Installing R on the Raspberry Pi

Installing R on the Raspberry Pi turned out to be trickier than I thought. I was only able to install version 3 package sources, but I wanted to have version 4.0.2. I tried following the guides posted here and here but always ran into the problem that the package `r-base-core` was not available for 4.0.

I ended up compiling R from source, which was surprisingly simple. I loosely followed the guide posted h

```
wget https://cran.rstudio.com/src/base/R-4/R-4.0.2.tar.gz
tar zxvf R-4.0.2.tar.gz
# uncomment first line in /etc/apt/source.list
sudo apt-get build-dep r-base
./configure --prefix=/opt/R/4.0.2 --enable-R-shlib --with-blas --with-lapack
```

If you run into the error:
`configure: error: PCRE2 library and headers are required, or use --with-pcre1 with UTF-8 support`
do

```
sudo apt-get install libpcre2-dev
sudo make
sudo make install

sudo ln -s /opt/R/4.0.2/bin/R /usr/local/bin/R
sudo ln -s /opt/R/4.0.2/bin/Rscript /usr/local/bin/Rscript
```

and that's already it. You should now have R ready to run on your Raspberry Pi.
Since we are operating on a headless Raspberry Pi, there is no point in trying to install RStudio.
I found, however, Nvim R to be an awesome headless substitute (if you are happy using vi, that is).

Setting up the E-Paper display

I simply followed the guide posted on the waveshare wiki.

Enable SPI

`sudo raspi-config` and then choose Interfacing Options -> SPI -> Yes.

Install BCM2835 libraries

Check what the newest version is. At the time of writing, it was `1.68`.

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.68.tar.gz
tar zxvf bcm2835-1.68.tar.gz
cd bcm2835-1.68/
sudo ./configure
sudo make
sudo make check
sudo make install
```

Install wiringPi

```
sudo apt-get install wiringpi
```

If you are on a Raspberry Pi 4, you'll need to update it

```
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
```

Communicating with the display from R

The Raspberry Pi can easily communicate with the display via some C or python library.
Luckily, both can be integrated in R. I opted for python, due to the amazing R package `reticulate` whic
the communication between R and python a breeze. I'd be interested, though to package the C library as
(eventually).

```
#python3
sudo apt-get update
sudo apt-get install python3-pip
sudo apt-get install python3-pil
sudo apt-get install python3-numpy
sudo pip3 install Raspberry Pi.GPIO
sudo pip3 install spidev
```

You'll also need `git` to download some additional code.

```
sudo apt-get install git -y
```

Then, clone the repository of waveshare that includes the needed libraries.

```
sudo git clone https://github.com/waveshare/e-Paper
```

The library we will need is in the folder `RaspberryPi&JetsonNano/python/lib`.

python code for communication

The python code below is used to send the bitmap file `screen-output.bmp` to the display.

```
#!/usr/bin/python
# -*- coding:utf-8 -*-
import sys
import os
import time
libdir="/home/pi/e-Paper/RaspberryPi&JetsonNano/python/lib"
if os.path.exists(libdir):
    sys.path.append(libdir)

from waveshare_epd import epd7in5_V2
from PIL import Image,ImageDraw,ImageFont

try:
    epd = epd7in5_V2.EPD()
    epd.init()
    Himage = Image.open("screen-output.bmp")
```

```
    epd.display(epd.getbuffer(Himage))
    epd.sleep()
except KeyboardInterrupt:
    epd7in5.epdconfig.module_exit()
    exit()
```

Note that I am not a python expert so this code may well be awful (but it works). The code bellow shows h this script (`display.py`) from within R.

```
library(reticulate)

use_python("/usr/bin/python3")
py_run_file("display.py")
```

Build the dashboard in R

Now that we can communicate with the display from python and know how to use python from within R, w think about what we want to display on the dashboard and how we implement it. Remember, all we have 800×480 pixels in black and white. Apart from this constrained though, you are free to put whatever you w I opted for a standard weather, calendar, and date setup, together with some randomly generated pixel ar implementation of a pixel-sprite-generator). The plotting is done entirely with `ggplot2`.

To reproduce my dashboard setup, simply clone the repository

`git clone` https://github.com/schochastics/e-Paper-dashboard.git

To use the code, you need to get an API key from openweathermap.org and set up the `gcalendr` package with your google calendar. The rest should work out of the box (except that it wor questions in the comments or on twitter.

Of course you do not have to follow my build. The following code snippet should be sufficient to build you scratch.

```
library(reticulate)
#load additional libraries, such as e.g. ggplot2
use_python("/usr/bin/python3")

#build your plot object

ggsave("raw-output.bmp",p,width=5,height=3,dpi = 160)
system("convert -colors 2 +dither -type Bilevel -monochrome raw-output.bmp sc
py_run_file("display.py")
```

The parameters set in `ggsave()` makes sure that the output is a 800×480 bmp file and the line thereafte true black and white image (which speeds up the rendering on the display). For the second line to work y `imagemagick`.

```
sudo apt-get install imagemagick
```

Here is an example dashboard with my code from github.

Fri Oct 16

12°C Manchester, UK
clouds

up: 07:38 down: 18:10

Tasks&Meetings

Testing Calendar
Oct 16 2020 15:30 - 16:30

Write R Code
Oct 16 2020 16:30 - 17:00

Write Blog Post
Oct 16 2020 17:00 - 18:00

Eat Pizza
Oct 16 2020 18:00 - 19:00

Entertain Daughter
Oct 16 2020 19:00 - 20:00