Final Report for ECE 445, Senior Design, Spring 2024

TA: Guozheng He

31 May 2024

Project No. 38

Mingjun Wei

Shaohua Sun

Ye Yang

Yinjie Ruan

By

Vehicular eDge computing system

**Abstract**

Our project, Vehicular Edge Computing System, aims to deploy the edge computing server on-board a vehicle for Vehicular Edge Computing (VEC). An on-board edge server is expected to provide data communication with less delay and the wind is utilized to cool down the edge server to reduce energy consumption. To achieve these, firstly we’ve designed a server shelter to fix it on the roof of the vehicle and a cooling system inside the shelter for wind cooling. Secondly, we verify the effectiveness of the design by both simulating the heat dynamics in software and conducting physical experiments. Thirdly, we’ve also designed the functionality of CPU frequency adjustment with respect to the wind speed and CPU temperature realized by two methods: one is based on PID control, the other is based on reinforcement learning algorithm. Finally, we model the application of the communication between the edge server and the user end about simple path planning and image recognition.

**Contents**

[1. Introduction 1](#_Toc168097514)

[1.1 Service Quality 1](#_Toc168097515)

[1.2 Energy Consumption 1](#_Toc168097516)

[1.3 Water-proof Urgency 2](#_Toc168097517)

[**2. Design** 3](#_Toc168097518)

[**2.1 Block Diagram** 3](#_Toc168097519)

[2.2 Shell 4](#_Toc168097520)

[2.2.1 Structural consideration 4](#_Toc168097521)

[2.2.2 Material selection 6](#_Toc168097522)

[2.3 Server 7](#_Toc168097523)

[2.3.1 The configuration of the server 7](#_Toc168097524)

[2.3.2 The CPU frequency self-adjustment 9](#_Toc168097525)

[2.3.3 The simulation of application of edge servers 10](#_Toc168097526)

[3. Design Verification 12](#_Toc168097527)

[3.1 Shell 12](#_Toc168097528)

[3.1.1 Introduction to specifications 12](#_Toc168097529)

[3.1.2 Experiments 12](#_Toc168097530)

[3.1.3 Wind tunnel test results 16](#_Toc168097531)

[3.1.4 Spray test results 16](#_Toc168097532)

[3.1.5 Campus driving test 18](#_Toc168097533)

[3.2 CPU Temperature Self-adjustment 22](#_Toc168097534)

[3.2.1 PID control 22](#_Toc168097535)

[3.2.2 Reinforcement Learning 22](#_Toc168097536)

[4. Cost 24](#_Toc168097537)

[**4.1 Parts** 24](#_Toc168097538)

[**4.2 Labor** 24](#_Toc168097539)

[5. Conclusion 25](#_Toc168097540)

[**5.1 Accomplishments** 25](#_Toc168097541)

[**5.2 Uncertainties** 25](#_Toc168097542)

[**5.3 Ethical considerations** 25](#_Toc168097543)

[**5.4 Future work** 26](#_Toc168097544)

[References 27](#_Toc168097545)

[Appendix A 31](#_Toc168097546)

[Appendix B 32](#_Toc168097547)

[Heat Transfer calculation 32](#_Toc168097548)

[Appendix C 34](#_Toc168097549)

[The code of PID control 34](#_Toc168097550)

[The code of reinforcement learning 36](#_Toc168097551)

# 1. Introduction

1.1 Service Quality

With the development of 5G technology, the vehicular edge computing (VEC), a distributed framework that enables the service by data sources closer to the user, is greatly advanced for reducing the delay of the communication between the user ends and the Data Clouds. Its applications, Autonomous Driving or the intelligent transportation system (ITS), have received great interest from academia and industry.

For the intelligent transportation system, the growing volume of data to be processed and the goal of short response time requires more and more smaller distance between the edge server and the user ends. The traditional architecture of the edge server, deployed indoor at the station, still encounters the problem of certain communication delay. Also the recent research has compromise on a common issue, that is the relatively short period of service time that a server can provide to a vehicle due to the fixed service range. This either impedes a relatively time-consuming task to finish within the time of the vehicle inside the service range, or requires the server to transfer the task to the next server and continue with it which is less efficient. By deploying the edge server on-board the vehicle, it reduces the communication delay further and enable longer service time and more flexible service range of the server with vehicles.

1.2 Energy Consumption

As the edge computing system is normally set in an air-conditioning room, the enormous power is used to maintain air conditioners and fans [1]. The electronic devices at a data center can consume over 40% of the total energy needed. However, as vehicles are mobile, the wind can be utilized to assist cooling. Therefore, we propose the passive cooling based on the wind, where air-conditioning is not deployed, for vehicular edge computing systems. We design the shelter of the edge server for passive cooling, so that a great proportion of the energy can be saved but still with steady cooling effectiveness.

At the present stage. There are already some cooling systems that are investigated to reduce energy consumption. Sheng Du proposed that a water-cooled LHP system can be mixed with a water-cooled CRAC system to form a hybrid cooling system, which contains both computer room air conditioner and hoop heat pipe to cool the CPU down [2]. Also, F.M. Naduvilakath-Mohammed proposed a hybrid cooling system with the method of both R134a charged Vapor Compression Refrigeration (VCR) and liquid cooling to help heat dissipation [3].

Besides saving energy from deploying more effective cooling systems, some strategies on the server itself are also considered. Zhi Xiong viewed server cluster energy consumption as a mixed integer linear programming (MILP) problem. He suggested a strategy to dynamically adjust the service state and CPU frequency to reduce the energy wasted [4]. It is a common method to optimize the on/off state and CPU frequency of servers in a cluster based on programming problems [5],[6],[7],[8]. Ali Asghari proposed that for mobile edge computing (MEC) architecture and Fog computing (FC) architecture, separate server placement with cost and latency optimized can be realized [9].

Some algorithms are used for lowering energy consumption. In Xiongwei Fei’s paper, a parallel AES algorithm uses both the GPU and CPU to balance the load according to the computing power of the CPU and GPU. The method also uses Nvidia Management Library (NVML) to adjust GPU frequency, achieve data transmission and computation overlap, make full use of GPU computing resources, and reduce energy consumption as much as possible [10].

On the algorithm part, Wenjun Lin proposed a multi-agent reinforcement learning-based method, in which the fan has dynamic speed adjustments to balance the center server’s performance and power consumption [11]. Yogesh Fulpagare proposed that for a 1U air-cooled server mockup, by using the Deep Reinforcement Learning (DRL) algorithm, separate fan controlling has advantage in saving more energy. [12] Wen-Xiao Chu also did research on the DRL algorithm, his team considered parameters include bypass phenomenon, transient fan operations and heat source influence to create a model that saves 16.7% of the energy compared to 40% fan duty [13].

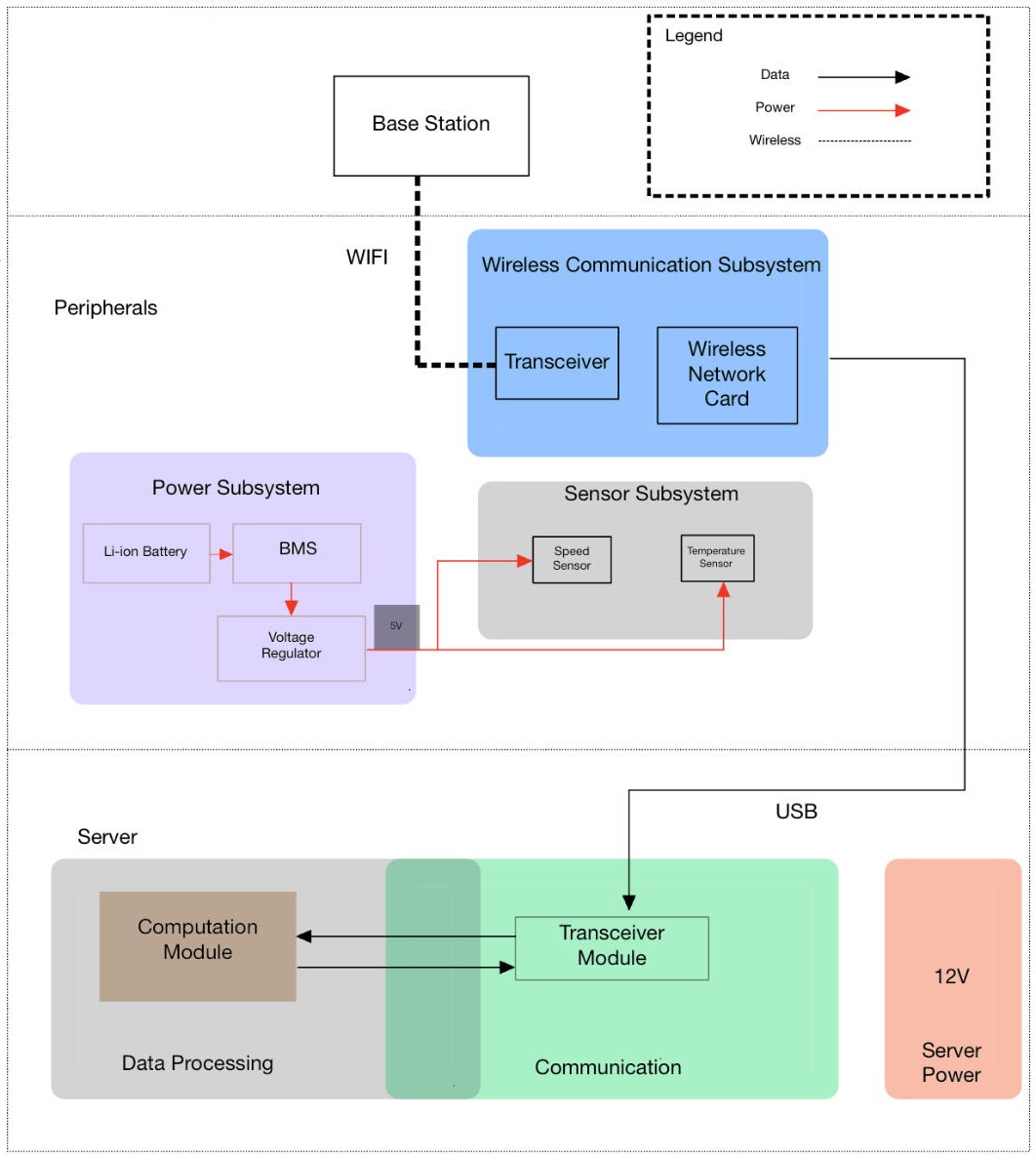
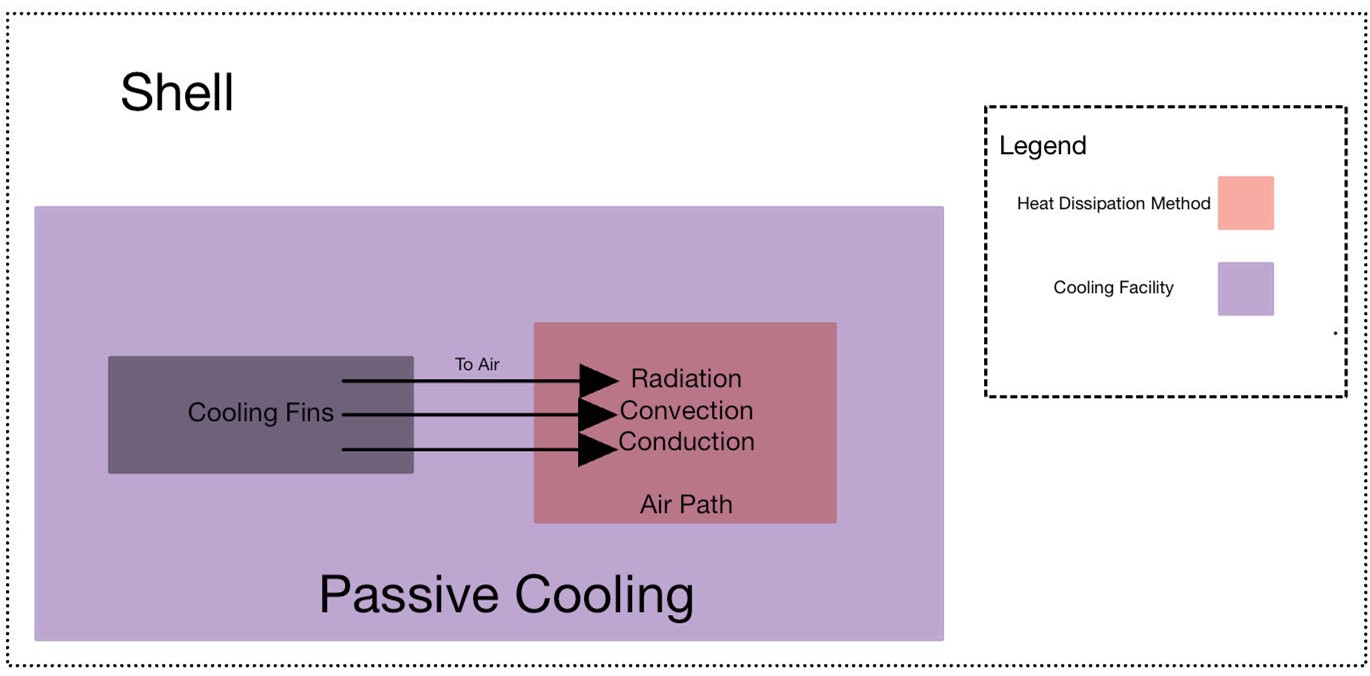
1.3 Water-proof Urgency

The environment out of mobile vehicles are very unstable. Edge computing systems on a mobile vehicle should be able to adapt well to different environments, especially on rainy days. Therefore, the server shelter should be well designed to combine the cooling capability as well as water-proof effectiveness in rainy days. However, these two aspects are basically contradictory to each other. Therefore, it is the main challenge to combine these two functions: water-proof and passive cooling.

**2. Design**

**2.1 Block Diagram**

Figure 1. Block Diagram



As is shown in the block diagram, our design about vehicular edge computing system is divided into three parts. The “shell” is the mechanical design of the server shelter; the “server” means that we will configure it and simulate its application; the “peripherals” is referred to some necessary components.

## 2.2 Shell

In the segment of mechanical design, a container shell was built to realize the functions of passive air cooling and waterproof.

The mechanical design of the container can be divided into 3 segments.

1. An aluminum-made [14] inner container to contain server.
2. A POM [15] & aluminum made bell-mouth air collector.
3. A sealed outer shell.

Each segment is designed separately and combined to reach all functionality.

### 2.2.1 Structural consideration

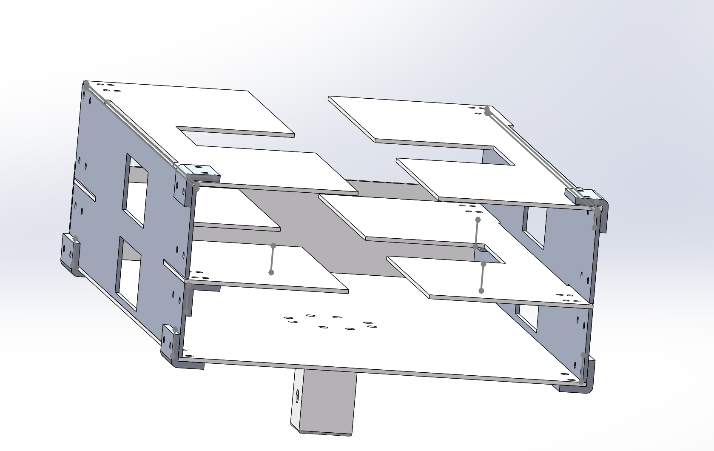
Below is a demonstration of the aluminum-made inner container.

Figure . Inner Container

The container is designed to contain a server. The voids designed on this container are used to contain heat pipes, where the fins on heat pipes stretch out from these voids to dissipate heat. A total of 6 towers of fins can be suited in the container, creating tremendous heat dissipation effects.

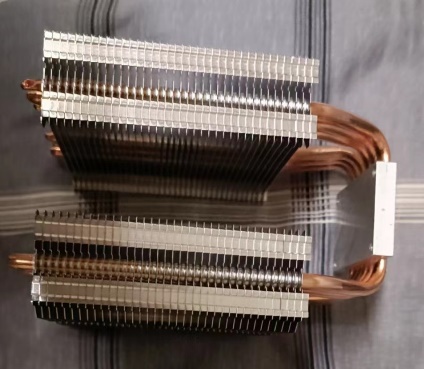


Figure . Heat Pipe with Two Towers of Fins

Below is a demonstration of POM & aluminum made bell-mouth air collector.

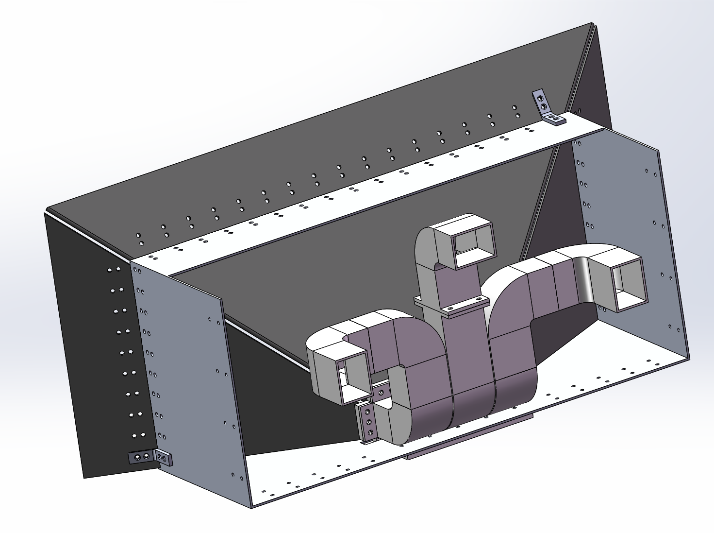


Figure 4. POM & Aluminum Made Bell-mouth Air Collector

The bell-mouth shaped boards constitute a collector of air. The inclination angle was selected to be 60◦ against an aluminum sheet metal component. This angle is a perfect fit to make space for three tunnels with a size of 50*mm* × 66*.*67*mm*. These tunnels constitute three air paths to allow passive air cooling. When vehicle operates with a speed, air accumulates from the air collector due to a funneling effect [16] and pass the air tunnel to cool down the server with convective cooling [17].

Below is a demonstration of a POM made sealed outer shell.

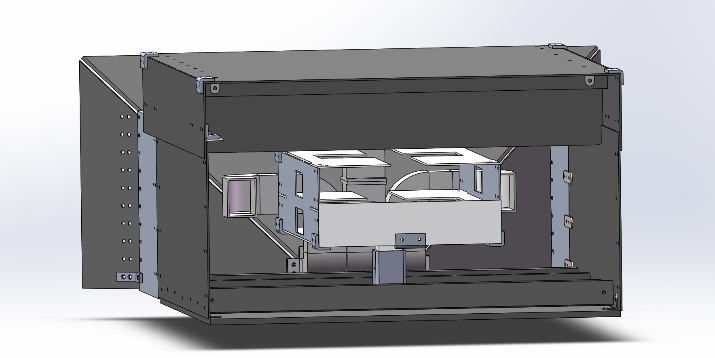


Figure . POM-Made Sealed Outer Shell

Figure 5 is also a complete demonstration of the mechanical part. The structure of the container and the outer shell is carefully designed. The outer shell is sealed from the top and the sides. Therefore, most of rains would be directly blocked by the shell. This Figure also demonstrates how exactly the wind is used to passively cool the server. The wind from the air tunnel would not directly be used to cool the server, it is used to cool the fins from the heat pipes that are attached to the server container. It is also why three tunnels are designed, to provide convective cooling to the fins of three heat pipes. The best advantage of this cooling method is that the server would not be affected by the water carried with the wind, if it rains. Another aspect that blocks the rain would be the vertical upward component of pipe that constitutes each air path (see Figure 4). It is anticipated that rain drops or even moisture can be directly blocked by these vertical lifts.

We can see that theoretically, our design has met the requirements of passive cooling and water-proof effectiveness, yet experiments are to be conducted to verify this (see chapter 5 and 6).

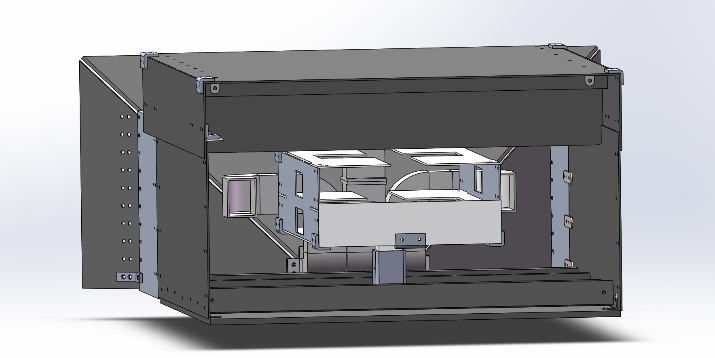
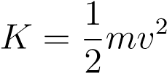


Figure 6. Complete Assembly from Another View

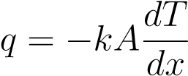
2.2.2 Material selection

The main goal of the design is to achieve a perfect balance between cooling and energy consumption. Since the design is mostly sealed, the waterproof should not be a significant element in choosing materials. From the famous kinematic energy equation [18]:

 (1)

We know that lighter material will consume less energy. Therefore, a first principle is that the overall material should be light enough.

To maximize cooling effect, considering the famous heat conduction equation [19]:

 (2)

Therefore, the heat conduction rate of the material is vital in heat dissipation. I chose Al1060 [20][21] for the material of inner shell, because it is light and has a very high heat conduction rate. For the outer shell, the selection should be more careful. In our design, the supporter for the bell-mouth shaped boards is Al6061. Although the heat conductivity of Al6061 cannot compare with that of Al1060, it is not designed for perfect heat conduction rate, rather it is designed to support both the bell-mouth shaped boards and the larger body made from POM. Al6061 was selected because it has better strength and is cheaper. Most of the outer shell should not be alloy. Considering in hot weather, the outer shell ought to be less heat conductive to create a lower inner temperature environment, while in cold days, the heat conductivity of the outer shell does not matter since server would be less likely to overheat in cold environment temperature. Therefore, the material of the majority of outer shell ought to be plastic. By careful selection, POM is used as the material of the rest of outer shell, the selection criteria are based on ease of manufacturing and strength. Because most plastic materials have similar thermal conductivity and density [22][23], while the Young’s Modulus that represent the strength between materials are poles apart [24]. From the Young’s Modulus table from the internet [24], we identified a plastic called POM, which is also easy to manufacture and light enough to withstand the weight of server [25]. Other materials that have better strength than POM is either expensive or hard to manufacture into thick boards. The only material that has better strength than POM and easy to manufacture is PEEK, however, it is way more expensive (85$/kg) than POM (around 0.7$/kg), we think its marginal value cannot place on a par with its cost) [26][27][28].

**Table 1: Heat Conduction Rate & Density of Materials**

|  |  |  |
| --- | --- | --- |
| **Materials** | **Conduction Rate(W/mK)** | **Density**  **(g/cm3)** |
| Al6061 | 167 | 2.75 |
| Al1060 | 234 | 2.7 |
| POM | 0.25-0.34 | 1.41 |
| PEEK (aborted) 0.25 | | 1.26 |

2.3 Server

For making the server with the adaptability to the condition of a moving vehicle, we firstly configurate the server. Secondly, we design the algorithm for CPU frequency self-adjustment which helps the CPU to adjust the frequency based on the environment and its temperature. Thirdly, we simulate the applications of edge servers, plan selection and photo recognition for the requests from the user end.

2.3.1 The configuration of the server

To configure the edge server, there’re five small parts of the work, including installing the Ubuntu operating system, driving a USB Wi-Fi adapter, deploying Deep Learning environment, how to collect CPU temperature data, and deploying Docker basics.

(1) “Ubuntu 22.04 LTS” of desktop version is installed.

(2) In order to make the server be able to work on vehicle normally, it needs to get wireless network access to the Internet as it’s impossible to tow a network cable on the road. So that the server can transmit or receive data from the Internet to communicate with the user ends. However, in real applications, the user ends are not limited to cellphones or laptops but a server on-board another car nearby or even infrastructures. It’s of great value in the future to enable the server compatible with different wireless communication protocols to exchange data.

We bought a USB Wi-Fi Adapter from company called “MERCURY” which is easily installed by plugging into the USB interface and deploying a corresponding driver. The advantages of applying a USB Wi-Fi adapter include but not limited to that there’s no tricky extra hardware needed to prepare, supporting beamforming, and being able to act like a hotspot.



Figure 7. The USB Wi-Fi adapter

(3) As there’s convenient Deep Learning library integrated in python called Pytorch, we deployed the Pytorch based on Anaconda, an integrated platform for packages in Python. For most of the data structures convenient for constructing a Neural Network, we downloaded the Torch, the library for efficiently operating multiple dimensional data, from its official website according to the hardware configuration.

(4) Preparing for further experiment of winding cooling, it’s needed to be ensured that we’re able to monitor the CPU temperature to verify the effectiveness of the cooling system. Psensor has user-friendly data display to monitor the CPU temperature by just plotting a curve of CPU temperature versus time. Also, it can run in the background without influencing the other processes. We ran a traditional and simple Deep Learning algorithm of Figure Recognition with “nmist” dataset. The CPU temperature was raised to the highest 45 degree of Celsius. It’s predictable that for more complex data processing, the CPU temperature will be easily higher than this.

Figure 8. The CPU temperature showed in Psensor

(5) We deploy the basic part of docker to make use of algorithm convenient. It has the advantages of managing the resources more efficiently, and it’s not needed to configure the environment for coding after deploying Docker.

2.3.2 The CPU frequency self-adjustment

In this part of the work, we design the algorithm in Python to realize CPU frequency self-adjustment based on the environmental wind speed and the CPU temperature. It has the purpose that if the cooling condition is good, the CPU can work with higher input power, for example by increasing its frequency, to improve the performance; if the cooling condition is poor, the CPU will automatically lower its input power, for example by reducing its frequency, to guarantee not overheat. The high-level goal is to utilize the CPU fully under the circumstance of saving energy by wind cooling.

In Linux system in the directory “/sys/devices/system/cpu”, there’re directories for each cpu with information about them respectively. When choosing one of them and enter the “cpufreq” directory, there is information about CPU frequency, including but not limited to the available CPU frequencies, the working modes of CPU.

Linux provides the API of adjusting the working modes of CPU with five governors. They are “performance”, “powersave”, “userspace”, “ondemand”, and “conservative”. The “performance ” governor sets the CPU frequency to the statistically highest within the allowed frequency range. The “powersave” governor on the contrary sets the CPU frequency to the statistically lowest within the allowed frequency range. The “userspace” allows the user program to set the CPU frequency to a specific allowed value in “scaling\_setspeed” file. The “ondemand” governor sets the CPU frequency based on the load, the CPU usage statistics over the last period. This has the requirement for the CPU to be sensitive to the real-time task loading and adjust the frequency quickly. The “conservative” governor also sets the CPU frequency according to the task load but by changing the frequency more gradually [29].

(1) The PID control.

The PID control, Proportional-integral-derivative control is the very common method of closed-loop control [30]. Here we adopt PID control because we want the temperature to be kept around 65 degrees of Celsius, for example, to balance between the performance and security. As smoothly adjusted, we don’t need to worry that the increasing CPU temperature caused by the increasing CPU frequency will surpass 65 degrees of Celsius too much to be over the threshold value of 80. However, we need to compel the CPU frequency to be set to the lowest immediately if the wind speed decrease abruptly to avoid overheat. That’s the disadvantage of the smooth change of PID control. The code is attached in Appendix C.

(2) The reinforcement learning algorithm.

The reinforcement learning algorithm is very suitable for the task involving with the long-time interactions with the environment [31]. The electronic device collects environmental data and makes actions based on the data. There’s rewarding function that judges to what extent the action meets the expectation. Then it adjusts the action next time, repeating the same process, until the training time is depleted. After training, the algorithm is expected to make decisions according to the environment situation. In our design, I chose Q-table as the main structure for the learning process. The code is attached in Appendix C.

2.3.3 The simulation of application of edge servers

We make some simple algorithms to work with path selection and photo recognition.

(1) For the path selection algorithm, as we all know, it requires the map and lots of calculation resources so it is better to ask server to select the proper path. When the starting point and destination are known, it first searches in the cache. If the cache contains the selected starting point and destination, it returns the saved path from the cache. Otherwise, it communicates with the server, informing the server of the starting point and destination, and requests the server to provide a path. It then returns the path given by the server and stores this pair of starting point, destination, and path in the cache. To improve smoothness, the cache is cleared every month.

The path selection algorithm consists of three different algorithms. The first algorithm utilizes the A\* algorithm, which can find the shortest path from the starting point to the destination in a graph. Both the server-side and client-side have this algorithm. The second algorithm is for the client-side, which manages the cache and retrieves paths from the cache based on the starting and destination points, or communicates with the server to obtain the path. The third algorithm is for the server-side, which executes the A\* algorithm based on the starting and destination points after communication with the client, and returns the path to the client. The communication between the client and server utilizes the Flask framework in Python, using the HTTP protocol, which allows obtaining services with knowledge of the server's IP address and corresponding port. Although the client-side also has a local path selection algorithm, obtaining the path through communication with the server can save computational resources. The path selection algorithm requires up-to-date maps and a certain amount of computational resources. In today's data-driven society, almost every pair of starting and destination points will be calculated by more than one consumer for the path. In this case, with the existence of a database on the server-side, duplicate calculations can be effectively avoided, saving computational resources. Additionally, if there is a large number of people selecting similar starting and destination points within a short period of time, the weights of the path selection algorithm can be manually adjusted to generate different paths and prevent congestion. Since the client-side path selection algorithm is the same, the generated paths are the same and do not have congestion prevention functionality.

(2)For the image recognition algorithm, when the client receives an image, it sends the image to the server and requests the server to provide the result. Upon receiving the image, the server saves it to a fixed storage location. It then uses a pre-trained deep learning model to recognize the image and transfers the recognition result to the client. The client returns the original image along with the image recognition result.

There are only two different algorithms in the image recognition algorithm. The first algorithm is the client-side algorithm, which allows the client to upload images, preprocess them, and then send them to the server for recognition. The server returns the recognition results, which are then displayed to the client. The second algorithm is the server-side algorithm, which is responsible for recognizing the images and returning the recognition results. A pre-trained CNN model from PyTorch is used for the image recognition task. During the training process, the model undergoes two rounds of convolution, activation, and pooling operations before producing the final output [32]. The trained model performs well in testing. The communication between the client and server also utilizes the Flask framework in Python. The image recognition algorithm has various applications in edge computing for autonomous driving. For example, it can be used to determine the distance between vehicles and adjust the speed to maintain a safe distance while increasing the overall speed. It can also calculate the deceleration and braking distance when encountering a red light. Additionally, image recognition can assist in emergency braking when pedestrians or obstacles suddenly appear. In the future, it is hoped that the path selection algorithm and image recognition algorithm can be integrated into a new communication algorithm, using the same port to improve port utilization efficiency.

# 3. Design Verification

3.1 Shell

3.1.1 Introduction to specifications

In the Mechanical Design part, specifically the design of container shell, two segments of specifications are desired.

* First is the ability to utilize the speed of vehicle to passive cooling.
* The second is the ability to avoid moistening the inner server.

3.1.2 Experiments

A bunch of experiments are designed to verify the specifications. To verify the effectiveness of passive cooling capacity and waterproofing, we conducted the following experiments.

* Ansys Fluent CFD simulation

After CAD modeling was completed, Ansys Fluent was used to conduct fluid analysis. During the simulation, several assumptions were made:

- The incoming air had a velocity of 10m/s with its temperature equal to ambient temperature 25°C.

- Air was treated as incompressible ideal gas with constant specific heat 1006.43J/(kg\*m), heat conductivity 0.0242W/(m\*K) and dynamic viscosity 1.7094e-5kg/(m\*s).

- Radiation was neglected in forced convection since it had orders of magnitude larger than radiation with 10m/s air velocity.

- Pseudo time step of the fluid (air) was set to 3s, and the pseudo time step of the fluid (aluminum) was set to 2000s since the air had much more sensitive transient responses during the simulation.

- The heat generation, which corresponds to the server power, was set to constant 300W.

- The was no thermal contact resistance between the heating element and the container. The container itself was the heat source.

The simulation results include the velocity vector field and the temperature distribution, shown in Figure 9 and Figure 10.

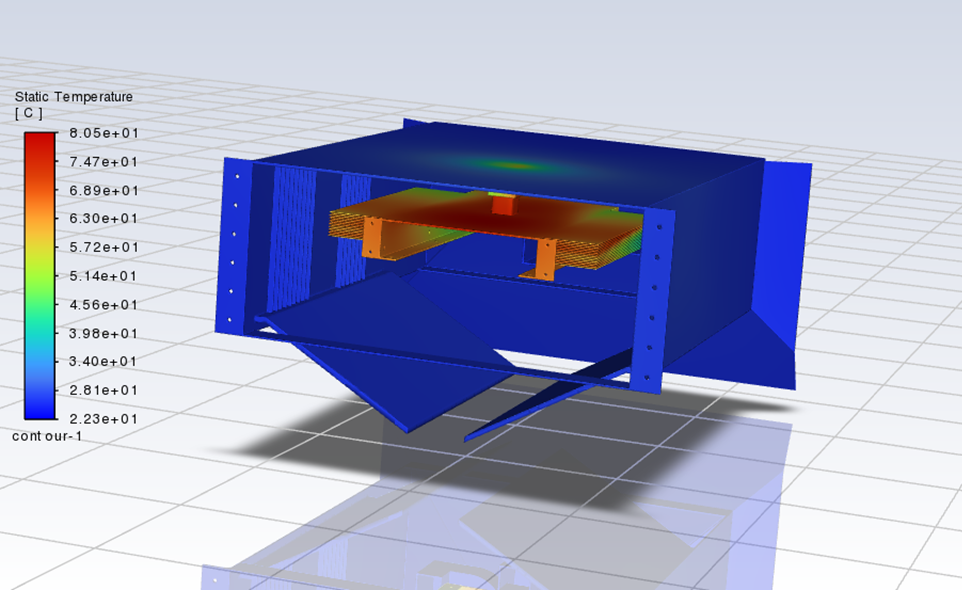
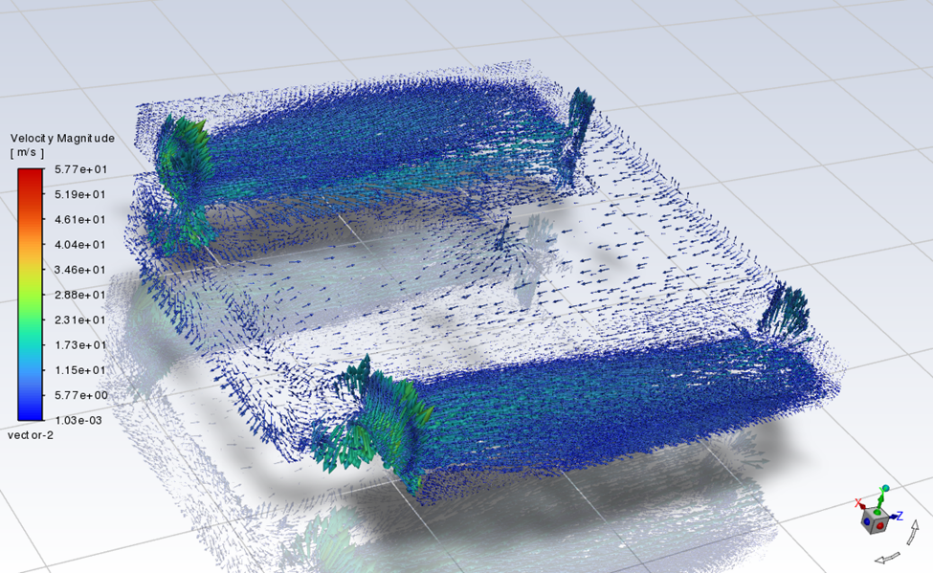
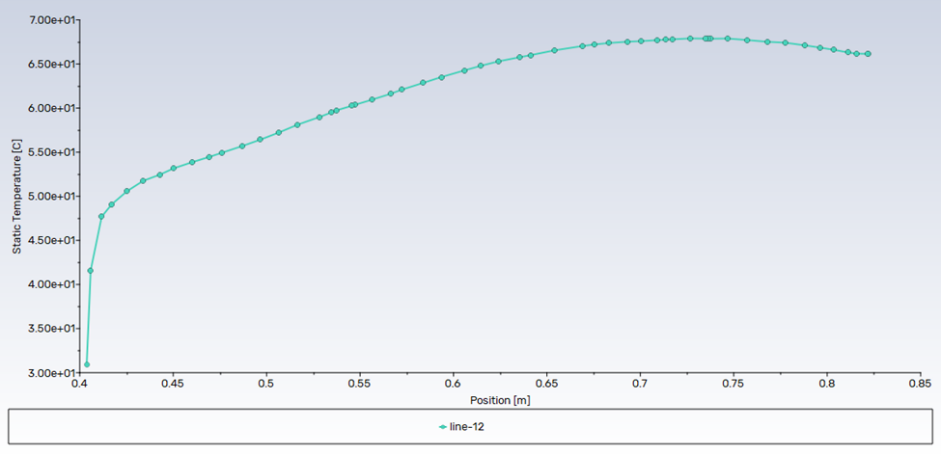


Figure 11. Temperature – Position Plot for a fin along wind direction

Figure 10. Velocity Vector Field of 3rd Iteration of the Container with Fins

Figure 9. Temperature Distribution of the Shell Design

We can tell from the figure that the heat dissipation effect basically meets the requirement. Although the static maximum is 80.5°C, it is acceptable since the power is held constant 300W when the actual server power is dynamic.

To specifically investigate the temperature change on a specific fin, a line was drawn on a fin from near along the wind direction. The result is shown in Figure 11. There is an obvious temperature change along the fin. At the upstream of the air flow field, the temperature is 30°C, which is very close to the environment temperature.

However, the temperature rises very rapidly from 30°C to 55°C within 8cm. The detailed velocity field is shown in Figure 11. Due to the close arrangement of the fins, some air does not flow through. However, the flow between each fin is steady and appears laminar, leading to a smooth temperature curve.

* Heating sheet convective cooling tests (conducted in natural environment and wind tunnel)

图片包含 游戏机, 飞机, 飞机场

已自动生成说明During the heating sheet convective test, we attached a regular CPU sized aluminum heating sheet to the heat pipe. This sheet has an approximately 300W power input and can be used to simulate the working mold of a CPU. The heating sheet is heated repeatedly both under natural environment and convective environment to simulate an on-board mode. Both wind tunnel experiments and on-board driving tests were conducted to acquire indirect and firsthand data.

Figure 12. Heating sheet convective test

Figure 13. Wind tunnel experiment

* On board experiments (deployed the device on board to conduct real-life driving tests)

汽车停在路上

已自动生成说明We brought an SUV to deploy our device and conducted real-life on-board tests. By driving the vehicle around the campus at different speeds, we acquired multiple firsthand temperature and velocity data to verify the effectiveness of passive cooling.

Figure 14. On-board Experiment

* Spray tests (conducted in wind tunnel to verify the waterproof effectiveness)

To verify the effectiveness of waterproofing, we hypothesized that huge drops of water cannot significantly affect the humidity within the container shell, the main humidity increase is caused by the moisture in the rain. Because huge water droplets cannot be carried within the wind and for the same amount of water, the total contact area between water droplets and air is smaller than the total contact area between moisture and air. Therefore, for moisture, it not only has faster vaporization speed but also can be conveyed by the wind to directly affect the wet container shell’s inner space.

图片包含 室内, 烤箱, 厨房, 房间

已自动生成说明With these hypotheses, we adopted a spray head with moisture mode to simulate rain. Under different wind tunnel speeds, we sprayed moisture to the air collector face on the container shell and recorded the maximum humidity within the container shell.

Figure 15. Spray Test in wind tunnel

3.1.3 Wind tunnel test results

The wind tunnel that can generate uniform wind field has an opening sized 0.5m\*0.5m. In normal experiments, the test object is placed inside the tunnel. However, due to the shell’s big size, we manually added some support beneath it and tested the velocity field. The result is shown in Table 3.1.

**Table 2. Air Speed at Different Locations of the Shell during Wind Tunnel Test**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tunnel Wind Velocity (m/s)** | **Upper Exhaust Manifold Outlet Wind Velocity (m/s)** | **Left (m/s)** | **Right (m/s)** | **Actual Wind Velocity (m/s)** | **Air Flux (m^3/s)** |
| 5.5 | 3.8 | 3.6 | 2.9 | 3.3 | 0.009843 |
| 7.18 | 5.28 | 4.79 | 4.46 | 4.85 | 0.013886 |
| 9.13 | 7.2 | 6.87 | 5.2 | 5.8 | 0.018416 |
| 12.5 | 10.68 | 10.2 | 7.85 | 7.9 | 0.027456 |

Where:

- Tunnel wind velocity represents the velocity that is being set in the tunnel

- Upper, left and right exhaust manifold outlets’ wind velocities represent the wind that is captured for cooling

- Actual wind velocity corresponds to the wind crossing the center of the trumpet shape

- Air flux is the average flux that goes through three exhaust manifolds

Due to water-proof limitations, the area of the exhaust manifold is limited to 0.002867m^3, with a 0.061m width and a 0.047 height. However, Table 3.1 exhibits that the wind velocity at the outlet of the manifolds is very close to actual/environmental wind velocity, indicating a good wind collecting ability. The air flux meets the ideal requirements of 0.00487m^3/s with all four velocities. Among all three manifolds, the one at the top has the highest outlet wind velocity and should have the highest heat dissipation ability.

3.1.4 Spray test results

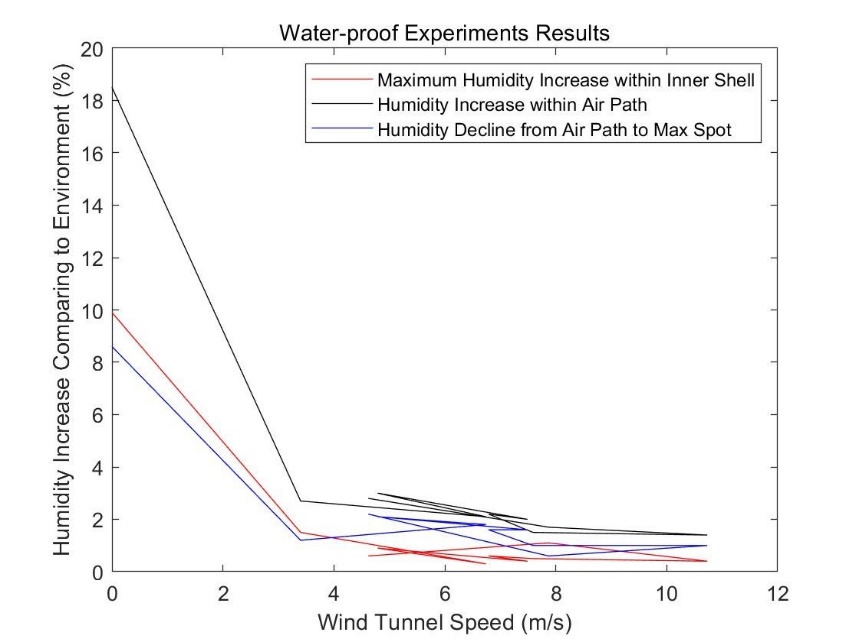
The test results that demonstrated the water-proof effectiveness proved to be extraordinarily positive. From the experiments, we recorded three sets of data. The first is the difference between the maximum humidity spot within the inner shell and the environmental humidity, second is the difference between the humidity within the air path and the environmental humidity, the third is the difference between the maximum-spot-humidity and the path humidity. The result is plotted as a figure below. The choice of the demonstrated values can avoid the influence exerted by shift of environmental humidity, which proved constant during our tests. The results demonstrate three curves have a same tendency in declining, which is odd compared to most people’s hypothesis. The humidity curve slowly declines as the wind tunnel speed increases, more oddly, there is a sharp decline, or rather, a sharp increase when the wind speed decreases close to 0m/s. This result is evidence of successful mechanical design. Since vehicles are normally operating at a high speed, a lower humidity increase at higher speed is much more desired. And the amount of increase in humidity is also satisfying. From Figure 24, the values that influence the electrical devices, increase of humidity inside the inner shell are normally within the range of 1% to 2%. This tiny amount demonstrates a fact that the capability of waterproof made rain drops negligible. Concerning the humidity increase at very low temperatures, more prospective prevention may be done, but the situation is not severe as well, since the increase of humidity within the inner shell is still lower than 8%. Next, I will raise a reasonable theory to explain the possible reason for these oddly declining curves.

Figure 16. Water-proof Effectiveness Visualization

Previously, I have noted that huge raindrops don’t have a significant effect on the increase of humidity, most humidity increase is brought up by moisture or extremely tiny water drops. Because they can be carried by the wind and directly affect humidity within the shell. However, the previous theory is a naive version. It can only be applied at very low wind speeds. A more complete version would be that the significance of moisture decreases as wind tunnel speed increases and the significance of huge rain drops increase as wind tunnel speed increases. This is because the air path is designed to have a vertical upward pipe. This section of piping makes no difference when wind speed is low. However, as wind speed increases, the vertically upward pipe wall could become a block to the moisture. Moisture that is carried with the wind hits the wall and condensates as small water drops that cannot be further carried away. Thus, the air passed the air path becomes dryer than before. This effect becomes more significant when the wind speed increases. So, the decline in moisture can be explained. Also, as the wind speed increases, the evaporation of huge water drops becomes more significant. Therefore, the amount of vapour increases as the wind speed increases. This could lead to an increase in humidity as the wind speed continues to grow since vapour cannot be blocked by a vertical wall. However, this effect has not been seen in wind tunnel experiments. Thus, we propose a prediction in a slight increase in humidity when the velocity is huge enough.

3.1.5 Campus driving test

Finally, we placed the shell on top of one vehicle, connected by a roof rack. For stability, the three manifolds’ inlets are at the bottom, shown in Figure 3.1.1. It is worth noting that we did not deploy a server, since it needed a customized heat pipe with fins to transfer heat from the heat component to the fins. Therefore, we replaced the server with an aluminum block, which has a constant power of 300W. The dynamic power and corresponding temperature calculation is conducted later.

**Table 3. Campus Driving Test Data**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Speed (km/h)** | **Temperature (°C)** | **Film Temperature (°C)** | **Pr** | **Kinematic Viscosity v (m²/s)** | **Thermal Conductivity k (W/m·K)** |
| 20 | 111.1 | 63.35 | 0.705110 | 1.956e-5 | 0.023599 |
| 25 | 103.4 | 59.50 | 0.704571 | 1.917e-5 | 0.023884 |
| 30 | 96.5 | 56.05 | 0.704088 | 1.883e-5 | 0.024139 |
| 31 | 100.0 | 57.80 | 0.704333 | 1.900e-5 | 0.024010 |
| 32 | 94.2 | 54.90 | 0.703927 | 1.871e-5 | 0.024224 |
| 33 | 97.8 | 56.70 | 0.704179 | 1.889e-5 | 0.024091 |
| 35 | 98.0 | 56.80 | 0.704193 | 1.890e-5 | 0.024084 |
| 46 | 87.1 | 51.35 | 0.703430 | 1.835e-5 | 0.024487 |
| 48 | 85.6 | 50.60 | 0.703325 | 1.828e-5 | 0.024542 |
| 49 | 90.9 | 53.25 | 0.703696 | 1.855e-5 | 0.024346 |
| 50 | 89.3 | 52.45 | 0.703584 | 1.847e-5 | 0.024406 |

The result of the campus driving test is shown in Table 3.3. The environmental temperature is 15.5°C. The driving speed of the vehicle ranges from 20km/h to 50km/h, and the equilibrium temperature at the surface of the heat source (aluminum block) is measured by a thermocouple on it. In the table, the film temperature is the average temperature of the block surface temperature and the environment temperature. The film temperature is then used to get the property, kinematic viscosity and thermal conductivity. The three property values are calculated from Table A.4 of “Fundamentals of Heat and Mass Transfer” [37].

We can tell from Table 3.3 that the measured temperature data are beyond the expected range. Even if the driving speed reached 50km/h, the surface temperature is 89.3°C, which is 14.3°C above the subscribed temperature limit.

Complete temperature vs. time curves and calculation results are demonstrated below.

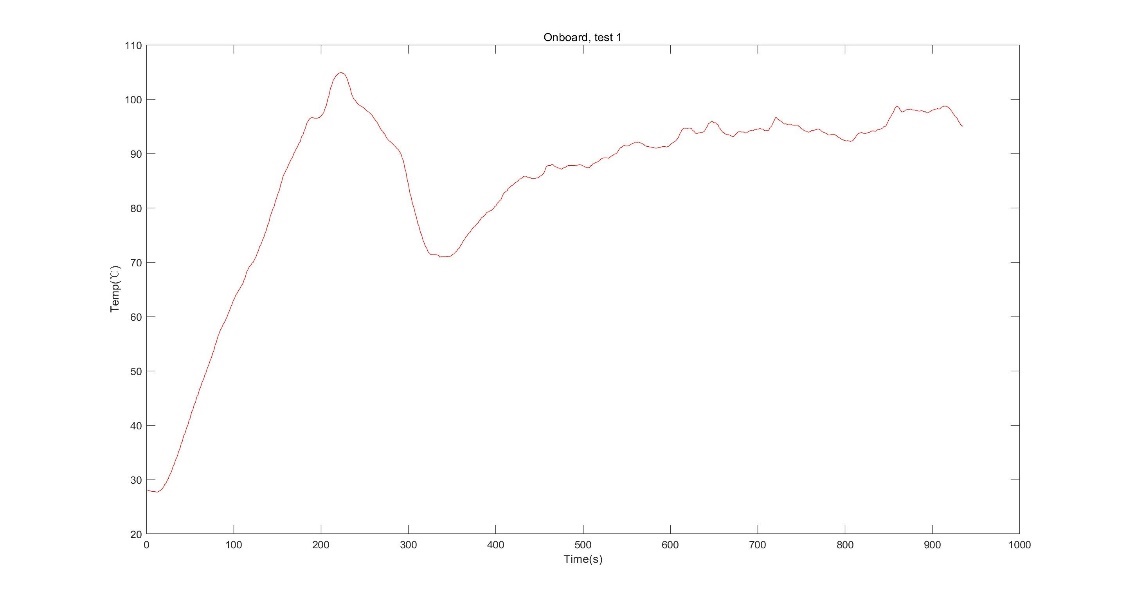
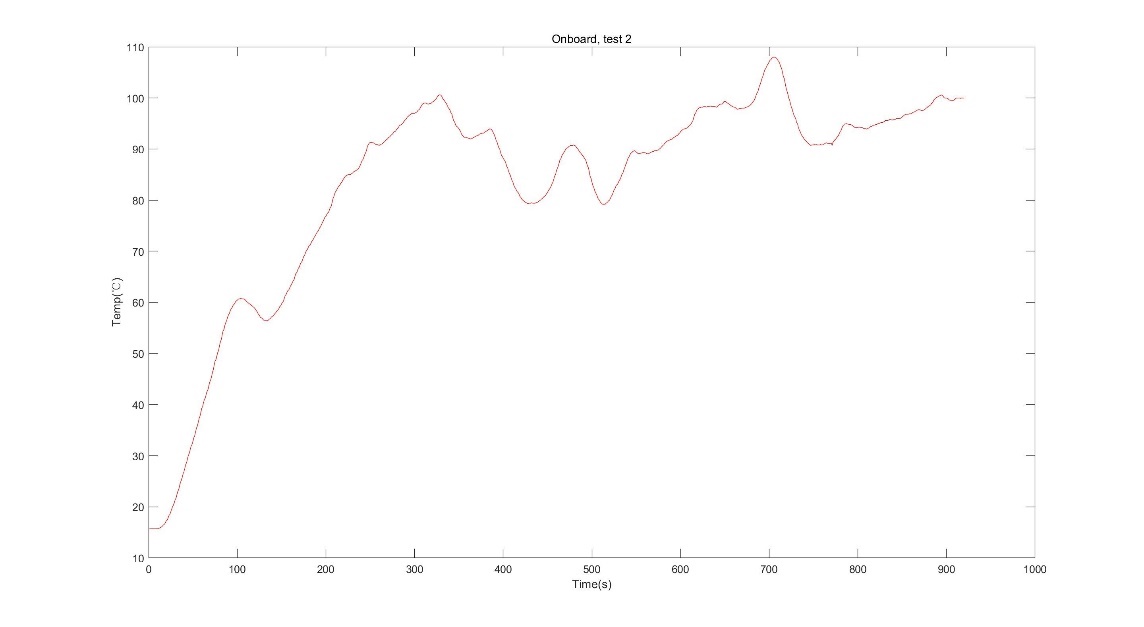
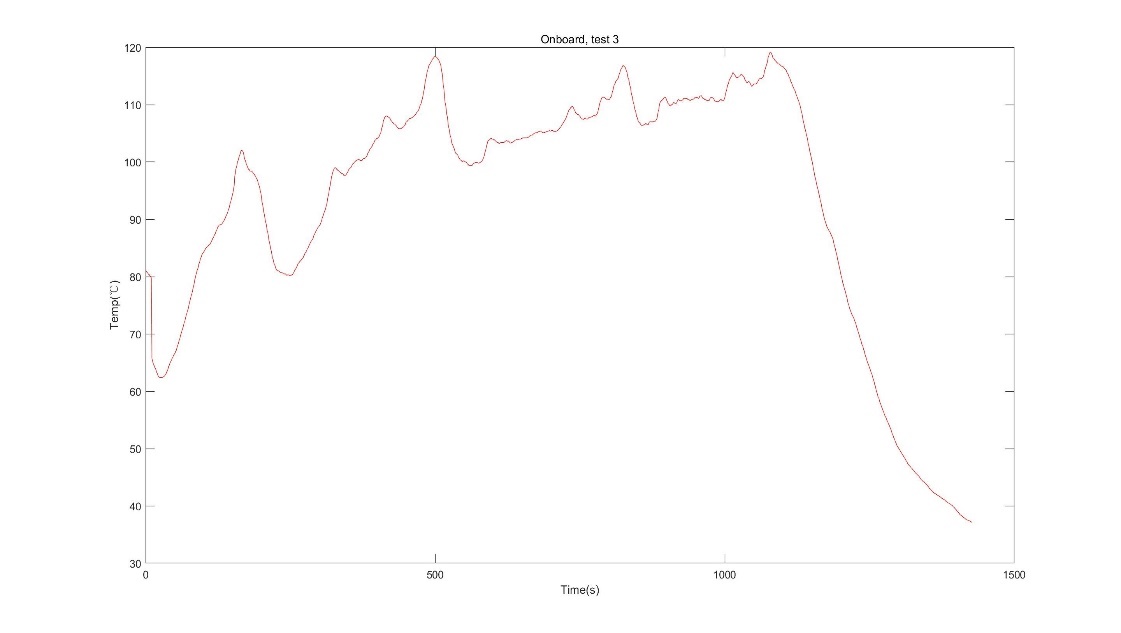


Figure 19. Temperature Curve of Onboard Test 3

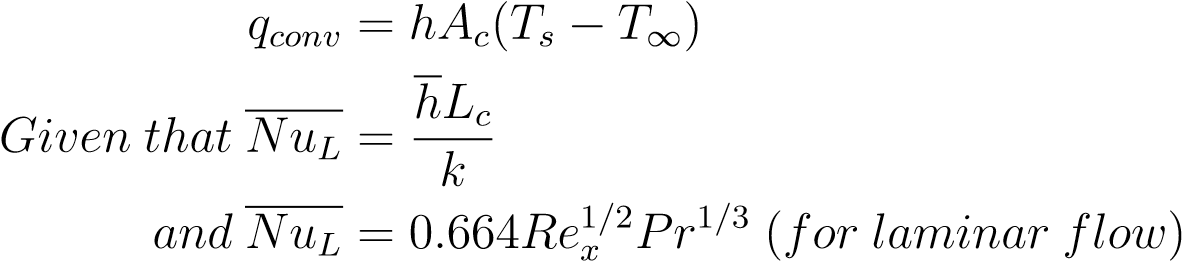
Figure 18. Temperature Curve of Onboard Test 2

Figure 17. Temperature Curve of Onboard Test 1

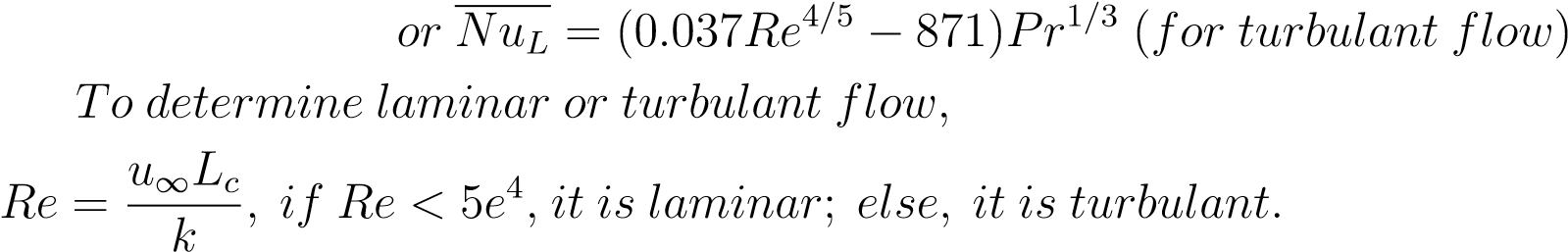
Figure 17 to 19 demonstrated the temperature curve of the heating pads when it is being heated constantly. The sharp decline in the curves represent power being turned off, which is to avoid overheat. Neglecting these sharp declines, we can see that countless fluctuations exist in the curves, some of which are particularly flat. These flat areas are recorded on a table, together with the current vehicle speed. Since the power is constantly turned on, we can confidently infer that the heat transferred from the heat pipe to the air is equivalent to the heat transferred to the heating pad. Since we can read the power output from a removable power supply, we then would have a quantitative measure of the cooling effectiveness.

The prerequisites for deriving a result of cooling effectiveness are the heat and mass transfer functions [33][34][35][36].

*Pinput* = *qconv* (3)

 (4)

(5)

 (6)

(7)

(8)

(9)

Given these equations, together with necessary temperature related air constants (Prantl Number, k, kinematic viscosity, *Tsurface*, *T*∞, *u*∞) derived from Table: Thermophysical Properties of Gases at Atmospheric Pressure[37], we have all the information to derive a relationship between air velocity, environmental temperature vs. the maximum convection heat that can be cooled from this device. From the messy information, it should be noted that the main focus should be put on the characteristic area *Ac*. The heat pipe is stacked by multiple layers of aluminum piece, so the characteristic contact area could be influenced by stacked aluminum pieces each other. Another unknown is characteristic length *Lc* in the Reynolds number equation, it can be directly substituted by the length of the heat pipe[38], which is approximately 0.1m.

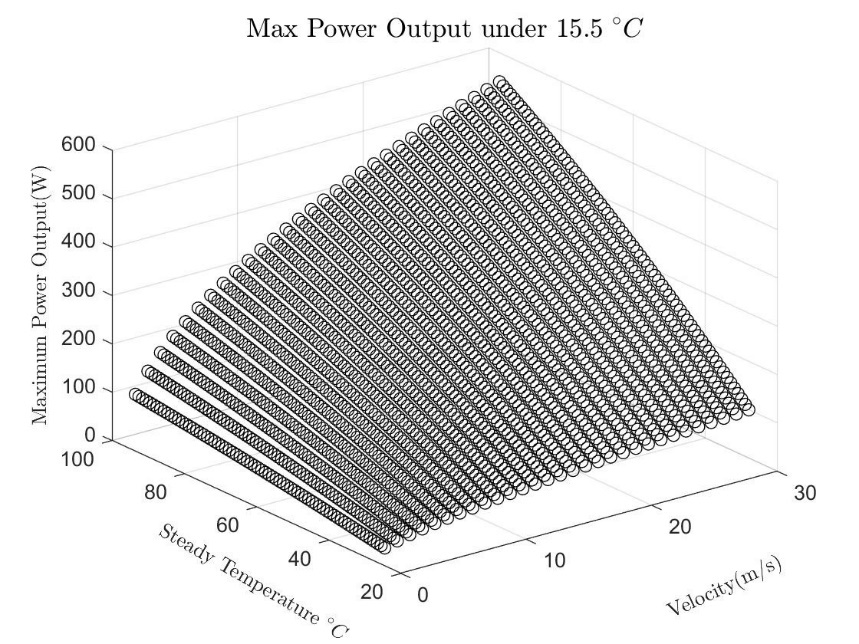
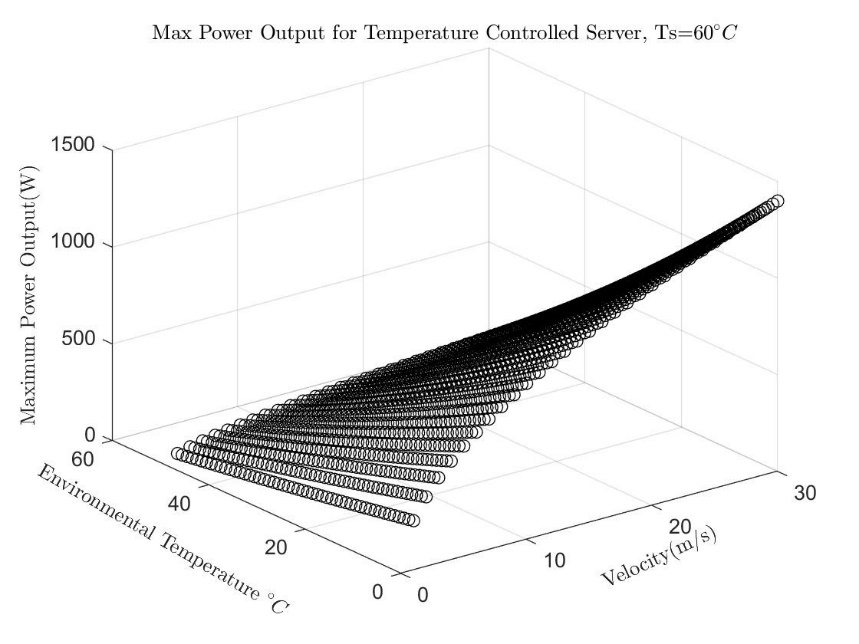
In Figure 20 and Figure 21, the same result is demonstrated in two different ways. To demonstrate the result, there are 4 variables to be controlled[39], Environmental Temperature *T*∞, air velocity *u*∞, steady temperature *Ts* from a constantly heated block and the corresponding power input *P*. We kept one variable as constant, one variable dependent and two as independent variables. In Figure 20, we kept the environmental temperature constant at 15.5 ◦*C*, which is the temperature which we conducted one of our experiments and is considered a suitable temperature under most conditions. In Figure 21, the constant value was the steady temperature, which is set at 60 ◦*C* that comes from the balance between the heat power and convective cooling rate. Therefore, Figure 21 is more desired. Because given such plots, we can quantitatively yield the computing rate of the computer, by inputting the environmental temperature, current air velocity and a previously set maximum temperature that can be maintained in the targeted unit. The output can be easily yielded as the maximum running power that can be generated by the targeted unit. This process can be easily done by a python program. The maximum running power can then also be used as an input to maintain the computing intensity of the CPU and GPU [40]. Normally they are most concerned in heat pipe cooling.

Figure 21. Cooling Effectiveness Result under Constant 60 ◦C Steady Temperature

Figure 20. Cooling Effectiveness Result under 15.5 ◦C Environmental Temperature

From Figure 20 and 21, we can see that the passive cooling effectiveness under low temperature and high air velocity has a bright result. Under the speed of 30m/s and temperature of 0 ◦*C*, the convective cooling rate can go up to 1000W, which is effective for almost all fully functioned edge computing servers [41]. It should be noted that the results are derived from one heat pad and one heat pipe to cool down.Since our device have a volume for three identical heat pipes when a server is installed. Therefore, the maximum power allowed in Figure 22 and 23 provide a weakened result of the cooling effectiveness. In reality, the maximum cooling capacity could be three times that demonstrated in Figure 22 and 23.

It also ought to be noted that in Figure 23, we set the allowed temperature of computing unit as 60 ◦*C*, which is a conservative estimation. The real maximum power allowed should be over 70 ◦*C*. Therefore, our designed product can have a very prospective performance in the future when operating at highways or a high latitude area.

However, under high environmental temperatures or low air velocity (low vehicular speed), the cooling rate is lowered to a dangerous level. Even if we increase the threshold temperature or time the max power of cooling by three. Although in our vision, this problem can be settled by a feedback control algorithm over the maximum computing power, a water cooling or other active cooling method may still be applied to handle the situations when a high computing operation speed is required even under a low vehicular speed.

3.2 CPU Temperature Self-adjustment

3.2.1 PID control

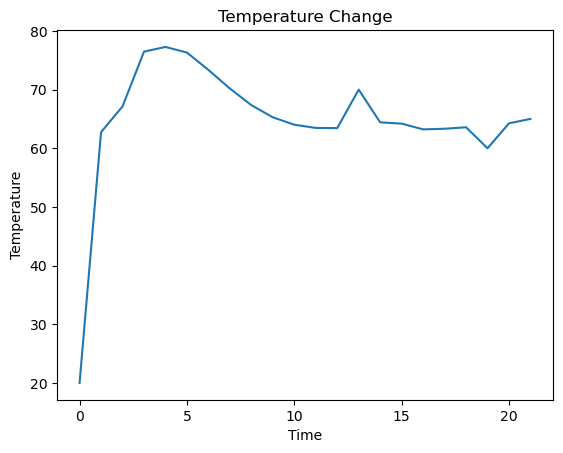
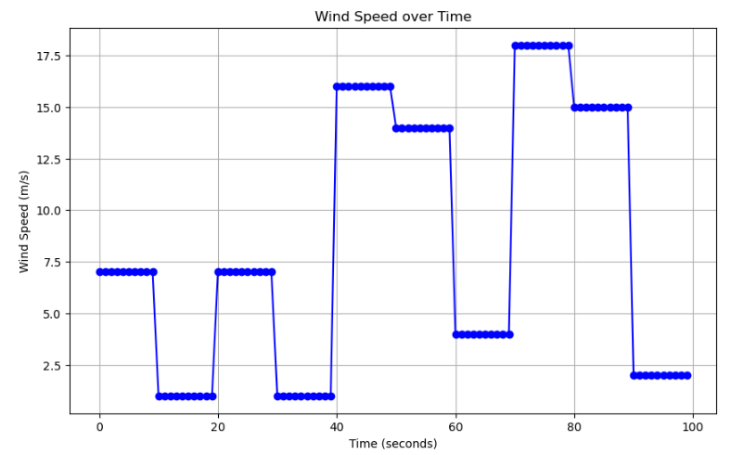
Firstly I tried PID control, the result is shown in the figure. We can see that the CPU temperature has the trend to stay around 60-70 degrees of Celsius but without surpassing the threshold value.

Figure 22. The simulation of PID control

3.2.2 Reinforcement Learning

Secondly, I test the effectiveness of the algorithm through simulation. There’re two reasons why I’m not trying to launch an experiment outdoors practically. One is that I cannot guarantee that if I disassemble the server, it can be restored intact later. The second is that if I make the code to execute training by interacting with the environment, the wind speed is relatively small as it’s not moving. In this case, the training process cannot help the CPU to smartly respond to higher wind speed.

Considering these, I decided to simulate the practical wind speed of the server may encounter at the roof of a vehicle by generating random integers very 1 second at the range of 0 to 20 m/s. The simulation time is 100 seconds.

Here shows three plots, the simulated environmental wind speeds varying with time, the CPU decisions on choosing its working frequency versus time, the simulated CPU temperature versus time, respectively. We can see again that the CPU temperature has the trend to stay around 60-70 degrees of Celsius but without surpassing the threshold value.

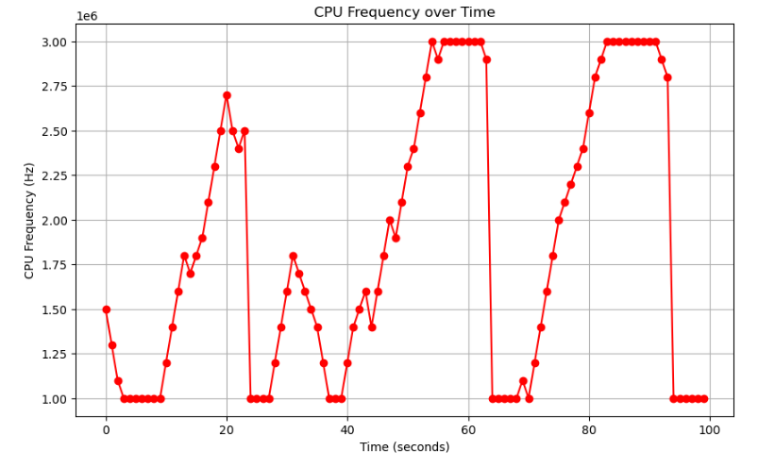


Figure 23. The simulation of wind speed

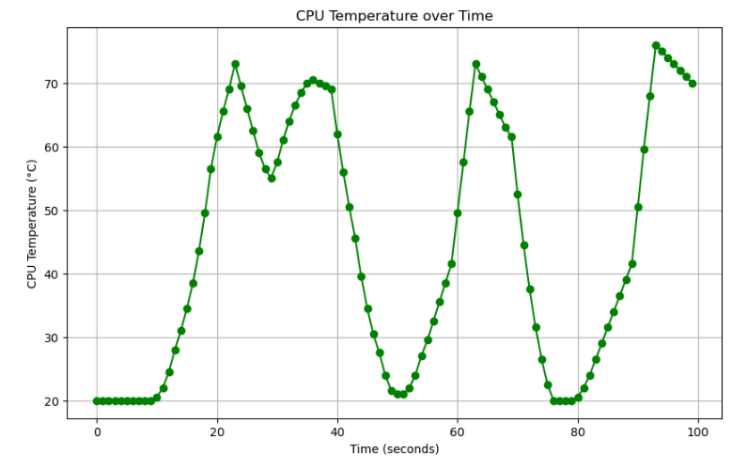


Figure 24. The simulation of CPU frequency

Figure 25. The simulation of CPU temperature

4. Cost

**4.1 Parts**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table 4. Parts Costs** | | | | |
| **Part** | **Manufacturer** | **Retail Cost ($)** | **Bulk Purchase Cost ($)** | **Actual Cost ($)** |
| POM Boards & Parts | Individual | 2.76 | 0.95 | 317.63 |
| Sheet Aluminum Parts | Individual | 7.98 | 2.93 | 59.38 |
| Coonector & Tools | Individual & Deli |  |  | 74.57 |
| Thermal Conductive Silicon | Leading Runner Company |  |  | 10.36 |
| Heat Pipes | Individuals |  |  | 16.43 |
| Aluminum Heating Sheet | Dayu Electronic Heat |  |  | 39.36 |
| Anemometer | Xima Tester |  |  | 18.78 |
| Thermal Couple | TASI |  |  | 52.2 |
| Bondage | Xun Ting |  |  | 4.64 |
| Server | Hankong |  |  | 130 |
| Total |  |  |  | 723.29 |

**4.2 Labor**

From the internet, we know that average salary for an ECE student from Illinois would be

$87,769 per year. The average salary per hour is $36. We assume 160 hours are needed for each student in this project. Therefore, the labor can be estimated by the equation:

$36 × 4 × 160 = $23040

5. Conclusion

**5.1 Accomplishments**

We’ve designed a server shell, and verified the cooling effectiveness of the shell both by simulating in software and conducting the physical experiments on-board a vehicle. We’ve designed the algorithm of CPU frequency self-adjustment by two methods, PID control and reinforcement learning, and also the simulation of applications of the server about path selection and photo recognition.

**5.2 Uncertainties**

For mechanical parts, due to cost control, a real server was never installed in our device, therefore, we can predict but cannot be certain about the effectiveness when an actual server is deployed.

The deployment of our device on a vehicle requires careful bondage on a roof rack. If the vehicle happens to travel with a high acceleration, the bondage may not prevent the device from slipping, due to heavy weight.

Currently, we only designed a container for a specific size of server. If a server of very small size or huge size requires to use our device, it may be undeployable.

The entire device is too heavy even with the light materials. It could exhaust the roof rack or the roof of the vehicle once constantly used.

For algorithm parts, as it’s difficult to get the exact dynamics of the CPU temperature, I failed to simulate the result of the exact CPU temperature versus time. Firstly, for the simulation of the wind speed, I only assumed the change of the wind speed at certain points but the wind speed kept unchanged during each period of one minute. However, in real scenario, the wind speed may change at any time.

Secondly, for the simulation of the CPU temperature changing, I only assumed the heat dissipation rate is linear with the wind speed. However in practical case, the CPU temperature changes based on both the environmental wind speed and its temperature. The wind speed and the heat dissipation are positive related, but not linear. It’s a sophisticated relation between wind speed, CPU temperature, and heat dissipation.

**5.3 Ethical considerations**

Some additional risks could be caused when a huge burden is deployed on a vehicle, in our case, a huge container shell is deployed on the roof of a vehicle. Following risks could be caused.

* A heavy weight added on the roof of car, the roof has a risk of collapse.
* An extra mobile addition of potential energy on the vehicle, causing risks of damaging self, other vehicles or road surfaces.
* An extra addition of height to the vehicle. When entering some height limited places like an underground parking lot, the roof of parking lot and the device itself could be damaged.

**5.4 Future work**

For mechanical parts, we want to improve the cumbersome steps by designing a matching roof rack, so that the device can be directly installed in it.

In the future, adjustable server container and air paths can be manufactured in order to fit different sized servers.

The device is still too heavy even with light materials, we want to improve the material selection to specifically designed hollow materials that can lighten the device even further without damaging its strength.

During our wind tunnel experiments, we discovered that too much wind was dissipated from the edge of the bell-mouth shaped front. Therefore, the design of straight boards may be too ill-considered, a specific curve-shaped front board may be better in air accumulating.

For algorithm parts, for the CPU frequency controlling algorithm, we only simulate in python. That will contain error to certain extent. In order to be more accurate, it's recommended to perform reinforcement learning outdoors to make the server interact with the outer environment directly.

It's necessary to equip the server the ability of adapting different wireless communication protocols. Not only communicating with the smartphone or laptop through the Internet, the server is expected to directly communicate with the user end, for example another server or infrastructure in IOT application.

# References

[1] L. Li, C. Jin, and X. Bai. “”A determination method on the supply airflow rate with the impact of leakage and bypass airflow of computer room air conditioning”.” (), [Online]. Available: https://www.sciencedirect.com/science/article/pii/ S2352710222010713 (visited on 11/01/2022).

[2] Sheng Du, Quan Zhang, Sikai Zou, Fanxi Meng, Lijun Liu, "Simulation analysis on energy consumption and economy of CPU cooling system based on loop heat pipe for data center," *Thermal Science and Engineering Progress*, vol. 45, 2023, Art. no. 102115. doi: 10.1016/j.tsep.2023.102115.

[3] F.M. Naduvilakath-Mohammed, M. Lebon, G. Byrne, A.J. Robinson, "An experimental investigation of vapor compression refrigeration cooling and energy performance for CPU thermal management," *Case Studies in Thermal Engineering*, vol. 54, 2024, Art. no. 103963. doi: 10.1016/j.csite.2023.103963.

[4] Z. Xiong, M. Zhao, Z. Yuan, J. Xu, L. Cai, "Energy-saving optimization of application server clusters based on mixed integer linear programming," *Journal of Parallel and Distributed Computing*, vol. 171, pp. 111-129, 2023. doi: 10.1016/j.jpdc.2022.09.009.

[5] Y. Hao, J. Cao, T. Ma, S. Ji, "Adaptive energy-aware scheduling method in a meteorological cloud," *Future Gener. Comput. Syst.*, vol. 101, pp. 1142-1157, 2019. doi: 10.1016/j.future.2019.07.061.

[6] S. Kwon, L. Ntaimo, N. Gautam, "Demand response in data centers: integration of server provisioning and power procurement," *IEEE Trans. Smart Grid*, vol. 10, no. 5, pp. 4928-4938, 2019. doi: 10.1109/TSG.2018.2871125.

[7] E. Parvizi, M.H. Rezvani, "Utilization-aware energy-efficient virtual machine placement in cloud networks using NSGA-III meta-heuristic approach," *Clust. Comput.*, vol. 23, pp. 2945-2967, 2020. doi: 10.1007/s10586-020-03060-y.

[8] L. Qi, Y. Chen, Y. Yuan, S. Fu, X. Zhang, X. Xu, "A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems," *World Wide Web*, vol. 23, pp. 1275-1297, 2020. doi: 10.1007/s11280-019-00684-y.

[9] A. Asghari, M.K. Sohrabi, "Server placement in mobile cloud computing: A comprehensive survey for edge computing, fog computing and cloudlet," *Computer Science Review*, vol. 51, 2024, Art. no. 100616. doi: 10.1016/j.cosrev.2023.100616.

[10] X. Fei, K. Li, W. Yang, K. Li, "Analysis of energy efficiency of a parallel AES algorithm for CPU-GPU heterogeneous platforms," *Parallel Computing*, vol. 94-95, 2020, Art. no. 102621. doi: 10.1016/j.parco.2020.102621.

[11] W. Lin, W. Lin, J. Lin, H. Zhong, J. Wang, L. He, "A multi-agent reinforcement learning-based method for server energy efficiency optimization combining DVFS and dynamic fan control," *Sustainable Computing: Informatics and Systems*, vol. 42, 2024, Art. no. 100977. doi: 10.1016/j.suscom.2024.100977.

[12] Y. Fulpagare, K.-R. Huang, Y.-H. Liao, C.-C. Wang, "Optimal energy management for air cooled server fans using Deep Reinforcement Learning control method," *Energy and Buildings*, vol. 277, 2022, Art. no. 112542. doi: 10.1016/j.enbuild.2022.112542.

[13] W.-X. Chu, Y.-H. Lien, K.-R. Huang, C.-C. Wang, "Energy saving of fans in air-cooled server via deep reinforcement learning algorithm," *Energy Reports*, vol. 7, pp. 3437-3448, 2021. doi: 10.1016/j.egyr.2021.06.003.

[14] W. Material, *Aluminum 6061, al 6061-t6 alloy properties, density, tensile yield strength, thermal conductivity, modulus of elasticity, welding*, https://www.theworldmaterial.com/ al-6061-t6- aluminum-alloy/, 2024-04.

[15] Omnexus, *Complete guide on polyoxymethylene (pom) or polyacetal plastic*, <https://omnexus.specialchem.com/selection-guide/polyacetal-polyoxymethylene-pom-plastic>, 202404.

[16] Aerographer/Meteorology, *Funnel effect*, https://www.tpub.com/weather2 /3- 22.htm#google*vignette*, 2024-04.

[17] O. Capatina, T. Grosan, I. Vadan, R. Trimbitas, S. Ignat, and A. Calarasu. “”The funnel effect and its practical benefits in wind applications”.” (), [Online]. Available: <https://ieeexplore.ieee.org/document/5520784> (visited on 05/28/2010).

[18] T. A. Team, *Kinematic equations: Explanation, review, and examples*, <https://www.albert.io> /blog/kinematic-equations/, 2024-04.

[19] T. Engineering, *What is heat equation – heat conduction equation – definition*, <https://www.thermal-engineering.org/what-is-heat-equation-heat-conduction-equation-> definition/, 2024-04.

[20] ALLOYWIKI, *Aluminum 1060*, https://alloy.wiki/aluminum-1060/, 2024-04.

[21] A. Materials, *Aluminium / aluminum 1060 alloy (uns a91060)*, <https://www.azom.com/> article.aspx?ArticleID=6587, 2024-04.

[22] P. Plastics, *Thermal properties of plastic materials*, <https://www.professionalplastics.com/> professionalplastics/ThermalPropertiesofPlasticMaterials.pdf, 2024-04.

[23] Omnexus, *Density of plastics: Technical properties*, <https://omnexus.specialchem.com/polymer-> property/density, 2024-04.

[24] Matmake, *Young’s modulus of common polymers and plastics*, https://matmake.com/properties/ youngs-modulus-of-polymers-and-plastics.html, 2024-04.

[25] S. L., V. P.M., and S. Chandran. “”Polyoxymethylene Handbook: Structure, Properties, Applications and Their Nanocomposites”.” (), [Online]. Available: [https](https://www.researchgate.net/publication/291854787_Polyoxymethylene_Handbook_Structure_Properties_Applications_and_Their_Nanocomposites) [:](https://www.researchgate.net/publication/291854787_Polyoxymethylene_Handbook_Structure_Properties_Applications_and_Their_Nanocomposites)  [//www.researchgate.net/publication/291854787](https://www.researchgate.net/publication/291854787_Polyoxymethylene_Handbook_Structure_Properties_Applications_and_Their_Nanocomposites) [Polyoxymethylene](https://www.researchgate.net/publication/291854787_Polyoxymethylene_Handbook_Structure_Properties_Applications_and_Their_Nanocomposites) [Handbook Structure](https://www.researchgate.net/publication/291854787_Polyoxymethylene_Handbook_Structure_Properties_Applications_and_Their_Nanocomposites) [Properties](https://www.researchgate.net/publication/291854787_Polyoxymethylene_Handbook_Structure_Properties_Applications_and_Their_Nanocomposites) [Applications](https://www.researchgate.net/publication/291854787_Polyoxymethylene_Handbook_Structure_Properties_Applications_and_Their_Nanocomposites) [and](https://www.researchgate.net/publication/291854787_Polyoxymethylene_Handbook_Structure_Properties_Applications_and_Their_Nanocomposites) [Their](https://www.researchgate.net/publication/291854787_Polyoxymethylene_Handbook_Structure_Properties_Applications_and_Their_Nanocomposites) [Nanocomposites](https://www.researchgate.net/publication/291854787_Polyoxymethylene_Handbook_Structure_Properties_Applications_and_Their_Nanocomposites) (visited on 03/2014).

[26] Plastiker, *Real time price list*, https://plasticker.de/preise/pms*en.php*?*kat* = *Mahlgutaog* = *Ashow* = *okmake* = *ok*, 2024-04.

[27] Omnexus, *Polyetheretherketone (peek): A complete guide on high-heat engineering plastic*, https://omnexus.specialchem.com/ selection-guide/polyetheretherketone-peek-thermoplastic, 2024-04.

[28] Alibaba, *Peek raw material peek granules price of peek per kg*, <https://www.alibaba.com/productdetail/Peek-raw-Material-PEEK-granules-> price1600865620848*.html*, 2024-04.

[29] D. Brodowski, N. Golde, R. J. Wysocki, and V. Kumar, "CPU frequency and voltage scaling code in the Linux(TM) kernel,". Available: <https://www.kernel.org/doc/Documentation/cpu-> freq/governors.txt.

[30] J. Monsen, "Fundamentals of PID Control," 26 June 2023. Available: <https://www.isa.org/intech-> home/2023/june-2023/features/fundamentals-pid-control.

[31] Synopsys, "What is Reinforcement Learning?". Available: <https://www.synopsys.com/ai/what-> is-reinforcement-learning.html.

[32] dl. (June 26, 2020). Quick Start Deep Learning tutorial based on PyTorch. Retrieve source:CSDN:https://blog.csdn.net/tensixchuan/article/details/1068629024

[33] T. Bergman, “Fundamentals of heat and mass transfer,” in 8th ed. Wiley, Dec. 2018, ch. 7, pp. 459–470.

[34] T. Bergman, “Fundamentals of heat and mass transfer,” in 8th ed. Wiley, Dec. 2018, ch. 7, pp. 403–420.

[35] T. E. Toolbox, *Reynolds number*, <https://www.engineeringtoolbox.com/reynoldsnumber-> d237*.html*, 2024-05.

[36] Pratik, *Nusselt number: Definition, formula, significance, examples*, <https://mechcontent.com/> nusselt-number/, 2024-05.

[37] T. Bergman, “Fundamentals of heat and mass transfer,” in 8th ed. Wiley, Dec. 2018, p. 1021.

[38] Simscale, *What is reynolds number?* <https://www.simscale.com/docs/simwiki/numericsbackground/what-is-the-reynolds-number/>, 2024-05.

[39] CrossVlidated, *What are some popular choices for visualizing 4-dimensional data?* <https://stats.stackexchange.com/questions/123767/what-are-some-popular-choicesfor-> visualizing-4-dimensional-data, 2024-05.

[40] S. Bhalachandra, B. Austin, S. Williams, and N. J. Wright, “Understanding the impact of input entropy on fpu, cpu, and gpu power,” *ArXiv*, vol. abs/2212.08805, 2022. [Online]. Available: <https://api.semanticscholar.org/CorpusID:254854298.>

[41] Y. Li and S. Wang, “An energy-aware edge server placement algorithm in mobile edge computing,” in *2018 IEEE International Conference on Edge Computing (EDGE)*, 2018, pp. 66–73. DOI: [10.1109/EDGE.2018.00016.](https://doi.org/10.1109/EDGE.2018.00016)

Appendix A

|  |  |  |
| --- | --- | --- |
| **Table 5. System Requirements and Verifications** | | |
| Requirement | Verification | Verification status  (Y or N) |
| 1. Wireless network connection | 1. Data transmitting/receiving    1. “Ping” successfully    2. Receive request of a desktop    3. Return the result bask to the desktop | Y |
| 1. CPU frequency adjustment based on wind speed and CPU temperature    1. Keep the CPU temperature below 75℃    2. Release the CPU performance moderately if cooling condition is allowed | 1. Simulation of CPU temperature vs. time curve    1. Code for PID control    2. Code for reinforcement learning | Y |
| 1. Passive Cooling    1. Wind speed accumulating    2. Heat pipe effectiveness    3. Matching between powers | 1. Physical experiments    1. Fan test    2. Wind tunnel test    3. On-board test | Y |
| 1. Water Proof    1. Humidity increase    2. No speed influence | 1. Physical experiments    1. Spray & wind tunnel tests    2. Spray & wind tunnel tests | Y |

Appendix B

Heat Transfer calculation

For the heat transfer, due to the container’s complex shape and 3-tunnel combining effect, the final temperature distribution is done by Fluent. In fact, computational fluid dynamics (CFD) is powerful in analyzing some cases in 3D spaces that some specific local analysis is required [23]. Here we roughly calculate the rate of convective heat loss.

With:

- is the ambient air temperature,

- is the surface temperature of the server,

- is the horizontal length of the server, which is seen as a flat plate,

- is the vertical length of the plate,

- is the wind speed,

- is the kinematic viscosity of the air,

- is the heat conductivity of the air,

- is the property of the air,

- is the heat rate,

- is the average heat convection coefficient

- is the area that meets the minimum requirements

- is Reynolds number.

Known: , , ,,

Analysis:

At 323K:

Therefore, the flow is turbulent.

Use correlation: （2-1） [24]

Notice that

Therefore:

is the average heat transfer coefficient.

For heat rate calculation:

With that

Appendix C

The code of PID control

import matplotlib.pyplot as plt

import random

class PIDController:

def \_\_init\_\_(self, target\_temp, kp, ki, kd):

self.target\_temp = target\_temp

self.kp = kp

self.ki = ki

self.kd = kd

self.error\_sum = 0

self.last\_error = 0

self.temperature\_history = []

self.change\_count = 0

def control(self, current\_temp):

error = self.target\_temp - current\_temp

# Proportional term

p = self.kp \* error

# Integral term

self.error\_sum += error

i = self.ki \* self.error\_sum

# Derivative term

d = self.kd \* (error - self.last\_error)

self.last\_error = error

# Control signal

control\_signal = p + i + d

return control\_signal

def plot\_temperature(self):

plt.plot(self.temperature\_history)

plt.xlabel('Time')

plt.ylabel('Temperature')

plt.title('Temperature Change')

plt.show()

pid\_controller = PIDController(target\_temp=65, kp=0.5, ki=0.25, kd=0.2)

current\_temp = 20

pid\_controller.temperature\_history.append(current\_temp)

while True:

control\_signal = pid\_controller.control(current\_temp)

current\_temp += control\_signal

random\_num = random.randint(1, 10)

pid\_controller.change\_count += 1

if random\_num == 1 and pid\_controller.change\_count >= 5:

if current\_temp !=70:

current\_temp = 60

pid\_controller.change\_count = 0

elif random\_num == 10 and pid\_controller.change\_count >= 5:

if current\_temp !=60:

current\_temp = 70

pid\_controller.change\_count = 0

pid\_controller.temperature\_history.append(current\_temp)

print("Current temperature:", current\_temp)

pid\_controller.plot\_temperature()

if abs(current\_temp - pid\_controller.target\_temp) < 0.1:

current\_temp = pid\_controller.target\_temp

break

The code of reinforcement learning

import numpy as np

import matplotlib.pyplot as plt

# Function to simulate environment (wind speed and temperature)

def simulate\_environment(cpu\_frequency, current\_temperature):

wind\_speed = np.random.randint(0, 20) # Wind speed in m/s

temperature = calculate\_cpu\_temperature(wind\_speed, cpu\_frequency, current\_temperature)

return wind\_speed, temperature

# Function to compute CPU temperature based on wind speed and CPU frequency

def calculate\_cpu\_temperature(wind\_speed, cpu\_frequency, current\_temperature):

max\_cpu\_frequency = 3000000 # Maximum CPU frequency in Hz

min\_cpu\_frequency = 1000000 # Minimum CPU frequency in Hz

baseline\_temp = 20 # Baseline temperature in Celsius without any load or wind

cooling\_effect = (wind\_speed / 20) \* 10 # Cooling effect of wind speed

heating\_effect = (cpu\_frequency - min\_cpu\_frequency) / (max\_cpu\_frequency - min\_cpu\_frequency) \* 10 # Heating effect of CPU frequency

# Adjust temperature based on cooling and heating effects

delta\_temperature = heating\_effect - cooling\_effect

new\_temperature = current\_temperature + delta\_temperature

if new\_temperature <= baseline\_temp:

new\_temperature = baseline\_temp

return new\_temperature

# Function to compute reward based on wind speed, temperature, and CPU frequency

def compute\_reward(wind\_speed, temperature, cpu\_frequency):

performance\_reward = cpu\_frequency / 3000000 # Reward for higher performance

#energy\_consumption\_penalty = (3000000 - cpu\_frequency) / 3000000 # Penalize for high energy consumption

wind\_reward = (wind\_speed - 10) / 10

temperature\_reward = 0

# Encourage CPU temperature to stay within the range from 55 to 70 degrees Celsius

if 55 <= temperature <= 70:

temperature\_reward = 3

elif temperature < 55:

temperature\_reward = 1 # Reward for being too cold, encourage higher frequency

# Penalize for high temperatures

if temperature > 75:

temperature\_reward = -3

elif temperature > 70:

temperature\_reward = -1

total\_reward = performance\_reward + wind\_reward + temperature\_reward

return total\_reward

# Function to select action using epsilon-greedy policy

def select\_action(state, q\_table, epsilon):

if np.random.rand() < epsilon:

action = np.random.randint(4) # Explore

else:

if state in q\_table:

action = np.argmax(q\_table[state])

else:

action = np.random.randint(4)

return action

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

max\_cpu\_frequency = 3000000 # Maximum CPU frequency in Hz

min\_cpu\_frequency = 1000000 # Minimum CPU frequency in Hz

initial\_wind\_speed = np.random.randint(0, 20)

threshold\_wind\_speed = 10 # Threshold wind speed for efficient cooling in m/s

# Initialize Q-table as a dictionary

q\_table = {}

# Define hyperparameters

alpha = 0.1 # Learning rate

gamma = 0.9 # Discount factor

epsilon = 0.1 # Exploration rate

# Initialize current CPU temperature

current\_temperature = 30 # Initial temperature in degrees Celsius

# Training loop

num\_episodes = 1000

max\_steps = 100 # Maximum number of steps per episode

for episode in range(num\_episodes):

cpu\_frequency = 1500000 # Initial CPU frequency

wind\_speed, temperature = simulate\_environment(cpu\_frequency, current\_temperature)

state = (wind\_speed // 2, int(temperature) // 5) # Discretized state representation

steps = 0 # Initialize step counter

done = False

while not done:

# Implement temperature-based logic to force CPU frequency changes

if temperature > 70:

cpu\_frequency = 0 # Set CPU frequency to 0 if temperature is too high

#elif temperature > 65:

# cpu\_frequency = max(cpu\_frequency - 200000, 1000000) # Decrease CPU frequency significantly

#elif temperature < 55:

# cpu\_frequency = min(cpu\_frequency + 200000, 3000000) # Increase CPU frequency significantly

action = select\_action(state, q\_table, epsilon)

if action == 0: # Increase CPU frequency only if temperature is safe

cpu\_frequency += 200000

elif action == 1:

cpu\_frequency += 100000

elif action == 2: # Decrease CPU frequency

cpu\_frequency -= 100000

elif action == 3:

cpu\_frequency -= 200000

cpu\_frequency = np.clip(cpu\_frequency, 1000000, 3000000)

wind\_speed, temperature = simulate\_environment(cpu\_frequency, temperature)

next\_state = (wind\_speed // 2, int(temperature) // 5) # Discretized state representation

reward = compute\_reward(wind\_speed, temperature, cpu\_frequency)

if state not in q\_table:

q\_table[state] = np.zeros(4)

if next\_state not in q\_table:

q\_table[next\_state] = np.zeros(4)

q\_table[state][action] += alpha \* (reward + gamma \* np.max(q\_table[next\_state]) - q\_table[state][action])

state = next\_state

steps += 1

if steps >= max\_steps:

done = True

print(f"Episode {episode + 1}: Total Reward = {reward}")

print("Training completed!")

# Simulation parameters

initial\_temperature = 20 # Initial temperature in degrees Celsius

initial\_cpu\_frequency = 1500000 # Initial CPU frequency in Hz

simulation\_time = 100 # Simulation time in seconds

# Simulate wind speeds

wind\_speeds = np.random.randint(0, 20, size=10) # Wind speed in m/s

# Initialize CPU temperature, frequency, and action arrays

cpu\_temperatures = [initial\_temperature]

cpu\_frequencies = [initial\_cpu\_frequency]

actions = []

# Simulate CPU temperature over time

for i in range(1, simulation\_time):

state = (wind\_speeds[i//10] // 2, int(cpu\_temperatures[i - 1]) // 5)

action = select\_action(state, q\_table, epsilon)

if cpu\_temperatures[i - 1] > 70:

cpu\_frequency = min\_cpu\_frequency # Set CPU frequency to 0 if temperature is too high

#elif cpu\_temperatures[i - 1] > 65:

# cpu\_frequency = max(cpu\_frequencies[i - 1] - 200000, min\_cpu\_frequency) # Decrease CPU frequency significantly

#elif cpu\_temperatures[i - 1] < 60:

# cpu\_frequency = min(cpu\_frequencies[i - 1] + 200000, max\_cpu\_frequency) # Increase CPU frequency significantly

else:

if action == 0: # Increase CPU frequency

cpu\_frequency = cpu\_frequencies[i - 1] + 200000

elif action == 1:

cpu\_frequency = cpu\_frequencies[i - 1] + 100000

elif action == 2:

cpu\_frequency = cpu\_frequencies[i - 1] - 100000 # Decrease CPU frequency

elif action == 3:

cpu\_frequency = cpu\_frequencies[i - 1] - 200000

cpu\_frequency = np.clip(cpu\_frequency, 1000000, 3000000)

cpu\_frequencies.append(cpu\_frequency)

actions.append(action)

cpu\_temperature = calculate\_cpu\_temperature(wind\_speeds[i//10], cpu\_frequency, cpu\_temperatures[i - 1])

cpu\_temperatures.append(cpu\_temperature)