Introduction to IoT

School Year 2023-2024

Valsalice

Course Structure

1	Introduction and Basics	
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

Open your Virtual Machines

1. Turn on your Laptops
2. Login using “User”
3. Open the **Virtual Box** program
4. Add the Virtual Machine (**Ctrl + A**)
5. Open the **VirtualBox** folder
6. Select the **nRF52840LAB** file
7. Click **Start**

Recap: hello_world exercise

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

1

```c
#include <stdio.h>
int main() {
    printf("Hello, World!\n");
    return 0;
}
```

2

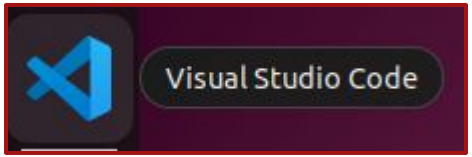
```
● → week01 git:(master) ✗ gcc hello_world.c -o output
```

3

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

Prepare the Coding Environment

1. Start 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!
```

Basic Input/Output Functions

- **`printf`**: C function for formatted output

```c
printf("Hello, World!\n");
```

- **`scanf`**: C function for formatted input

```c
char name[50];
scanf("%s", name);
```

Exercise

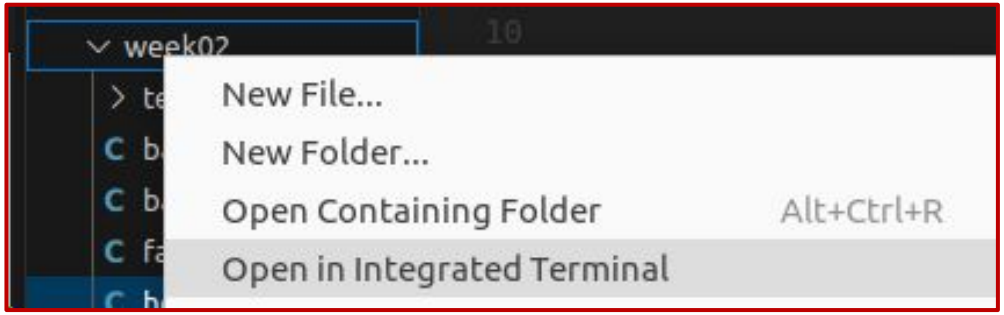
Write, compile and execute a program (**`hello_input.c`**) that:

1. Asks you to input your name
2. Says hello by printing out your name

TIP: Remember to use **`printf`** and **`scanf`**!

Compilation and Execution

Open `lab/week02` in the Terminal and use new `make` commands

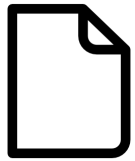
1. 

2. `make hello_input.build`

```
● → week02 git:(master) ✗ make hello_input.build
mkdir -p build
gcc -o build/hello_input hello_input.c
```

3. `make hello_input.run`

Exercise - Implementation



hello_input.c

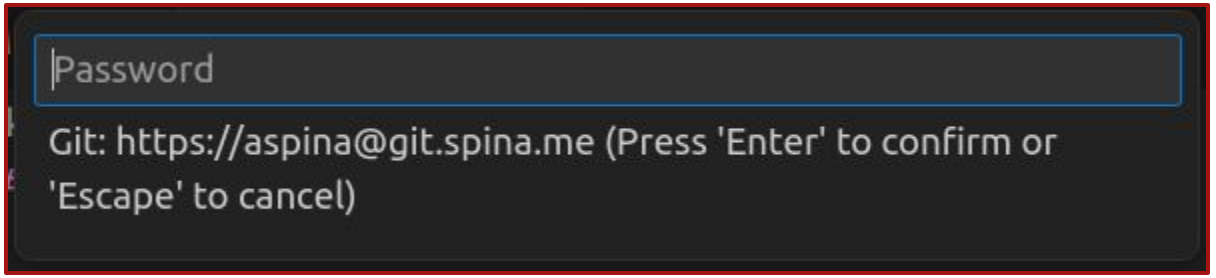
```c
#include <stdio.h>
int main() {
    char name[50];
    printf("Enter your name: ");
    scanf("%s", name);
    printf("Hello, %s!\n", name);
    return 0;
}
```

```
● → week02 git:(master) ✗ make hello_input.run
gcc -o build/hello_input hello_input.c
Enter your name: Alberto
Hello, Alberto!
```

Save remotely your Changes

1. `make save`

2.
```
➜  week02 git:(master) ✗ make save
Enter commit message: Add hello_input.c logic
```

3. 

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

Data Types

C has a number of primitive data types:

int `42` `1200` `1_200` `-3`

float `3.14` `0.00001` `-2.1`

char `'A'` `'@'` `'\n'`

bool `true` `false`

Strings are *NOT* a primitive data type, and have special syntax.

strings `"Hello"` `"A"` `"I am a full sentence!"`

Variables

A variable is a named container that stores data or values.

```
int x = 42;
float y = -0.12;
char w = 'A';
char z[50] = "Full sentence";
```

Booleans require a custom include statement:

```
#include <stdbool.h>
bool hello = true;
```

Variables

Variable declarations must contain a:

1. Type
2. Variable name, allowing underscores (_).
3. Equals sign (=)
4. End with a semicolon (;)

```
int x = 42;
float y = -0.12;
char w = 'A';
char z[50] = "Full sentence";
int valid_variable_name = 2000;
```

Format Specifiers

Format specifiers specify how data should be formatted or interpreted.

```c
int num = 123;
char name[50] = "John";

printf("The integer is: %d\n", num);
printf("The name is: %s\n", name);

scanf("%d", &num);
scanf("%s", name);
```

Type	Format Specifier	Example
char	%c	'A'
string	%s	"House"
int	%d	100
float	%f	6.98

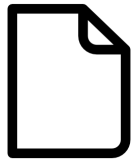
Exercise

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

1. Asks you to input your name
2. Asks you for your age
3. Asks you for your height
4. Says hello by printing out all the information you have given to it

To execute your code: `make user_info.run`

Exercise - Implementation

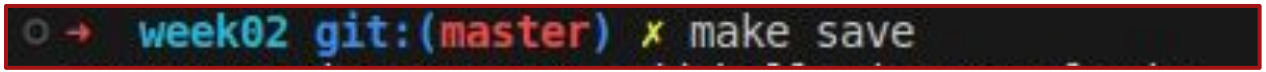


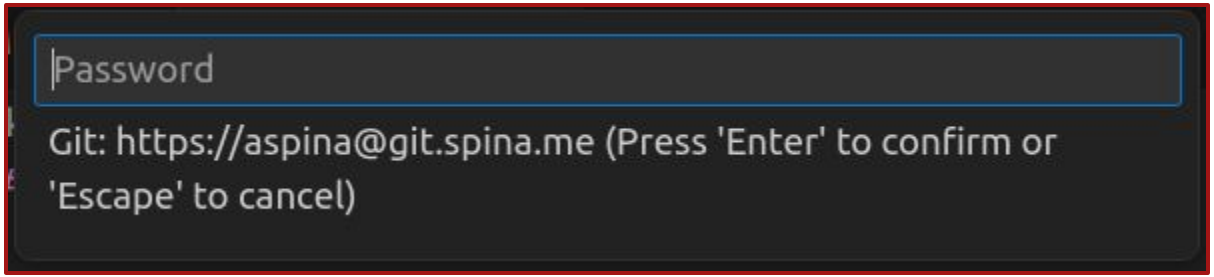
user_info.c

```c
#include <stdio.h>
int main() {
	char name[50];
	int age;
	float height;

	printf("Enter your name: ");
	scanf("%s", name);

	printf("Enter your age: ");
	scanf("%d", &age);

	printf("Enter your height in meters: ");
	scanf("%f", &height);

	printf("%s has age %d is %.2f meters tall.\n", name, age, height);
	return 0;
}
```

Save remotely your Changes

1. `make save`

2. 

3. 

4. 

End of Class

See you all next week!