

# Using CERNBox for a CERN OpenDays's Selfie-Station

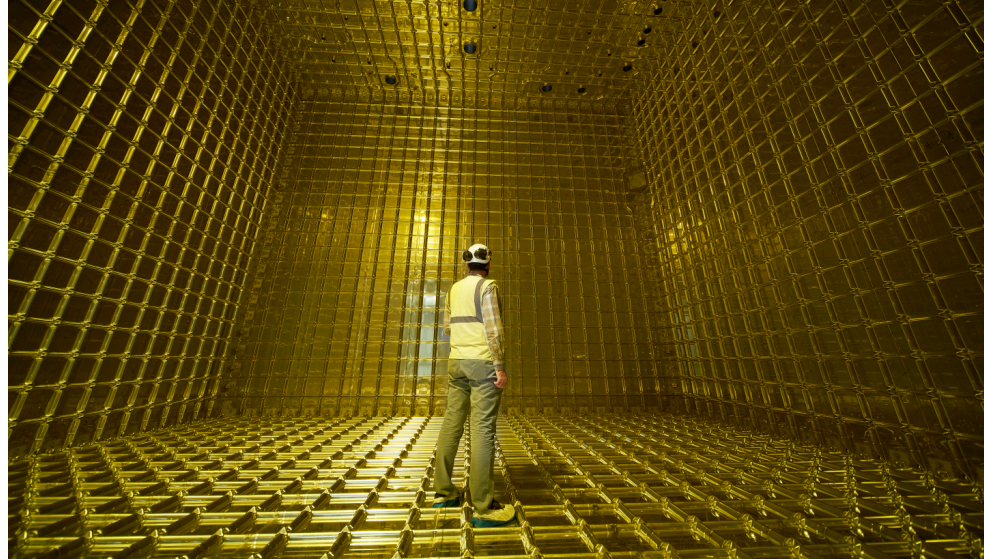
Manuel J. Rodriguez



CERNBox User Forum

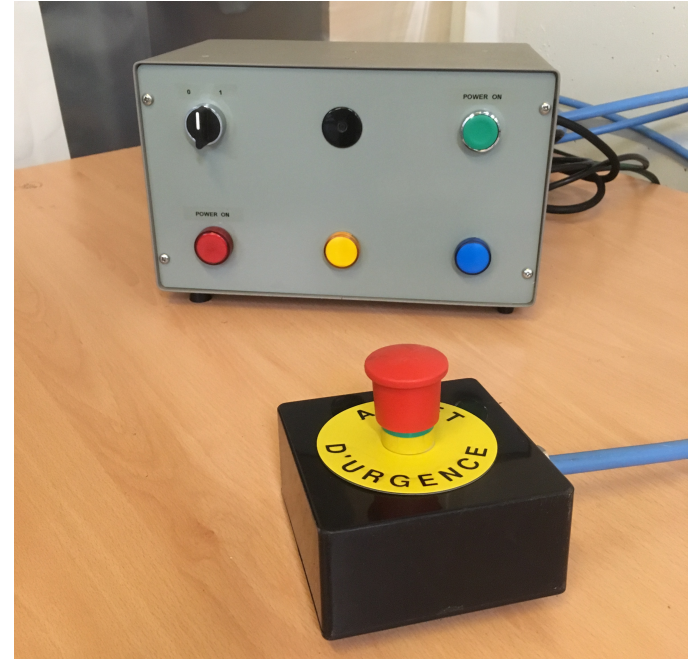
# Selfie Station for the Open Days

- For the last CERN Open Days we made a mockup of the inside of our detector
- We wanted our visitors to feel like in the image
- And we wanted them to keep it as a memory



# Selfie Station for the Open Days

- The idea was to press a button and have a picture
- We designed a “machine” to take pictures



# Selfie Station for the Open Days

- We wanted to keep it low cost.
- So the selfie station was Raspberry Pi based.
- However, we did not know how to give the pictures to users.



# Selfie Station for the Open Days

- However, we did not know how to give the pictures to users.
- Solutions:
  - Ask them to take a picture of the picture:
    - Bad quality 😞
  - Buy a mini photo printer to print them
    - Expensive 💰💰
    - Not eco-friendly (open days were asked to be paperless) 🌳😞
  - Get a link to allow them to download it! 😁
    - Full resolution, multiples copies, easy to share...

# How to get a link?

- I contacted the CERNBox team, asking them if there was a way to automatically upload a file, share it, and get its link.
- They replied that it was possible using the “reva-cli” tool, but...
- We are using a Raspberry Pi (ARM-based)
- So, they needed to cross-compile the tool to be ran on the Pi.
- Once we had a way to upload the images and get the link, we just needed to generate a workflow

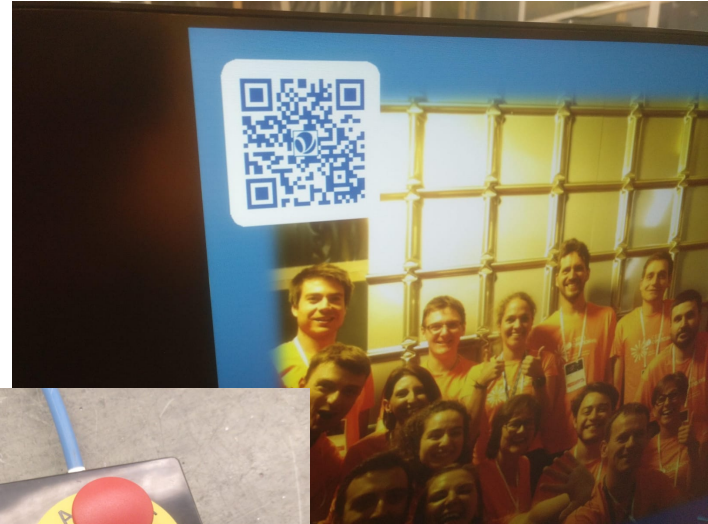
# Selfie Station workflow

- The user press the button
- Pi takes a photo
- Adds a frame to the photo
- Upload the picture to CERNBox using `curl POST`



# Selfie Station workflow

- Share it and get the public link using the `reva-cli` tool
- Generate a QR code with the link information
- Overlay the QR on the image to be displayed
- The image is hold on the screen until the user release the push button





# Code

- All the code is written in python
- It allows us easily to interact with the IO of the Pi
- Available on GitLab:

<https://gitlab.cern.ch/mjrodrig/selfistation>

```
def start_capture():
    camera.annotate_text = "3"
    sleep(1)
    camera.annotate_text = "2"
    sleep(1)
    camera.annotate_text = "1"
    camera.annotate_text = ""
    filepath = take_picture()
    camera.annotate_text = "Saving..."
    addFrame(filepath)

def addFrame(filepath):
    frame = Image.open(framepath)
    img = Image.open(filepath)
    img.paste(frame, (0,0),frame)
    img.save(filepath)

    uploadFile(filepath, img)

def uploadFile(filepath, img):
    subprocess.run(['curl', '-X', 'PUT', 'https://cernbox.cern.ch/cernbox/webdav'+filepath, '-data-binary', 'g'+filepath, '-u', 'npweb:Guta=Veve'],capture_output=True)
    getLink(filepath, img)

def getLink(filepath, img):
    mystring = subprocess.run(['~/home/pi/dev/rev-a-ctrl', 'sharing', 'public-link-create', filepath], stdout=subprocess.PIPE).stdout.decode('utf-8')
    url = "https://cernbox.cern.ch/index.php/3/" + mystring.split("\n")[0].split(": ")[1]
    genQR(url, img)

def genQR(urlpath, img):
    #urlpath = str(filepath)

    qrIng = qrcode.make(urlpath,
                        error_correction=qrcode.constants.ERROR_CORRECT_L,
                        version=1,
                        box_size=5,
                        border=4)

    qrIng_blue = Image.new('RGB',qrIng.size)
    qrIng_blue.paste(qrIng)

    data = np.array(qrIng_blue) # "data" is a height x width x 3 numpy array
    red, green, blue = data.T # Temporarily unpack the bands for readability

    # Replace white with red... (Leaves alpha values alone...)
    black_areas = (red == 0) & (blue == 0) & (green == 0)
    data[...,:][black_areas.T] = (14, 84, 161) # Transpose back needed

    qrIng_blue = Image.fromarray(data)
```



Thank you!