Cockle Badge Tutorial

A possibly newer copy of this document is available at http://shrimping.it/project/badge/tutorial.pdf

Equipment Provided

In your packs you should have...

- A USB A-to-MicroB 2 Amp (high-current) cable (@ShrimpingIt-orange)
- A NodeMCU v2 (Cockle)
- A WS2811 RGB LED display
- A 3-way jumper cable with 2.54mm pitch female sockets
- A 3xAAA switched battery pack with 2.54mm pitch female sockets
- A transparent Lanyard-style badge-holder
- A Lanyard cord (@ShrimpingIt-orange)

Preparing your Equipment

If you are in an official @ShrimpingIt workshop, pre-configured laptops will be provided and you can skip to the next step. To use your own machine, scroll down to Appendix A: Configuring your laptop, then come back here when you are finished.

If you have a Cockle provided by @ShrimpingIt, then Micropython should already be installed and you can skip to the next step. If not, then visit Appendix B: Configuring your Cockle, then come back here when you are finished.

Connecting to the Cockle

- You will need...
    - The Orange USB cable
    - The NodeMCU v2 (Cockle)
- Step 1: Plug in the Cockle
    - The cockle’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
    - @ShrimpingIt official USB cables can handle 2Amps. Cheaper cables often cannot power up

the Cockle or the lights.

- Step 2: Launch a 'Console'
    - On @ShrimpingIt linux laptops (with Kupfer launcher) press `CTRL+SPACE` together, type `terminal` and press `Return`
    - On Mac OS press the `Apple+Space` keys together, type `terminal` and press `Return`
    - On Windows, go to Start and Run, type `cmd.exe` and press `Return`
- Step 3: Connect over UART to the Cockle
    - On @ShrimpingIt linux laptops, type `screen /dev/ttyUSB0 115200`
    - Mac OS users can launch `screen` as well, except your UART will be called something like `/dev/tty.SLAB_USBtoUART`
    - On Windows, install Putty as a terminal emulator instead of Gnu Screen

If the Cockle has connected successfully, you should see three chevrons like this...

```
>>>
```

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

Troubleshooting

- If the console has gone blank, but the chevrons don't appear try the following until they do
    - try pressing `Return` in case the chevrons were already sent before we connected, and they should be sent again
    - try pressing the `CTRL+C` keys together in case a command was already running, and they should then appear to await a new command
    - visit Appendix B: Configuring your Cockle, to install Micropython and repeat

Issuing commands

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

This means that the Cockle 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)`

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)
```

Connecting the LED Display

- You will need...
    - The 8-LED Display (this is long and thin)
    - 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 board with the following three connections.

- Data In - labelled DIN
- Power - labelled 4-7VDC
- Ground - labelled GND

Use the jumper wires to attach to the Cockle as follows

- DIN —>D2
- 4-7VDC —> Vin
- GND —> GND

Now connect to your Cockle and run the following lines to set up the display and send a color...

```
from neopixel import NeoPixel
from machine import Pin
count = 8
output = Pin(4)
display = NeoPixel(output, count)
red = (255,0,0)
display[0] = red
display.write()
```

```
black = (0,0,0)
display[0] = black
display.write()
```

```
display[0]=(255,0,0)
display[1]=(0,255,0)
display[2]=(0,0,255)
display.write()
```

Introducing For

To set all the pixels, we can use a `for` loop block. A for loop in python runs a set of commands once for every item in a list. We can create a list of the different pixel positions between 0 and count using `range(count)`.

Note the first line finishes with a colon, and the commands to be repeated in the loop are indented (only `display[pos] = red` is repeated in the loop below).

```
for pos in range(count):
    display[pos] = red
display.write()
```

```
for pos in range(count):
    display[pos] = (0, 0, pos//count*255)
display.write()
```

```
for pos in range(count):
    display[pos] = (0, 0, (count - pos)//count*255)
display.write()
```

Introducing While

A `while` loop block repeats commands ***while*** an expression remains true. This is often used with

```
greenVals = [pos*255//count for pos in range(count)]
redVals = [(count - pos)*255//count for pos in range(count)]
blueVals = [0 for pos in range(count)]
while True:
    for pos in range(count):
        display[pos] = (greenVals[pos], redVals[pos], blueVals[pos])
    display.write()
```

```
from time import sleep
greenVals = [pos*255//count for pos in range(count)]
redVals = [(count - pos)*255//count for pos in range(count)]
blueVals = [0 for pos in range(count)]
offset = 0
while True:
    for pos in range(count):
        display[(pos + offset) % count] = (greenVals[pos], redVals[pos], bl
ueVals[pos])
    display.write()
    sleep(0.05)
    offset = offset + 1
```

Intro to Voltage Divider

USB is stepped from e.g. 5V down to 3.3V through a Low Dropout (LDO) regulator

Powering from batteries

- Wire through badge to 3.3V
    - 4x AAA battery pack
        - very cheap batteries - more life from Duracells
- Alternative Mobile Phone charger attached to Micro-USB
    -
        - potentially last for days

Finishing your Light-Up-Lanyard

The badges you've been provided with can be used to carry your Cockle, Battery pack and LED display.

Appendix A: Configuring your Own Laptop

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

- Ensure the CP2102 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 badges will run, but Python 2 will do

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

```
python -c "import serial.tools.list_ports;print serial.tools.list_ports.com
ports()"
```

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 it may report `['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 and replug your NodeMCU, and identify which port appears and disappears from the list.

Appendix B: Configuring your own NodeMCU

See https://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/intro.html or http://bit.ly/2f0JdZL for short

Appendix C: Programming over Wifi

See https://docs.micropython.org/en/latest/esp8266/esp8266/tutorial/repl.html or http://bit.ly/2eELBby for short.