A CONTEXT-AWARE AND ADAPTIVE SYSTEM TO AUTOMATE THE CONTROL OF THE AC WINDSHIELD USING AI AND INTERNET OF THINGS

Joshua Tian[1] and Yu Sun[2]

[1]Arnold O. Beckman High School, 3588 Bryan Ave, Irvine, CA 92602
[2]California State Polytechnic University, Pomona, CA, 91768, Irvine, CA 92620

ABSTRACT

In recent years, we have seen a huge increase in air conditioning usage [4]. However, much of this energy put into air conditioning is being wasted, which contributes to a far less environmentally friendly world and is inconvenient for many [5][6]. This paper develops a smart vent and a mobile app to regulate temperatures in different rooms of a home to create an efficient solution to save energy. This conservation of energy allows both the environment to be preserved as well as the financial burden of families in need to be alleviated. Controlled studies of the system provide evidence of the system's automated ability to be energy efficient.

KEYWORDS

AI, AC, Raspberry PI.

1. INTRODUCTION

Air Conditioners are a commonality throughout the world right now, with 87 percent of US homes equipped with some sort of cooling. However, with such a large amount of air conditioning every year comes a ridiculously high 100 million tons of carbon dioxide being released every year [10].

To address the issue of wasted energy, while there are many methods to harness more energy efficient forms of energy such as solar energy, these methods not only are extremely expensive, but also fairly inefficient, which makes using these alternative sources of energy impractical [7]. Although there are other practices to save energy without becoming more energy-efficient, these practices are purely based on the user's choice of using less energy and are therefore often deemed inconvenient. Without accountability, this inconvenience leads to many of these energy-conserving practices to be discontinued after a short period of time.

In this paper, we aim to create an energy efficient smart vent capable of being placed into an everyday home. This smart vent is controlled by a mobile app which allows the user to view the current temperature of the house as well as setting the target temperature of the room that the vent is in. By setting the app into heating or cooling mode, the smart vent will open or close to let the temperature of the house reach the user's target temperature. In addition, the user can manually open or close the vent by simply clicking a button on the app. Compared to alternative methods of conserving energy, the smart vent is a far more sophisticated solution, as it not only impacts the environment by conserving energy, but also lets the user adjust to different situations by using a mobile app [8]. Therefore, we believe that the smart vent is one of the best ways to help cut down wasted energy.

In multiple application scenarios, we demonstrate how the smart vent saves energy. Our goal is to compare the smart vent with non-smart vent energy levels. After finding the BTU for our AC system and how efficient it is, we pick a temperature that is common during the week, and then make note of this temperature as the starting temperature [9]. Next, we set the thermostat 5 degrees below the starting temperature. First, we use the smart vent with it being closed and then measure the time it takes for the thermostat to reach the target temperature. Then we use the same approach with the non-smart vent which is always opened. Finally, we have the target temperature in the room at a midway point so that the vent opens or closes in the middle. Then using the amount of time that the AC was running, along with the previous information, we calculate the amount of energy used. Through the final experiment, we can also see how after the smart vent closes halfway through the thermostat target temperature, the room cools or heats far quicker because it no longer needs to alter the temperature of that room. This is especially important, as it allows a house’s many different rooms to stay at different temperatures.

The rest of the paper is organized as follows: Section 2 gives the details on the challenges that we met during the experiment and designing the sample; Section 3 focuses on the details of our solutions corresponding to the challenges that we mentioned in Section 2; Section 4 presents the relevant details about the experiment we did, following by presenting the related work in Section 5. Finally, Section 6 gives the conclusion remarks, as well as pointing out the future work of this project.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Data reading and input

One of the most fundamental parts of this project is being able to read the temperature in the room and then making decisions off of it. Initially, in order to effectively read the temperature repeatedly over a long period of time, we decided to create a complex circuit system. This system is composed of a raspberry pi connected to a breadboard [11]. The breadboard would have all of the pins for the motor, and more importantly the pins for the digital temperature sensor. This temperature sensor would be able to constantly take in the temperature and then immediately have it be sent to the Raspberry Pi which would ultimately use that information for a multitude of things. However, this solution was far too complex and took up too much space to the point where we couldn’t fit it in the enclosed area in which we wanted to test it in. This becomes a problem because it ends up extremely difficult to test and apply to real world settings. To combat this, we have tried to use a shorter breadboard.

2.2. Data communication and storage

In order for the motor to be powered and the app to function, data communication and storage is essential [12]. In the cloud database Firebase, we store information such as whether the thermostat is heating or cooling, whether the vent is open or closed, the current and target temperature, and more. The Raspberry Pi also plays a huge role in this, as it both receives information from Firebase and sends information into Firebase. When the digital thermometer gets an updated temperature, the Raspberry Pi immediately transfers that data to Firebase and our backend application program likewise immediately takes that information to update the app. In addition, along with the temperature given by the thermometer, the Raspberry Pi also checks the many different conditions that we use to ensure the vent is opened and closed properly.

2.3. Updating the information on the app live

As an application that is centered around providing and changing information, a key part of that is to be able to update the information on the app live. We accomplished this by creating database reference variables which corresponded to a specific cell in the Firebase database. This allowed us to store all of our information in one place and be able to constantly update each variable by just editing the reference variable and uploading it to the database. Likewise, we were able to take information from the database and immediately reflect those changes on the app. In addition, using different kinds of functions.

3. SOLUTION

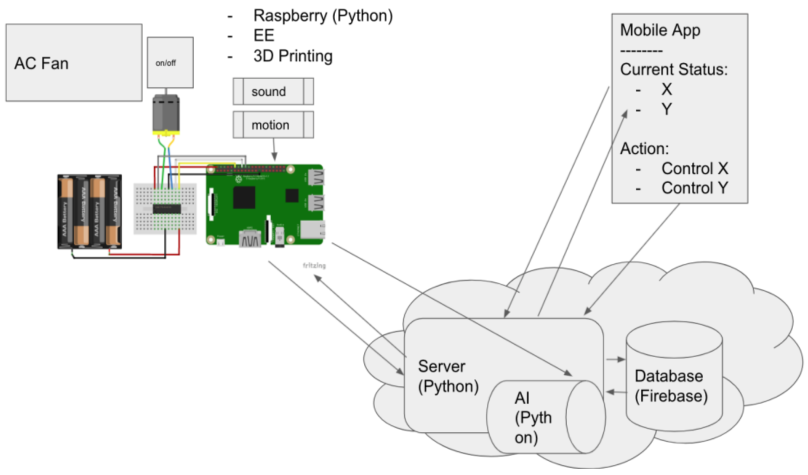
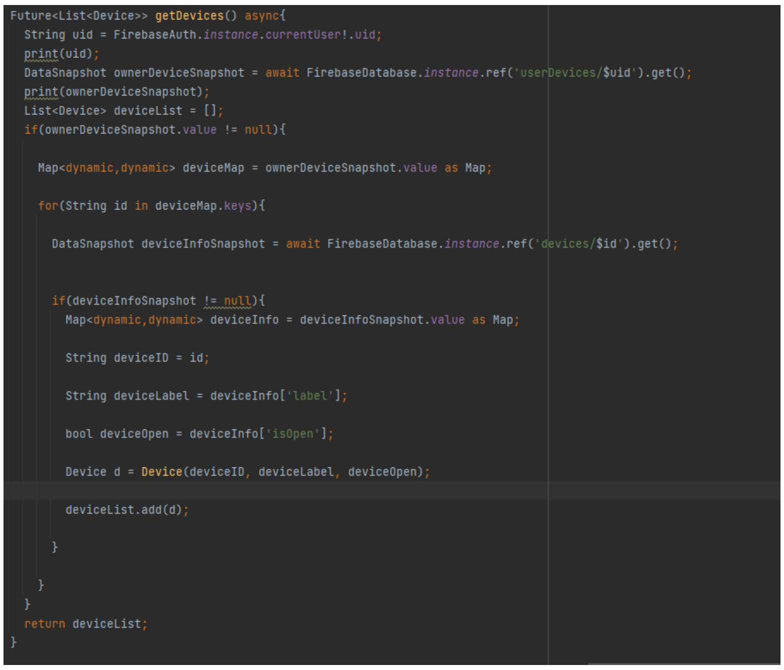


Figure 1. Overview of the solution

This solution is a smart vent system that allows the user to regulate the air flow in their home through altering different settings in a mobile app. The system has many components to ensure its proper function: The hardware and motorized parts, the raspberry pi and its code, the frontend application, and the backend cloud database. The application written in Dart in Flutter, can run on both iOS and Android devices, which allows for a diverse user base [13]. To start, we have modified a standard AC vent by connecting a 5V servo motor using linkage rods. The servo motor is attached to three of the pins on a breadboard which is circuited properly to supply the sufficient amount of power to the motor. In addition, a digital thermometer pin is also connected to the breadboard, allowing us to read the temperature of the surroundings. This breadboard is then naturally connected to the Raspberry Pi, which uses the built-in Python IDE to open or close the vent after reading the temperature. Subsequently, the Raspberry Pi will then upload the temperature reading to the Firebase database. The database stores many important pieces of information, such as the ID number of each vent, whether the thermostat system is cooling or heating, the current temperature, target temperature, and more. This information is not only used when determining whether the vent should be open or not, but is also the basis for the mobile app. The mobile app displays the current temperature and allows users to both manually control the vent or set it on auto mode. When the vent is in auto mode, the user will set a target temperature and the vent will open or close depending on whether the thermostat is heating or cooling, and then maintain the target temperature until a new one is set. This app is developed in the Android studio IDE using Google's open-source UI development kit Flutter. Flutter uses the object-oriented programming language, dart.

The tangible system of parts in this project is mainly composed of a 5V servo motor and a Raspberry Pi. The servo motor is fixed to the vent using a 3d printed motor mold that has M3 screw holes. The motor arm is connected to an M2 linkage rod which extends to screw into the arm of the vent. When the motor arm turns, it pulls or pushes the linkage rod which in turn pulls or pushes the vent arm to open or close the vent. The Raspberry Pi is attached to a breadboard which has the motor pins(GPIO18, 5V, and GND) connected. In addition, the digital thermometer used to intake the temperature readings is also connected to the breadboard.

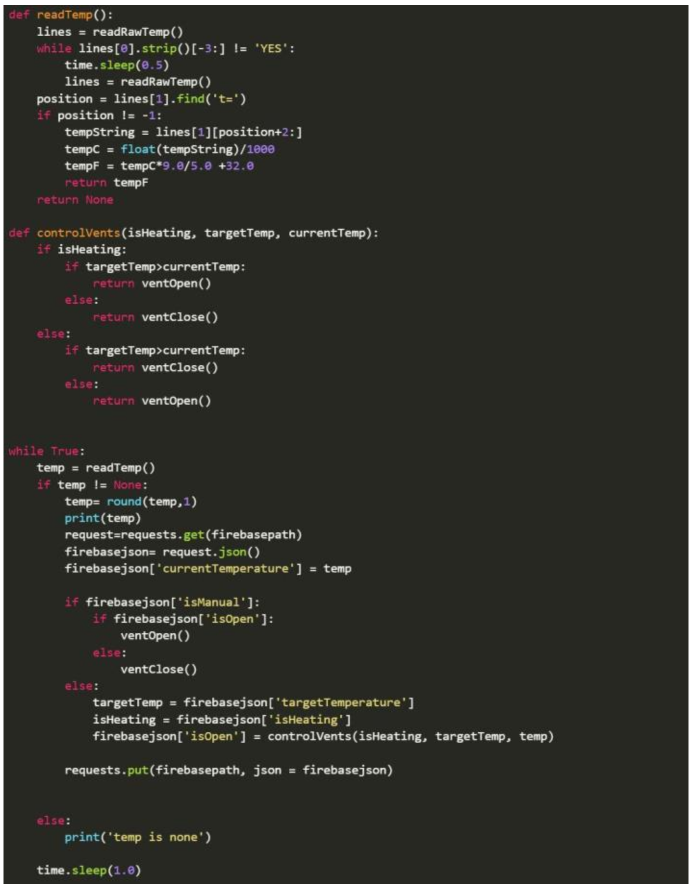
When first opening the app, and logging in to an account, you are brought to a page that lists all the devices that you have created. At the bottom there is also a button that allows you to create a new device by naming it and giving it a unique 6 digit device ID. Next, by selecting one of your devices you can see the current temperature of the room, the different settings, and the target temperature which you can increase or decrease using buttons on the screen.



```
Future<List<Device>> getDevices() async{
  String uid = FirebaseAuth.instance.currentUser!.uid;
  print(uid);
  DataSnapshot ownerDeviceSnapshot = await FirebaseDatabase.instance.ref('userDevices/$uid').get();
  print(ownerDeviceSnapshot);
  List<Device> deviceList = [];
  if(ownerDeviceSnapshot.value != null){

    Map<dynamic,dynamic> deviceMap = ownerDeviceSnapshot.value as Map;

    for(String id in deviceMap.keys){

      DataSnapshot deviceInfoSnapshot = await FirebaseDatabase.instance.ref('devices/$id').get();


      if(deviceInfoSnapshot != null){
        Map<dynamic,dynamic> deviceInfo = deviceInfoSnapshot.value as Map;

        String deviceID = id;

        String deviceLabel = deviceInfo['label'];

        bool deviceOpen = deviceInfo['isOpen'];

        Device d = Device(deviceID, deviceLabel, deviceOpen);

        deviceList.add(d);

      }

    }
  }
  return deviceList;
}
```

Figure 2. Screenshot of App code

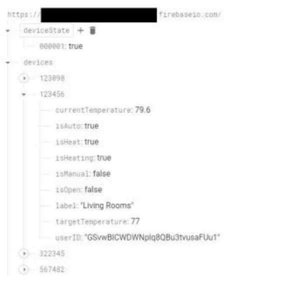
In this figure, we get all of the devices under a user by creating a snapshot and setting it as a reference to the user id in Firebase. We loop through all of the items in the deviceMap and for each item we store its device id, device label, and whether it is open or not. Next, using this information, we transfer those properties to a temporary device "d" and then add it to the device list which we ultimately end up returning. This device list is what you see on the homepage.



```
def readTemp():
    lines = readRawTemp()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.5)
        lines = readRawTemp()
    position = lines[1].find('t=')
    if position != -1:
        tempString = lines[1][position+2:]
        tempC = float(tempString)/1000
        tempF = tempC*9.0/5.0 +32.0
        return tempF
    return None

def controlVents(isHeating, targetTemp, currentTemp):
    if isHeating:
        if targetTemp>currentTemp:
            return ventOpen()
        else:
            return ventClose()
    else:
        if targetTemp>currentTemp:
            return ventClose()
        else:
            return ventOpen()


while True:
    temp = readTemp()
    if temp != None:
        temp= round(temp,1)
        print(temp)
        request=requests.get(firebasepath)
        firebasejson= request.json()
        firebasejson['currentTemperature'] = temp

        if firebasejson['isManual']:
            if firebasejson['isOpen']:
                ventOpen()
            else:
                ventClose()
        else:
            targetTemp = firebasejson['targetTemperature']
            isHeating = firebasejson['isHeating']
            firebasejson['isOpen'] = controlVents(isHeating, targetTemp, temp)

        requests.put(firebasepath, json = firebasejson)


    else:
        print('temp is none')

    time.sleep(1.0)
```

Figure 3. Screenshot of RPi Code

This figure shows the main parts of the Raspberry Pi code which is constantly running to check the temperature and adjust the motor position as needed. As shown in the while loop, we call the readTemp function at the beginning and using this temperature reading, we first update it into the database so that it can be used for the app. Next we check if the vent is being manually controlled, and if it is, then we won't change anything. However, if the vent is on auto mode, then we take the target temperature and the thermostat setting from the database and then call controlVents, which uses some boolean comparisons to determine whether the vent should open or close.



```
https://[redacted]firebaseio.com/
deviceState
    000001: true
devices
    123098
    123456
        currentTemperature: 79.6
        isAuto: true
        isHeat: true
        isHeating: true
        isManual: false
        isOpen: false
        label: "Living Rooms"
        targetTemperature: 77
        userID: "GSvwBICWDWNplq8QBu3tvusaFUu1"
    322345
    567482
```

Figure 4. Diagram of different parts of the database

These are the properties of one of my devices in Firebase. As shown, all of the necessary pieces of information that are shown on the app and used to open or close the vent are stored here. These variables are constantly updated both by the user in the app and by the Raspberry Pi when it gets a new temperature reading.

4. EXPERIMENT

4.1. Experiment 1

To evaluate the effectiveness of our solution, we compared the energy cost for cooling down the home using a normal vent and our smart vent. We created a situation in which the room that the smart vent was in didn't need to be cooled down, and as such, the smart vent closed by itself. In order to accurately determine the effectiveness of our solution, we ran this experiment multiple times in different rooms.

The results of the experiment show that when using a standard AC vent, the house cooled by 3 degrees in a span of roughly an hour. However, with the smart vent in this situation being closed, it took between roughly 54-58 minutes to cool the house by 3 degrees depending on which room the smart vent was in.

The experiment results shows that by using the smart vent, under the scenario where the specific room reaches its desired temperature and the smart vent closes, it lets the rest of the house heat or cool roughly 5% quicker. This decrease in time to change the temperature in the house means that the AC will be running for a shorter period of time which leads to lower energy cost.

5. RELATED WORK

K.J. Chua presented a solution that decreases the base energy output from a standard chiller plant [1]. This solution aims to save energy by decreasing the amount of energy that was needed to cool down a home normally, rather than trying to decrease energy waste. This is a unique approach that can be extremely impactful, but by ignoring the wasted energy, the potential of this solution is limited.

Adnan A.Kinsara, Moustafa M.Elsayed, and Omar M.Al-Rabghi created a solution that used Carbon Chloride as a liquid desiccant [2]. This liquid desiccant can hold water vapor which allows it to be able to overcome the latent part of the air conditioning load. This solution makes use of the characteristics of a typical air conditioning system and creates a chemical solution to make the system energy efficient.

P. Hengjinda and Dr. Joy Chen proposed a solution using an arduino microcontroller that monitors both the room and temperature outside a window [3]. Their intelligent controller continuously monitors the surrounding temperature and opens the window when the surrounding temperature equals the room temperature. This solution is very effective because it makes use of the natural environment to help speed up the air conditioning process.

6. CONCLUSIONS

In this project, we proposed an energy efficient air conditioning system to address the immense amount of wasted energy that air conditioning systems create. This system consists of a 5V servo motor attached to a standard AC vent's arm which is then attached to a Raspberry Pi. There is also a digital thermometer pin attached to the Raspberry Pi which supplies information to a backend database [14]. In addition, we have a mobile app where users can put certain settings into place. Using the temperature readings, as well as the target temperature and other factors determined by the user, the motor will open or close the vent. Experiments show that we save roughly 5% more energy by being able to close one vent in a room that doesn't need to be heated or cooled. By closing off a certain area of your home, it takes less time to cool the rest of the house which saves a lot of energy.

Currently the vent is a little bit inconvenient as there are many parts to the system which causes it to be extremely difficult to assemble. In addition, because it is battery reliant, it doesn't run for a very long time. There is also the issue of optimization with the thermometer readings, which could change depending on the location of the thermometer.

In the future, in order to find a more secure and optimal spot to place the Raspberry Pi and its thermometer, we will experiment with 3D printing as well as other methods to create a more stable system [15]. We will also try soldering the pins straight to the Raspberry Pi which will make it much more compact. In addition, I plan to add an A.I. based video recognition system that can recognize entry into the room and will open or close the vent based on user decision.

REFERENCES

[1] Chua, Kian Jon, et al. "Achieving better energy-efficient air conditioning–a review of technologies and strategies." Applied Energy 104 (2013): 87-104.

[2] Kinsara, Adnan A., Moustafa M. Elsayed, and Omar M. Al-Rabghi. "Proposed energy-efficient air-conditioning system using liquid desiccant." Applied Thermal Engineering 16.10 (1996): 791-806.

[3] Hengjinda, P., Dr Chen, and Joy Iong Zong. "An Intelligent Feedback Controller Design for Energy Efficient Air Conditioning System." Journal of Electronics and Informatics 2.3 (2020): 168-174.

[4] Ren, Xiaoxin, Da Yan, and Chuang Wang. "Air-conditioning usage conditional probability model for residential buildings." Building and Environment 81 (2014): 172-182.

[5] Clark, James H., and Duncan J. Macquarrie. "Environmentally friendly catalytic methods." Chemical Society Reviews 25.5 (1996): 303-310.

[6] Traore, Moussa K., and Gisela Buschle-Diller. "Environmentally friendly scouring processes." Textile Chemist & Colorist & American Dyestuff Reporter 32.12 (2000).

[7] Cuéllar, Amanda D., and Michael E. Webber. "Wasted food, wasted energy: the embedded energy in food waste in the United States." Environmental science & technology 44.16 (2010): 6464-6469.

[8] Neumerkel, René, and Stephan Groß. "A sophisticated solution for revealing attacks on wireless LAN." International Conference on Trust, Privacy and Security in Digital Business. Springer, Berlin, Heidelberg, 2006.

[9] Kakigano, H., M. Nomura, and T. Ise. "Loss evaluation of DC distribution for residential houses compared with AC system." The 2010 International Power Electronics Conference-ECCE ASIA-. IEEE, 2010.

[10] Buchanan, Andrew H., and Brian G. Honey. "Energy and carbon dioxide implications of building construction." Energy and Buildings 20.3 (1994): 205-217.

[11] Zhao, Cheah Wai, Jayanand Jegatheesan, and Son Chee Loon. "Exploring iot application using raspberry pi." International Journal of Computer Networks and Applications 2.1 (2015): 27-34.

[12] Prabhu, Narahari Umanath. Stochastic Storage Processes: queues, insurance risk, and dams, and data communication. No. 15. Springer Science & Business Media, 1998.

[13] Tracy, Kim W. "Mobile application development experiences on Apple's iOS and Android OS." Ieee Potentials 31.4 (2012): 30-34.

[14] Hadley, I. C. D., and R. D. Gould. "Inexpensive digital thermometer for measurements on semiconductors." International Journal of Electronics Theoretical and Experimental 70.6 (1991): 1155-1162.

[15] Shahrubudin, Nurhalida, Te Chuan Lee, and Rhaizan Ramlan. "An overview on 3D printing technology: Technological, materials, and applications." Procedia Manufacturing 35 (2019): 1286-1296.