CEC Continuing Education Center

DesignNews

Getting Started in TinyML with Arduino

DAY 3: The Magic Wand Gesture Recognition

Sponsored by

DigiKey

informa markets

Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

Dr. Don Wilcher

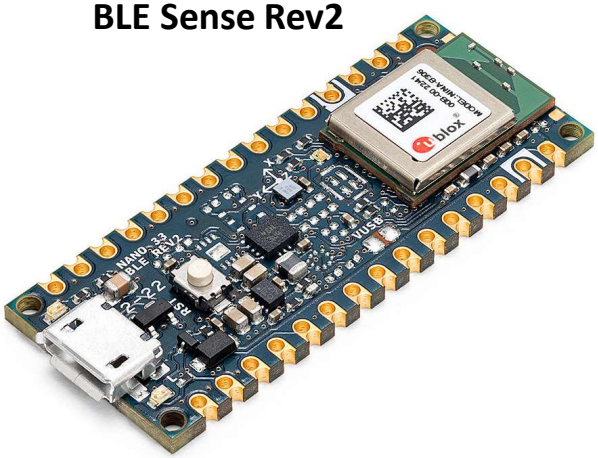
Visit ‘Lecturer Profile’ in your console for more details.

LinkedIn Page:
https://www.linkedin.com/in/dr-don-wilcher-ed-d-mseit-ee-ceta-2735151/

Patreon Page:
https://www.patreon.com/c/DrDon683

Course Kit and Materials

Arduino Nano 33 BLE Sense Rev2



Research Literature and Documentation

Tiny Machine Learning: Progress and Futures

Ji Lin Ligeng Zhu Wei-Ming Chen Wei-Chen Wang Song Han

Massachusetts Institute of Technology

https://tinyml.mit.edu

Abstract—Tiny Machine Learning (TinyML) is a new frontier of machine learning. By squeezing deep learning models into billions of IoT devices and microcontrollers (MCUs), we expand the scope of AI applications and enable ubiquitous intelligence. However, TinyML is challenging due to hardware constraints: the tiny memory resource makes it difficult to hold deep learning models designed for cloud and mobile platforms. There is also limited compiler and inference engine support for bare-metal devices. Therefore, we need to co-design the algorithm and system stack to enable TinyML. In this review, we will first discuss the definition, challenges, and applications of TinyML. We then survey the recent progress in TinyML and deep learning on MCUs. Next, we will introduce MCUNet, showing how we can achieve ImageNet-scale AI applications on IoT devices with *system-algorithm co-design*. We will further *extend* the solution from *inference to training* and introduce tiny on-device training techniques. Finally, we present future directions in this area. Today's "large" model might be tomorrow's "tiny" model. The scope of TinyML should evolve and adapt over time.

Index Terms—TinyML, Efficient Deep Learning, On-Device Training, Learning on the Edge

I. Overview of Tiny Machine Learning

Machine learning (ML) has made significant impacts on various fields, including vision, language, and audio. However, state-of-the-art models often come at the cost of high computation and memory, making them expensive to deploy. To address this, researchers have been working on efficient algorithms, systems, and hardware to reduce the cost of machine learning models in various deployment scenarios. There are two main subdomains of efficient ML: EdgeML and CloudML (Figure 1). While CloudML focuses on improving latency and throughput on cloud servers, EdgeML focuses on improving energy efficiency, latency, and privacy on edge devices. These two domains also intersect in areas such as hybrid inference [1, 2], over-the-air (OTA) updates, and federated learning between the edge and cloud [3]. In recent years, there has been significant progress in extending the scope of EdgeML to ultra-low-power devices such as IoT devices and microcontrollers, known as **TinyML**.

TinyML has several key advantages. It enables machine learning using only a few hundred kilobytes of memory which greatly reduces the cost. With billions of IoT devices producing more and more data in our daily lives, there is a growing need for low-power, always-on, on-device AI. By performing on-device inference near the sensor, TinyML enables better responsiveness and privacy while reducing the energy cost associated with wireless communication. On-device processing of data can be beneficial for applications where real-time decision-making is crucial, such as autonomous vehicles.

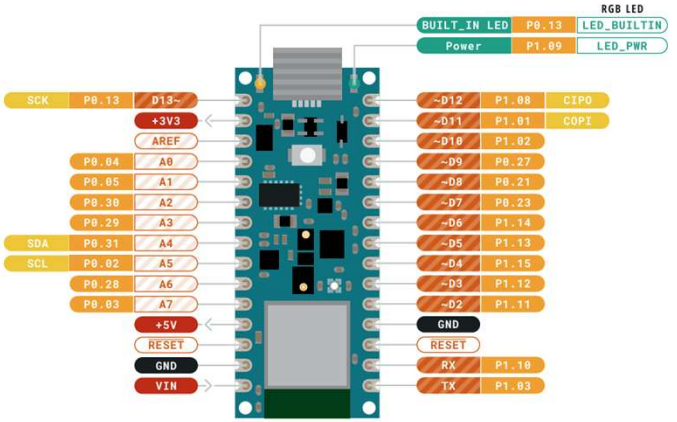
Fig. 1. Efficiency is critical for CloudML, EdgeML, and TinyML. CloudML targets high-throughput accelerators like GPUs, while EdgeML focuses on portable devices like mobile phones. TinyML further pushes the efficiency boundary, enabling powerful ML models to run on ultra-low-power devices such as microcontrollers.

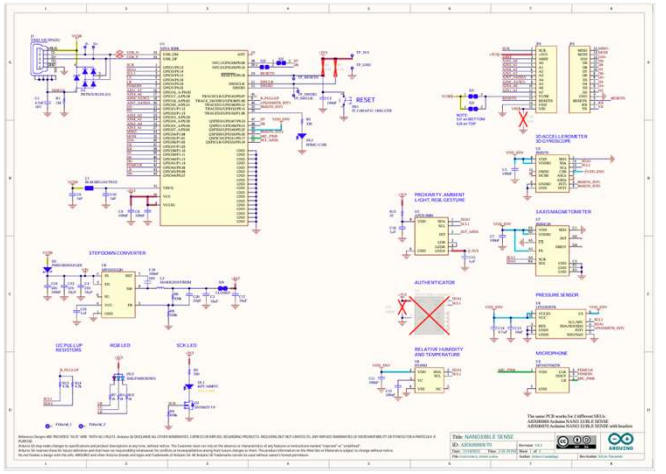
In addition to inference, we push the frontier of TinyML to enable on-device training on IoT devices. Irrevolutionizes EdgeAI through continuous and lifelong learning. Edge device can finetune the model on itself rather than transmitting data to cloud servers, which protects privacy. On-device learning has numerous benefits and a variety of applications. For example, home cameras can continuously recognize new faces, and email clients can gradually improve their prediction by updating customized language models. It also enables IoT applications that do not have a physical connection to the internet to adapt to the environment, such as precision agriculture and ocean sensing.

In this review, we will first discuss the definition and challenges of TinyML, analyzing why we can't directly scale mobile ML or cloud ML models for tinyML. Then we delve into the importance of system-algorithm co-design in TinyML. We will then survey recent literature and the progress of the field, presenting a holistic survey and comparison in Tables 1 and 10. Next, we will introduce our TinyML project, MCUNet, which combines efficient system and algorithm design to enable TinyML for both inference to training. Finally, we will discuss several emerging topics for future research directions in the field.

A. Challenges of TinyML

The success of deep learning models often comes at the cost of high computation, which is not feasible for use in TinyML applications due to the strict resource constraints of devices such as microcontrollers. Deploying and training AI models on MCU is extremely hard: No DRAM, no operating systems (OS), and strict memory constraints (SRAM is smaller than 256kB, and FLASH is read-only). The available resources on these devices are orders of magnitude smaller than those





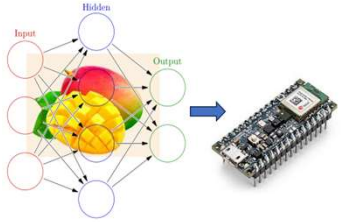
Research Perspective

"Today's large model might be tomorrow's *tiny model*."
[1] Lin et al., 2024

Agenda:

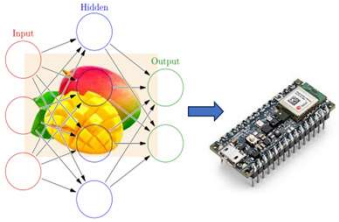
- What is the Magic Wand?
- Explanation of the Magic Wand?
- Why the Magic Wand Is Important?
- Lab: Building a Magic Wand

What is the Magic Wand?



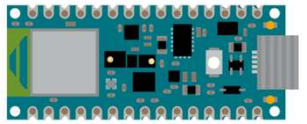
The Magic Wand example demonstrates how a very small microcontroller board, like the Arduino Nano 33 BLE Sense or SparkFun Edge, can run a Neural Network that recognizes simple hand-waving gestures using data from an onboard Inertial Measurement Unit (IMU).

What is the Magic Wand? . . .



The Arduino Nano 33 BLE Sense revision 2 uses the BMI270 IMU for detecting hand-waving gestures.

What is an Arduino Nano BLE Sense?. . .



Functional Overview

Version 2

Top of Board

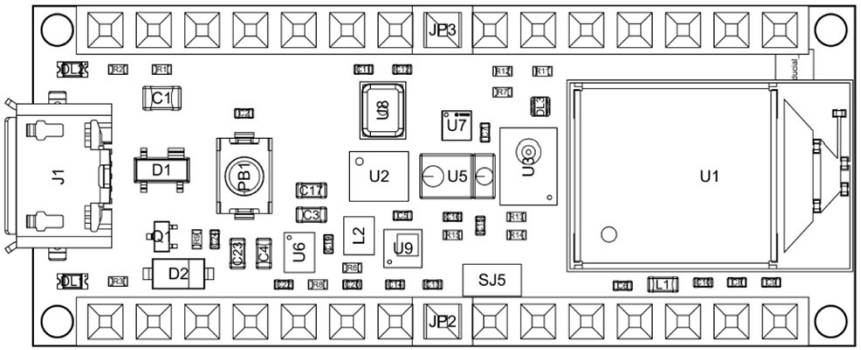


Illustration courtesy of Arduino.cc

Ref.	Description	Ref.	Description
U1	NINA-B306 Module Bluetooth® Low Energy 5.0 Module	U6	MP2322GQH Step Down Converter
U2	BMI270 Sensor IMU	PB1	IT-1185AP1C-160G-GTR Push button
U3	MP34DT06JTR MEMS Microphone	U8	HS3003 Humidity Sensor
U7	BMM150 Magnetometer IC	DL1	Led L
U5	APDS-9660 Ambient Module	DL2	Led Power
U9	LPS22HBTR Pressure Sensor IC		

Question 1

What idea does the Magic Wand demonstrate?

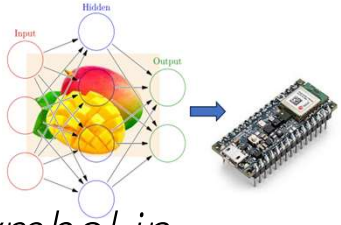
a) How large processors can run a neural network.

b) How 8-bit microcontrollers can run a neural network.

c) How very small microcontrollers can run a neural network.

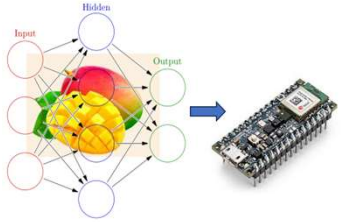
d) none of the above

What is the Magic Wand?. . .



It is named “Magic Wand” because the gesture looks like *drawing a symbol in the air*—as if the device were a wand casting a spell.

Explanation of the Magic Wand



System Overview

The Magic-Wand project is composed of the following stages:

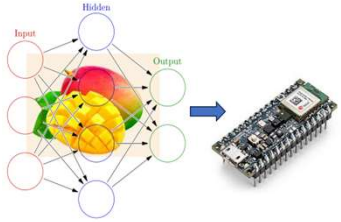
1. Data Acquisition

   Collect IMU data (accelerometer + gyroscope) from the microcontroller while the user performs gestures.

2. Pre-Processing

   Convert the time-series IMU samples into features suitable for a neural network (waveform windows).

Explanation of the Magic Wand



3. Model Architecture

Use a small neural network:

- Typically, a 1D convolutional neural network (Conv1D / CNN)
- Or a fully connected multilayer perceptron (MLP) in earlier versions
- Designed to fit in tens of kilobytes, not megabytes.

4. Training

Train the neural network on collected and labeled gesture examples.

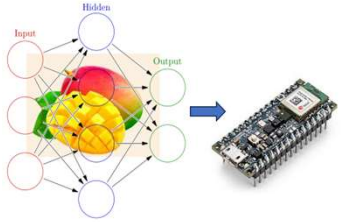
5. Conversion to TensorFlow Lite Micro

Quantize the model (int8), export to .tflite, and convert to a C array (model.h).

6. Deployment to MCU

Run inference on the embedded processor in real-time using IMU streams.

Explanation of the Magic Wand



2. What Gestures Does Magic-Wand Recognize?

The standard example uses three “air-drawing” gestures:

a) Wing

Flap-like gesture: up-down-up-down motion

b) Ring

A circular, loop-like movement

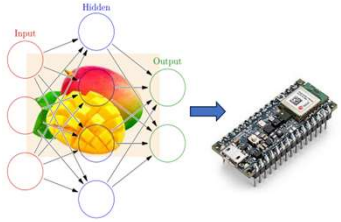
c) Slope

A diagonal upward or downward movement

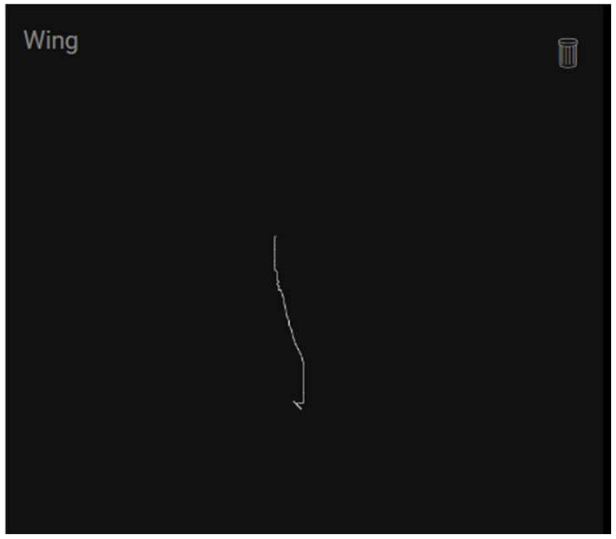
Note:

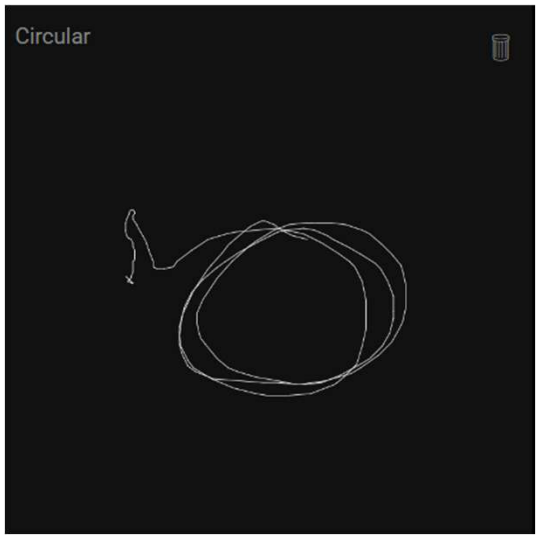
A no-gesture class is included to reduce false positives.

Explanation of the Magic Wand. . .



Recognize Magic-Wand Gestures



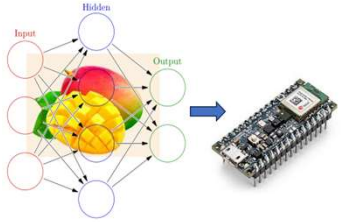


Question 2

Which gesture is not recognized by the Magic Wand?

a) swing
b) wing
c) ring
d) slope

Explanation of the Magic Wand. . .



3. Input Data: IMU Signals

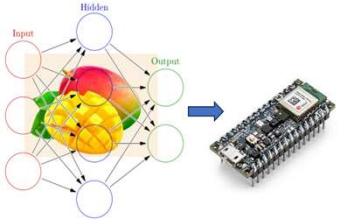
The IMU provides six sensor channels:

- Accelerometer: ax, ay, az
- Gyroscope: gx, gy, gz

Typical parameters:

- Sampling rate: 100 Hz
- Window length: 1 second → 100 samples
- Each input tensor shape:

Explanation of the Magic Wand. . .



What is a Tensor Shape?

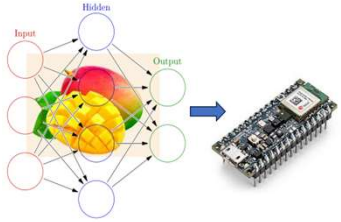
- A tensor shape is a tuple of integers that describes the length (number of elements) of each dimension (axis) in a tensor.

Example Tensor Shape: `(100, 6)`

- A tensor is a multidimensional array of numerical data.
- A tensor shape is a tiny dataset that is appropriate for a microcontroller.

Note:
A tuple is a finite sequence or ordered list of numbers.

Explanation of the Magic Wand. . .



4. Preprocessing and Feature Extraction

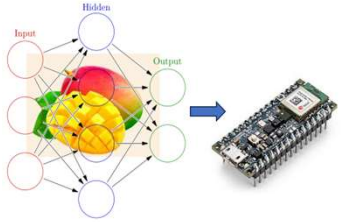
Raw IMU streams → sliding windows of 100-sample windows.

Key preprocessing steps:

- **Buffering**: collect 100 sequential readings.
- **Normalization**: scale sensor values.
- **Tensor creation**: shape into (100 time steps × 6 channels).

No Fast Fourier Transforms (FFT) is used in this example—it's purely time-domain CNN.

Explanation of the Magic Wand. . .



What is a Sliding Window?

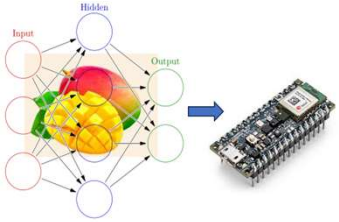
A sliding window is a technique used in signal processing, machine learning, and TinyML to analyze continuous data streams, like:

- IMU,
- audio,
- ECG,
- or vibration signals.

The technique involves breaking the data stream into small, overlapping segments.

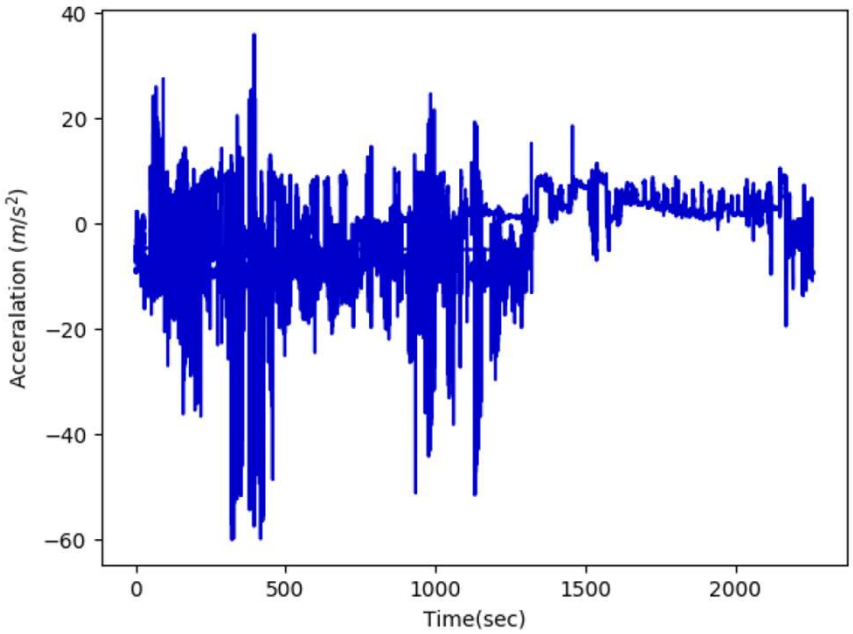
It is one of the *most important* concepts in the Magic-Wand TinyML application.

Explanation of the Magic Wand. . .



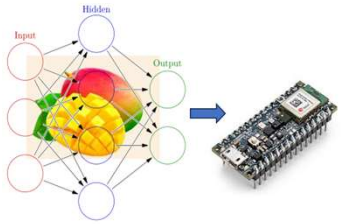
What is a Sliding Window?

Example: Accelerometer data



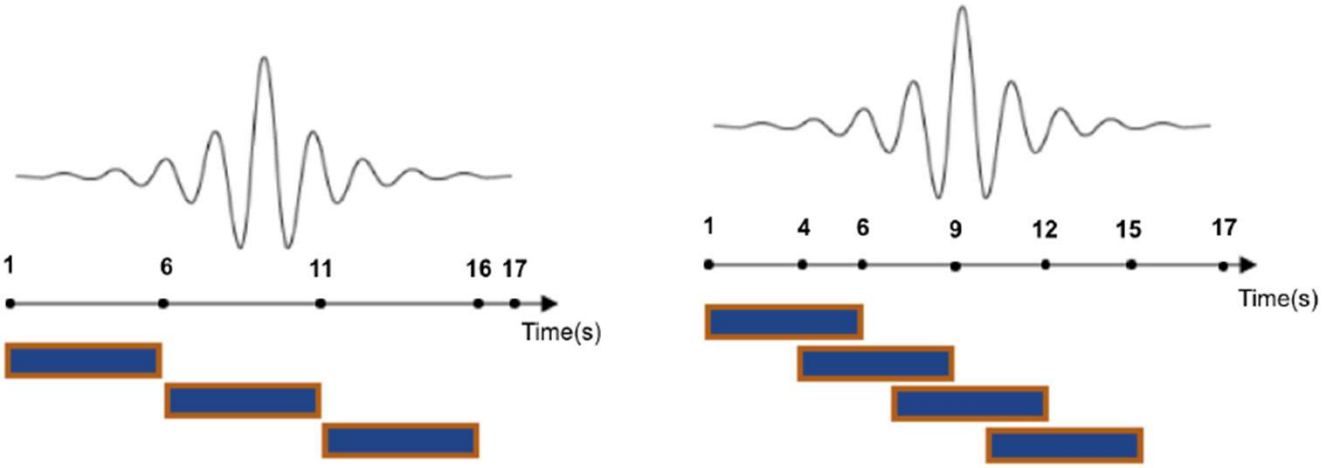
Dehghani et al.[5]

Explanation of the Magic Wand. . .



What is a Sliding Window?

Examples of Slicing Window



Dehghani et al.[5]

(a) Non-overlapping

(b) Overlapping

Question 3

What is a sliding window?

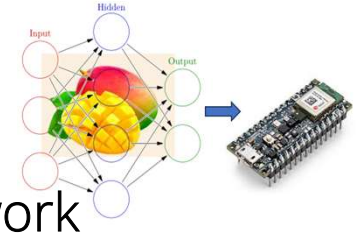
a) A technique involving processing a data stream into categories.

b) A technique involving moving a data stream into overlapping segments.

c) A technique involving the breaking of a data stream into small overlapping segments.

d) none of the above

Explanation of the Magic Wand. . .



5. Neural Network Architecture

The official TensorFlow example uses a 1D Convolutional Neural Network (CNN).

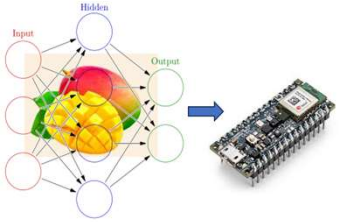
A typical architecture for the 1D CNN is shown below.

```
Input → Conv1D → ReLU → MaxPool → Conv1D → ReLU
      → Flatten → Dense → Dense(Output=4 classes)
```

Reason for CNN selection:

- Captures temporal patterns in IMU waveforms.
- Efficient on small microcontrollers.
- Very small parameter count (tens of KB).

Explanation of the Magic Wand. . .



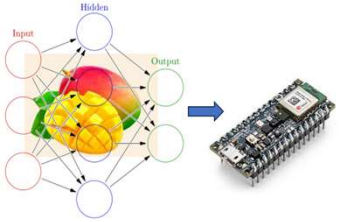
Final model memory:

- ~20–30 KB RAM
- ~20–25 KB Flash

Therefore, fits easily on:

- Arduino Nano 33 BLE Sense
- STM32 boards
- ESP32 boards
- SparkFun Edge (ultra-tiny)

Explanation of the Magic Wand. . .



6. Quantization and TFLite Micro Conversion

To run on microcontrollers:

1. Convert to TensorFlow Lite
2. Apply full integer quantization (int8)
3. Convert `.tflite` → `.h` file

The C array is compiled into Flash and executed entirely on-device.

7. Inference Loop on the Microcontroller

The firmware continuously:

Step 1 — Read IMU Data

At 100 samples/second, append new readings to a buffer.

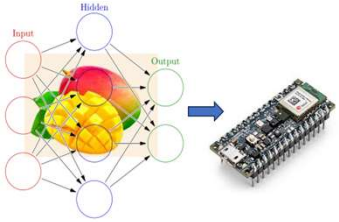
Step 2 — Generate a Window

Once 100 samples collected → create an inference window.

Step 3 — Run the Neural Network

Use TFLite Micro:

Explanation of the Magic Wand. . .



8. User Feedback / “Magic” Component

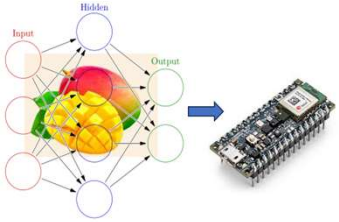
Typical output options:

- Built-in RGB LED changes color
- Serial Monitor prints gesture name
- Buzzers or motors activate (custom extensions)

Thus, it feels like casting spells:

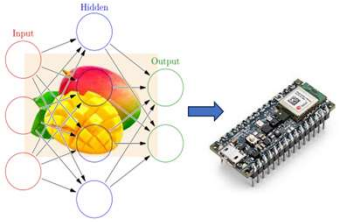
- Draw a *Wing* → LED glows blue
- Draw a *Ring* → LED glows green
- Draw a *Slope* → LED glows red

Why the Magic-Wand Project Is Important?



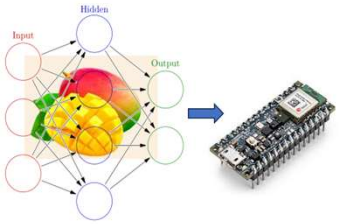
Why the Magic-Wand Project Is Important?
It demonstrates:
✓ Running deep learning on microcontrollers
Tiny memory footprint and no operating system.
✓ Real-time sensor inference
Fast enough to evaluate gestures at 10 Hz.
✓ End-to-end TinyML workflow
Dataset → model → quantization → deployment.

Why the Magic-Wand Project Is Important?. . .



✓ Practical applications
- Gesture-controlled wearables
- IoT human-machine interfaces
- Robotics
- Accessibility devices
- Educational TinyML teaching tool

Why the Magic-Wand Project Is Important?. . .



✓ Practical applications
- Gesture-controlled wearables
- IoT human-machine interfaces
- Robotics
- Accessibility devices
- Educational TinyML teaching tool

Question 4

Why is the Magic Wand project important?

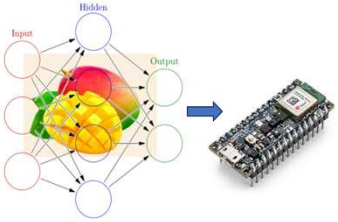
a) It demonstrates running reinforcement learning on microcontrollers.

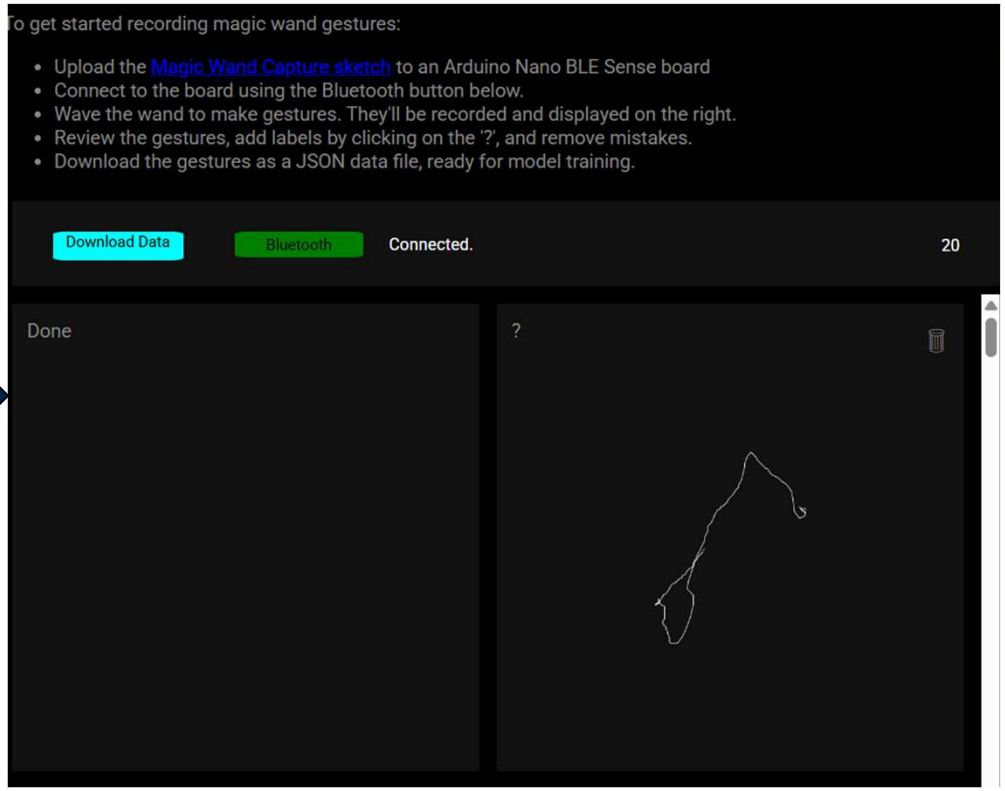
b) It demonstrates semi-real-time sensor inference.

c) It demonstrates running deep learning on microcontrollers.

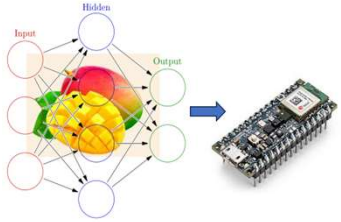
d) none of the above

Lab: Building a Magic Wand





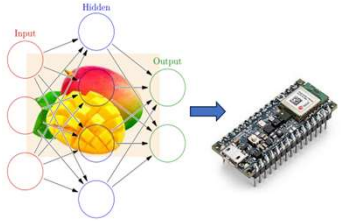
Lab: Building a Magic Wand. . .



Lab Objectives:

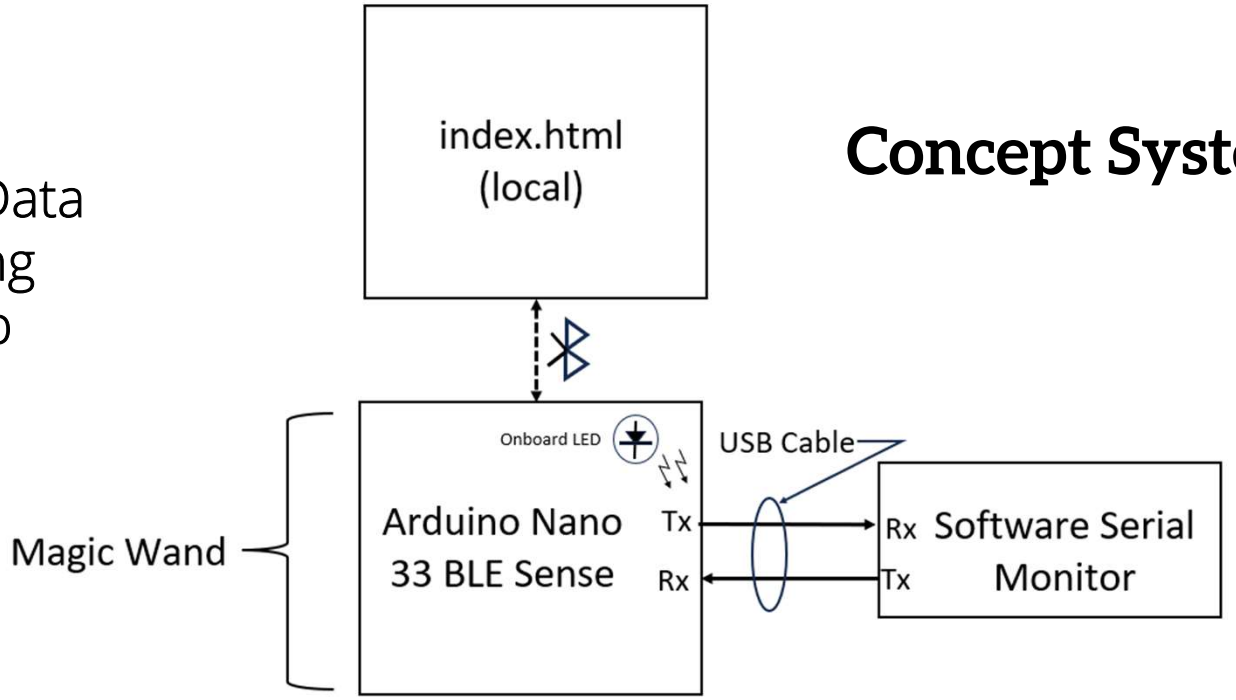
- Participants will learn to build a Magic Wand using the Arduino Nano 33 BLE Sense board and ordinary household items.
- Participants will learn to upload the Magic Wand code to the Arduino Nano 33 BLE Sense board.
- Participants will learn to perform gestures using the Magic Wand.
- Participants will learn to use a software terminal monitor to log gesture data from the Magic Wand.
- Participants will learn to analyze the gesture data produced by the Magic Wand.

Lab: Building a Magic Wand. . .



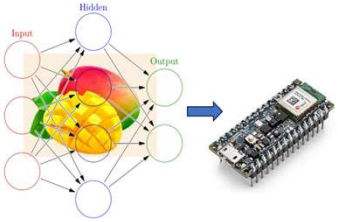
Simple Data Logging Setup

Concept System Block Diagram

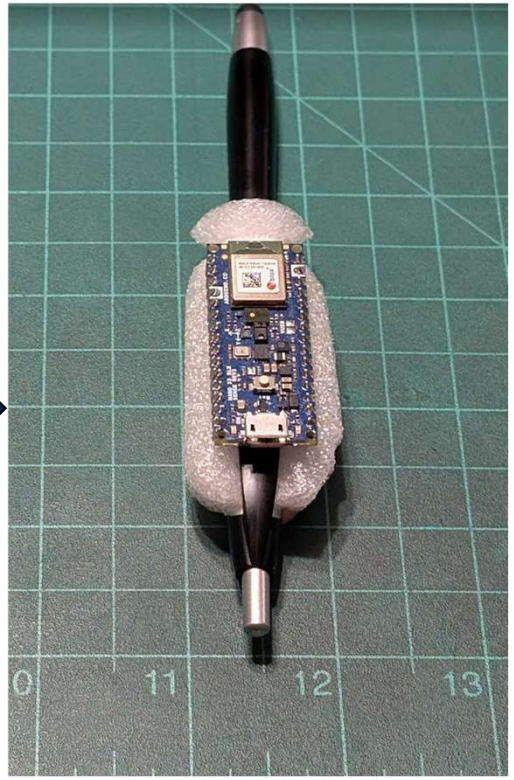


Note:
Software Serial Monitor can be Tera Term or Putty

Lab: Building a Magic Wand. . .

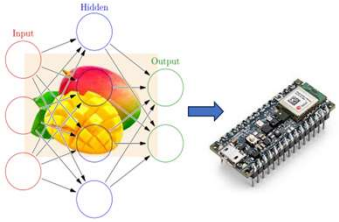


Ordinary household items with the Arduino Nano 33 BLE Sense



Assembled into a Magic Wand!

Lab: Building a Magic Wand. . .

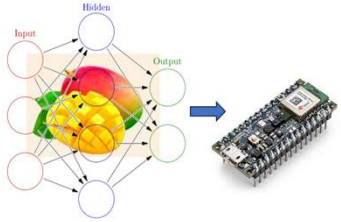


Upload the Magic Wand Code into the Arduino Nano 33 BLE Sense board

```
/* Copyright 2023 The TensorFlow Authors. All Rights Reserved.
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
    http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
==============================================================================*/

#include <ArduinoBLE.h>
#include "Arduino_BMI270_BMM150.h"
#include <TensorFlowLite.h>

#include <cmath>

#include "magic_wand_model_data.h"
#include "rasterize_stroke.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/micro/micro_log.h"
#include "tensorflow/lite/micro/micro_mutable_op_resolver.h"
#include "tensorflow/lite/micro/system_setup.h"
#include "tensorflow/lite/schema/schema_generated.h"

#define BLE_SENSE_UUID(val) ("4798e0f2-" val "-4d68-af64-8a8f5258404e")

#undef MAGIC_WAND_DEBUG
```

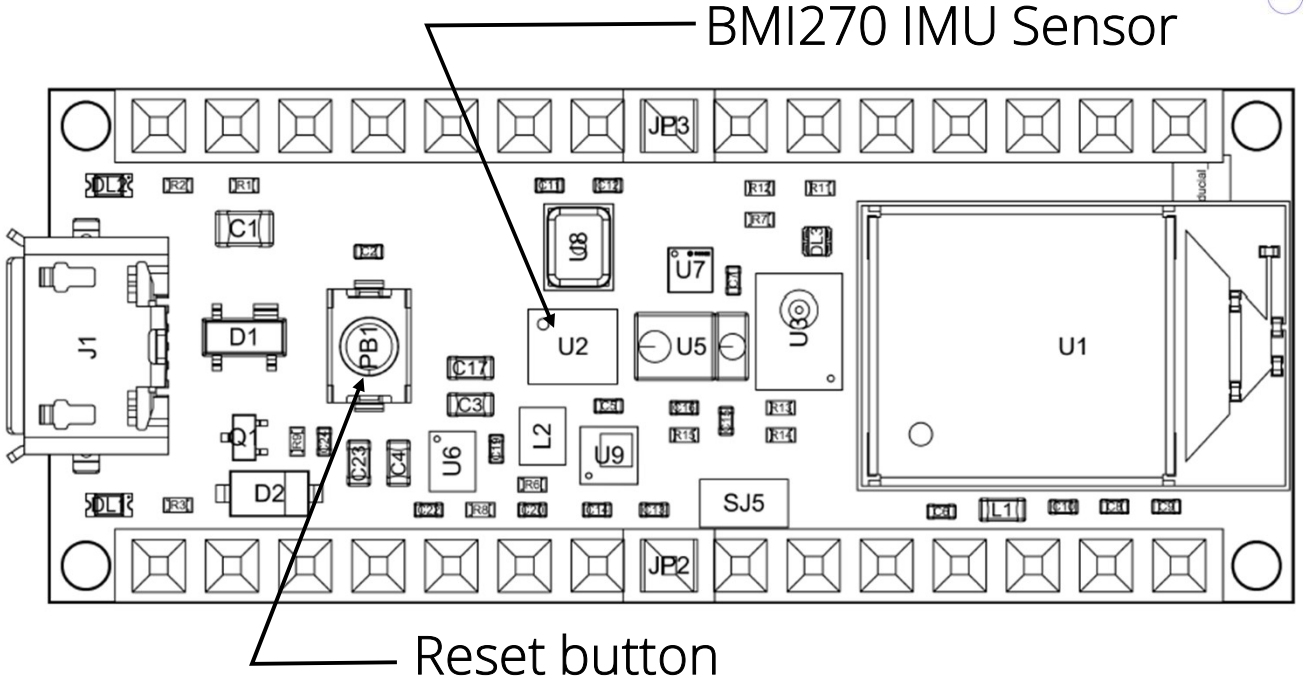
Arduino IDE: Files>Examples> Arduino TensorFlowLite> magic wand

Lab: Building a Magic Wand. . .

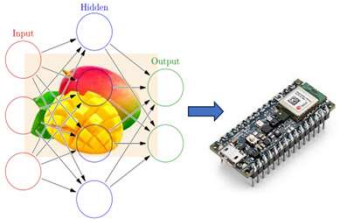


Software Code

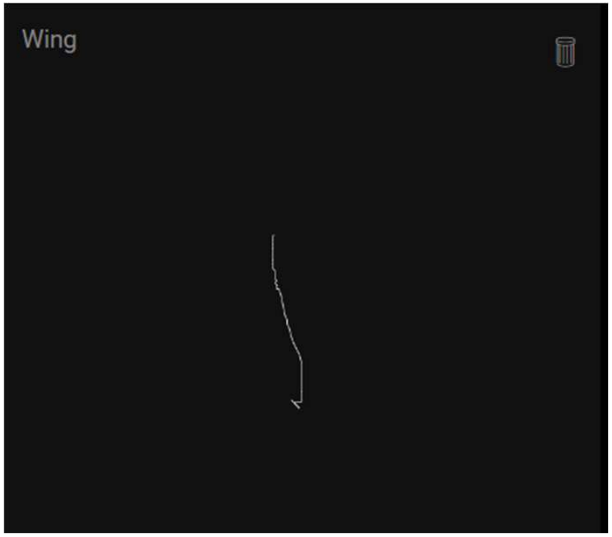
To put Arduino Nano 33 BLE Sense into program mode, press the Reset button twice

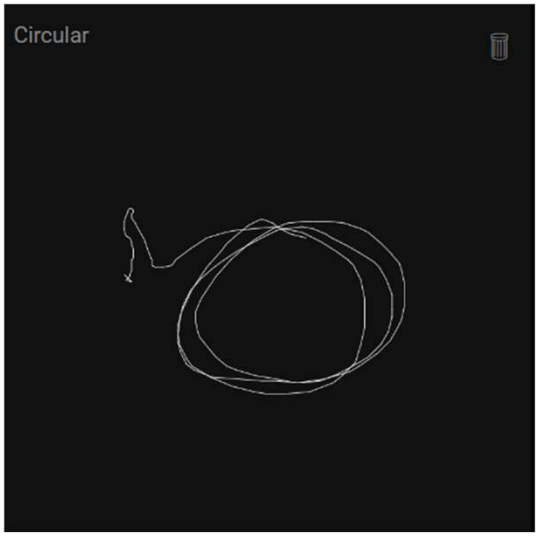


Lab: Building a Magic Wand. . .



Recognize Magic-Wand Gestures

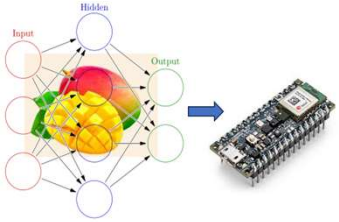


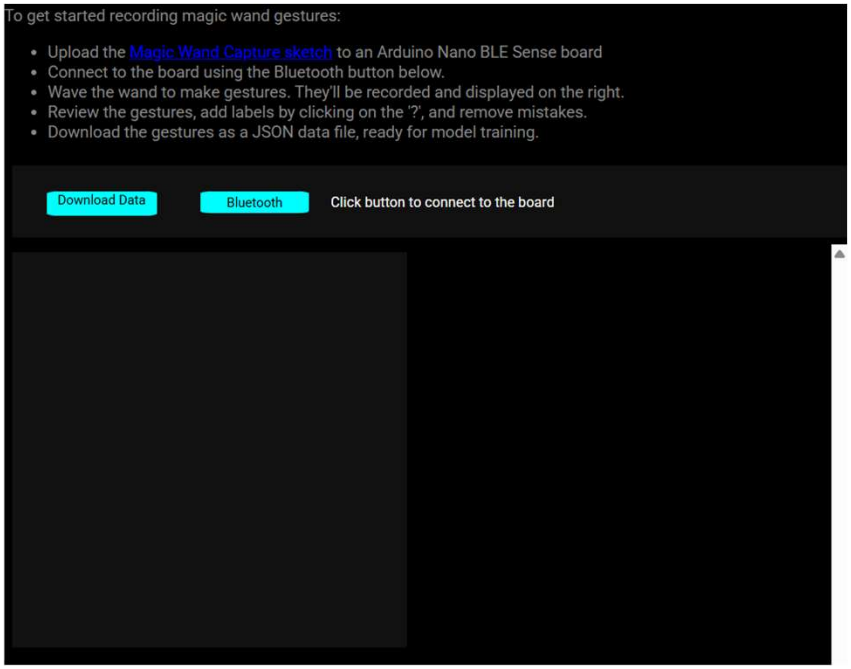


Lab: Building a Magic Wand. . .

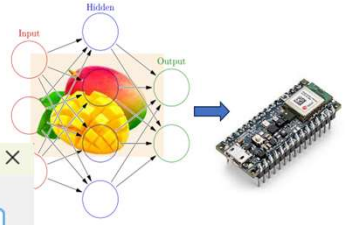
Open the index.html file to gain access to the gesture data collection tool for the Magic Wand

Bluetooth Pairing

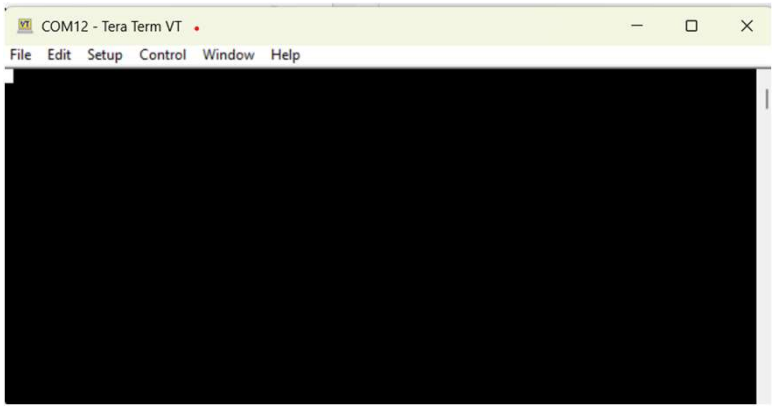


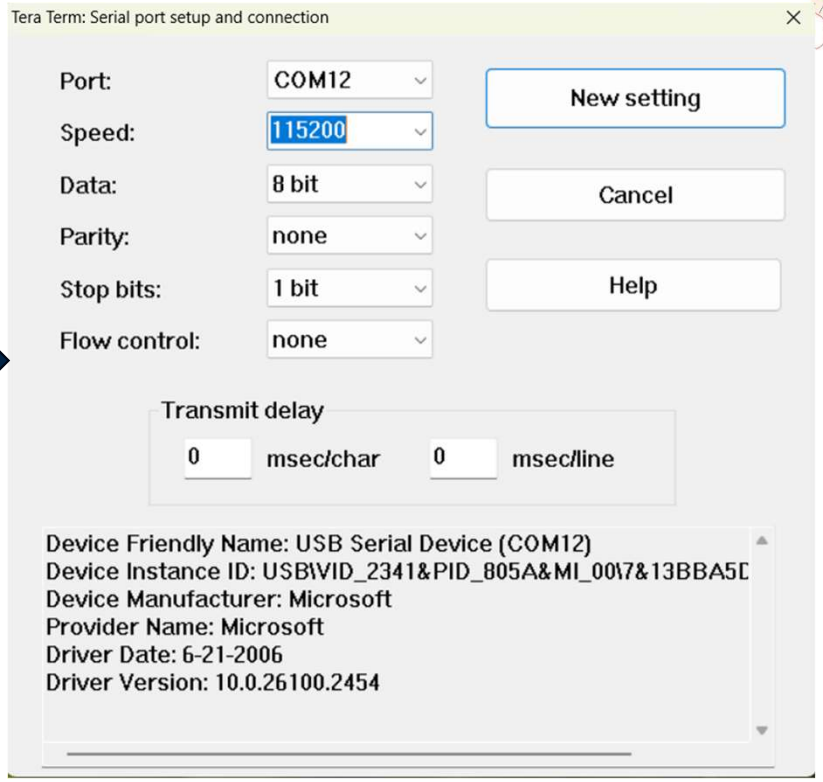


Lab: Building a Magic Wand. . .

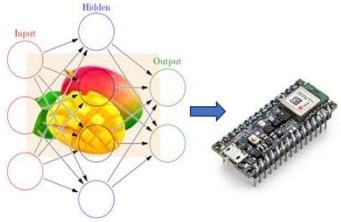


Tera Term Serial Port Setup

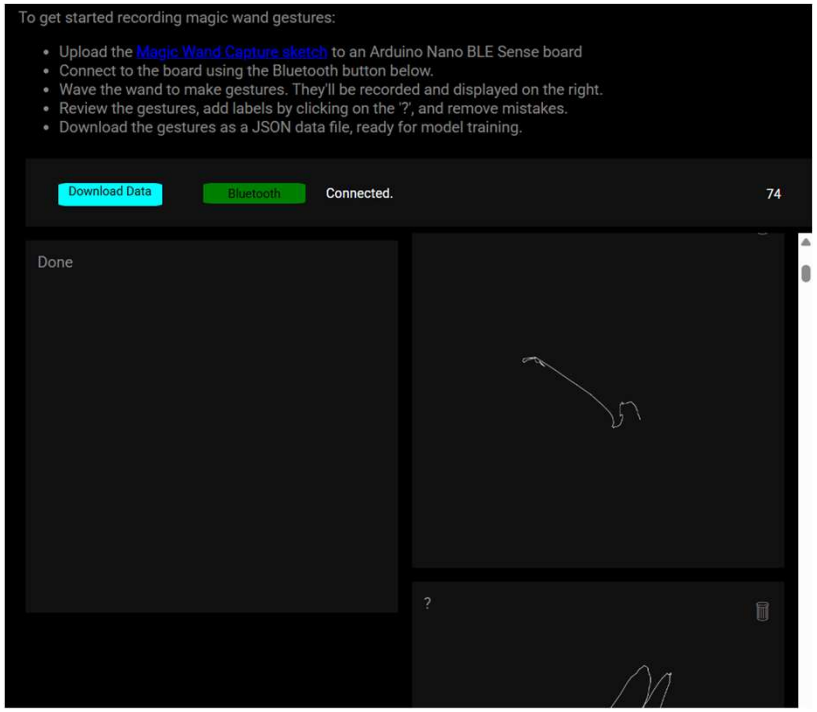




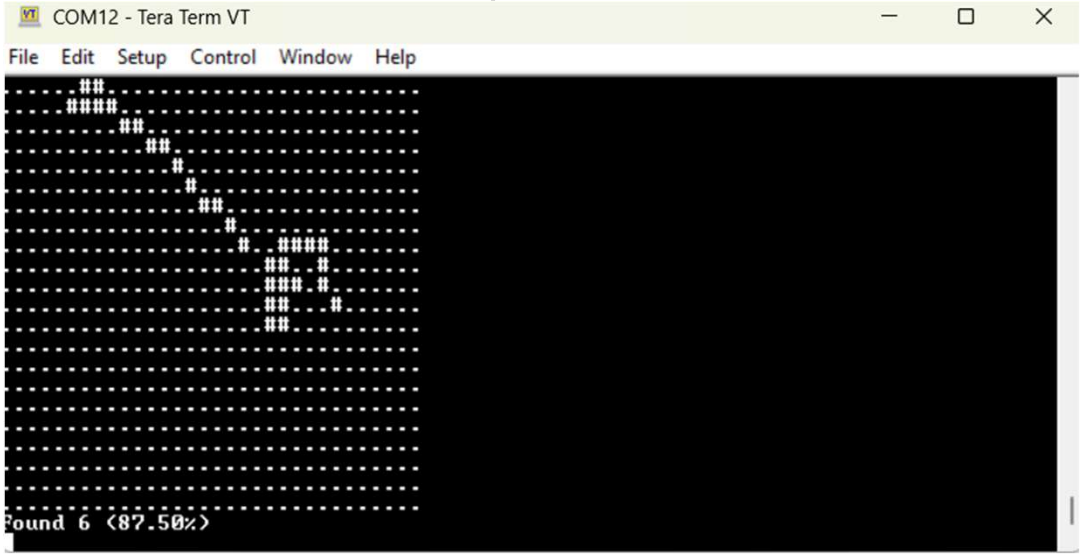
Lab: Building a Magic Wand. . .



Tera Term Logging Gesture Data: File>Log>create filename.

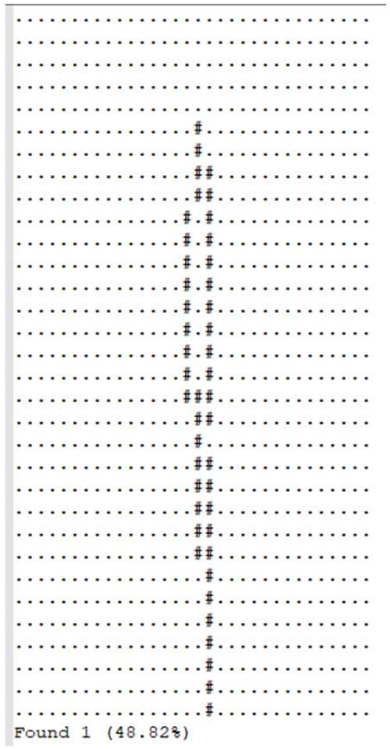


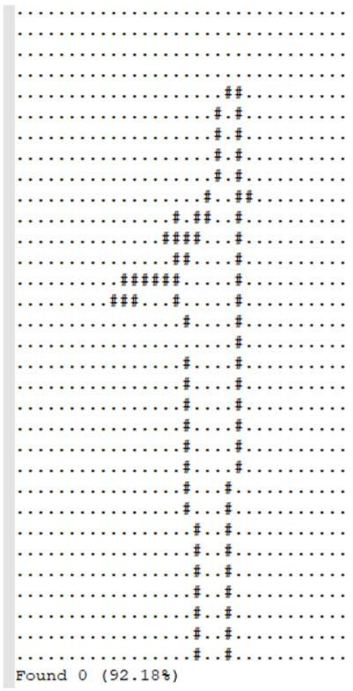
Slope

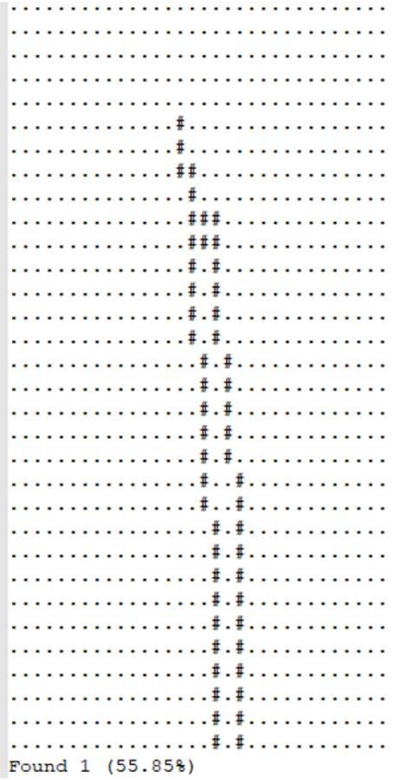


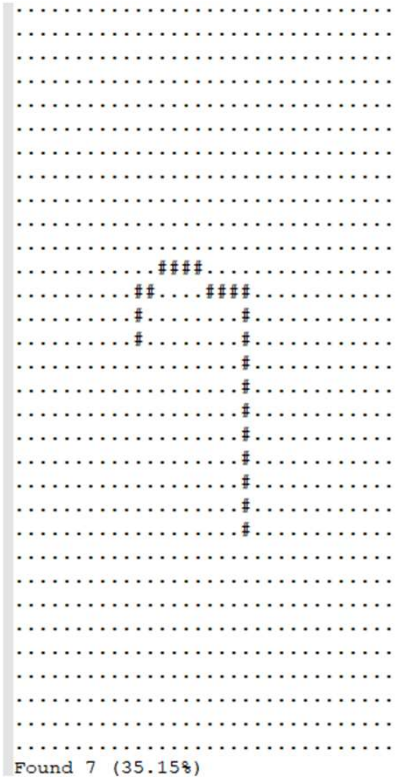
Lab: Building a Magic Wand. . .

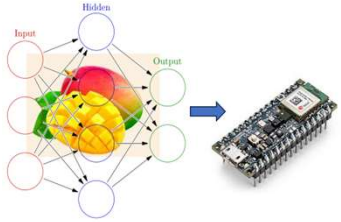
Logged Gesture Data





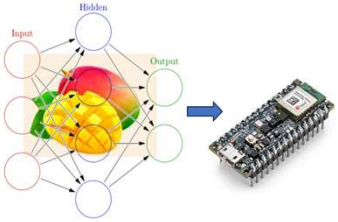






Wing

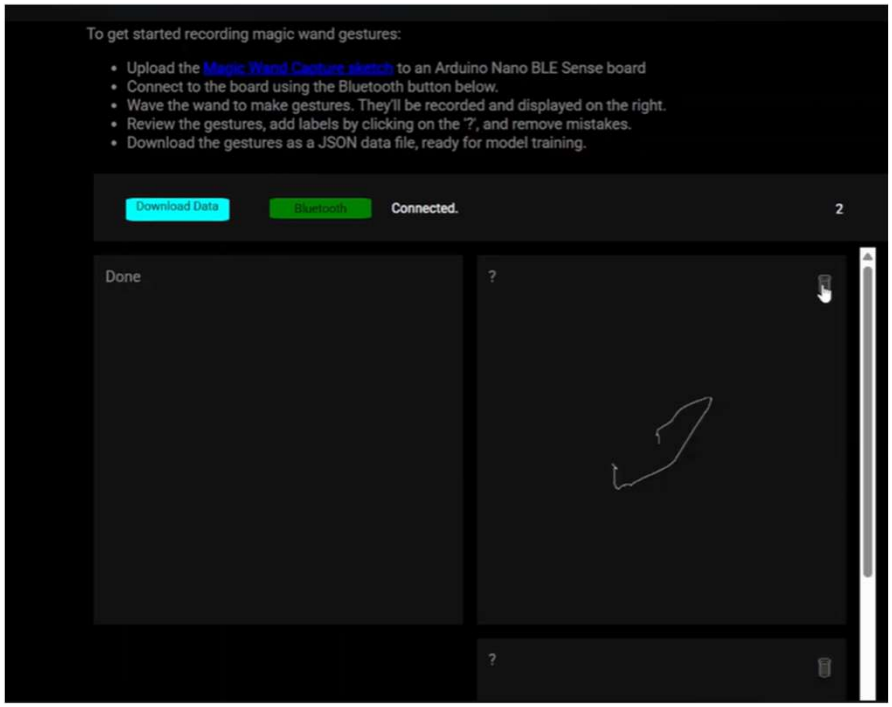
Lab: Building a Magic Wand. . .



Magic Wand Gesture Webpage

Click on the link to watch the Magic Wand Gestures

https://www.youtube.com/watch?v=joI8_DK3UEg



Question 5

Which component is required to enable program mode for the Arduino Nano 33 BLE Sense?

a) power button
b) Bluetooth app
c) BMI270 IMU
d) reset button

Thank you for attending

Please consider the resources below:

[1] J. Lin, L. Zhu, W. M. Chen, W. C.Wang, and S. Han, “Tiny machine learning: Progress and futures,” *arXiv*:2403.19076v2 [cs.LG], Jun. 2016. [Online]. Available: https://arxiv.org/abs/2403.19076
[2] R. Mathur, “A detailed intro to neural networks,” Aug. 2023. [Online]. Available: https://rikinmathur.substack.com/p/a-detailed-intro-to-neural-networks
[3] S. Heydari, Q. H. Mahmoud, “Tiny machine learning and on-device inference: A survey of applications, challenges, and future directions,” May. 2025. [Online]. Available: https://www.mdpi.com/1424-8220/25/10/3191
[4] D. Wilcher, “Designs News December 25 webinar code,” GitHub repository, Dec. 2025. [Online]. Available: https://github.com/DWilcher/DesignNews-WebinarCode/blob/main/December_25_Webinar_Code.zip
[5] A. Dehghani, O. Sarbishei, T. Glatard, and E. Shihab, “ A quantitative comparison of overlapping and non-overlapping sliding windows for human activity recognition using inertial sensors,” Nov. 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/22/5026

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets