POLITECNICO
MILANO 1863

SCUOLA DELL'INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

COMPUTER SCIENCE AND ENGINEERING

# Internet Of Things

# 1st Challenge

**Simone Pio Bottaro 10774229**

**Gabriele Lorenzetti 10730455**

**A.A 2024/2025**

# Introduction

The Wokwi project is based on an ESP32 board connected to an HC-SR04 ultrasonic distance sensor, which is used to measure occupancy of a parking spot. The board goes in a deep-sleep state every X seconds and then, it wakes up to calculate and notify the occupancy to a sink node, after this, the device goes again into deep-sleep waiting for the next cycle.

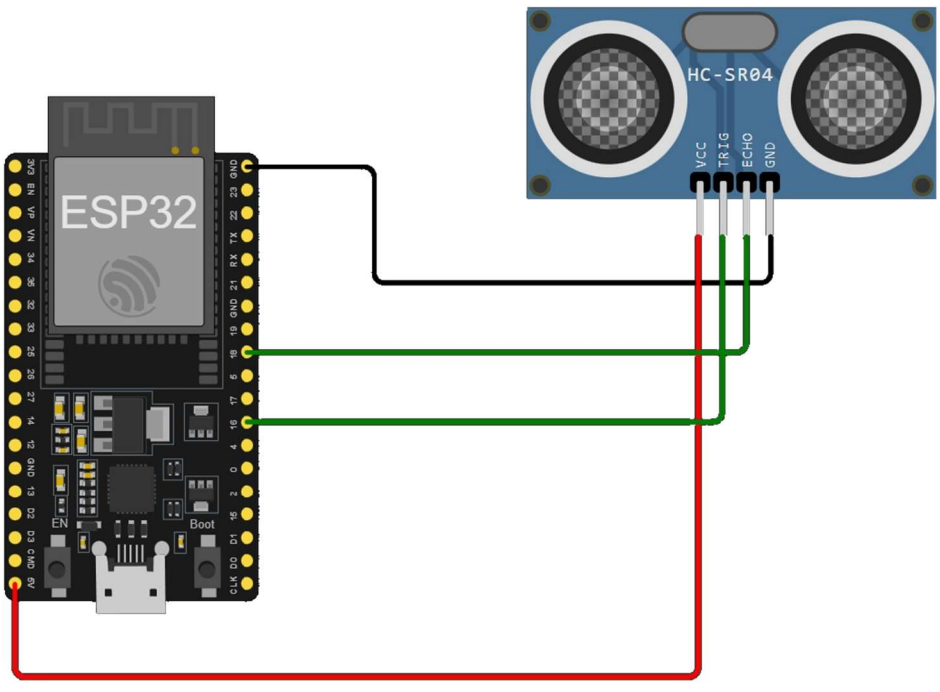
The communication is effectuated due Wi-Fi and ESP-NOW protocol.

The period of activity of the board can be summarized into the following steps:

1. **SETUP (idle)**: Configuration of the board and of the pins for the sensor (HC-SR04)
2. **MISURATION**: Using the sensor, the device starts a new distance measurement, the distance is compared with a distance reference of 50 cm. If the measurement is smaller or equal than the reference, will be notified to the sink the message "OCCUPIED", in the other cases, will be notified "FREE"
3. **WI-FI ON:** In this phase, Wi-Fi and ESP-NOW protocol are activated, preparing the board to connect itself to the sink
4. **TRANSMISSION**: The message previously generated is sent to the sink
5. **WIFI-OFF:** The Wi-Fi is turned off
6. **DEEP-SLEEP:** The board goes in Deep-Sleep for X seconds. The value X is calculated as requested using the formula: (*last two digits of leader person code*) % 50 +5. In our case the value obtained is: 29 % 50 + 5 = **34 seconds**

The transmission is effectuated using a Wi-Fi power of 2dB in order to preserve energy. For the simulation the board has been considered as sink node itself to visualize the message from the sink view. For this reason, a delay of 1 second has been include in the code before turning off the Wi-Fi.

The code can be found in the folder uploaded, written using VS Code editor, or following this link can be found the WokWi project: https://wokwi.com/projects/425852753337185281




**ESP-32 connected to HC-SR04 sensor**

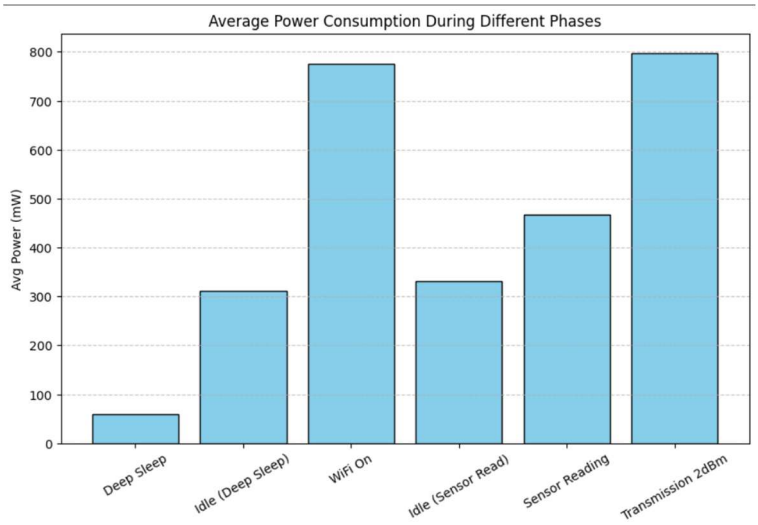
# Energy Consumption

In this section will be analysed all the aspects relative to the power and energy consumption.

## Power Estimation

First of all, have been analysed the csv files provided to calculate the average power consumption for each phase.

In the figure, we can see the plot generated from the analysis. To see the various passages, the details are inside the Jupyter Notebook uploaded.




Here are reported, for each phase, all the average power values calculated:

- **DEEP-SLEEP**: 59.661 mW
- **IDLE**: 311.672 mW
- **WIFI ON**: 775.489 mW
- **MISURATION**: 466.745 mW
- **TRANSMISSION**: 797.294 mW

## Time Estimation

For the time estimation, have been included around the code some print to understand when each phase has been concluded. To this porpoise was used the function ***micros()*** in order to receive the time spent, expressed in microsecond, during each phase. Due the difficulties of calculating the correct time using the WokWi simulator, we generated a csv file containing all the printed times of some simulation. Then, we chose the most frequent time printed for each phase. The followed passages are also present the Jupyter Notebook uploaded.

Here are reported the time calculated:

- **BOOT TIME**: 882 microseconds
- **MISURATION TIME**: 7355 microseconds
- **WIFI ON TIME**: 190526 microseconds

**Note**:
The times reported are referred to the time instants, from the board boot, when the phases have been executed.

- **MESSAGE SENT TIME**: 1190065 microseconds
- **WIFI OFF TIME**: 1196262 microseconds

## Energy Estimation and system lifetime

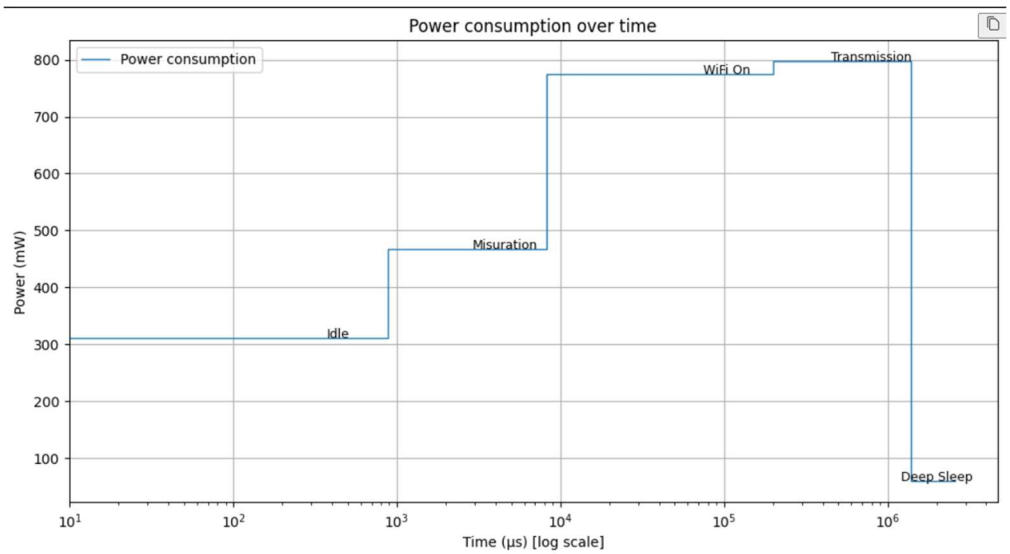
After the previous phases we finally was able to calculate the energy consumption for each phase.
The function *micros()* returned the times instant from the board initialization. To calculate the time of each phase, an estimate has been made calculating the difference between a print with the previous one.
Also these passages are present inside the Jupyter Notebook uploaded.

Considering the big difference between the deep-sleep of 34 seconds and the other phases effectuated in times of microseconds, the most expensive phase is obviously the deep-sleep.

Here are reported the energies calculated with a plot printed using a logarithmic scale in order to see the difference between the various phases:

- **DEEP SLEEP:** 2.028474 Joule
- **IDLE:** 0.00027489470400000003 Joule
- **WIFI ON:** 0.7799372049040001 Joule
- **TRANSMISSION:** 0.796926447466 Joule

**Total energy in a cycle**: 3.6086337874590004 Joule




The **Battery capacity** has been calculated using the provided formula:

(*last four digits of leader person code*) % 5000 + 15000, in our case,

4229%5000+15000 = **19229 Joule**

With these values, we can expect that the system will works for, approximately, ***5328 cycles*** with a total lifetime of ***52 hours***

# Possible Improvements

Eventually considering a real scenario, since we are monitoring a parking spot, we could prolongate the Deep-Sleep state since is itself the most energy efficient state.

We could propose two different options of improvement:

1. Longer intervals such as 60 seconds may be enough to register any change in parking occupancy. In this case, recalculating the energy consumption of a cycle and considering the total number of System Lifetime hour we will have a durance of approximately of **90 hours**.

   The passages followed are showed in the last part of the Jupyter Notebook

2. Considering also that the most expensive phase is registered when the Wi-Fi is turned ON and the transmission is going, we could consider a different approach, such as ***turning on the connection only in cases where the state of the spot changes.***

   To do this, we should consider a different approach of sleep. The Deep-Sleep, in fact, deletes all the previous variable stored. Using a **light-sleep** we could store the previous state and use it to determine if the Wi-Fi must be turned on to send the message to the sink node.

   A prototype of this version can be found in the zip file uploaded.