At this project, we will take an in depth look at the hardware for using Arduino interrupts to control AC power through a triac.

This work will be composed by the following tasks:

1. **Task 1. Study and measurements of the zero-crossing electronic circuit**
2. Create two versions of a WEB-based WIFI wireless control parameters input. Web-based configuration user interface: configures the dimmer power and the selection of the power plug output. Options: (2.1) the ESP generates the http page or (2.2) the ESP connect to an existing http page using socket. In both cases we need: buttons (activation/deactivation), text input (?) text output (status) value input (% of power) or slider (idem).
3. Development of a smartphone user application (Java/Android) to configure the device. An Android user application or a web site (previous version or WEB server) seen on a smartphone available to control the hardware circuit (select the output and its output associated power).

Task 1. Study and measurements of the zero-crossing electronic circuit

Context: we use an electronics circuit, called *Zero Detector*, in charge of the generation of a pulse each time the wave signal crosses the zero volts. This output will be used to generate an interrupt on the programable board (Arduino/ESP *Processor*). The program is in charge of controlling the power driver (*Power Output*) by selecting to output (one of 8) to activate, turn-off or graduate the output power.

Task: we will study the components constituting the zero-crossing module, as well as measure and plot the signals on several point of the circuits to understand its behaviour.

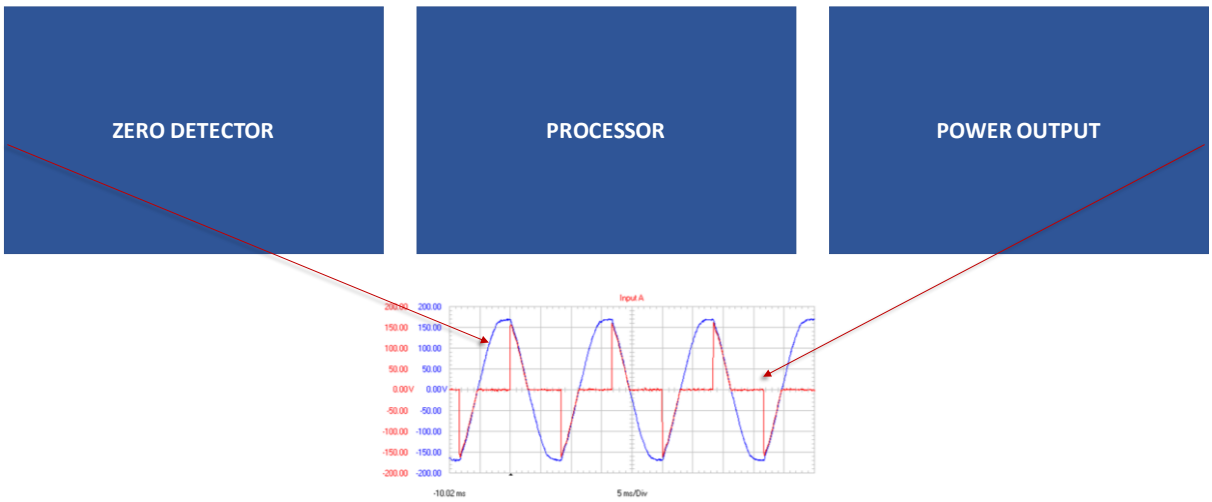


Figure 1: Overall system representation. The full system is component by the Zero Detector, the programmable board (Processor) and the Power Output.

Zero-cross detector

This circuit is in charge of sensing the zero transition of the sinus wave (220V input signal 50/60Hz frequency). The sinusoidal input line is rectified and isolated from the 1V output by using an optocoupler. The Optocoupler (phototransistor) is in charge of detecting when the signal crosses the zero volt to generate an output pulse. The output will be a square signal of 5V.

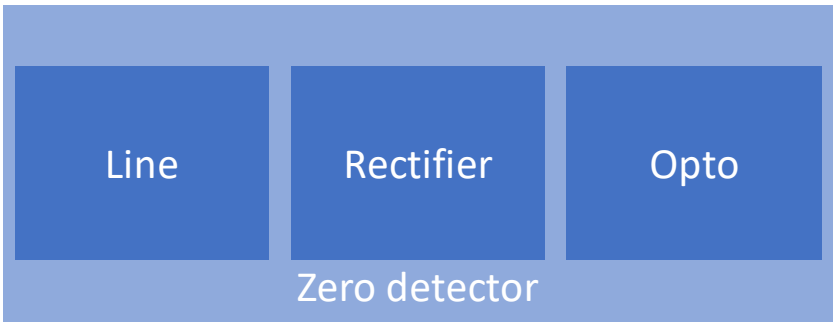


Figure 2 Zero crossing detector block diagram.

The 220V AC signal passes through 2 resistors (R1, R2) of 30k and is rectified by a diode bridge rectifier ($D_{1\text{-}4}$). The rectified signal *VIN_PR* (*1V* peak) travers a diode controlling the base of optocoupler (U1 4N27) output transistor. The output

VOUT will state at 0V when the diode drives the output transistor, otherwise it goes to 1V.

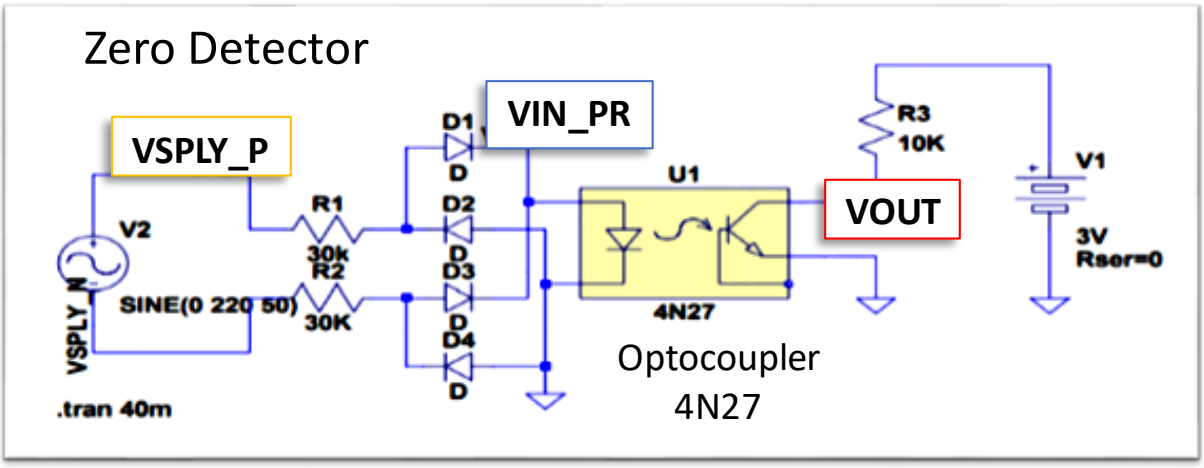


Figure 4 : Zero detector schematics. Circuit board connected to a 220V input power that is designed to detect the zero-cross thanks to the 4N27 optocoupler and delivers a 3V output.

The output of the optocoupler VOUT will drive the interrupt input of the processor (pin A2). The 10K Resistor (R3) polarizes the Opto-output with the processor voltage (V1), this is why the pin is connected to the pin +3V of the processor (ESP8622 or Arduino).

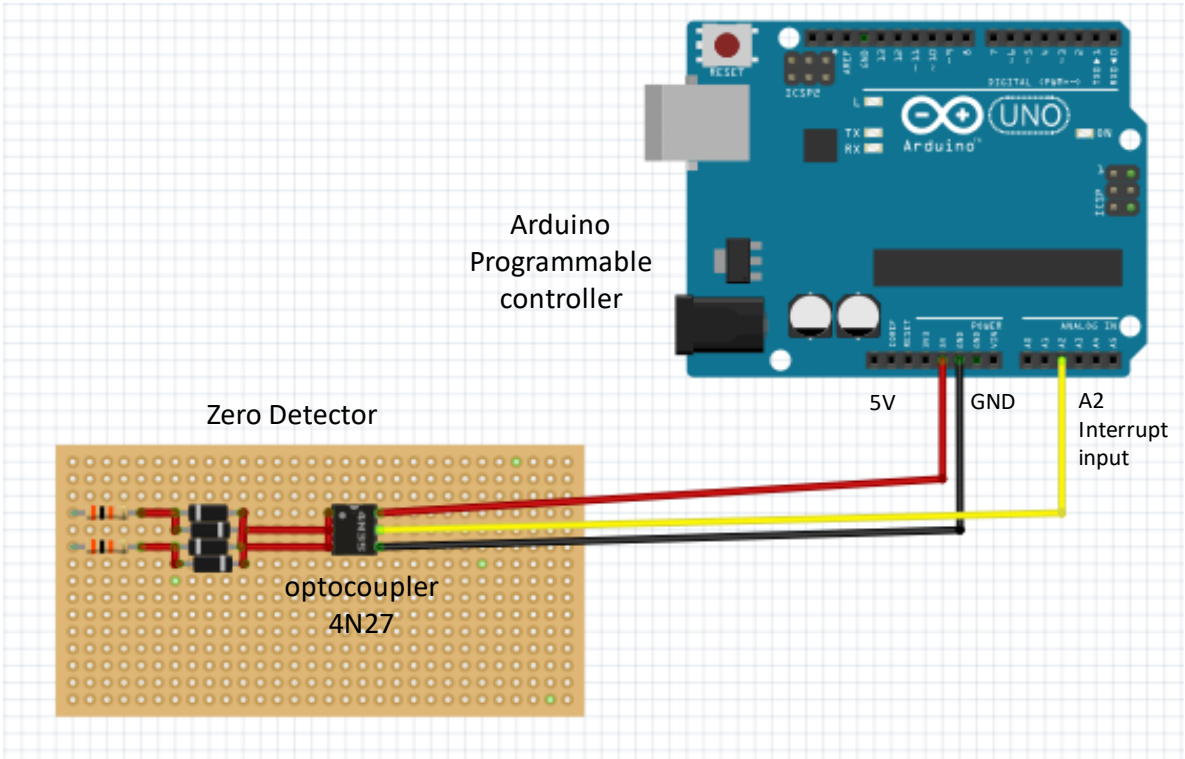


Figure 5 : Zero detector – Arduino board connection diagram. Fritzing schematic connecting the zero-detection circuit to the processor (Arduino board).

Note: the output of the optocoupler (4N27) will drive the interrupt (pin A2) routine.

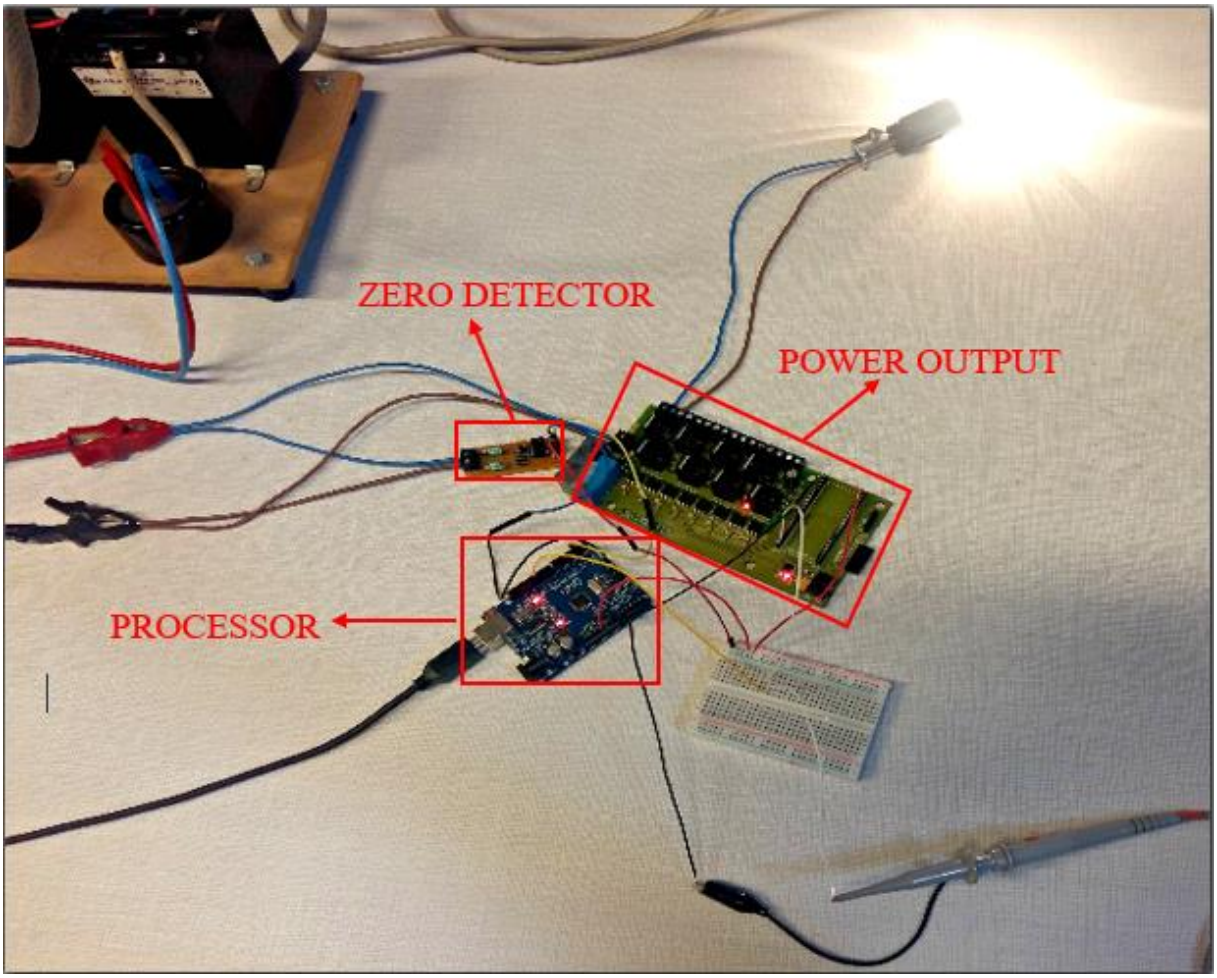


Figure 6 : Picture of the complete system controlled by an Arduino .

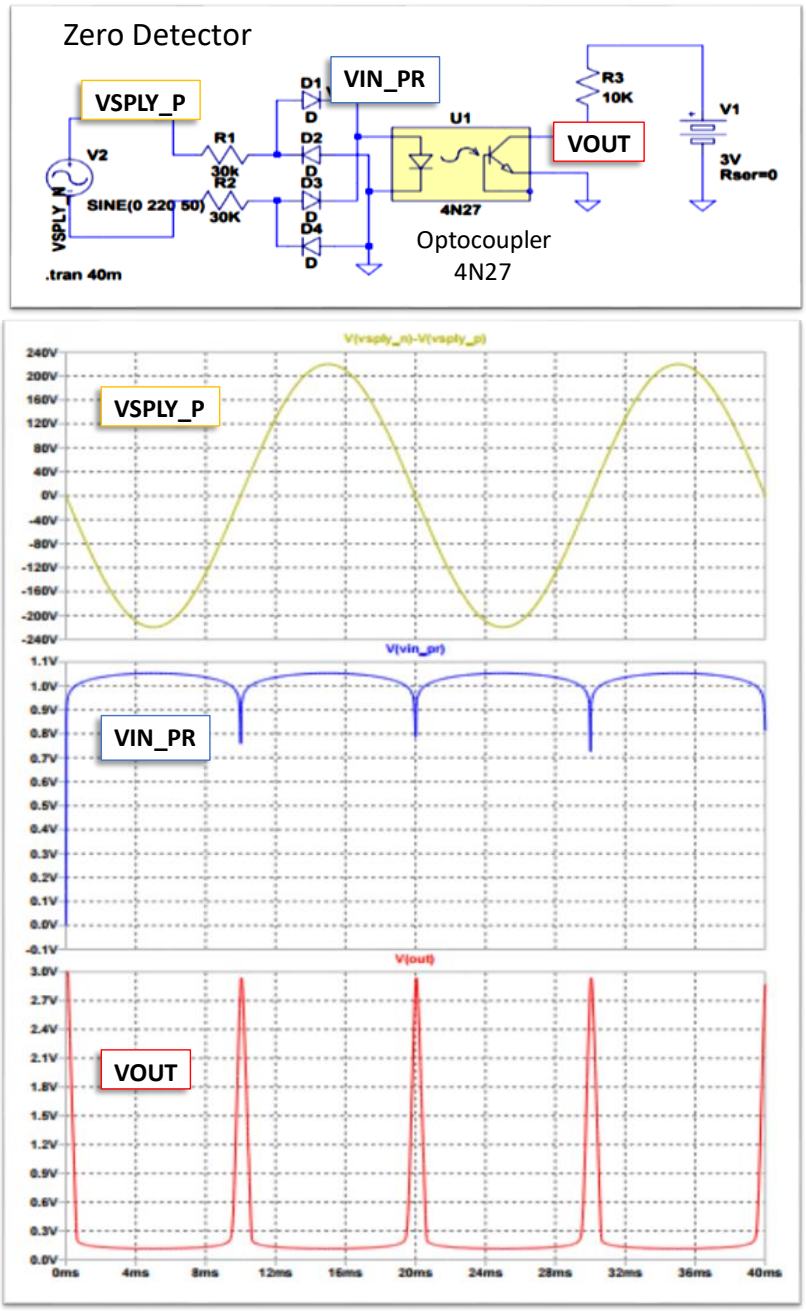


Figure 6 : Simulation of the Figure 3 on LTspice that draws power and optocoupler input and the general output .

You will find below the pseudo code:

I. Setup:

- Set LED 11 as output (it will be connected to the Triac)
- Attach the interrupt routine ***ZeroDetection*** to INT 2 (input from the Zero Cross Detection board).
- Initialize TimerOne library for the frequency we need.

II. Interrupt Routing ZeroDetection:

- Reset time step counter: StepCounter = 0
- Turn off out 11: LED = FALSE
- Calculate frequency step (us/step) using Arduino's timer at every zero cross .

III. Routine Timer

- **If** StepCounter >= **FireCount** then LED = TRUE
- Increase: StepCounter++

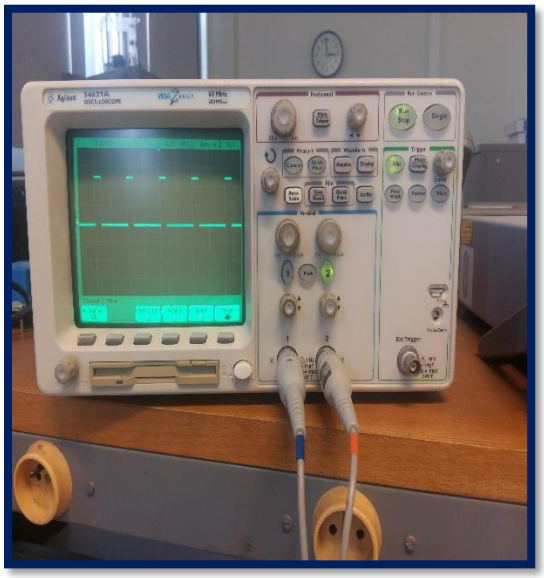
IV. The Main:

- **For** FireCount **from** 0 to MaxDuration **Loop**: wait 100ms

Testing:

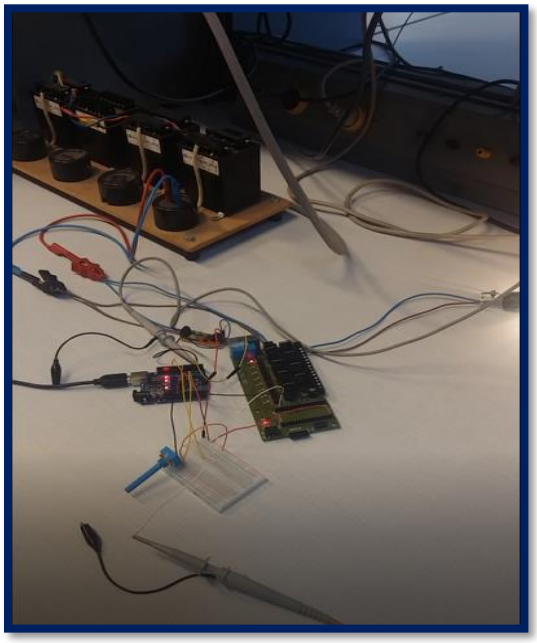
Test 1 : Arduino board driven by an oscillator (interrupt input)

A signal generator produces a square signal that drives the interrupt input pin of the Arduino board (pin A2). Upper square signal (5V period 20msec duration 5msec). As expected, the output goes to zero on a raising edge of the input signal (the interrupt). The main function increases the duration (FireCount) at each iteration step



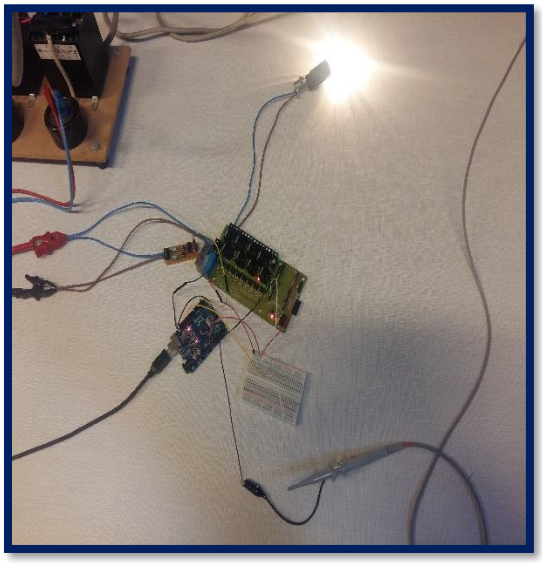
Results : Results are as expected , the variation has a very small error deficiency of 9 μs.

Test 2 : Controlling the full system with a potentiometer and visualizing the results on an oscillator .



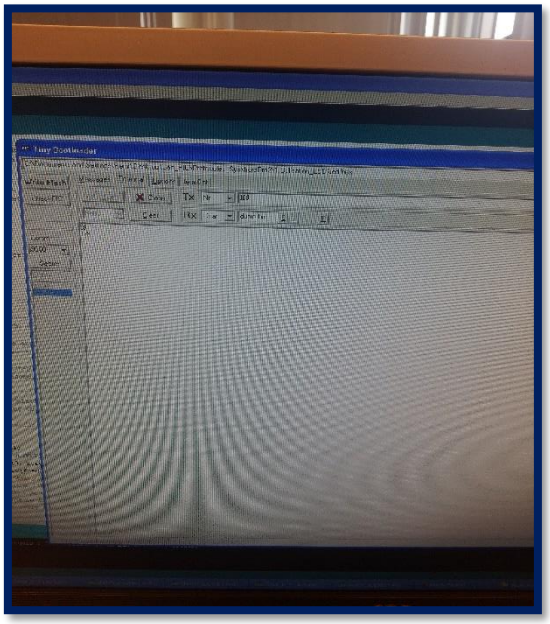
Results : The zero cross detection is up down as there's a displacement concerning the four quadrants of the triac.

Test 3 : Connecting a lamp to the previous system.



Results : The lamp varies accordingly to the potentiometer . The light is on starting from a certain value and a 100% shots the light down du to the quadrants problem (opposite reaction).

Test 4 : Using a serial communication and a terminal to enter values .

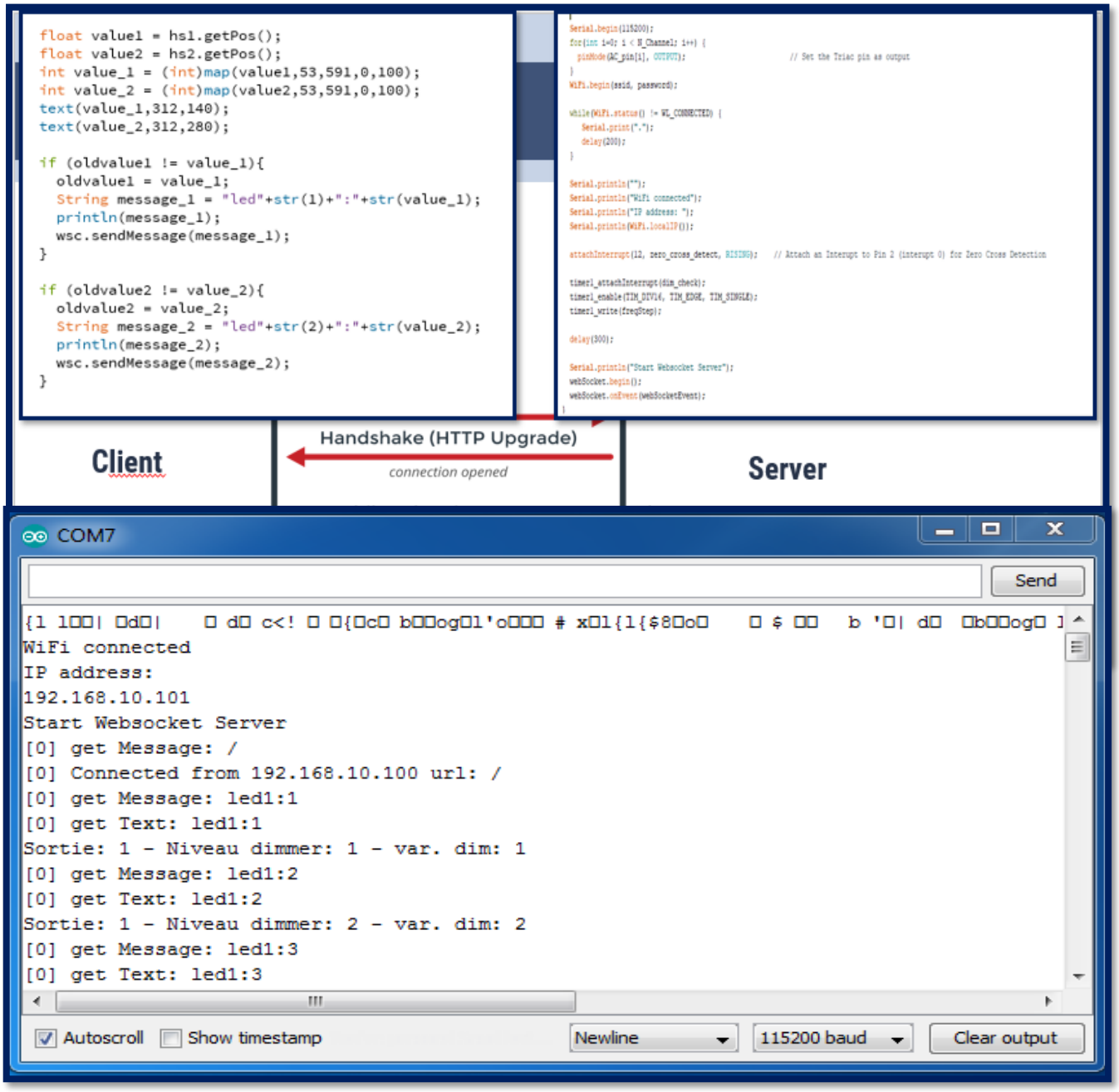


Result : Same results to the previous experiences , the lamp varies according to 100-x . X is the value given in the terminator .

Task 2. ESP8826 connection to the server and adequate application .

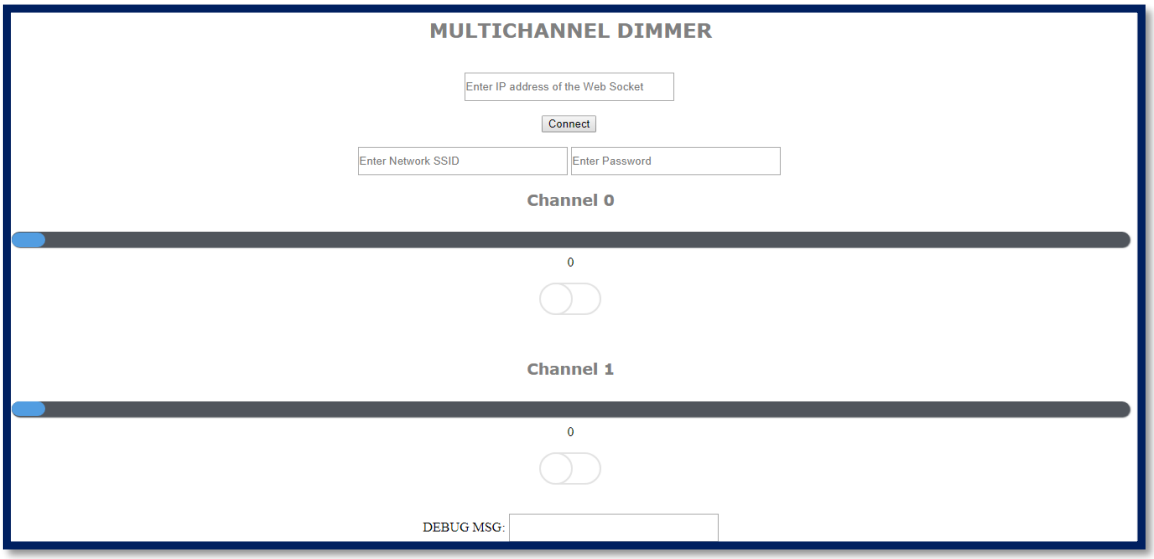
1) **Configuring ESP8826 as a web server** resulted in an infinite cycle of rebooting .

2) As a different alternative , **we have used Websocket as a protocol of connection** between the client and the server . Websockects allows to open a bi-directional communication channel and is much faster than classical protocols .



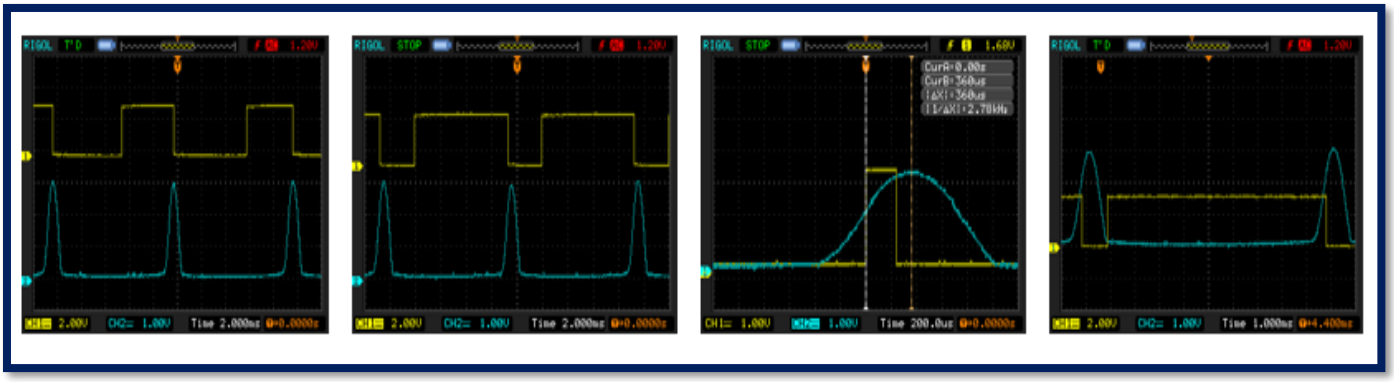
3) Connect to an existing web page:

In order to test the protocol connection on websocket , we used an existing web page found in www.instructables.com on the name of " Multichannel wireless light dimmer " of abhishek123 . The interface enable the user to know if there's a connection with the ESP8266 and sends messages of errors in case of a bug .



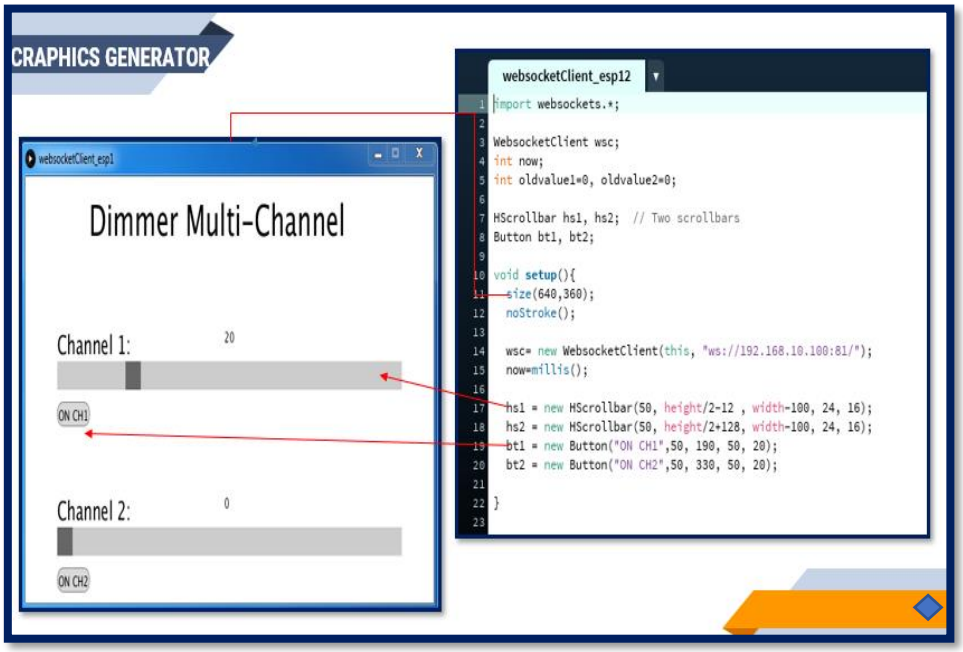
4) Visualization of the results on an oscilloscope:

Once the connection is set , the oscilloscope draws 2 signals . The upper signal is a square signal of a 5V adjusting accordingly to the value given on the interface . The lower signal shows when the zero cross occurs . A gab of 360 µs arised between the zero cross has happened and firing the triac .



5) Development of a processing application that will serve as a user interface:

The aim of using processing was to generate a user interface where a switch on-off button appears along with scrollbars to dim the energy level through different channels. First , we coded with java language to set the visual part of the interface then connected it to the ESP 8266 by adding a websocket library and by defining a protocol that collect the curser's position and convert it and send it .



6) Integration of the application into a web page:

After generating a processing application, we implemented it in a web page that the user can open in a browser. Processing code is writing in Java language. By adding the library processing.js and the code into a simple html page, the browser was able to identify the code now transcripted into JavaScript .

```html
<!DOCTYPE html>
<html>
    <head>
        <!-- En-tête de la page -->
      <title>Client1 - Processing.js Test</title>
      <script src="processing.js"></script>

    </head>

    <body>
        <!-- Corps de la page -->
        <h1>Processing.js Test -> Websocket Client v2</h1>
        <p>This is my first Processing.js web-based sketch:</p>
        <canvas data-processing-sources="Wsc_Client1.pde"></canvas>

    </body>
</html>
```

Conclusion :

In this project , we have been able to :

- Develop an interface user to switch on/off the PowerPlug or dim the energy level .
- Develop a microprocessor program to manage the zero detection , dimming level and multichannels .
- Connect the program to a web server via websockets .
- Use a wifi-chip that will allow the user to connect to the web page .

To take a look at the tutorial , you may visit : :https://youtu.be/lj0YIINehxg

Webography :

Links : https://www.instructables.com/id/Arduino-controlled-light-dimmer-The-circuit/

https://www.instructables.com/id/Multichannel-Wireless-Light-Dimmer/

https://forum.arduino.cc/index.php?topic=22512.15

https://www.instructables.com/id/ESP8266-Communication-With-Server-and-ESP8266/