



EGR680 High Level Implementation on FPGA

Final Project

PYNQ Embedded Design using Jupyter Notebooks

Professor: Dr. C. Parikh

Student: Dimitri Häring

December 02, 2018

Contents

1	Introduction	3
2	Design	3
2.1	Part I - RGB LED Driver	3
2.2	Part II - LED Groove Bar	4
3	Conclusion	4
4	Appendix	5
4.1	Python code Listings Part I - RGB LED Driver	5
4.2	Python code Listings Part II - LED Groove Bar	12
4.3	Python code Listings Part III - Music Synthesizer	13

List of Figures

1	Jupyter Notebook terminal, copy a file.	3
2	Groove LED Bar program output.	4

Listings

1	Jupyter Notebook terminal, copy a file.	3
2	Python code changed on line 45 of file <code>__init__.py</code>	4
3	Python code changed on line 99 of file <code>base.py</code>	4
4	Jupyter Notebook file <code>__init__</code> saved as <code>*.py</code> file.	5
5	Jupyter Notebook file <code>base</code> saved as <code>*.py</code> file.	6
6	Jupyter Notebook file <code>myrgbled</code> saved as <code>*.py</code> file.	8
7	Part II - LED Groove Bar Python code.	12

1 Introduction

The goal of laboratory ten is to familiarize the student with the Jupyter Notebook and debugging of hardware in Vivado.

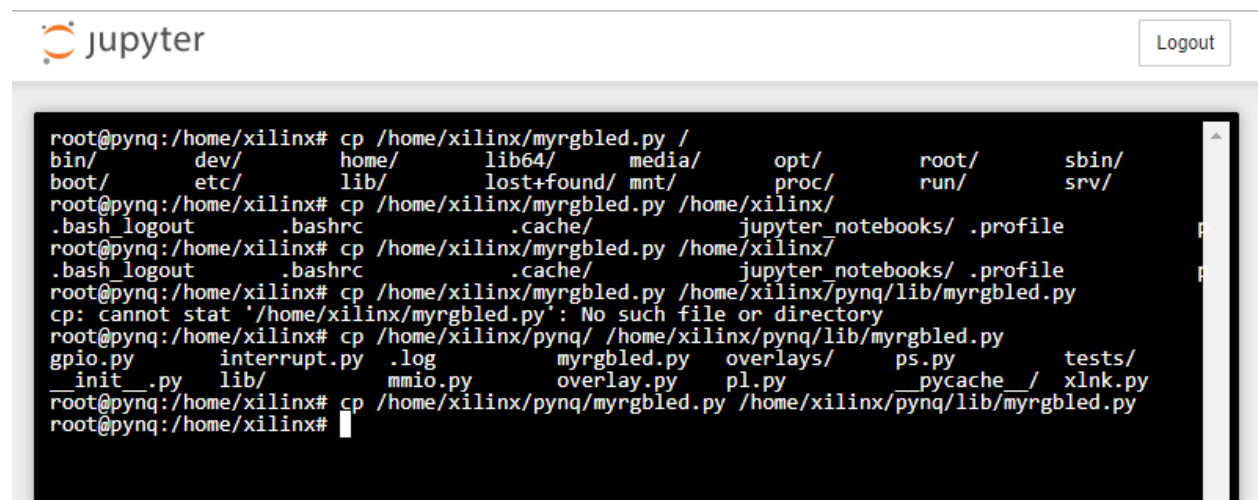
2 Design

In this section the design and decisions that were made to achieve the laboratory are discussed.

2.1 Part I - RGB LED Driver

From project description, 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.

- Create individual methods for enabling/disabling RED, GREEN and BLUE color of the RGB LED.
- PWM functionality for color mixing of the RGB LED.
- 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.

A screenshot of a Jupyter Notebook terminal window. The window has a title bar with the Jupyter logo and a 'Logout' button. The terminal shows a series of commands and their outputs. The first command is 'cp /home/xilinx/myrgbled.py /', which lists the contents of the root directory. The second command is 'cp /home/xilinx/myrgbled.py /home/xilinx/', which fails with a 'No such file or directory' error. The third command is 'cp /home/xilinx/myrgbled.py /home/xilinx/pynq/lib/myrgbled.py', which succeeds. The terminal text is as follows:

```
root@pynq:/home/xilinx# cp /home/xilinx/myrgbled.py /
bin/      dev/      home/     lib64/    media/    opt/      root/     sbin/
boot/     etc/     lib/      lost+found/ mnt/      proc/     run/      srv/
root@pynq:/home/xilinx# cp /home/xilinx/myrgbled.py /home/xilinx/
.bash_logout .bashrc  .cache/  jupyter_notebooks/ .profile
root@pynq:/home/xilinx# cp /home/xilinx/myrgbled.py /home/xilinx/
.bash_logout .bashrc  .cache/  jupyter_notebooks/ .profile
root@pynq:/home/xilinx# cp /home/xilinx/myrgbled.py /home/xilinx/pynq/lib/myrgbled.py
cp: cannot stat '/home/xilinx/myrgbled.py': No such file or directory
root@pynq:/home/xilinx# cp /home/xilinx/pynq/ /home/xilinx/pynq/lib/myrgbled.py
gpio.py      interrupt.py .log       myrgbled.py overlays/   ps.py      tests/
__init__.py  lib/        mmio.py    overlay.py pl.py       __pycache__ xlnk.py
root@pynq:/home/xilinx# cp /home/xilinx/pynq/myrgbled.py /home/xilinx/pynq/lib/myrgbled.py
root@pynq:/home/xilinx#
```

Figure 1: Jupyter Notebook terminal, copy a file.

Listing 1: Jupyter Notebook terminal, copy a file.

```
root@pynq:/home/xilinx# cp /home/xilinx/pynq/myrgbled.py /home/xilinx/pynq/lib/myrgbled.py
```

```
1 from .myrgbled import MYRGBLED
```

Listing 2: Python code changed on line 45 of file `__init__.py`.

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

Listing 3: Python code changed on line 99 of file `base.py`.

2.2 Part II - LED Groove Bar

The Groove LED Bar can be turned on in level increments from 0 to 10 where 0 is off and 10 all segments on. The brightness of the leds can be defined independently with an value from 0 to 3 where 0 is off, 1 is low, 2 is medium, and 3 is led brightness high. As graphical user interface (GUI) two integer slider are used SL1 and SL2 as shown in Figure 2. The source code is shown in Listing ?? in Section 4.2 of the appendix.



Figure 2: Groove LED Bar program output.

3 Conclusion

4 Appendix

The appendix contains code listings and other large information parts that contain partial or complete relevance to the reports topic.

4.1 Python code Listings Part I - RGB LED Driver

```
1 # Copyright (c) 2016, Xilinx, Inc.
2 # All rights reserved.
3 #
4 # Redistribution and use in source and binary forms, with or without
5 # modification, are permitted provided that the following conditions are met:
6 #
7 # 1. Redistributions of source code must retain the above copyright notice,
8 #    this list of conditions and the following disclaimer.
9 #
10 # 2. Redistributions in binary form must reproduce the above copyright
11 #    notice, this list of conditions and the following disclaimer in the
12 #    documentation and/or other materials provided with the distribution.
13 #
14 # 3. Neither the name of the copyright holder nor the names of its
15 #    contributors may be used to endorse or promote products derived from
16 #    this software without specific prior written permission.
17 #
18 # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 # AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
20 # THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
21 # PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
22 # CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
23 # EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
24 # PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
25 # OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
26 # WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
27 # OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
28 # ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29
30
31 # from .audio import Audio
32 # from .video import HDMI
33 # from .video import Frame
34 # from .dma import DMA
35 # from .trace_buffer import Trace_Buffer
36 # from .usb_wifi import Usb_Wifi
37
38 from .pynqmicroblaze import PynqMicroblaze
39 from .pynqmicroblaze import MicroblazeRPC
40 from .pynqmicroblaze import MicroblazeLibrary
41 from .axigpio import AxiGPIO
42 from .dma import DMA
43 from .dma import LegacyDMA
44 from .led import LED
45 from .myrgbled import MYRGBLED
46 from .switch import Switch
47 from .button import Button
48
49 from .arduino import Arduino
50 from .arduino import Arduino_DevMode
51 from .arduino import Arduino_IO
52 from .arduino import Arduino_Analog
53 from .arduino import Arduino_LCD18
54
55 from .pmod import Pmod
56 from .pmod import Pmod_DevMode
57 from .pmod import Pmod_ADC
58 from .pmod import Pmod_DAC
59 from .pmod import Pmod_OLED
```

```

60 from .pmod import Pmod_LED8
61 from .pmod import Pmod_IO
62 from .pmod import Pmod_IIC
63 from .pmod import Pmod_DPOT
64 from .pmod import Pmod_TC1
65 from .pmod import Pmod_TMP2
66 from .pmod import Pmod_ALS
67 from .pmod import Pmod_Cable
68 from .pmod import Pmod_Timer
69 from .pmod import Pmod_PWM
70
71 from .logictools import LogicToolsController
72 from .logictools import Waveform
73 from .logictools import BooleanGenerator
74 from .logictools import PatternGenerator
75 from .logictools import TraceAnalyzer
76 from .logictools import FSMGenerator
77
78 from . import video
79 from . import audio
80 from . import dma
81
82 __author__ = "Graham Schelle"
83 __copyright__ = "Copyright 2016, Xilinx"
84 __email__ = "pynq_support@xilinx.com"

```

Listing 4: Jupyter Notebook file `__init__` saved as *.py file.

```

1 # Copyright (c) 2017, Xilinx, Inc.
2 # All rights reserved.
3 #
4 # Redistribution and use in source and binary forms, with or without
5 # modification, are permitted provided that the following conditions are met:
6 #
7 # 1. Redistributions of source code must retain the above copyright notice,
8 #    this list of conditions and the following disclaimer.
9 #
10 # 2. Redistributions in binary form must reproduce the above copyright
11 #    notice, this list of conditions and the following disclaimer in the
12 #    documentation and/or other materials provided with the distribution.
13 #
14 # 3. Neither the name of the copyright holder nor the names of its
15 #    contributors may be used to endorse or promote products derived from
16 #    this software without specific prior written permission.
17 #
18 # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 # AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
20 # THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
21 # PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
22 # CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
23 # EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
24 # PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
25 # OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
26 # WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
27 # OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
28 # ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29
30
31 import pynq
32 import pynq.lib
33 import pynq.lib.video
34 import pynq.lib.audio
35 from .constants import *
36 from pynq.lib.logictools import TraceAnalyzer
37
38
39 __author__ = "Peter Ogden"
40 __copyright__ = "Copyright 2017, Xilinx"

```

```

41 __email__ = "pynq_support@xilinx.com"
42
43
44 class BaseOverlay(pynq.Overlay):
45     """ The Base overlay for the Pynq-Z1
46
47     This overlay is designed to interact with all of the on board peripherals
48     and external interfaces of the Pynq-Z1 board. It exposes the following
49     attributes:
50
51     Attributes
52     ~~~~~
53     iop_pmoda : IOP
54         IO processor connected to the PMODA interface
55     iop_pmodb : IOP
56         IO processor connected to the PMODB interface
57     iop_arduino : IOP
58         IO processor connected to the Arduino/ChipKit interface
59     trace_pmoda : pynq.logictools.TraceAnalyzer
60         Trace analyzer block on PMODA interface, controlled by PS.
61     trace_arduino : pynq.logictools.TraceAnalyzer
62         Trace analyzer block on Arduino interface, controlled by PS.
63     leds : AxiGPIO
64         4-bit output GPIO for interacting with the green LEDs LD0-3
65     buttons : AxiGPIO
66         4-bit input GPIO for interacting with the buttons BTN0-3
67     switches : AxiGPIO
68         2-bit input GPIO for interacting with the switches SW0 and SW1
69     rgbleds : [pynq.board.RGBLED]
70         Wrapper for GPIO for LD4 and LD5 multicolour LEDs
71     video : pynq.lib.video.HDMIWrapper
72         HDMI input and output interfaces
73     audio : pynq.lib.audio.Audio
74         Headphone jack and on-board microphone
75
76     """
77
78     def __init__(self, bitfile, **kwargs):
79         super().__init__(bitfile, **kwargs)
80         if self.is_loaded():
81             self.iop_pmoda.mdtype = "Pmod"
82             self.iop_pmodb.mdtype = "Pmod"
83             self.iop_arduino.mdtype = "Arduino"
84
85             self.PMODA = self.iop_pmoda.mb_info
86             self.PMODB = self.iop_pmodb.mb_info
87             self.ARDUINO = self.iop_arduino.mb_info
88
89             self.audio = self.audio_direct_0
90             self.leds = self.leds_gpio.channel1
91             self.switches = self.switches_gpio.channel1
92             self.buttons = self.btns_gpio.channel1
93             self.leds.setlength(4)
94             self.switches.setlength(2)
95             self.buttons.setlength(4)
96             self.leds.setdirection("out")
97             self.switches.setdirection("in")
98             self.buttons.setdirection("in")
99             self.rgbleds = ([None] * 4) + [pynq.lib.MYRGBLED(i)
100                                     for i in range(4, 6)]
101
102             self.trace_pmoda = TraceAnalyzer(
103                 self.trace_analyzer_pmoda.description['ip'],
104                 PYNQZ1_PMODA_SPECIFICATION)
105             self.trace_arduino = TraceAnalyzer(
106                 self.trace_analyzer_arduino.description['ip'],

```

Listing 5: Jupyter Notebook file base saved as *.py file.

```

1 # Copyright (c) 2016, Xilinx, Inc.
2 # All rights reserved.
3 #
4 # Redistribution and use in source and binary forms, with or without
5 # modification, are permitted provided that the following conditions are met:
6 #
7 # 1. Redistributions of source code must retain the above copyright notice,
8 # this list of conditions and the following disclaimer.
9 #
10 # 2. Redistributions in binary form must reproduce the above copyright
11 # notice, this list of conditions and the following disclaimer in the
12 # documentation and/or other materials provided with the distribution.
13 #
14 # 3. Neither the name of the copyright holder nor the names of its
15 # contributors may be used to endorse or promote products derived from
16 # this software without specific prior written permission.
17 #
18 # THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
19 # AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
20 # THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
21 # PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR
22 # CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
23 # EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
24 # PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
25 # OR BUSINESS INTERRUPTION). HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
26 # WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
27 # OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
28 # ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
29
30
31 from pynq import MMIO
32 from pynq import PL
33 import time
34
35 __author__ = "Graham Schelle"
36 __copyright__ = "Copyright 2016, Xilinx"
37 __email__ = "pynq_support@xilinx.com"
38
39
40 RGBLEDS_XGPIO_OFFSET = 0
41 RGBLEDS_START_INDEX = 4
42 RGB_CLEAR = 0
43 RGB_BLUE = 1
44 RGB_GREEN = 2
45 RGB_CYAN = 3
46 RGB_RED = 4
47 RGB_MAGENTA = 5
48 RGB_YELLOW = 6
49 RGB_WHITE = 7
50
51
52 class MYRGBLED(object):
53     """This class controls the onboard RGB LEDs.
54
55     Attributes
56     -----
57     index : int
58         The index of the RGB LED, from 4 (LD4) to 5 (LD5).
59     _mmio : MMIO
60         Shared memory map for the RGBLED GPIO controller.
61     _rgbleds_val : int
62         Global value of the RGBLED GPIO pins.
63
64     """

```



```

65 _mmio = None
66 _rgbleds_val = 0
67
68 def __init__(self, index):
69     """Create a new RGB LED object.
70
71     Parameters
72     -----
73     index : int
74         Index of the RGBLED, from 4 (LD4) to 5 (LD5).
75
76     """
77     # print("Changed LED Driver to MYRGBLED.") # debugging only
78     if index not in [4, 5]:
79         raise ValueError("Index for onboard RGBLEDs should be 4 or 5.")
80
81     self.index = index
82     if MYRGBLED._mmio is None:
83         base_addr = PL.ip_dict["rgbleds_gpio"]["phys_addr"]
84         MYRGBLED._mmio = MMIO(base_addr, 16)
85
86 def on(self, color):
87     """Turn on a single RGB LED with a color value (see color constants).
88
89     Parameters
90     -----
91     color : int
92         Color of RGB specified by a 3-bit RGB integer value.
93
94     Returns
95     -----
96     None
97
98     """
99     if color not in range(8):
100         raise ValueError("color should be an integer value from 0 to 7.")
101
102     rgb_mask = 0x7 << ((self.index-RGBLEDS_START_INDEX)*3)
103     new_val = (MYRGBLED._rgbleds_val & ~rgb_mask) | \
104         (color << ((self.index-RGBLEDS_START_INDEX)*3))
105     self._set_rgbleds_value(new_val)
106
107 def off(self):
108     """Turn off a single RGBLED.
109
110     Returns
111     -----
112     None
113
114     """
115     rgb_mask = 0x7 << ((self.index-RGBLEDS_START_INDEX)*3)
116     new_val = MYRGBLED._rgbleds_val & ~rgb_mask
117     self._set_rgbleds_value(new_val)
118
119 def red_on(self):
120     """Turn on a single RGB LED with color value red.
121
122     Parameters
123     -----
124     color : int
125         Color of RGB specified by a 3-bit RGB integer value.
126
127     Returns
128     -----
129     None
130
131     """
132     # if color not in range(8):

```

```

133 #         raise ValueError("color should be an integer value from 0 to 7.")
134 new_val = (MYRGBLED._rgbleds_val ) | (RGB_RED << ((self.index-RGBLEDS_START_INDEX)
*3))
135 #         print(MYRGBLED._rgbleds_val)
136 #         print(new_val)
137 #         print( (RGB_RED << ((self.index-RGBLEDS_START_INDEX)*3)) )
138 self._set_rgbleds_value(new_val)
139
140 def red_off(self):
141     """Turn on a single RGB LED with color value red.
142
143     Parameters
144     _____
145     color : int
146         Color of RGB specified by a 3-bit RGB integer value.
147
148     Returns
149     _____
150     None
151
152     """
153     #         if color not in range(8):
154     #             raise ValueError("color should be an integer value from 0 to 7.")
155
156     new_val = (MYRGBLED._rgbleds_val ) &~ (RGB_RED << ((self.index-RGBLEDS_START_INDEX)
*3))
157     self._set_rgbleds_value(new_val)
158
159 def green_on(self):
160     """Turn on a single RGB LED with color value green.
161
162     Parameters
163     _____
164     color : int
165         Color of RGB specified by a 3-bit RGB integer value.
166
167     Returns
168     _____
169     None
170
171     """
172     #         if color not in range(8):
173     #             raise ValueError("color should be an integer value from 0 to 7.")
174
175     new_val = (MYRGBLED._rgbleds_val ) | (RGB_GREEN << ((self.index-RGBLEDS_START_INDEX)
*3))
176     self._set_rgbleds_value(new_val)
177
178 def green_off(self):
179     """Turn on a single RGB LED with color value red.
180
181     Parameters
182     _____
183     color : int
184         Color of RGB specified by a 3-bit RGB integer value.
185
186     Returns
187     _____
188     None
189
190     """
191     #         if color not in range(8):
192     #             raise ValueError("color should be an integer value from 0 to 7.")
193
194     new_val = (MYRGBLED._rgbleds_val ) &~ (RGB_GREEN << ((self.index-RGBLEDS_START_INDEX)
*3))
195     self._set_rgbleds_value(new_val)
196

```

```

197 def blue_on(self):
198     """Turn on a single RGB LED with color value blue.
199
200     Parameters
201     -----
202     color : int
203         Color of RGB specified by a 3-bit RGB integer value.
204
205     Returns
206     -----
207     None
208
209     """
210     # if color not in range(8):
211     #     raise ValueError("color should be an integer value from 0 to 7.")
212
213     new_val = (MYRGBLED._rgbleds_val ) | (RGB_BLUE << ((self.index-RGBLEDS_START_INDEX)
214 *3))
215     self._set_rgbleds_value(new_val)
216
217 def blue_off(self):
218     """Turn on a single RGB LED with color value red.
219
220     Parameters
221     -----
222     color : int
223         Color of RGB specified by a 3-bit RGB integer value.
224
225     Returns
226     -----
227     None
228
229     """
230     # if color not in range(8):
231     #     raise ValueError("color should be an integer value from 0 to 7.")
232
233     new_val = (MYRGBLED._rgbleds_val ) &~ (RGB_BLUE << ((self.index-RGBLEDS_START_INDEX)
234 *3))
235     self._set_rgbleds_value(new_val)
236
237 def status(self):
238     rgb_mask = 0x7 << ((self.index-RGBLEDS_START_INDEX)*3)
239     return ((MYRGBLED._rgbleds_val )& ~rgb_mask)
240
241 def pwm(self, color, interval):
242     """Turn on a single RGB LED with color value red.
243
244     Parameters
245     -----
246     color : int
247         Color of RGB specified by a 3-bit RGB integer value.
248
249     Returns
250     -----
251     None
252
253     """
254     for x in range(0, 1):
255         self.blue_on()
256         time.sleep(interval)
257         self.blue_off()
258
259 def write(self, color):
260     """Set the RGBLED state according to the input value.
261
262     Parameters
263     -----

```

```

263         color : int
264             Color of RGB specified by a 3-bit RGB integer value.
265
266         Returns
267         -----
268         None
269
270         """
271         self.on(color)
272
273     def read(self):
274         """Retrieve the RGBLED state.
275
276         Returns
277         -----
278         int
279             The color value stored in the RGBLED.
280
281         """
282         return (MYRGBLED._rgbleds_val >>
283                 ((self.index-RGBLEDS_START_INDEX)*3)) & 0x7
284
285     @staticmethod
286     def _set_rgbleds_value(value):
287         """Set the state of all RGBLEDs.
288
289         Note
290         -----
291         This function should not be used directly. User should call
292         'on()' , 'off()' , instead.
293
294         Parameters
295         -----
296         value : int
297             The value of all the RGBLEDs encoded in a single variable.
298
299         """
300         MYRGBLED._rgbleds_val = value
301         MYRGBLED._mmio.write(RGBLEDS_XGPIO_OFFSET, value)

```

Listing 6: Jupyter Notebook file myrgbled saved as *.py file.

4.2 Python code Listings Part II - LED Groove Bar

```

1
2 # coding: utf-8
3
4 # # Part II - LED Groove Bar
5 # Demonstrates how the LED Groove Bar level is set with slider SL1. The brightness can be
   chosen in four levels with slider SL2.
6 #
7 # LED Bar Brightness
8 # - 0 = off
9 # - 1 = low
10 # - 2 = medium
11 # - 3 = hight
12
13 # In[2]:
14
15
16 # Steup the PYNQ board
17 from pynq.overlays.base import BaseOverlay
18 base = BaseOverlay("base.bit")
19
20 from pynq.lib.pmod import Grove_LEDbar
21 from pynq.lib.pmod import PMOD_GROVE_G1 # Import constants
22 import ipywidgets as widgets
23 from IPython.display import display

```

```

24
25 # For delays
26 from time import sleep
27
28 # Global values
29 g_ledBrightness = 3
30 g_leds = 0
31
32 # defined functions
33 def handle_slider1_change(change):
34     global g_leds
35     ledbar.write_level(change.new, g_ledBrightness, 1)
36     g_leds = change.new
37 def handle_slider2_change(change):
38     global g_ledBrightness
39     g_ledBrightness = change.new
40     ledbar.write_level(g_leds, change.new, 1)
41     # ledbar.write_brightness(ledbar.read(), change.new)
42
43
44 # Instantiate Grove LED Bar on PMODA and on Pmod2Grove G1
45 ledbar = Grove_LEDbar(base.PMODA, PMOD_GROVE_G1)
46 ledbar.reset()
47
48 # Flash 2 extreme LEDs of the LED Bar in a loop, dubbging only
49 # for i in range(5):
50 #     ledbar.write_binary(0b1000000001)
51 #     sleep(0.5)
52 #     ledbar.write_binary(0b0000000000)
53 #     sleep(0.5)
54
55 # GUI
56 slider1 = widgets.IntSlider(min=0, max=10, value=0, description='SL1')
57 slider2 = widgets.IntSlider(min=0, max=3, value=0, description='SL2')
58
59 slider1.observe(handle_slider1_change, names='value')
60 slider2.observe(handle_slider2_change, names='value')
61
62 display(slider1, slider2)

```

Listing 7: Part II - LED Groove Bar Python code.

4.3 Python code Listings Part III - Music Synthesizer