REPORT IoT CHALLENGE 1

VALENTINA VILLA 10705748,

SIMONE DI IENNO 10938038,

ALEX SPAGNI 10922295

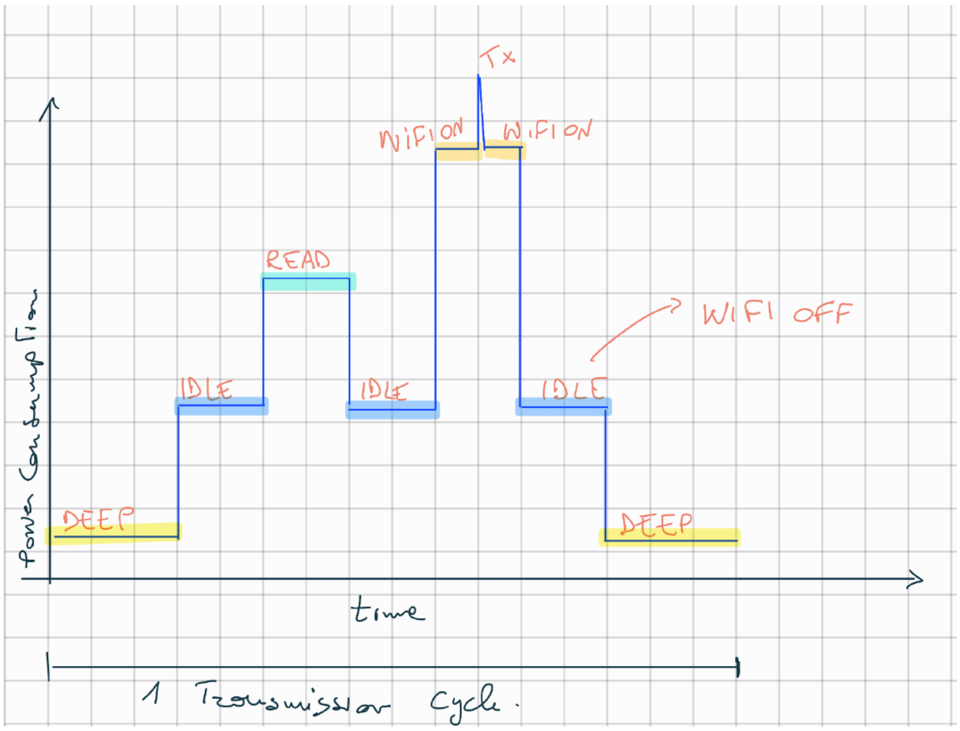
LINK PROGETTO WOKWI: https://wokwi.com/projects/392529887976911873

1) Code logic

To implement and start the system, we used an ESP-32 connected to an Ultrasonic sensor to detect the distance of a car from the sensor. We connected the TRIG pin of the sensor to pin 12 of the ESP-32 in OUTPUT mode, while the ECHO pin was connected to pin 14 of the ESP-32 in INPUT mode. Subsequently, we configured the ESP_NOW Wi-Fi, setting the address to Broadcast. Then the ESP-32 initiates measurement via the sensor using the TRIG pin and reads the result using the ECHO pin. At this point, by calculating the distance as duration/58 (cm) and based on its value, the ESP-32 will send a message about the parking status, either OCCUPIED or FREE. Finally, it waits for the message reception using a delay(100) before turning off the Wifi and entering deep sleep mode for X seconds. The wait could potentially be managed with a boolean variable; however, the issue encountered is that if the transmission fails, the ESP-32 would remain stuck checking the variable without entering deep sleep.
To correctly display the duration time of the various states (Idle, Read, Send, Wi-Fi ON), we have used the millis() function, which simply prints the timestamp of the running program.

2) Power and energy consumption

Regarding the considerations in point 1, based on the data from the .CSV files, we observed that they described the instantaneous power consumption values for each system state. In particular, building upon the implementation in point 1, we proposed manually sketching a graph representing the power consumption values within a single cycle. Subsequently, we calculated the average values for each state. Below is a draft of the graph for reference.



Specifically, at the beginning of a generic cycle, we consider the system in deep sleep mode, then it wakes up and transitions to an idle condition. While the ESP-32 is idle, the sensor reads the values (read sensor state), and then the wifi is activated (wifi ON state). At this point, the ESP-32 is ready for value transmission (TX state), before returning to the wifi ON state. In our case, in the implementation of point 1, before returning to deep sleep, the wifi is turned off, so the system returns to an idle situation.

For the calculation of the average power consumption, we considered the values from the CSV file (related to the specific state) and calculated the average using Excel directly (in particular, as we considered the printing of the durations of the various states in exercise 1, the power consumption of the read sensor was calculated as the difference between the average of its values and the average of the idle values; the same applies to the wifi on state, whereas for the TX state, we calculated the difference between the average of TX and the average of the wifi on values). As for energy consumption, we also had to calculate the "duration" (basing on timestamps printed through exercise 1) of the states within the generic transmission cycle under consideration. For deep sleep, we referred to the implementation in point 1 (in a cycle, the ESP-32 is asleep for 53 seconds (code id MOD 50 + 5)), while for all other states, we calculated the duration simply by taking the difference between the final and initial timestamps printed. Then, we calculated the energy consumption for each individual state (deep sleep, idle, reading sensor, wifi on, tx) as $E=P*\Delta t$ and subsequently summed all these values.

	Deep sleep	Idle	Read sensor	Wifi on	Tx
Power consumption (mW)	59.62438	311.2316	132.9885=(AvgRead - AvgIdle)	464.2579= (AvgWiFiOn - AvgIdle)	514.3971= (AvgTX - AvgWiFiOn)
Duration (s)	53	0.211	0.025	0.183	0.001
Energy (mJ)	3160.092	65.66986	3.324713	84.96	0.514397

Etot = 3314.56097 mJ

Finally, for calculating the number of cycles that can be performed with a battery, we simply divided the battery's energy by the energy calculated above.

Given that we have 5748+5 = 5753 Joule, we can do 1735.6753 full cycles.

3) Possible improvements

For what concern the improvements necessary to reduce the power consumption while maintaining the same functionalities we thought about different possibilities:

- At first, we could change dynamically the transmission power based on the distance of the sink node. This can be done using the following line of code: WiFi.setTxPower()
- Reducing the sample rate: We thought that we could reduce the frequency at which sensors send readings to the sink node since cars do not move as frequently in real case scenario. This means that the sensor will stay in the deep sleep state for longer period of time allowing us to save energy.
- Reduce the amount data transmitted: instead of sending a string FREE/OCCUPIED we could establish a standard where 0 means OCCUPIED and 1 means FREE. In this way less byte will be sent over the channel and slightly less power is needed.
- Use of low-energy components: use of sensors and components that consume as least energy possible