Once you have the RPi/display working you will find it not very convenient to update the displayed text, needing to log into the RPi, use an editor to modify the code, execute the code, examine the display, change the code, repeat as necessary until you get the display presentation you like. My simple script allows the user to update the eink display info from any browser. The RPi with display automatically connects to your network via wifi and can be used remotely. There is no login. It is on your LAN only for the update, then unplugged since the display does not need power to retain the image.

Before using my script, you should follow the guide [The Seeed Studio E-Ink 2.13 inch Triple colour Display Python guide](https://www.raspberryconnect.com/projects/42-hardware-addons/177-the-seeed-studio-e-ink-2-13-inch-3-colour-display-python-guide) to setup your Seeed Studio 2.13 inch Triple colour e-ink display on your Raspberry Pi Zero W. Portions of that guide are copied below with my notes for my configuration.



**2.13 inch e-ink display on a Raspberry Pi zero/w**

The display includes a 3v3 to 5v switch (for RPi set to 3.3V).

I found the following is Not necessary, it is default:

1. **import** RPi.GPIO as GPIO
3. # for GPIO numbering, choose BCM
4. GPIO.setmode(GPIO.BCM)

 Programming:

Another resources is from <https://github.com/soonuse/epd-library-python>. This is the one I used.

There are different drivers for different screen sizes and models. I have the 2.13 inch e-paper B screen which uses the b or bc driver.

For the screen to work you need a driver file which will have a name like epd2in13b.py and the config file named epdconfig.py in the lib folder or epdif.py if you are using the soonuse libraries. I actually put these two files in the same folder as my script.

These drivers may need to be modified to work with the Seeed e-ink driver board.**I did this, but not sure it is necessary.** This is because the gpio pin assignments need altering. If you edit the epdconfig or epdif files in a text editor you will see a pin assignment section like this

**#Pin definition**

* RST\_PIN = 17
* DC\_PIN=25
* CS\_PIN=8
* BUSY\_PIN=24

These need to be changed to;

**#Pin definitions**

* RST\_PIN = 13
* DC\_PIN=6
* CS\_PIN=5
* BUSY\_PIN=19

The screen uses SPi which uses pins 10 & 11. Make sure that the SPi driver has been enabled in the Raspberry Pi Configuration menu. The two buttons on the controller board use GPIO bcm pins 20 & 21.

Once you have the driver files and modified config file in a folder you can then add your own python3 program to bring the screen to life.

The screen requires two images to make the final image. One needs to represent black and other for red. These should be a 2 colour bmp files of 212 x 104 pixels. They can be created by hand in a image editor and saved to your program folder or created on the fly using pythons [Pillow graphics library](https://pillow.readthedocs.io/en/stable/). This is the simplest option if you intend to use text as Pillow can be used to convert text into an image as well as build the required image from multiple elements.

Install Pillow with **pip**:

$ sudo pip3 install Pillow==7.2 Note 7.2 is latest version compatible with python 3.5 ,which is installed on my Pi zero.

Then install spidev for the SPI bus control

$ sudo pip3 install spidev

To turn on SPi on the RPi, use $ sudo raspi-config. After enable in Interfacing options, rpi must be rebooted.

To check if SPI is enabled, we will use the ls command to list the available interfaces. Run the following command to list all of the current files within the /dev/spidev directory.

$ ls -l /dev/spidev\*

From this command, you should get back the following result.

pi@raspberrypi:~ $ ls -l /dev/spidev\*

crw-rw---- 1 root spi 153, 0 Jul 23 12:43 /dev/spidev0.0

crw-rw---- 1 root spi 153, 1 Jul 23 12:43 /dev/spidev0.1

You can see the two available channels through our SPI connection, channel 0, and channel 1.

At this point in the guide, you should now have the SPI interface enabled on your Raspberry Pi.

After copying the soonuse/epd-library-python to the Rpi, you can now try

$ python3 main.py

Once you have everything working you will find it not very convenient to update the displayed text, needing to log into the RPi, use an editor to modify the code, execute, examine the display, change the code, repeat as necessary until you get the display presentation you like. The following allows the user to update the eink display info from any browser. The RPi with display automatically connects to your network via wifi and can be used remotely. There is no login. It is on your LAN only for the update, then unplugged since the display does not need power to retain the image.

Install remi which is a GUI library for Python applications that gets rendered in web browsers.

$ sudo pip3 install remi # this installed remi-2021.3.2. May be necessary to do this again as last step, see next steps below.

Next step install PySimpleGUI, which Transforms the tkinter, Qt, WxPython, or Remi (browser-based) GUI frameworks into a simpler interface. The window definition is simplified by using Python core data types understood by beginners (lists and dictionaries).

$ sudo pip3 install pysimpleguiweb # This uninstalled remi-2021.3.2, then installed remi-2020.3.10 and then installed pysimpleguiweb-0.39.0.6 # note make sure it is -0.39.0.6 since -0.39.0 won’t work with -2021.3.2 Actually I ended up removing the pysimpleguiweb install and just put the one file PySimpleGUIWeb.py in mycode folder. Also see <https://pysimplegui.readthedocs.io/en/latest/#web-version-remi>

$ sudo pip3 install --upgrade remi==2021.3.2 # this reinstalls 2021.3.2

Finally,

I created a script which runs in any browser to edit the eink display: **myEinkWeb.py**

The **myEinkWeb.service** ensures the script is run after boot up.

Hope it is useful to you.

---- Duane, April 2022