Summary Document:

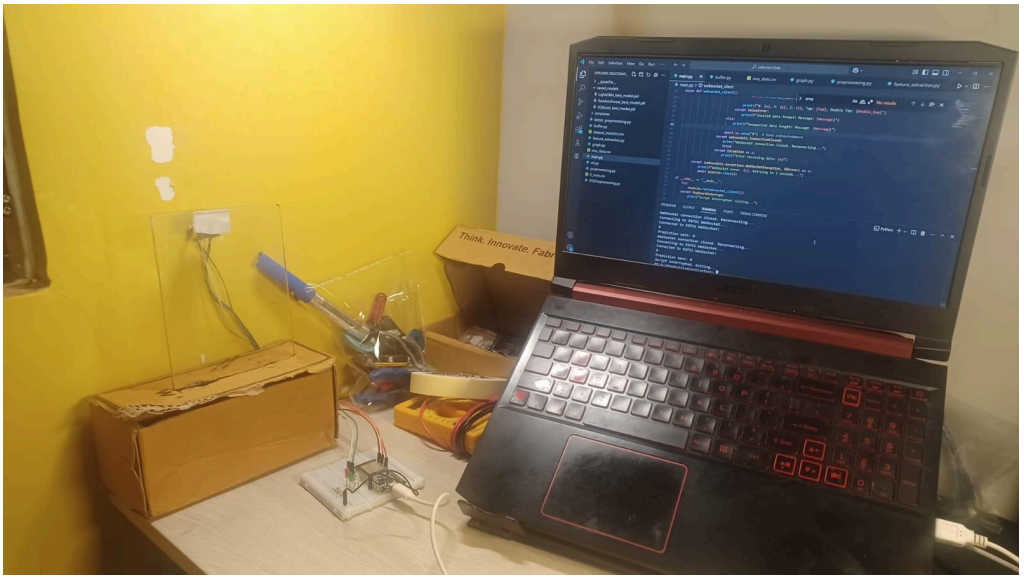
Vibration Sensing Touch Panel

Sustainability Lab
AI and Sensing for Sustainability

SRIP
IITGN

Applicant Details:

Name: Paranjay Chaudhary
Intern ID: 21935368



(*Fig 1: Overall Hardware setup; Do Not Mind The Mess!)*

1. Hardware Setup

The accelerometer chosen was the **ADXL345**, which was selected for its support of the I2C and SPI communication protocols. The ADXL345 has robust library support compared to other accelerometers, ensuring reliable sensor readings. It also has onboard hardware to detect if a tap or double tap has occurred, providing a simple yet powerful way of capturing target variables directly from the sensor itself.

I2C was chosen for communication between the sensor and the microcontroller due to its simplicity and reliability.

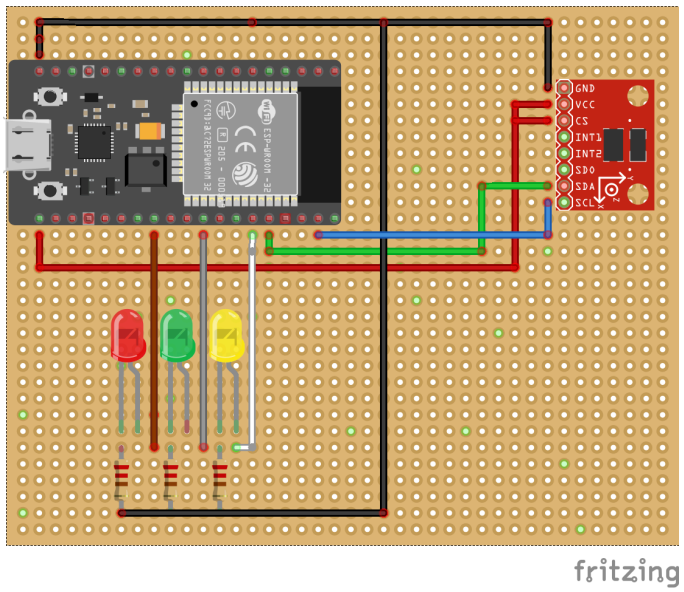
The library used was the **SparkFun ADXL345** sensor library due to its support for reading the registers where the values of the interrupt flags that indicate the tap events.

The **ESP32 WROOM32** was the microcontroller chosen for its high clock speed(80 - 240 MHz) while being simple to use compared to single-board computers such as the Raspberry PI. Another benefit was the ample interrupt-capable pins, ADC support, flash memory, and built-in Wi-Fi and Bluetooth capabilities.

Three **LEDs** were connected to display the following statuses:

- **Warning/Network Status**
- **Tap**
- **Double Tap**

These LEDs provide real-time feedback on sensor detection and help compare the hardware's performance against the machine-learning model.



(*Fig 2: Circuit Diagram)*

2. Software Setup

i. Communication Protocol

Communication between the ESP32 and the computer was established using **WebSockets** for bidirectional communication. **WebSockets** retain a high communication speed without compromising reliability, unlike UDP. This was implemented on the server side using the **AsyncWebServer** library on the ESP32, while the client side used Python's **WebSocket** and **asyncio** libraries. Although asynchronous communication was initially used, it was found to be unreliable and thus not extensively employed.

ii. Data Transmission

Data was sent wirelessly in the format of [X, Y, Z, Tap, Double_Tap], where **Tap** and **Double_Tap** are binary values (0/1). This data was logged in a CSV file, and a buffer was used during inference to store sensor values temporarily. Signal filtering was avoided due to the potential loss of information and limited knowledge of DSP.

iii. Machine Learning Models

Three machine-learning models were used for tap detection:

- **Random Forest**
- **LightGBM**
- **XGBoost**

These models were selected due to their strong performance on tabular data. The collected time-series data was converted into a supervised learning problem, and hyperparameter tuning was done using **GridSearchCV** and suitable parameter grids. An 80% test-train split was taken.

Cross Validation was done with Time Series Cross Validation **(TSCV).**

Feature selection was performed using **Recursive Feature Elimination (RFE)** with a Random Forest classifier, resulting in 25 selected features.

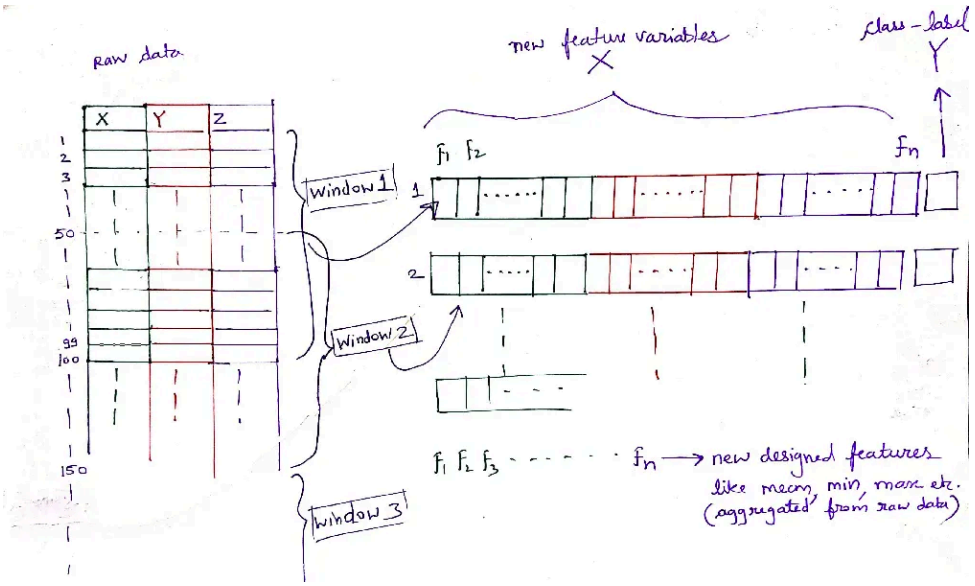
Model	Accuracy	Precision	Recall	F1 Score
RandomForest	0.98	0.97	1.00	0.98
LightGBM	0.98	0.97	1.00	0.98
XGBoost	0.97	0.96	1.00	0.98

(Table 1: Performance of ML Models Used when Not Detecting Taps)

Model	Accuracy	Precision	Recall	F1 Score
RandomForest	0.98	1.00	0.95	0.97
LightGBM	0.98	1.00	0.95	0.97
XGBoost	0.97	1.00	0.93	0.96

(Table 2: Performance of ML Models Used When Detecting Taps)

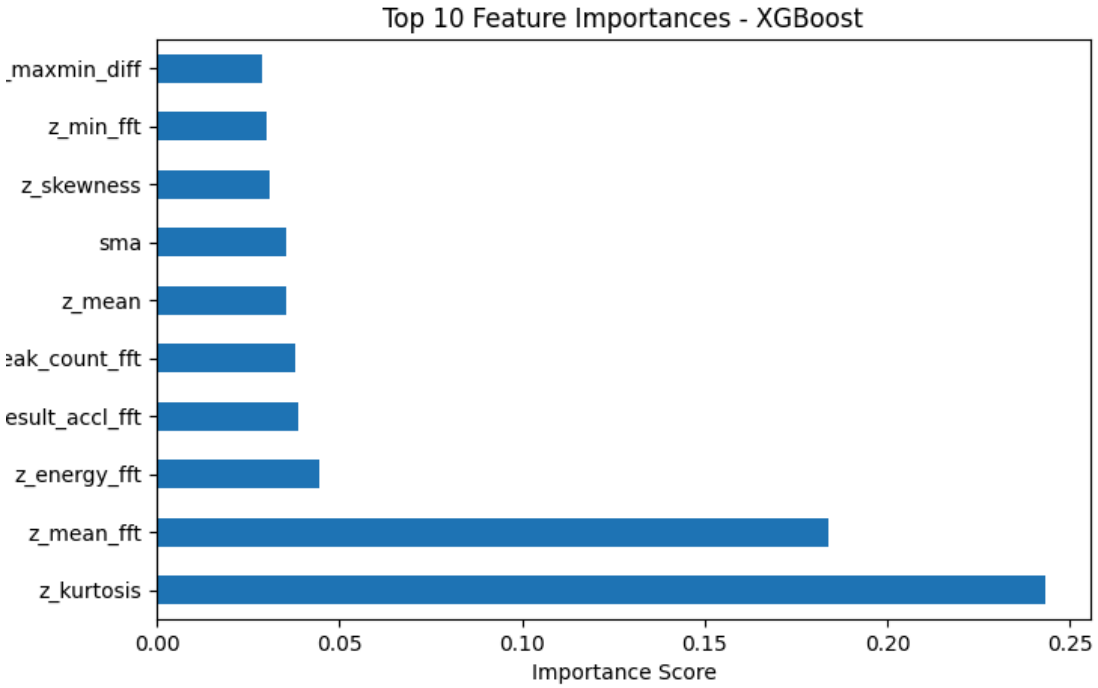
iv. Feature Extraction



(Fig3 : Sliding Windows With Overlap; Credited To: ***Pratik Nabriya*** *for this image and the preprocessing script; link to article)*

Feature extraction was performed using a sliding window approach with 100 samples per window (1 seconds at 100Hz) and 50% overlap (25 samples). Taking inspiration from Selection Task 1 (UCI HAR Dataset), statistical features such as mean, max, min, variance, and IQR were extracted from both the time and frequency domains using an **FFT**.

Additional features included **entropy**, **energy**, **signal magnitude area**, and **average acceleration**. All features were measured along the Z-Axis, the axis along which the user taps the glass. (Again, Credited to Mr. Pratik Nabriya)



(Fig 4: Plot of Feature Importances For XGBoost)

v. Inference Process

During inference, a **buffer** was used to store sensor values temporarily, and the same sliding window approach was used. Data from the buffer was then sent to a feature extraction function in the form of windows, which were then passed to a pre-trained model, and the prediction was forwarded to the ESP32.

3. Approach For Final Problem Statement

Another potential avenue of investigation would be to train an ML model with an **ADXL345**, then replacing the accelerometer with another, such as the **MPU6050, to** observe if this ML-based approach is independent of hardware. This could replace the current accelerometer with something much cheaper, allowing panels to use more sensors.

By dividing the glass panel into **three regions**, we can utilise the current hardware setup to predict which section a tap has occurred. If the ML model can do so, we can use this to help solve the final problem statement of accurately predicting where a tap has occurred on the glass panel.

Accomplishing this would allow us to extend this for four sensors; more sensors could allow for the use of a **Kalman Filter** (investigating sensor fusion).

Dividing the panel into a grid, the problem's complexity is now reduced to predict which grid was tapped by observing the outputs of the **four accelerometers**. The size of each cell in the grid could be decreased until model performance degraded, Increasing the precision of tap detection.

However, the current method of running inference might prove to be cumbersome. Despite using only one pre-trained ML model, feature extraction would require about **four times** the resources and time taken to process the outputs of one sensor, impacting its real-time performance.

4. Current Issues

- **Network Stability**: The connection between the computer and the hardware is prone to disconnections, possibly due to the ESP32 being overwhelmed by the data load.
- **Sensor Mounting**: The current mounting method (using double-sided tape) absorbs vibrations and reduces sensor sensitivity. The glass panel's movement in its cardboard box mount also affects stability.
- **User Interface**: The system currently lacks a user interface for tasks like switching models, creating training sessions, and managing data. A **Flask app with SocketIO** could be implemented to provide a local web server for easier interaction and monitoring.

5. Acknowledgements

I am writing to express my gratitude to the faculty at the **Sustainability Lab** at **IITGN** for providing this invaluable opportunity and keeping the internship open to all students.

I look forward to receiving your feedback and suggestions for improvement.