Vanguard Rainbow Tutorial

Get an updated version of this document at http://vgkits.co.uk/project/rainbow/tutorial.pdf . Text collaboratively maintained by @vgkits on github here

Equipment Provided

In your packs you should have…

- A Vanguard board (NodeMCU-M board pre-installed with Micropython, our libraries and startup scripts)
- An 8-pixel WS2811 RGB LED display
- A USB A-to-MicroB cable
- A 2-way red and black power cable with 2.54mm female sockets
- A single jumper wire with 2.54mm female sockets
- A 4xAA battery pack with switch and 2.54mm female sockets

Preparing your Components

Retail packs from @VGKits should have the following steps already completed…

- The NodeMCU-M board comes pre-installed with Micropython firmware, a set of useful libraries and a startup script for your Rainbow project. This turns it into a Vanguard board.
- WS2811 LED displays should have 2.54mm male pins soldered on both ends
- 4xAA battery packs should have 2.54mm female sockets crimped on the red and black cables

If you want to wipe and reinstall your board with a newer build, (or are configuring another supplier's ESP8266/ESP8285 to be a Vanguard board) follow the steps in *Appendix A* below.

Connecting the LED Display

- You will need…
    - The 8-pixel Display
    - The Jumper wires

Look on the back of your LED display. It should have three pins going into it, and three pins going out. Find the end of the display with the following three connections.

- Power - labelled 5V or 4-7VDC
- Ground - labelled GND
- Data In - labelled DIN

Use the jumper wires to attach from the **LED display** to the **Vanguard board** as follows

- 5V Power —> 5V (use red side of power cable)
- GND —> GND (use black side of power cable)
- DIN —> 14, (labelled '14', use the yellow jumper wire)

(Optionally) Connecting the Battery pack

The Vanguard board can run directly from USB power by attaching with the orange cable to a laptop, or using a mobile phone charger. However, for short demonstrations you can use the battery pack. The battery pack will drain quickly (within hours) if you leave it powered this way.

- You will need…
    - A 4xAAA battery pack with switch and crimped 2.54mm female sockets
    - The Vanguard board

Look at the label on the back of your Vanguard board. Find the following pins.

- Voltage In - labelled VIN
- Ground - labelled GND

Look at the battery pack. Find the power wire (red) and ground wire (black).

Attach from the Battery pack to the Vanguard board as follows

- Power (red) —> VIN
- GND (black) —> GND

Connecting to the Vanguard Shell via Wifi

- You will need…
    - The Orange USB cable
    - The Vanguard board
- Step 1: To plug in the Vanguard board to give it power
    - The Vanguard's USB Micro-B socket can be fragile. Be careful when plugging and unplugging the cable. If the cable doesn't slide in easily, try turning the plug upside down

- Step 2: To connect to the newly available wifi access point
    - Wifi networks have a name or ESSID. Normally your laptop takes a few minutes to refresh the list of available network ESSIDs in menus. Turning on and off your laptop's wifi can accelerate this.
    - The Vanguard appears with an ESSID containing a unique number like vanguard-34fe57
    - The default Wifi password is `vanguard`
- Step 3: To load the console in the browser
    - Wait for a bit
    - Connect your browser to http://192.168.4.1
    - When the page appears click on Connect
    - Type the password `vanguard` when challenged

Connecting to the Vanguard via Serial

A more formal way to connect to the Shell uses a 'serial' or 'UART' connection over the USB wire.

The Vanguard uses a CH340 UART module, (install the drivers from here) and the default connection speed (baud rate) is 115200. If the Vanguard has connected successfully, press CTRL+C to get control of the shell, and you should see three chevrons like this…

```
>>>
```

…and you can continue to the next step `Issuing Commands`

On all platforms

We have provided a script in the vanguard repository utilities folder called shell.py which makes it easy to connect to the console over a serial link using miniterm.py from pyserial. See Appendix A for instructions to download and configure our repository. Once complete…

- Use `cd` to change to the **utilities** folder
- Type `python3 shell.py` and press `Return`

On Linux

On a linux machine you can connect using Gnu Screen…

- Type `screen /dev/ttyUSB0 115200` and press `Return`

On Windows or Mac

Various terminal programs are available, such as Putty, GNU Screen, miniterm.py

Troubleshooting

- If the console has gone blank, but the chevrons don't appear try the following until they do
  - try pressing the `CTRL+C` keys together to try and kill any sequence of steps which was already running. This should return you to the chevrons.
  - visit Appendix A: Configuring your Vanguard, to reset your Vanguard board to 'factory' configuration

Issuing commands

After connecting, we should be in what is known as a REPL - a Read, Eval, Print Loop.

This means that the Vanguard board is…

- ***READING*** the commands we send
- ***EVALUATING*** the commands (running them, often generating a result)
- ***PRINTING*** the result (showing the result on screen)
- ***LOOPING*** back (returning to the READING step - over and over again)

Let's try it out to learn some fundamental programming concepts.

Core Programming Concepts

- Values
  - Type `4+4` and press `Return`. What happens? That was an arithmetic expression, which results in a number.
  - Type `4*4` with an asterisk instead and press `Return`. (hint `x` is treated as a letter in a computer language).
  - Enter `'Hello' + 'World'`
- Names
  - Enter `square = 4*4`
  - Enter `square` (we just assigned a value to a name, so we can refer to it later)
  - Enter `capital = 'Paris'`
  - Enter `capital`
- Steps
  - Enter `raw_input('What is the capital of Colombia? ')`
  - Enter `capital = raw_input('What is the capital of Colombia? ')`
- Groups of Values: Lists, Dictionaries

    - Enter `sequence = [3,4,5,6,7,8]`
    - Enter `sequence`
    - Enter `sequence[0]`
    - Enter `sequence[1:4]`
    - Enter `[num*num for num in sequence]`
- Groups of Steps: Blocks and Functions
    - Enter `range(2,6)`
    - Enter `def square(x):`
    - Press Tab, then Enter `return x*x`
    - Delete all spaces/tabs then press `Return`
    - Enter `square(4)`

Colors, Lists and Loops

- Lists
    - Enter `yellow = [255,255,0]`
    - Enter `yellow` and it should show your list
    - Enter `red` this should show a list which was previously defined
    - Enter `setPixel(0,red)`
    - Enter `setPixel(1,green)`
    - Enter `setPixel(2,blue)`

Libraries

Libraries contain reference implementations of functions for example `sqrt(4*4)` in the python `math` library calculates square roots.

Type the following two lines

```
from math import sqrt
sqrt(4*4)
```

Appendix A: Configuring your own NodeMCU

- Install **python3** on your laptop.
    - Our config scripts run in *python3* **and** *python2*. However, *micropython* is a dialect of *python3*. Our advice; learn one language version
- Install **pip3** on your laptop
- Install **pyserial**, **adafruit-ampy rshell**, and **esptool** using *pip3*. For example, linux and mac users would run commands in the console as follows…

```
sudo pip3 install pyserial
sudo pip3 install adafruit-ampy
sudo pip3 install esptool
sudo pip3 install rshell
```

- Get our scripts from the vanguard repository …
    - …download a snapshot of our scripts
      OR
    - …install git on your laptop and check out our repository if you are familiar with git and want to track future changes or contribute
- Download or unzip the repository contents to a location where you can find them. Launch a console and change to the `utilities` directory. Now run…

  `python deployall.py`
- Unplug and replug your Vanguard board to restart it.

This should complete the process of downloading and flashing the Micropython image for your board, uploading modules (libraries of code), and optionally uploading a behaviour to run on startup. The startup file is configured by the value `mainPath` in the utilities/config.py .

Appendix B: Configuring Laptop for Serial Connection

If you are using a home machine, you will need to

- Ensure the CH340 USB to UART drivers are installed
    - a restart of your machine is typically needed after this step
- Ensure Python3 is installed
    - Python version 3 is preferred, the same version as your board will run, but Python 2 will do

After completing these steps you should be able to plug in your NodeMCU-M, (see under `Powering Up`), then run a terminal or cmd.exe and copy-paste the following…

```
    python3 -c "from serial.tools.list_ports import comports;print([item.de
vice for item in comports()])"
```

If you have successfully completed configuring your laptop, the at least one serial port should be listed. If it reports `[]` (an empty list between square brackets), then the device or drivers have a problem.

on Linux the proper device may be named `['ttyUSB0']` or on Mac OS it may report `['tty.SLAB_USBtoUART']` . On both Linux and Mac OS the full port name should be prefixed by

`/dev/` making it `/dev/ttyUSB0`.

On Windows it may report `['COM4']`, meaning the Port Name is `COM4`

If more than one port appears in the list, unplug your NodeMCU, and re-run the command to identify which port appears and disappears from the list.