Tiny Embedded AI Voice Recognition Application

SPECIFICs, INTERACTIONS AND CONTROL OF NDP101

Start Sources:

• Documentation info: [Link 1](https://44-2.de/syntiant-ndp-101-always-on-low-power-speech-recognition/)

• Example Project with this Software & Hardware Implementation: [Link 2](https://github.com/happychriss/Goodwatch_NDP101_SpeeechRecognition)

In this part of the internship, I focused on collect details on the architecture of NDP101. This was done using some inputs given by professor Yildirim.

Immagine che contiene schermata, testo, Carattere, diagramma

Descrizione generata automaticamente

1. What is the power consumption of NDP101

According to NDP101 Test Resume.pdf file ( <https://cms.tinyml.org/wp-content/uploads/talks2022/Yousefi-Alireza-hardware-FINAL.pdf> ), the authors of this paper conducted a study testing the device using Edge Impulse interfacing with both word spot recognition and the use of accelerometer. Considering that NDP101 is an always-on power consumption application it manages to use a power consumption of 140 uW for audio/voice recognition and 100 uW, these data compared to other CPU/MCUs it should deliver according to the paper 20 times more throughput and 200 times less energy per inference. However, this was taken from another source that was shut down, the original link was <https://www.syntiant.com/news/keyword-spotting-power-comparison>, so a news shared from the Syntiant site, so I don’t know if that is outdated

1. What kind of features can be extracted using NDP101? What is the internal architecture? How can we program it? How much memory does it have? What kind of operations?

We know that the structure of NDP101 in specific should be like the following one:

Immagine che contiene testo, schermata, diagramma, Rettangolo

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, linea, numero

Descrizione generata automaticamente

The images come from: <https://44-2.de/syntiant-ndp-101-always-on-low-power-speech-recognition/>

The core inside the NDP101 is a Cortex-M0 containing the data with a ARM binary logic. The memory contained native in it is of 112KB, but it may be extended with a 32 Gb SD memory extension. The NDP101 only has an SPI communication that directly interacts with the host processor, but it technically should enable more targets, because according to the image below 5 of the pins can be used as GPIO, so may be used for communication. It looks like it is mainly used like a slave, so an external/master device sends a command via SPI or I2C that involves actions like model upload or audio sampling through a model and then returns the result to the start device. The flash memory instead is of 2MB on-board serial flash and uses a clock system for communication of 48MHz. So, this confirms that the board connects a normal Arduino ARM Cortex-M0+ based that is an Arduino MKR Zero with the NDP101 that bases primarily on the neural network. A more detailed and visual structure can be seen in this image, showing the structure of the board:

Immagine che contiene testo, Ingegneria elettronica, Componente elettrico, circuito

Descrizione generata automaticamente

I already described in the previous part, that we can already program the Arduino via it’s IDE and using an ingestion to support the program of it, instead to program the NDP101 directly should be required a SDK.

The board as has memory structure as summary of:

• Data RAM 112 KB

• Flash RAM 2MB

As of now it looks like is not available working directly on NDP101, because SDK is under NDA, so the possible operations can’t override the default action, which leads in loading a model to the neural network and sending the data via the SPI.

1. How can we program it w/o Arduino? Is there any SDK to program in pure C?

To program the NDP101 without using the Arduino is technically possible, because Syntiant provided a Software Development Kit (SDK) which integrates into any software environment. Source: <https://www.syntiant.com/ndp101#data_sheet> As IDE every software platform with Embedded-oriented programming can be used, but the problem arises because SDK of Syntiant NDP101 is under NDA and available only to business partners.

3.1) How can I program host processor? Is it ARM-Cortex?

The host processor (Arduino MKR Zero) is a Cortex-M0+ so 32bit ARM-based, the same as NDP101. The first because being a host acts like the master during communication and management and NDP101 will instead be the slave during the SPI communication, between the two when performing data exchange.

3.2) How can I control NDP101? Driver?

It should be controlled using a Host Processor, like the standard implementation of the device or we can use the GPIO pins that can be viewed in Figure 2.03 that shows 5 configurable GPIO pins that may be setup with other devices using a I2C or SPI communication. It is important to note that its slave nature, should be respected, so it should always have a host that could be different from the Arduino MKR Zero, but external interfacing should be possible on paper. To program the code, instead the answer is the same in the 3) question, so NDA limit.

3.3) Can I connect another MCU to NDP101?

According to the information, it should be possible to connect other MCUs via the GPIO pins, via a tertiary mode, but technically the communication that is desired is via SPI, between NDP101, which is a slave and another host processor, so if we would like to make more NDP101 communicate with another NDP101 there should be a platform to be a bridge between the two, so a master for them both. On this Syntiant, there is locally a master, which is the Arduino MKR Zero, but the interconnection should be of other nature.

3.4) What is PDM Connection?

PDM consists in Pulse Density Modulation, a signal represented in binary, like a stream of single bits. Always in image 2.03 can be seen that there is that connection that is theoretically dedicated to audio input, that then will be converted in PCM (Pulse Code Modulation), which is then a system that represents the signal in multiword according to the model uploaded and this should allow word-spotting. So that connection should be the audio input.

To spot this in-depth I digged in this source that described it in general and not in the particular of this Embedded System. Source: <https://users.ece.utexas.edu/~bevans/courses/rtdsp/lectures/10_Data_Conversion/AP_Understanding_PDM_Digital_Audio.pdf>

1. Is there any code in Edge Impulse github repo for NDP101? Is that code usable or useful?

Edge Impulse implemented for its system some libraries in support of NDP101 to facilitate the machine learning and provide and interaction without modifying the source code. The link to the firmware github repository: <https://github.com/edgeimpulse/firmware-syntiant-tinyml/tree/master>

To program NDP101, we would require the SDK and in the firmware is present an ingestion-SDK of the Syntiant system. To recall, SDK is a collection of libraries and documentation provided to build applications on a specific platform, instead the DSK Ingestion is a technique used to insert or modify an SDK into an application without modifying its source code, like in the case of Edge Impulse that created this firmware to let the user interface with NDP101. This should act on it without modifying the source code, so not acting on SDK. This repository provides a foundation for building custom firmware tailored to specific audio projects and the impulse allows integrating the impulse into this firmware to facilitate deployment and testing. Viewing the code, surely, we may use the ingestion of Syntiant, which points to custom feature extraction, so improving it and performing audio pre-processing, but we would not have access to the full pipeline from the model deployment to optimization, because of NDA. Files description in it:

• ei\_syntiant\_fs\_commands.cpp – file that handles data storage and retrieving on Arduino MKR Zero that integrates the Syntiant NDP101 and redirects in RAM, MicroSD and Serial Flash Memory

• ei\_device\_syntiant\_samd.cpp – file that represents the Edge Impulse device that runs on the Syntiant Platform, providing a device identification, sensor management, communication and inference control.

• ei\_device\_info.h – file that defines the device providing essential information and specifics like device type, sensors management, memory handling and timing functions.

To make able communication between 2 MCU we can take inspiration from syntiant.cpp in src folder that looks like a bridge between NDP101 and Arduino, so pick the results elaborated with NDP101 and sends to another desired MCU, considering that using the SPI the NDP101 is a slave, so the other should be a host or we may use a UART one using one channel for sending and one for receiving.