In [ ]:

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

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

Carregando bibliotecas...

In [1]:

```
!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: 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: pandas>=0.23 in ./miniconda3/envs/nilm_
0.4.3/lib/python3.7/site-packages (from seaborn) (0.25.3)
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: 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: 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: 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: 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 [ ]:

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

In [ ]:

```
# 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 [2]:

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

from nilmtk import DataSet
from nilmtk.utils import print_dict

hb = DataSet('teste17.h5')
# redd = DataSet('redd.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 [3]:

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

Out[3]:

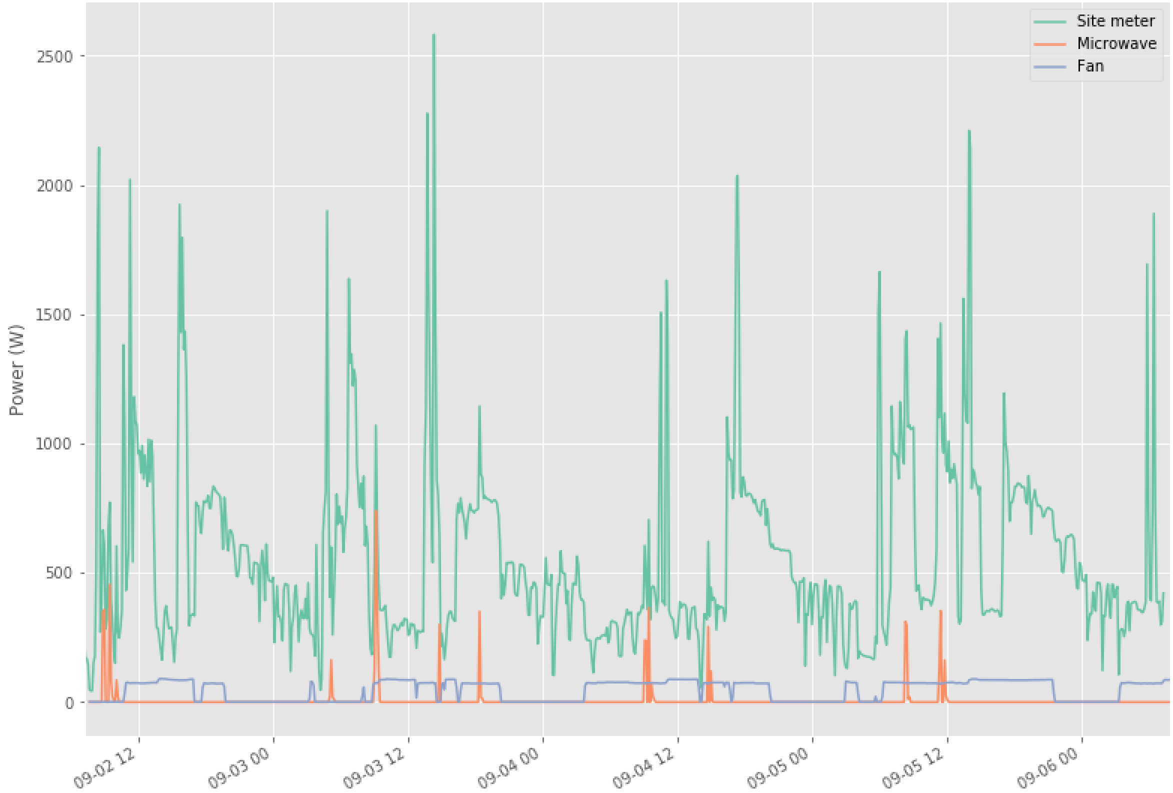
```
2021-09-02 07:14:34.515000-03:00    167.199997
2021-09-02 07:14:35.014000-03:00    167.199997
2021-09-02 07:14:35.513000-03:00    167.199997
2021-09-02 07:14:36.013000-03:00    167.199997
2021-09-02 07:14:36.527000-03:00    166.899994
Name: (power, active), dtype: float32
```

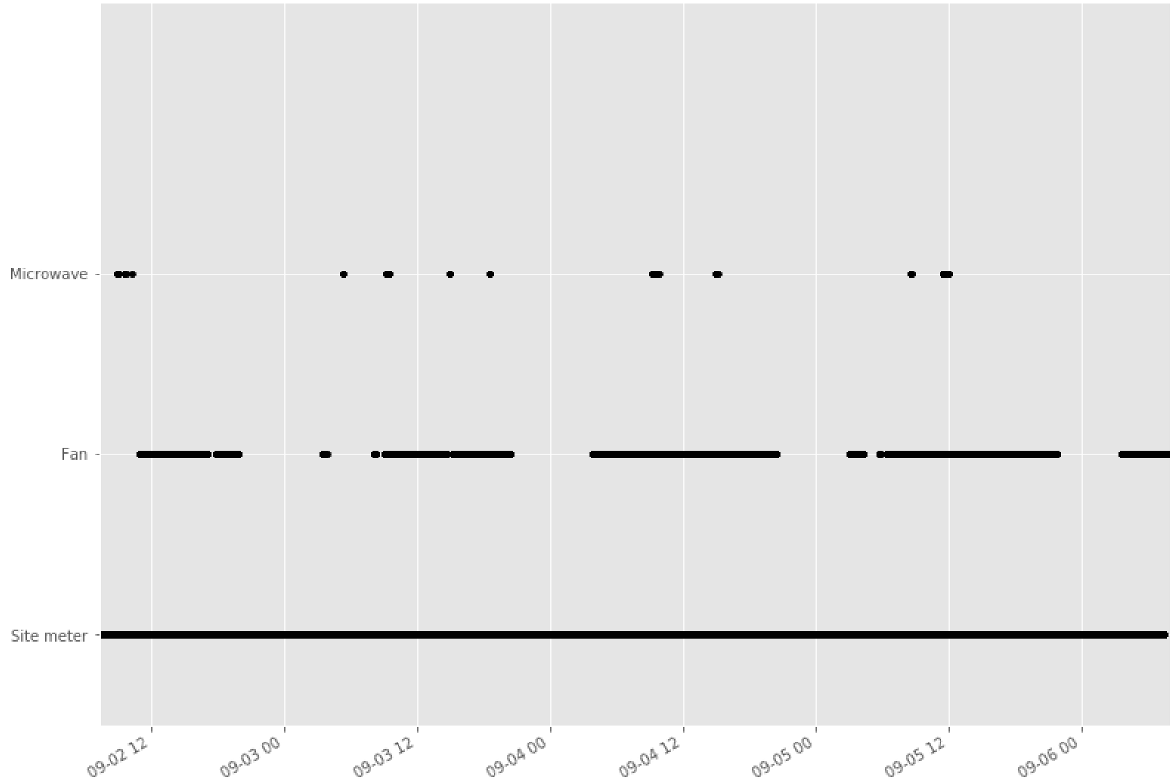
In [4]:

```
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[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f667cabb6d0>
```





Dados

Proporção de energia submedida

In [5]:

```
elec.proportion_of_energy_submetered()
```

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

Out[5]:

```
0.09288249528613458
```

Total Energy

In [6]:

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

Out[6]:

```
active    53.946047
dtype: float64
```

Energy per submeter

In [7]:

```
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[7]:

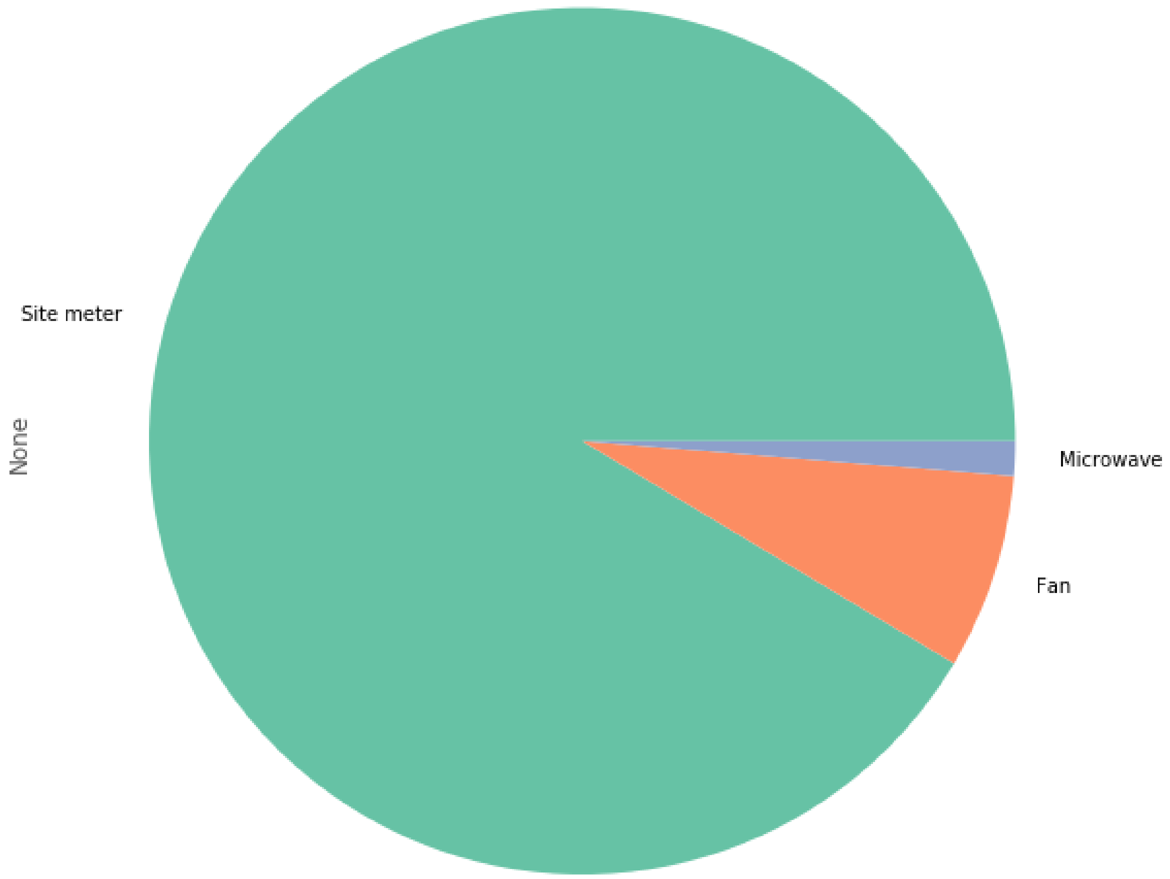
	(2, 1, HB)	(3, 1, HB)
active	4.298278	0.757815
apparent	NaN	NaN
reactive	NaN	NaN

Plot fraction of energy consumption of each appliance

In [5]:

```
# 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 [6]:

```
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[6]:

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

In [11]:

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

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

next(meter1.load()).tail()
```

Out[11]:

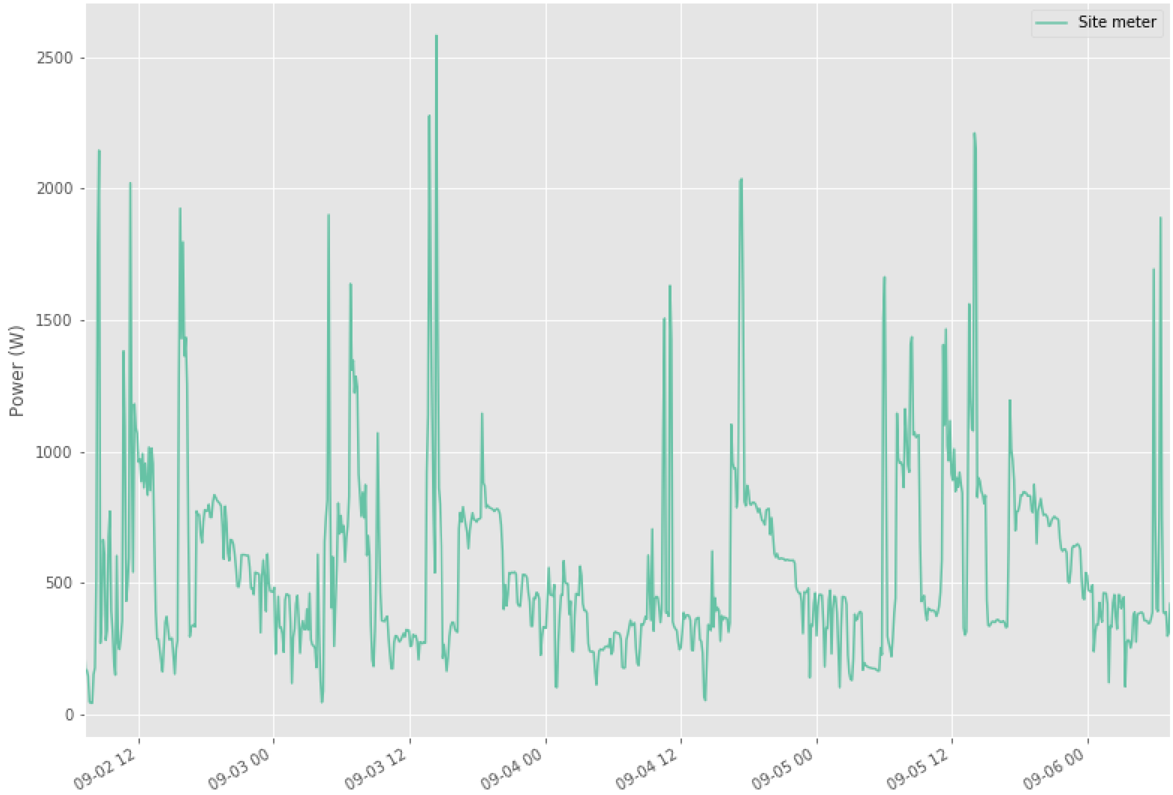
physical_quantity	voltage	frequency	power	power factor	current
type			active		
2021-09-06 07:24:13.545000-03:00	221.300003	60.0	458.399994	0.91	2.286
2021-09-06 07:24:14.060000-03:00	221.300003	60.0	458.399994	0.91	2.286
2021-09-06 07:24:14.559000-03:00	221.500000	60.0	458.299988	0.91	2.282
2021-09-06 07:24:15.042000-03:00	221.500000	60.0	458.299988	0.91	2.282
2021-09-06 07:24:15.557000-03:00	221.600006	60.0	459.000000	0.91	2.286

In [12]:

```
meter1.plot()
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f667a0fdd90>
```



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 [13]:

```
meter1.dropout_rate()
```

Out[13]:

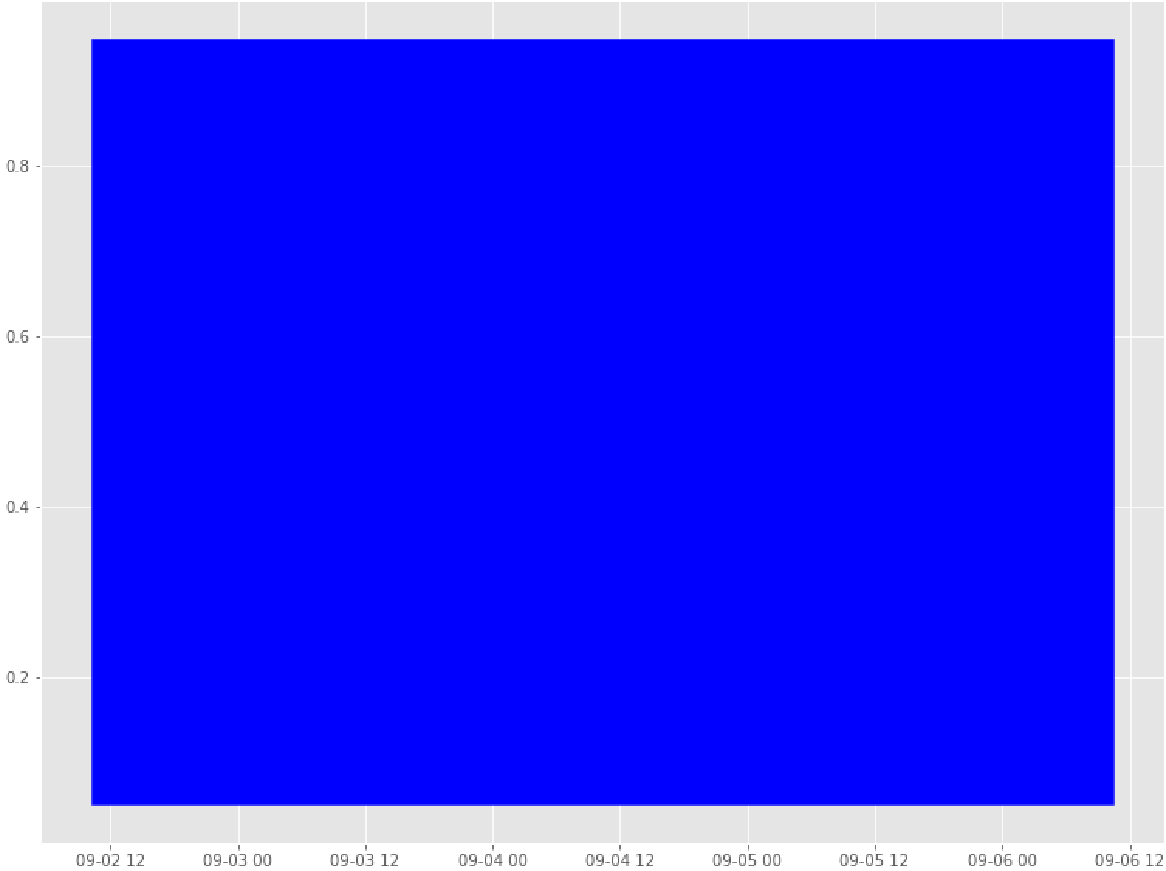
```
0.0002946431279545747
```

In [14]:

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

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f667b21ed10>
```



In [15]:

```
good_sections.combined()
```

Out[15]:

```
[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 [16]:

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

Out[16]:

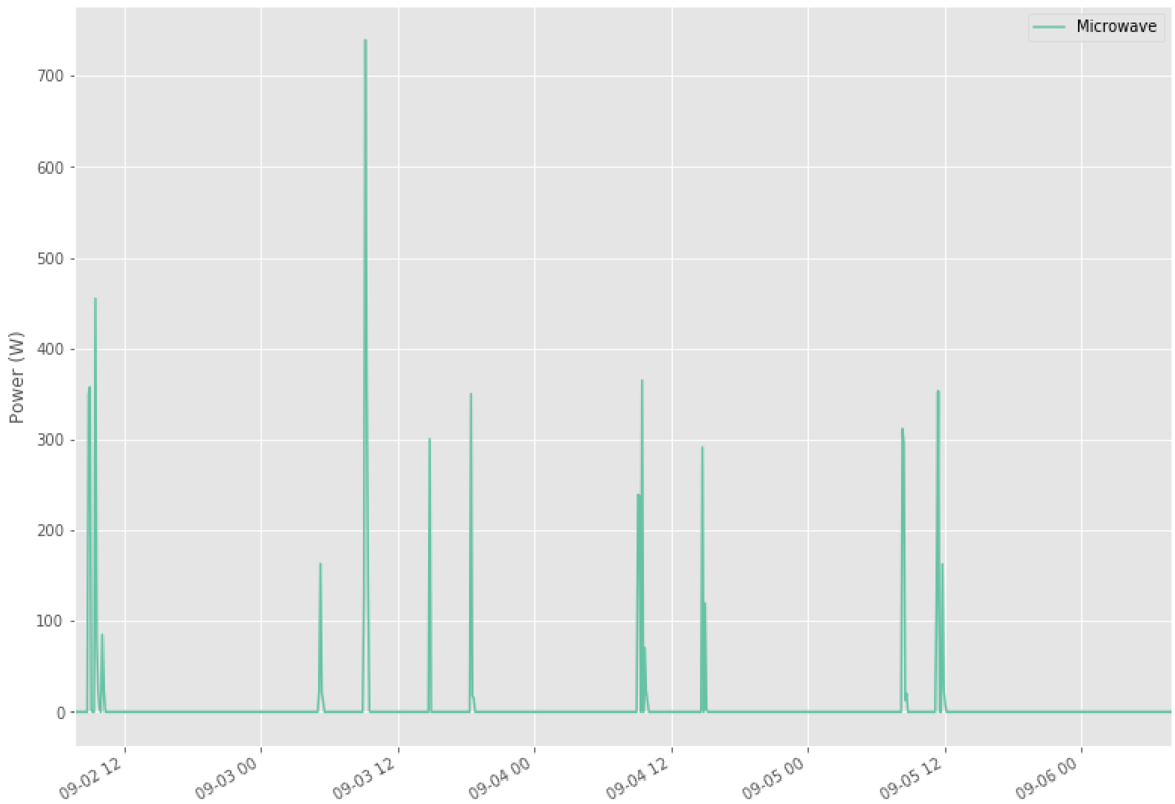
physical_quantity	voltage	power		power factor	power	current
type		apparent	active		reactive	
2021-09-02 07:47:51-03:00	221.882004	0.0	0.0	0.0	0.0	0.0
2021-09-02 07:47:56-03:00	221.882004	0.0	0.0	0.0	0.0	0.0
2021-09-02 07:48:01-03:00	222.406006	0.0	0.0	0.0	0.0	0.0
2021-09-02 07:48:06-03:00	222.143997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

In [17]:

```
microwave.plot()
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f667a10a910>
```



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 [18]:

```
microwave.dropout_rate()
```

Out[18]:

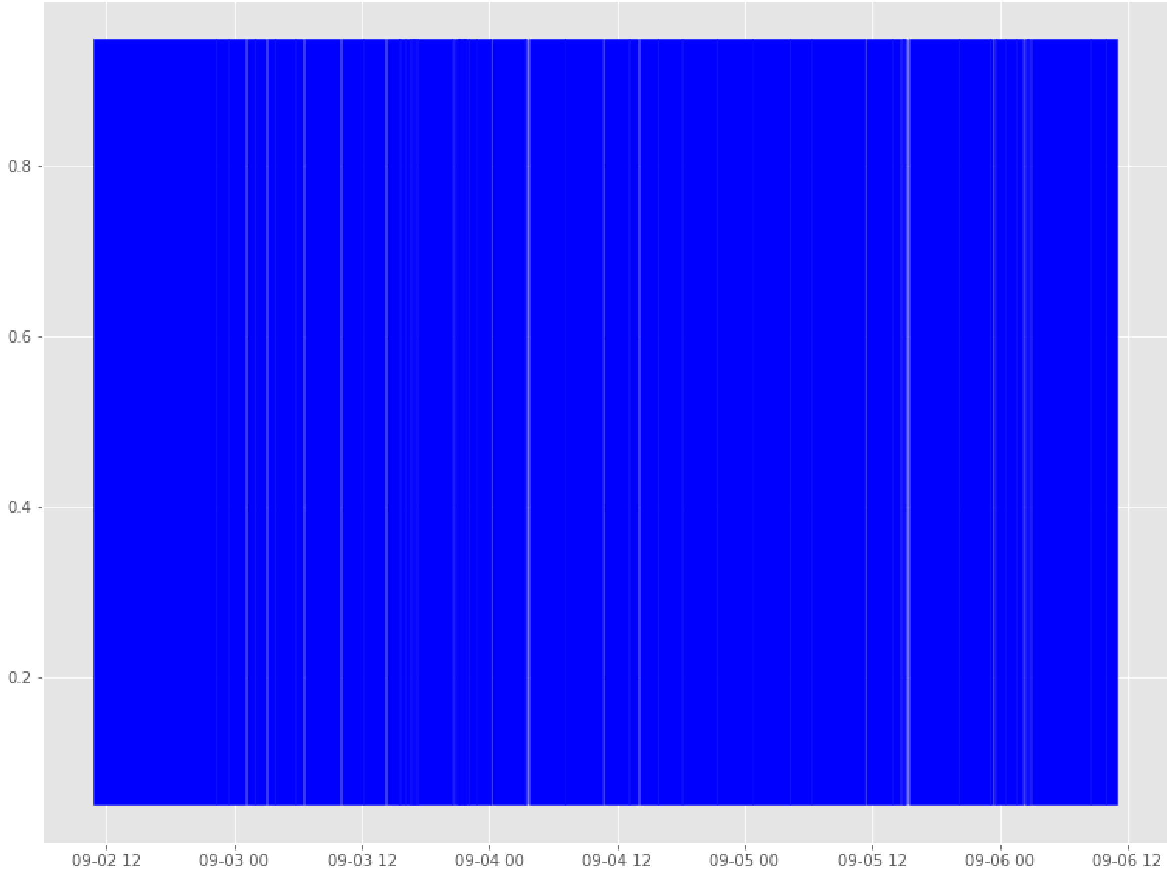
0.001913041828182637

In [19]:

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

Out[19]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f66785fc890>



In [20]:

```
good_sections.combined()
```

Out[20]:

```
[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 [21]:

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

Out[21]:

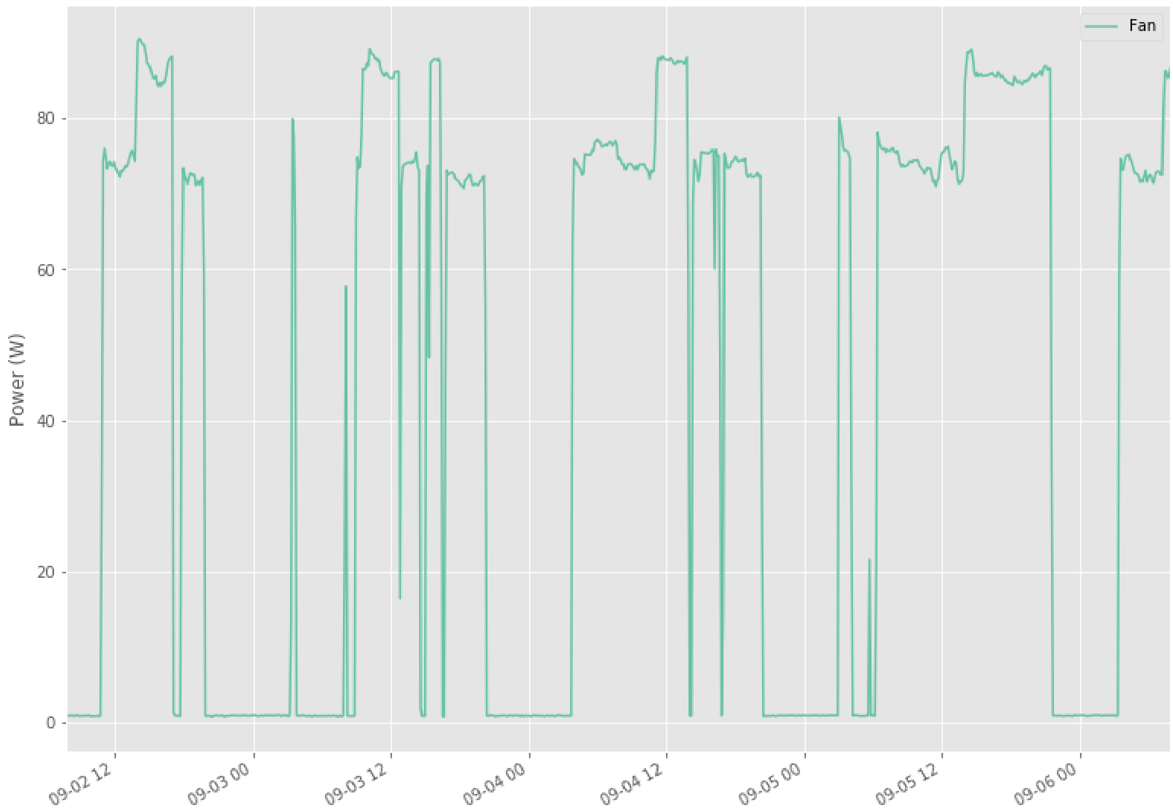
physical_quantity	voltage	power		power factor	power	current
type		apparent	active		reactive	
2021-09-02 07:47:51-03:00	222.287003	18.454000	0.767	0.04	18.400000	0.083
2021-09-02 07:47:56-03:00	222.546997	31.761999	1.091	0.03	31.700001	0.143
2021-09-02 07:48:01-03:00	222.028000	20.479000	1.091	0.05	20.400000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

In [22]:

```
fan.plot()
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f667845a050>
```

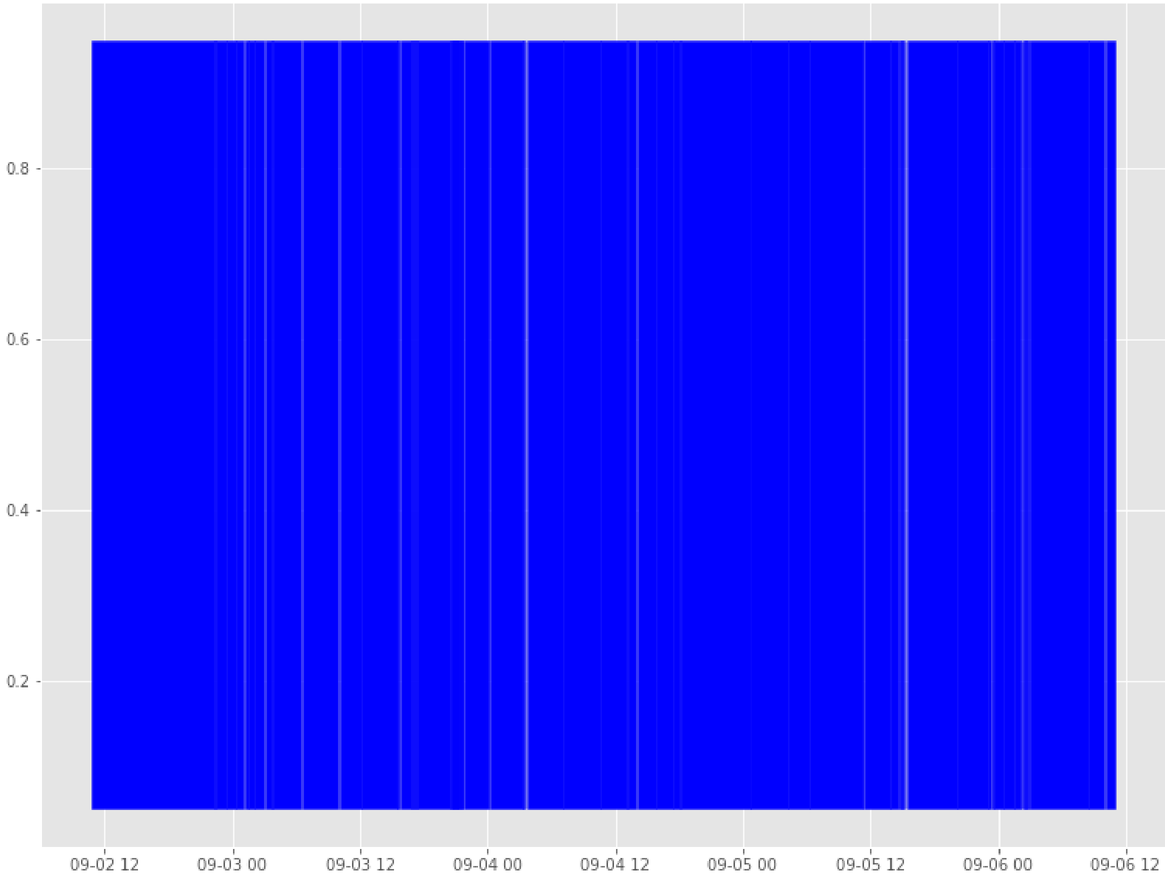


In [23]:

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

Out[23]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66783d2910>
```



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 [24]:

```
fan.dropout_rate()
```

Out[24]:

```
0.002014694526278486
```

In [25]:

```
good_sections.combined()
```

Out[25]:

```
[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 [26]:

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

Dataframe de correlação dos aparelhos

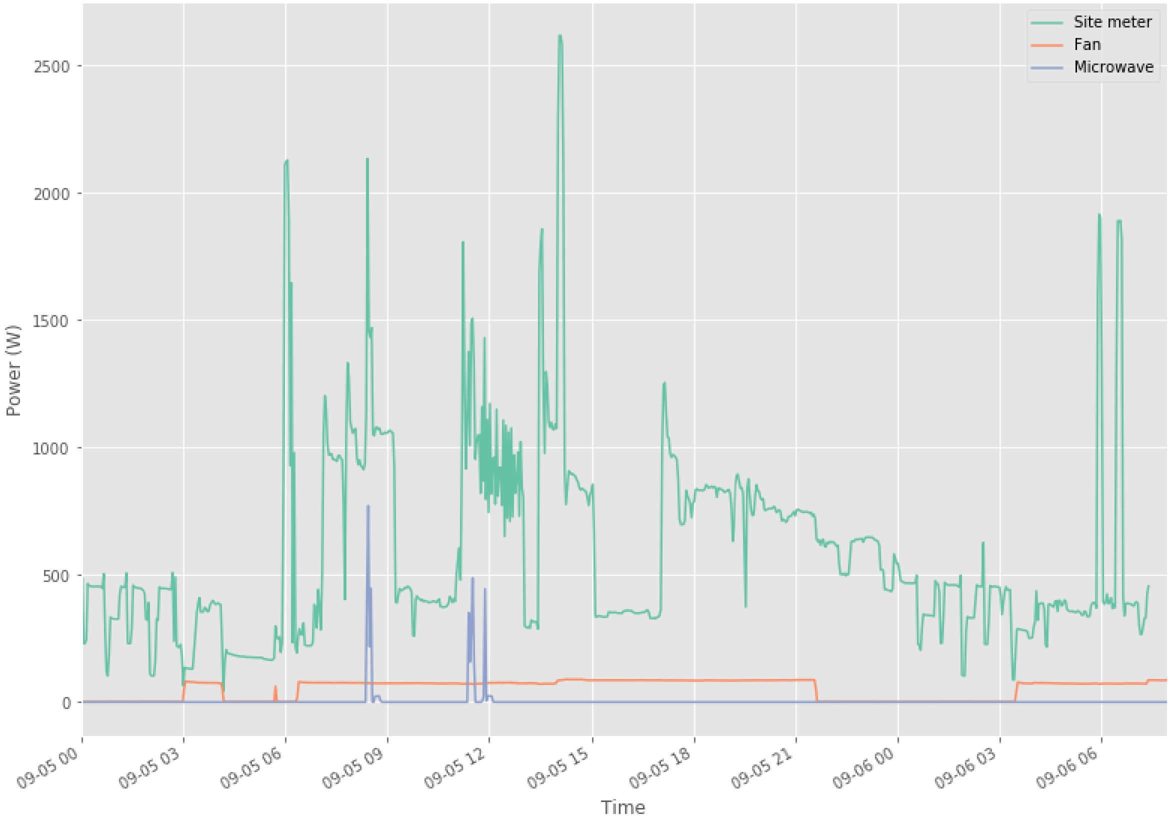
In [27]:

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

Traçar dados submedidos em um 1 dia

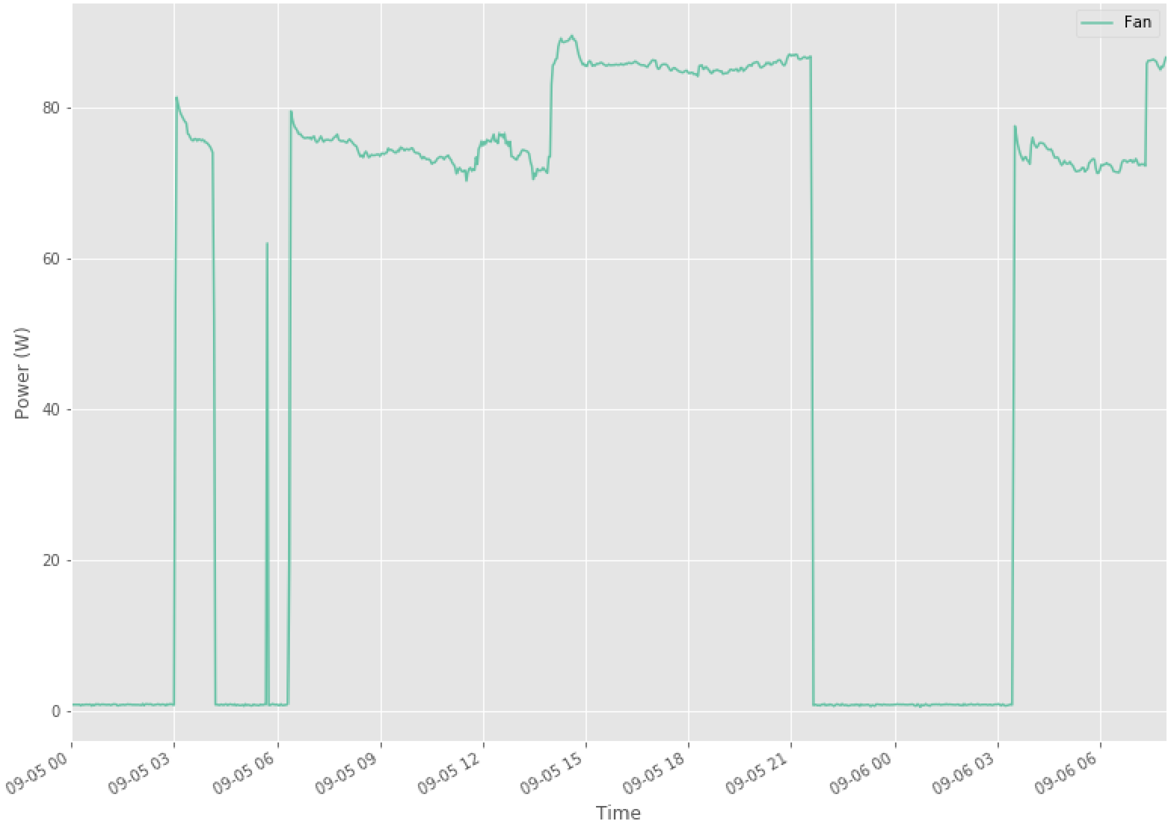
In [28]:

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



In [29]:

```
# 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 [2]:

```
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,Seq2Seq,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)](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)](https://github.com/nilmtk/nilmtk-contrib/issues/56)

In [11]:

```
d = {
  'power': {
    'mains': ['active'],
    'appliance': ['apparent']
#     'mains': ['active', 'frequency', 'power factor', 'current', 'voltage'],
#     'appliance': ['active', 'apparent', 'reactive', 'power factor', 'current', 'v
  },
  'artificial_aggregate': False,
  'sample_rate': 5,
  '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': {
      'hb': {
        '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.

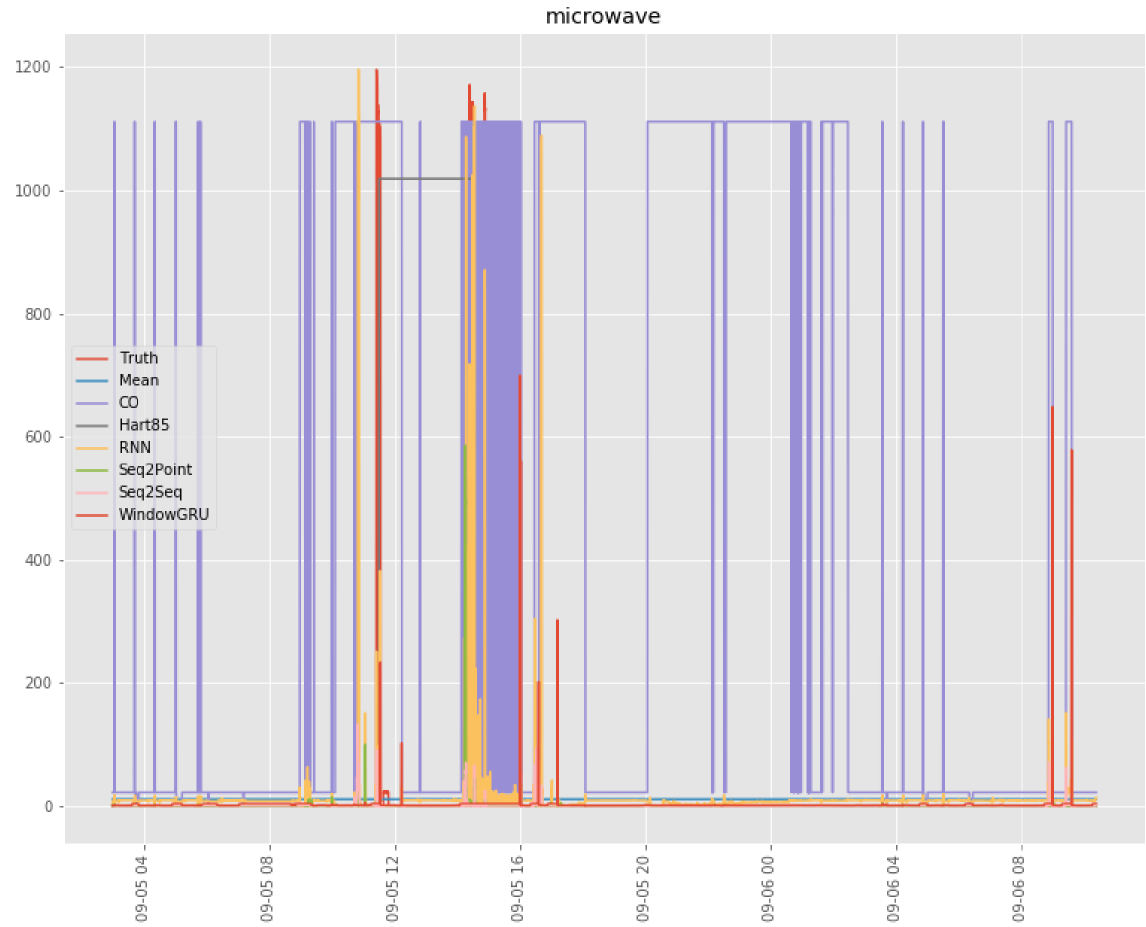
We can observe the prediction vs. truth graphs in the above cell. The accuracy metrics can be accessed using the following commands:

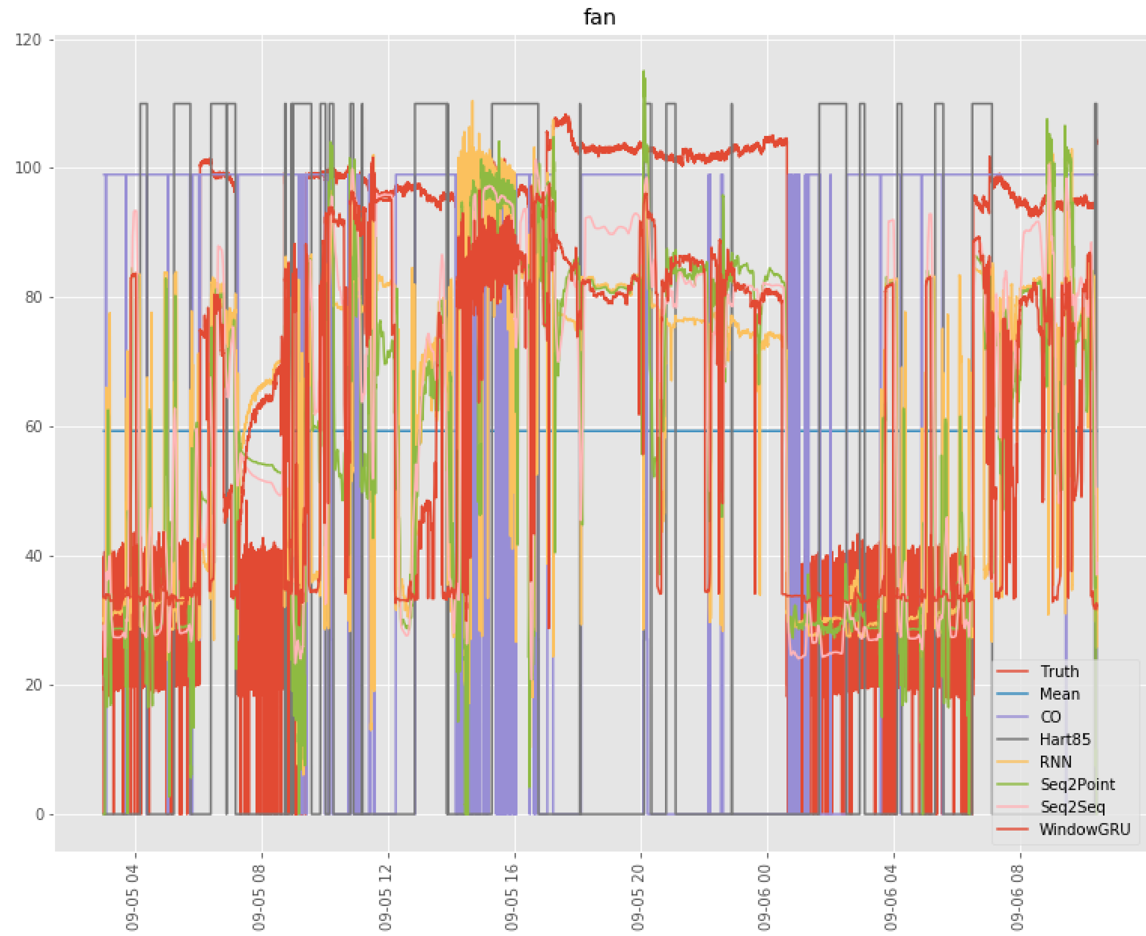
In [12]:

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

```
Joint Testing for all algorithms
Loading data for  hb  dataset
Dropping missing values
Generating predictions for : Mean
Generating predictions for : CO
...............CO disaggregate_chunk running.............
Generating predictions for : Hart85ave'
Finding Edges, please wait ...
Edge detection complete.
Creating transition frame ...
Transition frame created.
Creating states frame ...
States frame created.
Finished.
Generating predictions for : RNN
Generating predictions for : Seq2Point
Generating predictions for : Seq2Seq
Generating predictions for : WindowGRU
............  rmse  ..............
                Mean          CO      Hart85        RNN  Seq2Point
Seq2Seq  \
microwave  75.115949  652.180214  320.255745  90.794464  75.997086
74.058798
fan        37.900123   64.104072   74.820997  28.767712  28.246002
26.304125

           WindowGRU
microwave  75.727233
fan        29.800886
............  mae  ..............
                Mean          CO      Hart85        RNN  Seq2Point
Seq2Seq  \
microwave  16.872429  397.170288  100.987816  16.946432   6.496365
6.317146
fan        37.287132   48.720425   64.062813  22.901848  22.636448
20.133600

           WindowGRU
microwave   7.985390
fan        22.876396
............  relative_error  ..............
               Mean         CO     Hart85       RNN  Seq2Point   Seq
2Seq  \
microwave  1.302848   0.948665   2.739608  2.624932   1.525452  2.22
2900
fan        0.617993  25.188070  55.413353  0.419135   0.568237  0.36
8461

           WindowGRU
microwave   1.562649
fan         0.458613
............  r2score  ..............
               Mean         CO     Hart85       RNN  Seq2Point   Seq
2Seq  \
microwave -0.007559 -74.952354 -17.314701 -0.472057  -0.031335  0.02
0602
fan       -0.156958  -2.309845  -3.509033  0.333428   0.357386  0.44
```

```
2706

           WindowGRU
microwave  -0.024024
fan         0.284689
............  nde  ..............
               Mean        CO    Hart85       RNN  Seq2Point   Seq2S
eq  \
microwave  1.001126  8.692091  4.268286  1.210085   1.012870  0.9870
37
fan        0.466032  0.788244  0.920023  0.353737   0.347322  0.3234
44

           WindowGRU
microwave   1.009273
fan         0.366441
............  nep  ..............
               Mean         CO     Hart85       RNN  Seq2Point   Seq
2Seq  \
microwave  3.099038  72.950119  18.548904  3.112630   1.193218  1.16
0300
fan        0.508723   0.664713   0.874035  0.312459   0.308838  0.27
4691

           WindowGRU
microwave   1.466714
fan         0.312112
............  f1score  ..............
               Mean        CO    Hart85       RNN  Seq2Point   Seq2S
eq  \
microwave  0.041814  0.042490  0.197968  0.040739   0.083333  0.1330
28
fan        0.992823  0.831522  0.388553  0.992731   0.988951  0.9928
23

           WindowGRU
microwave   0.030476
fan         0.992823
```


microwave
Truth
Mean
CO
Hart85
RNN
Seq2Point
Seq2Seq
WindowGRU
0
200
400
600
800
1000
1200
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
CO
Hart85
RNN
Seq2Point
Seq2Seq
WindowGRU
0
20
40
60
80
100
120
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 [ ]:

```

```

In [ ]:

```
import numpy as np
import pandas as pd

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


cols = api_res.errors[0].columns
indexes = api_res.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))
```

In [ ]:

```

```