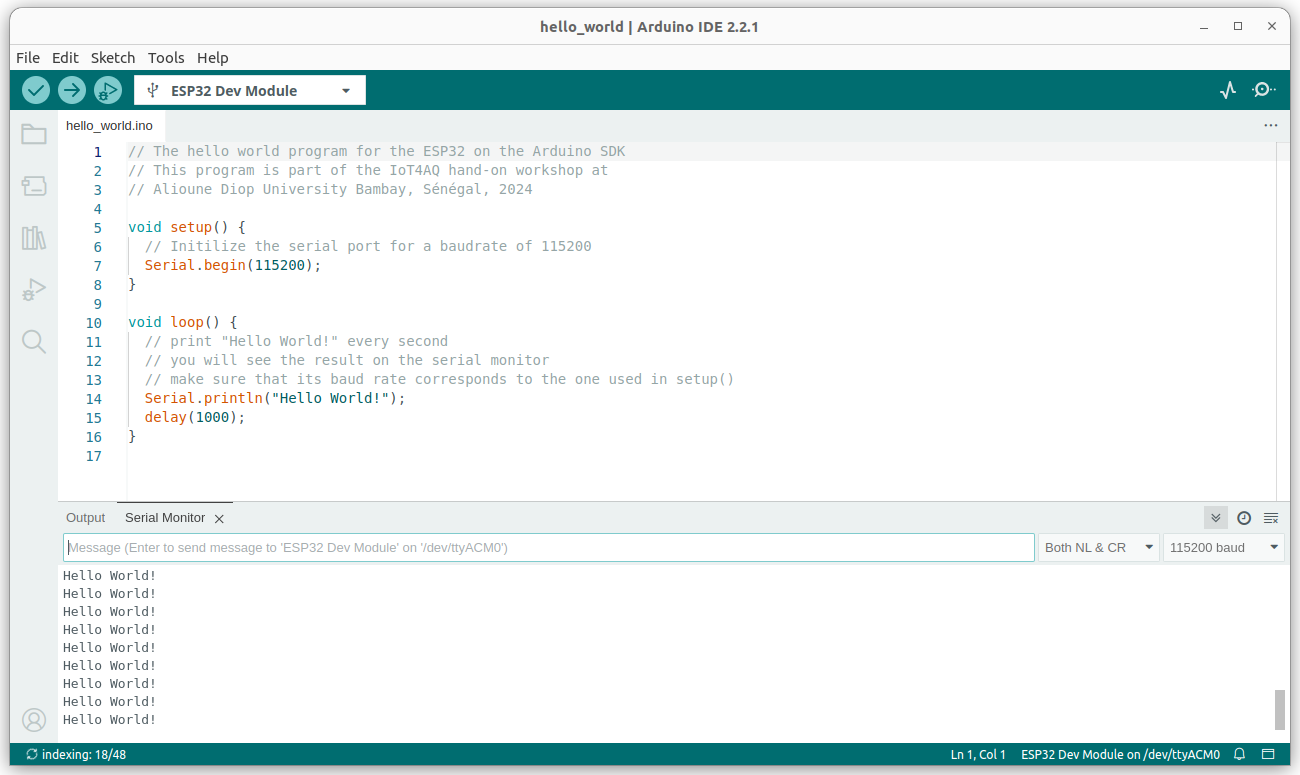
|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

Iot4AQ workshop exercises

## Basic C++ exercises

**Exercise 1:**Write a program that prints “Hello World!” every second. Use Serial.println to print the text to the serial monitor. Use a baud rate of 115200.  
To just compile the program, use the checked button on the left of the Arduino SDK window.  
To compile and upload, use the right arrow.  
The serial monitor can be started with the button that looks like a magnifying glass (top right)  
Make sure the serial monitor is also set to 115200 baud.  
This is what you should see:



**Exercise 2:**

Modify the program adding two numbers, that was shown during the lecture, and extend it to all four basic arithmetic operations: add, subtract, multiply, divide.

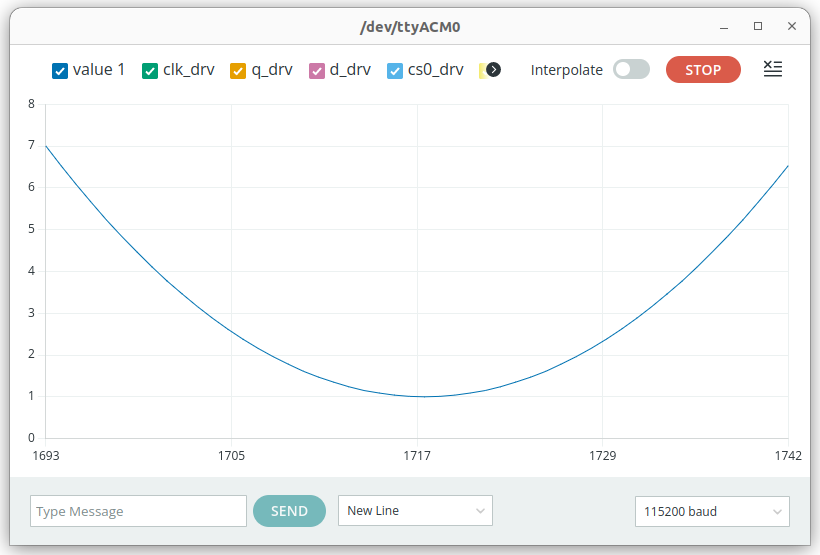
|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

What is the result when you divide and why?  
What will change it instead of declaring the variables as integers you declare them as floats?

**Exercise 3:**Write a program that counts up from zero to 40 and then down to zero again. Use “while” loops to do this.  
Then modify the program to use “for” loops.  
Modify the program again using 2 functions, one for the up counter and one for the down counter. Pass the maximum value to be reached, as a parameter.  
You may use the plotting function (the button left of the serial monitor button, showing a signal) to visualize the counters.

**Exercise 4:**  
Write a function that calculates a parabola. The formula is:  
y=a\*(x-h)2 + k  
Then use the function to print 50 values for x in the range of -2 .. 2. Have a look at the curve on the plotter.  
Try the values a=1.5, h=2.0, k=1.0

Once this is working, you may of course play with the values and see how the curve changes.  
How to calculate the x values? The total size of the range is 4 (-2 .. 2) and you need 50 x values. Therefore the step in x must be 4/50. Then you must offset these values by -2. The formula for the x values therefore is: x = 4.0/50.0 \* i – 2.0. If you do not believe this, you can print the x values to see that there are 50 values starting at -2 and running up to +2 (actually +2 – 4/50).

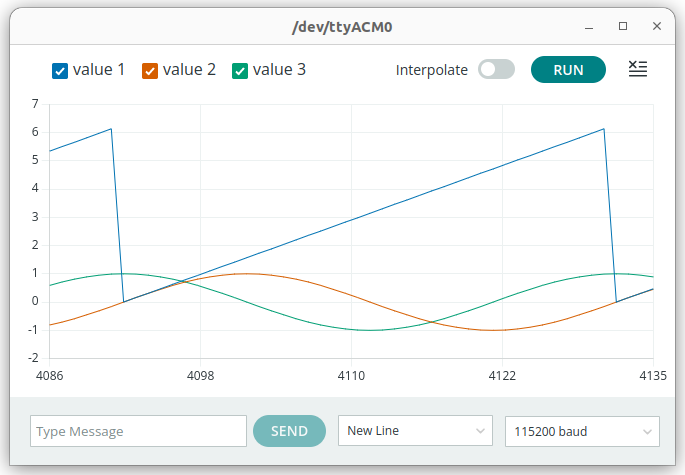


|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

## Too easy?

**Exercise 5:**Plot the sin and cos functions as you did with the counters above. You must include the math library at the beginning of your program to get access to the trigonometric functions as well as the constant M\_PI.   
#include <math.h>  
Use 40 values per period. This is how it looks like:

The blue curve corresponds to the x values of sin(x) or cos(x). To which function does the green and yellow curve belong?



|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

IoT4AQ workshop exercises

## Exercises on GPIO

**G**eneral **P**urpose **I**nput **O**utput (GPIO) are digital signal lines that can be programmed as input or output.

The ESP32 has an LED that is user programmable and very often a program blinking this LED is used to test the CPU hardware.

**Exercise 1:**

Blink: Write a program that blinks the LED on the CPU card at a frequency of 1 Hz (500ms on, 500 ms off)

**Exercise 2:**

SOS: In the marine world the morse code “SOS” was used to alert security crews that a boat was in difficulties and needed to be rescued. The code consists on 3 short pulses (“S” in morse code), 3 long pulses (“O” in morse code) followed by another 3 short pulses.  
Implement a program that sends the SOS signal on the user LED. Use an interval of 200 ms between light pulses, 200ms for the short light pulse and 700 ms for the long light pulse.  
You may use a function sending a light pulse that takes its length as a parameter.

**Exercise 3:**

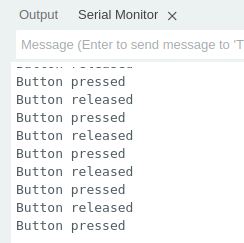
The CPU card has 2 push buttons. The first one (EN) is the reset button and the second one, boot, connected to GPIO 0 is used to put the CPU into flash mode. (You must press the boot button, keep it pressed while you press and release the reset button, and then release it).  
In case you do not use it to put the CPU into boot mode, you can use this button like any push button connected to any other GPIO line.

**Exercise 4:**

Write a program that reads the boot button every 100 ms and prints its state.



* Write a program that reads the push button every 100 ms but indicates its state only when it changes.

****

**Too easy?**

* Write a program increasing the light intensity of the user LED in a linear way and the decreasing it linearly again. Hint: Look up how **P**ulse **W**idth **M**odulation (PWM) works and how it is used on the Arduino SDK.

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

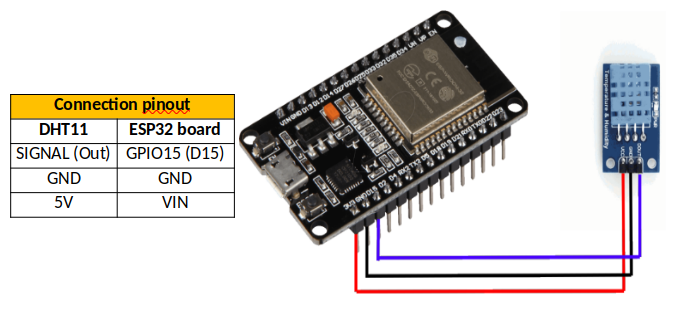
IoT4AQ workshop exercises

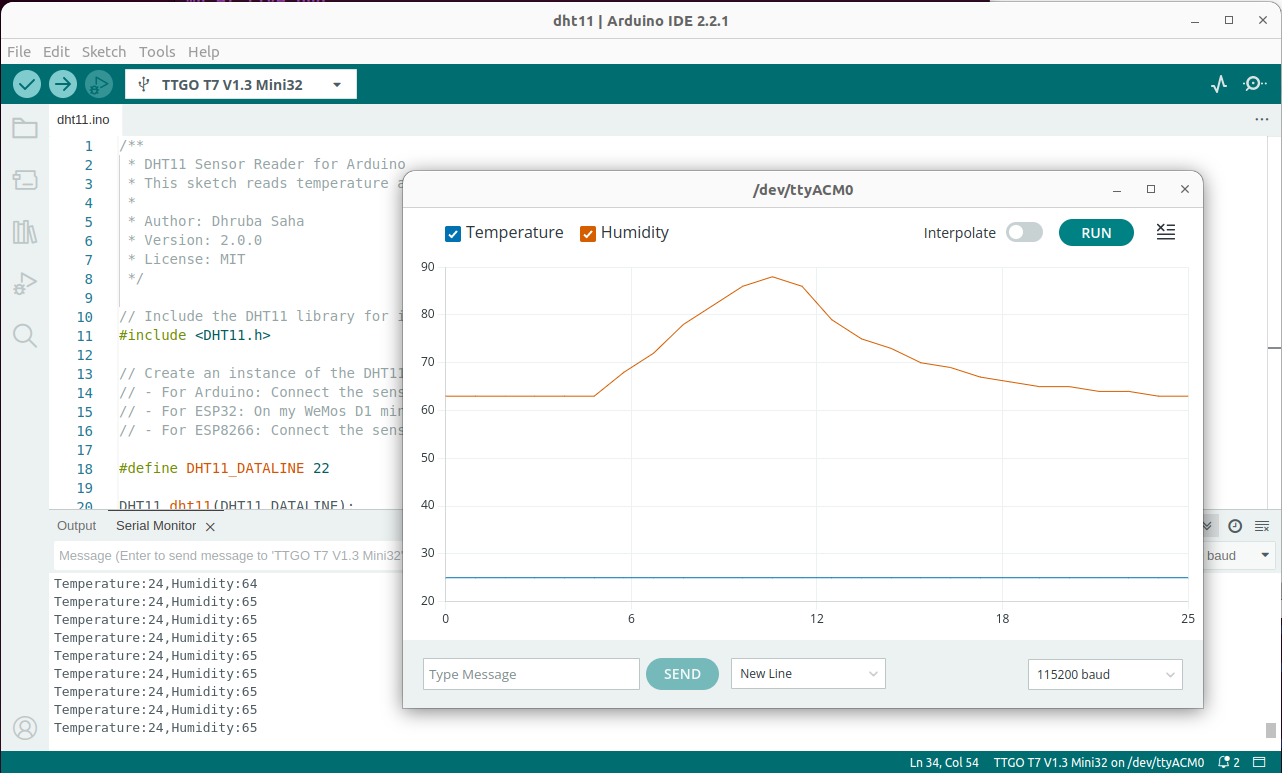
## Exercises on the DHT11

The DHT11 is a temperature and humidity sensor using its own proprietary serial communication protocol. A [DHT11 driver](https://github.com/dhrubasaha08/DHT11) is available on github. In order to understand this driver and how exactly the DHT11 works you must read the [DHT11 data sheet](https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf).

**Connections:**

**Exercise 1:**Initialize the DHT11 class  
Read the temperature and humidity and print the results. Make sure that a 1s interval between measurements is respected.   
Observe the measurements on the Arduino SDK plotter  
Check for possible errors and print them if they occur ( you may run the program without the DHT11 connected and observe what happens). On the screen dump below I put my finger onto the detector to see the rise in humidity.



****

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

**Too easy?**

… and you want to know the details on how the DHT11 protocol works? Be ready for a rough ride!  
Without using the driver:

**Exercise 2:**

set the data pin to output and start a measurement

once the measurement is started convert the data line to input and read its state every 4 µs

Shift the data read into an array of 32 uint32\_t integers: first bit: highest significant bit of the first integer, last bit: lowest significant bit of the last integer and print the array on the serial line.

Copy/paste the results into a text editor of your PC

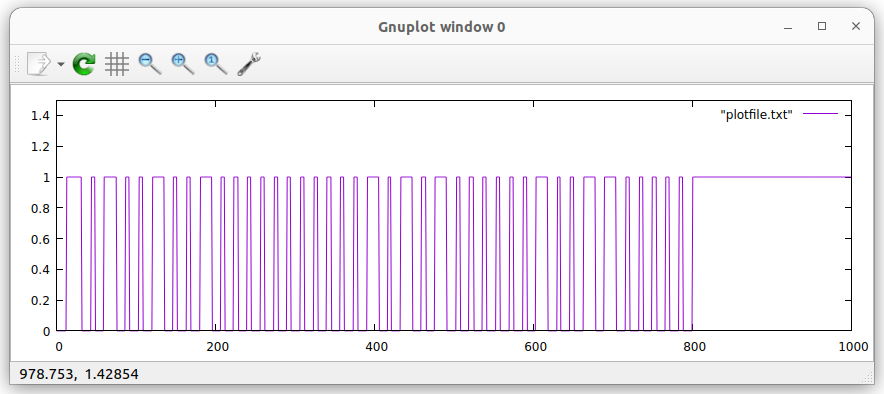
|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

**Exercise 3:**

Write a program to extract the zeros and ones into a new file.

Plot this file on your PC. It will show a   
logic analyzer type of plot.

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |



Extract the results from the protocol by hand. The long pulses correspond to “1” the short ones to “0”

Make sure the checksum is correct

In the above plot, the first long pulse is the response signal of the DHT11 (80 us) and should  
not be taken into account.  
Then we get:

0100 1001 = 0x49 = 73 relative humidity integral part in %  
0000 0000 = 0x00 = 0 relative humidity decimal part  
0001 0101 = 0x15 = 21temperature integral part in °C  
0000 0020 = 0x02 = 2 temperature decimal part  
0110 0000 = 0x60 checksum  
0x49 + 0x15 + 0x02 = 0x60 : checksum ok

Extract the results from the plot file by program on the PC.

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

Iot4AQ workshop exercises

## Exercises on the PlanTower PMS5003 dust sensor

**Exercise 1:**

Make the hardware connections according to the table below:

|  |  |  |
| --- | --- | --- |
| **PMS5003 pin number** | **PMS5003 pin significance** | **ESP32 GPIO pin number** |
| Pin 1 | Vcc | 5V |
| Pin 2 | GND | GND |
| Pin 3 | Set (suspend mode) | not used |
| Pin 4 | Rx (serial receive) | GPIO 17 |
| Pin 5 | Tx (serial transmit) | GPIO 16 |
| Pin 6 | Reset | not used |
| Pin 7 | NC (not connected) |  |

Install the ESP32:PMS5003 library into your Arduino SDK (should be already done)

Read the PMS5003 protocol message using the readMsg() method provided by the PMS5003 class. readMsg returns a structure of type pms5003Data.  
Print the values received for pm1.0, pm2.5 and pm10  
Print the message header.  
Print the checksum.

**Exercise 2:**

Read the raw data using the readRaw() and print them with printRaw(). readRaw returns a pointer to the raw data contained in the PMS5003 message. You can use this pointer as a parameter to printRaw()

Print the checksum as provided by the protocol message and the checksum calculated over the protocol entries and compare them.

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

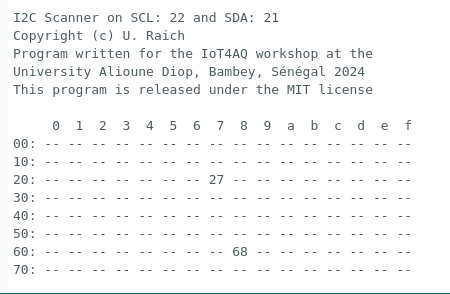
IoT4AQ workshop exercises

# Exercises on I2C

**Connections:** 3.3V, GND and SCL: GPIO 22, SDA: GPIO21, see below

**Exercise 1:**

Write a program that prints the I2C addresses of all I2C slaves connect to the bus. This is how the output of the program could look like:



The address 0x68 corresponds to the DS3231 RTC while 0x27 is the address of the PCF8574 I/O extender attached to the LCD display.

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

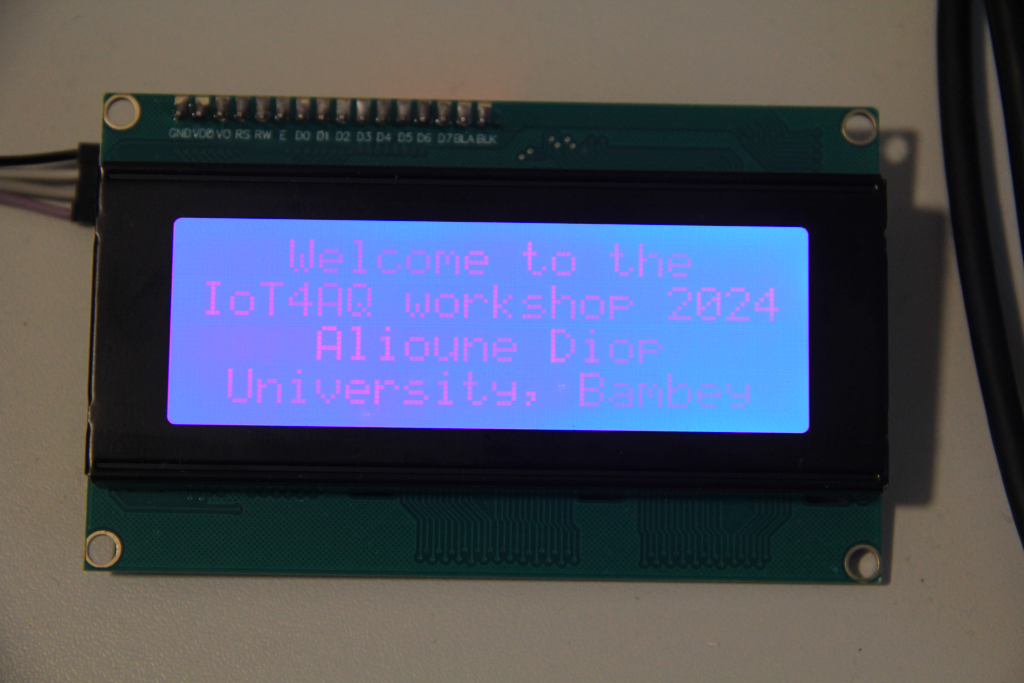
IoT4AQ Workshop Exercises

## Exercises on the 2004 LCD

The 2004 LCD can display 4 rows of 20 characters each.

**Exercise 1:**

Write the following text to the display:

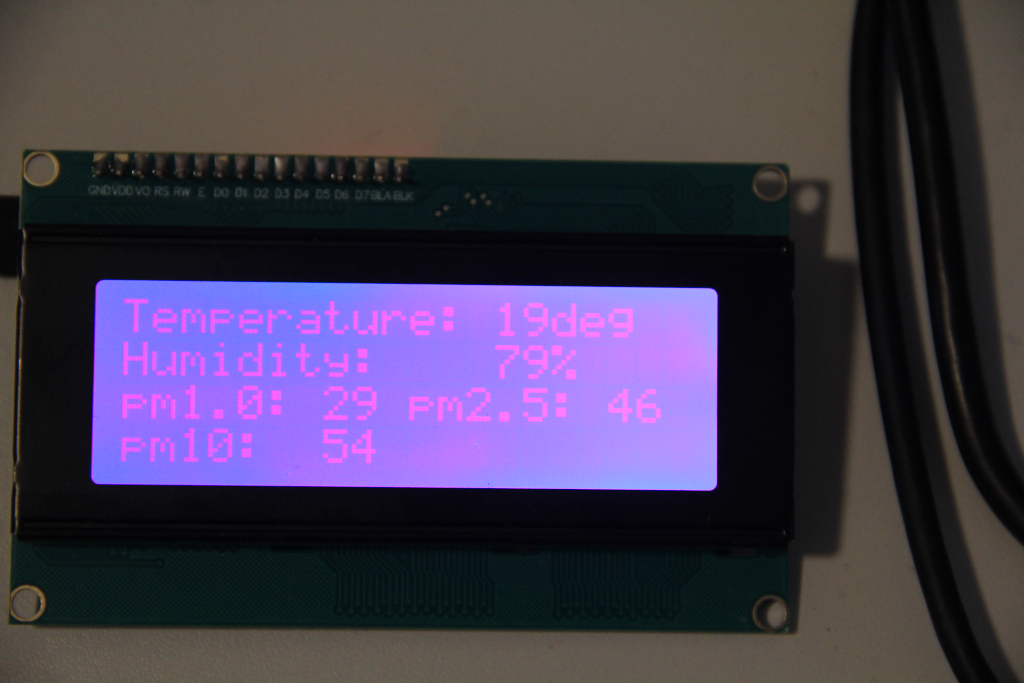
Welcome to the  
IoT4AQ workshop 2024  
 Alioune Diop  
 University, Bambey

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

**Exercise 2:**

Read the temperature and humidity from the DHT11  
Read the pm1.0, pm2.5 and pm10 values from the PMS5003

Print the results on the display in the following format:  
Temperature: 22deg  
Humidity: 56%  
pm1.0: 5 pm2.5: 6  
pm10: 10

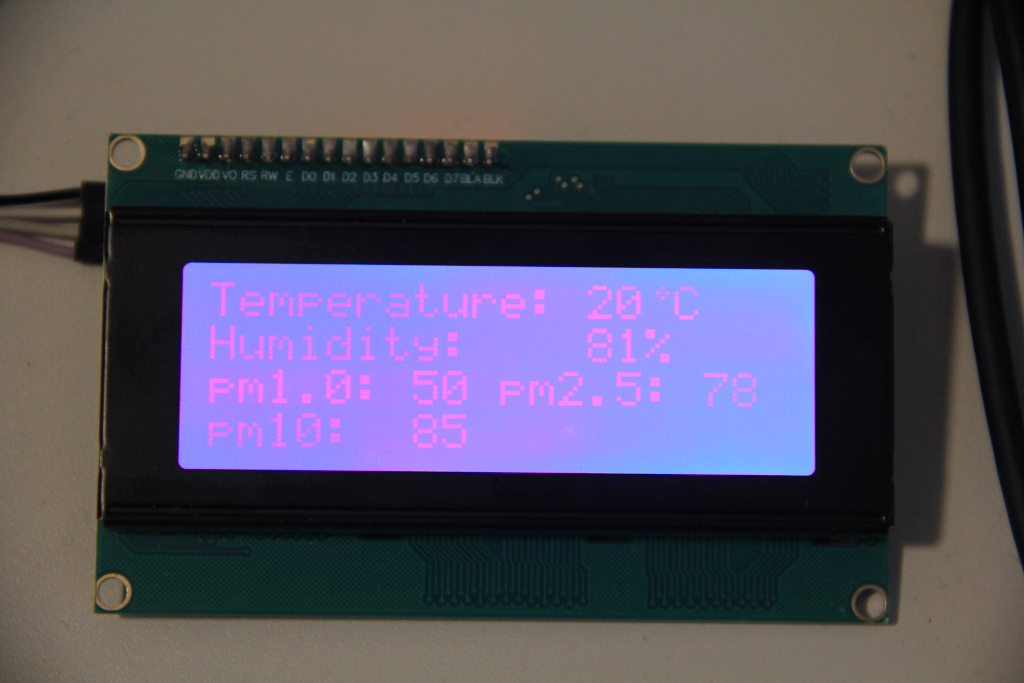
****

**Too easy?**

**Exercise 3:**

Replace “deg” in the second program by °C. The “°” character does not exist in the ROM character table but can be added as a user defined character.

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |



|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

IoT4AQ workshop exercises

## Exercises on the SD card

**Exercise 1:**

Provide a program that writes the following text to the file *meas.txt* on the SD card  
Welcome the the IoT4AQ workshop  
It is held at the Alioune Diop University Bambey (UADB),   
Sénégal, March 2024

Refuse to write to *meas.txt* if it exists already

**Exercise 2:**

Write a program reading the file *meas.txt* and printing its contents

**Exercise 3:**

Write a program deleting the file again

**Exercise 4:**

Read temperature and humidity from the DHT11 and save the values to the file *meas.txt*If the file exists already then open it in *append* mode.

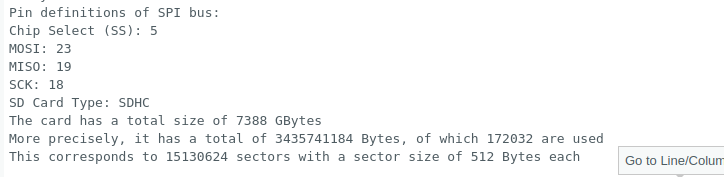
## Too easy?

**Exercise 5:**

Write a program that shows the file system tree

**Exercise 6:**

Provide a program that prints all relevant information on the SD card



|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

IoT4AQ workshop exercises

## Exercises of WiFi

**Exercise 1:**

Write a program that connects to the WiFi router

Print its IP address

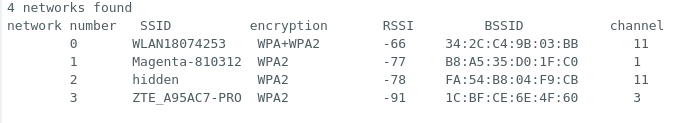
Check with ping that the ESP32 node has appeared on the network

### Too easy?

**Exercise 2:**

Disconnect from the network

Scan the WiFi network and print the relevant information for each node you have found



## Still too easy?

**Exercise 3:**

Write a TCP server on your PC and a TCP client on the ESP32

Send messages between the client and the server

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

IoT4AQ workshop exercises

# Exercises on the Real Time Clocks

## Introduction

If the ESP32 does not have access to the Internet we must read date and time from the battery backed DS3231 RTC. This however means that the DS3231 has been correctly set before.  
In our standalone IoT4AQ program we must first read the DS3231 and use its values to set the ESP32 RTC. The ES32 RTC will be used for all time functions after that.

The library ESP32Time gives access to the ESP32 RTC while the RTClib allows to read and write the DS3231.

**Exercise 1:**

Write a program to set the ESP32 RTC. Get time and date from NTP. You must connect to WiFi before you can request the time from NTP.  
Use configTime(gmtOffset, daylightOffset, ntpServer) to set the ESP32 RTC  
Read back the time and date from the ESP32 RTC and print it on the serial console

**Exercise 2:**

Modify the program to use the ESP32 RTC settings to set the DS3231 RTC. Use RTClib to accomplish this.

**Exercise 3:**

Write a program to get date and time from the DS3231 and print it.

**Exercise 4:**

Write a program that recuperates date and time from the DS3231 and uses the result to set the ESP32 RTC

### Too easy?

**Exercise 5:**

Access the DS3231 through I2C commands directly, using the wire library, instead of passing through RTClib

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

IoT4AQ workshop exercises

# Exercises on ThingSpeak

## Basics

**Exercise 1:**

* Register a user on ThingSpeak
* Create a channel with only one field (e.g. for temperature)
* Add a gauge widget to the dashboard

In the exercise solutions we have a program ThingSpeakBasic.ino that creates a triangular stream of values (20, 21, 22,… 40, 39, 38,...20) and sends them to the ThingSpeak channel and field every 15s.

* Study this program and run it to
* Observe that the chart is updated

|  |  |
| --- | --- |
| Temperature Graph | Temperature Gauge |

|  |  |
| --- | --- |
|  | *Internet of Things Lab for Air quality Monitoring (IoT4AQ)*  *International Workshop – Air Quality & IoT-based Air Sensors*  *14-15 March 2024, Alioune Diop University, Senegal* |

## Final ThingSpeak program

**Exercise 2:**

Create a channel with fields for

* temperature
* humidity
* pm 1.0, pm 2.5, pm 10

Read temperature, humidity and dust concentrations  
You need a delay of 1s between the measurements of temperature and humidity and dust concentrations. Therefore it takes 3 s for a complete set of values.  
Since ThingSpeak only takes values every 15s (for a free ThingSpeak account) we have time to read 5 complete sets of values and average them. Send the result to the ThingSpeak channel.

## Too easy?

**Exercise 3:**

Write a program that can run without Internet access. In this case you must get time and date from the Real Time Clock (set the ESP32 RTC at the beginning of the program from the external DS3231 RTC). Write the time stamp as well as the measured data onto the SD card. Write time,date and current measurement results onto the LCD display.