

Analysis and Improvement of a Remote System

- *Case study: Agricultural Machinery* -

MASTER THESIS

SSE4, Spring 2008.

María Isabel Díez del Val
Faith Oziofu Ogini Nielsen
Elisa Oteo Ovejero



Aalborg University
Department of Computer Science

TITLE:

Analysis and Improvement
of a Remote System

THEME:

Distributed Systems
and Semantics

PROJECT PERIOD:

01/02/2008 - 29/05/2008

PROJECT GROUP:

d606a

GROUP MEMBERS:

María Isabel Díez del Val
Faith Oziofu Ogini Nielsen
Elisa Oteo Ovejero

SUPERVISORS:

Alexandre David
Morten Kühnrich

NUMBER OF COPIES: 6

NUMBER OF PAGES: 94

CONCLUDED: 29/05/2008

SYNOPSIS:

The purpose of this master thesis is to design and implement a solution that provides access to data in a remote mobile device. The remote device in this case is part of an agricultural machinery and, it is able to switch between two independent networks; depending on factors like availability, bandwidth or cost. We further analyze the performance of the system by evaluating the transferred data from the device to another node, this will provide us with detail knowledge about the cost of making the transfers over wireless mediums. Also, we evaluate the time spent for switching between the independent networks, and this is further used for improving the switching speed.

Acknowledgments

Studying abroad is always hard: a different University, a different language, in general a different environment; that is the reason why I will be always grateful to all those that have been supporting me from Spain, no matter the distance and, of course, to the new people I have met in Denmark that have shown interest in the development of this Master Thesis. To all them, *tak*.

María Isabel Díez del Val

I would like to express gratitude to my entire family for their continuous love and support throughout my study. I am also thankful to Aalborg University for giving me the opportunity to acquire more knowledge by working with various research projects. Finally, I would like to say a big thank you to everyone that has supported me one way or another, I did benefit from all the discussions, guidance and wonderful supports.

Faith Oziofu Ogini Nielsen

I would like take a moment and remember my friends in Spain who have always been there for me, despite the distance. And also all of the wonderful new friends I have made here in Denmark. I will never forget you. And of course, most importantly, I want to thank my family for making it possible for me to be here, having one of the best experiences of my life. Thank you so much for all the love and support you have given me.

Elisa Oteo Ovejero

Finally, we all would like to thank our supervisors Alexandre David and Morten Kühnrich for the many helpful comments made throughout this project.

Aalborg, 29th May 2008

Summary

During this work, wireless access has been provided to a remote mobile device. This solution allows access to both real time data and logged data in a device that is part of an agricultural machinery.

To reach this objective, a server and database architecture was developed for providing and controlling access to the system. Moreover, a wired and GPRS connection was used for our implementation in order to provide Internet access for the device. GPRS was chosen as the wireless base for our implementation because it is both a wireless link and it has also provided a means for investigating mobility with the mobile device.

Furthermore, it was chosen to develop a model that will allow connection switch between the two links in use by the device. In this phase, different switching combinations have been made possible.

A number of tests have been conducted for the wired and wireless links used by the device, and results concerning data rate, bandwidth, uplink rate and downlink rate are presented.

Based on the above developments, the results show that remote access to data in the device is achievable, logging, compression and transfer of the logged data from the device is also possible, as well as switching between two contemporary networks. Moreover, considerable level of security is provided in the system.

Contents

1	Introduction	1
1.1	The LandIT Project	2
1.2	LYKKETRONIC	2
1.3	Overview of last semester project	3
1.4	Goals	3
1.5	The Embedded Computer	4
1.6	Outline	4
2	Software Requirements Specification	5
2.1	Scope	5
2.2	Definitions, acronyms and abbreviations	6
2.3	Overall Description	7
2.3.1	Product perspective	7
2.3.2	Product functions	7
2.3.3	User characteristics	8
2.4	Constraints	9
2.5	Specific Requirements	9
2.5.1	External Interfaces	9
2.5.2	Logical database requirements	9
2.5.3	Security	9

3	Wireless Technologies	11
4	Analysis	15
4.1	The system's from an end user's perspective	15
4.1.1	Use - Case Diagram	15
4.1.2	Use - Case Specification for realtime access	17
4.2	Intercommunication within the system	18
4.2.1	Network analysis	19
4.2.2	Data transmission pattern	19
4.3	Data collection/representation	20
4.3.1	Data representation using an Entity - Relationship diagram	20
4.3.2	Parameters to be logged	23
4.4	Security	23
5	Design	25
5.1	<u>Part I: Initial Design Ideas</u>	25
5.1.1	A system to extract data from a CANBus network	25
5.1.2	A system that allows the EC to use static IP address	26
5.1.3	Design for logging data in the EC	26
5.2	<u>Part II: Final Chosen Design</u>	28
5.2.1	Wireless technologies chosen	29
5.2.2	Why System Access Via an ACS	29
5.2.3	Details of the inner components of the architecture	31
5.2.4	Final Design for Logging data in the EC	32
5.2.5	Design to switch between WIFI and GPRS in the EC	34
5.2.6	Relational data model design at the ACS	36
5.2.7	Security Design	38

<i>CONTENTS</i>	xi
6 Implementation	41
6.1 Database Implementation	42
6.2 Intercommunication	44
6.2.1 Log, compression and transfer from the EC	45
6.2.2 Connection switch in the EC	48
6.2.3 Web Applications	50
6.2.4 System Security	53
6.3 Implementation problems and suggested solutions	54
6.3.1 WIFI dongle installation:	54
6.3.2 Shared memory access from Apache Web Server	55
6.4 System Configuration	56
6.4.1 ACS configuration	56
6.4.2 EC configuration	57
7 Tests and performance results	59
7.1 Test approach	60
7.2 The Test Setup	61
7.3 Test cases	62
7.3.1 Test Case I: System Startup	62
7.3.2 Test Case II: Connection Switch	66
7.3.3 Test Case III: Logs	68
7.3.4 Test Case IV: Viewing a job in the EC	69
7.4 Performance Study	72
7.5 Data transfer costs	73
8 User Guide	77
8.1 System setup	77
8.2 How to operate the system	78

8.2.1	Superuser	82
8.2.2	Technician	86
8.2.3	Farmer	87
8.2.4	Worker	87
9	Conclusion and Future Work	89
9.1	Main improvements in the present project	90
9.2	Future work	91
A	Details about the Embedded Computer	95
A.1	Hardware - Level Description	95
A.2	Software - Level Description	96
A.3	Human Machine Interface (HMI)	96
A.4	Examples of data to be logged in the EC	98
B	Scripts' source listings	99
B.1	Programs and configuration files on the EC	99
B.1.1	wvdial.conf	99
B.1.2	SshTools.py	100
B.1.3	info.conf	101
B.1.4	connectionswitch.py	102
B.1.5	ConnectionTools.py	106
B.1.6	launchall.py	109
B.1.7	wirewatchdog.py	110
B.1.8	configuration.h	112
B.1.9	main.c \mapsto Executable: getValues	113
B.1.10	sftpdialogue.sh	114
B.1.11	logging.c	115

B.1.12	sudoers	123
B.2	Programs and configuration files located on the ACS	124
B.2.1	porttesting.py	124
B.2.2	storingMySQL.c	126
B.2.3	xml2mysql.php	129
B.2.4	login.php (obfuscated version)	131

Chapter 1

Introduction

Today's agricultural practices require improved productivity and efficiency. A farmer's ability to save time and cut costs is essential in order to be able to compete in both the local and global market. Existing wired communication technology cannot yet meet all the needs of getting communication and information at anytime and anywhere. However, the integration of the Internet and wireless communication techniques make many applications change from ideal to reality, such as remote monitoring and maintenance services. We have seized this opportunity to implement a system that allows remote access to data in an embedded computer that is part of a farm tractor.

Although embedded systems were usually considered dedicated devices, they have capabilities which make them able to perform different tasks; furthermore, they have communication abilities that can guarantee the possibility of remote control, supervision and management. Nowadays, the trend in this sort of applications is to use a distributed architecture, since the advancement and spread of Internet technologies allow that electronic devices and systems can be connected together. This eases access to them by using web browsers, and it also makes it possible to exploit other tasks, such as automated control/monitoring, reporting and setting up of a device remotely by simply pressing a button.

This project is centered around all these ideas and also in switching between the communication channels in use by the embedded computer, and ensuring security in the entire system.

1.1 The LandIT Project

Our work is within the frame of the LandIT project, that is an industrial collaboration with AAU (Aalborg University). It is aimed to build technologies for communication and data integration between farming devices and other farming-related IT systems, both for operational and business intelligence purposes. Tekkva Consult is the project coordinator and the partners are [2]:

- Center for Embedded Software Systems (CISS)
- Skov Inc. (Climate Control for Stables)
- LYKKETRONIC Inc. (Devices for control and monitoring of field machines)
- The Federation of Contractors
- The Danish Agricultural Advisory Service

Our project is going to be specially designed for LYKKETRONIC.

1.2 LYKKETRONIC

The company LYKKETRONIC, which was founded in 1978, develop manufacture and implement electronic measuring, monitoring and control systems for tractors and machineries used for agriculture and forestry. The main goal of LYKKETRONIC is to provide customers within the sector of agriculture and forestry with machineries and technologies to help increase efficiency in their field work [16].

This company has provided us with an Embedded Computer(EC) which is part of a tractor system. The tractor consists of several sensory units; data from the different units concerning the activities of the tractor are sent through the system's BUS to the EC. The model of the EC is MICROSPACE-PCX48 made by the Swiss company DIGITAL-LOGIC. It is a protected industrial PC for in-vehicle computing and it is compatible with any standard PC and runs with common operating systems, such as Windows and Linux [1]

Currently, accessing and updating of data in the EC is via a touch screen directly connected to it. LYKKETRONIC wishes to add a functionality to the embedded computer, which is the ability to remotely access data stored in the EC from any location.

1.3 Overview of last semester project

Our past work [3] proposed a solution in which users are able to remotely access data of a farm tractor via an Access Control Server (ACS)¹. A user located anywhere in the world can access data about the activities of a tractor via the ACS using the Internet. This solution also makes remote control of the device possible. In addition, it presented an automatic switch between different types of network used by the embedded computer with limited functionalities.

The ACS provides an interface that requires the users to authenticate themselves in order to gain access to data of a tractor. Nevertheless, it did not distinguish different kinds of users, so every authenticated user had access to the same set of information.

The solution from the past work only provides access to realtime data in the embedded computer. But, there is also need to log data about the activities of the tractor over time, and this is intended to be used for long-term system evaluation.

1.4 Goals

The aim of the current project is to use the Internet as a medium for providing access to realtime data in a remote embedded computer. Also, LYKKETRONIC wishes to add another functionality that will provide access to logged data from different tractors, and this will be stored in a centralized location. Hence, we shall log data in the embedded computer and transfer the logged data to another node using the available communication channels. Because the coverage of wireless technologies can sometimes be limited, the embedded computer is expected to switch between two independent networks based on certain factors like availability, bandwidth, or cost. Efforts will be made to improve the time spent for switching between the different networks in use by the embedded computer in order to make it faster. Measurements and comparisons of the size and time to transfer data, and the cost of transferring the data from the embedded computer to a different node will also be taken and documented.

¹In the previous work, the ACS was known as “Proxy”. This name has been replaced by “Access Control Server”. The old name was changed, because this node does not entirely perform all the tasks of a Proxy, such as accumulating and saving files that are most often requested by the users in a special database, called “cache”, in order to increase the speed of users’ connection to the system.

1.5 The Embedded Computer

As mentioned in section 1.2, LYKKETRONIC has provided us with an embedded computer, which from now on we will refer to as (EC)². The EC is a MICROSPACE PCX48 that is designed to operate in harsh environments. Below is a picture of the EC. Its concrete hardware, software and parameter details can be found in Appendix A.



(a) Front view of the Embedded Computer (b) Back view of the Embedded Computer

Figure 1.1: The Embedded Computer

1.6 Outline

This section contains an outline of the overall structure of the report.

- Chapter 2 begins with a description of the system's requirement specification.
- Chapter 3 presents a short review of wireless technologies.
- Chapter 4 presents an analysis of the system.
- Chapter 5 presents the design in two parts, Part I contains some of our initial design ideas and Part II describes the chosen design.
- Chapter 6 presents an implementation of the chosen design, the system configuration and some of the problems we encountered with suggested solutions.
- Chapter 7 presents an analysis of the system performance with tests.
- Chapter 8 presents a user guide for the system.
- Chapter 9 concludes.
- Appendix contains source code.

²In the previous work, this device was referred to as Blackbox(BB). This name has been changed in this present paper to EC. This is because the name BB can be confused for a device whose internal workings are not understood by the users.

Chapter 2

Software Requirements Specification

We have chosen a subset of the items specified in the IEEE Recommended Practice for Software Requirements Specifications (Std 830-1993 [12]) that is relevant for stating the requirements of the system. The intention of the software requirements specification is to comprehensively and concisely define functional and non-functional requirements that the present project should be compliant to.

2.1 Scope

The main aspect of this project is providing access to information relating to a variable number of tractors located anywhere in the world. Any user with the right privilege is allowed to remotely access data of a device, so long as the device is connected to the Internet. It is also possible to remotely modify some parameters in the tractor using a web interface, this is necessary for the purpose of remote maintenance. Each device is connected to the Internet via a wireless medium, in this environment, the device should be able to choose the network to use based on factors like availability, cost etc. of connecting to a network at any given moment.

2.2 Definitions, acronyms and abbreviations

Access Control Server (ACS). A server is a computer system or an application program which provides services for connected clients. In our case, a client connects to the Access Control Server(ACS) to request for some services, for example, information about a tractor via a web page. The ACS provides the resource to the client. Clients are also able to access log files stored in the ACS.

CAN Bus. *Controller Area Network*, it is a high - speed serial data communications bus for real time control applications.

Contractor. A company that owns fleet of tractors, that can be hired out to different farms in different regions or locations.

Data. Data in each tractor or farm belongs to different farm owners and it may have private contents. For this reason, it is necessary to impose security to protect these data.

Farm. The group that use the tractors for working in their fields, they can choose to either own the tractors or hire the tractors they need from the contractors.

Firewall. Dedicated appliance, or software running on another computer, which inspects network traffic passing through it, and denies or permits packet passage based on a set of rules.

N.A.T. Network Address Translation (NAT, also known as Network Masquerading, Native Address Translation or IP Masquerading) is a technique of transceiving network traffic through a router that involves re-writing the source and/or destination IP addresses and usually also the TCP/UDP port numbers of IP packets as they pass through. Most systems using NAT do so in order to enable multiple hosts on a private network to access the Internet using a single public IP address.

Resources. Resources here are heterogenous and geographically spread and hence require solutions that considers these varieties and differences.

SSH. A Network protocol that allows data to be exchanged over an encrypted channel between two computers. Encryption provides confidentiality and integrity of data. SSH uses public-key cryptography to authenticate the remote computer and allows the remote computer to authenticate the user, if necessary. SSH is typically used to log into a remote machine and execute commands. It also supports tunneling, which is forwarding arbitrary TCP ports, it can also transfer files using the associated SFTP protocol. A SSH server, by default, listens on the standard TCP port 22.

Tractor. A farm vehicle used for working in the farm fields, it can be used for planting, spraying, etc. In our project, an EC is attached to each tractor.

Users. Persons authorized to interact with the system. They are geographically spread, they require ease of use, authorization and authentication. They need to access several tractors and they also require assured quality of service.

2.3 Overall Description

2.3.1 Product perspective

One of the main objectives of this project is the implementation of a system that provides access to remote data in an EC from anywhere. It is required that the data can be accessed in real time and also be logged for later use. Both realtime data and logged data has to be transferred over a wireless channel and this is made possible by using the “available” and “better” connection at any given moment; we say that a connection is available if it is always ready for use at any given point in time, and we say that a connection is better than another if its average cost is lower in terms of money and its transmission speed is higher.

2.3.2 Product functions

- Related with the embedded computer:
 - The system must allow remote access to it.
 - The system must provide automatic switching between different networks since a tractor can be in different locations at different times and the coverage of wireless technologies can be limited or expensive.
 - The system must regularly log data in a standard format that can easily be used by other applications.
 - The system must be able to send logged data to another node
 - The system must transfer the files in a compressed format from one node to another.

- Related with the ACS:
 - The system must store the logged data in the ACS.
 - The system must take care of administrative issues, these include:
 - * Registration, deletion and modification of the different users and their information.
 - * Registration, deletion and modification of different farms and their information.
 - * Addition of tractor to a farm, removal of tractor from a farm.

2.3.3 User characteristics

For accessing data in the system, four groups of users must be distinguished: farm owners, technicians, tractor drivers (workers) and a superuser. The users can do some actions on the system depending on the privileges defined for the group they belong to. These group of users are described below as the primary actors of the system:

Farmer. This is a farm owner, he can access information about the farm which he owns, the tractors working in the farm, and the drivers of his tractors(workers). He has access to information in an EC on the tractor belonging or working in his farm. He must be able to add, modify and delete all these elements.

Worker. This is the driver of a tractor to which the embedded computer is attached. He can access the information of the EC in the tractor he is driving.

Technician. This is the user that only has access to technical information about the EC. He has READ - WRITE access and his modification can change the behavior of this device.

Superuser. This is the system administrator. He has access READ - WRITE access to information of all the other users and also the filesystem.

2.4 Constraints

- The EC system must run in Linux CentOS.
- The EC will use a dynamic IP address; using a static IP address is not an option because the system is required to be used in different places and at different times.
- A network like HSPA would have been a good option. However, LYKKETRONIC has considered this technology not useful at the moment for the purpose of this project because farmers using their equipment are often located in rural districts far from urban neighborhood where HSPA coverage is available at the moment.

2.5 Specific Requirements

2.5.1 External Interfaces

In order to access real time data in a working tractor, Internet connection must be available in the tractor. The end-user must have a browser to display the web application with which to access data in the EC. By using a browser, an end user is not only capable of having remote access to the system, but can also remotely change specific parameters in the system.

2.5.2 Logical database requirements

- A user can have more than one address, electronic mail or telephone numbers.

2.5.3 Security

- The system must authenticate the users before allowing access to it.
- Sensitive data like passwords must be stored using some form of encryption.
- Data communication between the remote end and the EC must be encrypted.
- Secure versions of network protocols must be used.

Chapter 3

Wireless Technologies

Information gathering, processing and distribution is an important area of concern in Computer Science. Mobility has altered the mode of computing for most computer users. The Internet depends on protocols that were designed and optimized to perform well in wired networks. As an example, the Internet Protocol (IP) was not designed to be used in wireless networks in which mobile devices can change their point of connection to the Internet as they move. As a result of this, Mobile IP has been proposed to create an efficient and scalable mechanism for mobile devices to roam within the Internet. Another area where wireless networks can pose a problem is at the physical layer; here the probability of loss, delay, low bandwidth, etc. is high in comparison to wired networks, which can result in transmission errors.

Lots of services that require the use of the Internet need to always be connected at all times, and as a result of this a lot of portable devices like laptops, PDAs, hand-held computers, etc. require regular Internet connectivity. The connection of such devices to the Internet are commonly via wireless means in order to take full advantage of their portability.

There exist a lot of wireless standards today that can be used in different areas of wireless connectivity. Each of these standards offer different wireless technologies and each of these technologies offer advantages in either bandwidth or range. However, other parameters such as availability, reliability, quality of service (QoS), cost, etc. are also relevant when comparing the different technologies.

Some examples of existing technologies include WLAN, GSM/GPRS, HSPA, EDGE, WiMax, etc. One thing that is however common with these wireless technologies is that they all extend support for Internet protocols, and as mentioned earlier on, the Internet protocols can pose some problems while using them in wireless networks. These problems can either be as a result of a wireless channel experiencing packet loss or as a result of mobility associated with the nodes.

A WLAN is a wireless local area network that links two or more computers without using wires. It provides a transfer speed of up to 54Mbps with a range of up to 150 meters. Users of WLAN however have a constraint of un-guaranteed access to WIFI (see Std. 802.11 [13]) hot spots at all times they need Internet access. In such a situation, an alternative could be the use of a cell phone's modem combined with its sim card to access the Internet. This is done by using a data cable that will connect the mobile phone and a laptop or a hand held computer. Another choice could be the use of a GSM modem; which is particularly built for this purpose, with an inserted GSM sim card. Here, the GSM modem will be connected directly to a USB port on the computer. There is not much difference between both approaches in terms of support for data service because the efficiency of data transmission is determined by the transmission rate of the wireless network in use. GSM is a popular standard for mobile phones around the world[10]. Its ubiquity makes international roaming very common between mobile phone operators, thus enabling subscribers to use their phones in many parts of the world. This standard allows network operators to offer roaming services that allow subscribers to use their phones on GSM networks all over the world. A wireless data service that is available with almost every GSM network is GPRS.

GPRS refers to a high-speed packet data technology that guarantees constant connection. It can allow data transmission speed of up to 40 kbit/s and in most cases have worldwide coverage. It supports the widely used Internet Protocol (IP). Hence, a mobile phone modem combined with a GPRS sim card can provide constant access to the Internet. Part of the services provided by GPRS include Internet browsing where data transfer is typically charged per megabyte of transferred data. GPRS was developed so that GSM operators can meet up with the growing need for wireless data services that has grown as a result of the growth of the Internet.

Internet applications are characterized by demands like bursty traffic patterns, high throughput needs, etc. These demands are not well suited for circuit switched networks like GSM.

In order to take care of these demands, GPRS service was developed; this service is characterized by its packet switching nature which is able to handle bursty traffic patterns. This service also provides asymmetric uplink and downlink bandwidth.

Below is a list of some key differences in a comparison between GPRS and WIFI in terms of bandwidth, coverage, roaming, cost, and security.

- **Bandwidth:** By comparison, WIFI technology provides more bandwidth than GPRS, hence WIFI provides more data rate transfer resulting in higher throughput in WIFI than in GPRS.
- **Coverage:** This is an important requirement for data transfer customers that wish to access the Internet from multiple locations. By comparison, a GSM signal can provide a widespread coverage of several miles which can allow true mobility. WIFI on the other hand can only provide coverage of up to 150 meters from an access point. Thus, WIFI cannot compete in the area of coverage with GPRS.
- **Roaming:** Minimum coverage cause the need for roaming. WIFI coverage in cities or in a country is not near that of the cellular technology. On the other hand, GSM supports extensive regional and international roaming for voice and data. Although some WIFI operators now support roaming but the roaming coverage is nowhere near that of the GSM network.
- **Cost:** The only cost incurred in WIFI connection is the subscription fee, the connection is free in most cases. In addition, WIFI can be used for ad-hoc point-to-point connection. On the other hand, data communication over GPRS is billed per megabytes of transferred data and this can become expensive for use when sending large amount of data.
- **Security:** WIFI devices use security standards such as Wireless Equivalent Privacy (WEP) [14] to overcome the technology's vulnerabilities. These standards can however only protect data if they are used, and many WIFI users do not turn these security features on because they are difficult to configure. By comparison, the GPRS network is more secure than WIFI network because of the security features built into it. Its automatic end-to-end security features, makes users not to have to configure them manually.

A newer wireless technology called (High Speed Data Access) HSPA (a 3G technology)¹, makes it possible to receive large files and increase the speed of communication from mobile devices. The biggest advantage of this technology is that it is possible to send and receive data at rates up to four times higher than those in current GSM/GPRS networks (up to 384 Kb/s). The HSPA technology is quite similar to WiMax (see Std. IEEE 802.16 [15]) which is another wireless technology that is aimed at providing wireless data over long distances. WiMax is intended for wireless metropolitan area networks, and it is aimed at allowing higher data rates over longer distances as it combines the benefits of broadband and wireless together. Unlike the HSPA technology that is already available for use, WiMax technology is expected to be available in the near future.

This project requires a network like HSPA or WiMax, and choosing to use either of the two technologies would have been the best option. But, we are however constrained from using either of the two technologies because to use services provided by HSPA technology, it is necessary to be in an area covered by 3G signal. Also, devices with WiMax implementation are not yet on the market at the moment, hence there is no efficient network coverage for this technology yet.

¹It refers to a third generation of development in wireless technology that is designed to work over wireless air interfaces such as GSM, EDGE, TDMA and CDMA. It is aimed to keep people connected at all times and in all places.

Chapter 4

Analysis

Our project starts with a phase of problem analysis that will be helpful in correctly analyzing the system choices, by clearly defining the functions we must consider and find out in detail, in order to meet the specified requirements.

4.1 The system's from an end user's perspective

4.1.1 Use - Case Diagram

In this section we use the Use Case model to show the interaction between a user and the proposed system. This model is based on the Software Requirements Specification. The Use Cases are a representation of the future system that will allow us model the user requirements. Next diagram is the Use Case model of our system which shows the actors and actions that are carried out in the system.

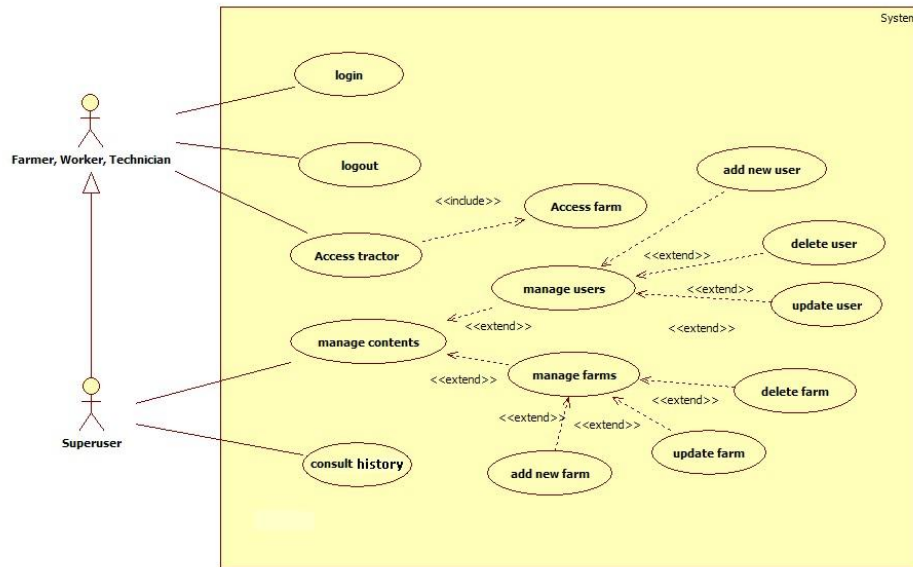


Figure 4.1: Use Case Diagram

We use include dependencies whenever one use case needs the behavior of another. An extended dependency is a generalization relationship where an extended use case continues the behavior of a base use case.

4.1.2 Use - Case Specification for realtime access

Below is one of the important use-case specification shown in figure 4.1. We have used only one, which is “Access tractor” for making this description. The nomenclature used are as follow: **Description** - describes the particular use-case. **Precondition** - states the prerequisite that must be satisfied for this use-case to be fulfilled. **Normal sequence** - describes the different steps in the particular use-case, and **Exception** - is to describe how to handle any failure that can occur in Normal Sequence in this case, for steps 4 and 5.

UC - 1	Access tractor	
Description	The system must behave as described in the present use case when a user wants to access real time data in a tractor.	
Precondition	The user must be previously authenticated in the system.	
Normal Sequence	Step	Action
	1	The system shows the list of all the farms the user has access to.
	2	The user chooses one of the farms on the list.
	3	The system displays the tractors related with the farm that was selected.
	4	The user chooses one of the tractors that has been listed before.
5	The system will show a new web application to access tractor data.	
Exceptions	Step	Action
	4	If there is no tractor online, this use case finishes.
	5	If there is a loss of connection, this use case finishes.

4.2 Intercommunication within the system

Since a tractor is a mobile entity, its ability to connect to the Internet will be made possible via a wireless medium. For a user to access data in a tractor, the user is also required to be connected to the Internet. Because the coverage of wireless technologies can sometimes be limited, the choice of at least two wireless mediums will be necessary. The choice of switching between the two wireless mediums as stated in the requirement will depend on whether a cheaper or faster connection is available.

The initial idea is presented in the picture below.

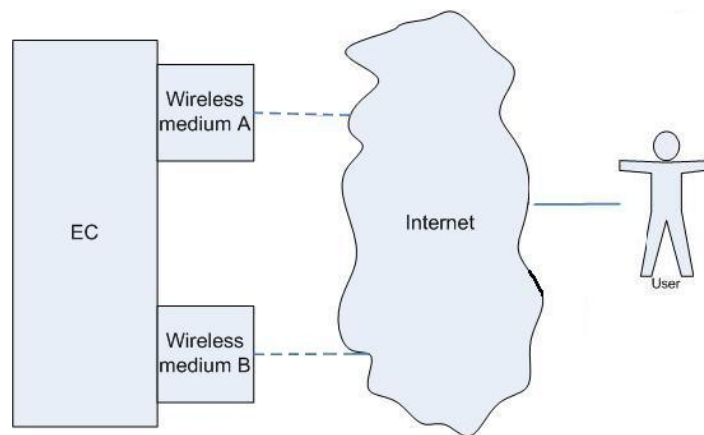


Figure 4.2: Abstract intercommunication system

Furthermore, access to data in an EC will be provided in two different forms namely: real time data access and logged data access.

4.2.1 Network analysis

Table 4.1 shows an initial network measurement made at the beginning to know the status of the GPRS network. It is to know how much data we are able to transfer in average using this network.

1. Mobile phone	
Model:	Nokia N71 GPRS class 10
SIM	CBB Mobile
GPRS download speed	200Kbps, 25Kb/sec transfer rate
GPRS upload speed	70Kbps, 8.8Kb/sec transfer rate
2. Data cable	
Interface:	DKU-2 USB data cable
Speed	460800 baud
3. Mobile computer(EC)	
Model:	Celeron 800mhz with 256MB RAM
Operating system:	CentOS 4.4, Kernel 2.6
Software packages:	pppd, wvdial

Table 4.1: Current network and device status

4.2.2 Data transmission pattern

Before going into further technical details, we will first present the behavior of the old system in terms of data transfer in the system. To do this, we analyzed the Apache Web Server, `access.log` (where all the data transferred over a period of time can be seen) for a representative work load. The results can be seen in the next diagram.

We can see a repetitive sequence with high peaks whenever a new page is transferred. This is because of the repeated transfer of some common parts for all the pages of the web application. The repetitive sequence is probably as a result of the transfer of a common part for each transfer from the system.

The measurement of the network information was taken at different intervals and the results varied for the different measurements. This variability of the throughput rates is as a result of radio conditions, and bandwidth fluctuations over time in GPRS networks.

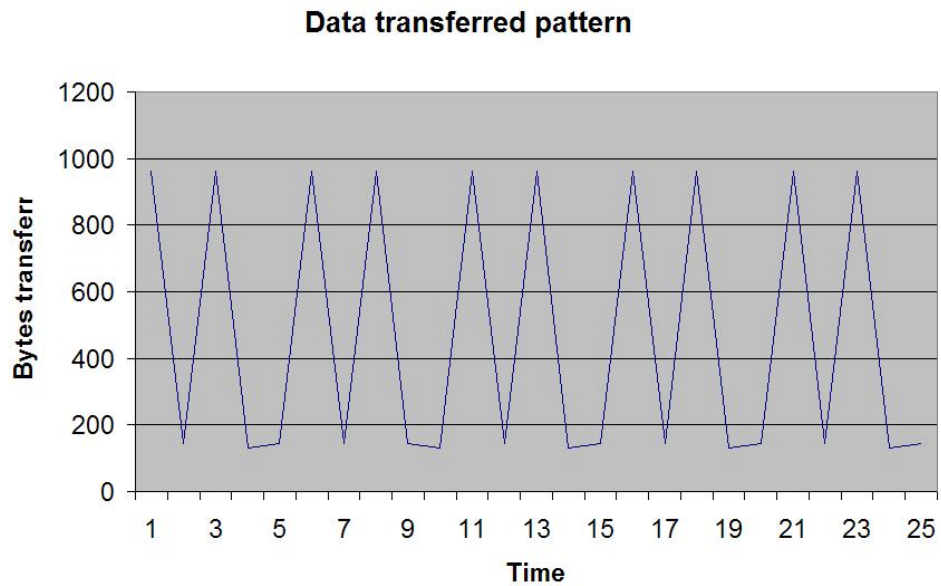


Figure 4.3: Pattern of data transfer

4.3 Data collection/representation

4.3.1 Data representation using an Entity - Relationship diagram

After finding out what the entities involved in the system are and thinking about how they are related, we can state:

Contact information. This entity represents data belonging to a user, i.e, his email, phone and address.

Farm. This another entity represents each of the farms that will use a tractor with the device. We will store the address, the contact information and the name of each of them.

Log. This is a registration of the different actions that have taken place in the system, which involves storing the date and the time in which any of the actions occurs.

Role. This represents each of the privileges a user can have in our system.

Tractor. This entity represents the machine to which each of the embedded computers is part of. We will have the tractor's name, serial id, MAC

address, the port through which it will make its connection, and the IP address of the node it should connect to in order to register itself.

User. This entity represents each of the persons that are authorized to use the system. We will store their username, password, firstname and lastname.

Variable. This represents each of the parameters that can be logged in the embedded computer. It is characterized by its name, value, units of measurement, and the time in which the value was taken.

Their relationships are:

belongs to. A farm can have many tractors, but a tractor can only belong to one farm.

has. A user can have multiple contact information, but a contact information belongs to one and only one user.

logs. A “tractor” registers the value of different parameters about the working of the tractor. A given value belongs to one and only one tractor, but a tractor can log multiple values for multiple variables.

plays. A user plays one and only one role in the system, but the same role can be played by many users.

related with. A user can be related with many tractors and viceversa.

stores. A user can register information about the action he has made in the system.

On next page, we present the entity - relationship diagram of our application. Its most peculiar characteristic is that a user is directly related with a tractor, and not with a farm, this is why we have to know, for example, in the case of workers, the tractors they are in charge of.

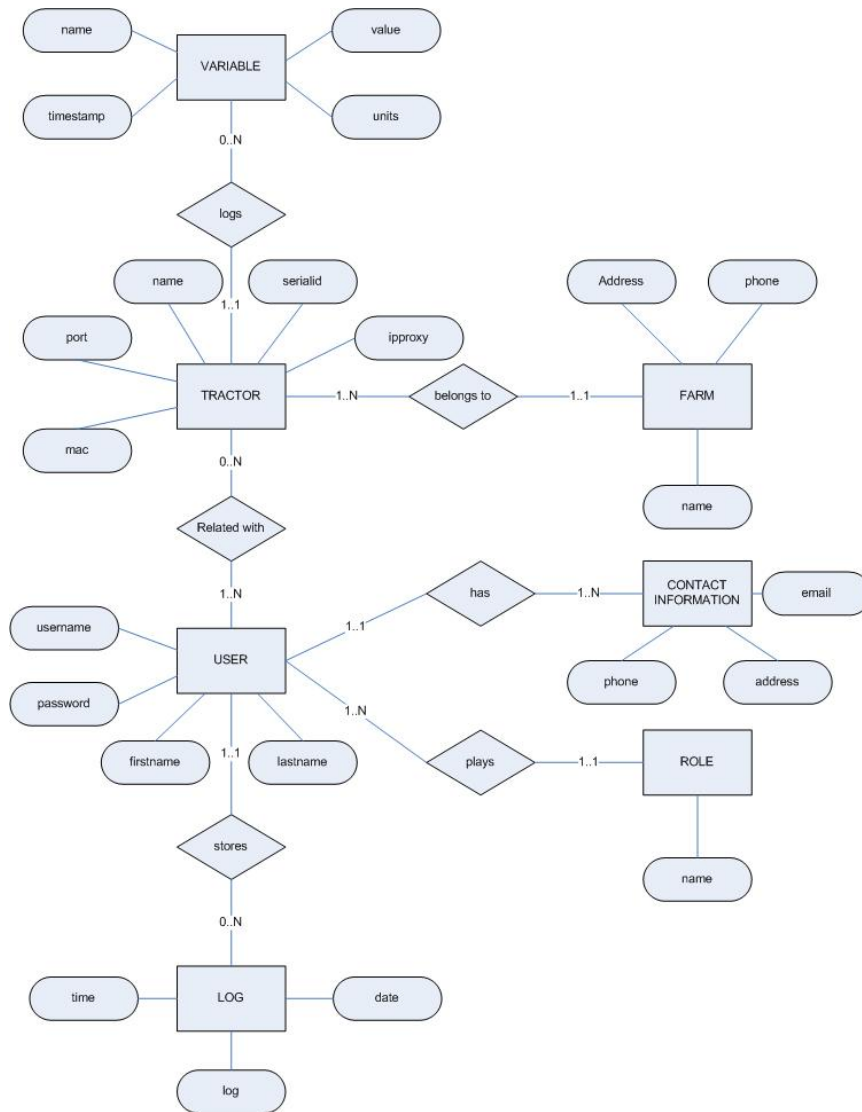


Figure 4.4: Entity - Relationship diagram

4.3.2 Parameters to be logged

For this project, we have selected a subset of parameters that will be used for logging; these parameters will also be used in the experiments. They are a subset of all the variables that can be read from a shared memory using an API. We will assume that these are the set of data whose values need to be logged in the EC, because of its potential interest for the system's users. The variables are;

Speed, Total trip, Battery voltage, Engine hours, Fuel level, Time to service, Tank content, Application rate, Current output, Spray pump, Spray line pressure, Total area, Amount of fertilizer, and Distance.

4.4 Security

Since data access in the embedded computer will be made via an Internet connection, we must take into account, first of all, what elements/contents of the system need protection, against what and, at this stage of the project, the general idea is to avoid, or minimize, this sort of problems.

For a hacker, the most interesting information in our system are those related with confidential information about the users such as user's login, password, or information related with the working of a given tractor. This sort of information should be stored or transmitted in such a way that a non-authorized user can not understand it, i.e, in some form of encryption.

Furthermore, it must not be possible in any way to make a ping or connect via SSH or SFTP to an embedded computer directly from outside. The objective is that nobody can see the state of a tractor (connected/not connected to Internet) if he is not authorized to, and of course, avoid the access to its internal information.

We must state at this point that the firewall of our system was already implemented in the previous semester and there are no evidence that it must be modified in the present work.

Chapter 5

Design

This chapter states how to achieve the requirements analyzed previously. The design is in two parts; part I describes some of our initial design ideas and part II describes the final chosen design.

5.1 Part I: Initial Design Ideas

We have looked at possible design solutions for the system. The main objective of the system to be developed in this project is to provide access to both realtime data and logged data in any working tractor.

We will now take a brief look at some of the previous design ideas we have considered and why they were not chosen. The first is a system that can extract data from a CANBus network, and the second is a system that can allow access to an EC by directly connecting to its public IP address. Next, is a design solution for logging data in the EC.

5.1.1 A system to extract data from a CANBus network

The idea of this design was to directly extract raw sensor data (using CANBus protocol) from a CANBus that is connected to the EC. In this approach, the extracted data will be organized and stored inside the EC. Functionalities will then be made to allow users access the stored data in the EC. This design was however not achievable because at the beginning of the project there was no means of reading data directly from the system's CANBus. Also, data in most part of the system were encrypted.

5.1.2 A system that allows the EC to use static IP address

Another design idea was a system that allows direct connection to the EC via a public IP address. In this case, each EC will have a static IP address with which users can connect to it. This idea was not chosen because the tractors are not expected to remain in a particular farm, a tractor can be moved to anywhere and at anytime because different farms will hire them. When a machine with a static IP address is moved to a different subnet, it will not work until it is reconfigured. Also, this approach will require the EC to use two fixed IP addresses; one for each of the wireless mediums in use.

5.1.3 Design for logging data in the EC

One of the main tasks in this work is to log data in a working tractor. In order to achieve this requirement, a possible design solution was the use of an API provided by LYKKETRONIC to read data from shared memory in the EC. With this API, the value of read data can be gathered from time to time, depending on the algorithm defined for doing this.

At this point, we first need to answer two questions, “what exactly will the logged parameters be used for? ” and “how often are these parameters expected to change?” This is because the value of some of the parameters change very often and this can affect the frequency with which we log such data, and this in turn depends on what the logs will be used for. Some other parameters are not likely to change very often, hence such parameters do not require regular log because logging them very often will result in repeated values for those parameters, which can be regarded as redundancy.

Logging data by using a compare function

The idea here is to use the parameters to read values from a shared memory and obtain their values.

A copy of the last read values are kept. Next time the parameters are read, a “compare function” is used to compare values of the new read with those of the last read. Only the values of the parameters that have changed since the last read will be written to the file.

The purpose of making a comparison is to ensure that only the changed values are written to the file. It is also to reduce the size of data written in each file. But, a careful study of the frequency with which each of the values are changing, combined with what the logged values will be used for, can reveal how often these values will change and when to store them. Hence, with enough information about what the logged data will be used for and how often the values are changing, it will not be necessary to make comparisons before storing each time we read the values. Also, with a good compression algorithm, the written files can be well compressed before sending.

Logging data in binary format

The idea here is to read the values of a given set of parameters from shared memory. The values will be written in binary format instead of using ASCII. This method will allow larger data set to be logged in each file because of its binary format. But, in order to use this method, it will be required to also have a corresponding program at the receiving side, to take care of converting the received binary files into either ASCII format or a standard format like XML for further use. Also, the EC requires enough space to store all the files before transfer.

Having described some of the design alternatives we thought about earlier on and their strong and weak sides, we will now look at the chosen design solution we intend to implement.

5.2 Part II: Final Chosen Design

The figure 5.1 shows a general system architecture of the proposed solution that will be used for accessing realtime data and logging data in an EC.

This consists of six sub-parts namely:

1. The GSM/GPRS Network
2. The Ethernet/WIFI Network
3. The Internet
4. An Access Control Server - ACS
5. The Embedded Computer - EC
6. Users

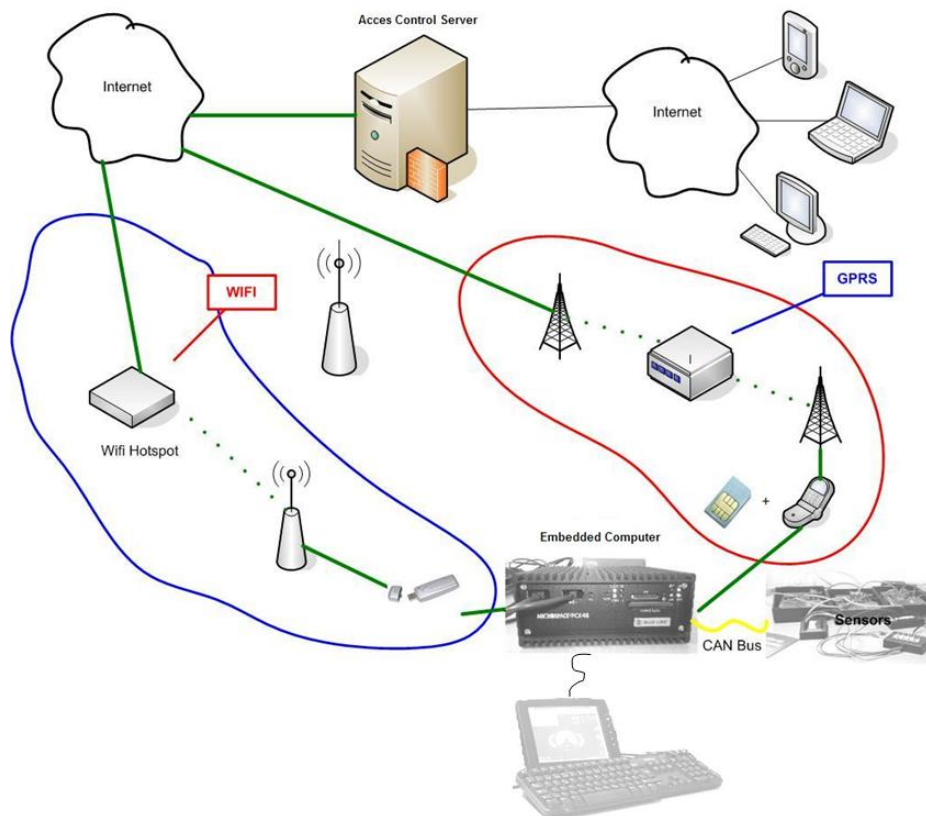


Figure 5.1: System Architecture

The previous diagram shows the components of the system and the relationships among them. Inner workings of each component of the system will be explained in the next subsection of this chapter.

5.2.1 Wireless technologies chosen

Even though HSPA technology can provide increased downlink and fast uplink for data transfer, LYKKETRONIC has considered this technology inappropriate at the moment for the purpose of this project because, as we already said, farmers using their equipment are often located in rural districts far from urban neighborhood where HSPA coverage is available at the moment. What we need is a network that can provide high bandwidth and long range like WiMax, and this technology should have been the best but it also has its constraints which are: devices with WiMax implementation are not yet on the market for the public, the implementation of this technology may vary from country to country and there are no efficient network coverage yet. So, for the present work, two technologies WIFI and GPRS will be considered. Detailed comparison between these two technologies can be found in page 13. So, the EC is expected to connect to the Internet using a WIFI or GPRS network.

5.2.2 Why System Access Via an ACS

We will now state the reasons for choosing the ACS architecture as a solution. We have chosen to use an architecture that will provide access to a mobile resource like the EC via an Access Control Server ACS. This architecture is intended to provide access to both realtime data and logged data of an EC so long as the EC is connected to the Internet. Currently the EC will connect to the Internet using a WIFI or GPRS network. For every connection it makes to the Internet, it will use either of the two interfaces to acquire a dynamic IP address. In our system, any router's NAT the EC is connected to will make it impossible to see the EC from the outside.

The ACS is thus meant to act as a master to all the connected ECs. Every EC that gets connection to the Internet has to report to its master (the ACS). Whenever an EC gets a connection to the Internet, it makes this report by creating an SSH tunnel from itself to the ACS and, as long as the EC is online, every data will be sent to and from it inside this tunnel. This means that it is the EC that will initiate an SSH connection to the ACS, and this helps to solve the problem of not knowing how to reach an EC that is online or connected to the Internet. With this solution, when an EC is online, realtime requests can be sent to and received from it, even while it is inside a LAN.

The ACS is also meant to act as a gatekeeper for whoever wishes to get data from the EC. In order to access any data in the system, a user must be authenticated at the ACS first. Also, the ACS controls access to the different tractors, it uses the access rights defined for the different user groups for granting access to the tractors.

Furthermore, it is designed to monitor the state of each EC continuously in order to update their appearance to the end users as online or offline.

The next diagram presents an scenario where we can see how the access to an EC should be made. It is an overview of the interaction between a user and the system.

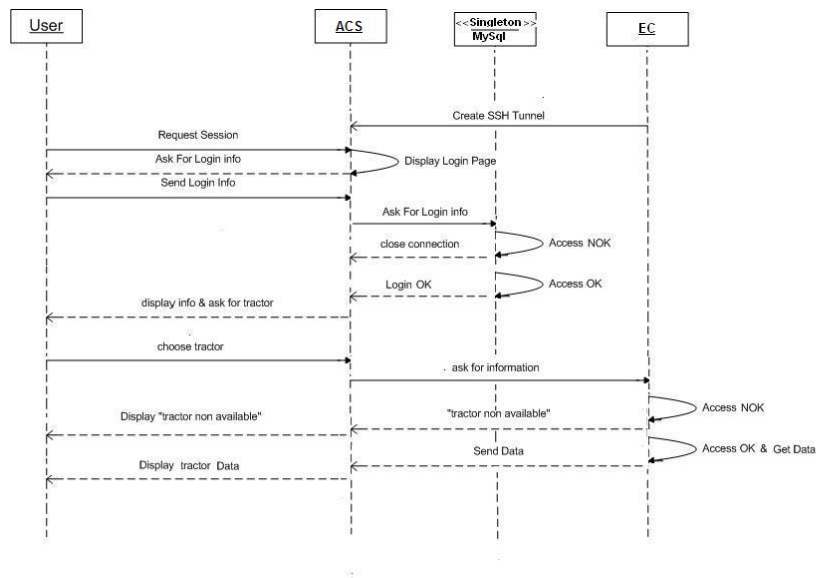


Figure 5.2: Sequence diagram of the Use Case “Access tractor”

From the diagram, the first step is for the EC on the tractor to make itself available by initiating a connection to the ACS. The user has to log in before asking for data from the EC. If the user is allowed to interact with the system, then the user’s demand for data is relayed by the ACS to the EC if the EC is available. After that, the EC responds to the ACS and data is then transmitted to the user. With the aid of this scenario, we are able to understand the connection between a user, the ACS and the EC.

5.2.3 Details of the inner components of the architecture

Having identified the main components of the proposed system, we shall now see the construction and explanation of the inner working of each of the components.

The GSM/GPRS Network. This component consists of a mobile phone with in-built modem, a GPRS sim card, and a cable to connect the EC to a phone, in order to provide Internet connection via a GSM network.

The WIFI Network. For now, an Ethernet cable is being used instead of WIFI; the USB WIFI dongle available for this project is a NETGEAR wg111T.

The Internet. For our setup, this will be the medium through which all data communication will be made.

The Access Control Server. This is categorized into two parts namely: hardware and software with the following configuration.

- The hardware: A Linux-based desktop computer.
 - Processor Intel Pentium 4 2.80 GHZ
 - Cache size: 512 KB
 - Hard Disk capacity: 17 GB
- The software:
 - Linux Ubuntu version: 7
 - Apache Web server version: 2.0.52
 - MySQL database version: 5
 - PHP version: 5
 - Web pages for login and data administration
 - Security certificate

The Embedded computer. Hardware and software details about the EC can be found in Appendix A.

5.2.4 Final Design for Logging data in the EC

In this design idea, the parameters to be logged are grouped into two categories. The first category are those set of parameters that are required to be logged more often because they change with high frequency. The second category consists of those parameters that do not require regular logging. This is because these set of parameters will not change very often, and regularly logging them will result in redundant data. So, the data variables to be logged are grouped the into two categories - “Often” and “NotVeryOften”. Each group of data is read and logged according to the unit of time defined by the logging function in the EC. The time units determine how often to read and write the data for the two groups. It also defines when to stop writing and close a file. Once the time defined by the function is reached, the file for the data being logged is closed and then compressed in the EC. Here is a list of the parameters used for logging the variables. As mentioned earlier, this is just a subset of the entire variable set in the system.

Often	Not very often
Speed	Battery Voltage
Total trip	Engine hours
Fuel level	Driving time
Tank content	
Application rate	
Current output	
Spray dump	
Spray line pressure	
Total area	
Amount of fertilizer	
Distance	

Table 5.1: Categorization of parameters to be logged

Formats for logging and compressing data in the EC

We thought about two possible ways of recording the data in our logs: plain text or XML. We chose to use XML because of its portability. The XML files will be compressed into bzip2 before sending to the ACS.

The two methods of file compression are lossy and lossless. Lossy compression applies lossy data compression to data. When this is used, the data can never be recovered exactly as it was before it was compressed. Lossless compression works by finding repeated patterns in the data and encoding the pattern in an efficient manner. Since lossy compression cannot yield the exact original data after decompression, it is not an ideal method of compression for critical data such as textual data. For this reason, we have chosen to use a lossless data compressor, bzip2.

bzip2 is a freely available, patent free and it is a high quality data compressor[4]. It typically compresses files to within 10% to 15% of the best available techniques (the PPM family of statistical compressors¹) whilst being around twice as fast at compression and six times faster at decompression. Among its main advantages, we can state that:

- The good quality of its compression reduces long distance network traffic.
- It supports recovery from media errors.
- It is portable. It runs on any 32 or 64 - bit machine with an ANSI C compiler. The distribution should compile unmodified on Unix and Win32 systems.

It is worth highlighting that when using compression algorithms, the difference between standard and maximum compression levels is small, especially when you consider the extra CPU time necessary to process the extra compression passes.

¹PPM is a specialized form of compression based on Markov Modeling. In general, PPM predicts the probability of a given character based on a given number of characters than immediately precede it. PPM is conceptually simple, but often computationally expensive[8].

Figure 5.3 shows the logging process in our system.

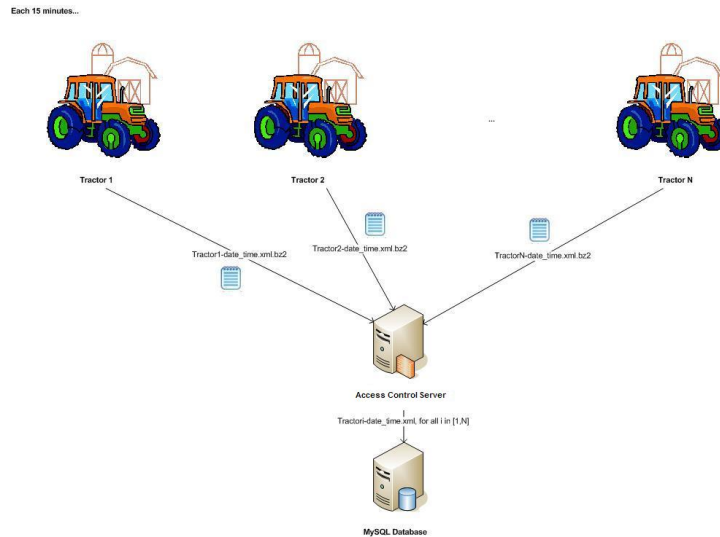


Figure 5.3: Logging process

5.2.5 Design to switch between WIFI and GPRS in the EC

Another requirement of the system is to be able to switch between the two networks in use. The design idea chosen is aimed at monitoring the available or cheapest connection in terms of cost and bandwidth, and either stop or start a second connection depending on its availability and cost.

In order to make the switching between the two networks, we would need at least one script running in the background. This script will be in charge of constantly monitoring the availability of the wired and GPRS connection. Depending on this result, corresponding tunnels will be created or closed. It also involves an update of a database in the ACS as appropriate for each type of connection in use.

It works by checking whether there is a change in the state of the connected GPRS modem and the wired cable. If the wired connection is available, it starts up a wired connection. If it is not, it checks for GPRS modem, if this is available, it starts up the GPRS connection. If it cannot find the GPRS interface at this point, it means no connection is possible, then it will remove all the existing connection information by closing any existing ssh tunnels, programs used for bringing up the interfaces and/or updating the database in the ACS. As can be seen in this description, the wired connection always has priority when it is available because of its lower price and higher bandwidth.

The switching process works as shown in the following UML activity diagram.

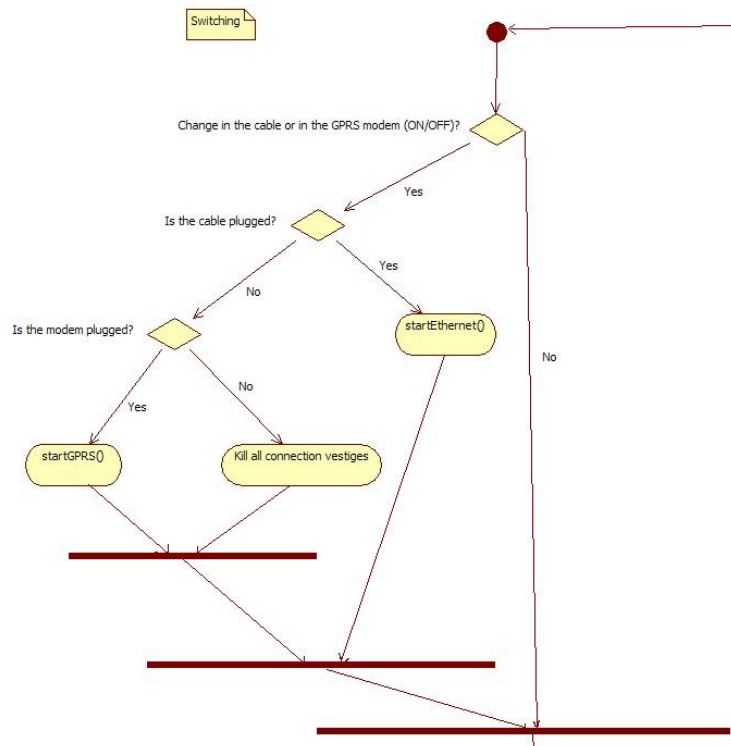


Figure 5.4: Switching Activity Diagram

5.2.6 Relational data model design at the ACS

Below is a list of the tables that is implemented in the next phase of our project. The attributes that are underlined represent the primary key of each of the entities/relationships.

- TRACTOR(id, name, serialid, ipproxy, port, mac, id-farm)
 - *id*: identifier of the tractor.
 - *name*: name of the tractor.
 - *serialid*: serial identifier of the tractor.
 - *ipproxy*: IP address of the ACS the tractor is associated to.
 - *port*: number of the port with which the tractor connects to the ACS.
 - *mac*: the mac address of the EC
 - *id-farm*: identifier of the farm the tractor belongs to.
- FARM(id, name, address, phone)
 - *id*: identifier of the farm.
 - *name*: name of the farm.
 - *address*: address of the farm.
 - *phone*: telephone number of the farm.
- RELATED-WITH(id-tractor, id-user)
 - *id-tractor*: identifier of the tractor.
 - *id-user*: identifier of the user the tractor is related to.
- USER(id, username, password, firstname, lastname, role)
 - *id*: identifier of the user.
 - *username*: the name the user has to enter to access the system.
 - *password*: key for gaining admittance into the system.
 - *firstname*: real name of the user.
 - *lastname*: surname of the user.
 - *role*: identifier of the user's role in the system.

- CONTACT-INFORMATION(id, id-user, phone, address, email)
 - *id*: identifier of the contact-information register.
 - *id-user*: identifier of the user this contact information belongs to.
 - *phone*: telephone number.
 - *address*: home address.
 - *email*: electronic mail address.
- LOG(id,userid,date,time,log)
 - *id*: identifier of the log register.
 - *userid*: identifier of the user that made the action that is being logged.
 - *date*: date in which the action took place.
 - *time*: time in which the action took place.
 - *log*: action that was carried out.
- VARIABLE(timestamp,name,idtractor,value,units)
 - *timestamp*: date and time in which the variable had a given value.
 - *name*: name of the variable that is being logged.
 - *idtractor*: identifier of the tractor the value of the variable belongs to.
 - *value*: value of the variable that is being logged.
 - *units*: units of measurement for the logged variable.

5.2.7 Security Design

The security architecture chosen for the system is presented in the figure below.

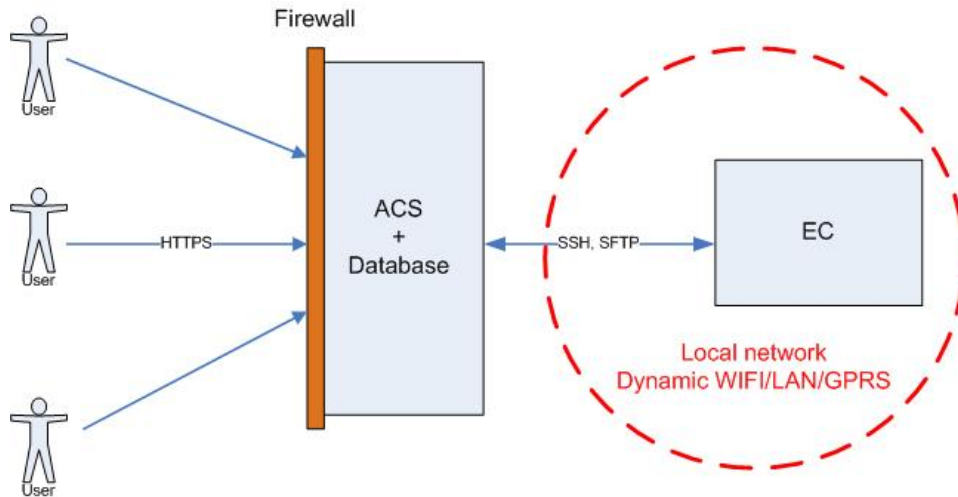


Figure 5.5: Security architecture

The idea is to have a firewall that acts as a filter. It is meant to accept the allowed packets and drop the unallowed packets, according to the rules defined for handling this. Moreover, for security - sensitive communication between a user and the ACS, HTTPS protocol is used. This is practically the same as HTTP but it adds an additional encryption/authentication layer between HTTP and TCP. As we already stated when presenting the why of the chosen architecture, the communication between the EC and the ACS is made by means of an SSH tunnel and the file transfer is by means of SFTP. This also adds encryption in the communications between the EC and the ACS.

On the other hand, for a user to get access to an EC it is required to log in using the right user name and password. In order to avoid unauthorized access to passwords stored in the database in the ACS, a solution is the use of a one-way encryption algorithm: MD5. The great advantage of using an irreversible encrypting algorithm such as MD5 for the storage of passwords is that even the superusers will not be able to know the passwords of the users by viewing the contents of the database.

In Cryptography, MD5 (Message - Digest Algorithm 5) is a widely used cryptographic hash function with a 128-bit hash value. As an Internet standard, MD5 has been employed in a wide variety of security applications. A MD5 hash is typically expressed as a sequence of 32 hexadecimal digits. This is a constraint when the type of data for the password attribute has to be chosen for the “user” table.

Recently, a number of projects have created MD5 “rainbow tables” which are easily accessible online and can be used to reverse many MD5 hashes into strings that collide with the original input, usually for the purposes of password hacking. However, if passwords are combined with a salt (prefix, suffix or both), rainbow tables become much less useful.

In order to choose how to generate the salt, we considered two possible attacks, namely, an attack on the source code of our application or on our database. If a random salt is generated, we should store it in the database together with the password which was encrypted with; however if somebody is able to access the database, we are in the same case as if we have not used salt. This is the reason why we chose to have the salt in the source code which will be running on the ACS, and then encrypt or obfuscate this code so that it cannot be viewed by anyone.

Since the encrypted code may require decryption, doing this will require entering a password, but we have not chosen this option because the problem of where and how to store the password will appear again. Hence we have chosen to simply obfuscate the code which will help us to hide the passwords’ salt and also to make any sort of reverse engineering more difficult.

Chapter 6

Implementation

This chapter covers implementation details of the design described in the previous chapter. We explain the configuration that will be made for the system to work, the structure of the programs that have been written, and we state some problems we encountered and propose solutions for solving them.

In general, our implementation is based on the configuration shown in figure 6.1, which is detailed throughout this chapter.

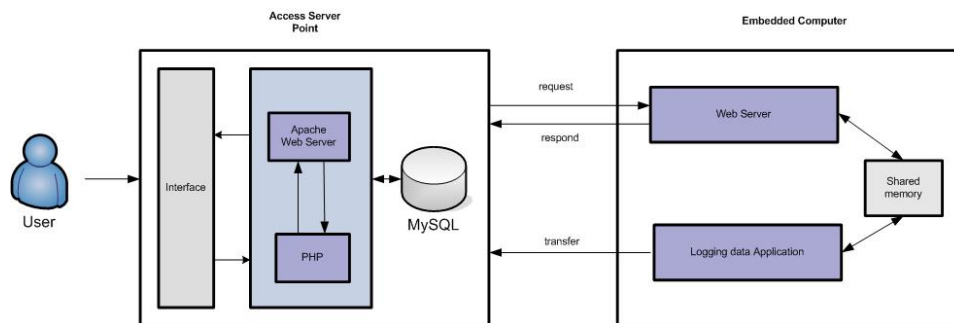


Figure 6.1: Implementation main structure

6.1 Database Implementation

The DBMS (*DataBase Management System*) chosen in this project is MySQL. This is because it is Open Source and has two important features: scalability and portability.

Below is the structure of the database tables in the ACS for controlling user's access to the system, it is also used for updating the port numbers from the EC:

contact-information

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto increment
id-user	int(11)	NO	MUL		
phone	varchar(15)	YES		NULL	
address	varchar(50)	YES		NULL	
email	varchar(50)	YES		NULL	

farm

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto increment
address	varchar(50)	NO			
phone	varchar(15)	NO			
name	varchar(30)	NO			

Log

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	NO	PRI	NULL	auto increment
userid	varchar(20)	YES		NULL	
date	date	NO		0000-00-00	
time	time	YES		NULL	
log	varchar(512)	YES		NULL	

related-with

Field	Type	Null	Key	Default	Extra
id-tractor	int(11)	NO	PRI		
id-user	int(11)	NO	PRI		

tractor

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	NO	PRI	NULL	auto increment
userid	varchar(20)	YES		NULL	
date	date	NO		0000-00-00	
time	time	YES		NULL	
log	varchar(512)	YES		NULL	

user

Field	Type	Null	Key	Default	Extra
id	int(10) unsigned	NO	PRI	NULL	auto increment
userid	varchar(20)	YES		NULL	
date	date	NO		0000-00-00	
time	time	YES		NULL	
log	varchar(512)	YES		NULL	

variable

Field	Type	Null	Key	Default	Extra
timestamp	varchar(15)	NO	PRI		
name	varchar(15)	NO	PRI		
value	int(11)	YES		NULL	
units	varchar(8)	YES		NULL	
idtractor	varchar(11)	YES		NULL	

6.2 Intercommunication

The entire transmission of data between an EC and ACS is done via **SSH Tunnel**. All TCP traffic are sent or received via the tunnel. This option does not only help to traverse the NATs router but it also provides a more secure alternative when compared to rlogin, telnet or rsh.

SSH counts on several methods to verify the identity of a remote user. One of them is the use of passwords, another is based on RSA authentication, i.e. RSA authentication generates a pair of keys - private/public that guarantees the identity of the user that is trying to connect to the remote machine. The pair of keys has the property that what is encrypted with one of them can be decrypted using the other one. However, it is not feasible to derive the private key from the public one.

On power up of the EC, the script “**launchall.py**” that is part of the start up sequence, checks for the availability of wired connection; if a wired connection is available, an SSH tunnel to the ACS is created. If the wired connection is not available in the EC on start up, it checks for the availability of GPRS connection, and if this is available, a GPRS SSH tunnel to the ACS is created.

When an EC succeeds in creating a tunnel, it makes a “**port update**” in the database in the ACS. In our case, it should use 8080 or 8081. This number depends on the type of connection that is currently in use in the EC, i.e, either wired or GPRS, and it is defined in the configuration file.

To be efficient, the ACS must regularly monitor and confirm the availability of the tractors connected to it. This is because after the EC updates the port database to indicate that it is online, it may loose connection to the Internet for some reasons. If this is the case, there is no way for the ACS to know about this, so as to update the availability of the tractor.

In order to prevent the ACS from showing that an EC is “online” when it has lost connection to the Internet, a subsequent Python script “**porttesting.py**” located in the ACS is designed to take care of updating the port attribute. The script works by regularly checking through the “tractor” table in the database in the ACS, if the port column is different from '0' it means that a connection with the tractor is possible. It then makes a ping to the EC, if there is a response from the EC, it indicates that the EC is still online, but if there is no response from the EC, it indicates that the EC is offline and then the script will reset the port value to '0'.

On the ACS, it is possible to see the tractors that are online. The names of the tractors that are online are shown in **green** and those that are offline are shown in **red**.

In the previous system, the entire scenario of accessing the ACS was simulated with the ACS inside AAU's Network. Presently, the ACS has obtained a public address; as this will be the case in the real world. With the public IP address, it is now possible to connect to the ACS from outside AAU network. The IP address has been changed to "130.225.192.134" and the necessary configurations have been made.

Once the system is on, a user can connect to the ACS using a web browser and the IP address. With a web browser like Firefox or Microsoft Internet Explorer, it is possible for a user after log in, to select the EC of interest. Clicking the name of an EC will display information from the EC on the user's web browser.

6.2.1 Log, compression and transfer from the EC

Figure 6.2 shows the UML component diagram for logging, compression and transfer of files from the EC.

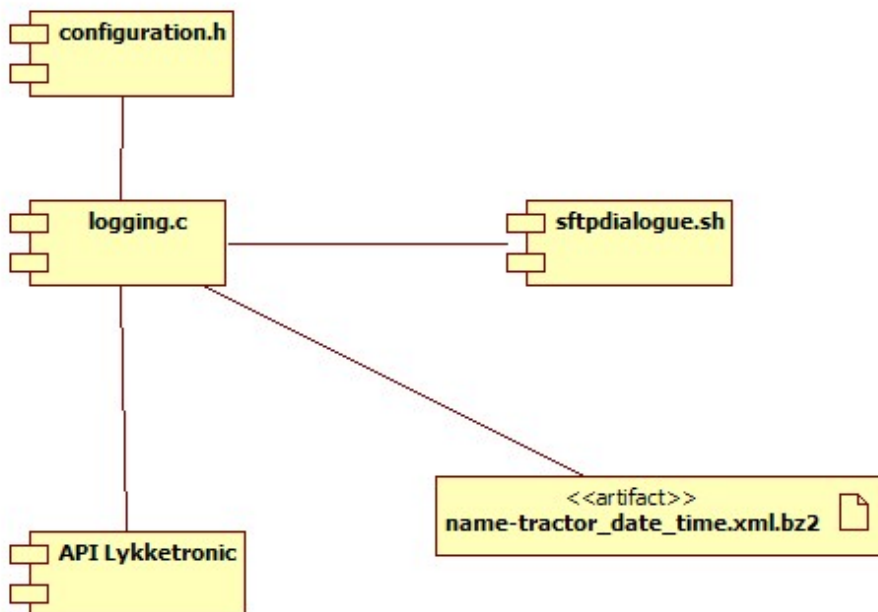


Figure 6.2: Logging application: components diagram

In order to log data in the EC, a sub-set of variables have been selected and is used for our experiments. The programming language chosen to develop the application is C, which presents integration advantages, since we will use the API¹ from LYKKETRONIC written in C.

The program “**logging.c**”² writes a log with the values of the variables that are passed as argument. As explained before, the parameters have been divided into two groups. Each group of parameters is logged according to the frequency defined for them. For our experiment, we have set the first group of parameters for reading every 5 seconds, while the second group of parameters are to be read every 30 minutes, and at every 15 minutes, the file is closed, compressed and sent to the ACS.

The purpose of transferring the files from the EC to the ACS is to save the amount of space that will be required for storing the files in the EC, since the space in the EC is limited. It is also to provide access to data from the different machines at a central location. This centralized solution eases maintainability tasks.

In order to be able to periodically log data, we made use of a POSIX signal, SIGALRM, to send and receive signals. This signal has an associated handler that is in charge of writing data into the log file and then, compressing it. The XML files are compressed with a .bz2 extension before sending to the ACS. The XML schema has been validated according the W3C syntax [26].

To save the cost of transferring the files, we will only transfer them when the EC is using the wired connection. This is because using GPRS connection to transfer the files can become too expensive, besides the files are not time critical data or urgently needed.

¹This API extracts the values of the different parameters of a tractor from shared memory of the embedded computer.

²As mentioned earlier on, the parameters used for logging are only a subset of the entire parameters in the system. It is possible to log more parameters with “logging.c” function because the function is not parameter dependent, it can easily be modified to use any number of parameters. Also, the frequency of reading and writing the value of each group of parameters can easily be modified according to the desired time frequency. For the file transfer, the function responsible for doing this can also easily be altered to use GPRS to make the transfer if the wired connection is unavailable.

For transferring the files from the EC, we have used secure FTP protocol, i.e, SFTP. One of the main advantages of this protocol is that the transferred files are encrypted. Because SFTP establishes dialogue or interaction with a user, we wrote a bash script to take care of the interaction using the following utilities:

spawn. It allows the execution of programs by command line.

expect. It waits for the string appearing on standard output.

send. It sends commands to the Access Control Server via standard input.

The transferred files are decompressed and stored in a database on arrival at the ACS. The UML diagram below shows the structure of the programs used to carry out this process:

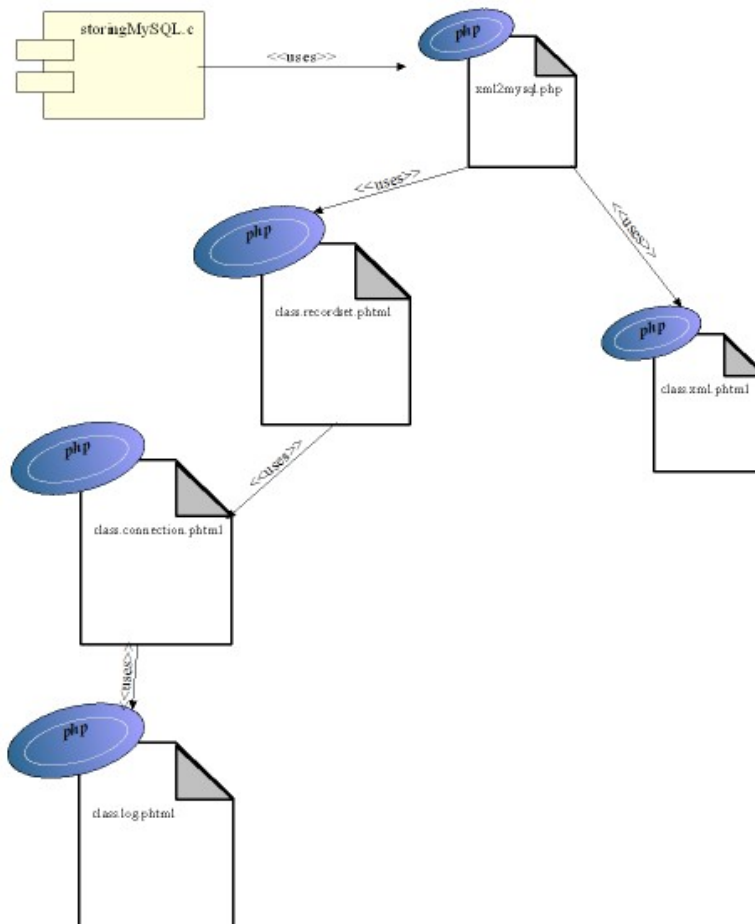


Figure 6.3: Decompression application: components diagram

6.2.2 Connection switch in the EC

Only one connection will be in use in the EC at any point in time. The default connection in the EC is the wired connection because it is cheaper and faster to use it when it is available. When the wired connection becomes unavailable, and a GPRS connection is available, then the EC will switch to the GPRS connection and vice-versa.

The main script responsible for this is “**wirewatchdog.py**”. It constantly runs in the background in the EC to check for a wired or GPRS interface.

The programming language used for the connection issues in our system is Python because it is fast since most of its modules are written in C and C++.

We briefly present the scripts that are used in order to make the switch in the EC possible. They are listed in alphabetical order:

connectionswhitch.py It contains the functions in charge of starting the different connections.

ConnectionTools.py This file contains the functions that are in charge of checking if there is a GPRS modem plugged, a cable, and, in this case, if the Internet connection is available.

info.conf It provides us with configuration information, such as port number, name of the tractors, login and password for accessing the AAU Network or the ACS Data Base.

SshTools.py It includes the functions for creating or closing an SSH tunnel.

Figure 6.4 shows a diagrammatic view of the scripts running in the EC and their relationships. The scripts in green color are launched at the start up of the system.

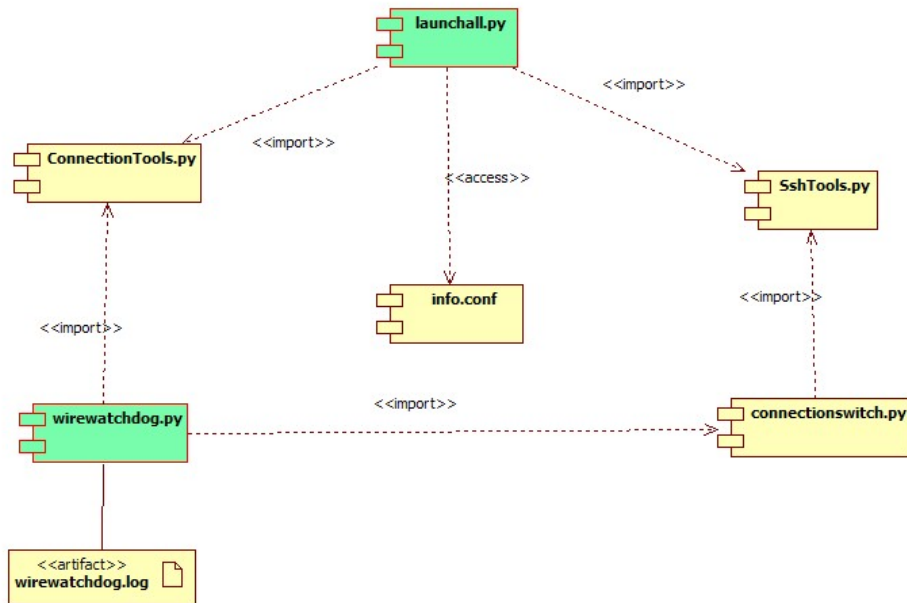


Figure 6.4: Relationships between the scripts running in the EC

The `<<import>>` UML dependency adds the content of the destination to the origin, while the `<<access>>` does not add it, only uses it.

6.2.3 Web Applications

Our project uses two different web applications in the EC. The first application is the original one, developed by LYKKETRONIC for their system (see Appendix A). When the EC is connected to the Internet using the wired connection, this web application is used. When the EC is connected to the Internet via a GPRS network, a second application is used. This second application is a lighter version that a user can use for requesting and viewing “snapshot” data. The purpose of making a lighter version of the web application is to cut down the size and cost of transferred bytes while using a GPRS connection.

We must state that the web applications have been programmed in PHP because it is Open Source, fast and portable to most platforms.

The appropriate page to be displayed for the corresponding connection is made possible by setting the virtual host paths in the Apache Web Server. The appropriate paths can be set in the following configuration file: `/etc/httpd/conf/httpd.conf`

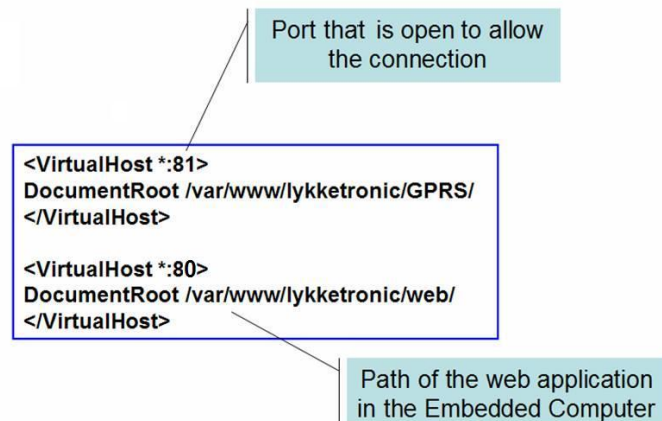


Figure 6.5: Configuration of the Virtual Hosts in the Embedded Computer

Figure 6.6 shows where we chose to host each of the web applications.

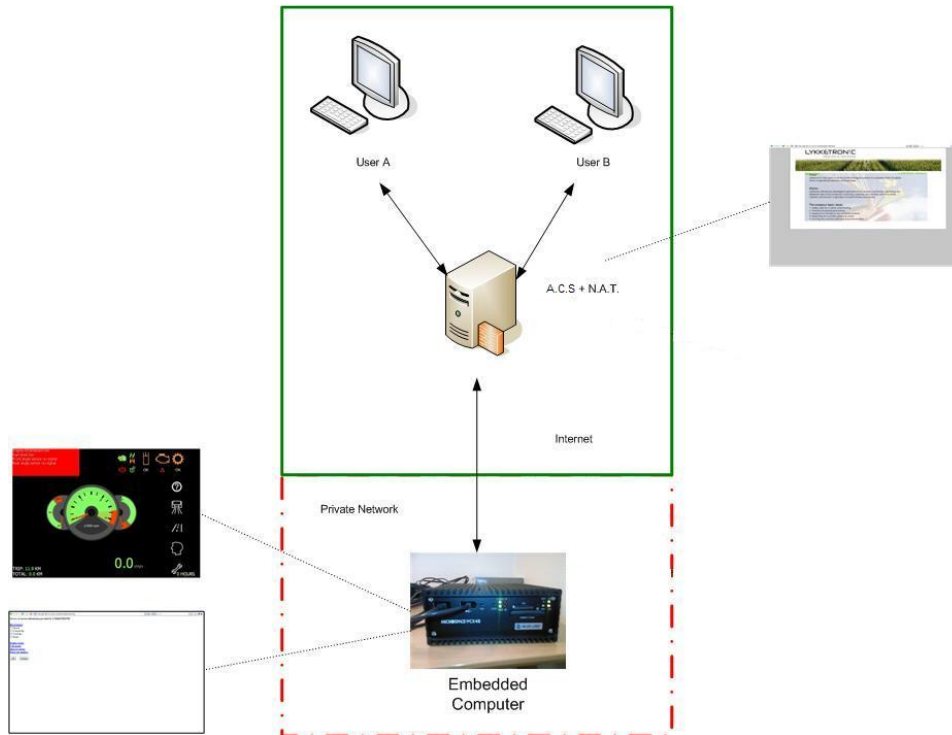


Figure 6.6: Location of the web application files.

The applications that access the shared memory of the EC, are also located here. This is because it makes the access easier. Moreover, a centralized interface directly from the ACS is not the best option, because not all the agriculture machinery have the same variables to be monitored.

Below is the structure of the web application located in the EC:

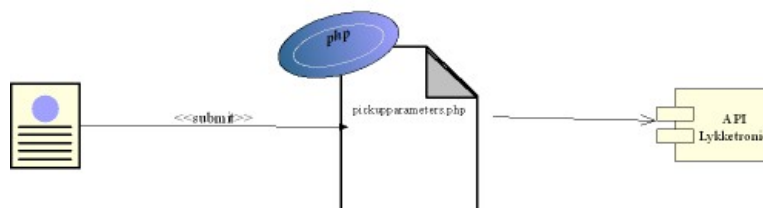


Figure 6.7: Web Application for GPRS connection

Here we can see how the web page that submits the web form where a user can select the variables he is interested in, makes a call to the API from LYKKETRONIC, which had to be previously modified by us in order to accept parameters.

The hierarchical structure of the Web Application used on the ACS for user authentication and contents administration is presented in the following diagram, it has been made according to UML WAE (Web Application Extension), defined by Jim Conallen[5].

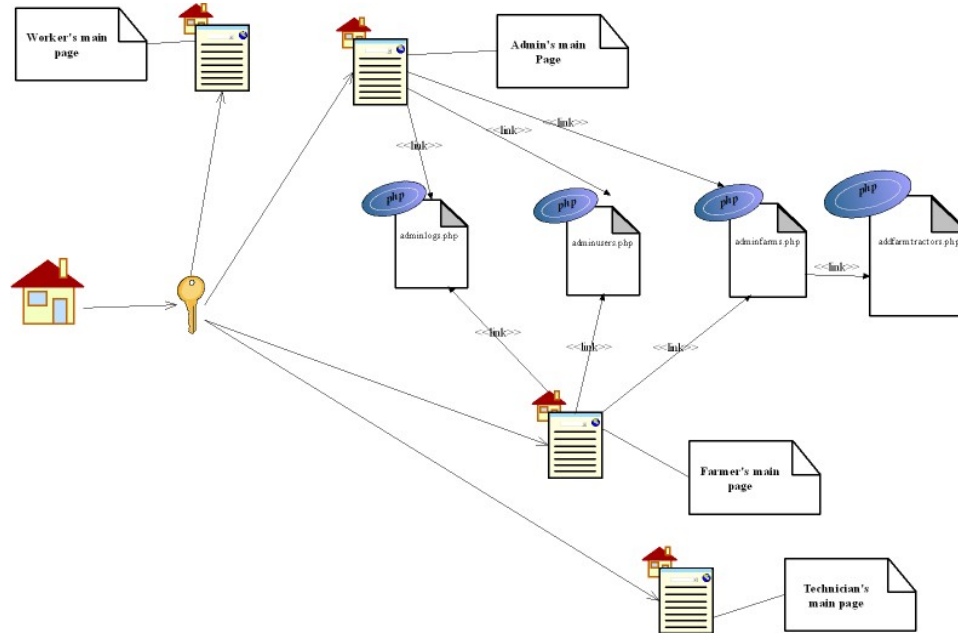


Figure 6.8: Administration Web Application

HTTP Error Codes. In our web applications, we have taken care of the most usual HTTP error codes, which inform us about some errors that may occur during the connection to a web sites. We can divide them into two different groups:

- Client errors
 - 404. File not found
 - 408. Request timeout
- Server Errors
 - 500. Internal Server Error
 - 501. Not implemented
 - 503. Service unavailable
 - 509. Bandwidth limit exceeded

We have configured the file `.htaccess` in each of the web application folders, this is to enable the appropriate error message to be shown according to the code received.

6.2.4 System Security

Iptables

Iptables tool was used for defining firewall rules in both the EC and the ACS. The rules are defined using shell scripts, designed to control packets coming and going out of the system.

By default, the firewall in the ACS will deny every connection to the SSH tunnel. Once a user is logged in to the login page of the ACS, a script is launched to add a rule to the firewall rules. The rule is to allow the IP address of the user to access the part of the SSH tunnel. When the user logs out or after a long time without any activity from the user, another script is launched to remove the rule that allowed the user's IP. address.

SSH and HTTPS

The use of SSH between the EC and the ACS combined with the use of HTTPS between a client (browser) and the ACS provides security in the transmitted data in the system. The network traffic generated between a client (browser) and Apache web server is through port 80, and it is not encrypted. In this way it is possible, to use an appropriate tool to observe the traffic and obtain, for example, passwords. A solution to this problem was to obtain a security certificate and redirect the traffic to the port 443 (HTTPS) instead of port 80 (HTTP). Traffic to port 443 are encrypted because of the use of SSL protocols(Secure Sockets Layer) and TLS (Transport Layer Security). This guarantees security and even if it is possible to intercept the traffics, a hacker will only be able to see chains of characters that have no meaning. This helps to secure the channel between a client and the ACS.

Obfuscation

SourceGuardian is the tool that LYKKETRONIC used for hidden the code of its web application. SourceGuardian is a PHP encoder; it protects PHP scripts by compiling the source code into a binary bytecode format, this is then supplemented by encryption. Obfuscation is also used within the process at various points [24].

The aim of obfuscation is to protect the PHP scripts so that no changes can be made to it. As part of the security level, an any change made to an encoded file will render it unusable. On page 131, an example where we use obfuscation to hide our source code from hackers can be found.

6.3 Implementation problems and suggested solutions

6.3.1 WIFI dongle installation:

Many WIFI vendors do not release specification of the hardware or provide a Linux driver for their wireless network cards. The NDISWrapper project implements Windows Kernel API and NDIS(Network Driver Interface Specification) API within Linux kernel. A Windows driver for wireless network card is linked to this implementation so that the driver runs natively as though it is in Windows, without binary emulation [18].

As we already said in the page 31, the WIFI dongle available for this project is a NETGEAR wg111T, with Atheros chipset. There exists a different initiative for this sort of chipset called MadWIFI [17]. We could not use this because it is not compatible with USB devices.

In order to be able to install the WIFI dongle in our system, we removed unnecessary data and software from the flash memory card in the EC, in order to create more space. The CentOS version was updated and its kernel was recompiled. After this, we could still not install the WIFI dongle. This is because there is no compatible NDISWrapper for this sort of kernel at the moment. Faced with this problem, we propose three possible solutions:

1. Change the operating system in the Embedded Computer to another one e.g., Ubuntu 7.10.
2. Acquire a new dongle with Linux drivers compatible with CentOS.
3. Write a driver for the current dongle.

The first option may not be convenient for LYKKETRONIC because of possible incompatibilities of their software with a different operating system. Secondly, there is no known dongle to install at the moment. The third problem, involves a difficult task that is outside the scope of this project.

6.3.2 Shared memory access from Apache Web Server

A problem we had during the implementation of the the light weight web application hosted in the Apache Server in the EC, is that the values of the data read from the shared memory in the EC appeared wrongly, but when the same the values were read by executing the C-program in command line they returned correctly.

At the beginning, we thought it was a problem of data representation within Apache, but finally, by checking `/etc/httpd/logs/error-log` we found out that the problem had to do with privilege issues.

The only apparent solution seemed to be executing Apache as a root user. Nevertheless, this solution was not acceptable because of security issues and furthermore, we needed to recompile the kernel with the flag `EXTRA_CFLAGS=-DBIG_SECURITY_HOLE`, in such a way that later on, we could change into the `httpd` configuration file the user “apache” (default user and group) by root.

Eventually, we were able to find a better solution. This consists of placing before the call to the modified-API from `LYKKETRONIC`, the typical command “`sudo`”, which also implies modifying the file `/etc/sudoers`. This file can also be found in Appendix B.

6.4 System Configuration

In this section, we present the reader information about all the files and directories that had to be created or modified for our implementation to success.

6.4.1 ACS configuration

\$HOME/porttesting.py It is in charge of scanning the ports that are being used at any given time.

/var/www/login Web applications for the login of users to the system.

/var/www/uptract For administrative tasks, it insert values into the database located in the ACS.

/etc/apache2

key.pem Private key.

certificateProxy.pem Autosigned certificate.

ports.conf If we are using ports different from port 80, apart from declaring it by means of the directive `NameVirtualHost` and `VirtualHost` it is also necessary to indicate it in this file by adding the list of port we are interested in using.

sites-enabled/ProxySSE4 The file that declares and configures the `VirtualHosts` ³.

\$HOME/decompression

storingMySQL.c The program that will be running in background, it extracts the data that has been sent by the EC in XML format and store it in the MySQL database.

compileStoringMySQL.sh Script used for compiling and storing `MySQL.c`

.ssh

authorized-users File with the public keys of the machines we can access.

id-rsa Private key of the ACS.

³The term `VirtualHosts` refers to the practice of maintaining more than one server on one machine, as differentiated by their apparent hostname

6.4.2 EC configuration

\$HOME/scripts Python files in charge of the different sorts of Internet connection (GPRS, wired) between the EC and the ACS, the system's configuration file and the firewall scripts.

sse4 Files in charge of the data gathering from the shared memory. Files in charge of the periodically logging and transferring files to the ACS are also included.

/var/www/lykketronic

GPRS Simple web page. READ - ONLY mode.

web Web page for the connection ethernet/WIFI. It is developed by Househam Sprayers Limited. Its source code is encrypted in such a way that it cannot be modified by us.

/etc/httpd/conf/httpd.conf Configuration of the Virtual Hosts.

.ssh

authorized-users Files with the public key of the machines we can access.

id-rsa Private key of the embedded computer.

/etc/rc3.d By adding references in this folder, we are also adding services that are going to be executed automatically when the EC is started at the network services level. Specifically we have two:

S99launchall.py Symbolically linked to \$HOME/scripts/launchall.py

S99wirewatchdog.py Symbolically linked to \$HOME/scripts/wire-watchdog.py

/etc/wvdial.conf When wvdial starts, it first loads its configuration information from the configuration file defined in /etc/wvdial.conf. The file contains basic information like modem type, speed, etc. along with information about the Internet Service Provider (ISP).

/etc/sudoers This file contains rules that users have to follow using sudo command.

Complete source codes of all the files can be found in the attached CD.

Chapter 7

Tests and performance results

In this chapter we present the test cases and the results we obtained when executing our system. The main objective is to show that the requirements are satisfied. Furthermore, we will take measurements that are of immediate importance and value to LYKKETRONIC. All details about the hardware and software are included in Table 7.1. Results from the tests are included in this chapter and references are made to the relevant tables and graphs.

7.1 Test approach

The main measurements to be made at the system have to do with the realtime and logged data that are sent from the EC. We will present all these aspects by following the GQM (Goals - Questions - Metrics)[7] approach.

Goals:

- G1.** To offer an estimation of the price that one access to the EC via a web interface in realtime costs by using any of the available technologies from a farm owner point of view.
- G2.** To know the advantage in economical terms for a normal user to download a page and not the other from a farm owner point of view.
- G3.** To find an estimation of the price it would cost for sending logs over a period of time (8 hours) with a GPRS connection.
- G4.** To know the economical advantage of compressing a log file before sending it to the ACS

Questions:

- Q1.** How much does connecting in realtime to a tractor by using an Ethernet connection costs?
- Q2.** How much does sending a log to the ACS costs?

Metrics:

- M1.** Kilobytes transferred using the simple web page.
- M2.** Kilobytes transferred using the LYKKETRONIC web page.
- M4.** Transmission cost by using GPRS with the simple page.
- M5.** Transmission cost by using GPRS with LYKKETRONIC's page.
- M6.** Kilobytes transferred in the transmission of a log file.
- M7.** Transmission cost by using GPRS for log.

In the following sections we will try to get these goals and answer these questions by using these metrics.

7.2 The Test Setup

Below is a description of the equipments that are used for carrying out the simulations and tests:

1. Mobile phone	
Model:	Nokia N71 GPRS class 10
SIM	CBB Mobile
GPRS download speed	200Kbps, 25Kb/sec transfer rate
GPRS upload speed	70Kbps, 8.8Kb/sec transfer rate
2. Data cable	
Interface:	DKU-2 USB data cable
Speed	460800 baud
3. Mobile computer(EC)	
Model:	Celeron 800mhz with 256MB RAM
Operating system:	CentOS 4.4, Kernel 2.6
Software packages:	pppd, wvdial
4. Access Control Server	
Model:	Fujitsu Siemens SCENIC P300
Operating system:	Ubuntu 7.10
Software packages:	Open ssh, dhclient
IP address:	130.225.192.134
5. Internet Connection	
Service provider:	DENet - Danish Network for Research and Education
Download speed:	3501kbps, 437.6Kb/sec transfer rate
Upload speed:	2258kbps, 282.3Kb/sec transfer rate

Table 7.1: Description of the equipments used during the test

7.3 Test cases

Each of the test cases below, were tested at least 5 times, in order to check that the system works and to obtain more accuracy in the results from the tests.

7.3.1 Test Case I: System Startup

In this scope, we define four different cases:

1. Ethernet cable and GPRS modem are plugged on system's power up.
 - Expected result: Ethernet connection is started.
 - Obtained result: OK.
2. No ethernet cable or GPRS modem is plugged on system's power up.
 - Expected result: Error message, "No connection could be started".
 - Obtained result: OK.
3. Only the Ethernet cable is plugged on system's power up.
 - Expected result: Ethernet connection is started.
 - Obtained result: OK.
4. Only the GPRS modem is plugged on system's power up.
 - Expected result: GPRS connection is started.
 - Obtained result: OK.

We also measured the time it takes to start up a wired and GPRS connection in the EC. The results are presented in table 7.2. The average times are highlighted in yellow.

Startup with wired (s)	Startup with GPRS(s)
12	124
12	115
12	115
12	124
12	119
12	119.4

Table 7.2: Time needed to startup wired and GPRS Internet connections

Results from Table 7.2 show that starting up the system with a GPRS connection is about 10 times slower when compared with the wired connection start up. The slow start up of the GPRS connection can be as a result of different factors such as: the time spent by wvdial for dialing the ISP or as a result of the high RTT experienced by GPRS connection. Another possible factor for the slow start up, may be because of the class of GPRS terminal in use. Currently, we are using GPRS class 10, and the higher a GPRS class, the faster the data rates. From the measurements carried out, we have a maximum data transfer speed of 70 kbps upload and 200 kbps download. But, this speed is also dependent on the type of hardware in use.

Which the following figures, we want to demonstrate the correct startup of the system.

Cases 1 and 3: The wired connection is started

```
[root@ibox ~]$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:30:59:03:68:64
          inet addr:192.168.81.75  Bcast:192.168.81.255  Mask:255.255.255.0
          inet6 addr: fe80::230:59ff:fe03:6864/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:346 errors:0 dropped:0 overruns:0 frame:0
          TX packets:265 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:39013 (38.0 KiB)  TX bytes:40298 (39.3 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:1483 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1483 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:685844 (669.7 KiB)  TX bytes:685844 (669.7 KiB)
```

Figure 7.1: Interfaces when starting up the system with cable (and maybe also GPRS modem) connected

The screenshot shows a terminal window titled "homer.cs.aau.dk - [default] - F-Secure SSH Client". The terminal displays the output of a MySQL query: "select * from tractor;". The output is a table with 12 rows and 7 columns: id, name, serialid, ipproxy, port, mac, and id_farm. The first row is "lab-tractor" with serialid 0105 and ipproxy 130.225.192.134. The remaining rows are "fake-tractor" entries with serialids from 0106 to 0124. The ipproxy for all is 130.225.192.134. The port is 8080 for the first row and 0 for the others. The mac address is 00:30:59:03:68:64 for the first row and NULL for the others. The id_farm column has values 1, 1, 1, 3, 4, 5, 6, 2, 4, 4, 4, 4.

```

Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> select * from tractor;
+----+-----+-----+-----+-----+-----+-----+
| id | name          | serialid | ipproxy      | port | mac                | id_farm |
+----+-----+-----+-----+-----+-----+-----+
| 1  | lab-tractor   | 0105     | 130.225.192.134 | 8080 | 00:30:59:03:68:64 | 1        |
| 2  | fake-tractor1 | 0106     | 130.225.192.134 | 0    |                    | 1        |
| 3  | fake-tractor2 | 0107     | 130.225.192.134 | 0    |                    | 1        |
| 5  | fake-tractor4 | 0109     | 130.225.192.134 | 0    |                    | 3        |
| 6  | fake-tractor5 | 0110     | 130.225.192.134 | 0    |                    | 4        |
| 7  | fake-tractor6 | 0111     | 130.225.192.134 | 0    |                    | 5        |
| 8  | fake-tractor7 | 0112     | 130.225.192.134 | 0    |                    | 6        |
| 9  | fake-tractor3 | 0108     | 130.225.192.134 | 0    | NULL              | 2        |
| 10 | fake-tractor8 | 0113     | 130.225.192.134 | 0    |                    | 4        |
| 11 | fake-tractor9 | 0119     | 130.225.192.134 | 0    |                    | 4        |
| 12 | Fake-tractor10 | 0124     | 130.225.192.134 | 0    |                    | 4        |
+----+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)

mysql>

```

Figure 7.2: Tractor table in the database when connected via cable

```

root    7135    1 0 16:58 ?    00:00:00 ssh -fNC -L 2222:130.225.192.134:22 -p 22 isabel@homer.cs.aau.dk
root    7137    1 0 16:58 ?    00:00:00 ssh -fNC -R 8080:127.0.0.1:80 -p 2222 ec@127.0.0.1
root    7139    1 0 16:58 ?    00:00:00 ssh -fNC -L 56789:127.0.0.1:56789 -p 2222 ec@127.0.0.1

```

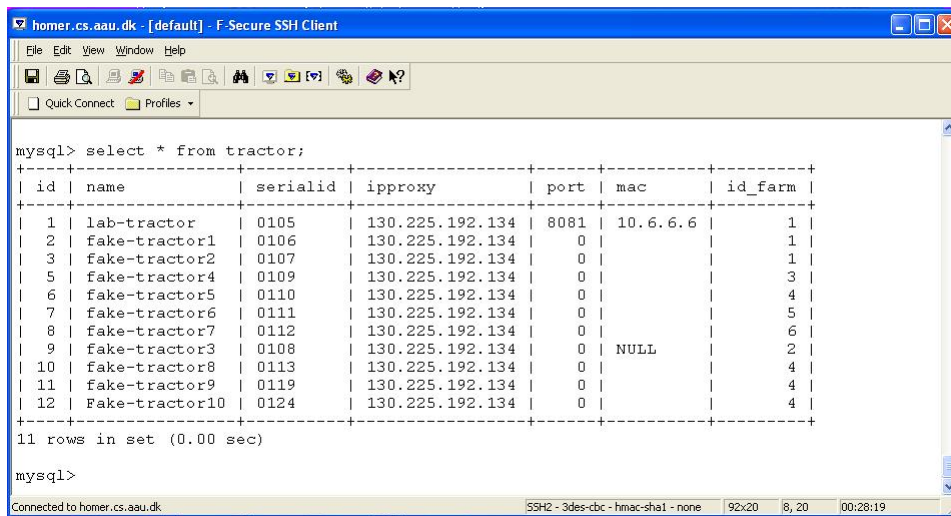
Figure 7.3: Tunnels created when connecting via cable

Case 4: The GPRS connection is started

```
lo          Link encap:Local Loopback
           inet addr:127.0.0.1  Mask:255.0.0.0
           inet6 addr: ::1/128 Scope:Host
           UP LOOPBACK RUNNING  MTU:16436  Metric:1
           RX packets:1694 errors:0 dropped:0 overruns:0 frame:0
           TX packets:1694 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:0
           RX bytes:734516 (717.3 KiB)  TX bytes:734516 (717.3 KiB)

ppp0       Link encap:Point-to-Point Protocol
           inet addr:10.20.97.39  P-t-P:10.6.6.6  Mask:255.255.255.255
           UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:472  Metric:1
           RX packets:33 errors:0 dropped:0 overruns:0 frame:0
           TX packets:40 errors:0 dropped:0 overruns:0 carrier:0
           collisions:0 txqueuelen:3
           RX bytes:6693 (6.5 KiB)  TX bytes:6036 (5.8 KiB)
```

Figure 7.4: Interfaces when starting up the system with only GPRS modem



The screenshot shows a terminal window titled "homer.cs.aau.dk - [default] - F-Secure SSH Client". The terminal displays the output of a MySQL query: "mysql> select * from tractor;". The result is a table with 12 rows and 7 columns: id, name, serialid, ipproxy, port, mac, and id_farm. The data is as follows:

id	name	serialid	ipproxy	port	mac	id_farm
1	lab-tractor	0105	130.225.192.134	8081	10.6.6.6	1
2	fake-tractor1	0106	130.225.192.134	0		1
3	fake-tractor2	0107	130.225.192.134	0		1
5	fake-tractor4	0109	130.225.192.134	0		3
6	fake-tractor5	0110	130.225.192.134	0		4
7	fake-tractor6	0111	130.225.192.134	0		5
8	fake-tractor7	0112	130.225.192.134	0		6
9	fake-tractor3	0108	130.225.192.134	0	NULL	2
10	fake-tractor8	0113	130.225.192.134	0		4
11	fake-tractor9	0119	130.225.192.134	0		4
12	fake-tractor10	0124	130.225.192.134	0		4

The terminal also shows "11 rows in set (0.00 sec)" and "mysql>" at the bottom. The status bar at the bottom of the window indicates "Connected to homer.cs.aau.dk" and "SSH2 - 3des-cbc - hmac-sha1 - none 92x20 8, 20 00:28:19".

Figure 7.5: Tractor table in the database when connected via GPRS

```
root 13858 0.0 0.1 5600 952 ?      Ss  17:05  0:00 ssh -fnc -L 2222:130.225.192.134:22 -p 22 isabel@homer.cs.aau.dk
root 14106 0.0 0.1 4232 712 ?      Ss  17:05  0:00 ssh -fnc -R 8081:127.0.0.1:81 -p 2222 ec@127.0.0.1
root 14370 0.0 0.1 4208 660 ?      Ss  17:05  0:00 ssh -fnc -L 56789:127.0.0.1:56789 -p 2222 ec@127.0.0.1
```

Figure 7.6: Tunnels created when connecting via GPRS

7.3.2 Test Case II: Connection Switch

For switching between the wired and the GPRS connection, the following combinations have been tested and are possible:

1. Switching from GPRS to wired.
 - Expected result: the wired connection is started.
 - Obtained result: OK.
2. Switching from wired to GPRS.
 - Expected result: GPRS connection is started.
 - Obtained result: OK.
3. Switching “from nothing” to GPRS.
 - Expected result: GPRS connection is started.
 - Obtained result: OK.
4. Switching “from nothing” to wired.
 - Expected result: wired connection is started.
 - Obtained result: OK.
5. Switching from wired to “no connection”:
 - Expected result: The SSH tunnels are closed, and the port used for connecting the tractor to the ACS, is reset to 0.
 - Obtained result: OK.
6. Switching from GPRS to “no connection”:
 - Expected result: The SSH tunnels are closed, and the port used for connecting the tractor to the ACS, is reset to 0.
 - Obtained result: OK.

After testing the connection switch solution, some encouraging results were produced in the time taken to switch between the two connections. The results are presented below.

Wired to GPRS (s)	GPRS to Wired (s)
27	18
38	15
30	12
27	14
30	13
34.4	14.4

Table 7.3: Time needed to do the switching in different directions

The values showed in yellow are the average time to do each of the switching; as expected, the switching from the wired connection to GPRS connection took longer since it is necessary to execute the `wvdial` command to bring up the `ppp0` interface and also possibly because of some of the reasons mentioned earlier.

The results from this test are satisfactory, since in the previous semester, it was only possible to switch from the wired connection to GPRS connection, and in most cases, it usually ends without success. In the cases where the previous scripts worked, the empirical experiments that were made shows a switching time of about 90 seconds. This shows that the new implementation has decreased this time to about one third.

From these results it is evident that the connection switching solution is able to switch between the different connection while it roams between the networks, and the switch is considerably fast compared to the previous solution.

The switching solution is not protocol dependent, hence it is not restricted to only GPRS and wired connections. The solution can be used with any type of IP based network interface - wired or wireless connections.

7.3.3 Test Case III: Logs

For logging the parameters we have taken the following scenarios into account:

1. Log with an increasing number of variables:
 - Expected result: All the variables passed as parameter have been logged and the results are correct.
 - Obtained result: OK
2. Transfer of the logs to the ACS via SFTP is possible.
 - Expected result: The logs have been transferred.
 - Obtained result: OK
3. Insertion of the log contents into the database
 - Expected result: The contents have been correctly stored in the database.
 - Obtained result: OK

Here we present some screen shots that were taken during the tests.

Figure 7.7: Example of an XML file with two variables logged

```
mysql> select * from variables;
```

timestamp	name	value	units	idtractor
10:10	prueba	NULL	NULL	NULL
10:11	prueba	18	l	0
10:13	prueba	18	l	0
12:34	variable1	5	l	0
22:34	variable1	5	l	5
22:22	variable1	5	l	5
10:10	variable1	5	l	9
21:11:18	Distance	256	km	1
21:11:18	Tank content	1211	l	1
21:11:23	Distance	261	km	1
21:11:23	Tank content	1211	l	1
21:11:28	Distance	266	km	1
21:11:28	Tank content	1211	l	1
21:11:33	Distance	271	km	1
21:11:33	Tank content	1211	l	1
21:11:38	Distance	276	km	1
21:11:38	Tank content	1211	l	1
21:11:43	Distance	281	km	1
21:11:43	Tank content	1211	l	1
21:11:48	Distance	286	km	1
21:11:48	Tank content	1211	l	1
21:11:53	Distance	291	km	1
21:11:53	Tank content	1211	l	1
21:11:58	Distance	296	km	1
21:11:58	Tank content	1211	l	1
21:12:03	Distance	301	km	1
21:12:03	Tank content	1211	l	1
21:12:08	Distance	306	km	1
21:12:08	Tank content	1211	l	1
21:12:13	Distance	311	km	1
21:12:13	Tank content	1211	l	1
21:12:18	Distance	316	km	1
21:12:18	Tank content	1211	l	1
21:12:23	Distance	321	km	1
21:12:23	Tank content	1211	l	1
21:12:28	Distance	326	km	1
21:12:28	Tank content	1211	l	1
21:12:33	Distance	331	km	1
21:12:33	Tank content	1211	l	1

Figure 7.8: Extracted “variables” table in the database in the ACS

7.3.4 Test Case IV: Viewing a job in the EC

At the completion of a job by a tractor, it is possible to remotely access the job parameters such as distance, amount, diving time, etc.

The idea of this test is to use both the light weight page we developed and LYKKETRONIC’s web page for viewing the current job information using GPRS connection.

1. Check that the page works with the given number of variables passed as argument.
 - Expected result: It displays the result of the the given number of variables.
 - Obtained result: OK
2. Check that the displayed results are the same as the values displayed using LYKKETRONIC’s page.
 - Expected result: The displayed result are the same as the one displayed on LYKKETRONIC’s page.
 - Obtained result: OK

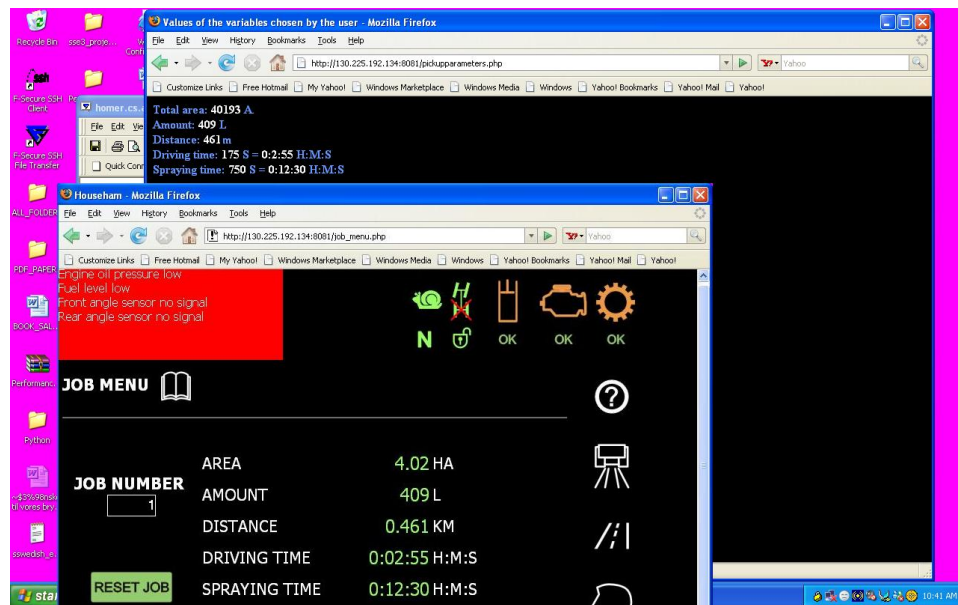


Figure 7.9: Web page comparison

Figure 7.9 shows the result of a job using LYKKETRONIC's page and the result of the same job using the light weight page. As can be seen, the values are the same.

From these results, in order to view the current job using LYKKETRONIC's page, it used a total of 2,669MB and it costed 21,18 DKK. For viewing the same job with the light weight page, it used a total of 462kB and it costed 3,60 DKK.

From these experiments, it is obvious that both pages have displayed the same values for the parameters hence, we can conclude that the information displayed by the light weight page is correct. Also, lesser kilobytes was used for uploading the same data set with the light weight page.

Since GPRS charge is per megabyte of transferred data, from this experiment, it is obvious that it costed 17,58 DKK more to access the same information by using the heavy weight web page.

From this experiment, it shows that if LYKKETRONIC makes a corresponding lighter web interface for accessing the entire system, it will be cheaper. The current light page we developed can only be used for accessing a subset of the entire system also, it is primarily developed for our experiments. Changing this feature can also help to reduce the overhead added by the current heavy weight web in terms of data transferred and cost.

Administration Web page in the ACS

1. Log in with different sort of users.
 - Expected result: The application presents the right menu for each of the different roles.
 - Obtained result: OK
2. Check the different options of adding, deletion and modification of users, farms and tractors.
 - Expected result: the users, farms and tractors are successfully added, deleted or modified respectively.
 - Obtained result: The users, farms and tractors are modified successfully.
3. Check the user access via URL.
 - Expected result: Faced with the try of accessing to a forbidden web page via URL, an error message is shown.
 - Obtained result: OK
4. Check what happens when we try to access a web page that does not exist.
 - Expected result: a customized error page.
 - Obtained result: OK

7.4 Performance Study

Wireless networks have a significant latency due to error mechanisms introduced at the link layer, as well as transmission delays in the radio access networks. For example, in GPRS, the RTT (Round Trip Time) varies between 1 and 1.5 seconds. Moreover, data rate is another limiting characteristic on these networks.

Some important parameters that have great effect on performance over wireless networks are the Maximum Segment Size (MSS), the maximum transmission window, the initial window size, the use of selective acknowledgements option and the time stamps option for more accurate RTT measurements.

We centered around the MTU¹; when considering the Bit Error Rate (BER) of a link, a small MTU increases the chance of successful transmission, since the possibility of frame damage is reduced. Moreover, large MTU will allow a quick growth in the TCP congestion window (in bytes), which can result in delay or retransmission.

For our GPRS connection, we have set the MTU to 472 bytes. We made this by using the next command when starting the GPRS connection:

```
ip link set ppp0 mtu 472
```

We chose this size of MTU because during the tests we observed that when the MTU of the Point-to-Point protocol²(ppp0) interface is about 500 bytes the response times are lower. If the transmission overheads become more significant as the packet size reduces [9], the only possible reason is that many transmission errors occurred and they must be retransmitted again; which entails an increase in the response times. In the next page, we present the different values of MTU we obtained, using the `ping` command, together with the rank of response time for each of them.

¹MTU refers to the largest packet that can be transmitted over a network, it is measured in bytes.

²Protocol that is used to establish the Internet connection via an habitual telephone or GPRS modem

MTU (bytes)	Rank of Response time (ms)
472	[209, 446]
500	[228, 442]
1000	[232, 778]
1500	[214, 774]

Table 7.4: Response time for different values of MTU

The maximum MTU for Point-to-Point protocol is 576 bytes. As can be observed, with large values for the MTU, the GPRS connection still works but the response time shoots up.

7.5 Data transfer costs

As we have already stated, we have defined two different applications to access data in the embedded computer in real time. In this section, we will present the advantage that this fact supposes.

The cost of one kilobyte of data transfer using information from CBB is about 0.0078DKK. LYKKETRONIC web application is quite heavy, with data size of 562KB. This implies that the price for uploading this page to the browser of an end user when the tractor is connected to Internet via a GPRS modem, is about 4.39DKK. If the users access real time data too many times, the costs for the farm will sharply increase.

Transferring this size of page can also be quite expensive, not only for the farm, but also for an end user using an expensive connection or a small device such as a PDA. Instead, if a lighter web page is used, the EC will end up transferring lesser kilobytes of data.

Although, in GPRS the time of the connection does not affect the price to pay, we must note that downloading the first web page using a GPRS connection needs about 22 seconds; on the contrary, displaying the lighter web page is much faster, it takes only about a second.

The next table is for log data transmission, as we can see, the size of the logs depends on the number of variables that have been logged. The meaning of each of the columns is explained below:

Variables	Uncompress log (KB)	Compress log (KB)	Price log (DKK)	Price 8 hours
1	19.64	0.70	0.01	0.23
2	37.89	0.95	0.01	0.30
3	38.01	0.92	0.01	0.30
4	38.12	0.93	0.01	0.30
5	57.76	1.11	0.01	0.35
6	57.87	1.28	0.01	0.41
7	77.00	1.27	0.01	0.41
8	97.36	1.44	0.01	0.46
9	117.50	1.82	0.02	0.58
10	136.62	1.74	0.02	0.56
11	157.30	1.91	0.02	0.61
12	176.25	2.44	0.02	0.78
13	194.33	2.77	0.02	0.89
14	212.93	3.41	0.03	1.09
15	213.04	3.31	0.03	1.06
16	233.21	3.62	0.04	1.16

Table 7.5: Size of the variables in compressed and uncompressed format, and price of transmission.

Figure 7.10 is a corresponding graph that shows how the size of the log increases as the number of variables increase. This also shows the advantage of making a compression after logging the files.

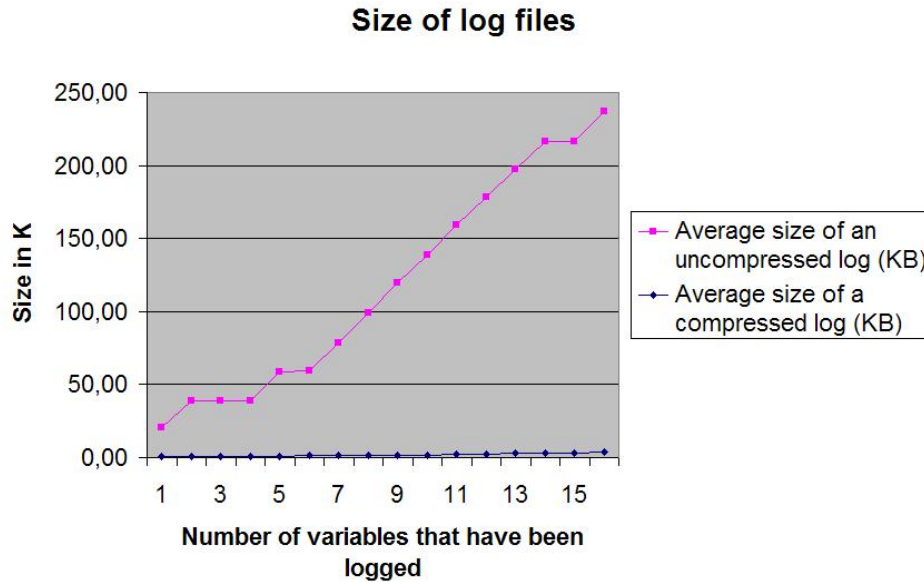


Figure 7.10: Size of the logs depending on the number of variables

The first column represent the number of variables logged, the second column represent the size of the uncompressed logs in (KB), the third column represent the size of the compressed logs in (KB), the fourth column represent the average price of transmitting a compressed log in (DKK), and the fifth column represent the average cost of transmitting compressed logs over a period of 8 hours.

The above results show that as the number of the uncompressed variables increase, the size also increases, as there is almost a linear increase in the file size as the number of variables increase. It also shows that even though we increase the number of variables to be logged, with a good compression algorithm, the file can easily be compressed to a reasonable size. This is because, compression algorithms perform better as the size of the file grows.

Chapter 8

User Guide

This section shows the system setup and how the users can interact with the system depending on the category of user stated in the analysis section. We have four different kinds of users and each one is able to do different tasks.

8.1 System setup

1. Start the database, located in the ACS, using the following command:

```
sudo /etc/init.d/mysql start
```

2. On the EC side, the scripts in charge of the wired/GPRS connection are automatically launched on start up of the system.

In order to start a GPRS connection, the modem must be connected to the EC and the PC suite mode must be chosen; otherwise, the EC will not recognize the device. After doing this, at the top left corner of the screen an icon will appear, this indicates that there is GPRS data connection. When the GPRS connection is started, two arrows, each in one direction will appear with discontinuous line. When the connection is fully established, this arrows will become fixed, as can be seen in the following screenshot.



Figure 8.1: Mobile phone screen when working as a GPRS modem

8.2 How to operate the system

When any user connects to the ACS, he sees the next screen:

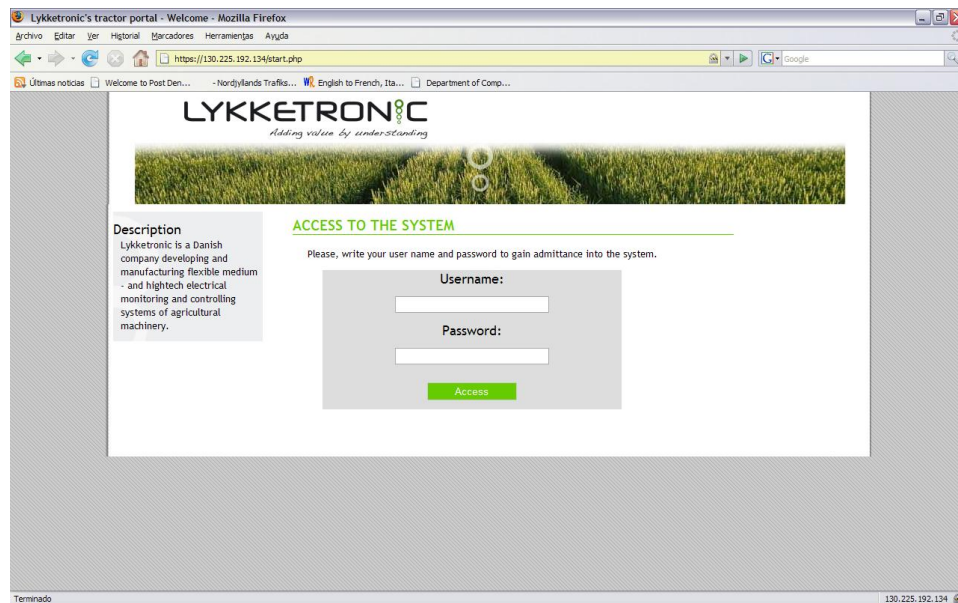


Figure 8.2: Login form

Afterwards, the user enters his name and password. With this information, the system knows if the user is allowed to interact with it or not. This point also, decides the interaction the different users can have with the system considering the different category they belong to.

After that, the user is allowed to continue. Then, the user can choose the farm that the tractor he wants to see belongs to. After selecting a farm, he then selects the tractor and click on the “Send” button.

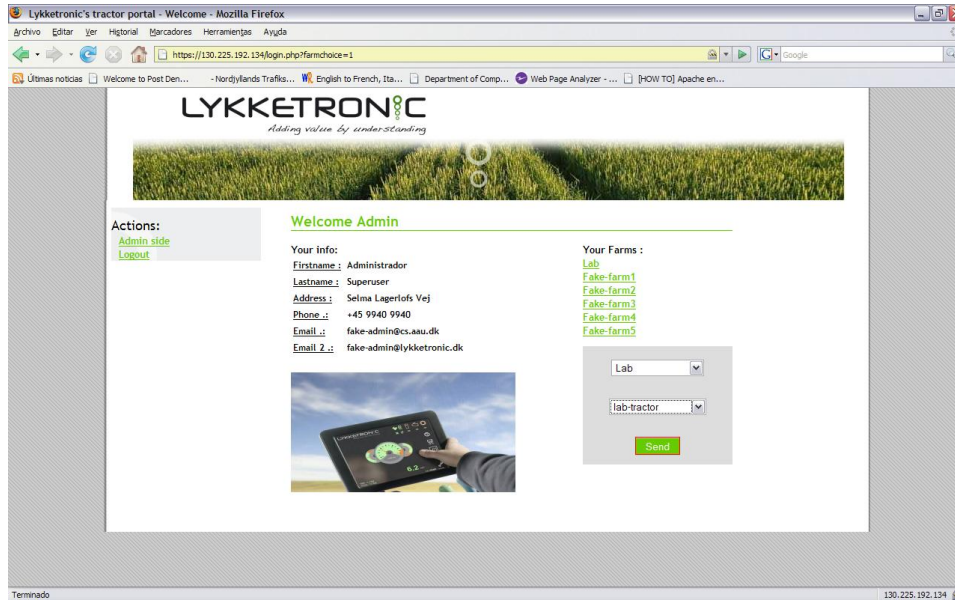


Figure 8.3: How to select a tractor

Afterwards the system shows the user a screen with the information belonging to the tractor selected. At that moment, the screen that the user sees is different; depending on whether the EC is connected via wired or GPRS.

If wired connection is available, the user will see the screen presented in picture A.1. Details can be found in the manual[11].

If GPRS connection is available, the user will see the screen presented on next page.

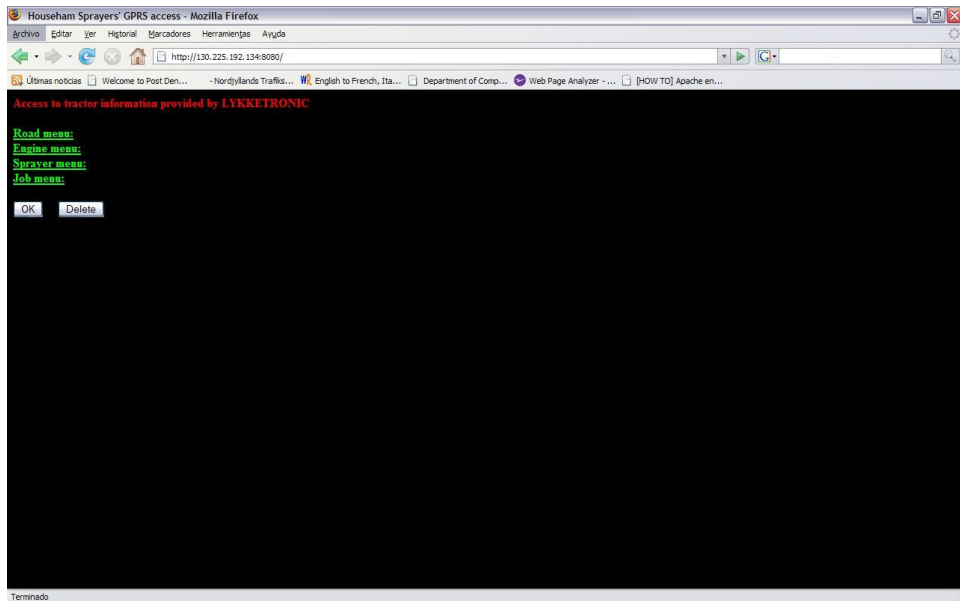


Figure 8.4: Light screen for GPRS connection

The different items can be expanded into menus; it is possible for a user to select the parameters he is interested in from the list and press the “OK” button.

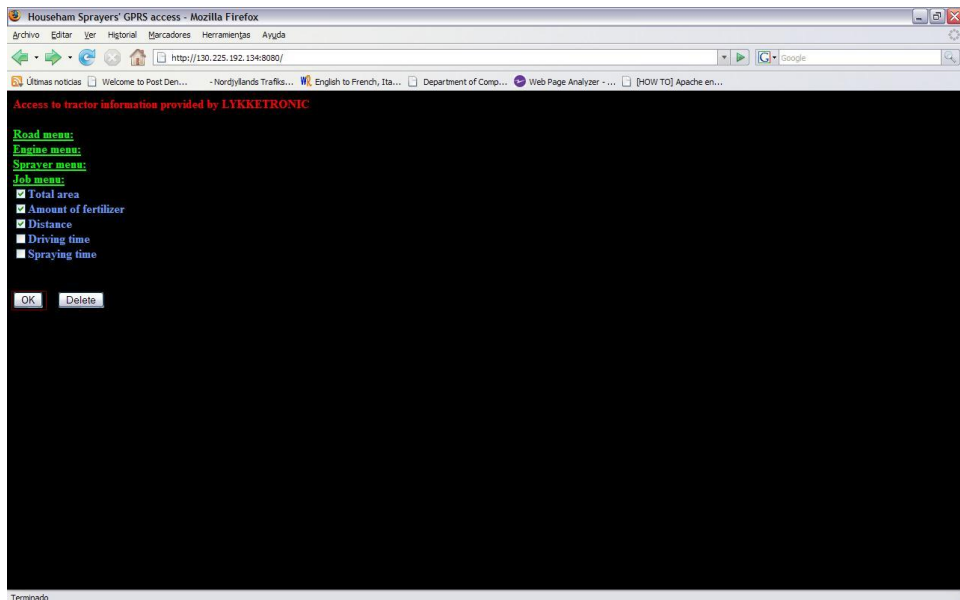


Figure 8.5: Expanded menus examples

Next, a snapshot of the values of the chosen parameters will be displayed. As an example, in figure 8.6 we can see the values of the parameters that were checked in figure 8.5.

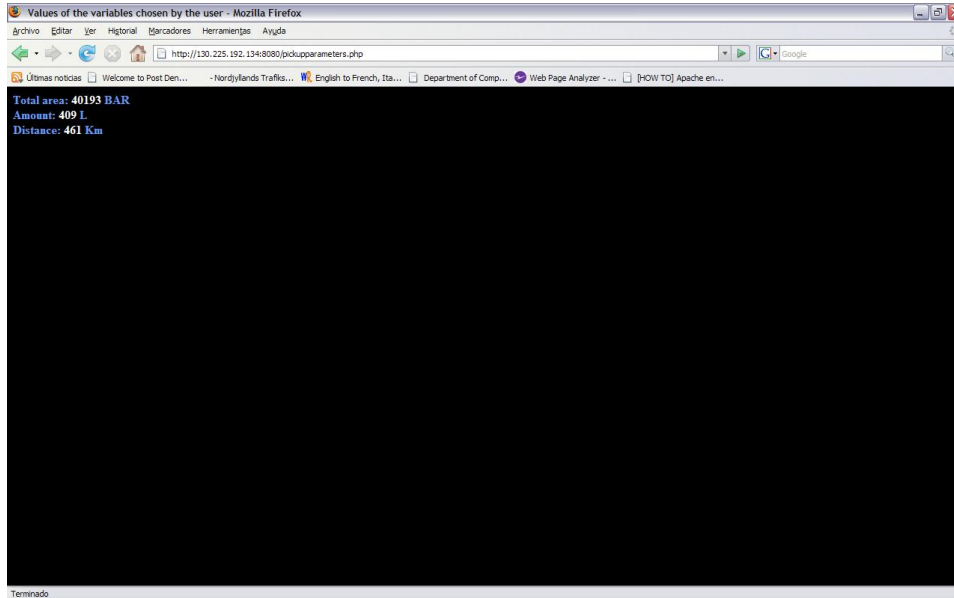


Figure 8.6: Snapshot of the instantaneous value os the selected variables

At this point we must divide this manual according to the different kinds of users we have.

8.2.1 Superuser

If the Superuser wants to access the administration tasks, he must select the link called “Admin side” that appear on the left of the screen.

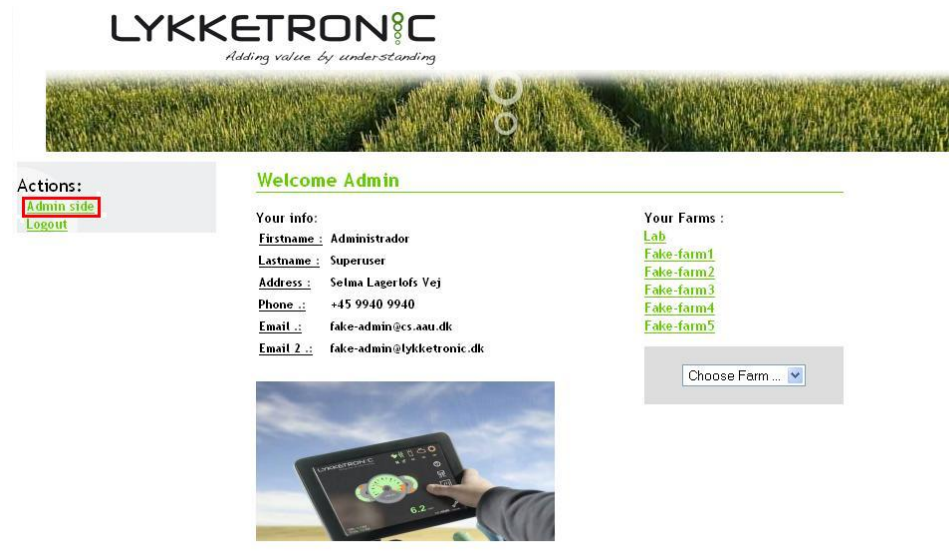


Figure 8.7: Admin side

When the Superuser clicks on that link, a new page appears. From this screen the superuser is able to manage:

Users. If the Superuser select that link, he is able to see the information that belongs to all the different users that belongs to the system. When the Superuser selects one user he sees the next screen with all the information about that user.

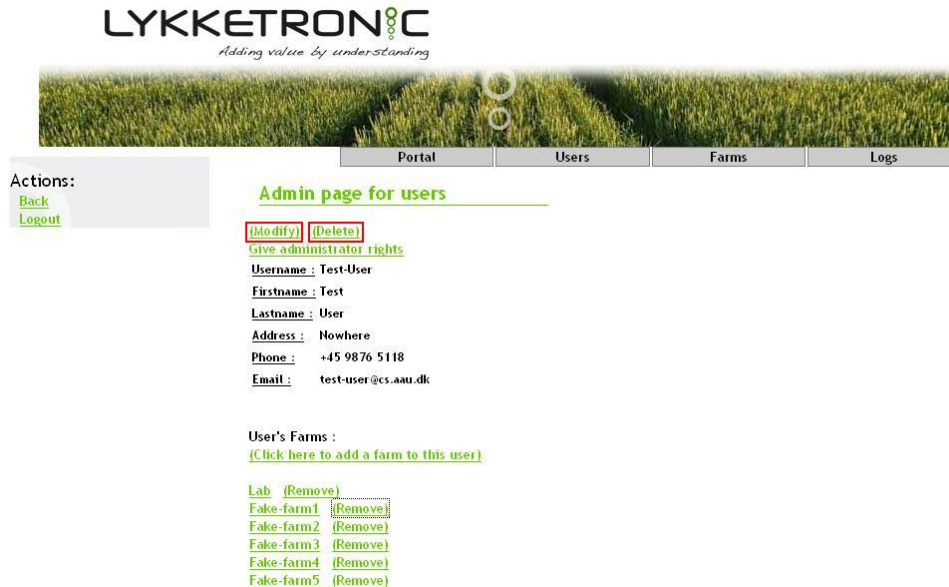


Figure 8.8: User detail

Afterwards the Superuser is able to see the user's information, modify or delete the user.

- If the superuser selects the link "Modify", he sees the next screen. Then he is able to modify whichever user's information he wants and press the button to accept the modification.
- If the "delete" link is selected, then the next screen appears and asks if he is sure about the action he is going to do (confirmation of destructive action). Then the user selects "Accept" and the system deletes the user which was chosen.
- The superuser is also able to add new users, to do that, the superuser must select "add new users". When he does this, he sees a form that he must fill in with the information about the new user.

Farms. If the superuser selects that link he is able to see the information that belongs to all the different farms which belong to the system. When the superuser selects one farm he sees the next screen with all the information about the farm. Afterwards, the superuser is able to see the farm's information, modify farm's, modify farm's information or delete the farm.

The screenshot shows the LYKKETRONIC web application interface. At the top, there is a navigation bar with tabs for 'Portal', 'Users', 'Farms', and 'Logs'. Below the navigation bar, the main content area is titled 'Admin page for farms'. On the left side, there is an 'Actions:' menu with links for 'Back' and 'Logout'. The main content area displays the following information:

- (Modify)** **(Delete)**
- Name :** Lab
- Address :** Selma Lagerslof Vej
- Phone :** +45 9940 9798

Below this information, there are sections for related users and tractors:

- Users related to this farm :**
 - [Worker](#)
 - [Farmer](#)
- Tractors related to this farm :**
 - [\(Click here to add a tractor to this farm\)](#)
 - lab-tractor - 0105 [\(Remove\)](#)
 - fake-tractor1 - 0106 [\(Remove\)](#)
 - fake-tractor2 - 0107 [\(Remove\)](#)

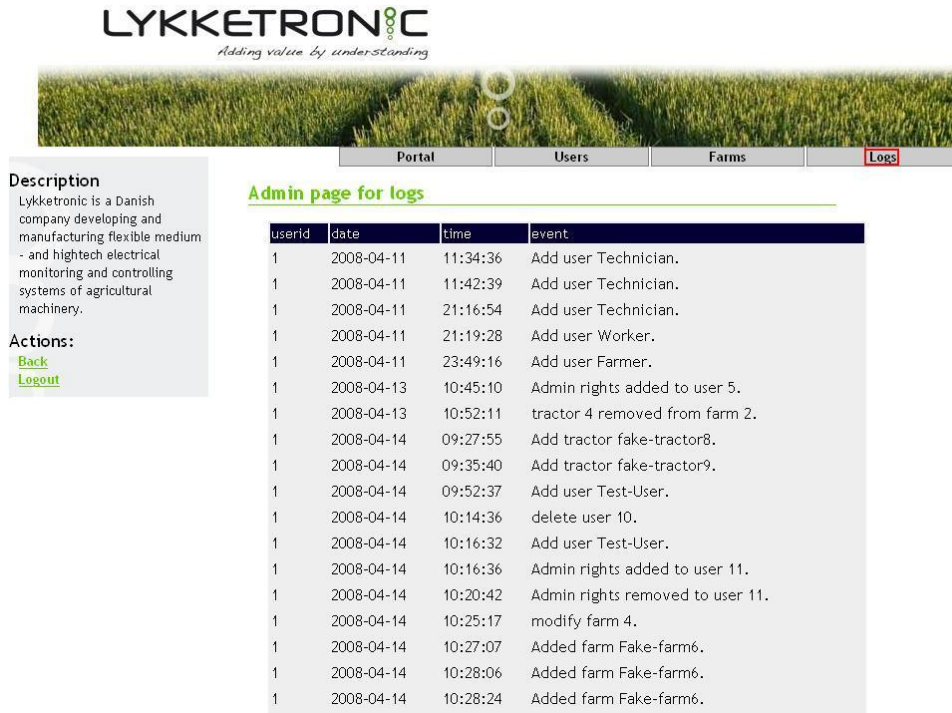
On the right side of the main content area, there is a sidebar menu with the following options:

- Add new farm
- Lab
- Fake-farm1
- Fake-farm2
- Fake-farm3
- Fake-farm4
- Fake-farm5

Figure 8.9: Farm details

- If the Superuser selects the link “Modify”, he sees the next screen. Then the superuser modify whichever farm's information he wants and select the button to accept the modification.
- If the Superuser selects the link “Delete” then he sees the next screen that ask him if he is sure about the action he is going to do. Then he selects “Accept” and the system deletes the farm which was chosen.
- The Superuser is also able to add new farms, to do that, the Superuser must select “add new farms”. When he does this, he sees a form that must be filled with the information of the new farm.

Logs. If the user select the “log” option, it is possible to see all the actions that have been made on the web site, as the next screen show us.



The screenshot shows the LYKKETRONIC web application interface. At the top, there is a navigation bar with four tabs: Portal, Users, Farms, and Logs. The 'Logs' tab is currently selected and highlighted in red. Below the navigation bar, the main content area is titled 'Admin page for logs' and contains a table with the following data:

userid	date	time	event
1	2008-04-11	11:34:36	Add user Technician.
1	2008-04-11	11:42:39	Add user Technician.
1	2008-04-11	21:16:54	Add user Technician.
1	2008-04-11	21:19:28	Add user Worker.
1	2008-04-11	23:49:16	Add user Farmer.
1	2008-04-13	10:45:10	Admin rights added to user 5.
1	2008-04-13	10:52:11	tractor 4 removed from farm 2.
1	2008-04-14	09:27:55	Add tractor fake-tractor8.
1	2008-04-14	09:35:40	Add tractor fake-tractor9.
1	2008-04-14	09:52:37	Add user Test-User.
1	2008-04-14	10:14:36	delete user 10.
1	2008-04-14	10:16:32	Add user Test-User.
1	2008-04-14	10:16:36	Admin rights added to user 11.
1	2008-04-14	10:20:42	Admin rights removed to user 11.
1	2008-04-14	10:25:17	modify farm 4.
1	2008-04-14	10:27:07	Added farm Fake-farm6.
1	2008-04-14	10:28:06	Added farm Fake-farm6.
1	2008-04-14	10:28:24	Added farm Fake-farm6.

On the left side of the page, there is a sidebar with the following content:

Description
 LykkeTronic is a Danish company developing and manufacturing flexible medium - and hightech electrical monitoring and controlling systems of agricultural machinery.

Actions:
[Back](#)
[Logout](#)

Figure 8.10: History

8.2.2 Technician

If the technician wants to access his own section he must select the link called “Technician side”.

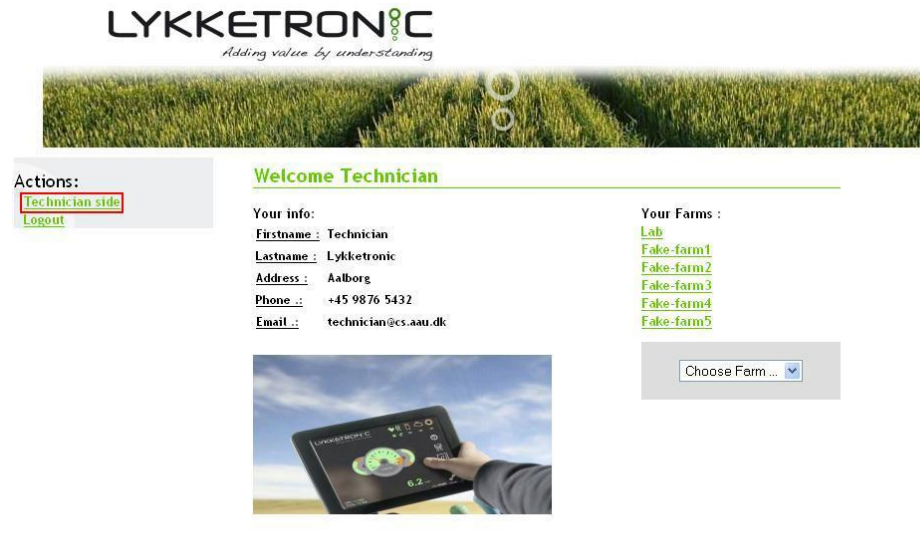


Figure 8.11: Technician side link

The technician is able to access the information belonging to the different farms and its tractors.

8.2.3 Farmer

If the farmer wants to see his own section, he must select the link called “Farmer side”. When the farmer clicks on that link, he sees the next screen.

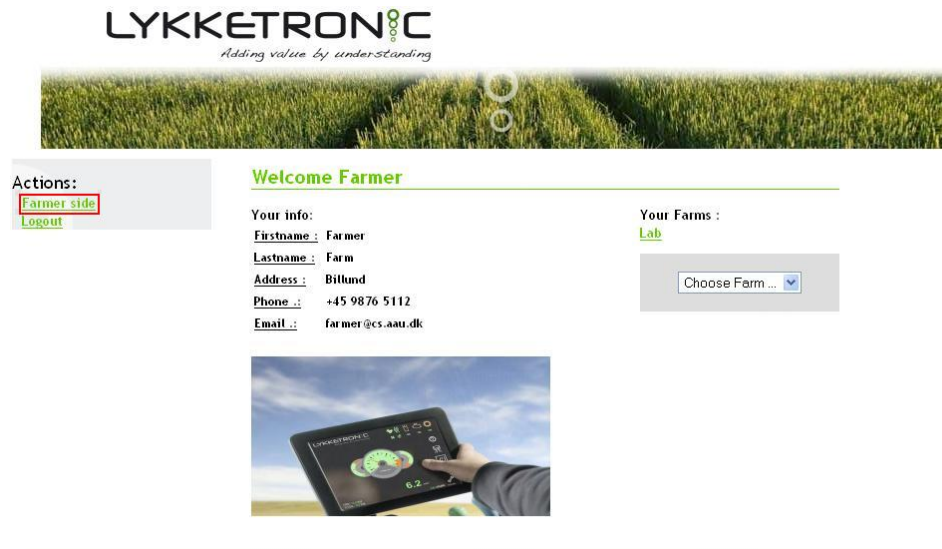


Figure 8.12: Farmer side link

In this screen the farmer is able to see his own farms and the users who works with those farmers. He can also add new tractors to his farm.

8.2.4 Worker

This user does not own a side; he is only able to see the farm which he works in and the tractors he is related to.

Chapter 9

Conclusion and Future Work

After concluding this project we can confirm the possibility of connecting an embedded computer to the Internet via different technologies, specifically, wired and GPRS. It is also possible to make a switch between the technologies in order to choose, at any given moment, the connection with the best price/performance ratio, amongst those that are available. The network switch design is a software based solution that could be easily portable to any other type of mobile node. Moreover, the results show that it is possible to access this embedded device from anywhere in the world, using a standard browser.

Also, a logging system has also been implemented, which demonstrates that the transfer of data files in batches from a embedded computer to a centralized node is possible, and that this transmission is faster and cheaper if the files are compressed before sending.

Finally, we have to state that we had a problem that we were unable to solve, the issue of WIFI configuration in the device. It was not possible to install the available WIFI dongle in the current operating system - CentOS, in the embedded computer. So we suggest a possible change of the operating system, preferably to Ubuntu. Once, the change has been made, with the right configuration, the present design can easily be adapted to use WIFI instead of the wired connection.

In next section, we will have a look to the improvements that our project has achieved with regards to the previous work, and also, some other possible ideas that could be interesting to implement in future.

9.1 Main improvements in the present project

- In the previous work, the file `launchall.py` was set to run on start up in run level 3, but it did not update the port attribute in the ACS database after start up. Now, this problem has been fixed and the port database is correctly updated.
- Only a unidirectional switch from wired to GPRS was possible before and when a mobile phone was plugged and the wired cable plugged off, there was loss of connection of the system to the Internet. Bidirectional switch is now possible, and with many combinations.
- The problem with the instability of the GPRS connection has been fixed.
- Drastic drop in the number of lines of code (LOC), on the side of maintainability and later reutilization.
- Before the kick off of this second project, we took measurements of the time taken to switch a connection, in order to be able to calculate the improvement achieved at the end. The time used for making a switch from the wired connection to GPRS connection, was in average about 90 secs, and now it is 30 secs., which is a supposed improvement of about 300%.
- The `wirewatchdog.py` script created a lot of children that were not killed later, so it caused wastage of CPU, resulting in slow switching time.
- The design of the database has changed to allow different sorts of roles within the system. Each user has different privilege, as stated in the Software Requirements Specification, for accessing, modifying, adding or deleting some of the contents stored on persistent storage.
- Now, it is also possible to store several telephone numbers, electronic mails, etc, for the same user.
- Sensitive data, such as user passwords, has been hashed to make it more difficult to see them either using brute force search or other procedures.
- In the previous system, the entire system simulation was done inside AAU network, hence the ACS was behind AAU'router. Now the ACS has a public IP address and a user is able to connect to it from anywhere. This is also how the ACS will be used in the real scenario.

9.2 Future work

Although we have developed a system that can provide remote access to data in a mobile device, extensions can be added through further work as discussed below.

- Possible implementation of a webcam with the right linux drivers can be made in the EC. With this, it is possible to remotely access video data and to also remotely control the camera using the right functionalities.
- Possible use of a GPS system to remotely control the tractor. One possible way to achieve this is by using a recording GPS system to record the tractor's route while working. The tractor can then be programmed to follow the same route. It can use this for example, to control spraying of the fertilizers. Programming the tractor routes can help to save a lot of money.

Bibliography

- [1] Advanced Digital-Logic Inc: <http://www.adlogic-pc104.com/products/systems/datasheets/MICROSPACE-PCX48.pdf>. Last visit: 22nd February 2008.
- [2] Aalborg University - Research: [http://vbn.aau.dk/research/landit\(5544401\)](http://vbn.aau.dk/research/landit(5544401)). Last visit: 12nd March 2008.
- [3] Buron, Anthony; Cothureau, Nicolas; Delaite, Guillaume; Eskilden, Jacob; Gourdin, Edouard; Monsonogo, Olivier; Ogini Nielsen, Faith Oziofu and Oteo Ovejero, Elisa; *Remote Access for a Wireless System*. Aalborg University, 2007.
- [4] BZIP.org: <http://www.bzip.org/>. Last visit: 26th April 2008.
- [5] Conallen, Jim; *Building Web Applications with UML Second Edition*. Boston, 2002.
- [6] EherAper: <http://etherape.sourceforge.net/>. Last visit: 18th April 2008.
- [7] Basili, Victor; Caldiera, Gianluigi and Rombach, Dieter; *The Goal Question Metric Approach*. University of Maryland.
- [8] DataCompression.info: <http://datacompression.info/PPM.shtml>. Last visit: 25th May 2008.
- [9] Ehlers, Niels; Stuckmann, Peter; Wouter, Bianca; *GPRSTraffic Performance Measurements*. Aachen University of Technology (Germany).
- [10] GSM World: <http://www.gsmworld.com/technology/gprs/index.shtml>. Last visit: 11th March 2008.
- [11] Househam Sprayers Limited; *TMC Computer. Operators Manual - Phase 1*.
- [12] IEEE; Std. 830-1993. *Recommended Practice for Software Requirements Specification*.

- [13] IEEE; Std. 802.11-2007. *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks.*
- [14] IEEE; Std. 801.11i-2004. *IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 6: Medium Access Control (MAC) Security Enhancements.*
- [15] IEEE; Std. 802.16-2001. *IEEE Standard for Local and Metropolitan area networks.*
- [16] LYKKETRONIC: <http://www.lykketronic.dk>. Last visit: 23rd February 2008.
- [17] Madwifi.org: <http://madwifi.org>. Last visit: 18th April 2008.
- [18] NDSIWrapper: <http://ndiswrapper.sourceforge.net/joomla>. Last visit: 18th April 2008.
- [19] MRTG: <http://www.mrtg.com/>. Last visit: 18th April 2008.
- [20] Ntop.org: <http://www.ntop.org/>. Last visit: 18th April 2008.
- [21] Mexperts: <http://www.presseagentur.com/digitallogic>. Last visit: 22nd February 2008.
- [22] TcpDump.org: <http://www.tcpdump.org>. Last visit: 18th April 2008.
- [23] Webexperto.com: <http://www.webexperto.com/articulos/art/179/codificar-contrasenas-con-md5/>. Last visit: 8th March 2008.
- [24] SourceGuardian.com: <http://www.sourceguardian.com>. Last visit: 15th May 2008.
- [25] Wireshark.com: <http://www.wireshark.com/>. Last visit: 18th April 2008.
- [26] XMLvalidation.com: <http://www.xmlvalidation.com/>. Last visit: 19th April 2008.

Appendix A

Details about the Embedded Computer

A.1 Hardware - Level Description

- Celeron M-800MHz
- 256MB DDR-RAM
- 6x USB V2.0 (2 front, 4 rear)
- LAN Ethernet 10/100 Base-T
- LPT, COM1 and COM2
- Video Channel-1 analog CRT
- Video Channel-2 digital DVI-D and LVDS
- Audio-In/Out Stereo, Mic In
- 1GB Compact Flash card Silicon Systems
- Dimensions: 160 x 190 x 66 mm

A.2 Software - Level Description

- The operating system running in the EC is Linux CentOS 4.4, a Linux distribution that is based on Red Hat Enterprise Linux (RHEL).
- Some of the installed packages and tools are - OpenSSH, IPtables, dhclient, pppd, Apache Web Server Version 2.0.52 and PHP 5 amongst others.

A.3 Human Machine Interface (HMI)

For the hardware system, a touch screen is the User Interface to the EC. A USB keyboard can also be attached instead of using the pop-up keypad that came along with the touch screen.

Below is screen shot of the Touch Screen, a click of an icon gives access to information of a subset of the activities in the tractor.



Figure A.1: A screen shot of the touch screen

Description of the different icons on the Touch Screen:

engine oil pressure low
fuel level low
front angle sensor no signal
rear angle sensor no signal

Power Down Button - When this unit is pressed, a pop-up window appears that can allow us to shut down the system. Note: Warning messages are indicated in a red box at the same top left hand corner of the screen.



Boom Lock Symbol 90° - For monitoring the angle between a sprayer mast and the booms to ensure that the booms are kept as close as possible to 90° at all times.



Headland Assistant - is used to aid the operator in different conditions for example, when on an uneven terrain or exceptionally tall crops. It works by lifting and lowering the booms to a predetermined height.



Hydraulic - This is used for displaying the operational status of the hydraulic. If operating condition is normal, OK is displayed beneath this icon and if a fault occurs, a red warning triangle is displayed beneath this icon.



Engine - This is used for displaying the operational status of the Engine such as the battery voltage, engine oil temperature, engine hours, etc. If operating condition is normal, OK is displayed beneath this icon and if a fault occurs, a red warning triangle is displayed beneath this icon.



Transmission Status - When this icon is pressed transmission status such as oil level, current pressure will be shown. If operating condition is normal, OK is displayed beneath this icon and if a fault occurs, a red warning triangle is displayed beneath this icon.



Help Menu - Pressing this icon allows the operators manual to be viewed as a PDF document.



Road Screen - Pressing this icon will automatically cause the 'Road-Mode' screen to appear on the screen. On the 'Road-Mode' screen, information like total distance traveled, engine speed, fuel level, etc can be seen.



Sprayer Control Screen - Pressing this icon will display the 'Spray Controller Screen'. This is used for controlling the spray, it can be used to put the spray in either auto or manual mode, set the desired application rate, etc.



Operator Setup Menu - When this icon is pressed, it enters the operator setup. It is designed to contain most frequently adjusted settings eg for the sprayer set up, it is used for setting up the wheel, density of spraying, nozzle setup, etc.



Technical Setup - This is for the technical machine setup. These settings should rarely need adjustment once setup correctly. Used for sprayer setup, it is used for setting up things like the flow meter, nozzle distance, wheel steering setup, show incoming packets from the CANBus, etc.

A.4 Examples of data to be logged in the EC

Some examples of data that can be interesting to log in the system are: Speed- the current speed at which the tractor is driving, Total trip- the total time in hours which the tractor has spent while carrying out a job, Battery voltage, Engine hours- the number of hours for which the engine has been running, Fuel level, Tank content- this gives reading of the current fuel content in the tank, Application rate- this is for indicating the rate at which a tractor is applying fertilizers, seedling, water, etc., Spray pump, Spray line pressure, Total area- the total area covered at the end of each job, Amount of fertilizer, and Distance.

For further information and examples, please consult the Operator's Manual[11].

Appendix B

Scripts' source listings

In this appendix, a copy of the main scripts and configuration files is included. Source code of the web applications is not presented because of its large size. We encourage the reader to consult the CD - ROM attached to this report to have access to all the code developed throughout the project.

B.1 Programs and configuration files on the EC

B.1.1 wvdial.conf

```
[Dialer Defaults]
Modem = /dev/ttyACM0
Baud = 460800
Init1 = ATZ
Init2 = ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0
Init3 = AT+CGDCONT=1,"IP",,"internet"
ISDN = 0
Modem Type = USB Modem
Phone = *99***1#
Username = sonofon
Password = sonofon
```

B.1.2 SshTools.py

```
import os
import commands

def Create_Ssh_Tunnel(forwardType, localPort, remoteHostIP, remotePort,
sshServerPort, sshUser, sshServerIP):
    cmd = "ssh -fNC -" + forwardType + " -L" + localPort + ":" + remoteHostIP +
    ":" + remotePort + " -L" + "-p" + sshServerPort + " -L" + sshUser +
    "@" + sshServerIP
    status = os.popen(cmd)
    print "Status: ", status

def Stop_Ssh_Tunnel():
    cmd = "killall ssh"
    status = os.popen(cmd)
    print "Status: ", status
```

B.1.3 info.conf

```
#Name of BlackBox  
[BlackBox]  
TractorName=lab-tractor
```

```
#Proxy Section  
[Proxy]  
IP=130.225.192.134  
User=ec  
Port=2222  
PortUpTract=56789
```

```
#Tunnel_Ports Section : List of the port the embedded computer  
#should use for creating tunnels  
[Tunnel_Ports]  
P1=8080  
P2=8081  
P3=8082
```

```
#Proxy-Gateway Section : Specific for AAU Network  
[Proxy-Gateway]  
IP=homer.cs.aau.dk  
User=isabel  
Port=22
```

```
#Information concerning the Access Control Server DB  
[DB_Info]  
DBhost=130.225.192.134  
DBuser=justinbridoux  
DBpasswd=saucisson  
DBname=d606a
```

B.1.4 connectionswitch.py

```
#!/usr/bin/env python
#-*- coding:utf8 -*-

#Import library
import os, logging, commands, httplib
from configobj import ConfigObj
import ConnectionTools
import SshTools
from time import sleep

#Name of script
rootDir = "/root/scripts"
progname="connectionsswitch"

#Variables
opentunnel = 0
confFile = rootDir + "/info.conf"

#Read conf file to get DB info
config = ConfigObj(confFile)
sectionName = config['BlackBox']
tracname = sectionName['TractorName']

sectionProxy = config['Proxy']
iproxy = sectionProxy['IP']
proxyuser = sectionProxy['User']
proxyport = sectionProxy['Port']
portuptractproxy = sectionProxy['PortUpTract']

sectionPort = config['Tunnel_Ports']
porta = sectionPort['P1']
portb = sectionPort['P2']
portc = sectionPort['P3']

sectionGW = config['Proxy_Gateway']
iphomer = sectionGW['IP']
gwuser = sectionGW['User']
gwport = sectionGW['Port']

sectionDB = config['DB_Info']
dbhost = sectionDB['DBhost']
dbuser = sectionDB['DBuser']
dbpasswd = sectionDB['DBpasswd']
dbname = sectionDB['DBname']

def wiredtunnel():
    #Try to open the tunnel for the wired connection
    SshTools.Create_Ssh_Tunnel
    ("L", proxyport, iproxy, gwport, gwuser, iphomer)
    SshTools.Create_Ssh_Tunnel
    ("R", porta, "127.0.0.1", "80", proxyport, proxyuser, "127.0.0.1")
    SshTools.Create_Ssh_Tunnel
```

```

("L", portuptractproxy, "127.0.0.1", portuptractproxy, proxyport,
 proxyuser, "127.0.0.1")

def gprstunnel():
    #Try to open the tunnel for the GPRS connection
    SshTools.Create_Ssh_Tunnel
    ("L", proxyport, ipproxy, gwport, gwport, gwuseriphomer)
    SshTools.Create_Ssh_Tunnel
    ("R", portb, "127.0.0.1", "81", proxyport, proxyuser, "127.0.0.1")
    SshTools.Create_Ssh_Tunnel
    ("L", portuptractproxy, "127.0.0.1", portuptractproxy, proxyport,
    proxyuser, "127.0.0.1")

def Proxy_DB_Update_HTTP(ipProxy, portUpTractProxy, tractorName, port):
    #Try to update on the database the port by wich a tractor is
    #connected to the proxy through
    os.popen
    ('elinks \ 'http://' + ipProxy + ':' + portUpTractProxy +
    '/updateTractor.php?name=' + tractorName + '&port=' + str(port) + '\ ')
    sleep(0.2)
    os.popen('killall elinks ')

def startwired():
    #Try to start the Ethernet connection
    #Try to stop the tunnel
    SshTools.Stop_Ssh_Tunnel()
    #Kill GPRS
    os.popen("killall wvdial")
    os.popen("route del *")

    #Make sure that eth0 is well connected on the web
    os.popen("killall dhclient")
    #os.popen("dhclient")
    cmd = "dhclient"
    status, output = commands.getstatusoutput(cmd)
    print status, output
    sleep(3)
    if ConnectionTools.connection_available("www.lykketronic.dk") == True:
        tunnelExistence=ConnectionTools.test_ssh_tunnel()
        if tunnelExistence == False:
            wiredtunnel()
            Proxy_DB_Update_HTTP(ipproxy, portuptractproxy,
            tracname, porta)
            print "The ethernet connection has been started"
    else:
        print "The ethernet connection could not be started"

def startgprs():
    #Try to start the GPRS connection
    #Try to stop the tunnel
    SshTools.Stop_Ssh_Tunnel()
    #Kill wired connection
    os.popen("killall dhclient")

```

```

#ppp0 has a route by default; if we let it as it was
#all the traffic will try to go to eth0
os.popen("route_del_default")
sleep(0.4)
#Try to create GPRS SSH-tunnel
pidParent=os.getpid()
a=os.fork()
if a==0:
    os.popen("wvdial_&*&")
else:
    sleep(15)
    cmd="kill_"+str(pidParent)
    print cmd
    gprsConnectionAvailable=ConnectionTools.test_modem_gprs()
    if gprsConnectionAvailable==True:
        tunnelExistence=ConnectionTools.test_ssh_tunnel()
        if tunnelExistence==False:
            gprstunnel()
            Proxy_DB_Update_HTTP(ipproxy, portuptractproxy,
            tracname, portb)
            os.popen("ip_link_set_ppp0_mtu_472")
        else:
            print "No_connection_could_be_started"

def noconnection():
    #Kill all the vestiges of Internet connections
    #Try to stop the tunnel
    SshTools.Stop_Ssh_Tunnel()
    #Kill possible connections
    os.popen("killall_dhclient")
    os.popen("killall_wvdial")
    Proxy_DB_Update_HTTP(ipproxy, portuptractproxy, tracname, 0)
    if tunnelExistence == False:
        wiredtunnel()
        Proxy_DB_Update_HTTP(ipproxy, portuptractproxy,
        tracname, porta)
        print "The_ethernet_connection_has_been_started"
    else:
        print "The_ethernet_connection_could_not_be_started"

def startgprs():
    #Try to start the GPRS connection
    #Try to stop the tunnel
    SshTools.Stop_Ssh_Tunnel()
    #Kill wired connection
    os.popen("killall_dhclient")
    os.popen("route_del_default")
    sleep(0.4)
    #Try to create GPRS SSH-tunnel
    pidParent=os.getpid()
    a=os.fork()
    if a==0:
        os.popen("wvdial_&*&")

```

```

else:
    sleep(15)
    cmd="kill "+str(pidParent)
    print cmd
    gprsConnectionAvailable=ConnectionTools.test_modem_gprs()
    if gprsConnectionAvailable==True:
        tunnelExistence=ConnectionTools.test_ssh_tunnel()
        if tunnelExistence==False:
            gprstunnel()
            Proxy_DB.Update_HTTP(ipproxy , portuptractproxy ,
            tracname , portb)
            os.popen("ip_link_set_ppp0_mtu_472")
    else:
        print "No_connection_could_be_started"

def noconnection():
    #Kill all the vestiges of Internet connections
    #Try to stop the tunnel
    SshTools.Stop_Ssh_Tunnel()
    #Kill possible connections
    os.popen("killall_dhclient")
    os.popen("killall_wvdial")
    Proxy_DB.Update_HTTP(ipproxy , portuptractproxy , tracname , 0)

```

B.1.5 ConnectionTools.py

```
#!/usr/env python
#coding:utf8-*-
import httplib, os, commands

def test_connection_ethernet():
    #Checks if the Ethernet cable is plugged on
    cmd="ethtool eth0 | grep \"Link detected: yes\""
    status, output = commands.getstatusoutput(cmd)
    if status == 0:
        if output != '':
            connection_ethernet = True
        else:
            connection_ethernet = False
    else:
        connection_ethernet = False
    return connection_ethernet

def test_modem_gprs():
    #Checks if the interface pp0 is brought up
    cmd="ifconfig | grep ppp0"
    status, output = commands.getstatusoutput(cmd)
    print output
    if status == 0:
        if output != '':
            modem_gprs = True
        else:
            modem_gprs = False
    else:
        modem_gprs = False
    return modem_gprs

def test_ssh_tunnel():
    #Checks if an SSH tunnel is already opened
    cmd="ps aux | grep isabel | wc -l"
    status, output = commands.getstatusoutput(cmd)
    if status == 0:
        difference=int(output)-2
        if difference > 0:
            ssh_tunnel = True
        else:
            ssh_tunnel = False
    else:
        ssh_tunnel = False
    return ssh_tunnel

def test_connection_phone():
    #Checks if the mobile phone is plugged to the embedded computer
    cmd = "lsusb | grep Nokia"
    status, output = commands.getstatusoutput(cmd)
    if status == 0:
        if output != '':
            connection_phone = True
```



```

        print "Mobile_phone_plugged_on_PC_Suite_Mode"
    else:
        connection_phone = False
else:
    connection_phone = False
return connection_phone

def test_ip_address():
    #Checks the IP address of the eth0 interface
    cmd="ifconfig_eth0_|_grep_inet"
    status, output = commands.getstatusoutput(cmd)
    if status == 0:
        if output != '':
            ip_address = True
        else:
            ip_address = False
    else:
        ip_address = False
    return ip_address

def test_dhclient():
    #Checks if the dhclient is already running in the embedded computer
    cmd="ps_aux_|_grep_dhclient"
    status, output = commands.getstatusoutput(cmd)
    #print "Output: ", output
    if status == 0:
        if output != '':
            status_dhclient = True
        else:
            status_dhclient = False
    else:
        status_dhclient = False
    return status_dhclient

def test_connection(websiteURL):
    #Check if it is possible to connect to a given web address
    conn = httplib.HTTPConnection(websiteURL)
    conn.request('GET', 'http://' + websiteURL + '/')
    resp = conn.getresponse()
    conn.close()
    if resp.reason == 'OK' or resp.reason.count("Moved") != 0:
        connection_status = True
    else:
        connection_status = False
    return connection_status

def connection_available(websiteURL):
    #Check if exists an Ethernet connection available
    if test_connection_ethernet() == True:
        if test_dhclient() == True:
            if test_ip_address() == True:
                if test_connection(websiteURL) == True:

```

```
                return True
            else:
                print "Not_able_to_reach_the_URL_specified"
                return False
        else:
            #logging.info("No IP Address")
            print "No_IP_Address"
            return False
    else:
        #logging.info("DHCP client issue")
        print "DHCP_client_issue"
        return False
else:
    #logging.info("Ethernet Cable Not Plugged")
    print "Ethernet_Cable_Not_Plugged"
    return False
```

B.1.6 launchall.py

```

#!/usr/bin/env python
# -*- coding: utf8 -*-

#Import library
import os, MySQLdb, logging, commands, httplib
from configobj import ConfigObj
import ConnectionTools
import connectionswitch
import SshTools
from time import sleep
#Name of script
programe = "launchall"
rootDir = "/root/scripts"

#Ethernet connection first
cmd = "dhclient"
status, output = commands.getstatusoutput(cmd)
print status, output
sleep(2)
if ConnectionTools.connection_available("www.lykketronic.dk") == True:
    #if a connection is established, start the ethernet connection procedure
    connectionswitch.startwired()
else:
    #Try to launch the GPRS connection
    connectionswitch.startgprs()

```

B.1.7 wirewatchdog.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

#####
LOGFILE = '/var/log/wirewatchdog.log'
#####

import sys, os, locale, datetime
import ConnectionTools
import connectionswitch
from time import sleep

#var
wirePreviousStatus = False

class Log:
    """file like for writes with auto flush after each write
    to ensure that everything is logged, even during an
    unexpected exit."""
    def __init__(self, f):
        self.f = f
    def write(self, s):
        self.f.write(s)
        self.f.flush()

def main():
    #change to data directory if needed
    os.chdir("/root/scripts")
    #redirect outputs to a logfile
    sys.stdout = sys.stderr = Log(open(LOGFILE, 'a+'))

    wirePreviousStatus = ConnectionTools.test_connection_ethernet()
    telephonePreviousStatus = ConnectionTools.test_connection_phone()

    #main loop
    while True:
        sleep(0.4)
        wireCurrentStatus = ConnectionTools.test_connection_ethernet()
        telephonePlugged = ConnectionTools.test_connection_phone()

        if (wireCurrentStatus != wirePreviousStatus)
        or (telephonePlugged != telephonePreviousStatus):
            wirePreviousStatus = wireCurrentStatus
            telephonePreviousStatus=telephonePlugged
            if wireCurrentStatus == True:
                connectionswitch.startwired()
            else:
                if telephonePlugged == True:
                    connectionswitch.startgprs()
                else:
                    connectionswitch.noconnection()
```

```
if __name__ == "__main__":
    try:
        pid=os.getpid()
        c = os.fork()
        if c > 0:
            # exit first parent
            sys.exit(0)
            cmd="kill_" +str(pid)
    except OSError, e:
        print >>sys.stderr, "fork_#1_ failed:_%d_(%s)" % (e.errno, e.strerror)
        sys.exit(1)

# start the daemon main loop
main()
```

B.1.8 configuration.h

```

/*****
 *          Title: configuration.h
 *   Description: header that includes the libraries that are necessary to do
 *                the logging process and also another configuration information
 *                Date: 2nd May 2008
 *                Author: group d606a – Software Systems Engineering 4
 *                       (Aalborg University)
 *****/

#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<signal.h>
#include<sys/time.h>
#include<string.h>
#include <unistd.h>
#include "var_id.h"

#define TRUE          1
#define FALSE        0
#define NUM_VARIABLES 16
#define VARIABLES_O  12 //Number of variables with a high frequency of change

typedef struct{

    char *name;
    unsigned int id;
    char *units;

}variable;

variable myMatrix[NUM_VARIABLES][2];
FILE *filew;
char filename[50];
char datew[9];
time_t t;
struct tm* today;
unsigned int timesno=0;
char callgetValues[30];
int valueReturned;

//-----

char tractorname[15]="lab-tractor";
unsigned int tractorid=1;

//-----

```

B.1.9 main.c ↦ Executable: getValues

```

/*****
 * DESCRIPTION: Program in C language that returns the value of a specified
 *              variable that has been passed as parameter.
 * PROJECT: Analysis and improvement of a remote system
 * GROUP: Group d606a – SSE4(Aalborg University)
 * DATE: 31st March 2008
 *****/

/***** Includes *****/
#include "PROJECT.h"
#include "compiler.h"
#include "PROJECT.h"
#include <stdio.h> /* printf */
#include <string.h> /* strerror */
#include <errno.h> /* errno */
#include <unistd.h> /* getopt, sleep */
#include <stdlib.h> /* atoi, atexit, EXIT_XXX */
#include <signal.h> /* signal */
#include <err.h> /* err */
#include <sched.h> /* sched_yield */
#include <time.h>M
#include <stddef.h> /* definition of NULL */
#include <sys/time.h> /* definition of timeval struct and prototyping of gettimeofday */
#include "var_id.h" /* List of LYKKETRONIC variables */
#include "main.h"

/***** Prototype *****/
int shm_rw( int var_id, BOOL write_m, SINT32 *value );

/***** Vars *****/
struct timeval tp;
MILLI_SEC t1 =0;
int rtn;

/***** Private func *****/

/*-----*/
extern MILLI_SEC TimeNow(void)
{
    rtn=gettimeofday(&tp, NULL);
    t1=(MILLI_SEC)((1000*(MILLI_SEC)tp.tv_sec)+(tp.tv_usec)/1000);
    return (t1);
}
/*-----*/

/*****
 * Name: main (main function)
 * Goal: show via standard output the value of a determined variable
 * Input: one parameter —> the variable whose value is wished to obtain
 *****/

```

```

* Output: Value of the variable passed as argument is displayed on standard
*         output
*****/

int main(int argc, char *argv[])
{
    SINT32 value_return;^M

    //Read from the shared memory
    shm_rw( atoi(argv[1]), SHM_READ, &value_return );

    printf("%ld", value_return);
    usleep(10000);

    return(1);
}

```

B.1.10 sftpdialogue.sh

```

#!/usr/bin/expect

spawn sftp justinbridoux@130.225.192.134
expect "sftp>"
send "lcd _logging\r"
expect "sftp>"
send "cd _lykketronic-logs\r"
expect "sftp>"
send "put _*\r"
expect "sftp>"
send "bye\r"
expect eof

```


B.1.11 logging.c

```

/*****
 *           Title: logging.c
 *   Description: program that will be running in background in the embedded
 *                computer of each of the tractors, picking up every five seconds
 *                and saving in a file the instantaneous values of the variables
 *                that a user has selected via a web interface. Every fifteen
 *                minutes the file (in XML format) will be closed and sent to the
 *                Access Control Server, where it will be processed.
 *           Date: 28th April 2008
 *           Author: group d606a – Software Systems Engineering 4 (Aalborg University)
 *****/

#include "configuration.h"

/*****
 *           Name: initialization
 *           Goal: inicialize the matrix of variables that the user wants to log
 *           Input: void
 *           Output: void
 *****/

void initialization(void){

    myMatrix [0] [0].name=" Speed" ;
    myMatrix [0] [0].id=VAR_ID_SPEED;
    myMatrix [0] [0].units="Km/h" ;

    myMatrix [1] [0].name=" Total_trip" ;
    myMatrix [1] [0].id=VAR_ID_TRIP_COUNTER_KM;
    myMatrix [1] [0].units="Km" ;

    myMatrix [2] [0].name=" Amount" ;
    myMatrix [2] [0].id=VAR_ID_AMOUNT;
    myMatrix [2] [0].units=" l" ;

    myMatrix [3] [0].name=" Distance" ;
    myMatrix [3] [0].id=VAR_ID_DISTANCE;
    myMatrix [3] [0].units="km" ;

    myMatrix [4] [0].name=" Fuel_level" ;
    myMatrix [4] [0].id=VAR_ID_FUEL_LEVEL;
    myMatrix [4] [0].units=NULL;

    myMatrix [5] [0].name=" Spraying_time" ;
    myMatrix [5] [0].id=VAR_ID_SPRAYING_TIME;
    myMatrix [5] [0].units=NULL;

    myMatrix [6] [0].name=" Tank_content" ;
    myMatrix [6] [0].id=VAR_ID_TANK_L;
    myMatrix [6] [0].units=" l" ;

    myMatrix [7] [0].name=" Application_rate" ;

```

```

myMatrix [ 7 ] [ 0 ]. id=VAR_ID_DOS_L_HA;
myMatrix [ 7 ] [ 0 ]. units=" l/Ha";

myMatrix [ 8 ] [ 0 ]. name=" Current _output" ;
myMatrix [ 8 ] [ 0 ]. id=VAR_ID_L_MIN;
myMatrix [ 8 ] [ 0 ]. units=" l/min";

myMatrix [ 9 ] [ 0 ]. name=" Spray _pump" ;
myMatrix [ 9 ] [ 0 ]. id=VAR_ID_PUMP_RPM;
myMatrix [ 9 ] [ 0 ]. units="RPM";

myMatrix [ 10 ] [ 0 ]. name=" Spray _line _pressure" ;
myMatrix [ 10 ] [ 0 ]. id=VAR_ID_PRESSURE;
myMatrix [ 10 ] [ 0 ]. units="BAR";

myMatrix [ 11 ] [ 0 ]. name=" Total _area" ;
myMatrix [ 11 ] [ 0 ]. id=VAR_ID_HA;
myMatrix [ 11 ] [ 0 ]. units="Ha";

/* Variables whose frequency of change is low */
myMatrix [ 12 ] [ 0 ]. name=" Battery _voltage" ;
myMatrix [ 12 ] [ 0 ]. id=VAR_ID_BATT_VOLT_SYSTEM;
myMatrix [ 12 ] [ 0 ]. units=" volts";

myMatrix [ 13 ] [ 0 ]. name=" Engine _hours" ;
myMatrix [ 13 ] [ 0 ]. id=VAR_ID_ENGINE_RUNNING_HOURS;
myMatrix [ 13 ] [ 0 ]. units=" hours";

myMatrix [ 14 ] [ 0 ]. name=" Driving _time" ;
myMatrix [ 14 ] [ 0 ]. id=VAR_ID_DRIVING_TIME;
myMatrix [ 14 ] [ 0 ]. units=NULL;

myMatrix [ 15 ] [ 0 ]. name=" Time _to _service" ;
myMatrix [ 15 ] [ 0 ]. id=VAR_ID_ENGINE_SERVICE_INTERVAL_HOURS;
myMatrix [ 15 ] [ 0 ]. units=" hours";

}

/*****
*      Name: connection_ethernet
*      Goal: detect if there is ethernet connection available at a given moment.
*      Input: void
*      Output: 0 if there is no ethernet connection available.
*****/

int connection_ethernet(void){

    FILE *pcom=NULL;
    char comand[100];
    char aux[100];

    sprintf(comand, "/sbin/ifconfig _| _grep _Ethernet");

```

```

pcom=popen (comand, "r" );

if (pcom==NULL){

    printf("Error _when _detecting _Ethernet _connection\n");
}
else{

    fgets (aux,100 ,pcom);
    pclose (pcom);
    return( strlen (aux) -1);
}

}

/*****
*      Name: finish
*      Goal: close the current file after 15 minutes and compress it.
*      Input: void
*      Output: void
*****/

void finish (void){

    unsigned int m;
    int pid;
    char remove_command [35];

    pid=fork ();

    if (pid==0){

        fprintf (filew , "</log>");
        fclose (filew );
        if (execl ("/usr/bin/bzip2" , "bzip2" , filename ,NULL)==-1){
            printf ("Error _compressing _data .\n\n");
            exit (-1);
        }
        exit (1);
    }

    if (pid < 0){

        printf ("Error _while _trying _to _compress _data .\n\n");
        exit (-1);
    }

    for (m=0;m<1;m++){
        if (pid > 0){
            wait (0);
        }
    }

}

```

```

        fflush(stdout);

        if (connection_ethernet()!=0){
            system("expect_sftpdialogue.sh");
            sprintf(remove_command,"rm_%s.bz2",filename);
            system(remove_command);
        }

        timesno=0;
    }

/*****
 *   Name: writeXML
 *   Goal: write a ''variable'' in the XML log
 *   Input: index of the variable in the initialitation matrix
 *   Output: void
 *****/

void writeXML(int z){

    fprintf(filew,"<variable>\n");
    fprintf(filew," \t<date>%s</date>\n",datew);
    fprintf(filew," \t<name>%s</name>\n",myMatrix[z][0].name);
    sprintf(callgetValues,"./getValues_%d",myMatrix[z][0].id);
    fprintf(filew," \t<value>");
    valueReturned=system(callgetValues);
    fprintf(filew,"%d",valueReturned);
    fprintf(filew,"</value>\n");
    fprintf(filew," \t<units>%s</units>\n",myMatrix[z][0].units);
    fprintf(filew," \t<id>%d</id>\n",tractorid);
    fprintf(filew,"</variable>\n");
    fflush(filew);

}

/*****
 *   Name: writeLog
 *   Goal: write information about the variables each time that the SIGALRM is
 *         received
 *   Input: void
 *   Output: void
 *****/

void writeLog(int something){

    unsigned int k;
    unsigned int m;
    unsigned int day;
    unsigned int month;
    unsigned int year;
    unsigned int hour;
    unsigned int minutes;

```

```

unsigned int seconds;

timesno++;

if (timesno==1){

t=time(NULL);
today=localtime(&t);
day=today->tm_mday;
month=today->tm_mon+1;
year=today->tm_year+1900;
hour=today->tm_hour;
minutes=today->tm_min;
sprintf
(filename, "logging/%s-%02d-%02d-%02d-%02d", tractorname, day, month, year, minutes);

strcat(filename, ".xml");

filew=fopen(filename, "a+");

fprintf
(filew, "<!-- Logging file generated automatically by group_d606a_ (Aalborg University) -->\n");
fprintf
(filew, "<?xml version='1.0' encoding='UTF-8'?>\n");
fprintf
(filew, "<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>\n");
fprintf
(filew, "\t<xsd:element name='variable'>\n");
fprintf
(filew, "\t\t<xsd:complexType>\n");
fprintf
(filew, "\t\t\t<xsd:sequence>\n");
fprintf
(filew, "\t\t\t\t<xsd:element name='date' type='xsd:string'/>\n");
fprintf
(filew, "\t\t\t\t\t<xsd:element name='name' type='xsd:string'/>\n");
fprintf
(filew, "\t\t\t\t\t\t<xsd:element name='value' type='xsd:double'/>\n");
fprintf
(filew, "\t\t\t\t\t\t\t<xsd:element name='units' type='xsd:string'/>\n");
fprintf
(filew, "\t\t\t\t\t\t\t\t<xsd:element name='id' type='xsd:string'/>\n");
fprintf(filew, "\t\t\t\t</xsd:sequence>\n");
fprintf(filew, "\t\t</xsd:complexType>\n");
fprintf(filew, "\t</xsd:element>\n");
fprintf(filew, "</xsd:schema>\n");

fprintf(filew, "<log>\n");
else{

if (timesno==180){
//15 minutes
for (m=VARIABLES_O;m<NUM.VARIABLES;m++){

```

```

    if (myMatrix[m][1].id==TRUE){

        t=time(NULL);
        today=localtime(&t);

        hour=today->tm_hour;
        minutes=today->tm_min;
        seconds=today->tm_sec;

        sprintf(datew,"%02d:%02d:%02d",hour,minutes,seconds);
        writeXML(m);
    }
}

finish();
}
else{
for (k=0;k<VARIABLES_O;k++){

    if (myMatrix[k][1].id==TRUE){

        t=time(NULL);
        today=localtime(&t);

        hour=today->tm_hour;
        minutes=today->tm_min;
        seconds=today->tm_sec;

        sprintf(datew,"%02d:%02d:%02d",hour,minutes,seconds);
        writeXML(k);
    }
}
}
}

/*****
*      Name: routine
*      Goal: establish the time between two signal SIGALRM
*      Input: void
*      Output: void
*****/

void routine(){

    struct itimerval val;
    long seconds=5, microseconds=0;

    val.it_interval.tv_sec=seconds;
    val.it_interval.tv_usec=microseconds;

```

```

    val.it_value.tv_sec=seconds;
    val.it_value.tv_usec=microseconds;
    setitimer(ITIMER_REAL, &val, NULL);

    signal(SIGALRM, writeLog);

    while(1)
    {
        pause();
        //printf("Signal received\n");
    }
}

/*****
 *   Name: programEnd
 *   Goal: close all the files that could be open at the end of the execution of the
 *         program, even if this exit is unexpected
 *   Input: void
 *   Output: void
 *****/

void programEnd(void){

    fcloseall();

}

/*****
 *   Name: main (main function)
 *   Goal: writing a log about the working of a tractor
 *   Input: one parameter —> the variable whose value is wished to obtain
 *   Output: value of the variable passed as argument is displayed on standard output
 *****/

int main(int argc, char *argv[]){

    initialization();

    unsigned int i=1;
    unsigned int j=0;

    while (i<argc){

        for (j=0;j<NUM.VARIABLES;j++){
            if (atoi(argv[i])==myMatrix[j][0].id){
                myMatrix[j][1].id=TRUE;
            }
        }

        i++;
    }
}

```

```
    }  
    routine ();  
    atexit (programEnd);  
    return (0);  
}
```


B.1.12 sudoers

```
# sudoers file .
#
# This file MUST be edited with the 'visudo' command as root.
#
# See the sudoers man page for the details on how to write a sudoers file .
#

# Host alias specification
apache ALL = (root) NOPASSWD: /var/www/lykketronic/GPRS/getValues

# User alias specification

# Cmnd alias specification

# Defaults specification

# User privilege specification
root    ALL=(ALL) ALL

# Uncomment to allow people in group wheel to run all commands
# %wheel    ALL=(ALL)    ALL

# Same thing without a password
# %wheel    ALL=(ALL)    NOPASSWD: ALL

# Samples
# %users    ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
# %users    localhost=/sbin/shutdown -h now
```

B.2 Programs and configuration files located on the ACS

B.2.1 porttesting.py

```

#!/usr/bin/env python
# -*- coding: utf8 -*-
#librairies MySQLdb to use mysql databases with python, httplib
#for using GET requests to test webpages^M
#the goal is to test the connection of the connected black boxes
import MySQLdb, httplib, commands

proxyIP = 'wwwproxy'
proxyPort = '3128'
useProxy = 'False'
websiteURL = '130.225.192.134'

DBhost='130.225.192.134'
DBuser='justinbridoux'
DBpasswd='saucisson'
DBname='d606a'

fichier="infob.log"

def test_connection(proxyIP, proxyPort, useProxy, websiteURL, websitePort):
#def test_connection(ipproxy, port, useProxy, websiteURL, port)
    conn = httplib.HTTPConnection("%s:%s"%(websiteURL, websitePort))
    try:
        conn.request('GET', '/')
        resp = conn.getresponse()
        if resp.reason == 'OK':
            connection_status = True
        else:
            connection_status = False
    except:
        connection_status = False
    conn.close()
    return connection_status

# connection and selection of the tractors having a port number different from 0
# meaning that they are connected
def db_update(DBhost, DBuser, DBpasswd, DBname):
    global fic
    connectionObject = MySQLdb.connect
        (host=DBhost, user=DBuser, passwd=DBpasswd, db=DBname)
    c = connectionObject.cursor()
    fic.write('NNN_host=%s, _user=%s, _passwd=%s, _db=%s \nNNN\n'%(DBhost, DBuser,
    DBpasswd, DBname))
    c.execute("""SELECT * FROM 'd606a'.tractor WHERE port!=0 """)
    #for each tractor selected we are testing by using the tunneled port
    #of the proxy
    for row in c.fetchall():

```

B.2. PROGRAMS AND CONFIGURATION FILES LOCATED ON THE ACS125

```
reply = test_connection(proxyIP, proxyPort, useProxy, websiteURL,
    '%s'%row[4])
print " tractor_:"
print row[1]
print " _port_="
print row[4]
if reply == True:
    print "Connection_ available"
else:
    print "Connection_unavailable:_Problem"
    fic.write('Connection_unavailable:_Problem_on_%s:_port_:
    %s_$$\n'%(row[1], row[4]))
    connectionObject.commit()
    connectionObject.close()
cmd="ls_|_grep_'infob.log'"
status, output = commands.getstatusoutput(cmd)
fic = open(fichier, 'w')
fic.write('Test')
db_update(DBhost, DBuser, DBpasswd, DBname)
fic.close()
```

B.2.2 storingMySQL.c

```

/*****
 *      Title: storingMySQL.c
 *      Description: program that will be running in background in the proxy.
 *                  It will extract the data that has been sent by each of the
 *                  tractors in XML format and store it in the MySQL database.
 *      Date: 28th April 2008
 *      Author: group d606a – Software Systems Engineering 4
 *              (Aalborg University)
 *****/

#include <stdio.h>
#include <dirent.h>
#include<stdlib.h>
#include<signal.h>
#include<string.h>
#include <unistd.h>
#include<sys/time.h>

/*****
 *      Name: storeMySQL
 *      Goal: store the informacion logged by the tractors into the database
 *            in the ACS
 *      Input: void
 *      Output: void
 *****/

void storeMySQL(int algo){

    int n, i;
    struct dirent **namelist;
    char comand[150];
    char url[200];

    //List of files (n is the number of files in the directory)
    n=scandir("../lykketronic-logs",&namelist ,NULL,NULL);

    i=0;
    while(i<n)

void storeMySQL(int algo){

    int n, i;
    struct dirent **namelist;
    char comand[150];
    char url[200];

    //List of files (n is the number of files in the directory)
    n=scandir("../lykketronic-logs",&namelist ,NULL,NULL);

    i=0;
    while(i<n)
    {

```

B.2. PROGRAMS AND CONFIGURATION FILES LOCATED ON THE ACS127

```
//If it is not the current directory or the superior
if ((strcmp(namelist[i]->d_name,"..")!=0)&&(strcmp(namelist[i]->d_name,".")!=0)){
//Copy the logs from the infolder to another one
sprintf
(comand,"cp_${HOME}/lykketronic-logs/%s_${HOME}/LOGS",namelist[i]->d_name);
system(comand);
//Decompress the logs that remain in the infolder
sprintf
(comand,"/bin/bunzip2_${HOME}/lykketronic-logs/%s",namelist[i]->d_name);
system(comand);
//Save their contents on the database
sprintf(url,
"elinks_http://130.225.192.134:56789/xml2mysql.php?xmlfilename=
-----/home/justinbridoux/lykketronic-logs/%s"
,namelist[i]->d_name);
system(url);
}
i++;

}

//Release memory
i=0;
while(i<n)
{
    free(namelist[i]);
    i++;
}

//All decompressed files are together in the same directory.
//Their content has been stored on the database.
//The compressed files are stored in another directory.
//Conclusion: we can remove the files in this folder
sprintf(comand,"rm_${HOME}/lykketronic-logs/*");
system(comand);

}

/*****
*      Name: programEnd
*      Goal: close all the files that could be open at the end of the execution of the
*            program, even if this exit is unexpected
*      Input: void
*      Output: void
*****/

void programEnd(void){

    fcloseall();

}

/*****
```

```

*      Name: routine
*      Goal: establish the time between two signal SIGALRM
*      Input: void
*      Output: void
*****

void routine(){

    struct itimerval val;
    //Time between to consecutive SIGALRM = 15 minutes
    long segundos=30, microsegundos=0;

    val.it_interval.tv_sec=segundos;
    val.it_interval.tv_usec=microsegundos;
    val.it_value.tv_sec=segundos;
    val.it_value.tv_usec=microsegundos;
    setitimer(ITIMER_REAL, &val, NULL);

    signal(SIGALRM, storeMySQL);

    while(1)
    {
        pause();
        //printf("Signal received\n");
    }
}

/*****
*      Name: main (main function)
*      Goal: insert into a database the information stored in the logs that the
*            tractors periodically send to the proxy
*      Input: void
*      Output: the information in the logs has been stored into the database in the
*             Access Control Server
*****/

int main(int argc, char** argv)
{
    routine();

    atexit(programEnd);

    return 0;
}

```

B.2. PROGRAMS AND CONFIGURATION FILES LOCATED ON THE ACS129

B.2.3 xml2mysql.php

```
<?php
// Group d606a - Software Systems Engineering (Aalborg University)
//
// xml2mysql
// +----- recordSet      // Instance of recordSet
// +----- xml            // Instance of XMLFile
// +----- xml2mysql()    // Initialize the instances of recordset and XMLFile
// +----- insertIntoMySQL( Name file , Name table) {
//
//
//
require("class.recordset.phtml");
require("class.xml.phtml");

class xml2mysql {

    var $recordSet;
    var $xml;

    #Initialize by creating the members
    function xml2mysql() {

        $this->recordSet = new recordSet();
        $this->xml = new XMLFile();

    }

    # Insert into the database's table
    function insertIntoMySQL($filename, $tablename) {
        $xml_file = fopen($filename, "r");
        $this->xml->read_file_handle($xml_file);

        $numRows = $this->xml->roottag->num_subtags();

        for ($i = 0; $i < $numRows; $i++) {
            $arrFields = null;
            $arrValues = null;

            $row = $this->xml->roottag->tags[$i];
            $numFields = $row->num_subtags();

            for ($ii = 0; $ii < $numFields; $ii++) {
                $field = $row->tags[$ii];
                $arrFields[] = $field->name;
                $arrValues[] = "\"\".\"$field->cdata.\"\"";
            }

            $fields = join(" ,_", $arrFields);
            $values = join(" ,_", $arrValues);

            $this->recordSet->exec("Insert_Into_.$tablename_($fields)_Values_($values)");
        }
    }
}
```

```
    }  
  }  
}  
  
$file_name=$_GET[ 'xmlfilename' ];  
$importation=new xml2mysql();  
$importation->insertIntoMySQL( $file_name , "variable" );  
  
?>
```


B.2. PROGRAMS AND CONFIGURATION FILES LOCATED ON THE ACS131

B.2.4 login.php (obfuscated version)

This is an example of obfuscation of our code that we used with the aim of hidden the salt that is used together with the password hashing. The original code of this file, and of the rest that have been also obfuscated can be found in the CD - ROM attached to this report.

```
<? eval(gzinflate(base64_decode('
3Vl9b9s2Gv/b+hSsLpjsQ2Kl6a64xZKCrLW3AeKy
OO4NQxYYskRHWiXRpag6WZvvg8fUrQU26l964Dt
DCSWyOf9hfyRPGu8s0WysEpalikrpqUIuej2B1Za
RFkVU+JELM9Z0QciZ2BZ6bx7MP1uOLl2BA8jwXiU
sDSizs0z37Z71kerc/C+ovye+MS+G14MX01Iulhw
dnd/SBaMCzIaX74hmpf8/P1wPCRpDNSOTfks2iY
sB17AKI5LatMAHV+X77PpqipqxQeHsSzacSKgt6B
Hz1NbUjnVETJILNlVwtBijCOOZDIoetj1HOKZsj3
5zcE/O247vA3GIURCCUQjqogEJCwiBm8ZaRMS0Hz
sCPp3oo0S3/HKV6JsCThrGQZPmUhoUYKWJXDGA8X
aRxaHXpHo65DwVlShkBRlkDziZQVYHETIIP3t6oU
aTHjacyqQ3eecroMs8ydL4/giy37ZUIcDN7VcPyf
4fh6PHxzORlOz1+/HqNPpOGTdDuhYUx5175gUSjA
olOSCLE4dV2kkzEBqgerLoGu3XcViywCW9bGnHSf
gaFUdKXSq6sfLn+8djJ2mxbOTU9VwXyK7/7B9KfL
K8ipfnduBji5CMty0ZiU7805385PxPHxv+1+Hv+r
WzP0+jZo/Rpr4XGZ/ZPMOctJVVJlgnlFB+LMKe
...
/rfVOX4LWvv06SnY1Nt0z6N26/o4JdGMra4RnsA5
bTUGWa6p0Gv3qK+UjPpNNWSkF2kcBkBYH+0m8D5F
CqStb50mmnzyiHwsyWsi5OA0CwWNP8tUJGSs2ZAM
j5BP7QU9PDE+7IIAtu3tCJxkAvWeLqFcSepdXeay
sUU8cemFvUE0pPBkjjvDJQ7jZswmNIHjPqqzNEQ
EhCFg36etc7qs4xFJ3imJ/Ljyf2UhHi69+2+i5FC
QElyKhIGHN8NJ+oOYPMVwSulAuBddNfTCVNIWNqs
rgptclm8SsLiFkbLapan8iZTC2UL3M0/hFkFs3bw
KmGspBhI0u9DayqCQEPTP9IaRH/+7A6x/i9aRAZg
vyZpw9t2K0A5/k/N0C6Px6Xv4NZT33yvCk4mor5Y
NbsAUdkeviZ4OnPdR5k1rBvEDWqD1ppNl6dTX7LJ
A5dKimoJ+adOYN4Z2WwrXtF/JGe7dFmrw1r38aqf
TEMZZx93V101X7i51iu+7ir9A0Orjg14l3HwGz8x
NONC+vZg1Yhbum1fzZPDza3xWVuggOQl9aqFtncQ
EmKL+q3OtAdkgzfnVcV482eE8W/SzHLo5EZer5hr
+lLcZxLB2zPohVvOqiI+iljG+Ok/RqOXL1+8GCC3
RH278Lx8ORp9843m2b+t6wvwjgoFPkXYoIGXFotK
EHG/MFudTaIMfzipX7X7V7SIoaE1I17UyM24hQXl
mvHQ/Llnzpgwv/fIFePxxzv7Lw==
'))); ?>
```