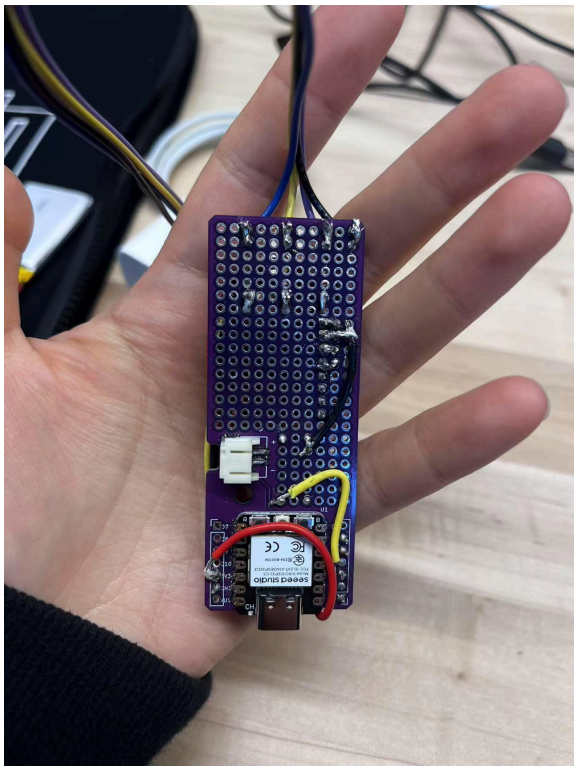
TECHIN 515
Prof. John Raiti, Prof. Luyao Niu
Adele Wang
19.05.2025

# Magic Wand Gesture Recognition System Report

- Pictures of hardware setup and connections




The hardware setup includes the following components:

- **ESP32 Development Board**: The central microcontroller used for processing data and executing the gesture recognition algorithm
- **MPU6050 IMU Sensor**: The sensor used to capture acceleration and gyroscopic data for detecting gestures
- **LED**: For visual feedback of gestures (e.g., lighting up when a gesture is recognized).
- **Battery**: For powering the system in a portable manner
- **Enclosure**: A protective case to house the components

The **MPU6050** sensor is connected to the **ESP32** as follows:

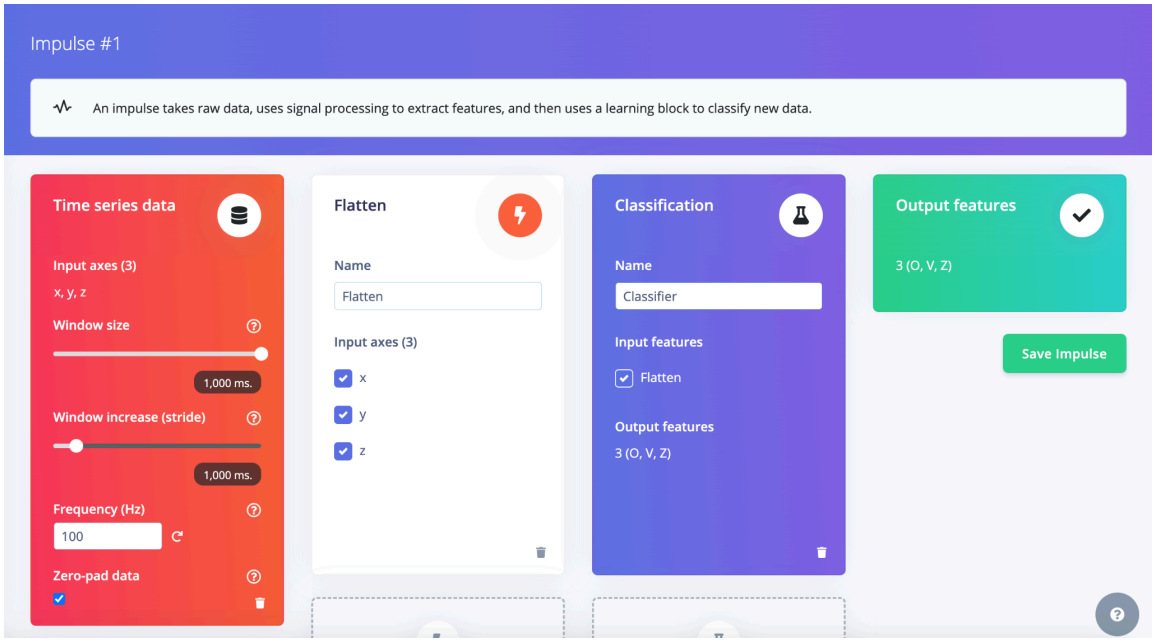
- **VCC** → 3.3V
- **GND** → GND
- **SDA** → GPIO4
- **SCL** → GPIO5
- **LED(R)** → GPIO0
- **LED(G)** → GPIO1
- **LED(B)** → GPIO2

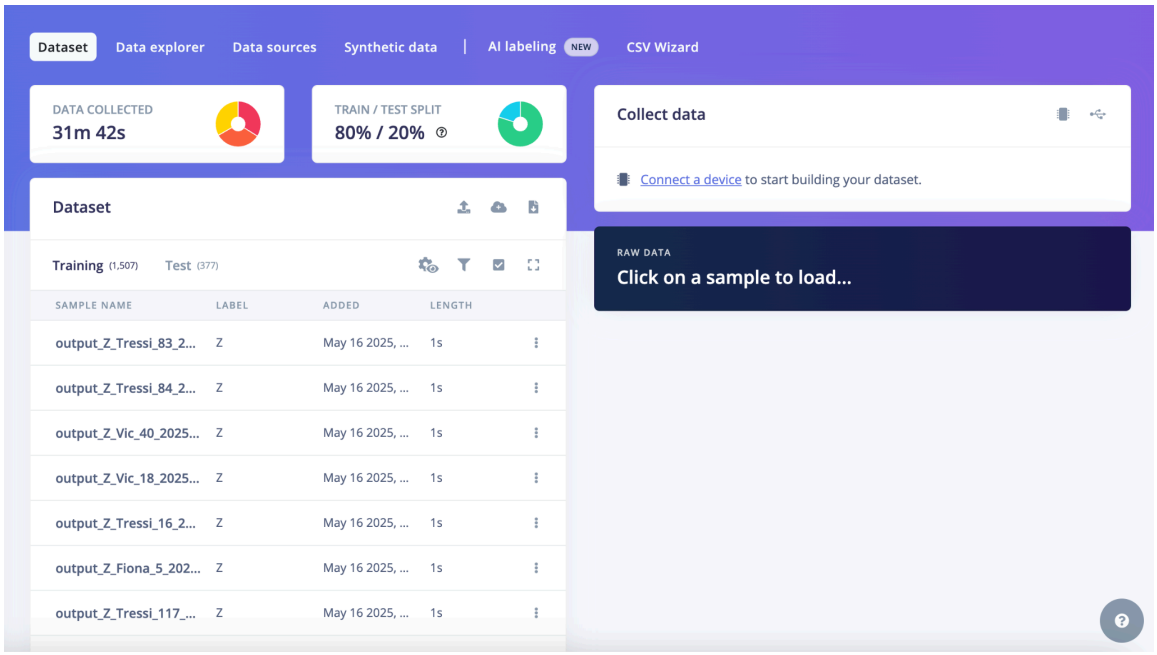
- Data collection process and results
  https://github.com/AdeleWang-47/TECHIN515-magic-wand/tree/main/src/dataset
  The data was collected using the gesture_capture.ino Arduino sketch and processed with the process_gesture_data.py Python script. Each gesture was performed at least 100 times to ensure data variability. The data includes four columns:
  Timestamp: The time when the sample was recorded
  x: X-axis acceleration (m/s²)
  y: Y-axis acceleration (m/s²)
  z: Z-axis acceleration (m/s²)

515-lab4-gesture-capture.ino

```ino
// Basic demo for accelerometer readings from Adafruit MPU6050

#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;
long last_sample_millis = 0;
bool capture = false;
char a;
unsigned long capture_start_time = 0;
const unsigned long CAPTURE_DURATION = 1000; // 1 second in milliseconds

void setup(void) {
  Serial.begin(115200);
  while (!Serial) {
    delay(10); // will pause Zero, Leonardo, etc until serial console opens
  }

  // Try to initialize!
  while (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    delay(10);
    //while (1) {
      //delay(10);
    //}
  }

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
  Serial.println("");
  delay(100);
}

void capture_data() {
  /* Get new sensor events with the readings */
  if ((millis() - last_sample_millis) >= 10) { // 10ms for 100Hz sampling rate

    last_sample_millis = millis();
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);
```

- Edge Impulse model architecture and optimization
  The model developed for gesture recognition used a neural network architecture to classify the gestures based on the accelerometer data. The Impulse Design in Edge Impulse included the following key steps:
  Feature Extraction (DSP Block): We used a Time-Series Processing block with a window size of 1000 ms and a stride of 1000 ms to extract features from the accelerometer data.
  Machine Learning Block: The model was trained using a neural network classifier with a fully connected layer architecture. We tuned the hyperparameters, including learning rate and number of epochs, to optimize the model's performance
  Using quantized (Int8) model for deployment, which reduces the model size and makes it more efficient for embedded systems


Impulse #1
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
Time series data
Input axes (3)
x, y, z
Window size
1,000 ms.
Window increase (stride)
1,000 ms.
Frequency (Hz)
100
Zero-pad data
Flatten
Name
Flatten
Input axes (3)
x
y
z
Classification
Name
Classifier
Input features
Flatten
Output features
3 (O, V, Z)
Output features
3 (O, V, Z)
Save Impulse


Dataset
Data explorer
Data sources
Synthetic data
AI labeling
NEW
CSV Wizard
DATA COLLECTED
31m 42s
TRAIN / TEST SPLIT
80% / 20%
Collect data
Connect a device to start building your dataset.
Dataset
Training (1,507)
Test (377)
SAMPLE NAME
LABEL
ADDED
LENGTH
output_Z_Tressi_83_2...
output_Z_Tressi_84_2...
output_Z_Vic_40_2025...
output_Z_Vic_18_2025...
output_Z_Tressi_16_2...
output_Z_Fiona_5_202...
output_Z_Tressi_117_...
Z
May 16 2025, ...
1s
RAW DATA
Click on a sample to load...

MODEL OPTIMIZATIONS

Model optimizations can increase on-device performance but may reduce accuracy.

**EON™ Compiler**

Same accuracy, 48% less RAM, 18% less ROM.

**Quantized (int8)**

Selected ✓

| | FLATTEN | CLASSIFIER | TOTAL |
|---|---|---|---|
| LATENCY | - | 3 ms. | **3 ms.** |
| RAM | 1.3K | 1.9K | **1.9K** |
| FLASH | - | 87.0K | **-** |
| ACCURACY | | | **88.33%** |

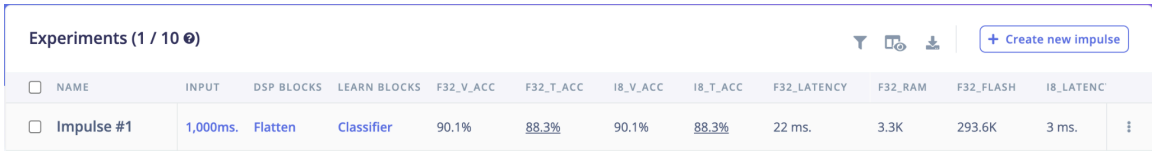
**Unoptimized (float32)**

Select

| | FLATTEN | CLASSIFIER | TOTAL |
|---|---|---|---|
| LATENCY | - | 22 ms. | **22 ms.** |
| RAM | 1.3K | 3.3K | **3.3K** |
| FLASH | - | 293.6K | **-** |
| ACCURACY | | | **88.33%** |

Estimate for Espressif ESP-EYE (ESP32 240MHz) - Change target

- Performance analysis and metrics



Experiments (1 / 10) + Create new impulse

| NAME | INPUT | DSP BLOCKS | LEARN BLOCKS | F32_V_ACC | F32_T_ACC | I8_V_ACC | I8_T_ACC | F32_LATENCY | F32_RAM | F32_FLASH | I8_LATENC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Impulse #1 | 1,000ms. | Flatten | Classifier | 90.1% | 88.3% | 90.1% | 88.3% | 22 ms. | 3.3K | 293.6K | 3 ms. |

- Answers to questions and your choices to all design options with justifications
  - Why should you use training data collected by multiple students rather than using your own collected data only? Think about the effectiveness and reliability of your wand

    Using data from multiple students improves generalization, ensuring the model works well across different users with varying hand sizes, speeds, and motion styles. This makes the wand more reliable and effective for a broader audience
  - Discuss the effect of window size. Consider
    - the number of samples generated

Larger windows capture more data points, increasing dataset size; smaller windows generate fewer data points
    - the number of neurons in your input layer of neural network
      Larger windows lead to more input features, requiring more neurons in the input layer
    - effectiveness when capturing slow-changing patterns
      Larger windows are better for capturing slow gestures, while smaller windows may miss subtle changes
- Give at least two potential strategies to further enhance your model performance
  Data Augmentation: Techniques like rotation, scaling, and noise injection can create more diverse training data, improving generalization
  Hyperparameter Tuning: Fine-tuning the learning rate, number of epochs, and neural network architecture can optimize model performance for better accuracy and efficiency

- Demo video link
  https://github.com/AdeleWang-47/TECHIN515-magic-wand/tree/main/media
- Challenges faced and solutions
  Data Quality and Gesture Variability
  Solution: Ensuring consistent data collection was difficult due to the natural variability of gestures. To mitigate this, we collected a large number of samples (at least 20 per gesture) and ensured gestures were performed with consistent motion
  Model Optimization for ESP32
  Solution: The initial model was too large to fit within the ESP32's memory. By quantizing the model and tuning the neural network architecture, we were able to reduce the model size while maintaining high performance