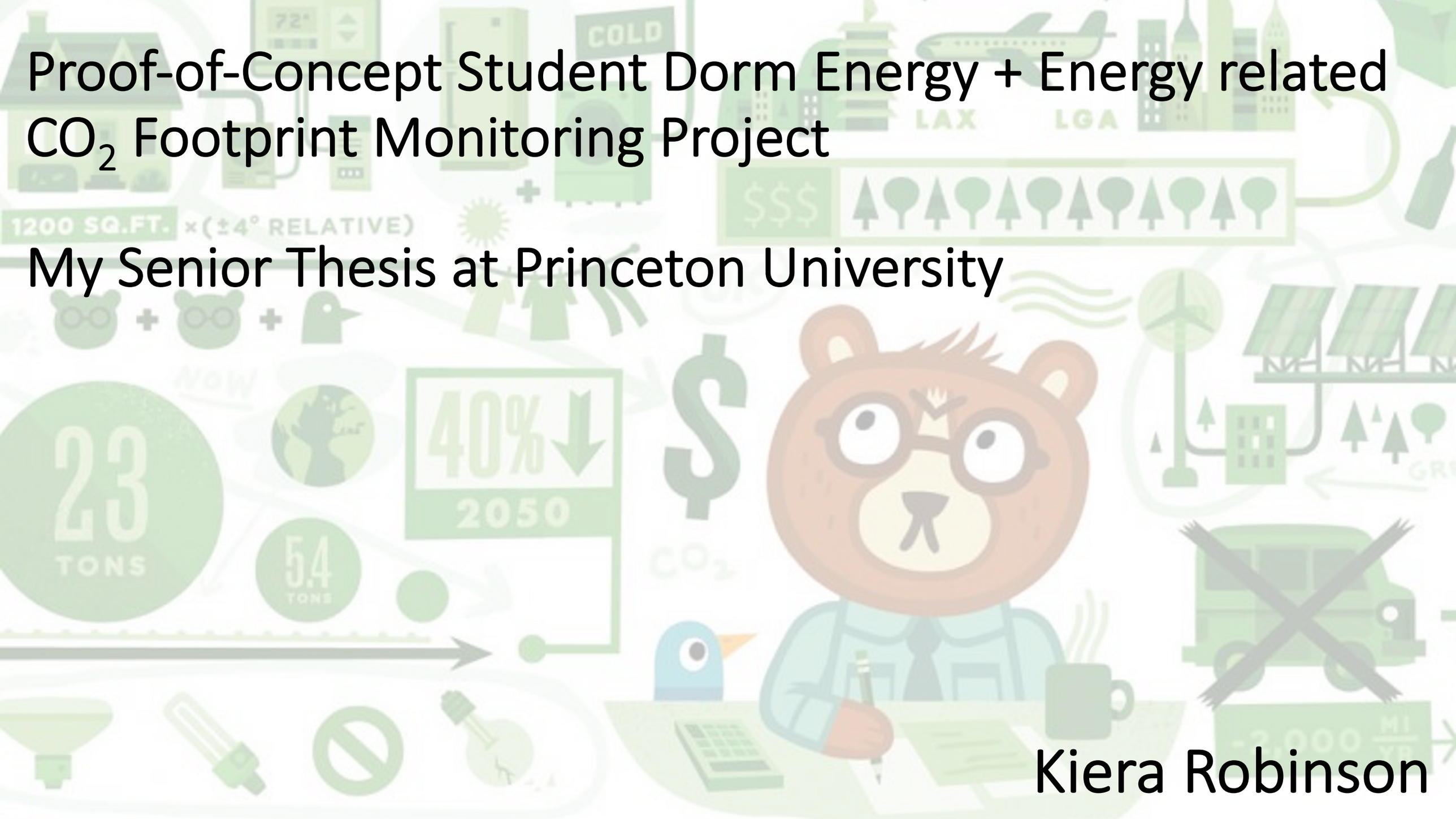
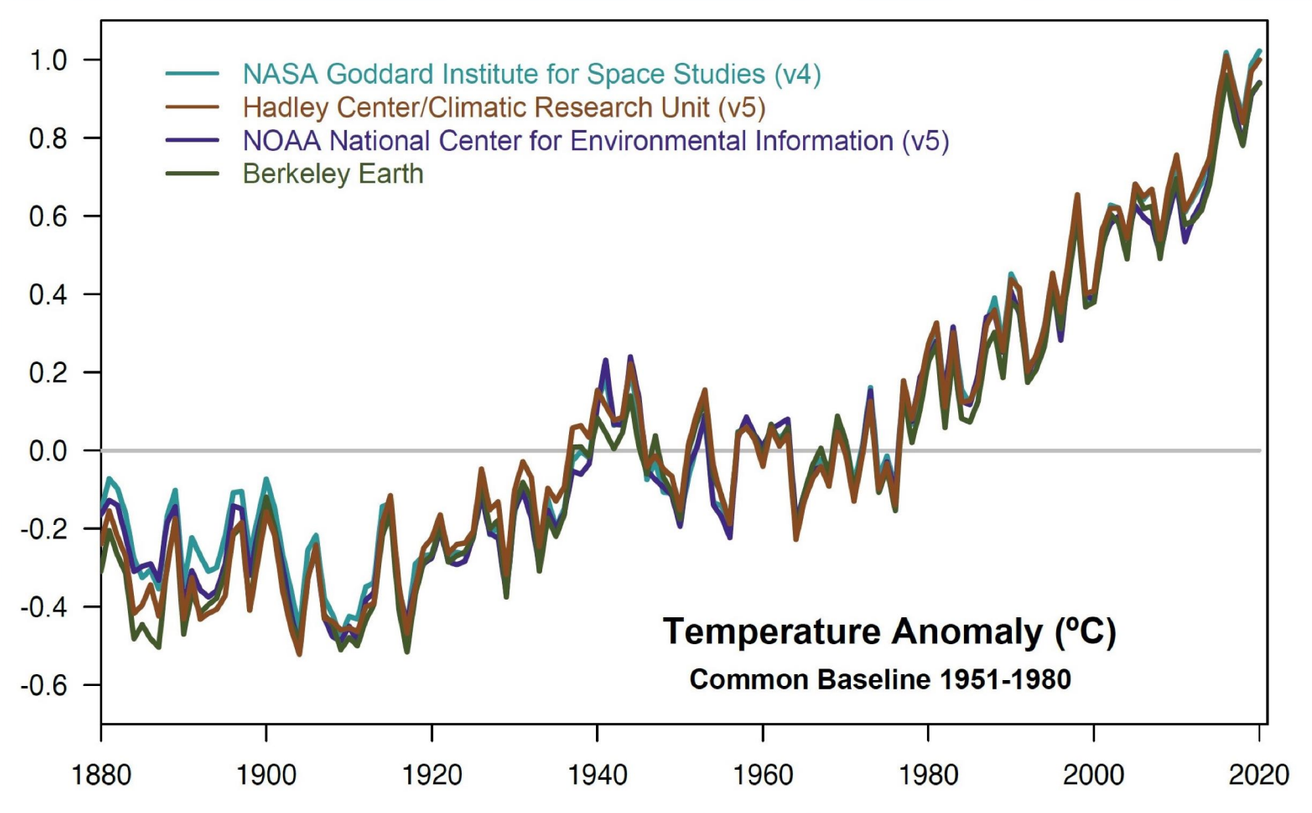
Proof-of-Concept Student Dorm Energy + Energy related CO_2 Footprint Monitoring Project

My Senior Thesis at Princeton University

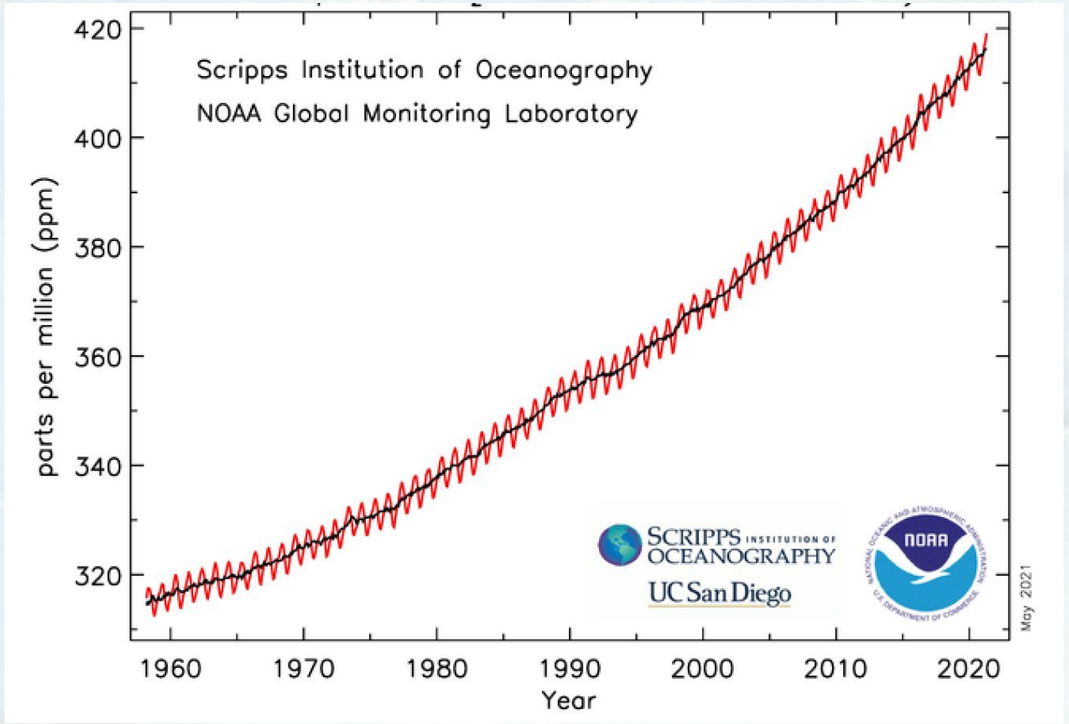


Kiera Robinson

The Problem – Climate Change is accelerating



Annual temperature anomalies from 1880 to 2019, using a baseline mean from 1951-1980, as recorded by NASA, NOAA, the Berkeley Earth research group, and the Met Office Hadley Centre (UK).
Source: National Aeronautics and Space Administration: Goddard Institute for Space Studies.



Atmospheric concentrations of carbon dioxide in the Earth's atmosphere recorded at the Mauna Loa Observatory from 1958 at 316 parts per million (ppm) to 2021 at 419 ppm.
Source: Scripps Institution of Oceanography

Princeton University's Campus Power

- Princeton University's Campus electricity comes from a complex system that utilizes a number of energy sources:
  - Solar Array: 65 KW on Frick Chemistry, 4.5 MW in West Windsor
  - Campus Power Plant: 15 MW, (2x) 260 KW backpressure steam turbines which operate whenever there is enough steam flow going to campus
  - Public Service Enterprise Group (PSE&G) Grid

High-Level Overview of Princeton's Microgrid

Residentail Buildings

Research Laboratories

Museums & Libraries

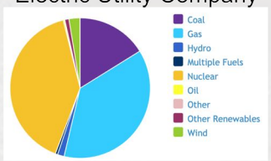
Princeton University Microgrid

Campus Solar Field

PRINCETON UNIVERSITY

SUNPOWER®

Electric Utility Company



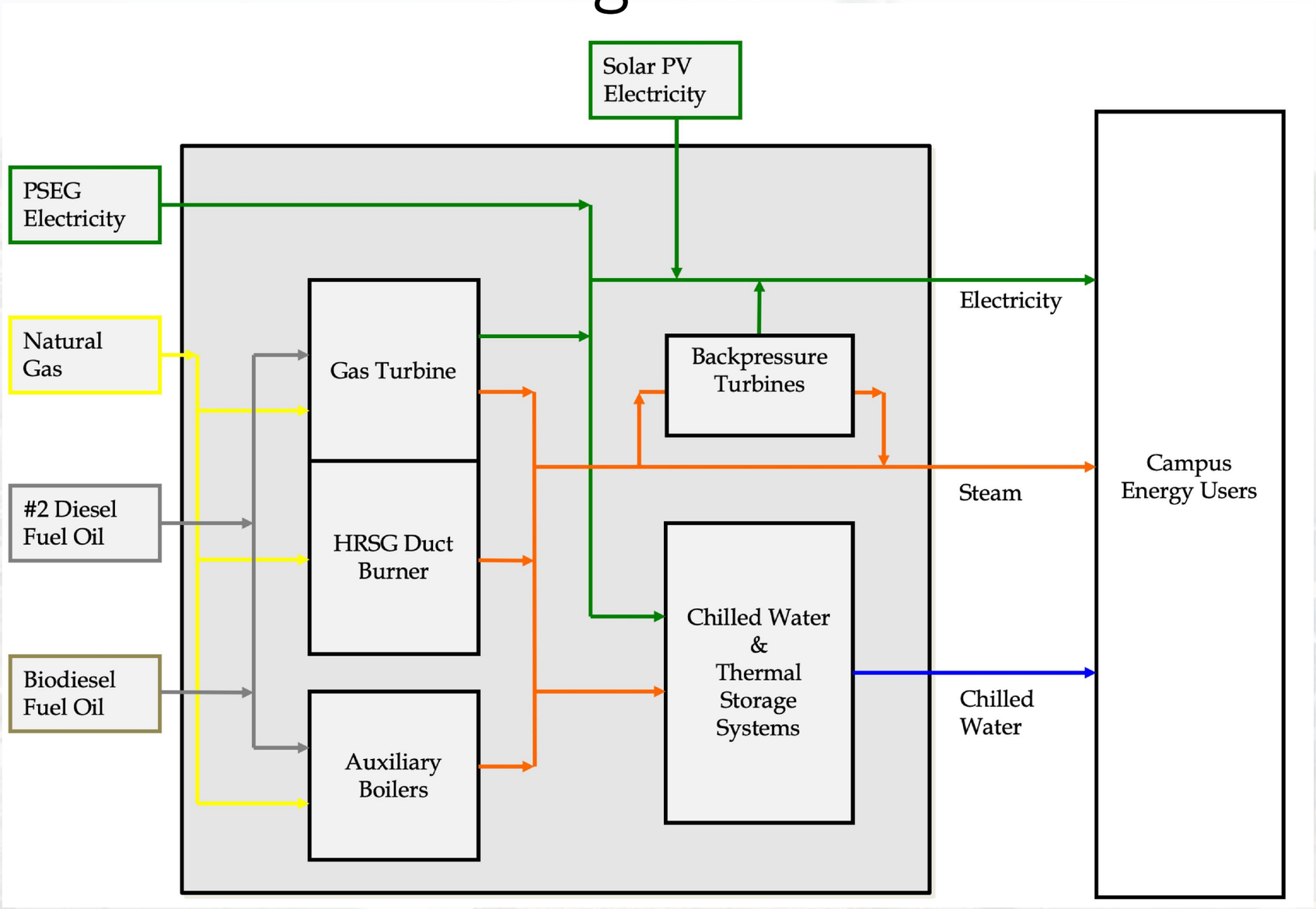
PSEG

pjm

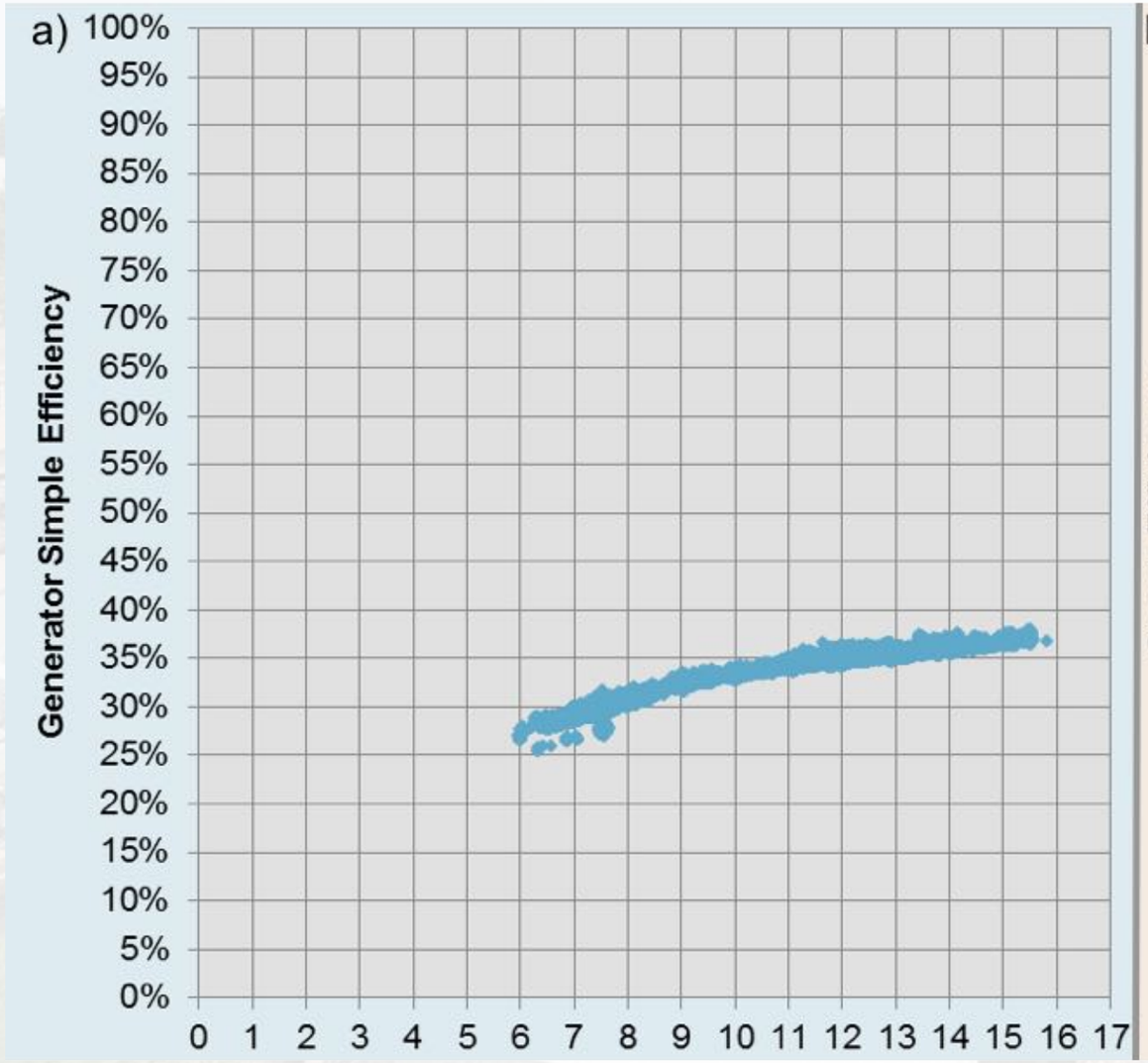
Campus Cogeneration Facility

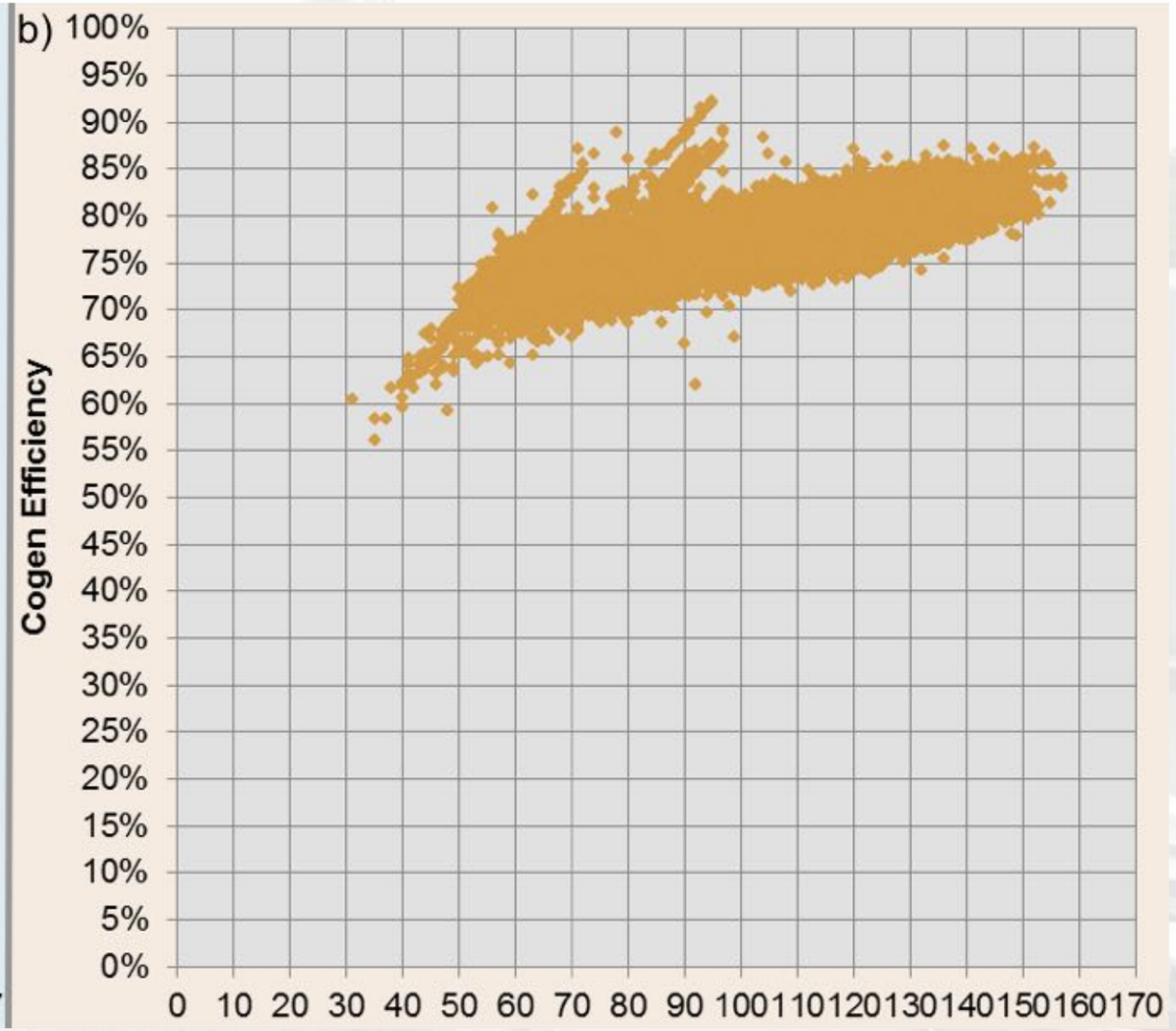
Facilities
Princeton University

Princeton University's Campus Power Balance Diagram

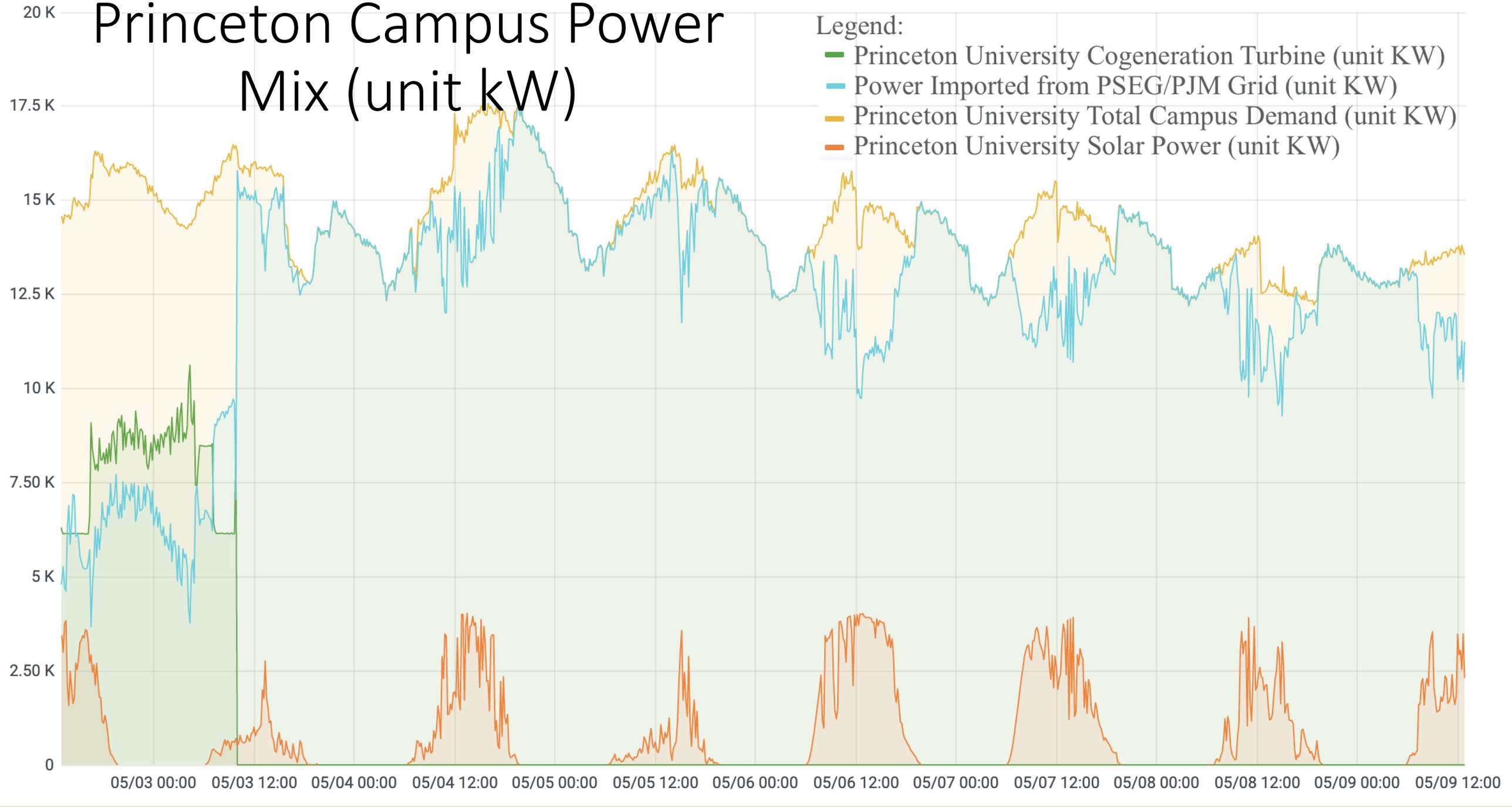


Princeton University's Cogen Plant offers more Efficiency

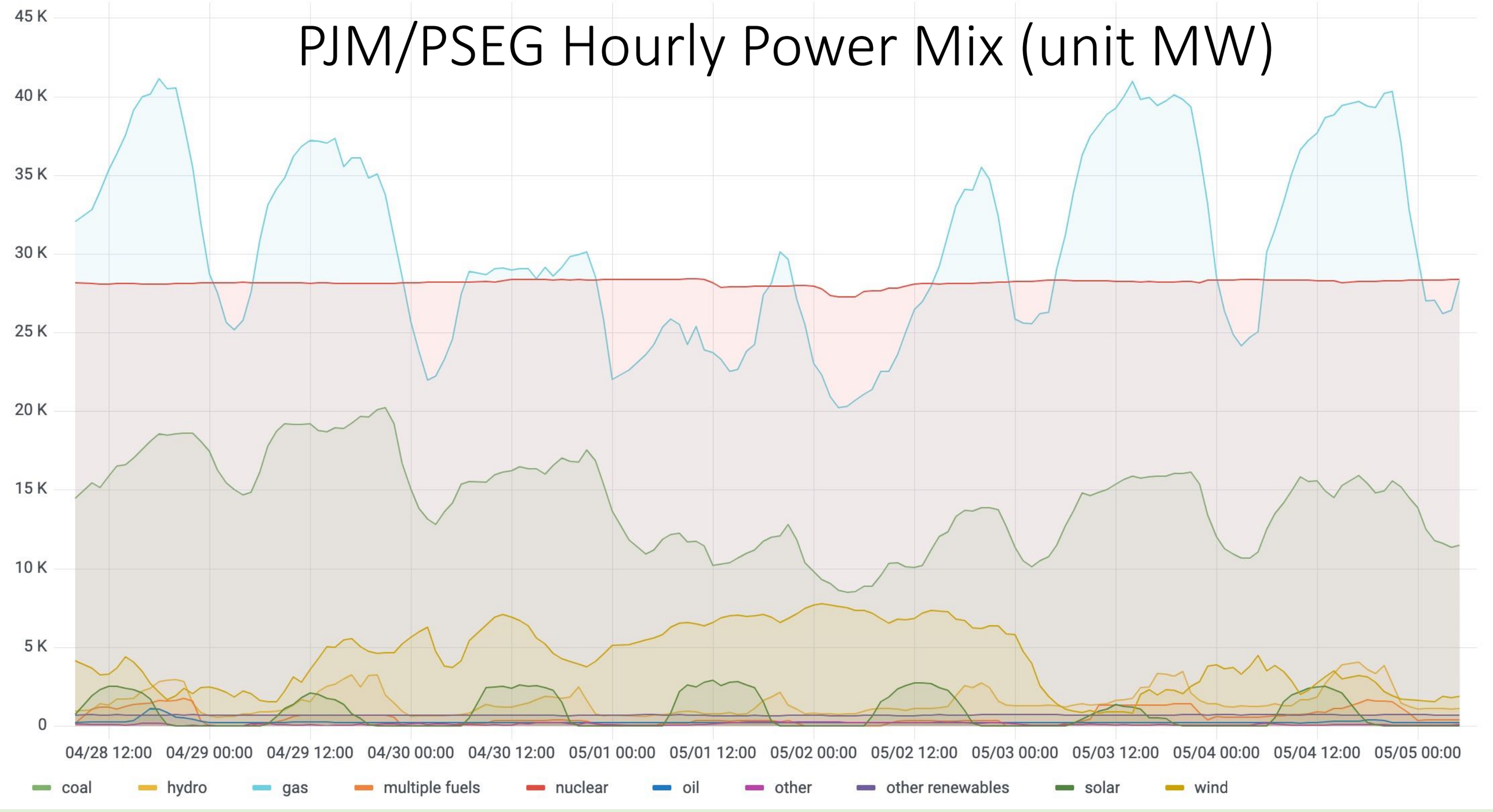




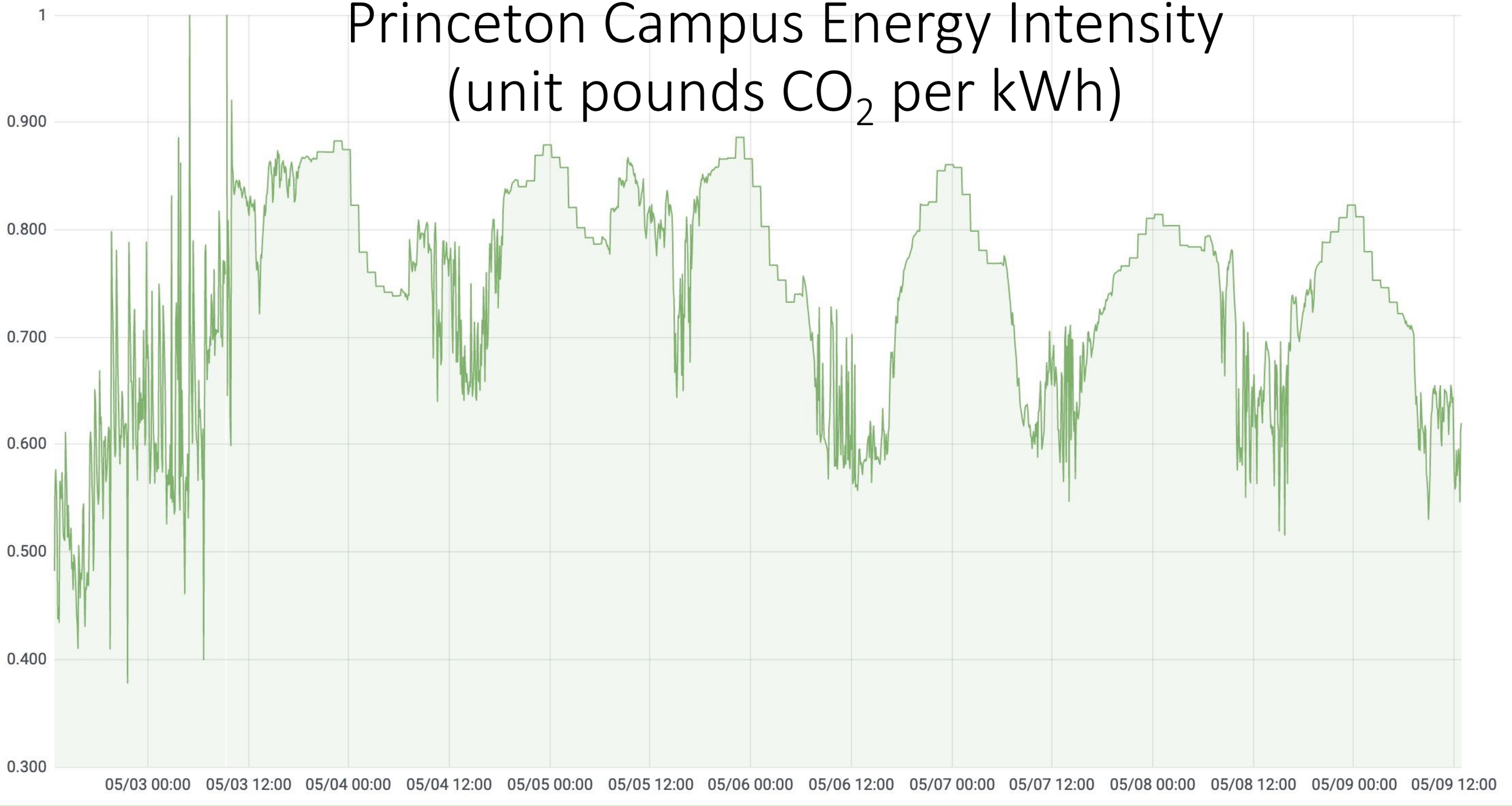
Princeton Campus Power Mix (unit kW)



PJM/PSEG Hourly Power Mix (unit MW)



Princeton Campus Energy Intensity (unit pounds CO_2 per kWh)



Princeton's Energy is lower in Carbon Intensity than that solely supplied by PSEG/PJM Grid

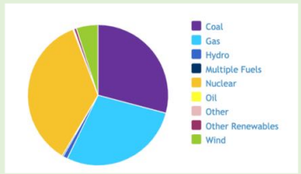
Fiscal Year	Net lbs CO_2 lbs/MWh	
	Princeton University	PSEG
2013	655.3	793.0
2014	765.7	806.0
2015	732.2	778.0
2016	744.2	725.0
2017	697.0	730.0
2018	N/A	759.0
2019	773.5	785.0
2020	N/A	745.3

Conversion coefficients for electric energy delivered to Princeton University from PSEG, and that distributed throughout the University from cogenerated electricity combined with that purchased from the grid (N/A indicates not available).

Project Question

Overview of Campus Power

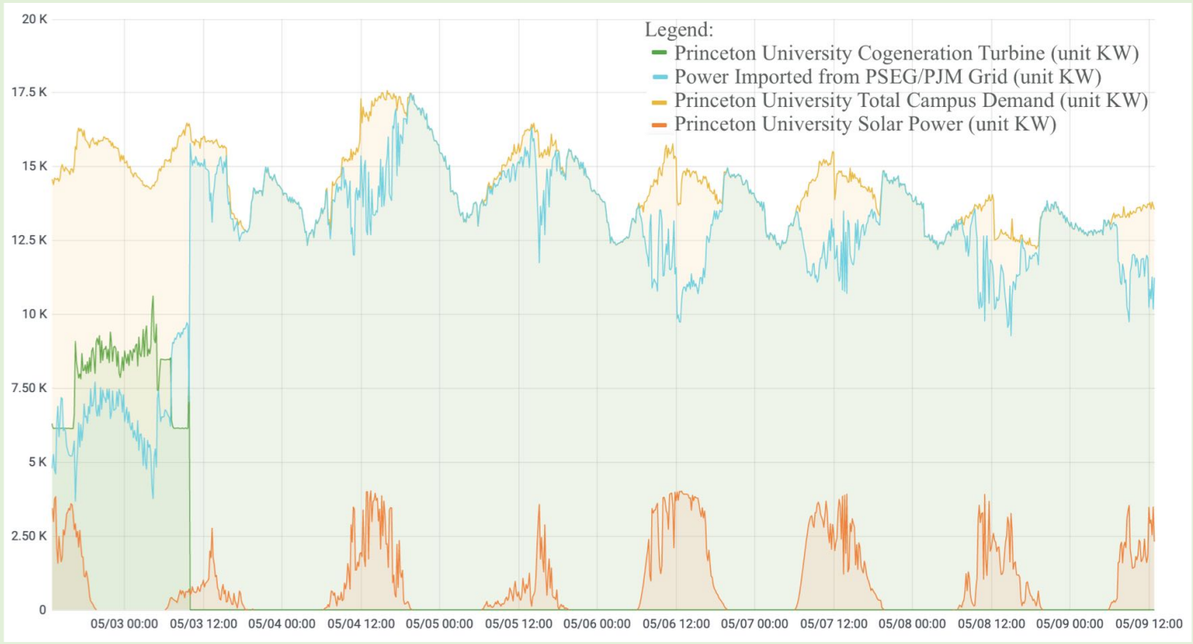
SUNPOWER



pjm PSEG

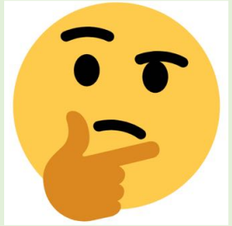
Facilities
Princeton University

Real-time Campus Power changes with time



Thus, the composition of energy supplied to campus dorms changes with time

- Is there a way to determine real-time carbon footprint and share it with students?
- Would their behaviors and/or attitudes towards climate change be affected?



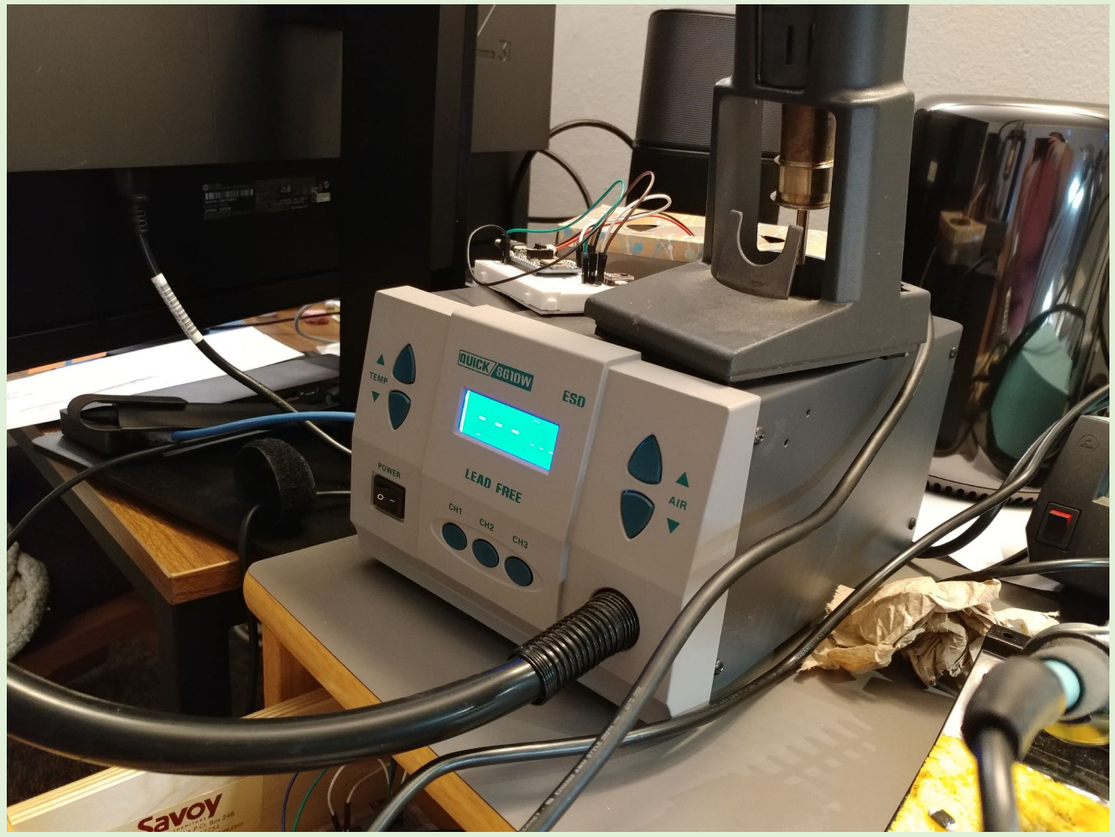
I will build something to do just that!

Methods

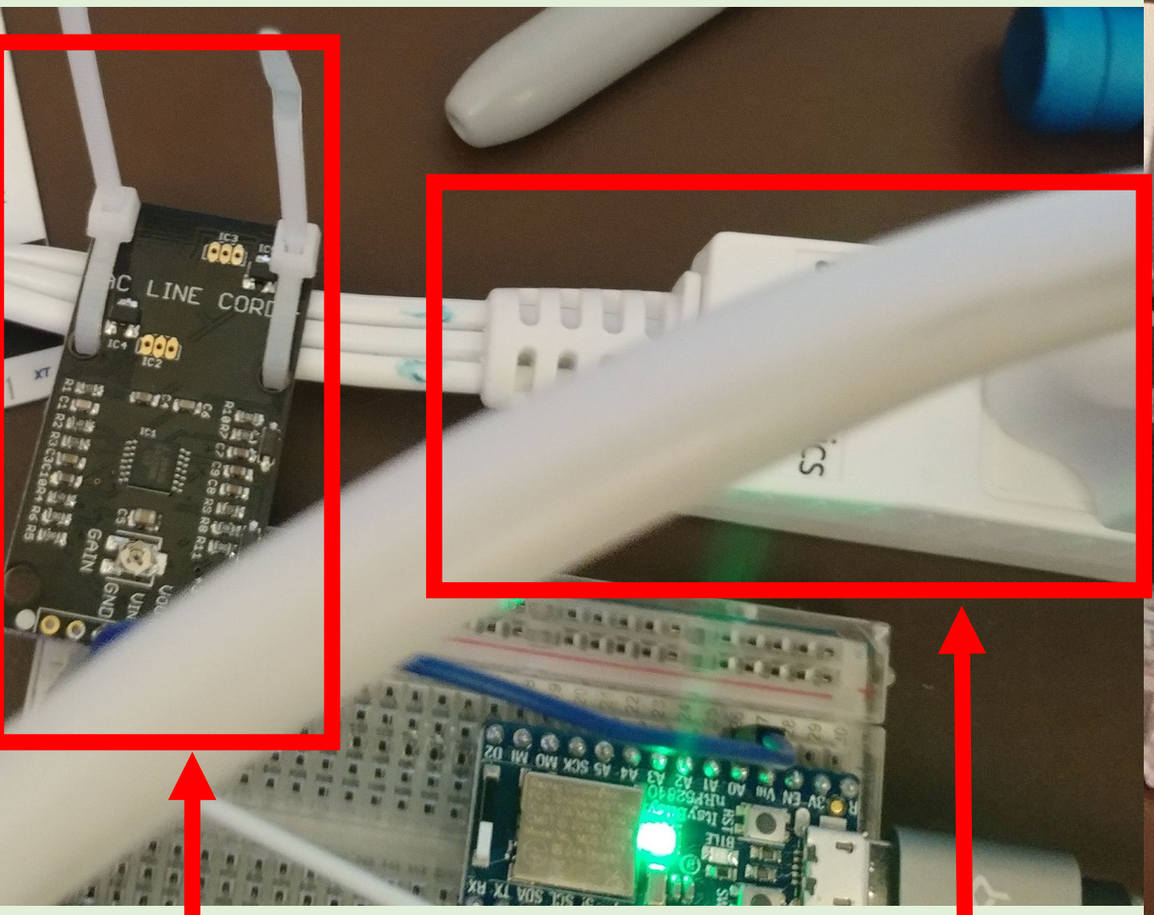
- Use IoT devices to transmit sensor readings
- Perform real time calculations based on real-time power composition
- Relay real time information to some Users with a Touchscreen Kiosk Device

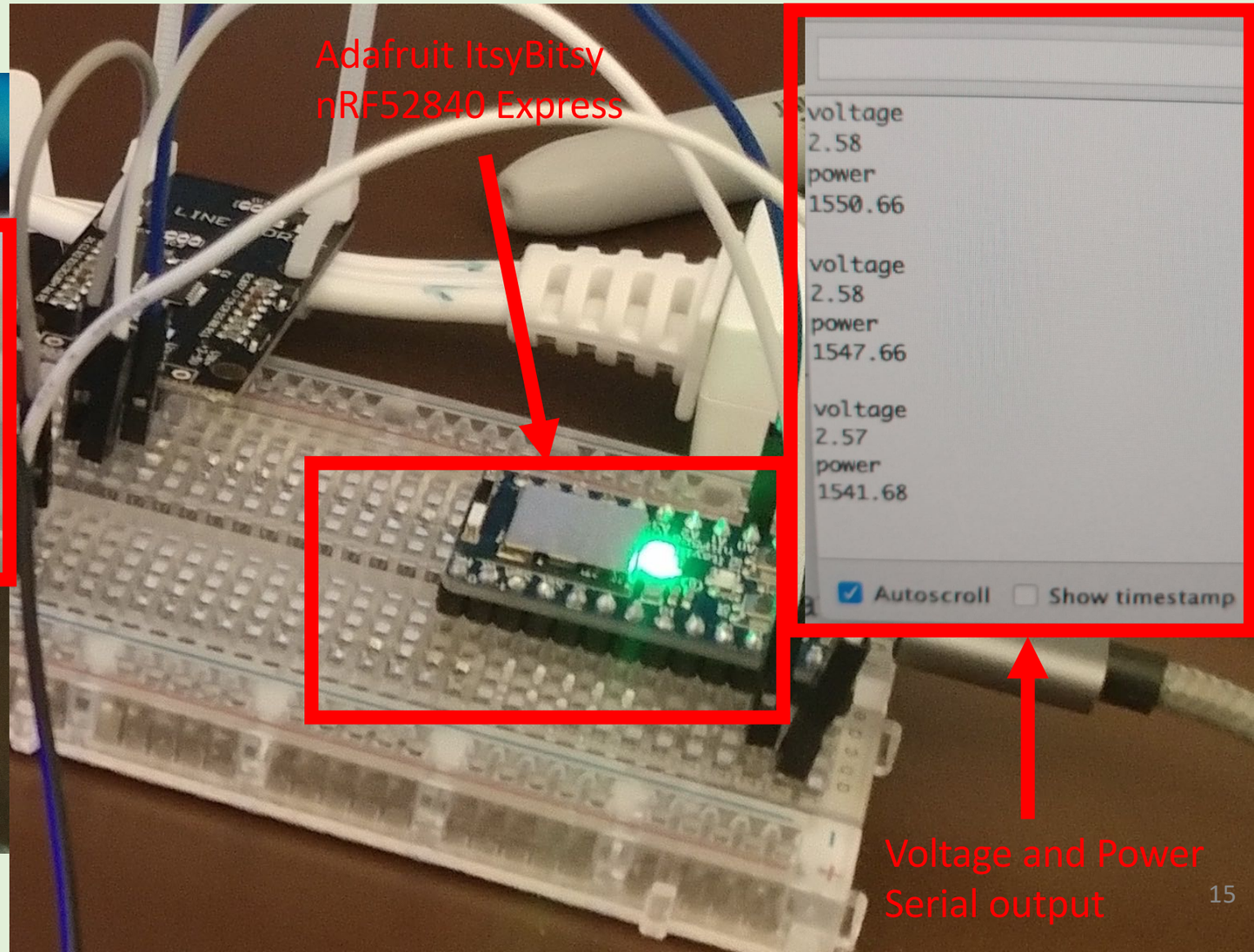
Early Prototyping

Solder Work Area in Student Dorm

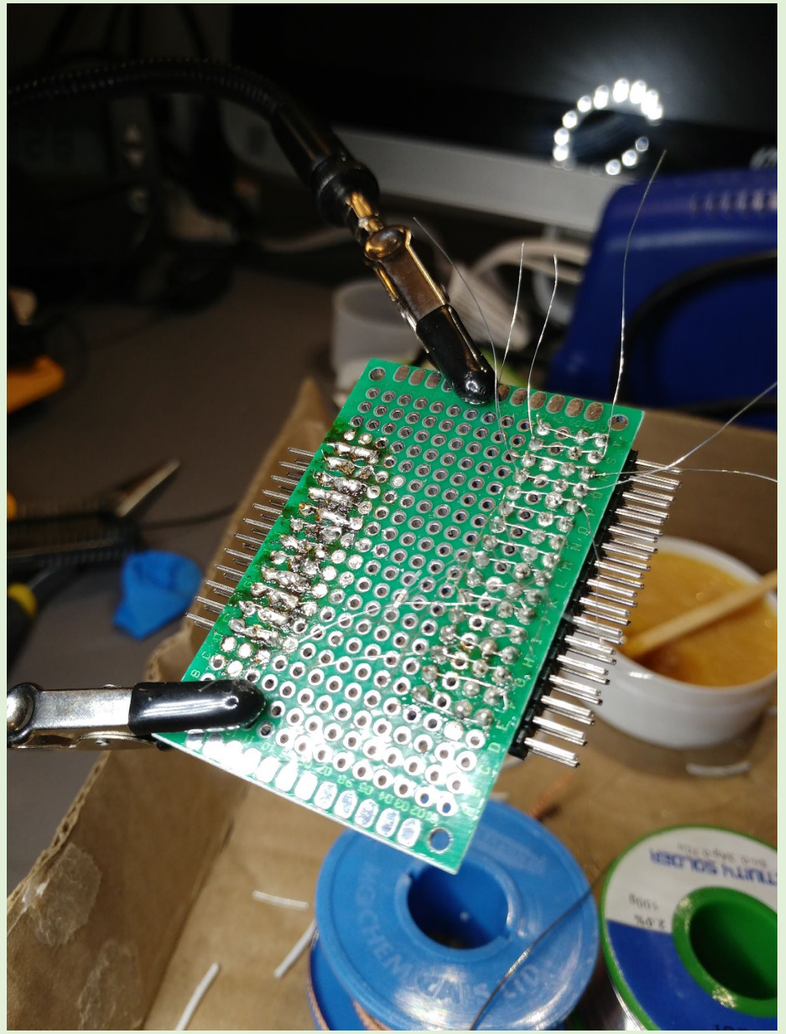


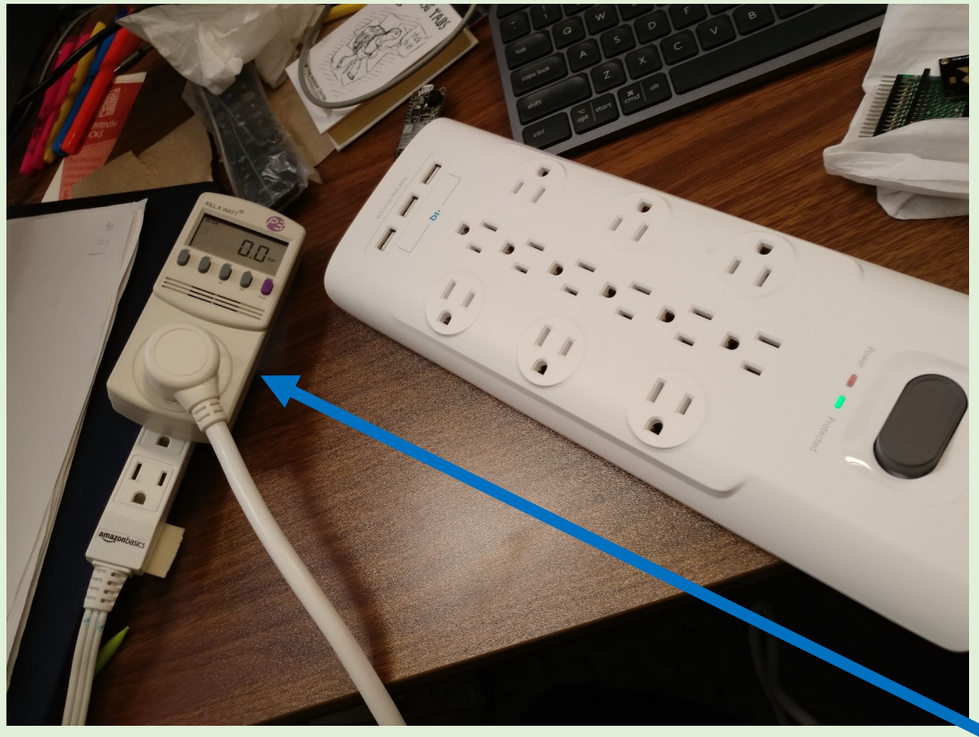
Set up Microcontroller with non-contact current sensor from Modern Devices

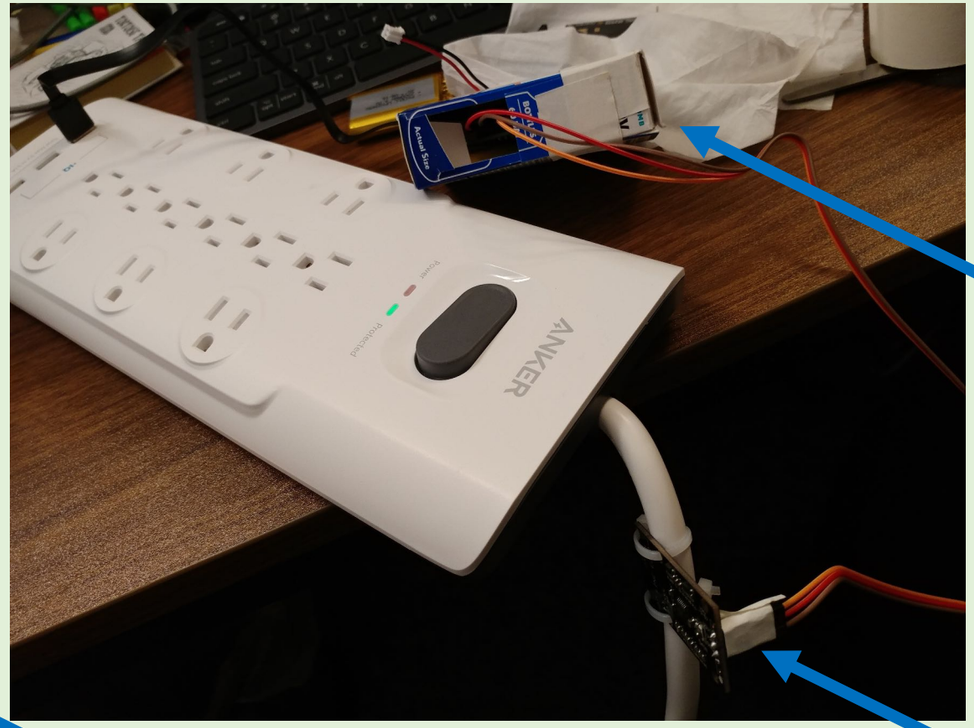




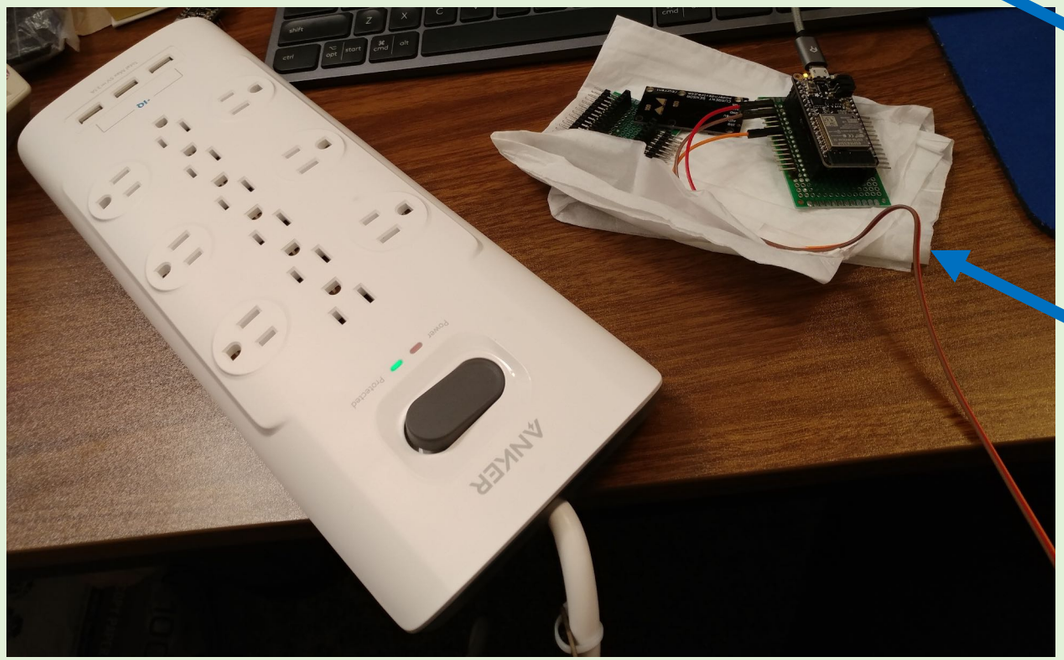
Soldering boards for prototyping







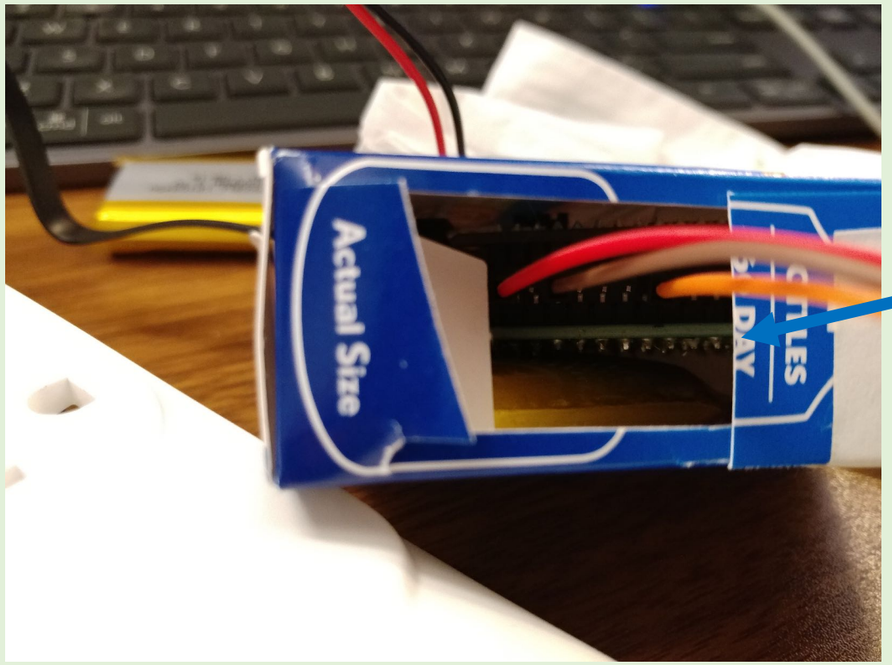
My “dodgy” attempt to make an enclosure



Calibrating current sensor

Non-contact current sensor wrapped around cord

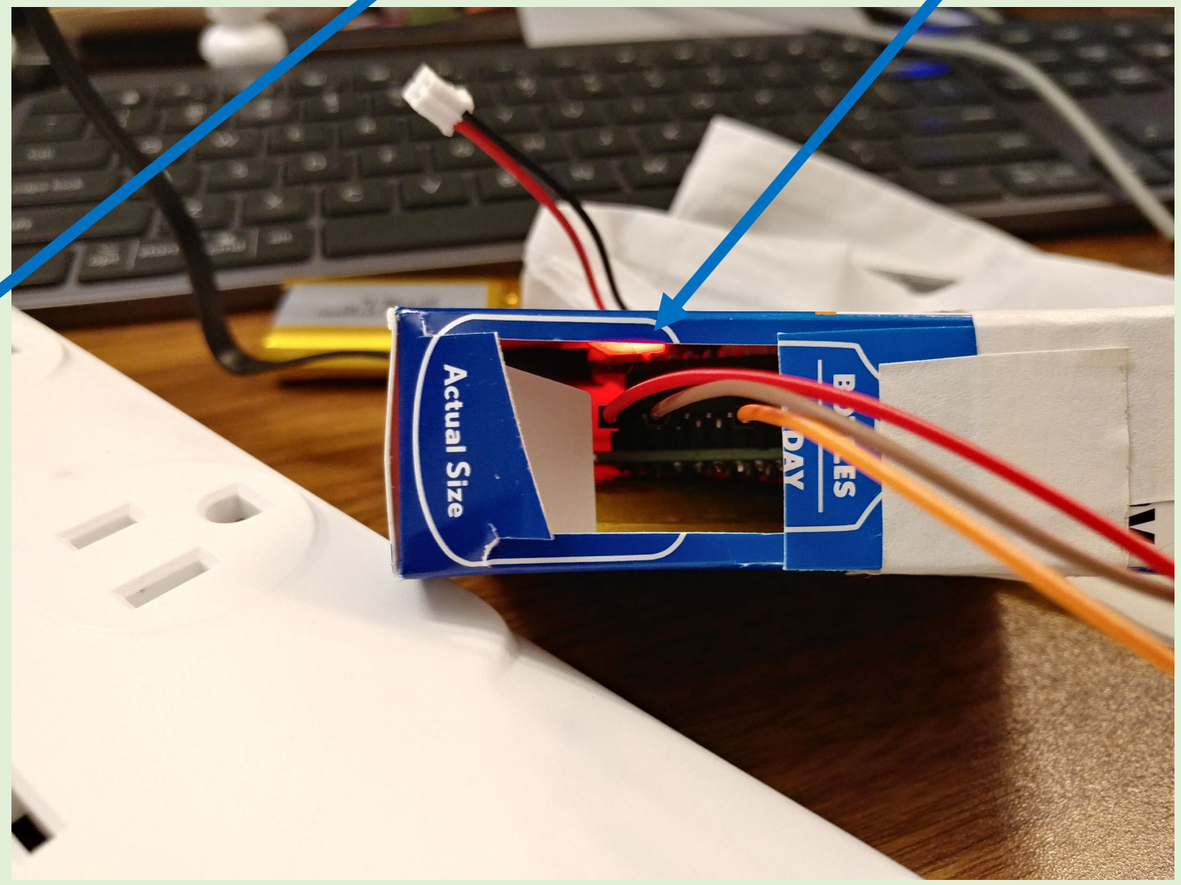
Microcontroller connected to current sensor

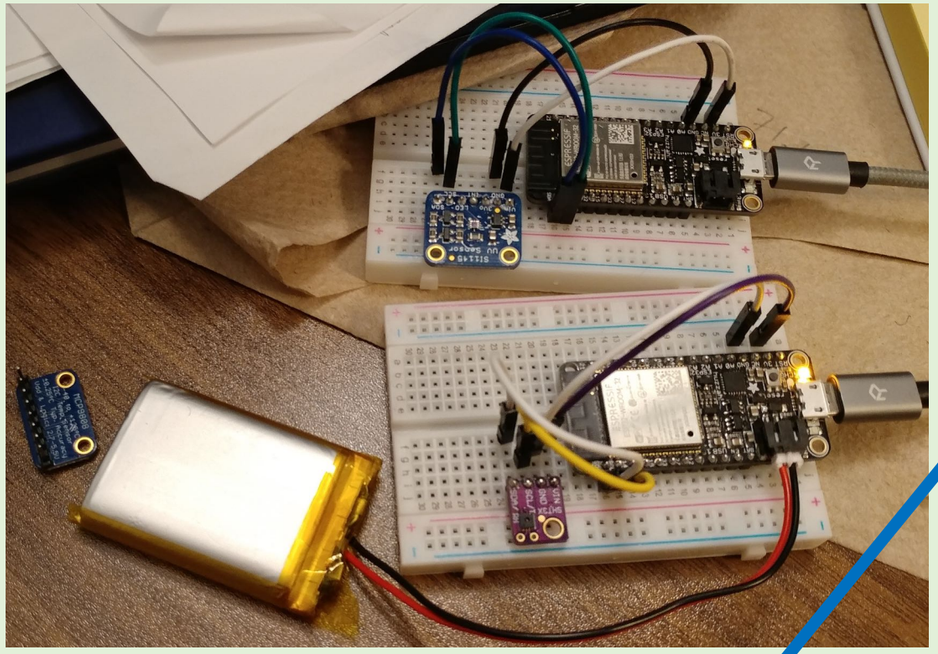


Cut out for the wires to current sensor

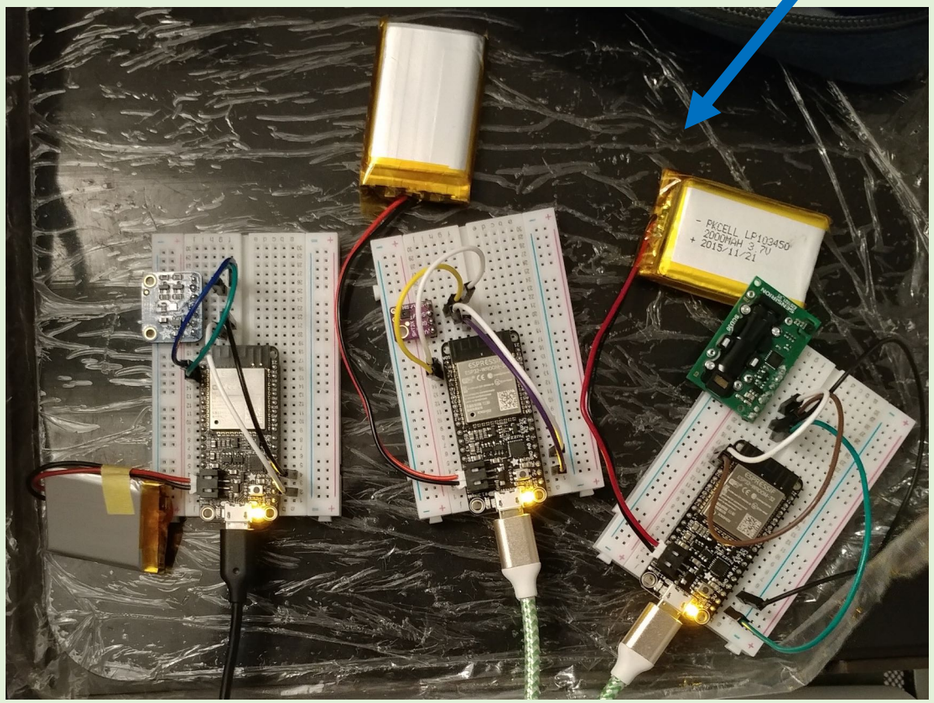
Red light means IT LIVES!!!

My “dodgy” enclosure but…
REDUCE! REUSE! RECYCLE!

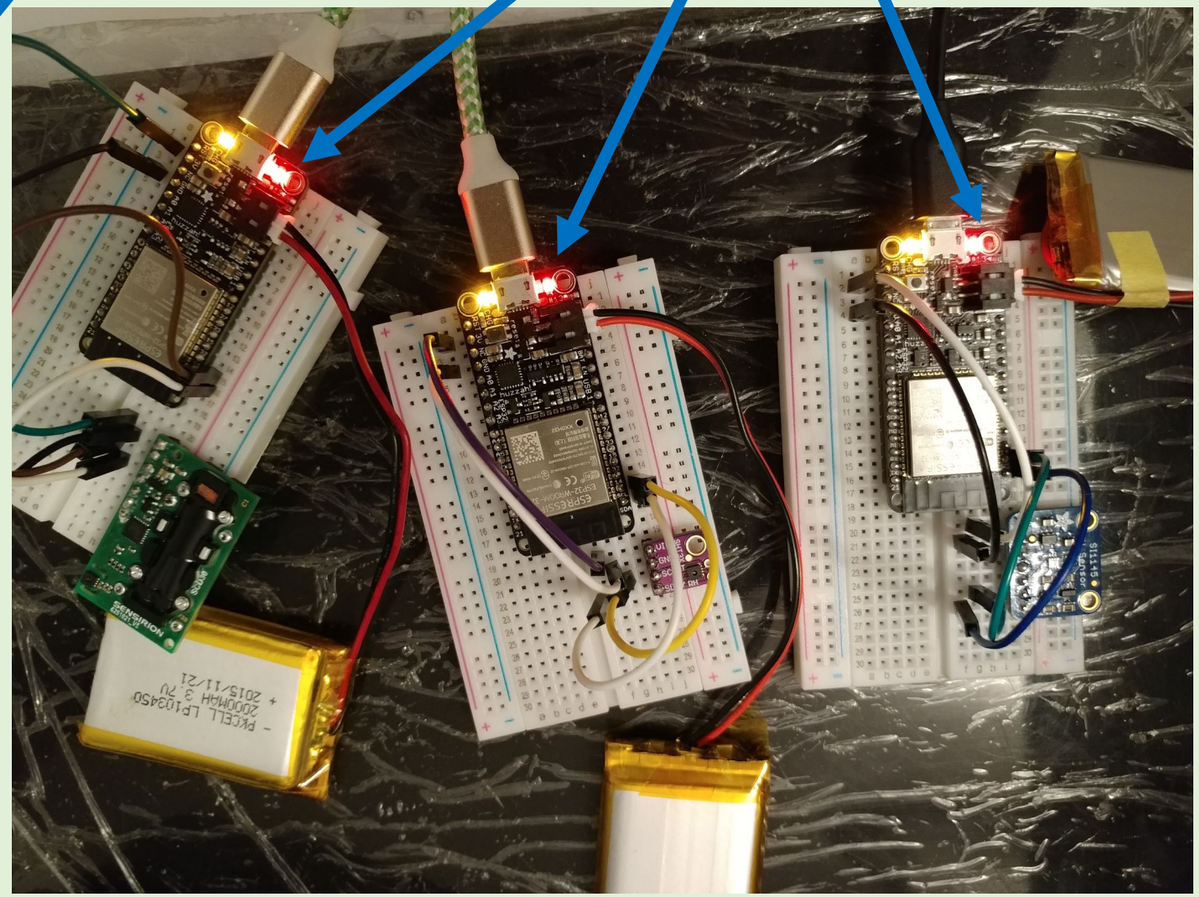




Environmental air and light sensing: Temp, RH, CO2, Lux



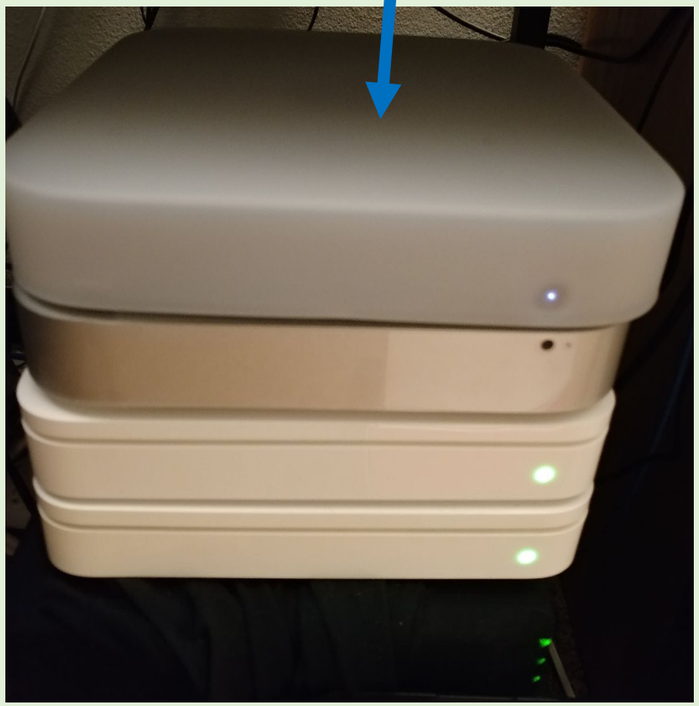
Red light flashes when data transmission is successfully requested by server and data is sent



Data fetching, processing, and hosting

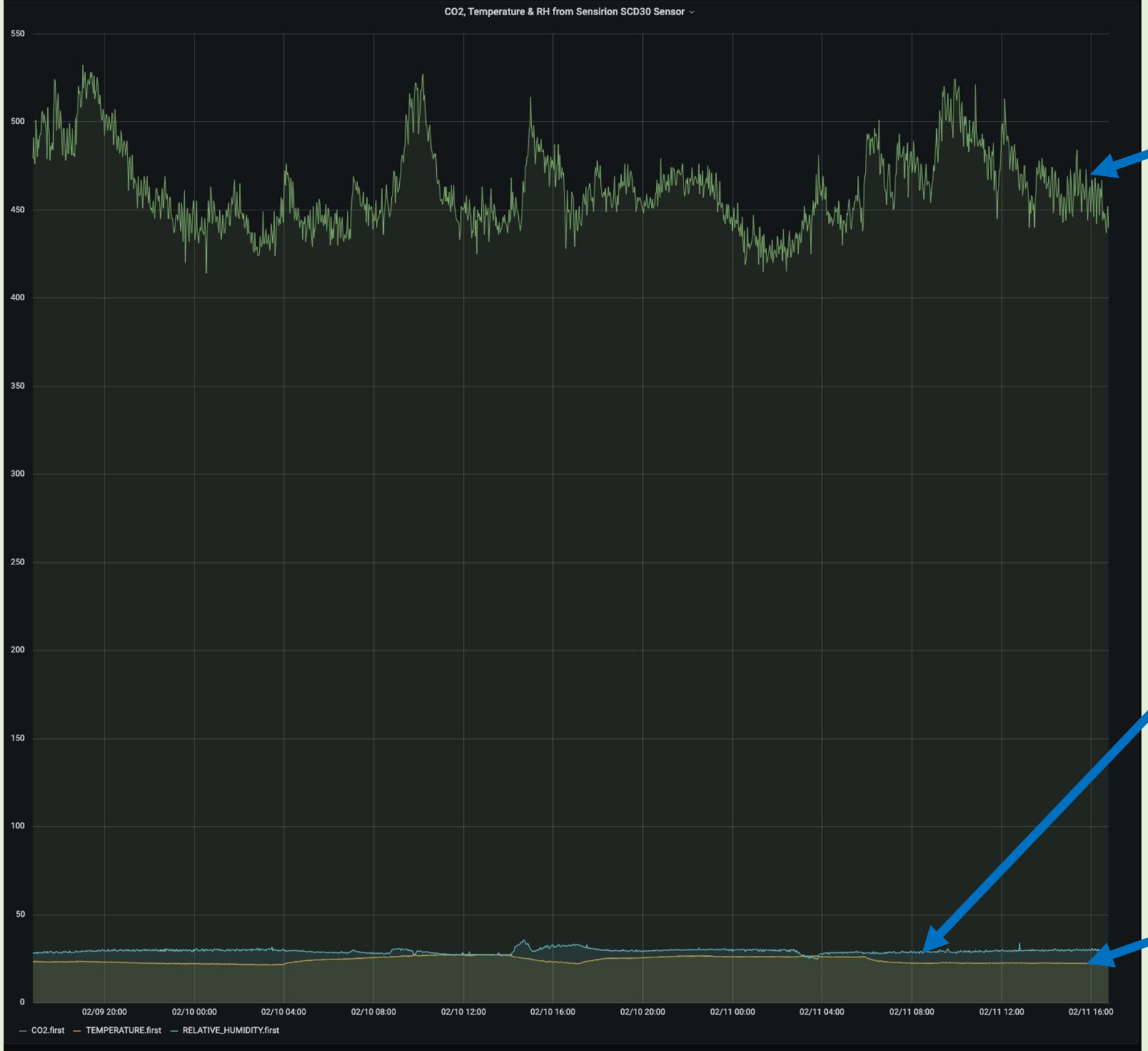
Apple Mac Pro workstations for computations and website serving

Apple Mac Mini for database management



Synology DS 416: Network Attached Storage Device

Environmental air sensing: Temperature, Relative Humidity, CO_2 from Sensirion SCD30

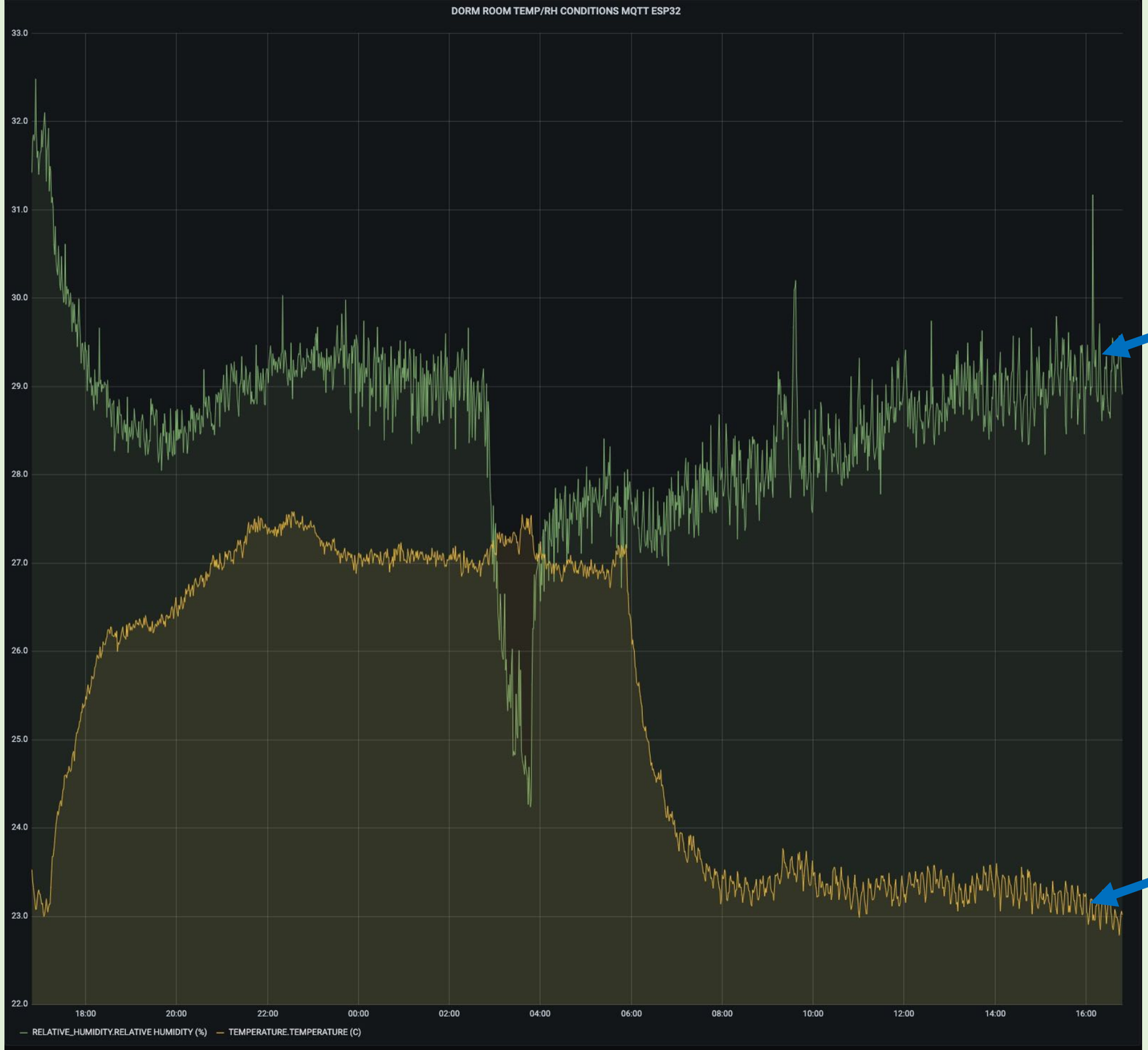


Room CO2; Spikes when I enter the room

Relative Humidity (%)

Temperature (°C)

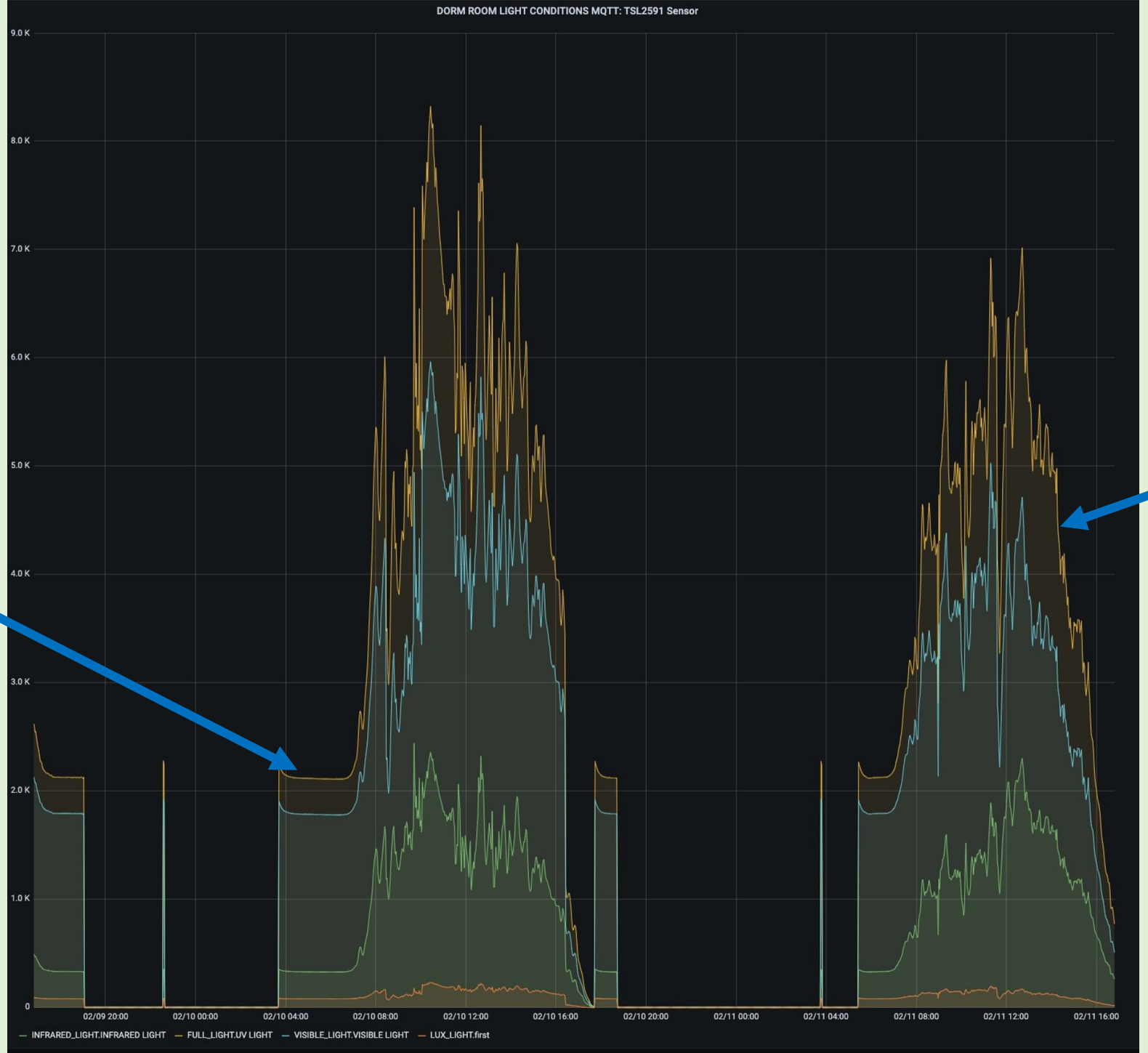
Environmental air sensing: Temperature, Relative Humidity



Relative Humidity (%)

Temperature (°C)

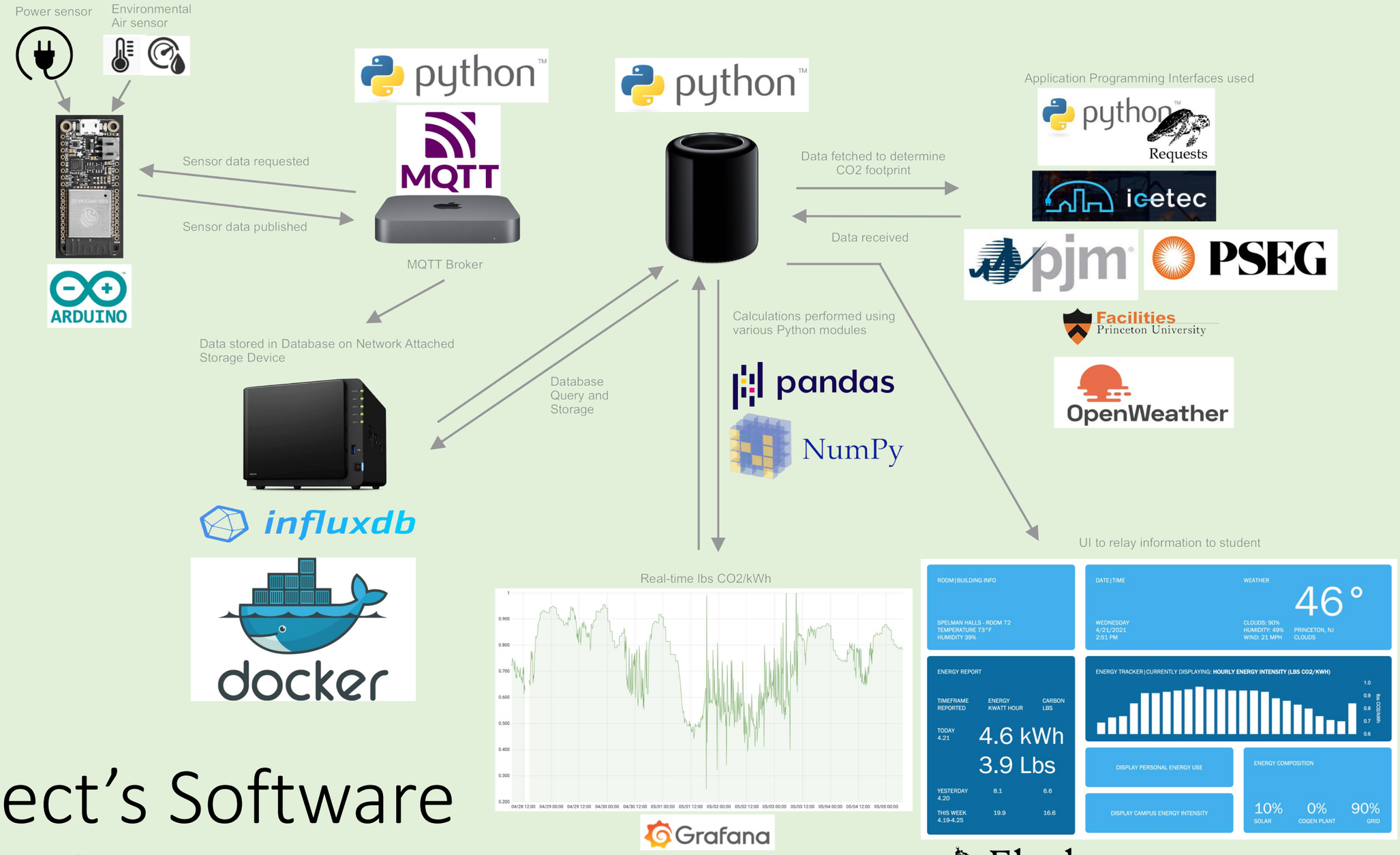
Environmental Light sensing: Lux, infrared, visible light



Artificial Lighting

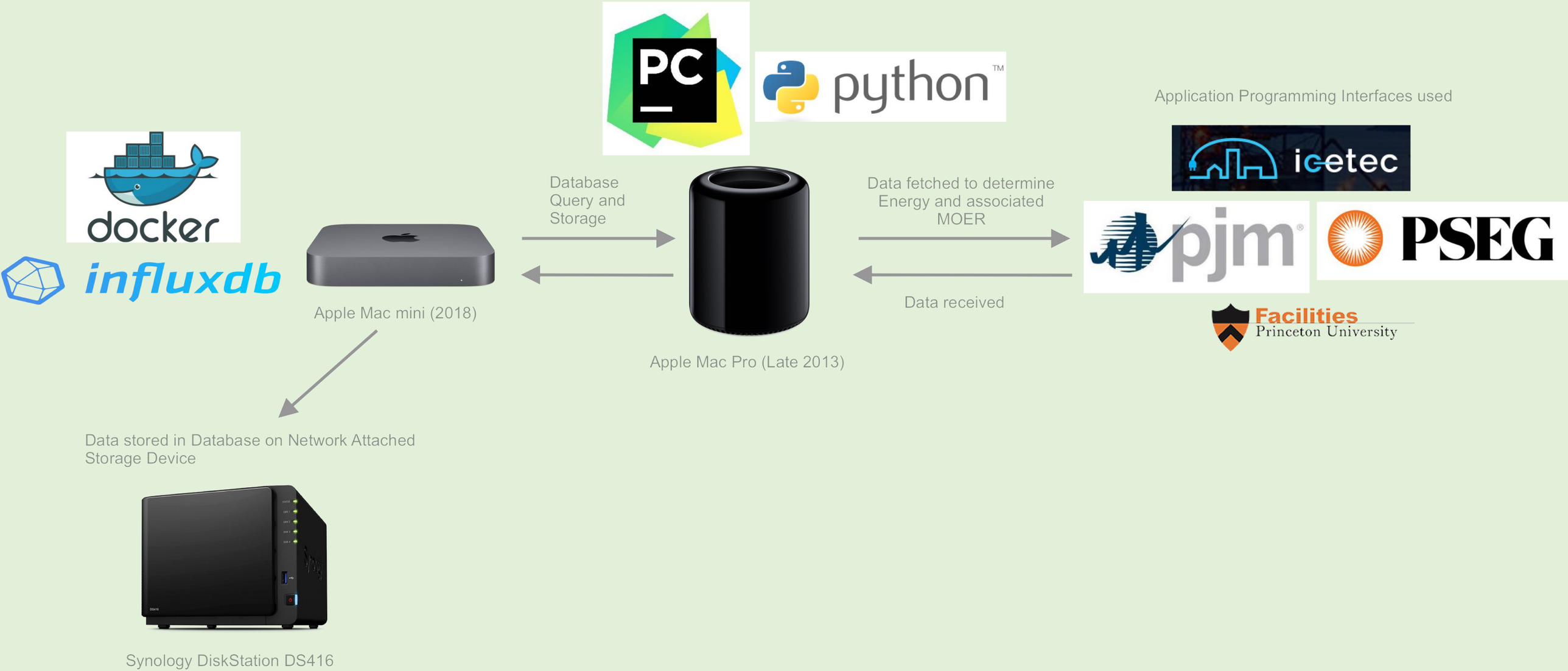
Natural Lighting

Final Design and Implementation

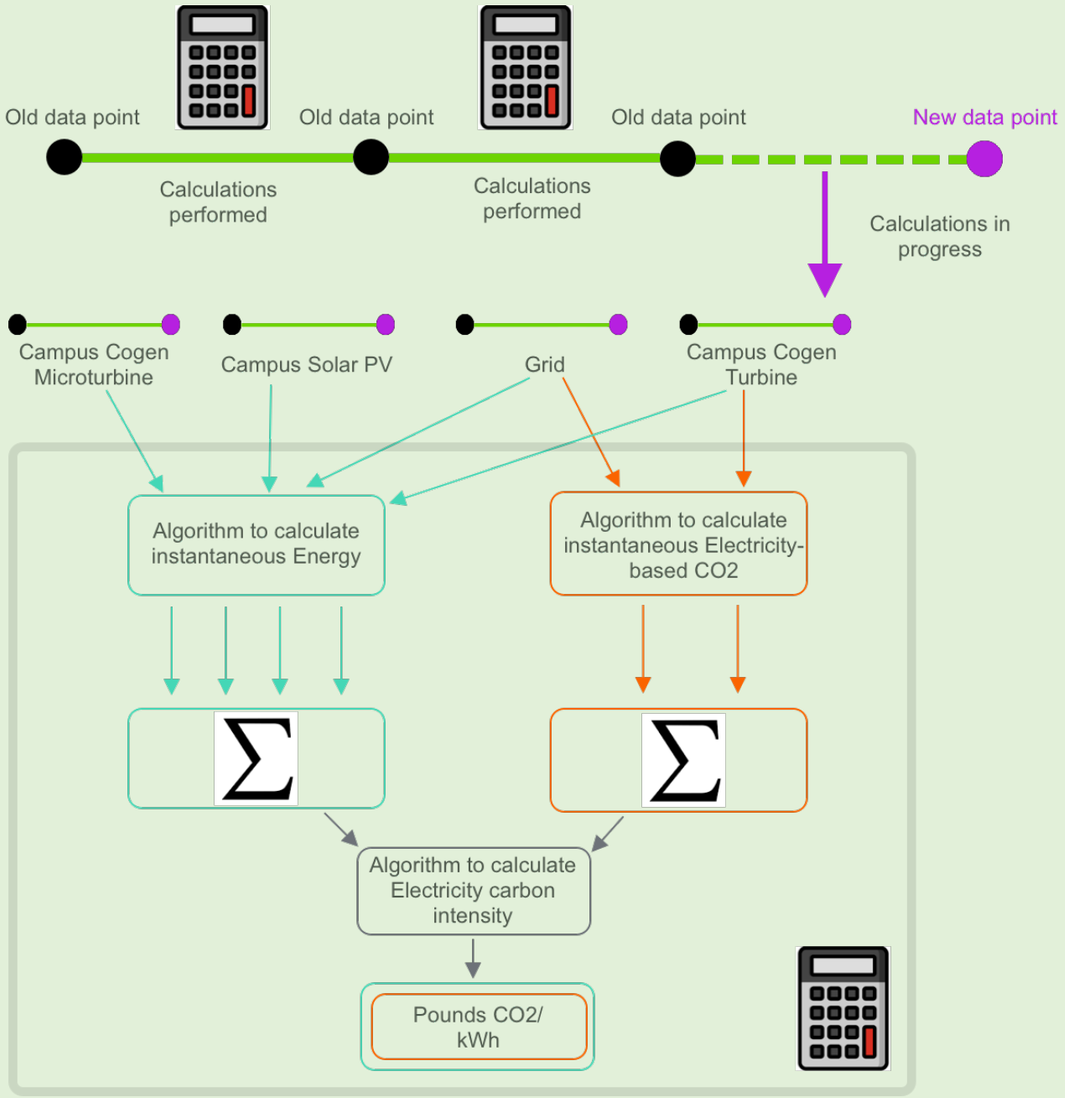


Project's Software Architecture

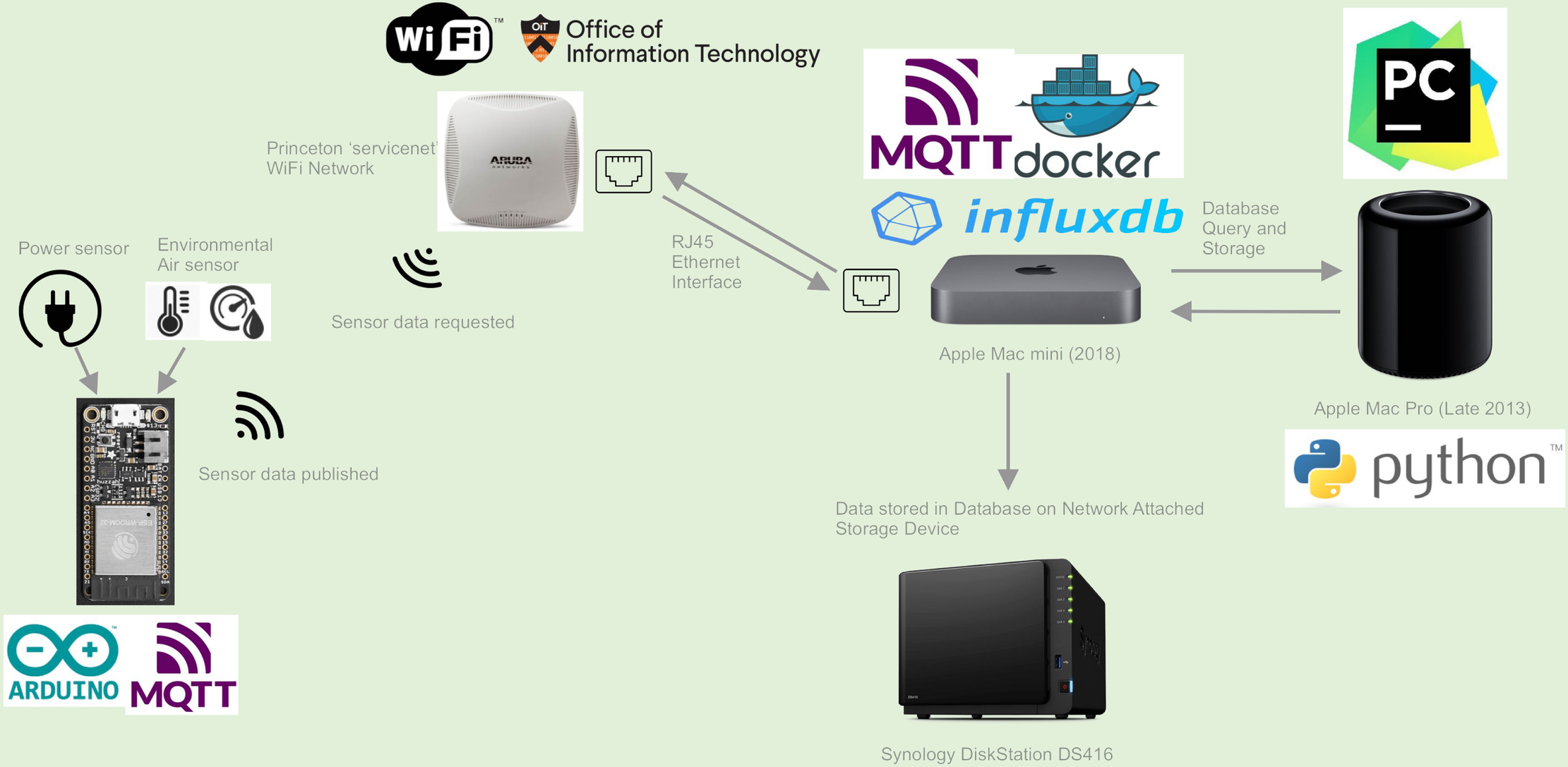
Apparatus and flow of data utilized to determine Campus Energy Calculations



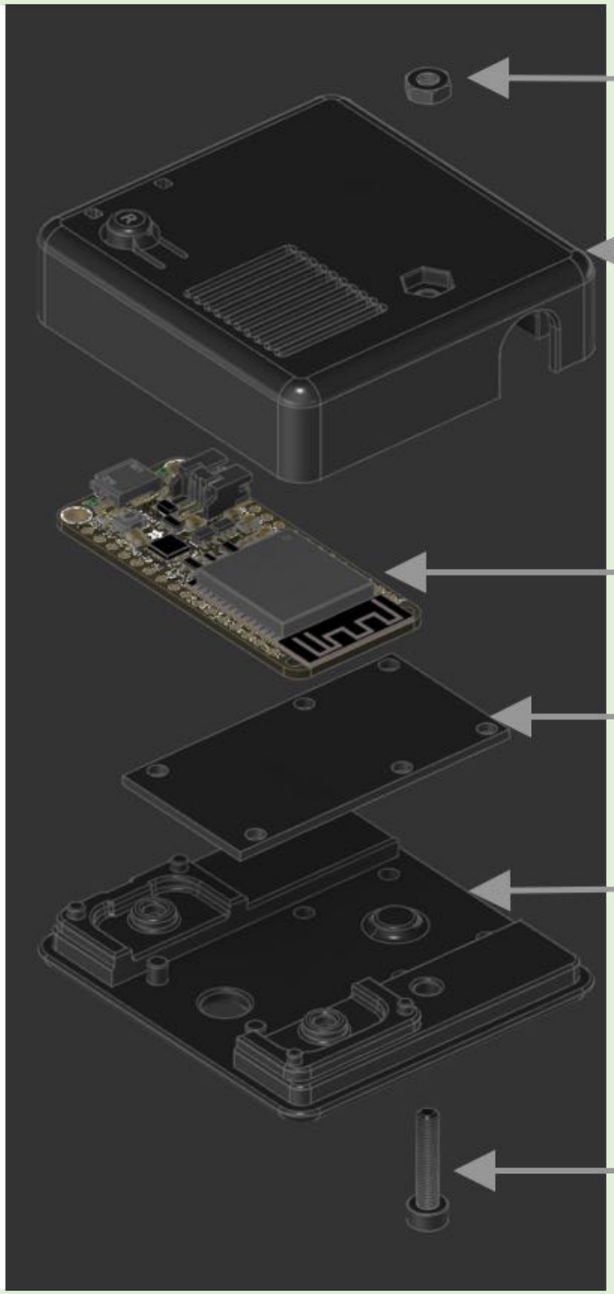
Calculation scheme used to determine instantaneous electricity-based carbon footprint



Apparatus and flow of data utilized in the Microcontroller based portion of this project.



CAD of Custom Power Sensor Enclosure



External Hex Bolt

Front cover of enclosure with i) pass-throughs for LED indicators and ventillation, and ii) protruded Reset button flap

ESP32 Microcontroller Unit

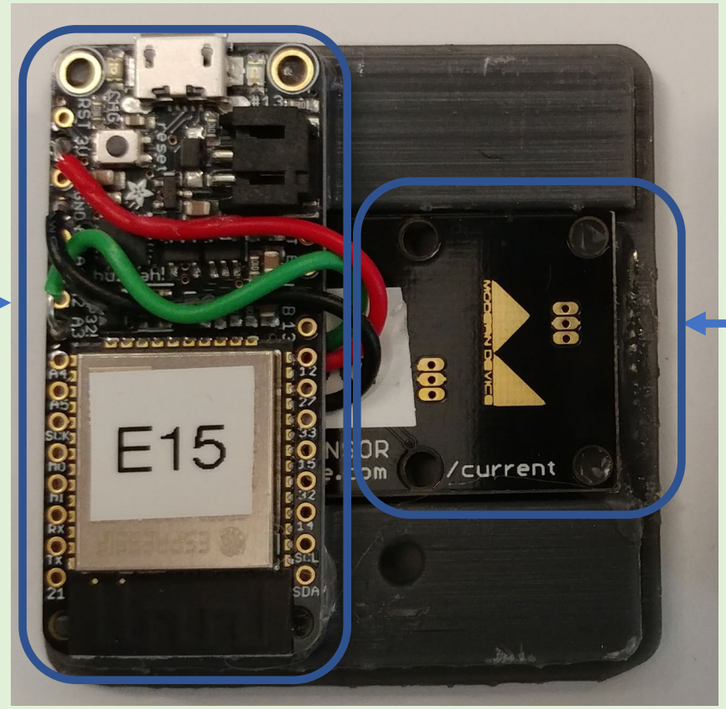
Modern Device Current Sensor

Back cover of enclosure with i) mountings protusions for MCU and sensor, and ii) pass through for sensor gain calibration

Hex Socket Screw

Custom Power Sensor Enclosure Assembled

Microcontroller Unit
ESP32 w/ WiFi and Bluetooth Classic/LE support

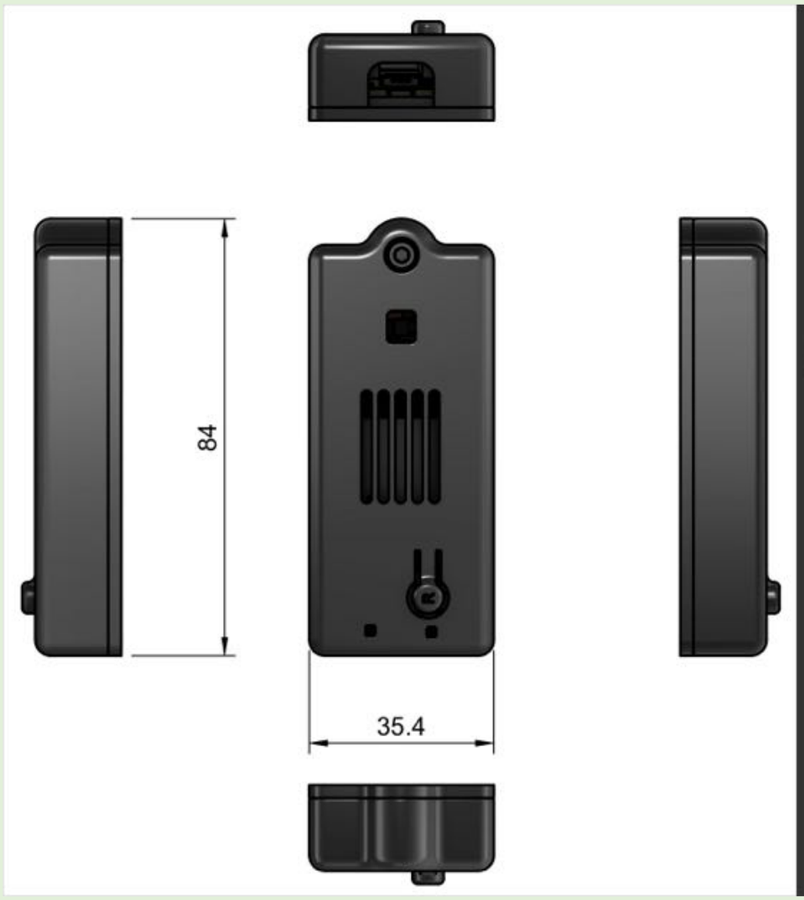


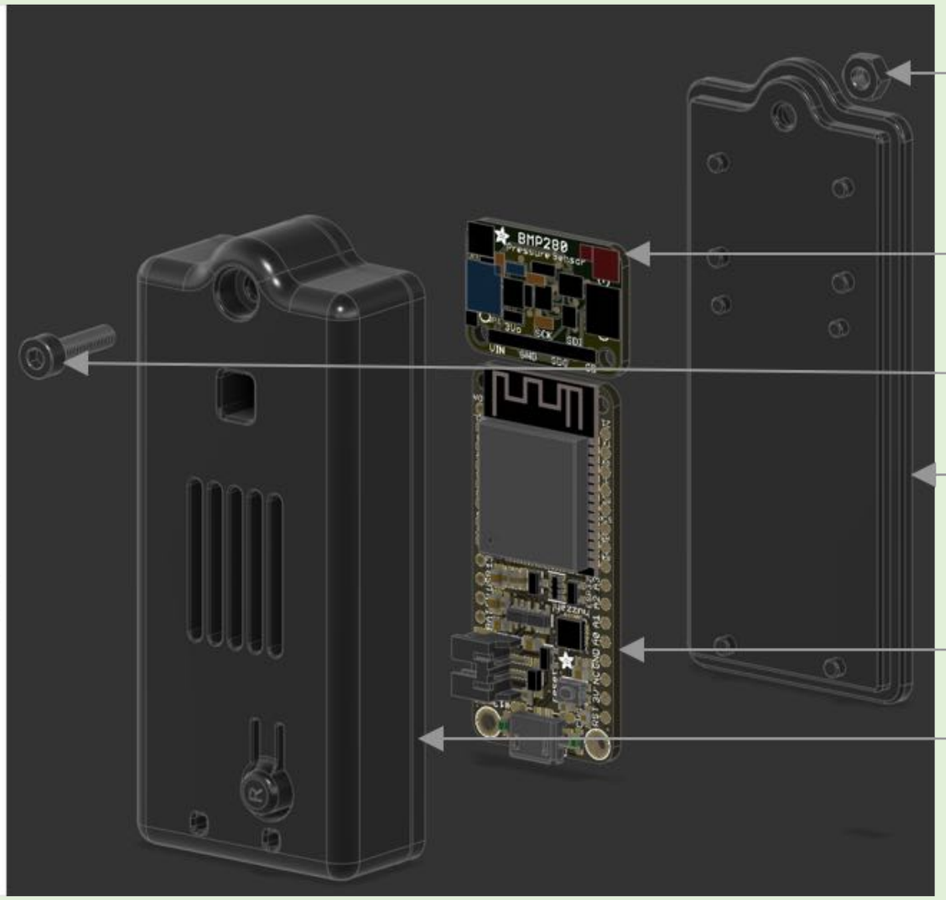
Modern Device Current Sensor
Incorporates two hall effect sensors to measure magnetic fields from current carrying wire


E11
E13
E15
E17
/current

Custom Power Sensor Assembled

CAD of Custom Environmental Sensor Enclosure





External Hex Bolt

Bosch BME280 Sensor

Hex Socket Screw

Back cover of enclosure
with i) mountings protusions for
MCU and sensor

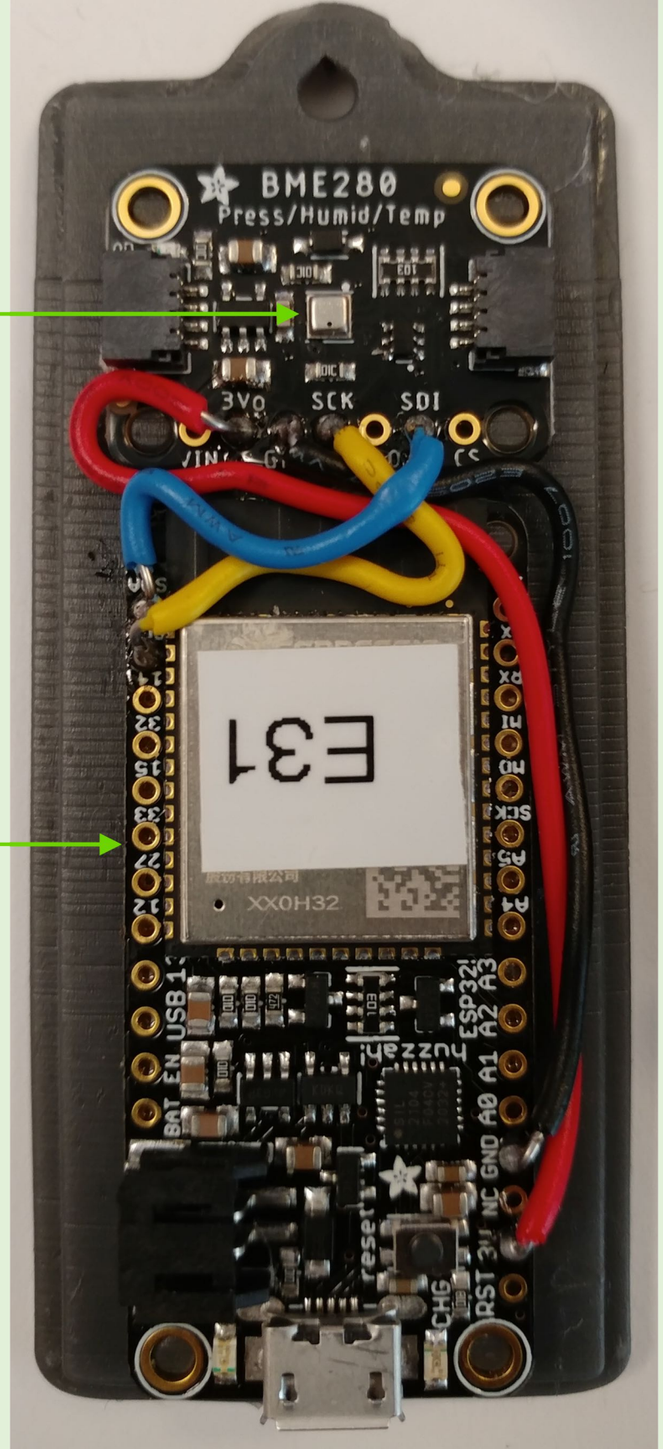
ESP32 Microcontroller Unit

Front cover of enclosure
with i) pass-throughs for LED
indicators, sensor, and ventillation,
and ii) protruded Reset button flap

Environmental Sensor Assembled

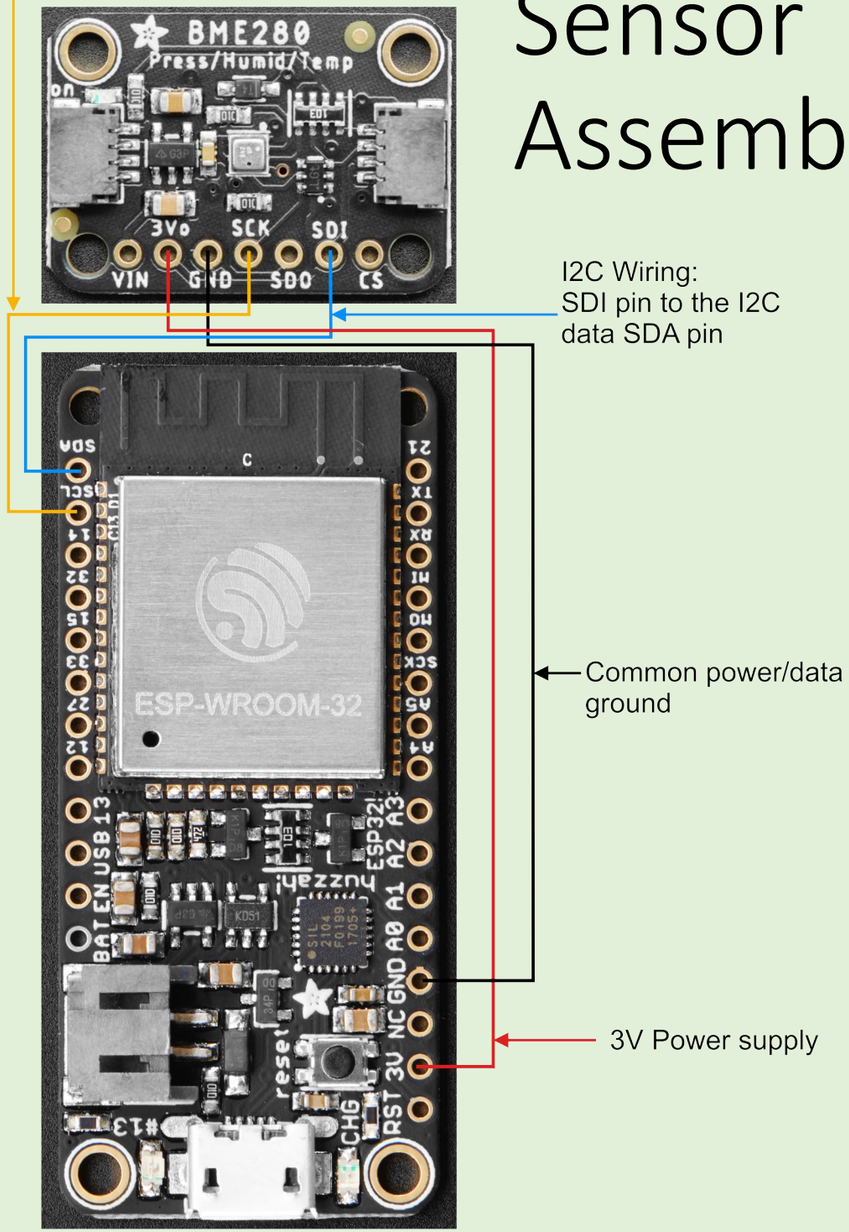
Bosch BME280 Sensor Environmental sensor of: temperature (±1.0°C accuracy), barometric pressure (±1 hPa absolute accuracy) and humidity (±3% accuracy)

Microcontroller Unit ESP32 w/ WiFi and Bluetooth Classic/LE support



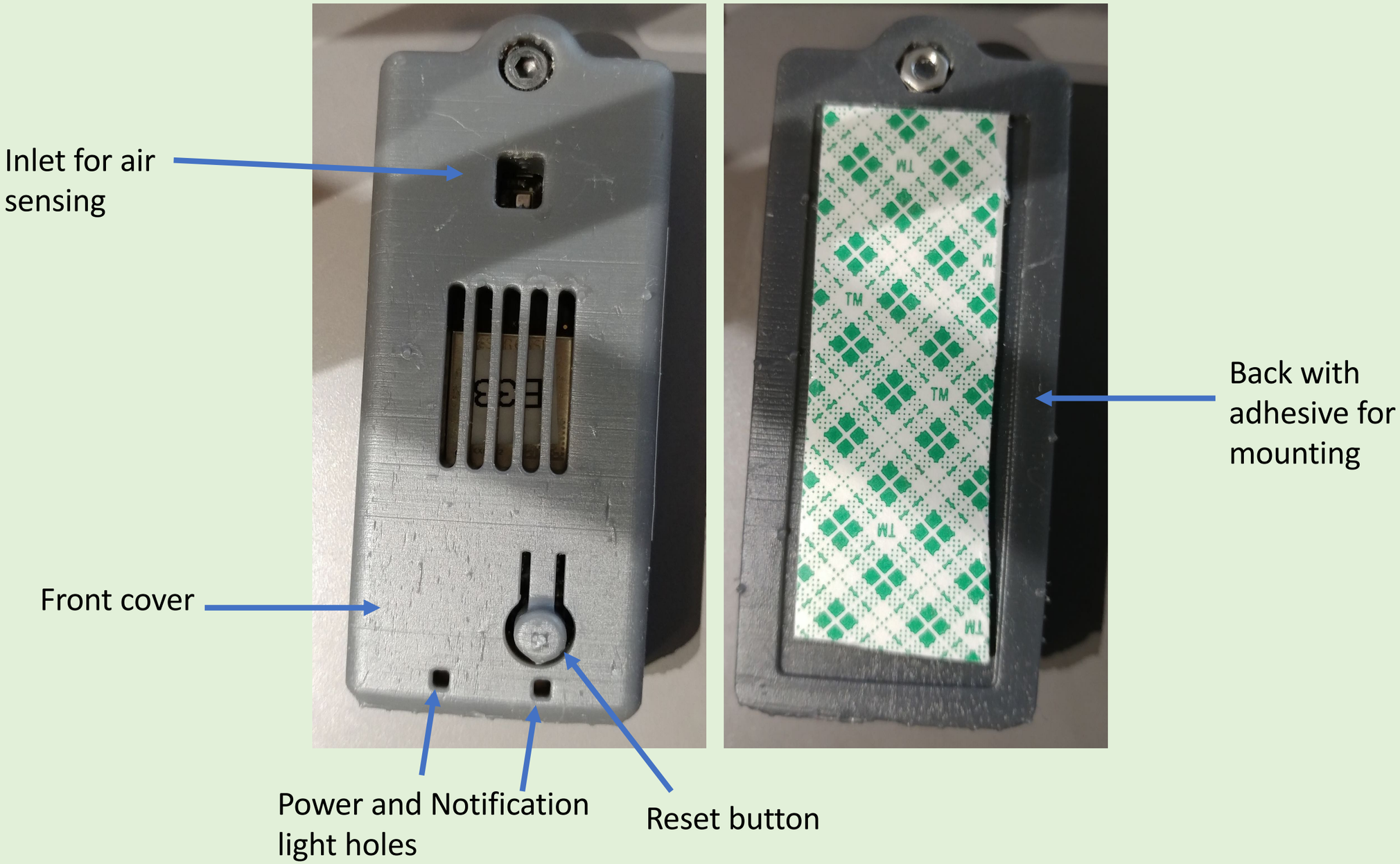
I2C Wiring:
SCK pin to the I2C clock SCL pin

I2C Wiring:
SDI pin to the I2C data SDA pin

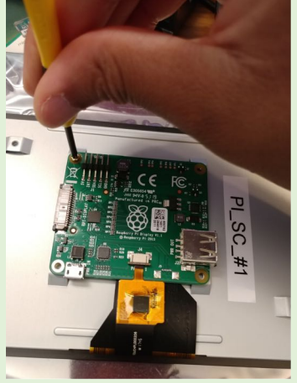


Common power/data ground

3V Power supply


Inlet for air sensing
Front cover
Power and Notification light holes
Reset button
Back with adhesive for mounting

Assembly of Raspberry Pi 4 powered kiosk used to display user interface for end user in the student dorms



Mounting of LCD 7 inch Touchscreen PCB Controller

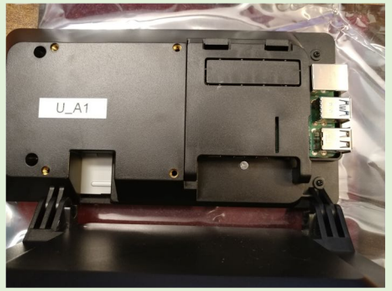
Attachment of LCD Ribbon Cable that carries video signal

Selection of Raspberry Pi 4 (4GB) Single Board Computer in Box

Insertion of storage drive to Raspberry Pi 4 (4GB) Computer, and labelling of the device for deployment

Assembly of back cover for the enclosure of the Touchscreen Kiosk

Connecting the Raspberry Pi 4 computer to the LCD Screen data cable



Final assembly and labelling of Raspberry Pi 4 Touchscreen Kiosk for deployment

One of many assembled touchscreen kiosks actively deployed and in use

User Interface

Indoor Conditions from Environmental Sensor

Graph container that cycles through: i) last 24 hours of hourly campus electricity CO2 intensity, ii) last 24 hours of hourly dorm room energy use automatically or on user selection

Weather Data From OpenWeather API

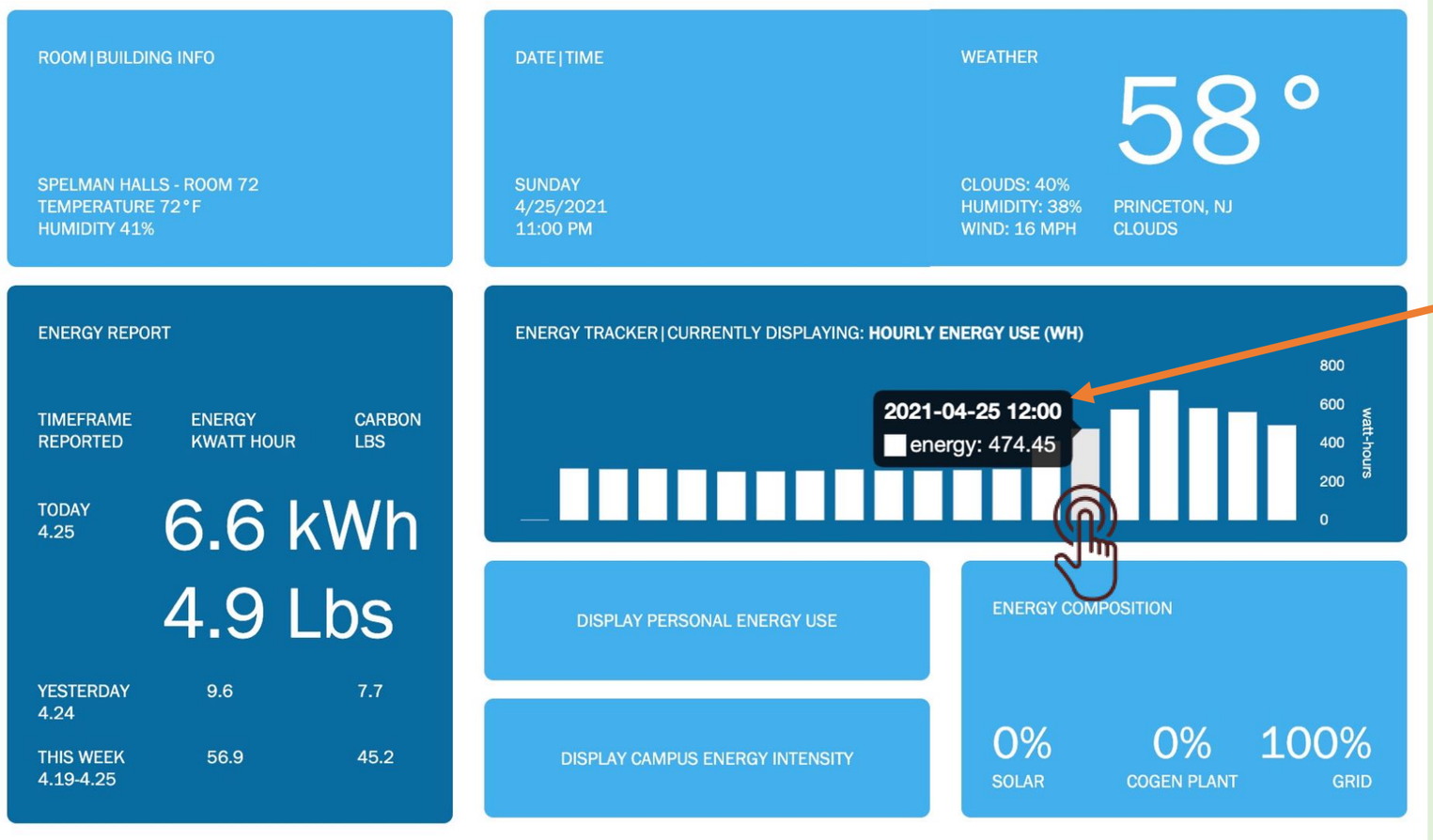
Energy Report includes: i) time-based Energy usage data from MCU Power Sensors, and ii) the associated CO2 footprint

Button to select graph of the last 24 hours of hourly campus electricity CO2 intensity

Button to select graph of the last 24 hours of hourly dorm room energy use

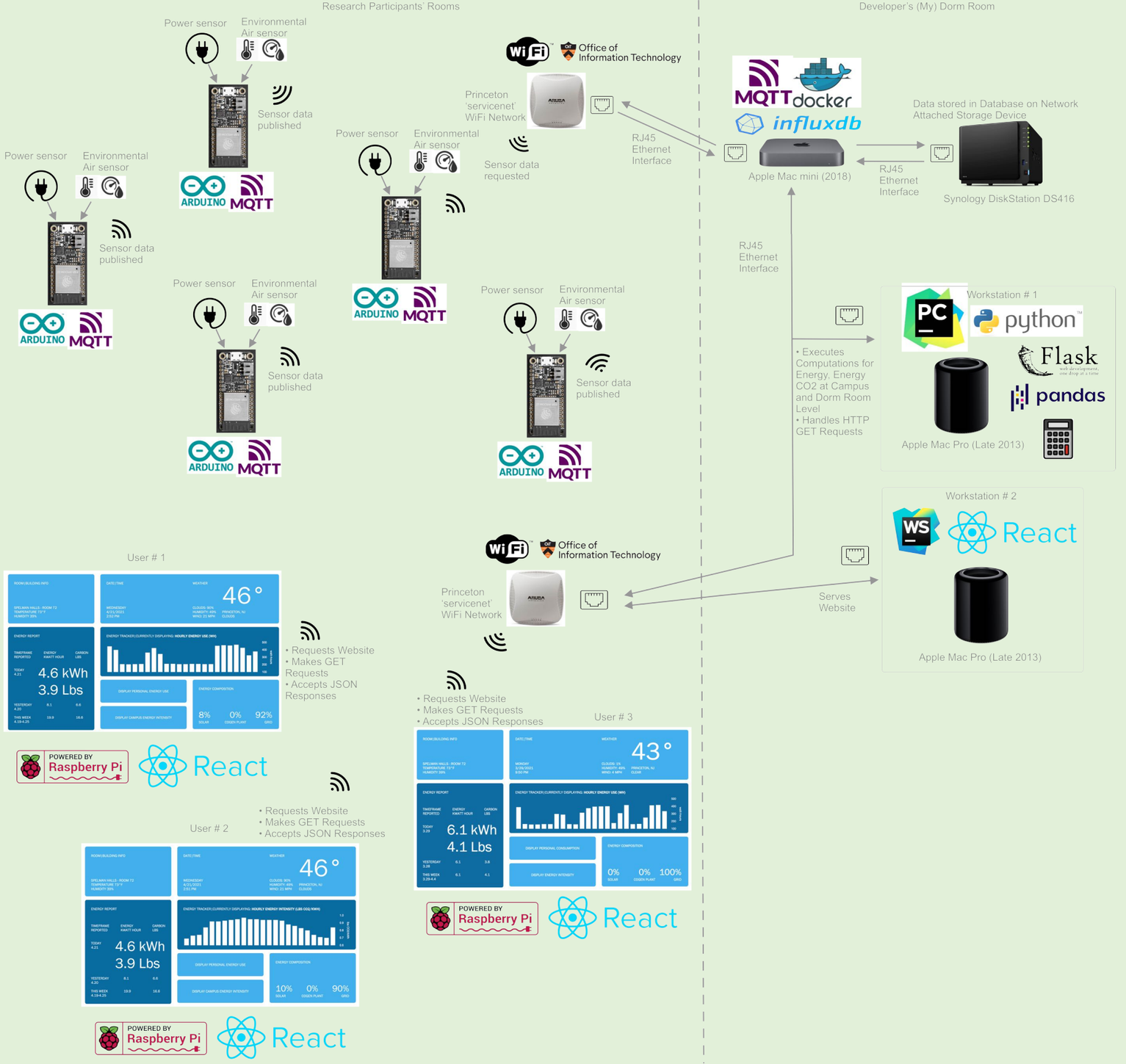
Real-time Campus Power Composition

Graph section of User Interface displaying Hourly Campus Electricity CO2 Intensity for the last 24 hours. Each bar in the graph can be selected by touch to uncover raw values for the selected hour.



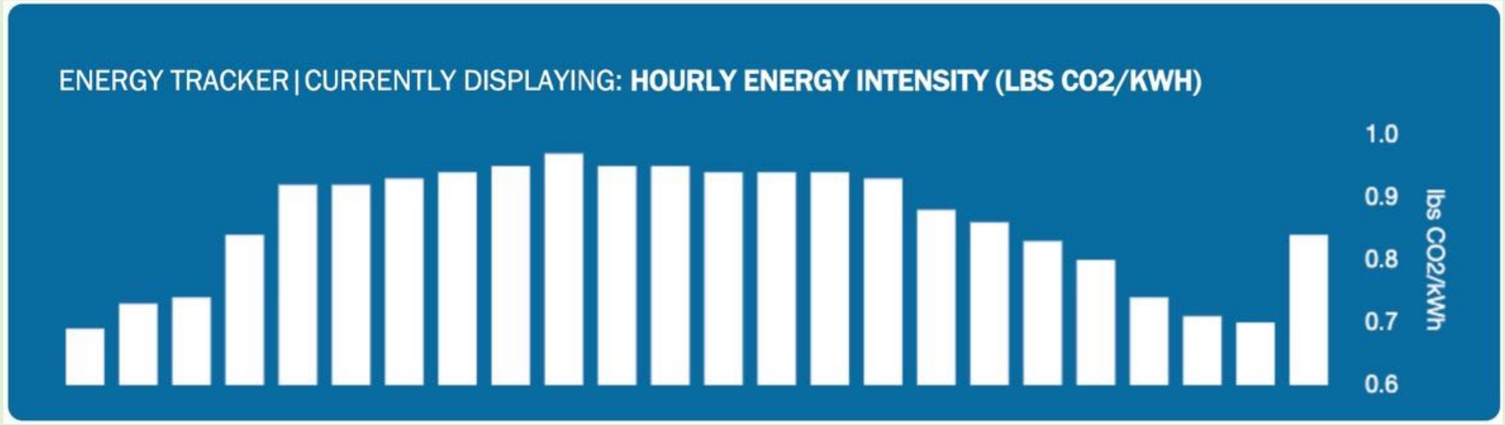
Graph section of User Interface displaying Hourly Energy Use for the last 24 hours. Each bar in the graph can be selected by touch to uncover raw values for the selected hour.

User Interface



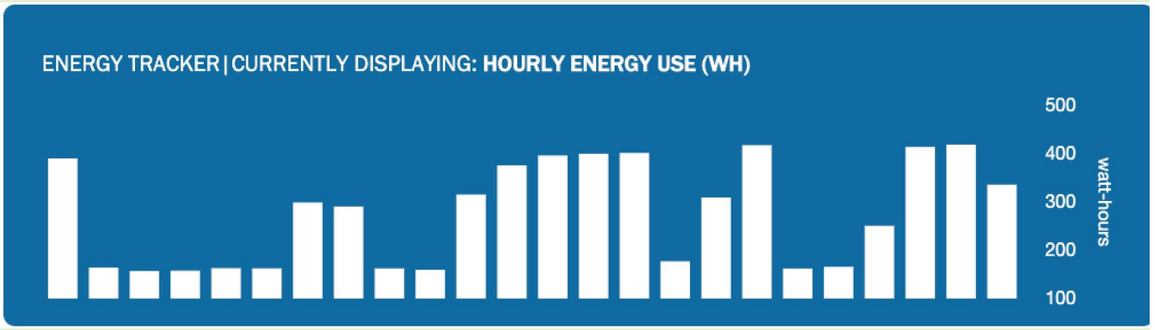
User Interface Back End


```
// Campus Power composition (not user specific):

// Root of API request
// http://140.180.133.113:4545/

// Full API request format
// root + /api/v1/resources/campus_power/current
// Example:
// http://140.180.133.113:4545/api/v1/resources/campus_power/current

// Response

{
    "campus_power_composition": {
        "campus_power_plant": 52,
        "grid": 47,
        "solar": 1,
        "unit_name": "percent",
        "unit_symbol": "%"
    },
    "timestamp_utc": "2021-02-22T14:55:45+00:00"
}
```

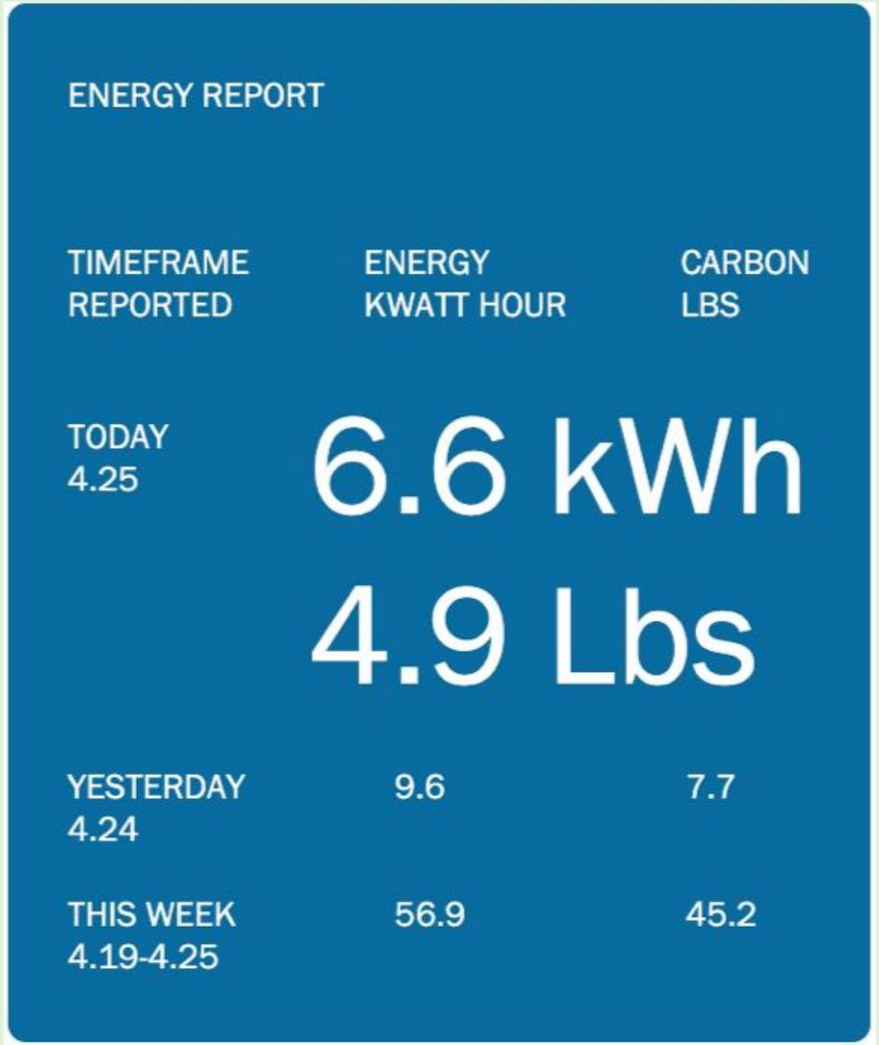



```
// Campus Power Emission Intensity (Graph) (not user specific)

// Root of API request
// http://140.180.133.113:4545/



// Full API request format
// root + /api/v1/resources/campus_energy_intensity/24hr
// Example:
// http://140.180.133.113:4545/api/v1/resources/campus_energy_intensity/24hr

// Response

{
    "campus_energy_intensity": {
        "2021-03-14T18:00:00": 0.58,
        "2021-03-14T19:00:00": 0.6,
        "2021-03-14T20:00:00": 0.61,
        "2021-03-14T21:00:00": 0.63,
        "2021-03-14T22:00:00": 0.61,
        "2021-03-14T23:00:00": 0.61,
        "2021-03-15T00:00:00": 0.6,
        "2021-03-15T01:00:00": 0.59,
        "2021-03-15T02:00:00": 0.59,
        "2021-03-15T03:00:00": 0.61,
        "2021-03-15T04:00:00": 0.62,
        "2021-03-15T05:00:00": 0.63,
        "2021-03-15T06:00:00": 0.64,
        "2021-03-15T07:00:00": 0.64,
        "2021-03-15T08:00:00": 0.56,
        "2021-03-15T09:00:00": 0.48,
        "2021-03-15T10:00:00": 0.47,
        "2021-03-15T11:00:00": 0.47,
        "2021-03-15T12:00:00": 0.48,
        "2021-03-15T13:00:00": 0.48,
        "2021-03-15T14:00:00": 0.47,
        "2021-03-15T15:00:00": 0.46,
        "2021-03-15T16:00:00": 0.49,
        "2021-03-15T17:00:00": 0.55,
        "2021-03-15T18:00:00": 0.64
    },
    "daily_graph_data_legend": "EPT_DateTime : lbs CO2/Wh",
    "timestamp_utc": "2021-03-15T22:24:09+00:00",
    "units": "lbs CO2/Wh"
}
```

```python
''' Function to get last 24 average hourly values for energy intensity of Princeton University.'''
def query_influxdb_campus_energy_emission_intensity() :
    new_list_of_values = []  # Initialize list
    data_dict_dt_lbsco2_mwh = { }  # Initialize dictionary
    query = influxdb_query_builder_campus_energy_intensity()
    query_str = str(query)
    campus_CO2_KWH_df = pd.DataFrame(influxdb_client.query(query_str).get_points())

    # convert the 'time' column to datetime format
    # Source: https://www.geeksforgeeks.org/convert-the-column-type-from-string-to-datetime-format-in-pandas-dataframe/
    campus_CO2_KWH_df['time'] = pd.to_datetime(campus_CO2_KWH_df['time'])

    # Set index of dataframe to column "time"
    campus_CO2_KWH_df.set_index('time' , inplace=True)

    # Change index timezone from UTC to 'US/Eastern'
    campus_CO2_KWH_df.index = campus_CO2_KWH_df.index.tz_convert('US/Eastern')

    # Pandas Resample - Calculations for the current day
    df_day = campus_CO2_KWH_df.resample("H").mean()

    # Format index times as string in format '%Y-%m-%dT%H:%M:%S' e.g 2021-03-14T16:00:00 and makes a list
    list_of_times = df_day.index.strftime('%Y-%m-%d %H:%M').tolist()[-24 :]

    # Makes a list of the values
    list_of_values = df_day['value'].tolist()[-24 :]

    # For loop to convert float values to 2 significant figures
    for value in list_of_values :
        value = round(float(value) , 2)
        new_list_of_values.append(value)

    # to convert lists to dictionary using zip()
    data_dict_dt_lbsco2_mwh = dict(zip(list_of_times , new_list_of_values))

    # Data dictionary to return
    data_dict_to_return = {
        "campus_energy_intensity" : data_dict_dt_lbsco2_mwh,
        "units" : "lbs CO2/Wh",
        "daily_graph_data_legend" : "EPT_DateTime : lbs CO2/Wh"
    }
    return data_dict_to_return
```




```
''' Function to get last 24 average hourly values for energy used in dorm room.'''
# Function to query InfluxDB Dorm Room (DR) Energy data
def query_influxdb_dorm_24hr_energy_data(sensor_name) :
    new_list_of_values_24hr_wh_sum = []  # Initialize list
    # Create query string for query to be made to InfluxDB
    query = influxdb_query_builder_dorm_energy(sensor_name)
    query_str = str(query)

    dorm_energy_df = pd.DataFrame(influxdb_client.query(query_str).get_points())


    # convert the 'time' column to datetime format
    dorm_energy_df['time'] = pd.to_datetime(dorm_energy_df['time'])

    # Set index of dataframe to column "time"
    dorm_energy_df.set_index('time' , inplace=True)

    # Change index timezone from UTC to 'US/Eastern'
    dorm_energy_df.index = dorm_energy_df.index.tz_convert('US/Eastern')

    # Pandas Resample
    # For 24 hour graph data
    df_day_24hr_mean = dorm_energy_df.resample("H").sum()

    # Format index times as string in format '%Y-%m-%dT%H:%M:%S' e.g 2021-03-14T16:00:00 and makes a list of last 25 values
    # Python | Get last N elements from given list
    list_of_times_24hr_sum = df_day_24hr_mean.index.strftime('%Y-%m-%d %H:%M').tolist()[-24 :]

    # Makes a list of last 24 values
    list_of_values_24hr_sum_kwh = df_day_24hr_mean['value'].tolist()[-24 :]

    # For loop to convert float values to 2 significant figures
    for value in list_of_values_24hr_sum_kwh :
        value_wh = value * 1000
        value_wh = round(float(value_wh) , 3)
        new_list_of_values_24hr_wh_sum.append(value_wh)

    # to convert lists to dictionary using zip()
    data_dict_dt_wh = dict(zip(list_of_times_24hr_sum , new_list_of_values_24hr_wh_sum))
    return data_dict_dt_wh


# Function to combine all sensor data and return last set of values
def dorm_daily_data_combination_u_kr() :
    list_of_dict_data = []  # Initialize list

    for name in list_of_sensor_names :
        list_of_dict_data.append(query_influxdb_dorm_24hr_energy_data(name))

    df_with_dorm_sensor_dt_wh = pd.DataFrame(list_of_dict_data)
    # Use the T attribute or the transpose() method to swap (= transpose) the rows and columns of pandas.DataFrame.
    # Neither method changes the original object, but returns a new object with the rows and columns swapped (= transposed
object).
    transposed_df_with_dorm_sensor_dt_wh = df_with_dorm_sensor_dt_wh.T

    # Makes a new column titled "Sum" that is the sum of all sensor values
    # Rounds sum values to 2 significant figures
    transposed_df_with_dorm_sensor_dt_wh['Sum'] = round(transposed_df_with_dorm_sensor_dt_wh.sum(axis=1) , 2)

    # Makes a new dictionary with timestamp and sum value data
    daily_24hr_sum_energy = dict(zip(transposed_df_with_dorm_sensor_dt_wh.to_dict('index') ,
transposed_df_with_dorm_sensor_dt_wh['Sum']))

    # Data dictionary to return
    data_dict_to_return = {
        "daily_graph_energy_wh_data" : daily_24hr_sum_energy,
        "daily_graph_data_legend" : "EPT_DateTime : Watt-Hour"
    }
    return data_dict_to_return
```




```python
''' Function to get sum of Dorm Room Energy from InfluxDB. '''
def query_influxdb_dorm_energy_data_u_kr() :
    # Initialize Variables
    total_kwh_today = 0.0
    total_kwh_yesterday = 0.0
    total_kwh_week = 0.0
    total_kwh_month = 0.0

    total_wh_today = 0.0
    total_wh_yesterday = 0.0
    total_wh_week = 0.0
    total_wh_month = 0.0

    return_data_dict = { }  # Initialize dictionary

    for name in list_of_sensor_names :
        # Create query string for query to be made to InfluxDB
        query = influxdb_query_builder_dorm_energy(name)
        query_str = str(query)
        dorm_energy_df = pd.DataFrame(influxdb_client.query(query_str).get_points())

        # convert the 'time' column to datetime format
        dorm_energy_df['time'] = pd.to_datetime(dorm_energy_df['time'])

        # Set index of dataframe to column "time"
        dorm_energy_df.set_index('time' , inplace=True)

        # Change index timezone from UTC to 'US/Eastern'
        dorm_energy_df.index = dorm_energy_df.index.tz_convert('US/Eastern')

        # Pandas Resample
        # Calculations for the current day
        df_day = dorm_energy_df.resample("D").sum()
        dorm_kwh_last_day = df_day.index.tolist()[-1]
        dorm_kwh_last_day_value = round(float(df_day['value'].tolist()[-1]) , 3)
        total_kwh_today = total_kwh_today + dorm_kwh_last_day_value

        # Calculations for the previous day
        dorm_kwh_previous_day_value = round(float(df_day['value'].tolist()[-2]) , 3)
        total_kwh_yesterday = total_kwh_yesterday + dorm_kwh_previous_day_value

        # Calculations for the week
        df_week = dorm_energy_df.resample("W").sum()
        dorm_kwh_last_week = df_week.index.tolist()[0]
        dorm_kwh_last_week_value = round(float(df_week['value'].tolist()[-1]) , 3)
        total_kwh_week = total_kwh_week + dorm_kwh_last_week_value

        # Calculations for the month
        df_month = dorm_energy_df.resample("M").sum()
        dorm_kwh_last_month = df_month.index.tolist()[-1]
        dorm_kwh_last_month_value = round(float(df_month['value'].tolist()[-1]) , 3)
        total_kwh_month = total_kwh_month + dorm_kwh_last_month_value

    # Converts KiloWattHour to WattHour
    total_wh_today = round((total_kwh_today * 1000))
    total_wh_yesterday = round((total_kwh_yesterday * 1000))
    total_wh_week = round((total_kwh_week * 1000))
    total_wh_month = round((total_kwh_month * 1000))

    # Round values to 1 significant figure for Kilowatthour values
    total_kwh_today = round(total_kwh_today , 1)
    total_kwh_yesterday = round(total_kwh_yesterday , 1)
    total_kwh_week = round(total_kwh_week , 1)
    total_kwh_month = round(total_kwh_month , 1)

    # Data dictionary to return
    data_dict = {
        "energy_report_dorm_power" : {
            'total_WH_today' : total_wh_today ,
            'total_KWH_today' : total_kwh_today ,
            'total_WH_yesterday' : total_wh_yesterday ,
            'total_KWH_yesterday' : total_kwh_yesterday ,
            'total_WH_week' : total_wh_week ,
            'total_KWH_week' : total_kwh_week ,
            'total_WH_month' : total_wh_month ,
            'total_KWH_month' : total_kwh_month ,
            'unit_WH' : "Watt-Hour" ,
            'unit_KWH' : 'Kilowatt-Hour'
        }
    }
    return data_dict
```

```
''' Function to get sum of Dorm Room Energy CO2 from InfluxDB. '''
def query_influxdb_dorm_co2_data_u_kr() :
    # Initialize Variable
    total_co2_today = 0.0
    total_co2_yesterday = 0.0
    total_co2_week = 0.0
    total_co2_month = 0.0

    return_data_dict = { }  # Initialize dictionary

    # Each loop is done for a sensor's data
    for name in list_of_sensor_names :
        # Create query string for query to be made to InfluxDB
        query = influxdb_query_builder_dorm_co2(name)
        query_str = str(query)
        dorm_co2_df = pd.DataFrame(influxdb_client.query(query_str).get_points())

        # convert the 'time' column to datetime format
        dorm_co2_df['time'] = pd.to_datetime(dorm_co2_df['time'])

        # Set index of dataframe to column "time"
        dorm_co2_df.set_index('time' , inplace=True)

        # Change index timezone from UTC to 'US/Eastern'
        dorm_co2_df.index = dorm_co2_df.index.tz_convert('US/Eastern')

        # Pandas Resample
        # Calculations for the current day
        df_day = dorm_co2_df.resample("D").sum()
        dorm_co2_last_day = df_day.index.tolist()[-1]  # Date Today (Last day)
        dorm_co2_last_day_value = round(float(df_day['value'].tolist()[-1]) , 3)  # C02 Today (Last day)
        total_co2_today = total_co2_today + dorm_co2_last_day_value

        # Calculations for the previous day
        dorm_co2_previous_day_value = round(float(df_day['value'].tolist()[-2]) , 3)  # C02 Yesterday (Previous day)
        total_co2_yesterday = total_co2_yesterday + dorm_co2_previous_day_value

        # Calculations for the week
        df_week = dorm_co2_df.resample("W").sum()
        dorm_co2_last_week = df_week.index.tolist()[0]
        dorm_co2_last_week_value = round(float(df_week['value'].tolist()[-1]) , 3)
        total_co2_week = total_co2_week + dorm_co2_last_week_value

        # Calculations for the month
        df_month = dorm_co2_df.resample("M").sum()
        dorm_co2_last_month = df_month.index.tolist()[-1]
        dorm_co2_last_month_value = round(float(df_month['value'].tolist()[-1]) , 3)
        total_co2_month = total_co2_month + dorm_co2_last_month_value

    # Round values to 1 significant figure
    total_co2_today = round(total_co2_today , 1)
    total_co2_yesterday = round(total_co2_yesterday , 1)
    total_co2_week = round(total_co2_week , 1)
    total_co2_month = round(total_co2_month , 1)

    data_dict = {
        'energy_report_co2' : {
            'total_co2_today' : total_co2_today ,
            'total_co2_yesterday' : total_co2_yesterday ,
            'total_co2_week' : total_co2_week ,
            'total_co2_month' : total_co2_month ,
            'unit_name' : "pound" ,
            'unit_symbol' : 'lb'
        }
    }
    return data_dict
```

```
// Energy Report Information
// Includes: Dorm room Power usage, Dorm room CO2 Footprint usage, Energy Report Timeframe labels (user specific)

// Root of API request
// http://140.180.133.113:4545/

// User in this case is u/kr/
// Full API request format
// root + /api/v1/resources/u/kr/dorm_power_co2/current
// Example:
// http://140.180.133.113:4545/api/v1/resources/u/kr/dorm_power_co2/current

// Response
```


```
{
    "daily_graph_energy_wh_data": {
        "2021-03-14T18:00:00": 3.36,
        "2021-03-14T19:00:00": 1.89,
        "2021-03-14T20:00:00": 1.89,
        "2021-03-14T21:00:00": 1.88,
        "2021-03-14T22:00:00": 1.84,
        "2021-03-14T23:00:00": 1.9,
        "2021-03-15T00:00:00": 1.87,
        "2021-03-15T01:00:00": 1.86,
        "2021-03-15T02:00:00": 1.9,
        "2021-03-15T03:00:00": 1.86,
        "2021-03-15T04:00:00": 1.86,
        "2021-03-15T05:00:00": 1.87,
        "2021-03-15T06:00:00": 1.86,
        "2021-03-15T07:00:00": 1.85,
        "2021-03-15T08:00:00": 1.88,
        "2021-03-15T09:00:00": 1.88,
        "2021-03-15T10:00:00": 3.61,
        "2021-03-15T11:00:00": 4.33,
        "2021-03-15T12:00:00": 4.54,
        "2021-03-15T13:00:00": 4.51,
        "2021-03-15T14:00:00": 4.51,
        "2021-03-15T15:00:00": 4.48,
        "2021-03-15T16:00:00": 4.5,
        "2021-03-15T17:00:00": 4.41,
        "2021-03-15T18:00:00": 4.42
    },
    "energy_report_co2": {
        "total_co2_month": 52.8,
        "total_co2_today": 2.6,
        "total_co2_week": 2.6,
        "total_co2_yesterday": 3.1,
        "unit_name": "pound",
        "unit_symbol": "lb"
    },
    "energy_report_dorm_power": {
        "total_KWH_month": 92.4,
        "total_KWH_today": 4.9,
        "total_KWH_week": 4.9,
        "total_KWH_yesterday": 6.3,
        "total_WH_month": 92386,
        "total_WH_today": 4906,
        "total_WH_week": 4906,
        "total_WH_yesterday": 6294,
        "unit_KWH": "Kilowatt-Hour",
        "unit_WH": "Watt-Hour"
    },
    "energy_report_timeframe": {
        "this_month": "3.2021",
        "this_month_fmt": "MM.YYYY",
        "this_week": "3.15-3.21",
        "this_week_fmt": "MM.DD-MM.DD",
        "today": "3.15",
        "today_fmt": "MM.DD",
        "yesterday": "3.14",
        "yesterday_fmt": "MM.DD"
    },
    "daily_graph_data_legend": "EPT_DateTime : Watt-Hour",
    "timestamp_utc": "2021-03-15T22:13:15+00:00"
}
```


```
// Air Sensor Information
// Includes: Dorm room Temperature, Relative Humidity, Sensor Location (Building & Dorm location) (user specific):

// Root of API request
// http://140.180.133.113:4545/

// User in this case is u/kr/
// Full API request format
// root + /api/v1/resources/u/kr/dormroom_enviro/current
// Example:
// http://140.180.133.113:4545/api/v1/resources/u/kr/dormroom_enviro/current

// Response

{
    "room_building_info": {
        "building_name": "SPELMAN HALLS",
        "full_room_location": "SPELMAN HALLS - ROOM 72",
        "humidity_units": "percent",
        "humidity_units_symbol": " %",
        "humidity_value": 39,
        "room_number": "72",
        "sensor_timestamp_utc": "2021-02-22T16:21:34+00:00",
        "temperature_units": "Fahrenheit",
        "temperature_units_symbol": " °F",
        "temperature_value": 73
    },
    "timestamp_utc": "2021-02-22T16:21:34+00:00"
}
```

Research Study Plan

Aware Cohort:

- Power Sensors
- Interactive touchscreen kiosk provided
- n = 4 students
- Duration = 2 weeks
- Survey at the beginning and end to assess attitudes

Blind Cohort:

- Power Sensors (only)
- No Interactive touchscreen kiosk provided
- n = 4 students
- Duration = 2 weeks
- Survey at the beginning and end to assess attitudes

Slide intentionally left blank