EE P 595A: TinyML
Lecture 1: Introduction to TinyML

Dept. of Electrical and Computer Engineering
University of Washington

Instructor: Dinuka Sahabandu (sdinuka@uw.edu)

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY of WASHINGTON

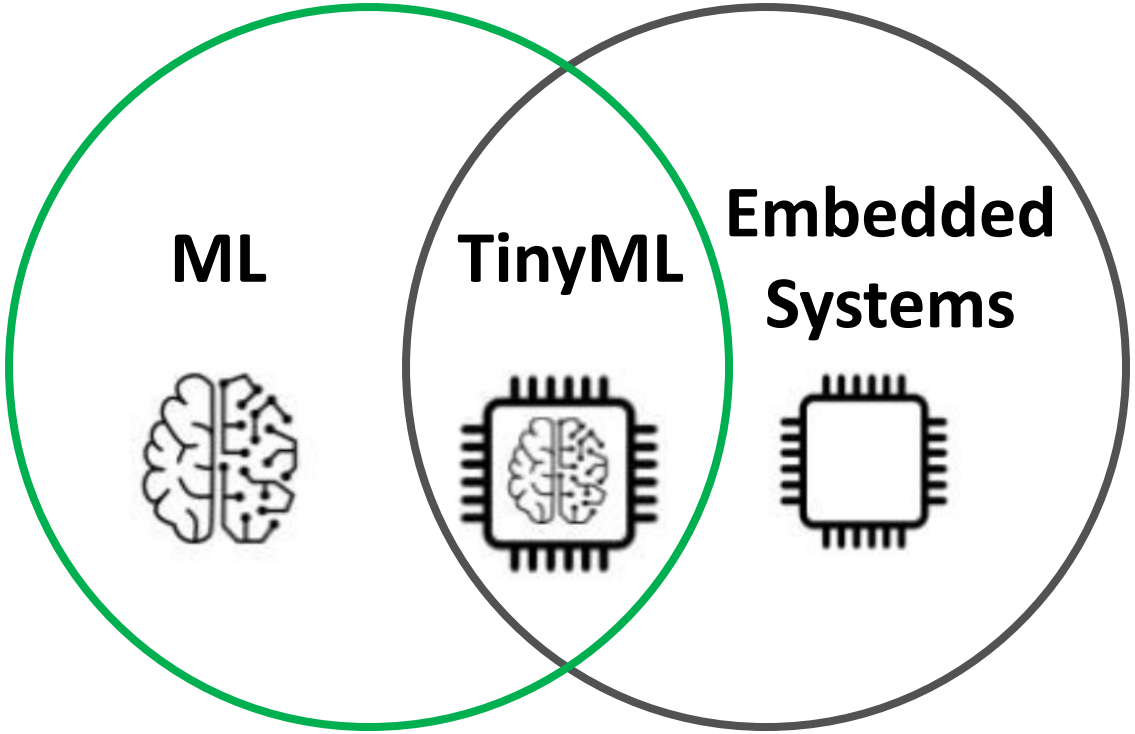
Topics Covered

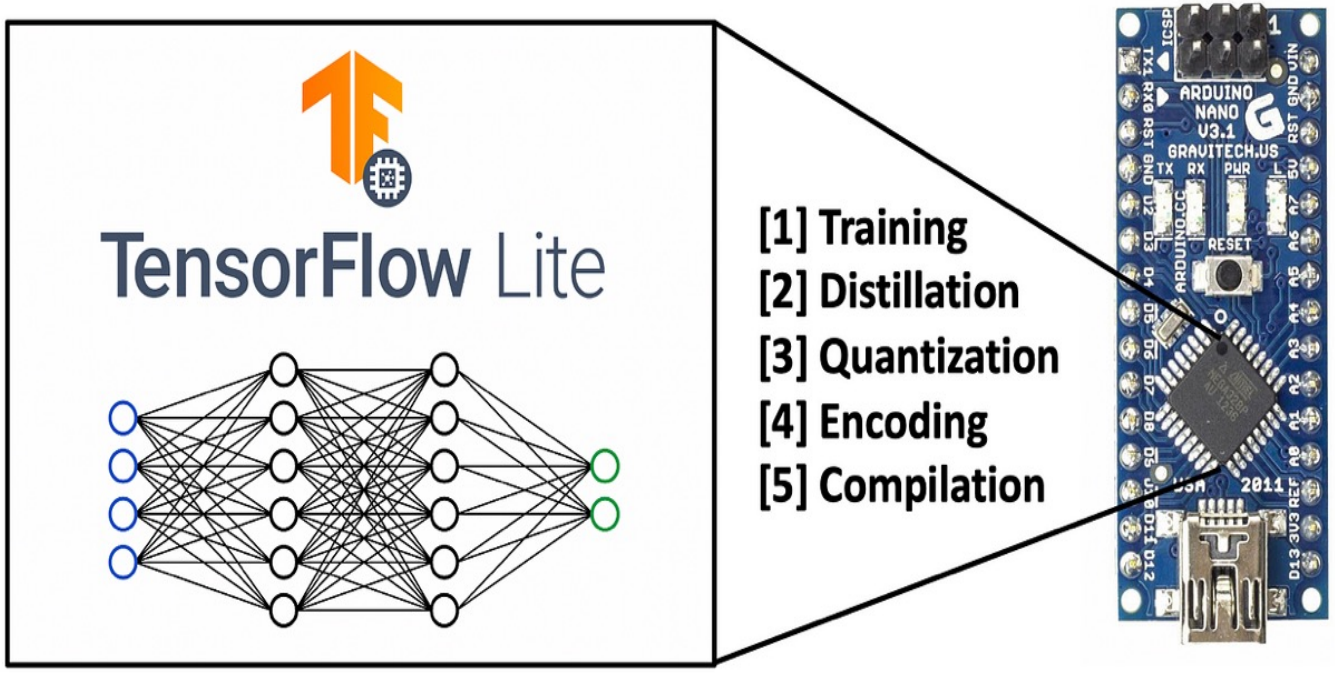
- Introduction to TinyML
- TinyML Landscape, Applications and Challenges
- Focus and Overview of PMP 595 TinyML Class
- Background Review on ML Algorithms and Neural Networks
- TinyML Lifecycle and Workflow
- Model Compression Techniques
- Lab 1: Introduction to Hardware and Software Used in the Course

❖Relevant reading from the Textbook [Warden and Situnayake; O'REILLY Publisher] is Chapter 1

Introduction to TinyML

- **TinyML:** Emerging area where **ultra large powerful ML models are converted into executables for embedded systems capable of** performing on-device sensor data analytics at extremely low power consumption, typically in the mW range and below that are battery operated

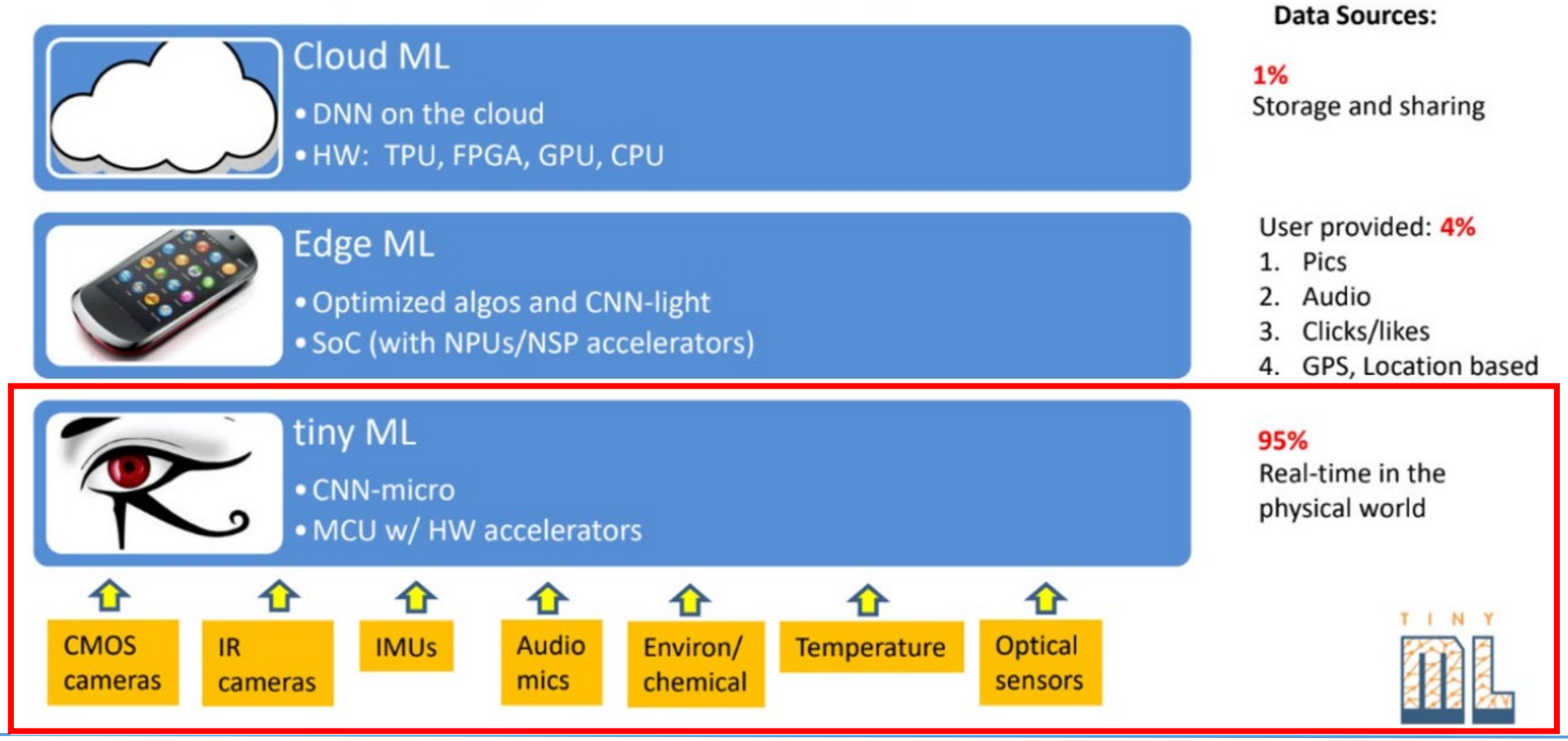




Source: https://towardsdatascience.com/tiny-machine-learning-the-next-ai-revolution-495c26463868

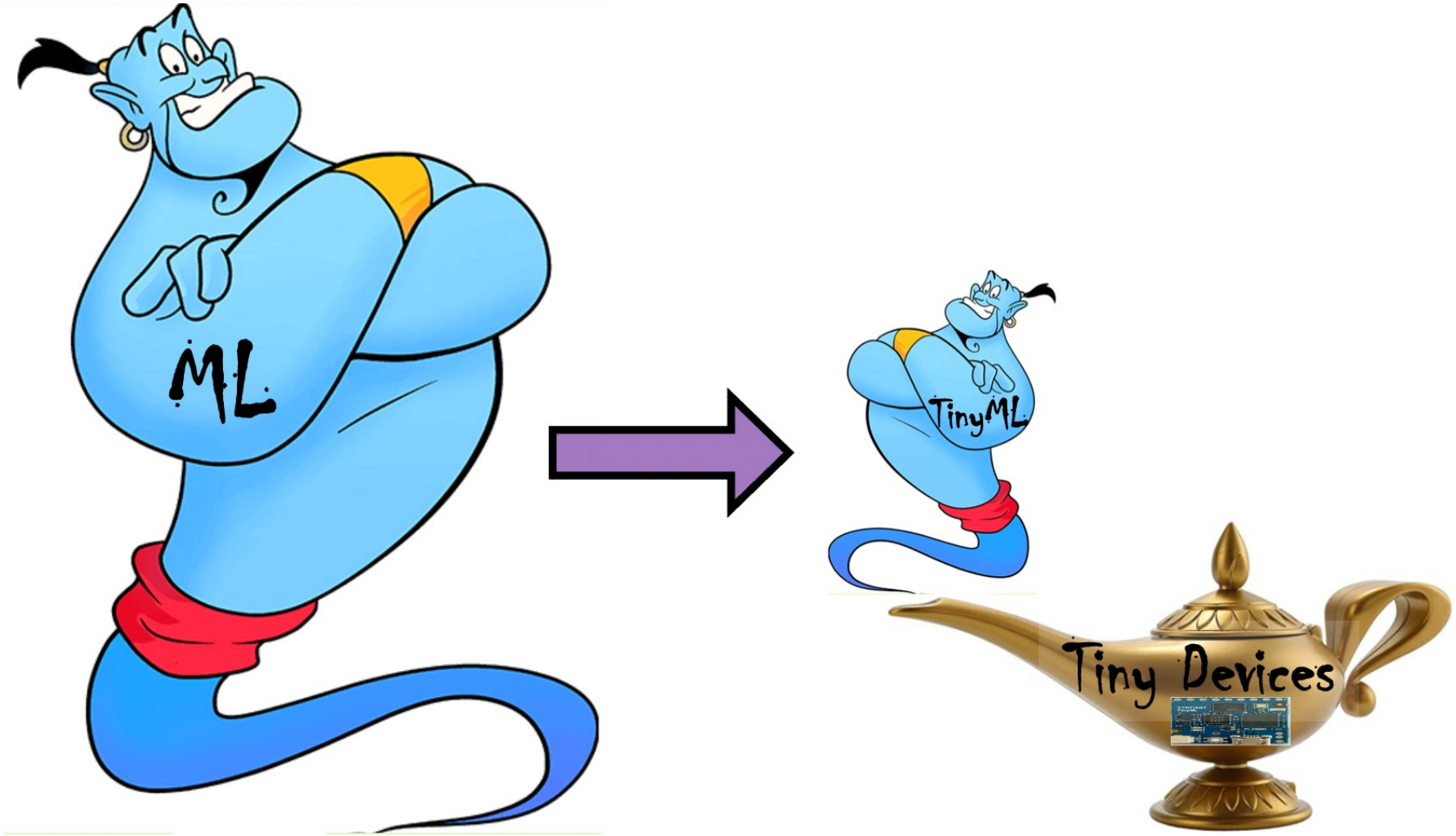
Introduction to TinyML

- TinyML is **real-time processing of time-series data that comes directly from sensors**



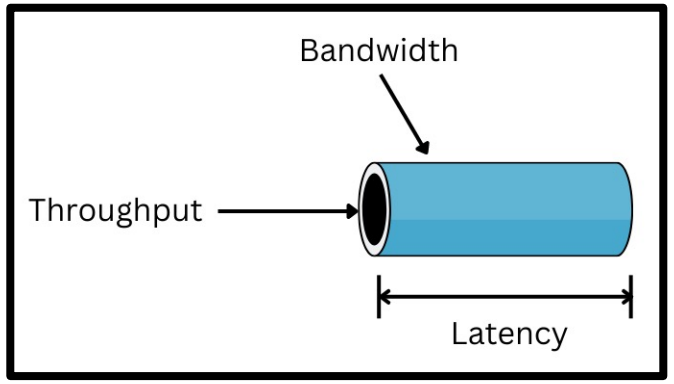
Source: https://www.tinyml.org/about/

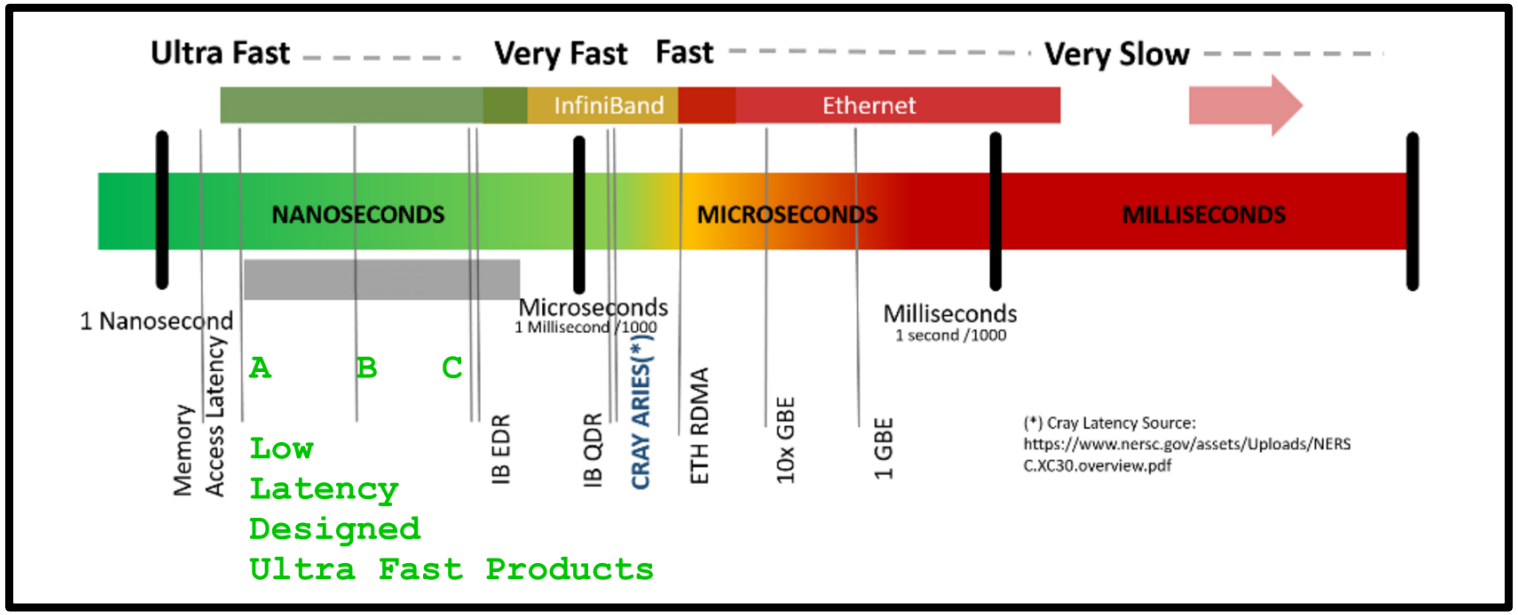
ML to TinyML



Benefits of TinyML

1. **Low Latency** - Optimized to have a reduced number of parameters.
2. **Privacy** - Enables on-device processing of data in real time.
3. **Low Power** - Optimized to run with fewer parameters and use reduced computations.
4. **Reduced Bandwidth** – Data is captured and processed on device.
5. **Independence from connectivity** –Data processing without the need for an internet connection.

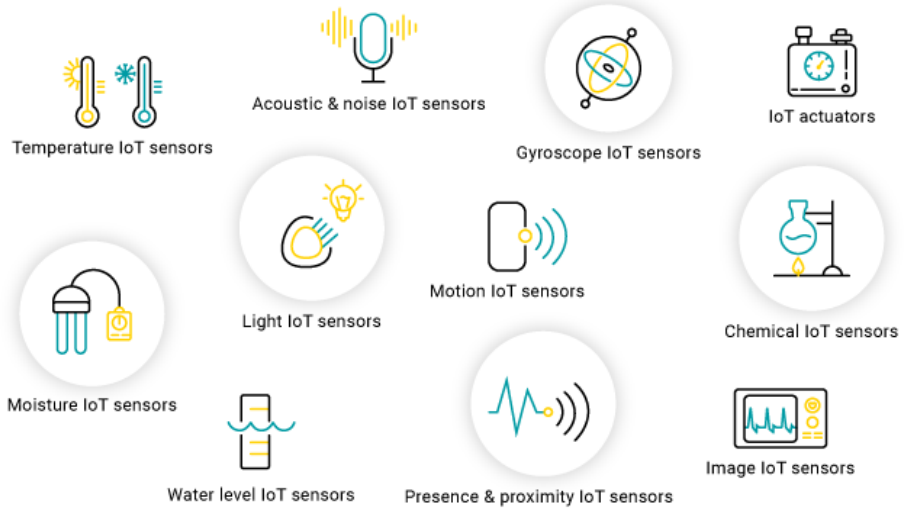




TinyML Landscape

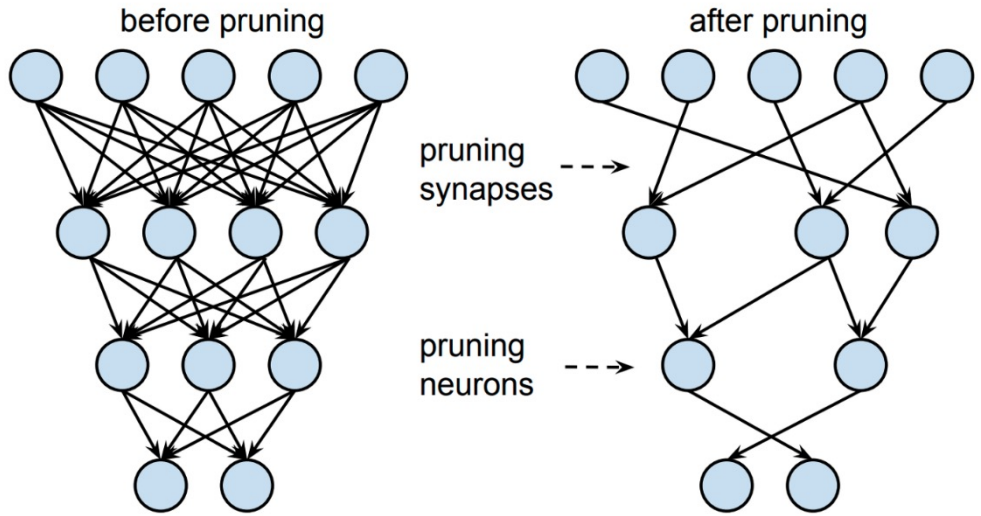
The TinyML ecosystem is fueled by:

1. Emerging commercial applications on **Artificial Intelligence of Things** (AIOT).
2. **Optimizing** algorithms, networks, and models down to 100 kB and below.
3. **Low-power** applications in vision and audio.



AIOT Heterogenous sensors

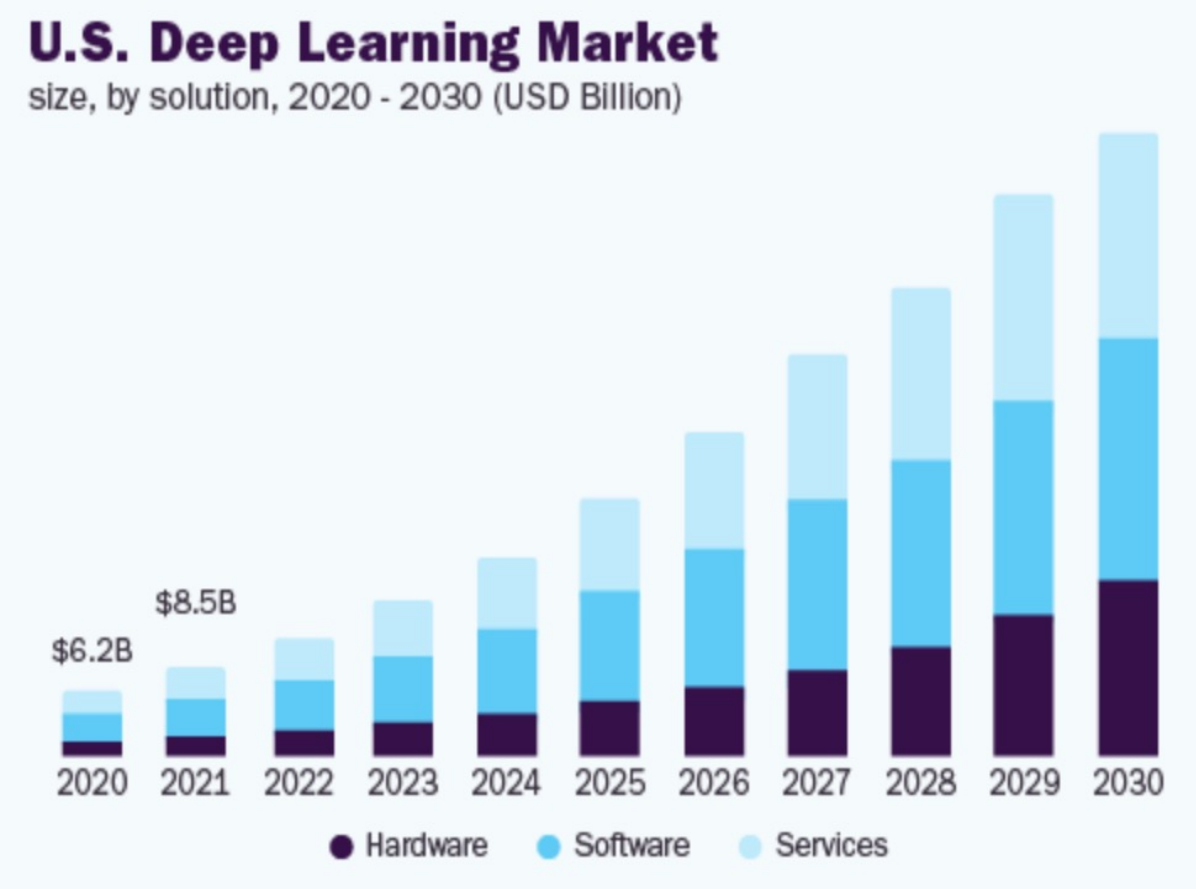
Source: https://www.avsystem.com/blog/what-is-resource-constrained-device/



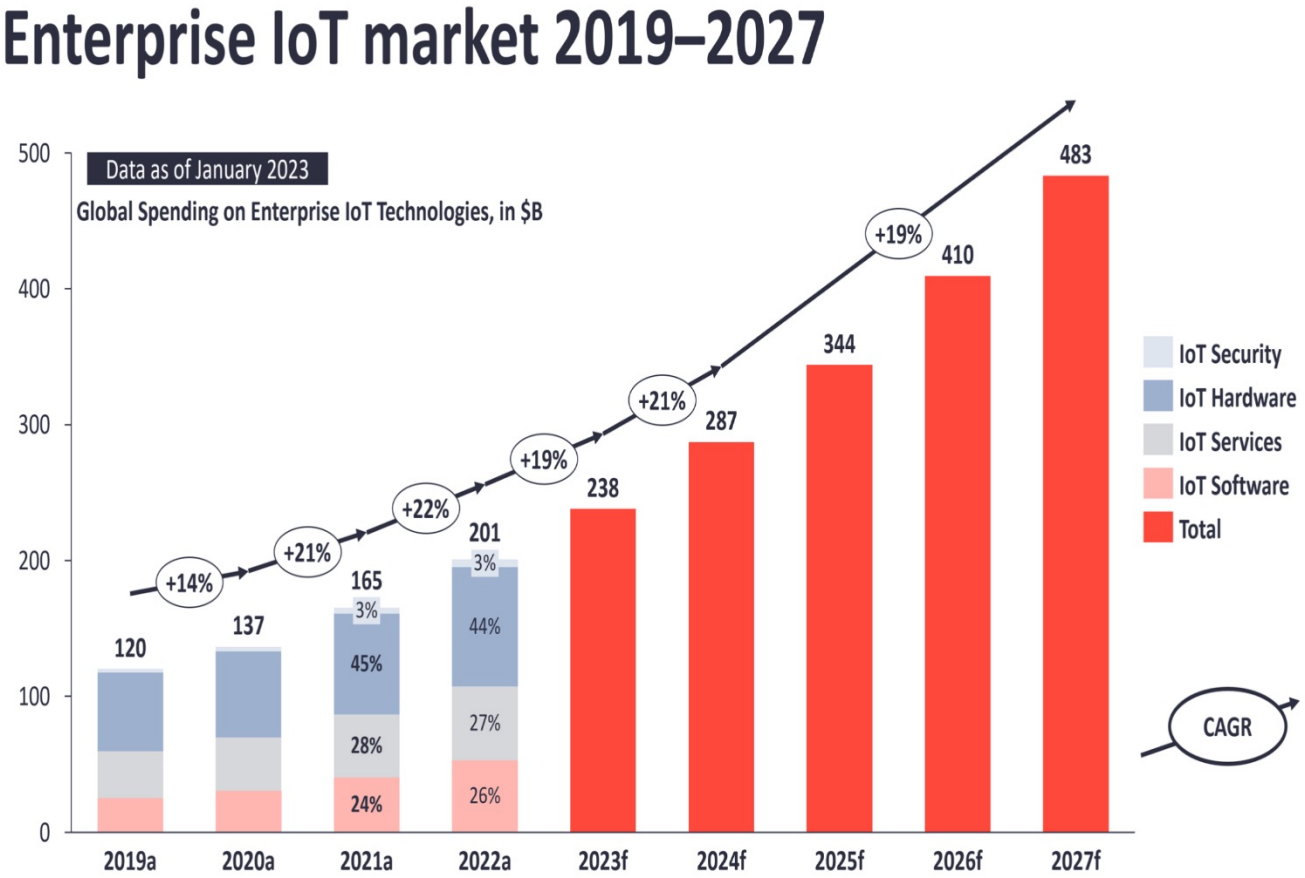
Model compression

Trends in TinyML

- Trend in ML and IoT (TinyML) market



Source: https://www.grandviewresearch.com/industry-analysis/deep-learning-market



Source: https://h9e3r9w2.rocketcdn.me/wp/wp-content/uploads/2023/02/IoT-market-size-2019-2027.png

TinyML Applications

- TinyML has **applications in agriculture, health, retail, energy industry**, and more…

Plant disease classification with TensorFlow Lite on Android

Source: https://yannicksergeobam.medium.com/plant-disease-classification-with-tensorflow-lite-on-android-part-2-c2d47371cea3

Solar Scare Mosquito: A solar-operated device that sits on stagnant water to create air bubbles at regular intervals to avoid the breeding of mosquitoes

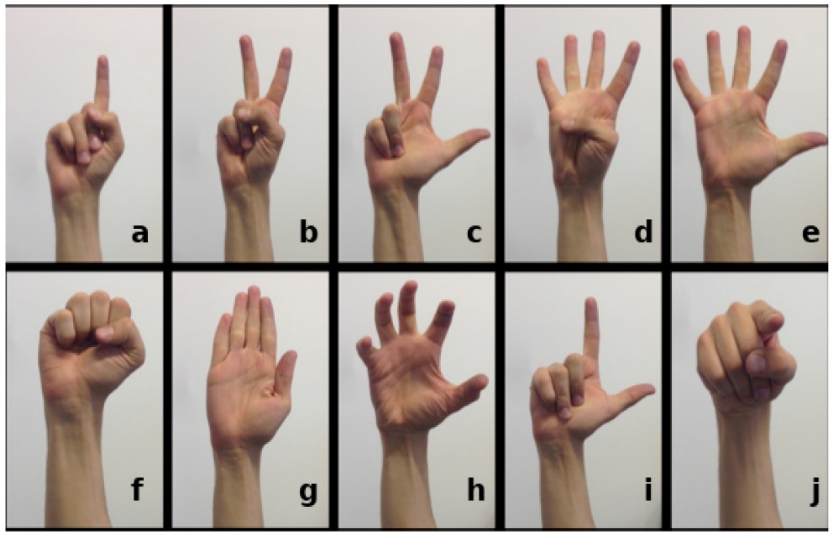
Source: https://theindexproject.org/award/nominees/6558

TinyML for keeping an eye on the inventory of goods on the shelf in retail establishments and sending out warnings when it runs low

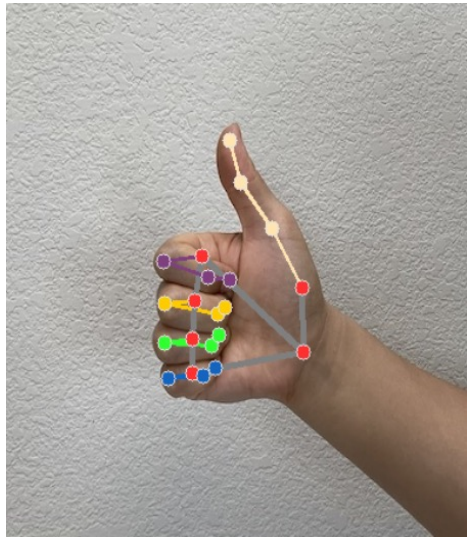
Source: https://www.supermarketnews.com/store-design-construction/amazon-go-goes-smaller

App#1– American Sign Language (ASL) Interpretation

- **What is the insight of this mapping from physical signing to English alphabets?**



One to One

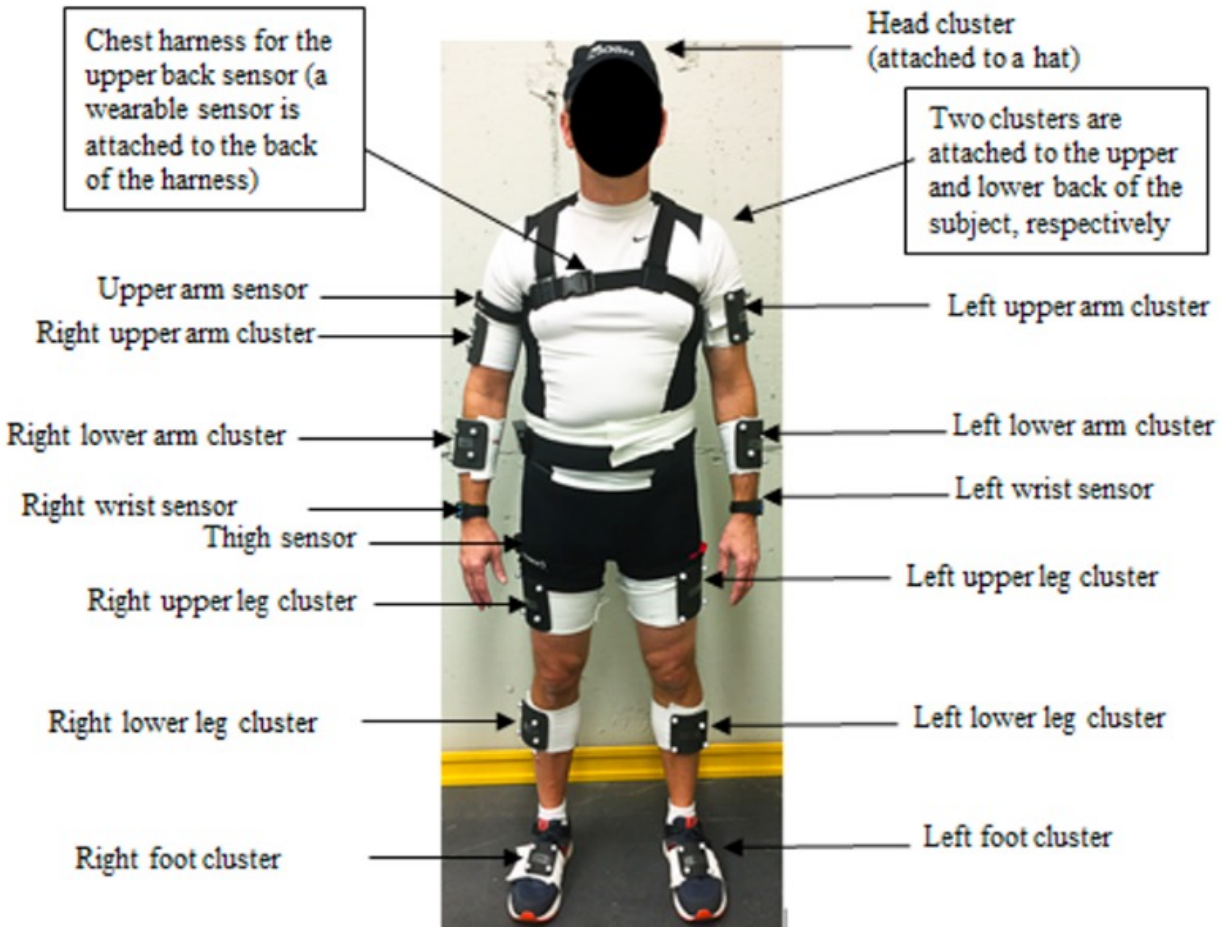


= “Which”

- **How can we map gestures to the machine-readable input?**
- **What is the mechanism for the mapping?**

App#1– American Sign Language (ASL) Interpretation

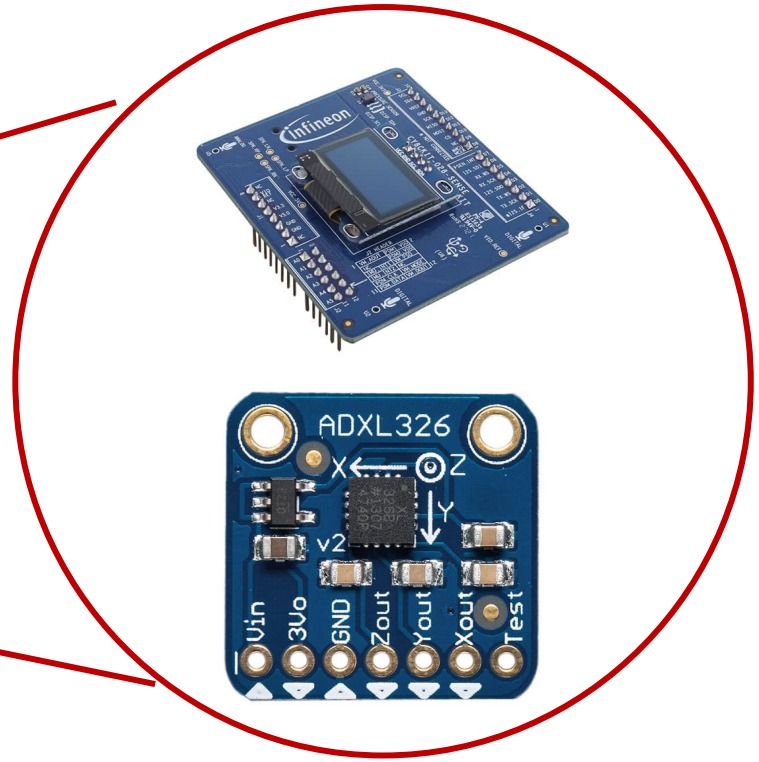
- A **gait sensor** is a device that measures and analyzes the movement of the foot and lower limbs during walking or running.
- Can we use gait sensors to capture motion for ASL?



App#1– American Sign Language (ASL) Interpretation

Problem: Develop Tiny hardware platform that will recognize ASL signs and convert to spoken words

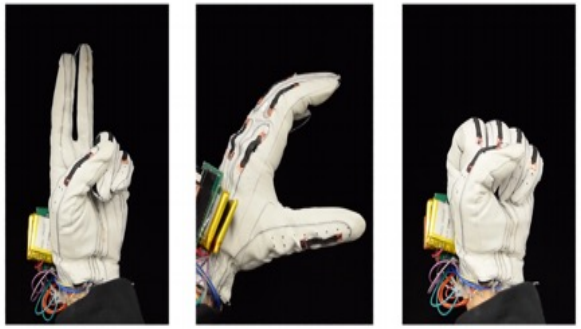
Wearable Glove Technology



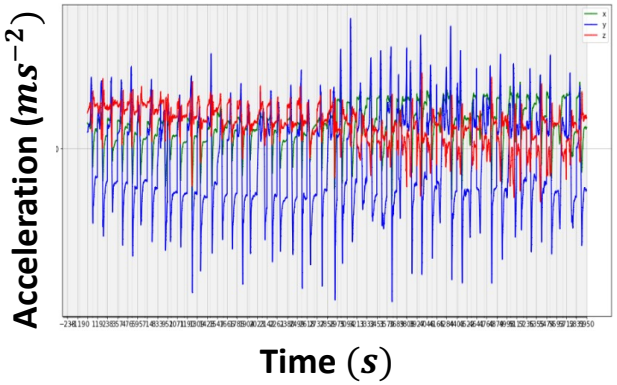
Accelerometer and Gyroscope sensors

App#1 – American Sign Language (ASL) Interpretation

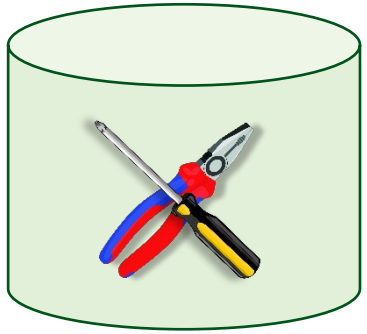
- **Solution**: Train a neural network to classify ASL gesture signals to their corresponding spoken words, compress the model, and embed it into edge device



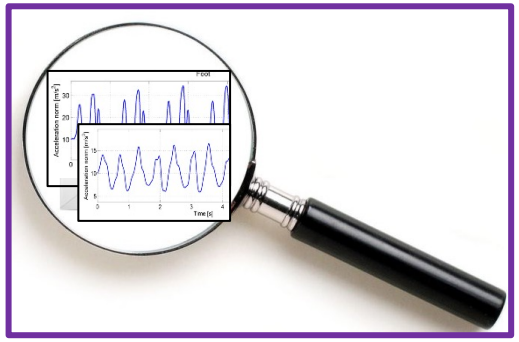
Output of sensors



Input Data to ML Model

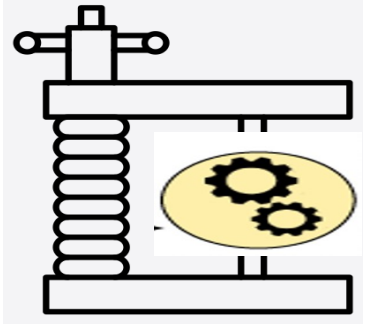


Pre-processing



Feature Extraction

ML Model



Model Compression

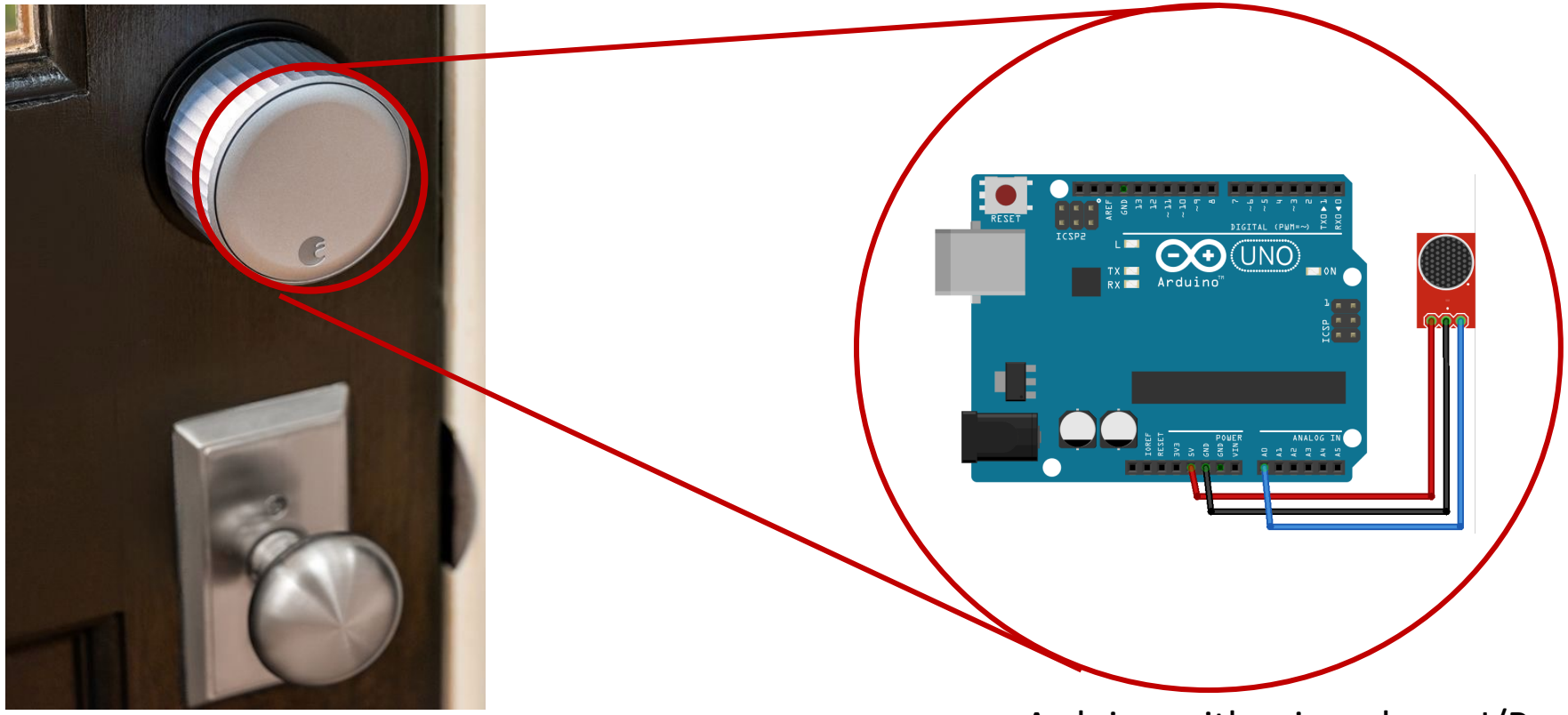
TinyML model Deployment on Edge Device

We	Why	When	With	While
0.2	0.6	0.05	0.05	0.1

TinyML model Output from Edge Device

App#2 - Smart Lock Audio Recognition

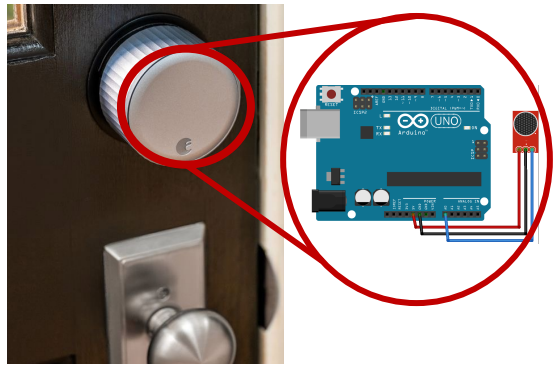
Problem: Build a tiny hardware platform that is aware of the ambient audio events around a door and detects intrusion



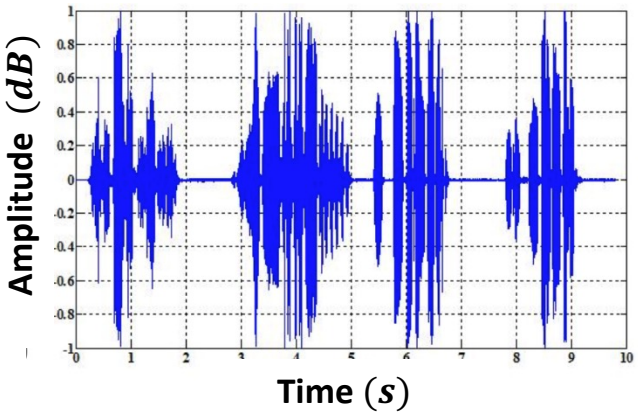
Arduino with microphone I/P

App#2 - Smart Lock Audio Recognition

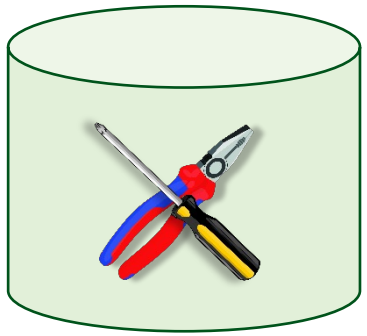
Solution: Perform classification on edge device by extracting and transforming the most significant audio features from the streaming data



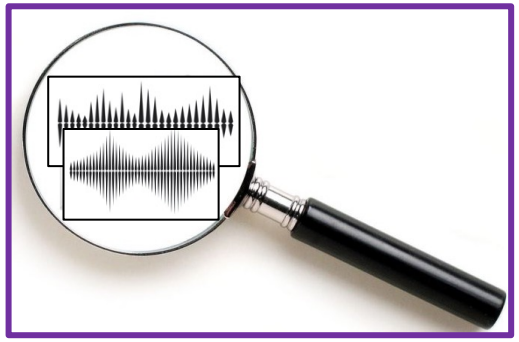
Output of sensors



Input Data to ML Model

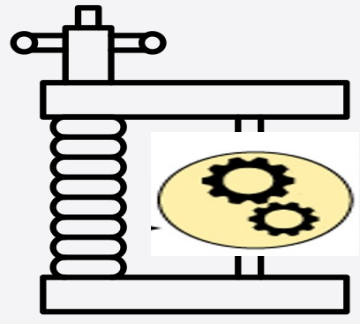


Pre-processing



Feature Extraction

ML Model



Model Compression

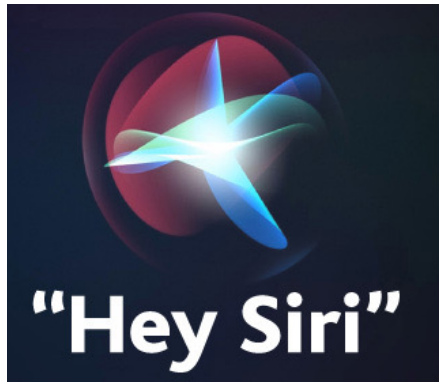
TinyML model Deployment on Edge Device

Locking	Knocking	Key I/O
0.23	0.62	0.15

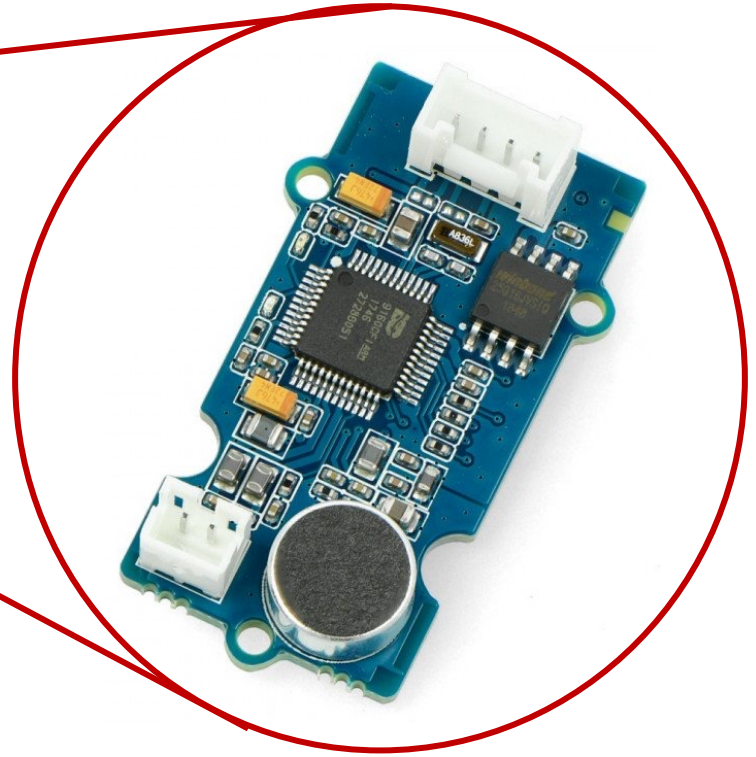
TinyML model Output from Edge Device

App#3 – Audio Wake Words

Problem: Build a Tiny hardware model which recognizes specific wake words and turns on to listen for further instructions



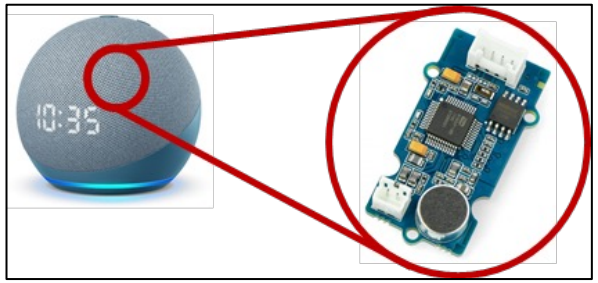
Device which responds to Audio Wake Word



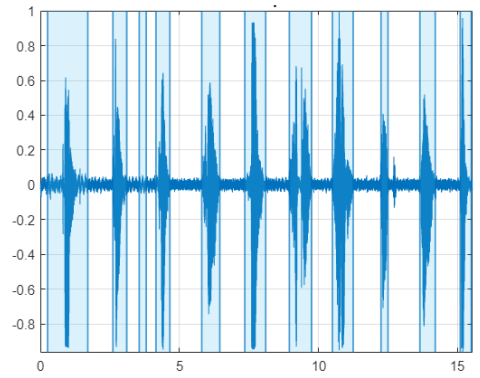
Microcontroller module with in-built microphone

App#3 – Audio Wake Words

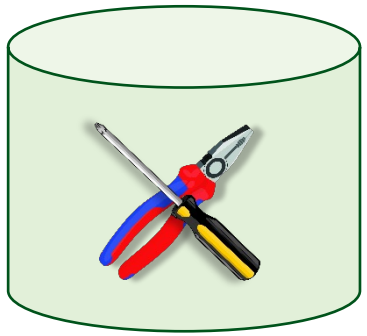
Solution: Perform classification on edge device by extracting and transforming the most significant audio features from the streaming data



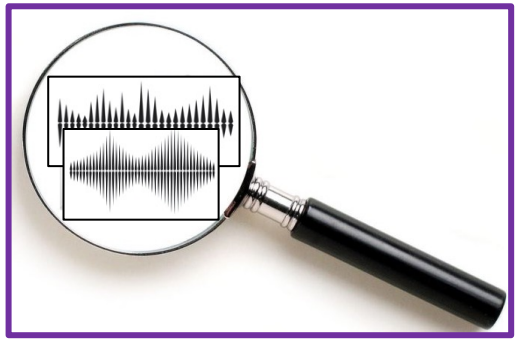
Output of sensors



Input Data to ML Model

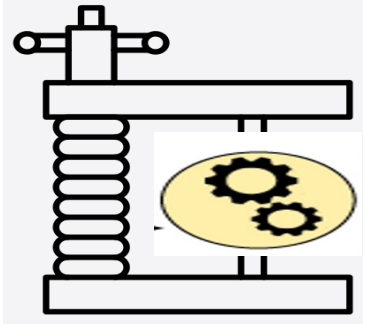


Pre-processing



Feature Extraction

ML Model



Model Compression

TinyML model Deployment on Edge Device

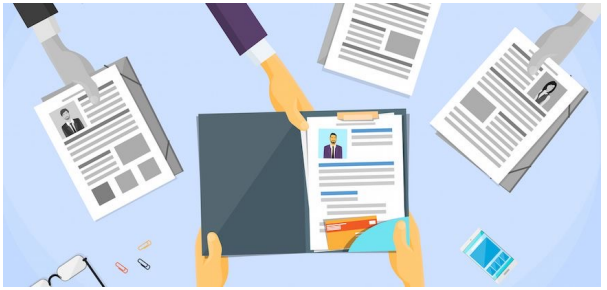
Speech I/P	Cat	Hello	Tree
Light Staus	OFF	ON	OFF

TinyML model Output from Edge Device

Applications

Society

- Human Activity Recognition Resume Screening
- Comment Classification
- Text Generation
- Intent Classification
- Neural Style Transfer
- Facial Emotion Classification
- Movie Review Sentiment Analysis



Finance

- Article Category Classification
- Document Denoising
- Financial Distress Prediction
- Churn Prediction for Bank Customers

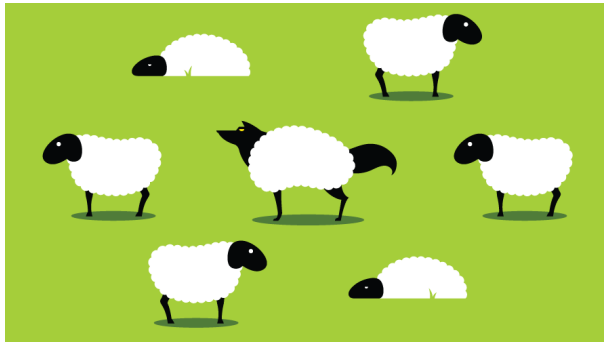
Applications

Security

- Credit Card Fraud Detection
- Malicious URL Detection
- Visual Wake Word Detection
- Encroachment Detection using anomalies
- Signature Forgery Detection
- Fingerprint Classification

Farming

- Arrhythmia Prediction using ECG
- Crop Weed Detection
- Abalone Age Prediction
- Fish Breed Classification
- Sheep Detection
- Flower Classification
- Tomato Disease Detection

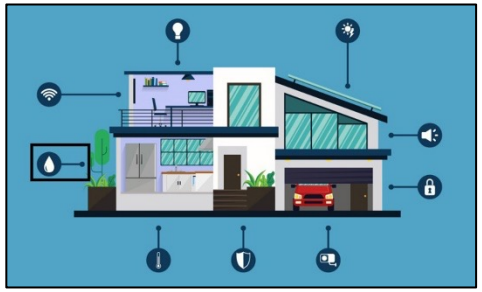


Applications

Oil and Gas

- Pump Failure Detection
- Predictive Maintenance
- Equipment Failure Detection

Home



- Speech Emotion Recognition
- Gender Recognition
- Song Genre Prediction
- Auto image Captioning
- Language Identification
- Glass Quality Assessment
- Fruits Classification
- Hit Song Prediction
- Fall Detection

Applications

Health

- Detecting Emotions
- Brain Tumor Detection
- Fetal Health Classification
- Diabetes Prediction
- Face Mask Detection
- Drug Classification
- Gesture Recognition with Muscle Activity
- Breast Cancer Detection
- Blood Cell Classification
- Parkinson's Disease Detection
- Heart Disease Prediction
- Alzheimer Detection …

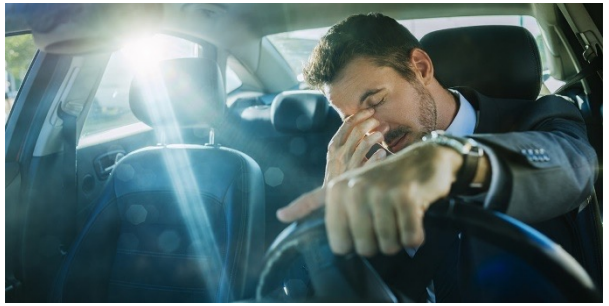
Applications

City

- Hate Speech and Offensive Language Detection
- Improving Image Resolution with Autoencoders
- Neural Machine Translator
- Detecting fires in road surveillance
- Disaster Identification
- Pedestrian Detection

Transportation

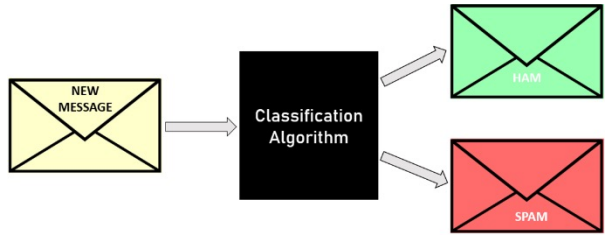
- Surface Crack Detection
- 3D Shape Detection
- Driver Drowsiness Detection
- Road Crack Detection
- Object Classification



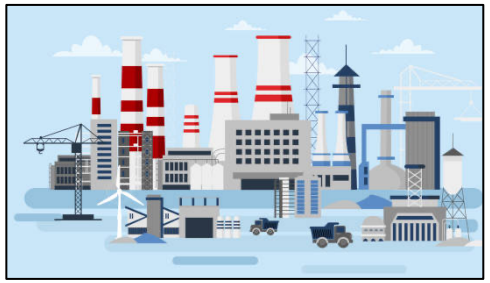
Applications

Retail

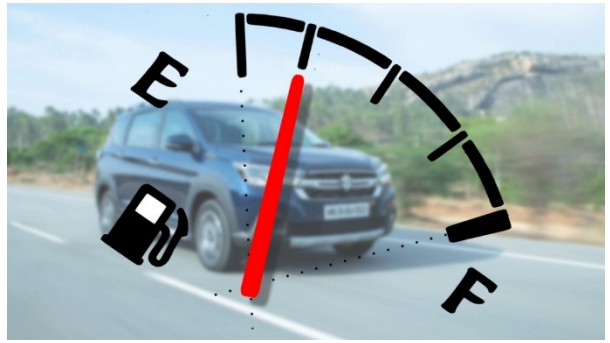
- Spam Text Classification
- Currency Notes Classification
- Clothes item Classification
- Marble Defect Classification
- Glass Classification
- Mobile Price Range Classification
- Gemstone Classification



Energy

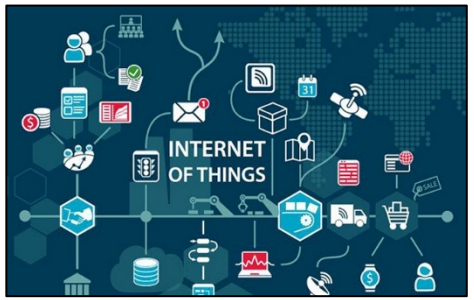


- Energy Consumption Prediction
- Fuel Consumption Prediction
- Fuel Efficiency Prediction

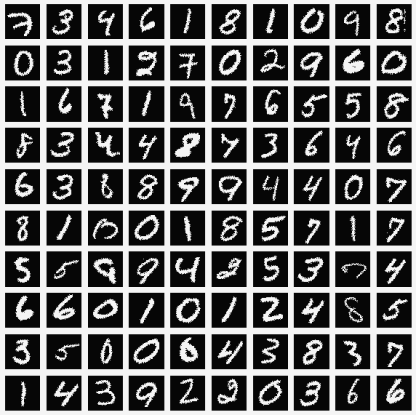


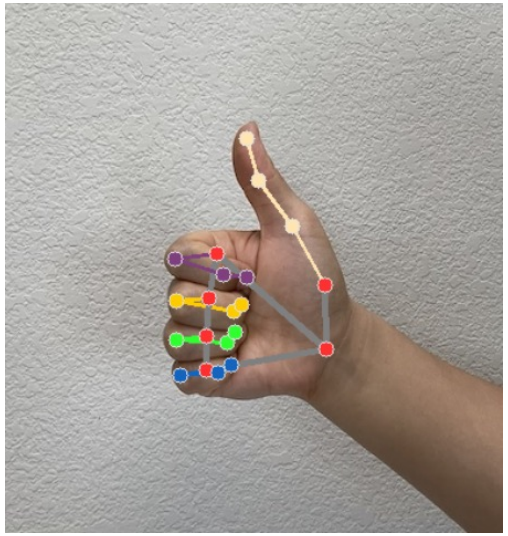
Applications

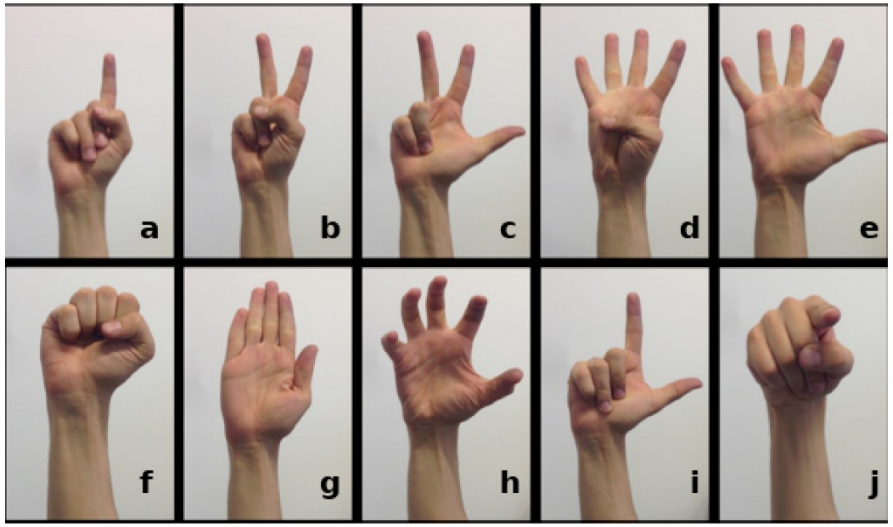
IoT



- PyTorch MNIST Vision App
- ASL Recognition
- CIFAR-10
- IoT Based Gesture Detection
- Hand Sign Recognition
- Shape Image Classification







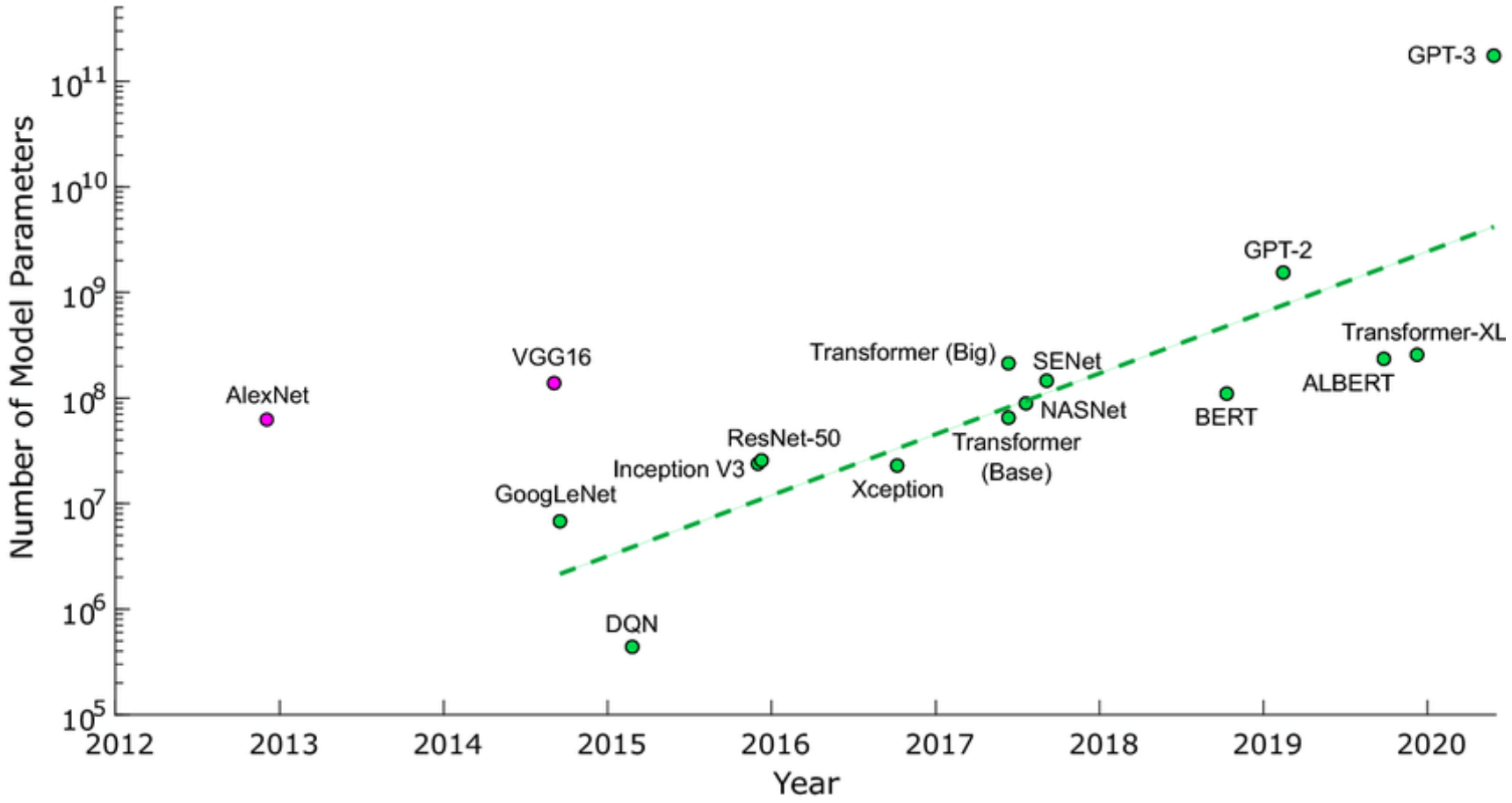
Hardware Challenges in TinyML

1. **Memory and I/O** operations are internal to the MC
2. **Low frequency** of operation can delay the time for job completion
3. **Power consumption** is much less for MCs

	Embedded Device	Tiny Device
Compute 	1 GHz – 4 GHz	**1 MHz – 400 MHz**
Memory 	512 MB – 64 GB	**2 KB – 512 KB**
Storage 	64 GB – 4 TB	**32 KB – 2 MB**
Power 	30 W – 100 W	**150 µW – 23.5 mW**

Software Challenges in TinyML

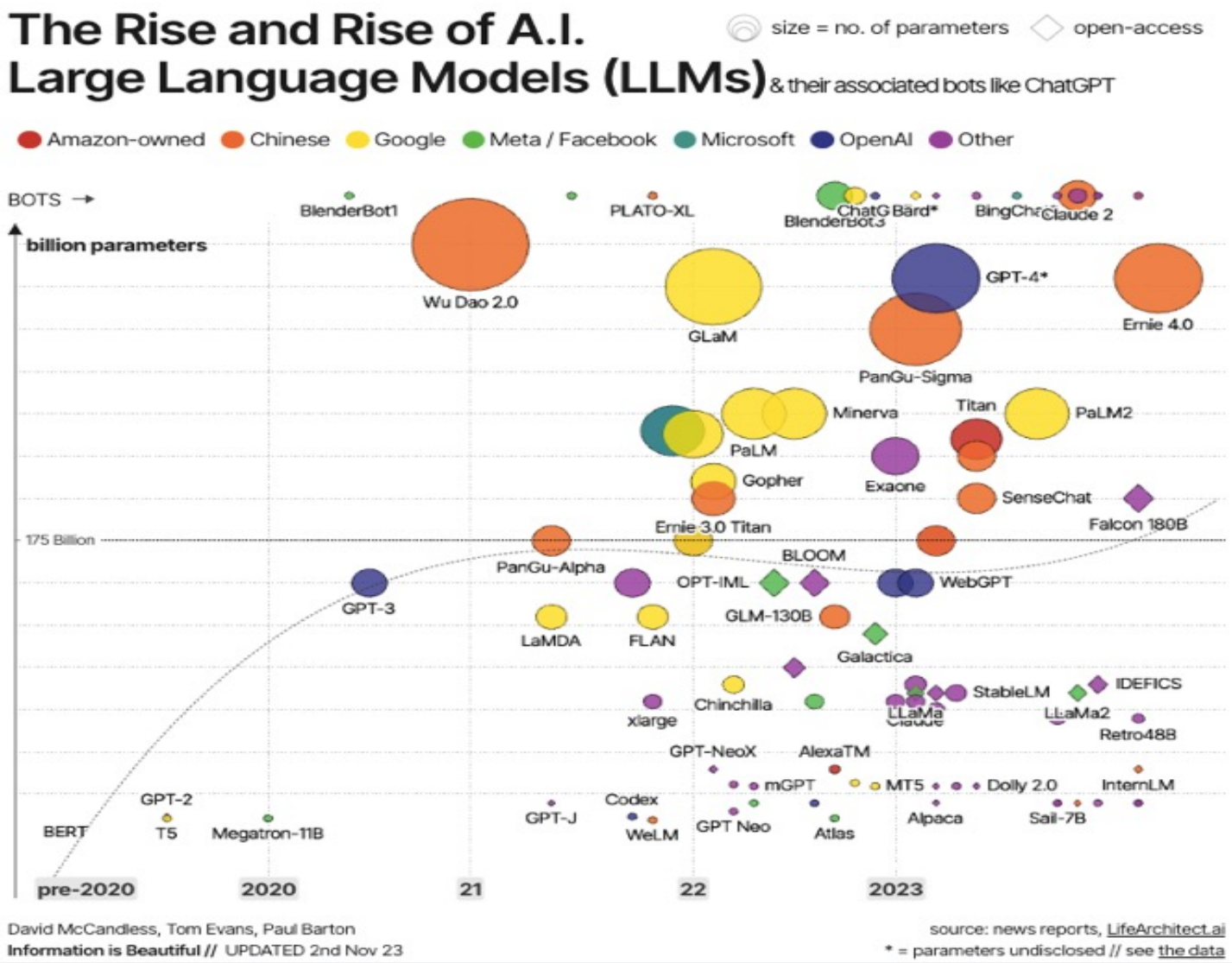
- Current ML models have **trillions of parameters** which is not viable for implementation on **tiny devices**



ML Model Size Growth Trends

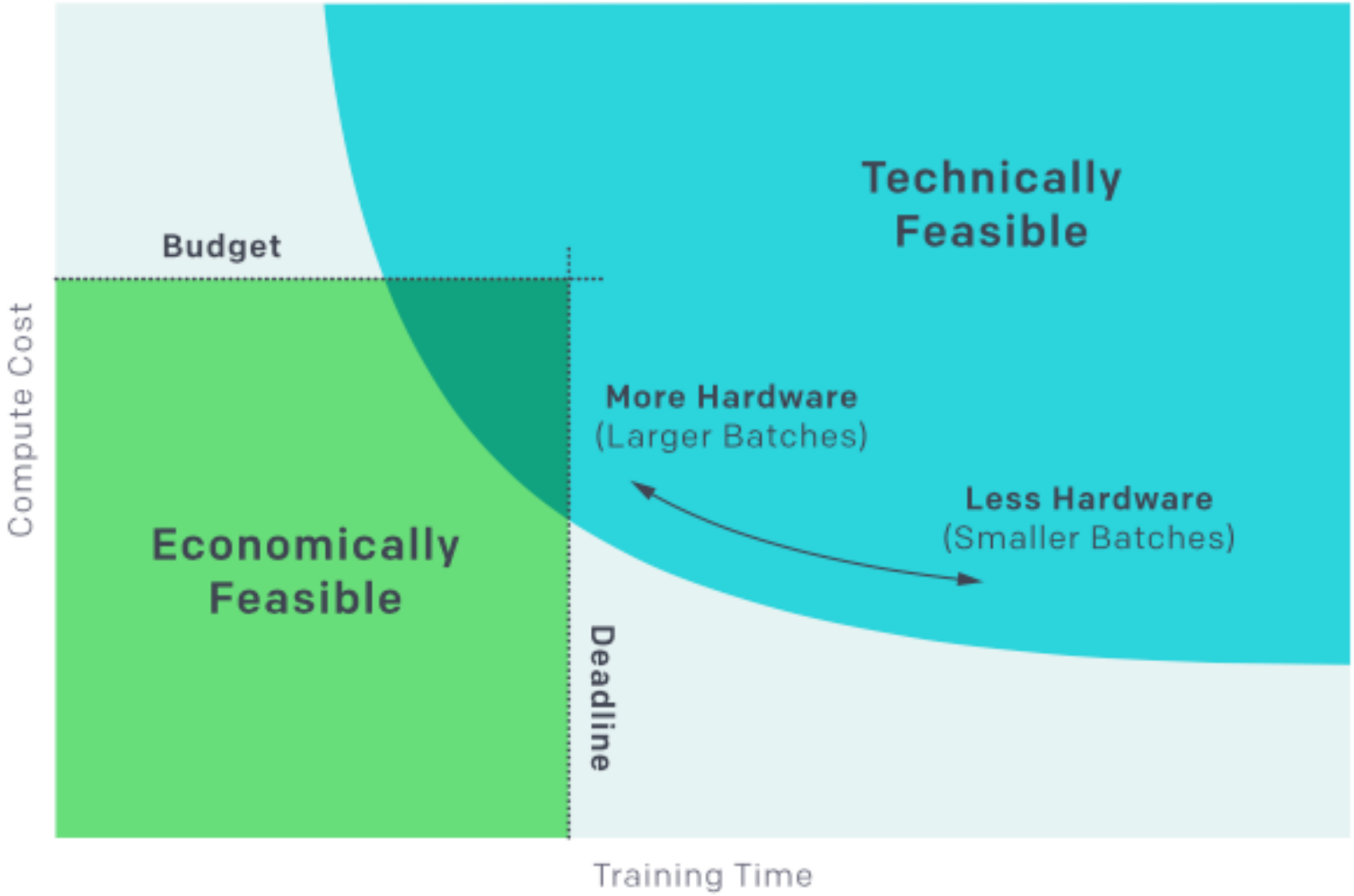
Source: https://openai.com

Software Challenges in TinyML



https://www.datacamp.com/blog/what-is-an-llm-a-guide-on-large-language-models

Software Challenges in TinyML

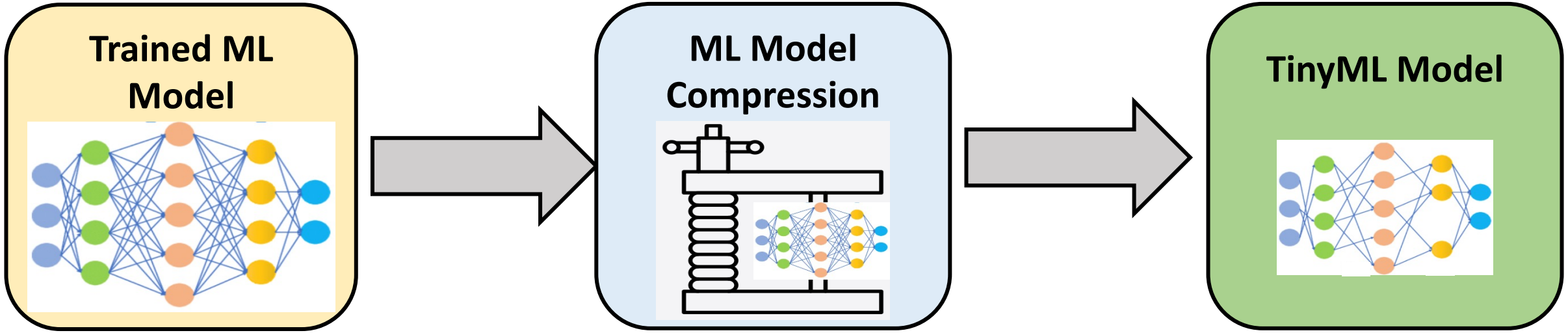


Source: https://openai.com

Solution to Software and Hardware Challenges in TinyM

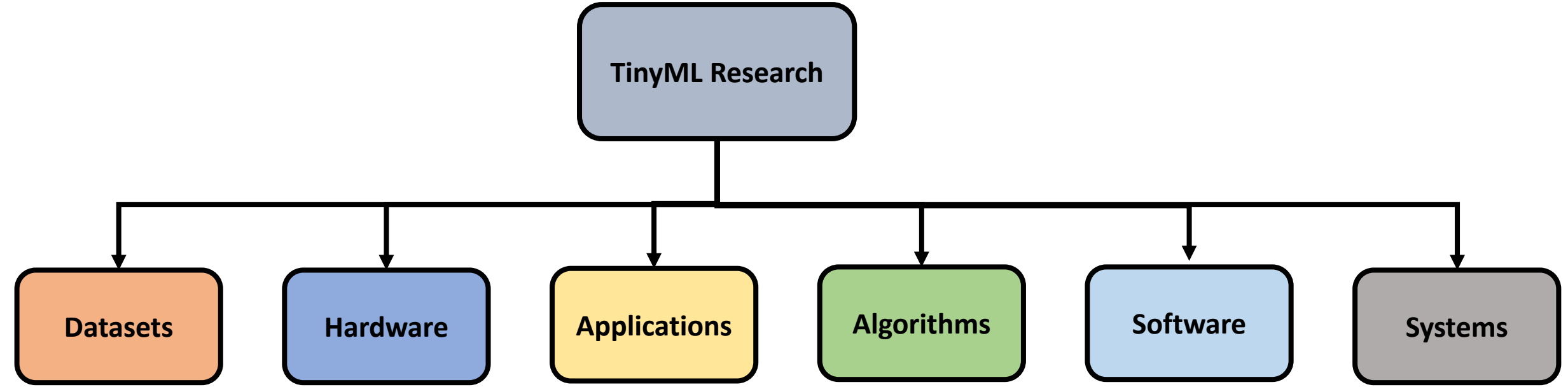
- There is a need to **develop highly compressed models while preserving accuracy** to be implemented on TinyML devices.

What is the approach that we can take?



Model Compression Workflow

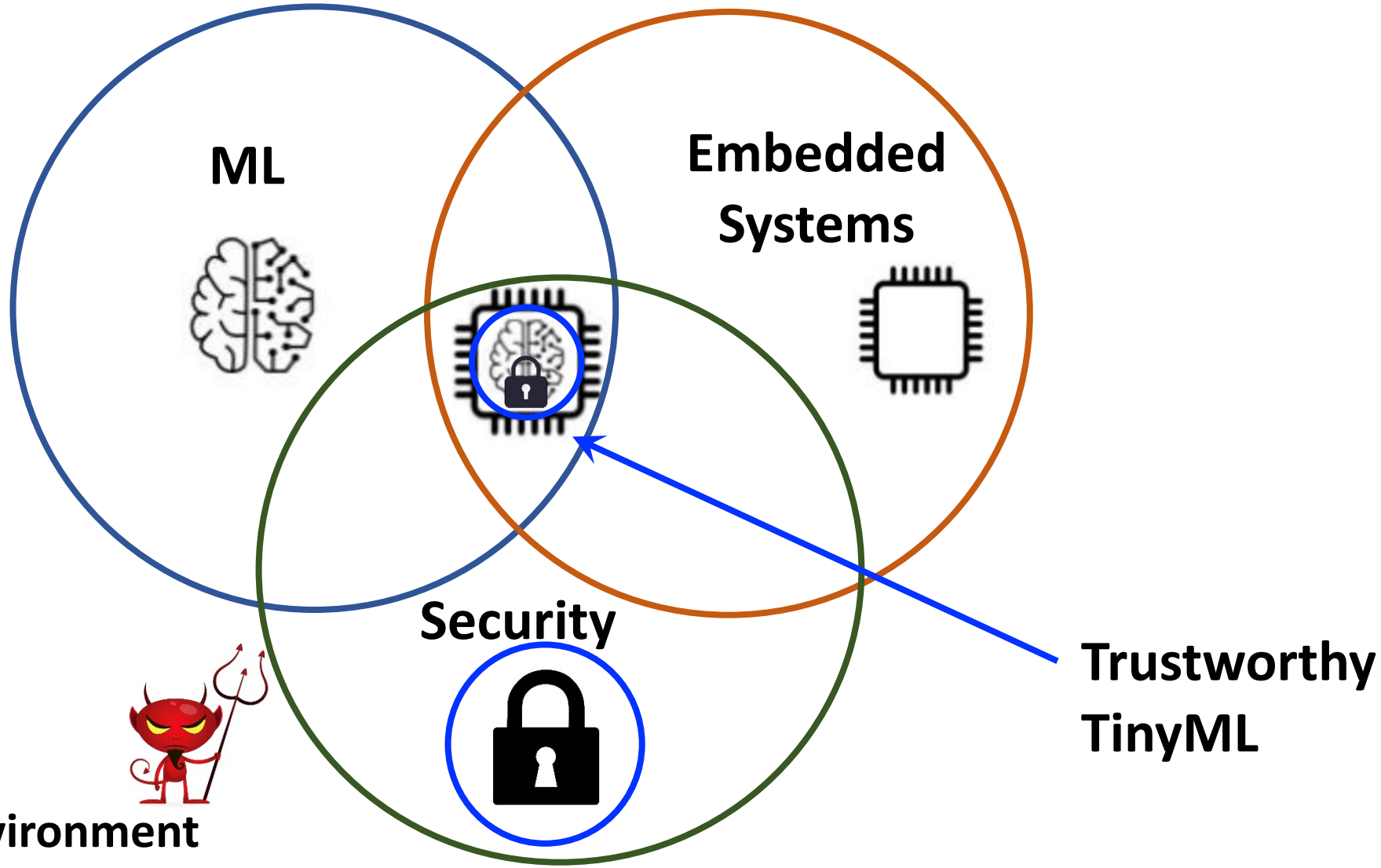
Research areas in TinyML



Datasets	Hardware	Applications	Algorithms	Software	Systems
• Public release of new datasets • Frameworks to automate dataset development • Survey and analysis of the quality of existing tiny datasets	• MCU and accelerator architecture design • Circuit and architecture design • Ultra-low-power memory system design	• Novel applications for emerging use cases • User behavior and system-user interaction	• Federated learning or stream-based active learning methods • Deep learning and traditional machine learning algorithms • Security and privacy implications.	• Interpreters and code generator frameworks • Optimizations for efficient execution • Neural architecture methods	• Profiling tools for measuring and characterizing performance and power • Hardware and software co-design

Source: https://www.tinyml.org/event/research-symposium-2023

Focus of this Course



5 min Break

Overview of TinyML Class

- **One of the first courses to bring ML, embedded systems, and IoT together**
- Through **Weeks 1-9, in each class**: (**Foundations--1&2 weeks)**
  - First Half: Discussion & Technical Background
  - Second Half: Lab (**Application development from week 3**)
- **Week 10: Possibly lecture or additional time for working in the project**
- **Week 11:** Final Project Presentations of the teams; Final project report submission
- All labs use **Python and C for coding**, and **we will provide needed modules** and also **work with the students during the labs**

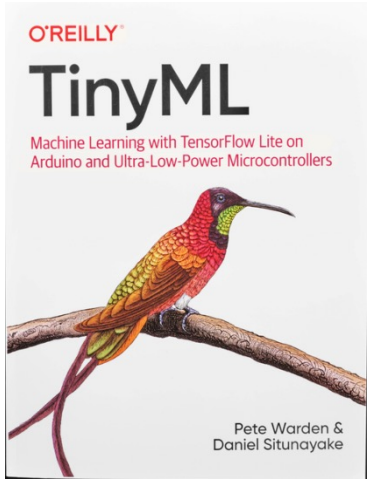
Overview of TinyML Class

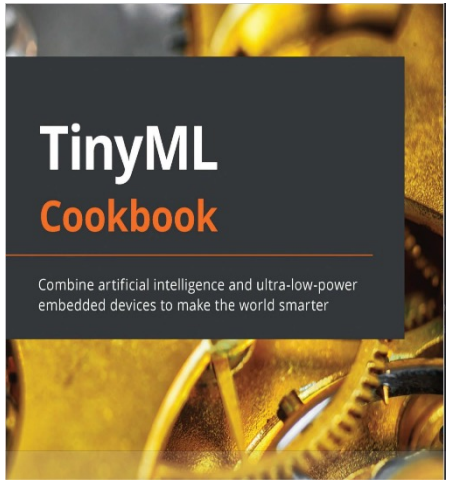
- What you will learn the following **How To's**:
  - ✓ **Develop TinyML models to solve real-word problems**
  - ✓ **Implement TinyML applications & explanation of needed ML algorithms**
  - ✓ **Use Python libraries** - NumPy, Pandas, Seaborn, and Scikit-learn
  - ✓ **Use TensorFlow for deep learning** and **TensorFlow Lite (TFLite) for TinyML**
  - ✓ **Using C language for deploying TinyML on Embedded Systems**
  - ✓ **Measure the performance** of the deployed TinyML models

- **Course Grade will be based upon 45% homework; 10% participation and quizzes; and a 45% final project)**

Overview of TinyML Class: Resources

- **Textbooks**:
  - **TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers** 1st Edition by Pete Warden and Daniel Situnayake [O'REILLY Publisher]
  - **TinyML Cookbook** by Gian Marco Lodice





- **Free Online Material**:
  - **TinyML Foundation**: https://www.tinyml.org
  - **Cainvas Platform**: https://cainvas.ai-tech.systems/gallery/

- **Software and Hardware**:
  - **TensorFlow Lite**: https://www.tensorflow.org/lite
  - **Tiny Machine Learning Kit Arduino (~$60)**

Course Summary: Topics Covered

- **Week 1: Introduction to TinyML**
  - TinyML Landscape, Applications and Challenges
  - TinyML Lifecycle and Workflow
  - Model Compression Techniques
  - Recap on Necessary ML Background: ML Algorithms, Neural Networks
  - Introduction to Hardware and Software Used in the Course

- **Week 2: Fundamentals of ML and TinyML**
  - Pruning ML models
  - Quantization Aware Training (QAT) and Post Training Quantization (PTQ)
  - Knowledge Distillation
  - Tiny Deep Learning
  - TensorFlow Lite (TFLite) for TinyML

Course Summary: Topics Covered

- **Week 3: TinyML for Keyword Spotting**
  - Background on Keyword Spotting and Streaming Audio
  - Challenges and Constraints in Keyword Spotting
  - Keyword Spotting Architecture and Data Collection
  - Model Training, Evaluation Metrics, and Deployment

- **Week 4: TinyML for Visual Wake Words**
  - Introduction to Visual Wake Words and Its Challenges
  - Visual Wake Words Dataset
  - MobileNets
  - Transfer Learning for Visual Wake Words
  - Model Training, Evaluation Metrics, and Deployment

Course Summary: Topics Covered

- **Week 5: TinyML for Anomaly Detection**
  - Background on Anomaly Detection and Signal Processing
  - Real and Synthetic Datasets
  - Unsupervised Learning (K-Means Clustering and Autoencorders)
  - Threshold Choice
  - Model Training, Evaluation Metrics, and Deployment

- **Week 6: Wizard Magic Wand**
  - Gesture Tracking through Bluetooth
  - CNN for Magic Wand Sketch
  - Data Collection and Labeling
  - Model Training, Evaluation Metrics, and Deployment

Course Summary: Topics Covered

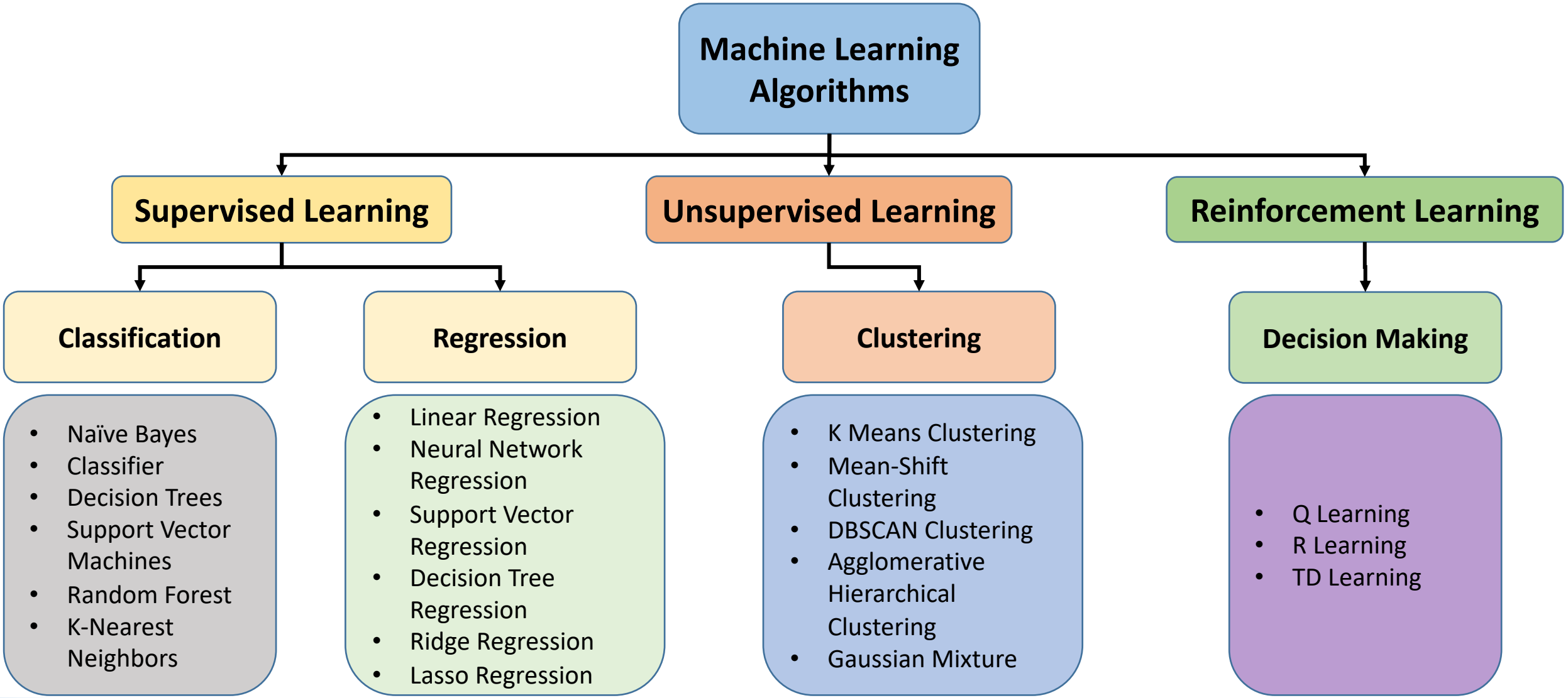
- **Week 7: TinyML for Predictive Maintenance**
  - Background on Predictive Maintenance Solutions and Industry Applications
  - Sensors, Sensor Data, and Interface
  - Accelerometer, Gyroscope, Barometer, and Magnetometer
  - TinyML Framework for Predictive Maintenance
  - Model Training, Evaluation Metrics, and Deployment

- **Week 8: TinyML for American Sign Language (ASL) Interpretation**
  - Background on ASL and ASL Interpretation
  - Gesture Motion Datasets and Features
  - Analyzing Gesture Motion Data using Neural Networks
  - TinyML Framework for ASL Interpretation
  - Model Training, Evaluation Metrics, and Deployment

Course Summary: Topics Covered

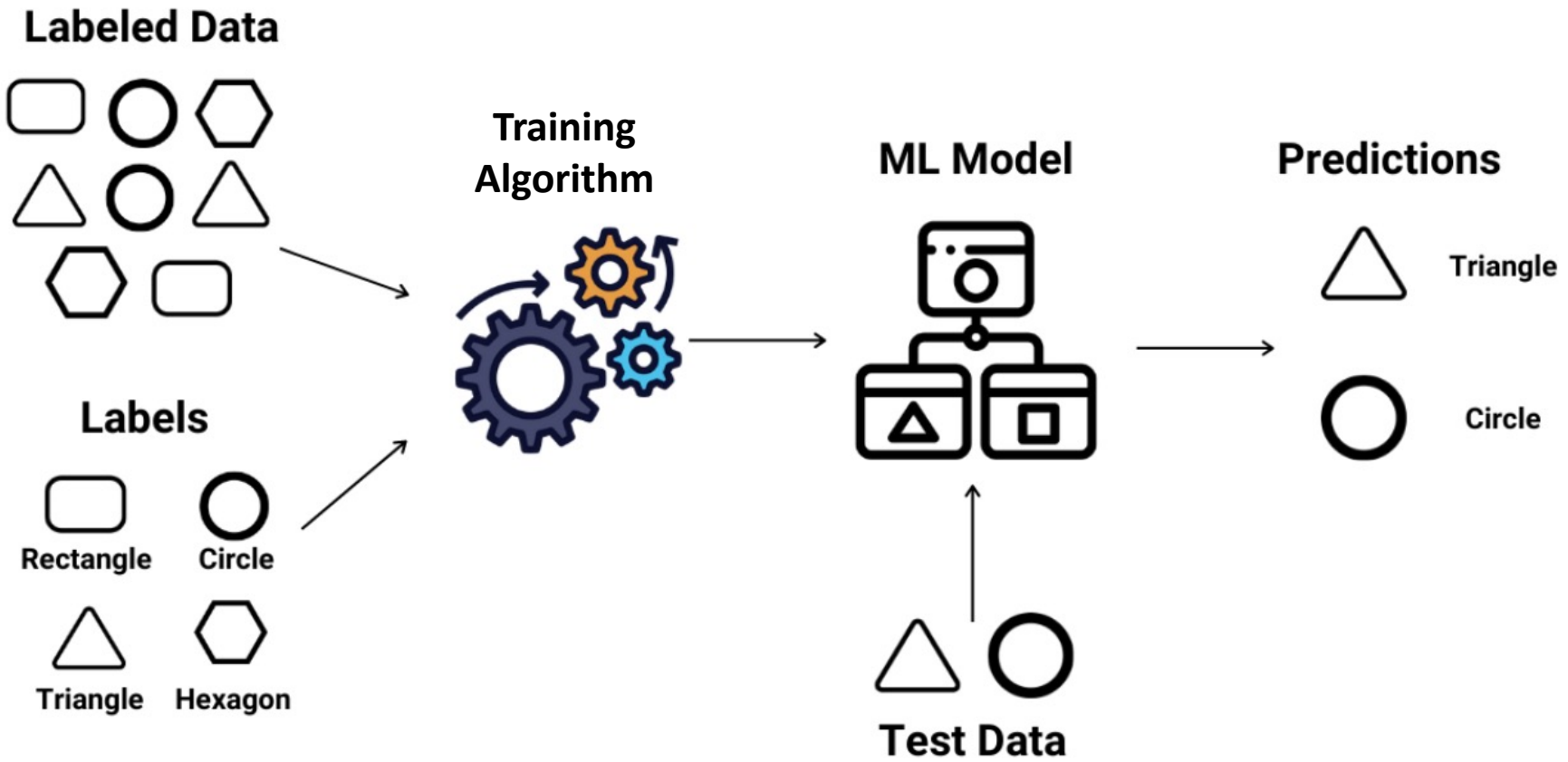
- **Week 9: Smart Lock Audio Recognition using TinyML**
  - Audio classification for deploying sensitive smart lock model
  - Data processing on audio data
  - Generate, train, and test a TensorFlow model using the SensiML Python SDK
  - Compile and flash the model to the edge device and display the inferred classes in the SensiML Open Gateway user interface
- **Week 10: Lecture or possibly additional time for the final project**
- **Week 11: Final Project Presentations**
  - **Each Group has 12 minutes (Suggested presentation – 9 minutes; Q&A— 3 mins)**
  - **Signup for the presentation order (Same as the project signup)**
- **Final report due on June 7th 11:59pm, 2024**

Machine Learning Algorithms



Supervised Learning

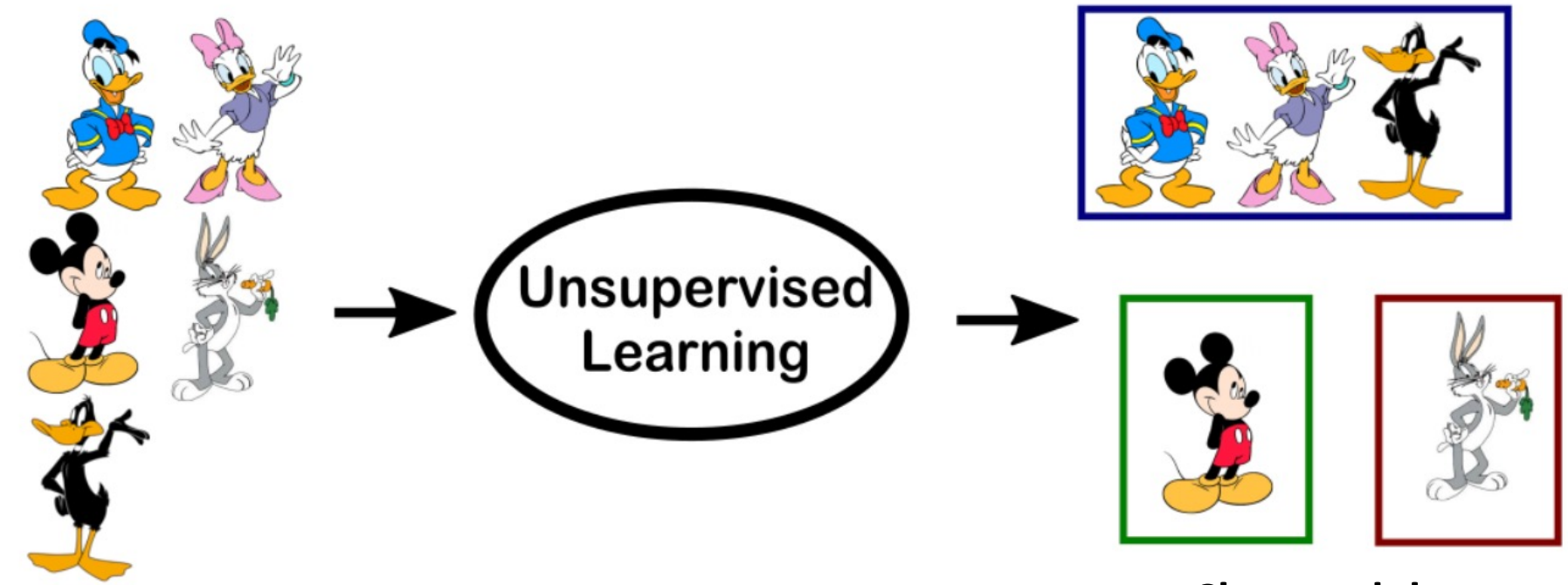
- Supervised learning methods **learn** from **labeled data** and use the **insight** gained to make decisions on the operational/testing data



Source: https://medium.com/@metehankozan/supervised-and-unsupervised-learning-an-intuitive-approach-cd8f8f64b644

Un-supervised Learning

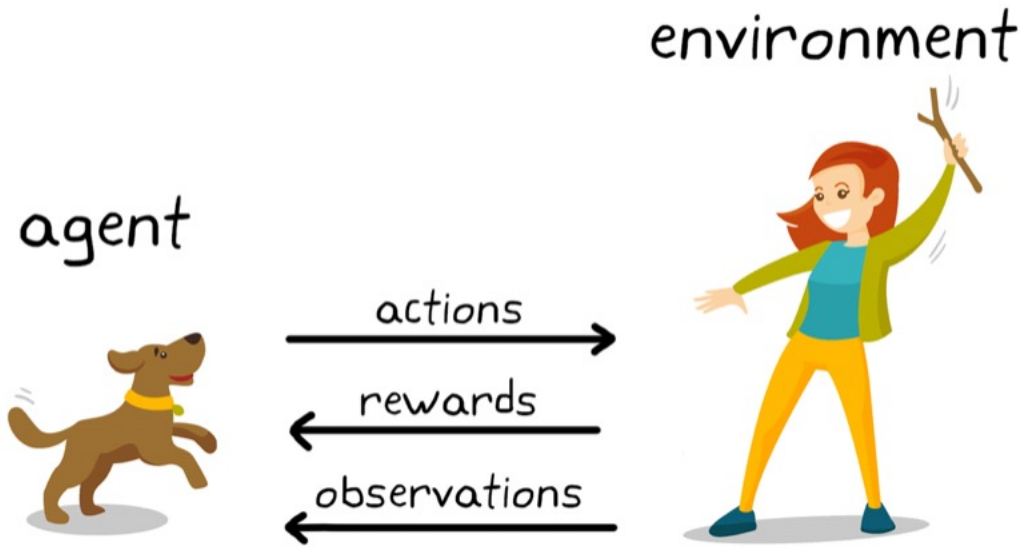
- Un-supervised learning technique is used when the initial data is **unlabeled**
- Draws insights by processing data whose structure is not known before hand



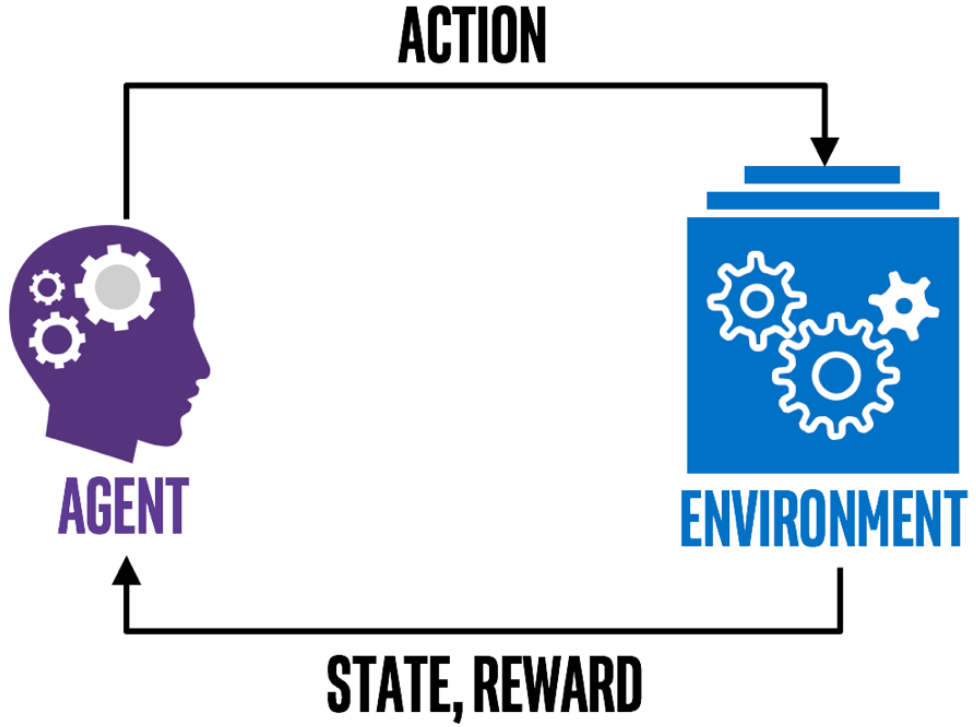
Source: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d

Reinforcement Learning

- Consider an agent that interacts with a certain environment, changing its state, and receives rewards (or penalties) for its actions
- Goal: Finding patterns of actions for the agent, by trying them all and comparing the results, that yield the maximum cumulative expected reward points in achieving the goal



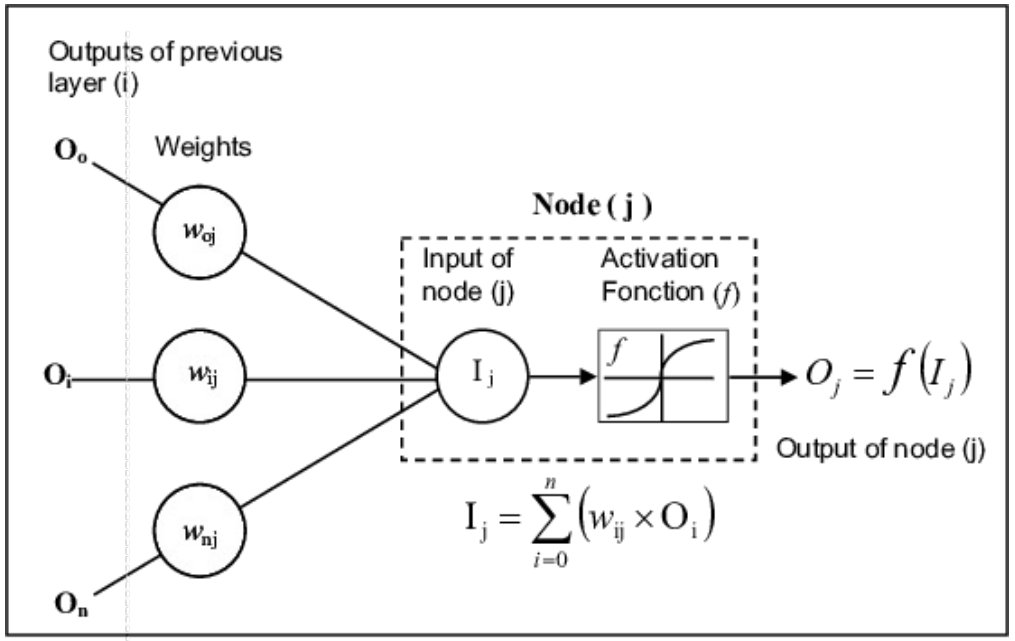
Source: https://www.mathworks.com/discovery/reinforcement-learning.html



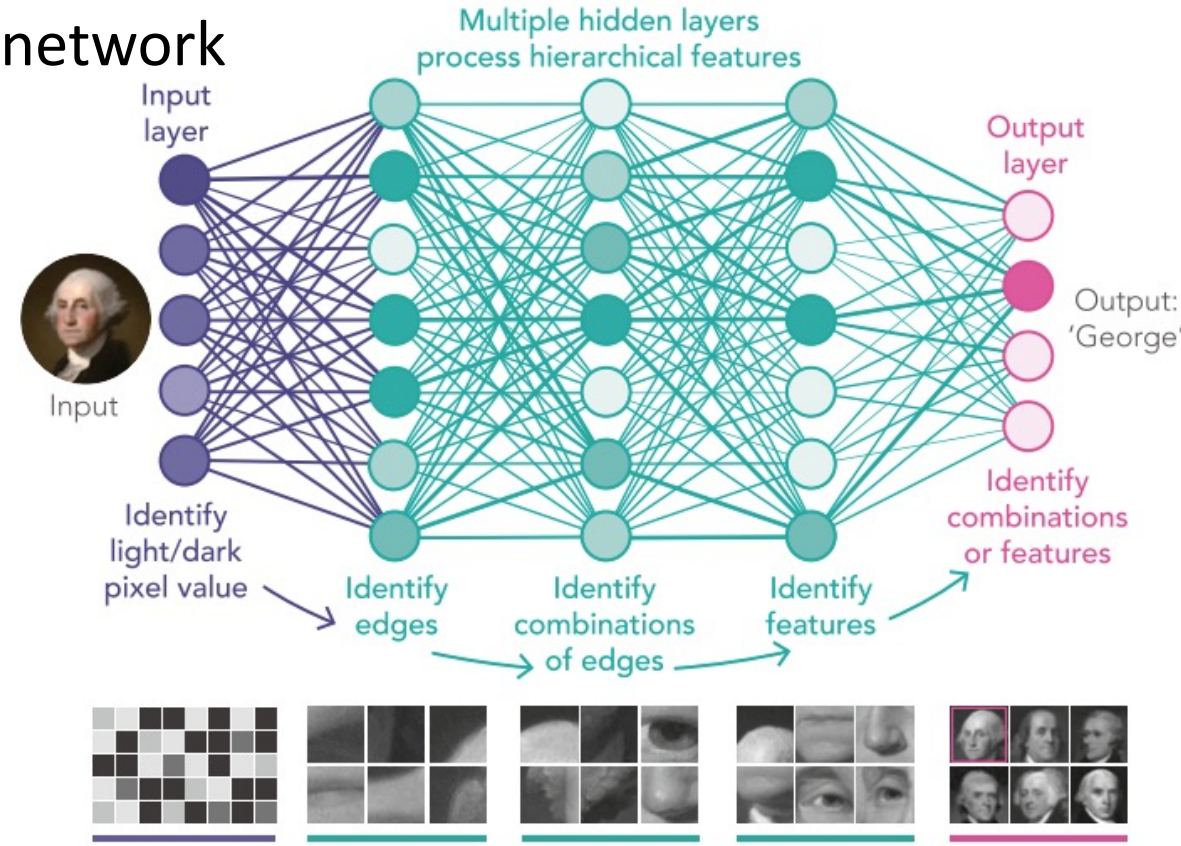
Source: https://intellabs.github.io/coach/

Deep Learning

- Basic unit is a node (also called “neuron”) which is composed of weights, biases, computing function, and an activation function
- Layers of nodes are used to form a deep neural network



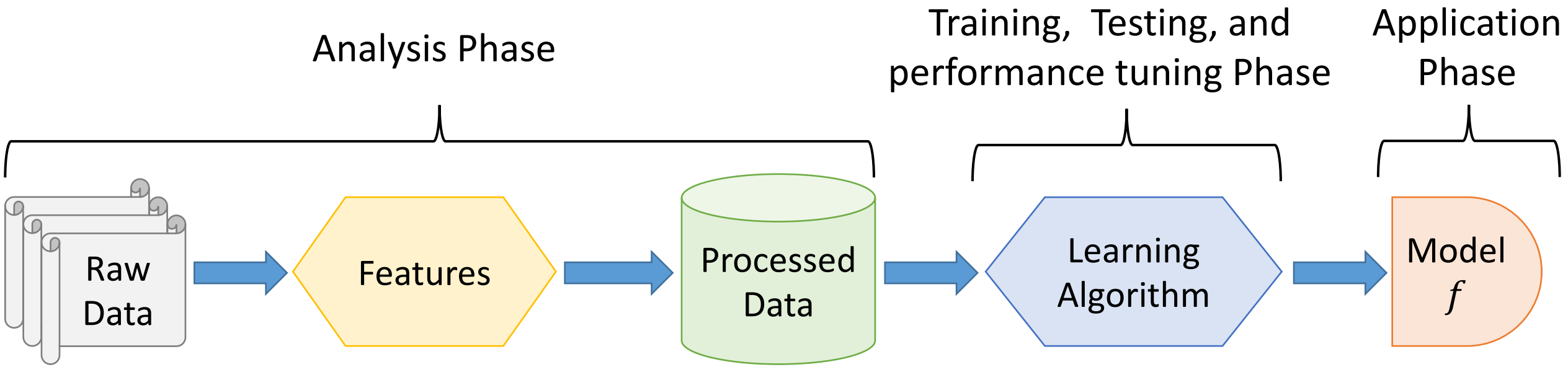
Schematic diagram of a node in deep neural network



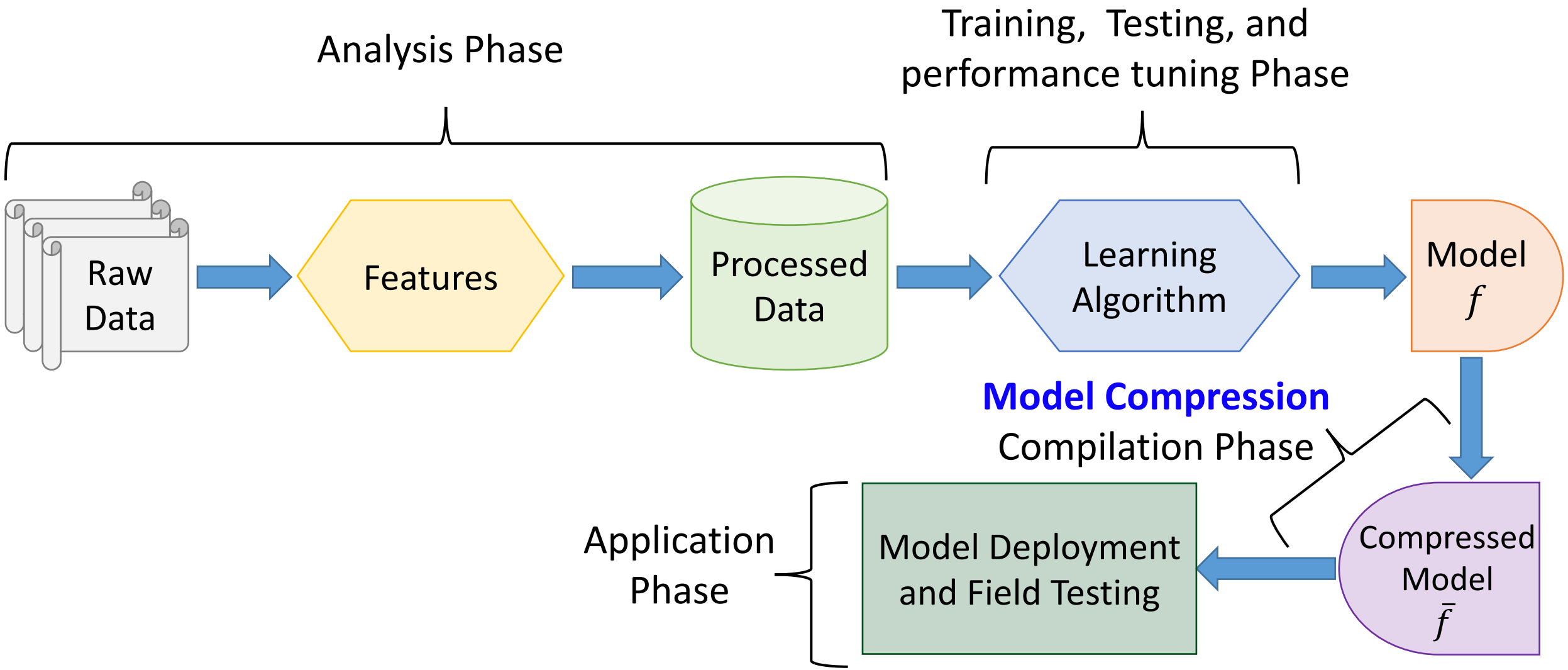
Schematic diagram of a deep neural network

Source: https://www.pnas.org/content/pnas/116/4/1074.full.pdf

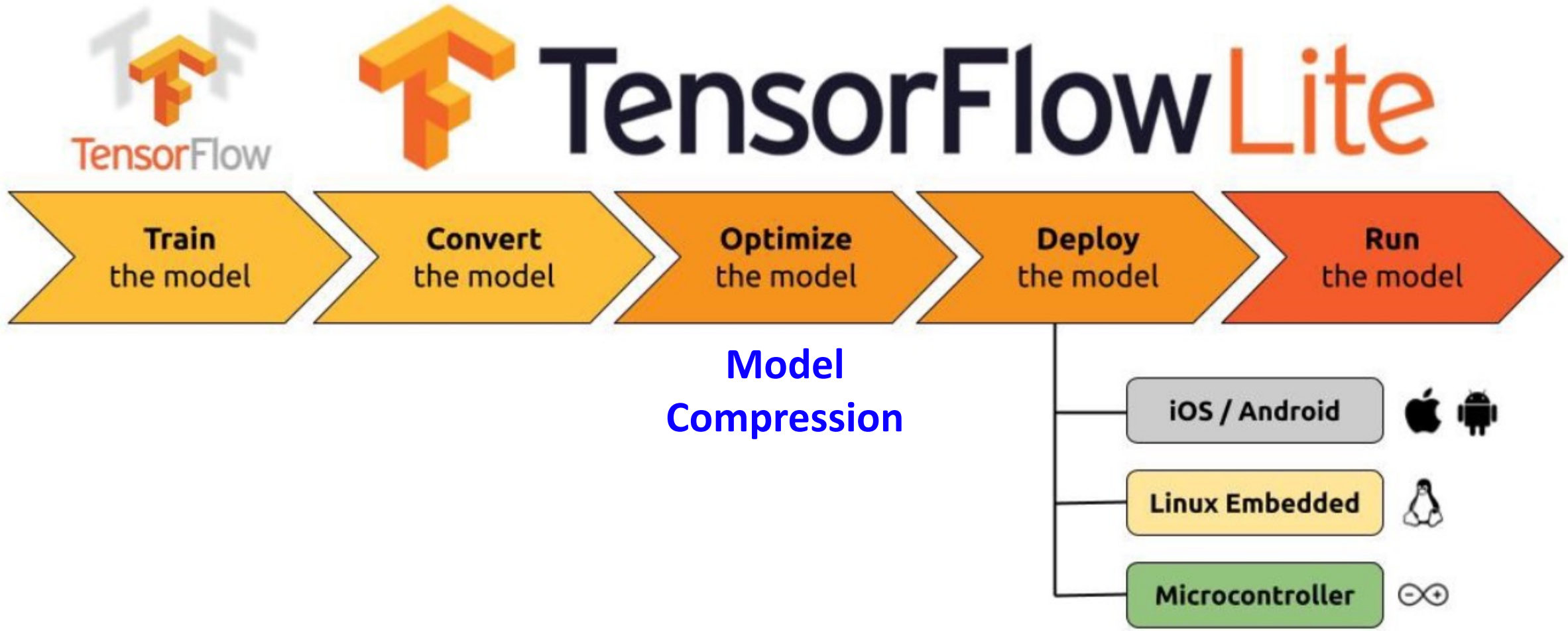
A Schematic View of ML Pipeline



A Schematic View of TinyML Pipeline



TensorFlow (TF) and TFLite Workflow for TinyML

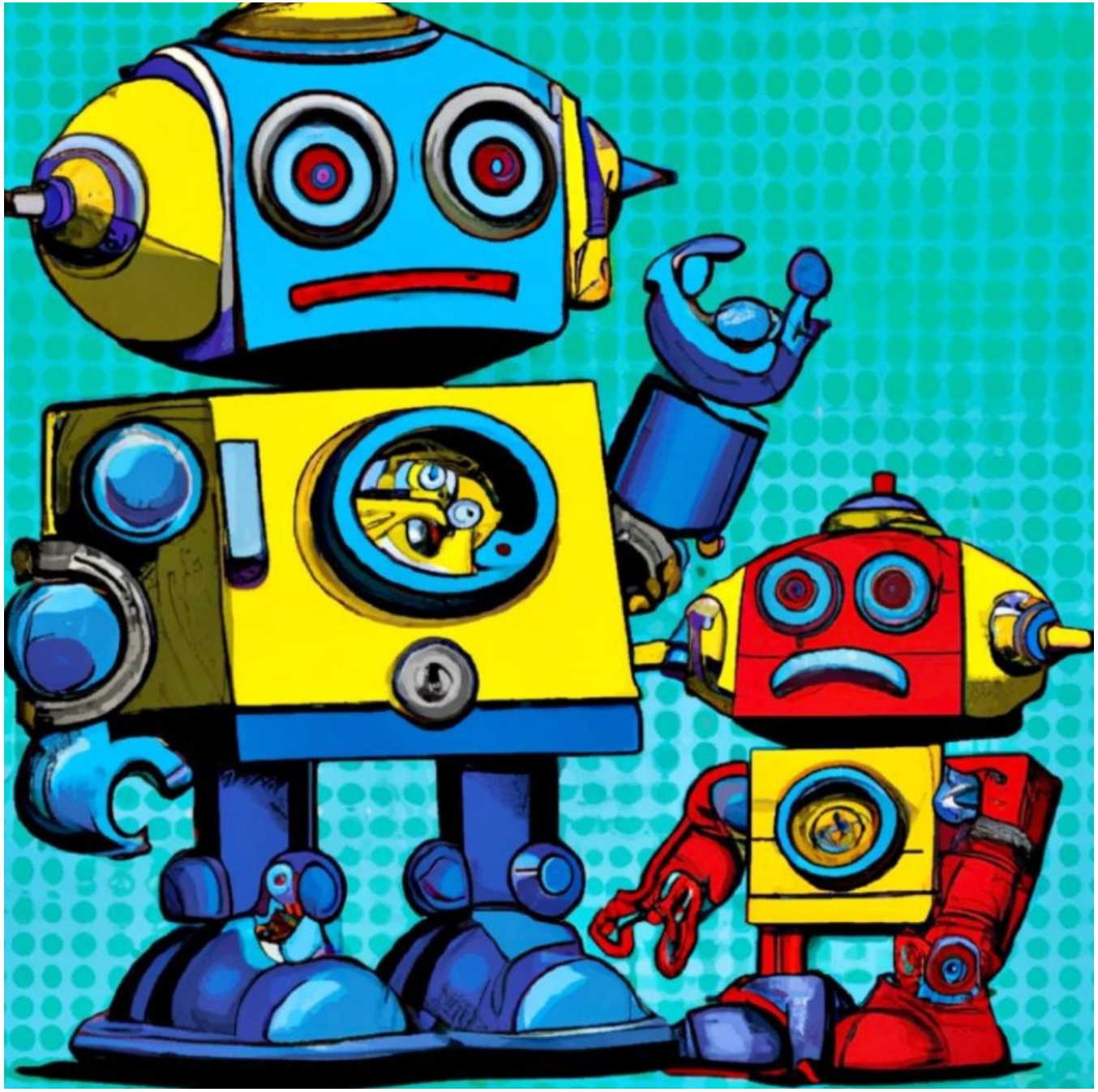


Source: https://leonardocavagnis.medium.com/tinyml-machine-learning-for-embedded-system-part-i-92a34529e899

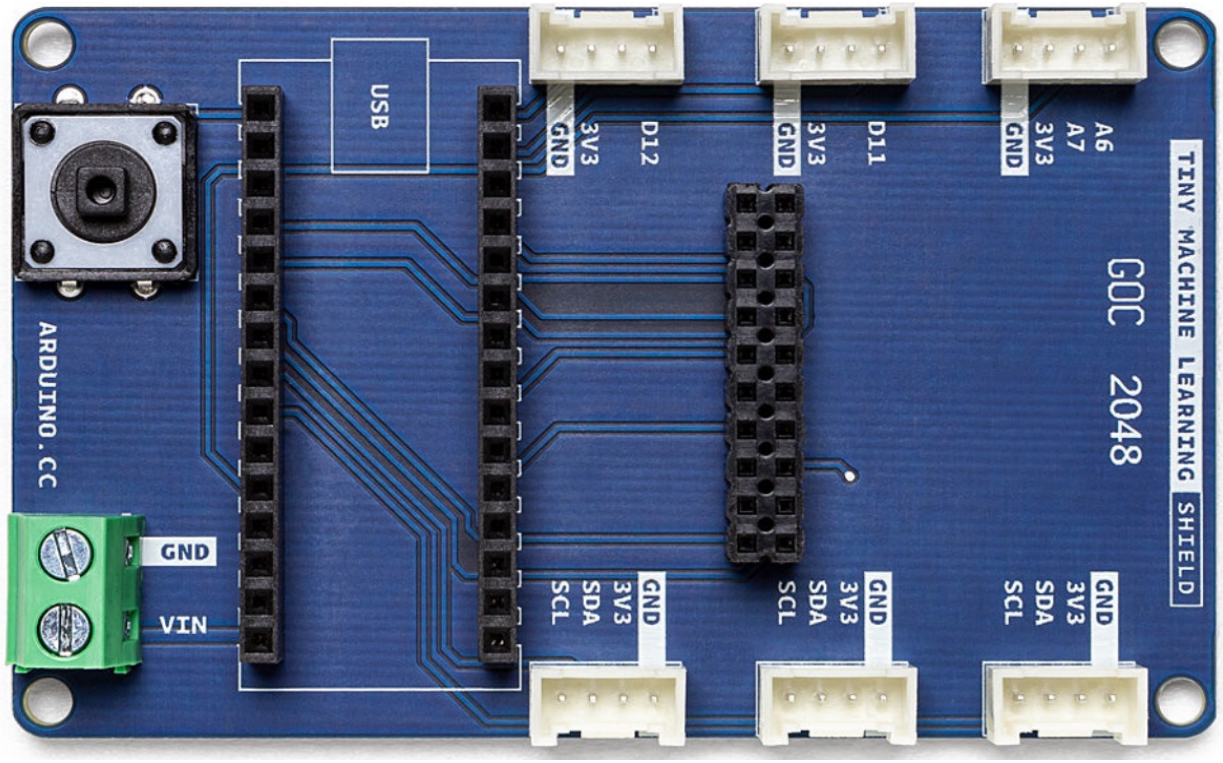
10 min Break

Today's Lab

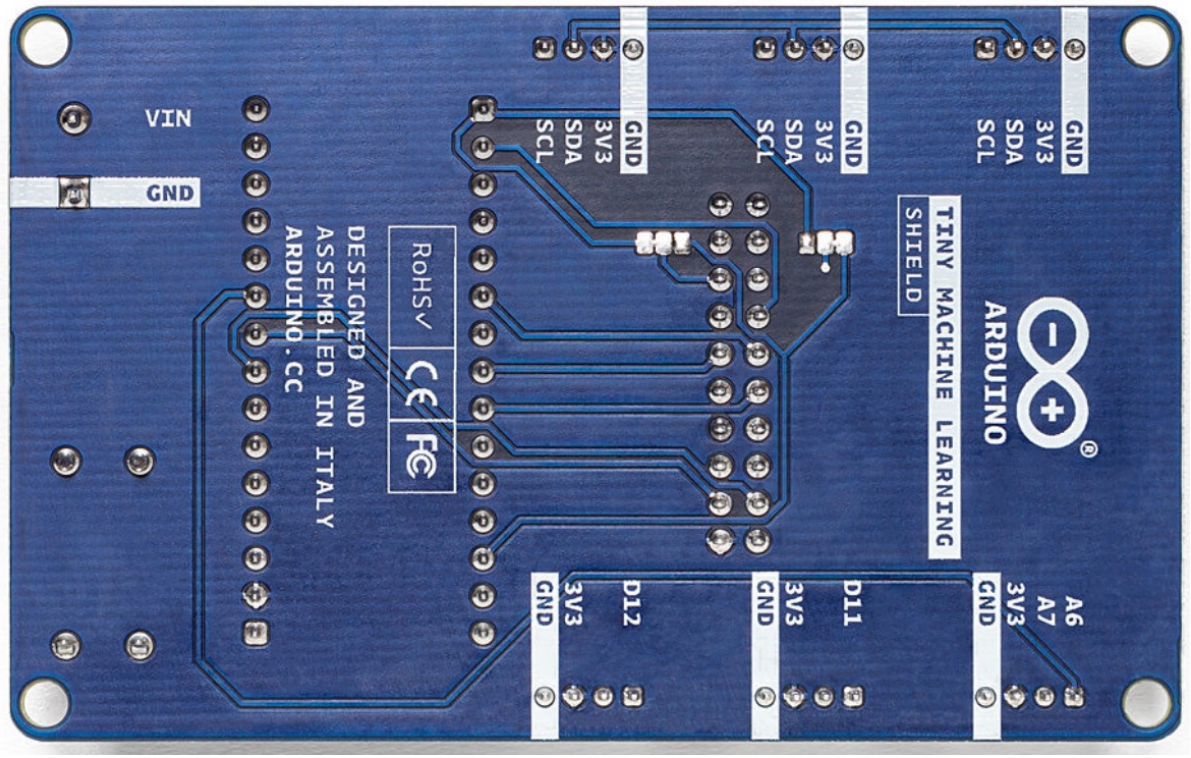
- Setting up TinyML hardware
- Setting up Arduino Desktop IDE
- The Arduino Blink Example
- Testing the Sensors



Components of TinyML Kit

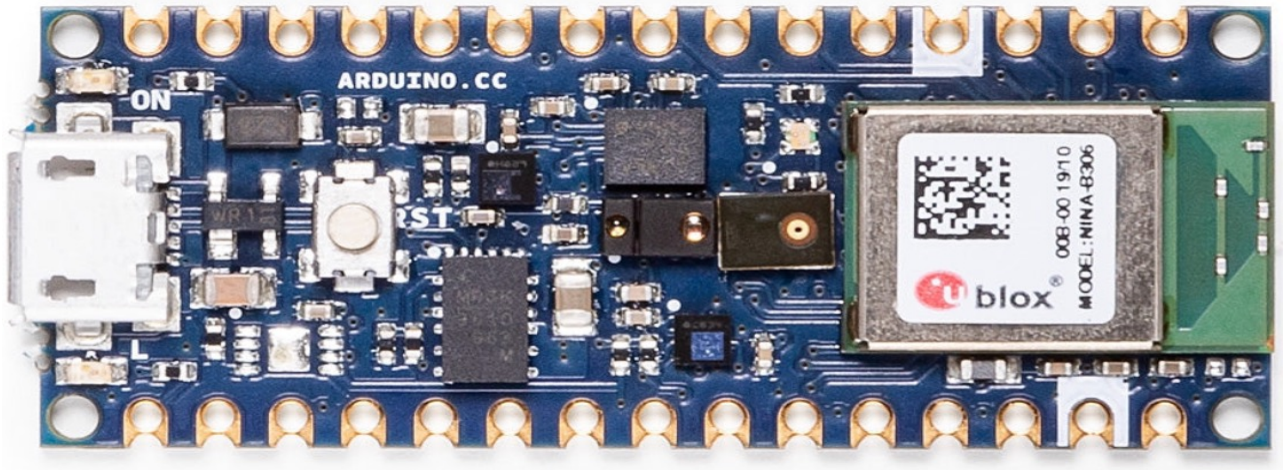


Tiny Machine Learning Shield (Front)



Tiny Machine Learning Shield (Back)

Components of TinyML Kit



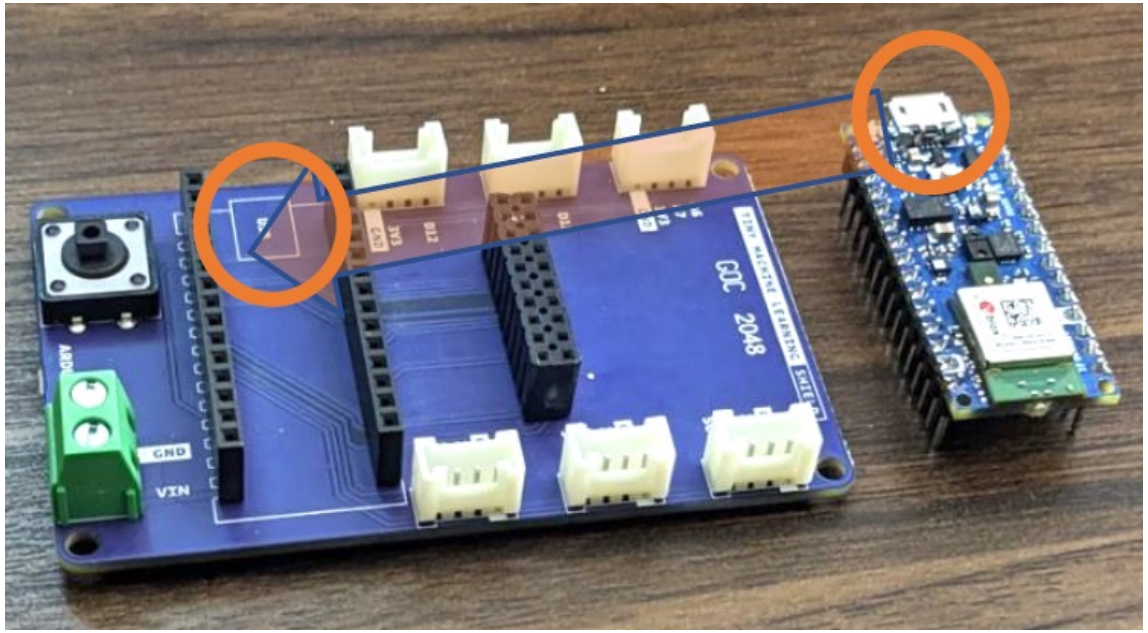
Tiny Machine Learning Shield

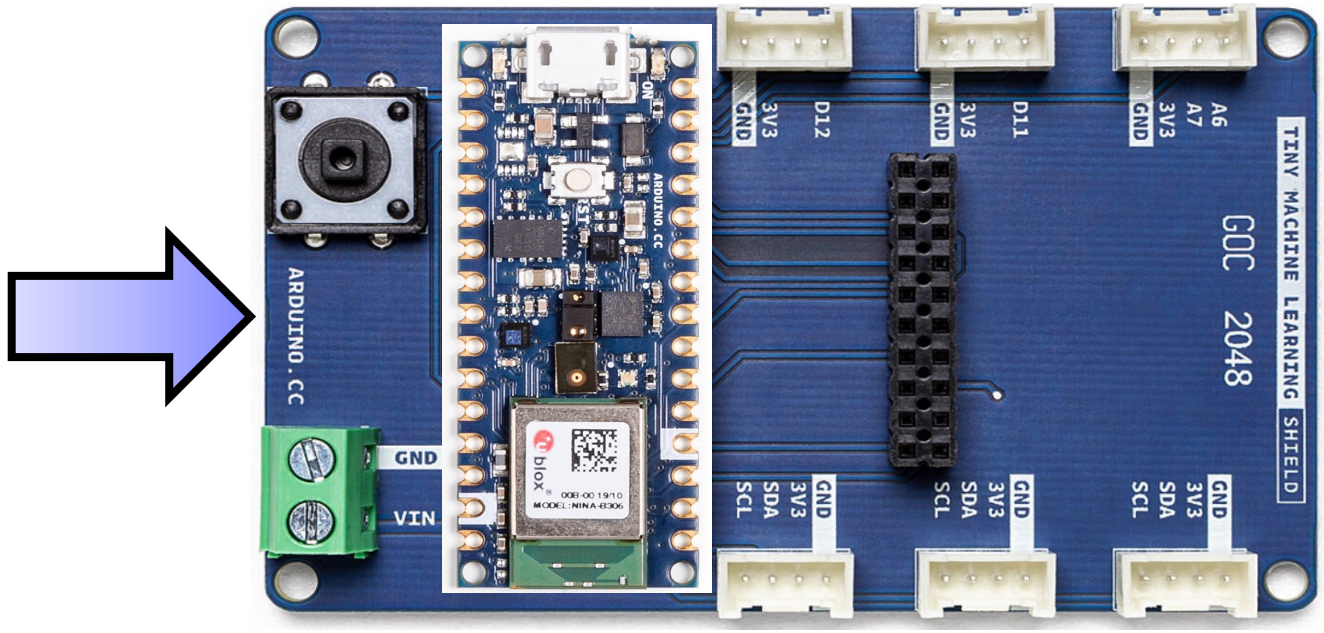
USB A – Micro USB Cable (1m)

1 OV7675 Camera

Setting up TinyML Kit

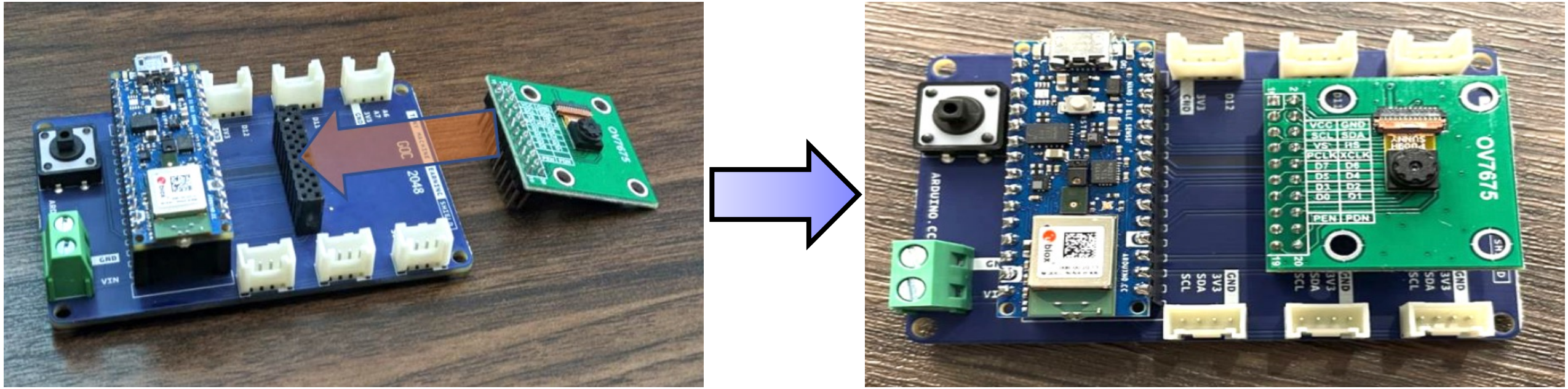
1. Slot the Nano 33 BLE Sense board into the Tiny Machine Learning shield





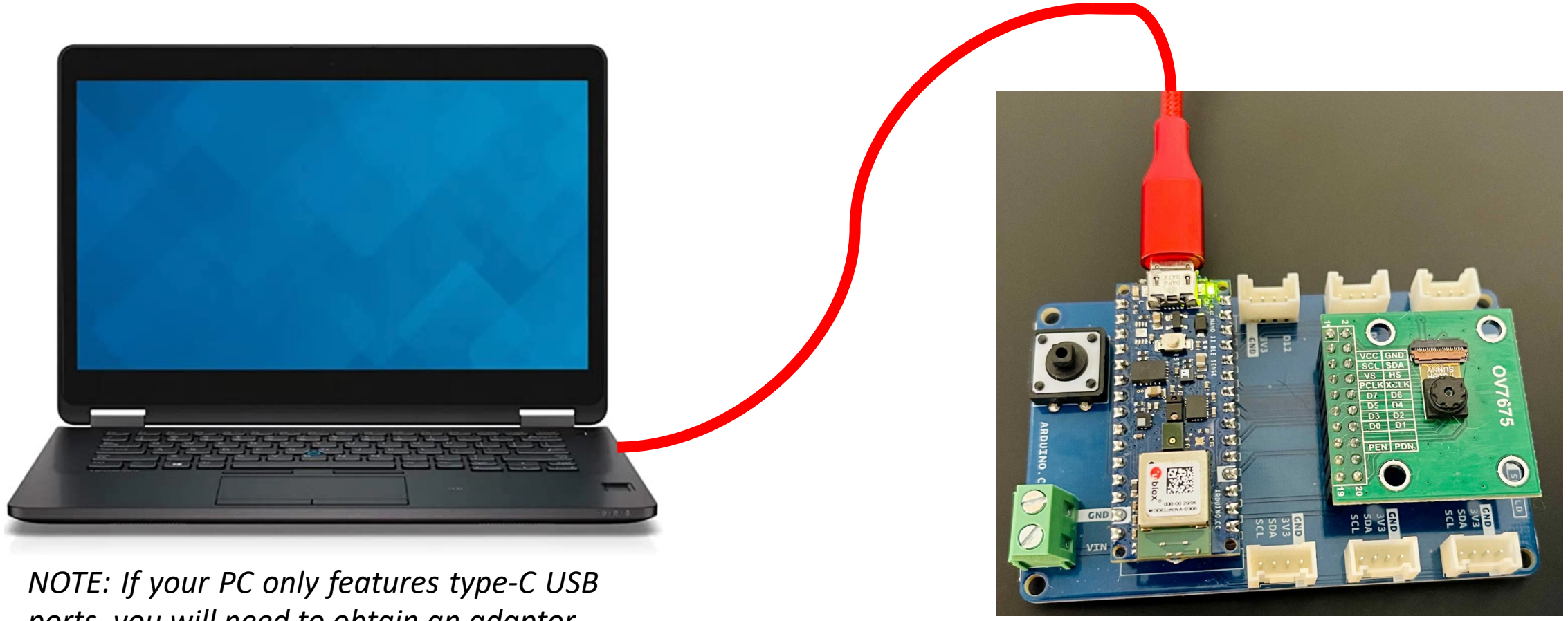
Setting up TinyML Kit

2. Slot the OV7675 camera module into the shield Tiny Machine Learning shield



Setting up TinyML Kit

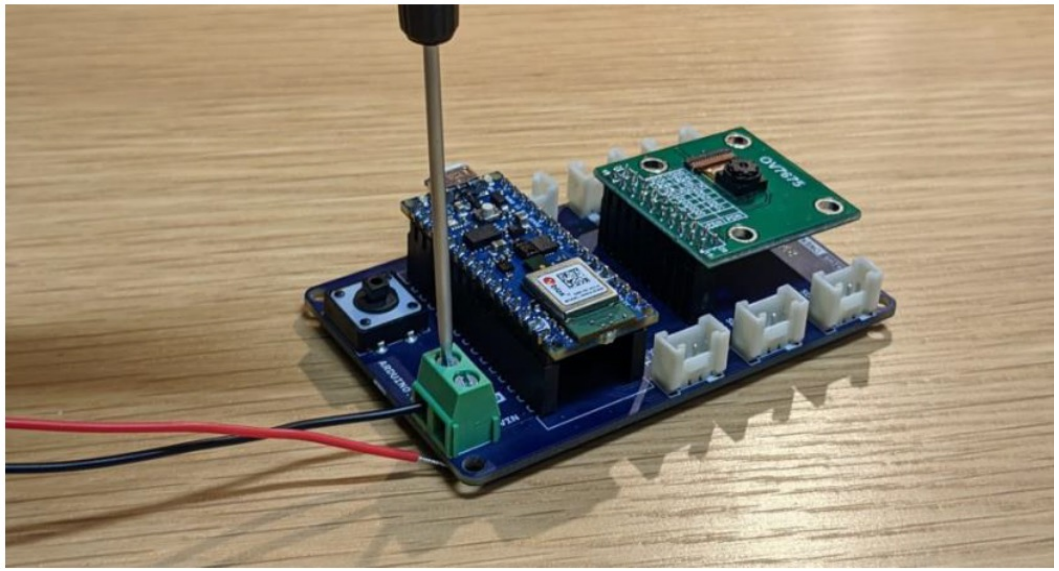
3. Use the USB cable (type-A to microB) to connect the Nano 33 BLE Sense to your machine

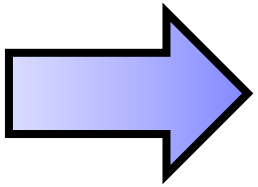


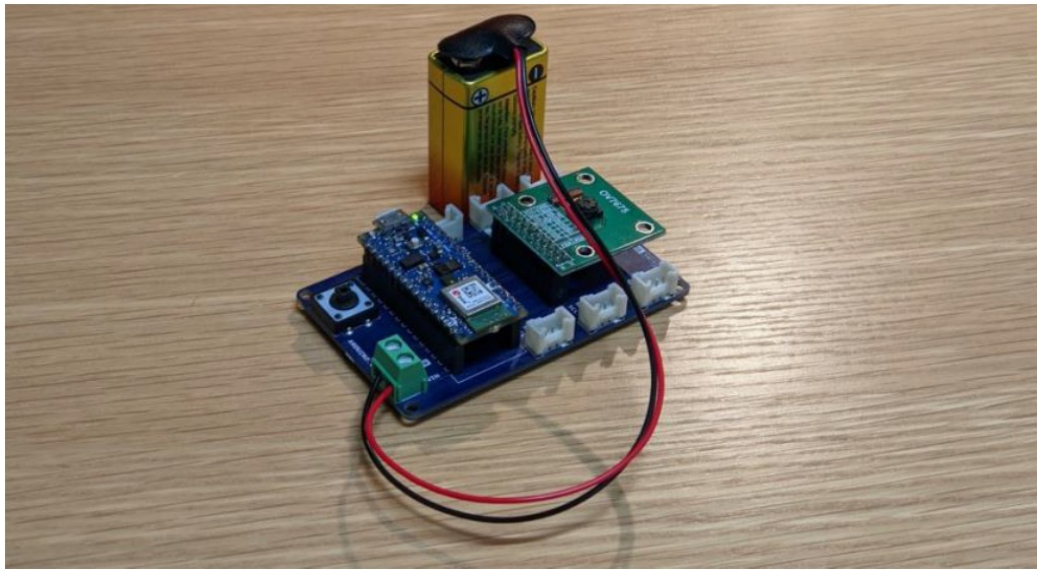
NOTE: If your PC only features type-C USB ports, you will need to obtain an adaptor

Setting up TinyML Kit

- Using a 9V battery to power Nano 33 BLE Sense
  - Screw down a wire leading from the negative battery terminal (black) to GND. (Most < 3mm flat-head screwdrivers will suffice here).
  - Repeat this process for the positive battery terminal (red) to VIN. And that's it you're all set to power your Arduino from a battery!

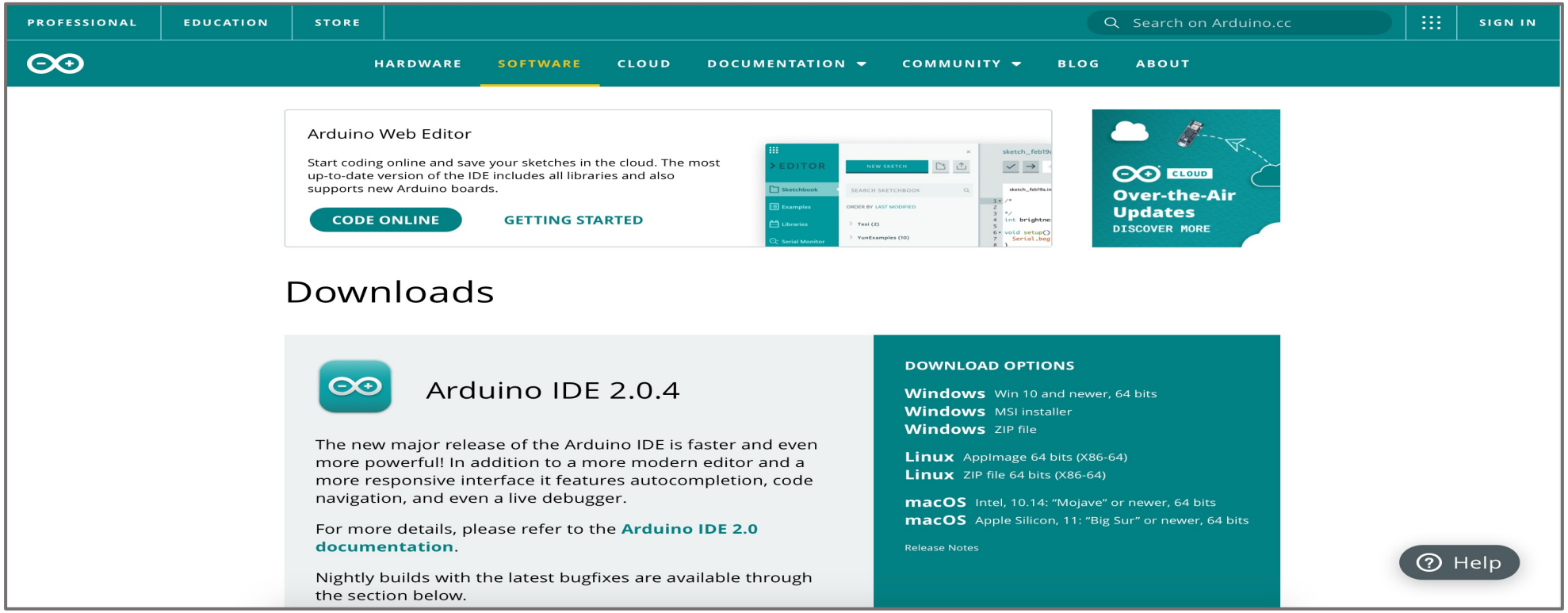






Setting up Arduino Desktop IDE

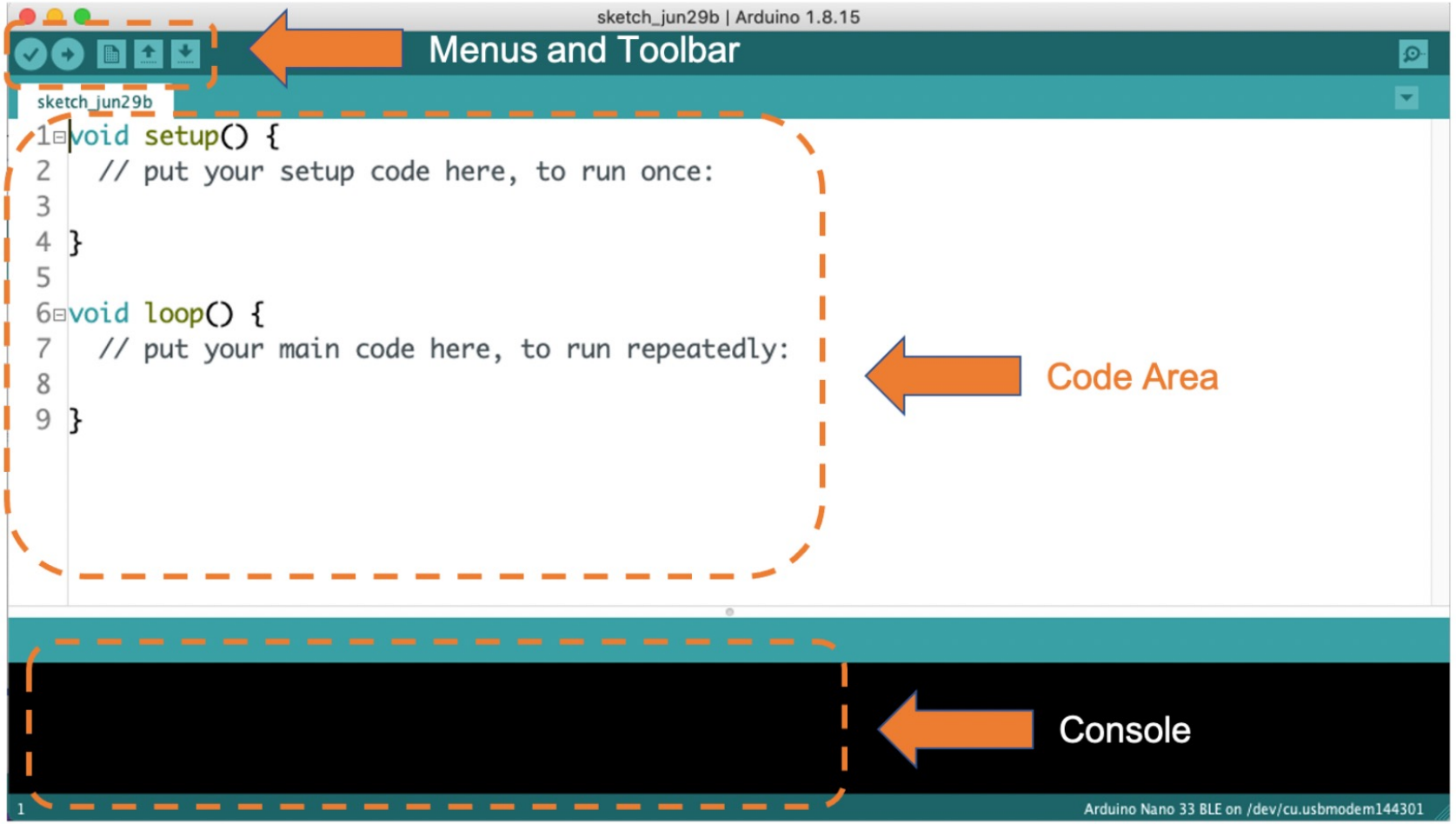
- **Integrated Development Environment (IDE):** An application that facilitates software development
- **Arduino Desktop IDE:** facilitate software development for microcontroller boards, in **C++**



- **Download link:** https://www.arduino.cc/en/software

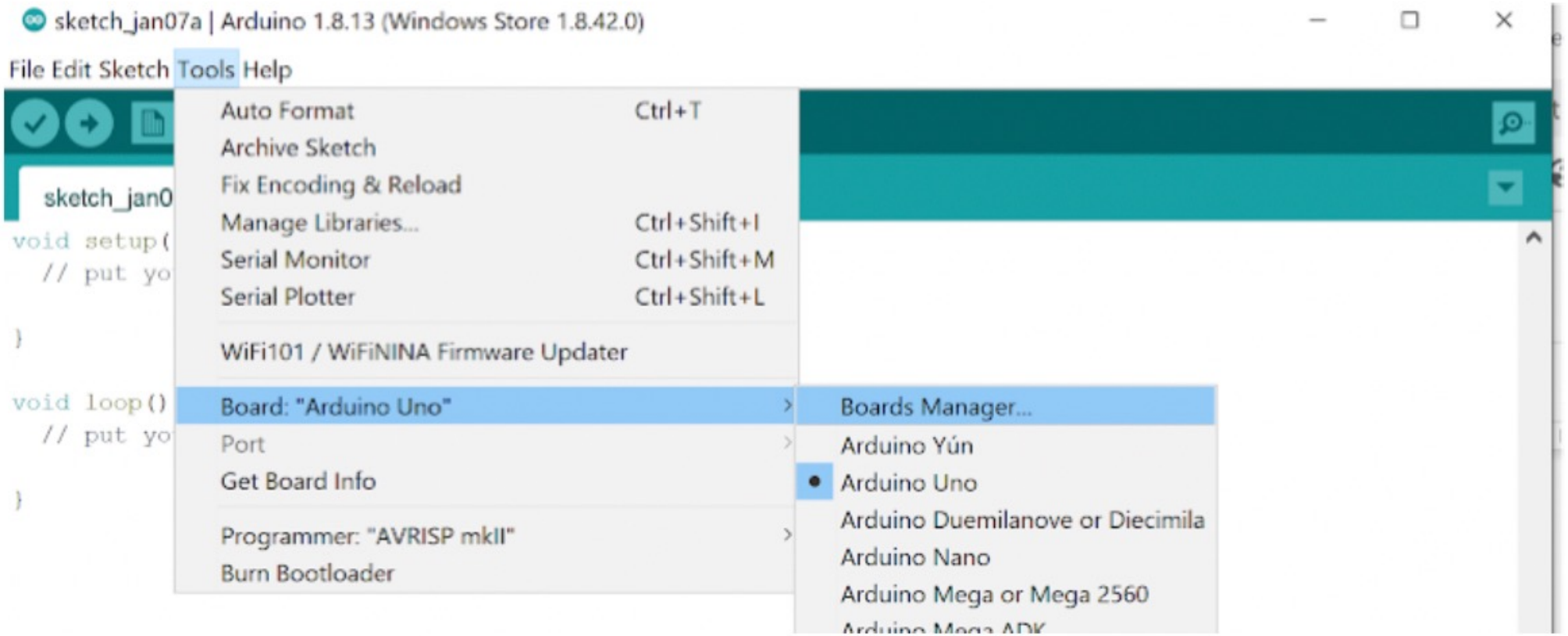
Setting up Arduino Desktop IDE

A Quick Tour of the IDE

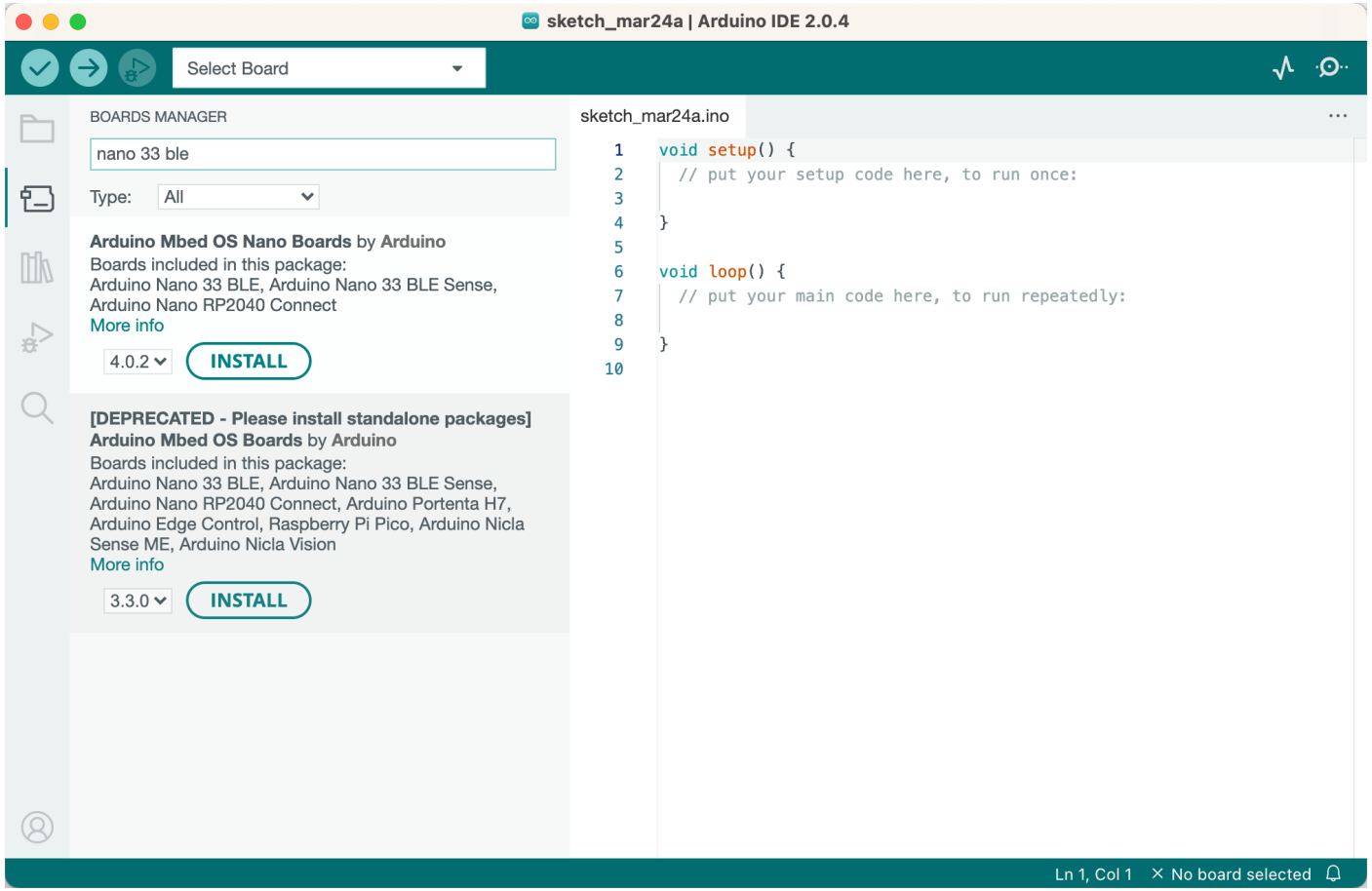


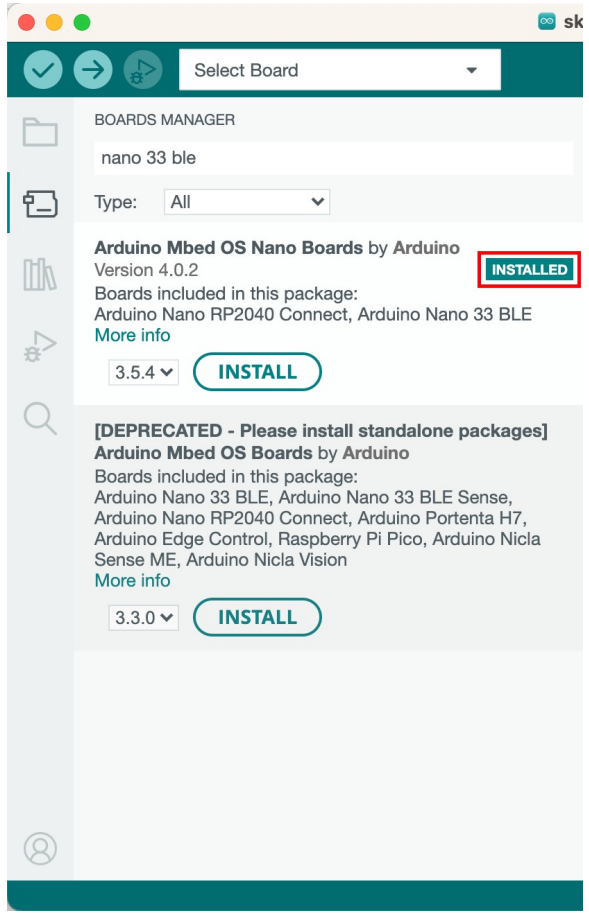
Setting up Arduino Desktop IDE

- Installing the Board Files for the Nano 33 BLE Sense



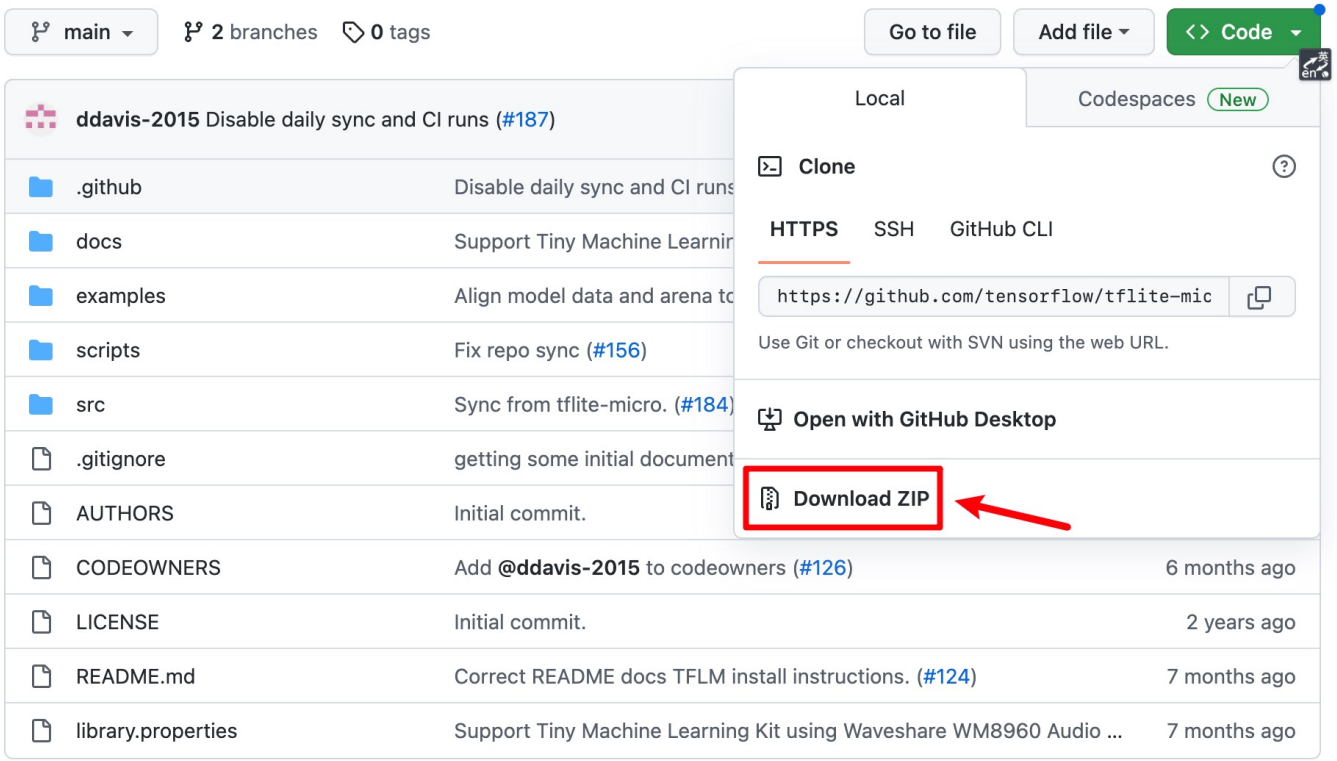
Setting up Arduino Desktop IDE





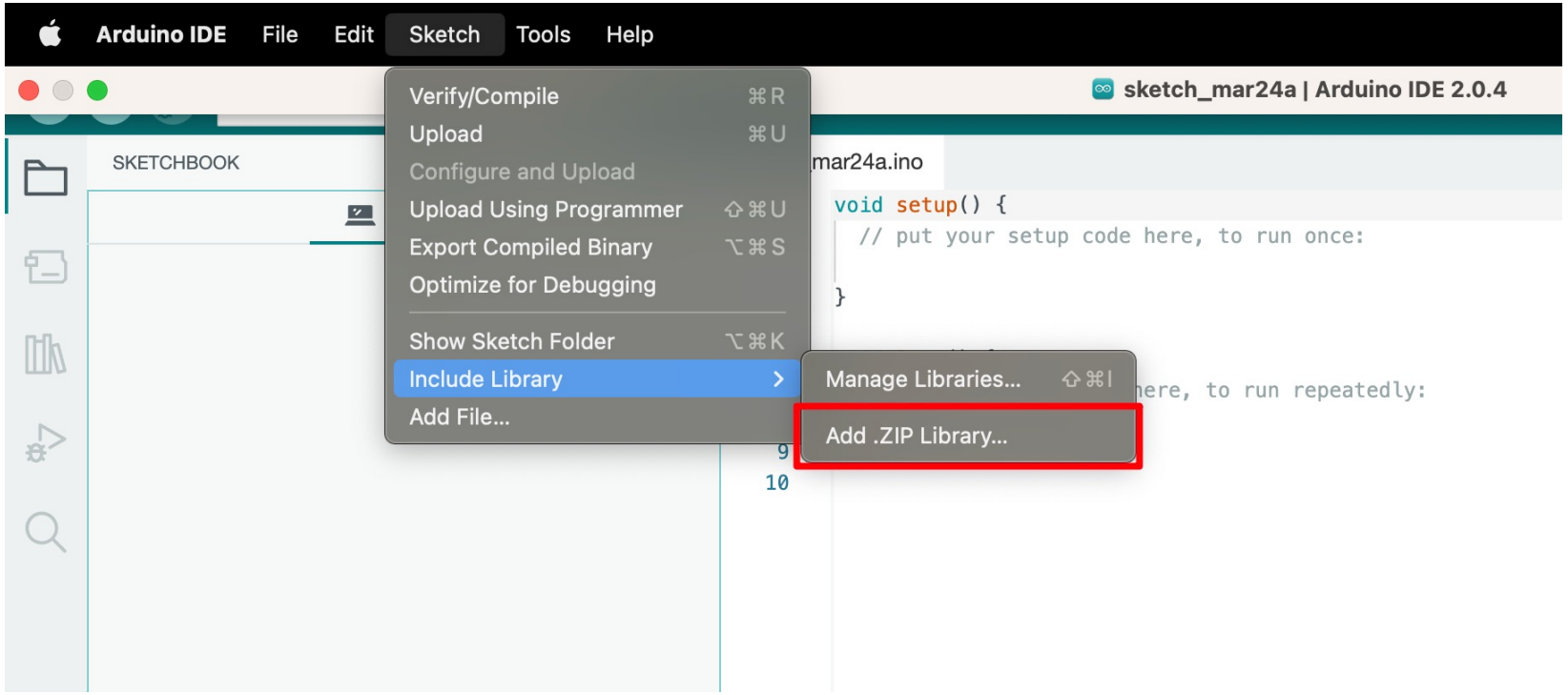
Setting up Arduino Desktop IDE

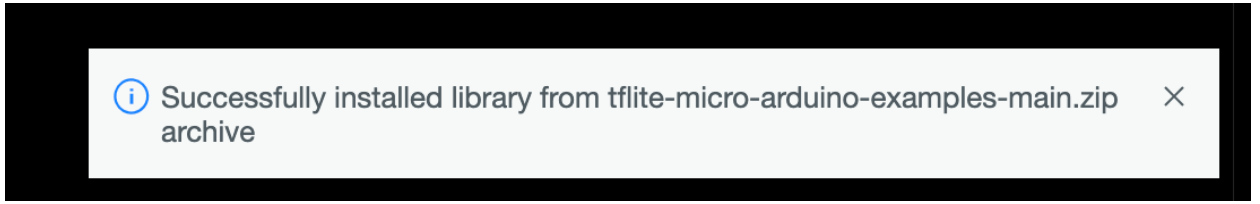
- Installing the Libraries needed
- **1. TensorFlow Lite Micro Library for Arduino**
- https://github.com/tensorflow/tflite-micro-arduino-examples#how-to-install



Setting up Arduino Desktop IDE

1. TensorFlow Lite Micro Library for Arduino

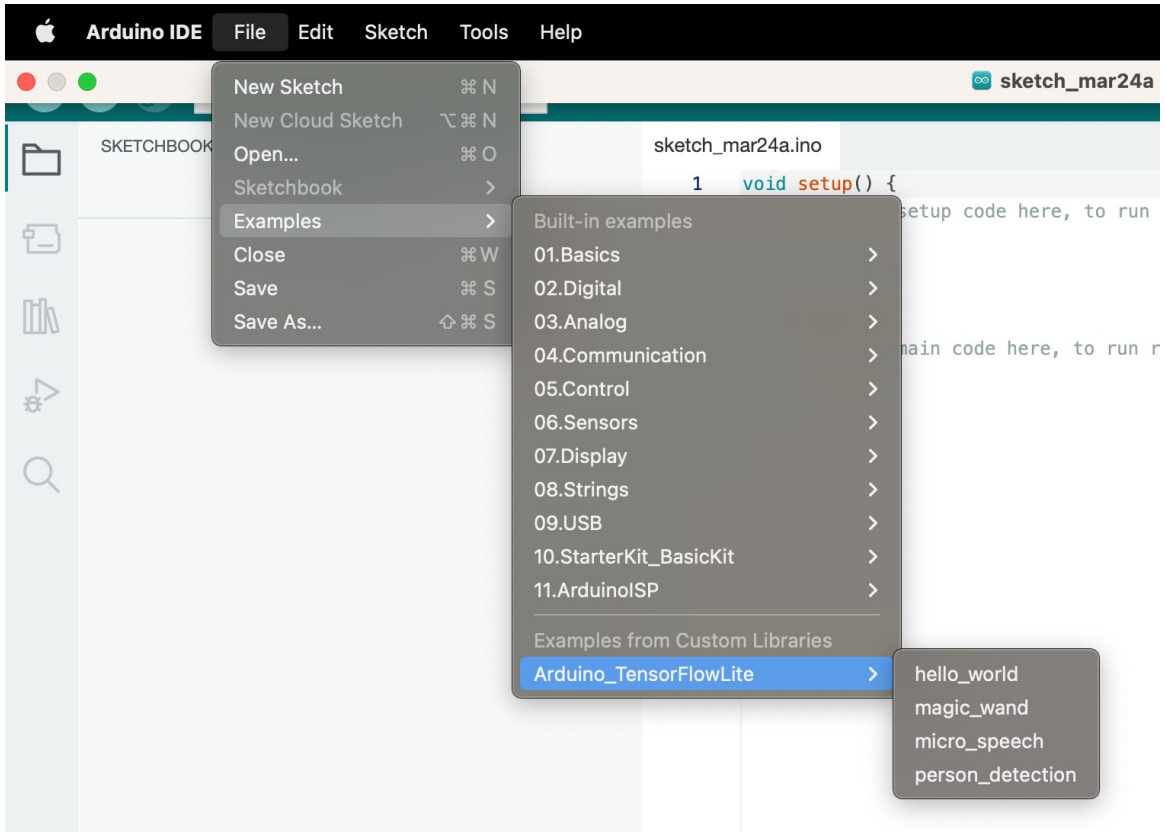




Setting up Arduino Desktop IDE

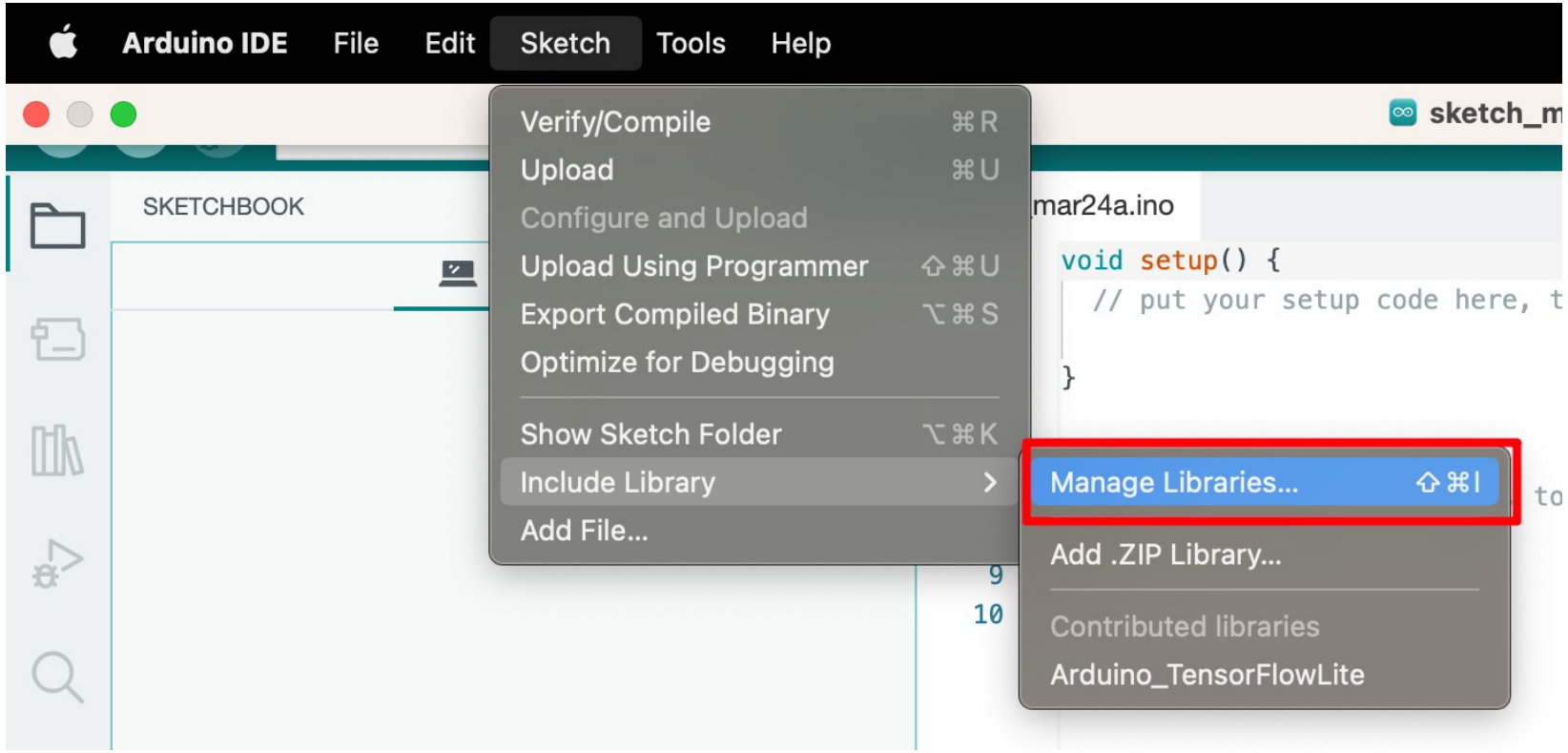
1. TensorFlow Lite Micro Library for Arduino

Now you can see…



Setting up Arduino Desktop IDE

2. Harvard_TinyMLx

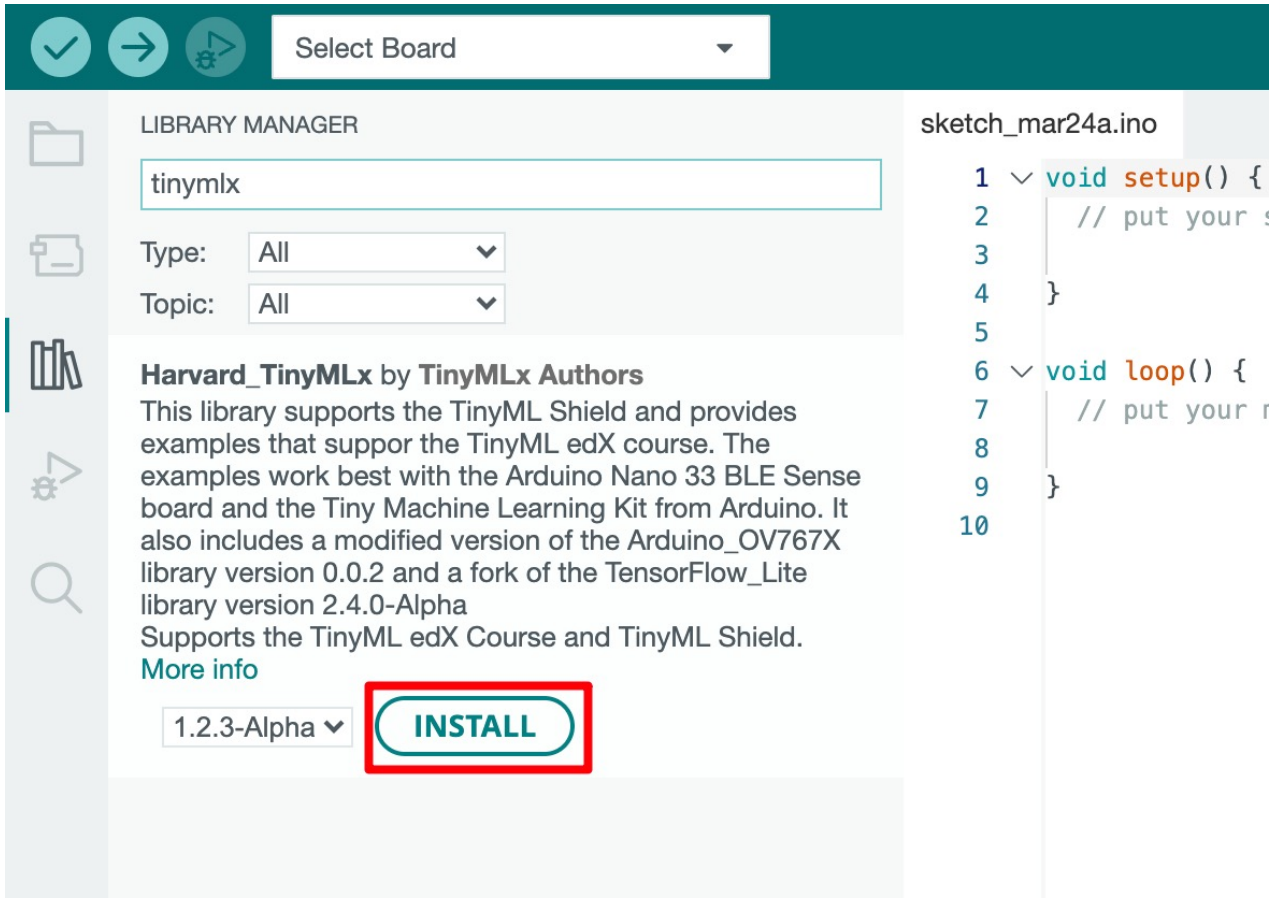


Setting up Arduino Desktop IDE

2. Harvard_TinyMLx

Search for tinymlx

And install!

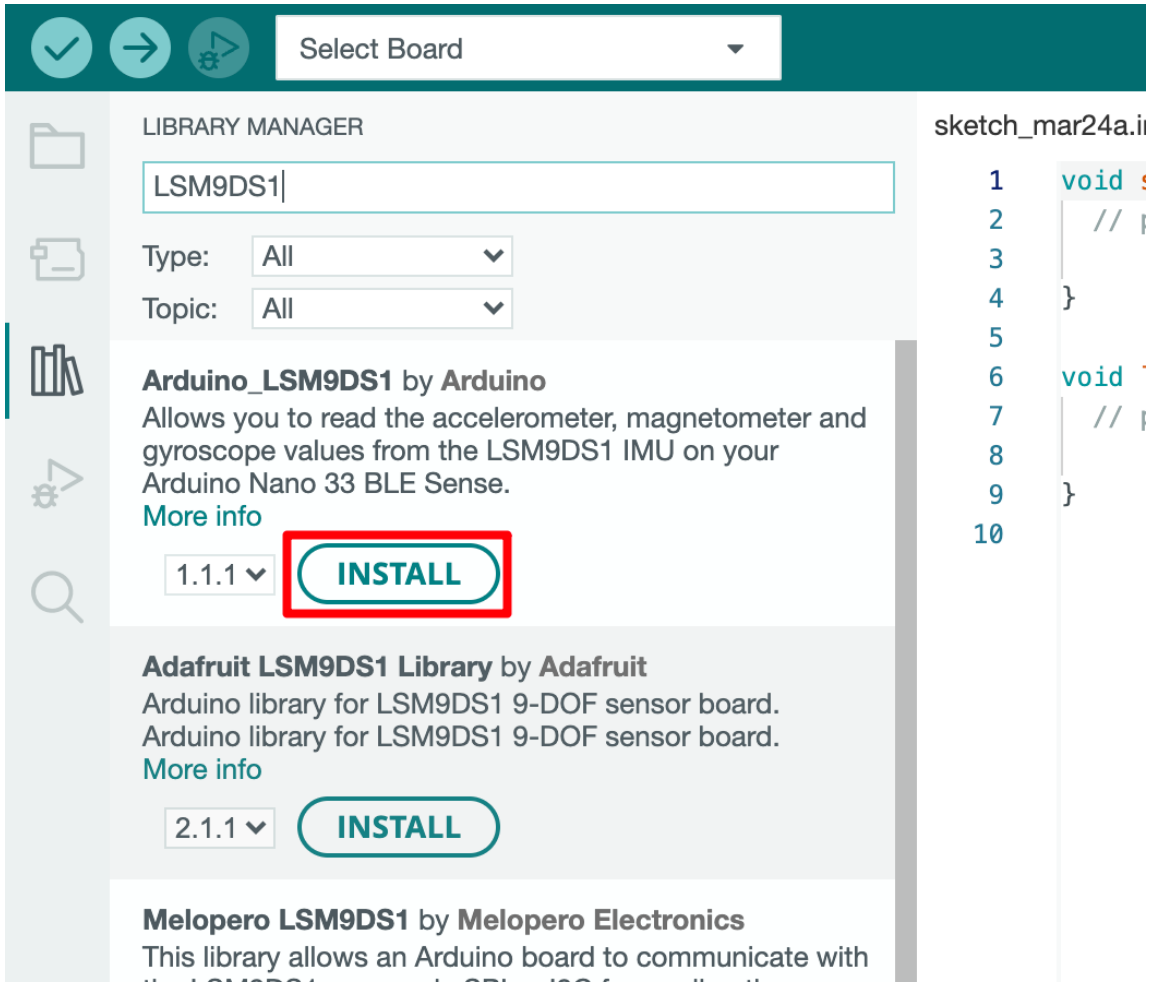


Setting up Arduino Desktop IDE

3. Arduino_LSM9DS1

Search for LSM9DS1

And install!

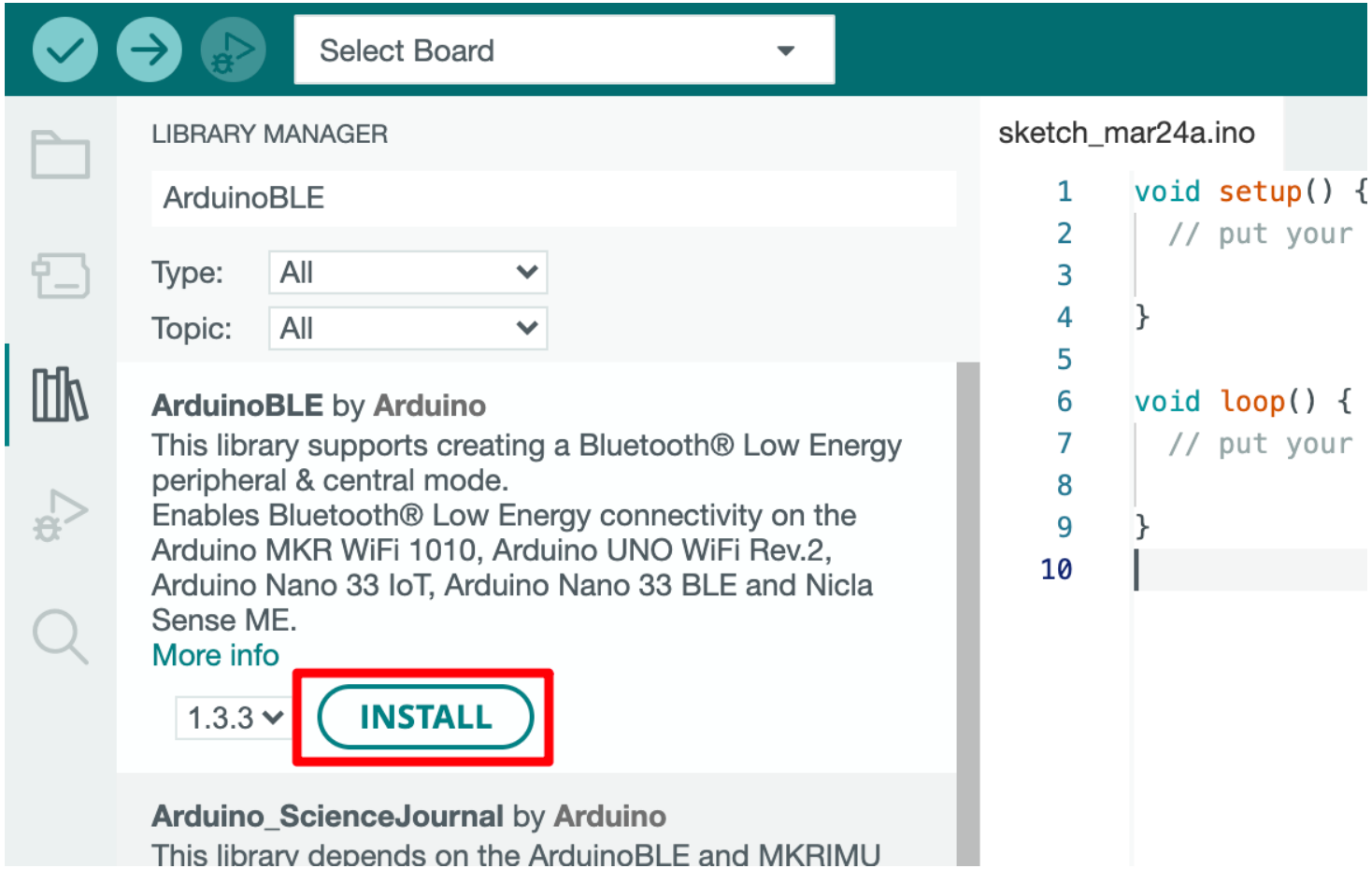


Setting up Arduino Desktop IDE

4. ArduinoBLE

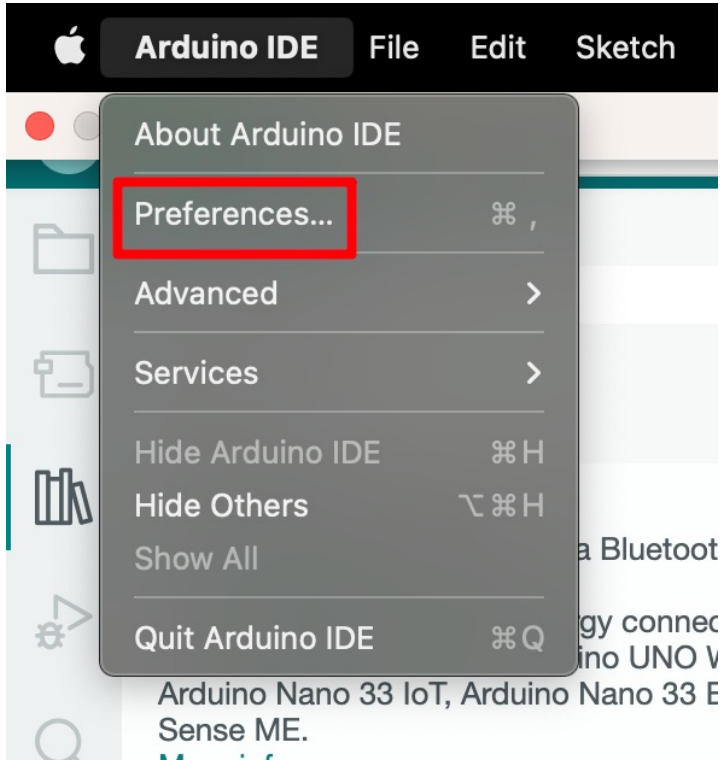
Search for ArduinoBLE

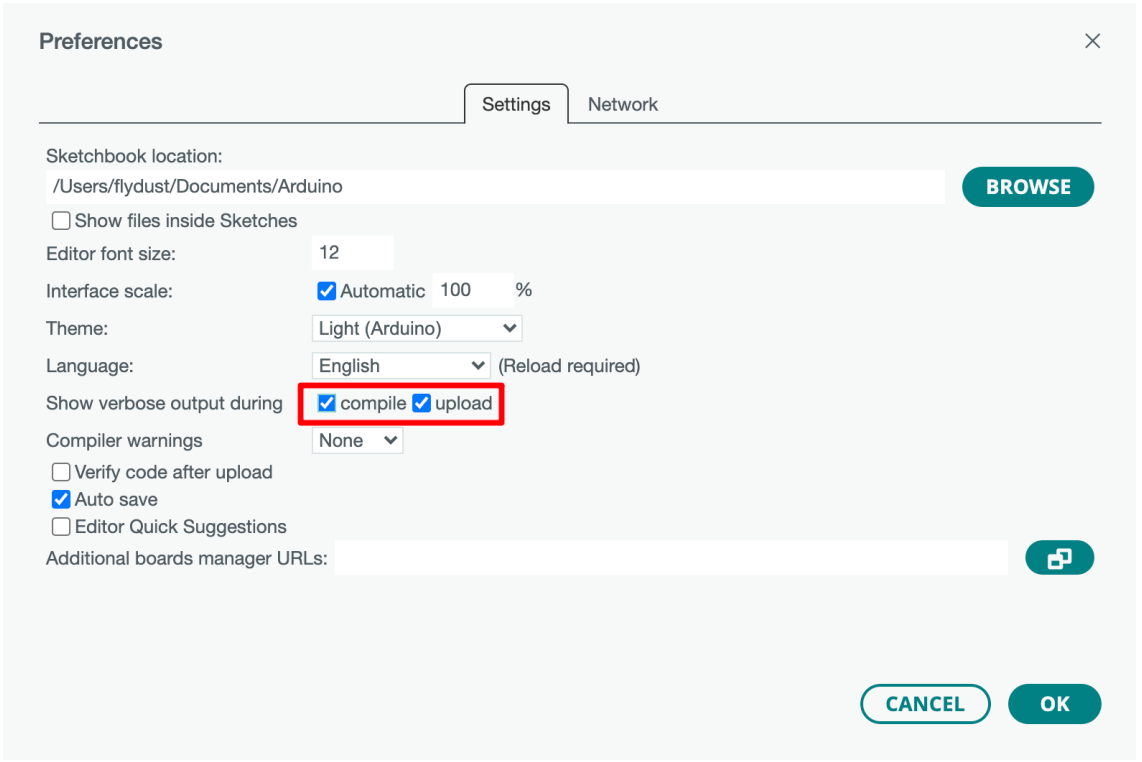
And install!



Setting up Arduino Desktop IDE

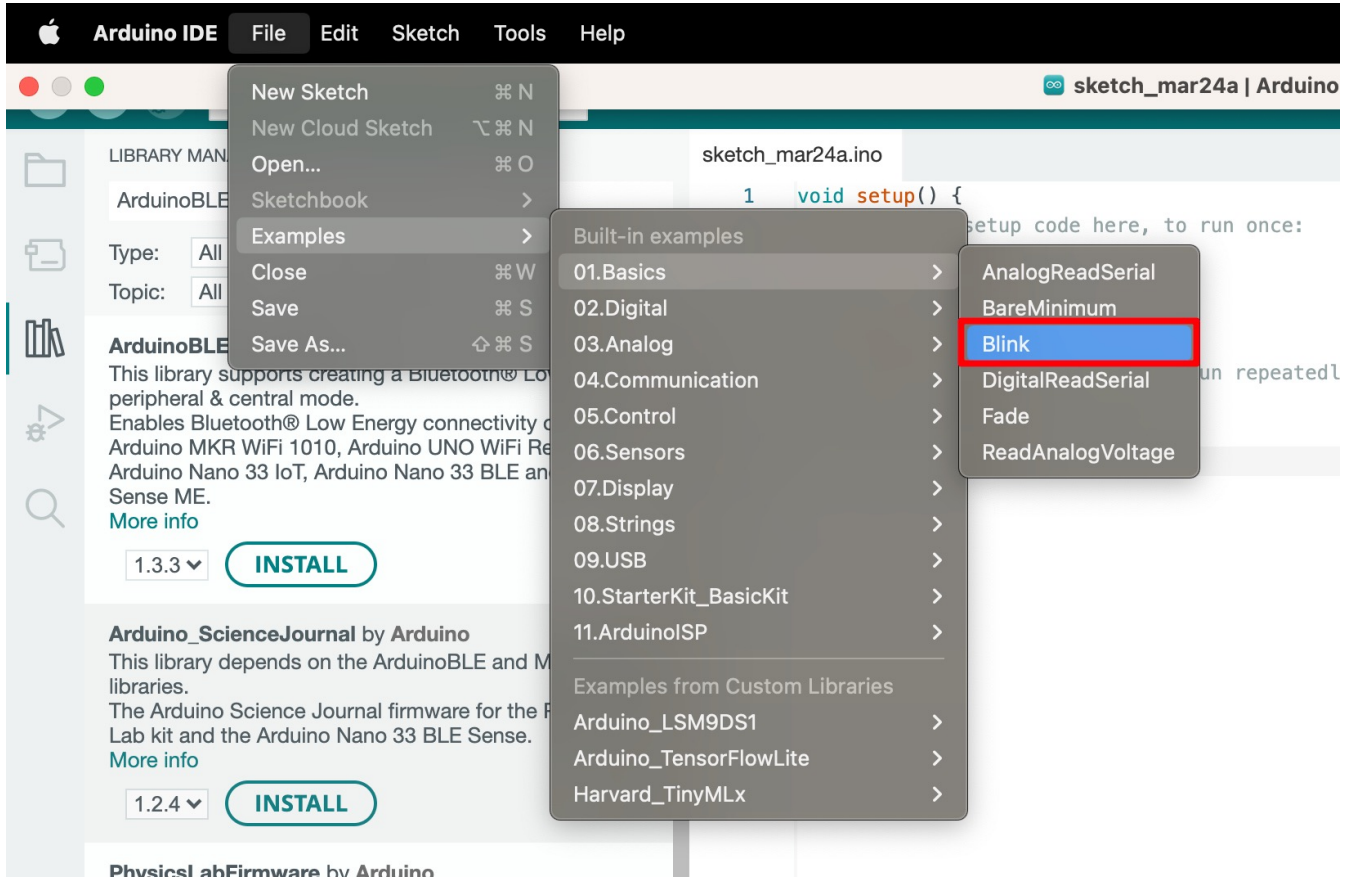
- Setting the preferences:
- show verbose output during: Compile and Upload





The Arduino Blink Example

Open the *Blink.ino* sketch, which you can find via the File drop-down menu. Navigate, as follows: File → Examples → 01.Basics → Blink.



The Arduino Blink Example

```
// the setup function runs once when you press reset or power the board

void setup() {
// initialize digital pin LED_BUILTIN as an output. pinMode(LED_BUILTIN, OUTPUT);

}
// the loop function runs over and over again forever

void loop() {
digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level) delay(1000); // wait for a second
digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW delay(1000); // wait for a second

}
```

The Arduino Blink Example

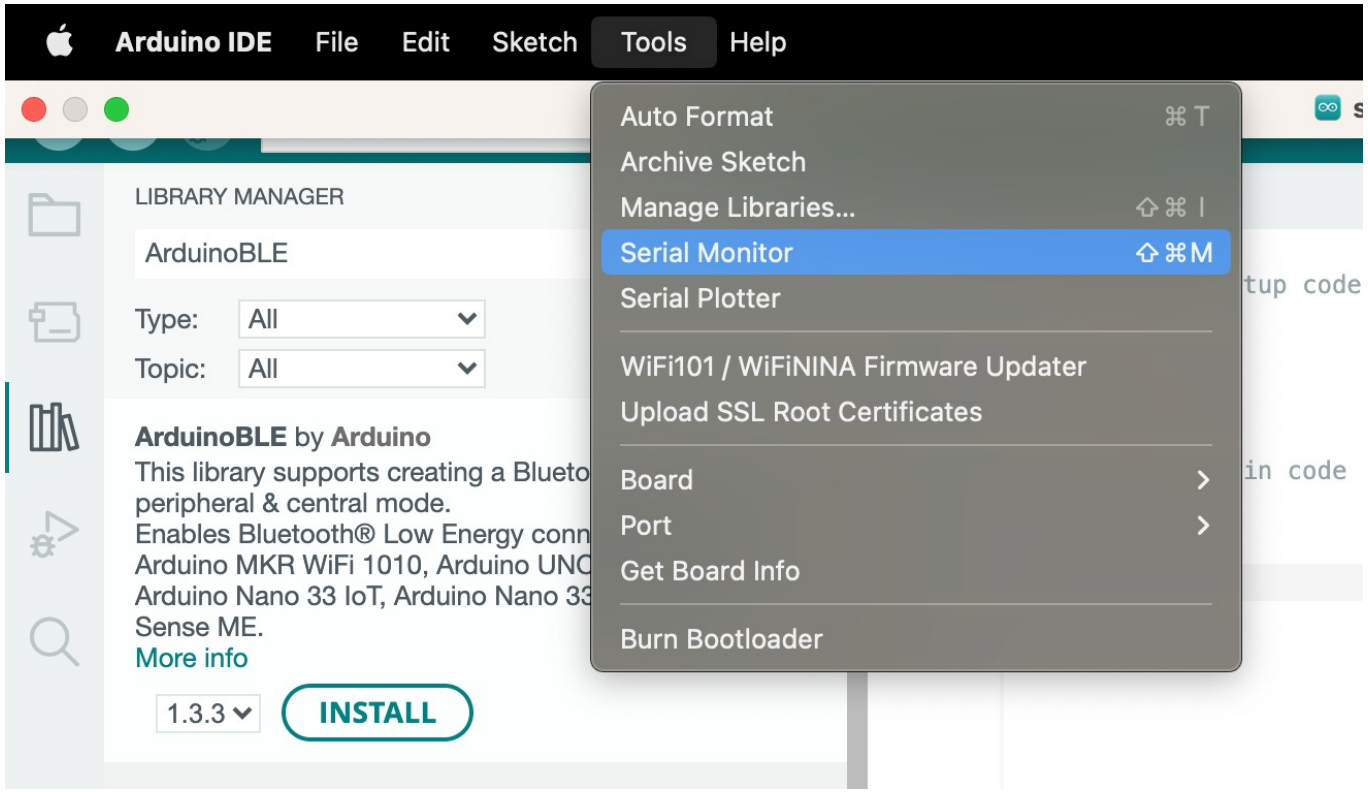

```
[========================     ] 85% (18/21 pages)
[=========================    ] 90% (19/21 pages)
[==========================   ] 95% (20/21 pages)
[============================] 100% (21/21 pages)
Done in 3.342 seconds
```

Press Reset button!

The Arduino Blink Example

The Serial Monitor and Plotter

You can open up the Serial Monitor (and Serial Plotter) by navigating through the menu to Tools → Serial Monitor or Tools → Serial Plotter.



Testing the Microphone

1. Use a USB cable to connect the Arduino Nano 33 BLE Sense to your machine. You should see the green LED power indicator come on when the board first receives power

2. Open the **test_microphone.ino sketch**, which you can find via the File drop-down menu. Navigate, as follows: **File → Examples → Harvard_TinyMLx → test_microphone**.
   - Note that this library could be found as: **File → Examples → INCOMPATIBLE → Harvard_TinyMLx → test_microphone**

3. Select the Arduino Nano 33 BLE as the board by going to **Tools → Board: <Current Board Name> → Arduino Mbed OS Boards (nRF52840) → Arduino Nano 33 BLE**

Testing the Microphone

4. Then select the USB Port associated with your board. This will appear differently on Windows, macOS, Linux but will likely indicate 'Arduino Nano 33 BLE" in parenthesis. You can select this by going to Tools → Port: <Current Port (Board on Port)> → <TBD Based on OS> (Arduino Nano 33 BLE). Where <TBD Based on OS> is most likely to come from the list below where <#> indicates some integer number
   - Windows → COM<#>
   - macOS → /dev/cu.usbmodem<#>
   - Linux → ttyUSB<#> or ttyACM<#>

5. Use the rightward arrow next to the **'compile' checkmark to upload / flash** the code.

Testing the Microphone

6. Open the Serial Monitor. You can accomplish this three different ways as shown below. If the Serial Monitor fails to open check to make sure the appropriate Port is selected. Sometimes the port is reset during the upload process.
   - Use the menu to select: **Tools → Serial Monitor**
   - Click on the **magnifying glass at the top right of the Arduino Desktop IDE**
   - Use the **keyboard shortcut**: **CTRL + SHIFT + M or CMD + SHIFT + M**

```
Welcome to the microphone test for the built-in microphone on the Nano 33 BLE Sense

Use the on-shield button or send the command 'click' to start and stop an audio recording
Open the Serial Plotter to view the corresponding waveform
```

Testing the Microphone

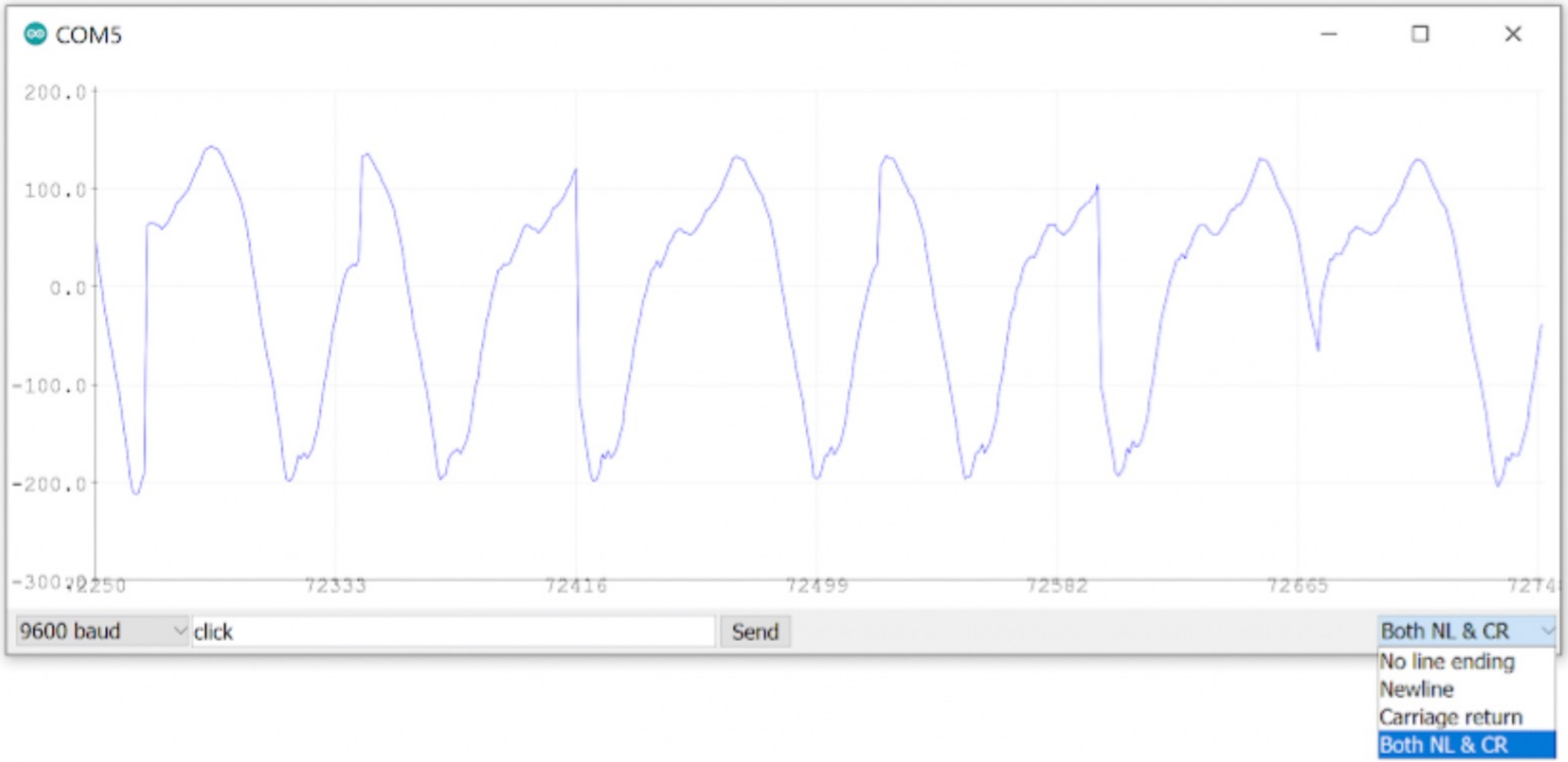
7. Press the on-shield button to start an audio recording. After you do, a stream of data should appear in the Serial Monitor.
8. Press the button again to stop the recording. If the numbers were changing then your microphone is most likely correctly recording audio! Congratulations! If you do not have the shield you can also control the starting and stopping of the waveform by sending the command click through the serial monitor.
9. In the drop down menu that reads **"No line ending"** by default at the bottom right of the Serial Monitor, you need to select **"Both NL & CR"**. This tells the Serial Monitor to send both a **NL = new line character** as well as a **CR = carriage** return character every time you send a message.

Testing the Microphone

10. To help visualize this a little better we are going to use the Serial Plotter. Note that **you cannot have both the Serial Monitor and Serial Plotter open at the same** time as the Arduino can only communicate over a single serial port. Importantly, this also means that you cannot upload new code to the Arduino if either the Serial Monitor or Plotter is open! You can open the Serial Plotter in two ways:
    - Use the menu to select: **Tools → Serial Plotter**
    - Use the keyboard shortcut: **CTRL + SHIFT + L or CMD + SHIFT + L**

11. Perform the same experiment, **press the on-shield button** (or send the serial command click) to **start an audio recording and again to stop it.**

Testing the Microphone



Testing the Inertial Measurement Unit

1. Open the **test_IMU.ino sketch**, which you can find via the File drop-down menu Navigate, as follows: **File → Examples → Harvard_TinyMLx → test_IMU**
2. Use the Tools drop down menu to select appropriate Port and Board
3. Then use the rightward arrow next to the 'compile' checkmark to upload / flash the code
   If you get **the error "Arduino_LSM9DS1.h: No such file or directory"** then please **go back to Setting up the Software and make sure you install the accelerometer, magnetometry, and gyroscope library!**
4. Open the Serial Monitor and you should see the following.

```
Welcome to the IMU test for the built-in IMU on the Nano 33 BLE Sense

Available commands:
a - display accelerometer readings in g's in x, y, and z directions
g - display gyroscope readings in deg/s in x, y, and z directions
m - display magnetometer readings in uT in x, y, and z directions
```

Testing the Inertial Measurement Unit

5. As a **reminder** you can **open the serial monitor in one of three ways shown below**. If the **Serial Monitor fails to open check to make sure the appropriate Port is selected**. Sometimes the port is reset during the upload process
   - Use the menu to select: **Tools → Serial Monitor**
   - Click on the magnifying glass at the top right of the Arduino Desktop IDE
   - Use the keyboard shortcut: **CTRL + SHIFT + M or CMD + SHIFT + M** depending on OS

6. Like with the microphone, in the drop-down menu that reads “No line ending” by default at the **bottom right of the Serial Monitor**, you need to **select** “**Both NL & CR**”

7. Now you can enter one of the **possible arguments** (**a, g, or m**) into the text entry bar across the top of the Serial Monitor and click “**Send**”

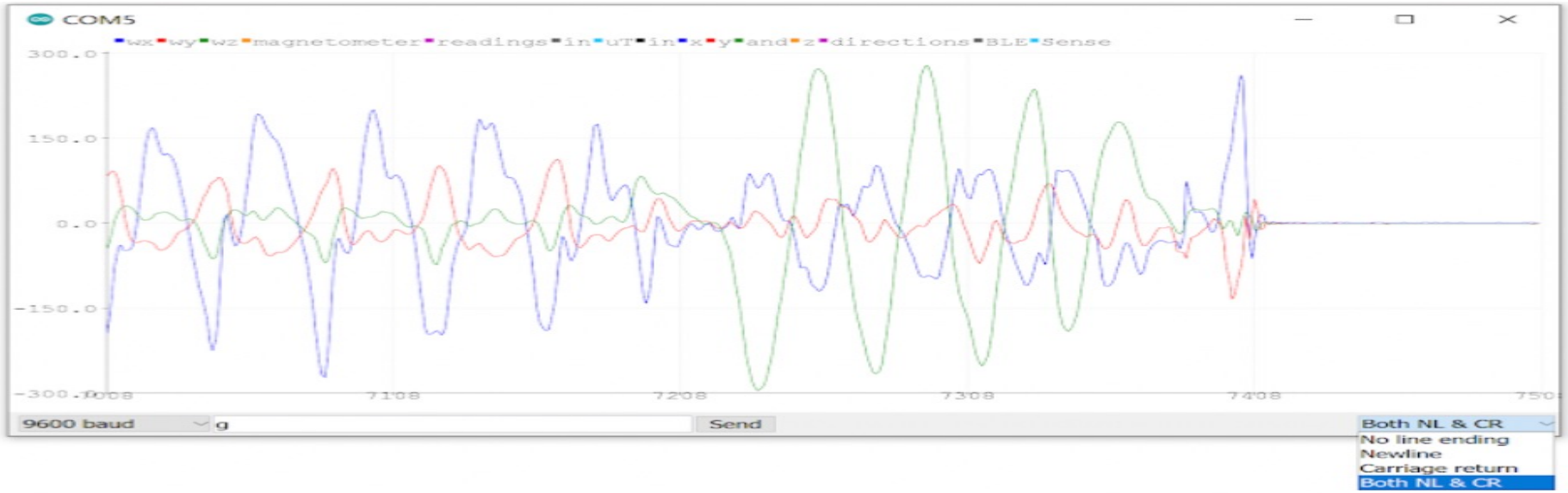
Testing the Inertial Measurement Unit

8. As mentioned in the Serial Monitor this will start to print the data coming from the **[a]ccelerometer** (computing acceleration in the x, y, or z direction), the **[g]yroscope** (computing the angular velocity -- the change in the roll, pith, and yaw), or the **[m]agnetometer** (computing the magnetic forces present on the IMU). Depending on your selection, you should now see a stream of corresponding data. However, like with the microphone this data is hard to interpret in raw form. So instead lets us the Serial Plotter

9. Close the Serial Monitor and open the Serial Plotter. As a reminder you can open the Serial Plotter in two ways:
   - Use the menu to select: **Tools → Serial Plotter**
   - Use the keyboard shortcut: CTRL + SHIFT + L or CMD + SHIFT + L (depending on OS)

Testing the Inertial Measurement Unit

10. With the Plotter open, you should see the same stream of data presented graphically. The most interesting one to plot is the **[g]yrocope**. Again, make sure you have selected "**Both NL & CR**" and **type "g"** into the **text entry box** (now at the bottom of the window) and **click** "**Send**."



11. Move the board around and observe the response. Can you figure out which axis of rotation is the x, the y, and the z (also known as roll, pitch, and yaw)? By moving the board around in different directions, you get different responses from the various lines in the Serial Plotter

Testing the OV7675 Camera

1. Open the test_camera.ino sketch, which you can find via the File drop-down menu. Navigate, as follows: **File → Examples → Harvard_TinyMLx → test_camera.**
2. As always, use the Tools drop down menu to select appropriate Port and Board.
3. Then use the rightward arrow next to the 'compile' checkmark to upload / flash the code. If you get the error "Arduino_OV767X.h: No such file or directory" then please go back to Setting up the Software and make sure you install the accelerometer, magnetometry, and gyroscope library! To help you debug please check out our FAQ appendix with answers to the most common errors! Note: if you get an error message that contains something like **'OV7675' was not declared in this scope** this means you already had a conflicting copy of the Arduino_OV767X library installed on your system. We have bundled a forked copy of it with our library so please uninstall the conflicting version.
4. If you get a message that "Your board was not found", this could be due that the MCU was "busy" running the previous code. Click the Reset button twice (you will see the Build in LED blinking slowly). Upload the sketch again.

Testing the OV 7675 Camera

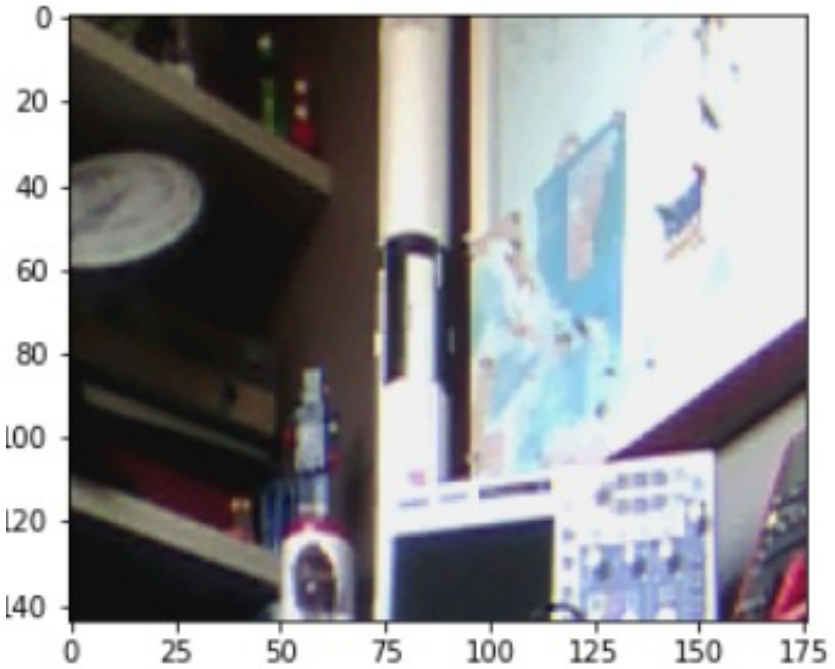
5. Open the Serial Monitor you should see:

```
Welcome to the OV7675 test
Available commands:
single - take a single image and print out the hexadecimal for each pixel (default)
live - the raw bytes of images will be streamed live over the serial port
capture - when in single mode, initiates an image capture
```

6. Type “single” in the text entry field and press send or hit the enter / return key. The camera is now primed to take a picture. Now you can either text “capture” or press the on-shield button to take a selfie (or photo). To get the photo oriented correctly make sure the “OV7675” text on the camera module (or the “Tiny Machine Learning Shield” on the shield) is oriented such that it would be readable by the subject of the photo (e.g., you if you are taking a selfie). **The camera does not have a large field of view so if you are taking a selfie make sure to hold the camera far away from your face.**

Testing the OV 7675 Camera

7. To view your image you will need to copy the sequence of hexadecimal digits that will print to the Serial Monitor and paste them into the notebook: **OV7675_Image_Viewer.ipynb.**



Lab 1 References

1) https://tinyml.seas.harvard.edu/assets/other/4D/22.03.11_Marcelo_Rovai_Handout.pdf

2) https://www.arduino.cc/en/software