

Christoph Stripf Electronics - Microcontroller - Layout

"Traffic Detected" Remote Unit

For Flightradar24-Feeder



Disclaimer:

The information provided is for general informational purposes only. While I strive to keep the information up to date and accurate, I make no representations or warranties of any kind, express or implied, about the completeness, accuracy, reliability, suitability, or availability with respect to the website or the information, products, services, or related graphics contained on the website for any purpose. Any reliance you place on such information is therefore strictly at your own risk. In no event will I be liable for any loss or damage including, without limitation, indirect or consequential loss or damage, or any loss or damage whatsoever arising from loss of data or profits arising out of, or in connection with, the use of this Information.

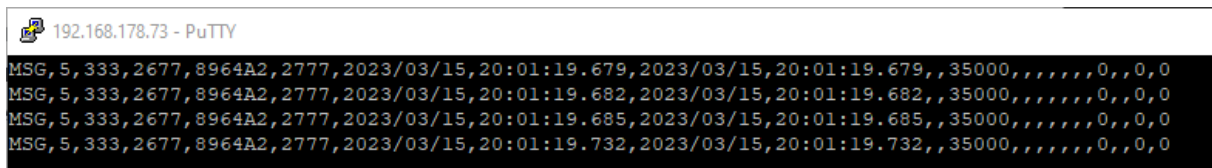
The solution for...:

Problem 1: My FR24-Feeder is mounted far away on my attic. Beside checking the website looking on my radar, I have no control if it works. The "Traffic Detected" Remote Unit signals that the feed to the Server of FR24 is in function by "listening" to the data stream. As long as it "hears" the stream running, the blue LED is on. If anything fails (or there is no aircraft in the remote area) the blue LED is off.

Problem 2: If you live in a "lost" area, there is the chance that you see just some aircraft per day. If you want to know, that there is traffic in your receive area, the blue LED informs you about that.

How does it work?

After connecting to your Wifi (the same network as your feeder is connected to) the unit opens a socket to the IP of your feeder and looks at Port 30003. On this Port, the feeder sends a stream of decoded information of the received ADS-B messages from the aircrafts in the receiving range. As long as this stream is active, the blue LED is turned on. Base for the project is a "ESP32 Wroom" module.



```
192.168.178.73 - PuTTY
MSG, 5, 333, 2677, 8964A2, 2777, 2023/03/15, 20:01:19.679, 2023/03/15, 20:01:19.679, , 35000, , , , , , 0, , 0, 0
MSG, 5, 333, 2677, 8964A2, 2777, 2023/03/15, 20:01:19.682, 2023/03/15, 20:01:19.682, , 35000, , , , , , 0, , 0, 0
MSG, 5, 333, 2677, 8964A2, 2777, 2023/03/15, 20:01:19.685, 2023/03/15, 20:01:19.685, , 35000, , , , , , 0, , 0, 0
MSG, 5, 333, 2677, 8964A2, 2777, 2023/03/15, 20:01:19.732, 2023/03/15, 20:01:19.732, , 35000, , , , , , 0, , 0, 0
```

Figure 1: Example for data stream on Port 30003

What you'll need

Shopping list

- 1 x ESP32 Wroom module
- 1 x Blue 3mm LED (or any other color) for the LED "traffic"
- 1 x Green 3mm LED (or any other color) for the LED "WIFI"
- 1 x Enclosure by your choice

Tools required

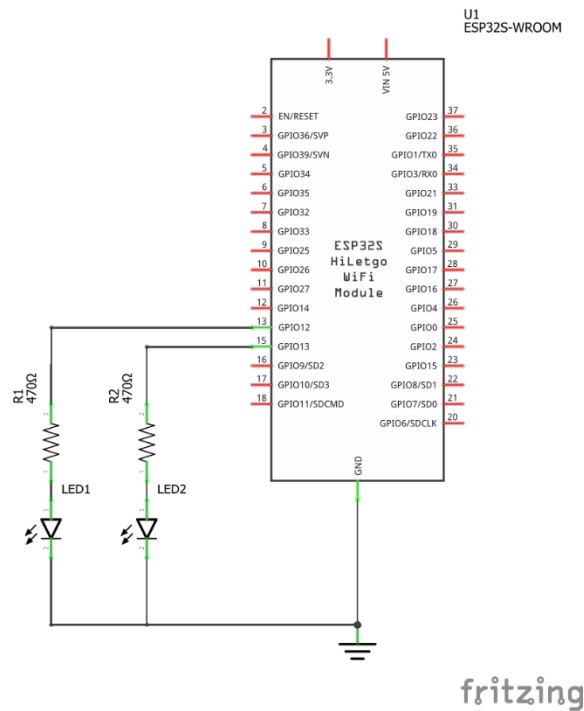
- Soldering iron
- Wire cutter
- Some wire
- A drill
- And all the other stuff... depends on your needs

Software required (recommended)

- Thonny

Putting the Hardware together

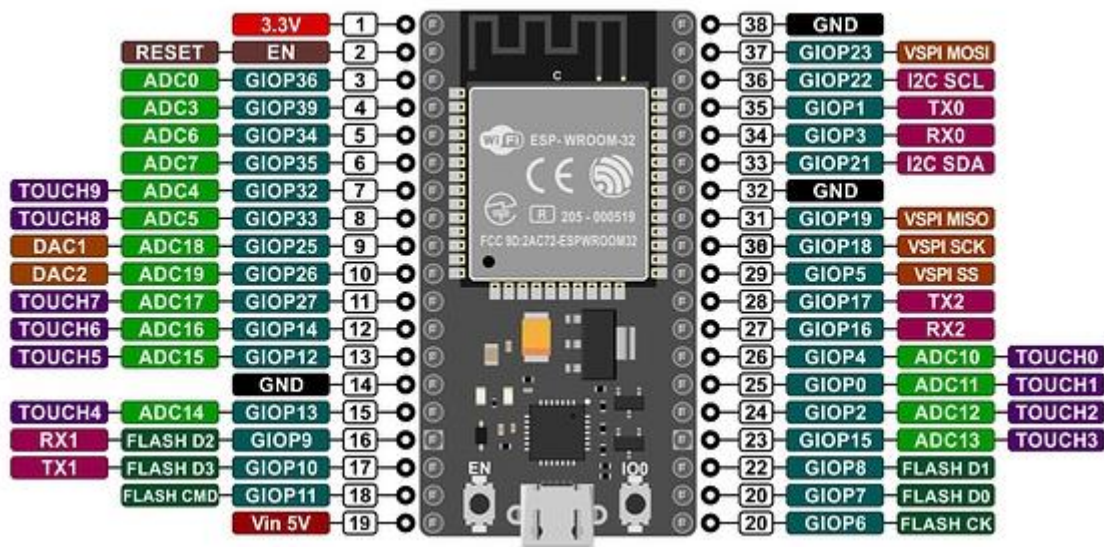
The Schematic:



fritzing

The Schematic is not that spectacular. Just two resistors and two LED's.

The LED's are connected to Pin 13 and Pin 15 of the ESP32. Pin 14 is GND. That makes the connection easy and neat. Using 470Ω resistors, allows to connect the LED's directly without overloading the pin.



The Software:

I am absolutely NOT a professional Programmer. So, please excuse, if the code does not fit any programming guidelines. The code is written in Micropython.

```
1. from machine import Pin
2. import time
3. import network
4. import socket
5. import gc
6. import machine
7.
8. gc.enable()
9.
10. #Host-Adress and Port of the Feeder (Port should be 30003)
11. HOST='192.168.178.73'
12. PORT=30003
13.
14. # WLAN SSID and Password
15. wlan_ssid = "SSID of your WIFI"
16. wlan_passwort = "Password of your WIFI"
17.
18. LED_Traffic = Pin(13, Pin.OUT)
19. LED_Traffic.off()
20. LED_WLAN = Pin(12, Pin.OUT)
21. LED_WLAN.off()
22.
23. WLAN_Error_Flag = False
24.
25. wlan = network.WLAN(network.STA_IF)
26.
27. def wlan_connect():
28.     while not wlan.isconnected():
29.         print("Connecting...")
30.         start = time.ticks_ms();
31.         wlan.active(False)
32.         wlan.active(True);
33.         time.sleep(0.5)
34.         wlan.connect(wlan_ssid,wlan_passwort)
35.         while not wlan.isconnected() and start + 5000 > time.ticks_ms():
36.             pass
37.
38. wlan_connect()
39. LED_Traffic.off()
40. LED_WLAN.off()
41.
42. if wlan.isconnected():
43.     LED_WLAN.on()
44.     print("WLAN Connected")
45.     WLAN_Error_Flag = False
46.
47. else:
48.     LED_WLAN.off()
49.     print("WLAN Connection failed")
50.     WLAN_Error_Flag = True
51.     wlan_connect()
52.
53. if wlan.isconnected():
54.     sock = socket.socket()
55.     sock.settimeout(60)
56.     sock.connect((HOST, PORT))
57.
58. while 1:
59.
```

```
60.     try:
61.
62.         if not wlan.isconnected():
63.             print("WLAN ERROR - Reconnect")
64.             wlan = network.WLAN(network.STA_IF)
65.             WLAN_Error_Flag = True
66.             LED_WLAN.off()
67.             LED_Traffic.off()
68.             time.sleep(10)
69.             wlan_connect()
70.
71.         if not WLAN_Error_Flag:
72.             Data = 0
73.             Data = sock.recv(8500)
74.
75.             if not Data:
76.                 LED_Traffic.off()
77.                 time.sleep(10)
78.                 sock.close()
79.                 sock = socket.socket()
80.                 sock.settimeout(60)
81.                 sock.connect((HOST, PORT))
82.
83.             LED_Traffic.on()
84.             time.sleep(0.5)
85.             LED_Traffic.off()
86.
87.             gc.collect()
88.
89.     except OSError as error :
90.         machine.reset()
```

So, after the Hardware is finished, you need to put some Information into the software:

- The SSID of your WIFI
- The WIFI-Password
- The IP-Address of the feeder you want to monitor

After saving the code as Main.py on the internal memory of the ESP32, you should have the finished product in front of you. The enclosure you want to use is up to you.

Questions? => Mail me: christoph.stripf@gmail.com

