Introduction to IoT

School Year 2023-2024

Valsalice

Introductions

- Alberto Spina
  - (2015) Valsalice Alumni
  - (2019) MEng Computing - Imperial College London
  - (2023/current) Software Engineer - London

Introductions - Icebreaker

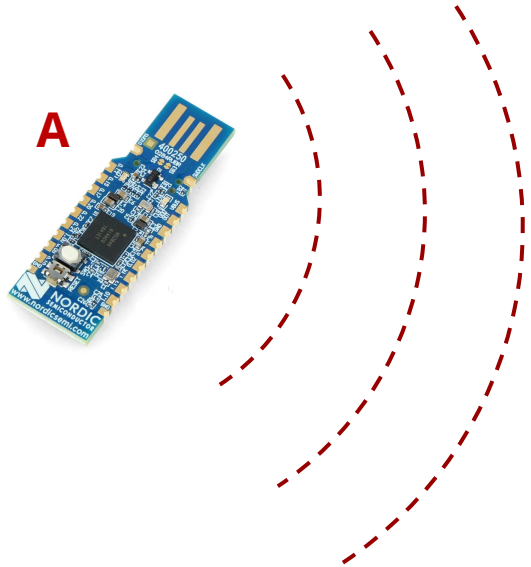
- What is your name?
- What grade are you in?
- Have you programmed before?
- What is an interesting fact about yourself?

Course Structure

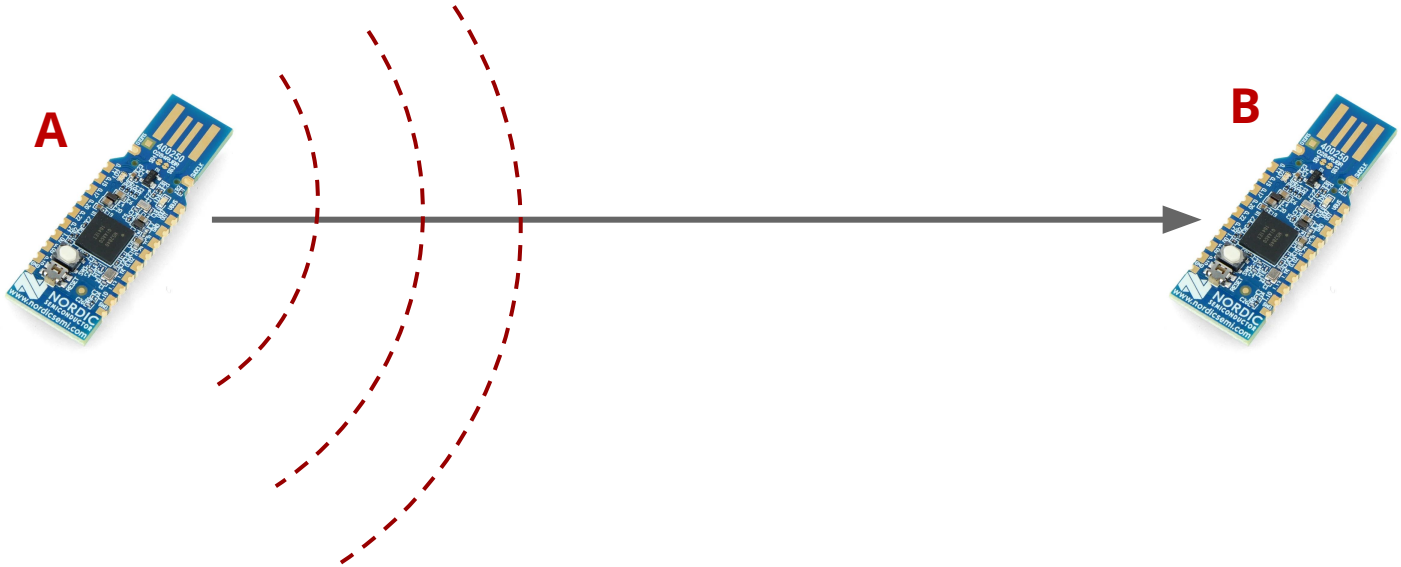
1	Introduction and Basics	OCT 17
2	Basic Data Types and Operators	OCT 24
3	Control Structures	OCT 31
4	Functions and Scope	NOV 7
5	Arrays and Strings	NOV 14
6	Pointers and Memory Management	NOV 21
A	Preprocessor and Macros	DEC 19
B	Advanced Data Structures	JAN 9
7	Introduction to Contiki-NG and nRF52840	NOV 28
8	Sensing and Actuating with Contiki-NG	DEC 5
9	Basic Communication and Networking	DEC 12
10	Introduction to RPL and Network Routing	JAN 16
11	Challenges in Wireless Communication	JAN 23
12	Advanced Protocols: TSCH and 6TiSCH	JAN 30
C	Advanced Topics in Wireless Communication	FEB 6
D	Reliable Data Transfer Challenge	FEB 20, FEB 27, MAR 5

= Core Topics = Optional Topics

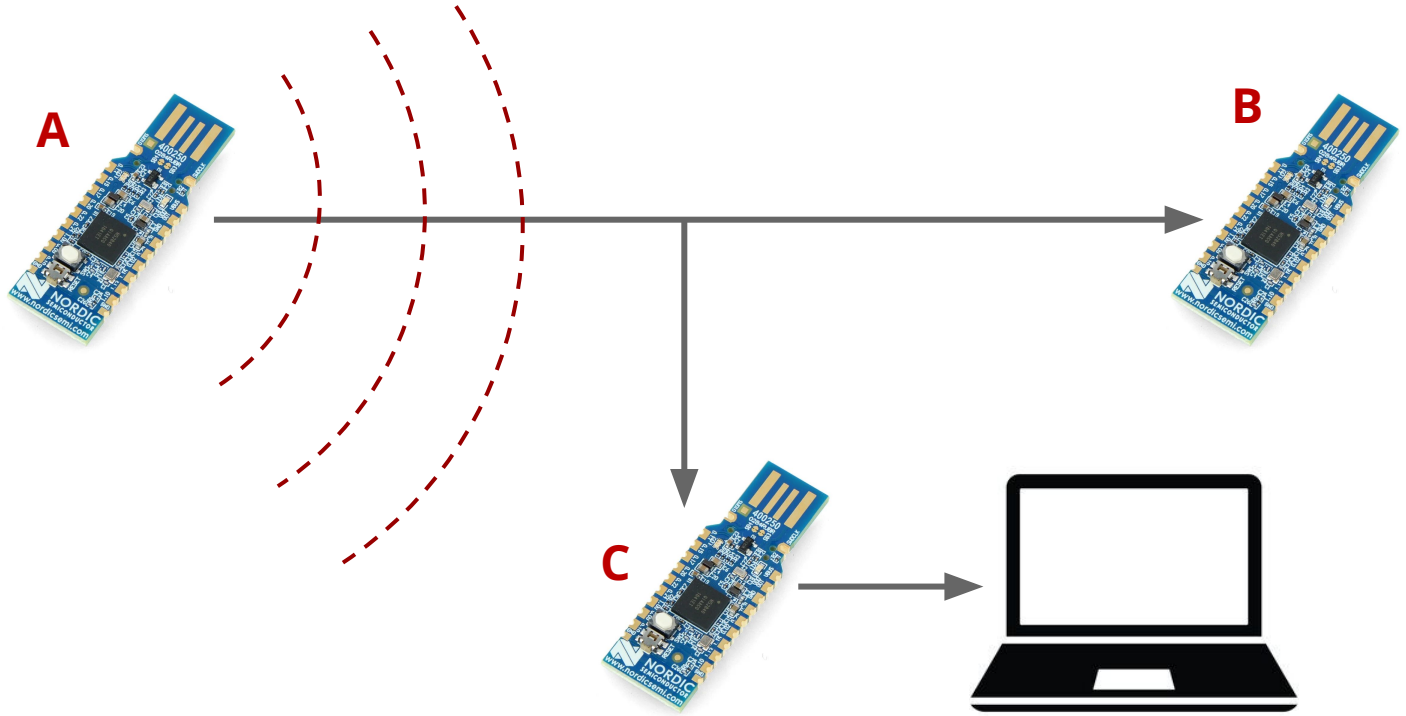
Live Demo



Live Demo



Live Demo



Course Objectives

- Learn to code using the **C programming language**.
- Learn to use **Contiki-NG** to program IoT devices.
- Learn to use **Cooja** to simulate Wireless Sensor Networks.
- Program **nRF52840** dongles to disseminate and aggregate data in the real world.

What is Coding?

Coding is the process of writing and creating **instructions in a programming language** to instruct a computer to **perform specific tasks** or functions.

What is IoT?

IoT stands for "**Internet of Things**." It refers to a **network of interconnected physical devices**, vehicles, buildings, and other objects embedded with **sensors**, software, and connectivity, allowing them to **collect and exchange data** over the internet.

Programming Languages

- C
- C++
- Java
- Python
- PHP
- Javascript
- MATLAB
- Assembly
- … many, many more!

History of C Programming Language

- Developed by Dennis Ritchie at Bell Labs in the early **1970s**.
- Evolved from an earlier language called “B”.
- **Standardized** the C language **in 1989** by the American National Standards Institute (ANSI)

Why use C?

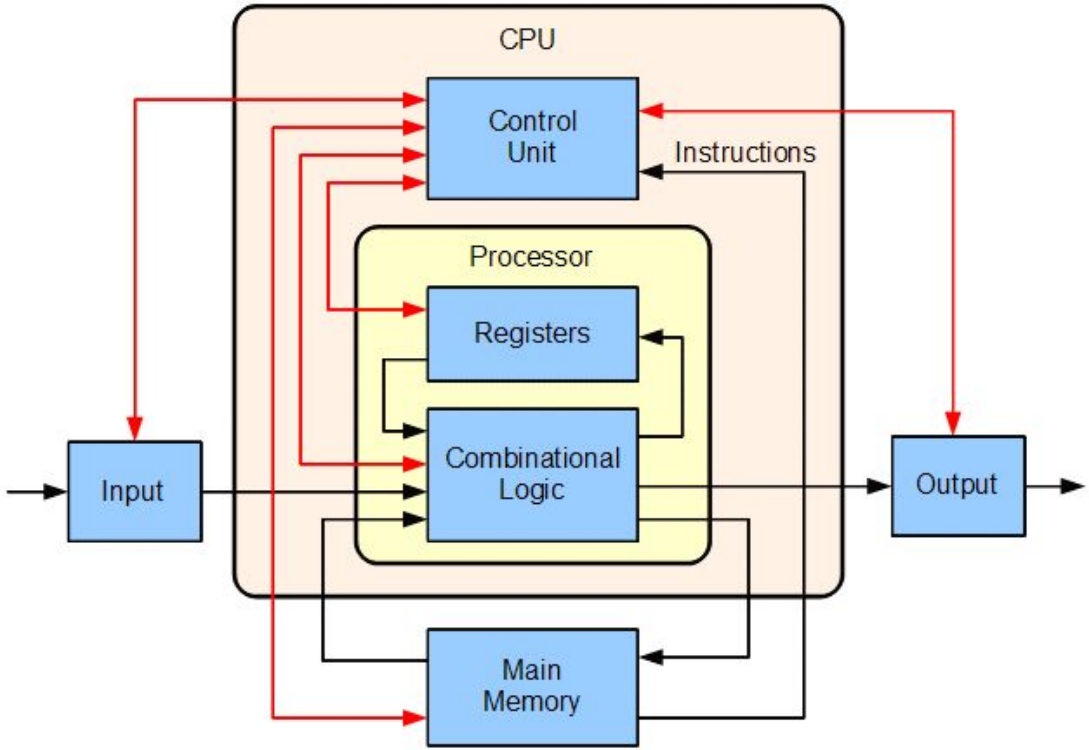
- **Portability**: designed to be platform-independent.
- **Efficiency**: fast and low-level memory access.
- **Versatility**: useful in a wide range of applications.
- Foundation of most embedded systems.

What is a Program?

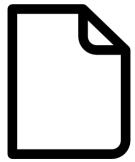
A program is a **set of instructions** or a sequence of code **written in a programming language** to tell a computer how to perform a specific task or solve a particular problem.

These **instructions** are designed to be **executed by the** computer's central processing unit (**CPU**).

Inside a CPU


CPU
Control Unit
Instructions
Processor
Registers
Combinational Logic
Input
Output
Main Memory

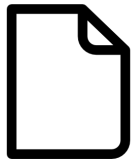
Machine Instructions (ARM Assembly)



program1.s

```
func1(int, int, int):
        add     r0, r0, r1
        add     r0, r0, r2
        bx      lr
```

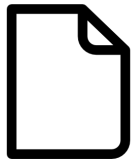
Machine Instructions (ARM Assembly)



program2.s

```
func2(int, int, int):
        cmp     r1, r2
        movlt   r1, r2
        cmp     r0, r1
        movlt   r0, r1
        bx      lr
```

Anatomy of a C Program

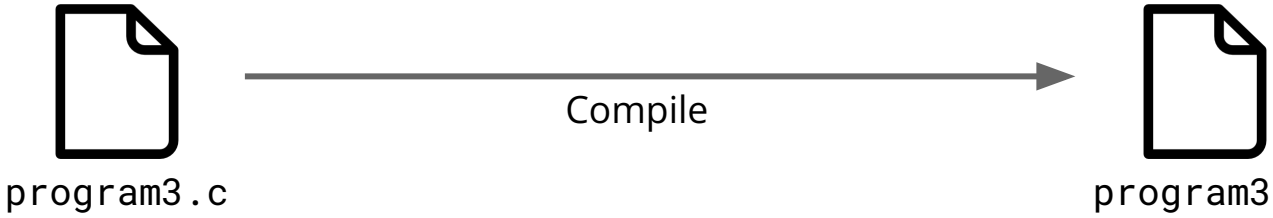


program3.c

```c
#include <stdio.h>


int main() {
    printf("Hello, World!\n");
    return 0;
}
```

Compiling a C Program

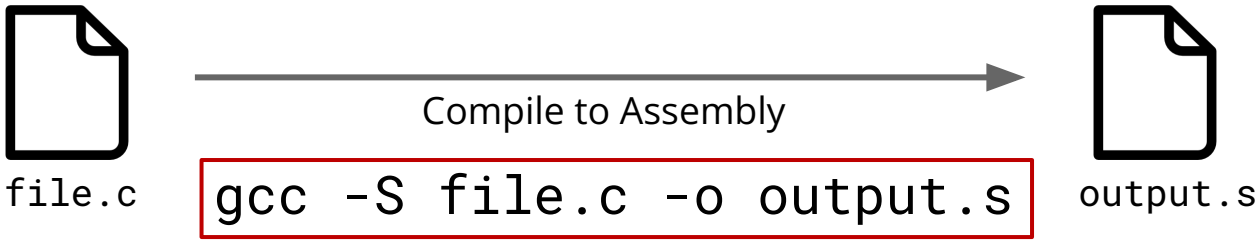


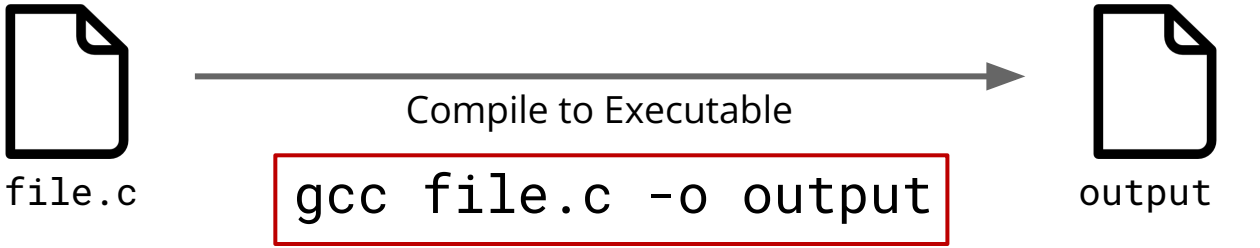

```c
#include <stdio.h>


int main() {
    printf("Hello, World!\n");
    return 0;
}
```

```
.LC0:
        .ascii  "Hello, World!\000"
main:
        push    {r3, lr}
        movw    r0, #:lower16:.LC0
        movt    r0, #:upper16:.LC0
        bl      puts
        movs    r0, #0
        pop     {r3, pc}
```

Compiling a C Program





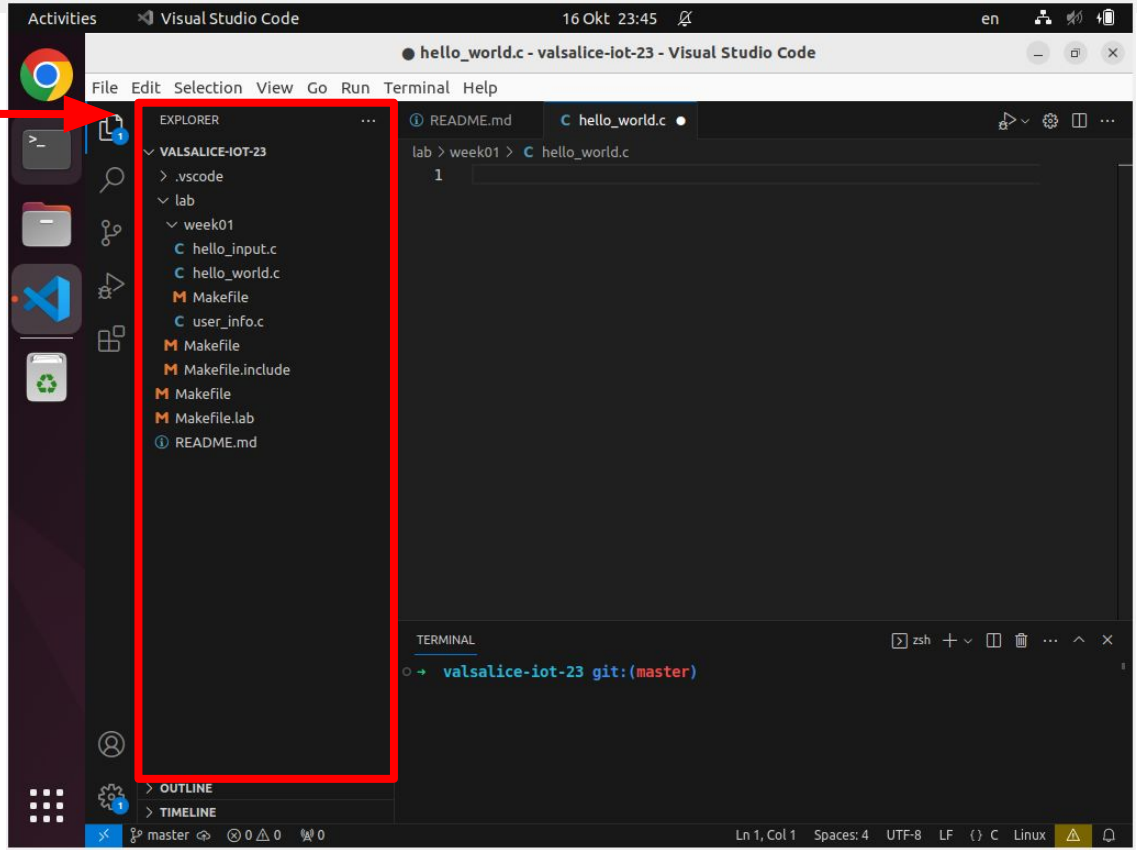
Virtualization and Ubuntu

VirtualBox and VSCode Setup

(Live Setup)

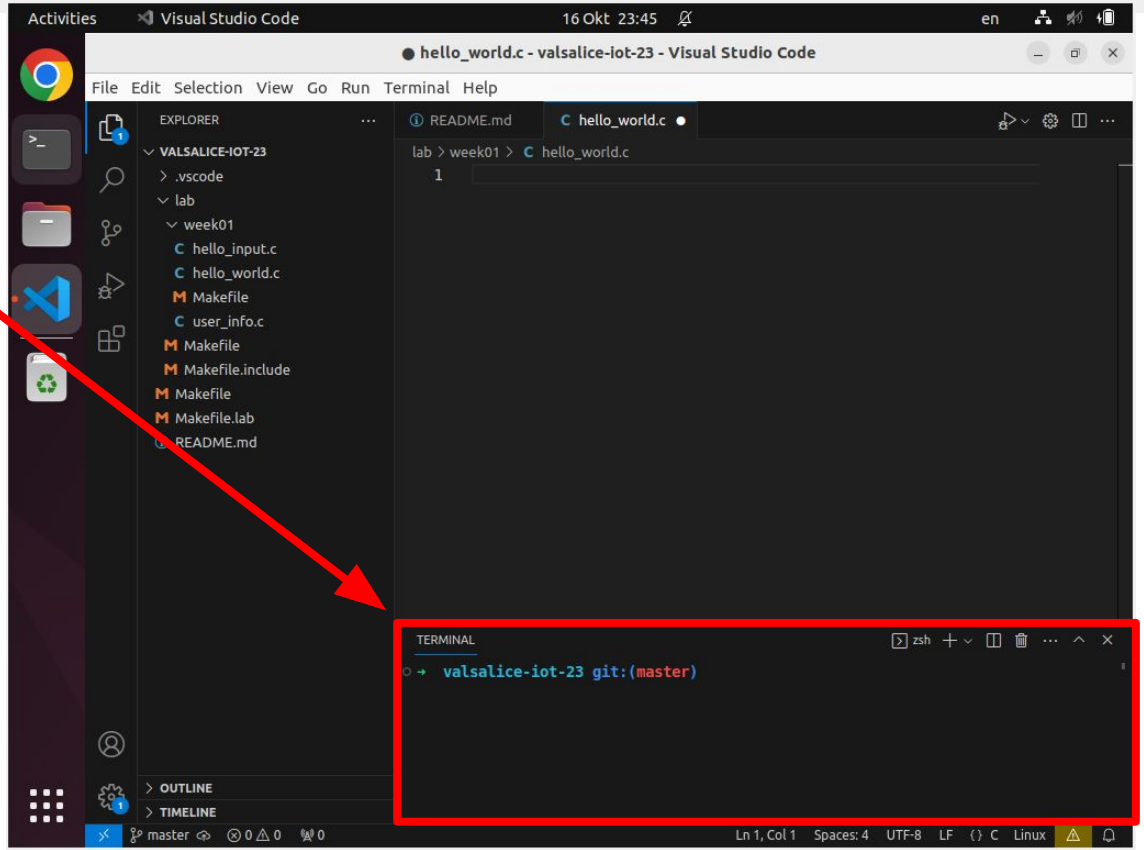
VSCode Structure

File Explorer



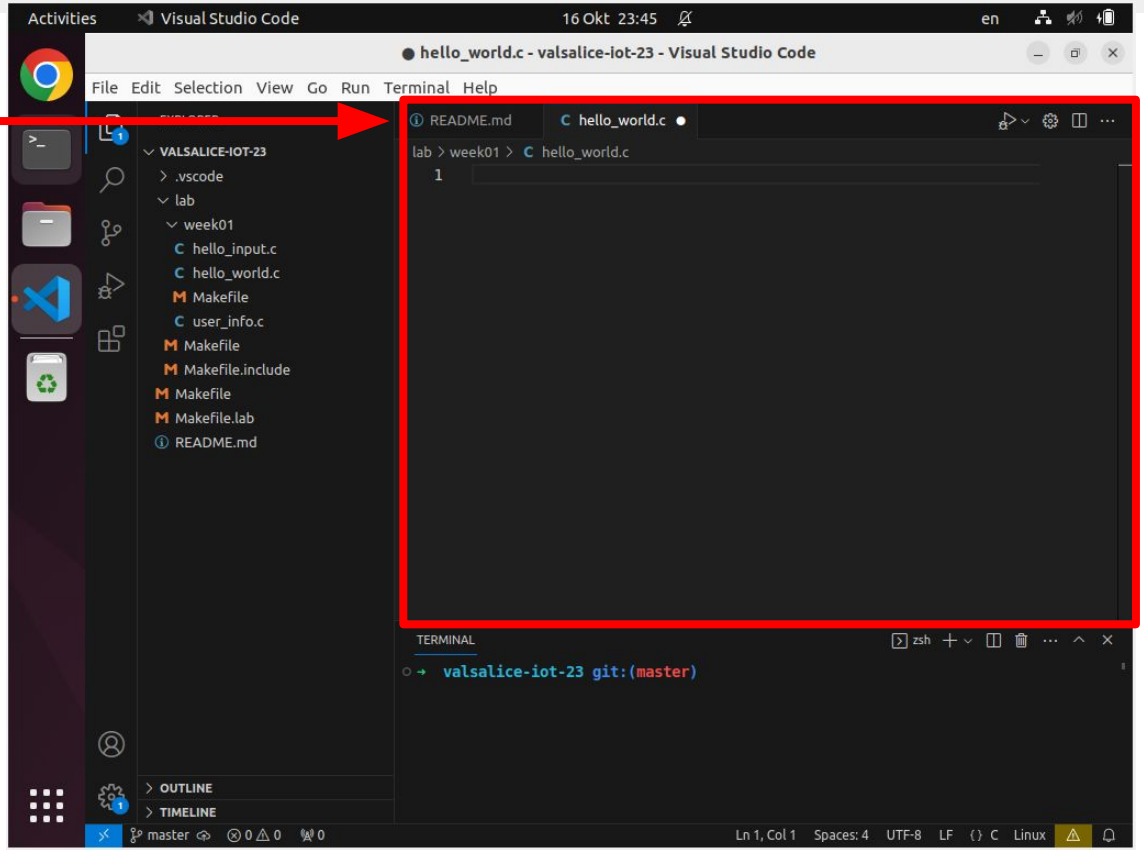
VSCode Structure

Terminal



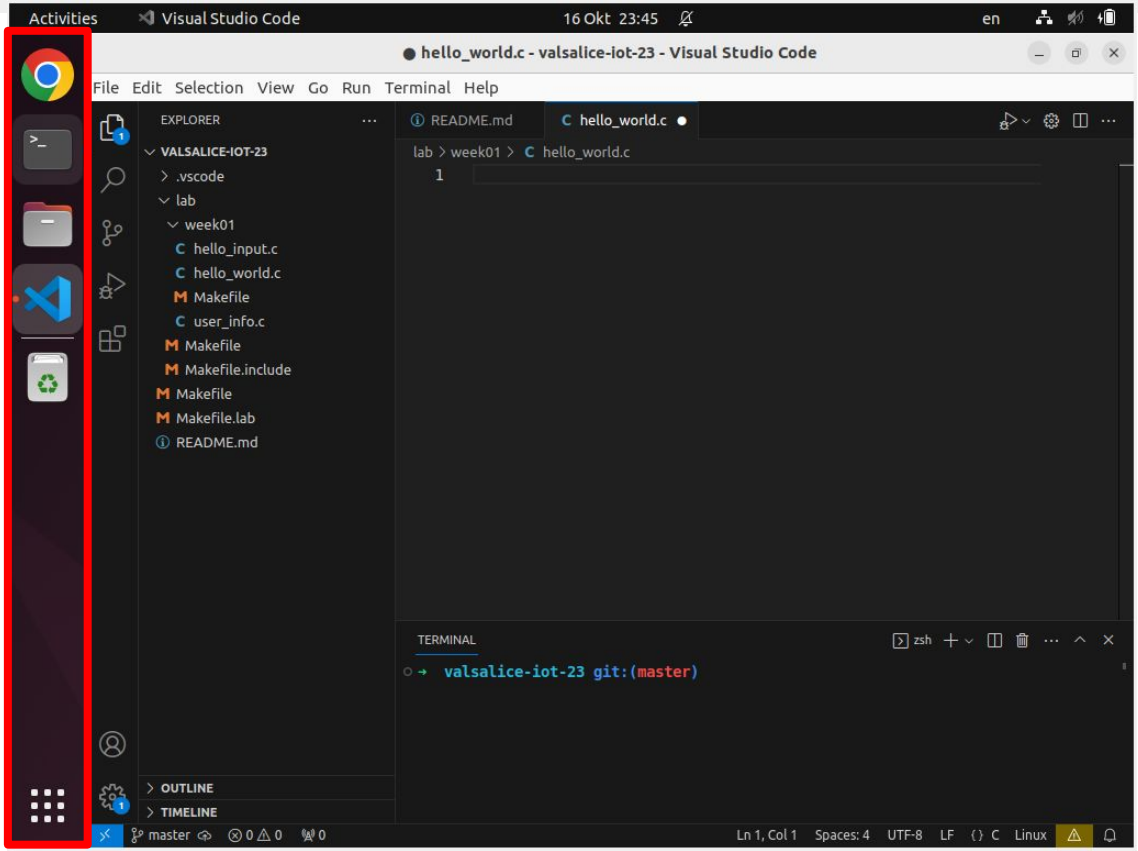
VSCode Structure

Code Editor



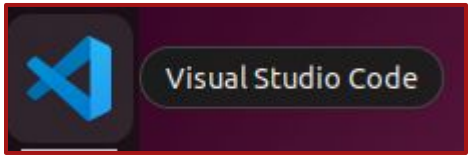
VSCode Structure

Ubuntu Programs



Prepare the Coding Environment

1. Open the Virtual Machine **nRF52840LAB**
2. Open **Visual Studio Code**



3. From the Terminal:

```
make setup
```

4.
```
valsalice-iot-23 git:(master) make setup
Enter your username:
```

5.
```
Password
```

6.
```
✅ Repository setup complete!
```

Exercise 1

Write, compile and execute a program (**`hello_world.c`**) that:

1. Prints out “Hello World!”

Exercise 1

Write, compile and execute a program (**`hello_world.c`**)

1

```
lab > week01 > C hello_world.c > ...
1   #include<stdio.h>
2
3   int main() {
4       printf("Hello World!\n");
5       return 0;
6   }
```

2

```
● → week01 git:(master) ✗ gcc hello_world.c -o output
```

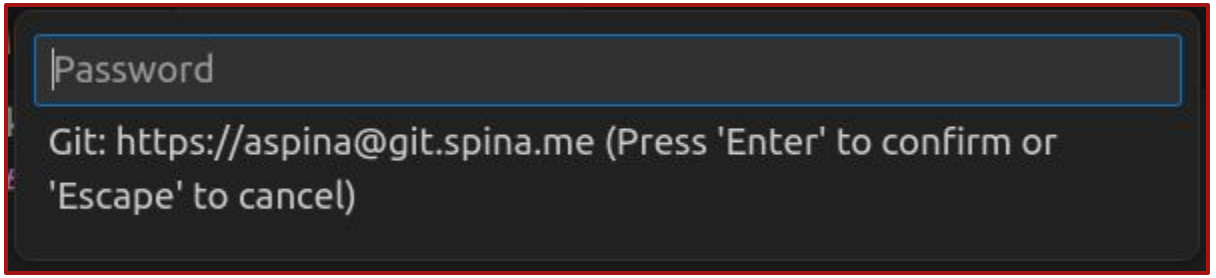
3

```
● → week01 git:(master) ✗ ./output
Hello World!
```

Save remotely your Changes

1. `make save`

2.
```
○ →  week01 git:(master) ✗ make save
  Enter commit message: Add hello_world.c file
```

3. 

4.
```
✅ Changes committed and pushed. All done!
```

End of Class

See you all next week!