

SAVONIA

University of Applied Sciences

THESIS – BACHELOR'S DEGREE PROGRAMME
TECHNOLOGY, COMMUNICATION AND TRANSPORT

AUGMENTED REALITY FOR CONTROLLING IOT DEVICES

AUTHOR Sojat Sebgaze

Field of Study Technology, Communication and Transport	
Degree Program Degree Programme in Information Technology, Internet of Things	
Author Sojat Sebgaze	
Title of Thesis Augmented reality for controlling IoT devices	
Date 31.05.2024	Pages/Number of appendices 48
Client Organisation /Partners Savonia University of Applied Sciences	
<p>Abstract</p> <p>The integration of Augmented Reality (AR) and the Internet of Things (IoT) represents a significant technological advancement, with potential applications across various sectors. This thesis explored how AR can be utilized as a user interface for controlling IoT devices, aiming to improve user experience and operational efficiency. By developing a prototype application, the research investigated the feasibility, challenges, and practical implications of this integration.</p> <p>The AR application was developed using the Unity real-time development platform and Vuforia Engine, enabling users to control an AC fan through virtual controls overlaid in the real-world environment. The implementation involved 3D scanning for interfacing virtual information, allowing users to adjust fan speed, monitor temperature readings in real time, and visualize components in detail. Despite facing technical challenges such as compatibility issues and data transmission latency, the application provided valuable insights into the practical integration of AR and IoT technologies.</p> <p>The successful implementation of the AR application demonstrated its potential to enhance user experiences by providing real-time visualization of IoT data and intuitive control over device parameters. The research highlights the importance of addressing technical and usability challenges for the broader adoption of AR-IoT solutions. In conclusion, this thesis contributes to advancing the knowledge in AR-IoT integration, paving the way for future innovations and applications in diverse sectors.</p>	
<p>Keywords AR, IoT, user interface, user experience, integration, real-time, prototype, data visualization, data exchange, overlay, compatibility.</p>	

CONTENTS

1	INTRODUCTION	6
2	LITERATURE REVIEW	7
2.1	Research Studies and Projects.....	7
2.1.1	Home Automation	7
2.1.2	Retail (Shopping Experience)	7
2.1.3	Education	8
2.1.4	Manufacturing Industry	8
2.2	Similar Technologies	9
2.3	Emerging Trends in AR and IoT.....	10
2.4	Challenges and proposed solutions	10
3	IMPLEMENTATION.....	12
3.1	Research design	12
3.2	IoT Device	14
3.2.1	Hardware.....	14
3.2.2	Software.....	15
3.3	AR Application	20
3.3.1	Setting Up Unity	20
3.3.2	3D Scan.....	24
3.3.3	Model Target Generator	27
3.4	AR + IoT Integration	32
3.4.1	Integrating Model Target	32
3.4.2	Connecting IoT Device and AR Application	34
3.4.3	Deploying and Building Mobile Application.....	39
4	FINDINGS AND ANALYSIS	41
4.1	Result	41
4.2	Limitations	42
4.3	Future Implications.....	43
5	CONCLUSION	44
	REFERENCES.....	45

LIST OF FIGURES

Figure 1. Flowchart for the application.....	13
Figure 2. Circuit Diagram	14
Figure 3. Code for Testing Components	15
Figure 4. Serial Monitor Result for Testing Components	16
Figure 5. Libraries and Constants	17
Figure 6. Functions.....	17
Figure 7. Setup and Loop	18
Figure 8. Serial Monitor Result	18
Figure 9. Temperature Reading.....	19
Figure 10. Motor Speed	19
Figure 11. Motor Speed Control.....	19
Figure 12. Create a New Project in Unity Hub	20
Figure 13. Sample Scene	21
Figure 14. Vuforia Package Download.....	21
Figure 15. Import Unity Package	22
Figure 16. Add AR Camera.....	22
Figure 17. Vuforia Configuration.....	23
Figure 18. Create License Key	23
Figure 19. Add App License Key	23
Figure 20. AC Fan Prototype	24
Figure 21. First 3D Scan With Polycam	25
Figure 22. Prototype with PCB.....	25
Figure 23. Second 3D Scan	26
Figure 24. Last 3D Scan.....	26
Figure 25. Create Model Target.....	27
Figure 26. Model Up Vector	27
Figure 27. Model Units.....	28
Figure 28. Coloring.....	28
Figure 29. Optimize Tracking	29
Figure 30. Advanced Guide View	30
Figure 31. Creating Training Session	30
Figure 32. Creating Database.....	31
Figure 33. Add Model Target in Unity	32

Figure 34. UI Canvas Layout.....	33
Figure 35. Model Target Test.....	33
Figure 36. Adding Script	34
Figure 37. Temperature Script 1.....	35
Figure 38. Temperature Script 2.....	35
Figure 39. Setting Up Temperature Script.....	36
Figure 40. Speed Script 1	37
Figure 41. Speed Script 2	37
Figure 42. Setting Up Speed Script.....	38
Figure 43. Android Settings.....	39
Figure 44. Build and Run Android App	40
Figure 45. AR Mobile Application	41
Figure 46. Components	41
Figure 47. Load Temperature.....	42

1 INTRODUCTION

The integration of Augmented Reality (AR) and the Internet of Things (IoT) represents a significant role forward in technology, with far-reaching implications across various sectors. This thesis explores the integration of AR and IoT, how AR technology can be utilized as a user interface for controlling IoT devices, potential challenges, and practical implications of this combination by developing a prototype.

The Internet of Things (IoT) encompasses a network of interconnected devices embedded with sensors, software, and other technologies, facilitating the exchange of data, and enabling applications that enhance security, comfort, and efficiency. Its applications span across diverse sectors such as smart homes, healthcare, smart cities, industry, and among others. (AWS.)

Augmented Reality (AR) overlays digital information onto the real world, creating an interactive experience that blends virtual and physical environments (Microsoft). Utilizing devices such as smartphones, tablets, or AR glasses, AR technology is particularly valuable in enhancing learning, problem-solving, and manufacturing processes within Industry 4.0.

The convergence of Augmented Reality (AR) and the Internet of Things (IoT) represents a groundbreaking frontier in technological innovation, offering unique opportunities to revolutionize various industries. By improving user experiences through seamless and interactive interfaces, revolutionizes various sectors such as manufacturing, healthcare, retail, and entertainment. For instance, in manufacturing, real-time IoT data overlaid with AR visuals allows technicians to monitor device performance, identify potential issues, and ensure a safer work environment while reducing downtime and streamlining maintenance workflows (Matiieshyn & Kalachova 2024). In education, the combination of AR and IoT offers immersive training experiences, allowing learners to interact with virtual environments overlaid on the real world. Similarly, in retail, IoT data can personalize shopping experiences, while AR technology enables customers to visualize products and make more informed purchasing decisions. Entertainment and gaming also benefit from the convergence of IoT and AR, blurring the boundaries between reality and the virtual world and creating engaging, personalized experiences for players and spectators alike. (Carroll II 2023.)

The purpose of the thesis is to explore the feasibility, challenges, and implications of using AR for controlling IoT devices. Aim to identify practical applications, potential limitations, and enhance user experiences through the integration of these technologies by addressing how can AR effectively control IoT devices and challenges with proposed solutions. Specifically, showcasing the AR application controlling AC fan. The interest in this topic was sparked by an immersive internship experience in AR, which provided invaluable insights into the practical applications and potential of AR technology in controlling IoT devices.

In the following chapters, the integration of AR and IoT technologies, literature, research methodology, and present findings from the practical application of AR controlling AC fan will be discussed. By the end of this thesis, hope to contribute valuable insights to the field of AR-IoT integration and pave the way for future advancements in this transformative area of technology.

2 LITERATURE REVIEW

In recent years, the integration of Augmented Reality (AR) and the Internet of Things (IoT) has gained increasing attention from researchers. This section provides an overview of recent research studies, projects, and emerging trends in the field, along with challenges faced in integrating AR with IoT devices and proposed solutions to address the problems.

2.1 Research Studies and Projects

There are quite a few research papers done on the topic, mainly focusing on one sector and the benefits that would be gained by overlaying IoT and AR technology. Selected research and articles will be studied in this section, on how the integration of IoT and AR benefit home automation, retail, education, and industry sectors.

2.1.1 Home Automation

Smart home automation has emerged as a promising field, particularly with the integration of augmented reality (AR) and Internet of Things (IoT) technologies. A study by Jaivignesh, Janarthanan, and Gnanalakshmi, demonstrated the concept of controlling home appliances using AR in conjunction with IoT through a Wi-Fi-based microcontroller. Their findings highlighted the complementary nature of AR and IoT, showing that visual interactive objects effectively monitored IoT devices. (Jaivignesh et al 2022.)

The researchers also developed an AR application capable of controlling various home appliances, including bulbs, fans, and laptops, while providing real-time monitoring of room temperature. Testing of the smart home automation model encompassed different devices, with analyses conducted based on Wi-Fi strength to ensure a user-friendly interactive environment.

A notable aspect of the study was the comparison of the sensitivity and performance of NodeMCU and ESP32 microcontrollers for controlling home appliances, with ESP32 demonstrating faster response times. These comprehensive findings underscore the successful integration of AR and IoT technologies for smart home automation, demonstrating the feasibility and effectiveness of using AR to control IoT devices and monitor home appliances in real-time.

2.1.2 Retail (Shopping Experience)

The integration of augmented reality (AR) with the Internet of Things (IoT) has shown significant promise in enhancing various aspects of human life, including shopping experiences. Jo and Kim explore this intersection in their study, highlighting the successful integration of AR with IoT to improve shopping experience realms. (Jo & Kim 2019.)

The key finding of their paper is the seamless combination of AR interfaces with IoT control interfaces, offering scalable AR services for IoT-ready products. This integration allows users to access, control, and interact with physical objects naturally and intuitively, thereby enhancing usability and user satisfaction in shopping scenarios.

Moreover, the authors emphasize the importance of standardized content representation and system interoperability protocols in ensuring efficient data exchange and interaction between physical ob-

jects and digital AR overlays. These protocols contribute to an enhanced overall shopping experience by facilitating smooth communication between the physical and digital realms.

2.1.3 Education

In recent years, the integration of Augmented Reality (AR) enhanced by Internet of Things (IoT) data has emerged as a promising approach to revolutionize digital education. Shoikova, Nikolov, and Kovatcheva dig deep into this intersection, focusing on its potential to significantly impact Smart Digital Education by providing real-time information in the user's environment. (Shoikova et al 2018.)

The key finding of their paper is the transformative potential of AR enriched by IoT data to revolutionize digital education. By leveraging real-time information in the user's environment, this integration offers dynamic and interactive learning experiences. The authors emphasize the importance of ongoing research, skills development, and the continuous adaptation of education frameworks to effectively leverage these new technologies.

Furthermore, the paper discusses the growing market for AR in education, with analysts predicting substantial growth by 2021. This growth reflects a rising recognition of the benefits of AR-enhanced learning environments. The authors discuss various trends in AR that are shaping education, including AR textbooks, AR gaming, and student-centric digital environments.

However, along with the opportunities presented by AR and IoT integration in education, the paper also acknowledges several challenges. These include issues related to access, equity, privacy, and data security. Despite these challenges, the authors highlight the vast opportunities associated with integrating AR and IoT data into educational settings, emphasizing the need for a balanced approach that maximizes benefits while addressing potential risks.

2.1.4 Manufacturing Industry

The integration of Augmented Reality (AR) and Internet of Things (IoT) technologies in manufacturing presents significant opportunities for enhancing efficiency and productivity. The synergy between AR and IoT allows for the visualization, analysis, and understanding of data in 3D, aiding in effective decision-making and problem identification. AR also enables remote assistance, allowing frontline employees to connect with experts in real-time for guidance on complex. (Adamska 2023.)

The combined benefits include improved data visualization, enhanced problem identification, smarter decision-making, human capability enhancement through training simulations, efficient asset management, cost reduction, and increased revenue through faster operations and new product/service development. Companies have been deploying these technologies to leverage these advancements, create supporting ecosystems, and ensure skilled personnel.

TechSight smart glasses from Honeywell Intelligrated revolutionize technical support for distribution centers and manufacturing operations. These cutting-edge glasses enable real-time collaboration between field technicians and remote support engineers, significantly reducing downtime and operational costs. With live video, audio, and augmented reality overlays, technicians can project their viewpoint to remote experts, who can provide instant diagnosis and troubleshooting guidance. The

hands-free design allows technicians to receive annotated schematics and instructions while making repairs, enhancing efficiency and accuracy. The augmented reality capabilities facilitate the resolution of complex issues, saving time and improving productivity. By leveraging the expertise of both field maintenance crews and technical support engineers, TechSight ensures fast, effective support, minimizing disruptions and maximizing operational uptime. (Kelton 2019.)

2.2 Similar Technologies

Several technologies share similarities with AR and can integrate with IoT to create even more powerful and immersive experiences. Some of the examples include Virtual Reality (VR), and Mixed Reality.

Virtual Reality (VR) creates a fully immersive digital environment, different from AR, which overlays digital information onto the physical world. However, VR can complement AR in several ways. For example, VR can be used for training technicians on maintaining complex IoT devices or for visualizing large datasets collected from IoT sensors, providing a safe and controlled environment for learning and analysis. (Termosa 2024.)

Mixed Reality (MR) merges the real and digital worlds, allowing users to interact with both simultaneously. Similar to AR, MR overlays digital information onto the physical world but goes a step further by enabling users to manipulate digital objects in real space. This makes MR particularly useful for applications like assembling IoT devices with step-by-step instructions overlaid directly on physical components or for overlaying maintenance information onto physical equipment, enhancing the precision and effectiveness of these tasks. (Termosa 2024.)

Augmented Reality (AR) overlays digital information onto the real world without completely shutting out the surroundings. This makes AR ideal for situations where awareness of the physical environment is necessary. AR can be accessed through smartphones and tablets, which most people already own, eliminating the need for bulky headsets required by VR and MR. This accessibility makes AR a convenient entry point for interacting with IoT devices, seamlessly integrating the digital world into our daily lives. AR is sometimes used loosely to describe MR glasses as well, this is because MR is a newer term, or because AR is more widely understood. AR overlays digital content onto the real world, while MR merges the real and digital worlds and allows interaction with both. (Martinez 2024.)

2.3 Emerging Trends in AR and IoT

The integration of Internet of Things (IoT) and Augmented Reality (AR) technologies opens up major possibilities across various industries. One emerging trend lies in remote assistance and maintenance, where AR overlays utilize real-time data from IoT sensors to guide technicians through repairs, reducing downtime and enhancing efficiency. (Lincoln 2021.)

Smart retail and marketing are also witnessing transformations as retailers integrate IoT sensors with AR applications to deliver personalized shopping experiences, leveraging data insights to offer tailored product information and promotions. (Bluestone 2023.)

In addition, the convergence of IoT and AR is revolutionizing fields like healthcare, where IoT devices collect patient data and AR interfaces provide medical professionals with real-time access during consultations. This integration enhances diagnosis accuracy and patient care quality. Meanwhile, smart cities benefit from IoT-enabled data collection and AR visualization, empowering urban planners with insights for better infrastructure decisions. These trends reflect a growing synergy between IoT's data-gathering capabilities and AR's immersive interfaces, promising transformative impacts across diverse sectors. (Fingent.)

2.4 Challenges and proposed solutions

Data Overload and Processing

The integration of IoT sensors generates massive volumes of data, overwhelming systems, and networks. AR applications require real-time access to this data for meaningful overlays, imposing challenges in data processing and transmission. (PTC.)

Implementing edge computing solutions to process data closer to the source, means processing data closer to where it's collected, which makes things faster and uses less bandwidth. Utilize data filtering to prioritize relevant information for AR overlays, ensuring efficient use of resources. (Suliman et al 2022.)

Compatibility and Standards

IoT devices and AR platforms often operate on different standards and protocols, seamless integration, and interoperability. Incompatibilities between hardware and software components can lead to integration complexities and limited scalability.

Promoting industry-wide standards and protocols for IoT and AR technologies to facilitate interoperability and seamless integration. Encourage collaboration between hardware manufacturers and software developers to ensure compatibility across devices and platforms. (InfiSIM 2024.)

Privacy and Security Concerns

The proliferation of IoT devices and AR applications raises concerns about data privacy and security. Vulnerabilities in IoT devices and AR platforms can expose sensitive information to unauthorized access and cyber threats. (InfiSIM 2024.)

Implementing robust security measures, such as encryption and authentication mechanisms, to protect data transmitted between IoT devices and AR applications. Employ privacy-enhancing technolo-

gies, such as differential privacy and anonymization techniques, to safeguard user data while preserving utility. (Verizon 2023.)

User Experience and Adoption

Despite the potential benefits, complex user interfaces and limited user familiarity with AR technologies may hinder adoption. Poorly designed AR overlay interactions can detract from the user experience and impede widespread adoption.

Prioritizing user-centric design principles to create intuitive AR interfaces that enhance usability and engagement. Provide training and educational resources to familiarize users with AR technology and its practical applications across various domains.

By addressing these challenges with innovative solutions, the integration of IoT and AR technologies can unlock new opportunities for transformative experiences and advancements across industries.

3 IMPLEMENTATION

For the practical implementation, an AR application has been developed to control an AC fan. This demonstrates how AR can effectively serve as a user interface for controlling IoT devices, providing real-time data and detailed component descriptions.

3.1 Research design

The prototype consists of four main components: the user interface, application layer, communication module, and hardware.

- **User Interface (UI):** The UI is the mobile device that displays AR elements and allows interaction with the AC fan. The AR application on the mobile device provides a digital overlay to control and monitor the fan, displaying real-time data from the IoT device. It provides control for adjusting the fan speed and monitoring the temperature.
- **Application Layer:** This layer includes the AR application and the IoT device. The AR application, developed in Unity, communicates with the server to fetch data, and send control commands. The IoT device is programmed to handle temperature readings and motor control, ensuring seamless interaction with the AR application.
- **Communication Module:** This module manages the communication protocol between the IoT device and the AR application. The IoT device connects to Wi-Fi and sends data to an HTTP server. The AR application accesses this data over the same network, allowing it to fetch sensor data and send control commands to the IoT device. It effectively links the sensor and microcontroller to the Wi-Fi network, facilitating smooth data exchange.
- **Hardware:** The hardware comprises the physical components that are being controlled, including the sensor, microcontroller, and other necessary components. These elements work together to receive commands from the AR application and execute the desired actions on the AC fan.

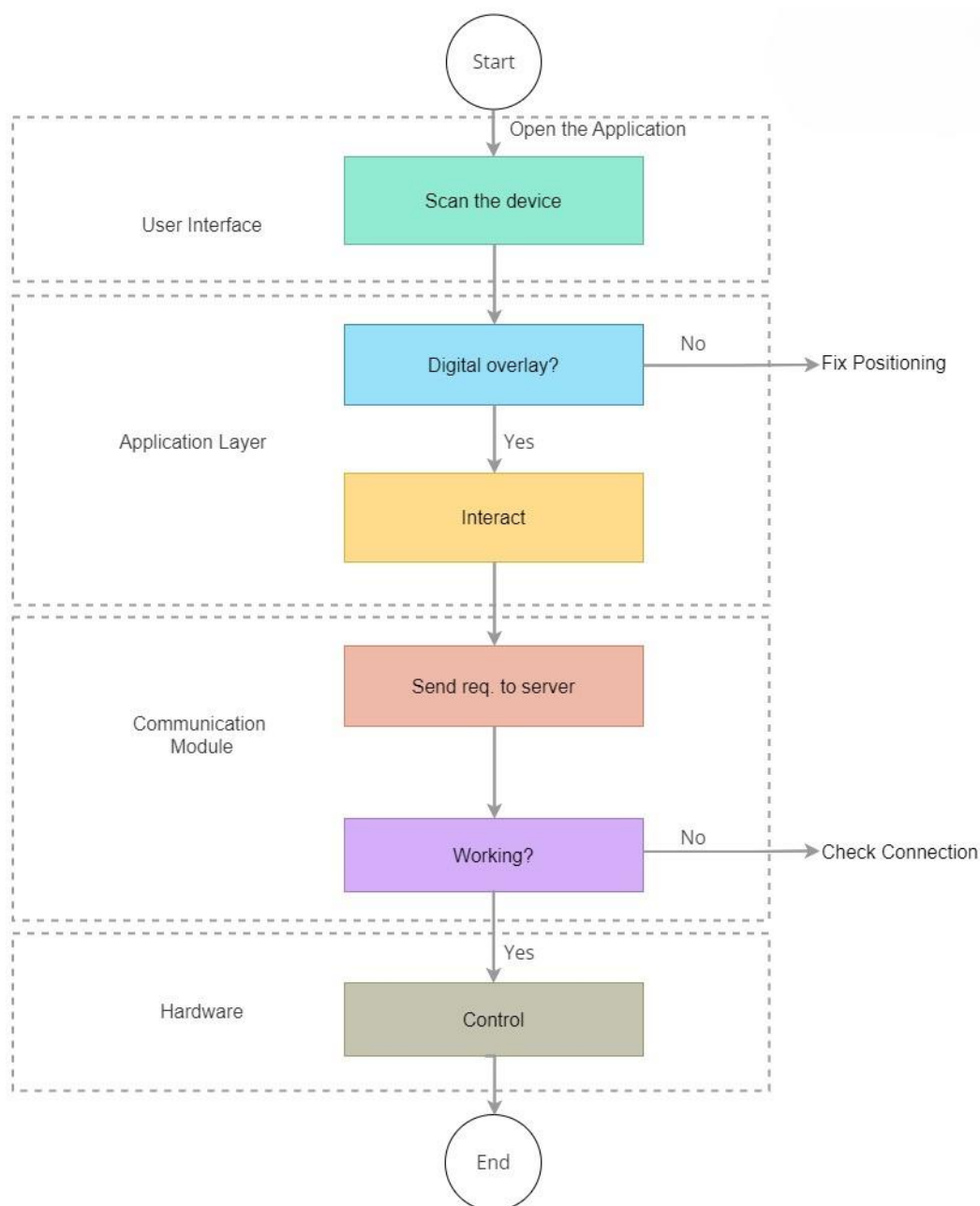


Figure 1. Flowchart for the application.

The flowchart in Figure 1 outlines the steps involved in using an AR application to control an AC fan. The process begins when the user opens the AR app on their mobile device. The application then scans the AC fan to detect its presence. If the app successfully overlays the digital controls onto the physical fan, the user can interact with the AR interface. This interaction allows the user to adjust settings, such as the fan speed, and monitor real-time data like temperature. If the device is not detected, the user needs to fix the positioning and try again. The AR app sends these control commands or data requests to an HTTP server, which communicates with the IoT device. If the server successfully processes these requests, the fan responds accordingly, allowing the user to continue controlling the fan. If there is an issue with the server request, the user is prompted to check the connection and try again. The process concludes when the user finishes interacting with the AR application, providing a seamless experience of controlling and monitoring the IoT device through AR.

3.2 IoT Device

3.2.1 Hardware

This subchapter details, the development of an IoT-enabled AC fan that can automatically adjust its speed based on temperature readings from the sensor. The project integrates various hardware components:

- **Microcontroller** - The ESP32-WROOM-32 is a powerful Wi-Fi, Bluetooth, and Bluetooth LE (low energy) all-in-one chip designed for various purposes, from simple sensor networks to demanding tasks like collecting industrial data. It has two cores that can be turned on or off to save power, and it can connect to the internet through Wi-Fi or directly to a phone with Bluetooth. (Espressif Systems 2019).
- **Temperature Sensor** –The AM2311A sensor measures both temperature and humidity very precisely. It uses advanced technology to ensure long-lasting accuracy and reliable readings and provides calibrated digital signal output. It uses the same library as DHT sensors. (Aosong Electronics.)
- **Transistor** - The BC547B is a general-purpose NPN transistor, it is designed for use in linear and switching applications. It controls the electric current in circuits, making it useful for turning things on and off and protecting circuits from surges. (ON Semiconductor.)
- **Diode** - 1N4007 is a Rectifier Diode with high current capability and low forward voltage drop. 1N4007G(LS) is a small diode, a one-way street for electricity. It allows current to flow in one direction (forward) easily, with a small voltage drop. In the other direction (reverse), it blocks current very well. (Diodes Incorporated.)
- **DC Motor** – as a fan.
- **9V battery**
- **Resistor**

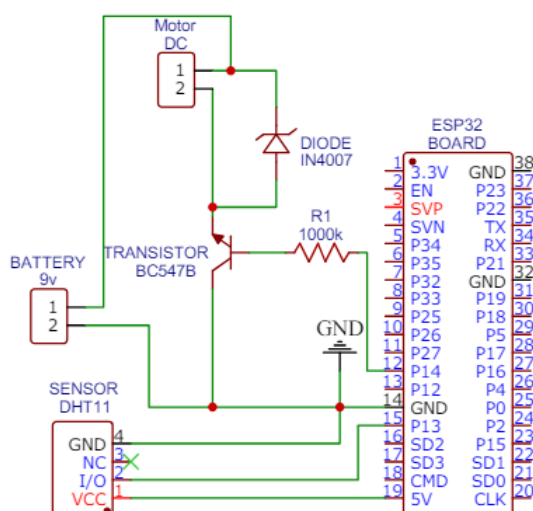


Figure 2. Circuit Diagram.

The circuit diagram in Figure 2 illustrates the setup for controlling an AC fan using an ESP32 micro-controller, powered by a 9V battery, and integrated with an AM2311A sensor for real-time temperature and humidity monitoring. A BC547B transistor serves as a switch to regulate the DC motor's speed based on signals from the ESP32-WROOM-32, while a 1N4007 diode protects against voltage spikes. A 1000-ohm resistor limits current to the transistor's base for proper function. Through communication with the sensor, the ESP32-WROOM-32 adjusts the motor speed, enabling automatic fan speed modulation.

3.2.2 Software

Before integrating the Wi-Fi functionality, the components were tested to ensure proper operation. The initial code was written to initialize the AM2311A sensor and ESP32-WROOM-32 read the temperature from the sensor, print the temperature data to the serial monitor, and control the motor speed using PWM(Pulse width modulation) based on a cyclic increment of a mapped speed value.

PWM is a technique used to control the speed of a motor. It works by switching the power on and off rapidly, varying the proportion of time (duty cycle) where power is on relative to the time it is off. (Electronics Tutorials.)

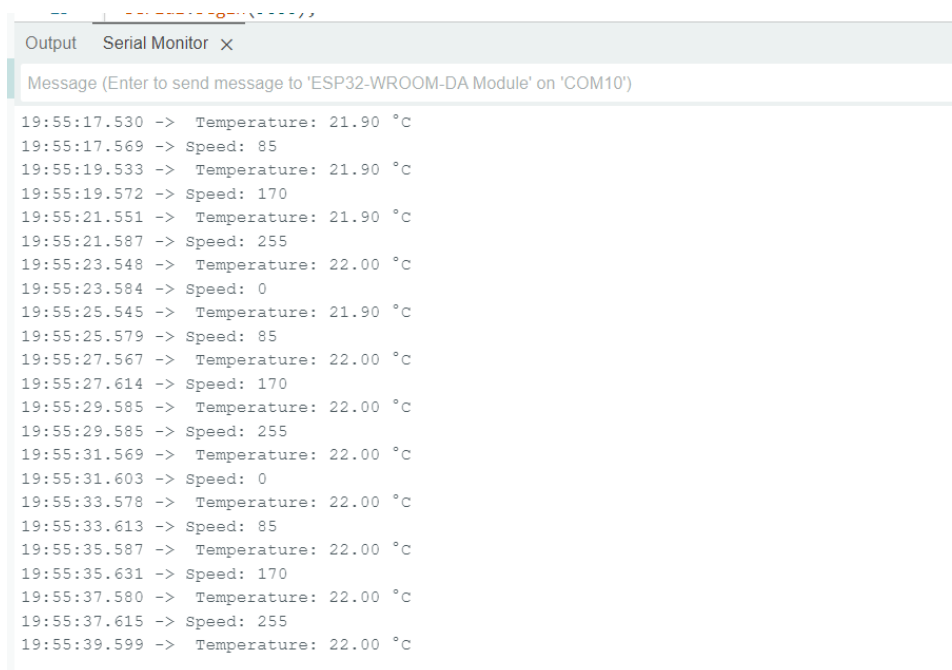
```

1  #include <DHT.h>;
2  //Constants
3  #define DHTPIN 13          //pin connected to
4  #define DHTTYPE DHT21     //DHT 21 (AM2301)
5  #define MOTOR_PIN 14
6  DHT dht(DHTPIN, DHTTYPE); //Initialize DHT sensor
7  //Variables
8  float temp;
9  int speed = 0;
10 int mappedSpeed = 0;
11
12 void setup() {
13   Serial.begin(9600);
14   dht.begin();
15   analogWrite(MOTOR_PIN, 0);
16 }
17
18 void loop() {
19
20   temp= dht.readTemperature();
21   Serial.print(" Temperature: ");
22   Serial.print(temp);
23   Serial.println(" °C");
24
25   speed = map(mappedSpeed, 0, 3, 0, 255);
26   analogWrite(MOTOR_PIN, speed);
27   mappedSpeed = (mappedSpeed + 1) % 4;
28   Serial.print("Speed: ");
29   Serial.println(speed);
30
31   delay(2000); //Delay 2 sec
32 }

```

Figure 3. Code for Testing Components.

The code in Figure 3 controls the speed of a motor using PWM (Pulse Width Modulation) based on cyclic increments of a mapped speed value. It first initializes the sensor and motor pin, then enters a loop where it reads the temperature from the AM2311A sensor and prints it to the serial monitor. The motor speed is determined by mapping a cyclically incremented variable (mappedSpeed) from a range of 0 to 3 to a PWM range of 0 to 255, generating four distinct speed levels (0, 85, 170, 255). The PWM signal is applied to the motor pin, adjusting the motor speed accordingly. The mappedSpeed variable is incremented in each loop iteration, cycling through the desired values to change the motor speed every 2 seconds, which is also printed to the serial monitor.



The screenshot shows a Serial Monitor window with the following output:

```

Output  Serial Monitor  X
Message (Enter to send message to 'ESP32-WROOM-DA Module' on 'COM10')
19:55:17.530 -> Temperature: 21.90 °C
19:55:17.569 -> Speed: 85
19:55:19.533 -> Temperature: 21.90 °C
19:55:19.572 -> Speed: 170
19:55:21.551 -> Temperature: 21.90 °C
19:55:21.587 -> Speed: 255
19:55:23.548 -> Temperature: 22.00 °C
19:55:23.584 -> Speed: 0
19:55:25.545 -> Temperature: 21.90 °C
19:55:25.579 -> Speed: 85
19:55:27.567 -> Temperature: 22.00 °C
19:55:27.614 -> Speed: 170
19:55:29.585 -> Temperature: 22.00 °C
19:55:29.585 -> Speed: 255
19:55:31.569 -> Temperature: 22.00 °C
19:55:31.603 -> Speed: 0
19:55:33.578 -> Temperature: 22.00 °C
19:55:33.613 -> Speed: 85
19:55:35.587 -> Temperature: 22.00 °C
19:55:35.631 -> Speed: 170
19:55:37.580 -> Temperature: 22.00 °C
19:55:37.615 -> Speed: 255
19:55:39.599 -> Temperature: 22.00 °C

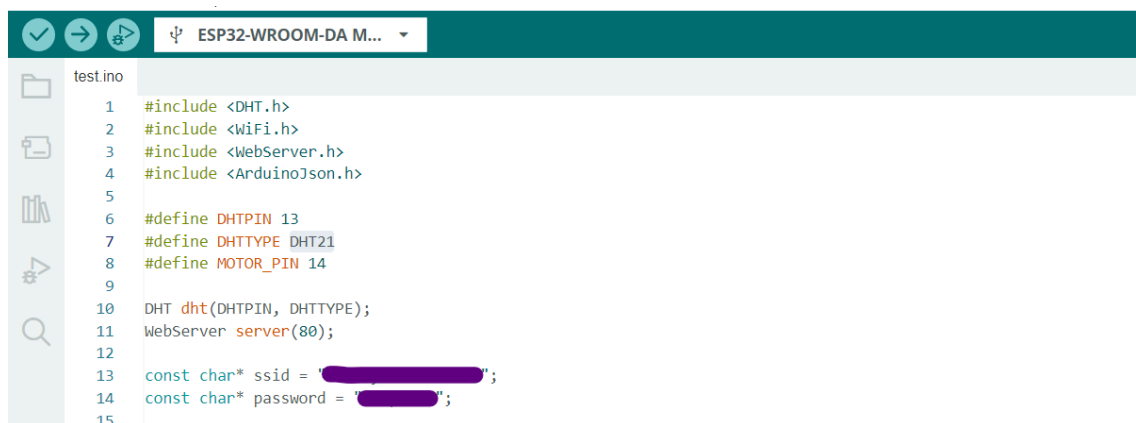
```

Figure 4. Serial Monitor Result for Testing Components.

The serial monitor shown in Figure 4 prints lines showing the current temperature reading from the sensor and the corresponding motor speed every 2 seconds.

After verifying the hardware components, the next step was to connect the ESP32 microcontroller to a Wi-Fi network and set up a web server. This allows remote monitoring and control of the AC fan through a web interface.

A web server stores website content processes client requests and delivers webpages through the internet using HTTP (Hypertext Transfer Protocol). The client refers to the web browser, which sends requests to the web server for specific website files. The web server processes the request, finds the requested file, and sends it back to the browser using HTTP. (MDN Web Docs.)



```

test.ino
1  #include <DHT.h>
2  #include <WiFi.h>
3  #include <WebServer.h>
4  #include <ArduinoJson.h>
5
6  #define DHTPIN 13
7  #define DHTTYPE DHT21
8  #define MOTOR_PIN 14
9
10 DHT dht(DHTPIN, DHTTYPE);
11 WebServer server(80);
12
13 const char* ssid = "XXXXXXXXXX";
14 const char* password = "XXXXXXXXXX";
15

```

Figure 5. Libraries and Constants.

The code in Figure 5 includes the required libraries for Wi-Fi, web server, DHT sensor (AM2311A sensor uses the same library as DHT sensors), and JSON handling. Define the pins for the temperature sensor and motor, the type of DHT sensor (AM2311A is equivalent to DHT21) and initialize the objects for the sensor and web server. At last, it defines the Wi-Fi SSID and password to connect to the Wi-Fi network.

```

16 void handleTemperature() {
17     float temp = dht.readTemperature();
18     if (isnan(temp)) {
19         server.send(500, "application/json", "{\"error\": \"Failed to read temperature from DHT sensor!\"}");
20     } else {
21         StaticJsonDocument<200> jsonDoc;
22         jsonDoc["temperature"] = temp;
23
24         String response;
25         serializeJson(jsonDoc, response);
26         server.send(200, "application/json", response);
27     }
28 }
29
30 void handleSpeed() {
31     int speed = server.arg("speed").toInt();
32     analogWrite(MOTOR_PIN, speed);
33
34     StaticJsonDocument<200> jsonDoc;
35     jsonDoc["speed"] = speed;
36
37     String response;
38     serializeJson(jsonDoc, response);
39     server.send(200, "application/json", response);
40 }

```

Figure 6. Functions.

As shown in Figure 6, the code creates functions to handle HTTP requests for temperature and motor speed.

- `handleTemperature()`: This function is called when a client sends an HTTP GET request to the `/temperature` endpoint. It reads the temperature from the sensor and sends it back to the client in JSON format.
- `handleSpeed()`: This function is called when a client sends an HTTP GET request to the `/speed` endpoint. It reads the speed value from the query string and sets the speed of the DC motor using `analogWrite()`. It then sends a JSON response back to the client that includes the new speed value. A new speed can be set by adding `/speed?speed=[desired value]` to the endpoint.

JavaScript Object Notation(JSON) is a simple way to format data, It's written in plain text, similar to how you create objects in JavaScript, which makes it easy for computers to understand and use this data. (MDN Web Docs.)

```

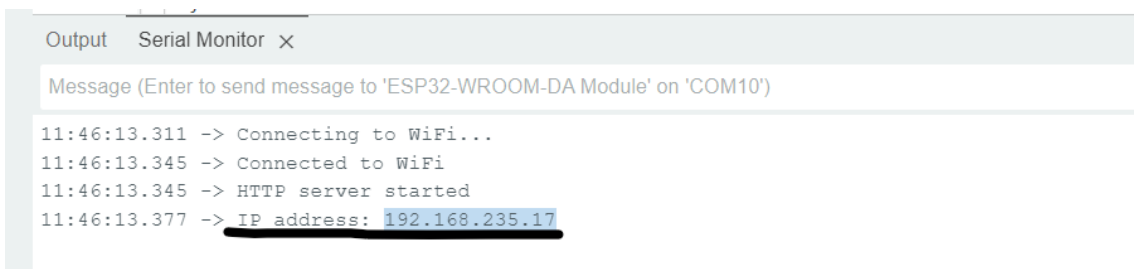
42
43 void setup() {
44     Serial.begin(9600);
45     dht.begin();
46     pinMode(MOTOR_PIN, OUTPUT);
47
48     WiFi.begin(ssid, password);
49     while (WiFi.status() != WL_CONNECTED) {
50         delay(1000);
51         Serial.println("Connecting to WiFi...");
52     }
53     Serial.println("Connected to WiFi");
54
55     server.on("/temperature", HTTP_GET, handleTemperature);
56     server.on("/speed", HTTP_GET, handleSpeed);
57
58     server.begin();
59     Serial.println("HTTP server started");
60
61     Serial.print("IP address: ");
62     Serial.println(WiFi.localIP());
63 }
64
65 void loop() {
66     server.handleClient();
67 }
68

```

Figure 7. Setup and Loop.

In the setup function shown in Figure 7, the ESP32-WROOM-32 initializes serial communication for debugging, the DHT sensor for temperature readings, and the motor control pin as an output. It then connects to the specified Wi-Fi network, prints connection status messages until connected, and starts the HTTP server with endpoints for reading temperature and setting motor speed. The server begins listening for incoming requests, and the ESP32-WROOM-32's IP address is printed for reference.

The loop function continuously calls `server.handleClient()`, which listens for, and processes incoming HTTP GET requests to the `/temperature` and `/speed` endpoints. This setup allows the microcontroller to handle client requests for temperature data and motor speed adjustments via Wi-Fi.



```

Output  Serial Monitor x
Message (Enter to send message to 'ESP32-WROOM-DA Module' on 'COM10')
11:46:13.311 -> Connecting to WiFi...
11:46:13.345 -> Connected to WiFi
11:46:13.345 -> HTTP server started
11:46:13.377 -> IP address: 192.168.235.17

```

Figure 8. Serial Monitor Result.

Uploading the code will initiate a connection check. The serial monitor will then display the board's status, including the Wi-Fi connection and server operation. If everything is running smoothly, will be able to see the IP address assigned to the board as shown in Figure 8.

Clients can access temperature readings and motor speed by visiting specific web addresses (end-points) on the board's IP address. These addresses are `/temperature` for temperature and `/speed` for motor speed.



The screenshot shows a web browser address bar with the URL `192.168.235.17/temperature`. The browser's developer tools are open, displaying the following network request details:

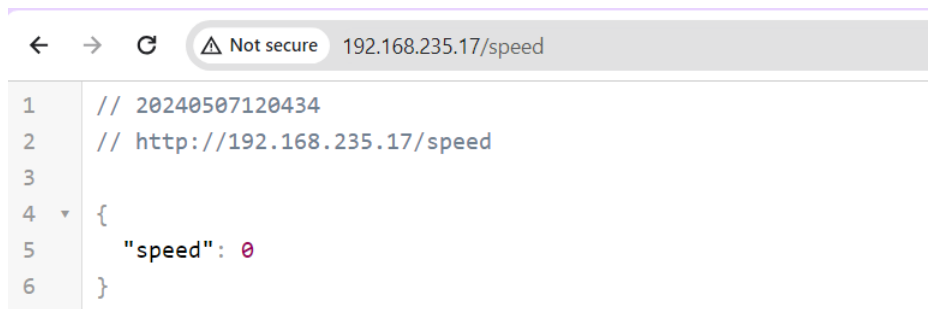
```

1 // 20240507120758
2 // http://192.168.235.17/temperature
3
4 {
5   "temperature": 23.60000038
6 }

```

Figure 9. Temperature Reading.

If the temperature reading is successful, the server responds with an OK status and a JSON object containing the temperature value as shown in Figure 9.



The screenshot shows a web browser address bar with the URL `192.168.235.17/speed`. The browser's developer tools are open, displaying the following network request details:

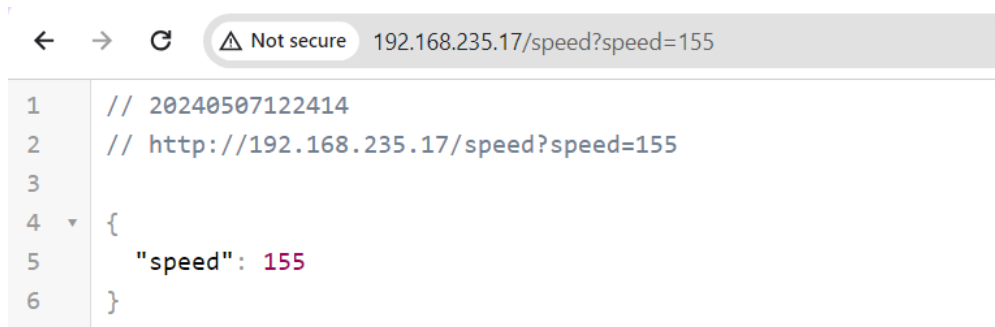
```

1 // 20240507120434
2 // http://192.168.235.17/speed
3
4 {
5   "speed": 0
6 }

```

Figure 10. Motor Speed.

If a client sends a request without specifying a speed value, the server will respond with the current motor speed or the default speed which is 0 as shown in Figure 10.



The screenshot shows a web browser address bar with the URL `192.168.235.17/speed?speed=155`. The browser's developer tools are open, displaying the following network request details:

```

1 // 20240507122414
2 // http://192.168.235.17/speed?speed=155
3
4 {
5   "speed": 155
6 }

```

Figure 11. Motor Speed Control.

To adjust motor speed, clients can send a request to the `/speed` endpoint on the board's IP address. This request should include the desired speed value for example `/speed?speed=155` and the motor speed will change accordingly as shown in Figure 11.

3.3 AR Application

3.3.1 Setting Up Unity

The AR application was developed in Unity Real-Time Development Platform. The Platform offers free access to the real-time 3D development platform used by professionals to create immersive experiences for students. After applying and consenting to data collection and processing, any student who enrolled in accredited educational institutions will be eligible to use and download the platform. (Unity Technologies.)

To install Unity Hub, download Unity Hub and run the downloaded installer following the instructions displayed on the download page. Upon the first launch, Unity Hub requests access to your file system and firewall to manage projects and access online resources. Grant these permissions to allow the Hub to function properly. Sign in with the student Unity ID created in the earlier stage.

Unity Hub is a central app for managing everything in Unity. It lets users download and switch between different versions of the Unity Editor software, organize game projects, and access learning materials like templates and tutorials. It also manages Unity accounts, and licenses, and get help from Unity, all within the Hub, making it a convenient starting point for any Unity development project. (Unity Technologies.)

To begin, open Unity Hub and select the latest Unity version compatible with Vuforia Engine. Next, choose for the 3D Core template, which includes essential features such as lighting and a camera, facilitating an easy start for any 3D project (Unity Technologies).

Name the project and proceed to create it, following the example shown in Figure 12.

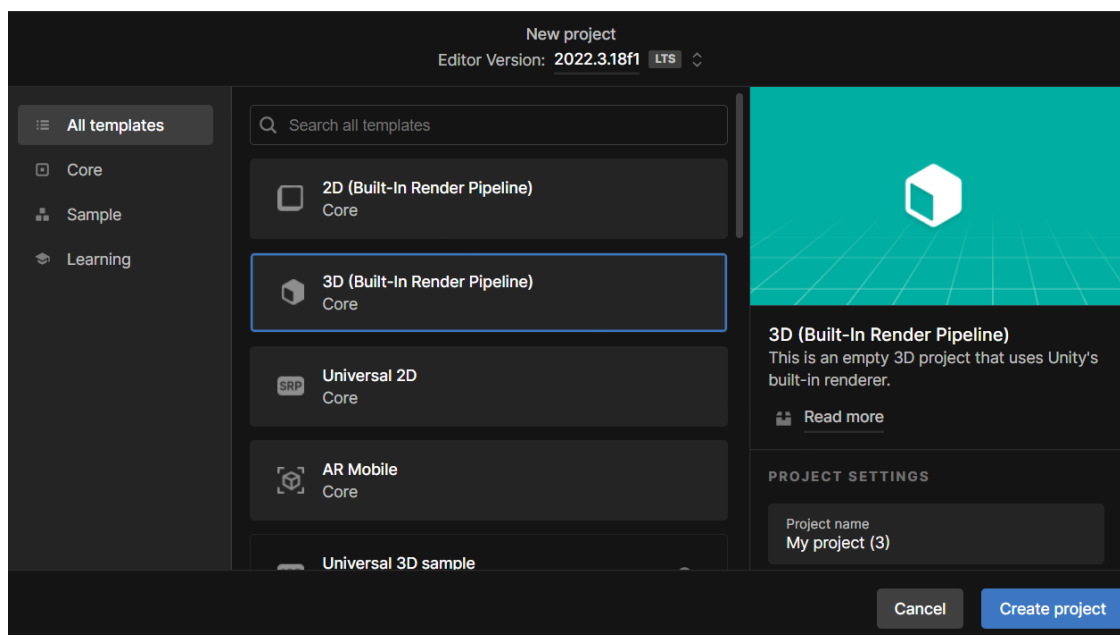


Figure 12. Create a New Project in Unity Hub.

The newly created project will launch, displaying a sample 3D scene as shown in Figure 13 that includes a pre-configured main camera. A Sample Scene acts as a container for the entire application, where the building blocks of the application will be placed. The building block includes objects, like

UI elements and targets. The application will be designed piece by piece in these blocks, by adding elements to make it interactive. (Unity Technologies.)

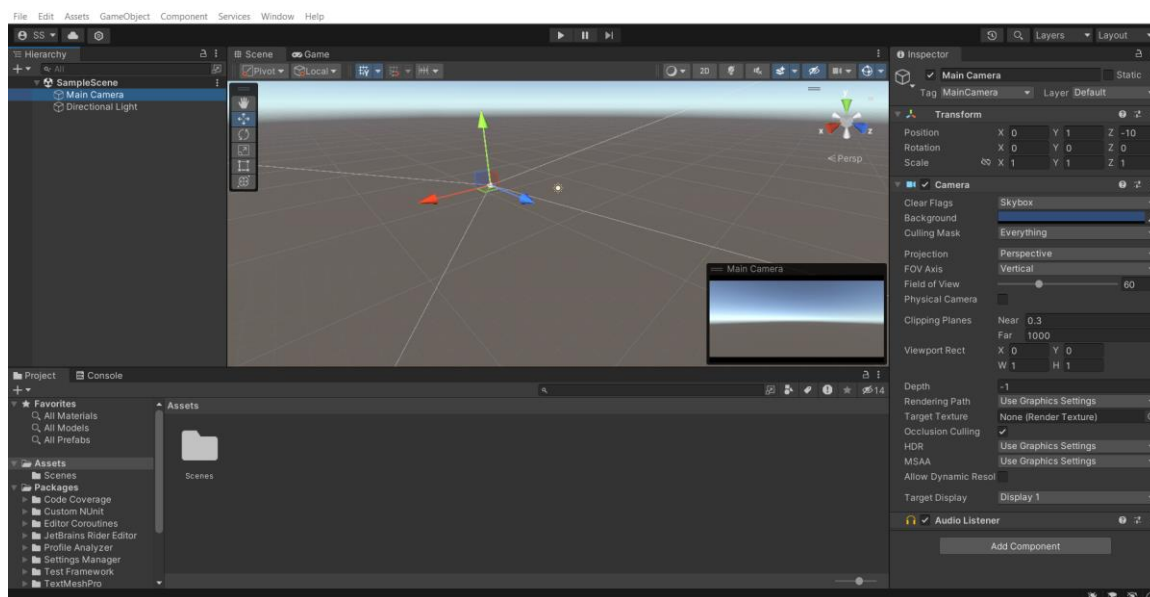


Figure 13. Sample Scene.

To create an AR application, the computer needs advanced functionality to recognize images, objects, and spaces. Vuforia Engine is a software development kit (SDK) for creating Augmented Reality apps and helping configure the app developed in Unity to interact with the real world. It supports AR app development for Android, iOS, Magic Leap, and Universal Windows Platform devices. The Vuforia Engine Extension is a Unity Asset Package that can be imported into a Unity project. (Vuforia.)

To integrate the Vuforia Engine Extension into the Unity project, start by downloading the installer from the Vuforia Developer Portal, as illustrated in Figure 14.

Release Version

10.22

Apply

By downloading the Vuforia Engine SDK, Samples and Tools, you agree to the [developer agreement](#).

Vuforia Engine 10.22

Use Vuforia Engine to build Augmented Reality Android, iOS, and UWP applications for mobile devices and AR glasses. Apps can be built with Unity, Android Studio, Xcode, and Visual Studio. Vuforia Engine can be easily imported into Unity by downloading and double-clicking the .unitypackage below.


 [Add Vuforia Engine to a Unity Project or upgrade to the latest version](#)
 add-vuforia-package-10-22-5.unitypackage (139.70 MB)
 MD5: 7ad684e0dc767b3c946249df0b44043c

Figure 14. Vuforia Package Download.

Import the downloaded Unity Package from the menu in Unity Assets -> Import Package -> Custom Package in the project. Import unity package window shown in Figure 15 will appear on the screen

and press import. Importing this package will run a script that automatically installs or upgrades the Vuforia Engine SDK in the project, ensuring the project has the latest functionalities.

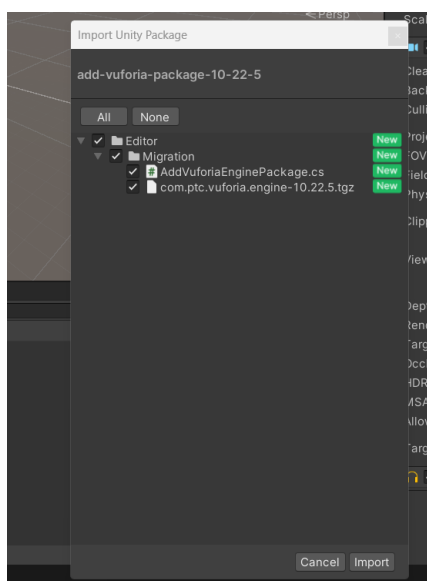


Figure 15. Import Unity Package.

After importing and updating all the necessary packages, Vuforia Engine will appear in the Game Object. A GameObject is the individual building block within a Scene. It represents any element displayed or interacted with in the application. Each GameObject can have properties like position, rotation, scale, and behaviors defined by scripts. (Unity Technologies.)

Go to the menu Game Object -> Vuforia Engine -> AR Camera and add an AR camera to the project. The Vuforia AR Camera captures the real world through a device and overlays virtual elements on top, creating AR. It uses Vuforia's technology like plane detection and tracking to seamlessly blend digital objects into the real environment. The main camera for this project can be deleted since the AR camera handles both real and virtual worlds. (Unity Technologies.)

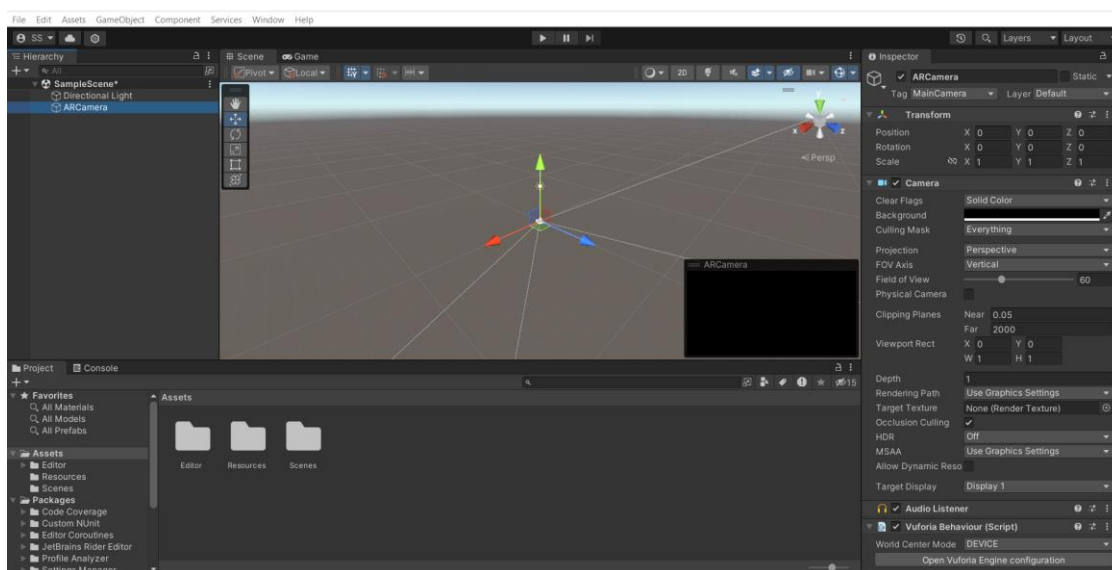


Figure 16. Add AR Camera.

The AR Camera like in Figure 16 will be part of a GameObject in the Scene after adding the Vuforia AR Camera.

To enable Vuforia Engine features and unlock the full potential of the application, the project needs a Vuforia App license key (Vuforia). By selecting the AR Camera, Open Vuforia Configuration from the Inspector window which can be found on the right side of the project as shown in Figure 17.

The Inspector window is the central hub for viewing and editing the properties of objects within the Unity project. It acts like a control panel for each element in the Scene. (Unity Technologies.)

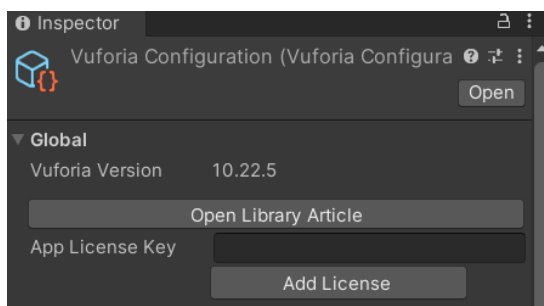


Figure 17. Vuforia Configuration.

To get the App license key, the user needs to have a Vuforia account. Head to the Vuforia developer portal, sign up for a free account and create a basic license as shown in Figure 18.

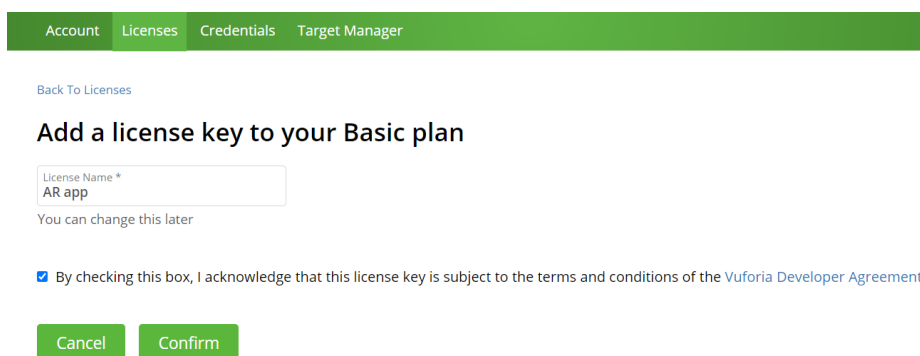


Figure 18. Create a License Key.

Access the created Vuforia License Key copy the key and paste it into the Unity project Vuforia configuration. Unity will automatically save the App license key as shown in Figure 19.

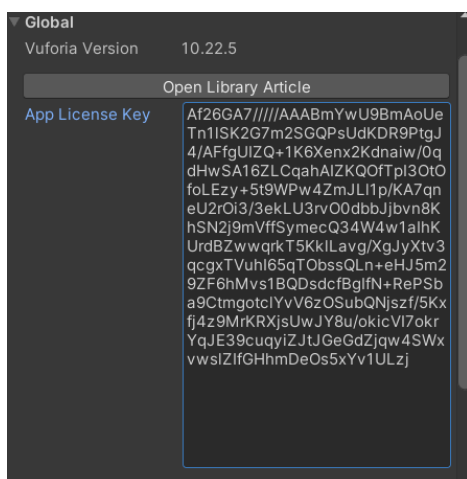


Figure 19. Add App License Key.

3.3.2 3D Scan

For the AR application to interact with the IoT device, needs a bridge between the real and virtual worlds. Model Targets empower the AR applications to identify and track specific objects in the real world. This can be achieved by using the object's 3D scan as a unique identifier. (Vuforia.)

Model Targets can be created from captures of real-life objects with 3D scanning technology or with photogrammetry using compositions of images. Polycam is a mobile application designed for creating 3D scans using photogrammetry, a technique that utilizes multiple photographs from different angles. By capturing photos of the object, Polycam processes these images to reconstruct a 3D model.

Capturing a good photo model in Polycam requires taking many photos (minimum 11) of the object from all sides, above, below, and with close-ups for detailed areas. Aim for at least half of each photo to overlap with the previous one, and make sure the object fills most of the camera frame when moving around it. The more perspectives captured, the better the final model will be. (Polycam.)

In this project, it was attempted to capture a 3D model of an AC fan prototype (see Figure 20).

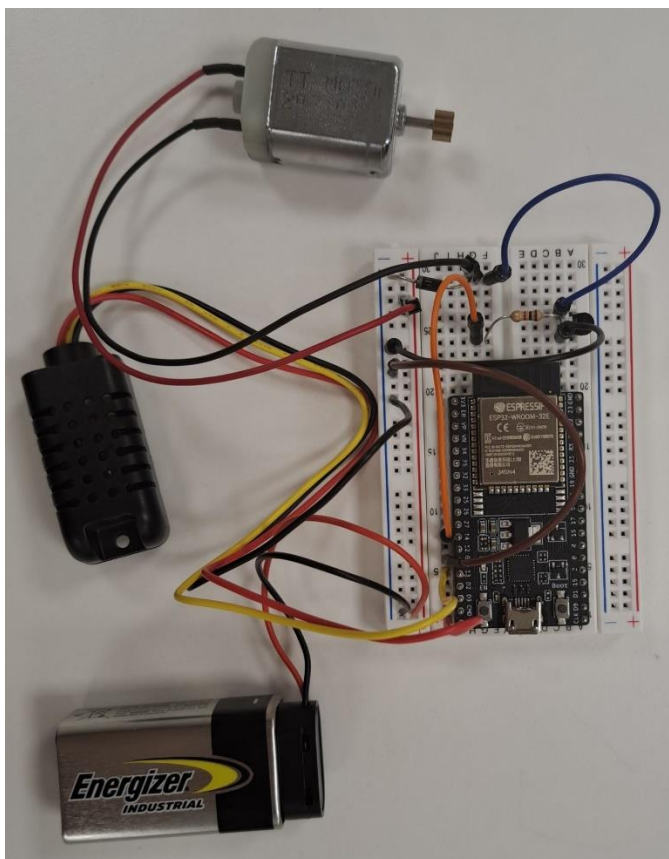


Figure 20. AC Fan Prototype.

The first attempt at capturing a 3D model wasn't perfect. The wires connecting the parts (like the temperature sensor, motor, and battery) were tricky for the scanner. They overlapped and made it hard to see each part. Because of this, some of the scans blended in with the background, as shown in Figure 21.

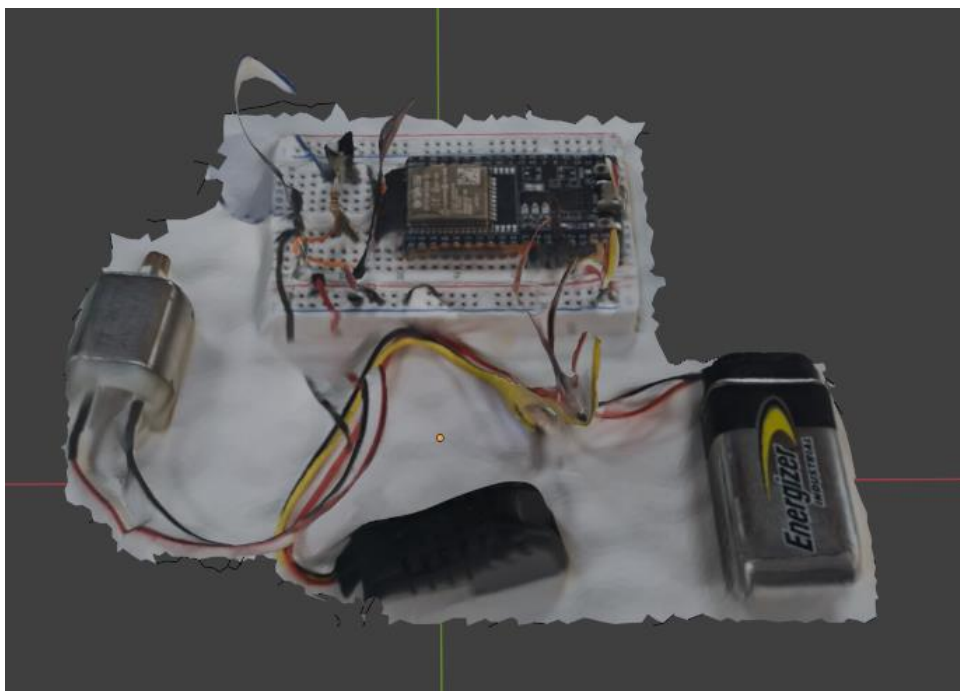


Figure 21. First 3D Scan With Polycam.

To improve the 3D scan by eliminating the wire clutter that caused issues in the first attempt, a standard breadboard was replaced with an Adafruit Perma-Proto PCB (printed circuit board). The Perma-Proto PCB is designed to replace messy breadboards for electronics prototyping, it combines the ease of use of a breadboard with the stability and durability of a PCB, making it easier to transfer circuits from prototyping to final projects (Adafruit).

This will let solder the transistors, resistors, diodes, and ESP32-WROOM-32 directly onto the PCB, removing messy wires. Also temporarily removed the battery, temperature sensor, and DC motor for the scan itself, focusing solely on capturing the core components for a cleaner and more defined 3D model as shown in Figure 22.



Figure 22. Prototype with PCB.

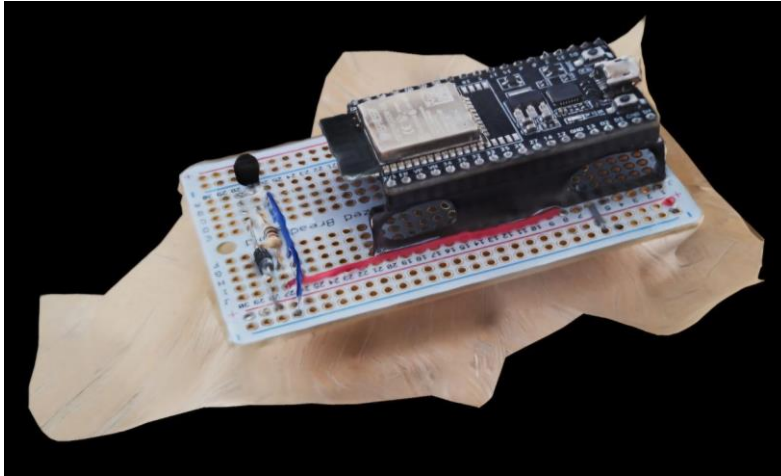


Figure 23. Second 3D Scan.

The second 3D scan (Figure 23) shows a significant improvement in clarity and accuracy compared to the first attempt. However, there's still a slight challenge in capturing the sides of the ESP32-WROOM-32. This is due to the IC (Integrated Circuit) socket used for soldering purposes. The material of the socket was causing reflections that made it difficult for the scanner to capture the board's sides effectively.

IC sockets act like little chip holders for circuit boards. They simplify board design by allowing easy removal and replacement of chips, perfect for repairs, upgrades, or experimenting without permanent soldering. (TE Connectivity.)

To avoid the reflection issue, the socket sides were covered with masking tape, leading to improved results. However, the model's alignment still requires minor adjustments. Blender, a free 3D software, was used to refine the final model and remove any unwanted surroundings (Blender Foundation).

Export the processed scan in Polycam to a selected file type (OBJ, FBX, etc.) and import the same file in Blender. Crop out any unwanted areas and align them in a straight line. Finally, adjust the model's smoothness in Shading Mode to get the perfect final texture (Polycam).

As shown in Figure 24, a precise 3D model was achieved.



Figure 24. Last 3D Scan.

3.3.3 Model Target Generator

The Model Target Generator (MTG) acts as a bridge between the 3D model scans and the Vuforia Engine, unlocking powerful Augmented Reality (AR) experiences. MTG analyzes the model to determine if its features are suitable for effective recognition in AR. It considers factors like surface details and complexity to ensure successful tracking. (Vuforia.)

Download and run the MTG installer from the Vuforia Developer Portal. Choose installation options and location. The following window shown in Figure 25 will appear and from the dashboard, it is possible to create the model target.

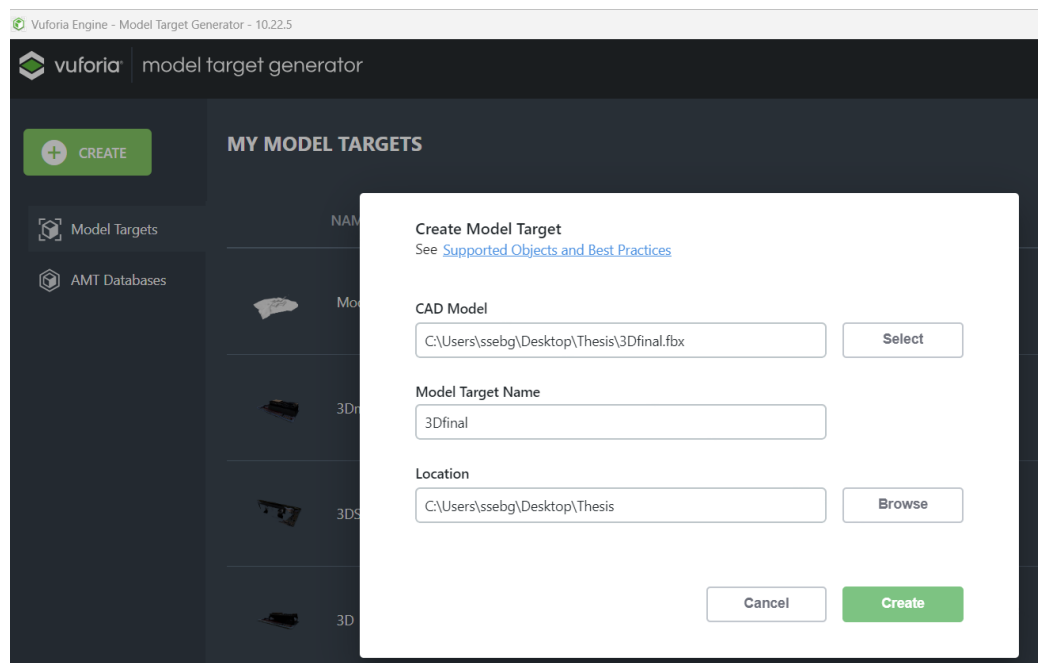


Figure 25. Create a Model Target.

Within MTG, navigate to the file selection window and browse to the correct location where the exported model file from the blender is saved. Selecting the wrong location will lead to errors. After successfully selecting the model file, it will appear in the MTG's 3D viewer as Shown in Figure 26.

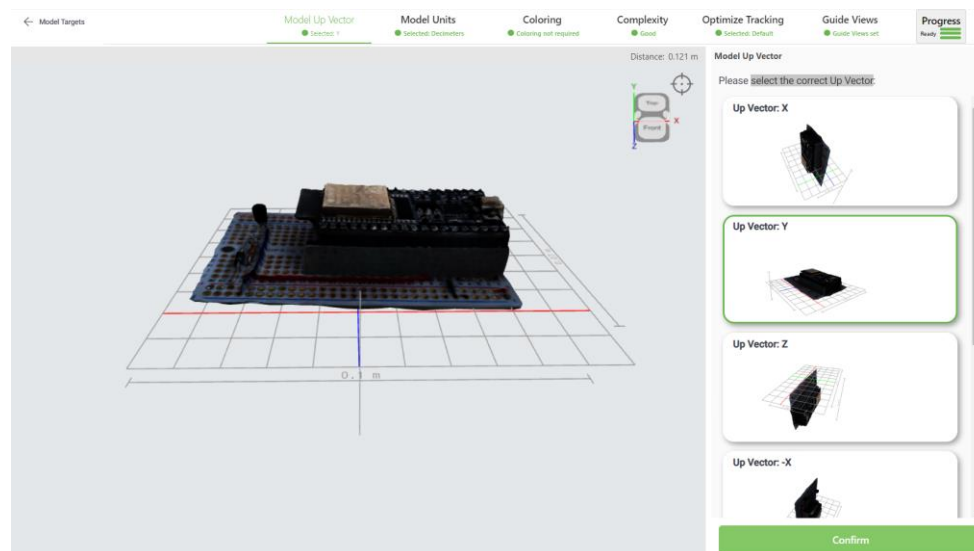


Figure 26. Model Up Vector.

The first step involves confirming the Model-up Vector. This essentially defines the model's orientation in the real world (Vuforia). MTG will provide different views to choose from the correct up vector that aligns with how the physical object stands upright.

The 3D model doesn't have pre-defined size information. To ensure the AR experience accurately reflects reality, MTG will set the Model Target's size to match the physical object. Choose the unit system that matches the physical object (meters, centimeters, inches, etc.) as shown in Figure 27. This can be obtained by measuring the size of the object in real life.

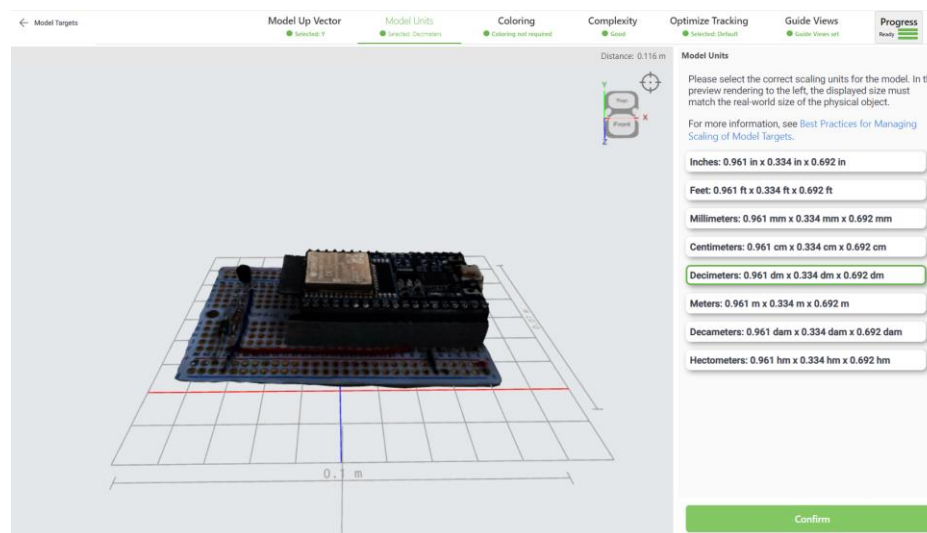


Figure 27. Model Units.

The Model Target Generator (MTG) offers automatic coloring for imported models with multiple parts or lack texture. Adding color to these models can significantly enhance tracking performance within the Vuforia Engine. Each part receives a unique color, providing the engine with more visual details to distinguish and track the object effectively in AR experiences. Models with existing textures already provide visual details for tracking. In these cases, coloring is unnecessary. If the texture accurately represents the physical object's appearance, select the "Realistic Appearance" setting before proceeding to the next step. (Vuforia.)

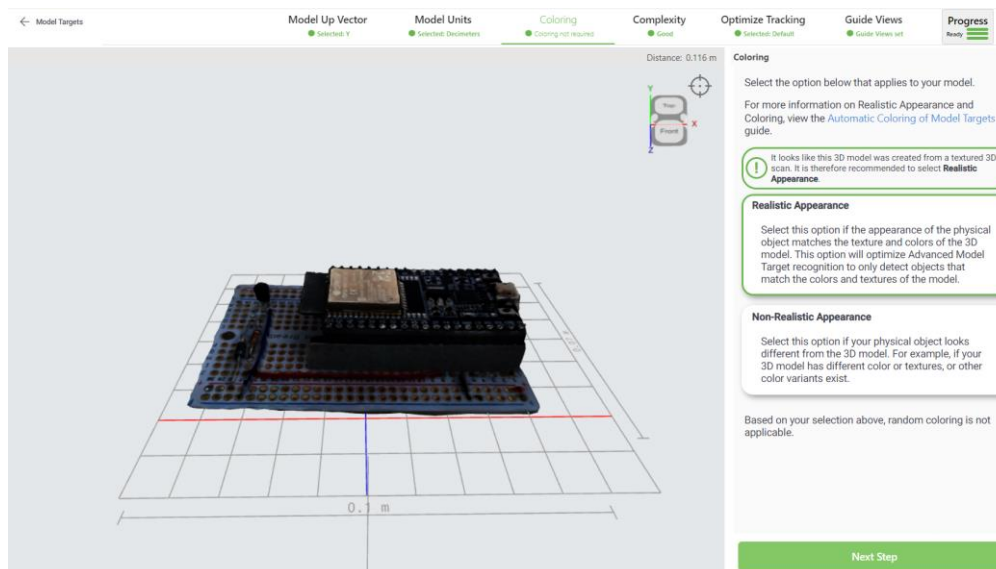


Figure 28. Coloring

The next tab measures the number of vertices (building blocks) and parts in the model. A complex model can slow down the AR app's performance. If the model meets the recommended specifications (maximum of 400,000 polygons or triangles, 20 parts, and 5 textures), this section can be skipped. If the model exceeds the recommended complexity, MTG will display a warning. The Model Target Generator (MTG) offers a tool to simplify the model directly before creating the AR target. This simplifies the model's structure, potentially improving performance. (Vuforia.)

For this project, model the complexity was in good status already.

The Optimize Tracking tab deals with how the Model Target is tracked in AR, including two options. Default mode works well for a broad range of model objects and is a good starting point. Low Feature Object mode is suitable for objects like cars, or those that are reflective, lack textures, and have smooth surfaces. These objects might have fewer visual details for the AR engine to track effectively. (Vuforia.)

As shown in Figure 29, the default mode works well with this model.

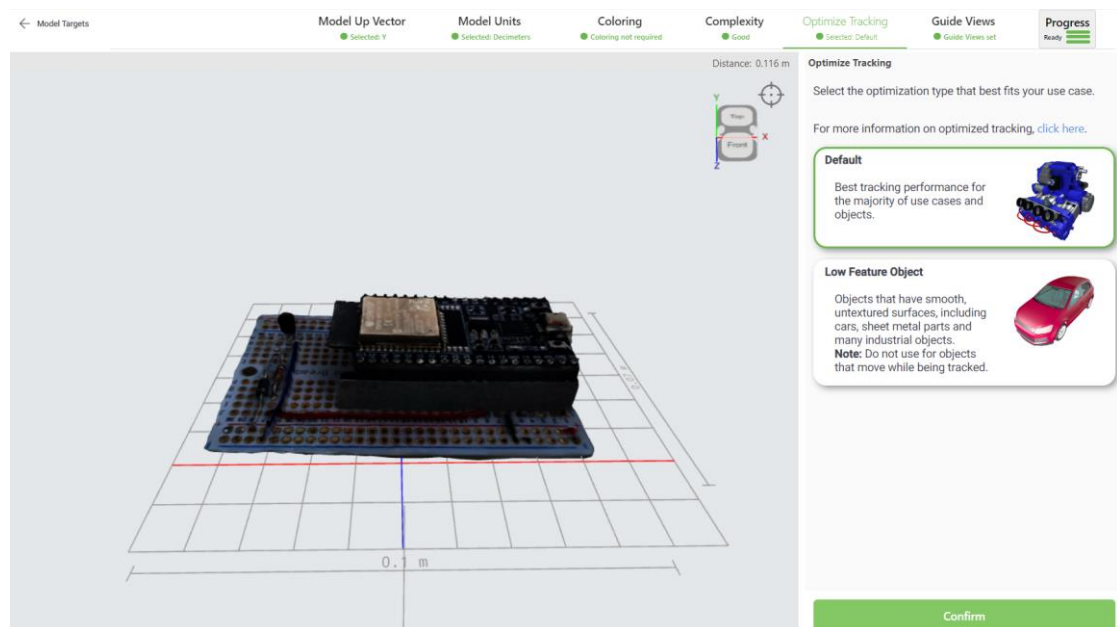


Figure 29. Optimize Tracking.

The next section sets up Guide Views, which are crucial for guiding users to initiate AR tracking of the Model Target. Guide Views: Guide Views are simple visual guides that appear as an outline of the model during the AR session. It needs to physically match the physical object to this outline to start the AR experience. Advanced Views are more complex recognition ranges (from one position up to a 360-degree detection range). Once defined, the AR experience automatically activates when the model is detected from any angle within those ranges. Setting up Advanced Views involves cloud training. (Vuforia.)

Before creating the Guide View, define the field of view based on how users hold their device digital eyewear (designed for smart glasses), landscape, and portrait.

The AC fan in Figure 30 has a fixed position. To create a user-friendly AR experience, Landscape View and Advanced Views for flexibility are chosen. 360° Dome Detection option within Advanced

Views ensures users can approach and detect the AC fan from any side, maximizing convenience during the AR experience. These settings, a Landscape view, and a 360° Dome Advanced View, create an intuitive and adaptable AR experience for users interacting with the stationary AC fan.

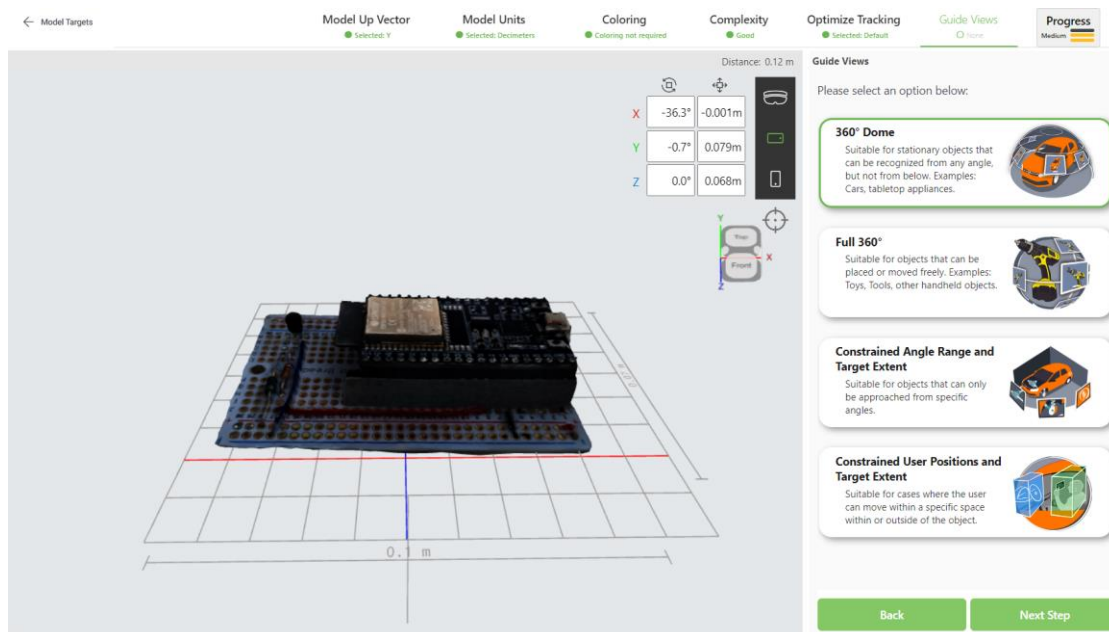


Figure 30. Advanced Guide View.

After defining an Advanced View for the Model Target, MTG initiates the training processes by creating a database as shown in Figure 31. A training session refers to the process of creating a database entry for the Model Target that allows the Vuforia Engine to effectively recognize and track it in AR experiences. The 3D structure and the defined Advanced View of the Model Target are input data for the training. (Vuforia.)

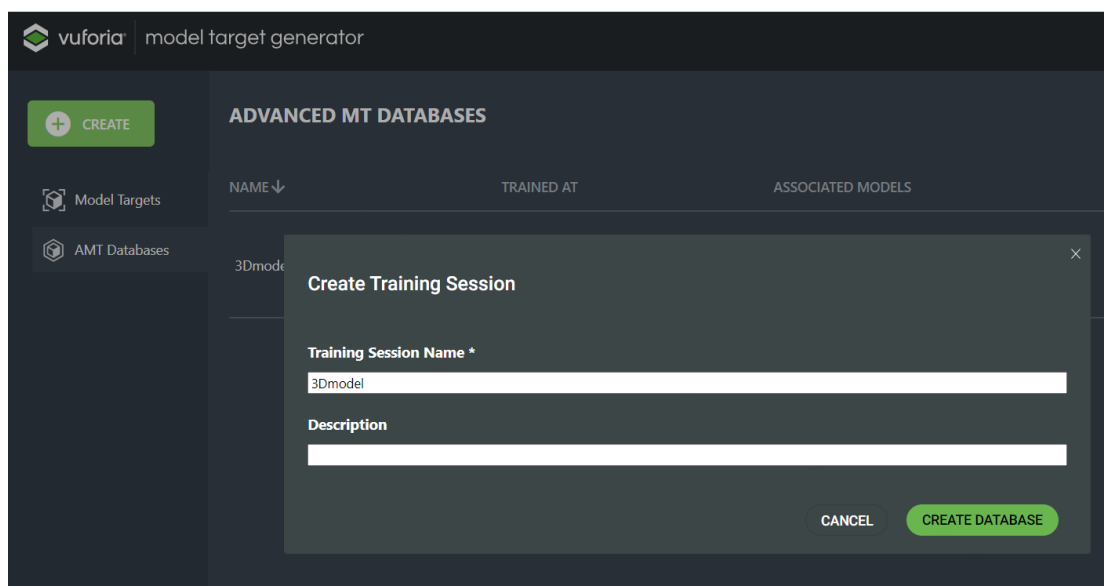


Figure 31. Creating Training Session.

Add the model to be copied in the database from the list shown in Figure 32 and start the training. If the model has multiple targets, it is possible to select more than one model for the training. Training the model might take time depending on the model.

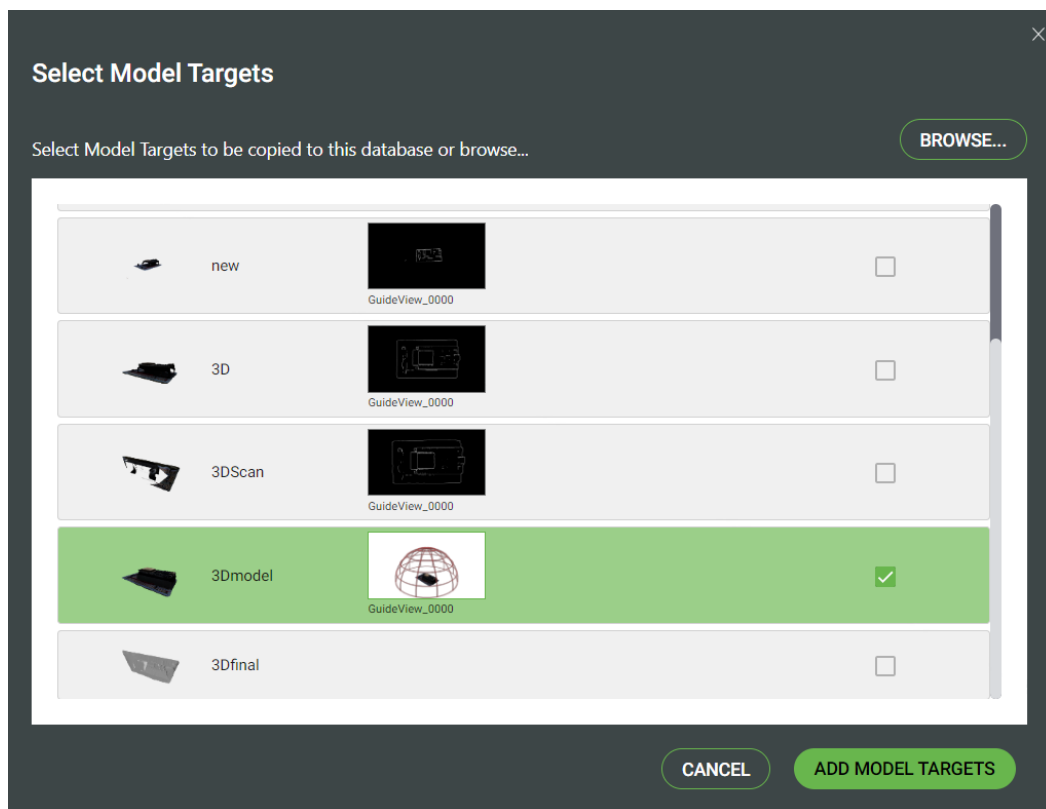


Figure 32. Creating Database.

After completing the training, the Model Target Generator will create a database that can be exported.

Exporting the database will save the dataset file folder in the computer, containing the Unity package that can be added to the project in Unity.

3.4 AR + IoT Integration

3.4.1 Integrating Model Target

Once successfully exported the Model Target package for the Model Target Generator (MTG), it's time to bring it into the Unity project for AR development. Open the Unity project, navigate to the Assets menu, and select Import Package -> Custom Package. Locate the exported Model Target database file(unity package) obtained from MTG and click import to bring the database into the Unity project.

In the Unity Hierarchy window, right-click on an empty space and select Vuforia Engine -> Model Target. The Hierarchy window is a fundamental tool for managing all the objects and components that make up the AR experience. It acts like a family tree, showing how objects are related to each other. (Unity Technologies.) This action adds a new Model Target object to the Scene as shown in Figure 33.

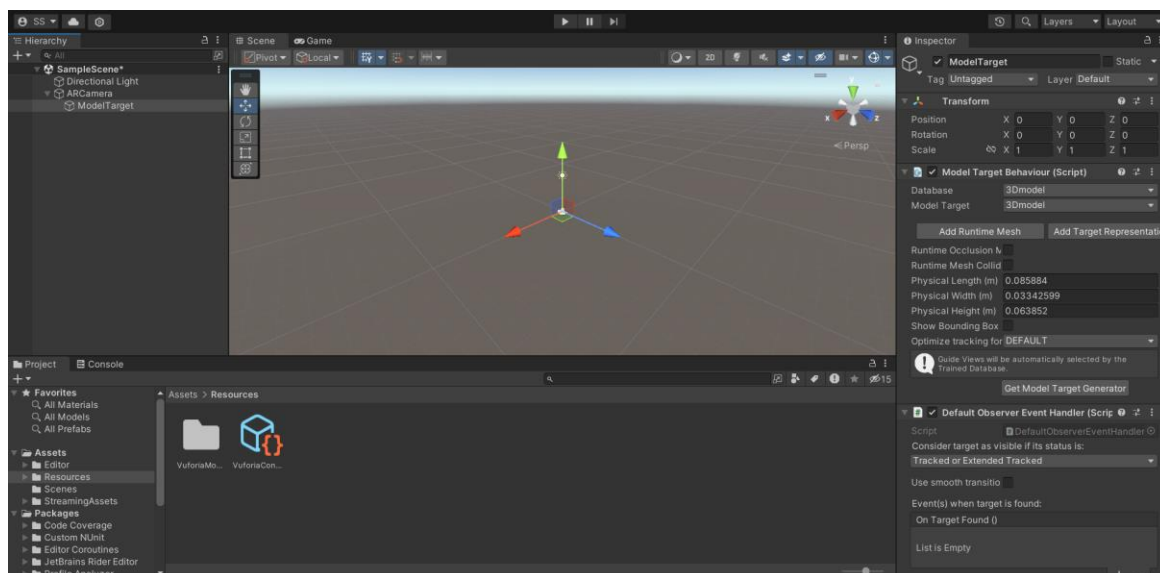


Figure 33. Add Model Target in Unity.

The AR Camera captures the real world through your device's camera. It's crucial to have the AR Camera at the top of the Hierarchy. When the AR Engine recognizes the physical object through the camera, the Model Target becomes active. Any UI(User Interface) elements added to appear on the screen should be placed as child objects under the Model Target in the Hierarchy. UI elements include any visual elements like buttons, menus, and text. (Unity Technologies.) By having UI elements nested under the Model Target, only become active and visible when the AR Engine recognizes the physical object. This ensures the UI overlays appear only when relevant to the user's AR experience.

For this project buttons, a text field, and imported 3D elements have been added to make the AR application interactive as shown in Figure 34. In the Hierarchy window, right-click on an empty space and select UI -> Canvas. This creates a canvas object, which acts as the foundation for the UI elements. By default, the canvas is placed under the Main Camera in the Hierarchy. Drag the Canvas object under the Model Target in the Hierarchy. This ensures the UI elements only appear when the Model Target is active (after a successful scan).

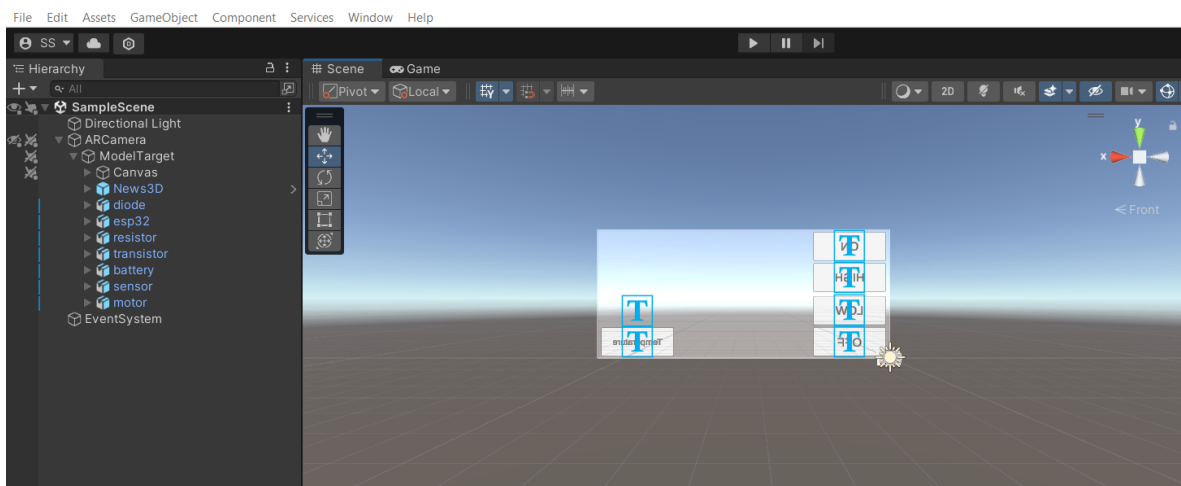


Figure 34. UI Canvas Layout.

With the Canvas selected, navigate to the Inspector window (usually on the right side). This window allows to modify properties of the selected object. Choose the desired UI elements from the menu. Options like "Button" and "Text" are suitable for this project. Buttons for controlling fan speed (on, off, high, low) and adjusting load temperature have been added to the Canvas. A text field has been included to display the current temperature. It is possible to adjust the position and size of each UI element within the Canvas directly in the scene view to match the project's needs.

3D pinpoint model downloaded from the internet to locate the components of the AC fan in the Model Target. Locate the downloaded 3D model file (e.g., FBX) and drag and drop the 3D model file from the project folder into the Hierarchy window. Similar to the Canvas, drag and drop the imported 3D model under the Model Target in the Hierarchy. This ensures it appears only after the Model Target is recognized.

Run the Unity project in Play Mode to test how the UI elements and 3D model appear after the Model Target scan. Make sure to adjust their positions, sizes, and behaviors in the Inspector window to achieve the desired user experience as shown in Figure 35.

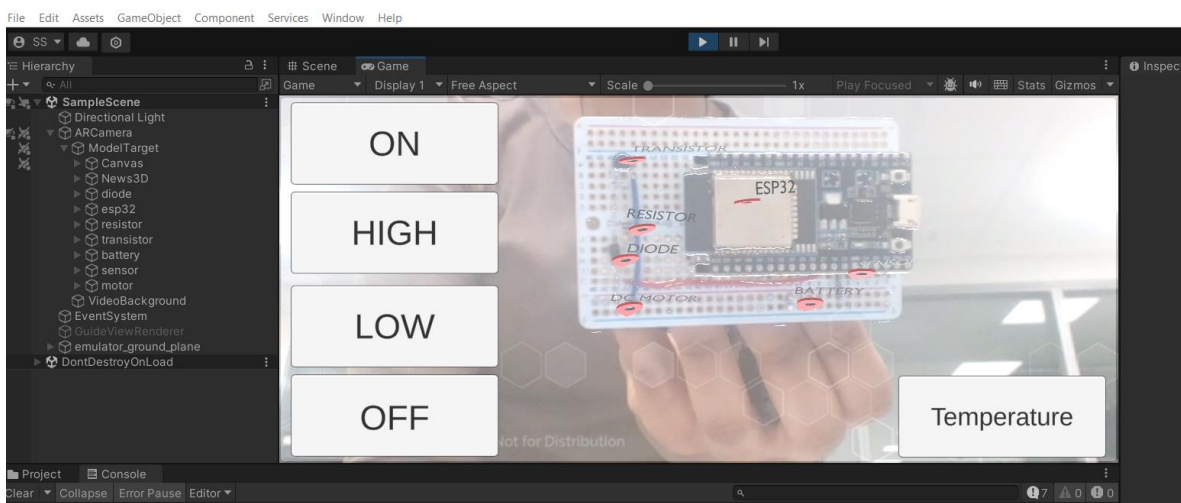


Figure 35. Model Target Test.

3.4.2 Connecting IoT Device and AR Application

In this stage, the goal is to establish communication between the IoT device and the AR application. This can be done by connecting the AR application to the server developed for the AC fan. This allows the app to exchange data with the server, potentially fetching real-time information or sending control commands.

Unity uses C# scripts to define the behavior of objects within your AR experience (Unity Technologies). To create a script, navigate to the Assets menu, Assets -> Create -> C#Script. This will create a new C# script file in the project's assets folder. In this case, the scripts "temperature" and "speed" as shown in Figure 36 which suggests they'll handle functionalities related to temperature reading and fan control respectively.

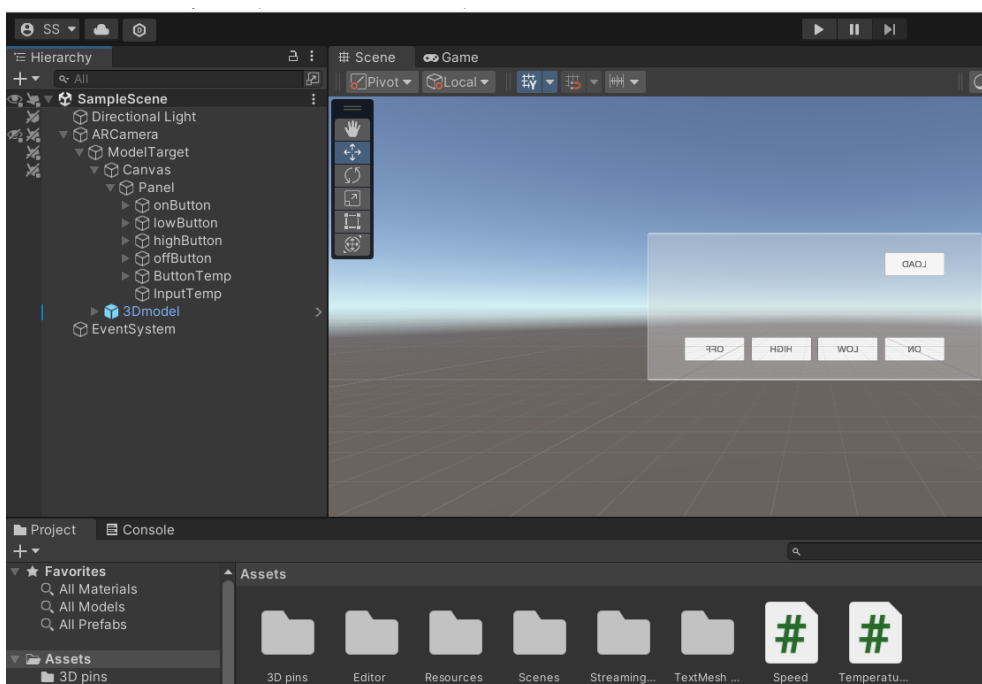


Figure 36. Adding Script.

Creating scripts in Unity only provides empty classes. To achieve a temperature reading on a button click, need to write code within the "temperature" script.

```

using System;
using System.Collections;
using TMPro;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;

Unity Script (2 asset references) | 0 references
public class Temperature : MonoBehaviour
{
    public TextMeshProUGUI Temp;
    public Button ButtonTemp;

    private string serverURL = "http://192.168.235.17/temperature";

    Unity Message | 0 references
    public void Start()
    {
        Temp = GameObject.Find("InputTemp").GetComponent<TextMeshProUGUI>();
        ButtonTemp.onClick.AddListener(OnButtonClicked);
    }
    1 reference
    public void OnButtonClicked()
    {
        StartCoroutine(GetTemperature());
    }
}

```

Figure 37. Temperature Script 1.

In the Temperature class shown in Figure 37, Temp is a text box called "InputTemp" where the temperature will be displayed and there's also a button named "ButtonTemp" to fetch the temperature. Finally, the web address (URL) of a server that provides temperature data. When the app starts, the script finds the "InputTemp" text box and remembers it. It also sets up the "ButtonTemp" button so that whenever it is clicked, a special function called "OnButtonClicked" runs. When "ButtonTemp" is pressed, the "OnButtonClicked" function jumps into action.

```

1 reference
IEnumerator GetTemperature()
{
    using (UnityWebRequest request = UnityWebRequest.Get(serverURL))
    {
        yield return request.SendWebRequest();

        if (request.result == UnityWebRequest.Result.ConnectionError ||
            request.result == UnityWebRequest.Result.ProtocolError)
        {
            Temp.text = "Error: " + request.error;
        }
        else
        {
            string jsonString = request.downloadHandler.text;
            TemperatureData temperatureData = JsonUtility.FromJson<TemperatureData>(jsonString);
            Temp.text = "Temperature: " + temperatureData.temperature.ToString() + " °C ";
        }
    }
}

[Serializable]
2 references
public class TemperatureData
{
    public float temperature;
}
}

```

Figure 38. Temperature Script 2.

The code in Figure 38, retrieves temperature data from the server. It creates a messenger (UnityWebRequest) to send a GET request to the server at a stored web address (serverURL). The code pauses briefly waiting for the server's reply. If there's an error reaching the server or receiving a

proper response, an error message is shown. Otherwise, the code decodes the server's message in JSON format to extract the temperature value and displays it on the "Temp" text field in the AR app's user interface.

After saving the code, drag and drop the Temperature script to the ButtonTemp and InputTemp fields in the GameObject. The script will appear in the inspector with both fields, labelled Temp and ButtonTemp, which need to be linked to the actual UI element.

Drag the "InputTemp" text field (where the temperature will be shown) from the Scene and drop it onto the "Temp" section in the Inspector window. This connects the script to the text field. Repeat the drag and drop process, but this time with the "ButtonTemp" button onto the "ButtonTemp" section in the Inspector. This connects the script to the button.

Find the "Button" component attached to the "ButtonTemp" button in the Inspector window. Click the "+" button next to the "On Click()" list. This list shows functions that can be triggered when the button is pressed. Drag the GameObject (the object that has the script attached) into the "Object" field that appeared. Finally, from the dropdown menu that pops up, select the "OnClickClicked" function in the script that handles the temperature fetching process as shown in Figure 39.

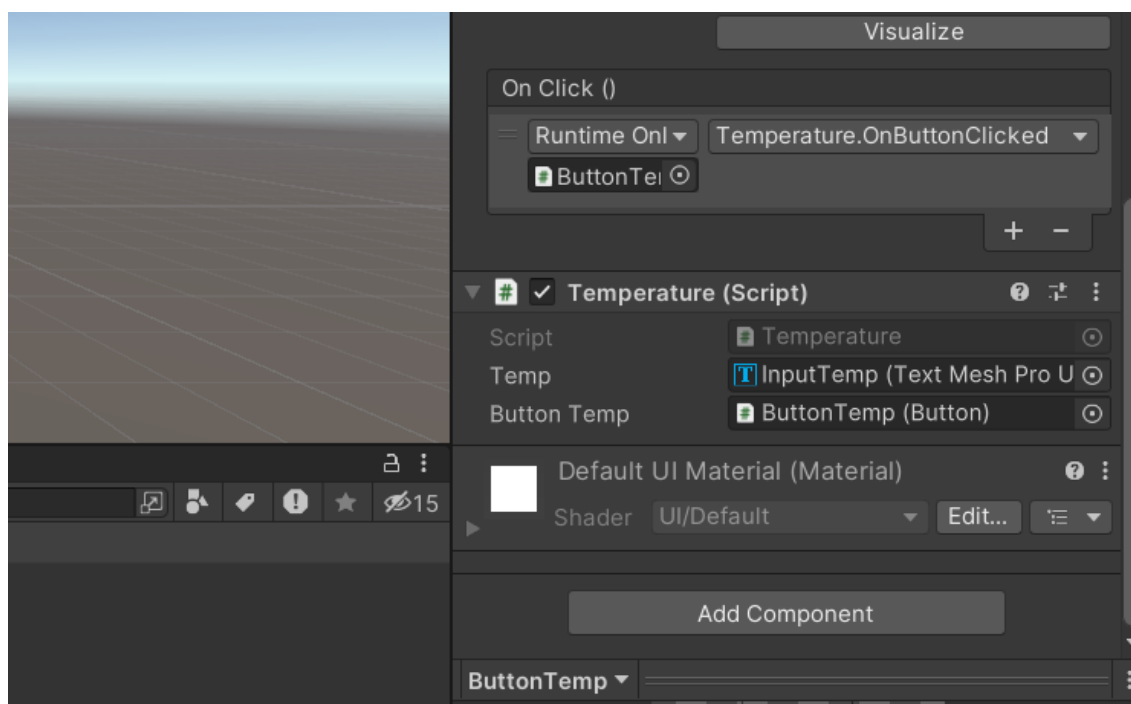


Figure 39. Setting Up Temperature Script.

The Speed script provides a basic framework for controlling a motor's speed through button presses. The script communicates with a server to turn the motor on/off and set its speed.

```

using System.Collections;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;
Unity Script (4 asset references) | 0 references
public class Speed: MonoBehaviour
{
    public Button onButton;
    public Button offButton;
    public Button highButton;
    public Button lowButton;

    private string serverURL = "http://192.168.235.17";

    Unity Message | 0 references
    public void Awake()
    {
        if (onButton == null || offButton == null || highButton == null || lowButton == null)
        {
            Debug.LogError("One or more buttons are not assigned.");
            return;
        }
        onButton.onClick.AddListener(TurnMotorOn);
        offButton.onClick.AddListener(TurnMotorOff);
        highButton.onClick.AddListener(SetHighSpeed);
        lowButton.onClick.AddListener(SetLowSpeed);
    }
}

```

Figure 40. Speed Script 1

The Speed class, as shown in Figure 40, has button references with different functions, motor on, motor off, and setting motor speed to high value and low value. Each button is attached to an event listener which calls a different function when pressed. The serverURL stores the base URL of the server controlling the motor.

```

1 reference
public void TurnMotorOn()
{
    StartCoroutine(SendRequest("speed=150"));
}

1 reference
public void TurnMotorOff()
{
    StartCoroutine(SendRequest("speed=0"));
}

1 reference
public void SetHighSpeed()
{
    StartCoroutine(SendRequest("speed=255"));
}

1 reference
public void SetLowSpeed()
{
    StartCoroutine(SendRequest("speed=100"));
}

4 references
IEnumerator SendRequest(string requestData)
{
    using (UnityWebRequest request = UnityWebRequest.Get(serverURL + "/speed?" + requestData))
    {
        yield return request.SendWebRequest();

        if (request.result != UnityWebRequest.Result.Success)
        {
            Debug.LogError("Request failed: " + request.error);
        }
    }
}

```

Figure 41. Speed Script 2.

Motor control shown in the Figure 41, works by calling the following functions:

- TurnMotorOn()- sends a request with "speed=150" data (turns motor on at medium speed).
- TurnMotorOff()- sends a request with "speed=0" data (turns motor off).

- SetHighSpeed()- sends a request with "speed=255" data (sets high speed).
- SetLowSpeed()- sends a request with "speed=100" data (sets low speed).

All control methods utilize the SendRequest coroutine, take a string containing speed data (e.g., "speed=100"), and send a GET request to the constructed URL formed by combining the serverURL, and "/speed?".

Load the script Scene's hierarchy and navigate to the GameObject that contains the buttons that control the motor speed. In the Project window, find the "Speed" script and drag it directly onto the GameObject in the hierarchy. Once the script is attached, the Inspector window will show its details. Look for sections labelled with the names given, in this case, "onButton," "offButton," "highButton," and "lowButton". Drag and drop each button GameObject from the hierarchy onto the matching field in the Inspector window for the Speed script. This assigns the actual buttons to the script's references, as shown in Figure 42.

In the Inspector window with the Speed script selected, look for the "Button" component attached to each button GameObject linked in the previous step. Click the "+" button next to the "On Click()" list for each button component. This list shows functions that can be triggered when the button is pressed. Drag the GameObject with the attached Speed script into the "Object" field that appears next to the "On Click()" list. From the dropdown menu that pops up, select the specific function in the Speed script that corresponds to the button's action.

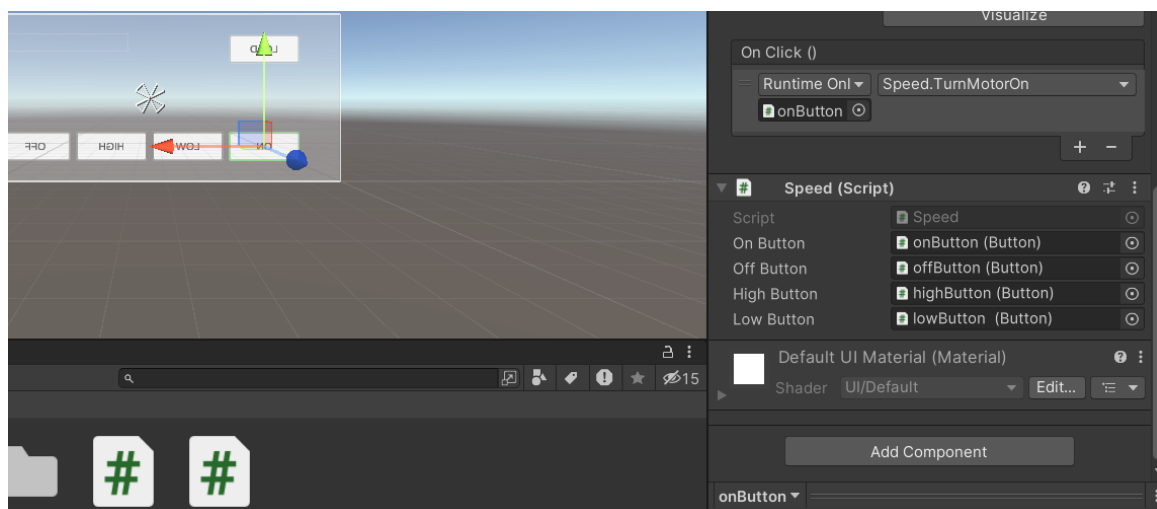


Figure 42. Setting Up Speed Script.

By following the above steps, it is possible to create a connection between the button presses and the Speed script. When a button is pressed, the "On Click()" event triggers the appropriate function in the script, which then sends a command to the server via the stored serverURL to control the motor's speed or state.

3.4.3 Deploying and Building Mobile Application

The final stage will be deploying and building the application on a mobile device. Before building the app for Android, some settings need to be adjusted to make sure it works properly on Android devices. Go to the Edit menu at the top of the Unity editor and click on Project Settings. This opens a window with various settings for your project. Navigate to Player, this section controls how the AR application will be built for different platforms. Under Player, select Android. This shows all the settings specific to Android builds.

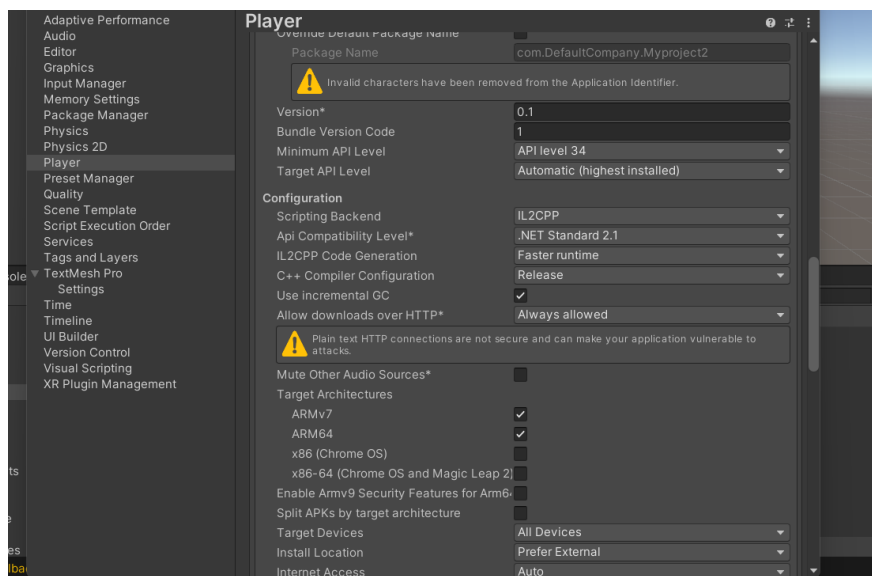


Figure 43. Android Settings.

Configuring Android Settings as shown in Figure 43:

- **Minimum API Level:** This setting determines which versions of Android the app can run on. Choose a level that supports the device targeted.
- **Target API Level:** This is ideally the latest Android version most devices have. It allows the app to use the newest features, but make sure it's still compatible with the minimum level.
- **Scripting Backend:** Choose IL2CPP. This is a way for Unity to handle the AR code, making it work on more devices and run faster (Unity Technologies).
- **Allow Downloads Over HTTP:** Since the app fetches data from a server, enable this option.
- **Target Architecture:** Choose the architecture (32-bit or 64-bit) that matches the devices targeted. This ensures your app runs smoothly on those specific phones.

Once successfully configured the settings, it's time to build the app. Go to the File menu, then Build Settings. Make sure the platform is set to Android. Click the Build and Run button as shown in Figure 44. Unity will create an Android app file that can be installed on a phone.

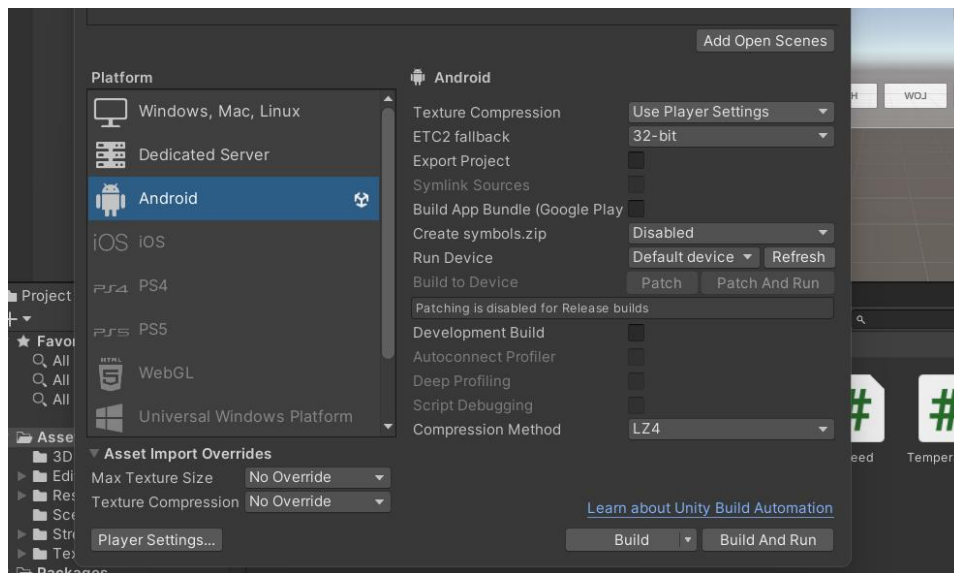


Figure 44. Build and Run Android App.

4 FINDINGS AND ANALYSIS

4.1 Result

The AR application was developed using Unity's real-time development platform and Vuforia Engine, taking a 3D scan for interfacing the virtual information. The application allowed users to interact with virtual controls overlaid in the real-world environment, enabling them to adjust parameters such as fan speed, monitor temperature readings in real-time, and visualize the components in detail.

The overall look of the application is shown in Figure 45.

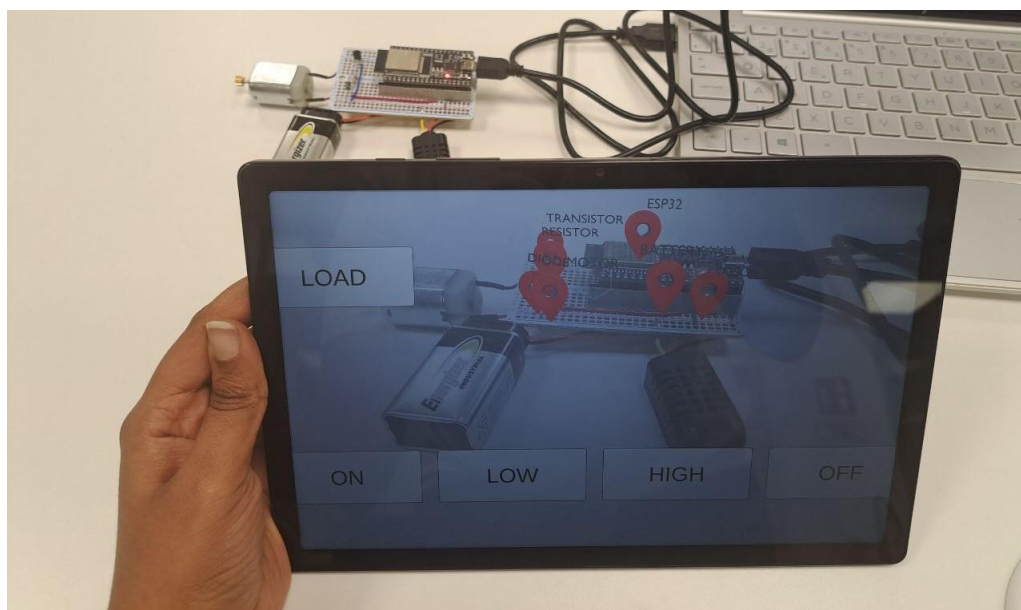


Figure 45. AR Mobile Application.

The Application locates the components of the AC fan with the label to provide information and visualize them easily as shown in Figure 46.

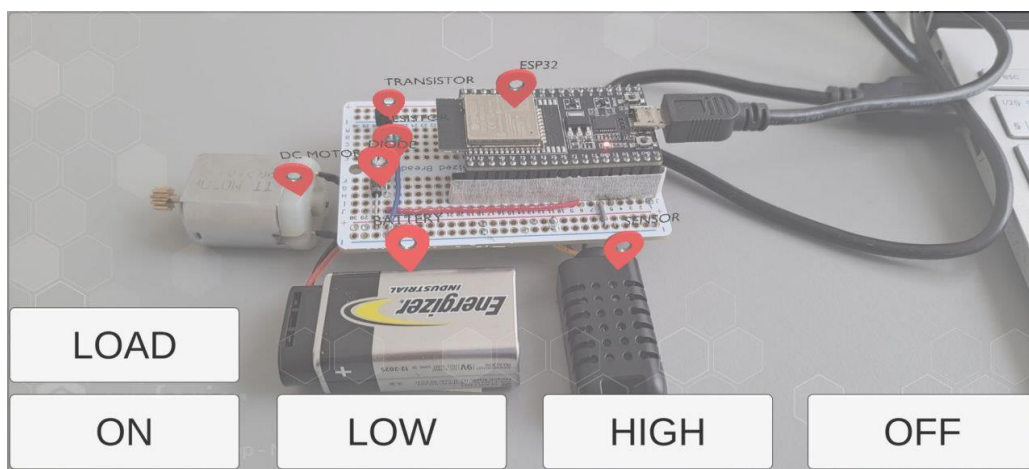


Figure 46. Components.

It also displays the real-time temperature reading every time the load button is pressed as shown in Figure 47.

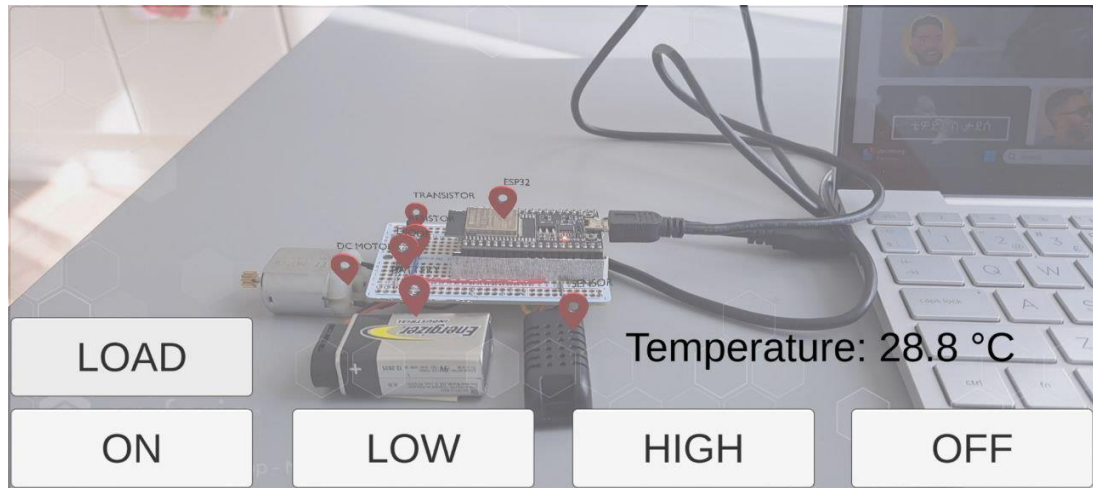


Figure 47. Load Temperature.

The AR-controlled AC application offers a range of benefits beyond simple convenience. It can be leveraged for repairs by visualizing internal components and diagnosing issues, streamlining the process, and boosting efficiency. Additionally, it optimizes energy consumption through real-time temperature monitoring, allowing for adjustments that achieve better energy savings. The app's ability to visualize the AC unit in place is also a valuable tool for planning layouts and ensuring proper airflow in new spaces.

The implementation of the AR application for controlling the AC fan yielded promising results. Users were able to intuitively manipulate virtual controls via the AR interface, effectively adjusting fan settings, monitoring temperature levels, and visualizing the components in detail. The real-time visualization of IoT data overlaid with AR visuals provided users with enhanced situational awareness, facilitating more informed decision-making.

4.2 Limitations

During the implementation phase, several limitations were encountered. Technical challenges related to the integration of AR technology with the IoT device included compatibility issues, latency in data transmission, and lack of resources. Integrating AR technology with the AC unit's IoT components was complex due to differences in software and hardware standards. Ensuring seamless communication between these systems required significant adjustments and troubleshooting. Additionally, real-time data transmission between the AC unit and the AR application faced latency problems, causing delays in updating the AR visualizations and affecting the accuracy and responsiveness of the application.

User experience aspects also needed further refinement. The initial design of the app was not intuitive, making it difficult for users to navigate and use it effectively. Users experienced delays and lag when interacting with the AR application, impacting overall functionality and satisfaction. Addressing these issues requires iterative testing, user feedback, and better resources to enhance compatibility, speed, and user interface design.

4.3 Future Implications

Despite these limitations, the successful implementation of the AR application for controlling the AC fan opens up promising avenues for future research and development. Further exploration is warranted to address technical challenges and enhance the stability of AR-IoT integration. Additionally, future iterations of the AR application could incorporate advanced features and functionalities to enrich the user experience and expand the scope of practical applications in diverse industries.

By addressing these challenges and leveraging the insights gained from this implementation, future advancements in AR-IoT integration hold the potential to revolutionize various sectors, from manufacturing and healthcare to retail and entertainment. The seamless convergence of AR and IoT technologies promises to redefine interactions and operations in the digital age, guiding in a new era of immersive and interconnected experiences.

5 CONCLUSION

The integration of Augmented Reality (AR) and the Internet of Things (IoT) holds immense potential to revolutionize interactions and operations across diverse sectors. Through the development and implementation of an AR application for controlling an AC fan, this thesis has demonstrated the feasibility and practical implications of leveraging AR technology as a user interface for IoT devices.

The successful implementation of the AR application showcased the intuitive nature of AR interfaces in controlling IoT devices, enhancing user experiences, and facilitating real-time monitoring and adjustment of device parameters. Furthermore, the visualization of IoT data overlaid with AR visuals provided users with enhanced situational awareness, contributing to more informed decision-making.

However, it is important to acknowledge the limitations and challenges encountered during the implementation process, including technical constraints and usability concerns. Addressing these challenges will be crucial for the widespread adoption and scalability of AR-IoT solutions in the future.

In conclusion, the successful implementation of the AR application for controlling the AC fan represents a significant step forward in exploiting the potential of AR-IoT integration. By critically examining the findings and offering insights into the broader implications of this technology fusion, this paper contributes to advancing knowledge in the field and paving the way for future advancements in AR-IoT integration.

REFERENCES

- Adafruit. Adafruit Perma-Proto Half-sized Breadboard PCB - Single. <https://www.adafruit.com/product/1609>. Accessed 4/2024.
- Amazon Web Services. What is IoT? <https://aws.amazon.com/what-is/iot/>. Accessed 1/2024.
- Adamska, Iwona 2023. The integration of IoT (Internet of Things) and augmented reality: A new paradigm in smart manufacturing. NSFlow Blog. 25/8/2023. <https://nsflow.com/blog/iot-and-ar>. Accessed 2/2024.
- Aosong Electronics. Product overview: AM2311A temperature and humidity sensor. <http://www.aosong.com/en/products-48.html>. Accessed 1/2024.
- AspenCore. Pulse Width Modulation. Electronic Tutorials Blog. <https://www.electronicstutorials.ws/blog/pulse-width-modulation.html>. Accessed 2/2024.
- Bluestone, PIM 2023. Augmented Reality in Retail: Enhancing Product Info for Smarter Choice. Bluestone PIM Blog. 8/8/2023. <https://www.bluestonepim.com/blog/ar-in-retail>. Accessed 3/2024.
- Blender Foundation. Blender. <https://www.blender.org/>. Accessed 4/2024.
- Carroll II, Brion 2023. The Intersection of IoT and Augmented Reality: Transforming Industries and Enhancing Experiences. LinkedIn. 24/6/2023. <https://www.linkedin.com/pulse/intersection-iot-augmented-reality-transforming-brion-carroll-ii/>. Accessed 3/2024.
- ChatGPT 2023. OpenAI. GPT-3.5. Accessed for language check. <https://chat.openai.com>. Accessed 5/2024.
- Diodes Incorporated. 1N4006G, 1N4007G - 1.0A standard rectifier. https://www.diodes.com/assets/Datasheets/1N4006G-1N4007G_LS.pdf. Accessed 1/2024.
- Espressif Systems 2019. ESP32-WROOM-32 datasheet. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf. 2/2024.
- Fingent. Augmented Reality in Healthcare: Use Cases, Examples, and Trends. Fingent Blog. <https://www.fingent.com/blog/augmented-reality-in-healthcare-use-cases-examples-and-trends/>. Accessed 3/2024.
- InfisIM 2024. The impact of IoT & AR on Modern Technology. InfisIM Blog. 19/1/2024. <https://infisim.com/blog/impact-of-iot-and-ar>. Accessed 2/2024.
- Jaivignesh, R, Janarthanan, R. D, & Gnanalakshmi, V. 2022. Smart Home Automation using Augmented Reality and Internet of Things. Journal of Physics: Conference Series, 2325(1), 012003. <https://doi.org/10.1088/1742-6596/2325/1/012003>. Accessed 2/2024.

Jo, Dongsik, & Kim, Gerard Jounghyun. 2019. IoT + AR: Pervasive and augmented environments for "Digi-log" shopping experience. *Human-centric Computing and Information Sciences*, 9, Article 1. <https://doi.org/10.1186/s13673-018-0157-x>. Accessed 3/2024.

Kelton, Mike 2019. Augmented reality smart glasses decrease cost and increase productivity. *Honeywell Blog*. 2.5.2019. <https://sps.honeywell.com/us/en/support/blog/automation/smart-glasses-decrease-cost-increase-productivity>. Accessed 5/2024.

Lincoln, David 2021. Augmented Reality Set to Take Maintenance to New Heights. *ABB Measurement Made Easy Blog*. 23/8/2021. <https://new.abb.com/products/measurement-products/measurement-products-blog/augmented-reality-set-to-take-maintenance-to-new-heights>. Accessed 3/2024.

Martinez, Pedro 2024. AR vs MR: Decoding the Key Differences between Augmented Reality and Mixed Reality. *Onirix Spatial AR Blog*. 5/3/2024. <https://www.onirix.com/ar-vs-mr/>. Accessed 5/2024.

Matiieshyn, Pavlo & Kalachova, Anastasiia 2024. IoT in manufacturing: How it revolutionizes the industrial landscape. *Lemberg Solutions Blog*. 3/1/2024. <https://lebergsolutions.com/blog/iot-manufacturing-how-it-revolutionizes-industrial-landscape>. Accessed 2/2024.

MDN Web Docs. JSON. <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON>. Accessed 5/2024.

MDN Web Docs. What is a web server? . https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_web_server. Accessed 5/2024.

Microsoft. What is Augmented Reality (AR) | Microsoft Dynamics 365. Microsoft Corporation. <https://dynamics.microsoft.com/en-us/mixed-reality/guides/what-is-augmented-reality-ar/>. Accessed 1/2024.

ON Semiconductor. BC547B: NPN Bipolar Junction Transistor. ON Semiconductor. <https://www.onsemi.com/products/discrete-power-modules/general-purpose-and-low-vcesat-transistors/BC547B>. Accessed 1/2024.

Polycam. Photo mode. <https://learn.poly.cam/photo-mode>. Accessed 4/2024.

Polycam. Polycam to Blender. <https://learn.poly.cam/polycam-to-blender>. Accessed 4/2024.

PTC. Unleashing the power of data with AR and IoT. PTC. <https://www.ptc.com/en/resources/iiot/white-paper/unleashing-power-data-ar-and-iiot/>. Accessed 5/2024. Accessed 2/2024.

Shoikova, E, Nikolov, R, & Kovatcheva, E 2018. Smart Digital Education Enhanced By AR AND IOT Data. In *INTED2018 Proceedings*. 12th annual International Technology, Education and Development Conference, Valencia, Spain. IATED. <https://doi.org/10.21125/inted.2018.1392>. Accessed 2/2024.

Suliman, N. A, Ricciardi Celsi, L, Li, W, & Villari, M 2022. Edge-oriented computing: A survey on research and use cases. *Energies* 5(2), 452. <https://doi.org/10.3390/en15020452>. Accessed 2/2024.

TE Connectivity. IC sockets. <https://www.te.com/en/products/connectors/sockets/ic-sockets.html#:~:text=These%20connectors%20are%20designed%20to,and%20easy%20repair%20and%20replacement>. Accessed 5/2024.

Tremosa, Laia 2024. Beyond AR vs. VR: What is the Difference between AR vs. MR vs. VR vs. XR. Interaction Design Foundation . 30/4/2024. <https://www.interaction-design.org/literature/article/beyond-ar-vs-vr-what-is-the-difference-between-ar-vs-mr-vs-vr-vs-xr>. Accessed 5/2024.

Unity Technologies. Creating scenes. <https://docs.unity3d.com/560/Documentation/Manual/CreatingScenes.html>. Accessed 3/2024.

Unity Technologies. GameObject. <https://docs.unity3d.com/560/Documentation/Manual/GameObjects.html>. Accessed 3/2024.

Unity Technologies. Get started with the Unity Hub. <https://learn.unity.com/tutorial/get-started-with-the-unity-hub/?tab=overview#637534d0edbc2a3adf9c965f>. Accessed 3/2024.

Unity Technologies. Hub manual. <https://docs.unity3d.com/hub/manual/index.html>. Accessed 3/2024.

Unity Technologies. Install Hub. <https://docs.unity3d.com/hub/manual/InstallHub.html>. Accessed 2/2024.

Unity Technologies. Products. <https://unity.com/products>. Accessed 3/2024.

Unity Technologies. The hierarchy windows. <https://docs.unity3d.com/Manual/Hierarchy.html>. Accessed 3/2024.

Unity Technologies. Using the Inspector. <https://docs.unity3d.com/Manual/UsingTheInspector.html>. Accessed 3/2024.

Verizon 2023. How AR and VR technology can enhance IoT applications. Verizon. <https://www.verizon.com/about/blog/vr-and-iot>. Accessed 5/2024.

Vuforia. Advanced model target databases. <https://developer.vuforia.com/library/model-targets/advanced-model-target-databases>. Accessed 4/2024.

Vuforia. Getting started with Vuforia Engine in Unity. <https://developer.vuforia.com/library/getting-started/getting-started-vuforia-engine-unity>. Accessed 3/2024.

Vuforia. How to create a model target. <https://developer.vuforia.com/library/model-targets/how-create-model-target>. Accessed 4/2024.

Vuforia. Introduction to Model Targets in Unity | Vuforia Library. Vuforia Engine Developer Portal. <https://developer.vuforia.com/library/model-targets/introduction-model-targets-unity>. Accessed 3/2024.

Vuforia. Model Target Generator user guide. <https://developer.vuforia.com/library/objects/model-target-generator-user-guide>. Accessed 4/2024.

Vuforia. Model targets. <https://developer.vuforia.com/library/objects/model-targets#:~:text=Model%20Targets%20enable%20Vuforia%20Engine,physical%20objects%20by%20their%20shape>. Accessed 4/2024.

Vuforia. Model targets supported objects and CAD model best practices. <https://developer.vuforia.com/library/model-targets/model-targets-supported-objects-cad-model-best-practices>. Accessed 4/2024.

Vuforia. Vuforia License Manager. <https://developer.vuforia.com/library/getting-started/vuforia-license-manager>. Accessed 2/2024.