

No-code Programming to

Get Started with TinyML

Designed to introduce beginners to the basics of Embedded Machine Learning with Wio Terminal and Codecraft graphical programming





Catalogue

Overview

Lesson 01	Introduction to TinyML using Wio Terminal and Codecraft	8
Lesson 02	Motion Recognition using built-in Accelerometer	12
Lesson 03	Gestures Recognition using built-in Light Sensor	32
Lesson 04	Wake-up Words Recognition using built-in Microphone	48
Lesson 05	Smell Recognition using Grove-Multichannel Gas Sensor	66
	More lessons coming soon	80

Afterword

Basic Course Information

Project Steps Course name: TinyML with Wio Terminal Free Course for Beginners

Course introduction

Learn how to train and Deep Neural Network models on Cortex–M core microcontroller devices like Wio Terminal using a graphical programming tool known as Codecraft. Course content features seven detailed step–by–step projects that allow the students to grasp basic ideas about modern Machine Learning which it can be used in microcontrollers with low–power consumption and smaller footprint to create intelligent, connected systems.

Course Overview

Course Description

Learn how to train and deploy deep neural network models on Cortex–M core microcontroller devices like Wio Terminal using a graphical programming tool known as Codecraft. Course content features seven detailed step–by–step projects that allow the students to grasp basic ideas about modern Machine Learning and how it can be used in microcontrollers with low–power consumption and smaller footprint to create intelligent, connected systems. After completing the course, students will be able to design and implement their own Machine Learning enabled projects on Cortex–M core microcontrollers starting from defining a problem to gathering data, training neural network models. Finally deploying it to the device to display inferred results or control other hardware appliances based on inferred data.

Course contents are based on the use of Codecraft which simplifies data collection, model training, and conversion pipeline.

This course does not require knowledge of programming or electronics. It will take you step–by–step through the necessary knowledge and quickly put it into practice in each project.



The general steps of each project are as follows:

1. Project Overview: An introduction to the project objectives to be accomplished in the lesson and the results to be achieved.

2. Background knowledge: The lesson will begin with an introduction to the new hardware and its electrical knowledge.
3. Practices: a. Model Creation b. Data Acquisition c. Training & Deployment d. Programming
4. ML Theory

The specific content includes:

1. Introduction to TinyML with Wio Terminal in Codecraft
2. Motion Recognition by using built–in accelerometer
3. Gestures Recognition by using built–in light sensor
4. Wake–up Words Recognition by using built–in microphone
5. Smell recognition by using Grove–Multichannel Gas Sensor
6. Sport recognition by using built–in accelerometer
7. Barcode recognition by using built–in light sensor
8. Face Recognition by using thermal imaging sensor
9. Creative projects and Summarization

ML knowledge:

1. Understand your input: input labels, dataset.
2. Understand your output: output, training performance.
3. Know the different model scales.
4. Know the Hyperparameter.
5. Model assessments.
6. Learn to improve the training performance.
7. Advanced knowledge of neural network: layers and their details.

Course Product Requirements

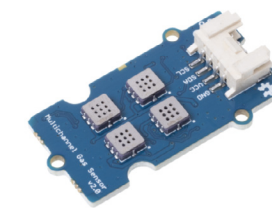
Hardware requirements (boards, modules):



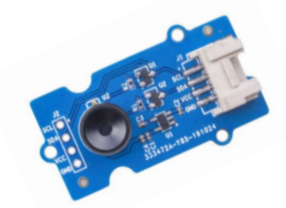
• Wio Terminal x1



• Grove cables x4



• Grove – Multichannel Gas Sensor v2



• Grove – Thermal Imaging Camera

Software requirements: Codecraft



<https://ide.tinkergen.com/>

Curriculum outline

No.	Name	Overview	Hardware
Getting started			
1	Introduction to TinyML with Wio Terminal in Codecraft	<ul style="list-style-type: none"> The basic theory of TinyML Introduction of Wio Terminal and grove Get started with Codecraft 	Wio Terminal
Projects			
2	Motion Recognition by using built-in accelerometer	<ul style="list-style-type: none"> Theory: Introduction of the accelerometer Practice: <ul style="list-style-type: none"> Model creation Data Acquisition Training & Deployment Programming ML Theory(Understand your input) 	Wio Terminal
3	Gestures Recognition by using built-in light sensor	<ul style="list-style-type: none"> Theory: Introduction of the light sensor Practice: <ul style="list-style-type: none"> Model creation Data Acquisition Training & Deployment Programming ML Theory(Understand your output) 	Wio Terminal
4	Wake-up Words Recognition by using built-in microphone	<ul style="list-style-type: none"> Theory: Introduction of the microphone Practice: <ul style="list-style-type: none"> Model creation Data Acquisition Training & Deployment Programming ML Theory (Model scale:SMALL,MEDIUM and LARGE) 	Wio Terminal

5	Smell recognition by using Grove-Multichannel Gas Sensor	<ul style="list-style-type: none"> Theory: Introduction of the Multi-channel Gas Sensor Practice: <ul style="list-style-type: none"> Model creation Data Acquisition Training & Deployment Programming ML Theory (Hyperparameter) 	Wio Terminal Grove - Multichannel Gas Sensor v2
Advanced Projects			
6	Sport recognition by using built-in accelerometer	<ul style="list-style-type: none"> Theory: Introduction of sport recognition Practice: <ul style="list-style-type: none"> Model creation Data Acquisition Training & Deployment Programming ML Theory (Model assessments) 	Wio Terminal
7	Barcode recognition by using built-in light sensor	<ul style="list-style-type: none"> Theory: Introduction of the barcode Practice: <ul style="list-style-type: none"> Model creation Data Acquisition Training & Deployment Programming ML Theory (Improve model) 	Wio Terminal
8	Face Recognition with thermal imaging sensor	<ul style="list-style-type: none"> Theory: Introduction of thermal imaging sensor,Face Recognition Practice: <ul style="list-style-type: none"> Model creation Data Acquisition Training & Deployment Programming ML Theory (Details of layers) 	Wio Terminal Grove - Thermal Imaging Camera (MLX90640)
Summarization			
9	Creative projects and Summarization	<ul style="list-style-type: none"> Summarization of ML theory Examples of creative projects 	

Lesson 01

Introduction to TinyML using
Wio Terminal and Codecraft

Wio Terminal

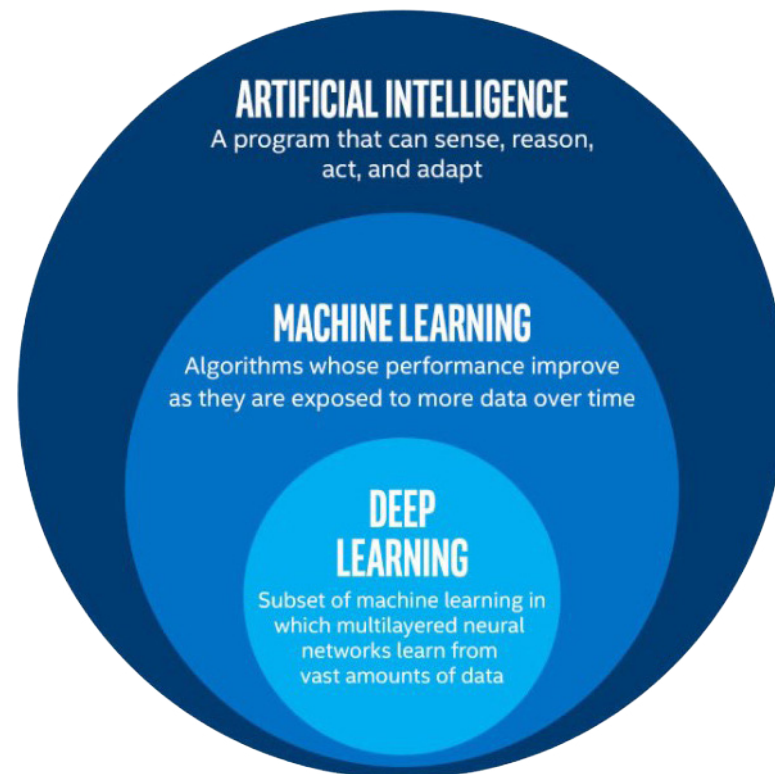
Codecraft



Theory

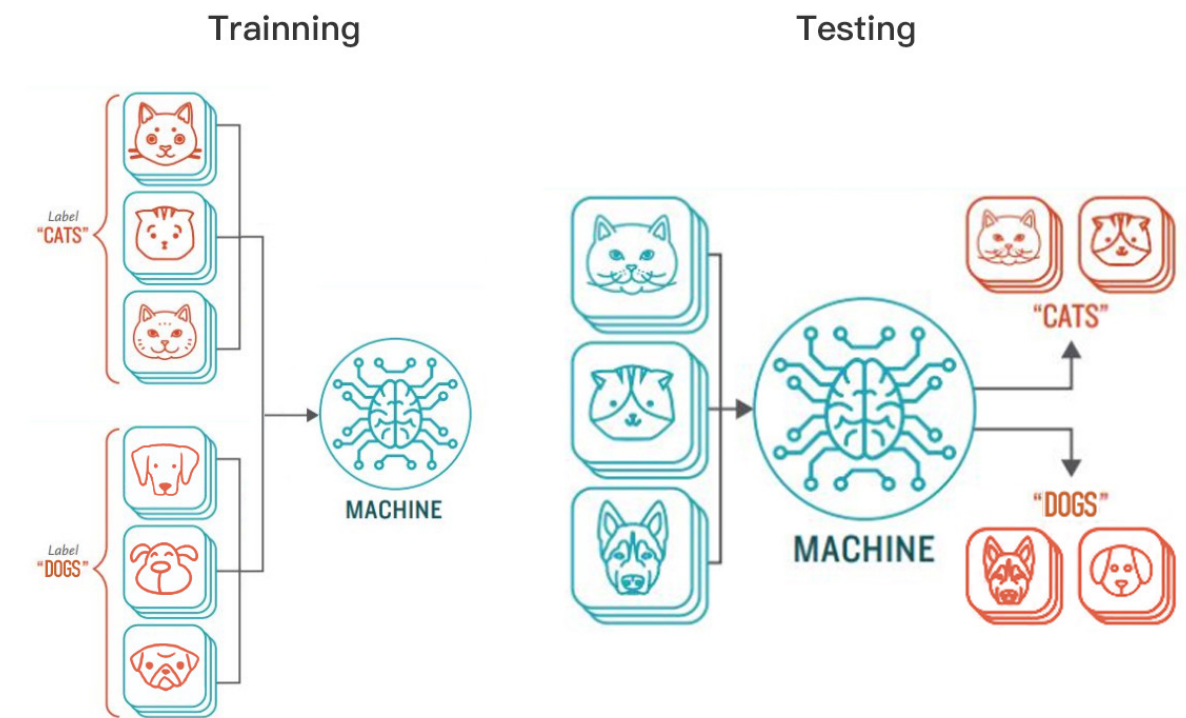
Machine Learning and Deep Learning

Machine learning is a branch of Artificial Intelligence (AI) that focuses on building applications that learn from data and improve their accuracy over time. This process is carried out without using traditional rule-based programming. Machine learning enables machines to work by observing, classifying, and learning from mistakes just like human beings. Deep learning is a subset of machine learning that uses deep artificial neural networks (hence the name) to learn from large amounts of data.



Artificial Neural Networks (ANNs) are the network of neurons that attempt to simulate a composition of human brain. ANNs are created by computer programming which behaves like the brain cells that connect to each other.

In order to learn ANNs, you will need to prepare a tremendous amount of information which is called “Train Set”. When you are trying to teach an ANN how to distinguish between a cat and a dog the train set would provide thousands of images tagged by dogs and cats that allow the neural network to learn. Once a neural network is trained with a significant amount of data, it will be able to classify the coming data based on what it taught to be to see (or hear, depending on the data set) throughout the different units. During the training process, the output of the machine is compared to the human-provided description of what should be observed. If they are matched, the machine is validated. If it’s incorrect, it uses Back Propagation to adjust its learning. Back Propagation is the process of going back through the layers to tweak the mathematical equation. Known as “Deep Learning”, this is what makes a network intelligent.



What is TinyML and why is TinyML important?

Normally, Deep Neural Networks require rather powerful computing resources to be trained and deployed. And yet, a branch of Machine Learning on the Edge or Embedded Machine Learning called TinyML has appeared recently. It represents a technique (or field of study) in machine learning and embedded systems that explores which machine-learning applications (once reduced, optimized, and integrated) can be run on the devices as small as microcontrollers. The ML, as you might have guessed stands for Machine Learning. Tiny in TinyML means that the ML models are optimized to run on very low-power and small-footprint devices such as various microcontrollers (MCUs).



Embedded devices come in all sorts of shapes and sizes, starting from “embedded supercomputer” Nvidia Jetson Xavier AGX to the tiniest of microcontrollers, for example, ESP32 or Cortex M0.



The following figure shows the GeeekNET ESP 32 development board.



Why embedded ML on microcontrollers is classified in a special category and even given its own cool name?

The answer is that it comes with its own set of advantages and limitations. The attraction of TinyML is in fact that MCUs are ubiquitous, small, consume small amounts of energy, and are comparatively cheap. Take ARM Cortex M0+ and the little Seeeduno XIAO board which is built around it – the board is as small as a thumb (20×17.5mm) and it consumes only 1.33 mAh of power (which means it can work ~112 hours on a 150 mA battery). (This time can be increased even further if the device is put in deep sleep). It costs as little as 5 USD.



Thanks to recent improvements in model optimization and the emergence of frameworks specifically created for running machine learning model inference on microcontrollers. It has become possible to give more intelligence to these tiny devices. We now can deploy neural networks on microcontrollers for **audio scene recognition** (for example, elephant activity or sound of breaking glass), **hot-word detection** (to activate device with a specific phrase) or even for simple **image recognition** tasks. The devices with embedded microcontrollers can be used to give new life and meaning to old sensors.

With Codecraft and Wio Terminal, it is now possible to experience the entire process of embedded machine learning without having to deal with a complex programming environment and extensive programming knowledge.

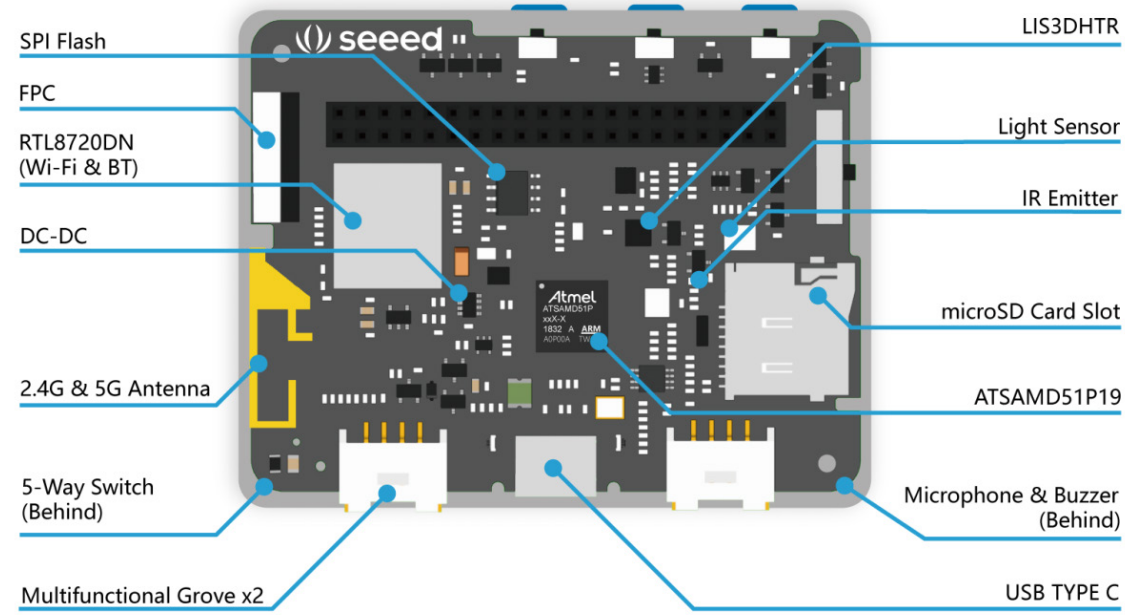
✓ Preparation

Wio Terminal

Wio Terminal is the perfect tool for getting started with IoT and TinyML. It is designed and produced by Seeed Studio and is well supported by various TinyML reasoning frameworks.



The Wio Terminal is using an ATSAM51P19 microcontroller with ARM Cortex-M4F running at 120MHz (boost up to 200MHz), 4MB of external flash memory, and 192KB of RAM. Wireless connectivity with Realtek RTL8720DN support. It is compatible with Arduino and MicroPython. It supports both Bluetooth and Wi-Fi providing a solid foundation for IoT projects. There is a 2.4-inch LCD screen on Wio Terminal, an on-board IMU (LIS3DHTR), microphone, buzzer, microSD card slot, light sensor, and IR emitter (IR 940nm). Most importantly there are two multi-functional Grove ports onboard for the Grove ecosystem and Raspberry Pi compatible 40-pin GPIO pins for additional add-on support.



Codecraft

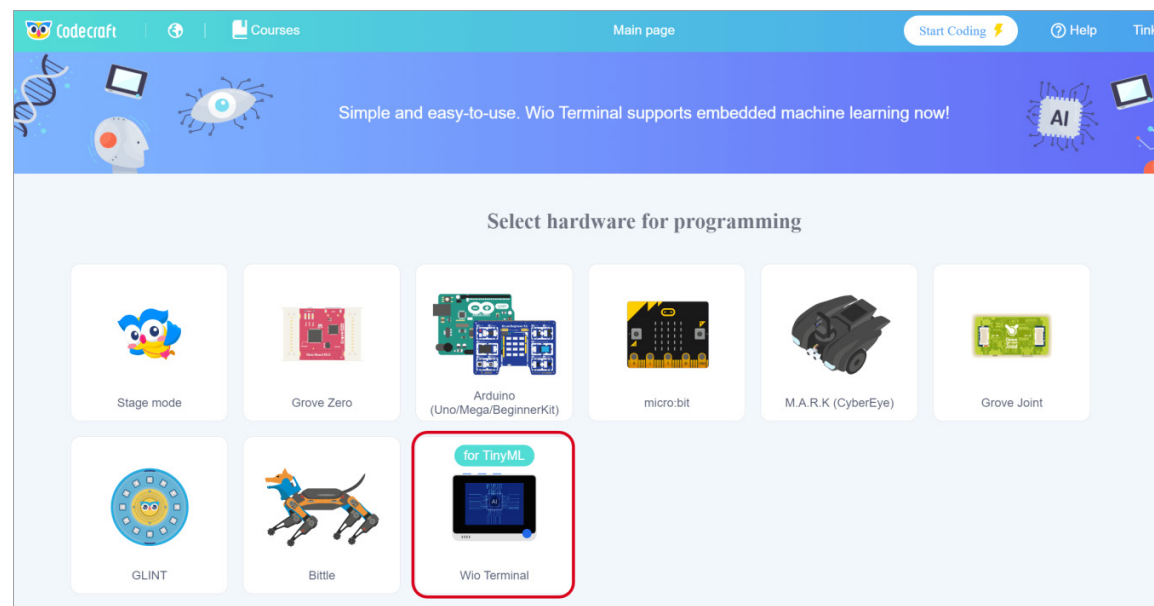
 Please Open <https://ide.tinker.gen.com/>

Software-wise we will be using Codecraft, a graphical programming platform. Powered by Edge Impulse, TinyML is easily accessible for beginners using Codecraft. It provides a beginner-friendly (yet powerful) web interface and toolkit for the whole TinyML pipeline starting from data collection all the way up to model deployment and extensive use.

Codecraft is a user-friendly development platform for machine learning on edge devices.

So, let's quickly go through how to use it to achieve our machine learning journey.

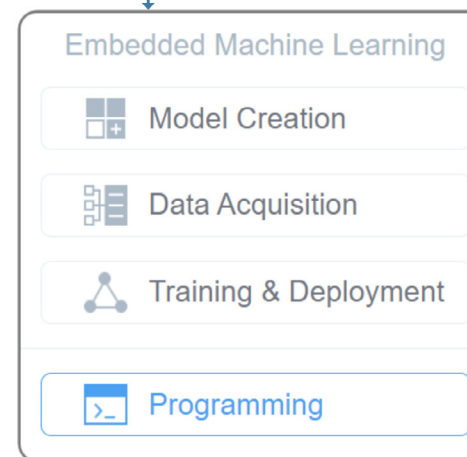
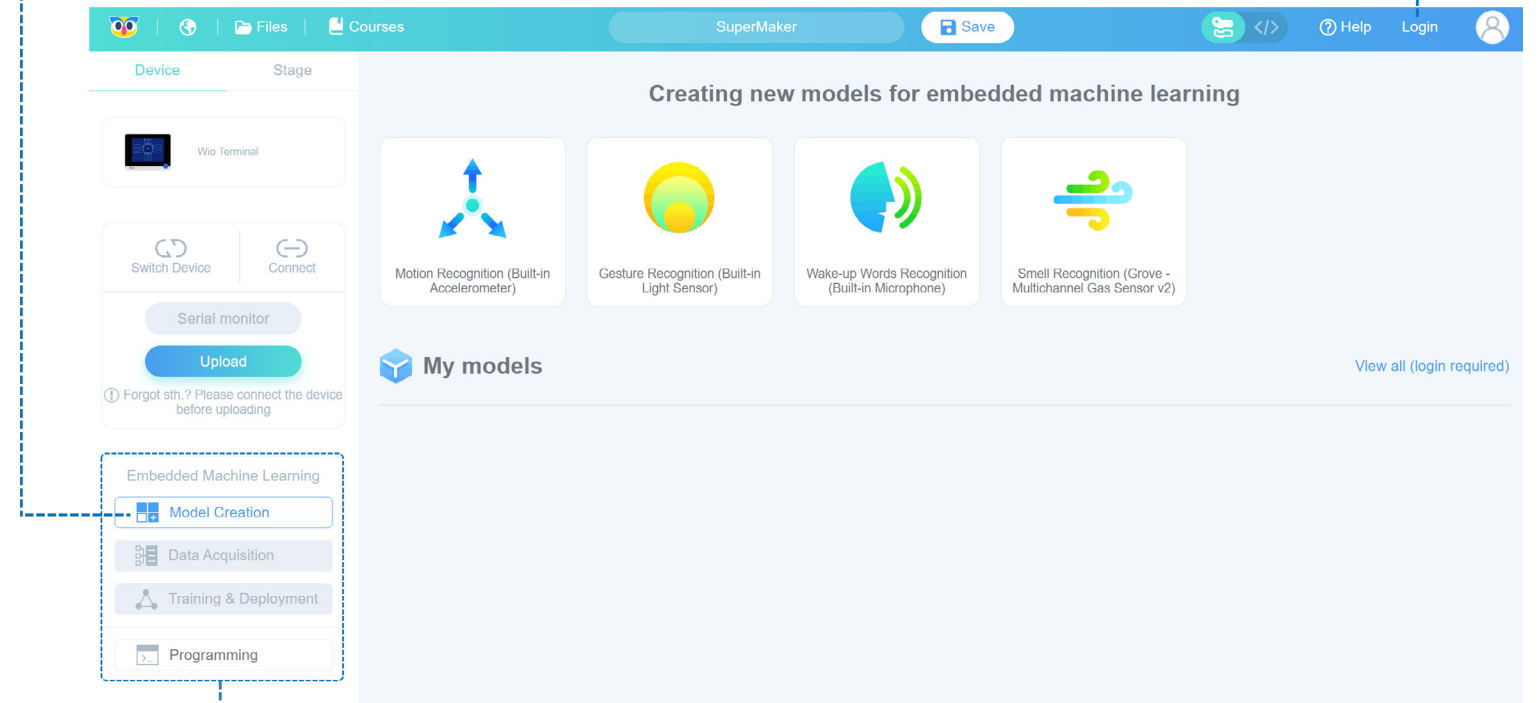
Click the icon of Wio Terminal in Codecraft home screen to enter Wio Terminal's embedded machine learning interface.



Then

1. To create and use models, you need to login (top right corner of the page). Requires Sign-Up for new users.

2. Click "Model Creation" to see the interface "Creating new models for embedded machine learning" where Codecraft provides machine learning frameworks for different types of sensor.



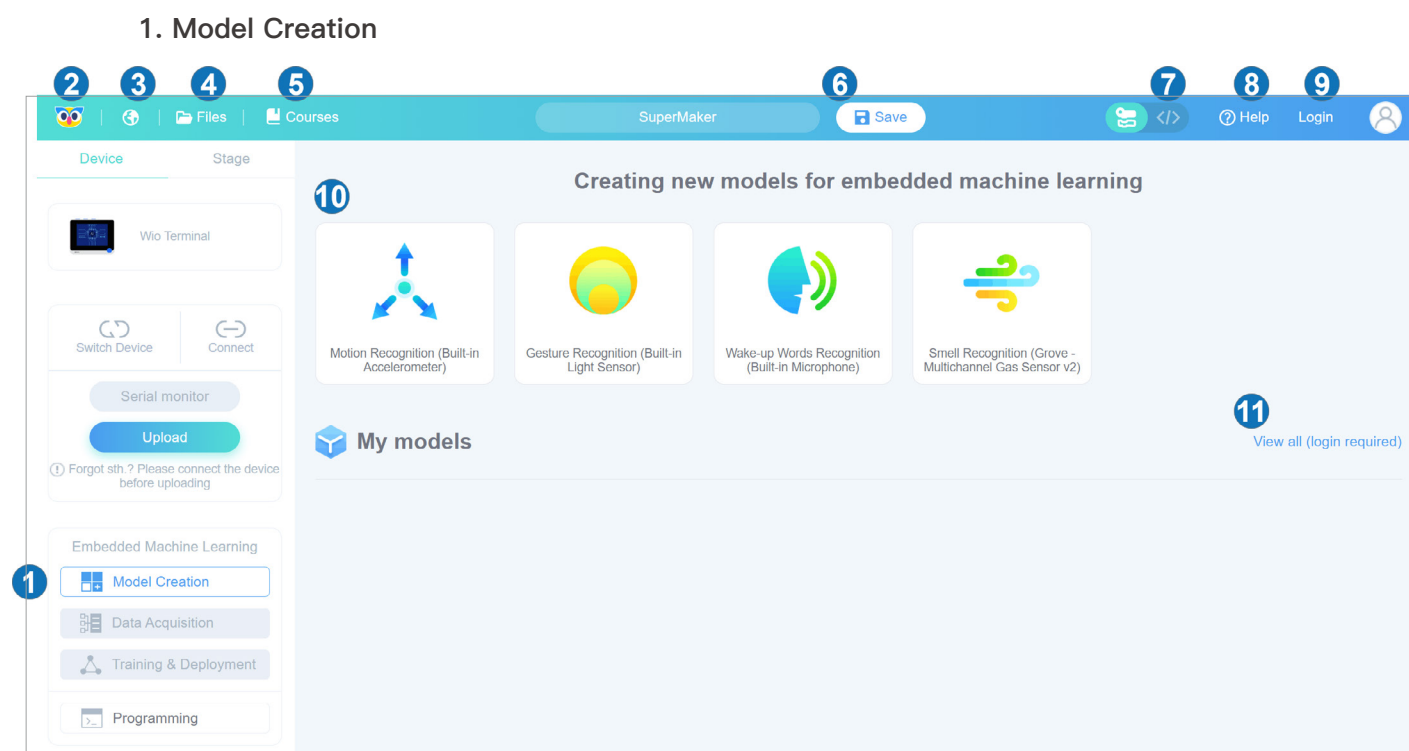
3. All machine learning processes are composed of 4 steps.

① **Model creation:** Create models based on the sensors to be used.

② **Data Acquisition:** Collect data using Wio Terminal.

③ **Training & Deployment:** Train the model with the collected data and deploy the trained model after the completion of training process.

④ **Programming:** Program to use the machine learning models.



1 Model Creation: The first step of TinyML. Click “Model Creation” to see the interface of “Creating new models for embedded machine learning”.

2 Return to Codecraft Home Page: You can click on it to return back to the Codecraft Home Page. If you have not saved the current program, it will prompt you to save it before you leave.

3 Language: Upon clicking this, the language options appear. You can change the language here.

4 Files: You can create a new online project or open a local project. You can save your projects to the cloud from a computer following the on-screen instruction.

5 Course Examples: Here you can check the courses about Codecraft.

6 Save Files Online: You can modify your project names and save them to the cloud. (Codecraft should be connected to the internet and logged in with your Codecraft account).

7 Blocks/Codes: You can toggle between blocks and code. The Programming language being used here is C.

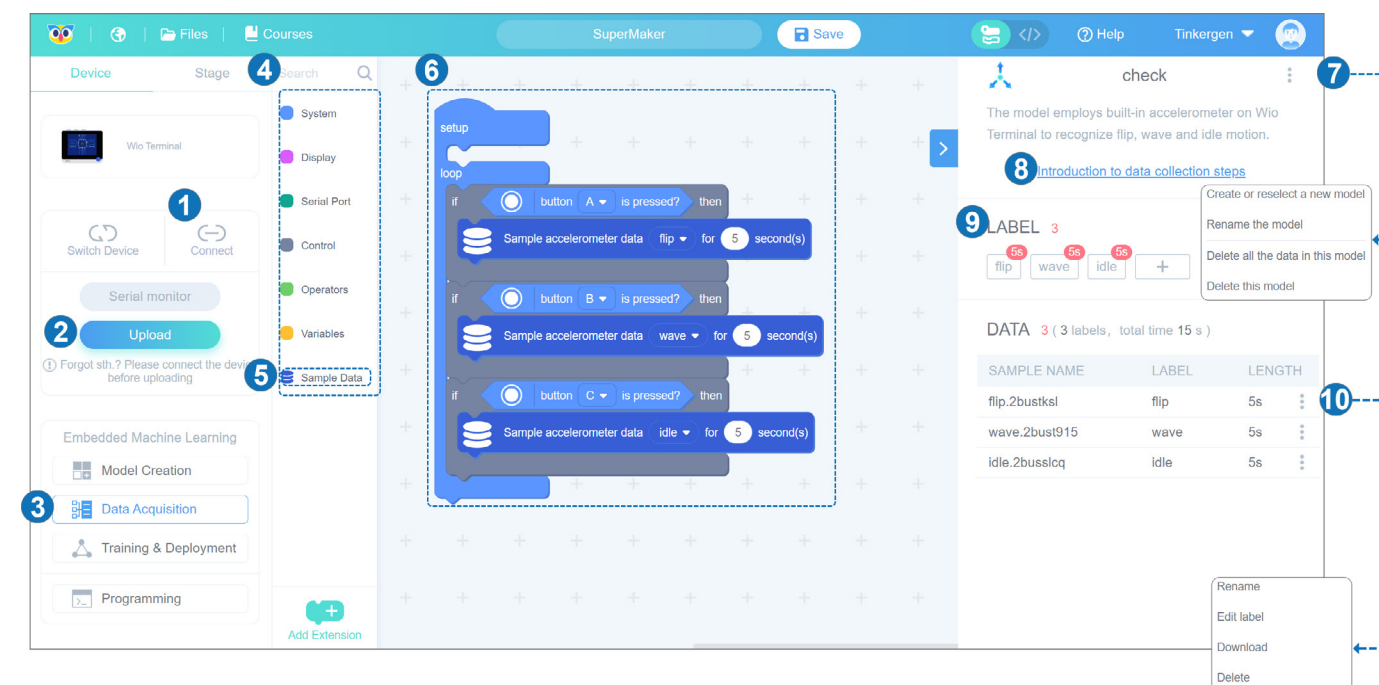
8 Help: We always appreciate your suggestions for Codecraft. We can’t do better without your involvement.

9 Login: You can visit your cloud projects, account settings, my invitation code, and log out. If you are not logged in, it will prompt you.

10 Model framework according to the sensor: You can create a model here by selecting the sensor you are going to use. Click the corresponding icon to create a model using a particular sensor. After creating a new model, the interface will automatically jump to the “Data Acquisition” interface.

11 My models: You can click here and view all the models you have created.

2. Data Acquisition



1 Connect: You can connect your device here.

2 Upload: Click here to upload your program.

3 Data Acquisition: The second step of TinyML.

4 Catalogue Area: You can select the blocks you need by category here.

5 Sample Data blocks: You can get the “default data acquisition program” and blocks related to sample data.

6 Default data acquisition program: You can upload the default data acquisition program directly to start the data acquisition journey.

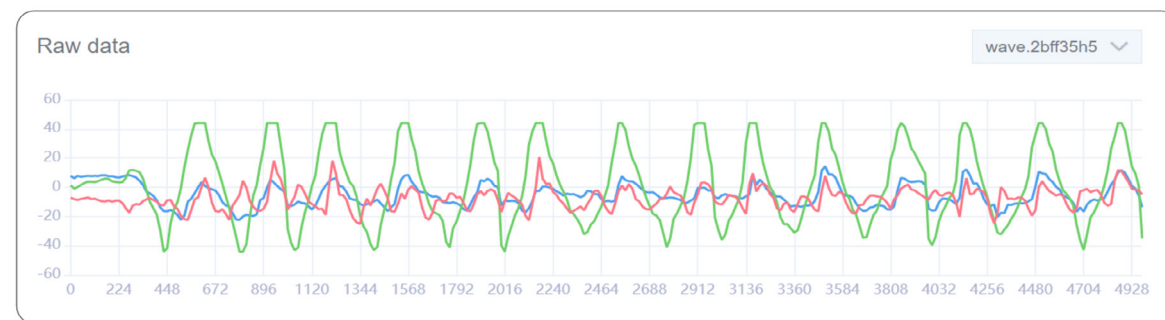
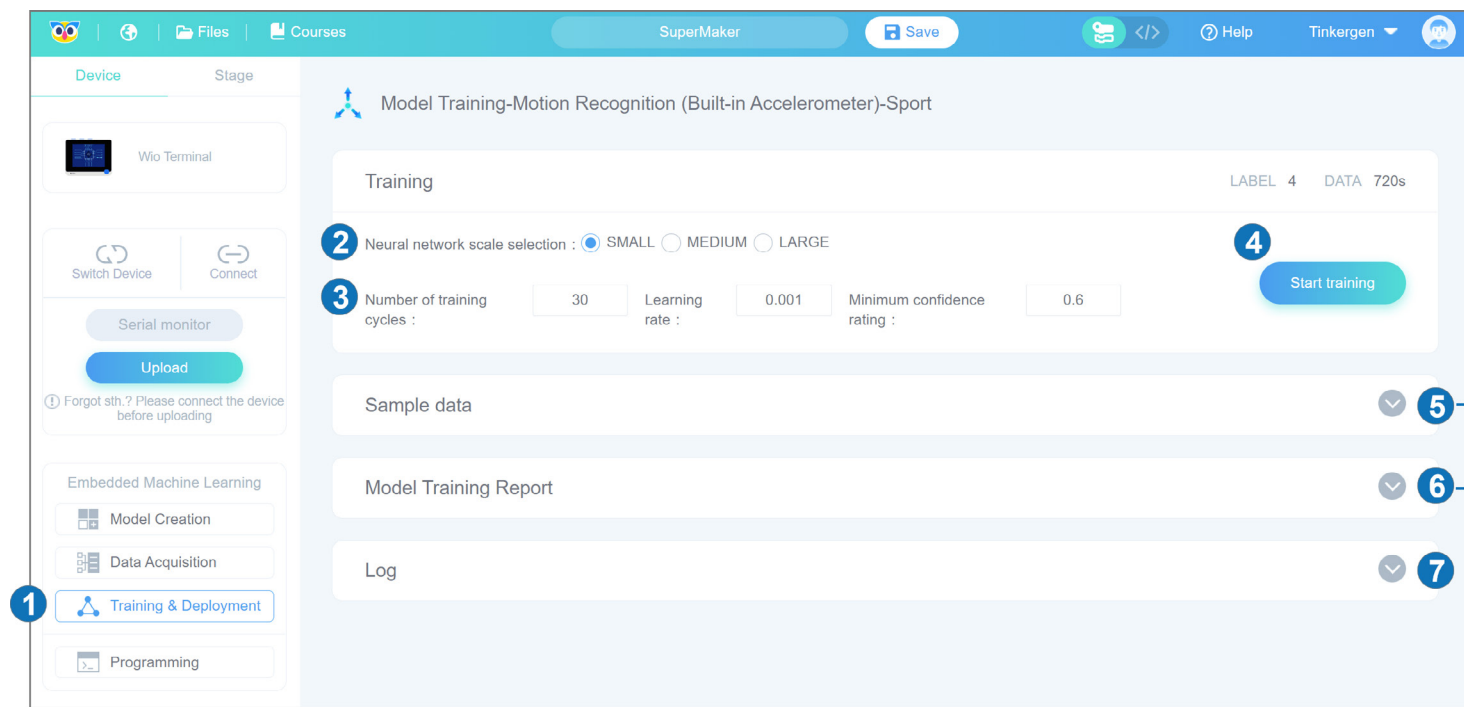
7 Operation on model: Here You can rename and delete the model and its data.

8 Introduction to data collection steps: Clicking here will show a pop-up window with the details of the data collection process.

9 Labels: There are default labels on the right side of the data acquisition interface. You can add and delete labels. To **Add a** custom label: Click “+”. To **Delete** the label: mouse over the label, an “x” will appear in the upper left corner of the label, click the “x” to delete it. To Modify the label: click on the label that will appear to modify the label pop-up window.

10 Operation on data: You can delete, rename and download data here.

3.Training & Deployment:



Model Training Report

Last training performance

Model version: int8

ACCURACY 100.0%

LOSS 0.02

Model deployment

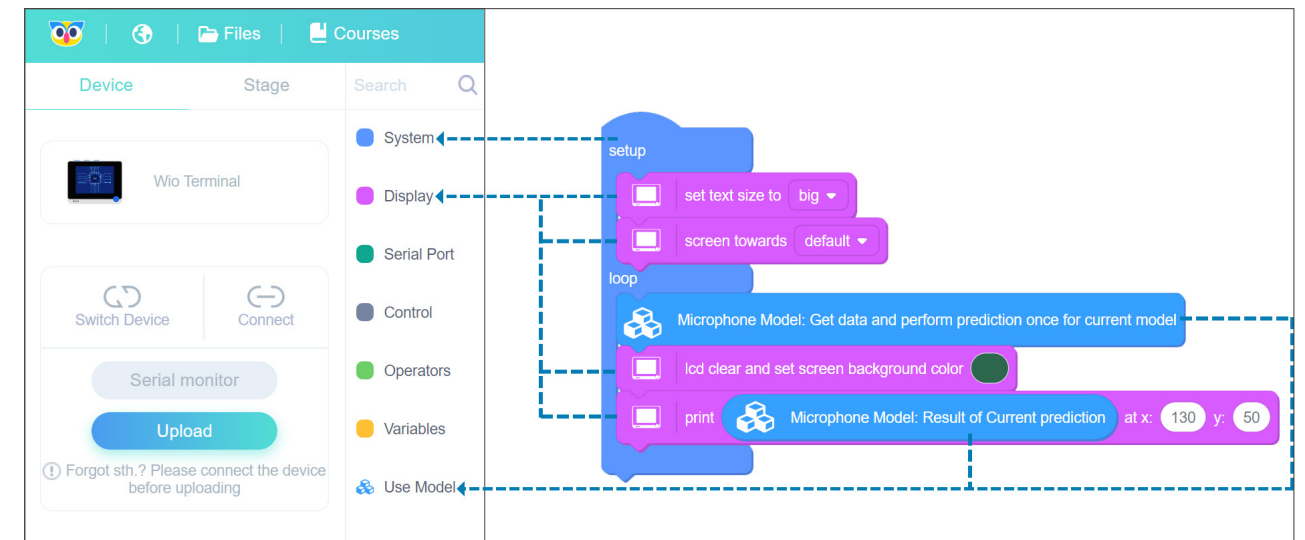
Confusion matrix

	flip	idle	wave
flip	100.0%	0	0
idle	0	100.0%	0
wave	0	0	100.0%
F1 SCORE	1.00	1.00	1.00

- 1 Training & Deployment:** The third step of TinyML.
- 2 Neural network scale:** You can select the suitable neural network size: one of small, medium, and large.
- 3 Parameters:** Codecraft provides default parameter values for each scale of the model. you can set the number of training cycles (positive integer), the learning rate (0~1 number), and the minimum confidence rating(0~1 number) by yourself.

- 4 Start training:** Click “Start Training” to train the model.
- 5 Sample data:** The collected data samples can be observed here.
- 6 Model Training Report:** You can observe the training results including the accuracy, loss, and performance of the model on Wio Terminal. If the training results are not satisfactory, you can go back to the first step of training the model and select another size of neural network or adjust the parameter settings and train again until you get a model with satisfactory results.
- 7 Log:** During training, the “Log” screen displays a training log. By observing the “Log”, you can deduce the model performance.
- 8 Model deployment:** Click here to deploy the ideal model.

4.Programming



You can complete the program by dragging and dropping the blocks. The course also provides the most concise sample code. You can find the corresponding blocks according to the block color.

★ Expansion Tutorials

To help understand the use of embedded machine learning. We provide 7 template projects. You can learn how to use Codecraft for embedded machine learning based on these template model examples. We hope these examples can help you in building more interesting applications.

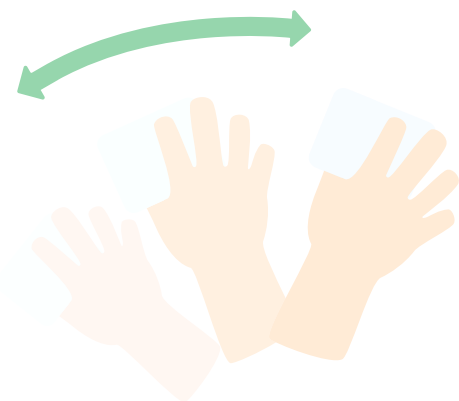
Lesson 02

Motion Recognition using built-in Accelerometer



Project Overview

In this lesson, we'll take on a task of gesture recognition where we will attempt to recognize three gestures (FLIP, WAVE, and IDLE) using Wio Terminal's built-in 3-axis accelerometer.



It is quite hard to accomplish using rule-based programming because gestures are not performed the same way all the time. Trying to solve this problem using traditional Programming requires hundreds of different rules for each mode of action. Even if we consider an idealized situation, we still miss a lot of motions such as speed, angle, and direction. A slight change in these factors requires a new set of rules to be defined but machine learning can handle these variations very easily.

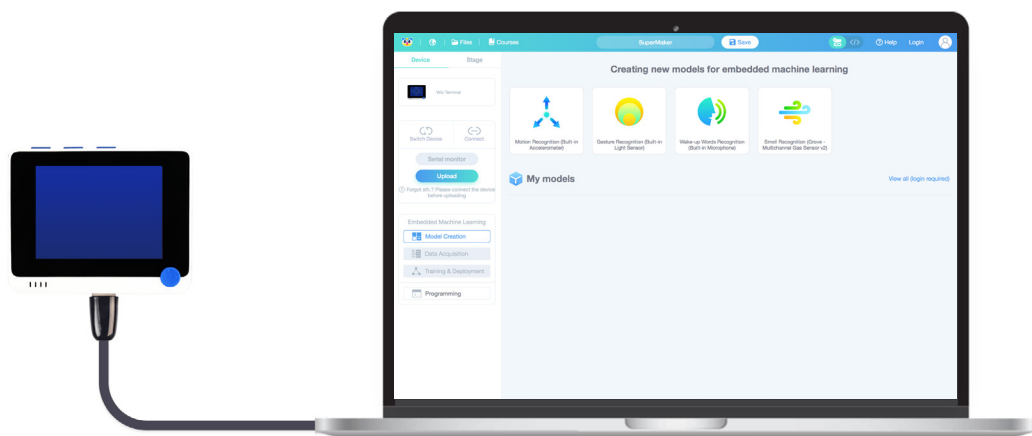
Expected results

Wio Terminal displays the current recognition results for the Wio Terminal's actions in real-time. FLIP, WAVE, and IDLE.



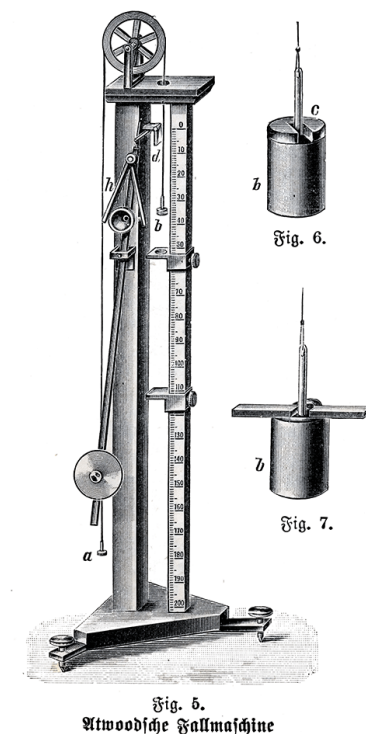
Preparation

Hardware requirements: Wio Terminal
 Connection method:

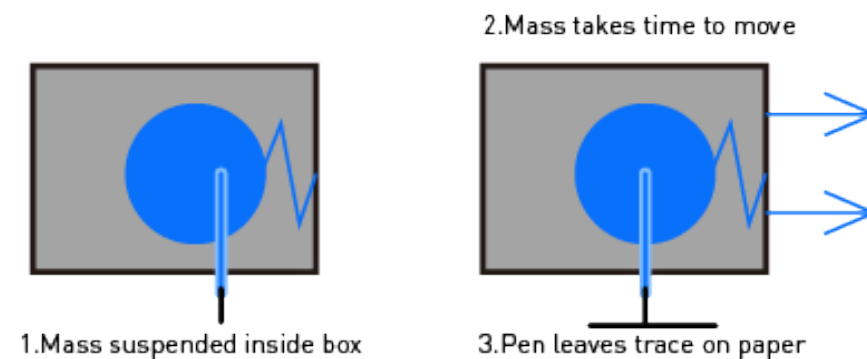


Theory

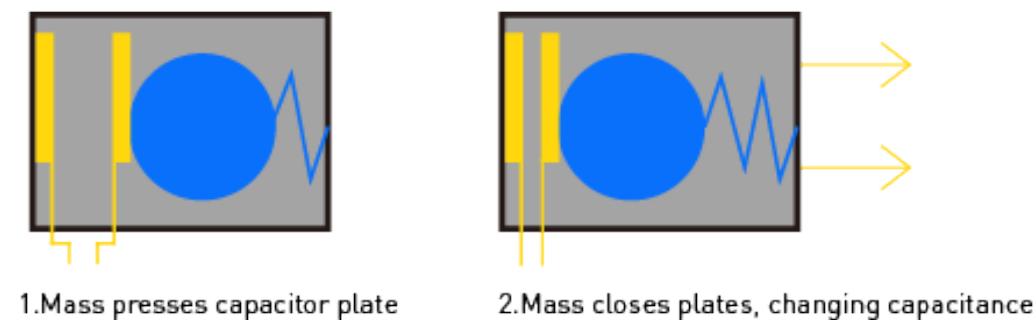
As you might guess from the name, accelerometers are devices that measure the acceleration of a body. It is defined as the rate of change of the velocity of an object. Accelerometers are capable of measuring acceleration either in meters per second squared (m/s^2) or in G-forces (g). A single G-force on Earth is equivalent to $9.8 m/s^2$. There are different kinds of accelerometers. The earliest accelerometers were based on mechanical architecture.



The first accelerometer was called the Atwood machine. It was invented by an English physicist George Atwood.



The accelerometers commonly used in mobile phones are MEMS (Microelectromechanical) accelerometers. The module used in Wio Terminal is called 3-Axis Digital Accelerometer (LIS3DHTR). Generally, the internal structure of accelerometers consists of Capacitive Plates. Some types of accelerometers use fixed capacitive plates, while some of them have the plates attached to minuscule springs that move internally depending upon the acceleration forces acting on the sensor.



Practice

Project Steps

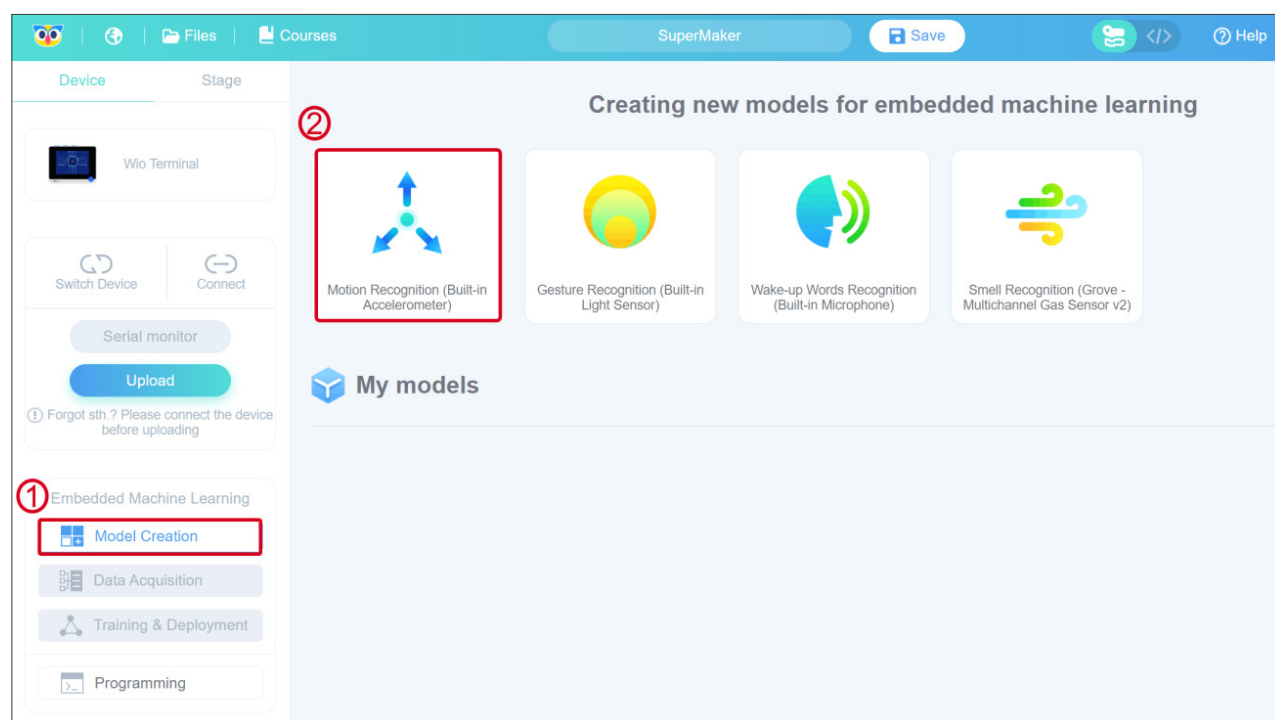
1. Create and Select Models
2. Data Acquisition
3. Training and Deployment
4. Programming

Please Open <https://ide.tinkerforge.com/>

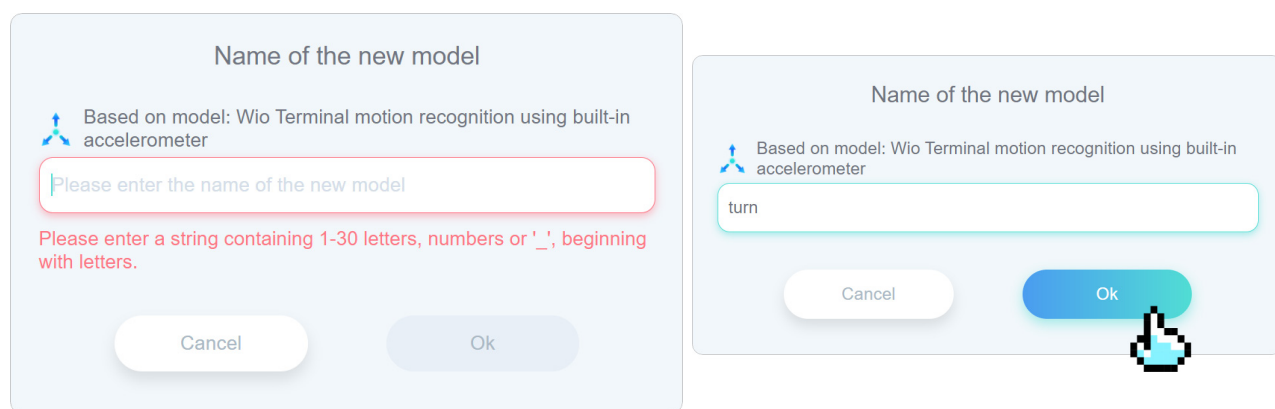
Step 1. Create and select models

1.1 Create the “Motion Recognition(Built-in accelerometer)” model

Click on “Model Creation”, click on “Motion Recognition(Built-in accelerometer)”, as shown in steps 1 and 2 below.



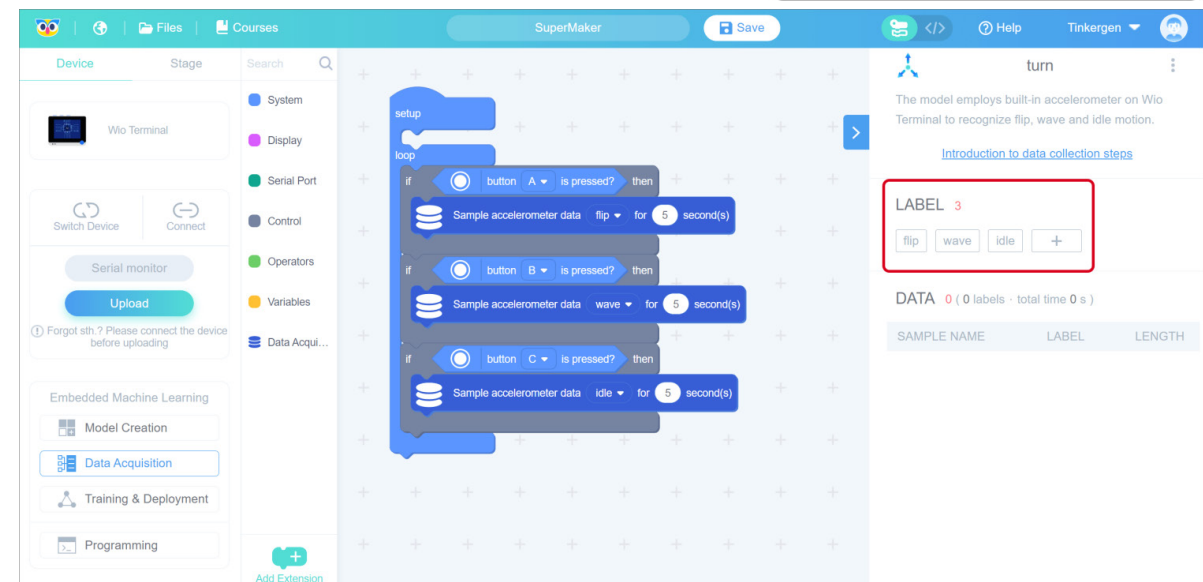
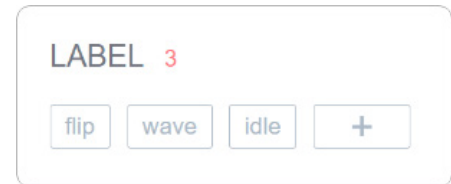
Enter a NAME according to the requirements. Click “OK”



Click “Ok” and it will automatically jump to the Data Acquisition interface.

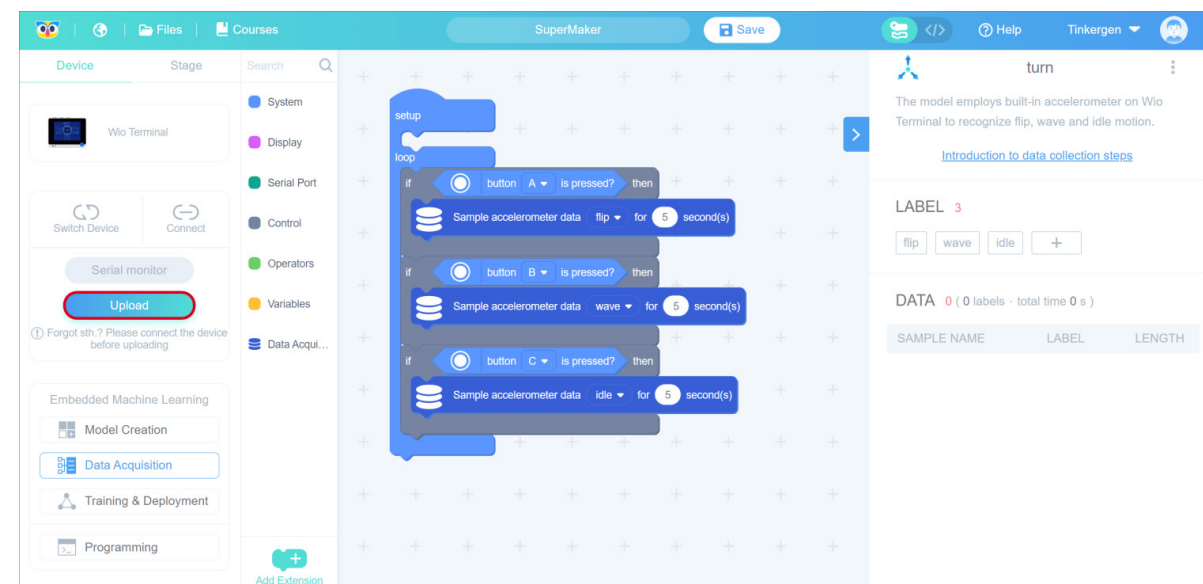
Step 2. Data Acquisition

2.1 Default label



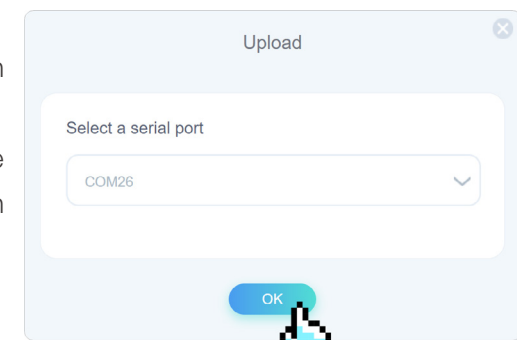
2.2 Connect the device and upload the default data acquisition program in Codecraft

Once the Wio Terminal is connected, in the Codecraft surface, click **Upload** as shown in the figure. This action will upload the default data acquisition program.

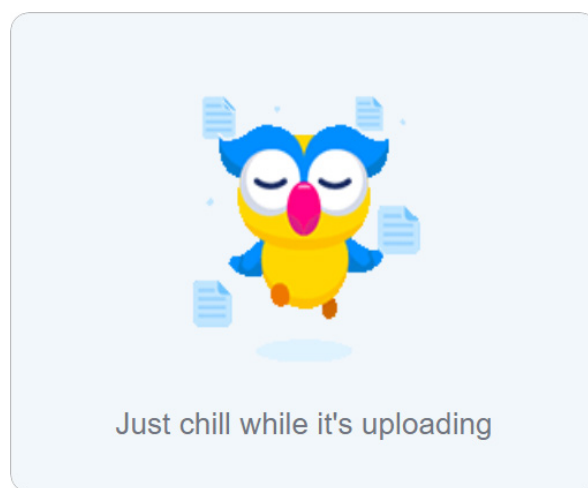


The “Upload” pop-up window will appear as shown in the figure below.

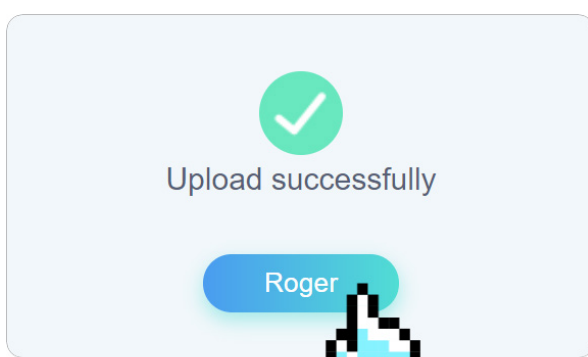
Select the serial port number corresponding to the current Wio Terminal (not necessarily COM26 as shown in the figure) and click the “OK” button.



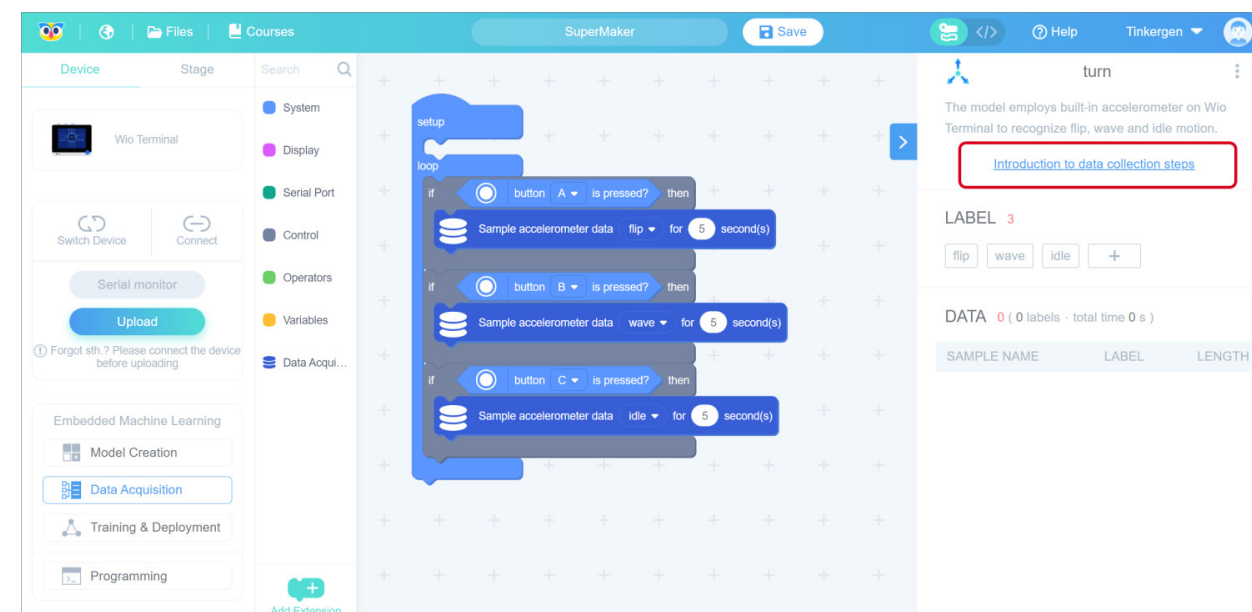
A pop-up window indicates that the program is being uploaded, please wait...



Usually, it takes 10 seconds to upload. Once the program is uploaded, the "Upload Successful" window will appear on the screen shown below. This is shown in the image below.



Click "Roger" to close the upload success pop-up window and return to the programming screen.



Attention:

- Wio Terminal button location.
- Animated GIF has been accelerated, the actual action can slightly slow down.
- Please notice the red tips.
- Point the cursor over Description Texts for more detailed content.

Use Wio Terminal to acquire data

1

Create labels and program for data acquisition

2


Connect to the Wio Terminal and upload the data acquisition program

3

Data acquisition


C B A

Motion Recognition (Built-in Accelerometer)




flip

Press A to start the data acquisition and flip the Wio Terminal several times until the...



wave

Press B to start the data acquisition and wave the Wio Terminal several times until the...



idle

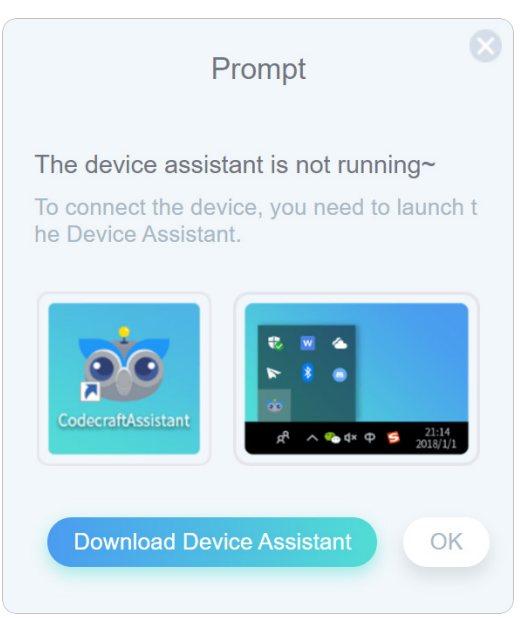
Press C to start the data acquisition and place the Wio Terminal on the desktop or hol...

Data acquisition for each movement is recommended to be performed more than six times with minor variations (speed, force).

Caution

For the web version of Codecraft, you may get the message in the image below if you don't install or run the Device Assistant.

Check this page for further information: [Download, installation and "Device Assistance" Usage](#)



2.3 Data Acquisition

In the upper right hyperlink, you will find a step-by-step introduction to data acquisition. Follow the instructions to collect data.

Use Wio Terminal to acquire data

1

Create labels and program for data acquisition

2

Connect to the Wio Terminal and upload the data acquisition program

3


Data acquisition

Motion Recognition (Built-in Accelerometer)



flip

Press A to start the data acquisition and flip the Wio Terminal several times until the...



wave

Press B to start the data acquisition and wave the Wio Terminal several times until the...



idle

Press C to start the data acquisition and place the Wio Terminal on the desktop or hol...

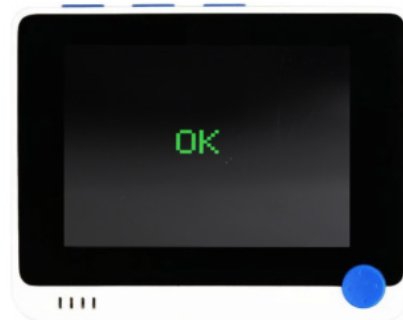
Data acquisition for each movement is recommended to be performed more than six times with minor variations (speed, force).

Starting and finish data acquisition according to the Wio Terminal display.

 This signal means data is being collected.

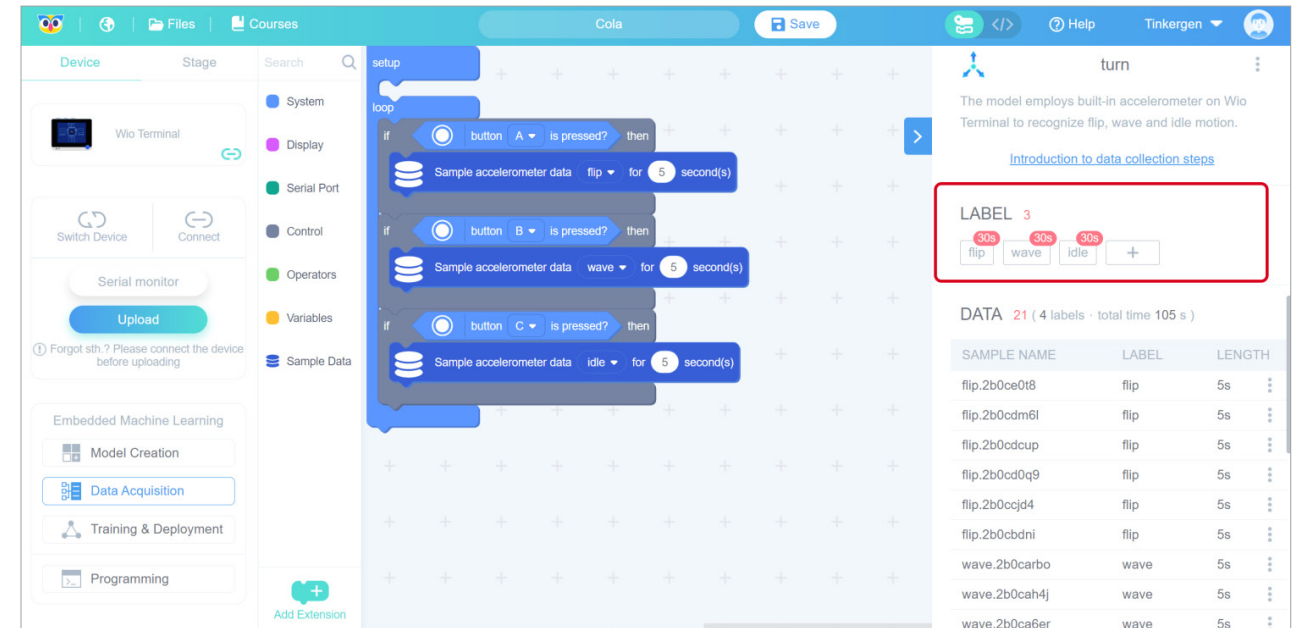


 OK means Data collection is complete.



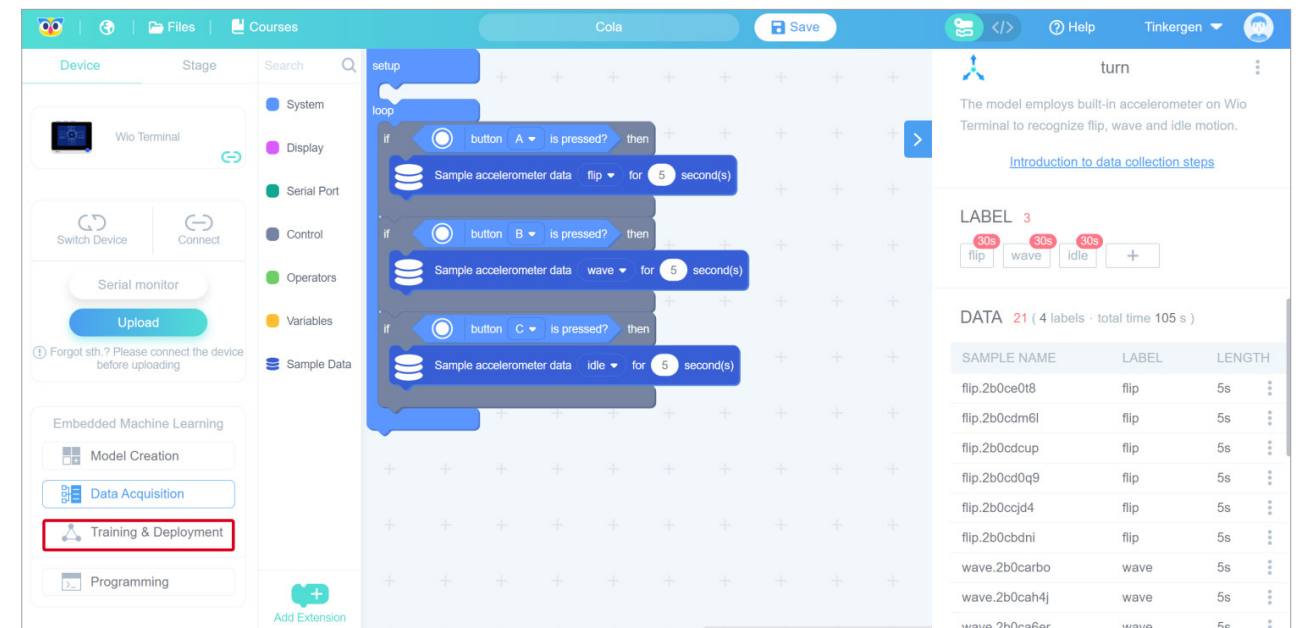
Wio Terminal shows that the data acquisition is finished and CodeCraft is still uploading the data. It takes 1~2s to transfer the data from Wio Terminal to CodeCraft.

Now, the data acquisition is finished.



SAMPLE NAME	LABEL	LENGTH
flip.2b0ce0t8	flip	5s
flip.2b0cdm6l	flip	5s
flip.2b0odcup	flip	5s
flip.2b0cd0q9	flip	5s
flip.2b0ccjd4	flip	5s
flip.2b0cbdni	flip	5s
wave.2b0carbo	wave	5s
wave.2b0cah4j	wave	5s
wave.2b0ca6er	wave	5s

Click on "Training & Deployment".



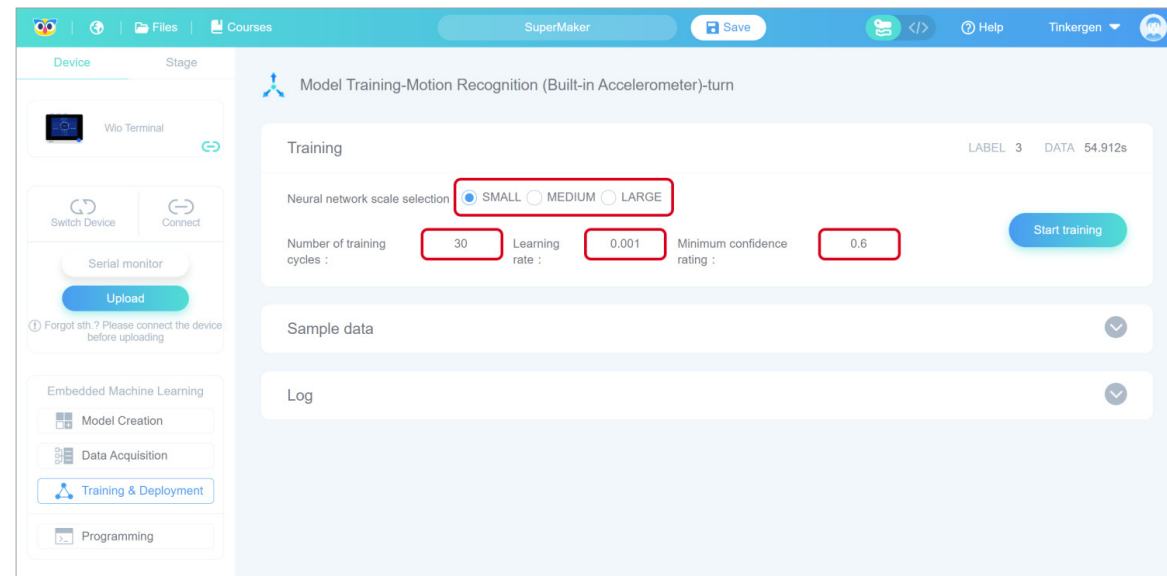
Step 3. Training and deployment

3.1 Set neural network and parameters

Select the suitable neural network size: one of small, medium, and large.

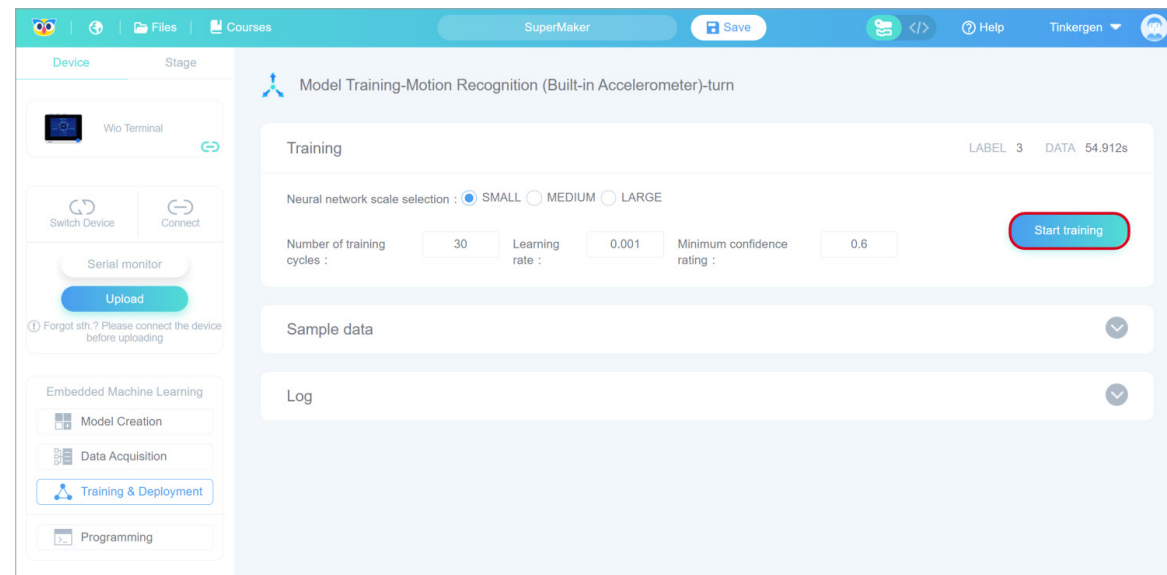
Set parameters, number of training cycles (positive integer), learning rate (number from 0 to 1), and minimum confidence rating(number from 0 to 1).

The interface provides default parameter values.

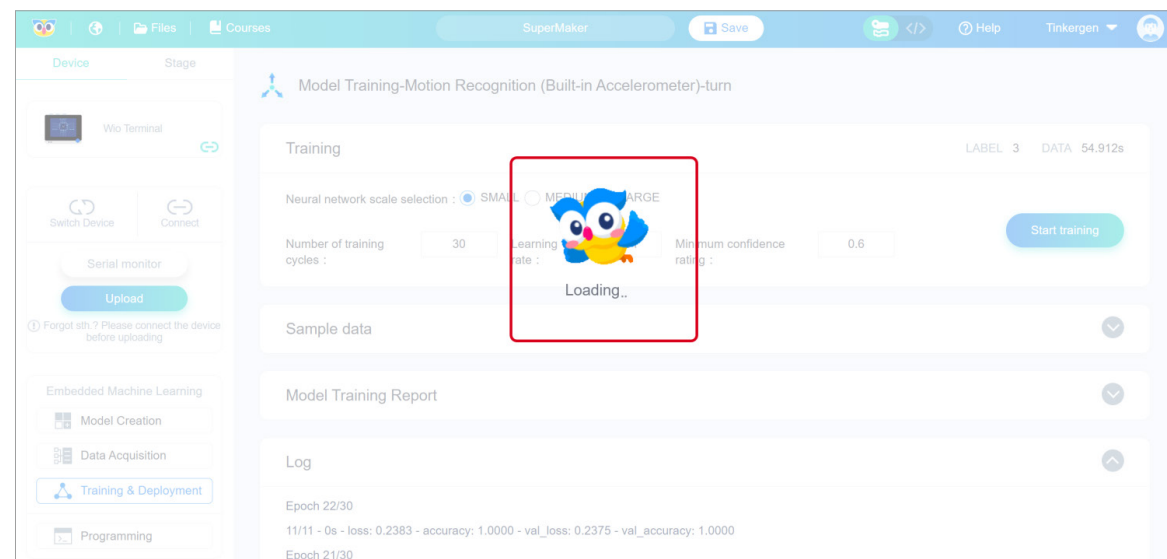


3.2 Start training the model

Click “Start training”.

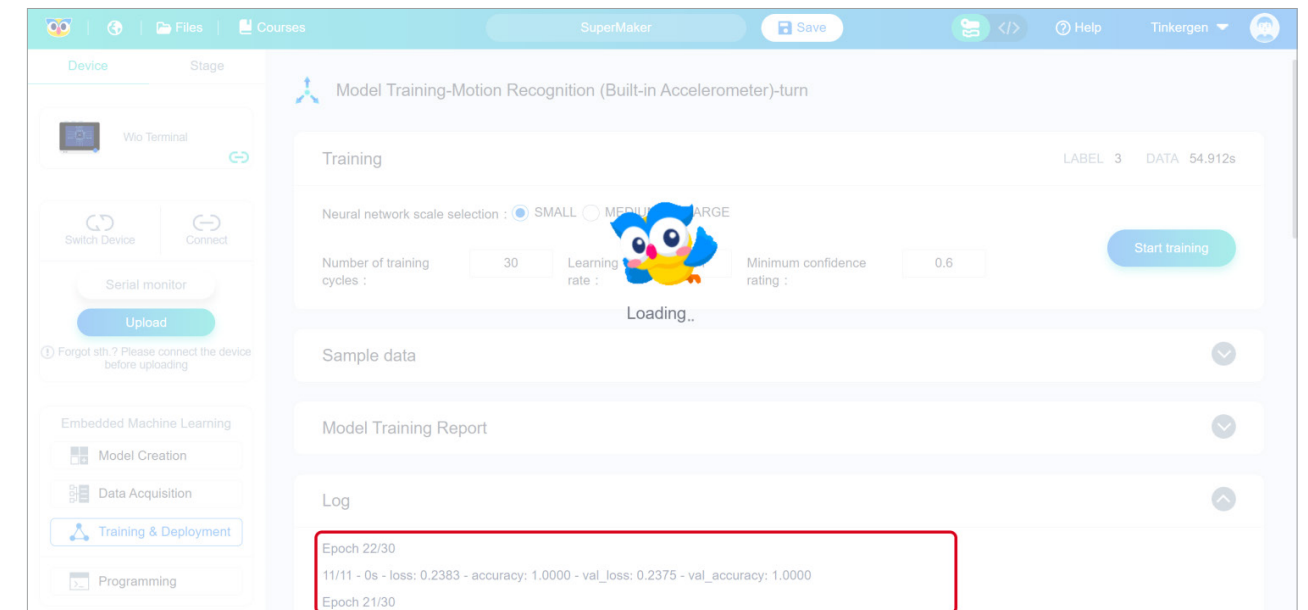


When you click “Start training”, the interface displays “Loading..”.

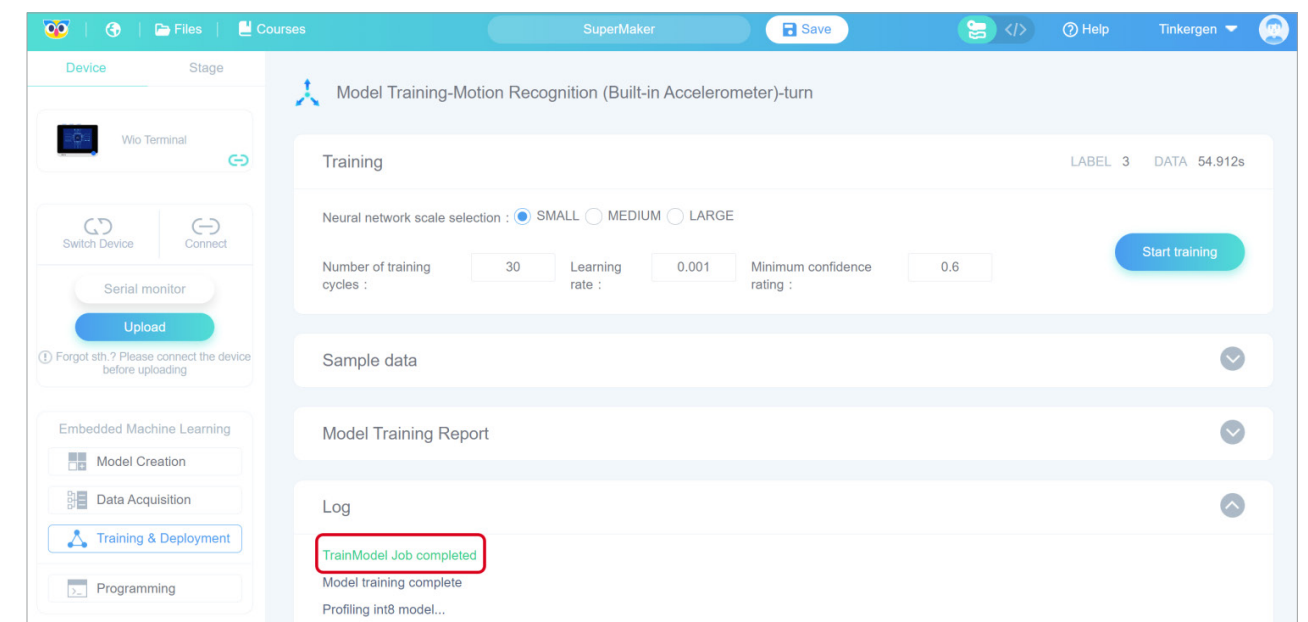


The duration of “Loading..” varies depending on the size of the selected neural network (small, medium and large) and the number of training cycles. The larger network size is and the more number of training cycles are, the longer it will take.

You can also infer waiting time by observing the “Log”. In the figure below, “Epoch: 22/30” indicates the total number of training rounds is 30 while 22 rounds have been trained.



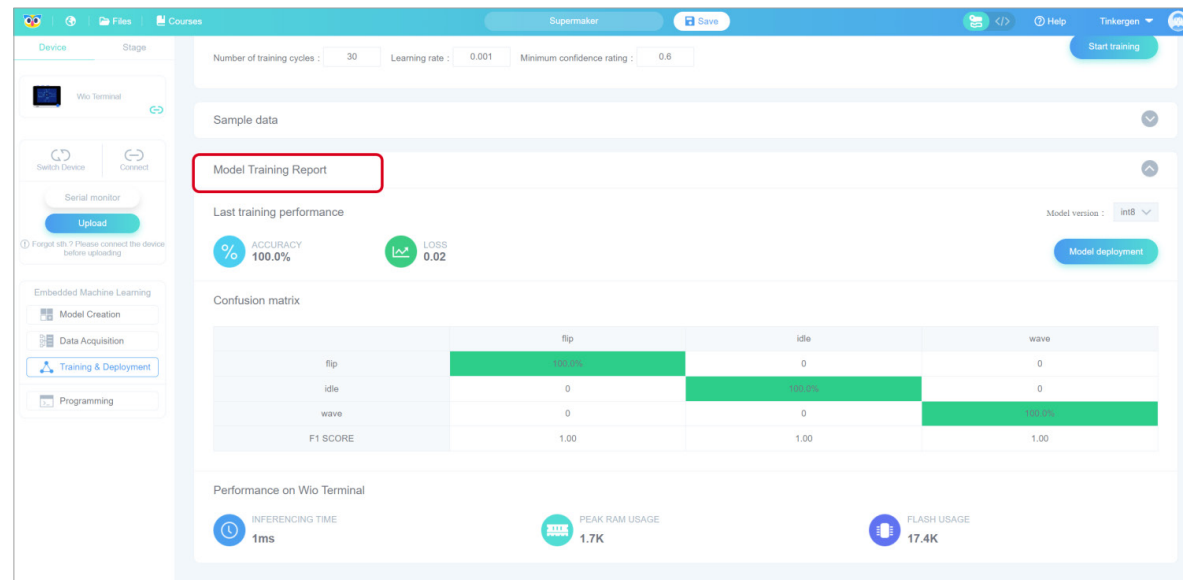
After loading, “TrainModel Job Completed” appears in the “Log” and a “Model Training Report” shows up on the interface.



3.3 Observe the model performance to select the ideal model

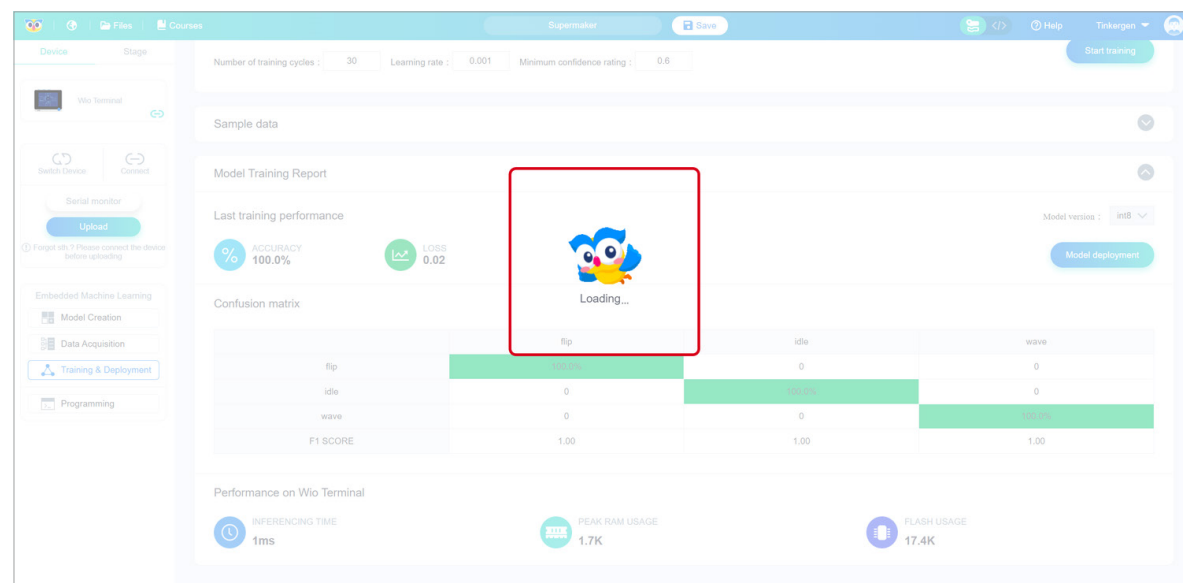
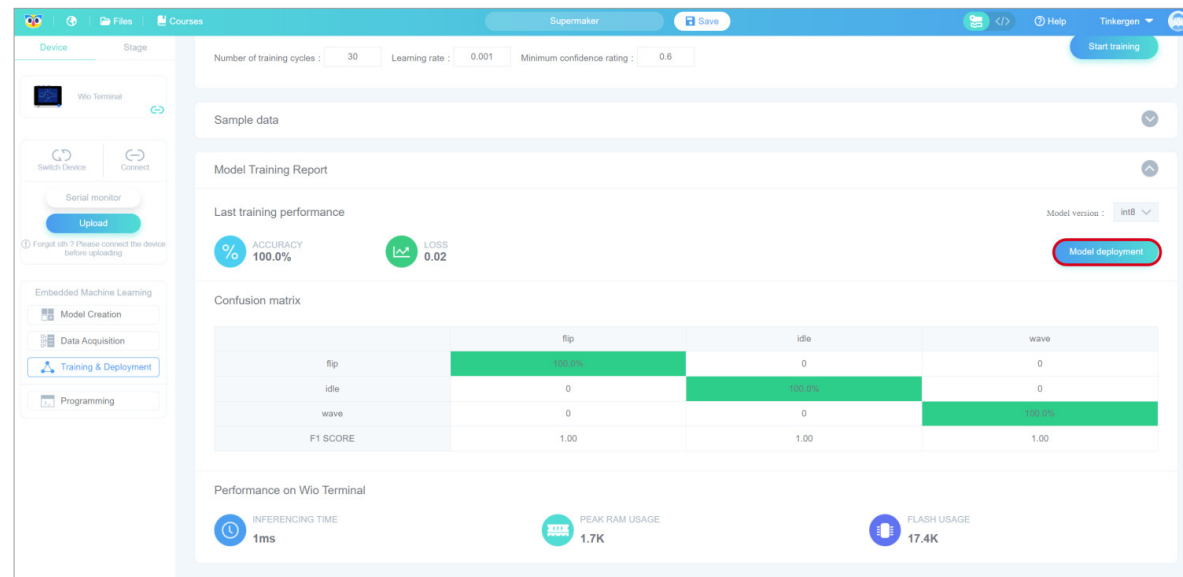
In the “Model Training Report” window, you can observe training results including the accuracy, loss and performance of the model.

If the training results are not satisfactory, you can go back to the first step of training the model, selecting other sizes of the neural network or adjusting the parameters and training it until you get a model with satisfactory results.

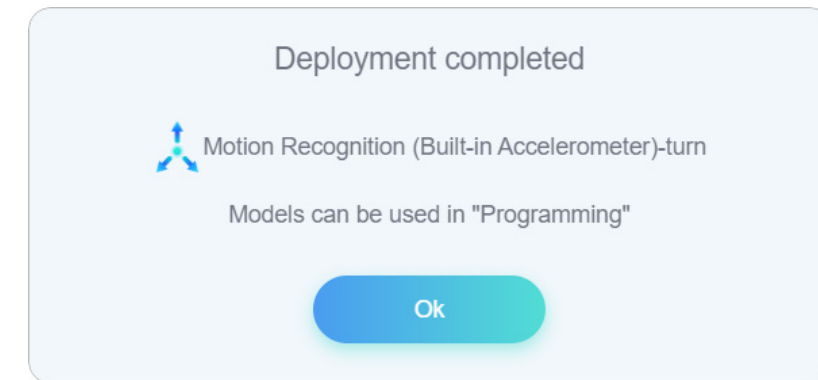


3.4 Deploy the ideal model

In the "Model Training Report" window, click "Deploy Model".



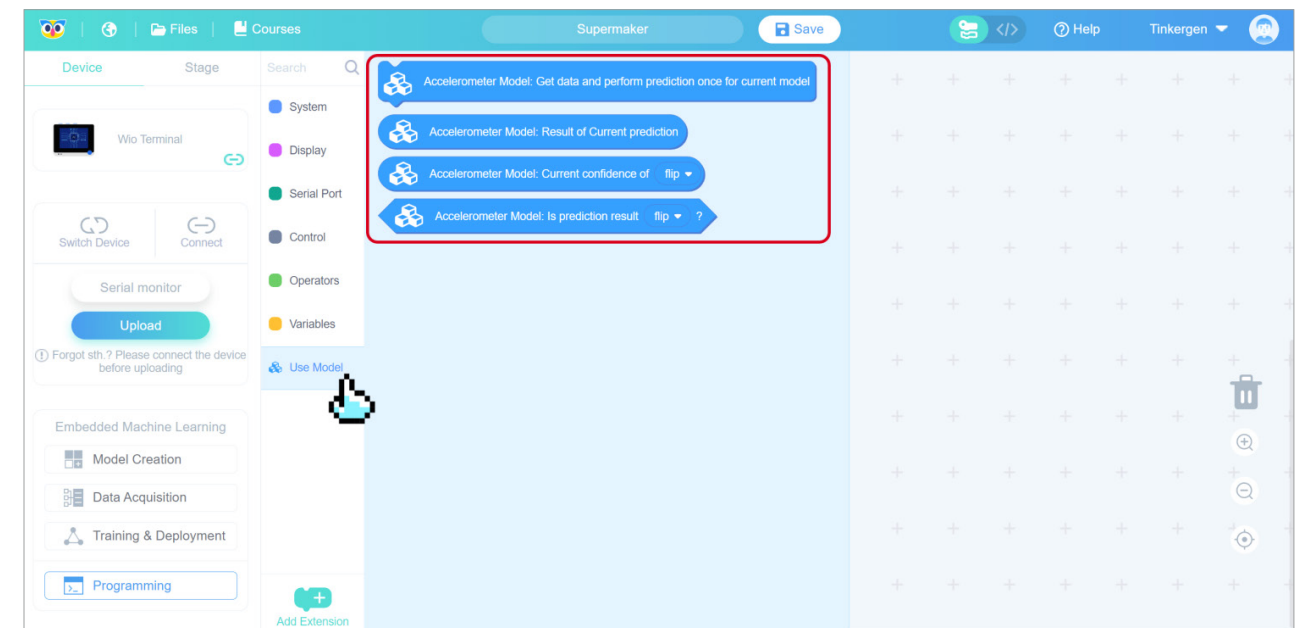
Once the deployment is completed, click "Ok" to jump to the "Programming" window



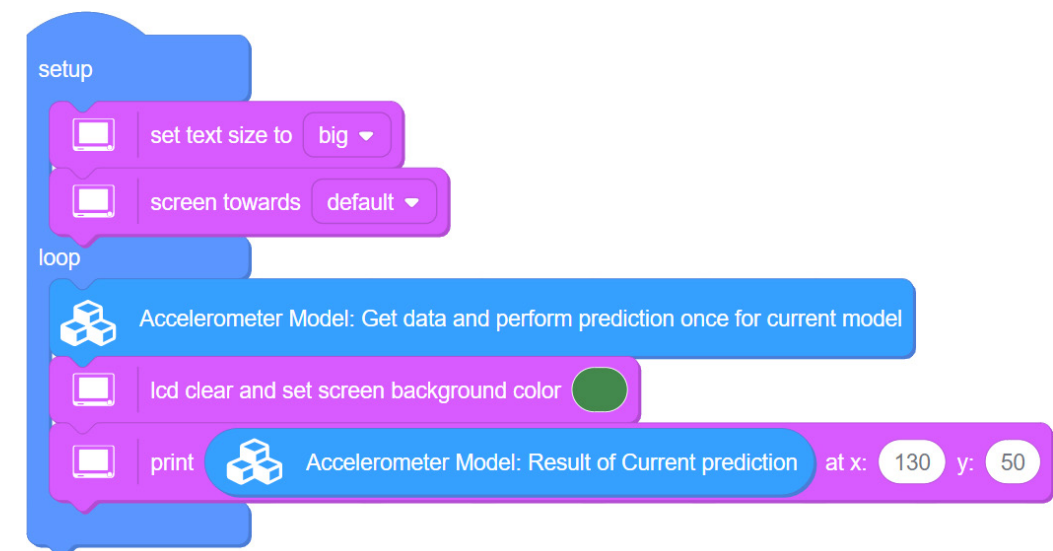
Step 4. Use and programming

4.1 Write the program for using the model

In the "Programming" interface, click on "Use Mode" to use the deployed model.

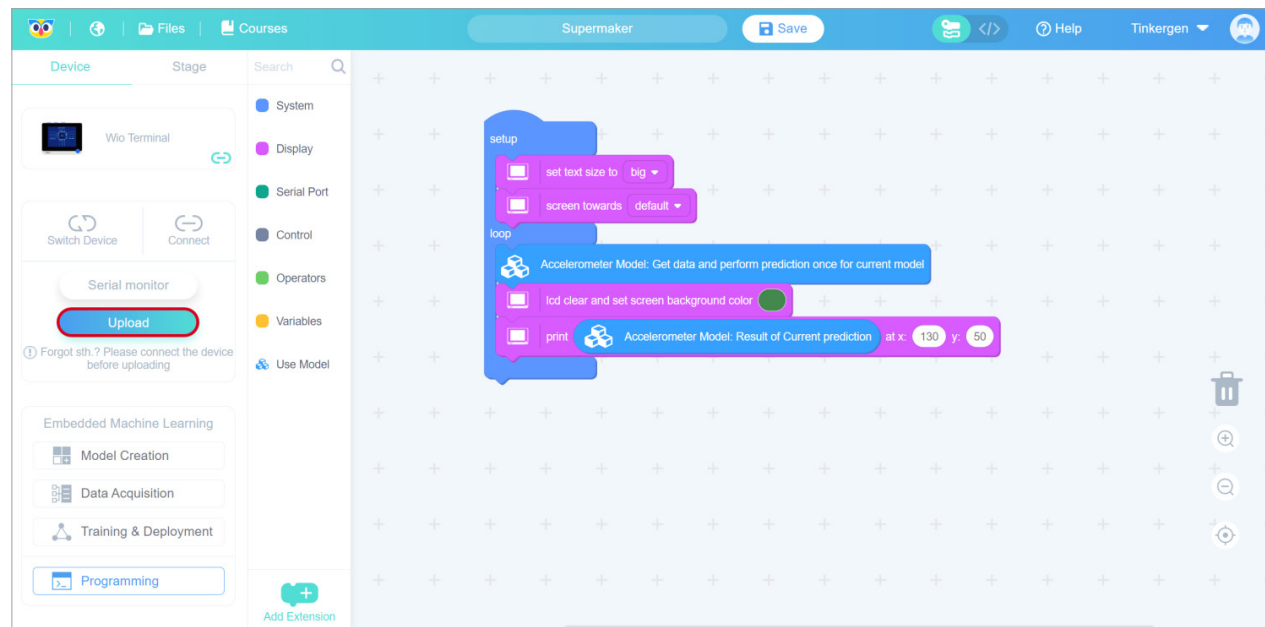


Try to use your model by writing the following program.

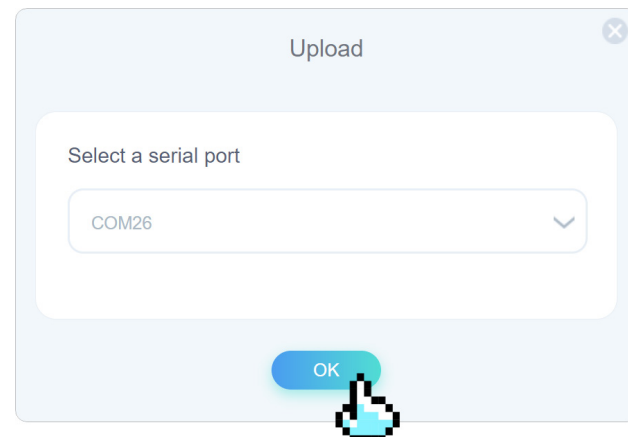


4.2 Upload the program to Wio Terminal

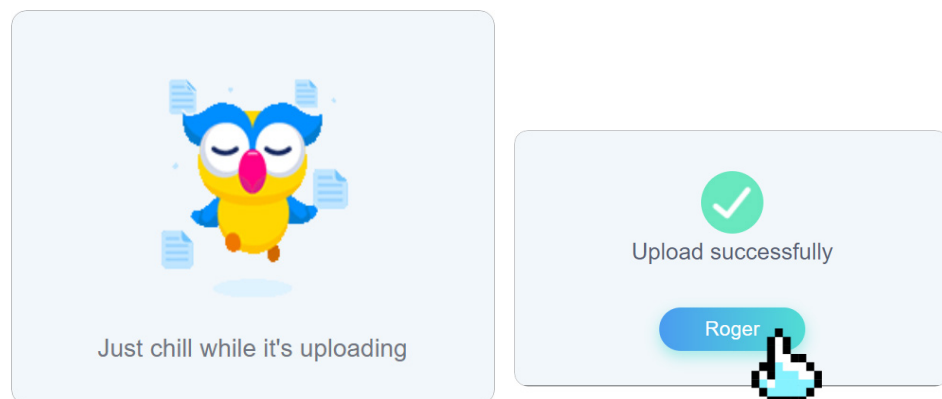
Click the “Upload” button.



Select the serial port number corresponding to the current Wio Terminal (not necessarily COM26 as shown in the figure) and click the “OK” button.



The first upload time usually cost longer and it increases with the complexity of the model. The uploading time for smaller models cost about 4 minutes or even longer(depending on the performance of your machine).



4.3 Wio Terminal test model

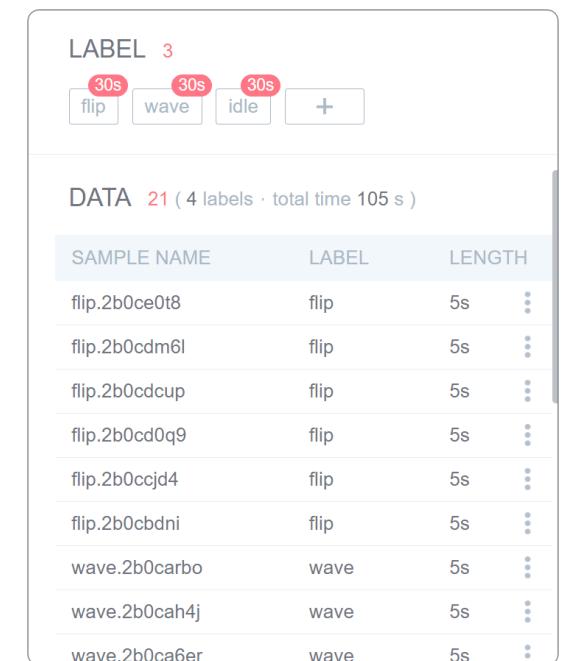
Flip your Wio Terminal to see whether its screen shows “Flip”.Try other motions, and see whether the Wio Terminal recognizes your motions.

Congratulations! You have completed your first TinyML model. I believe you are interested in understanding the significance of each operation you do, so let’s learn the basic theoretical knowledge of machine learning, first look at our input.

ML Theory(understand your input):

Input:

- Label
- Dataset
 - a. Training set, Val set, and Test set
 - b. Amount of dataset
 - c. Way to get data
 - d. Quality of dataset



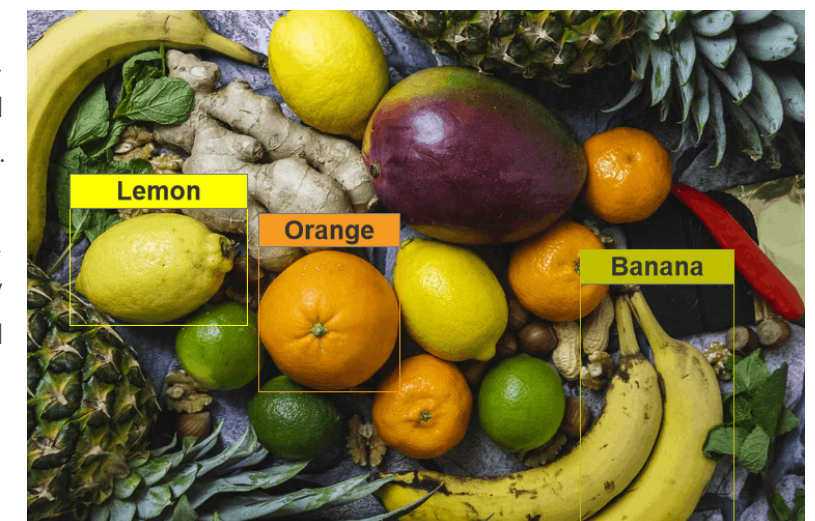
Input:

1. Label

1.1 What is Label:

As the name suggests, data is labeled when you add meaningful labels and tags or assign classes to the raw data that you’ve collected which means that you add labels to data and set a target. The AI learns by example.

Now, let’s see how this works. If you have a set of fruits labeled with “Lemon”, “Orange” or “Banana”. The classification trains your model to tell whether the fruit is “Lemon”. When the training is over, newly added fruit will be classified into one of these three groups.



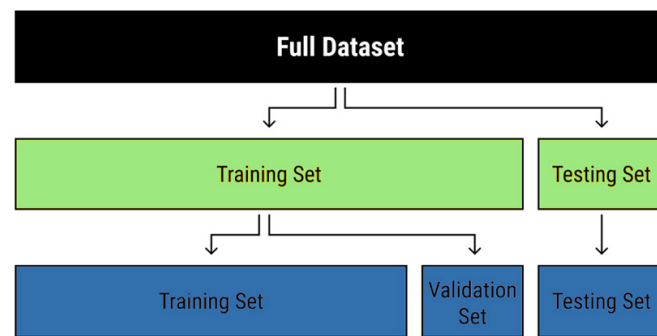
2. Dataset

2.1 What is Dataset:

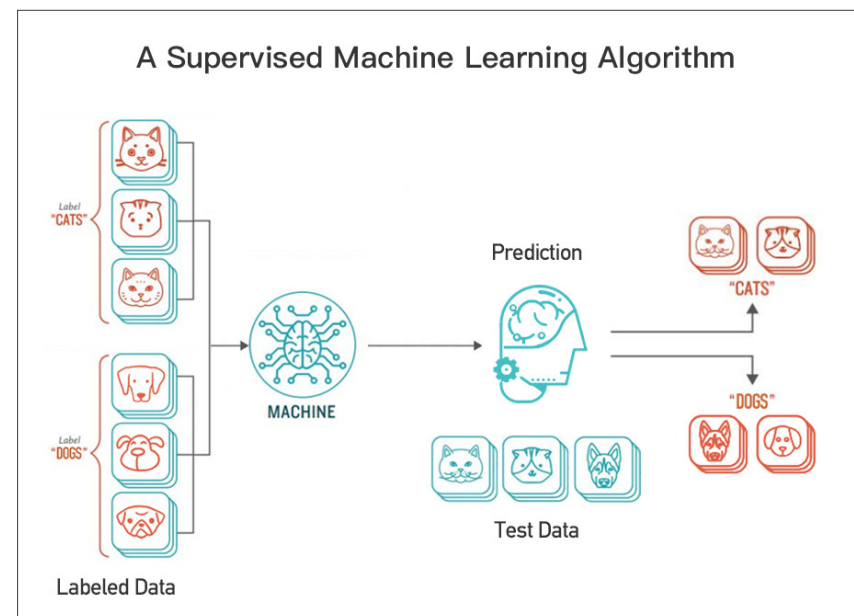
If you are looking for a working and reliable model, you need a large amount of relevant,

high-quality data. It means that you have to train your ML model by providing it with thousands of examples and the it can memorize correlations among your data pieces.

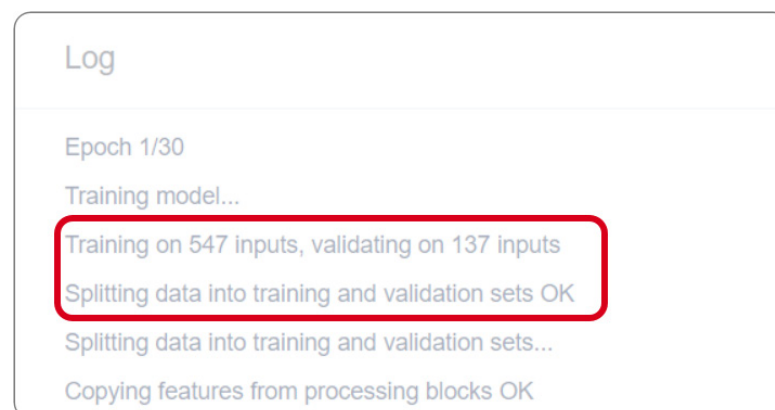
The Full Dataset is consists of training set and testing set:



The data you feed to the model is called training dataset while the data you provide to the model to predict is called testing set.



In addition, the training set can be divided into two groups. We can observe the Log after we start training, finding that Training Set will be split into “Training Set” and “Validation Set”.



Training set: Training set is the data that an algorithm uses to learn.

Validation set: After using the training set, the validation set is used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. We know the true label of each sample in the validation set, but pretend not to know it for the moment. Take validation set as input to the classifier, then receive the classification result. We can now peek at the true labels of the validation data and see if we guessed correctly. If we do this for every sample in the validation set, we can calculate the validation error. Hence the model occasionally sees this data, but never does it “Learn” from this.

2.2 How Much Training Data Is Enough?

Short answer: There is no correct answer.

Detailed answer: Different projects require different amounts of data.

The only way to know how much you’ll need is by trial and error but hopefully the following guidelines that may help you get a basic idea about why this question has no answer:

- Usually, more sophisticated models with more attributes and links between them (like ANNs) will require more data to train properly.
- The scope of application along with the complexity of the real-life phenomena that your model is being trained to predict also plays a role in how much training data is required. Beware of the exceptions and blind spots.
- With time, you will likely need to re-train or tweak your model as the trends that it predicts change which will require more data in the long term.

The amount of required training data depends upon a lot of factors. The “Introduction to data collection steps” for each example project has hints on how much data to collect, but this is just a general recommendation. You can try and then get more training data as you go on.

2.3 How to get Data



- Get by yourself: using the Wio Terminal with sensors and other devices.
- Open Source: some data science communities like Kaggle.
- Internet: according to your goal, search the dataset your want online.

2.4 How about the Quality of a Dataset:

- Accuracy
- Relevance

- Coverage
- Balance

Accuracy: Accuracy refers to how well the data describes real-world conditions which aims to describe. If you try teaching a computer to recognize a cat, you need to label the cat as “cat” rather than “dog” .

Relevance: The data you collect should also be useful for initiatives you plan to use it for. If it is not relevant to your goals, it is not useful for you. If you want to teach a computer to recognize a cat by its picture, you don't need to provide photos of cat's food. It's important to set goals for your data collection so that you know what kind of data to collect.

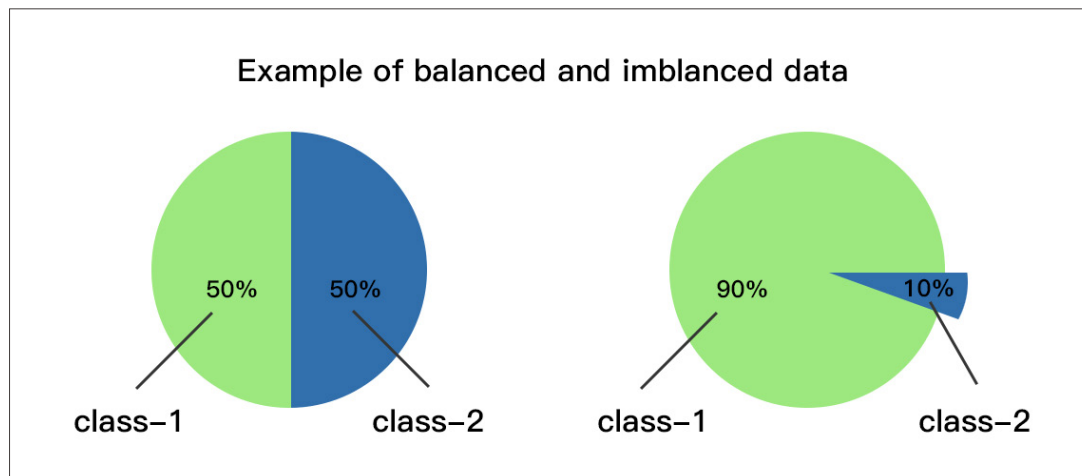
Coverage: If you try teaching a computer to recognize a cat, you'll need to show thousands of images of cats that differ in size, color, and form to make your model capable of identifying a cat accurately.

Balance: Imbalance is Common.

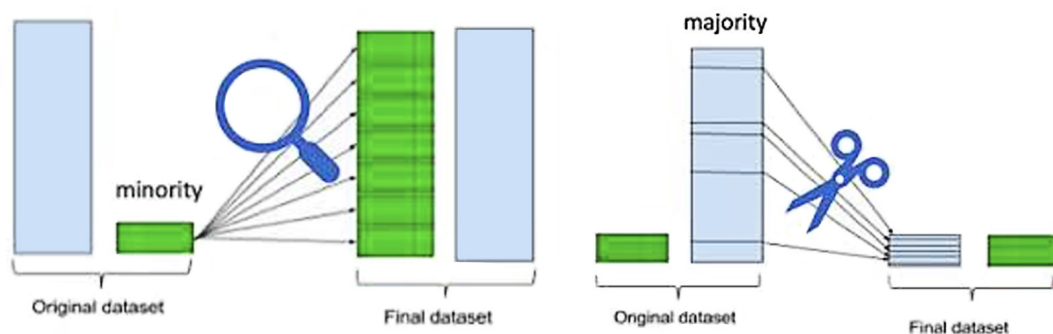
Most classification data sets do not have equal number of instances in each class but such a little difference does not matter.

What happens to our models when we train on an imbalanced dataset?

Our models look at the data and cleverly decides to always predict “Class-1” and achieve high accuracy, 90%.



How to Solve the Imbalance?

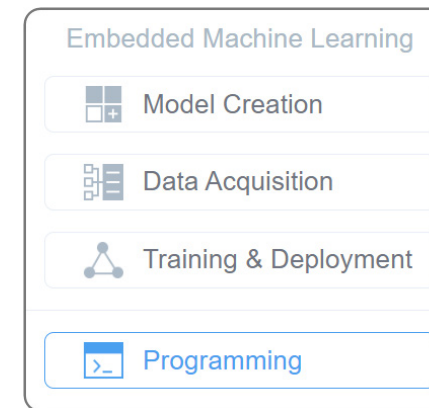


Over-sampling (Up Sampling): Increases the number of minority class members in the training set.

Under-sampling (Down Sampling): Reduces the number of majority samples to balance the class distribution.

★ Summarization

1. Theory: Accelerometer
2. TinyML practice



3. ML theory (Input)

- Label: Assign classes to the raw data that you have collected.
- Dataset:
 - a. Training set, Validation set and Testing set: the training set is fed into the model, the validation set is used to validate the results to improve the model and the testing set is used to predict.
 - b. Required Data volume: More is better.
 - c. Way to get data: Get by yourself, open sources, Internet.
 - d. Quality of a dataset: Accuracy, relevance, coverage, balance.

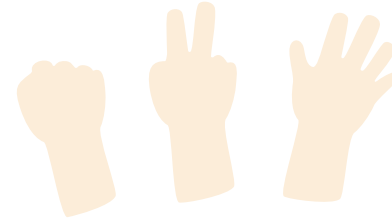
Lesson 03

Gestures Recognition using built-in Light Sensor



Project Overview

This model uses the Wio Terminal to recognize hand gestures: rock, paper and scissors using built-in Light sensor.



It is quite hard to accomplish using rule-based programming because gestures are not performed the same way all the time. Trying to solve this problem using traditional Programming requires hundreds of different rules for each mode of action. Even if we consider an idealized situation, we still miss a lot of motions such as speed, angle, or direction. A slight change in these factors requires a new set of rules to be defined but machine learning can handle these variations very easily.

A well-known possibility is to use camera sensors combined with machine learning to recognize gestures. Using a light sensor is equivalent to using only a one-pixel point of the camera to recognize gestures which is a completely different level of challenge. Let's meet the challenge with this project.

Expected results

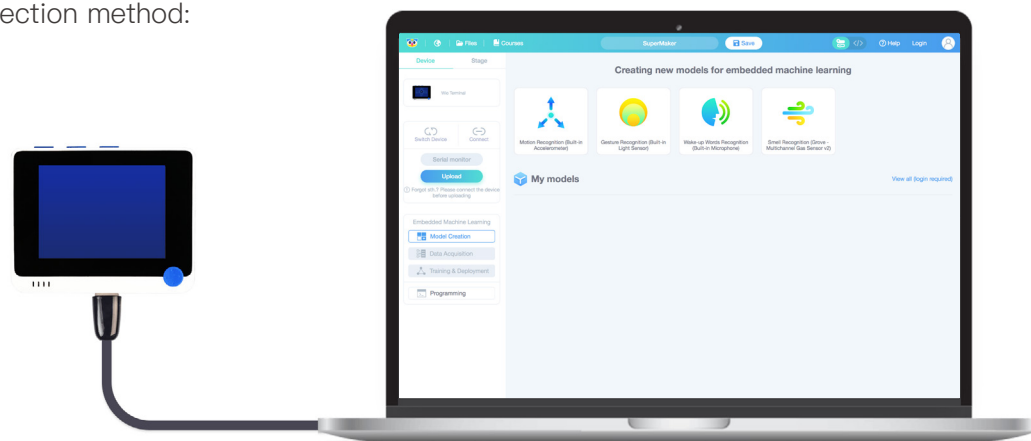
In the figure below, Wio Terminal displays the current results of gesture recognition in real-time. One of rock, scissors, and paper.



Material Preparation

Hardware requirements: Wio Terminal

Connection method:



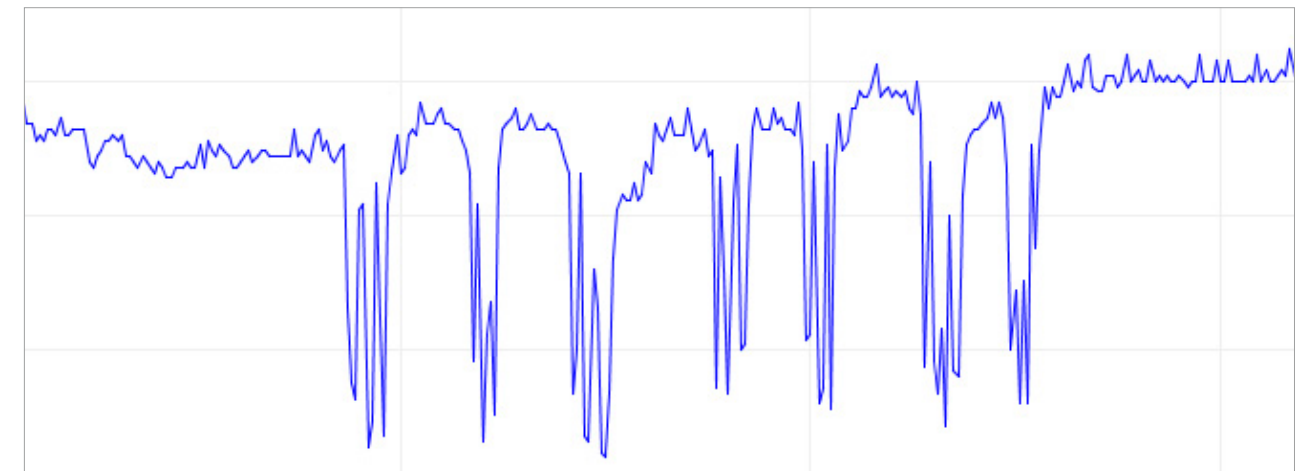
Theory

In this lesson, we are going to train and deploy a simple neural network for classifying rock-paper-scissors gestures with just a **single light sensor**.

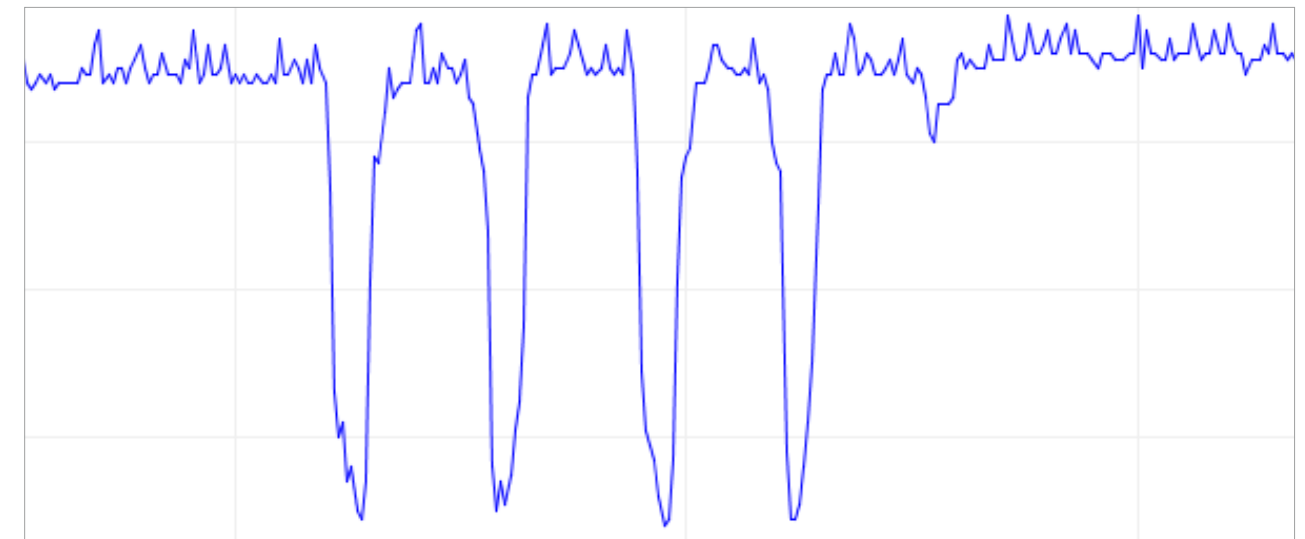


The working principle of this project is quite trivial. Different gestures being moved above the light sensor will block certain amount of light for certain periods of time. For example, for rock, we will have high values at first (nothing above sensor) but lower values when "rock" passes above the sensor and then high values again. For paper, we will have high-low-high-low-high-low-high-low values when each of the fingers in "paper" passes above the sensor.

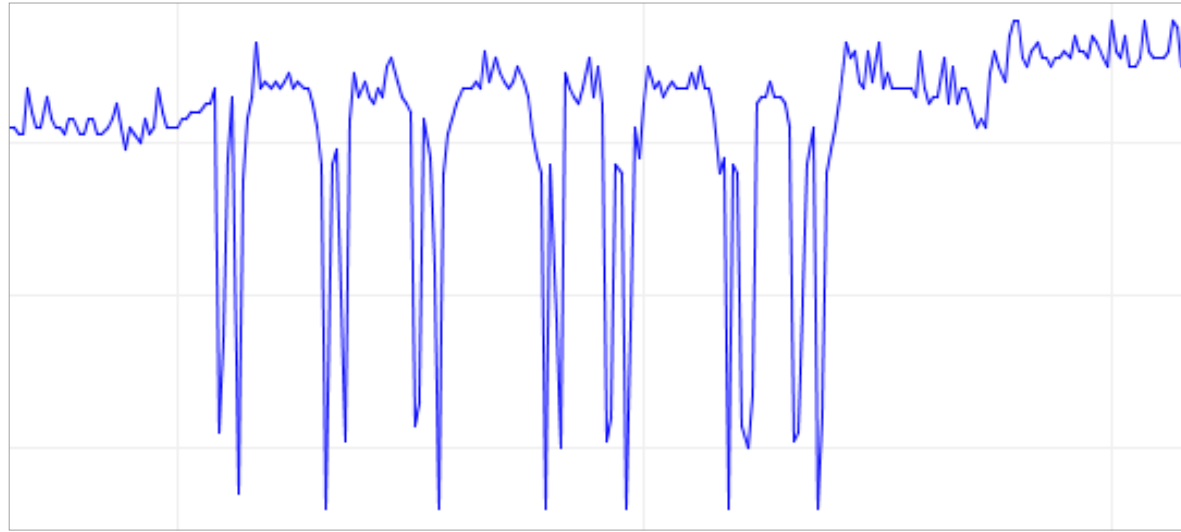
Paper



Rock



Scissors



There is a high variance in speed and amplitude of the values from the sensor which makes a great case for using machine learning model instead of a hand-crafted algorithm for the task.

Enter a NAME according to the requirements.

Name of the new model

Based on model: Wio Terminal gesture recognition using built-in light sensor

Please enter a string containing 1-30 letters, numbers or '_', beginning with letters.

Cancel
Ok

Click Ok and it will automatically jump to the Data Acquisition interface.

Step 2. Acquisition of data

2.1 Default label

LABEL 3

rock

paper

scissors

+

Practice

Project Steps

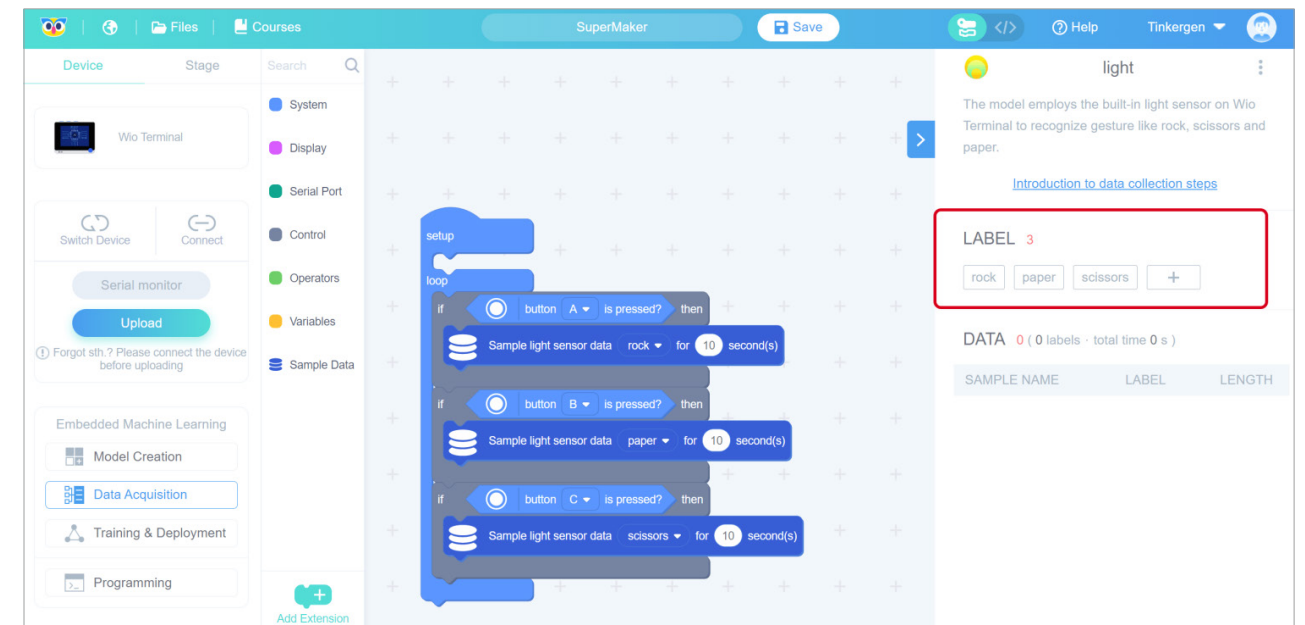
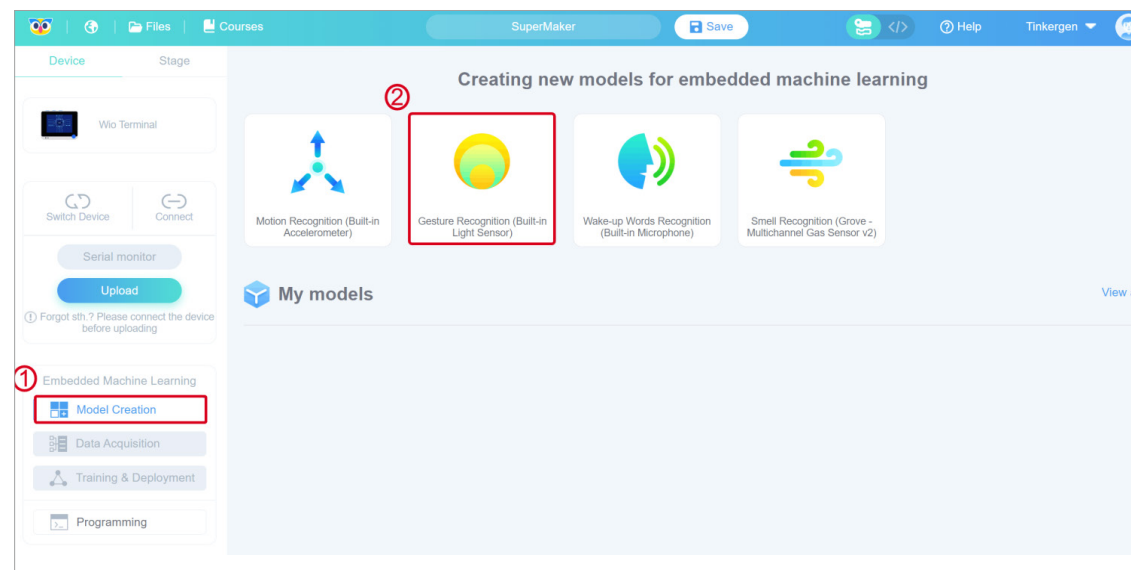
1. Creating and Selecting Models
2. Data Acquisition
3. Training and Deployment
4. Programming

Please Open <https://ide.tinkerjen.com/>

Step 1. Create and select models

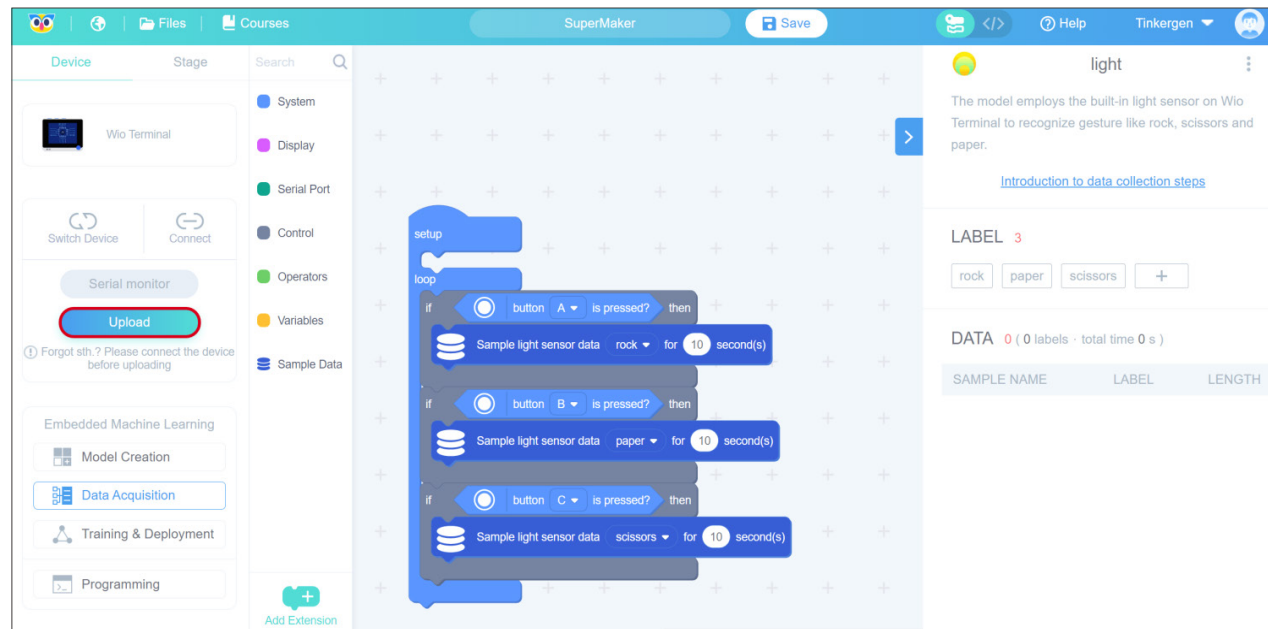
1.1 Create the “Gesture Recognition (Built-in Light Sensor)” model

Click on “Create and select model”, click on “Gesture Recognition (Built-in Light Sensor)”, as shown in steps 1 and 2 below.

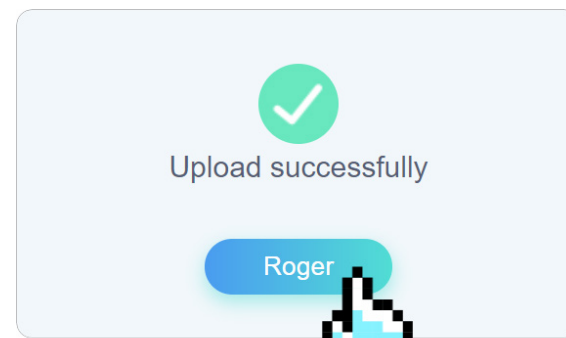


2.2 Connect the device and upload default data acquisition program in Codecraft

When the Wio Terminal is connected, in the Codecraft surface, click Upload as shown in the figure. This action will upload the default data acquisition program.



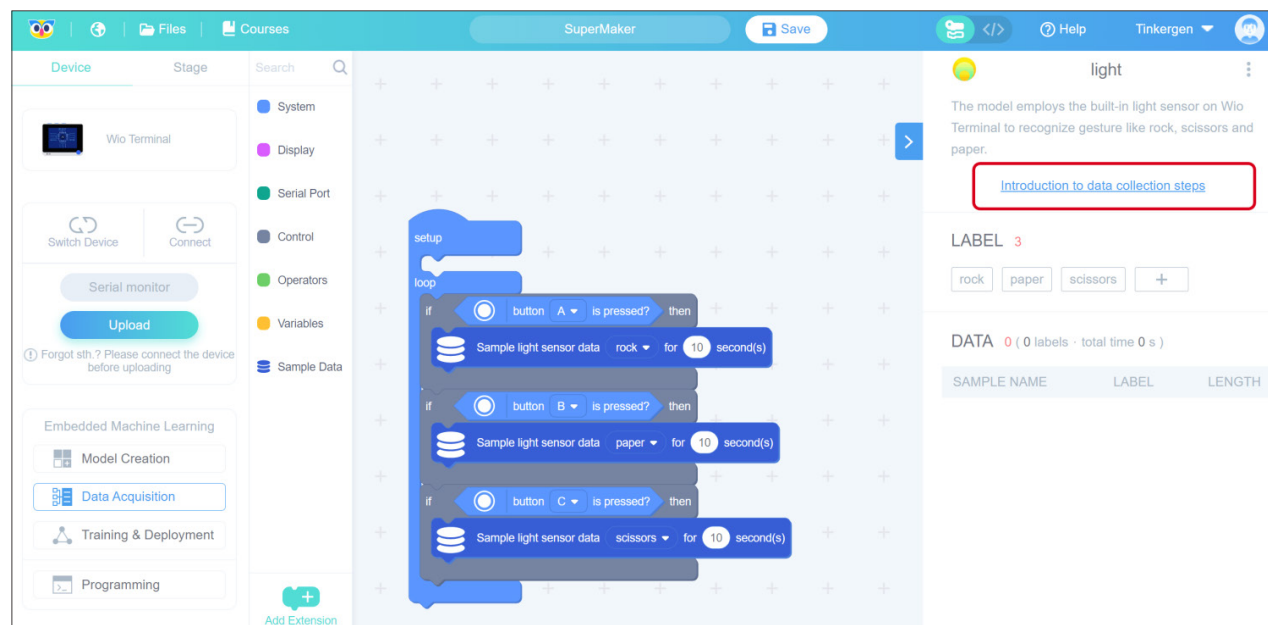
Usually it takes 10 seconds to upload. Once the program is uploaded, "Upload Successful" window will appear on the screen shown below. This is shown in the image below.



Click "Roger" to close the upload success pop-up window above and return to the programming screen.

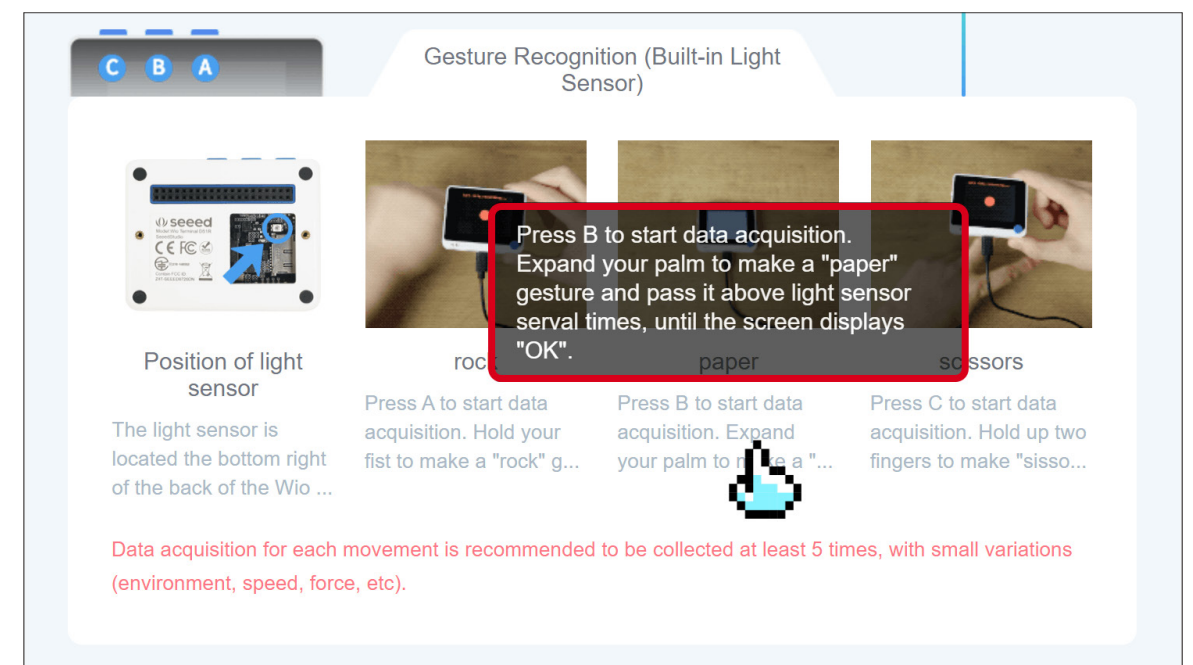
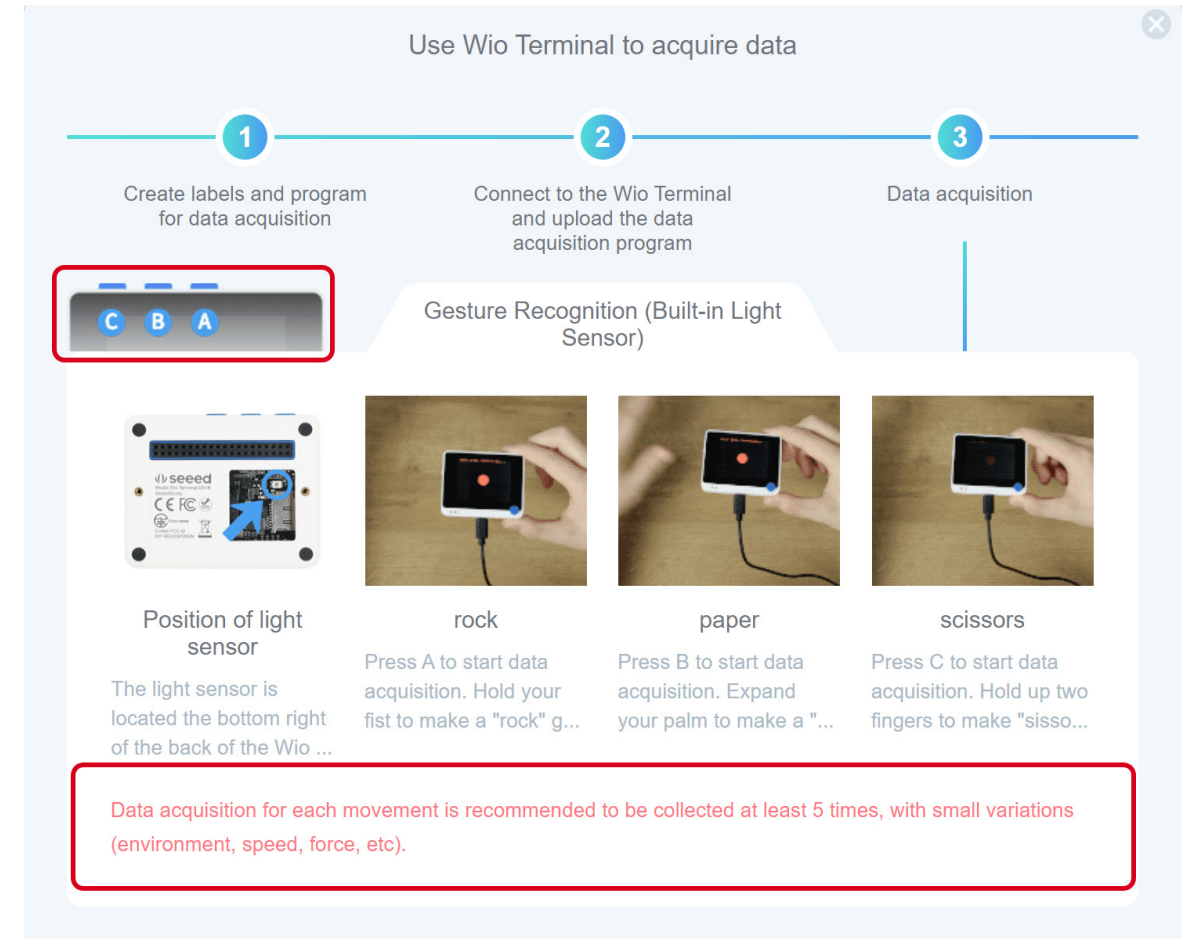
2.3 Acquisition of data

In the upper right hyperlink, you will find a step-by-step introduction to data acquisition. Follow the instructions to collect data.





Attention:

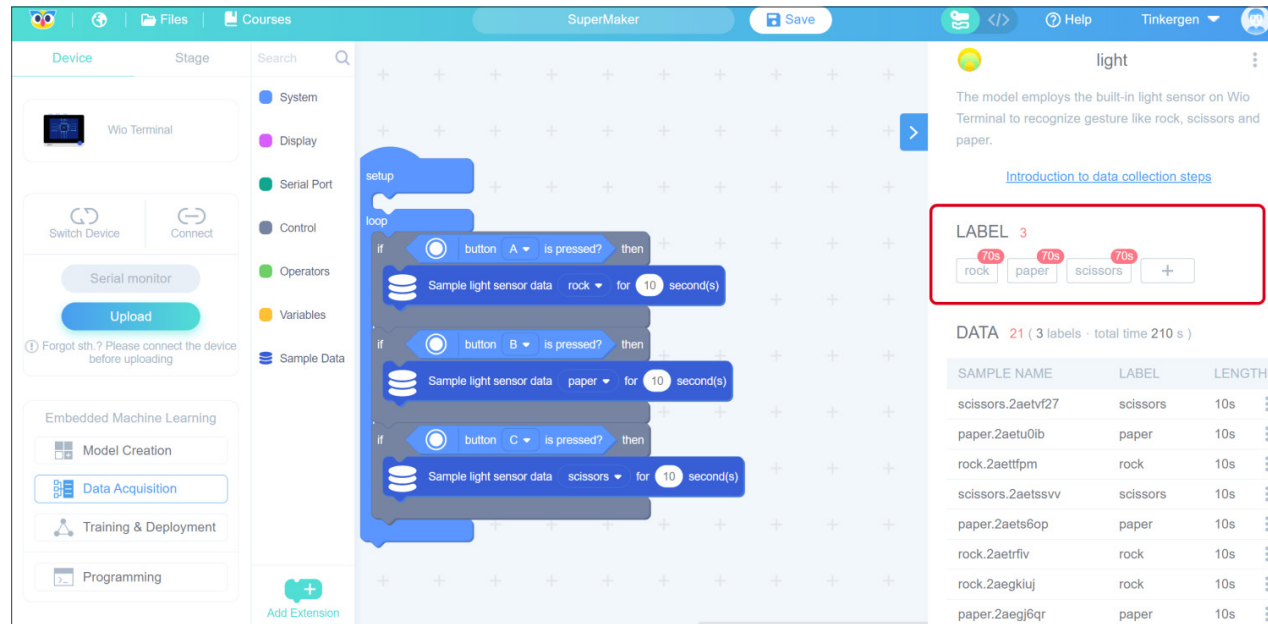
- Wio Terminal button location.
- Animated gif has been accelerated, the actual action can slightly slow down.
- Please notice the red tips.
- Point the cursor over Description Texts for more detailed content.



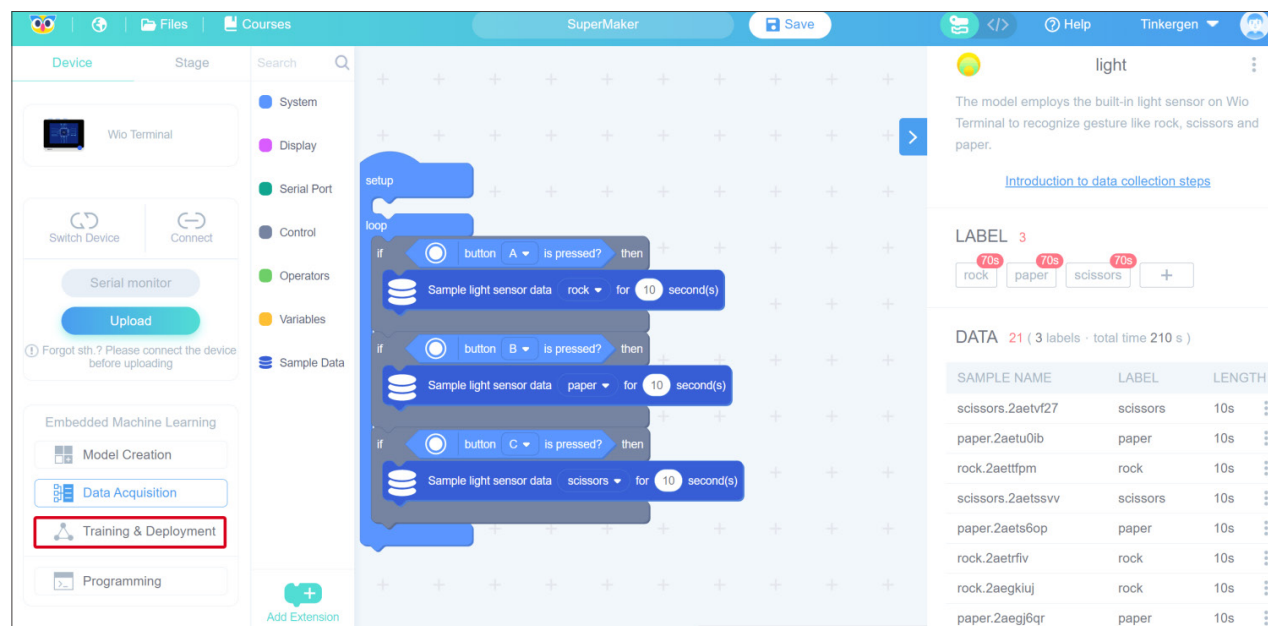
Wio Terminal will be displayed during the collection process.
 Start and end collecting data according to the Wio Terminal screen.

-  Indicates the data is being collected
-  Indicates the Data collection is complete

Now, the data acquisition is finished.



Click on “Training & Deployment”



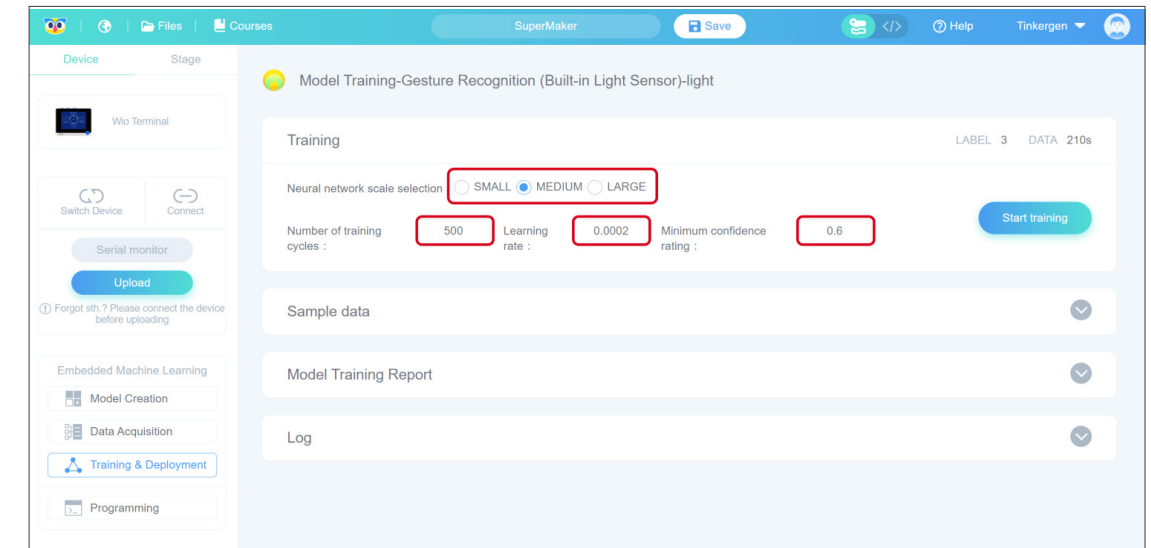
3. Training and deployment

3.1 Set neural network and parameters

Select the suitable neural network size: one of small, medium and large

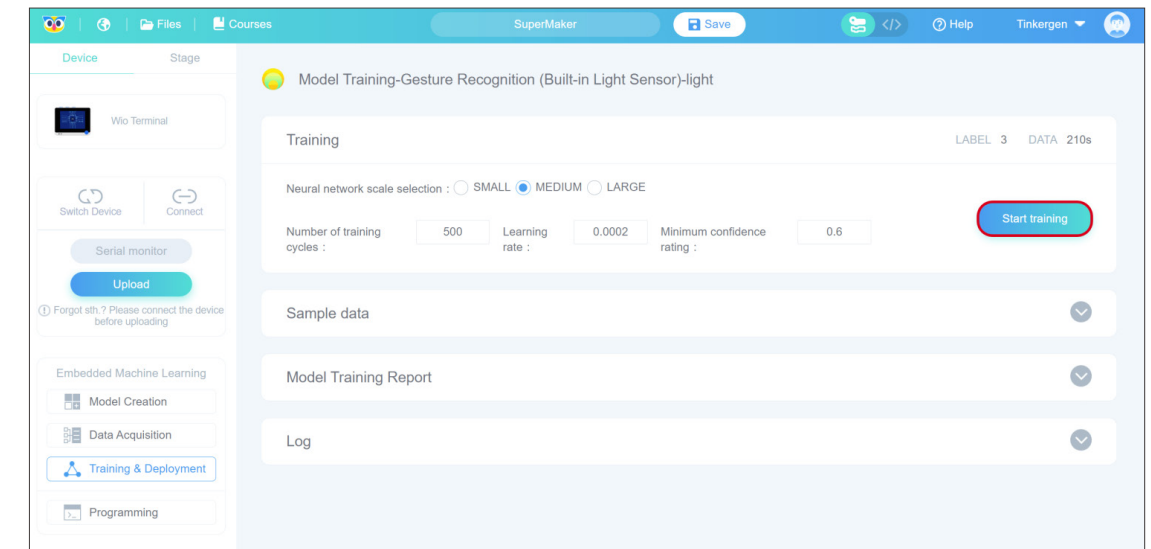
Set parameters, number of training cycles (positive integer), learning rate (number from 0 to 1) and minimum confidence rating(number from 0 to 1).

The interface provides default parameter values.
 In this case we are using MEDIUM. It will take quite a long time.

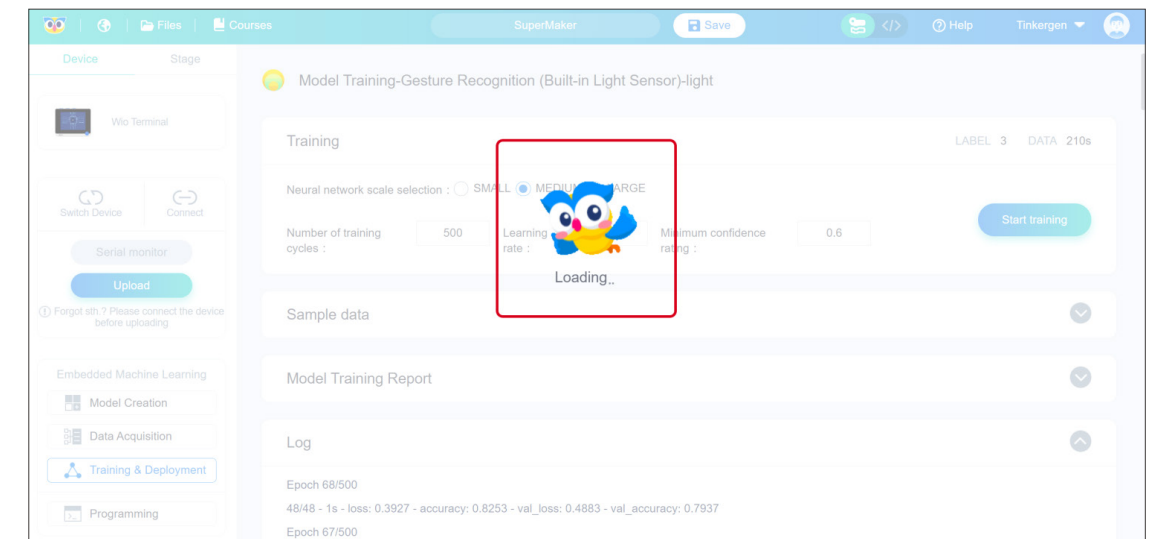


3.2 Start training the model

Click “Start training”.

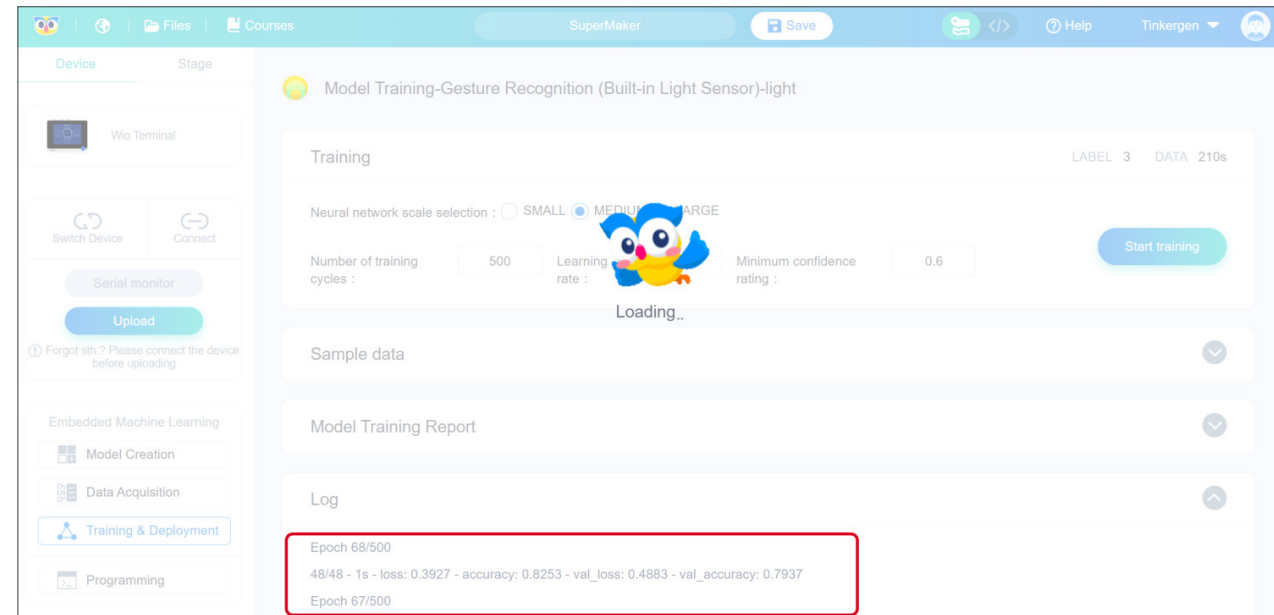


When you click “Start training”, the interface displays “Loading...”.



The duration of “Loading..” varies depending on the size of the selected neural network (small, medium and large) and the number of training cycles. The larger network size is and the more number of training cycles are, the longer it will take.

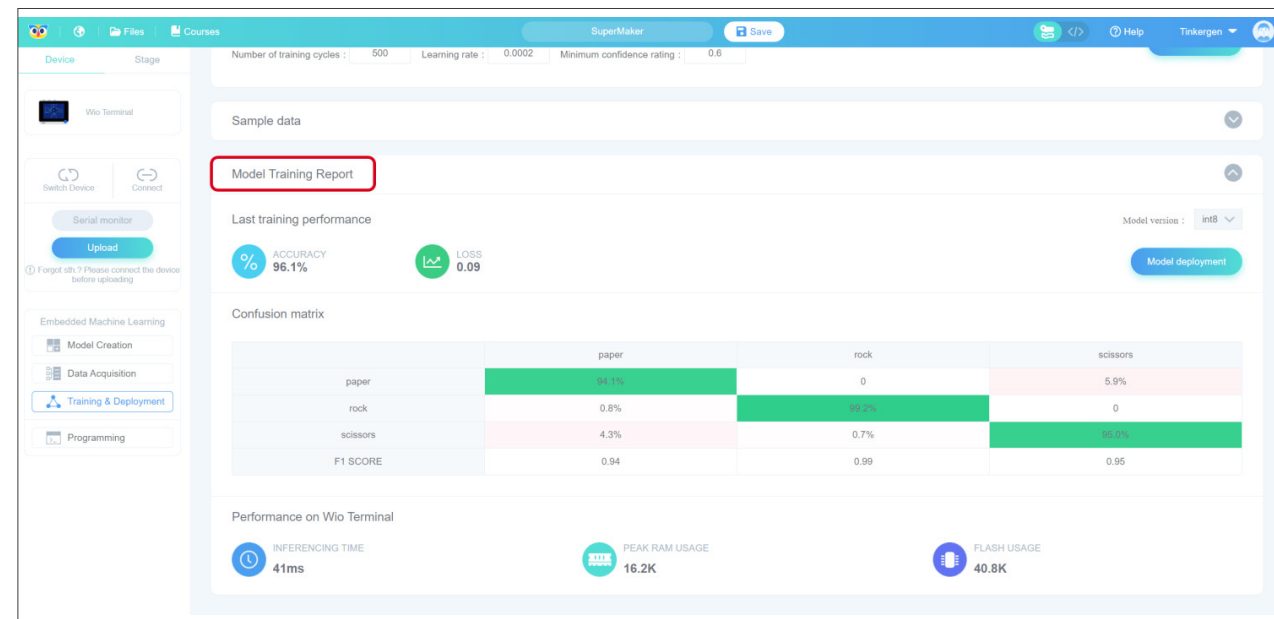
You can also infer the waiting time by observing the “Log”. In the figure below, “Epoch: 68/500” indicates the total number of training rounds is 500 while 68 rounds have been trained.




3.3 Observe the model performance to select the ideal model

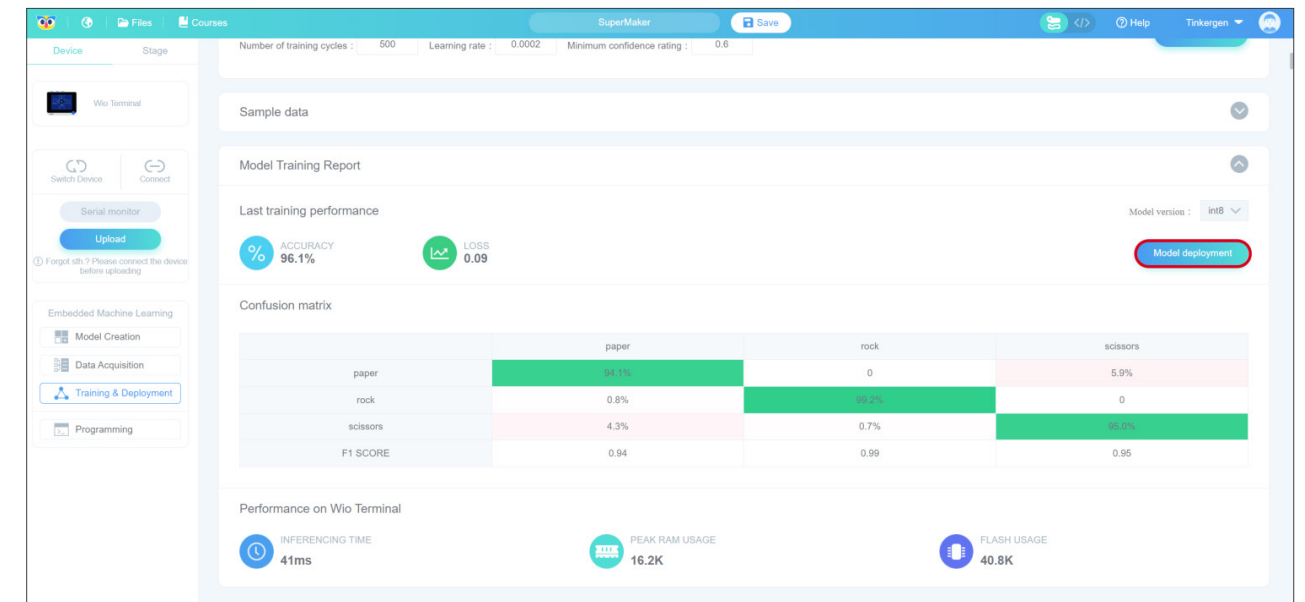
In the “Model Training Report” window, you can observe training results including the accuracy, loss and performance of the model.

If the training results are not satisfactory, you can go back to the first step of training the model, select another size of the neural network or adjust the parameters and train it until you get a model with satisfactory results.

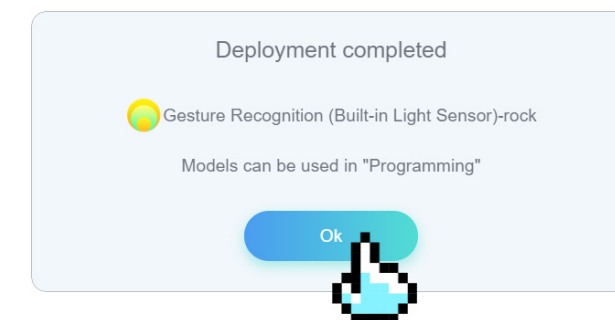


3.4 Deploy the ideal model

In the “Model Training Report” window, click  Model deployment



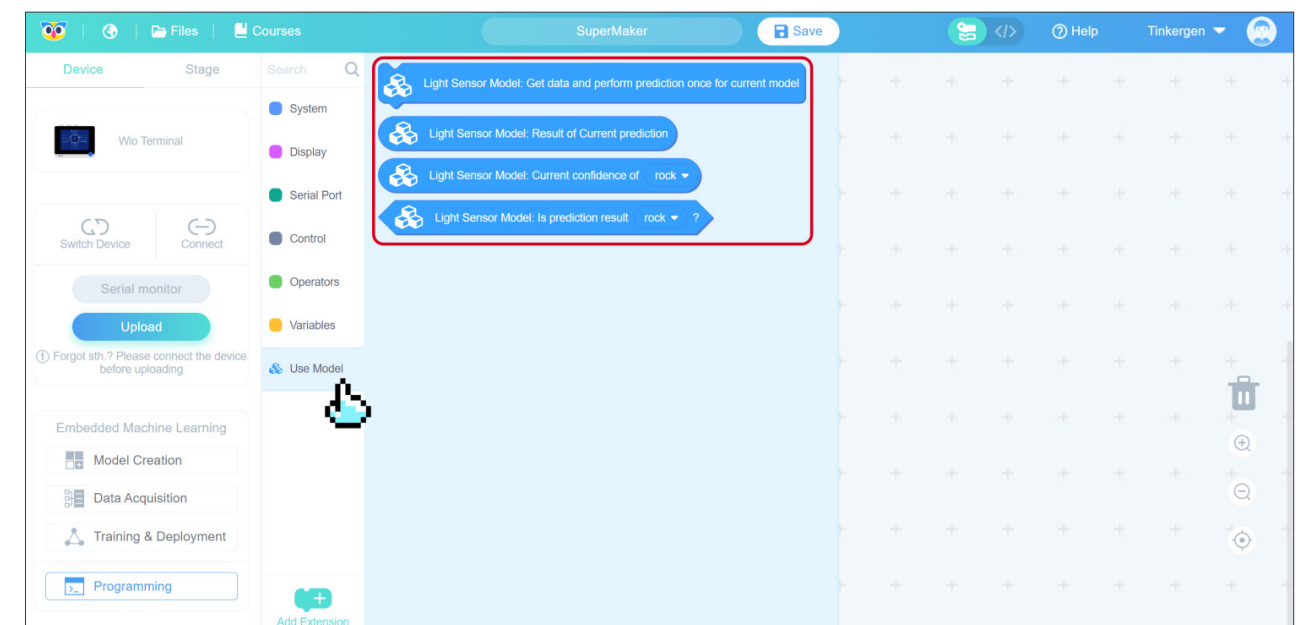
Once the deployment is completed, Click “Ok” to jump to the Programming window.



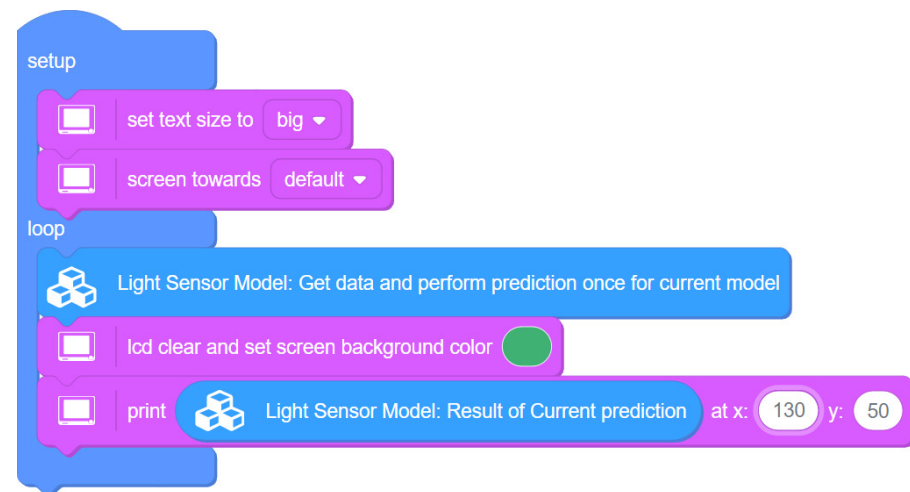
Step 4. Use and programming

4.1 Write the program for using the model

In the “Programming” interface, click on “Use Model” to use the deployed model.

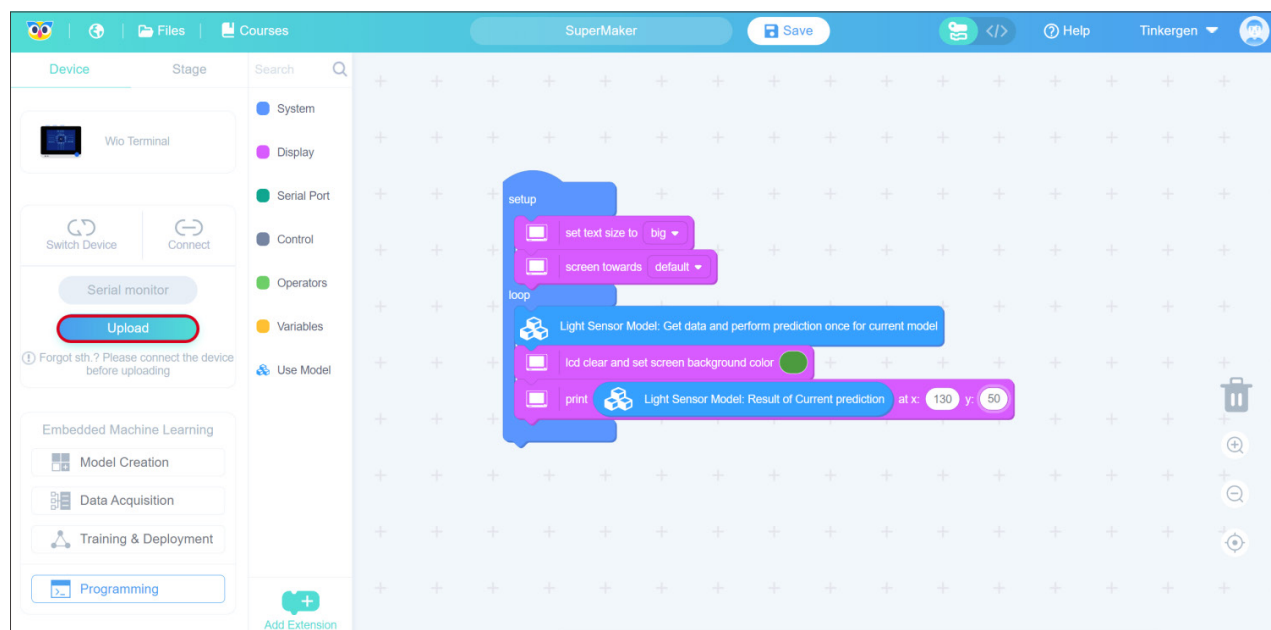


Try to use your model by writing the following programme.

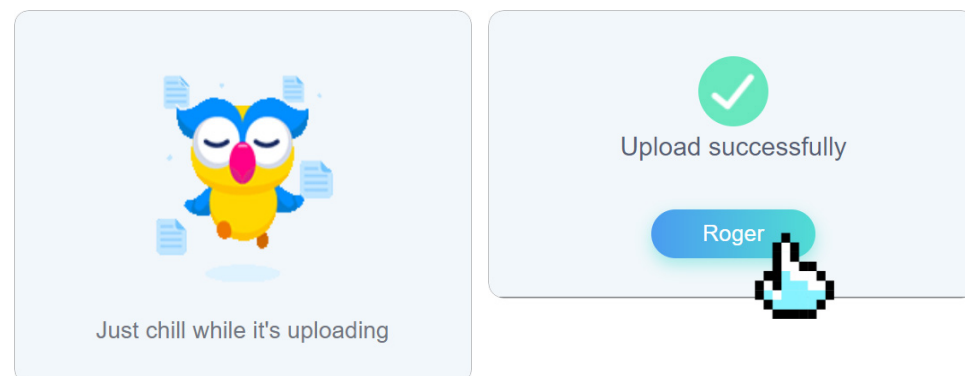


4.2 Upload the program to Wio Terminal

Click the “Upload” button.



The first upload time usually cost longer and it increases with the complexity of the model. The uploading time for smaller models cost about 4 minutes or even longer (depending on the performance of your machine).



4.2 Wio Terminal test model

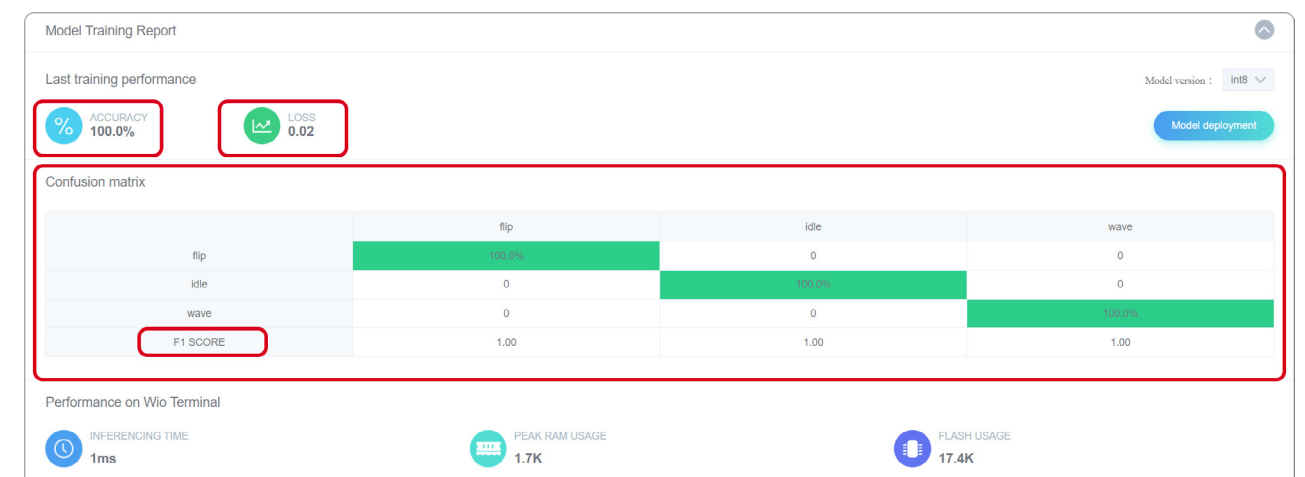
Make a gesture “scissors” to see if Wio Terminal’s screen can shows “scissors”. Try other gestures, and see if the Wio Terminal can recognize your gestures.

Congratulations! You have completed your TinyML model, I believe you are familiar with your input(dataset and labels) and know your output. A good score of “Accuracy” leads you to draw a conclusion that your model is working perfectly but it is not enough for you to evaluate the performance of your model. So, let’s take a look at the output.

ML Theory(understand your output):

Output (the training performance):

- ACCURACY
- LOSS
- Confusion matrix
- F1 SCORE



1. Accuracy

Machine learning model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training data. It is typically expressed in percentage value. Accuracy is the count of predictions where the predicted value is **equal to the true value**. It is often graphed and monitored during the training phase as the value is often associated with the overall or final model accuracy.

2. Loss

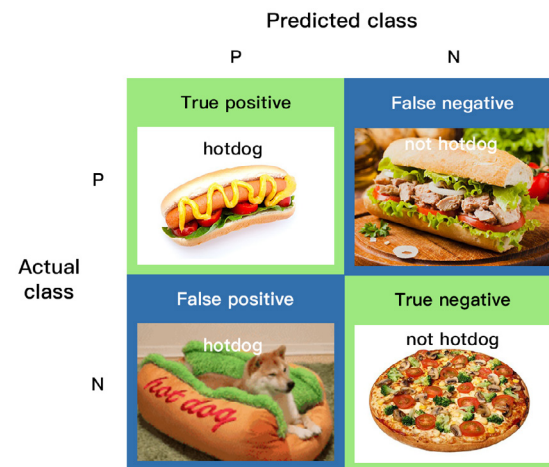
Unlike accuracy, loss is not a percentage. It is the sum of the errors that occur for each sample in the training or validation set. During the training process, the goal is to minimize this value. The loss function is usually used during training to find the “best” parameter values for the model.

A loss function determines the probability or uncertainty of a prediction based on how much prediction **varies from the true value**. If predictions deviate too much from actual results, loss function would cough up a very large number.

Relationship Between Accuracy and Loss

Accuracy and loss appear to be inversely proportional, as we can often observe that accuracy increases with the decrease in loss, but in reality, there is no mathematical relationship between the two. Accuracy and loss have different definitions and they measure different parameters. When the loss decreases the accuracy may also decrease.

3. Confusion Matrix



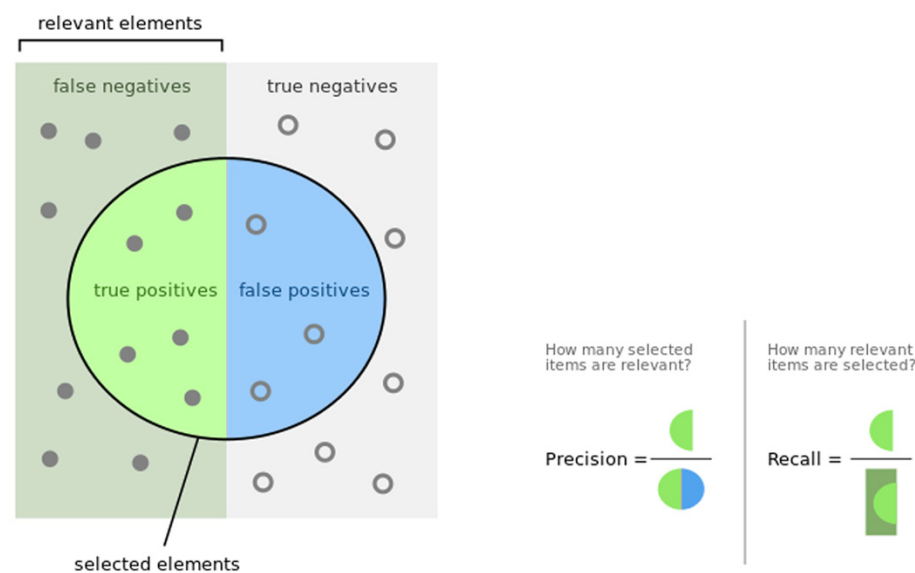
Confusion matrix

- A matrix that lays out the performance of a learning algorithm
- The confusion matrix is simply a square matrix that reports the counts of
 - True Positive (TP)
 - True Negative (TN)
 - False Positive (FP)
 - False Negative (FN)
 for predictions of a classifier

		Predicted class	
		P	N
Actual class	P	True positives (TP)	False negatives (FN)
	N	False positives (FP)	True negatives (TN)

2.4. F1 SCORE

The F1-score is a measure of a model's accuracy on a dataset.



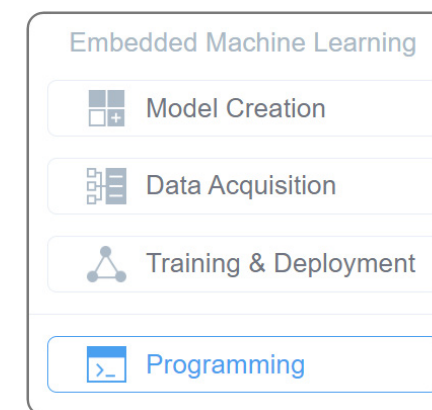
Why F1 in this way?

$$F1 = 2 \frac{PRE \times REC}{PRE + REC} = \frac{2}{1/PRE + 1/REC}$$

- Ideal case: PRE=1, REC=1, F1=1
- $0 \leq F1 \leq 1$
- A good F1 score means that the model has low FP and low FN, so it correctly identifies real threats and is not disturbed by false alarms.

★ Summarization

1. Theory: Light sensor
2. TinyML practice



3. ML theory (Output/the training performance)

- Accuracy: A method of measuring the performance of a model.
- Loss: A loss function, also known as a cost function determines the probability or uncertainty of a prediction based on how much prediction varies from the true value.
- Confusion Matrix: A matrix that lays out the performance of a learning algorithm

• F1 Score: $F1 = 2 \frac{PRE \times REC}{PRE + REC} = \frac{2}{1/PRE + 1/REC}$

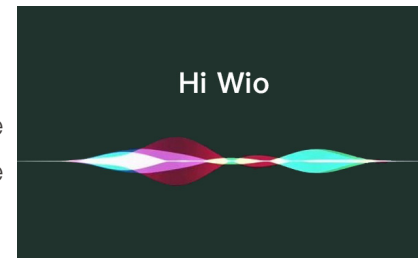
Lesson 04

Wake-up Words Recognition
using built-in Microphone



Project Overview

You might be familiar with “Hi Siri”, but how does the phone can listen to our voice and give us feedback? In this lesson, we will try to make our own “Hi Wio”.



This model uses Wio Terminal built-in microphone to collect vocal wake words and ambient sounds to train the model. This microphone also helps us to wake up the device with the wake-up word (“Hi Wio”).

Audio scene classification is a task where machine learning model needs to predict a class for audio segment. After this lesson, we will be able to build models to classify other kinds of audio, for example, “a crying baby”, “a cough”, “a dog barking”, etc.

Expected results

The desired effect is as follow: when “Hi Wio” is shouted at the Wio Terminal, the Wio Terminal could respond to you in real-time.

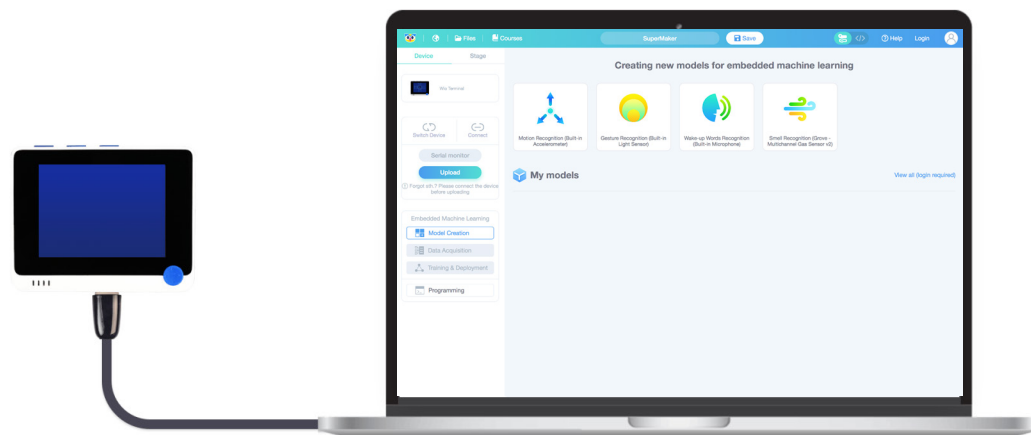


Material Preparation

To achieve the above results, we need:

Hardware requirements: Wio Terminal

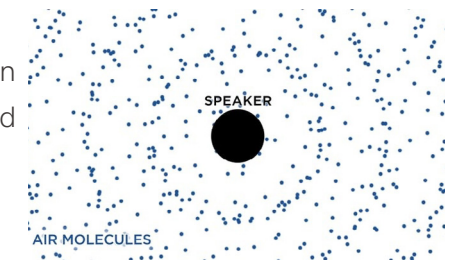
Connection method:



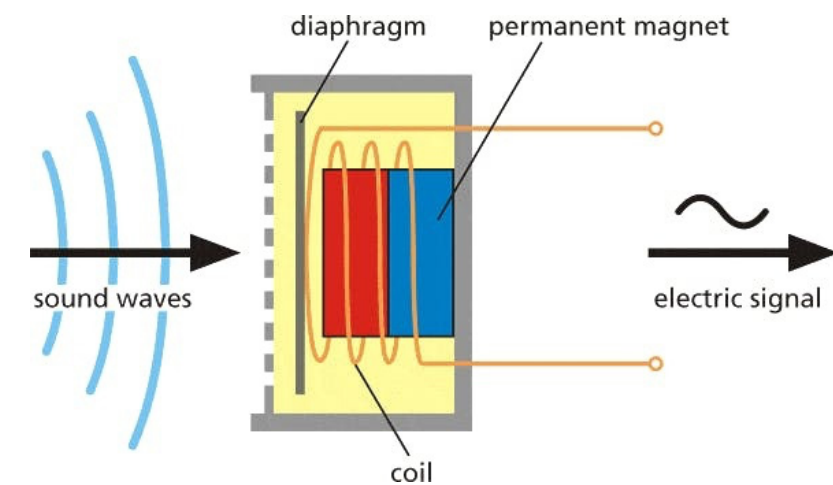
Theory

Let’s learn more about sound processing in computers.

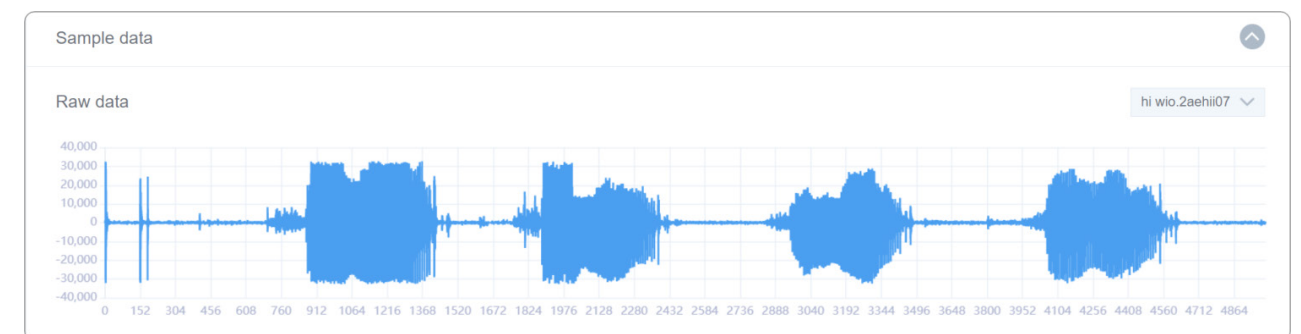
The sound is vibrated that propagates (or travels) as an acoustic wave through a transmission medium such as gas, liquid and solid.



The source of sound pushes the surrounding medium molecules, they push the molecules next to them, and so on. Upon reaching other objects, they vibrate slightly. We use this principle to microphones. The microphone membrane is pushed inward by the medium molecules and then goes back to its original position.



It generates an alternating current in the circuit, where voltage is proportional to the amplitude of the sound. The louder sound is, the more times membrane it pushes which it generates higher voltages at the output. We then read this voltage with an analog-to-digital converter and record at equal intervals.

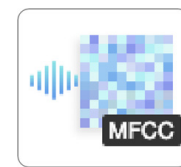
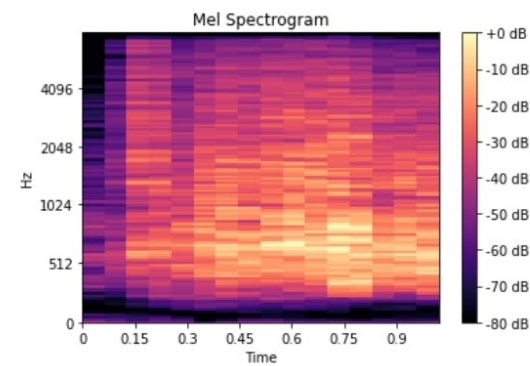
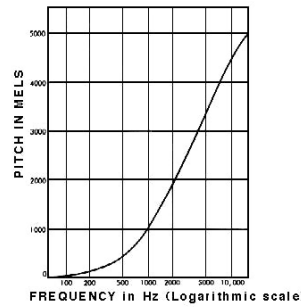


We can not do much with this raw sound representation but we can cut and process it into several parts or make it quieter or louder. In order to analyze the sound, it is, well, too raw.

Normal raw data contains a lot of information. Studies have shown that humans can not perceive frequencies on a linear scale. Humans are better at detecting differences in lower frequencies than higher frequencies. For example, we can easily tell the difference between 500 and 1000 Hz but we will hardly be able to tell a difference between 10000 and 10500 Hz. Even though the distance are the same.

In 1937, Stevens Volkman and Newmann proposed a unit of pitch such that equal distances in pitch sound equally distant to the listener. This is called the mel scale.

Mel Frequency Cepstral Co-efficients (MFCC) is an internal audio representation format which is easy to work with. This is similar to JPG format for images.



It is time to finally start with practical implementation.

Practice

Project Steps

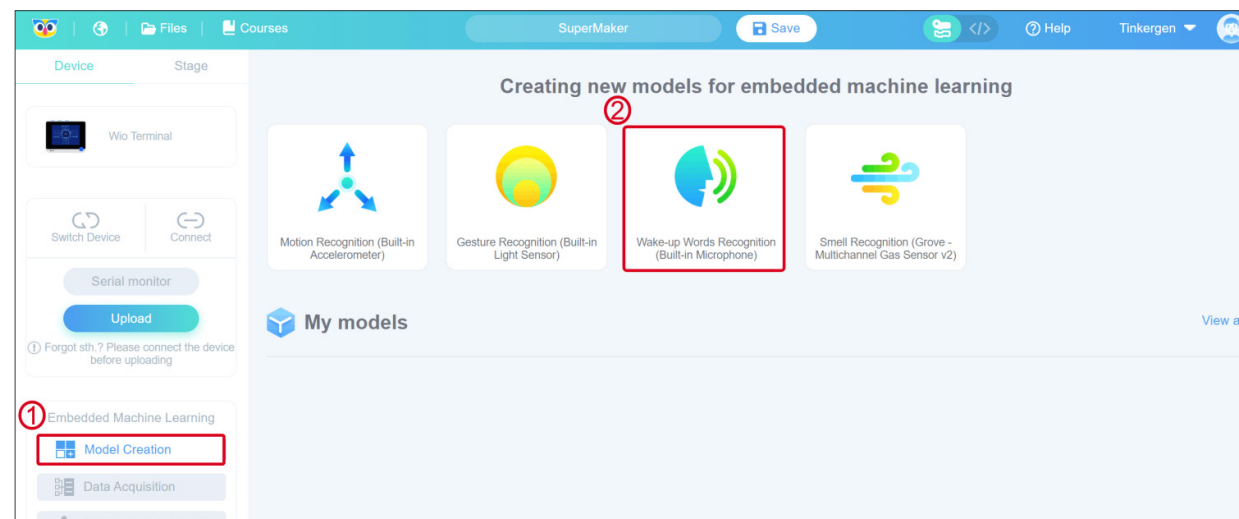
1. Creating and Selecting Models
2. Data Acquisition
3. Training and Deployment
4. Programming

Please Open <https://ide.tinkergen.com/>

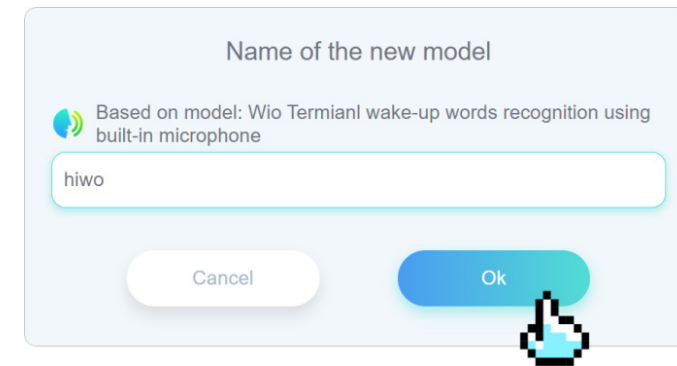
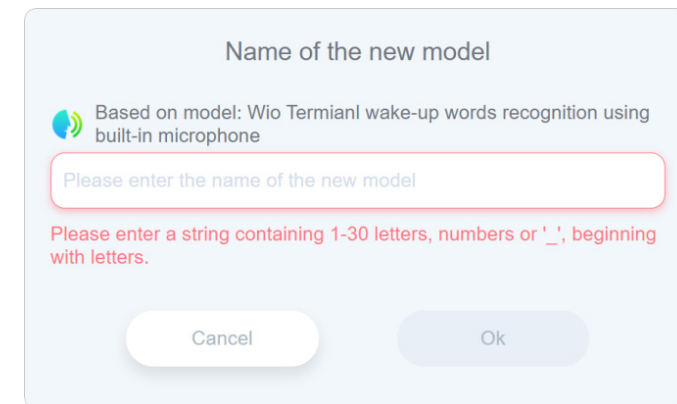
Step 1. Create and select models

1.1 Create a “Wake-up Words Recognition(Built-in Microphone)” model

Click on “Create and select model”, click on “Wake-up Words Recognition(Built-in Microphone)”, as shown in steps 1 and 2 below.



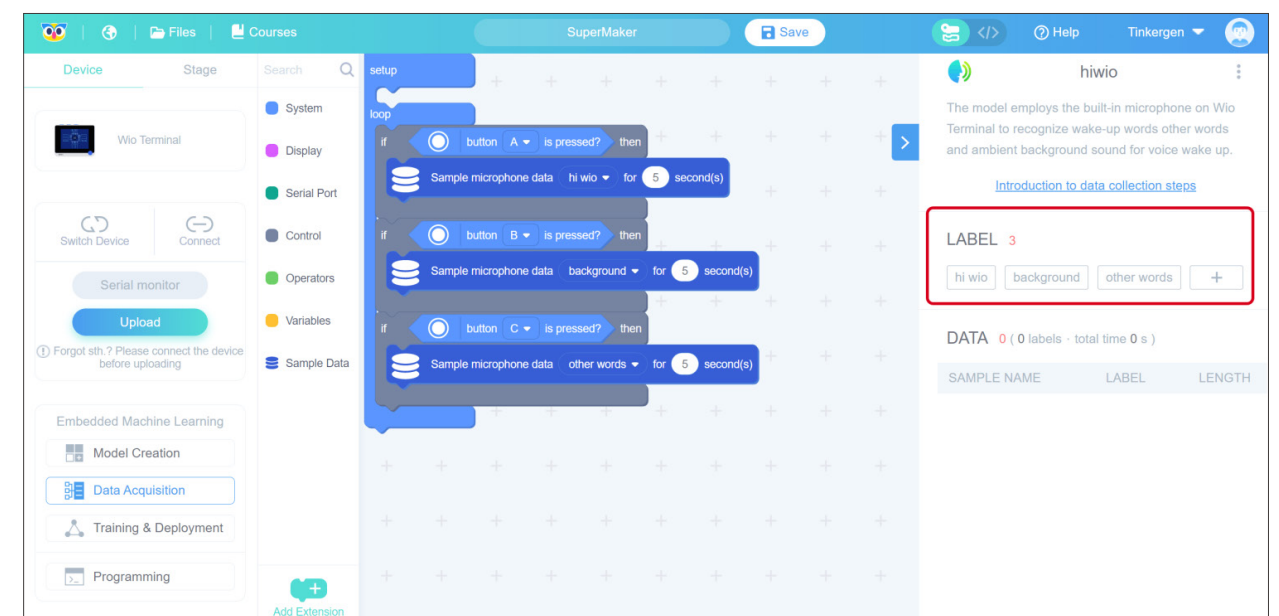
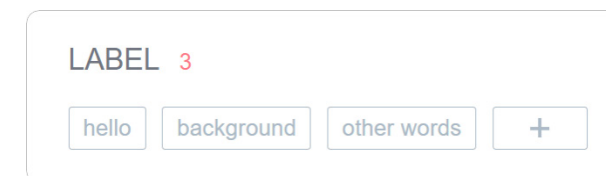
Enter a NAME according to the requirements.



Click Ok and it will automatically jump to the Data Acquisition interface.

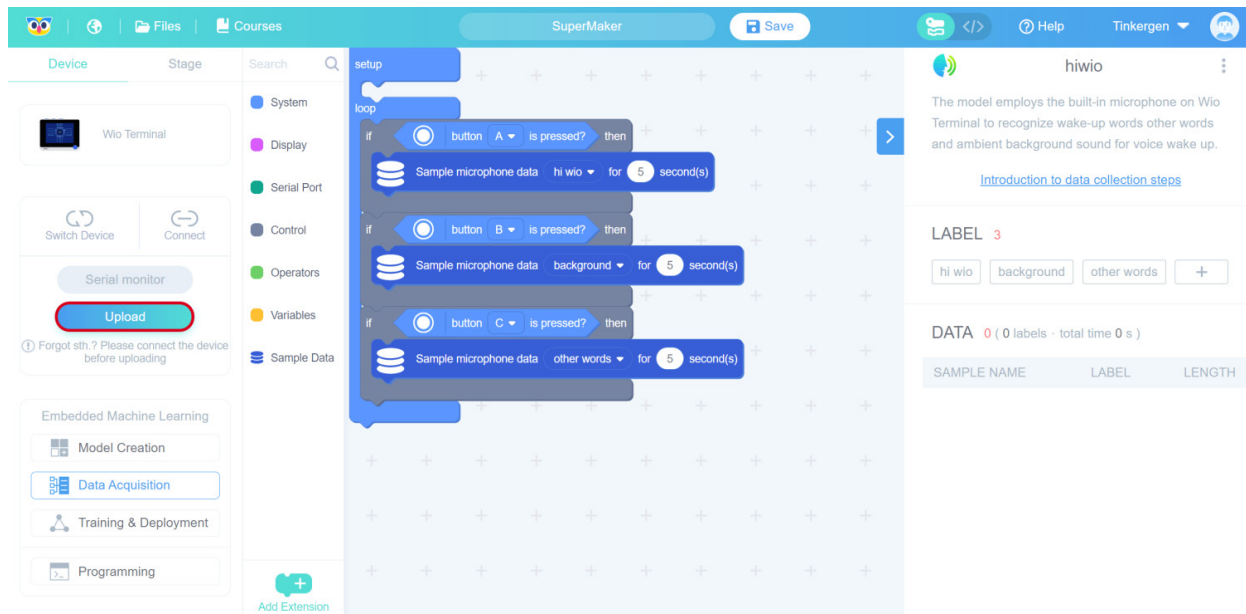
Step 2. Acquisition of data

2.1 Default label

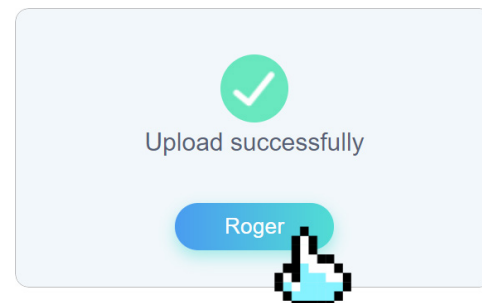


2.2 Connect the device and upload the default data acquisition program in Codecraft

When the Wio Terminal is connected, click **Upload** in the Codecraft window. This action will upload the default data acquisition programme.



Usually, it takes 10 seconds to upload. Once the program is uploaded, the “Upload Successful” window will appear on the screen shown below. This is shown in the image below.



Click “Roger” to close the upload success pop-up window above and return to the programming screen.

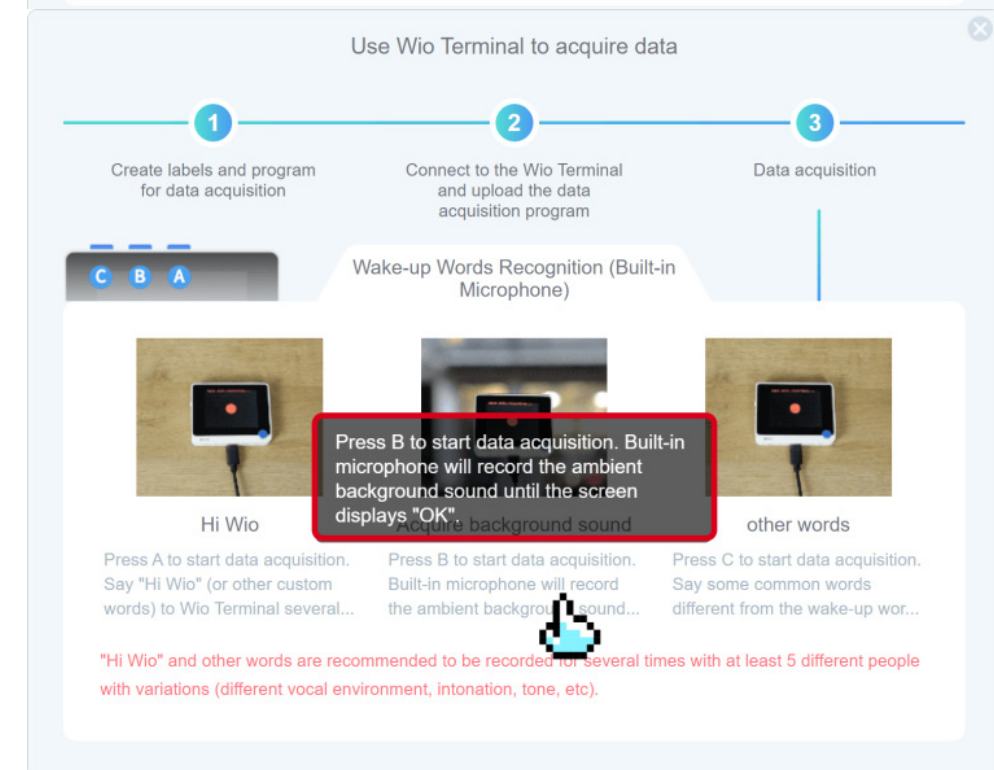
2.3 Acquisition of data

In the upper right hyperlink, you will find a step-by-step introduction to data acquisition. Follow the instructions to collect data.





Attention:

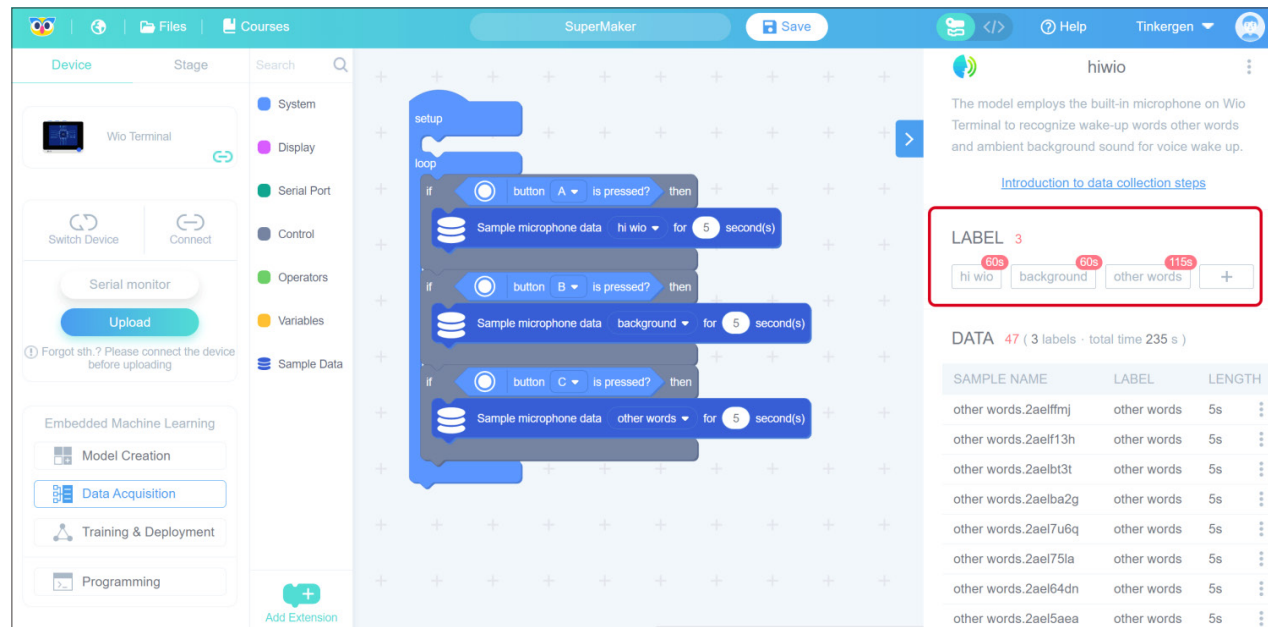
- Say words without interruption when recording with the labels “Hi Wio” and “other words” to ensure that the audio is as more different as possible as compared to other labels “background sound”.
- Wio Terminal button location.
- Animated GIF has been accelerated, the actual action can slightly slow down.
- Notice the red tips.
- Point the cursor over Description Texts for more detailed content.



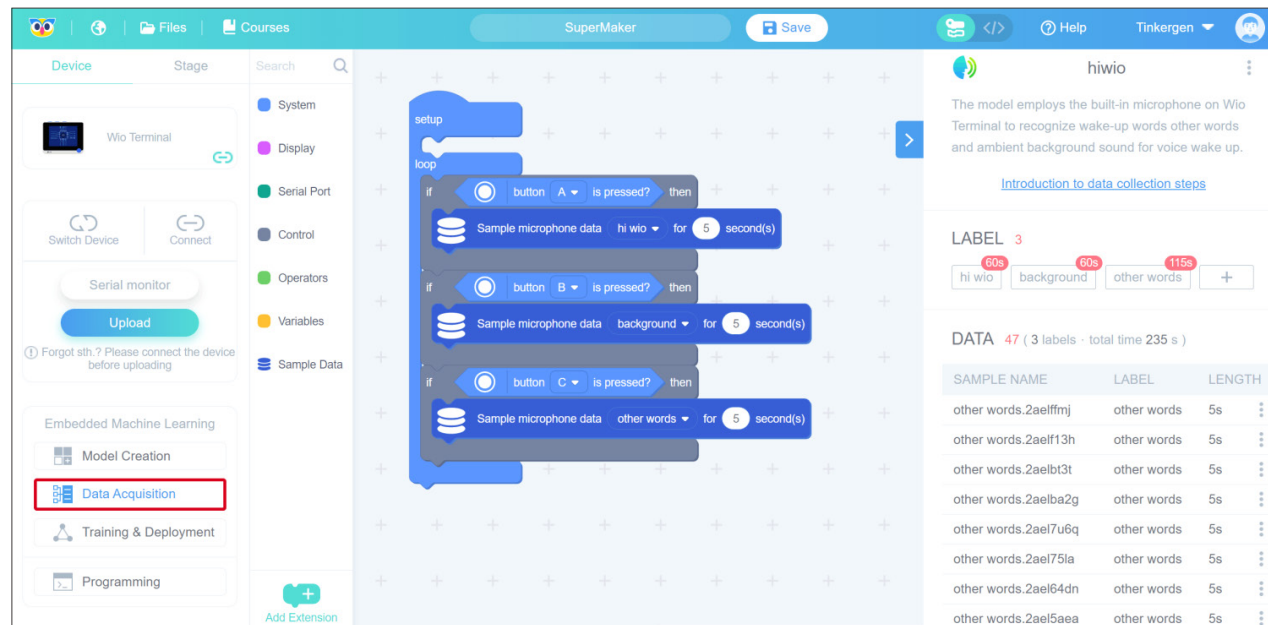
Start and finish data acquisition according to the Wio Terminal display.

-  This signal means data is being collected.
-  OK means the collection is complete.

Now, the data acquisition is finished.



Click on “Training & Deployment”



Step 3. Training and deployment

3.1 Set neural network and parameters

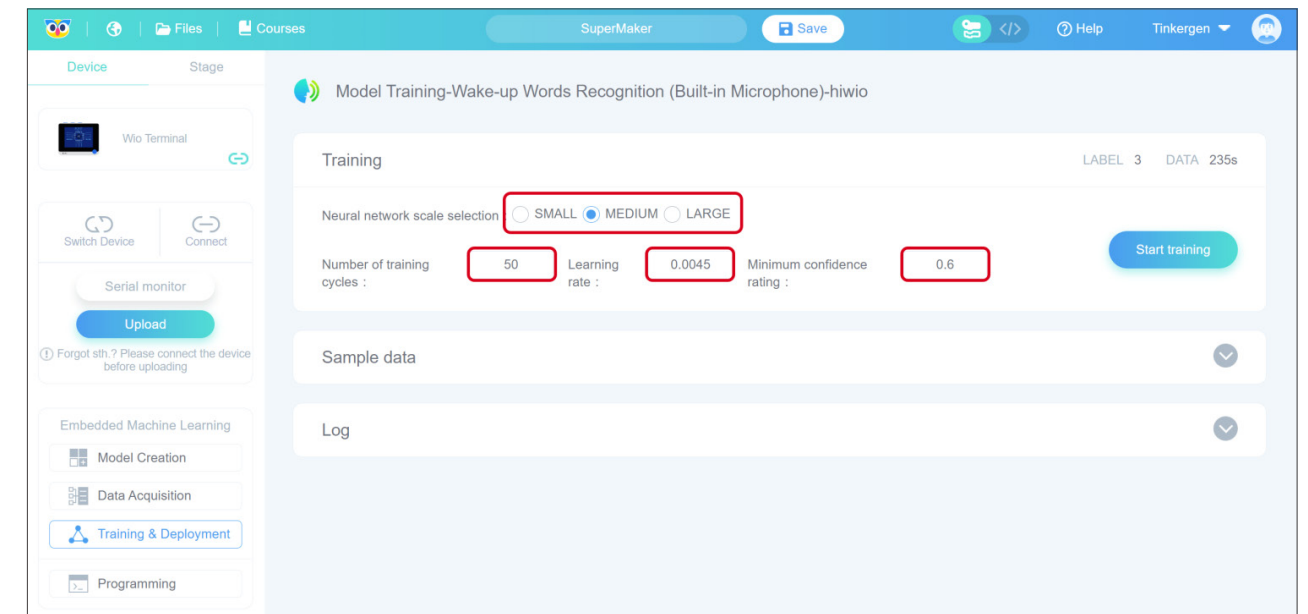
Select the suitable neural network size: one of small, medium and large

Set parameters, number of training cycles (positive integer), learning rate (number from 0 to 1)

and minimum confidence rating(number from 0 to 1).

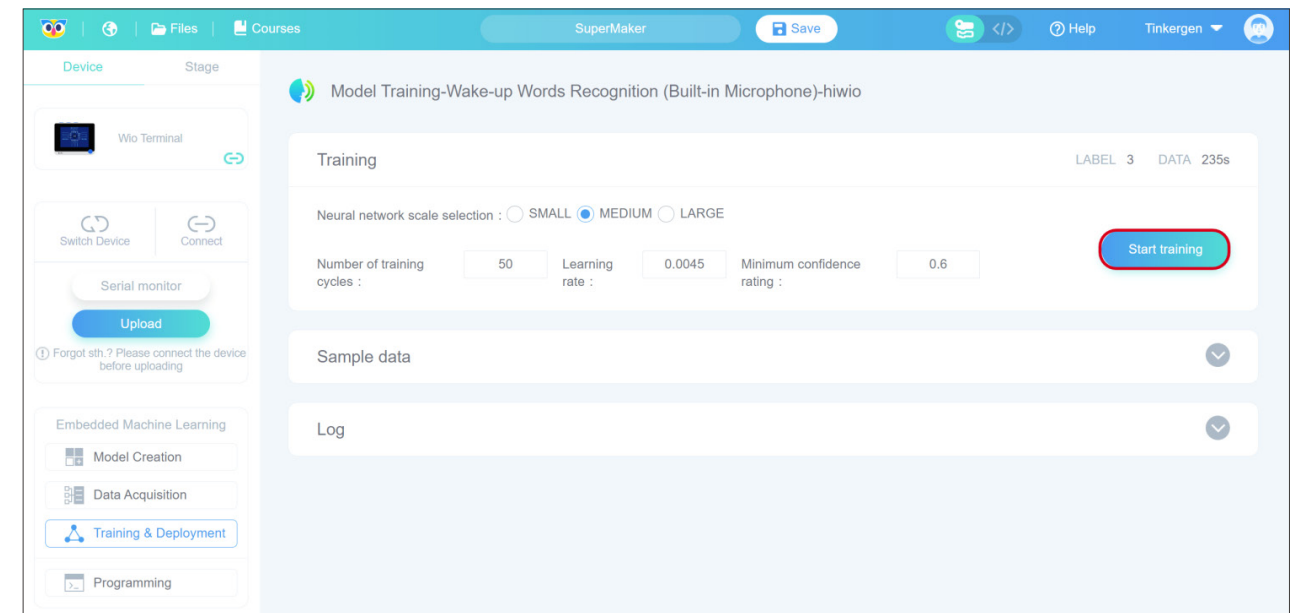
The interface provides default parameter values.

In this case, we are using MEDIUM. It will take quite a long time.

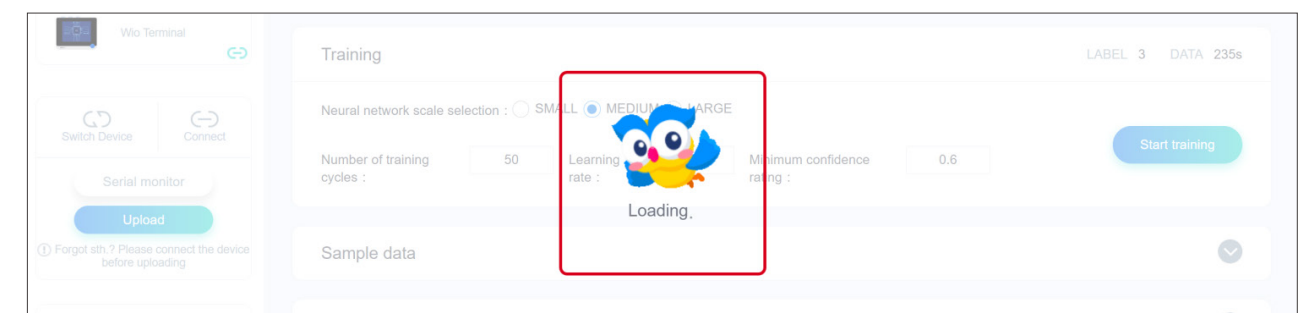


3.2 Start training the model

Click “Start training”



When you click “Start training”, the interface displays “Loading...”.



The duration of “Loading..” varies depending on the size of the selected neural network (small, medium and large) and the number of training cycles. The larger network size is and the more number of training cycles are, the longer it will take.

You can also infer the waiting time by observing the “Log”. In the figure below, “Epoch: 7/50” indicates the total number of training rounds is 50 while 7 rounds have been trained.

The screenshot shows the SuperMaker interface with the 'Model Training Report' window open. The 'Log' section at the bottom is highlighted with a red box, displaying the following text:

```
Epoch 8/50
49/49 - 1s - loss: 0.6209 - accuracy: 0.7138 - val_loss: 0.4930 - val_accuracy: 0.8601
Epoch 7/50
```

3.3 Observe the model performance to select the ideal model

In the “Model Training Report” window, you can observe training results including the accuracy, loss and performance of the model.

If the training results are not satisfactory, you can go back to the first step of training the model, select another size of the neural network or adjust the parameters and train it until you get a model with satisfactory results.

The screenshot shows the SuperMaker interface with the 'Model Training Report' window open. The window is highlighted with a red box. The report displays the following information:

- Neural network scale selection: SMALL MEDIUM LARGE
- Number of training cycles: 50
- Learning rate: 0.0045
- Minimum confidence rating: 0.6
- Last training performance: ACCURACY 95.1%, LOSS 0.15
- Confusion matrix:

	background	hi wio	other words
background	97.8%	2.2%	0
hi wio	0.9%	99.1%	0
other words	1.6%	7.1%	91.2%
F1 SCORE	0.97	0.93	0.95
- Performance on Wio Terminal: INFERRING TIME 7ms, PEAK RAM USAGE 8.1K, FLASH USAGE 31.9K

3.4 Deploy the ideal model

In the “Model Training Report” window, click [Model deployment](#)

The screenshot shows the SuperMaker interface with the 'Model Training Report' window open. The 'Model deployment' button is highlighted with a red box. The report displays the following information:

- Neural network scale selection: SMALL MEDIUM LARGE
- Number of training cycles: 50
- Learning rate: 0.0045
- Minimum confidence rating: 0.6
- Last training performance: ACCURACY 95.1%, LOSS 0.15
- Confusion matrix:

	background	hi wio	other words
background	97.8%	2.2%	0
hi wio	0.9%	99.1%	0
other words	1.6%	7.1%	91.2%
F1 SCORE	0.97	0.93	0.95
- Performance on Wio Terminal: INFERRING TIME 7ms, PEAK RAM USAGE 8.1K, FLASH USAGE 31.9K

Once the deployment is complete, click “Ok” to jump to the “Programming” window

The screenshot shows a dialog box titled 'Deployment completed' with the following text:

Wake-up Words Recognition (Built-in Microphone)-hiwio
Models can be used in "Programming"

The 'Ok' button is highlighted with a red box.

Step 4. Use and programming

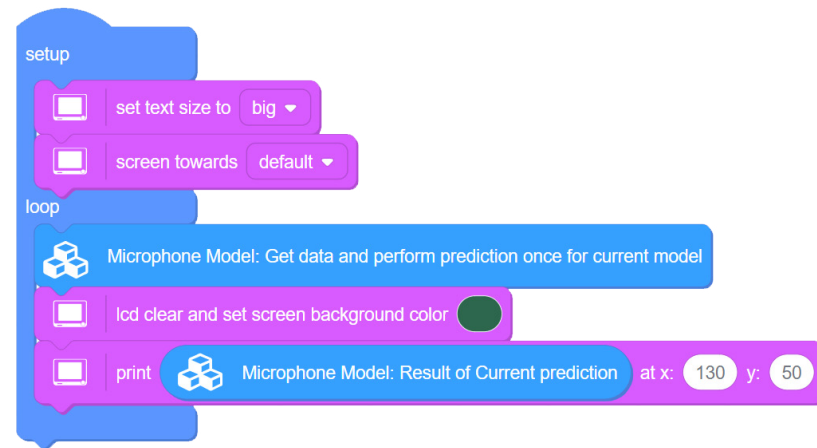
4.1 Write the program for using the model

In the “Programming” interface, click on “Use Model” to use the deployed model.

The screenshot shows the SuperMaker interface with the 'Programming' window open. The 'Use Model' button is highlighted with a red box. The programming area contains the following code:

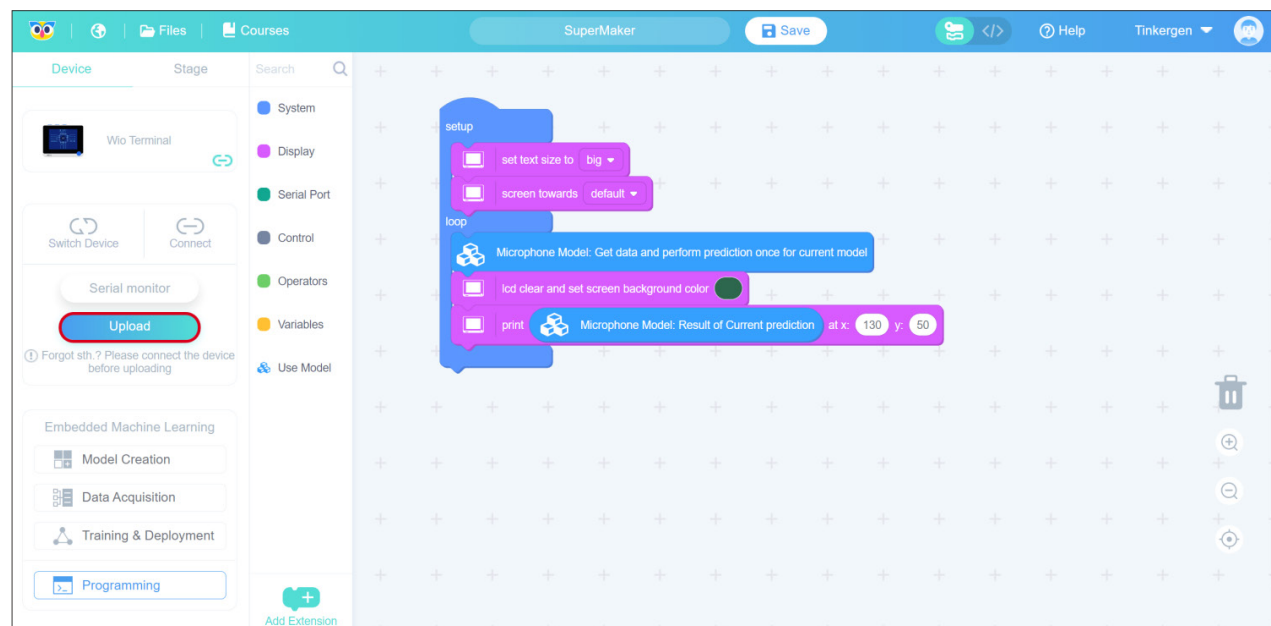
```
Microphone Model: Get data and perform prediction once for current model
Microphone Model: Result of Current prediction
Microphone Model: Current confidence of hi wio
Microphone Model: Is prediction result hi wio ?
```

Try to use your model by writing the following program.

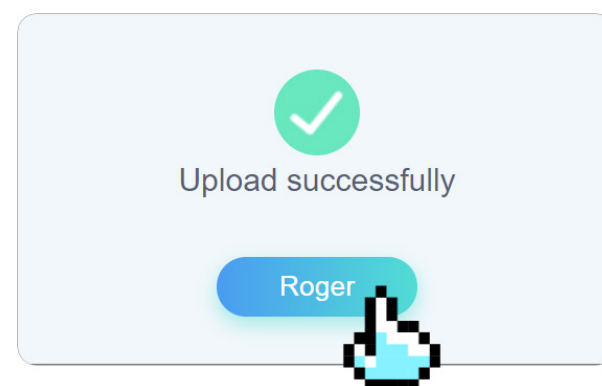
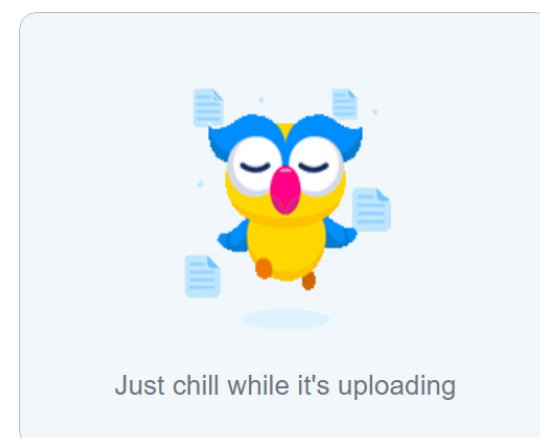


4.2 Upload the program to Wio Terminal

Click the "Upload" button.



The first upload time usually cost longer and it increases with the complexity of the model. The uploading time for smaller models cost about 4 minutes or even longer (depending on the performance of your machine).



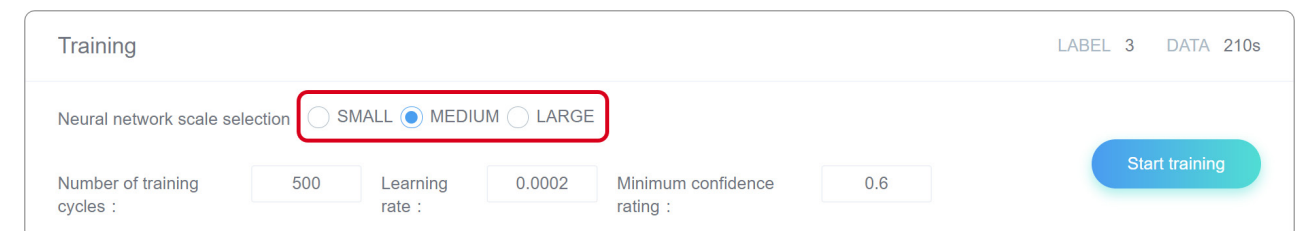
4.3 Wio Terminal test model

Try to say "Hi Wio" to your Wio Terminal to see whether it has been waked up.

Congratulations! You have completed your third TinyML model. I believe you have tried different scale models so as to get a better model performance. Let's take a look inside the model and deeply understand the "difference".

ML Theory(know the different scale models):

- Artificial Neural Network Architecture
 - a. Nodes
 - b. Layer
 - c. ANN
- Different scale Neural Network
 - a. Size
 - b. Width
 - c. Depth
 - d. Capacity
 - e. Architecture



1. Artificial Neural Network Architecture (ANN)

1.1 What is Nodes

Neuron nodes are the processing units of the network and inspired by neurons in the human brain. Neurons in deep learning models are nodes through which data and computations flow in order to build computer algorithms that behave similarly to neurons in the brain.

1.2 What is Layer

In deep learning models, layer is a structure or network topology in the architecture of the model which takes information from the previous layers and then passes information to the next layer.

1.3 What is ANN?

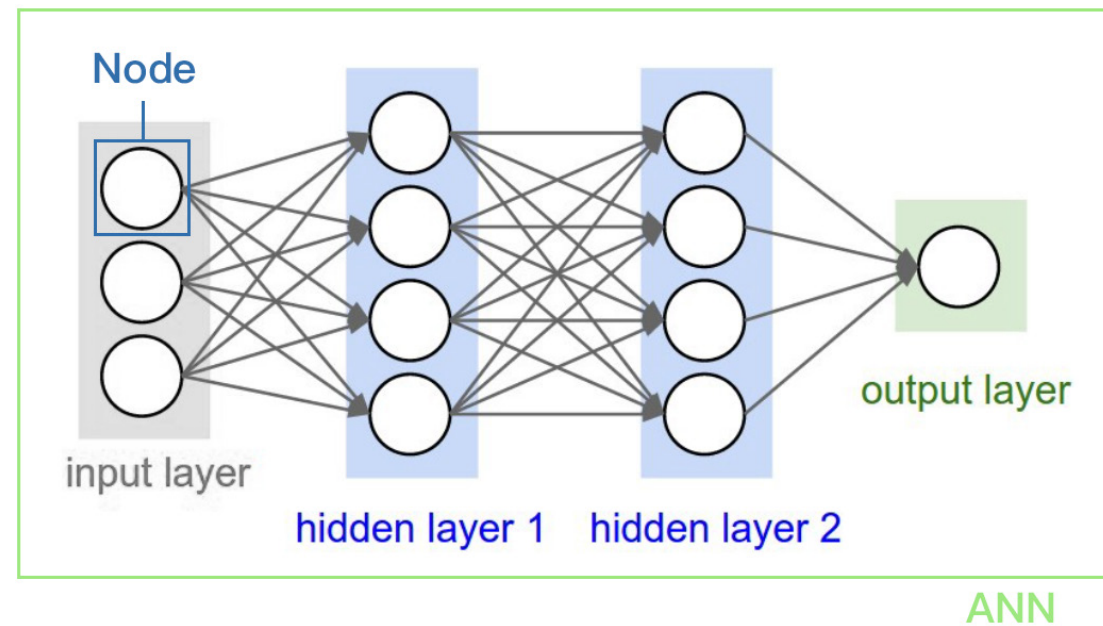
ANN is made of three layers namely input layer, output layer and hidden layer/s. There must be a connection from the nodes of the input layer with the nodes of the hidden layer and from each hidden layer node with the nodes of the output layer.

We can summarize the types of layers as follows:

- **Input Layer:** Input variables, sometimes called the visible layer. The input layer takes data from the network.
- **Hidden Layers:** Layers of nodes between the input and output layers. There may be one

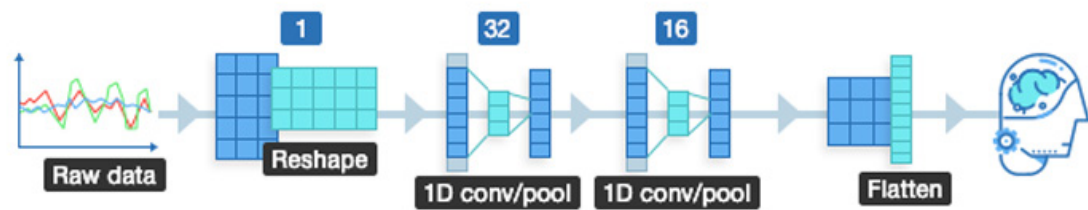
or more of these layers. The hidden layer receives the raw information from the input layer and processes them. The obtained value is transferred to the output layer.

- **Output Layer:** A layer of nodes that produces the output variables.

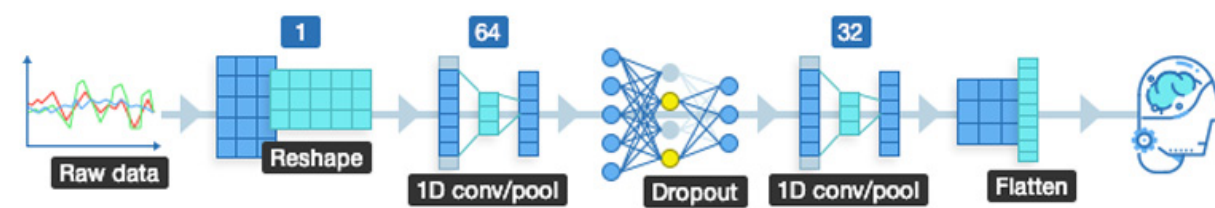


2. Different scales of Neural Network

SMALL



MEDIUM

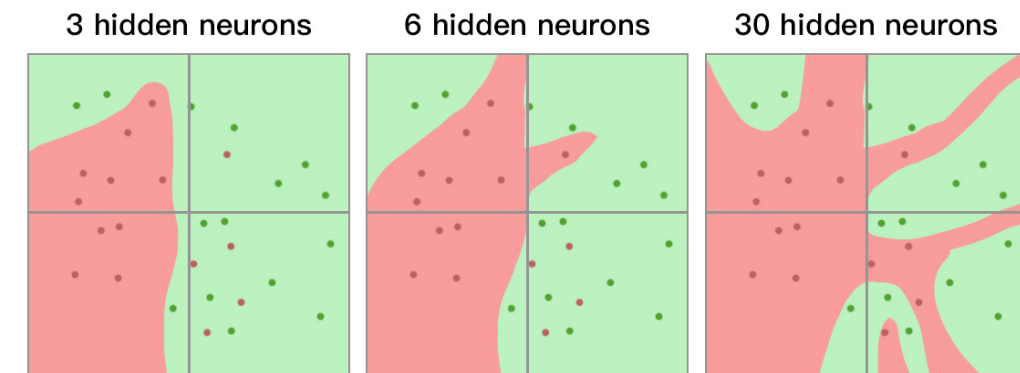


2.1 Size: The number of nodes in the model.

2.2 Width: The number of nodes in a specific layer.



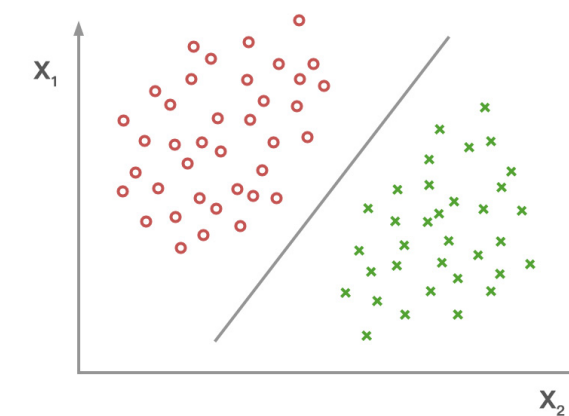
The following Image tells us the “Width” plays a significant role in a model.



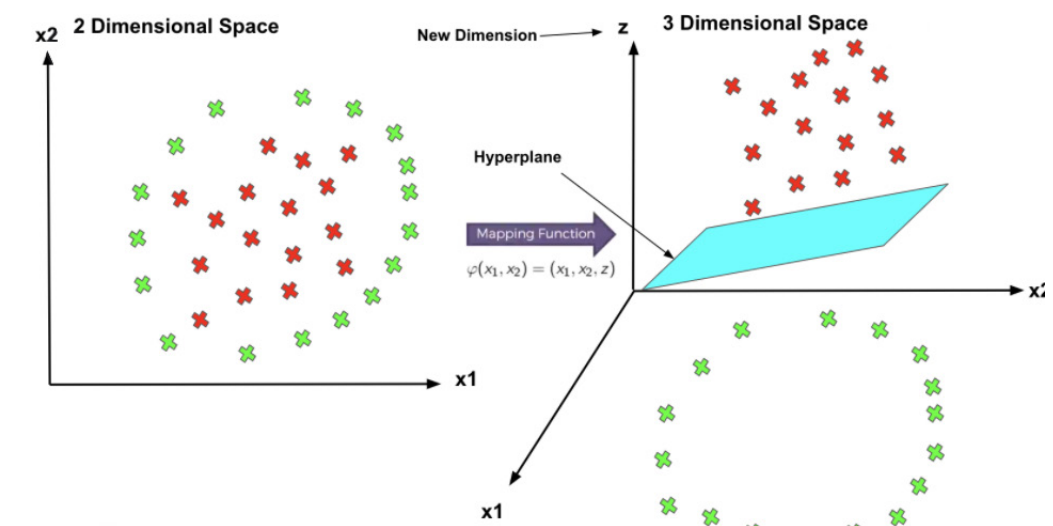
2.3 Depth: The number of layers in a neural network.

2.3.1 Why to Have Multiple Layers?

A single-layer neural network can only be used to represent linearly separable functions. It means two classes in a classification problem can be neatly separated by a line.



If your problem is relatively simple, perhaps a single-layer network would be sufficient. Most problems that we are interested in are not linearly separable.



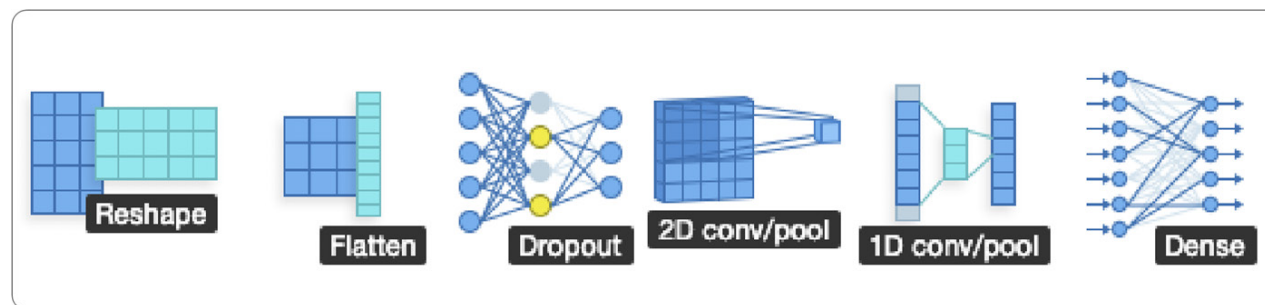
Multilayer perceptrons (MLPs) can be used to represent convex shaped regions. This means that they can learn to draw shapes around some examples in high-dimensional space, separating and classifying them by overcoming the limitations of linear separability.

(In fact, a 1987 paper by Lippmann, "Introduction to Computation with Neural Networks," has a theoretical finding that shows that MLPs with two hidden layers are sufficient to create classified regions of any desired shape. This is illuminating, although it should be noted that there is no indication of how many nodes are used at each layer, nor is it given how the weights are learned.

Further theoretical findings and proofs show that MLPs are general-purpose approximators. This means that using a hidden layer, MLPs can approximate any function we require).

2.4 Capacity: The type or structure of a function that can be learned by a network configuration. Sometimes called "representational capacity".

2.5 Architecture: The specific arrangement of the layers and nodes in the network. The different types of layers are: Reshape layer, Flatten layer, Dropout layer, Conv2D/pool layer, Conv1D/pool layer, Dense layers, etc



2.5.1 Why to Have Different Types Of Layers?

Different layers perform different transformations on their inputs. Some layers are better suited for specific tasks than others.

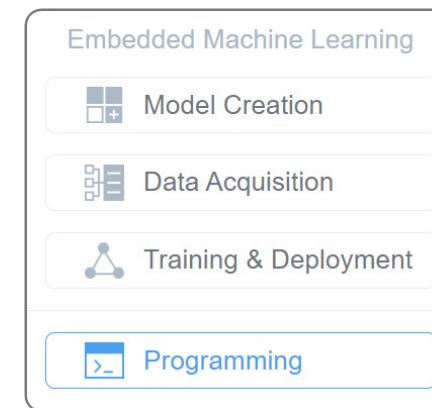
For example, a convolutional layer is usually used in models that deal with image data while Recurrent layers are used in models that work with time-series data. Fully connected layers connect each input to each output within its layer.

More details of layers are mentioned in Lesson 8.

In conclusion, SMALL, MEDIUM, and BIG models have different numbers of the hidden layer, types of layers, and num of neurons.

★ Summarization

1. Theory: Microphone
2. TinyML practice

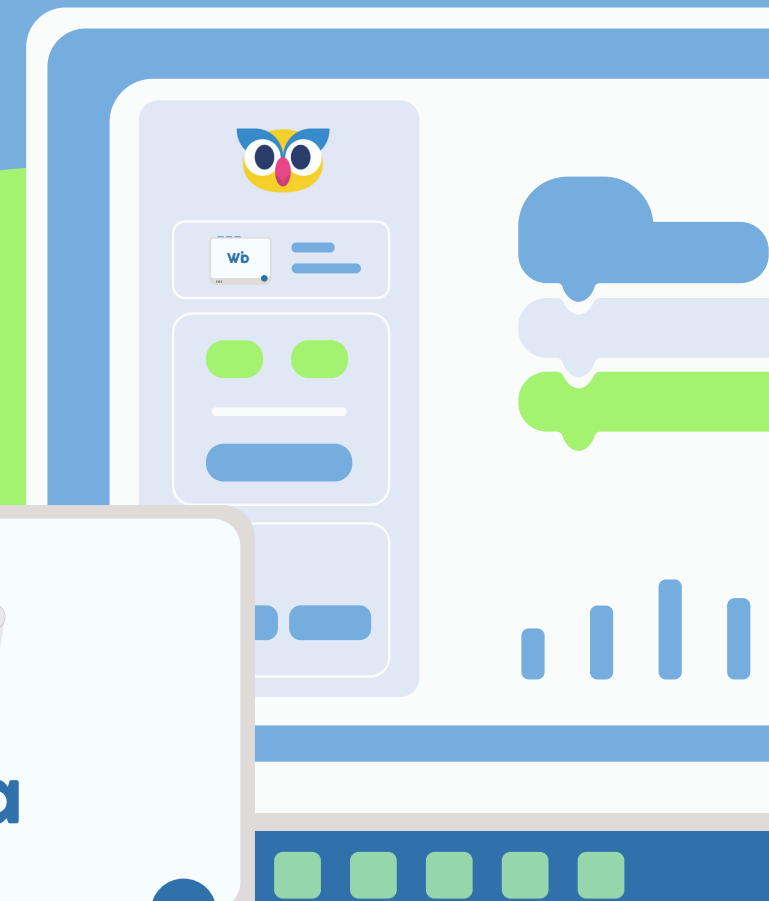
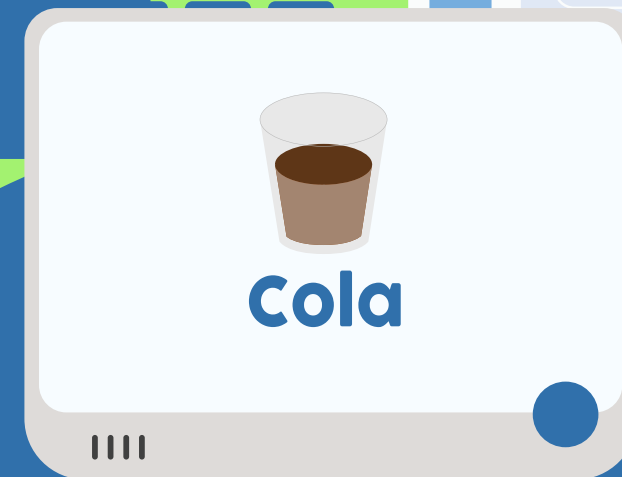
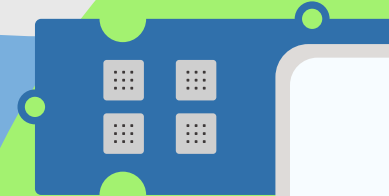


3. ML theory (Model scale)

- Artificial Neural Network Architecture
 - a. Nodes: Neuron nodes are processing units of the network.
 - b. Layer: In Deep Learning models, layer is a structure or network topology in the architecture of the model which takes information from the previous layers and then passes information to the next layer.
 - c. ANN: ANN is made of three layers namely input layer, output layer and hidden layer/s.
- Different scale Neural Network
 - a. Size: The number of nodes in the model.
 - b. Width: The number of nodes in a specific layer.
 - c. Depth: The number of layers in a neural network.
 - d. Capacity: The type or structure of a function that can be learned by a network configuration.
 - e. Architecture: The specific arrangement of the layers and nodes in the network.

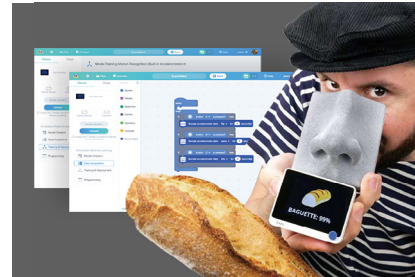
Lesson 05

Smell Recognition using Grove– Multichannel Gas Sensor



Project Overview

In this project, we are going to make an AI powered artificial nose that can sort alcohol from cola or identify whatever else you train it to smell. We use the Wio Terminal external Grove – Multichannel Gas Sensor v2 to collect different odor data and train the model to differentiate between cola, alcohol and air.



This is a project inspired by Benjamin Cabé's Artificial nose project.

"I spent quite some time trying to perfect my bread recipe, including trying to determine when my sourdough starter would be in the ideal condition to bake perfect baguettes." Cabé said. "Fast-forward to a few weeks later, I had assembled a full-blown (pun intended) artificial nose"

So after the project, you can try to build your own project detecting other odours.

Expected results

The desired result are shown in the Window below where the Wio Terminal displays the name of the currently detected alcohol or cola in real time.

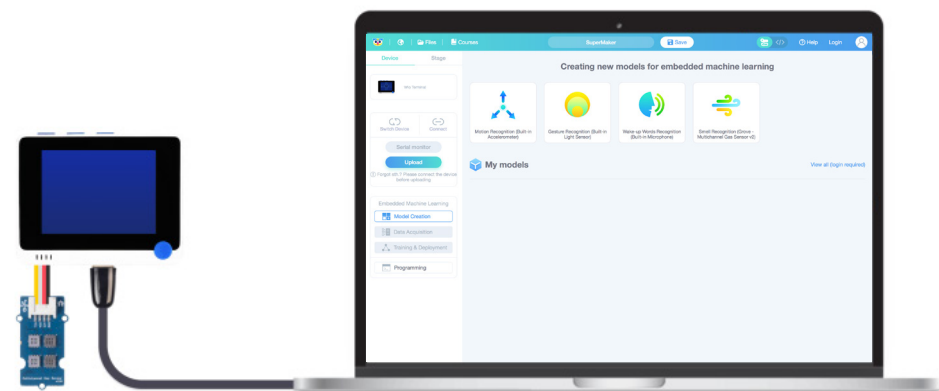


Material Preparation

Hardware requirements:

- Wio Terminal
- [Grove – Multichannel Gas Sensor v2](#)

Connection method:



· Note ·

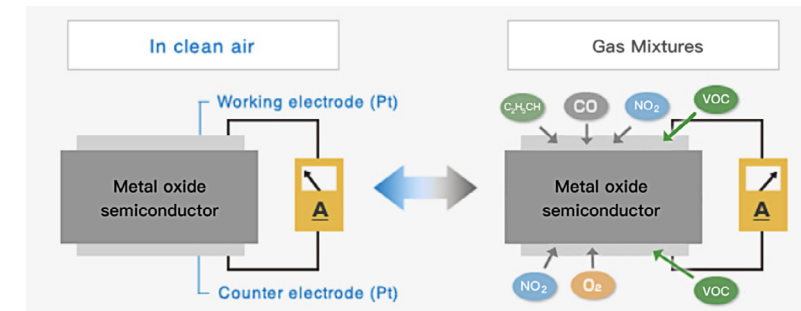
Before using it, the sensor needs to be preheated to achieve the internal chemical balance. The preheat voltage is consistent with its heating voltage V_H . The storage time and corresponding warm-up time are recommended as follows:

How Does Storage Time Affect The Recommended Warm-Up Time of This Sensor?

Storage time	Recommended warm-up time
Less than 1 month	No less than 24 hrs
1-6 months	No less than 48 hrs
Over 6 months	No less than 72 hrs

Theory

The gas sensor uses a MEMS process to fabricate a micro-thermal plate on a Si substrate using a metal oxide semiconductor material with low conductivity in clean air. When the sensor is exposed to a gas environment with odour of a specific gas, the conductivity of the sensor changes with the concentration of the gas being detected in the air. Higher the concentration of gas is, higher the conductivity of the sensor will be. The change in conductivity is converted to an output signal corresponding to the concentration of the gas using a simple circuit.



Grove – Multichannel Gas Sensor V2 has 4 measuring units, each of them is sensitive to various kinds of gases which means you are able to get four sets of data at the same time. Different sorts of gases can also be judged by these four sets of data.

It can detect a variety of gases besides Carbon monoxide (CO), Nitrogen dioxide (NO₂), Ethyl alcohol(C₂H₅CH), Volatile Organic Compounds (VOC) etc.



The content and combination of these four gases in a specific beverage or gas emitted by a substance is a set of values with certain characteristics. This set of characteristic values can be used to build a model to identify the gases emitted by different substances.

Practice

Project Steps

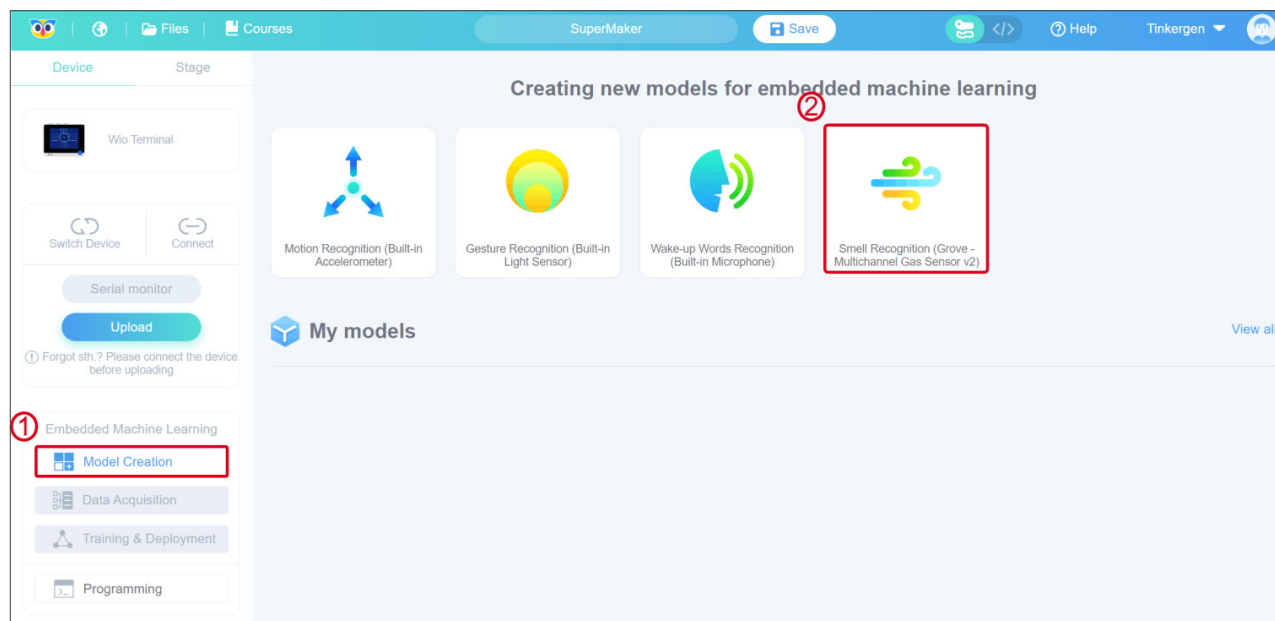
1. Creating and Selecting Models
2. Data Acquisition
3. Training and Deployment
4. Programming

Please Open <https://ide.tinkerjen.com/>

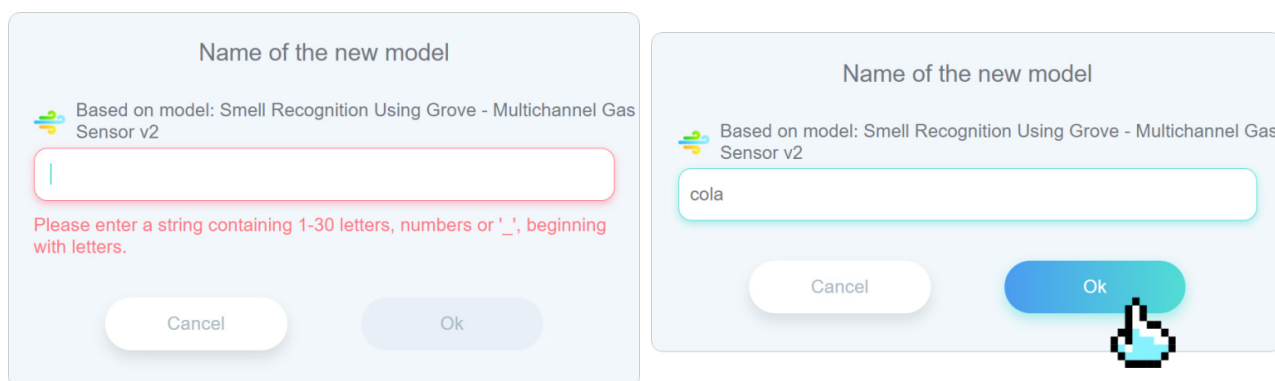
Step 1. Create and select models

1.1 Create a “Smell Recognition (Grove – Multichannel Gas Sensor v2)” model

Click on “Create and select model”, click on “Smell Recognition (Grove – Multichannel Gas Sensor v2)”, as shown in steps 1 and 2 below.



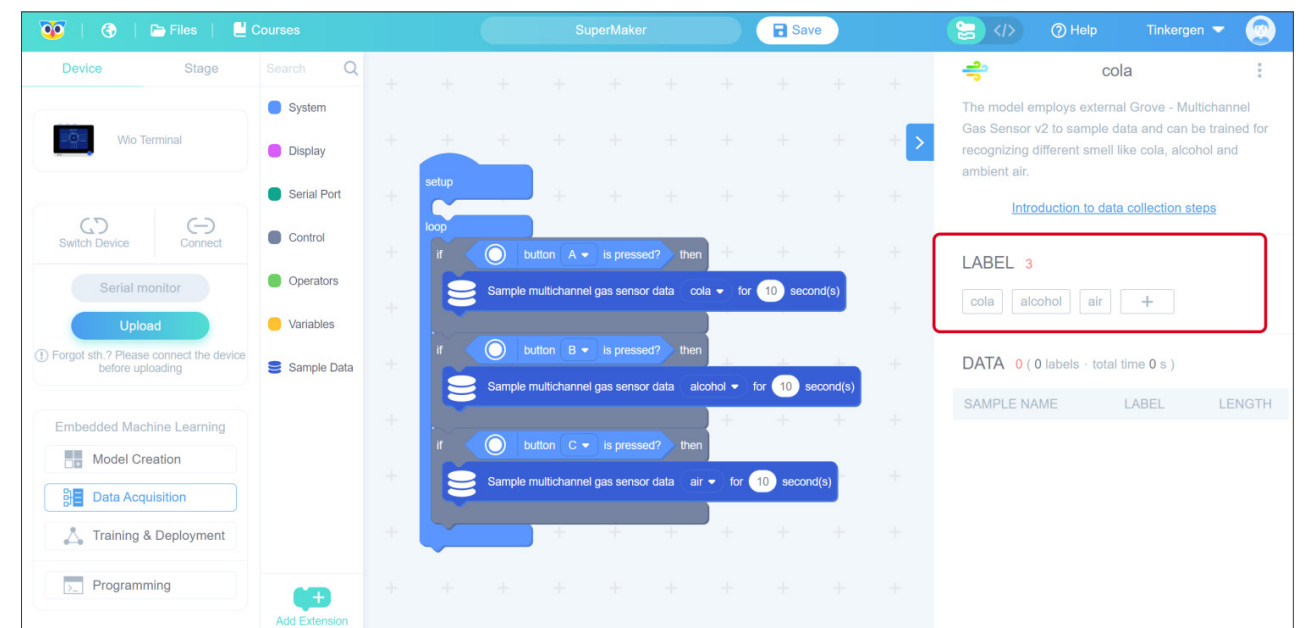
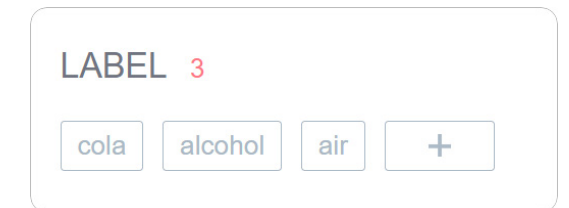
Enter a NAME according to the requirements.



Click Ok and it will automatically jump to the Data Acquisition interface.

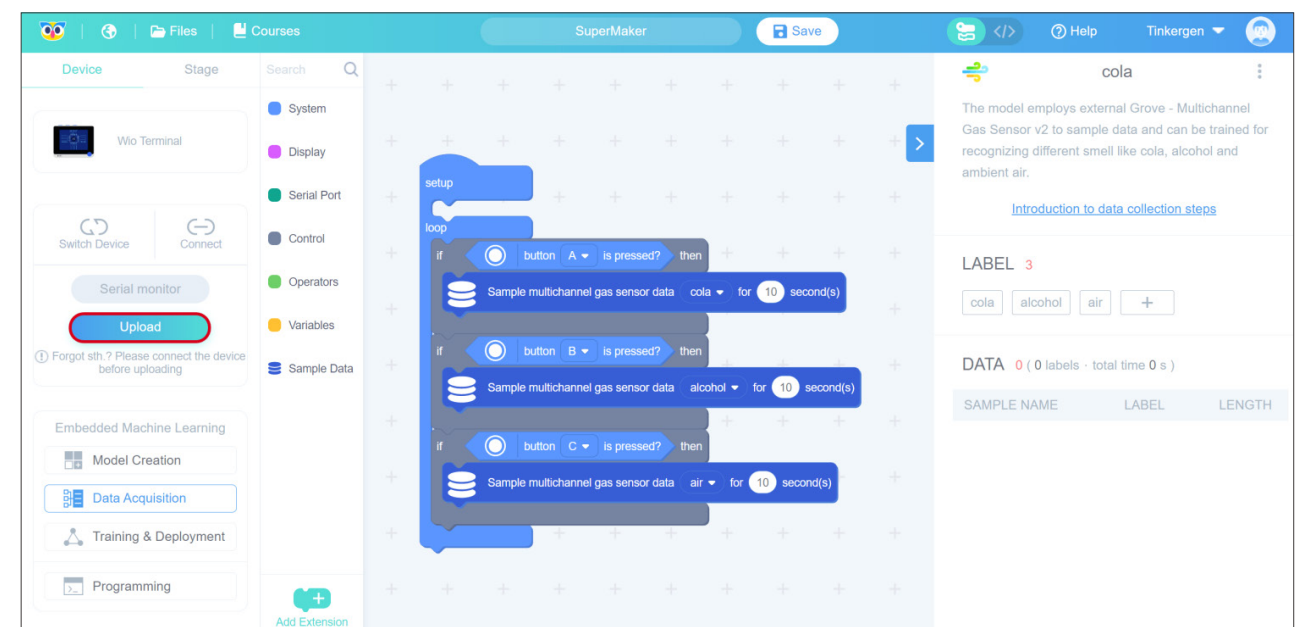
Step 2. Acquisition of data

2.1 Default label



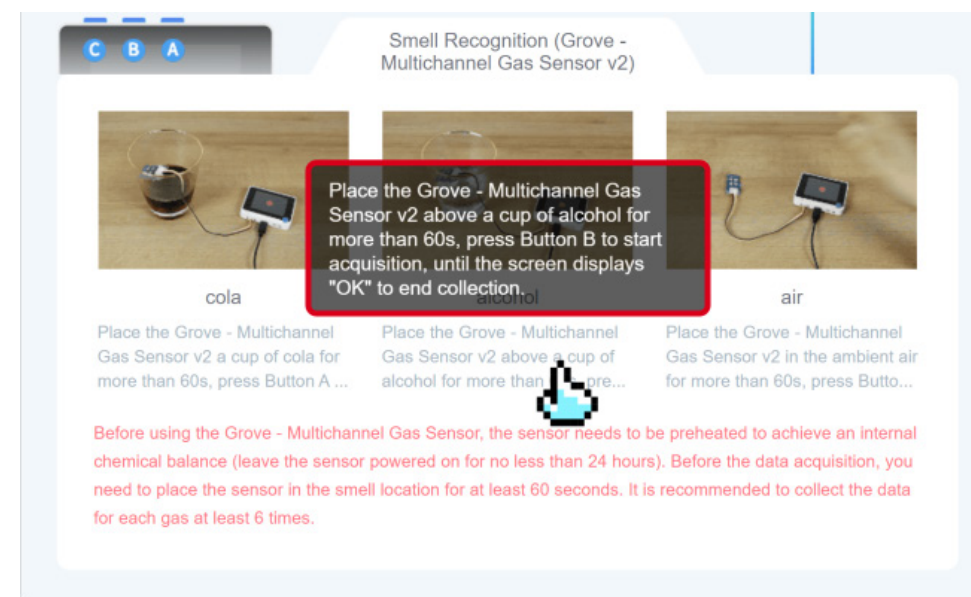
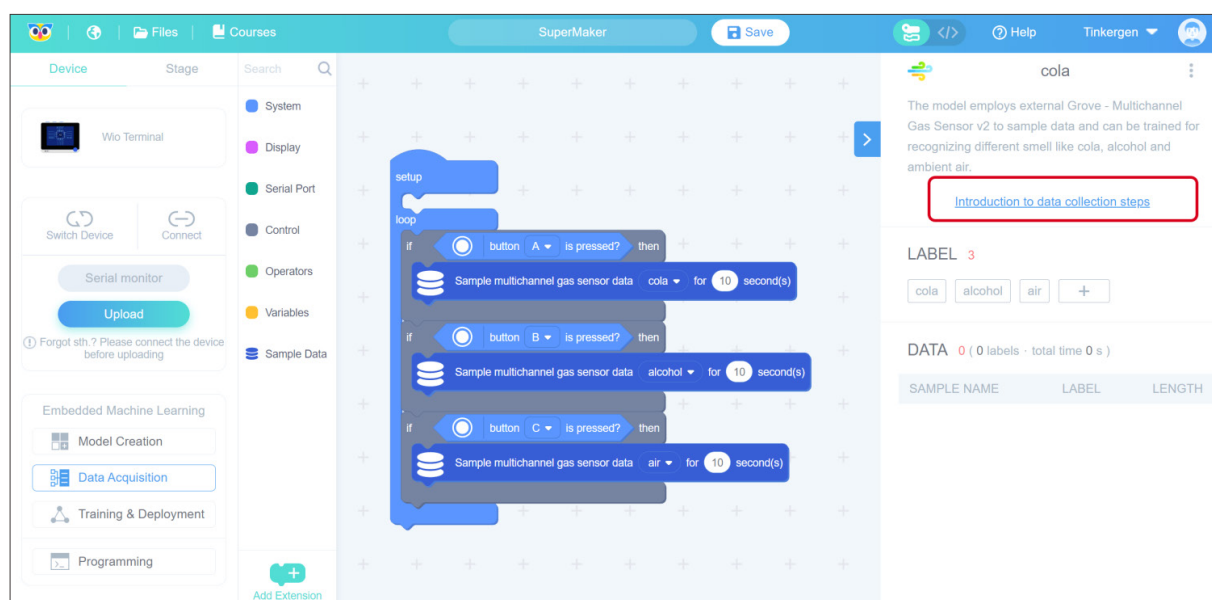
2.2 Connect the device and upload the default data acquisition program in Codecraft

When the Wio Terminal is connected, click **Upload** in the Codecraft window. This action will upload the default data acquisition programme.

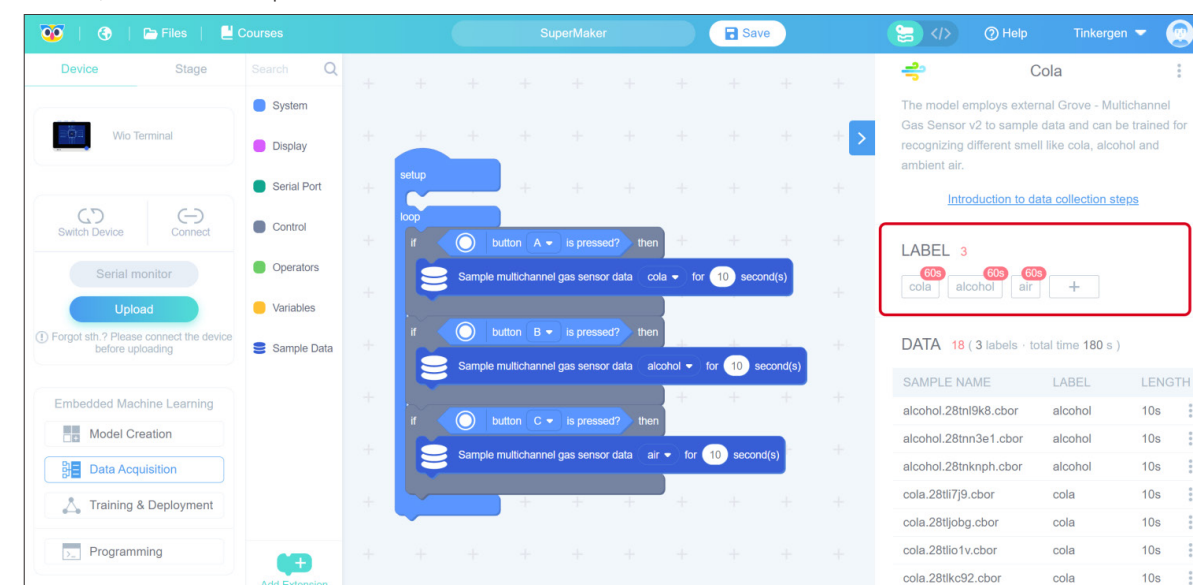


2.3 Acquisition of data

In the upper right hyperlink, you will find a step-by-step introduction to data acquisition. Follow the instructions to collect data.

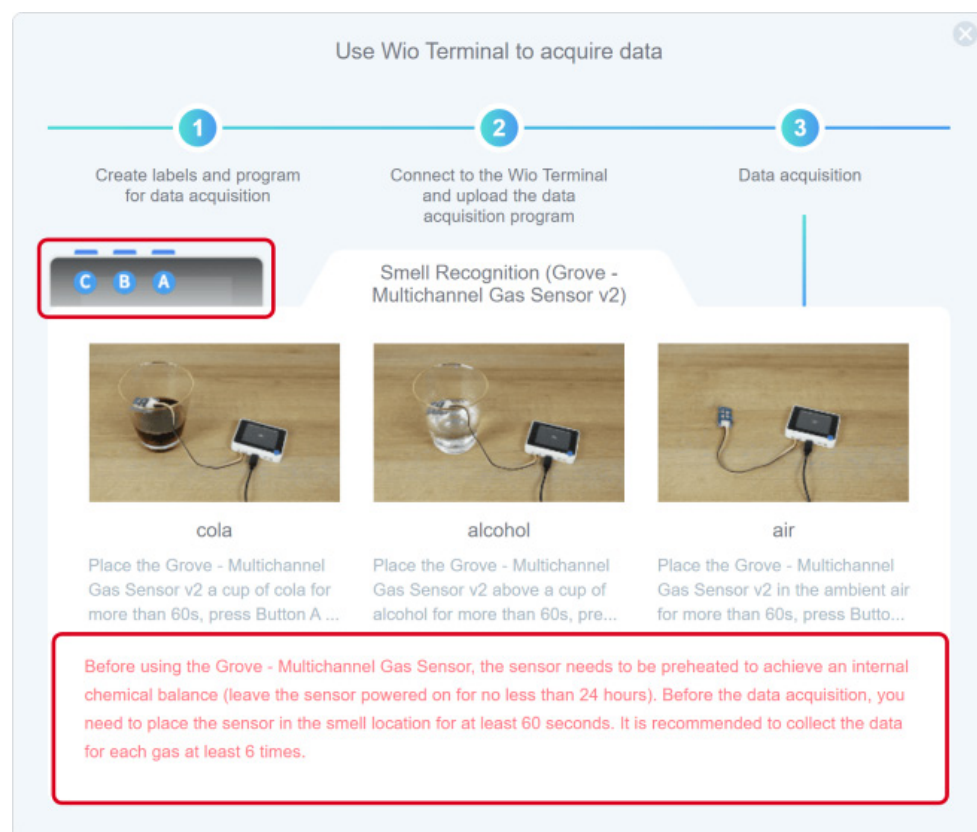


Now, the data acquisition is finished.

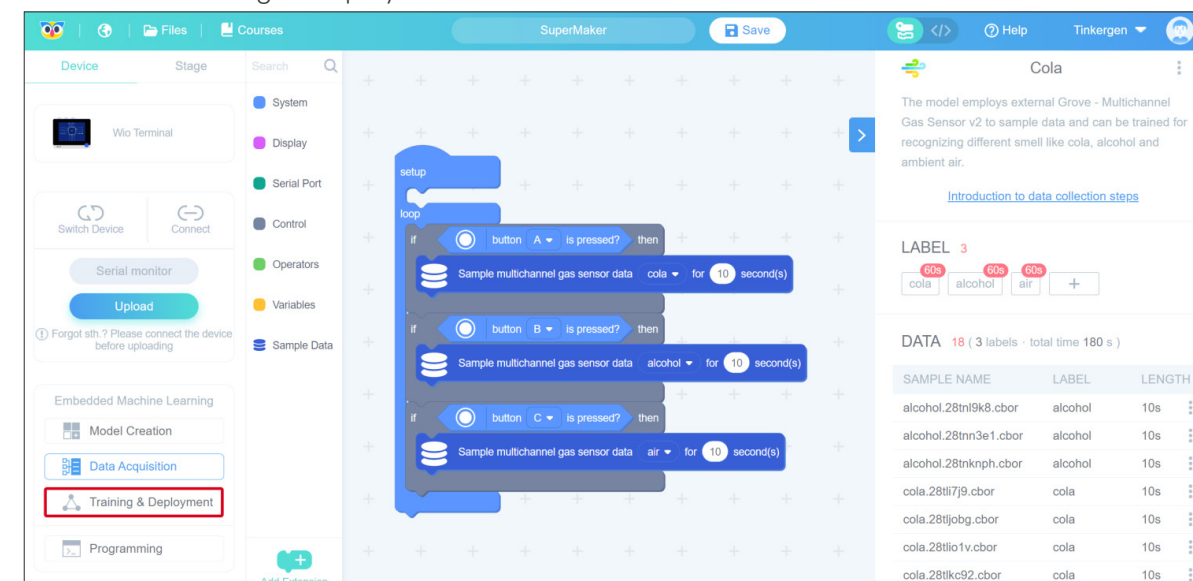


Attention:

- The sensor needs to be preheated before used to achieve the internal chemical balance.
- Wio Terminal button location.
- Animated gif has been accelerated, the actual action can slightly slow down.
- Please notice the red tips.
- Point the cursor over Description Texts for more detailed content.



Click on "Training & Deployment"



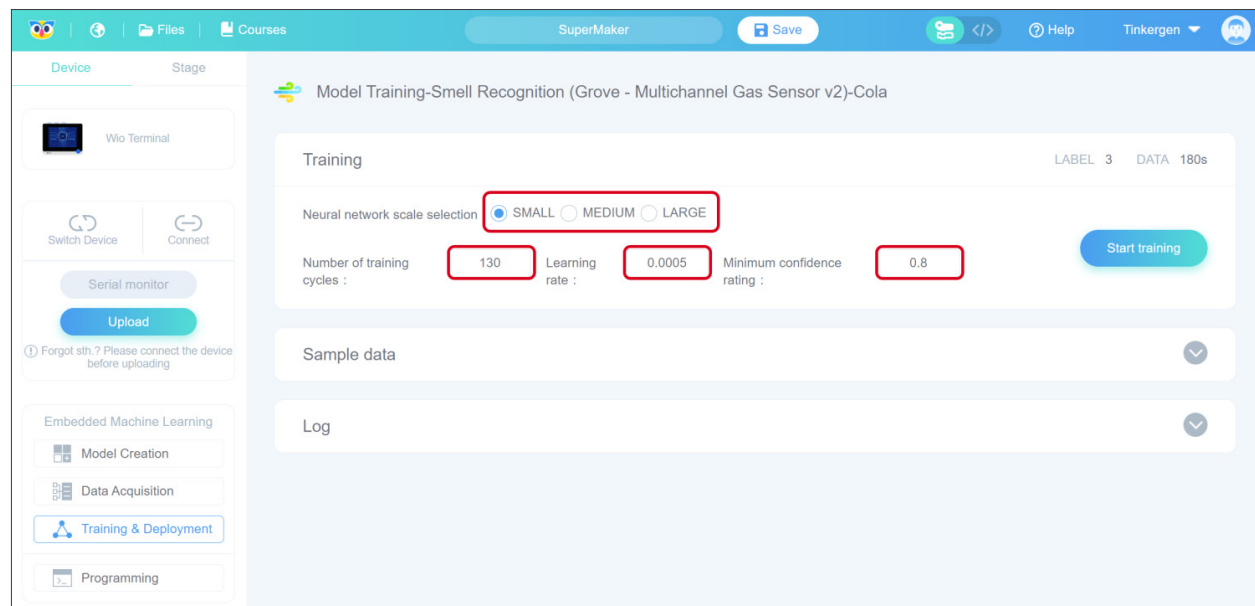
Step 3. Training and deployment

3.1 Set neural network and parameters

Select the suitable neural network size: one of small, medium and large

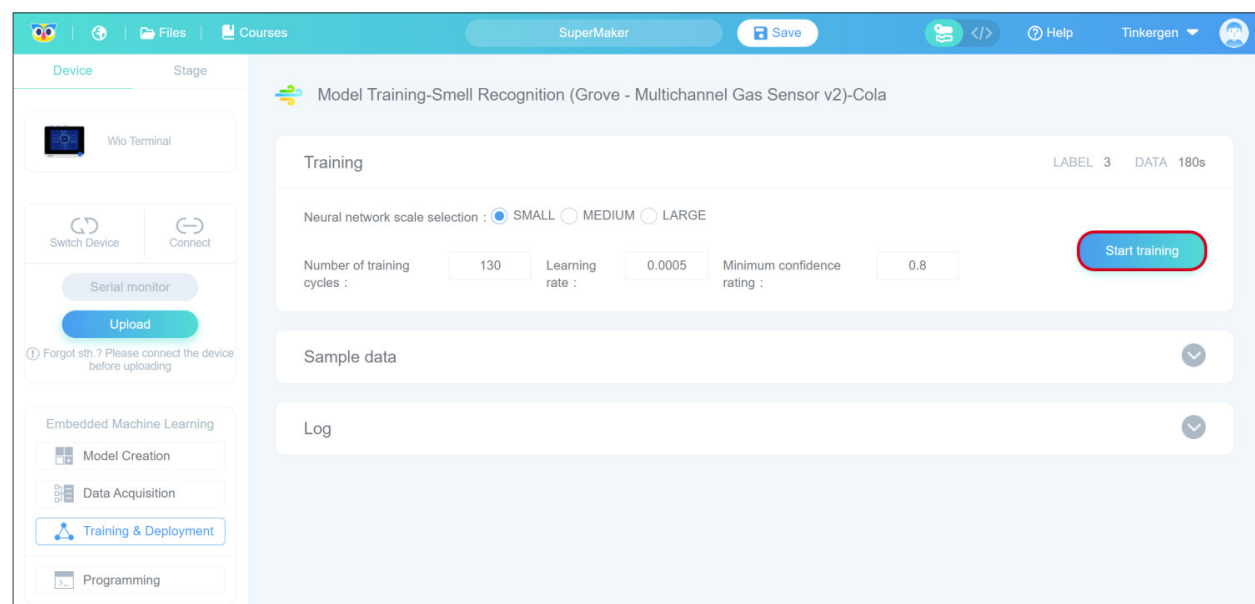
Set parameters, number of training cycles (positive integer), learning rate (number from 0 to 1) and minimum confidence rating(number from 0 to 1).

The interface provides default parameter values.



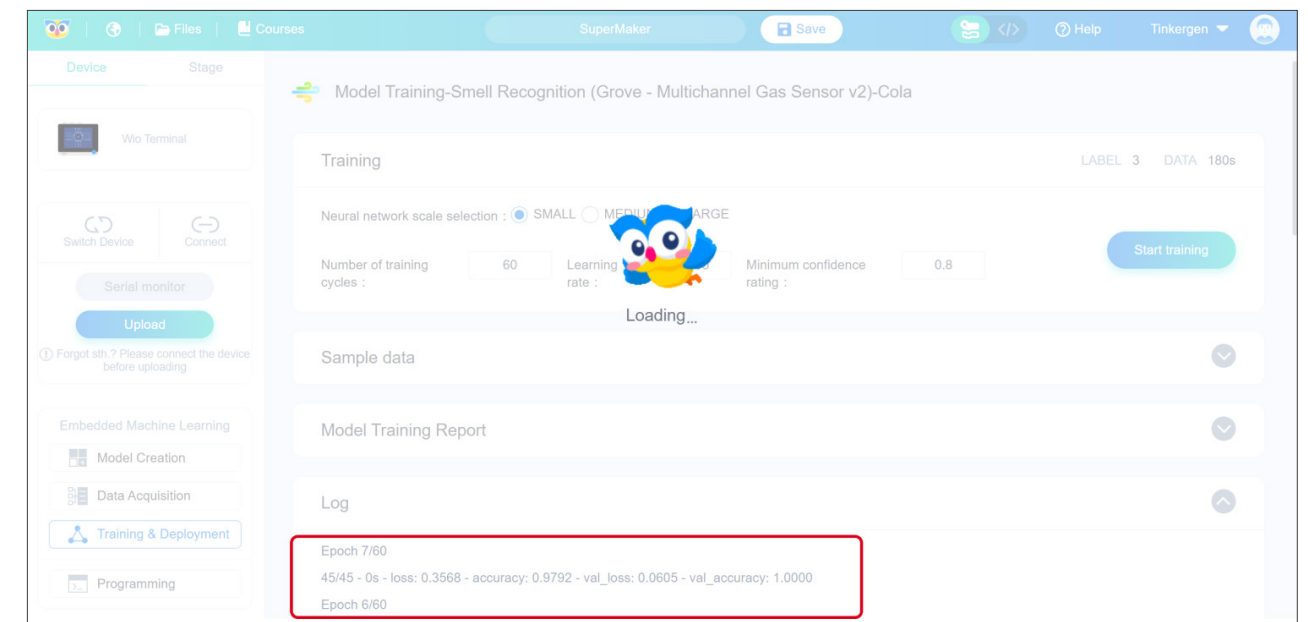
3.2 Start training the model

Click “Start training”



The duration of “Loading..” varies depending on the size of the selected neural network (small, medium and large) and the number of training cycles. Larger is the network size and number of training cycles, the longer it takes.

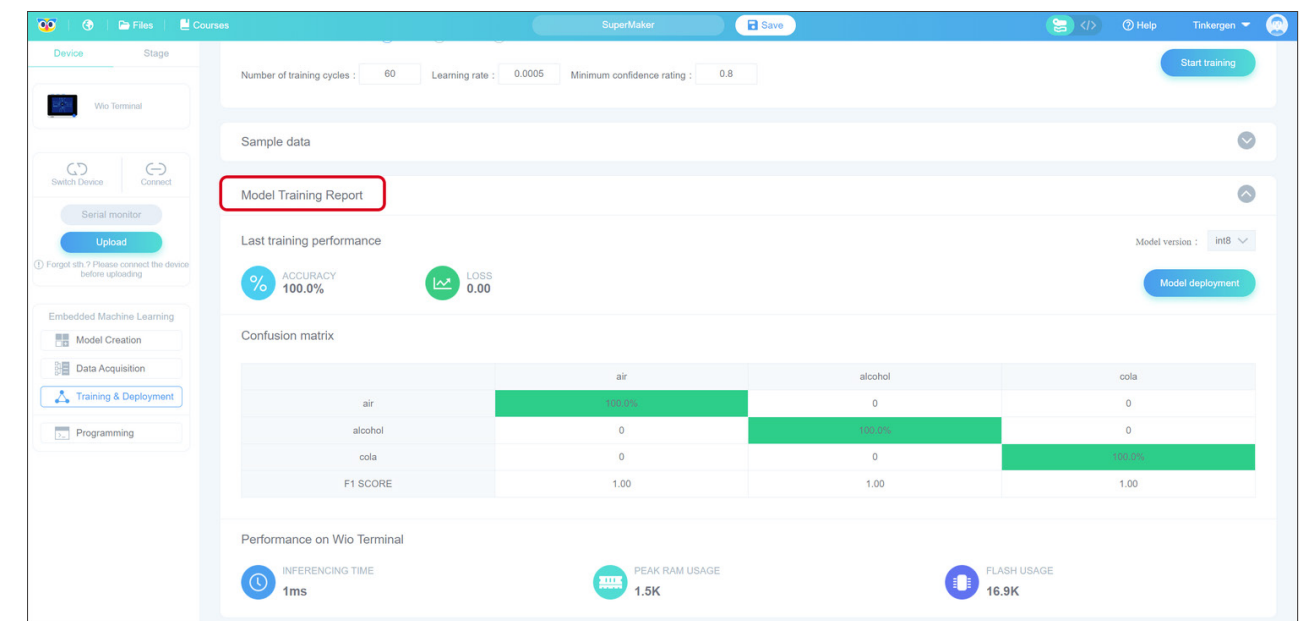
You can also infer the waiting time by observing the “Log”. In the figure below, “Epoch: 7/60” indicates the total number of training rounds is 60 while 7 rounds have been trained.



3.3 Observe the model performance to select the ideal model

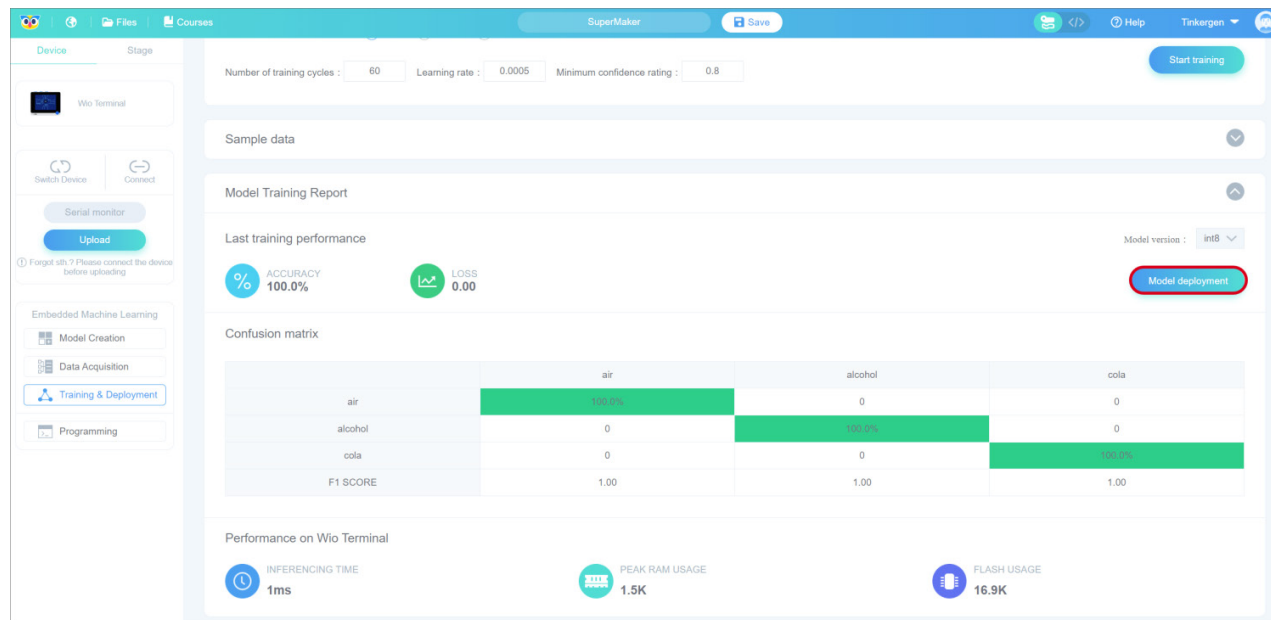
In the “Model Training Report” window, you can observe training results including the accuracy, loss and performance of the model.

If the training results are not satisfactory, you can go back to the first step of training the model, select another size of the neural network or adjust the parameters and train it until you get a model with satisfactory results.

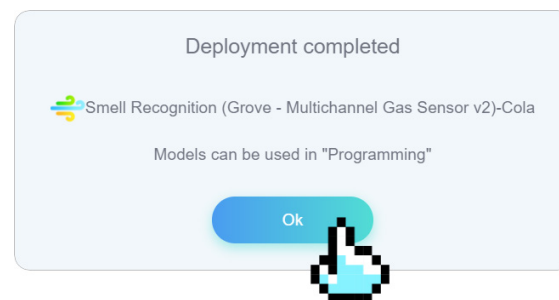


3.4 Deploy the ideal model

In the “Model Training Report” window, click [Model deployment](#)



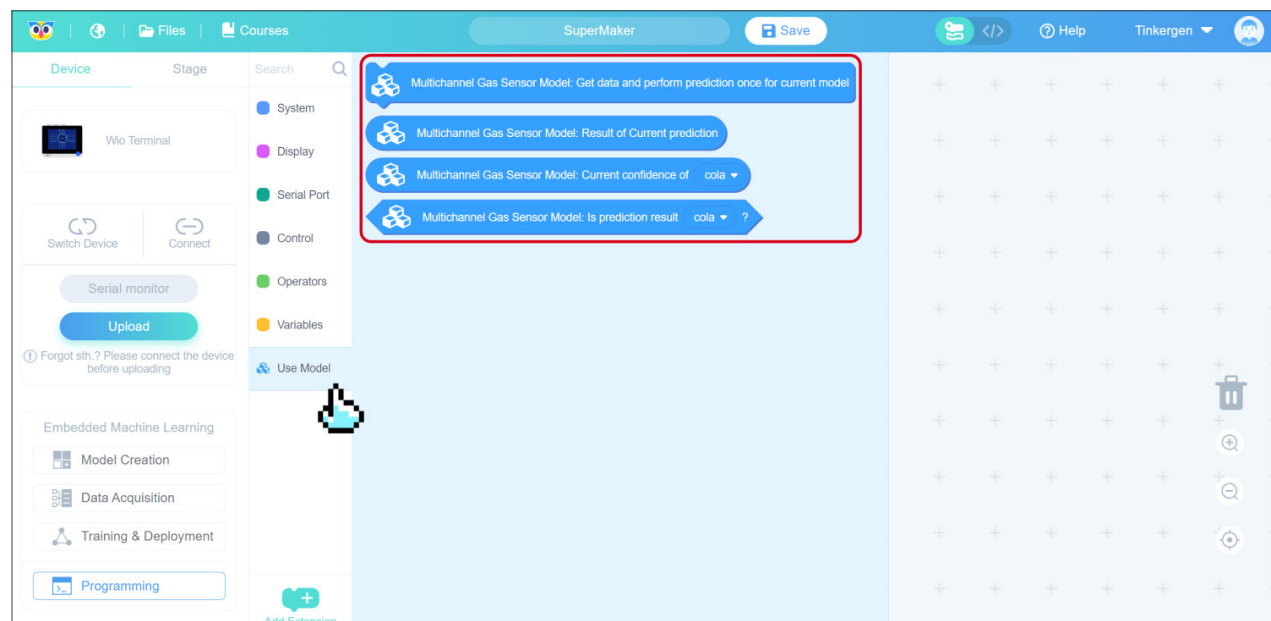
Once the deployment is complete, click “Ok” to jump to the “Programming” window



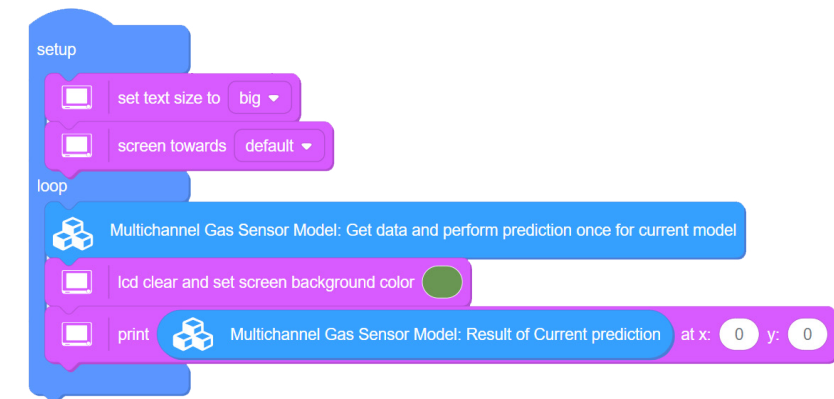
Step 4. Use and programming

4.1 Write the program for using the model

In the “Programming” interface, click on “Use Model” to use the deployed model.

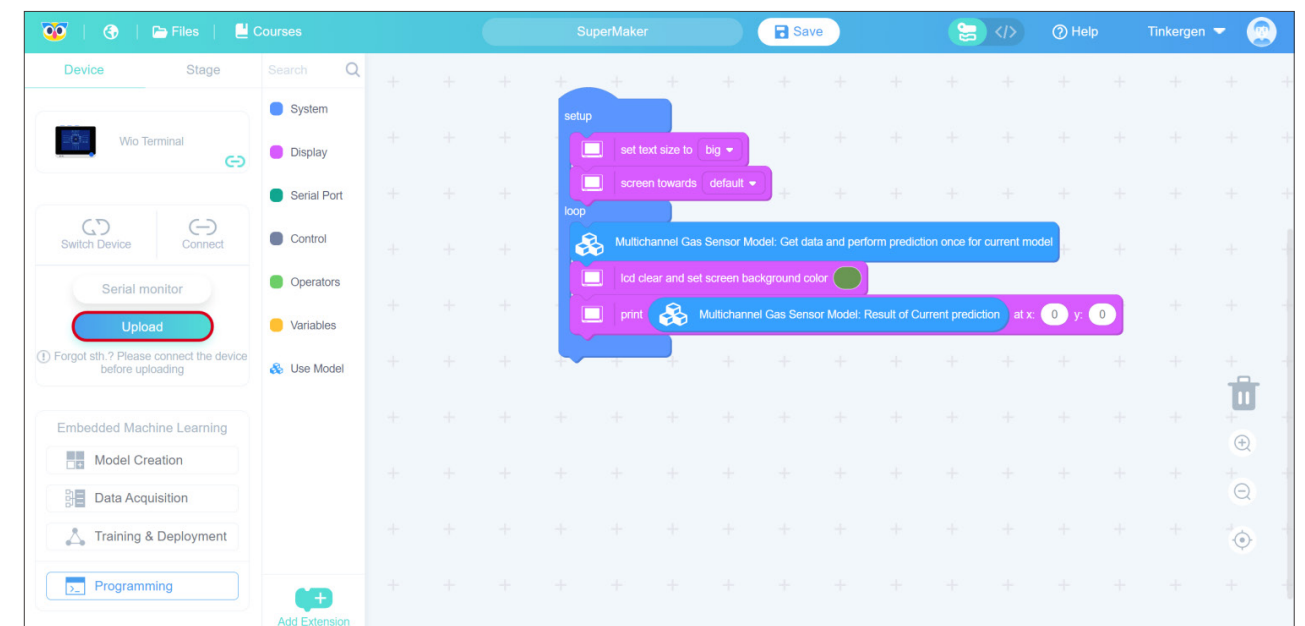


Try to use your model by writing the following program.



4.2 Upload the program to Wio Terminal

Click the “Upload” button.



4.3 Wio Terminal test model

Put your Grove – Multichannel Gas Sensor v2 near cola to see if Wio Terminal’s screen can show “cola”. Try other beverages, and see if the Wio Terminal can recognize your beverages.

Congratulations! You have completed your fourth TinyML model. I believe you have tried to set the parameters, the number of training cycles (positive integer), learning rate (0~1 number) and the minimum confidence rating (0~1 number) but how these parameters affect our model’s performance?

Let’s start learning these special parameters.

ML Theory(hyperparameters):

hyperparameters

What are Hyperparameters?

Hyperparameters are:

- the variables that determine the network structures
 - a. Number of Hidden Units
 - b. Number of Hidden Layers
- the variables which determine how the network is trained
 - a. Number of Training Cycles/Epoch
 - b. Learning Rate

Hyperparameters are set before training (optimizing the weights and bias).

In the last lesson, we have learned about the variables that determine the network structure.

In this lesson we will discuss the variables which determine how the network is trained.

Training LABEL 3 DATA 235s

Neural network scale selection : SMALL MEDIUM LARGE

Number of training cycles : 50 Learning rate : 0.0045 Minimum confidence rating : 0.6 Start training

1. Epoch/Number of Training Cycles

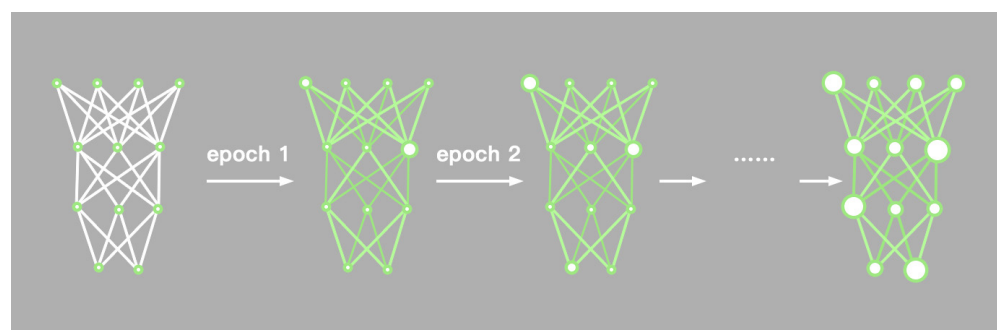
Number of Training Cycles are is called epoch. As a rule of thumb, you take your training data in small batches and feed it to your algorithm. A time period marked by feeding the whole of your training data set to the model is known as an epoch.

1.1 Set 'Number of training cycles' to 1. This will limit training to a single iteration and then click Start training.

1.2 Now change 'Number of training cycles to 2 and you'll see performance goes up.

Finally, change the 'Number of training cycles' to 100 or more and let the training to finish.

This epochs number is an important hyperparameter for the algorithm. It specifies the number of times that the entire training dataset passes through the training process of the algorithm. With each epoch, the internal model parameters will update.



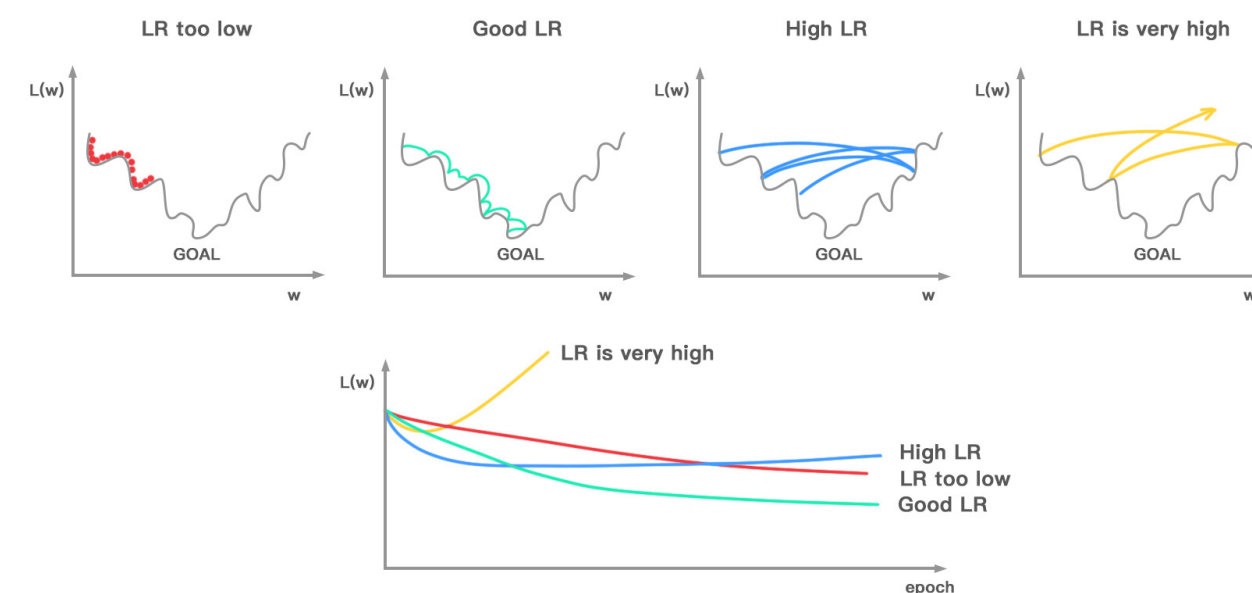
2. Learning Rate

The learning rate is a hyperparameter that controls the changes to the model in response to the estimated error each time the model are updated (as the picture above shows). Choosing the learning rate is challenging because if the value is too small, it may result in a long training process whereas a too large value may result in learning a sub-optimal set of weights too fast or an unstable training process.

The learning rate is the most important hyperparameter when configuring your neural network. Therefore, it is crucial to know the impact of different learning rates on model performance.

Just take our perfect model as the bottom of the valley, the learning rate is each pace we take to go to our target.

1. Learning rate is too low: We don't know when we can reach our target and we may reach a "fake" valley.
2. Good learning rate: It takes the proper time to reach the desired goal.
3. High learning rate: We always miss our way to get into the valley's entry.
4. Learning rate is Very High : Poor behavior.



★ Summarization

1. Theory: Gas sensor
2. TinyML practice

3. ML theory (Hyperparameter)

- The variables that determine the network structure
 - a. Number of Hidden Units
 - b. Number of Hidden Layers
- The variables which determine how the network is trained
 - a. Number of Training Cycles/Epoch: A time period marked by feeding the whole of your training data set to the model is known as an epoch.
 - b. Learning Rate: Controls the changes to the model in response LR to the estimated error each time the model weights are updated.

Embedded Machine Learning

Model Creation

Data Acquisition

Training & Deployment

Programming

More lessons coming soon

Advanced Projects			
Lesson 6	Sport recognition by using built-in accelerometer	<ul style="list-style-type: none"> • Theory: Introduction of sport recognition • Practice: <ul style="list-style-type: none"> ◦ Model creation ◦ Data Acquisition ◦ Training & Deployment ◦ Programming • ML Theory (Model assessments) 	Wio Terminal
Lesson 7	Barcode recognition by using built-in light sensor	<ul style="list-style-type: none"> • Theory: Introduction of the barcode • Practice: <ul style="list-style-type: none"> ◦ Model creation ◦ Data Acquisition ◦ Training & Deployment ◦ Programming • ML Theory (Improve model) 	Wio Terminal
Lesson 8	Face Recognition with thermal imaging sensor	<ul style="list-style-type: none"> • Theory: Introduction of thermal imaging sensor, Face Recognition • Practice: <ul style="list-style-type: none"> ◦ Model creation ◦ Data Acquisition ◦ Training & Deployment ◦ Programming • ML Theory (Details of layers) 	Wio Terminal Grove - Thermal Imaging Camera (MLX90640)
Summarization			
Lesson 9	Creative projects and Summarization	<ul style="list-style-type: none"> • Summarization of ML theory • Examples of creative projects 	

Afterword

Wio Terminal for Seed Wiki

<https://wiki.seeedstudio.com/Wio-Terminal-Getting-Started/>

Codecraft

<https://ide.tinkergen.com/>

Purchase link

Wio Terminal:

<https://www.seeedstudio.com/Wio-Terminal-p-4509.html>

Grove – Multichannel Gas Sensor v2:

<https://www.seeedstudio.com/Grove-Multichannel-Gas-Sensor-v2-p-4569.html>

Grove – Thermal Imaging Camera:

<https://www.seeedstudio.com/Grove-Thermal-Imaging-Camera-IR-Array-MLX90640-110-degree-p-4334.html>

Call for multilingual translation volunteers!

If you wish to translate course content into a language version other than English or Chinese, you can also contact us for support.

CONTACTS

Tel: +86-0755-86716703

Address: 1002, G3 Building, TCL International E City, 1001 Zhongshan Park Road, Nanshan District, Shenzhen

General: contact@chaihao.org

www.tinkergen.com

www.seeedstudio.com

Course Developers

This course was co-authored by the following SeeedStudio employees:

Authors: Huiyin Lai, Hao Yuan

Designer: Yihui Meng

Proofreaders: Lei Feng, Jianfeng Yu, M Aitesam

