ECOLE NATIONALE
SUPÉRIEURE
D'INFORMATIQUE

الجـــمهوريـــة الجزائريـــة الديمـــقراطيـــة الشـعبيـة
وزارة التعليـــم العـــالي والبـــحث العلمـــي
People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

Intelligent and Communicating Systems, ICS

2^{nd} Year Specialty SIL G1

Lab Report n°01

Title:

Introduction to using and manipulating a microcontroller (Arduino and sensors)

Studied by:

First Name: Yasmine

Last Name: DINARI

E_mail: ly_dinari@esi.dz

A. Theory

1. Microcontrollers

A microcontroller is a compact integrated circuit that acts as a small, self-contained computer designed for specific tasks in embedded systems. It combines a CPU, memory (RAM, ROM ...), and input/output peripherals on a single chip, enabling it to interact with sensors, actuators, and other components to control electronic devices.They are used in a wide range of applications, including home appliances, automotive systems, and medical devices. They collect data through input interfaces, process it using the CPU, and generate output responses. Their low cost, energy efficiency, and compact size make them ideal for embedded systems requiring reliability and autonomy.

Microcontrollers are based on two main architectures: **Harvard** and **Von Neumann**. The Von Neumann architecture uses a single memory space and bus for both data and instructions, which simplifies design but creates a bottleneck as the CPU cannot fetch both simultaneously. In opposite, Harvard architecture separates memory and buses for data and instructions, enabling parallel access that improves speed and efficiency. This makes Harvard architecture ideal for real-time applications.

Architecture	**Von Neumann**	**Harvard**
Memory	Single shared memory	Separate memory for data and instructions
Bus Structure	Single bus for data and instructions	Separate buses for data and instructions
Performance	Slower due to sequential access	Faster due to parallel access
Complexity	Simpler hardware	More complex hardware

2. LED and Pushbutton

An LED is an **output** device that converts digital signals from a microcontroller into visible light, providing visual feedback such as ON/OFF states. A pushbutton, is an **input** device that sends a digital signal (HIGH or LOW) to the microcontroller when pressed, allowing user interaction. To ensure stable and predictable behavior in circuits with pushbuttons, pull-up or pull-down resistors are used.

3. Arduino

Arduino is a family of microcontrollers consisting of development boards and an IDE that allows programming and interaction with sensors and external components. It is an

open-source platform that makes it accessible to beginners. Arduino boards, such as the popular Arduino Uno, feature digital and analog input/output pins, enabling them to control LEDs, motors, sensors, etc.

3.1. Hardware aspects

It includes a microcontroller (such as the SAMD21 Cortex-M0+), which integrates a CPU, memory, and I/O peripherals. The hardware provides GPIO (General Purpose Input/Output) pins for digital (8) and analog input/output(7) operations and PWM Pins (12), enabling interaction with external components like sensors or LEDs. It features memory types such as Flash (256 KB) for program storage, SRAM (32KB) for runtime variables, and EEPROM for persistent data. Many boards also include advanced features like WiFi or Bluetooth for IoT applications.

3.2. Structure of an Arduino program

An Arduino program is structured into two main functions:

setup()

Runs once when the board starts or resets. Used to initialize settings like pin modes or serial communication.

loop()

Runs continuously after setup(). Contains the main logic of the program.

3.3. Libraries and functions

Arduino offers many useful libraries to help us communicate with sensors and other inputs. To use a library, simply include it in your code with the include directive. This allows you to call the library's functions in your program and avoid repetition of the same block of code.

4. Installing and configuring the necessary tools for Arduino development

4.1. Study, connection of Arduino with the ”test lab,” and testing

First, go to the official Arduino website and download the Arduino IDE. Next, select the MKR1010 board model. Then, connect your Arduino to the computer using a USB cable and select the appropriate port. After that, install any necessary packages. Finally, test that everything is working correctly by using a basic program in the IDE, such as the Blink LED program.

4.2. Connecting a Pushbutton/LED to digital IO

Connect the LED's anode to a digital output pin on the breadboard and the cathode to the GND, use digital pins for the output. For the pushbutton, connect one side to a digital input pin and the other to GND. Then, write code in the IDE to detect the button's state.

4.3. Arduino protection

To protect Arduino components:

- Input pins: Use a 10 kΩ resistor as a pull-up or pull-down to stabilize voltage and avoid floating states.
- Output pins: Add a resistor (220 Ω to 330 Ω) when connecting LEDs or other components to limit current.
- LEDs and pushbuttons: Always use a 220 Ω to 330 Ω resistor with LEDs to prevent overcurrent. For pushbuttons, include a 10 kΩ pull-up or pull-down resistor to ensure stable operation.

5. Introduction to microcontroller simulators

Microcontroller simulators are software tools that mimic the behavior of microcontrollers, enabling developers to test and debug programs without using physical hardware. They simulate the microcontroller's architecture, memory, and peripherals, making them cost-effective and safe for experimentation. These tools provide features such as real-time monitoring.

Popular simulators include **Proteus**, widely used for Arduino and PIC microcontrollers, and **Tinkercad** Circuits, ideal for beginners. They are particularly useful in education and early development stages, allowing rapid prototyping and learning in a virtual environment.

B. ACTIVITY:

1. Connecting a LED

1.1. Turning on the built-in LED

To turn on the built-in LED of the Arduino MKR WiFi 1010, follow these steps:

- Install the Arduino IDE from the official website.
- Connect the Arduino MKR WiFi 1010 to your computer using a USB cable.
- Open the Arduino IDE, go to Tools > Board, and select Arduino MKR WiFi 1010.
- In Tools > Port, choose the correct port for your board to establish a connection.
- Write or load a program, such as the ”Blink” example, to toggle the built-in LED.
- Click the upload button (arrow icon) to send the program to the board.
- The built-in LED will blink if everything is set up correctly.

```
// setup(): function runs once when the arduino starts
void setup() {
  pinMode(LED_BUILTIN, OUTPUT); // Set the built-in LED as an output
}
// loop(): Repeats indefinitely after setup() has finished executing
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // Turn on the LED
  delay(2000); // Wait for 2 second
  digitalWrite(LED_BUILTIN, LOW); // Turn off the LED
  delay(2000); // Wait for 2 second
}
```

Listing 1: Arduino built-in LED Blink Code (every 2 seconds)

setup()

Runs once when the Arduino starts. Used to initialize settings, such as pin modes.

pinMode(pin, mode)

Configures the specified pin as an `INPUT` or `OUTPUT`. In this case, `pinMode(LED_BUILTIN, OUTPUT);` sets the built-in LED pin as an output.

loop()

Runs continuously after `setup()` finishes executing. Contains the main logic of the program.

digitalWrite(pin, value)

Sets a digital pin to `HIGH` (ON) or `LOW` (OFF).

digitalWrite(LED_BUILTIN, HIGH);

Turns the LED on.

digitalWrite(LED_BUILTIN, LOW);

Turns the LED off.

delay(milliseconds)

Pauses execution for the specified time in milliseconds.

1.2. Turning on an external LED

To connect the hardware, start by plugging the Arduino board into your computer using a USB cable for power and programming. Place the LED on a breadboard, ensuring the longer leg (anode) is connected to one end of a resistor (220Ω or 330Ω). Connect the other end of the resistor to a digital pin on the Arduino (e.g., pin 12). The shorter leg of the LED (cathode) should be connected to the ground (GND) pin on the Arduino.

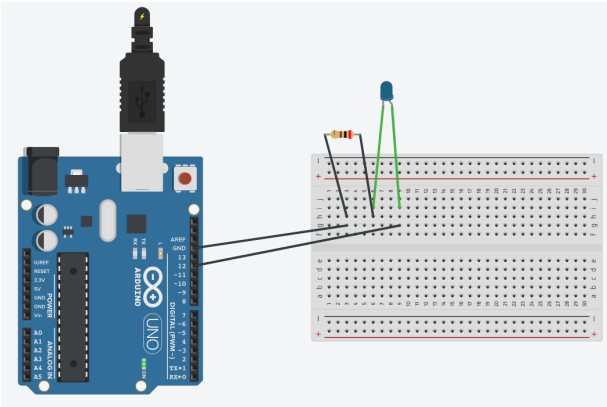

Figure 1: Virtual Arduino with an external LED on Tinkercad

```arduino
#define EXTERNAL_LED 12  // Define the external LED pin

void setup() {
  pinMode(EXTERNAL_LED, OUTPUT); // Set the external LED as an output
}

void loop() {
  digitalWrite(EXTERNAL_LED, HIGH);  // Turn on the external LED
  delay(2000);                       // Wait for 2 seconds
  digitalWrite(EXTERNAL_LED, LOW);   // Turn off the external LED
  delay(2000);                       // Wait for 2 seconds
}
```

Listing 2: Arduino external LED Blink Code (every 2 seconds)

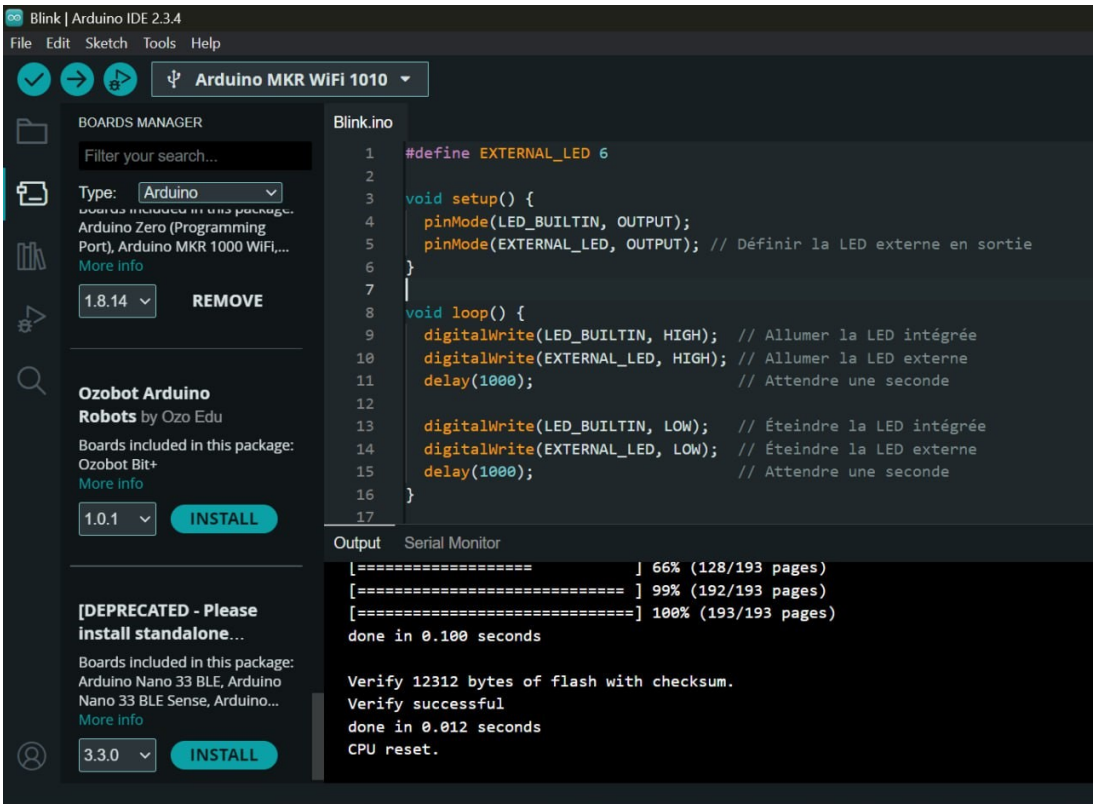

Figure 2: Complete Code on Arduino IDE

1.3. Other combination

This program controls both the built-in LED and an external LED, making them blink alternately. When the built-in LED is ON, the external LED is OFF, and vice versa, switching every second.

```arduino
#define EXTERNAL_LED 12  // Define the external LED pin

void setup() {
  pinMode(LED_BUILTIN, OUTPUT);  // Set the built-in LED as an output
  pinMode(EXTERNAL_LED, OUTPUT); // Set the external LED as an output
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // Turn ON built-in LED
  digitalWrite(EXTERNAL_LED, LOW);  // Turn OFF external LED
  delay(1000);                      // Wait for 1 second

  digitalWrite(LED_BUILTIN, LOW);   // Turn OFF built-in LED
  digitalWrite(EXTERNAL_LED, HIGH); // Turn ON external LED
  delay(1000);                      // Wait for 1 second
}
```

Listing 3: Arduino Internal and External LED Alternating Blink

2. Scan of wireless Wifi networks

2.1. Install the Library

To work with Wi-Fi on the MKR1010 board, we need to install the **WiFiNINA** library, which provides functions for handling network communication.

Steps to install:

- Open the Arduino IDE.
- Go to `Sketch > Include Library > Manage Libraries`.
- In the Library Manager, search for **WiFiNINA**.
- Click **Install** if it is not already installed.

After installation, include the library at the beginning of the sketch:

```arduino
#include <WiFiNINA.h>
```

Listing 4: Include the WiFiNINA library

2.2. Read and Understand Some Functions

The **WiFiNINA** library provides many functions for scanning available networks and managing connections. One important function is:

Scanning Available Networks

```
int numNetworks = WiFi.scanNetworks();
```

Listing 5: Scan for available networks

This function returns the number of detected networks and allows getting details about each one:

```
void scan_networks() {
    Serial.println("Scanning for available networks...");
    int numNetworks = WiFi.scanNetworks();

    for (int i = 0; i < numNetworks; i++) {
        Serial.print("Network ");
        Serial.print(i + 1);
        Serial.print(": ");
        Serial.print(WiFi.SSID(i));  // Get network name
        Serial.print(" (Signal Strength: ");
        Serial.print(WiFi.RSSI(i));  // Get signal strength
        Serial.println(" dBm)");
    }
}
```

Listing 6: Getting available networks

This function lists all available Wi-Fi networks along with their signal strength.

2.3. Memory Management

Efficient memory management is important when handling Wi-Fi operations on microcontrollers with limited ressources:

- **Freeing Unused Memory:** The function `WiFi.disconnect()` helps free memory by disconnecting from a network when no longer needed.
- **Using Static Buffers:** Instead of dynamically allocating memory, use fixed-size arrays when processing network data to avoid memory fragmentation.
- **Reducing Power Consumption:** Turning off Wi-Fi when not in use using `WiFi.end()` helps save power and free up resources.

Example:

```
WiFi.disconnect(); // Frees up memory used for connection
WiFi.end(); // Completely shuts down the Wi-Fi module
```

Listing 7: Memory Management functions

By carefully managing memory and connections, we can ensure a good performance of the MKR1010.

3. Conclusion

This lab focused on the practical use of the Arduino MKR WiFi 1010 microcontroller, emphasizing hands-on experimentation with LEDs, pushbuttons, and Wi-Fi functionalities. Using the Arduino IDE and Tinkercad simulator, we successfully implemented and tested various programs, including blinking built-in and external LEDs, alternating LED patterns, and scanning for wireless networks using the WiFiNINA library.

The lab also introduced key theoretical concepts such as microcontroller architecture (Harvard vs. Von Neumann), memory management (Flash, SRAM, EEPROM), and GPIO usage.

By combining theoretical knowledge with practical exercises, this lab provided a comprehensive foundation in microcontroller programming and embedded systems development. The use of Tinkercad highlighted the value of simulation tools for rapid prototyping and learning without physical hardware. Overall, this experience serves as a stepping stone toward more advanced IoT applications.