Table of Contents

1. Laboratory on the Internet of Things 2
1.1. Hardware 3
1.2. VSCode, PlatformIO, and project configuration 4
1.2.1. Running your code 5
1.3. Software 5
1.4. Networking 5
1.5. Network services 5
1.5.1. MQTT 6
1.6. Tracing 7
1.7. Laboratory scenarios 7
1.7.1. Scenario Lab 1 8
1.7.2. Scenario Lab 2 9
1.7.3. Scenario Lab 3 12
1.7.4. Scenario Lab 4 14
1.7.5. Scenario Lab 5 15

1.Laboratory on the Internet of Things

Welcome to the laboratory of the Internet of Things. Students are abou
laboratory tasks. Each task is to be ûnished with a report summarizing
The report should be prepared in PDF form and eventually ZIPped with
resources, etc.

1.1 Hardware

Each team is given the following IoT microcontroller:

- Wemos D1 R2 development board:

Fig. 1. WeMos D1 R2 development board

Note, this microcontroller, even if its board reminds Arduino Uno has 0..3.3V while Arduino Uno uses classical 0..5V logic) and has different following figure:

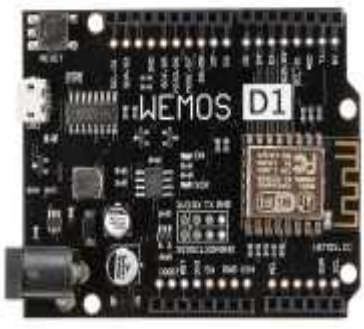


Fig. 2. WeMos D1 R2 pinout

- DHT 11 or DHT 22 temperature and humidity sensor



Fig. 3. DHT 11 (left) and DHT 22 (right) temperature and humidity sensor

- 2x16 characters LCD with keypad shield



Fig. 4. LCD shield 2×16 characters

1.2. VSCode, PlatformIO, and project conû guration

In any case, creating a new project under PlatformIO should follow thi
Board and Framework 5) see û gure



Fig. 5. Project conû guration settings in VSCode/PlatformIO

This conû guration brings the following project conû guration û le platformio.ini

```
[env:d1_mini]
platform = espressif8266
board = d1_mini
framework = arduino
```

It is also necessary to conû gure the serial port name used for uploadi
(if you use it). In the case of this microcontroller platform, the name port,
should be modiû ed with the following (assuming COM3 the serial port is, i.e.

```
...
upload_port = COM3
monitor_port = COM3
...
```

1.2.1 Running your code

To execute your code, you need to compile it and upload it to the micr
The easiest way to interact with PlatformIO is to use its blue ribbon b
VSCode (image 6).

Fig. 6: PlatformIO ribbon to execute most common actions

1.3 Software

You will be writing an application for end nodes (ESP8266) in C++ usi
Additionally, you may need to use an MQTT client, i.e., MQTT Explorer

> Computers in laboratory rooms 320, 321, and 336 use fr
> means that for every startup, computers are restored to their i

1.4 Networking

There is a WiFi 2.4GHz IPv4 network available to the students. This is
network is separated from the Internet, but all connected devices can
DHCP server with networking addressing scope 192.168.90.10-192.168

> Do not use static IP! Some services in the network
> overlapping causes the unavailability of your solution and thi
> Use only DHCP to obtain a network address.

To connect to the wireless network, use the following credentials:

- SSID: internal.IOT
- Passkey: IoTlab32768
- IP address: DHCP, 192.168.90.10-192.168.90.254

1.5 Network services

1.5.1 MQTT

There is an MQTT Broker, a binding WiFi PRIVATE network, and the PU
you can route MQTT messages from your end node devices connected t
to the external tools, i.e., cloud-based dashboard or storage via, i.e.,
IoT device through the PRIVATE network to MQTT Broker and subscrib
cloud using PUBLIC Campus network. Please note the context (thus, th
broker) changes depending on which device and network you're using.
should be of help.

You can access the MQTT broker from the Campus PUBLIC network usi
and credentials:

- Address: 157.158.56.54
- Port: 1883 for plain text, 8883 for secure communication
- User: vrel
- Password: vrel2018

You can also access the MQTT broker from the PRIVATE network using the fo
conûguration and credentials:

- Address: 192.168.90.5
- Port: 1883 for plain text, 8883 for secure communication
- User: vrel
- Password: vrel2018



Fig. 7 Zmitac network infrastructure for the IoT labs

1.6 Tracing

There is no direct debugging capability in this environment. Yet, it is messages to the PC console over a serial (serial over USB) port, whic serial communication, one must invoke begin with the appropriate bps singleton object (Serial) in the setup() function:

```
Serial.begin(9600);
```

9600 is the default speed, but you can change it in the platformio.ini file. It is possible to s virtually any message to the console (keep in mind, you need explicitl code) using:

```
...
Serial.print(...);
//or
Serial.println(...);
...
```

Both functions accept a wide range of argument types without needing

> Note: there is a limit on the character set that is ASCII special characters, which are language-specific.

1.7 Laboratory scenarios

All teams share the same scenario yet; please note that MQTT messag shall be coordinated among team members so as not to overlay one an

> Mind keeping your code between meetings. It is regular that meeting, you will extend your code by something extra.

1.7.1 Scenario Lab1

Getting ready with your development environment
In this scenario, you will write your first Hello World application for
write code that:

1. Flashes built-in LED every 1 second (1s on; 1s off);
2. Sends a message via serial port to the Serial Monitor: Hello World

> The GPIO the built-in LED is connected to is number 14. Phys
> in the middle of the mainboard, under the display, and its colo

```
#define LED_BUILTIN 14
```

Note, to use tracing with the Serial.print(&) singleton object, you need to tell VS C
port to use to receive tracing messages you send from your code. Minc
platformio.ini file the appropriate configuration (here COM3 is a sample;
configuration):

```
...
monitor_port = COM3
...
```

Obligatory:

- Provide a solution not using millis() nor delay()

Advanced:

- Use a timer/interrupt to flash the LED instead of an infinite loop.

Report:

- Describe your approach and include source code in the report

1.7.2 Scenario Lab2

** Using LCD **

In this scenario, you will write an application that reports on the LCD
Your code shall implement Hello message in the upper line and a counter tickin
second in the lower one.
Additionally, preset a label to LEFT/RIGHT/UP/DOWN when a key is pressed on the 6-key
located below the LCD display module. The label should appear in a lo
a ticking counter that is still working. The label should disappear whe

1.7.2.1 The LCD Display

To control LCD, it is easiest to use a software library (add to the

```
...
lib_deps = arduino-libraries/LiquidCrystal @ ^1.0.7
...
```

Your LCD shield is 2×16 characters. To initialize it use the following

```
constint rs = 12, en = 13, d4 = 4, d5 = 0, d6 = 2, d7 = 14;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

Remember to initialize it in your code; you need to tell the library the
display:

```
...
  lcd.begin(16,2); //2 lines, 16 characters each
...
```

1.7.2.2 Keypad

The LCD shield has 6 buttons. One is hard-wired to the reset function,
SELECT, UP, DOWN, RIGHT, LEFT) can be read through voltage analys
(Analogue to Digital Converter, Channel 0) existing in the ESP8266 ch
way that they short the connection, and every button creates a separa
up and respectable resistance, different for each button (see diagram). T
estimate readings, you can calculate the voltage appearing on the ADC

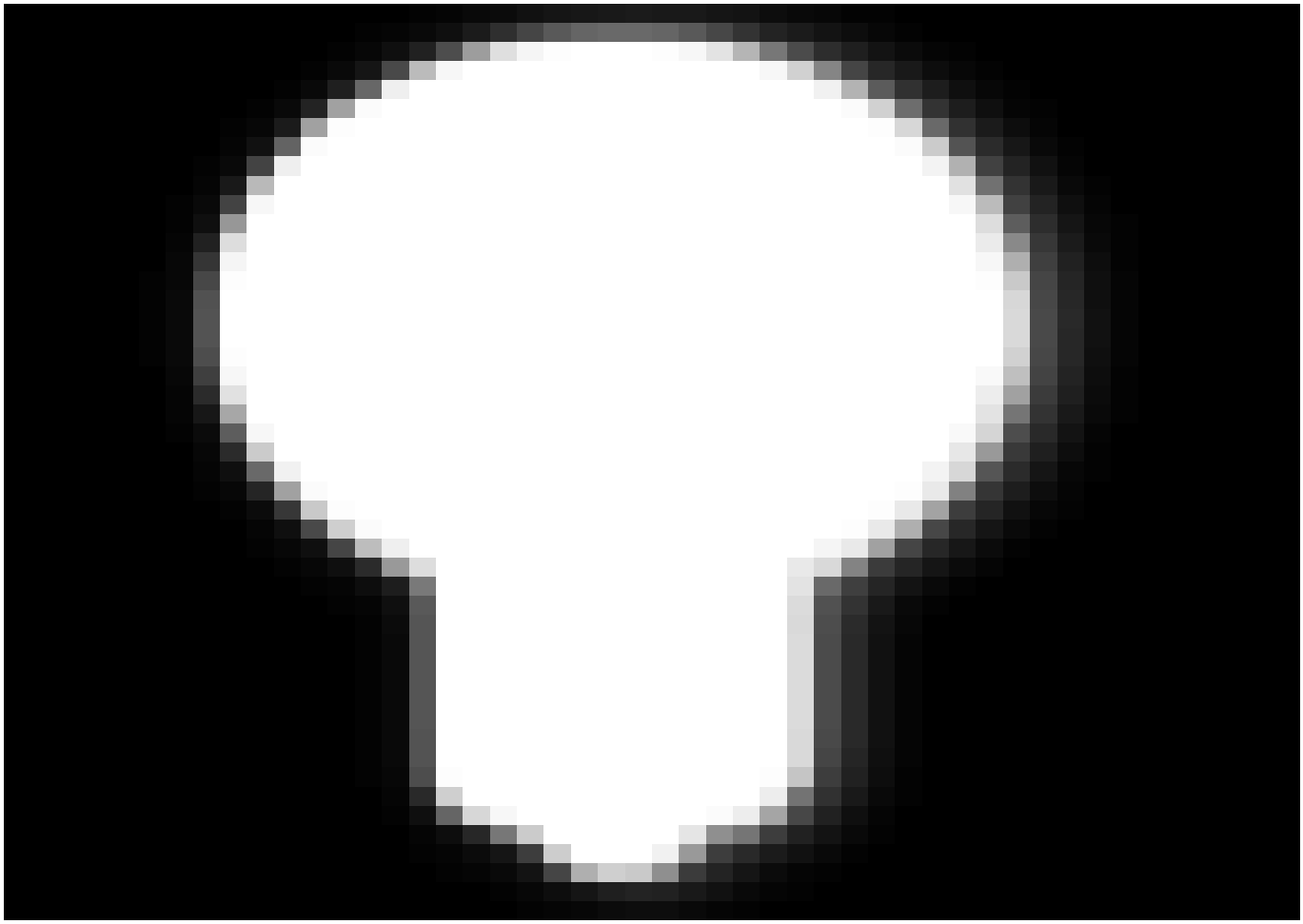


Fig. 2. 16×2 LCD shield connection diagram including buttons

Note there is an assumption that you press only one button at a time. Because when you press a button, it goes to the AD0, and you get maximum reading (because the maximum voltage is 5V, as shown in the diagram below, on readings AD0).

> Because of the LCD shield construction and the fact that ES the box on their AD0 input voltage range of 0..3.3V only (act voltage divider in the middle), you're unable to grab button S shorting it causes input voltage to on the AD0 to be ~3.7V, pressed, the input voltage on AD0 is 5V. Both are far beyond t you will always get a maximum reading (1024) in both cases distinguish them without electrical modification.

Reading an AD0 is as simple as:

```
#define ADCPIN 0
...
  int nAdcReading=0;
...
  nAD0= analogRead(ADCPIN);
...
```

Report:

- Describe how you solved the problem of the asynchronous key pres
- Include source code.

1.7.3 Scenario Lab3

** Presenting temperature and humidity on the local LCD screen **
In this scenario, you will measure temperature and relative humidity u
present it on the screen. In the first row, temperature and humidity
in the lower, measured value, right-aligned along with the measuremen
presenting temperature and humidity should be possible using the Up/
and changing the temperature between Celsius and Fahrenheit using th
Measurements should include (for temperature) and humidity. The temperat
should be present with one decimal point, while humidity should be an

1.7.3.1 Temperature and humidity sensor

There is a DHT11 (blue) sensor connected to the
There are two libraries needed to handle reading from the sensor: a co
library and a specific one for DHT sensors, both by Adafruit, so two m
platformio.ini file, and you need to refer to it in the source code:

```
lib_deps =
    adafruit/Adafruit Unified Sensor@^1.1.4
    adafruit/DHT sensor library@^1.4.3
```

```
#include <Adafruit_Sensor.h>
#include <DHT.h>
```

To access sensor readings, it is necessary to instantiate a communica
hardware to the code:

```
#define DHTTYPE DHT11   // DHT 11
#define DHTPIN 5
DHT dht(DHTPIN, DHTTYPE);
```

It is also necessary to initialize the sensor class and give it some tim

```
...
dht.begin();
delay(1000);
...
```

The sensor class returns data (among others) in a float containing tem
humidity (in percent):

```
...
  float hum = dht.readHumidity();
  float temp = dht.readTemperature();
...
```

Report:

- Present a circuit with connections (LCD, keypad, and sensor).
- Present an algorithm for both overall program û ow and for obtainir
- Include your code with appropriate comments.

1.7.4 Scenario Lab4

** Connecting to the network **
Connect your ESP8266 (as a client, STA mode) to the network. Write the IP address obtained from the DHCP server. Present the connection status (the upper row of the LCD) and the IP address in the second line.

Report:

- When using ESP in STA mode, observe whether the solution's subs new or the same IP address via the DHCP server.
- Include source code.

1.7.5 Scenario Lab5

Sending data to the network
In this scenario, you will implement a fully functional IoT end-node an
MQTT Broker. Use your solution from the last step of Lab4 and also La
humidity, present them on the screen as in the case of the Lab3 scena
as MQTT messages to the Broker.

> Mind agreeing with other sections to let you do not ove
> Topics. They should be unique for all teams!

Using any software-based MQTT client, subscribe to the messages usi
and observe results. You can use a PC computer to develop a solution

The simplest starting point is to use the pub-sub MQTT client library:

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

You may also want to deû ne MQTT topic(s) and constant payloads (i.e.
û ag) using statement. It is also a good idea to put connection const

```
//WiFi and MQTT server credentials
#define wifi_ssid "internal.IOT"
#define wifi_password "IoTlab32768"
#define mqtt_server "192.168.90.5"
#define mqtt_user "vrel"
#define mqtt_password "vrel2018"
//MQTT last will
#define lastWillTopic "/sut/aeii/iot-lab/321/node1/state"
#define lastWillMessage "off"
#define mqttWelcomeMessage "on"
```

Mind is your MQTT server address.

To use the pub-sub client, you must instantiate a global object using a
WiFiClient:

```
...
// WiFi & MQTT
WiFiClient espClient;
PubSubClient client(espClient);
...
```

Somewhere in the function, once your wiû is already connected, bind s
MQTT client object:

```
...
```

```
client.setServer(<server>,1883); //standard port is 1883, change according
if needed
...
```

Then, connect your MQTT pub-sub client to the server. You can deû ne need to repeat this process when the connection is down (for any reas procedure:

```
void reconnect() {
  // Loop until we're reconnected
  while(!client.connected()) {
    Serial.print("Attempting MQTT connection...")
    if (client.connect(<clientname>,mqtt_user,mqtt_password)){
      Serial.println("connected to MQTT")
    } else{
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds")
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

To publish a single message, use:

```
...
client.publish(<topic>,<payload>);
...
```

Note that receiving messages requires periodical execution of the client.loop() function. Add it to your void loop() function. If you send messages only (publish but d using client.loop(); is not obligatory.

Advanced:

- Implement a last will informing that the device has lost connecti disconnect) present the last available measurements as persistent by changing a persistent message on connect/disconnect to let the of your device. To implement controls like doors or home theaters, use keypad (you assume that you may need to restart the device to get it back on t implement a bistable logic: the first trigger of MQTT disconnect, the s click connects it back to the Broker).

To publish a last-will message, you may use an extended client.connect() th function (i.e., reconnect() function:

```
...
client.connect(<clientname>,mqtt_user,mqtt_password,lastWillTopic,0,
true, lastWillMessage)
...
```

Once connected, you may notify the dashboard (if any) that you're bac

```
...
client.publish(lastWillTopic,mqttWelcomeMessage);
...
```

Report:

- Provide your MQTT message Topics and Payload formats (semantic
- Include source code.

From:
https://dokuwiki.zmitac.aei.polsl.pl/ - RAU6 Dokuwiki

Permanent link:
https://dokuwiki.zmitac.aei.polsl.pl/en:edu:zmitac:macro:io_17137

Last update: 2024/04/21 16:17