Lab 5: Cyber-Physical Systems and Visualization

Exercise 3: Simple Report

The js and html code for the exercise is...

extend_dashboard_links.html

```
<!--extend dashboard links, Assembled and Modified by John Stuart-->
  <li>
    <a href="#" id="sidebar-input" class="first-level"><i class="fa
    fa-tags fa-fw"></i> Input</a>
  </li> <!--input link-->
  <li>
    <a href="#" id="sidebar-report" class="first-level"><i class="fa
    fa-area-chart fa-fw"></i> Report</a>
  </li> <!--report link-->
```

extend_dashboard_pages.html

```
<!--extend dashboard pages, Assembled and Modified by John Stuart-->
 <!--Defines layout of input page-->
      <div class="page hidden" id="page-input">
        <div id="page-wrapper">
          <div class="row">
            <div class="col-lg-12">
              <h1 class="page-header">Input</h1>
            </div>
            <!-- /.col-lg-12 -->
          </div>
          <!-- /.row -->
          <div class="row">
            <div id="content-input" class="col-lg-8">

  <!--inserts first name and last name form into page-->
              <form id="form-input">
                <fieldset class="form-group">
                  <label for="first-name">First Name</label>
                  <input type="text" class="form-control"
name="first-name" placeholder="" required="true">
                </fieldset>
                <fieldset class="form-group">
                  <label for="last-name">Last Name</label>
                  <input type="text" class="form-control"
name="last-name" placeholder="" required="true">
                </fieldset>
                <button type="submit" class="btn">Submit</button>
              </form>


            </div>
```

```
                <!-- /.col-lg-8 -->
                <div class="col-lg-2 note"></div>
                <!-- /.col-lg-2 -->
            </div>
            <!-- /.row -->

        </div>
        <!-- /#page-wrapper -->
    </div>
    <!-- /#page-input -->




<!--Defines layout of report page-->
    <div class="page hidden" id="page-report">
        <div id="page-wrapper">
            <div class="row">
                <div class="col-lg-12">
                    <h1 class="page-header">Report</h1>
                </div>
                <!-- /.col-lg-12 -->
            </div>
            <!-- /.row -->
            <div class="row">
                <div id="content-report" class="col-lg-8">


                </div>
                <!-- /.col-lg-8 -->
                <div class="col-lg-2 note"></div>
                <!-- /.col-lg-2 -->
            </div>
            <!-- /.row -->

        </div>
        <!-- /#page-wrapper -->
    </div>
    <!-- /#page-input -->
```

extend_dashboard.js

```
//Extend_Dashboard, Assembled and Modified by John Stuart
//loads etent_dashboard_links
var ul = $('ul#side-menu');
$.ajax({
  url : '/static/extend_dashboard_links.html',
```

```
  type: "get",
  success : function(response){
    console.log("Load /static/extend_dashboard_links.html");
    ul.append(response);
  }
});

//loads the pages defined in extend_dashboard_pages
var wrapper = $('div#wrapper');
$.ajax({
  url : '/static/extend_dashboard_pages.html',
  type: "get",
  success : function(response){
    console.log("Load /static/extend_dashboard_pages.html");
    wrapper.append(response);

    // Form submit call goes here.
    $("form#form-input").submit( onInputFormSubmit );
  }
});

///*
//  Add function to get points for report page
//*/

//takes points from wallflower_demo for use in plotting later
function getPoints( the_network_id, the_object_id, the_stream_id, callback ){
  var query_data = {};
  var query_string = '?'+$.param(query_data);
  var url = '/networks/'+the_network_id+'/objects/'+the_object_id;
  url += '/streams/'+the_stream_id+'/points'+query_string;

  // Send the request to the server
  $.ajax({
    url : url,
    type: "get",
    success : function(response){
      console.log( response );

      if( response['points-code'] == 200 ){
        var num_points = response.points.length
        var most_recent_value = response.points[0].value
        console.log("Most recent value: "+most_recent_value);
        console.log("Number of points retrieved: "+num_points);
        callback( response.points );
      }
    },
    error : function(jqXHR, textStatus, errorThrown){
      console.log(jqXHR);
    }
  });
}
```

```
// Call getPoints if Input or Report is selected
// ...added feature to dynamically update plot as new data becomes available
custom_sidebar_link_callback = function( select ){

 if (select == 'input') {

 }
 else if (select == 'report'){
  var plotCalls = 0;
  var plotTimer = setInterval( function(){
   getPoints('local','test-object','test-stream', function(points){
    console.log( "The points request was successful!" );
    loadPlot( points );
   });
   if( plotCalls > 20 ){
    console.log( 'Clear timer' );
    clearInterval( plotTimer );
   }else{
    plotCalls += 1;
   }
  }, 1000);
 }
}


/*
   Function to plot data points using Highcharts
*/
function loadPlot( points ){
 var plot = $('#content-report');
 // Check if plot has a Highcharts element
 if( plot.highcharts() === undefined ){
  // Create a Highcharts element
  plot.highcharts( report_plot_options );
 }

 // Iterate over points to place in Highcharts format
 var datapoints = [];
 for ( var i = 0; i < points.length; i++){
  var at_date = new Date(points[i].at);
  var at = at_date.getTime() - at_date.getTimezoneOffset()*60*1000;
  datapoints.unshift( [ at, points[i].value] );
 }

 // Update Highcharts plot
 if( plot.highcharts().series.length > 0 ){
  plot.highcharts().series[0].setData( datapoints );
 }else{
  plot.highcharts().addSeries({
   name: "Series Name Here",
   data: datapoints
  });
```

```
  }
}

var report_plot_options = {
   chart: {
      type: 'spline'
   },
   xAxis: {
     type: 'datetime',
     dateTimeLabelFormats: { // don't display the dummy year
        month: '%e. %b',
        year: '%b'
     },
   },
};




/*
  Add functionality to the input page form
*/
function onInputFormSubmit(e){
  e.preventDefault();

  var object_id = "obj-names";
  var stream_id = "stm-form-input";

  // Gather the data
  // and remove any undefined keys
  var data = {};
  $('input',this).each( function(i, v){
    var input = $(v);
    data[input.attr("name")] = input.val();
  });
  delete data["undefined"];

  console.log( data );

    var url = '/networks/'+network_id+'/objects/';
  url = url + object_id+'/streams/'+stream_id+'/points';
  var query = {
    "points-value": JSON.stringify( data )
  };

  // Send the request to the Pico server
  $.ajax({
    url : url+'?'+$.param(query),
    type: "post",
    success : function(response){
      var this_form = $("form#form-input");

      if( response['points-code'] == 200 ){
```

```
        console.log("Success");
        // Clear the form
        this_form.trigger("reset");
      }
// Log the response to the console
      console.log(response);
    },
    error : function(jqXHR, textStatus, errorThrown){
      // Do nothing
    }
  });

};
```

Exercise 4: Putting it All Together

The javascript, python, and html code for the exercise is...

SendData.ino

```
/*
  SendData (CPS)
  by John Stuart
  CE 186


Personal environment sensor and actuator. Reads the output of a photosensor
and an analog temperature sensor and sends that data through the hardware serial port
for handling and processing by python every 10 seconds. It also listens to for data from
the software serial and sets the brightness of an LED based on the expected thermal and
daylight acceptabiliy of the space. This LED could be a proxy for a switch to set the
power level of a fan or HVAC system.

In this system thermal and visual comfort is measured, but CO2 and PM could be measured
instead of light in order to tie results to health or productivity impacts.


*/
int led = 9;
int brightness = 0;
int sensorValueTemp = 0;
int sensorValuePho = 0;
char healthWarning;
int healthLevel = 1;
int count = 0;
int mV = 0;
int res = 0;
int tempC=0;
int incomingByte = 0;
int lux = 0;
```

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  pinMode(led, OUTPUT);  // Initialize the digital pins as an output.
  //pinMode(check, INPUT);  //initialize switch input
}

// the loop routine runs over and over again forever:
void loop() {



  // reads and sends sensor data
  sensorValuePho = analogRead(A0);   //read photresistor analog output
  sensorValueTemp = analogRead(A1);  //read temperature sensor analog output

  //Temperature sensor output is on a linear scale. It is converted degrees C by the arduino before
sending
  mV = map(sensorValueTemp, 0, 1023, 0, 5000); //convert to mV
  tempC = 0.1*mV-50; //convert to degrees C
  res = ((10230/sensorValuePho)-10);
  lux = 100/(res);


  Serial.println('L');
  Serial.println(lux);
  Serial.println('T');
  Serial.println(tempC);
  Serial.println();




  //checks for input from the serial port and displays current health warning and LED state
  //once every second for 10 seconds, before checking the sensor values again.
  count = 0;
  for(count;count<10;count++){
  // listens and converts incoming data from hardware serial.
  if(Serial.available()) {
        // Read the next incoming byte
        incomingByte = Serial.read();
        if (incomingByte >= 49 & incomingByte <= 51) {
          healthLevel = incomingByte - 48;
        }

        else{
          Serial.println("invalid input");
        }
        Serial.println(healthLevel);

  }
  brightness = map(healthLevel, 1, 3, 0, 255);  //map health level to brightness
```

```
  //sends results of actuation back to the hardware serial to display on the serial monitor
  //if (healthLevel == 1){
  //    healthWarning = "low";
  //}
  //else if (healthLevel == 2){
  //    healthWarning = "moderate";
  //}
  //else {
  //   healthWarning = "HIGH";
  //}

  Serial.write("Health Warning =  ");
  Serial.print(healthLevel);
  Serial.write("   LED Brightness =  ");
  Serial.print(brightness);
  analogWrite(led, brightness);  //change LED brightness
  Serial.println();
  delay(1000);   // delay 1 second
}
}
```

ListenAndSend.py

```
"""
   ListenAndSend
   John Stuart
   Reads temperature and light level data from the arduino and posts that data
   to a data stream on wallflower. Checks server for results and sends
   retrieved health level to arduino to actuate LED to 3 brightness levels
   (LED = off, medium, or high).
"""
# added sys.path block due to python error in finding correct file path
# import serial, json, and requests to read and send data to arduino and server
import sys
sys.path.append(r'C:\Python27\Lib\site-packages')
import serial

sys.path.append(r'C:\Python27\Lib')
import json

sys.path.append(r'C:\Python27\Lib\site-packages\pip\_vendor')
import urllib3
import requests

import time
import datetime
```

```
# Change the port name to match the port
# to which your Arduino is connected.
serial_port_name = 'COM3' # for Windows
ser = serial.Serial(serial_port_name, 9600, timeout=1)

delay = 5 # Delay in seconds

base = 'http://127.0.0.1:5000'
network_id = 'local'
header = {}

#### Delete existing objects ####
#deletes temperature and pho objects

try:

   query = {
         'object-name': 'temp-object'
   }
   endpoint = '/networks/'+network_id+'/objects/temp-object'
   response = requests.request('DELETE', base + endpoint, params=query, headers=header, timeout=120
)
   resp = json.loads( response.text )
   if resp['object-code'] == 201:
      print('Create object temp-object: ok')
   else:
      print('Create object temp-object: error')
      print( response.text )

   query = {
      'object-name': 'pho-object'
   }
   endpoint = '/networks/'+network_id+'/objects/pho-object'
   response = requests.request('DELETE', base + endpoint, params=query, headers=header, timeout=120
)
   resp = json.loads( response.text )
   if resp['object-code'] == 201:
      print('Create object pho-object: ok')
   else:
      print('Create object pho-object: error')
      print( response.text )

   query = {
      'object-name': 'result-object'
   }
   endpoint = '/networks/'+network_id+'/objects/result-object'
   response = requests.request('DELETE', base + endpoint, params=query, headers=header, timeout=120
)
   resp = json.loads( response.text )
   if resp['object-code'] == 201:
      print('Create object result-object: ok')
```

```
    else:
        print('Create object result-object: error')
        print( response.text )
except:
    print ("nothing to delete")




#### Create objects ####
#creates temperature object
query = {
        'object-name': 'temp-object'
}
endpoint = '/networks/'+network_id+'/objects/temp-object'
response = requests.request('PUT', base + endpoint, params=query, headers=header, timeout=120 )
resp = json.loads( response.text )
if resp['object-code'] == 201:
    print('Create object temp-object: ok')
else:
    print('Create object temp-object: error')
    print( response.text )

#creates temperature stream
query = {
    'stream-name': 'temp-stream',
    'points-type': 'i' # 'i', 'f', or 's'
}
endpoint = '/networks/'+network_id+'/objects/temp-object/streams/temp-stream'
response = requests.request('PUT', base + endpoint, params=query, headers=header, timeout=120 )
resp = json.loads( response.text )
if resp['stream-code'] == 201:
    print('Create stream temp-stream: ok')
else:
    print('Create stream temp-stream: error')
    print( response.text )


#creates light level object
query = {
    'object-name': 'pho-object'
}
endpoint = '/networks/'+network_id+'/objects/pho-object'
response = requests.request('PUT', base + endpoint, params=query, headers=header, timeout=120 )
resp = json.loads( response.text )
if resp['object-code'] == 201:
    print('Create object pho-object: ok')
else:
    print('Create object pho-object: error')
    print( response.text )

#creates light level stream
query = {
```

```
    'stream-name': 'pho-stream',
    'points-type': 'i' # 'i', 'f', or 's'
}
endpoint = '/networks/'+network_id+'/objects/pho-object/streams/pho-stream'
response = requests.request('PUT', base + endpoint, params=query, headers=header, timeout=120 )
resp = json.loads( response.text )
if resp['stream-code'] == 201:
    print('Create stream pho-stream: ok')
else:
    print('Create stream pho-stream: error')
    print( response.text )


#creates result object
query = {
        'object-name': 'result-object'
}
endpoint = '/networks/'+network_id+'/objects/result-object'
response = requests.request('PUT', base + endpoint, params=query, headers=header, timeout=120 )
resp = json.loads( response.text )
if resp['object-code'] == 201:
    print('Create object result-object: ok')
else:
    print('Create object result-object: error')
    print( response.text )

#creates result stream
query = {
    'stream-name': 'result-stream',
    'points-type': 'i' # 'i', 'f', or 's'
}
endpoint = '/networks/'+network_id+'/objects/result-object/streams/result-stream'
response = requests.request('PUT', base + endpoint, params=query, headers=header, timeout=120 )
resp = json.loads( response.text )
if resp['stream-code'] == 201:
    print('Create stream result-stream: ok')
else:
    print('Create stream result-stream: error')
    print( response.text )







###### SERVER communication functions ##########
#actual sending function
```

```
def storedata():


    # Set body (also referred to as data or payload). Body is a JSON string.
    #store light level points
    endpoint = '/networks/local/objects/pho-object/streams/pho-stream/points'
    query = {
            'points-value': Pho,
            'points-at': t
    }
    response = requests.request('POST', base + endpoint, params=query, headers=header, timeout=120 )
    resp = json.loads( response.text )
    if resp['points-code'] == 200:
        print( 'Update test-stream points: ok')
    else:
        print( 'Update test-stream points: error')
        print( response.text )


    #store temperature points
    endpoint = '/networks/local/objects/temp-object/streams/temp-stream/points'
    query = {
        'points-value': Temp,
        'points-at': t
    }
    response = requests.request('POST', base + endpoint, params=query, headers=header, timeout=120 )
    resp = json.loads( response.text )
    if resp['points-code'] == 200:
        print( 'Update test-stream points: ok')
    else:
        print( 'Update test-stream points: error')
        print( response.text )



    time.sleep(1)

    print Pho
    print Temp

    time.sleep(1)



# Retrieve the health level results from the server
def retrieveresults():
    # Retrieve (GET) names from the NameServer

    query = {}
    endpoint = '/networks/local/objects/result-object/streams/result-stream/points'
    address = base + endpoint

    # Form and send request. Set timeout to 2 minutes. Receive response.
```

```
    response = requests.request('GET', address, timeout=120 )

    # Text is JSON string. Convert to Python dictionary/list
    response = json.loads( response.text )
    hLev = response['Health Level']
    print hLev
    return hLev




###### ARDUINO communication functions ##########

# Run once at the start
def setup():
    try:
        print 'Setup'
    except:
        print 'Setup Error'


# Run continuously forever
def loop():
    # Check if something is in serial buffer
    if ser.inWaiting() > 0:
        try:
            # Read entire line
            # (until '\n')
            x = ser.readline()
            #print x, type(x)
            dType = x[0]
            if dType == 'L':
                Pho = int(ser.readline())
                print "Recieved Lux:", Pho, type(Pho)
                dType == ser.readline()
                Temp = int(ser.readline())
                print "Recieved Temp:", Temp, type(Temp)
                t =  datetime.datetime.utcnow().strftime("%Y-%m-%dT%H:%M:%S.%fZ")


                #### Post received data to server ####
                #store light level points
                endpoint = '/networks/local/objects/pho-object/streams/pho-stream/points'
                query = {
                        'points-value': Pho,
                        'points-at': t
                }
                response = requests.request('POST', base + endpoint, params=query, headers=header,
timeout=120 )
                resp = json.loads( response.text )
                if resp['points-code'] == 200:
```

```
                print( 'Update test-stream points: ok')
            else:
                print( 'Update test-stream points: error')
                print( response.text )


            #store temperature points
            endpoint = '/networks/local/objects/temp-object/streams/temp-stream/points'
            query = {
                'points-value': Temp,
                'points-at': t
            }
            response = requests.request('POST', base + endpoint, params=query, headers=header,
timeout=120 )
            resp = json.loads( response.text )
            if resp['points-code'] == 200:
                print( 'Update test-stream points: ok')
            else:
                print( 'Update test-stream points: error')
                print( response.text )



            time.sleep(1)

            print Pho
            print Temp

            time.sleep(1)

        else:
            #print dType
            print 'no new data'
            return [-1,-1]


    except:
        print "read/post error"

  # 100 ms delay
  time.sleep(0.1)
  return


# Run continuously forever
# with a delay between calls
def delayed_loop():
  try:
    print "Delayed Loop"
    #every 5 seconds, check the server with get request
    #and send value to arduino for LED actuation
    query = {}
    endpoint = '/networks/local/objects/result-object/streams/result-stream/points'
```

```
        # Form and send request. Set timeout to 2 minutes. Receive response.
        response = requests.request('GET', base + endpoint, params=query, headers=header, timeout=120 )
        print response
        # Text is JSON string. Convert to Python dictionary/list
        resp = json.loads( response.text )
        print resp
        hLev = resp['points'][0]['value']
        print hLev
        #converts health level to an 8 bit string to be read by arduino and sends
        hLevSend = str(hLev)
        ser.write(hLevSend.encode("utf-8"))
    except:
        print "no health level to send"

    #return
# Run once at the end
def close():
    try:
        print "Close Serial Port"
        #ser.close()
    except:
        print "Close Error"



###### MAIN function ##########

def main():
    # Call setup function
    setup()
    # Set start time
    nextLoop = time.time()
    while(True):
        # Try loop() and delayed_loop()
        try:
            data = loop()
            if time.time() > nextLoop:
                # If next loop time has passed...
                nextLoop = time.time() + delay
                delayed_loop()
                print ("LED brightness updated")
            else:
                print ("LED awaiting instructions")
        except KeyboardInterrupt:
            # If user enters "Ctrl + C", break while loop
            break
        except:
            # Catch all errors
            print "Unexpected error."
    # Call close function
    close()

# Run the program
```

```
main()
```

ListenAndProcess.py

```
"""
   ListenAndProcess
   John Stuart
   Retrieves data from the wallflower server, processes it, and sends results
   back to server
"""
# added sys.path block due to python error in finding correct file path
# import serial, json, and requests to read and send data to arduino and server
import sys
sys.path.append(r'C:\Python27\Lib\site-packages')
import serial

sys.path.append(r'C:\Python27\Lib')
import json

sys.path.append(r'C:\Python27\Lib\site-packages\pip\_vendor')
import requests

import time
import datetime



delay = 5 # Delay in seconds

base = 'http://127.0.0.1:5000'
network_id = 'local'
header = {}




###### SERVER communication functions ##########


def storeresults(results, t):


   endpoint = '/networks/local/objects/result-object/streams/result-stream/points'
   print results["Health Level"]
   query = {
      'points-value': results["Health Level"],
      'points-at': t
   }

   response = requests.request('POST', base + endpoint, params=query, headers=header, timeout=120 )
```

```
    resp = json.loads( response.text )
    print resp
    if resp['points-code'] == 200:
        print( 'Update result-stream points: ok')
    else:
        print( 'Update result-stream points: error')
        print( response.text )



def loop(timeL,tLE,tT,rN,resultsLast):
    # Retrieve (GET) data from the server
    # Set url address.
    #base = 'http://127.0.0.1:5000'
    try:
        endpoint = '/networks/local/objects/pho-object/streams/pho-stream/points'
        query = {}
        response = requests.request('GET', base + endpoint, params=query, headers=header, timeout=120 )
        resp = json.loads( response.text )
        if resp['points-code'] == 200:
            print( 'Receive pho-stream points: ok')
            print resp['points'][0]['value']
            phoIN = resp['points'][0]['value']
            t= resp['points'][0]['at']
            #if t == timeL:
            #    print ("no new data found")
            #    time.sleep(1)
            #    print "here5.11"
            #    return resultsLast
        else:
            print( 'Receive pho-stream points: error')
            print( response.text )


        #store temperature points
        endpoint = '/networks/local/objects/temp-object/streams/temp-stream/points'
        query = {}
        response = requests.request('GET', base + endpoint, params=query, headers=header, timeout=120 )
        resp = json.loads( response.text )
        if resp['points-code'] == 200:
            print( 'Recieve temp-stream points: ok')
            tempIN = resp['points'][0]['value']
        else:
            print( 'Recieve temp-stream points: error')
            print( response.text )

    except:
        print ("retreival failed")
    time.sleep(1)
    ##### process data ######
    try:
        #process the results. total exposure = sum of light levels retrieved
        readingNo = rN + 1
```

```
        totalLightExposure = tLE + phoIN
        avgLightExposure = totalLightExposure/readingNo
        totalTemp = tT + tempIN
        avgTemp = totalTemp/readingNo
        if phoIN < 45:
            if tempIN <= 25:
                healthLevel = 1
                rec = "Healthy conditions, no recommendation"
            elif tempIN > 25 & tempIN <= 27:
                healthLevel = 2
                rec = "Temperature is high, consider turning down the thermostat"
            else:
                healthLevel = 3
                rec = "Heat stroke imminent, activating fan"
        else:
            if tempIN <= 19:
                healthLevel = 1
                rec = "It's a bit bright, healthy but electric lights are unneeded"
            elif tempIN > 20 & tempIN <= 21:
                healthLevel = 2
                rec = "Temperature and light high, consider closing the blinds to avoid solar gain"
            else:
                healthLevel = 3
                rec = "Heat stroke imminent, activating automatic blinds and fan"

        print ("Health Level = "), healthLevel
        print ("Reading # = "), readingNo
        print ("Total Light Exposure = "), totalLightExposure
        print ("Average Light Exposure = "), avgLightExposure
        print rec

        results = {"t":t,"Rec":rec,"Health Level":healthLevel,
                   "Total Light Exposure":totalLightExposure,
                   "Total Temp": totalTemp,
                   "Average Light Exposure":avgLightExposure,
                   "Average Temp":avgTemp,
                   "Reading Number":readingNo
        }

        #collect data every 5 seconds
        time.sleep(5)
        return results
    except:
        print("processing failed")
    time.sleep(1)
###### POST and structure functions ##########

# Run once at the start
def setup():
    try:
        print "Setup"
    except:
        print "Setup Error"
```

```
# Run continuously forever
# with a delay between calls
def delayed_loop():
   print "Delayed Loop"

#Run once at the end
def close():
   try:
      print "Close ListenAndProcess"
      #ser.close()
   except:
      print "Close Error"



###### MAIN function ##########

def main():
   # Call setup function
   setup()
   # Set start time
   nextLoop = time.time()
   readingNo = 0
   totalLightExposure = 0
   totalTemp = 0
   tLast=0
   resultsLast = {"t":0,"Rec":0,"Health Level":0,
            "Total Light Exposure":0,
            "Total Temp": 0,
            "Average Light Exposure":0,
            "Average Temp":0,
            "Reading Number":0
   }
   t=0
   time.sleep(1)

   while(True):
      # Try loop() and delayed_loop()
      try:
         results = loop(t,totalLightExposure,totalTemp,readingNo,resultsLast)
         totalLightExposure = results["Total Light Exposure"]
         totalTemp = results["Total Temp"]
         readingNo = results["Reading Number"]
         t = results["t"]
         if t != tLast:
            print ("Sending Results: ")
            print results
            storeresults (results, t)
            tLast = t
```

```
            print tLast
         else:
            print("no results to store")
         if time.time() > nextLoop:
            # If next loop time has passed...
            nextLoop = time.time() + delay
            delayed_loop()
      except KeyboardInterrupt:
         # If user enters "Ctrl + C", break while loop
         break
      except:
         # Catch all errors
         print "Unexpected error."
   # Call close function
#    close()

# Run the program
#####NOTE: dont forget to restart the consoles if the server crashes.
#otherwise I'll get errors
main()
```

extend_dashboard_linksCPS.html

```
<!--extend dashboard links for CPS, Assembled and Modified by John Stuart-->
   <li>
      <a href="#" id="sidebar-input" class="first-level"><i class="fa
      fa-tags fa-fw"></i> Input</a>
   </li> <!--input link-->
   <li>
      <a href="#" id="sidebar-report" class="first-level"><i class="fa
      fa-area-chart fa-fw"></i> Report</a>
   </li> <!--report link-->
   <li>
      <a href="#" id="sidebar-light" class="first-level"><i class="fa
      fa-sun-o fa-fw"></i> Light Levels</a>
   </li> <!--pho link-->
   <li>
      <a href="#" id="sidebar-temperature" class="first-level"><i class="fa
      fa-fire fa-fw"></i> Temperature</a>
   </li> <!--temp link-->
```

extend_dashboard_pagesCPS.html

```
<!--extend dashboard pages for CPS, Assembled and Modified by John Stuart-->
 <!--Defines layout of input page-->
      <div class="page hidden" id="page-input">
```

```
        <div id="page-wrapper">
          <div class="row">
            <div class="col-lg-12">
              <h1 class="page-header">Input</h1>
            </div>
            <!-- /.col-lg-12 -->
          </div>
          <!-- /.row -->
          <div class="row">
            <div id="content-input" class="col-lg-8">

 <!--inserts first name and last name form into page-->
              <form id="form-input">
                <fieldset class="form-group">
                  <label for="first-name">First Name</label>
                  <input type="text" class="form-control"
name="first-name" placeholder="" required="true">
                </fieldset>
                <fieldset class="form-group">
                  <label for="last-name">Last Name</label>
                  <input type="text" class="form-control"
name="last-name" placeholder="" required="true">
                </fieldset>
                <button type="submit" class="btn">Submit</button>
              </form>


            </div>
            <!-- /.col-lg-8 -->
            <div class="col-lg-2 note"></div>
            <!-- /.col-lg-2 -->
          </div>
          <!-- /.row -->

        </div>
        <!-- /#page-wrapper -->
      </div>
      <!-- /#page-input -->




<!--Defines layout of report page-->
      <div class="page hidden" id="page-report">
        <div id="page-wrapper">
          <div class="row">
            <div class="col-lg-12">
              <h1 class="page-header">Report</h1>
            </div>
            <!-- /.col-lg-12 -->
          </div>
          <!-- /.row -->
```

```
            <div class="row">
                <div id="content-report" class="col-lg-8">


                </div>
                <!-- /.col-lg-8 -->
                <div class="col-lg-2 note"></div>
                <!-- /.col-lg-2 -->
            </div>
            <!-- /.row -->

        </div>
        <!-- /#page-wrapper -->
    </div>
    <!-- /#page-input -->


    <!--Defines layout of light page-->
    <div class="page hidden" id="page-light">
        <div id="page-wrapper">
            <div class="row">
                <div class="col-lg-12">
                    <h1 class="page-header">Light Levels</h1>
                </div>
                <!-- /.col-lg-12 -->
            </div>
            <!-- /.row -->
            <div class="row">
                <div id="content-light" class="col-lg-8">


                </div>
                <!-- /.col-lg-8 -->
                <div class="col-lg-2 note"></div>
                <!-- /.col-lg-2 -->
            </div>
            <!-- /.row -->

        </div>
        <!-- /#page-wrapper -->
    </div>
    <!-- /#page-input -->




            <!--Defines layout of temp page-->
    <div class="page hidden" id="page-temperature">
        <div id="page-wrapper">
            <div class="row">
                <div class="col-lg-12">
                    <h1 class="page-header">Temperature</h1>
                </div>
```

```
            <!-- /.col-lg-12 -->
          </div>
          <!-- /.row -->
          <div class="row">
            <div id="content-temp" class="col-lg-8">


            </div>
            <!-- /.col-lg-8 -->
            <div class="col-lg-2 note"></div>
            <!-- /.col-lg-2 -->
          </div>
          <!-- /.row -->

        </div>
        <!-- /#page-wrapper -->
      </div>
      <!-- /#page-input -->
```

extend_dashboard.js

```
//Extend_Dashboard, Assembled and Modified by John Stuart
//loads etent_dashboard_links
var ul = $('ul#side-menu');
$.ajax({
  url : '/static/extend_dashboard_linksCPS.html',
  type: "get",
  success : function(response){
    console.log("Load /static/extend_dashboard_linksCPS.html");
    ul.append(response);
  }
});

//loads the pages defined in extend_dashboard_pages
var wrapper = $('div#wrapper');
$.ajax({
  url : '/static/extend_dashboard_pagesCPS.html',
  type: "get",
  success : function(response){
    console.log("Load /static/extend_dashboard_pagesCPS.html");
    wrapper.append(response);

    // Form submit call goes here.
    $("form#form-input").submit( onInputFormSubmit );
  }
});

///*
//  Add function to get points for report page
```

```
//*/

//takes points from wallflower_demo for use in plotting later
function getPoints( the_network_id, the_object_id, the_stream_id, callback ){
  var query_data = {};
  var query_string = '?'+$.param(query_data);
  var url = '/networks/'+the_network_id+'/objects/'+the_object_id;
  url += '/streams/'+the_stream_id+'/points'+query_string;

  // Send the request to the server
  $.ajax({
    url : url,
    type: "get",
    success : function(response){
      console.log( response );

      if( response['points-code'] == 200 ){
        var num_points = response.points.length
        var most_recent_value = response.points[0].value
        console.log("Most recent value: "+most_recent_value);
        console.log("Number of points retrieved: "+num_points);
        callback( response.points );
      }
    },
    error : function(jqXHR, textStatus, errorThrown){
      console.log(jqXHR);
    }
  });
}


// Call getPoints if Input or Report is selected
// ...added feature to dynamically update plot as new data becomes available
custom_sidebar_link_callback = function( select ){

  if (select == 'input') {

  }
  else if (select == 'report'){
    var plotCalls = 0;
    var plotTimer = setInterval( function(){
      getPoints('local','test-object','test-stream', function(points){
        console.log( "The points request was successful!" );
        loadPlot( points );
      });
      if( plotCalls > 20 ){
        console.log( 'Clear timer' );
        clearInterval( plotTimer );
      }else{
        plotCalls += 1;
      }
    }, 1000);
  }
```

```
  else if (select == 'light'){
    var plotCalls = 0;
    var plotTimer = setInterval( function(){
      getPoints('local','pho-object','pho-stream', function(points){
        console.log( "The points request was successful!" );
        loadPlotPho( points );
      });
      if( plotCalls > 20 ){
        console.log( 'Clear timer' );
        clearInterval( plotTimer );
      }else{
        plotCalls += 1;
      }
    }, 1000);
  }
  else if (select == 'temperature'){
    var plotCalls = 0;
    var plotTimer = setInterval( function(){
      getPoints('local','temp-object','temp-stream', function(points){
        console.log( "The points request was successful!" );
        loadPlotTemp( points );
      });
      if( plotCalls > 20 ){
        console.log( 'Clear timer' );
        clearInterval( plotTimer );
      }else{
        plotCalls += 1;
      }
    }, 1000);
  }
}


/*
    Function to plot data points using Highcharts
*/
function loadPlotPho( points ){
  var plot = $('#content-light');
  // Check if plot has a Highcharts element
  if( plot.highcharts() === undefined ){
    // Create a Highcharts element
    plot.highcharts( report_plot_options_pho );
  }

  // Iterate over points to place in Highcharts format
  var datapoints = [];
  for ( var i = 0; i < points.length; i++){
    var at_date = new Date(points[i].at);
    var at = at_date.getTime() - at_date.getTimezoneOffset()*60*1000;
    datapoints.unshift( [ at, points[i].value] );
  }

  // Update Highcharts plot
```

```
  if( plot.highcharts().series.length > 0 ){
    plot.highcharts().series[0].setData( datapoints );
  }else{
    plot.highcharts().addSeries({
      name: "Photosensor Data",
      data: datapoints
    });
  }
}

var report_plot_options_pho = {
    chart: {
        type: 'spline'
    },
    title: {
        text: 'Light Level Data'
    },
    subtitle: {
        text: 'Created by John Stuart'
    },
     yAxis: {
    title: {
     text: 'Ambient light (lux)'
    },
     },
    xAxis: {
      type: 'datetime',
       title: {
        text: 'Time'
       },
      dateTimeLabelFormats: { // don't display the dummy year
          month: '%e. %b',
          year: '%b'
      },
    },
};



function loadPlotTemp( points ){
  var plot = $('#content-temp');
  // Check if plot has a Highcharts element
  if( plot.highcharts() === undefined ){
    // Create a Highcharts element
    plot.highcharts( report_plot_options_temp );
  }

  // Iterate over points to place in Highcharts format
  var datapoints = [];
  for ( var i = 0; i < points.length; i++){
    var at_date = new Date(points[i].at);
    var at = at_date.getTime() - at_date.getTimezoneOffset()*60*1000;
    datapoints.unshift( [ at, points[i].value] );
```

```
  }

  // Update Highcharts plot
  if( plot.highcharts().series.length > 0 ){
    plot.highcharts().series[0].setData( datapoints );
  }else{
    plot.highcharts().addSeries({
      name: "Temperature Sensor Data",
      data: datapoints
    });
  }
}

var report_plot_options_temp = {
    chart: {
        type: 'spline'
    },
    title: {
        text: 'Temperature Data'
    },
    subtitle: {
        text: 'Created by John Stuart'
    },
      yAxis: {
    title: {
      text: 'Temperature (C)'
    },
      },
    xAxis: {
      type: 'datetime',
       title: {
        text: 'Time'
       },
      dateTimeLabelFormats: { // don't display the dummy year
          month: '%e. %b',
          year: '%b'
      },
    },
};


/*
  Add functionality to the input page form
*/
function onInputFormSubmit(e){
  e.preventDefault();

  var object_id = "obj-names";
  var stream_id = "stm-form-input";

  // Gather the data
  // and remove any undefined keys
  var data = {};
```

```
  $('input',this).each( function(i, v){
    var input = $(v);
    data[input.attr("name")] = input.val();
  });
  delete data["undefined"];

  console.log( data );

    var url = '/networks/'+network_id+'/objects/';
  url = url + object_id+'/streams/'+stream_id+'/points';
  var query = {
    "points-value": JSON.stringify( data )
  };

  // Send the request to the Pico server
  $.ajax({
    url : url+'?'+$.param(query),
    type: "post",
    success : function(response){
      var this_form = $("form#form-input");

      if( response['points-code'] == 200 ){
        console.log("Success");
        // Clear the form
        this_form.trigger("reset");
      }
// Log the response to the console
      console.log(response);
    },
    error : function(jqXHR, textStatus, errorThrown){
      // Do nothing
    }
  });

};
```