µTouch Artifact Guide

Siyuan Wang, Ke Li, Jingyuan Huang, Jike Wang, Cheng Zhang, Alanson Sample, Dongyao Chen
(Artifacts correspond to the PerCom'26 paper "µTouch: Enabling Accurate, Lightweight Self-Touch Sensing with Pass

I. Scope

This guide describes the artifact supporting µTouch: hardware (Magway PCB + magnets) and software (BLE data collection, semi-supervised classifier). It targets reviewers who want to install, run, and validate the pipeline.

II. Bill of Materials & Requirements

A. Hardware (minimal)

- Magway PCB (Altium project in pcb/; assembled board). PCB design by Xiaomeng Chen.
- 1–2 passive N52 grade magnets (6–8 mm recommended).
- Host laptop: Ubuntu 20.04+ or macOS 12+, 4-core CPU, ≥8 GB RAM, BLE 4.0+ adapter.
- Optional: BLE USB dongle (if desktop lacks BLE).

B. Software

- Python 3.10; Conda recommended.
- Git with submodules; CMake/Make (only if rebuilding C++ solver).
- Dependencies from pip install -e .[dev].
- Latex/PDF tools not required for runtime; only for this guide.

III. Obtaining the Artifact

1) Clone the repository (now public): git clone –recurse-submodules git@github.com:Wangmerlyn/muTouch.git (HTTPS alternative for CI: https://github.com/Wangmerlyn/muTouch.git)
2) Activate env: conda create -n muTouch python=3.10; conda activate muTouch
3) Install deps: pip install -e .[dev]; pre-commit install (optional for lint).
4) Models: a full model snapshot is tagged at backup/3_dim-models-20260121. Download from GitHub Releases or checkout tag if needed for offline use.

IV. Setup & Configuration

1) Flash firmware: open Codes/Arduino/bleReadMultiple/bleReadMultiple.ino in Arduino IDE, target Bluefruit nRF52 Feather, and upload.
2) Find BLE address: python Codes/read_raw_ble/find_device.py; copy device MAC/UUID.
3) Calibration: python Codes/read_raw_ble/read_sensor.py --addr <BLE_ADDR> --out calibration.npy; perform figure-8 motions away from metal.
4) Offsets/scales: place generated offset-*, scale-* in calibration_files/ (or update paths in the scripts).
5) Models: ensure Codes/read_raw_ble/models/ contains the downloaded checkpoint set if you need pretrained classifiers.

V. Running the Artifact

A. Data capture

python Codes/read_raw_ble/read_sensor_real.py --addr <BLE_ADDR>

Outputs timestamped CSVs under datasets/.

B. Real-time classification

python Codes/read_raw_ble/read_sensor_real_classifier.py --addr <BLE_ADDR>

Ensure the script points to the latest offset-*, scale-*, and model files. Console prints detected gesture labels; logs are saved under datasets/.

C. Expected outcomes

- Face-touching: ≈93% accuracy (8 gestures) with 3 s fine-tuning/user.
- Scratch detection: ≈95% accuracy across 12 participants.
- Real-time loop maintains >30 Hz inference on a laptop CPU.

VI. Reproducibility Checklist

- Hardware reproducible: PCB sources + BOM (Magway) included.
- Software reproducible: All scripts + TS2Vec submodule; pinned deps in Codes/requirements.txt.
- Data: Calibration and small demo runs can be generated locally; full datasets are participant-specific and not included.
- Pretrained models: Provided via GitHub tag backup/3_dim-models-20260121.

VII. Troubleshooting

- BLE not found: retry find_device.py; check power and pairing blocks; use BLE dongle.
- Drifting predictions: recalibrate sensors; ensure distance from large metal; re-run offset/scale.

- Import errors: confirm submodule init (git submodule update –init –recursive) and Python path from repo root.

VIII. Time Budget for Reviewers

- Setup environment: 10–15 minutes.
- Flash firmware + calibration: 15 minutes.
- Run live classification demo: 5 minutes.

IX. Notes on Prior Work

The project builds on MagX (MobiCom'21) codebase for magnetic sensing; source: https://github.com/dychen24/magx. This artifact extends it to self-touch sensing and includes updated PCB by Xiaomeng Chen.