

# μTouch Artifact Guide

Siyuan Wang\*, Ke Li<sup>†</sup>, Jingyuan Huang\*, Jike Wang\*, Cheng Zhang<sup>‡</sup>, Alanson Sample<sup>‡</sup>, Dongyao Chen\*

\*Shanghai Jiao Tong University, China

<sup>†</sup>Cornell University, USA

<sup>‡</sup>University of Michigan, USA

## Quick Links

Code repo [github.com/Wangmerlyn/muTouch](https://github.com/Wangmerlyn/muTouch)  
MagX base [github.com/dychen24/magx](https://github.com/dychen24/magx)  
Models Tag backup/3\_dim-models-20260121  
(GitHub Releases → Assets)  
PCB sources pcb/ (muTouch Altium project;  
legacy Magway.\* filenames)

## I. Scope

This guide describes the artifact supporting μTouch: hardware (muTouch 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)

- muTouch PCB (Altium project in pcb/; assembled board; filenames use legacy Magway.\*). 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 fallback:
# https://github.com/Wangmerlyn/muTouch.git
```

- 2) Activate env:

```
conda create -n muTouch python=3.10
conda activate muTouch
```

- 3) Install deps:

```
pip install -r Codes/requirements.txt
```

- 4) Models: snapshot tag backup/3\_dim-models-20260121. Download binaries from GitHub Releases (Assets).
- 5) Set working directory for runtime scripts to Codes/:

```
cd Codes
```

## IV. Setup & Configuration

- 1) Flash firmware:  
Codes/Arduino/bleReadMultiple/bleReadMultiple.ino in Arduino IDE;  
select Feather nRF52; upload.
- 2) Find BLE address: `python read_raw_ble/find_device.py` (copy device MAC/UUID).
- 3) Hardcode BLE address: edit the `address = "..."` line near the bottom of `read_raw_ble/read_sensor.py`, `read_raw_ble/read_sensor_real.py`, and (if used) `read_raw_ble/read_sensor_real_classifier.py`.
- 4) Calibration capture (run inside Codes/):

```
python read_raw_ble/read_sensor.py
```

Do a brief figure-8 motion away from metal surfaces; CSVs are saved under `datasets/`.

- 5) Offsets/scales: place the latest files named `offset-*.npy` and `scale-*.npy` in `calibration_files/`. The scripts automatically load the newest files with those prefixes.
- 6) Models: ensure `read_raw_ble/models/` holds the downloaded checkpoint set if you need pretrained classifiers.

## V. Running the Artifact

### A. Data capture

```
python read_raw_ble/read_sensor_real.py
```

Outputs timestamped CSVs under `datasets/`.

### B. Real-time classification

```
python read_raw_ble/read_sensor_real_classifier.py
```

Ensure the script uses the latest `offset-*`, `scale-*`, and model files.

Console prints detected gesture labels; logs are saved under `datasets/`.

### C. Expected outcomes

- Face-touching:  $\approx 93\%$  accuracy (8 gestures) with 3 s fine-tuning/user.
- Scratch detection:  $\approx 95\%$  accuracy (9 gestures) with 3 s fine-tuning/user.
- Real-time loop maintains  $>30$  Hz inference on a laptop CPU.

### VI. Reproducibility Checklist

- Hardware reproducible: PCB sources + BOM (muTouch; files named Magway.\* for compatibility) 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. For Complete Instructions

Due to page limits, this PDF is a concise checklist. For full, continuously updated steps and tips, please refer to the repository README.md and the Markdown guide at artifact/ARTIFACT\_GUIDE.md. Use those as the authoritative references for reproduction and further code tweaks.

### IX. Notes on Prior Work

The project builds on MagX (MobiCom'21) codebase for magnetic sensing [1]. This artifact extends it to self-touch sensing and includes updated PCB by Xiaomeng Chen.

### References

- [1] D. Chen, M. Wang, C. He, Q. Luo, Y. Irvantchi, A. Sample, K. G. Shin, and X. Wang, "Magx: Wearable, untethered hands tracking with passive magnets," in Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (MobiCom '21), 2021, pp. 269–282.