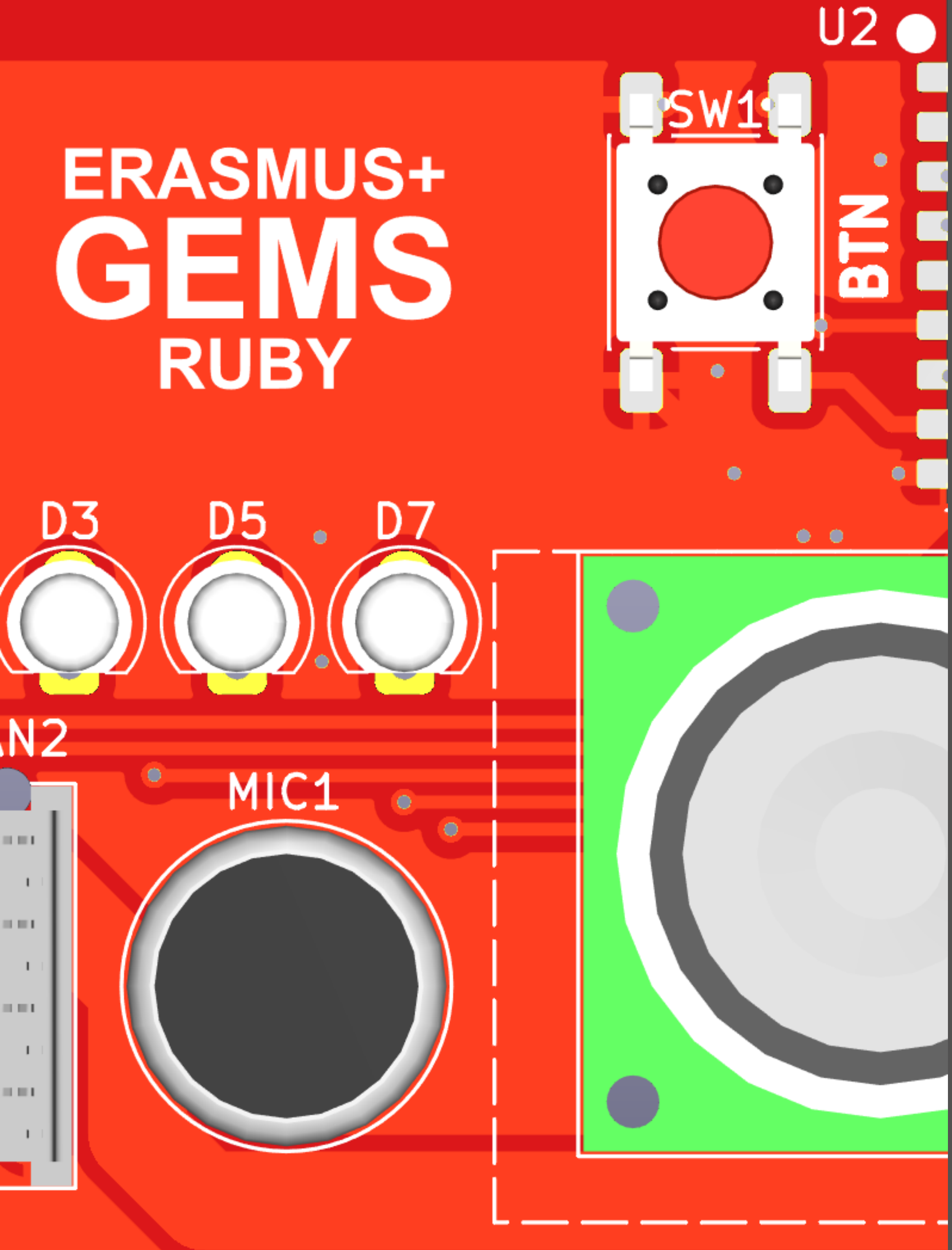


Analog to Digital Conversion

Tutorial

Daniel Pizarro Pérez (University of Alcala)

Index

Objectives

This tutorial is focused on

- Understanding the basic concepts of Analog-to-Digital (ADC) conversion.
- ADC conversion for the ESP32 micro-controller.
- Two-channel audio acquisition in the Ruby module

This tutorial is not focused on

- In-depth ADC control and analysis
- High Quality audio conversion

Basic concepts (I)

What is ADC?

Definition: Analog-to-Digital Conversion (ADC) is the process of converting continuous analog signals into a discrete digital form.

Why is ADC important?

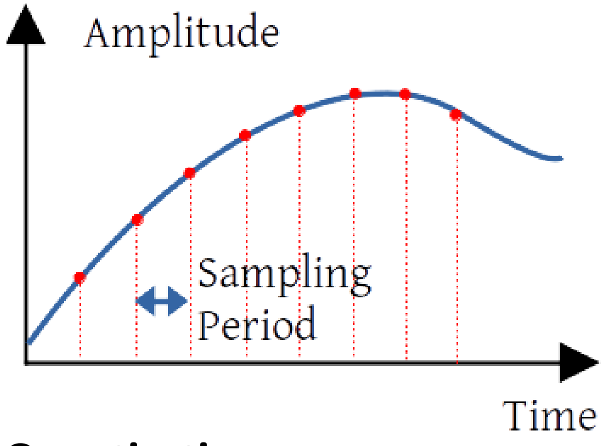
- Enables digital systems to interact with the analog world.
- Used in a wide range of applications such as audio processing, instrumentation, communications, and medical devices.

Basic Concept:

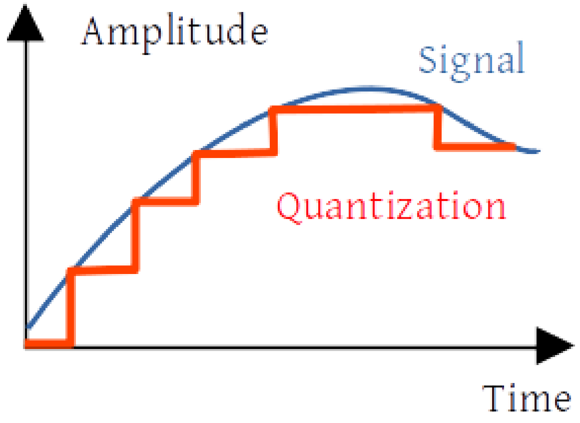
Sampling: Measuring the analog signal at discrete intervals.

Quantization: Converting the sampled values into a finite number of levels.

Sampling



Quantization



Basic concepts (II)

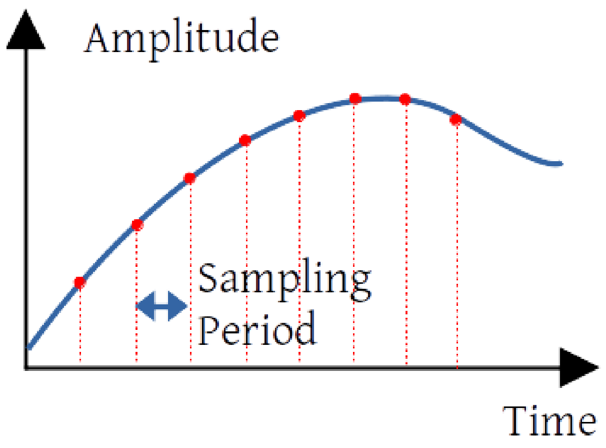
Key concept: Sampling Rate

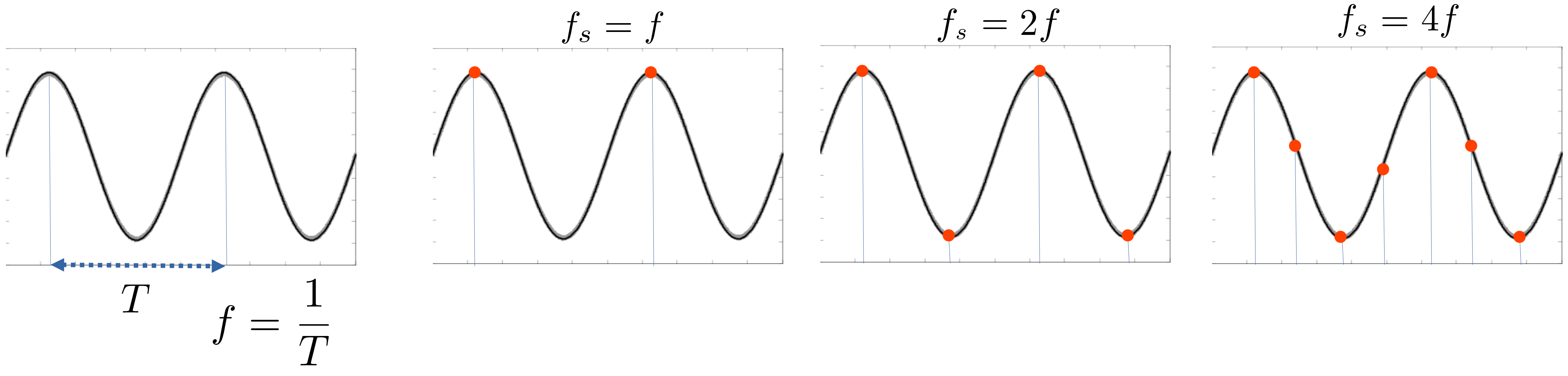
Definition: The number of samples taken per second, measured in Hertz (Hz).

Nyquist Theorem: To accurately represent an analog signal, the sampling rate must be at least twice the highest frequency present in the signal.

Sampling Rate (Hz) = 1 / Sampling Period (sec)

Sampling





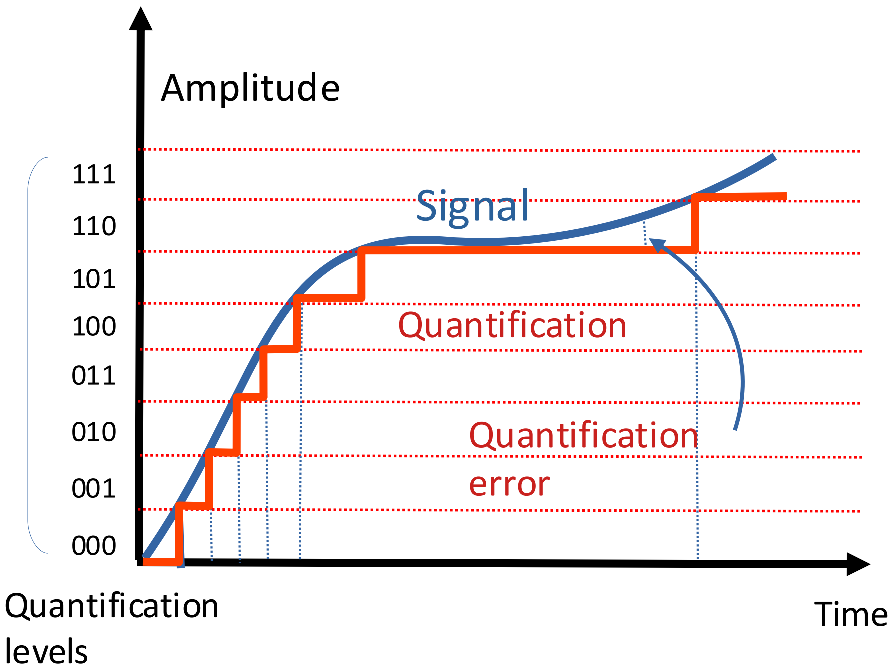
Basic concepts (II)

Key concept: Resolution

Definition: The number of distinct values that the ADC can produce, typically measured in bits.

Example: A 3-bit ADC can produce 8 (2^3) discrete values.

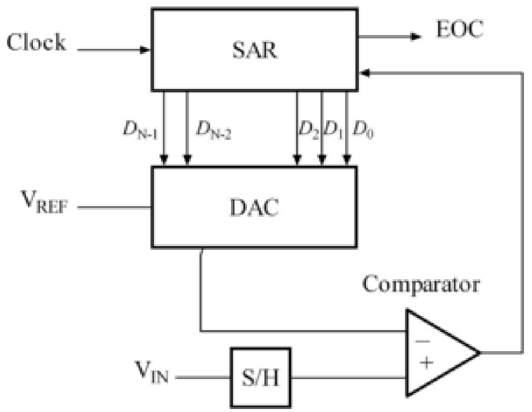
8 discrete values
3-bit ADC



Basic concepts (IV)

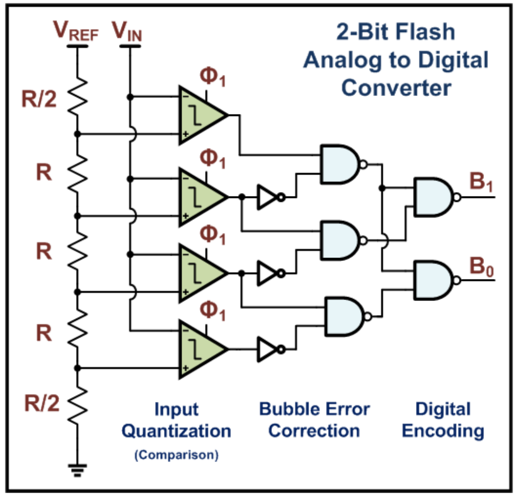
Key concept: Common ADC arquitectures

- Successive Approximation Register (SAR): Popular for moderate speed applications with high resolution.



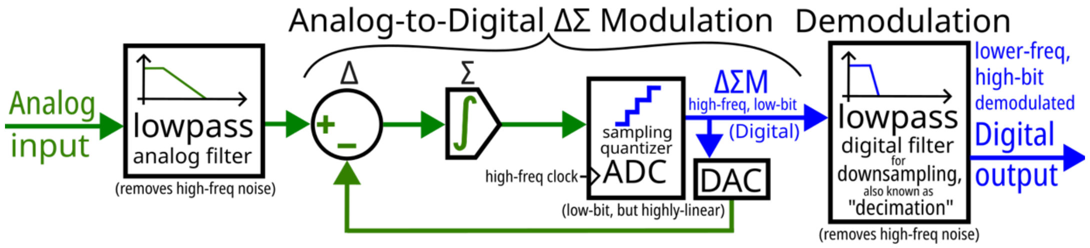
Source. wikipedia.

- Flash ADC: Extremely fast but consumes more power and has lower resolution.



Source. wikipedia.

- Sigma-Delta ADC: Used in high-resolution, low-speed applications like audio.



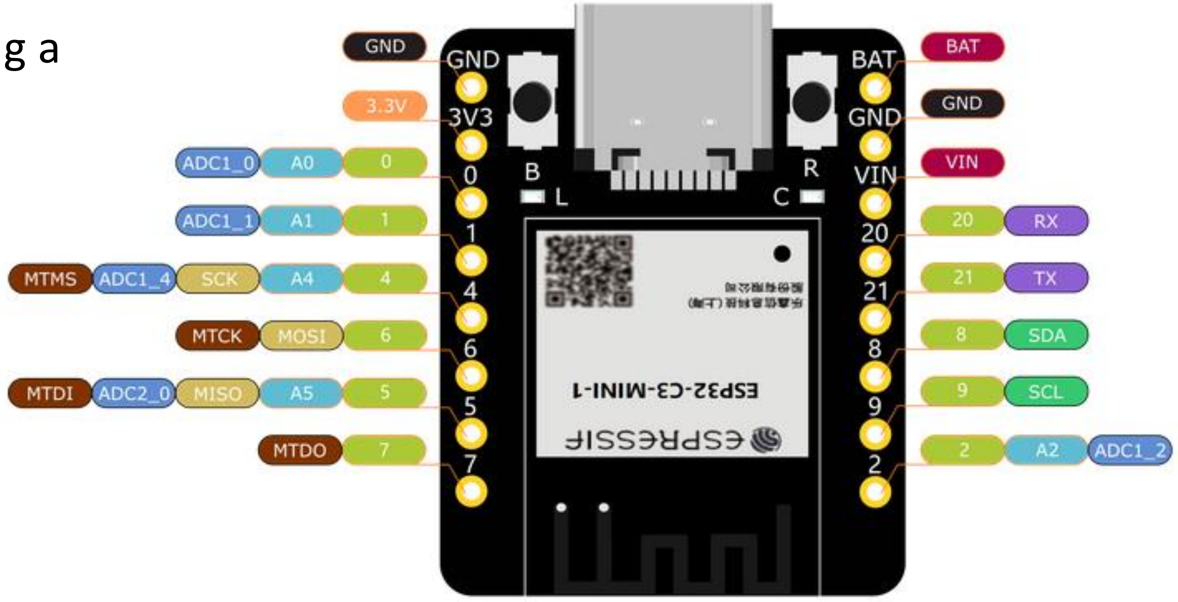
Source. wikipedia.

ADC in the Ruby module

ESP32-C3 [1]

- 2 SAR (Successive Approximation Register) ADCs, supporting a total of 6 measurement channels (analog enabled pins).
  ADC1: 5 channels: GPIO0 - GPIO4
  ADC2: 1 channels: GPIO5

- ADC1 Max 83 Khz single channel sampling frequency
- 12 bit conversion resolution.



[1]

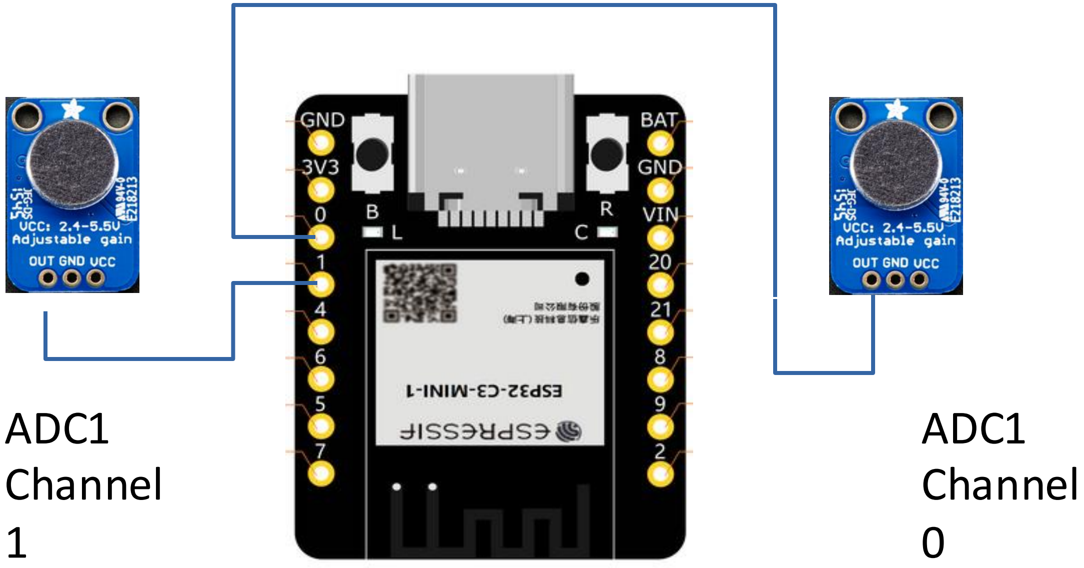
4 attenuation factors [2]

0 dB	0 mV ~ 750 mV
2.5 dB	0 mV ~ 1050 mV
6 dB	0 mV ~ 1300 mV
11 dB	0 mV ~ 2500 mV

ADC in the Ruby module

Microphone Sensors in the Ruby Module

- Two amplified microphones (MAX4466)
- Silence mean voltage 1.2 V
- Controlled Gain.



[1]

0 mV ~ 750 mV
0 mV ~ 1050 mV
0 mV ~ 1300 mV
0 mV ~ 2500 mV

11 dB Attenuation factor

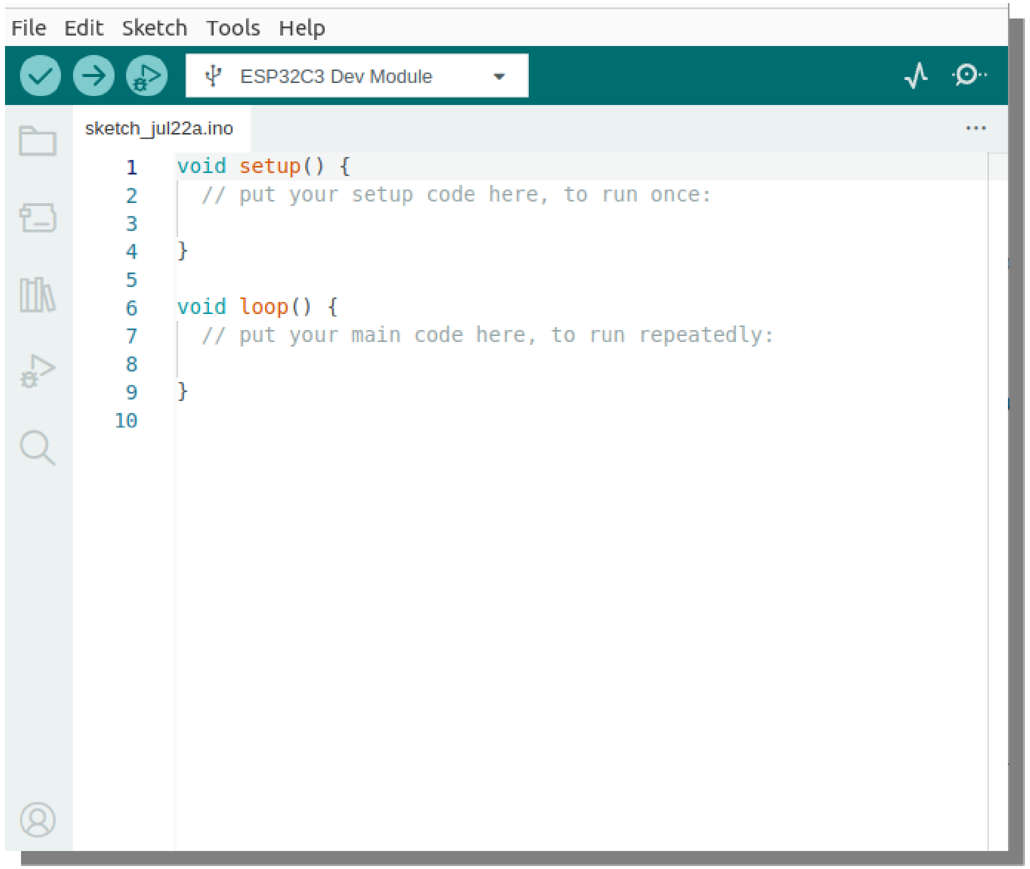
2-channel Audio Acquisition example

Objective:

- Capturing audio signals from two microphones connected to ADC1 channels 0 and 1.
- Resolution: 12 bits
- Sampling frequency: 20 Khz
- Visualization through serial port.

Environment:

- Arduino + ESP32 libraries



[3]

2-channel Audio Acquisition example

Code breakdown: Initialization and Setup

- Constants and Variables:
  - CONVERSIONS_PER_PIN: Number of conversions per pin.
  - Nsamples: Number of samples to collect.
  - Arrays sleft and sright for storing samples.
  - ADC Pin Configuration:
  - adc_pins[]: ADC channels used (0 and 2).
  - adc_pins_count: Number of ADC pins.

```
#define CONVERSIONS_PER_PIN 10
uint8_t adc_pins[] = {1, 0};
#define Nsamples 1024
int n;
int sleft[Nsamples];
int sright[Nsamples];
uint8_t adc_pins_count = sizeof(adc_pins) / sizeof(uint8_t);
volatile bool adc_coversion_done = false;
adc_continuous_data_t *result = NULL;
```

2-channel Audio Acquisition example

Code breakdown: Setup Function

- Serial Communication: Initialized at 115200 bits per second.
- ADC Configuration:
  - Set resolution to 12 bits.
  - Set attenuation to 11db.
  - Configure continuous ADC with pins, conversion count, frequency (20,000 Hz x 2 channels), and ISR callback.
  - Start continuous ADC conversions.

```
void setup() {
  Serial.begin(115200);
  analogContinuousSetWidth(12);
  analogContinuousSetAtten(ADC_11db);
  analogContinuous(adc_pins, adc_pins_count, CONVERSIONS_PER_PIN, 40000, &adcComplete);
  analogContinuousStart();
}
```

2-channel Audio Acquisition example

Code breakdown: Data Processing in Loop Function

- ISR Flag Check: Processes data if conversion is complete.
- Data Reading and Storage: Stores average mV values in arrays sleft and sright.
- Stop and Restart ADC: Stops ADC after collecting Nsamples and prints the data.
- Delay: Adds a delay for readability.

```
void loop() {
  if (adc_coversion_done == true) {
    adc_coversion_done = false;
    if (analogContinuousRead(&result, 0)) {
      sleft[n] = result[0].avg_read_mvolts;
      sright[n] = result[1].avg_read_mvolts;
      n++;
    } else {
      Serial.println("Error occurred during reading data.");
    }
  }

  if(n >= Nsamples) {
    analogContinuousStop();
    n = 0;
    for (int k = 0; k < Nsamples; k++) {
      Serial.printf("%d,%d\n", sleft[k], sright[k]);
    }
    delay(1000);
    analogContinuousStart();
  }
}
```

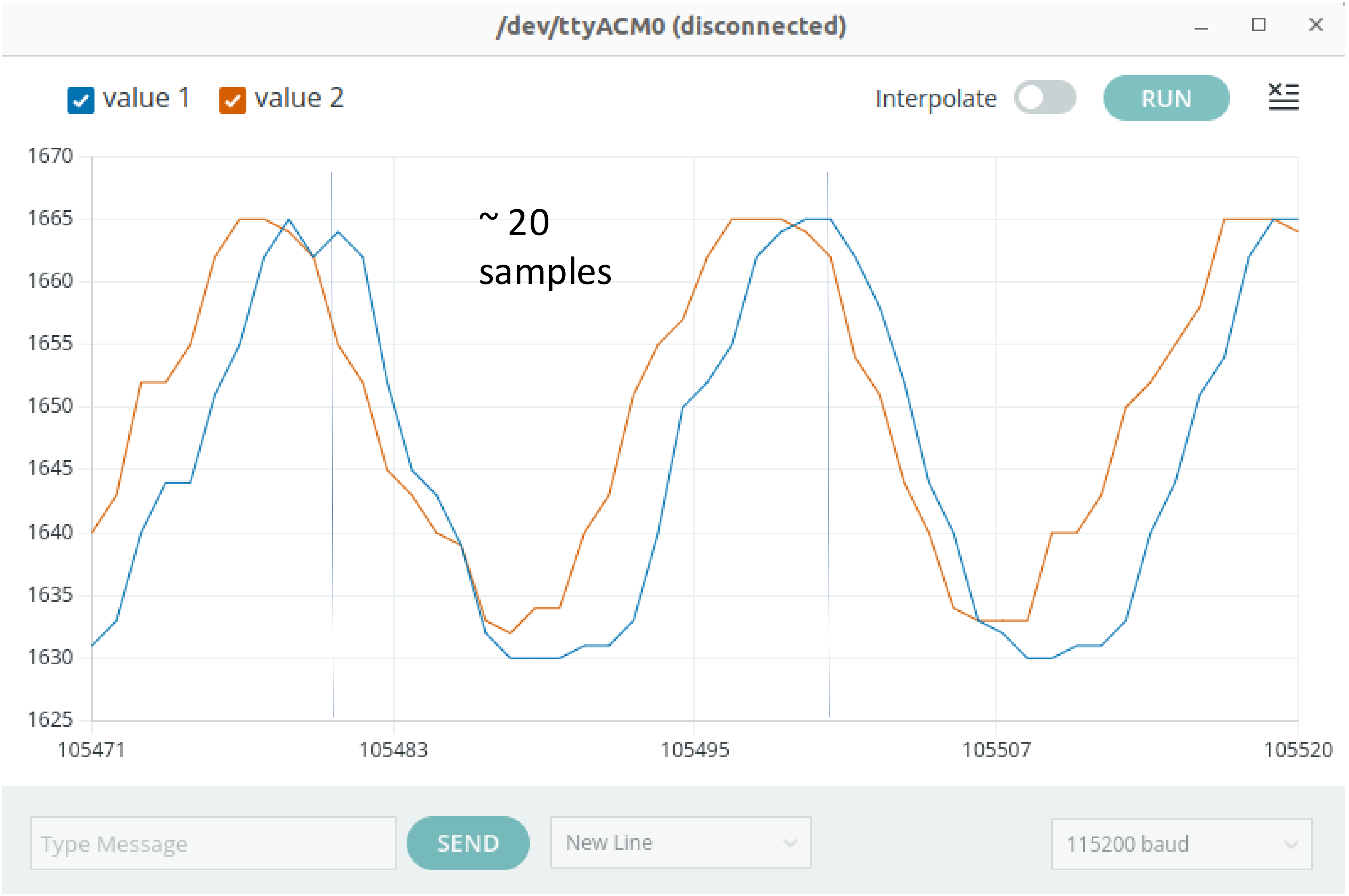
2-channel Audio Acquisition example

Experiment

- 1 Khz pure tone audio source
- Visualization in Arduino Serial Plotter

left right

1
Khz



~ 20
samples

Conclusions

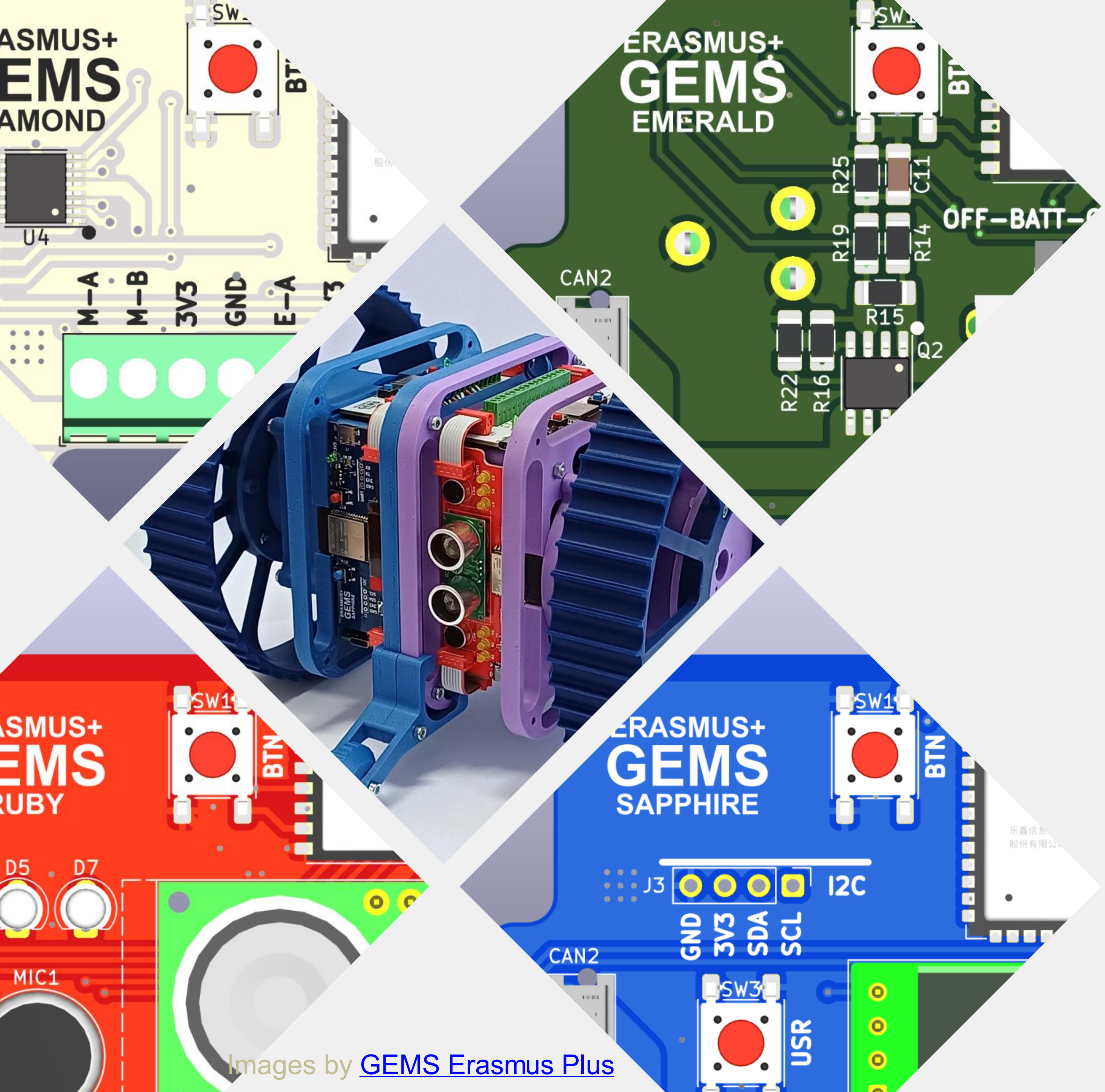
- Analog-to-Digital converters allow to capture analog signals in a digital system, such as a microcontroller.
- The Ruby module ESP32 microcontroller has 2 built-in multichannel ADCs.
- The microphone sensors in the Ruby module are connected to two channels in ADC1.
- This tutorial showed how to simultaneously acquire two signals from the microphones and send them with a serial port for visualization

References

[1] DF-ROBOT ESP32-C3 https://wiki.dfrobot.com/SKU_DFR0868_Beetle_ESP32_C3

[2] Espressif ADC Documentation ESP32 https://docs.espressif.com/projects/arduino-esp32/en/latest/api/adc.html

[3] Arduino Software https://www.arduino.cc/en/software



Images by [GEMS Erasmus Plus]

Thank you for watching!

Co-funded by the European Union