Plug-in Hybrid Vehicle (PHEV) Component Pre-Heater

Ria Pereira
Western Michigan University, ria.h.pereira@gmail.com

Follow this and additional works at: https://scholarworks.wmich.edu/honors_theses

Part of the Electrical and Electronics Commons

Recommended Citation
https://scholarworks.wmich.edu/honors_theses/2272
Plug-In Hybrid Electric Vehicle Component Pre-Heater

Nick Munyan, Ria Pereira, Mikhail Sokolov
Faculty Advisor: Dr. Atashbar
Electrical and Computer Engineering 4820
16 April, 2013
Faculty Sponsor: Dr. John Patten
Corporate Sponsor: DENSO Foundation
DISCLAIMER

This report was generated by a group of engineering seniors at Western Michigan University. It is primarily a record of a project conducted by these students as part of curriculum requirements for being awarded an engineering degree. Western Michigan University makes no representation that the material contained in this report is error free or complete in all respects. Therefore, Western Michigan University, its faculty, its administration or the students make no recommendation for use of said material and take no responsibility for such usage. Thus persons or organizations who choose to use said material for such usage do so at their own risk.
1.0 Abstract

In plug-in hybrid electric vehicles, the optimum temperatures of the cabin and various fluids are achieved through the dissipating heat of the internal combustion engine. In colder climates the initial warm-up of components requires the internal combustion engine to run causing unnecessary consumption of fuel.

This project improved the pre-heating system formerly in place. It uses a variety of electric heaters along with the positive temperature coefficient (PTC) heater and blower motor already installed on the vehicle. A system utilizing a microcontroller turns the heaters on and off based on the inputs from the temperature sensors installed on the vehicle.

In order to improve the previous system, this project utilizes seven relays instead of two. Each component is controlled by a designated relay, which improves the functionality and control of which components are pre-heated. With the addition of new heaters, this system is able to pre-heat more components, and is able to cycle which heaters are on, in order to achieve optimal temperatures.

Besides better control and improved functionality, this project improved the dependability of the system. The circuitry required to cycle the relays, along with the microcontroller, were mounted onto a printed circuit board, which was housed in an enclosure. The required relays were also housed in an enclosure and are easily replaceable in a case of a future failure.

This system interfaces with an Android application developed by previous groups. The application allows users to preset a schedule to control when the system automatically turns on based on their daily requirements.
2.0 Senior Design Project Report Release Form

WESTERN MICHIGAN UNIVERSITY
COLLEGE OF ENGINEERING AND APPLIED SCIENCES
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
KALAMAZOO, MICHIGAN 49008

SENIOR DESIGN PROJECT REPORT RELEASE FORM

In accordance with the "Policy on Patents and Release of Reports Resulting from Senior Design Projects" as adopted by the Executive Committee Of the College of Engineering and Applied Sciences on Feb. 9, 1989, permission is hereby granted by the individuals listed below to release copies of the final report written for the Senior Design Project entitled:

PROJECT TITLE: Plug-In Hybrid Electric Vehicle Component Pre-Heater

_____________________________________________________________

PROJECT SPONSOR* Did this project have a sponsor? YES X (see footnote) NO____

Contact person and email address &/or telephone  Dr. John Patten  john.patten@wmich.edu  (269) 276-3246

Company Name  Denso Foundation ________________________________________

Design team has requested sponsor to verify in writing to course coordinator that all promised deliverables have been received. YES X NO____ (please check)

Nick Munyan, Ria Pereira, Mikhail Sokolov

TEAM MEMBERS NAMES:

<table>
<thead>
<tr>
<th>NAME PRINTED</th>
<th>NAME SIGNED</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nick Munyan</td>
<td>___________</td>
<td>04/15/2013</td>
</tr>
<tr>
<td>Ria Pereira</td>
<td>___________</td>
<td>04/15/2013</td>
</tr>
<tr>
<td>Mikhail Sokolov</td>
<td>___________</td>
<td>04/15/2013</td>
</tr>
</tbody>
</table>

* Those teams with a sponsor must have sponsor provide the course coordinator with written evidence that they have provided the sponsor with a copy of the final project report as well as with other items that the team has promised to the sponsor. The evidence could be a short note via email, fax or US mail from the sponsor indicating receipt of a copy of the report and all promised deliverables.

J. Gesink 11/08
3.0 Sponsor Acknowledgement

The faculty sponsor for this project is Dr. John Patten, chair of the Manufacturing Engineering Department at Western Michigan University. Dr. Patten has substantial experience and knowledge of the systems utilized by the vehicle and has complete access to the test vehicle. Dr. Patten has sponsored similar projects in the past. Most of the research funding along with the test vehicle were provided by DENSO Foundation. The grant provided by DENSO was used to purchase components and gain access to needed information and was not used to compensate for the time spent on this project.
# Table of Contents

1.0 Abstract ......................................................................................................................... 3
2.0 Senior Design Project Report Release Form ................................................................. 4
3.0 Sponsor Acknowledgement .......................................................................................... 5
4.0 Introduction ..................................................................................................................... 9
   4.1 Need ............................................................................................................................ 9
   4.2 Scope .......................................................................................................................... 9
   4.3 Background ............................................................................................................... 9
   4.4 Literature Search ........................................................................................................ 10
   4.5 Patent Search ............................................................................................................. 13
5.0 Description ..................................................................................................................... 15
   5.1 Original Specifications ............................................................................................... 15
   5.2 Original Design Concept .......................................................................................... 15
   5.3 Modification to Original Design ............................................................................... 17
6.0 Implementation ............................................................................................................. 20
   6.1 Power Supplies ........................................................................................................... 20
   6.2 Relays ......................................................................................................................... 21
   6.3 Power Circuit Schematics .......................................................................................... 22
   6.4 Circuit Implementation .............................................................................................. 23
   6.5 Blower Motor and PTC Heaters ............................................................................... 26
   6.6 Bill of Materials Tables ............................................................................................. 29
   6.7 Code Description ....................................................................................................... 30
7.0 Results ........................................................................................................................... 33
8.0 Economic Analysis ........................................................................................................ 34
9.0 References ..................................................................................................................... 36
10.0 Appendix A – Circuit Calculations, Simulations, and Measurements ...................... A1
11.0 Appendix B – PCB Design .......................................................................................... B1
12.0 Appendix C – Test Results ......................................................................................... C1
13.0 Appendix D – Heaters ................................................................................................. D1
14.0 Appendix E – Code ..................................................................................................... E1
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1: Design Concept Block Diagram</td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>Figure 2: DC-DC Linear Regulator</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>Figure 3: Double Pole Single Throw Relay and Single Pole Single Throw Relay</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>Figure 4: Power Circuit for 120VAC Components</td>
<td></td>
<td>22</td>
</tr>
<tr>
<td>Figure 5: Power Circuit for 12 VDC Components</td>
<td></td>
<td>23</td>
</tr>
<tr>
<td>Figure 6: PCB Schematic</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>Figure 7: PCB Box</td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>Figure 8: Relay Housing</td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>Figure 9: PTC Heater Wiring Diagram</td>
<td></td>
<td>27</td>
</tr>
<tr>
<td>Figure 10: Blower Motor Control</td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>Figure 11: Blower Motor Control Circuit</td>
<td></td>
<td>28</td>
</tr>
<tr>
<td>Figure 12: Code Flowchart</td>
<td></td>
<td>31</td>
</tr>
<tr>
<td>Figure 13: Final Test Results</td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>Figure A1: Relay Circuit</td>
<td></td>
<td>A1</td>
</tr>
<tr>
<td>Figure A2: Circuit Simulation Results</td>
<td></td>
<td>A3</td>
</tr>
<tr>
<td>Figure B1: PCB - Top Layer</td>
<td></td>
<td>B1</td>
</tr>
<tr>
<td>Figure B2: PCB - Bottom Layer</td>
<td></td>
<td>B2</td>
</tr>
<tr>
<td>Figure B3: PCB - Mechanical</td>
<td></td>
<td>B3</td>
</tr>
<tr>
<td>Figure B4: PCB - 3D Rendering</td>
<td></td>
<td>B4</td>
</tr>
<tr>
<td>Figure C1: Oil Pan Control</td>
<td></td>
<td>C1</td>
</tr>
<tr>
<td>Figure C2: Battery Control</td>
<td></td>
<td>C1</td>
</tr>
<tr>
<td>Figure C3: Cabin Control</td>
<td></td>
<td>C2</td>
</tr>
<tr>
<td>Figure C4: Aux Heaters and Blower Motor Control</td>
<td></td>
<td>C2</td>
</tr>
<tr>
<td>Figure C5: Vehicle Warm-Up Control</td>
<td></td>
<td>C2</td>
</tr>
<tr>
<td>Figure C6: Oil Pan and Engine Block Heater</td>
<td></td>
<td>C3</td>
</tr>
<tr>
<td>Figure C7: Blower Motor, Aux Heaters and Cabin Heater</td>
<td></td>
<td>C3</td>
</tr>
<tr>
<td>Figure C8: Blower Motor, Cabin Heater, Block Heater and Aux Heaters</td>
<td></td>
<td>C4</td>
</tr>
<tr>
<td>Figure C9: Battery Heater, Cabin Heater, Blower Motor and Aux Heaters</td>
<td></td>
<td>C4</td>
</tr>
<tr>
<td>Figure C10: Temperature Dependent Test One</td>
<td></td>
<td>C5</td>
</tr>
<tr>
<td>Figure C11: Temperature Dependent Test Two</td>
<td></td>
<td>C6</td>
</tr>
<tr>
<td>Figure C12: Temperature Dependent Test Three</td>
<td></td>
<td>C7</td>
</tr>
<tr>
<td>Figure C13: Temperature Dependent, Final Test</td>
<td></td>
<td>C8</td>
</tr>
<tr>
<td>Figure D1: Cabin Heater</td>
<td></td>
<td>D1</td>
</tr>
<tr>
<td>Figure D2: Oil Pan Heater</td>
<td></td>
<td>D1</td>
</tr>
<tr>
<td>Figure D3: Engine Block Heater</td>
<td></td>
<td>D2</td>
</tr>
<tr>
<td>Figure D4: Battery Heater</td>
<td></td>
<td>D2</td>
</tr>
</tbody>
</table>
Table of Tables

Table 1: Power Availability Comparison ................................................................. 17
Table 2: PCB Bill of Materials .................................................................................. 29
Table 3: Relay Box Bill of Materials ........................................................................ 29
Table 4: Digital Bill of Materials and Totals .............................................................. 30
Table A1: Circuit Measurements .............................................................................. A3
4.0 Introduction

4.1 Need

Plug-in hybrid electric vehicles (PHEVs) have much lower starting efficiencies during the colder months of the year. This problem requires a system that will improve the gas mileage and create a comfortable environment of the vehicle during the cold months, without using any fuel in the process. A system was needed to be designed, built, and tested in order to pre-heat certain engine components, along with the cabin of the vehicle using only the grid electrical power.

4.2 Scope

The scope of this project entails the amount of work that could be done by three Western Michigan University full time students in the electrical engineering program. Since all three members of this group are in electrical engineering, this project focuses more on the realization of the power source circuits, and the control system within the microcontroller. The user interface and the Bluetooth communication aspect of the project have been reused from prior groups.

4.3 Background

The need for energy efficiency is becoming more and more prevalent in engineering design, especially in the automotive industry. Initially, hybrid electric vehicles were developed to accomplish this. Hybrid electric vehicles use a combination of a combustion engine and an electric propulsion system to function. The electric system reduces emissions, improves the fuel economy by using less gasoline, and lowers the overall operational cost of the vehicle. Now, plug-in hybrid electric vehicles (PHEVs) are being designed to further improve gas mileage, lowering operational costs. PHEVs, as the name suggests, utilize batteries that can be charged
while the car is plugged into the grid power supply. PHEVs reduce the amount of gasoline the car uses by allowing it to drive a short distance using only electric power with no gasoline. This electricity is available at a fraction of the cost of gasoline, further increasing the efficiency of plug-ins even more than ordinary hybrids.

Like all design solutions, there are challenges that come along with PHEVs. The biggest challenge facing plug-in electrics is the starting efficiency, the miles driven per gallon of gasoline of the vehicle. For maximum efficiency, certain components on the vehicle need to be at an optimal temperature. During the warmer months of the year this is not an issue. During the colder months of the year, however, the components are not warm when the car is first started and the efficiency is much lower. This project designs a system to pre-heat specific components while the car is plugged into the grid; this increases the efficiency of the vehicle when it is first turned on. Additionally, it is much more cost effective to pre-heat the car using the electric power from the grid instead of the combustion engine.

4.4 Literature Search

The first article, by Bradley and Frank, discusses some basic concepts behind PHEVs. It covers the different energy management modes: charge-sustaining, charge-depleting, electric vehicle, and engine only, and the requirements for each mode. It also discusses basic concepts for energy storage via the battery and the grid connection as a power source. The article provides overall conceptual explanations for PHEVs [1].

PHEVs require different AC-DC conversion techniques for all their components to function. Musavi discusses some different techniques to accomplish this. Its primary focus is on power factor corrected (PFC) rectifiers and their applications and characteristics in PHEV implementation. According to the article, the most popular PFC is a conventional boost
converter to rectify an AC signal to DC. The drawback to this is that there is a high capacitor ripple at the output and as the power level increases, the losses increase significantly. There are five different PFC circuit configurations discussed in the article [2].

In a thesis by Huang, many aspects of PHEV battery and charging characteristics are discussed. It also discusses the national standards for charging levels and their specifications. It covers the voltage and current ratings of each charging level. The thesis then goes on to describing the characteristics for powering charging stations, which is beyond the scope of this project [3].

Another source, from the IEEE magazine talks about optimizing energy from the grid while charging PHEVs. Moreover, it addresses how energy losses created in the battery could be used to heat it. Pre-heating the battery is highly beneficial in increasing the car’s mileage. The article proposes a current-controlled charging method which increases the power delivered to the battery consequently increasing its temperature [4].

A conference paper by Andrews et al. talks about the efficiency of a car driven from cold start. It further demonstrates how ambient temperature is an important variable in the car’s efficiency. It discusses the disadvantages of the lubricating oil at low temperatures. Additionally, it does a temperature study of the effect of the warm up times of the water and oil in the system on the efficiency of the car. It further lists the optimum times the coolant and oil need to be heated to reach a greater driving efficiency [5].

The sixth literature source talks about better predicting the range of PHEV, which is the distance it can travel before needing to be charged again. This is an important factor that must be as accurate as possible to the driver. Moreover, this range changes under different operating conditions and external temperature. The factors that it takes into consideration include the car speed, drive terrain, the HVAC system of the car and ambient temperature [6].
In “Improving Electric Vehicle Range Algorithms,” Rodgers discusses the working of the car batteries in cold weather conditions and further proposes a way to heat the batteries while connected to the grid supply. It describes a method of electrically powering a heater by the car batteries. The heater must be an element having a positive temperature coefficient. As a result, it increases resistance with the decreasing ambient temperature [7].

Pesaran et al. writes in a conference paper about the benefits of pre-heating the batteries in HEVs and suggests the use of AC current over DC current to do so. It further suggests the optimum temperatures the battery pack must be in, and discussing and comparing different methods of heating and cooling them. It also presents data on how much power would be needed to raise the temperature of a battery pack to different degree levels, and discussing its efficiency at these levels [8].

The final report from the mechanical engineering senior design group describes their project on making the entire HVAC system in the Toyota Prius electrical. It gives a background of the current heating system and proposes. This study further show the energy and power requirements for pre-heating the car. It illustrates the current heating system in the Generation Two Prius listing and describing the property of each of the components in it. The proposed system consists of an additional loop to the heating system that works without the operating of the engine [9].

The previous ECE group’s project proposal talks in depth about the technologies currently used by the vehicle. The proposal also talks about the system that will be implemented in the near future. The proposal includes some research done in regards to the range of the vehicle and how it could be improved by the proposed system. The group also includes the parameters used to implement the system, and includes some other solutions to the need [10].
4.5 Patent Search

Listed below are a set of various methods that have been patented; discussing various ways to improve the efficiency of an electric vehicle.

4.5.1 Method and system for enhancing the fuel economy of a hybrid electric vehicle (Patent: 8,239,082)

This patent talks about a system that comprises of a battery pack and a heat transfer device that transfers heat away from the battery. In this system, the user can opt for enhancing the fuel economy of the HEV. When this input is detected, the operation of the heat transfer devices increases; thereby, transferring heat from the battery pack at a faster rate. A fan is described as an option for a heat transfer device where the speed of the fan could be regulated for enhancing the fuel economy of the HEV [14].

4.5.2 System and method for vehicle temperature control (Patent: 8,118,237)

This is a patent on a system that controls the temperature of the cabin in the car by pre-heating or pre-cooling the cabin. This temperature control loop is an electric system that is powered by the grid supply. In the system the input signals comprise of the desired vehicle activation time, the current temperature of the vehicle cabin and the desired cabin temperature. Based on these inputs, the controller then determines the activation time of the temperature controlled loop and then activates the loop at this time to start the necessary heating or cooling processes. This system is then deactivated when either the vehicle activation time is reached or the desired cabin temperature is reached. The vehicle activation time is sent to the controller via internet protocol, text messaging or the controller-area network bus. This patent does not discuss the method of heating and cooling of the cabin, it just describes the overall working of the system of preheating or pre cooling the car [13].
4.5.3 Thermal security for hybrid vehicle recharging cable plugs device and method (Patent: 7,944,667)

This patent discusses installing an interrupter in the cable used for charging electric vehicles. The current in the cable is interrupted when the temperature in the plug reaches a specific threshold. This is shown necessary in the patent to prevent the overheating of the cable and current leakage when charged for longer periods of time [12].

4.5.4 Electric vehicle communication interface (Patent: 7,698,078)

This patent describes a communication interface for the electric vehicle and further communicating with this interface. It also aims to control and monitor the car battery. This would then be used for determining if the car needs to be charged, potential problems that would affect the battery performance and pre-determine the number of miles that the car could travel before requiring charge again. This system enables the user to initiate the heating or cooling of the battery; moreover, adjusting the battery temperature based on the parameters of external weather and durability of the battery [11].

4.5.5 Method and system for a vehicle battery temperature control (Patent: 7,154,068)

This patent highlights the necessities of preheating the battery during vehicle shut off. It further describes a method to do this which does not require any external power to power the heater. This external heater is an element having a positive temperature coefficient of resistance which draws current form the battery. Having a positive temperature coefficient of resistance, its resistance increases with decreasing ambient temperature; therefore, producing heat with the increasing resistance. This system is further designed to determine the shutdown conditions of the car to activate the system; moreover, regulating the activation based on the state of charge of the battery [7].
5.0 Description

5.1 Original Specifications

1.0 Physical Characteristics.

1.1 The circuitry must be able to fit into a glove compartment of size 6”×4”×3” or less for length, width, and depth respectively.
1.2 Must utilize a PCB to implement the needed converters and sensor circuitry.
1.3 Must include an LCD display in a visible location for the user to reference.

2.0 Functionality

2.1 User Interface
   2.1.1 Must utilize user presets for desired cabin temperature.
   2.1.2 Must display real-time information on the cabin, ambient, oil pan, coolant, and battery temperatures on the LCD screen.

2.2 Control System
   2.2.1 Must only function when the vehicle is plugged in.
   2.2.2 Must manipulate cabin temperature.
   2.2.3 Must draw less than 3 kW of power from the charger.
   2.2.4 Must control the valves in the coolant heater loop.
   2.2.5 Must shut off blower motor when cabin temperature reaches 30 °C.
   2.2.6 Must shut off oil heater when oil temperature reaches 50 °C.
   2.2.7 Must shut off coolant heater when coolant temperature reaches 100 °C.
   2.2.8 Must shut off all system components when car is unplugged.

2.3 Coolant Heater System
   2.3.1 Must use the coolant already in the system.
   2.3.2 May utilize a separate loop to pump and pre-heat the coolant.
   2.3.3 Must isolate the loop from the stock system.
   2.3.4 Must utilize the 12V pump already installed on the engine.

3.0 Hardware

3.1 Heaters
   3.1.1 Must utilize at least a 1000 W coolant heater.
   3.1.2 Must utilize an oil pan heater of at least 150 W.
   3.1.3 Must utilize 12V auxiliary pre-heaters already installed on the car.

3.2 Sensors
   3.2.1 Must read the cabin temperature in Celsius and Fahrenheit.
   3.2.2 May be located in an accessible place for ease of replacement.

5.2 Original Design Concept

To better understand the original design, one must understand the previous system. Prior to this project, the vehicle utilized a system that pre-heated the vehicle’s cabin, oil, engine block,
and the battery directly. Previous group used electric heaters to achieve optimal temperatures of these components. The components were to be cycled using a control system consisting of a microcontroller, temperature sensors, and user input from a Bluetooth capable Android device. The original design of this system replaced the cabin and engine block heaters with a single coolant heater. This heater was to be placed onto the coolant line inside the engine compartment. By the method of convection, the warmer coolant would have risen to the top, and the colder coolant sunk to the bottom, causing it to circulate.

Once the coolant circulated through the system, it passed through the heater exchange core. This core had PTC heaters which also turned on at this time. Along with the pre-heaters, the blower fan turned on in order to pass air through the heat exchanger, thus warming up the cabin, and indirectly warming up the battery. With this process, we planned to eliminate the cabin heater and the block heater. In order to control the process described above, the previous control system was to be modified.

Another point of improvement was re-designing the power supplies for the control system and the new coolant heater. Since the use of the 12 V auxiliary battery was undesirable, a power supply was to be utilized for certain components. This AC-DC converter drew power from the grid, thus not utilizing any battery power. The components of the system requiring 120 V AC were to be powered by a central power distribution point, which was to be fused from the grid for protection purposes. Since the microcontroller uses 10 V for power, a DC-DC converter was designed to reduce the supplied voltage from 12 V to 10 V. All of the circuitry mentioned above was to be designed and installed on a PCB to reduce the size and weight.

To supply power to the heaters, the system was to make use of relays and bipolar junction transistors. When needed, the microcontroller was to output logic high, turn on a transistor, which supplied 12 V to the coil of a relay, thus, closing the circuit on the contact side and
providing the component with power. For the components requiring 120 V AC, the system was to draw the power from the vehicle side of the charger inlet.

The microcontroller was to utilize inputs from sensors to make decisions on whether or not to turn certain components on or off. The microcontroller checked the inputs from the sensors every five seconds, switching the output as needed.

5.3 Modification to Original Design

This project deviates from the original design concept in a few ways. The first major change is the exclusion of the coolant heater due to the lack of the group’s expertise with internal combustion engines. This change required the system to use the previously installed heaters in order to heat the components. The lack of a coolant heater will be further discussed in the alternative solutions section of this report.

Additionally, the power source for the system needed to be changed. The system now draws the AC power from an extension cord, rated for 15 Amps at 120 V. This differs from the original design which was to use the charging station rated for 80 Amps at 240 V. This limited the available power for the components and prevented them from all running at the same time as was originally intended. Using Watt’s Law, the following table compares the available powers from two sources:

\[ P_{rms} \text{ (Watts)} = V_{rms} \text{ (Volts)} \times I_{rms} \text{ (Amps)} \]

<table>
<thead>
<tr>
<th>Power Available from Cord</th>
<th>Power Available from Charging Station</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_{rms} = 120 \text{ V} \times 15 \text{ A} = 1800 \text{ W} )</td>
<td>( P_{rms} = 240 \text{ V} \times 80 \text{ A} = 19200 \text{ W} )</td>
</tr>
</tbody>
</table>

Table 1: Power Availability Comparison
The main reason for not using the charging port was because the charger does not allow current to flow when the battery is fully charged. This contradicted the assumption of using the charger when the battery was fully charged. Another reason was the fact that there are two types of charging stations implemented across the country for this type of the plug. The first type is single phase 120 V line, rated at 15 Amps (same rating as the cord). The second type is a split phase 240 V at 80 Amps. Even if this system did utilize the on board plug and charger for power, it would still be limited to 1800 Watts. For these reasons, our control system was designed in order to not exceed the maximum current and power.

The final major change was the addition of a 250 W water bed heater. This heater was placed on top of the battery in order to directly increase the temperature of the battery housing. For the complete listing and information regarding the heaters, see Appendix D.

Figure 1 graphically shows the finalized design concept. As the legend describes, the red lines are 120 V AC power drawn from an extension cord. The dotted, non-bolded lines represent digital signals that switch the relays, communicate with the temperature sensors, and communicate with the Bluetooth device. The bolded dotted line represents the wireless Bluetooth link. And lastly, the solid lines represent DC power to the components and the user input into the Android app.
Figure 1: Design Concept Block Diagram
6.0 Implementation

6.1 Power Supplies.

The system utilizes an AC-DC power supply, along with a DC-DC linear regulator located on the PCB. A computer power supply is used as an AC-DC converter to provide the high current 12 V source for the blower motor and auxiliary heater along with the power to the PCB. The microcontroller requires a 7-12 V input. The voltage regulator changes the 12 V input to 10.5 V for the microcontroller as a safety precaution. With the use of these power supplies, the use of the 12 V car battery was eliminated.

Figure 2 shows the DC-DC voltage converter. This converter utilizes an LM340T5 linear voltage regulator in an adjustable output regulator configuration as suggested in its datasheet. The components surrounding the regulator were picked based upon availability and the following formula.

\[ V_{out} = 5V + \left( \frac{5V}{R1} + I_Q \right) * R2 \]

**Figure 2: DC-DC Linear Regulator**

For the sake of simplicity of the design, we picked R1 to be 1000 Ω, and \( I_Q \) for this particular regulator was 6 mA. From this we were able to calculate the value for \( R2 \) which was
500 Ω. We chose a 10 kΩ trimmer for $R_2$ and set its value to 500 Ω. The suggested capacitor value in the datasheet was 0.22 µF, however; we used a 0.47 µF capacitor since it was available.

6.2 Relays

This system utilizes two types of relays. The first type is rated 12 V DC at 30 A on the contact side. The second type is rated at 277 V AC at 30 A on the contact side. Both types are rated at 12 V DC on the coil side. The relays used for the 120 V AC components are DPST (Double Pole Single Throw). The relays powering 12 V DC components are SPST (Single Pole Single Throw), the reason for using this type of a relay was mainly the cost. Figure 3 shows the basic switching diagram of the two types of relays. In order to provide power with the DPST relays, we connected $B_1$ to the +120 V AC source, and $B_2$ to the neutral of that source. Pins $A_1$ and $A_2$ were connected to the positive and negative contacts of the heater. In the case of the SPST relays, pin B was connected directly to the +12 V DC source, and pin A was connected to the positive terminal of the heaters.

![Diagram of DPST and SPST relays]

Figure 3: Double Pole Single Throw Relay and Single Pole Single Throw Relay
6.3 Power Circuit Schematics

The following diagram shows a representation of a power circuit that cycles the heaters powered by the 120 V AC supply. As stated above, this circuit utilized DPST relays. The heaters are represented by load resistances. In the diagram 1N4003 acts as the protection diode, and the LED acts as the “ON” indicator. When the microcontroller pin was set to high, Q1 is biased into saturation, thus completing the circuit which turned on the contact of the relay. The diode placed in parallel with the relay is a protection diode for when the relay is turned off. Since the relay is inductive, when the voltage is turned to zero, a current is induced in the relay. This diode provides a path for the current to flow, and decrease to zero safely. This set up was used with the oil pan heater, the battery heater, the block heater, and the cabin heater.

![Power Circuit Diagram](image)

Figure 4: Power Circuit for 120VAC Components

Figure 5 shows the circuitry that powers the 12 V DC components of the system using a SPST relay. The components drew power from a dedicated rail of the AC-DC supply. Control of the relay was similar to the control used in the previous diagram. Like the 120 V circuit, the
diode in parallel with the relay is for protection purposes when the relay turns off. This circuit layout was used to power the blower motor, and both of the PTC heaters.

![Power Circuit Diagram](image)

**Figure 5: Power Circuit for 12 VDC Components**

### 6.4 Circuit Implementation

The above described circuits are mounted into two separate boxes. The first holds a PCB designed by the group, and the Arduino microcontroller. The four layer PCB was designed using PCB123 software and implements all of the circuitry, except for the relays. A separate box is used in order to house the relays. Figure 6 shows the schematic of the PCB circuitry. The circuitry required to run the components was first tested in the lab on a breadboard. There we were able to measure the currents that each component experiences, which matched our calculations. For the detailed information regarding circuit testing and design, please see the Appendix. Also please see the Appendix for the detailed renderings of the printed circuit board’s layers, and the 3D model.
Figure 6: PCB Schematic
As seen in Figure 6, the PCB also holds an “OneWire” temperature sensor represented by Q9. A fan is mounted onto the box, and is turned on whenever the temperature inside the box reaches a certain threshold. The remaining sensors’ power, reference, and the data line are connected through J5. Figure 7 shows all of the components mounted inside the box.

Figure 7: PCB Box

The chassis which houses relays, was custom built in order to hold four DPST and three SPST relays. Below is a picture of the box in the making. It can be seen from the picture that all of the relays are easily interchangeable, in case of a malfunction. Also, the power to the 120 V relays enters from a single line and is isolated with a 15 Amp fuse (not illustrated). From there the line is split in order to feed the four DPST relays.
The use of the vehicle’s blower motor, and the PTC heater assembly was another point of improvement for this project. The PTC heaters, also referred to as the quick heaters, are a set of electrical resistive heaters residing on the HVAC radiator core, also referred to as the heat exchanger. There are three separate PTC heaters, two of them have a resistance of approximately 1 Ω, and one of them has a resistance of approximately 0.5 Ω. This project utilizes the two 1 Ω heaters, each one being controlled by a separate 12 V relay. Once the power is supplied to the heaters, the blower motor also turns on in order to pass the air over the heat exchanger. With this system, the group is able to circulate, and warm up, air in the entire cabin.

Figure 9 shows the circuit diagram of the PTC heaters. This diagram also clearly shows the configuration of the heaters. As stated above, each resistor identified in figure 9 is 1 Ω, thus the resistance between pin A19-2 and pins A84-1 or A84-2 is 0.5 Ω, derived by:
In the 2012 Prius, the blower motor is controlled by the Air Conditioning Amplifier Unit (ACAU) using pulse width modulation (PWM). Figure 9 shows the circuit that was originally installed on the vehicle. As the PTC heaters diagram, this figure shows the pin numbers for every connection. From this figure, it could be seen that the blower motor is always directly connected to the battery and the chassis ground. The speed of the motor is determined by the ACAU based on the user inputs from the dashboard. The ACAU then changes the duty cycle of the PWM wave and adjusts the speed of the motor. When the signal at pin 2 is at DC, the blower motor is off, as the duty cycle decreases, the speed of the motor increases. With this information, we were able to turn the blower motor on and off with our microcontroller. Once the microcontroller was programmed to output a PWM wave at 50% duty cycle when the blower motor relay was on, we ran a test. The test was successful except for one malfunction. When our system was off and the vehicle was on, the blower motor would not function unless the PWM pin on the microcontroller was disconnected. The reason for that was the input resistance of the microcontroller pin was smaller than the input resistance of the blower motor, thus the signal was flowing into the

$$R_{1} || R_{2} = \frac{R_{1}R_{2}}{R_{1} + R_{2}} = \frac{(1)(1)}{1 + 1} = 0.5 \, \Omega$$

Figure 9: PTC Heater Wiring Diagram
microcontroller pin instead of the blower motor. In order to fix the problem, we had to use a separate NPN transistor to break the connection to the microcontroller when the system was off, allowing the car system to run correctly.

Figure 10: Blower Motor Control

The following figure shows where we made connections, and where the transistor was installed. Whenever the blower motor is turned on, pin 2 of the Arduino is set to high, thus turning on the transistor. At the same time, pin 3 sends a PWM to the blower motor signal pin. When the system is off, the transistor is also off, thus breaking the connection to the microcontroller.

Figure 11: Blower Motor Control Circuit.
6.6 Bill of Materials Tables

The parts for the development and building of this project were purchased from two websites. All of the electrical components on the printed circuit board were purchased through Digikey website. Table 1 shows the details for each part used on the board, and the pricing.

<table>
<thead>
<tr>
<th>Row</th>
<th>Quantity</th>
<th>DK Part Number</th>
<th>Part</th>
<th>Manufacturer</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>2N4401-APCT–ND</td>
<td>NPN Transistor</td>
<td>Micro Commercial Components</td>
<td>$ 0.28</td>
<td>$ 2.21</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>1N4003-E3/54GICT–ND</td>
<td>DIODE</td>
<td>Vishay</td>
<td>$ 0.29</td>
<td>$ 2.00</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>67-1755-ND</td>
<td>LED</td>
<td>Lumex Opto/Components Inc.</td>
<td>$ 1.65</td>
<td>$ 11.56</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>1.0KH-ND</td>
<td>RESISTOR</td>
<td>Yageo Corp.</td>
<td>$ 0.06</td>
<td>$ 0.52</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>330H-ND</td>
<td>RESISTOR</td>
<td>Yageo Corp.</td>
<td>$ 0.08</td>
<td>$ 0.58</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>26487-ND</td>
<td>CONNECTOR</td>
<td>TE Connectivity</td>
<td>$ 2.08</td>
<td>$ 6.24</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>SP043-10K-ND</td>
<td>TRIMMER</td>
<td>Vishay</td>
<td>$ 1.52</td>
<td>$ 1.52</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>P5137-ND</td>
<td>CAPACITOR</td>
<td>Panasonic</td>
<td>$ 0.20</td>
<td>$ 0.20</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>LM340T-5.0/NOPB-ND</td>
<td>VOLT REG</td>
<td>Texas Instruments</td>
<td>$ 1.66</td>
<td>$ 1.66</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>WA-T220-101E-ND</td>
<td>HEAT SINK</td>
<td>Ohmite</td>
<td>$ 1.70</td>
<td>$ 1.70</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>259-1355-ND</td>
<td>FAN</td>
<td>Sunon Fans</td>
<td>$ 5.02</td>
<td>$ 5.02</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>377-1546-ND</td>
<td>BOX</td>
<td>Bud Industries</td>
<td>$ 24.30</td>
<td>$ 24.30</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>A26488-ND</td>
<td>CONNECTOR</td>
<td>TE Connectivity</td>
<td>$ 2.83</td>
<td>$ 2.83</td>
</tr>
</tbody>
</table>

Table 2: PCB Bill of Materials

Table 2 presents the bill of materials used in creating the relay box. All of the components were also purchased from Digikey. The table also includes the prices for the relay sockets and the relay terminals.

<table>
<thead>
<tr>
<th>Row</th>
<th>Quantity</th>
<th>DK Part Number</th>
<th>Part</th>
<th>Manufacturer</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>PB679-ND</td>
<td>SPST Relay</td>
<td>Tyco Electronics</td>
<td>$ 3.19</td>
<td>$ 9.57</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>PB486-ND</td>
<td>DPST Relay</td>
<td>TE Connectivity</td>
<td>$ 13.03</td>
<td>$ 52.12</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>377-1546-ND</td>
<td>BOX</td>
<td>Bud Industries</td>
<td>$ 24.30</td>
<td>$ 24.30</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>PB714-ND</td>
<td>RELAY SOCKET</td>
<td>TE Connectivity</td>
<td>$ 6.29</td>
<td>$ 18.87</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>PB786CT-ND</td>
<td>RELAY TERMINAL</td>
<td>TE Connectivity</td>
<td>$ 0.42</td>
<td>$ 15.12</td>
</tr>
</tbody>
</table>

Table 3: Relay Box Bill of Materials
Table 3 shows the remaining components used in our system. These components were purchased from www.sparkfun.com, and were used for the control of our system. Also, this table includes the subtotals for each part category, and the grand total cost of our system, minus the minor materials such as wiring, crimps, and tape.

<table>
<thead>
<tr>
<th>Row</th>
<th>Quantity</th>
<th>Place of Purchase</th>
<th>Part</th>
<th>Description</th>
<th>Distributor Part Number</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td><a href="http://www.sparkfun.com">www.sparkfun.com</a></td>
<td>Microcontroller</td>
<td>Arduino Mega 2560 R3</td>
<td>DEV-11061</td>
<td>$58.95</td>
<td>$58.95</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td><a href="http://www.sparkfun.com">www.sparkfun.com</a></td>
<td>Bluetooth</td>
<td>Bluetooth Mate Silver</td>
<td>WRL-10393</td>
<td>$39.95</td>
<td>$39.95</td>
</tr>
</tbody>
</table>

PCB SUB-TOTAL: $60.34
RELAY BOX SUB-TOTAL: $119.98
DIGITAL SUB-TOTAL: $128.65
TOTAL: $308.97

Table 4: Digital Bill of Materials and Totals

The totals also do not reflect the costs to manufacture the printed circuit board. The main reason behind that was because the prices for PCBs change. If this system were to become mass produced, the costs for the PCB reduce dramatically as the number ordered increases.

6.7 Code Description

Figure 12 shows the flowchart for the code used in this project. It details the thought process of the microcontroller to determine what actions to take while the system is running. This code was developed using the Arduino software and is uploaded on the Arduino Mega 2560 microcontroller. The setup function sets the baud rate of the serial communication bus to 9600 bps and pin modes of all the microcontroller pins connected to all the heaters, box fan and PWM signal to outputs. It additionally locates the number of available sensors on the sensor bus.
Timer1 is the interrupt used and is initialized to occur once every second. In the interrupt subroutine we first check if any message is provided by the user via Bluetooth. This message is
compared to match various possible cases as seen in the flow chart. Next, it updates the clock and maintains it in the 12 hour format. It then checks if the off time is reached for a previously running preheat and turns off preheat. It further checks if a scheduled pre-heat time is reached and turns on preheat. Lastly, it read the temperatures off the six temperature sensors, updating its value every second.

Every five seconds the interrupt subroutine checks to see if the pre heat is set. If set, it turns on the engine block heater and checks conditions for turning on the other heaters. For the oil pan heater, the heater is set to turn on below 20°C and would continue to remain on until it reaches 25°C. Beyond 25°C the heater remains off and continues to be off until it falls below 20°C. Likewise the battery heater, blower motor and the two auxiliary heaters are governed by the battery temperature sensor. These heaters are turned on when the the battery temperature is below 12°C and continues to remain on until it reaches 13°C. Beyond 13°C the heaters turn off and continue to remain off until the battery temperature falls below 12°C. The cabin heater works exactly opposite to the battery heater. It is set to turn on when the battery temperature is above 13°C and remains on until the temperature reaches 12°C. Below 12°C the cabin heater turns off and continues to be off until the temperature of the battery goes above 13°C. Lastly, the PCB box fan heater is set to turn on whenever the temperature of the box is above 0°C.

The off time of a pre-heat is set to two hours from the time it started. All the heaters could not be turned on at the same time as there was not enough power available from the supply. Finally, the optimum temperature values were develop from running initial tests on each heater from which we studied the rates at which each component was being heated and cooled. For source code, see Appendix E.
7.0 Results

In order to properly gauge the effect of the heaters used in the PHEV, control tests were run for each component. In these tests, each heater was run individually for approximately one hour and the temperature of the critical components were measured. After one hour, the component was shut off and the critical component was measured as it cooled down for one hour. The blower motor and the two PTC heaters were considered one component since they turn off and on as one unit. In addition to running the components individually, the car was driven for approximately 30 minutes to observe the temperatures that the critical components reach while in use, so the system can heat to those temperatures. The results for the control tests and car test are given in Appendix B.

In the original implementation of our code, the battery heater turned on if it was below 20 °C and turned off when it reached 23 °C. When the battery heater turned off, the blower motor and PTC heaters also turned off, allowing the cabin heater to turn on. Likewise, the oil pan turned on if it was below 20 °C and would turn off when it reached 25 °C. After seeing what temperatures the car reached during normal operation, the battery temperature limits were changed. In the final version of our code, the battery heater turned on if it was below 12 °C and turned off when it reaches 13 °C, and the oil pan was unchanged, turning on if below 20 °C and turning off when reaching 25 °C.

Through the last test, we were able to show that the finalized code successfully controlled the heaters based on the temperature of the oil pan and the battery. The final test, which was conducted at an external temperature of 2.14 °C ran for one hour. The battery heated to a maximum of 15 °C in about 25 minutes before it started to cool down. The cool down process took an additional 30 minutes before the heater turned back on. During this process, the cabin temperatures fluctuated depending on which components were on at any given time. Over the
course of the test, the front cabin increased from 4.88 °C to 9.63 °C and the back cabin increased from 3.75 °C to 15.56 °C. Figure 13 show the data from the final implementation of our code.

![Final Test Results](image)

**Figure 13: Final Test Results**

### 8.0 Economic Analysis

The average commute time to work in the United States is 11.8 miles. According to the EIA, the average cost of electricity is $0.12/kWh and the average cost of gasoline is $3.43/gallon. The gas mileage of the PHEV Toyota Prius is 50 mpg in the winter and 110 mpg in the summer. This economic analysis will assume a 1 hour preheat for the PHEV.

The 250 W oil pan heater will likely run for the entire hour long pre-heat if the exterior temperature is near freezing or below, leading to a cost of:

\[
60 \text{ min} \cdot \frac{1 \text{ hr}}{60 \text{ min}} \cdot 250 \text{ W} \cdot \frac{0.12 \text{ \$/kWh}}{1000 \text{ W}} = \$0.03
\]

The 400 W engine block heater will run for the entire pre-heat. This leads to a cost of:

\[
60 \text{ min} \cdot \frac{1 \text{ hr}}{60 \text{ min}} \cdot 400 \text{ W} \cdot \frac{0.12 \text{ \$/kWh}}{1000 \text{ W}} = \$0.048
\]
The 250 W battery heater and the 500 W aux heater/blower motor combination is expected to run for 25 minutes, leading to a cost of:

\[ 25 \text{ min} \cdot \frac{1 \text{ hr}}{60 \text{ min}} \cdot 750 \text{ W} \cdot \frac{0.12 \text{ kW}}{kW \text{ hr}} \cdot \frac{kW}{1000W} = 0.0375 \]

The 900 W cabin heater will run whenever the battery heater, blower motor, and aux heaters are off, leading to a cost of:

\[ 35 \text{ min} \cdot \frac{1 \text{ hr}}{60 \text{ min}} \cdot 900 \text{ W} \cdot \frac{0.12 \text{ kW}}{kW \text{ hr}} \cdot \frac{kW}{1000W} = 0.063 \]

All of these components lead to a total cost of $0.18 for a 1 hour preheat. If the system is assumed to be used twice a day for 120 days out of the year, the yearly operational cost is estimated to be $43.20.

Even with pre-heating, it is unrealistic to assume the mileage will reach 110 mpg. If pre-heating can allow the car to reach 80 mpg the cost of the gasoline used is $0.51

\[ 11.8 \text{ mi} \cdot \frac{gal}{80 \text{ mi}} \cdot \frac{3.64}{gal} = 0.54 \]

Compared to the mileage without pre-heating when the car gets 50 mpg the cost of gasoline will be $0.81.

\[ 11.8 \text{ mi} \cdot \frac{gal}{50 \text{ mi}} \cdot \frac{3.64}{gal} = 0.86 \]

For a one way commute to work, this system will save on average $0.32, or $0.64 round trip. This leads to a yearly savings of approximately $76.80 in fuel, and an overall annual savings of $33.60. The system is estimated to cost approximately $424. This gives a breakeven period of 12.6 years before the fuel savings meet the cost of the project. While this project saves fuel consumption, it also provides benefit to the user by pre-heating the cabin, which is difficult to put a monetary value on.
9.0 References


10.0 Appendix A – Circuit Calculations, Simulations, and Measurements

This section details the calculations, simulations, and test measurements for the relay circuit that is implemented on the PCB to control the relays. For calculations, the relay coil is modeled as a 90 Ω resistance. The circuit is shown in figure A1. Hand calculations assumed ideal diodes so the voltage drop across them is negligible. These results were then compared to simulations to determine if they were within the specifications.

When there is no signal from the microcontroller, the transistor Q1 is off and no current flows through the circuit. When the microcontroller gives a 5V signal to the base of the transistor, it turns on, current flows through the relay and turns on the heater it is connected to. The resistance R1 is placed in parallel with the coil resistance to control the current that flows through the LED D2. The diode D1 is placed across the relay to allow the current to flow when the transistor turns off. Since the relay coil is an inductor, when the current is reduced to zero, a negative voltage appears across the coil. D1 allows the current to from this voltage to flow back through the coil until it reduces to zero.

![Figure A1: Relay Circuit](image)
Calculations

Assuming the diode is ideal and has no voltage drop across it, the resistor $R_1$ and the relay $R_{coil}$ produce a current divider between the two parallel resistors.

\[ I_{R1} = I_{LED} = \frac{R_{coil}}{R_1 + R_{coil}} I_S \]

\[ I_S = \frac{V_S}{R_{eq}} \]

The equivalent resistance is determined by the following:

\[ R_{eq} = \frac{R_1 R_{coil}}{R_1 + R_{coil}} = \frac{90(330)}{90 + 330} = 70.7 \, \Omega \]

Knowing the equivalent resistance allows the source current to be determined.

\[ I_S = \frac{V_S}{R_{eq}} = \frac{12 \, V}{70.7 \, \Omega} = 0.170 \, A = 17 \, mA \]

The source current allows the currents flowing through the LED and relay to be calculated.

\[ I_{R1} = I_{LED} = \frac{R_{coil}}{R_1 + R_{coil}} I_S = \frac{90}{330 + 90}(0.170) = 0.0364 \, A = 36.3 \, mA \]

\[ I_{coil} = \frac{R_1}{R_1 + R_{coil}} I_S = \frac{420}{330 + 90}(0.170) = 0.133 \, A = 133 \, mA \]

The LED has a maximum current rating of 100 mA and the relay needs 100 mA to close the other side of the relay. These calculations show that both of the specifications are met; there is not too much current flowing though the LED and there is enough current to close the relay circuit.

All of the source current then flows through the collector, is added to the current the flows into the base, which then goes through the collector to ground.
Simulations

The circuit was constructed and simulated using PSpice. The results of the simulation are shown in figure A2 below.

![Circuit Simulation Results](image)

Figure A2: Circuit Simulation Results

The simulated current though the LED is 33.6 mA and the current through the relay is 130.7 mA. These are close enough to the calculated values assuming an ideal diode, which justifies the assumption. It further confirms the resistor values used in the circuit.

Measurements

The circuit was constructed on a breadboard with discrete components and critical currents and voltages values were measured. The results are given below:

<table>
<thead>
<tr>
<th>Voltages</th>
<th>Measurements</th>
<th>Currents</th>
<th>Measurements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>12.18 V</td>
<td>Relay</td>
<td>106 mA</td>
</tr>
<tr>
<td>Relay</td>
<td>12.01 V</td>
<td>LED</td>
<td>20 mA</td>
</tr>
<tr>
<td>D1</td>
<td>12.05 V</td>
<td>R2</td>
<td>4.14 mA</td>
</tr>
<tr>
<td>R1</td>
<td>10.02 V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LED</td>
<td>2.03 V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q1</td>
<td>108 mV</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table A1: Circuit Measurements
11.0 Appendix B – PCB Design

The next four figures show renderings of the printed circuit board. Figure 16 clearly shows the copper traces on the top layer of the board. It also shows the component markings along with the pads for every hole.

Figure B1: PCB - Top Layer
Figure 17 shows the bottom layer copper traces, along with the pads for each hole. The bottom of the PCB did not have any component designators.

![Figure B2: PCB - Bottom Layer](image)

Figure 18 is the scaled down version of a mechanical drawing of the PCB. The figure shows the overall dimensions of the board, along with the board stackup and a summary of the drill sizes. Since this was a four layer board, the top and the bottom layers are signal layers. The inner layer closer to the top (Inner 1) is the +12 VDC layer. The inner layer closer to the bottom (Inner 2) is the ground plane.
Figure B3: PCB - Mechanical
Figure 19 shows a 3D rendering of the board. This illustration should help visualize and compare the drawings to the actual board.

Figure B4: PCB - 3D Rendering
12.0 Appendix C – Test Results

Control Tests

Before implementing a combination of heaters as a system, each component was tested individually to see how it preheats components on the PHEV. Unfortunately, the tests were not able to be run at the same control temperature due to weather conditions. In addition to the heaters, the car was also driven to determine what temperatures it tried to maintain for the oil pan and battery while in operation. The pre-heat system is designed to pre-heat the components to these operational temperatures.

1.0 Oil Pan Heater

Exterior temperature: 3.56°C (38.4°F)

![Oil Pan - Pre-Heat](image1.png)

![Oil Pan - Cool Down](image2.png)

Figure C1: Oil Pan Control

2.0 Battery Heater

Exterior temperature: 3.56°C (38.4°F)

![Battery - Pre-Heat](image3.png)

![Battery - Cool Down](image4.png)

Figure C2: Battery Control
3.0 Cabin Heater
Exterior temperature: 2.14°C (35.9°F)

Figure C3: Cabin Control

4.0 Aux Heaters and Blower Motor
Exterior temperature: 4.82°C (40.7°F)

Figure C4: Aux Heaters and Blower Motor Control

5.0 Car control test
Exterior temperature: 0.51°C (32.9°F)

Figure C5: Vehicle Warm-Up Control
Combination Tests

Components: Engine block heater, oil pan heater

Exterior temperature: 2.14°C (35.9°F)

Figure C6: Oil Pan and Engine Block Heater

Components: Blower motor, aux heaters, cabin heater

Exterior temperature: 5.7°C (42.3°F)

Figure C7: Blower Motor, Aux Heaters and Cabin Heater
Components: Blower motor, aux heaters, engine block heater, cabin heater

Exterior temperature: 5.68°C (42.2°F)

![Average Cabin Temperature](image)

Figure C8: Blower Motor, Cabin Heater, Block Heater and Aux Heaters

Components: Battery heater, cabin heater, blower motor, aux heaters

Exterior temperature: 9.04°C (48.3°F)

![Cabin Temperature](image) ![Battery Temperature](image)

Figure C9: Battery Heater, Cabin Heater, Blower Motor and Aux Heaters
Temperature Dependent Tests:

Components: All

Code Implementation: First Trial

Exterior temperature: 10.83°C (51.5°F)

---

**Figure C10: Temperature Dependent Test One**
Components: All

Code Implementation: Edited temperature limits

Exterior temperature: 1.95°C (35.5°F)

Figure C11: Temperature Dependent Test Two
Components: All

Code Implementation: Edited temperature limits

Exterior temperature: 2.37°C (36.3°F)

![Graphs of temperature over time for various components: All Temperatures, Oil Pan Temperature, Battery Temperature, Cabin Temperature.](image)

Figure C12: Temperature Dependent Test Three
Components: All

Code Implementation: Final temperature limits

Exterior temperature: 2.37°C (36.3°F)

Figure C13: Temperature Dependent, Final Test
13.0 Appendix D – Heaters

1.0 Cabin Heater

Voltage: 120 V AC

Power: 900 Watts

Manufacturer: Kat’s

Figure D1: Cabin Heater

2.0 Oil Pan Heater

Voltage: 120 V AC

Power: 250 Watts

Manufacturer: Kat’s

Figure D2: Oil Pan Heater
3.0 Engine Block Heater

Voltage: 120 V AC

Power: 400 Watts

Figure D3: Engine Block Heater

4.0 Battery Heater

Voltage: 120 V AC

Power: 300 Watts

Figure D4: Battery Heater
14.0 Appendix E – Code

/* This code was adapted by Ria Pereira, Nick Munyan, and Mike Sokolov. This code is meant to be implemented on a Arduino Mega 2560 microcontroller. Date Finished: 04/16/2013*/

#include <TimerOne.h>
#include <OneWire.h>

const int no_sensors=6; // number of sensors on the serial bus
uint8_t addr_sensors[no_sensors][8]; // a 2-d array storing the addresses of the sensors on the bus
String available_sensors[no_sensors]; // Variable in order to temporarily display sensors found

String addr;
boolean CELSIUS=1;
boolean decreasingOil=0;
boolean decreasingBattery=0;

float gettemperature_sensor(uint8_t *addr); // a function that returns the temperature value in C or F

String schedule[7] = {"00009","00009","00009","00009","00009","00009","00009"]; // a 7 element array for scheduling, storing values for seven days of the week
int day =0;
int hrs=10;
int mins=50;
int secs=30;
int AMPM=0;

int STOPVAR =99;
int OFF_TIME =99;
int OFF_TIME_HOURS =99;

float oilPanTemp = 100; // current temperature of the oil pan
float frontCabinTemp = 100; // current temperature of front cabin
float pcbBoxTemp = 100; // current temperature of the PCB box
float exteriorTemp = 100; // current exterior temperature
float backCabinTemp = 100; // current temperature of back cabin
float batteryTemp = 100; // current temperature of the battery

float optOilPanTempMin = 20; // minimum optimum temperature of the oil pan
float optOilPanTempMax = 25; // maximum optimum temperature of the oil pan
float optPCBboxTemp = 0; // optimum temperature of the PCB box (set very low so the fan is always on)
float optBatteryTempMin = 12; // minimum optimum temperature of the Battery
float optBatteryTempMax = 13; // maximum optimum temperature of the Battery

boolean oilPanHeatRelay=0; // relay turning on the oil pan heater
boolean blockHeaterRelay=0; // relay turning on the block heater
boolean batteryHeaterRelay=0; // relay turning on the battery heater
boolean cabinHeaterRelay=0; // relay turning on the cabin heater
boolean blowerMotorRelay=0; // relay turning on the blower motor
boolean auxRelayOne=0; // relay turning on the auxillary 1
boolean auxRelayTwo=0; // realy turning on the auxillary 2
boolean pcbBoxFan=0; // turing on the PCB box fan

OneWire ds(10); // uC pin connected to the data bus of the serial temperature sensors

void setup(void)
{
    Serial.begin(9600); // baud rate of the serial bus
// Setting ports as OUTPUT to drive relays
pinMode(41, OUTPUT); // uC pin connected to oilPanHeatRelay: Oil pan Heater
pinMode(42, OUTPUT); // uC pin connected to blockHeaderRelay: Block Heater
pinMode(43, OUTPUT); // uC pin connected to batteryHeaterRelay: Battery Heater
pinMode(44, OUTPUT); // uC pin connected to cabinHeaterRelay: Cabin Heater
pinMode(45, OUTPUT); // uC pin connected to blowerMotorRelay: Blower Motor
pinMode(46, OUTPUT); // uC pin connected to auxRelayOne: Auxiliary 1
pinMode(47, OUTPUT); // uC pin connected to auxRelayTwo: Auxiliary 2
pinMode(40, OUTPUT); // uC pin connected to pcbBoxFan: PCB Fan
pinMode(2, OUTPUT); // uC pin giving signal to MOSFET to give PWM

// Finding sensors
int c=0; // counter, counting the number of sensors on the bus
for(int i=0;i<no_sensors;i++)
{
    if(ds.search(addr_sensors[i]))
        c++;
}
if(c>0)
{
    for(int i=0;i<no_sensors;i++)
    {
        addr="";
        for(int j=0;j<8;j++)
        {
            addr=addr+addr_sensors[i][j],HEX;
        }
        available_sensors[i]=addr;
    }
}

Timer1.initialize(1000000); // Timer1 interrupt to occur every one second
Timer1.attachInterrupt(timerIsr);

// Keep the processor Busy
void loop()
{
}

void timerIsr()
{
    // Checks for any data available on Serial and passes into bluetoothMessage
    if(Serial.available() >0)
    {
        int bytes = Serial.available();
        String inData="";
        for(int i=0;i<bytes;i++)
        {
            inData+=(char)Serial.read();
        }
        bluetoothMessage(inData);
    }
    secs++;
if(secs>=60)     //Checks for switching of minutes,hours,AMPM, and days.
    {
        secs=0;
        if(mins<60)
            mins=mins+1;
        else
            mins=0;
    }

if(mins>=60)
    {
        mins=0;
        hrs=hrs+1;
        if(hrs==12)
            {
                if(AMPM==0)
                    AMPM=1;
                else
                    {
                        AMPM=0;
                        if(day<6)
                            day=day+1;
                        else
                            day=0;
                    }
                if(hrs>=13)
                    hrs=1;
            }
    }

if((OFF_TIME == mins)&&(OFF_TIME_HOURS == hrs))   // turns off the preheating system after 2 hours which was it's set OFF_TIME
    {
        oilPanHeatRelay=0;
        blockHeaterRelay=0;
        batteryHeaterRelay=0;
        cabinHeaterRelay=0;
        blowerMotorRelay=0;
        auxRelayOne=0;
        auxRelayTwo=0;
        pcbBoxFan=0;
    }

if(STOPVAR == mins)
    STOPVAR =99;

/*regulates, turns on/off, displays time, checks the scheduling that the user set : every 5 seconds*/

if(secs%5==0)
    {
        // displays the time every 5 seconds

        Serial.print("Time now: ");
        Serial.print(day);
        Serial.print(" ");
Serial.print(hrs);
Serial.print(":");
Serial.print(mins);
Serial.print(":");
Serial.print(secs);
Serial.println(AMPM);

// If the entire preheating system is off it checks if the current time has reached two hour before the scheduled time */

if(!oilPanHeatRelay && STOPVAR == 99)
{
    int dd = day;
    int tempHR = (((int)schedule[dd][0]-48) * 10) + ((int)schedule[dd][1]-48);
    //Serial.println(tempHR);
    int tempMIN = (((int)schedule[dd][2]-48) * 10) + ((int)schedule[dd][3]-48);
    //Serial.println(tempMIN);
    char tempAMPM = schedule[dd][4];
    //Serial.println(tempAMPM);
    if(tempAMPM != '9')
    {
        bool tempAMPMBool = 0;
        if(tempAMPM == '1')
            tempAMPMBool = 1;
        if(tempHR == hrs && tempMIN == mins && tempAMPMBool == AMPM)
        {
            oilPanHeatRelay = 1;
            blockHeaterRelay = 1;
            batteryHeaterRelay = 1;
            cabinHeaterRelay = 1;
            blowerMotorRelay = 1;
            auxRelayOne = 1;
            auxRelayTwo = 1;
            pcbBoxFan = 1;
            //Ensures the heater is not on for more than 2 hours.
            OFF_TIME = mins;
            if(hrs <= 10)
                OFF_TIME_HOURS = hrs+2;
            else if(hrs == 11)
                OFF_TIME_HOURS = 1;
            else if(hrs == 12)
                OFF_TIME_HOURS = 2;
        }
    }
}

for(int x=0;x<no_sensors;x++)// this for loop is used to read the temperature values of each temperature sensor on the serial bus
{
    float temp = gettemperature_sensor(addr_sensors[x]); // reads a temperature value from each sensor
    addr="";
    for(int y=0;y<8;y++)
    {
addr=addr+addr_sensors[x][y].HEX; //this gets the sensor address of the temperature sensor whose temperature reading was just read

// Comparing the address just read from the serial bus to each of its possible addresses
// Then it checks if the temps are within an allowable limit, and stores them in appropriate variables
// If not in allowable range, then it does not update, and uses the previous value

if(addr=="4021232170300180")
{
    if((temp>-100) && (temp<100)) // Checks that Temp is within allowable limit
        oilPanTemp=temp;        // Oil Pan temperature
}
if(addr=="4019713170300211")
{
    if((temp>-100) && (temp<100)) // Checks that Temp is within allowable limit
        frontCabinTemp=temp;       // FRONT cabin temperature
}
if(addr=="40641282440026")
{
    if((temp>-100) && (temp<100)) // Checks that Temp is within allowable limit
        pcbBoxTemp=temp;        // PCB box temperature
}
if(addr=="4078180170300126")
{
    if((temp>-100) && (temp<100)) // Checks that Temp is within allowable limit
        exteriorTemp=temp;        // Exterior temperature
}
if(addr=="409211524400212")
{
    if((temp>-100) && (temp<100)) // Checks that Temp is within allowable limit
        backCabinTemp=temp;     // BACK cabin
}
if(addr=="4022190400192")
{
    if((temp>-100) && (temp<100)) // Checks that Temp is within allowable limit
        batteryTemp=temp;       // Battery temperature
}

//Turns relays on/off based on if the sensor readings have reached optimum temperature: only if the preheat is set
//pcbBoxFan is the box fan

/*Checks if oilPanHeatRelay is set to high, turns it on if under a set temp.(Oil Pan) */
if(oilPanHeatRelay && STOPVAR == 99)
{
    if(oilPanTemp<optOilPanTempMin)
    {
        digitalWrite(41,HIGH);
        decreasingOil = 0;
    }
    else if(oilPanTemp>optOilPanTempMax)
    {
        digitalWrite(41,LOW);
        decreasingOil = 1;
    }
}
else if((oilPanTemp>optOilPanTempMin)&&(oilPanTemp<optOilPanTempMax)&&decreasingOil)
{
    digitalWrite(41,LOW);
    decreasingOil = 1;
}
else
    digitalWrite(41,HIGH);
}
else
digitalWrite(41,LOW);

//Checks if blockHeaterRelay is set to high, turns it on if under a set temp. (Block)
if(blockHeaterRelay && STOPVAR == 99)
    digitalWrite(42,HIGH);
else
    digitalWrite(42,LOW);

//Checks if batteryHeaterRelay is set to high, turns it on if under a set temp. (Battery)
if(batteryHeaterRelay && STOPVAR == 99)
{
    if(batteryTemp<optBatteryTempMin)
    {
        digitalWrite(43,HIGH);
        decreasingBattery = 0;
    }
    else if(batteryTemp>optBatteryTempMax)
    {
        digitalWrite(43,LOW);
        decreasingBattery = 1;
    }
    else if((batteryTemp>optBatteryTempMin)&&(batteryTemp<optBatteryTempMax)&&decreasingBattery)
    {
        digitalWrite(43,LOW);
        decreasingBattery = 1;
    }
    else
        digitalWrite(43,HIGH);
}
else
    digitalWrite(43,LOW);

//Checks if cabinHeaterRelay is set to high, turns it on if under a set temp. (Cabin)
if(cabinHeaterRelay && STOPVAR == 99)
{
    //turns cabin heater on if battery is above set temp
    if(batteryTemp<optBatteryTempMin)
    {
        digitalWrite(44,LOW);
        decreasingBattery = 0;
    }
    else if(batteryTemp>optBatteryTempMax)
    {
        digitalWrite(44,HIGH);
        decreasingBattery = 1;
    }
    else if((batteryTemp>optBatteryTempMin)&&(batteryTemp<optBatteryTempMax)&&decreasingBattery)
    {
        digitalWrite(44,LOW);
        decreasingBattery = 1;
    }
    else
        digitalWrite(44,HIGH);
}
else
    digitalWrite(44,LOW);
digitalWrite(44,HIGH);
decreasingBattery = 1;
}
else
digitalWrite(44,LOW);
}
else
digitalWrite(44,LOW);

//Checks if blowerMotorRelay is set to high, turns it on if under a set temp. (Blower)
if(blowerMotorRelay && STOPVAR == 99)
{
    if(batteryTemp<optBatteryTempMin)
    {
        digitalWrite(45,HIGH);
digitalWrite(2,HIGH);
analogWrite(3, 255*.50);
decreasingBattery = 0;
    }
else if(batteryTemp>optBatteryTempMax)
    {
        digitalWrite(45,LOW);
digitalWrite(2,LOW);
analogWrite(3, 0);
decreasingBattery = 1;
    }
else if((batteryTemp>optBatteryTempMin)&&(batteryTemp<optBatteryTempMax)&&decreasingBattery)
    {
        digitalWrite(45,LOW);
        digitalWrite(2,LOW);
analogWrite(3, 0);
decreasingBattery = 1;
    }
else
    {
        digitalWrite(45,HIGH);
        digitalWrite(2,HIGH);
analogWrite(3, 255*.50);
    }
else
    {
        digitalWrite(45,LOW);
        digitalWrite(2,LOW);
analogWrite(3,0);
    }

//Checks if auxRelayOne is set to high, turns it on if under a set temp. (Aux 1)
if(auxRelayOne && STOPVAR == 99)
{
    if(batteryTemp<optBatteryTempMin)
    {
        digitalWrite(46,HIGH);
decreasingBattery = 0;
    }
else if(batteryTemp>optBatteryTempMax)
    {
digitalWrite(46,LOW);
decreasingBattery = 1;
}
else if((batteryTemp>optBatteryTempMin)&&(batteryTemp<optBatteryTempMax)&&decreasingBattery)
{
    digitalWrite(46,LOW);
decreasingBattery = 1;
} else
digitalWrite(46,HIGH);
} else
digitalWrite(46,LOW);

// Checks if auxRelayTwo is set to high, turns it on if under a set temp. (Aux 2)
if(auxRelayTwo && STOPVAR == 99)
{
    if(batteryTemp<optBatteryTempMin)
    {
        digitalWrite(47,HIGH);
decreasingBattery = 0;
    }
    else if(batteryTemp>optBatteryTempMax)
    {
        digitalWrite(47,LOW);
decreasingBattery = 1;
    } else if((batteryTemp>optBatteryTempMin)&&(batteryTemp<optBatteryTempMax)&&decreasingBattery)
{
        digitalWrite(47,LOW);
decreasingBattery = 1;
} else
digitalWrite(47,HIGH);
} else
digitalWrite(47,LOW);

// Checks if pcbBoxFan is set to high, turns it on if under a set temp. (PCB Fan)
if(pcbBoxFan && STOPVAR == 99)
{
    // If pcb too hot, turn fan on
    if(pcbBoxTemp>optPCBboxTemp)
        digitalWrite(40,HIGH);
    else
        digitalWrite(40,LOW);
} else
digitalWrite(40,LOW);

// Subroutine that returns the temperature from the sensor's address
float gettemperature_sensor(uint8_t *addr)
{
    byte data[12];
    int x;
ds.reset();
ds.select(addr);
ds.write(0x44,1);

ds.reset();
ds.select(addr);
ds.write(0xBE,1);

for(x=0;x<9;x++)
    data[x] = ds.read();

int tr = data[0];
if(data[1] > 0x80) {
    tr = !tr+1;
    tr = tr*-1;
}
int cpc = data[7];
int cr = data[6];

tr = tr>>1;
float temp = ((data[1] << 8) + data[0] ) * 0.0625;
if(!CELSIUS)
    temp = (temp * 9) / 5 + 32;
return temp;

/* Subroutine that reads the string message and outputs various info*/

int bluetoothMessage(String message)
{
    // 'X' Prints the current time and the current status of the relays
    if(message[0] == 'X')
    {
        Serial.println("Status:");
        Serial.print("Time: ");
        Serial.print(day);
        Serial.print(" ");
        Serial.print(hrs);
        Serial.print(" ");
        Serial.print(mins);
        Serial.print(" ");
        Serial.print(secs);
        Serial.println(" Relays:");
        Serial.print(oilPanHeatRelay);
        Serial.print(blockHeaterRelay);
        Serial.print(batteryHeaterRelay);
        Serial.print(cabinHeaterRelay);
        Serial.print(blowerMotorRelay);
        Serial.print(auxRelayOne);
        Serial.print(auxRelayTwo);
        Serial.println(pcbBoxFan);
    }

    // 'S' Makes a schedule Edit
    if(message[0] == 'S')
{  //If scheduleAMPM = 9, that time is not set.
    //Subtracts 48 because it receives the ASCII value. ASCII for 0 is 48, 1 is 49
    //and so on.
    int scheduleHr = (int)message[5] - 48;
    int scheduleTenMin = (int)message[7] - 48;
    int scheduleMin = (int)message[8] - 48;
    //Gets hour bit wise. For 11 it is 1 + (1 * 10) = 11
    scheduleHr = scheduleHr + (scheduleTenHr * 10);
    //Gets minute bit wise. For 43 it is 3 + (4 * 10) = 43
    scheduleMin = scheduleMin + (scheduleTenMin * 10);
    char scheduleAMPM = '0';
    if(scheduleHr > 12)
    {
        scheduleHr -= 12;
        scheduleAMPM = '1';
    }
    String hour = (String)scheduleHr;
    STOPVAR = 99;
    //If disabled
    if(scheduleHr == 24 && scheduleMin == 0)
        scheduleAMPM = '9';
    //Update the new value for scheduleDay
    if(scheduleHr < 10)
    {
        schedule[scheduleDay][0] = '0';
        schedule[scheduleDay][1] = hour[0];
    }
    else
    {
        schedule[scheduleDay][0] = hour[0];
        schedule[scheduleDay][1] = hour[1];
    }
    schedule[scheduleDay][2] = message[7];
    schedule[scheduleDay][3] = message[8];
    schedule[scheduleDay][4] = scheduleAMPM;
    Serial.print("S ");
    Serial.print(scheduleDay);
    Serial.print(" ");
    Serial.print(scheduleHr);
    Serial.print(" ");
    Serial.println(scheduleMin);
// 'U' enables one to make an update in time
if(message[0] == 'U' || message[1] == 'U')
{
    int messageCount = 2;
    if(message[1] == 'U')
        messageCount = 3;

    int counter = 0;
    String tempString="";
    boolean endLine = 0;
    String data[9];

    while(!endLine)
    {
        if(message[messageCount] >= '0' & message[messageCount] <= '9')
            tempString += message[messageCount];
        if (message[messageCount] == ' ')
            // Handle delimiter
            data[counter] = tempString;
            data[counter] += "\n";
            tempString = "";
            counter = counter + 1;
    }
    if(message[messageCount] == '\n')
    {
        // end of line
        endLine = 1;
    }
    messageCount++;
}

day = data[4].toInt()-1;
hrs = data[5].toInt();
mins =data[6].toInt();
secs =data[7].toInt();

Serial.print("X Set for: ");
Serial.print(day);
Serial.print(" ");
Serial.print(hrs);
Serial.print(" ");
Serial.print(mins);
Serial.print(" ");
Serial.println(secs);

AMPM = 0;
if(hrs > 12)
{
    hrs -= 12;
    AMPM = 1;
}
//C 0' returns the temperature in celsius or else it remains in fahrenheit
if(message[0] == 'C')
{
    if(message[2] == '0')
        CELSIUS = 1;
    else
        CELSIUS = 0;
}

//Turns off preheating system
if(message[0] == 'A')
{
    oilPanHeatRelay = 0;
    blockHeaterRelay = 0;
    batteryHeaterRelay = 0;
    cabinHeaterRelay = 0;
    blowerMotorRelay = 0;
    auxRelayOne = 0;
    auxRelayTwo = 0;
    pcbBoxFan = 0;
    Serial.println("A");
}

//P' Message starts preheat
if(message[0] == 'P')
{
    Serial.print("SET RELAYS");
    oilPanHeatRelay = 1;
    blockHeaterRelay = 1;
    batteryHeaterRelay = 1;
    cabinHeaterRelay = 1;
    blowerMotorRelay = 1;
    auxRelayOne = 1;
    auxRelayTwo = 1;
    pcbBoxFan = 1;
    STOPVAR=99;
    OFF_TIME = mins;
    if(hrs <= 10)
        OFF_TIME_HOURS += 2;
    else if(hrs == 11)
        OFF_TIME_HOURS = 1;
    else if(hrs == 12)
        OFF_TIME_HOURS = 2;
    Serial.println("P");
}

//D' RequestS from the Android device for the current temperature.
if(message[0] == 'D')
{
    if(CELSIUS)
    {
        Serial.println("TEMPERATURES IN CELCIUS ");
        Serial.print(oilPanTemp);
        Serial.print(" ");
    }
Serial.print(frontCabinTemp);
Serial.print(" ");
Serial.print(pcbBoxTemp);
Serial.print(" ");
Serial.print(exteriorTemp);
Serial.print(" ");
Serial.print(backCabinTemp);
Serial.print(" ");
Serial.println(batteryTemp);
}
else
{
    Serial.println("TEMPERATURES IN FAHRENHEIT ");
    Serial.print(oilPanTemp);
    Serial.print(" ");
    Serial.print(frontCabinTemp);
    Serial.print(" ");
    Serial.print(pcbBoxTemp);
    Serial.print(" ");
    Serial.print(exteriorTemp);
    Serial.println(TEMPERATURES IN FAHRENHEIT ");
} // end of blueToothMessage