Part 05
-
Microphone

Version: 2021-06-01

Reading microphone data

The Nano RP2040 connect comes with the MP34DT05 microphone, which can be used to record audio. In this tutorial, we will setup a basic application that simply turns ON or OFF the built in RGB LED whenever a loud noise is recorded (for example snapping our fingers).

The MP34DT05 microphone
Microphones are components that convert physical sound into digital data. Microphones are commonly used in mobile terminals, speech recognition systems or even gaming and virtual reality input devices.
The MP34DT05 sensor is a ultra-compact microphone that use PDM (Pulse-Density Modulation) to represent an analog signal with a binary signal. The sensor's range of different values are the following:

- Signal-to-noise ratio: 64dB
- Sensitivity: -26dBFS ±3dB
- Temperature range: -40 to 85°C

If you want to read more about the MP34DT05 sensor you can take a look at the datasheet.

PDM library
The PDM library allows you to use PDM (Pulse-density modulation) microphones, such as the on board MP34DT05 on the Arduino Nano 33 BLE Sense.

To use this library

```
#include <PDM.h>
```

The library takes care of the audio that will be accessible also through the `ArduinoSound` Library.

Arduino Sound library
This library provides simple way to play and analyze audio data using Arduino on SAMD21 based boardsusing the I2S bus.

To use this library

```
#include <AudioSound.h>
```

Programming the board

We will now get to the programming part of this tutorial.

1. First, let's make sure we have the drivers installed. If we are using the Web Editor, we do not need to install anything. If we are using an offline editor, we need to install it manually. This can be done by navigating to `Tools > Board > Board Manager....` Here we need to look for the `Arduino Mbed OS Nano Boards` and install it.

2. We can now take a look at some of the core functions of this sketch:
   - `static const char channels = 1;` sets the number of output channels.
   - `static const int frequency = 16000;` sets the sampling frequency to 20 KHz.
   - `short sampleBuffer[512]` buffer to read samples into, each sample is 16-bits.
   - `while (!Serial)` prevents program from running until Serial Monitor is opened.
   - `PDM.begin(channels, frequency)` initializes the PDM library.
   - `Serial.print(sampleBuffer[i]` prints sample to the Serial Monitor / Plotter.

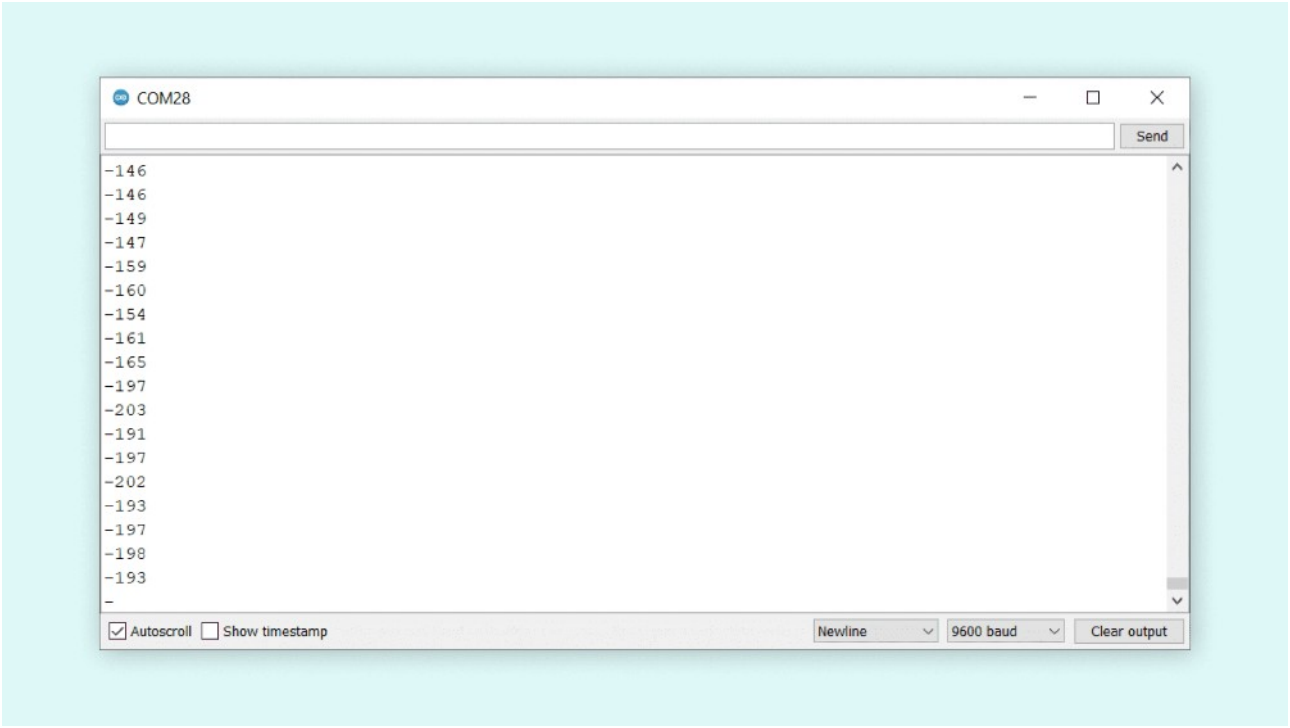
The sketch can be found in the snippet below. Upload the sketch to the board.

```
#include <WiFiNINA.h>
#include <PDM.h>

bool LED_SWITCH = false;

// default number of output channels
static const char channels = 1;

// default PCM output frequency
static const int frequency = 16000;

// Buffer to read samples into, each sample is 16-bits
short sampleBuffer[512];

// Number of audio samples read
volatile int samplesRead;

void setup()
{
  Serial.begin(9600);
  pinMode(LEDB, OUTPUT);
  while (!Serial);
  // Configure the data receive callback
  PDM.onReceive(onPDMdata);

  // Optionally set the gain
  // Defaults to 20 on the BLE Sense and -10 on the Portenta Vision Shield
  // PDM.setGain(30);

  // Initialize PDM with:
  // - one channel (mono mode)
  // - a 16 kHz sample rate for the Arduino Nano 33 BLE Sense
  // - a 32 kHz or 64 kHz sample rate for the Arduino Portenta Vision Shield
  if (!PDM.begin(channels, frequency))
  {
    Serial.println("Failed to start PDM!");
    while (1);
  }
}

void loop()
{
  // Wait for samples to be read
  if (samplesRead)
  {

    // Print samples to the serial monitor or plotter
    for (int i = 0; i < samplesRead; i++)
    {
      if (channels == 2)
      {
        Serial.print("L:");
        Serial.print(sampleBuffer[i]);
        Serial.print(" R:");
        i++;
      }
      Serial.println(sampleBuffer[i]);

      if (sampleBuffer[i] > 10000 || sampleBuffer[i] <= -10000)
      {
        LED_SWITCH = !LED_SWITCH;
        if (LED_SWITCH) {
```

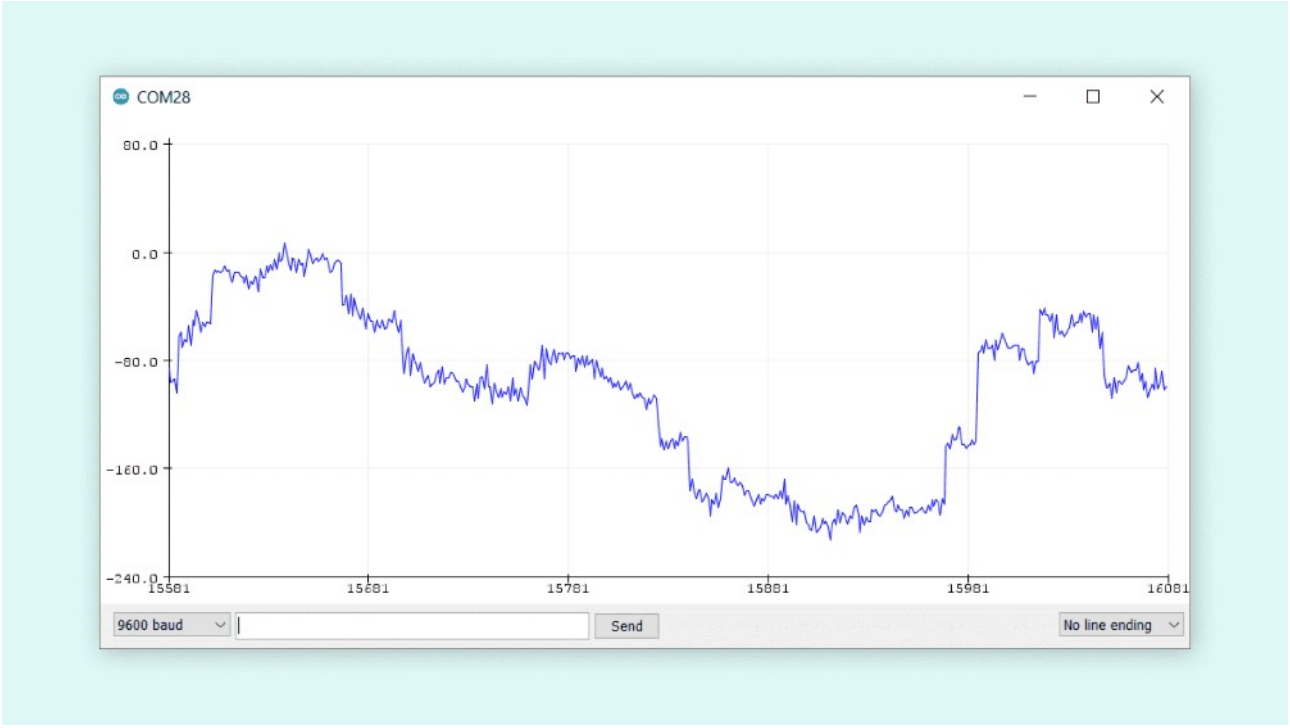
```
          Serial.println();
          digitalWrite(LEDB, HIGH);
          Serial.println("ON!");
          Serial.println();
          delay(1000);
        }
        else
        {
          Serial.println();
          digitalWrite(LEDB, LOW);
          Serial.println("OFF!");
          Serial.println();
          delay(1000);
        }
      }
    }

    // Clear the read count
    samplesRead = 0;
  }
}

/**
   Callback function to process the data from the PDM microphone.
   NOTE: This callback is executed as part of an ISR.
   Therefore using `Serial` to print messages
   inside this function isn't supported.
 * */
void onPDMdata()
{
  // Query the number of available bytes
  int bytesAvailable = PDM.available();

  // Read into the sample buffer
  PDM.read(sampleBuffer, bytesAvailable);

  // 16-bit, 2 bytes per sample
  samplesRead = bytesAvailable / 2;
}
```

Testing it out

After successfully uploading the code to the board, we will need to open the Serial Monitor to initialize the program. Once we open it, we can see that data is being printed rapidly. These are the audio samples that are record from the microphone. We can also open the Serial Plotter to instead view the data as a graph.



We can also open the Serial Plotter to instead view the data as a graph.



Now in the sketch, we have written a conditional that is activated whenever the sample value is either over 10000 or under -10000. This can be produced, by for example snapping our fingers, coughing loudly or if you prefer, yelling (we prefer the finger snapping method). Whenever this condition is met, the following things happen:

- The boolean LED_SWITCH changes.
- Depending on the state of LED_SWITCH, the blue pixel turns ON or OFF.
- The text "ON" or "OFF" is printed in the monitor.
- The program freezes for one second, then resumes (this is just to allow some time to see the status of the LED).

In the Serial Monitor we should see the following when the conditional is met, where the blue pixel will turn ON / OFF depending on the state of LED_SWITCH. The blue RGB pixel either turns ON/OFF when the condition is met.