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)
<table>
<thead>
<tr>
<th>Plastic Recycling process</th>
<th>Plastic Analyzer</th>
</tr>
</thead>
<tbody>
<tr>
<td>o Expensive</td>
<td>o Cheaper</td>
</tr>
<tr>
<td>o Large Scale</td>
<td>o User level</td>
</tr>
<tr>
<td>o Slow process</td>
<td>o Quick process</td>
</tr>
</tbody>
</table>
Different users

Employee of the shop

User

Employee of the waste disposal company

Stakeholder
AmI Main Steps

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

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

**Reasoning**
- define the most suitable trash.

**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
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.
The plastic analyzer was impossible to retrieve. So we integrate the Google Vision API in our Android App.
Android application

- PutExtra
-GetStringExtra
- AlarmManager
- BroadcastReceiver
- NotificationManager
Android application

Polling Service

AsyncTask classes

PSDM

Negozio_1 (id:1) : 58%
WasteContainer_1 (id:2) : 15%

Details:
Type: bin
Type of Waste: packaging
Name: Negozio_1
ID: 1
Volume: 13.3/23.0
57%

EMPTY THE BIN
BACK TO MENU

PSDM

Your color is: Red
Empty the bin into: WasteContainer_1
Data Exchange Format
Main

if _name_ == 'main':
    # initialize the list of dictionary with the information
    trash = init()
    # turn off the led strip
    turnOffStrip()
    # start a thread that handles sensor readings
    t = Thread(target=polling, args=[])
    t.start()
    # start the server on a public address
    app.run('0.0.0.0', port=8000)

Interaction with the sensors

def polling:
    while True:
        GPIO.wait_for_edge(GPIO.IN, GPIO.RISING)
        if GPIO.input(TRASH_GPIO):  # 0.000s time.sleep(0.0001)  # wait 0.0001s
            Notify the waste disposal company
            _notify the waste disposal company

Notify the waste disposal company

def _notify_the_waste_disposal_company(name):
    # define sender and receiver
    sender = 'your_email@gmail.com'
    receivers = ['receiver1@gmail.com', 'receiver2@gmail.com']
    # define the message
    message = 'Your message here'
    # send the email
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(username, password)
    server.sendmail(sender, receivers, message)
    server.quit()
Rest Functions

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

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

@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 default value
        for t in trash:
            if t['ID'] == ID:
                c['wastecontainer'] = chooseWasteContainer(currentVolume)
                break
        t = Thread(target=turnOnStripes, args=(colour,))
        t.start()
        return jsonify({'colour': c})
```

```python
def chooseWasteContainer(currentVolume):
    wasteContainer = "wasteContainer_X"
    minFreeSpace = 10000000000
    for t in trash:
        if t['type'] == wasteContainer:
            freeSpace = t['currentVolume'] + t['currentVolume']
            if freeSpace > currentVolume and freeSpace < minFreeSpace:
                minFreeSpace = freeSpace
                wasteContainer = t['name']
    return wasteContainer
```

```python
def turnOnStripes(colour):
    # color_value is a number between 0 and 255
    color_value = int(color_code[colour])
    # turn on the stripes...
    body = {'on': True, 'how': t: str(color_value) + '}'
    rest.send(PF, url_to_call, body, 'Content-Type: application/json')
    #...wait 30 seconds...
    time.sleep(30)
    #...and finally turn off the stripes
    body = {'on': False}
    rest.send(PF, url_to_call, body, 'Content-Type: application/json')
    # make the color available for other assignment
    free_color[colour] = True
    # function which send the XHR request to the Phillips Bridge for turn off the stripes
    def turnOffStripes():
        body = {'on': False}
        rest.send(PF, 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