Chilly Dog

Battery Powered Car Air Conditioner

Martin Rennaker

Grant Franklin

Yarentzy Magallanes

**Subsystem Reports**

REVISION – 1

30 April 2022

Subsystem Reports

for

Chilly Dog

Battery Powered Car Air Conditioning

Team <31>

Approved by:

Martin Rennaker 04/30/2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Leader Date

Kevin Nowka 04/30/2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Date

Sambandh Dhal 02/09/2022

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

T/A Date

**Change Record**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rev.** | **Date** | **Originator** | **Approvals** | **Description** |
| **-** | 4/30/2022 | Grant Franklin | Skyelar Head | Draft Release |

Table of Contents

[Table of Contents 3](#_Toc102252065)

[List of Tables 5](#_Toc102252066)

[List of Figures 6](#_Toc102252067)

[1. Introduction 7](#_Toc102252068)

[2. Power Supply Subsystem Report 8](#_Toc102252069)

[2.1. Power Subsystem Block Diagram 8](#_Toc102252070)

[2.2. Buck Converters 9](#_Toc102252071)

[2.2.1. MCU Converter 9](#_Toc102252072)

[2.2.2. Modem Converter 9](#_Toc102252073)

[2.3. Inverter/VAC 9](#_Toc102252074)

[2.4. Solar Panel Charging 9](#_Toc102252075)

[2.5. Solar Switching Circuit 9](#_Toc102252076)

[2.6. Subsystem Validation 10](#_Toc102252077)

[2.6.1. MCU Converter 10](#_Toc102252078)

[2.6.2. Modem Converter 12](#_Toc102252079)

[2.6.3. Inverter/VAC 14](#_Toc102252080)

[2.6.4. Solar Panel Charging 14](#_Toc102252081)

[2.7. Subsystem Conclusion 15](#_Toc102252082)

[3. Microcontroller Subsystem 16](#_Toc102252083)

[3.1 Microcontroller 16](#_Toc102252084)

[3.2 Temperature Sensor 17](#_Toc102252085)

[3.2.1 Communication Method 17](#_Toc102252086)

[3.2.1 Accuracy 18](#_Toc102252087)

[3.2.2 Validation 18](#_Toc102252088)

[3.3 UART Serial Protocol 19](#_Toc102252089)

[3.3.1 Validation 19](#_Toc102252090)

[3.4 Relay Control 19](#_Toc102252091)

[3.4.1 Validation 20](#_Toc102252092)

[3.5 AC Input 20](#_Toc102252093)

[3.5.1 Validation 20](#_Toc102252094)

[3.6 Microcontroller Subsystem Summary 21](#_Toc102252095)

[4 Application and Modem Subsystem 22](#_Toc102252096)

[4.1 Android App 22](#_Toc102252097)

[4.2 Modem 24](#_Toc102252098)

[4.3 Validation 24](#_Toc102252099)

[4.3.1 Android App 24](#_Toc102252100)

[4.3.2 Modem 25](#_Toc102252101)

[4.4 Application and Modem System Summary 26](#_Toc102252102)

[5. Appendix - A 27](#_Toc102252103)

List of Tables

Table 1. Power Consumption of DHT22

Table 2. DHT22 Sensor Comparison

List of Figures

Figure 1. Power Block Diagram

Figure 2. Switching Circuit Simulation

Figure 3. MCU Multimeter Output

Figure 4. Load Regulation Test Results

Figure 5. Line Regulation Test Results

Figure 6. Modem Multimeter Output

Figure 7. Load Regulation Test Results

Figure 8. Line Regulation Test Results

Figure 9. Solar Panel Output Voltage to Controller

Figure 10. Current Battery Voltage

Figure 11. Inputs and Outputs of the Microcontroller

Figure 12. MCU DHT22 Signal Exchange

Figure 13. UART Print

Figure 14. Relay Circuit

Figure 15. App and Modem Block Diagram

Figure 16. App Layout

Figure 17. App Sending Codes

Figure 18. AT Commands Used

# Introduction

The Chilly Dog will create a safe, cool environment for pets inside a car during the hot Texas summer. The system will take in the user’s inputted data on the application and send that data to the MCU to be interpreted. The MCU will determine whether or not the car is currently cool enough and if not, it will send a command to the VAC system to cool the car. The system will be powered by a lead acid battery and stepped up and down to the various loads using DC-DC converters and an inverter.

# Power Supply Subsystem Report

The power supply subsystem consists of a 12V lead acid battery that is charged with a 100W solar panel, after going through a solar charge controller. The battery powers two buck converters that step down voltage to an MCU and modem, as well as an inverter that steps up the voltage to a portable VAC system.

## Power Subsystem Block Diagram

Diagram, schematic

Description automatically generated

Figure 1. Power Block Diagram

## Buck Converters

### MCU Converter

The DC-DC Buck converter leading to the MCU successfully stepped down the 12V of the battery to ~3.32V at <150mA. The circuit consists of an isolated Buck converter with an inductor and 4 capacitors that regulate the voltage coming in and out of it. The converter did not have a sound data sheet and therefore no simulation was able to be made due to difficulties in finding suitable models. Due to this, validation on the breadboard and Perfboard was done to confirm correct operation of the circuit. It passed and gave valid outputs as seen in section 3.6.1. Over the summer, research and simulations into a non-isolated DC-DC converter will be done so the user can have more flexibility in the output voltage and current going to the MCU.

### Modem Converter

The DC-DC Buck converter leading to the modem successfully stepped down the 12V of the battery to ~5V at <2A. The circuit consists of an isolated Buck converter with all the voltage regulation required for the circuit already inside of it. The converter did not have a sound data sheet and therefore no simulation was able to be made due to difficulties in finding suitable models. Due to this, validation on the Perfboard was done to confirm correct operation of the circuit. It passed and gave valid outputs as seen in section 3.6.2. Over the summer, research and simulations into a non-isolated DC-DC converter will be done so the user can have more flexibility in the output voltage and current going to the Modem.

## Inverter/VAC

The lead acid battery is connected to the inverter using an o-ring connector with a fuse. The portable VAC system plugs into the female wall outlet port on the inverter and receives the stepped up 115V at ~7.2A.

## Solar Panel Charging

The solar panel charging system consists of a 100W solar panel, a solar charge controller, and the battery that gets charged. The solar panel provides a nominal voltage and current of 17.9V and 5.72A respectively, however it can provide a maximum voltage and current of 21.6V and 6.24A. The solar charge controller regulates the voltage to ~14V to give enough power to charge the battery, without overloading it.

## Solar Switching Circuit

Another part of the power supply that was not able to be validated was the solar switching circuit. This circuit was created to determine whether the solar panel was creating enough output to charge the battery, and if not, allow the 12V car outlet to provide charging. This circuit can be seen below.

Diagram, schematic

Description automatically generated

Figure 2. Switching Circuit Simulation

## Subsystem Validation

### MCU Converter

To validate the MCU DC-DC converter, an initial test for operation, as well as line and load regulation tests were performed on the circuit. The circuit took in 12V from the bench power supply to simulate the battery and outputted 3.32V as seen in Figure 3.

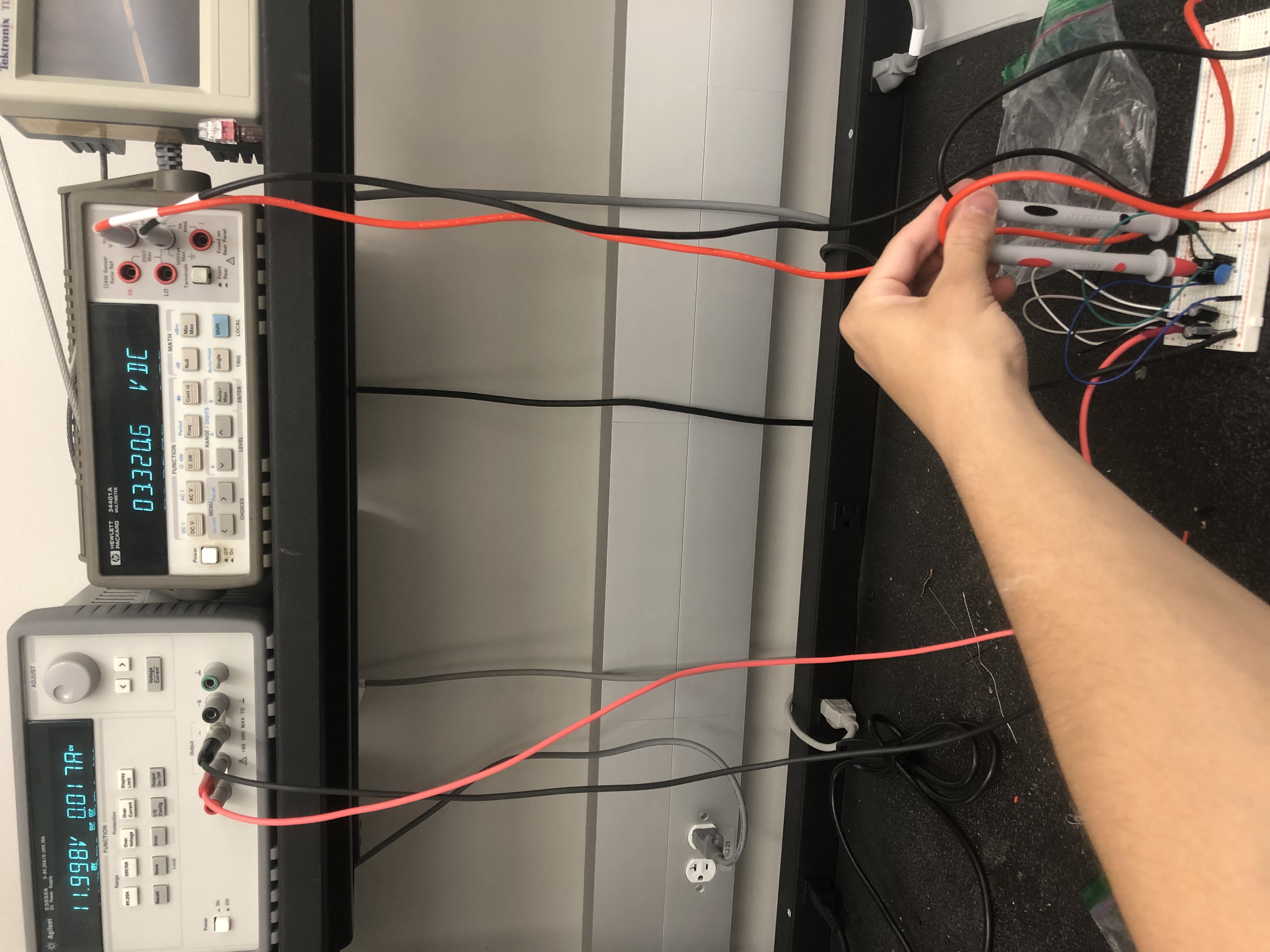


Figure 3. MCU Multimeter Output

The load regulation test was conducted by hooking up an e-load to the output nodes of the circuit and measuring the change in load voltage after changing the current draw from 0A to 250mA (Figure 3). The line regulation test stepped up the input voltage from 11.4V to 12.8V, measuring the change in the load voltage (Figure 4). The converter had an acceptable 7.03% load regulation and very good line regulation of 0.003%.

Chart, scatter chart

Description automatically generated

Figure 4. Load Regulation Test Results

Chart

Description automatically generated with medium confidence

Figure 5. Line Regulation Test Results

### Modem Converter

To validate the Modem DC-DC converter, an initial test for operation, as well as line and load regulation tests were performed on the circuit. The circuit took in 12V from the bench power supply to simulate the battery and outputted 5V as seen in Figure 5.



Figure 6. Modem Multimeter Output

The load regulation test was conducted by hooking up an e-load to the output nodes of the circuit and measuring the change in load voltage after changing the current draw from 0A to 2A (Figure 6). The line regulation test stepped up the input voltage from 11.4V to 12.8V, measuring the change in the load voltage (Figure 7). The converter had an acceptable 2.14% load regulation and very good line regulation of 0.006%.

Chart, scatter chart

Description automatically generated

Figure 7. Load Regulation Test Results

Table

Description automatically generated with medium confidence

Figure 8. Line Regulation Test Results

### Inverter/VAC

The inverter/VAC portion of the power supply subsystem was tested for functionality, since all the associated products were purchased, not made. The battery connected to the inverter through an O-ring with a fuse, and the VAC plugged into the inverter’s wall socket. The device turned on and was able to operate when the temperature was changed from the minimum to the maximum value.

### Solar Panel Charging

The charging of the battery is handled by the 100W solar panel. Since the panel outputs more than 14V needed for charging, the voltage is regulated by the solar charge controller. This can be seen in Figures 9 and 10, where the solar panel is outputting 19.1V and the battery is charging from the output of the charge controller while at a voltage of 12.6V.

A picture containing text, indoor

Description automatically generated

Figure 9. Solar Panel Output Voltage to Controller

A picture containing text, indoor

Description automatically generated

Figure 10. Current Battery Voltage

## Subsystem Conclusion

Each part of the power supply system operated as designed. The DC-DC converters outputted their respective voltages at a current regulated for their loads. The inverter successfully takes the DC battery voltage and changes it to an AC voltage for the VAC to use and operate within the range of operating temperatures. The solar panel gives a high enough voltage for the controller to step down and charge the battery. Moving forward, changes to the DC-DC converters will be made for more flexibility with loads, as well as implementation of a switching input system to provide a 12V car outlet as a secondary charging source.

# 3. Microcontroller Subsystem

The microcontroller subsystem serves as the control center of the Chilly Dog. It is responsible for reading temperature sensors and interpreting the battery life of the system outputting the data to the user’s application. It will receive a temperature set point by the user and control the on/off status of the VAC system. It will also alert the user when the temperature of the cabin has varied far from what the user sent. The Chilly Dog as a whole was created to maintain safe conditions for a pet inside a vehicle. It is critical that these sensors and responses are executed in a timely manner. For this reason, the functionality of each part of the microcontroller subsystem was thoroughly validated to meet our requirements.

Diagram

Description automatically generated

Figure 11. Inputs and Outputs of the Microcontroller

## Microcontroller

The selected microcontroller for the Chilly Dog was the PIC24FJ128GA204. The PIC24FJ is a flexible, cost-effective, low-power microcontroller equipped with innovative peripherals and plenty of flash program memory. Some of the key characteristics that led to the selection of this MCU include:

* 44 I/O Pins
* UART Serial Communication
* 12 Bit Analog Input Channels for A/D Conversion
* 5 Timers
* Mplab X IDE
* Programmed in C code
* Mplab Code Configurator
* Sufficient and Extensive Documentation

## Temperature Sensor

The temperature sensor used for the data collection of the internal cabin, and outdoor conditions is the DHT22. Due to the limited charge of the battery and our goal of having the device cool for a minimum of 2 hours, power is a precious resource. The selection of this sensor revolved around its low power consumption as summarized in the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Condition | Minimum | Typical | Maximum |
| Power Supply | DC | 3.3 V | 5 V | 6 V |
| Current Supply | Measuring | 1 mA | 1.3 mA | 1.5 mA |
| Current Supply | Stand-by | 40 uA | Null | 50 uA |

Table 1. Power Consumption of DHT22

### 3.2.1 Communication Method

A one wire bus is used for communication between the MCU and the DHT22. To begin data transmission, the sensor first waits for a low pulse of at least 1ms from the MCU followed by a high pulse for 40us. Once the signal is received the sensor will respond with a low pulse of 80us and a high pulse of 80us as illustrated below.

Diagram

Description automatically generated

Figure 12. MCU DHT22 Signal Exchange

Upon established connection, the data is sent through the bus in 40 bits. The right most 16 bits are allocated for the relative humidity data, followed by another 16 bits for the temperature data and the last 8 bits being the check sum. An example of data transmission can be seen below.

0000 0010 1000 1100 0000 0001 0101 1111 1110 1110

------------------------ ------------------------ ------------

16 bits humidity 16 bits temperature 8 bits check sum

*Humidity = 65.2%*

Binary to Decimal

0000 0010 1000 1100 -> 652

Divide by 10

652/10 = 65.2

*Temperature = 35.1 ℃*

Binary to Decimal

0000 0001 0101 1111 - > 351

Divide by 10

351/10 = 35.1

*Check Sum = 1110 1110*

Humidity 0000 0010 1000 1100

+ +

Temperature 0000 0001 0101 1111

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Check Sum 1110 1110

Once the 40 bits of data are sent, the sensor will again wait for the pulses of the MCU to take another reading.

### Accuracy

The DHT22 temperature and humidity sensor is capable of reading data every two seconds with an accuracy of +- 0.5 degrees Celsius and a resolution sensitivity of 0.1 degrees Celsius with a long transmission distance of up to 100 meters. This accuracy was critical for the scope of the Chilly Dog as we will provide the most accurate up to date temperature data possible to the user and alert them in real time when conditions fall below the desired setpoint temperature.

### Validation

The sensor was implemented into the MCU and tested thoroughly with the use of redundancy. Two DHT22 temperature sensors were placed side by side and readings were collected. The results of this cross examination can be seen below.

|  |  |
| --- | --- |
| **DHT22 Sensor #1** | **DHT22 Sensor #2** |
| 73.6 °F | 73.6 °F |
| 73.5 °F | 73.5 °F |
| 73.5 °F | 73.5 °F |
| 73.4 °F | 73.4 °F |
| 73.4 °F | 73.5 °F |
| 73.2 °F | 73.2 °F |
| 73.2 °F | 73.2 °F |

Table 2. DHT22 Sensor Comparison

Warm air was then blown on the sensors and an increase was detected and recorded.

Validation of the speed of data transmission was also conducted by redirecting the output signal of the sensor to an oscilloscope and taking measurements between data transmission beginning and end.

## UART Serial Protocol

The microcontroller of the Chilly Dog will be responsible for transmitting and receiving data to and from the user application. The means of this communication will be done through the serial communication protocol UART or Universal Asynchronous Receiver/Transmitter.

### Validation

UART has been used extensively within the microcontroller subsystem. The Chilly Dog MCU can receive and send data between MCU and MCU, and between MCU and console. One of the most useful purposes being redirecting print function to the console. Using this feature, debugging of code was simplified and used for incremental coding.

Graphical user interface

Description automatically generated

Figure 13. UART Print

## Relay Control

The MCU of the Chilly Dog will turn on and off the VAC system according to the cabin temperature and the temperature set by the user. This will be done using a relay.

### Diagram, schematic Description automatically generatedValidation

The validation of the MCUs responsibility of controlling the on/off status of the VAC system was completed using a Songle 5V electromagnetic relay. The relay was implemented using the circuit below. (Note: The PICFJ24 was not supported by this simulation software so an Arduino Nano is shown. This, however, was implemented using the PIC24FJ and its respective I/O pins.)

Figure 14. Relay Circuit

This circuit consists of the relay, an NPN transistor, two resistors, an LED, and a diode connected across the electromagnetic coil to prevent against a voltage spike or a backward flow of current.

## AC Input

Another function of the Chilly Dogs MCU is outputting the devices battery percentage. The MCU will be receiving a scaled down voltage from the battery and expected to interpret that voltage into a percentage of battery life.

### Validation

The scope of validation for this function in 403 revolved around being able to configure the MCU to receive an AC voltage and output it. This was completed using a potentiometer and a flashing LED. The potentiometer varied the resistance and therefore the voltage seen by the input pin in the arrangement of a voltage divider. The position of the potentiometer controlled the speed at which the LED toggled.

## Microcontroller Subsystem Summary

The PIC24FJ128GA204 fits and exceeds the needs of the Chilly Dog. It has been configured to read and process temperature data, communicate from MCU to MCU and from MCU to console, turn on or off a load, and receive and interpret AC voltages. Looking forward to ECEN 404, a relay capable of controlling a higher current will be necessary to power the VAC system, code will be written to manage alert cases, the PIC will communicate via UART with the application subsystem and the AC voltage will be received from the power subsystem in order to output battery percentage.

# Application and Modem Subsystem

Diagram

Description automatically generatedThe Application and Modem Subsystem has the goal of keeping the user informed that their pet is alive and well in the car. The app also serves as the method of controlling the set temperature of the VAC unit allowing the user to control the speed at which the car is cooled down. The modem allows the Chilly Dog to rely on cellular data which while not available all the time is more reliable than Wi-Fi or Bluetooth. This makes the Chilly Dog’s range theoretically infinite in the US if both the device with the app and The Chilly Dog are both within range of a cell tower. The app is also for error detection and has notifications to alert the user of malfunctions within The Chilly Dog, prolonged temperature discrepancies, or low battery. The following sections will go more in depth into the construction of each subsystem and then how each were validated.

Figure 15 App and Modem Block Diagram

## Android App

The application was constructed using Android Studio meant for an Android device, this was chosen due to IOS being more difficult to work with and Android allowing developers more ready access to features like texting. At its core the app is a SMS platform that is made to send and accept text messages from a specific user defined phone number, this being The Chilly Dog.

After install on first startup the app will ask the user for permission to send and receive text messages and will store this permission. The app will open to a not filled in home screen and the device status will read “Device Not Connected”. The user will have to navigate to the Add Device page and enter The Chilly Dog’s phone number. This will prompt the app to send a SMS to The Chilly Dog that reads “test” from here the app will wait until it receives a response from the phone number it messaged. If it does the connection status will be updated, this is handled internally as a Boolean. Once the connection is established the app will wait to receive messages from the connected device formatted with the prefix corresponding to the prefixes seen if figure 15, these will tell the app which section to put the received data. The Add a Device button has been made so it will only accept 10-digit entries which is the length of a phone number, anything else and it will simply inform the user that the entered value is invalid. Once a device is connected if the user attempts to connect another device the user will be told that a device is already connected and if they wish to add a new one, they must press the button again to confirm. In the future the app will be able to hold multiple devices.

The home screen on the app was designed so the most important information for the user is the most visible on startup, that is the internal car temperature, the outside temperature, and the battery remaining. These sections will be filled in by the texts received from the connected device and will update both when the app is in the foreground and background. At the bottom of the home page is the editable text box for the user to enter a desired set temperature for the VAC system. This box will only accept numbers and if the number is out of the range 40 – 90 the user will be asked to confirm that they wish to send this temperature. The temperature range is to help prevent the user from accidentally entering an extreme temperature, however they may wish to send that temperature, so they simply must confirm.

The last main feature of the app are the notifications, the app must notify the user of malfunction or battery issue to keep the user’s pet safe. The notifications page is simply a page full of toggles to different notifications. Currently the device only notifies if the battery reaches a certain percentage, and the corresponding notification is turned on. In the future however there will be more notifications this was more of a proof of concept to show that the app can send notifications. These notifications work both in when the app is in the foreground and background, will only be received if the condition is met and the toggle is on, and when tapped will navigate the user back to the app.

Graphical user interface, application

Description automatically generated

Table

Description automatically generatedFigure 16 App Screens, left to right: Home, Menu, Add Device, and Notifications

Figure 17 Message Prefixes

## Modem

The Modem is the middleman between the phone app and the microcontroller, this means that it receives a text from the app indicating the user’s desired temperature and sends texts to the app indicating the car’s internal temperature, the temperature outside the car, and the current battery remaining in The Chilly Dog. This modem will communicate to the microcontroller via UART and currently has been communicating to a laptop via built in UART to USB through PuTTY. The serial communication transmits AT commands which are commands made for COMS and can be used to configure the modem and send and receive text messages.

Graphical user interface, text, application, table, Excel

Description automatically generatedThe modem was going to communicate with a Raspberry Pi which was then going to communicate with the microcontroller, however the Pi wasn’t cooperating, and it is more efficient if in the final design the modem simply communicates directly with the microcontroller. On startup the microcontroller will send the commands “AT” and “AT+CMFG” which establish the initial handshake and set the SMS format to text as opposed to hex code, the other at commands used are shown in figure 17. The device will then wait until the “test” SMS is received and then it will send the updated in temperature, out temperature, and battery percentage. From here it will sit idle waiting for either an updated temperature for the VAC from the app or in, out temperatures, and battery percentage to send to the app.

Figure 18 AT commands used to communicate with the modem

## Validation

Testing these systems was interesting as they are both completely on the computer or on a mobile device meaning there aren’t many numeric results and instead differ. In Appendix A more detailed results will be shown this section will go over all the tests and how they were conducted, where the appendix will have the pictures as proof and those will simply have flowchart style explanations of what is happening.

### Android App

The testing of the app was the more extensive of the two as it has the most failure states. First the permissions request was verified, this was done by simply uninstalling and reinstalling the app and verifying that the app requested if it could have permission to send and receive text messages. The second conducted test was over the receiving of the temperature data and battery data. To do this, a device was connected to the app, in this case it was simply a cell phone, and from that device the text “itemp:78” was sent to the phone with the app. Then the app was checked to see if the inside temperature then displayed 78 degrees. These messages do not need to be checked if they are something other than a temperature as they will come from The Chilly Dog and therefor will not be anything other than what they are supposed to be. This test was repeated for the other two pieces of data that will be received from The Chilly Dog. The next test was the sending of a temperature to The Chilly Dog, to test this a temperature was entered into the box and then the send button was pressed. Then it was checked that the phone had sent a text to the connected device’s phone number that read “utemp:[temperature entered]”. This step also included some validation of making sure that the value the user entered was a valid number within the range 40 – 90. To test this, first a word was entered and then verified that the app responded by not sending the message and informing the user that it is an “Invalid Input”. Second a number out of range was sent and it was verified that the app responded by not sending the message and asking the user to confirm the entered temperature. Then the button was pressed again, and it was verified that it sent the out of range temperature because the user confirmed it.

The next set of tests conducted were on the notifications. To test these the 25% battery notification was toggled on, and the phone was sent the message “bat:25” which tells the app that the battery is at 25%. Then it was verified that this triggered the app to send a notification to the phone. This same test was conducted again however this time google chrome was opened to simulate the app being in the background, and the same result was verified, when send that the battery was at 25% the app would notify the phone. Both tests were conducted again although this time the 25% notification was toggled off and it was verified that both when in foreground and background when sent the 25% battery message the app did not notify the phone.

Finally, the device connection was tested this began with entering a phone number and verifying that the message “Connection Attempted” appeared and the attempted device received the “test” text message. Then from the device the temperature and battery information were sent to the phone, and it was verified that as soon as the first message was received the connection status was updated. Then the remove a device functionality was tested by pressing the remove button and verifying that the connection status was updated, and that the device information was removed. The text box then needed to be tested as it needs to only accept 10-digit numbers, this box was tested with words, improper length numbers, and floats and it was verified that each time the app returned “Invalid Input”. Finally, the app was connected to a device and another device was attempted to be connected to and it was verified that the app responded by informing the user that there already was a device connected and to confirm that they wanted to connect a new device by pressing the button again. Upon the press of the button again it was verified that the device attempted to establish a new connection with the new device.

### Modem

The testing of the modem had fewer components than the phone app as all that needed to be verified was the power consumption, and the send and receive SMS capability. To test the power requirement both micro-USB ports on the modem were utilized, those being the micro-USB power port and the micro-USB to UART port. The UART port was connected to the computer and interfaced with using PuTTY, while the power port was connected to a 5V 1A brick and then to the wall socket. The rest of the tests were then conducted using this setup to verify that the power requirements were met, and the modem was functioning properly using the desired power setup. The next test that was conducted was verifying the send text functionality. To do this the commands in order that were send were “at (enter) at+cmfg=1 (enter) at+cmgs=”+17372106468” (enter) test (Ctrl + Z)”. These commands, set up the initial handshake, configure the SMS mode to text, start a send text prompt to the entered phone number, and then send “test” the Ctrl + Z being the key combo to send the text. This phone number was used because it is Martin’s who was the person conducting the test. Then it was verified that these texts were received by the target phone. Finally, from the phone the message “Test” was sent back to the modem and using the command “at+cmgr=0” which reads the text in index 0 it was verified that the message “Test” was received from the correct phone number.

## Application and Modem System Summary

The phone app and modem function as intended and are ready to be integrated into the main system next semester. Remaining tasks are beautifying the app, adding more detailed notifications, adding multiple device functionality, and integrating the modem with the microcontroller. Overall, however the app is safe from misinputs and displays the information properly to the user in a form that is easy to read. The modem is configured to send and receive text messages and is ready to connect to the microcontroller that code simply must be written.

# Appendix - A

**Phone App and Modem Validation Images**

**Graphical user interface, website

Description automatically generated**

**Figure A.1** connected to virtual device in android studio and sending texts from it to update in temp, out temp, and battery life (top left, top right, bottom respectively)

**Graphical user interface

Description automatically generated**

**Figure A.2** These are the tests for sending a temperature, with no device connected (top right), non-number value (bottom left), temp out of range (two top right), correct entry (bottom right)

Graphical user interface

Description automatically generated**Figure A.3** This is showing that the notification will work when toggled on and sent the correct message.

Graphical user interface, application

Description automatically generated**Figure A.4** This is showing the battery being updated correctly but no notification because it is toggled off

Graphical user interface, application

Description automatically generated**Figure A.5** Showing that the app is in the background and the battery updated and the notification has popped up (the speech bubble with an exclamation point in the top)

Graphical user interface, text, application, chat or text message

Description automatically generated

**Figure A.6** Showing that when a correct number is entered the device attempts a connection

Timeline

Description automatically generated

**Figure A.7** showing the add a device does not accept incorrect types and will only accept a 10-digit value

Timeline

Description automatically generated

**Figure A.8** adding a new device while a device is already connected (left) and removing the device showing that it also wipes the device info (right)

Graphical user interface, text, application

Description automatically generated

**Figure A.9** Initial startup asking for permissions

Text

Description automatically generated

**Figure A.10** Receiving a text on PuTTY

Text

Description automatically generated

**Figure A.11** sending a text message via PuTTY on the modem