# Deploying the Pretrained KWS Model

In this reading we are going to first generate a binary file representing the pre-trained KWS model and then deploy that model to our Arduino using the Arduino IDE.

## Converting the TFLite Model File into a Binary Array

The first step in deploying the pretrained KWS model is to take the quantized TensorFlow Lite model you created in Course 2 and convert it into a binary array that we can load onto our microcontroller. And don't worry if you don't still have the `.tflite` file from Course 2 we have provided a direct link to the staff copy in the following Colab.

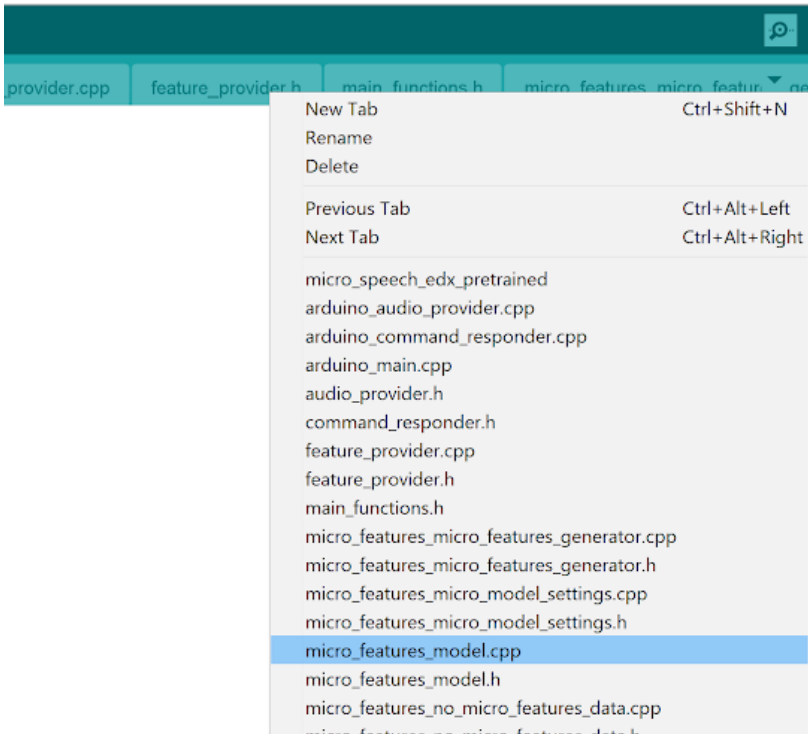
Please run this Colab and then either leave the tab open with the printout of the final binary file and/or download the final `.cc` model file and open it up in a separate text editor. In either case we will need to copy the data in that file into our Arduino sketch in the next section so make sure you keep it in an easy to access place!

https://colab.research.google.com/github/tinyMLx/colabs/blob/master/4-5-16-KWS-PretrainedModel.ipynb

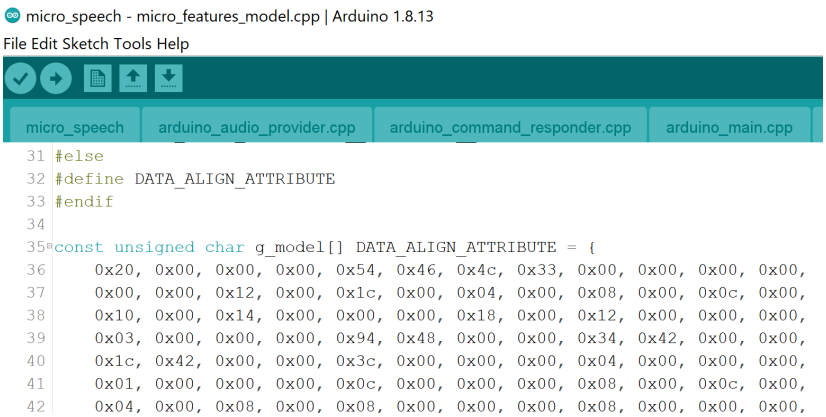

## Updating the Arduino Code

1. Open the micro_speech.ino sketch, which you can find via the File drop-down menu. Navigate, as follows: `File → Examples → Harvard_TinyMLx → micro_speech`.

2. Navigate to the `micro_features_model.cpp` file and update the model. You can find that file by selecting it from tabs across the top of the Arduino IDE. If that file is not visible you can navigate to it (or other additional files) by clicking on the downward facing triangle at the end of the tabs which will open up a dropdown showing all of the files.



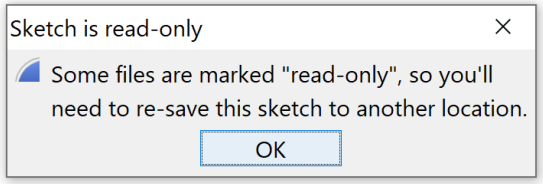

   a. Copy the binary model file contents from the `KWS_yes_no.cc` file into the `micro_features_model.cpp` file. **Make sure to only copy the binary data inside the `{ }` as the variable type is different in the downloaded or printed `KWS_yes_no.cc` file** (it is of type `unsigned char` in the `.cc` file but it needs to be of type `const unsigned char` with the `DATA_ALIGN_ATTRIBUTE` in the `.cpp` file). If you lost your `KWS_yes_no.cc` file, don't worry, you can [use the staff's copy!]



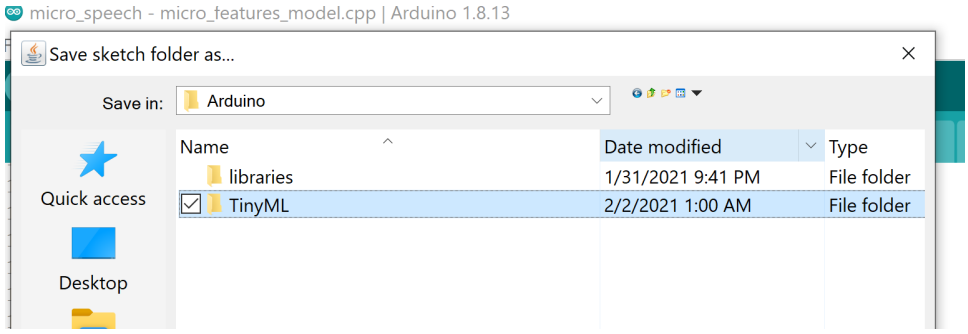
b. Next scroll all the way down to the bottom of the file and replace the model length. Again note that the `.cpp` file needs the variable to be of type `const int` while the `.cc` file will show `unsigned int`. Our suggestion again is to simply copy the numerical value.

```
1590     0x02, 0x00, 0x00, 0x00, 0x00, 0x00
1591     0x06, 0x00, 0x00, 0x00, 0x00, 0x16
1592     0x00, 0x00, 0x08, 0x00, 0x0a, 0x00
1593     0x04, 0x00, 0x00, 0x00, 0x00, 0x00
1594     0x00, 0x00, 0x08, 0x00, 0x0a, 0x00
1595     0x03, 0x00, 0x00, 0x00};
1596 const int g_model_len = 18712;
```

3. Next save your changes. You will most likely see a popup as shown below asking you to save a copy of the example as all examples are treated as “read-only” by default.



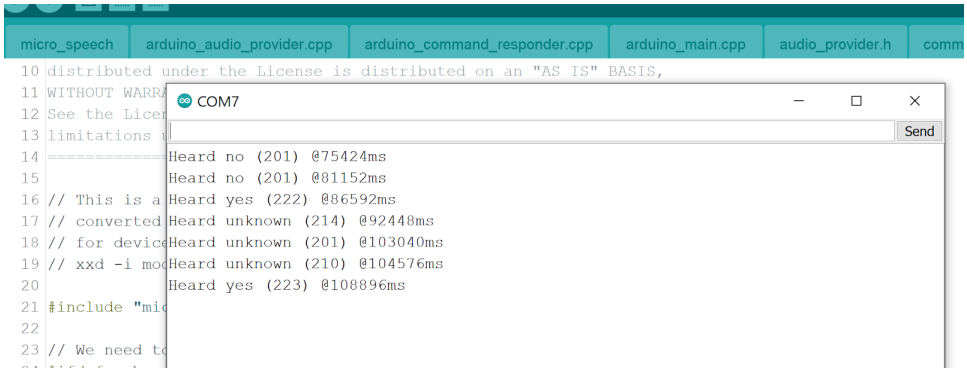

We suggest that you make a folder called e.g., `TinyML` inside of your `Arduino` folder. You can find your main `Arduino` folder either inside of your `Documents` folder or in your `Home` folder, and save it in that folder with a descriptive name like `micro_speech_edx_pretraied`. That said, you can save it wherever you like with whatever name you want!



# Deploying the Pretrained Model

1. Use a USB cable to connect the Arduino Nano 33 BLE Sense to your machine. You should see the green LED power indicator come on when the board first receives power.
2. As always, use the Tools drop down menu to select appropriate Port and Board.
    a. Select the Arduino Nano 33 BLE as the board by going to `Tools → Board: <Current Board Name> → Arduino Mbed OS Boards (nRF52840) → Arduino Nano 33 BLE.`
    b. Then select the USB Port associated with your board. This will appear differently on Windows, macOS, Linux but will likely indicate 'Arduino Nano 33 BLE" in parenthesis. You can select this by going to `Tools → Port: <Current Port (Board on Port)> → <TBD Based on OS> (Arduino Nano 33 BLE).` Where <TBD Based on OS> is most likely to come from the list below where <#> indicates some integer number
        i. Windows → `COM<#>`
        ii. macOS → `/dev/cu.usbmodem<#>`
        iii. Linux → `ttyUSB<#>` or `ttyACM<#>`
3. Use the rightward arrow next to the 'upload' / flash the code. You'll know the upload is complete when you red text in the console at the bottom of the IDE that shows 100% upload of the code and a statement that says something like "Done in <#.#> seconds."

    If you receive an error you will see an orange error bar appear and a red error message in the console (as shown below). Don't worry -- there are many common reasons this may have occurred. To help you debug please check out our FAQ appendix with answers to the most common errors!
4. You should now have a working Keyword Spotting model for "Yes" and "No" deployed to your Arduino! To test it out, open the Serial Monitor. You should start to see the result of the model "Yes, No, Unknown, Silence" begin to display in the Serial Monitor.



This should match the multicolor LED near the bluetooth module on the board changing colors as follows (we've shown a picture of it turning green below):

```
Yes     = Green LED
No      = Red LED
Unknown = Blue LED
Silence = Nothing
```