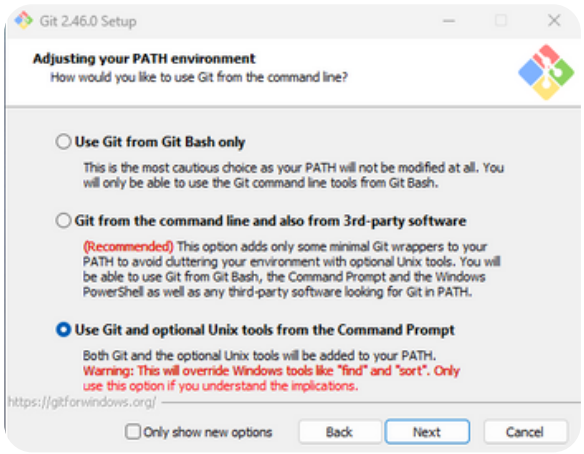
Tang Nano 9K Synthesis and Configuration Setup

Installing Git and Cloning the Repository

You can download Git for Windows from either https://git-scm.com/download/win or https://gitforwindows.org



Once you install Git for Windows and the applicable FPGA synthesis toolchain, you can clone the repository in the command line. This can be done either in a regular terminal or the Visual Studio Code (VS Code) terminal.

```
git clone https://github.com/yuri-panchul/basics-graphics-music.git
```

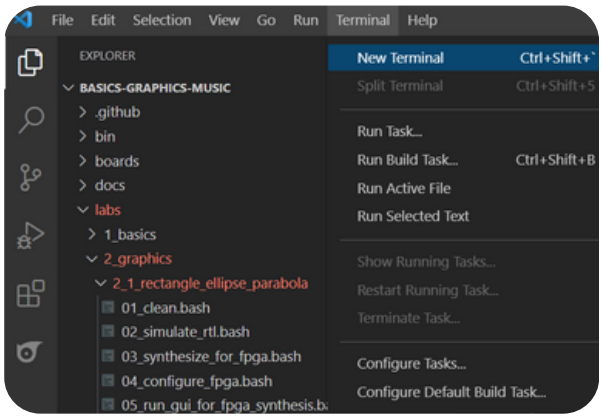
To run basics-graphics-music scripts under Linux, you need to install an open-source product called openFPGALoader, in addition to Gowin EDA. The installation instructions are at:

https://trabucayre.github.io/openFPGALoader/guide/install.html

We recommend to check the “Use Git and optional Unix tools from the Command Prompt”. The second choice should work as well, however if you check "Use Git from Git Bash only", you have to open a separate Git Bash console or use a Bash terminal in VS Code.

Creating a Terminal in VS Code

- Once you start VS Code, open the directory where you cloned the basic-graphics-music repository
- Then you create a terminal within VS Code



Then you enter the following commands

Under Windows:

```
cd .\labs\2_graphics\2_1_rectangle_ellipse_parabola\
bash 03_synthesize_for_fpga.bash
```

Under Linux you would enter:

```
cd labs/2_graphics/2_1_rectangle_ellipse_parabola
./03_synthesize_for_fpga.bash
```

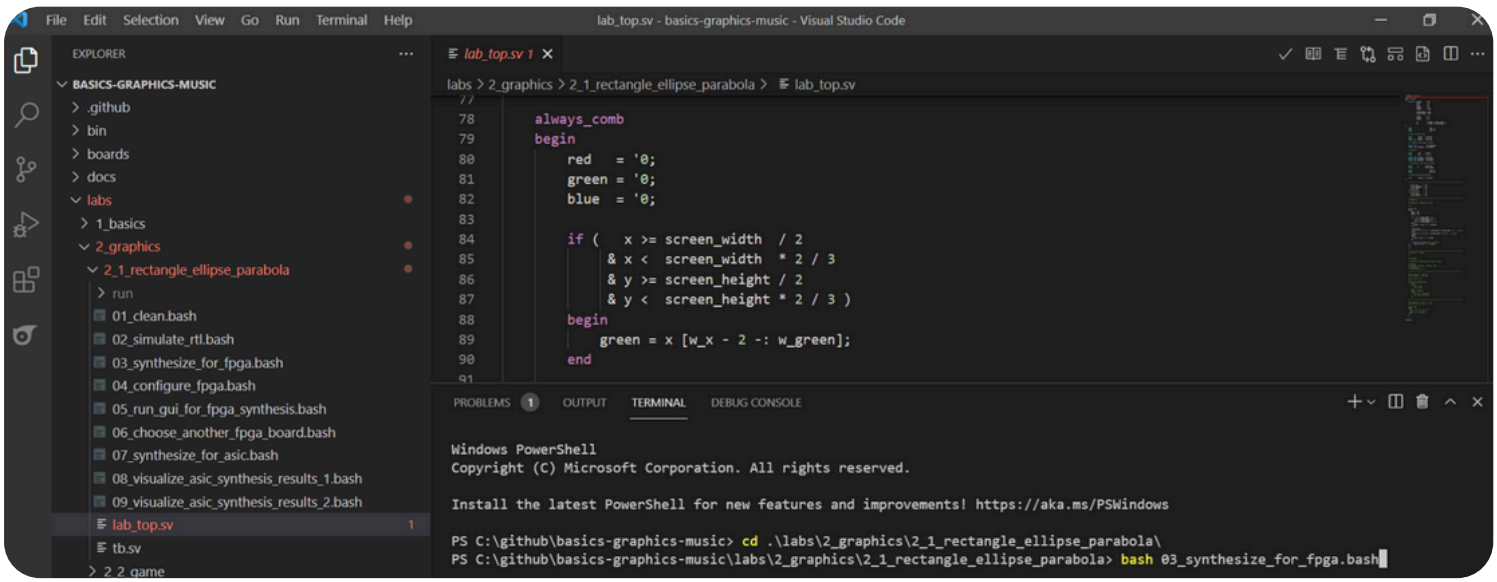
For Tang Nano 9K with 4.3-inch LCD and attached TM1638-based board, enter the number that corresponds to the configuration *tang_nano_9k_lcd_480_272_tm1638*.

If you don't have a TM1638 board, enter *tang_nano_9k_lcd_480_272_tm1638*.

If you are using the open-source Yosys-based toolchain OSS CAD Suite instead of Gowin EDA, use *tang_nano_9k_lcd_480_272_tm1638_yosys*.

If you connect the Tang Nano 9K board to the HDMI display, use the *tang_nano_9k_hdmi_tm1638* configuration.

2_1_rectangle_ellipse_parabola Terminal Screenshot



Tang Nano 9K Synthesis and Configuration Setup

OSS CAD Suite

You can experiment with synthesizing the examples using open-source toolchain OSS CAD Suite instead of Gowin EDA.

To do this, you need to:

1. Download a variant of the toolchain for your platform from https://github.com/YosysHQ/oss-cad-suite-build/releases

2. Unpack the downloaded file in some location, such as *~/oss-cad-suite*.

3. In Linux:

```
cd ~/oss-cad-suite
source environment
```

4. In an example, run the script *06_choose_another_fpga_board.bash* and select the number which corresponds to *tang_nano_9k_lcd_480_272_tm1638_yosys*, if you use TM1638 board, or *tang_nano_9k_lcd_480_272_no_tm1638_yosys* otherwise.

5. Now you can run the script *03_synthesize_for_fpga.bash* to synthesize the example. Note that not every basics-graphics-music example is compatible with the OSS CAD Suite at the moment.

GOWIN IDE Installation Location

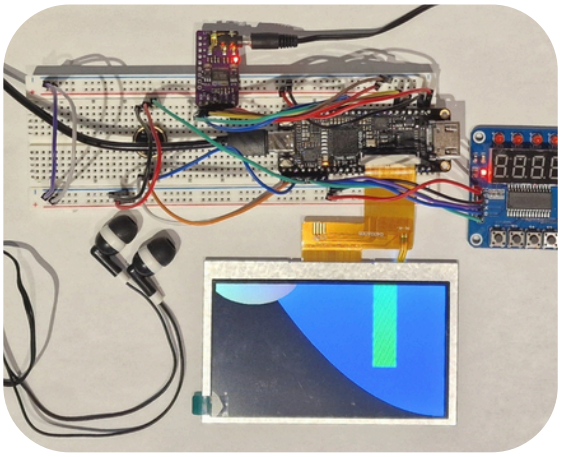
Now the synthesis script starts running. For this particular board, which uses Gowin FPGA, the script expects to find Gowin IDE installed in one of the default locations:

Linux:

1. $HOME/Gowin
2. $HOME/gowin
3. /opt/Gowin
4. /opt/gowin
5. /tools/Gowin
6. /tools/gowin

Windows:

1. C:\Gowin
2. D:\Gowin
3. E:\Gowin



A custom Gowin installation home location can be set by GOWIN_HOME environment variable such as:
`GOWIN_HOME=/home/verilog`

You can also use *GOWIN_VERSION_DIR* to specify the version subtree location, such as:
`GOWIN_VERSION_DIR=/home/verilog/gowin/0.99`

If the synthesis script failed at the end because the board is not connected, you don't need to re-run the synthesis; connect the board and run configuration only:

Checking if Everything on the Board is Working

Linux:

```
./04_configure_fpga.bash
```

Windows or Linux:

```
bash 04_configure_fpga.bash
```

The recommended labs to run to check that everything on the board is working:

***1_basics/1_09_hex_counter* – checks TM1638.**
***2_graphics/2_1_rectangle_ellipse_parabola* – checks graphics.**
***3_music/3_1_note_recognizer* – checks the microphone.**
***3_music/3_3_note_synthesizer* – checks the audio decoder.**
***4_microarchitecture/4_2_fifo/4_2_3_fifo_with_better_debug_1* – a microarchitectural lab, use keys to push and pop values from a FIFO queue.**
***5_cpu/5_1_schoolriscv* – A single-cycle minimalistic CPU which implements a subset of RISC-V architecture**