Microcontroller-based development

Digital electronics Homework (SG3-2024)

1. Presentation

In order to fully exploit the capabilities of a microcontroller, it is necessary to have sound knowledge of its internal functioning (registers, timers,...):
However, there are rapid prototyping tools that make it possible to reach a satisfactory (but non-optimized) result very quickly. Among these tools, the "Arduino" family includes development boards (hardware) and a development environment and many libraries (software).

The objective of this activity is to allow you to discover the world of microcontroller-based development using a development board created specifically for this, as well as a series of progressive exercises.

To carry out this session in the right conditions, you must install the prescribed software **BEFORE**.

Since the health crisis and the difficulty of sharing equipment, we now use an emulator for that Tutorial available on Tinkercad.com.

2. Hardware

For training purposes, the Arduino board is connected to a virtual extension board giving you access to a button, a potentiometer, 3 single LEDs and 2 RGB LEDs addressable in series (WS2812).

Component	Arduino pins (as in code)
Simple Led	10, 11, 12
Switch	2
Potentiometer	A0
RGB Led WS2812	3

Table 1 - Arduino/Shield pin mapping

3. Tutorial roadmap

The tutorial will make it possible to gradually reach the full use of the development board. It has four parts.

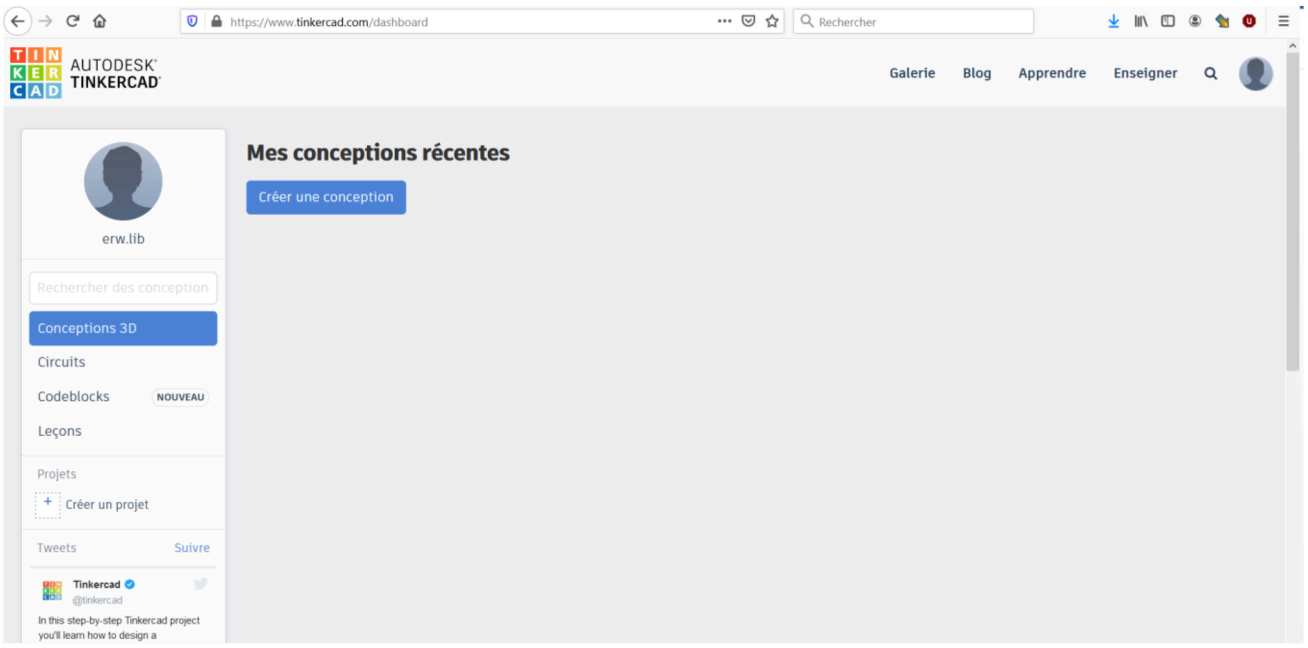
1. Installation and first contact with the framework
2. Basic structure of Arduino programs
3. Basic programs
4. Advanced Programs

To adapt to both those who discover the environment and those who have already used it, the subject is voluntarily long. The goal is not to finish the tutorial in the allotted time but to go as far as possible.

4. Installation and grip of Arduino

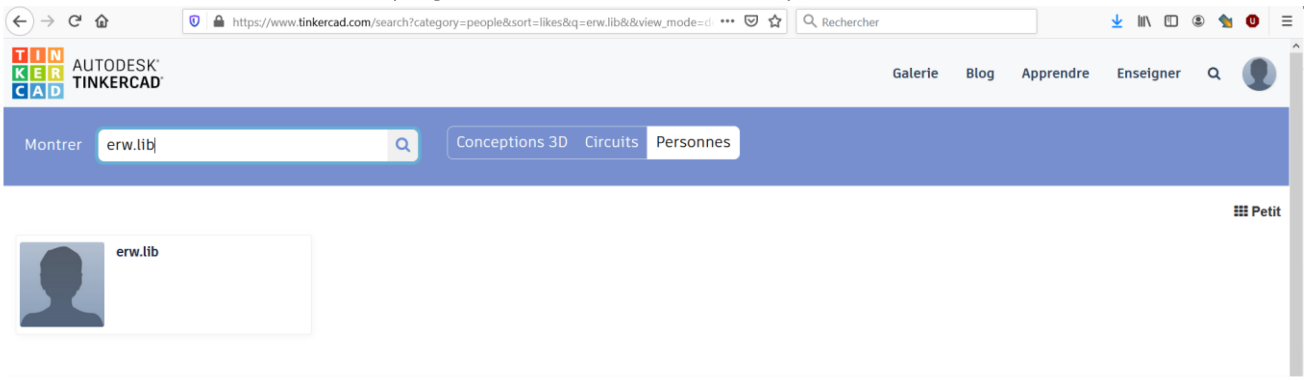
4.1. Creating an account and retrieving the schematic

We will use a simulator that is proposed by the Tinkercad community platform. You will need to create an account via the following link:
https://www.tinkercad.com/join (select the option « Créer un compte personnel »)
Once your access is activated, you should have an interface like the one below.

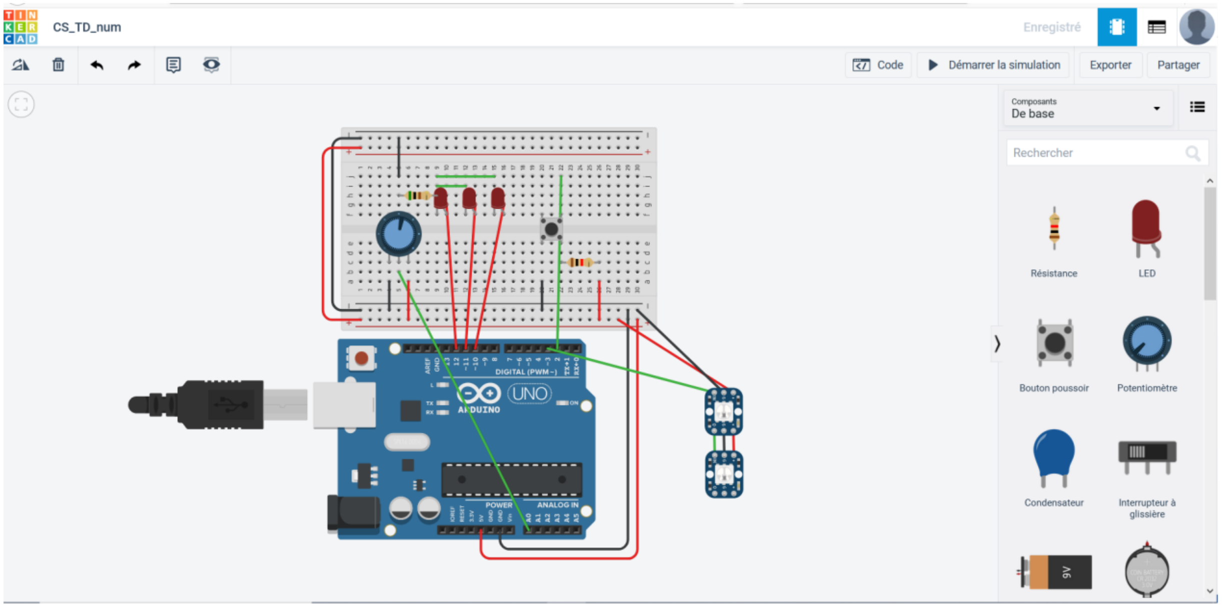


In this step-by-step Tinkercad project you'll learn how to design a

Click on the search icon in the top right corner and search for the person "erw.lib".



Click on the returned profile and you should find in the "Circuits" category the project "CS_TD_NUM" that you can copy and edit. You will then normally arrive at the following interface, showing the simulated assembly. The "Code" tab allows you to edit the code executed by the Arduino.



4.2. Installation validation: Blink

The "Hello World" of microcontroller programming is named Blink and makes blink a LED. This is the easiest way to check that the compiler is well configured and that everything is operational to transfer the compiled program to the microcontroller's Flash memory.

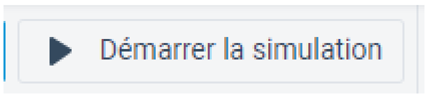
Blink opening

The Blink sample program is directly available in the simulator.

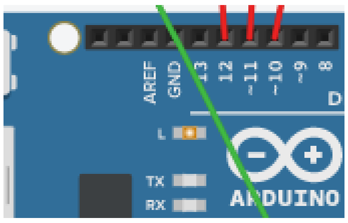
```
/*
 Blink
 Turns on an LED on for one second, then off for one second, repeatedly.
 This example code is in the public domain.
*/

int led = 13;
int LED1 = 10;
int LED2 = 11;
int LED3 = 12;
int SW = 2;
int POT = A0;

// the setup routine runs once when you press reset:
void setup() {
// initialize the digital pin as an output.
 pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
 digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
 delay(1000);               // wait for a second
 digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
 delay(1000);               // wait for a second
}
```

Compile and execute

We will not analyze this code right away. Let's start by compiling and executing it by clicking on



After an initialization phase, you can see the LED of the virtual Arduino board flashing.



5. Basic structure of Arduino programs

The language used for programming an Arduino board is C++-based (the keywords of C++ 11 are allowed--like those in C).

The structure of a program is broken down into three parts:

- A declaration area: which allows to describe the board (locations of LEDs, buttons, ...)
- The ***Setup ()***: executed only once it allows to configure the inputs/outputs of the board
- The ***Loop ()***: which will be executed in a loop as the name implies

```
// Déclarations
// ...

void setup() {
  // Une fois
}

void loop() {
  // En boucle
}
```

We will see, as we go through the training, that it is possible to add functions, classes, to use external libraries, ...

Each time you can (will) find a corresponding example among those provided in the IDE in the menu Files > Examples.

5.1. Declaring an output or an input

In the declaration area, it is possible to associate a variable name to the number of the corresponding pin on the Arduino board.

```
int led = 13; //  On most Arduino boards the pin 13 is a test LED
```

For outputs, it is mandatory to indicate the OUTPUT mode in the setup() area:

```
pinMode(led, OUTPUT); //  This pin will be able to deliver current
```

For digital inputs, it is mandatory to indicate the INPUT mode in the setup() area:

```
pinMode(led, INPUT); //  This pin will be able to deliver current
```

For inputs used in analog mode or interruptions, do not indicate anything.

6. Basic programs

To explore the possibilities of the board you will carry out a series of manipulations in a gradual way.

Reminder: Each time you can (will) find a corresponding example among those provided in the IDE.

6.1. Turn on an LED on the shield (red board)

Turn on one of the three LEDs, then two then the three using the previous instructions and the function: DigitalWrite (PIN, value);

For the number of the pins to be declared, refer to table 1

6.2. Detect a press of the button

Read the status of the push button and turn on one of the LEDs when pressed. Use the function: DigitalRead (PIN);

6.3. Interlude: Wait for a given time-blocking function

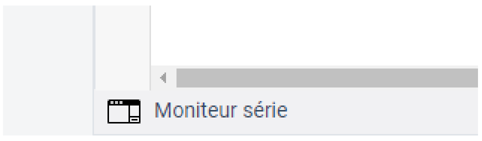
The function `delay(ms);` allows you to pause the program for a given time (ms) in milliseconds. One can notice that during that time the program/microcontroller is busy.

Make the led blink at a rate of 1Hz using delay.

Hint: *You've already seen this program in this document!*

6.4. Communicating with the outside world

The Arduino board has a serial link provided by the same USB cable used for programming. To access to the serial monitor, you must click on “Moniteur série” appearing at the bottom of the code window.



Using the serial class, set up a serial communication with the board and display a text ("Ok") every second on the Arduino IDE serial monitor.

In setup() add `Serial.begin(9600);` (9600 is the communication speed - baudrate)

In loop() add `Serial.print(val);`

Note 1: The function Serial.println(message); returns to the line after transmitting "message".

Note 2: Serial.print ("text"); displays *text* on the serial monitor.

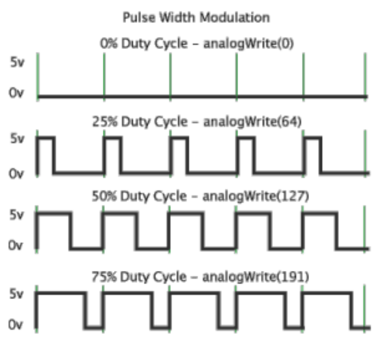
6.5. Read an analog value

Determine the position of the potentiometer using the function AnalogRead(PIN);

Use the serial link to "display" the percentage value.

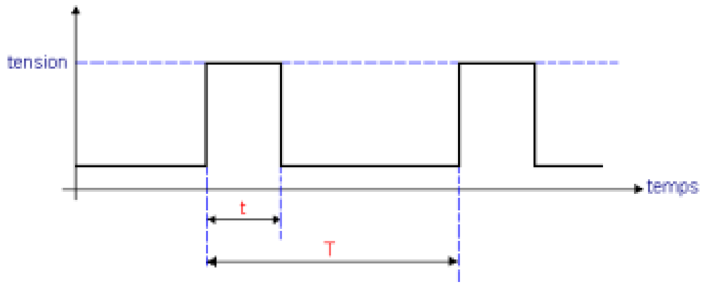
Note 1: The converter gives a 10-bit result.

6.6. Write an “analog” value on an output



Arduino does not allow to apply other voltage levels than "high" (logic 1) and "low" (logic 0). One solution is to simulate this using a pulse width modulation (PWM).

Idea: By rapidly alternate the output value between the "high" and the "low" state, the mean voltage of this signal can be expressed as follows: $V_{moy} = \alpha \times E$ With E the amplitude of the signal and α the duty cycle defined as the ratio between time (t) in the "high" state and the period of the T signal.



By applying such a signal to the diode, it is possible to control its luminous intensity.

The function `analogWrite(PIN, val) ;` sets up a PWM with a duty cycle equal to val/255 on the pin PIN.

Write a program that turns on the LED1, periodically changes the LED2 brightness and turns off the LED3.

6.7. How about we put everything together?

Write a program:

That changes the state of a LED (on or off) with each press of the button,

That allows the potentiometer to control the luminous intensity of the diode.

What if the press of the button changes the state of the LED between off or blink?

At this point, you should note that it is not possible to perform several actions at the same time, such as waiting for a press of the button and reading the value of the potentiometer while varying the luminous intensity of the diode.

It's now time to move on to an advanced usage of Arduino functions

7. Advanced Programs

7.1. Wait for a given time - non-blocking function

To give the control back to the program during the waiting stages, propose a solution to the exercise 6.7 for a non-blocking wait using the function: millis();

Index: *Examples > 02. Digital > BlinkWithoutDelay*

7.2. Detect a press on the button - clean method

Using the interrupts, retrieve a "button pressed" and "release button" information on the serial terminal.

Hints: search for attachInterrupt() *on the arduino.cc website*

```
attachInterrupt(digitalPinToInterrupt(pin), ISR, mode);
```

7.3. Using an external library to turn on the smart RGB LEDs

Since the Arduino community is very active, there are many libraries, including one dedicated to the control of WS2812 diodes (NeoPixel). This library is automatically installed with the Teensy tools.

To use it, simply add a LEDs object of the Adafruit_NeoPixel class to the declaration area:

```
#include <Adafruit_NeoPixel.h>
#define PIN 3 // le nom de la broche d'accès aux WS2812 sur la Teensy
#define LED_COUNT 2 // le nombre de WS2812 en série

// Create an instance of the Adafruit_NeoPixel class called "leds".
// That'll be what we refer to from here on...
Adafruit_NeoPixel leds = Adafruit_NeoPixel(LED_COUNT, PIN, NEO_GRB + NEO_KHZ800);
```

In setup() The LEDs object must be "started".

```
leds.begin();
```

Finally in the loop() it is possible to assign a different value to each LED.

```
leds.setPixelColor(0, 0xFF00FF);    // Set first LED to full red, no green, full blue
leds.setPixelColor(1, 0xFF, 0x00, 0xFF) ;  // set second LED to full red, no green, full blue
leds.show();
```

Create a program that independently changes the color of each led WS2812 depending on time.

7.4. Creating a state machine

You now have the necessary knowledge to make a state machine performing the following sequence:

- At the first press on the button, one single LED lights up
- At the second press on the button, two LED light up
- At the third press on the button, three LED light up
- At the fourth press on the button, all LEDs are back to their initial light off state

In any case, the potentiometer controls the light intensity of the LEDs.