ESP-NOW

jean-claude.feltes@education.lu 1.2024

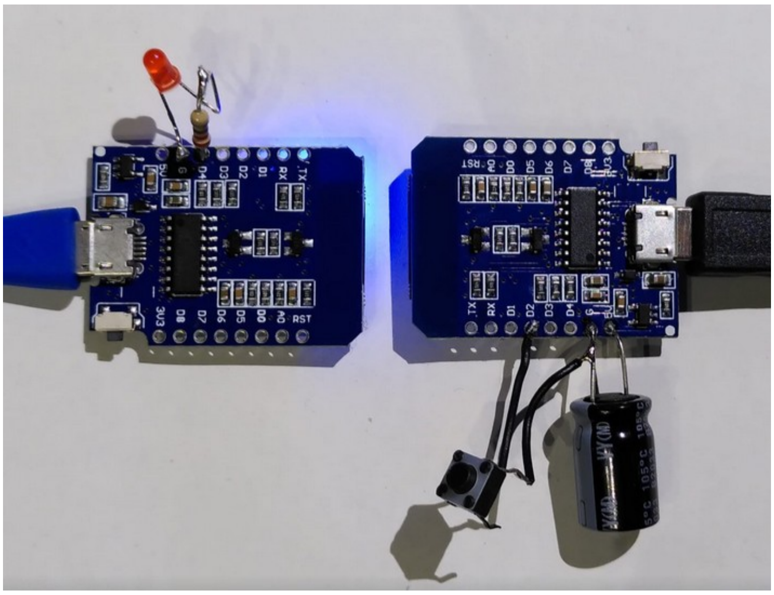
ESP-NOW is a protocol that doesn’t need any WiFi connection. Communication is directly from peer to peer. It is fast, but not as reliable as WiFi.

Here is a good tutorial from which I was strongly inspired for my examples:

https://www.donskytech.com/exploring-esp-now-in-micropython-a-learners-guide/

Simple sender and receiver

Hardware



ESP8266
Left: receiver with LED on D4 = GPIO2
Right: sender with button on D2 = GPIO4
Important: a 1000µF capacitor between +5V and GND buffers the supply voltage.
Without it there is a big chance that the sender doesn’t work!

esp_toolsFind MAC address:

Both use the beginning of a small library module esp_tools.py:

```
import network
import espnow

def init_esp(peer ):
    sta = network.WLAN(network.STA_IF)  # Or network.AP_IF
    sta.active(True)
    sta.disconnect()      # For ESP8266

    esp = espnow.ESPNow()
    esp.active(True)
    if peer:
        esp.add_peer(peer)
        print("Sender initialized")
        print("for receiver ", peer)
    else:
        print("Receiver initialized")
    return esp
```

The function init_esp takes peer as argument.
For the receiver peer is an empty string. For the sender, peer must be the MAC address of the receiver.

Finding the MAC address

find_MAC_address.py:

```
import network
wlan = network.WLAN(network.STA_IF)
wlan.active(True)

if wlan.active():
    mac_address = wlan.config("mac")
    print(mac_address)
else:
    print("Wi-Fi is not active.")
```

This must be run on the receiver. The result is than used for the sender.

The sender

sender_8266_02.py:

```
from espnow_tools import init_esp

from machine import Pin
import utime

btnpin = 4        # Pin 4 = D2! (Strange!)
peer = b'\xc4\xd8\xd5\x12\xec\xf0'      # found with find_MAC_address.py

button_pin = Pin(btnpin, Pin.IN, Pin.PULL_UP)
esp = init_esp(peer)

last_button_state = 1  # Assuming the button is not pressed initially
debounce_delay = 50  # Adjust this value to your needs (milliseconds)

while True:
    button_state = button_pin.value()

    if button_state != last_button_state:
        utime.sleep_ms(debounce_delay)
        button_state = button_pin.value()
        if button_state != last_button_state:
            if button_state == 0:
                message = "ledOn"
            else:
                message = "ledOff"
            print(f"Sending command : {message}")
            esp.send(peer, message)

        last_button_state = button_state
```

This program sends the messages “ledOn” or “ledOff” to the receiver if the button is pushed or released.

The receiver

receiver_8266_02.py:

```
ledpin = 2  # Pin 2 = D4!

from espnow_tools import init_esp
import network
import espnow
import machine

esp = init_esp("")      # peer = "" for receiver

led_pin = machine.Pin(ledpin, machine.Pin.OUT)

while True:
    _, msg = esp.recv()
    if msg:             # msg == None if timeout in recv()
        if msg == b'ledOn':
            print("LED ON")
            led_pin.on()
        elif msg == b'ledOff':
            print("LED OFF")
            led_pin.off()
        else:
            print("Unknown message!")
```

Here the peer is an empty string, signalizing we have not a sender but a receiver.
Depending on the message, the LED is turned on or off.

Automatic start

In boot.py add:

```
import receiver_8266_02
# or
# import sender_8266_02
```

Consumption

For sender and receiver I measured a current of abaout 75-80mA, which is quite high.