**Data Setup:**

Reliability test and improvement of a sensor system for object detection

Information Technology

Modules Autonomous Intelligent Systems and Machine Learning

By Dr. Peter Nauth & Dr. Andreas Pech

|  |  |
| --- | --- |
| Deblina Karmakar  1427365 | Indranil Saha  1427190 |
| deblina.karmakar@stud.fra-uas.de | indranil.saha@stud.fra-uas.de |

1. Data Collection

1.1 Hard Surface Measurements

We measured for hard surfaces at two distances: 1 meter (m) and 50 centimetres (cm). The measurements were taken to assess. For each distance, we obtained 1000 measurements.

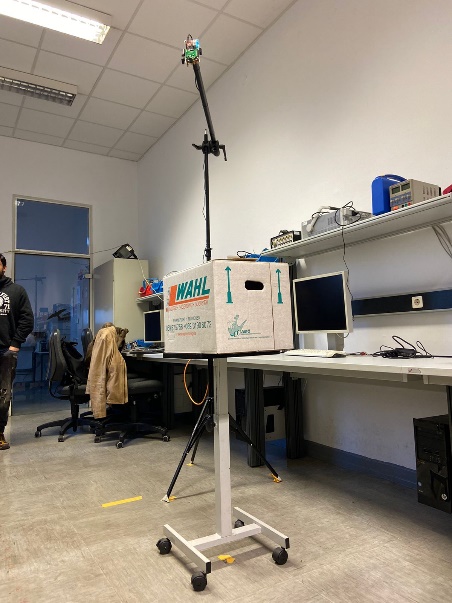
**Distance**:

1 meter (m)

50 centimetres (cm)

**Number of Measurements**:

1000 readings for each distance



1.2 Soft Surface (Person) Measurements

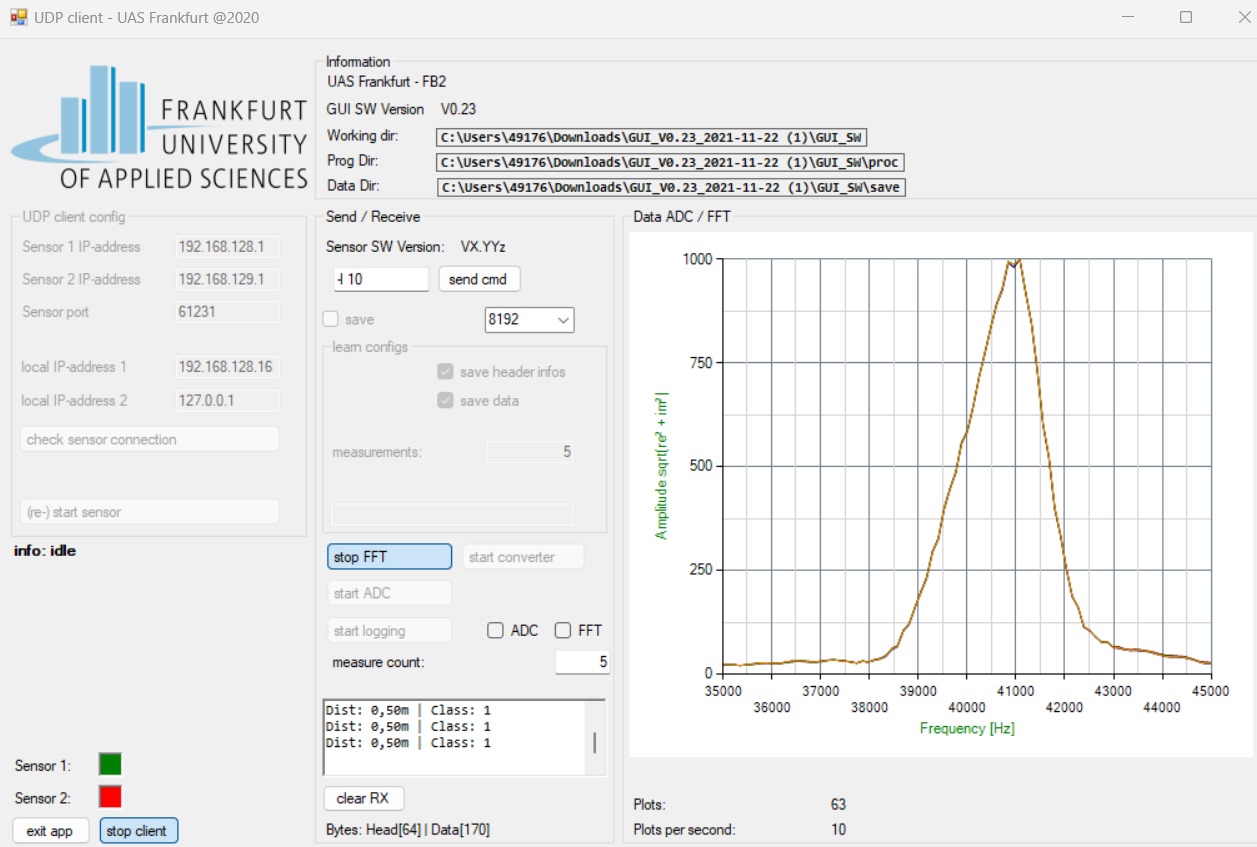
Similar measurements were taken on soft surfaces to take the readings. The following details outline the data collection process:

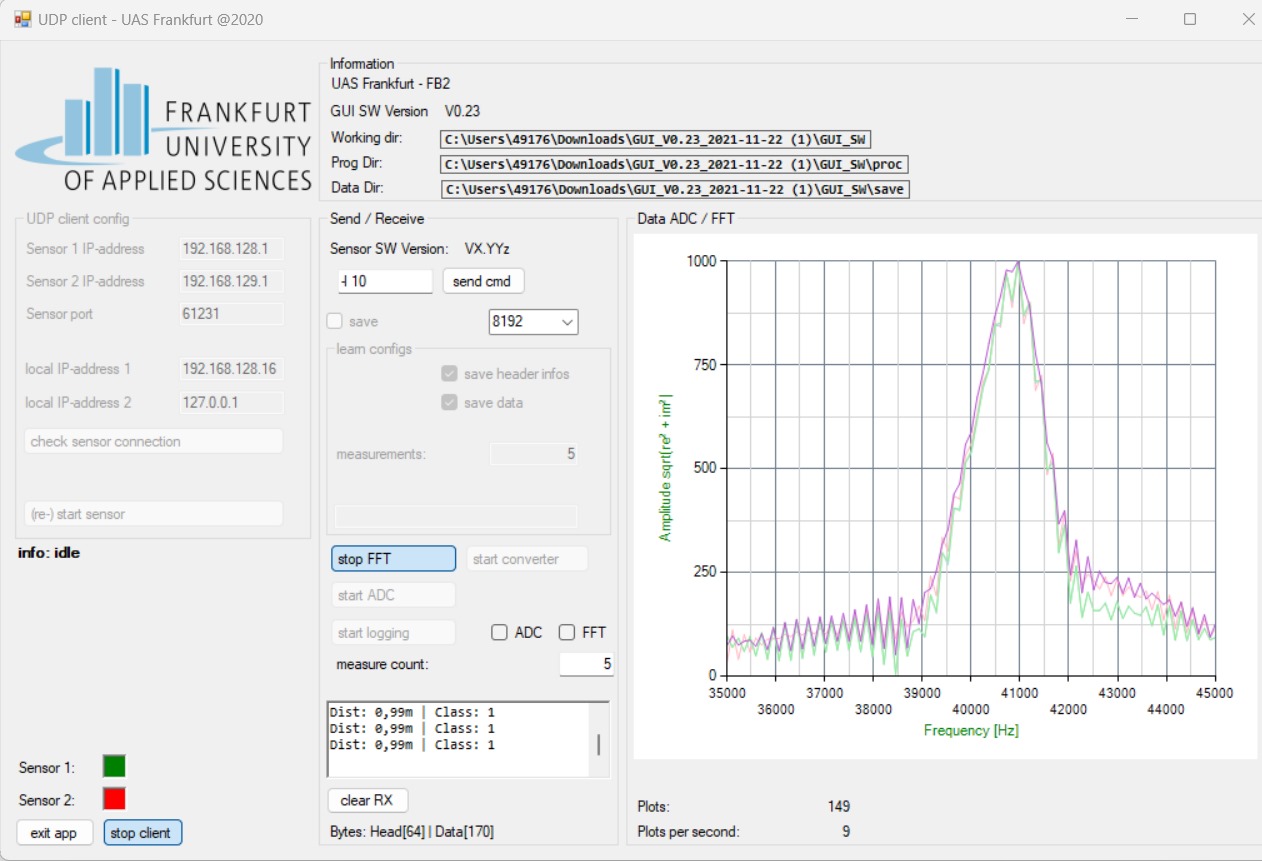
Number of Measurements:

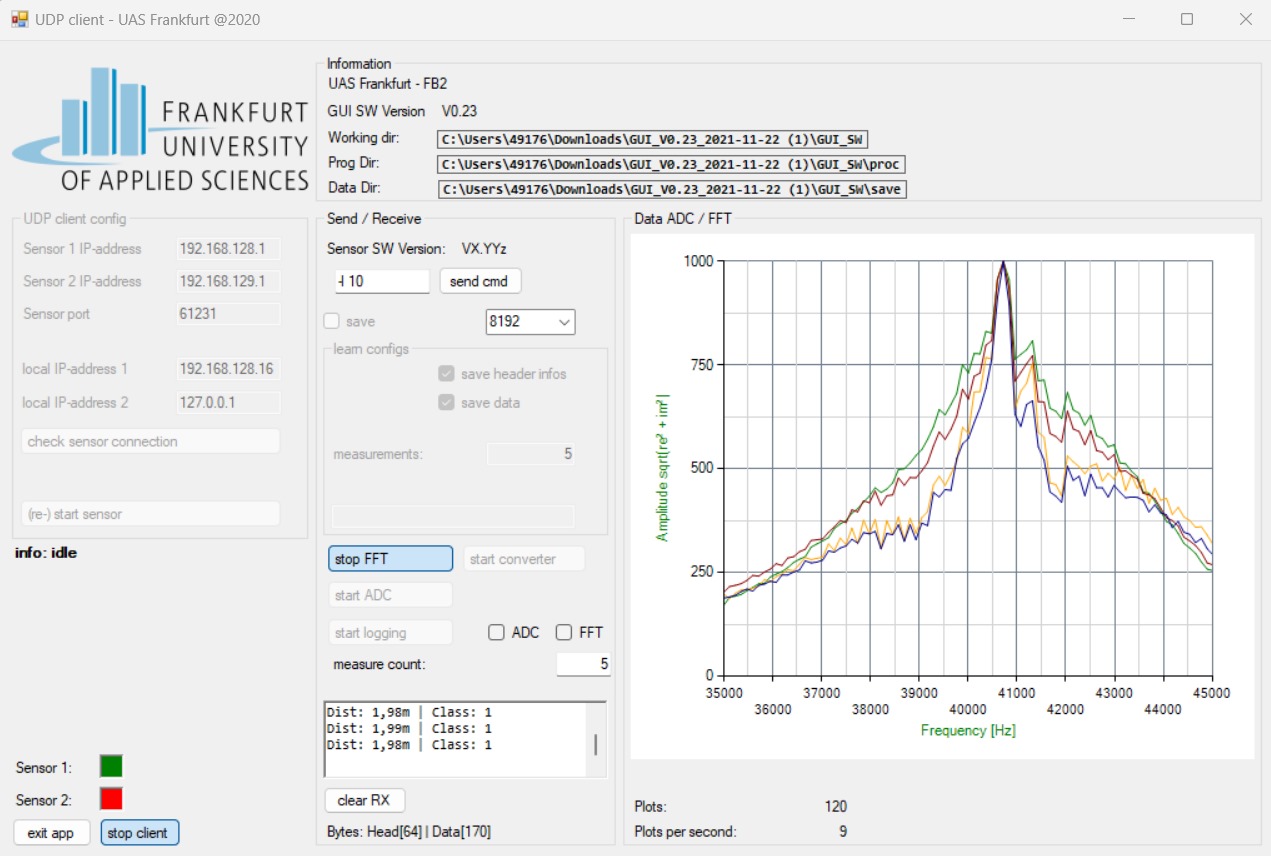
1000 readings



|  |
| --- |
| 1. **FIUS inbuilt distance measurement (dFIUS):** |







|  |
| --- |
| 1. **2. Manually measured distance using a folding meter stick (dMAN):** |

**Comparison**:

|  |  |
| --- | --- |
| **dFIUS** | **dMAN** |
| 1m | 0.99m |
| 50cm | 0.50m |

**Python program that automatically calculates and displays the distance dML between the sensor and the first reflection using the ML first echo detection algorithm given using the Source code.**

**Approach I:**

import smbus

import time

import numpy as np

from scipy.signal import find\_peaks

# Constants

I2C\_ADDRESS = 0x70

V\_SONIC\_WAVE = 343.2 # Speed of sound in m/s

ADC\_START\_DELAY\_US = int(0.30 \* 2 \* 1e6 / V\_SONIC\_WAVE)

ADC\_BUFFER\_DELAY\_US = int((16\_384 \* 512) / 1e3) # Assuming ADC\_BUFFER\_SIZE is 16384

ADC\_MID\_US = ADC\_START\_DELAY\_US + (ADC\_BUFFER\_DELAY\_US // 2)

# Setup I2C

bus = smbus.SMBus(1) # Typically 1 for /dev/i2c-1

def start\_ultrasonic():

# Write to I2C to start ultrasonic measurement

bus.write\_byte\_data(I2C\_ADDRESS, 0, 0x52) # 0x52 is a placeholder command

def read\_ultrasonic():

# Read from I2C

high\_byte = bus.read\_byte\_data(I2C\_ADDRESS, 1)

low\_byte = bus.read\_byte\_data(I2C\_ADDRESS, 2)

time\_of\_flight\_us = (high\_byte << 8) | low\_byte

return time\_of\_flight\_us

def get\_adc\_data():

# Placeholder for actual ADC data acquisition

# Replace with actual ADC reading code

return np.random.randn(16\_384) # Random data to simulate ADC data

def calculate\_distance(time\_of\_flight\_us):

return V\_SONIC\_WAVE \* (time\_of\_flight\_us / 2e6)

def measure\_distance():

start\_ultrasonic()

time.sleep(ADC\_START\_DELAY\_US / 1e6) # Convert microseconds to seconds

adc\_data = get\_adc\_data()

time.sleep(ADC\_BUFFER\_DELAY\_US / 1e6) # Wait for ADC buffer to fill

time\_of\_flight\_us = read\_ultrasonic()

distance = calculate\_distance(time\_of\_flight\_us)

return distance

# Example usage

if \_\_name\_\_ == '\_\_main\_\_':

distance = measure\_distance()

print(f"Measured distance: {distance} meters")

**Approach II:**

import numpy as np

import matplotlib.pyplot as plt

from redpitaya.overlay.mercury import mercury as overlay

def ml\_first\_echo\_detection(signal, sampling\_rate):

# ML first echo detection algorithm

threshold = 0.5 # Adjust threshold based on your specific use case

peaks, \_ = find\_peaks(signal, height=threshold)

if len(peaks) > 0:

return peaks[0] / sampling\_rate

else:

return None

def main():

# Initialize Red Pitaya overlay

ol = overlay()

rp = ol.red\_pitaya

# Parameters

sampling\_rate = int(rp.fpga.api.osc\_frequencies\_get(0))

# Configure and acquire data from channel 1 (assuming ultrasonic sensor is connected to channel 1)

rp.reset()

rp.waveform\_acquisition\_start()

rp.waveform\_acquisition\_trigger()

# Wait for acquisition to complete

while rp.waveform\_acquisition\_completed() == 0:

pass

# Retrieve acquired data

data = rp.waveform\_acquisition\_data(0, -1)

# Apply ML first echo detection algorithm

detected\_distance = ml\_first\_echo\_detection(data, sampling\_rate)

# Display results

if detected\_distance is not None:

print(f"ML First Echo Detected at a distance of {detected\_distance:.4f} meters.")

else:

print("No echo detected.")

# Plot the acquired data

plt.plot(data)

plt.title('Acquired Ultrasonic Signal')

plt.xlabel('Time (samples)')

plt.ylabel('Amplitude')

plt.show()

if \_name\_ == "\_main\_":

main()