Université de Paris

Analyseur d’air ambiant

Alexandre Guerassimov, Christian Qian

Projet Long 2019-2020

Fonctionnalités

Affichages

- Une mesure
- Un chemin optimal

Mesures

- Température
- Humidité
- Qualité de l'air
- Taux en particules
- Bruit ambiant

Architecture

Arduino

- Gestion des capteurs
    - Récupération des valeurs
    - Interprétation des valeurs
    - Enregistrement des données

Difficultés

- Conversion des valeurs de son
- Réception des valeurs pour les particules

Architecture

Python

Interface.py

- Extraction et vérification des données
- Création de l'interface
- Affichage des données et des trajets

algos.py

- Algorithmes de prédiction
  - Regression linéaire
  - Regression polynomiale
  - Perceptron

Architecture

Difficultés

Interface.py

- Coloration de la carte
  - Choropleth
  - Dessin en pixels

algos.py

- Regression
  - Valeur en X
- Perceptron
  - Paramètres

Programmation

```python
for e in self.listeFiles:
    if e == 'SG':
        continue
    self.searchName = re.search(r"([0-9]+-[0-9]+)_[0-9]+\.\w+", e)
    if self.searchName.group(1) not in self.listeDates:
        self.listeDates.append(self.searchName.group(1))
    self.tmp_file = pd.read_csv("./data/" + e, sep=';')
    self.tmp_file = self.tmp_file.dropna(axis='rows')
    self.tmp_file.loc[self.tmp_file['Particules'] <= 0.62, 'Particules'] = np.nan

    self.tmp_file.iloc[:, [7, 8, 10, 11]] = self.tmp_file.iloc[:, [7, 8, 10, 11]].apply(self.__mask, axis=0)

    self.tmp_file = self.tmp_file.reset_index(drop=True)
    self.tmp_listval = self.tmp_file.iloc[0, 0:7]
    self.tmp_listval = self.tmp_listval.append(self.tmp_file.iloc[:, 7:12].mean(skipna=True))
    self.listeData.append(self.tmp_listval)
```

Avant

```python
for e in self.listeFiles:
    if e == 'SG':
        continue
    self.searchName = re.search(r"([0-9]+-[0-9]+)_[0-9]+\.\w+", e)
    if self.searchName.group(1) not in self.listeDates:
        self.listeDates.append(self.searchName.group(1))
    self.tmp_file = pd.read_csv("./data/" + e, sep=';')
    self.tmp_file = self.tmp_file.dropna(axis='rows')

    self.tmp_listval = []
    self.tmp_listval.append(self.tmp_file['Temperature'].mean())
    self.tmp_listval.append(self.tmp_file['Humidite'].mean())
    self.tmp_listval.append(self.tmp_file['Particules'].mean())
    self.tmp_listval.append(self.tmp_file['Qualite'].mean())
    self.tmp_listval.append(self.tmp_file['Son'].mean())

    self.tmp_std = []
    self.tmp_std.append(self.tmp_file['Temperature'].std())
    self.tmp_std.append(self.tmp_file['Humidite'].std())
    self.tmp_std.append(self.tmp_file['Particules'].std())
    self.tmp_std.append(self.tmp_file['Qualite'].std())
    self.tmp_std.append(self.tmp_file['Son'].std())

    self.tmp_file = self.__mask(self.tmp_file, 'Temperature', 0)
    self.tmp_file = self.__mask(self.tmp_file, 'Humidite', 1)
    self.tmp_file = self.__mask(self.tmp_file, 'Particules', 2)
    self.tmp_file = self.__mask(self.tmp_file, 'Qualite', 3)
    self.tmp_file = self.__mask(self.tmp_file, 'Son', 4)

    self.tmp_listval = []
    self.tmp_file.reset_index()
    self.tmp_listval.append(self.tmp_file.iloc[0].Jour)
    self.tmp_listval.append(self.tmp_file.iloc[0].Mois)
    self.tmp_listval.append(self.tmp_file.iloc[0].Heure)
    self.tmp_listval.append(self.tmp_file.iloc[0].Minutes)
    self.tmp_listval.append(self.tmp_file.iloc[0].Secondes)
    self.tmp_listval.append(self.tmp_file.iloc[0].Lieu)
    self.tmp_listval.append(self.tmp_file.iloc[0].Couloir)
    self.tmp_listval.append(self.tmp_file['Temperature'].mean())
    self.tmp_listval.append(self.tmp_file['Humidite'].mean())
    self.tmp_listval.append(self.tmp_file['Particules'].mean())
    self.tmp_listval.append(self.tmp_file['Qualite'].mean())
    self.tmp_listval.append(self.tmp_file['Son'].mean())

    self.listeData.append(self.tmp_listval)
```

Après

```python
for e in self.listeFiles:
    if e == 'SG':
        continue
    self.searchName = re.search(r"([0-9]+-[0-9]+)_[0-9]+\.\w+", e)
    if self.searchName.group(1) not in self.listeDates:
        self.listeDates.append(self.searchName.group(1))
    self.tmp_file = pd.read_csv("./data/" + e, sep=';')
    self.tmp_file = self.tmp_file.dropna(axis='rows')
    self.tmp_file.loc[self.tmp_file['Particules'] <= 0.62, 'Particules'] = np.nan

    self.tmp_file.iloc[:, [7, 8, 10, 11]] = self.tmp_file.iloc[:, [7, 8, 10, 11]].apply(self.__mask, axis=0)

    self.tmp_file = self.tmp_file.reset_index(drop=True)
    self.tmp_listval = self.tmp_file.iloc[0, 0:7]
    self.tmp_listval = self.tmp_listval.append(self.tmp_file.iloc[:, 7:12].mean(skipna=True))
    self.listeData.append(self.tmp_listval)
```

COVID19

- Changement de batiment/plan
    - Absence de variété dans les trajets possibles
    - Affichage par groupe de salles
    - Gaspillage des données antérieures

Conclusion

Acquis

- Utilisation de l'Arduino
- Application de connaissances théoriques
  - Programmation et utilisation de méthodes de regression
  - Programmation et utilisation de perceptron

Améliorations

- Changement de langage pour l'interface ?
- Utilisation d'un Raspberry Pi pour plus d'autonomie de l'appareil ?