Instalação:

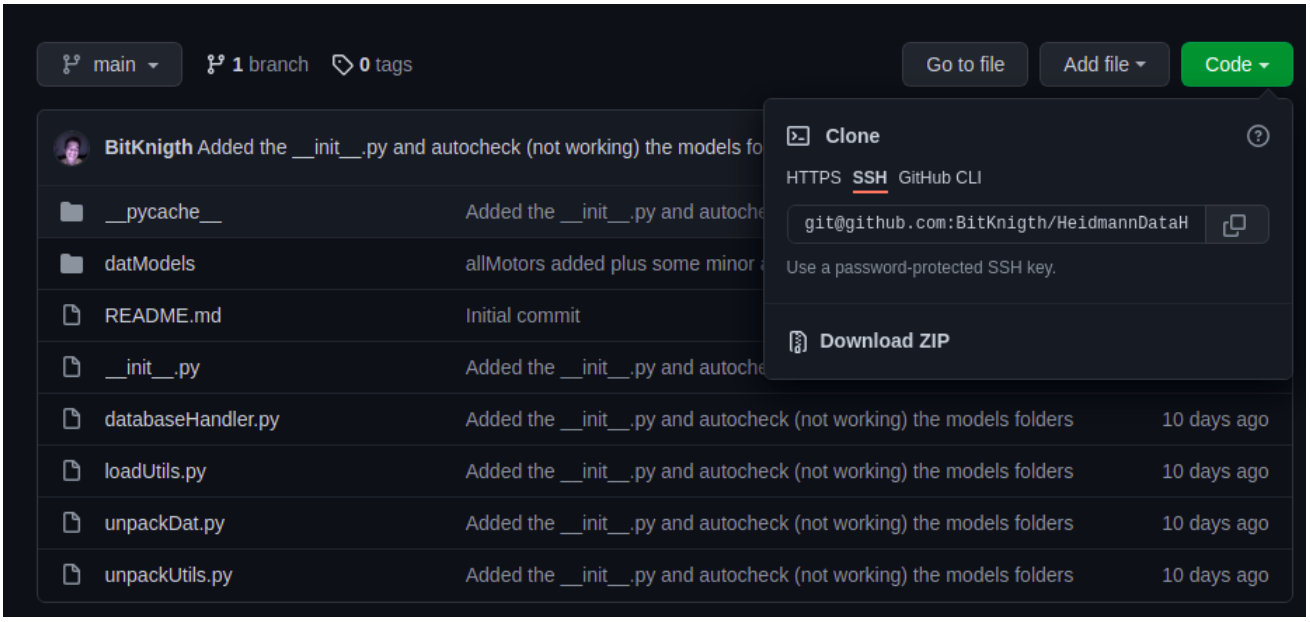
A biblioteca pode ser encontrada nos seguintes repositorios:
- GitHub: https://github.com/BitKnigth/HeidmannDataHandler

É possível instalar a biblioteca de duas formas:

1. Clonando o repositório para seu diretório de trabalho, usando o comando git clone:

```
git clone https://github.com/BitKnigth/HeidmannDataHandler
```

2. Ou então tambem é possível baixar o conteudo do repositório em formato zip, e extrair para seu diretório de trabalho:



Caso a instalação seja bem sucedida, a arvore do seu diretório deve se parecer com isso:

```
        workingDirectory/
->      ├── HeidmannDataHandler
        ├── yourFile1.py
        ├── yourFile2.py
        ├── ...
        └── ...
```

QuickStart

Aqui vai alguns exemplos mais praticos de utilização, assim que ela tiver funcional já coloco.

Funcionalidades

- ### Classe Loader

  Responsável por manipular* os dados de ruído dos motores.

- ***Obs.:*** Para utilizar metodos de instancia, é necessário criar uma instancia da classe para o motor desejado, sendo feito da seguinte forma:

```
loaderInstance = Loader(
                                        motor=str,
                                        speed=int
                                        )
```

Onde:

- *motor:* Nome do motor a ser carregado
- *speed:* Valor inteiro da velocidade desejada

- **modelsRowsArray**

  - Método de instancia
  - Gera um array contendo os dados do modelo de motor instanciado.

  ```
  modelRowsArray = loaderInstance.modelRowsArray(inLineAG=bool, inLineSpeed=bool)
  ```

  Parametros:

  - *inLineSpeed*: booleano, caso *True*, retorna o valor da velocidade em cada linha do array.
  - *inLineAG*: booleano, caso *True*, retorna o valor do angulo em cada linha do array.

  Retorna um array em que cada item é um array correspondendo a um ponto do dataset do motor escolhido, na seguinte forma:

  ```
  [
  [60, 10, 50, 101.52235158772478],
  [60, 10, 63, 100.65675502055247],
  [60, 10, 80, 100.60309609482263],
  ...
  ]
  ```

  - Onde a sequencia dos itens em cada ponto é priorizada da segunte forma:

  ```
  [velocidade, angulo, frequencia_nominal, ruído]
  ```

- **buildMatrix**

  - Método de instancia
  - *Gera uma matriz contendo os dados do motor da instância, onde cada linha é referente a uma frequencia, e cada coluna é referente a um angulo, retornando um array com as linhas da matriz, um array contendo referencias para as frequencias e outro para os angulos.*

  ```
          matrix, frequencie_label, angle_label = loaderInstance.buildMatrix()
  ```

  *Retorna tres arrays, sendo o primeiro a matriz em si, o segundo são os rotulos referentes as linhas da matriz, ou seja, a frequencia no indice **X** do array contendo o label, é a frequencia em questão na linha **X** da matriz*, o terceiro são os rotulos para as colunas da matriz, onde o angulo no indice **Y** do array, é o angulo em questão na coluna **Y**

  - ***Matriz:***

  ```
  [
          [101.522, 99.208, 99.817, 100.050, 100.465, 101.277, ..., 109.024],
          [100.656, 100.559, 99.550, 99.025,      99.696, 100.097, ..., 110.213],
           ...
  ]
  ```

$$\begin{vmatrix} & & & \\ f1(angulo1) & f1(angulo2) & \dots & f1(angulon) \\ f2(angulo1) & f2(angulo2) & \dots & f2(angulon) \\ f3(angulo1) & f3(angulo2) & \dots & f3(angulon) \\ \dots & \dots & \dots & \dots \\ fn(angulo1) & fn(angulo2) & \dots & fn(angulon) \end{vmatrix}$$

- ***frequencie_label:***

```
        [50, 63, 80, 100, 125, 160, 200, 250, 315, 400, 500, ..., 16000]
```

- ***angle_label:***

```
    [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160]
```

- **allMotors**

  - Método estático
  - Gera um array contendo dados de todos os motores.

```
motorsArray = Loader.allMotors(speed=int, inLineAg=bool)
```

Parametros:

- *speed*: valor inteiro da velocidade desejada.
- *inLineAg*: booleano, caso *True*, retorna o valor do angulo em cada linha do array.

Retorna um array em que cada item é um array correspondendo a um ponto do dataset do motor escolhido, na seguinte forma:

```
[
[60, 10, 50, 101.52235158772478],
[60, 10, 63, 100.65675502055247],
[60, 10, 80, 100.60309609482263],
...
]
```

- Onde a sequencia dos itens em cada ponto é priorizada da segunte forma:

```
[velocidade, angulo, frequencia_nominal, ruído]
```
