In [1]:

```
# from nilm_metadata import get_appliance_types
# appliance_types = get_appliance_types()
# print(appliance_types)

# import os
# os.getcwd()
```

Carregando bibliotecas...

In [2]:

```
!pip install seaborn

import seaborn as sns

from matplotlib import rcParams
import matplotlib.pyplot as plt
import pandas as pd
import nilmtk
from nilmtk import MeterGroup
from nilmtk.api import API
import warnings
warnings.filterwarnings("ignore")

plt.style.use('ggplot')
rcParams['figure.figsize'] = (13, 10)

# import pathlib
# pathlib.Path().resolve()
```

```
Requirement already satisfied: seaborn in ./miniconda3/envs/nilm_0.4.
3/lib/python3.7/site-packages (0.11.2)
Requirement already satisfied: scipy>=1.0 in ./miniconda3/envs/nilm_0.
4.3/lib/python3.7/site-packages (from seaborn) (1.7.1)
Requirement already satisfied: pandas>=0.23 in ./miniconda3/envs/nilm_
0.4.3/lib/python3.7/site-packages (from seaborn) (0.25.3)
Requirement already satisfied: matplotlib>=2.2 in ./miniconda3/envs/ni
lm_0.4.3/lib/python3.7/site-packages (from seaborn) (3.1.3)
Requirement already satisfied: numpy>=1.15 in ./miniconda3/envs/nilm_
0.4.3/lib/python3.7/site-packages (from seaborn) (1.19.5)
Requirement already satisfied: kiwisolver>=1.0.1 in ./miniconda3/envs/
nilm_0.4.3/lib/python3.7/site-packages (from matplotlib>=2.2->seaborn)
(1.3.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.
1 in ./miniconda3/envs/nilm_0.4.3/lib/python3.7/site-packages (from ma
tplotlib>=2.2->seaborn) (2.4.7)
Requirement already satisfied: cycler>=0.10 in ./miniconda3/envs/nilm_
0.4.3/lib/python3.7/site-packages (from matplotlib>=2.2->seaborn) (0.1
0.0)
Requirement already satisfied: python-dateutil>=2.1 in ./miniconda3/en
vs/nilm_0.4.3/lib/python3.7/site-packages (from matplotlib>=2.2->seabo
rn) (2.8.2)
Requirement already satisfied: six in ./miniconda3/envs/nilm_0.4.3/li
b/python3.7/site-packages (from cycler>=0.10->matplotlib>=2.2->seabor
n) (1.16.0)
Requirement already satisfied: pytz>=2017.2 in ./miniconda3/envs/nilm_
0.4.3/lib/python3.7/site-packages (from pandas>=0.23->seaborn) (2021.
1)
```

Converter

In [3]:

```
# from nilmtk.dataset_converters import convert_hb
# convert_hb('./BD/CASA/convert', './data/teste17.h5')
```

In [4]:

```
# st = pd.HDFStore("./data/teste17.h5")
# print (st.keys())

# print (st['/building1/elec/meter1'].head())
# print (st['/building1/elec/meter2'].head())
# print (st['/building1/elec/meter3'].head())

# st.close()
```

Carregando dataset

In [5]:

```
from nilmtk.api import API
import warnings
warnings.filterwarnings("ignore")

from nilmtk import DataSet
from nilmtk.utils import print_dict

hb = DataSet('teste17.h5')
#iawe = DataSet('/data/iawe.h5')

print_dict(hb.metadata)
print_dict(hb.buildings)
```

- **name**: HB
- **long_name**: The Reference Energy Disaggregation Data set
- **creators**:
  - Henrique
- **publication_date**: 2021
- **institution**: IFCE
- **contact**: henrique@ufc.br
- **description**: Several weeks of power data for 6 different homes.
- **subject**: Disaggregated power demand from domestic buildings.
- **number_of_buildings**: 1
- **timezone**: America/Fortaleza
- **geo_location**:
  - **locality**: Fortaleza
  - **country**: BR
  - **latitude**: -3.743443904897663
  - **longitude**: -38.526093995496886
- **related_documents**:
  - http://redd.csail.mit.edu (http://redd.csail.mit.edu)
  - J. Zico Kolter and Matthew J. Johnson. REDD: A public data set for energy disaggregation research. In proceedings of the SustKDD workshop on Data Mining Applications in Sustainability, 2011. http://redd.csail.mit.edu/kolter-kddsust11.pdf (http://redd.csail.mit.edu/kolter-kddsust11.pdf)
- **schema**: https://github.com/nilmtk/nilm_metadata/tree/v0.2 (https://github.com/nilmtk/nilm_metadata/tree/v0.2)
- **meter_devices**:
  - **eMonitor**:
    - **model**: sonoff
    - **manufacturer**: Powerhouse Dynamics
    - **manufacturer_url**: http://powerhousedynamics.com (http://powerhousedynamics.com)
    - **description**: ...
    - **sample_period**: 5
    - **max_sample_period**: 30
    - **measurements**:
      - {'physical_quantity': 'power', 'type': 'active', 'upper_limit': 1142, 'lower_limit': 0}
      - {'physical_quantity': 'power', 'type': 'apparent', 'upper_limit': 1215, 'lower_limit': 0}

        - {'physical_quantity': 'power', 'type': 'reactive', 'upper_limit': 901, 'lower_limit': 0}
        - {'physical_quantity': 'power factor', 'upper_limit': 1, 'lower_limit': 0}
        - {'physical_quantity': 'voltage', 'upper_limit': 232, 'lower_limit': 0}
        - {'physical_quantity': 'current', 'upper_limit': 6, 'lower_limit': 0}
      - **wireless**: True
  - **REDD_whole_house**:
      - **model**: pzem004t
      - **description**: ...
      - **sample_period**: 0.5
      - **max_sample_period**: 30
      - **measurements**:
        - {'physical_quantity': 'voltage', 'upper_limit': 230, 'lower_limit': 0}
        - {'physical_quantity': 'current', 'upper_limit': 15, 'lower_limit': 0}
        - {'physical_quantity': 'power', 'type': 'active', 'upper_limit': 3016, 'lower_limit': 0}
        - {'physical_quantity': 'frequency', 'upper_limit': 61, 'lower_limit': 0}
        - {'physical_quantity': 'power factor', 'upper_limit': 1, 'lower_limit': 0}
      - **wireless**: False

- **1**: Building(instance=1, dataset='HB')

Gráfico Geral

In [6]:

```
build = 1
elec = hb.buildings[build].elec
elec.mains().power_series_all_data().head()
```

Out[6]:

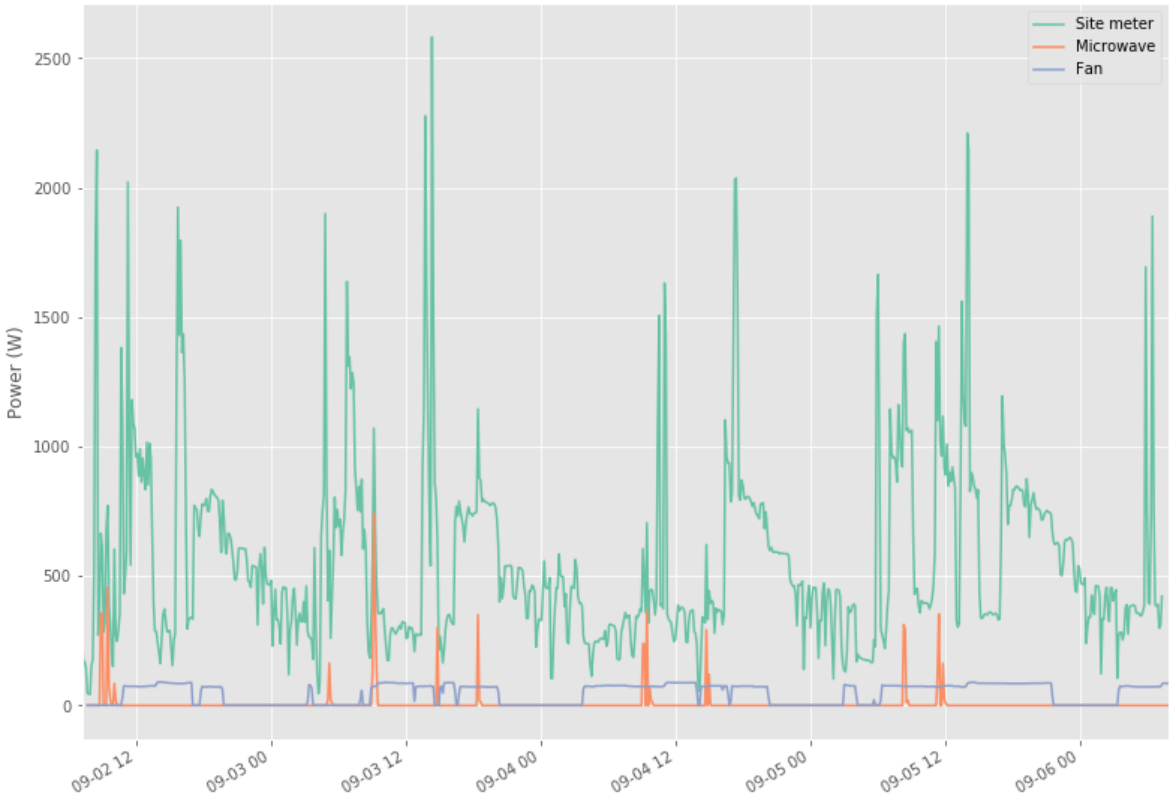
```
2021-09-02 07:14:34.515000-03:00    167.199997
2021-09-02 07:14:35.014000-03:00    167.199997
2021-09-02 07:14:35.513000-03:00    167.199997
2021-09-02 07:14:36.013000-03:00    167.199997
2021-09-02 07:14:36.527000-03:00    166.899994
Name: (power, active), dtype: float32
```

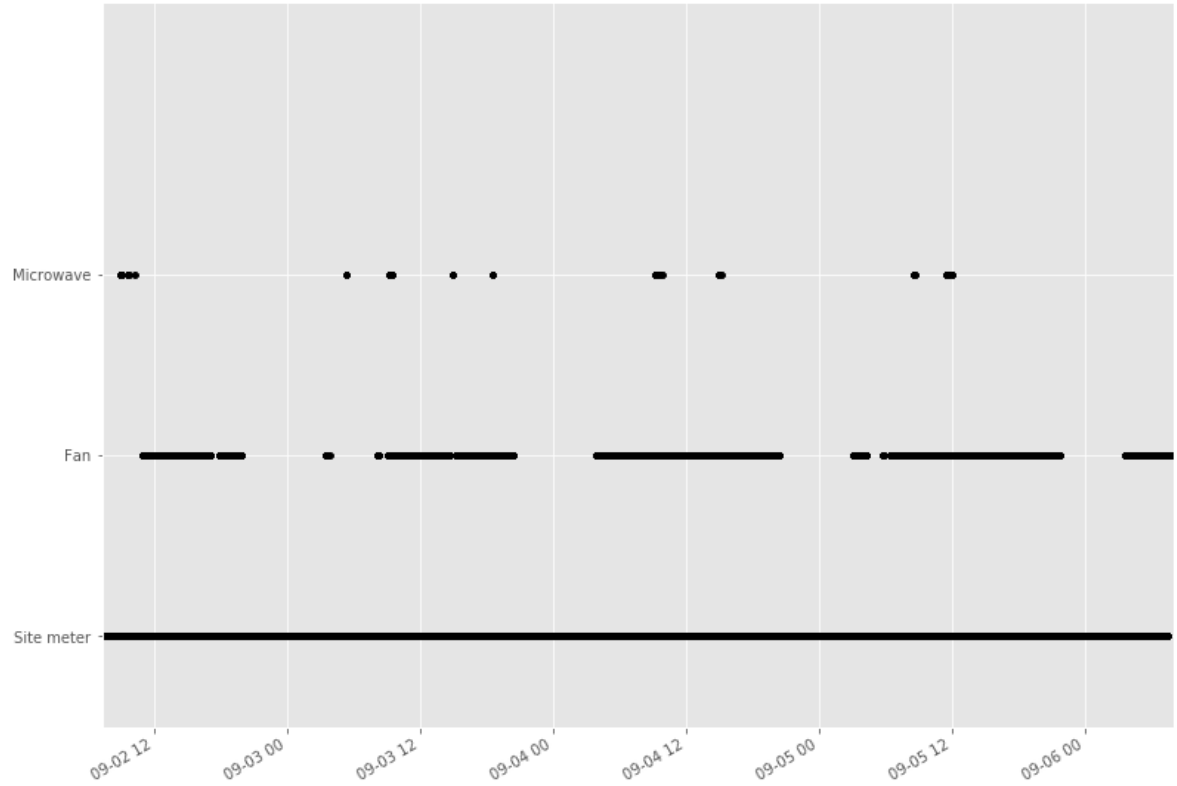
In [7]:

```
sns.set_palette("Set2", n_colors=5)
elec.mains().plot()
elec['microwave'].plot()
elec['fan'].plot()

# Set a threshold to remove residual power noise when devices are off
elec.plot_when_on(on_power_threshold = 40) # Plot appliances when they are in use¶

# elec.draw_wiring_graph()
```

Out[7]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff585be2790>
```





Dados

Proporção de energia submedida

In [8]:

```
elec.proportion_of_energy_submetered()
```

```
Running MeterGroup.proportion_of_energy_submetered...
```

Out[8]:

```
0.09288249528613458
```

Total Energy

In [9]:

```
elec.mains().total_energy()
```

Out[9]:

```
active    53.946047
dtype: float64
```

Energy per submeter

In [10]:

```
energy_per_meter = elec.submeters().energy_per_meter() # kWh, again
energy_per_meter
```

```
2/2 ElecMeter(instance=3, building=1, dataset='HB', appliances=[Applia
nce(type='microwave', instance=1)])
```

Out[10]:

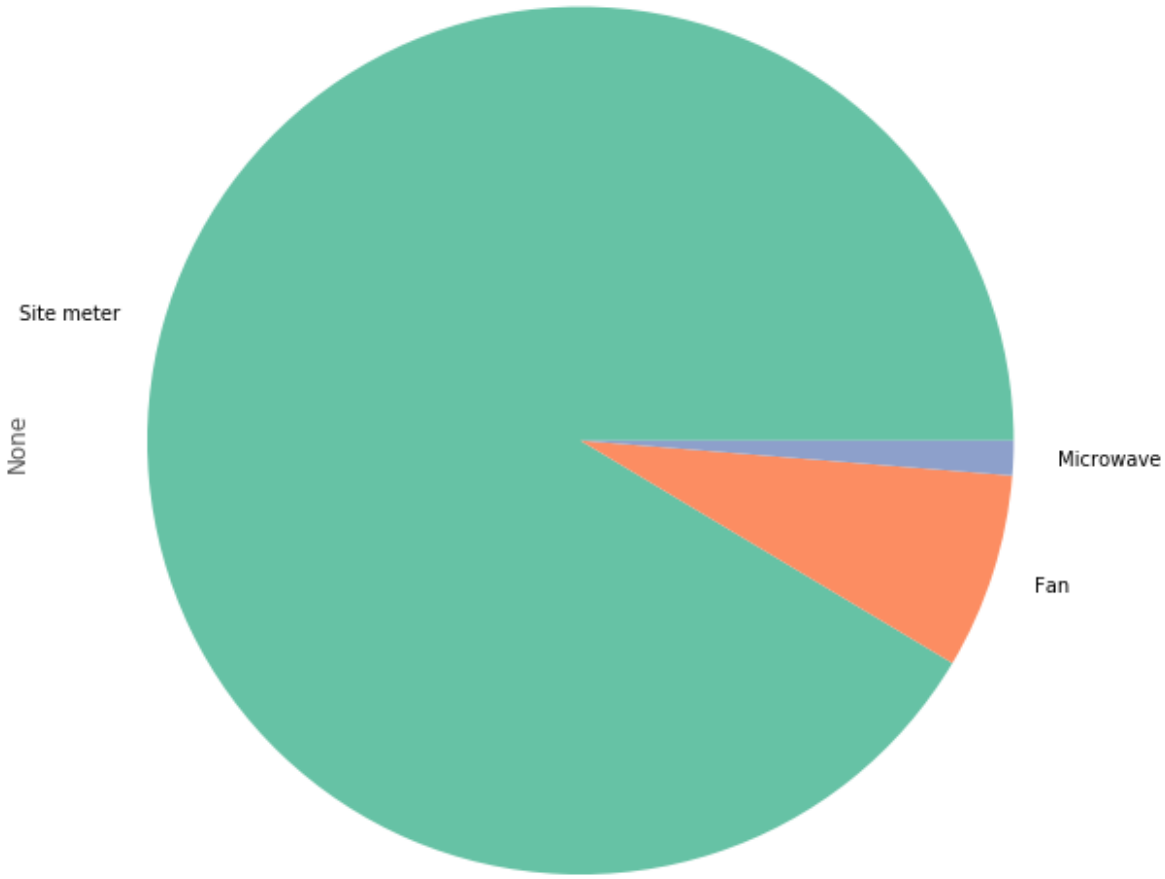
	(2, 1, HB)	(3, 1, HB)
active	4.298278	0.757815
apparent	NaN	NaN
reactive	NaN	NaN

Plot fraction of energy consumption of each appliance

In [11]:

```
# fraction = elec.submeters().fraction_per_meter().dropna()
fraction = elec.fraction_per_meter().dropna()
# Create convenient labels
labels = elec.get_labels(fraction.index)
plt.figure(figsize=(10,30))
fraction.plot(kind='pie', labels=labels);
```

```
3/3 ElecMeter(instance=3, building=1, dataset='HB', appliances=[Applia
nce(type='microwave', instance=1)])
```



Quadro Geral

In [12]:

```
print(elec)
elec.mains()
```

```
MeterGroup(meters=
  ElecMeter(instance=1, building=1, dataset='HB', site_meter, applianc
es=[])
  ElecMeter(instance=2, building=1, dataset='HB', appliances=[Applianc
e(type='fan', instance=1)])
  ElecMeter(instance=3, building=1, dataset='HB', appliances=[Applianc
e(type='microwave', instance=1)])
)
```

Out[12]:

```
ElecMeter(instance=1, building=1, dataset='HB', site_meter, appliances
=[])
```

In [13]:

```
from nilmtk.elecmeter import ElecMeterID##### Quadro Geral

meter1 = elec[ElecMeterID(instance=1, building=build, dataset='HB')]

next(meter1.load()).head()
```

Out[13]:

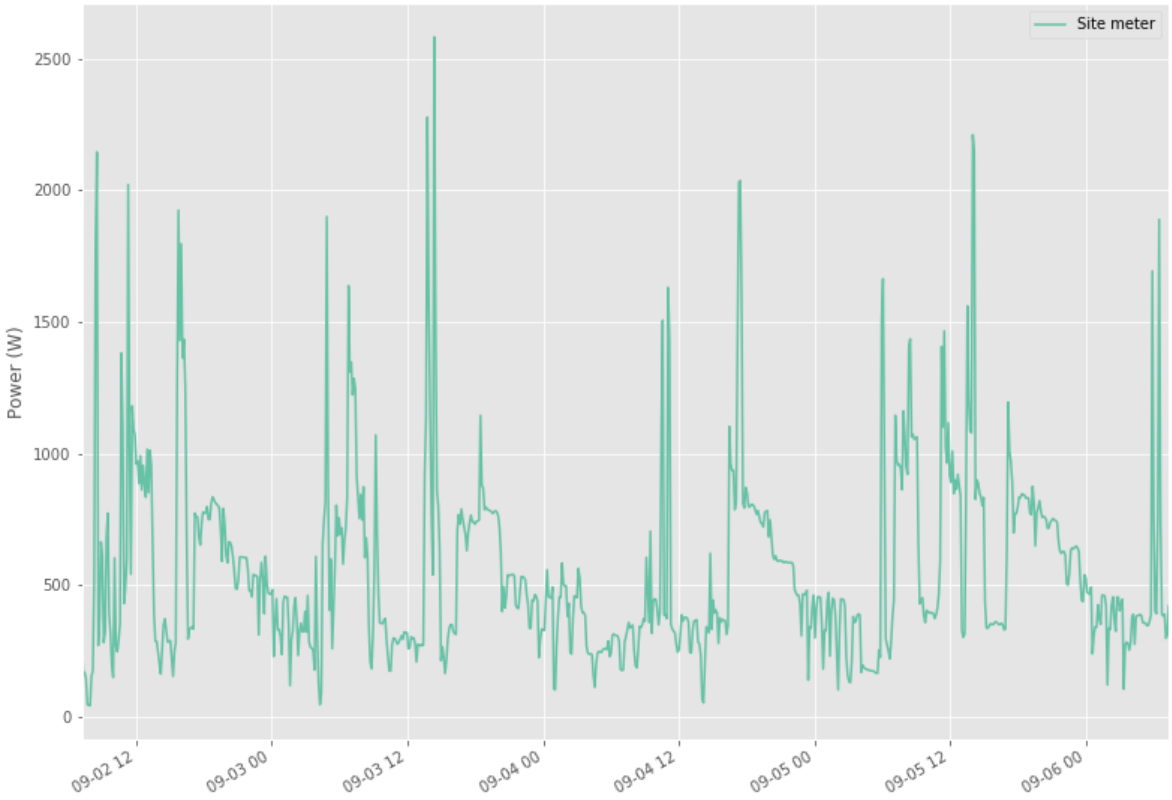
physical_quantity	voltage	frequency	power factor	current	power
type					active
2021-09-02 07:14:34.515000-03:00	221.600006	60.0	0.84	0.896	167.199997
2021-09-02 07:14:35.014000-03:00	221.600006	60.0	0.84	0.896	167.199997
2021-09-02 07:14:35.513000-03:00	221.600006	60.0	0.84	0.896	167.199997
2021-09-02 07:14:36.013000-03:00	221.600006	60.0	0.84	0.896	167.199997
2021-09-02 07:14:36.527000-03:00	221.500000	60.0	0.85	0.890	166.899994

In [14]:

```
meter1.plot()
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff581a13690>
```



A taxa de abandono é um número entre 0 e 1 que especifica a proporção de amostras ausentes. Uma taxa de abandono de 0 significa que nenhuma amostra está faltando. Um valor de 1 significaria que todas as amostras estão faltando

In [15]:

```
meter1.dropout_rate()
```

Out[15]:

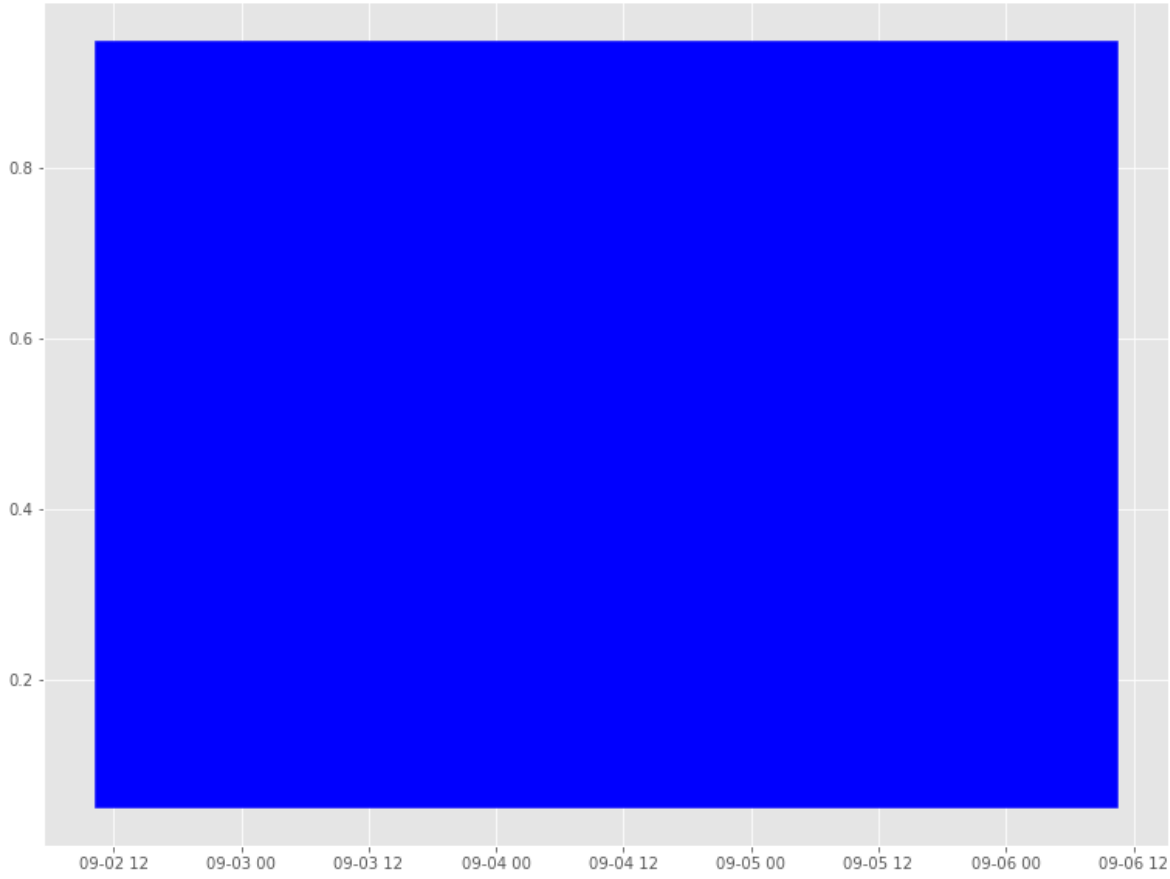
```
0.0002946431279545747
```

In [16]:

```
good_sections = meter1.good_sections(full_results=True)
good_sections.plot()
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff5819e3150>
```



In [17]:

```
good_sections.combined()
```

Out[17]:

```
[TimeFrame(start='2021-09-02 07:14:34.515000-03:00', end='2021-09-06 0
7:24:15.557000-03:00', empty=False)]
```

Microondas

In [18]:

```
microwave= elec['microwave']
#microwave.available_columns()
next(microwave.load()).head()
```

Out[18]:

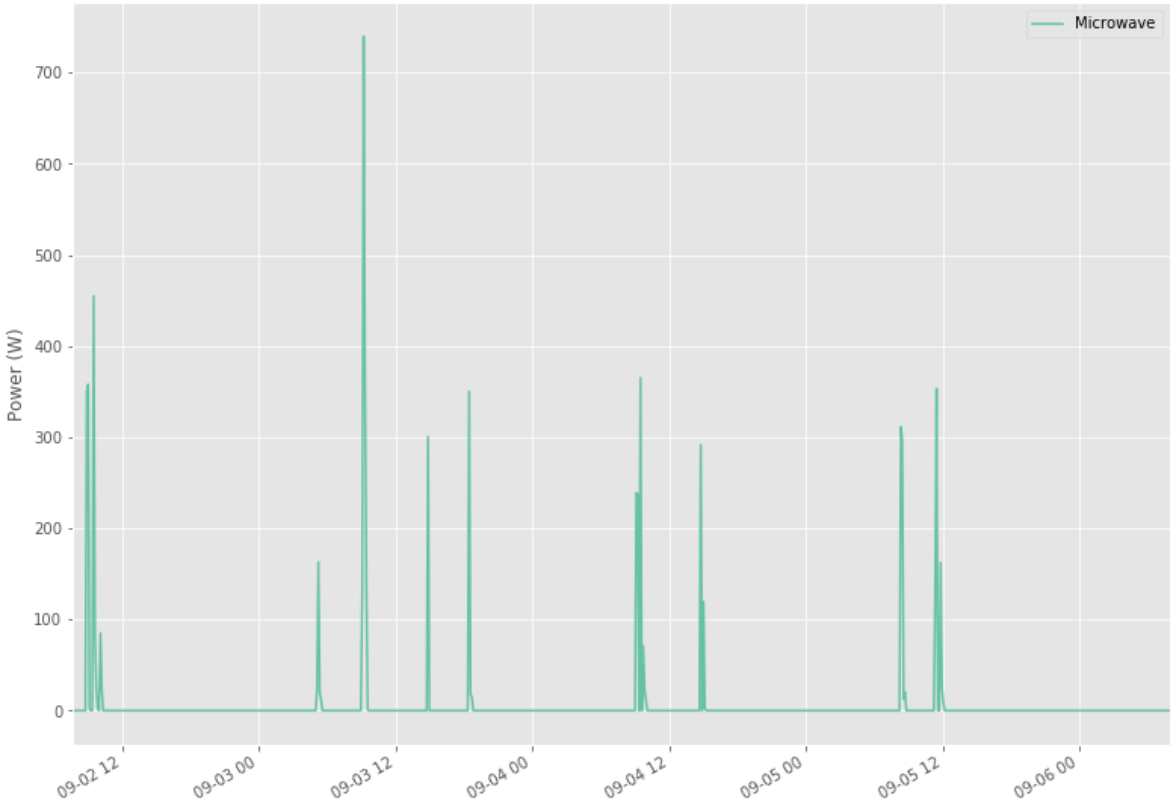
physical_quantity	power	voltage	power		current	power factor
type	active		reactive	apparent		
2021-09-02 07:47:51-03:00	0.0	221.882004	0.0	0.0	0.0	0.0
2021-09-02 07:47:56-03:00	0.0	221.882004	0.0	0.0	0.0	0.0
2021-09-02 07:48:01-03:00	0.0	222.406006	0.0	0.0	0.0	0.0
2021-09-02 07:48:06-03:00	0.0	222.143997	0.0	0.0	0.0	0.0
2021-09-02 07:48:11-03:00	0.0	221.621994	0.0	0.0	0.0	0.0

In [19]:

```
microwave.plot()
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff58191d410>
```



A taxa de abandono é um número entre 0 e 1 que especifica a proporção de amostras ausentes. Uma taxa de abandono de 0 significa que nenhuma amostra está faltando. Um valor de 1 significaria que todas as amostras estão faltando

In [20]:

```
microwave.dropout_rate()
```

Out[20]:

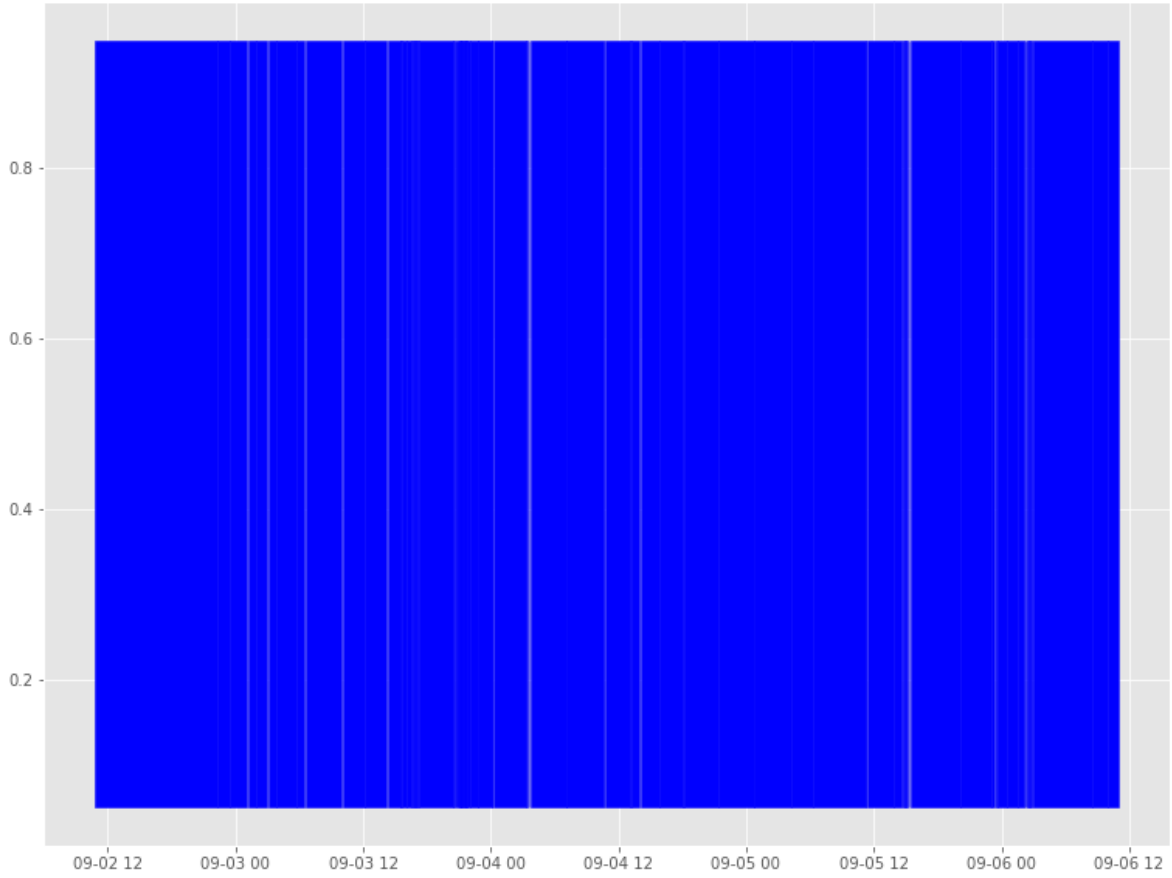
```
0.0019130418281826 37
```

In [21]:

```
good_sections = microwave.good_sections(full_results=True)
good_sections.plot()
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff58187aed0>
```



In [22]:

```
good_sections.combined()
```

Out[22]:

```
[TimeFrame(start='2021-09-02 07:47:51-03:00', end='2021-09-02 19:17:21
-03:00', empty=False),
 TimeFrame(start='2021-09-02 19:20:01-03:00', end='2021-09-02 20:25:21
-03:00', empty=False),
 TimeFrame(start='2021-09-02 20:26:06-03:00', end='2021-09-02 22:07:21
-03:00', empty=False),
 TimeFrame(start='2021-09-02 22:07:56-03:00', end='2021-09-02 22:59:11
-03:00', empty=False),
 TimeFrame(start='2021-09-02 23:00:03-03:00', end='2021-09-03 00:01:28
-03:00', empty=False),
 TimeFrame(start='2021-09-03 00:05:58-03:00', end='2021-09-03 00:51:43
-03:00', empty=False),
 TimeFrame(start='2021-09-03 00:53:23-03:00', end='2021-09-03 02:39:33
-03:00', empty=False),
 TimeFrame(start='2021-09-03 02:40:18-03:00', end='2021-09-03 03:30:48
-03:00', empty=False),
 TimeFrame(start='2021-09-03 03:33:08-03:00', end='2021-09-03 07:00:39
-03:00', empty=False),
 TimeFrame(start='2021-09-03 07:02:10-03:00', end='2021-09-03 09:10:29
-03:00', empty=False),
 TimeFrame(start='2021-09-03 09:11:09-03:00', end='2021-09-03 11:13:15
-03:00', empty=False),
 TimeFrame(start='2021-09-03 11:15:39-03:00', end='2021-09-03 12:31:03
-03:00', empty=False),
 TimeFrame(start='2021-09-03 12:31:49-03:00', end='2021-09-03 12:32:03
-03:00', empty=False),
 TimeFrame(start='2021-09-03 12:32:49-03:00', end='2021-09-03 12:39:04
-03:00', empty=False),
 TimeFrame(start='2021-09-03 12:40:29-03:00', end='2021-09-03 13:05:33
-03:00', empty=False),
 TimeFrame(start='2021-09-03 13:06:09-03:00', end='2021-09-03 13:28:08
-03:00', empty=False),
 TimeFrame(start='2021-09-03 13:28:53-03:00', end='2021-09-03 13:40:23
-03:00', empty=False),
 TimeFrame(start='2021-09-03 13:40:58-03:00', end='2021-09-03 13:49:08
-03:00', empty=False),
 TimeFrame(start='2021-09-03 13:50:28-03:00', end='2021-09-03 13:50:38
-03:00', empty=False),
 TimeFrame(start='2021-09-03 13:51:29-03:00', end='2021-09-03 13:55:33
-03:00', empty=False),
 TimeFrame(start='2021-09-03 13:56:54-03:00', end='2021-09-03 14:00:33
-03:00', empty=False),
 TimeFrame(start='2021-09-03 14:01:13-03:00', end='2021-09-03 14:05:53
-03:00', empty=False),
 TimeFrame(start='2021-09-03 14:07:48-03:00', end='2021-09-03 14:13:28
-03:00', empty=False),
 TimeFrame(start='2021-09-03 14:15:29-03:00', end='2021-09-03 17:29:05
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:30:14-03:00', end='2021-09-03 17:35:59
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:37:57-03:00', end='2021-09-03 17:39:02
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:40:12-03:00', end='2021-09-03 17:41:02
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:42:52-03:00', end='2021-09-03 17:43:02
```

```
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:44:12-03:00', end='2021-09-03 17:45:02
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:46:57-03:00', end='2021-09-03 17:47:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:49:02-03:00', end='2021-09-03 17:49:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:50:17-03:00', end='2021-09-03 17:51:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:52:17-03:00', end='2021-09-03 17:53:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:54:18-03:00', end='2021-09-03 17:55:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:55:47-03:00', end='2021-09-03 17:57:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:58:17-03:00', end='2021-09-03 17:59:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:00:17-03:00', end='2021-09-03 18:01:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:01:47-03:00', end='2021-09-03 18:03:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:03:47-03:00', end='2021-09-03 18:05:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:05:47-03:00', end='2021-09-03 18:06:52
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:08:22-03:00', end='2021-09-03 18:09:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:09:47-03:00', end='2021-09-03 18:10:12
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:10:52-03:00', end='2021-09-03 18:11:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:12:17-03:00', end='2021-09-03 18:13:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:14:17-03:00', end='2021-09-03 18:15:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:16:17-03:00', end='2021-09-03 18:17:08
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:18:17-03:00', end='2021-09-03 18:19:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:20:17-03:00', end='2021-09-03 18:21:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:22:17-03:00', end='2021-09-03 18:23:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:24:17-03:00', end='2021-09-03 18:25:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:26:17-03:00', end='2021-09-03 18:27:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:28:17-03:00', end='2021-09-03 18:29:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:30:17-03:00', end='2021-09-03 18:31:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:31:47-03:00', end='2021-09-03 18:33:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:33:47-03:00', end='2021-09-03 18:35:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:37:02-03:00', end='2021-09-03 18:37:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:37:47-03:00', end='2021-09-03 18:39:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:40:57-03:00', end='2021-09-03 18:44:08
-03:00', empty=False),
```

```
 TimeFrame(start='2021-09-03 18:44:48-03:00', end='2021-09-03 18:45:07
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:47:53-03:00', end='2021-09-03 19:05:41
-03:00', empty=False),
 TimeFrame(start='2021-09-03 19:06:21-03:00', end='2021-09-03 19:43:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 19:43:52-03:00', end='2021-09-03 19:47:36
-03:00', empty=False),
 TimeFrame(start='2021-09-03 19:48:41-03:00', end='2021-09-03 21:04:57
-03:00', empty=False),
 TimeFrame(start='2021-09-03 21:05:42-03:00', end='2021-09-03 21:07:08
-03:00', empty=False),
 TimeFrame(start='2021-09-03 21:07:47-03:00', end='2021-09-03 21:07:52
-03:00', empty=False),
 TimeFrame(start='2021-09-03 21:08:47-03:00', end='2021-09-03 21:09:47
-03:00', empty=False),
 TimeFrame(start='2021-09-03 21:12:07-03:00', end='2021-09-04 00:19:02
-03:00', empty=False),
 TimeFrame(start='2021-09-04 00:19:57-03:00', end='2021-09-04 00:30:42
-03:00', empty=False),
 TimeFrame(start='2021-09-04 00:31:22-03:00', end='2021-09-04 00:38:07
-03:00', empty=False),
 TimeFrame(start='2021-09-04 00:41:17-03:00', end='2021-09-04 04:06:02
-03:00', empty=False),
 TimeFrame(start='2021-09-04 04:06:42-03:00', end='2021-09-04 04:07:47
-03:00', empty=False),
 TimeFrame(start='2021-09-04 04:08:32-03:00', end='2021-09-04 07:33:38
-03:00', empty=False),
 TimeFrame(start='2021-09-04 07:34:13-03:00', end='2021-09-04 07:36:34
-03:00', empty=False),
 TimeFrame(start='2021-09-04 07:39:44-03:00', end='2021-09-04 10:03:52
-03:00', empty=False),
 TimeFrame(start='2021-09-04 10:05:12-03:00', end='2021-09-04 10:09:12
-03:00', empty=False),
 TimeFrame(start='2021-09-04 10:10:22-03:00', end='2021-09-04 11:00:13
-03:00', empty=False),
 TimeFrame(start='2021-09-04 11:02:08-03:00', end='2021-09-04 12:47:12
-03:00', empty=False),
 TimeFrame(start='2021-09-04 12:49:47-03:00', end='2021-09-04 15:01:52
-03:00', empty=False),
 TimeFrame(start='2021-09-04 15:02:53-03:00', end='2021-09-04 15:07:38
-03:00', empty=False),
 TimeFrame(start='2021-09-04 15:09:02-03:00', end='2021-09-04 18:24:13
-03:00', empty=False),
 TimeFrame(start='2021-09-04 18:25:28-03:00', end='2021-09-04 21:43:09
-03:00', empty=False),
 TimeFrame(start='2021-09-04 21:45:04-03:00', end='2021-09-04 21:45:29
-03:00', empty=False),
 TimeFrame(start='2021-09-04 21:46:09-03:00', end='2021-09-05 01:13:53
-03:00', empty=False),
 TimeFrame(start='2021-09-05 01:15:53-03:00', end='2021-09-05 03:12:39
-03:00', empty=False),
 TimeFrame(start='2021-09-05 03:13:28-03:00', end='2021-09-05 08:12:28
-03:00', empty=False),
 TimeFrame(start='2021-09-05 08:14:18-03:00', end='2021-09-05 08:15:28
-03:00', empty=False),
 TimeFrame(start='2021-09-05 08:16:03-03:00', end='2021-09-05 08:16:28
-03:00', empty=False),
 TimeFrame(start='2021-09-05 08:18:09-03:00', end='2021-09-05 10:48:08
-03:00', empty=False),
 TimeFrame(start='2021-09-05 10:49:29-03:00', end='2021-09-05 11:36:03
```

```
-03:00', empty=False),
 TimeFrame(start='2021-09-05 11:36:43-03:00', end='2021-09-05 11:37:53
-03:00', empty=False),
 TimeFrame(start='2021-09-05 11:40:08-03:00', end='2021-09-05 11:44:23
-03:00', empty=False),
 TimeFrame(start='2021-09-05 11:45:03-03:00', end='2021-09-05 11:48:18
-03:00', empty=False),
 TimeFrame(start='2021-09-05 11:50:18-03:00', end='2021-09-05 11:57:48
-03:00', empty=False),
 TimeFrame(start='2021-09-05 11:59:23-03:00', end='2021-09-05 12:12:53
-03:00', empty=False),
 TimeFrame(start='2021-09-05 12:15:03-03:00', end='2021-09-05 12:22:33
-03:00', empty=False),
 TimeFrame(start='2021-09-05 12:24:13-03:00', end='2021-09-05 17:08:24
-03:00', empty=False),
 TimeFrame(start='2021-09-05 17:09:34-03:00', end='2021-09-05 20:03:14
-03:00', empty=False),
 TimeFrame(start='2021-09-05 20:04:18-03:00', end='2021-09-05 20:23:08
-03:00', empty=False),
 TimeFrame(start='2021-09-05 20:25:59-03:00', end='2021-09-05 20:28:23
-03:00', empty=False),
 TimeFrame(start='2021-09-05 20:29:03-03:00', end='2021-09-05 20:37:38
-03:00', empty=False),
 TimeFrame(start='2021-09-05 20:39:58-03:00', end='2021-09-05 21:28:19
-03:00', empty=False),
 TimeFrame(start='2021-09-05 21:30:38-03:00', end='2021-09-05 22:27:48
-03:00', empty=False),
 TimeFrame(start='2021-09-05 22:30:28-03:00', end='2021-09-05 23:02:59
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:05:33-03:00', end='2021-09-05 23:07:58
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:10:48-03:00', end='2021-09-05 23:13:23
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:15:24-03:00', end='2021-09-05 23:17:54
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:20:38-03:00', end='2021-09-05 23:23:23
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:25:08-03:00', end='2021-09-05 23:27:43
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:28:53-03:00', end='2021-09-05 23:37:23
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:39:39-03:00', end='2021-09-05 23:48:13
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:48:58-03:00', end='2021-09-05 23:52:34
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:54:49-03:00', end='2021-09-05 23:57:19
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:58:19-03:00', end='2021-09-06 05:30:44
-03:00', empty=False),
 TimeFrame(start='2021-09-06 05:33:59-03:00', end='2021-09-06 07:01:29
-03:00', empty=False),
 TimeFrame(start='2021-09-06 07:02:09-03:00', end='2021-09-06 07:55:59
-03:00', empty=False)]
```

Ventilador

In [23]:

```
fan = elec['fan']
#microwave.available_columns()
next(fan.load()).head()
```

Out[23]:

physical_quantity	power	voltage	power		current	power factor
type	active		reactive	apparent		
2021-09-02 07:47:51-03:00	0.767	222.287003	18.400000	18.454000	0.083	0.04
2021-09-02 07:47:56-03:00	1.091	222.546997	31.700001	31.761999	0.143	0.03
2021-09-02 07:48:01-03:00	1.091	222.028000	20.400000	20.479000	0.092	0.05
2021-09-02 07:48:06-03:00	0.923	222.287003	31.200001	31.187000	0.140	0.03
2021-09-02 07:48:11-03:00	0.923	221.770004	23.200001	23.195999	0.105	0.04

In [24]:

```
fan.plot()
```

Out[24]:

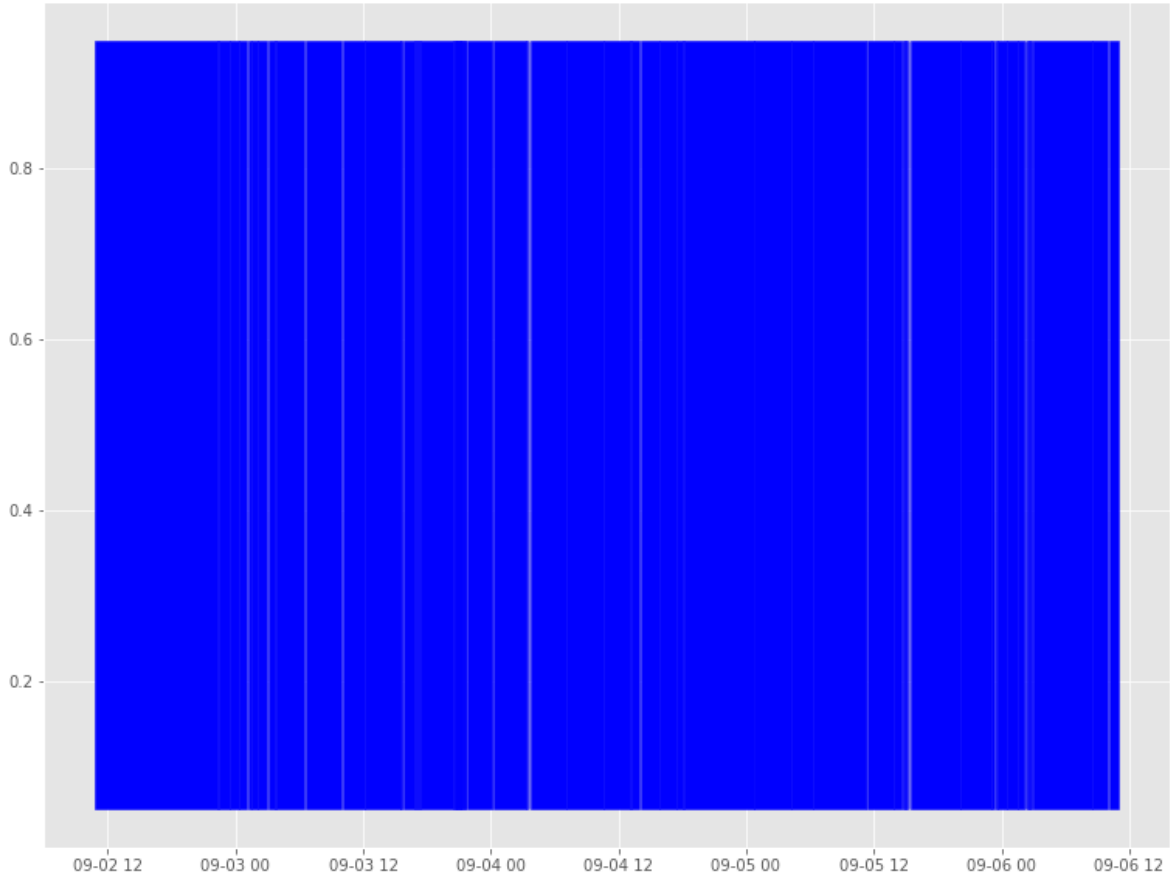
```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff5816b9410>
```

In [25]:

```
good_sections = fan.good_sections(full_results=True)
good_sections.plot()
```

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff5817dc4d0>
```



A taxa de abandono é um número entre 0 e 1 que especifica a proporção de amostras ausentes. Uma taxa de abandono de 0 significa que nenhuma amostra está faltando. Um valor de 1 significaria que todas as amostras estão faltando

In [26]:

```
fan.dropout_rate()
```

Out[26]:

```
0.002014694526278486
```

In [27]:

```
good_sections.combined()
```

Out[27]:

```
[TimeFrame(start='2021-09-02 07:47:51-03:00', end='2021-09-02 19:17:27
-03:00', empty=False),
 TimeFrame(start='2021-09-02 19:20:02-03:00', end='2021-09-02 19:27:32
-03:00', empty=False),
 TimeFrame(start='2021-09-02 19:28:25-03:00', end='2021-09-02 20:25:25
-03:00', empty=False),
 TimeFrame(start='2021-09-02 20:26:10-03:00', end='2021-09-02 21:18:32
-03:00', empty=False),
 TimeFrame(start='2021-09-02 21:19:17-03:00', end='2021-09-02 22:07:17
-03:00', empty=False),
 TimeFrame(start='2021-09-02 22:08:22-03:00', end='2021-09-02 22:31:57
-03:00', empty=False),
 TimeFrame(start='2021-09-02 22:32:32-03:00', end='2021-09-02 23:04:57
-03:00', empty=False),
 TimeFrame(start='2021-09-02 23:05:42-03:00', end='2021-09-03 00:01:27
-03:00', empty=False),
 TimeFrame(start='2021-09-03 00:05:57-03:00', end='2021-09-03 00:36:57
-03:00', empty=False),
 TimeFrame(start='2021-09-03 00:37:58-03:00', end='2021-09-03 00:51:43
-03:00', empty=False),
 TimeFrame(start='2021-09-03 00:53:28-03:00', end='2021-09-03 03:30:44
-03:00', empty=False),
 TimeFrame(start='2021-09-03 03:33:01-03:00', end='2021-09-03 07:00:16
-03:00', empty=False),
 TimeFrame(start='2021-09-03 07:02:00-03:00', end='2021-09-03 09:10:40
-03:00', empty=False),
 TimeFrame(start='2021-09-03 09:11:15-03:00', end='2021-09-03 12:30:51
-03:00', empty=False),
 TimeFrame(start='2021-09-03 12:31:41-03:00', end='2021-09-03 12:32:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 12:32:51-03:00', end='2021-09-03 12:38:56
-03:00', empty=False),
 TimeFrame(start='2021-09-03 12:40:26-03:00', end='2021-09-03 12:40:46
-03:00', empty=False),
 TimeFrame(start='2021-09-03 12:49:16-03:00', end='2021-09-03 13:50:43
-03:00', empty=False),
 TimeFrame(start='2021-09-03 13:51:33-03:00', end='2021-09-03 13:54:58
-03:00', empty=False),
 TimeFrame(start='2021-09-03 13:57:53-03:00', end='2021-09-03 14:05:53
-03:00', empty=False),
 TimeFrame(start='2021-09-03 14:07:13-03:00', end='2021-09-03 14:13:28
-03:00', empty=False),
 TimeFrame(start='2021-09-03 14:16:53-03:00', end='2021-09-03 14:25:18
-03:00', empty=False),
 TimeFrame(start='2021-09-03 14:26:03-03:00', end='2021-09-03 17:29:03
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:29:53-03:00', end='2021-09-03 17:36:03
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:37:56-03:00', end='2021-09-03 17:39:01
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:39:51-03:00', end='2021-09-03 17:41:01
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:42:06-03:00', end='2021-09-03 17:43:01
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:44:11-03:00', end='2021-09-03 17:45:01
```

```
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:45:56-03:00', end='2021-09-03 17:46:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:46:56-03:00', end='2021-09-03 17:47:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:47:56-03:00', end='2021-09-03 17:49:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:50:01-03:00', end='2021-09-03 17:51:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:52:01-03:00', end='2021-09-03 17:53:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:54:01-03:00', end='2021-09-03 17:55:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 17:56:11-03:00', end='2021-09-03 17:59:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:00:06-03:00', end='2021-09-03 18:01:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:02:11-03:00', end='2021-09-03 18:03:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:03:41-03:00', end='2021-09-03 18:05:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:05:46-03:00', end='2021-09-03 18:06:56
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:08:26-03:00', end='2021-09-03 18:09:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:09:56-03:00', end='2021-09-03 18:10:16
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:11:01-03:00', end='2021-09-03 18:11:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:12:01-03:00', end='2021-09-03 18:13:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:14:01-03:00', end='2021-09-03 18:15:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:16:16-03:00', end='2021-09-03 18:17:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:18:01-03:00', end='2021-09-03 18:19:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:20:01-03:00', end='2021-09-03 18:21:11
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:22:01-03:00', end='2021-09-03 18:23:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:24:11-03:00', end='2021-09-03 18:25:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:26:56-03:00', end='2021-09-03 18:27:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:28:16-03:00', end='2021-09-03 18:29:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:30:16-03:00', end='2021-09-03 18:31:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:31:46-03:00', end='2021-09-03 18:33:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:33:51-03:00', end='2021-09-03 18:35:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:35:46-03:00', end='2021-09-03 18:37:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:37:46-03:00', end='2021-09-03 18:39:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 18:48:11-03:00', end='2021-09-03 21:07:06
-03:00', empty=False),
 TimeFrame(start='2021-09-03 21:07:51-03:00', end='2021-09-03 21:08:06
-03:00', empty=False),
```

```
 TimeFrame(start='2021-09-03 21:12:01-03:00', end='2021-09-04 00:18:16
-03:00', empty=False),
 TimeFrame(start='2021-09-04 00:19:36-03:00', end='2021-09-04 00:30:21
-03:00', empty=False),
 TimeFrame(start='2021-09-04 00:31:26-03:00', end='2021-09-04 00:38:11
-03:00', empty=False),
 TimeFrame(start='2021-09-04 00:41:16-03:00', end='2021-09-04 04:05:06
-03:00', empty=False),
 TimeFrame(start='2021-09-04 04:06:56-03:00', end='2021-09-04 07:35:26
-03:00', empty=False),
 TimeFrame(start='2021-09-04 07:39:16-03:00', end='2021-09-04 10:03:31
-03:00', empty=False),
 TimeFrame(start='2021-09-04 10:05:16-03:00', end='2021-09-04 10:08:56
-03:00', empty=False),
 TimeFrame(start='2021-09-04 10:11:36-03:00', end='2021-09-04 10:59:31
-03:00', empty=False),
 TimeFrame(start='2021-09-04 11:01:21-03:00', end='2021-09-04 12:47:11
-03:00', empty=False),
 TimeFrame(start='2021-09-04 12:48:16-03:00', end='2021-09-04 14:28:56
-03:00', empty=False),
 TimeFrame(start='2021-09-04 14:30:21-03:00', end='2021-09-04 15:01:56
-03:00', empty=False),
 TimeFrame(start='2021-09-04 15:02:51-03:00', end='2021-09-04 15:07:36
-03:00', empty=False),
 TimeFrame(start='2021-09-04 15:09:01-03:00', end='2021-09-04 21:43:41
-03:00', empty=False),
 TimeFrame(start='2021-09-04 21:45:11-03:00', end='2021-09-04 21:45:41
-03:00', empty=False),
 TimeFrame(start='2021-09-04 21:46:16-03:00', end='2021-09-05 01:13:17
-03:00', empty=False),
 TimeFrame(start='2021-09-05 01:16:12-03:00', end='2021-09-05 03:12:57
-03:00', empty=False),
 TimeFrame(start='2021-09-05 03:13:32-03:00', end='2021-09-05 08:12:22
-03:00', empty=False),
 TimeFrame(start='2021-09-05 08:14:44-03:00', end='2021-09-05 08:15:29
-03:00', empty=False),
 TimeFrame(start='2021-09-05 08:16:04-03:00', end='2021-09-05 08:16:34
-03:00', empty=False),
 TimeFrame(start='2021-09-05 08:18:54-03:00', end='2021-09-05 10:48:09
-03:00', empty=False),
 TimeFrame(start='2021-09-05 10:49:29-03:00', end='2021-09-05 11:35:59
-03:00', empty=False),
 TimeFrame(start='2021-09-05 11:37:09-03:00', end='2021-09-05 11:37:54
-03:00', empty=False),
 TimeFrame(start='2021-09-05 11:40:09-03:00', end='2021-09-05 11:44:24
-03:00', empty=False),
 TimeFrame(start='2021-09-05 11:45:04-03:00', end='2021-09-05 11:48:49
-03:00', empty=False),
 TimeFrame(start='2021-09-05 11:50:04-03:00', end='2021-09-05 11:57:49
-03:00', empty=False),
 TimeFrame(start='2021-09-05 11:59:24-03:00', end='2021-09-05 12:12:54
-03:00', empty=False),
 TimeFrame(start='2021-09-05 12:15:04-03:00', end='2021-09-05 12:22:34
-03:00', empty=False),
 TimeFrame(start='2021-09-05 12:24:49-03:00', end='2021-09-05 17:08:24
-03:00', empty=False),
 TimeFrame(start='2021-09-05 17:09:59-03:00', end='2021-09-05 20:03:15
-03:00', empty=False),
 TimeFrame(start='2021-09-05 20:04:20-03:00', end='2021-09-05 20:23:10
-03:00', empty=False),
 TimeFrame(start='2021-09-05 20:26:00-03:00', end='2021-09-05 20:28:25
```

```
-03:00', empty=False),
 TimeFrame(start='2021-09-05 20:29:05-03:00', end='2021-09-05 20:37:40
-03:00', empty=False),
 TimeFrame(start='2021-09-05 20:40:00-03:00', end='2021-09-05 21:28:19
-03:00', empty=False),
 TimeFrame(start='2021-09-05 21:30:39-03:00', end='2021-09-05 22:27:50
-03:00', empty=False),
 TimeFrame(start='2021-09-05 22:30:30-03:00', end='2021-09-05 23:03:00
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:05:35-03:00', end='2021-09-05 23:08:00
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:10:50-03:00', end='2021-09-05 23:13:25
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:15:25-03:00', end='2021-09-05 23:17:55
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:20:40-03:00', end='2021-09-05 23:23:25
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:25:15-03:00', end='2021-09-05 23:27:45
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:28:55-03:00', end='2021-09-05 23:37:25
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:39:40-03:00', end='2021-09-05 23:48:15
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:49:00-03:00', end='2021-09-05 23:52:35
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:54:50-03:00', end='2021-09-05 23:57:20
-03:00', empty=False),
 TimeFrame(start='2021-09-05 23:58:20-03:00', end='2021-09-06 05:30:44
-03:00', empty=False),
 TimeFrame(start='2021-09-06 05:33:59-03:00', end='2021-09-06 07:01:05
-03:00', empty=False),
 TimeFrame(start='2021-09-06 07:02:45-03:00', end='2021-09-06 07:56:00
-03:00', empty=False)]
```

Autocorrelation Plot

In [28]:

```
# from pandas.plotting import autocorrelation_plot
# elec.mains().plot_autocorrelation();
```

Dataframe de correlação dos aparelhos

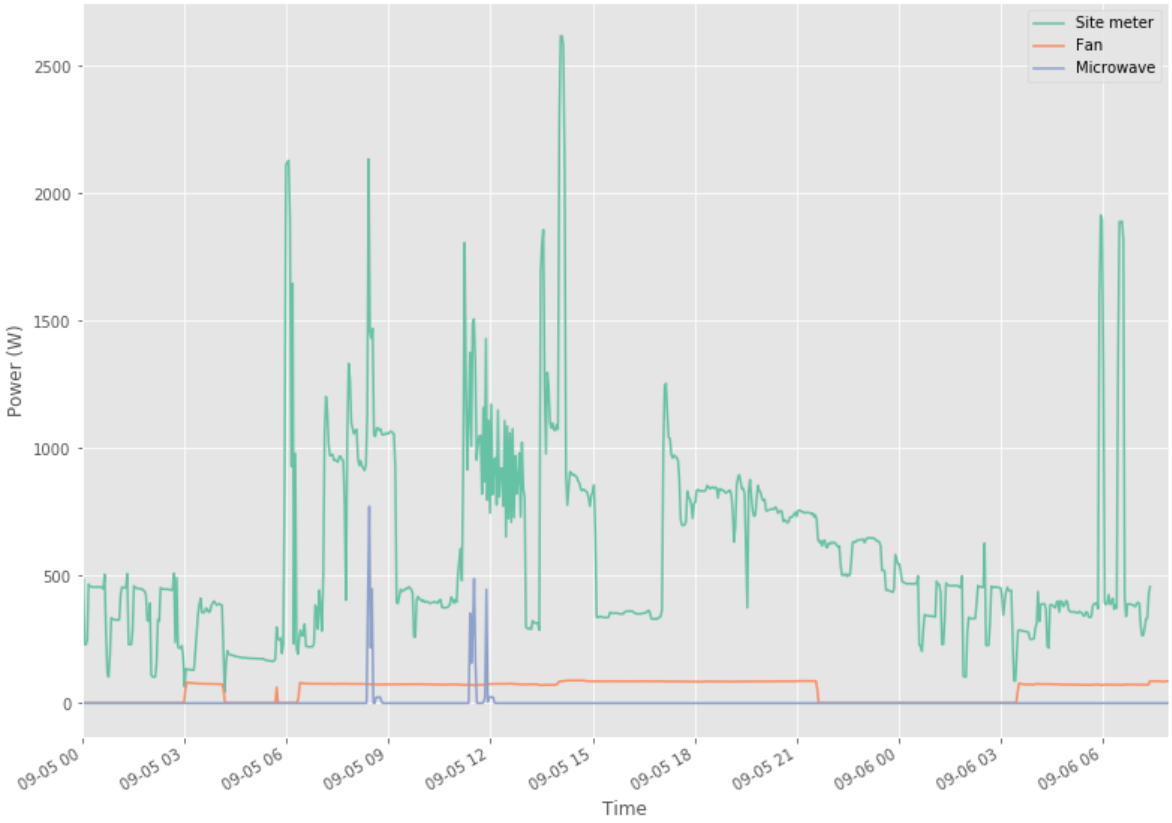
In [29]:

```
# correlation_df = elec.pairwise_correlation()
# correlation_df
```

Traçar dados submedidos em um 1 dia

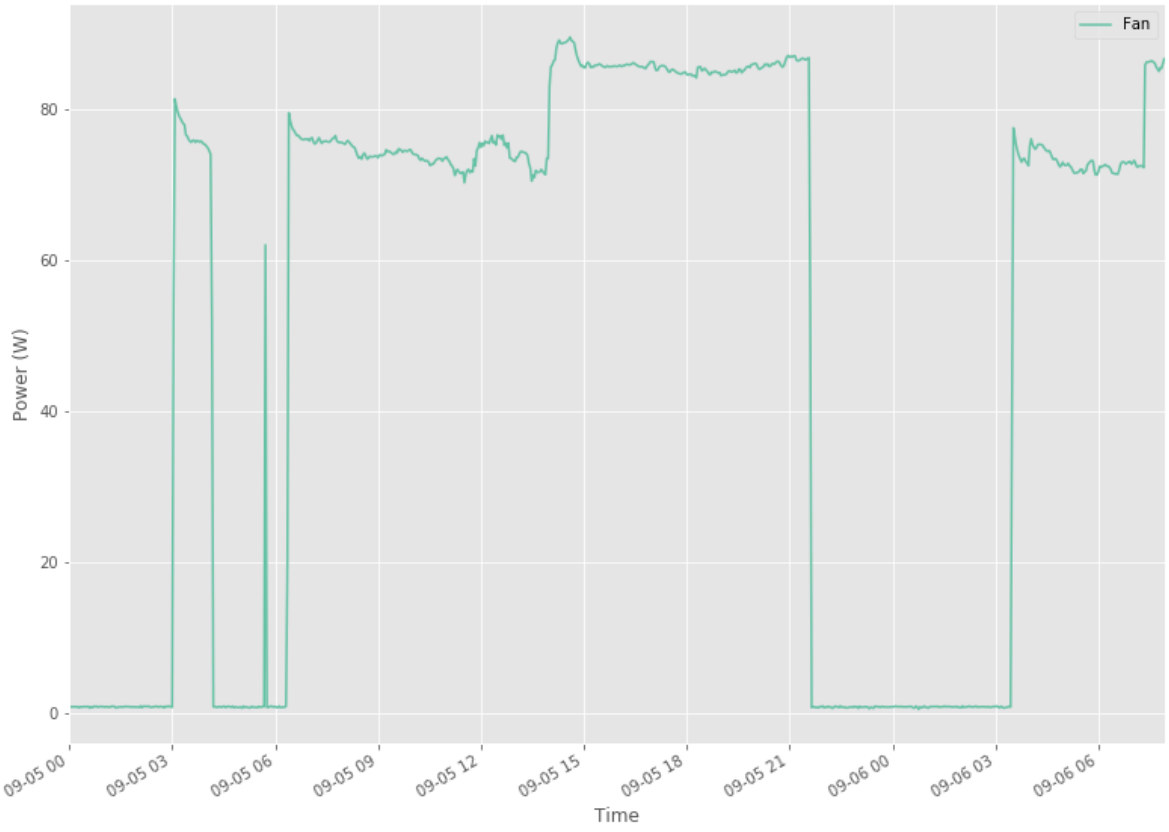
In [30]:

```
hb.set_window(start='2021-09-05', end='2021-09-07')
elec.plot();
plt.xlabel("Time");
```



In [31]:

```
# hb.set_window(start='2021-09-05 00:00:00', end='2021-09-06 23:59:59')
hb.set_window(start='2021-09-05', end='2021-09-07')

# elec['microwave'].plot()
elec['fan'].plot()
plt.xlabel("Time");
```



Importamos os algoritmos que desejamos executar os experimentos:

Mean: Mean Algorithm

Hart's Algorithm

CO: Combinatorial Optimization

Discriminative Sparse Coding

Additive Factorial Hidden Markov Model

Additive Factorial Hidden Markov Model with Signal Aggregate Constraints

DSC: Discriminative Sparse Coding

RNN: Long short-term memory - LSTM

DAE: Denoising Auto Encoder

Seq2Point*

Seq2Seq

WindowGRU/Online GRU: Similar a LSTM, mas usa Gated Recurrent Unit (GRU)

ELM

In [32]:

```
from nilmtk.disaggregate import Mean,CO,Hart85
# from nilmtk_contrib.disaggregate import AFHMM,AFHMM_SAC,DSC,RNN,Seq2Point,Seq2Seq
from nilmtk_contrib.disaggregate import RNN,Seq2Point,WindowGRU
```

```
Using TensorFlow backend.
```

Em seguida, inserimos os valores para os diferentes parâmetros no dicionário. Como precisamos de vários aparelhos, inserimos os nomes de todos os aparelhos necessários no parâmetro 'appliances'.

Métricas: https://github.com/nilmtk/nilmtk/blob/master/nilmtk/losses.py (https://github.com/nilmtk/nilmtk/blob/master/nilmtk/losses.py)

Error: https://github.com/nilmtk/nilmtk-contrib/issues/56 (https://github.com/nilmtk/nilmtk-contrib/issues/56)

In [37]:

```
d = {
  'power': {
    'mains': ['active'],
    'appliance': ['active']
#     'mains': ['active', 'frequency', 'power factor', 'current', 'voltage'],
#     'appliance': ['active', 'apparent', 'reactive', 'power factor', 'current', 'v
  },
  'sample_rate': 1,
  'display_predictions': True,
  'appliances': ['microwave', 'fan'],
  'methods': {
      'Mean':Mean({}),
#       "CO":CO({}),
      #'Hart85':Hart85({}),
      'RNN':RNN({'n_epochs':50,'batch_size':1024}),
      'Seq2Point':Seq2Point({'n_epochs':50,'batch_size':1024})
      #'Seq2Seq':Seq2Seq({'n_epochs':50,'batch_size':1024}),
      #'WindowGRU':WindowGRU({'n_epochs':30,'batch_size':1024})
  },
 'train': {
    'datasets': {
      'hb': {
        'path': 'teste17.h5',
        'buildings': {
              1: {
                'start_time': '2021-09-02',
                'end_time': '2021-09-04'
              },

        }
      },

    }
  },

  'test': {
    'datasets': {
      'REDD': {
        'path': 'teste17.h5',
        'buildings': {
              1: {
                    'start_time': '2021-09-05',
                    'end_time': '2021-09-07'
            }
        }
      }
    },
    'metrics':['rmse', 'mae', 'relative_error', 'r2score', 'nde', 'nep', 'f1score']
  }
}
```

raiz do erro quadrático médio (RMSE) e o erro médio absoluto (MAE)

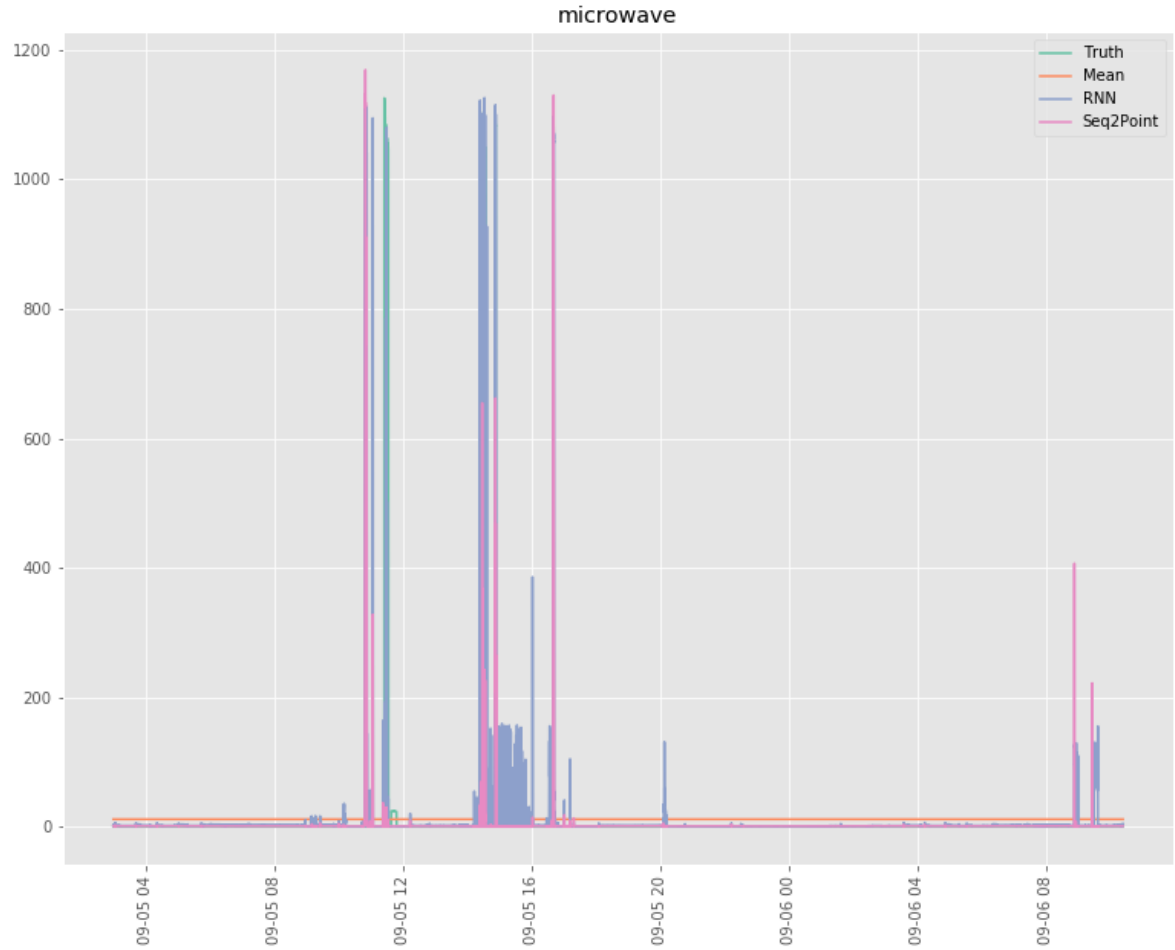
Quanto menor o seu valor, melhor é o modelo, já que a previsão se mostra mais próxima ao valor real. Comparando as duas métricas têm se que o RMSE penaliza desvios grandes, enquanto o MAE tem pesos iguais para todos os desvios.

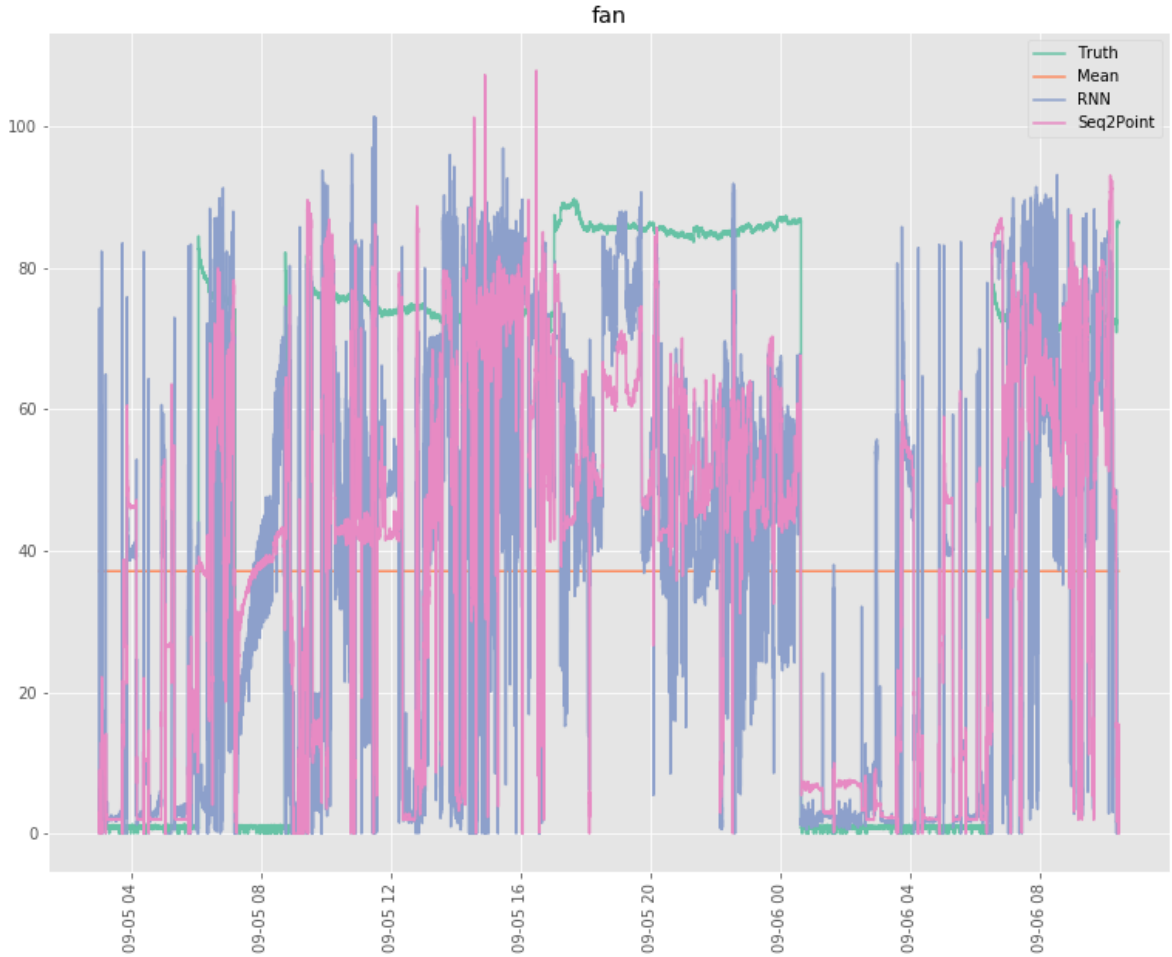
We can observe the prediction vs. truth graphs in the above cell. The accuracy metrics can be accessed using the following commands:

In [38]:

```
api_res = API(d)
```

```
Joint Testing for all algorithms
Loading data for  REDD  dataset
Dropping missing values
Generating predictions for : Mean
Generating predictions for : RNN
Generating predictions for : Seq2Point
............  rmse  ..............
                Mean        RNN  Seq2Point
microwave  71.093207  80.159929  83.607102
fan        39.968880  31.650862  29.777788
............  mae  ..............
                Mean        RNN  Seq2Point
microwave  15.880199   9.717364   8.270511
fan        39.604229  22.672867  22.667080
............  relative_error  ..............
                Mean       RNN  Seq2Point
microwave  1.297325  2.461882   2.714603
fan        1.039114  1.610404   1.420564
............  r2score  ..............
                Mean       RNN  Seq2Point
microwave -0.007444 -0.280794  -0.393320
fan       -0.142445  0.283589   0.365873
............  nde  ..............
                Mean       RNN  Seq2Point
microwave  1.001093  1.128766   1.177307
fan        0.630193  0.499042   0.469509
............  nep  ..............
                Mean       RNN  Seq2Point
microwave  3.095858  1.894408   1.612343
fan        0.773116  0.442598   0.442485
............  f1score  ..............
                Mean       RNN  Seq2Point
microwave  0.041279  0.242044   0.150158
fan        0.786693  0.871793   0.867027
```


microwave
Truth
Mean
RNN
Seq2Point
1200
1000
800
600
400
200
0
09-05 04
09-05 08
09-05 12
09-05 16
09-05 20
09-06 00
09-06 04
09-06 08


fan
Truth
Mean
RNN
Seq2Point
100
80
60
40
20
0
09-05 04
09-05 08
09-05 12
09-05 16
09-05 20
09-06 00
09-06 04
09-06 08

In [35]:

```
import numpy as np
import pandas as pd

vals = np.concatenate([np.expand_dims(df.values,axis=2) for df in d.errors],axis=2)


cols = d.errors[0].columns
indexes = d.errors[0].index


mean = np.mean(vals,axis=2)
std = np.std(vals,axis=2)
print ('\n\n')
print ("Mean")
print (pd.DataFrame(mean,index=indexes,columns=cols))
print ('\n\n')
print ("Standard Deviation")
print (pd.DataFrame(std,index=indexes,columns=cols))
```

```
---------------------------------------------------------------------------
-----
AttributeError                            Traceback (most recent call
 last)
/tmp/ipykernel_29632/4176508112.py in <module>
      2 import pandas as pd
      3
----> 4 vals = np.concatenate([np.expand_dims(df.values,axis=2) for df
in d.errors],axis=2)
      5
      6

AttributeError: 'dict' object has no attribute 'errors'
```

In [ ]: