DB4 Computer Science FAQs - 2022

How can I connect the ESP32 board to the Internet?
All the boards must be connected to the Internet using one of your smartphones. First, enable the hotspot function of your smartphone to create a local WiFi network (use a password of more than 8 characters, no spaces!). Then, on the ESP32 you can connect using the following snippet of code.

```
import network
# turn off the WiFi Access Point
ap_if = network.WLAN(network.AP_IF)
ap_if.active(False)
# connect the device to the WiFi network
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect("WIFI_SSID", "WIFI_PASSWORD")
```

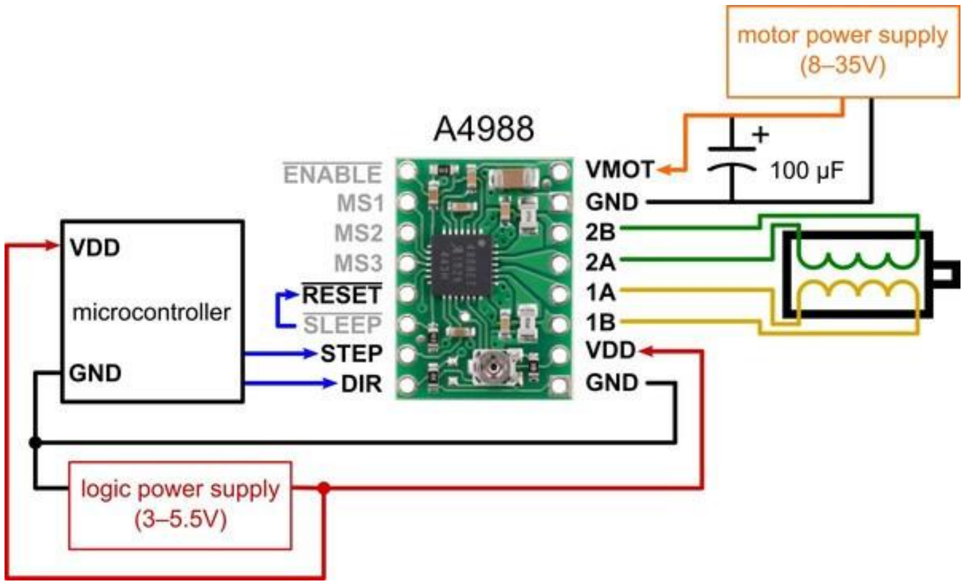
How do I test if I am connected to the the Internet?
Once you connect to your hotspot network, you can test if you are connected to the Internet using the following snippet of code, which prints the first 1000 characters received form the www.dtu.dk web-page. If you can receive data form the www.dtu.dk webpage, it means you can access the Internet.

```
import socket
request = b"GET / HTTP/1.1\nHost: www.dtu.dk\n\n"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("www.dtu.dk", 80))
s.settimeout(2)
s.send(request)
result = s.recv(10000)
print(result)
s.close()
```

What about Internet connection during night? Do I need to leave my smartphone overnight when I run a long experiment?
We do not recommend you leave your smartphone overnight when you run long experiments. We suggest that, beside uploading the data to AdafruitIO, you also to log them in a file. You can then download this file or, if you are up for a challenge, upload the data to AdafruitIO when the connection is available.

How do I get started with MQTT?
In addition to the links provided in the presentations. Please, take a look at the following links to get started with MQTT:
https://github.com/miketeachman/micropython-adafruit-mqtt-esp8266/blob/master/publishAdafruit.py
https://io.adafruit.com/api/docs/mqtt.html#adafruit-io-mqtt-api (follow the MicroPython uMQTT link)

How do I connect the A4988 motor driver to the microcontroller and to the motor?
As reference, look at the following image.



Note that there are two powers supplies. A low voltage 3.3V (or 5V) power supply for the logic and the 12V motor power supply. When the enable pin is 0, the motor is enabled; when 1, the motor is disabled. You can connect this pin to the microcontroller or directly to ground. The MS1, MS2, and MS3 pins regulate the size of the step and must be connected to 0 or 1 according to the desired step. Smaller steps have more torque, but less speed.

The motor connector pins 1B, 1A, 2A, and 2B are in the same order as in the motor cable connector.

Is it normal that the A4988 motor drivers warm up a lot?
Yes, they should be fine up to 85 degrees Celsius. However, if you want you can reduce the maximum current with the trimmer on the driver itself by rotating it unclockwise. Rotate it to the point where the current is just sufficient to rotate the pump.

What do the RGB sensor values represent?
When you read from the sensor with the provided library, you get **raw** values. These values are red, green, blue, and clear (16 bits each). These values are then normalized against the total light (clear component) and gamma correction is applied (power to 2.5) inside the **color_rgb_bytes_new(color_raw)** function that was given to you in the **i2c_test.py** script. This can be an issue for your measurements since the total light intensity (clear component) scales the r, g, b result. For example, if there is a strong blu component, the green, and red ones, even if present are cancelled out. **Therefore, we recommend you use the raw r, g, b components read from the sensor directly.** You can modify the **color_rgb_bytes_new(color_raw)** function as follow to also return the raw r, g, b, and clear values.

```
def color_rgb_bytes_new(color_raw):
    r, g, b, clear = color_raw
    if clear == 0:
        return (0, 0, 0)
    red   = int(pow((int((r/clear) * 256) / 255), 2.5) * 255)
    green = int(pow((int((g/clear) * 256) / 255), 2.5) * 255)
    blue  = int(pow((int((b/clear) * 256) / 255), 2.5) * 255)
    if red > 255:
        red = 255
    if green > 255:
        green = 255
    if blue > 255:
        blue = 255
    return (red, green, blue, int(r), int(g), int(b),
int(clear))
```

One more thing, you can change the sensor gain and the integration time of the sensor using the following two commands. The gain is just an amplification factor (increases the sensitivity). The integration time is the time when the sensor is sensitive (like the exposure time in a camera). Tune these values so that you have the maximum dynamic range, without reaching saturation of the read values.

```
sensor.integration_time(10) #value between 2.4 and 614.4.
sensor.gain(16) #must be a value of 1, 4, 16, 60
```

How do I remove the main.py file when Ampy cannot connect because the ESP32 is busy running the main.py itself?
The main.py file can be deleted from the Python shell using the serial terminal (e.g., Putty). Once connected to the serial terminal. Interrupt the execution of the current program (CTRL+C) and delete the file using the following commands:

```
import uos
uos.remove("main.py")
```

How can I use multiple PWM frequencies to control the motor (it seems that the PWM frequency is shared between all PWM outputs)?
The huzzah board MicroPython framework supports a single oscillator for all the 8 PWM channels. Thus, the frequency is shared. For this reason, we suggest to use PWM for the cooling pump (where speed can be important) and bit-banging (controlling the output pin with a dedicated function) for the food dosing pump (where the dosing amount is important).

How do I write to the OLED display?
You first need to load the package file ssd1306.py available on DTU Learn on your ESP32 board using Ampy. You can then look at the following snippet of code and to online resources to write text. Do not use the .mpy files you find online since they are not supported by the MicroPython framework.

```
import ssd1306
from machine import I2C, Pin
i2c = I2C(scl=Pin(22), sda=Pin(23), freq=100000)
oled = ssd1306.SSD1306_I2C(128, 32, i2c)
```

```
oled.fill(0)
oled.text("I", 0, 8)
oled.text("wrote", 8, 16)
oled.text("this!", 16, 24)
oled.show()
```

How do I communicate with multiple I2C devices (OLED display, RGB sensor)?
You first need to load the package file *.py for the device you want to use on your ESP32 board using Ampy. These are available on DTU Learn. You can then look at the example code i2c_test.py available on DTU Learn. Here, the OLED display, RGB sensor, and UV/IR/VISIBLE LIGHT sensor are used together. On the board, all the devices share the same I2C data --SDA-- and clock --SCL (or SCK)-- pins.

What is the speed of the serial port for communicating with the ESP32 board?
Baud rate: 115200

My ESP32 board is misbehaving or it seems dead. What should I do?
Ask Luca to flash the board (takes less than 5 minutes) or for a replacement.

I burned/destroyed some component or I need additional components to implement advanced/alternative functionality (resistors, capacitors, ICs, etc.). What should I do?
We have a selection of components in out DTU Compute lab. Ask Luca to bring what you need.