Synchronization of Acoustic Sensors in a Wireless Network

Malin Johansson

External supervisor: David Lindgren
Supervisor: Peter Johansson
Examiner: Michael Josefsson
Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår. Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera åktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art. Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsmann i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart. För ytterligare information om Linköping University Electronic Press se förlagets hemsida [http://www.ep.liu.se/].

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years starting from the date of publication barring exceptional circumstances. The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: [http://www.ep.liu.se/].

© Malin Johansson
Abstract

Geographically distributed networks of acoustic sensors can be used to identify and localize the origin of acoustic phenomena. One area of use is localization of snipers by detecting the bullet’s shock wave and the muzzle blast. At FOI Linköping, this system is planned to be adapted from a wire bounded sensor network into a wireless sensor network (WSN). When changing from wire bounded communication to wireless, the issue of synchronization becomes present. Synchronization can be achieved in multiple ways with different benefits depending on the method of choice. This thesis studies the synchronization method of using the highly accurate clock in Global Navigation Satellite System (GNSS) modules. This synchronization method is developed into an independent time stamping device that can be connected to each sensor in the WSN. This ensures that all sensors are synchronized to Coordinated Universal Time (UTC). The thesis starts with a pre-study where different solutions are investigated and evaluated. After the pre-study, a development stage is begun where the best solution is developed into a model to be easily implemented in the future. The result is a model consisting of a microcontroller, a timing module, and an ADC with built-in filter and amplification.
Acknowledgments

Firstly, I would like to thank FOI Linköping for the opportunity to do my thesis with you. A special thanks to my external supervisor David Lindgren at FOI Linköping, for the help with customizing this thesis and for all of the guidance in my work. At Linköping University, I would like to thank my supervisor Peter Johansson and my examiner Michael Josefsson, for all of your help and support.

At last, I would like to thank my family for supporting me and for believing in me. You have always given me the motivation to do the best that I can.

Linköping, June 2019
Malin Johansson
## Contents

| Abstract | iii |
| Acknowledgments | iv |
| Contents | v |
| List of Figures | vii |
| List of Tables | ix |

1. Introduction
   1.1 Background ........................................ 1
   1.2 Aim ........................................ 1
   1.3 Research questions ..................................... 2
   1.4 Delimitations ...................................... 2
   1.5 Related work ..................................... 2

2. Wireless Sensor Networks ........................................ 4
   2.1 Acoustic WSN ...................................... 4
      2.1.1 Synchronization .................................. 5
      2.1.2 Time stamping .................................. 5
      2.1.3 Latency ........................................ 5

3. Hardware ........................................ 6
   3.1 Time module ....................................... 6
      3.1.1 Voltage Controlled Oscillator .................... 6
      3.1.2 Short term errors ................................ 8
      3.1.3 Long term errors ................................ 9
      3.1.4 Phase locked loop (PLL) ......................... 9
      3.1.5 GPS Disciplined Oscillator (GPSDO) ............... 10
   3.2 GNSS module ........................................ 11
      3.2.1 Pulse Per Second (PPS) .......................... 11
      3.2.2 National Marine Electronics Association (NMEA) .... 12
    3.2.3 NEO-M8T ...................................... 12
   3.3 Microcontroller ....................................... 13
      3.3.1 Raspberry Pi 3 Model B+ .......................... 13
      3.3.2 Arduino MKR WiFi 1010 .......................... 14
   3.4 Dynamic microphones .................................. 14
      3.4.1 Butterworth filter ................................ 15
   3.5 Analog to Digital Converter ............................ 16
      3.5.1 TLV320ADC3100 .................................. 17

4. Software ........................................ 18
List of Figures

2.1 Explanation of a Wireless Sensor Network with star topology. .......................... 4
2.2 Time stamping one data packet of 0.1 seconds in 48 kS/s with included error mes-

sage .................................................................................................................. 5
3.1 An example of the output frequency with respect to the input voltage of a VCO. .... 6
3.2 a) Accurate and stable. b) Accurate but not stable. c) Not accurate but stable. d) 

Not accurate and not stable. .............................................................................. 7
3.3 Quantization error. ....................................................................................... 8
3.4 Jitter. ............................................................................................................. 9
3.5 Block schematic of a Phase Locked Loop ....................................................... 10
3.6 A GPS Disciplined Oscillator with 100 Hz. Vref is the voltage level that does not 

change the output frequency. ........................................................................... 10
3.7 Triangulation of the position of a receiver using three satellites. Seen from above 
at the left and from the side to the right. t1, t2 and t3 represent the distances from 
the satellites to the receiver. ............................................................................ 11
3.8 PPS signal and GPS data simplified. ............................................................. 11
3.9 Pin placement of the NEO-M8T testing board. .............................................. 13
3.10 Raspberry Pi 3 Model B+. ........................................................................... 14
3.11 Arduino MKR WiFi 1010. ............................................................................ 14
3.12 Frequency response of different filter orders. A0 = pass band gain, fc = cut-off 
frequency. ........................................................................................................ 15
3.13 Third order butterworth filter consisting one first order filter and one second order 
filter. .................................................................................................................. 15
3.14 The errors of an ADC caused by a faulty time stamp. In this case, the sample 
frequency is 100 kHz, and one sample is delayed with 5 µs which results in a 
voltage error of 50 mV. ................................................................................... 16
3.15 V = 5sin(2π fc t), where fc = 48 kHz. The red arrow marks one point where the 
sinusoidal signal has its maximum derivative. .................................................. 17
3.16 The required external components to the TLV320ADC3100 in the most common use

4.1 A WiFi router creates a path for the smart phone to access the Internet. The path 
between the smart phone and the WiFi router is connection-less. ...................... 18
4.2 The use of a WiFi router to establish a communication path between devices with-
out the need of Internet connection. In this case, an Arduino, a smart phone and a 
computer have the ability to communicate over the network without being physi-
cally connected to each other. ......................................................................... 19
4.3 An overview of U-center. The red box is the configuration view used for settings. 
The blue box is a message overview of the NMEA sentences that are available 
and what they contain. The green box is the text view, where all current NMEA 
sentences from the GNSS module are shown. To the right in U-center, there is an 
overview of the time, position, speed and satellites in view. ............................ 20
4.4 Data transmission of the I2C protocol. R/W stands for read/write, and ACK is an
acknowledgement bit set by the receiver. 20

4.5 Possible I²S system configurations. 21

5.1 Block schematic of the first alternative solution. The VCO is connected to a
counter, and with the help of the PPS of the GNSS module, and a microproces-
sor (Clock CTRL) to handle the clock, a GPSDO is created to ensure the frequency
of the VCO. The Clock CTRL sends the time stamps to a second microcontroller
that handles the ADC and sends the samples with the correct time stamp. 22

5.2 Block schematic of the second alternative solution. The Clock CTRL is im-
plemented in the other microcontroller. 23

5.3 Block schematic of the third alternative solution. In this solution, all buffers are
excluded and more of a sample based time stamp is implemented. The PPS of
the GNSS module is also processed by the ADC and can than be found with its
matching audio sample to achieve correctness. The microcontroller also reads how
many samples are provided in a second to frequency lock the VCO. 23

5.4 Block schematic of the first alternative solution using NEO-M8T. The time stamps
are set in the CTRL block to unburden the Arduino so that it can focus on only
matching the samples to the correct time stamp. The sample buffer can be internal
or external. 28

5.5 Block schematic of the second alternative solution using NEO-M8T. The ADC con-
verts both the audio from the sensor and the PPS from the NEO-M8T. Having this
information provides the Arduino with the exact sample that should be matched
to the start of every second. From this, the accuracy of the 48 kHz signal ensures
that the timing for the other time stamps is correct. 28

5.6 Block schematic of the alternative solution using NEO-M8T and the internal ADC
of the Arduino. The PPS is connected to an interrupt pin on the Arduino to ensure
correct pairing with the time stamp and the samples. The 48 kHz is also connected
to an interrupt pin to control the sample frequency. 28

5.7 The setup for testing the delay of the interrupt. 29

5.8 The delay of the interrupt. 29

5.9 The sending time of 0.1 s of data. 30

5.10 The final block schematic of the desired system. 31

5.11 The period of the 12 MHz TIMEPULSE2 from the NEO-M8T. 8 pulses with 83.33
ns period equals 750.0 ns. The signal is measured both over several pulses and
pulse by pulse with accurate results. 32

5.12 The specifications of filter A and amplification of the TLV320ADC3100. The red
horizontal line indicates the level where the signal is defined as muted. 33

5.13 A recommended order of performing the settings of the TLV320ADC3100. 33

5.14 A proposed schematic of the final design alternative. The NEO-M8T acts as the
master of the I²S communication between the Arduino and the ADC by setting
TIMEPULSE2 to 12 MHz for the MCLK. The 12 MHz signal is also divided to 48
kHz to control the WCLK. 36
List of Tables

3.1 Table of existing NMEA sentences ........................................ 12
3.2 Table of ZDA sentence data .................................................. 12
3.3 Time pulse specifications of the NEO-M8T ................................ 13
6.1 Table of evaluation of manual phase lock compared to NEO-M8T .... 25
6.2 Table of evaluation of Raspberry Pi 3 compared Arduino MKR WiFi 1010 ... 27
6.3 Table of evaluation of the design alternatives .............................. 30
6.4 Specifications of the TLV320ADC3100 PLL. MCLK is the input master clock and $f_s$ is the sample frequency .................................................. 32
1 Introduction

In this thesis, a method of an independent synchronization for wireless acoustic sensors is studied and developed. The research question has been put into a context for creating an interesting perspective with narrow tolerances regarding synchronizing time error. The context is described further in the following chapters.

1.1 Background

The Swedish armed forces uses multiple tools to be able to localize targets in a varying terrain. One of the tools is a geographically distributed network of acoustic sensors to be able to localize snipers with sensor fusion. Sensor fusion in this context means that data from multiple sensors are gathered to accomplish a reliable measurement to be able to identify and localize targets [1]. The localization is calculated from an algorithm that uses the time stamp of the audio from a bullet’s shock wave and the time stamp of the audio from the muzzle blast. With the combined information from several sensors the algorithm can calculate the origin of the sound, i.e. the location of the shooter. The algorithm and the need of synchronization is explained further in [2].

1.2 Aim

The purpose of this thesis is to evaluate and develop a system that, in the future, will solve the time synchronization of a sensor network with the help of a Global Navigation Satellite System (GNSS) module, for example a GPS. The main assignment of the sensor network is to discover and localize the origin of acoustic phenomenon. To guarantee that this system works properly, an accurate time synchronization is needed. Wire bounded communication ensures this, but with wireless communication, the synchronization becomes a problem. In the case studied in this thesis wire bounded communication is both unpractical and expensive due to the length of the cables which explains why this thesis is needed.
1.3 Research questions

By time stamping the audio samples with a highly accurate clock, one can guarantee that the sensors are having synchronized clocks. The time stamp is accurate and matched with the correct sample which leads to it being able to be analyzed afterwards together with the audio from the sensor samples from the other sensors with the same time stamp. The accuracy of the time stamping in this case has a maximum allowed time synchronization error of 0.01 milliseconds. In this thesis different designs of a solution are investigated. Those who fulfill the time requirement are evaluated according to work load, size, flexibility and cost. The design that best answers to these demands is developed for later on easy implementation. The research questions are as following:

1. What are the alternatives to achieve synchronized time stamps of wireless acoustic sensors using GNSS?
2. Which alternative guaranties a maximum time error of 0.01 milliseconds compared to Coordinated Universal Time (UTC)?
3. Is the proposed solving method feasible?

1.4 Delimitations

An individual synchronization device that always ensures synchronized samples with high accuracy is not only wanted in this particularly use. This thesis however will have its main focus in solving the case of acoustic sensors, but by having generality in mind hopefully it will result in a model that can be implemented into a synchronizing device for different kinds of sensors with minor modifications.

Since the thesis is performed at limited time, three design alternatives of solutions was generated beforehand by the client. The alternative solving methods can be adjusted and other new alternatives are also allowed. Block schematics of the provided alternatives are presented later in chapter 5.

1.5 Related work

Time synchronization has been studied a lot in the past decade and most of the solving methods can be divided into four main categories [3]:

1. Using time-synchronizing packets that use existent algorithms to synchronize the network. Known algorithms from this type is Reference Broadcasting Synchronization (RBS) [4], Timing-sync Protocol for Sensor Networks (TPSN) [5] and Flooding Time Synchronization Protocol (FTSP) [6]. They all use a tree-based topology with one master that provides timing for all nodes in the network. This causes time latency to accumulate if there are several nodes in one network and can therefore not guarantee accuracy.

2. A method using radio signals. This method can be divided into two parts.
   a) The use of the public radio stations that send data including time which is synchronized to UTC. The advantages of this method is the small amount of power consumption combined with no limits to having a clear sky compared to the GNSS method presented below. The disadvantage of this method is the accuracy. It can only guarantee a maximum synchronization error of a few dozen milliseconds [7]. This low accuracy depends on that the time provided only has a resolution of 0.1 seconds and is only provided once every minute [8].
b) Using a dedicated radio transmitter that sends a ping to nearby sensor nodes. The ping is a sort of message and when the sensor nodes receive it, the sensor nodes can synchronize to the ping and in that way they are synchronized to each other. This does not require the sensor nodes to be synchronized to UTC.

3. Using connection to a network such as Ethernet. Precision Time Protocol (PTP) and Network Time Protocol (NTP) are examples of this type of method. Both are protocols with separate timing networks that a system can be connected to. The main idea is that all nodes with sensors communicate with each other to decide a common time base that all nodes can calibrate to. These protocols are extremely time accurate, especially PTP, but they are not applicable in this case since the wanted system should work without too much data traveling in the network. [9]

Another issue with this kind of time synchronizing is the sampling frequency limit. Software solutions can not time stamp samples faster than 120 000 samples per second. This limitation is because of the delay when software starts accessing the hardware buffer. Hardware methods of time stamping is thereby preferable. [10]

4. A GNSS based method that uses the accuracy of the clock in a GNSS module which is explained further in chapter 3.2. Using a GNSS module at every sensor node to ensure that all sensors are equivalently synchronized to UTC. This method is what will be investigated further in this thesis. The time error of one solution of this method has been studied more in [3] where a maximum time error of ±250 nanoseconds was achieved which makes this method suitable for this thesis.
2 Wireless Sensor Networks

A Wireless sensor network (WSN) can be defined as a group of sensors that read the environment and communicates with each other or with a master device over a wireless link. The data of the sensors can then be combined and processed to extract the desired information. WSNs have been rapidly developed since the capabilities of small, low-cost and low-power components have increased and are now part of many systems. In figure 2.1 a wireless sensor network where all sensor nodes communicate directly to the master device is shown. This structure is also called star topology. [11]

![Figure 2.1: Explanation of a Wireless Sensor Network with star topology.](image)

Other types of structures are tree topology and MESH topology, where the sensor nodes can communicate with each other as well as the master. This will not be used in this thesis and is therefore not described more carefully.

2.1 Acoustic WSN

Acoustic WSNs have a wide range of areas of use. Fusing multiple wireless sensor data using different algorithms is used for localization, detection, monitoring and more. WSNs are used in military, medical, environmental and industrial areas thanks to its low cost and flexibility compared to wired communication. [12]
2.1.1 Synchronization

When combining information from several sensors it is important that the combining is happening in real time and that they are time synchronized. In WSNs, each sensor has its own internal clock and all events in the sensor is relative to this clock. What is unknown is the time and status of the clocks of the other sensors in the same network. To be able to process the data provided by the sensors correctly, it is of high importance that these are synchronized. This can be done in the different ways explained in Chapter 1.5. The synchronizing method of this thesis is a GPS based time stamping of the samples. [12]

2.1.2 Time stamping

If the sample frequency of the sensors is sufficiently accurate and with its frequency locked to UTC all individual samples can be matched with a time stamp. The samples from all wireless sensor nodes in the WSN are sent to a master controller. The master controller is in control of the algorithm and can then combine all data with the same time stamp from multiple sensors. This method requires high resolution of the time stamp and is not so data efficient since the bits required for the time stamp has to be transmitted for every sample.

To reduce the amount of data, a buffer of samples containing (for example) 0.1 s of data can be matched to a single time stamp. In this case, knowing that the sample frequency is stable and accurate and knowing which sample in the buffer the time stamp belongs to is crucial. This method however is more data efficient, and the time stamp resolution is only 0.1 s. It is also easy to add a byte for error messages, or how many satellites the GNSS module has in sight. See figure 2.2.

![Figure 2.2: Time stamping one data packet of 0.1 seconds in 48 kS/s with included error message.](image)

2.1.3 Latency

According to [13], the biggest source of synchronization error is message delivery latency. This latency can be divided into following four categories:

- **Send Time**: The time taken to build the message by the sender.
- **Access Time**: This is the time it takes to interrupt what the processor is doing to get ready to transmit.
- **Propagation Time**: The time for the message to transport from transmitter to receiver.
- **Receive Time**: The processing time for the receiver to store the message.

Latency will most likely not affect this thesis since the data containing the time stamp with the audio from the sensor is matched on beforehand. The latency will only cause a delay of the time a sound is being made to it being processed in the algorithm.
To ensure synchronization of acoustic sensors a time stamp can be paired with data from the sensor at that time. One way of making sure that the time stamp is accurate is to use a GNSS module. This method has previously been proven to have extremely small time errors in [3]. Following chapters will describe the hardware needed for synchronization.

3.1 Time module

The need for some sort of time module to achieve time synchronization is obvious. The following chapters explain what kinds of time errors these clocks cause and gives a possible solution for this issue, the phase locked loop.

3.1.1 Voltage Controlled Oscillator

Voltage controlled oscillators (VCOs) have a nominal frequency which is to some length adjustable. It is adjusted by setting the input voltage of the VCO to a specific level according to a graph. See figure 3.1 for an example on how this graph might look.

```
Figure 3.1: An example of the output frequency with respect to the input voltage of a VCO.
```
These graphs are different for different VCOs and can be found in their datasheets. By reading it, the tuning input voltage can be set to an optimal level for that specific case. To have any use of a VCO, there must be some sort of frequency reader connected to it so that the current frequency is known. Depending on if it is higher or lower than the desired frequency, the input voltage should be increased or decreased until the desired frequency is achieved.

Even if an oscillator is set to a specific frequency, some sort of errors will occur. According to [14] the time errors of oscillators can be divided into two main parts, accuracy and stability. Accuracy is how close the output frequency is to the nominal frequency over time, but fluctuation is not included. Stability on the other hand is a measurement of how little or how much the frequency varies. Figure 3.2 illustrates the two types of time errors.

![Figure 3.2](image)

Figure 3.2: a) Accurate and stable. b) Accurate but not stable. c) Not accurate but stable. d) Not accurate and not stable.

To get more understanding of these errors, the oscillator output can be described as a \( \sin (t) \) wave with a frequency of \( 2\pi v_0 \), a disturbance function \( (A + B(t)) \) and a possible time variation \( \psi(t) \). [15]:

\[
v(t) = (A + B(t)) \sin (2\pi v_0 t + \psi(t))
\]  

(3.1)

- \( t = \) true time
- \( v(t) = \) output voltage from the oscillator
- \( A = \) nominal amplitude
- \( B(t) = \) amplitude error
- \( v_0 = \) nominal frequency
- \( \psi(t) = \) phase error of the oscillator

Since the amplitude has nothing to do with time synchronization, a model of the phase error is described as following. [16]:

\[
\psi(t) = \phi + \dot{\phi} t + \ddot{\phi} \frac{t^2}{2} + q(t)
\]  

(3.2)
\( \phi = \) phase offset
\( \dot{\phi} = \) frequency offset (accuracy)
\( \ddot{\phi} = \) frequency drift (stability)
\( q(t) = \) random non-deterministic error

What this shows is that if two separate clocks start at different phases, there will be a constant phase offset in the two clocks independent of time as long as they have the same frequency drift and frequency offset, not counting with the random non-deterministic error. If there is a frequency offset however, the phase error \( \psi(t) \) will increase linearly with the time. In most cases there is also some sort of frequency drift compared to to nominal value, which causes the phase error to increase with a factor of \( \frac{t^2}{2} \). There will also be a random error \( q(t) \), caused of short term errors described in next chapter. [16]

### 3.1.2 Short term errors

Short term errors are often random and non-deterministic of statistical nature, meaning that the same input signal can cause different errors. Short term errors can be described as a stochastic variable \( X \), characterized by a mean \( \mu \) and a standard deviation \( \sigma^2 \).

\[ X \sim (\mu, \sigma^2), \text{ where } \mu = 0. \]

These errors can never be eliminated, but one can often be aware of them and thereby chose solutions that have small short time errors. Short time errors can be divided into two parts, quantization errors and jitter. Quantization errors do not affect the long-term stability since it has a mean of 0. It is shown in figure 3.3 [16]

![Figure 3.3: Quantization error.](image)

The meaning of jitter is differences of period time of the oscillator frequency and is caused by lack of accuracy. When the oscillator frequency period is constantly wider or more narrow than nominal width, it is a long term error of stability. Jitter is illustrated in figure 3.4 [16]
Oscillators that are used as a clock for Analog to Digital Converters (ADCs) have especially high demands on jitter when time stamps are used as synchronizing method. This is because the time stamps created might not match the registered audio level at that time if the oscillator is not jitter free. This will result in faulty data which is described more in chapter 3.5.

The short time errors have no long term effect on the stability or accuracy. The understanding of short time is however important since it always will be present and cannot be prevented. The maximum value of the random non-deterministic error ($q(t)$) therefore has to be known ($quantization_{max} + jitter_{max}$) so that the overall maximum error ($\psi(t)$) can be calculated.

3.1.3 Long term errors

It has already been mentioned that the long term time errors such as stability and accuracy have large effects on the oscillator frequency over time. The errors are depending a great deal on which type of oscillator and can be read from their datasheet. Luckily, this can be compensated for with the help of a phase locked loop.

3.1.4 Phase locked loop (PLL)

Phase locking means controlling the phase of a signal so it matches the phase of a input reference signal. By having an accurate and stable master reference input, several clocks can be locked to this frequency to guarantee that all clocks run at the same frequency over time. The main components of a PLL are a phase detector, a filter and a VCO. The phase detector compares the phase of the reference input with the VCO phase and adjusts the voltage amplitude to tune the VCOs frequency. The signal is then filtered to avoid noise and after filtering it sets the tuning input voltage of the VCO which adjusts its output frequency so it can be fed back into the phase detector. In figure 3.5 the use of a PLL is illustrated.
Figure 3.5: Block schematic of a Phase Locked Loop.

The oscillator output when using a PLL can then be described as a sinusoidal function similar to equation 3.1:

\[ v_o(t) = (A + B(t)) \sin(2\pi v_i t + \beta(t)) \]  
\[ t = \text{true time} \]
\[ v_o(t) = \text{output voltage from the oscillator} \]
\[ A = \text{nominal amplitude} \]
\[ B(t) = \text{amplitude error} \]
\[ v_i = \text{input frequency} \]
\[ \beta(t) = \text{phase error of the oscillator} \]

Where \( \beta(t) \) can be described as following:

\[ \beta(t) = \dot{\phi}(t) + q(t) \]  

Compared to the equation 3.2, this time, there is only left a possible drift \( (\dot{\phi}(t)) \) that is corrected every time the phase detector is updated and the random non-deterministic error \( q(t) \).

By using a PLL, an oscillator can be tuned to have a stable and accurate frequency as long as it is provided with a high precision clock as reference input. High precision clocks are for example found in GNSS modules.

### 3.1.5 GPS Disciplined Oscillator (GPSDO)

To achieve a high precision oscillator a combination of a GPS module and a phase locked loop can be used. The accuracy of a pulse sent from the GPS once every second is explained further in chapter 3.2.1. A bit counter can be connected to the output signal and the phase detector can be used as a comparator triggered by the pulse from the GPS. If the value of the counter is higher or lesser than the expected frequency, a tuning voltage can be set. An example of this is shown in figure 3.6.

Figure 3.6: A GPS Disciplined Oscillator with 100 Hz. Vref is the voltage level that does not change the output frequency.
If the counter only counts to 99 in one second, the VCO runs to slow and then V1 is adjusted to increase the frequency.

3.2 GNSS module

The term GNSS is a collection of all satellite navigation systems, for example GPS which is the American system, Galileo which is the European system and the Russian GLONASS system. For this thesis, GNSS as a term will be used when several systems are combined. Simplified, the positioning by GNSS is calculated by measuring the time it takes for a signal to reach the receiver from a satellite.

There are approximately 75 satellites operating which send a continuous signal containing the current time using an atom clock. When the signals from the satellites reach a receiver it is compared with the current time of the receiver. The time difference has an approximately linear relation to the distance between the transmitter and the receiver. By combining at least three satellites, the current position of the receiver can be triangulated into a position. For this to work properly, GNSS receivers are equipped with a stable oscillator as a clock that synchronizes to UTC. See figure 3.7

![Figure 3.7: Triangulation of the position of a receiver using three satellites. Seen from above at the left and from the side to the right. \( t_1, t_2 \) and \( t_3 \) represent the distances from the satellites to the receiver.](image)

3.2.1 Pulse Per Second (PPS)

At the beginning of every second, GNSS modules generate a signal called PPS. This signal has very high time accuracy of approximately 5 ns compared to the GNSS time. This makes it a good reference signal when synchronization is wanted. This signal can be used to phase lock a voltage controlled oscillator, or to determine when to read and set the time. See figure 3.8

![Figure 3.8: PPS signal and GPS data simplified.](image)
3.2.2 National Marine Electronics Association (NMEA)

Between two PPS signals, a serial signal containing information about time, status of the satellites etcetera is generated by the GNSS modules (Figure 3.8). This can be used to extract current time when time stamping is needed. Since NMEA only contain 1 second resolution, sample frequencies higher than this forces the signal to be complemented with the wanted resolution.

NMEA consists of several messages, also called sentences, with different contents such as information about time or position. Different sentences can be accessed by different settings. The different sentences are listed below in table 3.1 with a short description. [19]

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GGA</td>
<td>Time, position and fix type data.</td>
</tr>
<tr>
<td>GLL</td>
<td>Latitude, longitude, UTC time of position fix and status.</td>
</tr>
<tr>
<td>GSA</td>
<td>GPS receiver mode, used satellites, DOP values.</td>
</tr>
<tr>
<td>GSV</td>
<td>Number of satellites, satellite ID, elevation, azimuth, SNR.</td>
</tr>
<tr>
<td>MSS</td>
<td>SNR ratio, signal strength, frequency, bit rate from receiver.</td>
</tr>
<tr>
<td>RMC</td>
<td>Time, date, position, course and speed data.</td>
</tr>
<tr>
<td>VTG</td>
<td>Course and speed data.</td>
</tr>
<tr>
<td>ZDA</td>
<td>PPS Timing message (Synchronized to PPS).</td>
</tr>
<tr>
<td>150</td>
<td>OK to send message.</td>
</tr>
</tbody>
</table>

For synchronization, ZDA is an appropriate sentence of choice. Following sentence is an example of the ZDA message and the variables are explained in table 3.2.

ZDA message example:

\$GPZDA,101500,10,05,2019,00,00*4F

<table>
<thead>
<tr>
<th>Name</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message ID</td>
<td>$GPZDA</td>
<td>ZDA header</td>
</tr>
<tr>
<td>UTC Time</td>
<td>101500</td>
<td>HH:MM:SS</td>
</tr>
<tr>
<td>Day</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Month</td>
<td>05</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>2019</td>
<td></td>
</tr>
<tr>
<td>Local zone hour</td>
<td>00</td>
<td>Offset to UTC</td>
</tr>
<tr>
<td>Local zone minute</td>
<td>00</td>
<td>Offset to UTC</td>
</tr>
<tr>
<td>&lt;CR&gt;&lt;LF&gt;</td>
<td>*4F</td>
<td>End of message</td>
</tr>
</tbody>
</table>

3.2.3 NEO-M8T

NEO-M8T is a timing module produced by U-blox. The module has an integrated GNSS module compatible with both GPS, Galileo and GLONASS. NEO-M8T has high precision timing features that allows the user to set two different time-pulses to a given frequency with its phase locked to PPS and with the specifications in table 3.3. The NEO-M8T is also compatible with the NMEA protocol explained in the previous chapter. [20, 21]
Table 3.3: Time pulse specifications of the NEO-M8T

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-pulse frequency</td>
<td>0.25 Hz–10 MHz</td>
</tr>
<tr>
<td>Time-pulse accuracy</td>
<td>≤20 ns if clear sky and ≤500 ns if indoors.</td>
</tr>
<tr>
<td>Time-pulse jitter</td>
<td>±11 ns</td>
</tr>
</tbody>
</table>

To the NEO-M8T there is a board for easy start and testing applications with the pin placement in figure 3.9. The data is transmitted and received with the RX and TX pin, and the time-pulses are transmitted from TIMEPULSE and TIMEPULSE(2). An USB adapter can be connected to the board to connect the NEO-M8T to a computer where settings can be made easily by using U-blox program U-center (see chapter 4.2.1).

3.3 Microcontroller

For this thesis, a microcontroller is needed to control and process the data from the sensors and time from the GNSS module.

A microcontroller is a type of computer that contains a processor, memory, inputs and outputs and peripherals. It performs one specific task or application when being programmed. Two low cost brands that focuses at the market for beginners use are Raspberry Pi and Arduino. In following chapters one microcontroller from each brand will be presented more carefully.

3.3.1 Raspberry Pi 3 Model B+

Raspberry Pi 3 Model B+ is a microcontroller that runs with the 64-bit Broadcom BCM2837B0 processor. It has 1 GB RAM and an internal clock of 1.4 GHz. The controller has 40 programmable input/outputs and compatibility with both 2.4 GHz and 5 GHz WiFi. Interrupt timing delays are long in the Raspberry Pi since it is a software based microcontroller with a priority schedule rather than interrupt priority which is seen in hardware based controllers. A picture of the Raspberry Pi 3 Model B+ is presented in figure 3.10.

All models of the Raspberry Pi has its own operative system called Raspbian, and the microcontroller is also compatible with other operative systems if required. The Raspberry Pi is connected to a keyboard, mouse and a screen to run as a computer.

13
3.3.2 Arduino MKR WiFi 1010

The Arduino MKR WiFi 1010 is equipped with a ESP32 module made by U-BLOX which makes it compatible with 2.4 GHz WiFi. The processor of this controller is a 32-bit SAMD21 Cortex-M0+ that runs on a 48 MHz internal clock and has 32 KB SRAM and an additional 256 KB Flash memory. 8 programmable digital input/outputs, 1 analog output and 7 analog inputs with an internal ADC of 8–12 bits are available. The Arduino is also equipped with 8 external hardware based interrupt pins. Figure 3.11 shows an Arduino MKR WiFi 1010.

Compared to the Raspberry Pi, the Arduino does not have its own operative system and is programmed from a separate computer. It also has more hardware support for communication protocols (for example I²S which is explained further in chapter 4.3.2) than the Raspberry Pi.

3.4 Dynamic microphones

Dynamic microphones are acoustic sensors with no supply voltage needed. They convert sound into an electrical signal by using electromagnetism. This causes the voltage levels to be relatively small, in the millivoltage range. To transform the signal output from the microphone and to reduce noise levels, an amplifying filter has to be implemented.

For this thesis, a microphone with a frequency range from 40 Hz to 16 kHz and an output range from 0–30 mV is used. This microphone is only used for this thesis to prove the synchronization and has therefore no further requirements.
3.4.1 Butterworth filter

In applications where different frequencies occur at different times and the time between the different frequencies is of importance, a minimized phase distortion is wanted, also called a minimized group delay.

Bessel filters are the best type of filter since they have constant group delay. For this thesis however, a butterworth filter was chosen since this filter has the least distortion of sound and is therefore appropriate for testing the system.

A butterworth filter is a type of active filter with a flat frequency response in the frequency pass band. For the case of noise reduction a low pass filter is appropriate. By increasing the order of the filter, the frequency response drops faster. See figure 3.12.

![Frequency response of different filter orders.](image)

Figure 3.12: Frequency response of different filter orders. \( A_0 = \text{pass band gain}, \ f_c = \text{cut-off frequency.} \)

It is possible to establish a third order filter by combining a first order filter with a second order filter. In figure 3.13 this is illustrated. If this is an appropriate compromise of getting a sharp edge by the cut-off frequency without too much complexity.

![Third order butterworth filter consisting one first order filter and one second order filter.](image)

Figure 3.13: Third order butterworth filter consisting one first order filter and one second order filter.
Following calculation can be used to size the filter to the expected amplification and cut-off frequency.

\[ A_0 = \frac{R_2}{R_1 + R_2} \frac{R_5}{R_5 + R_6} V(IN) \]

\[ f_c = \frac{1}{2\pi\sqrt{R_3C_3}} = \frac{1}{2\pi\sqrt{R_6R_7C_2C_3}} \]

\( R_6 = R_7, C_2 = C_3 \)

### 3.5 Analog to Digital Converter

Audio processed by the acoustic sensor has to be converted into digital signals to be able to be processed by the microcontroller. A resolution of 24 bit is a requirement provided by FOI and is therefore described in this section.

The converter matches the incoming signal with the value of the 24 bit number that represents this voltage level. The maximum voltage represents the number \(2^{24} - 1 = 16,777,215\) and the minimum voltage represents the number 0. By this estimation, there will be a quantization error present, similar to the one explained in figure 3.3. Also timing of the measurement is of importance. If a signal level is registered at one time, and the time stamp is registered with some jitter, there will be a voltage error explained in figure 3.14.

![Figure 3.14: The errors of an ADC caused by a faulty time stamp. In this case, the sample frequency is 100 kHz, and one sample is delayed with 5 µs which results in a voltage error of 50 mV.](image)

To ensure that a ADC with a resolution of 24 bit is accurate to the last bit, the jitter of the oscillator can not be so large that the resulting voltage error exceed the resolution \(v_{24}\). The voltage error can therefore not exceed:

\[ v_{24} = \frac{V_{\text{max}}}{2^n} = \frac{5}{2^{24}} = 0.289 \mu V \]

Where \(V_{\text{max}}\) is the maximum input voltage of the ADC and \(n\) is the resolution. Since the filtering only allows frequencies below \(\frac{f_s}{2}\), the worst case scenario is a 24 kHz sinusoidal signal with an amplitude of 5 V, see figure 3.15. Combined with this, a sinusoidal signal has its maximum derivative where it crosses the time axis. In that point, the largest possible voltage errors occur when a constant time error is applied.
Figure 3.15: $V = 5\sin(2\pi \frac{f_s}{2} t)$, where $f_s = 48$ kHz. The red arrow marks one point where the sinusoidal signal has its maximum derivative.

With the sinusoidal curve in figure 3.15 the maximum allowed time error of the ADC clock ($t_{max}$) is calculated as following:

$$t_{max} = \left| 5\sin^{-1}\left(\frac{\pi}{2}\right) + 5\sin^{-1}\left(\frac{\pi}{2}\right) \right| = 22.64 \text{ ps}$$

The demands of the ADC clock jitter is therefore $\pm \frac{t_{max}}{2} = \pm 11.32 \text{ ps}$.

3.5.1 TLV320ADC3100

This ADC is a 2 channel 24-bit ADC which is intended for audio applications. It has internal filtering and a programmable amplification from 0 dB to 40 dB in steps of 0.5 dB which is an alternative to having external filtration and amplification.

In figure 3.16 the peripheral components to the ADC for proper use are shown.

Figure 3.16: The required external components to the TLV320ADC3100 in the most common use.
4 Software

Software for the settings and for the communication between components of this thesis is described in the following chapter.

4.1 WiFi

For this thesis, the data will be sent to a main computer over a WiFi network. WiFi is a technology that allows WiFi compatible devices to communicate over a wireless access point using radio signals. The most common use of WiFi is the use of a WiFi router that is connected to WiFi, by connecting WiFi devices such as a smart phone to the WiFi router then gives the smart phone access to Internet. See figure 4.1.

Figure 4.1: A WiFi router creates a path for the smart phone to access the Internet. The path between the smart phone and the WiFi router is connection-less.
A WiFi router however can allow devices to communicate over a more private type of Wireless Local Area Network (WLAN) without Internet connection, the only connection is the access point and the IP-address of the router which works as a address for the clients who wants to connect to the network. Data is transmitted to the access point and all clients can read it. See figure 4.2.

![Diagram of WiFi router connection](image)

Figure 4.2: The use of a WiFi router to establish a communication path between devices without the need of Internet connection. In this case, an Arduino, a smart phone and a computer have the ability to communicate over the network without being physically connected to each other.

### 4.1.1 User Datagram Protocol (UDP)

UDP is a protocol used to send data to other clients connected to the same WiFi network as the sender. However, it never establishes connection to the receiver which gives no guaranties that a receiver is there to interpret the data. This protocol ensures a fast way of transmitting data to a possible receiver but it is up to the receiver to manage eventual errors of the message, or missed messages.

When delivery speed is not of highest priority, a protocol called Transmission Control Protocol (TCP) can be used. TCP, compared to UDP establishes a connection to the receiver to ensure that no messages are lost.

### 4.2 GNSS module

GNSS modules sends their data serially over the TX and RX pin of the GNSS module. It sends all NMEA sentences every second and it is up to the microcontroller to extract the desired sentences or variables. In those cases where settings can be made to the GNSS module, for example the NEO-M8T, U-blox has a program for easy setup called U-center.

#### 4.2.1 U-center

U-center is a program made for testing and settings of the U-BLOX GNSS modules. It gives an overview of the number of satellites in view, the current position and the program can also be used to read the NMEA sentences. Figure 4.3 shows the main functions of the program.
4.3 ADC protocol

The settings of the standard ADC is done by using a so called Inter-Integrated-Circuit (I^2C) which is a protocol used to communicate between several devices. The audio is sent by a protocol called Inter-Integrated-Circuit Sound (I^2S) which is a protocol used for communication in audio applications.

4.3.1 Inter-Integrated-Circuit

I^2C is a protocol used to send and receive data between devices. It uses a two line bus that several devices can be connected to. There is one line for serial data (SDA) and one line for a serial clock (SCL) and devices connected to it can be either a master or a slave. Both masters and slaves can be transmitters and receivers of data, but the master is the device in control of the SCL. In other words, the master’s SCL pin is an output transmitted by the master and the slave uses its SCL pin as an input. All devices connected to the I^2C bus has individual address so that the transmitter can select which device to send data to. When no data is transmitted, both lines are held high. When the clock is high and the SDA is set to low, it is detected as a start condition. The data is transmitted according to figure 4.4.

Figure 4.4: Data transmission of the I^2C protocol. R/W stands for read/write, and ACK is an acknowledgement bit set by the receiver.
4.3.2 Inter-Integrated-Circuit Sound

The I²S protocol is used in a similar way as the I²C but is specified to audio applications. It has a bus containing 3 wires; a serial clock (SCK), a serial data (SD) and a word select (WS). The word select is implemented to be able to read from 2 channels and decide which channel that should be read at that moment. Just as in I²C, I²S devices can act as transmitters or receivers and masters or slaves. It is possible to use the transmitter as a master, the receiver as a master or a controller as a master, see figure 4.5.

![Possible I²S system configurations.](image)

I²S is mostly used as a type of serial communication from A/D or D/A converters, digital filters or signal processors. Microcontrollers that are meant to be compatible with audio projects are therefore equipped with the I²S protocol. [26]

4.4 Microcontroller

The microcontroller used for creating time stamps and sending the sensor data over a WiFi net is programmed in C++. Depending on the microcontroller of choice, different aiding libraries and functions are available for the different parts of the system. Both the Arduino MKR WiFi 1010 and the Raspberry Pi 3 Model B+ has support for UDP, GNSS communication, I²S and I²C.
The thesis work is separated into three main sections, pre-study, implementation and evaluation. How the different steps are executed is explained in the paragraphs below.

5.1 Pre-study

Firstly, a pre-study is made. The design alternatives provided by the client are presented below in figure 5.1, figure 5.2 and figure 5.3.

Figure 5.1: Block schematic of the first alternative solution. The VCO is connected to a counter, and with the help of the PPS of the GNSS module, and a microprocessor (Clock CTRL) to handle the clock, a GPSDO is created to ensure the frequency of the VCO. The Clock CTRL sends the time stamps to a second microcontroller that handles the ADC and sends the samples with the correct time stamp.
Figure 5.2: Block schematic of the second alternative solution. The Clock CTRL is implemented in the other microcontroller.

Figure 5.3: Block schematic of the third alternative solution. In this solution, all buffers are excluded and more of a sample based time stamp is implemented. The PPS of the GNSS module is also processed by the ADC and can than be found with its matching audio sample to achieve correctness. The microcontroller also reads how many samples are provided in a second to frequency lock the VCO.

Investigation and research considering time synchronization and alternative solutions results in aspects that have to be analyzed, excluded or investigated to reach one final solution. When a decision is made, the most important aspects of this function are listed and the list is
followed by pairwise comparison. The alternative that is better at most of the aspects is chosen. To simplify, all alternatives are not compared at once, but selections of main components are made first to sift away the unnecessary alternatives. Following tests are made:

- The use of PLL versus the use of the NEO-M8T timing module with respect to simplicity, adaptability, cost efficiency and accuracy over time.
- A comparison between the Raspberry Pi 3 and the Arduino MKR WiFi 1010 with respect to simplicity, internal ADC, internal clock speed, internal memory, cost and time accuracy.
- Updates of the design alternatives are evaluated with respect to simplicity, adaptability, number of components, capacity and time accuracy. The evaluation lead to further tests of parts of the designs.

5.2 Development of the design

Knowing which alternative is chosen, the block schematic is developed into a circuit that fulfill the wanted behavior. The microcontroller of choice is also programmed in this step. During this part, more questions are addressed and tested to make sure that the final product is as efficient as possible.

All components are set to produce the expected behavior using the software described in chapter 4. To ensure that the components and protocols work as desired, tests of the different parts of the system are made. These tests are performed to make sure that the settings are correct and applicable to a future assembled system. Further on, the microcontroller of choice is programmed to handle each of the following parts separately to discover eventual problems early.

- Sending of processed data using UDP.
- Reading the GNSS module NMEA sentence serially.
- Creating an accurate time stamp matched with a specific sample.
- Reading the ADC data using I²S.

When all parts are working correctly, or with limitations, a model is created that can be developed into a circuit in the future.
6 Results

In the following chapters, the results of this thesis research will be presented.

6.1 Pre-study

The pre-study aim was to analyze different design alternatives of the synchronization. Following chapters describe the different steps that lead to the decided solution. Firstly, an evaluation of two timing methods. After this an evaluation of different microcontrollers were made. Finally a decision of the design alternatives that is derived from the timing and microcontroller of choices.

6.1.1 Timing

Researching about GPS modules and how to establish a phase lock to a crystal oscillator, an alternative approach was found. Using a timing module from U-BLOX called NEO-M8T ensures a precise variable clock with a maximum accuracy error of 20 ns, and a maximum jitter of 11 ns. Because of this, the use of manual phase lock compared to the timing module NEO-M8T was evaluated according to table 6.1. The methods either get a "+" or a "−" depending on which alternative that best agrees with the description of that row.

Table 6.1: Table of evaluation of manual phase lock compared to NEO-M8T.

<table>
<thead>
<tr>
<th></th>
<th>Phase locked loop</th>
<th>NEO-M8T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplicity</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Adaptability</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Cost efficiency</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>Accuracy when lost GNSS signal</td>
<td>+</td>
<td>−</td>
</tr>
</tbody>
</table>

Summation

- The NEO-M8T eliminates the need for a phase locked loop by doing the same work internally. Minimal effort for implementing.

- The NEO-M8T can be programmed to adapt its frequency output to wanted sampling rate.
• The manually implemented phase locked loop is cost efficient, and leads to that a cheaper GNSS module can be purchased for approximately 10 $ instead of the NEO-M8T at approximately 50 $.

• The phase locked loops VCO will continue running at its frequency to its best ability. The NEO-M8T datasheet did not answer to the behavior of the TIMEPULSE when GNSS signal is lost.

Since the behavior of the NEO-M8T when GNSS signal is lost is unknown, the NEO-M8T was connected to a computer by a USB adapter. The module was connected to U-center (see chapter 4.2.1) and the settings in figure 6.1 was made. An oscilloscope was connected to the TIMEPULSE pin and the results are shown in figure 6.2.

Figure 6.1: The settings for the TIMEPULSE signal of the NEO-M8T. To the left there is a list of all items which can be adjusted, and to the right is the possible settings of the selected item from the list. The TIMEPULSE signal is active, with a frequency of 1 Hz, every pulse has the length of 0.1 seconds and the pulse is aligned to UTC time.
Figure 6.2: Actual TIMEPULSE output of the NEO-M8T with the selected settings. The pulses were measured after a few minutes without GNSS signal. All individual pulses were also measured to be precise.

The test showed that the NEO-M8T use its internal clock to continue producing the TIMEPULSE signals to its best capability. Since no time errors are measurable, the requirement of 22 ps is assumed to be fullfilled. A decision is therefore made to accept the limitation of possible long term errors when the GNSS signal is lost for this thesis due to the predominant advantages of the use of the NEO-M8T compared to a phase locked loop.

6.1.2 Microcontroller

Secondly, a comparison of two different microcontrollers were made, see table 6.2. The microcontrollers were picked for comparison because they are easy-to-use microcontrollers with different pros and cons that still both fulfill the basic needs for this project.

<table>
<thead>
<tr>
<th></th>
<th>Raspberry Pi 3</th>
<th>Arduino MRK WiFi 1010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplicity</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Internal ADC</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Internal clock speed</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>Internal memory</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>Cost</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Time accuracy</td>
<td>−</td>
<td>+</td>
</tr>
</tbody>
</table>

To summarize, the Arduino MRK WiFi 1010 is simpler to use, more time accurate and has an internal ADC. The Raspberry Pi 3 has a faster internal clock and more internal memory. The Arduino MRK WiFi 1010 was chosen due to it having a higher score than the Raspberry Pi and a new solution appeared, the use of the microcontroller’s internal ADC which will be investigated in the next chapter.

6.1.3 Design alternatives

Having decided to use the NEO-M8T timing module with the Arduino MRK WiFi 1010, the design alternatives were updated. The new alternatives were in no need of a phase locked loop, and with the Arduino MRK WiFi 1010, the communication can easily be implemented with an external WiFi router that the Arduino can be connected to. The updated alternatives are shown in figure 6.3 and 6.4. Another alternative that appeared was the possibility of using the internal ADC of the microcontroller. The alternative design is demonstrated in 6.5.
Figure 6.3: Block schematic of the first alternative solution using NEO-M8T. The time stamps are set in the CTRL block to unburden the Arduino so that it can focus on only matching the samples to the correct time stamp. The sample buffer can be internal or external.

Figure 6.4: Block schematic of the second alternative solution using NEO-M8T. The ADC converts both the audio from the sensor and the PPS from the NEO-M8T. Having this information provides the Arduino with the exact sample that should be matched to the start of every second. From this, the accuracy of the 48 kHz signal ensures that the timing for the other time stamps is correct.

Figure 6.5: Block schematic of the alternative solution using NEO-M8T and the internal ADC of the Arduino. The PPS is connected to an interrupt pin on the Arduino to ensure correct pairing with the time stamp and the samples. The 48 kHz is also connected to an interrupt pin to control the sample frequency.
Before evaluating these design alternatives, the delay of the interrupts and the time used by the Arduino to send data via WiFi had to be examined. If the delay of the interrupt is less than 3 µs and the sending time is < 0.1 s the third option will be compared with the other designs. The setup for this test of the interrupt is shown in figure 6.6.

![Figure 6.6: The setup for testing the delay of the interrupt.](image)

By using a program that loads the processor of the Arduino with different tasks and by measuring during a few minutes, the assumption is made that the interrupt delay is stable. Since the Arduino runs with a 48 MHz clock, this is equivalent with the processor taking 135 clock cycles to interrupt.

![Figure 6.7: The delay of the interrupt.](image)
The same setup was used for the test of the sending time, except for the CH1 on the oscilloscope that was disconnected. The idea of using the same setup is to be able to see the time it takes to send data including the delay of the interrupts since these will be used simultaneously.

![Figure 6.8: The sending time of 0.1 s of data.](image)

The test showed that the interrupt delay is stable at approximately 2.82 $\mu$s and the sending time of 0.1 s of data is approximately 60 ms (see figure 6.8) which leads to following evaluation of the design alternatives. The different design alternatives are rated from 1–3 depending on in which order they fulfill the description of that row. See table 6.3.

<table>
<thead>
<tr>
<th>1–3 points</th>
<th>Design 1</th>
<th>Design 2</th>
<th>Design 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Adaptability</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Number of components</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Use of the Arduino capacity</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Time accuracy</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Summation</td>
<td>7</td>
<td>12</td>
<td>11</td>
</tr>
</tbody>
</table>

The complexity to implement the different alternatives correlates to how much is implemented in software or not. The programs in the microcontrollers can always be adapted to specific needs, this implies that the third solution is the best. Adaptability refers to how easy it is to develop the solution in the future, having boundaries such as the fixed size of the ADC in the Arduino causes future work to be more difficult, hence the higher score for the second design. Designs with lesser number of components are smaller since the size of the final product decreases which leads to another score for the third alternative. When it comes to time accuracy, the uncertainties regarding multiple interrupts and how the timing is affected by it caused the second design to win that score.

According to table 6.3, the result of the pre-study of this thesis is design alternative 2, shown in figure 6.4. The final step of the pre-study was finding an ADC with internal adjustable filtering and amplification for audio context. The final block schedule is illustrated in figure 6.9.
6.2 Development of the design

Knowing which design that fulfilled the requirements most satisfyingly, the next step is to implement the separate parts and combining them into a working system.

6.2.1 NEO-M8T

Since the NEO-M8T was chosen, following functions are expected from the module:

- A PPS sent from TIMEPULSE with 1 Hz frequency.
- A clock signal from TIMEPULSE2 to establish 48 kHz sampling frequency in the ADC.
- Accessible NMEA sentences, which is included in the standard settings.

The PPS is set to be 10 ms long every start of a second to ensure that the ADC has enough samples to detect it. The settings are similar to the settings in Fig. 6.1, but with a pulse length of 10 000 µs.

To achieve a sampling frequency of 48 kHz, the TIMEPULSE2 has to be set to one of the values according to Table 6.4. The NEO-M8T has programmable output frequencies up to 10 MHz on TIMEPULSE2. Since these frequencies are divided from an internal clock, a test is made to guarantee that all of these frequencies are stable and jitter free.
All frequencies up to 8.192 MHz are unfortunately unstable. In figure 6.10 the 2.048 MHz
signal is shown which shows that the jitter is \( \pm 11 \) ns. Since this pulse is meant to be the clock
for the ADC, the jitter can maximally be \( \pm 11.32 \) ps (see chapter 3.5) to establish a signal with
accuracy to its least significant bit when 24 bits are used. A signal of 12 MHz was tested since
it is an even number and might match the internal clock of the NEO-M8T. The result was a
very stable signal, but not so square looking. See figure 6.11.

\[
\text{Expected period time: } T = \frac{1}{f} = \frac{1}{12 \times 10^6} = 83.333 \text{ ns}
\]

Figure 6.11: The period of the 12 MHz TIMEPULSE2 from the NEO-M8T. 8 pulses with 83.33
ns period equals 750.0 ns. The signal is measured both over several pulses and pulse by pulse
with accurate results.

The oscilloscope is not accurate enough to measure signals at such small levels as 11.62
ps, but since the signal showed no errors, the assumption is made that it is good enough. The
shape on the other hand is proven to depend on the band restriction of the measure probes.
They are tested at a reference square wave at 12 MHz with the same resulting shape as in
figure 6.11. Due to this restriction, the NEO-M8T 12 MHz signal is presumed to be square
formed.

6.2.2 TLV320ADC3100

The gain and filter of this ADC is programmed by setting different registers of the ADC by
I\(^2\)C communication according to the datasheet [27]. The ADC also has an internal PLL so it
can be locked to a specific sample frequency by changing the values of register P, R, J and D
according to table 6.4.

Table 6.4: Specifications of the TLV320ADC3100 PLL. MCLK is the input master clock and \( f_s \)
is the sample frequency.

<table>
<thead>
<tr>
<th>MCLK rate (MHz)</th>
<th>P</th>
<th>R</th>
<th>J</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_s = 48 \text{ kHz} )</td>
<td>1</td>
<td>1</td>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>2.048</td>
<td>1</td>
<td>1</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>4.096</td>
<td>1</td>
<td>1</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>6.144</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>8.192</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>12.0</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>1920</td>
</tr>
<tr>
<td>16.0</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1440</td>
</tr>
</tbody>
</table>

32
The filter of the ADC is set to a specific predetermined filter A (see figure 6.12) when the sample rate is 48 kHz. In all cases of PLL being used, following settings of registers must be made. They are different dividers of the signal and are more carefully described in [27].

- AOSR = 128
- NADC = 8
- MADC = 2

Figure 6.12: The specifications of filter A and amplification of the TLV320ADC3100. The red horizontal line indicates the level where the signal is defined as muted.

The program for the settings is available in appendix A.1 and follow a recommended order of settings from the datasheet [27]. The order of the settings are shown in figure 6.13. The program for this case set the MCLK to 12 MHz, meaning $P = 1$, $R = 1$, $J = 8$ and $D = 1290$ according to table 6.4. It also set the gain of the audio channel to 0 dB, and the gain of the channel meant for the PPS signal to -12 dB. The program also make the PLL available, initiates filter A and program the input signals. With these settings, the ADC produces its minimum value to its maximum value when the audio signal has an amplitude of 30 mV. The attenuation of the PPS channel derived from the voltage level. Since the output varies from 0 to its maximum with an amplitude of 30 mV and the PPS varies from 0 V to 3 V, -12 dB ensures that the output remains its value even if the PPS fluctuates a bit.

1. Define starting point:
   a. Power up applicable external hardware power supplies
   b. Set register page to 0
   c. Initiate SW reset
2. Program clock settings:
   a. Program PLL clock dividers $P$, $J$, $D$, and $R$
   b. Power up PLL
   c. Program and power up NADC
   d. Program and power up MADC
   e. Program OSR value
   f. Program I2S word length if required
   g. Program the processing block to be used
3. Program analog blocks:
   a. Set register page to 1
   b. Program MIBIAS, if applicable
   c. Program MICPGA
   d. Program routing of inputs and common mode to ADC input
   e. Unmute analog PGAs and set analog gain
4. Program ADC:
   a. Set register page to 0
   b. Power up ADC channel
   c. Unmute digital volume control and set gain

Figure 6.13: A recommended order of performing the settings of the TLV320ADC3100.
6.2.3 Arduino MKR WiFi 1010

The first part to be handled by the Arduino according to chapter 5.2 is the UDP send. This is done by using the UDP library of the Arduino which has the capability to connect to a WiFi network and send UDP messages the following program.

```c
#include <WiFiUdp.h>

char ssid[] = "xxx"; // network SSID (name)
char pass[] = "xxx"; // network password
int status = WL_IDLE_STATUS;
unsigned int localPort = 2390; // UDP port to write to
IPAddress ip = "xx.xx.xx.xx"; // IP address of the WiFi network
WiFiUDP Udp;

void setup()
{
    status = WiFi.begin(ssid, pass); // connects to WiFi
    Udp.begin(localPort); // initiates UDP
}

void loop()
{
    Udp.beginPacket(ip, localPort); // begin packet on intended port
    Udp.write("OUTPUT"); // message output
    Udp.endPacket();
}
```

The next part is reading the NMEA sentences of the NEO-M8T. The libraries NMEAGPS and GPSPort of the Arduino are used. Sentences are received on the RX pin of the Arduino and the correct sentence is extracted by only enabling the ZDA message in the NMEAGPS_cfg.h file in the NMEAGPS library. The time resolution has to be accurate with a resolution of a maximum of 0.01 ms due to the demand in the research question. With the ZDA setting and the resolution demand, the time is extracted by following program.

```c
#include <NMEAGPS.h>
#include <GPSPort.h>

NMEAGPS gps; // parses the GPS characters
gps_fix fix; // holds on to the latest values

void setup()
{
    Serial1.begin(9600); // Serial1 = RX(13) and TX(14)
}

void loop()
{
    while (gps.available( Serial1 )) // while there is a message to read
    {
        fix = gps.read();

        int h = fix.dateTime.hours; // hour (0..23) of last PPS
        int m = fix.dateTime.minutes; // minute (0..59) of last PPS
        int s = fix.dateTime.seconds; // second (0..59) of last PPS
        int micros = gps.micros(); // number of microseconds since last UTC second (~PPS)
    }
}
```
Developing this into an accurate time stamp is done by extracting the current time with microsecond resolution in the beginning of the program. The update of this time stamp is in the future planned to happen every 0.1 second by counting the samples of the ADC that is ensured to have accurate timing.

Finally, a program to read the data from the ADC is needed. The I2S library is used for this part. Following program is created for testing the function.

```c
#include <I2S.h>

int sample_left = 0;
int sample_right = 0;

void setup()
{
    //The settings of the ADC in appendix A.1
    Serial.begin(2000000);  // baud rate 2 000 000
    I2S.begin(I2S_PHILIPS_MODE, 24);  // initiates I2S, slave mode
                                        // (external clock)
}

void loop()
{
    if (I2S.available())  // if there are samples available
    {
        sample_right = I2S.read();  // read 24 bits
        sample_left = I2S.read();   // read next 24 bits
        Serial.println(sample_left);  // print samples from one channel
    }
}
```

The baud rate is set to the highest possible value of 2 000 000 bits/s which should be enough for a sample rate of up to 83 kHz. During this test however, the transmit buffer of the Serial.println() function seen in the code above is shown to be too slow to handle sample rates faster than 2 kHz. If the sample frequency is higher than this, the output swaps between the left and right channel and stops after a few seconds which makes debugging difficult.

To ensure that it is only the transmit buffer that causes the output to abort, a test is made. Instead of Serial.println(), a built in LED of the Arduino is set to blink if sample_left received a value. Without the transmit buffer, the LED never stops blinking. When the Serial.println() is added to the program again, the blinking stops after a few seconds. This ensures that it is the transmit buffer that is too slow for the sample rate regarding the aborted output.

The random swapping between channels is tested with a similar setup. The left channel is set to GND which outputs approximately 0, and the right channel to 3.3 V which outputs approximately $17 \times 10^6$ which is the maximum value with 24 bit resolution. The LED is programmed to blink if sample_left is lesser than $8 \times 10^6$ which is somewhere in the middle of these two values. Without the transmit buffer, the LED blinks constantly. With the Serial.println() added to the program, the LED alternates between blinking and not blinking.

6.2.4 Assembled model of the system

Derived from all previous results of the pre-study and the development, a proposed schematic of the wiring of the system was developed. All components are put onto a breadboard and connected according to figure 6.14.
Figure 6.14: A proposed schematic of the final design alternative. The NEO-M8T acts as the master of the I$^2$S communication between the Arduino and the ADC by setting TIMEPULSE2 to 12 MHz for the MCLK. The 12 MHz signal is also divided to 48 kHz to control the WCLK.

With this schematic, the Arduino can be programmed into the final system.
In this chapter a discussion of the results, the method that lead to the results and an ethical perspective is made.

7.1 GNSS vs PTP/NTP

In section 1.5, several methods to achieve synchronization is brought up. When time accuracy is of importance, using network protocols such as PTP or NTP and the use of a GNSS module’s PPS signal are the best options.

Using a network based method is beneficial in the way that it is a ready to implement method, already developed and defined. It is also very stable since the sensor nodes communicate with each other over the network to ensure that they are synchronized. This only requires one synchronizing device for a whole network, as long as this communication routes are available and is therefore a less expensive option to the individual time stamping using a GNSS module.

The GNSS method is beneficial over the network protocols since it is individual. If one or several sensor nodes are malfunctioning, the sensor nodes still operational can still run. Another benefit of having individual synchronization is that the amount of data traveling in the network is minimized due to the sensor nodes not having to communicate between each other. The minimization of data is positive since wireless communication is sensitive for external disturbance.

7.2 Pre-study

The pre-study overall gives no surprising results. Information that derived to the final design alternative in figure 6.9 is presented in the theory chapter. Since the pre-study of this thesis lays the ground to what system will be implemented in the future. It is particularly important to perform this step thoroughly to ensure a working system in the end.
7.2.1 Timing

One of the most important aspects of this thesis is the timing module since the demands of the future system are time accuracy based. Since both the PLL method and the NEO-M8T is proven to have high accuracy, the matter of simplicity and adaptability is the arguments for choosing the NEO-M8T over the PLL. The result derives from the fact that a form of GPSDO is already implemented in a single component. Therefore it is unnecessary to develop a circuit that solves something that some one else already has made. Another factor is that the NEO-M8T has software based settings, which makes it highly adaptable. To have adaptability in mind is mentioned in chapter 1.4 to increase the usability of the results in this thesis in other forms of applications with different sampling frequencies.

7.2.2 Microcontroller

When comparing the Arduino MKR WiFi 1010 to the Raspberry Pi 3 Model B+ they are both shown to be suitable for the system. The possibility of using the internal ADC of the Arduino is one of the reasons this is chosen. To establish if the internal ADC is an option, the interrupt has to be fast, accurate and the processor can not be over loaded.

Since the interrupt delay was longer than expected (see figure 6.7) and the UDP function of the microcontroller takes up more than half of the available time (see figure 6.8), the option of the internal ADC was dismissed. Since the internal ADC of the Arduino is not used, the evaluation of which microcontroller to use might not have the same outcome if it is done again.

7.2.3 Design alternatives

The result of the pre-study is a simple to adapt design with few components to solve the formulated problem. The resulting design comes naturally from what the theory says of the components. During the pre-study, several components are found that already solve parts of the final solution which results in only three components needed for the system. A microcontroller, a timing module and an ADC.

7.3 Development of the design

This part of the thesis has the aim to simplify the next step which will be implementation of the system. To achieve this, all settings of the components are tested to ensure that they perform according to the expected and desired characteristics.

The settings of the TIMEPULSE2 signal of the NEO-M8T was predicted to have less jitter for some frequencies than the result seen in figure 6.10 not the maximum jitter for all frequencies up to the maximum 8.192 MHz that is allowed by the ADC. This issue is fixed by trying the 12 MHz signal which seem to have no jitter measurable by the oscilloscope used. A square wave however was predicted which is not the result in figure 6.11. This is also explained by the limitations of the oscilloscope by measuring on a reference square wave. Due to the limitation of the measurement tool, assumptions are made that the signal is both jitter free and square formed.

When programming the ADC, nothing in the results stand out. The resulting outcome of the settings however is difficult to measure. Filter A for example, shown in figure 6.12 is assumed to function according to the specifications but are not measurable in the framework of this thesis. Different possibilities of gain and volume control are not investigated more carefully since the outcome of the settings in appendix A.1 operates in 30 mV, similar to a
Developing the Arduino and its different functions, both the UDP library and the NMEAGPS library work as expected. On the other hand, none of these functions were validated. No UDP receiver has been available and implementation of this would be too time consuming for this thesis.

Instead, the assumption that it works has been made since the time consumed for the UDP.send() has varied with different lengths of the message which is sent. Further on, in the NMEAGPS library, the function gps.micros() returns a value, but no measurement on how accurate this value is has been made.

New problem formulations are discovered when testing the I2S library of the Arduino. The transmit buffer limits the possibility for debugging and validation of the system at its intended sample frequency. To be able to evaluate the system, a UDP receiver is necessary. This is expected to work properly since the UDP has been proven to handle interrupts in chapter 6.1.3.

7.4 Feasibility

This thesis provides a model that facilitates an already existing system by removing the need for wired communication. The economical benefits for this implementation are not easily measured since the use of this thesis extends further than to the context of which it is put in. Through having adaptability and simplicity in mind during the pre-study, the user areas of this study have been kept wide.

In another perspective, the value of this thesis has its main perspective in usability. This since the network in this context can cover large areas with kilometer long cables if wireless synchronization is not implemented. With this in mind, this thesis has provided a highly feasible model of a system.

7.5 Method

Methodological flaws are present in most studies, this thesis included. One of these flaws is being demonstrated three possible solutions to the problem which constrains the creativity that might lead to new perspectives. Even if a broad pre-study with an open mind minimizes the risks of missing vital perspectives, there could be solutions that are never found.

While performing tests, it is found that the oscilloscope is not accurate enough to measure time errors of pico-level. If the tests would be made again with more accurate measurement equipment, it might not lead to the same results since some of the results are assumed. In addition to this, the evaluation method is not very scientific. To determine which alternative is the simplest could be seen as a question of interpretation. This means that someone else might perceive it in another way with a different outcome.

In this thesis, different types of sources are used. Finding sources in this area is not an issue since it is well studied in the past. For descriptions of component’s specifications, datasheets and product descriptions of the brand’s web page are used. This information is most likely accurate. For different methods and models, secondary sources such as previous thesis reports and scientific literature are used to demonstrate what previous results show and how it can be used for this case. The sources used are relevant and reliable.
sources are web pages which might have biased information which have been considered and the information has been determined to be accurate.

7.6 The work in a wider context

Research regarding technology meant for military use raises many ethical questions. In [28], a broad discussion about ethical issues with military research is made with different perspectives. For instance can the discussion have both a deontological and a consequential perspective. Deontology means that the focus should be on whether the action itself is morally questionable. Consequentialism on the other hand focuses on the consequence of the action.

From a deontological point of view, military research is no ethical issue. Technology is developed, and there is no difference in what it will be applied to later in the process. Researching for industry or military is no difference if the research itself is harmless.

If a consequential perspective is applied, there is more to discuss. This particularly context where snipers are being localized by the use of this system can be seen as a way of eliminating enemies. This perspective leads the discussion to a sort of trading issue. What will the results be if the sniper is not eliminated? The system could in this case rather be seen as a way for the military to defend themselves which removes the direct ethical dilemma of this thesis.

In addition to this, the thesis itself can be used for a wide range of application. Synchronized sensors are a requirement for both medical, industrial and civilian use.
This chapter summarizes the purpose of this thesis and how the results answers to the research questions in chapter 1.3. It also describes what work could be done in the future to develop the results of this thesis.

8.1 Summation

Developing a system that in the future can be implemented into a synchronizing device for acoustic sensor nodes in a wireless sensor network is the aim of this thesis. An independent time-stamping device is developed to the point where only implementation and evaluation remain. This indicates that the aim of this thesis is fulfilled.

The research questions of this thesis are as following:

1. What are the alternatives to achieve synchronized time stamps of wireless acoustic sensors using GNSS?

2. Which alternative guaranties a maximum time error of 0.01 milliseconds compared to Coordinated Universal Time (UTC)?

3. Is the proposed solving method feasible?

The alternatives to achieve synchronized time stamps using GNSS are presented during the course of this thesis. Some sort of timing module, either with a manually implemented PLL or with an already implemented timing module called NEO-M8T. A microcontroller is also needed to process the sensor data and for creating the time stamps, this could be a microcontroller of any choice with sufficient memory, clock speed and WiFi, I²S and I²C support. The last component is the ADC and filtering of the audio. This can be two separate parts, or a combination in the component TLV320ADC3100.

All of the proposed alternatives can be developed to guarantee a maximum time error of 0.01 milliseconds. Combined with the extra perspectives of work load, size, flexibility and cost, a design with the NEO-M8T, an Arduino MKR WiFi 1010 and the TLV320ADC3100 is the best solution for this application. The design is shown in figure 6.9.
Regarding the final research question, the proposed solving method is feasible. Adaptability and simplicity has been in focus of the pre-study which has lead to the system being worth to continue with in the future.

8.2 Future work

This thesis has provided all of the separate parts of the system, but not combined. In the future, the Arduino should be programmed with the combination of all parts tested during this thesis. When this is done, the system has to be evaluated to ensure that it provides accurate samples matched with the correct timestamp. One option for validation is to create a test environment that replicates the acoustic phenomena from a sniper to see if the system can be used for this application. The master computer of the WSN has to implement a UDP receiver where it received data over wired communication before.
A Appendices

A.1 Settings for the ADC

```cpp
#include <Wire.h>

void setup()
{
    pinMode(5, OUTPUT); // sets the digital pin 5 as output for RSTN

digitalWrite(5, LOW); // reset the ADC
delay(100); // wait for 100 ms
digitalWrite(5, HIGH); // set reset pin high

Wire.begin(); // join i2c bus
delay(100); // wait to ensure i2c connection

////////////////////////////////////////////////////////////////////////////////////////////
// ------------------------- 1 ----------------------------//
////////////////////////////////////////////////////////////////////////////////////////////
// 1b. Register page 0
Wire.beginTransmission(24); // transmit (write) to device #24 (ADC)
Wire.write(0x00); // register 0
Wire.write(0x00); // send one byte (0000 0000)
Wire.endTransmission(); // stop transmitting

////////////////////////////////////////////////////////////////////////////////////////////
// ------------------------- 2 -----------------------------//
////////////////////////////////////////////////////////////////////////////////////////////
// 1c. Initiate SW reset
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x01); // register 1
Wire.write(0x01); // send one byte (0000 0001)
Wire.endTransmission(); // stop transmitting

////////////////////////////////////////////////////////////////////////////////////////////
//------------------------- 2 ------------------------------//
////////////////////////////////////////////////////////////////////////////////////////////
```
// 2a. Program P, J, D, R PLL clock dividers
// PLL_CLKIN = MCLK, CODEC = PLL_CLKIN
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x04); // register 4
Wire.write(0x03); // send one byte (0000 0011)
Wire.endTransmission(); // stop transmitting
// P = 1, R = 1, PLL Powered down
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x05); // register 5
Wire.write(0x11); // send one byte (0001 0001)
Wire.endTransmission(); // stop transmitting
// J = 8
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x06); // register 6
Wire.write(0x08); // send one byte (0000 1000)
Wire.endTransmission(); // stop transmitting
// D = 1920 = 0111 1000 0000, bit 8:13 = 3
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x07); // register 7
Wire.write(0x03); // send one byte (0000 0011)
Wire.endTransmission(); // stop transmitting
// D bit 0:7 = 1000 0000 = 128 = 0x80
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x08); // register 8
Wire.write(0x80); // send one byte (0000 1000)
Wire.endTransmission(); // stop transmitting

// 2b. Power up PLL, P=1, R=1
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x05); // register 5
Wire.write(0x91); // send one byte (1001 0001)
Wire.endTransmission(); // stop transmitting

// 2c. Program and power up NADC = 8
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x12); // register 18
Wire.write(0x88); // send one byte (1000 1000)
Wire.endTransmission(); // stop transmitting

// 2d. Program and power up MADC = 2
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x13); // register 19
Wire.write(0x82); // send one byte (1000 0010)
Wire.endTransmission(); // stop transmitting

// 2e. Program OSR value = 128
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x14); // register 20
Wire.write(0x80); // send one byte (1000 0000)
Wire.endTransmission(); // stop transmitting
2f. Program I2S word length = 24 bits, I2S mode, BCLK output, WCLK input
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x1B); // register 27
Wire.write(0x38); // send one byte (0010 1000)
Wire.endTransmission(); // stop transmitting
// Enable BCLK_DIV to output BCLK from MCLK
Wire.beginTransmission(24); // transmit(write) to device #24
Wire.write(0x1E); // register 30
Wire.write(0x80); // sends one byte (1000 0000)
Wire.endTransmission(); // stop transmitting

2g. Program the processing block to be used (PRB_R1 = Filter A, no biquad filters)
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x3D); // register 61
Wire.write(0x01); // send one byte (0000 0001)
Wire.endTransmission(); // stop transmitting

3a. Set register page to 1
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x00); // register 0
Wire.write(0x01); // send one byte (0000 0001)
Wire.endTransmission(); // stop transmitting

3b. Program MICBIAS = not used
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x33); // register 51
Wire.write(0x00); // send one byte (0000 0000)
Wire.endTransmission(); // stop transmitting

3c. Program MICPGA
//Left settings = 0 dB (0..40 in 0.5 dB steps)
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x3B); // register 59
Wire.write(0x00); // send one byte (0000 0000)
Wire.endTransmission(); // stop transmitting
//Right settings = 0 dB
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x3C); // register 60
Wire.write(0x00); // send one byte (0000 0000)
Wire.endTransmission(); // stop transmitting

3e. Program routing of inputs and common mode to ADC input
//Left ADC input selection (differential), Microphone input
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x34); // register 52
Wire.write(0x7F); // send one byte (0111 1111)
Wire.endTransmission(); // stop transmitting
//Right ADC input single-ended, PPS input
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x37); // register 55
Wire.write(0xCF); // send one byte (1100 1111)
Wire.endTransmission(); // stop transmitting

// 4a. Set register page to 0
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x00); // register 0
Wire.write(0x00); // send one byte (0000 0000)
Wire.endTransmission(); // stop transmitting

// 4b. Power up ADC left and right channel
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x51); // register 81
Wire.write(0xC2); // send one byte (1100 0010)
Wire.endTransmission(); // stop transmitting

// 4c. Unmute digital volume control and set gain
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x52); // register 82
Wire.write(0x00); // send one byte (0000 0000)
Wire.endTransmission(); // stop transmitting

// Volume left channel = 0 dB
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x53); // register 83
Wire.write(0x00); // send one byte (0000 0000)
Wire.endTransmission(); // stop transmitting

// Volume right channel = -12 dB
Wire.beginTransmission(24); // transmit (write) to device #24
Wire.write(0x54); // register 84
Wire.write(0x68); // send one byte (0110 1000)
Wire.endTransmission(); // stop transmitting
Wire.end();
}
Bibliography


