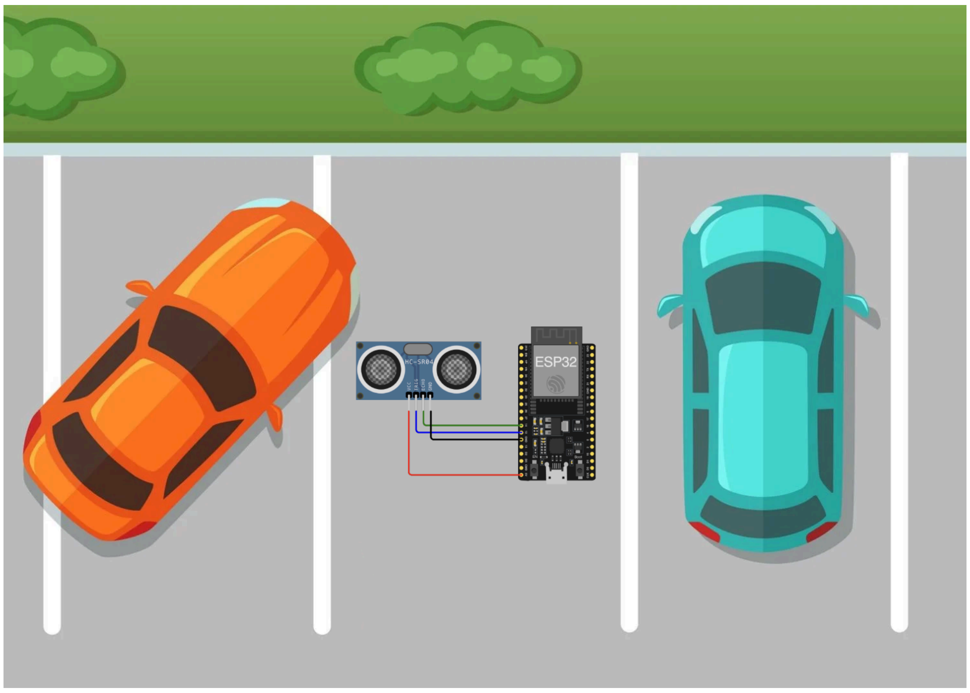
First Challenge - Internet Of Things

Professor: Redondi Alessandro - Academic Year: 2024/2025

Pianalto Riccardo (Person Code: 10835980)
Pica Mirko (Person Code: 10811404)

1. Code explanation

We were asked to develop on Wokwi a simple parking occupancy node using the ESP32 board, the HC-SR04 ultrasonic distance sensor and the ESP-NOW for communication. The node needed to find out the status of the parking spot FREE/OCCUPIED and communicate it to the central sink node.



We immediately defined the steps needed:

1) Read the value of the ultrasonic sensor (if distance is less than 50 cm then OCCUPIED, FREE otherwise);
2) Turn on WiFi, send the message to the sink node and turn off WiFi;
3) Going into deep sleep state for 9 seconds.

The code is pretty straight forward but a few clarifications are needed:

a) The deep sleep state duration was calculated using the given formula $04\%50 + 5 = 9$, where 04 are the last two digits of the leader person code;
b) We introduced $COLLECTING\_DATA$ as a way to get timestamps only when needed, this was used for the next points computations;
c) We send the message to the broadcast address so we could also receive the message for testing since in Wokwi is not possible to simulate multiple nodes;

d) We set the transmission power to 2 dBm instead of leaving it to the default value of 19.5 dBm to lower energy consumption, since we were told to assume the sink node is always reachable by every sensor node.

2. Energy computations and plots

2.1 Average power consumption in each stage

Using a python script and referring to the provided power consumption CSV files we calculated the needed values by selecting the right samples and averaging them.

This are the average values we got:

a) Deep sleep state: $59.66\, mW$
b) Idle state: $320.91\, mW$
c) Sensor reading state: $466.74\, mW$
d) WiFi state: $724.58\, mW$
e) Transmission state at 2dBm: $797.29\, mW$
f) Transmission state at 19.5dBm: $1221.76\, mW$

As expected the less power hungry state is the deep sleep, where the node will spend most of his lifetime.

2.2 Average time spent in each stage of the node

As anticipated, to estimate the time spent in each state we introduced timestamps (using the $micros()$ function) in various points of the code marking state changes.

We identified this state order:

1) Idle
2) Sensor Reading
3) Idle
4) WiFi On
5) Transmission
6) Wifi On
7) Idle
8) Deep Sleep

We already had the deep sleep duration so we needed to estimate the others. As suggested by the teaching assistant, we did various code runs to get the most accurate result and removed very skewed values.
We ended up with forty measurements, half with the parking spot occupied and half with it free. We decided this because of the operation of the ultrasonic sensor, which would be active for longer if the car park was free, and so we wanted an average.

This are the average values we got:

a) Idle state: $777.48\, \mu s$
b) Sensor reading state: $13720.3\, \mu s$

c) WiFi state: $189403.07\ \mu s$
d) Transmission state: $180.1\ \mu s$

For a final cycle duration of $9204.08\ ms$ of which $9\ s$ are in deep sleep state.

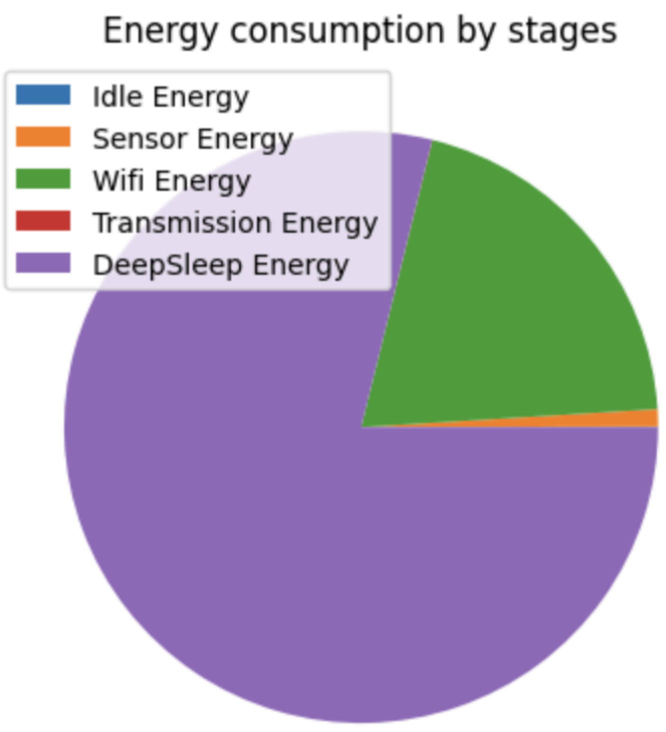
2.3 Average energy consumption of a transmission cycle and battery duration

Using the values calculated in the previous sections we calculated energy consumption of each state.

The values we got:

a) Deep sleep state: $536.95\ mJ$
b) Idle state: $0.25\ mJ$
c) Sensor reading state: $6.40\ mJ$
d) WiFi state: $137.24\ mJ$
e) Transmission state: $0.14\ mJ$

For a final energy consumption of one transmission cycle of $680.98\ mJ$.



As we can see from the graph, the deep sleep state consumes the most energy, followed by the wifi state and sensor reading state, transmission and idle states are negligible.

We were given a battery of $16404\ J$, calculated using the given formula $1404\%5000\ +\ 15000$, where 1404 are the last four digits of the leader person code. With this battery the node is able to do 24088 full cycles which translates to around two and a half days of lifetime (for more accurate data refer to the jupyter notebook).

3. Possible improvements to reduce the energy consumption

Looking at the energy consumption graph we can see that idle and transmission state are irrelevant so we look for improvements in other areas.

3.1 Sensor

We could try to swap it with a lower energy consumption one (e.g. an ultra low power proximity sensor) to minimize the sensor energy state. Or we could try to use an activation sensor to wake the node from deep sleep to do computation only when there is a change in the parking spot.

3.2 Transmision

WiFi is not really energy efficient so our first thought went to using another type of transmission (e.g. ZigBee, Z-Wave) but since we have seen only WiFi in class we decided to look for other improvements.
For example, we noticed that it is useless to send messages when the state of the parking doesn't change. Adding a variable in the code to keep track of the last sent message we could turn on wifi and transmit only when the state changes. We did some calculations assuming that fifty percent of the time there is a state change and we got six more hours of battery life (for more accurate data refer to the jupyter notebook).

3.3 Deep Sleep

Checking the status of the parking every nine seconds doesn’t make much sense because we assume that the state of a parking spot won't change in that time frame. To improve this state we could change the deep sleep duration making it longer or even dynamic, for example supposing that once a car parks it will stay there for at least an hour.
We did some calculations with longer sleep duration and obtained longer lifetime, with the more interesting result being ten minutes deep sleep where we obtained fifteen hours more (for more accurate data refer to the jupyter notebook).

Attached we leave the Jupyter Notebook