Introducing the Internet to the Weathervane

First A. Smriti Rai, Second B. Nand Patel, and Third C. Oscar Salinas

Abstract-**Background: The weathervane is an uncomplicated tool that measures the direction of how the wind is blowing. This tool has been built in various manners, but each serves the same purpose. In its most recent form, the weathervane has a tail, an arrow, and a simple compass made from two sticks that are labeled in the cardinal directions. The tool works best when placed at a higher altitude because it can catch the lightest winds. The tool catches wind through the tail and will pivot towards a direction which is indicated by where the arrow is pointing. Today, the weathervane is not as heavily used due to other technological developments that have been generated throughout the years. Most of the usage of the weathervane is purely for aesthetical reasoning which is why the authors of this journal have decided to implement the usage of technology into this simple tool.**

Architecture of the Project: The organization of this project starts with a Raspberry Pi 4 which is the brain of the weathervane. The authors have three sensors that will replicate the components of the weathervane that directly communicates with the Raspberry Pi. The Raspberry Pi will take the information it is receiving from the sensors and send the data it is receiving to a cloud database (AWS).

Software Components: In this project, the authors primarily used Python as the coding language and VSCodes as the IDE.

Hardware Components: There was a total of five different hardware components that were used in this project: Raspberry Pi 4, a Barometric Pressure Sensor (BME280), an Anemometer, a Samsung Galaxy S23 Ultra, and a Raspberry Pi Camera (RPI-CAM-V2).

Results: By combining the hardware/software components and a facial recognition process for the data to be secured to a certain user, the authors of this journal were able to bring the internet to a simple tool that has existed since 139 BC.

Index Terms— **OpenCV, Amazon Web Services (AWS), Python**

I. Background

THE history of the weathervane can be tracked dating back to 50 BCE and has remained as a simple tool since its existence*[3]*. The origin began in Athens by a Greek architect named Andronicus of Cyrrhus, who based the structure of the initial prototype on the God of the Sea, Triton. Not only did this tool have material to track wind, but the Greek architect also added the components of a sundial and water clock. The use of this tool was to be able to dictate what the weather would be for the day.

The development of the weathervane in Greece moved to the Asian region where the Chinese culture appended their own style to the weathervane. They ended up changing the initial shape which was replicated after Triton to animals that derive from the 12 animals that measure cycles of time (Chinese New Year animals) *[1]*. The shape then became birds and dragons which they started to mount on top of their huts to only track wind direction.

This instrument then became an essential tool for sailing which was adopted by Vikings. The idea of what was floating around from the Greeks and Chinese was altered by the Vikings to meet their own needs. They used cloth and metal strips at the mast of their ship to help the sailors understand the direction of the wind. As this caught on in the Scandinavian region, they removed the bird and dragon and began using creatures from the Norse fables (Viking Mythology) *[4]*.

The design transitioned to the European region which had a heavy existence during the medieval times. The structure slightly varied here once they decided to simplify the existing creation to simple arrows. They also made modifications to the top by changing it into a rooster, something that is very popular in the 21st century *[1]*. The symbolism for the European weathervane was recognized as a sign of daybreak, the triumph of light over dark since the change that was adopted was heavily religious based. The name emanated from the European era where they started by calling it a 'weather cock' in honor of the rooster they placed at the top of the weathervane about the Bible. 'Weather cock' then transitioned into 'vane' which comes from an Angle-Saxon word 'fane,' which meant flag *[1]*.

Through the centuries, the spread of the weathervane became prominent, and most regions took this instrument and made it their own. With this background given, the weathervane shows functional and symbolic purposes just by looking at the symbol that is at the top of the weathervane.

II. Architecture of the Project

The organization of this project begins with the CPU which is the Raspberry Pi 4. The Raspberry Pi 4 is the central brain that inputs and outputs data between the sensors and the cloud database (AWS). The two environmental sensors (BME280 and Anemometer) are connected to the Raspberry Pi through GPIO which is their primary way of communicating. The sensors are actively measuring wind speeds, temperature, humidity, pressure, and altitude. As this data is being collected by the sensors, the Raspberry Pi will read and output the data into a . CSV file. Once the user completes the user authentication through the camera sensor/facial recognition, the Raspberry Pi will send the two environmental sensors' data which has been compiled into a . CSV file to AWS.

III. Software components

This project revolves around three different software components: a Cloud database, and two IDEs.

A. Amazon Web Services (AWS)

- Amazon Web Services (AWS) is a cloud-based service that is used for storage and data processing. In the case of this project, the user can use this application to store the user who is using the weather station as well as a security measure to see who has been accessing their data.

B. OpenCV

- OpenCV is an open source that incorporates machine learning software. This software allows a plethora of opportunities for users to process images, image transformation, color space conversions, and filtering. This tool is also convenient for those who want to incorporate a camera because there are libraries this software includes that have precise camera calibration. This is a cross-platform application that we were able to integrate between our Linux devices which helped run the facial recognition process smoothly.

C. VSCode

- VScode is a Microsoft-based open-source software that can be used for a various number of different coding languages. This is available on multiple different operating systems but the author's project, Windows OS was applied.

D. Python

- Python is a widely used coding language that has existed since 1991. The appeal to this language is its versatility as well as how simple its readability is. Python is the only coding language that was used for the entire project.

IV. Hardware Components

This project revolves around five different hardware components: Three sensors, a CPU, and a mobile interface.

A. Raspberry Pi 4

- This component acts like a central processing unit (CPU) that is used to collect data from the sensors and acts as a communication device between the sensors and the cloud. This device outputs through a Linux operating system which is where the data is being produced. To connect the components to the CPU, there are wires connected to GPIO pins. The three jobs of this device are to: interact with the sensors to receive data, take the images from the camera for facial recognition, and send data to AWS.

B. BME280

- BME280 is known as the barometric pressure sensor that collects data from the environment. This sensor can receive temperature in Fahrenheit (F), humidity in percentage (%), pressure in Inches of Mercury (Hg), and altitude in meters (m). The component elevates the weathervane by adding not only the wind speed but four other measurements that can aid the user in determining further weather information.

C. Anemometer

- The Anemometer is a device that measures wind speed and direction. This tool is mainly used by meteorologists to help them identify weather patterns. This sensor will output data by using the number of rotations to calculate wind speed (m/s).

D. Samsung Galaxy S23 Ultra

- The Samsung Galaxy S23 Ultra is an Android device that was used to provide a network to the Raspberry Pi 4 through a hotspot. Without this hardware component, the Raspberry Pi would not be able to display real-time data or allow interactions with the user.

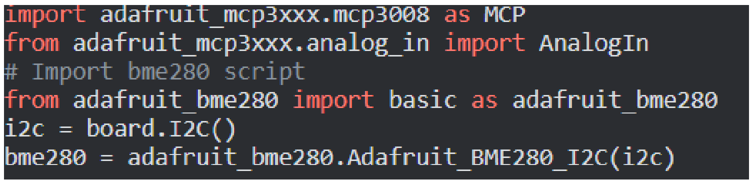
E. Raspberry Pi Camera (RPI-CAM-V2)

- The RPI-CAM-V2 is a simple camera that is used alongside the OpenCV code which operates the facial recognition of the project. This sensor's main role is to capture real-time footage for the user's security purposes.

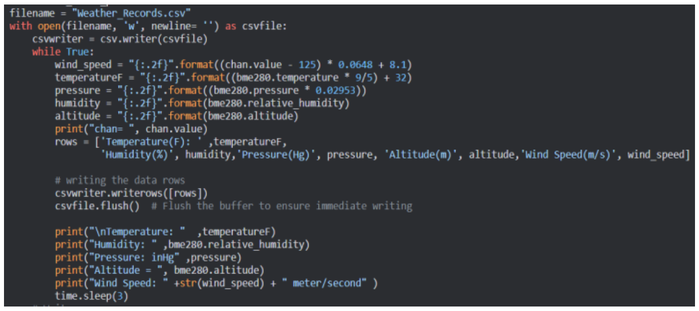
V. Results

This project can accurately record and upload data at regular intervals from a weathervane. The authors took multiple sensors to take accurate measurements and upload them to the cloud to be able to access the data remotely. The project will help deepen the knowledge of different sensors, Raspberry Pi, and use the internet to upload and access all the data that is being gathered by the various sensors.

To implement the internet to a simple tool like the weathervane begins with Raspberry Pi 4. This CPU was the center of communication between our three sensors, our cloud database, and the Python code that is used to input/output the data. The Raspberry Pi pulls data from the BME280 which tracks the humidity, temperature, altitude, and pressure as well as data from the Anemometer which is calculating the wind speeds based on rotation. To keep the communication between the sensors and CPU, the authors used 'adafruit' libraries to receive accurate data.


```
import adafruit_mcp3xxx.mcp3008 as MCP
from adafruit_mcp3xxx.analog_in import AnalogIn
# Import bme280 script
from adafruit_bme280 import basic as adafruit_bme280
i2c = board.I2C()
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)
```

While the Raspberry Pi is collecting the data from the two environmental sensors, the author's input code to make sure that the data that was being received from the sensors is placed into. CSV files. The. CSV was created by taking all the information that was being collected from the sensors and creating rows of information that would be refreshed every 2 seconds. The authors initially placed all the information in the terminal but had to find a solution because the communication between AWS and Python data needed to be through a . CSV file.

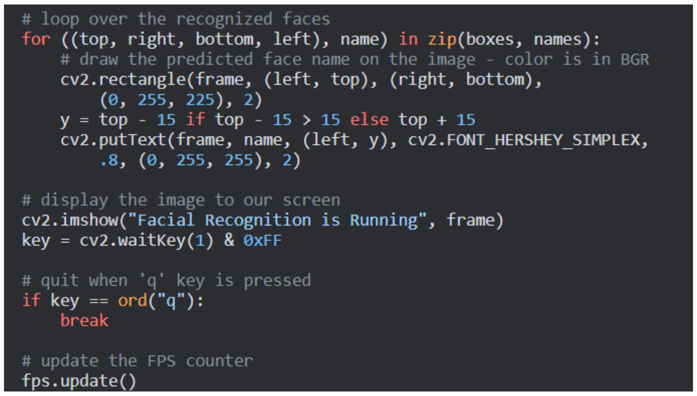


```
filename = "Weather_Records.csv"
with open(filename, 'w', newline= '') as csvfile:
    csvwriter = csv.writer(csvfile)
    while True:
        wind_speed = "{:.2f}".format((chan.value - 125) * 0.0648 + 8.1)
        temperatureF = "{:.2f}".format((bme280.temperature * 9/5) + 32)
        pressure = "{:.2f}".format((bme280.pressure * 0.02953))
        humidity = "{:.2f}".format(bme280.relative_humidity)
        altitude = "{:.2f}".format(bme280.altitude)
        print("chan= ", chan.value)
        rows = ['Temperature(F): ' ,temperatureF,
                'Humidity(%)', humidity,'Pressure(Hg)', pressure, 'Altitude(m)', altitude,'Wind Speed(m/s)', wind_speed]

        # writing the data rows
        csvwriter.writerows([rows])
        csvfile.flush()  # Flush the buffer to ensure immediate writing

        print("\nTemperature: "  ,temperatureF)
        print("Humidity: " ,bme280.relative_humidity)
        print("Pressure: inHg" ,pressure)
        print("Altitude = ", bme280.altitude)
        print("Wind Speed: " +str(wind_speed) + " meter/second" )
        time.sleep(3)
```

After completing the collecting data from the sensor portion, the authors begin to focus on adding security to the project. The authors used OpenCV to train the Raspberry Pi 4 to be able to analyze an array of images to distinguish which face belongs to the dataset retrieved by the weathervane. The two major components are a terminal and a camera, the rest will continue through OpenCV. Since this is just a prototype, the camera is held by whoever wants to access the data. The user will hold the camera to their face and the OpenCV will pull from the code as displayed below.



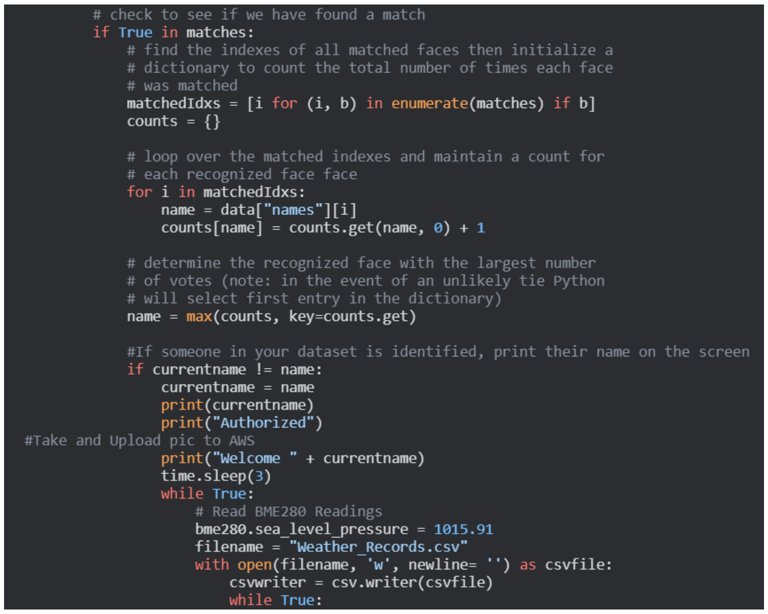
```
# loop over the recognized faces
for ((top, right, bottom, left), name) in zip(boxes, names):
    # draw the predicted face name on the image - color is in BGR
    cv2.rectangle(frame, (left, top), (right, bottom),
        (0, 255, 225), 2)
    y = top - 15 if top - 15 > 15 else top + 15
    cv2.putText(frame, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,
        .8, (0, 255, 255), 2)

# display the image to our screen
cv2.imshow("Facial Recognition is Running", frame)
key = cv2.waitKey(1) & 0xFF

# quit when 'q' key is pressed
if key == ord("q"):
    break

# update the FPS counter
fps.update()
```

The code above shows placing a square around the faces that are detected. After the face is detected in the frame measurements that are established, the name of any of the users will appear in the database or an unknown if it is not able to match the face. After the facial recognition can confirm that the user’s face matches a face in the database, a welcome ‘name of user’ will appear, and then a . CSV file will be uploaded to AWS.



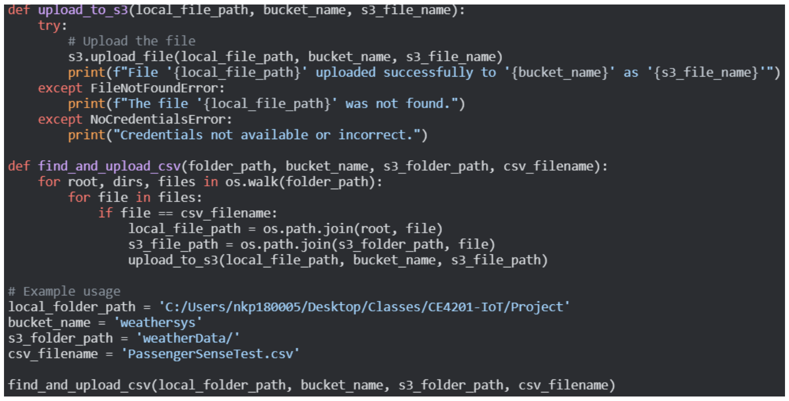
```
        # check to see if we have found a match
        if True in matches:
            # find the indexes of all matched faces then initialize a
            # dictionary to count the total number of times each face
            # was matched
            matchedIdxs = [i for (i, b) in enumerate(matches) if b]
            counts = {}

            # loop over the matched indexes and maintain a count for
            # each recognized face face
            for i in matchedIdxs:
                name = data["names"][i]
                counts[name] = counts.get(name, 0) + 1

            # determine the recognized face with the largest number
            # of votes (note: in the event of an unlikely tie Python
            # will select first entry in the dictionary)
            name = max(counts, key=counts.get)

            #If someone in your dataset is identified, print their name on the screen
            if currentname != name:
                currentname = name
                print(currentname)
                print("Authorized")
#Take and Upload pic to AWS
                print("Welcome " + currentname)
                time.sleep(3)
                while True:
                    # Read BME280 Readings
                    bme280.sea_level_pressure = 1015.91
                    filename = "Weather_Records.csv"
                    with open(filename, 'w', newline= '') as csvfile:
                        csvwriter = csv.writer(csvfile)
                        while True:
```

To communicate with AWS, we created another. PY file which allows the upload of the data to be added to a. CSV file.



```
def upload_to_s3(local_file_path, bucket_name, s3_file_name):
    try:
        # Upload the file
        s3.upload_file(local_file_path, bucket_name, s3_file_name)
        print(f"File '{local_file_path}' uploaded successfully to '{bucket_name}' as '{s3_file_name}'")
    except FileNotFoundError:
        print(f"The file '{local_file_path}' was not found.")
    except NoCredentialsError:
        print("Credentials not available or incorrect.")

def find_and_upload_csv(folder_path, bucket_name, s3_folder_path, csv_filename):
    for root, dirs, files in os.walk(folder_path):
        for file in files:
            if file == csv_filename:
                local_file_path = os.path.join(root, file)
                s3_file_path = os.path.join(s3_folder_path, file)
                upload_to_s3(local_file_path, bucket_name, s3_file_path)

# Example usage
local_folder_path = 'C:/Users/nkp180005/Desktop/Classes/CE4201-IoT/Project'
bucket_name = 'weathersys'
s3_folder_path = 'weatherData/'
csv_filename = 'PassengerSenseTest.csv'

find_and_upload_csv(local_folder_path, bucket_name, s3_folder_path, csv_filename)
```

To summarize the results of this project, we were able to successfully create a weather station that could aid in many aspects of the modern world.

VI. Abbreviations

Amazon Web Service (AWS), Barometric Pressure Sensor (BME280), Raspberry Pi Camera Module V2-8 Megapixel/1080p (RPI-CAM-V2), Comma Separated Values (.CSV), Python File (.PY)

Acknowledgment

All authors would like to thank Professor Lakshman Tamil for providing this opportunity for us.

References

A. References

- *[1] "Weathervane history highlights," Ferro Weathervanes, https://ferroweathervanes.com/weathervane-history-highlights/ (accessed Dec. 15, 2023).*
- *[2] "Weather Vane," Wikipedia, https://en.wikipedia.org/wiki/Weather_vane/ (accessed Dec. 15, 2023).*
- *[3] "The Story Behind Weathervanes," Copper Ridge Cupolas, https://www.copperridgecupolas.com/articles/the-story-behind-weathervanes/ (accessed Dec. 15, 2023).*
- *[4] "The History of Weathervanes," Valley Forge Cupolas and Weathervanes 866-400-1776, https://www.valleyforgecupolas.com/blog/the-history-of-weathervanes (accessed Dec. 15, 2023).*
-