IoT-Challenge 1

I)	Name: Yi Jiaxiang (leader)	Person Code: **10765316**
II)	Name: Mao Yang Hao	Person Code: 10705881
III)	Name: Cai Simona	Person Code: 10752734

Refer to WOKWI link：https://wokwi.com/projects/391980617546362881, answer the following questions:

1. Explanation of code logic

The logic of code is following:
Basically, we used two main functions:

- *checkOccupancy()*
- *SendStatus(String status)*

```
38  void checkOccupancy(){
39    long duration;
40    float distance;
41    // Start a new measurement:
42    digitalWrite(PIN_TRIG, HIGH);
43    delayMicroseconds(10);
44    digitalWrite(PIN_TRIG, LOW);
45
46    // Read the result:
47    duration = pulseIn(PIN_ECHO, HIGH); //read how much t
48    distance = duration*SOUND_SPEED / 2;
49
50    Serial.println("Distance in CM: "+String(distance));
51
52    if(distance < 50){
53      status = "OCCUPIED";
54    }else{
55      status = "FREE";
56    }
57    Serial.println(status);
58    delay(100);
59  }
```

We will measure the time it takes for the signal to become high, then multiply that time by the speed of sound to calculate the total distance the sonic wave travelled. Afterward, we will divide this total distance by two to determine the actual distance between the object and the sensor.

```
void sendStatus(String status) {
  esp_err_t result = esp_now_send(broadcastAddress,
                                  (uint8_t *)status.c_str(), status.length() + 1);
  delay(200); // Wait for transmission to complete
  if(result==ESP_OK){
    Serial.println("Broadcast message success");
    delay(100);
  }
}
```

Then we use ESP_NOW to deliver the messages of status of parking slot.

Here is our *setup()* function

```
61  void setup() {
62  //unsigned long activeTime = millis();//used to record active time
63    Serial.begin(115200);
64    pinMode(PIN_TRIG, OUTPUT);
65    pinMode(PIN_ECHO, INPUT);
66    delay(200);
67    WiFi.mode(WIFI_STA);
68    esp_now_init();
69    esp_now_register_send_cb(OnDataSent);
70    memcpy(peerInfo.peer_addr,broadcastAddress,6);
71    peerInfo.channel = 0;
72    peerInfo.encrypt = false;
73
74     if (esp_now_add_peer(&peerInfo) != ESP_OK) {
75      Serial.println("Failed to add peer");
76      return;
77    }
78  //check Parking Occupancy FREE or OCCUPIED
79    checkOccupancy();
80  //Sending status of Parking Slot to SINK NODE
81    sendStatus(status);
82
83  /*
84  //Used to record dutycycle FOR TEST
85  unsigned long endTime = millis(); //after test we get 0.67s
86  unsigned long activeDuration = endTime - activeTime;
87  Serial.println("this is activeDuration: "+String(activeDuration));
88  */
89    esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
90    Serial.flush();
91    delay(200);
92    esp_deep_sleep_start();
93
94  }
```

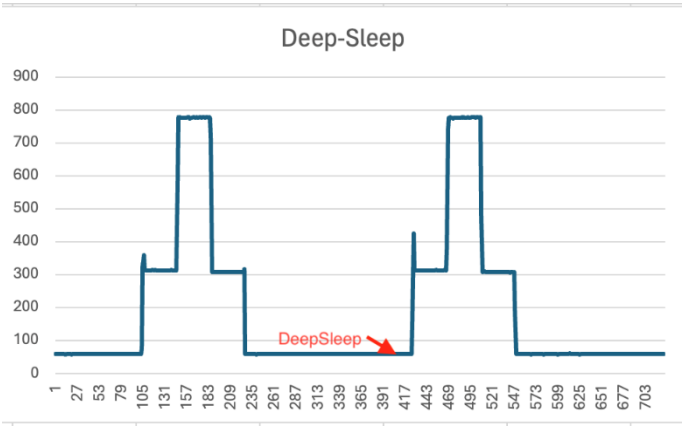
Each times ESP32 wake up from deep sleep mode it will check the occupancy and deliver the message to the ESP32 sink node in this case to broadcast.

We used *mills()* to compute the duty cycle of our ESP32

2. The estimation of the power and energy consumption and battery lifetime

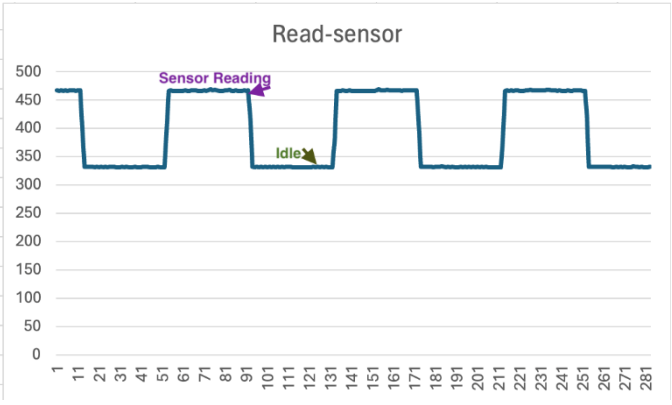
Q: 2.1 Estimate the average Power consumption for each state of the node (Deep Sleep state, Idle, Transmission State, Sensor reading)

A: refer to deep_sleep.csv, send_different_TX.csv, read_sensor.csv provided, we have:



2024-03-03 20:14:49.405132	59.71
2024-03-03 20:14:49.455779	59.61
2024-03-03 20:14:49.505012	59.89
2024-03-03 20:14:49.555232	59.43
2024-03-03 20:14:49.604778	59.38
2024-03-03 20:14:49.655317	59.86
2024-03-03 20:14:49.705451	59.71
2024-03-03 20:14:49.754956	60.04
2024-03-03 20:14:49.805273	59.64
2024-03-03 20:14:49.854786	59.89
2024-03-03 20:14:49.905240	60.43
2024-03-03 20:14:49.954703	59.96
2024-03-03 20:14:50.005332	59.50
2024-03-03 20:14:50.054677	59.24

In average: $P_{DeepSleep} = 60\,mW$

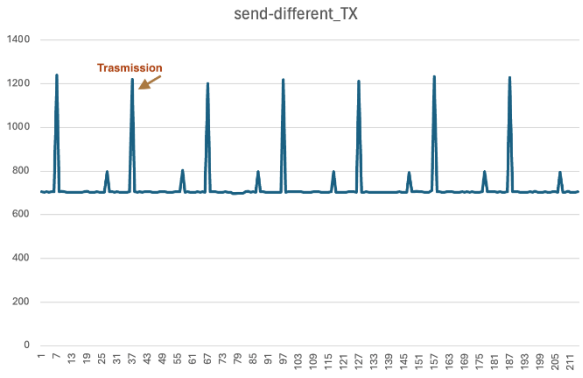


2024-03-03 20:24:59.978875	331.71
2024-03-03 20:25:00.028296	332.04
2024-03-03 20:25:00.078729	331.89
2024-03-03 20:25:00.128268	331.42
2024-03-03 20:25:00.178551	331.14
2024-03-03 20:25:00.228116	331.82
2024-03-03 20:25:00.278405	331.18
2024-03-03 20:25:00.328801	331.93
2024-03-03 20:25:00.378228	331.53
2024-03-03 20:25:00.428659	331.82
2024-03-03 20:25:00.478136	331.35
2024-03-03 20:25:00.528492	331.57
2024-03-03 20:25:00.577935	330.95

In average: $P_{Idle} = 331\,mW$

2024-03-03 20:24:58.829265	466.50
2024-03-03 20:24:58.878622	466.69
2024-03-03 20:24:58.929156	466.43
2024-03-03 20:24:58.978564	466.58
2024-03-03 20:24:59.028939	466.46
2024-03-03 20:24:59.079298	467.04
2024-03-03 20:24:59.128837	466.47
2024-03-03 20:24:59.179224	466.39
2024-03-03 20:24:59.228574	467.02
2024-03-03 20:24:59.279021	466.93
2024-03-03 20:24:59.328434	466.80
2024-03-03 20:24:59.378945	466.61
2024-03-03 20:24:59.428302	466.44

In average: $P_{SensorReading} = 466\,mW$



2024-03-03 19:04:34.752644	1218.18

In average: $P_{Trasmission} = 1218\,mW$

Q: 2.2 Estimate the Energy consumption of 1 transmission cycle

A: For the Deep Sleep State, we used X to referred deep sleep time, where: $X = 21s$

For the other State, we have to run the following code on our mini-project on Wokwi platform:

```
63    unsigned long activeTime = millis();//used to record active time
```

```
82   Used to record dutycycle FOR TEST
83   unsigned long endTime = millis(); //after test we get 0.67s
84   unsigned long activeDuration = endTime - activeTime;
85   Serial.println("this is activeDuration: "+String(activeDuration));
```

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc
E (7038) ESPNOW: esp now not init!
Distance in CM: 197.01
FREE
this is activeDuration: 680
ets Jul 29 2019 12:21:46
```

And on serial log we read time for Idle state, Transmission state and Sensor reading, including some neglecting delay: 0.68s

So,

the energy consumption in Deep Sleep state is: $E_{DeepSleep} = 60\,mW \cdot 21s = 1260mJ$,

the energy consumption in Idle, Transmission and Sensor reading state is:

$$E_{Idle+Transmission+SensorReading} = (331mW + 1218mW + 466mW) \cdot 0.68s = 1370.2mJ$$

In conclusion, the energy consumption of 1 Transmission cycle is:

$$E_{DeepSleep} + E_{Idle+Transmission+SensorReading} = 1260mJ + 1370.2mJ = 2630.2mJ \approx 2.63J$$

Q: 2.3 Estimate the time the sensor node last before changing the battery

A: We have the energy of battery Y joule, where: $Y = 5321J$, just computed in previous point, in $T_{transmission} = 21s + 0.68s = 21.68s$ and $E_{consumption} = 2.63J$ we can compute battery life:

$$Battery_{life} = \frac{E_{battery}}{E_{consumption}} \cdot T_{transmission} = \frac{5321J}{2.63J} \cdot 21.68s = 43862.844s \approx 12.18h$$

3. Comments Results and Improvements

Q: 3.1 Provide a small comment on the implemented system.

A: The ESP32 microcontroller is configured to monitor the occupancy status of a parking space using an ultrasonic sensor. It then uses ESP-NOW communication to send this status to a receiver node. After completing the initial checks and transmission, the microcontroller enters deep sleep mode to conserve power until the next scheduled wake-up time.

The parking space status (status) is stored in the RTC (Real-Time Clock) memory using the RTC_DATA_ATTR modifier. This type of memory persists even when the device enters deep sleep mode, ensuring that the status is retained across sleep cycles.

To detect the occupancy of the parking space, an ultrasonic sensor is employed. It sends out ultrasonic waves and measures the time it takes for them to bounce back, allowing the calculation of the distance to the nearest object. This distance is then used to determine whether the parking space is occupied or free.

To save power, the device enters deep sleep mode (*esp_deep_sleep_start()*) after checking the occupancy and sending the status. Deep sleep mode reduces power consumption by shutting down most device components while maintaining the RTC for wake-up timing. This extends the device's battery life during idle periods.

The *esp_sleep_enable_timer_wakeup()* function enables timer wake-up for scheduled operations. The device will automatically awaken after a specified amount of time (defined by TIME_TO_SLEEP), enabling periodic checks of the parking space status without the need for continuous operation.

Q: 3.2 Starting from the system requirements, propose some possible Improvements in terms of Energy Consumption without modifying the main task «Notify to a Sink node the occupancy state of a parking spot»

A: In terms of energy consumption, a potential improvement could involve having the ESP device check the occupancy status of the parking space upon waking from deep sleep mode. If the status remains unchanged (we stored status with RTC memory, which can survive from deep sleep mode), the device can enter deep sleep mode without activating Wi-Fi or sending a message to the receiver. This approach saves energy while still effectively monitoring the parking space status.