AI Hardware Project Milestone 1

Team Members: Nathan Hersel, Josiah Suwargo, Owen Singley

Setting Up The Hardware:

In order to set up the hardware for this project, the Google Coral Dev Board Micro was connected to a laptop running a Linux Virtual Machine so that code could be flashed to the dev board. Figure 1 shows a screenshot of the virtual machine that was used to execute the code for this project. While doing this we ran into several issues that had to be overcome. The first issue was with setting up the virtual machine and connecting the virtual machine to the internet. It was also determined that the cable being used at first was only a charging cable and was not capable of transmitting data from the computer to the Dev Board and vice versa. Once these issues were solved and the board was connected to the computer, as shown in Figures 2 and 3, the hardware interface image shown in Figure 4 showed a good connection to the laptop. Next, FreeRTOS and the source code for the Dev Board had to be downloaded. This was done by following the instructions shown in Figure 5. In order to test the board and the software that had been installed, a face detection example was flashed to the board and ran successfully (Figure 6). For this example, a green LED turned on whenever a face was put into view of the camera. This worked very smoothly so the team moved on to working with the image classification models. Overall, the hardware has been set up successfully and is running certain example programs as expected.

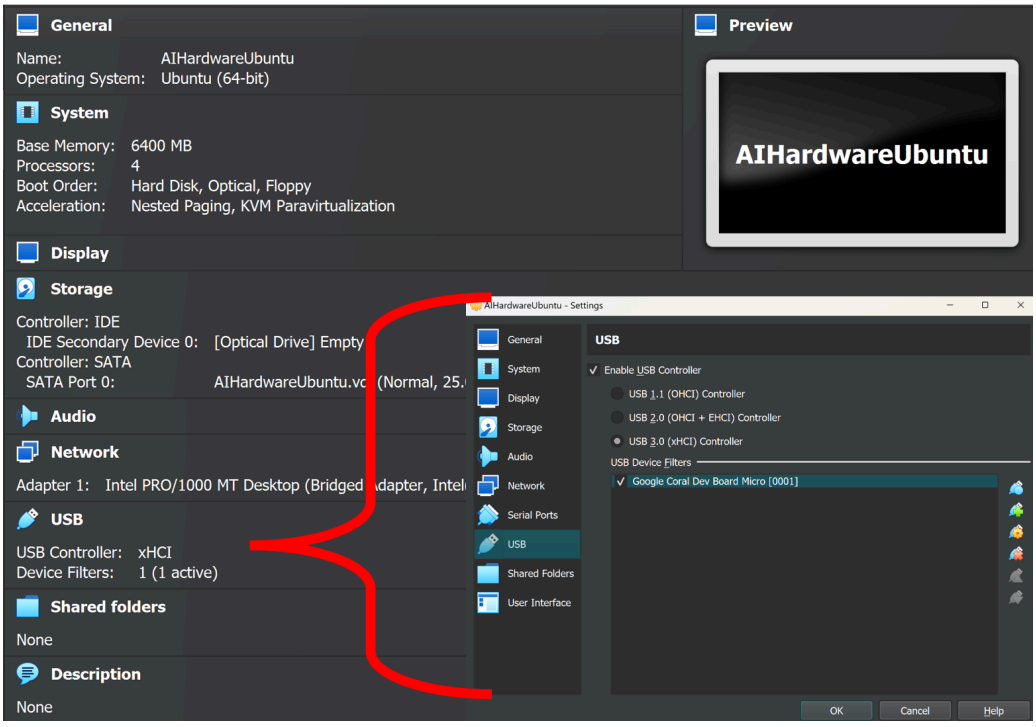

Figure 1. Ubuntu Virtual Machine with USB Configuration

Figure 2. Connected Google Coral Dev Board Micro to Computer

```
nathanhersel@Nathans-MacBook-Pro coralmicro % lsusb
Bus 002 Device 002: ID 18d1:9308 Google Inc. Coral Dev Board Micro  Serial: 220a
b00e828fd0ce
Bus 000 Device 000: ID 18d1:9308 Google Inc. USB 3.1 Bus
Bus 000 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
nathanhersel@Nathans-MacBook-Pro coralmicro %
```

Figure 3. Screenshot of Terminal Showing Connected Dev Board

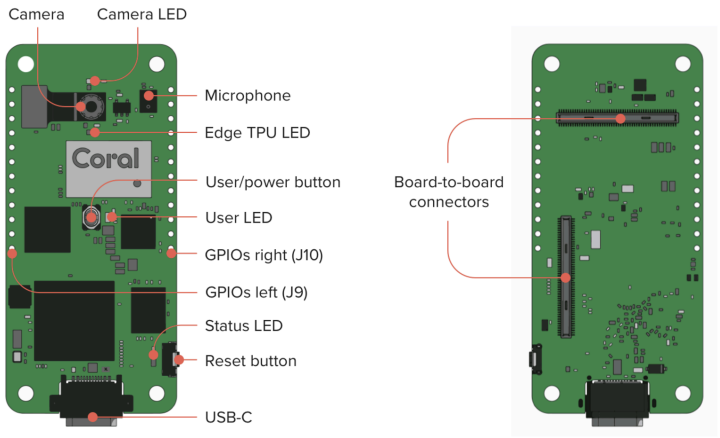

Figure 4. Google Coral Dev Board Micro Hardware Interfaces

4. Set up for FreeRTOS development

Note: FreeRTOS is our primary application framework for the Dev Board Micro, but you can also **build apps using Arduino**.

To start building apps with FreeRTOS, you need to download the Dev Board Micro source code and install some dependencies, as follows:

1. Clone the coralmicro repo and all submodules:

```
git clone --recurse-submodules -j8 https://github.com/google-coral/coralmicro
```

This could take a few minutes.

2. Now make sure you have the necessary development tools (such as **CMake**) by running this setup script:

```
cd coralmicro && bash setup.sh
```

Now you're ready to build and flash some apps to the board.

Figure 5. Instructions Followed to Setup FreeRTOS Development

5. Try the face detection example

The coralmicro repo includes a variety of code examples for you to try. Let's start with the face detection example, which runs a MobileNet face detection model on the Edge TPU:

1. With your terminal still in the `coralmicro` directory, build all the projects with this script:

```
bash build.sh
```

2. Plug in the board to your computer and verify that it's detected:

```
lsusb
```

You should see the board as `Google Inc. Coral Dev Board Micro`.

3. Flash the face detection example:

```
python3 scripts/flashtool.py -e detect_faces
```

Figure 6. Instructions Followed to Run the Face Detection Example

Identifying Models:

In this project, we utilized several AI models to achieve our goal. While there are many options to choose from, our team settled on these 3 models below:

1. MobileNetV1

   The MobileNetV1 was one of our more obvious choices in mind as it is already in the Coral micro repository, this makes for easier integration as the testing will be more consistent and compatible with our tools. We also liked the MobileNetV1 as it is very computationally efficient, making it faster to get results, while this does make it simple and fast, it also has a tradeoff in which it may not perform as well as the other models for more complex tasks.
2. MobileNetV2

   Similar to the MobileNetV1, the MobilenetV2 also has the advantage of already being in the repository, making it easier for integration and testing once more. But unlike the V1, the V2 does have improved accuracy whilst maintaining the same efficiency as the V1. It also is able to perform more complex tasks than the V1, giving it an edge. Despite all this it might still falter behind when compared to the new cutting edge-models.
3. InceptionV4

   We found that InceptionV4 was developed by Google specifically for image classification, which aligns with our goal. It is known for its exceptionally high accuracy which further benefits our project. Despite all the high specs the InceptionV4 has, it does have a very high computational cost when compared to the MobileNet models and has a slower inference time, despite all this we do think that it will be very worthwhile to use the Inceptionv4.

We plan on using each model and compare the results in the end, giving us a better understanding of the pros and cons of each model, allowing for more informed usage in the future.

Program Hardware:

First we tested the Coral Dev Board Micro with a pre-existing face detection model located on the coralmicro Github repository. To do this we flashed the board with the software using the Python flashtool. Once flashed, the board was expected to light up an LED if the on-board camera detected a face in its field-of-view. This program worked as expected. Since we had determined that the flashtool and the camera both functioned correctly, we moved on to flashing the image classification programs onto the board.

Located in the coralmicro Github repository are two image classification programs written in C++. One classifies images taken from the on-board camera and streams the output to a Python GUI. The other uses a hard-coded file path as input and prints the classification output to the serial console. We successfully flashed both of

these programs to the board separately, but ran into bugs with each program. The program using the Python GUI exhibited a hostname error. For the other program, we successfully connected to the board's serial console, but did not see anything printed to the console. While these bugs are ongoing, we have plans to test and resolve them in the near future.

In addition to programming the physical board, we began programming the cloud-based portion of our project as well. This involved setting up a Google Colab document and writing a Python program to take an image and classify it using the MobileNet v1, v2, and InceptionV4 models for image classification. We modeled much of the code after the homework on image classification.

Run First Iteration of Tests:

The first tests we were able to run on the Coral Dev Board Micro were to ensure that the camera and provided flashtool worked correctly. Both of these items were determined to be working as expected when tested with the face detection program described in the "Program Hardware" section.

Next we were able to test the image classification programs on the board. However, this testing did not produce as successful results. We were unable to receive a good output from the board when using either program, but we were able to identify some of the problems. We believe that the hostname issue could be resolved by identifying the hostname of the board using the "lsusb" command and using this as input to the Python GUI. It is also possible that because our board does not have the WiFi extension board, the Python GUI will not work properly because it requires an RPC server. For the image classification program using a file as the input, we think our issue was primarily due to accessing the serial console after the board had already printed. We believe a solution may be to alter the C++ code to delay the serial console print until after the user button has been pressed.