HeatHack Data Book

Contents

This website shows plots of temperature and relative humidity data for HeatHack's participating venues. To view your data, you'll need your venue number. We don't use venue names, for security.

We are still adding venues who recently had their sensors shipped to them, and getting the automatic transfer of data organised - please bear with us.

We will add more types of plots when we have more data coming in.

Data won't be transferred here from ThingSpeak or from our data email instantly. It currently happens once a week but we'll make that more frequent. If you want to see whether you have data coming in right now and you have an internet-connected sensor, you can look at your recent data on ThingSpeak:

- ThingSpeak data plots

If you prefer to plot your data in Excel, you can download your full data here:

- Data CSV files

If you any issues, for instance, with the timestamp format, let us know - our Excel handles it natively, but yours may not.

Temperature Plots (all venues)

Choose your venue from the dropdown menu. Then you can use the slider below the plot to explore your data. You can make the data "window" as small or as large as you want, and also slide the entire window to the left or the right. You can also choose a window size to be an hour, a day, a week, or a year using the buttons above the plot.

These plots work best on a large screen. If you can't see the right edge of the plot, try using your browser zoom features to get the full plot. On many browsers, ctrl-+ (holding the control button and the + sign at the same time) will zoom in and ctrl-minus (the control button with the minus sign) will zoom out. Zooming is also available from the browser menus. We are looking at what we can do to facilitate use on other screen sizes.

> **Bug work-around**
>
> If you change venue sometimes you will get the new data superimposed on the old data rather than replacing it. If this happens, reload the page.

```
# Imports
import ipywidgets as widgets
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from IPython.display import display


# Get the possible data venues
venuekeysfile = "venue-keys.csv"
dfVenueKeys = pd.read_csv(venuekeysfile)
dfVenueKeys = dfVenueKeys.dropna(subset=['channel_id'])



#give user option to select their venue
venueDropdown = widgets.Dropdown(
    options=dfVenueKeys['venue_id'],
    value=dfVenueKeys['venue_id'][0],
    description='Venue ID:',
    disabled=False,
)


container = widgets.HBox(children=[venueDropdown])

print(venueDropdown.value)

#Retrieve the venue and begin graphing
dfCollatedDataSet = pd.DataFrame(columns=['timestamp', 'entry_id', 'temperature',
'rh', 'voltage', 'venue_id'])
for index, venueSensorDetails in dfVenueKeys.iterrows():

    sensorMacOfSelection = venueSensorDetails['sensor_MAC']
    venueOfSelection = str(venueSensorDetails['venue_id'])
    dfTempDataSet = pd.read_csv('deviceData/'+ "venue_" + venueOfSelection +
"_with_device_" + sensorMacOfSelection + '.csv' )
    dfTempDataSet['timestamp'] = pd.to_datetime(dfTempDataSet['timestamp'])
    dfTempDataSet['venue_id'] = venueSensorDetails['venue_id']

    dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
    dfCollatedDataSet['timestamp'] =
pd.to_datetime(dfCollatedDataSet['timestamp'])
    print('Loading data for venue: ', venueSensorDetails['venue_id'])


print('Check')
dfCollatedDataSet.sample(6)




# Assign an empty figure widget with two traces
trace0 = go.Scatter(customdata=dfCollatedDataSet[dfCollatedDataSet['venue_id'] ==
0],
                    y=dfCollatedDataSet['temperature'],
                    x = dfCollatedDataSet['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='Temperature',
                    )

# trace1 = go.Scatter(customdata=dfCollatedDataSet[dfCollatedDataSet['venue_id']
== 0],
#                     y=dfCollatedDataSet['rh'],
#                     x = dfCollatedDataSet['timestamp'],
#                     mode='lines',
#                     hoverinfo='all',
#                     name='Relative Humidity',

# )


g = go.FigureWidget(data=trace0,
                    #data=[trace0, trace1],
                    layout = go.Layout(
                        yaxis=dict(range=[0,0])
                    ))


print("Job Done")
```

```
1
Loading data for venue:  1
Loading data for venue:  2
Loading data for venue:  3
Loading data for venue:  4
Loading data for venue:  6
Loading data for venue:  7
Loading data for venue:  8
```

```
/tmp/ipykernel_1873/3318412257.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1873/3318412257.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1873/3318412257.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1873/3318412257.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1873/3318412257.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1873/3318412257.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1873/3318412257.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In [1], line 35
     33 sensorMacOfSelection = venueSensorDetails['sensor_MAC']
     34 venueOfSelection = str(venueSensorDetails['venue_id'])
---> 35 dfTempDataSet = pd.read_csv('deviceData/'+ "venue_" + venueOfSelection +
"_with_device_" + sensorMacOfSelection + '.csv' )
     36 dfTempDataSet['timestamp'] = pd.to_datetime(dfTempDataSet['timestamp'])
     37 dfTempDataSet['venue_id'] = venueSensorDetails['venue_id']

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/util/_decorators.py:211, in deprecate_kwarg.
<locals>._deprecate_kwarg.<locals>.wrapper(*args, **kwargs)
    209     else:
    210         kwargs[new_arg_name] = new_arg_value
--> 211 return func(*args, **kwargs)

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/util/_decorators.py:331, in deprecate_nonkeyword_arguments.
<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/io/parsers/readers.py:950, in read_csv(filepath_or_buffer, sep,
delimiter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols,
dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows,
skipfooter, nrows, na_values, keep_default_na, na_filter, verbose,
skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser,
dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal,
lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding,
encoding_errors, dialect, error_bad_lines, warn_bad_lines, on_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision, storage_options)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (...)
    946     defaults={"delimiter": ","},
    947 )
    948 kwds.update(kwds_defaults)
--> 950 return _read(filepath_or_buffer, kwds)

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/io/parsers/readers.py:605, in _read(filepath_or_buffer, kwds)
    602 _validate_names(kwds.get("names", None))
    604 # Create the parser.
--> 605 parser = TextFileReader(filepath_or_buffer, **kwds)
    607 if chunksize or iterator:
    608     return parser

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/io/parsers/readers.py:1442, in TextFileReader.__init__(self, f,
engine, **kwds)
   1439     self.options["has_index_names"] = kwds["has_index_names"]
   1441 self.handles: IOHandles | None = None
-> 1442 self._engine = self._make_engine(f, self.engine)

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/io/parsers/readers.py:1735, in TextFileReader._make_engine(self,
f, engine)
   1733     if "b" not in mode:
   1734         mode += "b"
-> 1735 self.handles = get_handle(
   1736     f,
   1737     mode,
   1738     encoding=self.options.get("encoding", None),
   1739     compression=self.options.get("compression", None),
   1740     memory_map=self.options.get("memory_map", False),
   1741     is_text=is_text,
   1742     errors=self.options.get("encoding_errors", "strict"),
   1743     storage_options=self.options.get("storage_options", None),
   1744 )
   1745 assert self.handles is not None
   1746 f = self.handles.handle

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/io/common.py:856, in get_handle(path_or_buf, mode, encoding,
compression, memory_map, is_text, errors, storage_options)
    851 elif isinstance(handle, str):
    852     # Check whether the filename is to be opened in binary mode.
    853     # Binary mode does not support 'encoding' and 'newline'.
    854     if ioargs.encoding and "b" not in ioargs.mode:
```

```
    855         # Encoding
--> 856         handle = open(
    857             handle,
    858             ioargs.mode,
    859             encoding=ioargs.encoding,
    860             errors=errors,
    861             newline="",
    862         )
    863     else:
    864         # Binary mode
    865         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory:
'deviceData/venue_9_with_device_E8DB84DF97A9.csv'
```

```
updatemenu = []
buttons = []


# button with one option for each dataframe
for index, venue in dfVenueKeys.iterrows():
    buttons.append(dict(method='update',
                        label='Venue ' + str(venue['venue_id']),
                        visible=True,
                        args=[{'y':
[dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
['temperature'].values], #,

#dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
['rh'].values],
                               'x':
[dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
['timestamp'].values],
                               'type':'scatter',
                           #  'name': 'Temperature',

                               },
                               {
                                   'title.text': 'Temperature for Venue = ' +
str(venue['venue_id']),
                                   'title.font.color': 'green',
                                    ## KDs original for plotting RH and temp on
same plot
                                    #'yaxis.range':
[-5,dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
[['temperature', 'rh']].max().max()+5],
                                    ## adjusting for data range, temp only
                                    #'yaxis.range':
[dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
['temperature'].min()-5,dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['ve
nue_id']]['temperature'].max()+5],
                                    # fixed, because we get rogue readings on
cheap devices
                                   'yaxis.range': [0,30],
                                   'yaxis.title.text': 'Temperature',
                                   'xaxis.title.text': 'Timestamp'
                               },
                               ],
                        )
                )



# some adjustments to the updatemenus
updatemenu = []
your_menu = dict()
updatemenu.append(your_menu)

updatemenu[0]['buttons'] = buttons
updatemenu[0]['direction'] = 'down'
updatemenu[0]['showactive'] = True
# updatemenu[0]['active'] = 0



fig = go.Figure(g)
# add dropdown menus to the figure
fig.update_layout(showlegend=True,
              updatemenus=updatemenu,
              autosize = True,
              title= "Please select a venue to see your data.",
              width=1000,
              height=500,
)

fig.update_layout(
    hovermode='x unified',
    hoverlabel=dict(
        bgcolor="white",
        # font_size=16,
        font_family="Rockwell"
    )
)

# Add range slider
fig.update_layout(
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(
                     label="All",
```

```
                    step="all"
                    ),
                              dict(count=1,
                    label="Hour",
                    step="hour",
                    stepmode="todate"),
                dict(count=1,
                    label="Day",
                    step="day",
                    stepmode="backward"),
                dict(count=7,
                    label="Week",
                    step="day",
                    stepmode="backward"),
                dict(count=1,
                    label="Year",
                    step="year",
                    stepmode="backward")
            ])
        ),
        rangeslider=dict(
            visible=True,
        ),
        type="date"
    )
)


#fig.update_yaxes(range=[50, 60])


fig.add_hline(y=16, annotation_text='16C - usual minimum for children',
annotation_font_color="blue", line_color='red', layer='above', line_dash='dash')
# fig.update_yaxes(range = [-5, dfCollatedDataSet['temperature'].max()+5])
fig.show()
```

Relative Humidity Plots (all venues)

Choose your venue from the dropdown menu. Then you can use the slider below the plot to explore your data. You can make the data “window” as small or as large as you want, and also slide the entire window to the left or the right. You can also choose a window size to be an hour, a day, a week, or a year using the buttons above the plot.

These plots work best on a large screen. If you can’t see the right edge of the plot, try using your browser zoom features to get the full plot. On many browsers, ctrl-+ (holding the control button and the + sign at the same time) will zoom in and ctrl-minus (the control button with the minus sign) will zoom out. Zooming is also available from the browser menus. We are looking at what we can do to facilitate use on other screen sizes.

> **Bug work-around**
>
> If you change venue sometimes you will get the new data superimposed on the old data rather than replacing it. If this happens, reload the page.

```
# Imports
import ipywidgets as widgets
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
from IPython.display import display

# Get the possible data venues
venuekeysfile = "venue-keys.csv"
dfVenueKeys = pd.read_csv(venuekeysfile)
dfVenueKeys = dfVenueKeys.dropna(subset=['channel_id'])




#give user option to select their venue
venueDropdown = widgets.Dropdown(
    options=dfVenueKeys['venue_id'],
    value=dfVenueKeys['venue_id'][0],
    description='Venue ID:',
    disabled=False,
)


container = widgets.HBox(children=[venueDropdown])

print(venueDropdown.value)

#Retrieve the venue and begin graphing
dfCollatedDataSet = pd.DataFrame(columns=['timestamp', 'entry_id', 'temperature',
'rh', 'voltage', 'venue_id'])
for index, venueSensorDetails in dfVenueKeys.iterrows():

    sensorMacOfSelection = venueSensorDetails['sensor_MAC']
    venueOfSelection = str(venueSensorDetails['venue_id'])
    dfTempDataSet = pd.read_csv('deviceData/'+ "venue_" + venueOfSelection +
"_with_device_" + sensorMacOfSelection + '.csv' )
    dfTempDataSet['timestamp'] = pd.to_datetime(dfTempDataSet['timestamp'])
    dfTempDataSet['venue_id'] = venueSensorDetails['venue_id']

    dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
    dfCollatedDataSet['timestamp'] =
pd.to_datetime(dfCollatedDataSet['timestamp'])
    print('Loading data for venue: ', venueSensorDetails['venue_id'])


print('Check')
dfCollatedDataSet.sample(6)




# Assign an empty figure widget with two traces
trace0 = go.Scatter(customdata=dfCollatedDataSet[dfCollatedDataSet['venue_id'] ==
0],
                    y=dfCollatedDataSet['rh'],
                    x = dfCollatedDataSet['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='RH',
                    )

# trace1 = go.Scatter(customdata=dfCollatedDataSet[dfCollatedDataSet['venue_id']
== 0],
#                     y=dfCollatedDataSet['rh'],
#                     x = dfCollatedDataSet['timestamp'],
#                     mode='lines',
#                     hoverinfo='all',
#                     name='Relative Humidity',

# )


g = go.FigureWidget(data=trace0,
                    #data=[trace0, trace1],
                    layout = go.Layout(
                        yaxis=dict(range=[0,0])
                    ))


print("Job Done")
```

```
1
Loading data for venue:  1
Loading data for venue:  2
Loading data for venue:  3
Loading data for venue:  4
Loading data for venue:  6
Loading data for venue:  7
Loading data for venue:  8
```

```
/tmp/ipykernel_1852/3757134659.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1852/3757134659.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1852/3757134659.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1852/3757134659.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1852/3757134659.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1852/3757134659.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
/tmp/ipykernel_1852/3757134659.py:39: FutureWarning: The frame.append method is
deprecated and will be removed from pandas in a future version. Use pandas.concat
instead.
  dfCollatedDataSet = dfCollatedDataSet.append(dfTempDataSet, ignore_index=True)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In [1], line 35
     33 sensorMacOfSelection = venueSensorDetails['sensor_MAC']
     34 venueOfSelection = str(venueSensorDetails['venue_id'])
---> 35 dfTempDataSet = pd.read_csv('deviceData/'+ "venue_" + venueOfSelection +
"_with_device_" + sensorMacOfSelection + '.csv' )
     36 dfTempDataSet['timestamp'] = pd.to_datetime(dfTempDataSet['timestamp'])
     37 dfTempDataSet['venue_id'] = venueSensorDetails['venue_id']

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/util/_decorators.py:211, in deprecate_kwarg.
<locals>._deprecate_kwarg.<locals>.wrapper(*args, **kwargs)
    209     else:
    210         kwargs[new_arg_name] = new_arg_value
--> 211 return func(*args, **kwargs)

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/util/_decorators.py:331, in deprecate_nonkeyword_arguments.
<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    325 if len(args) > num_allow_args:
    326     warnings.warn(
    327         msg.format(arguments=_format_argument_list(allow_args)),
    328         FutureWarning,
    329         stacklevel=find_stack_level(),
    330     )
--> 331 return func(*args, **kwargs)

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/io/parsers/readers.py:950, in read_csv(filepath_or_buffer, sep,
delimiter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols,
dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows,
skipfooter, nrows, na_values, keep_default_na, na_filter, verbose,
skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser,
dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal,
lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding,
encoding_errors, dialect, error_bad_lines, warn_bad_lines, on_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision, storage_options)
    935 kwds_defaults = _refine_defaults_read(
    936     dialect,
    937     delimiter,
   (...)
    946     defaults={"delimiter": ","},
    947 )
    948 kwds.update(kwds_defaults)
--> 950 return _read(filepath_or_buffer, kwds)

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/io/parsers/readers.py:605, in _read(filepath_or_buffer, kwds)
    602 _validate_names(kwds.get("names", None))
    604 # Create the parser.
--> 605 parser = TextFileReader(filepath_or_buffer, **kwds)
    607 if chunksize or iterator:
    608     return parser

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/io/parsers/readers.py:1442, in TextFileReader.__init__(self, f,
engine, **kwds)
   1439     self.options["has_index_names"] = kwds["has_index_names"]
   1441 self.handles: IOHandles | None = None
-> 1442 self._engine = self._make_engine(f, self.engine)

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/io/parsers/readers.py:1735, in TextFileReader._make_engine(self,
f, engine)
   1733     if "b" not in mode:
   1734         mode += "b"
-> 1735 self.handles = get_handle(
   1736     f,
   1737     mode,
   1738     encoding=self.options.get("encoding", None),
   1739     compression=self.options.get("compression", None),
   1740     memory_map=self.options.get("memory_map", False),
   1741     is_text=is_text,
   1742     errors=self.options.get("encoding_errors", "strict"),
   1743     storage_options=self.options.get("storage_options", None),
   1744 )
   1745 assert self.handles is not None
   1746 f = self.handles.handle

File /opt/hostedtoolcache/Python/3.8.14/x64/lib/python3.8/site-
packages/pandas/io/common.py:856, in get_handle(path_or_buf, mode, encoding,
compression, memory_map, is_text, errors, storage_options)
    851 elif isinstance(handle, str):
    852     # Check whether the filename is to be opened in binary mode.
    853     # Binary mode does not support 'encoding' and 'newline'.
    854     if ioargs.encoding and "b" not in ioargs.mode:
```

```
    855         # Encoding
--> 856         handle = open(
    857             handle,
    858             ioargs.mode,
    859             encoding=ioargs.encoding,
    860             errors=errors,
    861             newline="",
    862         )
    863     else:
    864         # Binary mode
    865         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory:
'deviceData/venue_9_with_device_E8DB84DF97A9.csv'
```

```
updatemenu = []
buttons = []

# button with one option for each dataframe
for index, venue in dfVenueKeys.iterrows():
    buttons.append(dict(method='update',
                        label='Venue ' + str(venue['venue_id']),
                        visible=True,
                        args=[{'y':
[dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
['rh'].values], #,

#dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
['rh'].values],
                               'x':
[dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
['timestamp'].values],
                               'type':'scatter',
                            #  'name': 'Temperature',

                               },
                               {
                                   'title.text': 'RH for Venue = ' +
str(venue['venue_id']),
                                   'title.font.color': 'green',
                                   #'yaxis.range':
[-5,dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
[['temperature', 'rh']].max().max()+5],
                                   #'yaxis.range':
[dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
['rh'].min()-5,dfCollatedDataSet[dfCollatedDataSet['venue_id']==venue['venue_id']]
['rh'].max()+5],
                                   # fixed, because we get rogue readings on cheap
devices
                                   'yaxis.range': [0,100],
                                   'yaxis.title.text': 'RH',
                                   'xaxis.title.text': 'Timestamp'

                               },
                               ],
                    )
                )

# some adjustments to the updatemenus
updatemenu = []
your_menu = dict()
updatemenu.append(your_menu)

updatemenu[0]['buttons'] = buttons
updatemenu[0]['direction'] = 'down'
updatemenu[0]['showactive'] = True
# updatemenu[0]['active'] = 0



fig = go.Figure(g)
# add dropdown menus to the figure
fig.update_layout(showlegend=True,
              updatemenus=updatemenu,
              autosize = True,
              title= "Please select a venue to see your data.",
              width=1000,
              height=500,
)

fig.update_layout(
    hovermode='x unified',
    hoverlabel=dict(
        bgcolor="white",
        # font_size=16,
        font_family="Rockwell"
    )
)

# Add range slider
fig.update_layout(
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(
                     label="All",
                     step="all"
                     ),
                               dict(count=1,
                     label="Hour",
                     step="hour",
                     stepmode="todate"),
```

```
                dict(count=1,
                     label="Day",
                     step="day",
                     stepmode="backward"),
                dict(count=7,
                     label="Week",
                     step="day",
                     stepmode="backward"),
                dict(count=1,
                     label="Year",
                     step="year",
                     stepmode="backward")
            ])
        ),
        rangeslider=dict(
            visible=True
        ),
        type="date"
    )
)


#fig.update_yaxes(range=[50, 60])


#fig.add_hline(y=16, annotation_text='16C - usual minimum for children',
annotation_font_color="blue", line_color='red', layer='above', line_dash='dash')
# fig.update_yaxes(range = [-5, dfCollatedDataSet['temperature'].max()+5])
fig.show()
```

Comparing sensors - our calibration plots

These are cheap sensors, and before we send out the thermal monitors we try to remember to test them against a set of monitors plus a few commercial loggers that use a different kind of sensor. We do the testing in batches of around 10 thermal monitors, sometimes with Lascar EL-WIN-USB loggers alongside, and try to vary the temperature and relative humidity as best we can without dedicating lots of time to it. We don't send out any "duff" sensors, but RH readings always vary a bit, with a few sensors reading on the high or low side.

You can see how your sensor performed here.

Sensors 11-17

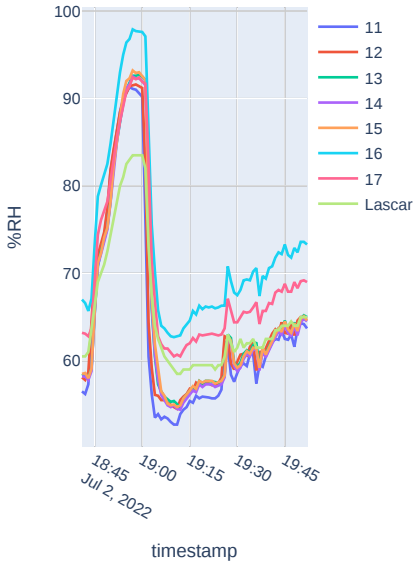
If you view the data, please read more about the process and its limitations. Lascars aren't designed to respond as quickly to change as the thermal monitor and our test conditions change more rapidly than buildings do. The RH readings are only intended to be accurate in the 20-80% RH range. In the plots, lines can be turned on and off by clicking on them in the legend. Also try double-clicking.

- HeatHack Blog Post about the tests

Relative Humidity

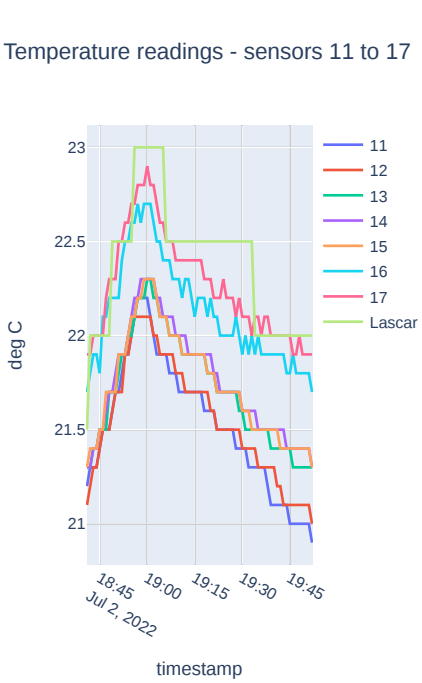
```
import plotly.graph_objects as go

import pandas as pd
df = pd.read_csv("sensors-11-17.csv")
df["timestamp"] = pd.to_datetime(df['timestamp'])

RHtrace11 = go.Scatter(customdata=df,
                    y=df['h11'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='11',
                    )
RHtrace12 = go.Scatter(customdata=df,
                    y=df['h12'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='12',
                    )

RHtrace13 = go.Scatter(customdata=df,
                    y=df['h13'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='13',
                    )
RHtrace14 = go.Scatter(customdata=df,
                    y=df['h14'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='14',
                    )

RHtrace15 = go.Scatter(customdata=df,
                    y=df['h15'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='15',
                    )
RHtrace16 = go.Scatter(customdata=df,
                    y=df['h16'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='16',
                    )
RHtrace17 = go.Scatter(customdata=df,
                    y=df['h17'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='17',
                    )
RHtraceLascar = go.Scatter(customdata=df,
                    y=df['Lascar-RH'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='Lascar',
                    )

g = go.FigureWidget(data=[RHtrace11, RHtrace12, RHtrace13, RHtrace14, RHtrace15,
RHtrace16, RHtrace17, RHtraceLascar])
g.layout.title = 'RH readings - sensors 11 to 17'
g.layout.xaxis.title= 'timestamp'
g.layout.yaxis.title = "%RH"


fig = go.Figure(g)

fig.show()
```

RH readings - sensors 11 to 17



Temperature

```
temptrace11 = go.Scatter(customdata=df,
                    y=df['t11'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='11',
                    )
temptrace12 = go.Scatter(customdata=df,
                    y=df['t12'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='12',
                    )

temptrace13 = go.Scatter(customdata=df,
                    y=df['t13'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='13',
                    )
temptrace14 = go.Scatter(customdata=df,
                    y=df['t14'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='14',
                    )

temptrace15 = go.Scatter(customdata=df,
                    y=df['t15'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='15',
                    )
temptrace16 = go.Scatter(customdata=df,
                    y=df['t16'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='16',
                    )
temptrace17 = go.Scatter(customdata=df,
                    y=df['t17'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='17',
                    )
temptraceLascar = go.Scatter(customdata=df,
                    y=df['Lascar-temp'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='Lascar',
                    )

g = go.FigureWidget(data=[temptrace11, temptrace12, temptrace13, temptrace14,
temptrace15, temptrace16, temptrace17,temptraceLascar])
g.layout.title = 'Temperature readings - sensors 11 to 17'
g.layout.xaxis.title= 'timestamp'
g.layout.yaxis.title = "deg C"


fig = go.Figure(g)

fig.show()
```



Sensors 18-24

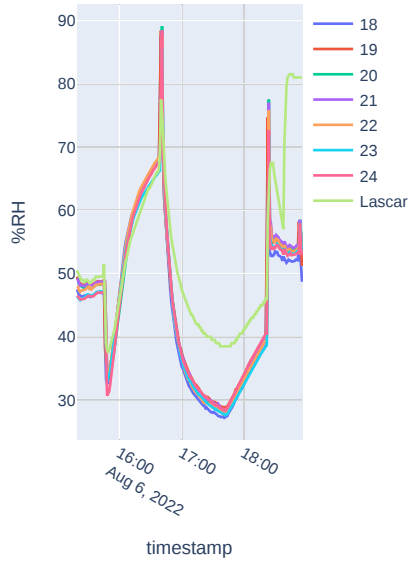
If you view the data, please read more about the process and its limitations. Lascars aren't designed to respond as quickly to change as the thermal monitor and our test conditions change more rapidly than buildings do. The RH readings are only intended to be accurate in the 20-80% RH range. In the plots, lines can be turned on and off by clicking on them in the legend. Also try double-clicking.

- HeatHack Blog Post about the tests

Relative Humidity

```
import plotly.graph_objects as go

import pandas as pd
df = pd.read_csv("sensors-18-24.csv")
df["timestamp"] = pd.to_datetime(df['timestamp'])

RHtrace18 = go.Scatter(customdata=df,
                    y=df['h18'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='18',
                    )
RHtrace19 = go.Scatter(customdata=df,
                    y=df['h19'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='19',
                    )

RHtrace20 = go.Scatter(customdata=df,
                    y=df['h20'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='20',
                    )
RHtrace21 = go.Scatter(customdata=df,
                    y=df['h21'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='21',
                    )

RHtrace22 = go.Scatter(customdata=df,
                    y=df['h22'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='22',
                    )
RHtrace23 = go.Scatter(customdata=df,
                    y=df['h23'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='23',
                    )
RHtrace24 = go.Scatter(customdata=df,
                    y=df['h24'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='24',
                    )
RHtraceLascar = go.Scatter(customdata=df,
                    y=df['Lascar-RH'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='Lascar',
                    )

g = go.FigureWidget(data=[RHtrace18, RHtrace19, RHtrace20, RHtrace21, RHtrace22,
RHtrace23, RHtrace24,RHtraceLascar])
g.layout.title = 'RH readings - sensors 18 to 24'
g.layout.xaxis.title= 'timestamp'
g.layout.yaxis.title = "%RH"


fig = go.Figure(g)

fig.show()
```

RH readings - sensors 18 to 24



Temperature

```
temptrace18 = go.Scatter(customdata=df,
                    y=df['t18'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='18',
                    )
temptrace19 = go.Scatter(customdata=df,
                    y=df['t19'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='19',
                    )

temptrace20 = go.Scatter(customdata=df,
                    y=df['t20'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='20',
                    )
temptrace21 = go.Scatter(customdata=df,
                    y=df['t21'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='21',
                    )

temptrace22 = go.Scatter(customdata=df,
                    y=df['t22'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='22',
                    )
temptrace23 = go.Scatter(customdata=df,
                    y=df['t23'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='23',
                    )
temptrace24 = go.Scatter(customdata=df,
                    y=df['t24'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='24',
                    )
temptraceLascar = go.Scatter(customdata=df,
                    y=df['Lascar-temp'],
                    x = df['timestamp'],
                    mode='lines',
                    hoverinfo='all',
                    name='Lascar',
                    )

g = go.FigureWidget(data=[temptrace18, temptrace19, temptrace20, temptrace21,
temptrace22, temptrace23, temptrace24,temptraceLascar])
g.layout.title = 'Temperature readings - sensors 18 to 24'
g.layout.xaxis.title= 'timestamp'
g.layout.yaxis.title = "deg C"


fig = go.Figure(g)

fig.show()
```

Temperature readings - sensors 18 to 24