Hochschule Emden/Leer
Department of Electrical Engineering and Computer Science
Industrial Informatics

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Machine Vision with Arduino Portenta H7

Getting Started With the Portenta Vision Shield Camera

Vijay Singh

June 13, 2024

Table of Content I

Introduction

This presentation shows you how to capture frames from the **Arduino Portenta Vision Shield Camera module** and visualize the video output through a **Processing sketch.**

Objectives

- Capturing the frames from the camera
- Sending the frames as a byte stream through a Serial connection.
- Visualising the frames in Processing

Importance

Real-time object detection has applications in surveillance, industrial automation, robotics, and IoT, among others.

Required Hardware and Software

Description

The project involves the following components:

- **Arduino Portenta H7:** The core microcontroller unit providing processing power and resources for boarding different shields.
- **Portenta Vision Shield:** An accessory for the Portenta H7, equipped with a camera module and display for image capture and processing.
- **USB-C cable:** A pre-trained model deployed on the Portenta H7 for object detection tasks.
- **Arduino IDE:** A arduino software to capture the image data by the onboard camera.
- **Processing:** A processing software helps to visualize the data

Instructions:

Accessing the Portenta Vision Shield's camera data is done with the help of both Arduino and the Processing IDE. The Arduino sketch handles the capture of image data by the on-board camera, while the java applet created with Processing helps to visualize this data with the help of a serial connection. The following steps will run you through how to capture, package the data through the serial port and visualize the output in Processing.

Figures

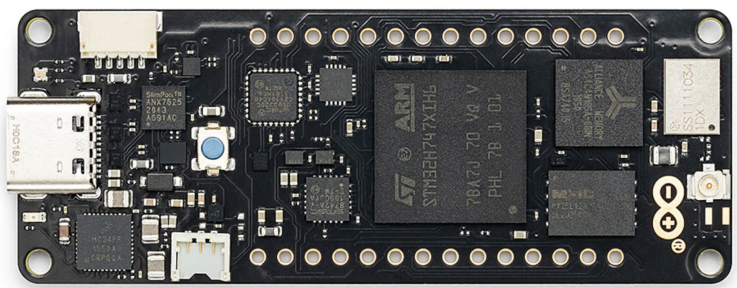


Figure1: Arduino PortentaH7

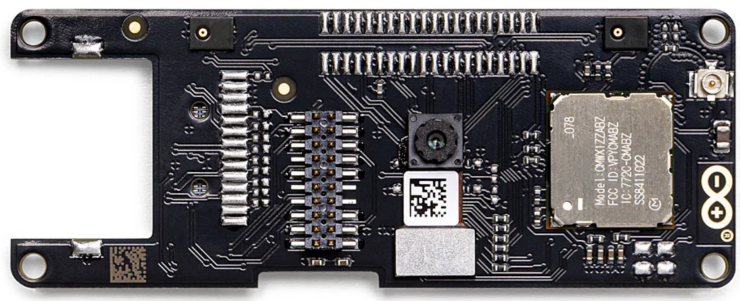


Figure2: PortentaH7 Vision Shield

1. The Basic Setup:

- a. Connect the Portenta Vision Shield to your Portenta H7 as shown in the figure. The top and bottom high density connecters are connected to the corresponding ones on the underside of the H7 board. Plug in the H7 to your computer using the USB-C® cable.

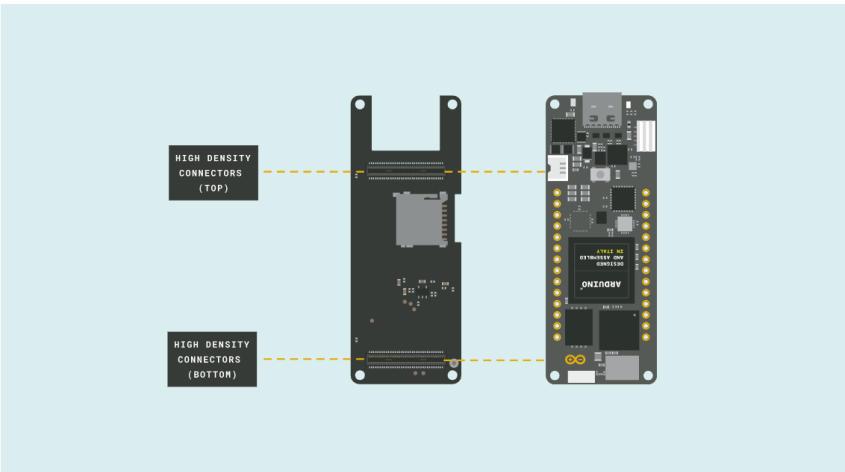


Figure3: Connection VS

- b. Open the board manager in the Arduino IDE and install the latest version of the Portenta Core which is v1.3.2

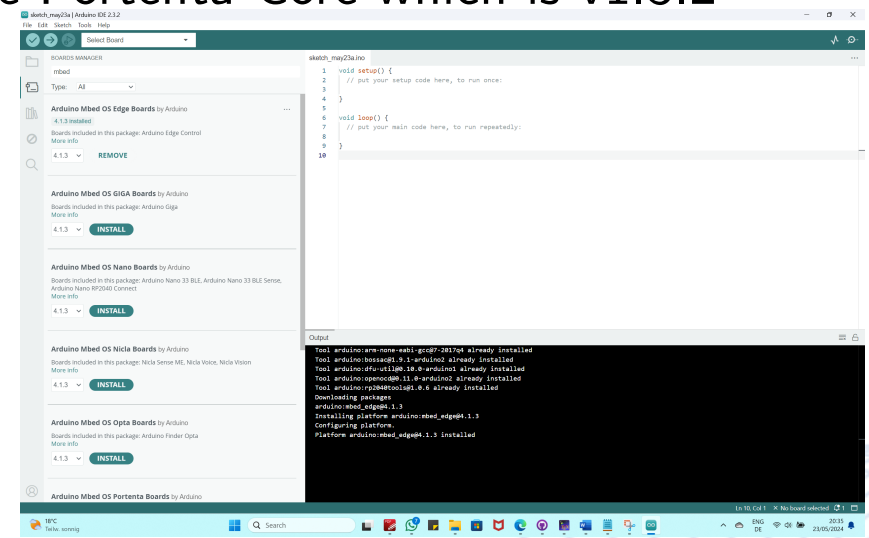


Figure4: Installation

2. Capturing the Frames

Create a new Arduino sketch called CameraCapture.ino.

To capture the frames you will need to use the functions contained in **camera.h** which comes with the Portenta core. This library contains all APIs related to frame capturing, motion detection and pattern recognition. Include the header file in your sketch. content...

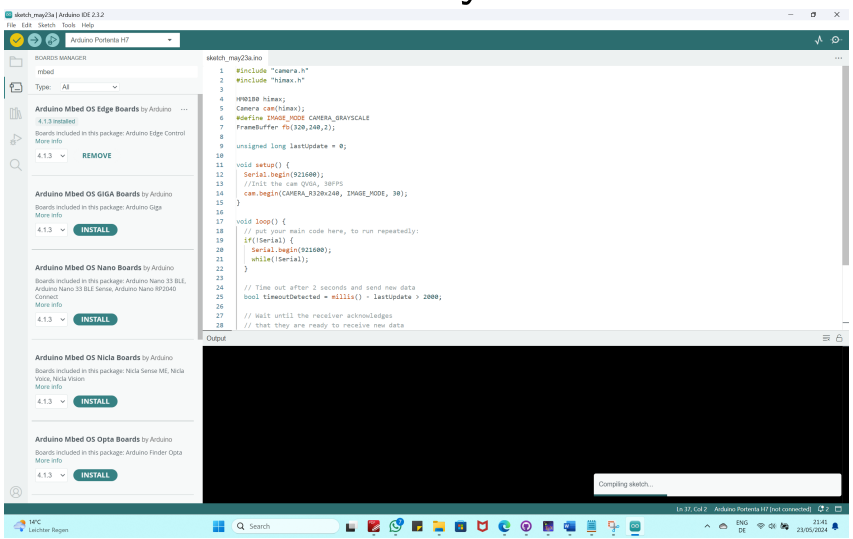


Figure5: Compiling

Capture.ino

```
#include "camera.h"
#include "himax.h"

HM01B0 himax;
Camera cam(himax);
#define IMAGE_MODE CAMERA_GRAYSCALE
FrameBuffer fb(320,240,2);

unsigned long lastUpdate = 0;

void setup() {
  Serial.begin(921600);
  //Init the cam QVGA, 30FPS
  cam.begin(CAMERA_R320x240, IMAGE_MODE, 30);
}

void loop() {
  // put your main code here, to run repeatedly:
  if(!Serial) {
```

Continue...

```
Serial.begin(921600);
while(!Serial);
}
// Time out after 2 seconds and send new data
bool timeoutDetected = millis() - lastUpdate >
    2000;

// Wait until the receiver acknowledges
// that they are ready to receive new data
if(!timeoutDetected && Serial.read() != 1)
    return;

lastUpdate = millis();

// Grab frame and write to serial
if (cam.grabFrame(fb, 3000) == 0) {
  Serial.write(fb.getBuffer(), cam.frameSize());
}
```

Processing

Download the Processing software for windows
https://processing.org/download

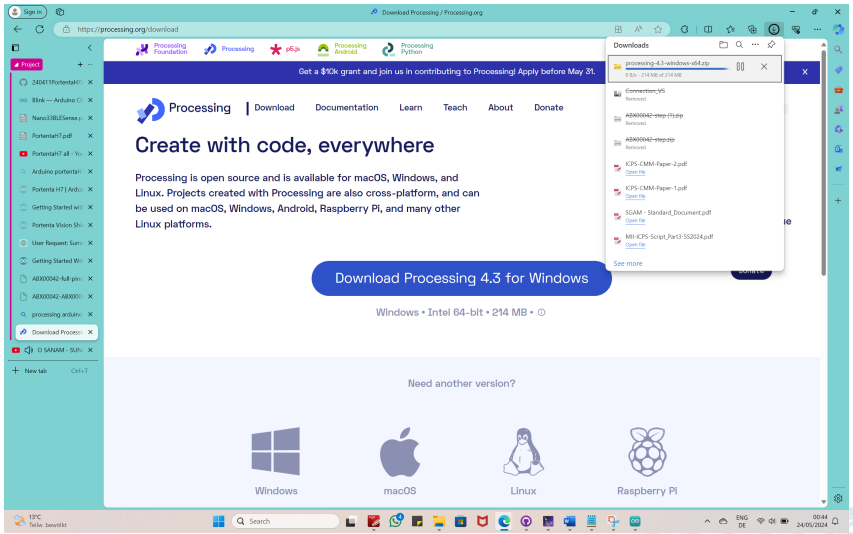
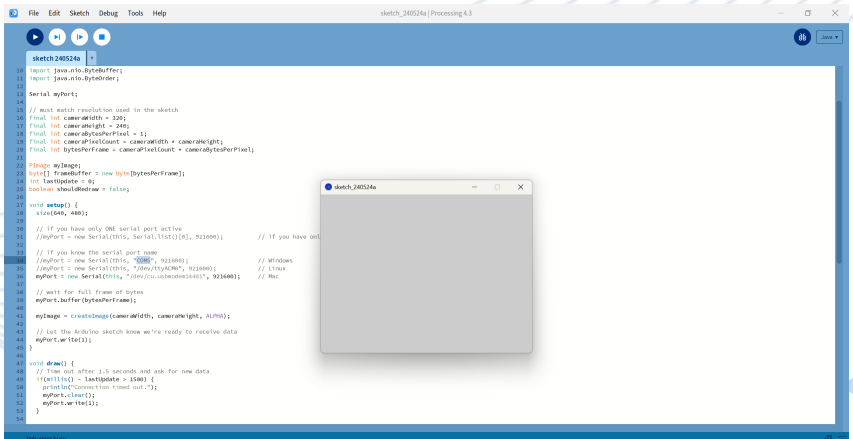


Figure6: Downloading



CameraViewer.pde

CODE

```
/*
import processing.serial.*;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;

Serial myPort;

// must match resolution used in the sketch
final int cameraWidth = 320;
final int cameraHeight = 240;
final int cameraBytesPerPixel = 1;
final int cameraPixelCount = cameraWidth *
    cameraHeight;
final int bytesPerFrame = cameraPixelCount *
    cameraBytesPerPixel;

PImage myImage;
byte[] frameBuffer = new byte[bytesPerFrame];
```

Continue...

```
int i = 0;

while (bb.hasRemaining()) {
  // read 8-bit pixel
  byte pixelValue = bb.get();

  // set pixel color
  myImage.pixels[i++] = color(Byte.
      toUnsignedInt(pixelValue));
}

myImage.updatePixels();

// Ensures that the new image data is drawn
    in the next draw loop
shouldRedraw = true;

// Let the Arduino sketch know we received
    all pixels
```

Upload the Sketch

Select the right serial port on your IDE and upload the Arduino sketch to your Portenta H7. After a successful upload, run the CameraViewer.pde sketch in Processing. You should be able to see the rendered camera output on the Processing canvas.

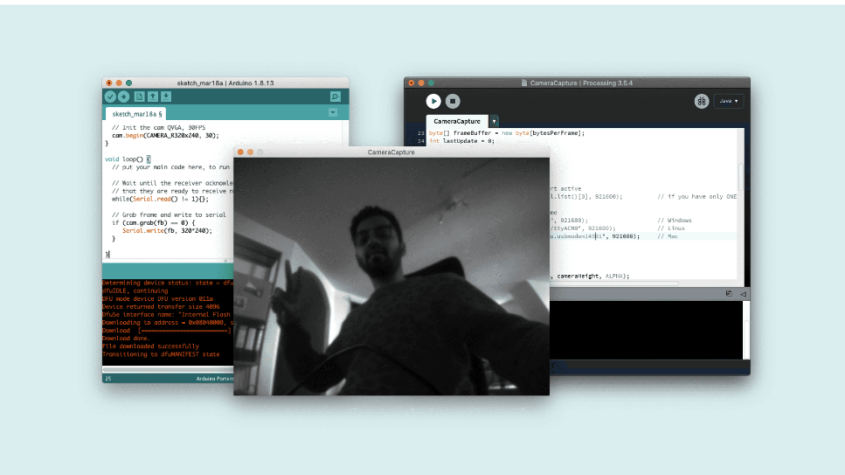


Figure8: Camera Output

Applications

Real-time object detection with the Arduino Portenta H7 and Vision Shield has numerous applications across various industries:

- **Surveillance and Security Systems:** Enhance security measures by detecting and identifying intruders or suspicious objects in real-time.
- **Industrial Automation and Quality Control:** Ensure product quality by identifying defects or anomalies on manufacturing lines.
- **Robotics and Autonomous Navigation:** Enable robots and autonomous vehicles to perceive and react to their surroundings, enhancing safety and efficiency.
- **Smart Home Devices and IoT Applications:** Create intelligent devices capable of recognizing and responding to human activities or environmental changes.

Real-time object detection provides valuable insights and automation capabilities in diverse fields, making it a versatile and powerful technology for modern applications.

Conclusion

- The project showcases the potential of Arduino vision shield
- By leveraging the computational power of the Arduino Portenta H7 and the image processing capabilities of the Vision Shield, complex tasks like object detection can be performed efficiently and accurately in real-time.
- The project opens up possibilities for a wide range of applications in industries such as surveillance, industrial automation, robotics, and IoT, where real-time object detection is crucial for decision-making and automation.

Future Work

- After discussion with Peers

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Thank you

for your attention