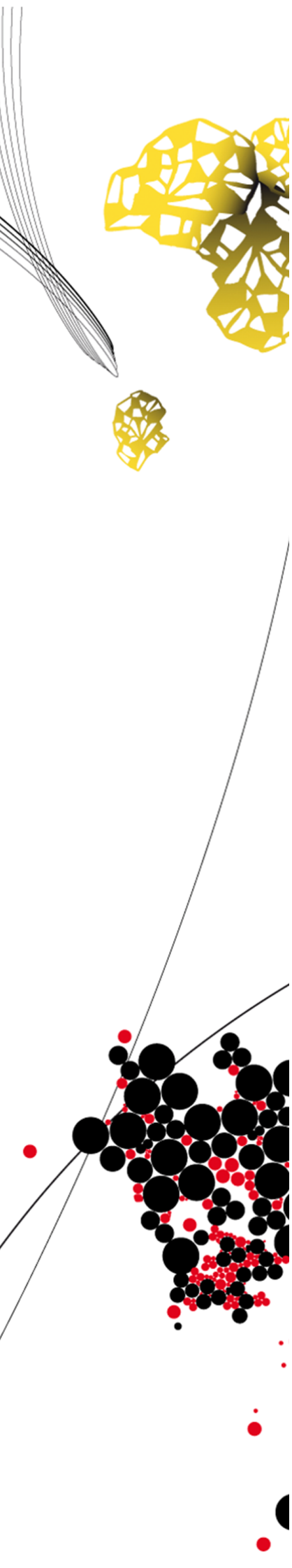


UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics & Computer Science

Implementation of a Reliable 2D Approximator Using Sonar

Remi Jonkman (s1563599)

B.Sc. Thesis

July 2017

Supervisors:
H. Kerkhoff
H. Scholten

Computer Architecture for Embedded Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

Standard imminent collision detection systems use either laser or radar or both to function. Although this works very well, considerable costs are involved, such that these systems are only incorporated in expensive cars. A cheaper solution to detect an imminent collision has already been proposed and uses four anisotropic magnetoresistance (AMR) sensors in combination with an approximator consisting of three sonar sensors [1]. The implementation of a reliable approximator running on a PLASMA central processing unit (CPU) with a standard deviation less than 7 centimeters will be the core of this research. Reliability also increases the demands on other aspects such as software and hardware. Especially the sonar sensors should have a predictable behavior during their entire lifespan. For this reason, part of this research also focuses on determining the behavior of sonar sensors by means of an electrical and temperature stress test.

The sonar sensor that is used in the design endures a series of stress tests. There are weekly experiments with a test setup to gather data that is nothing more than a series of approximated distances. These distances are used to determine the standard deviation compared to the actual distance. The standard deviation did not show any change in sensor behavior. It however showed unreliability of the sensor at nonzero angles.

Two sonar sensors are used as a receiver and one as a transmitter in the approximator. This method enables us to approximate the angle and distance of an object. The code running on the PLASMA CPU is written as a finite state machine (FSM) with various error checks and captures in order to keep the system running. Although results showed the error margins were below the required standard deviations, the range could be improved when using other sonar senors.

Abstract

Sensors in cars can be used to detect an imminent collision. Where expensive cars are usually equipped with laser or radar technologies, cheap cars lack these technologies. A cheap but reliable solution to detect collisions is examined in this report. The solution includes four AMR sensors and three sonar sensors in combination with a fault tolerant network of four PLASMA CPUs and will be investigated in project IMMORTAL. It consists of different parts with one being the central topic of this report, namely an approximator. Sonar sensors are part of an approximator and it is important to understand their degradation. An endurance test with a weekly data capture on a specified test setup, showed that the standard deviation remains the same after all stress tests. When included in the approximator, the resulting distance and angle of an object could be approximated closely when the object is in range.

Contents

List of acronyms

AMR	anisotropic magnetoresistance
ASIC	application specific integrated circuit
CPU	central processing unit
FPGA	field programmable gate array
FPU	floating point unit
FSM	finite state machine
NOC	network on chip
PZT	lead zirconate titanate
RISC	reduced instruction set computer
UART	universal asynchronous receiver or transmitter
VHDL	VHSIC (very high speed integrated circuit) hardware description language

Chapter 1

Introduction

1.1 Motivation

The implementation of a reliable position estimator presented here is a direct consequence of a previous work of Taghvaeeyan [1]. He presented a way to estimate the movement of an object towards a car with only AMR sensors and sonar.

1.2 Framework

The research was performed at the University of Twente as a bachelor assignment. It has been carried out in collaboration with the Computer Architecture for Embedded Systems group and is part of project IMMORTAL (Integrated Modelling, Fault Management, Verification and Reliable Design Environment for Cyber-Physical Systems). All research presented here will be used as a starting point to create an application specific integrated circuit (ASIC) to detect an imminent collision with multiple PLASMA CPUs, AMR sensors and the proposed approximator in this document.

1.3 Research question

Eventually, when this estimator will be part of an imminent collision detection system, safety will be very important, hence all research questions focus on reliability.

1. How to implement a 2D position approximator using sonar and a PLASMA CPU with standard deviations smaller than 7 centimeters and 5°?
2. Do HC SR04 sensors degrade when stressed with either overvoltage or extreme temperatures?

1.4 Report organization

The document is organized according to the workflow. When reading this, you have already found the introduction. Chapter two, Analysis, will inform the reader about all parts that were involved in this research. Possible problems and background information are given. Chapter three will explain the code that runs on the PLASMA CPU. The next chapter, Execution, will describe test setups and how the tests were performed. Chapter five will show the results of experiments. The results are followed by conclusions and a discussion in chapter six. The last part of this document includes a bibliography and a few appendices.

Chapter 2

Analysis

The purpose of this chapter is to define the system and analyze the components that will be used. Important information about components will be highlighted whilst also keeping an eye for potential problems.

2.1 System layout

Eventually, there should be a system consisting of four PLASMA cores that are connected via a fault tolerant network on chip (NOC). Code running on these cores will manage four AMR sensors and an approximator consisting of three sonar sensors. It will be impossible to implement everything with only a limited amount of time, hence this report focuses on the approximator.

2.2 Sonar Sensor

Inter-Vehicle distances can be measured in different ways, where every method has its advantages and disadvantages. As already stated in the introduction, this document will follow the method provided in [1] and as such various sonar sensors are combined in an approximator.

2.2.1 Final Sensor

Industry offers a lot of different sonar sensors, but they are mostly divided into two categories, namely open or closed sensors. Open sensors usually have a much larger range than closed sensors. However, they can not be used in wet conditions where closed sensors are preferred.

The sensor in this prototype will be the HC SR04 that can be bought in a price range between 0.80$ and 2$ per sensor. The average quality and hence the relia-

bility of these sensors is nonetheless questionable. For this reason, a part of this research stresses these sensors to check their behavior. In a real life application, the HC SR04 alone is not sufficient, since it is an open type sensor, in a prototype it will sufficient.

2.2.2 Operation

Sonar sensors operate in the inaudible range of the human ear, that is sound waves with a frequency above 20 kHz. The HC-SR04 operates at 40 kHz. Figure 2.1 shows the appropriate timing diagram. The trigger pin should be high for at least 10 microseconds, which has been tested. Shorter time intervals give unpredictable results. Sometimes your sensor will return a value or it will not even notice your trigger and just stay idle. An instance after the falling edge of the trigger, the HC-SR04 emits 8 bursts of sound waves and when reflections of the sound waves return, the echo pin will go high. Eventually, the distance will be as in equation 2.1.

$$distance_{cm} = \frac{(t_{falling} - t_{rising})}{c} \tag{2.1}$$

where,

1. $t_{falling}$[s] is the time instant of the falling edge of the echo pulse;
2. t_{rising}[s] is the time instant of the rising edge of the echo pulse;
3. c is a constant that is derived from the speed of sound ($343\frac{m}{s}$). Normally $c = 58.31\frac{\mu s}{cm} \approx 58\frac{\mu s}{cm}$

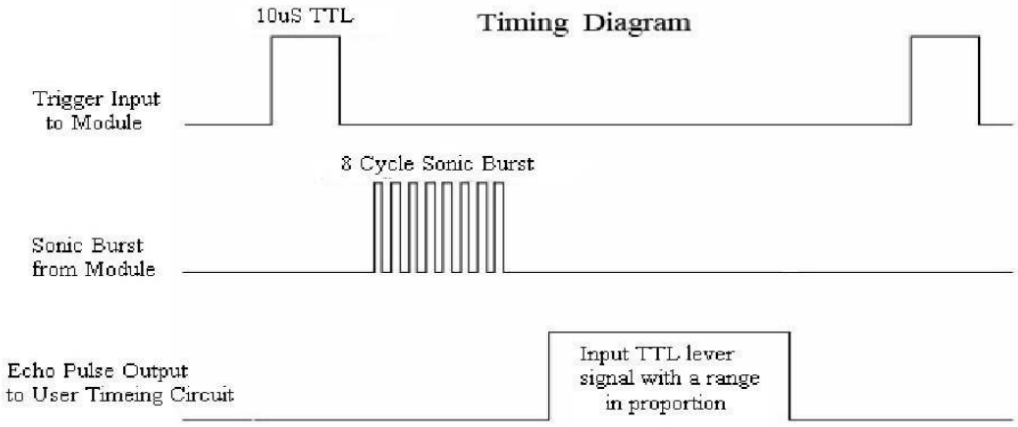


Figure 2.1: HC SR04 Timing Diagram

2.2.3 Interference

In all audio applications interference is a well known behavior when multiple sources emit sound. Although the proposed setup only uses one source, there is still a chance of interference. Sound waves reflect from anything that is near the sensor and as such interference patterns are introduced. Generally, this does not pose a problem, but when the sonar sensors are probed at very high speeds, this might give unpredictable results due to this phenomenon. The easiest solution is to wait a few microseconds after reading the sensor, then the ultrasonic waves will fade out. Another solution, that has not been investigated further, is to use interference algorithms.

2.2.4 Piezo-Electric Transducer

The HC SR04 module uses two piezoelectric transducers, figure 2.2 shows such a transducer. Piezoelectric materials have a reversible effect between their mechanical and electrical state. When a voltage is applied across the faces of such a material it will result in the deformation of the material. In case of an acoustic sensor rapid switching of voltages causes changing pressure levels, in other words sound. Another transducer could receive these pressure levels, since there will be a potential difference.

Figure 2.2: Sonar Transducer Element

Internals of the sonar transducer element are made of lead zirconate titanate (PZT). As a consequence the transducer might also be used as temperature sensor.

2.3 Endurance Test

Car sensors demand high reliability. Safety sensors, as a collision detection system, are never allowed to stop working. Unfortunately, at some point during production, there will be cases where sensors stop functioning properly, so to ensure this will not injure the occupant, a proper error management system should be in place. Even more so, false readings because of material degradation inside the sensors are not desirable either. Degradation causes sensors to behave differently in time.

Stressing sensors with an endurance test will show this changing behavior, such that degradation can be approximated and control systems can deal with this.

2.3.1 Test Procedure

The endurance test stresses two sonar sensor populations with different means. The first population consists of all transducer elements of four senors. These were stressed with a week long interval test. A unique test segment takes one hour, where the temperature inside the chamber is approximately 125 ℃ for 45 minutes then there is a cooling down period to 30 ℃ and afterwards the temperature goes back to 125 ℃. The transducer elements have to be desoldered and re-soldered to their original circuit boards every time a data set will be taken.

The second population consists of only two sensors. These sensors are stressed with a 20% overvoltage across their positive and negative terminal. Their nominal voltage is 5V, hence the potential difference across the terminals is 6V. This test will thus apply an overvoltage to the entire sensor including its circuit board, so nothing has to be desoldered.

2.3.2 Test Setup

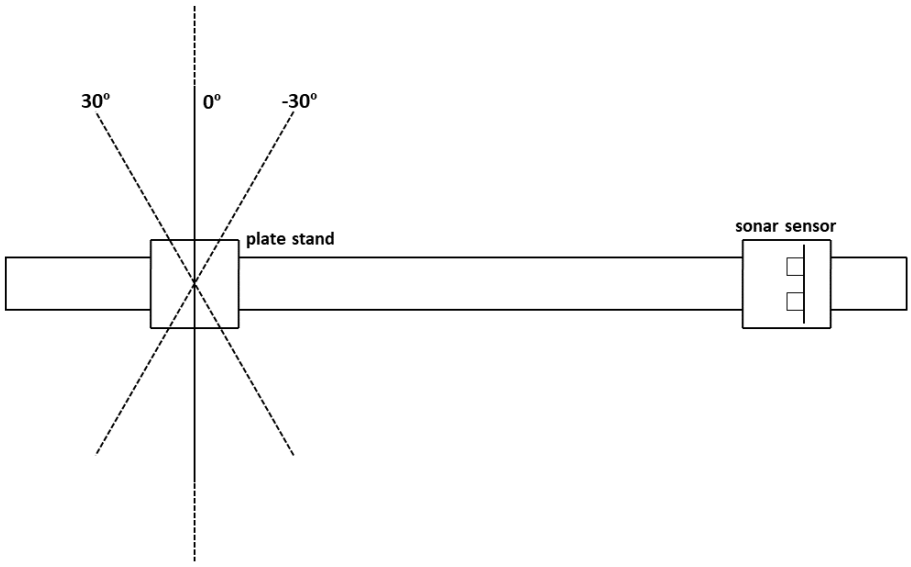


Figure 2.3: Test Setup

Figure 2.3 depicts the setup is used to examine the sonar sensors. On the right is a sonar sensor on a fixed stand. The sonar sensor can easily be swapped and is

connected to a microcontroller. In the middle is an aluminum plate that can move to the left and right. On the wooden rail is a centimeter scale to measure the actual distance. Additionally the moving stand can be swapped to use either a 30° stand or -30° stand.

2.4 PLASMA CPU

A 32-bit reduced instruction set computer (RISC) processor, known as PLASMA [2], will be used as microcontroller. Steve Roads originally created the PLASMA core and has declared it open source. Its sources have henceforth been cloned and will be used in project IMMORTAL [3]. PLASMA code consists of VHSIC (very high speed integrated circuit) hardware description language (VHDL) and C code, where the VHDL code will run on a field programmable gate array (FPGA). The actual PLASMA CPU has not changed in project IMMORTAL, although the universal asynchronous receiver or transmitter (UART) system was updated. Besides, the IMMORTAL ASIC will use four PLASMA cores with a fault tolerant NOC. Since this project is part of IMMORTAL, adapted PLASMA sources [4] will be used.

2.4.1 Advantages

There are a lot of different CPU designs available. The PLASMA CPU has generally a few advantages compared to other architectures. First of all, it is open source and can henceforth be edited and used freely, while you are only obliged to mention the original author. Moreover the architecture is implemented with VHDL, such that there is a lot of flexibility in adding your own function dependent peripherals. The MIPS architecture will also result in a relatively small core size compared to other processor types due to its RISC nature. The last advantage is very useful from a developers point of view. The MIPS architecture has toolchains available on all operating systems, hence it is rather easy to compile the appropriate sources.

2.4.2 Disadvantages

For now, the PLASMA CPU does not incorporate a floating point unit, hence all floating point calculations will be done in software. As a consequence system performance decreases drastically compared to hard float devices when complex calculations have to be carried out. Secondly the **MIPS!** (**MIPS!**) architecture dictates a single interrupt pin, such that only one device at the time can be monitored through interrupts.

2.5 Field Programmable Gate Array

The PLASMA core will be synthesized on an Cyclone V FPGA. The Cyclone V is embedded on an Altera DE1 SOC with two times 40 GPIO pins. This is generally enough for one PLASMA core when all GPIO inputs and outputs are mapped on them.

2.6 Position Estimation

Taghvaeeyan proposed a way to implement position estimation in 2D using multiple sonar sensors in [1]. That method uses two sonar receivers and a single transmitter and is also used in this research. The calculations require arcsine, powers, cosine, subtraction, addition, division and multiplication to be implemented in a floating point math library for the PLASMA CPU.

Chapter 3

Implementation

All sensors will be connected to a FPGA running the PLASMA CPU. The methods summarized in the previous chapter will be implemented in code, which will be dissected and explained below.

3.1 PLASMA Code

The application running on the PLASMA is implemented as an FSM machine (see figure 3.1). A FSM works very well for repeating behavior, such as continuous sensor probing and calculations. The state variable q_x where $x \in \{a, b, c, d, e, f, g\}$ represents all states that are currently programmed. Their state properties and transitions will be explained in the next sections.

3.1.1 Init: q_a

Also known as the initial state, the program always starts here. Init resets all sensor specific variables with one exception, the sensor identifier itself. It also sets both multiplexers (see appendix B), such that the correct echo pulses are received and the correct trigger will be sent. After preparation the trigger pin will go high and a timer (t_c) will be set. Then there will be a transition to the next state.

3.1.2 Trigger High: q_b

Within this state, only the trigger pin will go low when the timer finishes. Afterwards, the external interrupt pin will be configured to wait for an echo signal.

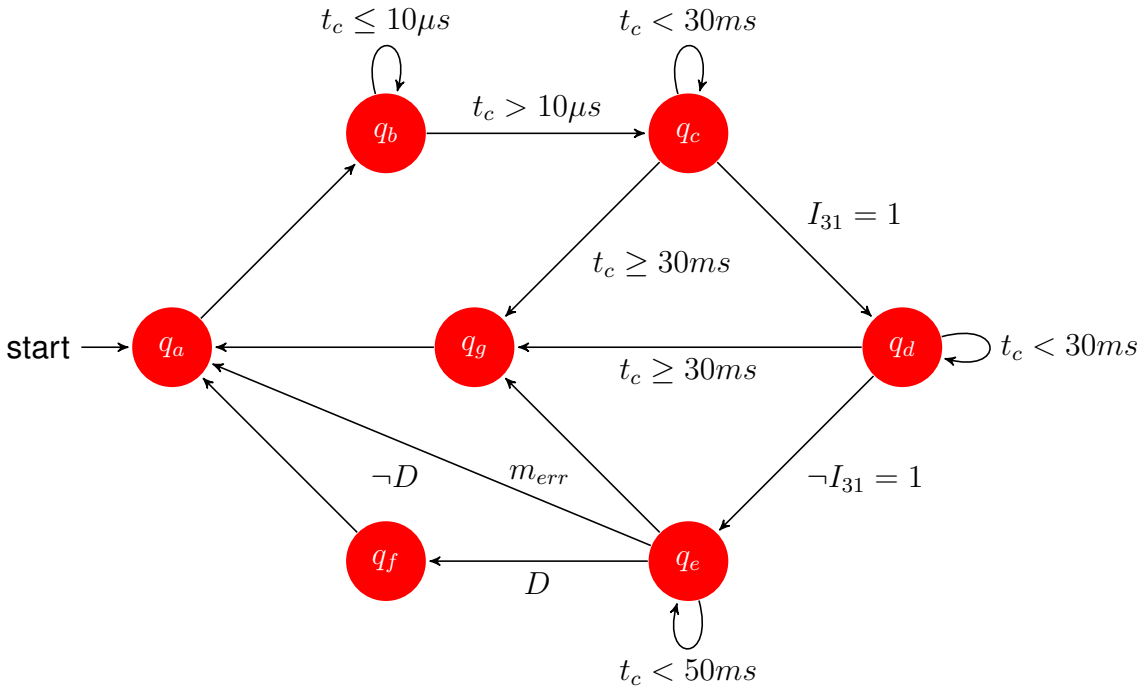


Figure 3.1: State Diagram Representing Software Behavior

3.1.3 Echo Wait: q_c

Whilst awaiting a response, a timer signal is looping to ensure the program will not freeze when there is no response. If a timeout occurs, the program will go to its error state to check for the error and reconfigure if necessary. Without errors, the signal state will change to q_d.

3.1.4 Echo High: q_d

When the external interrupt pin, GPIO0_31 (hence I_{31}), is triggered, this state becomes valid. Whenever the sensor stops working properly, a timeout ensures the system will not stall in this state. The FSM will wait for the interrupt pin to go low again, after which it will start the calculations.

3.1.5 Calculate: q_e

Although the name suggest this state only performs calculations, this is not entirely true. The current sensor configuration requires two program cycles till both sensors are read, only then calculations can be carried out. If, during the calculations, an error occurs (m_{err}), the error state will be triggered. To avoid interference, a timer is set to allow the state to transition when the calculations are done and the timer has finished. At last, the sensor identifier will be updated to the next sensor or it will be

reset. The program can then go to the display state or the initial state, depending on the debug setting ($D \vee \neg D$) of the system.

3.1.6 Display: q_f

The display state does what the name suggests, it shows the results of calculations. Date monitoring is optional and can be omitted.

3.1.7 Error: q_g

All states causing an error, set an environment variable that contains the actual error message that occurred. That message will be displayed and then there will be some reconfiguration before going to the initial state. Reconfiguration ensures continuous performance.

3.2 Math Library

For now, the PLASMA CPU cannot use standard C math libraries. All of them fail at some point to emulate soft float behavior. As such, a math library had already been written, but in line with this project, the library required some extension.

Every two program cycles, the arcsine function is required to complete the state. Typically, implementations approximate the original function, but with some trade-offs. As the PLASMA is not the fastest CPU, especially without floating point unit (FPU), an approximation(eq. 3.1) [5] with a relatively large error at $|x| \approx 1$ is chosen. This part of the domain is not used during the calculations, since the sonar sensors are incapable of handling such large angles.

$$arcsin(x) = \frac{x\pi}{2} \frac{a_0 + a_2x^2 + a_4x^4 + a_6x^6}{b_0 + b_2x^2 + b_4x^4 + b_6x^6} \tag{3.1}$$

where

$$\begin{aligned} a_0 &= 0.318309886 & b_0 &= 0.5 \\ a_2 &= -0.5182875 & b_2 &= -0.89745875 \\ a_4 &= 0.222375 & b_4 &= 0.46138125 \\ a_6 &= -0.016850156 & b_6 &= -0.058377188 \end{aligned} \tag{3.2}$$

Chapter 4

Execution

The following sections describe the execution of the endurance test and will introduce several difficulties that arose during the tests.

4.1 Stress Test Methodology

The setup described in section 2.3.2 was used, together with a Vötsch VT4004 temperature chamber and a Delta dual power supply E018-0.6D. Six entirely new HC-SR04 sensors were unpacked and marked in order to uniquely identify them. The sensors were then split and put into two different groups. Unfortunately, the initial idea to capture five datasets changed due to a faulty test setup. Looking back to figure 2.3, the sonar sensor is placed too low, such that the sound waves will reflect from the plate stand and not from the aluminum plate. In the case were the sensor and plate are perpendicular to each other, this will not invalidate all data, since the actual distance can still be calculated by means of the Pythagorean theorem. All other data is faulty, since the angle does not matter anymore and the reflected waves come from a perpendicular plane, such that all three angles result in exactly the same data. Increasing the height of the sonar sensor relative to the rail will give the appropriate results. The initial idea was still completed till the fifth data capture, but in parallel there were three other data points captured from the fourth data point till the last, to account for the faulty test setup.

4.1.1 Temperature Stress Test

All four sensors in this group have been following the schedule in table 4.1. There is an anomaly in the first week, where the minimum temperature is only 50 °C. Originally a program took an hour to complete and looped then back to the beginning. It consisted of a five minute cooldown to 30 °C and then 5 minutes to get it to 125 °C

Date	Time		Temperature (℃)
Initial Data Capture			
05/30	16:12 PM	Minimum	50
06/06	11:30 AM	Maximum	125
Second Data Capture			
06/06	02:08 PM	Minimum	30
06/13	09:17 AM	Maximum	125
Third Data Capture			
06/13	02:14 PM	Minimum	30
06/20	09:23 AM	Maximum	125
Fourth Data Capture			
06/20	03:32 PM	Minimum	30
06/27	10:41 AM	Maximum	125
Fifth Data Capture			
06/27	17:01 PM	Minimum	30
07/04	12:10 PM	Maximum	125
Sixth Data Capture			

Table 4.1: Temperature stress test schedule

and 50 minutes of a constant temperature at 125 ℃. In the original program, a five minute cooling down period was too short, hence from the second week onwards, the program was changed to incorporate a ten minute cooling down period and the constant temperature segment became only 45 minutes.

4.1.2 Voltage Stress Test

This group, consisting of only two sensors, followed the schedule as in table 4.2. Again, there were only 4 weeks required, however, the fourth data capture became useless, since someone turned off the voltage source. The last week is therefore acquired to account for this lost week. The test setup stood in a laboratory with a nearly constant temperature. Although the temperature was set at 21.7 ℃, there was still a bit of fluctuation.

Date	Time		Temperature (℃)
Initial Data Capture			
05/30	16:12 PM	Minimum	21.5
06/06	11:30 AM	Maximum	21.9
Second Data Capture			
06/06	02:08 PM	Minimum	21.5
06/13	10:16 AM	Maximum	21.9
Third Data Capture			
06/13	02:14 PM	Minimum	21.5
06/20	10:30 AM	Maximum	21.9
Fourth Data Capture			
06/20	03:32 PM	Minimum	21.5
06/27	13:19 PM	Maximum	21.9
Fifth Data Capture			
06/27	17:01 PM	Minimum	21.5
07/04	12:20 PM	Maximum	21.9
Sixth Data Capture			

Table 4.2: Voltage stress test schedule

4.2 Final Measurements

The final measurements were conducted in a way similar to that of the endurance test. The sonar system was kept at a fixed point whilst moving a large object with a flat front towards the sensor. The object could only move perpendicular to the sensor, exactly like during the stress tests, at different angles. The purpose of these measurements is to retrieve the standard deviations of the distance and angle. Besides, possible bugs can be encountered and repaired whilst doing the experiments.

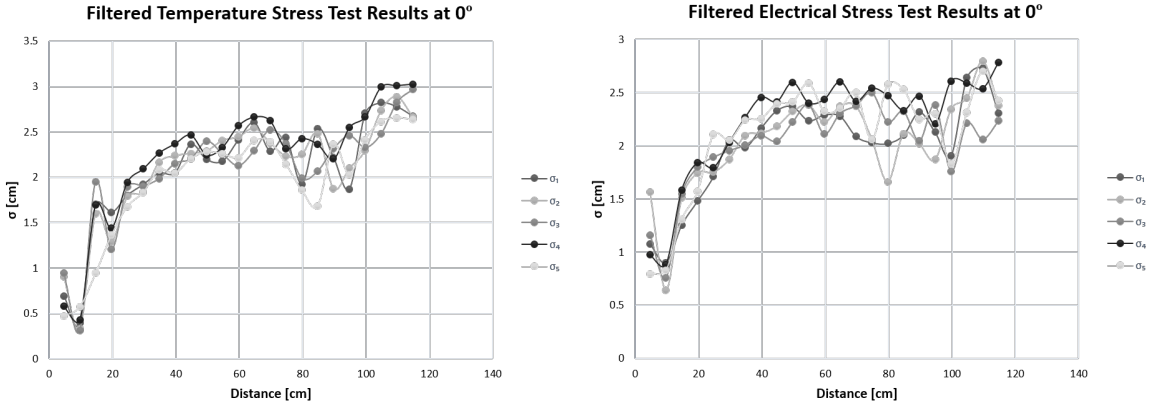
Chapter 5

Results

Sonar sensors have been tested according to the procedures described in section 2.3 and the tests have been executed as described in chapter 4. There is only interest in general changing behavior, sensor specific graphs are therefore not shown, only standard deviations, since these already summarize the results of the entire population.

5.1 Endurance Test

The results of the two test setups (see section 4.1) are split into two subsections below. In general, the standard deviation does not change after all stress test have been performed. There are, however, striking differences between the filtered and unfiltered results, especially when the angles are $\pm 30°$. Every sonar sensor has a specific beam angle, for these sensors it is approximately 60°. Hence $\pm 30°$ is a border case and there are already some faulty readings.



(a) *(b)*

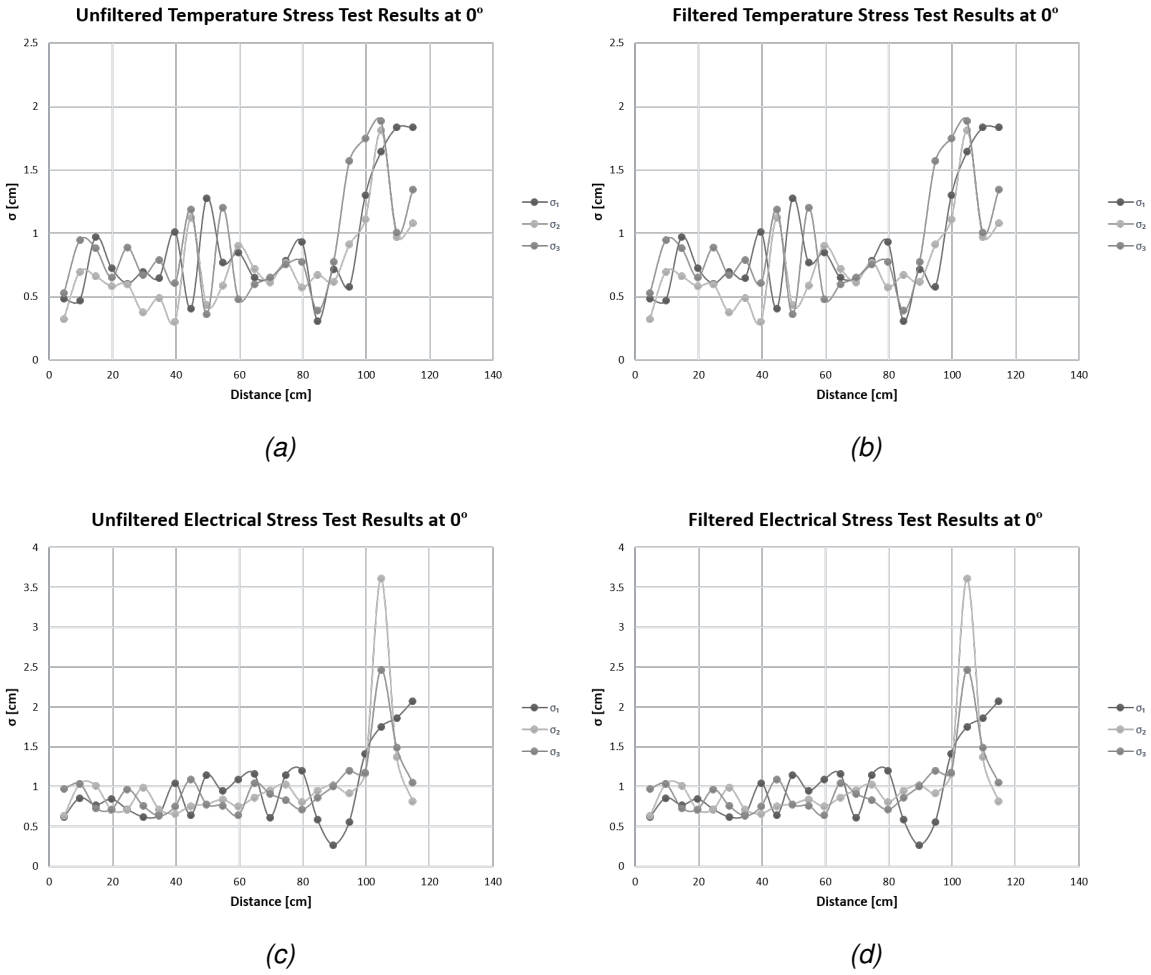
Figure 5.1: Filtered and Unfiltered Standard Deviation of the Different Stress Tests at 0°

Results - Original Setup

Figure 5.1 shows a small difference between the filtered and unfiltered results. The sensors have their smallest deviation below 20cm and it increases to approximately 2.3cm when the distance reaches 40cm. The same behavior is noticed at the other angles (see appendix A).

Results - Adapted Setup

All graphs in figure 5.2 are similar. In both test cases, filtering does not change the graph and there is a higher deviation around 100cm. There is however a difference to the first test setup, namely all deviations are a lot smaller in figure 5.2.



(a) *(b)*

(c) *(d)*

Figure 5.2: Filtered and Unfiltered Standard Deviation of the Different Stress Tests at 0°

Figures 5.3 and 5.4 show similar results. Without filtering, the sensors give a lot of false readings, especially at distances smaller than 40cm and around 80cm. Filtered results still have their highest deviation in between 20 and 40cm. Besides that, the different stress tests show the same behavior at all capture points when their angles are similar.

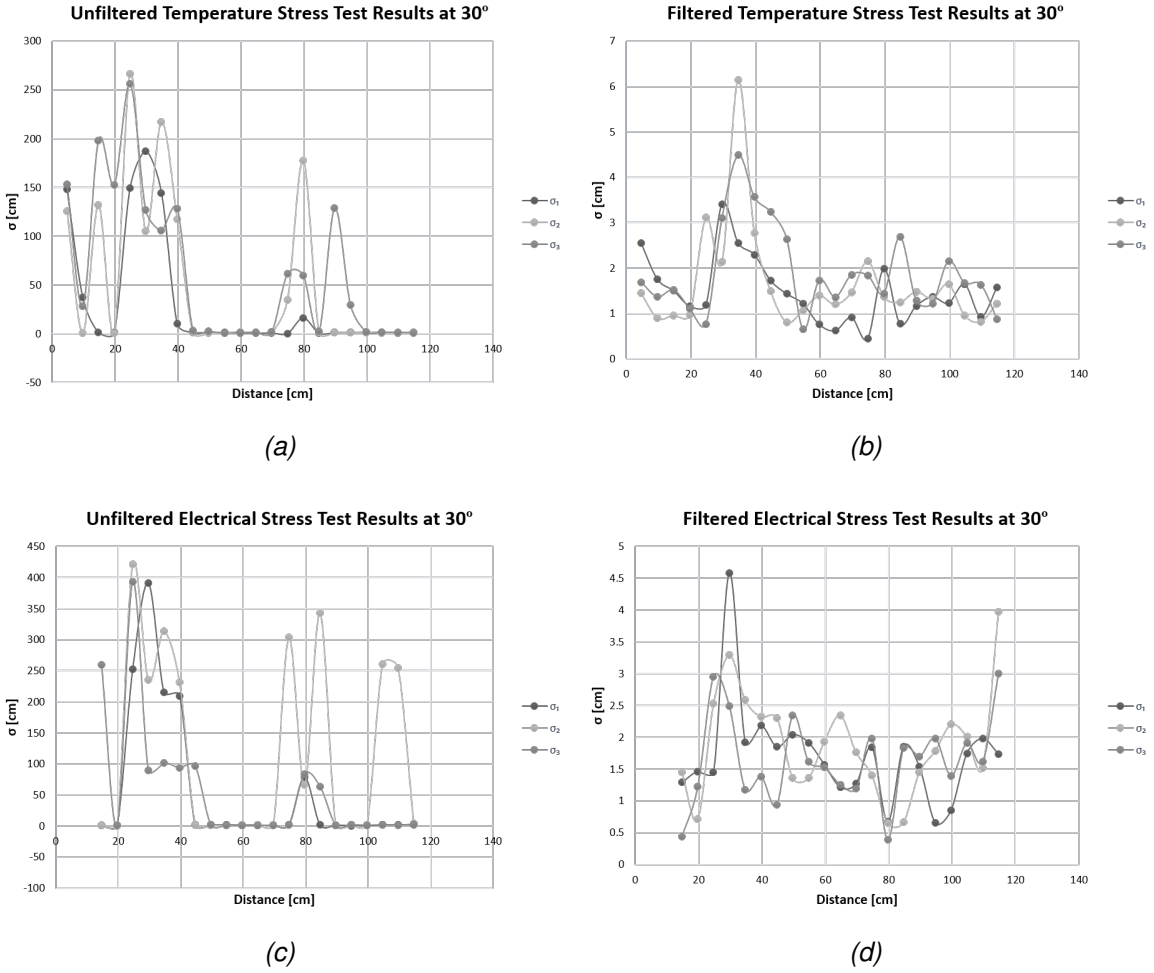


Figure 5.3: Filtered and Unfiltered Standard Deviation of the Different Stress Tests at 30°

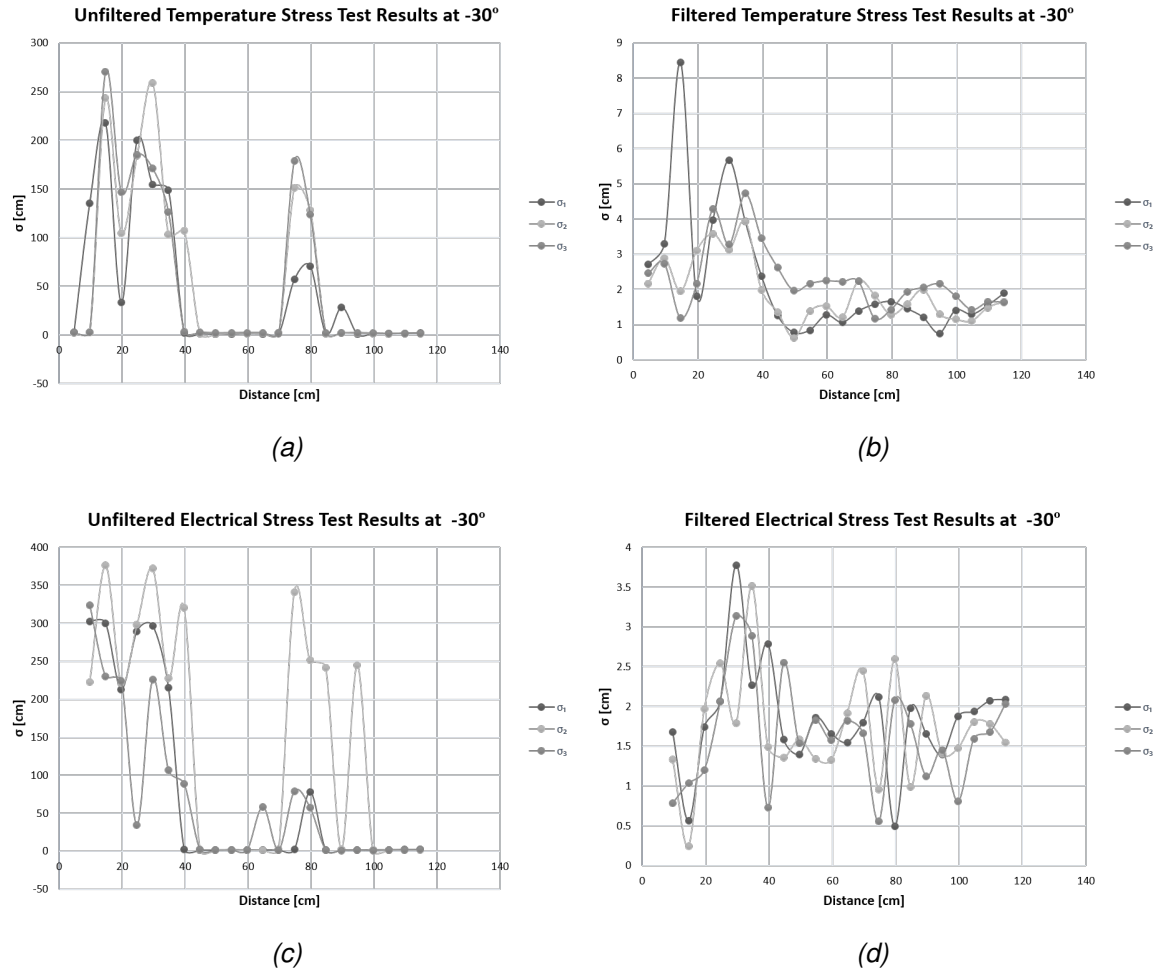


Figure 5.4: Filtered and Unfiltered Standard Deviation of the Different Stress Tests at -30°

5.2 Final System Test

Figure 5.5 hosts the resulting distances and angles when testing the sonar system. The system has a difficult job when capturing positive angles at small distances. One data set, with an angle at 30° and distance 10cm, was entirely invalid. The distance was seven times higher than it should be and therefore it was not included in the graphic. Another dataset at -30° and distance 27.5cm is also not included, since the sonar system was not able to see the object. The object itself was very large, but it was apparently not in range of the sensor. There were also various experiments with small objects, but this did not produce what was expected. They were either not seen by the system or it produced faulty reading. Table 5.1 shows the standard deviations of the measurements. Whenever there is a nonzero angle, the standard deviation is larger than when the angle is zero, as a consequence, the

same is true for the standard deviation.

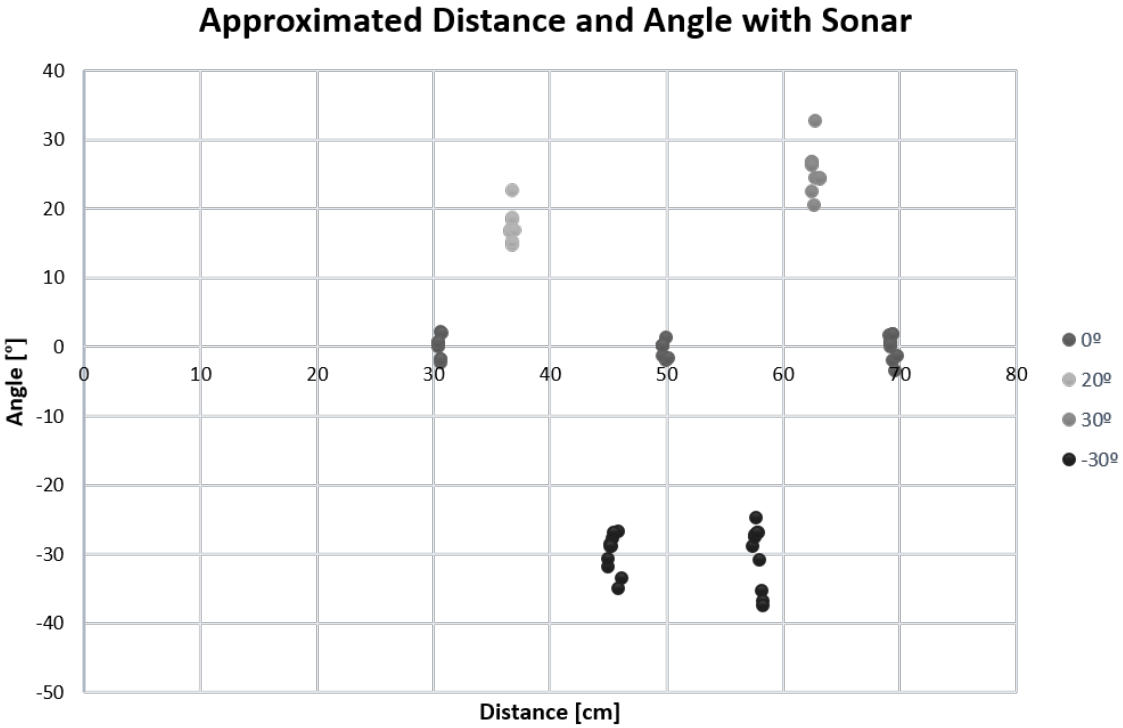


Figure 5.5: Approximated Distance and Angle with Sonar

Table 5.1: Standard Deviations of Figure 5.5

Angle [°]	**Distance [cm]**	**Distance σ_d [cm]**	**Angle σ_d [°]**
0	30	0.54	1.48
	50	0.26	1.15
	70	0.70	1.66
20	36.5	0.28	3.30
30	67.5	4.83	5.37
-30	48	2.61	2.67
	64	6.18	4.40

Chapter 6

Conclusions and recommendations

6.1 Conclusions

The work presented in chapter 3 shows that rather simple code can already produce an useful position estimator. Most errors are automatically catched and repaired such that the system operates continuously. Although not explicitly investigated, there seems to be a correlation between a nonzero angle and the standard deviation in the measurement as is shown in table 5.1. The results show that the approximator successfully approximates both distance and angle, since the standard deviations are below their required maximum. Although the approximator functioned, the reliability could still be improved when increasing sensitivity of the sonar sensors.

The HC SR04 endured stress tests very well. The results show no degradation after weeks of stress testing. Moreover, behavior of the sensors is unreliable at $\pm 30°$ as can be concluded from the unfiltered standard deviations. At these angles, filtered results show the largest standard deviation at distances around 30 centimeters. This contradicts the results at $0°$ where the largest standard deviation is above 100 centimeters for both filtered and unfiltered data. At an angle of $0°$, deviations at all distances never exceed two centimeters, whereas at nonzero angles filtered results show peaks exceeding four centimeters.

6.2 Discussion

Experiments were adapted to fit within the time scope of a bachelor assignment. The experiments would be worth even more with some improvements. Some basic improvements include increased sensor populations, higher stress voltages and temperatures and of course longer stress test periods, not only three or four weeks, but for example six months. It would also be interesting to adapt the current temperature stress test program to cool down to 20 degrees Celsius instead of 30 degrees Celsius, since this would be even closer to the operating range of the sensor. Last of all, the test setup could be adapted to include more angles and the entire range (here $\approx$400cm) of the sensor.

Apart from experiments, code running on the PLASMA CPU could be optimized further. Currently, interference keeps a hard limit on the sample rate, but a paper [6] introduced in 1998 already a technique to compensate for interference. Such techniques would allow for a higher sampling rate and therefore better real time position estimation. Another form of hardware optimization would be integrating a FPU or by using another sonar sensor with a higher sensitivity, especially the beam angle is currently critical. Software optimization when using compiler flags could also improve sample rates and running time.

Bibliography

[1] S. Taghvaeeyan and R. Rajamani, “Two-dimensional sensor system for automotive crash prediction,” in *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, February 2014.

[2] S. Roads, “Plasma CPU,” http://opencores.org/project,plasma, August 2014.

[3] “Immortal - integrated modelling, fault management, verification and reliable design environment for cyber-physical systems,” https://www.utwente.nl/ctit/research/research_projects/international/horizon%202020%20-%20collaborative_projects/immortal/.

[4] “Bonfire immortal chip 2017 git repository,” https://github.com/Project-Bonfire/Bonfire.

[5] “Fast computation of functions on microcontrollers,” http://www.olliw.eu/2014/fast-functions/, October 2015.

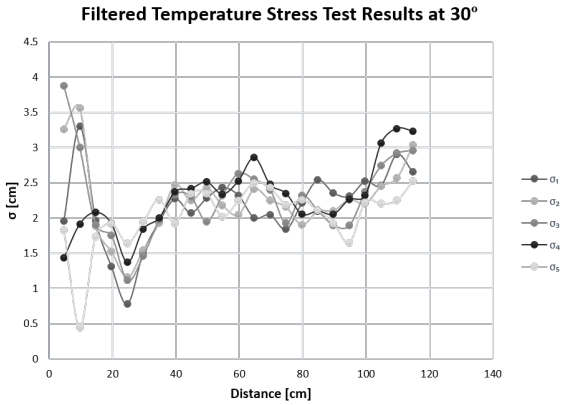
[6] B. Wirnitzer, “Interference cancelation in ultrasonic sensor arrays by stochastic coding and adaptive filtering,” in *IEEE International Conference on Intelligent Vehicles*, 1998.

Appendix A

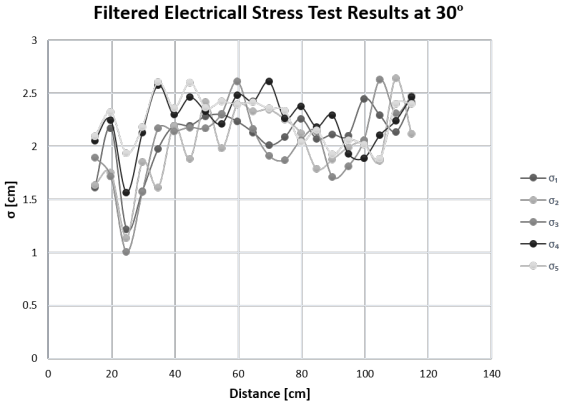
Corrupt Test Results

As already has been explained in chapter 4, there were a few problems regarding the test setup. Although the results cannot be used to identify the HC-SR04 behavior at angles, it still shows similar results as in section 5.1. In these cases, only the unfiltered results are shown, since they yield the same graphs as the unfiltered results.

A.1 Results at 30°

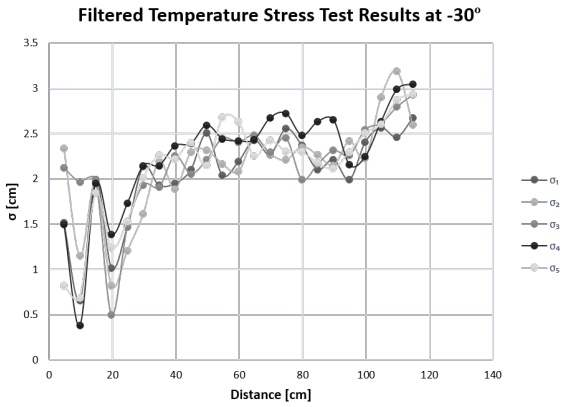


(a) Temperature stress test results at 30°

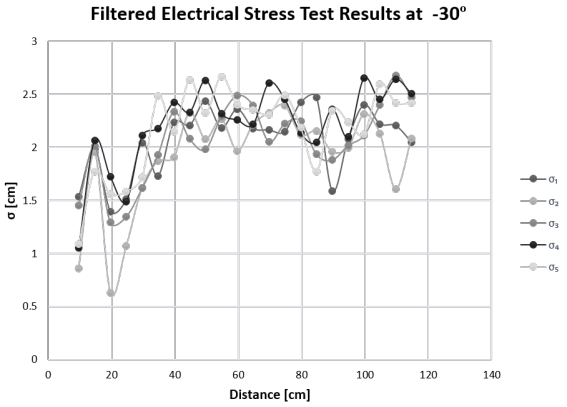


(b) Voltage stress test results at 30°

A.2 Results at -30°



(a) Temperature stress test results at -30°



(b) Voltage stress test results at -30°

Appendix B

Circuit Diagrams

The circuit in figure B.1 shows how one sonar transmitter and two sonar receivers are connected to the Altera FPGA. Figure B.2 depicts the reset circuit connected to the plasma core.

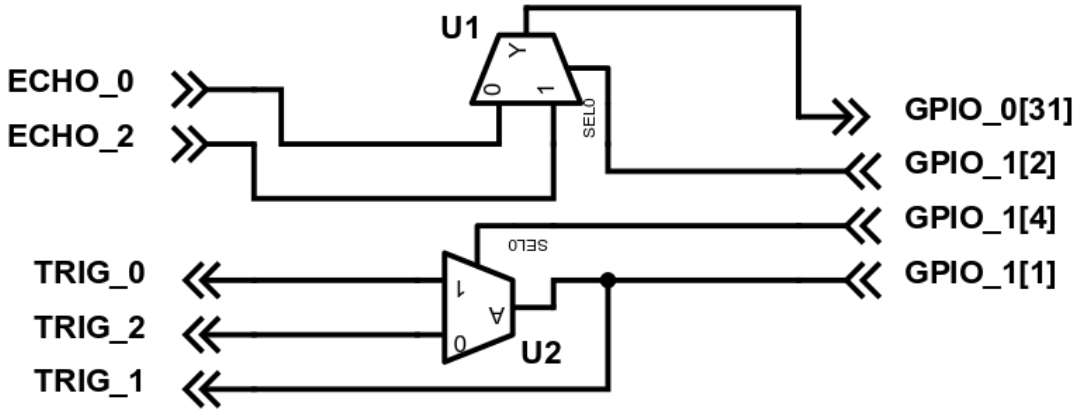


Figure B.1: Two Receivers and One Transmitter Connected to FPGA

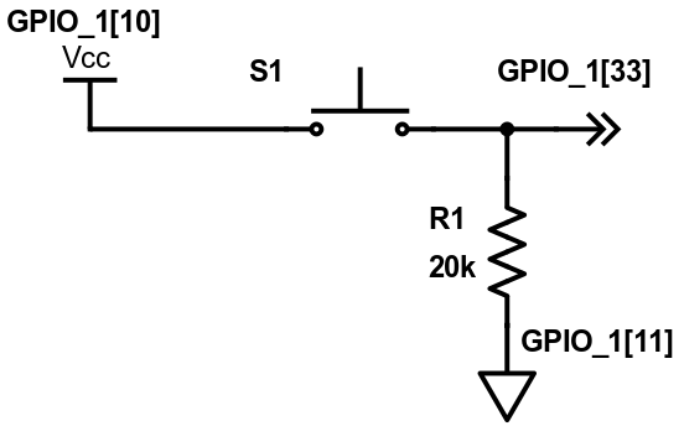


Figure B.2: FPGA Reset Circuit