Imperial College London

IBM

ELECTRICAL AND ELECTRONIC ENGINEERING

3rd YEAR CONSULTANCY GROUP PROJECT

DOCUMENTATION - AI WEATHER ART BOT

Student Names:

Lee Choi
Gonzalo Fraile
Syafiqah Mohamed Saifulaman
Leonidas Tsigkounakis
Emma Wardle

Academic Supervisor:

Prof. Bikash Pal

Industrial Supervisor:

Prof. John McNamara

Contents

1 Design history: from concept to final product

1.1 Project brief

In collaboration with IBM, our group was tasked to build an AI Weather Bot called KAIROS that helps the elderly access and enjoy digital services. It is common for the elderly to feel hesitant towards adopting new technology, considering it unfamiliar and strange and as a result they may, unfortunately, overlook the numerous quality of life improvements that new technology, such as AI, can offer. To bridge this gap, we created an interactive piece of art that is simple to use and would inspire users to engage with it, emphasizing its artistic qualities rather than its technological nature. Our Weather Bot offers real-time weather information by appropriately rotating four scene discs to match the current weather description and by displaying relevant weather data such as temperature, precipitation and wind speed on a display as well. The other defining feature of our product involves the integration of IBM's Watson Assistant to act as a personal assistant that processes voice commands from the user such as asking for the weather forecast or even for the latest news. This enables a conversational interface that is interactive and adds personality to our product.

1.2 MoSCoW priority list

After consulting with IBM, it was agreed upon that implementing a MoSCow priority list would be essential before embarking on building KAIROS. A MoSCow priority list is required before beginning any project since it ensures a planned and methodical approach to project management. The MoSCoW technique, which stands for Must, Should, Could, and Won't-have, can be used to prioritize project requirements and outputs given a limited time frame which in our case was 8 weeks. The Must-have criteria are the essential functionalities and features required for the project's success. To guarantee that the project fulfills its key objectives, they must be prioritized above all else. Should-have requirements are significant but not critical, and they can be postponed if required. Could-have needs are desirable but not required, and they can be explored if time and resources allow. Finally, the Won't-have needs are purposefully left out of the project's scope. The finalised MoSCow list for our project is shown below. The list was shared with IBM who then granted approval to initiate the project.

Must have:

- Mechanical moving art piece
- WiFi or Cellular connectivity
- Access to weather API weather data
- Display for important weather information
- Access to IBM's Watson Assistant
- Text-to-Speech (TTS) and Speech-to-Text (STT)

Should have:

- User-friendly interface and abstraction
- On/Off switch
- Voice volume control

Could have:

- News updates
- Plays podcasts and music from Spotify
- Reminders (medications etc), timers and alarms
- Remote control for people with limited mobility
- Voice-enabled emergency calls (activated by a key word)
- Different voice options

Won't have:

- An app associated with it
- Overwhelming complexity
- Speaking without being prompted
- Battery power

1.3 Building KAIROS

1.3.1 Building the Weather Bot mechanism

At the project's inception, IBM specifically directed us to follow the Weather Bot instructables tutorial page by DIY Machines [1] for constructing the chasis/framework and mechanism.

The building process of the Weather Bot begins by 3D printing the main base and all the other necessary components. The base consists of two parts, the front and the rear section (fig. 1a, fig. 1b and fig. 2). The front section hosts the scene discs and is the side in which the user can see the current weather forecast illustrated by the four discs. The rear section houses the servo motors the gears, and is screwed onto the main base for robustness. Apart from the base, the Weather Bot consists of four 3D-printed scene discs that create a diorama illustrating the weather forecast, four large cogs directly attached to each scene disc and four small cogs that fit into the servo motors which allow rotation of the discs. There are also some other 3D-printed components that provide further support between pieces.

In terms of electronic components, the Weather Bot is powered by a single USB cable, making it easy to place anywhere. The power cable powers an ESP32 microcontroler which is programmed to configure and then rotate each scene disc according to the weather data from OpenWeather's API [2]. We ultimately opted for the OpenWeather API due to its highly accurate and hourly updated information. Additionally, the ESP32 is configured to display temperature, precipitation, and windspeed data for both the current weather and the forecast for the next day on a 2.7-inch e-ink display. More details about the materials sourced can be found at the "Bill of Materials (BOM)" section on page 10 (BOM). There are also four continuous rotation servo motors attached to the rear base for each disc. Finally there are four limit switches which are activated by dents on the scene discs to estimate the initial angle of each scene disc in order for the configuration/initialisation stage to take place.

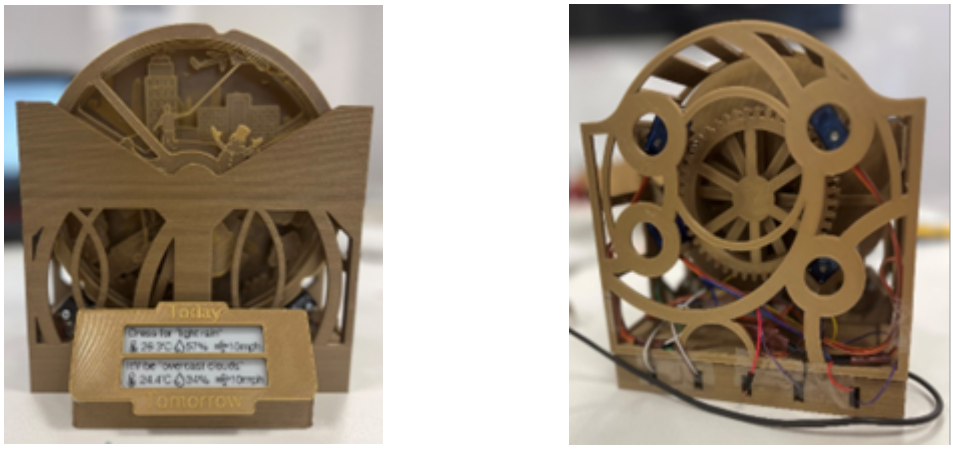
Each of the four scene discs represents a specific weather condition. All discs are divided into three parts that separate their corresponding weather condition into three subsections.

The first disc, which is closest to the display, represents the current temperature outside. It displays a snowman if the temperature is below a pre-determined threshold which is defined in the ESP32 code on the `WeatherBot.ino` file that can be found on our GitHub repository on page 9 (GitHub). If the temperature is higher than that threshold but lower than another pre-defined upper bound it displays a tree and if the temperature is even higher it displays an ice-cream shop.

In a similar way, the second disc represents the current wind speed. The thresholds for wind speed are also defined in the `WeatherBot.ino` ESP32 code file and represent low, medium and high wind speeds. These are depicted artistically by a person walking their dog, a kid flying their kite and by a hurricane respectively.

The third disc displays the amount of cloud cover in the sky. It is divided into sections indicating clear skies, partly cloudy conditions, or overcast skies. The art of the disc reflects the current cloud conditions by displaying a plane flying in the air with no clouds, a city with a few clouds on top and by a city with many clouds above respectively.

The fourth and final disc is the precipitation disc and represents the probability of precipitation (rain or snow). The sections on that disc relate to no rain, light rain and heavy rain. These weather conditions are artistically represented by a flock of birds flying above a city, by a few rain droplets and lastly by many rain droplets respectively.



(a) Front view **(b)** Rear view

Figure 1: Front and rear view of the Weather Bot

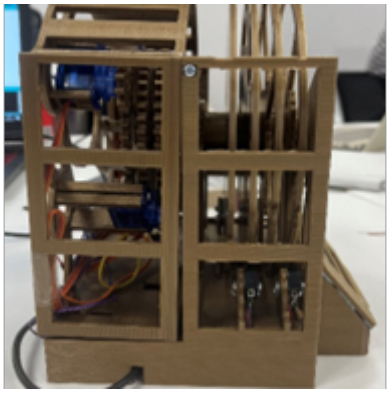


Figure 2: Side view of the Weather Bot

Upon powering the ESP32 and given that the `WeatherBot.ino` code has been uploaded beforehand, it is assumed that the orientation of the scene discs is unknown. Thus, rotation of each disc initiates, leading to the activation of their respective switches due to dents on their sides as mentioned before. The presence of the dents allows a one-to-one mapping of the spacing between each dent to the orientation of the discs. As the discs rotate, the switches accurately encode and relay information about their orientation. After the configuration stage is over, the ESP32 connects

to WiFi and then fetches data from the OpenWeather API. Subsequently it commands the servo motors to rotate each of the discs appropriately to visually display current weather readings in a more intuitive and engaging manner. Users can quickly grasp information about temperature, precipitation probability, wind speed, and cloud cover by observing the positions of the respective discs.

1.3.2 Integrating IBM's Watson Assistant

IBM's Watson Assistant is integrated into the project using the Python programming language. In order to process the vocal commands from the user we used Watson's Speech-to-Text (STT), Assistant, and Text-to-Speech (TTS) services. It is important to note that we only had access to the Lite (free) version of IBM's Watson Assitant which ultimately restricted our ability to fully leverage its capabilities for our project. What follows refers to the Python code found on our GitHub that was used to successfully integrate IBM's Watson Assistant to our Weather Bot.

The `api.py` Python file contains the API keys and URLs for the Watson Assistant. However, this file is not visible in the repositories as it is private and specific to our project. These API keys can be created by anyone free of cost. Once created, these variables can be accessed from the `watson.py` file.

The `watson.py` Python file contains all the code related to Watson-specific features. The `authenticate_stt()`, `authenticate_assistant()`, and `authenticate_tts()` functions create new clients with access to the corresponding services. Since these functions return a client adapter, they must be assigned to a client variable as soon as they are called. The `get_transcript()` function uses the Watson STT service. It takes a .WAV audio file as input and sends the audio data to IBM's Watson Cloud services. All data processing is in fact done in the IBM Cloud service which returns a string transcript of the input. The `create_session()` function creates a session with the Watson Assistant, similar to "opening a conversation with an assistant." This session must be created only once when KAIROS is first activated. Additionally, the responses are all programmed on the Watson website, which is documented later. The `message()` function sends the user's question or request to IBM's Watson assistant. Once the message is successfully sent, IBM's Watson Assistant provides a response to the user in JSON format. The response can be easily extracted by navigating through the response to the `text` key. The `synthesise()` function uses IBM's Watson TTS service. It takes Watson Assistant's response from the previous function as input, sends the string to the Watson Cloud to convert it to a WAV audio file, and saves this audio file as `response.wav` in the `sample` directory mentioned earlier.

Thirdly, the `sound.py` file contains all the Python code related to the voice commands features of KAIROS. There are only two functions programmed in this file: `play()` and `record()`. The `pyaudio` library was used for this. The `play()` function plays an audio file to the default output peripheral of the device that calls the function, and the `record()` function records sound from the default input peripheral that calls the function. We encountered technical problems with this part of the project because our laptop specifications differ greatly from those of a Raspberry-Pi. The code will not run if the Raspberry-Pi ha no recognizable input or output device. Therefore, the tests for the `sound.py` file were delayed due to the unavailability of input and output devices for the Raspberry-Pi in the early stages of the project. Additionally, the Raspberry-Pi reads and writes more samples per data frame. While PCs work with 1024 samples per data frame, the Raspberry-Pi encounters an error with this chunk size and must be increased to 4192 samples per data frame.

To integrate the "News" feature from our Could-have list, the `bbc.py` file was created to crawl the top three most-read news articles from the BBC news website. Ideally, this part of requesting data should have been integrated into the IBM's Watson Assistant. However, due to API unavailability, we programmed this functionality on the Raspberry-Pi itself. We used the `requests` library to crawl the news website and passed it to the `BeautifulSoup` library's HTML parser to extract the top three most-read news articles.

This data is then processed in the Raspberry-Pi which runs the `main.py` Python file to function as the assistant bot. It also includes some data processing to determine Watson's response due to the limitations of Watson's Lite version.

2 Testing and video recordings

The final product satisfies all of the Must-have and Should-have requirements from our initial MoSCoW list. Also, we managed to integrate the "news updates" feature from the Could-have list. The product works effectively and has been tested several times as shown in the test videos section below.

2.1 Testing KAIROS

Throughout the project, we conducted several tests to verify the independent functionality of each component before integration.

Regarding the ESP32, we divided the testing process into three parts: motor function, screen display, and API connection. Initially, we hard-coded weather data into the relevant sections on the ESP32 code in order to evaluate motor function, ensuring that the main axle rotated accurately and correctly and that the algorithm determined each disc's initial orientation. Next, we tested the screen by verifying that the simulated data was correctly displayed on the e-ink display screen. Lastly, we incorporated various print statements into our ESP32 code to validate the ESP32's successful connection to the OpenWeather API and to confirm the receipt of accurate information.

We then combined all three aspects, ensuring that the art pieces moved to the appropriate orientation and that the correct weather information were displayed on the screen based on the data retrieved from the weather API. Finally, we attached the microphone onto the main base of the Weather Bot which connected to the Raspberry-Pi that was in turn used to capture the voice commands that were then fed into IBM's Watson Assistant for processing. The attached videos below demonstrate the accurate functioning of the Weather Bot and the succesful intergation of IBM's Watson Assistant respectively.

2.2 Video recordings

Video recordings of the mechanical art scene discs rotating correctly and of successful connection to the OpenWeather API:

[1] - https://clipchamp.com/watch/cRbhtcgXlEn

[2] - https://clipchamp.com/watch/NHTLcMe30Rd

Video recording of IBM's Watson Assistant answering correctly when asked what today's weather is:

[3] - https://clipchamp.com/watch/EbOq1RtXJ9f

3 Time management and meeting records

3.1 Gantt chart

Gantt charts provide a visual representation of a project's timeline and key milestones. They predominantly illustrate the sequence of tasks and their respective duration. Below is the Gantt chart for the AI Weather Bot project.

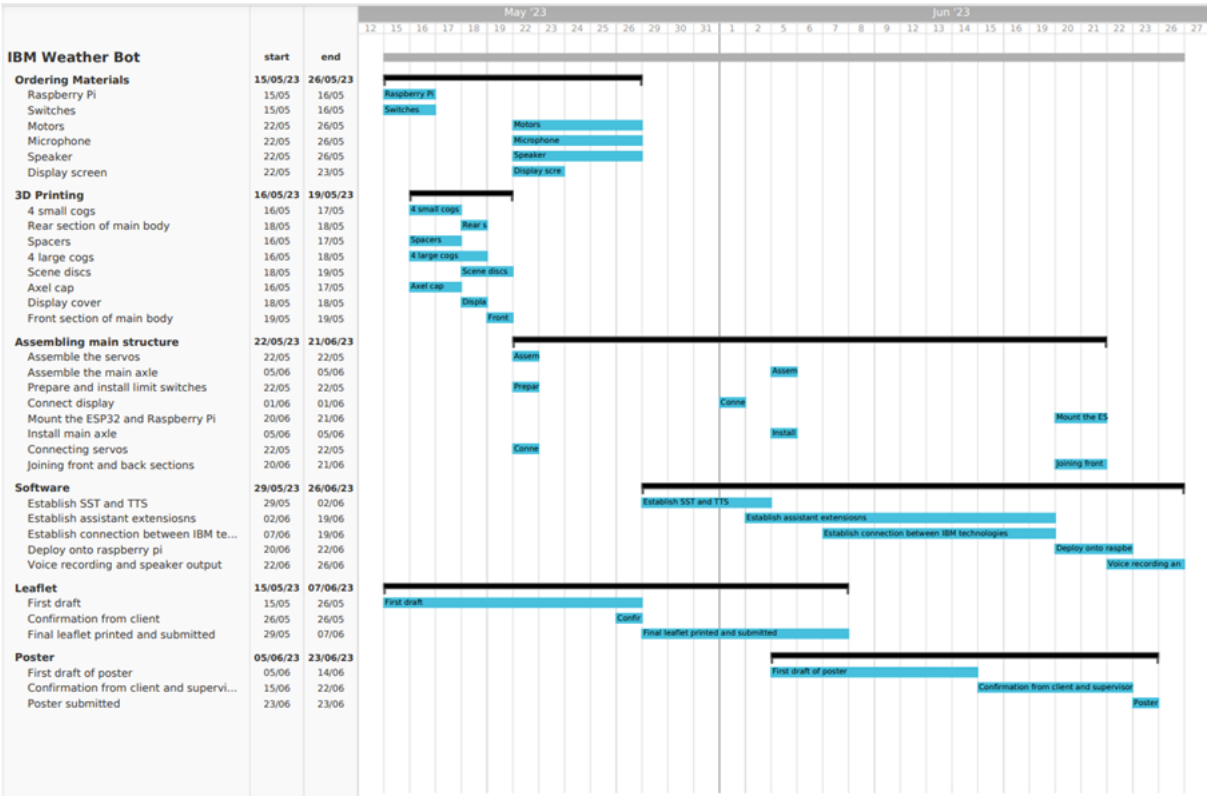


Figure 3: Gantt chart for the AI Weather Bot project timeline

3.2 IBM meetings record

The table below organises the main topics discussed in each meeting with our industrial supervisor from IBM, prof. John McNamara.

Date	Meeting description
02/05/23	Introductory meeting Discussed project outline Client requested the creation of a MoSCoW list Client requested we look through IBM SkillsBuild and review relevant topics
11/05/23	Informed client of progress including commencement of 3D printing and of immediate next steps (looking over Watson to see how it integrates to the Weather Bot)
12/05/23	Gained client approval of our MoSCoW list to begin the project Communicated our planned actions to initiate the project to the client Client granted his consent to our proposed strategy and we started using Watson's Text To Speech (TTS) First draft of leaflet
24/05/23	Sent client update of project progress Completion of 3D printing, assembling of the product, struggles with finding a Raspberry-Pi due to shortage
26/05/23	Sent client a copy of our draft leaflet, which he approved
02/06/23	Attended IBM's studio tour at their London office Met prof. John McNamara in person at their innovation studio
21/06/23	Poster finalised Sent client a copy of the finalised poster Received confirmation of poster, with no added notes or changes

3.3 Group meetings record

The table below displays the meeting contents and discussions with our group.

Date	Meeting description
01/05/23	Discussed IBM badges that client had asked us to complete
03/05/23	Created MoSCoW list Each created an IBM cloud account at the request of the client Created excel spreadsheet of all the sections on IBM Skills Build that were relevant to the project
04/05/23	Group meeting with EEE about project management and team work
10/05/23	Meeting with EEE supervisor Explained project and received access to 3D printers
15/05/23	Meeting to discuss what hardware was required and place orders for materials
16/05/23	Started leaflet design and initial draft
19/05/23	Started to assemble product
22/05/23	Issues with not having a Raspberry-Pi module and tried to come up with solutions and alternatives
25/05/23	Continued to build product structure
30/05/23	Received a Raspberry-Pi 3B + module from Mr. Halimi of the EEE labs
05/06/23	Started working on documentation Initial ideas for the poster Speech-to -Text (STT)
15/06/23	First poster draft finished
19/06/23	Uploaded ESP32 code to test the motor function Discovered errors in the code when trying to connect to the API
20/06/23	Continued to fix ESP32 code as we encountered some errors with the weather API integration
22/06/23	Tested ESP32 code Confirmed correct functioning of motors and screen Tested IBM Watson Confirmed correct functioning of IBM's Watson Assistant when asked what the weather is Finalised the product and prepared for the demo day

4 GitHub repository and bill of materials (BOM)

4.1 Github repository

Our group frequently used GitHub (https://github.com/littlecsi/AI-Weather-Art) to upload and track our work for the project. The software aspect of the project related to the implementation of IBM's Watson was developed using the Python programming language, as the program will run on a Raspberry-Pi 3 B+ model. Our workflow involved programming and testing the Python files on our PCs, and then testing functionality on the Raspberry Pi. We chose this workflow because the Lite version of the Raspberry Pi's operating system has no GUI and only a terminal interface. Therefore, it was important to write and test the code separately on our PCs instead of directly coding on the Raspberry-Pi, as it would have been difficult to use VIM or the like to write code on the terminal.

The GitHub repository consists of four directories: `esp32`, `pi`, `sample`, and `watson`. The `esp32` directory contains the code for the ESP32. The `pi` directory contains the Python files that will be tested on the Raspberry-Pi. The `sample` directory contains audio files created and used for IBM's Watson Assistant. The `watson` directory contains the Python files created for testing IBM's Watson Assistant such as Speech-to-Text (STT), Assistant features, and Text-to-Speech (TTS). Once these files were successfully tested, the code was modularized.

4.2 Bill of materials (BOM)

The following bill of materials (BOM) provides a comprehensive inventory of all the components and materials required for the project. It includes information such as the name of the parts, the distributor from which we obtained said items and finally the per unit and total costs.

Item	Distributor	Unit cost	Total cost
180 degree motors	Farnell	£3.912	£23.472
Raspberry pi module	Farnell	£33.88	£33.88
Waveshare 2.9” E-ink Display (296x128)	Amazon	£24.99	£24.99
Electrical wire	EEE Labs	FREE	FREE
ESP32	EEE Labs	FREE	FREE
Limit Switches (x4)	RS Online	£3.08	£18.48
Micro continuous / 360 degree rotation servos (x4)	Farnell	£5.652	£22.608
USB cable (for uploading code and powering project)	EEE Labs	FREE	FREE
M2.5x14 Bolts (x12)	EEE Labs	FREE	FREE
M2.5x8 Bolts (x7)	EEE Labs	FREE	FREE
Nuts (x10)	EEE Labs	FREE	FREE
90 Degree / Corner Header Pins	EEE Labs	FREE	FREE
Bronze and gold PLA filament for 3D printing	EEE Labs	FREE	FREE
Raspberry-Pi 3 B+	EEE Labs	FREE	FREE
Microphone	Farnell	£2.808	£2.808
Speaker	Farnell	£3.348	£3.348
Stripboard	RS Online	£9.91	£9.91
TOTAL COST			**£139.50**

5 Ethical considerations

The purpose of technology and artificial intelligence (AI) is to empower and improve the quality of life of people around the world. However, we recognise that AI technology can also raise significant ethical concerns which must be clearly presented and addressed. Achieving that requires engineers and scientist who develop the algorithms to take a step back and assess the implications of their work in a broader socioeconomical and environmental context. Given the fact that the popularity of AI technology is rapidly increasing, it is also key to acknowledge that literacy regarding the subject must not be taken for granted. It is therefore necessary to also provide an explainable overview of the operation of KAIROS without sacrificing transparency and detail.

Kairos is a device which aims to bridge the gap between the elderly and AI technology. Its main purpose is to display the weather forecast in an artistic manner by rotating four scene discs stacked on top of each other. It also displays weather forecast data such as temperature and precipitation on a screen below the wheels. The AI aspect of the project, however, is introduced when developing its secondary features which rely on IBM’s Watson Assistant which handles all Speech-to-Text (STT) and Text-to-Speech (STT) conversions. In essence, KAIROS can be vocally prompted to do specific tasks such as calling friends, playing music and reading the news. Thus, the AI aspect of KAIROS revolves around Natural Language Processing also known as NLP.

Ethical considerations when developing and using NLP bots, especially when targeted at the elderly, require careful attention to ensure that their privacy, dignity and safety are respected [3].

Safeguarding the privacy and personal information of our users is a priority to us. Data collection, storage and processing are only done after informed consent and are implemented with security protocols to minimise the risks of unauthorised access and data breaches. Specifically, IBM’s Watson Assistant service, which handles all language processing tasks, is designed with ethical principles and security measures, providing a secure environment for data processing. Users can trust that their information is handled responsibly and securely within the system.

We recognise that an NLP bot does not substitute human connection and interaction but instead complements the everyday life of its user by improving overall quality of life. Loneliness and mental health issues among the elderly are significant concerns [4], and our aim is to develop a bot that provides empathy and companionship. We strive to create an AI bot that understands and acknowledges the unique needs and emotions of elderly users, aiming to support in a considerate and empathetic manner.

It is important to also consider that the elderly population may be more vulnerable to cybersecurity threats, scams and abuse. IBM's Watson Assistant has robust security measures to ensure that the data is kept safe. Moreover, it is important to note that IBM does not share unique insights from that data to inform the AI solutions they develop [5]. We understand the importance of maintaining the privacy and confidentiality of sensitive information shared with the bot and we strive to provide a secure environment where users can confidently interact with our AI bot, knowing that their personal information is well-protected.

6 Sustainability report

Sustainable products aim to minimize their negative impact on the overall environment, whether that's nature or society. Therefore, we carefully checked the materials and practices used in order to assess the sustainability implications of our Weather Bot. We made a conscious decision to use PLA filament to 3D print our components. PLA is derived from renewable natural sources such as corn, making it an environmentally friendly option. Notably, PLA is also biodegradable under the right conditions, further enhancing its sustainability.

In terms of hardware, we ultimately used a Raspberry-Pi 3 B+ single-board computer module for integrating IBM's Watson Assistant. The Raspberry-Pi complies with EU regulations on the supply of electronic and computing equipment and ensures compliance with responsible procurement practices. In addition, Sony Group Corporation, the manufacturer, has a policy that extends to the procurement of "conflict minerals". To comply with environmental standards, its PCB assembly complies with the ROHS directive, which prohibits the use of harmful elements such as Lead in manufacturing [6]. In addition, the Raspberry-Pi 3 B+ requires only 2.5-3.5 Watts, making it energy efficient.

Inclusivity was a major focus of our project, especially on getting the elderly people involved in new technology. During the design phase, we prioritized simplicity to maximize user engagement. As a result, a prototype was developed with no unnecessary features that would complicate the interaction. Instead, the user can simply connect the device to their WiFi and enjoy KAIROS's company.

Our product also incorporates rigorous corporate governance principles that prioritize strict privacy and security measures as discussed previously. Each device is equipped with a unique API key and password that guarantees authorized access and authenticates software running on KAIROS. IBM and our group strictly comply with data protection regulations, guarantee the confidentiality of all data collected and never transfer or publish them.

By carefully considering sustainability, inclusivity and privacy, we have developed a product that not only includes environmentally friendly materials and principles, but also provides a user-friendly and safe experience for our everyone.

7 References

References

[1] DIY-Machines, "Weatherbot - a motorised weather machine." https://www.instructables.com/WeatherBot-a-Motorised-Weather-Machine-3D-Printabl/.

[2] OpenWeather, "Openweather-api." https://openweathermap.org/.

[3] UNESCO, "Recommendation on the ethics of artificial intelligence." https://unesdoc.unesco.org/in/documentViewer.xhtml?v=2.1.196&id=p::usmarcdef_0000381137&file=/in/rest/annotationSVC/DownloadWatermarkedAttachment/attach_import_e86c4b5d-5af9-4e15-be60-82f1a09956fd%3F_%3D381137eng.pdf&locale=en&multi=true&ark=/ark:/48223/pf0000381137/PDF/.

[4] IBM, "Everyday ethics for artificial intelligence." https://www.ibm.com/watson/assets/duo/pdf/everydayethics.pdf.

[5] IBM, "Watson privacy, compliance and security." https://www.ibm.com/watson/assets/duo/pdf/Watson-Privacy-and-Security-POV_final_062819_tps.pdf.

[6] Raspberry-Pi, "Environmental impact of the raspberry pi." https://forums.raspberrypi.com/viewtopic.php?t=63620.