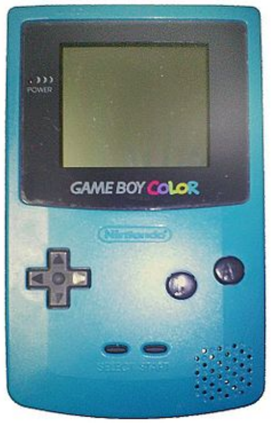
It is amazing what you can do with 128 pixels

(Using Telegram as an Interface for an LED wall)

Xavier Orduña
Python Meetup BCN
26th November

Back to the Origin

(84 x 47 pixels)



(160 x 144 pixels)

(2.560 x 1.600 pixels)

The Challenge

Paul Durov 11:40 PM
A paper plane logo is not cool. Try a banknote plane…

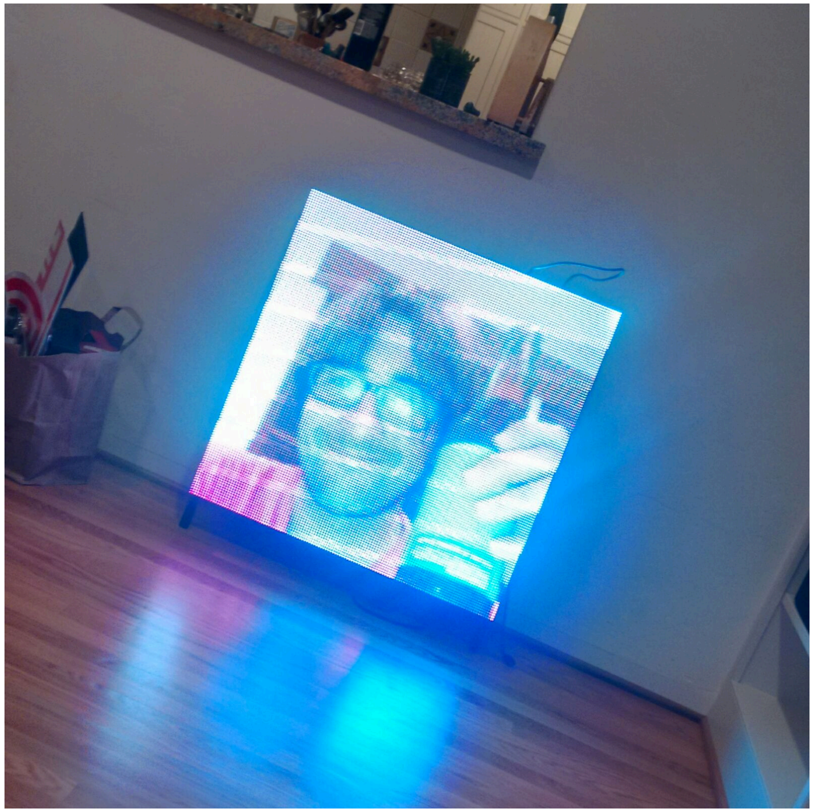
+1 (646) 724-1844 11:38 PM
Thanks for the video you sent me, I forwarded it to my friends!

Kim Dotcom 11:31 PM
You guys sure you don't want to host software and video here?

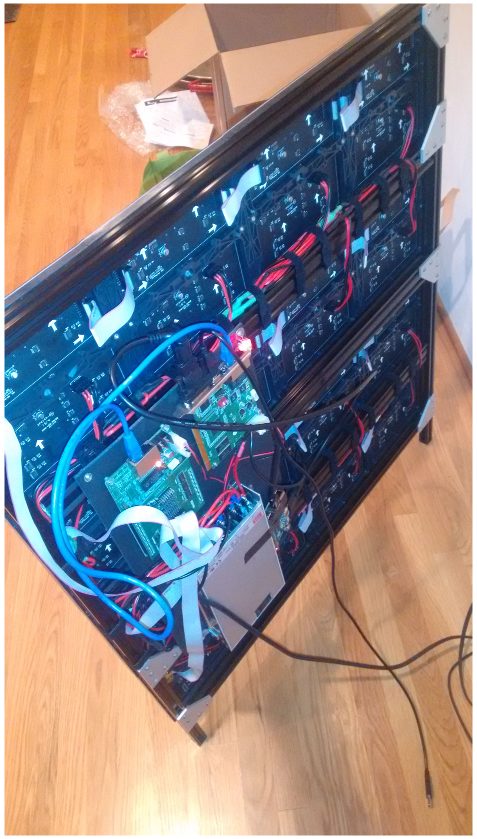
Mark Z 11:28 PM
I see Telegram runs pretty fast. Is it HTML5?

Telegram 11:20 PM
Paul,
We detected a login into your ac…

Weekend Plans! 11:15 PM
Mark Z left chat



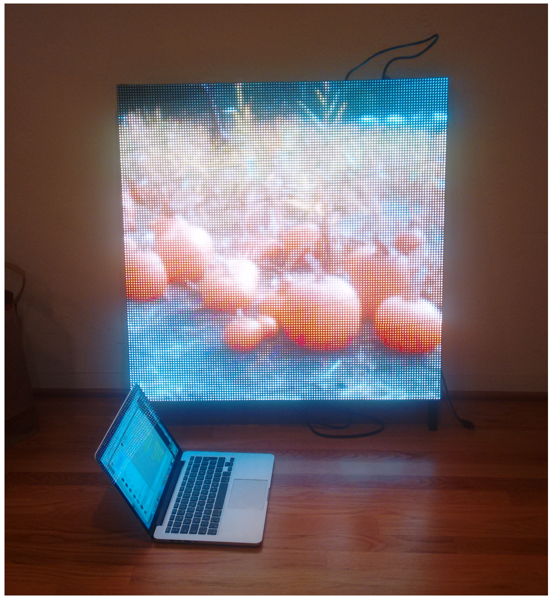
The Screen



128 px

128 px

80 cm

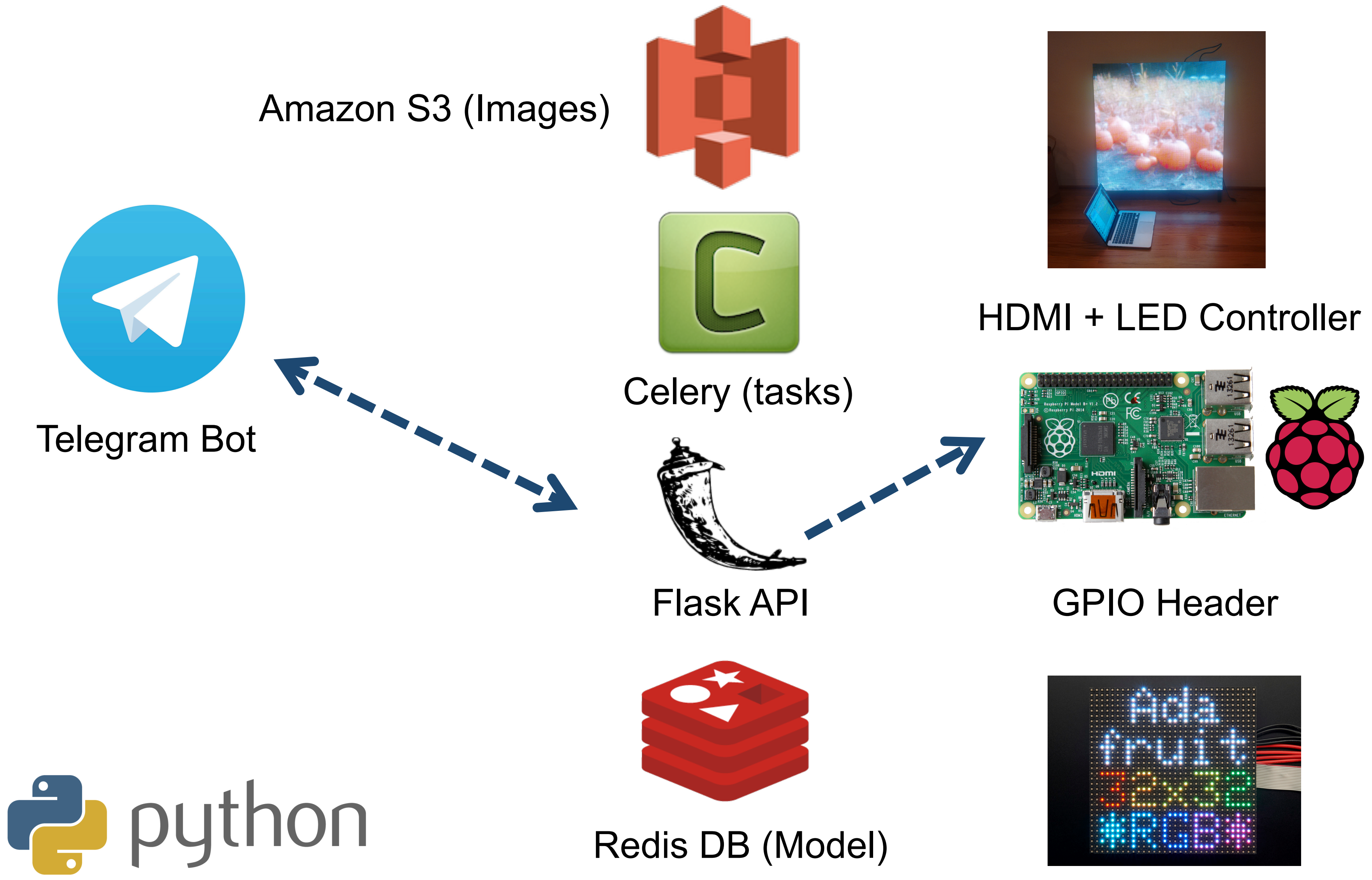


80 cm

A lot of wiring!!

About 1000 EUR in electronics!

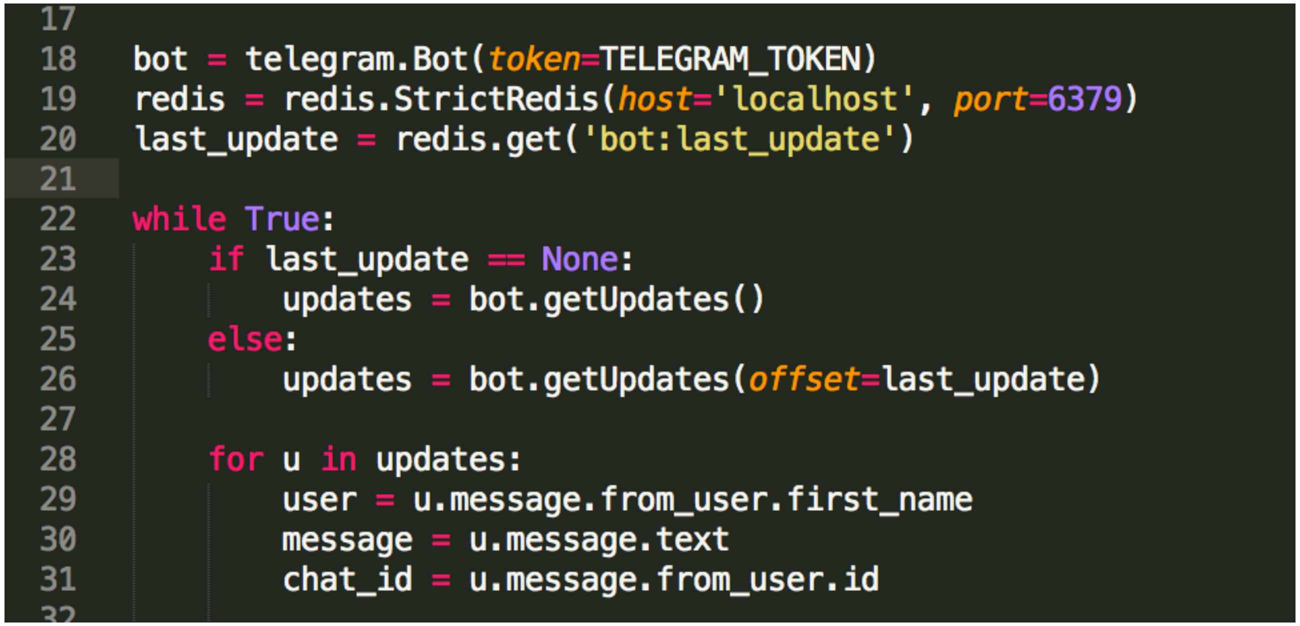
Architecture



Telegram Bot

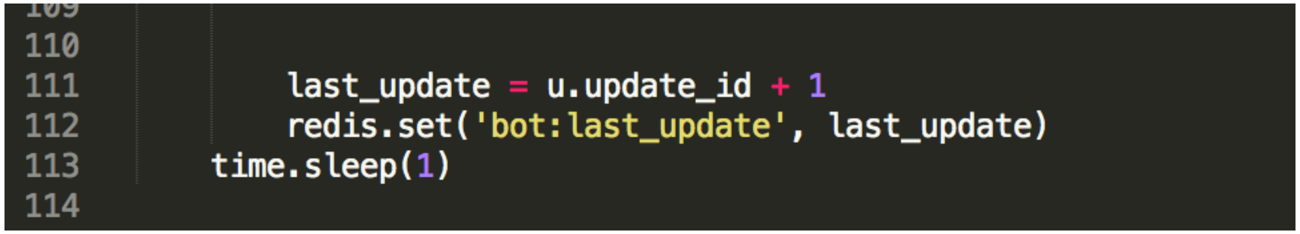
- API: https://core.telegram.org/bots/api
- Python Library: https://github.com/leandrotoledo/python-telegram-bot
- New bot token: @BotFather
- Bot name: walledbot

Telegram Bot



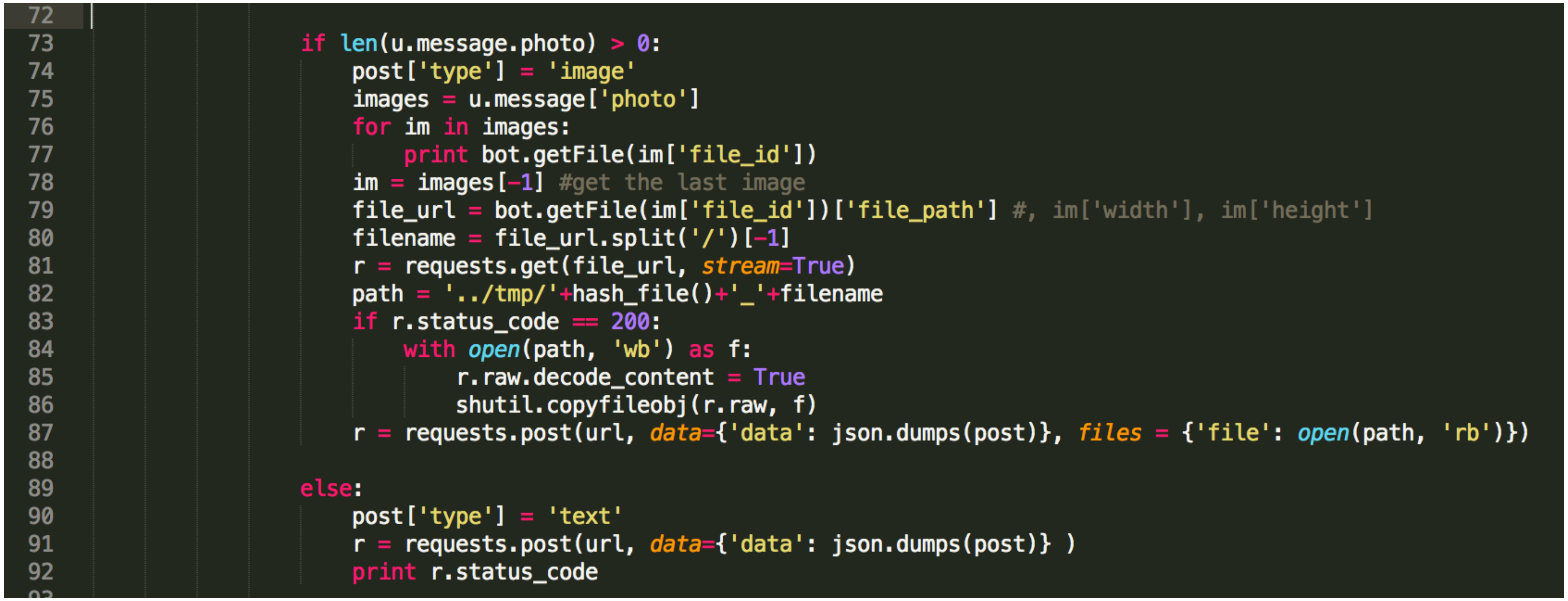
```python
bot = telegram.Bot(token=TELEGRAM_TOKEN)
redis = redis.StrictRedis(host='localhost', port=6379)
last_update = redis.get('bot:last_update')

while True:
    if last_update == None:
        updates = bot.getUpdates()
    else:
        updates = bot.getUpdates(offset=last_update)

    for u in updates:
        user = u.message.from_user.first_name
        message = u.message.text
        chat_id = u.message.from_user.id
```

(here goes the logic with the messages)



```python


        last_update = u.update_id + 1
        redis.set('bot:last_update', last_update)
    time.sleep(1)
```

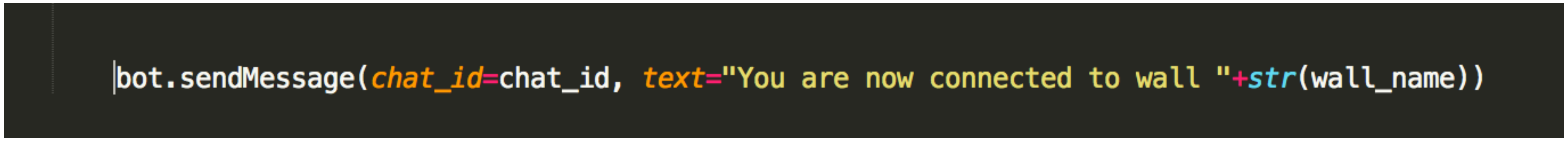
Telegram Bot



```python

                if len(u.message.photo) > 0:
                    post['type'] = 'image'
                    images = u.message['photo']
                    for im in images:
                        print bot.getFile(im['file_id'])
                    im = images[-1] #get the last image
                    file_url = bot.getFile(im['file_id'])['file_path'] #, im['width'], im['height']
                    filename = file_url.split('/')[-1]
                    r = requests.get(file_url, stream=True)
                    path = '../tmp/'+hash_file()+'_'+filename
                    if r.status_code == 200:
                        with open(path, 'wb') as f:
                            r.raw.decode_content = True
                            shutil.copyfileobj(r.raw, f)
                    r = requests.post(url, data={'data': json.dumps(post)}, files = {'file': open(path, 'rb')})

                else:
                    post['type'] = 'text'
                    r = requests.post(url, data={'data': json.dumps(post)} )
                    print r.status_code
```

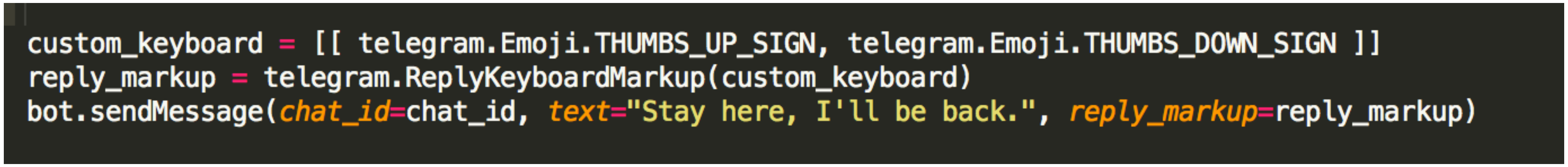
(gets a photo or text and posts it to our API)

Telegram Bot



```
bot.sendMessage(chat_id=chat_id, text="You are now connected to wall "+str(wall_name))
```

(Send a message to the user)



```
custom_keyboard = [[ telegram.Emoji.THUMBS_UP_SIGN, telegram.Emoji.THUMBS_DOWN_SIGN ]]
reply_markup = telegram.ReplyKeyboardMarkup(custom_keyboard)
bot.sendMessage(chat_id=chat_id, text="Stay here, I'll be back.", reply_markup=reply_markup)
```

(Send a keyboard to the user)

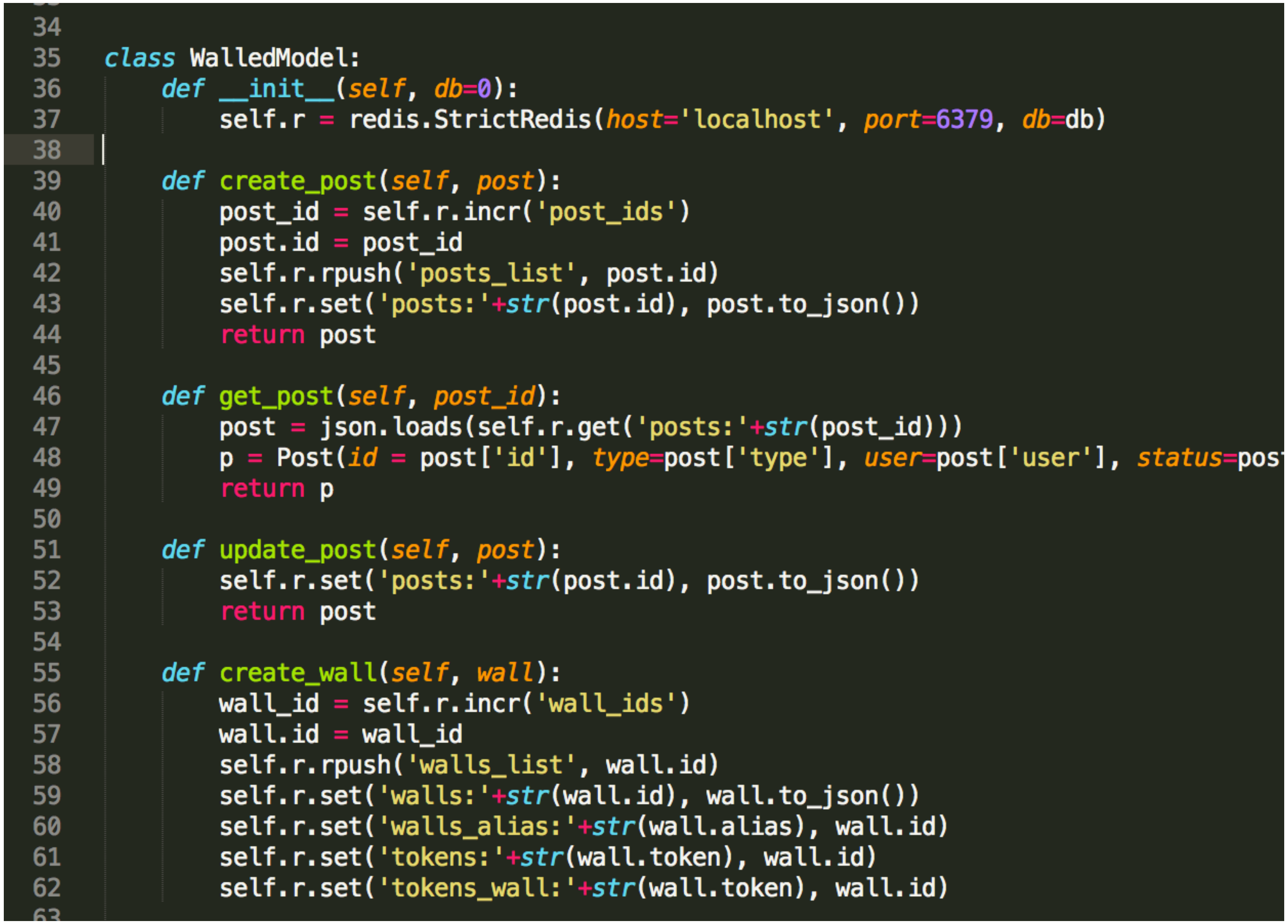
Model in Redis

- The most easy and FUN database
- You can represent complex models (http://redis.io/topics/twitter-clone)
- Key, value storage with steroids (sets, lists, hashes)
- All operations are atomic
- You can use Redis Desktop Manager (linux, mac, windows)

Model in Redis

- `walls_list` list of wall id's
- `wall_ids` counter with last wall id
- `posts_ids` counter with last post id
- `walls:` list with wall ids
- `walls:WALL_ID` hash with wall data
- `posts:POST_ID` hash with post data
- `walls_posts:WALL_ID` list with posts id's

Model in Redis

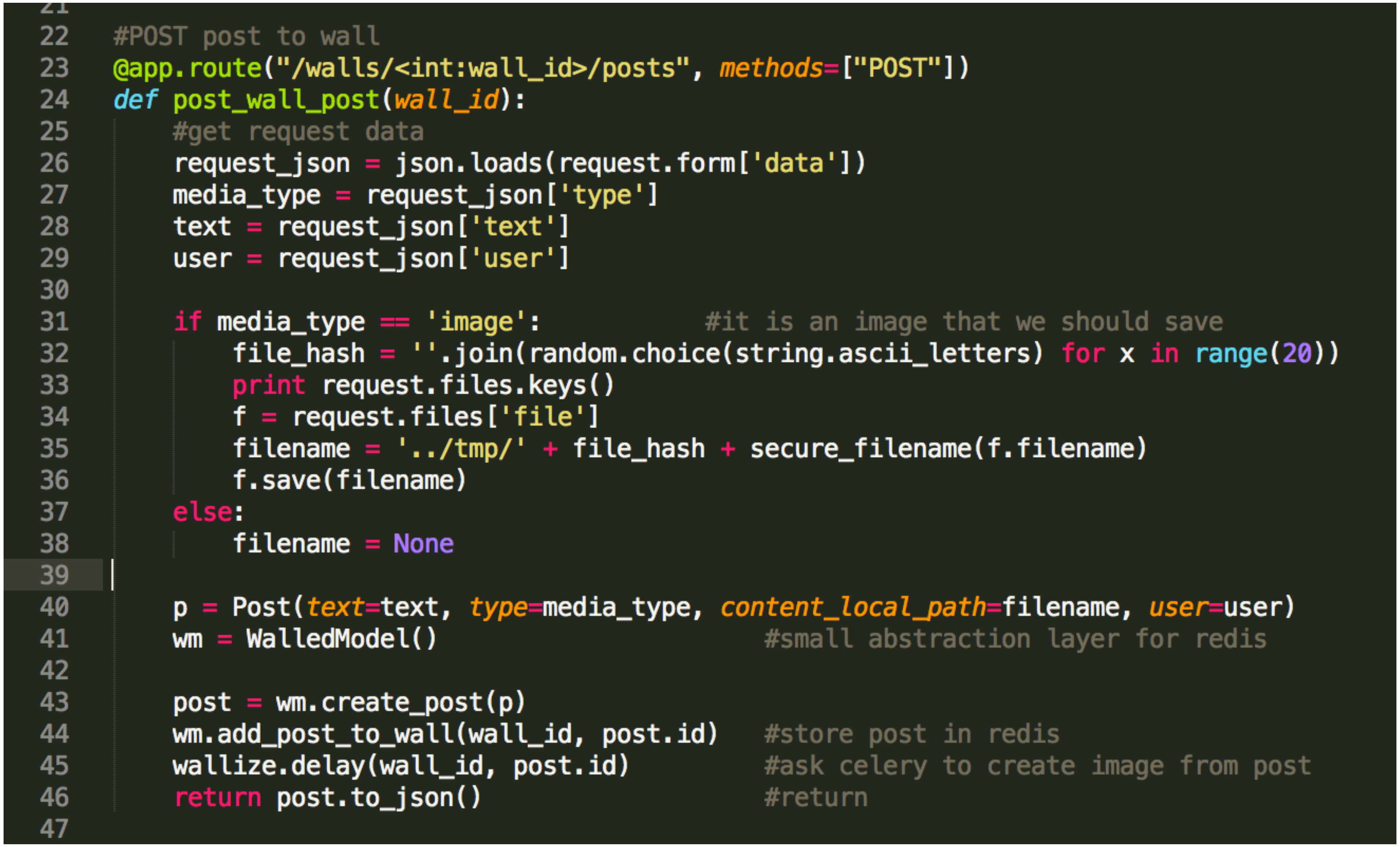


```python

class WalledModel:
    def __init__(self, db=0):
        self.r = redis.StrictRedis(host='localhost', port=6379, db=db)

    def create_post(self, post):
        post_id = self.r.incr('post_ids')
        post.id = post_id
        self.r.rpush('posts_list', post.id)
        self.r.set('posts:'+str(post.id), post.to_json())
        return post

    def get_post(self, post_id):
        post = json.loads(self.r.get('posts:'+str(post_id)))
        p = Post(id = post['id'], type=post['type'], user=post['user'], status=pos
        return p

    def update_post(self, post):
        self.r.set('posts:'+str(post.id), post.to_json())
        return post

    def create_wall(self, wall):
        wall_id = self.r.incr('wall_ids')
        wall.id = wall_id
        self.r.rpush('walls_list', wall.id)
        self.r.set('walls:'+str(wall.id), wall.to_json())
        self.r.set('walls_alias:'+str(wall.alias), wall.id)
        self.r.set('tokens:'+str(wall.token), wall.id)
        self.r.set('tokens_wall:'+str(wall.token), wall.id)
```

API (Flask)

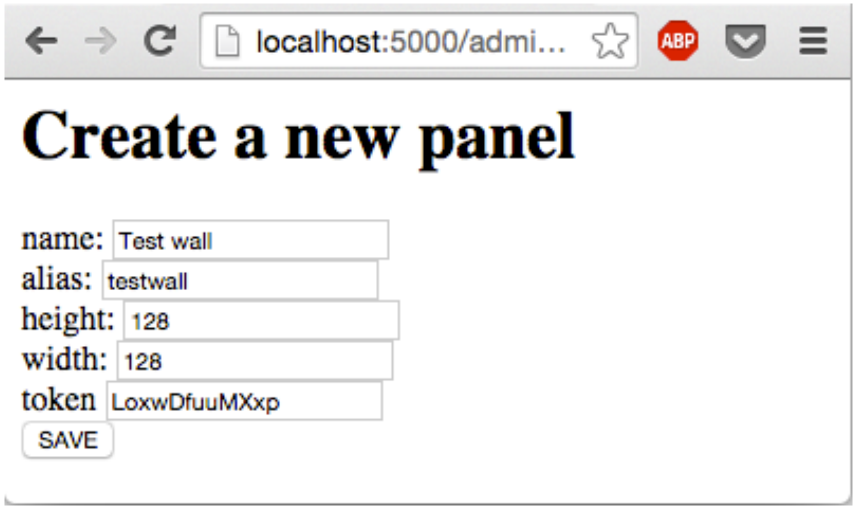
- Its easy and well documented
- Almost everything included (jinja2, sessions, cookies, request, …)
- Very similar to bottle
- Can be deployed using uWSGI or run standalone
- API and Web app all in one!

API (Flask)



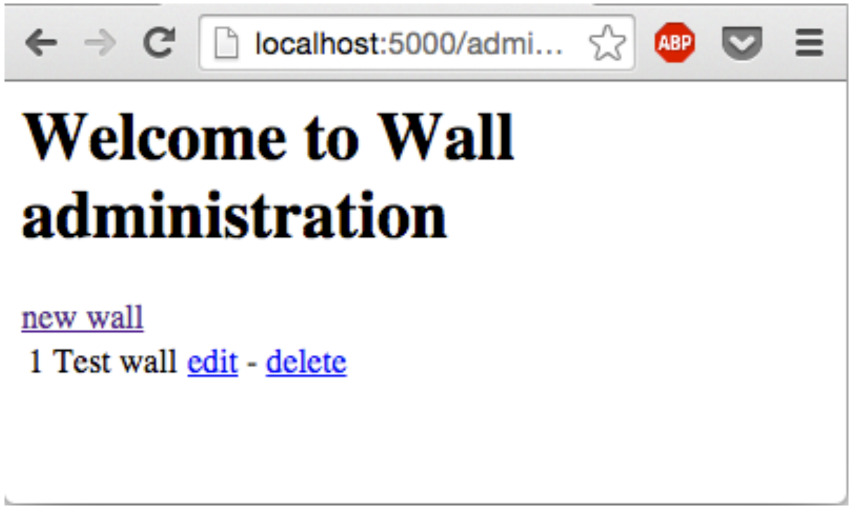
```python
#POST post to wall
@app.route("/walls/<int:wall_id>/posts", methods=["POST"])
def post_wall_post(wall_id):
    #get request data
    request_json = json.loads(request.form['data'])
    media_type = request_json['type']
    text = request_json['text']
    user = request_json['user']

    if media_type == 'image':           #it is an image that we should save
        file_hash = ''.join(random.choice(string.ascii_letters) for x in range(20))
        print request.files.keys()
        f = request.files['file']
        filename = '../tmp/' + file_hash + secure_filename(f.filename)
        f.save(filename)
    else:
        filename = None

    p = Post(text=text, type=media_type, content_local_path=filename, user=user)
    wm = WalledModel()                         #small abstraction layer for redis

    post = wm.create_post(p)
    wm.add_post_to_wall(wall_id, post.id)   #store post in redis
    wallize.delay(wall_id, post.id)         #ask celery to create image from post
    return post.to_json()                   #return
```

API (Flask)


localhost:5000/admi...

Create a new panel

name:	Test wall
alias:	testwall
height:	128
width:	128
token	LoxwDfuuMXxp

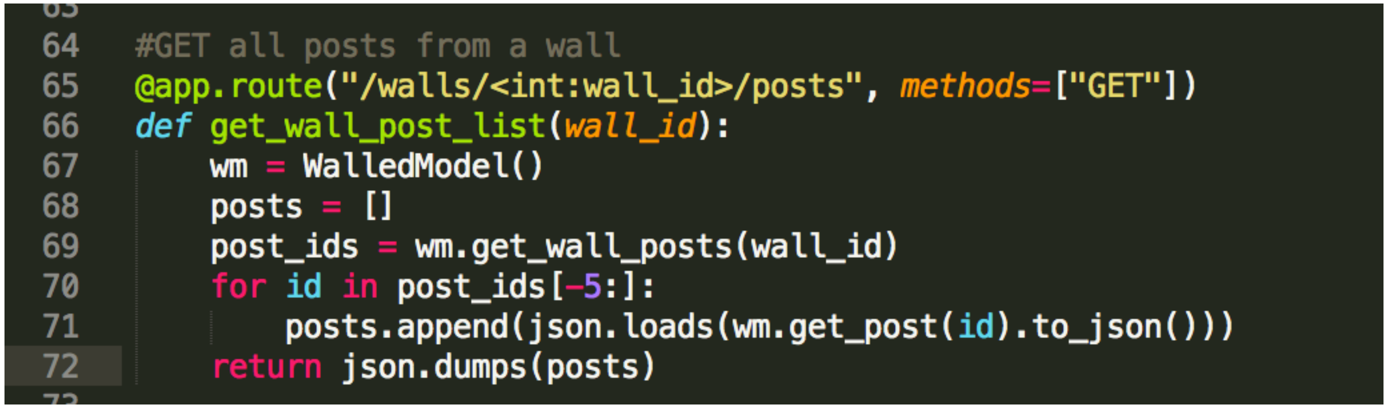

localhost:5000/admi...

Welcome to Wall administration

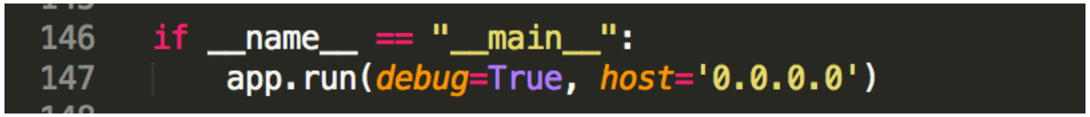
new wall

1 Test wall edit - delete

API (Flask)



```python
#GET all posts from a wall
@app.route("/walls/<int:wall_id>/posts", methods=["GET"])
def get_wall_post_list(wall_id):
    wm = WalledModel()
    posts = []
    post_ids = wm.get_wall_posts(wall_id)
    for id in post_ids[-5:]:
        posts.append(json.loads(wm.get_post(id).to_json()))
    return json.dumps(posts)
```

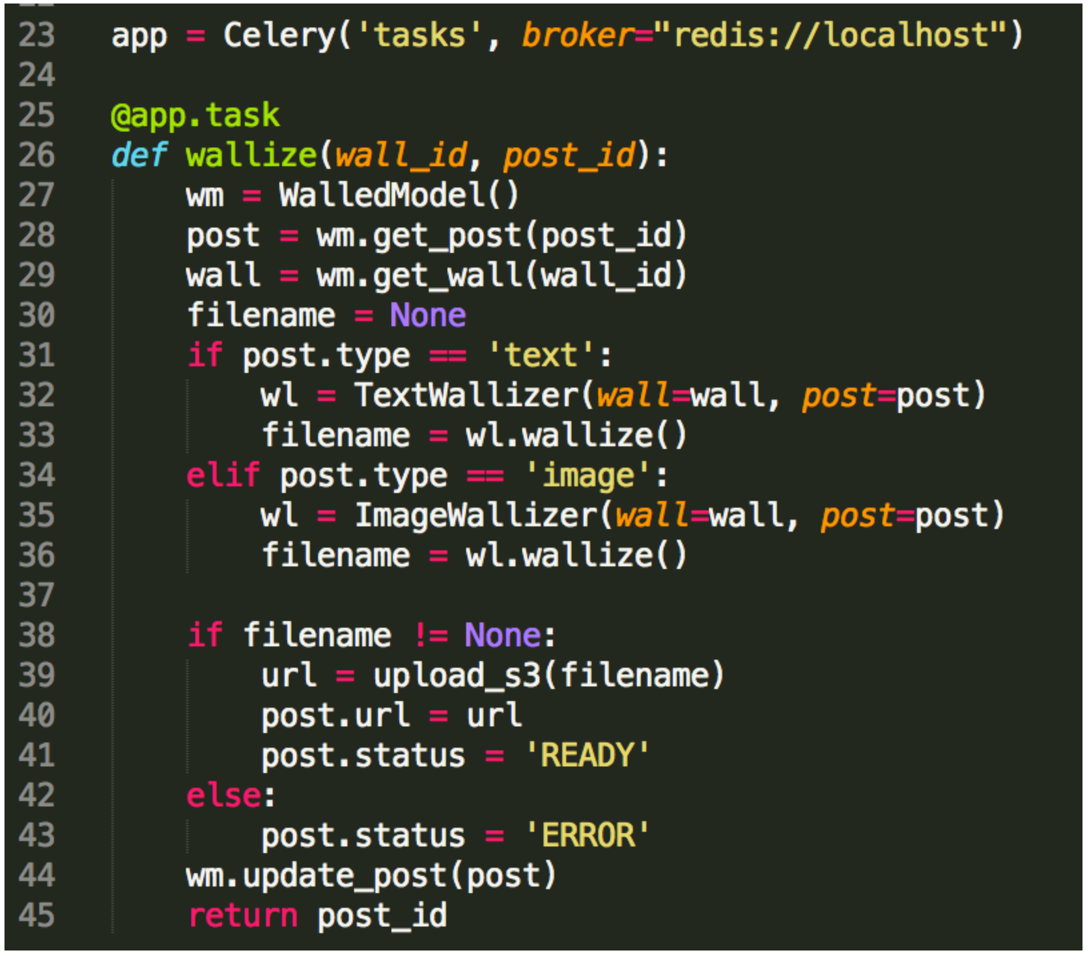


```python
if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0')
```

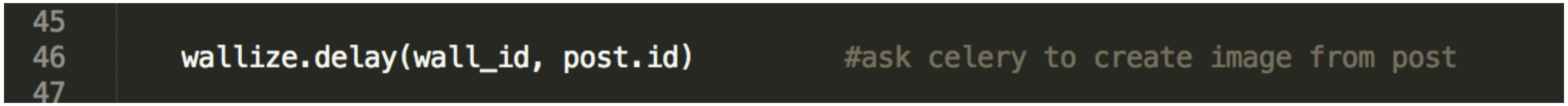
Celery Task Mngt

- Task manager easy to use!
- It can use Redis as a Backend and Broker
- But also MongoDB or RabbitMQ

Celery Task Mngt

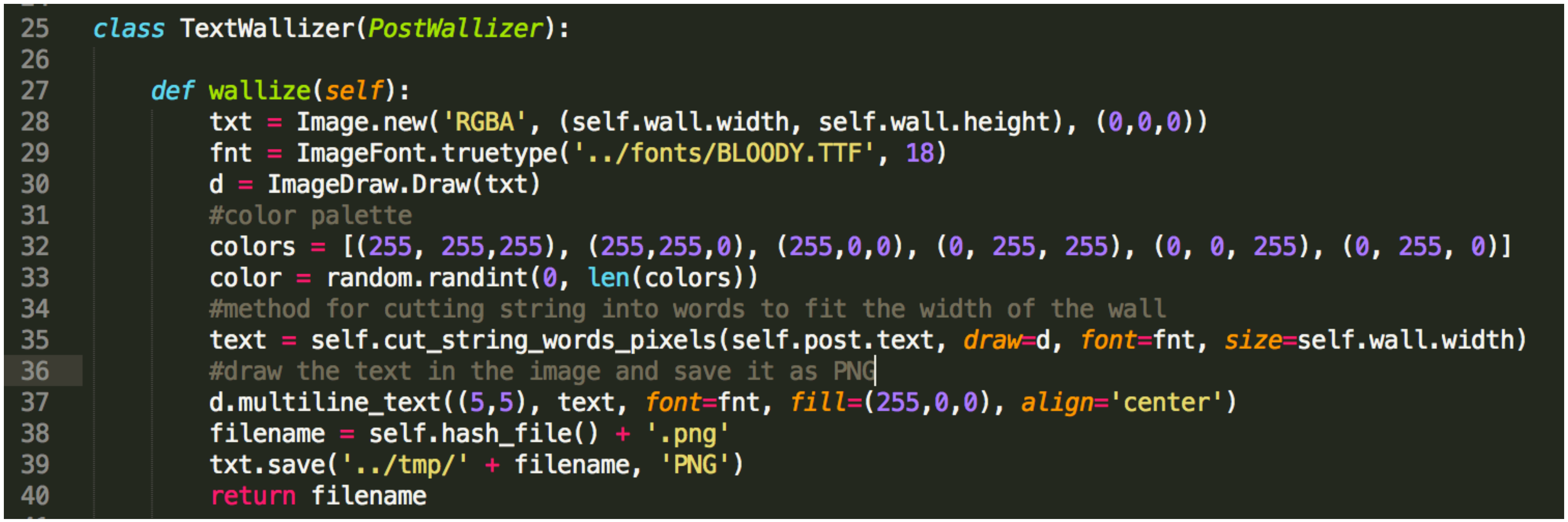


```python
app = Celery('tasks', broker="redis://localhost")

@app.task
def wallize(wall_id, post_id):
    wm = WalledModel()
    post = wm.get_post(post_id)
    wall = wm.get_wall(wall_id)
    filename = None
    if post.type == 'text':
        wl = TextWallizer(wall=wall, post=post)
        filename = wl.wallize()
    elif post.type == 'image':
        wl = ImageWallizer(wall=wall, post=post)
        filename = wl.wallize()

    if filename != None:
        url = upload_s3(filename)
        post.url = url
        post.status = 'READY'
    else:
        post.status = 'ERROR'
    wm.update_post(post)
    return post_id
```

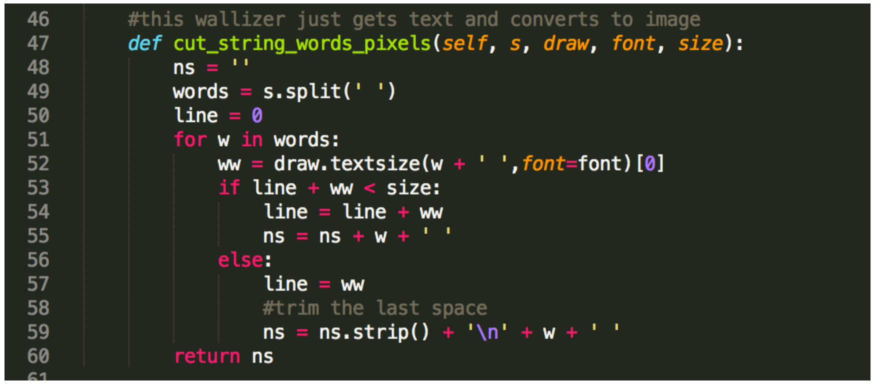


```python
    wallize.delay(wall_id, post.id)          #ask celery to create image from post
```

Convert to image



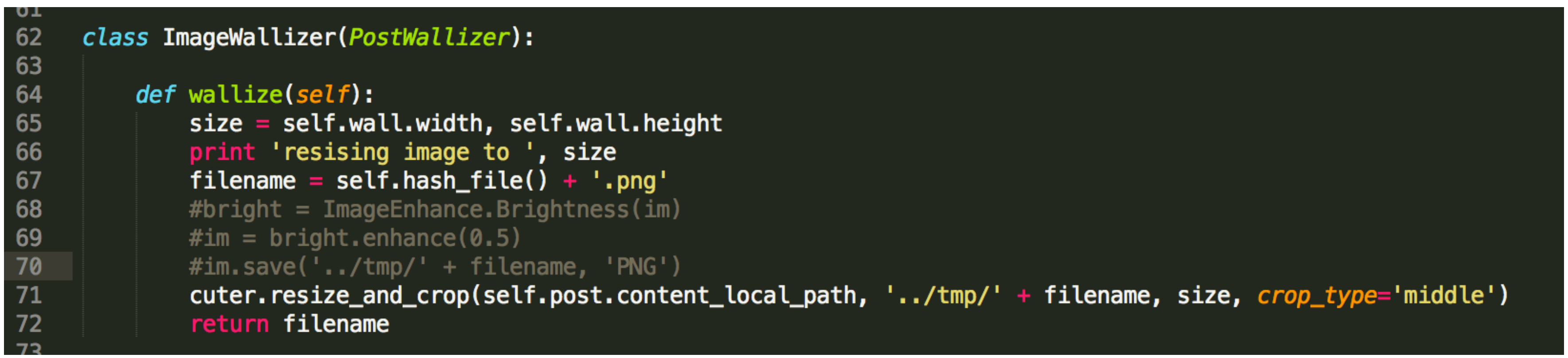
```python
class TextWallizer(PostWallizer):

    def wallize(self):
        txt = Image.new('RGBA', (self.wall.width, self.wall.height), (0,0,0))
        fnt = ImageFont.truetype('../fonts/BLOODY.TTF', 18)
        d = ImageDraw.Draw(txt)
        #color palette
        colors = [(255, 255,255), (255,255,0), (255,0,0), (0, 255, 255), (0, 0, 255), (0, 255, 0)]
        color = random.randint(0, len(colors))
        #method for cutting string into words to fit the width of the wall
        text = self.cut_string_words_pixels(self.post.text, draw=d, font=fnt, size=self.wall.width)
        #draw the text in the image and save it as PNG
        d.multiline_text((5,5), text, font=fnt, fill=(255,0,0), align='center')
        filename = self.hash_file() + '.png'
        txt.save('../tmp/' + filename, 'PNG')
        return filename
```



```python
    #this wallizer just gets text and converts to image
    def cut_string_words_pixels(self, s, draw, font, size):
        ns = ''
        words = s.split(' ')
        line = 0
        for w in words:
            ww = draw.textsize(w + ' ',font=font)[0]
            if line + ww < size:
                line = line + ww
                ns = ns + w + ' '
            else:
                line = ww
                #trim the last space
                ns = ns.strip() + '\n' + w + ' '
        return ns
```

Fonts from: www.dafont.com

Convert to image



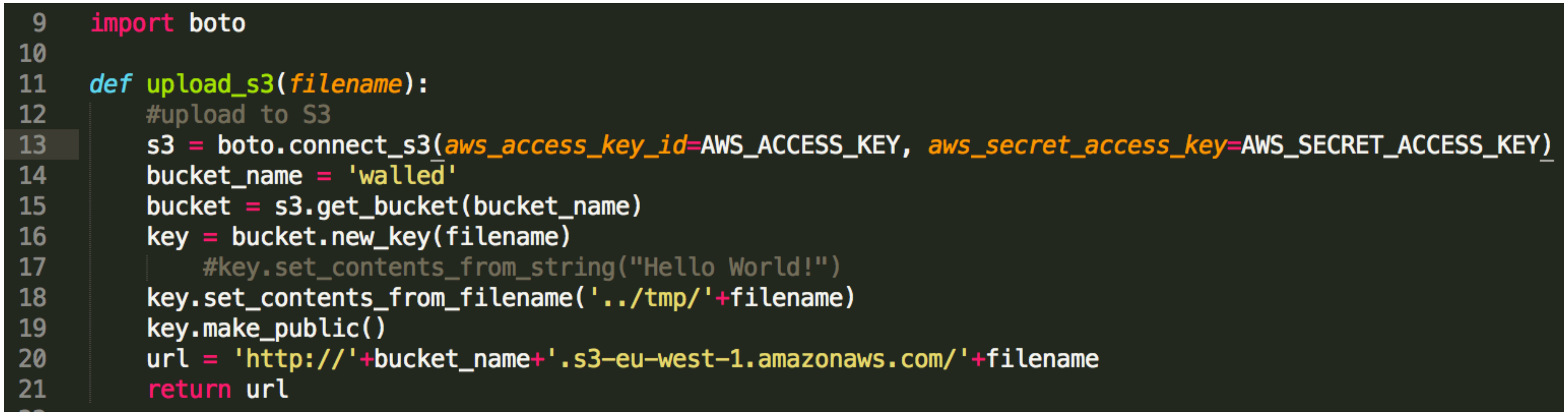
```
class ImageWallizer(PostWallizer):

    def wallize(self):
        size = self.wall.width, self.wall.height
        print 'resising image to ', size
        filename = self.hash_file() + '.png'
        #bright = ImageEnhance.Brightness(im)
        #im = bright.enhance(0.5)
        #im.save('../tmp/' + filename, 'PNG')
        cuter.resize_and_crop(self.post.content_local_path, '../tmp/' + filename, size, crop_type='middle')
        return filename
```

after a couple of hours trying to resize an image,
I decied to use a script found in somewhere: https://gist.github.com/sigilioso/2957026

Amazon S3

- Easy place to store images
- Organized in buckets
- Best library is “boto”
- You should create new keys for your application
- The AWS backend is sometimes confusing

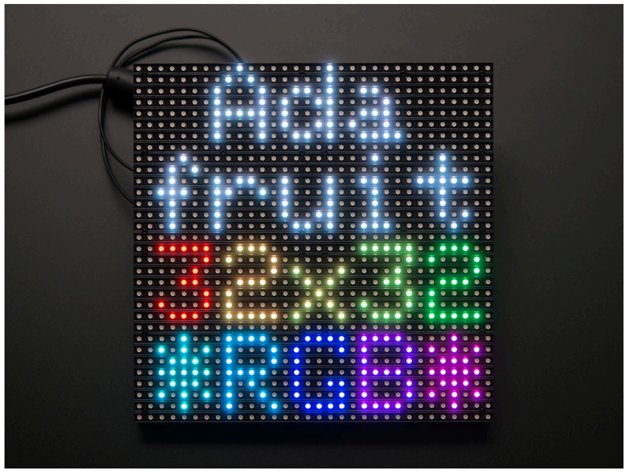
Amazon S3

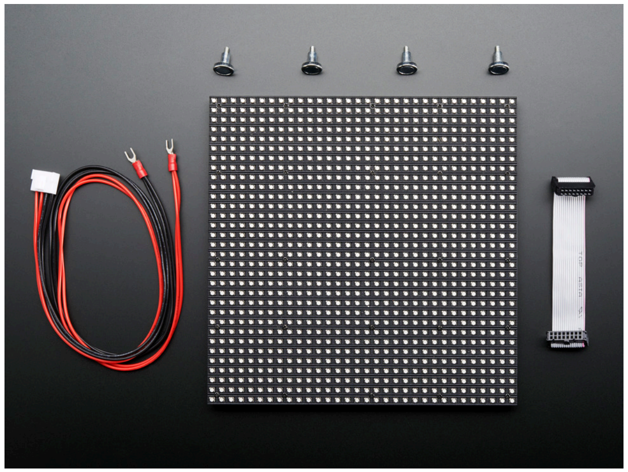


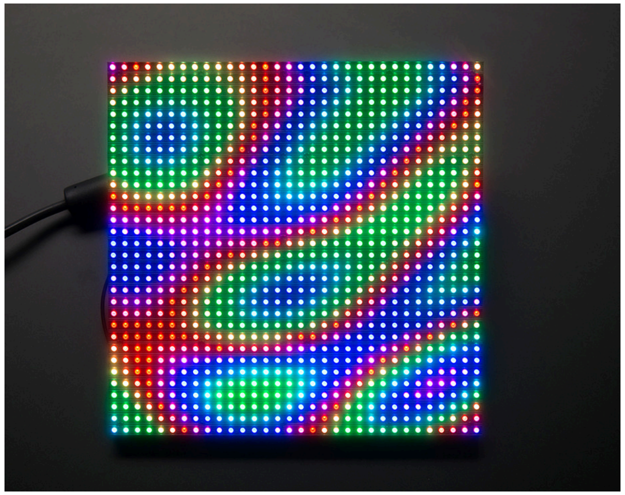
```python
import boto

def upload_s3(filename):
    #upload to S3
    s3 = boto.connect_s3(aws_access_key_id=AWS_ACCESS_KEY, aws_secret_access_key=AWS_SECRET_ACCESS_KEY)
    bucket_name = 'walled'
    bucket = s3.get_bucket(bucket_name)
    key = bucket.new_key(filename)
        #key.set_contents_from_string("Hello World!")
    key.set_contents_from_filename('../tmp/'+filename)
    key.make_public()
    url = 'http://'+bucket_name+'.s3-eu-west-1.amazonaws.com/'+filename
    return url
```

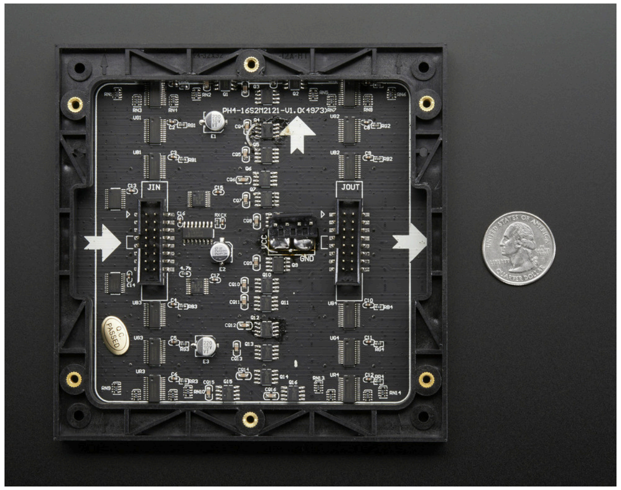
adafruit

LED Matrix



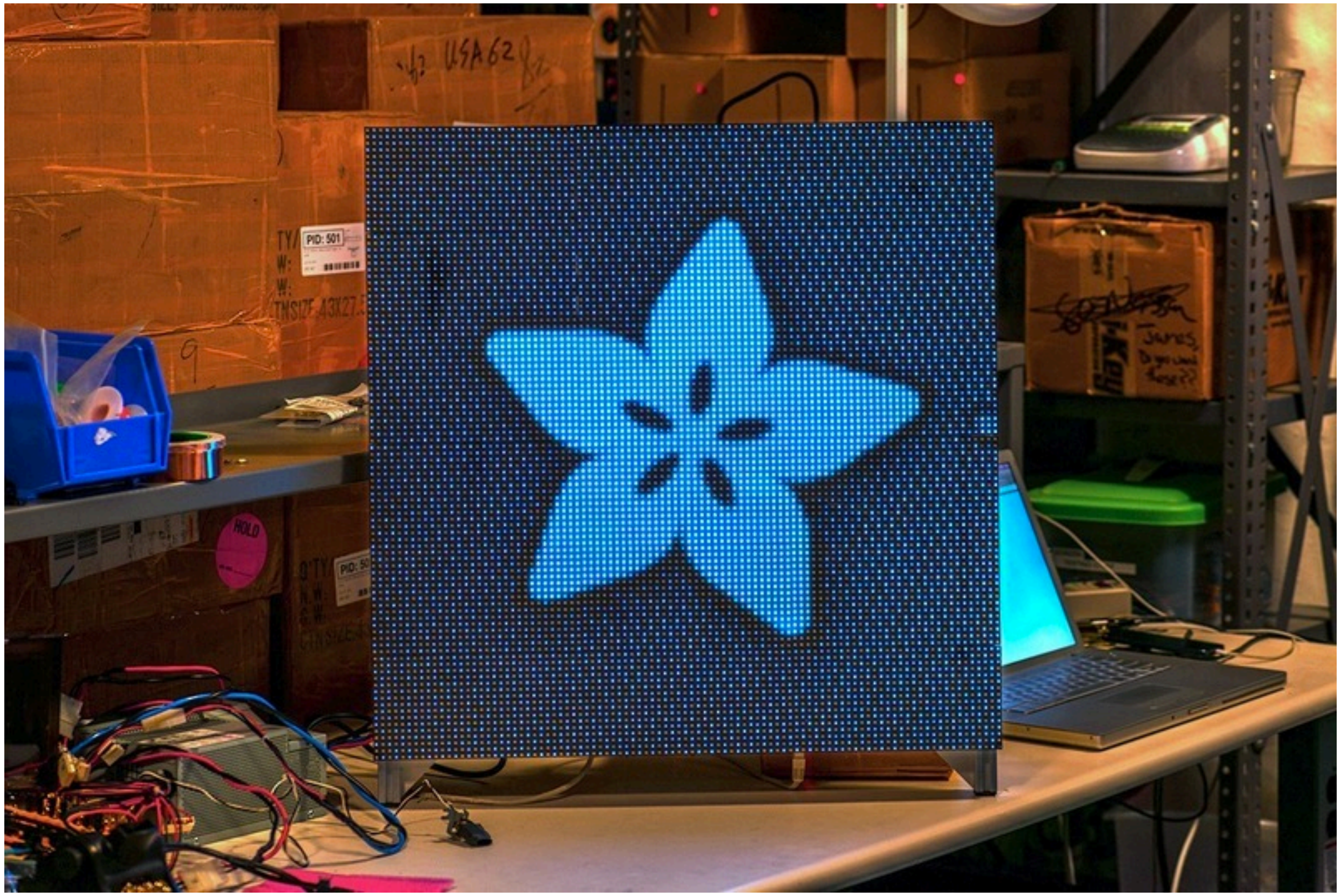






Available at: http://www.adafruit.com/products/1484 (40 USD)

LED Wall

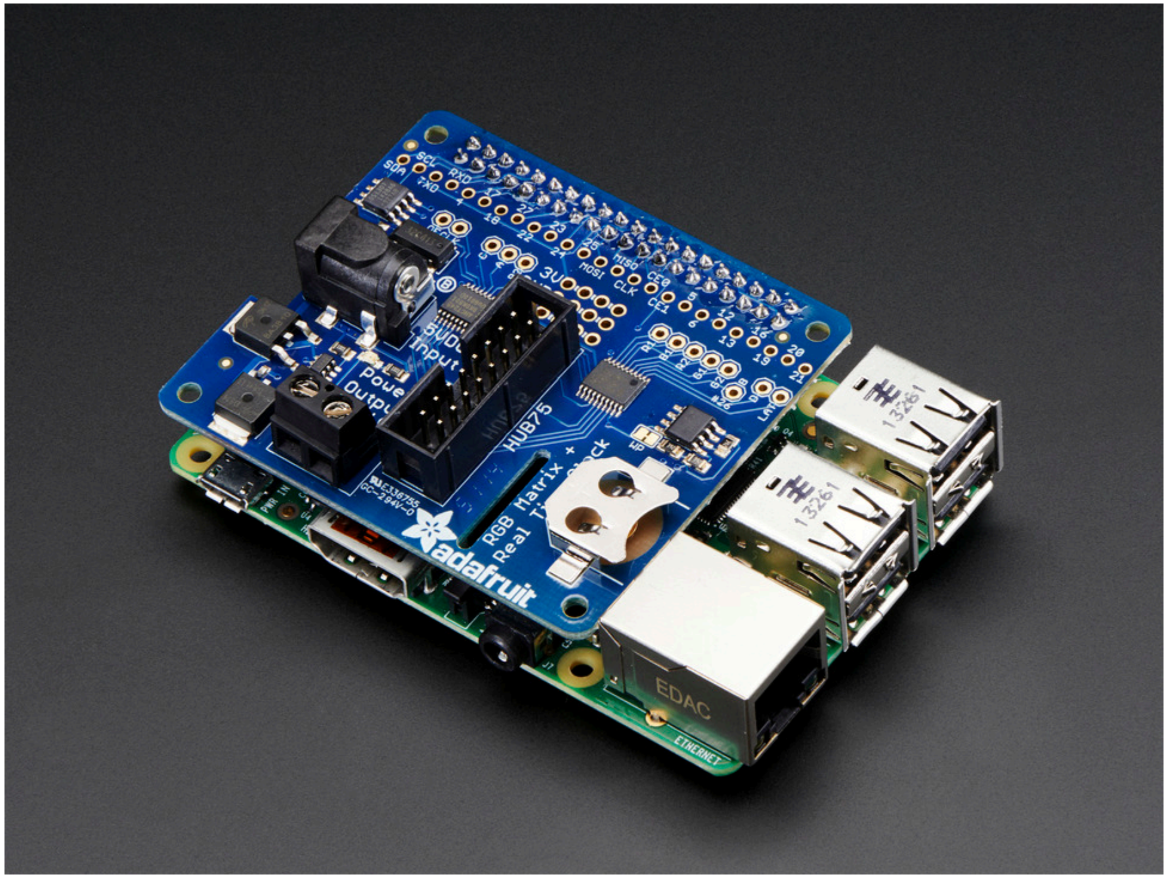


Tutorial here: https://learn.adafruit.com/adafruit-diy-led-video-wall/overview
It is fun to build it, but you can find it already mounted and ready to go!

RaspberryPI

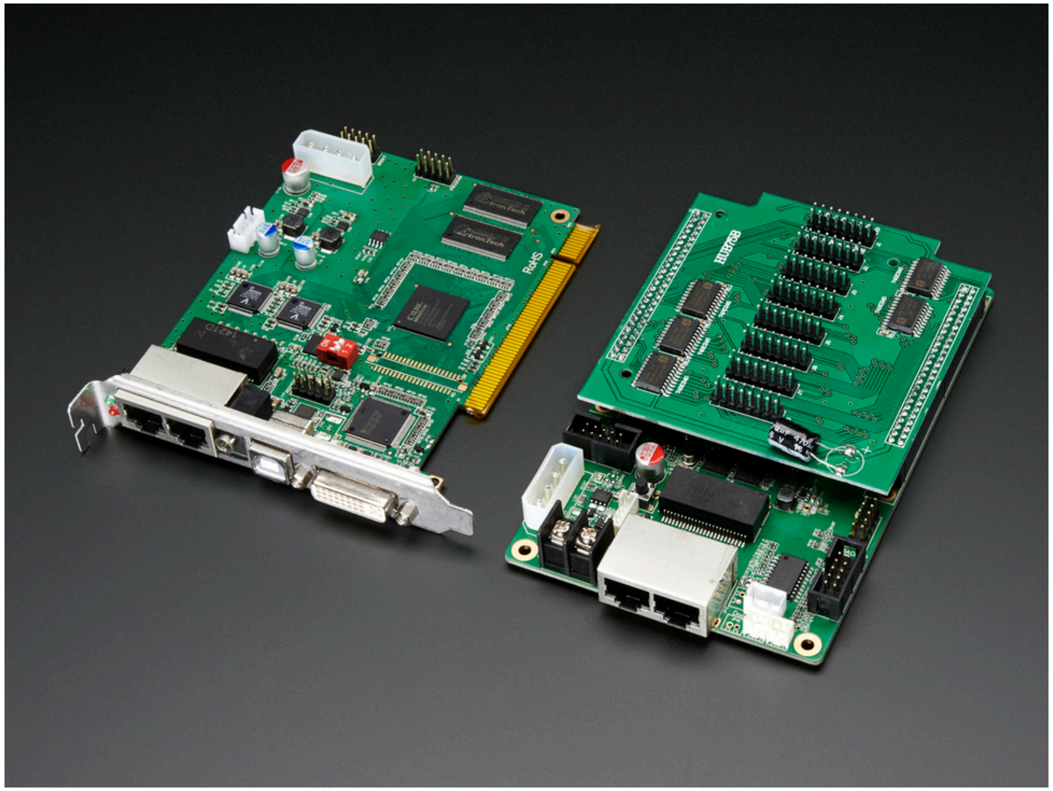
- Last version: Raspberry PI 2 Model B
- Less than 40 USD
- 1 Gb RAM
- To speedup development share the home folder using samba
- If you plan to play with it, get a keyboard and a screen.

RaspberryPI



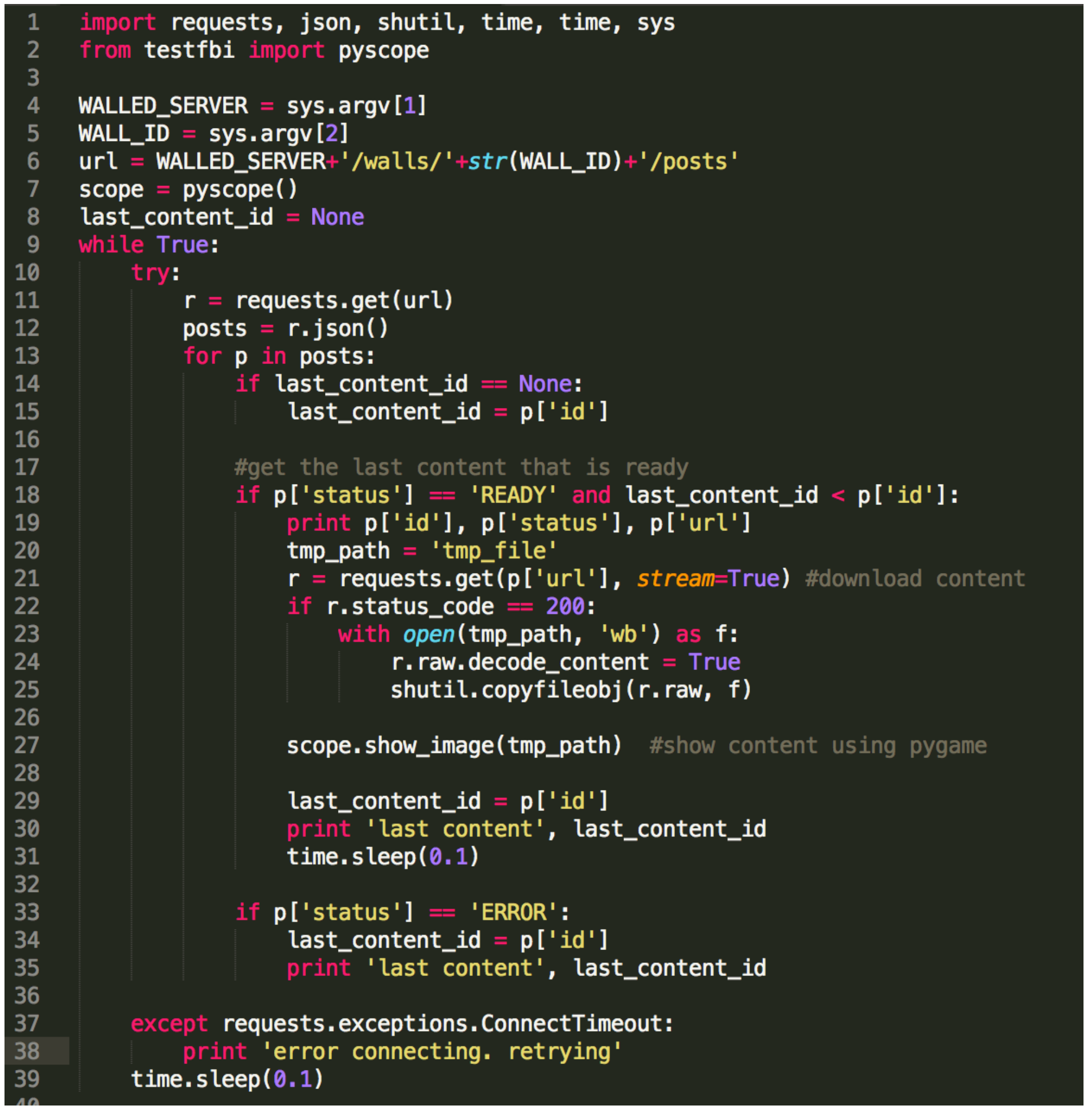
LED GPIO HAT available at Adafruit, it can manage up to 4 32x32 LED Matrix

RaspberryPI



Chinese LED wall controller.
Input is DVI and it maps W x H pixels to LED wall (300 USD)

RaspberryPI

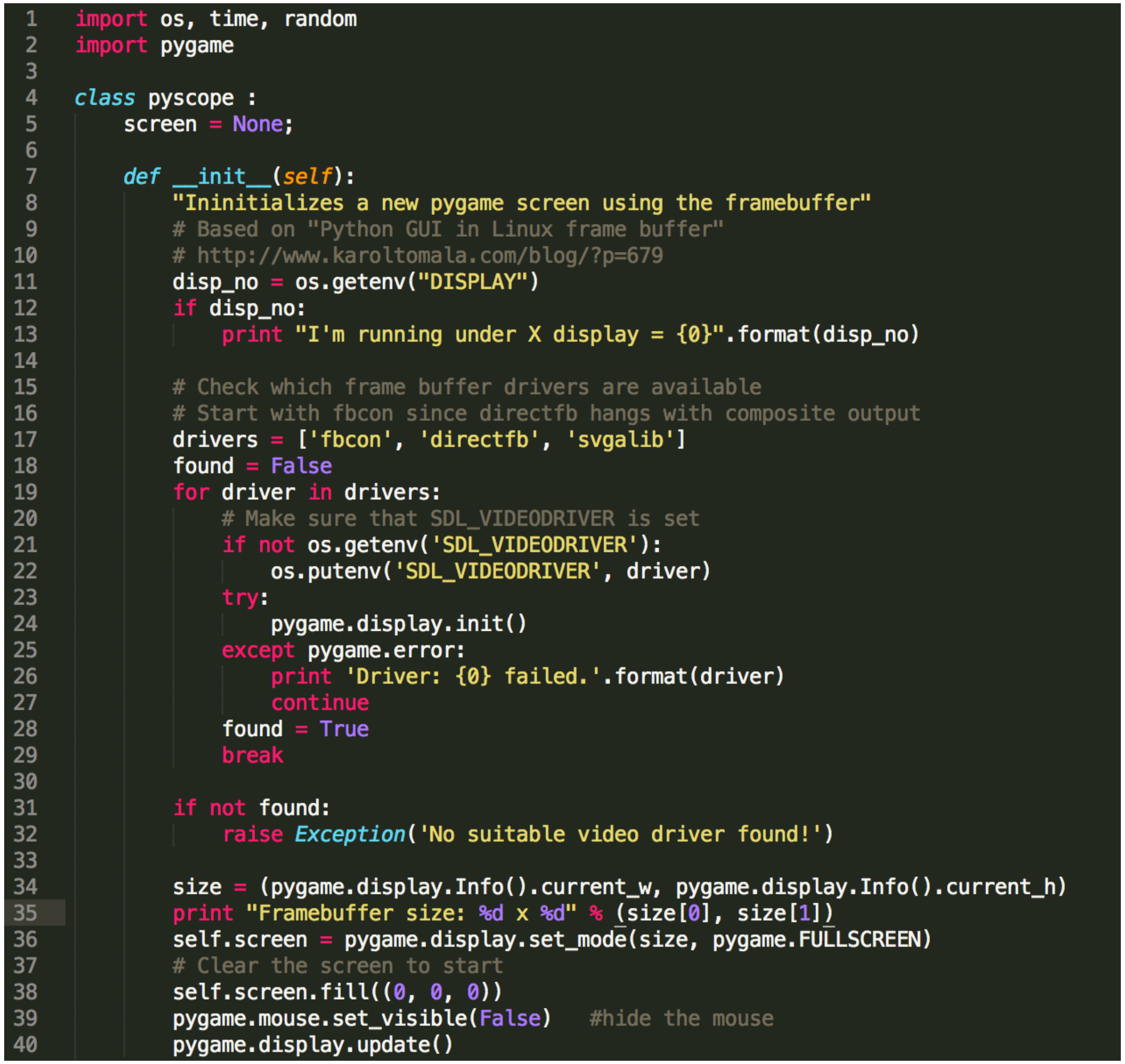


```python
import requests, json, shutil, time, time, sys
from testfbi import pyscope

WALLED_SERVER = sys.argv[1]
WALL_ID = sys.argv[2]
url = WALLED_SERVER+'/walls/'+str(WALL_ID)+'/posts'
scope = pyscope()
last_content_id = None
while True:
    try:
        r = requests.get(url)
        posts = r.json()
        for p in posts:
            if last_content_id == None:
                last_content_id = p['id']

            #get the last content that is ready
            if p['status'] == 'READY' and last_content_id < p['id']:
                print p['id'], p['status'], p['url']
                tmp_path = 'tmp_file'
                r = requests.get(p['url'], stream=True) #download content
                if r.status_code == 200:
                    with open(tmp_path, 'wb') as f:
                        r.raw.decode_content = True
                        shutil.copyfileobj(r.raw, f)

                scope.show_image(tmp_path)  #show content using pygame

                last_content_id = p['id']
                print 'last content', last_content_id
                time.sleep(0.1)

            if p['status'] == 'ERROR':
                last_content_id = p['id']
                print 'last content', last_content_id

    except requests.exceptions.ConnectTimeout:
        print 'error connecting. retrying'
    time.sleep(0.1)
```

Framebuffer

- PyGame enables us to access the framebuffer directly
- We can show images or sprites
- Programs to modify framebuffer must run with sudo

Framebuffer



```python
import os, time, random
import pygame

class pyscope :
    screen = None;

    def __init__(self):
        "Ininitializes a new pygame screen using the framebuffer"
        # Based on "Python GUI in Linux frame buffer"
        # http://www.karoltomala.com/blog/?p=679
        disp_no = os.getenv("DISPLAY")
        if disp_no:
            print "I'm running under X display = {0}".format(disp_no)

        # Check which frame buffer drivers are available
        # Start with fbcon since directfb hangs with composite output
        drivers = ['fbcon', 'directfb', 'svgalib']
        found = False
        for driver in drivers:
            # Make sure that SDL_VIDEODRIVER is set
            if not os.getenv('SDL_VIDEODRIVER'):
                os.putenv('SDL_VIDEODRIVER', driver)
            try:
                pygame.display.init()
            except pygame.error:
                print 'Driver: {0} failed.'.format(driver)
                continue
            found = True
            break

        if not found:
            raise Exception('No suitable video driver found!')

        size = (pygame.display.Info().current_w, pygame.display.Info().current_h)
        print "Framebuffer size: %d x %d" % (size[0], size[1])
        self.screen = pygame.display.set_mode(size, pygame.FULLSCREEN)
        # Clear the screen to start
        self.screen.fill((0, 0, 0))
        pygame.mouse.set_visible(False)   #hide the mouse
        pygame.display.update()
```

The Result

Thank you!

code:

xavier.orduna@gmail.com