WheelPoser Touch Sense and Logging System

BLE-Based Touch Interface for Wheelchair Ground Truth Data Capture

Tatyana Cruz
tcruz@andrew.cmu.edu

May 5th, 2025

Contents

1 Abstract

This project presents the development and implementation of a Bluetooth Low Energy (BLE) enabled Touch Sense and Capture (TSC) system designed to support the WheelPoser project's goal of improving motion capture for manual wheelchair users. The system uses capacitive touch sensing to detect contact between the user and the wheelchair pushrim, capturing ground truth data for pose estimation. The hardware is built around the Arduino Nano 33 BLE, which integrates capacitive sensing and wireless communication into a compact, wheel-mounted form factor. Sensor data is processed locally using filtering and debounce logic, then transmitted in real time to a Python-based client application. The client supports both live visualization and structured data logging, enabling immediate feedback and post-session analysis. This bidirectional architecture provides a lightweight and low-power solution for capturing ,with reasonable estimation, with high-resolution touch interaction data. The current prototype demonstrates stable BLE connectivity, accurate session logging, and robust touch visualization, establishing a foundation for future integration with the larger WheelPoser Project.

2 Introduction

Motivation: Enhancing motion capture for manual wheelchair users. The overall objective of the WheelPoser [1] project is to create accurate representation of the pose of wheelchair users. This enhanced pose information would allow users to have better access to their health data allowing them to monitor their health and prevent injuries. The Touch Sense and Capture (TSC) System was created to aid in the data capture portion of the larger Wheelposer project.The touch data captured would be used as ground truth data to help track the time of the user points contact and release.

Use of capacitive touch sensing and BLE for mobility tracking.

Capacitive Touch Sensing: Touch capacitive sensors are rather common, used in touch screen devices and proximity sensing for both small scale and industrial applications[3] . In the context of this project the primary goal was to utilize the touch capacitive sensing to capture the touches of wheelchair users. The use of capacitive sensing was motivated by size and usability constraints. The simplicity and the ability to have a small form was motivated by the wheelchair application of the sensor. The size of the peripheral device had to be small and light enough to comfortably fit in the area between the spokes of the wheel, since the user would be actively pushing the wheel. In terms of usability, the ability to create sensing points with the application of conductive material around the wheel was the second point of reasoning for using Capacitive Touch Sensing. This approach would prove to cause challenges with drift and as an effect sensing accuracy (See Challenges and Limitations).

Bluetooth Low Energy (BLE) transmission: The TSC system was designed to detect user touch with a single wheel. In order for users to freely use the wheel and rotate, there would need to be a separation of the sensing

hardware and the data processing software. To accomplish this data transmission, the use of BLE was used for its wireless data transmission and low energy consumption[2]. While BLE has a maximum distance for use, the relatively short distance between the client device and the peripheral made this constraint nominal. The low energy consumption of BLE allows for larger usage when battery-powered, which assists in allowing for smaller sized hardware systems. Using BLE also allowed for the usage of the Arduino Nano 33, a small IOT device,for hardware and firmware processing, as BLE is integrated into the board.

Summary of hardware and software setup.: The system architecture integrates the capacitive sensing interface with wireless BLE data transmission to facilitate real-time acquisition and visualization of touch input from the wheelchair pushrim. User interactions with the pushrim surface result in variations in the capacitive sensor signal, which is processed locally on an Arduino Nano 33 BLE microcontroller and post-processed on the client side. The onboard firmware performs continuous sampling and applies threshold-based filtering to discern touch events from signal noise. Touch events, along with optional raw sensor values, are transmitted to a client device,in most cases a computer, via Bluetooth Low Energy (BLE). This wireless communication enables untethered data collection, preserving the natural mobility of the user during operation. On the client side, a Python-based client application interfaces with the BLE peripheral to receive, decode, and manage incoming data. The software,functionality this is done with separate scripts, supports two primary modes of interaction:

Live Visualization: Real-time plotting tools are used to display touch events and sensor signal magnitudes, allowing for immediate visualization assessment during data collection sessions.

Data Logging: Received data is saved into CSV format line by line, including timestamps, signal values, and binary touch state indicators. These structured logs support quantitative analysis after the data collecting session. This bidirectional hardware-software pipeline ensures a framework to collect touch data and as a result generate ground truth data for touches.

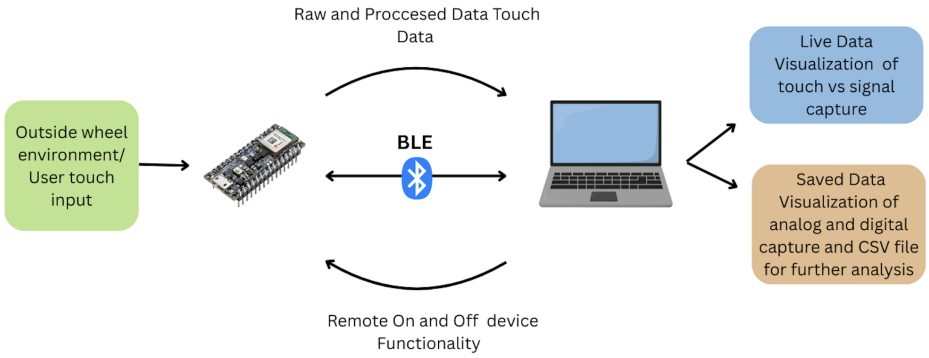

Figure 1: Hardware and Software Interaction

3 Design Overview

3.1 Hardware

The hardware layout is intended to work in tandem with the Arduino Nano 33 BLE, which serves as both the capacitive touch sensing controller and the BLE transmitter. The system uses a two-wire capacitive sensor configuration:

Pin 4 (Send Pin): Connected to one side of a resistor (R1), which is in series with a capacitor (C1) connected to ground. This setup forms a part of the capacitive sensing circuit. The pad or touch plate is also connected at this node, allowing it to respond to changes in the surrounding electric field when touched.

Input 1 (Blue Wire):This wire is attached at the same point as Pin 4. The opposite end is attached to the touch pad or conductive surface.

Pin 2 (Receive Pin): Connected through a second resistor (R2) to a separate signal input line, marked as RED_IN. This pin receives the influence of the touch interaction and completes the capacitive sensor pair used by the CapacitiveSensor library.

Input 2 (Red Wire):This wire is attached in series with Pin 2 and R2. The opposite end is attached to the touch pad or conductive surface.

Ground: Connected directly to the capacitor, stabilizing the measurement and reducing environmental noise.

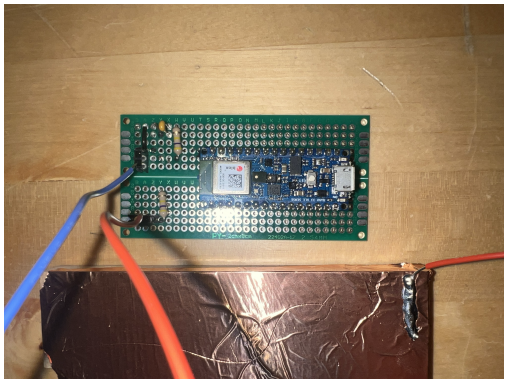

Figure 2: PCB design

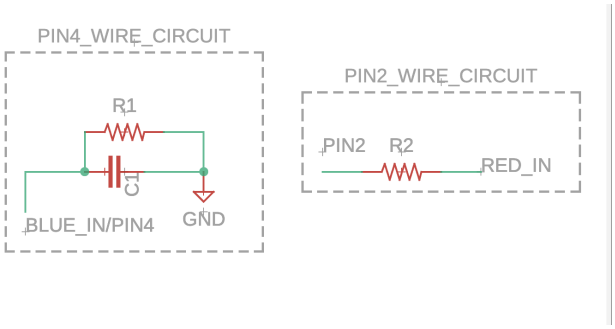

Figure 3: Circuit Diagram

Parts List for Capacitive Touch Sensor Circuit

Component	Label	Description	Value / Notes
Resistor	R1	Pull-up resistor with capacitor	470 kΩ (typical)
Capacitor	C1	Capacitor for noise filtering	Ceramic, 10–100 nF
Resistor	R2	Series resistor on sense pin	470 kΩ (recommended)
Microcontroller	–	Arduino Nano 33 BLE	BLE-enabled
Connection Wire	–	From Pin 4 to touch plate (BLUE_IN)	Shielded jumper if needed
Connection Wire	–	From Pin 2 to plate (RED_IN)	Touch input sensing line
Ground Connection	–	Ground for capacitor circuit	Connect to Arduino GND

3.2 Arduino Software

The TCS system integrates a capacitive touch sensor with a Bluetooth Low Energy (BLE) communication framework to capture and log manual wheelchair propulsion data. The hardware platform is built around the Arduino Nano 33 BLE, which features a built-in BLE module.The software utilizes the built-in LED on the Arduino Nano 33 to show when the controller senses touch. The sensor readings are filtered using a baseline calibration and and heuristic function for a thresholding algorithm to reduce noise and false positives.

3.2.1 Touch Sensing Algorithm

Baseline calibration and signal filtering. The baseline calibration is used through out the script to mitigate drift as the capacitive touch sensor is sensitive to outside factors the effect what the baseline none-touch value is. This function is controlled by the getGetAverageReading function. To reduce noise in the capacitive touch signal, a moving average filter is used to compute a stable baseline. Given N samples of raw sensor data x_i, the baseline or filtered reading is calculated as:

$$\text{AverageReading} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

This filter smooths transient fluctuations and adapts to environmental drift, forming the reference against which touch detection is evaluated.

Oscillation smoothing and debounce logic.

To avoid false touch toggling caused by minor signal jitter around the detection threshold, the algorithm uses an oscillation counter over a window of the W

most recent touch states $h_i \in \{0, 1\}$. The number of transitions (oscillations) is computed as:

$$\text{OscillationCount} = \sum_{i=1}^{W-1} \delta(h_i \neq h_{i-1})$$

Where $\delta(\cdot)$ is an indicator function:

$$\delta(\text{true}) = 1, \quad \delta(\text{false}) = 0$$

The final touch decision is made by comparing the oscillation count to a threshold T:

$$\text{TouchActive} = \begin{cases} 1, & \text{if OscillationCount} \geq T \\ 0, & \text{otherwise} \end{cases}$$

This debounce function ensures that only consistent or rapidly changing patterns are interpreted as valid touch events.

3.2.2 BLE Configuration

The BLE Characteristic (ie. the type of transmission functionality the device has) is set to read, write and notify. The peripheral device is set this way to have bidirectional feedback. The notify setting allows for constant data transmission removing the need for client side polling.

3.3 Client Side Software

3.3.1 BLE service and characteristic setup.

Similarly to the Arduino, the BLE configuration is set up to read and write, but inversely it subscribes to notifications, which allows the device to continually receive data. This functionality is controlled by `find_arduino()` and `run()`.

3.3.2 Touch post processing

For an additional layer of processing, a heuristic threshold function, mirroring the one on the Arduino side, is used to verify values and touch events. This functionality is handled by `handle_notification()`.

3.3.3 Real-time data Visualization

During an active session, the client can display a live plot of sensor data. This visualization includes a continuous line graph showing raw capacitive values, as well as red markers indicating detected touch events. By displaying this information in real time, the system verifies the responsiveness and accuracy of the touch detection system. It also serves as a debugging tool to observe patterns, false positives, or missed touch events as they happen. This is controlled by `update_plot()`.

3.3.4 Post Session Visualization and Aggregation of Logged Touch Data

Once a session is complete, the recorded data is visualized in two key forms: an analog plot and a digital classification plot. The analog plot shows the continuous sensor readings over time, with touch events visually marked. The digital plot simplifies the session into binary states (touched or not touched), for easier interpretation of the touch pattern. These visual outputs serve as documentation and support further analysis or annotation of the session data. This is generated using `plot_session_data()`.

3.3.5 Session states control logic Visualization

Keyboard session control logic manages the start, stop, and end of data recording. It relies on keyboard input and interrupts to initiate a new session, create a session

name, and terminate recording. During a session, the system actively records and processes incoming data. When the session ends, files are closed and data plots are generated. This logic ensures that each recording session is clearly managed saving the CSV and graph data. This is implemented in `keypress_monitor()`.

4 Discussion and Results

4.1 Results

The current prototype demonstrates a functional integration between the capacitive touch sensing hardware and the client-side software via BLE. The BLE communication is stable across typical operational distances and supports real-time data transmission from the Arduino Nano 33 BLE to a client device. Core components such as session initialization, user-controlled session termination, and real-time feedback are fully implemented.

The session control logic in the client allows users to start, name, and end recording sessions dynamically. This structure ensures that each dataset is clearly bounded and traceable, facilitating organized data collection. Once a session is concluded, the system automatically triggers the generation of analog and digital plots. These visual outputs not only help validate sensor performance, but also support post-session analysis and pattern recognition. The prototype covers key functional aspects, but some limitations remain, such as drift in the capacitive signal baseline. These areas point toward future enhancements involving adaptive filtering, error correction, and perhaps expansion to multi-channel sensing.

Overall, the system successfully establishes a bidirectional communication pipeline that enables ground truth data capture for wheelchair pushrim contact, supporting the broader goals of the WheelPoser project.

Sample session output.

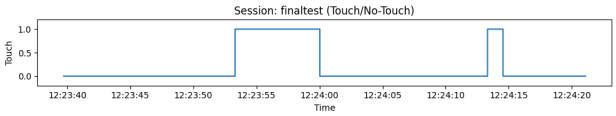

Figure 4: Digital Output

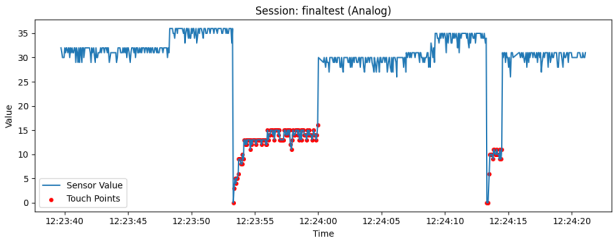

Figure 5: Analog Output

4.2 Challenges and Limitations

One of the primary challenges during prototype testing was the sensitivity of the capacitive touch sensor to environmental factors. These factors cause the baseline signal to drift, leading to false positives or missed touches. While the system uses a dynamic threshold and periodic recalibration to mitigate these effects, ambient conditions still introduce variability that impacts sensing reliability. In addition, the presence of nearby conductive materials or inconsistent user grounding can result in erratic readings, further complicating signal interpretation. These limitations show the need for more advanced filtering techniques, such as adaptive or machine-learned thresholds, and potential shielding or grounding improvements in hardware design.

5 Conclusion

Ultimately, the Touch Sense and Capture system creates a framework for a robust data transmission system for capacitive touch sensing in a small form. With additional research into guarding against drift, the touch capacitance sensor has the potential to enable high-quality data collection and non-invasive monitoring of pushrim interaction for wheelchair users. This data can contribute to more accurate pose models, giving wheelchair users more information and, as a result, greater agency over their health.

6 Appendix

6.1 Code Base

All of the code and hardware diagrams can be found on github @ `https://github.com/Tatyana-AC/Wheelposer_touch`

References

[1] Yunzhi Li et al. "WheelPoser: Sparse-IMU Based Body Pose Estimation for Wheelchair Users". In: *Proceedings of the 26th International ACM SIGACCESS Conference on Computers and Accessibility*. New York, NY, USA: Association for Computing Machinery, 2024. DOI: 10.1145/3663548.3675638. URL: https://doi.org/10.1145/3663548.3675638.

[2] Chendong Liu, Yilin Zhang, and Huanyu Zhou. "A Comprehensive Study of Bluetooth Low Energy". In: *Journal of Physics: Conference Series* 2093.1 (2021), p. 012021. DOI: 10.1088/1742-6596/2093/1/012021. URL: https://doi.org/10.1088/1742-6596/2093/1/012021.

[3] Robert Puers. "Capacitive Sensors: When and How to Use Them". In: *Sensors and Actuators A: Physical* 37–38 (1993), pp. 93–105. ISSN: 0924-4247. DOI: 10.1016/0924-4247(93)80019-D. URL: https://www.sciencedirect.com/science/article/pii/092442479380019D.