Table of Contents

1. INTRODUCTION

This project mainly focuses on Runway Management System and various problems of it, when uncertainty take place due to the inefficient working of the internal parts of the plane which are believed to work efficiently and guide the controllers in various aspects with regard to speed, distance, temperature and humidity etc.

So, the agenda of this documentation is to show step-by-step mechanism of how it can be implemented with IoT and take further to solve these real life scenarios eternally.

1.1 PROBLEM DEFINITION

Ensuring safety while landing, this prototype attempts to facile landing process and regulate the speed accordingly with respect to decreasing distance between plane and runway line. Throwing light to DHT sensor, it is mainly to meant to read the temperature and humidity of the climate and report. If any unanticipated vagaries found, it reports to the pilots not to travel further and halt the landing process in the nearby airport.

1.2 BLOCK DIAGRAM

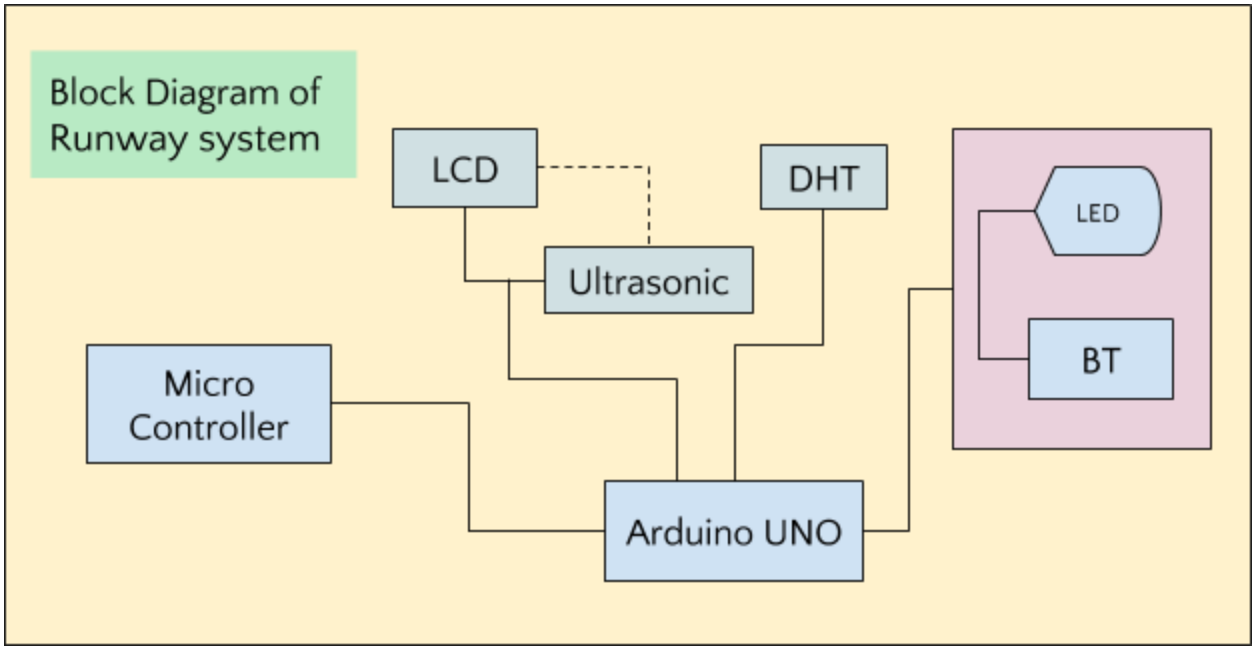

Fig 1: Block diagram of Runway Management System

2. SYSTEM ANALYSIS

2.1 FUNCTIONAL SPECIFICATION

The functions of this prototype is to suffice proactive safeness while the plane is landing. DHT and Ultrasonic sensors are highly leveraged for the needful function.

2.2 REQUIREMENT SPECIFICATION

2.2.1 Software Requirements

Operating System: Any Operating System, Windows 98 and above, Mac OS and Linux distributions.

Software: Arduino IDE

2.2.2 Hardware Requirements

The hardware components used for this prototype are as follows,

1. Arduino UNO Microcontroller
2. LCD Module 16 x 2
3. Bluetooth Module HC-05
4. DHT Sensor
5. Ultrasonic Sensor
6. Jumper Wires
7. Breadboard
8. Buzzer
9. LED
10. Resistors

Arduino UNO Microcontroller

Arduino UNO is an open source tool which is readily available. The main advantage of using this controller is that it is very easy to implement as it follows the object oriented programming paradigm for the implementation of the code. Any special functionality can be easily shared are interfaced by importing the required library files. It has 14 digital pins and 6 analog pins which can be used to input or output the data [1].

LCD Module

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. It is usually 16 pin interface [1].

Bluetooth Module HC-05

The LCDs have a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. It is usually 16 pin interface. The default mode is slave mode [1].

DHT Sensor

It is the temperature and humidity sensor. It measures humidity and temperature of the surrounding simultaneously [1].

Ultrasonic Sensor

Ultrasonic sensors are based on measuring the properties of sound waves with frequency above the human audible range. Ultrasonic sensors are non-intrusive in that they do not require physical contact with their target, and can detect certain clear or shiny targets otherwise obscured to some vision-based sensors [1].

Jumper Wires

A jumper wire is an electrical wire or group of them in a cable with a connector or pins at each end which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering.

Breadboard

A breadboard is a construction base for prototyping of electronics. A breadboard is used to build and test circuits quickly before finalizing any circuit design [1].

Buzzer

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric. Typical uses of buzzers and beepers include alarm devices, timers, and confirmation of user input such as a mouse click or keystroke [1].

LED

This has been used for validation.

Resistors

Resistors resist the flow of electricity and the higher the value of the resistor, the more it resists, and the less electrical current will flow through it [2].

3. SYSTEM DESIGN

3.1 MODULAR SPECIFICATION

MODULE 1: Measuring distance by Ultrasonic sensor.

Distance is measured and landing speed is regulated for safe landing.

MODULE 2: Distance numerals on the LCD screen.

Here we have considered the distance in multiples of 10 counting back from 50 to 20 then Approaching Minimum and then finally Successful.

MODULE 3: Temperature validations.

This temperature validation detects for any unfavorable climatic conditions and notifies pilots to land plane not on the destined runway but any near by runway line.

MODULE 4: Bluetooth Voice validations.

In this module, we have made use of AMR application (Android Meets Robot) integrating our bluetooth module. Now this takes voice as input and it will be stringified to meet conditions for validations.

3.2 COMPONENT DESCRIPTION

3.2.1 Arduino UNO

Arduino UNO is open-source computer hardware and software company, project and user community that designs and manufactures microcontroller-based kits for building digital devices and interactive objects that can sense and control objects in the physical world.

3.2.2 Bluetooth Module HC-05

We used an application as mentioned to power this module's functionality. With this application we send voice inputs and validate some of the conditions that plane is dependent on.

4. IMPLEMENTATION

4.1 CODING STANDARDS

Coding standards are a set of guidelines for a specific programming language that recommends programming style, practices, and methods for each aspect of a piece program written in the language. Coding conventions are not enforced by compilers. As a result, not following some or all of the rules has no impact on the executable programs created from the source code [3].

Since we are using IDE, this saves our time and faciles coding experience and sort of debugging.

4.2 Source Code

Reading climatic vagaries

```
#include <DHT.h>
DHT dht;
void setup() {
  Serial.begin(9600);
  Serial.println("Status\tHumidity (%)\tTemperature (C)\t(F)");
  dht.setup(8);
}
void loop() {
  fluctuations();
  delay(500);
}
void fluctuations() {
  delay(1000);
  float hum = dht.getHumidity();
  float temp = dht.getTemperature();
  //Serial.print(dht.getStautusString());
  Serial.print(dht.getStatusString());
  Serial.print("\t");
  Serial.print(hum, 1);
  Serial.print("\t\t");
  Serial.println(temp, 1);
  if (hum < 40 && temp < 12) {
    lcd.setCursor(0, 0);
    lcd.print("Land your Plane");
    lcd.setCursor(5, 1);
    lcd.print("nearby");
  } else { saferunway(); }
}
```

Measuring distance to regulate speed

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
const int trigPin = 9;
const int echoPin = 10;
long dura;
int dista;
void setup() {
  lcd.begin(16,2);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
void saferunway() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  dura = pulseIn(echoPin, HIGH);
  dista = dura * 0.034 / 2;
  //Serial.println(dista);
  if (dista < 60 && dista >= 50) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Distance = 50");
    //lcd.print(dista);
  } else if (dista < 50 && dista >= 40) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Distance = 40");
    //lcd.print(dista);
  } else if (dista < 40 && dista >= 30) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Distance = 30");
    //lcd.print(dista);
  } else if (dista < 30 && dista >= 20) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Distance = 20");
    //lcd.print(dista);
  } else if (dista < 20 && dista >= 6) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Approaching");
    lcd.setCursor(9, 1);
    lcd.print("Minimum");
```

```
    //lcd.print(dista);
  } else if (dista < 6 && dista >= 0) {
    voiceyBT();
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Successful");
    //lcd.print(dista);
  } else {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("No Signs");
    //lcd.print(dista);
  }
}
```

Bluetooth voice module

```
String voice;
void voiceyBT() {
  while (Serial.available()) {
    delay(10);
    char chr = Serial.read();
    if (chr == '#') { break; }
    voice += chr;
  }
  int len = voice.length();
  if (len > 0) {
    if (voice == "*landed") {
      digitalWrite(led, HIGH);
    } else if (voice == "*okay" || voice == "*ok") { digitalWrite(led, LOW); }
  }
  voice = "";
}
```

4.3 SCREENSHOTS

These are temperature readings through our program,

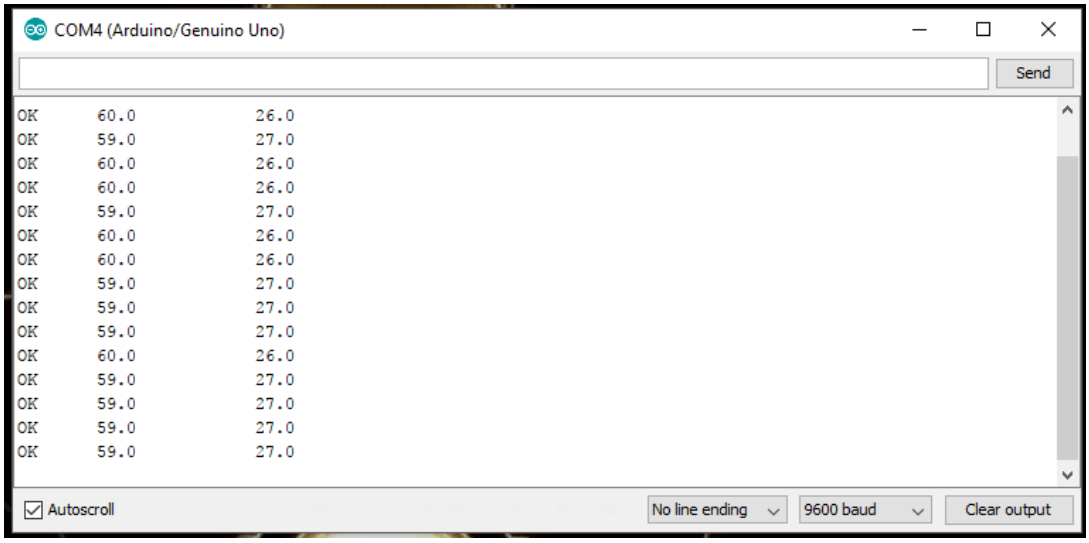

Fig 2: Temperature and Humidity readings.

Graphical representations

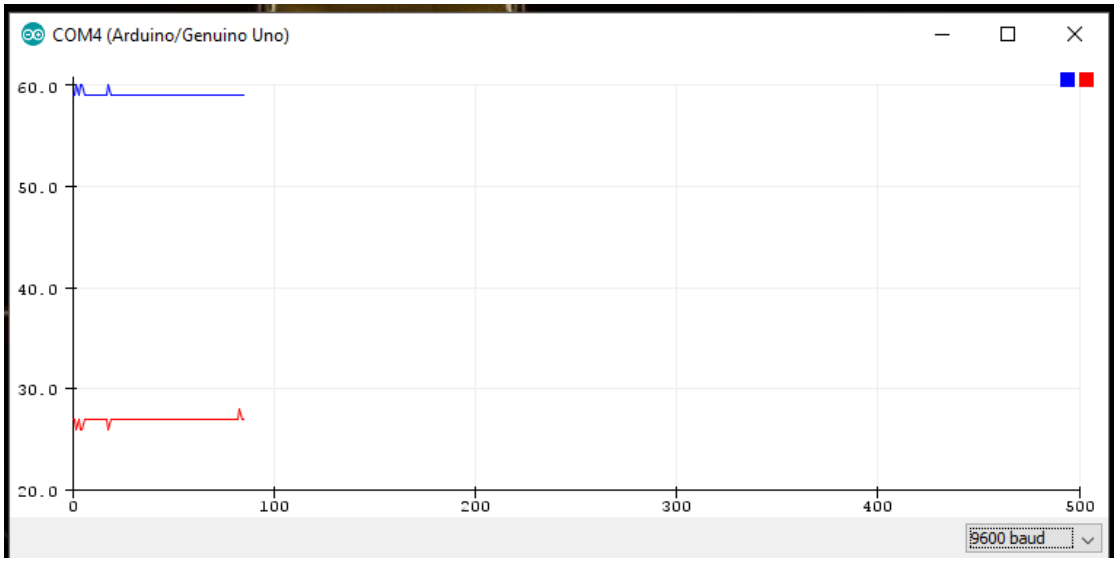

Fig 3: Graphical representation of Temperature and Humidity changes.

Red → Temperature, Blue → Humidity.

5. CONCLUSION

This project is just a prototype of how IoT can be used to automate the Runway Management System. Safe landing is the first and foremost thing that is to be taken care. Hence, keeping safety as the highest priority, this automated Runway System aids to solve the difficulties in some or major aspects contrived by internal segments of the plane with regard to distance notifying or temperature notifying that are highly favourable for safe landing. Therefore, this completes our project.

5.1 LIMITATIONS

- This prototype does not meet the constraints of the plane if it is meeting violent collision.

5.2 FUTURE ENHANCEMENT

- This project can further be automated that can meet all the constraints and validations with respect to safety and security.

6. REFERENCES

[1] Arduino Tutorials and References [Online] Available: https://www.arduino.cc/

[2] Resistor Wikipedia Article [Online] Available: https://en.wikipedia.org/wiki/Resistor

[3] Coding standards Wikipedia [Online] Available:
https://en.wikipedia.org/wiki/Coding_conventions