A reliable timing system using RFID

Mälardalens Högskola

Academy of Innovation, Design and Engineering

Jonas Telander Håkansson
Lennart Eriksson

Thesis for the degree of Bachelor in Computer Science

2015-05-27

Examinator: Mats Björkman

Supervisor: Martin Ekström

Sammanfattning

Tidtagningssystem för tävlingar utomhus har nyligen börjat utnyttja Radio Frequency ID (RFID)-teknik, för att förenkla och noggrannare kunna registrera alla deltagares tider. Det finns ett antal sådana applikationer på marknaden redan idag, men ingen av dem har tagit hänsyn till problemet med otillförlitlig kommunikation. Under en tävling används många RFID-läsare samtidigt som avläsare vid olika stationer. Eftersom samtliga oftast är kopplade över ett nätverk till en och samma dator som hanterar alla registreringar, vad händer om en kabel går av, eller om datorn kraschar? Den här rapporten beskriver examensarbetet som utfördes framförallt för att åtgärda det problemet. Arbetet gick ut på att förbättra och utöka ett existerande tidtagningssystem, genom att lägga till ny funktionalitet och utöka med den tillförlitlighet som krävs. En state-of-the-art-analys gjordes på relaterade arbeten, med fokus på mellanlagring och tidssynkronisering, för att hitta ett lämpligt sätt att säkerställa tillförlitlig kommunikation. Mjuk- och hårdvara som kan användas till arbetet har undersökts, och problemen som behövde lösas delades upp i 4 mindre frågor, som alla besvarades, och presenteras i den här rapporten. Resultatet är ett system där en applikation, som är implementerad i en Raspberry Pi mikrodator, hjälper den befintliga mjukvaran genom att säkerställa att alla registreringar under en tävling skickas till huvuddatorn och därigenom sparas till databasen, även om kommunikationen tillfälligt bryts.

Abstract

Timing systems for outdoor competitions recently started to use Radio Frequency ID (RFID) technology, to simplify and make precise time registrations of the participants. A couple of application for this usage exists on the market as of today, but none of them has addressed the issue of communication reliability. With a lot of RFID readers placed in different location during a competition, acting as checkpoints and connected to one network and having one computer managing all the registrations, what happens if a cable breaks or a computer crashes? This report describes the thesis work that was conducted mainly in order to address that issue. The goal of the thesis work has been to improve and extend an existing software used for timekeeping by adding new functionality and extending it with the reliability that was requested. A state-of-the-art analyze was made among related work, with focus on intermediate storage and time synchronization, in order to find the most suitable way of securing reliable communication. Software and hardware related to the work was researched, and the problems that needed to be addressed were divided into four research question which were all answered, and presented in this report. The result is a system where an application, implemented in a Raspberry Pi single board computer, helps the existing software to ensure that all the registrations during a competition are at some point sent to the main computer and subsequently stored in the competition database, even if the communication is temporarily broken.

Acronyms

API	Application Programming Interface
LAN	Local Area Network
LLRP	Low Level Reader Protocol
LTK	LLRP Toolkit
NTP	Network Time Protocol
PTP	Precision Time Protocol
RFID	Radio Frequency ID
SBC	Single Board Computer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UHF	Ultra High Frequency

Table of Contents

Table of Figures and Tables

1. Introduction

Radio Frequency ID (RFID) is a technology that enables automatic identification using radio waves, without the need of physical contact with the objects [1]. It is being used in a wide range of fields, including electronic toll system [2], tracking [3], bookshelf management [4], and so on. One area where it has recently become more popular is within timing systems in sports competitions. This thesis work aims to improve such a system, for a local company involved in this business, with an important set of functionality that so far is missing.

In this chapter, the background for this thesis work and why it is being made is presented. The problem formulation is also included, which specifies more closely which problems need to be solved.

1.1 Thesis Background

This thesis is carried out for a company called Västerås Tidtagning [5]. They are, for now, a non-profit organization working with timekeeping at various competitions. Their main area of expertise as of today is swimming competitions indoors, but they also carry out timekeeping for outdoor running and swimming competitions. In these events, timekeeping is carried out using RFID tags, attached to each participant. The tag is read by RFID readers placed in specific locations, to get the correct time at checkpoints and/or finish line. Each reader is connected to the same main computer, where all registered readings are sent and stored for results.

Their prominent work so far has opened the eyes of the Swedish swimming federation who wants them to co-host larger competitions, and they have been asked to carry out triathlons as well. In order to handle these kind of events, however, they need to upgrade their equipment, and their software in particular.

Today they are using a system from Agee race timing [6]. This is a very basic system and does not have all the functionality required for the competitions Västerås Tidtagning have been offered to host. Therefore it is not an option for the future. The basics of a new software exists, designed with the goal of replacing Agee race timing. However, it still lacks crucial functionality in order to be fully approved. The main issue that Västerås Tidtagning need to solve is the reliability of the system. This thesis project aims to investigate whether, and in that case how, it is possible to add the required and desired functionality to the existing software, according to the company's vision.

1.2 Problem Formulation

Before the new software can be acceptable, a set of problems need to be addressed. The main issue is that since outdoor competitions are carried out in harsh environments, communication reliability is imperative. No registrations are allowed to get lost since that could in worst case nullify the whole competition. Besides that, some pure functionality has to be added before the Swedish swimming federation can approve it as a time keeping software.

The problems can be divided into four subcategories, of which one (the first) is the most important and main problem. These four subcategories are listed below as questions 1-4.

Q1: How can a reliable communication between each reader and the main computer best be guaranteed?

In today's system, if a network connection fails, an Ethernet cable breaks, or if the computer software simply crashes, all registered readings during the down time are lost. This is of course not acceptable, and this is the primary problem that has to be solved. If a connection is lost to the main computer, all registered readings still has to be saved for later use. This is a functionality which no current system available has, and Västerås Tidtagning feels this is important in order to provide an as reliable timing system as possible. Their vision is to use some kind of intermediate storage connected to the reader, to act as a middleware between the reader and the main computer, and this will be examined in order to see if it is applicable.

Q2: How is registration of each participant at the starting line best carried out?
A demand from the Swedish swimming federation is that each participant is electronically registered at the starting line. This is a safety measure, mostly used for swimming competitions in open water, where the organizers need to know exactly how many participants are in the water at any given time.

Q3: How should the information for the commentators be updated?
The current system doesn't have any function for sending any kind of information to commentators. So far, the commentators have used the raw data presented on the main computer, or simply not used the time data at all. For bigger competitions, commentators are more important, and Västerås Tidtagning wishes to be able to present all necessary data for them in a suitable way.

Q4: How are the new additions best implemented in the current system?
Västerås Tidtagning has tried to find a complete system that has all these required and desired functionality, but hasn't succeeded. And since there is not enough time for this thesis to implement a completely new system, all new additions must be able to seamlessly fit into the current basic software shell that is available.

2. Research Method

For this thesis, the research is divided into two different methods. The first is through interviews. Since the area of this thesis is somewhat narrow, and not common knowledge, quantitative research methods, i.e. questionnaires, are not a good way to retrieve reliable information. Instead, interviews with key persons with specific knowledge connected to the thesis work, and with insight into the area, can be conducted. The interviews can perhaps more be seen as a consultation, since most of the answers will come from the company itself. The main research method will be a literature study, to determine the state-of-the-art. Related works need to be checked, to determine a suitable solution of the problems, and due to the time constraint of the thesis, and the nature of the problems, the literature study is probably the most suitable and effective method to use. Since the company already have compared systems that could cover the requirements, but not found anyone suitable, the literature study should be concentrated on the actual problems and how to solve them.

Q1 is the main problem to solve, and also the most important one. Therefore, this will be the main area for the literature study, both on what can be done and how to actually solve it. The main areas of research will be intermediate storage and time synchronization, since every reader must have the exact same time, down to one tenth of a second.

For Q2-3, interviews or consultations can be conducted. In Q2, the company is the preferred party to consult, while for Q3 it could be suitable to ask questions to a speaker with experience from these kind of events.

For Q4, the research method will be to simply test the created software with the current system, to check that it works.

3. Background

This chapter will discuss the state-of-the-art and related work on this subject, as well as sections covering both the hardware and software connected to this thesis. The state-of-the-art analyze focuses mainly on the areas related to the thesis work that can be used to answer the research questions.

3.1 State-of-the-art

This section discusses the state-of-the-art analyze and related work of this thesis. It is divided into subcategories, covering the main areas that needs to be analyzed in order to answer the research questions. These areas include available RFID chip timing systems, the usage of intermediate storage, and time synchronization.

3.1.1 RFID chip timing systems

Some RFID based timing systems already exists. Agee Race Timing [6] is the system currently being used by Västerås Tidtagning. This software supports running, bicycle and swimming competitions, but since it only supports one reader per race, it is not a realistic option for the company. Another system, called RFID Race Timing Systems [7] is a more complete system, which software can handle several readers. But besides its high price, which is a large factor for Västerås Tidtagning, it suffers from the same issues as all other current RFID timing systems, namely reliability. No system available can handle a loss of communication between the reader and main computer without losing all new registrations during the down time. Since this is what Västerås Tidtagning wants, it will have to be created.

3.1.2 Intermediate storage

While there basically doesn't exist any work where timekeeping in sports is carried out using some form of intermediate storage, other related work do exist. M Adi M Sarbini *et al.* presents in their paper [8] that use of RFID readers with intermediate storage could help monitoring the speed of the traffic in and around Brunei. They present a design of an Ultra-High Frequency (UHF) RFID system that monitors the speed of vehicles in Brunei, where the number of injuries caused by speeding traffic is very high. In the paper they propose the use of a Raspberry Pi connected to an RFID reader, where all information would be computed and stored before being transferred via a secure 3G connection to a remote administrations server.
In another report, K. A. Salunkhe *et al.* [9] describes the use of computers as intermediate storage that is capturing measurements being sent from a Frequency Measurement Device, putting a time stamp on them, and then forwarding the data to a central server. This setup is used in a Wide-Area Measurement System in India, where they have a server located in Bombay and 6 client stations spread out on different locations over the country. While these computers are more of a stationary type, they still show the practical use of intermediate storage and forwarding.
R. Danymol *et al.* [10] describes a way of receiving FM signals, and then decoding and forwarding them to another computer in real time via LAN. The real time forwarding task is carried out by a Raspberry Pi and although it is not powerful enough to actually decode the signals, it serves perfectly as a way of forwarding the signals to a remote computer. The authors even describes the potential uses of the Raspberry Pi as "astonishing". Another related work, using a Raspberry Pi connected to an RFID reader, was the work of Michael Rushanan at the University of Michigan [11], where an RFID client, implemented in a Raspberry Pi, was needed to be submerged in concrete inside a cinder block. As stated on his blog, the Raspberry Pi was used because "This client should be accessible, reliable,

and deployable. . . . Also, the client should be energy efficient if it is to operate over large time deltas.” These features are all important also in this thesis, and gives a good idea what can be accomplished.

3.1.3 Time synchronization

The two main time synchronization protocols available are Network Time Protocol (NTP) and Precision Time Protocol (PTP). As stated by R. Ratzel and R. Greenstreet in their article [12], NTP has for many years been the obvious choice for time synchronization. However, as they point out, the newer PTP has higher synchronization performance, being able to synchronize clocks to “within tens of nanoseconds of each other”, in perfect conditions. This vastly outperforms NTP’s threshold of milliseconds. PTP is also designed to be used over Local Area Network (LAN), whereas NTP is typically used over the internet. However, in order to achieve maximum performance, PTP is dependent on boundary clocks, transparent switches and other hardware support, and as A. N Novick *et al.* [13] pointed out, in some environments PTP hardly outperforms NTP at all. NTP on the other hand, can maintain its millisecond precision without any specialized hardware.
Earlier referenced article by K. A. Salunkhe *et al.* [9] also describes how the measurements are time stamped before being sent to the central server. Six different clients needs a good time synchronizer, and even though they state NTP is not as accurate as a GPS scheme, they believe it is “inexpensive and elegant”, and a good choice if the precision of NTP is enough.

3.2 Software

In this thesis work some different standards and protocols have been researched and used. These protocols are either required to perform some tasks, such as time synchronization, or helps other tasks, such as communicating with the RFID readers. This section covers the most important ones used in the thesis work, as well as a short summary of the existing software this thesis work aims to complete.

3.2.1 Low Level Reader Protocol

The Low Level Reader Protocol (LLRP) is an interface protocol ratified by EPCglobal [14] in 2007. It is designed to standardize the programmatic interface between RFID readers from different manufacturers and clients. This protocol makes it easier for developers to construct RFID middleware, in order to create unique software communicating with RFID readers that support LLRP.

3.2.2 Transmission Control Protocol

The Transmission Control Protocol (TCP) is the network transport protocol used in this thesis work. It is a core protocol of the Internet Protocol Suite and provides reliable, ordered and error-checked delivery of packets between applications, running on hosts, communicating over IP network. It is used both as a protocol between the Raspberry Pi and the main computer, as well as between the Raspberry Pi and the RFID reader, since LLRP is based on TCP.

3.2.3 Network Time Protocol

To make sure the time on each station is synced with each other and the main computer, NTP has been utilized. NTP is an internet protocol dating back to 1980’s, and the NTP implementation documentation [15] states that it has since then been “widely used to synchronize a computer to Internet time servers or other sources, such as a radio or satellite receiver or telephone modem service.” It can use multiple redundant servers in order to achieve an accuracy of less than a millisecond on LANs, according to the documentation.

3.2.4 TimeChip, the existing software

Besides the timing system currently in use by Västerås Tidtagning, they also have the shell of a new system. This system contains all the basic functionality, such as setting up competitions and delivering results. It also has the functionality of handling several readers simultaneously, as well as readers of different models. But as stated earlier, it lacks a lot of required functionality, and this thesis work aims to complete this system, in order to make it approvable for bigger events.

3.3 Hardware

A variety of hardware has been necessary for the work of this thesis. While no hardware has been built, the understanding of how the used hardware works is very important, as the software will need to be created in a way that works flawlessly with them. These hardware include RFID readers, single board computers (SBCs) and antennas. Figure 3.1 shows how all the hardware parts is supposed to be connected with each other, and this section then covers some of the parts a little closer.

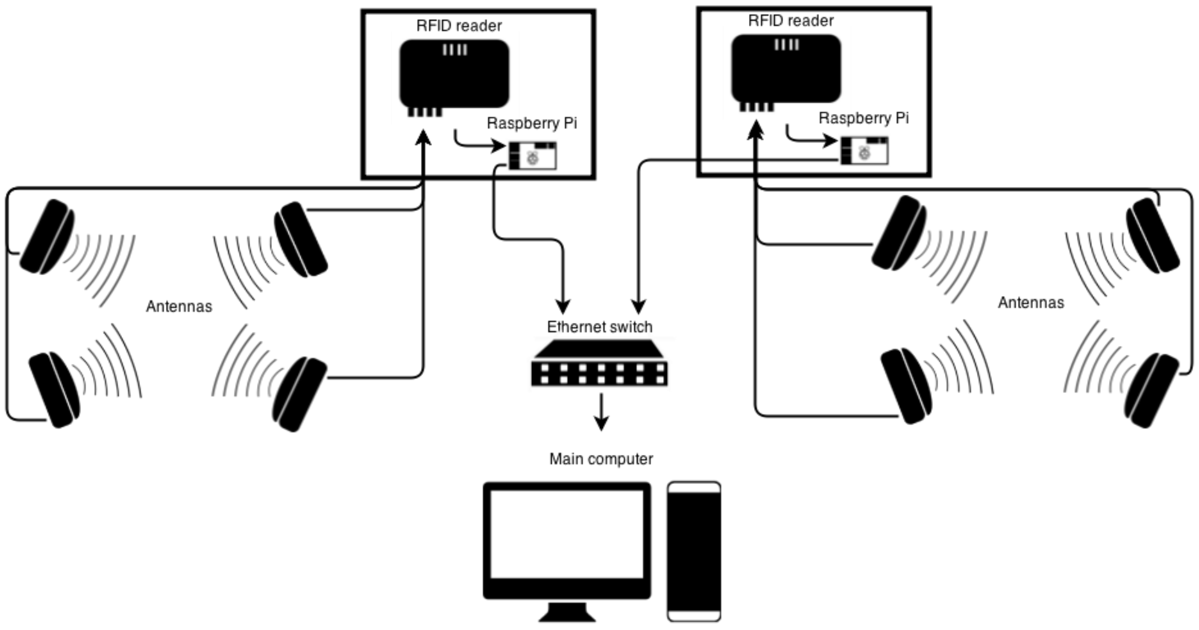


Fig. 3.1 – All connected hardware parts for the timing system. Each pair of RFID reader and Raspberry Pi are encapsulated together within a box, to keep them protected from weather, etc. Antenna cables are then connected between each RFID reader and its group of antennas, and an Ethernet cable is connected between each Raspberry Pi and the main computer, via a switch.

3.3.1 Antennas

Västerås Tidtagning uses a wide band antenna from Motorola, called AN480-CL66100WR. It is a single port antenna that is made to handle different environments, including outdoors, and works perfectly with Motorola's fixed RFID readers, but also with readers of other brands.

3.3.2 RFID readers

For readers, Västerås Tidtagning currently uses two different models. Both are from Motorola, called FX7400 and FX7500 respectively. Both have 4 antenna ports, as well as

an Ethernet port and a USB port. Both are high performance readers, but the newer FX7500 is built in a different way, with better hardware components and Linux operating system, compared to Windows MC for the FX7400. According to an article at RFID Journal [16], these differences gives the FX7500 faster read and write speeds, and increased read sensitivity.
The most important feature of these models, however, is that they both support LLRP. Part of this thesis work is to make sure the software is compatible with as many different RFID readers as possible, in case Västerås Tidtagning would like to use other models in the future. While it is not possible within the scope of this thesis work to actually test a wide variety of readers, with LLRP being one of the most used open source standards for RFID readers, it is imperative to design software compatible with it. And for that these two readers is a perfect choice.

3.3.3 Single Board Computers

For storing the registrations in case of a communication loss, microcomputers, or single board computers has been researched. The suggestion from Västerås Tidtagning was to use a Raspberry Pi, which probably is the most popular SBC [17]. While this was the original plan, a brief comparison between different models still had to be made, shown in section 4.1.

4. Technical Description

During the work with this thesis, some decisions regarding the technical parts and the design of the implementation has been made. This chapter describes just that; the process of designing and implementing the required new functionality and reliability for the timing system. It is divided into sections covering the different parts of the work and includes different choices made along the process.

4.1 Choosing hardware for intermediate storage

While the decision of using an SBC for intermediate storage was ratified as a suitable method, it still had to be decided which model to use.
After some research, Raspberry Pi 2 Model B and BeagleBone Black turned out to be the most suitable contenders. These are two of the most popular models available, and are both pretty similar hardware wise. A comparison of hardware and key features was made between Raspberry Pi and BeagleBone, shown in table 4.1. Together with a benchmark performed by Adafruit [18], it shows that the Raspberry Pi has better overall performance, although they are both very close. The BeagleBone however, has the advantages of having 4GB of internal storage space included, and comes with an operating system pre-installed. This was discussed with the company, but since the operating system takes up a bit of the storage space, it was decided that a total of 4GB was too little anyway, and extra storage space (i.e. a micro-SD-card) still needed to be added. Therefore, the choice was to go with the Raspberry Pi.

Model:	**Raspberry Pi 2 Model B**	**BeagleBone Black V3**
Price:	~400SEK	~650SEK
Processor:	Quad-core ARM Cortex A7 0.9 Ghz	Single-core ARM Cortex A8 1Ghz
Memory:	1GB	512MB DDR3
Storage:	- (Micro-SD)	4GB flash storage
USB:	4	2
Network:	LAN	LAN
OS:	No pre-installed (Win10-ready)	Linux pre-installed
Power:	5V, 1A	5V 3A
Measurements (BxH):	85x56 mm	86x53 mm

Table 4.1 – Comparison between Raspberry Pi 2 Model B and BeagleBone Black V3

4.2 Constructing RFID middleware for Raspberry Pi

Most RFID readers, including the FX-series from Motorola used by Västerås Tidtagning today, come with software Application Programming Interface (API), to ease the construction of own applications able to communicate with the reader. However, this thesis work aims to create software that can work with RFID readers from different brands, and therefore a more generic way needs to be found. The obvious standard to use is LLRP, but a few decisions still had to be made.

4.2.1 Choosing medium for communication with the RFID reader

With the use of Motorola FX7400 and FX7500 as readers there were a few possibilities to communicate; via Bluetooth, USB or Ethernet. Bluetooth was ruled out immediately because, first of all, it requires a separate hardware plugged into the Motorola, and secondly, its wireless nature might be too unreliable [19]. Besides, it would not be guaranteed to work with other readers than the Motorolas. USB would be the preferred choice, since there would be no need of extra hardware such as switches, to handle the Ethernet cables between all units. However, not all readers are sure to have USB-ports, making Ethernet the most logical choice considering the generic part of the program. Fortunately, according to Motorola's Integrator guide for the FX7500 [20], Motorola readers can handle communication over USB the same way as over Ethernet, just by pointing to the correct IP address. Because of this, the same code can be used both for Ethernet solutions as well as USB solution. While it is not certain all readers with USB support this, it is a very good solution for the company and this thesis work.

4.2.2 How to communicate with the RFID reader

Since LLRP is a low level protocol, the communication messages between the middleware and RFID reader has to be converted to and from a high level language. In this thesis work, the LLRP ToolKit (LTK) [21] has been used to make the transition smoother. The LTK is a toolkit developed to make it easier to create software that connects to RFID readers and it includes libraries to help with settings and communication to the reader. The LTK is not intended for Raspberry Pi and there was a few problems compiling the program containing it. But by writing the program on a computer running Ubuntu and using a cross compiler to build it, the program was able to run even on the Raspberry Pi. Though there had to be a few changes in the make files and installing the cross compiler "gcc-arm-linux-gnueabi", as posted by the earlier mentioned Michael Rushanan at the University of Michigan [22]. The default settings of the LTK gave more ineffective results compared to Motorola's own software in terms of read speed. But by tweaking some of the settings, the software actually improved to handle tags with 2 ms apart, compared to the Motorola version on Windows which registered tags with 6-7 ms apart.

4.2.3 How to communicate with the main computer

The main objective for the communication between the Raspberry Pi and the main computer is to send packets containing information over a wired network. It is more important that the packets arrive late than not at all. For this, TCP is the perfect choice. As C.R. Pakanati *et al.* states in their report comparing TCP, UDP and the newer TFRC protocol [23], "TCP is the transport protocol which is fine tuned to wired networks." and "TCP is suitable for applications which require reliability…". In order to use TCP for this thesis work, however, some options needed to be set. These were the SOCKET and TCP options "SO_SNDTIMEO", "SO_SNDBUF", and "TCP_USER_TIMEOUT".
The "SO_SNDBUF" option sets the size of the send buffer, which is a memory buffer to where all data is written before it is actually sent through TCP. During sending operations, it is not until this buffer is full that the socket can realize a connection has been lost, and therefore this is set to its minimum allowed value, which according to the Linux manual pages [24] is 2048. In order to specify how fast the socket should acknowledge the connection as lost after the buffer is full, the "SO_SNDTIMEO" option needs to be set. This option tells the socket how fast it should block a send operation if the send buffer is full, i.e. call a timeout, and is set to 2ms for this thesis work.
These two options are set to be able to detect a broken connection as fast as possible, but once a connection is broken, the TCP allows transmitted data to remain unacknowledged for approximately 15min, before it forcibly closes the connection. To avoid this happening, in case of a longer connection loss, this timeout, controlled by the "TCP_USER_TIMEOUT" option, is set to 4 hours.
Another issue is if a connection is broken, and a process is trying to send something to it anyway. If that happens, the process will receive a "SIGPIPE" signal, which will terminate it. Since this scenario very well can happen during a competition, the flag

“MSG_NOSIGNAL” had to be added to the send function, to be able to handle it without the process terminating [25].

4.2.4 How to save tag registrations in case of connection failure

The problem with being in a harsh environment is that almost nothing can be accounted for. Runners or bikers might run over a cable, causing it to break, and programs can crash unexpectedly. What should happen with the data if it does not get sent to the main computer? TCP ensures that the packets that is written to the buffer will be sent before it throws it away. However, in case of a connection loss TCP isn’t of much help. Therefore, in the case that the send buffer is filled and causes a timeout, the data that is read from the RFID reader should not be blocked or thrown away. Instead this data should be saved on the Raspberry Pi. Either by storing it in main memory of the process or write it to a file. Since the Motorola FX7500 have the capacity to fetch more than 1200 RFID tag readings per second [26] a broken connection even for a few seconds might lead to a lot of data to handle. Due to the arbitrary, and possibly large number of registrations that might occur during a connection loss, storing them in a file instead of in the main memory makes it easier to manage. When a connection breaks, the process creates a new thread that tries to reestablish it, while the main thread continues to fetch information from the RFID reader and saves it to a file. When the connection is reestablished, the main thread switches back to sending the registrations instead of saving them. The child thread then sends the saved information from the file, before exiting.

4.3 Time synchronization

The time synchronization is a very important part. With the RFID readers connected to different SBCs, it is imperative they have the exact same time, to avoid faulty results in the end. This thesis work made use of NTP, due to its reliability without extra hardware and the fact that it’s precision of milliseconds is more than enough for this work, where the required precision is a tenth of a second.
The operating system of the Raspberry Pi comes with an NTP daemon pre-installed. The NTP daemon is the part that synchronizes the system clock against different special servers through internet. To instead synchronize from another computer, a redirection of the daemon to point to a specific IP address is required.
While this is easily done, it requires the host computer to have an NTP server installed. This means an NTP server has to be installed on any computer Västerås Tidtagning uses as their main computer during a competition. For this thesis work, the NTP server from Meinberg software has been used [27]. It is also important that the main computer adds its local server (i.e. 127.0.0.1) as a valid time sync server, or the synchronization will fail if no internet connection is available, which sometimes is the case at races.
Another aspect is the drift time of the SBCs. Different computers might drift in time, depending on aspects such as placement, temperature, hardware, etc. This means that it might not be enough to sync the time only once before the races, but perhaps during them as well. The NTP daemon syncs continuously after being started, but it does this by changing the clock frequency over time to match the server’s. In other words, it might drift too much in the beginning of a race, before it can completely sync the frequency. Because of this, some tests were made to see if it would be necessary to sync the units more often than NTP does automatically at startup. The tests and their results are presented in Appendix A, and shows that there is almost no drift during five hours after startup. This means the built in synchronization of NTP on the Raspberry Pi is enough for this thesis work.

4.4 Registration at start line

In order to register the competitors before the competition a discussion was held with Västerås Tidtagning. It was decided to use an extra station and put it as a gate where all the participants have to pass before entering the starting line, shown in figure 4.1.

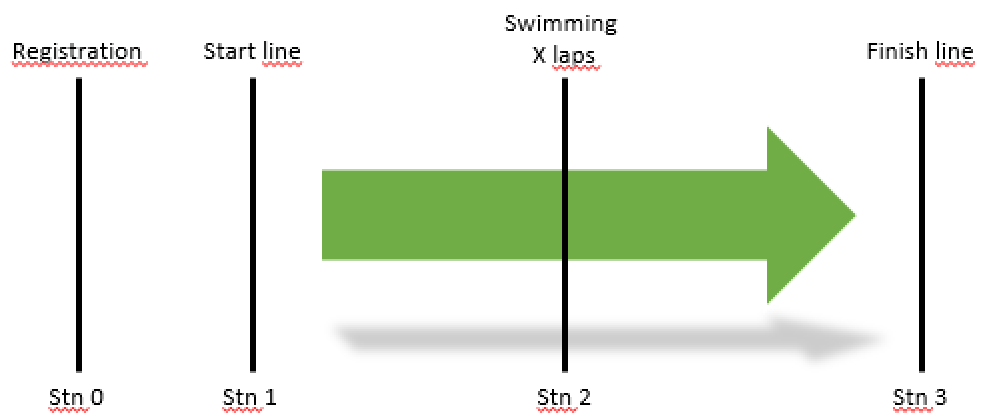


Fig 4.1 – A possible setup of a swimming competition using four RFID reader stations, of which the first is only used for registration of participants before the race.

This station should not be strictly coupled to the competition since the race should not have to start when the registration is performed. The registration process should not be started too early either though, to avoid competitors passing by the registrations check point and getting registered even though they have not really entered the starting line. The functionality is implemented in the existing time keeping software to make it easy to employ at a competition.

4.5 Commentator information

Discussions was held with both Västerås Tidtagning, the host of a triathlon event and the planned commentator for that event, in order to decide the design of the commentator software. Competitions are divided into different classes, such as seniors, juniors, women, men, etc., and some guidelines were decided concerning this. The user interface should be able to show all registered participants if needed, but also be able to only show one certain class of a race. It has to be very easy to navigate between different competitions and classes. Furthermore, the leading 5 participants of each class should be presented with a little more focus, since they are the most interesting in terms of commentating. For each registered participant, there should be columns showing position, name, club, time and trailing time from leader. The application is implemented as a standalone software, and it fetches all data from the database, to which the main system stores it. It checks the database for new data once every second and presents it on the screen. One second delay between the updates was agreed to be enough, but it can be altered if needed.
While it was useful to receive some insight from a commentator of a competition, it had to be taken under consideration that not all competitions are alike. Therefore the commentator software still has to be made in a bit more generic way, and this has been done under consultation with Västerås Tidtagning.

4.6 Implementing new functionality into the current system

Not a lot of things had to be altered in the current system to be able to include the new features. The current system is built in a way that new readers can easily be added in a generic fashion. This means the only addition needed for the middleware was an

application, implemented as a library, which can connect to the Raspberry Pi, receive all necessary information from it, and forward it into the current system in the correct format.
When adding the commentator software, the database containing all stored information, such as events, participants, timestamps, etc., had to be updated with additional records. Besides this, there were only some minor bugs discovered that had to be fixed, such as the system displaying the wrong time when recovering from a crash, due to a wrongfully entered time format.

5. Result

This chapter contains the results of this thesis work. Different tests of the system has been conducted during implementation. There has been no time for tests with actual participants, but with the use of antennas, readers, RFID tags and the constructed software, simulated races has been conducted. Different scenarios have also been staged during the tests, such as software failure, broken cables, etc., in order to identify any misbehavior.
The results are divided into four subcategories, corresponding to each research question, and are answered below.

Q1: How can a reliable communication between each reader and the main computer best be guaranteed?
This problem has been solved by connection a Raspberry Pi SBC to each RFID reader. The software on the Raspberry Pi forwards all registrations from the reader to the main computer. In case of a communication breakdown, the Raspberry Pi switches to store all new registrations on its own memory. At the same time, it continuously tries to reconnect to the main computer. Once the connection is reestablished, it sends all stored registrations to the main computer, and switches back to forwarding all new registrations in real time. Figure 5.1 shows a swim lane flow chart of the application in charge of making sure no registrations are lost.

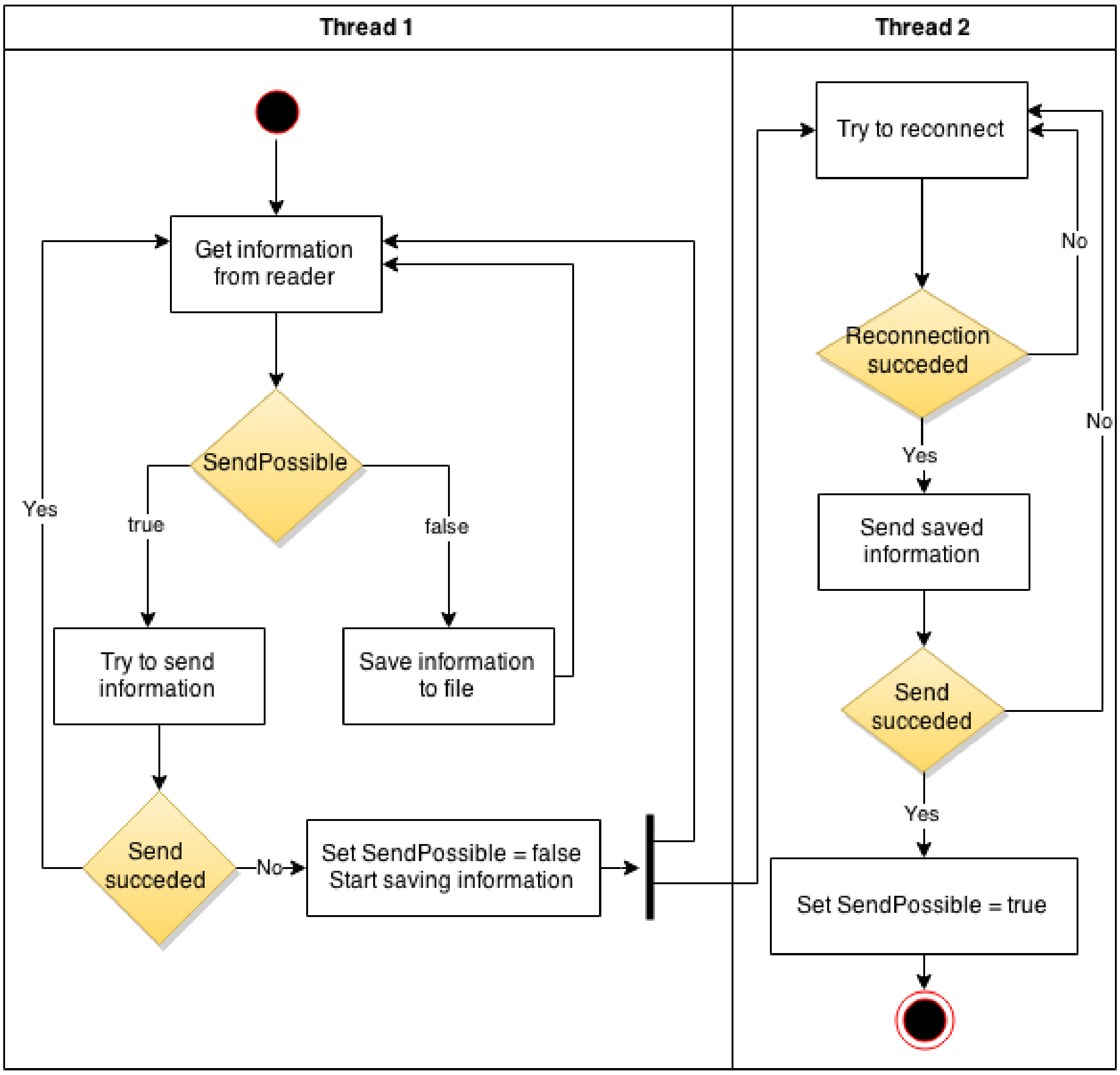
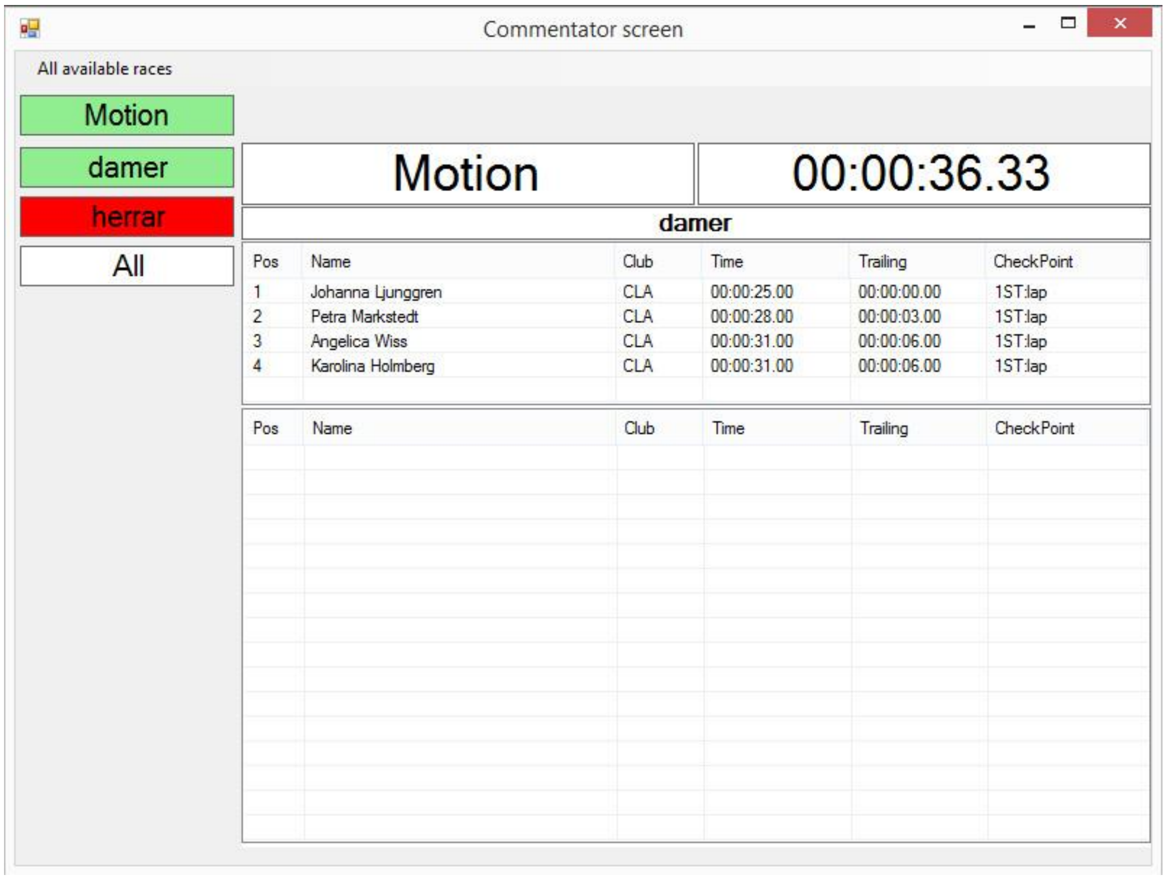


Fig. 5.1 – Swim lane flow chart describing the main functionality of the application in charge of making sure no registrations are lost during a competition

Q2: How is registration of each participant at the starting line best carried out?
This has been solved by adding some extra functionality to the existing software. An extra option to simply start a registration has been added when a new competition has been created. The actual registration can be performed by an identical setup as used for other time registrations. If needed, the station performing the starting line registration can immediately after be used as a normal check point station or finish line registration.

Q3: How should the information for the commentators be updated?
A stand-alone application was created to solve this problem. It displays the necessary information from the database containing all results from the events, by updating the screen once every second. The program can handle multiple competitions at the same time, and since all participants are divided into classes, it can show only one class at a time, or all of them on the same screen. Buttons on top and to the left of the results allows easy navigation between races and classes. If a participant of a class that is currently not on display runs past a check point, the corresponding button turns red for a couple of

seconds, to alert the commentator of it. A screenshot of the commentator application is shown in figure 5.2.



Pos	Name	Club	Time	Trailing	CheckPoint
1	Johanna Ljunggren	CLA	00:00:25.00	00:00:00.00	1ST:lap
2	Petra Markstedt	CLA	00:00:28.00	00:00:03.00	1ST:lap
3	Angelica Wiss	CLA	00:00:31.00	00:00:06.00	1ST:lap
4	Karolina Holmberg	CLA	00:00:31.00	00:00:06.00	1ST:lap

Pos	Name	Club	Time	Trailing	CheckPoint

Fig. 5.2 – Screenshot of commentator software, showing registered participant of the competition "Motion" and class "damer", marked with green, and simultaneously showing something has happened at the class "herrar" by making it red

Q4: How are the new additions best implemented in the current system?
Not much had to be altered in the current system in order to implement the new functionality. The system is built in a way that new readers can be added in a generic fashion. This means the only addition needed for the new reader setups to work was a class, added to the application as a library, which can connect to the Raspberry Pi, receive all necessary information from it, and forward it into the current system in the correct format.
Besides this and the addition for starting line registration and commentator system, there were basically not anything else needed to be adjusted in the original system.

6. Discussion

In this chapter, our own views around this thesis work are presented. The discussion is divided into sections covering each research question, and covers our thoughts on how each question was answered and if there were any particular decisions made along the way.

Q1: How can a reliable communication between each reader and the main computer best be guaranteed?

This was the main question, and the one we dedicated most time to solve. Västerås Tidtagning had a vision of using some kind of intermediate storage to make sure no registrations are lost in case of communication failure. After some research, we believed this to be the most suitable solution as well. However, some decisions had to be made along the way for it to work. From choosing the preferred hardware to which time synchronization protocol to use. We believe we managed to, through research, find out which were the most suitable choices, and applicate them accordingly. We thought about using the connectionless User Datagram Protocol (UDP) for communication instead of TCP, which is the more obvious choice for this type of communication. The reason was it would be easier in case of communication failure, since no connection had to be re-established. However, since communication failures still will be pretty rare, we opted to go for the more reliable TCP.

In the end we developed a product which is pretty unique in the area of sports timing systems. The risk of having cables broken, or computers crashing shouldn't be as worrying for the event hosts anymore, since registrations will still be stored in case it happens, when using this product. As of now, it obviously only works with TimeChip, the software intended to be used by Västerås Tidtagning. But it's pretty generic, and any program handling a timestamp, a RFID tag ID and an antenna ID as input should be able to make use of it, with just a little bit of adjustments.

Q2: How is registration of each participant at the starting line best carried out?

Being one of the smaller tasks in this thesis, registration of participants in a competition is rather important, especially in swimming competitions. The use of an extra station before the start line is, and probably will be, a valid solution for this. By using the same software and station setup for the registration as the one used for checkpoints in a competition, the need of extra stations is reduced. This brings the cost of the entire system down a bit since the most expensive part at the moment is the RFID readers. In the early stage we discussed if an extra screen should be used to list the registered participants to give them a verification that they have been registered. This was put aside later on since time was not enough and the importance of that was not as high as other parts.

Q3: How should the information for the speakers be updated?

When it comes to the speaker information there was a few discussions in terms of how the information should be presented and what kind of information that should be found. Since Västerås Tidtagning has a touch-screen computer with the resolution 4:3 it was decided that the format of which the commentator screen should be 4:3 and that the navigation should be considered touch friendly. The commentator is concerned with the information in a live like manner and therefore the screens is updated with a short interval and notifies the commentator if anything happens. The notification is done by making the button for the specific competition and class red for a few seconds. This is a way to make sure that the commentator will be able to see that new information is to be found but still not forcing him/her to look at that specific competition. Also the commentator doesn't have to look at all competitions for an event at the same time, it is possible to show or hide any competition at any time so that the commentator doesn't have to be flooded with controls to choose from.

Q4: How are the new additions best implemented in the current system?
As stated in the previous chapter, the existing software is very generic, and it is fairly easy to implement the new functionality into it. We believe we did it in a good fashion, and if you look at it from the other end, we simultaneously managed to make the Raspberry Pi pretty generic as well. It should be pretty easy to include the Raspberry Pi and its software into any system, as long as it can handle a timestamp, an RFID tag ID and an antenna ID as input.

7. Conclusion and Future work

This chapter summarizes the work made in the thesis. The first section contains the conclusion of this thesis work, summarizing the work done and what has been presented in this report. The second section presents ideas for future work related to this thesis.

7.1 Conclusion

This thesis work has been focused around improving a timing systems software for Västerås Tidtagning. In order for them to be able to use it for larger competitions, some new functionality had to be added. The required functionality was divided into four research questions and research was made in the related area of this thesis work, to find the state-of-the-art as well as how the research questions could be solved. The results show that we have improved the timing system used by Västerås Tidtagning, to meet the requirements from both themselves and the Swedish swimming federation. The main part has been to create a support application, implemented in a Raspberry Pi. When connected to a RFID reader, this software handles the registrations read by the RFID reader, sending them to the main computer if there is a connection, or storing them on its own memory otherwise. This way, the issue of unreliable communication between each reader and the main computer has been solved. Furthermore, we have created an additional functionality which allows each participant to be registered before each competition. This has been done to meet a required security demand from the Swedish swimming federation, in order to actually know how many are participating and, more importantly, how many are in the water. A commentator software has also been created, showing necessary information about each registration for the commentators of the competitions, so they can inform the audiences how the race is going. All these new functionality has been implemented in a way to fit into the existing software that is to be used by Västerås Tidtagning, to avoid the need of a completely new system.
All goals of this thesis work was reached within the required time and tests has been performed to verify that the software actually works.

7.2 Future Work

Some additional work can still be researched in order to improve the software even further. This includes removing redundant registration information already on the Raspberry Pi. As it is now, the Raspberry Pi sends, or stores, every single registration. The Motorola RFID reader can register over 1200 registration per second, meaning it can easily become a lot of registrations. The redundant information is then removed in the main computer, but if this was to be done already in the Raspberry Pi, the memory needed to store the registrations in case of a communication failure wouldn't necessarily be as big. The question is whether this would be a too big task for the Raspberry Pi to handle, while simultaneously handling new registrations and sending the unique ones to the main computer.
Other potential future work include making the communication from the Raspberry Pi to the main computer wireless. While this would eliminate the risk of cables breaking, wireless communication brings other reliability questions, and would need to be researched thoroughly before eventually adding it.

Bibliography

[1] S. Sundaresan, R. Doss, S. Piramuthu and W. Zhou, "Secure Tag Search in RFID Systems Using Mobile Readers," *IEEE Transactions on Dependable and Secure Computing,* vol. 12, no. 2, pp. 230-242, Apr. 2014.

[2] E. Rich, "E-ZPass and the Ohio Turnpike: Adoption and integration of," *Journal of Cases on Information Technology,* vol. 10, no. 1, pp. 32-51, 2008.

[3] F. Thiesse, E. Fleisch and M. Dierkes, "LotTrack: RFIDbased Process," *IEEE Pervasive Computing,* vol. 5, no. 1, pp. 47-53, 2006.

[4] S. C. Yu, "Implementation of an innovative RFID application in libraries," *Library Hi-Tech,* vol. 26, no. 3, pp. 398-410, 2008.

[5] "Västerås Tidtagning website," Västerås Tidtagning, [Online]. Available: http://www.tidtagning.se/. [Accessed 28 April 2015].

[6] B. Agee, "Agee Race Timing Website," Agee Race Timing, [Online]. Available: http://www.ageeracetiming.com. [Accessed 14 May 2015].

[7] "RFID Race Timing System website," RFID Race Timing System, [Online]. Available: http://www.rfidtiming.com. [Accessed 14 May 2015].

[8] M. Sarbini, S. Hassan, T. Jiann and P. Ahmad, "Design of a RFID-based speed monitoring system for road vehicles in Brunei Darussalam," in *International Conference on Computer, Communications and Control Technology (I4CT)*, Sept. 2014.

[9] K. Salunkhe, G. Gajjar, S. Soman and A. Kulkarni, "Implementation and Applications of a Wide Area Frequency Measurement System synchronized using Network Time Protocol," in *PES General Meeting | Conference & Exposition*, July 2014.

[10] R. Danymol, T. Ajitha and R. Gandhiraj, "Real-Time Communication System Design using RTL-SDR and Raspberry Pi," in *International Conference on Advanced Computing and Communication Systems (ICACCS)*, Dec. 2013.

[11] M. Rushanan, "Raspberry Pi RFID Client," Arbitrary Blog Execution, Nov. 2014. [Online]. Available: http://michael-rushanan.blogspot.se/2014/11/raspberry-pi-rfid-client.html. [Accessed 4 May 2015].

[12] R. Ratzel and R. Greenstreet, "Toward Higher Precision," *Magazine Queue - Networks,* vol. 10, no. 8, p. 40, Aug. 2012.

[13] A. Novick, M. Weiss, K. Lee and D. Sutton, "Examination of time and frequency control across wide area networks using IEEE-1588v2 Unicast Transmissions," in *Proceedings of the 2011 Joint IEEE International Frequency Control Symposium and European Frequency and Time Forum*, July 2011.

[14] "EPCGlobal website," GS1, [Online]. Available: http://www.gs1.org/epcglobal. [Accessed 7 May 2015].

[15] D. Mills, "The Network Time Protocol (NTP) Distribution," Website of David L. Mills, Phd, Professor, March 2014. [Online]. Available: http://www.eecis.udel.edu/~mills/ntp/html/index.html. [Accessed 4 May 2015].

[16] M. Roberti, "Motorola Introduces All-New Fixed Reader," RFID Journal, Jan. 2014. [Online]. Available: http://www.rfidjournal.com/articles/view?11354. [Accessed 10 May 2015].

[17] E. Brown, "Survey Results: Top 10 hacker SBCs," LinuxGizmos.com, May 2014. [Online]. Available: http://linuxgizmos.com/top-10-hacker-sbcs-survey-results/. [Accessed 27 April 2015].

[18] "Introducing the Raspberry Pi 2 - Model B, Benchmarks & Performance Improvements," Adafruit, Feb. 2015. [Online]. Available: https://learn.adafruit.com/introducing-the-raspberry-pi-2-model-b/performance-improvements. [Accessed 27 April 2015].

[19] M. Ekström, "Towards Predictable and Reliable Wireless Communication in Harsh Environments," Mälardalen University, Västerås, 2013.

[20] "FX7500 RFID Integrator Guide," Symbol Technologies, Inc., Feb. 2015. [Online]. Available: https://atgsupportcentral.motorolasolutions.com/content/emb/docs/manuals/MN000026A02a.pdf. [Accessed 14 May 2015].

[21] "LLRP Toolkit website," 2008. [Online]. Available: http://llrp.org/. [Accessed 1 May 2015].

[22] M. Rushanan, "LLRP Toolkit on ARM (Raspberry Pi)," Arbitrary Blog Execution, June 2013. [Online]. Available: http://michael-rushanan.blogspot.se/2013/06/llrp-toolkit-on-arm-raspberry-pi.html. [Accessed 4 May 2015].

[23] C. Pakanati, M. Padmavathamma and N. Reddy, "Performance Comparison of TCP, UDP, and TFRC in Wired Networks," in *International Conference on Computational Intelligence & Communication Technology (CICT)*, Feb. 2015.

[24] M. Kerrisk, "socket - Linux socket interface," Linux man pages online, May 2015. [Online]. Available: http://man7.org/linux/man-pages/man7/socket.7.html. [Accessed 11 May 2015].

[25] G. Shaw, "Prevent a process from terminating when writing to a broken pipe," micro HOWTO, [Online]. Available: http://www.microhowto.info/howto/prevent_a_process_from_terminating_when_writing_to_a_broken_pipe.html. [Accessed 11 May 2015].

[26] "Product Spec Sheet - Motorola FX7500," Motorola, 2013. [Online]. Available: http://www.abetech.com/AbeTech/files/0b/0b73baca-7b03-4033-ae93-21b3b431ac5f.pdf. [Accessed 4 May 2015].

[27] "Meinberg NTP Software Downloads," Meinberg Global, [Online]. Available: https://www.meinbergglobal.com/english/sw/ntp.htm. [Accessed 5 May 2015].

Appendix A

Drift Time test

These tests were carried out by connecting a computer and a Raspberry Pi with an Ethernet cable through a network switch, similar to when it will be used live. The computer then sent a signal to the Raspberry Pi, and at the same time registered its current exact time. When the Raspberry Pi received the signal, it made an own time stamp. These two time stamps were then compared to find out how much they differed from each other. In order to test different scenarios, the original startup time of the Raspberry Pi was altered for the tests, to simulate the unit not being used for different periods of time, with various original time difference. One final test was conducted with the Ethernet cable being pulled out between each time stamp, in order to discover the drift time in case of connection loss.
The different simulated scenarios were as follows:

- Test A: 12 hours original time difference.
- Test B: 3 months original time difference
- Test C: 6 months original time difference
- Test D: Cable unplugged between time stamps.

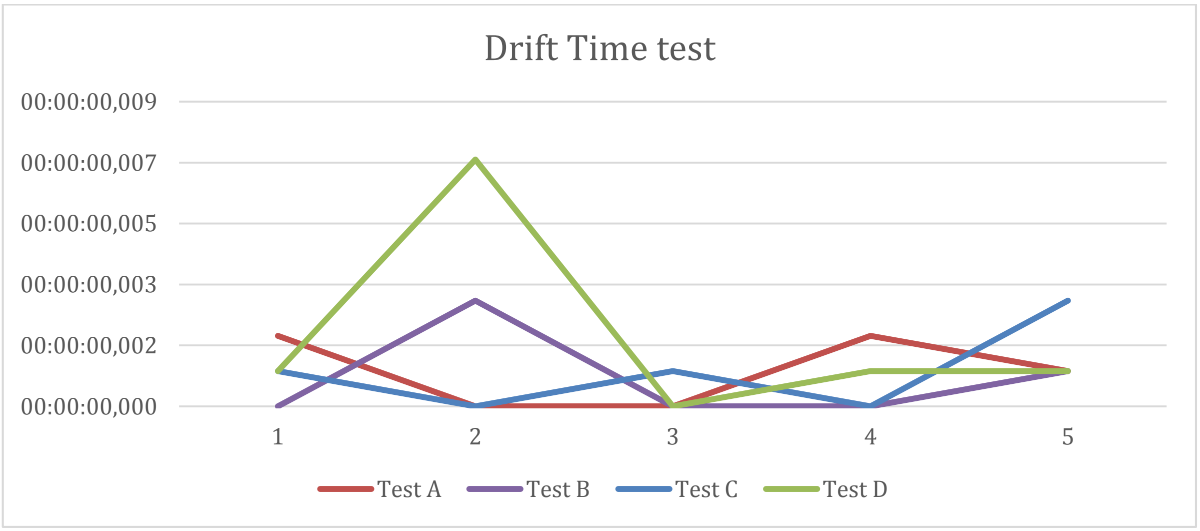


Fig. A.1 – Test results from the Drift Time test, conducted to find out how well the built in time synchronization of NTP works

The results, presented in figure A.1, show very little drift time, no matter the original time difference. Test D showed that after first synchronization, the Raspberry Pi drifted 6ms during the first hour. When reconnected, it synced again, and this time adjusted the clock frequency to make it more accurate, making the following time stamps almost perfectly synced, even with no connection.
These tests shows that the built in time synchronization is accurate enough Västerås Tidtagning's requested precision and this thesis work.