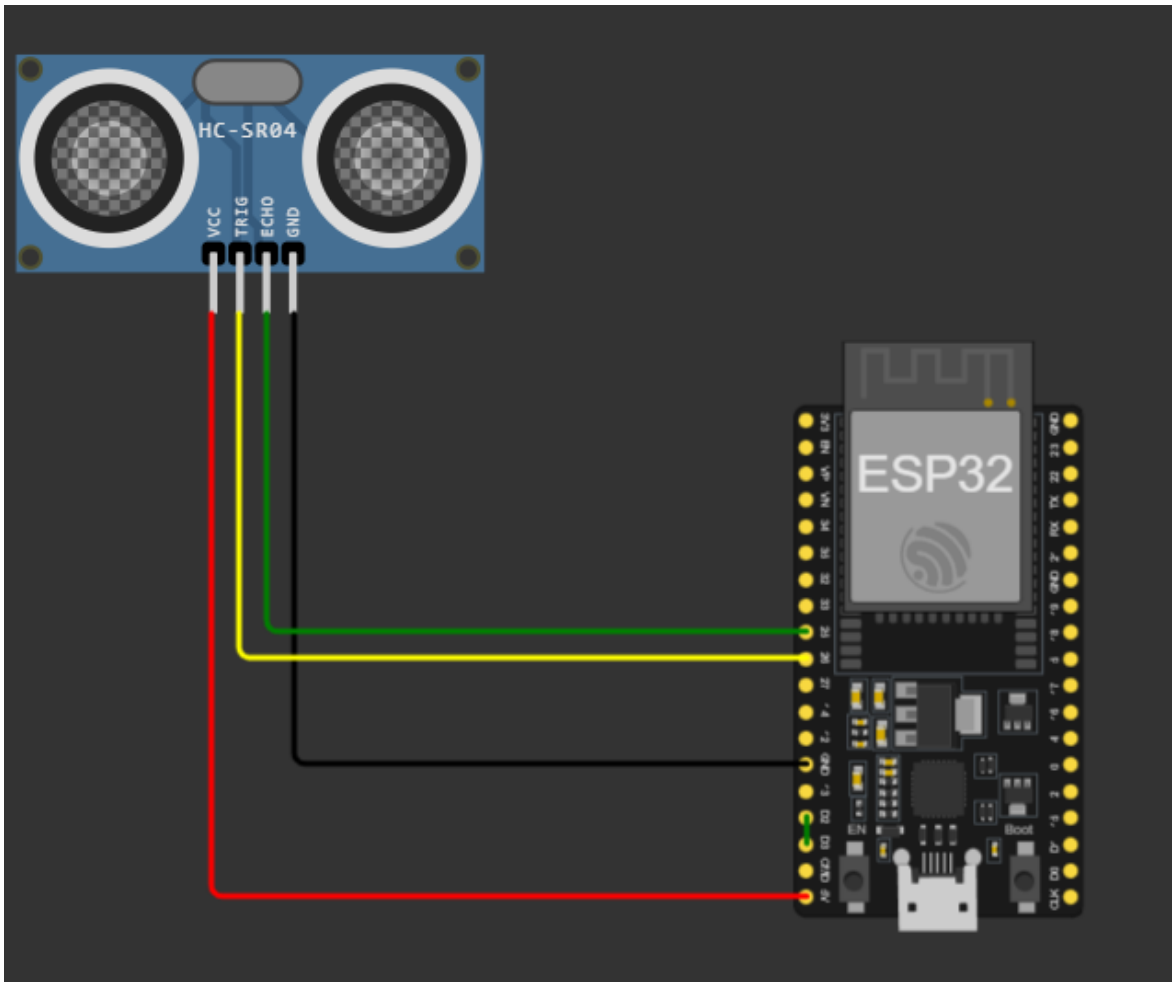
IOT_project1

Briscini matteo [10709075]

INDEX:

Requirements summary



The goal of the project is to implement parking occupancy node, on an ESP32 connected with the HC-SR04 ultrasonic distance sensor. using esp_now for wifi communication purposes, and a battery as an alimentation supply.

> **note:** the parking is considered OCCUPIED if the measured distance is less than 50cm

Implementation

The node is battery powered so minimizing battery usage is critical, in order to minimizing power consumption the sensor stays in deep sleep for the largest part of its duty cycle. The wake-up timer is set to 30 seconds, after the wake the sensor will perform the following operations in the specified order:

1. perform a sensor data measurement
2. turn on wifi
3. send sensor reading (& wait for eventual response)
4. turn of wifi
5. set the EPS32 in deep sleep mode

> **note:** the whole implementation is provided here

Perform a sensor data measurement

The HC-SR04 ultrasonic distance sensor 4 pins:

- **VCC** voltage supply 5v
- **GND** ground
- **TRIG** require a puls of 10uS to start the measure
- **ECHO** to get the distance

> **note:** the distance is retrieved in terms of pulse durations (microseconds) the following formula is used to compute the distance in centimeters.
>
> $$D_{cm} = T_{pulseduration} \cdot 0.034/2$$

As specified in the documentation this sensor can read distances between 2 cm - 450cm. Following is provided the function that performs the sensor measurements and the data structure used to save the results.

```
String sensorDataRead(int sensorInput, int sensorData){
 digitalWrite(sensorInput, LOW);   //for safety reasons
 delayMicroseconds(2);

 //start sensors reading
 digitalWrite(sensorInput, HIGH);
 delayMicroseconds(10);
 digitalWrite(sensorInput, LOW);

 //read
 int read = (pulseIn(sensorData, HIGH)*0.034/ 2)<limitDistance;
 state = (ParkingSpotState)read;

 return ParkingSpotStat_STRING[state];
}
```

```
#define GENERATE_STRING(STRING) #STRING,
#define FOREACH_PARKING(PARK) \
        PARK(FREE)   \
        PARK(OCCUPIED)  \

enum ParkingSpotState{
  FREE = 0,
  OCCUPIED = 1
};
static const char *ParkingSpotStat_STRING[] = {
    FOREACH_PARKING(GENERATE_STRING)
};
```

Send info on the parking state

In this phase, the node turns on the wifi, sends the message (FREE or OCCUPIED) to the specified MAC address (throw esp now protocol), waits for 2000 microseconds (to avoid communication error and to get an eventual response), and turns off the wifi again.

> **note:** in reality, the two last operations are implemented inside the goToDeepSlee; function they are reported ear for clearness.

Following are provided all the functions concern this phase:

```
void wifiSetup(){
 //wifi setup
 WiFi.mode(WIFI_STA);
 esp_now_init();
 esp_now_register_send_cb(onDataSent);
 esp_now_register_recv_cb(onDataRecv);
 memcpy(peerInfo.peer_addr, broadcastAddress, 6);
 peerInfo.channel = 0;
 peerInfo.encrypt = false;
 esp_now_add_peer(&peerInfo);
}
```

```
void sendBroadcastMessasge(String message){
 wifiSetup();

 WiFi.setTxPower(transmissionPower);
 esp_now_send(broadcastAddress, (uint8_t*)message.c_str(), message.length()+1);
}
```

```
void onDataSent(const uint8_t *mac_addr, esp_now_send_status_t status){
  if(status!=0)Serial.println("network error");
}
```

> **note:** it's possible to customize the TxPower value, following you would find evaluations for 2dBm and 19.5dBm.

Go to deep sleep

as we said above to save battery the device goes into deep sleep mode for 30 seconds, the following is provided the function that performs this operation:

```
void goToDeepSleep(){
 delayMicroseconds(2000);  //for safety reasons
 WiFi.mode(WIFI_OFF);
 esp_sleep_enable_timer_wakeup(dutyCyclePeriod * uS_TO_S_FACTOR);
 esp_deep_sleep_start();
}
```

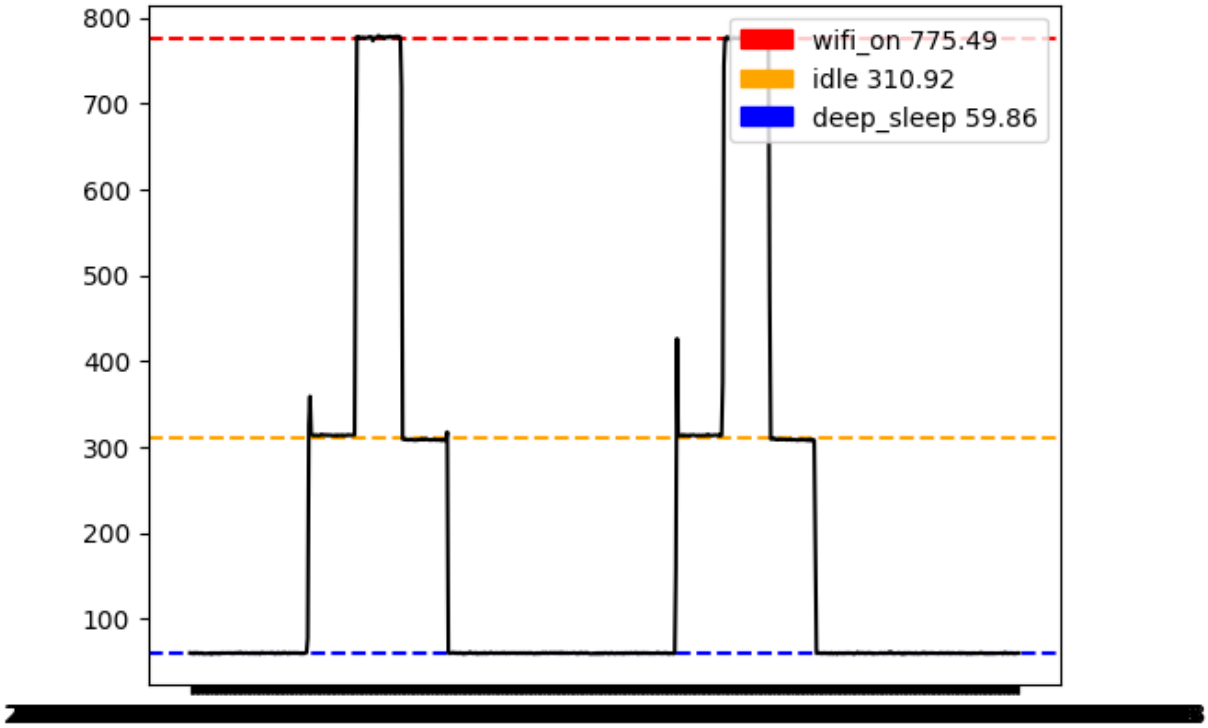
Data analysis

To estimate battery life we measured real energy consumption data in different situations. on the provided samples we had to compute the k-mean estimator to ignore measurement noise.

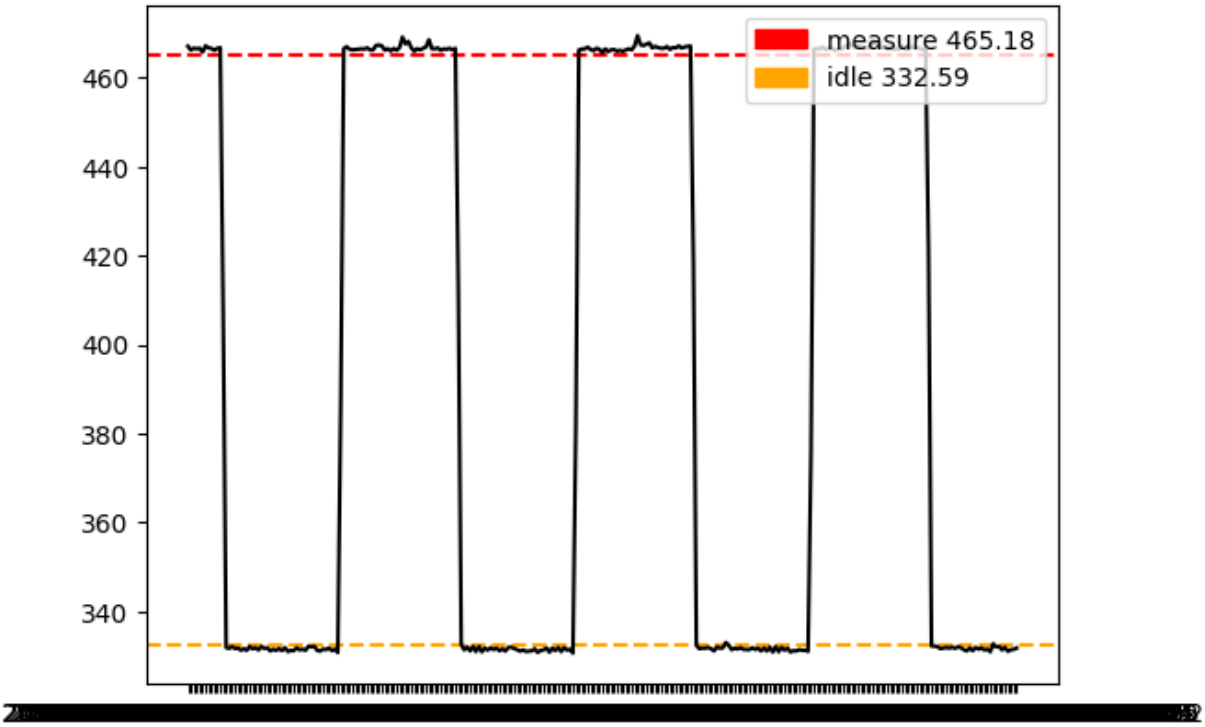
> **note:** we have notice differents consumption data in identical device state in different situations (probabily caused by experimentl erros), we are goin allways to use the worst data.

Following you can find a summary of the data analysis in terms of graphycal rappresentation and tables, but the full analysis on data is available here.

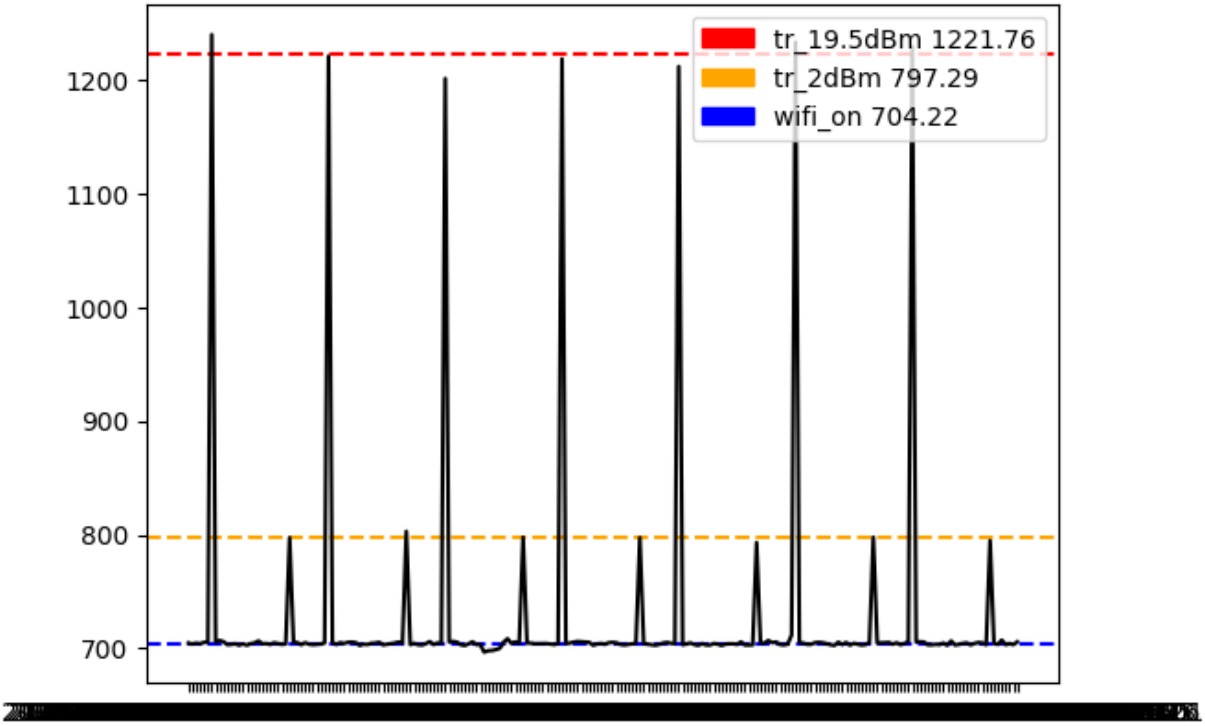
esp32 duty cycle



hc-sr04 energy absorption



esp_now energy absorption



Required data summary

state	total energy absorption
during deep sleep	59.86
during idle state	332.59
during HC-SR04 distance measure	465.18
with wifi antenna on	775.49
during signal transmission (2dBm)	797.29
during signal transmission (19.5dBm default)	1221.76

Battery life estimation

The duty cycle of the node can be split in 6 phases: deep sleep, wake up, perform a distance measure, send the provided data, wait for response, go to sleep.

$$T_{sleep} \Rightarrow T_{wake-up} \Rightarrow T_{sensor-measure} \Rightarrow T_{tx} \Rightarrow T_{wifi-on} \Rightarrow T_{go-to-sleep}$$

The required time to wake up ad to go to sleep is near to zero and the required energy amount is limited, so this 2 phase can be ignored.
So the total energy consumption for a single cycle will be:

$$E_{tot} = E_{sleep} \cdot T_{sleep} + E_{sensor-measure} \cdot T_{sensor-measure} + E_{tx} \cdot T_{tx} + E_{wifi-on} \cdot T_{wifi-on}$$

we need to evaluate each term independently.

Deep sleep power absorption

As said above the device is goin to stay in deep sleep mode for 30 seconds so:

$$E_{sleep} \cdot T_{sleep} == 1.795,8 \cdot 10^{-3} J$$

Distance measure power absorption

The sensor measures the pulse duration, so the time required to perform a measure change with the measured distance as expressed in the following formula:

$$T_{sensor-measure} = \frac{distance \cdot 2}{0,034} J$$

Let's consider the worst cases:

- **free parking spot:** $ distace = 450cm (max distance) \implies T_{sensor-measure} = 26470\cdot10^{-9}s$
- **occupied parking spot:** $distace = 50cm \implies T_{sensor-measure} = 2941\cdot10^{-9}s$

As specified in the documentation the HC-SR04 requires, as a start-measure signal, that the input trig stay high for $\delta t =10\cdot10^{-9}s$,
The worst case possible, in terms of power absorption required to perform distance measurements, is when the parking spot is always free:

$$E_{sensor-measure} \cdot (T_{sensor-measure} + \delta t) = 1,2 \cdot 10^{-5} J$$

A more reasonable situation is when the parking spot is free for 50% of the performed measures:

$$E_{sensor-measure} \cdot T_{sensor-measure} = E_{sensor-measure} \cdot (T^{free}_{sensor-measure} \cdot 50$$

Transimtions power absorption

We need to compute the transmition time, ESP_now allow to send packet with 250bytes of data, assuming a header of 6 Bytes (MAC_address size) L = 256byte = 2048 bit.
Assuming the protocol always sends the whole packet and knowing that the transmission rate is set to 1Mbps (as default), we can compute the transmission time as follows:

$$T_{tx} = \frac{L}{R} = 2,048 \cdot 10^{-3} s$$

We measured the instantaneous absorption for 2 different transmissions (2dBm, 19.5dBm default setting).

$$E^{2dBm}_{tx} \cdot T_{tx} = 1,632 \cdot 10^{-3} J$$

$$E^{19.5dBm}_{tx} \cdot T_{tx} = 2,502 \cdot 10^{-3} J$$

Waiting for response power absorption

To avoid communication errors (and eventually receive messages) the sensors will wait with wifi antenna on for 2ms, we can compute the energy consumption in this phase as follows:

$$E_{wifi-on} \cdot T_{wifi-on} = 1,551 \cdot 10^{-3} J$$

Total energy absorption and battery life estimation

We want to evaluate the battery life, in terms of numbers of completed cycles, for both the power transmission settings.

- power of transmission 2dBm

  $$E^{2dBm}_{tot} = E_{sleep} \cdot T_{sleep} + E_{sensor-measure} \cdot T_{sensor-measure} + E^{2dBm}_{tx} \cdot T_{tx} + E_{wifi-on} \cdot T_{wifi-on} \simeq 1,799 J$$

- power of transmission 19.5dBm

  $$E^{19.5dBm}_{tot} = E_{sleep} \cdot T_{sleep} + E_{sensor-measure} \cdot T_{sensor-measure} + E^{19.5dBm}_{tx} \cdot T_{tx} + E_{wifi-on} \cdot T_{wifi-on} \simeq 1,799 J$$

> **note:** there isn't a significant advantage in reducing the transmission power of the wifi antenna, in fact, the number of cycles does not change.
>
> $$\implies N_{cicles} = \frac{9080}{E^{2dBm}_{tot}} \simeq 5047$$
>
> In conclusion, the sensor has enough power for above 42h.

Possible improvements

This chapter provides some implementation improvements designed to reduce the total energy absorption.

Send and forget

A first simple improvement is applying a send-and-forget paradigm to wifi messages, avoiding keeping the wifi antenna on for 2000 microseconds after sending the sensor state through the ESP_now message.
This improvement allows us to save $1,551\cdot10^{-3} J $ the equivalent of 4,35 cycles.

Longer deep-sleep time

It is reasonable to think that increasing the deep sleep time is a great improvement, but this is not the case, due to the limited amount of power and time required to perform a parking state measure and send it to the sink node.
Modificando, ad esempio, il tempo di sonno profondo del sensore a 3 minuti è possibile eseguire 842 cicli equivalenti a 42,1 ore.
This limited improvement can't justify the compromise in terms of the sensor's responsiveness.

Non constant deep-sleep time

Similar to what is normally done for commercial movement sensors based on Zigbee, we can change the deep sleep time based on the last sensor state (parking spot free or not) applying a longer time for the state that requires a larger amount of energy (the free park in this case).
As discussed in the previous paragraph this can't improve in a significant way the battery life. Still, if in our specific application, we are more interested detect when the parking becomes free (instead of when the parking becomes occupied) the compromise is probably acceptable.

Transmit only on changes

The idea is to save the data read from the HC-SR04 sensor in the flash memory (nonvolatile). At each new cycle the data D(t) is compared with the previous data saved in the flash D(t-1), so the sensor will communicate with the sink node and update the data in the flash memory iff $D(t) \neq D(T-1)$, otherwise the ESP32 can go directly into deep sleep without powering on the wifi antenna and send the data to the risk node.
Assuming the parking spot changes status, in mean, every 3 hours, with this approach we avoid comunicate the node status 5035 times in 42h, saving about $12.63J$ (avoiding transmition at 19.5dBm) the equivalent of 7,01 cycles.

To implement this improvement are required simple modifications to the implementation described above, following you can find the two basic functions used to achieve those modifications:

```
#include <Preferences.h> //required to save boolean into nonvolatile memory
Preferences p;

void saveCurrentState(ParkingSpotState state){
  bool parkingNonVolatileState = (state==OCCUPIED);
  p.putBool("parkingState", parkingNonVolatileState);
}

ParkingSpotState getPreviousState(){
  bool parkingNonVolatileState = p.getBool("parkingState", false);
  ParkingSpotState state = (ParkingSpotState) int(parkingNonVolatileState);
  return state;
}
```

> **note:** the modified implementation for this improvement is provided here.