EGR 680
Fall 2018
Final Project

PYNQ Embedded Design using Jupyter Notebooks

Due Date: Last day of class

Objectives

- Create custom Python Drivers to call in the Jupyter Notebooks
- Use the PMOD interface to communicate with Grove devices
- Use C functions in Python notebook to develop a Music Synthesizer using Grove Led bar and Grove Buzzer

---

Part I – Creating a Custom RGB LED Python Driver

For the first part of the project, you will be creating a custom Python driver using the Spyder IDE. This driver will include a class from which you can call methods from the Jupyter Notebook. All of the methods you were using in the last lab were called from the drivers running behind the notebook. These drivers are the communication link between the Processing System (PS) and the Programmable Logic (PL) for a lot of the peripheral devices on PYNQ. Although, not all drivers communicate this way as you will see later in the project. In this part, you will be adding functionality to the last lab by creating your own RGB LED driver to use in the Notebook. You will use the new RGB LED driver to then create an RGB LED color mixer.

1. Power on the board and connect it to you PC. Once the board has booted successfully, go to your PCs file explorer and navigate to the shared **PYNQ** folder. Reference the last lab if needed.

2. Next, navigate to the **PYNQ** driver folder by clicking **PYNQ → lib**. Here you will find the installed drivers.

3. Copy the "***rgbled.py***" file by selecting and right-clicking or **CTRL+C**.

4. Now, navigate back two folders where the **PYNQ** folder resides and paste the "***rgbled.py***" driver here. Rename this driver "***myrgbled.py***". Alternatively, try pasting and renaming this file directly into the **PYNQ** driver folder if your shared permissions allow. If your permissions allow you to paste directly to the driver file, you may skip the next two steps and work directly from this folder.

5. Open a Jupyter Notebook session and click **New → Terminal** to open a **PYNQ** Linux terminal within the Notebook as shown in Figure 1.

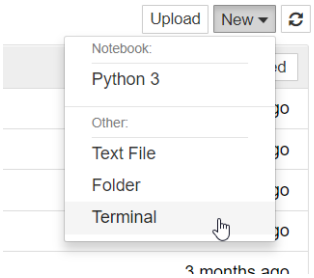

Figure 1: Opening a terminal in the Jupyter Notebook

6. To paste the new driver into the **PYNQ** driver library folder, type the command as shown in Figure 2 into the terminal. This command places a copy of ***"myrgbled.py"*** located in the top level folder into the driver library folder. Moving forward, you will need to perform this copy operation every time you make an edit to your driver and are ready to run it in the Notebook. You will do all of your editing outside of the **PYNQ** driver library and paste/overwrite to update the file in the driver library.

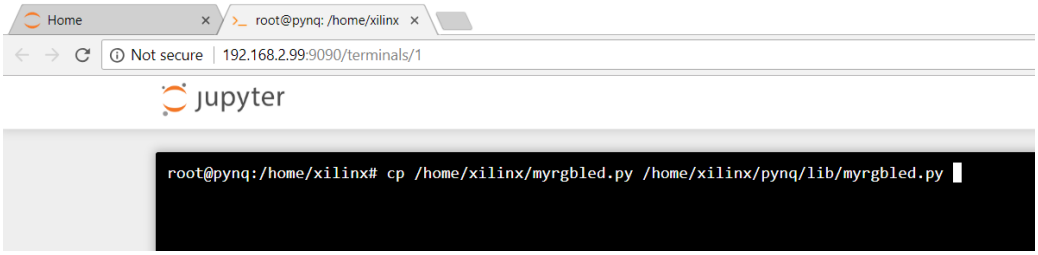

Figure 2: Copying a file in PYNQ

You will need to create the RGB LED color mixer using the **ipywidgets** integer or float slider. In theory, you should be able to create an infinite amount of colors with the combination of red, blue, and green LEDs. However, with digital electronics, we are limited to the width of the driving data bus or processing system to create the colors.

Figure 3 shows the Vivado **IP** for the RGB LEDs. This **IP** is instantiated in the top-level Verilog file of the base overlay design. When you load the ***"base.bit"*** bitsteam in the Notebook, this IP is included in the overlay. As you can see, the output of the **IP** is 6-bits wide. The 3 LSBs of the output are connected to the **LD-4** R-G-B I/O pins, and the 3 MSBs are connected to the **LD-5** R-G-B I/O pins.

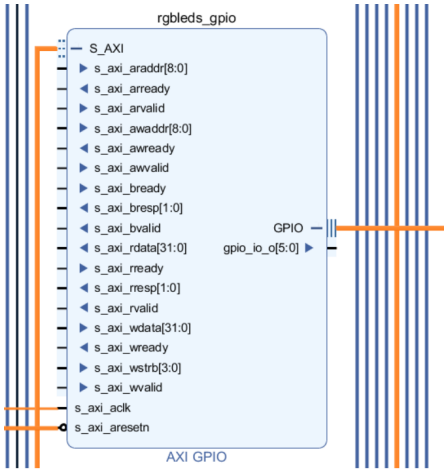

Figure 3: RGB LED Vivado IP in the base overlay design

7. Now that you have a new RGB LED driver, you need to edit the ***"__init__.py"*** driver to import your new driver. Then you need to initialize the driver within the ***"base.py"*** driver. You will find "base.py" driver file under ***pynq → overlays → base*** directory structure. You can use any editor you'd like to edit your driver, but Spyder is recommended since it handles the spacing and tabbing needed for Python very well.

8. Perform steps 2-6 again to make the appropriate edits to the ***"__init__.py"*** and the ***"base.py"*** drivers. This time, **DO NOT RENAME THE DRIVERS**. You need to import your new driver in the ***"__init__.py"*** driver and edit the line in Figure 4 of the ***"base.py"*** driver to use your new driver instead of the old one.

```
99          self.rgbleds = ([None] * 4) + [pynq.lib.MYRGBLED(i)
100                                         for i in range(4, 6)]
```

Figure 4: Editing the *base.py* driver

9. Now open "***myrgbled.py***" file and make appropriate changes (replace ***RGBLED*** with ***MYRGBLED*** intelligently). Save the file. Now run your PYTHON application file from your previous lab and make sure it still works.

- **Demonstrate successful working to your instructor.**

10. You need to modify your "***myrgbled.py***" as follows:

- Create individual methods for enabling/disabling RED, GREEN and BLUE color of the RGB LED. You must use these methods for controlling the RGB LED in the Notebook.
- PWM functionality for color mixing of the RGB LED.

11. You will need to create a new PYTHON application file that will do the following:

- Create 3 color mixing sliders using ipywidgets. One slider per color (R-G-B).
- Create toggle buttons for all four of the green LEDs using ipywidgets.
- Create a way to set a flashing rate of the green LEDs using ipywidgets.
- Create a neat and organized GUI for all of the LED functionality.

- **Demonstrate successful working to your instructor.**

Part II – PMOD Interface and Grove LEDs

In this part, you will be using the PMOD interface to communicate with a Grove LED Bar. You cannot connect the LED bar directly to **PYNQ**, you need to use the Grove Adapter as shown in Figure 5 below. The LED bar has an on-board MY9221 IC to drive the individual LED bars. The communication protocol is a two-wire serial interface including data in, data out, and a data clock. Writing the driver for this IC is beyond the scope of this course, so you will use the drivers already installed on the **PYNQ** board. In the last part, the Python drivers communicated directly with the PL. The PMOD Vivado **IP** in the base overlay uses a **Microblaze** softcore processor instead, similar to the **ZYNQ** softcore processor that you utilized earlier in the course. The driver for the MY9221 is written in **C** and is running on the **Microblaze** processor. The communication between the **C** drivers on **Microblaze** and the Python drivers is handled with a mailbox system that Xilinx has developed. The Python driver reads and writes to this mailbox to control the LED bar. Your objective in this part is to design a GUI that uses an integer slider and drop-down menu to control the LED bars.

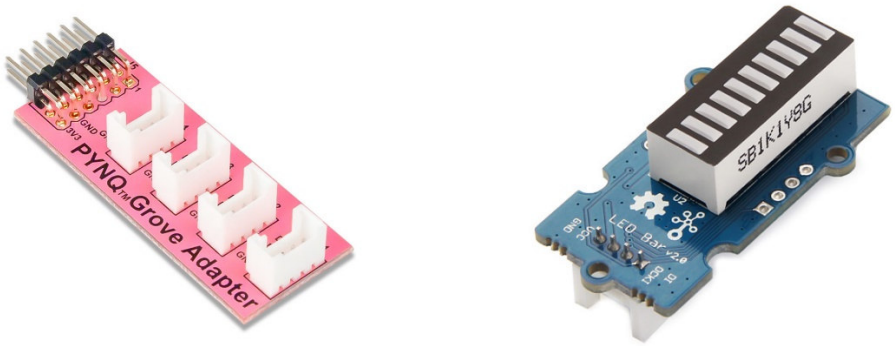

Figure 5: PYNQ PMOD Grove adapter and Grove LED bar

1. Plug in the **PYNQ** Grove Adapter board into ***PMOD A*** or ***PMOD B*** with the connectors facing up.

2. Then, plug the interface cable into the LED bar and the other end of the cable into ***G1*** on the adapter.

3. Power up the board and wait for it to boot.

4. Open a Jupyter Notebook, create a folder called "***FinalProject***" and then create a new Python3 file and rename that as "***LEDBar_Grove***" to create ***"LEDBar_Grove.ipynb"***.

5. Now type the following code as shown in Figure 6.

Grove LED Bar example

This examples shows how to use the Grove LED bar on the PYNQ-Z1 board. A Grove LED Bar and Pynq Grove Adapter are required. The LED bar is an example of a peripheral driven by GPIO. The LED bar is attached to the G1 connection on the Pynq Grove Adapter, which is connected to PMODA on the Pynq-Z1 board. The LED bar has 10 LEDs. 8 Green LEDs, 1 Orange LED, and 1 Red LED. The brightness for each LED can be set independently. All functions supporting the LED bar are implemented in the C code running on the IOP driving the LED bar.

In [2]:
```
# Setup the PYNQ board
from pynq.overlays.base import BaseOverlay
base = BaseOverlay("base.bit")

from pynq.lib.pmod import Grove_LEDbar
from pynq.lib.pmod import PMOD_GROVE_G1 # Import constants

# For delays
from time import sleep

# Instantiate Grove LED Bar on PMODA and on Pmod2Grove G1
ledbar = Grove_LEDbar(base.PMODA,PMOD_GROVE_G1)
ledbar.reset()

# Flash 2 exterme LEDs of the LED Bar in a loop
for i in range(5):
    ledbar.write_binary(0b1000000001)
    sleep(0.5)
    ledbar.write_binary(0b0000000000)
    sleep(0.5)
```

Figure 6: Example Grove LED Bar code

6. Make sure that 2 extreme LEDs flash 5 times.

7. Now modify the code such that it does the following:

    a. Create an integer slider SL1 that goes from 0 – 10 (representing 10 LEDs on the LED Bar).
    b. Create another integer slider SL2 that goes from 0 – 3 (representing following levels of intensity:
        i. 0 → OFF, 1 → LOW, 2 → MEDIUM, 3 → HIGH
    c. When you move the slider SL1, the number that slider returns should turn ON that many LEDs on the LED bar.
        i. For example, if slider returns 1 then then rightmost LED should turn ON.
        ii. For example, if slider returns 2 then the rightmost 2 LEDs should turn ON and so on.
    d. When you move the slider SL2, then number that slider returns should set the appropriate intensity for the LEDS that are turned ON.

NOTE: Grove LEDBar driver is available under *pynq → lib → pmod → pmod_grove_ledbar.py*

- **Demonstrate successful working to your instructor.**

Part III – Music Synthesizer

In this part, you will learn how to invoke C functions inside a Python notebook. An example is available for you to look at on the PYNQ image. Browse to ***base → microblaze*** and open "***microblaze_c_libraries.ipynb***" file. Understand the example pertaining to GPIO as both Grove LedBar and Grove Buzzer are gpio interfaces.

1. In Jupyter Notebook, under folder called "***FinalProject***", create a new Python3 file and rename that as "***MusicSynthesizer***" to create ***"MusicSynthesizer.ipynb"***.

2. Create a ***Markdown*** or ***Heading*** section and give a brief description of your project.

NOTE: Grove LedBar module will be connected to PMOD A connector G1 and Grove Buzzer module will be connected to PMOD A connector G4.

3. Create a new cell and add the lines of code that imports all the necessary BIT files and PYTHON libraries.

4. Create another cell and incorporate the necessary C functions for designing a Music Synthesizer.

NOTE: Refer to the link *https://github.com/Xilinx/PYNQ/tree/master/pynq/lib/pmod*
***and refer to pmod_grove_buzzer.c* and *pmod_grove_ledbar.c* files for the C functions that you can use.**

5. Make sure to have one main function that calls all the necessary functions to mimic a music synthesizer. You have to use your own musical melody that has 10 distinct notes and turn ON the corresponding led on the ledbar when that note is played.

6. Create another cell and call the function mentioned in step 5.

- **Demonstrate successful working to your instructor.**

Project Deliverables

You are required to turn in a hard copy of the report. Report should have the following items:

- Cover page with one page description of each part
- Printout of your RGBLed driver and the associated application file with comments
- Printout of your Grove LedBar application file with comments
- PRINTOUT OF YOUR music Synthesizer application file with comments

You also have to demonstrate your design and turn in a compressed version of the project