How to backup files in the middle of (almost) nowhere

Petr Pulc

Faculty of Information Technology,
Czech Technical University
Prague, Czech republic

7. 10. 2017

What you want:
- portable device
- automatic backup on media insertion
- as cheap as possible

Assumptions:
- You have some parts lying around
- You are willing to bodge a little

Solution?
- Single board PC
- HDD (SSD if rich)
- LEDs (segment display better)

Single board PC:

- Versatile
- Affordable
- Small
- GPIO!

Banana Pi
+ Dual Core
+ SATA
+ Able to be powered from battery

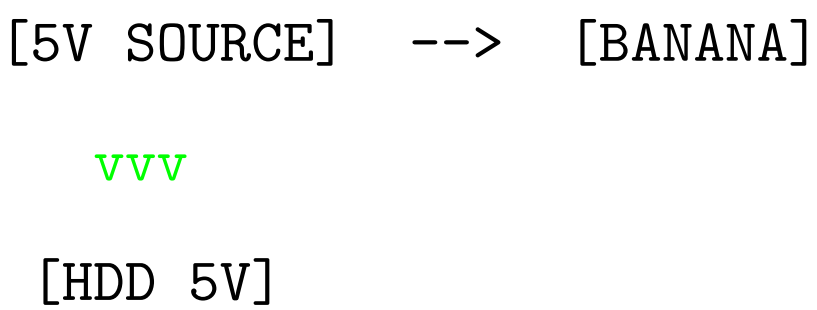
Challenge 1 – Power

```
 [BAT 3,6V]   -->   [BANANA]

? STEP-UP ?

  [HDD 5V]
```

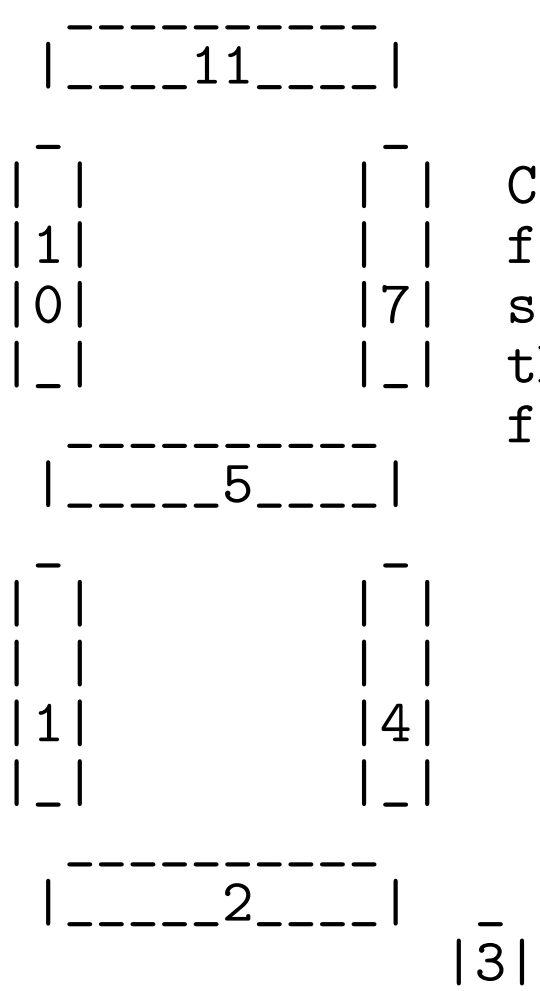
Compromise 1 – USB Power



Challenge 2 – Minimise user interface

- Use what you have lying around
- Show when ready
- Sync on media insertion
- Show progress without full display
- Show amount of free space on HDD

Solution 2.1 – 7 Segment display



Common anodes:
first digit - 12
second digit - 9
third digit - 8
fourth digit - 6

- 12x GPIO required (up to 17 available on CONN3)
  - 4 to select digit (raise to 3,3V)
  - 8 to select segments (drop to 0)
- 8x 110 Ohm resistors
- Piece of old IDE cable
- Solder

display .py

```
#!/usr/bin/env python3
import time

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)

SEGMENTS = [18, 16, 12, 10, 3, 8, 5, 7]
NUMBERS = [19, 15, 13, 11]
```

```
CHARS = {' ': (1, 1, 1, 1, 1, 1, 1, 1),
         '0': (0, 0, 0, 1, 0, 0, 0, 1),
         '1': (1, 1, 0, 1, 1, 0, 1, 1),
         # ...
         ',': (1, 1, 1, 1, 1, 0, 1, 1),
         '%': (1, 0, 1, 1, 1, 0, 1, 1),
         'K': (1, 0, 1, 0, 0, 1, 1, 1),
         'M': (0, 1, 1, 1, 0, 0, 1, 1),
         'G': (0, 0, 1, 1, 0, 0, 0, 1),
         'T': (1, 0, 1, 0, 0, 1, 0, 1),
         'd': (1, 1, 0, 0, 0, 0, 0, 1),
         'o': (1, 1, 1, 0, 0, 0, 0, 1),
         'n': (1, 1, 1, 0, 0, 0, 1, 1),
         'e': (0, 0, 1, 0, 0, 1, 0, 1),
         # ...
         }
```

```
def init():
    for p in SEGMENTS:
        GPIO.setup(p, GPIO.OUT, initial=1)
    for p in NUMBERS:
        GPIO.setup(p, GPIO.OUT, initial=0)

def out_zip(pins, vals):
    for p, v in zip(pins, vals):
        GPIO.output(p, v)

def clear():
  for p in SEGMENTS:
      GPIO.output(p, 1)
  for p in NUMBERS:
      GPIO.output(p, 0)
```

```
def show(text, duration=1):
    for _ in range(0, duration * 50):
        number = 0
        for char in text[:4]:
            out_zip(SEGMENTS,
                    CHARS.get(char,
                              (1, 0, 0, 0, 0, 0, 0, 1)
                             ))
            GPIO.output(NUMBERS[number], 1)
            time.sleep(0.004)
            GPIO.output(NUMBERS[number], 0)
            number += 1
```

show.py

```
#!/usr/bin/env python3
import display
import sys
display.init()
display.show(sys.argv[1].rjust(4), 2)
```

Solution 2.2 – Show when ready – /etc/rc.local:

```
/usr/local/bin/gpio mode 2 out
/usr/local/bin/gpio write 2 0
# same for GPIO 3, 7, 12

/usr/local/bin/gpio mode 0 out
/usr/local/bin/gpio write 0 1
# same for GPIO 1, 4, 5, 8, 9, 15, 16
```

Solution 2.3 – Sync on media insertion

/etc/udev/rules.d/99_sync.rules:

```
KERNEL=="sd?1", SUBSYSTEMS=="usb", ACTION=="add",
RUN+="???"
```

Challenge 2.3.1 – What should we run?

- Long scripts get killed
- Multiple USB devices at once, one only once

Solution 2.3.1.1 – Service

/etc/udev/rules.d/99_sync.rules:

```
KERNEL=="sd?1", SUBSYSTEMS=="usb", ACTION=="add",
RUN+="/bin/systemctl --no-block start sync.service"
```

/etc/system.d/system/sync.service:

```
[Unit]
Description=SD backuper

[Service]
Type=simple
ExecStart=/opt/mount_n_sync.sh
```

Solution 2.3.1.2 – Pass device identifier

/etc/udev/rules.d/99_sync.rules:

```
KERNEL=="sd?1", SUBSYSTEMS=="usb", ACTION=="add",
RUN+="/bin/systemctl --no-block start sync@%k.service"
```

/etc/system.d/system/sync@.service:

```
[Unit]
Description=SD backuper
BindsTo=dev-%i.device

[Service]
Type=simple
ExecStart=/opt/mount_n_sync.sh %I
```

Challenge 2.3.2 – Mount and sync

- Sync media independently (different id on format)
- Mount manually
- (Show remaining space)
- Copy only new files
- Unmount and clean-up
- (Show remaining space)

Solution 2.3.2.[^4]

/opt/mount_n_sync.sh:

```
# Get information on block device
export `blkid -o export /dev/$1`
```

```
# Mount
mkdir "/media/$UUID"
mount -o ro "$DEVNAME" "/media/$UUID"
```

```
# Show free before backup
/opt/show.py Free
/opt/show_df.py
```

```
# Do backup
/opt/rsync.py $UUID
```

```
# Unmount and clear mount point
umount "$DEVNAME"
rmdir "/media/$UUID"

sync
```

```
# Show free after backup
/opt/show.py Free
/opt/show_df.py
```

show_df.py

Almost as simple as `df -h /mnt/backup...`

```
import subprocess
import display

with subprocess.Popen(['df', '-h', '/mnt/backup'],
                      stdout=subprocess.PIPE) as p:
    data = p.stdout.readlines()

display.init()
display.show(str(data[1]).split()[3].rjust(4), 2)
```

Challenge 2.3.2.4

The display.show IS BLOCKING

... for now it is helpfull, but:

We need to get progress DURING the sync!

Solution 2.3.2.4

/opt/rsync.py:

```
import subprocess
import sys
from threading import Thread
import display

display.init()

def print_thread():
    global text
    while True:
        display.show(text, 1)
        if text == 'done' or text == 'FAIL':
            display.show(text, 5)
            break
```

```
def rsync_thread():
    global text
    tty = open('/dev/tty1', 'w')

    with subprocess.Popen(['rsync', '-a',
                           '--info=progress2,flist0',
                           '/media/'+sys.argv[1],
                           '/mnt/backup/'],
                          stdout=subprocess.PIPE,
                          bufsize=1,
                          universal_newlines=True) as p:
        for line in p.stdout:
            tty.write(line)
            out = line.split()
            if len(out) > 2:
                text = out[1].rjust(4)
    if p.returncode == 0:
        text = 'done'
    else:
        text = 'FAIL'
```

```
text = 'init'
thread1 = Thread(target=print_thread)
thread2 = Thread(target=rsync_thread)
thread1.start()
thread2.start()
thread2.join()
thread1.join()
```

DEMO

Incomplete source code can be found on Github:

https://github.com/petrpulc/segment-display

Thank you