Go Green, Go Home

“Make America Green Again”

**CE 186: Design of Cyber-Physical Systems**

**Final Report**

**16 December 2016**

 *Kan Ito*

12974257_568162763363819_4854629158720156753_n.jpg *Sara Mitchell*

 *Liam Weaver*

*Brandon Wong*

**ABSTRACT**

Residential energy consumption in the United States is projected to increase by over 10% as population and access to advanced technologies continue to rise.[8] Ideally, this demand will be met by a higher percentage of renewable sources than that of the current energy mix. However, renewable energy generation is inconsistent with time of energy use, which can lead to renewable curtailment and use of expensive, high-emission peaker plants. In order to take advantage of the cleaner energy produced during the day, consumer’s load schedules can be optimized to minimize emissions through direct load control and interruptible control management of non-critical loads. This concept is demonstrated by developing an optimization problem that creates load schedules based on time constraints reported by the user and data about the emission intensity of electricity throughout the day. Preliminary results yield over 1800 kg CO2 equivalent saved per household per year and an over 20% reduction of peak power demand. Scaling this solution to communities could revolutionize how residential energy is consumed. By integrating consumer energy demand, clean energy generation, and optimized control within a cyber-physical network, communities can transition away from high-emissions fossil fuels towards an efficient and sustainable electricity grid.

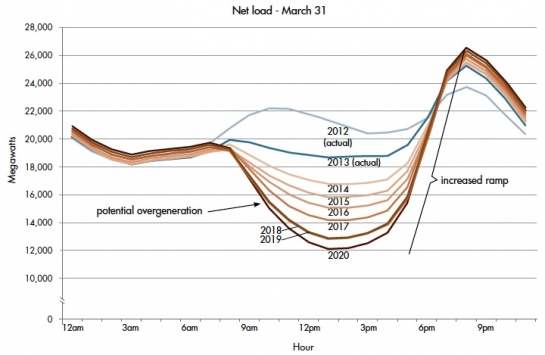
**I. INTRODUCTION**

**1.1 Motivation and Background**

Approximately 67% of the electricity in the U.S. is produced using fossil fuels, and electricity production accounts for 30% of fossil fuel use.[9] These numbers are decreasing as more renewable energy is being generated, but it is challenging to integrate these technologies at large scales while maintaining the reliability of the system.

Within the current system, electrical power is most often used at the same time it is produced because current storage technology can be inefficient and expensive. When fossil fuels are the energy source, this is manageable because the amount of electricity produced can be steady and controlled or ramped up for high demand. Solar and wind power cannot be controlled in this way. They are intermittent sources that can cause instability within the grid by producing a lot of electricity at inopportune times or very little electricity during periods of high demand. This instability creates inefficiencies within the system and additional expenses for grid maintenance.

In places where there is high solar energy penetration, there may be issues with over generation during the day and the need for a fast ramp up of other energy sources to satisfy demands in the evening when solar panels are not generating power. This problem is often illustrated with the “Duck Curve,” which displays daily power demand minus renewable generation, shown in **Figure 1**. This exponential peak power is met with marginal peaker plants, most often fossil fuel-based, which are capital intensive investments often times only online for 5% of the year (Source).



**Figure 1.** The Duck Curve.[[1]](#footnote-1)

Surges in power generation during the day can cause grid failures if the energy is not being used. To avoid these failures, renewable energy production may be curtailed or the utility may request that customers use more energy during these times.[10] Optimizing and automating the latter incentive to flatten the duck curve while maintaining full convenience for users would save substantial costs to the utility and ultimately the customers as well. It is clear that an effective method to address the problems associated with rising emissions and peak ramp demand is to align energy use with renewable energy generation.

**1.2 Relevant Literature**

A recent analysis of average hourly production data from the California Independent System Operator (CAISO) was conducted in order to “Revisit the California Duck Curve.”[12] The analysis looked at data from January 2011 to June 2016 and found that the issues associated with oversupply have developed faster than originally projected and that the maximum ramp is getting steeper each year.[12]

Load control management can benefit the integration of renewable energy into the grid. Shifting loads to consume abundant solar energy available during the day reduces the risks associated with overgeneration, decreases daily peak demands, and reduces emissions.[1] Direct load control and interruptible load control management are the most common load management programs.[2] Direct load control can be controlled by the utility or a third party to reshape the load curve. An example is the utility cycling the consumer’s large current drawing appliances such as refrigerators or air conditioners.[2] Interruptible load management involves consumers agreeing to reduce their energy demands during times of peak energy use, and it is often noticed.[2] Most controllable loads are small-scale and dispersed, and consumers will have more chances to schedule these controllable loads as smart homes and the smart grid develop.[2] In recent years, optimal load schedules have been developed and are mainly used for peak shaving, load shifting, minimizing production costs, or meeting reliability requirements.[2]

To assess environmental impact costs of residential demand side management, researchers developed an optimization algorithm to shift noncritical residential loads with the goal of reducing both energy cost and the emissions due to generation.[3] They found that it is possible to reshape total power demand to reduce cost and emissions, but reducing cost may conflict with reducing emissions at certain points.[3] In order to defer and shape loads to use cleaner energy, data about the carbon footprint of electricity coming from the grid is necessary.[4] WattTime, whose software has been implemented in Sutardja Dai Hall on UC Berkeley’s campus, can detect when there is clean energy available on the California grid and signal the HVAC unit to run at those times, optimizing energy use while staying within temperature setpoints to avoid occupant discomfort.[5] Another relevant project uses WattTime to minimize the CO2 emissions produced for charging electric vehicles.[6] EVs require a lot of electricity, but they are a highly time-shiftable load, making them an important focus of load management. After plugging the EV in and providing information about when the car is going to be used again, an appliance called “JuiceBox Green” schedules charging to minimize CO2 emissions.[6] For some parts of the country, this appliance can determine whether an EV is responsible for more or less emissions than a conventional vehicle.[6]

**1.3 Focus of this Study**

Go Green, Go Home collects energy demands from users and executes them based on when energy generation is cleanest and most readily available.

**II. TECHNICAL DESCRIPTION**

**2.1 Process/ Data Journey**

The cyber-physical infrastructure developed is schematically depicted in **Figure 2**. The program begins when the user provides information about when their energy demands need to be met (i.e. when their laundry needs to be done and their car needs to be charged) by entering it on a web interface. This information is stored in the Wallflower server. From there, the Python code retrieves the user information through “GET” requests and stores it as a variable. Predictions about the carbon intensity of electricity for each hour of the selected day are simultaneously drawn from the WattTime API, cleansed, and stored as the optimization cost function in Python.

A system of matrix infrastructure is developed within the Python code to formulate the optimization problem to minimize the carbon emissions by manipulating the decision variables: the power and energy for each load at each timestep. 24 power timesteps and 25 energy timesteps represent a period of 24 hours. The optimization and simulation begin at 6 PM to represent a user returning home from work and inputting the load constraints for the following day. The Mixed Integer Linear Program (MILP) is solved by routing the problem through the CloudLP optimization server and the resulting lists of schedules are stored as global variables in Python.

**2.X Actuation**

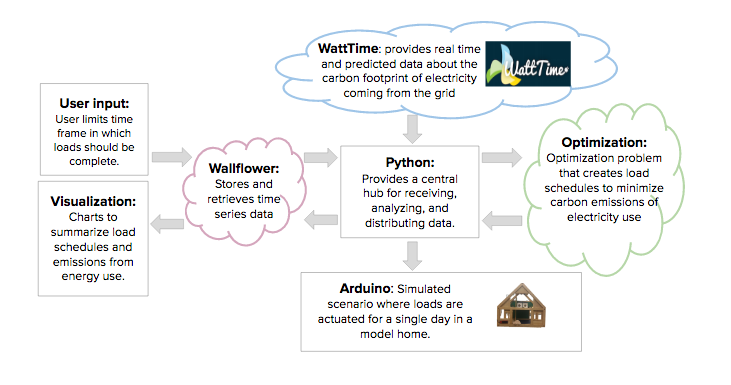
The loads of the household unit is actuated to simulate such dynamics of the optimization. The deferrable loads, the washer and dryer, are simulated using a low voltage DC motor. The electric vehicle charge, time stamp, and HVAC power is displayed on an LCD screen mounted on the wooden model house. The numbers are updated every simulated hour. The fixed loads, lights, are represented with an array of lights for the two floors in the wooden house. One washer motor is designed to be powered from the external battery through the passage of the low voltage relay. The dryer motor is directly powered from the single board computer (Arduino) via a transistor-resistance control to prevent back emf (electromagnetic force) to the Arduino pins. The LEDs are arranged in parallel to hold higher voltage for better illuminance. The LCD display controls were provided using the set controls set by the manufacturer, Sparkfun?

Finally, visualized & actuated.

Text

Text

text



**Figure 2. Schematic of the cyber physical system**

The Physical layer of the code: To effectively simulate the schedules optimized in the CPS, the following hardware configuration. Here looky at **Table 1** it’s cool.

**Table 1: Bill of materials: Kan: What else did we use?**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Material** | **Function** | **Source** | **Quantity** | **Cost** |
| Doll House | House actuators | Craigslist | 1 | $25 |
| Gearmotor/Transistor | Actuator to represent deferrable demands | CE 186-supplied | 2 | $0 |
| Battery Holder | Replicate Electric Vehicles/ Power Arduino | CE 186-supplied | 1-3 | $0 |
| Miscellaneous Electronics | Hardware needed to connect devices and actuate lighting | CE 186-supplied and/or owned by group members | Wires, LEDs, Arduinos | $0 |
| Relay switches | Relay of low voltage systems | Sparkfun | 2 | $10 |
| LCD Screen | Shows EV state of charge and HVAC power | CE 186 Supplied and/or owned by group members | 1 | $0 |
| External Batteries | Storage of external storage (6V) | Commercial Retail | 4 | $7 |

The cyber-layer of the CPS is represented in **Table 2**. Spread across Python, Servers,

Text

Text

Text

**Table 2: Summary of software codes**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Function** |
| CloudLPApp.py | Server | Executes optimization problem |
| extend\_dashboard\_links.html | HTML | Creates links for pages on the wallflower server |
| extend\_dashboard pages.html | HTML | Houses the HTML and CSS code for the “Schedule Loads”, “Emissions Summary” and “Scheduling Summary” pages |
| extend\_dashboard pages.js | JS | Adds functionality to the user input page |
| ListenAndDoEverything.py | Python | Creates streams and objects on the wallflower server in order to store data. Retrieves data from the WattTime and the Wallflower Server. |
| wallflower-atto-master.py | Server | Stores the user inputs, retrieves the optimized schedules and emissions data streams from Python and is used to display visualizations about the results. |
| script.en.js d3.v3.min.js  data2.txt | JS | Visualize the carbon intensity in a “sunburst” plot |
| GGGH\_actuate.ino | Arduino | Communicates with ListenAndDoEverything and transforms into controlled actuation signals. |

Ω

**2.2 A mathematical description of the data analysis**

The four different loads selected for this analysis were separated into independent functions to ease debugging and scalability of the program. The loads are outlined further below: Shapeable Load (Charging an Electric Vehicle), Shapeable Load (HVAC: regulating indoor air temperatures with an air conditioner and electric heater), Deferrable Load (Washer/Dryer), and Fixed Load (Lights). The final program contained 211 decision variables and 633 constraints.

***2.2.1 Shapeable Load (Electric Vehicle)***

The shapeable load, in this case, an electric vehicle, will be modeled by a linear programming problem. The problem statement is

The vector consists of the power consumed by the appliance at each hour and the total energy consumed (stored) by the appliance by each hour :

The costs, , represent the predicted carbon intensity (“cleanliness”) for each hour of the day as obtained from WattTime. They are padded by 24 zeros to create a vector.

The cost function only considers the power used at each time step. WattTime gives carbon intensity in units of g CO2/kW, meaning that has units of g CO2. Additionally, carbon intensity depends on the time of day that energy *is consumed from the grid*, not how much energy is stored in the battery at that time of day.

The constraints of this function are:

In constraint (1), the “ceiling” is the capacity of the electric vehicle’s battery (e.g. 24 kWh). The “floor” increases linearly with time, since we assume that there is some maximum power (e.g. 6.6 kW) that can be delivered to the battery by the charger, and some deadline by which the vehicle must be fully charged. and are constants and is determined by the user.

The floor will be negative early on, but at a critical point , the floor will rise above zero and constrain . If the vehicle is totally uncharged before , it will need to continuously charge at in order to meet the deadline. Notice that when , , effectively creating an equality constraint .

Constraint (2) tells us that the cumulative energy consumed/stored by the next time step is equal to the cumulative energy consumed/stored at the previous time step, plus the power at that time step times the size of the time step. This is similar to Newton’s method as a linear approximation to a differential equation.

Constraint (3) tells us that the power consumed at any time step must be between 0 and .

***2.2.2 Shapeable load (HVAC)***

Heating and cooling follows a similar idea and can use the same cost function and constraints. We proposed using random numbers to determine the magnitude of heating and cooling required. For example, if a random number from 0 to 10 is chosen, 0 could signify do nothing, 1 to 5 could signify heat (to varying degrees), and 6 to 10 could signify cool (to varying degrees). Depending on the random number, , the total energy delivered by the HVAC system to the house could change. would also be some maximum power of the HVAC system. would be determined by the user, and would likely be something like 5 PM (when the user gets home, the temperature must be acceptable).

~~Since a real house is not perfectly insulated, some leakage rate applies to the heated or cooled air in the house. This could be roughly modeled by modifying constraint (2) as:~~

~~(2)~~

~~Where is some constant leakage of energy. This penalizes turning on the HVAC too early, as the leakage would be applied to each hour.~~

Although a real house is not perfectly insulated, we assumed that losses were negligible. A future area of study could include incorporating a heat loss model to the system.

***2.2.3 Deferrable load (washer/dryer)***

The deferrable load is a washer and dryer cycle. We assume that the washer and dryer produce a known load profile (known power and duration), however, they must run consecutively, and once they start they cannot be interrupted. We can model this problem as an integer programming problem. The problem statement is

The vector consists of logical (true/false) values for the washer and dryer being activated at each time step, [*followed by the total energy consumed by the system at each time step*] (the last part is not necessary for the optimization, but may be useful if we are interested in the house’s total energy consumption).

The costs need to be weighted by the known power consumed by the washer and dryer, and . Sample values used were and . This way, the ones and zeros in are converted to values of power for each time step in kW. Again, the values are weighted with zeros since they do not contribute to the carbon intensity calculation:

Our constraints are as follows:

Constraints (1) and (2) require that and be integers. Constraints (3) and (4) require that and be between 0 and 1. Therefore, the only possible values for and are .

Constraint (5) broadly states that the dryer’s state in time step must be the same as the washer’s state in time step . If the washer is off at time , then the dryer must be off at time ; if the washer is on at time , the dryer must be on at time . In practice, this constraint needs to be coded somewhat exhaustively when converted to the matrix form, codifying

Constraints (6) and (7) refer to the desired time range ***T*** in which the deferrable load can occur. This is determined by the user. is a vector with ones in all the time steps that fall within the desired time range, and zeros in other locations. As an example, if the desired time range to begin the wash/dry cycle was from midnight to 10 AM, the vector would be

When we multiply by or , we set an equality constraint requiring that the product be 1. This means that within and , there must be a single “1” value somewhere within the desired time range. Although the zeros will negate any effects from values of and outside of the desired time range, there is no reason for these to be nonzero, as it would unnecessarily increase the cost function.

***2.2.4 Fixed Load (Lights)***

Lighting of the home was considered an inflexible behavior and thus a fixed load. The vector was constant in each simulation, and is shown below:

The hours of lighting were based on a when a typical family may have their lights on, yielding use from 6pm to 12am at night and 7am to 10am in the morning.

***2.2.5 General notes on optimization and linearity of the problem***

We proposed doing the optimizations separately for each appliance, i.e. performing separate linear programming (LP) and integer programming (IP) problems. Although it is possible to formulate the problem as one large mixed integer linear programming (MILP) problem, it is not necessary, unless there is some overarching constraint (e.g. total power that can possibly be consumed by all appliances in the house) that applies to multiple appliances.

Since the cost function is linear, it follows that, for appliances represented by and ,

Additionally, the operator is distributable, that is

In Python, we reformulate the constraints in terms of the following matrix equations:

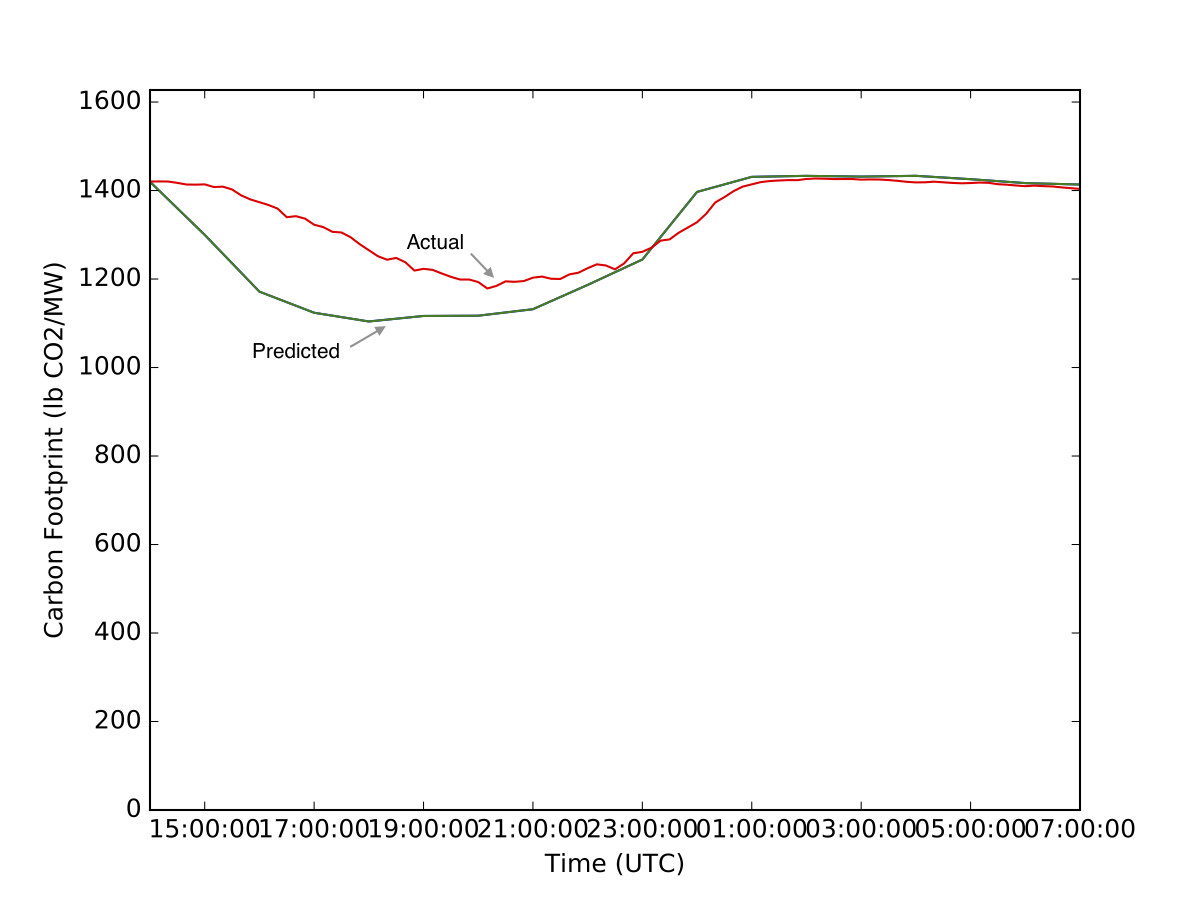
A constraint like can be rewritten as two inequality constraints, and .

It is also notable that and indeed this is what optimization programs do to create one large matrix in the form of .

**2.3 A presentation and analysis of the data collected & Visualization Tools**

The Python code requests predictions about the carbon intensity for the given day from the WattTime API (Figure X). This is stored into a variable to be used as a cost function.

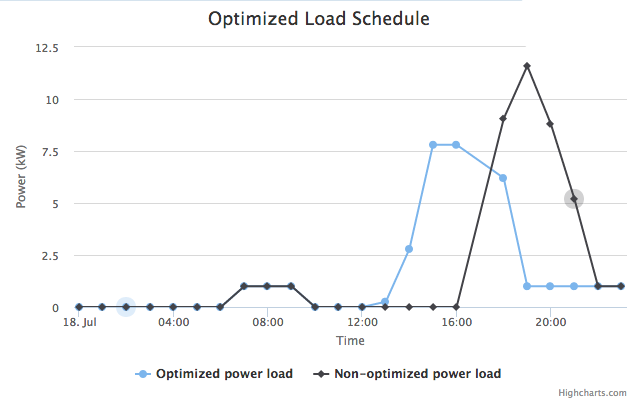
**Figure X** illustrates a sample from the WattTime API for a typical day.



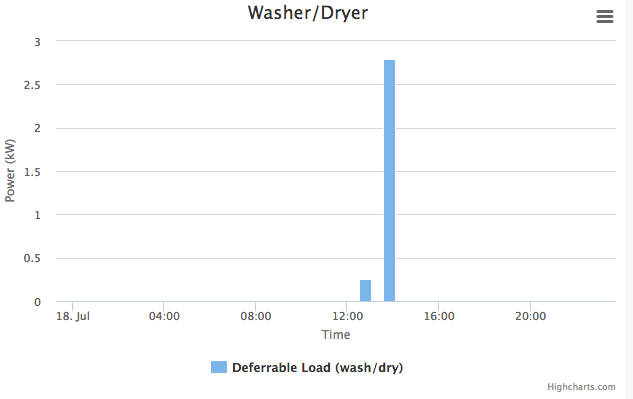
**Figure X. Watt Time**

* Watt Time > Data
* Streams from Optimization
* Load profiles
* Include snapshot of python code?
* Optimized schedules and emissions-- example?
* Percent difference in emissions for multiple days
  + table?
* Reduction in peaks for multiple days
  + table?
* Think about scaling the number up--how much pollution is prevented over a year-- how much is prevented if multiple households adopt the system

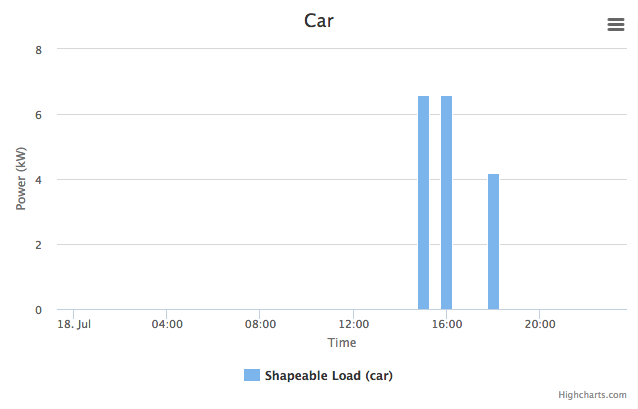
In order to easily visualize how the optimization changes the load pattern and reduces peak demands, a graph depicts the optimized power load for the system compared to a non-optimized power load (Figure X). We can see that the peak power demand is reduced by roughly 20% by shifting and spreading the peak load. Charts displaying the hourly power use of the individual loads are also displayed to easily view schedules and determine when specific loads will be satisfied (Figures X and X). Finally, the carbon intensity of the house, and the carbon footprint of the individual loads can be seen in a “sunburst” plot (Figure X). The charts shown below access data as streams in the Wallflower.Atto structure.



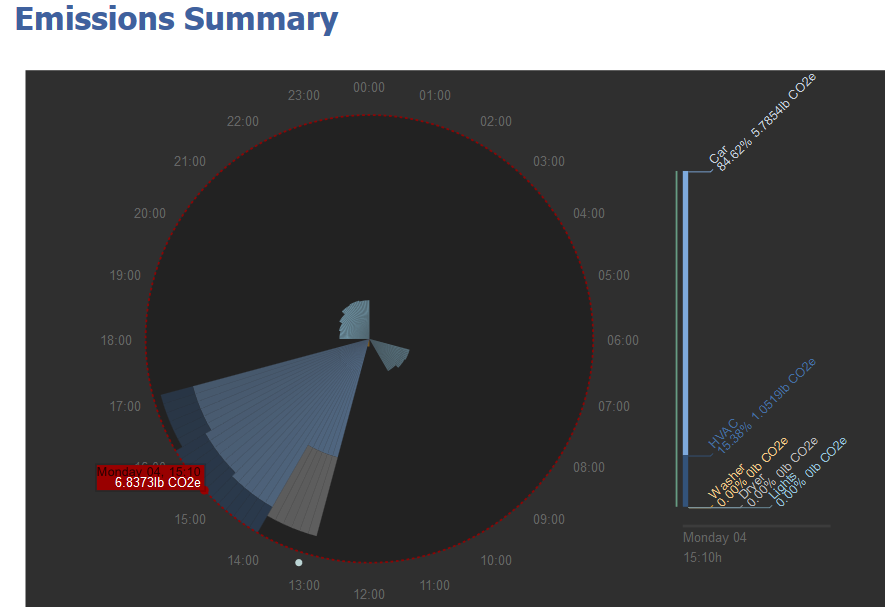
**Figure X.** Example of optimized and non-optimized power loads. Note the reduction and shift in peak demand in the optimized power load (blue).



**Figure X.** Example of load schedule for the washer and dryer



**Figure X.** Example of load schedule for electric vehicle charging



**Figure X.** The carbon intensity of each load is visualized on a “sunburst” plot

**III. DISCUSSION**

**What infrastructure system problem does it solve? How does it solve it? What innovations does it provide?**

Replacing fossil fuels with renewable energy is necessary to create a sustainable energy system. This project provides a solution to the challenges associated with renewable energy integration and the “duck curve.” By scheduling residential loads to minimize emissions from electricity use, it shifts power use to moments when renewable energy generation is higher, reducing the risk of overgeneration and lowering peak demands. This system is innovative in that it provides consumers an easy way to create load schedules that make their electricity use “greener” without disrupting their normal routines or requiring them to buy new appliances. The system does not rely on the utility or economic incentives for load management.

Although electricity prices are not considered in the optimization, users may save money with the system. For those with rooftop solar systems, their energy use can be shifted to times when their panels are generating electricity. This can reduce the amount of electricity they need to purchase from the grid when their panels are not generating any electricity. Also, the system may help customers with time-of-use (TOU) rate plans. With time-of-use pricing, customers pay more for electricity during peak hours, and the optimization is meant to shift loads away from hours of peak demand.

**IV. SUMMARY**

The aim of this project is to maximize the efficiency of energy systems and help to integrate renewable technologies into the grid. The environmental impact of electricity use varies throughout the day depending on the most optimal times for capturing renewable resources such as solar or wind. The current electrical grid makes no prioritization toward these resources and seeks simply to satisfy instant demand from consumers. Certain electricity demands within communities are initiated by the consumer but don’t necessarily need to be executed instantly, such as a washing machine or an electric vehicle plug in. This project prototypes a typical household and a demand-response system whereby users provide inputs about their personal schedules and then the timing of their electricity use is optimized to minimize emissions. The demands are then ‘actuated’ based on the greenest and most economical time to use the energy. The proposed system could be scalable for anything from a small house to an integrated ‘smart city’ of the future. Preliminary results yield over 1800 kg CO2 equivalent saved per household per year and an over 20% reduction of peak power demand.

|  |
| --- |
| GGGH\_actuate.ino |
| #include <SoftwareSerial.h>  #include <CmdMessenger.h>  int washcount = 0;  byte leds = 0;  int washPin = 8;  int dryPin = 6;  int EVPin = 2;  int HVACPin = 10;  int lightPin1 = 7;  int lightPin2 = 5;  int lightPin3 = 4;  int lightPin4 = 3;  SoftwareSerial mySerial(3,EVPin); // pin 2 = TX, pin 3 = RX (unused)  void setup() {  // initialize serial communication at 9600 bits per second:  Serial.begin(9600);  mySerial.begin(9600);  pinMode(HVACPin,OUTPUT); // HVAC  pinMode(13,OUTPUT);  pinMode(washPin,OUTPUT); // Washer  pinMode(dryPin,OUTPUT); //Dryer  pinMode(EVPin,OUTPUT); // EV  pinMode(lightPin1, OUTPUT);  pinMode(lightPin2, OUTPUT);  pinMode(lightPin3, OUTPUT);  pinMode(lightPin4, OUTPUT);  }  void loop() {  //Listen to Python and actuate brightness of LED  if( Serial.available() > 0){    float hvac = 0;  //Serial.readBytesUntil(character, buffer, length)  float EV = Serial.parseFloat(); // EV charge => LCD Display  int wash= Serial.parseInt(); // Washer/dryer => DC Motor  int dry = Serial.parseInt(); // Dryer  hvac = Serial.parseFloat(); // HVAC => LCD?  float lights = Serial.parseFloat(); // Lights=> LEDs  int time = Serial.parseInt(); // Time=>LCD  //float a = cmdMessenger.readFloatArg();  String EVstring = String(EV);  if (EV > 0.0) {  digitalWrite(EVPin,HIGH);  }  if (EV == 0.0) {  digitalWrite(EVPin,LOW);  }  if (wash == 1) {  digitalWrite(washPin,HIGH);  }  if (wash == 0) {  digitalWrite(washPin,LOW);  }  if (dry == 1) {  digitalWrite(dryPin,HIGH);  }  if (dry == 0) {  digitalWrite(dryPin,LOW);  }  if (lights == 1.0) {  digitalWrite(lightPin1,HIGH);  digitalWrite(lightPin2,HIGH);  digitalWrite(lightPin3,HIGH);  digitalWrite(lightPin4,HIGH);  }  if (lights == 0.0) {  digitalWrite(lightPin1,LOW);  digitalWrite(lightPin2,LOW);  digitalWrite(lightPin3,LOW);  digitalWrite(lightPin4,LOW);  }  if (hvac > 0.0) {  digitalWrite(HVACPin,HIGH);  }  if (hvac == 0.0) {  digitalWrite(HVACPin,LOW);  }  int soc = 41;  clearscreen();  mySerial.write("EV: ");  mySerial.print(EVstring);  mySerial.write(" KWH");  mySerial.write(" Time: ");  mySerial.print(time);  mySerial.write(":00");  delay(2000);  clearscreen();  mySerial.write("HVAC: ");  mySerial.print(hvac);  mySerial.write(" KW ");  delay(2000);  clearscreen();  delay(100);    }  //delay(4000); // there is a half second delay in the communication between python and arduino  }  void clearscreen()  {  mySerial.write(254); // reset cursor  mySerial.write(128);  mySerial.write(" "); // print blank  mySerial.write(" ");  mySerial.write(254); // reset cursor  mySerial.write(128);  } |

**V. REFERENCES**

**We need to make sure these are in order at the end**

1. Denholm, P., O’Connell, M., Brinkman, G., Jorgenson, J., “Overgeneration form Solar Energy in California: A Field Guide to the Duck Chart.” National Renewable Energy Laboratory (2015), <http://www.nrel.gov/docs/fy16osti/65023.pdf>
2. Shen, J., Jiang, C., Li, B., “Controllable Load Management Approaches in Smart Grids.” *Energies* 2015, 8, 11187-11202; doi: 10.3390/en81011187
3. Tsagarakis, G., Thomson, C., Collin, A., Harrison, G., Kiprakis, A., McLaughlin, S., “Assessment of the Cost and Environmental Impact of Residential Demand-Side Management.” *IEEE Transactions on Industry Applications* 2016, 52(3)
4. WattTime. Watttime.org, Accessed 10/13/16 8:00 PM
5. WattTime. “HVAC RoofTop Application.” Retrieved from <http://watttime.org/wp-content/uploads/2015/09/Commercial-HVAC.pdf> on 13 October 2016.
6. eMotorWerks. “Emissions Reduction of Electric Vehicle Charging.” Retrieved from <http://watttime.org/wp-content/uploads/2015/09/Solution-Brief_EVSE.pdf> on 13 October 2016.
7. Pacific Northwest National Laboratory. “What is the VOLTTRONTM Platform?” Retrieved from <http://transactionalnetwork.pnnl.gov/volttron.stm> on 13 October 2016.
8. U.S. Energy Information Administration. “Total energy supply and disposition demand.” Retrieved from <http://www.eia.gov/forecasts/aeo/data/browser/#/?id=4-AEO2016&cases=ref2016~ref_no_cpp&sourcekey=0> on 14 October 2016.
9. Pollock, Z, ”Top Ten Utility Smart Grid Deployments in North America”, gtm:, 05.17.2012. Retrieved from <http://www.greentechmedia.com/articles/read/top-ten-utility-deployments-in-north-america> on 15 September 2016.
10. International Renewable Energy Agency, “Smart Grids and Renewables.” Retrieved from <http://www.irena.org/documentdownloads/publications/smart_grids.pdf> on 23 September 2016.
11. Le Floch, C., Bansal, S., Tomlin, C.J., Moura, S., Zeilinger, M.N. (2016). Plug-and-Play Model Predictive Control for Load Shaping and Voltage Control in Smart Grids. Retrieved from <https://arxiv.org/pdf/1606.06682v1.pdf>
12. <http://www.scottmadden.com/wp-content/uploads/2016/10/Revisiting-the-Duck-Curve_Article.pdf>

1. California Independent Systems Operator (2016). *Fast Facts*. Retrieved from <https://www.caiso.com/Documents/FlexibleResourcesHelpRenewables_FastFacts.pdf> [↑](#footnote-ref-1)