AMS Sustainable Food Truck: Technology Assessment & Energy Management

Guilherme Ono Sens

University of British Columbia

VOL 400

August 01, 2014

AMS Sustainable Food Truck: Technology Assessment & Energy Management

By: Guilherme Ono Sens
Supervisor: Dr. Paul Lusina
Date: August 2014

Abstract

The AMS at UBC has a goal to drastically reduce the Green House Gas emission in the next decade. For such manner, a sustainable food truck fits their purpose and the culture of the city of Vancouver. This project involves a simulation system as well as an energy management algorithm to predict the behaviour of the electrical system of the truck and to best suit its needs, considering all the equipment is powered by solar panels (as a primary source), fuel cells (secondary source) and a battery (back-up source). Also, the project includes the implementation of each subsystem and their integration to the point where the relevant data can be displayed on the screen. The user will be able to customize the simulations, choosing which subsystem will be turned on or off.

Scenarios for the simulations were tested, such as a busy sunny day (during summer), providing enough data to state that the mobility system of the truck may not be supported by the sources. Moreover, unless the technologies for equipment and energy generation are well chosen, energy shortage might be expected in bad-case scenarios, such as a cloudy day.

The results of these simulations will be used to create and design specifications of the truck's energy systems in future stages of the AMS Sustainable Food Truck project.

Contents

1.INTRODUCTION

The Alma Mater Society (AMS) at UBC has been engaged in the environmental matters since 2007, when its Environmental Policy was created [1]. Thus, in 2008, the AMS Lighter Footprint Strategy was presented in order to "provide a framework for fostering environmental justice in our own operations and lobbying for sustainability practices through our relationships with the University community and broader society" [2]. Within this strategy, the goal of reducing the Green House Gas (GHG) emission by 33 percent until 2020 was set.

Considering the dependency on fossil based fuel technologies (which not only disrupt the planet's carbon cycle, but also are finite, meaning the resources may one day no longer be available for human daily use), it is necessary to switch to newer and greener technologies.

In a city like Vancouver, where food-trucks are commonly seen in the streets, the project of a sustainable food-truck that considers zero GHG emission is one step to achieve the goals established by AMS. This project considered powering the truck's equipment (such as microwave, fridge, deep fryer, griddle, ventilation and a variety of devices) with green sources of energy, including solar panels, fuel cells and a battery.

Starting with a custom energy evaluation tool, designed using MATLAB and Simulink, to simulate operation scenarios for the truck, this project is capable of estimating high, medium and low energy usage, which may include sunny and cloudy day operations (to simulate the high and low limits of the solar energy generated), busy and quiet days, in order to define how the loads (equipment) and energy generation/storage system will behave in these situation. Moreover an energy management strategy was included in the simulations with the purpose to find the optimal operation of the system.

A energy management algorithm was implemented prioritizing the energy usage of each source in a way that the solar energy, fuel cell and battery work together to best suit the loads' energy requirements.

Furthermore, this project does not include financial evaluation of the system

and technologies used nor the truck's mobility, meaning that the motor was not considered as a load to the system. However, this possibility can be added to the algorithm in a further stage of the project.

2. METHODS

The system described in this project was developed using MATLAB and Simulink to create subsystems capable of simulating the behavior of the sources and loads energy usage as well as the power profiles of such.

2.1. The Sources

2.1.1. Solar Panels

The solar panel subsystem was made simple on a MATLAB function block that receives inputs from a Radiation.mat file and from the user (number of panels, area of each panel, efficiency), multiplying these inputs to generate a Solar_Power signal.

Figure 1 shows the subsystem that represents the solar panel. Appendix 1demonstrates the code for the function block multiplying the inputs.

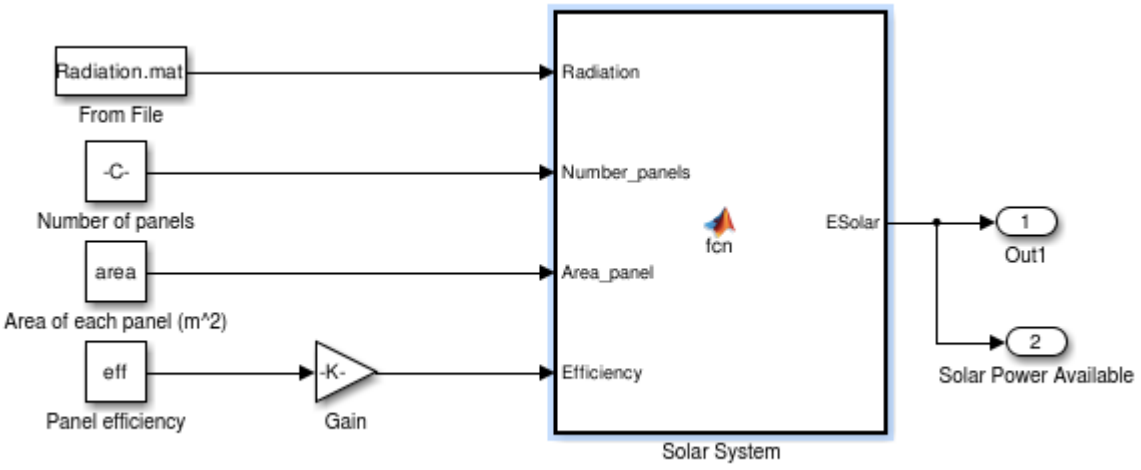

Figure 1 - Solar Panel Subsystem

The MATLAB function block was chosen over a simple multiplying block to implement this subsystem due to its capacity to be customized, therefore, modified if needed in the future.

2.1.2. Fuel Cell

The Fuel cell subsystem needed more attention than the previous source, once the powered required had to be updated constantly and a limit to the maximum power output had to be added.

Figure 3 shows the subsystem, which receives an input from the management system that includes the amount of power required by the loads that the primary source (solar cells) was not able to provide. This input ranges between 0 (zero) and the maximum output rate provided by the user. Because the fuel in the cell is finite, another limiter had to be implemented, this time in the form of a switch that turns the cell off when its energy consumption reaches the maximum energy provided for the period of simulation. The discrete integration block integrates the instant power generating an ascendant energy usage curve that is limited to the maximum fuel energy times the efficiency of the cell.

In Figure 3, outputs 2 and 3 will not be used by the management system. They will be used by a plotting system instead, in order to better analyze the data generated and to follow the behaviour of this source.

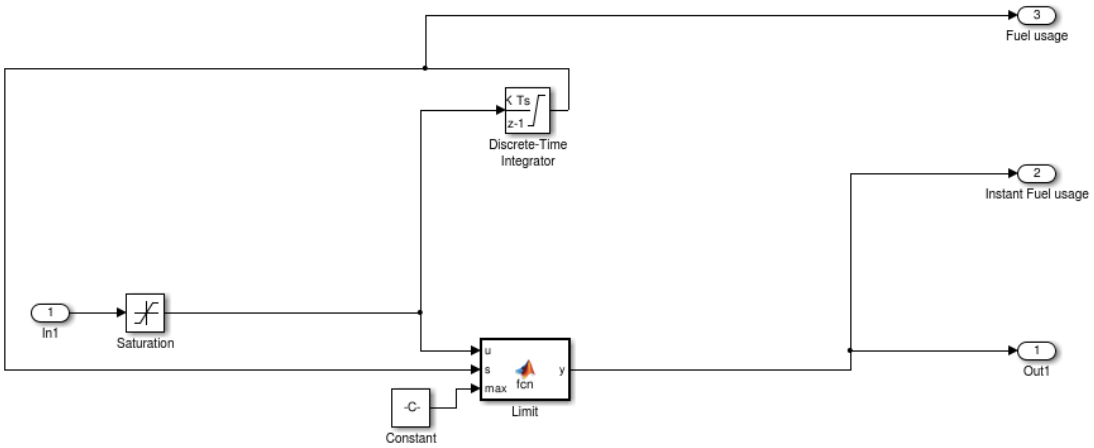

Figure 2 - Fuel Cell Subsystem

2.1.3. Battery

The battery subsystem was the most complex one of the project to be implemented considering its needs and parameters. The subsystem needed to be updated on the power required by the loads, and should be able to discharge and charge, also updating its own available energy.

Figure 4 shows the battery model. The small system on the top is the representation of the discharging system, in which the power required is the input and it is limited by the available energy, once it reaches 0 (zero), the signal is stopped, assuring that the battery does not provides more energy than its maximum capacity.

The system below represents the charging and storage systems integrated. Input 2 provides the energy going in, being limited by the energy available, one it reaches its maximum capacity. Integrating the signal provides the energy curve which will be sent to a sum block (adding the energy_in curve and the initial energy in the battery and subtracting the energy_out curve, thus generating a curve that shows the energy available).

All the outputs other than 1 will be used by the plotting system to generate the battery usage profile.

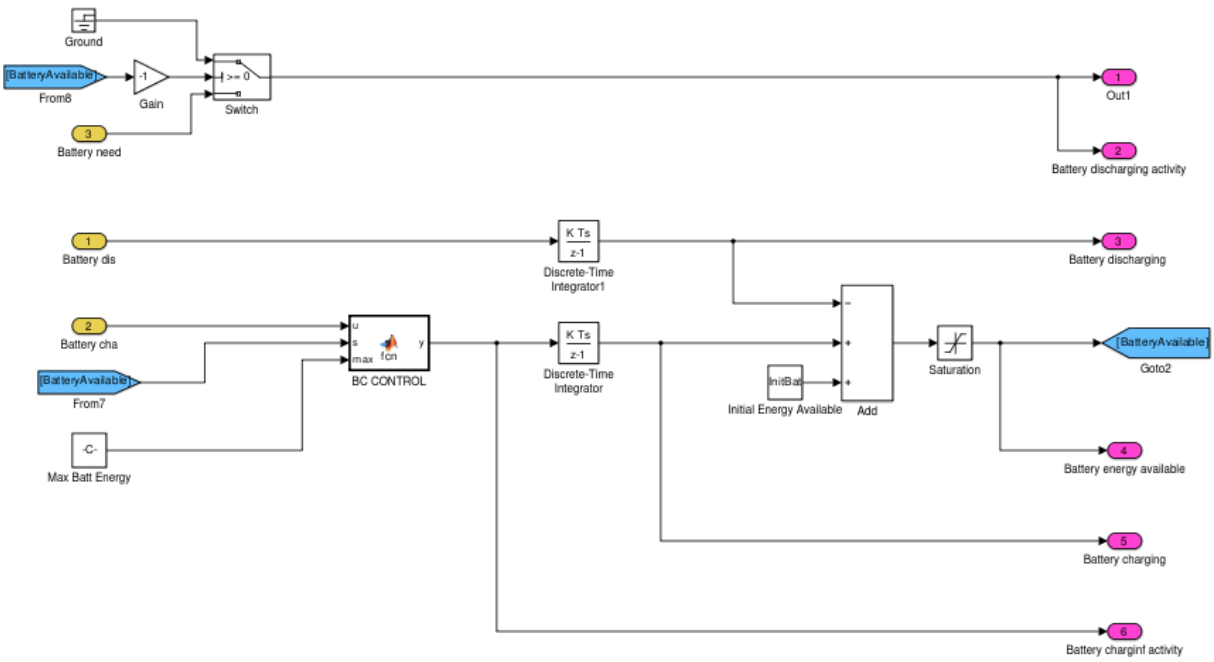

Figure 3 - Battery subsystem

2.2. Loads

The Loads subsystem receives Load Profiles files as inputs. These profiles were created off-line (on MATLAB using the code in Appendix 4) based estimates of usage patterns and each load's power.

This subsystem basically sums all the loads profiles and generates in single signal to be analyzed by the management algorithm. It also added a switch to each load as seen in Figure 5, which can be operated by the user (who is now capable of choosing the loads that will be operating in the simulation). Figure 6 demonstrates the interface that allows the user to turn on or off the loads by typing 1 or 0 respectively.

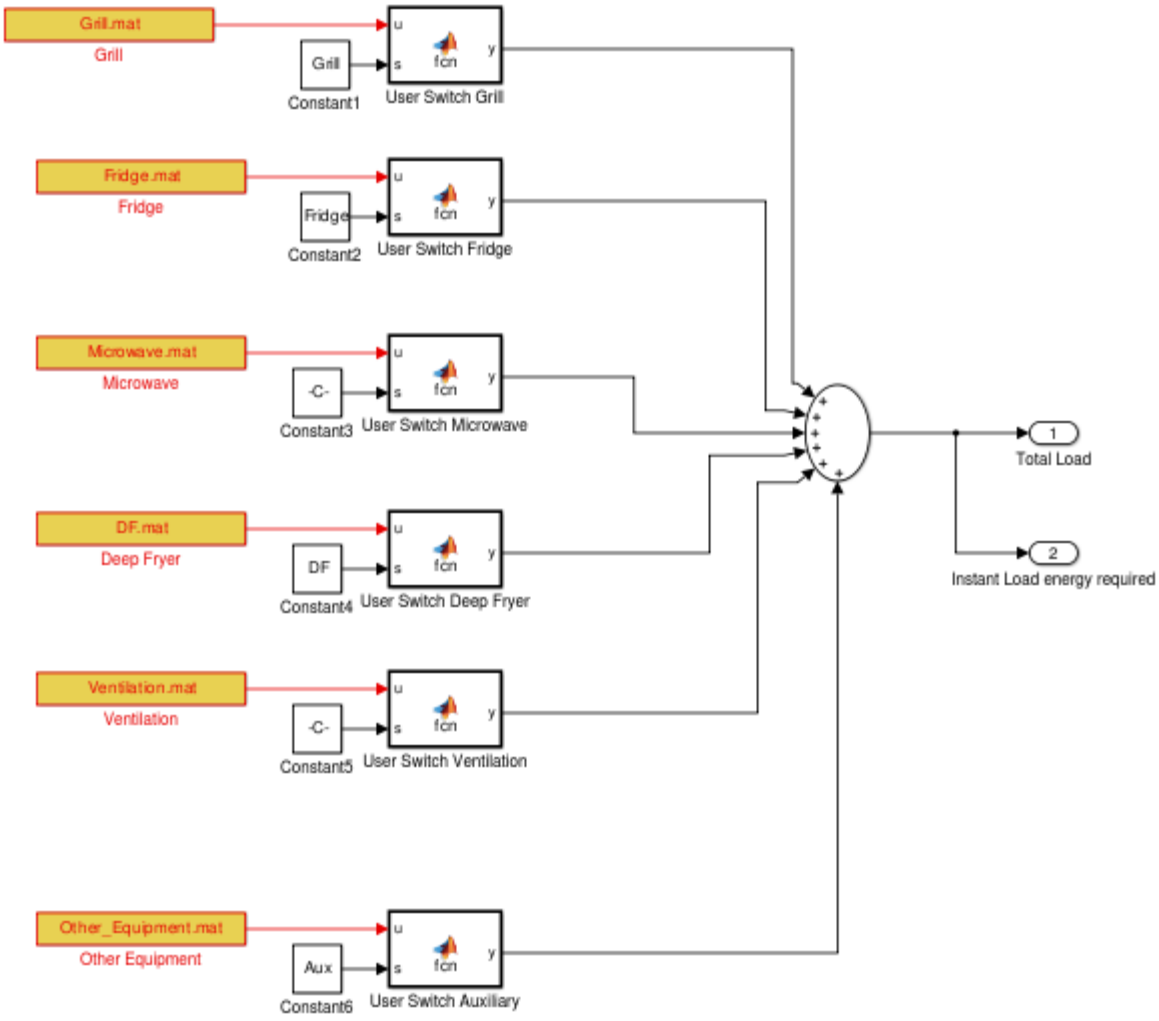

Figure 4 - Loads subsystem

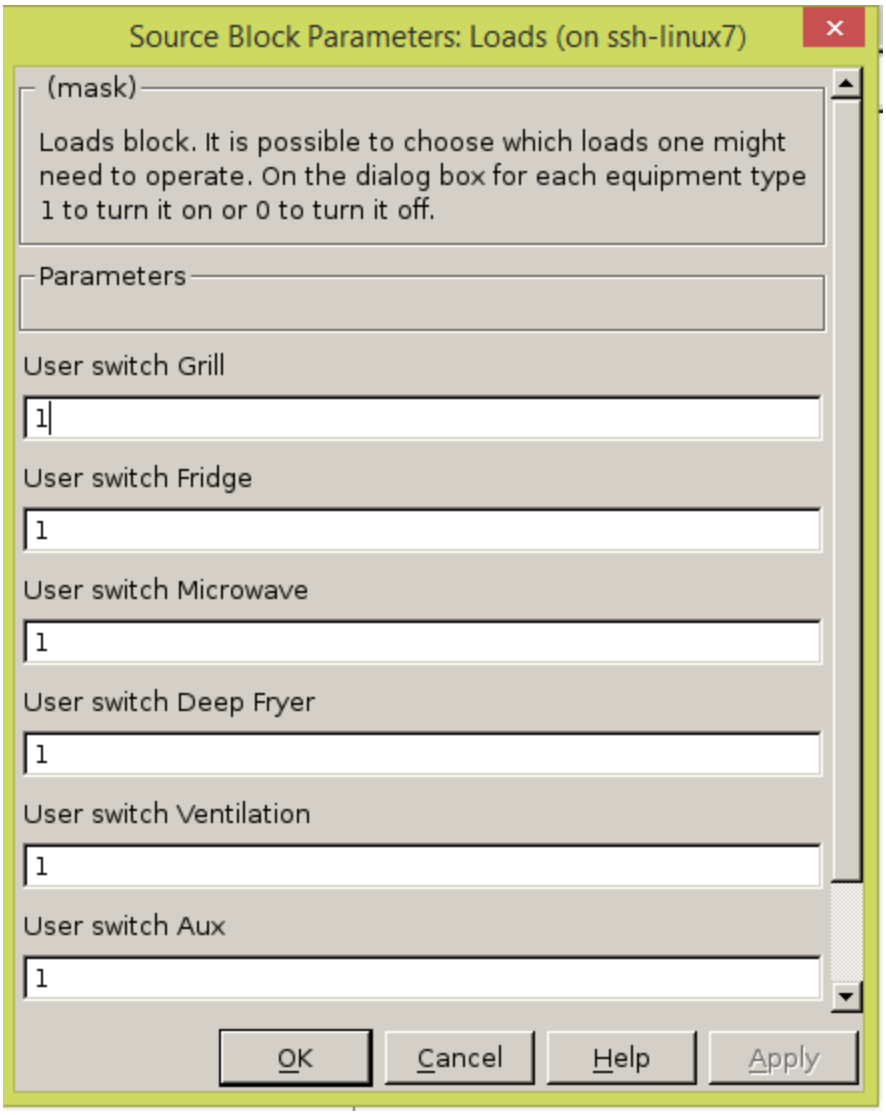

Figure 5 - User switch interface for the loads

2.3. The Management Algorithm and Subsystems

The Management strategy is the main part of the project, assuming it adopts a suitable solution to the source prioritization in order to fulfill the energy needs or to demonstrate the viability of the system in terms of energy consumption and storage.

In this algorithm, the solar power was considered primary, which means it will always be on, unless switched off by the user. As a secondary source, the fuel cells provide the energy needed by the loads when the solar power is not sufficient. And finally, a battery was used as a back-up source to assist both primary and secondary sources when they cannot satisfy the demand.

In Figure 7, it is possible to note that the strategy adopted utilizes many switches to manage the sources and loads as well as to give the user the power to choose which subsystem will be operating during a simulation. The outputs of the management block labeled "s1, s2, s3, s4 and s5" are the decisions taken by the algorithm, these are sent to the switches, which will activate or deactivate the adjacent subsystems. It can be noted that "s5" has no connection, this is due to its purpose to deactivate the loads when the system is overloaded. This function not implemented. For simulation purposes, it is useful to know at which point the energy provided by the loads was not enough, in addition to the amount of energy that couldn't be satisfied. Therefore, the loads' emergency shutdown was not implemented, although the entry for such is ready. To do so, it would only require a switch between the loads and the management block, as well as some minor changes on the final lines of the management code.

Figure 8 demonstrates the flowchart for the MATLAB function block that characterizes the management algorithm. The code to this block is represent in Appendix 3. It is important to notice that the sequence repeats for every time step as it is necessary to constantly compare the sources to the loads.

Furthermore, the management system provides as outputs the final power balance when power from sources and loads have already been processed, the fuel and battery power needed to satisfy the demand in addition to the excessive power that is sent to the battery charging system.

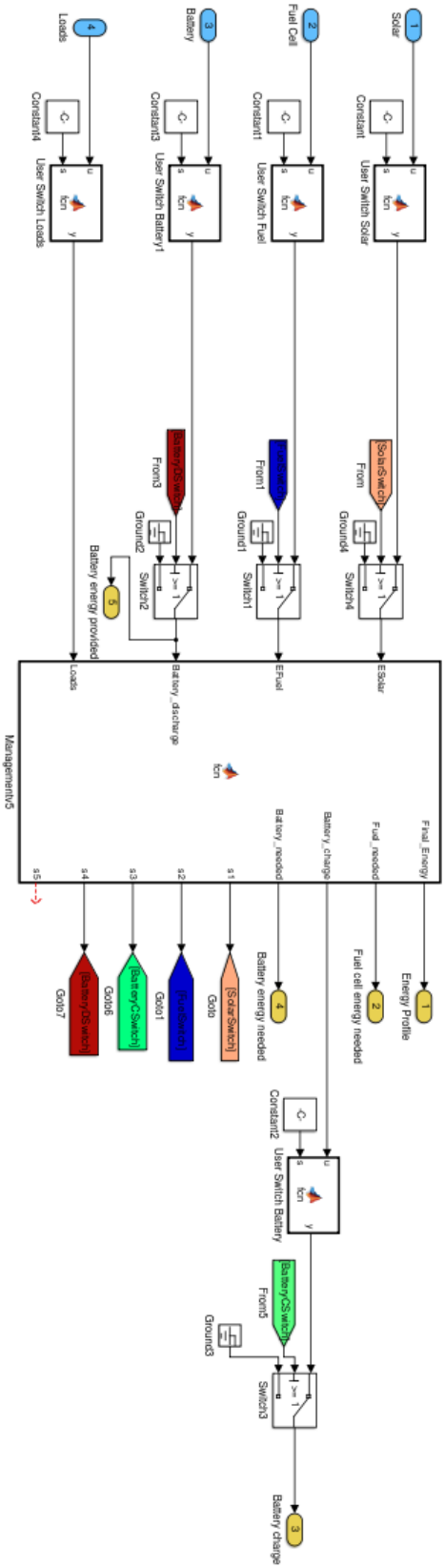

Figure 6 - Management subsystem

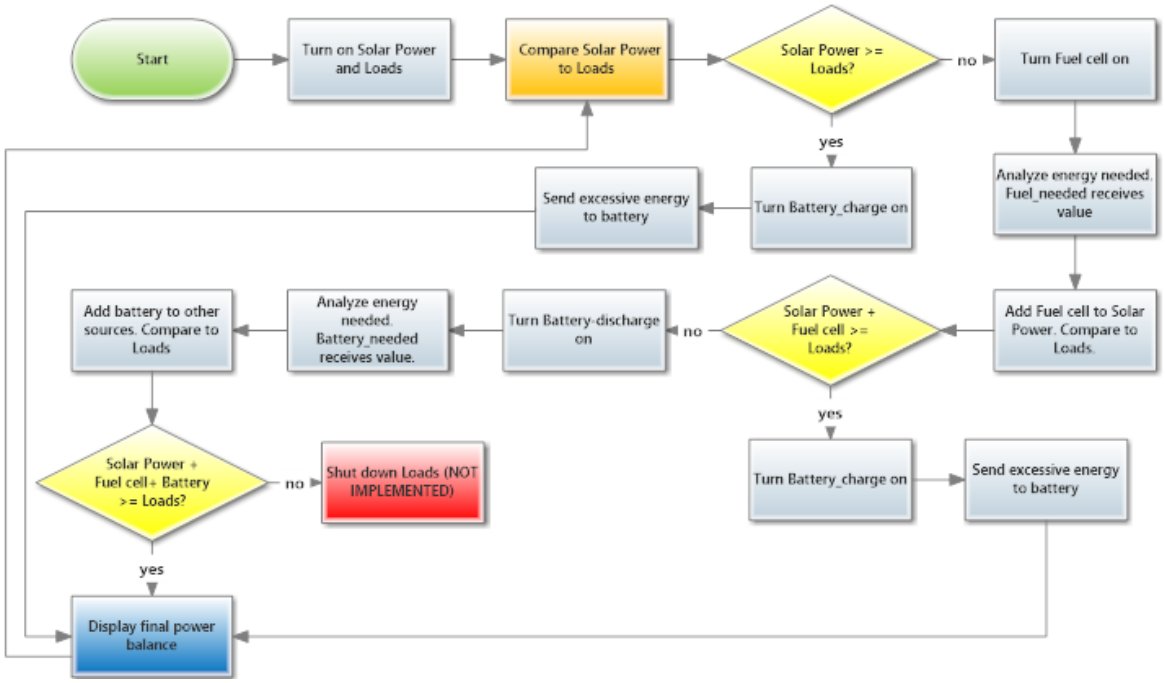

Figure 7 - Management Algorithm

2.4. Fully assembled simulation System and Plotting System

When fully assembled (Figure 9), the system picks specific points of the subsystems (as seen in purple tags) and sends this information to the plotting system (Figure 10) that will generate the graphs and data to be analyzed by the user.

In Figure 10, the yellow scope blocks represent the most relevant information to be evaluated, therefore, they are highlighted for easy identification.

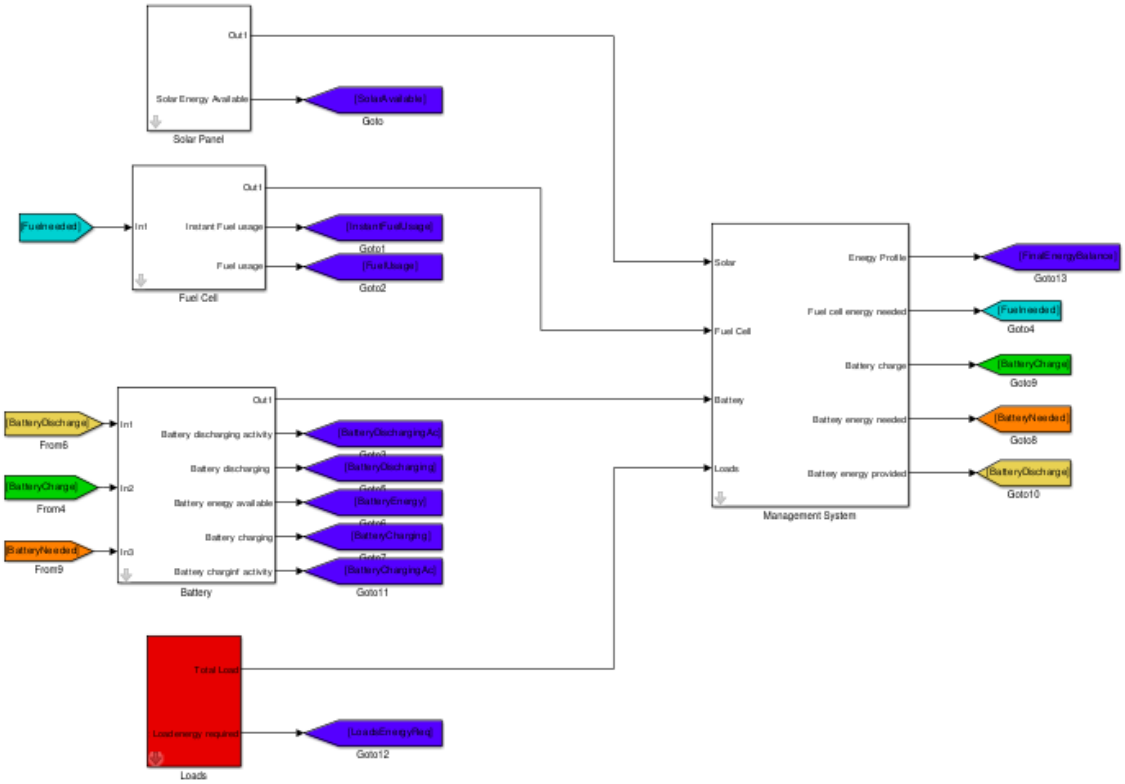

Figure 8 - Fully Assembled Simulation System

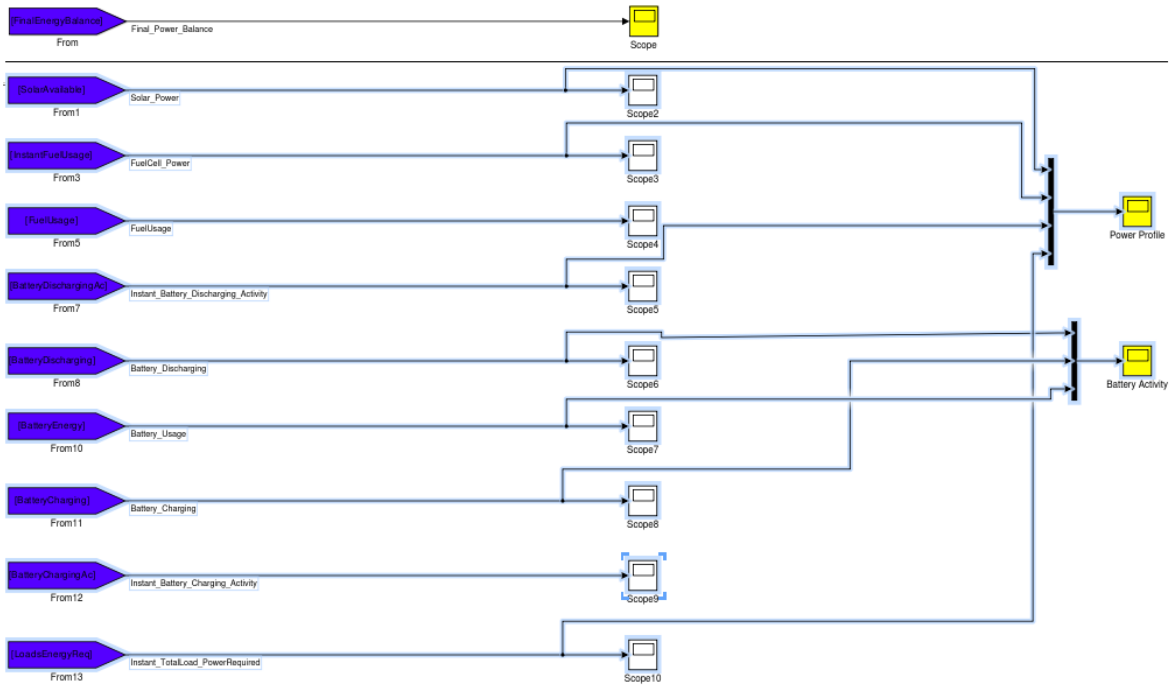

Figure 9 - Plotting system (displaying the highlighted scopes)

3.RESULTS

3.1. Scenario 1 : Sunny busy day (Summer)

For this scenario it was considered an operation time of 8 hours and day time from 6:00AM to 9:00PM (which determines the amount of solar energy available to power the loads and recharge the battery). Table 1 shows some specifications for this scenario which includes the operation time from 10:00AM to 6:00PM. All the parameters displayed on this table were used to create the loads profiles and program the sources.

Simulation Scenario 1

Busy Sunny day Operation Profiles (Summer)

Power Source	Max Output		Operating Time (hours)
Solar panels	10*0.24kWh*15h = 36kWh		6AM - 9PM
Fuel cells	30kWh with a maximum output rate of 5000Wh		Depends on Power Output Rates
Battery	10kWh		Depends on demand
Loads	Power (W)	Simulation Intervals (5 minutes/cycle)	Operating Time (hours)
Deep fryer	5000	ON During operation time	10AM - 6PM
Griddle	5000	ON 90 minutes OFF 30 minutes	10AM - 6PM
Microwave oven	1000	ON 5 minutes OFF 5 minutes	10AM - 6PM
Fridge	700	ON 10 minutes OFF 50 minutes	10AM - 6PM
Ventilation	200	ON During operation time	10AM - 6PM
Others	150	ON During operation time	10AM - 6PM

Table 1 - Sunny busy day scenario

In the graphs below (Figures 11 to 13) , it was used 14 (fourteen) solar

panels with an area of 0.5m^2 each, which happen to be the first value to satisfy the loads when added to a 10kWh Battery and 30kWh Fuel cell (with a maximum output rate of 5kWh). It is important to notice that the x-axis of every graph represents the time in hours as in a full day and for Figures 11 and 12, the y-axis represents power (in Watts) in each step, However, Figure 13 represents the battery energy usage (in Wh). The energy was found by integrating the power by the time step. Figure 11 shows the final power balance through the day, after the loads have been subtracted from the sources. Moreover, it is possible to view how the each subsystem behaved during the operation time in Figure 12.

As additional data to the analysis, Figure 13 illustrates the battery energy usage (Energy stored during the day, discharging and charging systems operations).

Figure 10 - Final Power Balance [Power(W) vs. Time(h)]

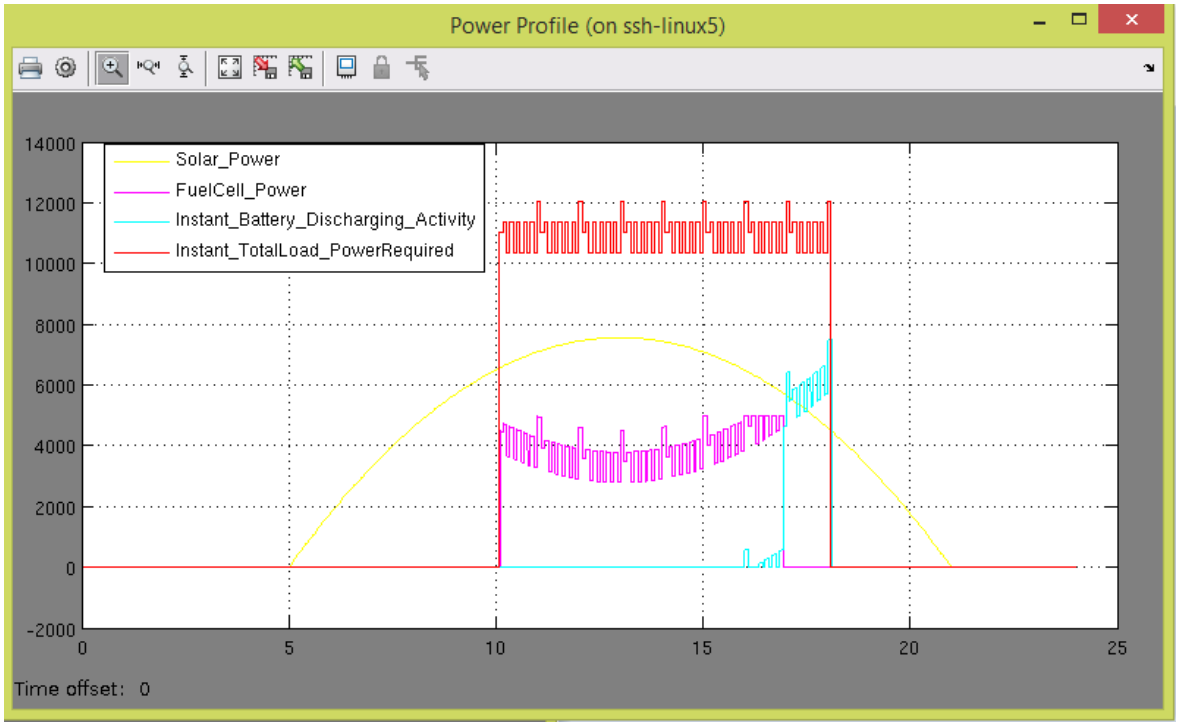

Figure 11 - Sources and loads power profiles [Power (W) vs. Time (h)]

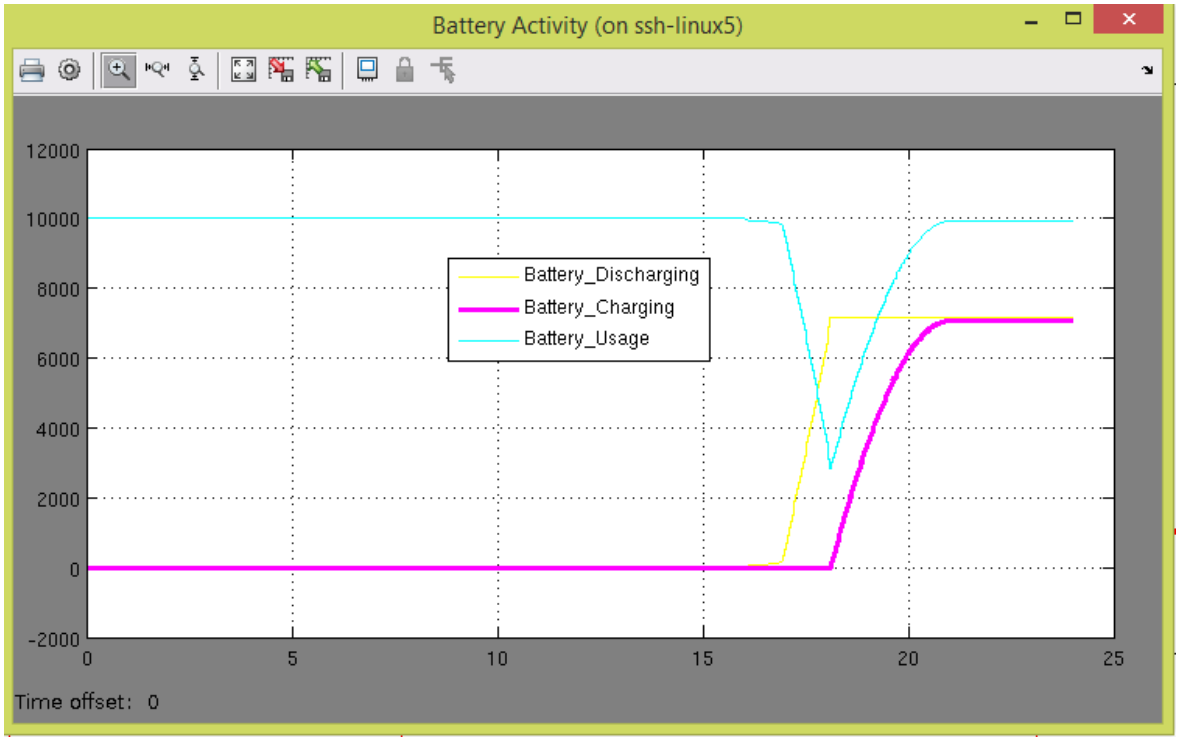

Figure 12 - Battery activities [Energy (Wh) vs. Time (h)]

Running this scenario, all the sources had to be used simultaneously to meet the loads requirements. However, the energy stored in the battery was not used to its maximum, making it possible to almost fully recover the energy used only using the solar power that was left after the operation time.

4. CONCLUSION

In summary, the results of the simulator were consistent and they reveal that it is possible to run a sustainable food truck with heavy loads on solar, fuel cell and battery energy on a sunny day using the parameters selected and the load profiles assumed. However, the data acquired also states that it is not possible to integrate the motor to this system in order to help move the truck, considering the energy was short and there was not enough left in the battery to complete such task. Which also reveals that a cloudy and busy day could face problems if all the loads were to be working at the same time. It can be deduced from this scenario that another sunny day scenario not happening during summer would have few changes, such as less energy charged into the battery after the operation hours.

Although the project of a sustainable food truck requires zero GHG emission. The AMS should consider other technologies to implement to the truck's mobility system as well as the electrical system in order to prevent energy shortage while operating the truck. Bio-fuels might be a suitable option to be used in the motor or even in a generator (one could make good use of another source of energy).

5.APPENDIX

5.1. Appendix 1 (Solar Panel code)

```
1   function ESolar  = fcn(Radiation,Number_panels,Area_panel, Efficiency)
2   %#codegen
3 - ESolar = Radiation * Number_panels * Area_panel * Efficiency;
```

5.2. Appendix 2 (Fuel cell Limit MATLAB function block)

```
function y = fcn(u, s, max)
%#codegen

y = u;


if s == max
y = 0;

else
    y = u;
end
```

5.3. Appendix 30 (Management Algorithm Code)

```
function [Final_Energy,Fuel_needed, Battery_charge, Battery_needed, s1, s2, s3, s4, s5] = ...
                                                    fcn(ESolar,EFuel,Battery_discharge, Loads)
%#codegen
Final_Energy = ESolar - Loads;
s1=1;
s2=1;
s3=0;
s4=0;
s5=1;
Fuel_needed = 0;
Battery_charge = 0;
Battery_needed = 0;
if ESolar >= Loads
s1 = 1;
s2 = 0;
s3 = 1;
s4 = 0;
Final_Energy = ESolar  - Loads;
Battery_charge = ESolar - Loads;
Battery_needed = 0;
end
if ESolar < Loads
    s1=1;
    s2=1;
    s3=0;
    s4=0;
    Fuel_needed = Loads - ESolar;
    if Fuel_needed < 0
        Fuel_needed = 0;
    end

    if (ESolar + EFuel) >= Loads
```

```
        s3=1;
        s4=0;
        Final_Energy = ESolar + EFuel - Loads;
        Battery_charge = ESolar + EFuel - Loads;
    end
    if (ESolar + EFuel) < Loads
        s3=0;
        s4=1;
        Battery_needed = Loads - ESolar - EFuel;
        if (ESolar + EFuel + Battery_discharge)>= Loads
        Final_Energy = ESolar + EFuel + Battery_discharge - Loads;
        end
        if (ESolar + EFuel + Battery_discharge) < Loads
            Final_Energy = ESolar + EFuel + Battery_discharge - Loads;
        end
    end

end
```

5.4. Appendix 4 (MATLAB codes for generating input profiles)

```
% EECE 490L - AMS Sustainable Food Truck
% Source / Load Profile Generation Script
% March 2014
% Paul Lusina

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Operating Conditions
% Daily Vector, 5 min intervals
clear all
Scenario_Name = 'Scenario_Files';
Sample_Time = 5; % minutes
Sample_Time_Hours = 5/60;

%Check if scenario directory exists, otherwise create it
Dir_List = ls;
Dir_Exists = 0; % flag set to 1 if directory exists
for indx = 3:size(ls,1)
    if eval(['strcmp(''' Scenario_Name ''',''' ...
            Dir_List(indx,1:size(Scenario_Name,2)) ''')'])
        Dir_Exists = 1;
    end
end

if Dir_Exists == 0
    mkdir('.',Scenario_Name);
end

% Day time vector (24 hours)
Day_Start = 0; % 24 hour time
Day_End = 24; % 24 hour time
Day_Window = (Day_End - Day_Start)*60; % minutes
Day_Number_of_Samples = floor(Day_Window / Sample_Time); % samples
Day_Vector_Hours = Day_Start:Sample_Time/60:Day_End-(Sample_Time/60);
Day_Vector_Min = Day_Start*60:Sample_Time:Day_End*60 -(Sample_Time);
Day_Vector_Steps = 1:size(Day_Vector_Min,2);

% Operating time vector
Operation_Start = 10; % 24 hour time
Operation_End = 18; % 24 hour time
Operation_Start_Sample_Number = floor(Operation_Start*60/Sample_Time)+1;
Operation_End_Sample_Number = floor(Operation_End*60/Sample_Time)+1;

% Operation Vector
Zeros_Start=zeros(1,Operation_Start_Sample_Number);
Ones_Operation=ones(1,(Operation_End_Sample_Number)...
    -Operation_Start_Sample_Number);
Zeros_End=zeros(1,(Day_Number_of_Samples)...
    -Operation_End_Sample_Number);
Operation_Vector = [Zeros_Start Ones_Operation Zeros_End];
```

```
% Save Data
Operation_Data = save_time_data( ...
    'Operation', Day_Vector_Hours, Operation_Vector, 'units', Scenario_Name);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generation Solar Profile, Variables
Sun_Up_Hour = 6;
Sun_Down_Hour = 20;
Radiation_Max = 800; %W/m^2

% Solar Operating time vector

Solar_Operation_Start_Sample_Number = floor(Sun_Up_Hour*60/Sample_Time)+1;
Solar_Operation_End_Sample_Number = floor(Sun_Down_Hour*60/Sample_Time)+1;

% Solar Operation Vector
Solar_Zeros_Start=zeros(1,Solar_Operation_Start_Sample_Number);
Solar_Ones_Operation=ones(1,(Solar_Operation_End_Sample_Number)...
    -Solar_Operation_Start_Sample_Number);
Solar_Zeros_End=zeros(1,(Day_Number_of_Samples)...
    -Solar_Operation_End_Sample_Number);
Solar_Operation_Vector = [Solar_Zeros_Start Solar_Ones_Operation
Solar_Zeros_End];

Solar_Operation_Data = save_time_data( ...
    'Solar_Operation', Day_Vector_Hours, Solar_Operation_Vector, 'units',
Scenario_Name);


% Solar Profile, Parabolic Equation
Sun_Up = floor(Sun_Up_Hour*60/Sample_Time);
Sun_Down = floor(Sun_Down_Hour*60/Sample_Time);
Sun_Shine = (Sun_Down-Sun_Up);
Sun_Mid = Sun_Shine/2+mod(Sun_Shine+1,2);
Sun_Shine_Vector = [0:Sun_Shine];
Radiation_Points_X = [0 Sun_Mid Sun_Shine];
Radiation_Points_Y = [0 Radiation_Max 0];
Radiation_Coefficients = ...
    polyfit(Radiation_Points_X,Radiation_Points_Y,2);
Radiation_Curve = ...
    Radiation_Coefficients(1)*(Sun_Shine_Vector).^2+ ...
    Radiation_Coefficients(2)*(Sun_Shine_Vector);
Radiation_Vector = [...
    zeros(1,Sun_Up) ...
    Radiation_Curve ...
    zeros(1,Day_Number_of_Samples-Sun_Down-1)].* ...
    Solar_Operation_Vector;

% Save Data
Radiaion_Data = save_time_data( ...
    'Radiation', Day_Vector_Hours, Radiation_Vector, 'W/m^2', Scenario_Name);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load, Microwave, Variables
```

```
Microwave_Power_Max = 1000; %Watts
Microwave_On = 5; % Minutes
Microwave_Off = 5; % Minutes

% Microwave Profile - Duty Cycle
Microwave_Vector = cycle_function(Microwave_Power_Max, ...
    Microwave_On,Microwave_Off, ...
    Sample_Time, Day_Number_of_Samples, Operation_Vector);

% Save Data
Microwave_Data = save_time_data( ...
    'Microwave', Day_Vector_Hours, Microwave_Vector, 'W', Scenario_Name);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load, Fridge, Variables

Fridge_Power_Max = 100; %Watts
Fridge_On = 10; % Minutes
Fridge_Off = 50; % Minutes

% Fridge Profile - Duty Cycle
Fridge_Vector = cycle_function(Fridge_Power_Max, ...
    Fridge_On,Fridge_Off, ...
    Sample_Time, Day_Number_of_Samples, Operation_Vector);

% Save Data
Fridge_Data = save_time_data( ...
    'Fridge', Day_Vector_Hours, Fridge_Vector, 'W', Scenario_Name);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load, Air Conditioning, Variables

AC_Power_Max = 1200; %Watts
AC_On = 90; % Minutes
AC_Off = 30; % Minutes

% AC Profile - Duty Cycle
AC_Vector = cycle_function(AC_Power_Max, ...
    AC_On,AC_Off, ...
    Sample_Time, Day_Number_of_Samples, Operation_Vector);

% Save Data
AC_Data = save_time_data( ...
    'AC', Day_Vector_Hours, AC_Vector, 'W', Scenario_Name);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load, Ventilation, Variables

Ventilation_Power_Max = 200; %Watts
Ventilation_On = 5; % Minutes
Ventilation_Off = 0; % Minutes

% Ventilation Profile - Duty Cycle
Ventilation_Vector = cycle_function(Ventilation_Power_Max, ...
```

```
    Ventilation_On,Ventilation_Off, ...
    Sample_Time, Day_Number_of_Samples, Operation_Vector);

% Save Data
Ventilation_Data = save_time_data( ...
    'Ventilation', Day_Vector_Hours, Ventilation_Vector, 'W', Scenario_Name);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load, Other Equipment, Variables

Other_Power_Max = 110; %Watts
Other_On = 5; % Minutes
Other_Off = 0; % Minutes

% Ventilation Profile - Duty Cycle
Other_Vector = cycle_function(Other_Power_Max, ...
    Other_On,Other_Off, ...
    Sample_Time, Day_Number_of_Samples, Operation_Vector);

% Save Data
Other_Data = save_time_data( ...
    'Other_Equipment', Day_Vector_Hours, Other_Vector, 'W', Scenario_Name);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Load, Grill, Variables

Grill_Power_Max = 3000; %Watts
Grill_On = 5; % Minutes
Grill_Off = 0; % Minutes

% Grill Profile - Duty Cycle
Grill_Vector = cycle_function(Grill_Power_Max, ...
    Grill_On,Grill_Off, ...
    Sample_Time, Day_Number_of_Samples, Operation_Vector);

% Save Data
Grill_Data = save_time_data( ...
    'Grill', Day_Vector_Hours, Grill_Vector, 'W', Scenario_Name);
```

```
% % Save Routine
% save AMS_Food_Truck_Source_Load_Profiles.mat

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Cycle_Vector = cycle_function(Power_Max,Cycle_On,Cycle_Off,
Sample_Time, Num_Samples, Op_Vector)

% Power_Max = maximum power used (Watts)
% Cycle_On = number of minutes the device is on
% Cycle_Off = number of minutes the device is off
% Sample_Time = sample time in minutes
% Num_Samples = total number of samples to be simulated
% Op_Vector = vector in which the system is on

%Sample variables
%clear all
%Power_Max = 1000; %Watts
%Cycle_On = 5; % Minutes
%Cycle_Off = 10; % Minutes
%Sample_Time = 5;
%Num_Samples = 90;
%Op_Vector = [zeros(1,30) ones(1,30) zeros(1,30)];

% Profile - Duty Cycle
On = floor(Cycle_On/Sample_Time);
Off= floor(Cycle_Off/Sample_Time);
Cycle = [ones(1,On) ...
    zeros(1,Off)];
Num_Cycles = floor(Num_Samples/ ...
    length(Cycle));
Num_Points = Num_Cycles*length(Cycle);
Curve = Power_Max*reshape( ...
    Cycle'*ones(1,Num_Cycles), ...
    1,Num_Points);
if Num_Points == Num_Samples
    Cycle_Vector = Curve.*Op_Vector;
else
    Buffer_Zeros = zeros(1,Num_Samples-Num_Points);
    Cycle_Vector = [Curve.*Op_Vector ...
        Buffer_Zeros];
end % End If

end % End Function
```

```
function Vector_Data = save_time_data( ...
    VectorName, VectorTime, VectorData, VectorUnits, DataFolder)
%
% This function creates and saves information in a time series format which
% can be read by a simulink module.
%
% VectorName - name describing the data
% VectorTime - vector with the time samples
% VectorData - data points corresponding to the time sample
% VectorUnits - units of the data points
% DataFolder - location where the file is stored
%

% Test Data
% clear all
% VectorName = 'Test_Vector';
% VectorTime = 1:10;
% VectorData = ones(size(VectorTime));
% VectorUnits = 'm';
% DataFolder = 'Example_Files';

% Construct the timeseries vector.
Vector_Data= timeseries;
Vector_Data.Name = VectorName;
Vector_Data.Time = reshape(VectorTime,max(size(VectorTime)),1);
Vector_Data.Data = reshape(VectorData,max(size(VectorData)),1);
Vector_Data.DataInfo.Units = VectorUnits;

eval(['cd ' DataFolder])
eval(['save(''' VectorName ''', ''Vector_Data'', ''-v7.3'')'])
eval(['cd ..'])



function FigureData = save_figure(DirectoryName, CurrentFigure,
FigureFormat, ...
    FigureFile)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% This function loads time data
% DirectoryName:  Directory where data is stored
% FileName: File name with the data
%
% DirectoryName = 'Example_Files';
% FigureFile=strcat('AC','_','Profile');
% CurrentFigure = 1;
% FigureFormat = '-dpng';

eval(['cd ' DirectoryName]);
print(CurrentFigure,FigureFormat, FigureFile)
eval(['cd ..']);

FigureData = 1;
```

6.REFERENCES

[1] “AMS Sustainability Initiatives”. Available: http://amssustainability.ca/initiatives/;

[2] “The AMS Lighter Footprint Strategy”. Available: http://amssustainability.ca/the-ams-lighter-footprint-strategy/;

[3] Shau Hua (Raymond) Hou. “'Peak Shave' Energy Managing tool for Energy Subsystems Design of AMS Sustainable Food Truck.”, Final Report, Electrical and Computer Engineering Department, University of British Columbia, 2014