**Touch Recognition frame for non-touch screen laptops**

**Abstract:**

The advent of touch screen technology has revolutionized the way we interact with digital devices, offering a more intuitive and immersive user experience. However, a significant portion of the existing laptop user base still relies on non-touch screen devices. This project introduces a novel solution to bridge this gap – a touch-responsive frame designed to seamlessly integrate with non-touch screen laptops. By retrofitting traditional laptops with touch capabilities, this innovation aims to enhance user interaction, productivity, and overall user satisfaction. This article describes an adoptable and customizable touch recognition system, that uses a sensor frame and a desktop application that runs code files with respective to the given laptop/desktop there by producing a device independent adoptable solution for non – touchscreen laptops. Increasing the hand tracking accuracy using a time-based recognition algorithm. Using small sized Time of Flight (ToF) sensors to measure an optimum distance with reduce the space and size of the device. At the end, developed devices and applications-based hand tracking through ToF sensors are discussed. Limitations, future aspects and research directions related to this filed are also discussed.

**Keywords:** human-computer interaction, ToF sensors, device adoptable, dynamic code running.

**1.Introduction:**

In the modern era, computing technology has seamlessly integrated into all facets of our daily existence, making human-machine interaction increasingly unavoidable. There is a prevailing belief that the continuous advancement of computer and display technology is inevitable. The interface enabling human communication with machines and computers is termed the human-computer interface (HCI). Traditional HCI methods, such as keyboards, mouse are considered somewhat limiting for user interface development. In contrast, the utilization of touchscreen is seen as a more natural and effective means of facilitating interaction between humans and computers. But over 80% current laptops/desktops in use are non-touch screen. The widespread adoption of touch screen laptops may diminish the usage of non-touch laptops, potentially contributing to an increase in electronic waste. So, to avoid this reuse of such laptops needs to be promoted.

Using a non-adoptable touch responsive frames/technologies can result in device specific production. Which further increase e-waste indirectly. A highly adoptable device is a demanded solution. Instead of just relaying on hardware, this article explains how to integrate the hardware with a software for making it highly adoptable. The hardware part of made using vl53l0x sensors that are small in size and measure a range of 40cm which is optimum for a laptop/desktop. The software part consists of a desktop application made using PyQt5 and a python file for fetching data from hardware using pyserial and control the GUI of the system using pyautogui.

[**“Research on Gesture Recognition System Using Multiple Sensors Based on Earth’s Magnetic Field and 1D Convolution Neural Network”**] In sensor recognition commonly used sensors are myoelectric sensors, flexible resistive materials, optical sensors, acceleration sensors, gyroscopes, magnetic field sensors etc. Multiple sensor redundant measurements methos commonly used include computer vision, accelerometer and flexible sensor composites; accelerometer and magnetic field sensor composites, acoustic sensor composites. The data form sensors are processed through setting thresholds, machine learning and other algorithms to achieve gesture recognition.

**Working:**

The frame is made up of VL53L0X (Time of Flight) sensors. Whenever the hand interaction takes place. The sensor measures the distance from the finger to the frame and the data is read by the MegaPro2560 and share it to the desktop application. The desktop application is built using the pyQT5 library using python. The desktop application consists a python file that reads the data from processor. The file is system dependent, therefore to make the whole system independent. Respective files for each laptop are made available in a web application and made is to download and use. By selecting the desired file for the system and running the desktop application. The file reads the data and identifies the sensor from which the data is extracted by decoding the data. Based on it the Graphical User Interface (GUI) of the system is accessed via pyautogui library and the cursor is mode to the (X, Y) coordinate where the Y is determined by the position of the sensor in the frame and the X is determined by the distance value received from the sensor. Since, working with real time data can lead to errors in the output, a time recognized algorithm is used to overcome such errors in the system. By calculating the previous time data and current time data and measuring the difference between them and taking difference as the error factor and taking necessary changes.

The Vl53L0X sensors are I2C devices in MegaPro2560, The I2C lines will be used for the connection with the SCL and SDA pins. The MegaPro2560 has pins D20 and D21 as SDA and SCL pins respectively. Connecting more I2C devices requires specific memory address for each device to share the information. Since the device uses over 15 I2C devices this can result in memory space conflicts to overcome this we require a microcontroller with more memory space and digital pins, therefore MegaPro2560 is used in this application because of the following specifications.

|  |  |
| --- | --- |
| **Specifications** | **Properties** |
| Digital I/O pins | 54 (with 15 PWM pins) |
| Flash Memory | 256 KB of which 8 KB used by bootloader |
| SRAM | 8KB |
| EEPROM | 4KB |

During I2C communication the shorting of SCL and SDA pins can result in high peak voltage fluctuations that result in microprocessor failure. Therefore, such situations can lead to component replace. So, to avoid that proper connection analysis is made and even the SCL and SDA pins are used as common for all the I2C devices but overlapping of those pins are avoided in the PCB layout.

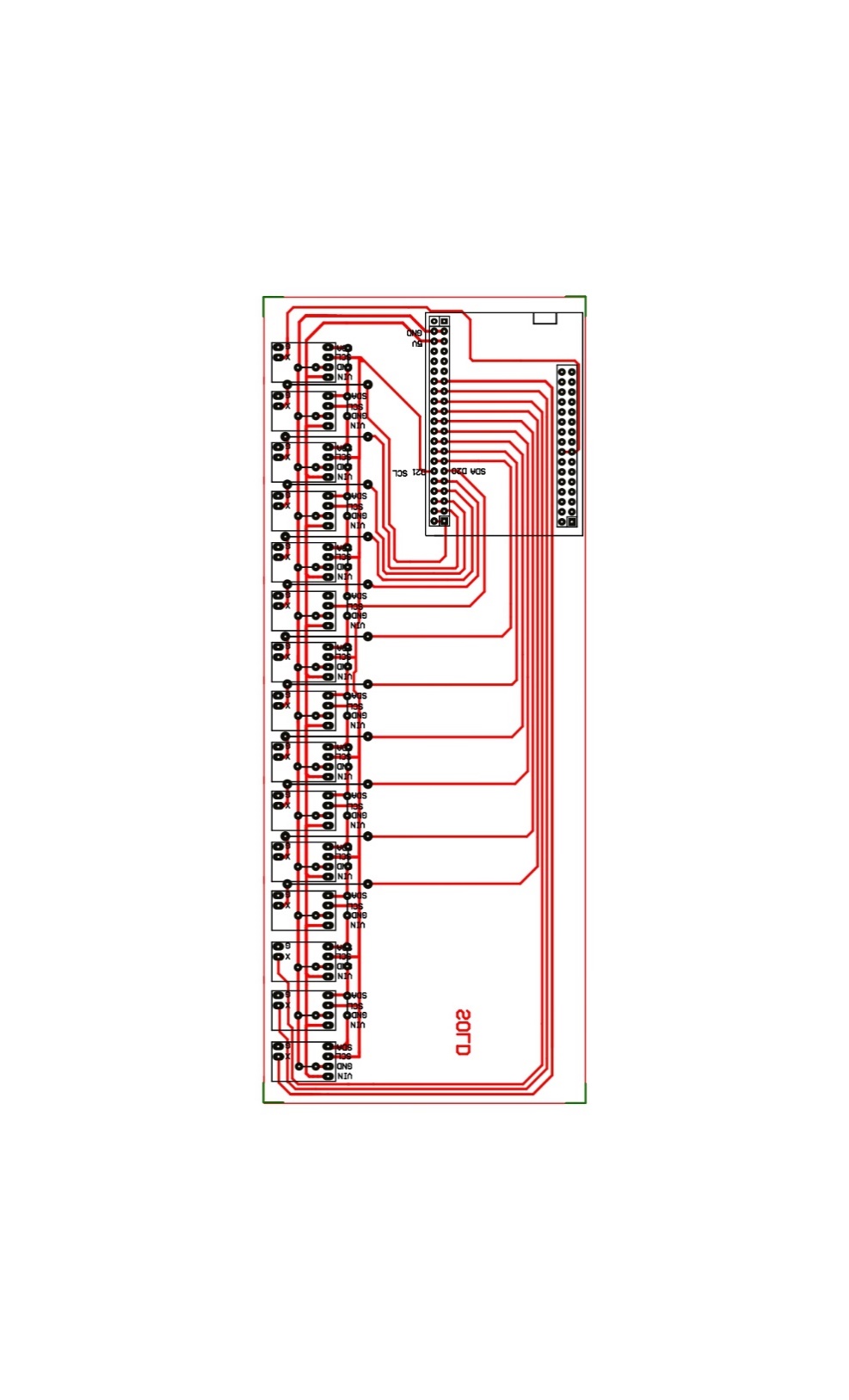
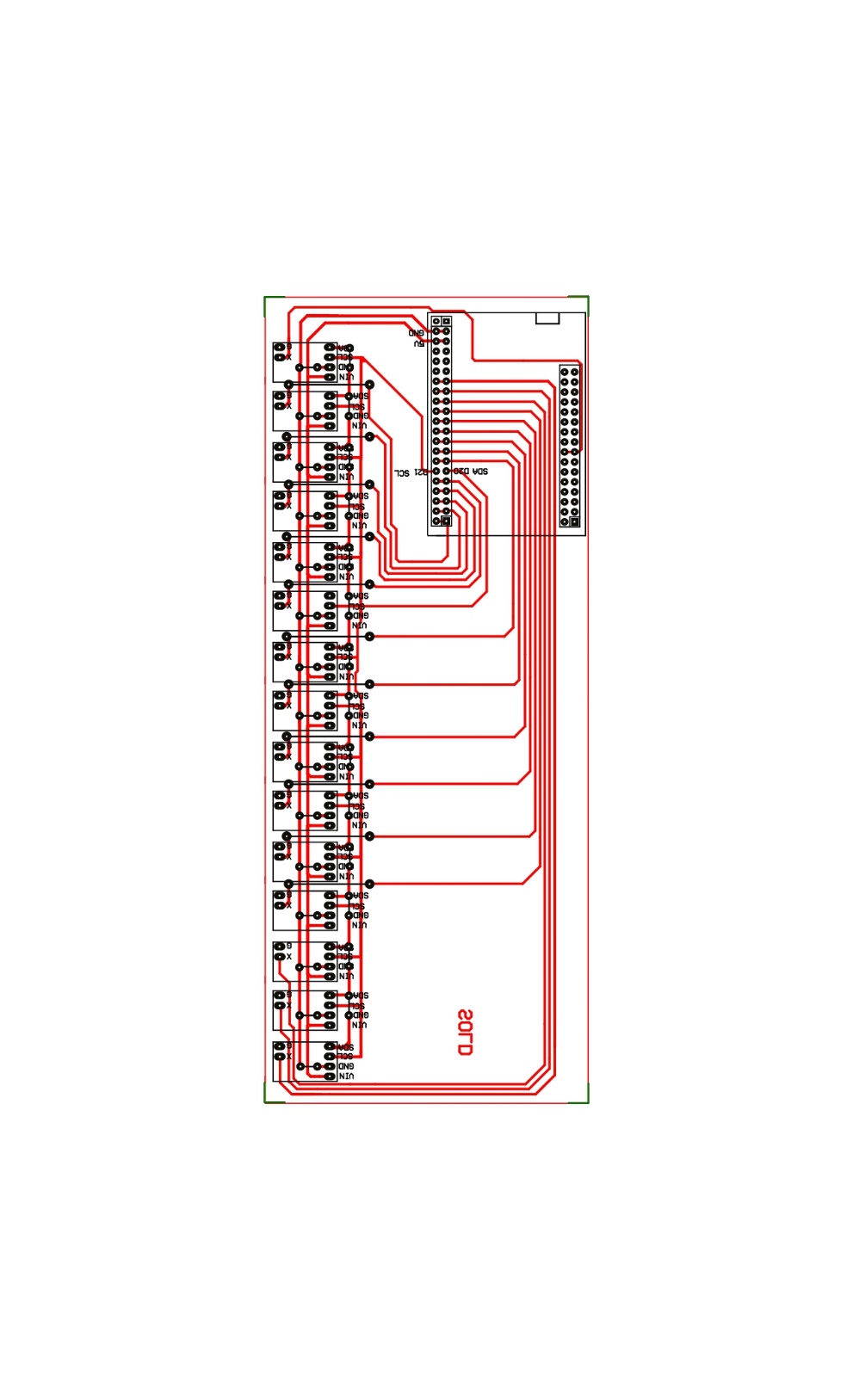


Fig: PCB Design

All the VL53L0X pins are connected in Vin, GND, SCL and SDA pins in common, therefore to access them separately the XSHUT pin is used to shutdown VL53L0X devices that are not in use. While assigning XSHUT pins there are chances that data conflicts take place with adjacent pins during Real Time data fetching. This can be overcome by utilizing the extra digital pins of the MegaPro2560. Each digital pin is connected to its adjacent digital pin and made a separate pair as shown in the below diagram. During coding anyone of the pins in the pair is used not the both. Poor connection or coding in this method can lead to component failure. So proper care needs to be taken during coding.



Pairing of pins

The VL53L0X measures the data in millimeters. During this process, while out of range it gives a value of about 8191. This is used to define when sensor is uninterrupted. No actions need to be taken when this value occurs.

There are [microcontrollers](https://www.arduino.cc/reference/en/language/functions/usb/keyboard/) such as Leonardo, Micro, Due etc. that support Mouse and Keyboard access using mouse and keyboard libraries. But their Mouse libraries lack several options such as to move the cursor to a particular coordinate and to check the current position of the cursor. Only relative coordinate movements are possible. Whereas python has several libraries such as pyautogui, pygetwindows, pywinauto that has direct GUI access with necessary mouse and keyboard functions. The hardware data from the Arduino can be fetched by the python using pyserial library. When you read data from Arduino using Python and Pyserial, the data is typically transmitted as bytes. In the context of serial communication, each character or piece of data is represented as a byte (8 bits).

The byte format you see is a binary representation of the data. Each byte can represent a character, number, or other information depending on how the Arduino is configured to send data. To interpret the data correctly, you need to know the data format set on the Arduino and decode the bytes accordingly in your Python script. For example, if you're sending ASCII characters, you can use the decode() method to convert the received bytes to a string.

import serial

ser = serial.Serial('COMx', 9600) # Replace 'COMx' with your Arduino's serial port

while True:

data = ser.readline()

decoded\_data = data.**decode('utf-8')** # Assuming UTF-8 encoding

print(decoded\_data)

The data from the microcontroller is sent in the format

(**Sensor position: distance measured)**

This helps to identify from which sensor the data is received in the python. In python, the pyautogui library is used for performing mouse and keyboard functions, because it has more specific required functionalities when compared with other GUI access libraries in python. The sensor position and distance measured are used to fix the y and x coordinates respectively. The average length of a human finger’s tip is around 0.5cm, also each VL53L0X sensors are of 0.5cm. Therefore, each sensor can be placed one after other in a vertical manner, no gaps or spot of non-recognition takes place in using VL53L0X in y axis. For x axis, since the output of VL53L0X sensors are measured in millimeters, every 50mm(0.5cm) increments the x axis value by 1.

|  |  |
| --- | --- |
| Distance Measured | X coordinates |
| 0 | 0 |
| 5 | 30 |
| 10 | 60 |
| 15 | 90 |
| 20 | 120 |
| 25 | 150 |

|  |  |
| --- | --- |
| Sensor Position | Y coordinates |
| 1 | 0 |
| 2 | 50 |
| 3 | 100 |
| 4 | 150 |
| 5 | 200 |
| 6 | 250 |

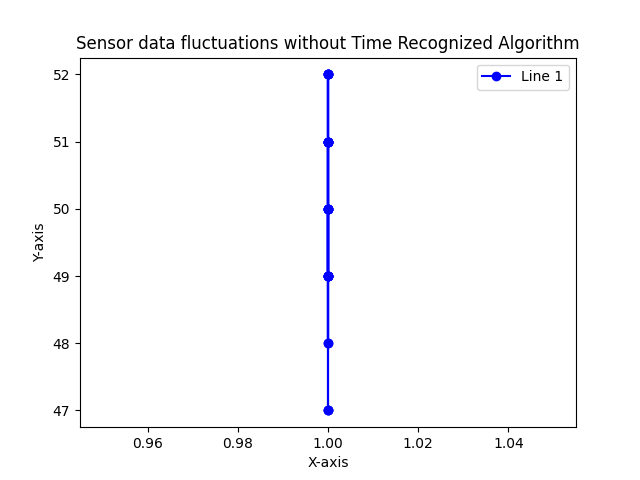
For real time application we will be using an infinite loop for reading data continuously and performing action. During this loop after performing every mouse and keyboard functions such as click() and press(), it is necessary to perform the reverse process such as keyUp() and mouseUp() respectively. In order to avoid unintended extra mouse clicks and key presses.

The VL53L0X sensors have an accuracy of about 98% in measuring distance. But using multiple sensors leads to overlapping of sensor data during Real Time analysis. This can result in unexpected outcomes. To avoid this a **Time Recognized algorithm** is used. Which considers the current time data and previous time data from the sensor using the ‘time’ library of python. The difference of data from both current and previous time is taken and the value of error is calculated. Based the value of error respective actions will be taken place as remain in the same position or move to the next position. The performance of Time Recognized algorithm is discussed in the results and observation section.

In Airbar the device is place on the bottom of the screen, the device placement depends upon the speakers available at the monitor frame, which is not suitable for desktops and laptops with no such magnets. Therefore, the signed aligned placement of this frame makes the device convenient to use with any kind of laptops and desktops.

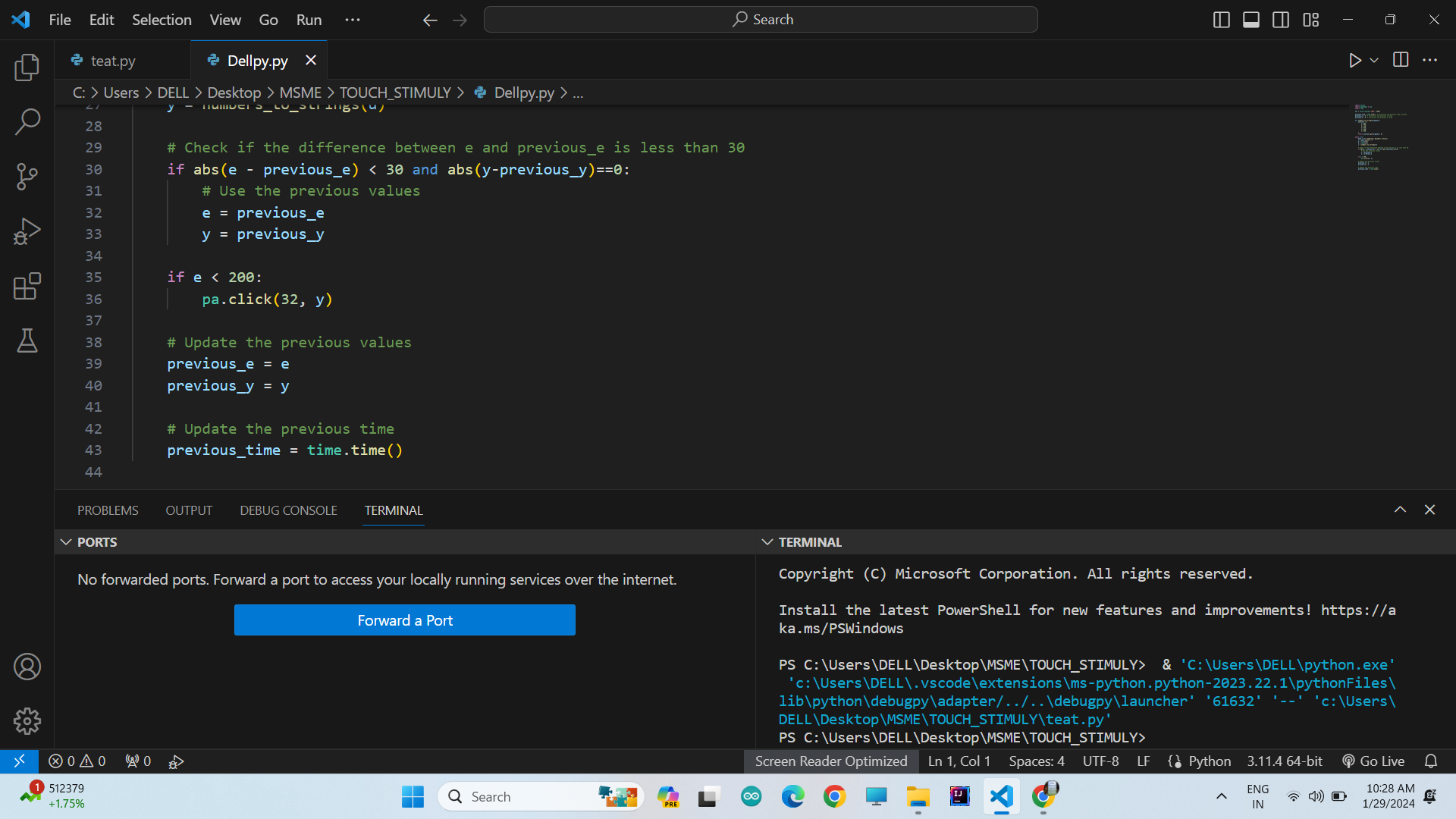
**Results and Observations:**

To analyze the system performance, the fluctuations of data in a single sensor is measured over a time period of 60 milliseconds. An object is placed over the axis of sensor at a measured distance and compared with the data received from the sensor. In the first observation without the use of Time recognized algorithm visible fluctuations in data are noticed. In the below fig.3 the graph shows how the data starting at a value of 47 fluctuates to 52 in 60 milliseconds. These observable errors are common in real time systems, but in a Human Computer Interaction System, such errors can drastically affect the user experience and leads to unexpected behavior of the system.

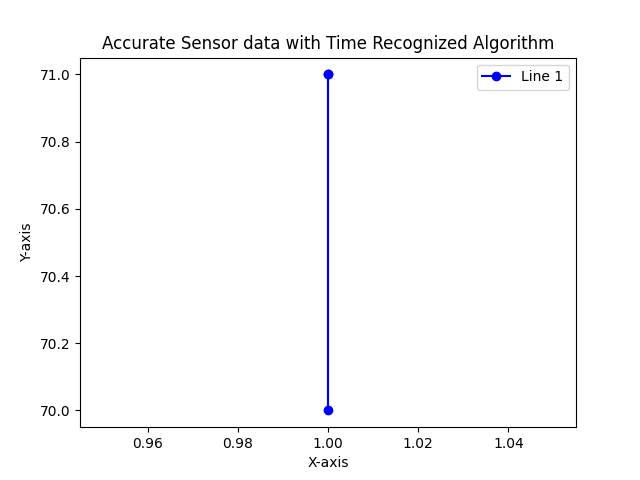


But by applying the Time Recognized algorithm such fluctuations can be reduced to a minimum error rate or no error rate.

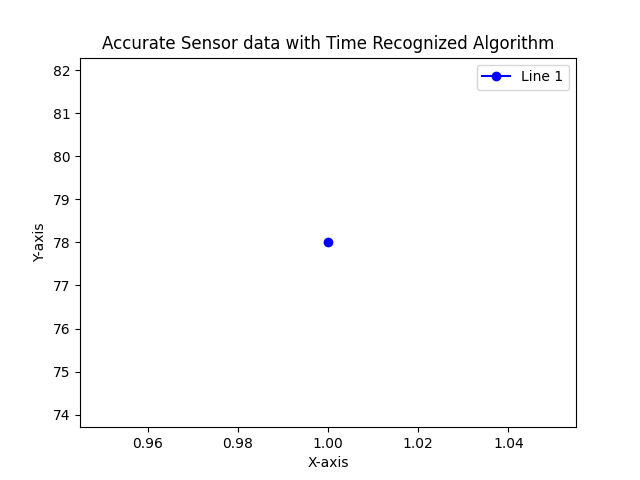
**Time-Recognized algorithm:**



In the below graph fig.4 working on the same time period of 60 milliseconds, the sensor data value of 70 is fluctuated to 71. where the error rate is just 1. Which doesn’t make any noticeable change in the output of the system.



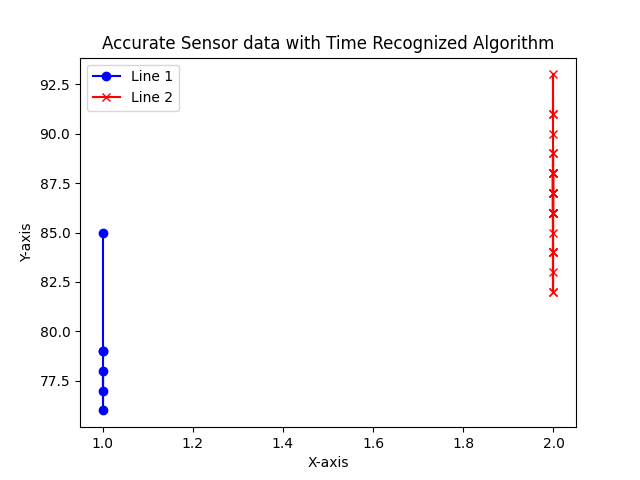
While this algorithm can be even more accurate in some situations. Consider, the below graph Fig.5 the sensor data has a pin point value of 78. So fluctuations happened in the time interval of 60 milliseconds and therefore the error rate is zero.



Using multiple sensors on a single I2C line can also cause potential errors since both sensors are active at the same time because the neighbor sensors can have unintentional touches, also the value of desired sensor can be suppressed with the value of undesired sensor. Those errors can also be rectified using this algorithm. Let’s consider the following cases.

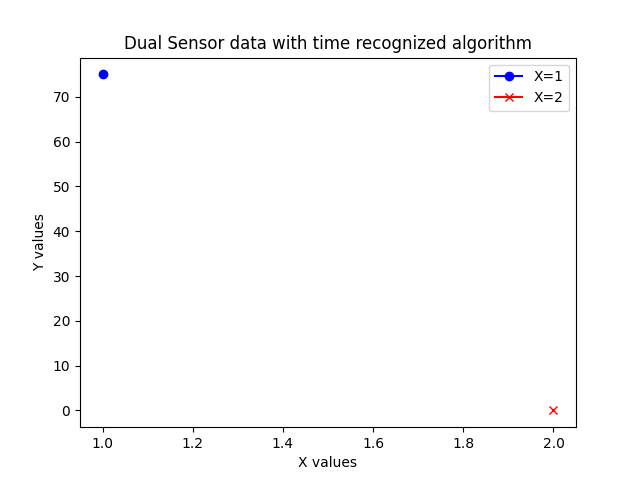
**Case:1** Fetching data of 2 sensors without using the algorithm

Here in the graph fig.6 the value of first sensor is ranges from 70 to 85 while the value of 2 sensor also produces values at a range of 82.5 to 92.5. These errors are highly noticeable and cause protentional failure in the system output.



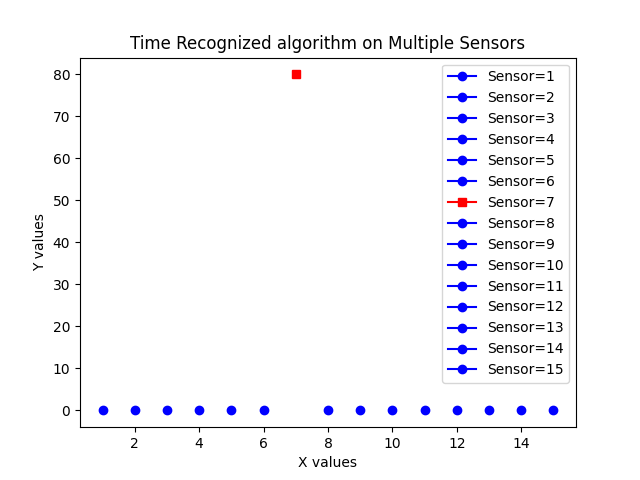
**Case:2** Fetching data of 2 sensors by using the algorithm

Here in the graph fig7. by applying the algorithm, the unwanted data from the sensor 2 are suppressed and now only the values from sensor two are considered therefore System performance is improved



**Case :3** Applying the algorithm on multiple sensors

This method also works fine in 15 sensors packed sensor frame. As shown in the below graph. Where the 7 sensor is in active state and gives the output whereas the other sensor values are suppressed.



**Summary:**

This method improves the overall user experience and the makes any screen touch responsive in irrespective of the inbuilt hardware in the laptop. The adoptability method makes the system easy to use in any laptop/ desktop. Thus, providing a hustle free experience. The implementation of Time recognized algorithm helps us to get a desired output even during real time processing. The system can be further improved in future with more functionalities than click() and move(). Further improvements, can helps to replicate all the functions that are possible with a touch screen laptops.