كلّيات التقنية العليا
HIGHER COLLEGES OF TECHNOLOGY

AEROVOLT INNOVATORS

Data-Driven Airborne Wind Energy (AWE) Generation

ELE 2903 -Sophomore Design Project (CRN- 23183)

Final Report

Prepared for:
Dr. Yasser Alamoudi

Prepared by

Team Leader	**ID**
Hadi Saif	H00540225
Team members	**ID**
Bethel Yohannes	H00540236
Blen Assefa	H00540245
Kalkidan Getnet	H00540242
Mhretewold Endashaw	H00540239

May 2025

Table of Contents

Abstract

This report explores the feasibility and potential of Airborne Wind Energy (AWE) systems as a sustainable and efficient source of clean energy in the UAE. Unlike traditional wind turbines, AWE systems can access stronger, more consistent high-altitude winds while requiring less land and infrastructure, making them ideal for regions with limited space and variable surface wind conditions. Our project focuses on developing a small-scale, cost-effective AWE prototype that integrates low-cost hardware with intelligent, data-driven control to demonstrate the practicality of this technology.

The prototype features a helium-filled, tethered platform equipped with a wind turbine, generator, and an ESP32 microcontroller connected to onboard sensors. These sensors collect real-time data such as wind speed, temperature, humidity, altitude, voltage, and current during flight. Although storing the generated energy in a 12V battery remains a future step, our current focus was on data acquisition and system analysis. The prototype was iteratively refined, resulting in a more stable design and improved data transmission.

To optimize power output, we trained a Reinforcement Learning (RL) model using the collected data within a simulated environment that accurately reflected real-world behaviour. The model learned to autonomously adjust the system's altitude by controlling the tether, responding to environmental changes to find the most energy-efficient flight conditions. This allowed us to simulate how intelligent automation could replace manual adjustment and improve overall system performance.

As a proof of concept, we also built a small simulation city powered by the system's output, featuring automated lighting controlled by a photoresistor and Arduino. While simple, it offered a glimpse into how airborne wind systems could integrate into real-world energy infrastructures.

Ultimately, this project demonstrates how combining airborne wind energy with real-time sensing and reinforcement learning can lead to an adaptive, scalable, and low-cost renewable energy solution. It offers a promising alternative for clean power generation in land-constrained or wind-variable regions, contributing to the global push for more sustainable energy systems.

I. Introduction

People used wind energy to propel boats along the Nile River as early as 5,000 BC. By 200 BC, simple wind-powered water pumps were in use in China, and windmills with woven reed blades were grinding grain in Persia and the Middle East [1]. For centuries, people harnessed the power of the wind through windmills. Nowadays, we take this technology for granted, but windmills were one of the key advancements that our ancestors used to support their communities. In fact, before the advent of steam engines and electricity, wind was the only source of power that mankind could control. Even 3,700 years later, we still use windmills, although they have transformed into sophisticated devices primarily used to generate electricity. While it is true that wind is one of the most promising renewable energy sources available, it remains underutilized. This is mainly due to the challenges that conventional wind turbines face regarding land usage and the specific locations required for optimal operation. As a result, wind energy as a renewable resource is not developing as rapidly as solar energy or other renewable options. In this report, we will explore a more reliable and promising form of wind energy harvesting known as Airborne Wind Energy. We will also investigate the feasibility of a small-scale airborne wind energy system prototype for applications in the UAE, driven by real-time data collection and utilization.

Traditional wind turbines

Modern wind turbines

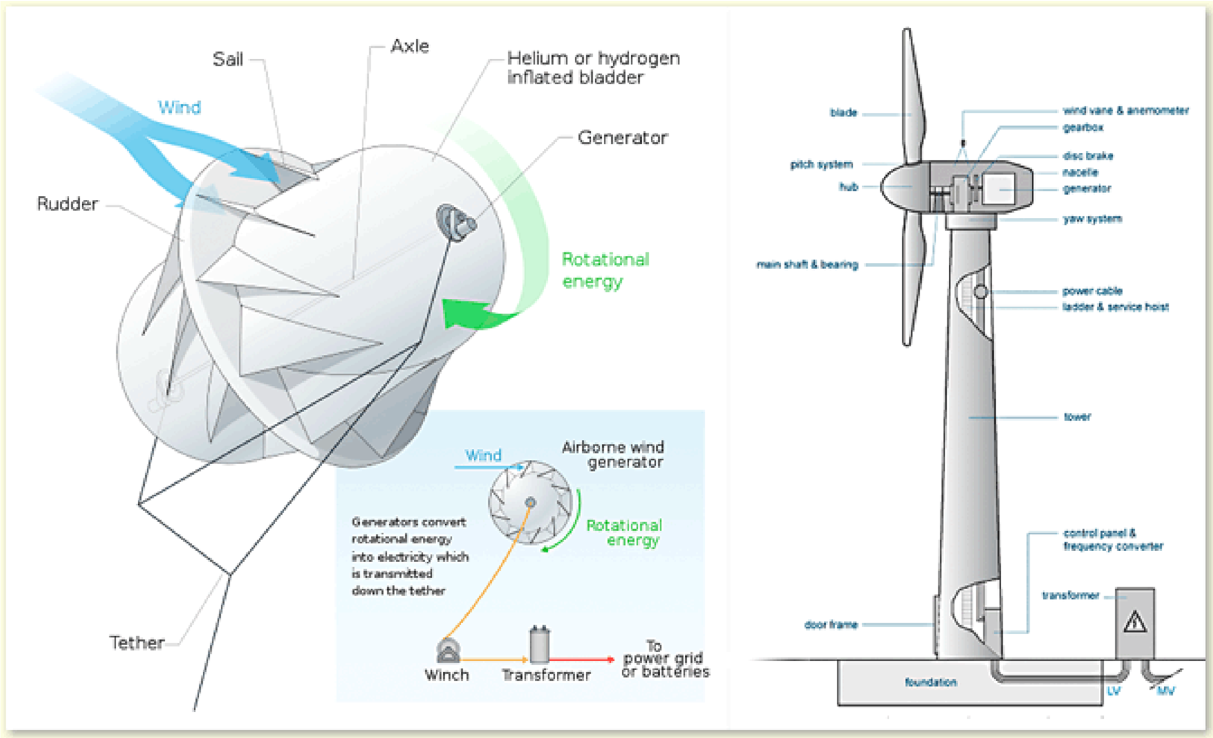
Objectives

- Explore Airborne Wind Energy as a renewable Energy source
- Compare Airborne Wind Energy with Conventional Wind Towers
- Design and develop small-scale prototype of AWES to test feasibility
- Collect relevant data and use the collected data to maximize energy outputs
- Examine the effect of utilizing real-time data on wind and weather on system performance

II. Literature Review

In 1970, an American engineer, Miles Lloyd, working at Lawrence Livermore National Laboratory laid the foundation for a new form of wind energy harvesting technique called Airborne Wind Energy [2]. Airborne Wind Energy is the conversion of wind energy into electricity using tethered flying aircraft [3]. Unlike traditional wind turbines that require large tower structures, an AWE (Airborne Wind Energy) system uses a tether cable to anchor the system to the ground and use turbines attached to some form of aircraft, which can be a kite, a blimp or drones [4].

The aircraft reaches high altitudes where the wind causes the turbines to rotate, hence driving a generator to convert the mechanical energy into electricity. The electricity is then transferred to the ground via transmission wires within the tether cable. Most of these flying devices are designed to fly at altitudes ranging from 200 to 1000 meters or more, where wind speed is much faster, reliable and more consistent than winds near the surface or where traditional wind towers stand. Wind at higher altitudes is also considered to be an energy resource that has not been exploited so far [4]. This reduces the cost and environmental footprint of wind energy generation. The paper by Miles Lloyd showcased the quantitative analysis for energy output of tethered flying systems compared to a similar sized traditional wind turbines and indicated that tethered flying systems could produce up to 3 times more power [2].



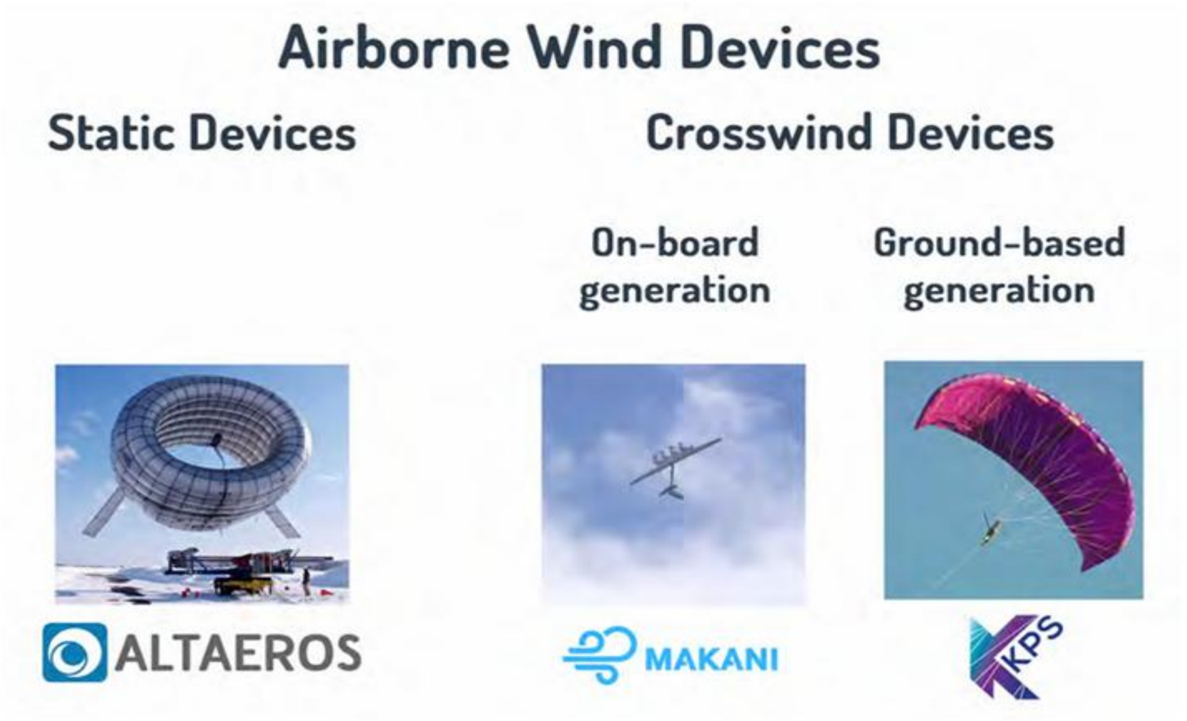
Implementation of AWE system

Research on AWESs started in the mid-seventies, with a rapid acceleration in the last decade. A number of systems based on radically different concepts have been analyzed and tested. Several prototypes have been developed all over the world and the results from early experiments are becoming available, but all these designs have 2 things in common:

A. Tethered Flying Device (Kite, balloons, drones, etc.)
B. Generator

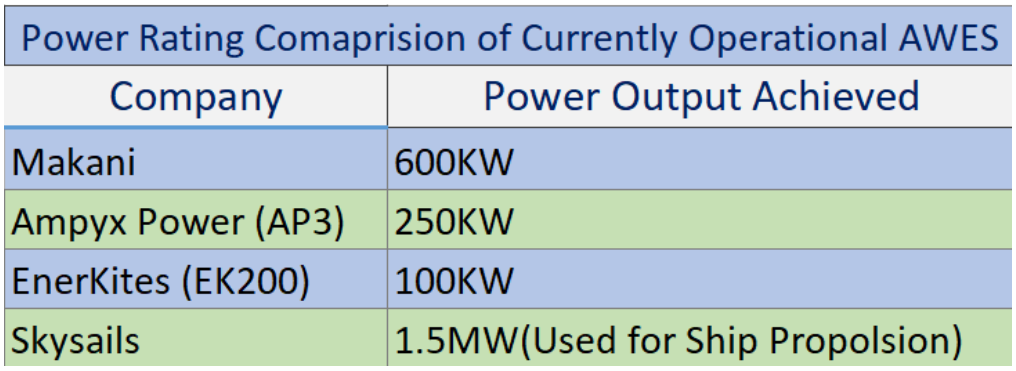
The system distinguishes itself from traditional wind energy systems by deviating from the conventional tower and blade setup. Instead, AWE takes a unique classification approach, which can be described as follows:

1. Static/Aerostatic devices
2. Crosswind/Aerodynamic devices
    - Ground based systems
    - Onboard systems



Static devices use Lighter-than-air devices like helium balloons to stay aloft, while crosswind devices like kites and gliders utilize their unique aerodynamic design that assists in producing lift to keep the system aloft.

1. Static Devices: This system uses buoyant, helium-filled structures to lift small turbines or tension tethers. These devices ascend and stabilize with minimal active maneuvering, either generating electricity onboard—transmitted to the ground via a conductive tether—or driving a ground-based generator through the tether's pull, as seen in systems like Altaeros Energies' Buoyant Airborne Turbine [5].
2. Cross wind Devices: They are classified into two based on the location of the generator.
   A. Ground-based generation: In this system, the mechanical work of a traction force generates electrical energy on the ground. This force is transferred from the aircraft to the ground system via ropes, which are reeled in and out in a repeated cycle. This cycle of winding the tether and unwinding drives the motor to produce power. Some energy is consumed when winding the system, however the net power produced is significantly higher than the power consumed to reel in the system [6].
   B. On-board generation: Electrical energy is generated on the aircraft and transmitted to the ground via a specialized rope carrying electrical cables. Wind turbines are commonly used to convert the wind's energy into electrical power. This innovative approach allows for electricity generation to occur on board the aircraft itself [6].



Power Rating Comaprision of Currently Operational AWES	
Company	Power Output Achieved
Makani	600KW
Ampyx Power (AP3)	250KW
EnerKites (EK200)	100KW
Skysails	1.5MW(Used for Ship Propolsion)

Table: Power rating comparison of Currently operational AWE companies

Advantages of AWE system

❖ **Uses Less Materials**

- Replacing the tower of a wind turbine by a lightweight tether substantially reduces the material consumption by up to 90%, thus decreasing the environmental impact with regards to the carbon footprint over the life cycle as well as reducing visual impacts [7].
- Typical Wind turbine towers have mass of 60,000KG/MW while equivalent ground-based systems have airborne weight of just 100KG/MW. This translates to 60 times mass reduction. This consequently applies the 'reduce' rule which is one of the most valuable Circular Economy options [3].

❖ **Cost Reduction**- since AWE systems require less material and are cheaper to install and maintain.

- The decrease in capital costs (CAPEX) due to low material use, the increase in capacity factor, the easier logistics and quick set-up as well as the high power density per square-kilometre can potentially lead to a substantial reduction of the levelized costs of wind energy (LCOE) [7].
- Studies suggest that AWE could cost 0.5 -1.5 cents per Kilo-Watthour which is over 3 to 10 times cheaper than conventional wind turbines that cost 5 -12 cents per Kilo-Watthour [8].

❖ **Access to untapped wind resources at high altitudes**

- High Altitude winds are usually stronger and more consistent than surface wind, however we are still unable to utilize this promising renewable source to its full potential.

❖ **Potentially low environmental impact**

- AWE is expected to contribute much less carbon footprint and generate less noise than traditional wind turbines.

❖ **Less Land Usage**

- In 2018, a sector study on the potential of AWE presented by the European Commission showed that by utilizing Just 1% of the land of the European Union, AWE systems can fulfill 100% of the electricity demand of the region. This is proof of the potential of AWE [9].

❖ **Mobility**

- AWE systems can be easily utilized in various locations such as offshore locations, mountainous locations and even in remote islands.

Disadvantages of AWE system

- **Emerging Technology and limited Data**
  - AWE is still in its Infancy and required further research and development before full scale global adoption [3].
- **Technical Challenges**
  - The system still faces issues such as lack of fully automatized launching and landing systems as well as challenges in developing lightweight and durable materials. Components like tethers, lightweight airframes, and control systems must endure high mechanical stress, extreme weather, and thousands of operational cycles (e.g., reeling in/out for ground-generation systems).
- **Regulatory Hurdles**
  - Many regions have strict air space regulations, requiring permits, no-fly zone compliance, and visibility markers (e.g., lights on tethers), which increase operational costs. Securing approval for widespread deployment, especially near populated regions or flight corridors, is a slow and uncertain process, stalling projects despite technical feasibility.
- **Safety Concern**
  - AWE systems pose unique safety hazards, such as tether breakage or device crashes, which could endanger people, property, or aircraft below. Operating at 200–1,000 meters, they overlap with low-altitude airspace, raising collision risks near airports or urban areas. For instance, a loose kite or balloon could drift into flight paths, and conductive tethers (used in onboard-generation systems) might pose electrical hazards if severed during storms, complicating emergency response protocols.

Applications of AWE system

- Off-Grid and Remote Power Supply
  - AWE systems are ideal for delivering electricity to isolated areas without access to traditional grids or where building wind turbine towers is impractical. Their lightweight, portable design allows rapid deployment in rugged terrains, islands, or rural regions.
- Disaster Relief and Emergency Power
  - In disaster-stricken areas where infrastructure is damaged, AWE systems can provide temporary electricity quickly. Their compact, transportable nature suits rapid setup by relief teams. It offers a sustainable alternative to fossil-fuel generators, with minimal setup time (hours vs. days for traditional solutions).
- Microgrid and Community Power Systems

  - Small-scale AWE units (10–100 kW) can power microgrids for villages, farms, or eco-communities, especially in windy regions like coastal areas or highlands. This system Provides decentralized, renewable energy with lower land use than solar or wind farms, ideal for sustainable development projects.
- ❖ Hybrid Renewable Energy Systems
  - AWE can integrate with solar, battery storage, or traditional wind to create hybrid setups, balancing intermittent generation and leveraging high-altitude winds during low solar or surface wind periods.
- ❖ Research and Environmental Monitoring
  - Beyond power generation, AWE platforms like aerostats can carry sensors for atmospheric research, wind profiling, or climate monitoring while producing energy to sustain operations. It combines energy production with scientific utility, offsetting costs in research budgets.
  -

Significance of AWE system for UAE

- ❖ Provide Access to Wind Energy
  - UAE is a low-wind region. This makes us less ideal for traditional wind turbines, however, with AWES, UAE can exploit the strong and more consistent wind at higher Altitudes, expanding its renewable energy portfolio beyond solar power.
- ❖ Reduced Land Usage
  - Building traditional wind farms requires a lot of space. And land, being an asset in UAE, means building wind farms in the region is expensive and not viable. As such, AWES requires less area and is a more promising wind energy source.
- ❖ AWES can be deployed around coastlines of the country
  - AWES can be strategically deployed along the UAE's extensive coastlines, such as those along the Persian Gulf, where offshore winds at altitude are stronger and more consistent. Tethered to buoys or small platforms, these systems can generate power in deep waters without the need for expensive seabed foundations, making them a practical solution for coastal cities like Dubai and Abu Dhabi to meet local energy demands sustainably.
- ❖ Technological Innovation
  - UAE is one of the leading countries in the adoption of renewable energy. Utilizing the emerging field of AWES can lead to a new economic opportunity for the country as well as put UAE as an expert in this innovative field.

III. Design Concept

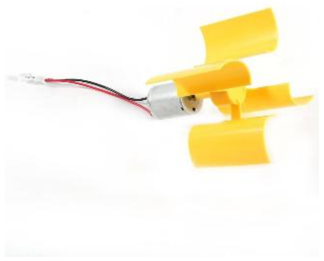
After comparing the different classifications of AWE systems, we decided to go with a lighter than Air system (Aerostatic device). These systems are usually more stable and energy efficient compared to crosswind devices. For instance, kites require a constant and strong wind blow to launch and stay aloft which makes it a less viable choice for a low wind region like the UAE. In addition to this generating energy using a crosswind system is much more sophisticated and requires complex engineering.

We also decided to utilize onboard generation which involves attaching turbine and generator to the aircraft. This is because ground-based generation requires stronger winds than UAE's environment has to offer.

The Design concept for the prototype is as follows:

A. Blimp-Based AWE system – A blimp is helium filled to provide lift and remain airborne with minimal energy consumption. The structure is constructed using lightweight and durable materials to withstand high-altitude winds. The main reason behind this choice is blimps stability and energy efficiency. A motor-driven tether system is implemented to control the navigation.

B. On board generation – The wind turbine and generator are attached to bottom of the blimp. It is then carried to high altitudes where the turbines drive the generator to convert the kinetic energy of the wind into electricity. We utilized a 24V wind turbine, which effectively produced power at elevated altitudes. The generated electricity is then transmitted via a tether to the ground.

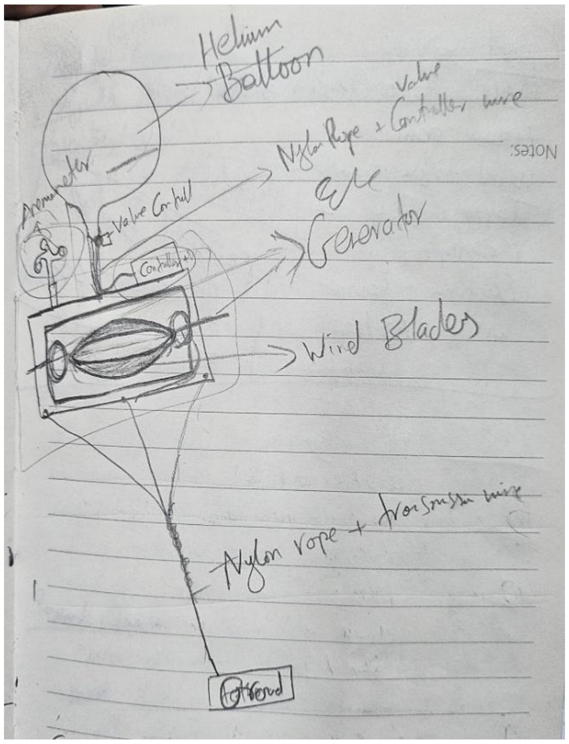


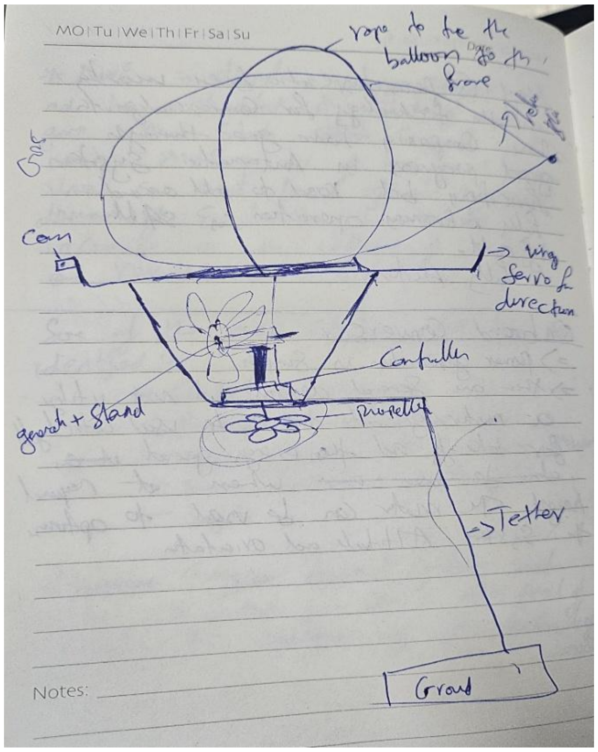
C. Arduino modules for sensors and data collection and transmission: We chose Arduino because it is simple and accessible for use. It also has a number of sensors that can be used for a variety of applications.

D. ESP32 – is a microcontroller that has Wi-Fi and Bluetooth capabilities. Since it is also compatible with Arduino sensors, ESP32 is ideal for transmitting and receiving data over long range.

Design Concept Sketch

The **balloon** acts as a lightweight, elevated platform, potentially simulating an airborne device that captures environmental data. It is stabilized by a **circular cardboard holder**, to anchor the balloon while maintaining the system's lightweight and low-cost nature. The **shelf** serves as a stable base to support the system's components, while the **Arduino with sensors** is the brain of the setup, gathering and processing weather information like temperature, wind, or humidity. The overall design reflects a modular and easily replicable approach, combining low-cost materials with accessible technology to optimize data collection for energy generation.





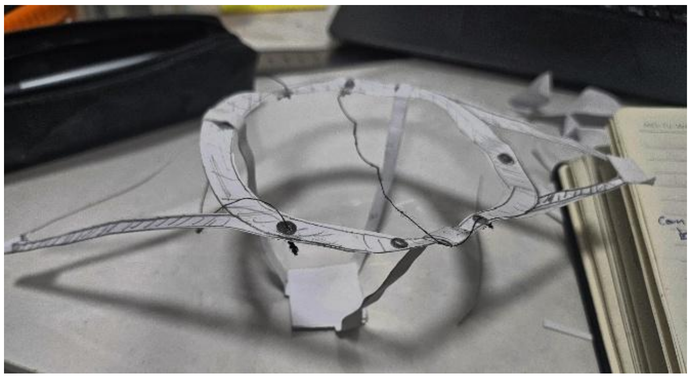


Fig: First simple design sketch.

Illustration of our System

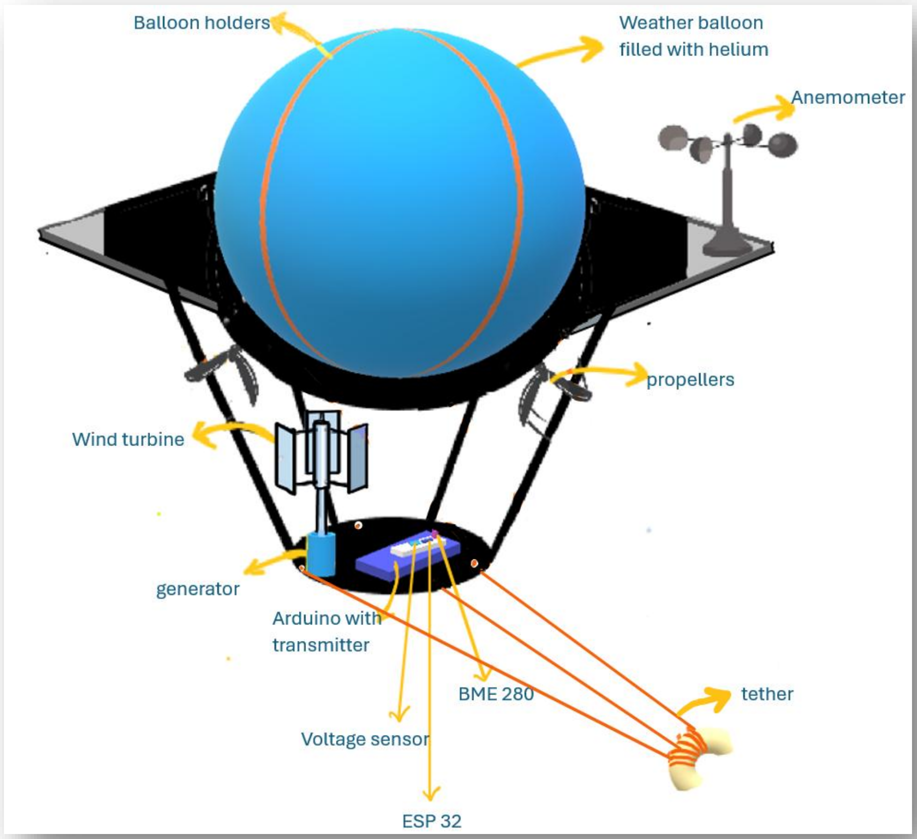


Fig: Second design sketch

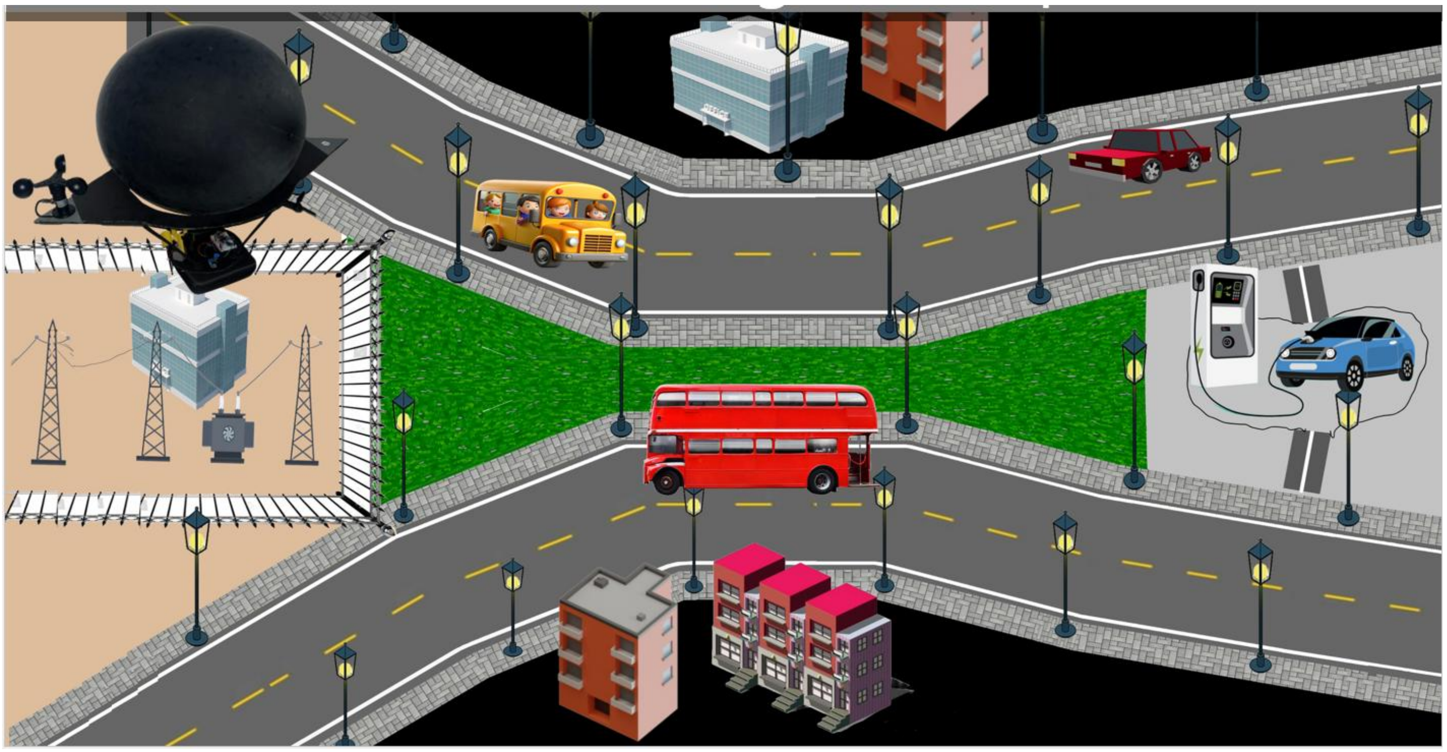
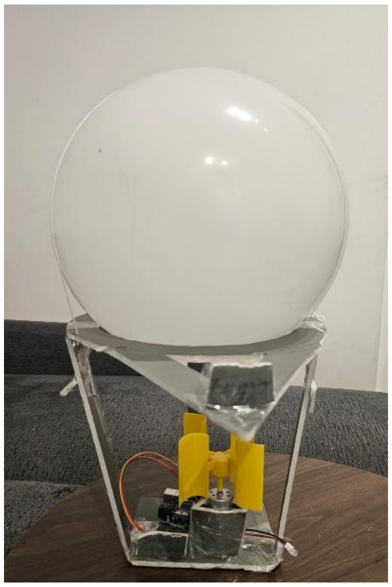
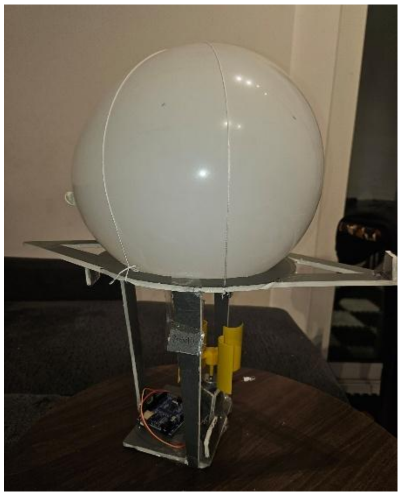
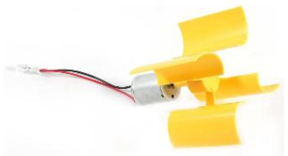
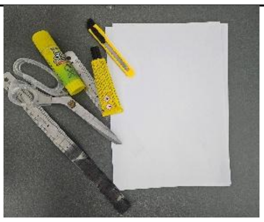
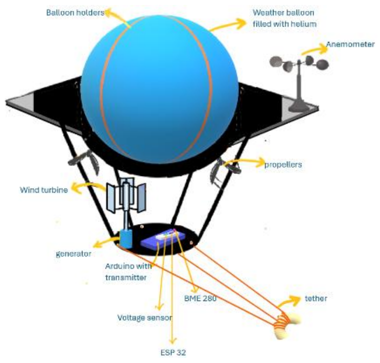
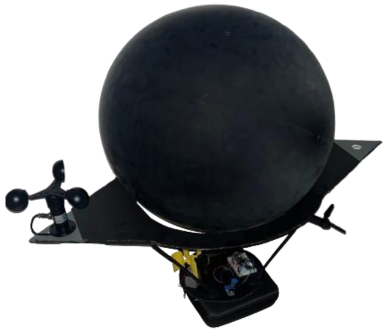


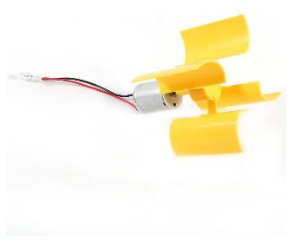
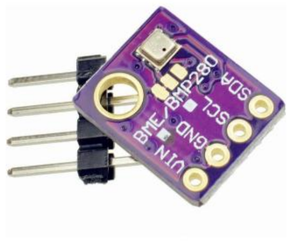
Fig: Final design sketch

Key components of the Final Prototypes

- ✓ **ESP32 –** Collected sensor data, transmitted it wirelessly, and executed control decisions from the reinforcement learning model.
- ✓ **BME280 Sensor** – Used to capture temperature, humidity, pressure, and altitude during flight, providing environmental context for training reinforcement learning agent.
- ✓ **Wind Turbine with Generator** – Positioned below the balloon to harness wind energy at higher altitudes and convert it into electricity for measurement and storage.
- ✓ **Voltage & Current Sensors** – Connected to the turbine to measure the generated power. This helped us calculate real-time power output and evaluate system performance.
- ✓ **Portable anemometer** –Used to measure wind speed at altitude.
- ✓ **Relay** – Used to control the charging and discharging process of the energy storage system.
- ✓ **Photoresistor (LDR)** – Detects light levels to automatically turn LEDs on or off in the simulation city.
- ✓ **Arduino** – Used in our simulation city to control lights via a phototransistor, demonstrating how stored energy from the AWE system can power real-life infrastructure.
- ✓ **12V Battery** – Stores the electricity generated by the AWE system for later use, such as powering electrical components.

Prototype	Material Used	Picture	Quantity	Price
First Prototype  	Foam Board		1	5 AED
	Birthday Balloon		1	8 AED
	Sewing Threads		1	1 AED
	Vertical Wind Turbine		1	30 AED
	Stationary Items (Scissors, Paper, Glue, Duct Tape)		1	15 AED
	Total Cost			59 AED

Second Prototype  Balloon holders Weather balloon filled with helium Anemometer propellers Wind turbine generator Arduino with transmitter BME 280 tether Voltage sensor ESP 32 	Foam Board		1	5 AED
	Balloon 36 Inch		1	6 AED
	Helium Tank		1	135 AED
	Wood Sticks		4	5 AED

	Anemometer		1	92 AED
	ESP32		1	25 AED
	Vertical Wind Turbine		1	30 AED
	Motor with Propellers		4	60 AED
	BME 280 Humidity, Temperature and Pressure sensor		1	35 AED

	Breadboard		1	8 AED
	Nylon Threads as Tether		1	2 AED
	Electric Wires		1	9 AED
	Total Cost		-	450 AED
Final Prototype	Removed propellers	-	-	-
	Bigger balloons	-	2	25
	Current Sensors		1	47
	Voltage sensors		1	35

	12 V battery		1	35
	Relay		1	25
	Photoresistor		1	10
	Total Cost			177

IV. Prototypes

PROTOTYPE I

Our first prototype, while not airborne, provided valuable insight into the design of our final model. We constructed the **pillar and shelf** using foam board, which offered both lightweight properties and structural support. The **balloon** was securely fastened with a thread, ensuring stability during testing. Components were attached to the framework using plaster for durability and easy adjustments. This early version acted as a foundation for refining our concept and envisioning the next steps toward achieving a functional airborne system.

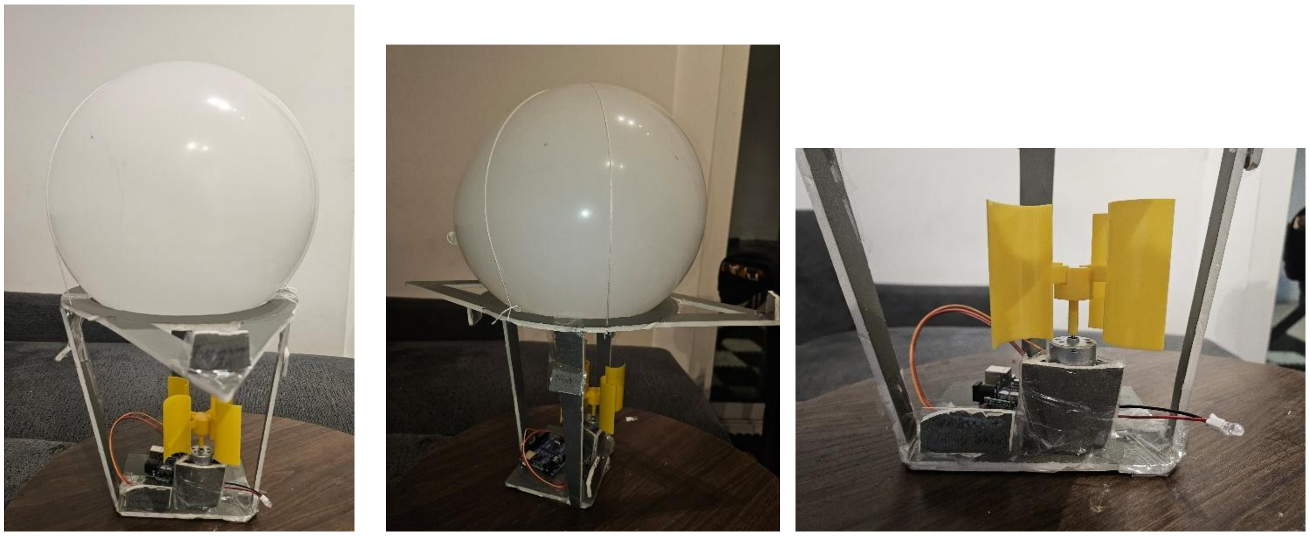
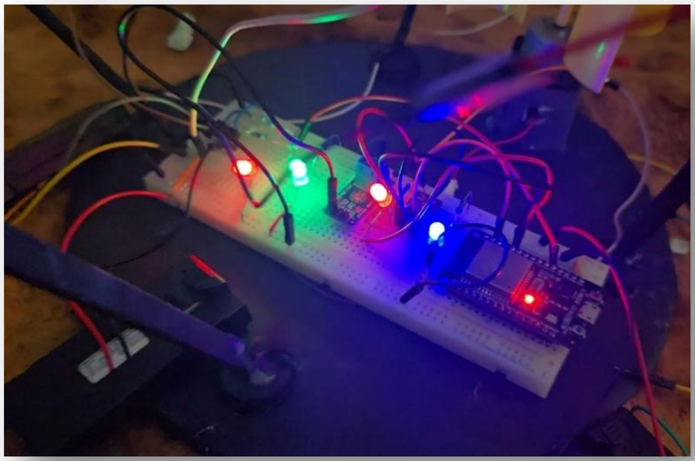


Fig: First prototype

PROTOTYPE II

Our second prototype achieved flight and offered improvements in design and functionality. The **pillars** were made from wood to provide a stable framework, while the **foam board** was laminated to prevent bending or tearing. The components, including sensors and the ESP32, were attached to the system using a glue gun for secure placement. The **balloon** was inflated with helium, allowing it to become airborne and enabling more effective testing in conditions closer to its intended use.

The **receiver on the ground** played a vital role in the system's operation. It collected real-time weather data from the airborne sensors and could also access historical data from the past 15 days. This capability ensured a more comprehensive understanding of weather patterns and allowed the data to be stored for future use. The receiver's performance proved to be reliable, helping us gather all the necessary information to refine and optimize the system.



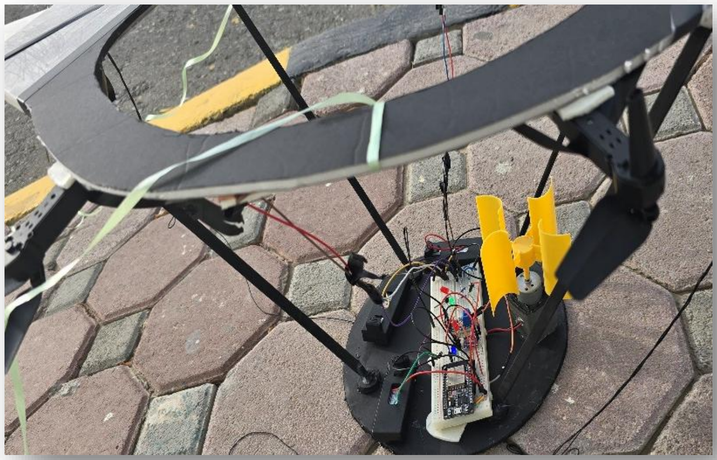
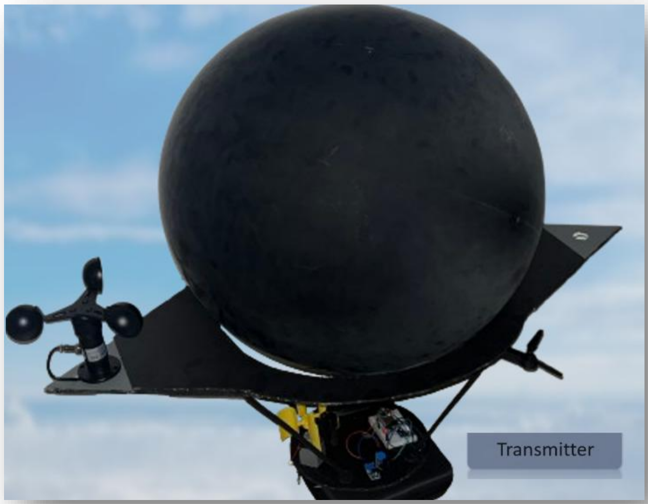


Fig: Wiring

These developments in the second prototype provided a clearer path forward as we continued working towards a fully functional airborne wind energy and weather monitoring system.



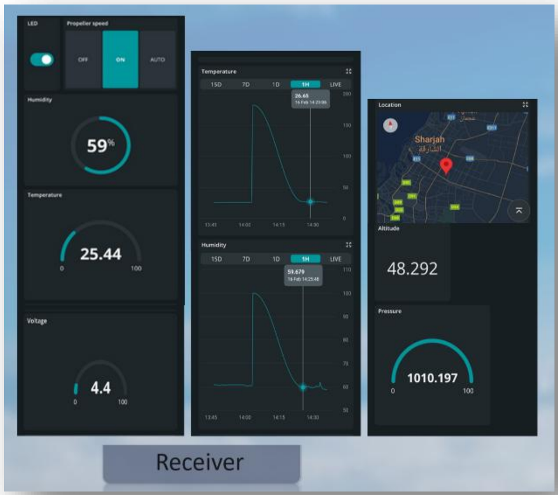


Fig: Second prototype

FINAL PROTOTYPE

For our third and final prototype, we kept the same basic design but made several important improvements based on what we learned from the first two versions. The first prototype didn't fly—it was mainly built to help us visualize the concept and understand how everything would fit together. The second prototype did achieve flight, but it only stayed in the air for a short time because the balloon couldn't lift the full system effectively.

To solve this, we used two large 2m diameter balloons in the final prototype and filled them with as much helium as possible. This time, the system stayed airborne for around 2 hours, which gave us plenty of time to collect real-time weather data for our machine learning model.

One of the biggest problems we faced in Prototype II was keeping the system balanced during flight. So, in this version, we added wooden sticks around the frame to give it more rigidity and support. We also decided to remove the propellers, since the increased lift from the balloons made them unnecessary.

We also improved the tether system, making it more stable and easier to control while the prototype was flying. With all these updates, the final design gave us longer flight time, better data collection, and overall, more stable performance.

In addition to the flying prototype, we built a small simulation city powered by a battery connected to our system. Using photoresistors and Arduino, we demonstrated how stored energy from the AWE setup could be used in real-life applications like smart street lighting.

This prototype was a big step forward for our project. It brought together everything we'd learned and showed us that our system could realistically work in real-life scenarios.

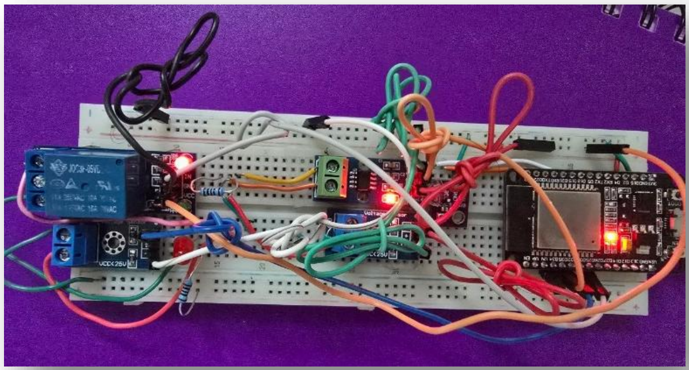


Fig: Wiring of the sensors

Fig: Final prototype

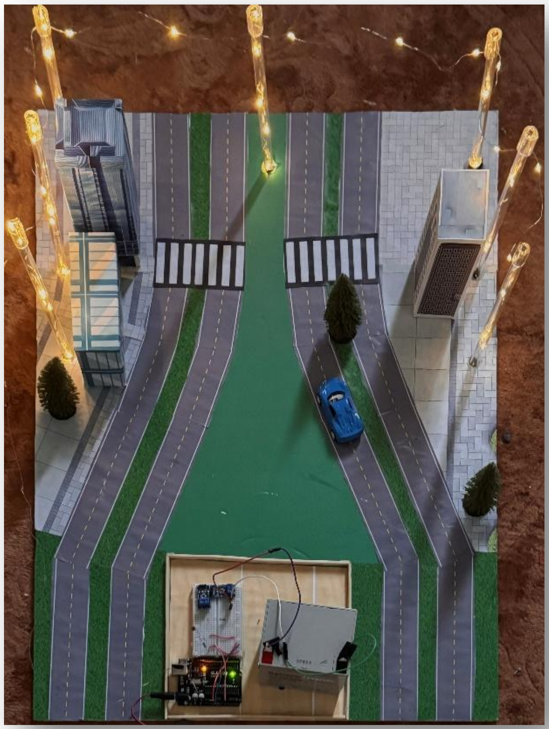


Fig: Simulation City

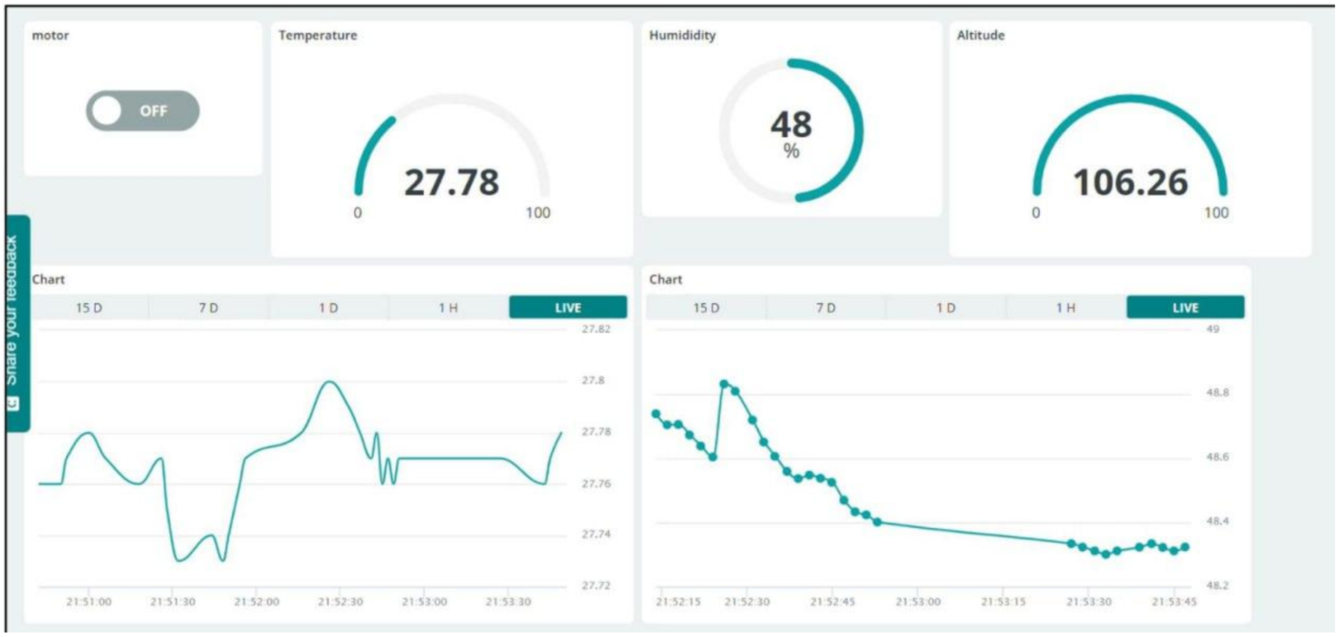
The code for our transmitter-receiver circuit

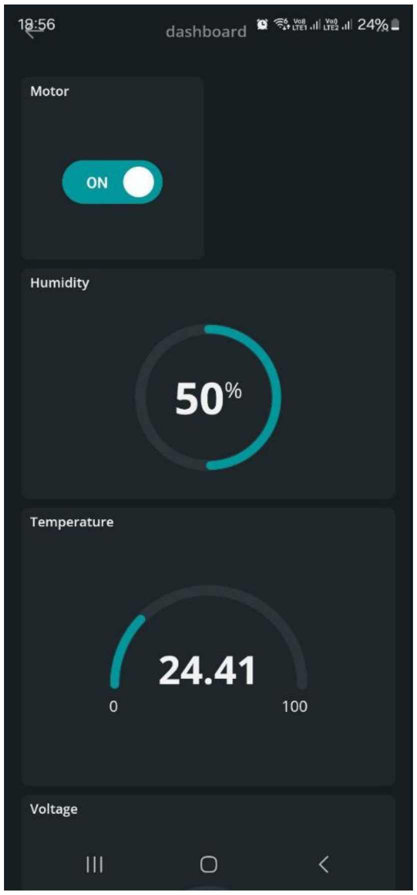
```
#include <math.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <Adafruit_BME280.h>
#include <Wire.h>
#include "thingProperties.h"  // For Arduino IoT Cloud

// WiFi credentials
const char* ssid     = "606 5G";
const char* password = "MEKM@606";

// Sensor setup
Adafruit_BME280 bme;
float seaLevelPressure = 1016;

// Sensor pins
const int relayPin = 26;
const int voltageSensorPin = 32;
float batteryVoltage = 0.0;
```

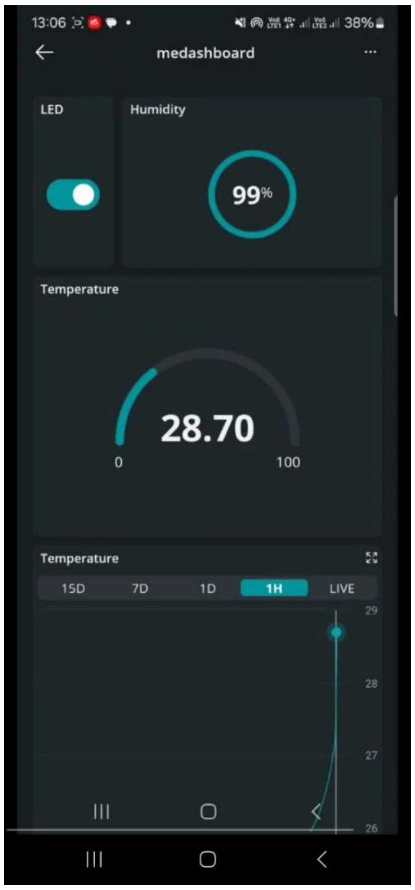
```
const float maxVoltage = 12.0;
const float voltageDividerRatio = (3.3 / 4095.0) * (11.0 / 2.0);

#define VOLTAGE_SENSOR_PIN 34
#define CURRENT_SENSOR_PIN 35

// Global variables

float pressure    = 0.0;
float windspeed   = 2.5;


float powers[3] = {0.0, 0.0, 0.0};
float windSpeeds[3] = {0.0, 0.0, 0.0};
float altitudes[3] = {0.0, 0.0, 0.0};
// Timing
unsigned long previousMillis = 0;
const long interval = 10000;  // 10 seconds

void setup() {

  pinMode(relayPin, OUTPUT);
  digitalWrite(relayPin, LOW);
  Serial.begin(115200);
  delay(100);

  // Arduino IoT Cloud setup
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();

  // Local WiFi connection for HTTP
  Serial.print("Connecting to WiFi ");
  Serial.print(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected to WiFi");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());

  // BME280 setup
  if (!bme.begin(0x76)) {
    Serial.println("BME280 not found! Check wiring.");
    while (true) delay(10);
  }
  Serial.println("BME280 initialized.");

  pinMode(2, OUTPUT); // For LED
  pinMode(4, OUTPUT); // Motor pins
  pinMode(5, OUTPUT);
}

void loop() {
  ArduinoCloud.update(); // Keep IoT Cloud updated

  readBME();
  readVoltage();
  readCurrentSensor();

  int sensorValue = analogRead(voltageSensorPin);
  batteryVoltage = sensorValue * voltageDividerRatio;

  Serial.print("Battery Voltage: ");
  Serial.println(batteryVoltage);

  if (batteryVoltage >= maxVoltage) {
    digitalWrite(relayPin, LOW);  // Stop charging
    Serial.println("Battery fully charged. Stopping charge.");
  } else {
    digitalWrite(relayPin, HIGH);  // Continue charging
    Serial.println("Charging...");
  }

  delay(1000);


  // Shift historical data
  for (int i = 0; i < 2; i++) {
    powers[i] = powers[i+1];
    windSpeeds[i] = windSpeeds[i+1];
    altitudes[i] = altitudes[i+1];
  }

  powers[2] = voltage * current;

  windspeed = pow(voltage, 1.0/3.0) * 4.82;

  windSpeeds[2] = windspeed;
  altitudes[2] = altitude;

  unsigned long now = millis();
  if (now - previousMillis >= interval) {
    previousMillis = now;
    sendData();
    sendDataToServer();
    sendProcessedData();
  }
}

void readBME() {
  temperature = bme.readTemperature();
  pressure    = bme.readPressure() / 100.0F;
  humidity    = bme.readHumidity();
  altitude    = bme.readAltitude(seaLevelPressure);

  Serial.println("----- BME280 Readings -----");
  Serial.print("Temperature: "); Serial.println(temperature);
  Serial.print("Pressure: "); Serial.println(pressure);
  Serial.print("Humidity: "); Serial.println(humidity);
  Serial.print("Altitude: "); Serial.println(altitude);
}

void readVoltage() {
  int rawVoltage = analogRead(VOLTAGE_SENSOR_PIN);
  int voltage = (rawVoltage / 4095.0) * 19.7;
  Serial.print("Voltage: ");
  Serial.print(voltage);
  Serial.print(" V | Raw: ");
  Serial.println(rawVoltage);
}

void readCurrentSensor() {
  float current = readCurrent(CURRENT_SENSOR_PIN);
  Serial.print("Current: ");
  Serial.print(current, 3);
  Serial.println(" A");
}

float readCurrent(int pin) {
  float sumCurrent = 0;
  const int samples = 1000;

  for (int i = 0; i < samples; i++) {
    int sensorValue = analogRead(pin);
    float voltage = (sensorValue * 3.3) / 4095.0;
    float current = (voltage - 3.3 / 2) / 0.185;  // Assuming ACS712-5A
    sumCurrent += current;
    delay(1);
  }
  return sumCurrent / samples;
}

// ------------------ DATA SENDING ------------------

void sendData() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("WiFi not connected – skipping send.");
    return;
  }

  WiFiClient client;
  HTTPClient http;
  const char* url = "http://192.168.185.48:5000/data";

  Serial.print("[HTTP] Connecting to "); Serial.println(url);
  http.begin(client, url);
  http.addHeader("Content-Type", "application/json");

  String json = "{";
  json += "\"temperature\":" + String(temperature) + ",";
  json += "\"humidity\":" + String(humidity) + ",";
  json += "\"wind\":" + String(windspeed) + ",";
  json += "\"altitude\":" + String(altitude) + ",";
  json += "\"voltage\":" + String(voltage) + ",";
  json += "\"current\":" + String(current);
  json += "}";

  int httpResponseCode = http.POST(json);
  if (httpResponseCode > 0) {
    Serial.print("Data sent. Response: ");
    Serial.println(httpResponseCode);
  } else {
    Serial.print("Error sending data: ");
    Serial.println(httpResponseCode);
  }

  http.end();
}

void sendDataToServer() {
  if (WiFi.status() != WL_CONNECTED) return;

  HTTPClient http;
  String url = "http://192.168.185.48:5000/data";
  http.begin(url);
  http.addHeader("Content-Type", "application/json");

  String json = "{";
  json += "\"temperature\":" + String(temperature) + ",";
  json += "\"humidity\":" + String(humidity) + ",";
  json += "\"wind\":" + String(windspeed) + ",";
  json += "\"altitude\":" + String(altitude) + ",";
  json += "\"voltage\":" + String(voltage) + ",";
  json += "\"current\":" + String(current);
  json += "}";

  int response = http.POST(json);
  if (response > 0) {
    Serial.print("Data sent to server. Response: ");
    Serial.println(response);
  } else {
    Serial.print("Server error: ");
    Serial.println(response);
  }

  http.end();
}

void sendProcessedData() {
  if (WiFi.status() != WL_CONNECTED) return;

  HTTPClient http;
  String url = "http://192.168.185.48:5000/processed";
  http.begin(url);
  http.addHeader("Content-Type", "application/json");

  float powerChange = powers[2] - powers[1];

  String json = "{";
  json += "\"wind_speed\":" + String(windSpeeds[2]) + ",";
  json += "\"altitude\":" + String(altitudes[2]) + ",";
  json += "\"power_change\":" + String(powerChange);
  json += "}";

  int response = http.POST(json);
  if (response > 0) {
    Serial.print("Processed data sent. Response: ");
    Serial.println(response);
  } else {
    Serial.print("Error sending processed data: ");
    Serial.println(response);
  }

  http.end();
}


// ----------------- CONTROL -----------------

void controlMotor() {
  if (motor == true) {
    digitalWrite(4, HIGH);
    digitalWrite(5, HIGH);
  } else {
    digitalWrite(4, LOW);
    digitalWrite(5, LOW);
  }
}

void onMotorChange() {
  controlMotor();
}

void onLEDChange() {
  digitalWrite(2, led ? HIGH : LOW);
}


void onTemperatureChange() {
  Serial.println("Temperature changed!");
}
```

Our Weather Station Dashboard

Weather station readings from our dashboard







How does our system Work?

System Used:

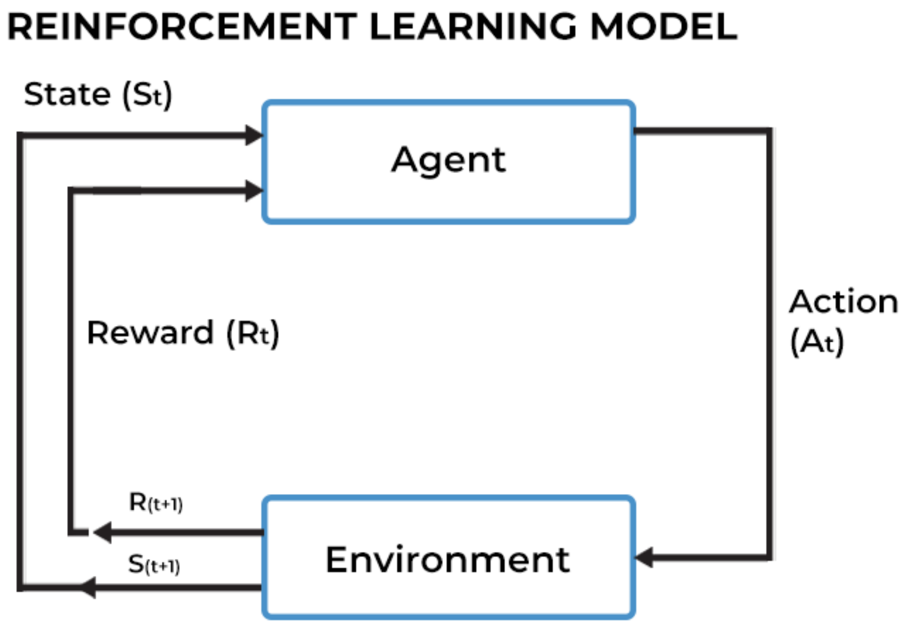
- Lighter-than-air aircraft
- Onboard power generation and sensing
- Energy storage and wireless communication

Process:

- A **helium-filled balloon** lifts the system, which includes a **wind turbine and generator** to produce electricity, and **ESP32 microcontroller** to handle data processing, communication, and control.
- The system is equipped with sensors including a **BME280** (temperature, pressure, humidity, altitude), a **voltage sensor and current sensor**.
- The generated power is stored in a **12V battery**, and a **relay** is used to control its charging and discharging process.
- A **DC motor connected to the tether** is used to adjust the system's altitude. This motion is automated through a **Reinforcement Learning (RL) model**, which makes decisions based on wind speed, altitude, and power output to optimize flight height.
- All collected data—including weather conditions and electrical output—is transmitted in real time via **Wi-Fi** to a **ground control interface** using HTTP.
- The system logs and sends raw data to a local server. This was then processed and used in model training and decision-making.
- A small **simulation city** powered by the stored energy is used to demonstrate real-life application. LEDs in the model city are controlled via **photoresistors** to simulate automatic lighting based on ambient conditions.

V. Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning where an agent learns to make decisions by interacting with its environment to maximize a defined reward. Unlike supervised learning, where the model learns from labeled data, or unsupervised learning, where the model seeks patterns in unlabeled data, RL is based on trial-and-error. The agent receives feedback in the form of rewards or penalties, which guide its learning process over time [16].

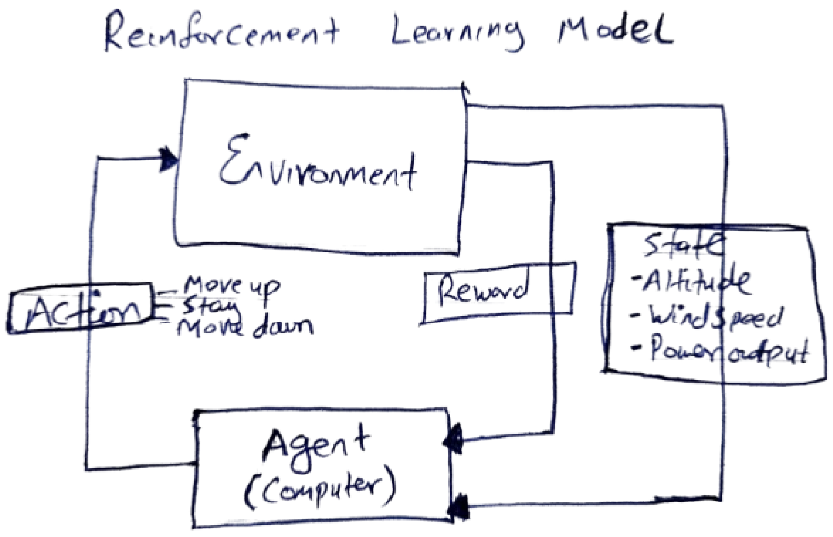


Implementation in our system

For our AWE system, we trained an RL model with the following components:

- Environment – A simulated environment in Pygame was used to train and test the system.
- State –
    1. Altitude
    2. Wind speed
    3. Power output from the wind turbine.

- Action – using a motor the agent will reel the tether causing the system to
    1. Move upward
    2. Stay in place or
    3. Move downward
- Reward – based on the action and state the agent will receive a reward equal to the difference in the power output before and after the action is taken
    1. Positive reward, if the power output is maximized
    2. Negative reward, if the power output falls

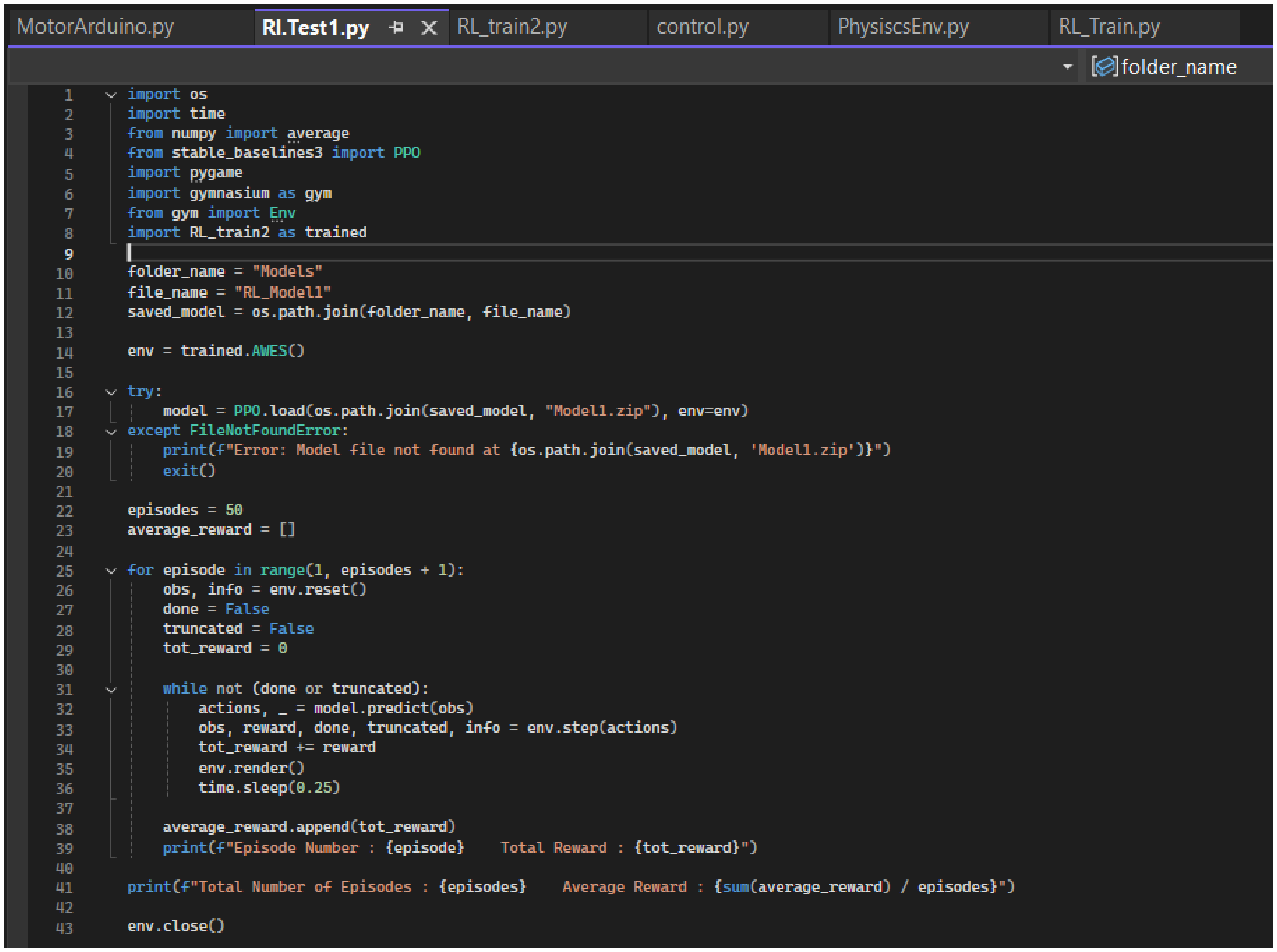


Significance of RL model to our system

The integration of reinforcement learning (RL) into our Airborne Wind Energy (AWE) system significantly enhances its performance by enabling **autonomous control** and **real-time optimization** of energy generation. Traditionally, the operation of such systems requires manual control of altitude and position, which are usually kept in a fixed range rather than responding to the changing weather conditions. However, this is inefficient and fails to fully exploit the dynamic nature of the environment. This is why Our RL-powered system:

- **Continuously learn and adapt** to real-time weather data,
- **Identifies the optimal altitude** for maximum wind energy extraction,
- **Automate the control process**, reducing human intervention and errors,
- And ultimately, **increase total power output** by maintaining the best operating conditions.

Code snapshot for the RL model



```python
import os
import time
from numpy import average
from stable_baselines3 import PPO
import pygame
import gymnasium as gym
from gym import Env
import RL_train2 as trained

folder_name = "Models"
file_name = "RL_Model1"
saved_model = os.path.join(folder_name, file_name)

env = trained.AWES()

try:
    model = PPO.load(os.path.join(saved_model, "Model1.zip"), env=env)
except FileNotFoundError:
    print(f"Error: Model file not found at {os.path.join(saved_model, 'Model1.zip')}")
    exit()

episodes = 50
average_reward = []

for episode in range(1, episodes + 1):
    obs, info = env.reset()
    done = False
    truncated = False
    tot_reward = 0

    while not (done or truncated):
        actions, _ = model.predict(obs)
        obs, reward, done, truncated, info = env.step(actions)
        tot_reward += reward
        env.render()
        time.sleep(0.25)

    average_reward.append(tot_reward)
    print(f"Episode Number : {episode}    Total Reward : {tot_reward}")

print(f"Total Number of Episodes : {episodes}    Average Reward : {sum(average_reward) / episodes}")

env.close()
```

VI. Testing

Test 1

Once the prototype was completed, we performed our systems' first test on February 17, 2025. In this stage of our project, we primarily focused on testing the feasibility of the system and effectiveness of data retrieval.

Finding From Test1

1)The prototype had a flight time of 10 seconds reaching altitude of around 4meters from surface. The main reason for the limited altitude is due to insufficient helium volume. We also concluded that the system was too heavy for the helium to produce enough lift.

2)We were able to retrieve accurate reading from the sensors at horizontal distance of approximately 6 meters. This indicates that the sensors are functional and data transmission using ESP32 is successful.

3)Need for bigger volume of helium (Bigger Balloon) to lift the system.

4)The structure must be reinforced using stronger material to ensure durability.

Final Test

In our final test using Prototype III on May 6, 2025, conducted after implementing key design improvements, the system achieved a stable and extended flight time of approximately 2 hours. The upgraded lift system, using two 2 meter diameter helium-filled balloons, successfully lifted the entire setup to a maximum altitude of 16 meters, allowing us to collect real-time environmental data continuously. Sensor readings were transmitted effectively over a wireless connection, confirming the reliability of our ESP32-based communication system. Additionally, the improved tether control and structural reinforcements provided much better balance and flight stability. This test validated the system's ability to operate autonomously, collect data for our reinforcement learning model, and simulate a real-world energy generation scenario.

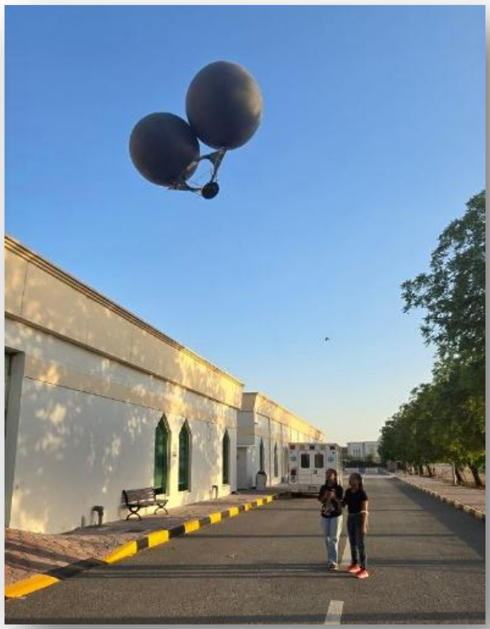


Fig: Final testing

Results and Analysis

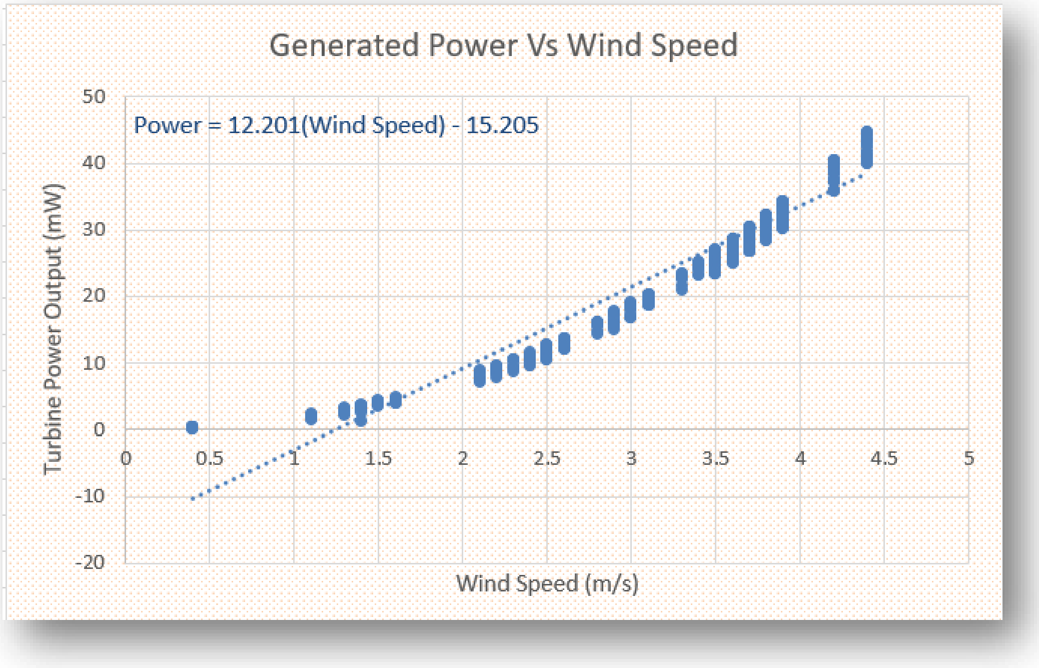


Fig. Relationship Between Wind Speed and Generated Power Output

The above graph illustrates the linear correlation between wind speed (measured in m/s) and the turbine's power output (in mW), based on over **1000** data points collected during prototype testing. The power output is calculated from the readings of current and voltage sensors, using the formula **P = IV**. The trendline indicates a strong positive relationship, following the equation: **Power = 12.201 × (Wind Speed) - 15.205**. As wind speed increases, the generated power output rises accordingly, validating the effectiveness of our airborne wind energy system in converting kinetic wind energy into usable electrical power. The linear trend also highlights the consistency of data and the reliability of sensor readings under varying atmospheric conditions.

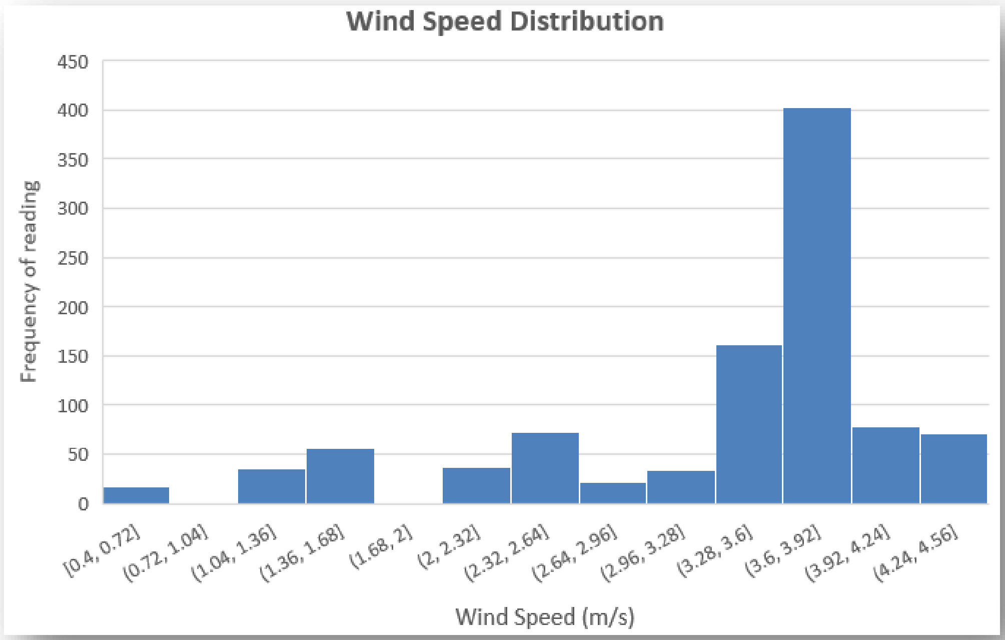


Fig. Wind speed frequency distribution

The bar graph above shows how often different wind speeds were recorded, measured in meters per second (m/s). On the x-axis, you can see the different wind speed ranges, and the y-axis shows how many times each range was recorded. Most of the wind speed readings are between **3.6** and **3.92 m/s**, however the overall wind speed distribution was irregular and in lower wind speed ranges, especially below 3 m/s.

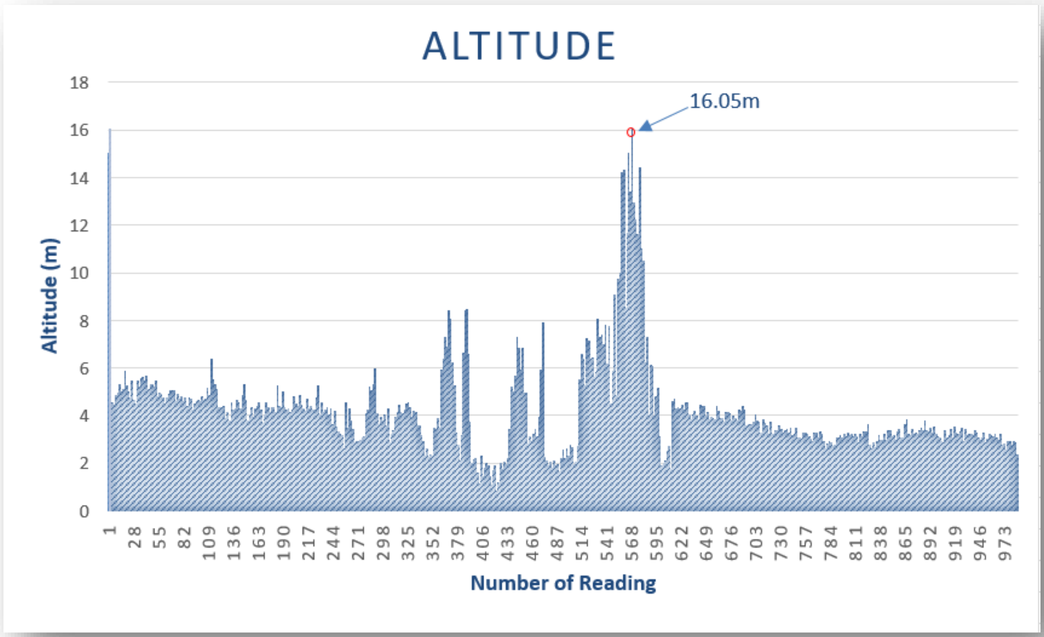


Fig. Wind speed frequency distribution

The graph above shows how altitude (in meters) changed throughout the experiment. For most of the readings, the altitude stays between about **3 and 5 meters**, with some small ups and downs. There are a few spots where the altitude spikes higher, especially around the middle of the graph, where it reaches the highest point of **16.05 meters**. After this peak, the altitude quickly drops back down to the usual range.

This data indicates that our system is capable of maintaining a relatively stable flight altitude during operation while also having the potential to reach significantly higher elevations when required. The ability to achieve an altitude demonstrates the system's lifting capacity and highlights the effectiveness of the balloon and tether design improvements made in the final prototype.

Simulated Environment and RL Agent

After collecting and analyzing the data from our prototype, we were able to extract valuable insights by plotting various graphs such as wind speed vs. power output, wind speed distribution, and altitude variation as seen in the above section. These visualizations helped us better understand the relationship between environmental conditions and system performance.

With this information in hand, the next step was to optimize the system using Reinforcement Learning (RL). To do that, we first cleaned and prepared the collected data, then fed it into our RL model for training. Rather than training the model in real-time using the physical prototype—which would be slow, difficult to control, and potentially damaging (since the agent will require to rapid and repeatedly move to explore the enviroment) ,so we chose to train the model in a simulated environment.

We carefully designed this simulated environment to closely mimic the behaviours and conditions of our real-world system. This included incorporating realistic wind speed variations, power output responses, and altitude adjustments, all based on our actual collected data by training a regression model. This regression model will provide accurate depiction of the enviroment by learning the patterns and relationships in the enviromental data.

Once the simulated environment was ready, we trained the RL model using this setup. The model learned to automatically adjust the system's altitude by controlling the "tether" (in the simulated environment) in a way that maximized power output. Through this process, we successfully demonstrated the potential of applying reinforcement learning to automate and optimize the airborne wind energy system without relying on manual adjustments.

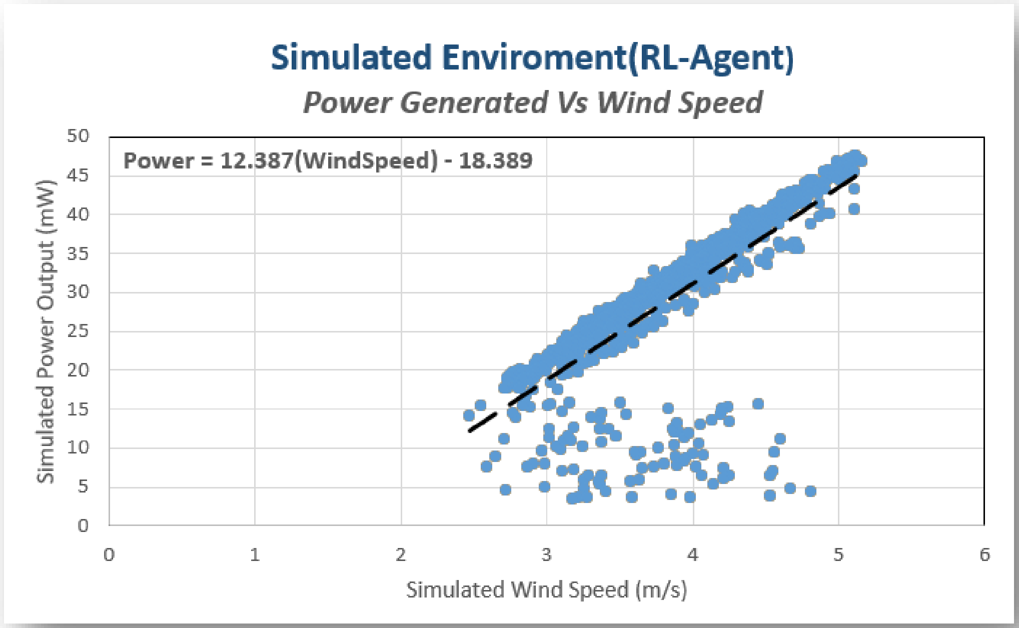


Fig. Relationship Between Wind Speed and Generated Power Output (In the simulated environment)

This scatter plot shows the relationship between simulated wind speed (in m/s) and power output (in mW) in our Reinforcement Learning (RL) environment. Each point represents a simulation data sample, and the trendline (Power = 12.387 × Wind Speed – 18.389) illustrates a strong positive correlation between wind speed and power output. The graph shows that as wind speed increases—particularly above 3 m/s—power output rises more consistently.

This result matches what we expected based on our real-world data and shows that the simulation provided an accurate representation of the real environment. The RL model learned to optimize power output by taking advantage of higher wind speeds in the simulation.

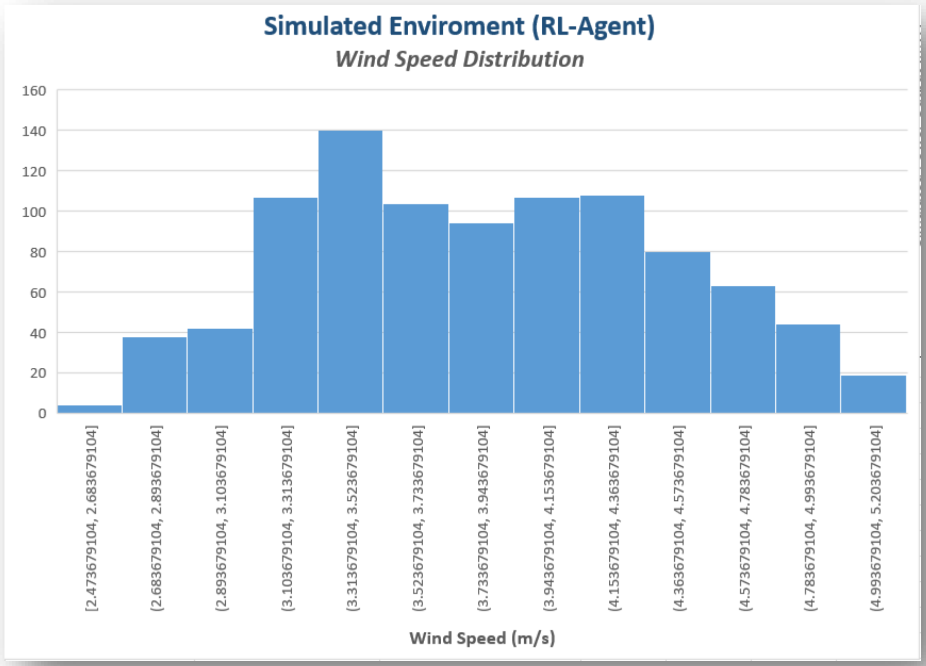
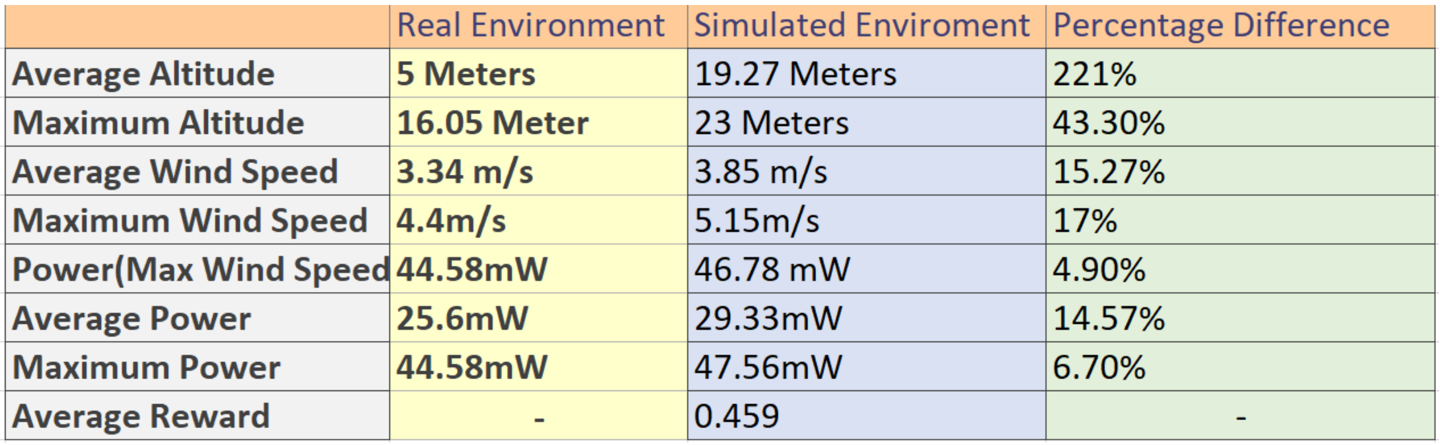


Fig. Wind speed frequency distribution in the simulated environment

The bar graph above shows how the wind speeds distribution (in m/s) in the RL simulation. Most wind speeds were between 3.1 and 4.1 m/s, with the 3.3–3.5 m/s range happening most often (about 140 times). Very low and very high wind speeds were rare. Compared to the real-world data without RL, the simulated environment had a more distributed wind speeds but at higher wind speed ranges than the system without RL Agent. This shows the RL-Agent was able to control the altitude to allow the system to maintain flight at higher wind speed range.

Comparison between results with and without RL agent

To evaluate the effectiveness of integrating a Reinforcement Learning (RL) agent into our Airborne Wind Energy (AWE) system, we conducted a comparative analysis between the system's performance in the real environment without RL model and in a simulated environment where the RL model was actively optimizing tether control.



	Real Environment	Simulated Enviroment	Percentage Difference
Average Altitude	5 Meters	19.27 Meters	221%
Maximum Altitude	16.05 Meter	23 Meters	43.30%
Average Wind Speed	3.34 m/s	3.85 m/s	15.27%
Maximum Wind Speed	4.4m/s	5.15m/s	17%
Power(Max Wind Speed	44.58mW	46.78 mW	4.90%
Average Power	25.6mW	29.33mW	14.57%
Maximum Power	44.58mW	47.56mW	6.70%
Average Reward	-	0.459	-

Table. Performance Comparison Between Real Environment and RL-Controlled Simulated Environment

Altitude Performance

- Average Altitude:
  In the real environment, the system maintained an average altitude of 5 meters, while in the simulation with RL, the altitude increased significantly to 19.27 meters, showing a 221% improvement.
  This increase suggests that the RL model was able to consistently identify and maintain higher, more energy-efficient altitudes—altitudes that were harder to achieve or maintain manually in the real-world tests.

- Maximum Altitude:
  The maximum altitude recorded in the real environment was 16.05 meters, compared to 23 meters in the simulation. This reflects a 43.3% increase, indicating that the RL agent was more effective at exploring higher altitudes to find optimal wind conditions.

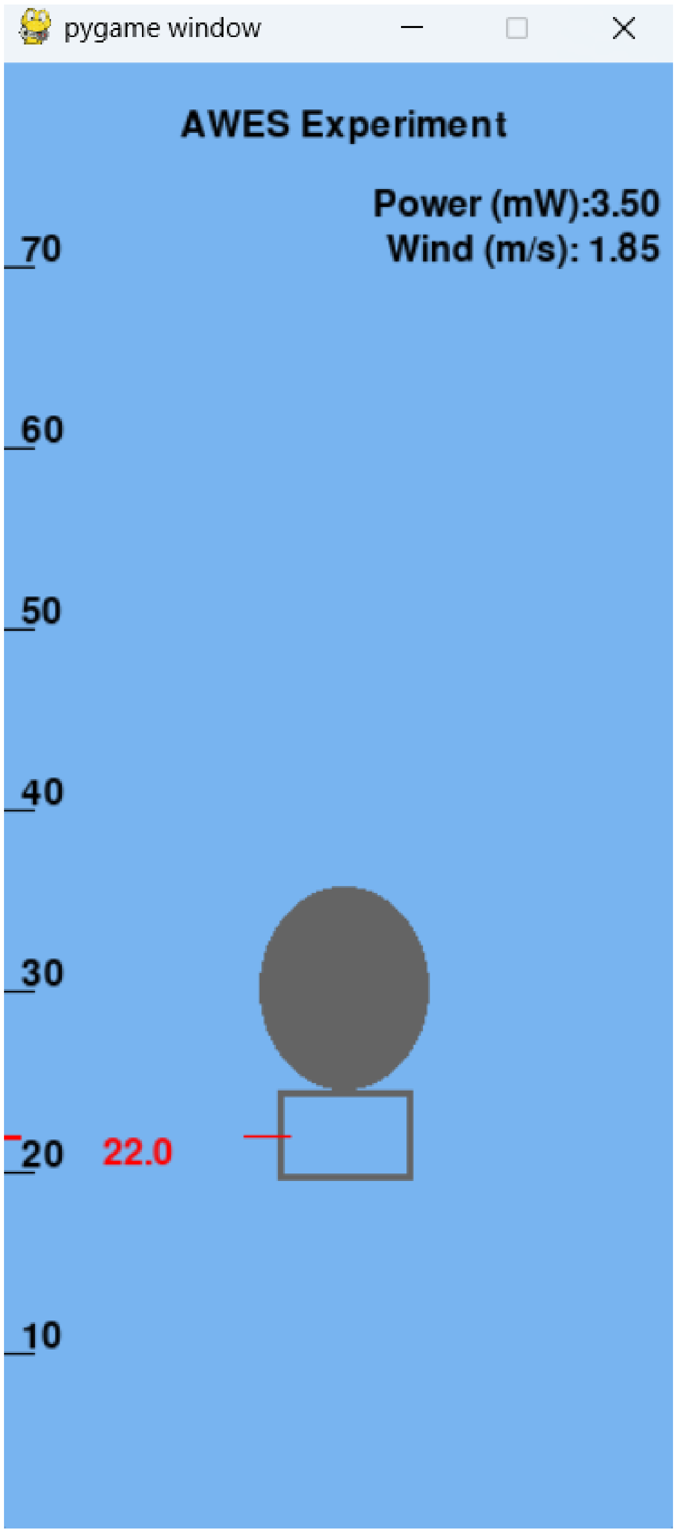
Wind Speed

- Average Wind Speed:
  Wind speed of the system with RL Agent averaged 3.85 m/s, compared to 3.34 m/s for the system without the agent, a 15.27% increase. This suggests the RL model learned to maintain the system at altitudes where wind speeds were consistently stronger.
- Maximum Wind Speed:
  The peak wind speed observed in the simulation was 5.15 m/s, compared to 4.4 m/s in the real environment, an increase of 17%.

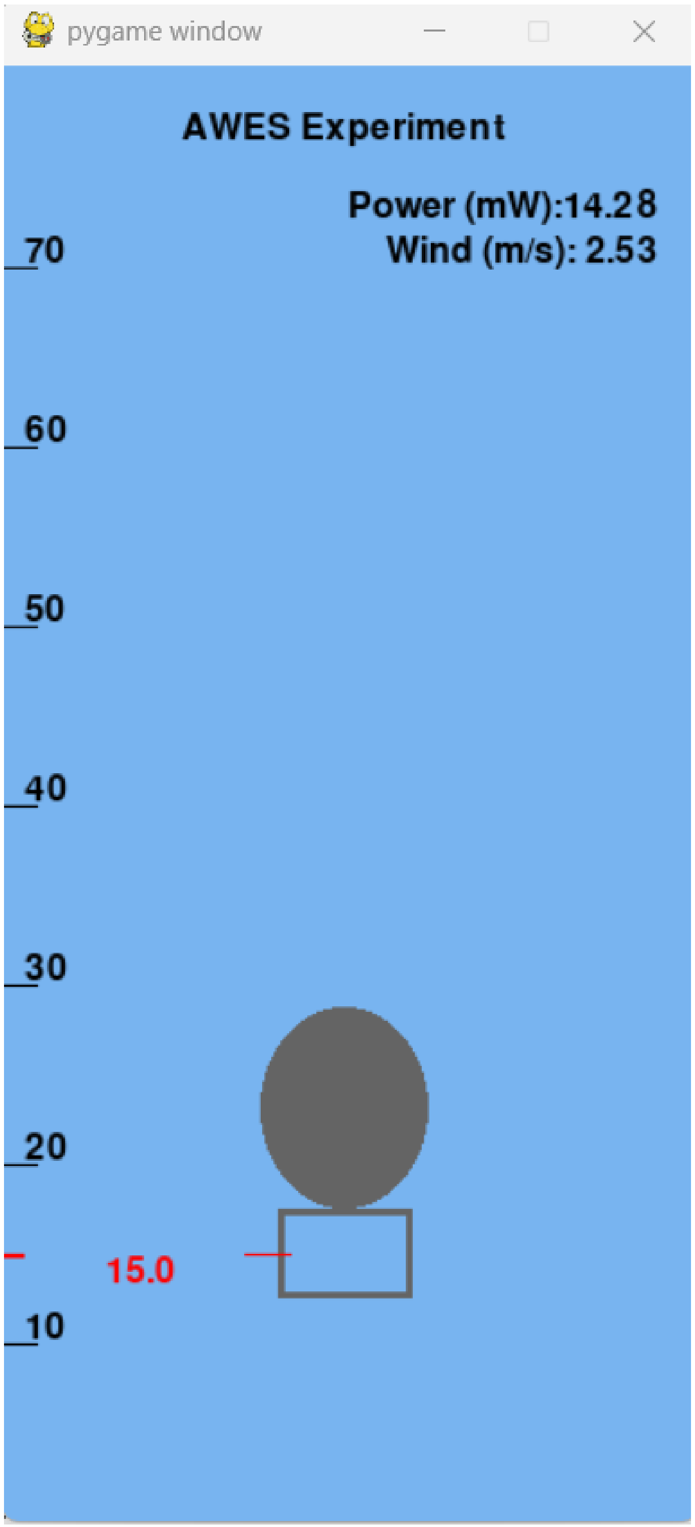
Power Output Comparison

- Power at Max Wind Speed:
  In the real environment, the system generated 44.58 mW at its peak wind speed. In the simulated environment, this increased slightly to 46.78 mW, a 4.9% improvement, suggesting the RL agent effectively exploited the relationship between altitude and wind strength.
- Average Power Output:
  The RL Agent achieved an average power output of 29.33 mW, compared to 25.6 mW in the system wihtout the agent, which was a 14.57% increase. This improvement is critical, as it demonstrates that the RL model not only found stronger wind currents but also maintained them more efficiently over time.(Refer to Wind Speed distribution table in simulated enviroment)

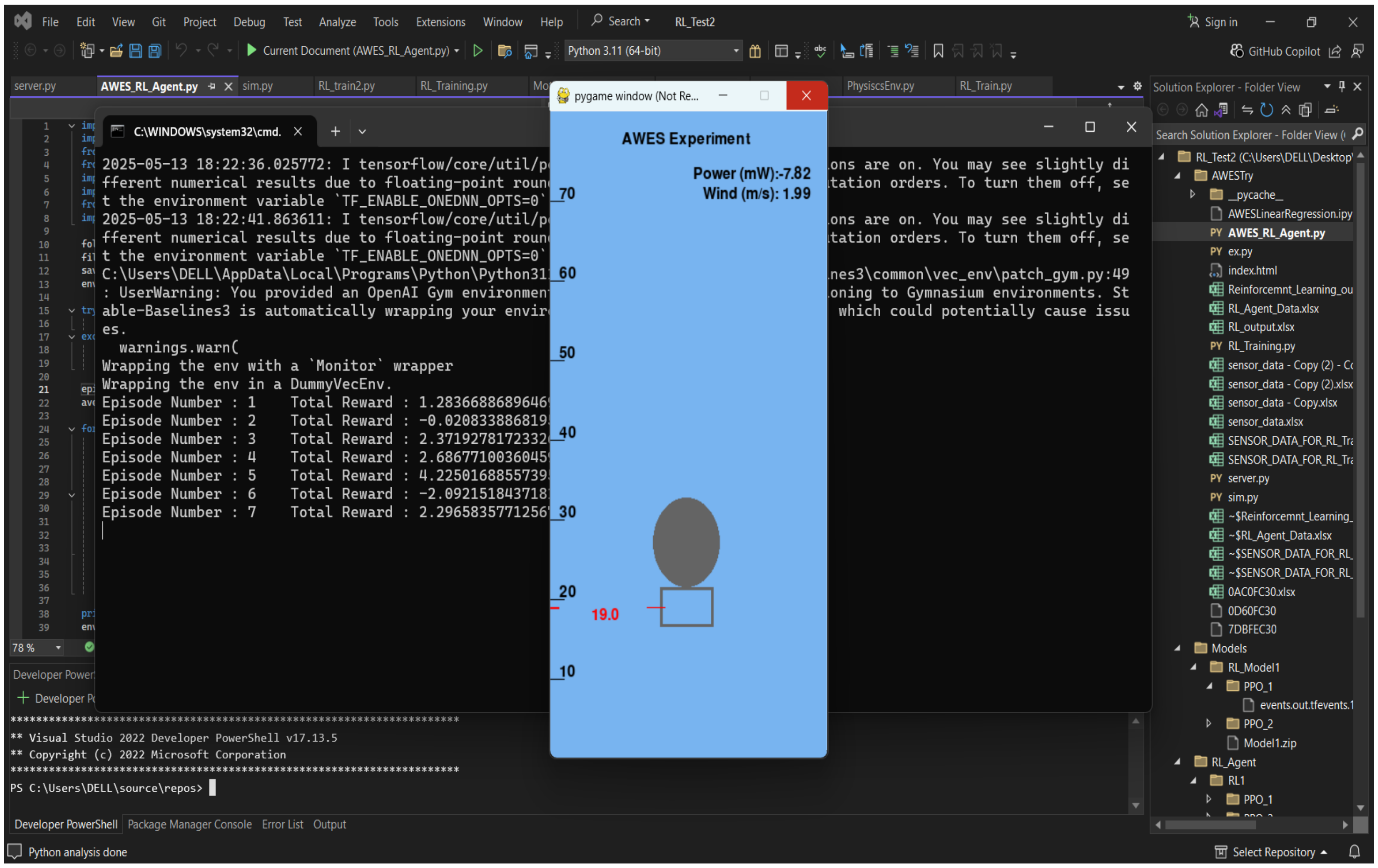
Pygame simulation of the Reinforcement Learning Agent



Power (mW):3.50
Wind (m/s): 1.85



Power (mW):14.28
Wind (m/s): 2.53



```
2025-05-13 18:22:36.025772: I tensorflow/core/util/p        ons are on. You may see slightly di
fferent numerical results due to floating-point roun        tation orders. To turn them off, se
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`
2025-05-13 18:22:41.863611: I tensorflow/core/util/p        ons are on. You may see slightly di
fferent numerical results due to floating-point roun        tation orders. To turn them off, se
t the environment variable `TF_ENABLE_ONEDNN_OPTS=0`
C:\Users\DELL\AppData\Local\Programs\Python\Python31        nes3\common\vec_env\patch_gym.py:49
: UserWarning: You provided an OpenAI Gym environmen        oning to Gymnasium environments. St
able-Baselines3 is automatically wrapping your envir         which could potentially cause issu
es.
  warnings.warn(
Wrapping the env with a `Monitor` wrapper
Wrapping the env in a DummyVecEnv.
Episode Number : 1    Total Reward : 1.28366886896469
Episode Number : 2    Total Reward : -0.02083388681 9
Episode Number : 3    Total Reward : 2.37192781723 32
Episode Number : 4    Total Reward : 2.68677100360459
Episode Number : 5    Total Reward : 4.22501688557395
Episode Number : 6    Total Reward : -2.09215184371 8
Episode Number : 7    Total Reward : 2.29658357712567
```

```
**********************************************************************
** Visual Studio 2022 Developer PowerShell v17.13.5
** Copyright (c) 2022 Microsoft Corporation
**********************************************************************
PS C:\Users\DELL\source\repos>
```

VII. Conclusion

This report brought together the journey of designing, building, and testing a small-scale Airborne Wind Energy (AWE) system. Starting from an initial concept, we developed multiple working prototypes, gradually improving their stability, lift, and data collection capabilities. Along the way, we focused not only on building the physical system, but also on making it smarter and more efficient through a data-driven approach.

Using real-time data collected during flight, we trained a reinforcement learning model in a simulated environment to optimize power output. The model learned how to control the system's altitude automatically, aiming to keep it in the best possible position for energy generation. Through testing and analysis, we were able to compare how the system performed with and without intelligent control, gaining meaningful insights into the benefits of automation.

Overall, we can conclude that integrating real-time data collection with reinforcement learning in an airborne wind energy system significantly improves its ability to adapt, optimize performance, and move toward more intelligent, efficient, and scalable renewable energy solutions.

VIII. Future Work

Future work could focus on deploying the trained reinforcement learning model in a real-world environment to test its performance under actual wind conditions. Additionally, implementing a working energy transfer system—from the airborne generator to ground-level storage—would complete the power cycle and demonstrate real-time usage. These steps would bring the system closer to real-world application and validate its efficiency beyond simulation.

IX. References

[1] “History of wind power,” EIA, Apr. 20, 2023. [Online]. Available: History of wind power - U.S. Energy Information Administration (EIA).

[2] AWESCO, "AWE Explained," AWESCO, [Online]. Available: https://awesco.eu/awe-explained/.

[3] Stephanie Mann, An Introduction to Airborne Wind Technology and Cost Reduction Trends, AP-0020, Feb. 2019.

[4] Airborne Wind Energy, Alternative Energy Tutorials. [Online]. Available: https://www.alternative-energy-tutorials.com/wind-energy/airborne-wind-energy.html.

[5] “Altaeros Energies Overview,” MIT Technology Review, [Online]. Available: https://www.technologyreview.com/altaeros;

[6] R. Rzouq, Rua', A. Mun'am Suboh, and N. Tamimi, “EXPLORING AIRBORNE WIND ENERGY: A COMPARATIVE STUDY AND THE POTENTIAL FOR IMPLEMENTATION IN JORDAN,” vol. 19, no. 15, 2024.

[7] “Benefits | Airbornewindeurope,” *Airbornewindeurope.org*, 2024. https://airbornewindeurope.org/benefits.

[8] H. S. Koppisetti, A. Mangalagiri, and C. Rayanki, "Airborne Wind Energy System," International Research Journal of Engineering and Technology (IRJET), vol. 09, no. 02, pp. 177-178, Feb. 2022.

[9] Energypedia, "Introduction to Airborne Wind Energy," Energypedia, Aug. 29, 2023. [Online]. Available: https://energypedia.info/wiki/Introduction_to_Airborne_Wind_Energy.

[10] International Renewable Energy Agency (IRENA), "Wind energy," IRENA, 2018. [Online]. Available: https://www.irena.org/technologies/wind-energy.

[11] History of Windmills, "Windmill history," History of Windmills, 2025. [Online]. Available: https://www.historyofwindmills.com.

[12] United States Environmental Protection Agency, Renewable Energy Fact Sheet: Wind Turbines, Aug. 2013.

[13] Arduino, "Introduction to Arduino," Arduino.cc, 2024. [Online]. Available: https://www.arduino.cc/en/Guide/Introduction.

[14] Dema Daoun, Fabiha Ibnat, Zulfikar Alom, Z. Aung, and Mohammad Abdul Azim, “Reinforcement Learning: A Friendly Introduction,” *Lecture notes in networks and systems*, pp. 134–146, Aug. 2021, doi: https://doi.org/10.1007/978-3-030-84337-3_11.

Project Management

Task	Assigned Member(s)	Duration	Timeline (Week #)	Notes
Background Research and Ideation	All members	2 weeks	Weeks 1–2	Concept, goals, system plan
Project Design	Blen, Hadi, Mhretewold	2 weeks	Weeks 3–4	prototype review
Arduino/ESP32 Coding	Mhretewold	2 weeks	Weeks 5–6	IoT cloud website setup
Wiring & Electronics	Bethel, Mhretewold, Hadi	2 weeks	Weeks 7–8	Circuit design, integrating sensors
Prototype Assembly	Blen, Kalkidan, Bethel	2 weeks	Weeks 7–8	Final physical build
Testing & Data Collection	All members	1 week	Week 10	Field test, sensor output
RL model Development and Training	Hadi	1 week	Week 11	Model training, results logging
Budget Management	Bethel	Ongoing	Weeks 1–12	Component tracking, cost sheet
PowerPoint Preparation and Presentation	Blen, Hadi, Mhretewold	Ongoing	Weeks 1-12	Drafting full presentation
Report Writing	Blen, Hadi, Kalkidan	Ongoing	Weeks 1–12	Drafting full report