CEC Continuing Education Center

DesignNews

Getting Started in TinyML with Arduino

DAY 1 : What is Tiny Machine Learning

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

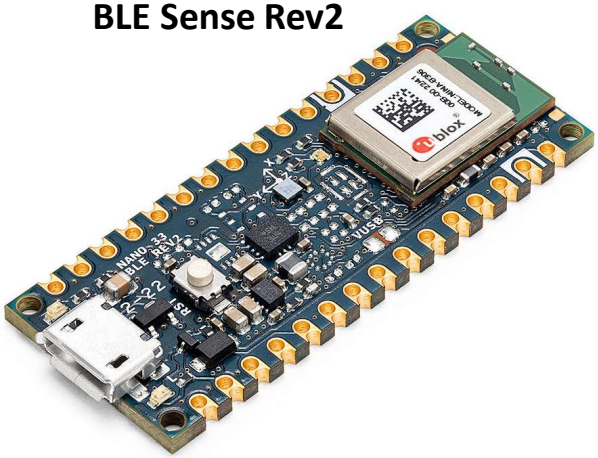
Visit ‘Lecturer Profile’ in your console for more details.

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.

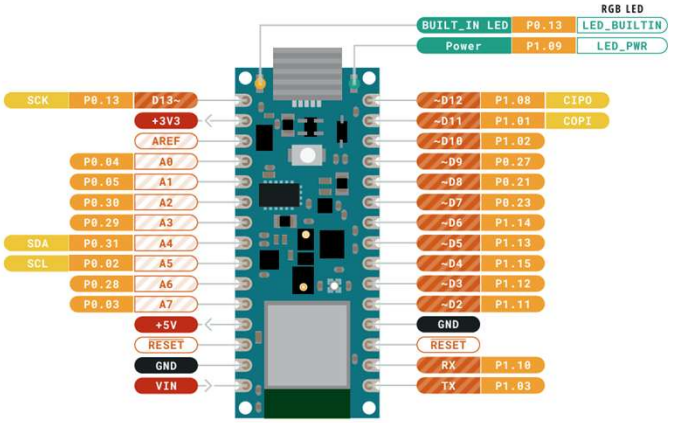
Fig. 1. Efficiency is critical for CloudML, EdgeML, and TinyML. CloudML targets high-throughput accelerators like GPUs, while EdgeML focuses on portable devices like mobile phones. TinyML further pushes the efficiency boundary, enabling powerful ML models to run on ultra-low-power devices such as microcontrollers.

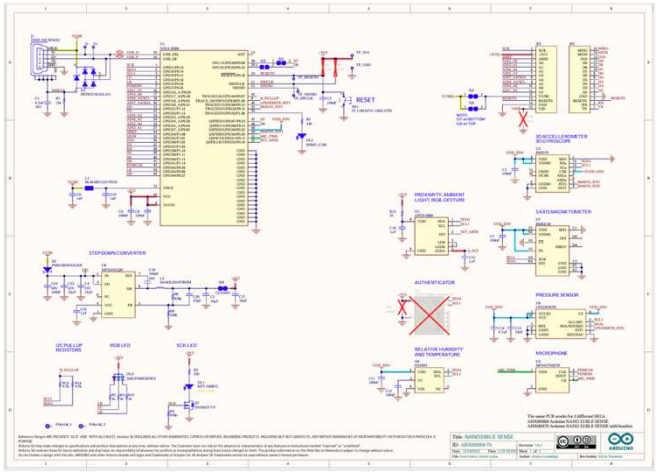
In addition to inference, we push the frontier of TinyML to enable on-device training on IoT devices. Irrevolutionizes EdgeAI through continuous and lifelong learning. Edge device can finetune the model on itself rather than transmitting data to cloud servers, which protects privacy. On-device learning 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





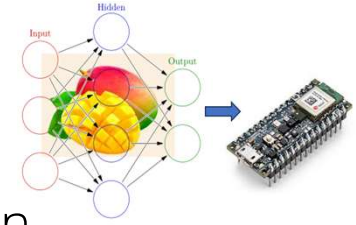
Research Perspective

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

Agenda:

- Tiny Machine Learning Definitions
- Common Terminology and Definitions
- TinyML Applications
- Arduino Nano 33 BLE Sense Overview
- Lab: Unique Hello World Demonstrator

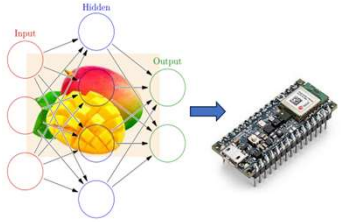
Tiny Machine Learning Definitions



TinyML (Tiny Machine Learning) is a field of machine learning focused on designing, optimizing, and deploying ultra-low-power, small-footprint ML models that can run directly on resource-constrained devices such as microcontrollers and embedded sensors.
These devices typically operate with capabilities.

- Kilobytes to a few megabytes of memory
- Millijoule-level energy budgets
- No operating system or limited RTOS
- Low computational capability (tens of MHz CPUs)

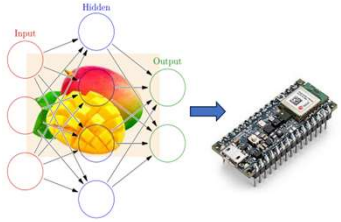
Tiny Machine Learning Definitions. . .



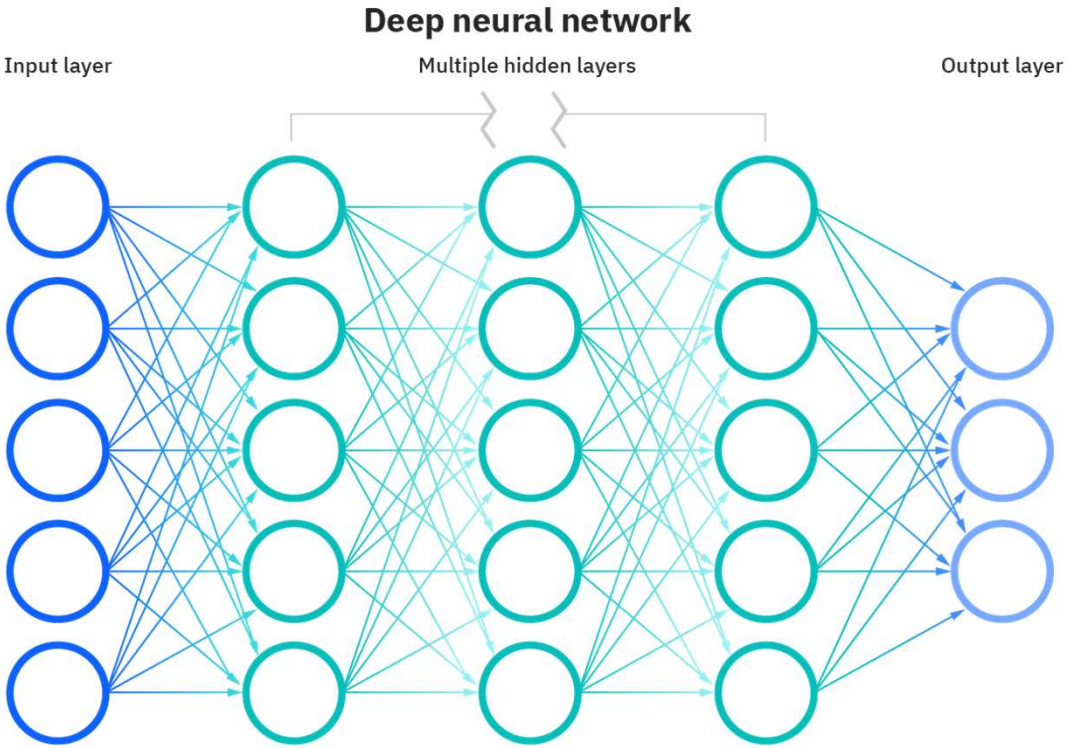
TinyML is a new development frontier in ML whereby deep learning is used for a variety of industrial and consumer applications. [1] [2].

- Deep Learning (DL) is a computing approach to model the brain's way of working.
- DL uses a neural network to train a model. The focus of the model is to understand the relationship between:
  a) various inputs
  b) and their corresponding outputs.

Tiny Machine Learning Definitions. . .



A Typical Deep Learning Model
[2] Mathur, 2013

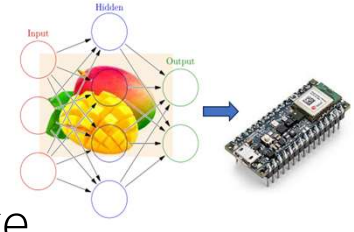


Question 1

What is TinyML?

a) A new approach to developing smart watches.
b) A new strategy to developing business organizations.
c) A new development frontier in ML whereby DL is used for a variety of industrial and consumer applications.
d) none of the above

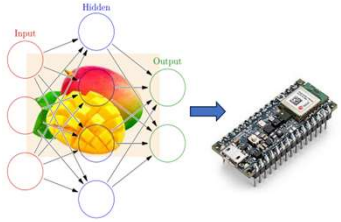
Tiny Machine Learning Definitions. . .



Machine Learning (ML) fundamentally is a technique that is able to make predictions using computers.

- Creating ML programs is different than traditional software development.
- In traditional software development, the algorithm is planned by the software engineer.
- In ML development, the software engineer needs an understanding of the data and the appropriate measurements to code the application.

Common Terminology and Definitions



What Is Inference?

Inference is the process of a trained model making a prediction on new data.

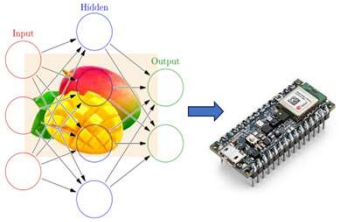
Once the model is trained:

a) You feed it a new sensor reading, audio signal, image, or IMU data

b) It computes outputs

c) You interpret the results (e.g., “gesture = up”)

Common Terminology and Definitions. . .



Inference in TinyML

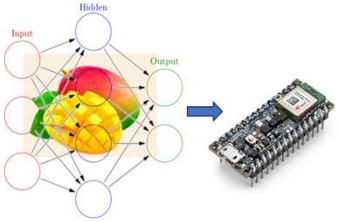
Inference happens on the microcontroller, not the cloud.

For example, with the Arduino Nano 33 BLE Sense:

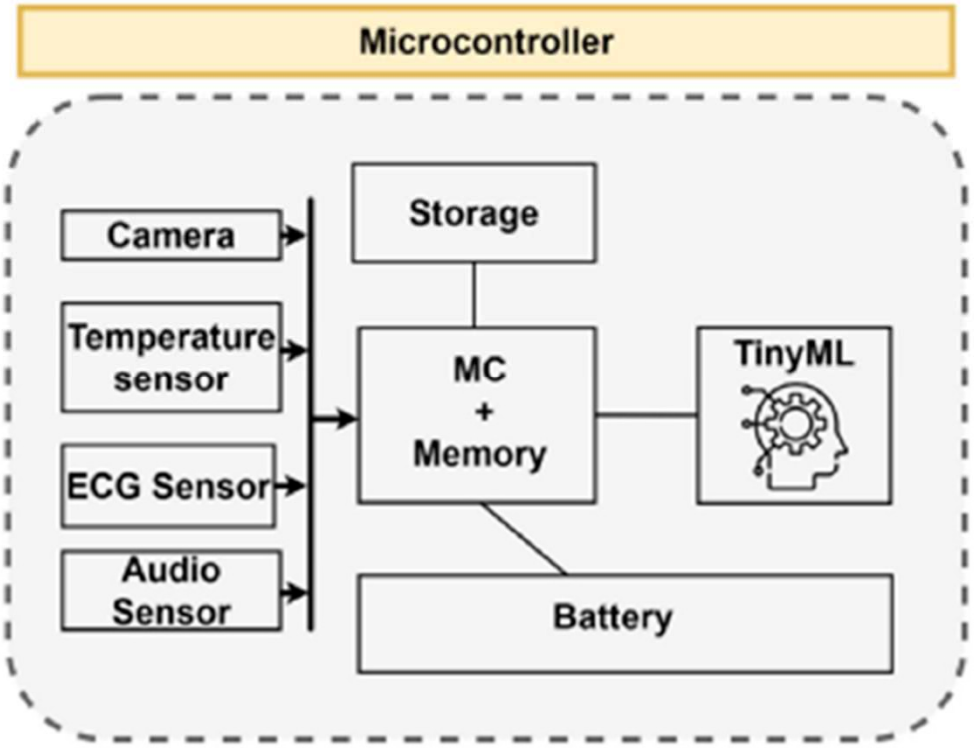
- Read 64 IMU samples
- Preprocess them
- Run the tiny neural network
- Output the predicted gesture
- Turn on an LED or send a BLE message

All of that is **one inference**.

Common Terminology and Definitions. . .



Inference in TinyML
Inference happens on the microcontroller, not the cloud.



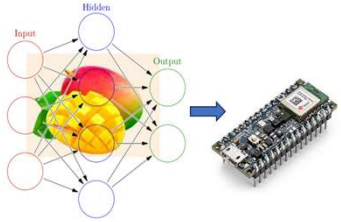
[3] Heydari & Mahmoud, 2025

Question 2

Which component is used by Inference?

a) microelectronics
b) microcontroller
c) digital ICs
d) none of the above

Common Terminology and Definitions. . .



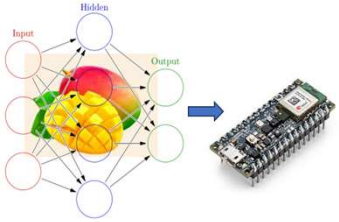
Epoch

An **epoch** is one complete pass of the entire training dataset through the neural network during training.

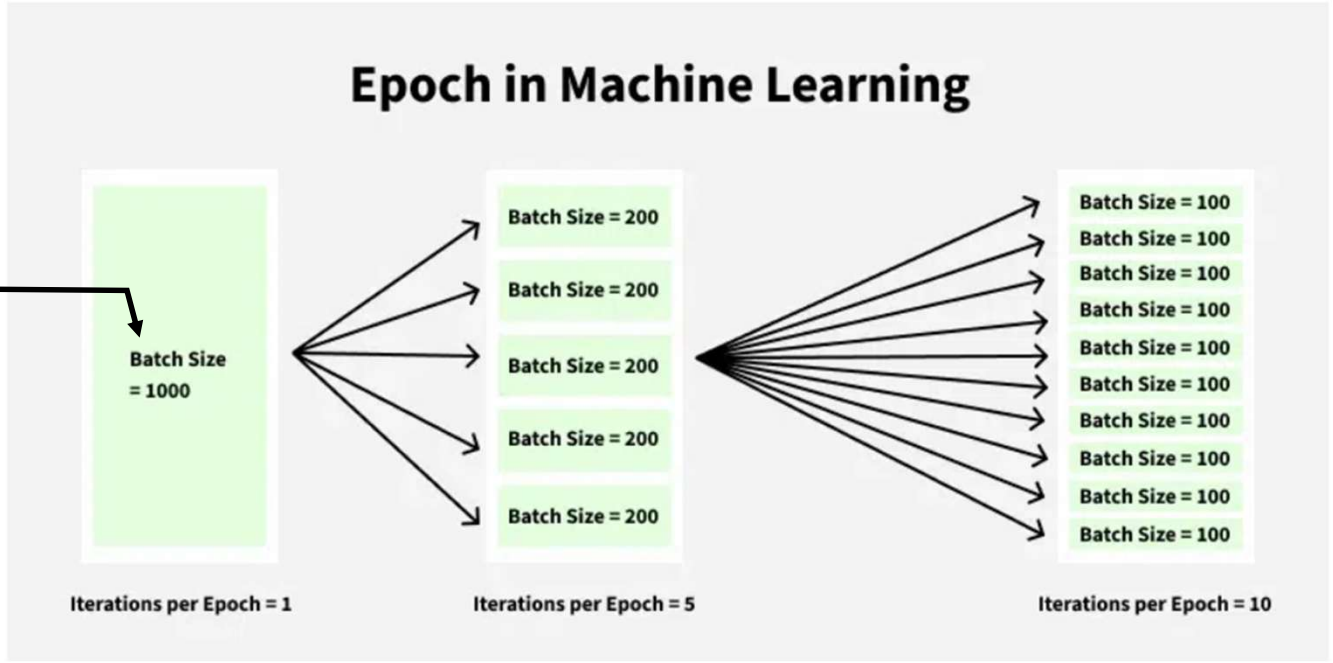
Another perspective on thinking of an epoch.

a) If you have 1,000 training samples, and you train for 30 epochs
b) Then your model sees each of those 1,000 samples 30 times during training.

Common Terminology and Definitions. . .

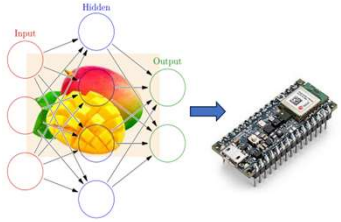


Epoch Illustrated Example:



Picture courtesy of geeksforgeeks

Common Terminology and Definitions. . .



Why multiple epochs?

Each epoch helps the model:

- Adjust its internal weights
- Reduce error
- Improve accuracy

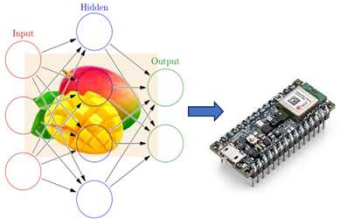
Models rarely learn enough with just 1 epoch.

Common TinyML practice:

a) 20–100 epochs

b) depending on model size and dataset.

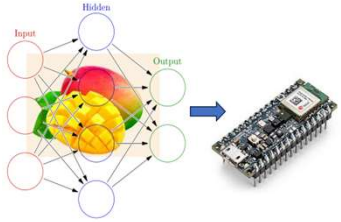
Common Terminology and Definitions. . .



Summary Table

Term	Phase of ML	What It Means	Example
Epoch	**Training phase**	One full pass through the training dataset	Model sees all IMU gesture samples once
Inference	**Deployment phase**	Model makes a prediction on new data	Board reads IMU, predicts "circle gesture"

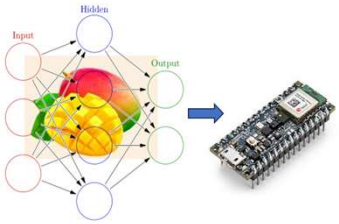
TinyML Applications



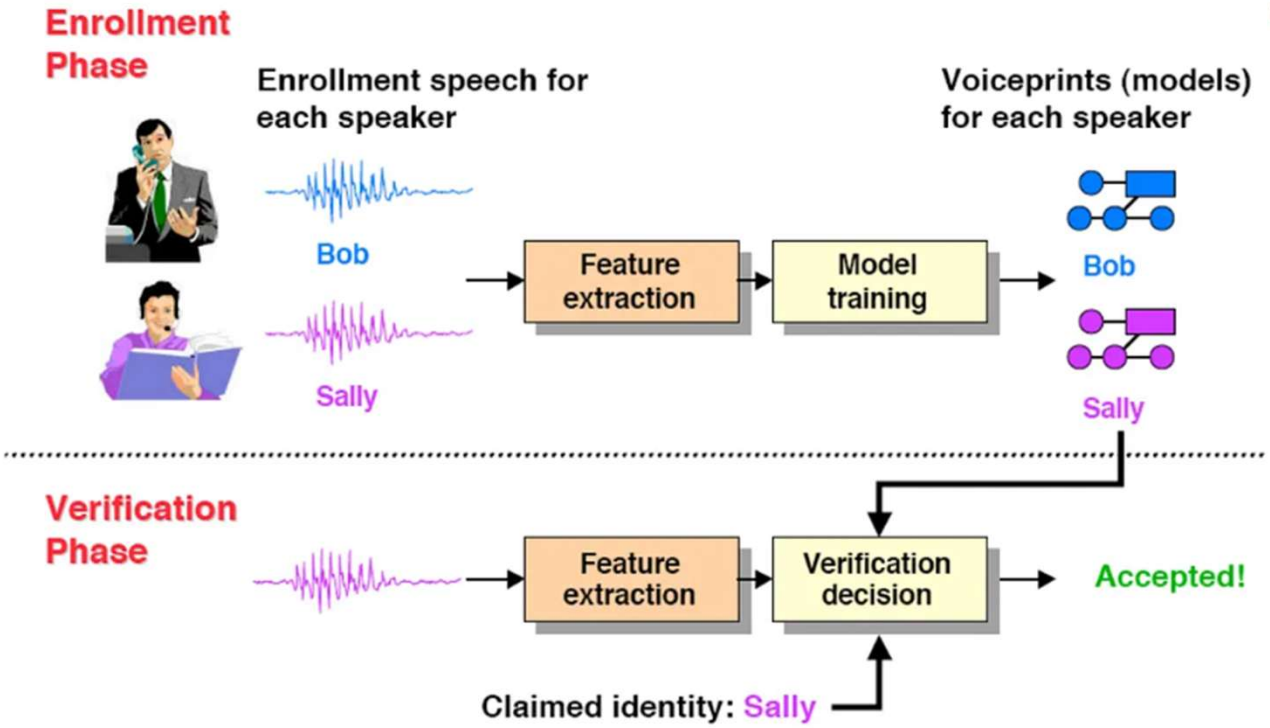
The democratization cost of the deep learning models for Internet of Things (IoT) devices, TinyML, has been applied to many practical applications. Some applications include:

- Personalized healthcare:
  - a) TinyML can allow wearable devices like smart watches can track the status or provide suggestions.
  - b) Body pose estimation for elderly healthcare.
- Wearable applications:
  - a) TinyML can assist people with wearable IoT devices for speech applications.
    - i. keyword spotting
    - ii. Automatic speech recognition
  - b) Speaker Verification- Voice to pre-registered voice print.

TinyML Applications. . .



Speaker Verification
Reynolds & Heck

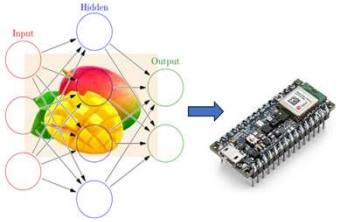


Question 3

In reviewing slide 21, which descriptor defines Speaker Verification?

a) automatic speech recognition
b) voice to pre-registered voice print
c) keyword spotting
d) none of the above

TinyML Applications. . .



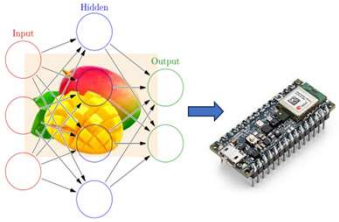
Smart Home:

a) TinyML can enable object detection.
b) image recognition
c) face detection on IoT devices
d) such items enable the building of smart environments

Human Machine Interface:

a) TinyML can enable human-machine interface applications.
    i. hand gesture
    ii. touch
b) TinyML is capable of predicting and recognizing sign languages.

TinyML Applications. . .



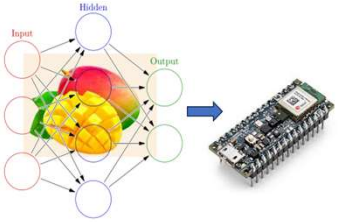
Wio Terminal

What's Inside?:

ATSAMD51P19 and ARM Cortex-M4F at 120MHz

SeeedStudio

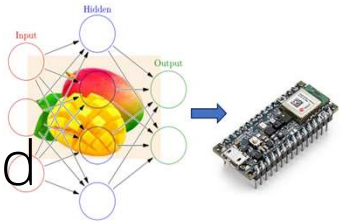
TinyML Applications. . .



Human Machine Interface:

Wio Terminal

Arduino Nano 33 BLE Sense Overview Description

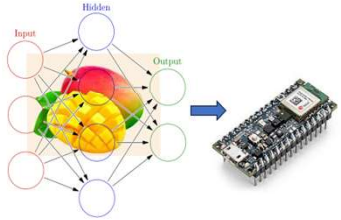


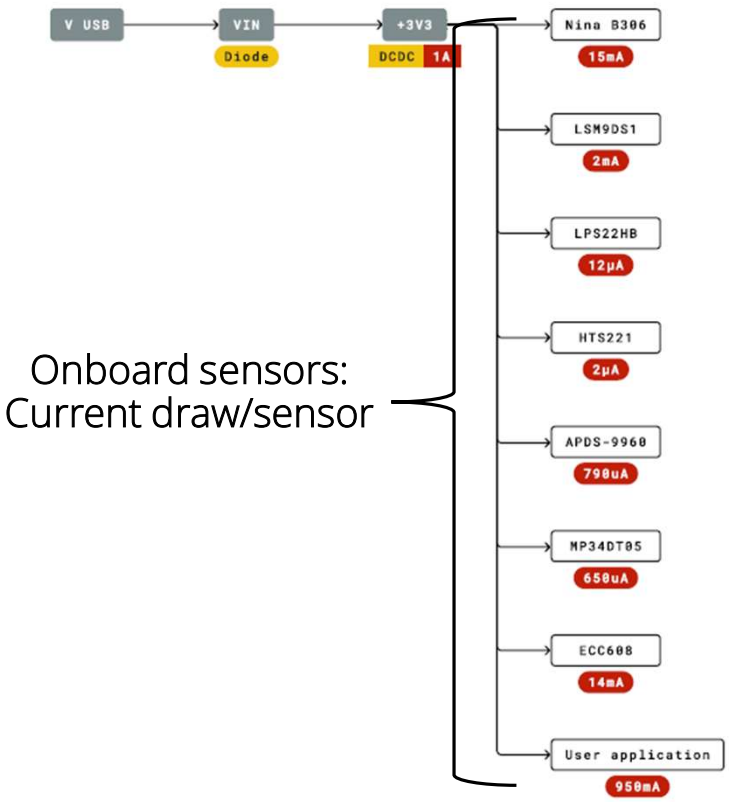
- An Internet of Things (IoT) platform to perform sensing and actuating in a physical environment.
- A Cortex M4F processor with an operating frequency of 64MHz.
- The Arduino Nano 33 BLE Sense board has the following items
  - a) Bluetooth Low Energy (BLE).
  - b) Eight onboard sensors
  - c) Analog and Digital port pins
- The Arduino Nano 33 BLE Sense board can be powered in 3 ways.
  - a) USB connector
  - b) Vin (header connector)
  - c) Vusb (header connector)

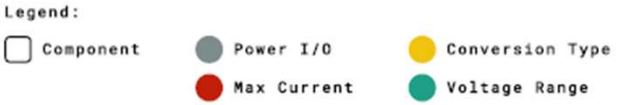
Arduino Nano 33 BLE Sense Overview

Functional Overview

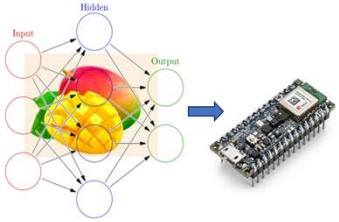
Approaches to Powering the Arduino Nano 33 BLE Sense board





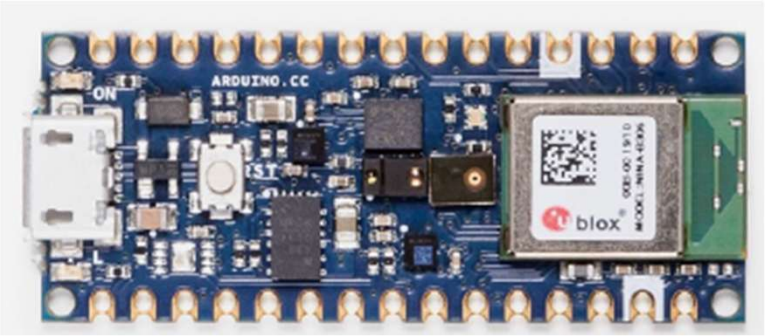


Arduino Nano 33 BLE Sense Overview. . .

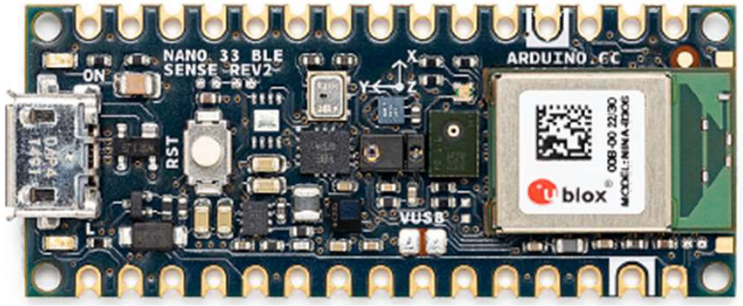


Functional Overview

Original Board

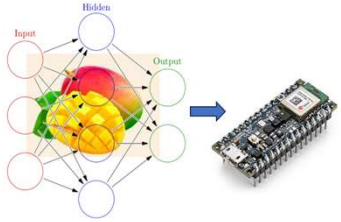


Version 2



Pictures courtesy of Arduino.cc

Arduino Nano 33 BLE Sense Overview. . .



Functional Overview

Version 2

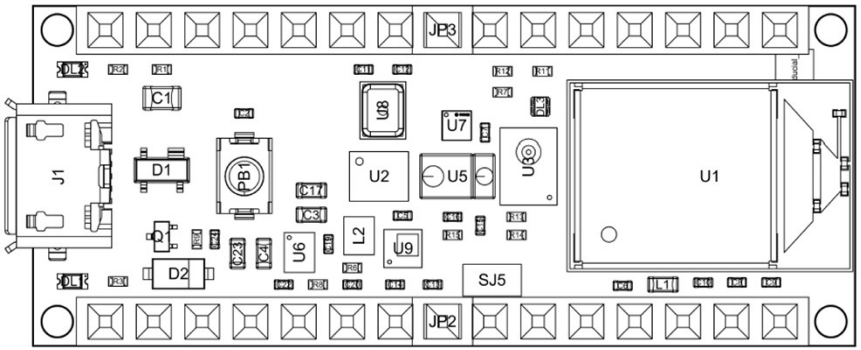
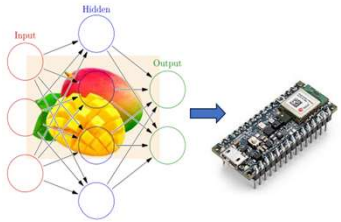


Illustration courtesy of Arduino.cc

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		

Arduino Nano 33 BLE Sense Overview. . .



Functional Overview

Version 2

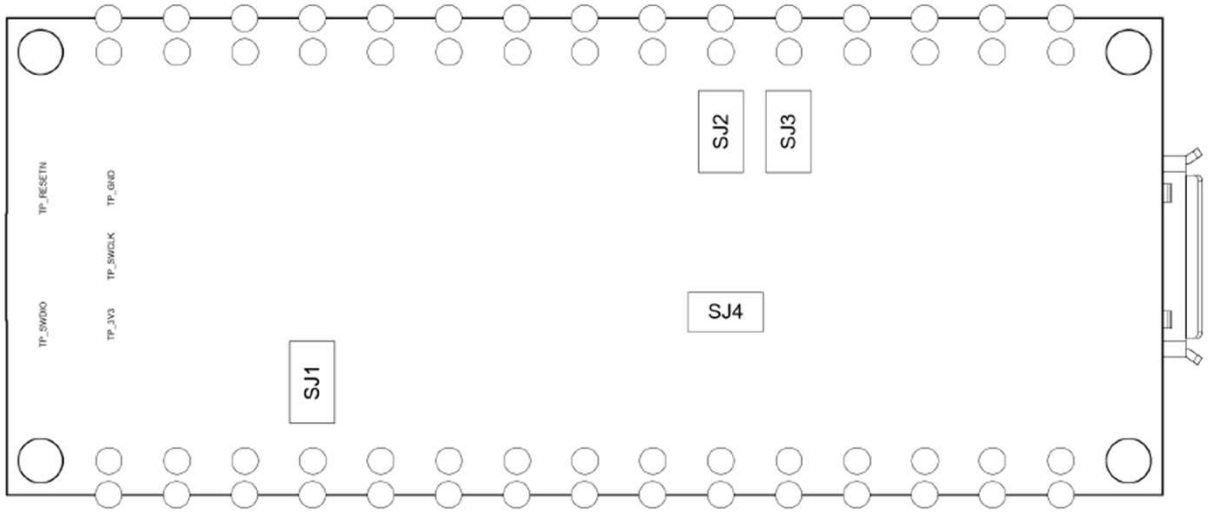


Illustration courtesy of Arduino.cc

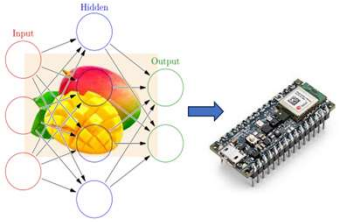
Ref.	Description	Ref.	Description
SJ1	VUSB Jumper	SJ2	D7 Jumper
SJ3	3v3 Jumper	SJ4	D8 Jumper

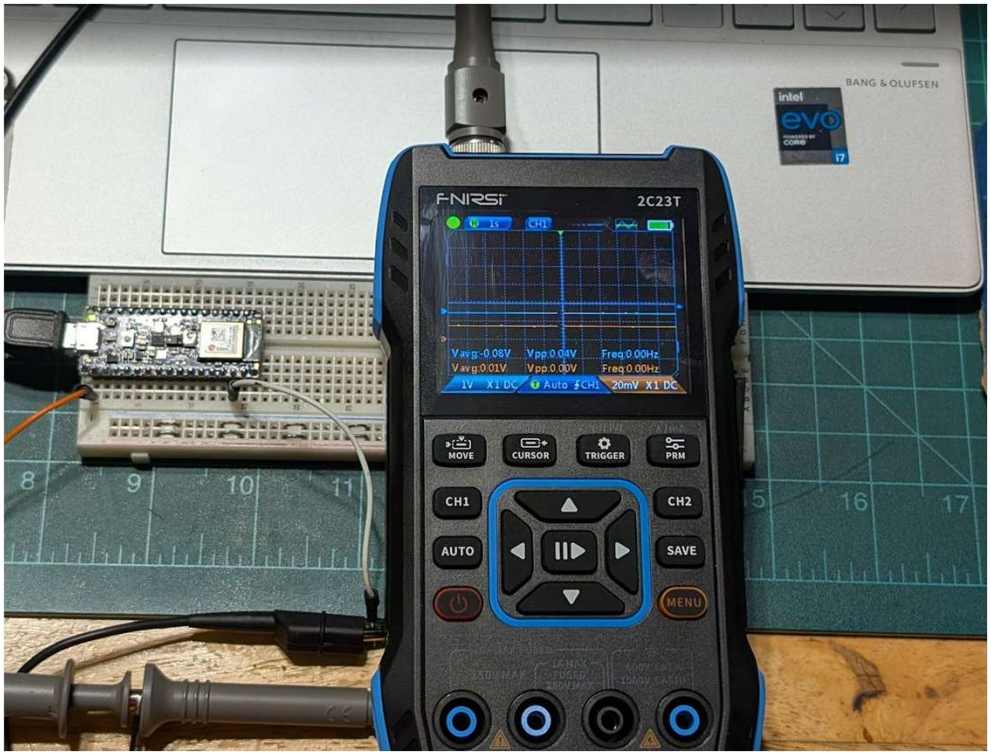
Question 4

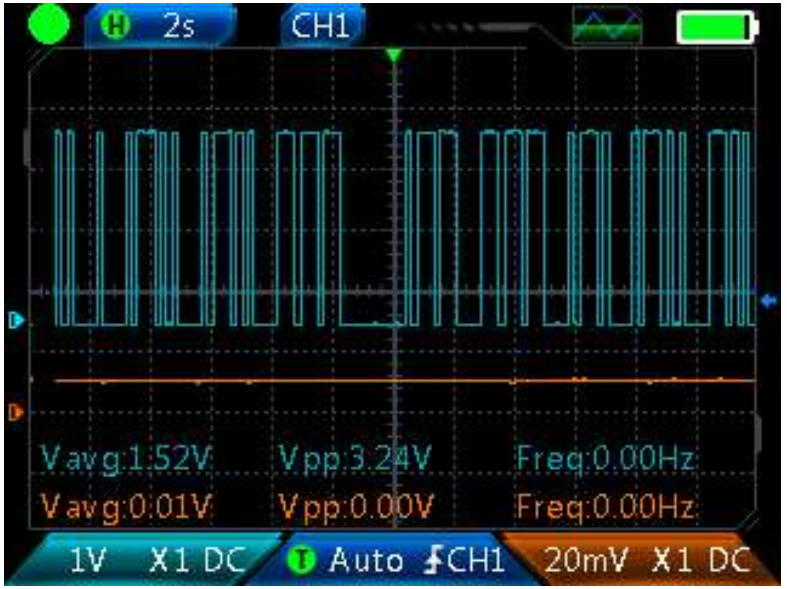
In reviewing slide 29, which component aligns with reference designator U3?

a) MP34DT06JTR MEMS Microphone
b) BMI270 Sensor IMU
c) APDS-9660 Ambient Module
d) none of the above

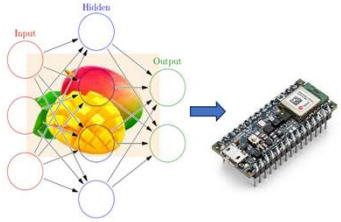
Lab: Unique Hello World Demonstrator







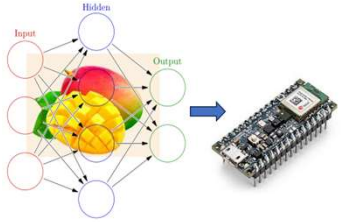
Lab: Unique Hello World Demonstrator. . .



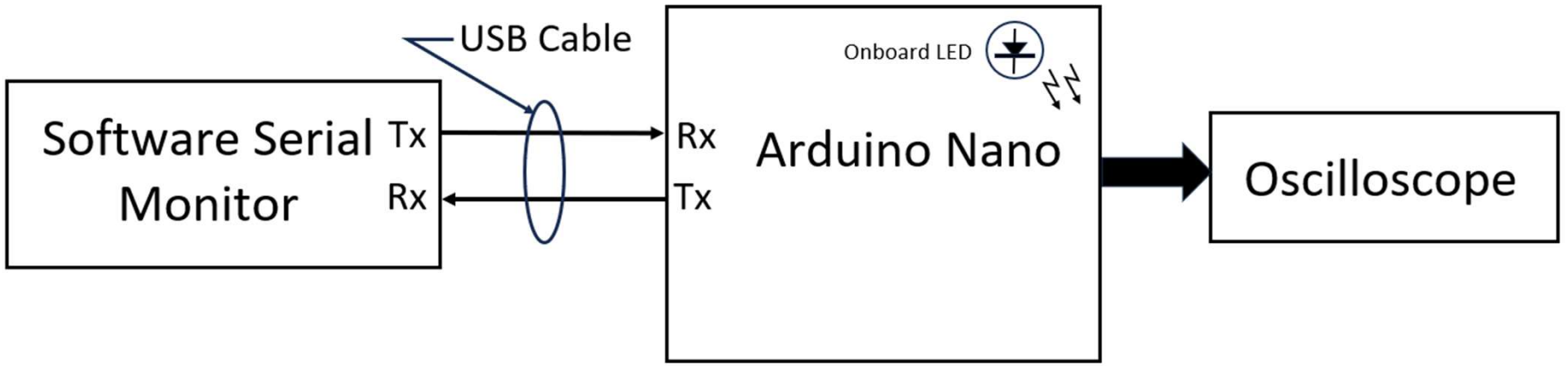
Lab Objectives:

- Participants will learn to obtain the Arduino Nano 33 BLE Sense board using the Arduino IDE Board Manager
- Participants will learn to program the Arduino Nano 33 BLE Sense to flash Morse code using the onboard LED.
- Participants will learn to upload the Morse Code Flasher program onto the Arduino Nano 33 BLE Sense and observe the message typed on a Software Serial Monitor.
- Participants will learn to attach an oscilloscope and observe the Morse Code data on the instrument's screen.

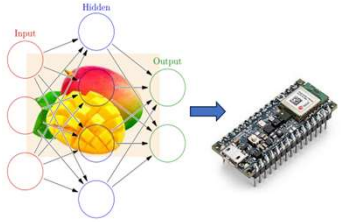
Lab: Unique Hello World Demonstrator. . .



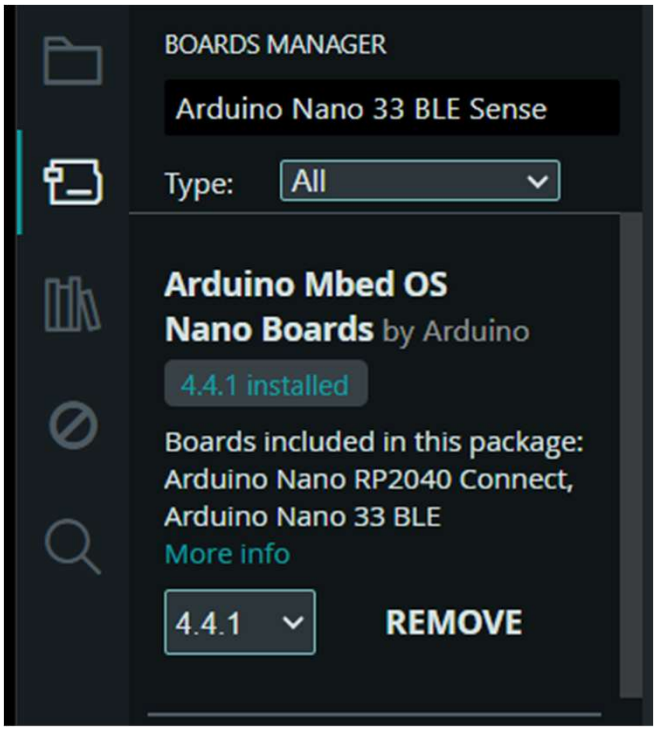
Concept System Block Diagram



Lab: Unique Hello World Demonstrator. . .

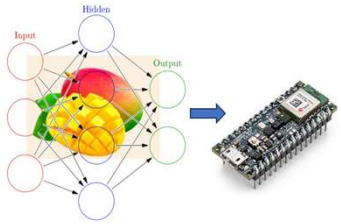


Arduino IDE version 2.3.6



Obtaining board resources for the Arduino Nano 33 BLE Sense

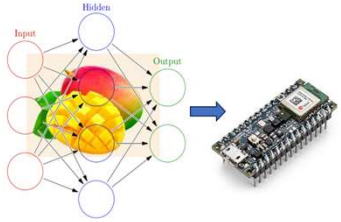
Lab: Unique Hello World Demonstrator. . .



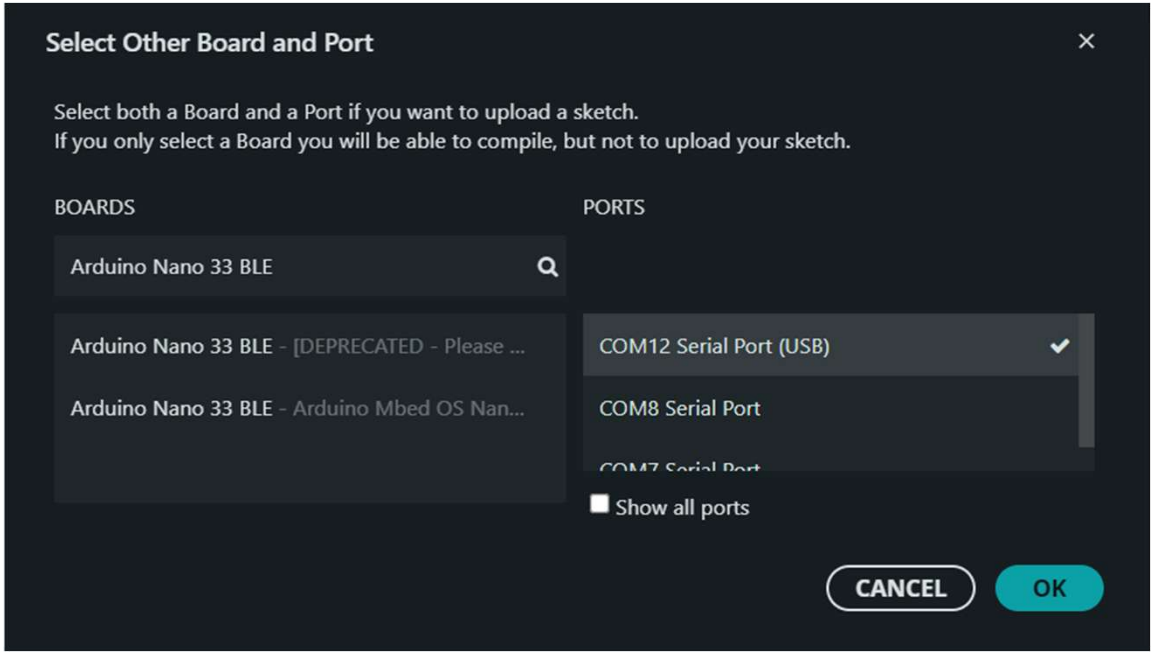

```
/*  Morse Code Flasher
    Arduino Nano 33 BLE Sense Rev 2
    - Type a word into the Serial Monitor
    - Press Enter
    - Arduino flashes the onboard LED (pin 13) in Morse Code
*/

const int ledPin = LED_BUILTIN;   // Onboard LED (D13)
String inputString = "";          // Stores user input
bool stringReady = false;         // Flag to indicate input completed

// Morse code dictionary
struct MorseMap { char letter; const char *code; };

MorseMap morseTable[] = {
  {'A', ".-"},   {'B', "-..."}, {'C', "-.-."}, {'D', "-.."},  {'E', "."},
  {'F', "..-."}, {'G', "--."},  {'H', "...."}, {'I', ".."},   {'J', ".---"},
  {'K', "-.-"},  {'L', ".-.."}, {'M', "--"},   {'N', "-."},   {'O', "---"},
  {'P', ".--."}, {'Q', "--.-"}, {'R', ".-."},  {'S', "..."},  {'T', "-"},
  {'U', "..-"},  {'V', "...-"}, {'W', ".--"},  {'X', "-..-"}, {'Y', "-.--"},
  {'Z', "--.."},
  {'0', "-----"},{'1', ".----"},{'2', "..---"},{'3', "...--"},{'4', "....-"},
  {'5', "....."},{'6', "-...."},{'7', "--..."},{'8', "---.."},{'9', "----."},
  {' ', " "}       // Space between words
};

const int morseCount = sizeof(morseTable) / sizeof(morseTable[0]);

// Morse timing (ms)
```

Morse Code Flasher: Partial Program

Lab: Unique Hello World Demonstrator. . .

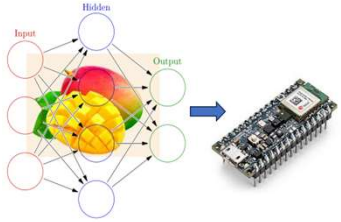


Arduino IDE version 2.3.6



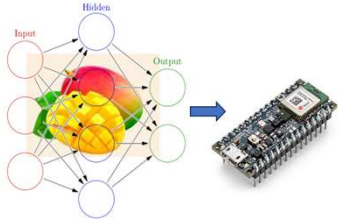
Selection of the Board and COM port

Lab: Unique Hello World Demonstrator. . .



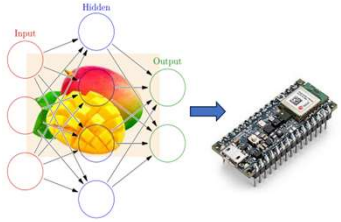
Measurement Setup

Lab: Unique Hello World Demonstrator. . .

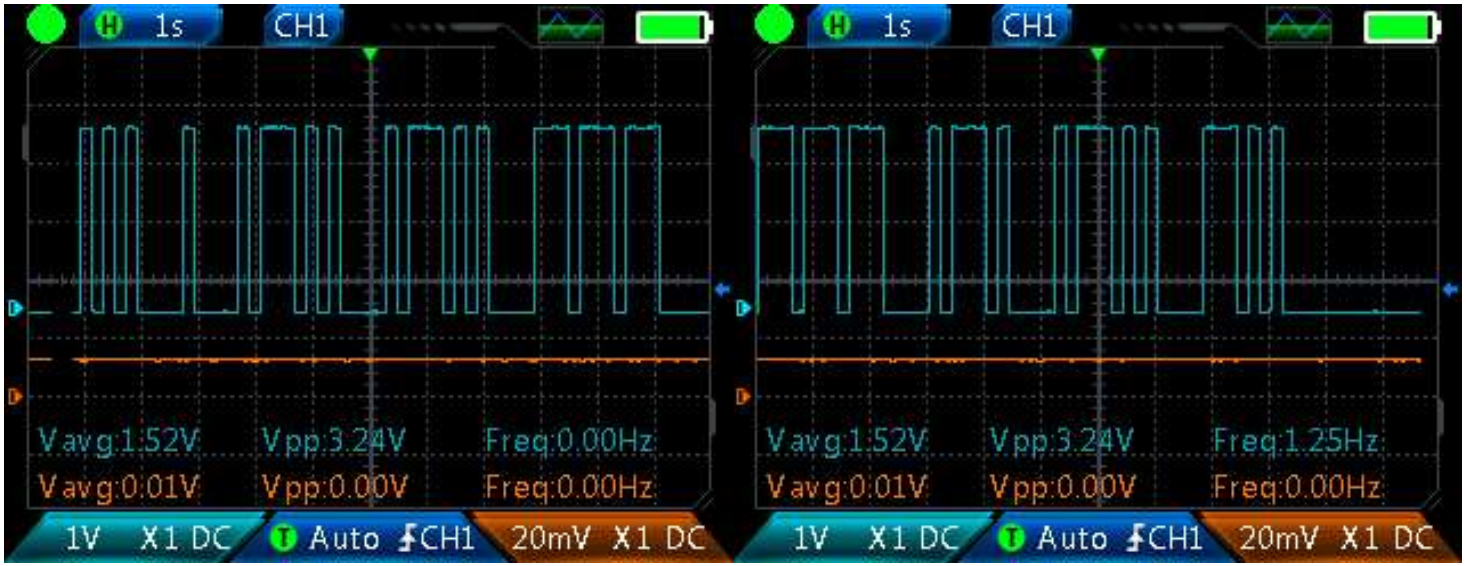


Tera-Term Software Serial Monitor

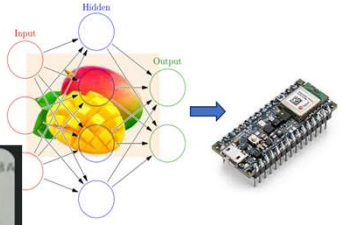
Lab: Unique Hello World Demonstrator. . .



Oscilloscope Binary Bits Capture: Hello World !!!

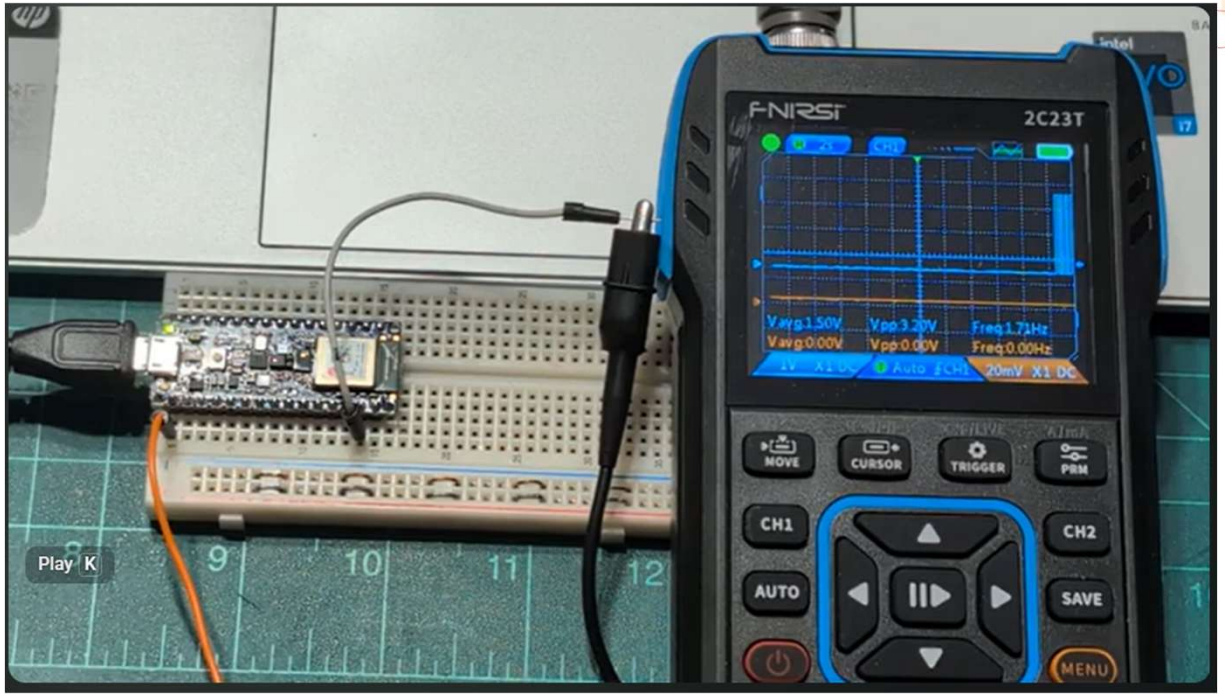


Lab: Unique Hello World Demonstrator. . .



Arduino Nano 33 BLE Sense Morse Code Flasher video
Click the link below:

https://youtu.be/nugevsAHvmU



Question 5

Which approach was used to establish the Morse Code symbols for transmitting a message shown on slide 36?

a) list
b) tuple
c) table
d) none of the above

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

CEC Continuing Education Center

DesignNews

Thank You

Sponsored by

DigiKey

informa markets