

VALENCIA COLLEGE

**Department of Electrical and Computer Engineering Technology (ECET)
Division of Engineering, Computer Programming, and Technology (ECPT)**

EET 4950

Senior Design Report

**Solar-Powered Ventilation with
Controlled Airflow for Parked
Automobiles**

Submitted by

Daniel Eisenbraun and Ian Matheson

Supervised by

Dr. Ejaz

April 14, 2023

Abstract

With summer temperatures reaching record highs and average temperatures rising globally, the management of heat in transportation is an issue of comfort and efficiency. Within an hour, a car in 95 °F weather can reach temperatures of up to 138 °F [1]. Such extreme temperatures are uncomfortable and lower the efficiency of car climate control systems by imposing a large initial load upon them [2].

This paper presents research, development, and testing of a solar-powered ventilation system with controlled airflow for parked cars. Research is focused on existing ventilation methods for parked cars and solar-powered ventilation systems.

The system uses arrays of DC fans secured to each car door via automobile vent visors to remove heat from a parked car through cracked windows. The external temperatures at each window are continuously monitored to control the direction of airflow through the car in response to the temperature gradient between windows. Power is provided by the battery of the car, whose charge is maintained by a roof-mounted solar panel through a PWM solar charge controller. To avoid draining the battery, the number of active fans will be controlled depending on the voltage of the battery.

Acknowledgments

We would like to first thank our advisor, Dr. Ejaz, for his guidance throughout the senior proposal. He offered the original inspiration for the proposed project and helped us make some pivotal decisions throughout the process. In everything we did, he gave invaluable feedback on how to improve or simplify, all while offering un-ending support. As a result, the proposed project is one that we are eager to bring into the world.

Our gratitude must also be extended to IEEE Region 3 and their Section Chair, Allen Jones. They provided us with a unique grant opportunity for projects addressing climate change. This grant, along with the worsening state of our planet's climate, heavily incentivized us to select a project that addresses climate change. The Region 3 Projects Committee offered excellent, detailed feedback on our application, which helped us develop the project to a higher standard. The funding was approved and will cover our entire budget.

Special thanks go to Dr. Radu Bunea, professor of Photonics at Valencia College, for suggesting ultrasonic proximity sensors as an alternative to infrared proximity sensors. Further thanks to Valencia College alumnus Geoff Colwell for offering his time to answer our questions regarding Peltier cooling devices for the first revision of the project. He helped us select a specific device to test and helped us to decide to ultimately drop Peltier devices.

We extend our undying gratitude to Duncan Kurtz for 3-D printing the intake nozzles for the project. Duncan printed multiple prototypes and even delivered the final product to the group.

Additionally, we would like to thank the project evaluators for their time and effort as they read this report. The opportunity to present our project and receive expert critique is vital to our education and the success of the project.

Finally, our love and gratitude go out to our family and friends (and cats), who offered feedback and endless support throughout.

Table of Contents

Abstract	ii
Acknowledgments	iii
Table of Contents.....	iv
List of Figures	vi
List of Tables.....	ix
Chapter 1.....	1
1.1 Introduction.....	2
1.2 Motivation.....	3
1.2.1 Ancillary Load Reduction	3
1.2.2 Comfort.....	4
1.3 System Overview	5
1.3.1 Principles	5
1.3.2 Engineering Requirements	5
1.3.3 Engineering Specifications.....	7
1.4 Objectives and Features	9
1.4.1 Power Module	9
1.4.2 Control Module	9
1.4.3 Cooling Module.....	9
1.4.4 Uses and Limitations.....	10
1.5 Comparison of Similar Products	10
1.5.1 Solar Sun Shields.....	10
1.5.2 Vent Visors / Window Cracking	10
1.5.3 HONUTIGE and MACHSWON Solar-powered Car Ventilators.....	11
1.5.4 ACOOL Ventilator for Parked Cars	12
1.5.5 The 1992 Mazda 929 Solar Powered Cooling System.....	13
1.6 Project Funding.....	15
Chapter 2.....	16
2.1 System Introduction	17
2.2 Cooling Module	17
2.2.1 Vent Visors.....	17
2.2.2 DC Centrifugal Blowers.....	17
2.2.3 Nozzles	18
2.2.4 Airflow Mode Diagrams.....	19
2.3 Control Module.....	20
2.3.1 Hardware Components	20
2.3.2 Software Components	26
2.4 Power Module.....	30
2.4.1 Primary Power Supply.....	30
2.4.2 Secondary Power Supplies	31
2.5 Power Budget.....	32
Chapter 3.....	34
3.1 Assembly and Contribution	35
3.2 Control Module Design.....	36
3.2.1 Control Module Hardware.....	36
3.2.2 Control Module Software.....	49
3.3 Cooling Module Design.....	57
3.3.1 Cooling Module Hardware.....	57
3.3.2 Vehicle Implementation and Issues.....	64
3.4 Power Module Design.....	67
3.4.1 Main Design	67
3.4.2 Solar Panel, Battery, and Charge Controller	68
3.4.3 Load Connections and Voltage Regulators	71
3.5 Troubleshooting	74

3.5.1	Raspberry Pi Pico	74
3.5.2	Battery Level Monitoring Software and MCU ADC Issues	74
3.5.3	Thermistor Circuit	79
3.5.4	Proximity Sensor	82
3.5.5	Relays and the Switch to Transistors for Fan Switching.....	90
3.5.6	Transistor Array Fan Activation Issues	91
3.5.7	Miscellaneous Issues and Fixes.....	94
3.6	Testing and Results.....	95
3.6.1	Acceptance Test Procedure and Test Setup.....	95
3.6.2	High-Level Requirement Testing	97
3.6.3	Medium-Level Requirement Testing	116
3.6.4	Low-Level Requirement Testing.....	120
3.6.5	Summary of Test Results.....	122
Chapter 4	124
4.1	Budget and Timeline.....	125
4.1.1	Budget.....	125
4.1.2	Timeline.....	126
4.2	Environmental Aspects	131
4.3	Health and Safety.....	131
4.4	Ethical Aspects.....	131
4.5	Social Aspects.....	132
4.6	Sustainability.....	133
Chapter 5	134
5.1	Summary and Conclusion.....	135
5.2	Suggestions for Future Work.....	135
References	138
Appendix A	143
A1.	Electrical Characteristics for NTC3950 Thermistor	144
A2.	Resistance Table and Graph for NTC3950 Thermistor	144
A3.	Specifications for RL5015B Centrifugal DC Blower	145
A4.	Electrical Characteristics for Boost Converter ME2108A.....	146
A5.	HC-SR04 Electrical Characteristics and Timing Diagram	146
Appendix B	148
B1.	Code	149
About Us	155
	Ian Matheson.....	155
	Daniel Eisenbraun	155

List of Figures

Figure 1.1 - Unit Under Test.....	2
Figure 1.2 - Main system block diagram	7
Figure 1.3 - Set of vent visors.....	11
Figure 1.4 – MACHSWON (a) and HONUTIGE (b) Ventilators	11
Figure 1.5 - ACOOL Ventilator Features	12
Figure 1.6 – 1992 Mazda 929	14
Figure 1.7 – Mazda 929 Ventilation Diagram with Solar Moon-Roof.....	14
Figure 2.1 - OEM U8220 2k000 vent visors.....	17
Figure 2.2 - DC Blower Dimensions	18
Figure 2.3 - Front Nozzles	18
Figure 2.4 - Rear Window Nozzle Design.....	19
Figure 2.5 - Cooling Modes Illustrated.....	19
Figure 2.6 – Raspberry Pi Pico Pinout.....	20
Figure 2.7 - Adafruit PCF8575 I2C GPIO expander board	21
Figure 2.8 – Thermistors.....	21
Figure 2.9 - PC817 Photocoupler.....	22
Figure 2.10 - HC-SR04 Ultrasonic Proximity Sensor and Timing Diagram.....	22
Figure 2.11 - 2N3904 NPN Transistor Pinout	23
Figure 2.12 - 1N4001 Diode	23
Figure 2.13 - Transistor array wiring diagram.....	24
Figure 2.14 - (a) 11 Position Terminal Block Plug, Female (b) 11 Position Terminal Block Header, Male	25
Figure 2.15 - 4-Pin IP65 Connector	26
Figure 2.16 - IP54 Waterproof Outdoor Electrical Box	26
Figure 2.17 – Solar PV (Left) [30], 121R Battery (Top Right), Cables (Bottom Right)...	31
Figure 2.18 – Lead Acid Battery PWM Charge Controller	31
Figure 2.19 - Adjustable Buck Converter	32
Figure 2.20 - 5V Boost Converter.....	32
Figure 3.1 - Full System Schematic	35
Figure 3.2 - Control Module Schematic Diagram	36
Figure 3.3 - MCU pinout	37
Figure 3.4 - Battery Level Monitoring Divider circuit	38
Figure 3.5 - Thermistor divider circuit	39
Figure 3.6 – HC-SR04 Wiring Diagram.....	40
Figure 3.7 - Prototyping board connection diagram for MCU and sensor components. ...	41
Figure 3.8 - Prototyping board connection diagram for PCF8575.	42
Figure 3.9 - Connection diagram for the transistor array.....	42
Figure 3.10 - System wiring diagram for determining required connectors.....	43
Figure 3.11 - Front and back of MCU and Sensor Prototype Circuit	43
Figure 3.12 - Soldered PCF8575 Prototype Circuit.....	44
Figure 3.13 - Front of the Transistor Array prototyping board.....	44
Figure 3.14 - Back of the Transistor Array Prototyping Board.	45
Figure 3.15 - Building and soldering the terminal block prototyping boards.....	45
Figure 3.16 - 5-Pin IP65 connector connections.....	46
Figure 3.17 - Wiring harnesses from the terminal blocks.....	47

Figure 3.18 - Fully wired control module.....	48
Figure 3.19 - Installation of PWM charge controller and ON/OFF switch in the lid of the control box.	48
Figure 3.20 - Closed control box.	49
Figure 3.21 - Main Loop Flowchart.....	50
Figure 3.22 – Flowchart for the Get Battery Mode function.	52
Figure 3.23 - Flowchart for Get Window Position function.	53
Figure 3.24 - Flowchart for Get Window Temperatures function	54
Figure 3.25 - Get Cooling Mode.....	55
Figure 3.26 - Flowchart for Fan Control function	56
Figure 3.27 - Placement of DC Blowers on Vent Visors.....	57
Figure 3.28 - DC Blower Array Attached to Car	58
Figure 3.29 - Nozzle design draft and measurement	60
Figure 3.30 - First revision of nozzle design	60
Figure 3.31 - Failed nozzle print.	61
Figure 3.32 - Nozzle with support printed in the epoxy channel.	61
Figure 3.33 - (a) Cura Slicer nozzle render with support blocker. (b) Clean print of the first revision of the nozzle.....	62
Figure 3.34 – First Revision Nozzle Minimal Clearance	62
Figure 3.35 - Second Revision of the Nozzle.	63
Figure 3.36 - Print of Second Nozzle Revision	63
Figure 3.37 - Third version of the nozzle design.	64
Figure 3.38 - Build-up with IP65 Cable Connections.....	65
Figure 3.39 - Torque Damage Incurred by Vent Visors and Wire Break.....	66
Figure 3.40 -3M Adhesive Spray Applied to Vent Visors with Paint	67
Figure 3.41 – Power Module Schematic Diagram.....	68
Figure 3.42 – 100W Solar Panel Design Layers.....	69
Figure 3.42 – Approximate Solar Panel Power Generation Graph.....	69
Figure 3.43 – Battery and Solar Panel Connection Points.....	71
Figure 3.44 – SPDT Switch and PWM Charge Controller Installed to Lid	71
Figure 3.45 – Buck Converter Functionality Test	72
Figure 3.46 – Boost Converter Schematic	73
Figure 3.47 - USB Connection to Raspberry Pi Pico.....	73
Figure 3.48 – Battery Voltage Divider ADC Test	75
Figure 3.49 – ADC Voltage Test using DMM	76
Figure 3.50 – Vin vs Vadc Error Chart.....	77
Figure 3.51 – Percentage Error Chart	77
Figure 3.52 – Vin vs 9 Adjusted Vadc Graph.....	78
Figure 3.53 - Thermistor output isolation using an op-amp.	80
Figure 3.54 - Testing the op-amp isolated thermistor circuit.....	80
Figure 3.55 - (a) Pot1 set to midpoint and Pot2 to lower extreme. (b) Pot1 is in the same position, and Pot2 is set to match V1 from (a).....	80
Figure 3.56 – Photocoupled temperature sensor schematic	81
Figure 3.57 - Photocoupled temperature sensor on breadboard	81
Figure 3.58 - (a) IR Proximity Sensor Circuit. (b) Detection of an object	83
Figure 3.59 – IR Proximity Sensor Circuit Schematic	83
Figure 3.60 – IR Board LED Test.....	84
Figure 3.61 - (a) IR Sensor output with window blinds closed. (b) With blinds open.	84
Figure 3.62 - (a) Ultrasonic Echo signal with no object present and (b) with an object present.....	85

Figure 3.63 - Ultrasonic sensor components soldered to the MCU prototyping board	85
Figure 3.64 – Ultrasonic Sensors with Connections and LED Indicator	86
Figure 3.65 – Ultrasonic Sensors All Active off Control Module	87
Figure 3.66 - Testing integrated operation of all four ultrasonic proximity sensors.	88
Figure 3.67 – Timing Diagram for Ultrasonic Sensors.....	89
Figure 3.68 – Ultrasonic Sensor Code Main Loop	89
Figure 3.69 – Multi-Sim Test for Compatibility with 2N3904 Transistors	91
Figure 3.70 – Breadboard Testing the 2N3904 Transistors	91
Figure 3.71 - Transistor Circuit with PCF8575.	92
Figure 3.72 - Schematic for PCF8575 Connections to Transistor Array.....	93
Figure 3.73 - PCF8575 Pullup Resistor Wiring.....	93
Figure 3.74 - Resoldered Fan Wiring	94
Figure 3.75 - Baseline Temperature Comparison Test 4/8/2023	97
Figure 3.76 - Temperature Testing Setup	98
Figure 3.77 - PWM Charge Controller View and Weather App Screen	98
Figure 3.78 - Test Vehicle Temperature vs Reference Vehicle Temperature	100
Figure 3.79 - Temperatures and Cooling Modes vs. Time. 3/14/2023	100
Figure 3.80 - Graph of Test Vehicle Temperature vs Time.....	101
Figure 3.81 - Temperatures and Cooling Modes vs. Time. 3/18/2023	102
Figure 3.82 – Temperature vs. cooling mode vs. standard deviation from 12:40 PM to 1:20 PM, 3/18/2023.....	103
Figure 3.83 - Temperatures and Cooling Modes vs. Time. 3/20/2023	104
Figure 3.84 - Graph of Test Vehicle Temperature vs Time.....	105
Figure 3.85 - Temperatures and Cooling Modes vs. Time. 3/25/2023	105
Figure 3.86 - Temperatures and Cooling Modes vs. Time. 3/26/2023	106
Figure 3.87 - Reference Vehicle with Sunshield and Test Vehicle	107
Figure 3.88 - Test Vehicle Temperature vs Sunshield Reference Vehicle Temperature.	108
Figure 3.89 - Reference Vehicle Window Cracking.....	108
Figure 3.90 - Test Vehicle Temperature vs Cracked Window Reference Vehicle Temperature	110
Figure 3.91 - Graph of Optimized Crossflow vs Non-Optimized Crossflow Temperatures	112
Figure 3.92 - Interior Inspection Post Rainfall	114
Figure 3.93 - Route Taken and Speed Observed	115
Figure 3.94 - Visual Inspection of Vent Visors and Activation of UUT.....	115
Figure 3.95 - PWM and DMM Voltage at Shutoff.....	117
Figure 3.96 - PWM Voltage Post Re-connection of Solar Panel.....	117
Figure 3.97 - Window Obstruction Test Before and After	118
Figure 3.98 - Load Shutoff Check Windows Up/Down	119
Figure 3.99 - Driver Front Seat POV	120
Figure 3.100 - Driver Back Windows POV	121
Figure 3.101 - LED Indicator Test Window Positions	122
Figure 4.1 - Two Survey Responses	132
Figure 5.1 – Survey Response for Proposed System Modification	137

List of Tables

Table 1.1 – Project Engineering Requirements	6
Table 1.2 –Engineering Specifications	7
Table 1.3 - Effects of Windshield Cover on Solar Panel Performance	13
Table 2.1 – Power Budget.....	33
Table 3.1 - Given values from the NTC3950 datasheet.....	38
Table 3.2 - Sensor Wire Color Code.....	46
Table 3.3 – Battery voltages and their respective ADC voltages and battery modes.....	51
Table 3.4 - Vin and Vadc Measurements and Percentage Error	76
Table 3.5 – Trendline Error and Adjusted Vadc Voltages.....	78
Table 3.6 - New thermistor circuit test	81
Table 3.7 - Baseline Temperature Comparison Test Data from 4/8/2023	97
Table 3.8 - Temperature Test 3/14/2023.....	99
Table 3.9 - Temperature Test 3/18/2023.....	101
Table 3.10 - Temperature Test 3/25/2023.....	104
Table 3.11 - Sun Shield Comparison Test Results 3/20/2023	107
Table 3.12 - Window Cracking Comparison Test Results 3/26/2023	109
Table 3.13 - Optimized vs Non-Optimized Crossflow Temperature Test Results	111
Table 3.14 – PWM Voltages During Temperature Tests	113
Table 3.15 - Fans Active with Decreasing Battery Level.....	116
Table 3.16 - Load Shutoff Test Results	119
Table 3.17 - LED Indicator Test Results	122
Table 3.18 – Summary of Test Results	123
Table 4.1 – Project Budget.....	125
Table 4.2 – Project Design Phase Timeline	127
Table 4.3 - Proposed Timeline by Week	128
Table 4.4 – Actual Timeline	129

Chapter 1

Introduction

Summary

This chapter will introduce and discuss the design of the Solar Powered Ventilation with Controlled Airflow for Parked Cars as well as the motivation for the project. This section will cover the requirements, specifications, and success criteria set by the team to ensure that the final product functions as intended. Lastly, this chapter will cover similar products and compare their effectiveness.

- 1.1 Introduction**
- 1.2 Motivation**
- 1.3 System Overview**
- 1.4 Objectives and Features**
- 1.5 Comparison of Similar Products**
- 1.6 Project Funding**

1.1 Introduction

The Solar Powered Ventilation System for Parked Cars is a device designed to remove heat from vehicles parked in direct sunlight. The device consists of four modified vent visors attached to the windows of the vehicle, each with an array of centrifugal DC blowers that draw in and exhaust air out of the windows of the vehicle. Temperature data are gathered from temperature sensors housed in the vent visors and used to determine the optimal air flow direction for maximum cooling. The vent visors and supporting electronics are wired to a control box housed inside the vehicle. The device is powered by the battery of the car. A solar panel mounted to the roof of the vehicle keeps the battery charged. *Figure 1.1* shows the device installed on the vehicle with a solar panel on the roof.



Figure 1.1 - Unit Under Test

1.2 Motivation

The primary motivating factors for the creation and implementation of our project are described in this section, including ancillary load reduction and comfort.

1.2.1 Ancillary Load Reduction

The National Renewable Energy Laboratory (NREL) has identified car ventilation improvement as one of the main players in vehicle ancillary load reduction [2]. The fuel consumption of a car can be increased by up to 20% due to AC usage. Over 10 years, the fuel burned by the AC can produce up to 4,600 kg of CO₂ over 10,000 km per year [3]. The AC's efficiency is especially diminished when the car first starts up after being parked in sunlight for extended periods, as it must work harder to remove the trapped heat from the car. To avoid discomfort, some people turn on the car and run the AC for a few minutes before getting in. Others open the windows while they run the AC as they begin driving to ventilate out the trapped heat. To run the AC of a vehicle at its maximum cooling, the engine must be active for the duration of the cooling period which, if temperatures in the vehicle are meant to be maintained, would require burning approximately half a gallon of fuel per hour [4].

A study by Chen et al. demonstrated that a temperature range of 5.4 °F was possible between 5 ventilation methods [5]. Since a 1.8 °F decrease in parked car air temperature can lead to 4.1% AC power savings [6], a difference of 5.4 °F due to the ventilation method can be considered significant. Thus, the development of a novel method for ventilation becomes worthwhile.

We can calculate potential energy savings based on the engineering requirements laid out later in this report using the rearranged equation for Specific Heat (1.1).

Equation for Specific Heat
$$Q = mC_p\Delta T \tag{1.1}$$

We will assume a 95 °F (35 °C) day with Sunny conditions which would cause a vehicle to reach an unregulated internal temperature of ~135 °F [1]. Given that the specific heat capacity of air at constant volume C_p is 1.005 kJ/kg. K at 300K (26.85°C) [7], the mass flow rate of the air using our ventilation array, assuming a cubic meter of air

is 1.225kg and a total internal volume of 102 ft³, we can calculate the mass of air within the vehicle as 3.526kg and apply it to our calculation for specific heat.

$$Q = (3.526 \text{ kg}) \left(1.005 \frac{\text{kJ}}{\text{kgK}} \right) (57.222^\circ\text{C} - 22.222^\circ\text{C}) = 124.027\text{kJ}$$

We can rearrange the variables to solve for the energy required under various temperature circumstances to determine how much energy is saved when the ΔT required to reach a comfortable temperature of 72°F (22.222°C) is reduced by the implementation of our device in a worst-case scenario.

In the worst-case scenario, the internal temperature reaches 130% Ambient or 45.5°C. Therefore, the energy required to reduce the internal temperature to 72°F upon start-up can be calculated as follows:

$$Q = (3.526 \text{ kg}) \left(1.005 \frac{\text{kJ}}{\text{kgK}} \right) (45.5^\circ\text{C} - 22.222^\circ\text{C}) = 82.489\text{kJ}$$

Therefore, if the device meets the engineering requirements, the energy requirements of the AC system to displace the initial heat would be reduced by a minimum of 33.5%. While this does not directly decrease the fuel consumed by the AC system, it reduces the time the AC needs to be on to reach these temperatures, thus reducing the overall consumption of fuel.

1.2.2 Comfort

Car interior temperature is a major factor in passenger comfort [8]. Discomfort due to temperature also produces a significant negative effect on driver vigilance and reaction time [9]. Thus, climate control is very important in cars to reduce discomfort and improve driver function and safety. Reducing the internal air temperature while parked will decrease the temperature differential between the outside air and the interior air, which will reduce passenger discomfort upon entering the vehicle. It will also decrease the amount of time required to cool down the vehicle to a comfortable temperature.

1.3 System Overview

In this section, we go into finer detail on the design and execution of the solar-powered ventilation system. Block diagrams and software flowcharts for the system design and each of its modules will be given and discussed.

1.3.1 Principles

The main principle of our project is to create a device that reduces the internal temperature of a vehicle without the use of the vehicle's integrated climate control system and to do so more effectively than existing products. In essence, the device is a secondary cooling system that does not use as much power as the AC system under continuous use. Furthermore, the project aims to provide a proof of concept for optimized airflow using a feedback control system. As the project is a proof of concept, it will not aim to be consumer ready. It will instead aim to provide a bed of research for future endeavors in this field.

1.3.2 Engineering Requirements

Figure 1.2 shows the block diagram for each of the Power, Control, and Cooling Modules and how they are used for this project. The requirements for the system as well as verification and success criteria are explicitly stated in *Table 1.1*. The table prioritizes everything that the device should aim to accomplish indicated by High, Medium, and Low-level categories. Methods for verifying that the device meets the requirements are given under Verification and Success Criteria.

Table 1.1 – Project Engineering Requirements

Level	Requirements: "The device shall..."	Verification and Success Criteria
High	Remove heat from a parked car such that the maximum internal temperature is no more than 130% of the maximum ambient temperature.	Internal and external air temperatures will be continuously monitored and plotted over time to be checked against specifications. Testing will be performed after the vehicle has been driven to simulate everyday use.
	Optimize Ventilation Airflow for maximum cooling based on temperature data from all four windows	Optimized Airflow will Provide $\geq 30\%$ More Cooling than Non-Optimized Airflow Within the Vehicle. - Will provide the most cooling between other methods of heat removal: - Sun Shields - Window Cracking - Crossflow Ventilation
	Charge the battery of the car with a solar panel such that it provides power to the system by keeping the battery of the vehicle charged when sunny.	The solar panel will prevent the battery from draining such that the system can run continuously (without the load cutting) for ≥ 8 hours while the sun is out.
	Keep all rain out of the vehicle while in use.	The device must keep water out of the vehicle while the device is installed and active. The electronics must be kept safe from the rain.
	Stay secured to the vehicle while driving	Drive the car with the ventilation fans installed at highway speeds (90 mph maximum) and ensure that structural integrity is not compromised.
Medium	Increase or decrease the number of active fans according to the voltage of the battery.	Number of fans active according to battery voltage and estimated percentage. 0 ---> Vbat < 12.0 V 6 ---> Vbat ≥ 12.0 V (71.4%) 12 ---> Vbat ≥ 12.3 V (85.7%) 16 ---> Vbat ≥ 12.6 V (100%)
	Run automatically unless switched off by the user.	The device will turn off when the battery voltage is < 12.0 V, windows are too high, or the switch is set to "OFF". It will reactivate when the battery voltage is ≥ 12.15 V, the windows are low enough, and the switch is "ON".
	Not hamper the closing of the window.	The window does not touch the device at any point in opening or closing.
	Automatically shut off when any of the windows are closed.	The windows must be open enough to allow airflow and for proper optimization of the airflow.
Low	Not hamper the vision of the driver.	Videos / Photos taken will show the POV from the driver to verify that the device does not obscure the driver's vision.
	Have an LED indicator for each window sensor that turns on when the window is sufficiently rolled down for system operation.	The LED will activate when the proximity sensor no longer detects the presence of the window.

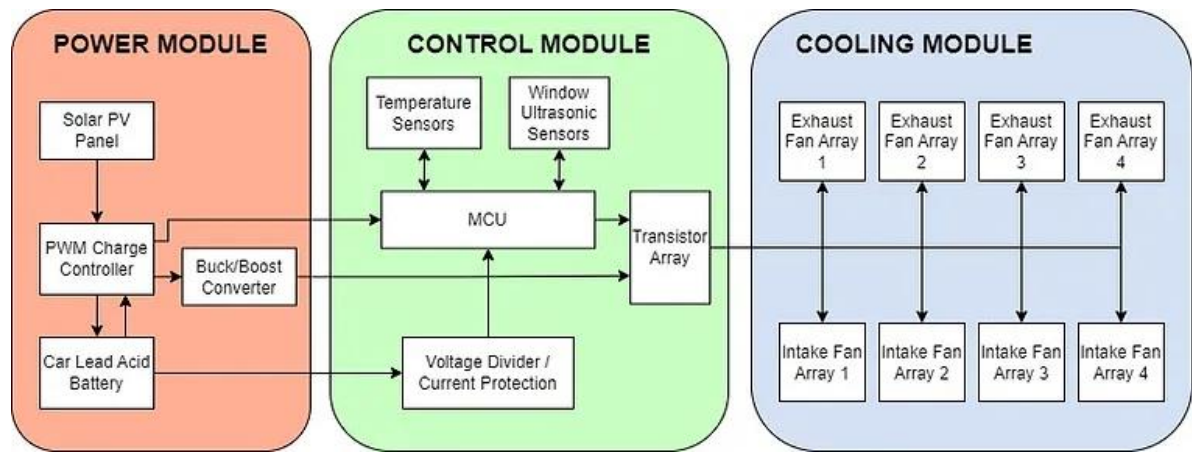


Figure 1.2 - Main system block diagram

1.3.3 Engineering Specifications

The specifications for each component required in each block of the power, control, and cooling modules are given in *Table 1.2*. The justification and criteria of each component are also given, along with the group member who took lead for the work for each block. This table specifies exactly what is required of each component and why.

Table 1.2 –Engineering Specifications

Module	Block	Components	Engineering Specifications	Justification & Criteria	Responsibilities
Power Module	Solar PV Panel		Solar Cell Array will provide at least 100W of power to the system.	Without solar power, the system would quickly drain the battery of the vehicle.	Daniel
	PWM Charge Controller		Charge Controller Suitable for Lead Acid Battery - Protects battery from overcharge - Overcurrent protection - Inverse current protection - 30A Max output current - 5V, 2.5A USB Power supply - Over-discharge protection	Allows the vehicle's battery to be charged while providing the system with a constant DC voltage and current. Front panel USB power supply can be used for MCU.	Daniel
	Boost Converter		Input: DC 1.1V-5V 500mA Max Output: DC 5V 380mA Max Efficiency > 90%	Allows for the Vcc of the Ultrasonics Sensors to be supplied the proper voltage which is necessary for operation.	Daniel
	Buck Converter		Input: DC 10V-15V 15A Max Output: DC 10V-15V 10A Max Efficiency > 90%	Keeps the load voltage constant at 12 V to run the fans.	Daniel

Control Module	MCU		The flash requirement is at least 2MB. Needs at least 1 Core. At least 42 GPIO Pins. At least 3 ADC ports. Can operate between 3.3V to 5V, up to 3A	Microcontroller is needed for temperature sensors, window position sensors, battery level monitoring, and fan control based on sensors.	Ian	
	Peripherals	Thermistor Circuits	100kΩ at 25°C Thermistor will provide a sufficient range of temperatures based on voltage to the MCU for airflow optimization.	Resistance from 70°F to 140°F will correspond to a voltage step sufficient to be read by the MCU. As temperature increases, the voltage going into the MCU will decrease based on the voltage divider.	Ian & Daniel	
		Ultrasonic Sensor	- 3.3V Supply - 3.3V Input. - <= 3.3V Output. - Range: >= 4 in. - Directional	Ultrasonic sensor changes output based on proximity to an object. Used to detect the presence of windows.	Ian & Daniel	
		LED Indicator	Green LED used for "OK" or "GO" indication when IR Sensor does not detect window. 2.0V at 20mA max	This will indicate to the user when the window has been sufficiently lowered for operation.	Ian	
		Battery Level Monitoring Circuit	Battery Level Voltage Divider will provide approx. 200mV difference between full charge and lowest allowable charge.	Battery Level monitoring will allow for a 5 V read at no greater than 16 mA to prevent damage to the MCU.	Daniel	
	Fan Switches	Transistors	Slow switching time At least 200 mA At least 20V	For switching the number of active fans	Ian & Daniel	
		Flyback Diodes	Flyback Diode suitable for Inductive Load of Fans -1.1V Forward Voltage -50V Maximum Blocking Voltage -1A Forward Current	A flyback voltage spike of negative 12 volts must be dissipated to prevent potential damage to fans and transistor switches.	Ian & Daniel	
		SPDT Switch	24V, 6A	On/Off switch for system required.	Daniel	
	Cooling Module	Ventilation Fan Arrays	16 Intake Fans	Dimensions: 50x50x15mm Voltage: 12V Max Current: 0.08A Max RPM: 5700±10%	Ventilation fans are required to remove heat from the vehicle.	Ian & Daniel
			16 Exhaust Fans			
Connectors and Wiring		5 Pin IP65 Connector and Cable	- 28 AWG or thicker - Waterproof connector	Allows all sensor connections to be run along one cable from each window.	Ian	
		Terminal Blocks for Sensors	- 11 Pin Male / Female - 12 Pin Male / Female - 5.08mm Pitch	Gets all sensor connections onto prototyping board for easy wiring to MCU	Ian	
		4 Pin IP65 Connector and Cable	- 28 AWG or thicker - Waterproof connector	Allows all fan connections to be run along one cable from each window.	Daniel	
		Terminal Blocks for Fans	- 14 Pin Male/Female - 5.08mm Pitch	Gets all sensor connections onto prototyping board for easy wiring to transistor array.	Ian	
Intake Nozzles		These nozzles must be custom 3D printed for the 12V fans.	Directs intake fan airflow into the car.	Ian		

1.4 Objectives and Features

The main objective of the Solar-powered Ventilation system is to remove trapped heat from a parked vehicle such that the air temperature in the vehicle is as close to the ambient air temperature outside the vehicle as possible. To accomplish this goal, the system will utilize a form of optimized air crossflow between the intake and exhaust fans attached to the vehicle to maximize the cooling provided by ventilation. The system will draw its power primarily from the battery of the car, which will be kept charged by a solar PV array. The system will be composed of three modules: the Power Module, Cooling Module, and Control Module.

1.4.1 Power Module

This module consists primarily of the solar PV array, car battery, and supporting electronics such as the PWM Charge Controller, buck converter, and boost converter which are required to convert the DC voltage and power provided by the solar panel into other DC values used by the MCU, fans, and sensors contained within the Cooling and Control Modules. This system will also allow excess power from the solar panel to charge the battery of the car while the system is in use. Lastly, the user will be able to turn the system on or off at will using an ON/OFF switch.

1.4.2 Control Module

The control module consists of a Raspberry Pi Pico MCU programmed with software that controls the cooling module based on data gathered by sensors. Temperature sensors will gather the temperature at each window. The MCU compares these temperatures to optimize the airflow direction through the vehicle. Proximity sensors will determine the position of each window. The system will only run if all windows are cracked enough for airflow. A voltage divider will cut the voltage of the car battery to a level that can be read by the MCU. The MCU will keep track of battery voltage and cut the number of active fans as the voltage lowers. Below 12.0 V, the MCU will turn off all fans.

1.4.3 Cooling Module

The cooling module consists of centrifugal DC fans, mounting hardware, nozzles for directing airflow from the intake fans, and vent visors which will be used to mount the fans to the space above each window of the vehicle.

1.4.4 Uses and Limitations

This device does *not* provide active cooling and is thus limited in its capacity to cool the vehicle to, at the very least, the outside ambient air temperature. Therefore, the project will be tested with the consideration of the lower limit of the temperature in mind.

1.5 Comparison of Similar Products

Conceptually, the device itself is not new, but most products do not appear to provide a significant level of cooling. This could be due to the methods of ventilation or heat rejection employed. Our design aims to improve on existing products while making use of a higher capacity of solar power than would typically be used in similar products.

1.5.1 Solar Sun Shields

Solar Sun Shields are made by a variety of manufacturers and purport to reduce the internal temperature of a vehicle by deflecting sunlight away from the vehicle. This method of keeping a vehicle cool has its merits since a lot of sunlight enters the vehicle through the large windshield, getting absorbed by interior surfaces and radiated as heat. However, as the sun changes position throughout the day, its ability to reject sunlight is compromised if sunlight happens to enter through the other windows of the vehicle.

1.5.2 Vent Visors / Window Cracking

Vent visors, like the ones used in the project, are installed on the frame of the vehicle outside the window. This allows the owner to crack the window of the car, facilitating ambient ventilation of the vehicle while it is parked without risk of rain entering the vehicle. Typically, cracking the windows results in the internal temperature being 10°F cooler than a vehicle that does not crack its windows under sunny conditions, but only temporarily [10]. Under the worst-case scenario, our product aims to ensure the internal temperature under such circumstances would only reach 130% of the ambient, or 123.5°F, providing 5% additional cooling at a minimum. Window cracking is also subject to the presence of wind to create an effective ventilation flow in and out of the vehicle. Our device emulates that effect with the addition of directional fans.



Figure 1.3 - Set of vent visors.

1.5.3 HONUTIGE and MACHSWON Solar-powered Car Ventilators

The HONUTIGE and MACHSWON Ventilators are like our design in that they use fans to ventilate air through the vehicle through the windows, but neither directly claims to cool the vehicle. Instead, both devices exhaust air from the vehicle to keep the interior air fresh rather than cool the vehicle. Both are also similar in that they utilize solar power to charge a battery and/or operate the device in the absence of the battery. Both models are remarkably similar in form factor and operation. Our product would ultimately provide cooling and fresh air, rendering this product unnecessary.

As seen in Figure 1.4 (a) and (b), both models utilize a single digital display for the temperature, a solar panel that projects out from the mount on the window, a set of switches to toggle the fans, and a light indicator for the state of charge of the battery.

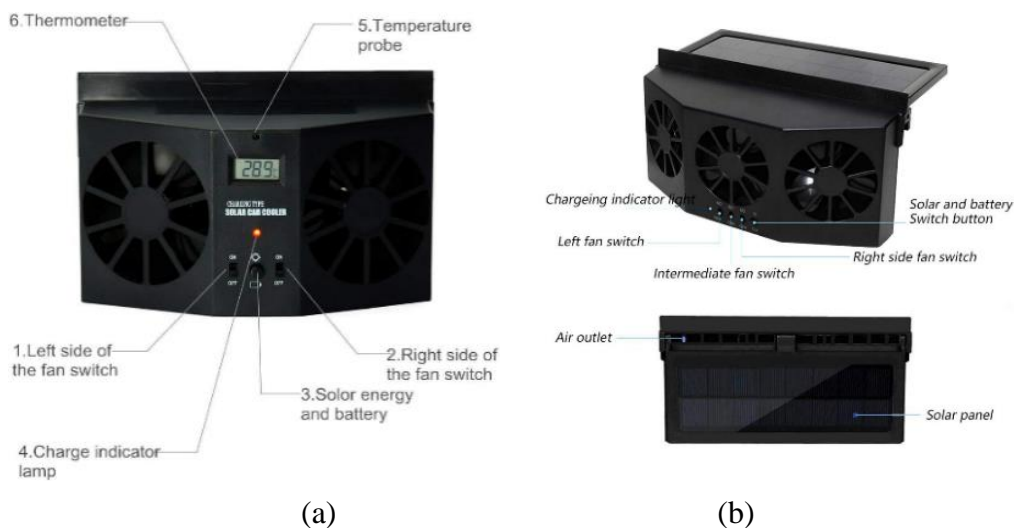


Figure 1.4 – MACHSWON (a) and HONUTIGE (b) Ventilators [11], [12]

1.5.4 ACOOL Ventilator for Parked Cars

The ACOOL Car Ventilator is much like our project, given that it uses crossflow ventilation, utilizes solar power and the car battery, and has hardware that is window mounted as a means to cool the vehicle. Where it differs is the overall cooling capability, cooling method, power taken from solar, and form factor. In addition, the ACOOL system adds an air filter and claims to clean the air that comes into the vehicle, something our project does not intend to accomplish.

As shown in *Figure 1.5*, the ACOOL Ventilator uses a solar panel placed beneath the windshield. While compact, this reduces the overall power output of the solar panel by as much as 45% based on testing done during the initial stages of our proposal as shown in *Table 1.3*. This severely limits the total power available to the fans for cooling without the risk of running down the battery of the car. The ACOOL system uses 8 total fans and claims to replace the air in the car every 5 minutes based on the volume displaced by the fans. Our system opts to use a roof-mounted panel that is more than capable of powering up to 16 active fans and optimizes airflow through the vehicle based on temperature differences between windows. The ACOOL system does not change the airflow direction, simply taking it in from one side and exhausting it from the other.

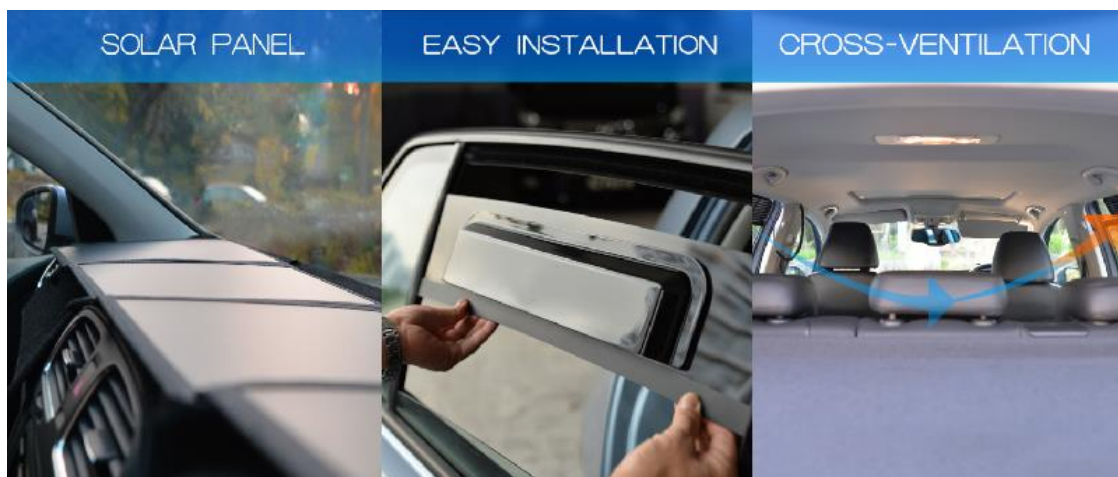


Figure 1.5 - ACOOL Ventilator Features [13]

Table 1.3 - Effects of Windshield Cover on Solar Panel Performance

Time	Above Windshield					Under Windshield				
	I _{sc} (A)	V _{oc} (V)	P (W)	Panel Temp		I _{sc} (A)	V _{oc} (V)	P (W)	Panel Temp	
				°C	°F				°C	°F
12:00 PM	0.29	5.55	1.61	43.3	110.0	0.13	5.20	0.68	38.3	101.0
1:00 PM	0.37	5.35	1.98	52.8	127.0	0.16	5.00	0.80	52.2	126.0
2:00 PM	0.28	5.24	1.47	55.9	132.6	0.18	4.96	0.89	56.3	133.3
3:00 PM	0.41	5.35	2.19	60.0	140.0	0.19	5.00	0.95	59.4	139.0
4:00 PM	0.33	5.37	1.77	56.5	133.7	0.17	4.96	0.84	56.3	133.3
5:00 PM	0.26	5.40	1.40	54.4	130.0	0.12	5.09	0.61	53.3	128.0
Average	0.32	5.38	1.74	53.8	128.9	0.16	5.04	0.80	52.6	126.8
Avg Power Loss Due to Windshield Cover						45.80%				

In addition, the unit is mounted within the window frame itself, which requires a custom insert for the vehicle in question, much like our product requires a custom vent visor for the make and model of a vehicle. Vent-visors, however, are more ubiquitous with models already available for manufacture or sale for a variety of vehicles.

Lastly, the ACOOL ventilator makes use of crossflow ventilation as shown above in *Figure 1.5* which is a suitable method for cooling an area [14]. Through some initial testing, however, it was determined that additional cooling is achievable by taking in air from the coolest areas of the car and exhausting the air from the hottest areas. Testing methods and data are covered in chapter 3.

1.5.5 The 1992 Mazda 929 Solar Powered Cooling System

The Mazda 929, shown in *Figure 1.6*, was a luxury sedan built by Mazda in 1992 with features that, at the time, were unique to Mazda. We will focus primarily on the solar-powered ventilation system present within the car which works very similarly to our system and would likely be an excellent reference for future implementation into newer vehicles. This vehicle and its implementation of air ventilation cooling is the strongest inspiration for this project.



Figure 1.6 – 1992 Mazda 929 [15]

The Mazda 929 utilizes a set of solar cells that are imbedded into the glass sunroof of the vehicle. These solar cells power a set of fans that extract hot air from the back of the vehicle while creating a pressure gradient sufficient to draw in outside air from the currently existing ventilation system located toward the front of the vehicle. This is shown in *Figure 1.7*.

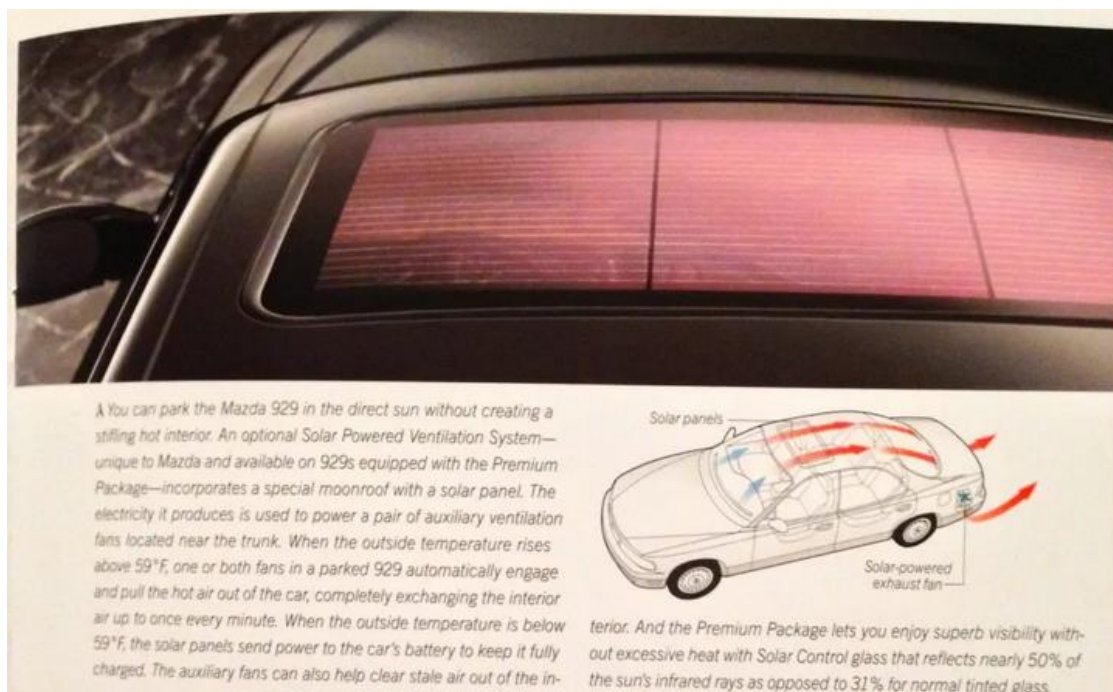


Figure 1.7 – Mazda 929 Ventilation Diagram with Solar Moon-Roof [16]

Based on the diagram, we can see that the Mazda 929 Ventilation is like that of the ACOOL System in that it provides a form of Crossflow Ventilation taking in air from one side of the vehicle and exhausting it out of another side. Also, the issue of solar efficiency through the windshield is resolved due to the moon-roof being located on the

roof and in direct sunlight with nothing to absorb or attenuate the incoming light. The Mazda 929 system also trickle charges the battery of the vehicle if there is sufficient power provided by the panel, which reflects the operation of our system. Like the ACOOL system, the Mazda 929 system does not optimize the direction of the airflow. In Chapter 3, we will show how this reduces the potential cooling capacities of both designs.

It is unclear why the ventilation system implemented in the Mazda 929 was not added to later models nor explored as an option by other car manufacturers. One reason may be related to poor sales in the US. At the time in 1992, the MK2 Mazda 929 with Solar-Powered Ventilation cost \$34,125 which would cost approximately \$72,542 adjusted for inflation [15], [16]. Despite the many technologically advanced features of the vehicle at the time such as the solar-powered ventilation system, fuzzy logic cruise control, and a 4-wheel steering system the competition present such as the Acura Legend and Infiniti J30 may have kept Mazda from securing a sound hold in US markets [17]. Also, the fact that this technology was only available as a part of the luxury market sets it apart from the goals of our project. As will be discussed in Chapter 4, the cost associated with a device such as the one proposed is a major contributor to desirability and accessibility within the consumer base.

1.6 Project Funding

Our project holds the potential for increasing the efficiency of the car's AC system by reducing the initial load. It also aims to make the car more comfortable upon entry. Therefore, the proposed project addresses climate change as both a remedy and an adaptation. For this reason, an application for grant funding for projects addressing climate change was submitted to IEEE Region 3. The application was reviewed by the Region 3 Projects Committee and feedback was given. Once all feedback was addressed, the project and application were updated and resubmitted for approval. The project was approved on Nov. 16, 2022, and most of the project budget was covered by the grant.

Chapter 2

Background Research

Summary

In this chapter, we discuss the technical aspects of our design including the Power, Control, and Cooling Modules as well as a description of the Power Budget for the project.

- 2.1 System Introduction**
- 2.2 Cooling Module**
- 2.3 Control Module**
- 2.4 Power Module**
- 2.5 Power Budget**

2.1 System Introduction

The project is comprised of a cooling module, a control module, and a power module. The background research for the hardware and software components for each of the modules is given in this chapter. Any already existing components will be discussed.

2.2 Cooling Module

The cooling module consists of vent visors, arrays of DC centrifugal blowers, and nozzles for intake fans.

2.2.1 Vent Visors

Each of the car's windows was fitted with a vent visor assembly denoted by a numerical value 1 through 4. The front driver window is designated as Vent Visor 1, the back driver as Vent Visor 2, the front passenger as Vent Visor 3, and the back passenger as Vent Visor 4. Custom-fitted vent visors are available for most cars, but for this project, we used the OEM U8220 2k000 for the 2012 Kia Soul. Fans were secured to the visors using nuts and bolts. Spacers were used to separate the intake fan inlets from the surface of the vent visor.



Figure 2.1 - OEM U8220 2k000 vent visors [18]

2.2.2 DC Centrifugal Blowers

RBL5015B 2.4 CFM DC Blowers were attached in rows on the vent visors with adequate spacing made to accommodate the nozzles of each fan. On the front windows, 5 intake and 5 exhaust fans were mounted, while on the back windows, 3 intake and 3 exhaust fans were mounted. After attachment, a clear water-resistant epoxy was placed over the mounting hardware to ensure that no rain could enter the vehicle through the holes drilled in the vent visor.

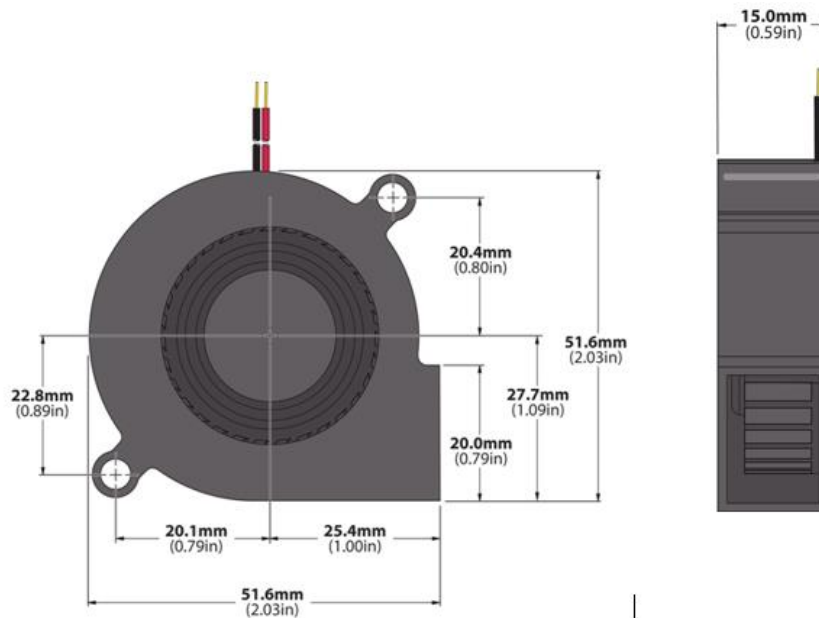


Figure 2.2 - DC Blower Dimensions [19]

2.2.3 Nozzles

Nozzles were custom 3-D printed for intake fans to direct airflow into the car. These were designed with an in-channel attachment for the fans which would have an adhesive placed within to create a seal to allow for better airflow and structural stability. Two variations of the nozzle were developed since the spacing requirements for the front and back windows were different. Shown in *Figure 2.3* is the front window variation which features a longer channel than that of the back.

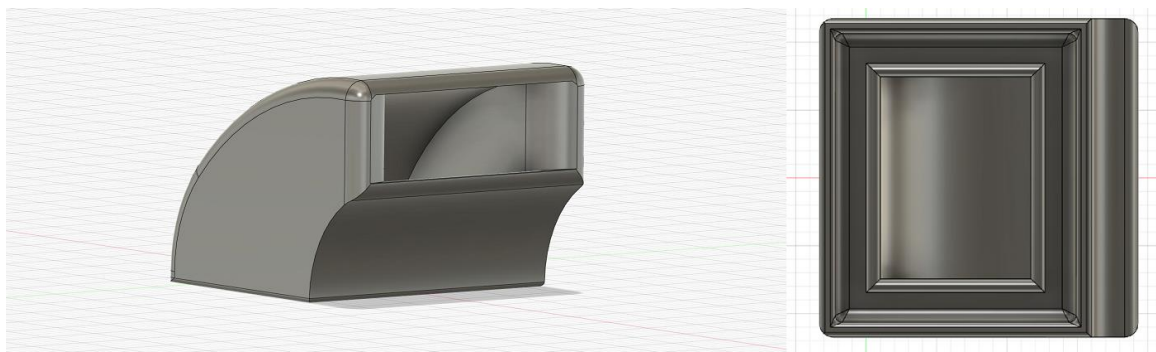


Figure 2.3 - Front Nozzles

The back window variation is shown in *Figure 2.4*. It has a shallower curvature and a lower channel depth to accommodate the strict spacing requirements of the back window vent visor.

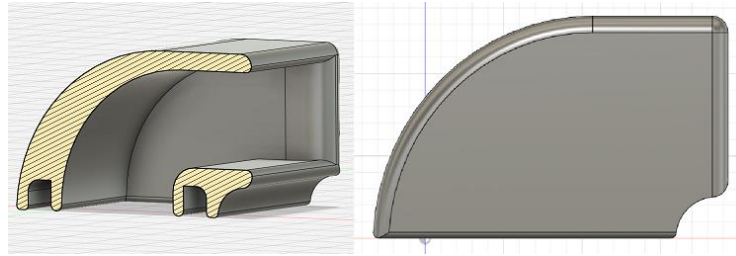


Figure 2.4 - Rear Window Nozzle Design

2.2.4 Airflow Mode Diagrams

The device directs airflow through the vehicle depending on the temperatures measured at each window. The airflow diagrams in *Figure 2.5* define 4 airflow modes. These diagrams dictate the possible airflow modes that the system can induce.

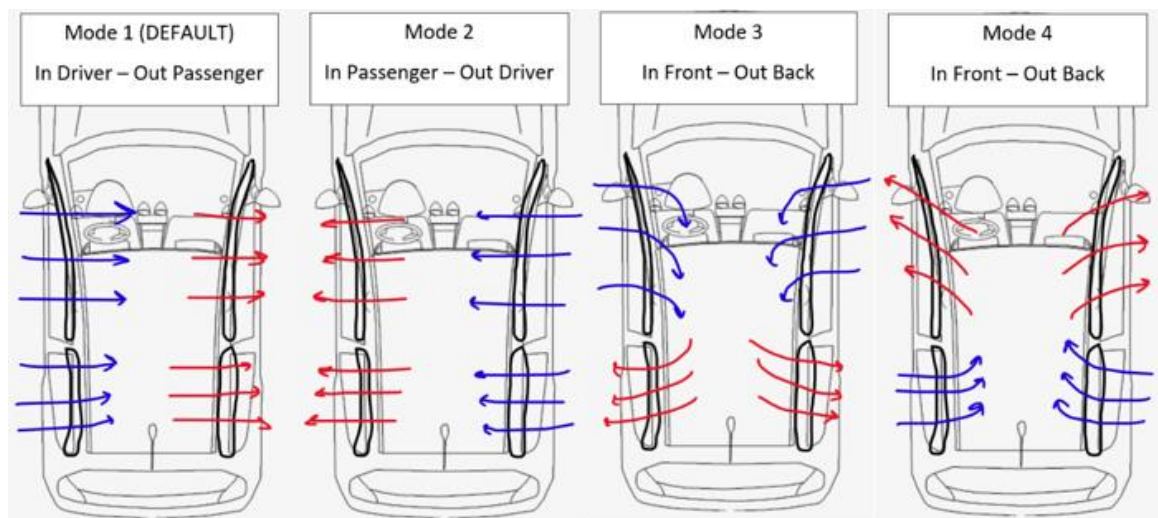


Figure 2.5 - Cooling Modes Illustrated.

In Cooling Mode 1, the temperature difference between the driver's side and passenger side windows is greater than or equal to the difference between front and back windows. The driver's side is cooler than the passenger side. In this mode, the air is taken in from the driver's side and exhausted from the passenger side.

In Mode 2, the temperature difference between the driver's side and passenger side is greater than or equal to the difference between front and back and the passenger side is cooler than the driver's side. In this mode, the air is taken in from the passenger side and exhausted from the driver's side.

In Mode 3, the temperature difference between the front and back windows is greater than the difference between the driver’s side and passenger side windows. The front windows are cooler than the back windows. In this mode, the air is taken in from the front of the vehicle and exhausted from the back of the vehicle.

In Mode 4, the temperature difference between the front and back windows is greater than the difference between the driver’s side and passenger side windows and the back windows are cooler than the front windows. In this mode, the air is taken in from the back of the vehicle and exhausted from the front of the vehicle.

2.3 Control Module

The proposed Control Module consists of two modules: the hardware and software modules. Data gathered from the hardware module are interpreted by the software module, which then controls the cooling module via control module hardware.

2.3.1 Hardware Components

The hardware for the control module includes an MCU, Temperature Sensors, Ultrasonic Proximity Sensors, Battery Level Monitor, and Transistor Array.

MCU

At the heart of the Control Module is the MCU which acts to control the system. For this, the Raspberry Pi Pico was used. It is a low-cost MCU with a high clock speed of 133MHz, 264kB of SRAM, and 2MB of onboard flash, all of which is sufficient for the software module. The chip itself comes with 26 GPIO pins, three of which are 12-bit ADC ports that will be used to take readings from the various sensors.

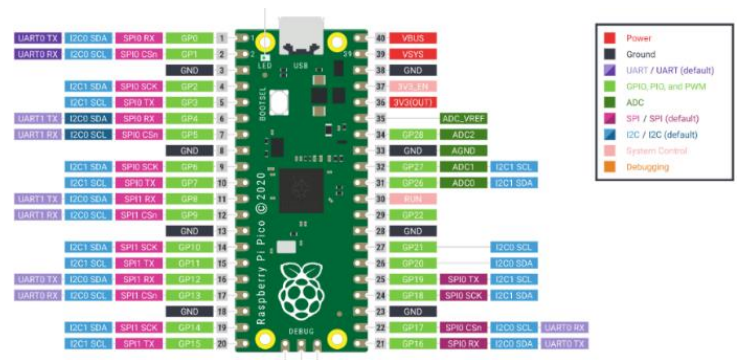


Figure 2.6 – Raspberry Pi Pico Pinout [20]

Since the Raspberry Pi Pico only has 26 GPIO pins while the project requires 35, the Adafruit PCF8575 breakout board will be used. It will provide 16 additional GPIO pins, bringing our total GPIO count to 42.

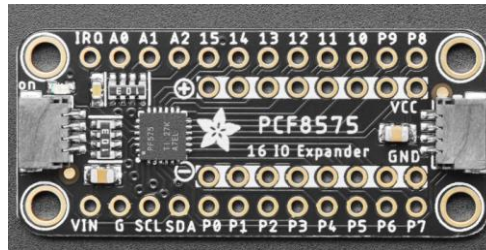


Figure 2.7 - Adafruit PCF8575 I2C GPIO expander board [21]

Battery Voltage Divider

To measure the battery voltage, a resistive voltage divider is used to cut down the voltage seen by the ADC of the MCU.

Temperature Sensors

Resistive circuits using NTC3950 Thermistors were used to gather the temperature at each window. The thermistors are designed in such a way that the resistance of the Thermistor drops as the temperature increases. In the case of our Thermistors, the nominal value at 25°C is 100k Ω . For every $\pm 1^\circ\text{C}$ change in temperature, there is a change by $\pm 5\text{k}\Omega$ in the thermistor.



Figure 2.8 – Thermistors [22]

By placing a thermistor in a voltage divider at the input to the ADC, the temperature variation will correspond to the changing voltage across the thermistor as its resistance changes. PC817 Photocouplers are used to isolate the outputs of each thermistor circuit to the ADC.



Figure 2.9 - PC817 Photocoupler [23]

Ultrasonic Proximity Sensors

To know whether the window is down, ultrasonic proximity sensors are used. The model used for this purpose is the HC-SR04 ultrasonic sensor which functions by taking a trigger pulse input from the MCU at 10 μ s and creating a 40kHz soundwave at the transmitter. This wave of sound is sent out and reflected back to the sensor if an object is present. If an object is detected, the receiver generates an echo pulse that corresponds to the distance from the transmitter of the detected object i.e., the greater the distance from the transmitter, the larger the pulse width of the echo pulse returned to the MCU.

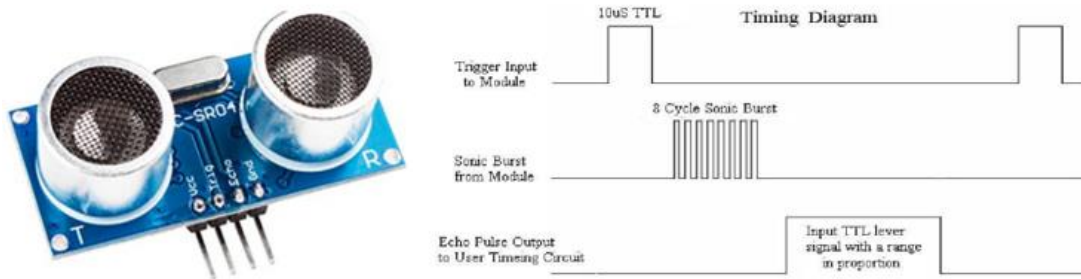


Figure 2.10 - HC-SR04 Ultrasonic Proximity Sensor and Timing Diagram [24]

Green Indicator LED

To indicate the status of the system, a green indicator LED is used. The LED is placed on the front portion of Vent Visor 1 on the front driver’s side window. It is placed in clear view of the driver. When any of the windows are up, the system is off, and the LED will be off. If all windows are sufficiently lowered, the system is active, and the green LED will be on.

Transistor Array

To activate and deactivate the fans with the MCU, a switch was needed that could pass 12.0 V through to the fans when triggered by the MCU GPIO pins. For this purpose, an

array of 24 2N3904 transistors was wired with their bases connected to the various GPIO pins of the MCU. These transistors allow for loads of up to 200mA to be properly powered with minimal power loss across the transistors. As each fan uses only 80mA per fan, this means each transistor can reasonably provide enough current for up to 2 fans per transistor.

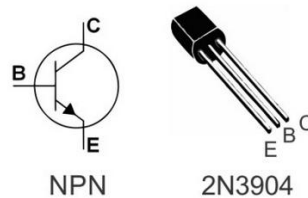


Figure 2.11 - 2N3904 NPN Transistor Pinout

The transistor array is broken up into 4 sections of 6 transistors each that are then separated further into intake and exhaust transistors with 2 fans attached per transistor on Vent Visors 1 & 3 and 1 fan attached per transistor on Vent Visors 2 & 4. As stated previously in the MCU section, since the Raspberry Pi Pico only has 26 GPIO pins, The Adafruit MCP23017 I2C breakout board was used to provide additional GPIO functionality to control the transistor array and thus the fans. Transistors connected to the MCU through the I2C Expander board require a 1 k Ω resistor added in series with the GPIO output to properly trigger the transistors. Lastly, a 1N4001 flyback diode is attached in parallel to the collector of each transistor to help dissipate the inductive load of the DC Blowers when they are deactivated.

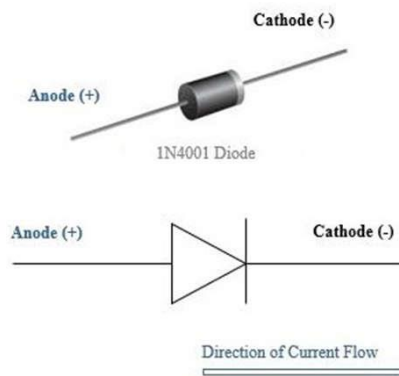


Figure 2.12 - 1N4001 Diode [25]

To carry out the airflow modes detailed in the cooling mode section, the MCU will switch fans on and off through the transistor array. With a total of 10 fans in each of the front windows and 6 in each back window, they are separated into arrays of intake and exhaust fans. The front driver-side window corresponds to Array 1, the back driver-side window to Array 2, the front passenger-side window to Array 3, and the back passenger-side window to Array 4. Each fan array consists of intake and exhaust groups, each with 3 fans or fan pairs, each with 3 fans or fan pairs.

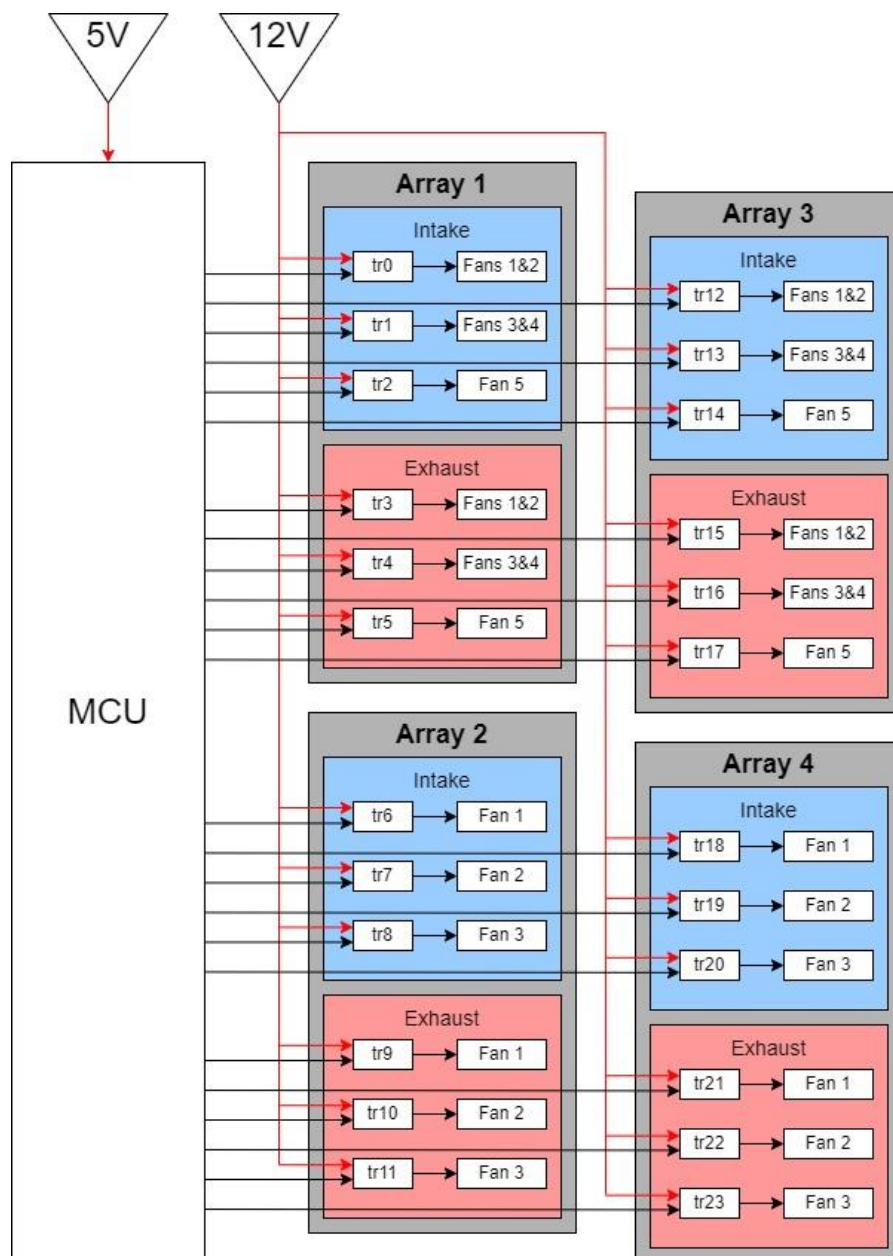


Figure 2.13 - Transistor array wiring diagram

Prototyping Board

A kit of various small through-hole prototyping boards was used to create all the connections for the control module. The boards have a through-hole pitch of 5.08 mm. The group decided to build the control module on prototyping boards for increased flexibility. Swapping components and fixing connection issues on the prototyping boards proved to be quick and simple.

Terminal Blocks

Terminal blocks are used to connect all cabling to the control module, creating an organized wiring harness for each group of sensors and fans. The blocks used have pins for through-hole soldering with a 5.08 mm pitch.

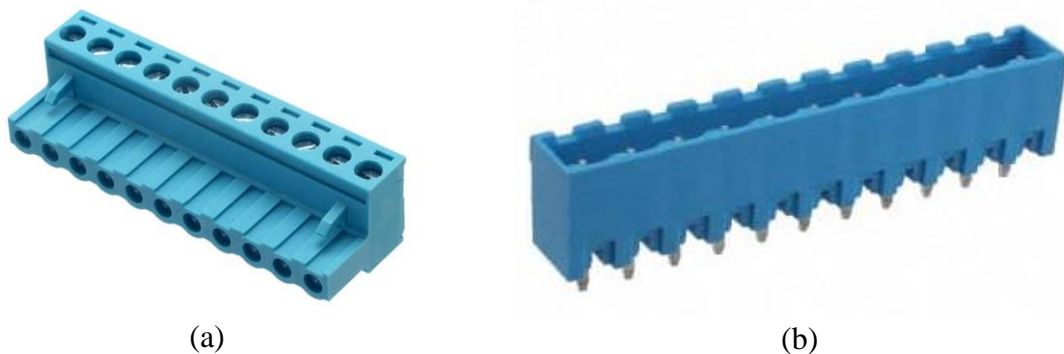


Figure 2.14 - (a) 11 Position Terminal Block Plug, Female (b) 11 Position Terminal Block Header, Male [26]

IP65 and IP67 Connectors and Cables

Multi-pin cables and IP65 waterproof connectors are used to connect the fans and sensors to the control module. The sensors are connected using 5-pin cables and connectors, while 4-pin cables and connectors are used for the fans. IP67 waterproof connectors and cables are used to connect the solar panel to the PWM charge controller. The IP65 and IP67 connectors are easy to connect and disconnect, making assembly and disassembly of the project quick and painless.



Figure 2.15 - 4-Pin IP65 Connector (ALITOVE 5 Pairs 4 Core Electrical LED Connector IP65 with 20cm 18AWG 4X 0.5mm² Cable for Car Truck Boat Indoor/Outdoor LED Strip Lights/String, n.d.)

Weather-Proof Box

Given the risk of rain entering the vehicle during testing, as the device had not yet been confirmed to keep water out of the vehicle, a method of housing the components of the control module was needed. To this end, the use of an outdoor electrical box was chosen because it has wire routing and adequate space for the terminal blocks, MCU, and other supporting electronics as well as room to house the PWM charge controller.



Figure 2.16 - IP54 Waterproof Outdoor Electrical Box [28]

2.3.2 Software Components

The tools used to create the software for the control module are detailed in this section. These include the Thonny integrated development environment (IDE), CircuitPython, required CircuitPython Libraries, and any programs, methods, or code sourced from the internet. CircuitPython is largely the same as Python, so an introduction to Python will be given.

Python

Python is a high-level programming language with syntax that is easy to write and understand. It often reads like plain English. Since it is a high-level programming language, it must be interpreted by the computer to do tasks. This means it takes longer to execute tasks, as the code must go through multiple stages of abstraction before it can be read and executed by the machine. This presents no issue for the project since fast switching times and processing speeds are not required by the project. The ease of programming in Python sped up the development process significantly, making it an excellent language for the project.

Syntax

In Python, white space serves a function. Indentation tells Python how to interpret scope. This forces the code to be neatly indented, making it easier to read. For example, the code below will throw an error if the second line indentation is removed.

```
if 1<2:  
    print("One is less than two.")
```

Comments

Comments are started with #. They are useful for improving the readability of code.

```
print("Hello World!") # This is a comment
```

Variables

In Python, there are no variable declarations. A variable is created when a value is assigned to it. The data type is automatically assigned to the variable depending on the data assigned to it, though the datatype can also be manually specified.

```
Vprev = 0.000 # Default previous voltage reading  
bat_prev = 0 # Default previous battery mode
```

Lists

Lists are variables that store multiple values. They are assigned in the same way as variables, but all items in the list are placed in square brackets. Items in the list are separated by commas. Lists can be indexed as shown in the example below.

```
>>> On_Daniels_Desk = ["Coffee", "Bananas", "Oscilloscope"]
>>> print(On_Daniels_Desk[1])
Bananas
```

If-Else

Conditional statements are made using if, else, and elif. These use logic to determine if a condition is satisfied or not. Different codes can be executed based on the results.

```
x = 5
y = 10

if x<y:
    print("x is less than y")
elif x==y:
    print("x is equal to y")
else:
    print("x is greater than y")
```

Loops

Two types of loops are used in this project: while loops and for loops. While loops will execute code as long as a condition remains true.

```
i = 0
while i<5:
    print(i)
    i = i+1
```

For loops execute code for each item in a sequence.

```
roast = ["Light", "Medium", "Dark"]

for coffee in roast:
    print(coffee)
```

Functions

Functions are like miniature programs that run only when they are called. Functions can be called from the main program or other functions. They can also be called from other files. They can be defined with or without input arguments. In the example below, a function is defined without an input argument. It will print all the items in the roast list.

```
def brew():
    roast = ["Light", "Medium", "Dark"]
    for coffee in roast:
        print(coffee)

brew()
```

In the following example, an input argument *num* is passed into the function and multiplied by 4. The function returns the result.

```
def times_4(x):
    x = x*4
    return x

num = 8
new_num = times_4(num)
print(new_num)
```

CircuitPython

The Raspberry Pi Pico startup guide recommends MicroPython for use with the board. This project uses the PCF8575 GPIO Expander, which is not supported by MicroPython. Luckily, Adafruit's CircuitPython has support for both the Raspberry Pi Pico and the PCF8575. Thus, CircuitPython was downloaded and installed on the Raspberry Pi Pico.

CircuitPython, like MicroPython, is a version of Python developed for use with microcontrollers. It comes with an extensive library of hardware support for various Adafruit sensors and peripherals. This project uses the `adafruit_pcf8575.mpy` library to program the I2C GPIO expander board.

In terms of language, CircuitPython is virtually identical to Python with a few minor variations. For example, CircuitPython does not yet support `match/case` statements. Instead, `if` and `elif` are used.

Thonny IDE

Thonny is a basic IDE that works well for programming the Raspberry Pi Pico with Python. It is recommended in the Raspberry Pi Pico startup guide. It also supports CircuitPython.

Programs & Methods

The following programs or methods were downloaded or copied from the internet for use in this project. Little or no modification was needed.

Except Method

Following the `try` clause in the main loop is the `except` clause. This code was adapted from Adafruit's instructional page on data logging [29]. By blinking the onboard LED, it indicates on the MCU if the filesystem is full, or some other error occurs.

```
except OSError as e: # Typically when the filesystem isn't writeable...
    delay = 0.5 # ...blink the LED every half second.
    if e.args[0] == 28: # If the filesystem is full...
        delay = 0.25 # ...blink the LED faster!
    while True:
        pi_led.value = not pi_led.value
        time.sleep(delay)
```

boot.py

The following program was downloaded directly from the Adafruit website, also from their instructional page on data logging. It allows CircuitPython to write to the filesystem when the Raspberry Pi Pico first starts up.

```
# SPDX-FileCopyrightText: 2021 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
boot.py file for Pico data logging example.
Make the filesystem writeable by CircuitPython.
"""

import storage
storage.remount("/", readonly=False)
```

2.4 Power Module

The power module contains only hardware, comprised of the primary power supply and secondary power supplies.

2.4.1 Primary Power Supply

Power is provided to the system by the 121R 12 Volt Battery of the car, which is kept charged by a 100W Solar PV panel mounted to the roof of the test vehicle. It is wired via cables that run down the body of the car to the location of the battery from the PWM Charge Controller.

These will both be wired to a Charge Controller for the Lead Acid battery which will act as not only a charging circuit but provide the same output voltage as the current voltage of the car battery as well as a 5V USB port, at up to 30A and 2.5A respectively, to the Load, MCU, and other peripherals with over-discharge protection built in.



Figure 2.17 – Solar PV (Left) [30], 12V Battery (Top Right) [31], Cables (Bottom Right)[32]



Figure 2.18 – Lead Acid Battery PWM Charge Controller [33]

2.4.2 Secondary Power Supplies

Because the load output of the PWM Charge Controller is dependant on the voltage of the car battery, a means of regulating the voltage up or down to 12.4V at 1.28A or greater was needed to provide adequate power to the DC blowers. To that end, a generic 150W Buck Converter was selected based on this criterion. It was purported to take an input of 5V to 30V DC and output a stable low ripple voltage of 1.25V to 30V with 93% efficiency. Upon receiving the converter, this was verified at 92.85% efficiency under load and wired to the transistor array for use.



Figure 2.19 - Adjustable Buck Converter [34]

A similar issue arose when we began to implement the HC-SR04 Ultrasonic Sensors seen in the Control Module section of the report. Most of the sensors and GPIO peripherals for this project were tailored to run off 3.3V from the MCU, however, the Ultrasonic sensors required 5V to operate properly. To this end, a Generic 5V boost converter capable of outputting up to 5V at 300mA to raise the voltage from the 3V3 pin of the MCU up to 5V is required by the sensors. The current draw per sensor is low at only 15mA when working. Thus, the current requirement from the converter was more than adequate for our goals.



Figure 2.20 - 5V Boost Converter [35]

2.5 Power Budget

The power budget in *Table 2.1* gives the power requirements of each component. The power module section shows how much power will be supplied by the Power Module with losses through the PWM Charge Controller and Buck Converter accounted for. The cooling and control modules draw power from the battery. The wattage listed for the solar panel is its rated wattage. This represents the maximum power that it can output. Thus, when the solar panel provides maximum power, the battery of the vehicle will charge.

Table 2.1 – Power Budget

Module	Component	Qty.	Voltage (V)	Current (A)	Total Current (A)	Power (W)
Power Module	Solar PV Panel	1	18.15	5.51	5.51	100
	PWM Charge Controller	1	18.15	-1.31	-1.31	-25
	Buck Converter	1	14.4	-0.364	-0.364	-5.25
Total Power Provided by Power Module						69.75
Cooling Module	DC Blowers	16	12.1	0.08	1.28	15.52
Control Module	Raspberry Pi Pico H	1	5	0.091	0.091	0.455
	Temp Sensors	4	3.3	33E-6	132E-6	435.6E-6
	Battery Level Monitoring	1	12	120E-6	120E-6	0.0014
	Ultrasonic Sensor	4	5	0.015	0.060	0.300
	Transistors (2 Fans)	4	0.36	0.160	0.640	0.231
	Transistors (1 Fan)	4	0.21	0.080	0.240	0.068
Total Power Drawn from Battery						16.58

Chapter 3

Contribution

Summary

This chapter details the design process and the contributions that each member of the team made toward implementing the design. The troubleshooting and testing of the project are also discussed. Testing results are documented for each requirement of the project.

- 3.1 Assembly and Contribution**
- 3.2 Control Module Design**
- 3.3 Cooling Module Design**
- 3.4 Power Module Design**
- 3.5 Troubleshooting**
- 3.6 Testing and Results**

3.1 Assembly and Contribution

Assembly of the project consisted of drilling and spacing holes out along the OEM Vent Visors, designing and 3D printing nozzles for the DC Blowers, affixing those nozzles to the DC Blowers, creating wiring harnesses that would reach all windows and vent visors, and creating a box that housed the supporting electronics such as the Terminal Blocks for connections, the MCU, Circuitry, Converters, and PWM Charge Controller. Ian worked on the wiring of the terminal blocks, creating the 3D model for the nozzles, soldering the vast number of connections to the MCU, creating the wiring harnesses, and coding the software. Daniel worked primarily on the vent visors, ensuring adequate spacing was provided for each of the DC Blowers and Nozzles, implementing the Power Module to the housing for supporting electronics, testing new components and how to implement them, testing procedures for requirements, and designing the initial Thermistor and Battery Level Monitoring circuits. The full schematic of the system is shown below.

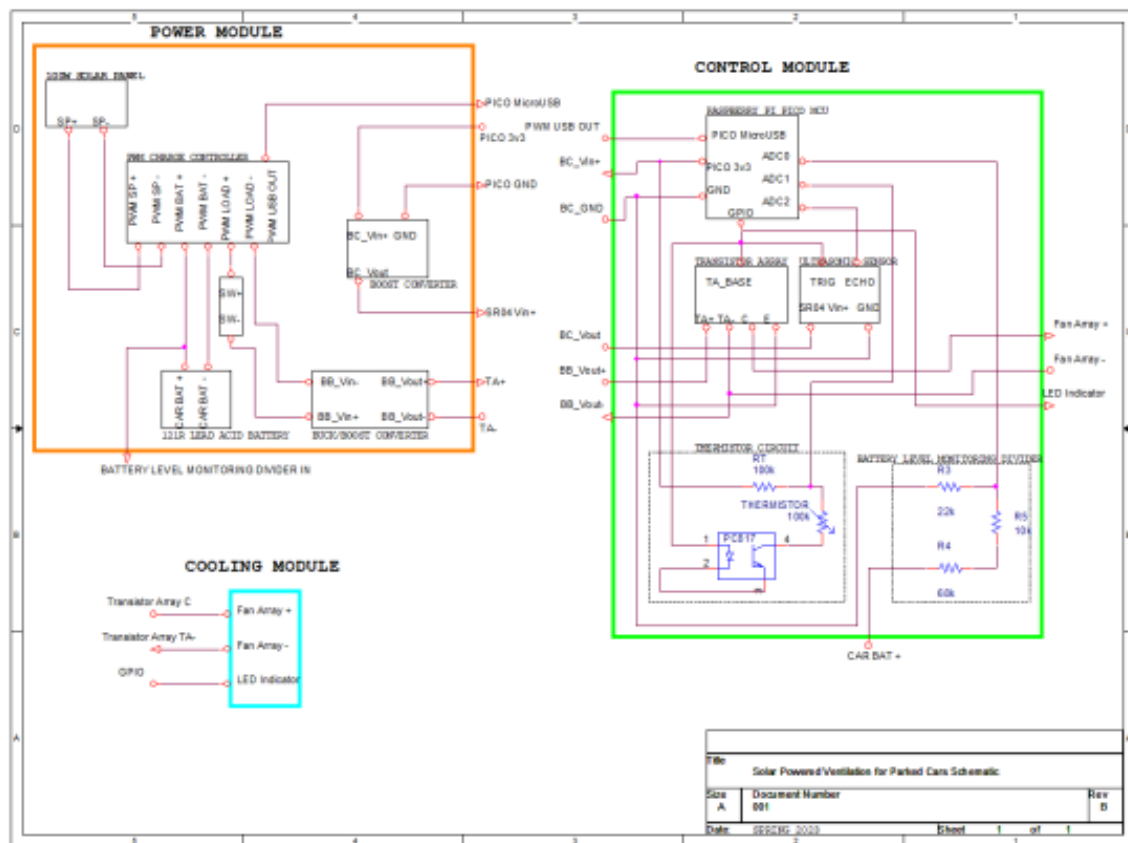


Figure 3.1 - Full System Schematic

3.2 Control Module Design

In this section, the design and build process of the control module hardware and software will be covered in detail.

3.2.1 Control Module Hardware

The Control Module is made up of the MCU, battery level monitoring divider, temperature sensors, and proximity sensors. The schematic diagram below shows how all components are connected. Note that the GPIO pin is used to signify general connections made to GPIO ports on the MCU.

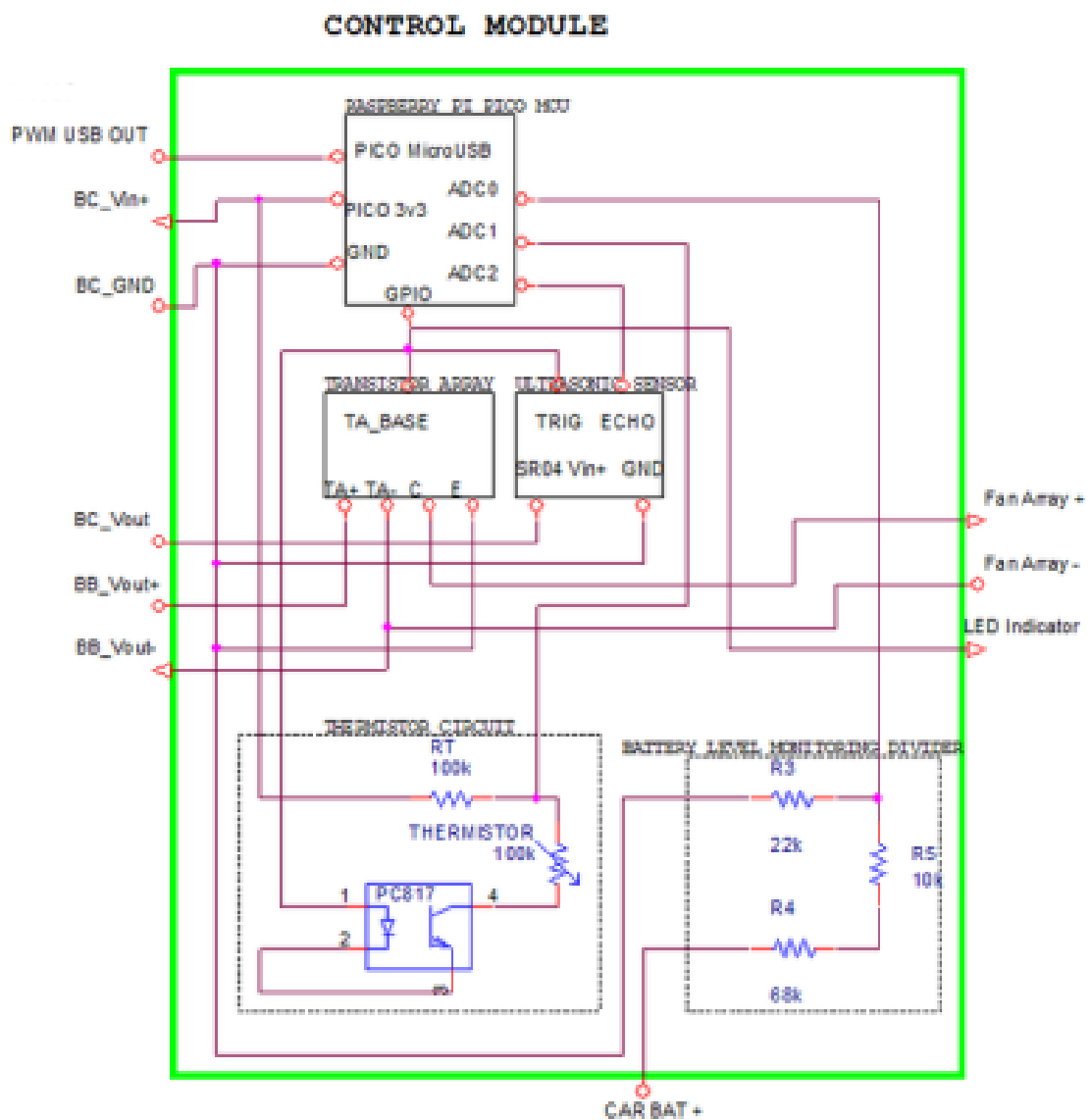


Figure 3.2 - Control Module Schematic Diagram

MCU

A Google Sheet was used to lay out all the pins and their connections for the Raspberry Pi Pico and the Adafruit PCF8575 GPIO expander. This is shown in the figure below.

	tr0	GP0	1	Raspberry Pi Pico	40		
	tr1	GP1	2		39		
		GND	3		38	GND	
	tr2	GP2	4		37		
	tr3	GP3	5		36	3V3	
	tr4	GP4	6		35		
	tr5	GP5	7		34	GP28	ADC2 Ultrasonic Echo
		GND	8		33	GND	
	tr6	GP6	9		32	GP27	ADC1 Temp Sensor
	tr7	GP7	10		31	GP26	ADC0 Battery Sensor
	tr8	GP8	11		30		
	tr9	GP9	12		29	GP22	Ultrasonic Trig
		GND	13		28	GND	
	tr10	GP10	14		27	GP21	therm4
	tr11	GP11	15		26	GP20	therm3
	tr12	GP12	16		25	GP19	therm2
	tr13	GP13	17	24	GP18	therm1	
		GND	18	23	GND		
	tr14	GP14	19	22	GP17	I2C SCL	
	tr15	GP15	20	21	GP16	I2C SDA	
		3V3	VIN	PCF8575 GPIO Expander	IRQ		
		GND	G		A0		
		GP17	SCL		A1		
		GP16	SDA		A2		
	tr16	get_pin(0)	P0		15	get_pin(15)	LED
	tr17	get_pin(1)	P1		14		
	tr18	get_pin(2)	P2		13		
	tr19	get_pin(3)	P3		12		
	tr20	get_pin(4)	P4	11			
	tr21	get_pin(5)	P5	10			
	tr22	get_pin(6)	P6	P9			
	tr23	get_pin(7)	P7	P8			

Figure 3.3 - MCU pinout

Battery Voltage Divider

The maximum voltage of the car battery is assumed to be 15 V. The battery should never actually reach a voltage this high. The battery voltage divider is designed such that the output to the ADC is 3.3 V when the input from the battery is 15 V. This is because the ADC pins on the Raspberry Pi Pico should not exceed 3.3 V to avoid the risk of damaging the board.

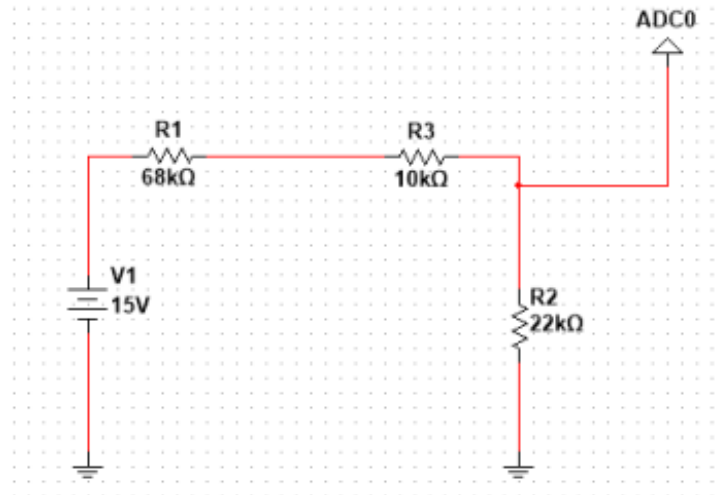


Figure 3.4 - Battery Level Monitoring Divider circuit

Thermistor Circuit

Thermistors are used for the four temperature sensors in the project. A thermistor is a semiconductor that acts like a resistor whose resistance is dependent on the temperature [36]. The resistance of the thermistor can be used to calculate the temperature.

Basic Thermistor Circuit & Temperature Calculation

The thermistors used for the sensors are NTC3950 thermistors. The values in Table 3.1 were obtained from their datasheet [37].

Table 3.1 - Given values from the NTC3950 datasheet.

Given	Description
$T_0 = 298.15\text{ K}$	Thermistor zero power temperature.
$R_0 = 100\text{ K}\Omega$	Thermistor zero power resistance at temperature T_0 .
$\beta = 3950\text{ K}$	Thermistor β parameter.

To determine resistance, the thermistor is first placed in a voltage divider circuit as shown in Figure 3.5. R_k is selected to match R_0 so that the ADC voltage will be halved at 25 °C. The voltage seen by the MCU's ADC is then used to calculate the resistance of the thermistor R_{Th} using the voltage divider equation in (3.1).

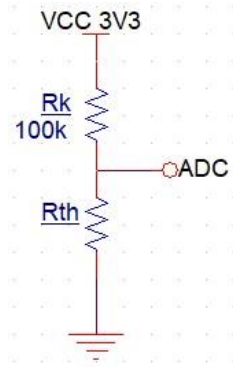


Figure 3.5 - Thermistor divider circuit

Voltage Divider Equation
$$V_{ADC} = \frac{V_{CC}R_{Th}}{Rk + R_{Th}} \quad (3.1)$$

Voltage Divider Solved for
$$R_{Th} = \frac{V_{ADC}Rk}{V_{CC} - V_{ADC}} \quad (3.2)$$

 R_{Th}

Since the β parameter has been given, the β parameter equation can be used to solve for the temperature, T [38]. This equation is derived from the Steinhart-Hart equation, which is a widely used third-order approximation of the thermistor transfer function relating temperature and resistance.

β Parameter Equation
$$R_{Th} = R_0 e^{\left[\beta\left(\frac{1}{T} - \frac{1}{T_0}\right)\right]} \quad (3.3)$$

β Parameter Equation Solved
$$T = \frac{1}{\frac{\ln\left(\frac{R_{Th}}{R_0}\right)}{\beta} + \frac{1}{T_0}} \quad (3.4)$$

for T

The resulting T value is in Kelvin, so 273.15 is subtracted from it to get the temperature in Celsius, then converted to Fahrenheit in (3.6).

Kelvin to Celsius
$$T_C = T - 273.15 \quad (3.5)$$

Celsius to Fahrenheit
$$T_F = \frac{9}{5}T_C + 32 \quad (3.6)$$

Ultrasonic Proximity Sensors

The HC-SR04 requires 5V for the Vcc, so a boost converter was required. The Echo pin is also 5V, so a voltage divider was used to cut the voltage in half at the ADC.

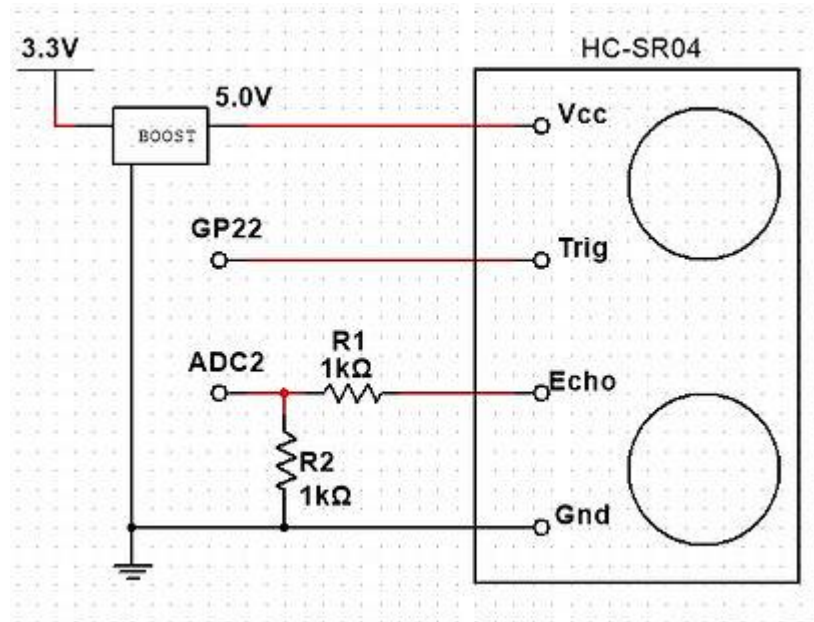


Figure 3.6 – HC-SR04 Wiring Diagram

Green Indicator LED

Current limiting resistance is required to drive the green LED. Green LEDs typically have a forward voltage of 2.0 V and a rated forward current of about 20 mA. With a supply of 3.3 V, Ohm's law is used to calculate the minimum resistance value required to limit the current. With no 65 Ω resistors or anything close on hand, a parallel combination of three 220 Ω resistors creates an equivalent resistance of about 73 Ω , which is suitable for driving a green LED.

Ohm's Law
$$R_L = \frac{3.3\text{ V} - 2.0\text{ V}}{0.02\text{ mA}} = 65\ \Omega \quad (3.7)$$

Parallel Resistance
$$R_{LED} = \left(\frac{3}{220}\right)^{-1} = 73.333\ \Omega \quad (3.8)$$

Prototype Planning

The control module has a lot of connections, so proper planning was crucial. Connection diagrams were made in Google Sheets. The group did not have large prototyping boards to work with, so the control module was done in sections and each section was planned carefully. This method made creation and modification of the diagrams simple and quick. The diagrams were mirrored and printed out so that wiring could be drawn out by hand.

The MCU connection diagram is shown below. It includes the Raspberry Pi Pico and components for the battery voltage divider, temperature sensors, and ultrasonic proximity sensor.



Figure 3.7 - Prototyping board connection diagram for MCU and sensor components.

The PCF8575 GPIO expander was placed on a separate prototyping board since there was not enough room on the MCU board. The components for the green indicator LED were also made on this prototyping board.



Figure 3.8 - Prototyping board connection diagram for PCF8575.

The connection diagram for the transistor array is shown below. It includes flyback diodes at the collector and current limiting resistors at the base of each transistor.

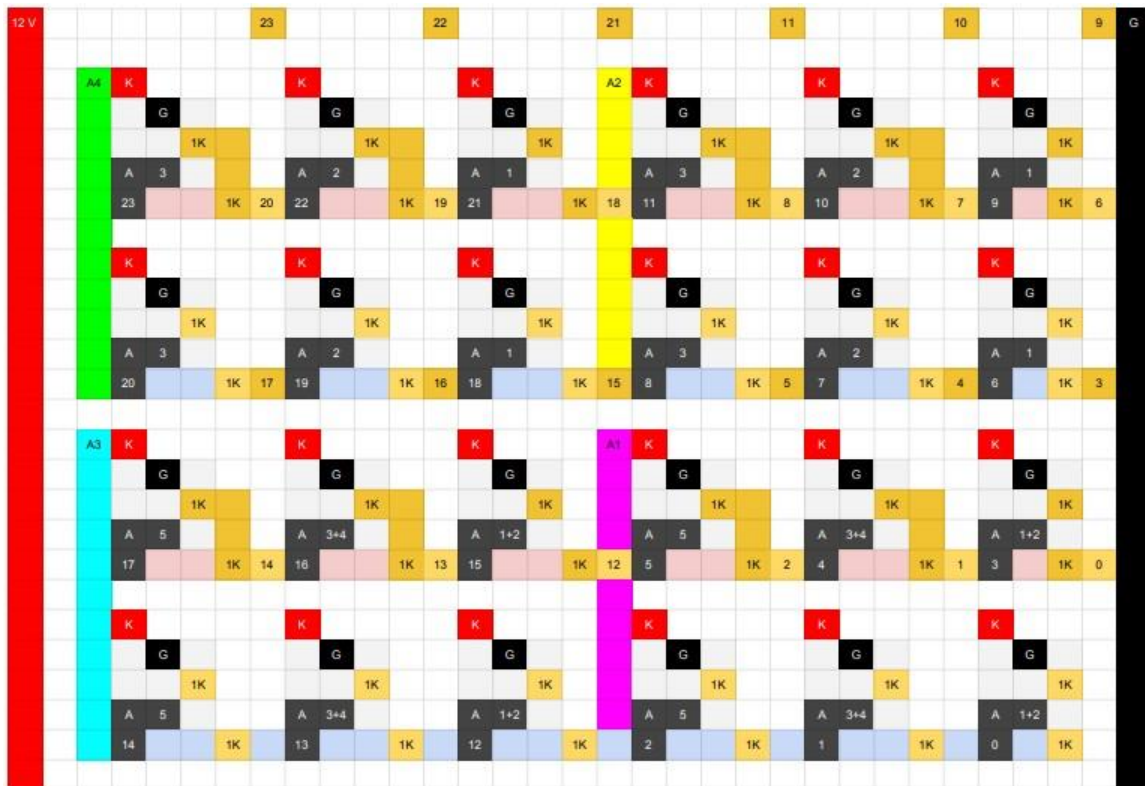


Figure 3.9 - Connection diagram for the transistor array.

To determine the connectors, terminal blocks, and cabling required by the project, the wiring diagram below was drafted. It shows how all connections from the windows will reach the control module and the number of pins required for each terminal block.

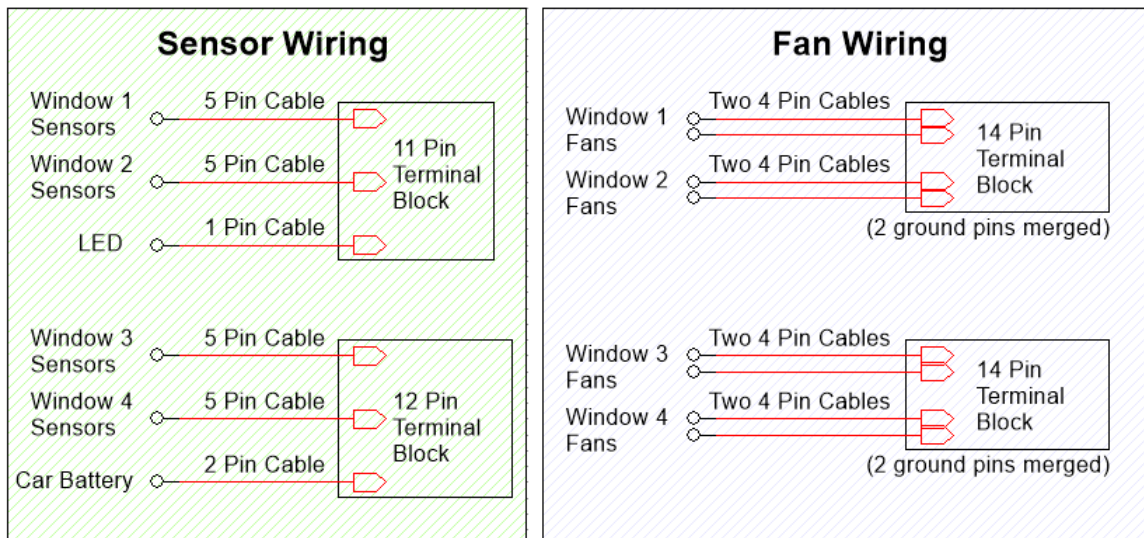


Figure 3.10 - System wiring diagram for determining required connectors.

Prototype Wiring and Soldering

The following section details the build process of the control module.

MCU and Sensors

Using the connection diagram, the MCU and all sensor components were soldered to the prototyping board.

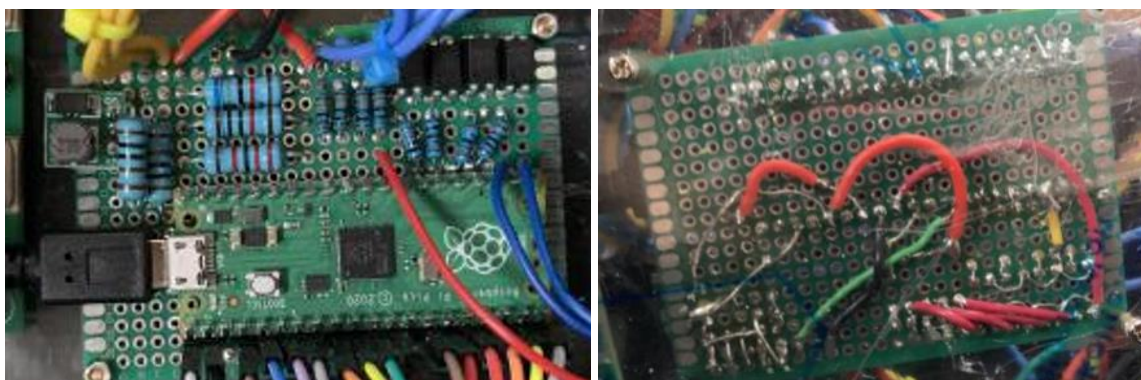


Figure 3.11 - Front and back of MCU and Sensor Prototype Circuit

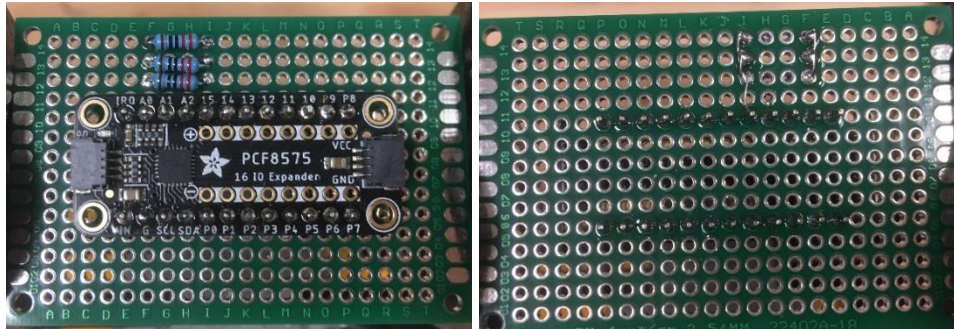


Figure 3.12 - Soldered PCF8575 Prototype Circuit

Wiring the Transistor Array

To wire the transistor array, a larger prototype board was used. The array includes the transistors, flyback diodes, and current limiting resistors at each base. All connections are made on the back of the board.

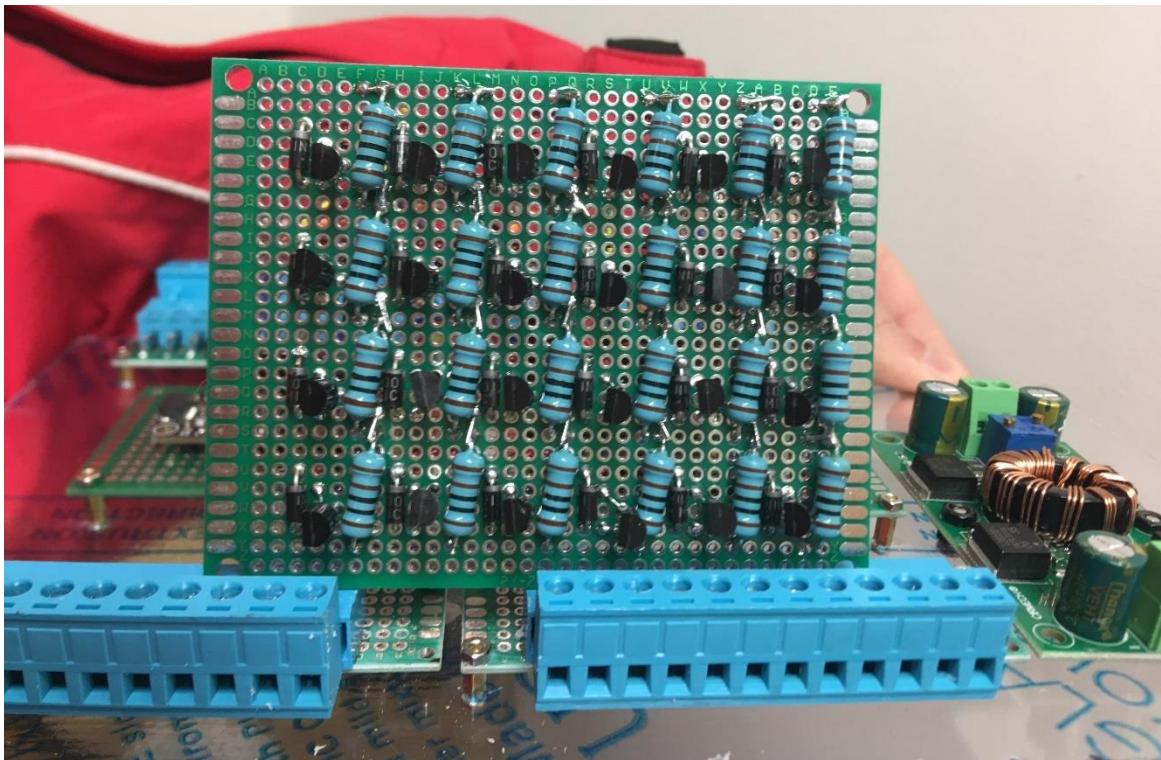


Figure 3.13 - Front of the Transistor Array prototyping board.

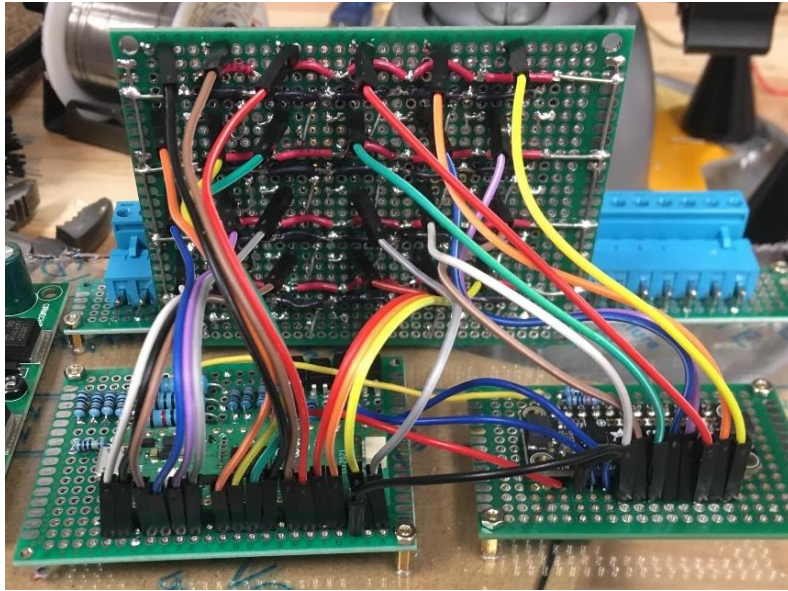


Figure 3.14 - Back of the Transistor Array Prototyping Board.

Terminal Block Prototyping Boards

Unfortunately, the pins on the terminal blocks were too large for the prototyping boards. To get around this, the terminal blocks were epoxied to the boards. The pins were jumped and soldered to the boards using some spare resistor leads.

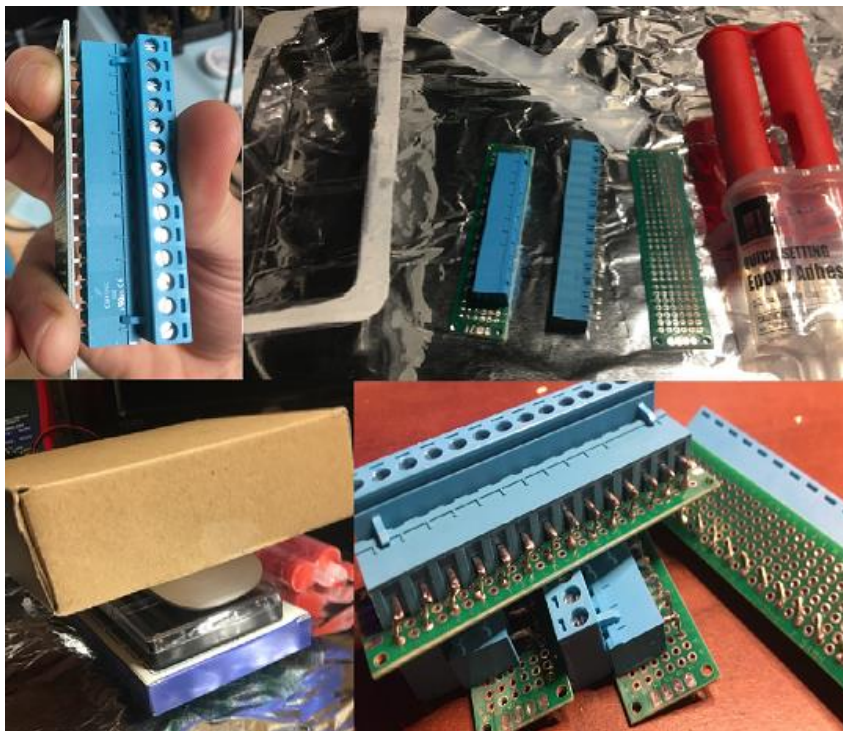


Figure 3.15 - Building and soldering the terminal block prototyping boards.

4-Pin and 5-Pin IP65 Cable Connections

The IP67 and IP65 connectors came separate from their respective cables. Each length of cable was measured and cut at about 6.25 ft. The individual leads were soldered and wrapped using heat-shrink wrap. Another layer of heat-shrink wrap was applied to the cable itself to protect the bare leads.








Figure 3.16 - 5-Pin IP65 connector connections.

Terminal Block Wiring and Control Box Build

Wire color coding for the transistor array was done in rainbow order, such that yellow corresponds to transistor 0, green to transistor 1, blue to transistor 3, etc. Zip-ties were used to keep transistor groupings organized and visible. Connections from the sensor components to the sensor terminal blocks were made using the color code in the chart below. The completed wiring is shown in *Figures 3.17 and 3.18*.

Table 3.2 - Sensor Wire Color Code

	GROUND
	ULTRASONIC ECHO
	ULTRASONIC VCC
	ULTRASONIC TRIG
	THERM +

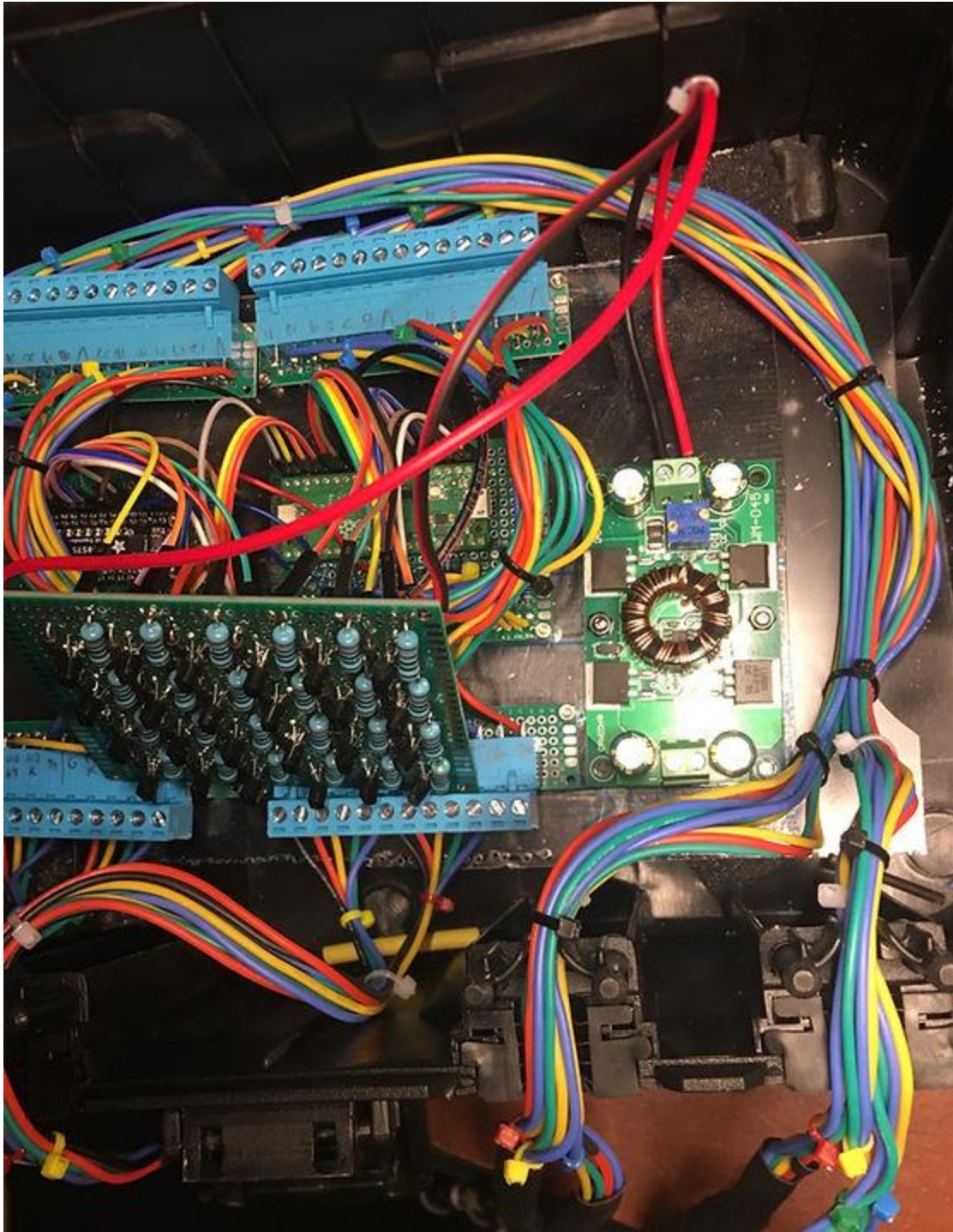


Figure 3.17 - Wiring harnesses from the terminal blocks.

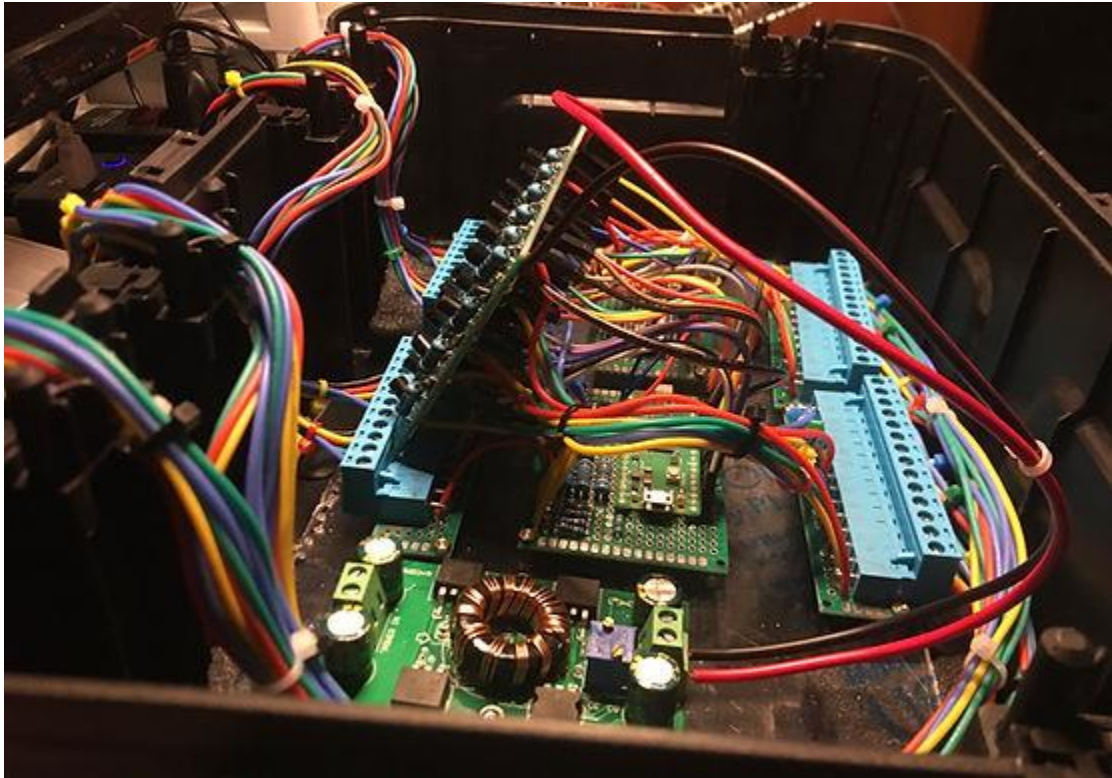


Figure 3.18 - Fully wired control module.

The PWM charge controller was installed in the lid of the control box. The SPDT ON/OFF switch was also installed and epoxied in place. Wires were soldered to the switch for the positive load connection.



Figure 3.19 - Installation of PWM charge controller and ON/OFF switch in the lid of the control box.

Finally, the IP65 connectors were soldered to the freshly created wiring harnesses. These allow the control box to be easily disconnected from the fans. With this, the control box build was complete.



Figure 3.20 - Closed control box.

3.2.2 Control Module Software

The software module will control the cooling module by changing the number of active fans as well as the airflow direction based on information gathered from the sensors listed in section 2.3.1. The program executes five functions in sequence: Get Battery Mode, Get Window, Get Window Temperatures, Get Cooling Mode, and Fan Control. These functions make up the Main Loop.

Main Loop

When provided power, the MCU automatically runs the main loop. It first calls the Get Battery Mode function, which reads the voltage of the car battery and returns a battery mode based on that voltage. Next, Get Window is called, which determines and returns the window position using the connected ultrasonic sensors. Get Window Temperatures then reads the temperature at each window and returns them as a list. Get Cooling Mode compares these temperatures to determine the cooling mode which will most effectively cool the car. Finally, Fan Control takes the battery mode, window position, and cooling mode, and turns specified fans ON or OFF.

Once the main functions are complete, the Main Loop toggles the MCU's onboard LED, waits for 10 minutes, and then checks for OS Error. If no error is caught, then the main loop repeats. If an error is caught and the file system is full, then the onboard LED blinks slowly. If the file system is not full, then the onboard LED blinks quickly. The flowchart for the main loop is given below.

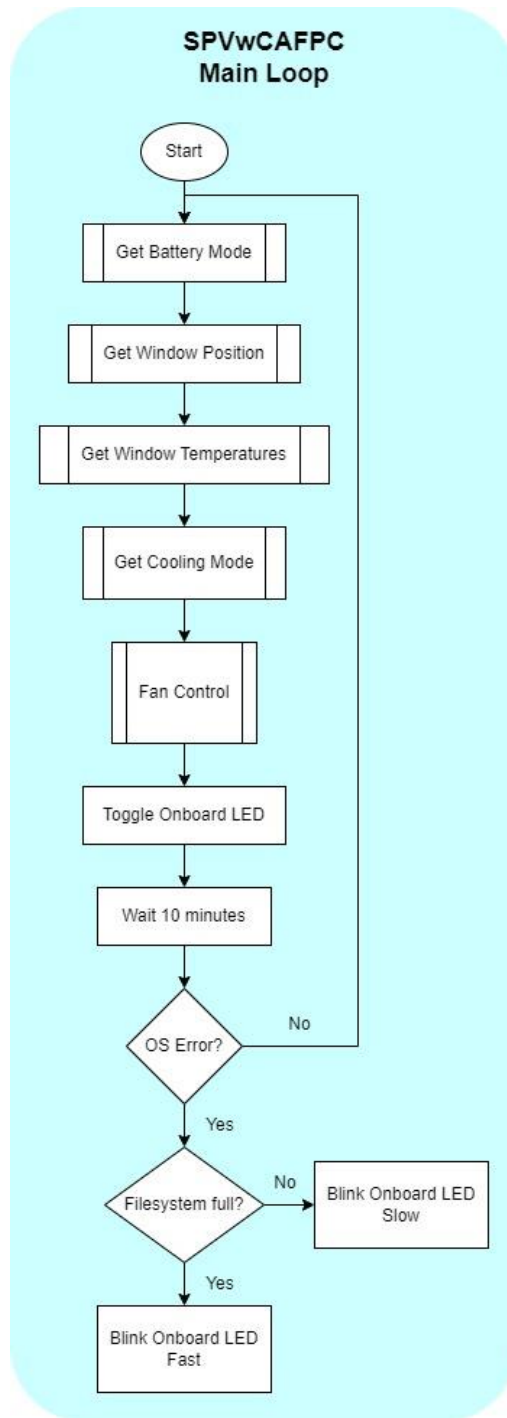


Figure 3.21 - Main Loop Flowchart

Get Battery Mode

The first time the main loop runs, two variables are assigned to zero before Get Battery Mode is called: V_{prev} and bat_prev . V_{prev} represents the ADC input voltage from the previous iteration of the main loop. bat_prev represents the battery mode from the previous iteration. Entering the function, the MCU reads the raw value at the ADC, calculates the voltage, and assigns it to V_{in} . The Raspberry Pi Pico has a 12-bit ADC. However, this is scaled up to 16 bits to comply with CircuitPython's API [39]. Therefore, the following equation can be used to convert the raw ADC value to a voltage.

$$\text{Conversion from Raw Value to Voltage for the Pico ADC} \quad V_{ADC} = raw * \frac{3.3}{65535} \quad (3.9)$$

Next, the function calculates the voltage of the battery from V_{in} . The calculated value is used strictly for monitoring purposes, whereas V_{in} is the actual voltage seen at the ADC to be used for comparison to the battery mode thresholds. Next, if the previous battery mode was equal to zero, then the lower threshold is set to 12.15 V. If not, then the lower threshold is set to 12.00 V.

The following procedure acts as a sort of hysteresis to keep the battery mode from bouncing back and forth rapidly when the voltage at the ADC is close to one of the thresholds. First, an if statement checks for at least a 50 mV difference between the input voltage and the previous voltage. If this condition is satisfied, then the battery mode is switched based on the thresholds in *Table 3.3*. If not, then the battery mode is set equal to the previous battery mode. If the previous battery mode and the current battery mode are equivalent, then V_{prev} is reset. Before returning the battery mode, the program sets the previous battery mode equal to the current battery mode. It also writes the calculated battery voltage to the datalog file.

Table 3.3 – Battery voltages and their respective ADC voltages and battery modes.

Battery Voltage	ADC Voltage	Battery Mode	Battery Mode Name
12.6	2.773	3	HIGH
12.3	2.707	2	MED
12.0	2.641	1	LOW

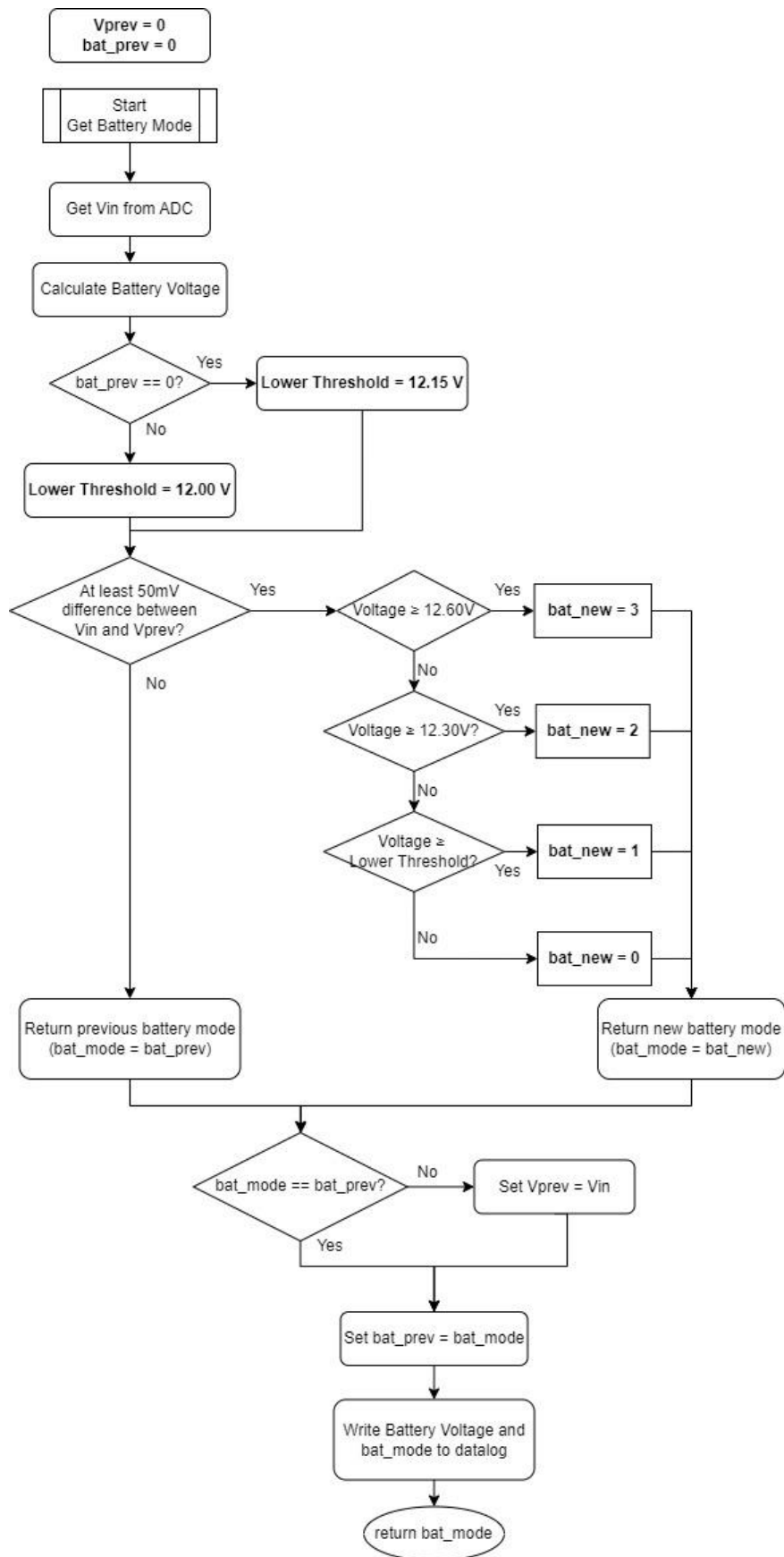


Figure 3.22 – Flowchart for the Get Battery Mode function.

Get Window Position

The flowchart below shows how the Get Window Position function works. After setting the distance variable to 0, the MCU sends a 10us pulse to the HC-SR04. Immediately after, the MCU reads the ADC connected to the HC-SR04 Echo pin every microsecond for a specified number of repetitions. Once 150 repetitions have occurred, the program returns the average distance value and compares it to a calibrated distance threshold of 25000. If the distance is less than this value, then the window is up and the green indicator LED on the window 1 vent visor is turned OFF. Otherwise, the window is down and the green LED is turned ON. Finally, the window position is written to the datalog and returned by the function.

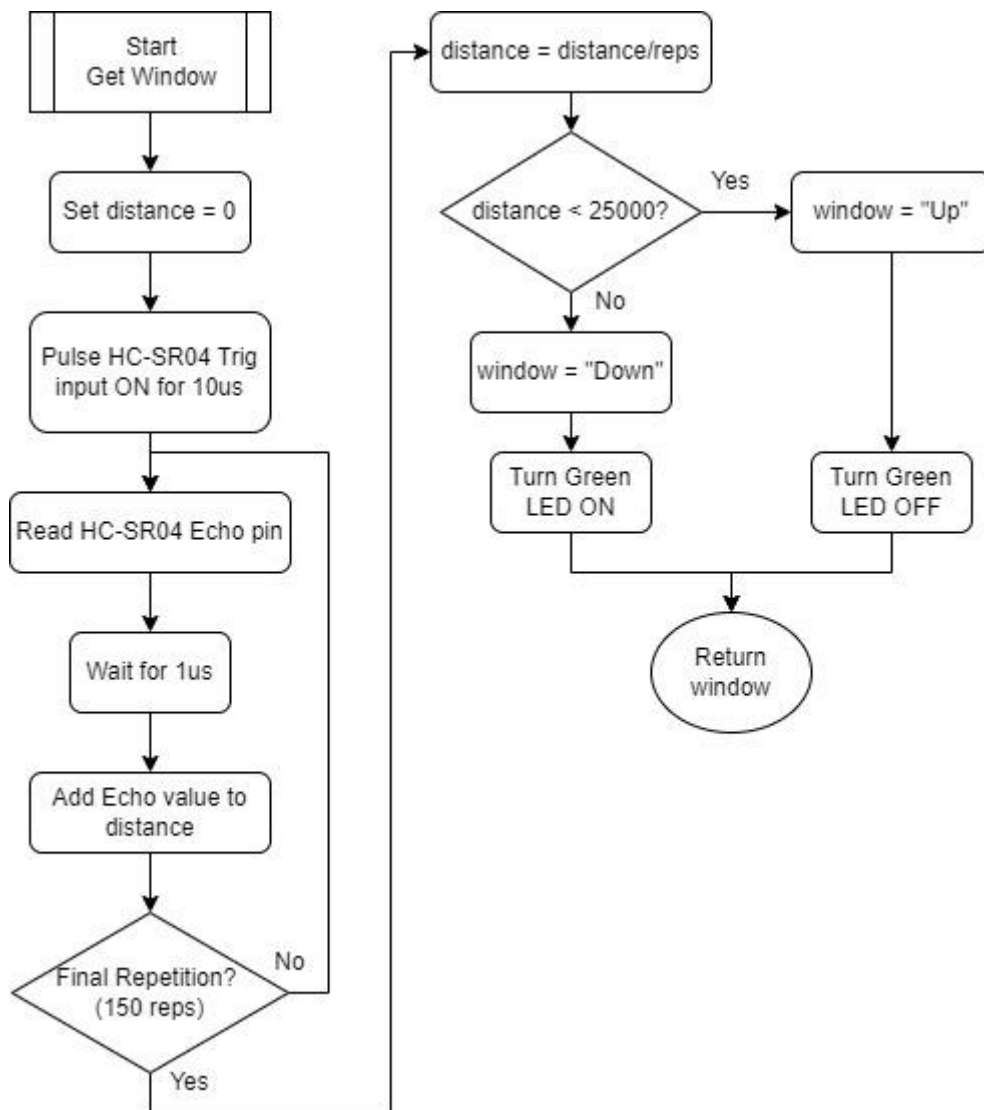


Figure 3.23 - Flowchart for Get Window Position function.

Get Window Temperatures

There are four thermistor circuits, one for each window. The Get Window Temperatures function enables one thermistor, reads the voltage at the ADC, disables the thermistor, calculates the temperature from the ADC voltage, and repeats for each of the four thermistor circuits. Then, it writes all four temperatures to the datalog and returns them.

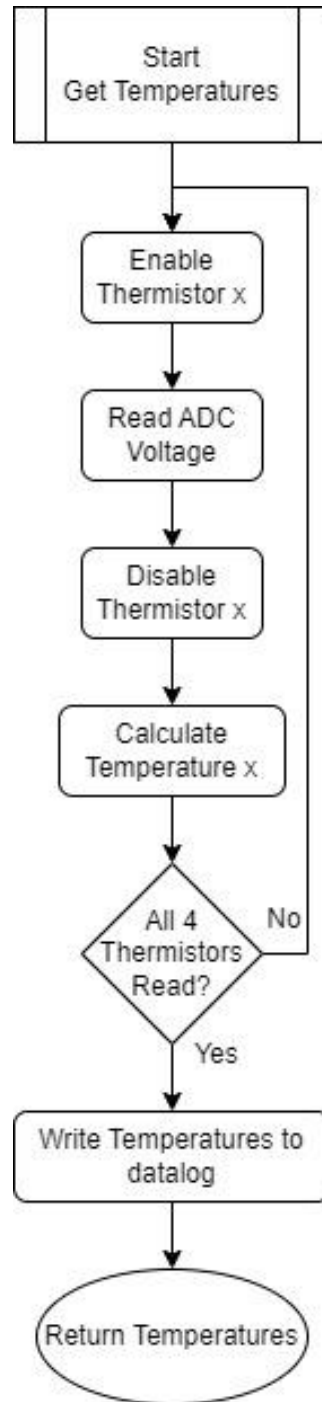


Figure 3.24 - Flowchart for Get Window Temperatures function

Get Cooling Mode

With all sensor data collected, the MCU moves on to determining the optimal cooling mode. Get Cooling Mode, like Get Battery Mode, has hysteresis built in to prevent rapid switching near thresholds. Two global variables hold the previous mode and previous temperature standard deviation. At the start of the function, the new temperature standard deviation is taken. If the difference between the new and previous deviation is greater than 3, then a cooling mode is selected. The previous mode is then set to the selected mode, and the previous temperature deviation is set to the new temperature deviation. If the condition is not satisfied, then the mode returned by the function is set as equivalent to the previous mode.

The cooling mode is decided by comparing the sums of the temperatures on the four sides of the car. The driver side is defined as the sum of temperatures at windows 1 and 2. The passenger side is the sum of temperatures at windows 3 and 4. The front temperature is the sum of temperatures at windows 1 and 3, while the back is the sum of temperatures at windows 2 and 4. If the difference between the front and back is less than or equal to the difference between driver and passenger, then mode 1 or 2 is selected. Mode 1 is selected if the driver's side is cooler than the passenger side. Once a mode is selected, it is written to the datalog and then returned by the function.

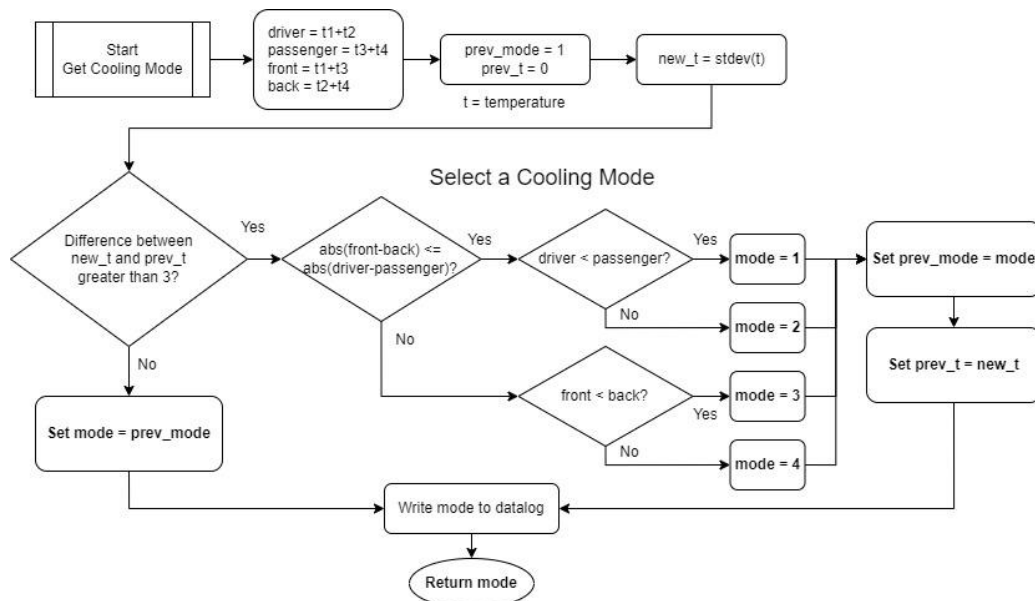


Figure 3.25 - Get Cooling Mode

Fan Control

The Fan Control function acts on the sensor data and selected cooling mode. It does this by turning fans on and off per the cooling modes. For example, if the windows are up, then all fans are turned OFF. For cooling mode 1, intake fans on the driver side are turned on while exhaust fans on the driver side are turned off, and the reverse is true on the passenger side. The battery modes come into play at the end of the function, turning off a large group or small group of fans depending on the battery mode.

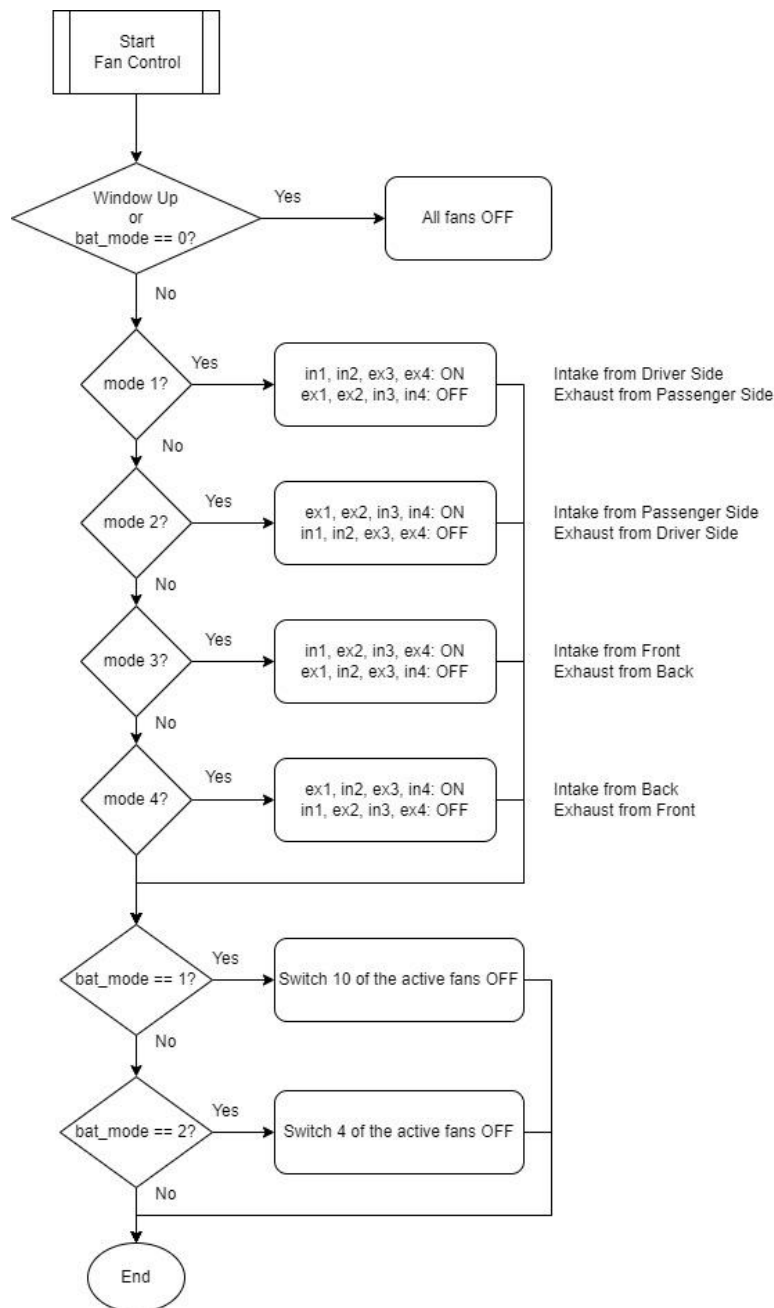


Figure 3.26 - Flowchart for Fan Control function

3.3 Cooling Module Design

In this section, we detail the design and construction process for the cooling module.

3.3.1 Cooling Module Hardware

Vent Visors

To properly mount a ventilation system to the vehicle without needlessly damaging or modifying the body of the vehicle, an aftermarket set of OEM U8220 2k000 Vent Visors was ordered from eBay. These are made of durable cast acrylic which holds up to both rough handling and the drilling of holes required to mount our ventilation array. The visors are designed to allow for passive ventilation of the car without the risk of rain entering the vehicle thus helping us meet our requirements for the project. As an added benefit, they are also purported to reduce sun glare and reduce wind noise while driving. The specific model, OEM U8220 2k000 was chosen due to its large offset from the body of the car which would provide adequate space for our fan array, nozzles, and hardware.

Nuts, Bolts, and Spacers

To mount the fans within the Vent Visors, a series of holes needed to be drilled along the length of the visors and as evenly spaced as possible while also attempting to leave room for our sensors and wiring, the front vent visors can hold a total of 10 fans each and the back vent visors can hold 6 fans each. Fans are attached via a set of 3M 25mm screws and 3M Hex Nuts. Half of the fans in each vent visor are oriented such that the intake of the centrifugal blower faces the vent visor. In between the vent visor and fans is a small air gap provided by a set of spacers within the hardware stack between the fan and vent visor.



Figure 3.27 - Placement of DC Blowers on Vent Visors

Centrifugal DC Blowers

The DC Blowers were chosen due to their relatively slim profile as compared to other options and an overall airflow rate of 2.4 CFM per fan. With 16 fans active at any one time, this means that the air within the vehicle, with an interior volume of 102 ft³, would be replaced every 2.65 minutes and brings the mass flow rate of the array to 0.056kg/s.



Figure 3.28 - DC Blower Array Attached to Car

As stated in chapter 1, we are basing our device on the assumption that it can reduce the internal temperature of the vehicle to reduce the initial heat load placed upon the AC System. Calculations for the airflow required to perform this task are shown below based on the heat load requirements for our system as a function of the mass flow rate of our system.

For a vehicle using our device, we can calculate the required CFM to meet our 130% specification by using the equation for heat load, equation 3.10. For this, m becomes the mass flow rate of our system flow rate of 0.056kg/s.

Equation for Cooling Heat Load [40]
$$H_c = m(C_p)(\Delta T) \quad (3.10)$$

In the Worst-Case Scenario, the internal temperature reaches 130% of Ambient or 45.5°C. Therefore, the power required to reduce the internal temperature to 123.5°F for our system is as follows:

$$H_c = (0.056 \text{ kg/s}) \left(1.005 \frac{\text{kJ}}{\text{kgK}} \right) (57.222^\circ\text{C} - 45.5^\circ\text{C}) = 0.6597 \text{ kW}$$

Using this information, we can calculate the airflow required for the system to perform its task. For this, we use the equation for calculating the air flow rate required for ventilation cooling based on heat load where q_c represents the quantity of airflow required for ventilation air cooling, H_c represents the Heat Load of the system, ρ represents the density of air, C_p representing the specific heat capacity of air, t_o representing the outlet temperature of the cooling fans, and t_r representing the temperature of the space being cooled.

$$\begin{array}{l} \text{Equation for Quantity of Air} \\ \text{Required for Air Cooling [41]} \end{array} \quad q_c = \frac{H_c}{\rho C_p (t_r - t_o)} \quad (3.11)$$

To meet our specification of 130% we apply our heat load of 0.6597kW to the equation to determine the airflow rate required. For the inlet temperature, we will use the average air temperature for a day wherein the high reaches 95°F. Using data from June of 2022 with a monthly high of 98°F and an average ambient temperature of 83°F [42], we will assume the average temperature of the inlet of the fans is 83°F (28.333°C) throughout a hot Summer day.

$$q_c = \frac{(0.6597kW)}{\left(\frac{1.293kg}{m^3}\right)\left(\frac{1.005kJ}{kg.K}\right)(57.222^\circ C - 28.333^\circ C)} = 0.01757 \frac{m^3}{s}$$

Converting the values from m^3/s to CFM, we get 37.23 CFM required to meet our 130% specification. As each of our fans provides 2.4 CFM per fan, and at any time a total of 16 fans are active, we can clock in our total CFM at 38.4 CFM meeting this requirement for a 95°F day. It should be noted that losses associated with the nozzles in the next section are possible as well as increased ventilation being provided passively by the cracking of the windows themselves for the ventilation system to operate thus helping or hindering the ability to meet our requirements respectively.

Nozzle Design

Nozzles were designed for the intake fans. A rough sketch was drafted, as shown below. The idea was to have the nozzle fit on the inner portion of the fan outlet. The nozzle would have a lip over the edge of the fan outlet to provide a channel for epoxy. The dimensions of the fan outlet were measured using a set of calipers.

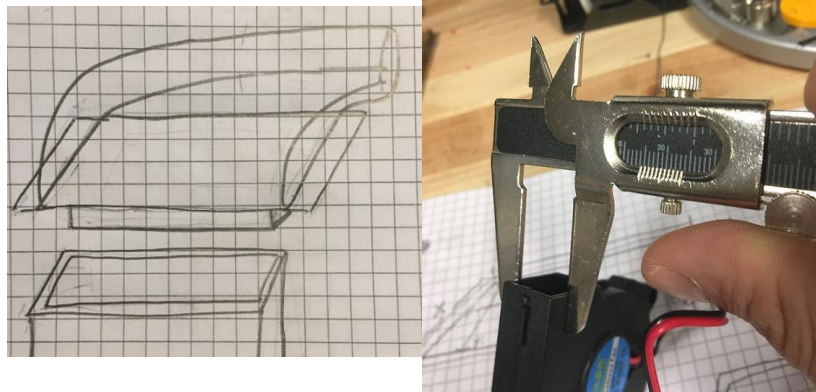


Figure 3.29 - Nozzle design draft and measurement

First Version

Once the dimensions were measured, the nozzle was sketched in Autodesk Fusion 360, which is free for students at Valencia College. The first version is shown below. On the bottom, there is an inner lip and an outer lip. The inner lip fits into the fan outlet, while the outer lip fits over the fan outlet and creates a channel for epoxy.



Figure 3.30 - First revision of nozzle design

Once the model was finished, Ian consulted his friend Duncan Kurtz for advice on 3D printing. Duncan owned a 3D printer and offered to print a prototype. Ian exported an STL file from Fusion 360 and sent it to Duncan. Duncan's first attempt used the default support structures auto-generated by Cura Slicer. This failed since the supports extended into the nozzle and they were impossible to remove.

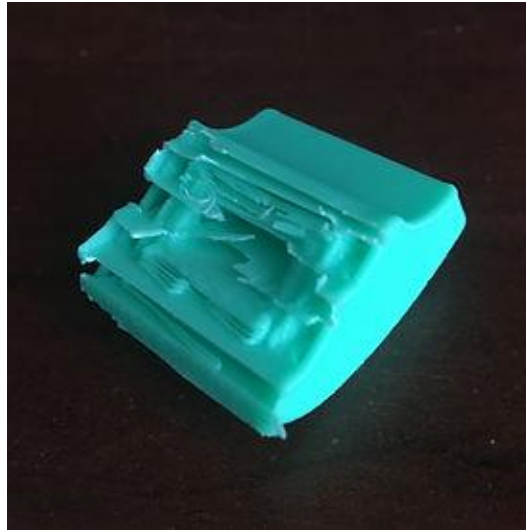


Figure 3.31 - Failed nozzle print.

For his next attempt, Duncan used tree supports instead of default supports. Tree supports are a lighter form of support, and they are easier to remove from the object. This resulted in the image below, where supports were printed in the epoxy channel and proved difficult to remove.

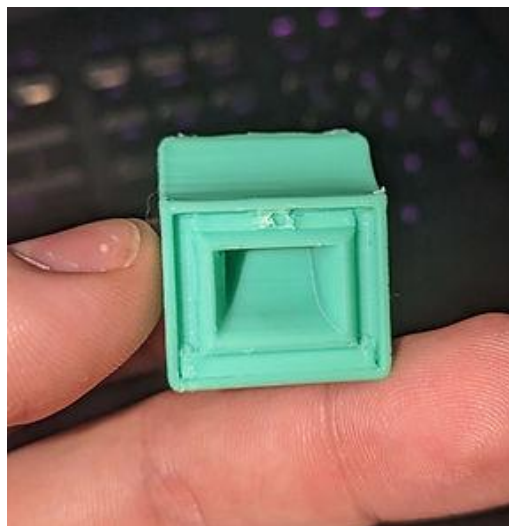
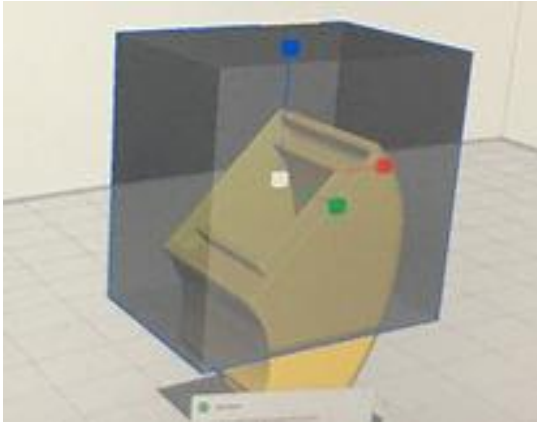


Figure 3.32 - Nozzle with support printed in the epoxy channel.

For Duncan's final adjustment, he rotated the model and placed a support blocker over the nozzle's inlet, intuiting that the inlet would be strong enough without support. Tree supports would be printed only at the nozzle's outlet. These tactics resulted in an excellent print, shown below.



(a)



(b)

Figure 3.33 - (a) Cura Slicer nozzle render with support blocker. (b) Clean print of the first revision of the nozzle.

The nozzles fit very well on the fans, but they had a few issues. The walls of the inlet were quite thick, restricting the airflow considerably. The nozzles also got dangerously close to touching the window when placed on the car.



Figure 3.34 – First Revision Nozzle Minimal Clearance

Second Version

To resolve these issues, the length of the nozzle was shortened by a millimeter and the inlet and outlet walls were made thinner. Now the inner lip would be only a millimeter thick. This design is shown below.



Figure 3.35 - Second Revision of the Nozzle.

The second version was printed as shown below. This version had far better airflow without compromising structural integrity. It was decided that these nozzles would be suitable for the project, and the full batch of nozzles was printed.

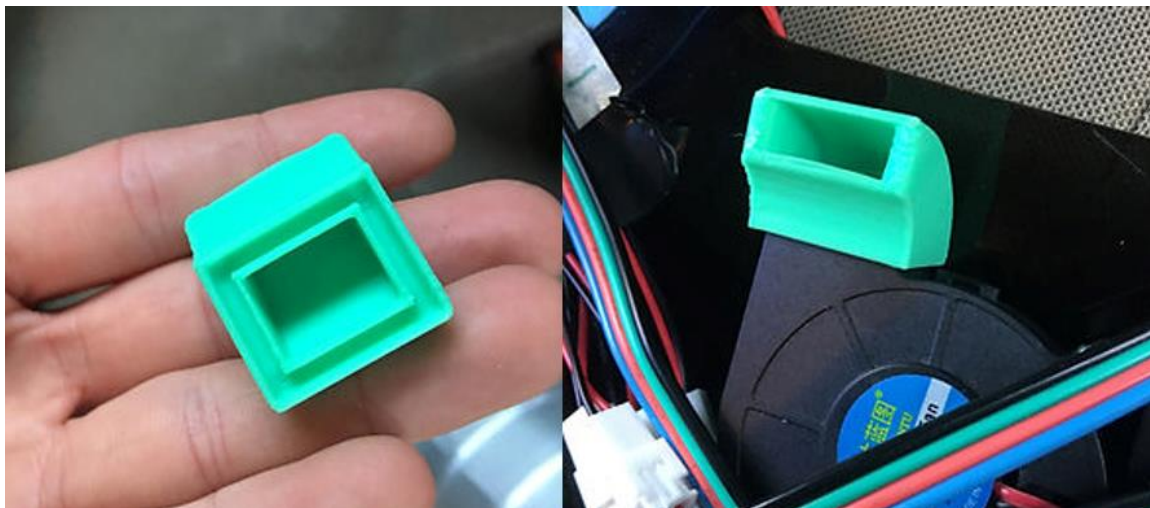


Figure 3.36 - Print of Second Nozzle Revision

Reorientation of Fans

Because the back window vent visors do not provide as much coverage as the front, the nozzles stuck out. This was worrisome for rain since the window would have to be opened lower than the vent visor to allow for proper airflow. Thus, the fans were moved and reoriented such that the nozzles were on top. This required a third version of the nozzle. The nozzles had to be squished down so they wouldn't be obstructed by the lip between the window and the car door.

Third Version

The following model was created in Fusion 360 based on the previous design. Since these nozzles are for the back windows, only six of them were needed. The nozzles were printed and installed on the fans.

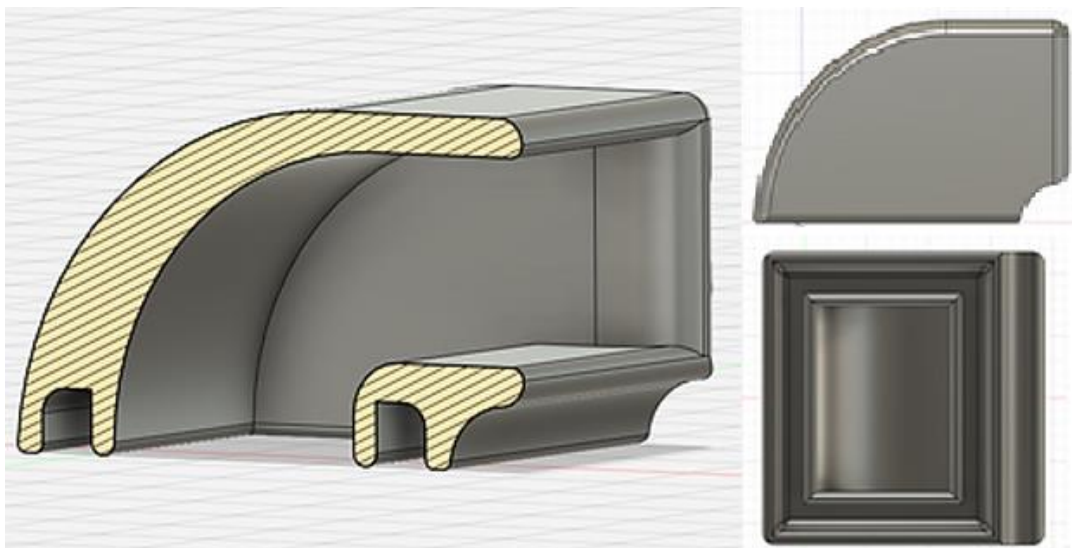


Figure 3.37 - Third version of the nozzle design.

3.3.2 Vehicle Implementation and Issues

After loosely securing the fans to the vent visors using the mounting hardware, wiring connections were made using sets of male 2-pin connectors and 4-pin IP67 Cables. The positive leads of the 2-pin connections were soldered to individual positive connections from the IP67 cables for the back windows and doubled up in pairs for 2 groups in the front windows. As each of the fan groups had a common ground, only 4 pins were necessary for power to be provided to the fans in each vent visor.

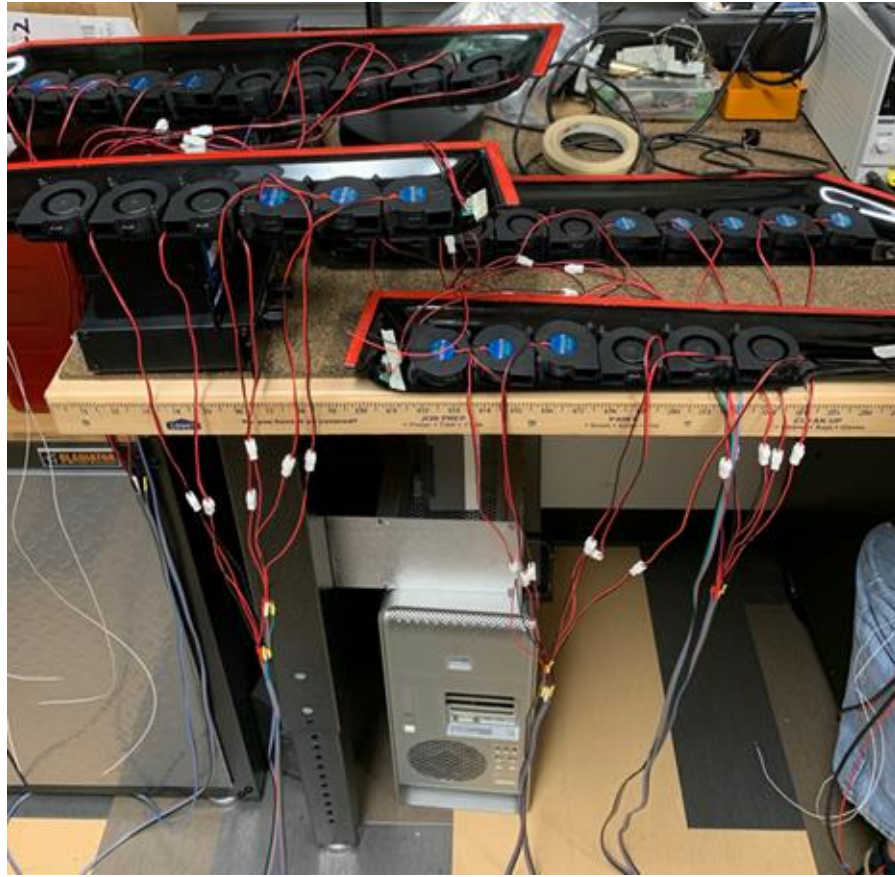


Figure 3.38 - Build-up with IP65 Cable Connections

Due to the need to drive with the device attached to the Test Vehicle, a need to reduce the risk of hardware becoming loose due to excess vibrations was needed. For this, a Blue 242 Loctite Threadlocker was applied to each of the bolts and nuts used to hold the fans in place. This Threadlocker was chosen because it is removable from the mounting hardware. Following the application of Threadlocker, the gaps between the rear of the hex bolts and vent visors were sealed using a clear waterproofing epoxy to ensure rain was unable to enter the vent visor through the holes made.

During the final assembly, when all sensors and fans had been connected through sets of IP65 and IP67 cables on the vent visors, the nuts used to keep the fans in place were torqued down with Loctite applied. Zip-ties were used for wire management. It was noticed upon attaching the vent visors to the vehicle that hairline cracks had begun to form in the Acrylic, and some sections had begun to fracture entirely. Furthermore, one of the connections between our fans and the Control Module had come undone due to over-extension and needed resoldering.



Figure 3.39 - Torque Damage Incurred by Vent Visors and Wire Break

The cause of the cracking was somewhat apparent. During our previous testing and integration of sensors to the vent visors, they had all been attached loosely to the device via clips, tape, or loose hardware. After tightening the bolts, there was an imbalance in the forces applied to the rigid acrylic causing the material to flex and bend in a way it was not meant to. At the points where we had drilled, we had created points of weakness in the material as well from which the bending caused cracks to form and splinter across the vent visors.

Given the potential for catastrophic failure if the device was on the car without any repairs, a combination of the same clear epoxy used to cover the gaps between the bolts and visors was used as well as Loctite Epoxy for some of the largest gaps on the vent visors. In addition, an outer layer of 3M Adhesive Spray was applied to the vent visors. Aesthetically, they are an abomination as of now, but they are sturdy once again.



Figure 3.40 -3M Adhesive Spray Applied to Vent Visors with Paint

Aesthetically, vent visors had an off-yellow hue due to the application of the 3M Adhesive Spray. To counter this, a layer of black paint was applied to each of the vent visors. Although not critical, it did improve the overall appearance of the visors.

3.4 Power Module Design

3.4.1 Main Design

Power to our System is provided by the Power Module. The Power Module, as stated in Chapter 2, consists primarily of the 100W Solar Panel, PWM Charge Controller, Battery, switch, and Voltage Regulators.

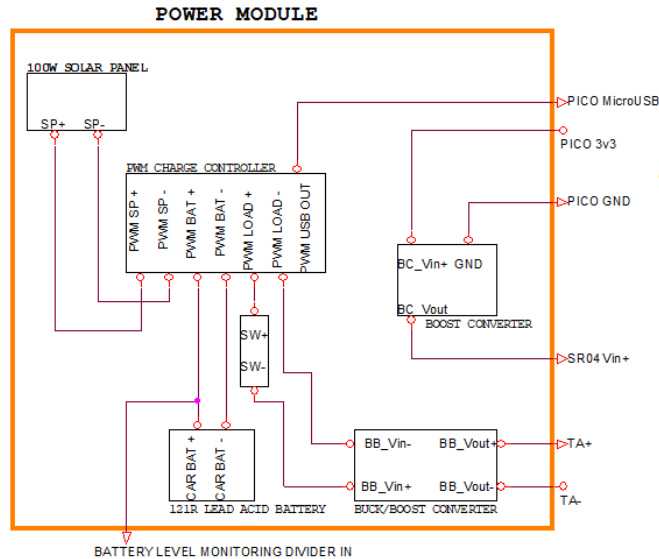


Figure 3.41 – Power Module Schematic Diagram

The Battery of the vehicle is first wired into the PWM Charge Controller via the appropriate BAT+ and BAT- terminals. The PWM Controller monitors the voltage of the battery and allows the user to set parameters for maximum float voltage and minimum load cut-off voltage. Next, the Solar Panel is wired to PWM Charge Controller to the appropriate SP+ and SP- terminals. If the solar panel is detected by the device, the Controller will display a solar panel icon and flashing arrow indicating there is power being provided by the solar panel. Lastly, the load of our system is connected through an SPST Switch to the appropriate LOAD+ and LOAD- terminals of the controller from which the voltage supplied to the load will be equivalent to the current voltage of the battery. To supply a consistent 12V to the Control Module, a voltage regulator is wired at the output of the SPST Switch. The Raspberry Pi Pico is powered separately through a USB connection on the PWM Charge Controller which feeds a 3v3 signal back to the power module for conversion to 5V for use back in the control module.

3.4.2 Solar Panel, Battery, and Charge Controller

Solar Panel

For our project, we required a solar panel to keep the battery charged while the system runs. To that end, the Alrska 12V 100W solar panel was chosen due to its more than adequate power output for our project as well as protective characteristics from weather conditions and possible debris.

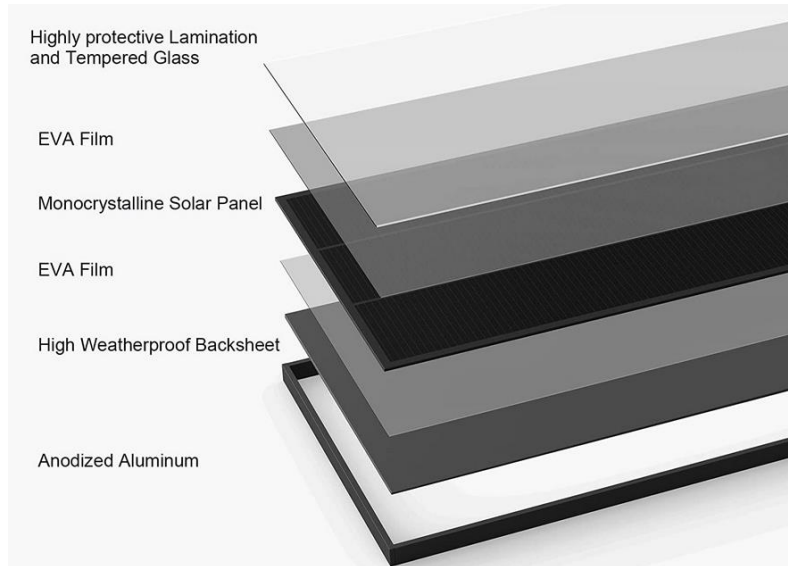


Figure 3.42 – 100W Solar Panel Design Layers [30]

Given the losses through the PWM charge controller, it was also necessary to have a solar panel that would provide more than 16.6W that our load draws as shown in the power budget. Moreover, the need for this solar panel to provide that power until the end of the day at approximately 5:00 pm. Based on data from the US Energy Information Administration, the amount of power provided by a solar panel that is laid flat with no apparent tracking, like the one used in our project, produces only 25% of the power that it would at its peak at or after 5:00 pm [43].

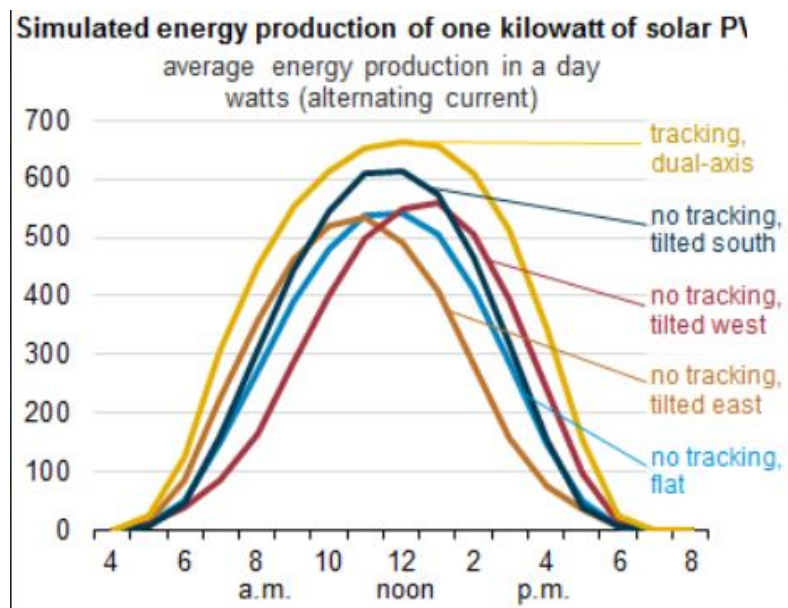


Figure 3.42 – Approximate Solar Panel Power Generation Graph [43]

Accounting for the losses through the PWM charge controller, we note that our approximate power output at 5:00 pm would be roughly 17.44W. After this point in the day, the angle of the sun is reduced to the point where power drawn from the battery is not kept up with by the solar panel, but the device will continue to run. Given, however, that the sun has begun to set by this point, the device does not need to run for much longer as it is assumed to not just be the end of the workday, thus the driver will use the car shortly, but the heating effects from the sun are also reduced because of the reduced exposure to sunlight. During our proposal stage, it was also assumed that the number of fans active at once would be higher than what was capable of being attached to our vent visors and that the current draw would be higher thus, at the time, necessitating the 100W panel.

PWM Charge Controller and Battery

The PWM Charge Controller was chosen due to the necessity of charging the car battery, which provides power to the load and voltage regulators through the PWM, and due to the need to have a method of powering our system which did not rely entirely on solar power which is, as stated in the previous paragraph, subject to have changes in power production throughout the day. It is therefore better to use solar power to charge the battery and draw power from there as it provides a more consistent source of power. Connection to the Solar Panel is made via a set of 10AWG wires with appropriately waterproofed male and female ends compatible with those provided on the Alriska 100W Solar Panel and routed through the rear passenger door to the PWM Charge Controller. Waterproof cabling and connectors were necessary due to the potential for rain during testing and subsequently the risk of creating a short circuit in the body of the vehicle.

Similarly, connections to the battery are made using a set of 10AWG cables with appropriate terminal ends for connection to the battery of the car. The Terminals are secured via hex nuts on the terminal posts of the battery and torqued to prevent slipping loose. Wiring is routed beneath the hood of the vehicle, then into the driver-side door of the vehicle, and finally to the PWM Charge Controller.



Figure 3.43 – Battery and Solar Panel Connection Points

3.4.3 Load Connections and Voltage Regulators

To allow the user to control the output of the system, a simple SPST Switch rated for 125V at 6A is connected in series with the lead of the PWM LOAD+ terminal to the voltage regulator. As the System does not draw more than 2.4A, this was deemed acceptable. This switch cuts primary power from reaching the transistor array of the control module, ensuring that no fans can run while the switch is off. The MCU, however, will continue to run and monitor the status of the battery, windows, and temperatures as it is powered through a different connection. The switch in question and PWM Charge Controller are placed within the lid of the weather-proof box described in Chapter 2 with wiring being soldered to the pins of the switch.



Figure 3.44 – SPDT Switch and PWM Charge Controller Installed to Lid

Buck Converter

At the output of the switch is a Buck Converter capable of regulating the output voltage of our PWM Charge Controller down to the 12V necessary for the DC Blowers of the Cooling Module to function properly without sacrificing too much more in the way of efficiency as the PWM Charge controller already has an approximately 25% loss associated with its method of charging the battery. To this end, a high-efficiency buck converter rated for up to 150W at 5A was acquired and tested to verify its purported efficiency of 93%. A mock-up test running 5 fans off the Converter with an input maximum of 14.6V showed an efficiency of 92.85%.

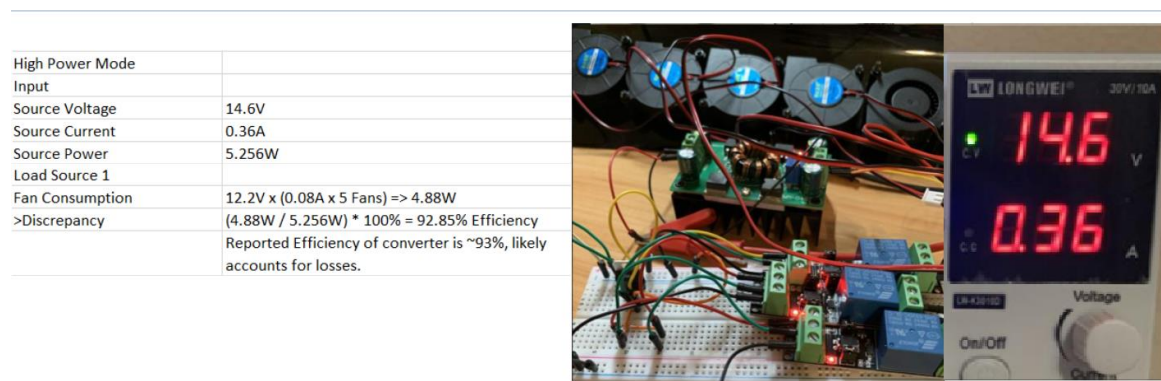


Figure 3.45 – Buck Converter Functionality Test

Notably, the converter came pre-installed with an aluminum heat sink which was removed before integration into the weatherproof box which housed our electronics. While there were initial concerns of overheating during operation, it was decided that the low current draw of 1.28A from the fans relative to its purported maximum of 5A would not cause any issues.

Boost Converter

Due to the presence of the HC-SR04 Ultrasonic Sensors used in the control module, a method of generating a 5V signal at up to 60mA was required. To that end, a Boost Converter was acquired which makes use of the 2108A IC, 1uH inductor, and SS14 surface mount diode to generate a stable 5V output at up to 300mA.

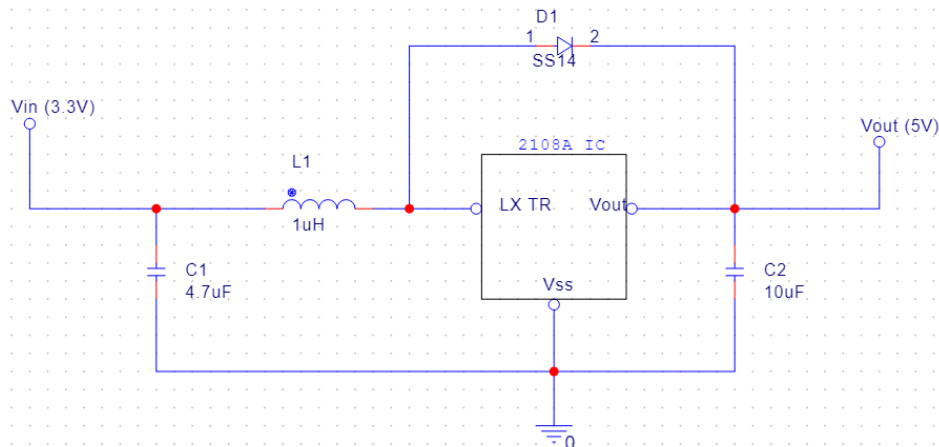


Figure 3.46 – Boost Converter Schematic

According to the Datasheet for the 2108A, the efficiency of the converter is approximately 85%. Given the somewhat low power consumption of the sound sensors at 300mW, the efficiency of this converter was not scrutinized as heavily as that of the Buck converter used to control the voltage supplied to the fans. The trigger input for the sound sensors did not need to be boosted to 5V as the 3.3V output of the GPIO pins used in the control module was sufficient to trigger the sound sensors.

Raspberry Pi Pico Connection

The Raspberry Pi Pico did not require any voltage regulation due to its direct connection to the PWM Charge Controller via USB. The USB connection on the PWM Charge Controller was capable of providing a consistent 5V at up to 2A which far exceeded the requirements for our MCU.



Figure 3.47 - USB Connection to Raspberry Pi Pico

3.5 Troubleshooting

In this section, all troubleshooting for the project will be discussed. This will show how the group worked through issues with various parts of the design.

3.5.1 Raspberry Pi Pico

The group saw the Raspberry Pi Pico fail in two distinct ways: corrupt filesystem and total failure.

Corrupt Filesystem

Using CircuitPython, the Raspberry Pi Pico shows up in Windows File Explorer as CIRCUITPY. It acts like a USB drive and files can be copied to and from it. Occasionally, however, the board's filesystem became corrupt. When this happened, the board showed up as a standard USB Drive with none of the project files intact. To fix this, the code below is entered into the Read Eval Print Loop (REPL).

```
import storage
storage.erase_filesystem()
```

This erases the filesystem and restarts the board. The board boots as an empty CIRCUITPY drive. All the project files are erased, but they can easily be loaded onto the board from a backup drive. Google Drive made this exceptionally easy, as the group had it set up to back up the CIRCUITPY drive any time it was plugged into the computer.

Total Failure

While troubleshooting the battery voltage divider, the 12 V positive lead of the test power supply briefly came in contact with one of the GPIO pins on the Raspberry Pi Pico. The board immediately shut down. It gave no sign of life when unplugged and plugged back in. The filesystem was inaccessible. The group attempted to start the board in bootloader mode to try and reinstall CircuitPython, but this did not work. The board was deemed broken, and a replacement board was ordered.

3.5.2 Battery Level Monitoring Software and MCU ADC Issues

ADC Calibration Using Polynomial Correction Curve

As discovered in the testing, the voltage read by the ADC is not accurate to the actual input voltage. The group saw an approximately 200mV drop in the ADC reading

compared to the DMM-measured voltage. Furthermore, the drop increased as the input voltage increased.

To gather data on this phenomenon, Ian first made an edit to the `get_battery` function so that it would take an average of 1000 readings. In the code below, the number of reps was set to 1000. The program was made to print out `Vadc`.

```
V = []
for _ in range(reps):
    V.append(raw*3.3/65535)
    time.sleep(0.001)
Vadc = sum(V)/reps
```

Next, a power supply was set up to be read by the control module.



Figure 3.48 – Battery Voltage Divider ADC Test

Starting with the PSU set to 12.00 V, DMM measurements of the PSU voltage and the ADC voltage were taken. The program was run to obtain the average ADC voltage reading. The PSU voltage was then increased by 0.30 V increments and the measurements were repeated through 15.00 V. The photos below show the actual supply voltage, the measured supply voltage, and the measured ADC voltage from left to right. Below these photos is the `Vadc` obtained in the program's printout.



Figure 3.49 – ADC Voltage Test using DMM

The measurements were recorded in a [google sheet](#) for analysis. They are also shown in the table below.

Table 3.4 - Vin and Vadc Measurements and Percentage Error

Supply Voltage (V)	Vin (V)	Vadc (V)	Vin - Vadc (V)	Error %
12.00	2.641	2.585	0.056	2.120%
12.30	2.707	2.653	0.054	1.995%
12.60	2.773	2.715	0.058	2.092%
12.90	2.839	2.774	0.065	2.290%
13.20	2.905	2.836	0.069	2.375%
13.50	2.971	2.896	0.075	2.524%
13.80	3.037	2.956	0.081	2.667%
14.10	3.103	3.025	0.078	2.514%
14.40	3.169	3.084	0.085	2.682%
14.70	3.235	3.149	0.086	2.658%
15.00	3.301	3.217	0.084	2.545%

The chart below compares the measured Vin to the ADC voltage. It shows how the difference between Vin and Vadc increases slightly as the supply voltage increases.

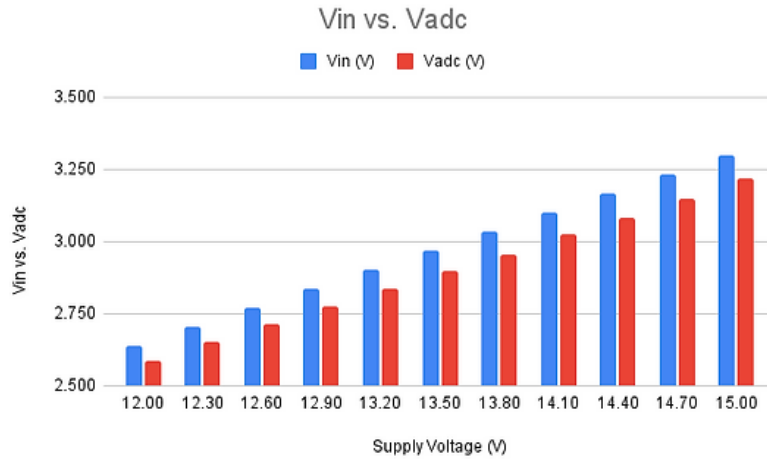


Figure 3.50 – Vin vs Vadc Error Chart

The % Error was calculated using the following equation and then plotted against the ADC voltage. This is shown in blue. In red is a polynomial trendline generated by google sheets. The equation for this trendline is given in the legend along with its R². The R² value describes how well the trendline fits the data, with a perfect fit having a value of 1.00.

Percentage Error

$$\%E = \frac{V_{in} - V_{adc}}{V_{in}} * 100\% \quad (3.12)$$

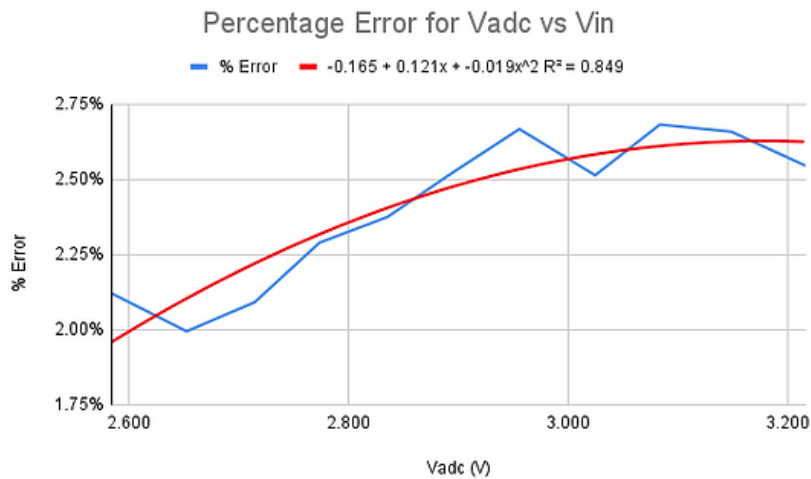


Figure 3.51 – Percentage Error Chart

The equation for the trendline above was applied to a new column to calculate the Trendline Error % for each supply voltage level. Finally, the Vadc Adjusted column is the Vadc added to the product of Vadc and the trendline error.

Table 3.5 – Trendline Error and Adjusted Vadc Voltages

Trendline Error %	Vadc Adjusted (V)
2.082%	2.639
2.228%	2.712
2.346%	2.779
2.445%	2.842
2.534%	2.908
2.607%	2.971
2.666%	3.035
2.716%	3.107
2.745%	3.169
2.762%	3.236
2.762%	3.306

The chart below compares the measured Vin to the Adjusted ADC voltage. The adjusted ADC voltage fits the input voltage much better than the raw ADC voltage.

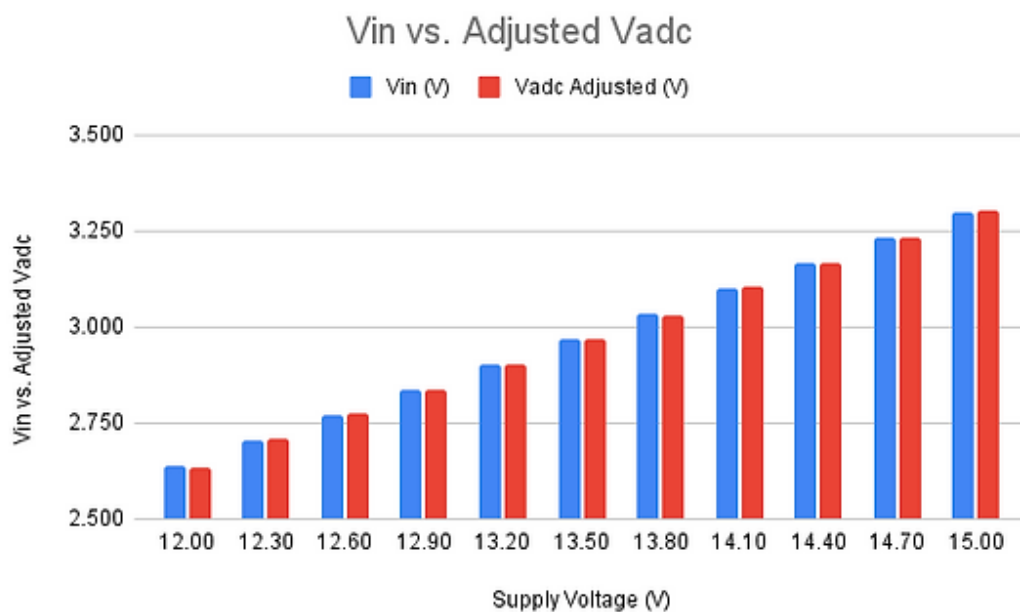


Figure 3.52 – Vin vs 9 Adjusted Vadc Graph

The same adjustment was applied to the `get_voltage` function in the code below. The accuracy of the ADC voltage should be drastically improved by this adjustment.

```
def get_voltage(raw, reps): # Converts digital ADC reading to voltage and takes average over
    specified number of repetitions
    V = []
    for _ in range(reps):
        V.append(raw*3.3/65535)
        time.sleep(0.001)
    Vadc = sum(V)/reps
    Vin = Vadc + Vadc*(-0.1662 + 0.121*Vadc + -0.019*Vadc**2) # Sum of Vadc and the product of ADC
    voltage and trendline error %
    return Vin
```

The thresholds in the `battery_mode` function were also edited to better reflect the `Vin` measurements from the DMM.

```
def battery_mode(Vin): # Determines which battery mode 0-3
    # if difference between Vin and previous voltage is greater than 20 mV:
    if abs(Vin - Vprev) >= 0.050: # Switch battery mode
        if Vin < 2.641:
            bat_new = 0 # 0 active fans
        elif 2.641 <= Vin < 2.707:
            bat_new = 1 # 6 active fans
        elif 2.707 <= Vin < 2.773:
            bat_new = 2 # 12 active fans
        elif 2.773 <= Vin:
            bat_new = 3 # 16 active fans
        return bat_new
    else: # return the previous battery mode (default 0)
        return bat_prev
```

3.5.3 Thermistor Circuit

With only three ADC ports on the Raspberry Pi Pico, the project requires that all four thermistor circuits can be read by the same ADC. To do this, the outputs of each thermistor circuit must be isolated from each other. Starting with just two thermistor divider circuits, the first attempt was a summing amplifier circuit, as shown in *figure 5* below. To test this design, potentiometers were used in place of the thermistor divider circuit.

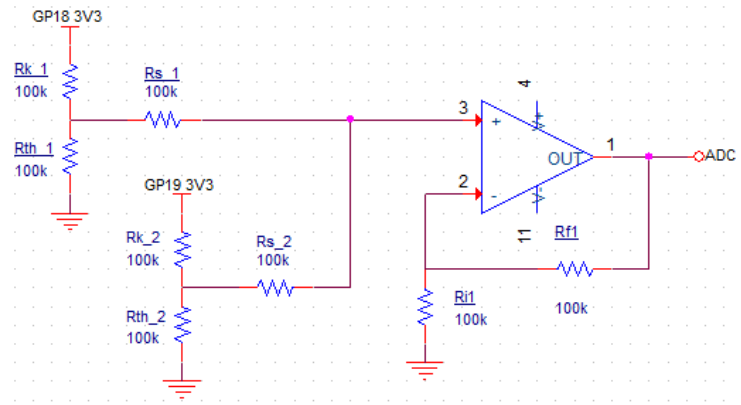


Figure 3.53 - Thermistor output isolation using an op-amp.

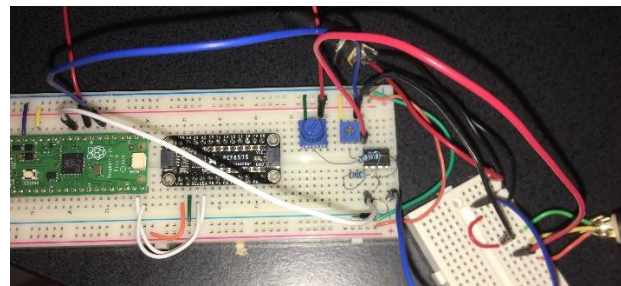
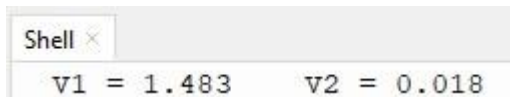
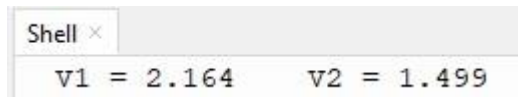


Figure 3.54 - Testing the op-amp isolated thermistor circuit.

The above design failed in isolating the outputs of each thermistor circuit. The voltage output of each thermistor divider was dependent on the other. The screenshots below showcase this behavior. In (a), the first potentiometer was set to its midpoint so that the voltage was divided in half. The second potentiometer was set to its lower extreme. In (b), the first potentiometer was left in the same position, and the second potentiometer was increased to match $V1$ from (a).



(a)



(b)

Figure 3.55 - (a) Pot1 set to midpoint and Pot2 to lower extreme. (b) Pot1 is in the same position, and Pot2 is set to match $V1$ from (a).

Daniel had the idea to use photocouplers to isolate the outputs of each temperature sensor going into the ADC. The following circuit was designed and built on a breadboard using potentiometers.

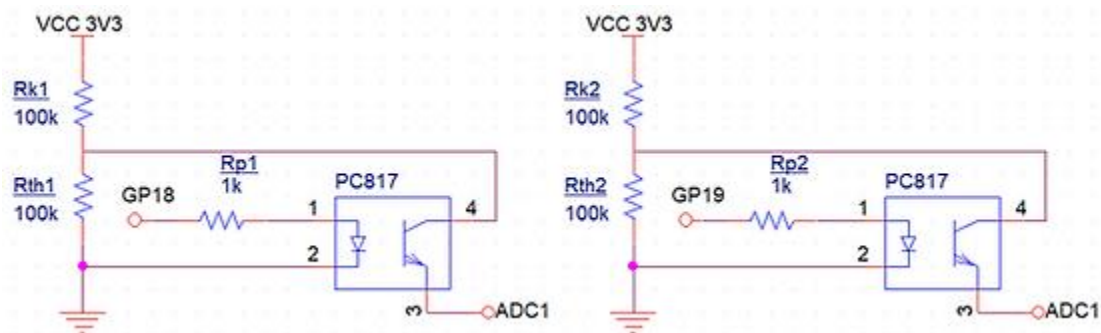


Figure 3.56 – Photocoupled temperature sensor schematic

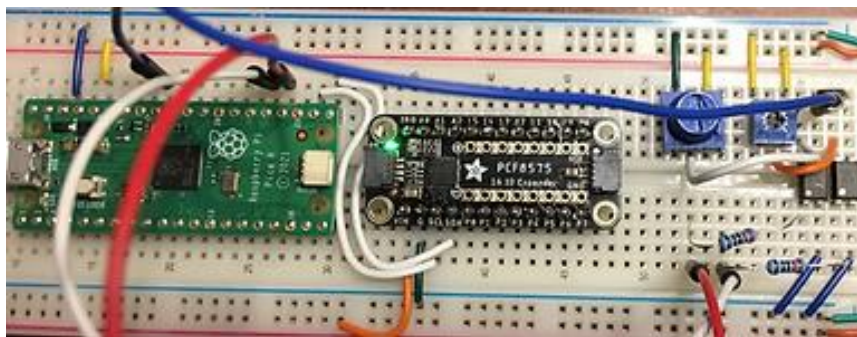


Figure 3.57 - Photocoupled temperature sensor on breadboard

To test whether the outputs were still dependent on each other, a measurement was taken with one potentiometer set midway while the other was set to its lower extremity. Next, a second measurement after changing only the second potentiometer to match the first.

Table 3.6 - New thermistor circuit test

Left Potentiometer: Mid-way Right Potentiometer: Lower extremity	Temperature 1: 75.147 Temperature 2: 416.536
Left Potentiometer: Mid-way Right Potentiometer: Mid-way	Temperature 1: 74.872 Temperature 2: 75.698

The circuit succeeded in isolating the outputs of the two thermistor circuits. One concern with this circuit is that the voltage read by the ADC may be slightly inaccurate due to any voltage drop across the PC817's phototransistor. When measured, however, only 0.0024 mV appeared across the pins. Regardless, the circuit can be calibrated to a known good temperature sensor by offsetting the input voltage in the code.

3.5.4 Proximity Sensor

IR Proximity Sensors were initially selected for the detection of the windows. The design process for these sensors is detailed below along with the reasons for failure and swap to ultrasonic proximity sensors. The troubleshooting of the ultrasonic proximity sensors is also discussed.

IR Proximity Sensor

This section details the design and troubleshooting process of the IR Proximity Sensor. A simple IR proximity sensor can be made with only IR LEDs, IR photodiodes, and a couple of resistors. It was an apt solution in principle, but the operating environment rendered the IR Proximity Sensor useless.

IR LEDs and IR Photodiodes

IR LEDs emit infrared light when a forward voltage is provided across them. Contrarily, an IR photodiode generates a current when infrared light shines upon them. They are designed to be reverse biased, as the resistance of the diodes increases proportionally to infrared light, thus increasing the reverse current [44].

IR Proximity Sensor Operating Principles

A basic IR proximity sensor schematic is given in *figure 5*. When GP22 is high, no voltage is seen across the IR LED. When GP22 goes low, a forward voltage is applied to the IR LED, which produces infrared light. When an object approaches the proximity sensor, the infrared light bounces off it and hits the IR photodiode, which produces a current and voltage. The current and voltage are therefore proportional to the proximity of the object. When the ADC reads the voltage, it reads the proximity of the object.

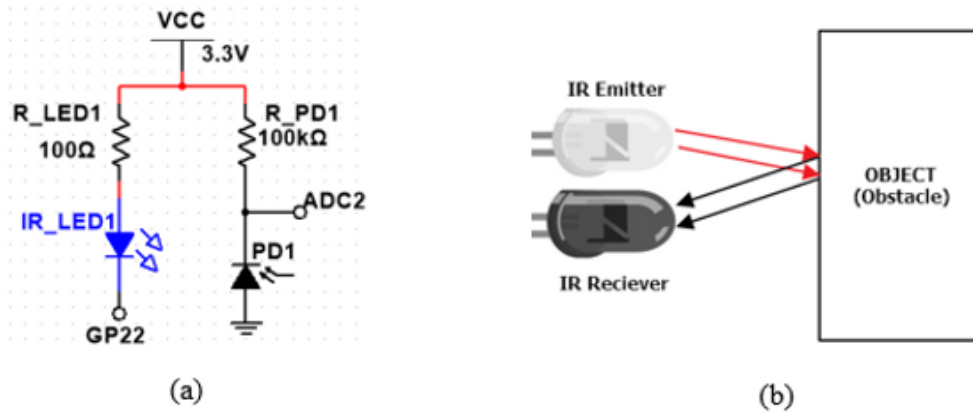


Figure 3.58 - (a) IR Proximity Sensor Circuit. (b) Detection of an object [45]

IR Proximity Sensor Design

Since four proximity sensors are needed and only one ADC is available to read them, the following circuit was designed to turn on all IR LEDs at once and read all IR photodiodes at once. Since the IR photodiodes are in parallel, their voltages sum at the ADC. Once the ADC voltage falls below a specified threshold, all windows can be said to be down.

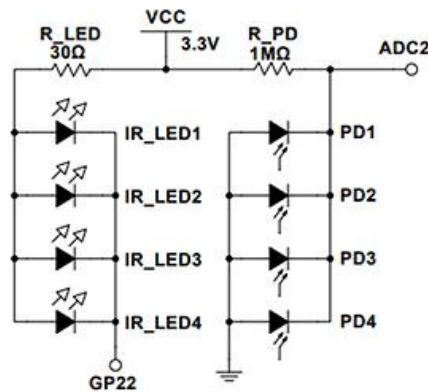


Figure 3.59 – IR Proximity Sensor Circuit Schematic

A program was written to the MCU that reads the distance of an object from the sensor. It then takes the measured distance and determines whether the window is up or down based on a specified threshold. If the windows are down, then a green indicator LED will be turned on.

Testing the Proximity Sensor

The IR Proximity sensors were tested after soldering them to the prototype board. The results were promising, as they adequately detected nearby objects. The green LED switched ON and OFF properly with the window status.

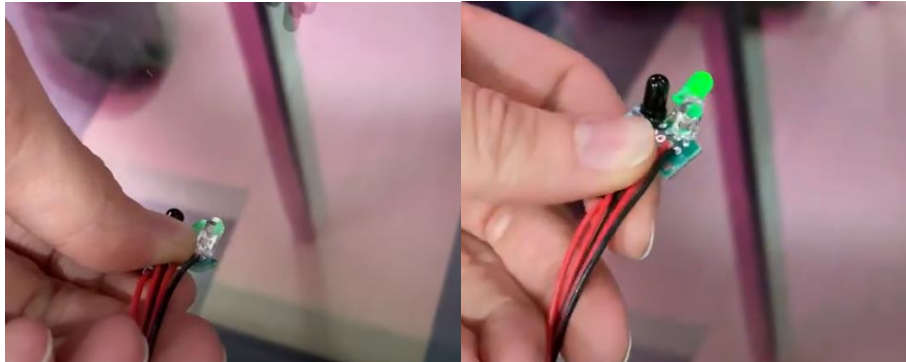


Figure 3.60 – IR Board LED Test

Once attached to the fan vent-visor assemblies and mounted to the car, the IR sensors did not work as they did in the lab. There was hardly any difference seen by the ADC when an object approached the sensor. This is because the IR Photodiodes were being saturated by the sun. The group confirmed this by performing a test that compared the IR sensor output when exposed to sunlight and not exposed to sunlight. This renders the IR sensors useless, as they will always be used outside for this project.



Figure 3.61 - (a) IR Sensor output with window blinds closed. (b) With blinds open.

Dr. Radu Bunea, a professor at Valencia College, suggested that ultra-sonic sensors be used in place of IR sensors. They effectively perform the same task but use sound instead of light. They are also far cheaper than quality optical filters.

Swap to Ultrasonic Proximity Sensor

To replace the IR Proximity sensor, the HC-SR04 Ultrasonic Proximity Sensors were acquired. A test was performed which confirmed their functionality as proximity sensors with a function generator and an oscilloscope.

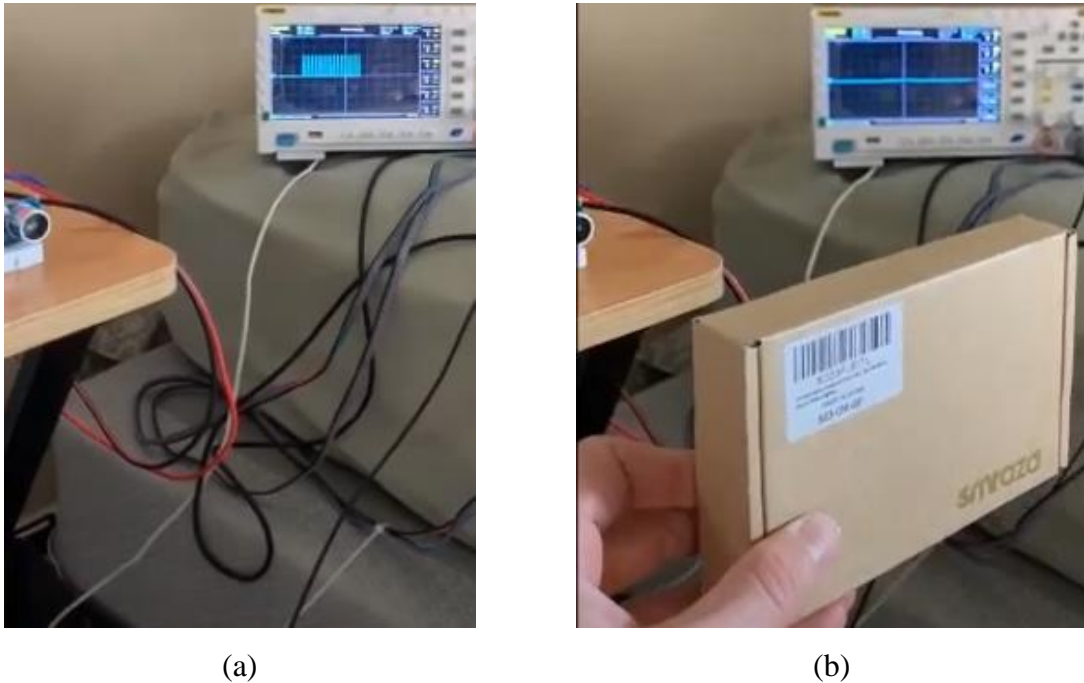


Figure 3.62 - (a) Ultrasonic Echo signal with no object present and (b) with an object present.

System Integration of Ultrasonic Sensors

Luckily, this sensor has the same number of pins that the IR sensor had, so implementing it into the control module was relatively painless. To start, the schematic in *figure #* was drawn up to use as a reference while soldering. Next, the IR circuit components were desoldered from the MCU board. The completed circuit is shown below.



Figure 3.63 - Ultrasonic sensor components soldered to the MCU prototyping board

2-pin connectors were used to connect the ultrasonic sensors to the control module. These were soldered and wrapped directly to the pins. The green indicator LED's cathode was also soldered directly to the ground pin of the window 1 ultrasonic sensor. A single-pin connector was used to connect the LED's anode.



Figure 3.64 – Ultrasonic Sensors with Connections and LED Indicator

Next, the proximity sensor code had to be modified. Ian kept the code very similar to the original proximity code. It essentially pulses the Trig pin at 50 Hz and reads the Echo pin at the same rate. When an object approaches the sensor, the average voltage at the Echo pin should increase.

```
# Initializing ADC2 and GPIO pins
Echo = analogio.AnalogIn(board.GP28) # Ultrasonic Echo pin
Trig = DigitalInOut(board.GP22) # Ultrasonic Trigger Pin
Trig.direction = Direction.OUTPUT

def readEcho(reps):
    distance = 0

    for _ in range(reps): # Measure distance from sensor for specified num of repetitions
        difference = 0
        # Get ambient Echo
        Trig.value = True # Active high
        time.sleep(0.01) # 500 Hz
        reflectEcho = Echo.value # Ambient Echo measurement

        # Get window Echo
        Trig.value = False
        time.sleep(0.01) # 500 Hz
        ambientEcho = Echo.value # Measurement of reflected Echo

        # Storing the difference and adding to distance
        difference = ambientEcho - reflectEcho
        distance += difference

    return distance/reps
```

To determine whether the window is up or down, the threshold is set in the code below.

```
def up_down(distance): # Determines window position from distance
    if distance > 48000:
        window = "Up"
        led.value = False
    else:
        window = "Down"
        led.value = True
    return window
```

To test the sensors, all four of them were connected and set up on the bench such that they were aimed out into the room. A hard reflective object was placed in front of each ultrasonic sensor and then removed one at a time. The distance value was observed. With no object present, the distance value came out to about 50000. When an object was placed right in front of any of the sensors, a drop of about 5000 occurred, putting the distance around 45000. The threshold was set so that if the distance exceeded 48000, the window would be registered as "Down". Else, the window registers as "Up".

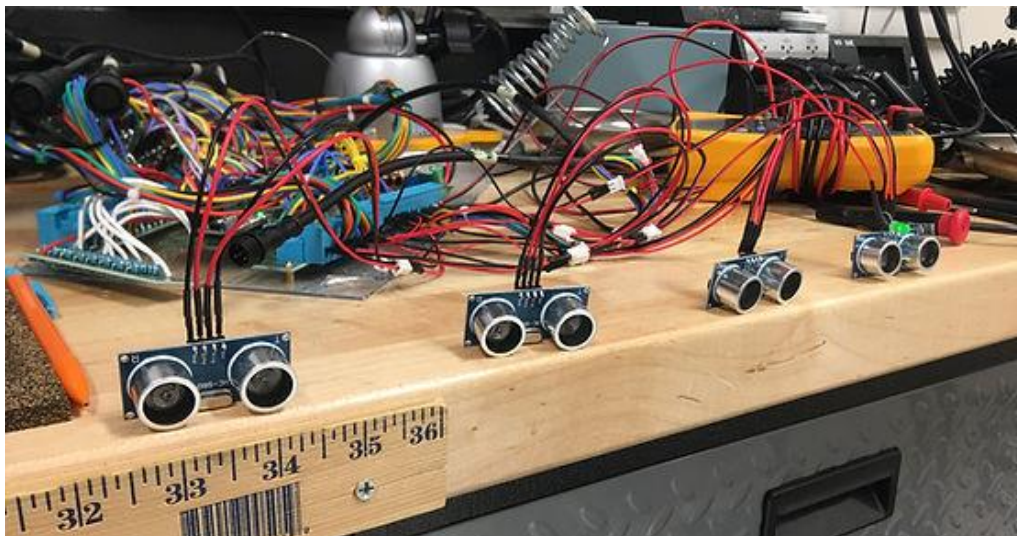


Figure 3.65 – Ultrasonic Sensors All Active off Control Module

With the threshold set properly, Ian ran the code and moved the object in front of each of the sensors in three-cycle intervals to produce the printout shown below. The green LED attached to the window 1 sensor lit up when the window was "Down" and turned off when the window was "Up," as expected.



Figure 3.66 - Testing integrated operation of all four ultrasonic proximity sensors.

Ultrasonic Proximity Sensor Testing and Troubleshooting

Once the vent visors were patched up and adhered to the testing vehicle, the group set about testing the ultrasonic proximity sensors. The sensors immediately presented some issues. The distance value came back lower than it did in the lab and the sensors weren't as sensitive to objects in close proximity. Also, one of the sensors seemed to be far less sensitive than the others, making it impossible to properly set the switching threshold.

First, the group used an oscilloscope to verify that the correct signals were being sent and received by each sensor. Once this was verified, the group began modifying the program to adjust the sensitivity of the sensors. A variety of pulse frequencies and repetitions were tested to no avail. The group decided that a rewriting of the program could prove beneficial at this stage.

Going by the HC-SR04 datasheet [46], the sensor can be triggered by a 10us or greater pulse. At the falling edge of this pulse, the transmitter will emit an 8-cycle sonic burst at 40 kHz. The Echo Pulse lever signal output which follows is proportional to the proximity of the sensor to the object it is directed towards.

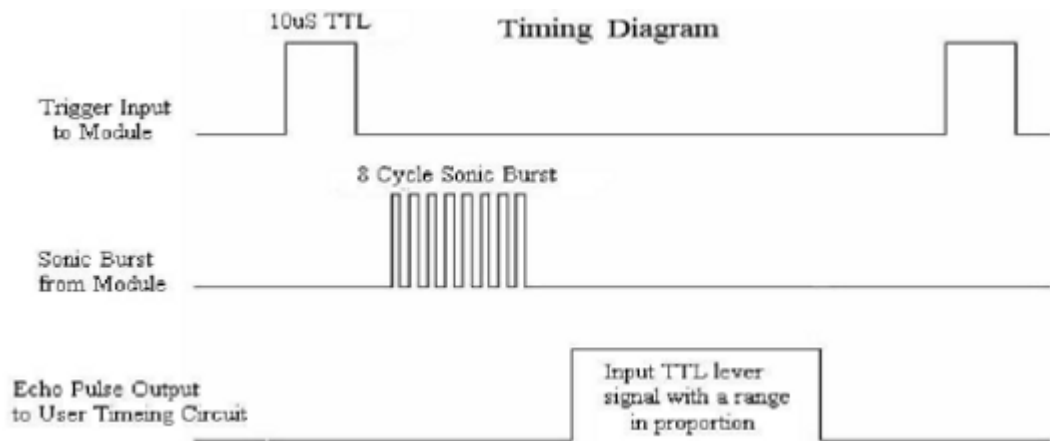


Figure 3.67 – Timing Diagram for Ultrasonic Sensors

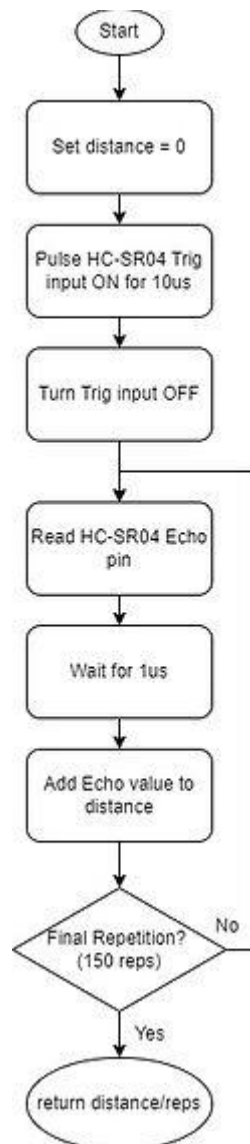


Figure 3.68 – Ultrasonic Sensor Code Main Loop

The flowchart above shows how the new program works. After setting the distance variable to 0, the Pico sends a 10 μ s pulse to the HC-SR04. Immediately after, the Pico reads the ADC connected to the HC-SR04 Echo pin every microsecond for a specified number of repetitions. Once the specified number of reps has occurred, the program returns the average distance value.

Testing the program in the vehicle, the sensors were far more sensitive and consistent, but one sensor had reduced sensitivity. The group adjusted the number of repetitions until the sensitivity of all the sensors was great enough. 150 repetitions worked very well, and a new switching threshold was set at 25000.

The group tested the functionality of the proximity sensors according to the test procedures laid out by Daniel: by rolling down all the windows and then rolling them up one at a time. Per the engineering requirements, power should be disconnected from the fans if any of the windows are rolled up too high for airflow. The green LED should also be on when the windows are down and off if they are up.

The results of the test indicate that the proximity sensor circuit and control are functioning exactly as intended. When any of the windows are rolled up too high for airflow, the fans and the green LED shut off. When all windows are low enough, the fans and green LED automatically turn back on. This constitutes a pass for the engineering requirement concerning window sensors.

3.5.5 Relays and the Switch to Transistors for Fan Switching

The group initially intended to use optocoupled relays to switch the 12V fans on and off. They proved to be cumbersome due to their size and current draw. Because the project required 24 relays, housing and wiring them all would have been challenging. Furthermore, each relay drew about 100 mA to 150 mA. Since the Raspberry Pi Pico's 3.3V pin should not supply more than 300 mA, an external power supply would need to be used.

The group decided to try using transistors to switch the fans instead of relays. 2N3904s fit the project's requirements. The 2N3904 has slow switching times and can handle up to 200 mA with minimal voltage drop, making them suitable for the project. The simulation below confirmed the group's suspicions.

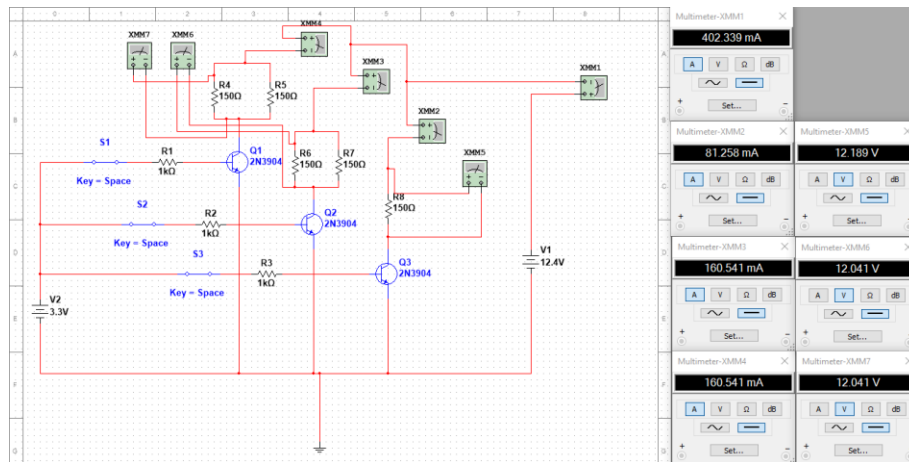


Figure 3.69 – Multi-Sim Test for Compatibility with 2N3904 Transistors

The group ordered and tested the 2N3904s. Daniel tested an input of 14.6V into the regulator with a 12.3V to 12.4V output wired to the collector of each Transistor. 1kOhm resistors were wired into the input of the base with a 3.3V signal (emulating the MCU GPIO signal) to confirm the functionality of the transistor and its capability to activate a fan array. The group ultimately decided to use transistors instead of relays.

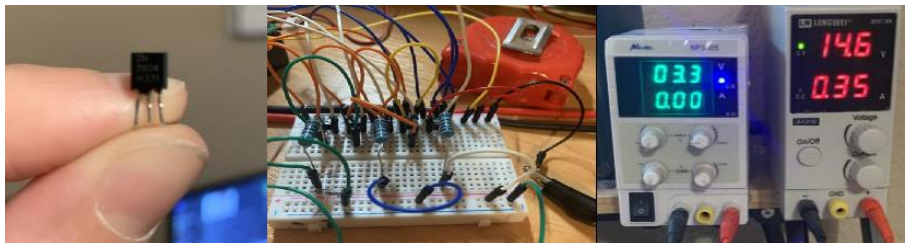


Figure 3.70 – Breadboard Testing the 2N3904 Transistors

3.5.6 Transistor Array Fan Activation Issues

The fan control system involves the MCU, the GPIO Expander, the transistor switching array, the fans themselves, and a heap of connections. Thorough testing and troubleshooting were required to get this system to function correctly.

Fan Control Initial Test

After connecting all fans to the control module, the group first activated the entire transistor array to verify that power was being provided to the fans through the array. An entire fan array was non-functioning, along with a few random fans. Next, the group verified that each fan array was being triggered by its corresponding section on the

transistor array. To do this, each transistor's base was activated one at a time from the fan control program. This made it immediately obvious that two of the arrays had been swapped while plugging them in.

Once the fan array cables were swapped to the correct positions, the group went through a checklist of all the transistors and made notes on each of the non-functioning ones. In doing this, the group discovered that none of the transistors connected to the PCF8575 were being triggered. Using an oscilloscope, the group discovered that the voltage at PCF8575 GPIO pins was very low.

Troubleshooting the PCF8575 GPIO Expander

Doing a little bit of forum research, the consensus seemed to be that pull-up resistors are required to drive transistors from the GPIO on the PCF8575. The group tested this on one of the pins by connecting a 1 k Ω pull-up resistor in the following configuration:

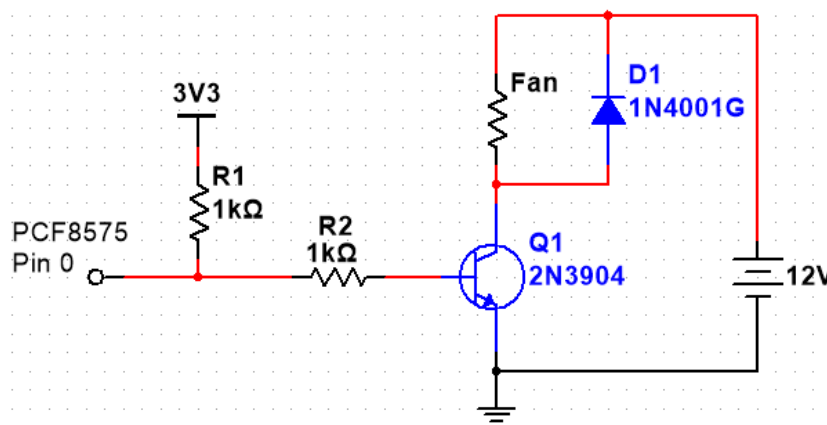


Figure 3.71 - Transistor Circuit with PCF8575.

The above circuit worked, allowing the PCF8575 to trigger the base of the transistor, allowing current to flow through the fan. Since eight of the transistors receive their base from the PCF8575, the group had to add a pull-up resistor for each of the eight pins. The group calculated that the current limiting resistors at the bases of these transistors would need to be swapped to 10 k Ω . This gives a parallel resistance of 1.375 k Ω to the 3.3 V supply.

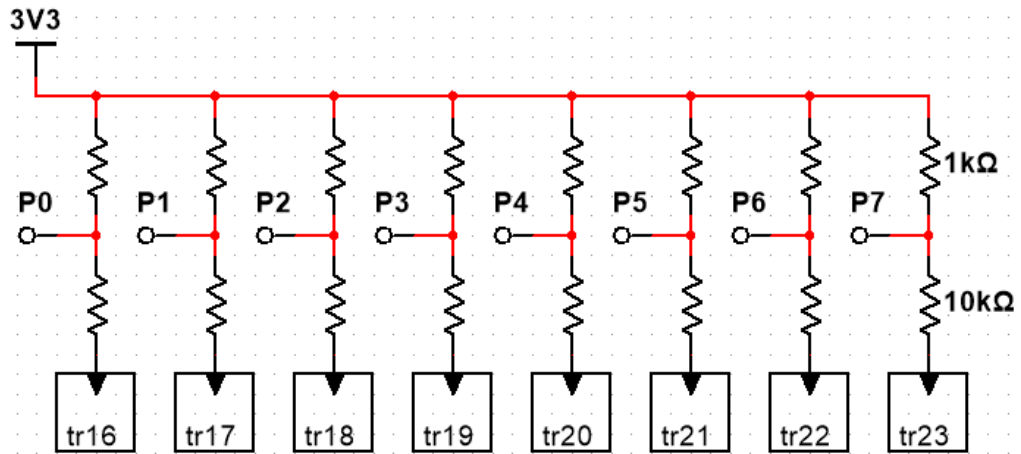


Figure 3.72 - Schematic for PCF8575 Connections to Transistor Array

Since there was not enough space on the PCF8575 prototyping board, a new board had to be added to the control box to hold and connect all the resistors.

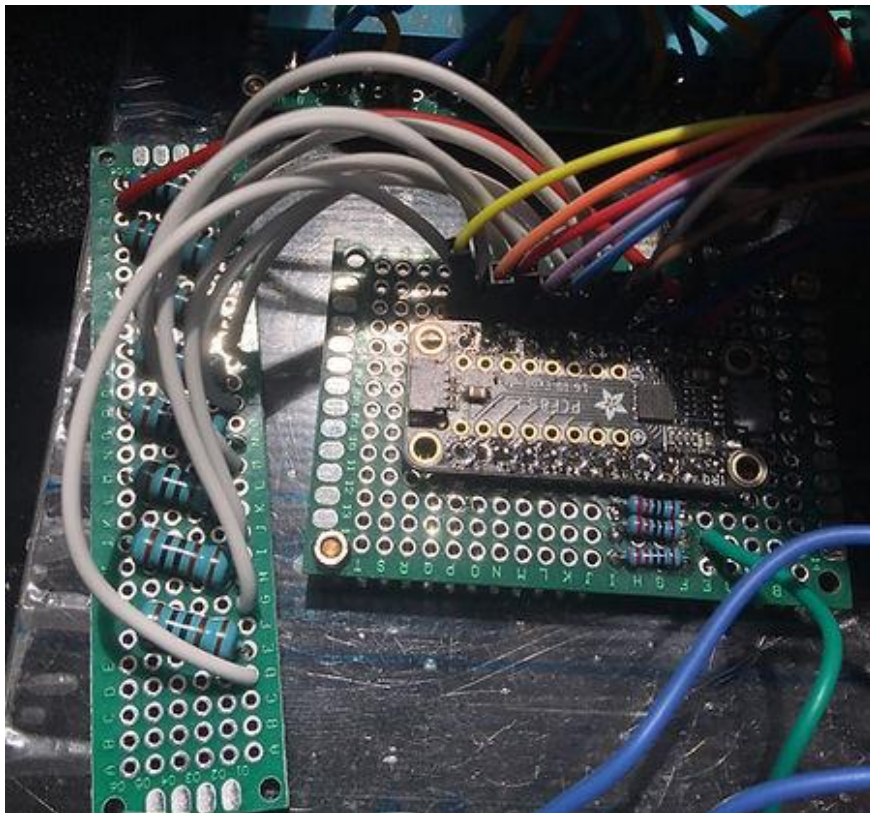


Figure 3.73 - PCF8575 Pullup Resistor Wiring

Once all changes were made, the group tested fan control from the PCF8575 and verified that the transistors and fans were being triggered and getting power appropriately. This brought the non-functioning array and a few other fans back to life

Troubleshooting Transistor Issues

Triggering the entire transistor array once more, only a few fans remained non-functioning. The group took an oscilloscope to the corresponding transistors to troubleshoot. See *Figure 3.9* for Transistor locations.

- Transistor tr1 saw no voltage at the collector. Visually inspecting all connections from the transistor to the terminal block, the group discovered that the jumper from the terminal block to its prototyping board was disconnected. Soldering a new connection solved the issue.
- Transistor tr5 saw no voltage at the base. The group used the oscilloscope to determine that the GPIO signal from the Pico was not reaching the board. A visual inspection revealed that the pin was not soldered properly. The connection was soldered, and the issue was resolved.

Once these issues were addressed, all fans ran when triggered and shut off completely when not. The group moved on to system control tests.

3.5.7 Miscellaneous Issues and Fixes

During group testing, the wiring for one of the fans broke off. The wires were soldered back onto the fan PCB as shown below.



Figure 3.74 - Resoldered Fan Wiring

3.6 Testing and Results

In this section, we will detail the testing performed and the methodology to meet our engineering requirements as detailed in Chapter 2. As will be detailed further in this section, given the time constraints for this project and the time required per most high-level requirement testing, most tests were performed once.

3.6.1 Acceptance Test Procedure and Test Setup

Before the start of testing, an Acceptance Test Procedure, or ATP, which details the testing methods to be used and the pass/fail criteria for each requirement was created to verify our engineering requirements detailed in Chapter 2. Along with the test setup requirements, this document lays out the specifics of how each requirement will be met and documents the overall results. This document can be viewed under [Project > Acceptance Test Procedure](#) on our website.

Test Setup

Power shall be supplied by the battery of the vehicle through the 30A PWM Solar Charge Controller. Connections from the battery of the vehicle to the Charge controller must first be made by connecting the positive and negative rails of the car battery to the input positive and negative terminals of the Charge Controller. The battery of the car shall be recharged while there is sufficient sunlight by wiring a 100W solar panel to the Charge Controller after the connections to the battery have been made. Lastly, the UUT will be connected to the output of the Charge Controller with a switch acting as the means of breaking the connection to the load. The Raspberry Pi Pico which controls the system will be supplied power by the USB port of the Charge Controller.

A reference vehicle that fits within the same EPA Size class shall be used to compare internal temperatures throughout the testing period. Two temperature sensors will be placed in the front and back seat, on the floor of the vehicle, and **kept out of direct sunlight** for continuous monitoring of the internal temperature in the Test and Reference vehicle. Four Temperature sensors will be attached to each window location for continuous monitoring of the external temperature of the vehicle at each of the windows. An average between the front seat temperature and back seat temperature will be taken at the end of testing to verify requirements. The temperature at each of the

windows will be monitored by software on the Raspberry Pi Pico and compared against those of the external temperature sensors for validation of functionality. (NOTE: External Sensors are subject to direct sunlight and will have a higher temperature than those within the vent visors, thus the temperature monitoring is to verify the device can chart a difference between each location)

(NOTE: THE TEST VEHICLE MUST BE IN THE SAME DIRECTIONAL ORIENTATION AS THE REFERENCE VEHICLE.)

When testing for rain rejection, visual tests will be performed to determine if water has entered the vehicle at any point where the vent visors are installed.

For Driving portions of the test, the vent visors will be fully attached to vehicle locations using the adhesive strips that come with the vent visors. Before any driving tests, clips or other methods may be used to attach the vent visors to the vehicle so long as the method of attachment does not interfere with the function of the device.

Verification of Reference Vehicle Validity

Both the test vehicle and the reference vehicle fall into the same EPA size class, but many other variables may contribute to temperature differences between vehicles, i.e., vehicle color, interior color, windshield size, etc. A control test was conducted to account for these variables. The test is designed to compare the differences in internal temperature between vehicles with no cooling methods active. In this test, both the test vehicle and the reference vehicle were set up following the Test Setup procedure above. The vehicles were parked next to each other in a sunny location and oriented in the same direction. All windows on both vehicles were closed. The solar-powered ventilation system was turned off, and temperature data were recorded every hour for 7 hours, starting at 10:30 AM.

The data in *Table 3.7* shows that between the test vehicle and the reference vehicle, there is an average internal temperature difference of approximately 0.5 °F, which is a percentage difference of about 0.5%. Furthermore, *figure 3.75* shows that neither of the vehicles remains hotter than the other for the duration of the test. The reference vehicle was hotter than the test vehicle until 2:30 PM when the test vehicle became hotter and stayed hotter for the rest of the test period. A potential cause for this could be that the test vehicle's windshield is more vertical than the reference vehicle, so the interior has more protection when the sun is directly overhead or behind the vehicle.

As the sun begins to drop in the sky in the late afternoon, more sunlight enters through the windshield and strikes the black interior of the test vehicle, causing the interior to heat up rapidly. Meanwhile, the reference vehicle has a beige interior which doesn't absorb as much heat.

Table 3.7 - Baseline Temperature Comparison Test Data from 4/8/2023

TEST VEHICLE				REFERENCE VEHICLE			Outside Ambient Temperature
Time	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	
10:30:00 AM	80.0	81.1	80.6	83.0	84.5	83.8	85.0
11:30:00 AM	86.0	87.1	86.6	89.4	90.3	89.9	87.0
12:30:00 PM	98.0	99.8	98.9	103.0	95.4	99.2	88.0
1:30:00 PM	101.1	100.4	100.8	106.3	98.0	102.2	89.0
2:30:00 PM	102.8	109.2	106.0	110.0	101.9	106.0	89.0
3:30:00 PM	106.8	114.9	110.9	110.7	102.3	106.5	89.0
4:30:00 PM	109.7	114.7	112.2	111.0	103.4	107.2	89.0
5:30:00 PM	107.9	110.1	109.0	109.3	102.9	106.1	88.0
		Average Internal Temp:	100.6		Average Internal Temp:	100.1	

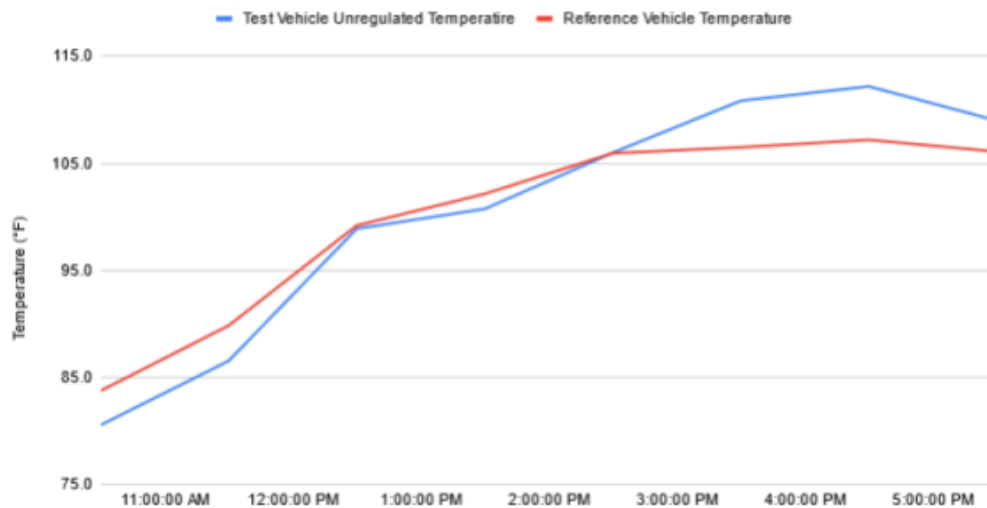


Figure 3.75 - Baseline Temperature Comparison Test 4/8/2023

3.6.2 High-Level Requirement Testing

130% Spec Temperature Test

Per the ATP, our first test was performed to determine the effectiveness of the device such that it will keep the maximum internal temperature of the vehicle to no more than 130% of the maximum ambient temperature. For this, the unit ran for 8 hours from 9:30 am to 5:30 pm with internal temperatures logged from both the UUT and reference

vehicle at the start of every hour. External temperature sensors mounted at each window were also logged at the start of every hour.



Figure 3.76 - Temperature Testing Setup

The outside temperature was logged using the Apple Weather App along with the high for the day which was used as the value to determine if the UUT maximum internal temperature exceeded an internal temperature of 130% of the maximum outside ambient temperature. Lastly, the Voltage as seen on the PWM Charge controller was logged for comparison against the battery level monitoring performed by the Raspberry Pi Pico. For validation of the 130% Spec, a reference vehicle is not necessary.



Figure 3.77 - PWM Charge Controller View and Weather App Screen

At the end of the testing period, the data was placed into an Excel Sheet for analysis to confirm whether the UUT meets our high-level requirement. Data gathered from other comparison tests can also be used to confirm this requirement as the reference vehicle has no bearing on whether the UUT can meet this requirement on different days.

Test 1: 130% Spec Temperature Test 3/14/2023

Table 3.8 - Temperature Test 3/14/2023

TEST VEHICLE				REFERENCE VEHICLE			Outside Ambient Temperature
Time	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	
9:30:00 AM	61.4	61.8	61.6	69.6	70.0	69.8	57.0
10:30:00 AM	70.0	71.9	71.0	84.4	86.6	85.5	60.0
11:30:00 AM	80.3	81.9	81.1	98.4	99.9	99.2	64.0
12:30:00 PM	88.9	89.6	89.3	108.2	108.1	108.2	68.0
1:30:00 PM	92.6	92.6	92.6	116.9	113.4	115.2	71.0
2:30:00 PM	95.5	94.5	95.0	121.9	115.9	118.9	72.0
3:30:00 PM	95.5	95.0	95.3	125.1	116.5	120.8	73.0
4:30:00 PM	91.4	96.7	94.1	121.7	106.4	114.1	74.0
5:30:00 PM	92.3	98.1	95.2	97.0	110.8	103.9	73.0
		Average Internal Temp:	86.1		Average Internal Temp:	103.9	

As seen in *Table 3.8*, the highest average internal temperature reached in the Test Vehicle was 95°F. The high ambient temperature for that day reached 74°F placing our 130% value at 96.2°F. This means our device met the spec at 128.78% of the maximum ambient temperature. Charted in *Figure 3.78* is the Test Vehicle Temperature vs the Reference vehicle over the day. The ambient temperature is plotted for comparison.

TEST 1: PASS [✓] / FAIL [] 3/14/2023 MAX TEMP: 128.78% of Ambient

The temperature data taken from each window by the MCU were plotted over time to see how the device changes airflow direction over time. *Figure 3.79* shows that the system remained in cooling mode 2 for most of the day, taking air in from the passenger side and exhausting it from the driver's side. Towards the end of the day, the temperature difference was diminished, and the system switched to mode 1 and mode 4.

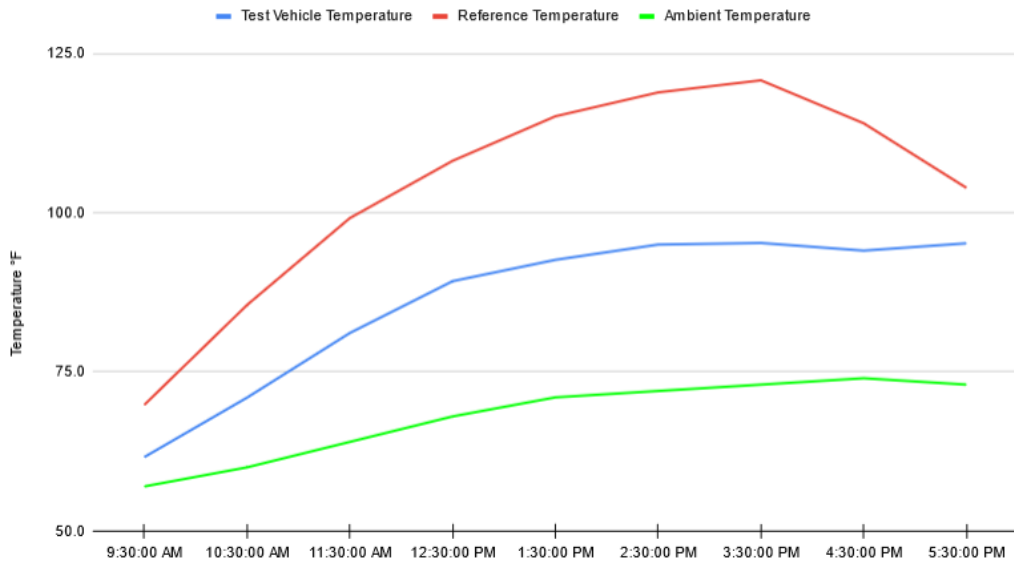


Figure 3.78 - Test Vehicle Temperature vs Reference Vehicle Temperature

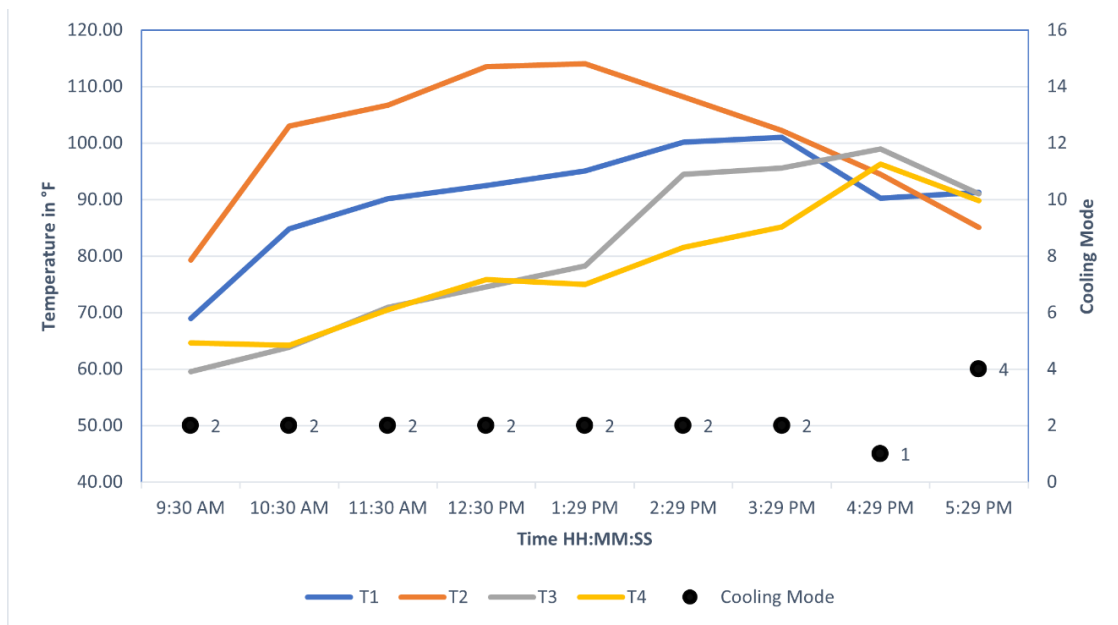


Figure 3.79 - Temperatures and Cooling Modes vs. Time. 3/14/2023

Test 2: 130% Spec Temperature Test on 3/18/2023

This test was started later in the day to view the effects of starting later and if it would have any effects on the outcome. The maximum ambient temperature for the day was 85°F putting our 130% spec at 110.5°F. As seen in *Table 3.9*, the maximum internal temperature reached was 92.9°F putting us at 109.29% of ambient.

Table 3.9 - Temperature Test 3/18/2023

Test Vehicle				
Time	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	Outside Ambient Temperature
11:00:00 AM	76.6	78.5	77.6	76.0
12:00:00 PM	82.7	82.5	82.6	78.0
1:00:00 PM	86.9	86.0	86.5	78.0
2:00:00 PM	90.3	90.0	90.2	84.0
3:00:00 PM	93.6	92.2	92.9	85.0
4:00:00 PM	91.1	92.9	92.0	85.0
5:00:00 PM	88.8	90.5	89.7	83.0
6:00:00 PM	86.7	85.6	86.2	78.0
7:00:00 PM	84.0	83.3	83.7	77.0
		Average Internal Temp:	86.8	

Charted in Figure 3.80 is the temperature data for the test vehicle over the day. The ambient temperature is also plotted for comparison and validation of results.

TEST 2: PASS [✓] / FAIL [] 3/18/2023 MAX TEMP: 109.29% of Ambient

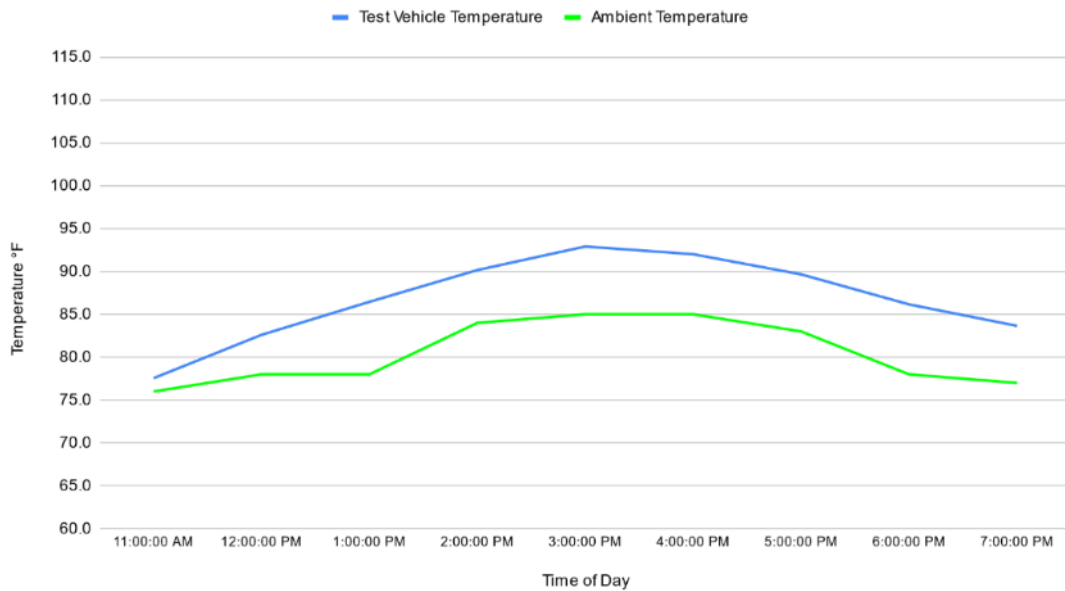


Figure 3.80 - Graph of Test Vehicle Temperature vs Time

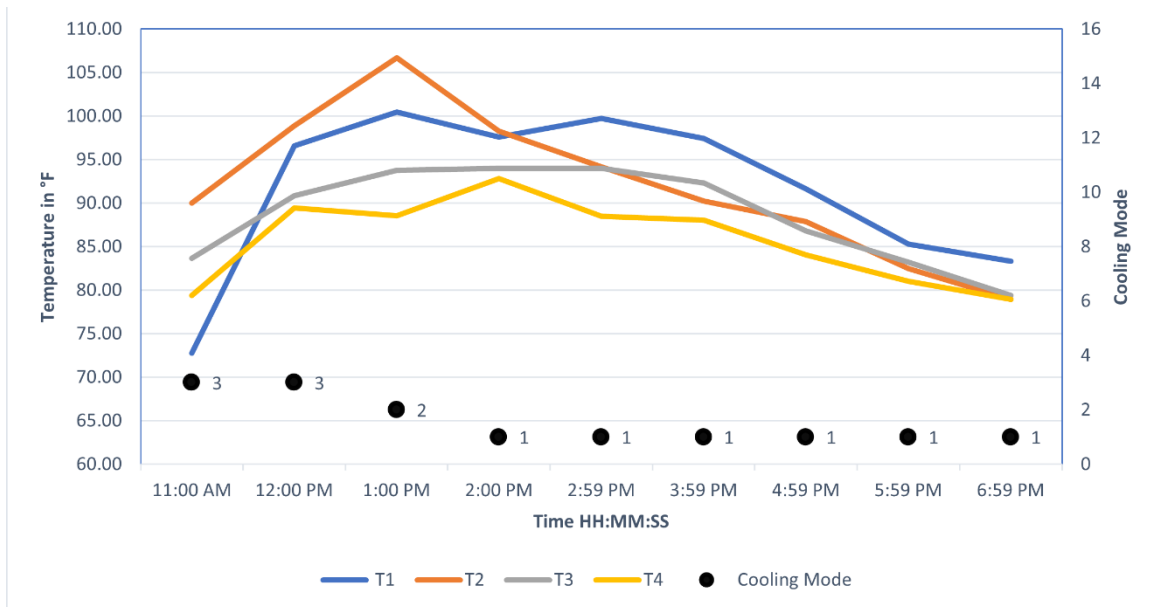


Figure 3.81 - Temperatures and Cooling Modes vs. Time. 3/18/2023

Figure 3.81 shows how the cooling modes change according to temperature differences during the 130% spec test. Oddly, the device switches into and stays in mode 1 throughout the later portion of the testing period even though the driver side was hotter than the passenger side, calling for mode 2. This can be explained by the switching threshold. For the MCU to switch the cooling mode, the difference between the standard deviation of the new temperature data and the previous temperature data must be greater than 3. Referring to figure 3.82, there is a drop of 4.17 in the temperature standard deviation from 1:00 PM to 1:10 PM. Having exceeded the threshold, the MCU looked at the temperature data gathered at 1:10 PM, determined that the driver side was cooler than the passenger side, and switched the cooling mode to mode 1. After 1:10 PM, the standard deviation threshold was not met again, which is why the device stayed in mode 1 even after the driver side became hotter than the passenger side at 1:20 PM. This led the group to believe that the device's cooling capabilities may be improved by reducing the switching threshold for cooling modes.

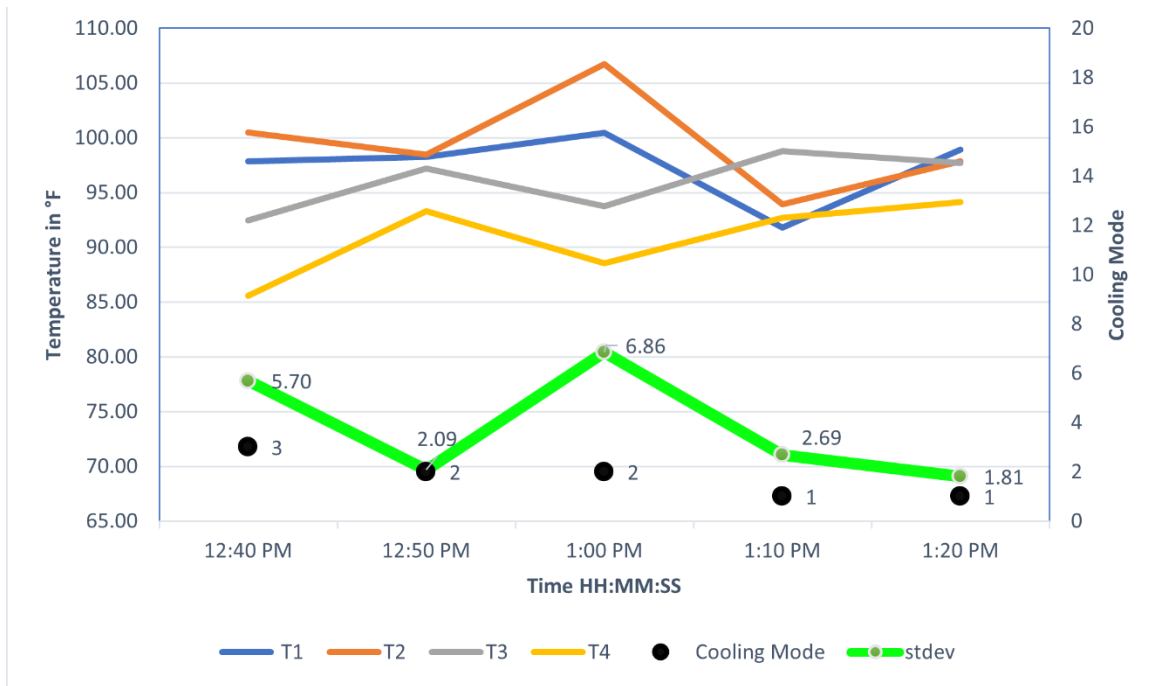


Figure 3.82 – Temperature vs. cooling mode vs. standard deviation from 12:40 PM to 1:20 PM, 3/18/2023

Test 3: Sunshield Comparison Test on 3/20/2023

See the Sunshield Comparison Temperature Test section for detailed Temperature data.

The maximum internal temperature reached in the UUT was 88.3°F. With a maximum external ambient temperature of 69°F, this puts the 130% spec at 89.7°F. Therefore, the device once again met the requirement at 127.97% of the maximum outside ambient temperature. Notably, the total sun exposure through the front windshield plays a significant role in the overall increase in temperature experienced by the vehicles which are discussed in the Sunshield Comparison Test section of this report.

TEST 3: PASS [✓] / FAIL [] 3/20/2023 MAX TEMP: 127.97% of Ambient

As seen in *figure 3.83*, the device stayed in mode 1 for the entire testing period. This is because the passenger side was hotter than the driver side for the duration of the test, so the device takes air in from the driver side and exhausts it from the passenger side.

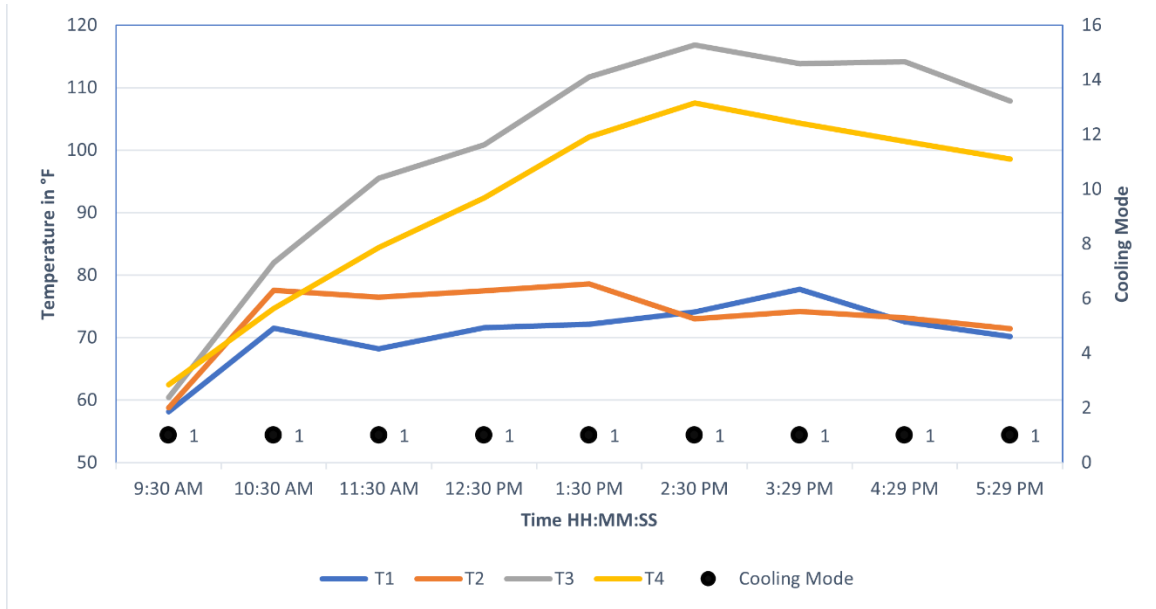


Figure 3.83 - Temperatures and Cooling Modes vs. Time. 3/20/2023

Test 4: 130% Spec Temperature Test on 3/25/2023

The maximum ambient temperature for the day was 91°F putting our 130% spec at 118.3°F. As seen in Table 3.10, the maximum internal temperature reached was 103.6°F putting us at 113.85% of the maximum ambient temperature.

Table 3.10 - Temperature Test 3/25/2023

TEST VEHICLE				
Time	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	Outside Ambient Temperature
9:30:00 AM	79.0	80.3	79.7	79.0
10:30:00 AM	89.9	86.8	88.4	83.0
11:30:00 AM	95.0	91.0	93.0	86.0
12:30:00 PM	95.2	97.4	96.3	89.0
1:30:00 PM	98.8	100.2	99.5	90.0
2:30:00 PM	100.5	102.1	101.3	91.0
3:30:00 PM	103.0	104.1	103.6	89.0
4:30:00 PM	102.2	102.0	102.1	88.0
5:30:00 PM	98.6	97.0	97.8	85.0
		Average Internal Temp:	95.7	

Charted in Figure 3.83 is the temperature data for the test vehicle over the day. The ambient temperature is also plotted for comparison and validation of results.

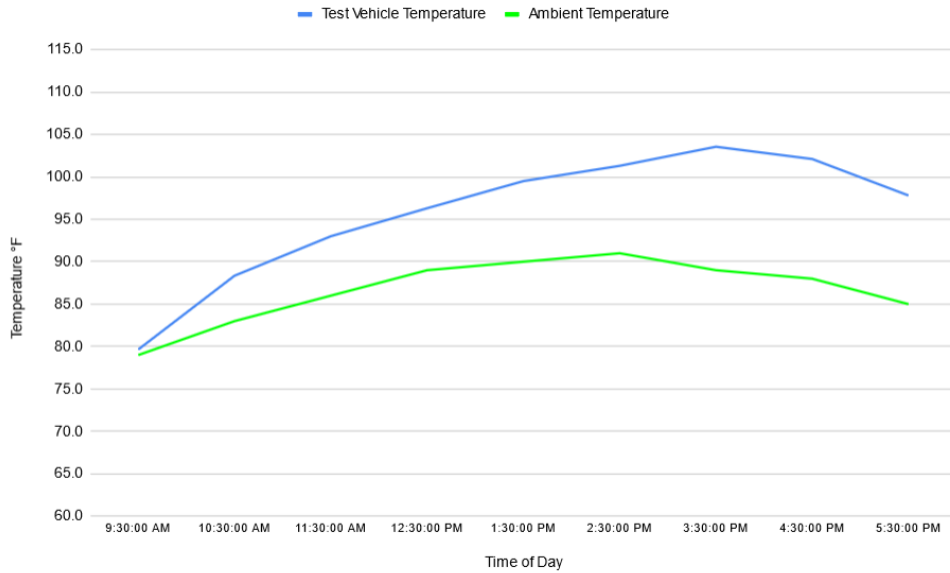


Figure 3.84 - Graph of Test Vehicle Temperature vs Time

TEST 4: PASS [✓] / FAIL [] 3/25/2023 MAX TEMP: 113.85% of Ambient

Rapid temperature changes occurred during this test. As shown in *figure 3.85*, the device starts in mode 2 as the passenger side is significantly cooler than the driver side. Around 12:30, the driver's side temperature falls below that of the passenger side, pushing the device into cooling mode 1. Around 2:30 PM, the difference between front and back temperatures exceeded the temperature difference between driver and passenger, and the front of the car became hotter than the back of the car. Thus, the MCU switched to cooling mode 4 to take air in from the back of the car and exhaust it from the front.

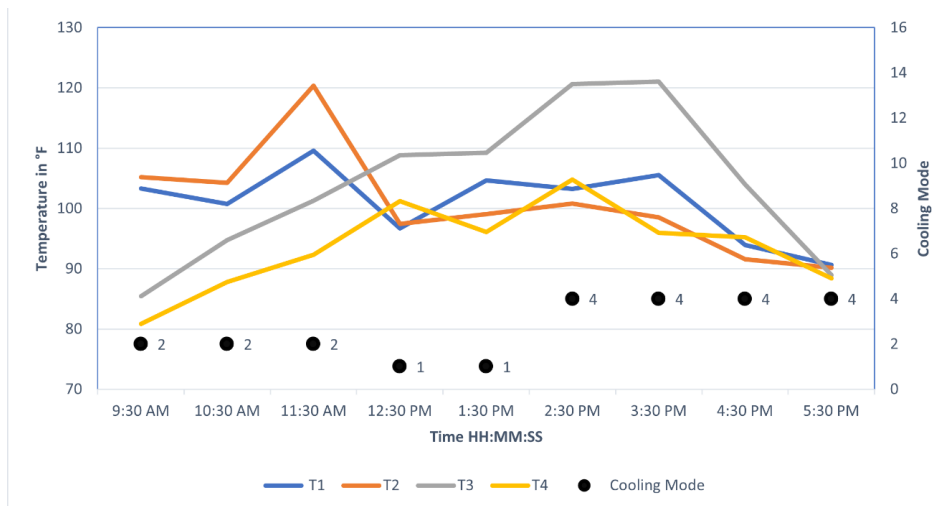


Figure 3.85 - Temperatures and Cooling Modes vs. Time. 3/25/2023

Test 5: Window Cracking Comparison Test

See the Window Cracking Comparison Test section for detailed Temperature data. The maximum internal temperature reached in the UUT was 103.8°F. With a maximum external ambient temperature of 91°F, this puts the 130% spec at 118.3°F. Therefore, the device once again met the requirement at 114.07% of the maximum outside ambient temperature. Notably, cracking the window provides comparable cooling to unoptimized crossflow as will be discussed in the Crossflow Comparison Test section of this report.

TEST 5: PASS [✓] / FAIL [] 3/26/2023 MAX TEMP: 114.07% of Ambient

During this test, the cooling modes followed the same path as the previous test. Since the weather conditions were similar and the orientation and positioning of the car remained the same, this shows that the cooling modes follow the path of the sun.

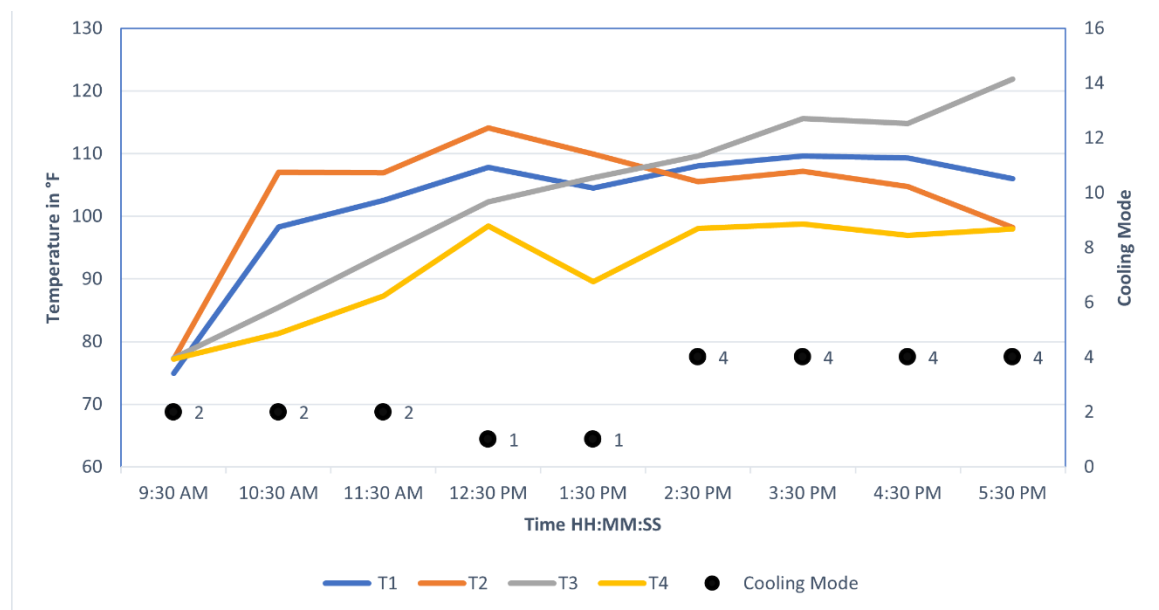


Figure 3.86 - Temperatures and Cooling Modes vs. Time. 3/26/2023

Sun Shield Comparison Temperature Test

To properly compare the cooling provided by the sun shield vs the UUT, we repeated the temperature test following our Test Setup but with the addition of a sun-shield placed in the front windshield of the reference vehicle.



Figure 3.87 - Reference Vehicle with Sunshield and Test Vehicle

The UUT ran for 8 hours from 9:30 am to 5:30 pm on 3/20/2023. This test was performed to show that the UUT provided more cooling than the sun-shield over 8 hours.

Table 3.11 - Sun Shield Comparison Test Results 3/20/2023

Time	TEST VEHICLE			Outside Ambient Temperature	REFERENCE VEHICLE		
	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp		Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp
9:30:00 AM	51.9	53.4	52.7	65.8	62.4	64.1	52.0
10:30:00 AM	59.9	60.3	60.1	68.7	65.3	67.0	56.0
11:30:00 AM	67.9	67.9	67.9	72.6	70.0	71.3	60.0
12:30:00 PM	73.5	76.9	75.2	77.1	76.0	76.6	63.0
1:30:00 PM	82.0	80.0	81.0	84.0	82.6	83.3	66.0
2:30:00 PM	85.7	84.0	84.9	90.6	87.5	89.1	68.0
3:30:00 PM	86.4	86.3	86.4	96.1	92.0	94.1	69.0
4:30:00 PM	88.6	88.0	88.3	96.8	96.8	96.8	69.0
5:30:00 PM	87.3	87.8	87.6	98.1	98.0	98.1	68.0
		Average Internal Temp:	76.0		Average Internal Temp:	82.2	

As seen in *Table 3.11*, the average internal temperature of the Reference vehicle with a sun-shield was 82.2°F while that of the Test Vehicle was 6.2°F lower at 76.0°F thus meeting our requirement for providing more cooling. It should be noted that the sun-shield was quite effective during the earlier hours of the day when the sun was hitting it directly but quickly became ineffective as the sun came in through the passenger windows on the reference vehicle. This can be seen in *Figure 3.87* where from 9:30 am until 11:30 am and at approximately 1:30 pm the sun was no longer hitting the sun-shield directly but instead was coming in through the uncovered side windows of the reference vehicle. The maximum internal temperature reached in the Test Vehicle was 88.3°F while

the reference vehicle hit a maximum of 98.1°F toward the end of the day. What this means is that the sun-shield is very effective when the sun is hitting it but otherwise becomes ineffective as soon as the sun enters the vehicle through another window.

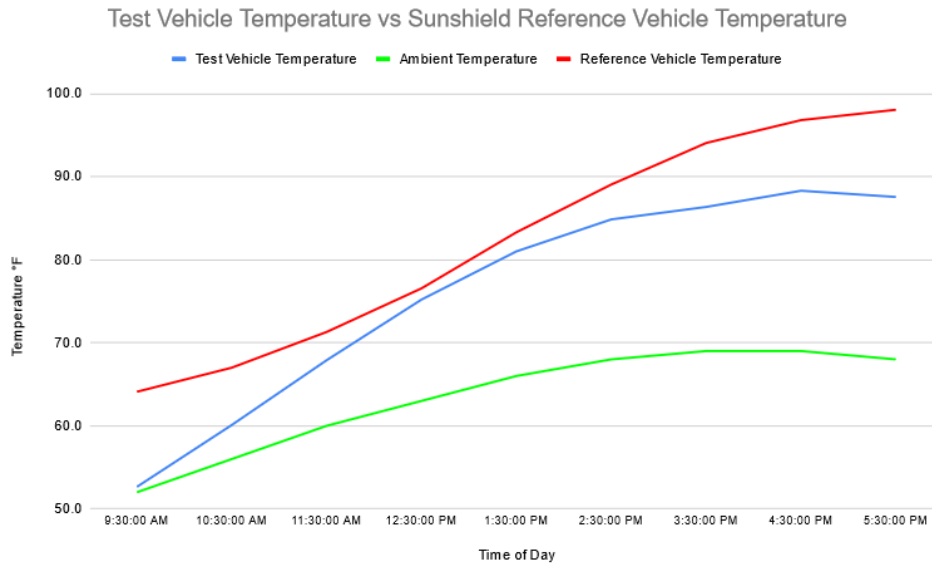


Figure 3.88 - Test Vehicle Temperature vs Sunshield Reference Vehicle Temperature

Window Cracking Comparison Temperature Test

Once again, we repeated the test setup described in section 3.6.1 but cracked the windows of the reference vehicle by the same amount as that of the UUT. That being the reference vehicle had its back windows cracked by 1.5 inches and the front windows by 4 inches.



Figure 3.89 - Reference Vehicle Window Cracking

The UUT ran for 8 hours from 9:30 am to 5:30 pm on 3/26/23. This test was performed to show that the UUT provided more cooling than simply cracking the windows over 8 hours.

Table 3.12 - Window Cracking Comparison Test Results 3/26/2023

TEST VEHICLE				REFERENCE VEHICLE			
Time	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	Outside Ambient Temperature
9:30:00 AM	75.2	76.0	75.6	83.0	81.8	82.4	75.0
10:30:00 AM	78.5	79.4	79.0	86.2	86.5	86.4	78.0
11:30:00 AM	82.9	84.1	83.5	91.6	88.9	90.3	83.0
12:30:00 PM	86.8	87.5	87.2	97.2	95.0	96.1	85.0
1:30:00 PM	92.5	93.2	92.9	102.0	105.4	103.7	89.0
2:30:00 PM	97.0	95.1	96.1	107.3	105.0	106.2	91.0
3:30:00 PM	99.5	99.8	99.7	105.3	106.9	106.1	90.0
4:30:00 PM	101.1	102.5	101.8	105.5	107.0	106.3	91.0
5:30:00 PM	102.6	105.0	103.8	106.0	110.3	108.2	90.0
		Average Internal Temp:	91.0		Average Internal Temp:	98.4	

As seen in *Table 3.12*, the average internal temperature of the Reference vehicle with the windows cracked was 98.4°F while that of the Test Vehicle was 7.4°F lower at 91.0°F thus meeting our requirement for providing more cooling. This difference may have been even greater as the data is slightly skewed. At 2:30 pm a shadow was cast on the front windshield of the reference vehicle by a nearby tree thus reducing sunlight exposure and reducing the overall increase in temperature for the reference vehicle. Notably, cracking the windows does still provide a fair amount of cooling. Using the temperature data from the next test on 3/27/2023 (shown in the next section) which has very comparable weather, we can see that the vehicle reaches an unregulated average internal temperature of 103.3°F. Meaning cracking the window provides about 4.9°F of cooling but the UUT still wins out by providing 12.3°F of cooling.

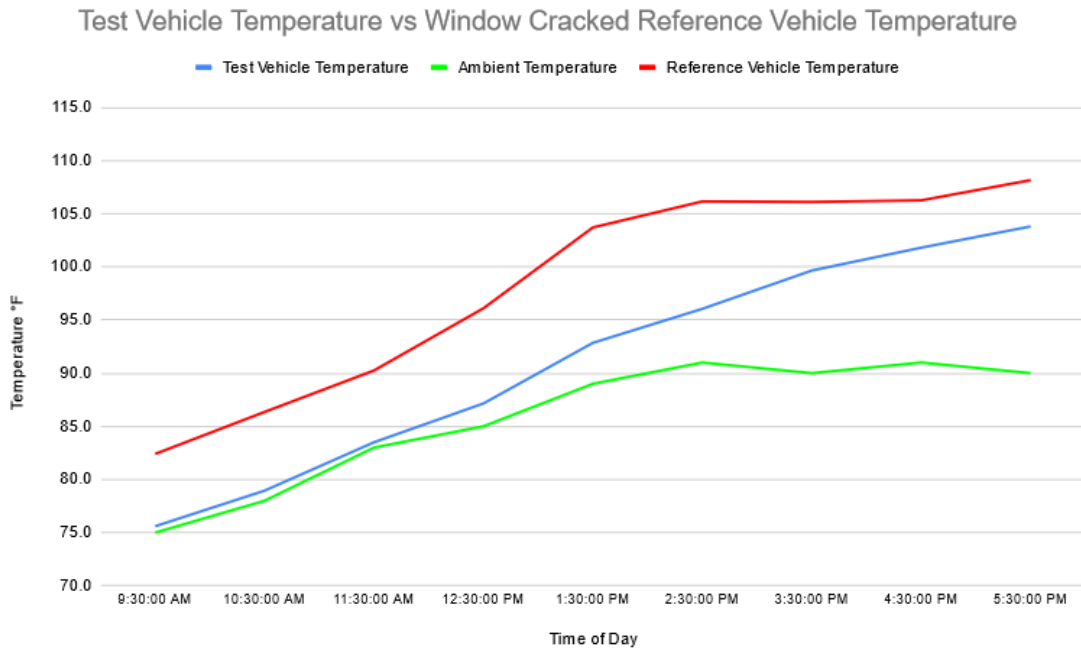


Figure 3.90 - Test Vehicle Temperature vs Cracked Window Reference Vehicle Temperature

Crossflow Ventilation Comparison Temperature Test

To test the validity of our method of cooling against non-optimized crossflow, the MCU software was modified for one day to only have the airflow of the vehicle intake from the passenger side and exhaust from the driver side. The temperature test was then repeated for comparison of temperature data against that of a previous test. The Window Cracking Comparison test performed on 3/26/2023 had remarkably similar weather conditions as that of the Crossflow Comparison Test performed on 3/27/2023 and was used as the data set to compare the cooling methods of our method of Optimized Crossflow against that of Non-Optimized Crossflow. Once again, the test ran for 8 hours from 9:30 am to 5:30 pm. To pass, the difference in cooling provided by the optimized crossflow must have been greater than or equal to 30% than that of non-optimized crossflow ventilation.

Table 3.13 - Optimized vs Non-Optimized Crossflow Temperature Test Results

Non-Optimized Test Vehicle				Reference Vehicle			
Time	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	Outside Ambient Temperature
9:30:00 AM	81.2	80.5	80.9	83.9	83.4	83.7	77.0
10:30:00 AM	82.7	83.1	82.9	89.2	84.9	87.1	80.0
11:30:00 AM	83.1	88.2	85.7	95.0	90.0	92.5	83.0
12:30:00 PM	90.2	93.3	91.8	95.0	101.0	98.0	85.0
1:30:00 PM	95.3	96.3	95.8	106.3	101.2	103.8	88.0
2:30:00 PM	100.1	104.8	102.5	111.9	108.5	110.2	90.0
3:30:00 PM	103.6	107.4	105.5	115.8	113.5	114.7	90.0
4:30:00 PM	103.8	107.3	105.6	118.0	123.0	120.5	91.0
5:30:00 PM	103.8	107.2	105.5	117.5	121.0	119.3	90.0
		Average Internal	95.1		Average Internal	103.3	
Optimized Test Vehicle (3/26/2023)				REFERENCE VEHICLE (3/26/2023)			
Time	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	Interior Temp Sensor 1	Interior Temp Sensor 2	AVG Internal Temp	Outside Ambient Temperature (3/26/2023)
9:30:00 AM	75.2	76.0	75.6	83.0	81.8	82.4	75.0
10:30:00 AM	78.5	79.4	79.0	86.2	86.5	86.4	78.0
11:30:00 AM	82.9	84.1	83.5	91.6	88.9	90.3	83.0
12:30:00 PM	86.8	87.5	87.2	97.2	95.0	96.1	85.0
1:30:00 PM	92.5	93.2	92.9	102.0	105.4	103.7	89.0
2:30:00 PM	97.0	95.1	96.1	107.3	105.0	106.2	91.0
3:30:00 PM	99.5	99.8	99.7	105.3	106.9	106.1	90.0
4:30:00 PM	101.1	102.5	101.8	105.5	107.0	106.3	91.0
5:30:00 PM	102.6	105.0	103.8	106.0	110.3	108.2	90.0
		Average Internal Temp:	91.0		Average Internal Temp:	98.4	

As seen in Table 3.13, the non-optimized crossflow resulted in an average internal temperature within the test vehicle of 95.1°F. On a similar day, the Optimized Crossflow resulted in an average internal temperature of 91.0°F. During this final test, the reference vehicle received no method of cooling and reached an average internal temperature of 103.3°F. Comparing the results between the two days, we can see that the cooling provided by Optimizing the Crossflow is greater than that of the non-optimized by 4.1°F. As stated previously, however, for the device to meet our requirements it must provide greater than or equal to 30% more cooling than the non-optimized variant. We can solve this using Equation 3.3.

Equation for Percentage Difference

$$\%Diff = \frac{|V_1 - V_2|}{\left|\frac{V_1 + V_2}{2}\right|} \times 100\% \quad (3.13)$$

For our results, we can calculate the percentage difference by substituting V_1 with our average cooling amount provided by the optimized crossflow on 3/26/2023 of 12.3°F and V_2 with our average cooling provided by the non-optimized crossflow on 3/27/2023 of 8.2°F.

$$\%Diff = \frac{|12.3^{\circ}F - 8.2^{\circ}F|}{\left| \frac{12.3^{\circ}F + 8.2^{\circ}F}{2} \right|} \times 100\% = 40\%$$

With this information, we see that our device provides 40% more cooling than non-optimized crossflow on a day with comparable weather. This data is further corroborated by viewing the temperature differences as shown in *Figure 3.90*.

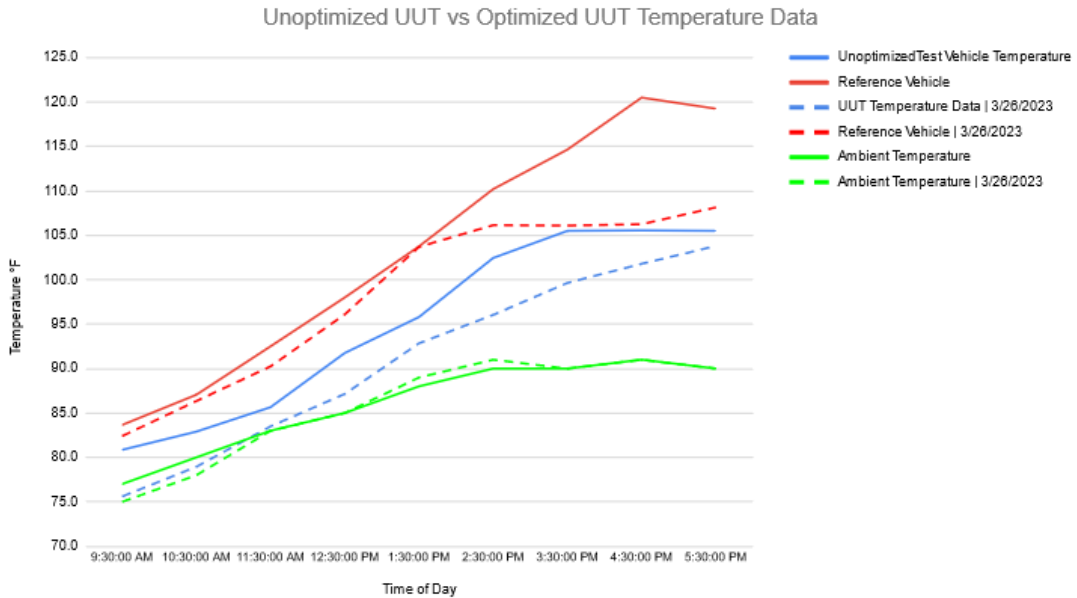


Figure 3.91 - Graph of Optimized Crossflow vs Non-Optimized Crossflow Temperatures

Battery Life Duration Test

To verify that our device could run for 8 hours or more, the battery level was monitored throughout the testing period for our Temperature Tests. To pass, the battery must maintain a voltage above 12V for the unit to remain active, the lowest battery level reached while the solar panel was wired into the PWM Charge Controller was logged as the pass/fail criteria for the UUT at 12.7V on 3/20/2023. **So long as the Solar Panel is plugged in, the device will not drop to 12.0V while the sun is out.**

Table 3.14 – PWM Voltages During Temperature Tests

Time	V_PWM 3/14	V_PWM 3/20	V_PWM 3/26	V_PWM 3/27
9:30:00 AM	12.9	12.7	13.3	13.3
10:30:00 AM	13.4	12.7	13.7	13.7
11:30:00 AM	13.7	13.2	14.4	13.7
12:30:00 PM	14.4	13.3	14.4	14.4
1:30:00 PM	14.4	13.3	14.4	14.4
2:30:00 PM	14.4	13.7	14.4	14.4
3:30:00 PM	14.4	13.7	14.4	14.4
4:30:00 PM	14.4	13.7	13.1	13.7
5:30:00 PM	13.0	13.5	13.1	13.7

Rain Rejection Test

From March to April, we have had remarkably little rain as compared to previous years. According to Spectrum News 13, Orlando is currently experiencing a deficit of rain of more than 5 inches since the start of 2023 and more than seven inches going back to December 1st of 2022 [47]. Given this information, our window for testing for rain rejection was restricted greatly. To attempt to combat this, a drive to Tallahassee FL on a weekend where it was purported to have storms was taken. While there, the rain was just as sporadic as in Orlando with only a 1-hour window presenting itself with a significant amount of rainfall which was used to determine the effectiveness of the UUT to reject rain. The unit was confirmed active before the start of the testing period. At the end of the testing period, [visual inspections](#) were performed to determine if the rain had entered the vehicle in any significant amount and whether the electronics held up.

A small amount of rain entered through the front windows where the vent visors themselves had a gap between the body of the vent visor and the body/window of the car. Given that these vent visors were not designed by the team, but a prefabricated item that had this flaw, the rain which entered the vehicle was not deemed a design flaw of the system as the cooling module itself did not bring water into the vehicle. The back windows which had full coverage from the vent visors did not see any water enter the vehicle. If the front vent visors had been longer to cover the gap, it is reasonable to assume they too would've kept the rain out. The rain which did enter was not a large amount and was comparable to opening your car door during a heavy downpour in terms of the overall amount. No damage to the inside of the vehicle nor electronics on the UUT was noted.

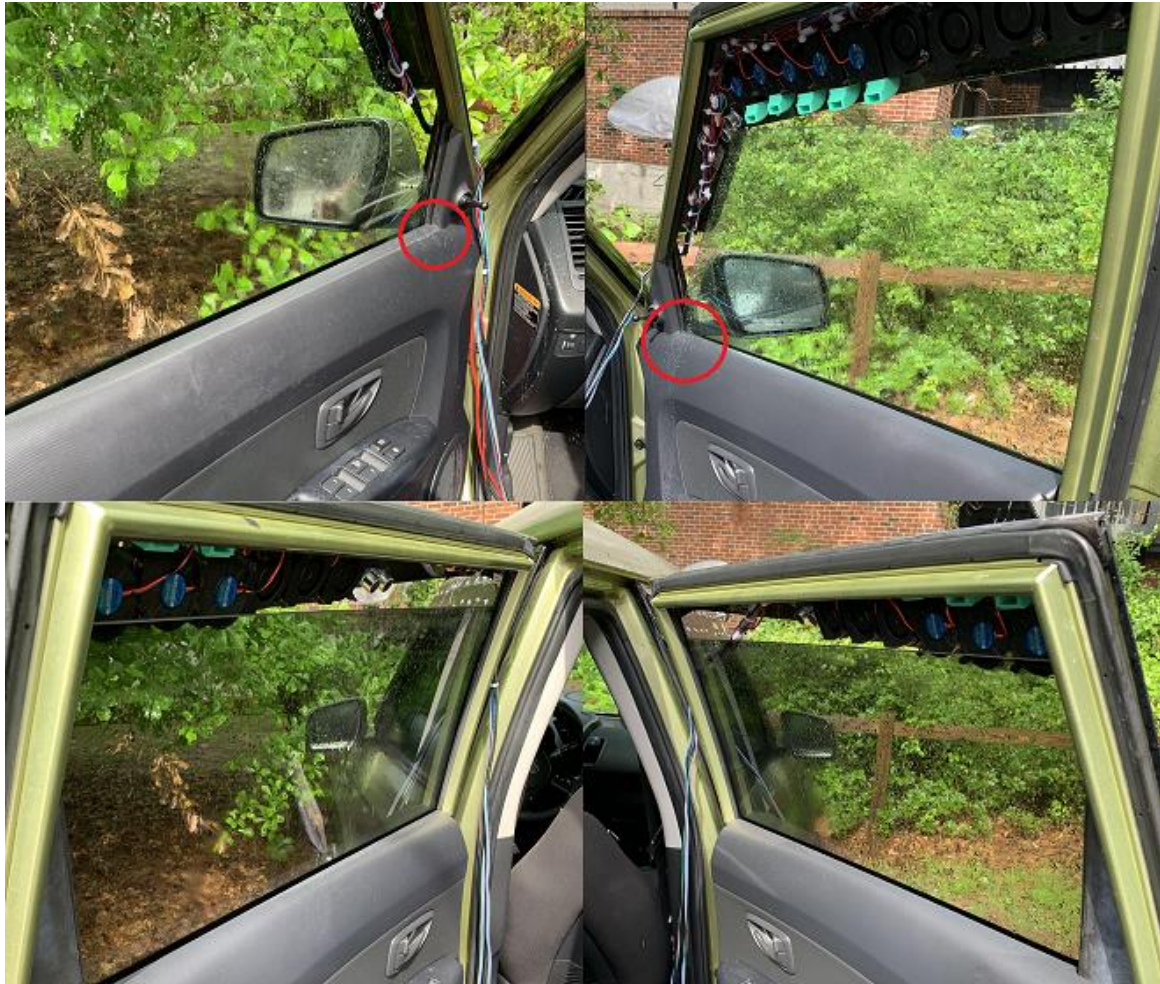


Figure 3.92 - Interior Inspection Post Rainfall

Structural Integrity Test

After installing the Cooling Module to the car in a semi-permanent manner using the factory-supplied adhesive strips, the vehicle was driven from Apopka, FL to Tallahassee, FL following I-75N to I-10W. This test was performed to show that the vent visors could handle a high-speed drive and remain intact following exposure to the forces subjected to them by high-speed air passing over and around them. On these interstate highway systems, the speed limit is 70mph with a minimum speed of 50mph. [During testing](#), the maximum speed hit while driving was 80mph for brief periods but a sustained average of 70mph for the trip.

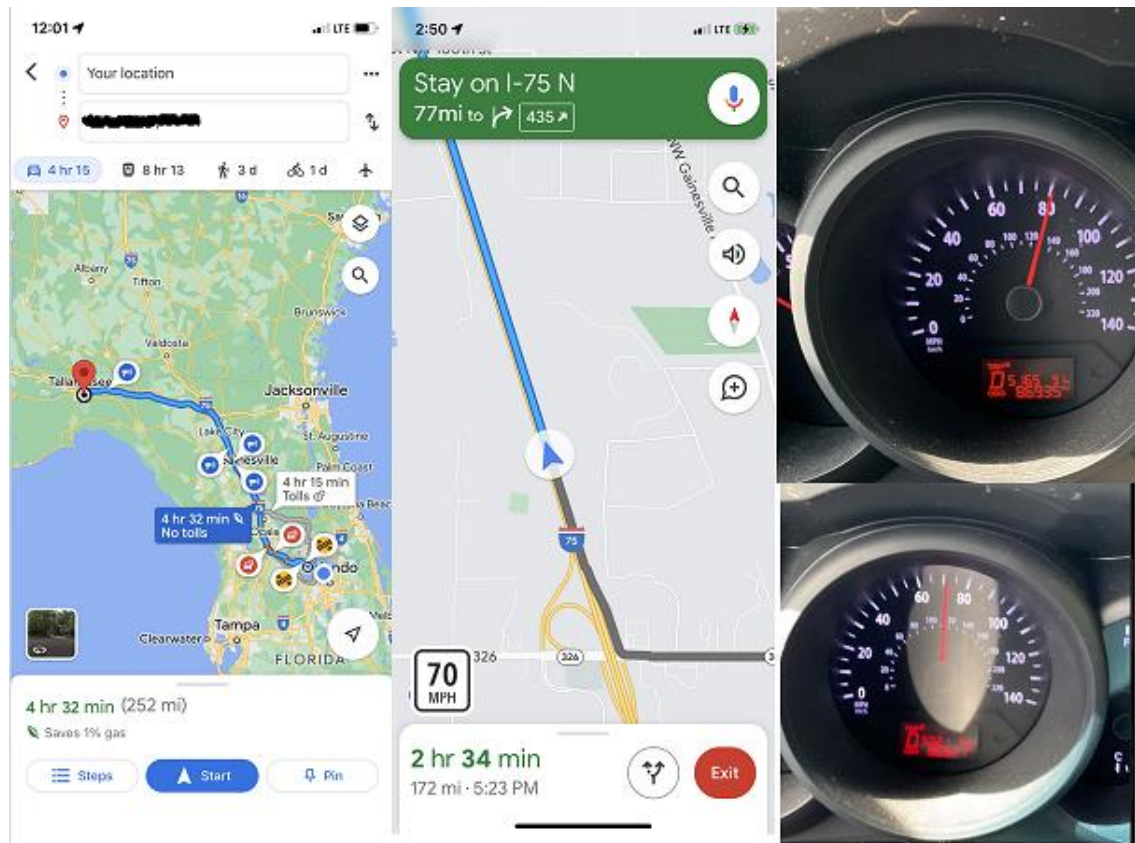


Figure 3.93 - Route Taken and Speed Observed

Upon arrival at the destination in Tallahassee FL, a visual inspection was performed to determine if any damage had been incurred during the trip. No signs of damage were present and the UUT was activated to confirm functionality post-drive.



Figure 3.94 - Visual Inspection of Vent Visors and Activation of UUT

3.6.3 Medium-Level Requirement Testing

Battery Voltage Monitoring and Switching Test

To verify that the number of active fans during the testing period would decrease in proportion to the voltage supplied by the battery of the car, the solar panel was disconnected from the PWM charge controller and detached from the UUT. The unit would run continuously and the number of fans active would be charted based on the battery level and time.

Table 3.15 - Fans Active with Decreasing Battery Level

Time	V_PWM	Battery Level	Fans Active
5:30:00 PM	12.6V	3	Front : 10 Back : 6
6:00:00 PM	12.5V	2	Front : 8 Back : 4
7:00:00 PM	12.4V	2	Front : 8 Back : 4
8:00:00 PM	12.3V	2	Front : 8 Back : 4
9:00:00 PM	12.3V	2	Front : 8 Back : 4
10:00:00 PM	12.3V	2	Front : 8 Back : 4
11:00:00 PM	12.3V	2	Front : 8 Back : 4
12:00:00 AM	12.3V	2	Front : 8 Back : 4
1:00:00 AM	12.2V	1	Front : 4 Back : 2
8:00:00 AM	12.1V	1	Front : 4 Back : 2
9:00:00 AM	12.1V	1	Front : 4 Back : 2
10:00:00 AM	12.1V	1	Front : 4 Back : 2
11:00:00 AM	12.1V	1	Front : 4 Back : 2
12:00:00 PM	12.1V	1	Front : 4 Back : 2
1:00:00 PM	12.1V	1	Front : 4 Back : 2
2:00:00 PM	12.1V	1	Front : 4 Back : 2
3:00:00 PM	12.1V	1	Front : 4 Back : 2
4:00:00 PM	12.1V	1	Front : 4 Back : 2
5:00:00 PM	12.1V	1	Front : 4 Back : 2
6:00:00 PM	12.1V	1	Front : 4 Back : 2
7:00:00 PM	12.1V	1	Front : 4 Back : 2
8:00:00 PM	12.1V	1	Front : 4 Back : 2
9:00:00 PM	12.1V	1	Front : 4 Back : 2

Battery Level 3 corresponds to a voltage of 12.6 V or greater across the battery. Under these conditions 16 total fans are active. Below 12.6 V to 12.3 V, 12 fans are active. Below 12.3 V 6 fans are active. Lastly, at 12 V, 0 fans are active. As stated in the troubleshooting section, the MCU initially failed to disable the load when the battery voltage fell to 12 V or below. This was corrected by modifying the program run by the MCU. The functionality was tested in the next section in the automatic run/shutoff capability test.

Automatic Run / Shutoff Capability Test

To verify that the load would not activate following the battery voltage reaching 12V, the unit was allowed to run for as long as needed for the battery to discharge to 12V with the solar panel disconnected from the PWM Charge Controller, so it did not recharge. Following the testing done on 3/27/2023, the device code was modified briefly to have the fans run at maximum power to run down the battery as quickly as possible to confirm functionality as well as a fix for the battery mode code relating to load shutoff by Ian. The device ran from 6:12 PM on 3/27/2023 to 8:49 AM on 3/28/2023 before shutting down at 12.0V on the PWM read as 12.01V on the DMM.



Figure 3.95 - PWM and DMM Voltage at Shutoff

Plugging in the solar panel, the device quickly began to charge with the PWM hopping to 12.1V and the fans reactivating at 12.15V.



Figure 3.96 - PWM Voltage Post Re-connection of Solar Panel

Window Obstruction Test

The UUT had a variety of wires and sensors attached to it which may have interfered with the operation of the windows in their ability to open and close. To verify that the UUT did not obstruct the window as it opened and closed, a [recording](#) was taken to show that the UUT did not prevent the window from closing nor did any wiring from the UUT get pinched by the windows.



Figure 3.97 - Window Obstruction Test Before and After

Load Shutoff / Window Condition Test

When any of the windows of the vehicle are rolled up past a point, the intake fans cannot get air into the vehicle thus the load will not activate as airflow optimization cannot be achieved if any of the 4 windows are rolled up. To confirm this, the windows would be toggled up and down individually to confirm if each window could disable the load if rolled up. Shown in *Table 3.16* are our [results](#).

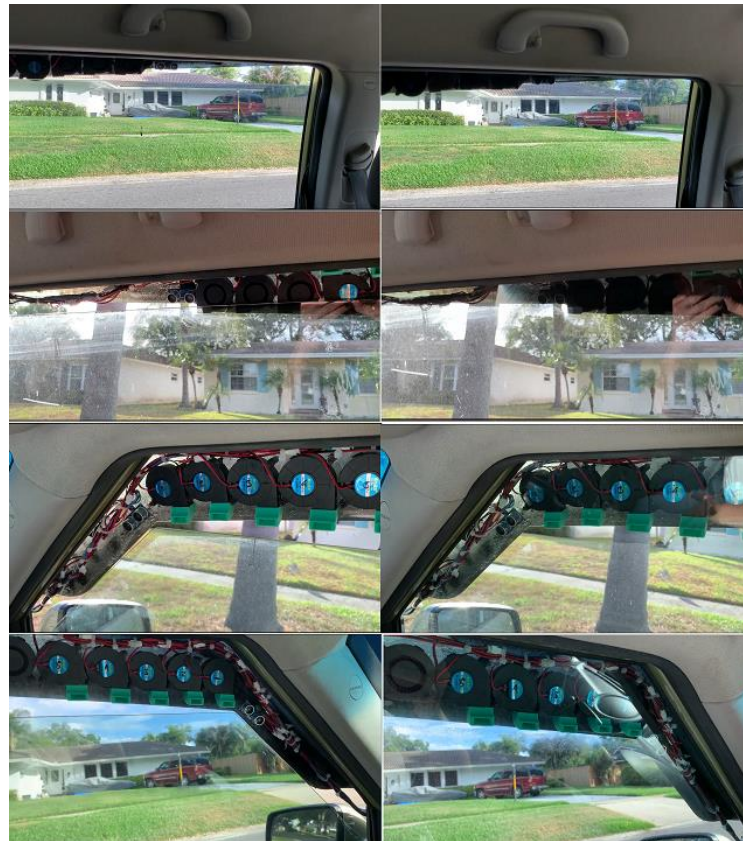


Figure 3.98 - Load Shutoff Check Windows Up/Down

Table 3.16 - Load Shutoff Test Results

WINDOW	WINDOW STATUS	Cooling Module Status
1	UP	OFF
1	DOWN	ON
2	UP	OFF
2	DOWN	ON
3	UP	OFF
3	DOWN	ON
4	UP	OFF
4	DOWN	ON

3.6.4 Low-Level Requirement Testing

Visual Obstruction Test

For a user to drive safely, the UUT must not obstruct the vision of the driver. Specifically, the driver must be able to see out all windows, see out of the rear-view mirror, and be able to view the rear-view mirrors on each side of the vehicle. To confirm, a camera is used to take POV shots from the perspective of the driver. Shown below is the POV of the Front Seat from the view of the driver.



Figure 3.99 - Driver Front Seat POV

The driver can see out of both the front passenger and driver windows. The front-facing windshield does not have any obstruction from the installation of the cooling module. The rear-view mirrors are completely visible from the perspective of the driver and no wiring blocks the view out of the rear-view window.

The Driver is also capable of peering out of each of the back windows for merging while driving. As shown below, the cooling module does not hinder the vision of the driver.



Figure 3.100 - Driver Back Windows POV

LED Indicator Test

To indicate to the user that all windows are rolled down, thus the UUT is ready for use, a green indicator LED will activate if all windows are rolled down and be inactive if any window is rolled up. Photos for each condition were taken for confirmation.



Figure 3.101 - LED Indicator Test Window Positions

For each of the above positions, the window location is charted, and the status of the LED is indicated in *Table 3.17*.

Table 3.17 - LED Indicator Test Results

WINDOW	WINDOW STATUS	LED INDICATOR
1	Up	OFF
1	Down	ON
2	Up	OFF
2	Down	ON
3	Up	OFF
3	Down	ON
4	Up	OFF
4	Down	ON

3.6.5 Summary of Test Results

Table 3.18 gives a list of all conducted tests and whether or not each test passed or failed the criteria. The project passed all tests. Thus, the engineering requirements for the project have been met.

Table 3.18 – Summary of Test Results

TEST NAME	PASS	FAIL
130% Spec Temperature Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Sun Shield Comparison Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Window Crracking Comparison Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Crossflow Ventilation Comparison Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Battery-Life Duration Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Rain Rejection Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Stuctural Integrity	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Battery Voltage Monitoring and Switching Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Automatic Run / Shutoff Capability Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Window Obstruction Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Load Shutoff / Window Condition Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Visual Obstruction Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
LED Indicator Test	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Chapter 4

Non-Technical Issues

Summary

In this chapter, we will discuss the non-technical aspects of the proposed project such as the budget, environmental aspects, social aspects, ethical considerations, and sustainability.

- 4.1 Budget and Timeline**
- 4.2 Environmental Aspects**
- 4.3 Health and Safety**
- 4.4 Ethical Aspects**
- 4.5 Social Aspects**

4.1 Budget and Timeline

This section gives an overview of the project budget and timeline.

4.1.1 Budget

Table 4.1 gives the project budget. The budget includes all materials which were used in the project. It is broken down into project components and testing materials and gives totals for each of these as well as a grand total. As stated in section 1.6, the project has been funded by IEEE Region 3, which reimbursed the group for the proposed budget of \$526.34, nearly all of the spent budget.

Table 4.1 – Project Budget

Item	Unit Price	Qty.	Vendor	Subtotal
Project Components				
Kia Soul Sport Visors OEM U8220 2k000	\$28.95	1	eBay	\$28.95
Brushless 12V DC Blower (4-Pack)	\$7.99	11	Amazon	\$87.89
100W Solar Panel	\$79.99	1	Solar Savings World	\$79.99
PWM Charge Controller for Lead Acid Battery	\$12.98	1	Amazon	\$12.98
Boost Converter 8.5V-50V to DC 10V-60V	\$11.99	1	Amazon	\$11.99
SPDT Switch	\$0.82	1	Digi-Key	\$0.82
Raspberry Pi Pico	\$5.00	1	AdaFruit	\$5.00
Raspberry Pi Pico Replacement	\$12.79	1	Amazon	\$12.79
ADAFRUIT PCF8575 I2C 16 GPIO EXP	\$5.95	1	Digi-Key	\$5.95
Thermistor (5-Pack)	\$8.43	1	Amazon	\$8.43
HC-SR04 Ultrasonic Distance Sensor (5pcs)	\$11.39	1	Amazon	\$11.39
2N3904 Transistors (100pcs)	\$5.99	1	Amazon	\$5.99
1N4001 Diodes (100pcs)	\$5.99	1	Amazon	\$5.99
TERMINAL BLOCK, FEMALE, 11 Pin	\$3.09	1	Digi-Key	\$3.09
TERMINAL BLOCK, MALE, 11 Pin	\$1.32	1	Digi-Key	\$1.32
TERMINAL BLOCK, FEMALE, 12 Pin	\$3.01	1	Digi-Key	\$3.01
TERMINAL BLOCK, MALE, 12 Pin	\$1.44	1	Digi-Key	\$1.44
TERMINAL BLOCK, FEMALE, 14 Pin	\$3.51	2	Digi-Key	\$7.02
TERMINAL BLOCK, MALE, 14 Pin	\$1.68	2	Digi-Key	\$3.36
Outdoor Electrical Box	\$24.99	1	Amazon	\$24.99

10 Gauge Wire (10ft)	\$19.99	1	Amazon	\$19.99
22 AWG 4 Conductor Wire 32.8 Ft	\$11.99	1	Amazon	\$11.99
26 AWG 5 Conductor Wire 25 Ft.	\$14.99	1	Amazon	\$14.99
Colored 22 AWG Wire	\$16.95	1	Amazon	\$16.95
IP67 Solar Panel Extension Cable	\$19.99	1	Amazon	\$19.99
4-pin IP65 Connectors (5 Pairs Male/Female)	\$12.99	1	Amazon	\$12.99
5-Pin IP65 Connectors (5 Pairs Male/Female)	\$11.99	1	Amazon	\$11.99
Mini-Micro 2.54mm 2 Pin Connectors (100pcs)	\$13.99	1	Amazon	\$13.99
M3 Hex Male Brass Standoff Set	\$11.99	1	Amazon	\$11.99
Loctite Threadlocker Blue 242 Nut/Bolt Locker	\$7.09	1	Amazon	\$7.09
Quick Setting Epoxy	\$1.99	2	Harbour Freight	\$3.98
Electronic Components (Assorted Resistors, Capacitors, Diodes, LEDs, IR LED, ICs)	\$7.00	1	Supplied by Group	\$7.00
18 Gauge Wire	\$7.00	1	Supplied by Group	\$7.00
Screws and Nuts	\$10.00	1	Supplied by Group	\$10.00
Total Cost of Components				\$492.33
Testing Materials				
Temperature Sensors (4PC, Internal)	\$13.99	2	Amazon	\$27.98
Temperature Sensors (2PC, External)	\$10.49	2	Amazon	\$20.98
Car Windshield Sunshade	\$18.99	1	Amazon	\$18.99
Total Cost of Testing Materials				\$48.96
Grand Total				\$541.29

4.1.2 Timeline

Table 4.2 outlines the plan of action for the design phase of the project. This is given by a Gantt chart, which breaks down the timeline by each task as it aligns with each week. *Table 4.3* gives a more detailed description of what the group aimed to accomplish during each week.

Table 4.2 – Project Design Phase Timeline

Month:	December				January					February				March				April		
Week:	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
Week of:	12/5	12/12	12/19	12/26	1/2	1/9	1/16	1/23	1/30	2/6	2/13	2/20	2/27	3/6	3/13	3/20	3/27	4/3	4/10	4/17
Tasks																				
Secure Materials	█	█	█																	
Sensor Circuit Design		█	█																	
Sensor Construction			█	█																
Read Sensors Programming			█	█																
Sensor Testing				█	█															
Vent-Visor Fan Array Assembly				█	█															
Fan & Relay Wiring					█	█														
Fan Control Programming					█	█	█	█												
Full System Assembly						█	█													
Testing and Verifying Power Module							█	█	█											
Testing and Verifying Fan Control								█	█	█	█									
Final Testing												█	█	█	█	█	█			
Report Writing															█	█	█	█	█	
Presentation Rehearsal																	█	█		
Design Report Due																			█	
Design Presentation																				█

Table 4.3 - Proposed Timeline by Week

Week	Group Objectives
Week 16: 12/5 - 12/12	<ul style="list-style-type: none"> • Ordering all materials (Winter break)
Week 17: 12/12 - 12/19	<ul style="list-style-type: none"> • Ordering all materials and confirming arrival (Winter break) • Design thermistor circuit, IR sensor circuit, and battery sensor circuit
Week 18: 12/19 - 12/26	<ul style="list-style-type: none"> • Finish sensor design. Begin constructing sensor circuits. • Begin programming MCU to read all sensors and switch.
Week 19: 12/26 - 1/2	<ul style="list-style-type: none"> • Finish sensor construction. Begin assembling vent-visor fan arrays. • Finish sensor read programs. Begin testing the sensors with MCU. Test GPIO expander.
Week 20: 1/2 - 1/9	<ul style="list-style-type: none"> • Finish vent-visor fan construction. Begin wiring fans and relays. • Finish sensor testing. Begin programming the fan control software.
Week 21: 1/9 - 1/16	<ul style="list-style-type: none"> • Wrap up fan and relay wiring. Start assembling the power module and control module. • Continue fan control programming.
Week 22: 1/16 - 1/23	<ul style="list-style-type: none"> • Have the system fully assembled for testing. Begin testing the power module. • Finish programming fan control software. Integrate all programs in the main program.
Week 23: 1/23 - 1/30	<ul style="list-style-type: none"> • Continue testing the power module. Test PWM controller with solar panel and battery. • Begin testing the fan control module.
Week 24: 1/30 - 2/6	<ul style="list-style-type: none"> • Gather data on solar panel power production. • Continue testing fan control. Verify airflow modes.
Week 25: 2/6 - 2/13	<ul style="list-style-type: none"> • Continue testing fan control and finish verifying airflow modes.
Week 26: 2/13 - 2/20	<ul style="list-style-type: none"> • Finalize system testing. Everything must work by the end of this week.
Week 27: 2/20 - 2/27	<ul style="list-style-type: none"> • Begin cooling tests. Verify High-Level requirements.
Week 28: 2/27 - 3/6	<ul style="list-style-type: none"> • Continue cooling tests. Verify High-Level requirements.
Week 29: 3/6 - 3/13	<ul style="list-style-type: none"> • Continue cooling tests. Verify Medium-Level requirements. • Begin writing the report. Abstract and Chapter 1
Week 30: 3/13 - 3/20	<ul style="list-style-type: none"> • Continue cooling tests. Verify Medium-Level requirements. • Continue writing the report. Chapter 2
Week 31: 3/20 - 3/27	<ul style="list-style-type: none"> • Continue cooling tests. Verify Low-Level requirements. • Continue writing reports. Chapters 3 and 4.
Week 32: 3/27 - 4/3	<ul style="list-style-type: none"> • Wrapping up testing. • Continue writing the report. Table of Contents, Figures, and Tables. • Begin working on the presentation
Week 33: 4/3 - 4/10	<ul style="list-style-type: none"> • Finish report. • Finish presentation.
Week 34: 4/10 - 4/17	<ul style="list-style-type: none"> • Submit the report. Rehearse presentation.
Week 35: 4/17 - 4/24	<ul style="list-style-type: none"> • Give the design presentation.

Table 4.4 – Actual Timeline

Week	Group Objectives
Week 16: 12/5 - 12/12	<ul style="list-style-type: none"> - Ordered the first set of materials and had them delivered to each group member based on their respective responsibilities.
Week 17: 12/12 - 12/19	<ul style="list-style-type: none"> - Confirmed Arrival of materials, some would not arrive until the start of January. - Thermistor and Battery Level Monitoring Circuit Designed and tested in Multi-Sim. - Confirmed the function of Raspberry Pi PICO and PCF8575 Expander.
Week 18: 12/19 - 12/26	<ul style="list-style-type: none"> - Tested Thermistor and Battery Level Monitoring Circuit on a breadboard and confirmed voltage differences would be sufficient for reading by the Raspberry Pi PICO ADC. - Coded relay triggering program and program for determining the battery charge level for control of fans. - Temperature sensor circuit testing continued.
Week 19: 12/26 - 1/2	<ul style="list-style-type: none"> - Reworked Battery Level Monitoring Circuit, maximum charge of battery could be as high as 15 V, and voltage division needed to be adjusted. - Refined battery level code based on battery level new monitoring circuit.
Week 20: 1/2 - 1/9	<ul style="list-style-type: none"> - Identified the need to simplify the battery voltage divider. - Use of optocoupler for voltage isolation of thermistor circuit discussed. - Current draw of relays requires a solution as the 3V3 pin of the PICO would not provide sufficient current
Week 21: 1/9 - 1/16	<ul style="list-style-type: none"> - Reworked Temperature Sensor circuit and code with electrical isolation for proper reading. - Simplified the battery voltage divider and updated the code to reflect the change in the number of fans used. - Began Initial Construction of Vent Visor Arrays, only 32 of 44 capable of mounting to vent visors. - Solar Panel and PWM Charge Controller Tested, need for voltage regulator identified. - 3.3V signal for relays needed, LD33V solution tested. A heating issue was identified.
Week 22: 1/16 - 1/23	<ul style="list-style-type: none"> - Sensors codes consolidated into one program. - Began soldering and wiring the control module and sensors. - Attached Vent Visors to the vehicle using clips to verify fans do not obstruct the opening/closing of windows. - Voltage Regulator Tested, confirmed stable 12V output. - Power Consumption test of fans confirmed that the current draw of fans is far lower than initially thought. - Current draw of relays was identified as a potential issue and discussed the transition to using transistors, given that each fan draws less current than anticipated.
Week 23: 1/23 - 1/30	<ul style="list-style-type: none"> - PICO and Expander soldering completed - Connection Planning for sensors/vent visors to control module discussed - Transistor Solution simulated and tested using 2N3904 Transistor

	<ul style="list-style-type: none"> - IP65 Cables and Transistors Ordered - Reverse Voltage issue identified during simulation, need for flyback diode identified
Week 24: 1/30 - 2/6	<ul style="list-style-type: none"> - Sensor Cabling, MCU connections, terminal blocks, and transistor array soldered for fan control. - Flyback Diode 1N4001 Tested and confirmed viable. - Ordered Diodes for soldering and additional 4 Wire Cables for connection to fans. - Mounting Board sized and drilled for Prototype Board Mounting.
Week 25: 2/6 - 2/13	<ul style="list-style-type: none"> - Tested soldered boards with Thermistor Circuits, IR Sensors, Battery Level Monitoring, and Fan Controls - Pico damaged by 12V power supply hitting pin during testing: Re-Ordered a board for further tests and installed the new board. Verified temp and IR sensor operation.
Week 26: 2/13 - 2/20	<ul style="list-style-type: none"> - Tested, troubleshot, and fixed fan control circuits. - Tested and troubleshot all functions of code for functionality with sensors. - Mock test performed wiring unit to the vehicle with Solar Panel and Charge Controller - IR sensor issue identified.
Week 27: 2/20 - 2/27	<ul style="list-style-type: none"> - IR Sensor issue identified previously had troubleshooting done, confirmed saturation issue of photodiodes - Began printing and testing of nozzles for intake fans, and confirmed sizing against windows did not cause obstruction. - Revisions will be made to nozzles for additional clearance from windows and to increase airflow.
Week 28: 2/27 - 3/6	<ul style="list-style-type: none"> - Use of Ultrasonic Sensors proposed and tested, viable solution - Code added for monitoring of temperature, cooling mode, battery level, etc. over the testing period - Baseline test of UUT done. Confirmed the issue with Thermistor on Windows 2 and 4.
Week 29: 3/6 - 3/13	<ul style="list-style-type: none"> - Modified fan position - Redesigned and reprinted intake nozzles for the rear windows - Implemented, tested, and troubleshot Ultrasonic Proximity Sensors - Began writing the report and Acceptance Test Procedure
Week 30: 3/13 - 3/20	<ul style="list-style-type: none"> - Began full system temperature testing. - Began battery level testing. - Fixed the Get Battery Mode function issue.
Week 31: 3/20 - 3/27	<ul style="list-style-type: none"> - Continued writing the report. - Completed sun shield comparison and cracked window comparison tests. - Continued temperature testing.
Week 32: 3/27 - 4/3	<ul style="list-style-type: none"> - Continued writing the report. - Completed non-optimized crossflow comparison test and structural integrity test. - Fixed another Get Battery Mode function issue and verified the full functionality of the battery monitor and battery modes.

Week 33: 4/3 - 4/10	- Continued writing the report. - Completed rain test and a reference vehicle control test.
Week 34: 4/10 - 4/17	- Finished writing the report and submitted it. - Created the PowerPoint presentation
Week 35: 4/17 - 4/24	- Rehearsed the presentation - Gave the presentation

4.2 Environmental Aspects

Solar-powered ventilation for parked cars is a very environmentally friendly project, as it has the potential to reduce the power consumption of the AC of the car [48], thus reducing CO₂ emissions for gas cars and extending their range. The carbon footprint of manufacturing the device is minimal. Solar panels become carbon neutral after about three years of use [49]. Vent visors are made of acrylic, which is not biodegradable but can be recycled [50]. The proposed DC blower fans are made of plastic and can be recycled upon failure [51]. As proof of concept, any materials used for the construction of the device are subject to change if desired by the manufacturer.

4.3 Health and Safety

The Health and Safety of all involved in the project were a top priority. Given the high current that can potentially be drawn by the battery of the car, up to 15A, and the high output current of the solar PV array, up to 5.23A, extreme caution was taken when wiring the system together being sure that no potential electrical shorts or contacts were present that could endanger an individual using or working on the device. Given the body of the car is metallic, extra care was taken to ensure that the wiring was properly shielded.

4.4 Ethical Aspects

This report follows IEEE Standards which set the standards for our ethical behavior and citation format. This report follows the IEEE citation standards for all information that requires a citation, all of which are found in a reference table at the end of this report. The Code of Ethics as established by IEEE was adhered to by all members of the team during every step of the implementation of this project. We have an ethical responsibility to hold paramount the safety, health, and welfare of the public, to accept and seek honest

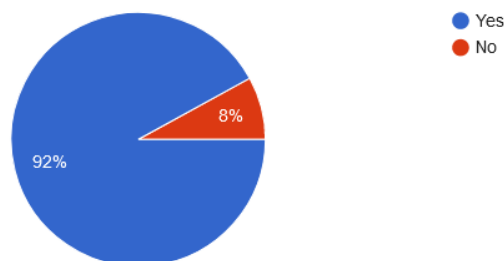
criticism of our work, and of course, we must support colleagues in following the code of ethics at all points during and after this project [52].

4.5 Social Aspects

Regarding the potential impact of our proposed system on society at large, it is difficult to gauge the full ramifications of such a device, however, we sent out a survey to try and understand if this device would be received well. The full survey can be found on our website. Of the 25 participants, 92% of the respondents said they would be interested in a device that would remove heat from their parked vehicles by ventilating air through the windows. Further breaking the data down, 40% said they would prefer the device to come pre-installed, 20% preferred aftermarket, and the last 40% would like either, as shown below.

2) Would you be interested in a device which could remove heat from the vehicle while parked by ventilating air through the windows?

25 responses



3) Would you prefer to buy such a device after-market and install it yourself? Or would you prefer it to come pre-installed by the manufacturer of the vehicle?

25 responses

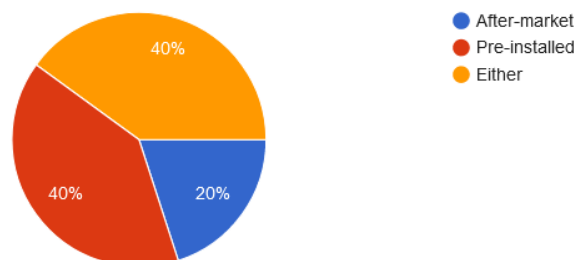


Figure 4.1 - Two Survey Responses

4.6 Sustainability

The device was meticulously constructed to ensure that the vent-visor fan assembly is rugged and capable of withstanding highway speeds. Vent visors are built with these in mind. Brushless DC fans have a longer lifespan than brushed DC fans and take about 50,000 hours of use. Solar panels have a lifespan of about 20 years. The Raspberry Pi Pico has a 7–10-year lifespan.

Chapter 5

Conclusion

Summary

This chapter provides the summary and conclusion of this report and discusses future work that could be done to improve the project.

- 5.1 Summary and Conclusion**
- 5.2 Suggestions for Future Work**

5.1 Summary and Conclusion

Solar-powered Ventilation with Controlled Airflow for Parked Automobiles is a system that reduces the internal temperature of vehicles that are parked and exposed to the sun for long durations. With this system, the user will simply need to crack the windows partially to allow for airflow, flip a switch to turn the device on, and the system will run an optimized crossflow ventilation software which will activate various intake and exhaust fans to displace the heat within the vehicle by bringing in coolest air possible and exhausting hot air.

Testing results showed that the engineering requirements for the project were met. Namely, the system satisfies the top-level requirement of reducing the maximum internal temperature to under 130% of the maximum ambient temperature. The optimized crossflow ventilation system also cools the car better than window cracking, sun shield, and non-optimized crossflow ventilation. In addition, the system does not require a solar panel since it can run at full power for 10.5 hours without the solar panel; 2.5 hours longer than the proposed requirement of 8 hours with the solar panel. The structural integrity of the system was not compromised when driving at highway speeds. The device runs automatically unless switched off by the user and it does this based on window position and battery voltage.

5.2 Suggestions for Future Work

Though the project was successful in satisfying the requirements put forth by the group, it was intended as a proof of concept rather than a market-ready solution. Therefore, a lot of work remains to be done if a similar system should be brought to the consumer market.

Further research should be conducted on optimized crossflow devices that change the direction of airflow based on temperature differences throughout the vehicle. Due to time constraints, the group was unable to conduct tests aiming to find the optimal sensitivity of cooling mode switching thresholds. Differences due to switching frequency also remained untested. The group also did not have time to test the temperature sensor position. For all tests, the temperature sensors were located under the protection of the vent visor. The sensors may be even more responsive when placed in direct sunlight, which could lead to improvements in airflow direction switching.

The intended audience of the project proposal was electric vehicle manufacturers who are installing solar roofs in their models and are looking to maximize the potential of the solar roof. During testing, however, it was found that the system draws so little power that the solar panel is rendered completely unnecessary. A future revision of this project could run solely on the battery of the car and omit the solar panel. This would cut down significantly on the cost of the project and make it far easier to install after-market.

The form factor of the project could be made significantly more elegant. For one, the control module could be made much smaller by using printed circuit boards. Swapping through-hole components for surface mount components would reduce the control module footprint drastically. For this project, the PWM charge controller was placed in the control box, but a future revision without the solar panel would not require the charge controller. Thus, the size of the control box could be reduced even further. The control module was already able to fit under a car seat, but an updated version could easily be more discrete, potentially fitting in the pocket of a back seat or the center console.

The cabling footprint could also be minimized by using custom crimped ribbon cables or some kind of multi-line cable snakes such that only one cable runs from each window to the control module. Fitting connectors to the control box would make installation and removal far easier.

The vent visor assembly could be further improved by designing vent visors meant specifically for this purpose. Such a vent visor might include a channel for wiring so that wires aren't visible. An internal mounting strip for the fans could be designed using a less rigid material so that cracking does not occur upon installation. The fans and vent visors could even be integrated into one solid design, which would eliminate the need for any mounting hardware. Treating the vent visors and fans as a single unit also opens many doors for increasing airflow by manipulating the shapes of the intake and exhaust channels.

In this project, a large quantity of DC Blowers is used to intake and exhaust air through the vehicle, but only half of them will be in use at any given time. This is because the fans chosen for the project are not reversible by polarity, so each vent-visor fan assembly must include an intake array and an exhaust array. The number of fans

required could be effectively halved if reversible-polarity DC fans were used, but existing models were too costly to consider for our project. Manufacturers of this product may choose to custom design fans that are meant to reverse airflow direction when their polarity is reversed. This may also make the design more visually elegant and discreet, as fewer fans will be required. Alternatively, the system could be designed such that it utilizes the car's built-in ventilation system.

Another version of this product that would be easy for consumers to install would include a battery and solar panel built into each vent visor and communicates with the control module wirelessly. The fan power consumption data provided in section 2.5 of this report could be used to determine the size requirements of these batteries and solar panels. Further testing could then be conducted to determine the optimal battery and solar panel combination that minimizes size and cost and continues to run for eight hours. Wireless communication could be cheaply and elegantly achieved with Bluetooth.

Another alternative might use a solar panel installed in a sun shield and a lithium-ion battery to avoid wiring from the car battery. Further testing would need to be conducted to determine the heat rejection capabilities of a solar panel sun shield.

One survey response raised a valid concern regarding individuals of certain income brackets being priced out should the device exclusively be installed into newer cars, as affording a new vehicle is not always an option, and having an after-market version available would help alleviate this issue.

System B sounds like a good idea until you think about how poor people will be priced out. People who cannot afford to fix their vehicle's air conditioning would benefit the most from System A, if it was affordable enough....

Figure 5.1 – Survey Response for Proposed System Modification

Relating this to our ethical considerations, it would be advisable to take steps where possible to reduce the cost of the system to make this proposed system available to as wide a group as possible to not socially exclude those of various income brackets given that the system appears to be desired by an initial survey group.

References

- [1] J. Null, “Vehicle Heating Dynamics,” *Department of Meteorology and Climate Science, San Jose State University*, 2018. https://www.noheatstroke.org/vehicle_heating.htm (accessed Oct. 27, 2022).
- [2] J. Rugh and R. Farrington, “Vehicle Ancillary Load Reduction Project Close-Out Report: An Overview of the Task and a Compilation of the Research Results,” 2008. [Online]. Available: <http://www.osti.gov/bridge>
- [3] “Vehicle air conditioning.” <https://www.nrcan.gc.ca/energy-efficiency/transportation-alternative-fuels/personal-vehicles/choosing-right-vehicle/tips-buying-fuel-efficient-vehicle/factors-affect-fuel-efficiency/vehicle-air-conditioning/21030> (accessed Oct. 29, 2022).
- [4] S. Dhec, “Idling: Why It’s a Problem and What You Can Do”, Accessed: Apr. 08, 2023. [Online]. Available: <http://www2.epa.gov/cleandiesel/clean-school-bus>
- [5] S. Chen, B. Du, Q. Li, and D. Xue, “The influence of different orientations and ventilation cases on temperature distribution of the car cabin in the hot soak,” *Case Studies in Thermal Engineering*, vol. 39, p. 102401, Nov. 2022, doi: 10.1016/j.csite.2022.102401.
- [6] J. P. Rugh, T. J. Hendricks, and K. Koram, “Effect of Solar Reflective Glazing on Ford Explorer Climate Control, Fuel Economy, and Emissions,” 2001.
- [7] I. Urieli, “Specific Heat Capacities of Air - (Updated 7/26/08).” https://www.ohio.edu/mechanical/thermo/property_tables/air/air_Cp_Cv.html (accessed Apr. 08, 2023).
- [8] A. Szczurek and M. Maciejewska, “Categorisation for air quality assessment in car cabin,” *Transp Res D Transp Environ*, vol. 48, pp. 161–170, Oct. 2016, doi: 10.1016/j.trd.2016.08.015.
- [9] F. Norin and D. P. Wyon, “Driver Vigilance - The Effects of Compartment Temperature,” *SAE Technical Papers*, Feb. 1992, doi: 10.4271/920168.
- [10] D. Bradley, “Trying to keep your car cool? Cracking windows won’t always help,” *Scripps Media, Inc*, Jul. 06, 2022. <https://www.kshb.com/news/local-news/trying-to-keep-your-car-cool-cracking-windows-dont-always-help> (accessed Apr. 13, 2023).
- [11] “MACHSWON Solar Powered Car Fan Auto Air Vent Radiator ABS Solar Powered Energy Saving Exhaust Fan 2W Portable Air Purifiers.” <https://www.amazon.com/Powered-Radiator-Exhaust-Portable-Purifiers/dp/B092PLF1D6> (accessed Apr. 08, 2023).
- [12] “Amazon.com: HONUTIGE Solar Powered Car Ventilator, Solar Powered Car Exhaust Fan, Car Radiator, Eliminate The Peculiar Smell Inside The Car and Can

- Be Used for General Types of Cars (Black): Electronics.” https://www.amazon.com/SELLMORE-Ventilator-Radiator-Eliminate-Peculiar/dp/B07QHW849X/ref=sr_1_5?crid=1CXCHOBV7F4SJ&keywords=car+window+ventilation&qid=1667344027&qu=eyJxc2MiOiI1LjA3IiwicXNhIjojoiNC4wOCIsInFzcCI6IjIuMTYifQ%3D%3D&srefix=car+window+ventilation%2Caps%2C103&sr=8-5 (accessed Oct. 31, 2022).
- [13] “ACOOOL: Car Ventilator for Parked Cars.” <https://www.gadgetany.com/acool-car-ventilator-for-parked-cars/> (accessed Oct. 31, 2022).
- [14] H. Holder, “Cross Ventilation: Best Strategies and Benefits,” *Architropics*. <https://architropics.com/cross-ventilation/> (accessed Oct. 31, 2022).
- [15] B. Russ, “Mazda 929 (1993),” *The Auto Channel*. <https://www.theautochannel.com/vehicles/new/reviews/wk9335.html> (accessed Apr. 08, 2023).
- [16] B. Saur, “Cohort Capsule: 1992 Mazda 929 – The Ur-Millenia | Curbside Classic,” *Curbside Classic*, Oct. 09, 2014. <https://www.curbsideclassic.com/blog/cc-capsule/cohort-capsule-1992-mazda-929-the-ur-millenia/> (accessed Apr. 08, 2023).
- [17] A. Kwanten, “Mazda’s forgotten second-gen 929 was a luxury moonshot - Hagerty Media,” *Hagerty*, May 27, 2022. <https://www.hagerty.com/media/car-profiles/mazdas-forgotten-second-gen-929-was-a-luxury-moonshot/> (accessed Apr. 08, 2023).
- [18] “New Genuine Kia Soul Sport Visor Rain Guards Kit 4-Piece Set OE U82202K000 | eBay.” <https://www.ebay.com/itm/293065464095> (accessed Apr. 08, 2023).
- [19] “RBL5015B RB5015 Series DC Blower from Pelonis Technologies, Inc.” <https://www.thomasnet.com/catalogs/item/1278185-1332-3001180-23583/pelonis-technologies-inc/rb5015-series-dc-blower/> (accessed Apr. 08, 2023).
- [20] “Raspberry Pi Documentation - Raspberry Pi Pico and Pico W,” *Raspberry Pi*. <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html> (accessed Apr. 08, 2023).
- [21] “Adafruit PCF8575 I2C 16 GPIO Expander Breakout [STEMMA QT / Qwiic] : ID 5611 : \$5.95 : Adafruit Industries, Unique & fun DIY electronics and kits.” <https://www.adafruit.com/product/5611> (accessed Apr. 08, 2023).
- [22] “Amazon.com: DaFuRui 5Pack 100K ohm NTC 3950 Thermistors/Temp Sensor for Reprap Creality Ender 3, Ender 3 Pro, CR-10 10S 3D Printer(1m/3.3Ft) : Industrial & Scientific.” https://www.amazon.com/gp/product/B09TRM6Q9B/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1 (accessed Apr. 08, 2023).
- [23] “PC817X2CSZ9F SHARP - Optocoupler | THT; Ch: 1; OUT: transistor; Uinsul: 5kV; Uce: 80V; DIP4 | TME - Electronic components.”

- <https://www.tme.com/us/en-us/details/pc817x2csz9f/optocouplers-analog-output/sharp/> (accessed Apr. 08, 2023).
- [24] “Ultrasonic Ranging Module HC-SR04,” *ELEC Freaks*, Accessed: Apr. 08, 2023. [Online]. Available: www.Electfreaks.com
- [25] Billily, “1N4007 Diode: Pinout, Equivalent, Applications [FAQ].” <https://www.apogeeweb.net/circuitry/1n4007-pinout-equivalent-applications.html> (accessed Apr. 08, 2023).
- [26] “TBP01P1-508-11BE CUI Devices | Connectors, Interconnects | DigiKey.” <https://www.digikey.com/en/products/detail/cui-devices/TBP01P1-508-11BE/10238376?s=N4IgTCBcDaIwAYwFoBsBmAnJ5A5AiiALoC%2BQA> (accessed Apr. 08, 2023).
- [27] “ALITOVE 5 Pairs 4 Core Electrical LED Connector IP65 with 20cm 18AWG 4X 0.5mm² Cable for Car Truck Boat Indoor/Outdoor LED Strip Lights/String.” https://www.amazon.com/dp/B07Q4PVLTL?psc=1&ref=ppx_yo2ov_dt_b_product_details (accessed Apr. 08, 2023).
- [28] “Amazon.com: Flemoon Large Outdoor Electrical Box (12.5 x 8.5 x 5 inch), IP54 Waterproof Outdoor Extension Cord Cover Weatherproof, Protect Outlet, Plug, Socket, Timer, Power Strip, Holiday Light Decoration, Black : Tools & Home Improvement.” https://www.amazon.com/gp/product/B09NLW5HMX/ref=ppx_yo_dt_b_searchasin_title?ie=UTF8&th=1 (accessed Apr. 08, 2023).
- [29] K. Rembor, “Data Logger | Getting Started with Raspberry Pi Pico and CircuitPython | Adafruit Learning System,” *Adafruit*. <https://learn.adafruit.com/getting-started-with-raspberry-pi-pico-circuitpython/data-logger> (accessed Mar. 25, 2023).
- [30] “Amazon.com : Alrska 100 Watt Solar Panel 12 Volt High-Efficiency Shingle-Tech Monocrystalline Module PV Charger for RV Battery Boat Caravan and Other Off-Grid Applications : Patio, Lawn & Garden.” https://www.amazon.com/gp/product/B09X4JRK78/ref=ppx_yo_dt_b_searchasin_title?ie=UTF8&th=1 (accessed Apr. 08, 2023).
- [31] “Super Start Premium Battery Group Size 121R 121RPRMJ | O’Reilly Auto P.” <https://www.oreillyauto.com/detail/c/premium/super-start-premium-battery-group-size-121r/ssbo/121rprmj/v/a/142939/automotive-car-2017-kia-soul-ev> (accessed Apr. 08, 2023).
- [32] “10 Gauge Wire Tinned Copper Tray Cable - Connect Charge Controller and Battery for Solar Panel MPPT RV Automotive Marine Boat (10FT 10AWG).” https://www.amazon.com/gp/product/B09NK9VVQ4/ref=ppx_yo_dt_b_searchasin_title?ie=UTF8&psc=1 (accessed Apr. 08, 2023).
- [33] “[Upgraded] 30A Solar Charge Controller, Black Solar Panel Battery Intelligent Regulator with Dual USB Port 12V/24V PWM Auto Parameter Adjustable LCD Display (30a).”

- https://www.amazon.com/gp/product/B08L8TBCK6/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&th=1 (accessed Apr. 08, 2023).
- [34] “Amazon.com: Auto Boost Buck Converter, 5A(Max 10A) DC 5V-30V to 1.25-30V Voltage Regulator Constant Voltage Constant Current CV CC Auto Step-Up/Down Boost Converter Solar Charging Power Supply Module 150W : Electronics.”
https://www.amazon.com/gp/product/B08HS76XG1/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1 (accessed Apr. 08, 2023).
- [35] “Comidox 1.5V 1.8V 2.5V 3V 3.3V 3.7V 4.2V to 5V DC-DC Step Up Power Module Voltage Boost Converter Board 0.9-5V to 5V 5PCS.”
https://www.amazon.com/gp/product/B07L76KLRV/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1 (accessed Apr. 08, 2023).
- [36] “What Is A Thermistor And How Does It Work?,” *Omega Engineering, Inc.*
<https://www.omega.com/en-us/resources/thermistor> (accessed Jan. 15, 2023).
- [37] “Specifications for NTC Thermistor.”
- [38] “Thermistor,” *Wikipedia.*
https://en.wikipedia.org/wiki/Thermistor#B_or_%CE%B2_parameter_equation (accessed Jan. 15, 2023).
- [39] C. Nelson, “ADC | CircuitPython Libraries on any Computer with Raspberry Pi Pico | Adafruit Learning System,” *Adafruit.*
<https://learn.adafruit.com/circuitpython-libraries-on-any-computer-with-raspberry-pi-pico/adc> (accessed Jan. 16, 2023).
- [40] “Heat Load Formula - Meaning, Calculation, Solved Examples and FAQs,” Apr. 07, 2023. <https://www.vedantu.com/formula/heat-load-formula> (accessed Apr. 08, 2023).
- [41] “Design of Ventilation Systems,” *Engineering ToolBox*, 2003.
https://www.engineeringtoolbox.com/design-ventilation-systems-d_121.html (accessed Apr. 08, 2023).
- [42] “Weather in June 2022 in Orlando, Florida, USA.”
<https://www.timeanddate.com/weather/usa/orlando/historic?month=6&year=2022> (accessed Apr. 08, 2023).
- [43] O. Comstock, “U.S. Energy Information Administration - EIA - Independent Statistics and Analysis.” <https://www.eia.gov/todayinenergy/detail.php?id=18871> (accessed Apr. 08, 2023).
- [44] “IR Photodiodes | Engineering 360,” *GlobalSpec.*
https://www.globalspec.com/ds/1031/areaspec/spec_enhanced_ir (accessed Jan. 17, 2023).

- [45] Sharath, “IR Proximity Sensor with Arduino – FactoryForward,” *Factory Forward*. <https://www.factoryforward.com/ir-proximity-sensor-arduino/> (accessed Jan. 17, 2023).
- [46] “Ultrasonic Ranging Module HC-SR04,” *ELEC Freaks*, Accessed: Mar. 13, 2023. [Online]. Available: www.ElecFreaks.com
- [47] C. Gilson, “The lack of rainfall is adding up for Central Florida,” *Spectrum News 13*. <https://www.mynews13.com/fl/orlando/weather/2023/03/31/orlando-weather-blog-template> (accessed Apr. 08, 2023).
- [48] J. Rugh and R. Farrington, “Vehicle Ancillary Load Reduction Project Close-Out Report: An Overview of the Task and a Compilation of the Research Results,” 2008. [Online]. Available: <http://www.osti.gov/bridge>
- [49] “Carbon Footprint of Solar Panel Manufacturing | Cool Effect.” <https://www.cooleffect.org/solar-carbon-footprint> (accessed Nov. 14, 2022).
- [50] “Is Acrylic Bad for the Environment?” <https://www.ourendangeredworld.com/eco/is-acrylic-bad/> (accessed Nov. 14, 2022).
- [51] “Are Fans Recyclable? – Temperature Master.” <https://temperaturemaster.com/are-fans-recyclable/> (accessed Nov. 14, 2022).
- [52] “IEEE - IEEE Code of Ethics,” *IEEE*. <https://www.ieee.org/about/corporate/governance/p7-8.html> (accessed Apr. 08, 2023).

Appendix A

Relevant Datasheet Information

Summary

In this appendix we present the Relevant Data for the NTC3950 100k Ω Thermistor, HC-SR04 Sound Sensor, Boost Converter, and DC Blowers data from which is used earlier in the report for meeting engineering requirements.

- A1. Electrical Characteristics for NTC3950 Thermistor**
- A2. Resistance Table and Graph for NTC3950 Thermistor**
- A3. Specifications for RL5015B Centrifugal DC Blower**
- A4. Electrical Characteristics for Boost Converter 2108A**
- A5. HC-SR04 Electrical Characteristics and Timing Diagram**

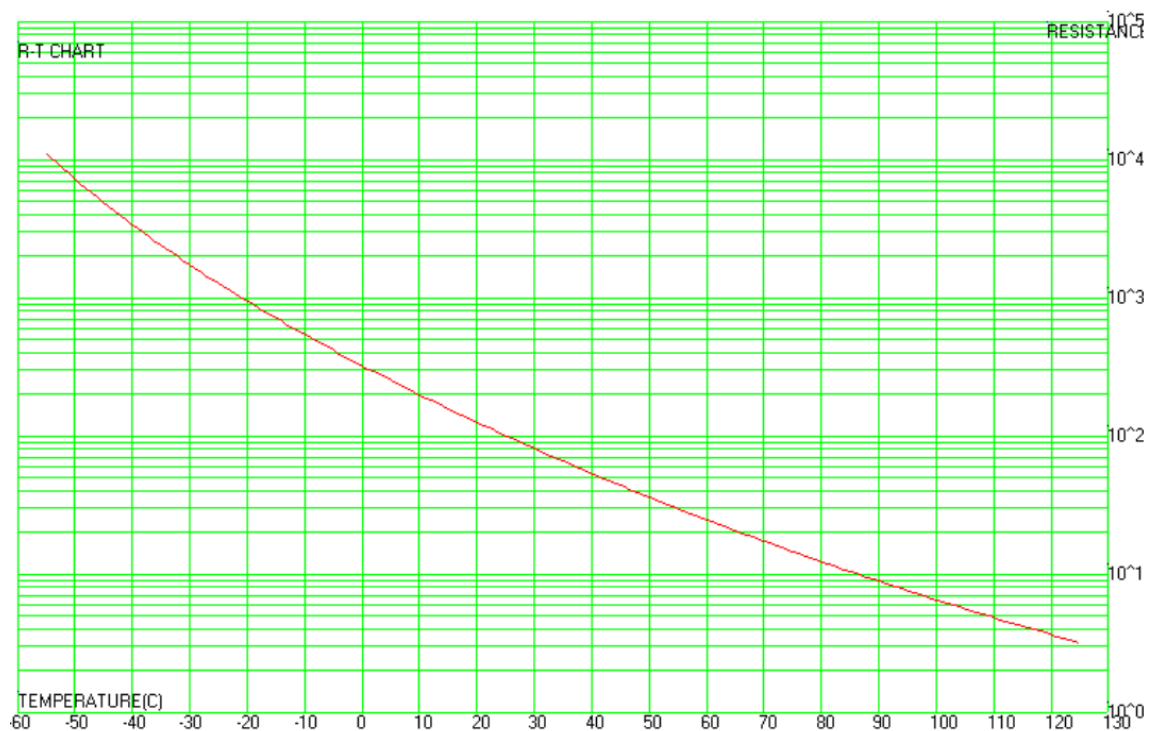
A1. Electrical Characteristics for NTC3950 Thermistor

Thermistor Electrical Characteristics [37]

	Item	Symbol	Test conditions	Unit	Specification
4.1	Zero Power Resistance at 25°C	R_{25}	$T_a=25\pm 0.05^\circ\text{C}$ Test Power $\leq 0.1\text{mW}$ Test in fluid liquid	$\text{K}\Omega$	$100\pm 1\%$
4.2	B-value	$B_{25/50}$	$B=[(T_a \times T_b)/(T_b - T_a)] \times \ln(R_a/R_b)$ $T_b=50^\circ\text{C} \pm 0.1^\circ\text{C}$	K	$3950\pm 1\%$
4.3	Thermal dissipation Coefficient	δ	In still air	$\text{mW}/^\circ\text{C}$	≥ 2
4.4	Thermal time constant	τ	In still air	sec	≤ 7
4.5	Insulation resistance	/	100V/DC 1min	$\text{M}\Omega$	≥ 100
4.6	Operating temperature	/	/	$^\circ\text{C}$	-55 ~ 125
4.7	R&T-table	/	/	/	See attached table
4.8	Resistance tolerance	/	/	/	See attached curve

A2. Resistance Table and Graph for NTC3950 Thermistor

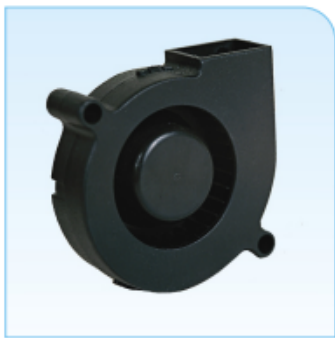
Resistance vs Temperature Graph for NTC3950 [37]



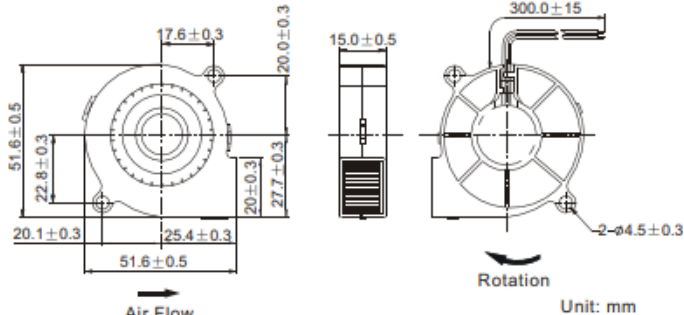
Resistance and Tolerance Chart for NTC3950 [37]

A3. Specifications for RL5015B Centrifugal DC Blower

51x51x15mm
B Series

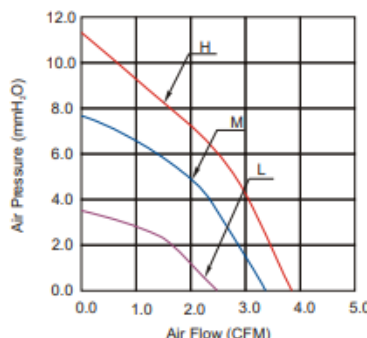


■ Dimensions Drawing




Unit: mm

■ Performance Curves



General Specification:

- Frame and Impeller : Thermal Plastic , UL 94V-0
- Lead Wires : UL Type
(+) : Red (-) : Black
- Operation Temperature : -10℃~70℃ ,
35%~85%RH
- Storage Temperature : -40℃~80℃ ,
35%~85%RH



Listed Model	Bearing System	Rated Voltage	Operation Voltage	Rated Current	Rated Speed	Air Flow	Air Pressure	Noise Level	Available Features (Optional)				Weight g
		VDC	VDC	A	RPM	CFM	mmH ₂ O	dBA	Tachometer Output	Rotation Detector	Thermal Control	Pulse Width Modulation	
RBH5015S5	Sleeve	5	3.5~5.75	0.33	4,500	3.8	11.3	37	•	•			26
RBM5015S5			3.5~5.75	0.22	4,000	3.4	7.4	35	•	•			
RBL5015S5			3.5~5.75	0.13	3,000	2.4	3.5	26	•	•			
RBH5015B5	Ball		3.5~5.75	0.33	4,500	3.8	11.3	37	•	•			
RBM5015B5			3.5~5.75	0.22	4,000	3.4	7.4	35	•	•			
RBL5015B5			3.5~5.75	0.13	3,000	2.4	3.5	26	•	•			
RBH5015S	Sleeve	12	6.0~13.8	0.14	5,000	4.4	11.5	41	•	•			
RBM5015S			6.0~13.8	0.09	4,000	3.4	7.4	35	•	•			
RBL5015S			8.0~13.8	0.05	3,000	2.4	3.5	26	•	•			
RBH5015B	Ball		6.0~13.8	0.14	5,000	4.4	11.5	41	•	•			
RBM5015B			6.0~13.8	0.09	4,000	3.4	7.4	35	•	•			
RBL5015B			8.0~13.8	0.05	3,000	2.4	3.5	26	•	•			
RBH5015S2	Sleeve	24	12.0~27.6	0.12	5,000	4.4	11.5	41	•	•			
RBM5015S2			12.0~27.6	0.08	4,000	3.4	7.4	35	•	•			
RBL5015S2			12.0~27.6	0.05	3,500	2.9	7.3	30	•	•			
RBH5015B2	Ball		12.0~27.6	0.12	5,000	4.4	11.5	41	•	•			
RBM5015B2			12.0~27.6	0.08	4,000	3.4	7.4	35	•	•			
RBL5015B2			12.0~27.6	0.05	3,500	2.9	7.3	30	•	•			

Specification Chart for RL5015B DC Centrifugal Blower [19]

A4. Electrical Characteristics for Boost Converter ME2108A

Electrical Characteristics for ME2108A

Electrical Characteristics:

Measuring conditions: Unless otherwise specified, $V_{IN}=V_{out}*0.6$, $V_{SS}=0V$, $I_{OUT}=10mA$, $T_{opt}=25^{\circ}C$.

ME2108Axx/Cxx $F_{osc}=180kHz$

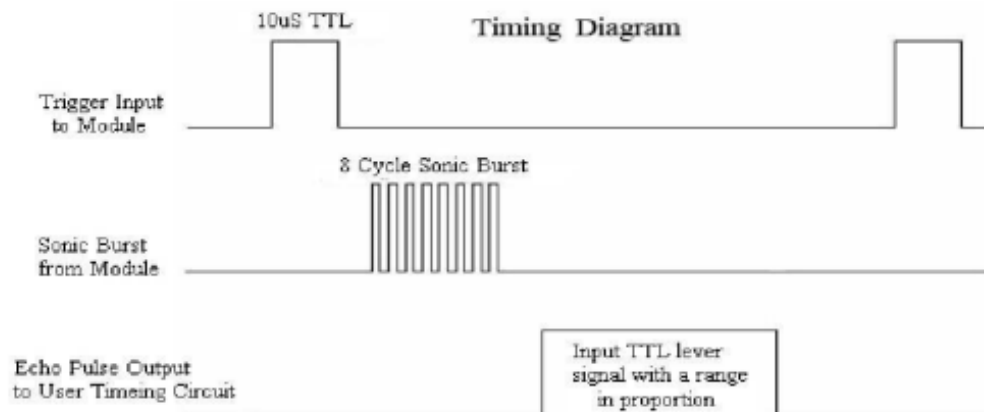
SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
V_{OUT}	Output Voltage		$V_{out}*0.975$	V_{out}	$V_{out}*1.025$	V
V_{start}	Oscillation Start-up Voltage	$I_{OUT}=1mA$, $V_{IN}: 0 \rightarrow 2V$		0.8	0.9	V
V_{hold}	Oscillation Hold Voltage	$I_{OUT}=1mA$, $V_{IN}: 2 \rightarrow 0V$		0.45		V
I_{DD1}	Supply Current 1	No external component $V_{out}=V_{out}*0.95$,		50		μA
I_{DD2}	Supply Current 2	$V_{out}=V_{out}+0.5V$		9		μA
I_{LX}	Lx Switching Current	$V_{LX}=0.4V$, $V_{out}=V_{out}*0.95$		360		mA
I_{LXleak}	Lx Leakage Current	$V_{out}=V_{LX}=6V$			0.5	μA
F_{osc}	Oscillation Frequency	$V_{out}=set$ $V_{out}*0.95$		180		kHz
Maxdty	Duty Ratio	on(V_{LX} "L")side		84		%
EFFI	Efficiency			85		%

A5. HC-SR04 Electrical Characteristics and Timing Diagram

Electric Parameter

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: $\mu\text{S} / 58 = \text{centimeters}$ or $\mu\text{S} / 148 = \text{inch}$; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Appendix B

Project Software

Summary

In this appendix we present the main code used on the Raspberry Pi Pico which allows the project to function.

B1. Code


```

    if distance < 22000:
        window = "Up"
        led.value = False
    else:
        window = "Down"
        led.value = True

    return window

def proximity(): # Main loop proximity sensor function
    # Get distance to window and then determine if it is up or down
    distance = readEcho(150) # 150 reps
    window = up_down(distance)
    print("distance = {} Window = {}".format(distance, window))

    # Logging Data
    datalog = open("/data.txt", "a")
    datalog.write('{}.'.format(window))
    datalog.flush()
    datalog.close()

    return window

# \\\\\\\\\\\\\\\\\\\\\\\
# Fan Control
# //\\\\\\\\\\\\\\\\\\\\\\

# Initializing transistor array GPIO
tr = [DigitalInOut(board.GP0), DigitalInOut(board.GP1), DigitalInOut(board.GP2),
      DigitalInOut(board.GP3),
      DigitalInOut(board.GP4), DigitalInOut(board.GP5), DigitalInOut(board.GP6),
      DigitalInOut(board.GP7),
      DigitalInOut(board.GP8), DigitalInOut(board.GP9), DigitalInOut(board.GP10),
      DigitalInOut(board.GP11),
      DigitalInOut(board.GP12), DigitalInOut(board.GP13), DigitalInOut(board.GP14),
      DigitalInOut(board.GP15),
      pcf.get_pin(0), pcf.get_pin(1), pcf.get_pin(2), pcf.get_pin(3),
      pcf.get_pin(4), pcf.get_pin(5), pcf.get_pin(6), pcf.get_pin(7)]

for i in range(len(tr)): # Set all tr GPIO directions to output
    if i <= 15:
        tr[i].direction = Direction.OUTPUT
    else:
        tr[i].switch_to_output(value=False) # Sets pins to output with low level default

# Fan array intake / exhaust groupings
in1 = tr[0:3] # Array 1, Intake
ex1 = tr[3:6] # Array 1, Exhaust
in2 = tr[6:9] # Array 2, Intake
ex2 = tr[9:12] # Array 2, Exhaust
in3 = tr[12:15] # Array 3, Intake
ex3 = tr[15:18] # Array 3, Exhaust
in4 = tr[18:21] # Array 4, Intake
ex4 = tr[21:24] # Array 4, Exhaust

# Battery mode groupings
# If bat_mode = 1, turn these fans off:
b1 = [tr[1], tr[4], tr[7], tr[10], tr[13], tr[16], tr[19], tr[22],
      tr[2], tr[5], tr[8], tr[11], tr[14], tr[17], tr[20], tr[23]]

# If bat_mode = 2, turn these fans off:
b2 = b1[8:16]

prev_mode = 1
prev_t = 0

def cooling_mode(t): # Function uses temperature readings from the four windows to
    determine cooling mode

```

```

global prev_mode
global prev_t
t1,t2,t3,t4 = t[0], t[1], t[2], t[3]
# Car side definitions
driver = t1+t2
passenger = t3+t4
front = t1+t3
back = t2+t4

new_t = stdev(t)

# Mode selection
if abs(new_t - prev_t) > 3: # if temperature differences between windows are at great
    enough compared to previous temps, choose a mode.
    if abs(front-back) <= abs(driver-passenger):
        # if the temp difference between driver and the passenger is at least that of front
        and back, select mode 1 or 2
        if driver < passenger: # Airflow: Driver --> Passenger
            mode = 1
        elif driver > passenger: # Passenger --> Driver
            mode = 2
    else: # select mode 3 or 4
        if front < back: # Front --> Back
            mode = 3
        elif front > back: # Back --> Front
            mode = 4
    prev_mode = mode # Set previous mode to new mode
    prev_t = new_t # Set the previous temp deviation to the new temp deviation
else: # adopt the previous mode
    mode = prev_mode

# Logging Data
datalog = open("/data.txt", "a")
datalog.write('{}\n'.format(mode))
datalog.flush()
datalog.close()

#print("Cooling Mode: {} | new_t = {:.2f} | prev_t = {:.2f}".format(mode,new_t,
prev_t))
return mode

def fan_control(window, mode, bat_mode): # Takes window position, cooling mode, and battery
mode
# Activates and deactivates fans accordingly

# Window Control
if window == "Up" or bat_mode == 0: # All fans should be OFF
    for fans in tr: # for all fans connected to transistor array:
        fans.value = False # turn each fan off

else: # System is active
    # Cooling Mode Control
    # Define lists of fans to switch on and off
    if mode == 1:
        on = [in1, in2, ex3, ex4] # Intake | Driver | front/back
        off = [ex1, ex2, in3, in4] # Exhaust | Passenger | front/back

    elif mode == 2:
        on = [ex1, ex2, in3, in4] # Intake | Passenger | front/back
        off = [in1, in2, ex3, ex4] # Exhaust | Driver | front/back

    elif mode == 3:
        on = [in1, ex2, in3, ex4] # Intake | Front | driver/passenger
        off = [ex1, in2, ex3, in4] # Intake | Back | driver/passenger

    elif mode == 4:
        on = [ex1, in2, ex3, in4] # Intake | Back | driver/passenger
        off = [in1, ex2, in3, ex4] # Exhaust | Front | driver/passenger

```

```

for fans in chain(*on):
    fans.value = True # turn all fans in on list ON
for fans in chain(*off):
    fans.value = False # turn all fans in off list OFF

# Switch number of active fans based on battery mode
if bat_mode == 1:
    for fans in b1:
        fans.value = False
if bat_mode == 2:
    for fans in b2:
        fans.value = False

# \\\\\\\\\\\\\\\\\\\\\\\
# Main Loop
# \\\\\\\\\\\\\\\\\\\\\\\

try:
    while True:
        # Read Sensors
        bat_mode = get_battery() # Get battery mode
        #bat_mode = 3 # Uncomment to force battery mode 3
        window = proximity() # Get window position from all four windows
        T = temperature() # Get temperatures from all four windows

        # Control Fans
        cool_mode = cooling_mode(T) # Determine cooling mode
        #cool_mode = 2 # Uncomment to force cooling mode 2
        fan_control(window, cool_mode, bat_mode) # Switch fans based on input arguments

        pi_led.value = not pi_led.value # Toggle onboard LED
        time.sleep(598.10) # 10 mins minus all other sleeps
        #time.sleep(5) # Short sleep

except OSError as e: # Typically when the filesystem isn't writeable...
    delay = 0.5 # ...blink the LED every half second.
    if e.args[0] == 28: # If the filesystem is full...
        delay = 0.25 # ...blink the LED faster!
    while True:
        pi_led.value = not pi_led.value
        time.sleep(delay)

```

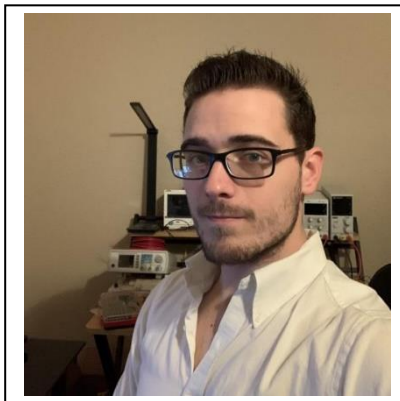
About Us

Ian Matheson



- Degrees:
 - A.S. in Sound & Music Production
 - A.S. in Sound & Music Technology – Audio Engineering Specialization
- Technical Certificates:
 - Audiovisual Production
 - DANTE Certification
 - Basic Electronics Technician
 - Advanced Electronics Technician
- Currently pursuing B.S. in Electrical & Computer Engineering Technology – Electronics Specialization (with a program 3.90 GPA, and overall 3.80 GPA)
- Currently working as a Lab Assistant in the Sound and Music Technology Department

Daniel Eisenbraun



- Degrees:
 - A.S. in Electronics Engineering Technology – Electronics Specialization
- Technical Certificates:
 - Basic Electronics Technician
 - Advanced Electronics Technician
- Currently pursuing B.S. in Electrical & Computer Engineering Technology – Electronics Specialization (with a program 3.32 GPA, and overall 3.15 GPA)
- Currently working as a Laser Technician at Northrop Grumman Mission Systems, experience with the E2LRF and TBEAR systems.