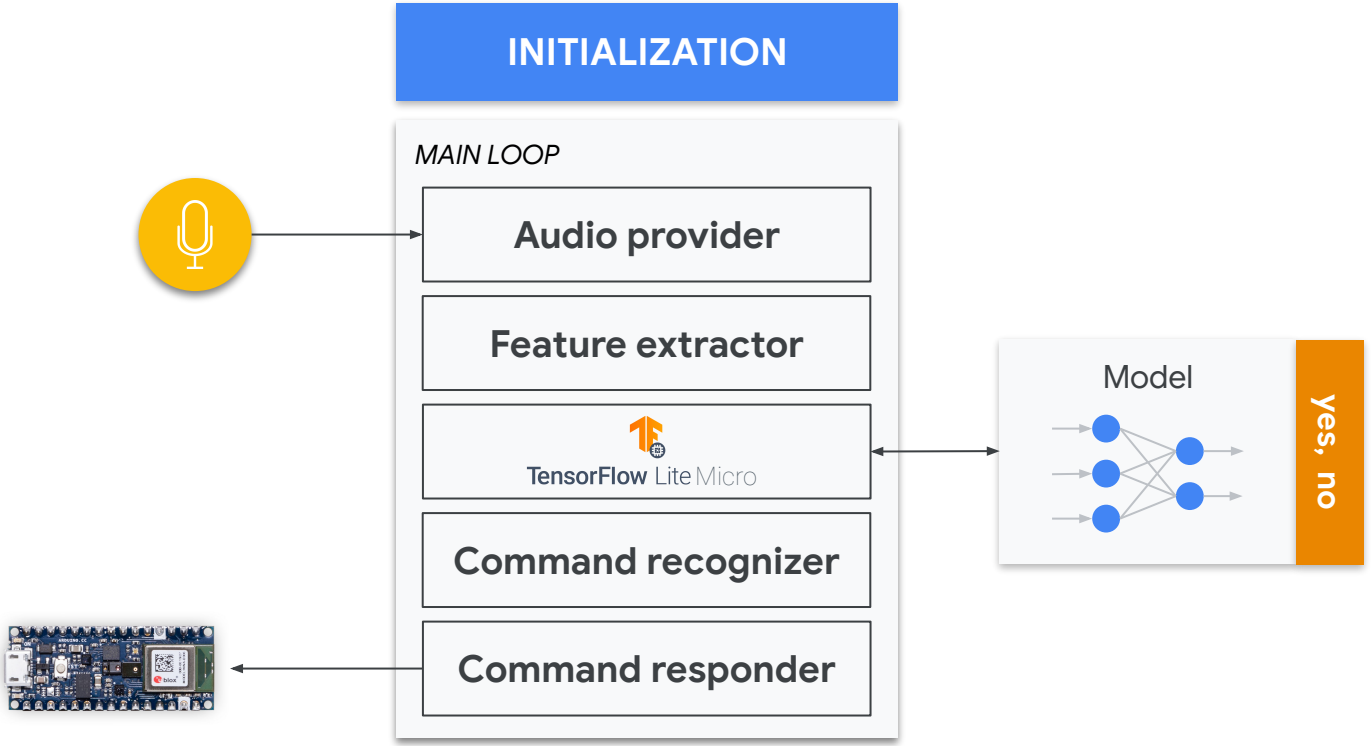
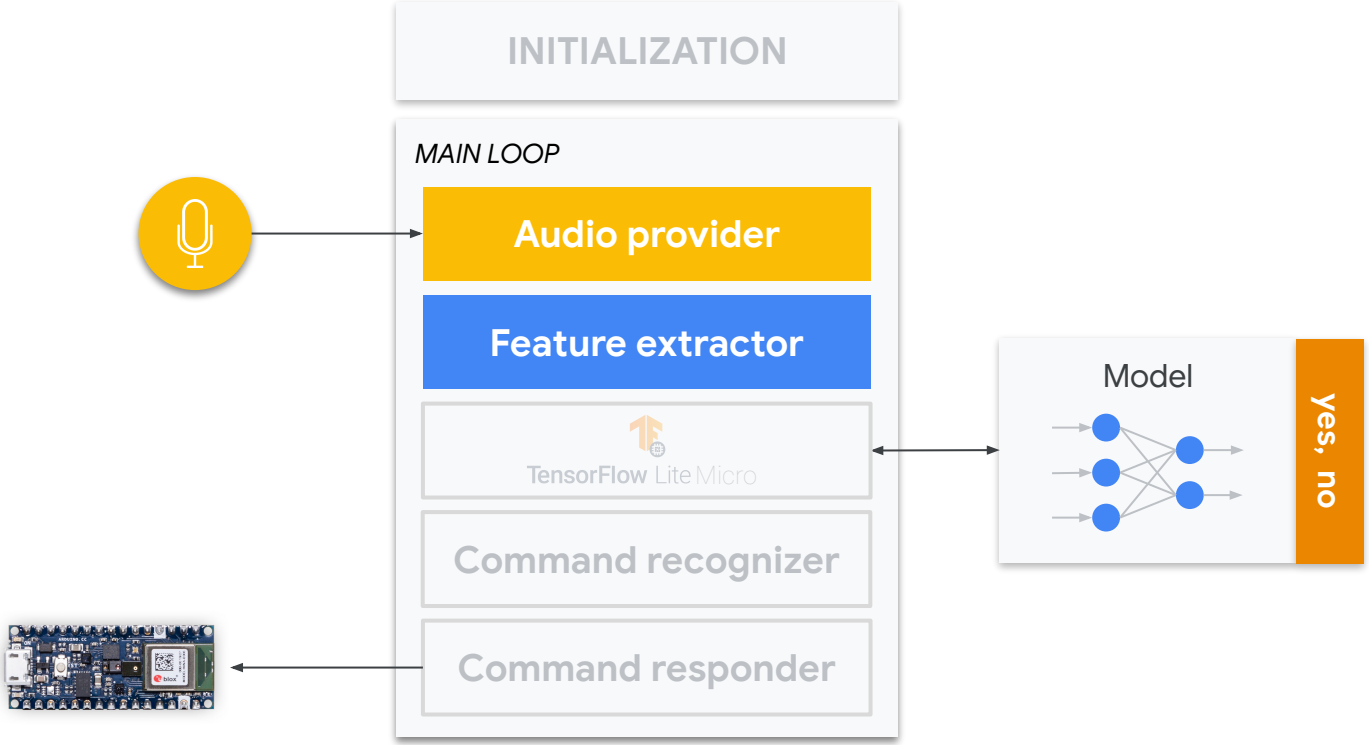
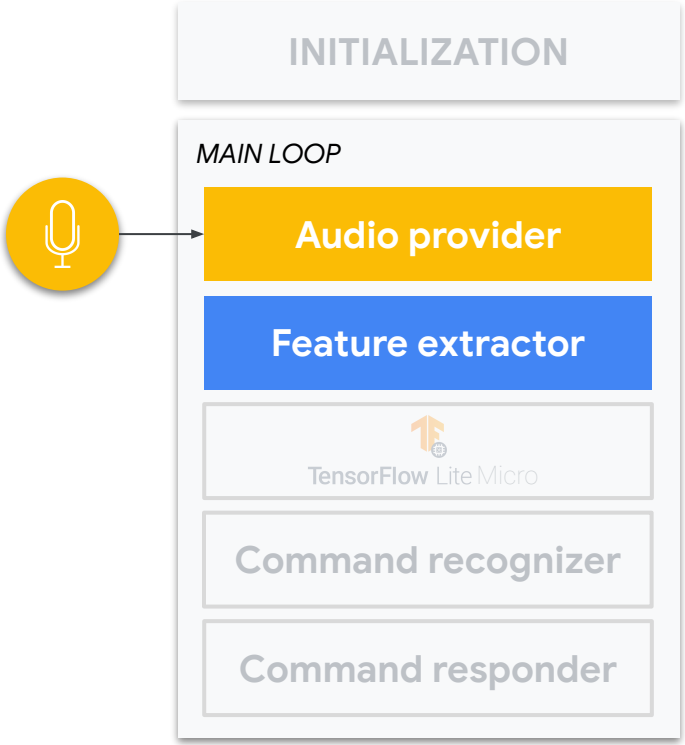
KWS Pre-Processing

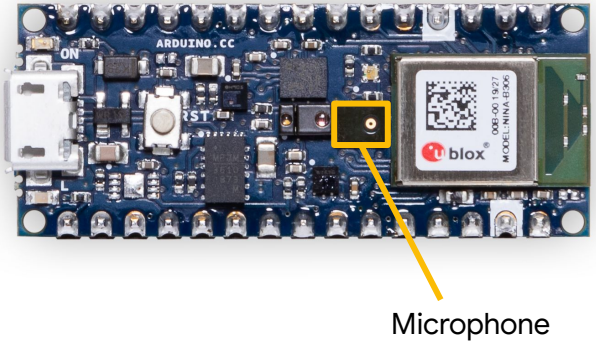
Keyword Spotting **Components**



Keyword Spotting **Components**




INITIALIZATION
MAIN LOOP
Audio provider
Feature extractor
TensorFlow Lite Micro
Command recognizer
Command responder

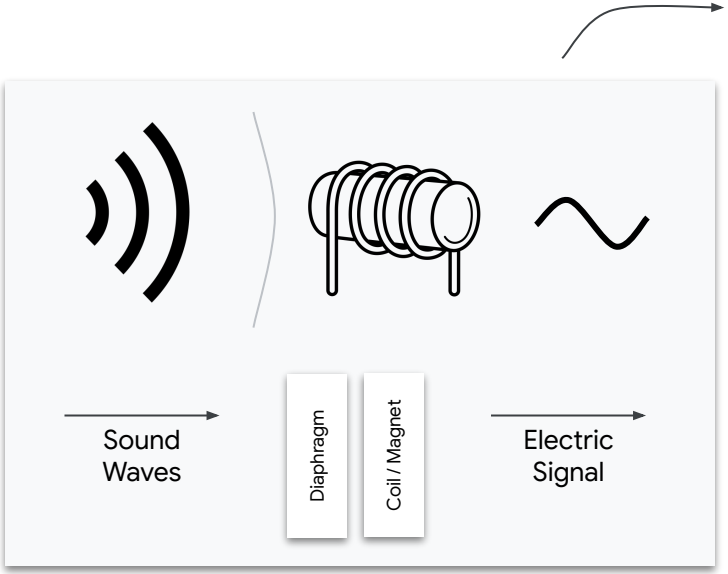

ARDUINO.CC
ON
RST
blox
Microphone

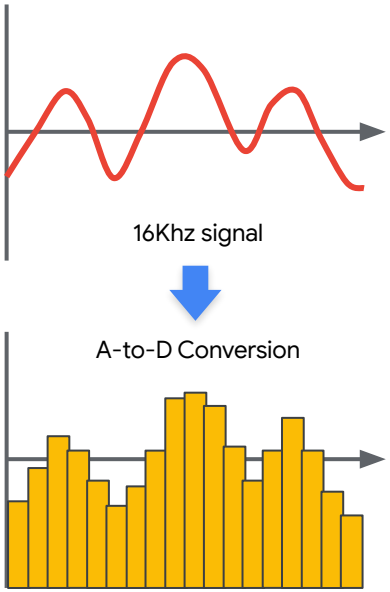
Audio provider

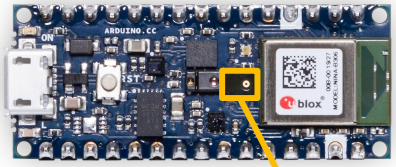
configure microphone

get audio samples

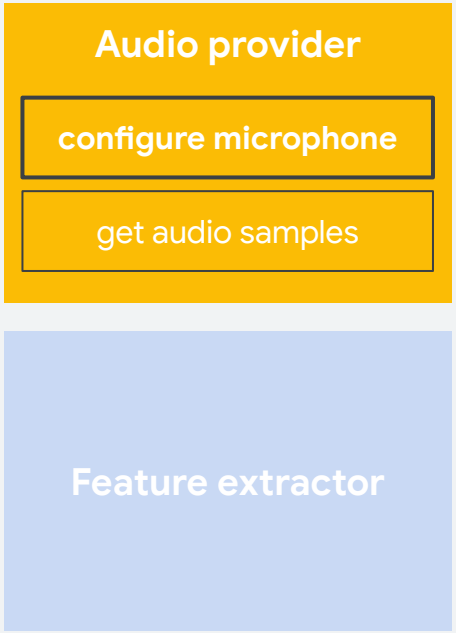
Feature extractor







Microphone




```
TfLiteStatus InitAudioRecording(tflite::ErrorReporter* error_reporter) {
  // Hook up the callback that will be called with each sample
  PDM.onReceive(CaptureSamples);

  // Start listening for audio: MONO @ 16KHz with gain at 20
  PDM.begin(1, kAudioSampleFrequency);
  PDM.setGain(20);

  // Block until we have our first audio sample
  while (!g_latest_audio_timestamp) {
  }

  return kTfLiteOk;
}
```

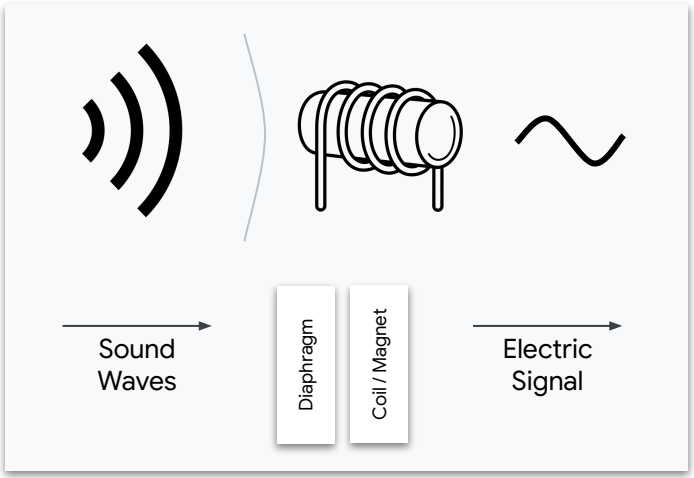
Audio provider

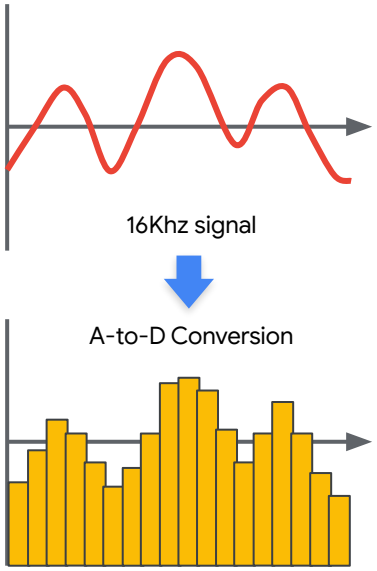
configure microphone

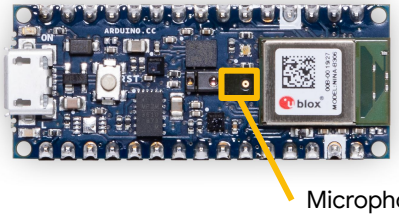
get audio samples

Feature extractor

REPEAT








```
TfLiteStatus InitAudioRecording(tflite::ErrorReporter* error_reporter)
```

0	1	2	3	...			N

Audio provider

configure microphone

get audio samples

Feature extractor

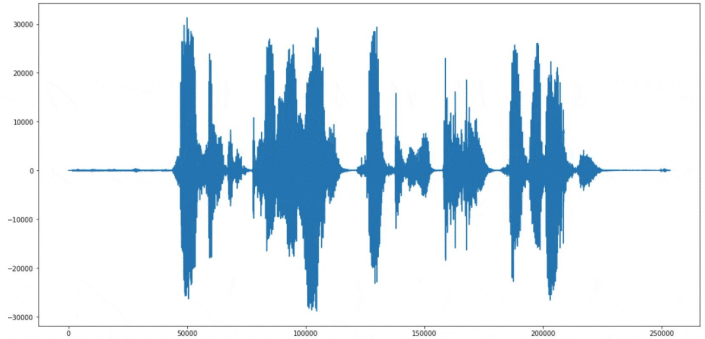
```
// This is an abstraction around an audio source like a microphone, and is
// expected to return 16-bit PCM sample data for a given point in time. The
// sample data itself should be used as quickly as possible by the caller, since
// to allow memory optimizations there are no guarantees that the samples won't
// be overwritten by new data in the future. In practice, implementations should
// ensure that there's a reasonable time allowed for clients to access the data
// before any reuse.

// The reference implementation can have no platform-specific dependencies, so
// it just returns an array filled with zeros. For real applications, you should
// ensure there's a specialized implementation that accesses hardware APIs.

TfLiteStatus GetAudioSamples(tflite::ErrorReporter* error_reporter,
                             int start_ms, int duration_ms,
                             int* audio_samples_size, int16_t** audio_samples);
```

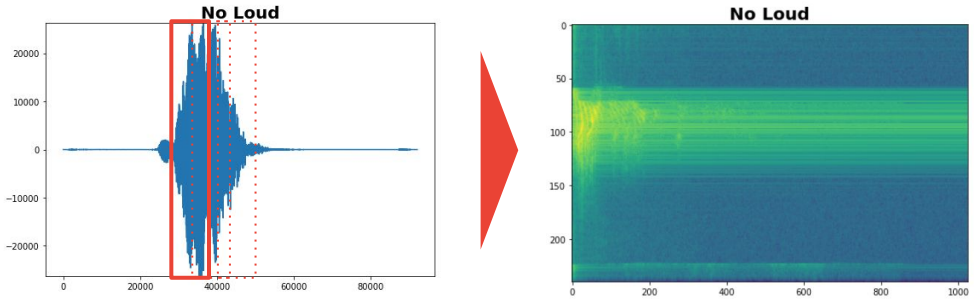
Audio provider

Feature extractor

gather audio data

select features



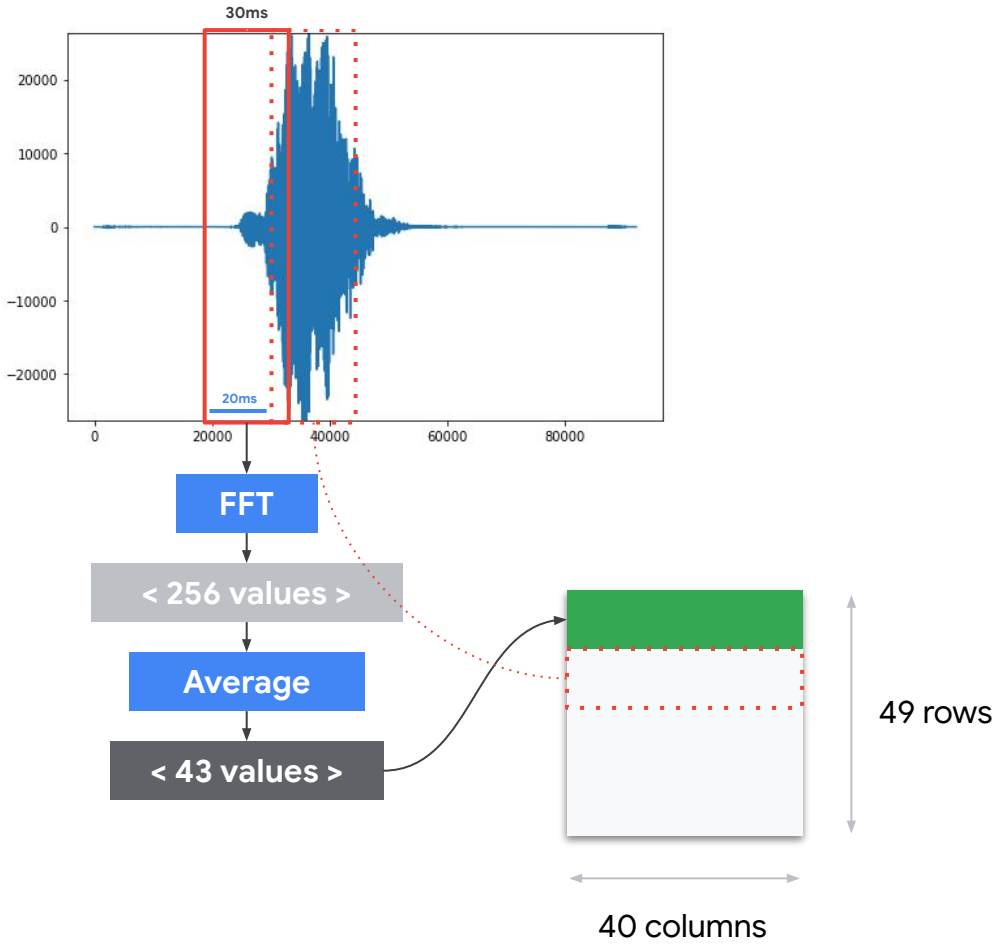


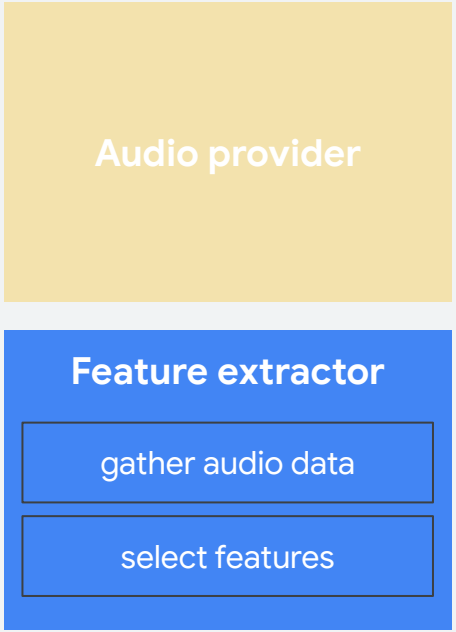
Audio provider

Feature extractor

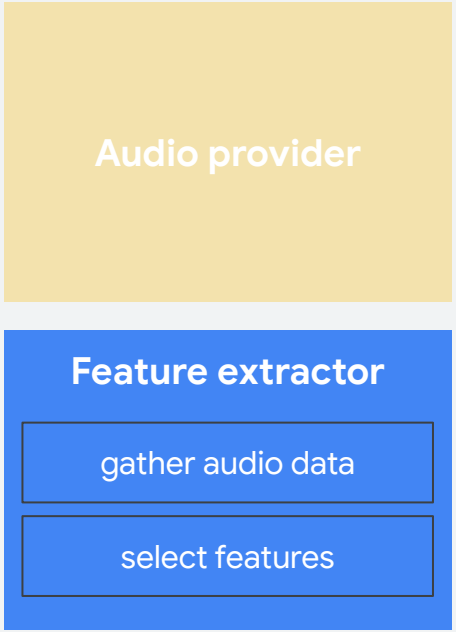
gather audio data

select features






```
void setup() {
// Prepare to access the audio spectrograms from a microphone
// or other source that will provide the inputs to the neural network.

static FeatureProvider static_feature_provider(kFeatureElementCount,
                                               feature_buffer);
...
}
```




```
void setup() {
// Prepare to access the audio spectrograms from a microphone
// or other source that will provide the inputs to the neural network.

static FeatureProvider static_feature_provider(kFeatureElementCount,
                                               feature_buffer);
...
}

// The name of this function is important for Arduino compatibility.
void loop() {
  // Fetch the spectrogram for the current time.

  const int32_t current_time = LatestAudioTimestamp();
  int how_many_new_slices = 0;
  TfLiteStatus feature_status = feature_provider->PopulateFeatureData(
      error_reporter, previous_time, current_time,&how_many_new_slices);
  if (feature_status != kTfLiteOk) {
    TF_LITE_REPORT_ERROR(error_reporter, "Feature generation failed");
    Return;
}
```

Audio provider

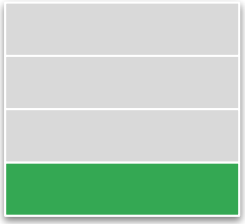
Feature extractor

gather audio data

select features

while buffer != FULL

Get audio samples



Audio provider

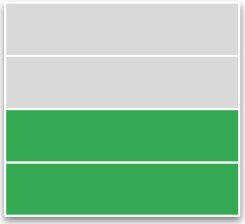
Feature extractor

gather audio data

select features

while buffer != FULL

Get audio samples



Audio provider

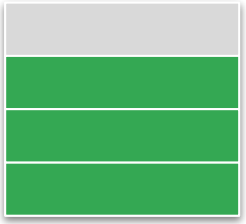
Feature extractor

gather audio data

select features

while buffer != FULL

Get audio samples



Audio provider

Feature extractor

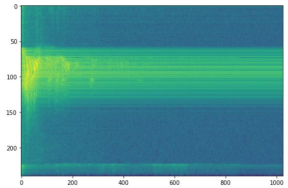
gather audio data

select features

while buffer != FULL

Get audio samples

extract features

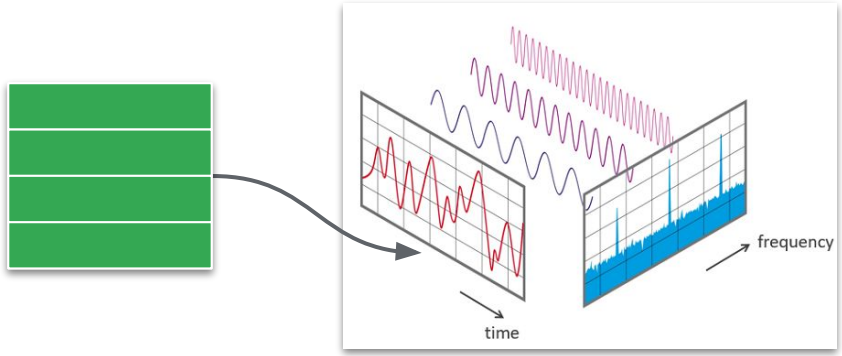


Audio provider

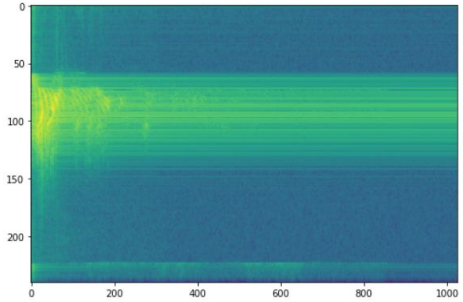
Feature extractor

gather audio data

select features




```
size_t num_samples_read;
TfLiteStatus generate_status = GenerateMicroFeatures(
    error_reporter, audio_samples, audio_samples_size, kFeatureSliceSize,
    new_slice_data, &num_samples_read);
if (generate_status != kTfLiteOk) {
  return generate_status;
}
```



Audio provider

Feature extractor

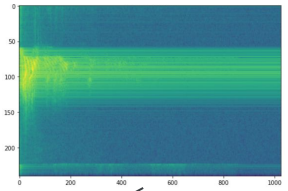
gather audio data

select features

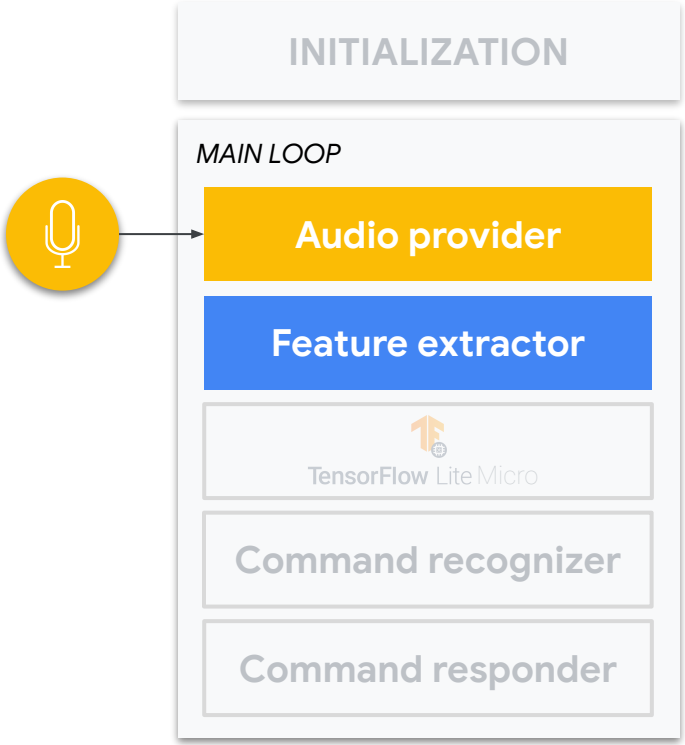
while buffer != FULL

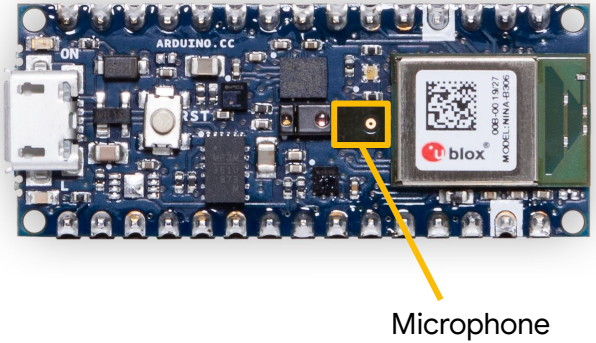
Get audio samples

extract features



to inference


INITIALIZATION
MAIN LOOP
Audio provider
Feature extractor
TensorFlow Lite Micro
Command recognizer
Command responder


ARDUINO.CC
ON
RST
blox
Microphone

Audio provider

Feature extractor

gather audio data

select features