UMONS
Université de Mons

Faculté Polytechnique

Servo-motor control using NI ELVIS Design Systems Development Board

Hardware Software Platforms Project

Antoine DUFRANE
Mathieu FOTSO
Louise LEGER

POLYTECH MONS

Supervised by Professor
Carlos VALDERRAMA

Mai 2019

pôle hainuyer

Contents

Introduction

This project was carried out at the UMONS Faculty of Engineering in April-May 2019 as part of the Hardware-Software Platforms course in first master of Electrical Engineering with Professor C. Valderamma. Its goal is to discover the Digital Systems Development Board by National Instrument with NI Elvis and more particularly to control a servo-motor with it.

This written tutorial explains in details the steps followed during the project, from the software needed to the actual control of a servo-motor, via the control of LED's and buttons on the board. Other work materials can be found on the *GitHub* site of the project (FPGA-with-NI-Elvis) like a video tutorial and a presentation, as well as all source codes.

Chapter 1

Preliminary installations

This chapter will expound the way of installing the Digital Systems Development Board (DSDB) as well as all software needed to use it. Finally, before getting started with the board, the way of creating a new project will be explained.

1.1 Software to install

In order to use the NI Elvis Digital System Development Board (DSDB), a few software need to be installed, in the following order. They can also be found in the *GitHub* site.

- **LabVIEW 2018** : `http://www.ni.com/download/labview-development-system-2018/7407/en/`
- **LabVIEW FPGA Module 2018**: `http://www.ni.com/download/labview-fpga-module-2018/7351/en/`
- **LabVIEW 2018 FPGA Module Xilinx Tools**: `http://www.ni.com/download/labview-fpga-module-2018/7353/en/`
- **DSDB Driver**: `http://www.ni.com/download/digital-systems-development-board-driver-fo 7441/en/`

1.2 Digital Systems Development Board Installation

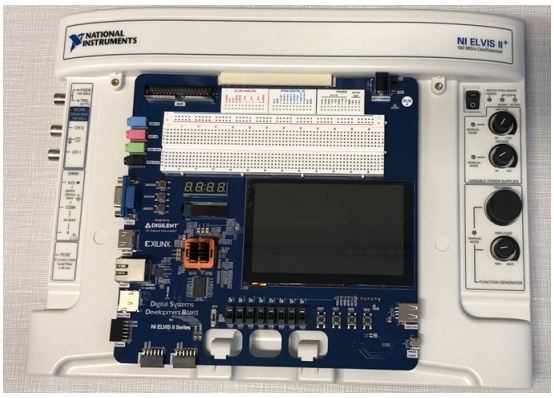
During this project, the Digital Systems Development Board was used by using the NI Elvis board by National Instrument as a power supply. Nevertheless, other ways of powering it can be used.

The steps to properly connect the DSDB are the following:

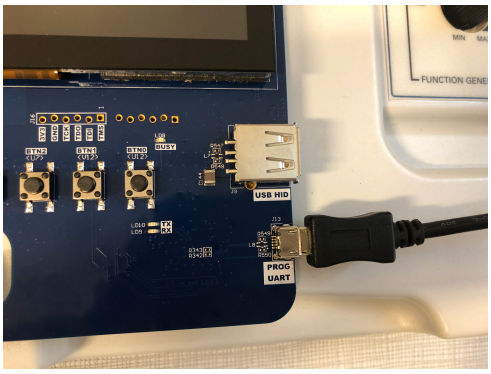
1. insert the DSDB into the NI Elvis Board, as represented in figure 1.1a;
2. connect the NI Elvis Board to the power supply;
3. connect the UART programming USB port to the computer (figure 1.1b);
4. turn on the boards, and check that LED *POWER*, *ACTIVE* and *PGOOD* are on, as shown on figure 1.2a;
5. select a memory, here *JTAG* (figure 1.2b).

Concerning the memory, the following options are provided :

- SD: the code is written on an external SD card;
- QSPI: the code is written on the memory of the board;
- JTAG: the code is written on the volatile memory of the board.

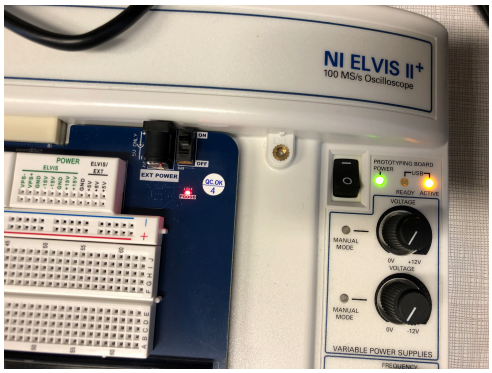


(a) NI Elvis board (white) and Digital Systems Development Board (blue)

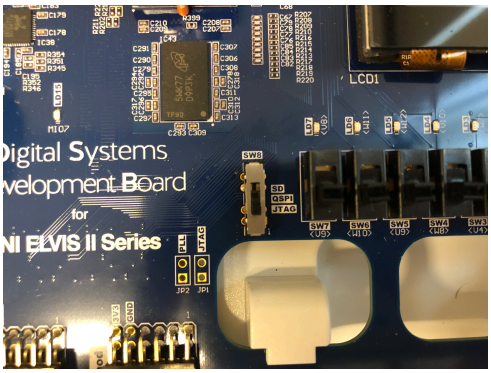


(b) Connection of the DSDB to the computer

Figure 1.1



(a) Powering og the board



(b) Selection of the memory

Figure 1.2

1.3 Create a new project on LABVIEW

In order to create a new project on *LABVIEW*, the following steps need to be accomplished:

1. open *LABVIEW*;
2. click on "Create a new project";
   A new window will open
3. click on "empty project" then "finish";
   A window "project explorer" will open
4. right click on "work station", then "new", then "Targets and peripherals";
   A window "Add targets and peripherals" will open

5. tick “new target or new peripheral”;
6. open “Digilent” and select “Digital Systems Development Board” and click on “OK”;
   All ports of the board are now displayed
7. right click on “FPGA Target: DSDB,” then “New” and finally “VI”.

After following these steps, three windows should be opened: one displaying all ports of the board, called “project explorer”, one “diagram window”, where coding blocs will be positioned (figure 1.3), and one “front side window”, which is the interface with the user, as shown in figure 1.4 .

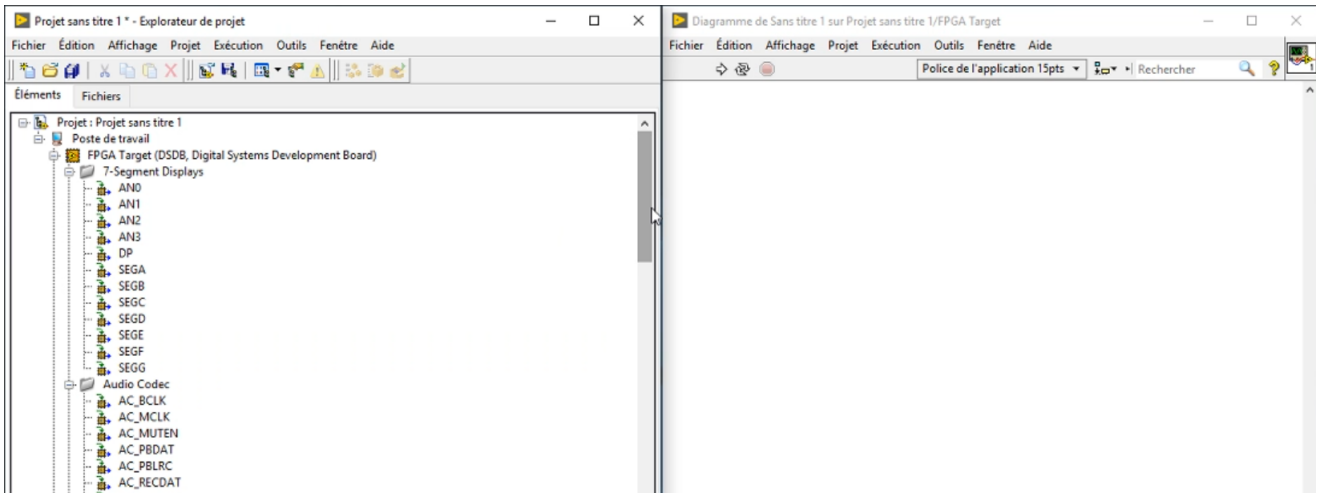


Figure 1.3: Project explorer and diagram windows

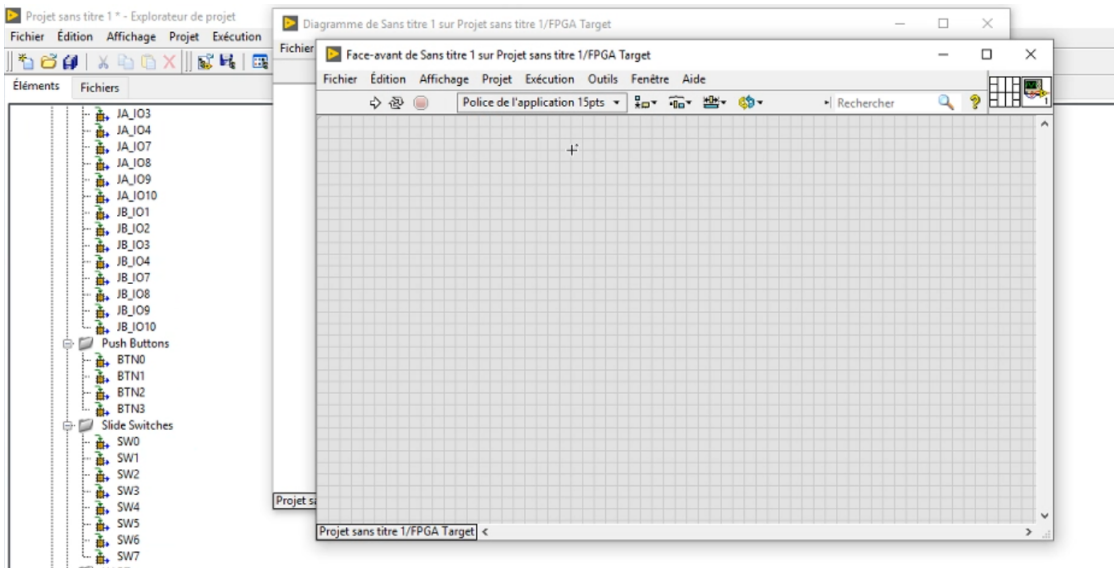


Figure 1.4: Front side window

Chapter 2

Tests with the DSDB

In the chapter, the way of controlling LED (from the board or external) with buttons or with a Pulse-Width Modulation will first be expound, followed by the way of controlling a servo-motor using the same elements (buttons or PWM). These are also the steps followed during this project to better understand the functioning of the board.

2.1 LED activation

Before working with the servo-motor, it is interesting to see how to turn LED's on and off with the board.

2.1.1 LED from the board

Little LED's and buttons can be found on the Digital Systems and Development Board. We will first learn how to control them. To do so, here are the steps to follow:

1. select form the "project explorer" window the name corresponding to the LED you want to turn on and off;
2. drag and drop it in the diagram window (figure 2.1);
3. add a while loop (search for "while loop" in the search bar) and create a square as shown on figure 2.2;
4. add a "false constant" as a stop condition for the loop. In this case, it means that the loop will never end. If you want to stop the loop by pressing a button for instance, connect the corresponding bloc to the red button of the loop;
5. select a button from the project explorer window and drag and drop it in the while loop;
6. connect both LED and button blocs;

Import the code and the board: the steps to import the code on the board are always the same, and are the following:

1. in the diagram window, click on the white arrow (execute) on the top of the window;
2. save your project;
3. in the "Select Compile Server" window, tick "Use the local compile server" and "OK";

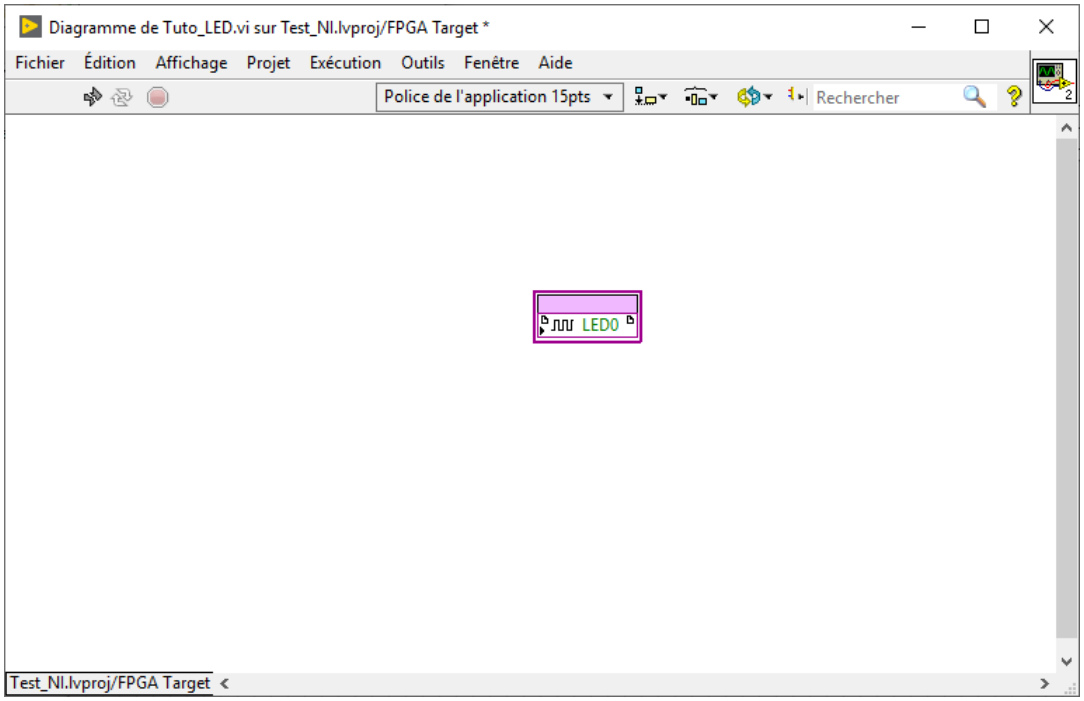


Figure 2.1: LED block in LABVIEW

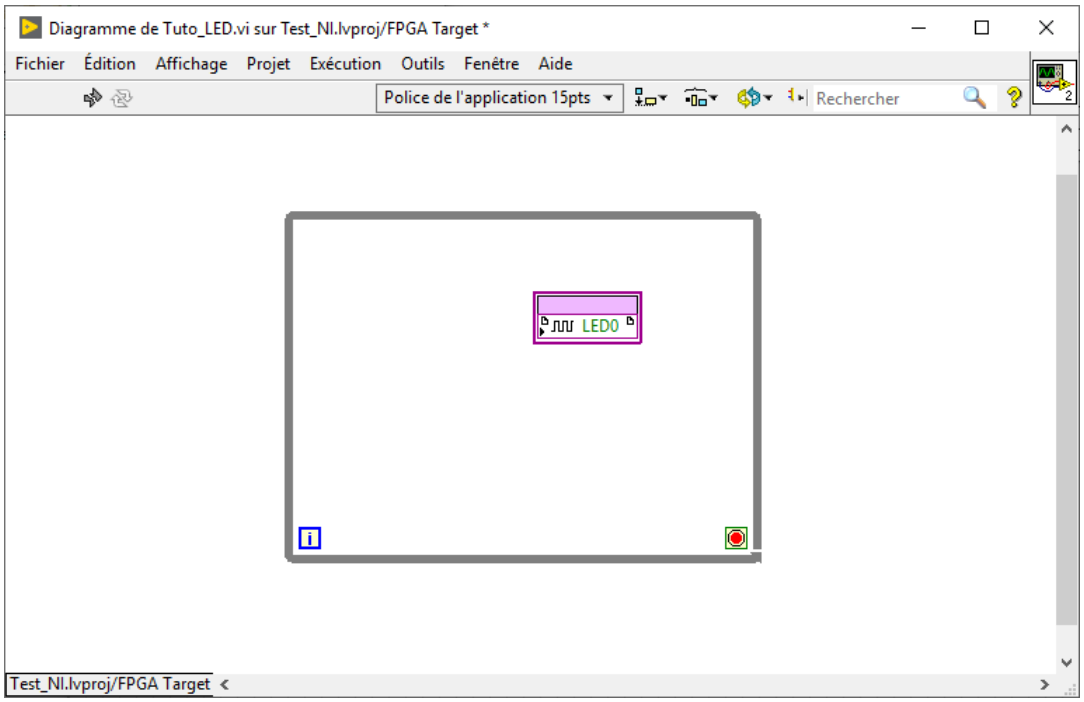


Figure 2.2: While loop in LABVIEW

4. compilation should be launched. This can take a few minutes.

After compiling, pressing the selected button should turn on the selected LED.

2.1.2 External LED

Now, instead of controlling an on-board LED, we want to control an external one. To do that, we first need to connect it properly, by using one resistor, a LED and two wires. The

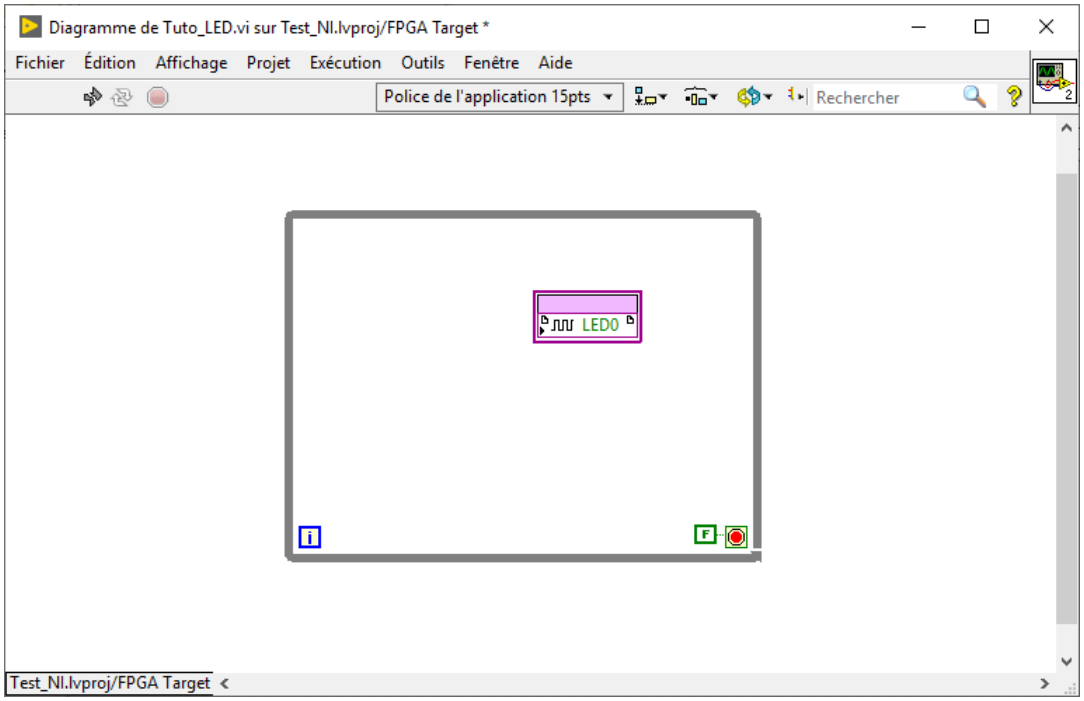


Figure 2.3: While loop condition

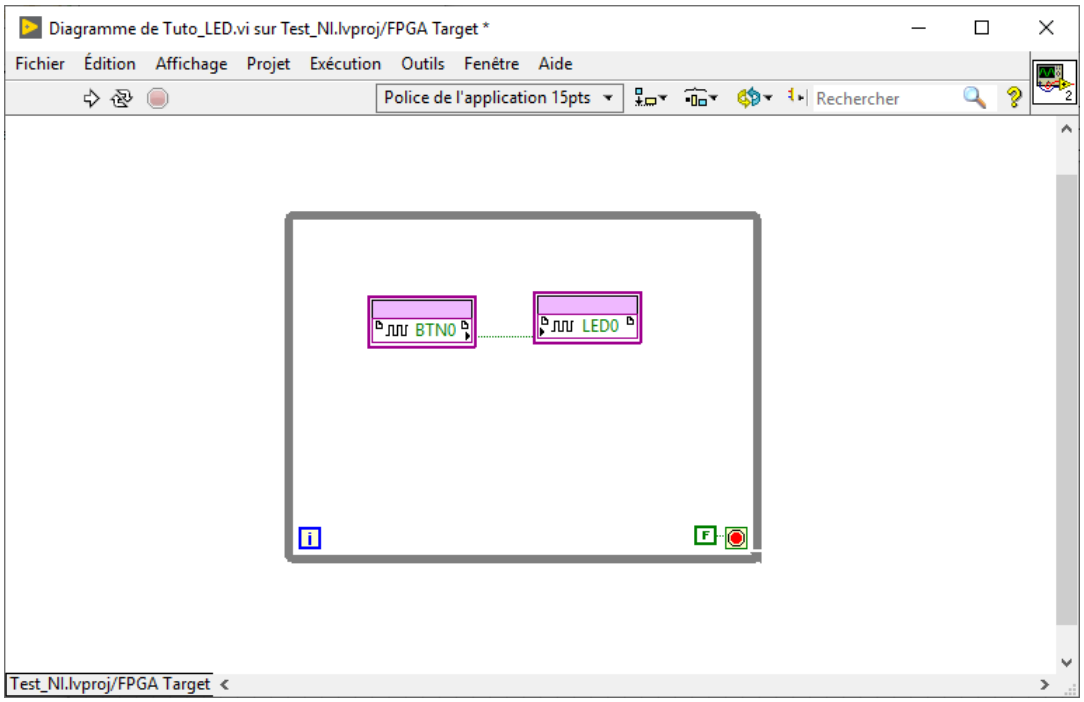


Figure 2.4: Complete diagram

schematic is represented in figure 2.5 The LED is here connected the a digital port of the board (D0).

First, we want to control the LED by pressing a button of the board, button 0 for instance. Steps are exactly the same as for the on-board LED, except that the LED bloc is replaced by the bloc corresponding to the digital port D0, named “BB3_DiO0”. The final schematic is represented in figure 2.6.

In a second time, we can make the LED blink. In order to do that, a square wave

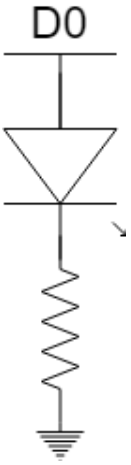


Figure 2.5: LED blinking schematic

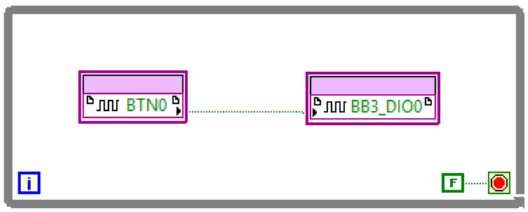


Figure 2.6: External LED control using a button

signal has to be added, as displayed in figure 2.7a, which will create the blinking of the LED. The steps to find the square wave are the following:

1. right click in the diagram window in order to display functions;
2. select “FPGA Math Analysis”;
3. select “generation”;
4. select “square wave” and slide it into the diagram window.

By double-clicking on the square wave bloc, its properties can be changed. Here, we will chose a frequency of 50 Hz and a Boolean data type. Also, a bloc to fix the value of the duty-cycle of the square wave should be added, as shown on figure 2.7b. To do so, follow these steps:

1. right click on the last port of square wave bloc;
2. select “create”;
3. select “command”, a bloc should appear.

Import the code on the board, and the LED should blink.

(a) Square wave with LED

(b) Duty-cycle control

Figure 2.7

2.2 Servo-motor control

After getting to know the DSDB by controlling LED's, the control of a servo-motor can be considered. First, it will be monitored by a Pulse-Width Modulation, then by pressing buttons on the board.

2.2.1 Servo-motor with PWM

Connecting the servo-motor

For the connection of the servo-motor, 3 wires will be needed. One connected to 5 V, one to the ground and one the a digital port of the board, here D0 (figure 2.8).

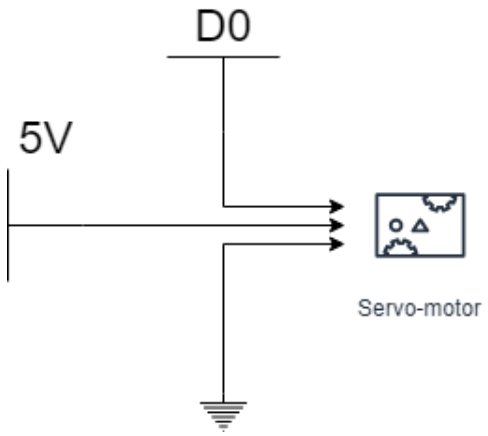


Figure 2.8: Servo connection schematic

Coding the servo-motor with PWM

The first steps to control a servo-motor are exactly the same than the one used for the LED: if we consider that the digital port of the servo-motor is the same than for the LED (D0), slide the corresponding bloc, add a while loop and its stop condition, and connect a square wave to the digital port.

For the servo-motor, its angle will depend on the duty cycle. If the user wants a particular angle, the latter has the be transformed into a duty cycle. To do so, we know that, for the servo-motor we used, 0° corresponds to 2 % and 180° to 11 %. By linear interpolation, we find the following line equation:

$$D = 0.000555 * \text{ angle} + 0.02$$

The line equation has to be implemented in LABVIEW. Add, multiply and constant blocs can be found when right clicking in the while loop and selecting "Numeric". The schematic bloc can be found in figure 2.9.

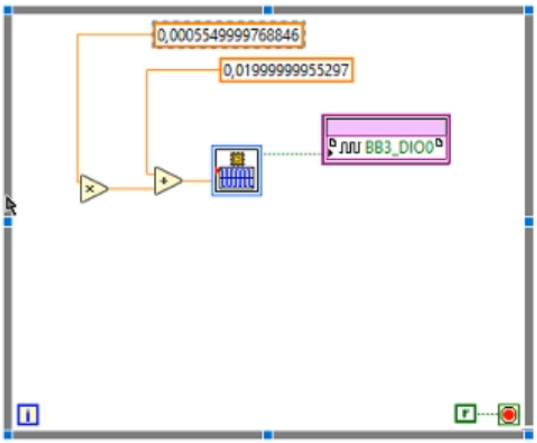


Figure 2.9: Line equation in LABVIEW

Finally, an angle control bloc has to be added, as shown in figure 2.10. In order to do so, follow these steps:

1. right click on the last port of the multiply bloc;
2. select "create";
3. select "command", a bloc should appear.

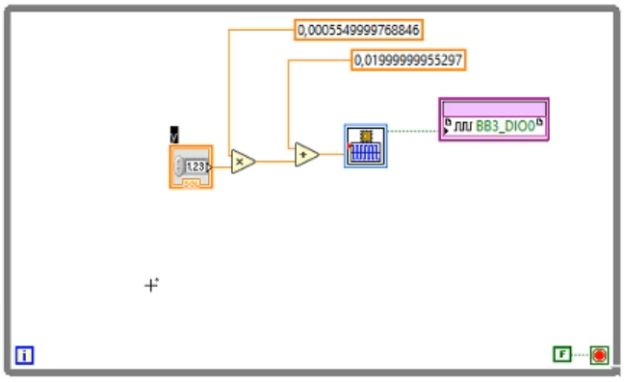


Figure 2.10: Angle control for the servo-motor

Finally, an interface with the user can be added, a rotating button for instance (figure 2.12). To do so:

1. double click on the new bloc: this should open the front-side window;

2. select the element with “0” and right click ;

3. select “replace”, “numeric” and finally the kind of interface wanted. Here, a rotating button is chosen (figure 2.11);

4. to adapt the button to the application, right click on the button: the properties window opens;

5. in the “in scale” part, select 180 for the maximum and click on OK.

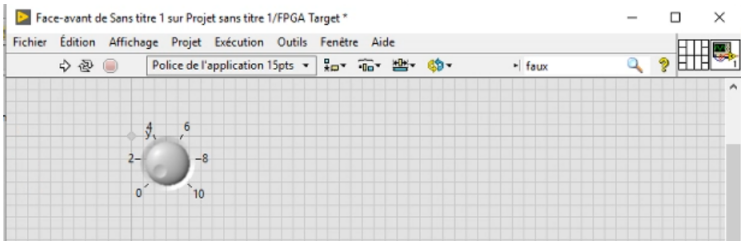


Figure 2.11: Basic button

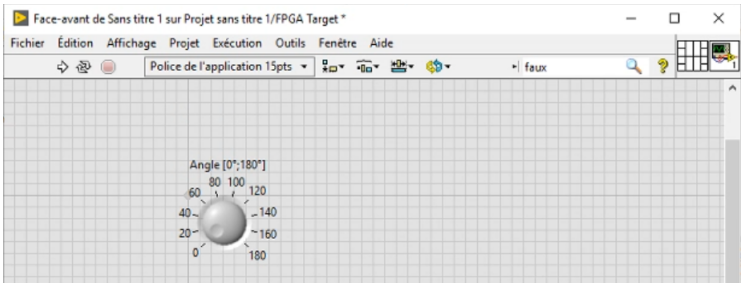


Figure 2.12: Interface with the user

2.2.2 Servo-motor control with buttons

The final step of the project is to control the servo-motor by pressing buttons on the board. The goal here is that, when pressing one button, the servo angle will increase, when pressing an other one, the servo angle will decrease, and when a third button is pressed, the servo angle resets to 90°. The full schematic bloc can be found in figure 2.16.

1: those blocs are the same as before, making the angle to duty cycle conversion and PWM.

2: limitation of the mechanical servo position. Indeed, the counter can’t go higher than 180° and lower than 0° in order to not brake the servo-motor.

3: command of the servo by pressing buttons. If button 1 is pressed, the angle will increase and the LED 1 will turn on. If the button 2 is pressed, the angle will decrease and LED 2 will turn on. Finally, if button 0 is pressed, the angle will reset to 90°.

The value of the angle is displayed on the computer screen as well.

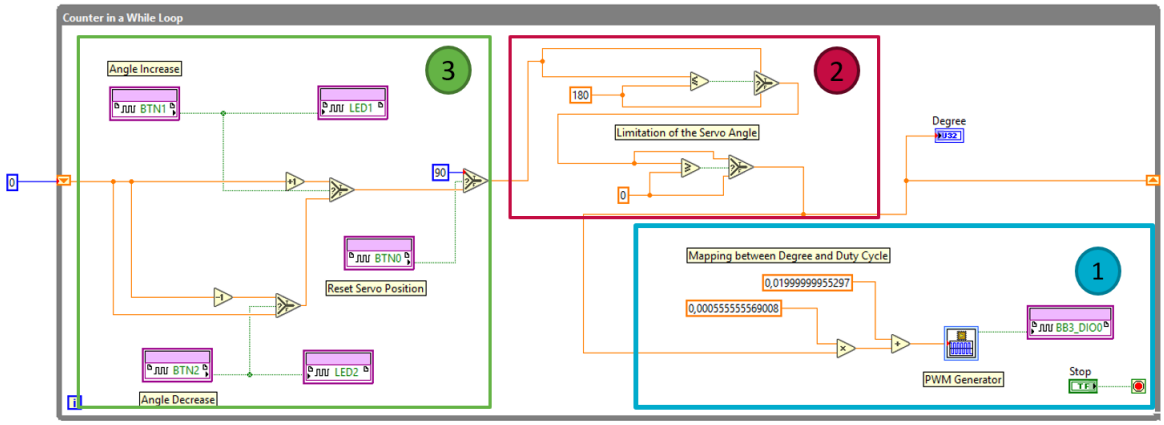


Figure 2.13: Control of a servo-motor with buttons

2.3 Test bench

In order to verify the correctness of our servo control, we need to take a look at the output signal.

The control of the servo is achieved by sending a PWM (pulse-width modulation) signal at 50 Hz. The parameters for the pulses are the minimal pulse width, the maximal pulse width, and the repetition rate. Given the rotation constraints of the servo, neutral is defined to be the center of rotation. Different servos will have different constraints on their rotation, but the neutral position is always around 1.5 milliseconds (ms) pulse width.

In our case, for an angle control between 0° and 180°, the duty cycle must be equal to respectively 2% and 11%.

For an angle of 0°:

$$D = 0,4/20 = 2\%$$

For an angle of 90°:

$$D = 1,4/20 = 7\%$$

For an angle of 180°:

$$D = 2,2/20 = 11\%$$

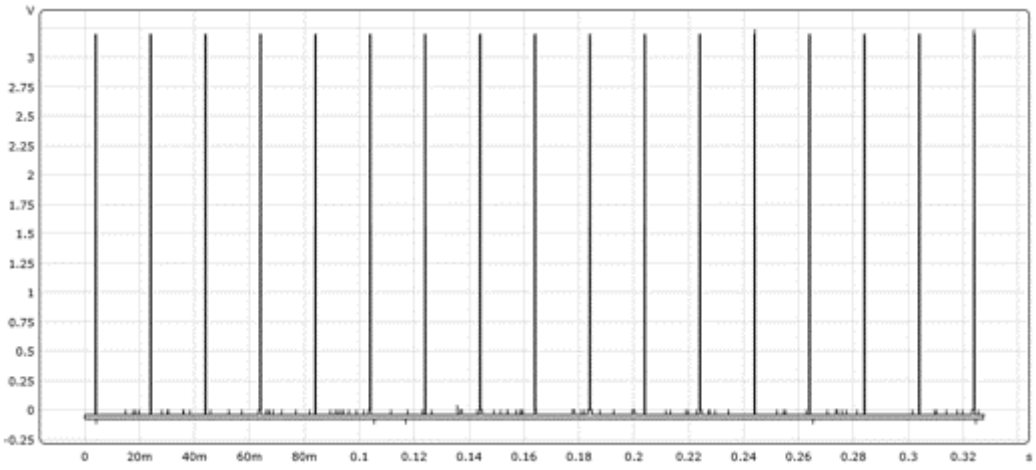


Figure 2.14: PWM signal with a duty cycle of 2%

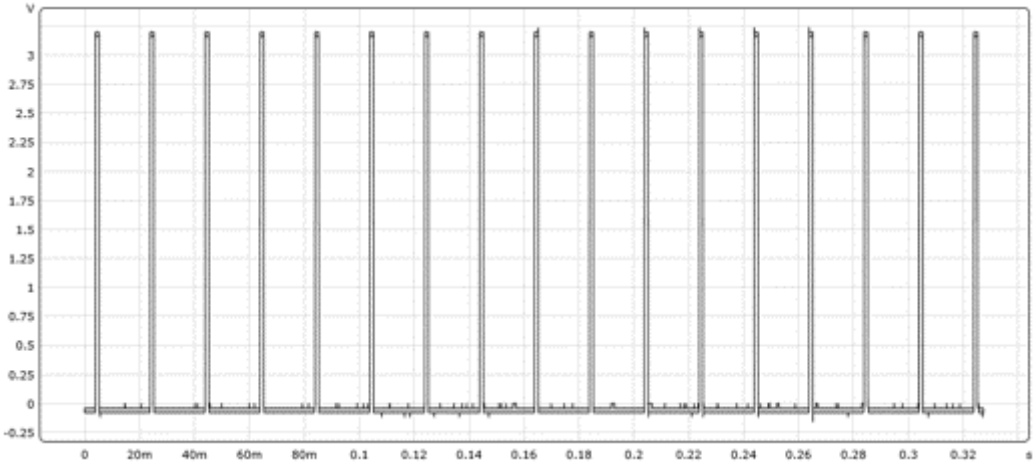


Figure 2.15: PWM signal with a duty cycle of 7%

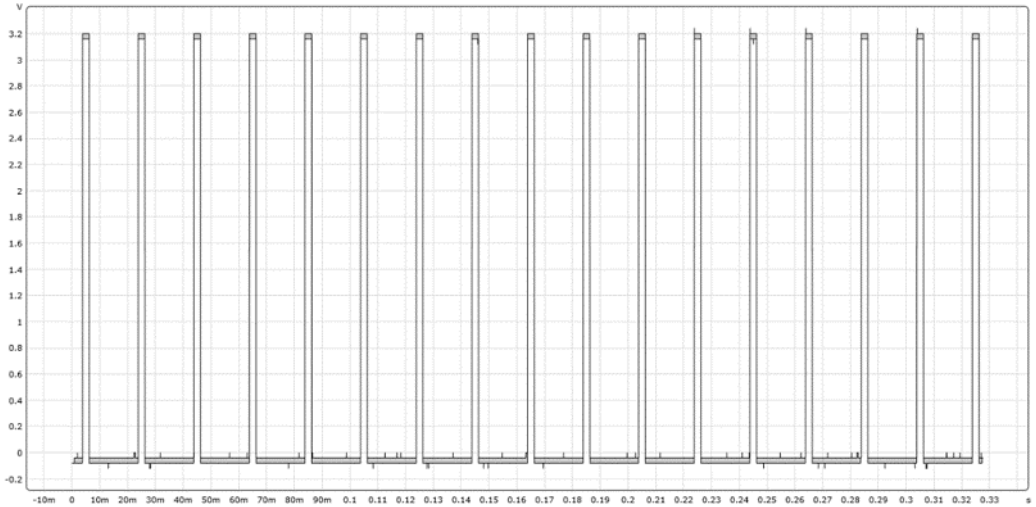


Figure 2.16: PWM signal with a duty cycle of 11%

Chapter 3

Functioning of the board

How does the software work ? LabView generates a report after the compilation. In this report, all the steps of the compilation can be found. The compiler is Vivado by Xilinx. The programming of the board is made using blocks which are interpreted in a VHDL code. The VHDL pseudo-code of each block is accessible only during the compilation, after this step all files are deleted. A example of a report is given in figure 3.1.

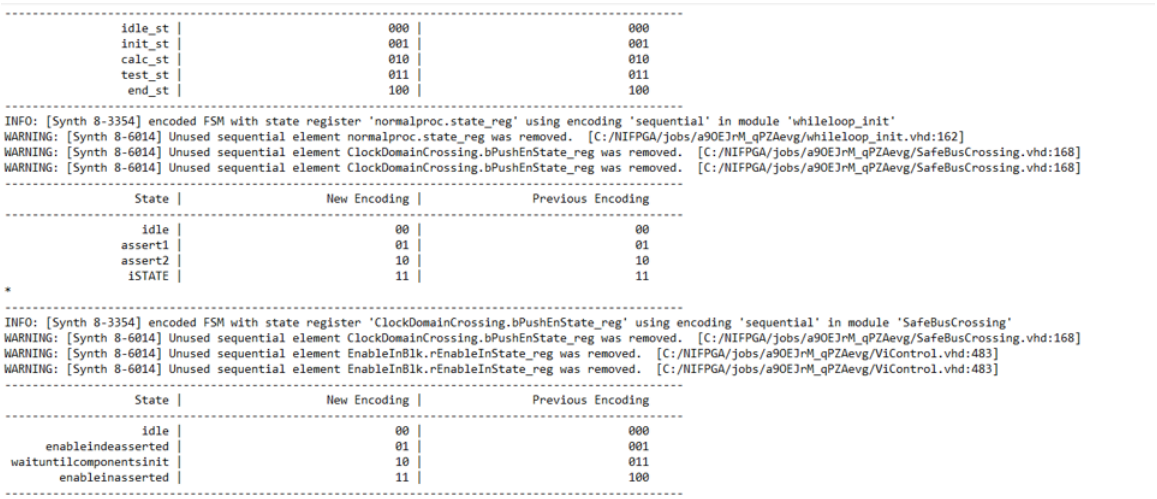
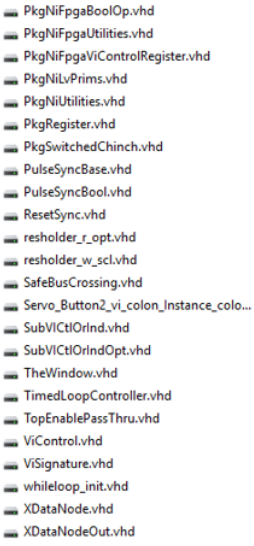


Figure 3.1: Compilation report

We can find for instance synthesising modules section, truth tables , cell usage, optimisation section, ...

The VHDL codes can be found in this directory : `C:\NIFPGA\jobs`



(a) Folder with VHDL codes

```vhdl
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

library UNISIM;
    use UNISIM.all;
    use UNISIM.vcomponents.all; -- for BSCAN

library work;
    use work.PkgNiUtilities.all;
    use work.PkgCommunicationInterface.all;
    use work.PkgCommIntConfiguration.all;

entity DSDB is
    port (
        --------------------------------
        --Clocks
        --------------------------------
        Sysclk125mhz          : in std_logic; -- Oscillator

        --------------------------------
```

(b) Example of VHDL code

Figure 3.2

Conclusion

This project consisting of the discovery of the NI Elvis Digital Systems Development Board by National Instrument, allowed us to handle a new hardware, the board, but also a new software, which is LABVIEW. At the end, we were able to control a servo-motor in different ways, as well as LED's.