FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Master Thesis

RP-US-C Sensor Application for Autonomous Illumination Control

Submitted by: Aamir Muhammad

under the guidance of

Dr. Prof. Peter Nauth, Dr. Prof. Andreas Pech

In

Information Technology (M.Sc.) Frankfurt University of Applied Science

November 2021

Abstract

In this thesis, I have mainly discussed the use of Ultrasonic Sensor with an intention to explore the possibility of object recognition. To be specific, developing a system that can differentiate human from non-human objects, so-called box classifier, and evaluating its performance. The main technologies in this work are ultrasonic signal and features extracted from the FFT (Fast Fourier Transform) of the received ultrasonic signal using C programming. 10 different features have been recorded for the statistical analysis on basis of which decisions has been made to differentiate between hard and soft object. For realization of object recognition and estimation of its feasibility with these approaches, a hardware device called Red Pitaya has been used. Red Pitaya is a hardware device consisting of a microprocessor and an ultrasonic sensor. The device was employed to gather a set of numerical values converted from the reflected analog signals of targeted objects. When it comes to software, a UDP client which is developed in C# language is used extensively for the programming, analysis, and control of Red Pitaya and SRF02 ultrasonic sensor. Open source GCC compiler is incorporated, in addition to those other libraries are used to build an application for the Autonomous Illumination Control System. To make changes in the Red Pitaya software and SCP connection is established by means of WinSCP to navigate to the folder of the program and in parallel for a shell connection like Putty has been established to make the program and check for possible errors. Two LEDs are used white LED for human and blue LED for other than human. To control the ON/OFF functionality of the LEDs 2 Relays has been employed which work as push buttons, which will turn ON and OFF the LED when the corresponding Relay is switched ON for 2 seconds. C as a programming language has been used to implement a methodology for the classification. Lastly, for an approach to object recognition using these technologies with high performance, this report discusses the encountered challenges and possibilities for further research.

Table of Contents

Chapter 1 Introduction

1.1 Overview

Over the past few decades, Use of Information and Communication Technology (ICT) and its development have grown rapidly. In terms of object recognition by computer systems, without involving any or less humans' intervention.

1.2 Motivation

In the sphere of Information Technology, the topics of machine learning and its applications have been fascinating for many people not only who study and work in this field but also who don't have any knowledge about computer science. Many people are looking at the rapid growth of technologies of machine learning with interests although there are still some people afraid of its side-effect. There are some expectancies and worries about what and how it affects humans' lives in the future.

Humans have been using ultrasonic signals in many fields for different purposes. Even more, dolphins and bats are typical animals that use ultrasound. The bat reflects the ultrasonic waves generated by itself and detects the distance and direction between the object and the prey, and the dolphin shoots the ultrasonic waves to find the prey and use it for communication. Sonar, short for Sound Navigation and Ranging, is a well-known example that is an apparatus to get information from the ultrasonic waves. Mostly, the Sonar used to detect vessels or fish under the water. On the other hand, the bat shows that use of ultrasonic waves in the air is also available.

In terms of solutions to object recognition problems, there are numerous approaches. However, the decision which methods to use highly depends on the set of input and output data. For the scope of this thesis work, by using given software and hardware tool set, research and development will be conducted using the ultrasonic signals as an input and Analog to Digital conversion (ADC) and the features of FFT model to produce an output because of targeted object recognition.

1.3 Objective

There are numerous research going on in the path to object recognition or detection using approaches of machine learning together with a different type of sources as input data, such as images and videos. This thesis work is little different than the normal machine learning approach but is a small contribution to the same. There are many solutions and research regarding the topic. If it would be possible to realize object recognition with such tools and sound wave data, then it would be a great contribution.

ANNs (Artificial Neural Networks) are broadly employed for many AI (Artificial Intelligence) applications including, robotics, computer vision, speech recognition and etc. ANN results in state-of-the-art accuracy on various application of AI tasks. But what we will be doing is not training a model and predicting the results, instead the approach is to decides on features extracted from the received signals on run time.

This thesis work aims at building an object recognition system that is consisting of an ultrasonic sensor and a box classifier. The system works as a binary classifier that decides or predict if the target is a human or a non-human object based on a received ultrasonic signal. Finally the whole process need to be automated. Originally the ultrasonic sensor was started by the UDP client but now is programmed so by the help of cronjob to start automatically at the reboot.

1.4 Report Structure

This thesis document starts with an introduction to the ultrasonic sensing followed by the basics of the ultrasonic sensor used in Chapter 2. In the same chapter ADC and FFT is discussed as we will be working with the features extracted from the FFT of the received signals, so it is important to explain it.

In Chapter 3 the development, tools, software and methodology of the whole project has been described. The communication protocols and implementation details are also discussed. A section about the automation of the Red Pitaya is given in this chapter. Furthermore, the statistical analysis of the features is given which provide the basis for the classification algorithm.

Other sensors are compared with ultrasonic sensor in chapter 4. Also, areas of applications and future possibilities in given. Finally, the references are stated from where the research data has been gathered.

Chapter 2 Ultrasonic Sensing

Ultrasonic sensors detect objects in close proximity using sound waves over 20 kHz, akin to how bats make use of echolocation to avoid colliding with barriers. Ultrasonic sensors are widely utilized in the automotive industry for applications for advanced driver assistant systems (ADAS), particularly, assistance during parking where four to sixteen sensors are employed to identify impediments at the same time as car parking. These sensors are also utilized in robots and several other applications that necessitate accurate position, presence, or proximity sensing in the industrial sector. This document will discuss Ultrasonic time of fight sensing and also, considering the system and other elements that influence ultrasonic sensing [1]. The detailed applications will be discussed in theChapter 4.

2.1 Time-of-Flight Sensing

The physics of sounds is discussed in this part, as well as the advantages of utilizing ultrasonic sensors in a range of applications.

2.1.1 Principles of Ultrasound

Without making physical contact, these sensors know how to compute distance and identify the existence of an entity. They achieve this as a result of creating and listening for an ultrasonic echo. The effective range in air varies depending on the sensor and object qualities. It might range from a small number of centimeters to several meters. An object within the sensor of the field of view creates and sends out ultrasonic pulsations, which are echoed in the sensor by the ultrasonic sensor or transducer [1].

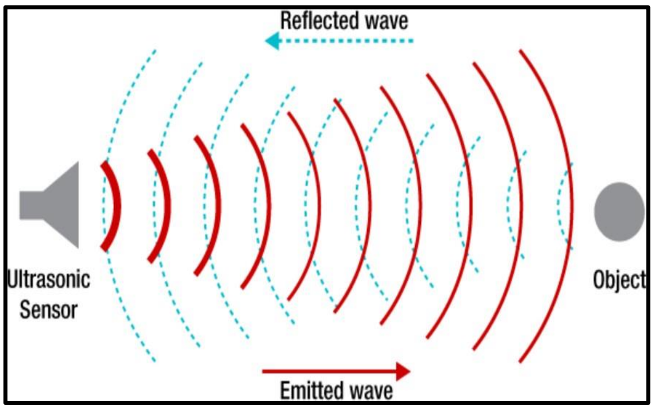


Figure 2.1 Ultrasonic Time-of-Flight Measurement [1]

Furthermore, ultrasonic sensors are piezoelectric transducers that can transform the electrical signal to mechanical vibration and mechanical vibration back to an electrical signal. Accordingly, the ultrasonic sensor serves, in a monostatic approach, as a transceiver, acting together as a microphone and a speaker at the same time [1].

Moreover, these sensors can detect the difference of time between the transmitted and received or expected echo. The acquired Round-Trip time, however, can be utilized to compute the space connecting the item and the sensor as the speed of sound is well-known as a variable. The Ultrasonic Distance Calculation is shown in Equation 2-1.

$$d_{One\ Way} = \frac{t_{Round\ Trip} \times V_{Sound}}{2} \tag{2-1}$$

The measurement of the time flight is grounded on the transmission of the sound time in the technology of Ultrasound sensing. It is worth noting that the speed of sound changes through the air with temperature. As found, the sound speed in dry air at 20 degrees centigrade is 343m/s or 2.9 seconds per kilometer [1].

2.1.2 Why Use Ultrasonic Sensing?

Regardless of color, shape, or transparency, ultrasonic sensors can sense a wide range of materials. It just requires that the target substance must be liquid or solid. Oil, water, plastic, metal, glass, rocks, woods, sand, and other hard materials that do not absorb the sound can all be detected without touching them.

Sound can be reflected by these materials through the air. Certain things, such as surfaces with a slope that divert the echoed sound away from the sensor or porous objects like squashy clothes, sponges, can be difficult to be detected. More reflected ultrasonic energy is absorbed by these materials [1].

2.1.3 Comparison of Ultrasound to other technologies of sensing

Given their excellent quick response, low cost, and excellent resolution, infrared sensors can be employed and intended for barrier detection. Given its non-linear qualities and reliance on reflectance attributes, IR sensors, on the other hand, necessitate knowledge of the surface properties before deployment. Because different surface materials reflect and absorb IR light in different ways, reliable distance measurements necessitate target material identification [1].

Ultrasonic technology and optical-based sensing technologies are both based on the same idea. Optical technology, on the other hand, makes use of LED to generate waves of light and spot flight time that may then be converted using the principle of the speed of light. As light travels at a considerably faster rate than sound, ultrasonic sensing is slower than optical sensing. However, it has limits in the immediate surroundings of illumination settings, as well as smoky or foggy environments, where the light receptor finds it difficult to detect the produced light. Detecting clear objects like glass or water is also difficult with optical sensors. Ultrasonic waves bounce off certain materials, although light passes through them [1].

Moreover, as an alternative to a single time-of-flight measurement, technologies based on Radar and LIDAR strive to deliver multi-pointed data. This enables very precise data points as well as the capacity to map out and discern small events in the surroundings. These systems, however, are substantially more expensive than the other alternatives described previously due to their expanded capability [1].

Distinction between PIR, Ultrasonic, and mWave are summarized in Table 2.1

	Passive Infrared	**Ultrasonic**	**Optical ToF**	**mmWave**
Detection Range	0.1 to 5 m	0.1 to 10 m	0.01 to 20 m	0.01 to 100+ m
Resolution	Few cm	Few mm (transducer dependent)	Few mm (optics dependent)	Few mm (range dependent)
Field of View	Up to 180°	5° to 120°	0.15° to 120°	5° to 160°
Current Consumption	<5 mA	72 mW to 336 mW (active) 2-9 mW (standby/sleep)	100 µW to 200 mW (active) ~ 80 µW (standby/sleep)	0.5 W to 1.5 W
Solution / Module Size	Medium	Medium	Small	Large
Aesthetics	Requires lens to achieve range and wide field of view	Exposure to medium for longer range	Hidden behind dark glass	Penetrates most materials (not metal)
Measuring Medium Speed	Infrared light (emitted by object)	Sound	Light	Light
Single Sensor System Cost (US$)	< $1	$1 - $3	$1.5 - $4	$18 - $26
Key Differentiation	• Limited performance in high heat environments and corner regions • Insensitive to slow motion • Prone to false positives	• Effectively detect solid and transparent glass surfaces • Able to detect objects in a smoke/gas-filled environment	• Target localization (up to 3 zones of detection) • Precise long-range measurements	• Provides range, velocity, and angle data • Can penetrate non-metal materials • Intelligent object differentiation

Table 2.1 Comparison of Proximity Sensing Technology [1]

2.1.4 Typical Ultrasonic-Sensing Applications

Ultrasonic sensing applications can be divided into three categories:

2.1.4.1 Ranging Measurement

Within each time-of-flight transaction, it measures the distance traveled by one or more items to or from the sensor. The rate at which the sensor updates the distance is determined by how time-consuming the echo-listen mode is. The greater the detectable range, the longer the sensor waits for the echo [1].

For example: obstacle avoidance sensors in robotics, level transmitters, and Ultrasonic Park assist sensors.

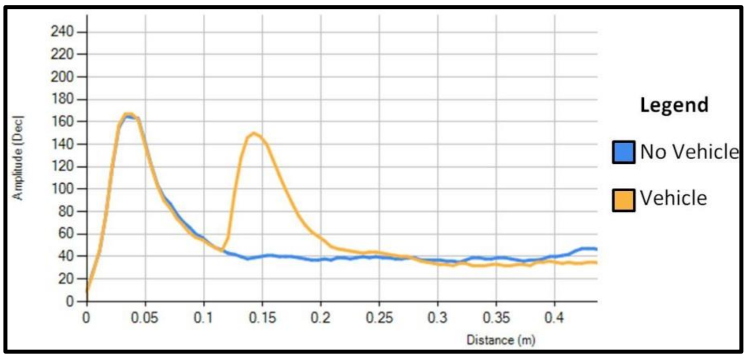


Figure 2.2 Echo Response for Vehicle Presence [1]

2.1.4.2 Proximity Detection

A physical change in the sensing environment leads to a considerable shift in the ultrasonic echo signature. This ultrasonic sensing binary technique is less contingent on the range and more on the stability of the eco signature [1].

For example: detection of vehicles in the premises of parking, edge and cliff detection in robots, surveillance and stability system, and object detection as shown in Figure 2.2.

2.1.4.3 Surface Type Detection

Softness and hardness of the material can be assessed not directly with measurements through ultrasonic using data of raw ultrasonic echo rather than time-of-flight measurement. Ultrasonic sound waves are more effective because it bounces off sturdier materials with less loss. Many sound waves are absorbed by softer surfaces like carpet and foam which ultimately results in a slighter echo response. [1]

For example: terrain type detection in lawnmowers and floor type detection in vacuum robots as shown in Figure 2.3.

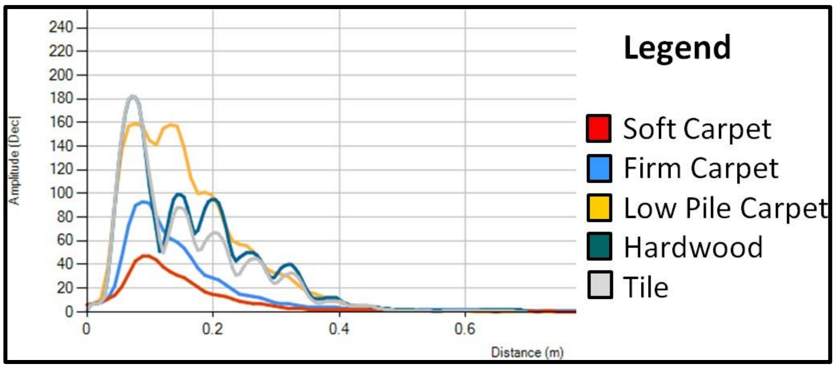


Figure 2.3 Ultrasonic Surface Type Detection [1]

2.2 Considerations of Ultrasonic Systems

The ultrasonic systems are made up of the following components:

- Analog-Front End to impel the source as well as regulate the received signal
- Ultrasonic sensors
- Transducers
- Additional signal processing ability to enrich the measured data with intelligence
- Analog to digital convertor

The analog front-end section is in charge of operating the transducer and amplifies and filters the procured echo data so that it can be processed ahead. In discrete and AFE solutions, the signal process is carried out entirely by the control unit, however, with its in-chip intelligence in the ASSP solution, signal processing is split between the integrated DSP and control unit [1].

2.2.1 Introduction to the Ultrasonic System

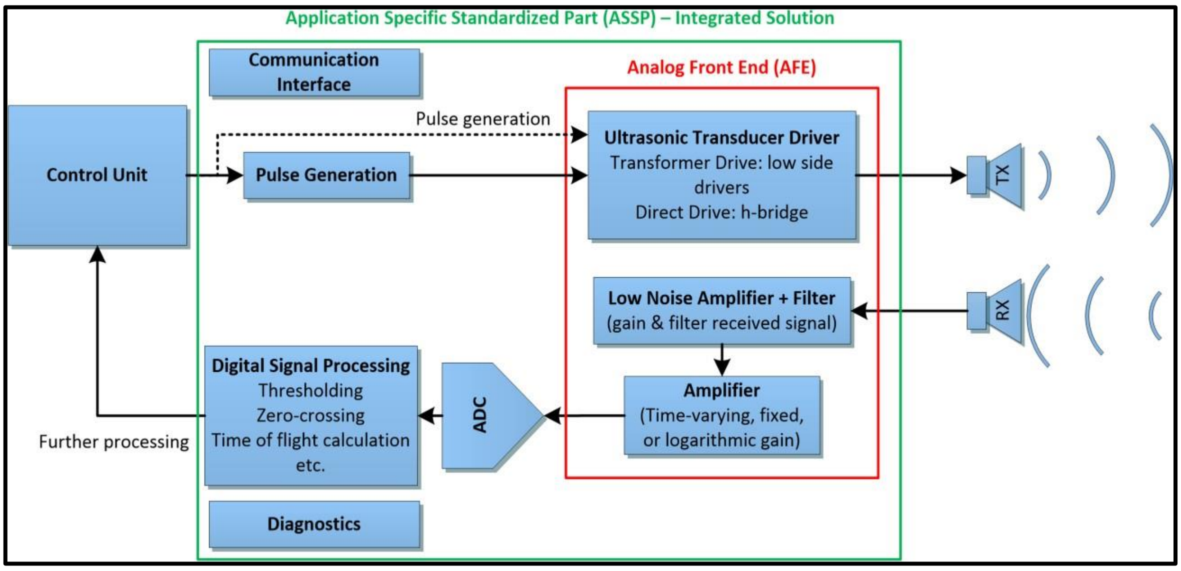

Application Specific Standardized Part (ASSP) – Integrated Solution
Communication Interface
Analog Front End (AFE)
Pulse generation
Control Unit
Pulse Generation
Ultrasonic Transducer Driver
Transformer Drive: low side drivers
Direct Drive: h-bridge
TX
Low Noise Amplifier + Filter (gain & filter received signal)
RX
Digital Signal Processing
Thresholding
Zero-crossing
Time of flight calculation
etc.
ADC
Amplifier (Time-varying, fixed, or logarithmic gain)
Further processing
Diagnostics

Figure 2.4 Ultrasonic System Level Block Diagram [1]

The ultrasonic system knows how to

- Be entirely discrete that made up of diodes, filters, amplifiers, and several other unreceptive apparatuses.
- Be an incorporated Analog Front End
- MCU must be completely incorporated on-chip.

It is worth noting that the transducer you choose has a big impact on the ultrasonic module's overall performance. The remaining part of this part explains how to choose a transducer, frequency, and topology, as well as, how to improve performance using optimization techniques [1].

2.2.2 The signal processing and ultrasonic echoes

It advocates driving transducers with a sign or square wave at the midpoint frequency in order to attain the most excellent outcomes. The system must then listen for return echoes caused by a thing in the field of view of the transducer transmits a resonance or echoes at its resonance frequency. Before sending the signal to an ADC, an Ultrasonic system often filters the returning echo to reduce clamor and amplify it up [1].

Moreover, Designers can confirm that the reflected echo is that of the transducer after looking at the zero-crossing frequency data. This can also be used to identify motion and its direction which is a variation in the frequency of the given-out sound waves. The data could be transmitted to the ADC for additional processing of signals when the returning signals have been filtered and gained suitably. The signals from the ADC output are shown in Figure 2.5

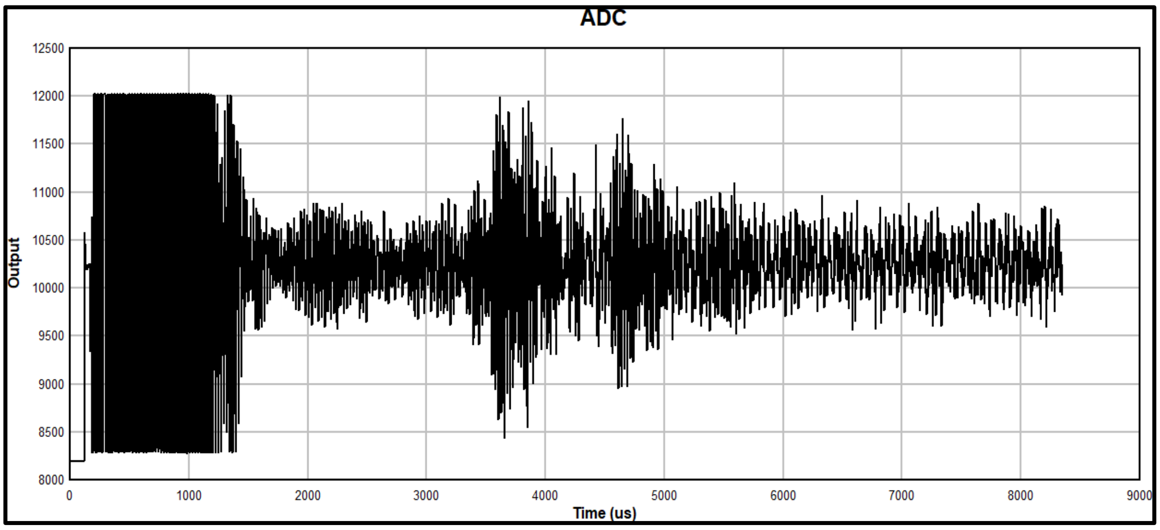


Figure 2.5 Typical Output From ADC

Later than the signal has been digitalized, it is all set to be processed by a microcontroller or a digital signal processor. To eliminate out-of-band noise, it first passes through a bandpass filter. The output is shown in Figure 2.6

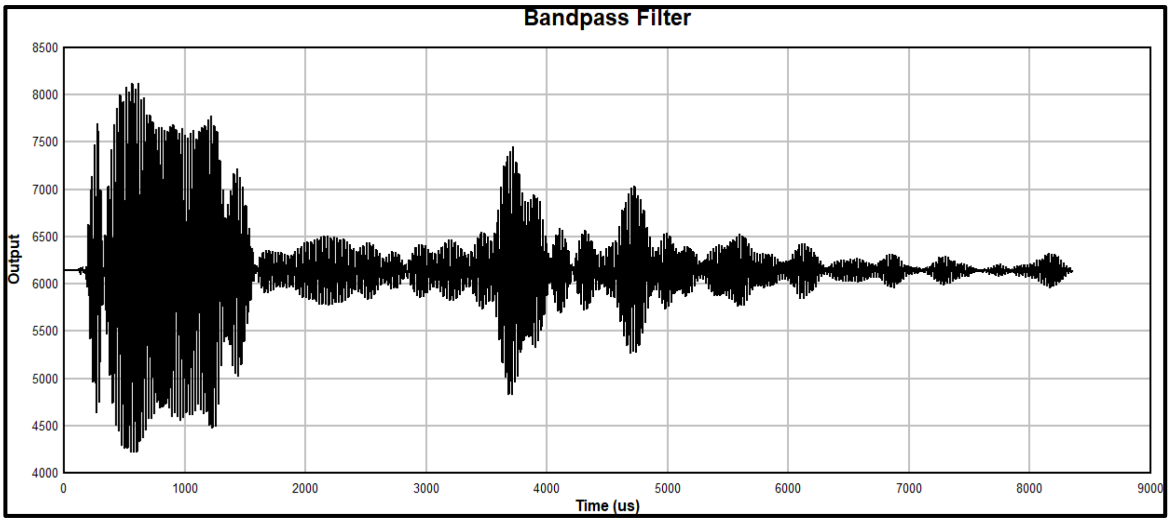


Figure 2.6 Typical Output from Bandpass Filter [1]

The signal is then rectified to take out the supreme value of signals, as shown in the Figure 2.7.

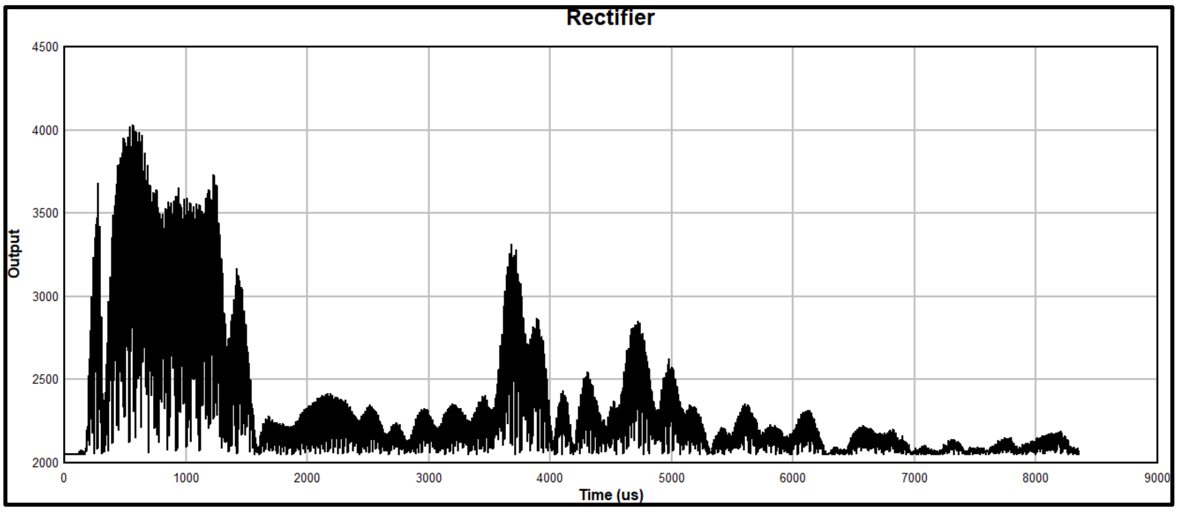


Figure 2.7 Typical Output from Rectifier [1]

After rectification, a peak hold is commonly used before applying a low-pass filter to guarantee that the amplitude of rectifying signals is unfiltered. Figure 2.8 shows how a demodulated output can be created using the peak hold and low-pass filter. The process makes it simple to use entries to further modify the signals, such as removing clamor, extracting time-of-flight data as well as amplitude data and width echo. The demodulated signal is also known as envelope signal [1].

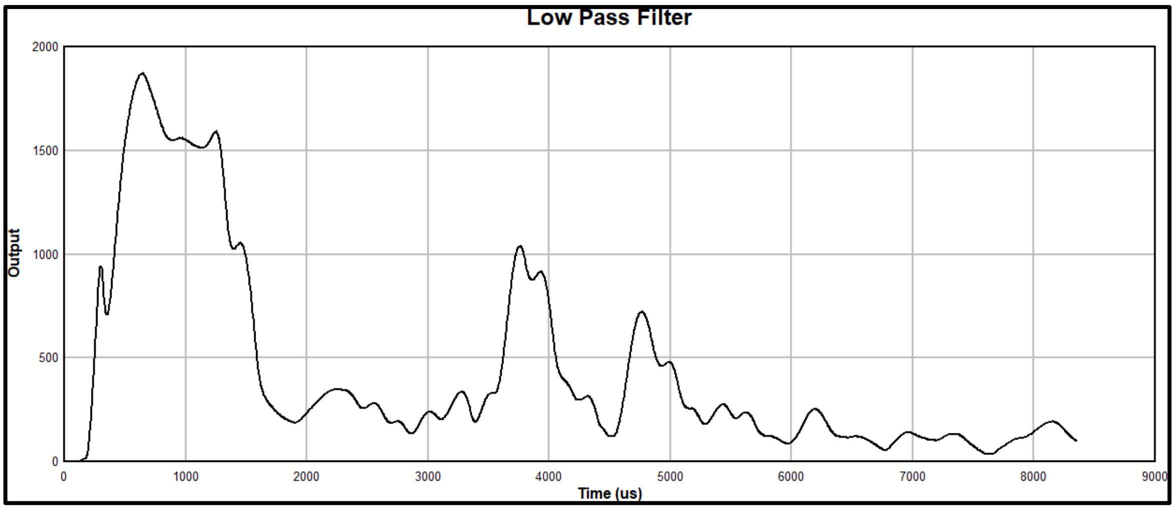


Figure 2.8 Typical Output from Low-Pass Filter [1]

2.2.3 Transducer Topologies

There are two types of transducer topologies: monostatic and bistatic as shown in Figure 2.9. The topology has to be determined by the short-term needs. The former occurs when only the transducer broadcasts an echo and listens for returning echoes. In most cases, it is the most gainful technique. The disadvantage of this type is that the excitation of ringing-decay of sensors causes a blind zone, limiting the least range recognition. This blind zone can be minimized in a monostatic system by adding a dampening resistor [1].

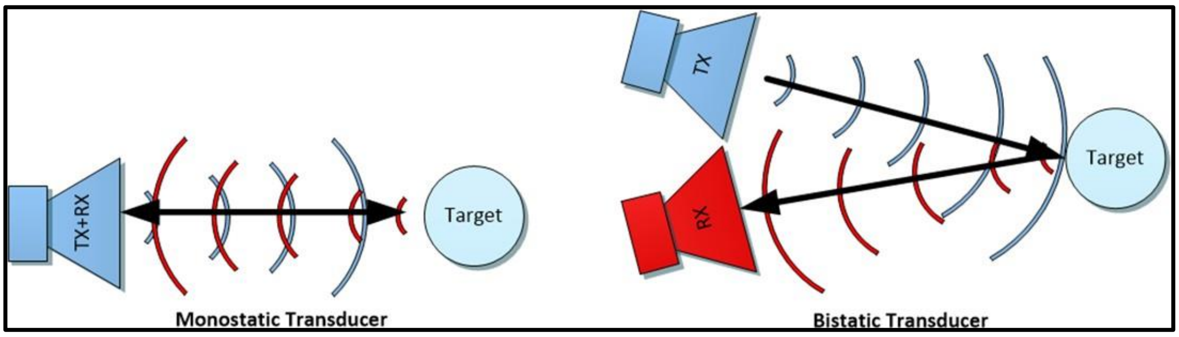


Figure 2.9 Monostatic vs. Bistatic Configuration [1]

A bistatic topology with two separate transducers must be used to avoid ringing degradation. The disadvantage of utilizing the bistatic approach is that it necessitates additional calibration because the time-of-flight calculation must take into account the position of the inward echo at the receiver end [1].

2.2.4 Transducer Frequencies

For air-coupled applications, ultrasonic transducers operate at frequencies between 30 and 500 kHz. The rate of attenuation increases as the ultrasonic frequency rises. As a result, long-range sensors should be low frequency as such 30 -80KHz while short-range frequency should be high frequency [1].

Moreover, the ringing decay of the higher frequency sensors is less allowing for a minimum detection range. Transducers in the 1-MHz band are commonly utilized for liquid-level sensing.

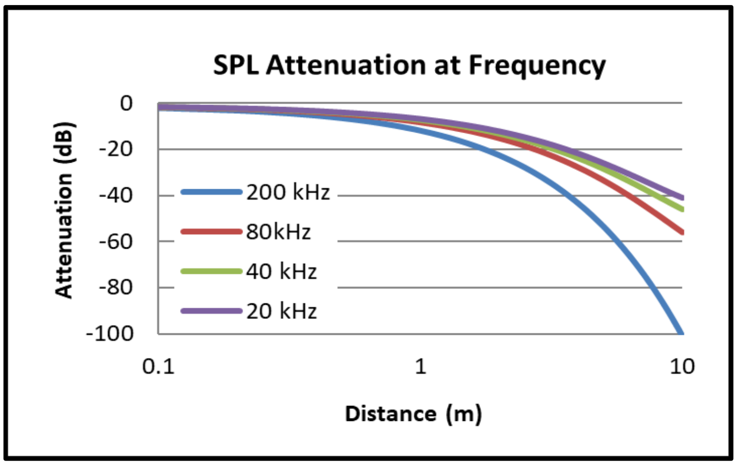


Figure 2.10 Correlation of Distance Measured and Frequency

The following equation depicts the relationship between distance, and resolution, directivity, attenuation, and frequency:

↑ Frequency: ↑ Resolution :: ↑ Narrower Directivity :: ↑ Attenuation :: ↓Distance

Transducers know how to lessen to fifteen 15° or broad 180° field of vision. The field of view narrows as the frequency increases. A restricted field of view can also be created with a low-frequency transducer by putting a "horn" in the region to direct the reverberations into a tighter pattern [1].

2.2.5 Pulse Count

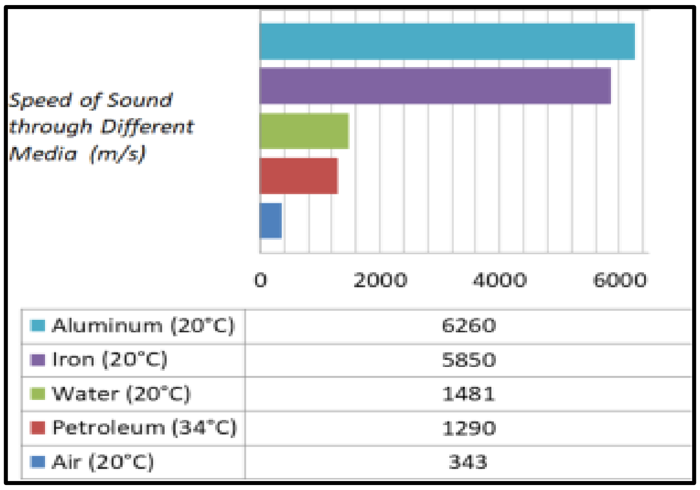
Another characteristic of the ultrasonic sensors is pulse count which is illustrated by the pulses rate released by a transducer. The SPL will increase as the pulse count increases, however, as the transducer bursts for a longer period, the minimum detection range decreases [1].

2.3 Factors that influence Ultrasonic Sensing

Because waves of sound having a frequency of 20 kHz or higher are inaudible to humans, they are referred to as ultrasonic.

2.3.1 Transmission Medium

The transmission qualities of sound, as well as the speed of sound, vary depending on the medium. Ultrasonic sensors are designed to transmit sound waves through air, liquids, or solids, but they are rarely designed to transmit sound waves through more than one medium [1].



🟦 Aluminum (20°C)	6260
🟪 Iron (20°C)	5850
🟩 Water (20°C)	1481
🟥 Petroleum (34°C)	1290
🟦 Air (20°C)	343

Figure 2.11 Speed of Sound Through Different Media [1]

Because the attenuation of ultrasonic waves in the air rises with frequency, the frequency range of air-coupled ultrasonic application is limited to less than 500KHz. Transducers in the low Mhz range can be used in liquid and solid applications for excellent accuracy.

2.3.2 Acoustic Impedance

To identify things with significant acoustic impedance mismatch, sound waves can pass through a variety of mediums. The product of density and acoustic velocity is defined as acoustic impedance (Z). When compared to most liquids or solids, the air has substantially lower Acoustic Impedance [1].

Material	**Density** (kgm^{-3}) (2)	**Acoustic Velocity** (ms^{-1}) (3)	**Acoustic Impedance** $(kgm^{-2}s^{-1} \times 10^{6})$ (4)
Air	1.3	330	.00429
Water	1000	1450	1.45
Muscle	1075	1590	1.70
Aluminum	2700	6320	17.1
Iron	7700	5900	45.43
Steel	7800	5900	46.02
Gold	19320	3240	62.6
Skin	1109	1540	1.6

Table 2.2 Acoustic Impedances of Target Materials [1]

The dissimilarity between two objects in the acoustic impedance is known as impedance mismatch, as shown in equation 2-2. More energy is replicated at the interface between the two channels as the mismatch in the acoustic impedance is bigger.

$$Reflection\ Coefficient = R = \left(\frac{Z_2 - Z_1}{Z_2 + Z_1}\right)^2 \qquad 2\text{-}2$$

Example 1: Skin and Air:

Air has an acoustic impedance of 00429 while the skin has an acoustic impedance of 1.6. Equation 2-3 is obtained by plugging these numbers into the reflection coefficient.

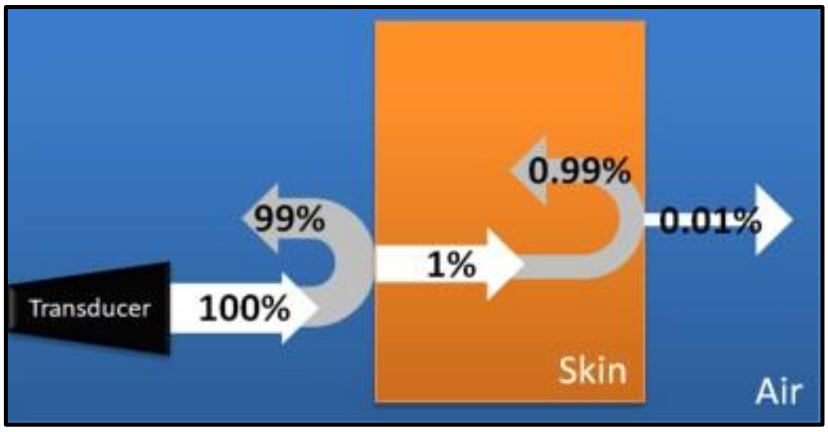


Figure 2.12 Reflection Coefficients for Skin and Air Boundaries [1]

$$\left(\frac{Z_{skin} - Z_{air}}{Z_{skin} + Z_{air}}\right)^2 = \left(\frac{1.6 - 0.00429}{1.6 + 0.00429}\right)^2 = 0.99 \qquad 2\text{-}3$$

The amount of energy reflected, absorbed within the material, and permeated through is determined by doing this calculation at each barrier [1].

Example 2: Steel and Water:

The same calculation shows that a steel and water boundary reflect 8 percent of the transmitted echo in liquid-based detection using equation 2-3.

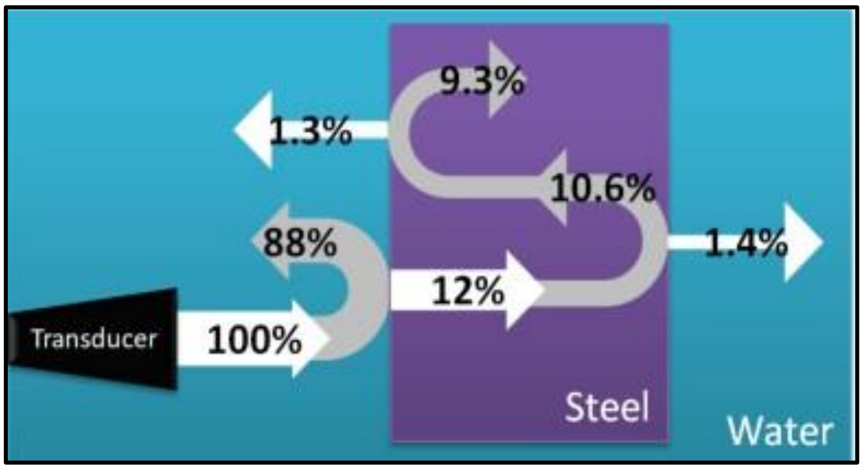


Figure 2.13 Reflection Coefficients for Water and Steel Boundaries [1]

2.3.3 Radar Cross Section

The ability of a target to reflect ultrasonic waves to the transducer is measured by the radar cross-section. Curved or slanted objects can disperse the bulk of the ultrasonic waves directed at the object which ultimately results in a feeble echo response.

- Flat
- Large
- Smooth
- Dense

The best answers come from objects that satisfy these conditions, such as a wall or a floor. The sensory response will be reduced by small things or things that incompletely deflect sound (such as flora and fauna). When possible, for improving sensing responsiveness, a planner entity should encounter the sensor at a 90-degree angle. Larger angular deviations are possible on round or stiff surfaces. Figure 2.14 shows how the form of a target surface affects the reflected ultrasonic waves.

The radar cross-section of a target is classified in Equation 2-4 (σ):

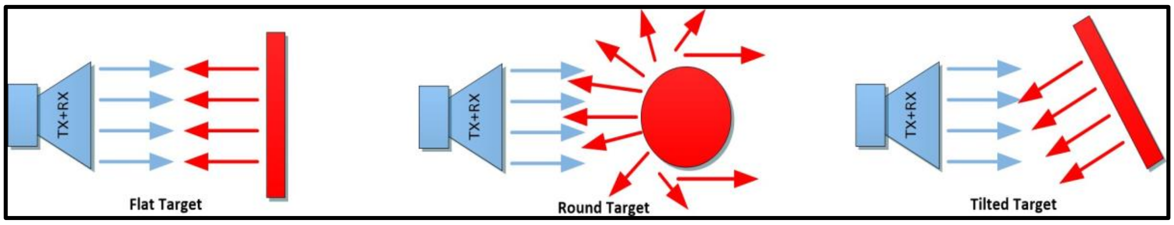


Figure 2.14 Ultrasonic Echoes Based on Target Shape [1]

$$\sigma = \text{Projected cross section} \times \text{Reflectivity} \times \text{Directivity} \qquad 2\text{-}4$$

2.3.4 Ambient Conditions (Temperature, Humidity, Debris)

External elements of the environment, to count few humidity, warmth, and in-band ambient clamor, influence the velocity of an air-coupled ultrasonic echo. As the temperature rises, the sensing range diminishes. Although the sensing rate reduces when humidity rises, the impacts are generally overlooked since they are minor. The rate of attenuation varies with temperature and humidity [1].

$$V_{Sound} = 331\frac{m}{s} + \left(0.6\ \frac{m}{s°C} \times Temperature(°C)\right) \qquad 2\text{-}5$$

Airborne material, such as rain, snow, or dust, can reduce ultrasonic energy and change the sensor's field of vision. Minor dust or dirt deposits do not affect the operation of a closed-face transducer [1].

2.4 SRF02

The SRF02 is an ultrasonic rangefinder with a single transducer on a tiny PCB. It has two interfaces as serial and I2C. The sequential interface is a typical TTL level UART set-up with 9600 baud, no uniformity in bits, and 1 start and 2 stop that can be directly linked to the ports of serial of any microcontroller. On a single serial and signal I2C, up to 16 SRF02s can be linked. One of the new instructions in the SRF02 is the capability to drive an ultrasonic burst, not including a reception cycle as well as the capability to complete a reception cycle without a prior burst. It has become a popular request for a feature on our sonars, and the SRF02 is the first to get it. As it utilizes one transducer equally for reception and transmission, the minimum range of SRF02 is elevated than the other double transducer ranges. The SRF02, like all of our rangefinders, has a minimum measuring range of roughly 22cm (8.7 inches). It may compute in uS, cm, or inches [2].

2.4.1 Modes of Operation

The SRF02 includes two different operating methods. Serial and I2C modes both are available. For Serial Mode, the Mode pin is joined to 0v Ground, however, for I2C Mode, it is not connected or connected to +5v Vcc.

Figure 2.15 SRF02 Ultrasonic Range finder [2]

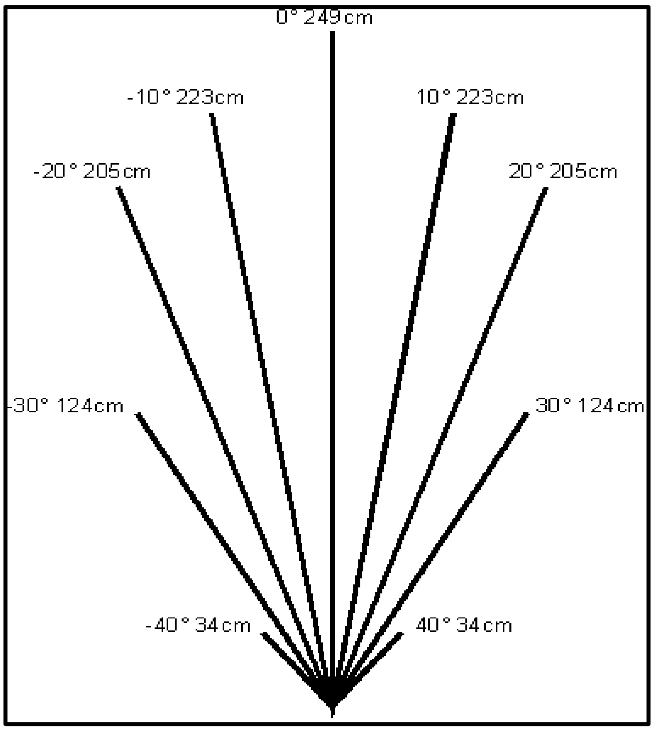


Figure 2.16 Measured beam pattern for the SRF02 [2]

2.4.2 I2C Communication

It must be made sure that the mode pin is not connected to anything when using the SRF02 in I2C mode it must be left disconnected. Accepted controllers, such as PicAxe, stamp BS2p, 00Pic along with a broader range of microcontrollers support the I2C bus. The SRF02 functions similarly to the widely used 24xx series EEPROMs in terms of programming, with the exception that the address of I2C is completely different. The default shipping address of SRF02 is 0xE0. It knows how to set to any of the 16 addresses E0, E2, E6, E4, EC, EA, FA, F0, F2, F4, FE, and FC which allow the usage of up to 16 sonars [2].

2.4.3 Connections

The "Mode" pin, which contains an inbuilt pull-up resistor, should be kept disconnected. On the I2C bus, the SDA and SCL must have a pull-up register to +5v. You just need one pair of resistors for each module, not two. Rather than the slaves, they are usually found with the bus master. The SRF02 is never a bus master and is always a slave. I recommend 1.8k resistors if you need them. You don't need to add any more pull-up resistors to some modules, such as the OOPic [2].

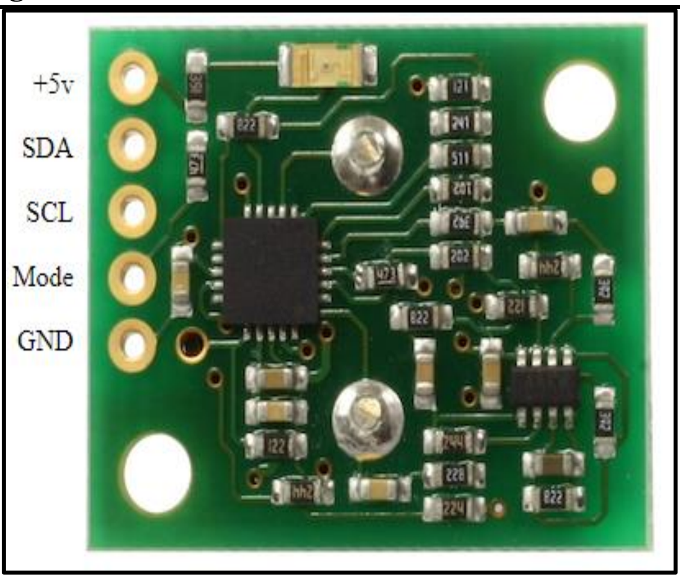


Figure 2.17 Connections [2]

2.4.4 Registers

The SRF02 is represented by a set of six registers.

Location	Read	Write
0	Software Revision	Command Register
1	Unused (reads 0x80)	N/A
2	Range High Byte	N/A
3	Range Low Byte	N/A
4	Autotune Minimum - High Byte	N/A
5	Autotune Minimum - Low Byte	N/A

Table 2.3 Registers [2]

The lone location that can be written is 0. The command register is positioned at location 0 and is used to initiate a ranging session. It is not readable. The SRF02 software revision can be found by reading from location 0. The SRF02 will not react to instructions on the I2C interface while ranging up to 56 milliseconds.

In positions 2 and 3, the unsigned result of 16 bits of the most recent ranging is saved which is high byte first. Depending on the instructions, this number is the range in centimeters, the range in inches, or the flight duration in microseconds. A value of 0 designates that no substance was discovered. It is pertinent to say that not to start a ranging quicker than ever 65ms to allow the preceding disintegration to die away. The 16bit unsigned minimum range is found at locations 4 and 5. This is the adjoining range to which the sonar becomes able to measure. Further information is given below in the Autotune section [2].

2.4.5 Commands

It is a set of three commands that start with a range of 80 to 82 and return the results in microseconds, centimeters, or inches. The identical function is performed by another set of three commands, that is 86 to 88, but the burst is not transmitted. These are utilized when sonar has communicated the burst. You are responsible for synchronizing the commands of the two sonars. A command 92 sends a rapture with no ranging, then a series of commands alter the address of I2C [2].

Command		Action
Decimal	Hex	
80	0x50	Real Ranging Mode - Result in inches
81	0x51	Real Ranging Mode - Result in centimeters
82	0x52	Real Ranging Mode - Result in micro-seconds
86	0x56	Fake Ranging Mode - Result in inches
87	0x57	Fake Ranging Mode - Result in centimeters
88	0x58	Fake Ranging Mode - Result in micro-seconds
92	0x5C	Transmit an 8 cycle 40khz burst - no ranging takes place
96	0x60	Force Autotune Restart - same as power-up. You can ignore this command.
160	0xA0	1st in sequence to change I2C address
165	0xA5	3rd in sequence to change I2C address
170	0xAA	2nd in sequence to change I2C address

Table 2.4 I2C Commands [2]

2.4.6 Ranging

To begin a range, write one of the above commands to the instruct register, then wait for the specified period for the range to complete before reading the results. At the start of each range, the echo buffer is cleaned. The range can be read from sites 2 and 3 after the range lasts for 66mS [2].

2.4.7 Checking for Completion of Ranging

To wait for ranging to finish, you don't need to utilize a timer on your controller. You can use the actuality that the SRF02 would not act in response to one act of I2C when ranging to your advantage. You will get 255 (0 into FF). it happens as if nothing is driving the SDA, it is pulled high. When the ranging is finished, the SRF02 will act in response to the I2C bus again, so just maintain reading the register until and unless it is no longer 0xFF (255). The sonar data can then be read. While the SRF02 is ranging, your controller can use this time to do other things. After commencing the range, the SRF02 will always be ready in 70 milliseconds [2].

2.4.8 LED

When the red LED is turned on, it flashes a code for the address of I2C as shown below. During the "ping" while ranging, it also gives a brief flash.

2.4.9 Changing the I2C Bus Address

Only sonar on the bus is required to alter the SRF02's I2C address. After the address, write the three sequence commands in the right order. Write the following to address 0xE0 to modify the sonar address that is presently at the default shipped address (0xE0) to 0xF2 such as (0xA0, 0xAA, 0xA5, 0xF2). To alter the I2C address, these commands must be delivered in the correct order, and no other commands may be presented in the midst of the sequence. It must be delivered to the command address at 0, resulting in four separate I2C bus write transactions. If you fail to tag the sonar with its address after you have finished, turn it on without sending any commands. The address of SRF02 will be flashed on the LED. Its address is indicated by a single lengthy flash followed by a series of shorter lights. Sending a command to the SRF02 stops the flashing immediately [2].

Address		Long Flash	Short flashes
Decimal	Hex		
224	E0	1	0
226	E2	1	1
228	E4	1	2
230	E6	1	3
232	E8	1	4
234	EA	1	5
236	EC	1	6
238	EE	1	7
240	F0	1	8
242	F2	1	9
244	F4	1	10
246	F6	1	11
248	F8	1	12
250	FA	1	13
252	FC	1	14
254	FE	1	15

Table 2.5 I2C address of the SRF02 [2]

A bus collision will occur if more than one sonar is tuned to the same address, with many unforeseen consequences.

Note that the SRF02 only has one module address recorded. The comparable Serial Mode address will change if you alter it:

0xE0, 0xE2, 0xE4, 0xE6, 0xE8, 0xEA, 0xEC, 0xEE, 0xF0, 0xF2, 0xF4, 0xF6, 0xF8, 0xFA, 0xFC, 0xFE, 0xF2, 0xF4, 0xF6, 0xF8, 0xFA, 0xFC, addresses that use the I2C protocol

0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x10A, 0x0B, 0x0C, 0x0D, 0x Serial addresses that are equivalent [2].

2.4.10 Autotune

The SRF02 does not need to be calibrated by the user. You turn on your computer and make use of the SRF02.

On the inside, tuning cycles operate in the background on regular basis. The transducer continues to ring for a length of time after the ultrasonic burst has been transmitted. This ringing is what restricts the SRF02's closest range of measurement, which is generally 22cm of non-detection. The SRF02 can detect the ringing time of the transducer and adjust its sensing threshold to match it, giving it the best possible performance. The recognition threshold is ready to 28cm when the device is turned on (11). The transducer ring is promptly backed up by the tuning algorithms. This happens in no more than a half-second at a full scan pace, after 5-6 ranging cycles. The tuning algorithms continue to monitor the transducer, after that, it is backing up the threshold as far as possible or erases it out a little if necessary. Nevertheless, it runs in the backdrop and has no blow on the scan speed [2].

If necessary, registers 4 and 5 can be read to determine the minimum range. This value is returned in the same units as the range: uS, cm, or inches. You can also re-tune the SRF02 by typing command 96, however, you should disregard this instruction. During our testing, it is used.

2.5 ADC and FFT

2.5.1 Theoretical Background

When it comes to working with digital systems that work or communicate with real-time signals Analog-to-Digital Converters are critical components. With the IoT(Internet of things) rapidly becoming part of everyday life, these digital devices must be able to interpret real-world time signals to reliably offer the crucial information. In this section, we will look at how ADCs and the theory behind them.

2.5.2 How ADCs Work

An Analog signal is a signal that has a constant progression with uninterrupted values in the real world even though there are some cases where it can be finite, light, temperature, sound, and motions are all examples of these types of signals. In the process, a signal is conked out into progressions that rely on time serial or sampling rate while digital signals are characterized discrete values in a sequence. The best way to illustrate this is with a diagram. Figure 2.18 illustrates the difference between digital and analog signals [3].

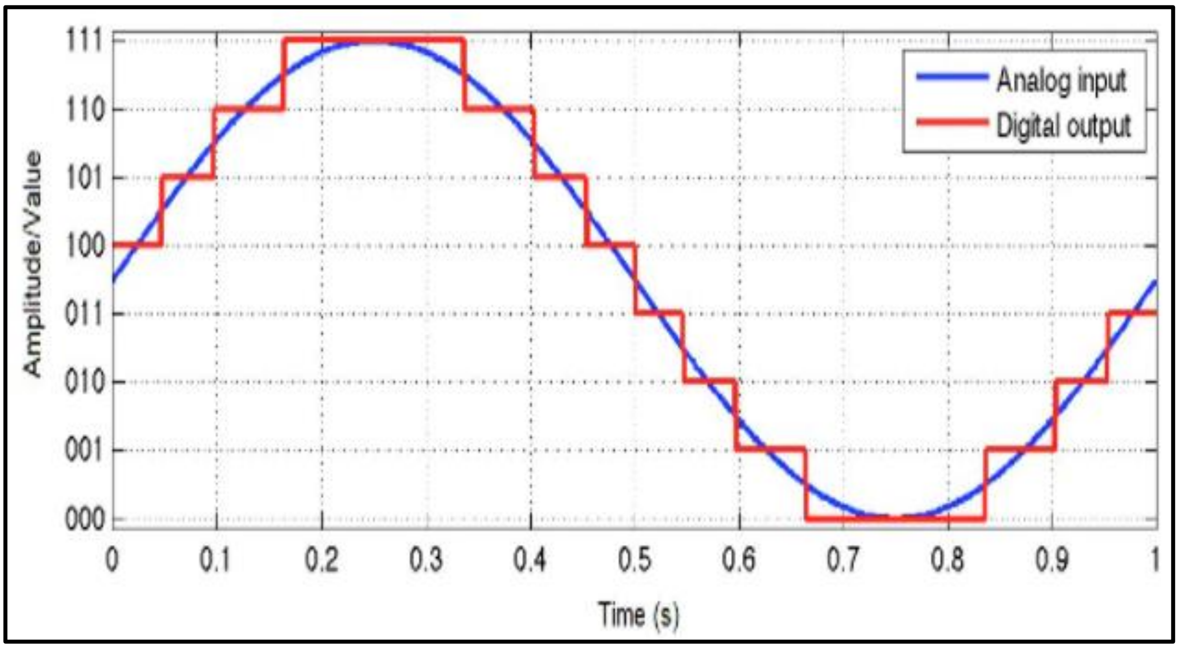


Figure 2.18 Analog and Digital Signal [3]

Moreover, values cannot be read by microcontrollers unless they are digital data. This is due to the fact that microcontrollers only see levels of voltage which are determined by analog to digital converters resolution and the system voltage.

While translating analog signal to digital signal, an analog-digital converter run on a set of the method. They mock-up the signal, then enumerate it to set up the resolution of the signals before dispatching it to the system in order to interpret the digital signals. The sample resolution along with rate are the two significant features of ADC [3].

2.5.3 Sampling Rate/Frequency

The sampling rate, also identified as sampling frequency, of an ADC, is proportional to its speed. The sampling rate is expressed in “sample per second” whereas the elements are in SPS or Hz if one is utilizing sampling frequency. It simply relates to the number of data points or samples taken in a second. The higher the frequency the ADC can handle the more samples it takes [3].

One of the important equations of the sample rate is given below:

$$F_s = \frac{1}{T} \tag{2-6}$$

Where

Fs= frequency or rate of sample

T= Sample time period or interval between samplings.

In Figure 2.18, for example, Fs appears to be 20 S/s, but the T appears to be 50ms. Despite the lower sample rate, the signal resembled that of the original analog signal. It is due to the original signal frequency being 1Hz which means that the frequency rate was high-quality enough to renovate a similar signal.

Now, what turns out if the sample rate is much slower? One may wonder. It is crucial to recognize the rate of ADCs sampling as one will have to be acquainted with if this will produce aliasing or not. Because of the sampling, when a digital representation or a signal is built, it is not the same as the original image or signal [3].

The ADC would not be capable to recreate the original analog signals if the rate of sampling is sluggish and the signal frequency is high, causing the system to read inaccurate data. Figure 2.19 is an excellent illustration.

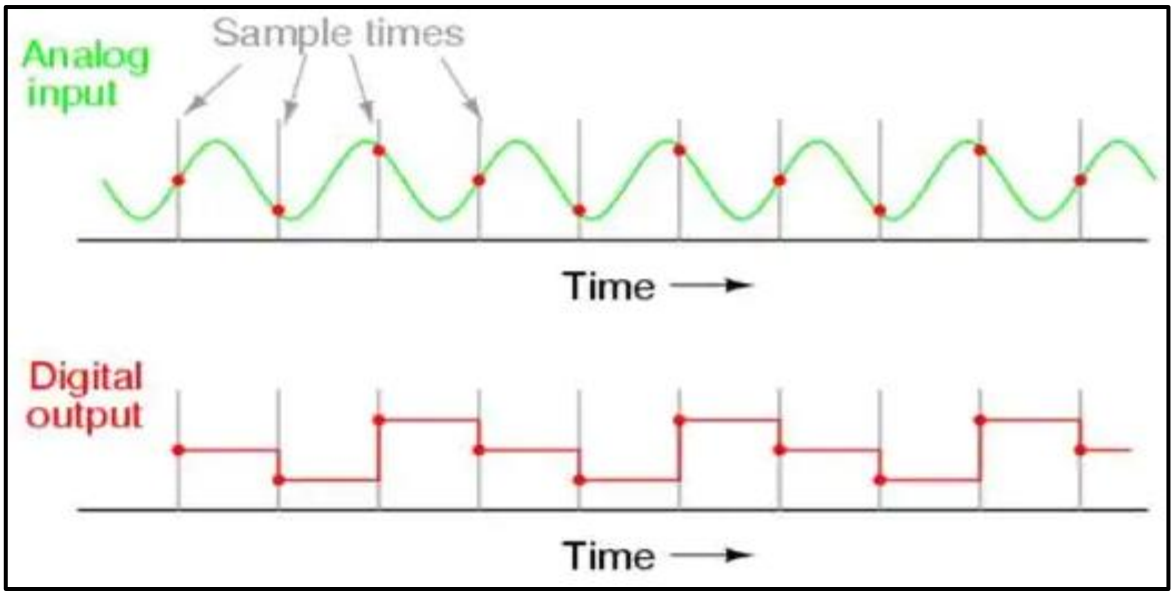


Figure 2.19 Aliasing Example [3]

In this example, it is easy to see where the sampling occurs in the analog input signal. Seeing that the sampling rate is insufficient to keep abreast with the analog signal, the yield of the digital signal is not even near the original signal. As a result of aliasing, the digital system will no longer be capable to see the entire picture of the analog stream.

The Nyquist theorem is a good rule of thumb to use when determining whether or not aliasing will occur. To replicate the original digital signal, according to the theorem, the sampling frequency or rate must be at least twice as high as the highest frequency in the signal. The Nyquist is calculated using the following equation [3].

$$F_{Nyquist} = 2F_{max}$$

$$F_{Nyquist} = Nyquist\ Frequency$$

$$F_{max} = the\ maximum\ frequency\ appeared\ in\ signal$$

Suppose that, if the signal an individual feeding into the digital system has the greatest frequency of 100KHz, your sampling rate of ADC must be at least 200KS/s. this will allow the original signal to be successfully reconstructed.

It is also worth noting that outside noise can bring an unexpectedly high frequency into an analog signal, causing the signal to be disrupted since the sample could not manage the frequency of additional noise. Before starting the ADC and the sampling, it is always a good idea to include a low pass filter (Anti-aliasing filter) to avoid unexpected high frequencies from entering the system [3].

2.5.4 ADC Resolution

The precision of ADC could be linked to its resolution. The bit length of ADC can be used to determine its resolution. Figure 2.20 depicts a simple example of how it evaluates the digital signal and produces a more accurate signal. The 1bit only has two levels as you can see. The levels grow as the bit length is increased, assembling the signal more similar to the original analog signal [3].

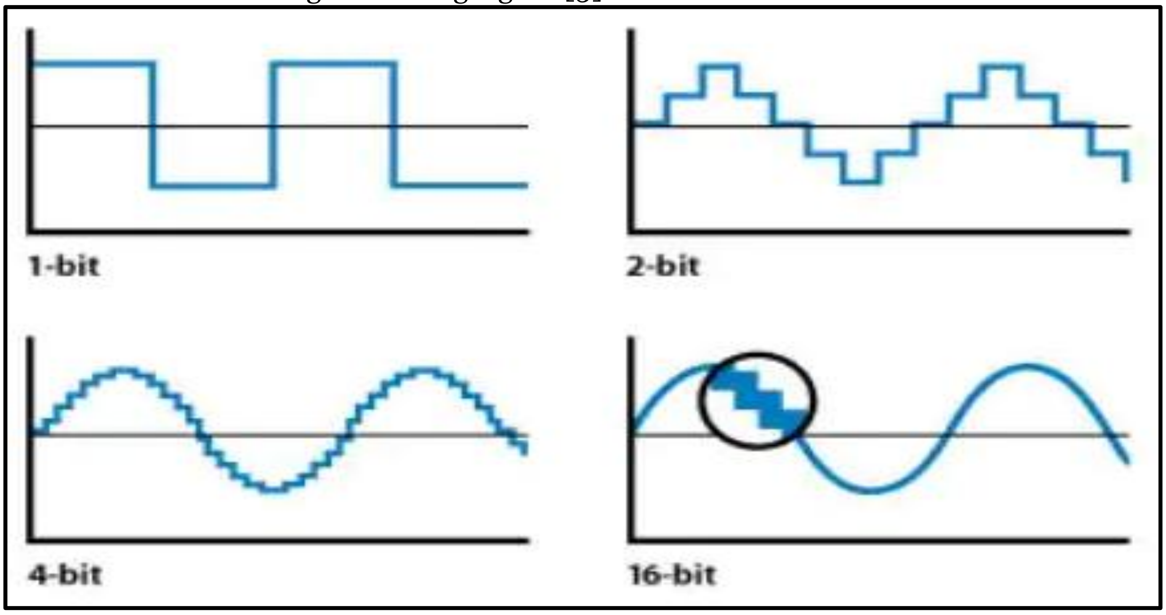


Figure 2.20 Effects of Resolution on digital signals [3]

It is vital to know if you require an appropriate voltage level for your system to read. Both the bit length and reference voltage influence the resolution. These equations assist you in determining the entire signal resolution you are attempting to put in in voltage terms.

$$Step\ Size = \frac{V_{Ref}}{N}$$

$$V_{Ref} = the\ reference\ voltage\ or\ range\ of\ voltages$$

$$N = ADC\ total\ level\ size$$

$$Step\ Size = each\ one\ level\ resolution\ in\ voltage\ r = terms$$

The following equation is used for finding the size of N.

$$N = 2n$$

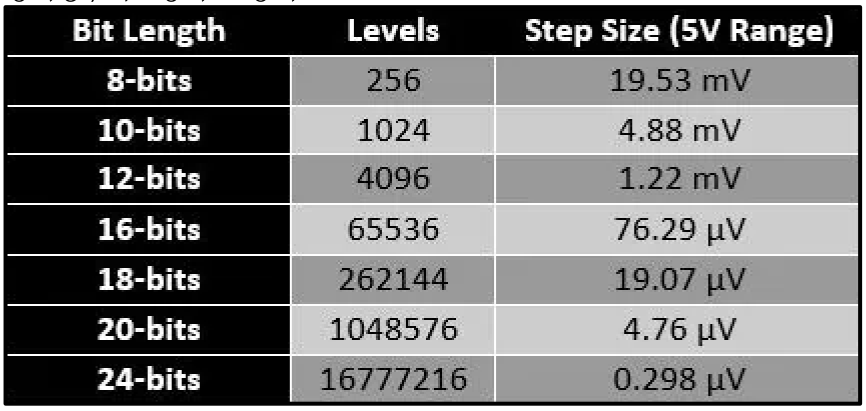
$$where \quad n = bit\ size$$

Let us, for an instance, say that you need to interpret a sine wave with a voltage range of 5 volts. The ADC has a 12-bit size. When you plug twelve (12) to n into equation 4, the result is 4096. With that information and a 5v voltage reference, you will have.

$$\frac{5V}{4096}\ (step\ size)$$

Moreover, you will discover that the step size is roughly 1.22 mV or 0.00122V. This is precisely because the digital system will be competent to detect voltage alteration with a precision of 1.22 mV [3].

If the bit length of ADC was only two bits, for example, the accurateness will drop accordingly to 1.24V which is terrible because it can only tell the system for voltage level that is 5V, 3.7V, 2.5V, 1.25V, and 0V.



Bit Length	Levels	Step Size (5V Range)
8-bits	256	19.53 mV
10-bits	1024	4.88 mV
12-bits	4096	1.22 mV
16-bits	65536	76.29 µV
18-bits	262144	19.07 µV
20-bits	1048576	4.76 µV
24-bits	16777216	0.298 µV

Table 2.6 Bit Length and Number of levels [3]

Table 2.6 depicts the most common bit lengths and the number of levels they contain. It, in addition, indicates what a step size of 5V reference will be. As the bit length grows, one can see how accurate it becomes. Understanding the resolution of ADC and sample rates demonstrates how indispensable it is to grasp these parameters [3].

2.6 Fast Fourier Transformation FFT - Basics

Simply listen to analyze a Fast Fourier Transform. An individual ear does automatic and involuntary calculations that would take days of mathematical study for the mind to complete. The ear creates a transform by transforming sound—pressure waves flowing across space and time—hooked on a spectrum, which is a description of sound as a collection of volumes at different pitches. The brain subsequently converts this information into audible sensations [4].

The Fast Fourier transform (FFT) is an important measurement technique in the realm of acoustics and audio measurement. It breaks down a signal into its numerous spectral components and, as a result, frequency information. FFTs are engaged in machine or value control, conditioned monitoring, and system defect analysis. This part discuss how FFTs works as well as the factors involved and how they affect the measurement outcome.

In strict terms, the FFT is an optimal algorithm for implementing the Discrete Fourier Transformation. Over a period of time, a signal is sampled and split into its frequency components. These components are made up of single sinusoidal oscillations at various frequencies, each with its own amplitude and phase. The diagram below shows how to make this change. The signal has 3 separate overlooking frequencies across the period being observed [5].

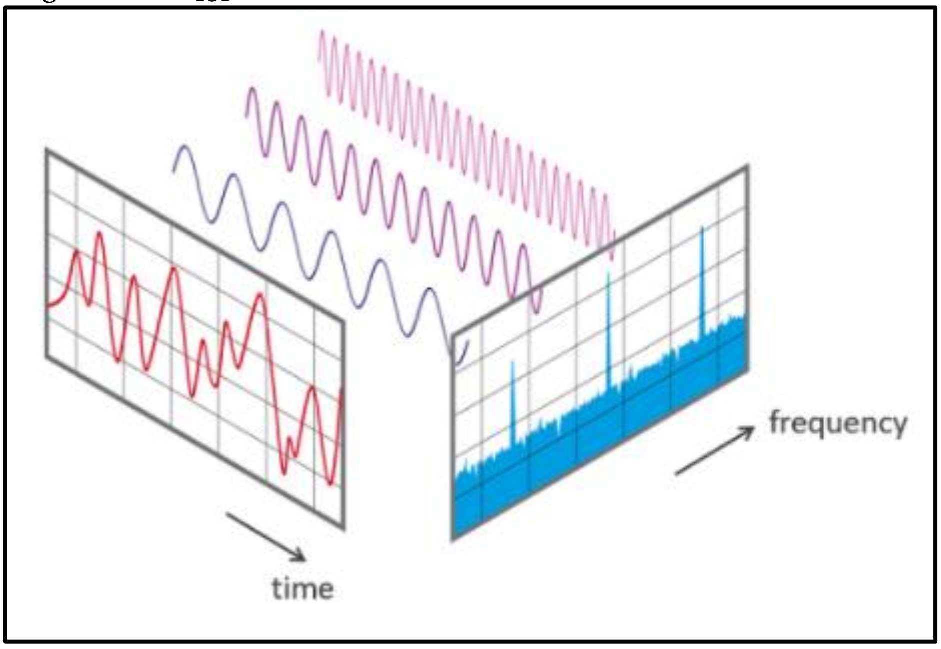


Figure 2.21 Signal in Time and Frequency Domain [5]

2.6.1 Step By Step

The initial stage involves scanning a segment of the signal and storing it in memory for further processing. There are two parameters that are important.

- The sampling frequency, also known as the sampling rate, is the average number of samples per second, which is 48KHz in this case.
- The quantity of the sample chosen, and block length is always an integer power of 2. For example, 2 power 10 is equal to 1024 samples.

Further measurement parameters can be calculated using the two basic parameters BL and fs.

Bandwidth fn(= Nquist frequency). This value is the potential highest frequency that the FFT can determine.

$$fn = \frac{fs}{2}$$

For an instance, components of frequency up to 24 kHz can theoretically be computed with a sample rate of 48 kHz. Due to analog factors, the practically feasible value in an analog system is usually slightly lower, for an instance, at twenty kHz [5].

Measurement duration D. The Sampling rate of BL and fs determines measurement duration.

$$D = \frac{BL}{fs}$$

At fs = 48 kHz and BL = 1024, this gives way to 1024/48000 Hz = 21.33 ms

Resolution Of Frequency (df). The resolution of the frequency is the distance between two measurement outcomes in terms of frequency

$$df = \frac{fs}{BL}$$

At fs = 48 kHz and BL = 1024, this gives a df of 48000 Hz / 1024 = 46.88 Hz.

To put into practice, the frequency of sampling is frequently a system-supplied variable. However, the measurement period and frequency resolution can be labeled by selecting the BL. The subsequent is true [5].

- Because of the short BL, measurement repetitions are quick, and the frequency resolution is coarse.
- With a long BL, measuring repetitions are slower and frequency resolution finer.

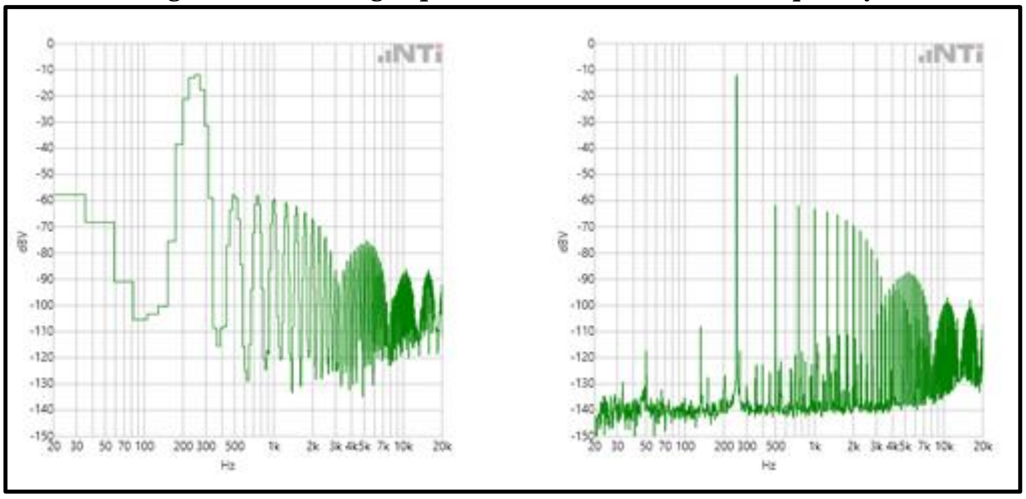


Figure 2.22 Signal Representation with small and large block length [5]

2.6.2 To the infinity

The sample signaled segment is assumed to be repeated for an endless amount of time in the Fourier transformation. Two interferences can be drawn from this.

1. The FFTs can only be used with periodic signals.
2. The signal segment to be sampled must have an even number of periods.

As can be observed, condition two would only apply to a small number of signals. A block of 2^n samples with various values would begin and end the sampling a signal whose frequencies are not part of the df necessary manifold. This causes a smeared FFT spectrum and a jump in the time signal which is also known as leakage [5].

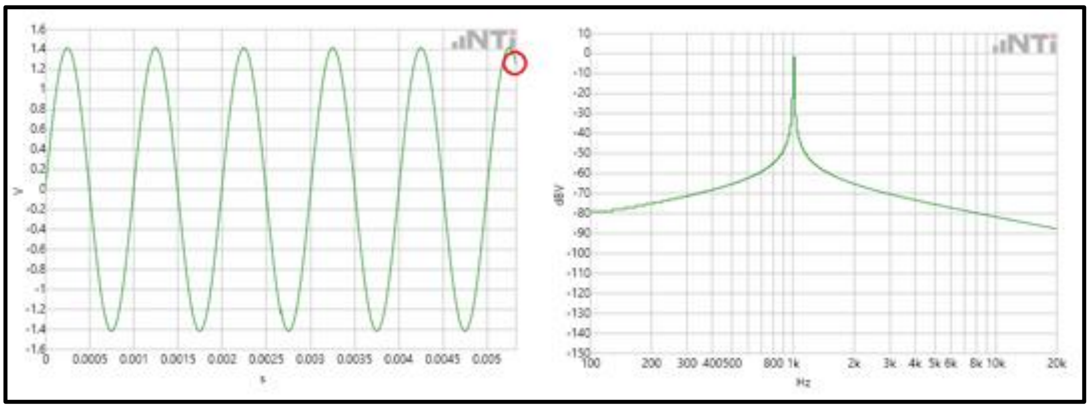


Figure 2.23 smeared spectrum of un-windowed time signal [5]

2.6.3 Windowing

To prevent this smearing, the signal sample is windowed in practice. The sample

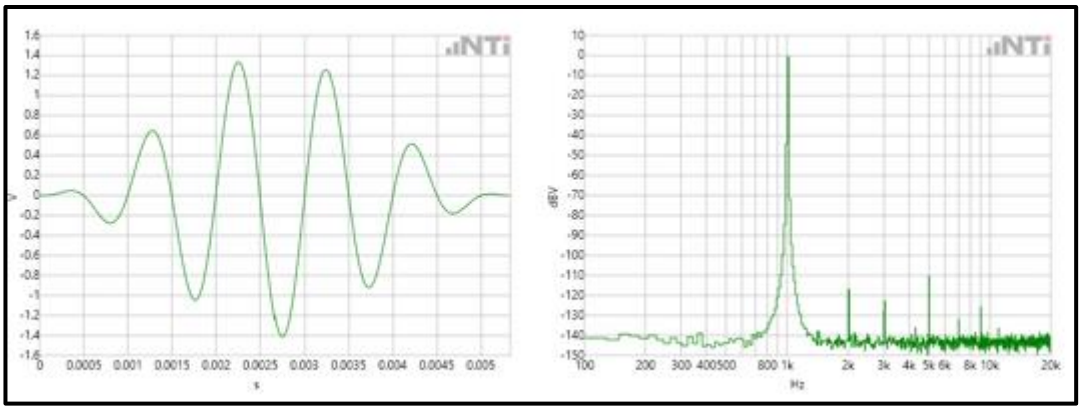


Figure 2.24 Spectrum of the windowed time signal [5]

signal is switched off and on more or less gently using a weighting algorithm. As a result, the sampled and the succeeding windowed signal starts and ends at the same amplitude. Without a hard transition, the sample can be repeated regularly [5].

2.6.4 Nyquist Theorem

Nyquist discovered fundamental criteria in analog signal sampling which inculcates that the fs must be at least twice the maximum signal frequency. For an instance, if a signal with frequencies up to 24KHz needs to be sampled, a rate f sampling of at least 48KHz would be at least necessary [5].But what happens if the system is fed signals that are higher than the Nyquist frequency?

2.6.5 Aliasing

A signal is sampled with a suitable quantity of samples in most cases. The Six kHz for an instance is sampled eight times every cycle with a 48Khz sampling rate, while 12 KHz is only tested 4 times per cycle. Only two samples are available at the Nyquist Frequency. It is still possible to create the signal without losing two or more samples. If there are less than two samples available, artifacts emerge that are not present in the original sampled signal [5].

2.6.6 Mirror frequencies

These abnormalities appear as mirror frequencies in the FFT. When the Nf is surpassed, the signal is echoed at this unreal limit, returning to the usable frequency region. The video below displays an FFT system with a sampling rate of 44.1 kHz. This system receives a 15 kHz to 25 kHz sweep signal [5].

Before scanning, anti-aliasing filters are used to filter out the undesired mirror frequencies. Frequencies exceeding the Nyquist frequency are muted by the filter.

2.6.7 Averaging of Spectra

Capturing many FFT blocks and determining mean values from them is typically helpful in the study of non-periodic signals, such as music or noise. There are two approaches that can be taken:

The traditional mean is as follows: A large number of FFTs are analyzed. In the averaged final result, each outcome is weighted equally. This method works well for measurements that have a set duration.

A fixed number of continuous measurement results are also taken into account here. The weighting, on the other hand, is indirectly proportional to the age of the results. The most previous measurement is taken into account the least, while the good number of recent measurements throws in the most to the average result. whenever the spectrum is constantly observed over a long period, this exponential average is used [5].

2.6.8 Power vs. Peak detector

The number of measurement results can be decoupled as of the FFT block length in modern high-resolution FFT analyzers. This increases the measurement performance time, which is notably noticeable for higher-resolution FFTs. With a 2MB BL, for example, more than 1 million points are no longer required to be computed and embodied. but simply the number required for display, such as 1024. There are two ways to specify the value for each FFT bin:

1. MaxPeak uses the maximum value of the FFT findings. This kind is ideally suited for displaying FFTs visually.
2. The FFT findings are summed and energetically averaged here in Power. When the FFT is utilized for calculation, this is required [5].

2.6.9 Calculations with FFT results

Signals are mostly visualized via FFTs. There are, however, cases in point where FFT outcomes are used in calculations. Simple levels of illustrated frequency bands, for example, can be determined using the RSS (Root Sum Square) technique [5].

Chapter 3 Development

3.1 System overview

To achieve the objective of this thesis work, development is mainly consisting of 4 different tasks as show in Figure 3.1, which are an implementation of modules that are responsible for ultrasonic signal data acquisition, processing the acquired data, classifying on the basis of features obtained and Notifying about its class. In addition, it requires a module that can control the overall workflow that starts from receiving ultrasonic waves to display the result of object recognition based on the received ultrasonic waves.

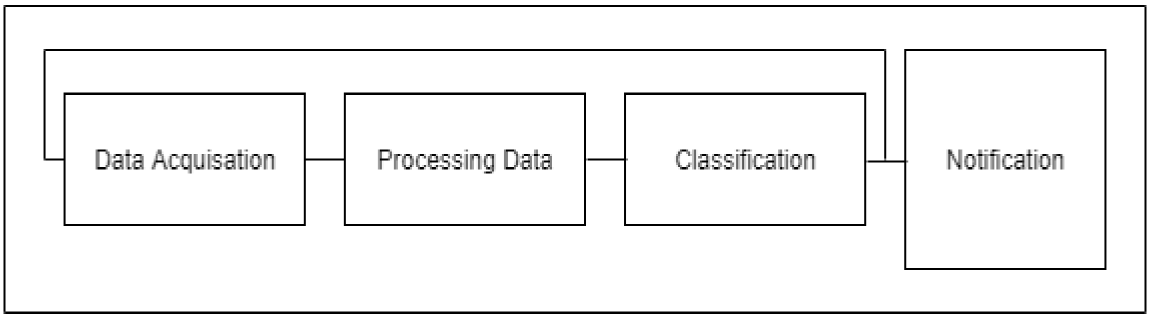


Figure 3.1 Block Diagram of Overall Process

Note that, the sensor device was given developed in previous work by Mr. Michalik, so that the scope of this work has nothing to do with hardware development. Therefore, in this chapter, the implementation of those modules is mainly dealt with. Also, about the remaining hardware setup i.e., connection of the Relays with DIO pins and the 2 lights is done by Mr. Umansky.

Furthermore, the overall process is done repeatedly to come up with the best result by changing the distance and the position of the ultrasonic sensor. Also, different features were tested by adjusting its threshold values to a point where the output will be as to the expectation as possible. The main box classifier is a simple classifier which will look for the 2 features only F1 and F4. Which will be thoroughly discuss in section 3.2.

In the whole process 4 threads are running. One is the main thread which will be starting the other 3 threads each thread is waiting for a trigger and remains in blocked mode. This is done with the help of mutexes.

3.1.1 Main components

The development of the object recognition system is in the scope of this work. Figure 3.1 shows a block diagram of the system. As it shows above, from the data acquisition to object class notification works repeatedly. Figure 3.2 shows the

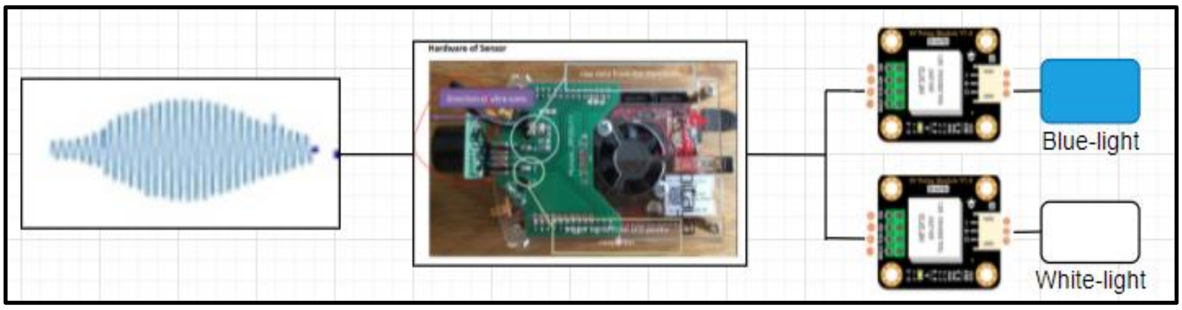


Figure 3.2 Overall architecture of the object recognition system

overall architecture of the system.

As it shows above, the overall architecture of the system is mainly consisting of 4 components, that are ultrasonic signals which are reflected from an object as input data, an instrument called Red Pitaya with an ultrasonic sensor, which will be handling the ADC, FFT, feature extraction and classification part, 2 Gravity HF3FA relays each connected with an LED. The blue LED for hard objects i.e., objects other than humans and white LED for soft objects i.e., humans. In this work, for programming the sensor and establishing a connection between the instrument and the application a UDP client is used which is developed by Mr. Daniel Schäfer.

3.1.2 Overall process flow

Measurements start from the Red Pitaya itself as it is programmed to do so. In 3.8 a detail description is presented on how the Red Pitaya will start automatically make use of Cronjob. The Red Pitaya manages all process flows which is from a sending start command of measurement to the end of giving an output to the DIO pins namely DIO3_P and DIO2_P. The DIO pins in turn operate the relays and the corresponding lights gets ON or OFF depending on the object in front of the sensor.

At first, it sends predefined commands to run the instrument. The instrument is used to conduct and control measurements. The device called Red Pitaya contains an ultrasonic sensor that sends ultrasonic waves and receives its reflected wave from a targeted object. When the device receives the reflected ultrasonic waves from a target object, it digitizes the analog signal, Calculate the Fast Fourier

Transform of the received signal and extract a total of 10 features out of it.

Based on these features a box classifier is developed for the classification of hard and soft objects. So, if we have human in front of the Ultrasonic sensor DIO2_P will turn the relay for 2 seconds and white light will ultimately turn ON. To turn the light ON and OFF relay must operate for 2 seconds. And if we have other than human the white light will turn OFF after the relay trigger for 2 seconds and blue light will turn ON showing that human is not present. In this way we know if any human is present or not. This whole procedure runs repeatedly until it gets an interruption from the experimenter.

3.2 Statistical Analysis of Features

3.2.1 Extracted Features

After the Analog signal has been received back at the Red Pitaya. First it is converted to digital signal to make it compatible for the instrument to work on the digital signal after that the FFT has been calculated for it. Certain features have been calculated from the frequency spectrum of the FFT. Based on these features classification has been done. In our case we will calculate 10 different types of features from the FFT. Which are detailed as follows.

- F1: Mean difference between frequencies of neighbored measurements of center frequency (max. peak), i.e., moving average over 10 frequency differences of neighbored center frequency (difference sum)
- F2: Center frequency (max. peak)
- F3: Mean of F2 over 10 measurements
- F4: Variance of F2 over 10 measurements
- F5: Number of peaks (within frequency range around 39.5 – 40.5 kHz ±500 Hz of F2)
- F6: Mean of F5 over 10 measurements
- F7: Variance of F5 over 10 measurements
- F8: Mean frequency distance of 3 peaks (left of center frequency, center frequency, right of center frequency (max. peak)
- F9: Mean of F8 over 10 measurements
- F10: Variance of F8 over 10 measurements

In the following sections to come a graphical view has been presented for certain features mainly F1, F4 and F10. As these are some of the features on the basis of which we can describe a classification criterion which will ultimately fulfill the Autonomous Illumination control Task in hand. These graphs are presented to make it easier for us to visualize the data in hand.

3.2.2 Difference between Feature 1 from a distance 50cm

The following graph shows the feature F1 extracted which is Mean difference between frequencies of neighbored measurements of center frequency (max. peak), i.e., moving average over 10 frequency differences of neighbored center frequency. So, we just show here a data samples of only 490 samples but in reality, the sensor is constantly taking data every 100ms (milli second). We are classifying between the classes of Human, Empty Seat and Object based on run time features extracted. Here we are not training our model like most other neural networks. This is a simple classification task.

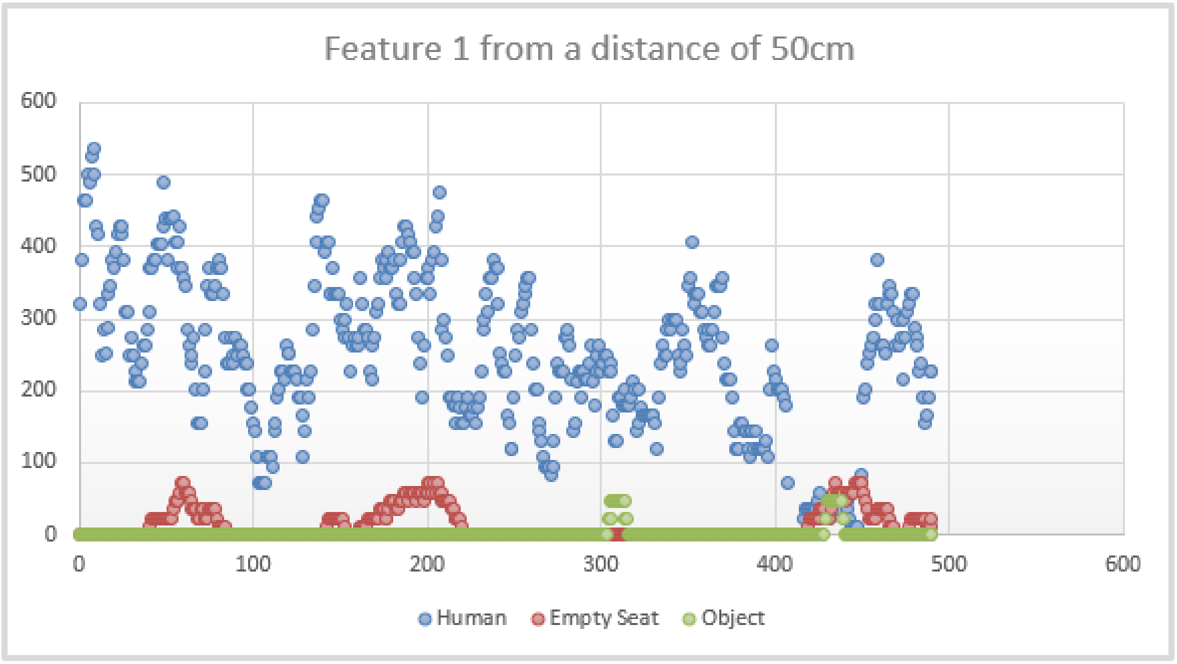


Figure 3.3 Statistical Analysis of F1 from a distance of 50cm

In this case we maintained a distance of 50cm from the sonic sensor SRF02 to analyze the results.

3.2.3 Difference between Feature 4 from a distance of 50cm

The following graph shows the feature F4 extracted which is Variance of F2 over 10 measurements and F2 is the peak frequency. So, we just show here a data samples of only 490 samples but in reality, the sensor is constantly taking data every 100ms (milli second). We are classifying between the classes of Human, Empty Seat and Object based on run time features extracted. Here we are not training our model like most other neural networks. This is a simple classification task.

In this case we maintained a distance of 50cm from the sonic sensor SRF02 to analyze the results. Bear in mind as discuss previously that hard object will return almost 100percent of the signal while soft objects like clothes or human will absorb an amount

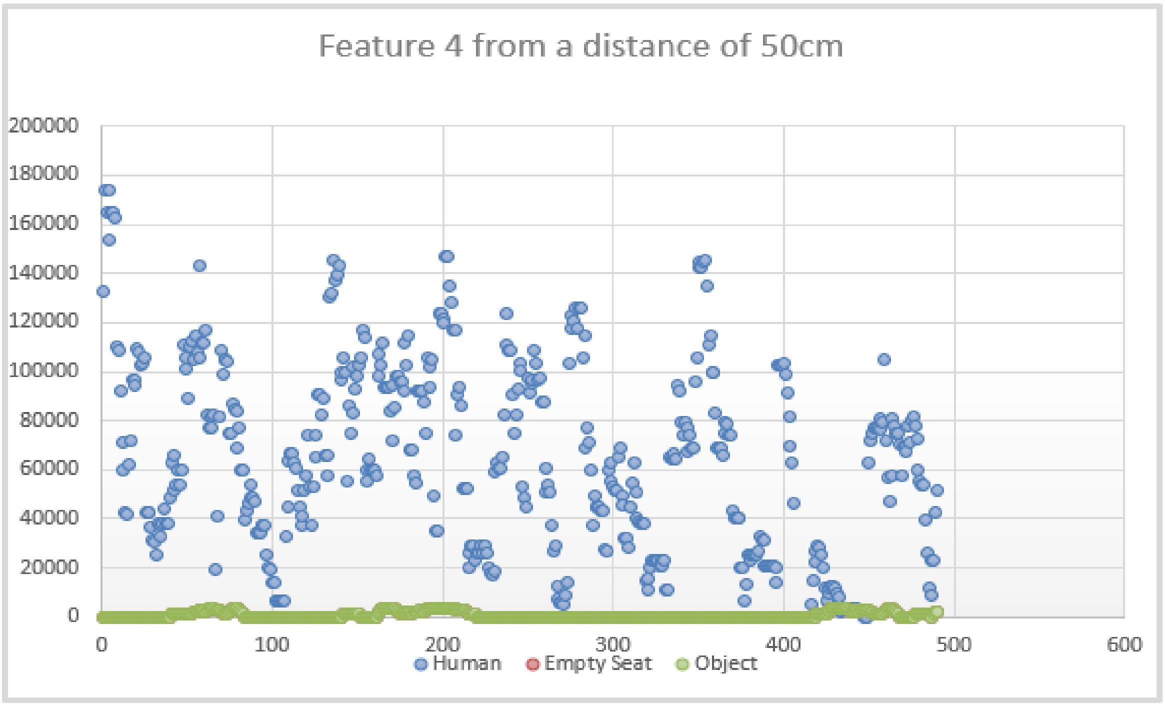


Figure 3.4 Statistical Analysis of F4 from a distance of 50cm

of the ultrasonic signals.

3.2.4 Difference between Feature 10 from a distance of 50cm

The following graph shows the feature F10 extracted which is Variance of F8 over 10 measurements. Where F3 is Mean frequency distance of 3 peaks (left of center frequency, center frequency, right of center frequency (max. peak) So, we just show here a data samples of only 490 samples but in reality, the sensor is constantly taking data every 100ms (milli second). We are classifying between the classes of Human, Empty Seat and Object based on run time features extracted. Here we are not training our model like most other neural networks. This is a simple classification task.

In Figure 3.5 we maintained a distance of 50cm from the sonic sensor SRF02 to analyze the results. From the output it is obvious that feature 10 is not a good option to choose instead we can work with feature 1 and feature 4. Because the 2 object classes are quite apart in those cases.

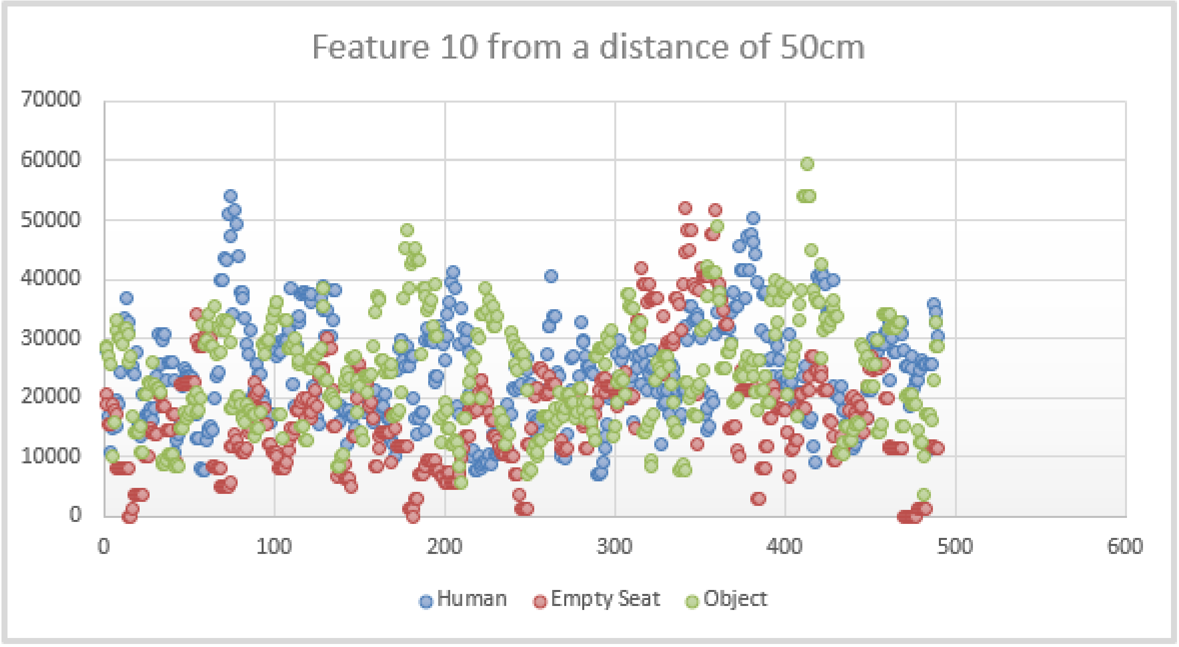


Figure 3.5 Statistical Analysis of F10 from a distance of 50cm

3.2.5 Difference between Feature 1 from a distance of 70cm

The following graph shows the feature F1 extracted which is Mean difference between frequencies of neighbored measurements of center frequency (max. peak), i.e., moving average over 10 frequency differences of neighbored center frequency. So, we just show here a data samples of only 490 samples but in reality, the sensor is constantly taking data every 100ms (milli second). We are classifying between the classes of Human, Empty Seat and Object based on run time features extracted. Here we are not training our model like most other neural networks. This is a simple classification task.

In this case we maintained a distance of 70cm from the sonic sensor SRF02 to analyze the results. As we can see from the Figure 3.6 for about 98 percent of the time we have a clear border between the Object class 1 and Object class 2 around the value of 100. The 2% exception account for many reasons discuss in Ultrasonic Sensing chapter. So, by selecting a threshold value of about 100 will most probably better results for object recognition. Also, from this we can deduce that as the distance increased by 20cm the results are better. If we have a distance of 1 meter between the object and the ultrasonic sensor it improves further.

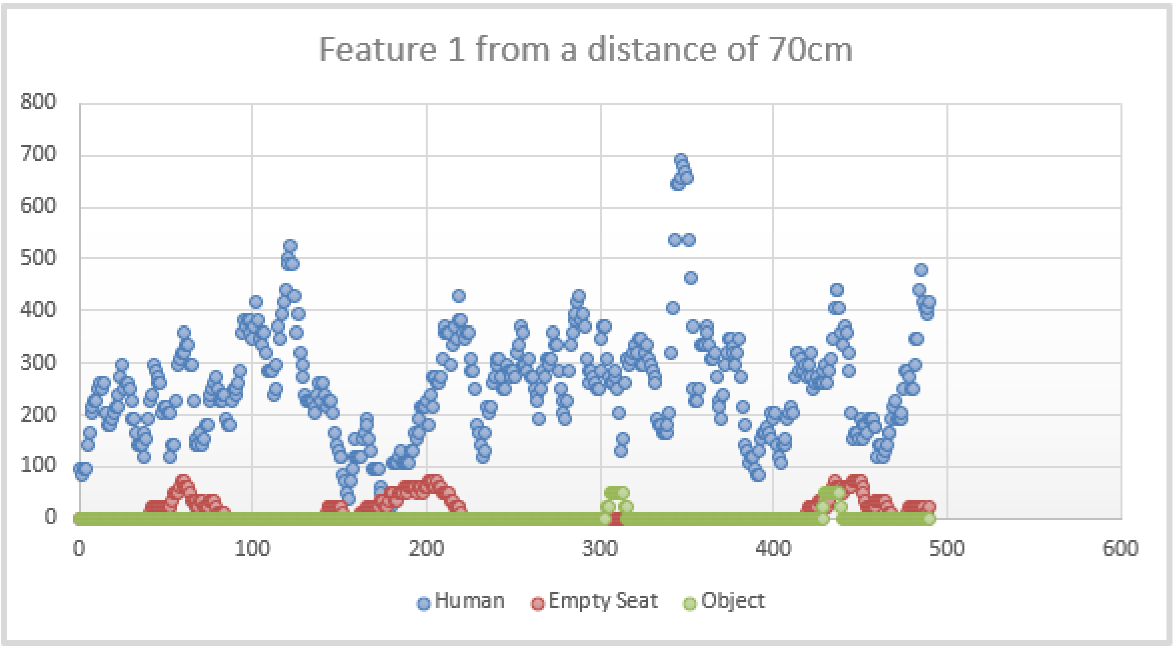


Figure 3.6 Statistical Analysis of F1 from a distance of 70cm

3.2.6 Difference between Feature 4 from a distance of 70cm

The following graph shows the feature F4 extracted which is Variance of F2 over 10 measurements and F2 is the peak frequency. So, we just show here a data samples of only 490 samples but in reality, the sensor is constantly taking data every 100ms (milli second). We are classifying between the classes of Human, Empty Seat and Object based on run time features extracted. Here we are not training our model like most other neural networks. This is a simple classification task.

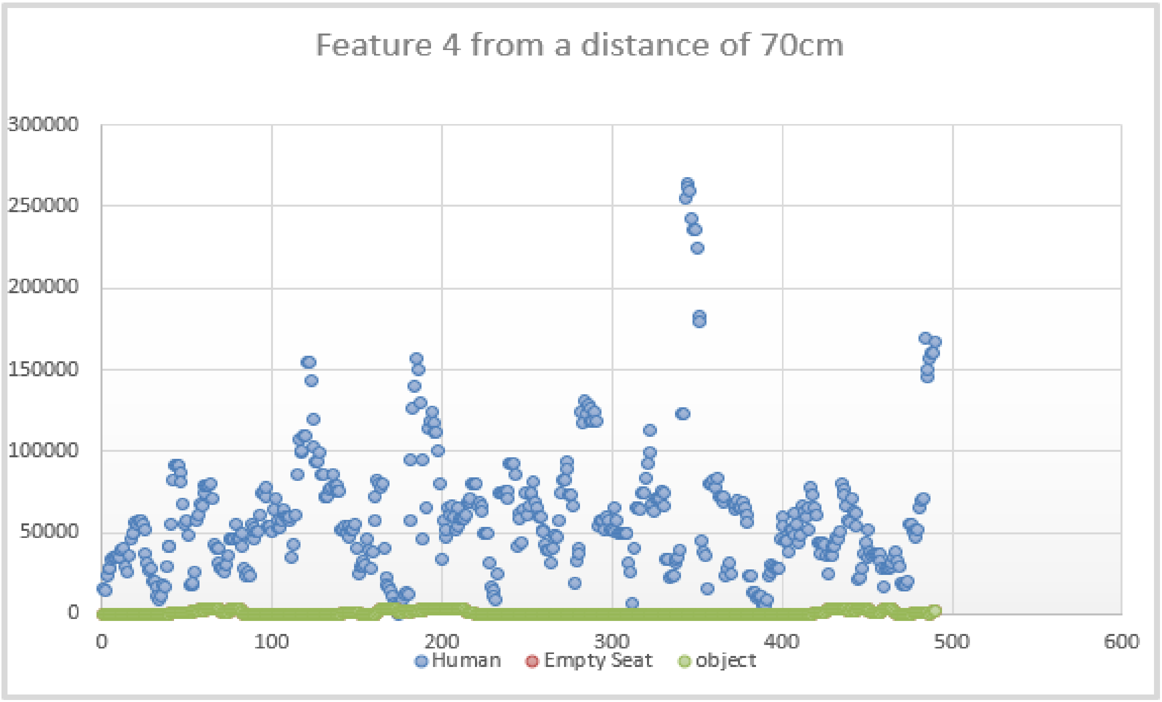


Figure 3.7 Statistical Analysis of F4 from a distance of 70cm

In this case we maintained a distance of 70cm from the sonic sensor SRF02 to analyze the results.

3.2.7 Difference between Feature 10 from a distance of 70cm

The following graph shows the feature F10 extracted which is Variance of F8 over 10 measurements. Where F3 is Mean frequency distance of 3 peaks (left of center frequency, center frequency, right of center frequency (max. peak) So, we just show here a data samples of only 490 samples but in reality, the sensor is constantly taking data every 100ms (milli second). We are classifying between the classes of Human, Empty Seat and Object based on run time features extracted. Here we are not training our model like most other neural networks. This is a simple classification task.

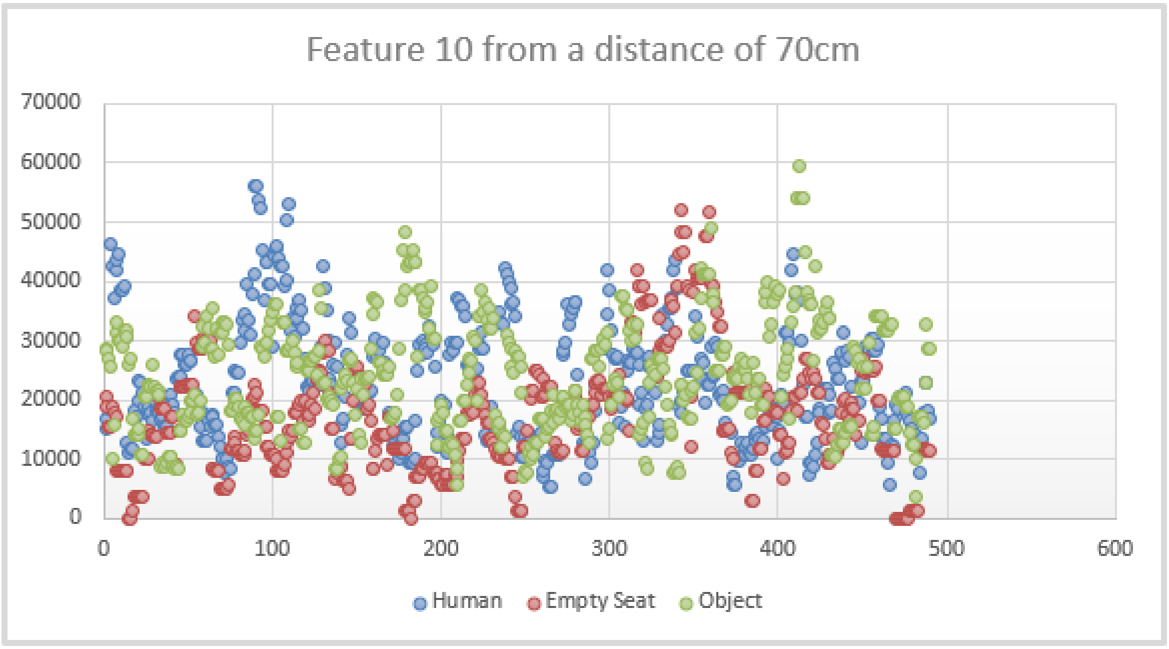


Figure 3.8 Statistical Analysis of F10 from a distance of 70cm

In this case we maintained a distance of 70cm from the sonic sensor SRF02 to analyze the results. From the output it is obvious that feature 10 is not a good option to choose instead we can work with feature 1 and feature 4. Because the 2 object classes are quite apart in those cases.

3.3 Used Software

All used software for the project is provided under the GNU General public license. The RP runs a Linux operating system so there a GCC (GNU compiler collection) is used to compile C(++) code. The provided makefile in the directory is created by Daniel Schäfer but offered as a part of the software and distributed under the GNU General license.

To read, write, modify and save any kind of software, the following programs are used in this project:

1. Notepad++
2. WinSCP
3. Putty (portable)
4. MS Visual Studio 2019

3.3.1 WinSCP

It works on windows platforms and is also available in different install-packages for windows. It uses the "secure copy protocol" to provide a remote view of a PC in the same network. The SCP server on the RP is the counter part for the client version on the windows machine.

3.3.2 Notepad++

It is a very powerful software which can not only be used as a simple text-editor. The principal function based on the file-extension which is opened. In case of an opened C-file the extension ".c" is used, so the Notepad++ starts new text-parsing. Different keywords are highlighted to get the feeling of an IDE. The precompiler defines have another color than "void" or the variable declaration "uint8_t". The programmer is also able to see the hierarchy of different used brackets for example the "if" or "switch" cases.

3.3.3 Putty (portable)

It is used to establish connections like SSH, Telnet, FTP and RAW formats. The core idea is to have at least one single program used as an "all in one" application. Only one part out of more than 8 functions is used, the SSH connection. The RP offers an SSH connection as a server on default. The remote Secure Shell can be entered via Telnet.

3.3.4 MS Visual Studio 2019

It is used to create a C# project with all necessary software bundles needed for communication to / with RP.

3.3.5 Software Core

All Software for RP is written in C. The software for Microsoft machines is written in C#. On RP the software runs as a daemon in background. Four threads are used to let the binary code run on the system. With the first thread main is started and three new threads are created.

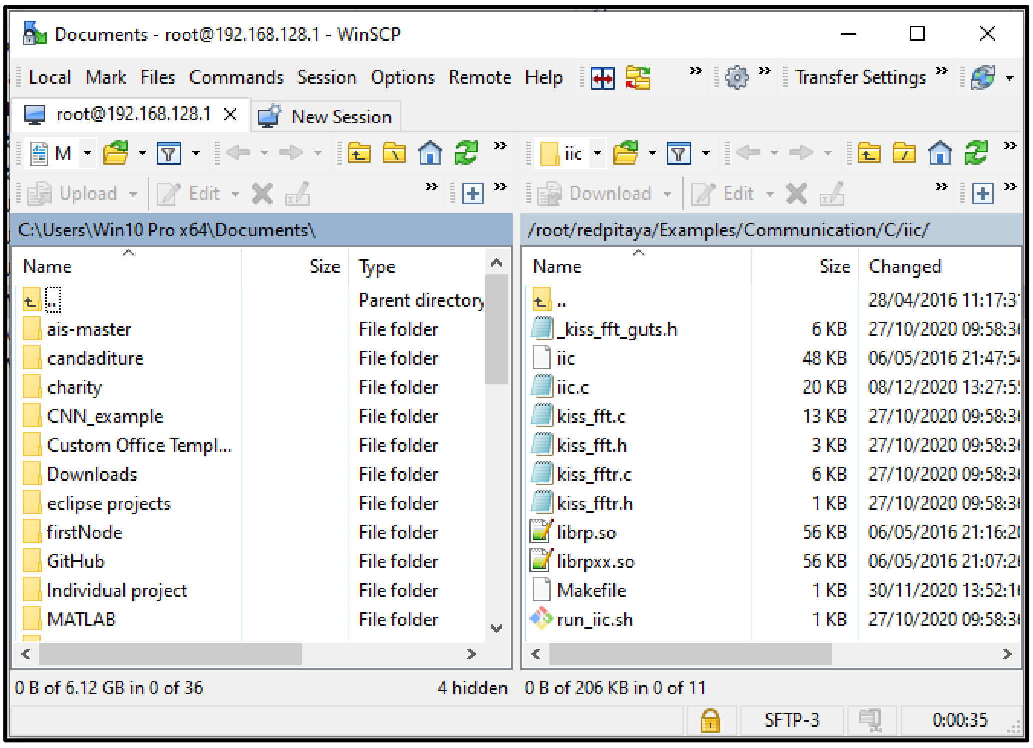
The second thread is listening on port 61231 by a UDP server. The thread is in blocked mode until a message comes in via UPD. The message is parsed and gives a signal to the third thread to run.

The third thread which is Measure Thread is waiting for a signal from the server thread, which is able to deliver a command and a value. For example, “-a 1” means: “start ADC output”.

The fourth thread will start inside the third thread every time human, or no human is detected by the sensor. This thread is in blocked mode until a message comes in and start the White or Blue LED depending on the class of the Object. Inside the LedThread the current state of the LEDs is maintained. So, every time the thread is called from the MeasureThread the LedThread is able to keep track of the last status of the LEDs.

3.3.6 Change the RP software

If the SW of the RP should be change, open an SCP connection by using WinSCP and navigate to the folder “/root/iic”. IP address of the RP has to be entered and the port 22 is used. For user authentication user “root” with PW “root” is used. The folder “iic/src” contains all necessary code, which can be changed and tested.


Documents - root@192.168.128.1 - WinSCP

C:\Users\Win10 Pro x64\Documents\

Name	Size	Type
..		Parent directory
ais-master		File folder
candaditure		File folder
charity		File folder
CNN_example		File folder
Custom Office Templ...		File folder
Downloads		File folder
eclipse projects		File folder
firstNode		File folder
GitHub		File folder
Individual project		File folder
MATLAB		File folder

/root/redpitaya/Examples/Communication/C/iic/

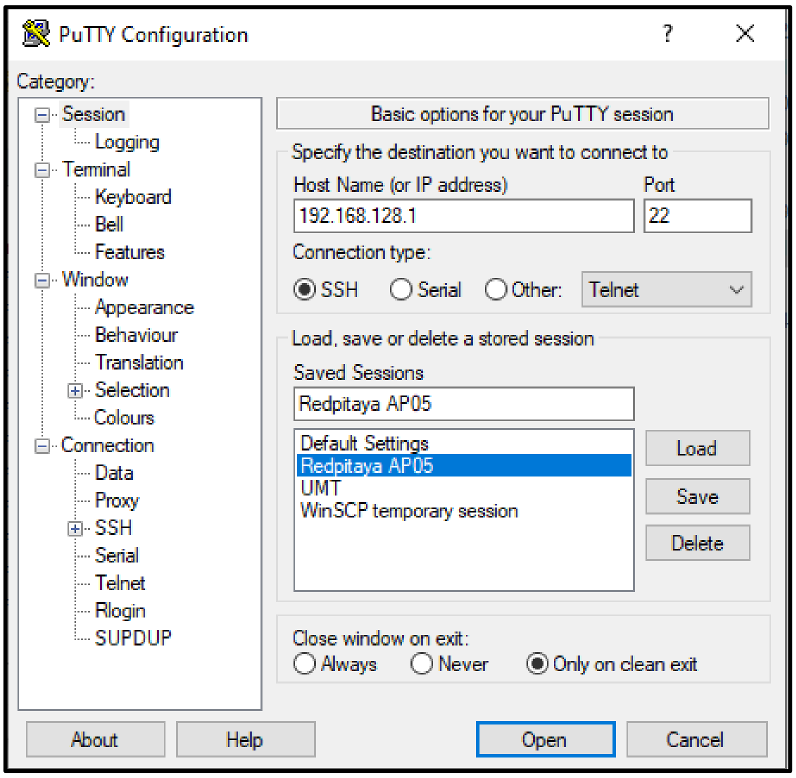
Name	Size	Changed
..		28/04/2016 11:17:3
_kiss_fft_guts.h	6 KB	27/10/2020 09:58:3
iic	48 KB	06/05/2016 21:47:5
iic.c	20 KB	08/12/2020 13:27:5
kiss_fft.c	13 KB	27/10/2020 09:58:3
kiss_fft.h	3 KB	27/10/2020 09:58:3
kiss_fftr.c	6 KB	27/10/2020 09:58:3
kiss_fftr.h	1 KB	27/10/2020 09:58:3
librp.so	56 KB	06/05/2016 21:16:2
librpxx.so	56 KB	06/05/2016 21:07:2
Makefile	1 KB	30/11/2020 13:52:1
run_iic.sh	1 KB	27/10/2020 09:58:3

0 B of 6.12 GB in 0 of 36   4 hidden   0 B of 206 KB in 0 of 11

SFTP-3   0:00:35

Figure 3.10 WinSCP Connection to RP

In parallel a shell connection has to be opened by using a program similar to “putty”. Open a connection by also using user “root” and PW “root” to enter remote command shell (SSH).


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.128.1   Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

Load, save or delete a stored session

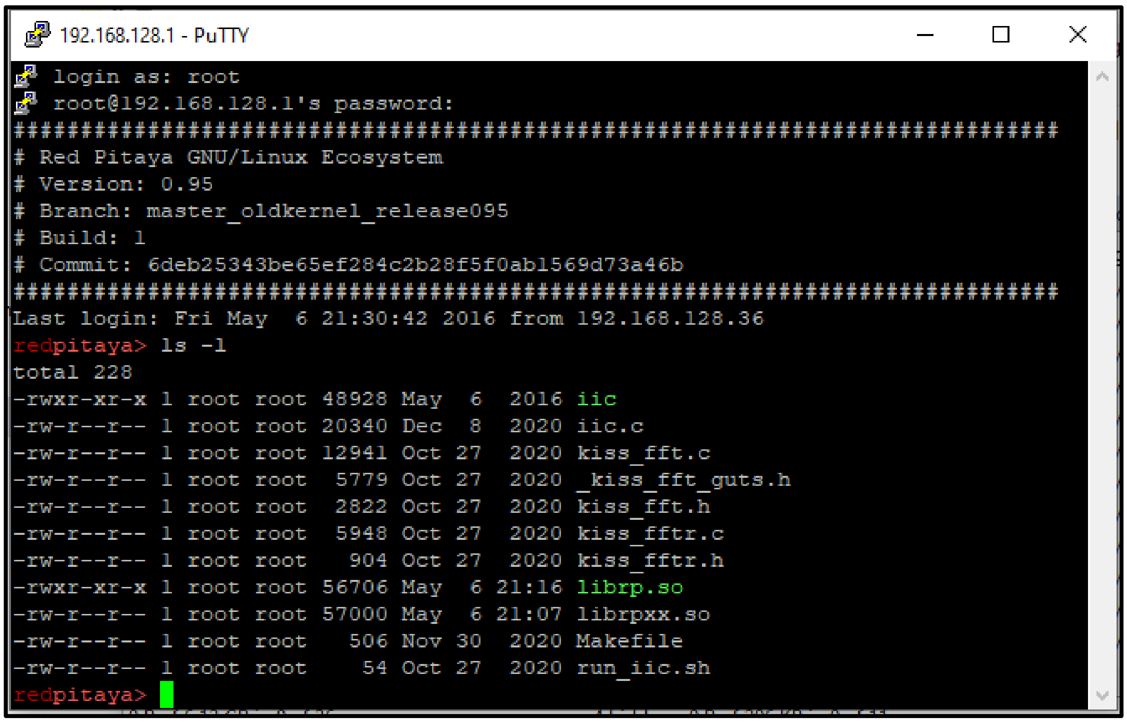
Saved Sessions: Redpitaya AP05

Default Settings
Redpitaya AP05
UMT
WinSCP temporary session

Close window on exit: Always Never [✓ Only on clean exit]

Figure 3.9 Putty Configuration

If the command “ls -l” is executed, the content of the current folder is shown.



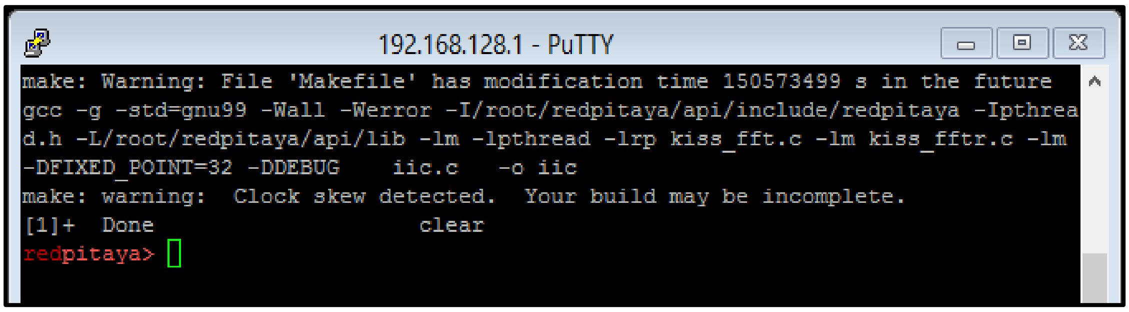
192.168.128.1 - PuTTY

```
login as: root
root@192.168.128.1's password:
##############################################################################
# Red Pitaya GNU/Linux Ecosystem
# Version: 0.95
# Branch: master_oldkernel_release095
# Build: 1
# Commit: 6deb25343be65ef284c2b28f5f0ab1569d73a46b
##############################################################################
Last login: Fri May  6 21:30:42 2016 from 192.168.128.36
redpitaya> ls -l
total 228
-rwxr-xr-x 1 root root 48928 May  6  2016 iic
-rw-r--r-- 1 root root 20340 Dec  8  2020 iic.c
-rw-r--r-- 1 root root 12941 Oct 27  2020 kiss_fft.c
-rw-r--r-- 1 root root  5779 Oct 27  2020 _kiss_fft_guts.h
-rw-r--r-- 1 root root  2822 Oct 27  2020 kiss_fft.h
-rw-r--r-- 1 root root  5948 Oct 27  2020 kiss_fftr.c
-rw-r--r-- 1 root root   904 Oct 27  2020 kiss_fftr.h
-rwxr-xr-x 1 root root 56706 May  6 21:16 librp.so
-rw-r--r-- 1 root root 57000 May  6 21:07 librpxx.so
-rw-r--r-- 1 root root   506 Nov 30  2020 Makefile
-rw-r--r-- 1 root root    54 Oct 27  2020 run_iic.sh
redpitaya>
```

Figure 3.11 Command Line in Redpitaya

By typing in the “cd /root/iic” the correct folder is entered. Now the command **make** can be executed like below.

One warning can appear, but no doubts about that. The RP has no chance to establish a connection to any timeserver (NTP-server), so the time starts at 1900 or something like that.



192.168.128.1 - PuTTY

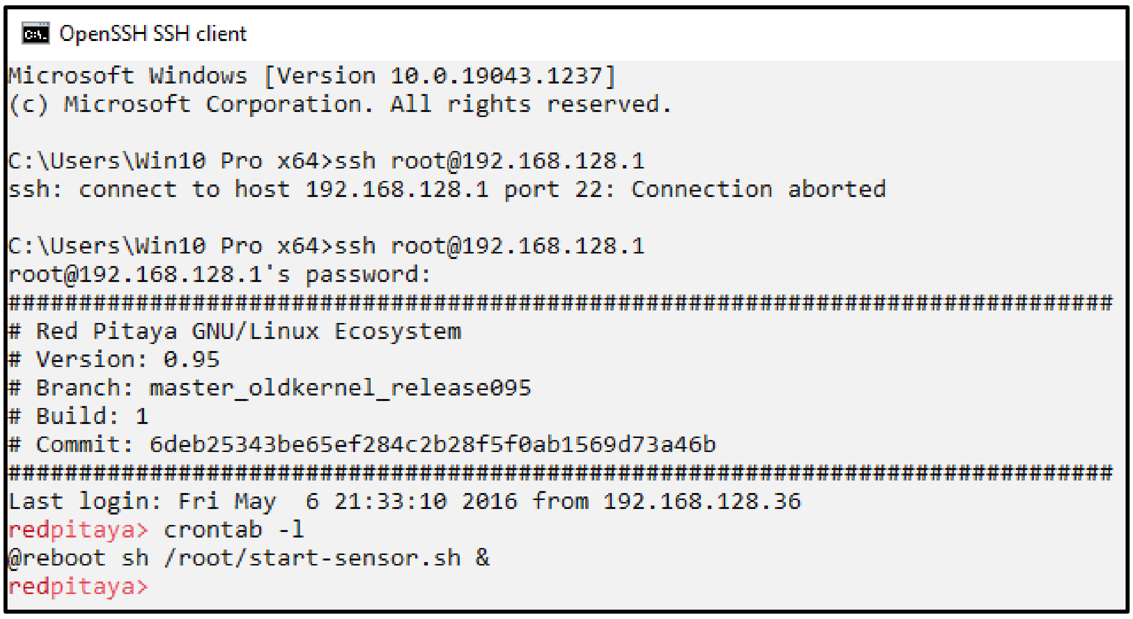
```
make: Warning: File 'Makefile' has modification time 150573499 s in the future
gcc -g -std=gnu99 -Wall -Werror -I/root/redpitaya/api/include/redpitaya -Ipthrea
d.h -L/root/redpitaya/api/lib -lm -lpthread -lrp kiss_fft.c -lm kiss_fftr.c -lm
-DFIXED_POINT=32 -DDEBUG    iic.c    -o iic
make: warning:  Clock skew detected.  Your build may be incomplete.
[1]+  Done                          clear
redpitaya>
```

Figure 3.12 Successful Compiling of SW

Other than this procedure we can use command prompt also. First, we have to connect the Red Pitaya to the WLAN through Wi-Fi device. Then start CMD as an administrator. After that write the following command.

ssh root@192.168.128.1

And use the password “redpitaya”. 192.168.128.1 is the IP of the instrument by means of which we can get access to it. The procedure of access Red Pitaya can



```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Win10 Pro x64>ssh root@192.168.128.1
ssh: connect to host 192.168.128.1 port 22: Connection aborted

C:\Users\Win10 Pro x64>ssh root@192.168.128.1
root@192.168.128.1's password:
##############################################################################
# Red Pitaya GNU/Linux Ecosystem
# Version: 0.95
# Branch: master_oldkernel_release095
# Build: 1
# Commit: 6deb25343be65ef284c2b28f5f0ab1569d73a46b
##############################################################################
Last login: Fri May  6 21:33:10 2016 from 192.168.128.36
redpitaya> crontab -l
@reboot sh /root/start-sensor.sh &
redpitaya>
```

Figure 3.13 Accessing Redpitaya through CMD

be seen in the Figure 3.13 below.

3.3.7 Tools and Environment

An ultrasonic sensor that is mounted on Red Pitaya was given. It was developed by Mr. Michalik prior to this work. To develop the main application, Visual Studio 2019 was used. The Main application is compatible with a higher version of Visual Studio. For the purpose of developing Classifier, C programming language, and additional libraries such as Linux i2c-dev, Kiss_fft, and etc. were employed. In addition to that, for the data preprocessing to deliver the data to the filters and different feature extractor functions, additional libraries were used, such as kiss_fftr, kiss_fft_guts etc. Especially, the process of classification will work in run time without using training and prediction.

The whole process works in parallel with the help of 4 threads. One is the main thread i.e., the Time Thread which will ultimately run the other 3 threads. One thread is for UDP server this thread is listening on port 61231. The message is parsed and gives a signal to the third thread to run. The third thread is waiting a signal from the server thread. This thread will start ADC and measurement by means of the ultrasonic sensor. And the measurement thread will trigger the Led Thread which will control the ON/OFF functionality of the classifier. The box classifier implemented will check on the feature F1 and F4 with a value of threshold.

Where F1 is the Mean difference between frequencies of neighbored measurements of center frequency (max. peak), i.e., moving average over 10 frequency differences of neighbored center frequency. And F4 is Variance of the center max peak over 10 measurements. Lastly, this work was conducted in the windows machine with visual studio community version 2019, and all the versions of these tools are for windows 10 64bit environment.

No	Component	Tool	Version
1	Ultrasonic Sensor	A sensor mounted on RP	Develop by Mr. Michalik
2	Environment	Operating System Processor Memory Storage	Debian GNU/Linux 8 (jessie) Dual core ARM Cortex A9+ and FPGA DDR3 RAM 512MB (4Gb) microSD up to 32Gb
3	Main application	C and C# in visual studio	2019
4	Libraries	Linux i2c-dev Kiss_fft Kiss_fftr	
5	LED light	LED PWR121	
6	Relay	HF3FA	Gravity version 4
7	Webcam	Camera for validation	Logitech

Table 3.1 Development Tools & Environment

3.4 LED PWR121

Two LED PWR121 lights has been used in this project. One is white light which will indicate the presence of Human or soft object. The second one is blue in color which will signify the presence of hard object or no object.

These are chargeable lights which starts charging when get connected to the power source through a USB cable with an input of 5 volts. A small red LED shows that currently it is charging, and a green LED signify that it has been charged 100%.

Normally this LED comes with and ON/OFF button to the side which when pressed for 2 seconds will turn ON PWR121. Just pressing the button will decreases the luminance and pressing it again for 2 seconds will turn OFF PWR121. But in our case will not manually press the button rather this functionality will be controlled by the RELAY module attached to the Red Pitaya [6].

So, in principle the RELAY will be controlled by the Red Pitaya controller and the controller will take action upon the received reflected ultrasonic signals from the object or person in front of it.

Figure 3.14 PWR121 Light [6]

3.5 RELAY module

We will describe here briefly the relay modules that we used in our project. We use 2 modules of HF3FA gravity version 4.0. It is compatible with Red Pitaya, 5V Arduino boards and also other boards like mbed, Intel, Raspberry Pi, BBB and other 3.3V devices. It is designed specifically for low-voltage devices.

Up to 10A current can be handled plus an LED on the board indicates the state of the relay. 1 NO and 1 NC terminal is included. A 3-pin interface is provided to make it plug and play. A wide range of power inputs are available. The Operating Voltage range is 2.8V ~ 5.5V [7].

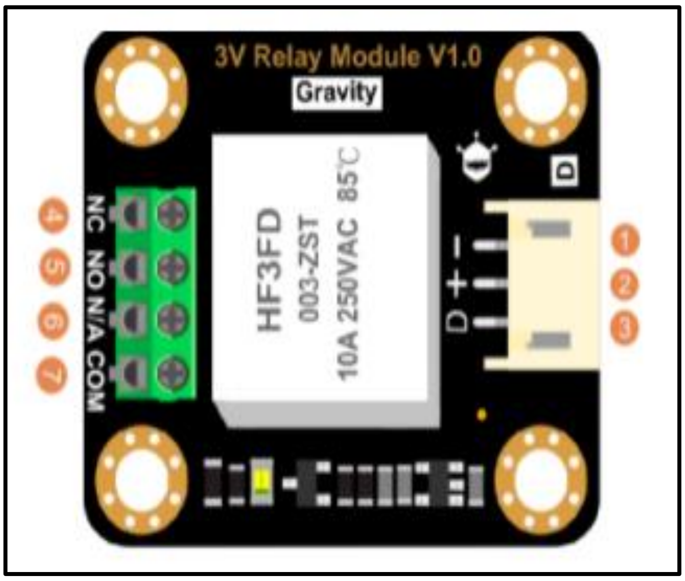


Figure 3.15 Relay Board Overview [7]

Table 3.2 shows the number of pins and its description.

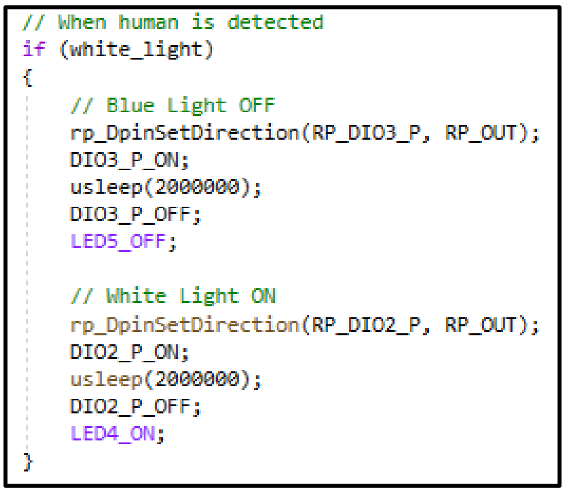
Num	Label	Description
1	-	GND
2	+	VCC
3	D	Control signal
4	NC	Normally Closed
5	NO	Normally Open
6	N/A	Empty Terminal
7	COM	Common port

Table 3.2 Terminals [7]

3.5.1 Working Principle of Relay

Basic working principle of relay is electromagnetic induction and electromagnetic attraction. When the circuit of the relay gets a high signal, it energizes the electromagnetic field which is known as electromagnetic induction which in turn produces the temporary magnetic field. By means of this the relay armature can open and close the connections. Those connections which are normally closed (NC) will open and those which are normally opened (NO) will get closed.

3.5.2 Sample Code



```
// When human is detected
if (white_light)
{
    // Blue Light OFF
    rp_DpinSetDirection(RP_DIO3_P, RP_OUT);
    DIO3_P_ON;
    usleep(2000000);
    DIO3_P_OFF;
    LED5_OFF;

    // White Light ON
    rp_DpinSetDirection(RP_DIO2_P, RP_OUT);
    DIO2_P_ON;
    usleep(2000000);
    DIO2_P_OFF;
    LED4_ON;
}
```

Figure 3.16 Activation of Relays

Above is the code snippet from the project which will activate the relay DIO2_P and DIO3_P which is taking signal from the DIO pin 2 and pin 3. These relays will get activated when the Red Pitaya sense that a human is in range. And the white light connected to the relay DIO2_P will illuminate. and the blue light will turn OFF.

3.5.3 Expected Results

The LED will change its state every time when the Ultrasonic Sensor encounter any change in object being observed, and you will hear a loud of pop at the same time. If the object in front remains the same the relay will not switch.

- When the module gets HIGH signal, NO terminal is connected with the COM port, and the LED is ON.

- When the module gets LOW signal, NO terminal is disconnected from the COM port, and the LED is OFF.

3.6 I2C Protocol

In this section we will be presenting the I2C “Inter-Integrated Circuit” protocol which basically combing some of the properties of SPI and UART communication protocol. SPI and UART are also quite often used but every protocol has its own advantages and disadvantages.

I2C protocol are widely in use for the projects such as barometric pressure sensors, OLED displays screens and accelerometer modules or gyroscope.

3.6.1 Introduction

It is a possibility using I2C to connect multiple slaves or multiple masters to a single slave or single master most like the SPI protocol. This is an important feature of I2C protocol if we are using a single monitor to show the results or if we want to access a single memory card by a number of microcontrollers. And other features of UART communication is used like only 2 wires are in used i.e., SDA and SCL to exchange data between devices. So, in short I2C is a best combination of both UART and SPI [8].

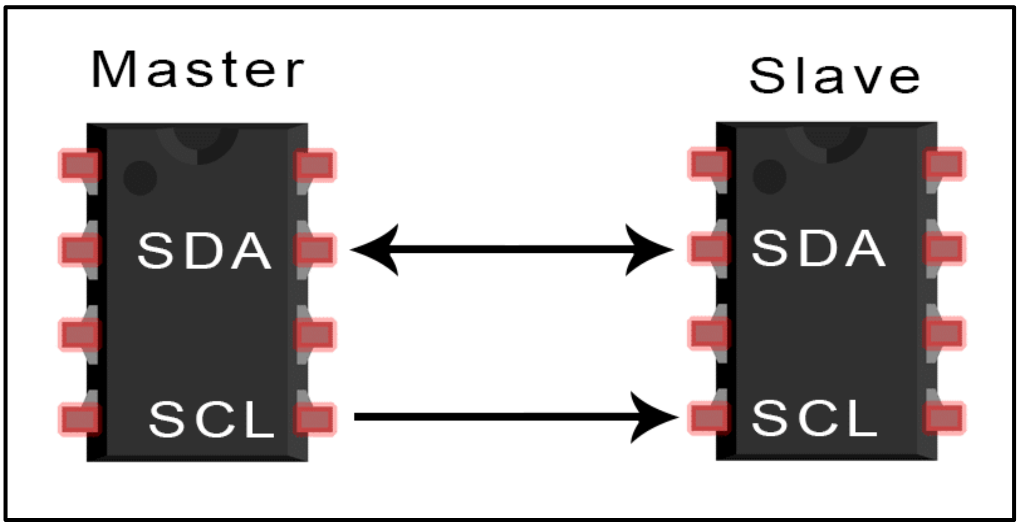


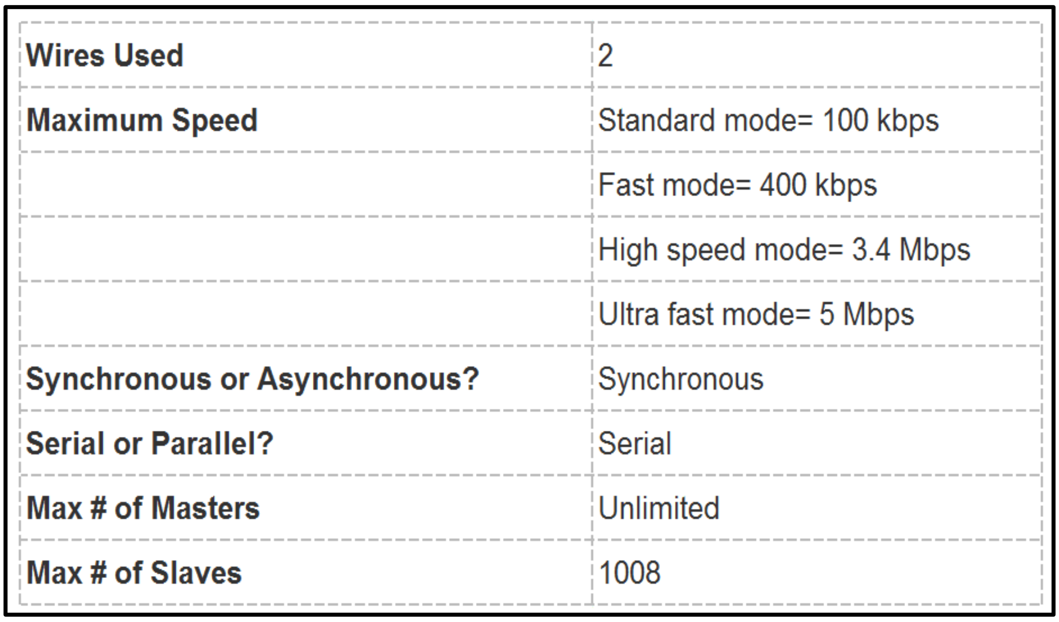
Figure 3.17 Single Master Single Slave [8]

SDA (Serial Data) – To send and receive data between slave and master this line is used.

SCL (Serial Clock) – This line is used for the clock signals.

Data has been transferred from one end to other end by a bit-by-bit fashion making use of a single wire (the SDA line). That is why I2C is termed as a serial communication protocol.

I2C is a synchronous protocol just like SPI, in synchronized protocols master is always in control of the clock signal. For both the master and slave same clock is used to synchronize the output bits to the sampling bits of the clock.



Wires Used	2
Maximum Speed	Standard mode= 100 kbps
	Fast mode= 400 kbps
	High speed mode= 3.4 Mbps
	Ultra fast mode= 5 Mbps
Synchronous or Asynchronous?	Synchronous
Serial or Parallel?	Serial
Max # of Masters	Unlimited
Max # of Slaves	1008

Table 3.3 Characteristics of I2C Protocol [8]

3.6.2 Working of I2C

Here we will discuss the message format of I2C or synonymously the header of I2C. So, each packet which is being transferred is termed as a Message. And a message is divided into frames of data. Each message contains the address of the slave as a binary address. This frame is known as Address Frame, and one or more data frames that include the information in form of bits that we want to transmit. The full message is enclosed into a start and stop conditions, 1 bit is reserved for read/write. To acknowledge if a data frame is received ACK/NACK bits are incorporated [8].

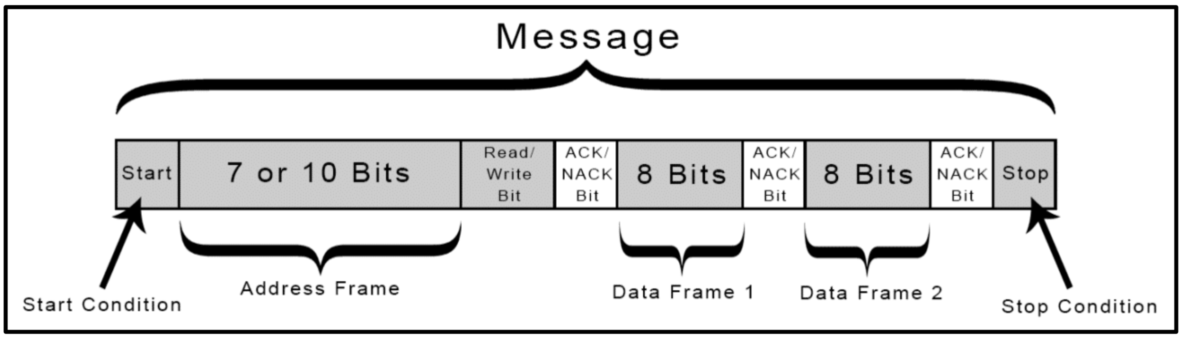


Figure 3.18 I2C Message, Frame and Bit [8]

Start Condition: SCL line changes from high to low after the SDA line changes from a high voltage level to a low voltage level.

Stop Condition: The SCL line changes it state from a low voltage level to a high voltage level before the SDA line changes from low voltage level to high voltage level.

Address Frame: Each slave is identified by an Address Frame. This Address frame is normally 7-10 bit long and is unique for each slave. Whenever master wants to communicate to a particular slave its address is used in the Address Frame.

Read/Write Bit: This is a single bit which is indicating if the master is waiting for the data to be received from the slave side or wants to send the data to the slave. If this bit is low voltage that is 0 the master is sending if it is high voltage that is 1 master is requesting data from the slave.

ACK/NACK Bit: This bit is notifying about the status of the last data frame. Each data frame is followed by ACK/NACK bit. The receiving end will send an ACK bit to the sender if the data frame is received successfully if it did not deliver properly a NACK bit is sent [8].

3.6.3 Addressing in I2C

SPI is using slave select lines to address a slave. I2C doesn't have lines like SPI. So how in I2C is it possible to point out the correct slave to which we want to address. For this very reason Address Frame in incorporated in each message of I2C protocol. This process is termed as addressing in many protocols. After the start condition Address frame is placed so every slave is checking the Address Frame and taking an action depending on the address if it belongs to them.

The address or the message will be sent to each slave connected to it when the master wants to communicate with a particular slave. The address has been checked by each slave to its own address. If the address doesn't match, the slave does nothing, and the SDA line will be high voltage. And if the address matches slave sends a low voltage ACK to the master.

3.6.4 Read/write bit

This is a single bit which is indicating if the master is waiting for the data to be received from the slave side or wants to send the data to the slave. If this bit is low voltage that is 0 the master is sending if it is high voltage that is 1 master is requesting data from the slave [8].

3.6.5 The data frame

The first data frame will be sent out by the master if it receives an ACK bit from one of the slaves.

The data frame is 8 bits long and its size is fixed, the most significant bit MSB will be the first one followed by the others. An ACK/NACK bit is attached to each data frame to notify about the status of the data frame. The next data frame will be sent only if the last data frame is Acknowledged. Either master or slave can receive the ACK bit depending on who is sending or receiving the packet.

The transmission of data frames will be discontinued be the master after the all the data has been exchanged successfully. This is done by a stop condition. This stops condition at the end of message will halt the transmission. Stop condition happens when the SCL line changes it states from a low voltage level to a high voltage level before the SDA line changes from low voltage level to high voltage level [8].

3.6.6 I2C data transmission steps

1. The master transmits the start condition to each, and every slave connected by changing the SCL line from a high voltage level to a low voltage level after switching the SDA line from high to low.

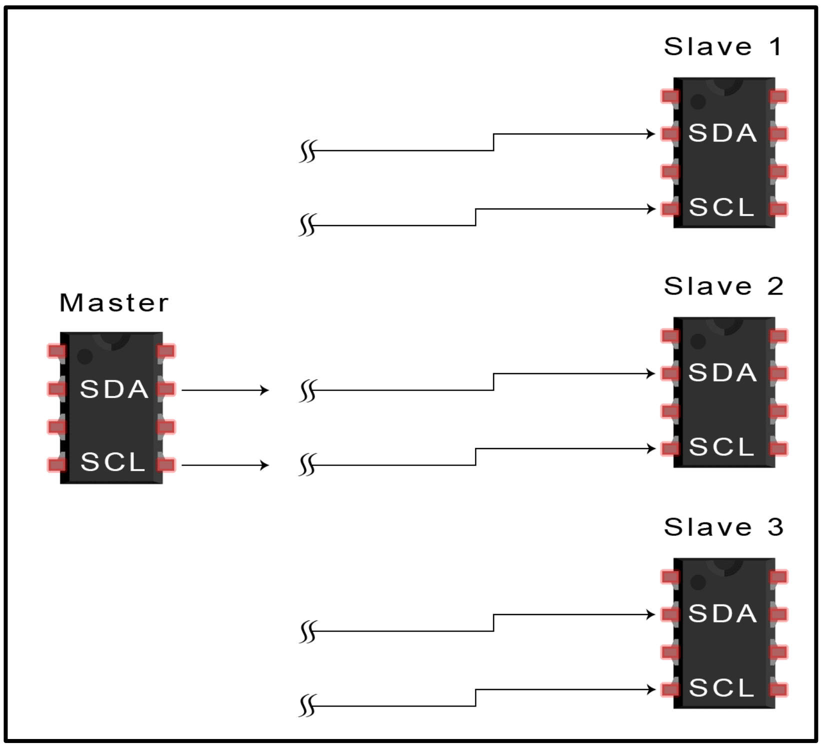


Figure 3.19 Data Transmission Diagram START CONDITION [8]

2. Along with the read/write bit the master broadcast a 7- or 10-bit address frame of the slave it chooses to connect with.

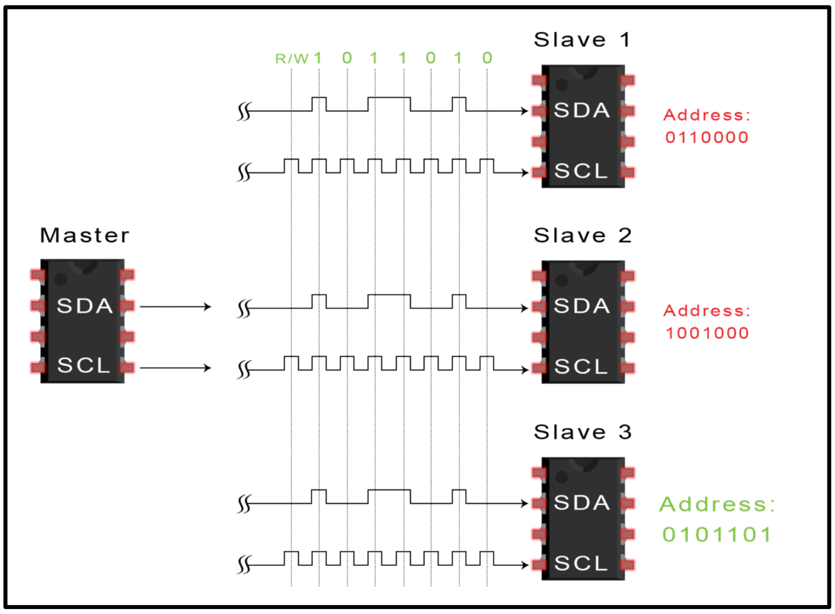


Figure 3.20 Data Transmission Diagram ADDRESS FRAME [8]

3. After broadcasting the address of the intended slave to which master wants to communicate each slave compares the address with its own address. when the address matches, the slave sends back an ACK bit by making the SDA line low for one bit. If the address from the master does not match the slave will do nothing and keep the SDA line high may be for other slave to make it low to notify the master [8].

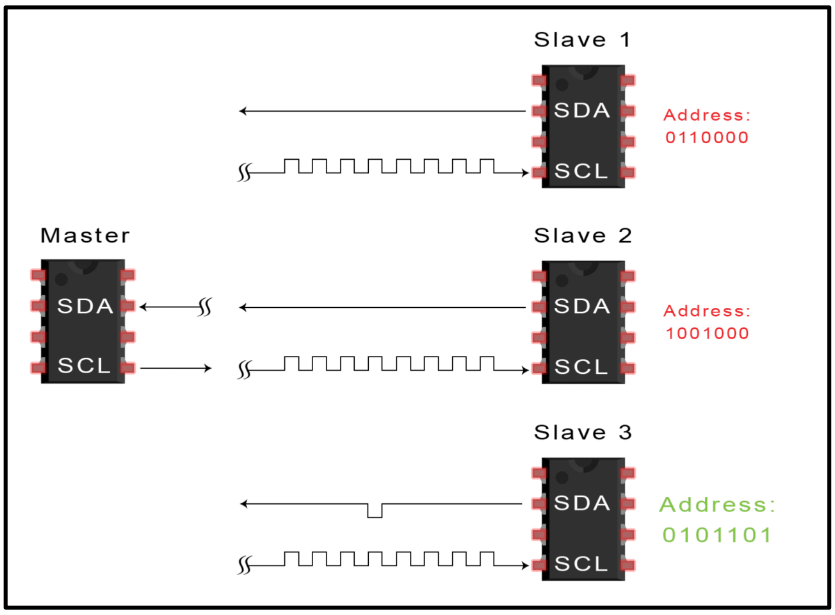


Figure 3.21 Data Transmission Diagram ACK Bit Slave to Master [8]

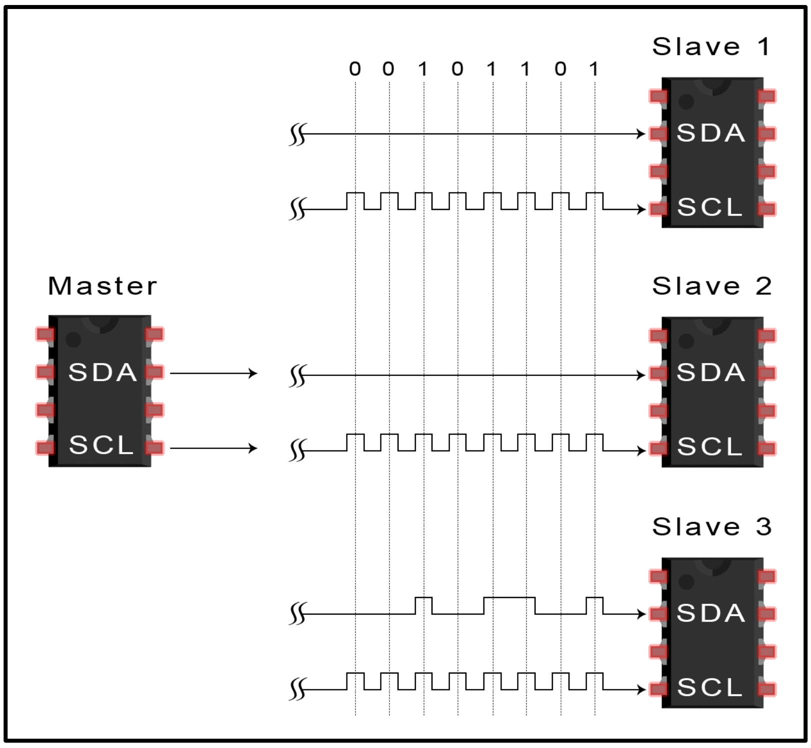


Figure 3.22 Data Transmission Diagram Data Frame [8]

4. The master receives of sends the data frame:

5. After the frame is received successfully an ACK bit is sent back to the sender to notify about the success of transmission for that particular data frame. This process will be done for every data frame [8].

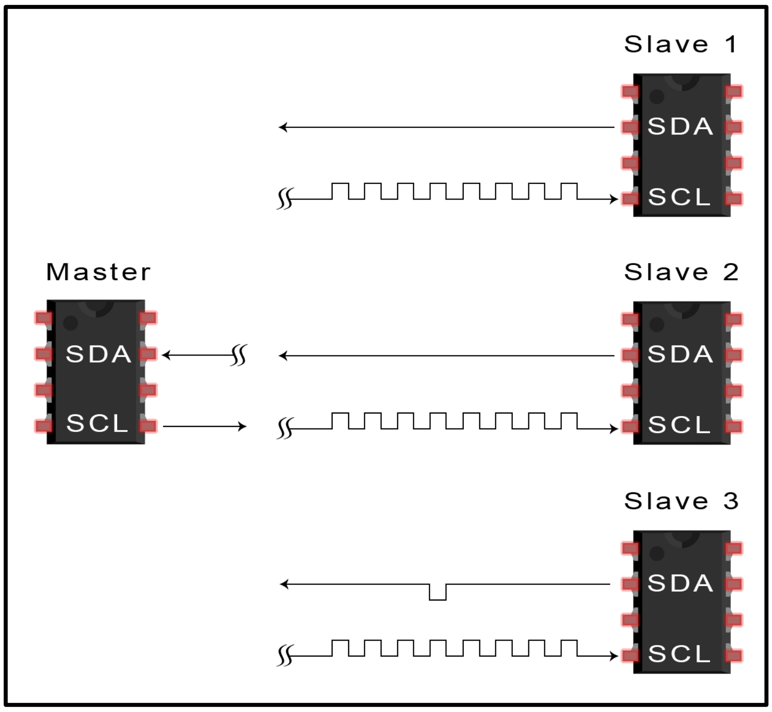


Figure 3.23 Data Transmission Diagram ACK Bit Slave to Master [8]

6. By changing SCL high before changing SDA a stop condition is send to the slave which will ultimately stops the transmission.

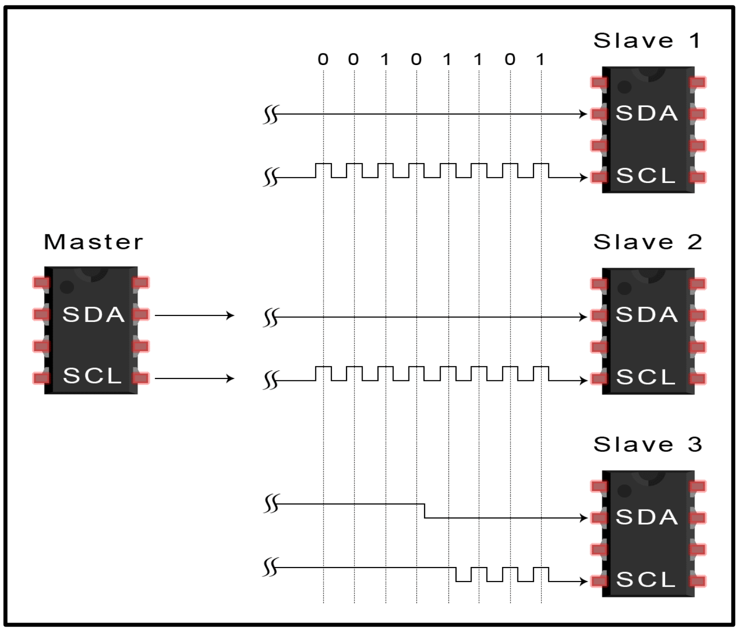


Figure 3.24 Data Transmission Diagram Stop Condition [8]

3.6.7 Multiples slaves single master

A single master can control multiple slaves by making use of addressing. 127 different addresses can be accessed by making use of a 7-bit address space. 10-bit addresses are uncommon, as it will be redundancy because 10 bit means a totally of 1024 different slaves can be addressed. Make the connection as below if one wants to connect a master to multiple slaves. Use 4.7K Ohm pull-up resistors which will connect the Vcc to SDA and SCL lines.

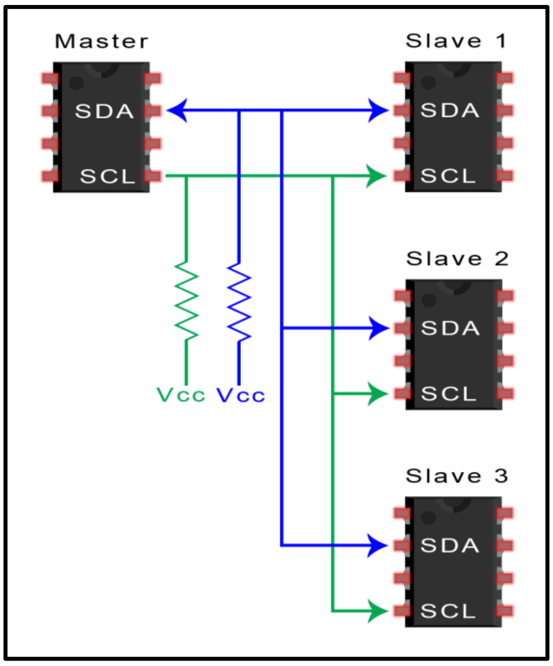


Figure 3.25 Single Master Multiple Slaves [8]

3.6.8 Multiple slaves with multiple masters

Multiple masters can be linked to multiple slaves or single slaves. The difficulty with multiple masters in one system arises when many masters try to send or receive data at the same time over the same data line. To solve this issue, each master needs to find out if the SDA line is high or low before sending a message. If the SDA line is low, this indicates that any other master is using the line, and all the remaining masters must wait to transmit their messages. If the SDA line is high, then it's allowed to send the message. To connect multiple slaves to multiple masters, make use of the following diagram, with 4.7K Ohm resistors which will connect the Vcc to SDA and SCL lines [8].

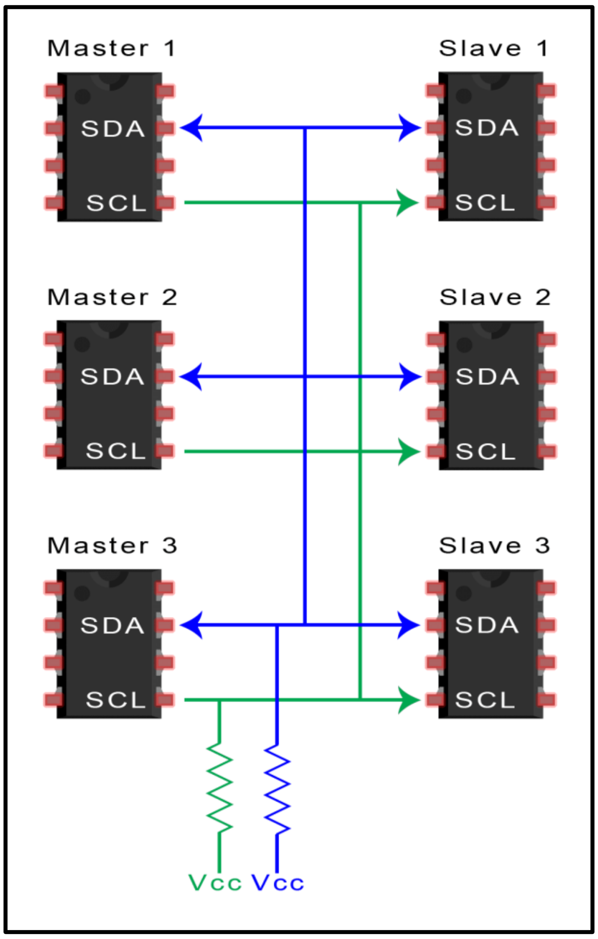


Figure 3.26 Multiple Masters Multiple Slaves [8]

3.6.9 Advantages and disadvantages of I2C

I2C protocol seems to be a bit complicated but complication brings a lot more advantages, also depends on the task in hand one must observe the advantages and disadvantages of using I2C. following are the advantages and disadvantages of using I2C [8].

3.6.10 Advantages

- Only two wires used for the whole process.
- Can support multiple slaves and multiple masters
- For confirmation of each frame delivery ACK/NACK bit is used.
- Less complicated hardware than UARTs
- Very famous and widely used protocol

3.6.10.1 Disadvantages

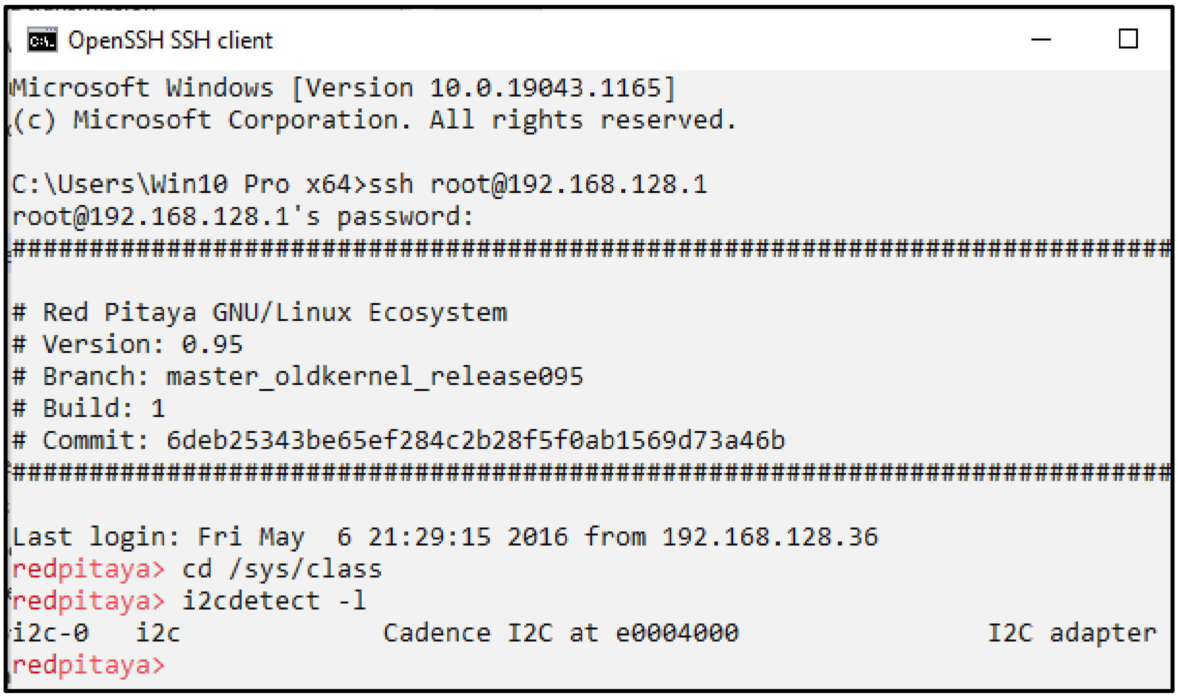
- Data is transferred with a less speed than SPI
- Data frame is very small and limited.
- To implement it much complicated hardware is needed than SPI.

3.7 I2C Linux Kernel

Linux/i2c-dev.h is a header file that is used in this project. And this header file shows and provide information on make use of /dev/i2c and interface with I2C devices. The I²C bus is quiet often employed to attach somewhat low-speed sensors and other computer peripheral to instrument differing in complexity from that of a motherboard or just a simple microcontroller. By making use of only 2 pins on the host controller we can easily and accurately take control of the attached multiple devices. That is why I²C is particularly prevalent. As discuss in the previous section we can use different modes of I²C, but as per the project demand we have 1 master and 1 slave so the communication will be purely between them [9].

Kernel driver is operating in the i2c devices. But also, all the peripherals can be access from a user space through an adapter with the dev interface. For that reason, i2c module will be loaded.

Starting from 0 each and every sensor or any other peripheral device is assigned with a number. This can be checked in the path /sys/class/i2c-dev/ to see which registered sensor has what number. Also, it is possible to check it with a command that we can run in command line window as shown in Figure 3.27.



```
Microsoft Windows [Version 10.0.19043.1165]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Win10 Pro x64>ssh root@192.168.128.1
root@192.168.128.1's password:
###########################################################################

# Red Pitaya GNU/Linux Ecosystem
# Version: 0.95
# Branch: master_oldkernel_release095
# Build: 1
# Commit: 6deb25343be65ef284c2b28f5f0ab1569d73a46b
###########################################################################

Last login: Fri May  6 21:29:15 2016 from 192.168.128.36
redpitaya> cd /sys/class
redpitaya> i2cdetect -l
i2c-0   i2c             Cadence I2C at e0004000                 I2C adapter
redpitaya>
```

Figure 3.27 I2C Adapter

After connecting to the Red Pitaya through ssh and using the "i2cdetect -l" to obtain a list of all i2c devices registered at the moment on the Red Pitaya. i2cdetect is component of the i2c-tools package.

I2C devices has character device files which shows the attached minor devices and the major devices. Each device has a number. As a convention the major device takes the device number 89. The remaining minor devices take the number in the format like "i2c-%d" (i2c-0, i2c-1, ..., i2c-10, ...). i2c minor devices can take a reserved space of 256 devices [9].

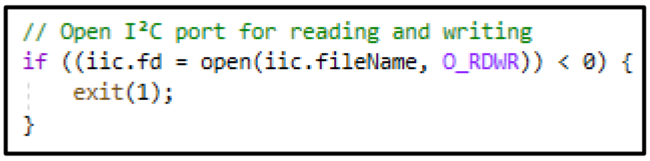
If we want to access ultrasonic sensor through an i2c interface of Red Pitaya in a C program first, we need to include the below headers in our program.

#include <Linux/i2c-dev.h>

After that we will check for the registered i2c devices in the location /sys/class/i2c-dev/ or alternatively we can run "i2cdetect -l". Sensors can get numbers dynamically it is not fixed. So, it is possible to presume about it beforehand. It is a best practice to check it [9].

3.7.1 Opening of the Bus

To open the character device files we will use the following lines of code as shown in Figure 3.28.



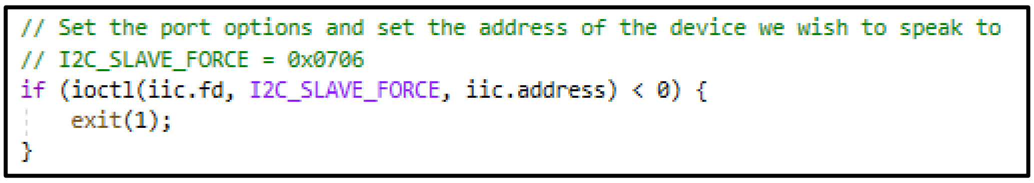
```
// Open I²C port for reading and writing
if ((iic.fd = open(iic.fileName, O_RDWR)) < 0) {
    exit(1);
}
```

Figure 3.28 Opening Bus

To interact with the I2C peripheral devices in simple way the first and important thing to start this communication is to open the bus for writing and reading. Same as we are doing for any other files. If we use fopen the call may buffer and stay on bus rather we use a simple open command. A non-negative integer known as file descriptor is returned when open command is executed. If the user doesn’t have rights or access to /dev/i2c-2 may get an error and the process may fail this is a typical failure. By adding the user to the list of permitted users this problem can be removed and will give the authority to adjust the file. A permanent solution to this issue is to add a udev rule which will combine the group device of I2C [9].

3.7.2 Starting communication

When the device has been opened, its specification is necessary for further operations. We can set the address of the slave to which we would like to communicate here in this case as shown in Figure 3.29 master wants to communicate to I2C slave of the address 0x0706. Now that the address is setup, we can use plain I2C to communicate.



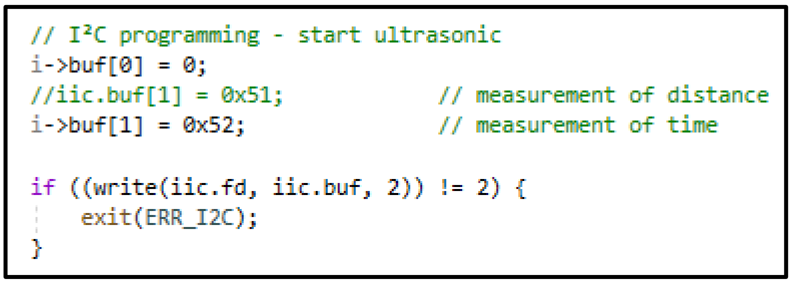
```
// Set the port options and set the address of the device we wish to speak to
// I2C_SLAVE_FORCE = 0x0706
if (ioctl(iic.fd, I2C_SLAVE_FORCE, iic.address) < 0) {
    exit(1);
}
```

Figure 3.29 Initiating Communication

We must start the communication with ultrasonic sensor which is a slave here in this case after the bus is successfully acquired. As from the previous discussion we know that I2C will do this by sending a message which contain 7-bit address of the slave followed by a read/write bit. The bit is set to 1 for reads and to 0 for write. 0 means low and 1 means high. Addressing the slaves is a common manufacturing error, as the manufacturers have no set standards. Some manufacturers are using a seven-bit number, which means that first the address is required to shift 1 bit to the left and then append a read/write bit to it at the end. Other manufacturer will support an eight-bit number and assume the user will take care of the last bit and sit it accordingly for the read/write operation. Majority of the manufacturers will not mention this, and the user have to find it out by trial-and-error method.

3.7.3 Writing to ADC

To obtain data from Ultrasonic Sensor the write system call is initiated. Write demands a buffer in which the data will be stored, a file handle, and the number of bytes to write must be mentioned. **Write** will try to write the number of bytes given and will return the total number of bytes written, making use of that we can check for errors. In Figure 3.30 we can see the buffer, **write** statement and error handling.

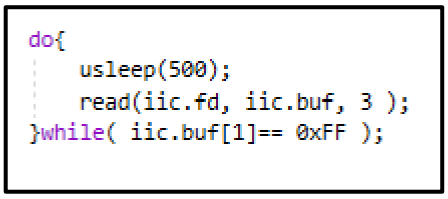


```
// I²C programming - start ultrasonic
i->buf[0] = 0;
//iic.buf[1] = 0x51;            // measurement of distance
i->buf[1] = 0x52;               // measurement of time

if ((write(iic.fd, iic.buf, 2)) != 2) {
    exit(ERR_I2C);
}
```

Figure 3.30 Writing to the ADC

3.7.4 Reading from ADC

The read segment of code is used to obtain data from the I^{2}C peripheral. Read demands a buffer to store the data, a file handle, and a number of bytes to read. Read will try to read the number of bytes stated and will return the total number of bytes for reading. The reading process is mentioned in Figure 3.31 below.

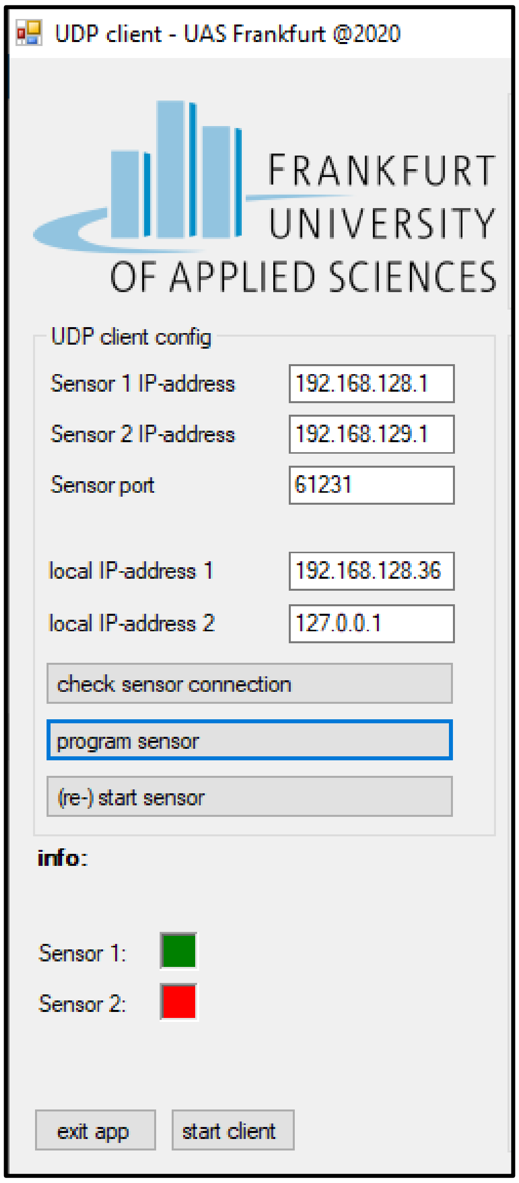


```
do{
    usleep(500);
    read(iic.fd, iic.buf, 3 );
}while( iic.buf[1]== 0xFF );
```

Figure 3.31 Reading from ADC

3.8 Automate Program and Starting Sensor

Previously if we used to make changes in our code according to our needs to account for different aspects of the scenario. The desired changes will be implemented and saved. After that with the help of the UDP client the new source code will be **programmed** in the Red Pitaya. When the sensor program successfully without any error we can **start or restart** the SRF02 sensor, which will work according to our classification method already implemented. As shown in Figure 3.32


UDP client - UAS Frankfurt @2020

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

UDP client config

Sensor 1 IP-address	192.168.128.1
Sensor 2 IP-address	192.168.129.1
Sensor port	61231
local IP-address 1	192.168.128.36
local IP-address 2	127.0.0.1

info:

Sensor 1:	🟩
Sensor 2:	🟥

Figure 3.32 Programming and Starting Sensor

After all the required changes been completed and our module is working perfectly, we want that the for the future use of the module it should start the sensor automatically after power is supplied to it. So that we don't have to start it manually from UDP Client. For that reason, we want to automate our process and command our hardware to do certain step of process after each restart. To achieve this behavior a job scheduler, known as **cron** is used.

3.8.1 Cron

It is a utility on the server which is mainly doing repetitive or scheduler task on Linux or Unix like operating systems. where we use a file known as **crontab** which contains commands and intervals. And the job which we will be doing by making use of command is known as **cronjob** [10]**.**

On Unix-like operating systems, the cron sometimes recognized as cron job, is a task scheduler. Cron is a tool used by users who set up and maintain software systems to plan shell scripts or commands to run at specific intervals or times. Although its general nature makes it handy for tasks like files downloading from the Servers and providing email at regular intervals when received, it is often used to automate system care or administration.

The repetitive tasks have been assigned to the Cron to make it easy for humans. It works as a daemon from the background to run scheduled process at its declared time. This daemon is built in Linux. Tables like format file is used know as cron tables or more simply **crontab** for already described scripts and commands [10].

In our case what we need is to command the Red Pitaya to start the sensor when it is power ON. This can be done by using a special syntax, so that we can configure to execute the commands to run automatically. This configuration by using commands and syntax is known as **cronjob**.

3.8.2 Requirements

- Linux or Unix Operating System
- Terminal window/ Command line window
- Administrator rights as a user or sudo privileges.

3.8.3 Existing Cron Jobs

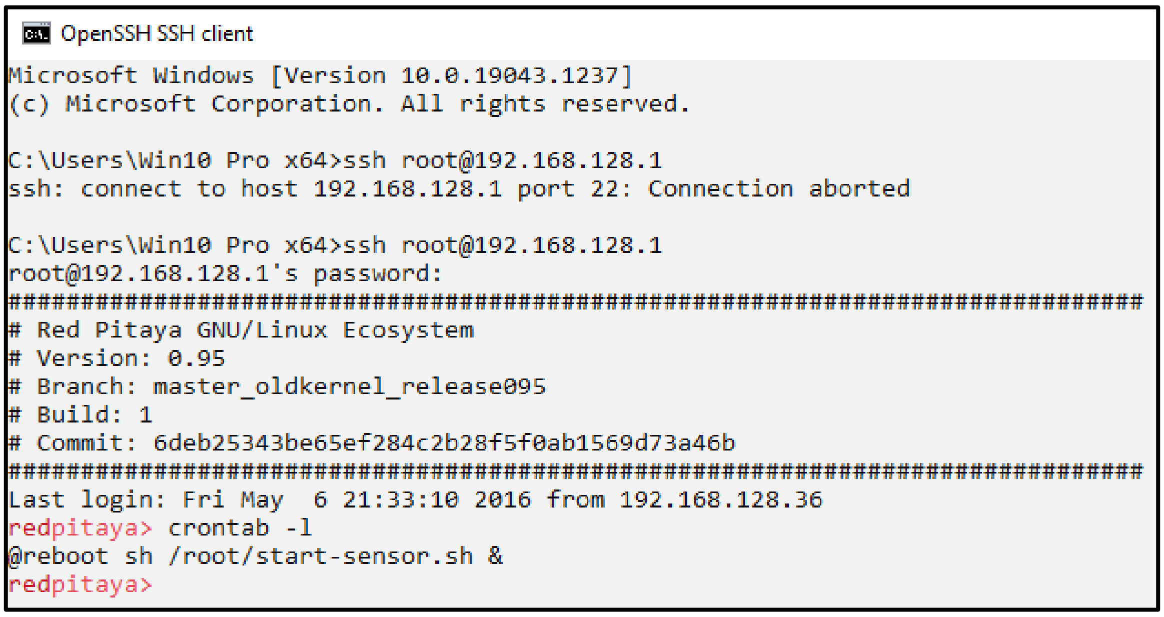
To check all the cron jobs running on a system we can simply use the command as given below.

crontab –l

by using this command, we can see all the cron jobs. There is no need to go to the cron tables configuration file and check it. This command can be used in a terminal window but first we have to establish a ssh connection for the Red Pitaya. After we have access to it, we can check it. The following command is used to establish a ssh connection. The password is root.

ssh root@192.168.128.1

in **Error! Reference source not found.** we can check that we have currently 1 cron job running and what it is basically doing is that it is starting a shell scripting file



```
OpenSSH SSH client
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Win10 Pro x64>ssh root@192.168.128.1
ssh: connect to host 192.168.128.1 port 22: Connection aborted

C:\Users\Win10 Pro x64>ssh root@192.168.128.1
root@192.168.128.1's password:
##############################################################################
# Red Pitaya GNU/Linux Ecosystem
# Version: 0.95
# Branch: master_oldkernel_release095
# Build: 1
# Commit: 6deb25343be65ef284c2b28f5f0ab1569d73a46b
##############################################################################
Last login: Fri May  6 21:33:10 2016 from 192.168.128.36
redpitaya> crontab -l
@reboot sh /root/start-sensor.sh &
redpitaya>
```

Figure 3.33 SSH Connection and List of Cron Jobs

name **start-sensor.sh** at each reboot. What we have in this file will be discuss in the next sections.

The owner of the crontab file receive an email when it runs. If the owner wants to keep track of the changes happening because of it but it will create an overhead of emails so if will be better to avoid this feature. As this is an optional feature, we can restrain from this by using the command string given below with the & operator [10].

v w x y z sh /root/sensor-start.sh &

3.8.4 Basic Crontab Syntax

Cron checks the cron tables which are the configuration files for a list of orders to perform. To understand the configuration files daemon is using a specific style of syntax.

To configure a cron job first and the most important thing is to understand each element of the syntax. In this way we will be able to set up a cron job. Following is the standard format of crontab line.

v w x y z /root/sensor-start.sh output

- **v w x y z** the first 5 fields is to state the date/time and repetition of the task.
- In the next field, the **/root/sensor-start** specifies the location and file you want to run.
- The last field mentioned as **output** is non-compulsory. It describes how the system informs the handler after the task is completed.

The first five fields are numbers that shows how frequently the syntax will execute. Every position is separated by a blank space. That is in essence a particular value [10].

3.8.5 Execution of Command

This segment states the command to perform. It represents the location in the directory of files and also the filename which is normally a shell scripting file. For example:

/root/sensor-start.sh

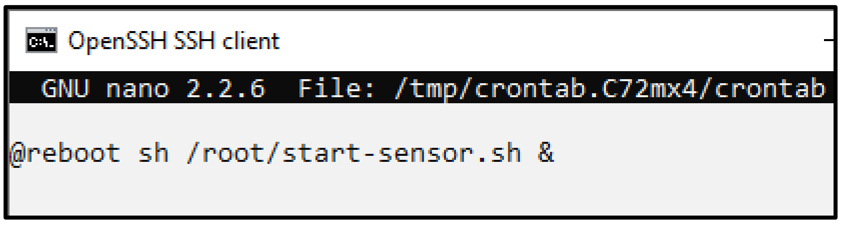
In our example, the command search in the root directory of the Red Pitaya and runs the sensor-start.sh script. And in this file, we can specify any commands we would like to execute [10].

3.8.6 Crontab File Editing

To check the configuration files of cron tables in Red Pitaya we can enter the following segment of command in CMD terminal window. In this way we can open the cron tables.

crontab –e

We can put a number of repetitive tasks each task must be in a separate line. In our case we have simply one task which is starting the SRF02 sensor at each reboot. So, no need to start it from UDP client. @reboot is the command for demanding the module to do which follows after it as we can see in **Error! Reference source not found.**. Sh s hows that the file which will start at reboot is a shell scripting file.



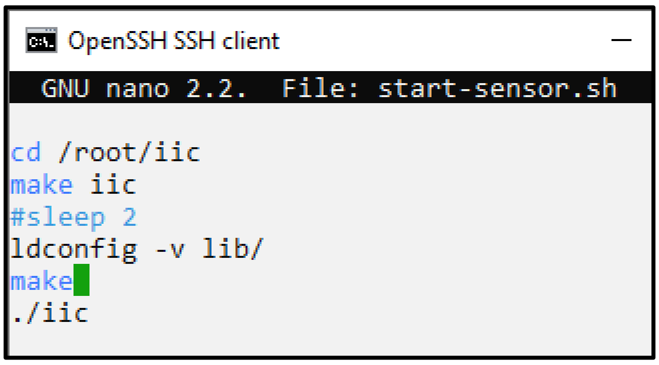
```
OpenSSH SSH client
 GNU nano 2.2.6  File: /tmp/crontab.C72mx4/crontab

@reboot sh /root/start-sensor.sh &
```

Figure 3.34 Configuration Tables

We can make changes as per our needs in start-sensor.sh file. After that we need to save the file and exit. Cron daemon does not need to restart to apply the changes being made. It will go through the instructions and perform which is provided [10].

3.8.7 Start-sensor.sh

In this file all the steps have been mentioned each in separate line which is in accordance with what we are doing in Form1 of the UDP client. Each line of code is described below.



```
OpenSSH SSH client
 GNU nano 2.2.  File: start-sensor.sh

cd /root/iic
make iic
#sleep 2
ldconfig -v lib/
make
./iic
```

Figure 3.35 Start Sensor Shell Scripting file

- First line of command is to instruct the daemon to move to root/iic directory.
- Second command make the iic file which means to build and compile the source code in iic file.
- Third line of command is just for testing purpose to give a delay of 2 sec which is already commented.
- Fourth line is to checks the header and file names of libraries it meets when establishing which versions should have their link renewed.
- Fifth line of command is to build, compile and run the whole program.
- Sixth line of command is to start the sensor with ./iic command

3.8.8 Using Operators

For efficiency, cron syntax also uses operators. Operators are special characters that perform operations on the provided values in the cron field.

(*) : An asterisk is used for all values. This operator can be incorporated in cases where if we want to run a task for all days, all weeks or all months.

(,) : A comma denotes distinct individual values.

(-) : A dash shows a range of certain values.

(/) : A forward-slash is used to divide a value into steps.

By the use of special characters, we can automate the most difficult tasks also by adding the advantages of cron tables we can achieve tremendous automation results [10].

Chapter 4 Areas of Applications

4.1 Introduction

Development of electronics and technology in the last two decades has assisted improvement in innovation. Microcomputers in modern vehicles help the driver with security, assistance and control. Electronic systems control most functions in vehicles. There is a lot of research and development, to increase the information exchange in the vehicle-to-vehicle communication and vehicle to infrastructure. Advanced Driver Assistance Systems (ADAS) handle more complex and complicated tasks. ADAS include lane-keeping assistance (LKA) and Adaptive Cruise Control (ACC), both assist a smooth drive [11].

4.2 Reduction of Accidents

One of the main objectives of EU is to reduced number of accidents and severity of accidents. Global automobile industry aims to reduce the number of accidents. More than 1.2 million people have died in accidents, and more than 50 million people [11] injured globally. The age group 15 – 44, are more prone to accidents. Predictions claim that road accidents will be the second cause of death by the next decade. Therefore, autonomous vehicles have a vital role to play to improve safety on the roads.

4.3 Decision making of Autonomous Vehicles

In addition to innovation, autonomous vehicles have to overcome the risk that the human driver would face in the event of an accident. There is a drastic improvement in autonomous vehicles to reduce accidents. One of the primary goals of autonomous vehicles is to determine the environment and make decisions. The automobile industry has to continually improve decision making ability of automobile for safe and reliable drive [11].

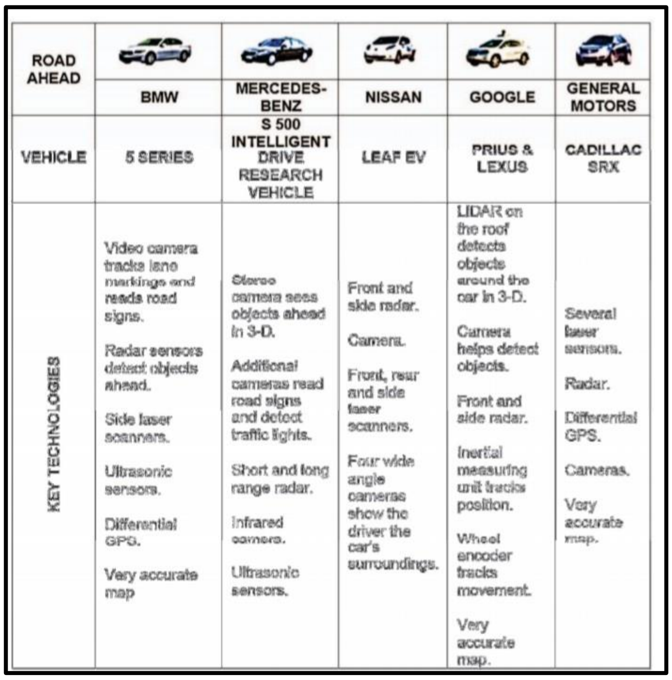
4.4 Saving Energy and Reduced Emissions

Global Challenge is to make environment-friendly vehicles. Society can progress better with low carbon emissions by achieving 80 reductions of emissions by 2050. To achieve environment-friendly cars, there should be reduced emission by 60 in 2040 and 40 by 2030. Companies functioning in automobile sector should take this into account and work towards the goal. Active green driving aims at fuel economy and reduced emissions. The vehicle calculates route profile based on

1. Topographical digital maps which are used for predictive vehicle management 2. Cloud, to receive information from sensors of autonomous cars and continuously provide real-time information 3. Up to date road database located in sensors [11].

4.5 Classification of Autonomous Cars

Many automobile companies are adding the autonomous feature to their existing design and creating a prototype for testing purpose. These include all the well-known OEM's. Based on the National Highway Traffic Safety Administration, autonomous technology is classified as shown in the Figure 4.1 [12].



ROAD AHEAD	BMW	MERCEDES-BENZ	NISSAN	GOOGLE	GENERAL MOTORS
VEHICLE	5 SERIES	S 500 INTELLIGENT DRIVE RESEARCH VEHICLE	LEAF EV	PRIUS & LEXUS	CADILLAC SRX
KEY TECHNOLOGIES	Video camera tracks lane markings and reads road signs. Radar sensors detect objects ahead. Side laser scanners. Ultrasonic sensors. Differential GPS. Very accurate map	Stereo camera sees objects ahead in 3-D. Additional cameras read road signs and detect traffic lights. Short and long range radar. Infrared camera. Ultrasonic sensors.	Front and side radar. Camera. Front, rear and side laser scanners. Four wide angle cameras show the driver the car's surroundings.	LIDAR on the roof detects objects around the car in 3-D. Camera helps detect objects. Front and side radar. Inertial measuring unit tracks position. Wheel encoder tracks movement. Very accurate map.	Several laser sensors. Radar. Differential GPS. Cameras. Very accurate map.

Figure 4.1 Existing Autonomous Classification in OEM Entities [12]

4.5.1 Level 1- Function specific Automation:

Human Driver is entirely responsible for the vehicle control, decision and drive, i.e., the driver has hands-on steering and foot on the pedal all the time. Specific functions are automated, which includes cruise control, guidance of lane and parallel parking.

4.5.2 Level 2- Combined Function Automation:

In this level, automation of various functions is integrated such as adaptive cruise control with lane centering. Drivers can be disengaged from vehicle operation under certain conditions and is responsible for monitoring, however expected to be available all the time.

4.5.3 Level 3 -Limited Self-Driving Automation:

Drivers do not monitor the system continuously. The driver can enable all safety-critical functions based on certain criteria. The drive can be dependent on the vehicle to control and monitor changes in various environmental conditions that require transiting the drive control back to the driver.

4.5.4 Level 4 - Full Self-Driving Automation:

In this level, vehicles can carry out all driving functions as well as monitoring environmental conditions for the entire journey, and it can operate with or without a driver. To achieve this, automobile companies have started testing their pioneer prototype and are trying to achieve accuracy and confidence to ensure blind faith in autonomous vehicles. The transition from conventional to autonomous vehicles should

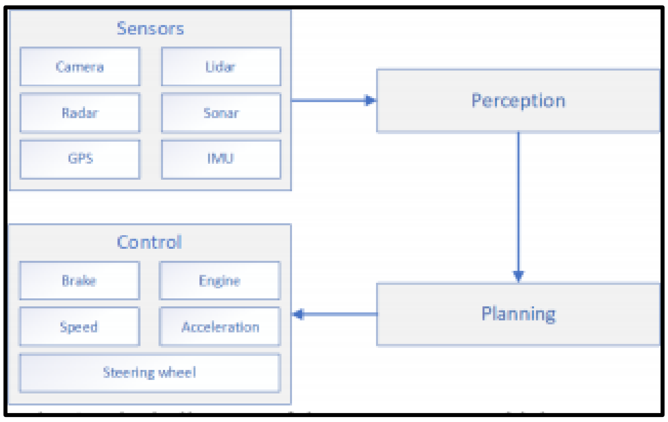


Figure 4.2 Block Diagram of Autonomous Vehicle [13]

be progressive to provide best user experience, enabling companies to scale up [12].

4.5.5 Objectives of Autonomous Vehicles:

The above Figure 4.2 explains the principle of an autonomous vehicle. It involves sensors to sense the surrounding, based on comprehensive data gathered. Accordingly, perception is made, and a concrete plan to control the vehicle is carried out, which results in different behavior of autonomous vehicles. In the below section, the essential objectives of the autonomous vehicles are explained [12].

4.5.6 Perception

The vehicle can understand and interpret its changing environment, enabling it to avoid accidents and monitor obstacles along the path of travel. Electronic devices used for this purpose are sensors such as LiDAR, RADAR, GPS and Mono- vision camera. LiDAR (Light Detection and Ranging systems) uses laser range finders which emit light beams to calculate the distance from obstacles and the time of flight until obstacles in the environment return a reflection. Similarly, Radio detection and ranging, known as RADAR, is another significant sensor for autonomous vehicles. RADAR systems utilize signal 's time of flight to calculate the range of objects in the environment. Monovision camera sensors have a single source of vision, making it simple and cheap where the

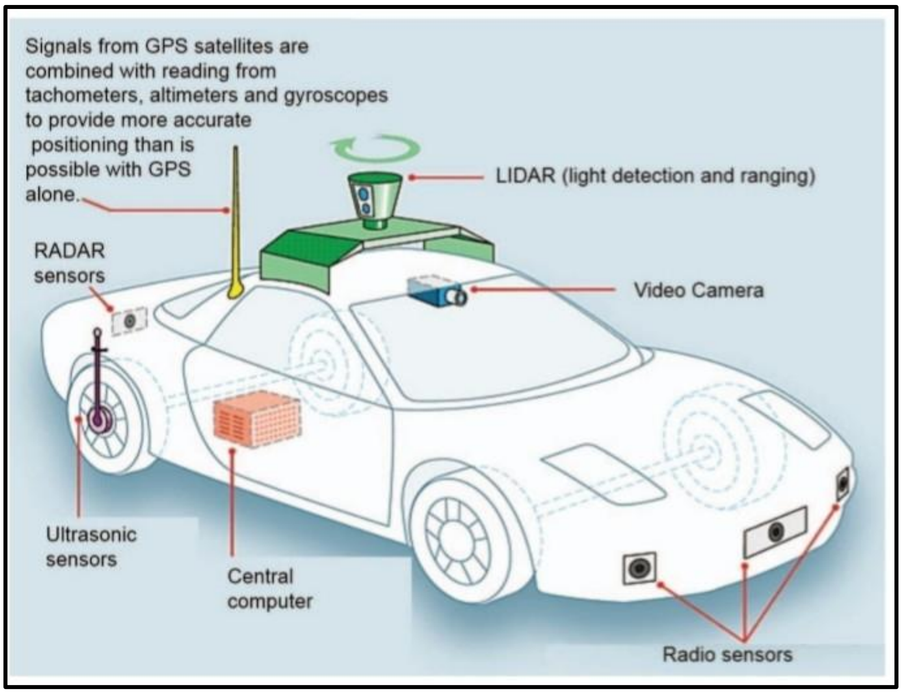

Signals from GPS satellites are combined with reading from tachometers, altimeters and gyroscopes to provide more accurate positioning than is possible with GPS alone.
LIDAR (light detection and ranging)
RADAR sensors
Video Camera
Ultrasonic sensors
Central computer
Radio sensors

Figure 4.3 Perception [12]

video feed is used for understanding the surrounding environment. The monovision camera takes into account, fixed infra- structured lane markings, parking symbols speed limit. The below figure Figure 4.3 contains the perception aspect of the autonomous vehicle [12].

4.5.7 Motion Planning

Motion planning consists of path variables, which needs to be controlled to avoid any mishap. Those path variables are:

4.5.7.1 Steering

Motion planning should be carried out efficiently in such a way that the vehicle is able to steer through all types of static and dynamic traffic.

4.5.7.2 Speed

Motion planning module should be performed efficiently to control the speed of the vehicle according to changing environment. Based on time taken, fuel consumption, distance from obstacles and other constraints, autonomous vehicles should be able to assign costs to each path. Autonomous systems should be able to manage and create local paths which can be followed. The best path based on time, cost, traffic and other constraints should be chosen among all available paths.

4.5.7.3 Navigation

Vehicles for localization use sensor suites, i.e., calculating their current geo-graphic position. For localization, global positioning systems (GPS) is used. Vehicle GPS systems continuously receive signals from orbiting satellites to calculate the current position on the global coordinates. To calculate the position of the vehi cle on the road, coordinates are cross-referenced with maps of the road network. GPS works with inertial navigation systems (INS) which consists of gyroscopes and accelerometers to continuously determine position, orientation and velocity of the vehicle without external references.

4.5.7.4 Behavior

Once the autonomous vehicle has understood its environment, completed the motion planning and navigated in the best possible route, it is time for the autonomous vehicles to act. Based on all specific functions and parameters, an autonomous vehicle takes decision [12].

4.6 Sensors in Autonomous Vehicles

The control system for the autonomous vehicle is based on environmental sensors which are providing information about the surrounding. The system contains a remote RADAR which is placed in the vehicle's front bumper. To detect the vehicle at high speed, long-range RADAR is used. The LiDAR Laser Scanner maps objects which are at a medium distance from the vehicle. There are also multilayer laser scanners. Optical cameras are used to identify traffic signals and lanes. Ultrasonic sensors and RADARS are used to map objects which are of front medium and short-range. The sensors in the autonomous vehicles are as shown below in the Figure 4.4 [11].

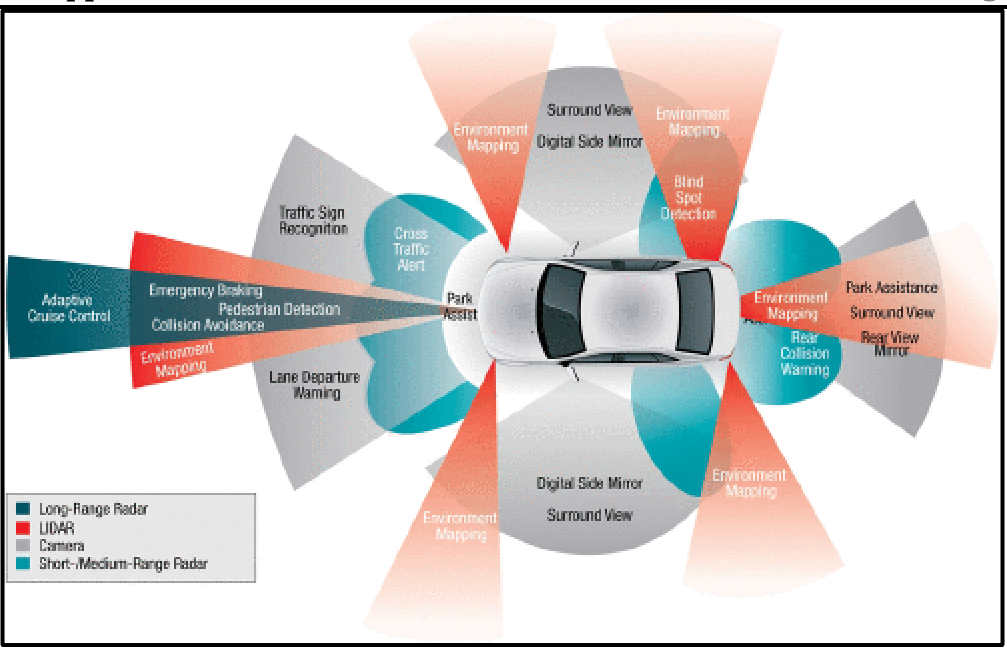
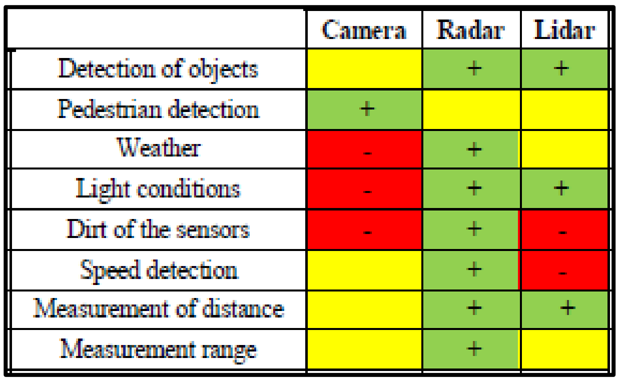


Figure 4.4 Sensor Layout of Autonomous Vehicles [11]

4.6.1 Sensor Properties

The main goal of environmental sensors is to interpret the surrounding environment and provide comprehensive information. Vehicles are mounted with a variety of sensors to provide near, medium and distant range information. Sensors are placed in different locations to understand the effects of variation in weather, to detect different angle and to calculate the maximum distance of the observed object. Besides, the sensor system must understand the varying reflection or intensity of the illumination from the surface of the detected systems. During the change in direction of the vehicle or the drive through the tunnel exit, there might be a sharp change of light intensity [11]. The below Figure 4.5 shows a comparison of properties between sensors, which are part of autonomous vehicles. From the table, we can interpret that there is no sensor which has an application for all scenarios. Therefore, the combination of sensors could be used for suitable applications. Depending on the car manufacturer, the configuration of the sensor is different for different cars. Some cars contain 6-8 spatial cameras that cover 360 degrees area having a front range camera of 250 meters, eight ultrasonic sensors to detect hard and soft objects , a RADAR with advanced data processing in the front end of the vehicle to provide additional space data and being able to measure the distance in



	Camera	Radar	Lidar
Detection of objects	🟨	🟩 +	🟩 +
Pedestrian detection	🟩 +	🟨	🟨
Weather	🟥 -	🟩 +	🟨
Light conditions	🟥 -	🟩 +	🟩 +
Dirt of the sensors	🟥 -	🟩 +	🟥 -
Speed detection	🟨	🟩 +	🟥 -
Measurement of distance	🟨	🟩 +	🟩 +
Measurement range	🟨	🟩 +	🟨

Figure 4.5 Comparison of Sensors [11]

varied weather conditions such as heavy rain, fog and dust.

The difference between RADAR and LiDAR is the fundamental nature on which they work. The LiDAR uses optical beam, and the RADAR works on the principal of sound wave. RADAR sensors can detect transparent objects such as windshields of cars, whereas LiDAR cannot detect transparent objects as the light passes through these materials. By linking the data of all the sensors, a comprehensive solution to weather problems can be derived. Data fusion from the sensors could assist in analyzing a complex environment for the control system of the autonomous vehicle [11].

4.6.2 Types of Sensors

4.6.2.1 Camera

Camera sensors are one of the first sensor types implemented in autonomous vehicles to achieve perception. For the automobile industry, it is one of the most crucial decision to choose right sensors and sensor systems. Variety of cameras in multiple numbers are mounted on driver-less vehicles. Camera assists in visualizing real-time environment in which the autonomous vehicle is travelling, also making them less expensive compared to RADAR or LiDAR. Camera sensors can be used in texture interpretation and classification, requiring high computational power to process the information - the most significant disadvantage of camera sensors. HD cameras produce millions of pixels per frame. Creating intricate imaging requires 30 to 60 frames per second, resulting in multi-megabyte data which is required to be processed in real-time. The many applications of using camera sensors in autonomous vehicles are perception, semantic segmentation, end to end autonomous driving and human-machine interaction [11].

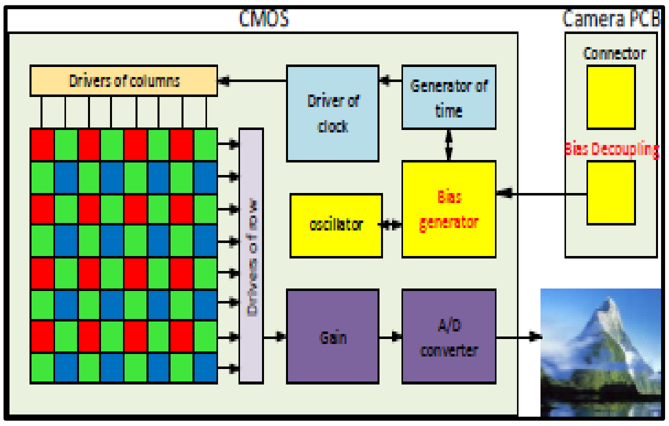


Figure 4.6 CMOS Camera [11]

Cameras in the autonomous vehicle are usually located behind the windshield and can also be located at different locations of the vehicle. There are two types of camera sensors used in the automotive vehicles, complementary metal-oxide-semiconductor (CMOS), and charge-coupled devices (CCD). Photons are converted to voltage for each pixel in CMOS sensors. Several transistors are located close to each pixel to measure and to amplify the signal from each pixel. The main advantage of this arrangement is that the entire collection of data takes place in the chip. This technique results in CMOS chip rate. The main difference in the chip reading technique is that the architecture of the sensor is significantly different. CMOS sensor as shown in the Figure 4.6 helps in converting the charge of each pixel into an electrical signal and functions are directly performed on the chip that results in less sensitivity, whereas in CCD, total photons per pixel is measured. To capture color, a color filter or a three-chip camera is required, making CCD chip more sensitive in comparison to CMOS.

Car cameras can be classified based on its position in the autonomous vehicle as well. There are front cameras, rear cameras, lateral cameras and ones manufactured by the automobile manufacturer. Camera classification can also be based on color. Color can be classified as black and white, monochrome or one color of RGB and another RGB color. These cameras can also be classified into mono-space perception or stereo-space perception. The color of the environment can affect the precision of some camera functions and reliability. Most essential features can be analyzed using black and white camera, which captures only varied brightness levels of each pixel. By including a color, there can be a scope for improvement in the performance. For instance, identification of traffic signs could be more reliable by using red pixel sensitive camera. There is significant impact on 3D vision by Stereo cameras, which helps in calculating the distance of the object [11].

4.6.2.2 Ultrasonic sensors

Ultrasonic sensors are applied in short distance parking assistance and are placed usually on the vehicle bumper. Ultrasonic sensor uses sound waves above 20kHz (threshold of human hearing). These sensors help to measure the distance between the object on which the sensor is oriented and the sensor. When the ultrasonic sensors perceive the object in its orientation, the sensor is activated, transmits an audible signal to the object and signal is reflected from the object back to the sensor. The sensor then calculates the time between transmission and reception of the audible waves, determining the distance of the object from the sensor. Ultrasonic sensors as shown in the Figure 4.7 mainly used as parking assistants in the automotive industry.

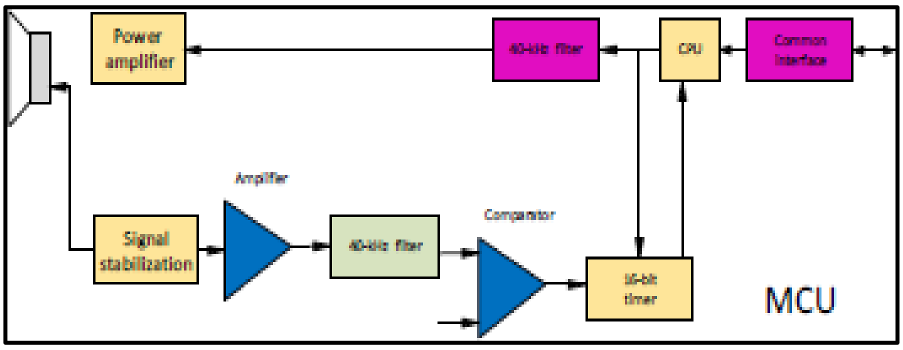


Figure 4.7 Ultrasonic Sensors [11]

Ultrasonic sensor works on the principle of generation of sound waves. During the transmission of sound waves, the threshold for the receiver is set, as interference between the receiver and the generated sound waves are eliminated. Farther the object, the threshold decreases as the elapsing time from the start of the transmission to the reflected wave diminishes the amplitude, depending on the reflection capability of the material [11].

4.6.2.3 LiDAR

LiDAR is Light Detection and Ranging. It makes use of infrared laser beam to calculate the distance covered between the near object and the sensor [13]. LiDAR which are commonly used needs light of 900 nm wavelength, and some LiDAR perform better in rain and fog as they use long wavelength.

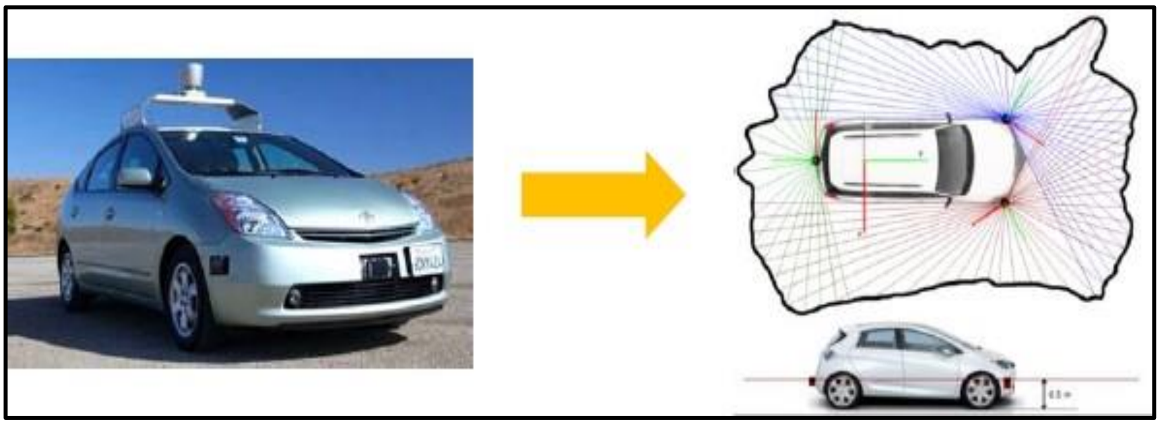


Figure 4.8 LiDAR Systems [14]

LiDAR functions similar to that of ultrasonic distance measurement, located on the front of the bumper of the autonomous vehicle. The main goal of sensors as shown in Figure 4.8 to detect other vehicles in the lane and static objects such as obstacles along the pavement and pedestrians. The LiDAR is designed either as 2D systems, which has a rotating mirror to map the distance from the vehicle or as 3D systems where the multidimensional mirror is providing 3D space information. This process can be repeated at the rate of 5-50 times per second, supporting the creation of real-time 3D view of the surroundings.

To measure the distance of the objects from the sensor there are two methods:

1. Measurement of time between the beam generated and received, after reflection.

2. Measurement of distance of the sensor by calculating the phase shift between the transmitted and the reflected beam. The primary advantage of using sensors is that they can measure range, accuracy and speed. The main disadvantage of LiDAR sensors is it cannot be used on transparent objects such as glass [11].

4.6.2.4 RADAR

RADARS are mounted on cars to detect objects surrounding the vehicle. These sensors are also used in detecting hazardous situation, in the event of which the drive can be controlled by interfering with the braking system and vehicle control. The RADAR systems are not only used to detect objects, but also to determine the vehicle's relative speed.

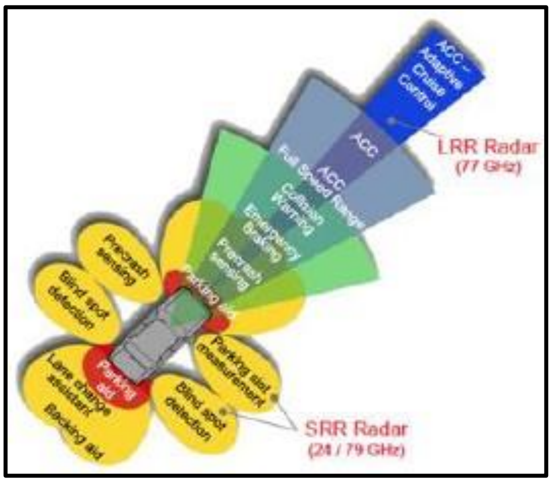


Figure 4.9 RADAR Sensors [14]

RADAR works on the principle of calculating difference between the transmitting and receiving signals due to the reflection from the obstacle. The distance is calculated by measuring the elapsed time between transmitter and receiver signal. The relative speed of the vehicle is based on the principle of Doppler effect. The speed of the detected object and its direction can be measured based on the frequency of the reflected signal and i phase shift. RADAR works on two different bands namely, 24GHz and 77GHz [11].

Chapter 5 Abbreviations

A	
ADAS	Advanced Driver Assistant Systems
AFE	Analog Front End
ASSP	Application specific standardized part
ADC	Analog to Digital conversion
ANN	Artificial Neural Network
AI	Artificial Intelligence
B	
BL	Block Length
F	
FFT	Fast Fourier Transform
B	
GCC	GNU Compiler Collection
I	
ICT	Information and Communication Technology
I2C	Inter-Integrated Circuit
L	
LED	light emitting diode
LIDAR	Light Detection and Ranging
M	
MCU	Micro Control unit
R	
R	Radio Detection and Ranging
S	
SCP	Secure Copy
U	
UDP	User Datagram Protocol

Chapter 6 References

[1] A. W. Mubina Toa, "Ultrasonic Sensing Basics," Texas Instruments, Texas, 2019.

[2] Robot-Electronics, "SRF02 Ultrasonic range finder," [Online]. Available: https://www.robot-electronics.co.uk/htm/srf02techI2C.htm.

[3] M. Gudino, "Engineering Resources: Basics of Analog-to-Digital Converters," Arrow Electronics, 17 April 2018. [Online]. Available: https://www.arrow.com/en/research-and-events/articles/engineering-resource-basics-of-analog-to-digital-converters.

[4] "FFT (Fast Fourier Transform) Waveform Analysis," DATAQ INSTRUMENTS, [Online]. Available: https://www.dataq.com/data-acquisition/general-education-tutorials/fft-fast-fourier-transform-waveform-analysis.html.

[5] "Fast Fourier Transformation FFT - Basics," NTI AUDIO, [Online]. Available: https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft.

[6] LogiLink, "Magnetic LED Light," https://www.manhattanshop.it/amfilerating/file/download/file_id/12125/.

[7] DRROBOT, "Gravity Digital Relay Module Arduino & Raspberry Pi Compatible SKU DFR0473," [Online]. Available: https://wiki.dfrobot.com/Gravity__Digital_Relay_Module__Arduino_&_Raspberry_Pi_Compatible__SKU__DFR0473.

[8] S. Campbell, "BASICS OF THE I2C COMMUNICATION PROTOCOL," DIY Electronics, 2016. [Online]. Available: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/.

[9] The Linux Foundation, "The Linux Kernel Archives," [Online]. Available: https://www.kernel.org/doc/Documentation/i2c/dev-interface.

[10] V. Kaplarevic, "How to setup a cron job in Linux," 9 January 2020. [Online]. Available: https://phoenixnap.com/kb/set-up-cron-job-linux.

[11] J. A. M. D. P. D. Michal Taraba, "Utilization of modern sensors in autonomous vehicles," IEEE, Mikulov, Czech Republic, 2018.

[12] A. K. J. M V Rajasekhar, "Autonomous vehicles: The future of automobiles," IEEE International Transportation Electrification Conference (ITEC), Chennai, India, 2015.

[13] N. J. a. V. D. Jelena Kocić, "Sensors and Sensor Fusion in Autonomous," Telecommunications Forum (TELFOR), Belgrade, Serbia, 2018.

[14] P. R. R. Tummala, "Autonomous Cars:Radar, Lidar, Stereo Cameras," IEEE-CPMT Workshop – Autonomous Cars, 2018.