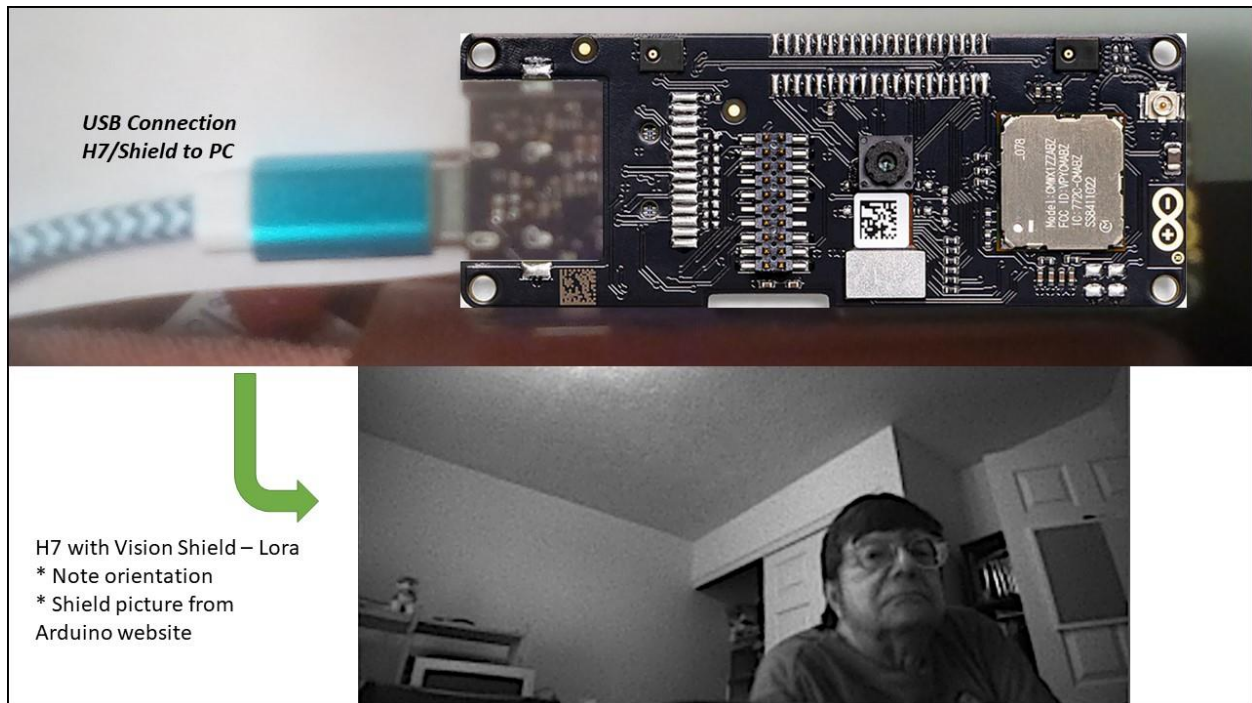


## Accessing and Displaying the Portenta Vision Shield - LoRa Camera Image

Begins a process of combining Portenta H7 with Vision Shield – LoRa by acquiring, sending, and displaying a camera image via USB connection.



**Skill Level:** Intermediate

**Tags:** Image, LoRa, Remote

### Contributors

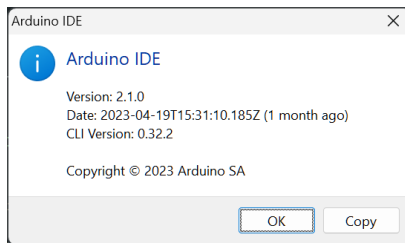
In performing this project, I learned a great deal from a project by Pablo Marqu nez (<https://docs.arduino.cc/tutorials/portenta-vision-shield/camera-to-bitmap-sd-card?queryID=7e0ba1ee5832016d62b016e8d34786c6>) and from another project by Lenard George and Sebastian Romero (<https://docs.arduino.cc/tutorials/portenta-vision-shield/getting-started-camera>).

### Components

- [Arduino Portenta H7 Microcontroller Board](#)
- [Arduino Portenta Vision Shield – LoRa](#)
- [Arduino USB Type-C Cable 2-in1](#)

If you have the H7 microcontroller board and the [Ethernet version of the vision shield](#), that combination will work as well. I stress the LoRa version of the vision shield because my ultimate intention is to transmit the camera’s images via LoRa broadcast. Future projects will build toward that goal.

## Tools:



The [Arduino IDE](#) is used to develop, upload, test, and debug code for the Arduino H7 microcontroller board. v2.1 of the IDE, the version I use, is different from the 1.x version. Be patient with it. You will grow to like it. Various “getting started” tutorials are [available](#).

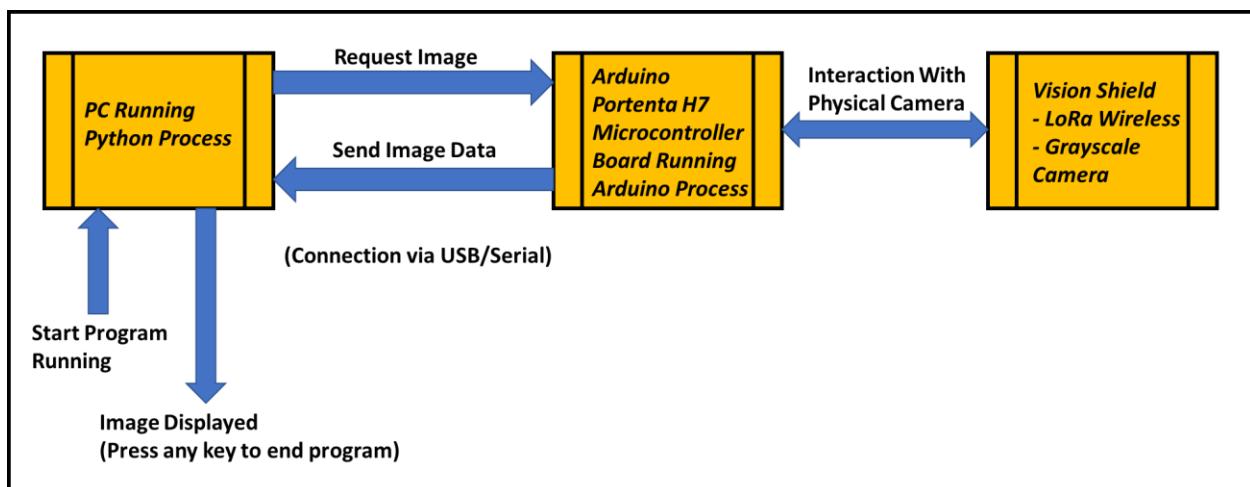
## Applications and Platforms

In many ways, this project is an example of distributed processing. In particular, it is an example of query/response. Arduino waits to be asked for a camera image to be sent. Another process, not resident on the Arduino, makes the request, receives the data, and displays the result. That second process is written in [Python](#) v3.10. (Use a later version if you wish.) For myself, I like to use the [PyCharm IDE – Community Edition](#) IDE. This project was produced on a Windows 11 computer. However, there is nothing about the resulting software that is peculiar to Windows operating systems, except the name of the serial port within the Python process.

## Project Description

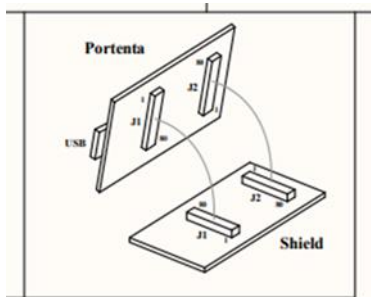
In any application involving sensors placed in remote locations, it is necessary to be able to access their output without having to be physically present. Thus the growth of remote sensing. By combining the Arduino Portenta H7 microcontroller board with its Vision Shield – LoRa one can easily imagine taking pictures remotely and receiving the associated data via LoRa broadcast. This should be doable without necessarily using LoRaWAN or any third-party services. Of course, streaming video would not be possible. But, very-low framerates should certainly be possible. That is the overall goal.

In this project, we simply exercise the camera on the Vision Shield – LoRa, which is connected to the Arduino Portenta H7 microcontroller board. We determine the camera’s physical orientation and then access the camera’s output according to that orientation. The data is sent via USB/serial connection to a PC, where the data is interpreted and displayed by a process written in Python.



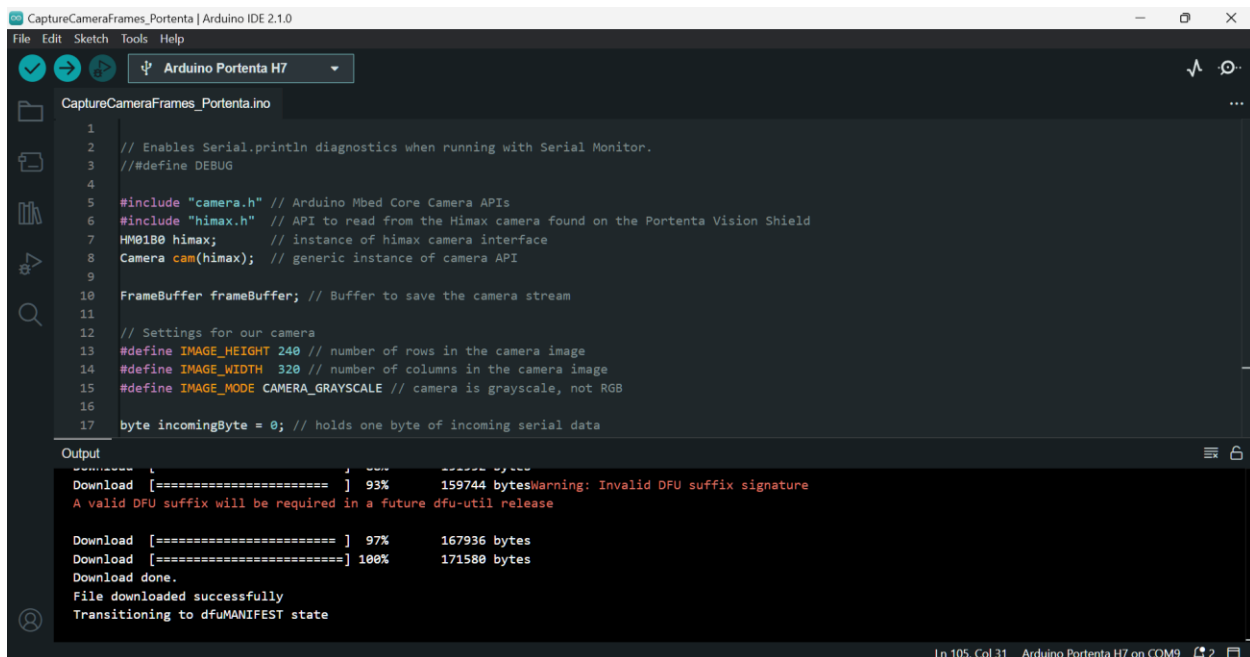
## Arduino Process

First connect your shield to the H7. Page 5 of the [shield's schematics](#) shows how to do that.



Notice that area to the left of the shield's J1 connector is horseshoe-shaped on the LoRa version and contains an Ethernet connection for the Ethernet version. As you make the connection, beware of both boards' pins and other mountings. Do not put a lot of pressure on them. Some videos I have seen recommend using a wrist grounding strap when working with the boards. A light pressure when connecting the boards will result in a clear "snap" when the connectors are in place. A light tug will not release them. This took me a couple of tries to get right. Once the boards are mounted to each other, open the

Arduino IDE's board manager. Then install the latest version of the Portenta Core. Now you are ready to upload the Arduino's code.



```
1 // Enables Serial.println diagnostics when running with Serial Monitor.
2 // #define DEBUG
3
4 #include "camera.h" // Arduino Mbed Core Camera APIs
5 #include "himax.h" // API to read from the Himax camera found on the Portenta Vision Shield
6 HW01B0 himax; // instance of himax camera interface
7 Camera cam(himax); // generic instance of camera API
8
9
10 FrameBuffer frameBuffer; // Buffer to save the camera stream
11
12 // Settings for our camera
13 #define IMAGE_HEIGHT 240 // number of rows in the camera image
14 #define IMAGE_WIDTH 320 // number of columns in the camera image
15 #define IMAGE_MODE CAMERA_GRAYSCALE // camera is grayscale, not RGB
16
17 byte incomingByte = 0; // holds one byte of incoming serial data
```

Output

```
Download [=====] 93% 159744 bytesWarning: Invalid DFU suffix signature
A valid DFU suffix will be required in a future dfu-util release
Download [=====] 97% 167936 bytes
Download [=====] 100% 171580 bytes
Download done.
File downloaded successfully
Transitioning to dfuMANIFEST state
```

Once the upload is completed, the H7 will be running and awaiting a request to send an image. That request is sent by the Python process.

Within the Arduino's code are comments that offer details on what the code is doing. See the `CaptureCameraFrames_Portenta` directory for this code.

## Python Process

Once started, the Python process connects to the Arduino via the serial port, requests an image, receives that image, and creates a display from that data. When you are finished looking at the displayed result, press any key to end the process. This particular code just runs once but the Arduino stays ready for another request. The code is run again when another image is desired. This basic arrangement can be embedded in a more involved user-driven process.

Here is the code for the Python process. There are comments in the code that offer details on what is going on. See the `CaptureCameraFrames_Python` directory for this code.

### **Closing Thoughts**

With this project we have taken a brief step into using the Arduino Portenta H7 Vision Shield – LoRa. We can now reliably access and transmit an image taken by the shield's camera. Transmission is via a PC's USB/serial connection to the Arduino. This shows we know how to access the camera and that we understand how its data is structured. Now that we have that basic ability. There is more that can be done. It would be great if I could find `camera.h` and `himax.h` since that might lead to greater understanding of camera access and the resulting data structure.