Introduction to Arduino Prototyping

Edge-Computing with Arduino

Nicola Dall'Ora

Sebastiano Gaiardelli

UNIVERSITÀ di VERONA
Dipartimento di INFORMATICA

Cyber-Physical & IoT Systems Design

Outline

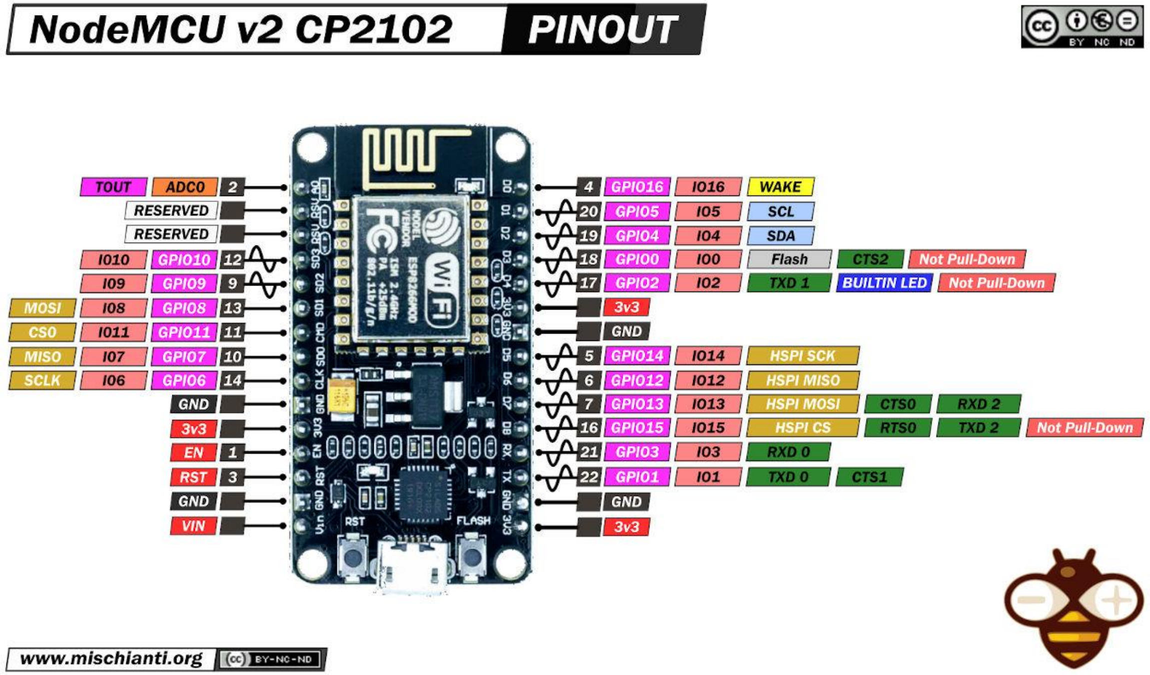
- Introduction
  - ESP8266/ ESP32
  - ESP-IDF
  - Arduino/ Arduino Pro
  - First prototype with ESP8266
- From sensors to cloud
  - ThingSpeak Cloud (Matlab)
  - Alternatives Cloud
- Project and report details

Introduction

ESP8266

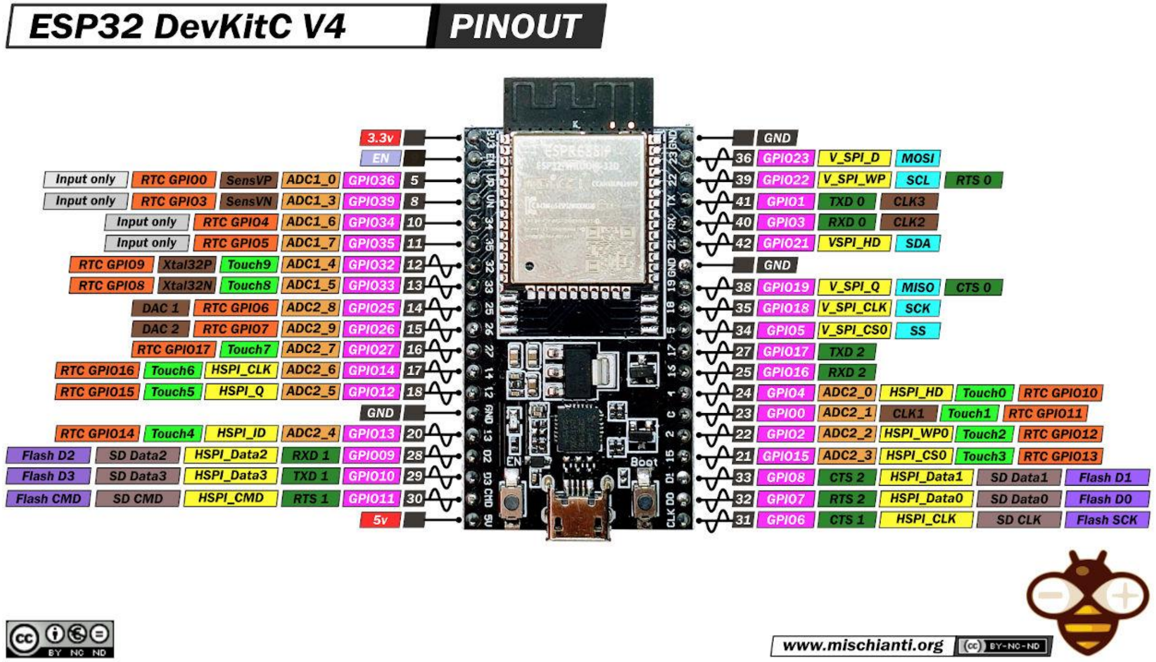
- Low-cost wireless device (<1$)

- Low-power consumption with 3 different standby modes:
  - Modem-sleep (15mA)
    - Turns off Wi-FI
  - Light-sleep (0.4mA)
    - Turns off Wi-FI
    - CPU & Clock suspended at intervals
  - Deep-sleep (~20µA)
    - Only RC active to wake up the device at intervals



ESP32

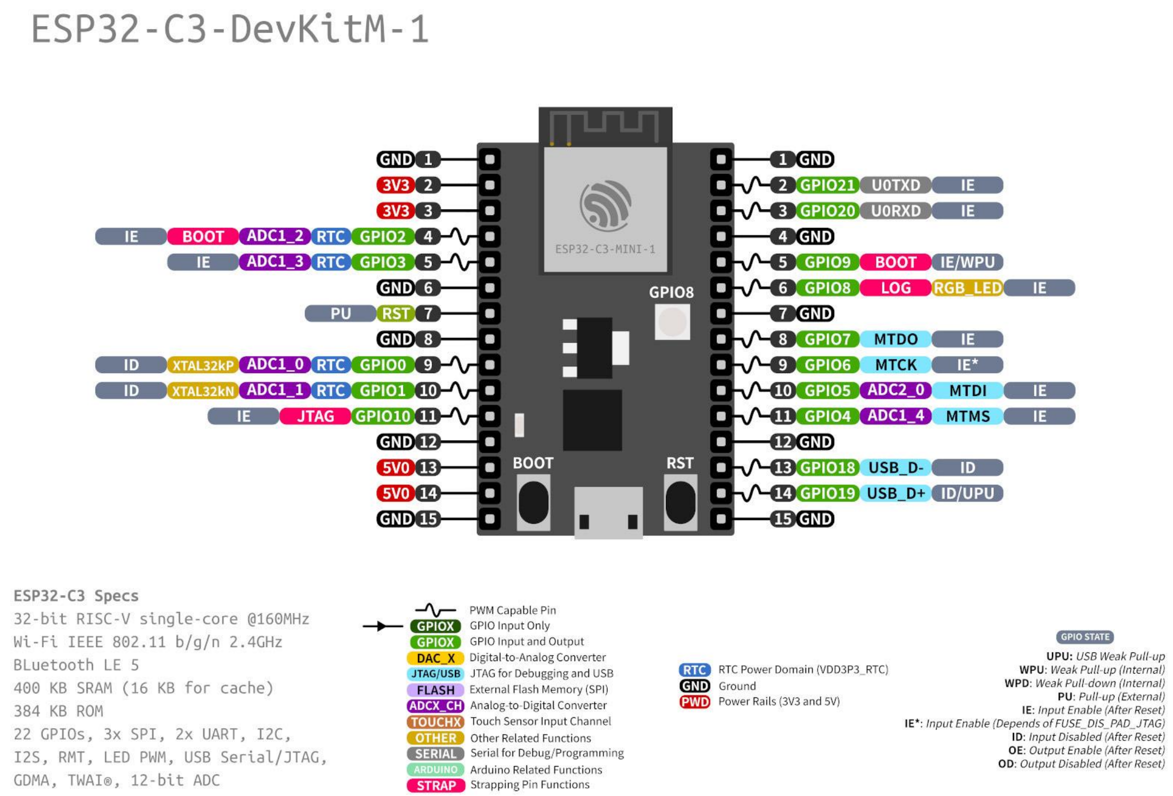
- Low-cost wireless device
- Extends the feature of the ESP8266 with:
  - Dual core with ultra-low power co-processor
  - Wi-Fi & Bluetooth
  - Advanced peripheral interface
    - GPIO, I2C, I2S, SPI, SD/MMC/SDIO
  - Secure boot and Flash Encryption



New ESP32 C3 (RISC-V)

RISC-V®
RISC-V: The Free and Open RISC Instruction Set Architecture

- ESP32-C3 is a single-core Wi-Fi and Bluetooth 5 (LE) microcontroller SoC, based on the open-source RISC-V architecture. It strikes the right balance of power, I/O capabilities and security, thus offering the optimal cost-effective solution for connected devices.
- The availability of Wi-Fi and Bluetooth 5 (LE) connectivity not only makes the device's configuration easy, but it also facilitates a variety of use-cases based on dual connectivity.

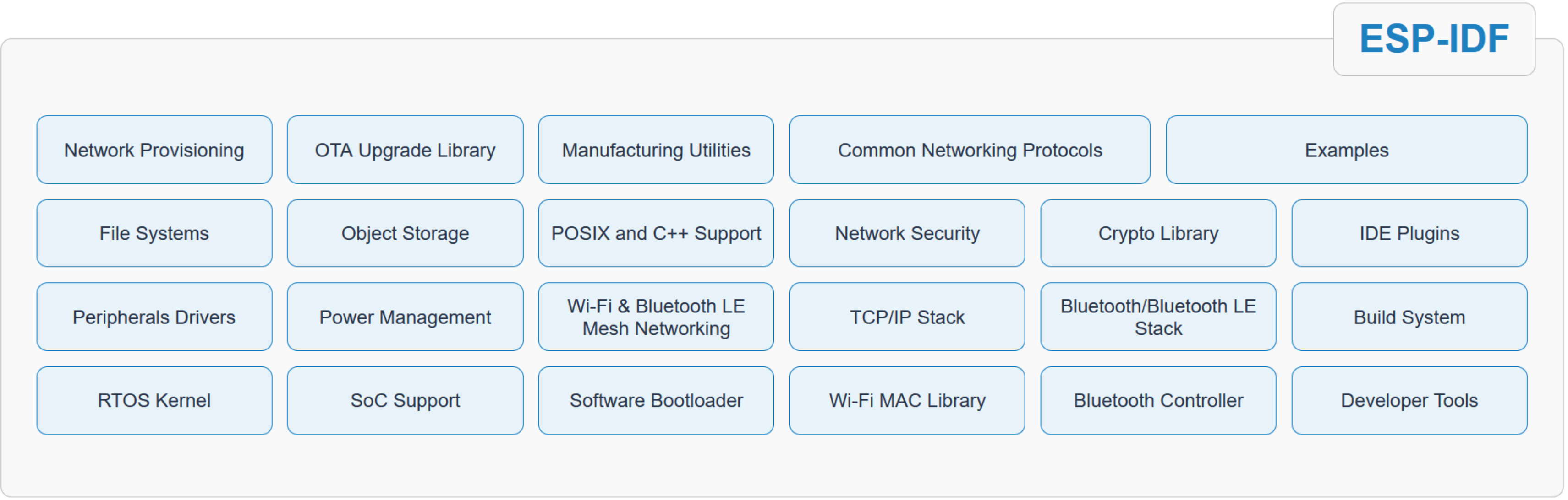


ESP8266 vs ESP32

Specifications	ESP8266	ESP32	ESP32-C6
MCU	Xtensa Single-Core 32-bit L 106	Xtensa Dual-Core 32-bit LX6 600 DMIPS	Dual-Core RISC-V (High Performance and Low Performance)
Wi-Fi	802.11 b/g/n, HT20	802.11 b/g/n, HT40	802.15.4 (Zigbee and Thread)
Bluethooth	N/A	Bluethooth 4.2 and below	Bluethooth 5.0 and below
Typical Frequency	80 MHz	160 MHz	160 MHz
SRAM	160 kBytes	512 kBytes	512 Kbytes HP, 16 Kbytes LP
Flash	SPI Flash up to 16 MBytes	SPI	SPI MBytes
GPIO	17	36	30
Hardware/Software PWM	None/ 8 Channels	1/ 16 Channels	1/10 Channels
SPI/I2C/I2S/UART	2/1/2/2	4/2/2/2	3/2/2/2
ADC	10-bit	12-bit	12-bit
CAN	N/A	1	1
Ethernet MAC Interface	N/A	1	1
Touch Sensor	N/A	Yes	Yes
Temperature Sensor	N/A	Yes	Yes

IoT Development Framework (ESP-IDF)

- Network-enabled Real Time Operating System (RTOS) built for Espressif's ESP32 family of components
  - Built on the FreeRTOS Kernel



- Source code: https://github.com/espressif/esp-idf
- Documentation: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html

IoT Development Framework (ESP-IDF)

- It bundles a network stack using LwIP and adapted the RTOS for their Wi-Fi, Bluetooth, and Thread modems
  - External components and libraries are available, including a build system and programming tools

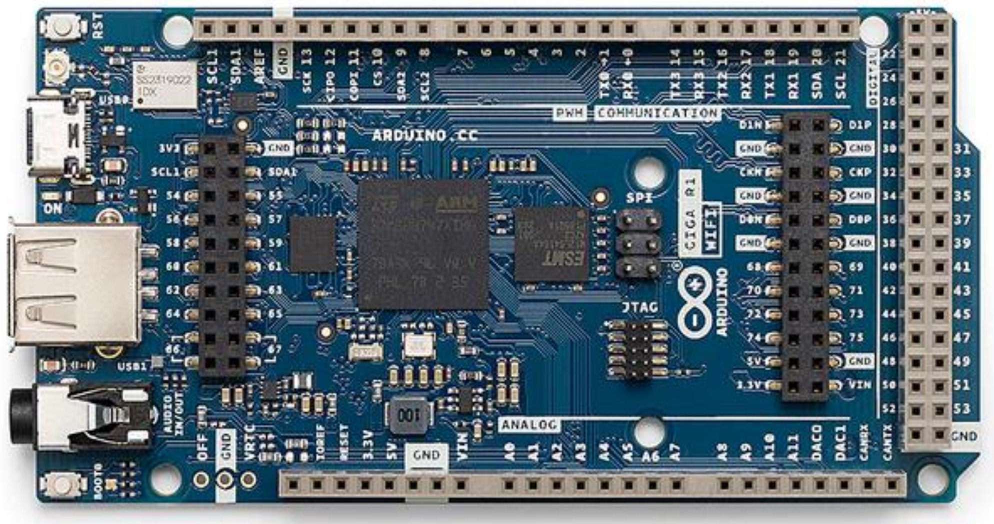
- Similar to a distribution
  - Can be personalized on the needs

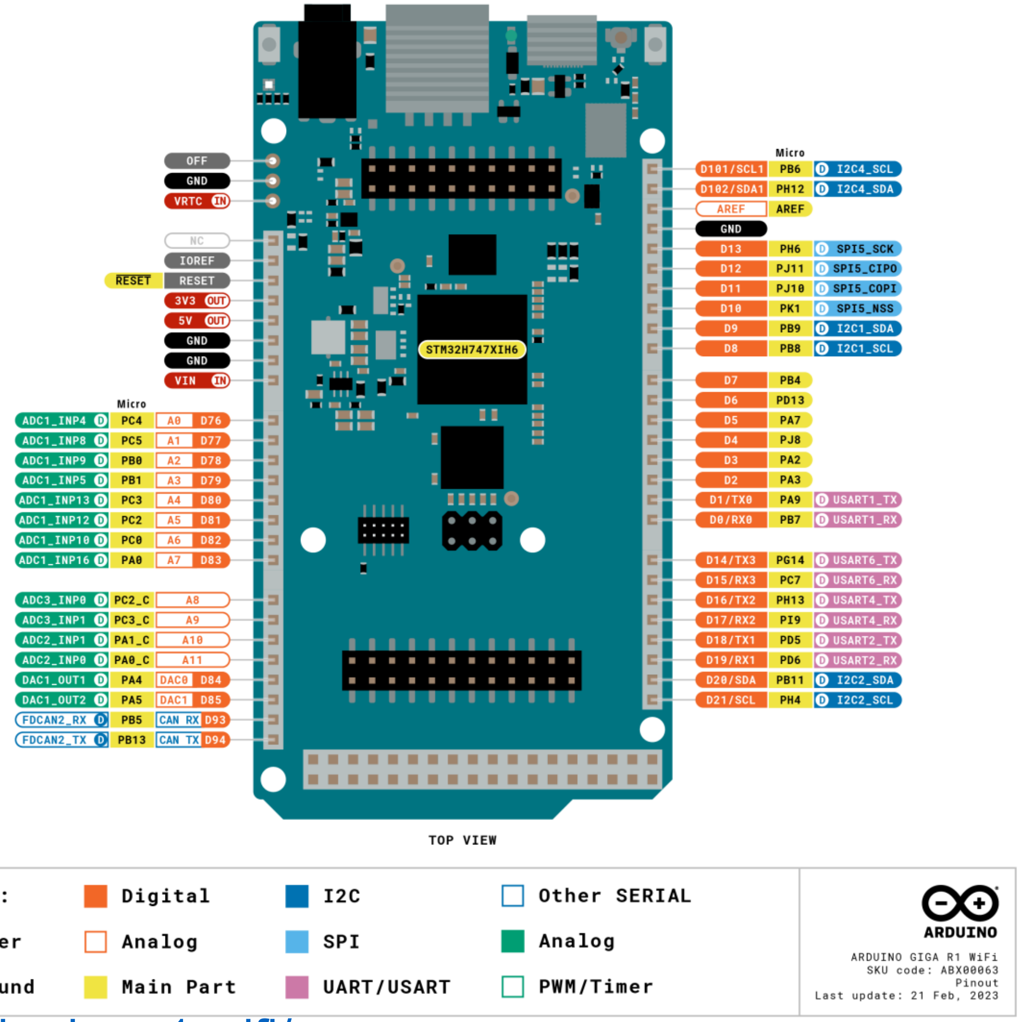
- It supports unit testing
  - Exploiting hardware simulator (e.g., QEMU, Renode, SimAVR)

New Arduino Giga R1 WiFi (ARM)

The Arduino GIGA R1 WiFi is designed for ambitious makers who want to step up their projects:

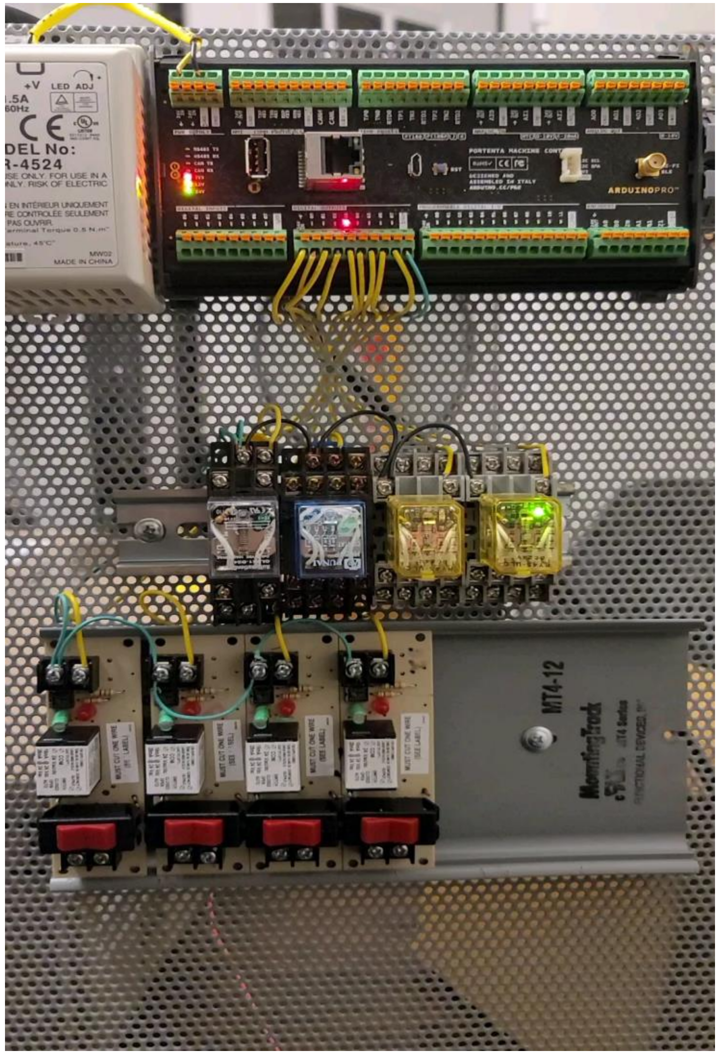
- The dual-core microcontroller allows you to run two Arduino program simultaneously
- Contains several interfaces;
- Enables Tiny ML applications.

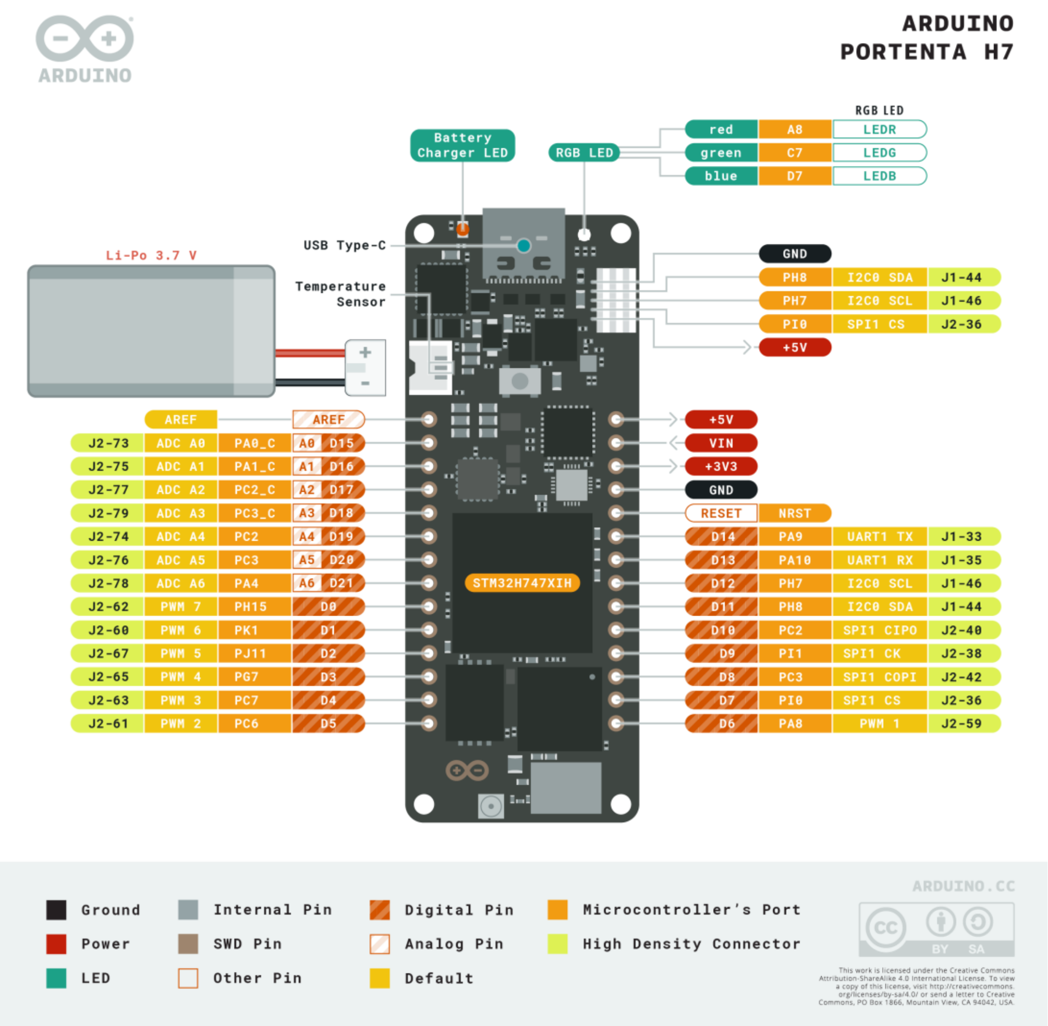




https://blog.arduino.cc/2023/03/01/step-up-your-game-with-giga-r1-wifi/

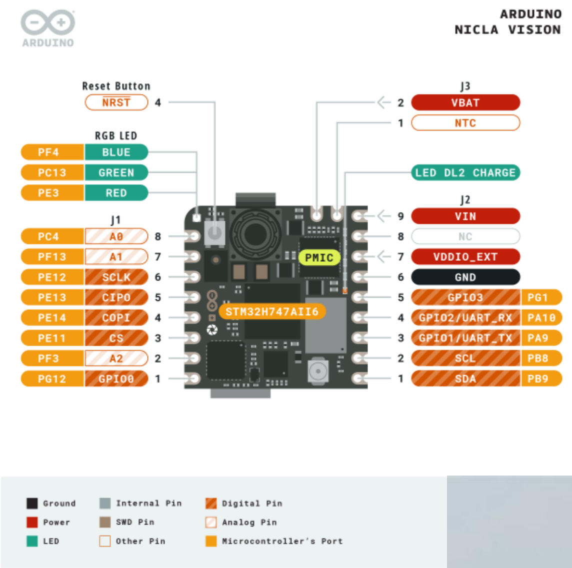
Arduino PRO Series (ARM)

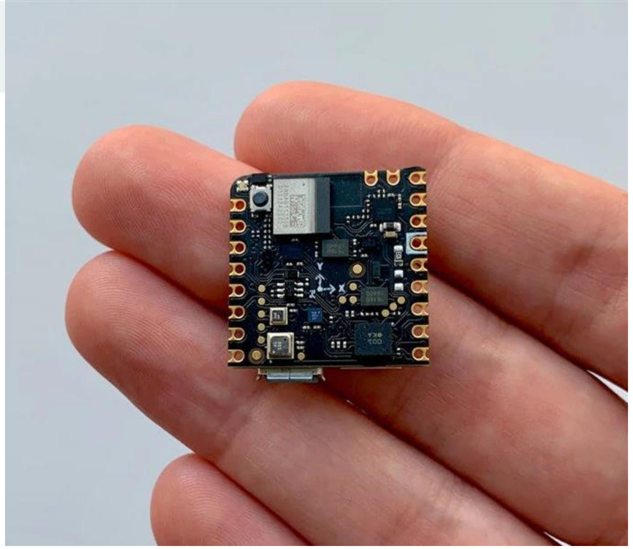




Arduino Pro (ARM)

- The **Nicla Sense ME** is a tiny, low-power tool that sets a new standard for intelligent sensing solutions. With the simplicity of integration and scalability of the Arduino ecosystem, the board combines four state-of-the-art sensors from Bosch Sensortec:
  - BHI260AP motion sensor system with integrated AI
  - BMM150 magnetometer
  - BMP390 pressure sensor
  - BME688 4-in-1 gas sensor with AI and integrated high-linearity, as well as high-accuracy pressure, humidity and temperature sensors.



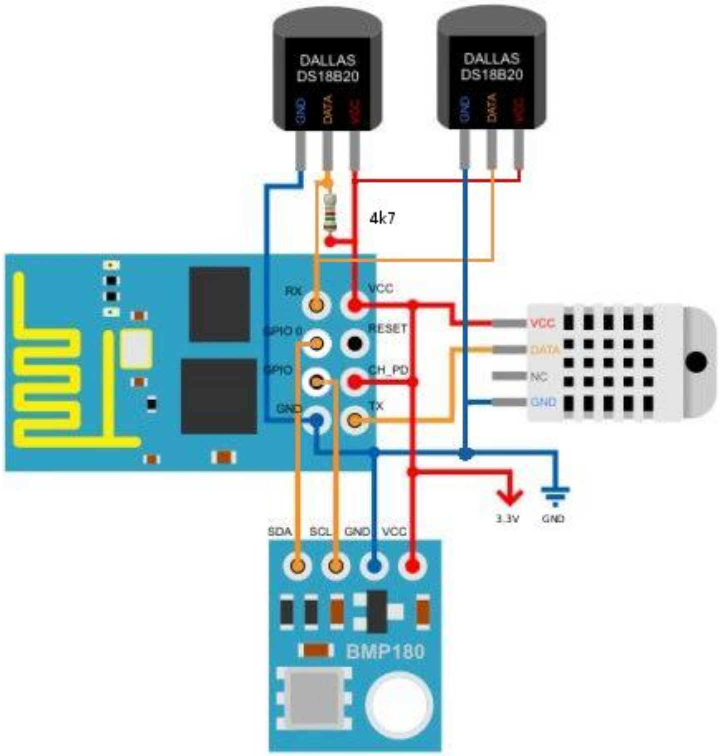


First Prototype with ESP8266

First Prototype with ESP8266 (1/6)

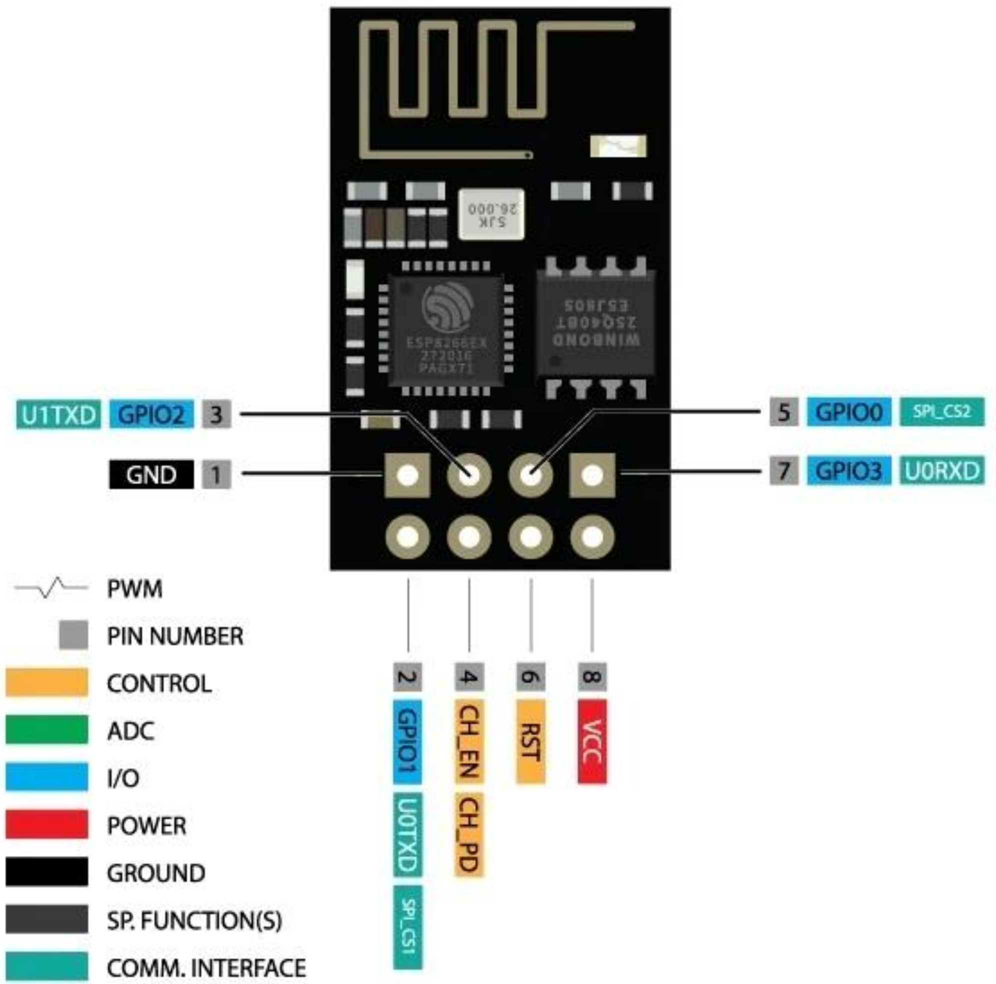
- Prototype of a sensing platform, with an Arduino board, a digital temperature sensor DS18B20 and a WiFi module ESP8266-01. The Arduino is used only as serial interface to the computer and to power the ESP module.
- To be able to compile and load binaries into the ESP8266, open the Arduino IDE and copy the following string in File/Preferences/Additional Board Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json

- Example of usage of ESP8266:



First Prototype with ESP8266 (2/6)

- The ESP8266 is a cheap module with a 32-bit MCU, an antenna and integrated Wi-Fi. It supports standard IEEE 802.11 b/g/n and complete TCP/IP protocol stack. It is powered by a voltage between 3.0 V and 3.6 V.
- Pin configuration:
  - GND - the ground pin
  - VCC - the power supply
  - RST - the reset pin
  - TX - transmission pin to the serial port
  - RX - Receive pin from the serial port
  - CH_PD - the chip enable
  - GPIO-0 - internal pull-up for booting mode
  - GPIO-2 - internal pull-up for reading data from the sensor

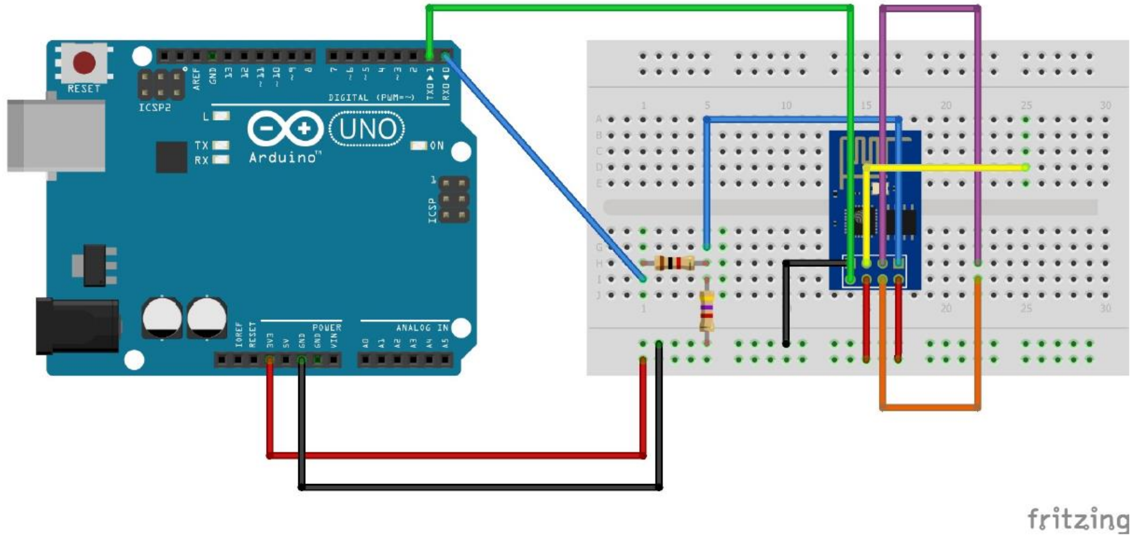


First Prototype with ESP8266 (3/6)

- Usage of a voltage divider:

$$V_{OUT} = V_{IN} * \frac{R_2}{(R_1 + R_2)}$$

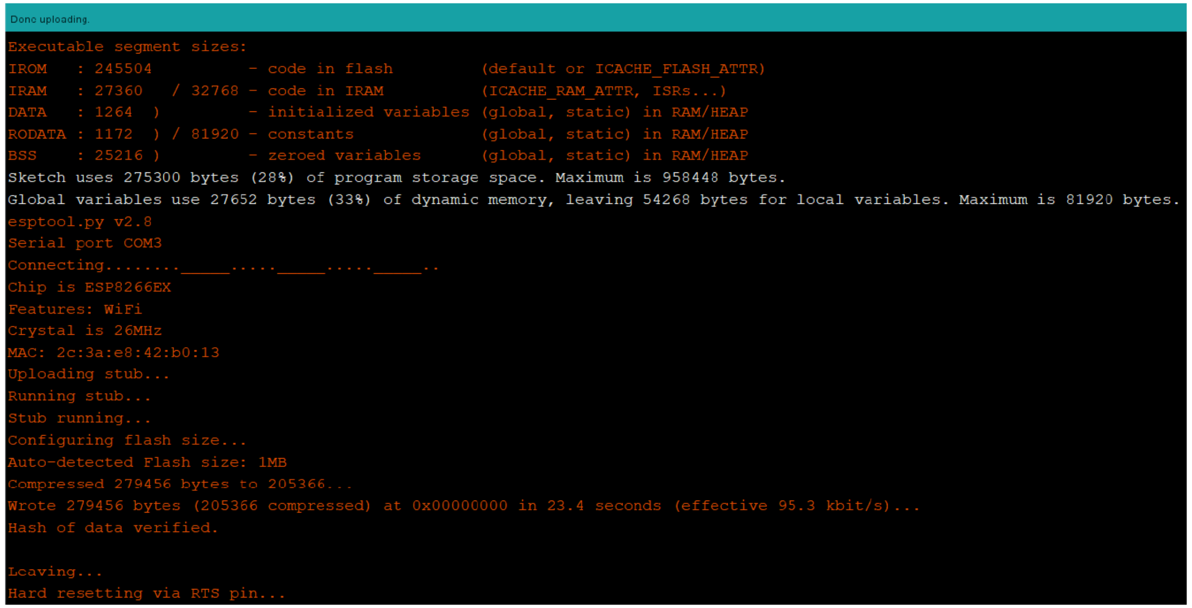
Since the Arduino is giving 3.3 V, the threshold that sets the level to high for the ESP8266 is 2.475 V. Therefore, using a voltage divider, with two resistors of 1 kΩ and 4.7kΩ, the input from Arduino goes from 3.3 V to 2.721 V, sufficient to power the module, but with lower voltage.



First Prototype with ESP8266 (4/6)

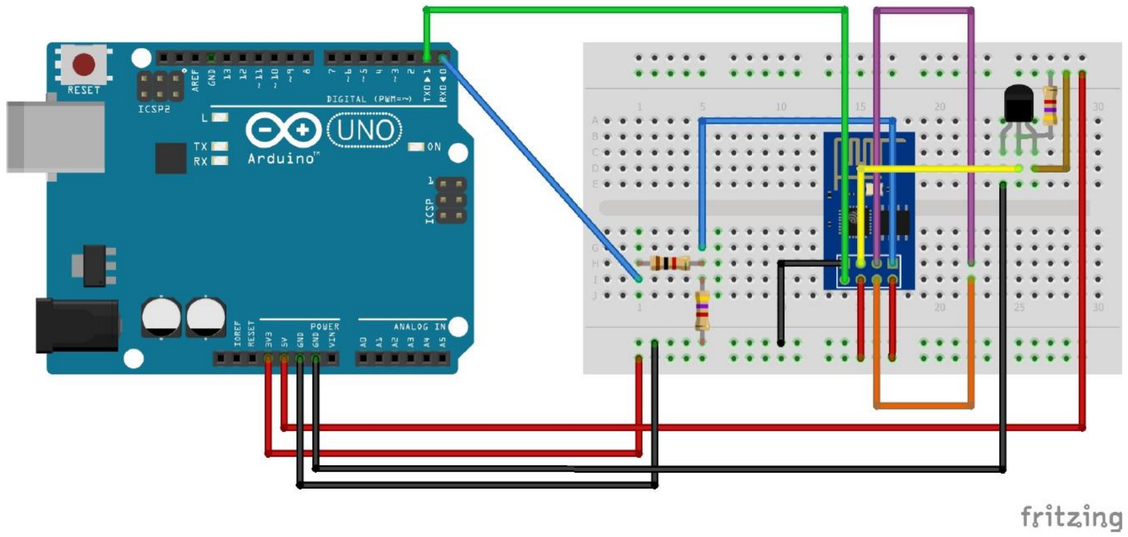
- Load binaries into the ESP8266:
  - Since we are using Arduino only for poweing the ESP module and for a load interface, we need to go to Tools/Board: Generic ESP8266 Module/ESP8266 Boards and select Generic ESP8266 module. If you don’t have these fields, go back to Arduino section and make sure to add the json file in the settings. The last step is to select the proper serial port in Tools/Port.
  - To load binary into the ESP8266 module, click on the Arrow icon in the upper-right corner of the Arduino IDE. Before pressing it, plug both RST and GPIO-0 into GND, to trigger the boot-mode. If compiling is successfull, all the binary and memory information will be printed on the console. Here, if you have already done the flashing process, you will be familiar with the next step. Unplug RST to trigger the reset of the module and the loading should start. Plug and unplug RST if nothing happens. Once the loading is finished, Hard resetting via RTS pin... should be printed on the console.

- An hard reset is needed to start the execution. Before doing it, open Tools/Serial monitor. Make sure to have the same baud as in the source files (it should be 115200). Now, unplug GPIO-0, plug RST into GND and unplug it after a second. Now, if the reset went right, you should see the output in the serial monitor. This process should be only done once. Whenever the Arduino will be powered, the ESP module will try to connect to a Wi-Fi and the starts to send data.



First Prototype with ESP8266 (5/6)

- The DS18B20 is a smart temperature sensor which includes analog-to-digital conversion and provides a binary value as a one-wire transmission on the data pin.
- Considering the flat side as reference:
  - LEFT pin: connected to GND
  - CENTRAL pin: connected to GPIO-2 of the Wi-Fi module and also connected to VCC through a 4.7kΩ resistor
  - RIGHT pin: connected to VCC
- The DS18B20 sensor can be connected to the 5 V of the Arduino board



First Prototype with ESP8266 (6/6)

Sketch of code used to read the temperature data from the sensor and print it to the console.

Standard libraries are used:

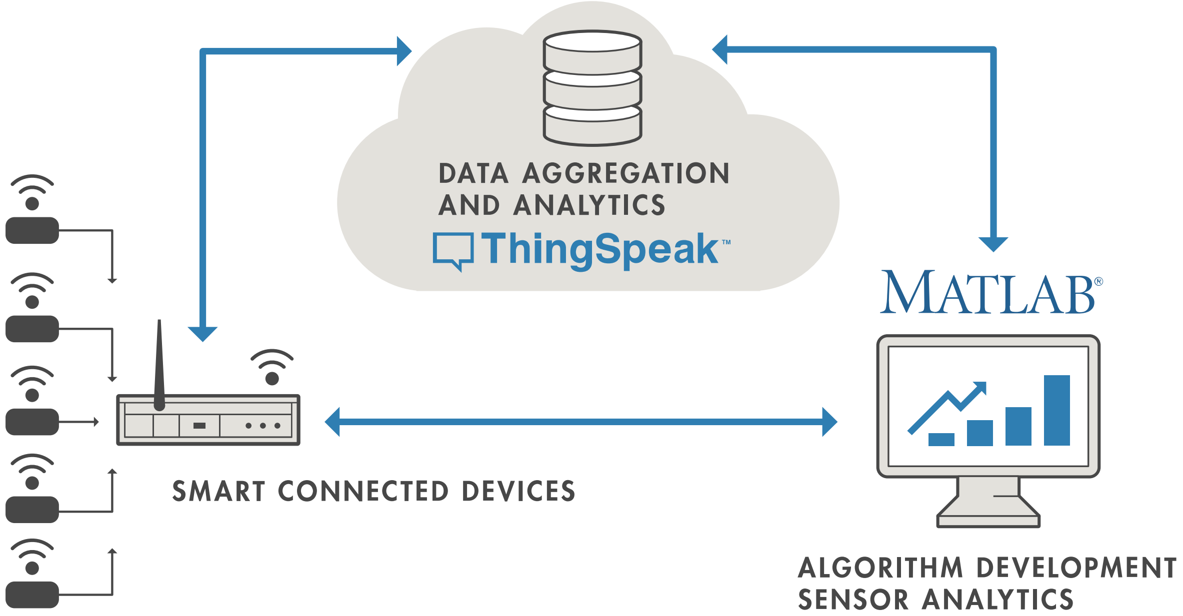
- DallasTemperature
- OneWire

```
#include <DallasTemperature.h>
#include <OneWire.h>

// pin number, this is GPIO-2
#define ONE_WIRE_BUS 2

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup()
{
    // init the serial communication at 9600 baude
    Serial.begin(9600);

    // init the sensors (there could be more than one!)
    sensors.begin();

    // optional
    // default is 9, it goes from 9 to 12
    // increasing the resolution makes the readings
    // more accurate but also slower
    sensors.setResolution(10);
}

void loop()
{
    // request the data from for all possible attached sensors
    sensors.requestTemperatures();

    // retrieve data from first sensor (index 0)
    float temp = sensors.getTempCByIndex(0);

    // sends data to serial port
    Serial.print("Temperature: ");
    Serial.println(temp);
}
```

From data to cloud

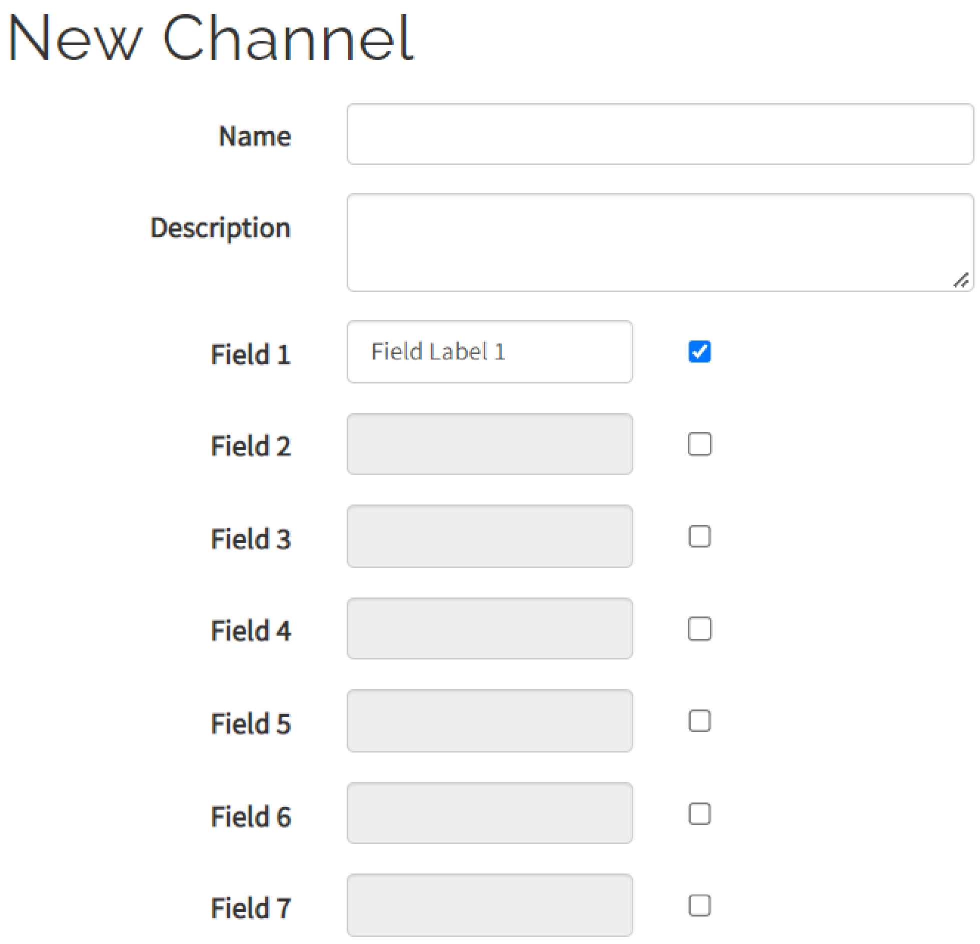
ThingSpeak

- Analytic **IoT platform**
  - **Collect** data from sensors, “things”
    - Through HTTP or MQTT
  - **Visualize** data instantly
  - Support **constrained devices**
    - E.g., ESP32, ESP8266, Raspberry Pi
- **Analyze data**
  - MATLAB integration allows users to run scheduled code on data coming into ThigSpeak
- **Act** on data
  - E.g., send a tweet when the temperature in your backyard reaches 32 degrees



ThingSpeak Channels

- ThingSpeak organizes data into **channels**
    - Allows data **read** and **write** through API Keys
    - Allows to **analyze** and **visualize** data
    - Allows **triggering** actions
        - E.g., device commands



- Each channel is made up of **8 fields**
    - Store 8 streams of data (Temp, Humidity, etc.)
    - Max update frequency limited to 15 seconds

Data Analysis

- ThingSpeak is integrated with **MATLAB** in the Cloud
- Use the **Apps** Tab to use MATLAB inside ThingSpeak
  - Statistics and Machine Learning Toolbox
  - Curve Fitting Toolbox
  - Control System Toolbox
  - Signal Processing Toolbox
  - Neural Network Toolbox
  - DSP System Toolbox
  - Image Processing Toolbox
  - Etc.

Name

Remove outliers from wind speed data 2

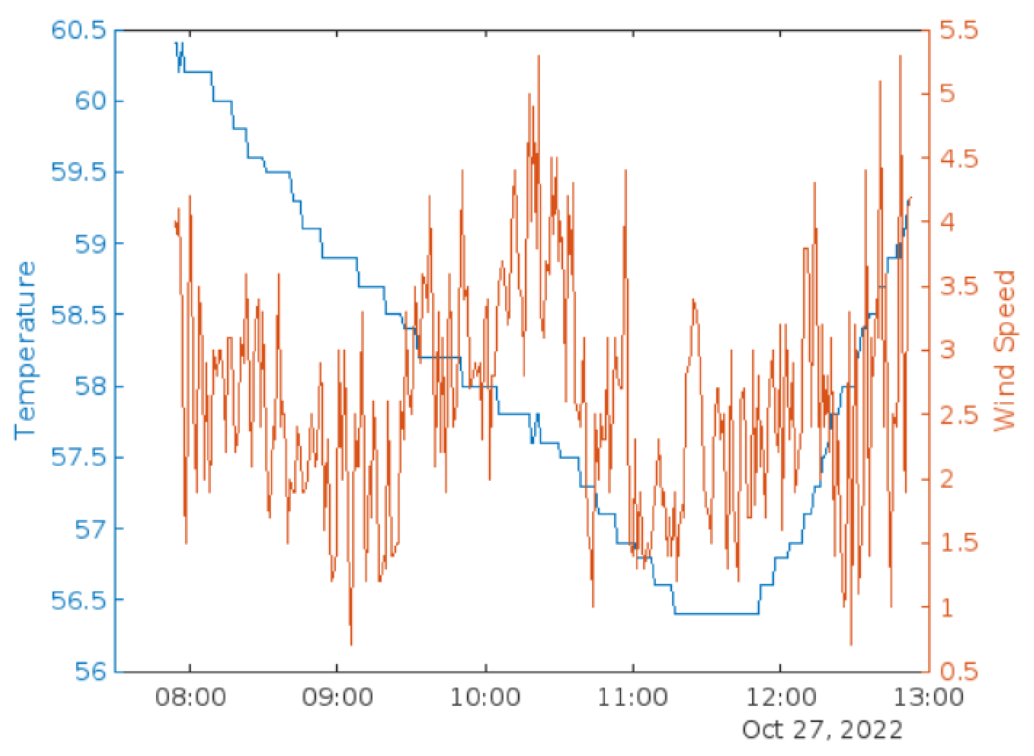
MATLAB Code

```
% Read wind speed data from a ThingSpeak channel and remove outliers, if
% present. Write the cleaned data to another channel for storage.

% Channel 12397 contains data from the MathWorks weather station, located
% in Natick, Massachusetts. The data is collected once every minute. Field
% 2 contains wind speed data.

% Channel ID to read data from
readChannelID = 12397;
% Wind Speed Field ID
windSpeedFieldID = 2;

% Channel Read API Key
% If your channel is private, then enter the read API Key between the '' belo
readAPIKey = '';

% Read the last 6 hours of wind speed data from the MathWorks weather
% station channel. Learn more about the thingSpeakRead function by going to
% the Documentation tab on the right side pane of this page.

[windSpeed,timeStamp] = thingSpeakRead(readChannelID,'fields',windSpeedFieldI
                                        'NumMinutes',360, ...
                                        'ReadKey',readAPIKey);

% We will use the isoutlier function. With the default settings, this
% calculates whether a value is more than three scaled median absolute
% deviations away from the median. (See MATLAB documentation for the
% isoutlier function for more details.) If any such points are found, we
% will delete the point and the corresponding timestamp.

```

Data Visualization

- Data can be viewed and explored using **MATLAB plots**
  - Area plots
  - Line plots
  - Scatter plots

MATLAB Plot Output



- Plots can be **embedded** into any website

Device Interaction

- **TalkBack**
  - Queue a command for a device
- **ThingHTTP**
  - Device communication based on HTTP POST and GET request
- **TimeControl**
  - Execute a ThingSpeak action at a specific time or on a regular schedule
- **ThingTweet**
  - Send device alerts via Twitter

Configuring ThingSpeak

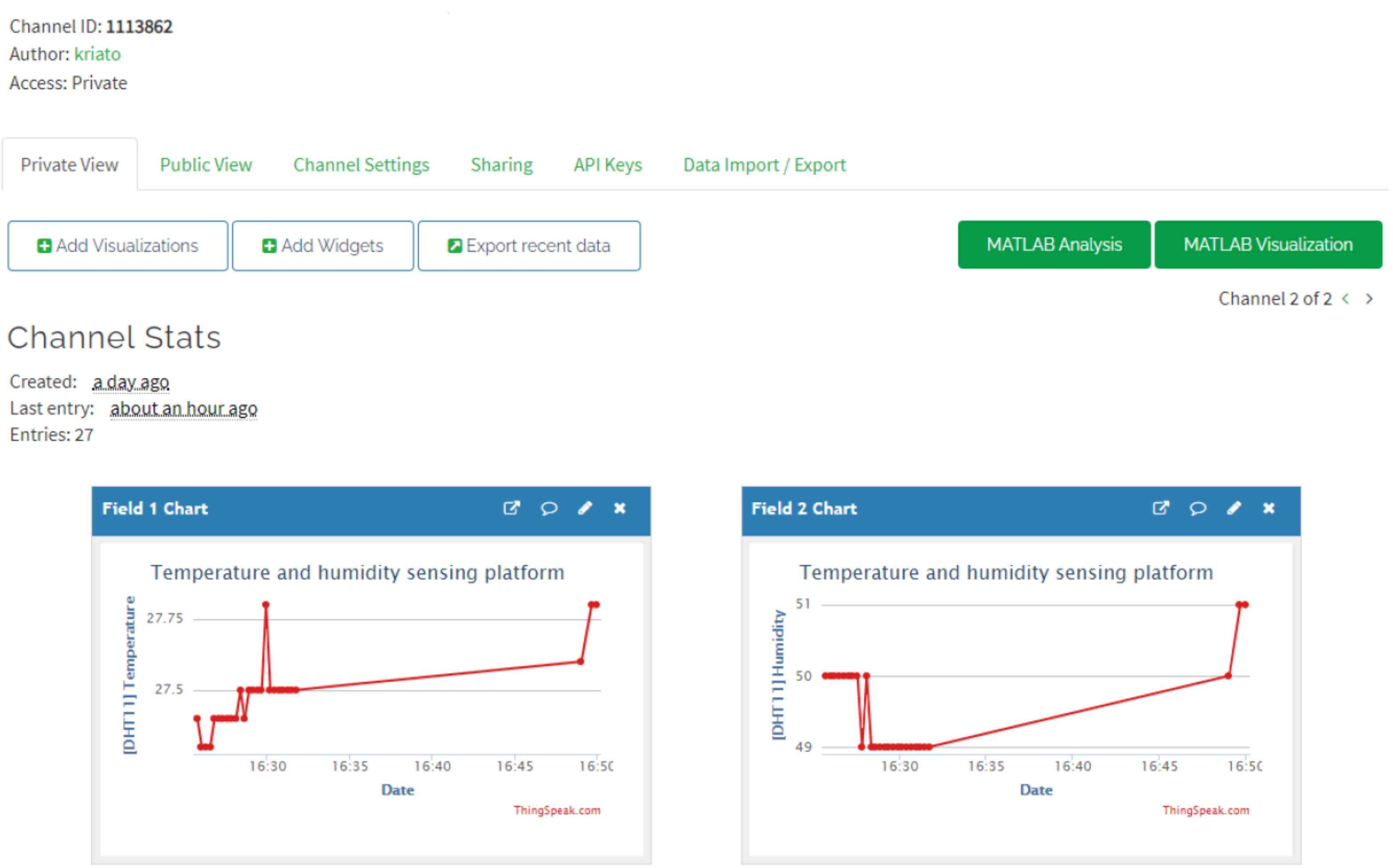
1. Create an account at https://thingspeak.com/

2. Create a new channel
   - Define name, description and its fields:
     - [DHT11] Temperature;
     - [DHT11] Humidity;
     - [DS18B20] Temperature;

3. Save the **API Read/Write** Key and the **Channel ID**

4. Add visualizations/widgets

Configuring ThingSpeak



Code snippets

- Retrieve all data stored in channel and compute the average with standard deviation, the minimum and the maximum for all the fields:

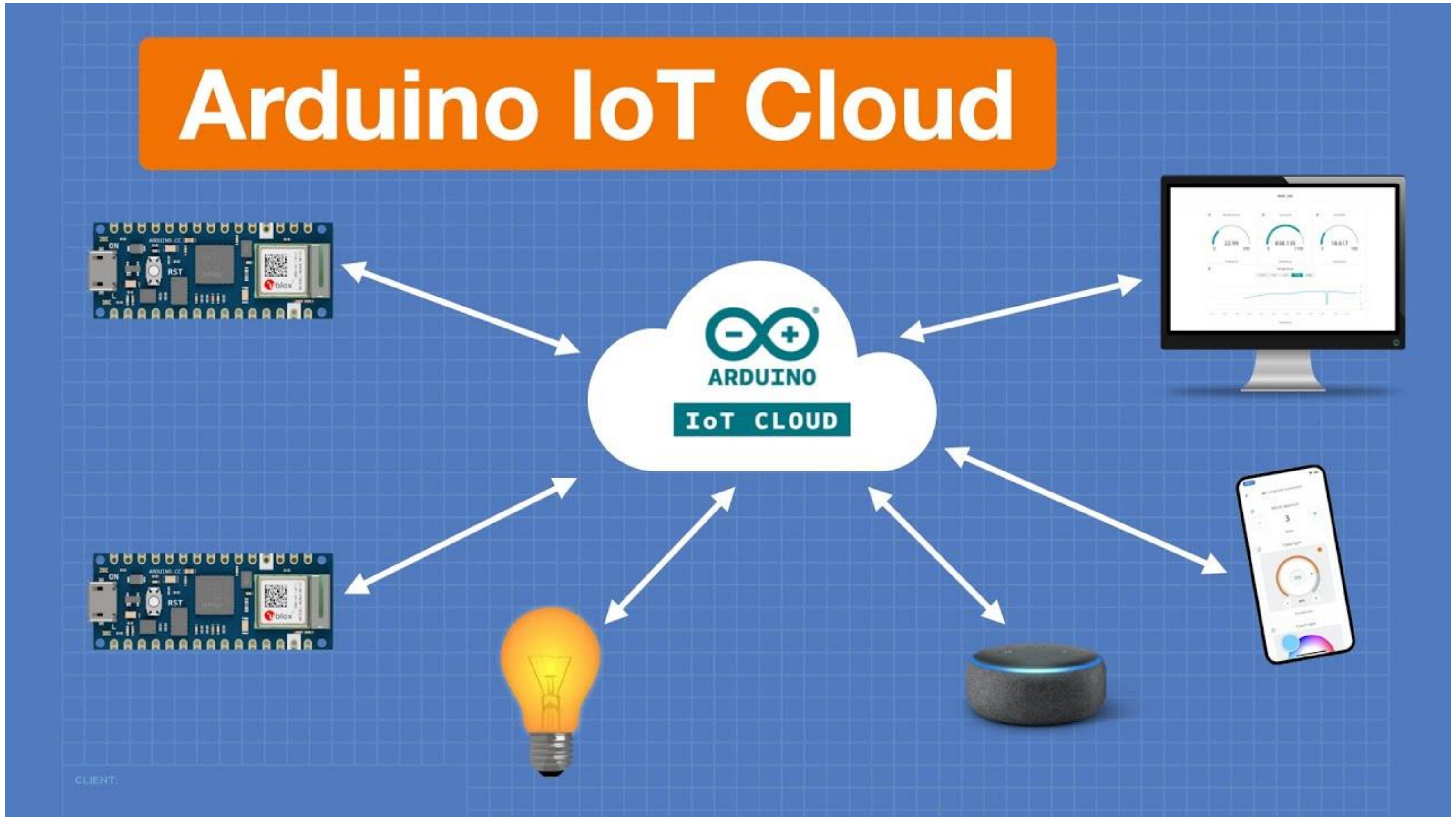
```
chID = 0123456;
tempFieldID_dht11 = 1;
humFieldID_dht11 = 2;
tempFieldID_ds18 = 3;

readAPIKey = 'IN23RTH3R3';

config = 'NumMinutes';
```

Code snippets

```
configValue = 60;

temp_dht11 = thingSpeakRead(chID, 'Fields', tempFieldID_dht11,
        config, configValue,'ReadKey', readAPIKey);
hum_dht11 = thingSpeakRead(chID, 'Fields', humFieldID_dht11,
        config, configValue,'ReadKey', readAPIKey);
temp_ds18 = thingSpeakRead(chID, 'Fields', tempFieldID_ds18,
        config, configValue,'ReadKey', readAPIKey);

avgTemp_dht11 = mean(temp_dht11);
stdTemp_dht11 = std(temp_dht11);
avgHum_dht11 = mean(hum_dht11);
stdHum_dht11 = std(hum_dht11);
avgTemp_ds18 = mean(temp_ds18);
stdTemp_ds18 = std(temp_ds18);

maxTemp_dht11 = max(temp_dht11);
minTemp_dht11 = min(temp_dht11);
maxHum_dht11 = max(hum_dht11);
minHum_dht11 = min(hum_dht11);
maxTemp_ds18 = max(temp_ds18);
minTemp_ds18 = min(temp_ds18);


fprintf('[DHT11] Average temperature: %f with %f std\n',
        avgTemp_dht11, stdTemp_dht11);
fprintf('[DHT11] Average humidity: %f with %f std\n',
        avgHum_dht11, stdHum_dht11);
fprintf('[DS18B20] Average temperature: %f with %f std\n',
        avgTemp_ds18, stdTemp_ds18);

fprintf('[DHT11] Min/Max temperature: %f/%f\n',
        minTemp_dht11, maxTemp_dht11);
fprintf('[DHT11] Min/Max humidity: %f/%f\n',
        minHum_dht11, maxHum_dht11);
fprintf('[DS18B20] Min/Max temperature: %f/%f\n',
        minTemp_ds18, maxTemp_ds18);
```

Alternative Cloud Services

- Arduino Cloud: https://docs.arduino.cc/cloud/iot-cloud

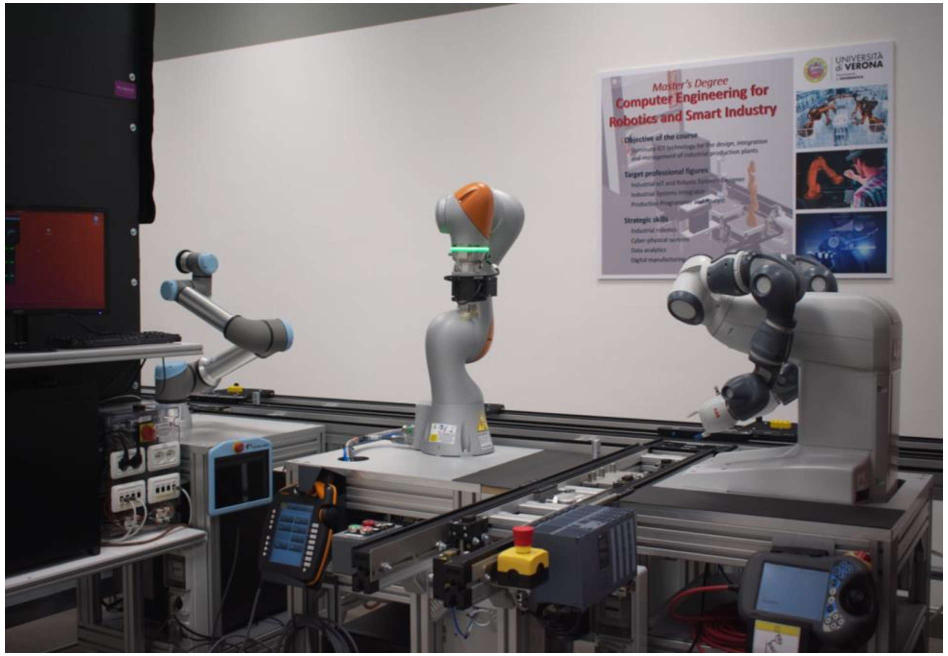


The ICE Laboratory

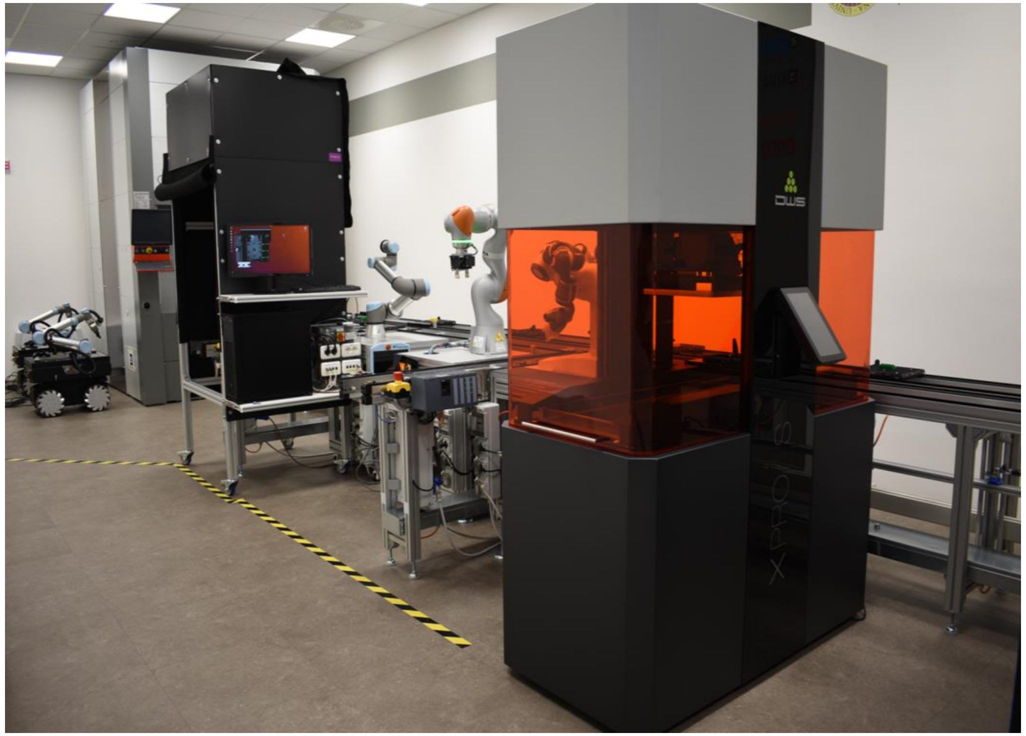
Industrial Computer Engineering Laboratory

ICE Laboratory (1/2)

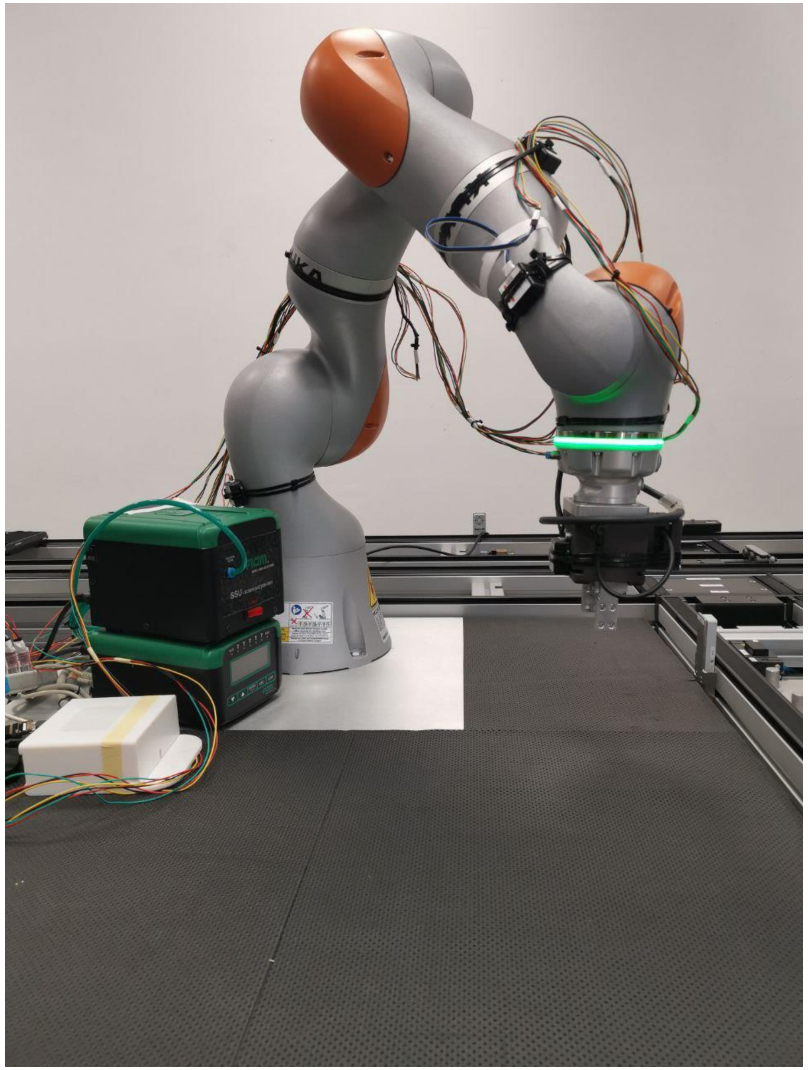
- Applied research, technology transfer, and collaboration with the local area: a production line created according to the principles of Industry 4.0 focused on education, training, and innovation
- Further details: https://www.icelab.di.univr.it/?lang=en

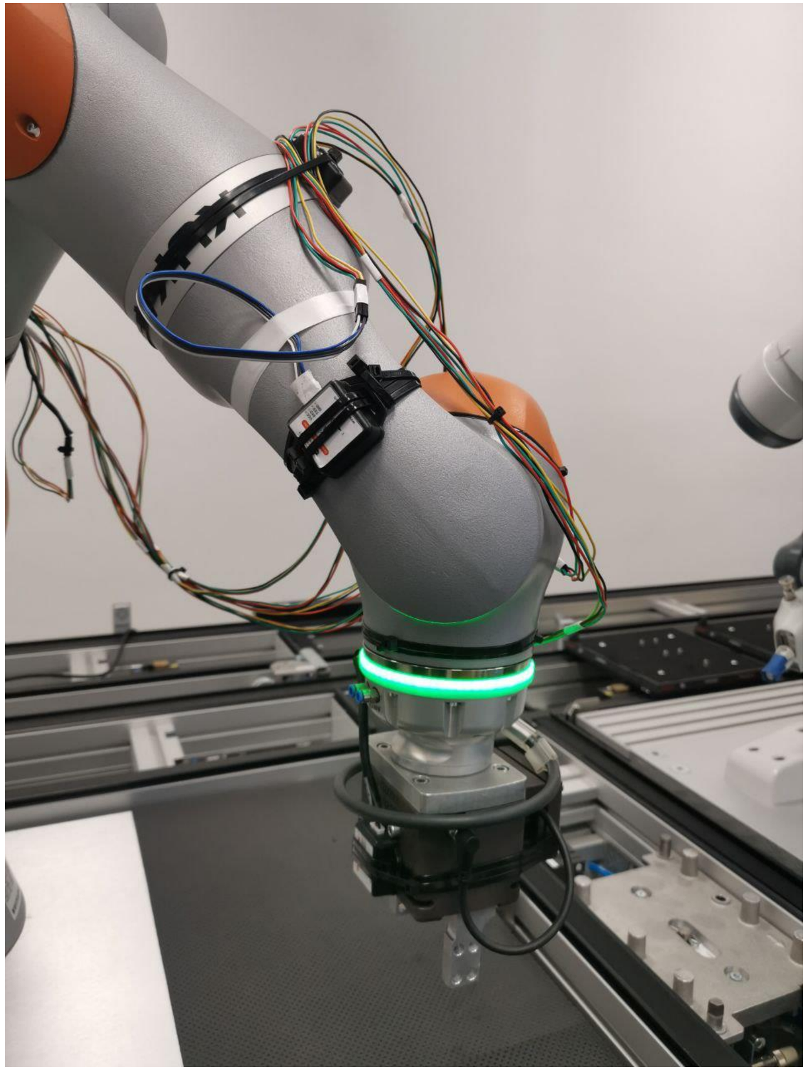


ICE Laboratory (2/2)

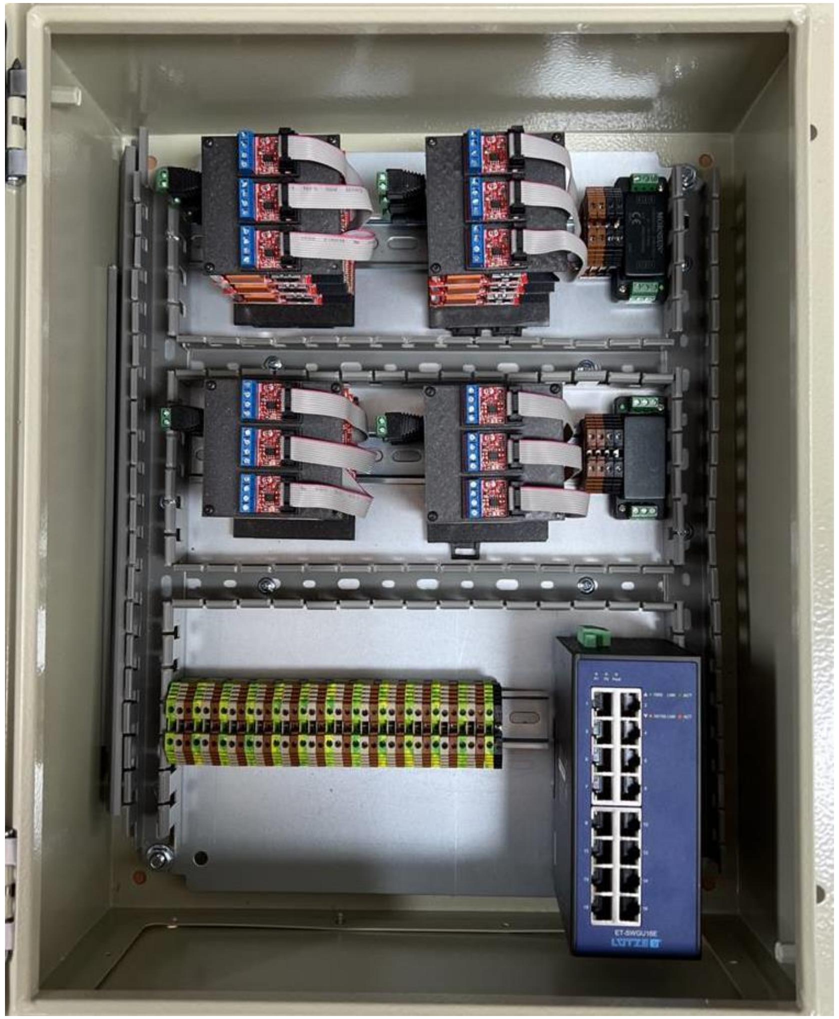


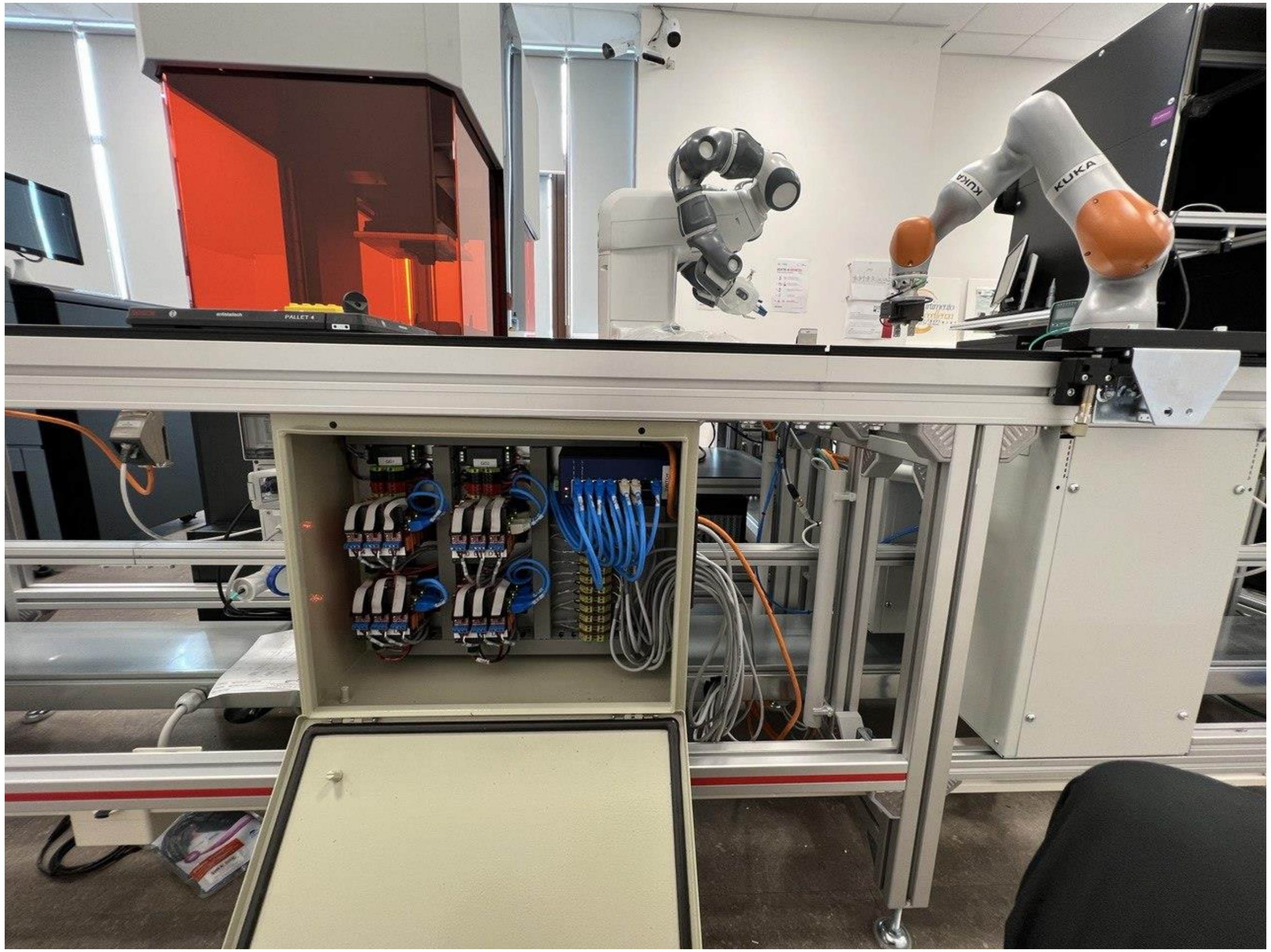
Application of Prototyping in ICE





Application of Prototyping in ICE





Final Project and Report

A.A. 2022/2023

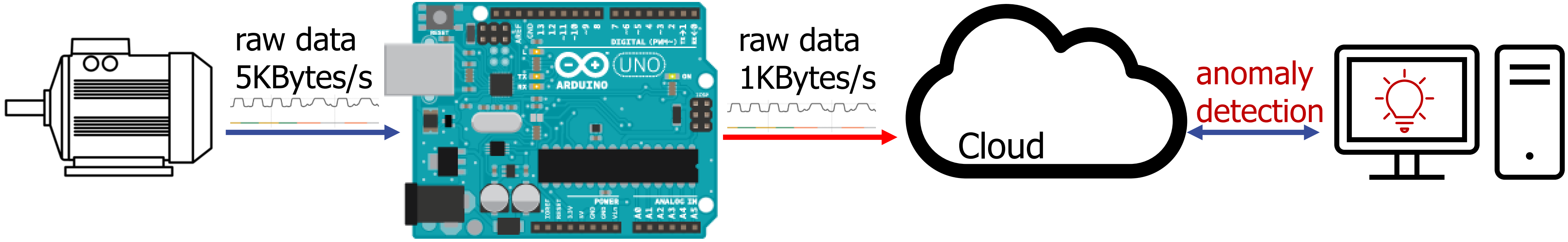
Final Project and Report Details

- Only students that have partecipate in presence to the lectures can present the final project and report of the course!
- The final exam allows to obtain 3 CFU (evaluation expressed in 30)
  - The evaluation is based on the project complexity, report, and exposition;
  - The project details need to be discussed with the teacher before to implementation phase; and also it should be completed in this year (2023);
  - The project could be based on different boards: Arduino, ESP or combination of them connected through serial protocols or wireless protocols;
  - The project could use many sensors/actuators;
  - The boards need to communicate through the web in order to save and process dynamically data coming from the sensors by exploit Cloud services (ex: Arduino Cloud, ThingSpeak Cloud, MQTT + Grafana, Plotly Dash, *etc*.);
  - The report of the project need to be built with Latex (no Word, PowerPoint, *etc*.). Overleaf (https://www.overleaf.com) is a good way to produce Latex documents;
  - The report need to contain all the project informations, including schema of the hardware components drawed with Fritzing;
  - For coding it is possible to use Arduino IDE or VS Code + Platformio extension.

Thesis proposal

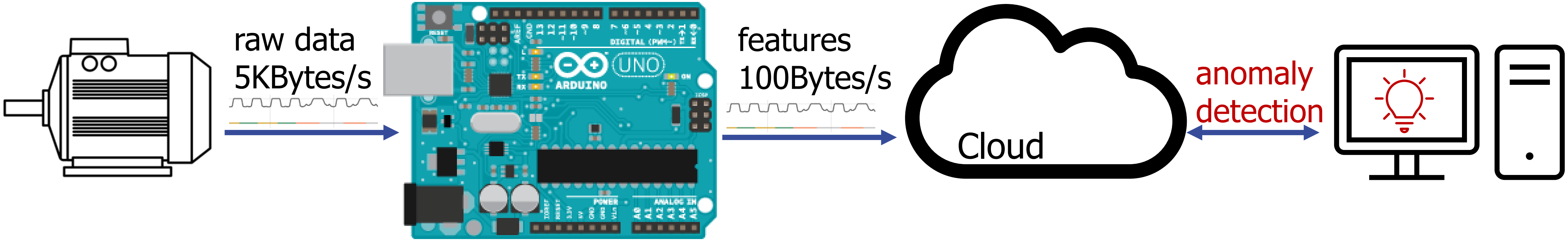
TinyML – Anomaly Detection

- Usually, data are collected and transmitted to the Cloud. In the cloud, raw data are processed to identify anomalies
- Drawbacks:
  - High network usage;
  - Wireless protocols can send a limited quantity of data;
  - Expensive (pay per request/space);
  - Anomalies are identified have a delay;



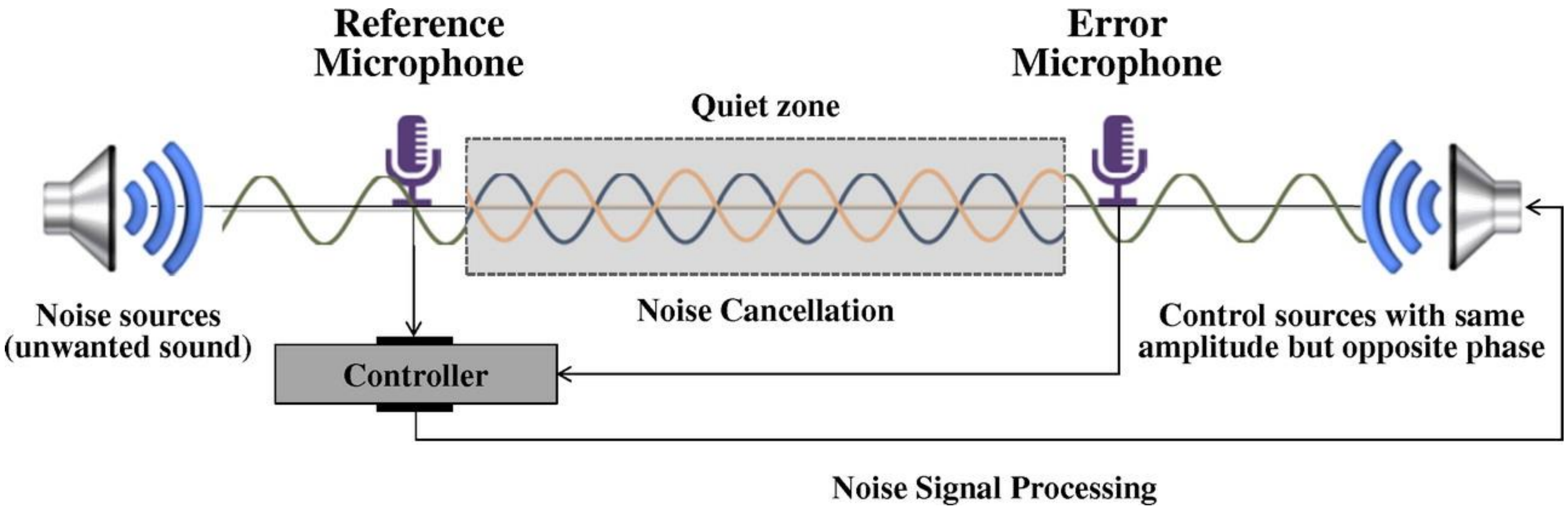
TinyML – Anomaly Detection

- On-device processing to extract features on the device.
  - Transmit few bytes of data to the cloud (features)
- Pipeline
  - Train a ML Model with collected data
  - Optimize & Convert the model
  - Convert to firmware



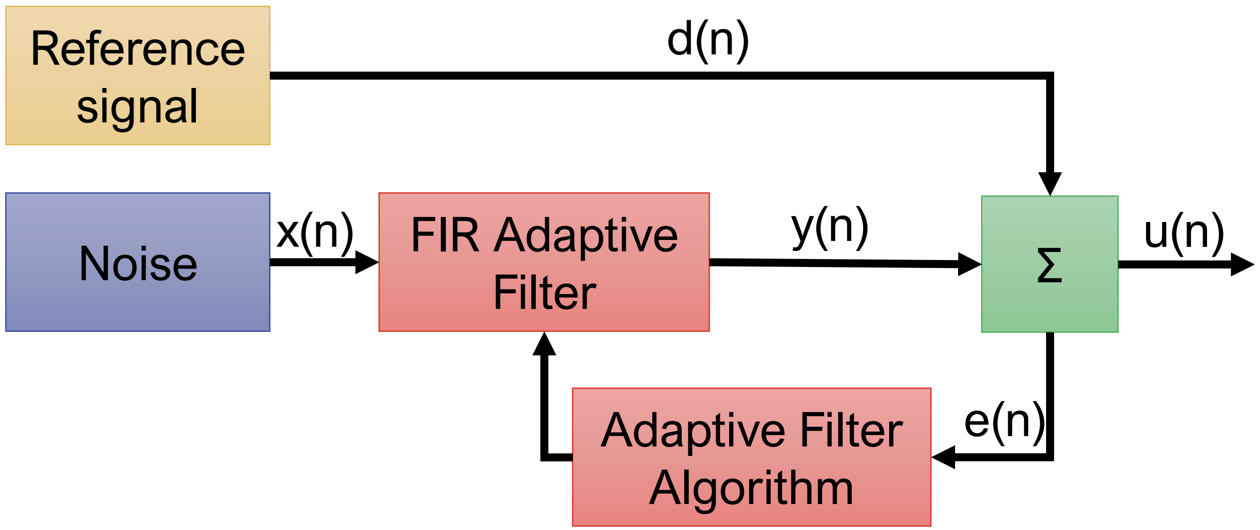
Active noise control

- Active noise control (ANC) has received much attention in recent years. In an ANC system a secondary source is introduced to generate anti-noise with equal amplitude but of opposite phase.



Active noise control

- The controller implements an adaptive filter which adapts itself to the input signal given to it.



$$y(n) = \sum_{k=0}^{N} w_k(n) * x(n-k)$$

Cyber-Physical & IoT System Design Group (CISD)

- CISD consortium is a cluster of research groups at the Dept. of Engineering for Innovation Medicine, University of Verona.
- CISD aims to address, through multi-domain knowledge, the complexity of the design and verification in Cyber-physical and IoT Systems applied to different contexts, from the Industrial IoT (IIoT), agriculture, to healthcare.
- Thesis/stages proposals: https://cisd.di.univr.it/thesis/available/