TensorFlow Lite for Circuit Playground
Bluefruit Quickstart

Created by lady ada

https://learn.adafruit.com/tensorflow-lite-for-circuit-playground-bluefruit-quickstart

Last updated on 2021-11-15 07:54:21 PM EST
# Table of Contents

**Overview**  
- Parts  

**Setup For Compiling Examples**  
- TensorFlow Libraries  
- Select Board  
- For PyBadge/EdgeBadge  
- For Circuit Playground Bluefruit

**Sine Wave Demo**  
- Serial plotter sine wave demo compile & upload  
- Arcada display output sine demo compile & upload

**Customized Wave Demo**  
- Re-creating the Default Sine Wave Model  
- Creating a Cosine Wave Model  
- Loading Models From Internal Storage

**Gesture Demo**  
- Serial out gesture demo compile & upload  
- Wing  
- Ring  
- Slope  
- Arcada display output gesture demo compile & upload

**Micro Speech Demo**  
- Micro speech demo compile & upload
Overview

Machine learning has come to the ‘edge’ - small microcontrollers that can run a very miniature version of TensorFlow Lite to do ML computations.

But you don't need super complex hardware to start developing your own TensorFlow models! Using our beginner board, the Circuit Playground Bluefruit you can build & test various examples that run on the nRF52840 chip + all the cool sensors built in!

Parts

You'll need at least a Circuit Playground Bluefruit

Circuit Playground Bluefruit - Bluetooth Low Energy
Circuit Playground Bluefruit is our third board in the Circuit Playground series, another step towards a perfect introduction to electronics and programming. We've...
https://www.adafruit.com/product/4333

Many of the projects work a lot better, with graphical output, when you add on a TFT Gizmo!

Circuit Playground TFT Gizmo - Bolt-on Display + Audio Amplifier
Extend and expand your Circuit Playground projects with a bolt on TFT Gizmo that lets you add a lovely color display in a sturdy and reliable fashion. This PCB looks just like a round...
https://www.adafruit.com/product/4367
Setup For Compiling Examples

We're going to be using the popular Arduino IDE to compile and load code. Start by following the PyBadge setup guide to

1. Install the latest desktop Arduino IDE (https://adafruit.it/Fmm)
2. Install Adafruit SAMD board support package (https://adafruit.it/Fmn) (If programming a SAMD board like the Edge/PyBadge)
3. Install Adafruit nRF52 board support package (https://adafruit.it/Hwb) (If programming an nRF52 board like the Circuit Playground Bluefruit)
4. Install all the Arcada Libraries (yes there's a lot of them!) (https://adafruit.it/EUk)

TensorFlow Libraries

Now install the [Arduino TensorFlow library 1.15.0-ALPHA](https://adafruit.it/EUk) with the library manager

Make sure you don't pick the pre-compiled release version

If you see 'precompiled' in the name, install the non-precompiled version from the dropdown

**You must use 1.15-0-ALPHA (not-precompiled) - no other version will work!**

Next, install Adafruit TensorFlow Lite

And finally, for the speech demos, grab the Adafruit Zero PDM library
Select Board

Almost ready! Before we're ready to compile some examples!

Plug in the board into your computer with a known-good data/sync cable. Select the right board in the Tools download

For PyBadge/EdgeBadge

For some examples you will want to set Optimize to Fastest and set CPU Speed to 180MHz (overclocking). This will give 6-10x speedup. For the first few examples, it isn't necessary. Make sure to select USB Stack: TinyUSB

For Circuit Playground Bluefruit

You can use the default settings:
Sine Wave Demo

This is the "hello world" demo of TensorFlow Lite. It has a simple model that has been trained to generate a sine wave when a linear input is given. It's a good way to verify you have a working toolchain!

If you want to load demo this immediately to your Circuit Playground Bluefruit with TFT Gizmo, here is the UF2 file which you can 'drag-n-drop' onto your BOOT diskdrive to load the example (follow the instructions here on how to load UF2 files if you've never done it before (https://adafruit.it/GA4))

CPB_TF_SineWave_Gizmo.UF2
https://adafruit.it/Hwc

Serial plotter sine wave demo compile & upload

Let's start with the plain Arduino TensorFlow demo. Don't forget you have to perform all the steps in the previous page for installing Arduino IDE, Adafruit nRF52 support, libraries, and board/port selection!
Compile & upload this example!

Upon success, you may see the LED on the board pulsing. The best way to see the output is to select the Serial Plotter

You'll see a sine wave on the plotter!

If you want to see a more sinusoidal output go to arduino_constants.cpp
and change

```c
const int kInferencesPerCycle = 1000;
```

to

```c
const int kInferencesPerCycle = 200;
```

Then re-upload

![Waveform Graph]

**Arcada display output sine demo compile & upload**

Arcada is our library for handling displays and input - we have so many different boards and displays, we need a unifying library that would handle displays, filesystems, buttons, etc. For many boards, you don't need to do anything special to figure out the pinouts or part numbers!

Load up the Adafruit_TFLite->hello_world_arcada example

You can upload this sketch to your board and you'll get an animated wave on the screen.
The majority of the work is in this file that initializes the display on the first inference, then draws a ball on every successful inference. The curve of the ball creates the sine wave!

```c
/* Copyright 2019 The TensorFlow Authors. All Rights Reserved.
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
==============================================================================*/

#include "output_handler.h"
#include "Arduino.h"
#include "Adafruit_Arcada.h"
extern Adafruit_Arcada arcada;

// The pin of the Arduino's built-in LED
int led = LED_BUILTIN;

// Track whether the function has run at least once
bool initialized = false;

// helper function to let us scale floating point values nicely
double mapf(double x, double in_min, double in_max, double out_min, double out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}

// Animates a dot across the screen to represent the current x and y values
void HandleOutput(tflite::ErrorReporter* error_reporter, float x_value, float y_value) {
  // Do this only once
  if (!initialized) {
    // Add some text to describe what's up!
    arcada.display->fillScreen(ARCADA_BLACK);
    arcada.display->setTextColor(ARCADA_WHITE);
    arcada.display->setTextSize(1);
    const char *header = "TensorFlow Lite";
    const char *footer = "Sine wave model";
    arcada.display->setCursor((arcada.display->width()-strlen(header)*6)/2, 0);
    arcada.display->print(header);
    arcada.display->setCursor((arcada.display->width()-strlen/footer)*6)/2,
    arcada.display->height()-8);
    arcada.display->print(footer);
    initialized = true;
  }

  // map the x input value (0-2*PI) and the y value (-1.5 to 1.5)
  // to the size of the display
  float pixel_x, pixel_y;
  static float last_pixel_x, last_pixel_y;
  pixel x = mapf(x_value, 0, 2*3.1415, 0, arcada.display->width());
  pixel y = mapf(y_value, -1.75, 1.75, 0, arcada.display->height());
  if (pixel x == 0) {
    // clear the screen
    arcada.display->fillRect(0, 10, arcada.display->width(), arcada.display->width()-20, ARCADA_BLACK);
  }
```
Customized Wave Demo

The sine wave demo is great to do initial experimentation with training new simple single input->output models.

Google TensorFlow has a great guide here

The detailed part of the tutorial is in this colab script. Colab is great because its fully hosted, runs in any web-browser without using your CPU to do the training!

Re-creating the Default Sine Wave Model

Visit the colab script here:

And run all the script!

It may take a few minutes. When its complete you'll get an array of data at the bottom:
Grab that text starting with

```
unsigned char sine_model_quantized_tflite[] = {
```

and ending with

```
unsigned int sine_model_quantized_tflite_len = 2640;
```

(the number may vary)

```
unsigned char sine_model_quantized_tflite[] = {
  0x18, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0xe0, 0x00,
  0x18, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x14, 0x00,
  0x0e, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x10, 0x0a, 0x00, 0x00,
  0xb8, 0x05, 0x00, 0x00, 0xa0, 0x05, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00,
  ...more here....
  0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x0a, 0x00, 0x0c, 0x00, 0x07, 0x00, 0x00, 0x00, 0x08, 0x00,
  0x0a, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x09, 0x03, 0x00, 0x00, 0x00
};
unsigned int sine_model_quantized_tflite_len = 2640;
```

Now visit the hello_world_arcada sketch and fine the sine_model_data tab:

And paste that output from the notebook, replacing the

```
unsigned char sine_model_quantized_tflite[] = {
```

down to

```
const int g_sine_model_data_len = 2640;
```
Recompile and upload to your badge or Circuit Playground Bluefruit, you should get the exact same sine wave demo!

Before you can compile, don't forget to change the declaration of the array from 'unsigned char sine_model_quantized_tflite' to 'unsigned const char g_sine_model_data' and the length variable from 'unsigned int sine_model_quantized_tflite_len' to 'const int g_sine_model_data_len'

OK maybe not so exciting. Let's try changing the model.

Creating a Cosine Wave Model

Let's get wild and crazy now, by making a cosine wave model. Find the line in the script where we create the `y_values` from the `x_values`
And change it to a cos function!

```python
# Calculate the corresponding sine values
y_values = np.cos(x_values)

# Plot our data. The 'b.' argument tells the library to print blue dots.
plt.plot(x_values, y_values, 'b.')</plot
plt.show()
```

Run the colab script from this point down (or the whole thing) to get a brand new `unsigned char sine_model_quantized_tflite` array and follow the steps you did before to replace the model array in the hello_world sketch with your new model. Re-upload to now get cosine wave output, which looks like the plot above.
## Loading Models From Internal Storage

The PyBadge/EdgeBadge/Circuit Playground Bluefruit have 2MB of internal storage. We can use that to store TensorFlow models, so that we don't have to go through the recompilation step above. Instead, the model is loaded from that storage flash, we can read/write to the flash over USB by dragging-and-dropping, just like a USB key!

Upload the `hello_world_arcada` sketch. If you have a PyBadge/EdgeBadge, this time make sure to select TinyUSB as the USB stack since that will activate mass storage support. You don't have to specifically select TinyUSB for nRF52 (e.g. Circuit Playground Bluefruit).

Now when you upload, on reset you'll get a new disk drive appearing on your computer, it'll probably be name `CIRCUITPY` but you can rename it if you like.

Download this models zip file and drag the two `.tflite` files to `CIRCUITPY`
Rename one of the files to model.tflite - when this file is available to read, this will tell our sketch to load the model from disk instead of from memory!

Once you've renamed the file, click the reset button, you should get an alert like this:

Once you've renamed the file, click the reset button, you should get an alert like this. Press the A button on the badge or Circuit Playground Bluefruit (its the left button) to begin the model inference run.

Once you've proved that you're running one of the files, try renaming the other model file to model.tflite and reset. That way you can prove that its running from the disk.

You can go back to the colab script you ran, and look in the Files tab to find the `sine_model_quantized.tflite` file that is converted in the last stage. You can download that file directly.

Try creating new tflite files that model different functions!

---

**Gesture Demo**

The PyBadge has a built-in accelerometer (LIS3DH) which you can use to detect tilt and motion. The accelerometer outputs 3 axes of acceleration data, and we can use that to train and infer gestures using TensorFlow!

If you want to load this demo immediately to your CircuitPlayground Bluefruit with TFT Gizmo here is a UF2 file, which you can 'drag-n-drop' onto your CPLAYBOOT
diskdrive to load the example (follow the instructions here on how to load UF2 files if you've never done it before (https://adafruit.it/GuD))

CPB_Gizmo_Magic_Wand.UF2
https://adafruit.it/Hwd

Serial out gesture demo compile & upload

Let's start with the plain Arduino TensorFlow demo. Don't forget you have to perform all the steps in the previous page for installing Arduino IDE, Adafruit nRF52 support, libraries, and board/port selection!

We adapted the default gesture demo to use the LIS3DH, so you cannot use the example in the Arduino TensorFlowLite library Instead, use the one in Adafruit TensorFlow Lite called magic_wand

Compile & upload this example!

Upon success, you may see the LED on the board pulsing. The best way to see the output is to select the Serial Monitor
You'll see steaming data coming out on the Serial Monitor. This is the 3 axis accelerometer data. We output it so that you can have some more debugging data which can be really handy when training/debugging gestures. You can also plot it with the Serial Plotter if you like (close the Monitor first)

Move and twist the badge to see the red/green/blue lines change.

Close the Plotter and re-open the monitor to see the streaming data again. This time, With the Playground NEOPIXEL Ring facing you, and the USB port pointing to the ceiling perform one of three gestures:
Wing

This gesture is a W starting at your top left, going down, up, down up to your top right

When that gesture is detected you'll see the front NeoPixels turn yellow, and the following print out on the Serial Monitor:

```
Unlike the later TFT demo, this demo needs to have the neopixels facing you, the orientation matters!
```

Ring

This gesture is a O starting at top center, then moving clockwise in a circle to the right, then down, then left and back to when you started in the top center

When that gesture is detected you'll see the front NeoPixels turn purple, and the following print out on the Serial Monitor:
Slope

This gesture is an L starting at your top right, moving diagonally to your bottom left, then straight across to bottom right.

When that gesture is detected you'll see the front NeoPixels turn light blue, and the following print out on the Serial Monitor:

Arcada display output gesture demo compile & upload

Arcada is our library for handling displays and input - we have so many different boards and displays, we need a unifying library that would handle displays, filesystems, buttons, etc. For many boards, you don't need to do anything special to figure out the pinouts or part numbers!

Load up the Adafruit_TFLite->magic_wand_arcada example
You can upload this sketch to your board. After upload it will show up on your computer as a disk drive called CIRCUITPY (unless you changed it)

Click this button to download the gesture images and audio clips

Circuit Playground Bluefruit Gesture Images

https://adafruit.it/Hwe

Navigate through the zip file to examples\magic_wand_arcada\cplay_files then drag the files directly onto the CIRCUITPY drive like so:

Click reset on the Bluefruit to restart, and you should get the graphics displaying so that you can run the demo untethered!
Setup and configuration of the accelerometer and screen is done in the accelerometer_handler

```c
/* Copyright 2019 The TensorFlow Authors. All Rights Reserved. 
Licensed under the Apache License, Version 2.0 (the "License"); 
you may not use this file except in compliance with the License. 
You may obtain a copy of the License at 
http://www.apache.org/licenses/LICENSE-2.0 

Unless required by applicable law or agreed to in writing, software 
distributed under the License is distributed on an "AS IS" BASIS, 
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. 
See the License for the specific language governing permissions and 
limitations under the License. */

#include "accelerometer_handler.h"
#include <Arduino.h>
#include "Adafruit_Arcada.h"
extern Adafruit_Arcada arcada;

/* this is a little annoying to figure out, as a tip - when 
* holding the board straight, output should be (0, 0, 1) 
* tilting the board 90* left, output should be (0, 1, 0) 
* tilting the board 90* forward, output should be (1, 0, 0); 
*/

#if defined(ADAFRUIT_PYBADGE_M4_EXPRESS)
// holding up with screen/neopixels facing you 
const int X_POSITION = 1;
const int Y_POSITION = 2;
const int Z_POSITION = 0;
const bool INVERT_X = true;
const bool INVERT_Y = true;
const bool INVERT_Z = false;
#endif

#if defined(ARDUINO_NRF52840_CIRCUITPLAY)
```

©Adafruit Industries
const int X_POSITION = 1;
const int Y_POSITION = 2;
const int Z_POSITION = 0;
const bool INVERT_X = true;
const bool INVERT_Y = true;
const bool INVERT_Z = false;

#include "constants.h"

// A buffer holding the last 200 sets of 3-channel values
float save_data[600] = {0.0};
// Most recent position in the save_data buffer
int begin_index = 0;
// True if there is not yet enough data to run inference
bool pending_initial_data = true;
// How often we should save a measurement during downsampling
int sample_every_n = 1;
// The number of measurements since we last saved one
int sample_skip_counter = 0;

uint32_t last_reading_stamp = 0;

TfLiteStatus SetupAccelerometer(tflite::ErrorReporter* error_reporter) {
    // Wait until we know the serial port is ready
    //while (!Serial) { yield(); }

    arcada.pixels.setBrightness(50); // Set BRIGHTNESS to about 1/5 (max = 255)

    arcada.accel->setRange(LIS3DH_RANGE_4_G);
    arcada.accel->setDataRate(LIS3DH_DATARATE_25_HZ);
    float sample_rate = 25;

    // Determine how many measurements to keep in order to meet kTargetHz
    sample_every_n = static_cast<int>(roundf(sample_rate / kTargetHz));

    error_reporter->Report("Magic starts!");

    return kTfLiteOk;
}

bool ReadAccelerometer(tflite::ErrorReporter* error_reporter, float* input,
        int length, bool reset_buffer) {
    // Clear the buffer if required, e.g. after a successful prediction
    if (reset_buffer) {
        memset(save_data, 0, 600 * sizeof(float));
        begin_index = 0;
        pending_initial_data = true;
    }
    // Keep track of whether we stored any new data
    bool new_data = false;
    // Loop through new samples and add to buffer
    while (arcada.accel->haveNewData()) {
        float x, y, z;
        // Read each sample, removing it from the device's FIFO buffer
        sensors_event_t event;
if (!arcada.accel->getEvent(&event)) {
    error_reporter->Report("Failed to read data");
    break;
}

// Throw away this sample unless it's the nth
if (sample_skip_counter != sample_every_n) {
    sample_skip_counter += 1;
    continue;
}

float values[3] = {0, 0, 0};
values[X_POSITION] = event.acceleration.x / 9.8;
values[Y_POSITION] = event.acceleration.y / 9.8;
values[Z_POSITION] = event.acceleration.z / 9.8;

x = values[0];
y = values[1];
z = values[2];

if (INVERT_X) {
    x *= -1;
}
if (INVERT_Y) {
    y *= -1;
}
if (INVERT_Z) {
    z *= -1;
}
Serial.print(x, 2);
Serial.print(","); Serial.print(y, 2);
Serial.print(","); Serial.println(z, 2);

last_reading_stamp = millis();
// Write samples to our buffer, converting to milli-Gs
save_data[begin_index++] = x * 1000;
save_data[begin_index++] = y * 1000;
save_data[begin_index++] = z * 1000;

// Since we took a sample, reset the skip counter
sample_skip_counter = 1;
// If we reached the end of the circle buffer, reset
if (begin_index >= 600) {
    begin_index = 0;
}
new_data = true;

// Skip this round if data is not ready yet
if (!new_data) {
    return false;
}

// Check if we are ready for prediction or still pending more initial data
if (pending_initial_data && begin_index >= 200) {
    pending_initial_data = false;
}

// Return if we don't have enough data
if (pending_initial_data) {
    return false;
}

// Copy the requested number of bytes to the provided input tensor
for (int i = 0; i < length; ++i) {
    int ring_array_index = begin_index + i - length;
    if (ring_array_index < 0) {
        ring_array_index += 600;
```cpp
While the LED/Display output is done in the output_handler.cpp

/* Copyright 2019 The TensorFlow Authors. All Rights Reserved.
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
==============================================================================*/

#include "output_handler.h"
#include "Arduino.h"
#include "Adafruit_Arcada.h"

extern Adafruit_Arcada arcada;

void HandleOutput(tflite::ErrorReporter* error_reporter, int kind) {
  // The first time this method runs, set up our LED
  static int last_kind = -1;

  static bool is_initialized = false;
  if (!is_initialized) {
    pinMode(LED_BUILTIN, OUTPUT);
    is_initialized = true;
  }
  // Toggle the LED every time an inference is performed
  static int count = 0;
  ++count;
  if (count & 1) {
    digitalWrite(LED_BUILTIN, HIGH);
  } else {
    digitalWrite(LED_BUILTIN, LOW);
  }

  // Print some ASCII art for each gesture
  if (kind == 0) {
    error_reporter->Report("WING:
*         *         *
 *       * *       *
  *     *   *     *
   *   *     *   *
    * *       *
<br>
*
* *
* *
* *
* *
* *
* *
* *
WING
   ");
  ImageReturnCode stat = arcada.drawBMP((char *)"wing.bmp", 0, 0);
  if (stat != IMAGE_SUCCESS) {
    arcada.display->fillScreen(ARCADA_BLACK);
    arcada.display->setCursor(20, 20);
    arcada.display->setTextColor(ARCADA_YELLOW);
    arcada.display->setTextSize(ceil(arcada.display->width() / 30));
    arcada.display->print("WING");
  }
  arcada.WavPlayComplete("wing.wav");
  arcada.pixels.fill(arcada.pixels.Color(50, 50, 0));
}
```

©Adafruit Industries
Micro Speech Demo

The Circuit Playground Bluefruit has a built-in microphone which you can use to detect audio and speech. The mic outputs monophonic digital sound waves, and we can use that to train and infer gestures using TensorFlow!

If you want to load this demo immediately to your CPB, here is a UF2 file, which you can 'drag-n-drop' onto your CPLAYBOOT diskdrive to load the example (follow the
instructions here on how to load UF2 files if you've never done it before (https://adafru.it/GuD)

Circuit_Playground_Bluefruit_YesNo.UF2
https://adafru.it/HCa

Micro speech demo compile & upload

Don't forget you have to perform all the steps in the previous page for installing Arduino IDE, Adafruit nRF52 support, libraries, and board/port selection!

We adapted the default speech demo to use various kinds of audio input, so you can not use the example in the Arduino TensorFlowLite library Instead, use the one in Adafruit TensorFlow Lite called micro_speech_arcada

You can upload this sketch to your board. After upload it will show up on your computer as a disk drive called CIRCUITPY (unless you changed it)

Click this button to download the detection/info images

Circuit Playground MicroSpeech files
https://adafru.it/HCb

©Adafruit Industries
Navigate through the zip file to examples\micro_speech_arcada\gizmo_files then drag the files directly onto the CIRCUITPY drive like so:

Click reset on the Playground to restart, and you should get the graphics displaying so that you can run the demo untethered!

Setup and configuration of the microphone and screen is done in the audio_provider

/* Copyright 2018 The TensorFlow Authors. All Rights Reserved.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

==============================================================================*/

/* Copyright 2018 The TensorFlow Authors. All Rights Reserved. */
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

#include "audio_provider.h"
#include "micro_features_micro_model_settings.h"
#include <Adafruit_Arcada.h>

#if defined(__SAMD51__)
#define USE_EXTERNAL_MIC A8  // D2 on pybadge
#define USE_EDGEBADGE_PDMMIC
#define AUDIO_OUT A0         // uncomment to 'echo' audio to A0 for debugging
#define DEFAULT_BUFFER_SIZE 512
#endif

#include <Adafruit_ZeroPDMSPI.h>
Adafruit_ZeroPDMSPI pdmspi(
&
PDM_SPI);

#endif
#if defined(ARDUINO_NRF52840_CIRCUITPLAY)
#include <PDM.h>
define DEFAULT_BUFFER_SIZE DEFAULT_PDM_BUFFER_SIZE
extern PDMClass PDM;
static volatile int samplesRead; // number of samples read
#endif

extern Adafruit_Arcada arcada;

void CaptureSamples();

namespace {

bool g_is_audio_initialized = false;
const expr int kAudioCaptureBufferSize = DEFAULT_BUFFER_SIZE * 16;
int16_t g_audio_capture_buffer[kAudioCaptureBufferSize];
// A buffer that holds our output
int16_t g_audio_output_buffer[kMaxAudioSampleSize];
// Mark as volatile so we can check in a while loop to see if
// any samples have arrived yet.
volatile int32_t g_latest_audio_timestamp = 0;
// Our callback buffer for collecting a chunk of data
volatile int16_t recording_buffer[DEFAULT_BUFFER_SIZE];

volatile int max_audio = -32768, min_audio = 32768;
} // namespace

#if defined(ARDUINO_NRF52840_CIRCUITPLAY)
// IRQ handler
static void onPDMdata() {
// query the number of bytes available
int bytesAvailable = PDM.available();

// read into the sample buffer
PDM.read((int16_t *)recording_buffer, bytesAvailable);

// 16-bit, 2 bytes per sample
samplesRead = bytesAvailable / 2;
}
```
void TIMER_CALLBACK() {
    static bool ledtoggle = false;
    static uint32_t audio_idx = 0;
    int32_t sample = 0;

#if defined(USE_EDGEBADGE_PDMMIC)
    uint16_t read_pdm;
    if ( !pdmspi.decimateFilterWord( &read_pdm )) {
        return; // not ready for data yet!
    }
    sample = read_pdm;
#endif

    // tick tock test
    //digitalWrite(LED_BUILTIN, ledtoggle);
    //ledtoggle = !ledtoggle;

#if defined(ARDUINO_NRF52840_CIRCUITPLAY)
PDM_IRQHandler(); // wait for samples to be read
if ( samplesRead ) {
    max_audio = -32768, min_audio = 32768;
    for ( int i=0; i<samplesRead; i++ ) {
        max_audio = max(recording_buffer[i], max_audio);
        min_audio = min(recording_buffer[i], min_audio);
    }
    CaptureSamples();
    samplesRead = 0;
}
    // we did a whole buffer at once, so we're done
    return;
#endif

    if ( audio_idx >= DEFAULT_BUFFER_SIZE ) {
        CaptureSamples();
        max_audio = -32768, min_audio = 32768;
        audio_idx = 0;
    }

#if defined(USE_EXTERNAL_MIC)
sample = analogRead(USE_EXTERNAL_MIC);
    sample -= 2047; // 12 bit audio unsigned 0-4095 to signed -2048--2047
#endif
#if defined(USE_EDGEBADGE_PDMMIC)
sample -= 32676; // from 0-65535 to -32768 to 32768
#endif
#if defined(AUDIO_OUT)
analogWrite(AUDIO_OUT, sample+2048);
#endif

    recording_buffer[ audio_idx ] = sample;
    max_audio = max( max_audio, sample );
    min_audio = min( min_audio, sample );
    audio_idx++;
}

TfLiteStatus InitAudioRecording(tflite::ErrorReporter* error_reporter) {
    Serial.begin(115200);
    Serial.println("init audio"); delay(10);
    // Hook up the callback that will be called with each sample
    #if defined(USE_EXTERNAL_MIC)
        arcada.timerCallback( kAudioSampleFrequency, TIMER_CALLBACK );
    analogReadResolution(12);
    #endif
    #if defined(USE_EDGEBADGE_PDMMIC)
        pdmspi.begin( kAudioSampleFrequency );
    #endif
}
Serial.print("Final PDM frequency: "); Serial.println(pdmspi.sampleRate);
#endif
#if defined(AUDIO_OUT)
analogWriteResolution(12);
#endif
#if defined(ARDUINO_NRF52840_CIRCUITPLAY)
PDM.onReceive(onPDMdata);
if (!PDM.begin(1, 16000)) {
  Serial.println("Failed to start PDM!");
  while (1) yield();
}
arcada.timerCallback(kAudioSampleFrequency, TIMER_CALLBACK);
#endif

// Block until we have our first audio sample
while (!g_latest_audio_timestamp) {
  delay(1);
}
return kTfLiteOk;
}

void CaptureSamples() {
  // This is how many bytes of new data we have each time this is called
  const int number_of_samples = DEFAULT_BUFFER_SIZE;
  // Calculate what timestamp the last audio sample represents
  const int32_t time_in_ms =
    g_latest_audio_timestamp +
    (number_of_samples / (kAudioSampleFrequency / 1000));
  // Determine the index, in the history of all samples, of the last sample
  const int32_t start_sample_offset =
    (g_latest_audio_timestamp * (kAudioSampleFrequency / 1000));
  // Determine the index of this sample in our ring buffer
  const int capture_index = start_sample_offset % kAudioCaptureBufferSize;
  // Read the data to the correct place in our buffer, note 2 bytes per buffer entry
  memcpy(g_audio_capture_buffer + capture_index, (void*)recording_buffer, DEFAULT_BUFFER_SIZE*2);
  // This is how we let the outside world know that new audio data has arrived.
  g_latest_audio_timestamp = time_in_ms;

  int peak = (max_audio - min_audio);
  Serial.printf("pp %d\n", peak);
  //int normalized = map(peak, 20, 2000, 0, 65535);
  //arcada.pixels.setPixelColor(0, arcada.pixels.gamma32(arcada.pixels.ColorHSV(normalized)));
  //arcada.pixels.show();
}

TfLiteStatus GetAudioSamples(tflite::ErrorReporter* error_reporter, int start_ms, int duration_ms, int* audio_samples_size, int16_t** audio_samples) {
  // Set everything up to start receiving audio
  if (!g_is_audio_initialized) {
    TfLiteStatus init_status = InitAudioRecording(error_reporter);
    if (init_status != kTfLiteOk) {
      return init_status;
    }
    g_is_audio_initialized = true;
  }
  // This next part should only be called when the main thread notices that the
  // latest audio sample data timestamp has changed, so that there's new data
  // in the capture ring buffer. The ring buffer will eventually wrap around and
  // overwrite the data, but the assumption is that the main thread is checking
  // often enough and the buffer is large enough that this call will be made
  // before that happens.
  // Determine the index, in the history of all samples, of the first
  // sample we want
  const int start_offset = start_ms * (kAudioSampleFrequency / 1000);
// Determine how many samples we want in total
const int duration_sample_count =
    duration_ms * (kAudioSampleFrequency / 1000);
for (int i = 0; i < duration_sample_count; ++i) {
    // For each sample, transform its index in the history of all samples into
    // its index in g_audio_capture_buffer
    const int capture_index = (start_offset + i) % kAudioCaptureBufferSize;
    // Write the sample to the output buffer
    g_audio_output_buffer[i] = g_audio_capture_buffer[capture_index];
}

// Set pointers to provide access to the audio
*audio_samples_size = kMaxAudioSampleSize;
*audio_samples = g_audio_output_buffer;
return kTfLiteOk;
}

int32_t LatestAudioTimestamp() { return g_latest_audio_timestamp; }

While the LED/Display/audio output is done in the command_responder.cpp

void RespondToCommand(tflite::ErrorReporter* error_reporter,
    int32_t current_time, const char* found_command, uint8_t score, bool is_new_command) {
    static bool is_initialized = false;
    if (!is_initialized) {
        pinMode(LED_BUILTIN, OUTPUT);
        // Pins for the built-in RGB LEDs on the Arduino Nano 33 BLE Sense
        is_initialized = true;
    }
    static int32_t last_command_time = 0;
    static int count = 0;
    static int certainty = 220;
    if (is_new_command) {
        error_reporter->Report("Heard %s (%d) @%dms", found_command, score, current_time);
        if (found_command[0] == 'y') {
            last_command_time = current_time;
            ImageReturnCode stat = arcada.drawBMP((char*)"yes.bmp", 0, 0);
            if (stat != IMAGE_SUCCESS) {
                // Toggle the LED
                digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
            } else {
                // Draw the BMP image
                ImageReturnCode stat = arcada.drawBMP((char*)"yes.bmp", 0, 0);
                if (stat != IMAGE_SUCCESS) {
                    // Toggle the LED
                    digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
                }
            }
        } else {
            // Draw the BMP image
            ImageReturnCode stat = arcada.drawBMP((char*)"yes.bmp", 0, 0);
            if (stat != IMAGE_SUCCESS) {
                // Toggle the LED
                digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
            }
        }
    }
}
arcada.display->fillScreen(ARCADA_BLACK);
arcada.display->setCursor(20, 20);
arcada.display->setTextColor(ARCADA_GREEN);
arcada.display->setTextSize(ceil(arcada.display->width() / 30));
arcada.display->print("YES");
}
arcada.WavPlayComplete("yes.wav");
arcada.pixels.fill(arcada.pixels.Color(0, 50, 0));
arcdar.pixels.show();
}  

if (found_command[0] == 'n') {
    last_command_time = current_time;
    ImageReturnCode stat = arcada.drawBMP((char *)"no.bmp", 0, 0);
    if (stat != IMAGE_SUCCESS) {
        arcada.display->fillScreen(ARCADA_BLACK);
        arcada.display->setCursor(20, 20);
        arcada.display->setTextColor(ARCADA_RED);
        arcada.display->setTextSize(ceil(arcada.display->width() / 30));
        arcada.display->print("NO");
    }
    arcada.WavPlayComplete("no.wav");
arcdar.pixels.fill(arcada.pixels.Color(50, 0, 0));
arcdar.pixels.show();
}

if (found_command[0] == 'u') {
    last_command_time = current_time;
    ImageReturnCode stat = arcada.drawBMP((char *)"no.bmp", 0, 0);
    if (stat != IMAGE_SUCCESS) {
        arcada.display->fillScreen(ARCADA_BLACK);
        arcada.display->setCursor(20, 20);
        arcada.display->setTextColor(ARCADA_LIGHTGREY);
        arcada.display->setTextSize(ceil(arcada.display->width() / 30));
        arcada.display->print("???");
    }
    arcada.pixels.fill(arcada.pixels.Color(10, 10, 10));
arcdar.pixels.show();
}

// If last_command_time is non-zero but was 1 seconds ago, zero it
// and switch off the LED.
if (last_command_time != 0) {
    if (last_command_time < (current_time - 1000)) {
        last_command_time = 0;
        // draw intro
        ImageReturnCode stat = arcada.drawBMP((char *)"howto.bmp", 0, 0);
        if (stat != IMAGE_SUCCESS) {
            arcada.display->fillScreen(ARCADA_BLACK);
            arcada.display->setCursor(0, 0);
            arcada.display->setTextColor(ARCADA_WHITE);
            arcada.display->setTextSize(ceil(arcada.display->width() / 180.0));
            arcada.display->println("Hold microphone");
            arcada.display->println("approx. 6-8\" away");
            arcada.display->println("from mouth and say");
            arcada.display->println("either YES or NO");
        }
        arcada.pixels.fill(arcada.pixels.Color(0, 0, 0));
arcdar.pixels.show();
    }
    // If it is non-zero but <3 seconds ago, do nothing.
    return;
}

++count;
if (count & 1) {

// Otherwise, toggle the LED every time an inference is performed.
++count;
if (count & 1) {
digitalWrite(LED_BUILTIN, HIGH);
} else {
  digitalWrite(LED_BUILTIN, LOW);
}