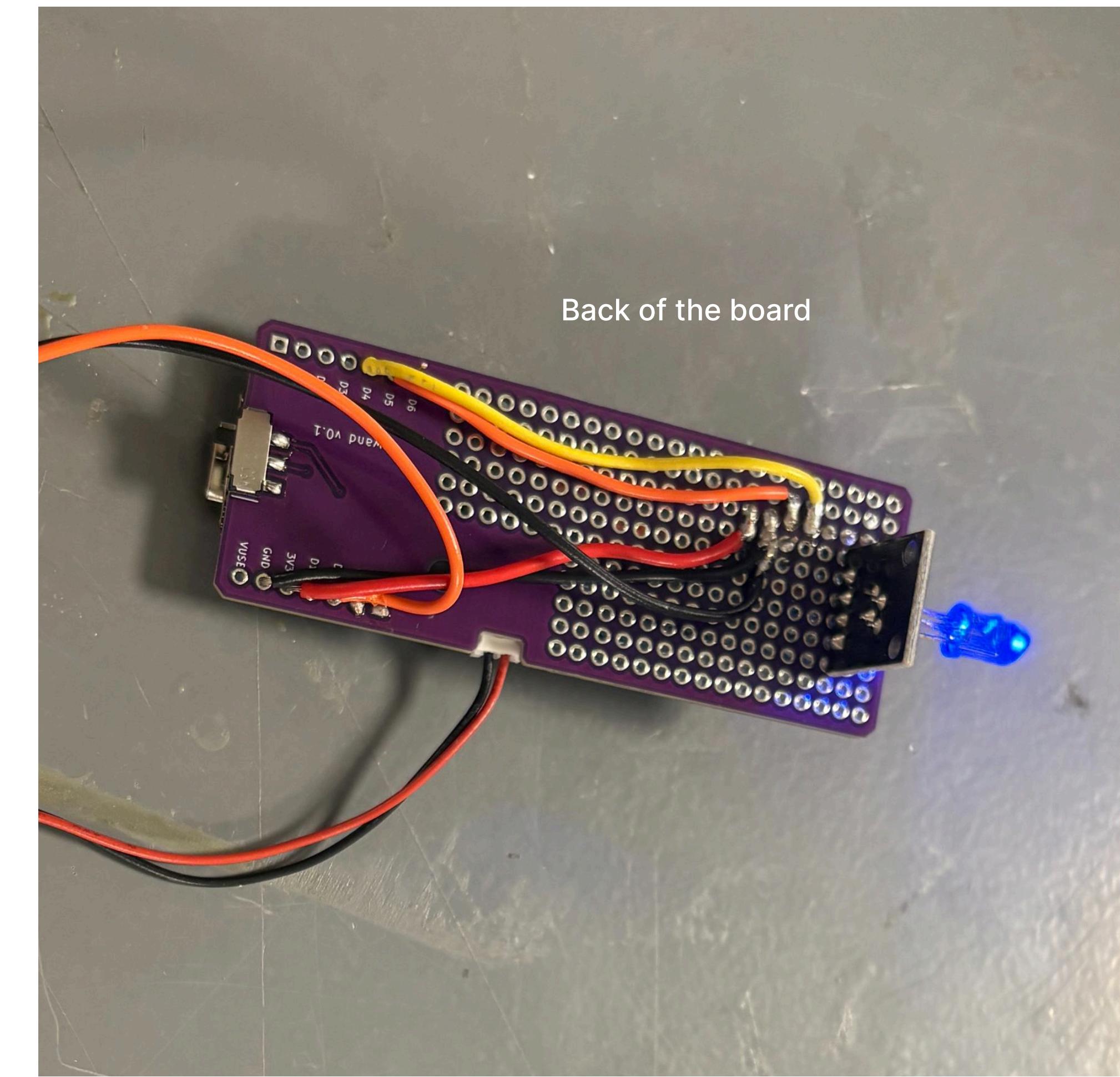
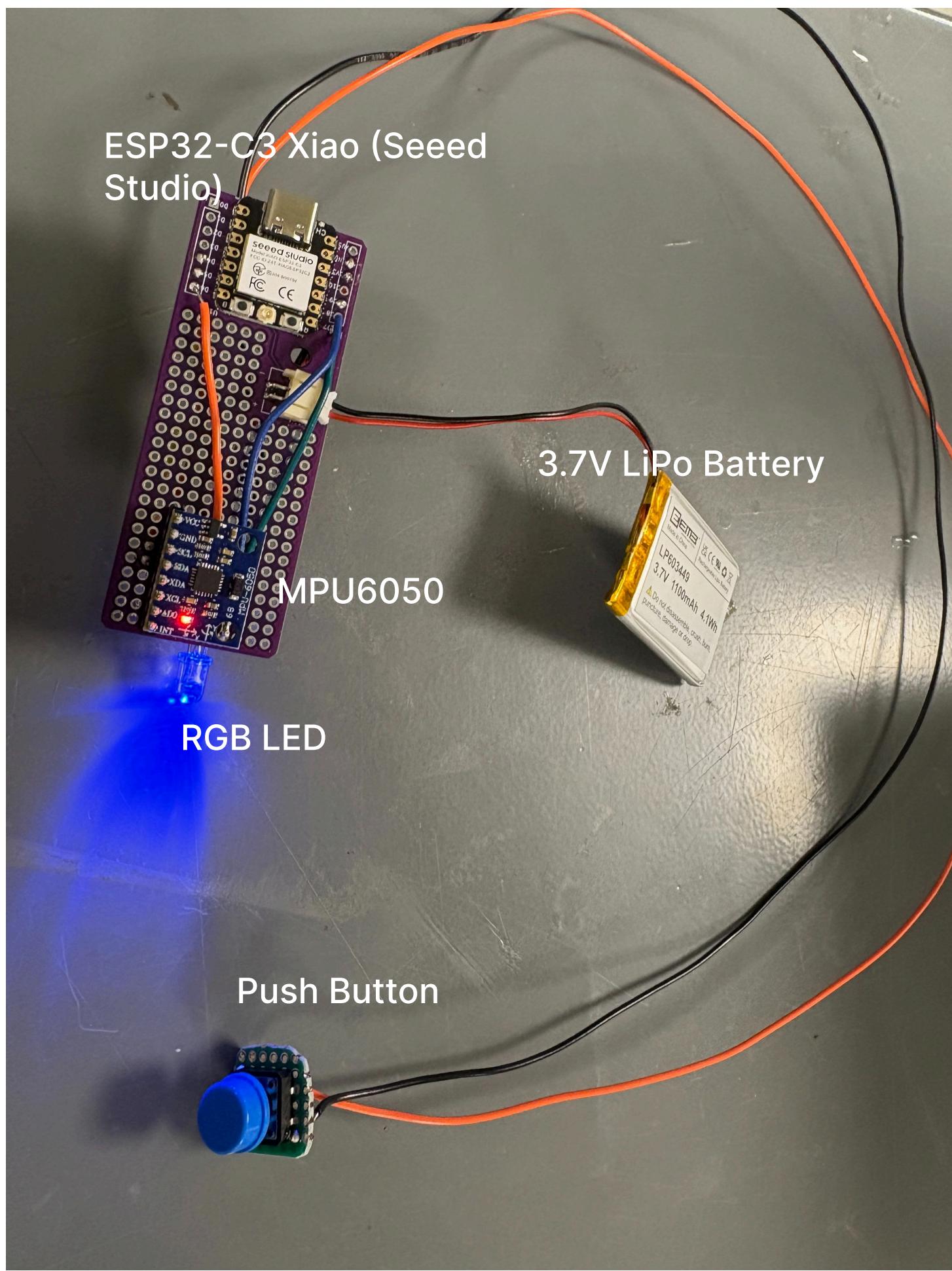


*TECHIN 515A*

# TECHIN515 Lab 4 - Magic Wand

Auria Zhang  
May 19, 2025

# Pictures of hardware setup and connections



# Data collection process and results

```
Capture started...
Saved 101 samples to data/0/output_0_ko_97_20250519_020824.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_98_20250519_020826.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_99_20250519_020828.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_100_20250519_020830.csv
^C
Exiting...
Serial connection closed
(.venv) (base) linjunzhang@Linjuns-MacBook-Pro-2 gesture_capture % python process_gesture_data.py --gesture "Z" --person "ko" --port /dev/cu.usbmodem101

Created directory: data/V
Connecting to ESP32 on /dev/cu.usbmodem101 at 115200 baud...
Connected! Waiting for gesture data...
Press Ctrl+C to exit
Send 'o' to start capture (will automatically stop after 1 second)
Capture started...
Saved 101 samples to data/V/output_V_ko_1_20250519_021028.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_2_20250519_021031.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_3_20250519_021033.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_4_20250519_021035.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_5_20250519_021037.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_6_20250519_021039.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_7_20250519_021041.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_8_20250519_021043.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_9_20250519_021044.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_10_20250519_021046.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_11_20250519_021048.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_12_20250519_021049.csv
Capture started...
Saved 101 samples to data/V/output_V_ko_13_20250519_021051.csv
```

```
(.venv) (base) linjunzhang@Linjuns-MacBook-Pro-2 gesture_capture % python process_gesture_data.py --gesture "Z" --person "ko" --port /dev/cu.usbmodem101

Connecting to ESP32 on /dev/cu.usbmodem101 at 115200 baud...
Connected! Waiting for gesture data...
Press Ctrl+C to exit
Send 'o' to start capture (will automatically stop after 1 second)
Capture started...
Saved 101 samples to data/Z/output_Z_ko_1_20250519_014919.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_2_20250519_014932.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_3_20250519_014936.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_4_20250519_014939.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_5_20250519_014942.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_6_20250519_014948.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_7_20250519_014950.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_8_20250519_014953.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_9_20250519_014955.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_10_20250519_014958.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_11_20250519_015000.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_12_20250519_015003.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_13_20250519_015005.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_14_20250519_015007.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_15_20250519_015010.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_16_20250519_015012.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_17_20250519_015014.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_18_20250519_015017.csv
Capture started...
Saved 101 samples to data/Z/output_Z_ko_19_20250519_015019.csv
```

```
Created directory: data/0
Connecting to ESP32 on /dev/cu.usbmodem101 at 115200 baud...
Connected! Waiting for gesture data...
Press Ctrl+C to exit
Send 'o' to start capture (will automatically stop after 1 second)
Capture started...
Saved 101 samples to data/0/output_0_ko_1_20250519_020522.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_2_20250519_020527.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_3_20250519_020529.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_4_20250519_020531.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_5_20250519_020533.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_6_20250519_020535.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_7_20250519_020537.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_8_20250519_020539.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_9_20250519_020541.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_10_20250519_020543.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_11_20250519_020544.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_12_20250519_020546.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_13_20250519_020549.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_14_20250519_020551.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_15_20250519_020553.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_16_20250519_020555.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_17_20250519_020556.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_18_20250519_020558.csv
Capture started...
Saved 101 samples to data/0/output_0_ko_19_20250519_020600.csv
Capture started...
```

# Data collection process and results

The screenshot displays a user interface for managing a dataset. At the top, there are two summary cards: "DATA COLLECTED" showing "5m 7s" with a pie chart icon, and "TRAIN / TEST SPLIT" showing "80% / 20%" with a pie chart icon. Below these is a main section titled "Dataset" containing a table of samples. The table has columns for "SAMPLE NAME", "LABEL", "ADDED", and "LENGTH". The "LABEL" column shows "O" for all samples. The "ADDED" column shows "Today, 02:16:53" and the "LENGTH" column shows "1s" for all samples. A dark overlay box covers the right side of the table, containing the text "RAW DATA" and "Click on a sample to load...". On the far right, there is a circular button with a question mark.

SAMPLE NAME	LABEL	ADDED	LENGTH
output_O_ko_73_2025...	O	Today, 02:16:53	1s
output_O_ko_94_2025...	O	Today, 02:16:53	1s
output_O_ko_44_2025...	O	Today, 02:16:53	1s
output_O_ko_47_2025...	O	Today, 02:16:53	1s
output_O_ko_3_20250...	O	Today, 02:16:53	1s
output_O_ko_41_2025...	O	Today, 02:16:53	1s
output_O_ko_42_2025...	O	Today, 02:16:53	1s
output_O_ko_46_2025...	O	Today, 02:16:53	1s
output_O_ko_89_2025...	O	Today, 02:16:53	1s
output_O_ko_68_2025...	O	Today, 02:16:53	1s
output_O_ko_07_2025...	O	Today, 02:16:53	1s

# Edge Impulse Model Architecture and Optimization

The screenshot shows the Edge Impulse configuration interface for a Spectral Analysis block. On the left, the 'Raw features' section displays a list of numerical values: 1.0127, -1.6257, 9.9240, 0.9553, -1.5658, 10.0700, 0.8... Below it, the 'Parameters' section includes:

- Filter**: Scale axes (1), Input decimation ratio (1), Type (none).
- Analysis**: Type (FFT), FFT length (16), Take log of spectrum? (checked), Overlap FFT frames? (checked), Improve low frequency resolution? (unchecked).

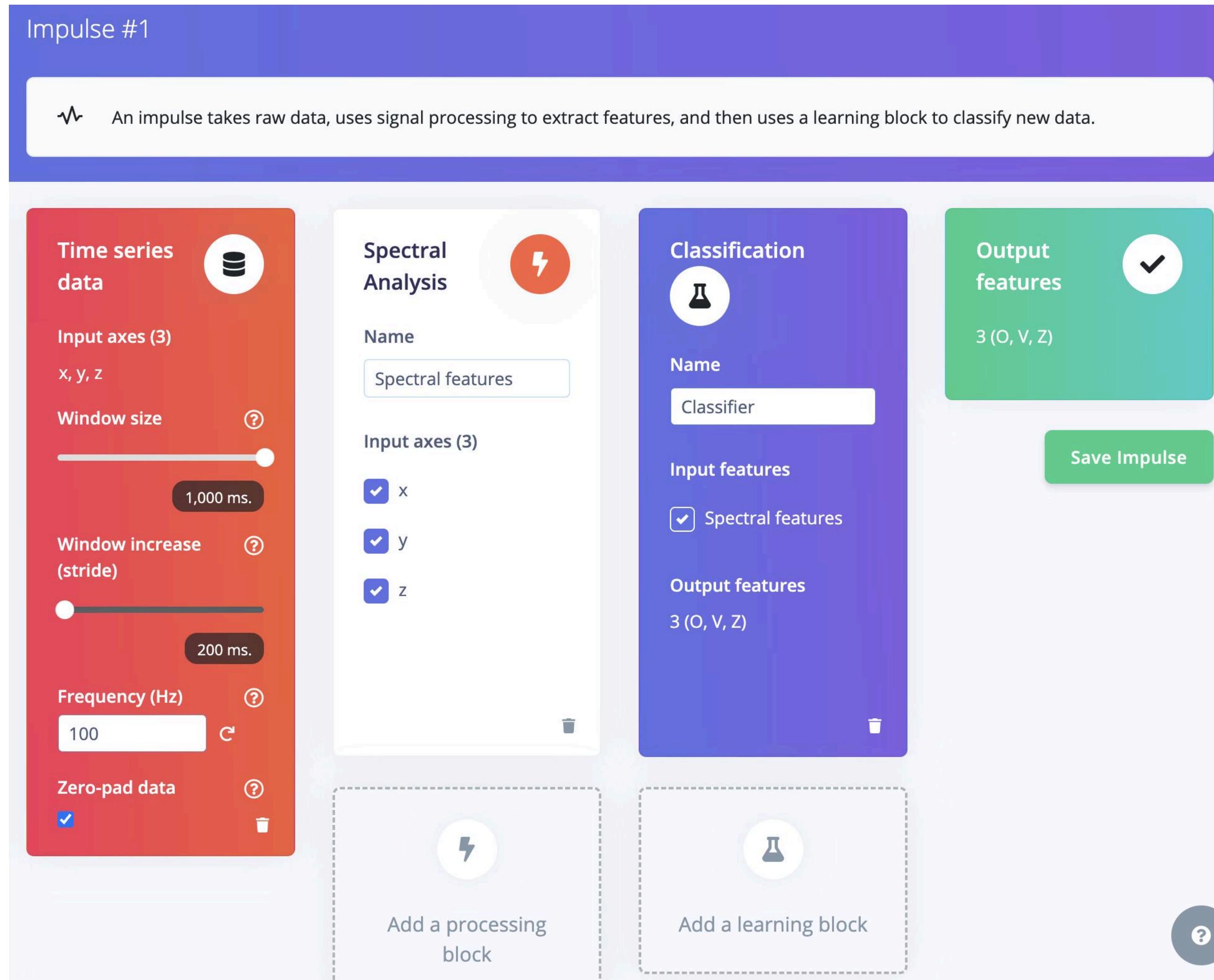
A blue 'Autotune parameters' button is located above the analysis section. At the bottom is a 'Save parameters' button.

On the right, the 'DSP result' section shows two plots: 'After filter' and 'Spectral power (log)'. The 'After filter' plot shows three time-domain signals (red, blue, green) over 1000 samples, with a prominent peak around sample 660. The 'Spectral power (log)' plot shows the energy distribution across frequencies from 0.00 to 50.00 Hz, with red and blue curves showing higher energy than green at lower frequencies.

The 'Processed features' section at the bottom lists the resulting numerical features: 11.0878, 0.7248, 0.6614, 2.3573, 3.7628, 2.9165, 2.5097, 2.0438, 1.8022, ...

I used the Spectral Analysis block in Edge Impulse to convert raw acceleration data into frequency-domain features. This is particularly effective for gesture recognition, as different hand movements produce distinct signal energy distributions across frequencies.

# Edge Impulse Model Architecture and Optimization



I designed and optimized the model in Edge Impulse using the following configuration:

- Input: Time-series data from MPU6050 (x, y, z) at 100Hz
- Window size: 1000ms (1 second gesture capture)
- Window increase: 200ms (stride)
- DSP block: Spectral Features (for frequency-domain analysis)
- Classifier: 2-layer dense neural network
- Output classes: 3 gesture labels — Z, O, and V

This structure effectively captured distinguishing features across the three gestures while maintaining low computational cost suitable for on-device inference.

# Performance Analysis and Metrics

The screenshot shows the Edge Impulse web interface for model development. On the left, a sidebar lists navigation options: Dashboard, Devices, Data acquisition, Experiments, EON Tuner, Impulse design (with sub-options Create impulse, Spectral features, Classifier), Retrain model, Live classification, Model testing, and an Upgrade Plan section. The main area displays a neural network architecture with an input layer (39 features), two dense layers (20 and 10 neurons), and an output layer (3 classes). A button to "Save & train" is present. To the right, the "Last training performance" section shows accuracy at 100.0% and loss at 0.16. A confusion matrix table indicates perfect separation between gesture classes O, V, and Z. Below this, a "Metrics" table lists Area under ROC Curve, Weighted average Precision, Weighted average Recall, and Weighted average F1 score all at 1.00. At the bottom, a "Data explorer" section provides a scatter plot of training data points, color-coded by correctness (O - correct, V - correct, Z - correct, O - incorrect, Z - incorrect).

	O	V	Z
O	100%	0%	0%
V	0%	100%	0%
Z	0%	0%	100%
F1 SCORE	1.00	1.00	1.00

METRIC	VALUE
Area under ROC Curve	1.00
Weighted average Precision	1.00
Weighted average Recall	1.00
Weighted average F1 score	1.00

As shown in the confusion matrix below, the classifier correctly identified every gesture in the validation set. There were no misclassifications, and each label (Z, O, V) was predicted with perfect precision.

The Data Explorer visualization further confirms the separation between the three gesture classes. All samples clustered cleanly by class, with no significant overlap between gesture types. This indicates that the model architecture — a 2-layer dense neural network with spectral features — was effective for this task. The performance is strong enough for real-time embedded deployment on the ESP32-C3 device.

## Answers to Design Questions and Justifications

This system was designed for standalone use, so a physical button was used to trigger gesture capture instead of relying on serial input. A 1000ms window was selected to match the natural length of a hand gesture. Spectral Features were used as the DSP block to extract frequency-domain patterns, which are effective for distinguishing motion types.

The model architecture includes two dense layers (20 and 10 neurons), offering a balance between performance and deployment size. Quantization to int8 was applied to reduce memory usage and support ESP32 inference. During data collection, RGB LED feedback was added to confirm the current label, reducing the risk of mislabeling.

All decisions were made to ensure smooth real-time performance, minimize user error, and maintain compatibility with the ESP32-C3's limited resources.

# Discussion

## Discussion: Effect of Window Size

The window size determines how many time steps are included in each training sample. A larger window (e.g., 1000ms) generates fewer but more complete samples, which is helpful for capturing the full span of slower or complex gestures. However, it also increases the input vector size, which results in a larger number of neurons in the input layer and higher computational cost.

In this project, a 1000ms window with 100Hz sampling produced 300 features ( $x/y/z \times 100$ ). This size was chosen to balance coverage and model size, ensuring that even slow-changing patterns like circular "O" gestures are fully captured without truncation.

## Discussion: Strategies to Improve Model Performance

1. Data Augmentation: Introduce slight noise, temporal shifts, or mirrored gestures to increase variability and reduce overfitting, especially since the dataset was collected from a single user.
2. Additional Sensor Channels: Incorporate gyroscope data from the MPU6050 in addition to accelerometer input. This would provide angular velocity information, improving gesture separability for similar movement paths.
3. (Optional) Ensemble or Hybrid Models: Combine a lightweight CNN with dense layers to extract both temporal and spatial patterns more effectively (though this may exceed MCU limits).

## Demo Video Link

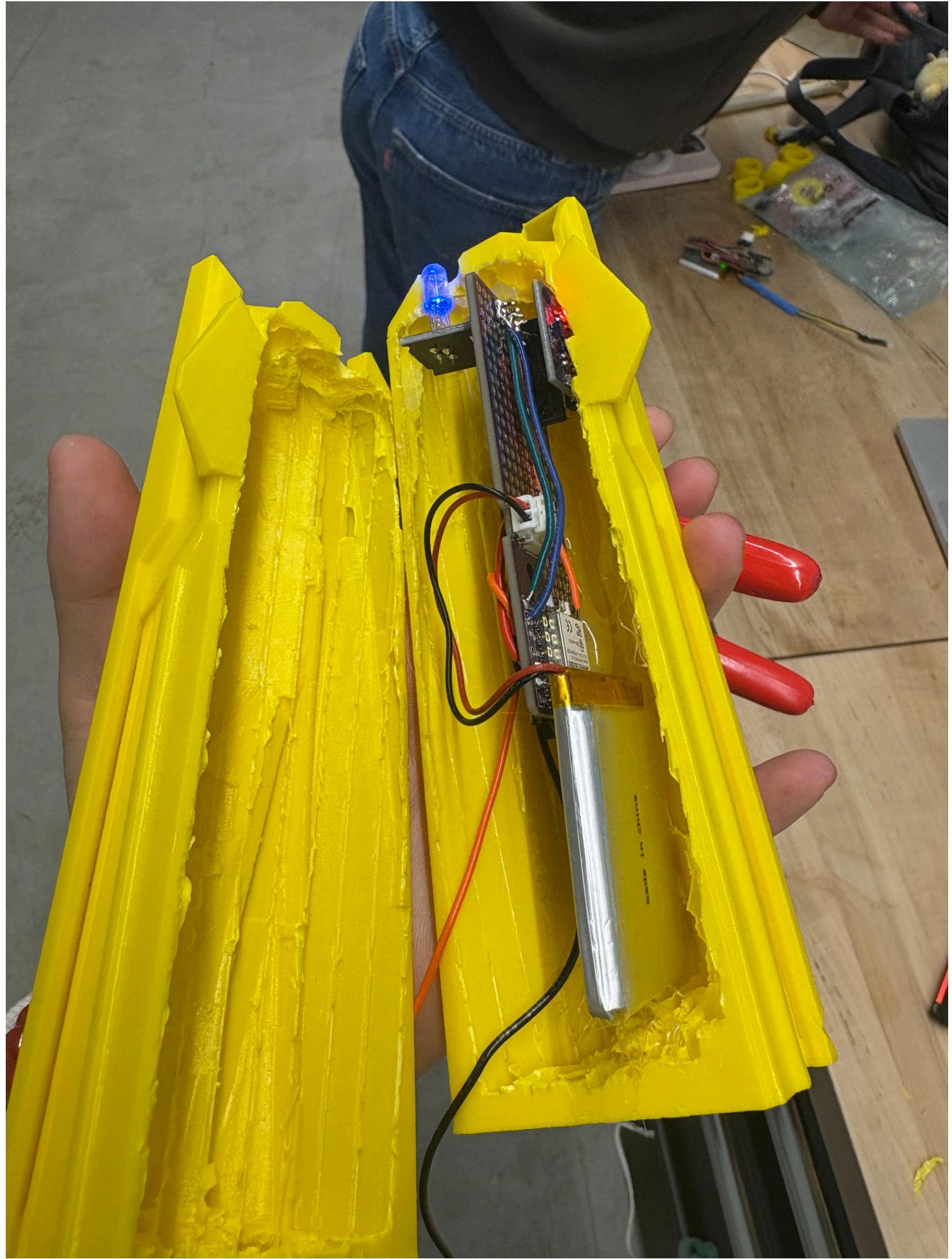
<https://youtube.com/shorts/fXiE956W-jA?feature=share>

## Challenges Faced and Solutions

During development, one key challenge was serial port conflict between the Arduino IDE and the Python data collection script. Since both tools attempted to access the same USB port, switching between them often caused connection failures. This was resolved by explicitly closing the Serial Monitor before launching the Python script, and ensuring only one tool accessed the port at a time.

Another issue involved incorrect gesture labeling during data collection. Forgetting to update the gesture class before pressing the capture button led to mislabeled samples. To address this, we added RGB LED color feedback to indicate the active label in real time, reducing user error and improving dataset quality.

# Enclosure and Battery



## Enclosure and Battery

