PSDM

PACKAGING SORTING DEVELOPMENT MACHINE

Simone Brigante
Francesco Casciaro
Alessandro Chiotti
Giacomo Deleo

Website URL :https://ami-2017.github.io/PSDM

The idea

Plastic Recycling process

VS

Plastic Analyzer

Other kind of plastic

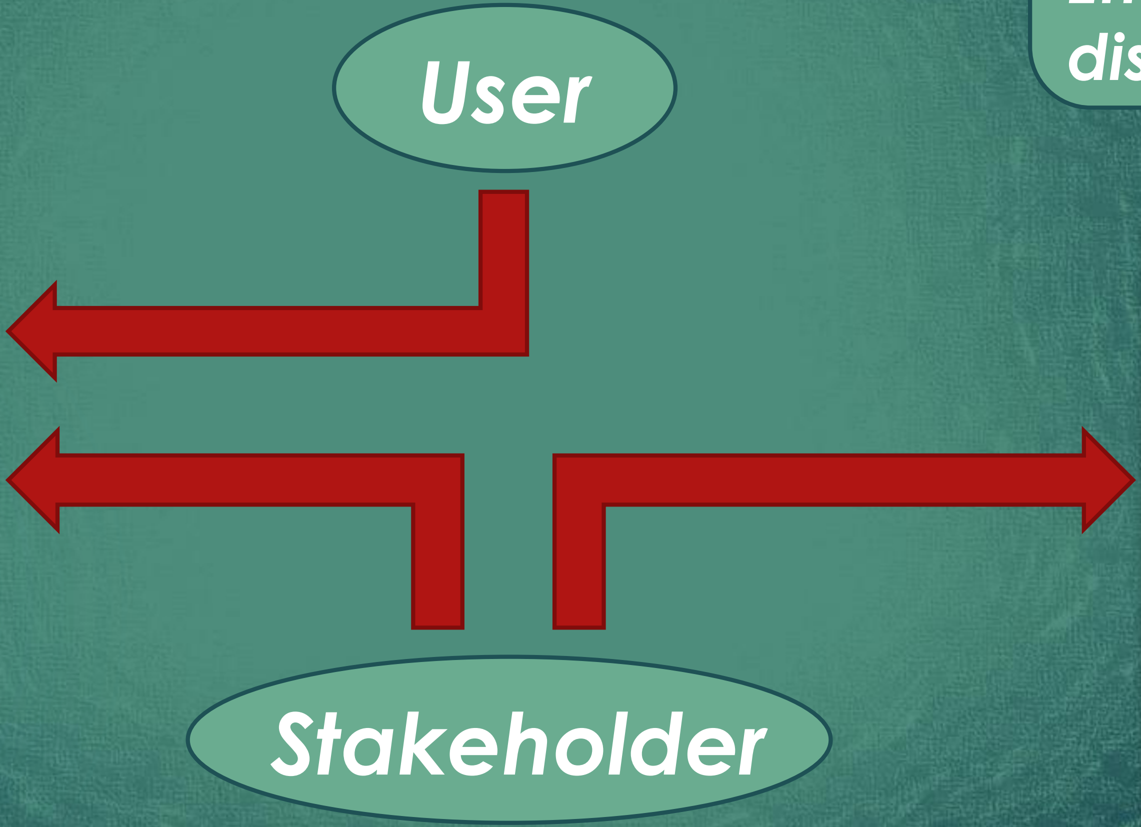
Packaging (PET, PVC, PE-HD)

Plastic Recycling process	*Plastic Analyzer*
○ ***Expensive***	○ ***Cheaper***
○ ***Large Scale***	○ ***User level***
○ ***Slow process***	○ ***Quick process***

Different users

Employee of the shop

Employees of the waste disposal company



AmI Main Steps

Sensing

- *distinguish plastic*
- *detecting the volume of trash*

Reasoning

- *define the most suitable trash.*

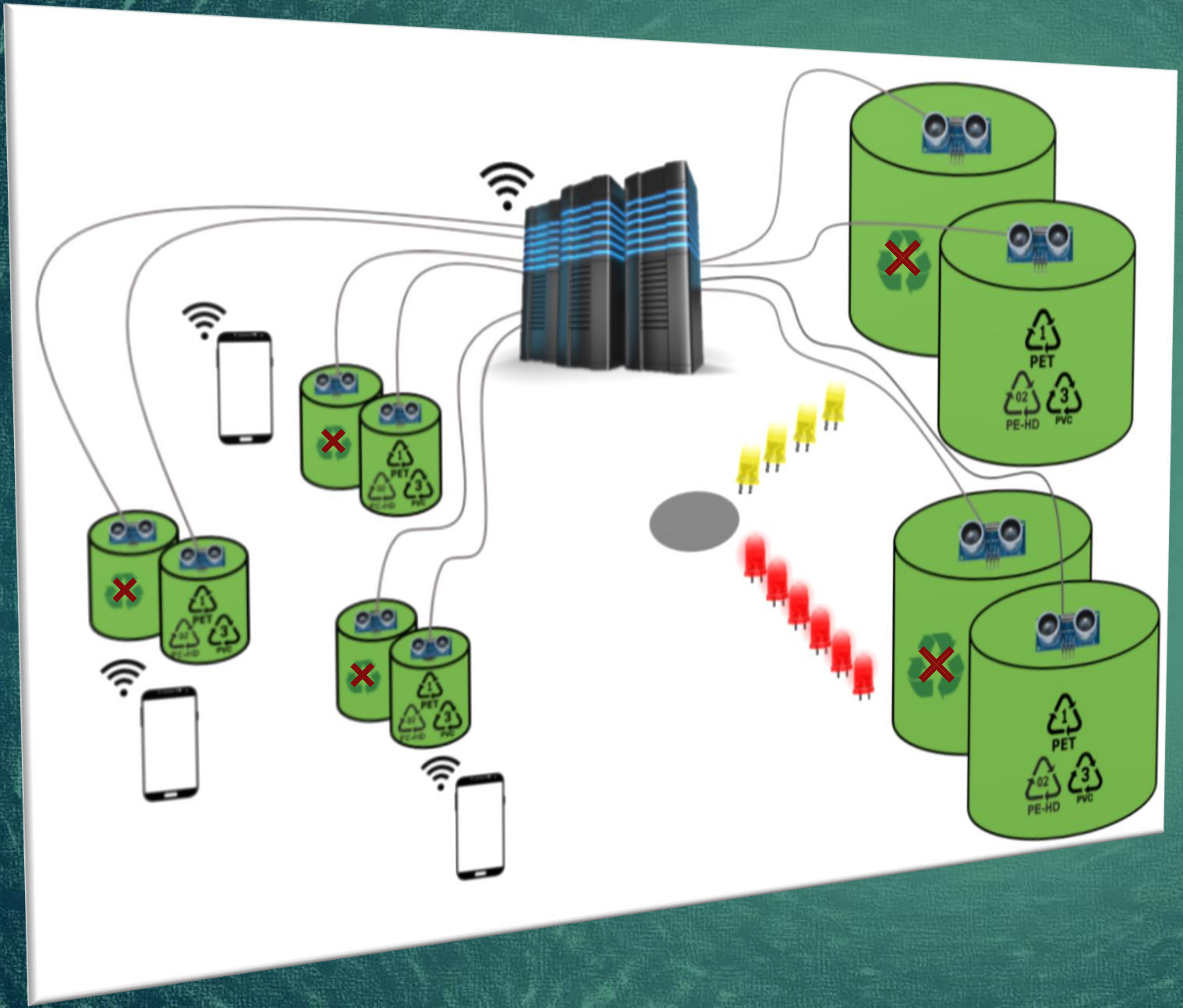
Acting

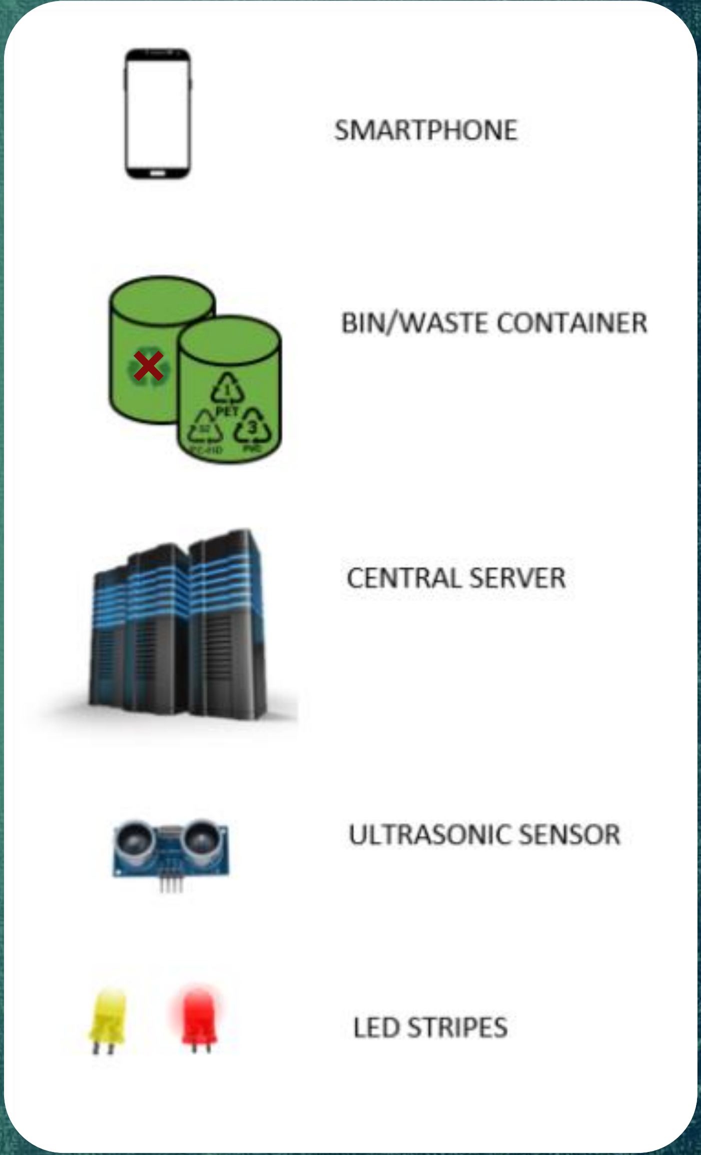
- *light path to the correct waste container*
- *Call the waste transport company when a container is full*

Interacting

- *At the end of the day or if a bin is full it will remind the user to throw out the trash*

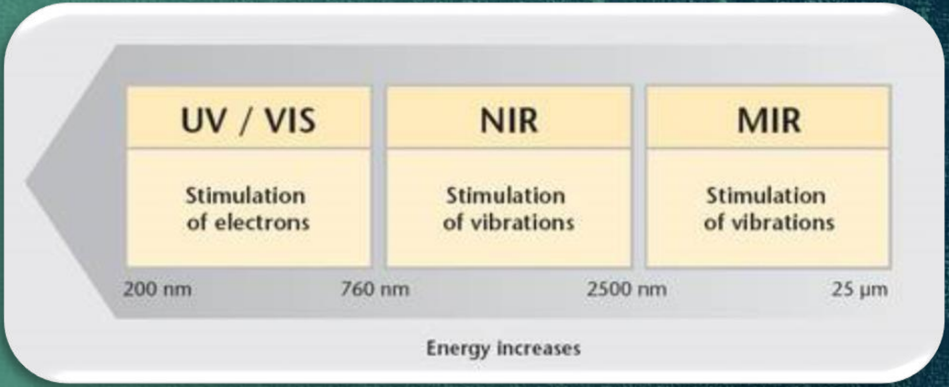
System Architecture

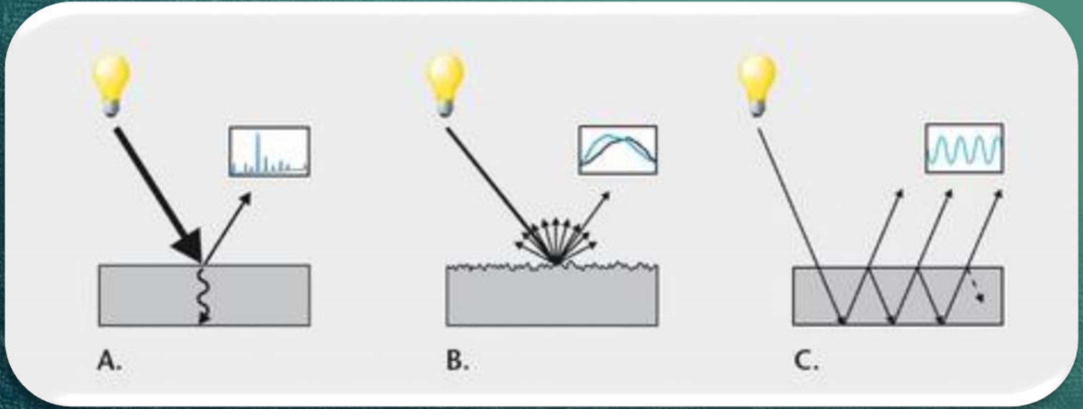




Plastic Analyzer

Near-infrared spectroscopy (NIRS) is a spectroscopic method that uses the near-infrared region of the electromagnetic spectrum (from about 700 nm to 2500 nm). NIR radiation has less energy/photon but does excite molecular vibrations.



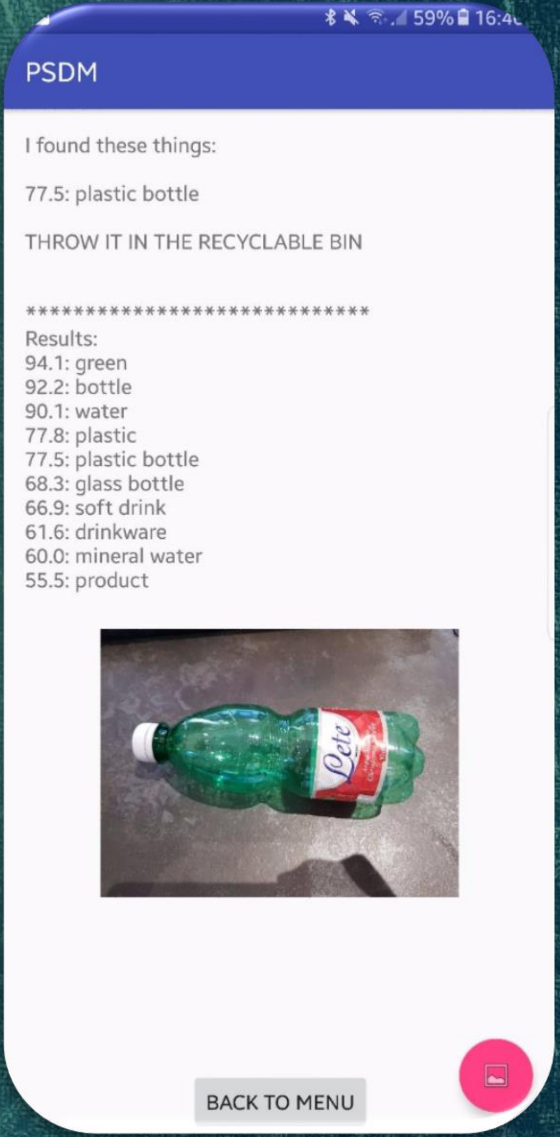


A. Absorbance/Reflection
B. Surface effects
C. Interface effects

Google Cloud Vision

The plastic analyzer was impossible to retrieve

So we integrate the Google Vision Api in our Android App



I found these things:

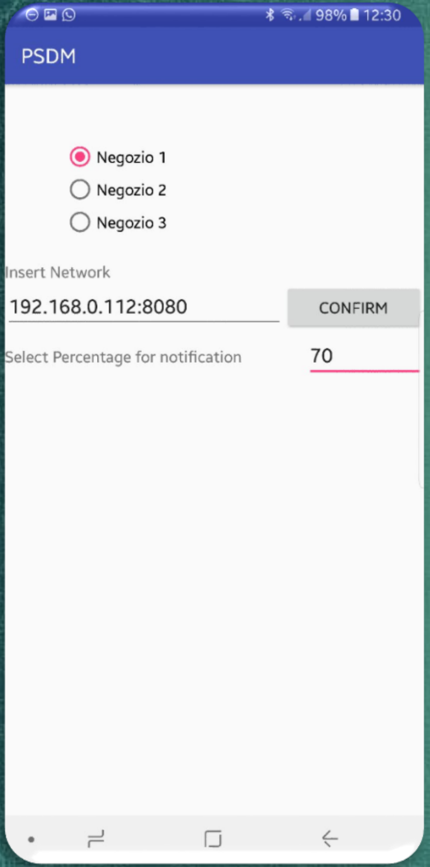
77.5: plastic bottle

THROW IT IN THE RECYCLABLE BIN

****

Results:
94.1: green
92.2: bottle
90.1: water
77.8: plastic
77.5: plastic bottle
68.3: glass bottle
66.9: soft drink
61.6: drinkware
60.0: mineral water
55.5: product

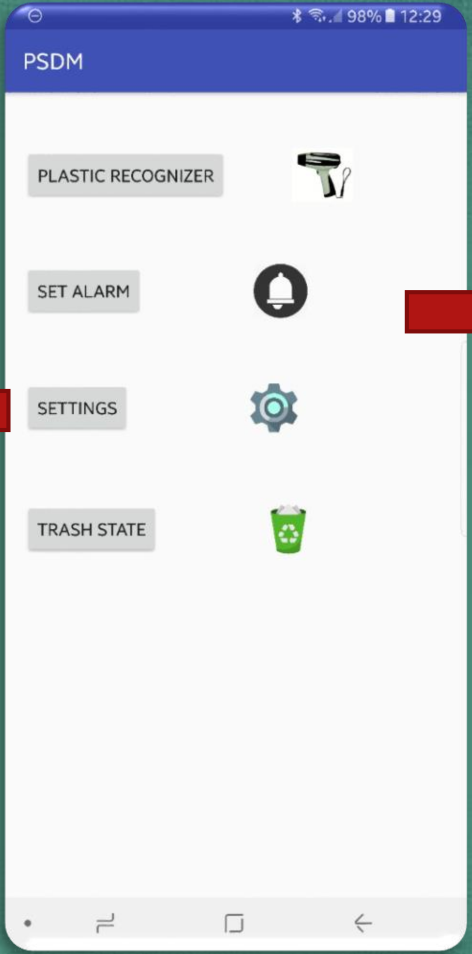
Android application

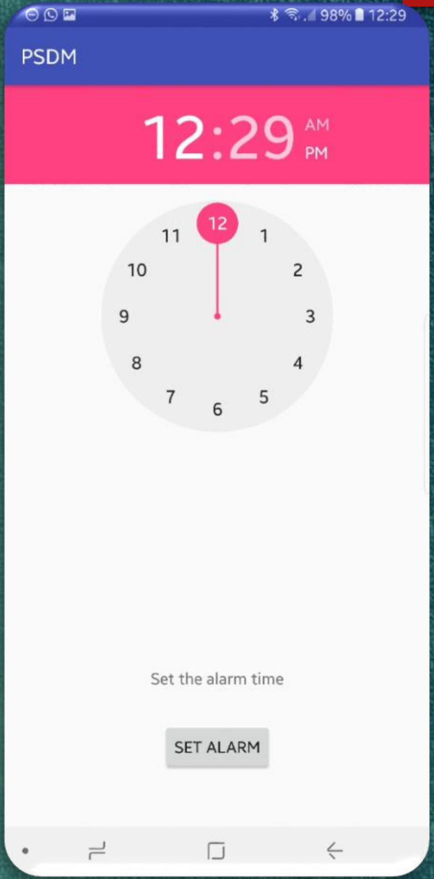

PSDM

- [✓ Negozio 1]
- Negozio 2
- Negozio 3

Insert Network
192.168.0.112:8080
CONFIRM

Select Percentage for notification 70

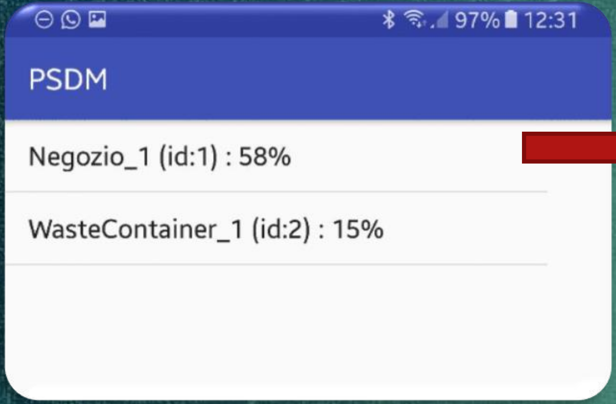




- *PutExtra*
- *GetStringExtra*

- *AlarmManager*
- *BroadcastReceiver*
- *NotificationManager*

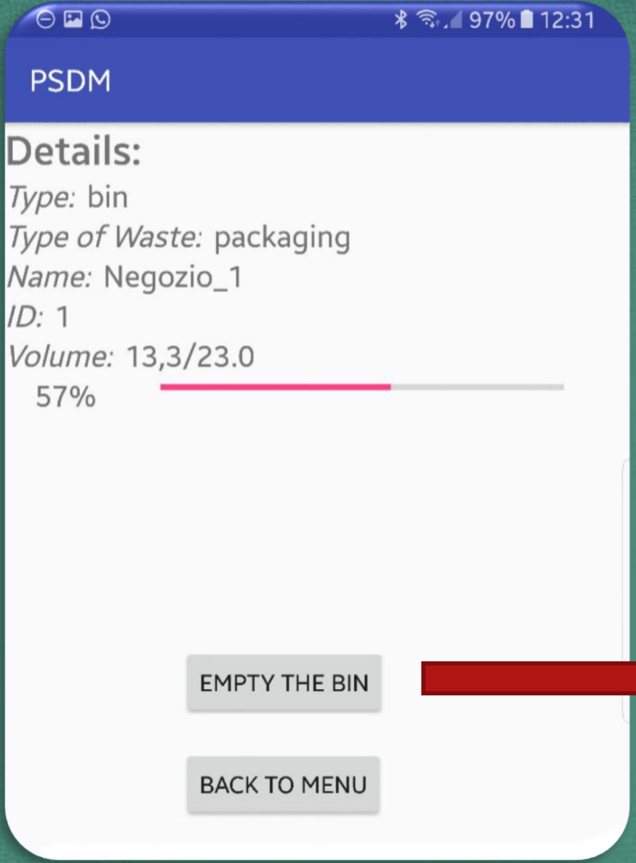
Android application



PSDM

Negozio_1 (id:1) : 58%

WasteContainer_1 (id:2) : 15%



PSDM

Details:

Type: bin

Type of Waste: packaging

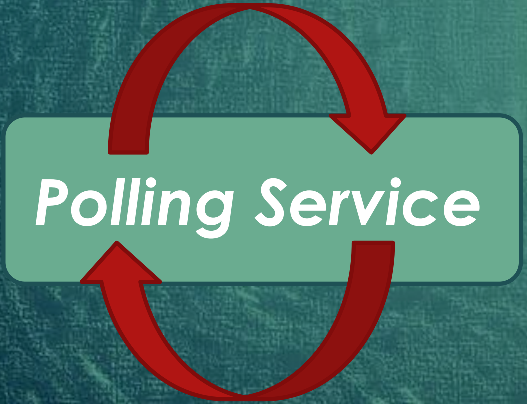
Name: Negozio_1

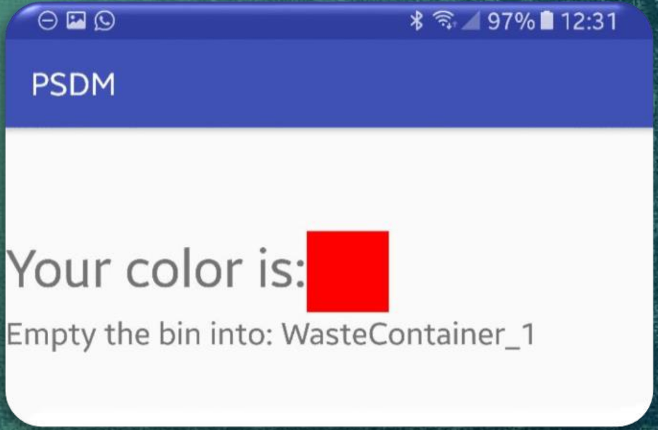
ID: 1

Volume: 13,3/23.0

57%

AsyncTask classes



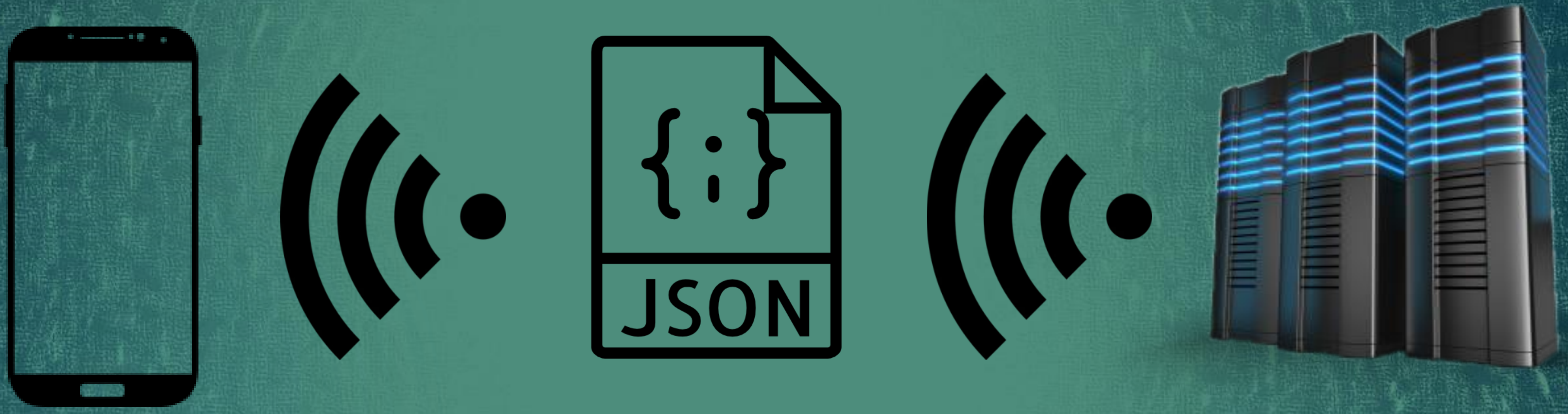


PSDM

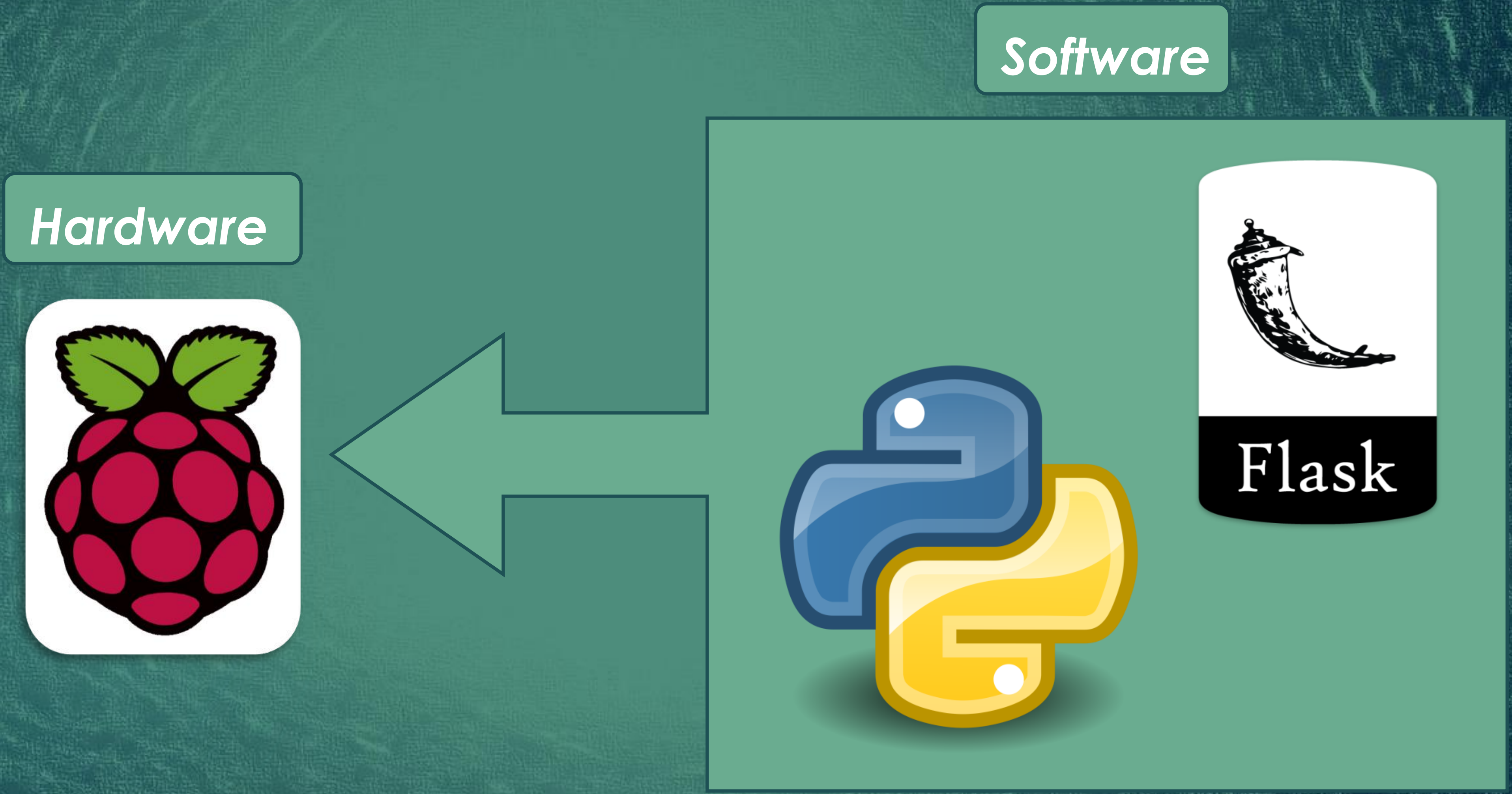
Your color is: 🟥

Empty the bin into: WasteContainer_1

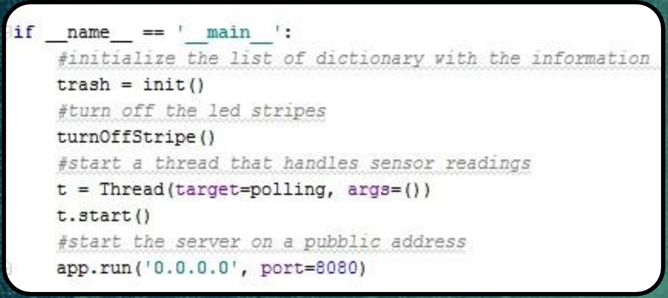
Data Exchange Format



Server Side

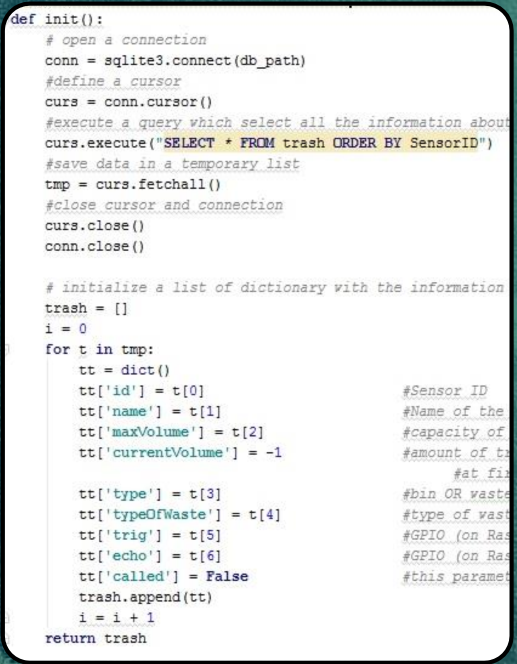


Main



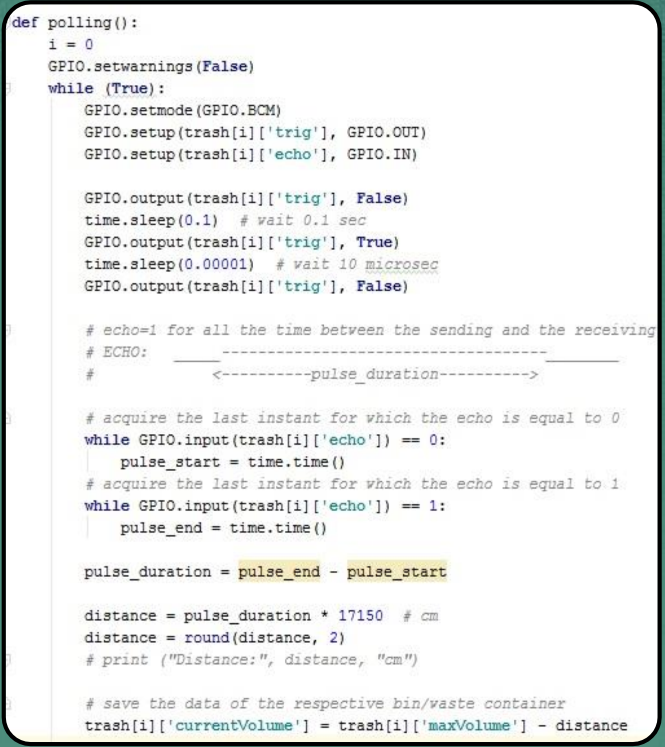
```
if __name__ == '__main__':
    #initialize the list of dictionary with the information
    trash = init()
    #turn off the led stripes
    turnOffStripe()
    #start a thread that handles sensor readings
    t = Thread(target=polling, args=())
    t.start()
    #start the server on a pubblic address
    app.run('0.0.0.0', port=8080)
```

Database reading

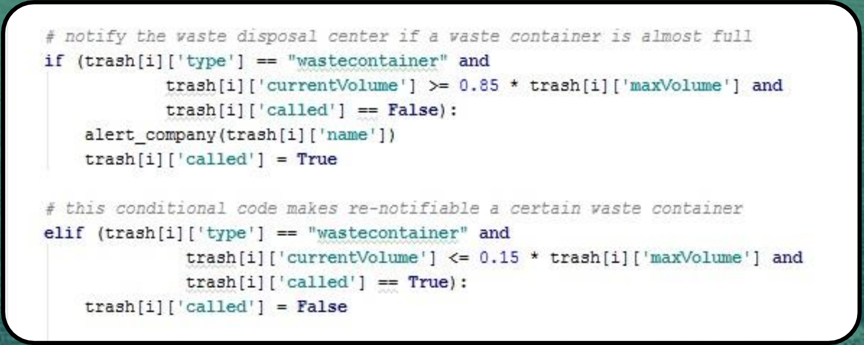


```
def init():
    # open a connection
    conn = sqlite3.connect(db_path)
    #define a cursor
    curs = conn.cursor()
    #execute a query which select all the information about
    curs.execute("SELECT * FROM trash ORDER BY SensorID")
    #save data in a temporary list
    tmp = curs.fetchall()
    #close cursor and connection
    curs.close()
    conn.close()

    # initialize a list of dictionary with the information
    trash = []
    i = 0
    for t in tmp:
        tt = dict()
        tt['id'] = t[0]                         #Sensor ID
        tt['name'] = t[1]                       #Name of the
        tt['maxVolume'] = t[2]                  #capacity of
        tt['currentVolume'] = -1                #amount of tr
                                                      #at fi
        tt['type'] = t[3]                       #bin OR waste
        tt['typeOfWaste'] = t[4]                #type of wast
        tt['trig'] = t[5]                       #GPIO (on Ras
        tt['echo'] = t[6]                       #GPIO (on Ras
        tt['called'] = False                    #this paramet
        trash.append(tt)
        i = i + 1
    return trash
```

Interaction with the sensors

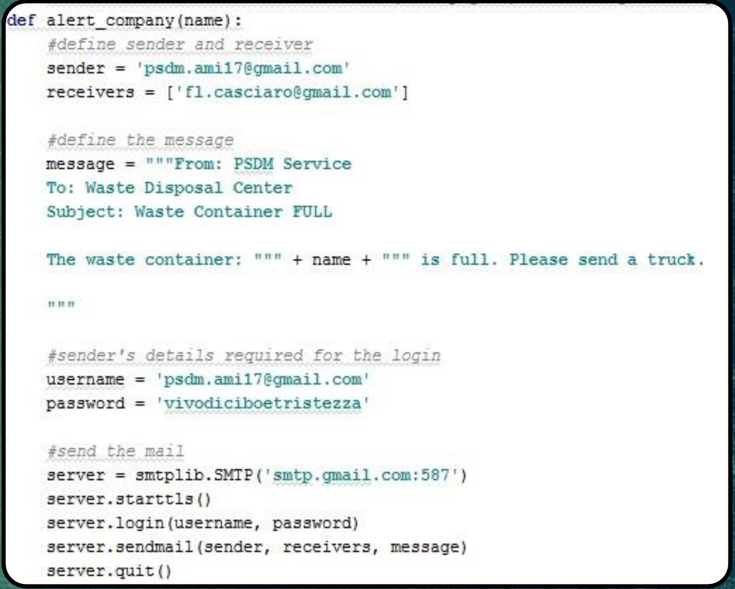


```
def polling():
    i = 0
    GPIO.setwarnings(False)
    while (True):
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(trash[i]['trig'], GPIO.OUT)
        GPIO.setup(trash[i]['echo'], GPIO.IN)

        GPIO.output(trash[i]['trig'], False)
        time.sleep(0.1)  # wait 0.1 sec
        GPIO.output(trash[i]['trig'], True)
        time.sleep(0.00001)  # wait 10 microsec
        GPIO.output(trash[i]['trig'], False)

        # echo=1 for all the time between the sending and the receiving
        # ECHO:   _____------------------------------------_________
        #              <----------pulse_duration---------->

        # acquire the last instant for which the echo is equal to 0
        while GPIO.input(trash[i]['echo']) == 0:
            pulse_start = time.time()
        # acquire the last instant for which the echo is equal to 1
        while GPIO.input(trash[i]['echo']) == 1:
            pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start

        distance = pulse_duration * 17150  # cm
        distance = round(distance, 2)
        # print ("Distance:", distance, "cm")

        # save the data of the respective bin/waste container
        trash[i]['currentVolume'] = trash[i]['maxVolume'] - distance
```



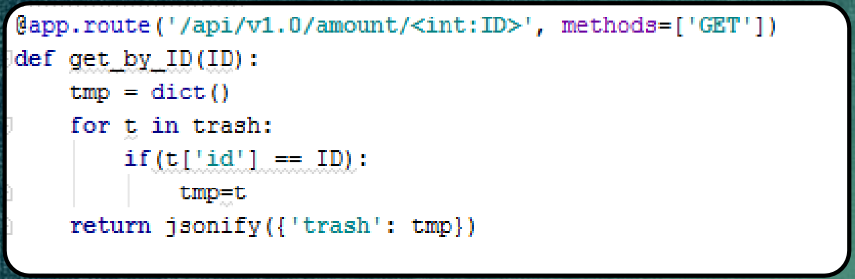
```
# notify the waste disposal center if a waste container is almost full
if (trash[i]['type'] == "wastecontainer" and
            trash[i]['currentVolume'] >= 0.85 * trash[i]['maxVolume'] and
            trash[i]['called'] == False):
    alert_company(trash[i]['name'])
    trash[i]['called'] = True

# this conditional code makes re-notifiable a certain waste container
elif (trash[i]['type'] == "wastecontainer" and
            trash[i]['currentVolume'] <= 0.15 * trash[i]['maxVolume'] and
            trash[i]['called'] == True):
    trash[i]['called'] = False
```

Notify the waste disposal company

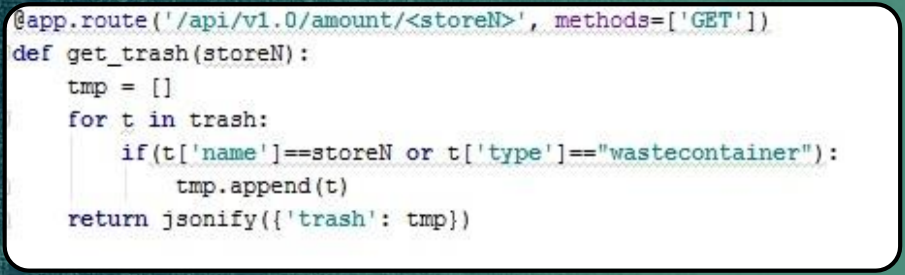


```
def alert_company(name):
    #define sender and receiver
    sender = 'psdm.ami17@gmail.com'
    receivers = ['fl.casciaro@gmail.com']

    #define the message
    message = """From: PSDM Service
    To: Waste Disposal Center
    Subject: Waste Container FULL

    The waste container: """ + name + """ is full. Please send a truck.

    """

    #sender's details required for the login
    username = 'psdm.ami17@gmail.com'
    password = 'vivodiciboetristezza'

    #send the mail
    server = smtplib.SMTP('smtp.gmail.com:587')
    server.starttls()
    server.login(username, password)
    server.sendmail(sender, receivers, message)
    server.quit()
```

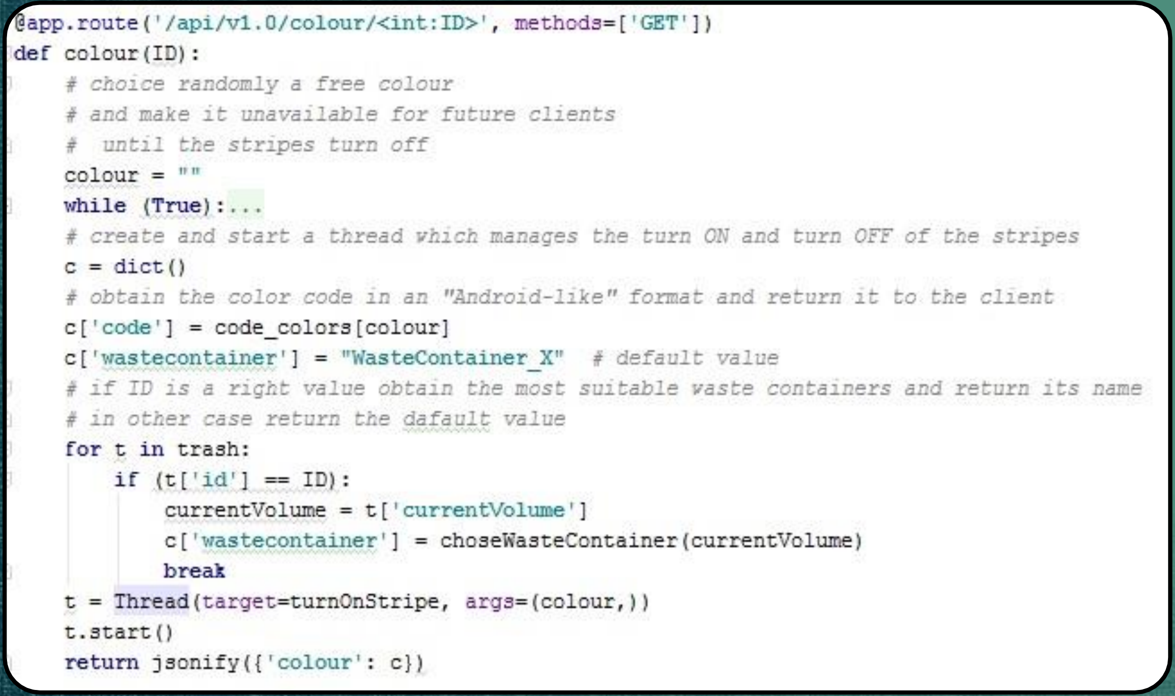
Rest Functions



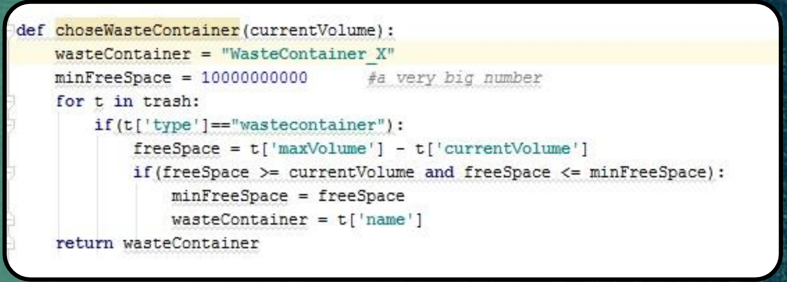
```
@app.route('/api/v1.0/amount/<int:ID>', methods=['GET'])
def get_by_ID(ID):
    tmp = dict()
    for t in trash:
        if(t['id'] == ID):
            tmp=t
    return jsonify({'trash': tmp})
```



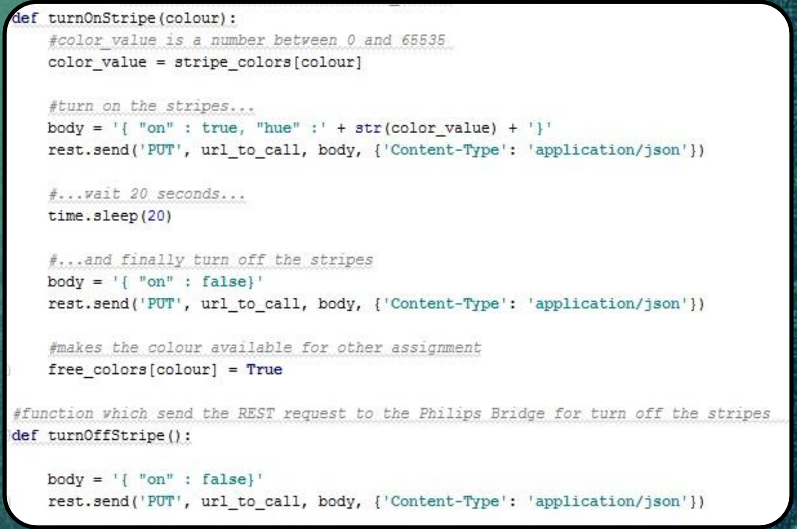
```
@app.route('/api/v1.0/amount/<storeN>', methods=['GET'])
def get_trash(storeN):
    tmp = []
    for t in trash:
        if(t['name']==storeN or t['type']=="wastecontainer"):
            tmp.append(t)
    return jsonify({'trash': tmp})
```



```
@app.route('/api/v1.0/colour/<int:ID>', methods=['GET'])
def colour(ID):
    # choice randomly a free colour
    # and make it unavailable for future clients
    #  until the stripes turn off
    colour = ""
    while (True):...
    # create and start a thread which manages the turn ON and turn OFF of the stripes
    c = dict()
    # obtain the color code in an "Android-like" format and return it to the client
    c['code'] = code_colors[colour]
    c['wastecontainer'] = "WasteContainer_X"  # default value
    # if ID is a right value obtain the most suitable waste containers and return its name
    # in other case return the dafault value
    for t in trash:
        if (t['id'] == ID):
            currentVolume = t['currentVolume']
            c['wastecontainer'] = choseWasteContainer(currentVolume)
            break
    t = Thread(target=turnOnStripe, args=(colour,))
    t.start()
    return jsonify({'colour': c})
```



```
def choseWasteContainer(currentVolume):
    wasteContainer = "WasteContainer_X"
    minFreeSpace = 10000000000      #a very big number
    for t in trash:
        if(t['type']=="wastecontainer"):
            freeSpace = t['maxVolume'] - t['currentVolume']
            if(freeSpace >= currentVolume and freeSpace <= minFreeSpace):
                minFreeSpace = freeSpace
                wasteContainer = t['name']
    return wasteContainer
```



```
def turnOnStripe(colour):
    #color_value is a number between 0 and 65535
    color_value = stripe_colors[colour]

    #turn on the stripes...
    body = '{ "on" : true, "hue" :' + str(color_value) + '}'
    rest.send('PUT', url_to_call, body, {'Content-Type': 'application/json'})

    #...wait 20 seconds...
    time.sleep(20)

    #...and finally turn off the stripes
    body = '{ "on" : false}'
    rest.send('PUT', url_to_call, body, {'Content-Type': 'application/json'})

    #makes the colour available for other assignment
    free_colors[colour] = True

#function which send the REST request to the Philips Bridge for turn off the stripes
def turnOffStripe():

    body = '{ "on" : false}'
    rest.send('PUT', url_to_call, body, {'Content-Type': 'application/json'})
```

Thank's for your attention

Simone Brigante
Francesco Casciaro
Alessandro Chiotti
Giacomo Deleo

Website URL :https://ami-2017.github.io/PSDM