

Project hardware used in this guide

A live video application that detects objects on the Linux processor, transfers the highest confidence value through the Bridge, and shows the result on the onboard LED matrix.

How to use this guide

This guide is designed for engineers, makers and application developers who want a complete, repeatable Edge AI vision workflow on Arduino UNO Q. Follow Parts 1 to 4 for the supplied example. Continue to Part 5 only if you want to train and deploy a custom object-detection model.

BEFORE YOU BEGIN

Complete the one-time board setup before attaching the powered hub and peripherals. Keep the computer and UNO Q on the same Wi-Fi network when using Network Mode.

Document map

Section	Purpose	What it contains
Part 1	Plan	Architecture, expected result, requirements and readiness checks
Part 2	Prepare	Board configuration, powered hub and camera connection
Part 3	Build	Import, configure and run the supplied application
Part 4	Validate	Confirm browser output, LED matrix behaviour and model selection
Part 5	Extend	Create a custom dataset, train in Edge Impulse and deploy
Reference	Resolve	Troubleshooting, validation checklist, glossary and next steps

Contents

1. Project overview
2. System architecture and data flow
3. Requirements and pre-flight checks
4. Initial UNO Q setup
5. Hardware and camera setup
6. Import and run the application
7. Validate the result
8. Select a different AI model
9. Train and deploy a custom model
10. Troubleshooting
11. Validation checklist
12. Extensions and glossary

1. Project overview

1.1 Objective

Build a complete Edge AI vision application on Arduino UNO Q. A USB camera captures live video, the Linux processor performs local object detection, and the Bridge transfers the highest confidence score to the microcontroller for display on the onboard LED matrix. The result combines real-time vision inference with deterministic hardware control on one platform.

1.2 Why object detection at the edge?

Video object detection enables embedded systems to identify and classify objects in a live scene. Running inference locally can reduce cloud dependency, improve privacy and shorten response time. Typical application areas include industrial automation, robotics, smart retail, surveillance, quality inspection, autonomous systems and IoT vision.

1.3 Expected result

- Live camera feed displayed in a web browser.
- Bounding boxes and confidence values over detected objects.
- Highest confidence value sent from Linux to the MCU through the Bridge.
- Confidence percentage shown on the onboard LED matrix.
- Display cleared after 1.5 seconds without a detection.

SUCCESS CRITERION

The system is working when the browser shows an annotated live stream and the LED matrix updates with the highest current confidence value.

1.4 Why Arduino UNO Q?

Arduino UNO Q combines the application-processing capability of the Qualcomm Dragonwing™ QRB2210 MPU with the deterministic real-time control of an STM32U585 MCU. The QRB2210 runs Debian Linux for AI, computer vision, connectivity, and application processing, while the STM32U585 runs Arduino sketches on Zephyr RTOS for precise, low-latency hardware control. This heterogeneous architecture allows AI inference and real-time control to operate together on one platform, making UNO Q suitable for intelligent vision, robotics, industrial automation, and IoT applications.

Through Arduino App Lab, developers can quickly deploy pre-trained or custom AI models, connect USB cameras, and integrate hardware responses. The Arduino UNO Q offers several unique technical advantages:

Capability	Project value
Hybrid Linux + MCU Architecture	Combine Linux-based AI processing with dedicated MCU control for simultaneous vision processing and real-time hardware control.
Edge AI Capability	Run AI inference locally enables low-latency object detection while enhancing data privacy.
Bridge communication	Enable seamless data exchange between Linux and the MCU, with detection results visualized on the onboard LED matrix.

Capability	Project value
Rapid AI development	Use Arduino App Lab's low-code workflow to quickly develop and deploy AI-powered applications.
Model choice	Switch between pre-trained vision models or deploy custom models trained with Edge Impulse.
Custom AI Model Deployment	Train and deploy domain-specific vision models with Edge Impulse to accelerate prototyping and proof-of-concept development.
Hardware integration	Combine AI with sensors, displays, motors, communications, and industrial controls for production-ready embedded applications.

2. System architecture and data flow

The project contains four connected layers. Understanding the boundaries between them makes configuration and troubleshooting easier.

Stage	Component	Function
1	Camera input	USB webcam supplies the live video stream.
2	Linux application	The QRB2210 runs the Video Object Detection Brick and identifies objects.
3	Bridge transfer	The application sends the highest confidence score to the MCU domain.
4	Real-time output	The STM32 MCU updates the onboard LED matrix and clears stale values.

DATA FLOW

Camera → Linux inference → highest confidence value → Bridge → MCU → LED matrix. The full annotated video remains in the browser interface.

2.1 Application components

- USB camera for image capture.
- Video Object Detection Brick for inference.
- Browser-based interface for the annotated stream and confidence threshold.
- Arduino sketch on the MCU for LED matrix output.
- Optional Edge Impulse connection for custom-model training and deployment.

2.2 Source project

The supplied example is based on the [karlsoderby/cv-workshop-example repository](#). This guide reorganises the workflow, adds readiness checks, validation criteria, troubleshooting and extension guidance.

3. Requirements and pre-flight checks

3.1 Hardware

Item	MPN
Arduino UNO Q	ABX00173
USB Type-A to USB-C cable	MC002480
8-in-1 USB-C hub	TPX00241
USB-C cable for hub power	TPX00243
USB hub power supply	TPX00242
USB webcam	Any compatible model
Computer	Arduino App Lab installed
Keyboard and mouse combo	USB or wireless
HDMI display and cable	As required



Figure 1. Hardware components used for the project

3.2 Software and accounts

- Arduino App Lab installed on the computer.
- Arduino account for App Lab and the Edge Impulse integration.
- Edge Impulse Studio account only if training a custom model.
- Downloaded object-detection-example.zip application package.
- Wi-Fi network available to both computer and UNO Q.

Download: [Arduino App Lab](#) | [Edge Impulse Studio](#)

3.3 Pre-flight checklist

- ☐ Board has completed its first-time configuration.
- ☐ Linux password and Wi-Fi credentials are available.
- ☐ Computer and UNO Q can use the same Wi-Fi network.
- ☐ Powered hub, camera and cables are ready.
- ☐ Example ZIP is downloaded and remains compressed for import.

POWER NOTE

Use the recommended powered hub when several USB peripherals are connected. Connect hub power before starting the complete setup.

4. Initial UNO Q setup

Complete this section once for a new board or after a reset that requires reconfiguration.

1. Connect UNO Q directly to the computer using the USB-C cable.
2. Open Arduino App Lab and select the detected UNO Q.
3. Set the Linux operating-system password.
4. Configure the Wi-Fi credentials.
5. Apply a firmware update if App Lab prompts you.
6. Confirm that the board details appear in App Lab.

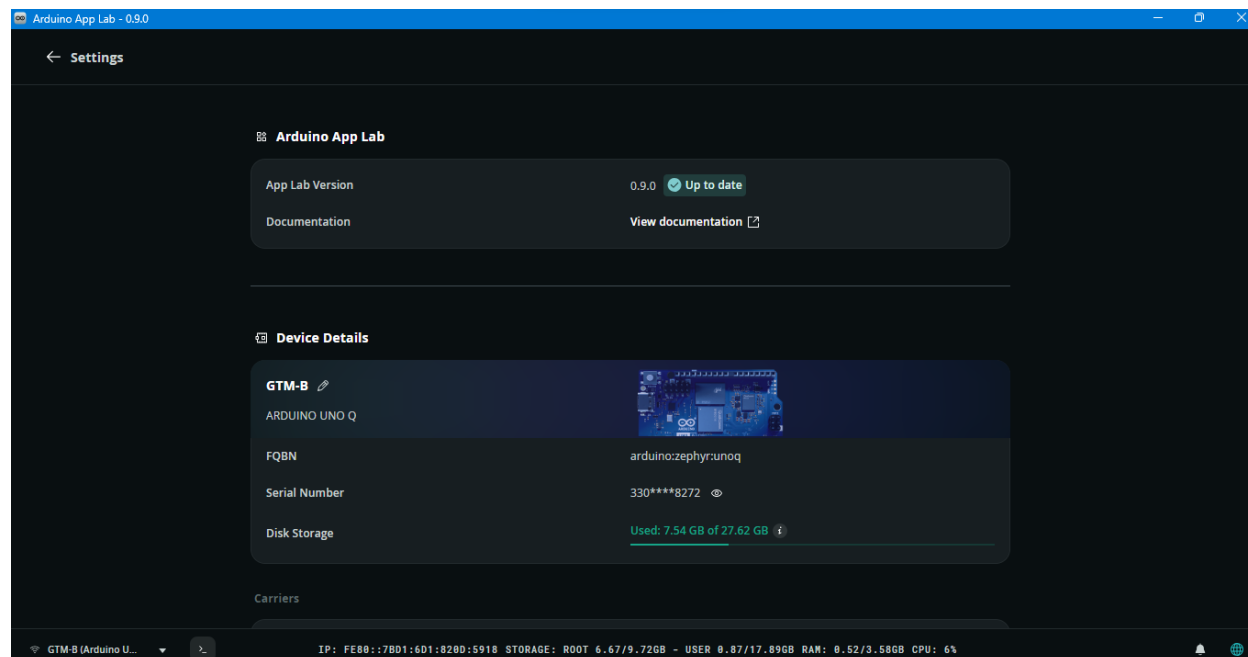


Figure 2. UNO Q device details displayed in Arduino App Lab

ONE-TIME STEP

After the initial configuration succeeds, disconnect the direct USB connection and continue in Network Mode for the camera setup.

5. Hardware and camera setup

5.1 Connect the powered hub

1. Disconnect UNO Q from the computer.
2. Connect the hub power supply to the USB-C hub.
3. Attach the powered USB-C hub to UNO Q.
4. Connect the USB webcam to the hub.
5. Connect the keyboard, mouse and HDMI output if they are required for your setup.
6. Power and start the system.

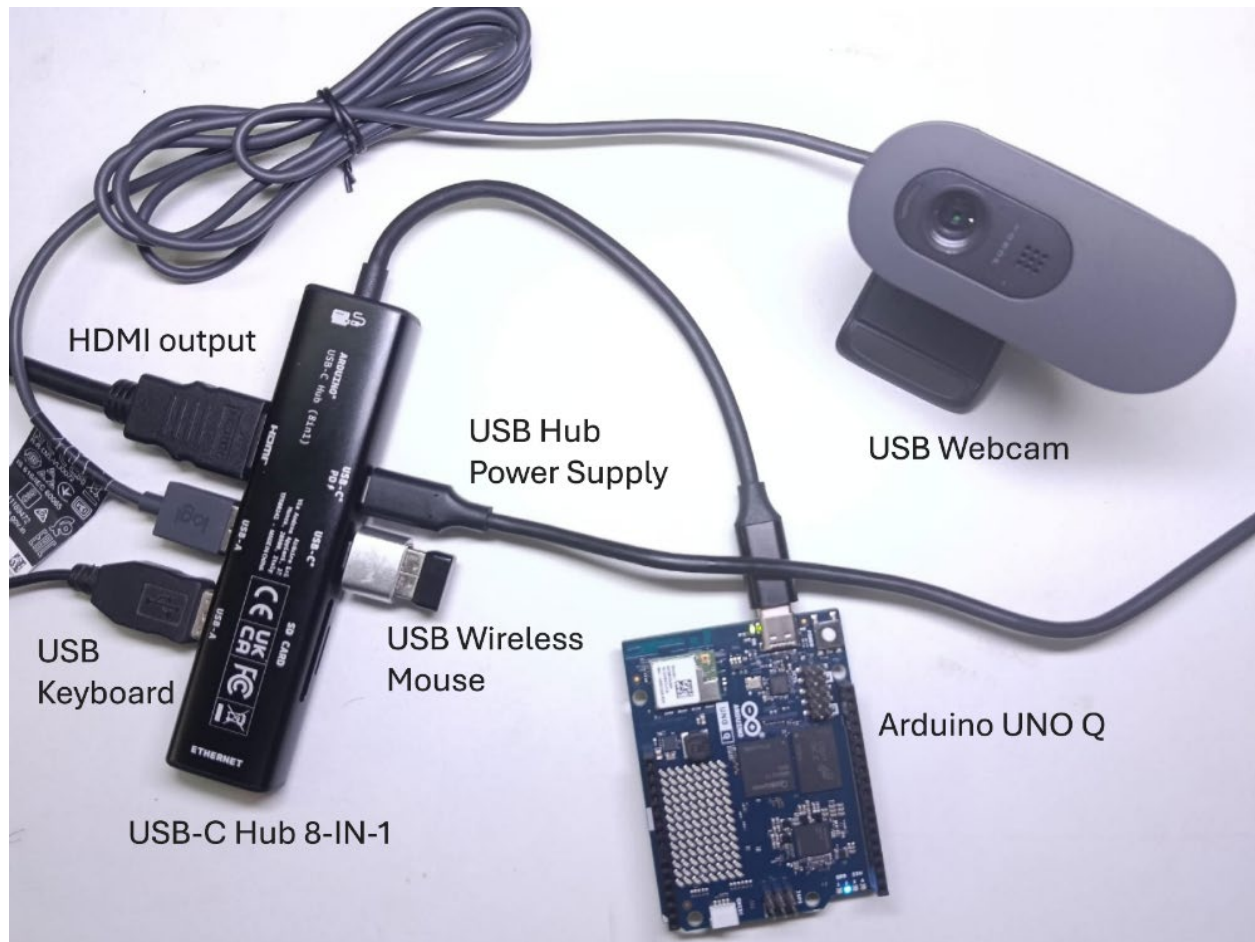


Figure 3. Complete peripheral connection layout



Figure 4. Powered hub connected to UNO Q and USB camera

CONNECTION CHECK

The camera must be connected to the hub that is attached to UNO Q. A camera connected only to the computer is not available to the board application.

5.2 Switch to Network Mode

- Connect the computer running App Lab and UNO Q to the same Wi-Fi network.
- Open Arduino App Lab.
- Wait for the board to appear automatically.
- Select UNO Q and confirm it is connected before importing or running the app.

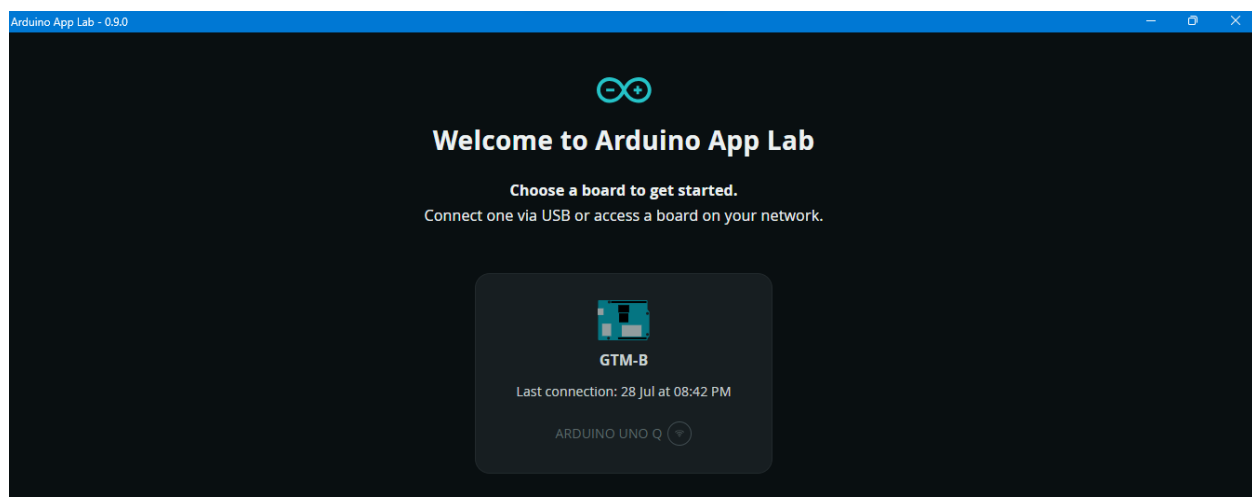


Figure 5. UNO Q available from the Arduino App Lab welcome screen

6. Import and run the application

6.1 Download the example

Download the [object-detection-example.zip package](#) and keep it as a ZIP file for import.

6.2 Import into App Lab

1. Open My Apps.
2. Select Create New App.
3. Choose Import App.
4. Select object-detection-example.zip.
5. Wait for import to finish.
6. Confirm that the application appears under My Apps.

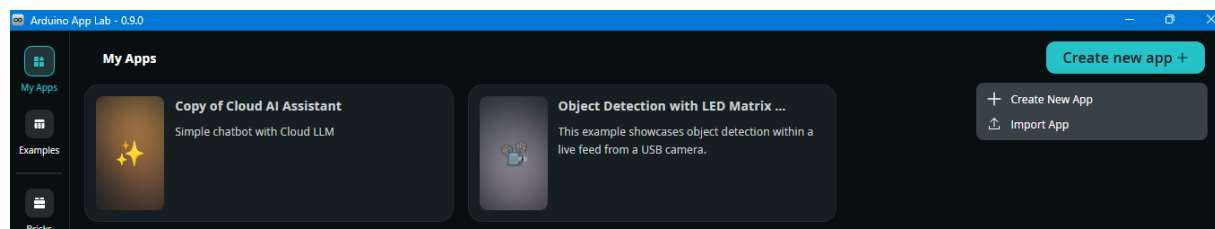


Figure 6. Create or import an application from My Apps

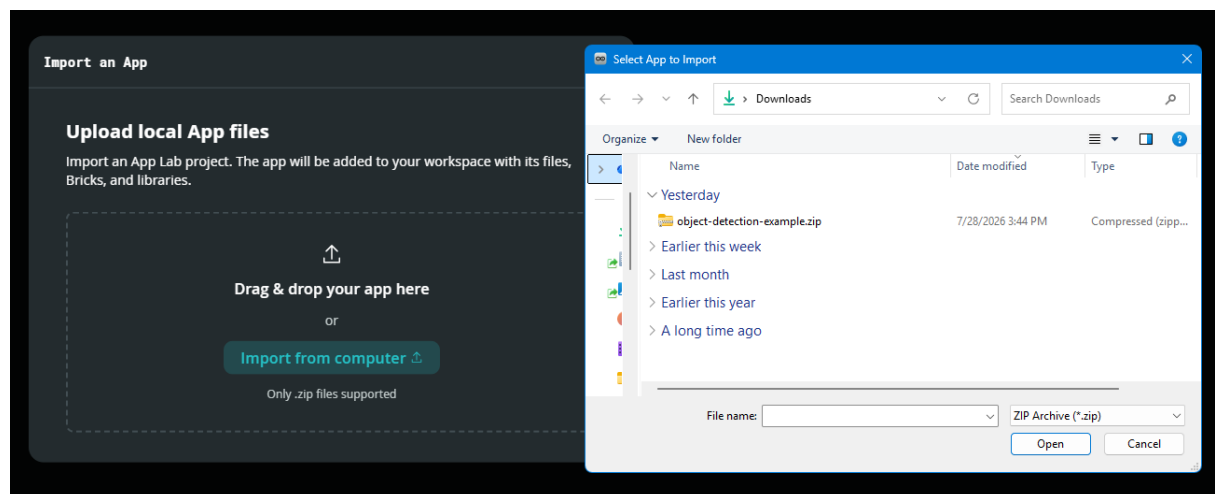


Figure 7. Importing the downloaded ZIP package

IF IMPORT FAILS

Confirm that you selected the ZIP package itself and that the download completed. Re-download the repository package if the archive is incomplete.

6.3 Run the application

1. Open the imported project.
2. Select Run in Arduino App Lab.
3. Wait for the browser interface to open.
4. If the page does not open automatically, use the board IP address with port 7000 in the format `http://<UNO-Q-IP-ADDRESS>:7000`.
5. Allow the camera stream to initialise.

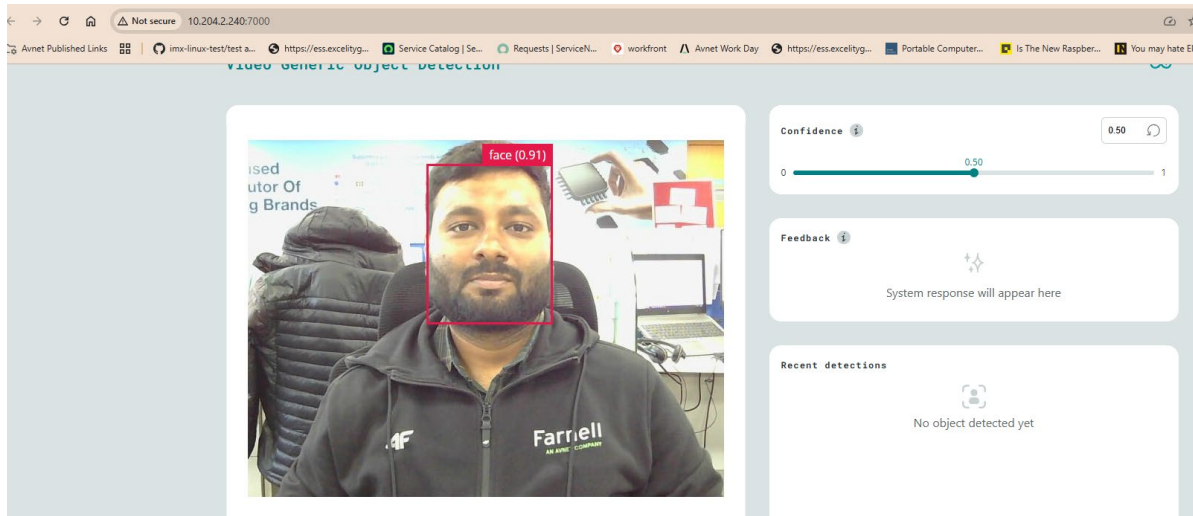


Figure 8. Browser interface showing live face detection

NETWORK DETAIL

The browser address must use the IP address assigned to your UNO Q. The placeholder text is not a usable address.

7. Validate the result

7.1 Browser output

- Place a supported object in front of the camera.
- Confirm that the live video updates without a frozen frame.
- Look for a bounding box around the detected object.
- Check that a class label and confidence value appear.
- Adjust the confidence control only if the interface provides it and the test requires a different threshold.

7.2 LED matrix output

- Confirm that the onboard LED matrix shows the highest confidence value as a percentage.
- Move the object or present another supported object and confirm the value changes.
- Remove detectable objects from view and confirm the matrix clears after 1.5 seconds.

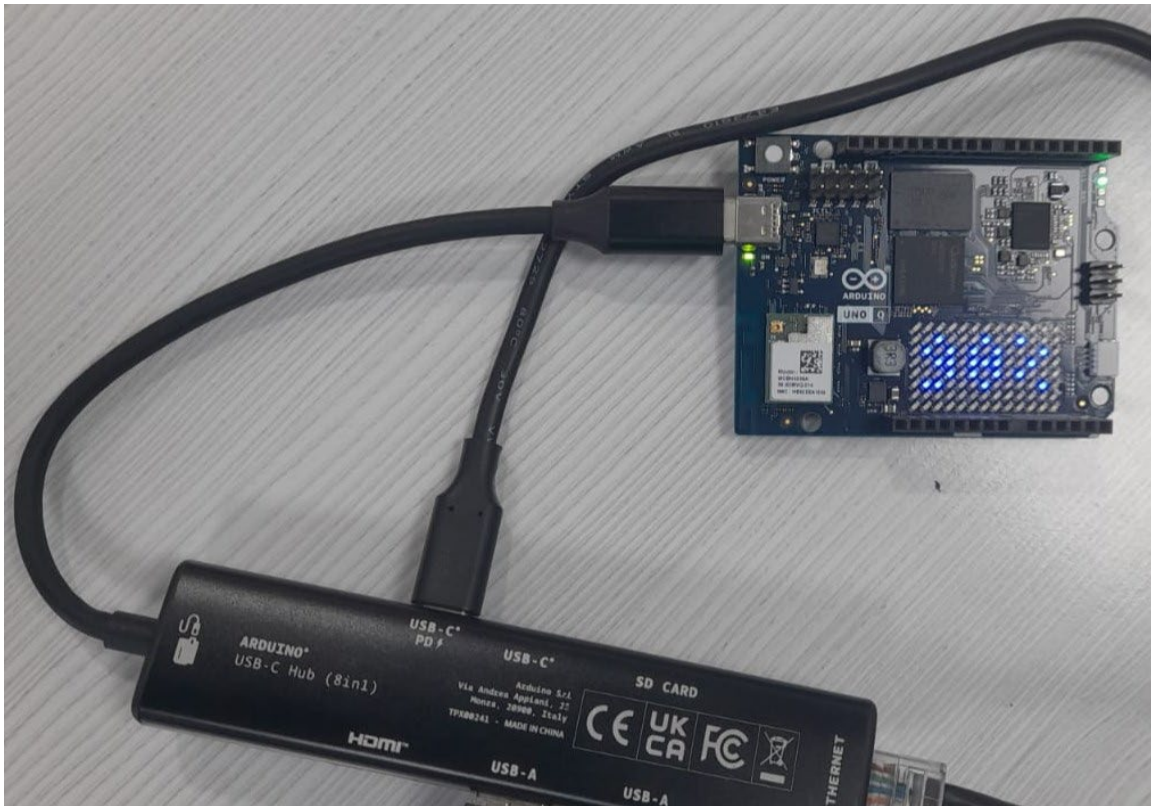


Figure 9. UNO Q LED matrix showing the confidence output

INTERPRETATION

A confidence score is the model's estimate for a detection. It is not a guarantee that the classification is correct. Validate performance with varied, representative scenes.

8. Select a different AI model

1. Open the Video Object Detection Brick.
2. Select the AI Models tab.
3. Choose Face Detection, General Purpose Object Detection or Hand Gesture Recognition.
4. Run the application again after changing the model.
5. Present an object or gesture supported by the selected model.

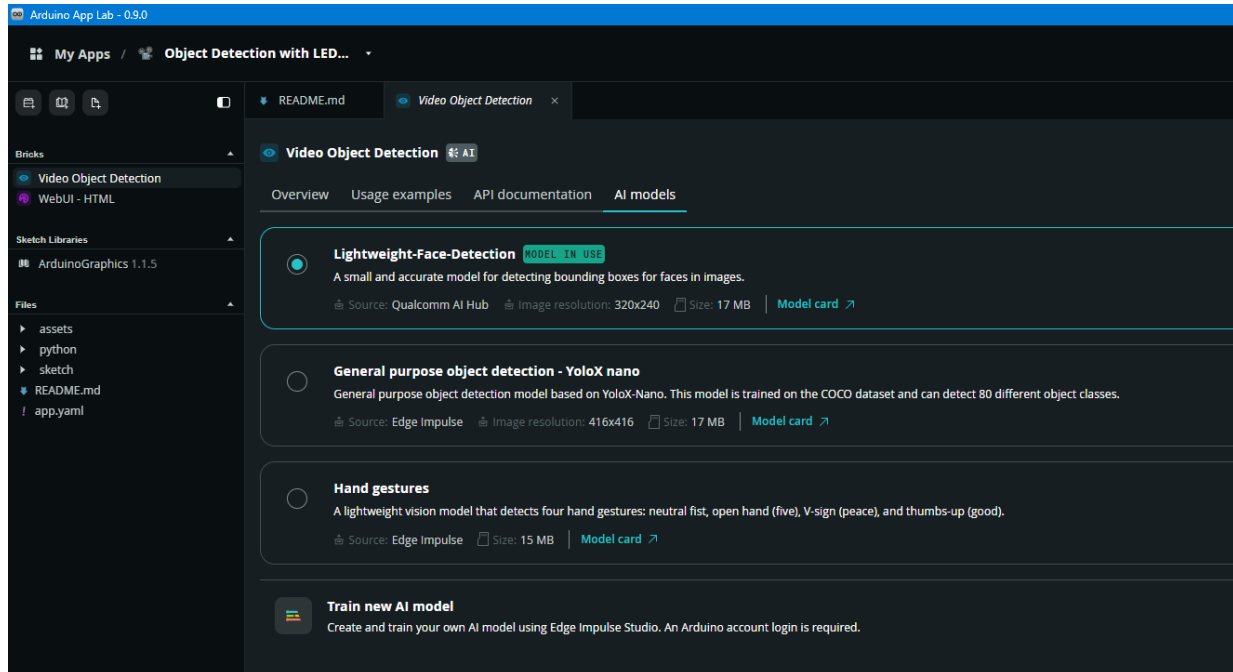


Figure 10. Model choices in the Video Object Detection Brick

8.1 General-purpose object detection

Select General Purpose Object Detection - YOLOX-Nano to test a broader range of objects. The source project identifies this model as trained on the COCO dataset with 80 object classes. Confirm detection by checking the bounding box, label and confidence value in the browser.

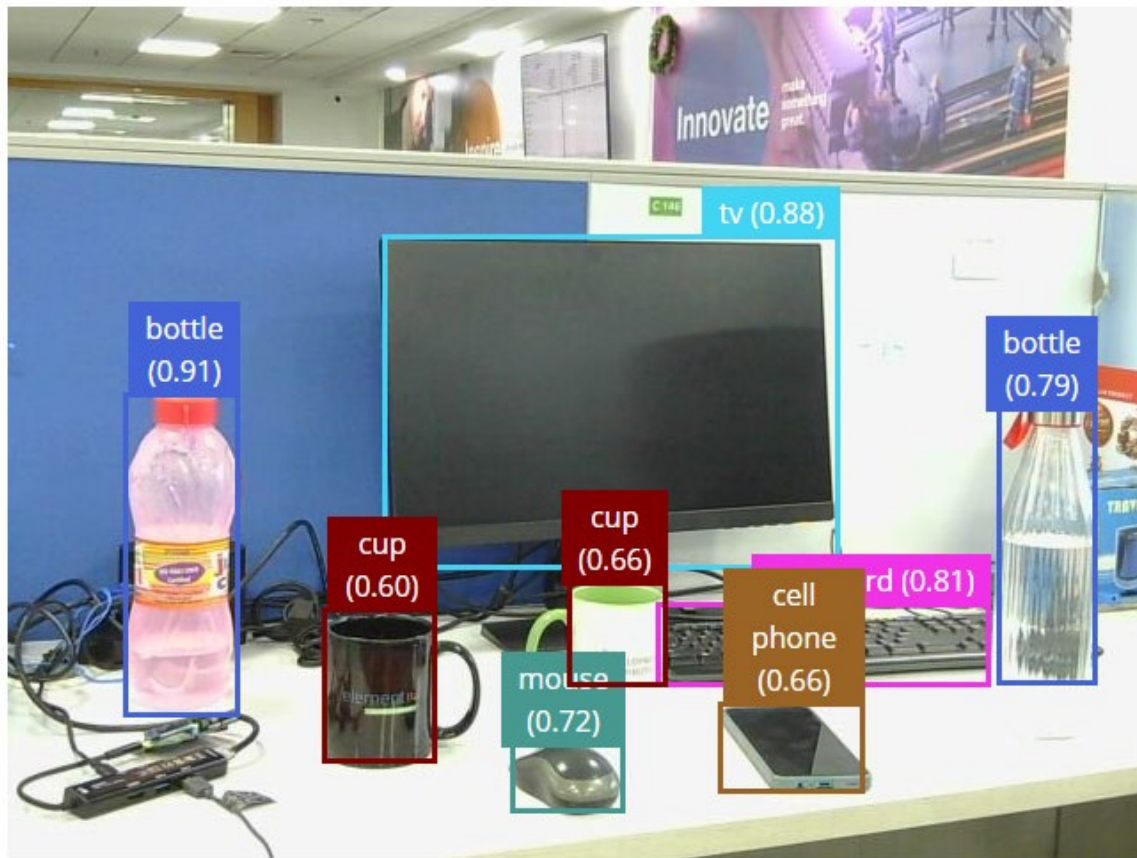


Figure 11. Example output from the general-purpose object detector

MODEL SELECTION TIP

Choose the model that best matches the target scene. A domain-specific custom model may be more appropriate when the supplied classes do not represent the required objects.

9. Train and deploy a custom model

Use this optional workflow when the built-in models do not recognise the objects required by your application. The example below uses fruit classes, but the same sequence can be adapted to another permitted dataset.

9.1 Sign in and create a project

1. In App Lab, select Train New AI Model.
2. Sign in or create an Arduino account when prompted.
3. Continue to Edge Impulse Studio using the connected account.
4. Create a new project and give it a clear, descriptive name.
5. Choose the appropriate project visibility and access settings.

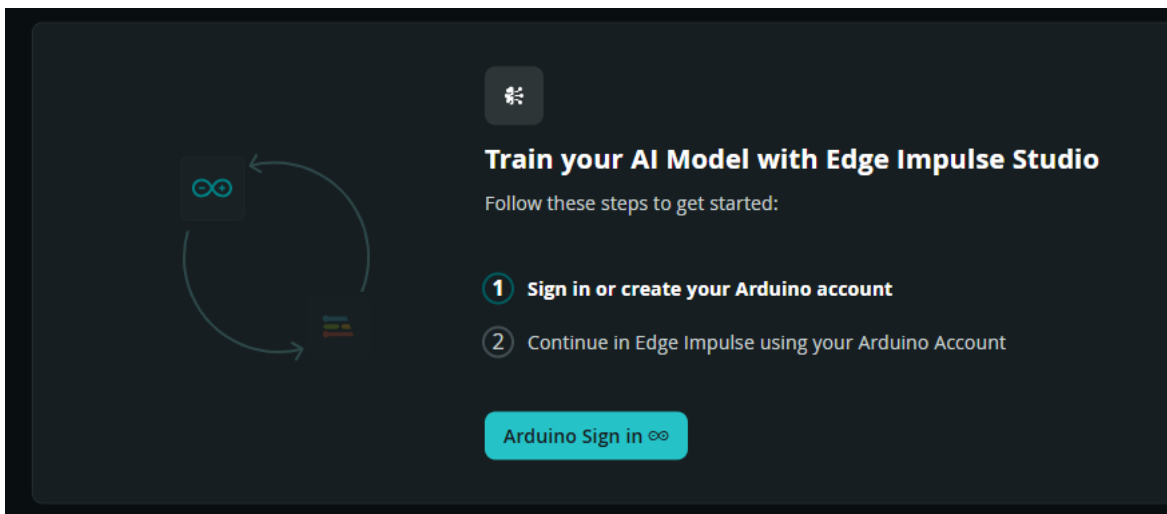


Figure 12. Starting the Edge Impulse training workflow

Create a new project

Enter the name for your new project:

Enter a name for this project

Choose your project template:

- ☒ **Per:** 60 m hours of data, limited collaboration.
- ☐ **Ent:** No job or data size limits, higher performance, custom blocks.

Create under organization: **Edge Impulse Experts**

Choose your project setting:

- ☐ **Public**
Anyone on the internet can view and clone this project under the [3-Clause BSD license](#). Only invited users will be able to edit.
- ☒ **Private (0 of 3 remaining)**
Only invited users can edit and view your project.
To request additional projects, [contact sales](#)

Create new project

Figure 13. Creating a project in Edge Impulse Studio

9.2 Build and label the dataset

Collect representative images for every object class. Include variation in angle, distance, background, lighting and object position. Use only images you own, have permission to use, or obtain from appropriately licensed datasets.

- Define short, consistent class names before labelling.
- Draw a bounding box around each target object.
- Assign the correct class to every box.
- Review labels for missing objects, loose boxes and inconsistent names.
- Keep training and test examples representative of the intended use environment.

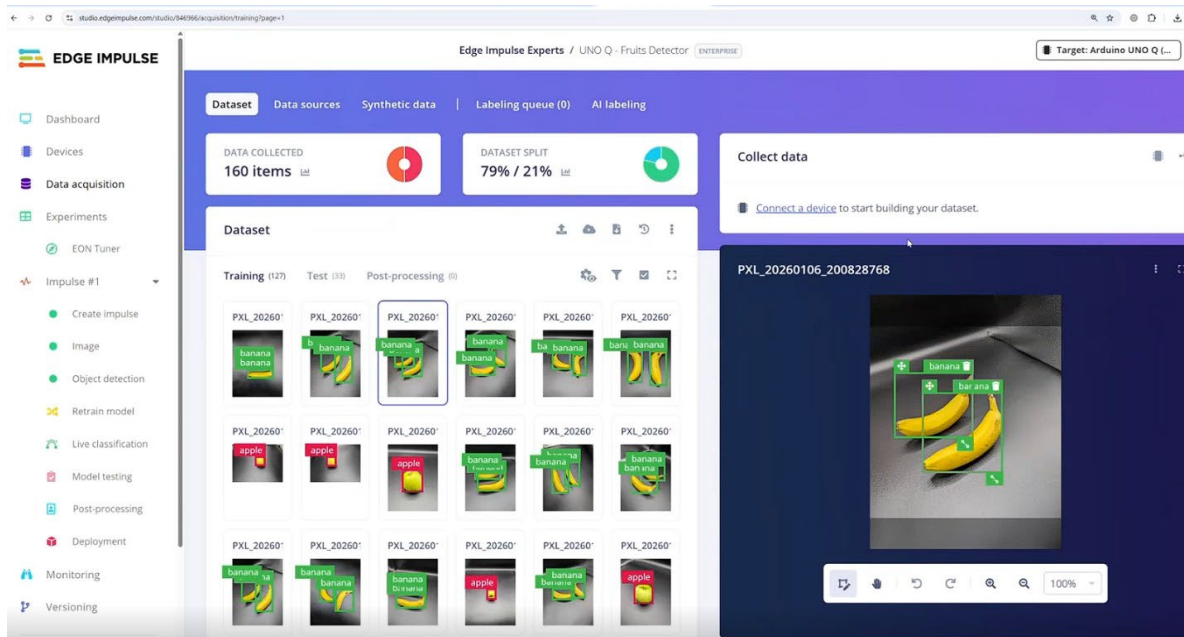


Figure 14. Dataset and bounding-box labelling example

DATASET QUALITY

More images alone do not guarantee a better model. Consistent labels and representative variation are essential for a useful evaluation.

9.3 Create the impulse

An impulse defines the processing pipeline: the input data, image-processing block and learning block used to train the detector. Start with the default settings, then adjust dimensions, colour handling, training cycles or learning rate only when the data and evaluation results justify a change.

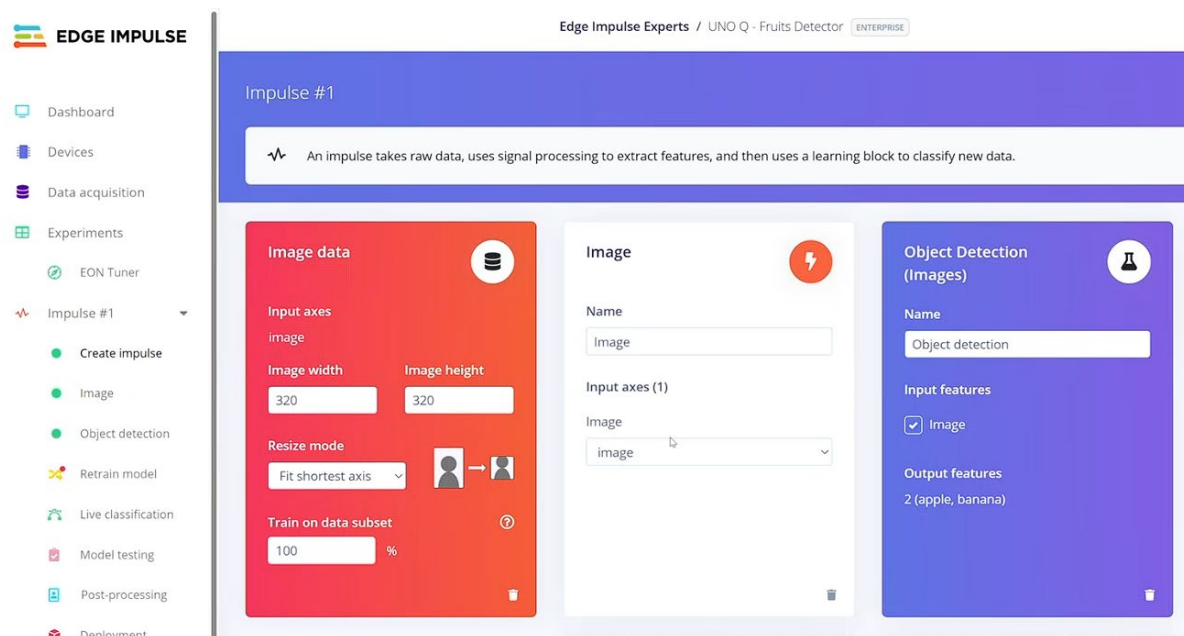


Figure 15. Image input, processing and object-detection blocks

- Set the image dimensions and resize method.
- Choose whether the pipeline uses colour information.
- Save the image-processing parameters.
- Generate and review the feature data.
- Proceed to the object-detection learning block.

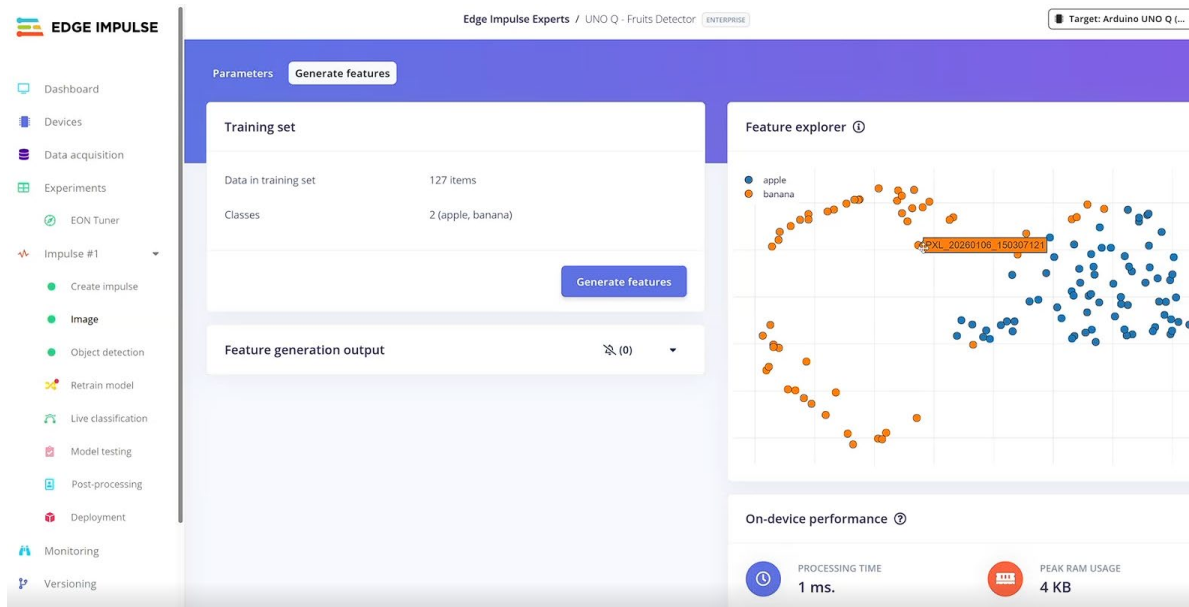


Figure 16. Generated features and feature explorer

9.4 Train and evaluate

- Review the neural-network training settings.
- Begin with the default training cycles and learning rate.
- Select Save and Train.
- Review model performance, inference-time estimate and memory information.
- Investigate weak or confused classes before deployment.

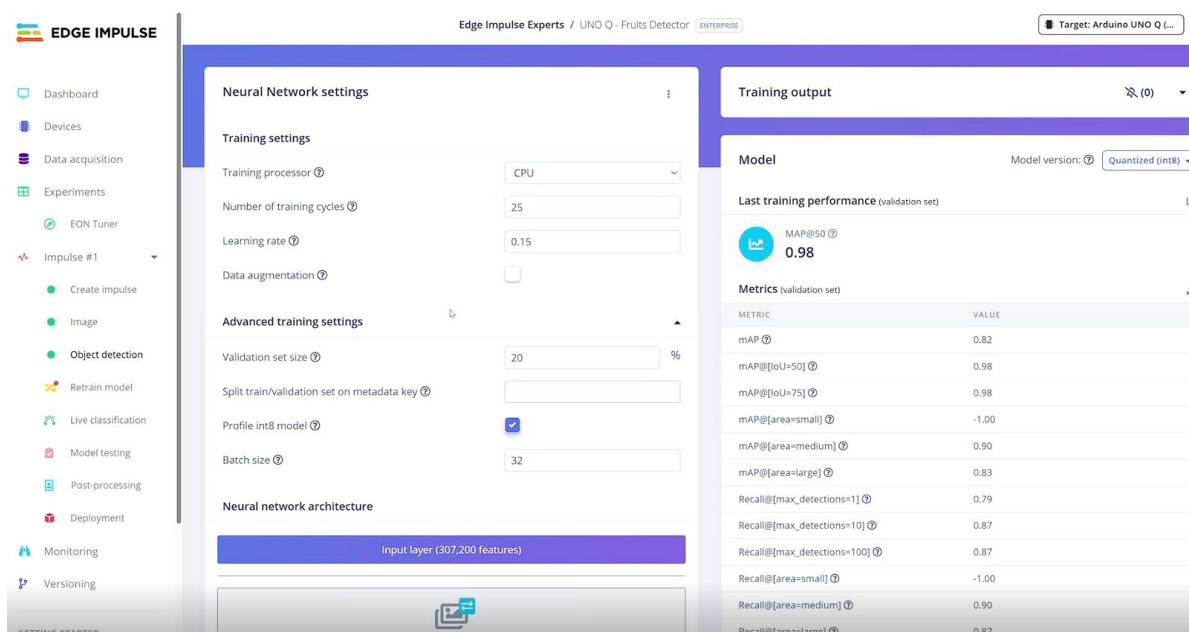


Figure 17. Training settings and model-performance results

EVALUATION CHECK

Use the test data and real camera scenes to check whether the model generalises beyond the images used for training.

9.5 Deploy to Arduino App Lab

During deployment, choose the available optimisation that matches the project goal. A quantised model can provide a smaller footprint and lower memory use, while a 32-bit floating-point model retains higher numerical precision with a larger footprint.

EDGE IMPULSE Edge Impulse Experts / UNO Q - Fruits Detector ENTERPRISE Target: Arduino UNO Q (...)

Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.

Deployment target

Arduino UNO Q
An EIM binary for the Arduino UNO Q CPU that implements the Edge Impulse Linux protocol.

Model optimizations and performance

Model optimizations can increase on-device performance but may reduce accuracy. Performance estimate for Arduino UNO Q (Qualcomm QRB2210).

Quantized (int8)

	IMAGE	OBJECT DETECTION	TOTAL
LATENCY	1 ms.	560 ms.	561 ms.
RAM	4.0K	N/A	4.0K
FLASH	-	-	-
ACCURACY	-	-	-

Unoptimized (float32)

	IMAGE	OBJECT DETECTION	TOTAL
LATENCY	1 ms.	357 ms.	358 ms.
RAM	4.0K	N/A	4.0K
FLASH	-	-	-
ACCURACY	-	-	-

To compare model accuracy, run model testing for all available optimizations. [Run model testing](#)

Latest build (current impulse)

v27 (Arduino UNO Q)
Today, 12:00:27 [Go to Arduino](#)

[View all deployments](#)

Run this model

Scan QR code or launch in browser to test your prototype



[Launch in browser](#)

Figure 18. Arduino UNO Q deployment target in Edge Impulse

1. Complete the deployment workflow for the Arduino UNO Q target.
2. Return to Arduino App Lab.
3. Open the AI Models tab in the object-detection Brick.
4. Select the model shown from the connected Edge Impulse account.
5. Run the application and test the target classes.

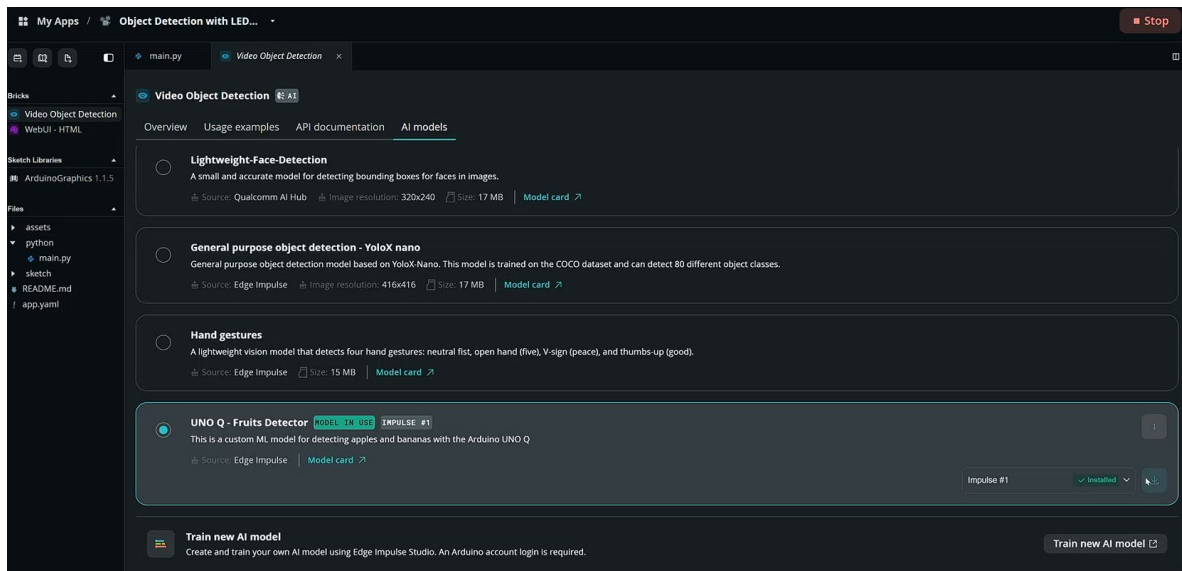


Figure 19. Custom model available in Arduino App Lab

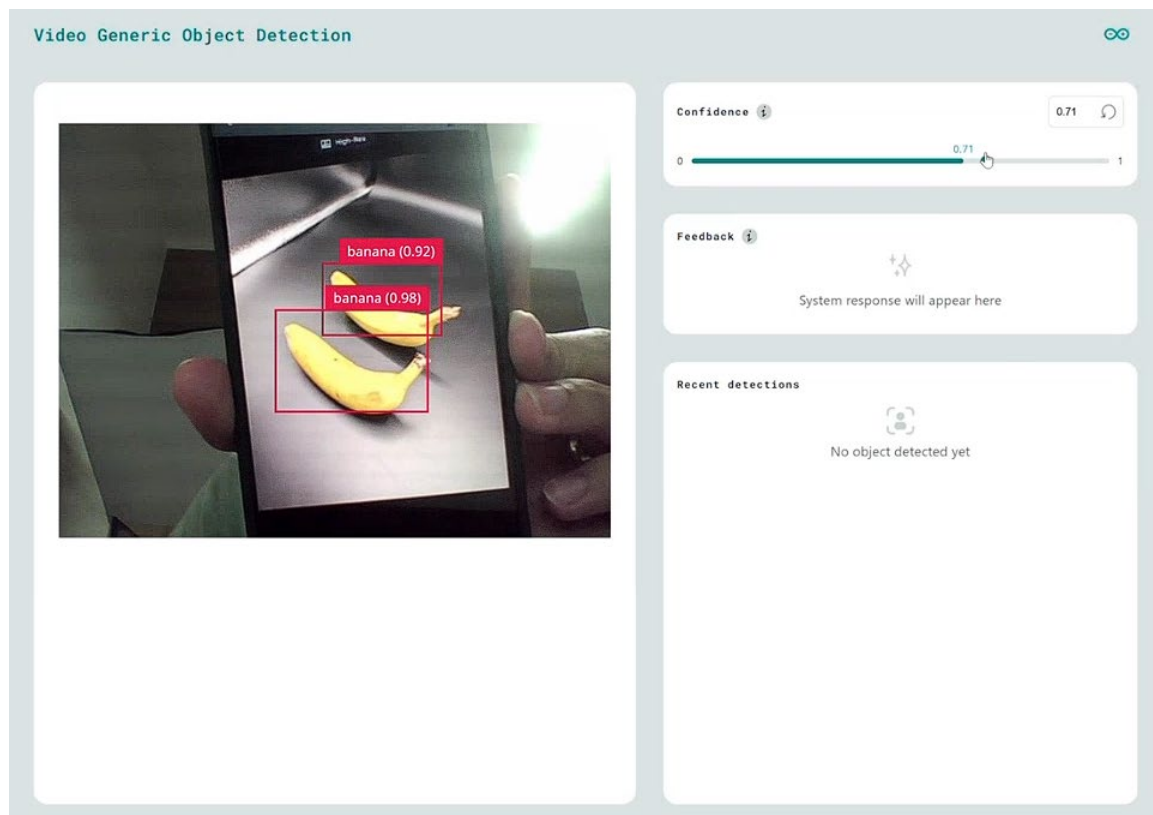


Figure 20. Custom fruit detector running in the browser interface

NO MANUAL IMPORT REQUIRED

The source workflow states that models from the connected Edge Impulse account appear directly in Arduino App Lab for selection.

10. Troubleshooting

Use the symptom-first checks below. Change one item at a time, then rerun the application so that the effect of each change is clear.

Symptom	Likely area	Recommended check
UNO Q does not appear in App Lab	Computer and board are on different networks; board setup incomplete	Confirm the one-time setup, then place both devices on the same Wi-Fi network and reopen App Lab.
Camera stream is blank	Camera not connected to the UNO Q hub; hub not powered	Reconnect the camera to the powered hub attached to UNO Q. Restart the application.
Browser does not open	Automatic launch did not occur	Open the board IP address on port 7000 using the documented address format.
No bounding boxes	Unsupported object, poor scene, or model mismatch	Improve lighting, move the target into view, and confirm the selected model supports that target.
Low or unstable confidence	Object is small, partly hidden, blurred or unlike training data	Change distance and lighting. For a custom model, add representative examples and retrain.
LED matrix does not update	Inference works but Linux-to-MCU output is not reaching the sketch	Confirm the complete project is running, then stop and rerun it. Verify the browser still produces confidence values.
Custom model is missing	Account connection or deployment workflow is incomplete	Confirm App Lab and Edge Impulse use the connected Arduino account, then review the deployment step.
Import fails	Wrong file selected or incomplete archive	Select the downloaded ZIP package and re-download it if needed.

ESCALATION EVIDENCE

When asking for support, capture the board connection state, selected model, browser output, exact error text and a photo of the hardware connections. Do not share passwords or private network credentials.

11. Validation checklist

Area	Acceptance check	Status
Platform	UNO Q appears in Arduino App Lab	☐
Network	Computer and board are on the same Wi-Fi network	☐
Hardware	Powered hub and camera are connected to UNO Q	☐
Application	Example imports and appears in My Apps	☐
Browser	Live camera stream is visible	☐
Inference	Bounding box, class and confidence appear	☐
Bridge	Highest confidence is transferred to the MCU	☐
Display	LED matrix shows the confidence percentage	☐
Timeout	Matrix clears after 1.5 seconds without detection	☐
Model change	Alternative built-in model runs successfully	☐
Custom workflow	Custom model appears and detects target classes	☐

Test record

Board / identifier: _____

App Lab version: _____

Selected model: _____

Test date: _____

Result and notes: _____

12. Extensions and glossary

12.1 Suggested project extensions

Extension	Concept
Actuation	Use the MCU output to trigger a light, buzzer, relay or motor when a selected object exceeds a defined threshold.
Quality inspection	Train a custom model for acceptable and defective examples, then connect the result to a local status indicator.
Robotics	Use detections as an input to a navigation or pick-and-place proof of concept.
Logging	Record class, confidence and timestamp data for controlled test analysis.
Human interface	Replace the percentage-only output with class-aware feedback on a connected display.
Performance study	Compare models across lighting, distance, target size and scene complexity using the same test plan.

DESIGN RESPONSIBLY

Before moving beyond a lab demonstration, assess safety, privacy, data rights, false detections, fail-safe behaviour and the consequences of automatic actuation.

12.2 Glossary

Term	Meaning in this guide
AI inference	Running a trained model on new input to produce a prediction.
Bounding box	Rectangle drawn around a detected object in an image.
Bridge	Mechanism used by the project to exchange data between the Linux and MCU domains.
Confidence score	Model estimate associated with a detection.
Edge AI	AI processing performed close to the data source, such as on the device.
Impulse	Edge Impulse processing and learning pipeline.
MPU	Application processor used for Linux, AI and computer-vision workloads.
MCU	Microcontroller used for deterministic hardware control.

Term	Meaning in this guide
Quantised model	Model represented with reduced numerical precision to reduce resource requirements.
Training dataset	Labelled examples used to fit the model.
Validation / test data	Examples used to assess performance beyond the training images.

Conclusion

This project demonstrates an end-to-end Edge AI vision workflow on Arduino UNO Q. A USB camera supplies the live stream, the QRB2210 processor performs object detection, and the Bridge sends the highest confidence score to the STM32 MCU for display on the onboard LED matrix. Arduino App Lab manages the application workflow, while Edge Impulse provides a path to custom-model training and deployment.

NEXT STEP

Use the validation checklist to establish a working baseline, then choose one controlled extension and document the new acceptance criteria before changing the application.

References used in this guide

Brand styling follows the Avnet primary palette, headline/body typography hierarchy and restrained graphic treatment.

- [Arduino App Lab download](#)
- [Source example repository](#)
- [Example application ZIP](#)
- [Edge Impulse Studio](#)