Module: R5: RV-fpga

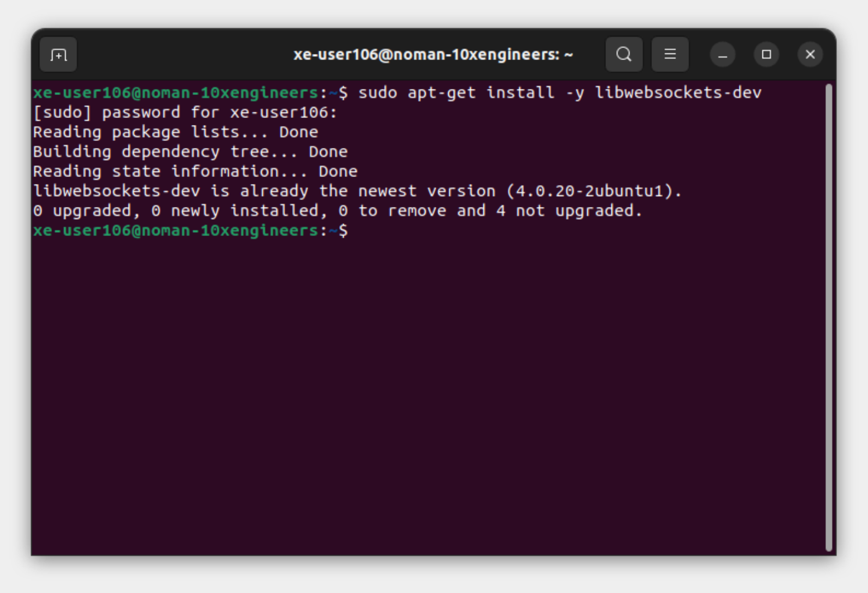
Section: Installations **Task:** Tools

Task 1.2

RVfpga-ViDBo

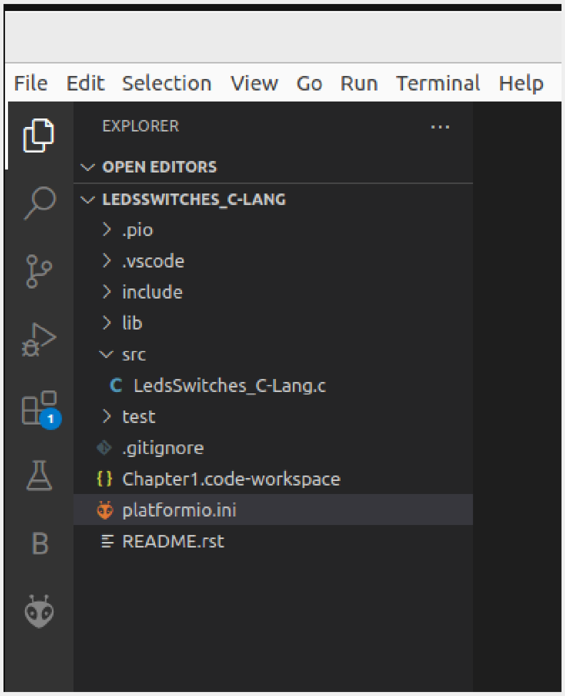
- **Testing:**
  - **RVfpga-ViDBo**
    1. For using this simulator in Ubuntu it is required to first install websockets library, which can be easily achieved by executing the following command in a terminal:

       ```
       sudo apt-get install -y libwebsockets-dev
       ```

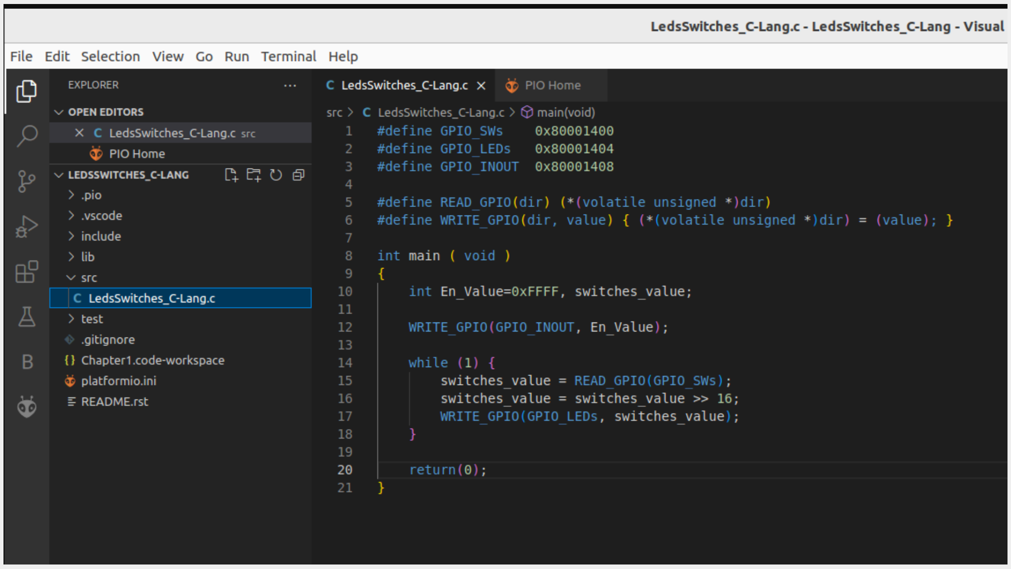


```
xe-user106@noman-10xengineers:~$ sudo apt-get install -y libwebsockets-dev
[sudo] password for xe-user106:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
libwebsockets-dev is already the newest version (4.0.20-2ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 4 not upgraded.
xe-user106@noman-10xengineers:~$
```

2. Opened the specified folder of example program in VS Code:



3. PlatformIO will now open this program, LedsSwitches_C-Lang, that reads the switch values on the Nexys A7 board and writes their value onto the LEDs on the board. We can view the LedsSwitches_C-Lang program by expanding the src folder and double-clicking on LedsSwitches_C-Lang.c.

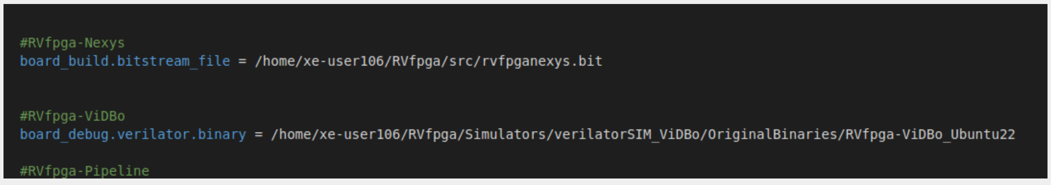


```c
#define GPIO_SWs    0x80001400
#define GPIO_LEDs   0x80001404
#define GPIO_INOUT  0x80001408

#define READ_GPIO(dir) (*(volatile unsigned *)dir)
#define WRITE_GPIO(dir, value) { (*(volatile unsigned *)dir) = (value); }

int main ( void )
{
    int En_Value=0xFFFF, switches_value;

    WRITE_GPIO(GPIO_INOUT, En_Value);

    while (1) {
        switches_value = READ_GPIO(GPIO_SWs);
        switches_value = switches_value >> 16;
        WRITE_GPIO(GPIO_LEDs, switches_value);
    }

    return(0);
}
```

4. Open file **platformio.ini**. Then, established the path to the provided RVfpga-ViDBo simulator binary by editing the following line

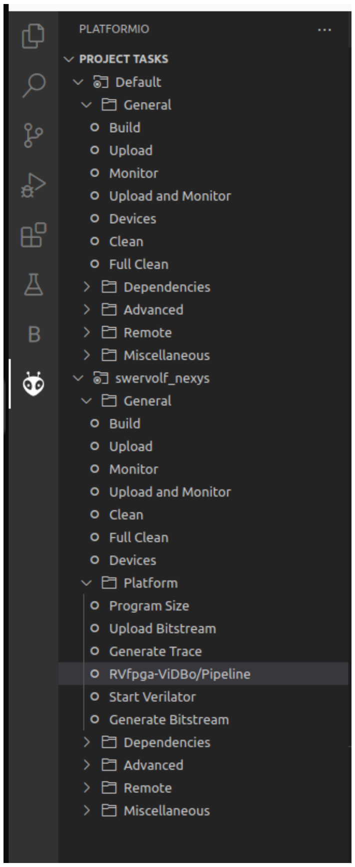
(replaced [Path-To-RVfpga] with the appropriate path in my system and chose the binary that corresponds to my Ubuntu version):

```
board_debug.verilator.binary = /home/xe-user106/RVfpga/Simulators/verilatorSIM_ViDBo/OriginalBinaries/RVfpga-ViDBo_Ubuntu22
```

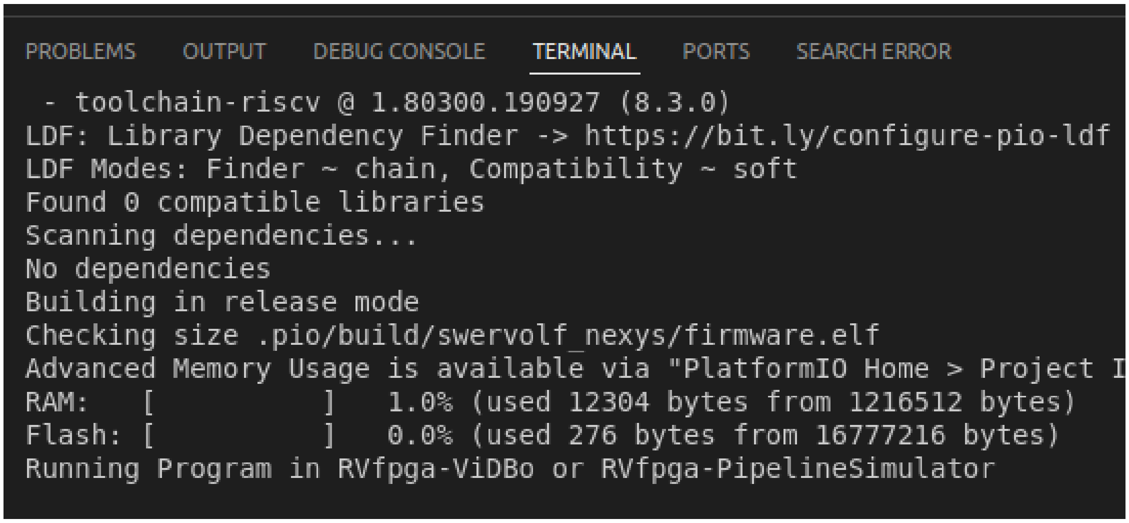


```
#RVfpga-Nexys
board_build.bitstream_file = /home/xe-user106/RVfpga/src/rvfpganexys.bit


#RVfpga-ViDBo
board_debug.verilator.binary = /home/xe-user106/RVfpga/Simulators/verilatorSIM_ViDBo/OriginalBinaries/RVfpga-ViDBo_Ubuntu22

#RVfpga-Pipeline
```

5. Executed the RVfpga-ViDBo simulator from PlatformIO:
   - Click on the PlatformIO button on the left side.
   - Expand Project Tasks > env:swervolf_nexys > Platform and clicked on **RVfpga-ViDBo/Pipeline**.



This first compiles the program and then launches the Verilator simulation of the RVfpga SoC running this program.

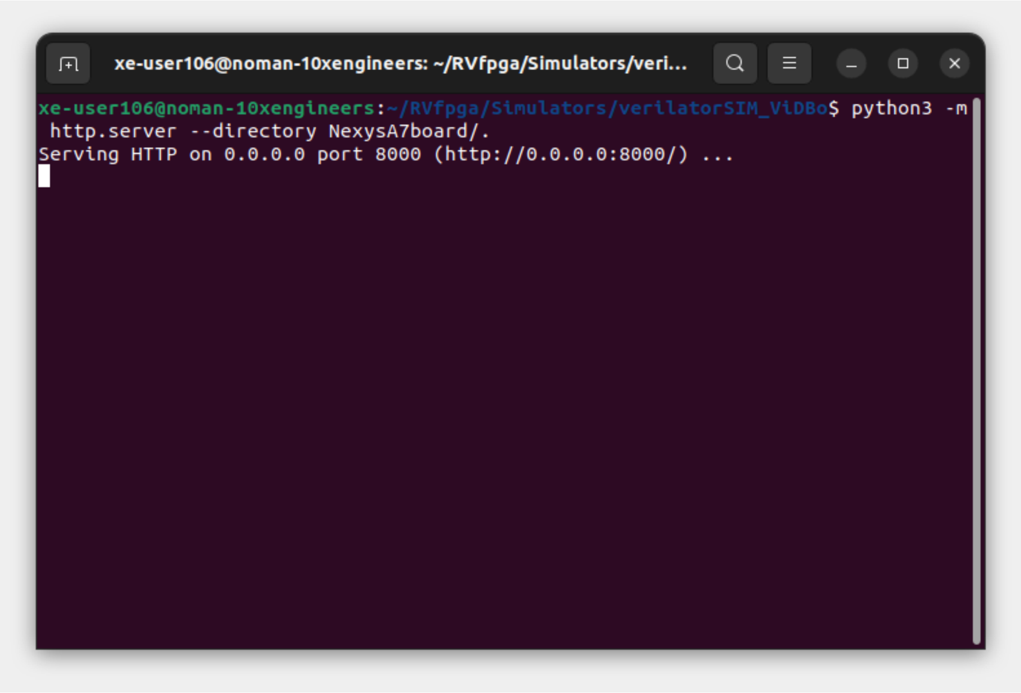


```
 - toolchain-riscv @ 1.80300.190927 (8.3.0)
LDF: Library Dependency Finder -> https://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 0 compatible libraries
Scanning dependencies...
No dependencies
Building in release mode
Checking size .pio/build/swervolf_nexys/firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project I
RAM:   [          ]   1.0% (used 12304 bytes from 1216512 bytes)
Flash: [          ]   0.0% (used 276 bytes from 16777216 bytes)
Running Program in RVfpga-ViDBo or RVfpga-PipelineSimulator
```

6. Once the RVfpga-ViDBo is executing, I launched the python server simulating the Nexys A7 board.
7. Open a terminal. Just go into the ***/RVfpga/Simulators/verilatorSIM_ViDBo*** directory.

   Execute the python server by running the following command:

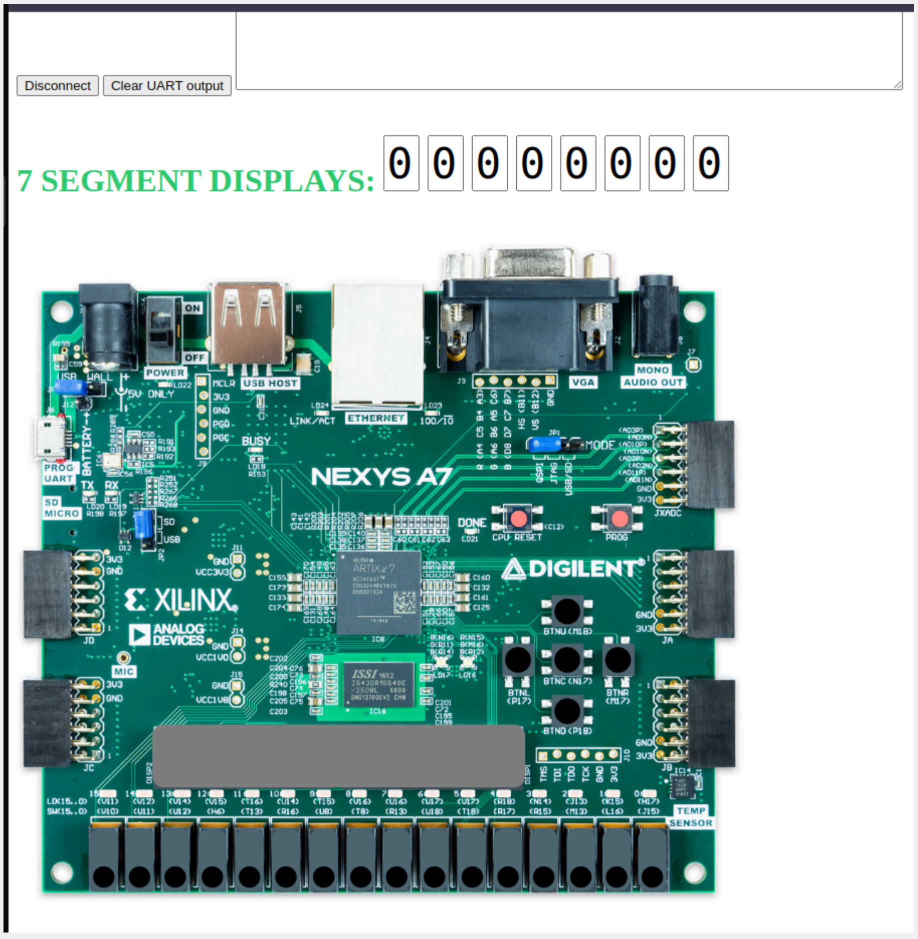
   ```
   python3 -m http.server --directory NexysA7board/.
   ```



```
xe-user106@noman-10xengineers:~/RVfpga/Simulators/verilatorSIM_ViDBo$ python3 -m
 http.server --directory NexysA7board/.
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

8. Open a browser and connect to http://localhost:8000/nexys-a7.html.

   Click on Connect to the board and test the program.



7 SEGMENT DISPLAYS: 0 0 0 0 0 0 0 0