atmel ATMega328 and the firmware we had already developed.

4.2 Web Interface

A web interface was developed order for users to easily take advantage of the sensor system. Users can login to the system and then use the functionalities that the prototype service offers. The login process and the services that are available are described in this section. For simple users (assumed to typically be farmers who farm one or more fields) after logging in, the user can access the data associated with their fields. These data are provided in different forms by:

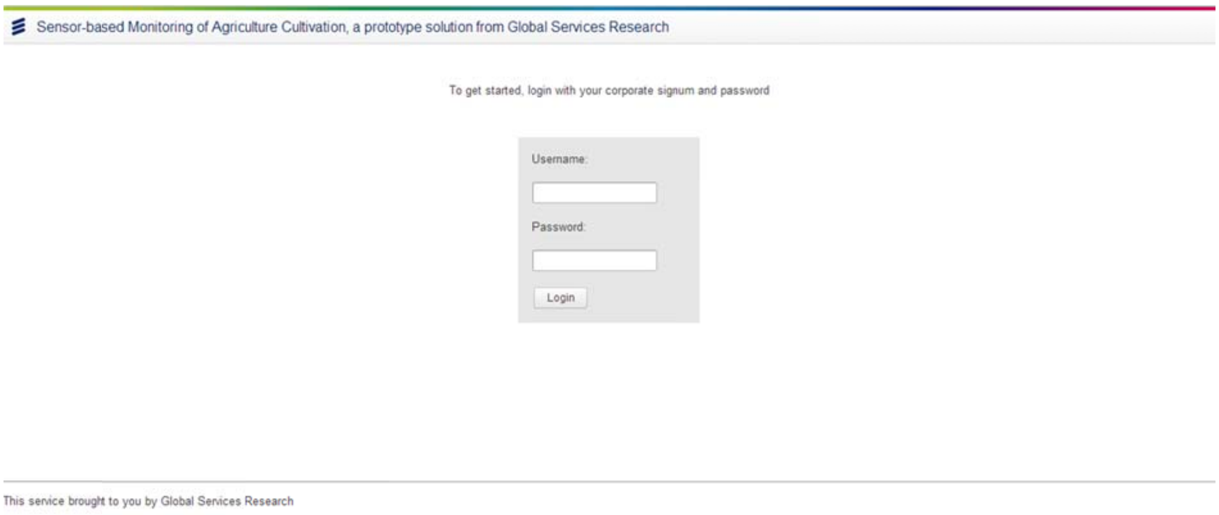
[*] Here we consider that the part of the database that contains such rules forms a knowledge base.

- A real-time monitoring service for all the fields and sensors (respectively),
- Data analysis tools that include presentation of historical data, combination of data series, and annotations that users themselves can add according to their needs, and
- An alerting system that notifies the user when single or combination of measurements is meeting specific rules that the user has specified.

When administrators login to the system, they can use the administrative interface which offers different functionality than the user interface. Administrators can access the list of users, the list of fields, and the list sensors and then add or edit any of these entries. A user-friendly interface was developed in order to make the process of inputting or editing of the sensor locations and characteristics, and adding or editing the list of fields and users easy. This interface includes interactive maps that allow the administrator to draw fields and to add the location of each sensor simply by clicking on the map.

In the following sections, these different processes are each presented along with the corresponding screenshots.

Every user first encounters the login screen shown in Figure 4-1. This requires that the user enter their user name and corresponding password. These user names and passwords can be managed using any of the techniques[*] that are supported by the Apache HTTP Server.


Sensor-based Monitoring of Agriculture Cultivation, a prototype solution from Global Services Research

To get started, login with your corporate signum and password

This service brought to you by Global Services Research

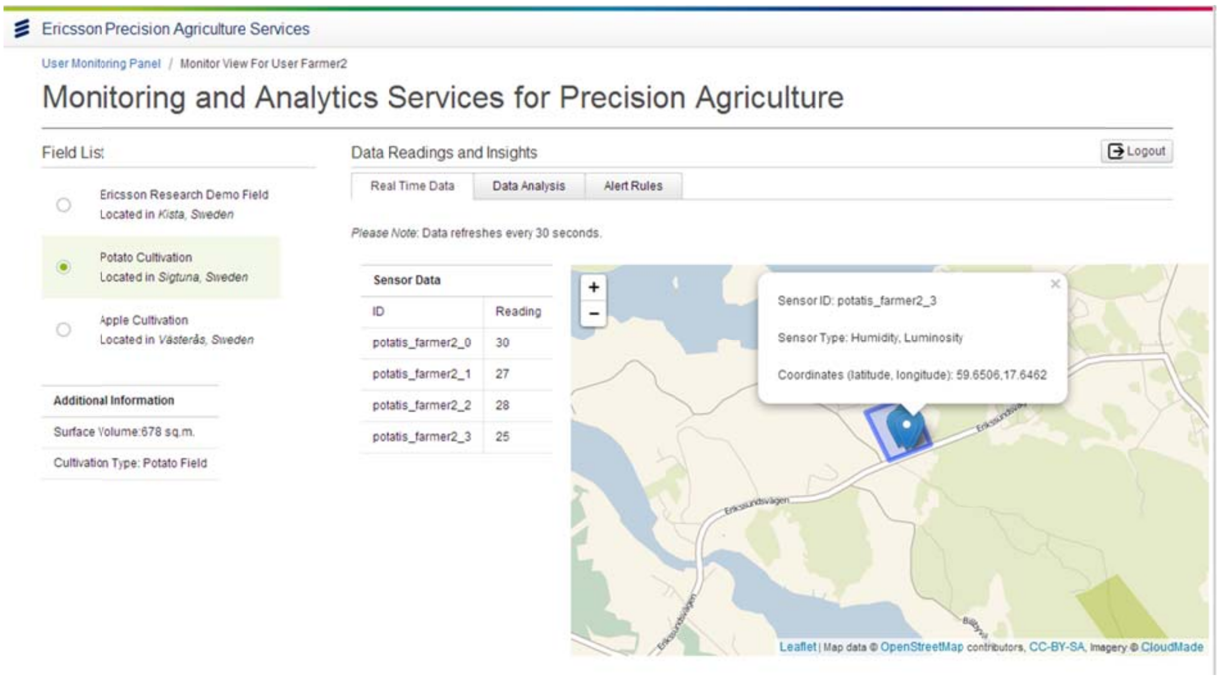
Figure 4-1: Login Screen

4.2.1 Interface for users

After a user logs into the system, the interface that the user is able to use are presented in this section by using screenshots and giving a short description of each screen. These screens are all part of the User Monitoring Panel.

In Figure 4-2, the real time data of the user "*farmer2*" are presented through the "Real Time Data" tab, where the table with the sensor data includes the latest measurements and when the user's pointing input device "hovers" over an item, a pop-up balloon appears on the corresponding sensor pin on the map showing the sensor's identifier (ID), type, and location. On the left side of the page, a field list is located along with some additional information for the currently selected field. When a field is selected, the sensor data and the map are automatically refreshed using AJAX to maintain the real-time character of this service.

[*] See http://httpd.apache.org/docs/current/howto/auth.html for details of these different techniques.


Ericsson Precision Agriculture Services

User Monitoring Panel / Monitor View For User Farmer2

Monitoring and Analytics Services for Precision Agriculture

Field List

Ericsson Research Demo Field
Located in *Kista, Sweden*

Potato Cultivation
Located in *Sigtuna, Sweden*

Apple Cultivation
Located in *Västerås, Sweden*

Additional Information
Surface Volume:678 sq.m.
Cultivation Type: Potato Field

Data Readings and Insights

Please Note: Data refreshes every 30 seconds.

Sensor Data

ID	Reading
potatis_farmer2_0	30
potatis_farmer2_1	27
potatis_farmer2_2	28
potatis_farmer2_3	25

Sensor ID: potatis_farmer2_3

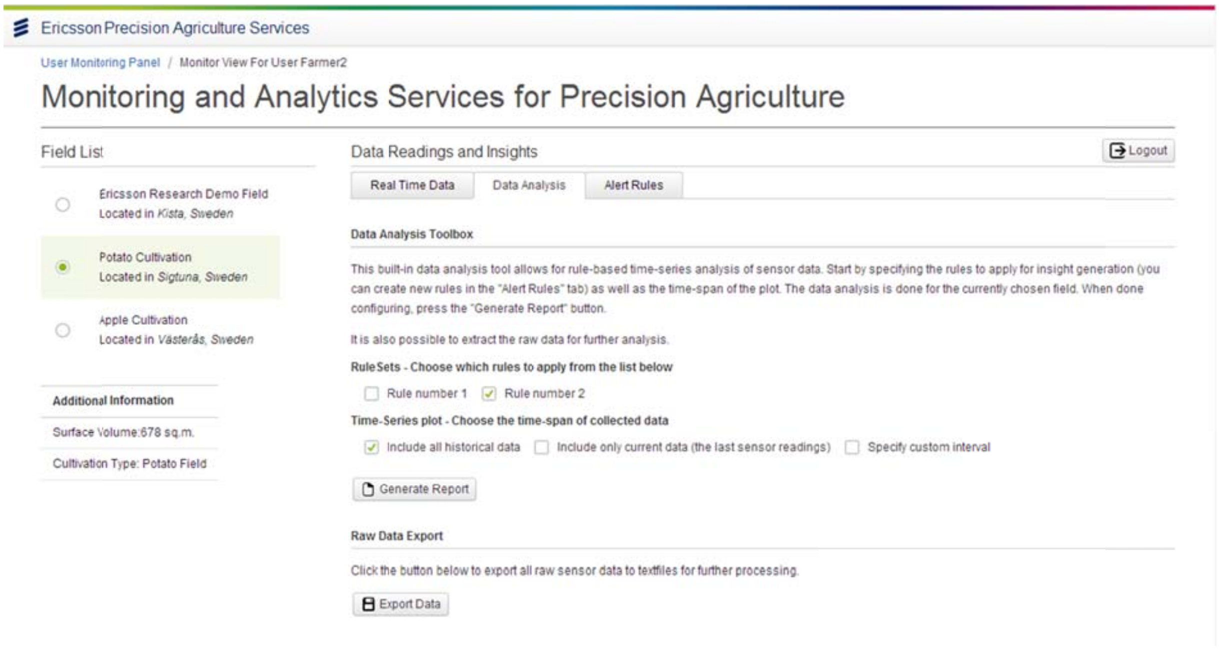
Sensor Type: Humidity, Luminosity

Coordinates (latitude, longitude): 59.6506,17.6462

Logout
Real Time Data
Data Analysis
Alert Rules
Eriksundsvägen
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © CloudMade

Figure 4-2: Real time data tab in monitoring service interface.

The second tab ("Data Analysis") on the monitoring and analytics service interface is presented in Figure 4-3. This interface includes a description of the provided service and help on how it works. After the user selects the rules that they want to be applied, a report can be generated (by pressing the "Generate Report" button) or the data can be exported to a text file (by pressing the "Export Data" button). A sample report is shown in Figure 4-4. The measurement data were manually generated for demonstration purposes and were not collected by the system.


Ericsson Precision Agriculture Services

User Monitoring Panel / Monitor View For User Farmer2

Monitoring and Analytics Services for Precision Agriculture

Field List

Ericsson Research Demo Field
Located in *Kista, Sweden*

Potato Cultivation
Located in *Sigtuna, Sweden*

Apple Cultivation
Located in *Västerås, Sweden*

Additional Information
Surface Volume:678 sq.m.
Cultivation Type: Potato Field

Data Readings and Insights

Data Analysis Toolbox

This built-in data analysis tool allows for rule-based time-series analysis of sensor data. Start by specifying the rules to apply for insight generation (you can create new rules in the "Alert Rules" tab) as well as the time-span of the plot. The data analysis is done for the currently chosen field. When done configuring, press the "Generate Report" button.

It is also possible to extract the raw data for further analysis.

RuleSets - Choose which rules to apply from the list below

☐ Rule number 1 [✓ Rule number 2]

Time-Series plot - Choose the time-span of collected data

[✓ Include all historical data] ☐ Include only current data (the last sensor readings) ☐ Specify custom interval

Raw Data Export

Click the button below to export all raw sensor data to textfiles for further processing.

Figure 4-3: Data Analysis tab in the monitoring service interface.

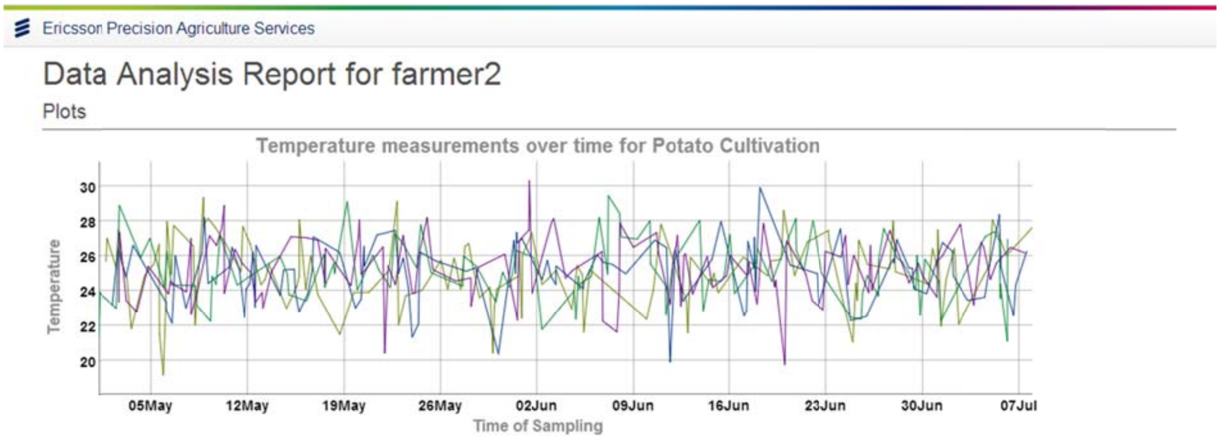
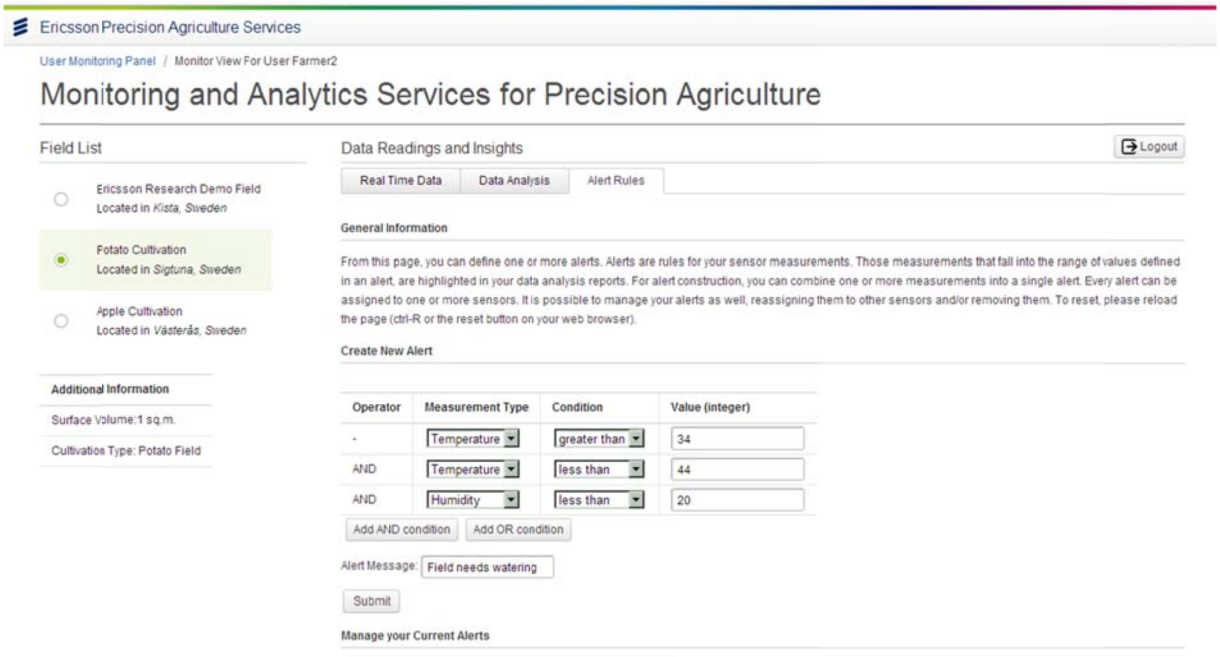


Figure 4-4: Sample report of temperature measurements over a two month period of time for potato cultivation. These results present the combined values of several (fake) sensors.

The alerting service setup is presented in Figure 4-5. A general description and the management of alert rules are included in the "Alert Rules" tab. The user can create new rules or edit custom rules that are to be applied on the data. When the specified conditions are met, an alert message is sent to the user. An example of an alert that is to be generated when the temperature is greater than 34^{o}C and less than 44^{o}C and the humidity is less than 20% is shown in the figure. When these conditions are met a message saying that the field needs watering is sent to the user. Users can add custom rules to suit their own needs. As shown in the figure the interface makes it easy for the user to define logical equations over sensor values to trigger an alert. In the future this could be extended to functions, rather than simply local equations involving current sensor values.


Ericsson Precision Agriculture Services
User Monitoring Panel / Monitor View For User Farmer2
Monitoring and Analytics Services for Precision Agriculture
Field List
Ericsson Research Demo Field
Located in Kista, Sweden
Potato Cultivation
Located in Sigtuna, Sweden
Apple Cultivation
Located in Västerås, Sweden
Additional Information
Surface Volume:1 sq.m.
Cultivation Type: Potato Field
Data Readings and Insights
Logout
Real Time Data
Data Analysis
Alert Rules
General Information

From this page, you can define one or more alerts. Alerts are rules for your sensor measurements. Those measurements that fall into the range of values defined in an alert, are highlighted in your data analysis reports. For alert construction, you can combine one or more measurements into a single alert. Every alert can be assigned to one or more sensors. It is possible to manage your alerts as well, reassigning them to other sensors and/or removing them. To reset, please reload the page (ctrl-R or the reset button on your web browser).

Create New Alert

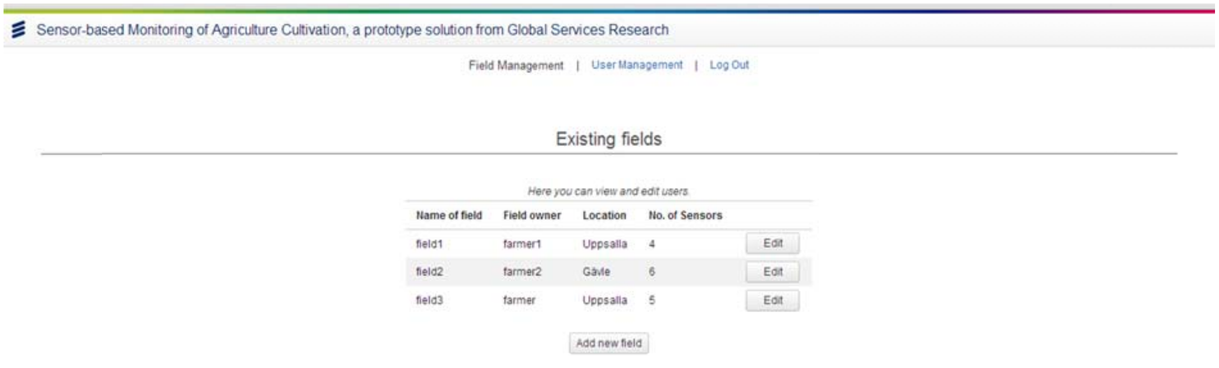
Operator	Measurement Type	Condition	Value (integer)
-	Temperature	greater than	34
AND	Temperature	less than	44
AND	Humidity	less than	20

Alert Message: Field needs watering

Figure 4-5: Alert rules tab in monitoring and analytics service interface.

4.2.2 Administrative interface

When administrators login to the system, they can access the list of the existing fields as shown in Figure 4-6. This list contains the name of the field, the owner (who owns the current rights of cultivation), the location and the number of sensors that are used in each field. This information can be edited and new fields can be added. The editing and addition processes are illustrated using the screen shown in Figure 4-7. This set of operations utilizes the first link at the top of the page.


Sensor-based Monitoring of Agriculture Cultivation, a prototype solution from Global Services Research

Field Management | User Management | Log Out

Existing fields

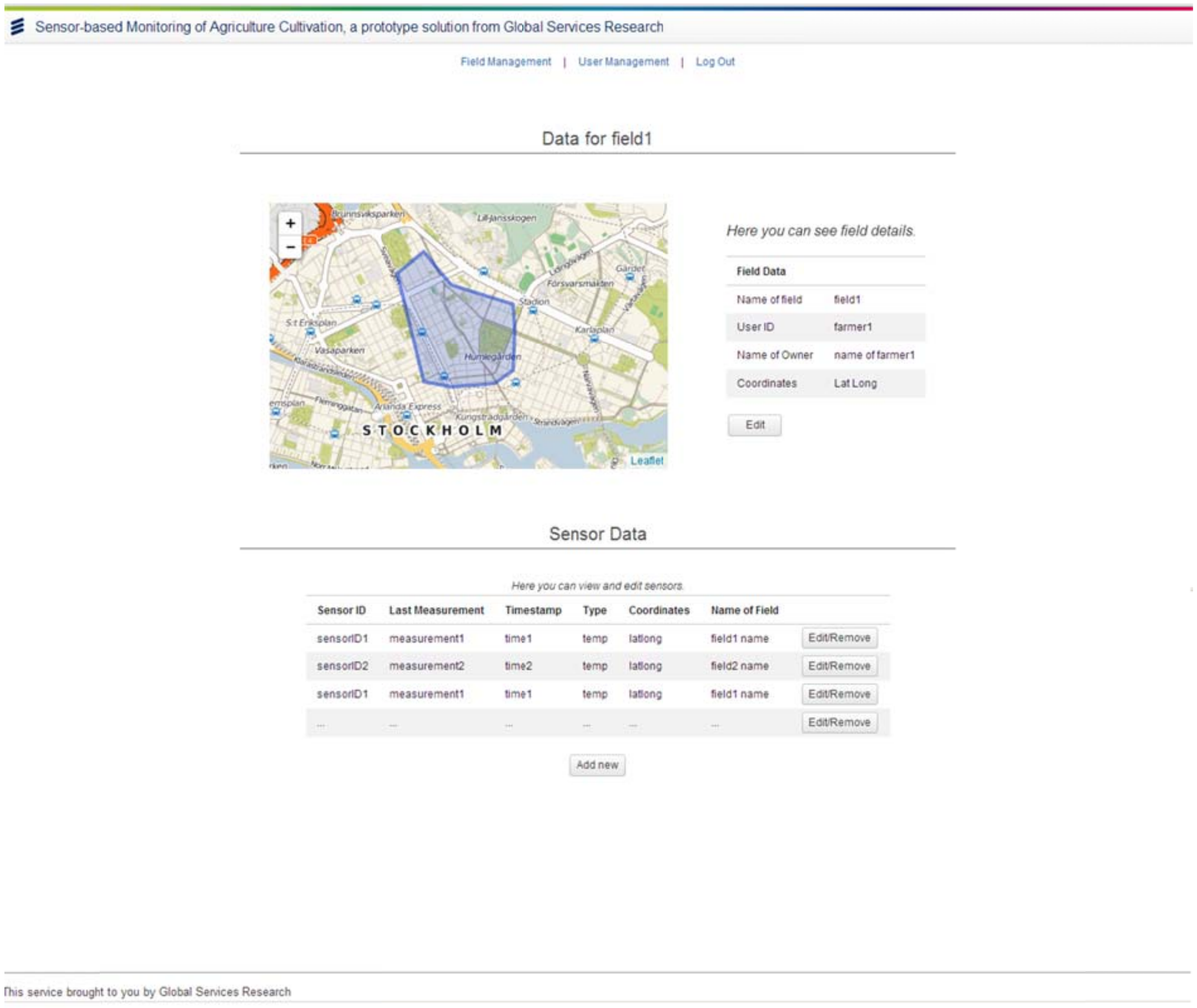
Here you can view and edit users.

Name of field	Field owner	Location	No. of Sensors	
field1	farmer1	Uppsalla	4	Edit
field2	farmer2	Gävle	6	Edit
field3	farmer	Uppsalla	5	Edit

Add new field

Figure 4-6: Existing fields list

In Figure 4-7, the field is described with all the desired information that is to be stored about this field in the database (such as owner name, field name, location, the field shape on the map, and the information about the sensors that are placed and used in this field). It is important to note that editing the set of fields updates the corresponding tables in the database for this field.


Sensor-based Monitoring of Agriculture Cultivation, a prototype solution from Global Services Research

Field Management | User Management | Log Out

Data for field1

Here you can see field details.

Field Data	
Name of field	field1
User ID	farmer1
Name of Owner	name of farmer1
Coordinates	Lat Long

Edit

Sensor Data

Here you can view and edit sensors.

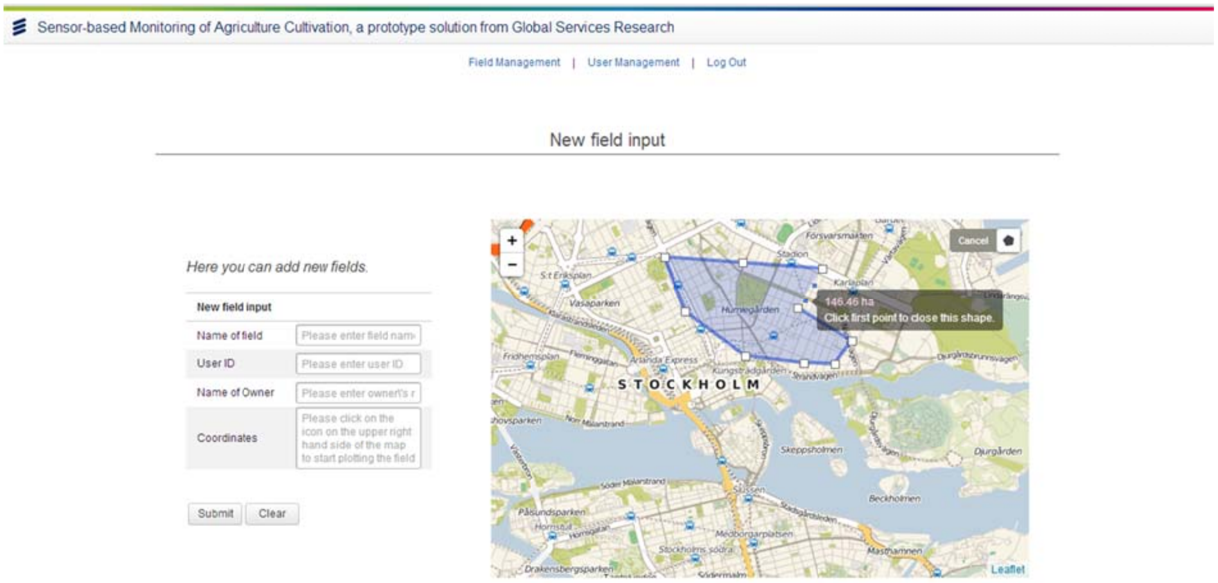
Sensor ID	Last Measurement	Timestamp	Type	Coordinates	Name of Field	
sensorID1	measurement1	time1	temp	latlong	field1 name	Edit/Remove
sensorID2	measurement2	time2	temp	latlong	field2 name	Edit/Remove
sensorID1	measurement1	time1	temp	latlong	field1 name	Edit/Remove
...	...	...	...	...	...	Edit/Remove

Add new

This service brought to you by Global Services Research

Figure 4-7: Field and related sensor data

The shape of the field is designated using the map with a pointer device, and then the coordinates are entered into the corresponding field of the form (associated with this web page) prior to it being submitted. This process utilizes a web page such as that shown in Figure 4-8.



Sensor-based Monitoring of Agriculture Cultivation, a prototype solution from Global Services Research

Field Management | User Management | Log Out

New field input

Here you can add new fields.

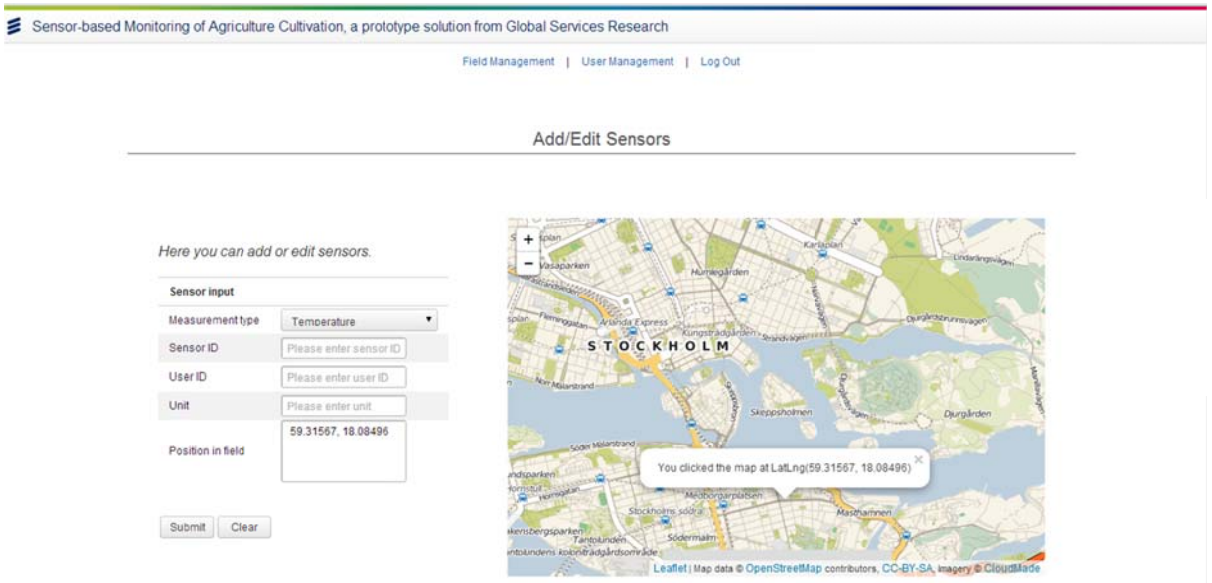
New field input	
Name of field	Please enter field nam
User ID	Please enter user ID
Name of Owner	Please enter owner's r
Coordinates	Please click on the icon on the upper right hand side of the map to start plotting the field

Submit Clear

146.46 ha
Click first point to close this shape.

Figure 4-8: New field input page

The sensors in a field can be added or edited by choosing the type of sensor, then adding information such as sensor id, user id, measurement unit, and the position of the sensor. The position of the sensor is retrieved by clicking on the map. The web page for editing and adding sensors is presented in Figure 4-9.



Sensor-based Monitoring of Agriculture Cultivation, a prototype solution from Global Services Research

Field Management | User Management | Log Out

Add/Edit Sensors

Here you can add or edit sensors.

Sensor input	
Measurement type	Temperature
Sensor ID	Please enter sensor ID
User ID	Please enter user ID
Unit	Please enter unit
Position in field	59.31567, 18.08496

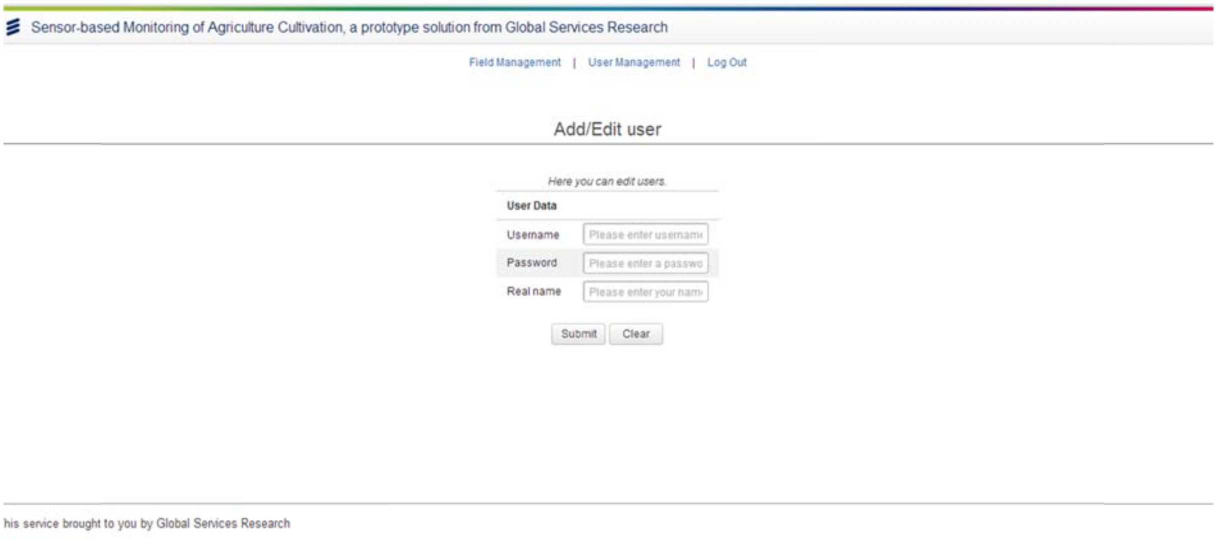
Submit Clear

You clicked the map at LatLng(59.31567, 18.08496)

STOCKHOLM
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © CloudMade

Figure 4-9: Add/Edit sensor data using the map

A user management page is accessed by clicking on the “User Management” link in the menu at the top of the page. Figure 4-10 shows how this add or edit functionality is presented to the administrative user.


Sensor-based Monitoring of Agriculture Cultivation, a prototype solution from Global Services Research

Field Management | User Management | Log Out

Add/Edit user

Here you can edit users.

User Data	
Username	Please enter usernam
Password	Please enter a passwo
Real name	Please enter your nam

his service brought to you by Global Services Research

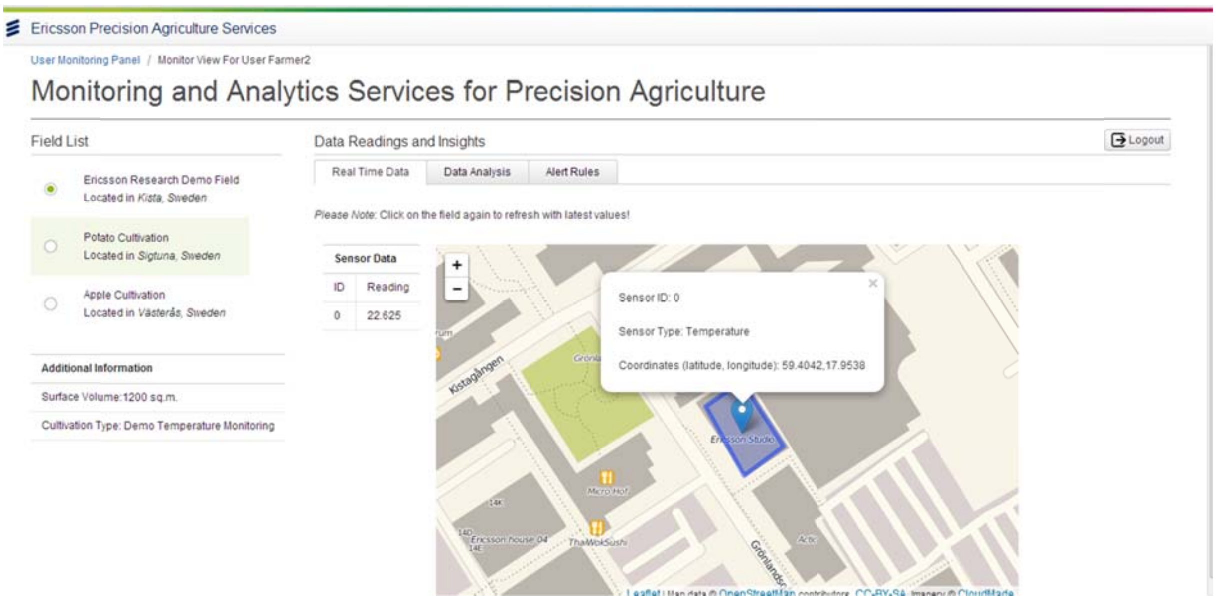
Figure 4-10: Add/Edit user

4.3 Evaluation

The evaluation of the prototype is a very important part of the project. This evaluation considers the design and implementation process. The first criterion is that the implemented prototype works as it should and fulfills the goals that it was designed for. In the following sections, the testing, the interface with which the users can interact, and the value of the outcome are presented. A business analysis is also included in order to present the options that would create revenue stream.

4.3.1 Presenting and testing

In Figure 4-11 the operation of the system as it was working on Ericsson’s Research Day 2013 is shown. This conference took place in Ericsson Studio. The location of this studio is shown on the map. There was one sensor sending data to the server with the sensor id “0”.


Ericsson Precision Agriculture Services

User Monitoring Panel / Monitor View For User Farmer2

Monitoring and Analytics Services for Precision Agriculture

Field List

Ericsson Research Demo Field
Located in *Kista, Sweden*

Potato Cultivation
Located in *Sigtuna, Sweden*

Apple Cultivation
Located in *Västerås, Sweden*

Additional Information
Surface Volume:1200 sq.m.
Cultivation Type: Demo Temperature Monitoring

Data Readings and Insights

Please Note: Click on the field again to refresh with latest values!

Sensor Data

ID	Reading
0	22.625

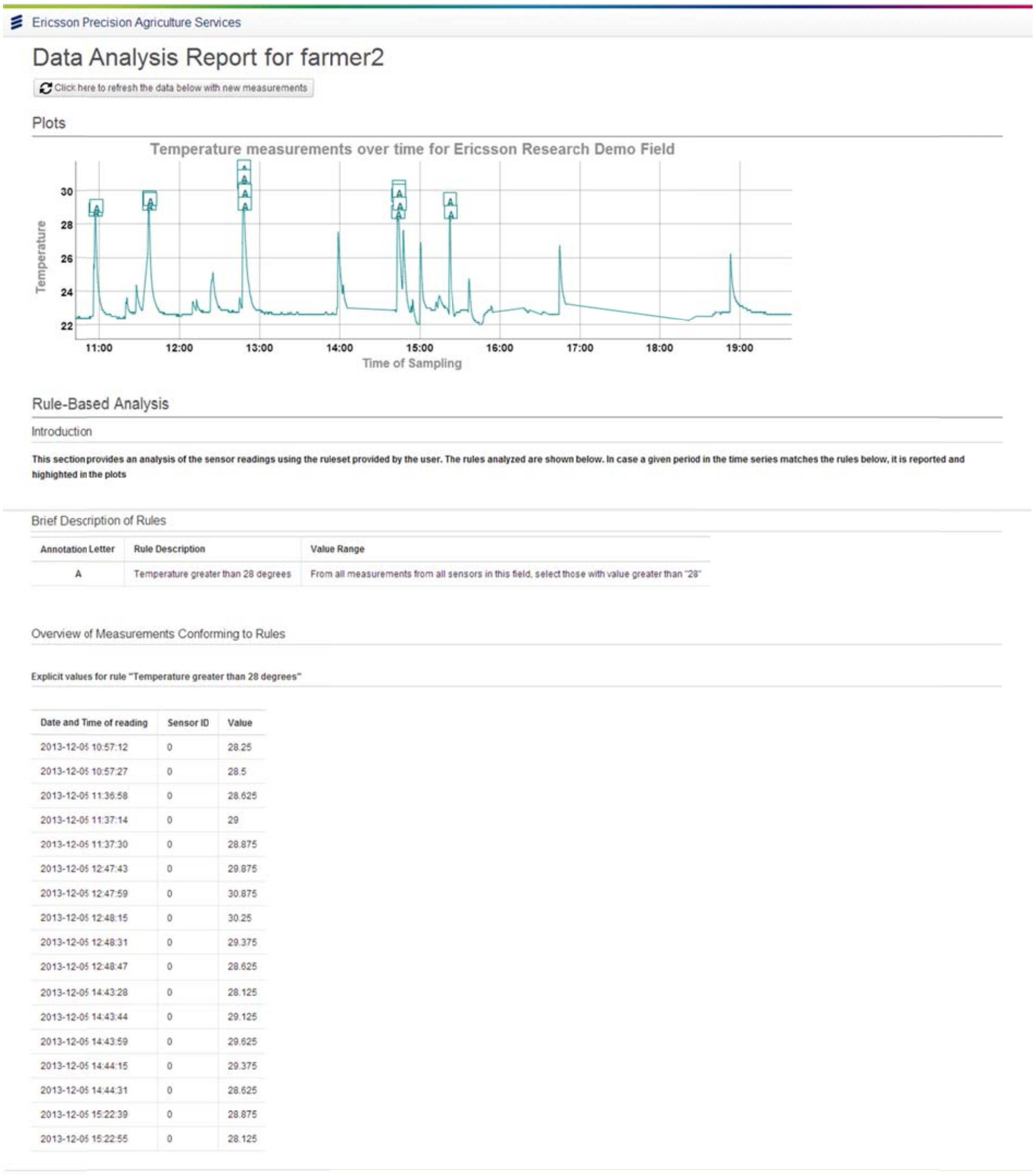
Sensor ID: 0

Sensor Type: Temperature

Coordinates (latitude, longitude): 59.4042,17.9538

Figure 4-11: Real time data of demo in Ericsson Research Day 2013

The name of the user that was used for this testing was "farmer2". This farmer is the owner of the field and sensor "0". Figure 4-12 presents the data that were collected over the course of the whole day that the demo took place. In this diagram, the spikes show the peaks in temperature due to trials by people that wanted to see if the system actually reacted when they try to warm-up the temperature sensor. The rest of the web page shows a custom rule and a table with the values that meet the rule's triggering requirements. The "A" labels on the diagram are annotations for the rule "A" that show that the rule was matched, corresponding in this case to when the measurement was more than 28^{o}C.


Ericsson Precision Agriculture Services
Click here to refresh the data below with new measurements
Plots
Temperature measurements over time for Ericsson Research Demo Field
Temperature
30
28
26
24
22
A
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
Time of Sampling

Data Analysis Report for farmer2

Rule-Based Analysis

Introduction

This section provides an analysis of the sensor readings using the ruleset provided by the user. The rules analyzed are shown below. In case a given period in the time series matches the rules below, it is reported and highighted in the plots

Brief Description of Rules

Annotation Letter	Rule Description	Value Range
A	Temperature greater than 28 degrees	From all measurements from all sensors in this field, select those with value greater than "28"

Overview of Measurements Conforming to Rules

Explicit values for rule "Temperature greater than 28 degrees"

Date and Time of reading	Sensor ID	Value
2013-12-05 10:57:12	0	28.25
2013-12-05 10:57:27	0	28.5
2013-12-05 11:36:58	0	28.625
2013-12-05 11:37:14	0	29
2013-12-05 11:37:30	0	28.875
2013-12-05 12:47:43	0	29.875
2013-12-05 12:47:59	0	30.875
2013-12-05 12:48:15	0	30.25
2013-12-05 12:48:31	0	29.375
2013-12-05 12:48:47	0	28.625
2013-12-05 14:43:28	0	28.125
2013-12-05 14:43:44	0	29.125
2013-12-05 14:43:59	0	29.625
2013-12-05 14:44:15	0	29.375
2013-12-05 14:44:31	0	28.625
2013-12-05 15:22:39	0	28.875
2013-12-05 15:22:55	0	28.125

Figure 4-12: Real time data presentation in Ericsson Research Day 2013

There were no problems during the demo and the system ran flawlessly. Presenting the operation of this prototype at such a conference and interacting with so many people, was very useful because it gave us a lot of insights and some hidden problems were discovered. Some of these problems and insights are presented as future work.

4.3.2 Value

As it was described in section 4.1, designing a system that works according to a generic rule is ineffective and would have limited or no value. The implementation of the precision agriculture system prototype that we have implemented should provide value in several different ways. The potential benefits of the usage of the prototype system are presented in order to extract the most valuable and relevant features.

First of all, operating such a system and attracting increasing numbers of users means that more and more data will be stored in the database. This large amount of data should enable the system to provide more information to these users. Big data is very valuable nowadays and with the appropriate usage of this database, very valuable information can be extracted. For example, experts can extract information which can provide input to new rules that could improve agricultural yields while also being sustainable.

A second potential advantage of the system is the building of a network that entities can use in order to collect information and statistics. News channels, ministries, meteorological companies, and insecticide companies are some of the entities that could be interested to use the network in order to retrieve information and thus reduce their costs or improve their own products.

Automatically storing the data in a database and providing the capability to present this data whenever and however the user wants, could be very useful for users who want to keep records of their cultivation schemes. For example, the combination of historical data with weather data could help farmers compare their yield in different years, enabling them to plan how they should act in the future.

In some countries, there are reinsurance companies that offer insurance for crops. For example, by applying this system with its historical information about conditions, these companies could predict if a potential disaster could happen due to weather conditions or identify that there is a bad batch of seeds that have been planted in one or more fields.

Another usage that could immediately be implemented along with the prototype monitoring system is a machine-to-machine (M2M) actuation system. The decision for M2M communication could be activated by rules that the user has set. For example, rules could match the current environment conditions to a set of set points, then trigger a water pump to irrigate a specific part of the field that needs watering. The water pump could also be shutoff via another trigger. This set of rules could save a lot of water, while making the watering process more effective; hence improving the overall sustainability of agriculture.

4.3.3 Business models

In this section, the possible business models are described. The business analysis was carried out by the team using the business canvas research method. In the following sections, the business models are presented.

4.3.3.1 HW and SW sales model

The main business model used for precision agriculture solutions is a model for hardware and software sales optionally combined with customer support services. This model dictates that farmers using the precision agriculture solutions should have high competence level in sensor installation and operation in accordance with knowledge regarding installation and analytics software usage.

From the supplier’s side this involves a lot of focus in R&D needs to be put on the user-friendliness of the solutions provided, taking into account the fact that the user is not sensor or analytics specialist.

4.3.3.2 Advisory services model

Trying to remove the need of the farmer having the competence to deploy and operate the solution, there can be another option. This option could be to work with agriculture consultancies that would communicate with farmers and manage the relationship with the farmer apart from deploying and operate the final solution. This option would change the role of the actual customer to be assigned to the agriculture consultancy.

The combination of the precision agriculture solution with the competence of the consultancy, results to a much more outcome driven model where the value proposition is not the hardware and software, but it resides in monitoring and alarming for problems, giving advice to the farmer.

The optimal offering in this scenario would be a performance based revenue model. In this model the advisor is in charge of maximizing yield, which can in other words means that the farmer can load some of the risk to the advisory services provider.

4.3.3.3 Machine manufacturer partnership

During our background research and interviews, we identified the fact that machines used in farming are becoming more and more intelligent. Many of the manufacturers, like the CLAAS, Deutz-Fahr and John Deere are using sensors on their equipment, trying to gather data related to yields and machine performance.

Analysis from this data, combined with the sensor data from a precision farming solution can provide better and more complete analytic capabilities. Thus, it would be an option to establish partnerships with machine manufacturers, trying to provide and an end-to-end solution for data gathered in farming.

Taking into account that the major cost of machinery can be made part of the deal and make it possible for the farmer to get financing included with machinery and analytics solution, this turns into a very tempting offer.

On the contrary, for the supplier of the sensor and analytics solution, the risk is always present since the lifecycle farming machinery is long. Taking this into consideration, it can a potential issue with uptake of the sensor and analytics business if this is the only business model chosen.

4.3.3.4 as-a-Service model

The precision agriculture solution can be potentially modeled as-a-service, where the initial cost for the farmer would include acquiring sensors and gateways and a monthly subscription fee for storage, data provision and analytics services provided.

The solution provider would be responsible for the creation and maintenance of a social platform, in order to enable farmers share their information, knowledge and experience.

This information exchange already takes place, but relies mainly on phone communication, as it is indicated by interview results.

During the interviews that took place, information extracted indicated that farmers mainly rely on variety of data sources for weather information, market data, crop advice etc. It is possible though, for the solution provider to gather all this data through analytics and then contextualize the information for the specific farmer’s situation.

Social clusters of nearby farmers or farmers with land in similar environments, crops and conditions could be brought together creating value for the farmer.

4.3.3.5 Combinations of the 4 main business models

If development of a sustainable business around precision agriculture solution is the final goal then none of the aforementioned business models would be sufficient.

On a primary level a basic HW and SW business model can help gaining experience in the real life field and feed an initial data set. The latter would help the analytics platform to be further developed. To support this early phase there could be a specific approach towards agriculture consultancies who would become users of the solutions to advice farmers.

After this initial phase and assuming that business is established, and farmers get used to the idea of precision agriculture, a more long term strategy can be implemented. Partnerships with machine manufacturers are more likely to happen and combinations of machinery data with sensor data can be stored and analyzed in the analytics platform, also establishing relationship with farmer in a long term manner.

5 Conclusions and Future work

This chapter includes a conclusion of the project along with the jobs that are not in the scope of this thesis and should be done in the future. After the future work, a part that describes the reflections of this project is included.

5.1 Conclusions

In this document, a solution for monitoring agricultural environments was presented. The system can act as an early warning system for upcoming threats, a monitoring system constantly reporting on the status of farms or livestock or as a recommendation system for prospective farmers. It is claimed that such a system is relevant in the network society and that there is sufficient technical knowledge (software and hardware components, standardized network protocols) to render its implementation not only feasible, but also cost effective. These claims are substantiated through a review of already-existing monitoring technologies including network protocols, open-source software and cheap hardware components which are used for the implementation and through a background study on the business value of such a system, a process which uses the business model canvas framework to suggest value propositions. A prototype system is presented, demonstrating its functionality for retrieving data from sensors, relaying these data through a gateway and storing and analyzing the data on a server. Subsequently, the results are presented to users via a web interface. The system features a custom sensor design for power efficiency, data encryption for security, cost effectiveness using off-the shelf, cheap components, as well as scalability end ease of use. The business relevancy established through the business model canvas approach backed by stakeholder interview sessions, coupled with the technical maturity of a basic, yet scalable and robust technical implementation provides the necessary momentum for a future real deployment somewhere in agricultural Sweden.

5.2 Future work

Since the project is a prototype that was developed under some limitations and in short time, there are some tasks that should be done in the future and would develop the system to a more mature state. These steps are described below.

The development of the platform board is necessary in order to make it more robust, thus manufacturing the board is important for the future progress of the system. A modular design that should give the opportunity to users of using energy sources, connectivity and sensors as modules could be a very useful and easy-to-use solution. A potential support of different platforms could also be an addition in the system that could spread the usage of the system with already applied solutions.

Regarding the communication between the components of the system, a server-to-platform communication stream should be implemented. This direction of communication is not implemented for this prototype but it would be important to be implemented for updating the firmware or variables on platform.

The analytics services should also include more complex tools. A schedule function should be implemented in order for the users to plan the frequency of the sensing for every sensor. A migration to the cloud for the server is important for the scalability of the system, as well. More complex tools like extending alerts to enable the use of functions, rather than simply local equations involving current sensor values could be a useful addition. These functions could even consider historic sensor values in order to identify patterns.

The most important and useful job that has to be done is the real field testing for extended time and with several sensor platforms and sensors deployed in fields. This will provide feedback that could be meaningful for the further development of the system and would include the users' insights and real needs.

5.3 Required reflections

This thesis project presents the implementation of a system that is designed and built with a primary goal: more effective and proactive agriculture cultivation. The use of low-cost equipment and open source software solutions, affects the economic aspect of the project and especially when considering the scalability of the potential usage. Furthermore, by using this system, natural resources and working hours are not wasted and thus, the agricultural economy and society are affected with profit. Regarding the ethical issues, in terms of privacy, the data that are generated and stored in the system are protected from the end-to-end security and the users can access only their own data. Considering the possibilities and limitations in work, the proposed future work can be used as a guideline for continuously developing the system.

References

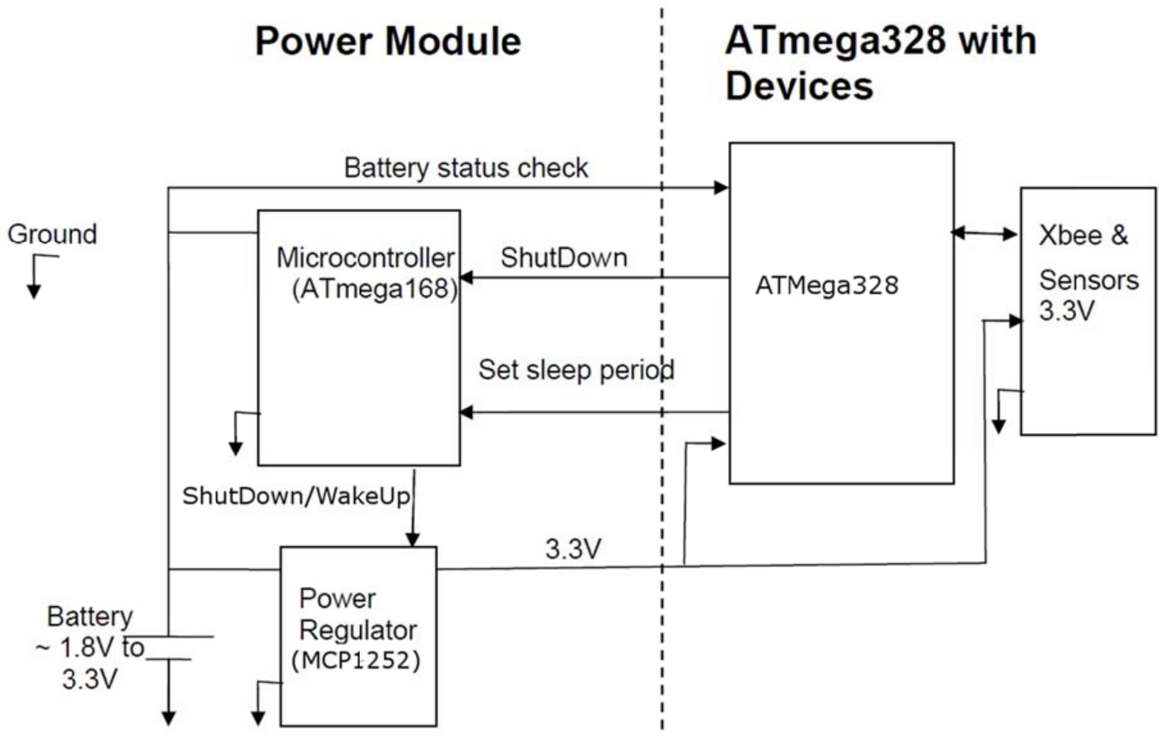
1. Xin Dong, Mehmet C. Vuran, and Suat Irmak, “Autonomous precision agriculture through integration of wireless underground sensor networks with center pivot irrigation systems.” Ad Hoc Networks, Volume 11, Issue 7, September 2013, Pages 1975–1987, DOI: 10.1016/j.adhoc.2012.06.012 Online. Available: http://www.sciencedirect.com/science/article/pii/S1570870512001291. [Accessed: 20-Sep-2013].
2. Robert Olsson, “A Practical Guide to Wireless IEEE 802.15.4 Networks”, Available: http://herjulf.se/products/WSN/sensors/wsn_practical_guide.html. [Accessed: 25-Sep-2013].
3. Texas Instruments, “6LoWPAN Sub1GHz Evaluation kit - CC-6LOWPAN-DK-868 - TI Tool Folder (Obsolete).” Online. . Available: http://www.ti.com/tool/cc-6lowpan-dk-868. [Accessed: 25-Sep-2013].
4. Freescale, “FRDM-KL25Z Product Summary Page.” Online. . Available: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FRDM-KL25Z&tid=vanFRDM-KL25Z. [Accessed: 25-Sep-2013].
5. Yiming Zhou, Xianglong Yang, Liren Wang, and Yibin Ying “A Wireless Design of Low-Cost Irrigation System Using ZigBee Technology.” In the proceedings of the International Conference on Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '09. (Volume:1), Wuhan, Hubei, DOI: 10.1109/NSWCTC.2009.231, 25-26 April 2009, pp. 572 - 575 Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4908331&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F4908190%2F4908191%2F04908331.pdf%3Farnumber%3D4908331. [Accessed: 19-Sep-2013].
6. M. Nesa Sudha, M.L. Valarmathi, and Anni Susan Babu, “Energy efficient data transmission in automatic irrigation system using wireless sensor networks.” Computers and Electronics in Agriculture, Volume 78, Issue 2, September 2011, pp. 215–221, DOI: 10.1016/j.compag.2011.07.009, Online. . Available: http://www.sciencedirect.com/science/article/pii/S0168169911001724. [Accessed: 19-Sep-2013].
7. “Contiki Hardware.” Online. . Available: http://www.contiki-os.org/hardware.html. [Accessed: 27-Sep-2013].
8. “Contiki: The Open Source Operating System for the Internet of Things.” Online. . Available: http://www.contiki-os.org/index.html. [Accessed: 16-Sep-2013].
9. “Sensor Model Language (SensorML) | OGC®.” Online. . Available: http://www.opengeospatial.org/standards/sensorml. [Accessed: 27-Sep-2013].
10. “SensorML Specification | OGC Network.” Online. . Available: http://www.ogcnetwork.net/SensorML_Spec. [Accessed: 16-Sep-2013].
11. “The GeoJSON Format Specification.” Online. . Available: http://www.geojson.org/geojson-spec.html. [Accessed: 20-Sep-2013].
12. “amee | Blog – AMON vs. SensorML/O&M.” Online. . Available: http://blog.amee.com/2012/04/27/amon-vs-sensormlom/. [Accessed: 20-Sep-2013].
13. “AMON v3.1”Online. . Available: http://amee.github.io/AMON/. [Accessed: 20-Sep-2013].
14. “Smart Objects Tutorial, IETF-80 Prague Introduction to Resource-Oriented Applications in Constrained Networks” Online. Available: http://www.iab.org/wp-content/IAB-uploads/2011/04/Shelby.pdf. [Accessed: 19-Sep-2013].
15. Z. Shelby, K. Hartke, and C. Bormann, “Constrained Application Protocol (CoAP)”, Internet-draft, Expires: December 30, 2013, June 28, 2013, draft-ietf-core-coap-18 Online. . Available: https://datatracker.ietf.org/doc/draft-ietf-core-coap/?include_text=1. [Accessed: 19-Sep-2013].
16. N. Kushalnagar, G. Montenegro, and C. Schumacher, ‘IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals’, Internet Request for Comments, vol. RFC 4919 (Informational), August 2007, Available at http://www.rfc-editor.org/rfc/rfc4919.txt.
17. “Wireless Sensor Networks Research Group.” Online. . Available: http://www.sensor-networks.org/?page=0823123150. [Accessed: 19-Sep-2013].

18. “IEEE SA - 802.15.4-2011 - IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs).” [Online]. Available: http://standards.ieee.org/findstds/standard/802.15.4-2011.html. [Accessed: 18-Feb-2014].
19. “ZigBee Implementation in Intelligent Agriculture Based on Internet of Things” EMEIT-12, part of series: Advances in Intelligent Systems Research, ISBN: 978-90-78677-60-4, Online. Available: http://www.atlantis-press.com/publications/aisr/emeit-12/index_emeit-12.html?http%3A//www.atlantis-press.com/php/paper-details.php%3Fid%3D3626. doi:10.2991/emeit.2012.408 [Accessed: 19-Sep-2013].
20. “protocol comparison.” Online. . Available: http://www.dresden-elektronik.de/funktechnik/wireless/ieee-802154/protocol-comparison/?L=1. [Accessed: 22-Sep-2013].
21. “telecombretagne/Arduino-IPv6Stack • GitHub.” Online. . Available: https://github.com/telecombretagne/Arduino-IPv6Stack/. [Accessed: 25-Sep-2013].
22. “telecombretagne/Arduino-pIPv6Stack • GitHub.” Online. . Available: https://github.com/telecombretagne/Arduino-pIPv6Stack. [Accessed: 25-Sep-2013].
23. A General Inductive Approach for Analyzing Qualitative Evaluation Data, American Journal of Evaluation, June 2006 27:237-246
24. “Ericsson Labs | Ericsson M2M Solution Supports Crab Farming in China.” [Online]. Available: https://labs.ericsson.com/blog/ericsson-m2m-solution-supports-crab-farming-in-china. [Accessed: 17-Feb-2014].
25. “RaspbianAbout - Raspbian.” [Online]. Available: http://www.raspbian.org/RaspbianAbout. [Accessed: 18-Feb-2014].
26. “Announcing the ADVANCED ENCRYPTION STANDARD (AES).” , Federal Information, Processing Standards Publication 197, November 26, 2001 [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf. [Accessed: 18-Feb-2014].
27. “FIPS 81 - Des Modes of Operation. CIPHER BLOCK CHAINING (CBC) MODE, APPENDIX C” [Online]. Available: http://www.itl.nist.gov/fipspubs/fip81.htm. [Accessed: 18-Feb-2014].
28. “RSA Laboratories - PKCS #5: Password-Based Cryptography Standard.” [Online]. Available: http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-5-password-based-cryptography-standard.htm. [Accessed: 18-Feb-2014].
29. “Oracle HTTPS for Client Connections.” [Online]. Available: http://docs.oracle.com/cd/B14099_19/web.1012/b14013/ohttps2.htm. [Accessed: 18-Feb-2014].
30. “Device Cloud: Driving the Internet of ANYthing.” [Online]. Available: http://www.etherios.com/products/devicecloud/. [Accessed: 27-Sep-2013].
31. “A scalable database for a remote patient monitoring system – Ruslan Mukhammadov” [Online]. Available: http://www.divaportal.org/smash/get/diva2:637470/FULLTEXT01.pdf [Accessed: 27-Sep-2013]
32. “MySQL:: The world’s most popular open source database.” [Online]. Available: http://www.mysql.com/. [Accessed: 22-Feb-2014].
33. “OpenStreetMap.” [Online]. Available: http://www.openstreetmap.org. [Accessed: 27-Sep-2013].
34. “Leaflet - a JavaScript library for mobile-friendly maps.” [Online]. Available: http://leafletjs.com/. [Accessed: 27-Sep-2013].
35. “JSON vs XML: How JSON Is Superior To XML.” [Online]. Available: https://www.udemy.com/blog/json-vs-xml/. [Accessed: 24-Feb-2014].
36. "More than 50 billion connected devices – taking connected devices to mass market and profitability" Ericsson white paper 284 23-3149 Uen | February 2011 [Online]. Available: http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf. [Accessed: 22-Oct-2013]
37. “RXTX Library for Serial Communication”. [Online]. Available: http://rxtx.qbang.org/wiki/index.php/Main_Page. [Accessed: 27-Sep-2013].
38. “Arduino Playground - OneWire.” [Online]. Available: http://playground.arduino.cc/Learning/OneWire#. [Accessed: 25-Feb-2014].
39. Maxim Integrated Products, Inc., “Analog, linear, and mixed-signal devices from Maxim.” [Online]. Available: http://www.maximintegrated.com/. [Accessed: 25-Feb-2014].
40. “bildr » One Wire Digital Temperature. DS18B20 + Arduino.” Friday, July 8th , 2011 [Online]. Available: http://bildr.org/2011/07/ds18b20-arduino/. [Accessed: 25-Feb-2014].

41. Digi International Inc., “Digi XBee® Wireless RF Modules - Digi International.” [Online]. Available: http://www.digi.com/xbee/. [Accessed: 25-Feb-2014].
42. “DS18S20 High-Precision 1-Wire Digital Thermometer” 19-5474; Rev 8/10, August 2010 [Online]. Available: http://datasheets.maximintegrated.com/en/ds/DS18S20.pdf . [Accessed: 19-Mar-2014]
43. Maxim Integrated Products, Inc., “Overview of 1-Wire Technology and Its Use - Tutorial.” [Online]. Available: http://www.maximintegrated.com/app-notes/index.mvp/id/1796. [Accessed: 25-Feb-2014].
44. Microchip, “Low Noise, Positive-Regulated Charge Pump - 21752a.pdf.” [Online]. Available: http://ww1.microchip.com/downloads/en/devicedoc/21752a.pdf. [Accessed: 26-Feb-2014].
45. “Representational State Transfer (REST) “,Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000 [Online] Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm . [Accessed: 24-Mar-2014].

Appendix A Schematics of third prototype

In Appendix Figure A-1, the schematic of the third prototype platform is presented as designed by Maxim Teslenko.



Appendix Figure A-1: Schematic of Third prototype

Appendix B Power Consumption calculation for third prototype

The following calculations are based on measurements of the platform consumption that were carried out by Maxim Teslenko.

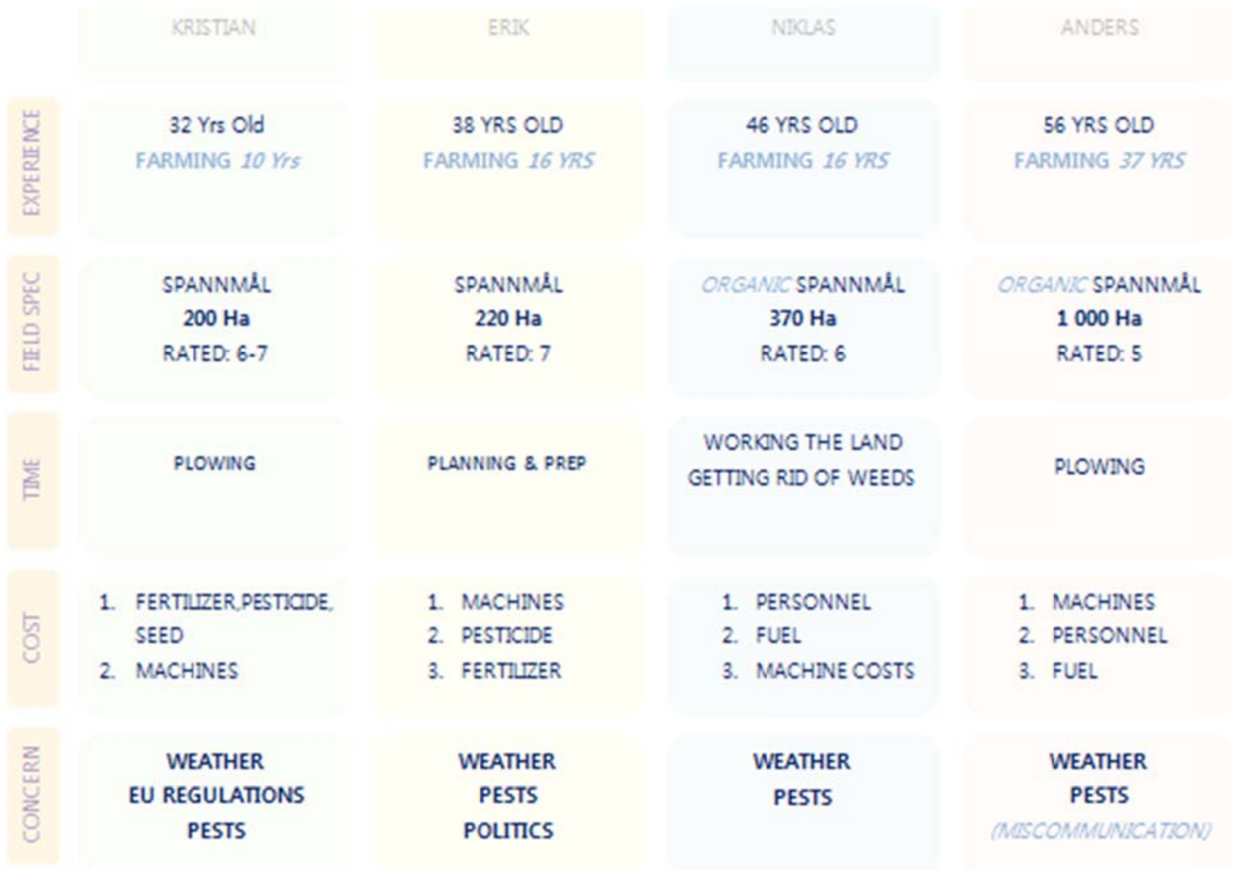
The consumption of the platform is shown below:

- With Xbee on, Thermometer active: **139mA** (note that due to voltage drop on multi-meter of approximately 1V effective voltage on device terminals was 2.7V)
- With Xbee on, Thermometer disconnected: **137mA** (note that due to voltage drop on multi-meter of approximately 1V effective voltage on device terminals was 2.7V)
- With Xbee disconnected, Thermometer on: **30mA**
- With Xbee disconnected, Thermometer disconnected: **28mA**
- In sleep mode in all configurations: **4µA** (note it is 0.004mA)

If we assume 1 second activity of the board **every 20 minutes**, we get on average over active/sleep cycle (1s*139mA+1200s*0.004mA)/1201s=0.12mA current consumption by the whole system. For the pack of 6 AA batteries it gives 5400mAh/0.12mA=45000hours or **5.1 years**. 2 AA batteries have 3 times less energy giving us approximately **1.7 years of operation**. Increasing sleep time has a limited effect on the lifetime of the system. Even if the system is always asleep battery lifetime only doubles. On the other hand, any reduction of sleep time by factor X effectively reduces battery lifetime by the same factor X.

Appendix C Farmer interview summary

The interview was carried out by Nina Washington and pertains to farmers growing spannmål, organic and regular, farming land in Vikbolandet, situated east of Norrköping, Sweden.



	KRISTIAN	ERIK	NIKLAS	ANDERS
EXPERIENCE	32 Yrs Old FARMING *10 Yrs*	38 YRS OLD FARMING *16 YRS*	46 YRS OLD FARMING *16 YRS*	56 YRS OLD FARMING *37 YRS*
FIELD SPEC	SPANNMÅL **200 Ha** RATED: 6-7	SPANNMÅL **220 Ha** RATED: 7	*ORGANIC* SPANNMÅL **370 Ha** RATED: 6	*ORGANIC* SPANNMÅL **1 000 Ha** RATED: 5
TIME	PLOWING	PLANNING & PREP	WORKING THE LAND GETTING RID OF WEEDS	PLOWING
COST	1. FERTILIZER, PESTICIDE, SEED 2. MACHINES	1. MACHINES 2. PESTICIDE 3. FERTILIZER	1. PERSONNEL 2. FUEL 3. MACHINE COSTS	1. MACHINES 2. PERSONNEL 3. FUEL
CONCERN	**WEATHER** **EU REGULATIONS** **PESTS**	**WEATHER** **PESTS** **POLITICS**	**WEATHER** **PESTS**	**WEATHER** **PESTS** *(MISCOMMUNICATION)*

Appendix Figure C-1: Farmer interview summary

Appendix D Storage Server code

```
import java.io.*;
import java.net.*;
import java.nio.charset.Charset;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.util.Date;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import javax.net.ssl.SSLServerSocket;
import javax.net.ssl.SSLServerSocketFactory;
import javax.net.ssl.SSLSocket;

class storageServer {

    static int listeningPort = 9876;
    static byte[] keyData = new byte[] { 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2,
3, 4, 5, 6, 7, };

    public static void main(String args[]) throws Exception {
        if (args.length == 0) {
            System.out.println("Usage: \nUDPServer [u|x|t]"
                    + "\nExplanation follows:"
                    + "\n'u' is unencrypted packet retrieval"
                    + "\n'x' is encrypted packet retrieval"
                    + "\n't' is packet retrieval through an SSL tunnel");
            return;
        }

        else {
            new DBAccess().readDataBase();

            if (args[0].compareTo("u") == 0) {
                DatagramSocket serverSocket = new
DatagramSocket(listeningPort);
                byte[] receiveData = new byte[1024];

                while (true) {
                    DatagramPacket receivePacket = new DatagramPacket(
                            receiveData, receiveData.length);
                    serverSocket.receive(receivePacket);
                    String sentence = new String(receivePacket.getData());
                    System.out.println("RECEIVED: " + sentence);
                    DBAccess dao = new DBAccess();
                    dao.insertMeasurement(sentence);
                }
            } else if (args[0].compareTo("x") == 0) {
                DatagramSocket socket = new DatagramSocket(listeningPort);
                byte[] receivedData = new byte[1024];
                Cipher aesCipher;

                {
                    while (true) {
                        try {
                            aesCipher = Cipher
                                    .getInstance("AES/CBC/PKCS5Padding");
                            IvParameterSpec zeroIV = new IvParameterSpec(
                                    new byte[aesCipher.getBlockSize()]);
                            SecretKey key = new SecretKeySpec(keyData,
"AES");

                            aesCipher.init(Cipher.DECRYPT_MODE, key,
zeroIV);

                            DatagramPacket packet = new DatagramPacket(
                                    receivedData, receivedData.length);
                            socket.receive(packet);
                            final byte[] plaintextBinary =
aesCipher.doFinal(
                                    packet.getData(), 0,
packet.getLength());

                            String plaintext = new String(plaintextBinary,
                                    Charset.forName("UTF-8"));
                            System.out.println("RECEIVED: " + plaintext);

                            DBAccess dao = new DBAccess();
                            dao.insertMeasurement(plaintext);
                        } catch (Exception e) {
                            if (socket != null) {
                                socket.close();
                            }
                            System.out
                                    .println("Exception while receiving
packet !");

                            System.out.println("Error:\n" + e);
                            break;
                        }
                    }

                }
            }

            else if (args[0].compareTo("t") == 0) {
                System.out.println("Configuring server for SSL ...");
                System.out
                        .println("Caution: In SSL mode, always start server
before any sensor "
                                + "transmission otherwise SSL connection
handshake will fail!");
                SSLServerSocketFactory sslserversocketfactory =
(SSLServerSocketFactory) SSLServerSocketFactory
                        .getDefault();
                SSLServerSocket sslserversocket = (SSLServerSocket)
sslserversocketfactory
                        .createServerSocket(listeningPort);
                System.out
                        .println("Done with configuration, waiting for
incoming connection at port "
                                + listeningPort);

                final String[] enabledCipherSuites = {
"SSL_DH_anon_WITH_RC4_128_MD5" };

sslserversocket.setEnabledCipherSuites(enabledCipherSuites);

                while (true) {
                    serveIncomingConnections(listeningPort,
sslserversocket);
                }

            } else {
                System.out.println("Usage: \nUDPServer [u|x|t]"
                        + "\nExplanation follows:"
                        + "\n'u' is unencrypted packet retrieval"
                        + "\n'x' is encrypted packet retrieval"
                        + "\n't' is packet retrieval through an SSL
tunnel");
                return;
            }
        }
    }

    private static void serveIncomingConnections(int port,
            SSLServerSocket sslserversocket) {

        try {

            SSLSocket sslsocket = (SSLSocket) sslserversocket.accept();

            InputStream inputstream = sslsocket.getInputStream();
            InputStreamReader inputstreamreader = new InputStreamReader(
                    inputstream);
            BufferedReader bufferedreader = new BufferedReader(
                    inputstreamreader);

            String string = null;
            while ((string = bufferedreader.readLine()) != null) {
                System.out.println("RECEIVED: " + string);
                System.out.flush();
                DBAccess dao = new DBAccess();
                dao.insertMeasurement(string);
            }
        } catch (Exception ex) {
            System.out.println("Connection Exception!");
        }
    }

}

class DBAccess {

    private static Connection connect = null;
    private static Statement statement = null;
    private static ResultSet resultSet = null;
    static String url = "jdbc:mysql://localhost:3306/agridb";
    static String username = "root";
    static String password = "712MFD43";

    public void insertMeasurement(String sentence) throws SQLException {

        String measurement_type = "Temperature";
        String unit = "Celsius";
        Date date = new Date();
        Timestamp timestamp = new Timestamp(date.getTime());
        System.out.println(timestamp);

        // The Temp value is retrieved
        String[] tokens = sentence.split(" ");
        int tokenCount = tokens.length;
        String strTemp = tokens[tokenCount - 1];
        String sensor_num = tokens[tokenCount - 2];
        // System.out.println("string: " + strTemp);

        float decTemp = Float.parseFloat(strTemp); // decTemp is the
temperature
                                                   // in float format
        System.out.println("The float temp is: " + decTemp);
        String insertQuery = "INSERT INTO measurements(frequency ,
sensor_id , timestamp , measurement_type , unit , value)"
                + " VALUES (null , '"
                + sensor_num
                + "' , '"
                + timestamp
                + "' , '"
                + measurement_type
                + "' , '"
                + unit
                + "' , '"
                + decTemp + "')";
        System.out.println(insertQuery);

        try {
            System.out.println("Connecting database...");
            connect = DriverManager.getConnection(url, username, password);
            System.out.println("Database connected!");
        }

        catch (SQLException e) {
            throw new RuntimeException("Cannot connect the database!", e);
        }

        Statement statement = connect.createStatement();
        int insertCount = statement.executeUpdate(insertQuery);
        System.out.println(insertCount + " inserts were done.\n");

        System.out.println("Closing the connection.");
        if (connect != null)
            try {
                connect.close();
            }

            catch (Exception e) {
                System.out.println("Exception: " + e.toString());
            } finally {
                close();
            }

    }

    public void readDataBase() throws Exception {
        try {
            System.out.println("Loading driver...");
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("Driver loaded!");

            try {
                System.out.println("Connecting database...");
                connect = DriverManager.getConnection(url, username,
password);
                System.out.println("Database connected!");
            }

            catch (SQLException e) {
                throw new RuntimeException("Cannot connect the database!",
e);
            }

            statement = connect.createStatement();
            setResultSet(statement.executeQuery("select * from
measurements"));
            writeResultSet(getResultSet());
        } finally {
            System.out.println("Closing the connection.");
            if (connect != null)
                try {
                    connect.close();
                }

                catch (Exception e) {
                    throw e;
                } finally {
                    close();
                    if (connect != null)
                        try {
                            connect.close();
                        }

                        catch (Exception e) {
                            if (!(e instanceof SQLException)) {
                                throw e;
                            }
                        } finally {
                            close();
                        }
                }
        }
    }

    public void writeMetaData(ResultSet resultSet) throws SQLException {
        // Now get some metadata from the database
        // Result set get the result of the SQL query

        System.out.println("The columns in the table are: ");
        System.out.println("Table: " +
resultSet.getMetaData().getTableName(1));
        for (int i = 1; i <= resultSet.getMetaData().getColumnCount(); i++)
{
            System.out.println("Column " + i + " "
                    + resultSet.getMetaData().getColumnName(i));
        }
    }

    public static void writeResultSet(ResultSet resultSet) throws
SQLException {
        // ResultSet is initially before the first data set
        while (resultSet.next()) {
            String frequency = resultSet.getString("frequency");
            String sensor_id = resultSet.getString("sensor_id");
            Date timestamp = resultSet.getDate("timestamp");
            String measurement_type =
resultSet.getString("measurement_type");
            String unit = resultSet.getString("unit");
            String value = resultSet.getString("value");
            /*
             * System.out.println("Frequency: " + frequency);
             * System.out.println("Sendsor id: " + sensor_id);
             * System.out.println("Timestamp: " + timestamp);
             * System.out.println("Measurement type: " + measurement_type);
             * System.out.println("Unit: " + unit);
System.out.println("Value:"
             * + value);
             */
        }
    }

    // You need to close the resultSet
    private static void close() {
        try {
            if (getResultSet() != null) {
                getResultSet().close();
            }

            if (statement != null) {
                statement.close();
            }

            if (connect != null) {
                connect.close();
            }
        }

        catch (Exception e) {

        }
    }

    public static ResultSet getResultSet() {
        return resultSet;
    }

    public static void setResultSet(ResultSet resultSet) {
        DBAccess.resultSet = resultSet;
    }

}
```

TRITA-ICT-EX-2014:25