CHAPTER-1

INTRODUCTION

An aircraft without a human pilot on board is referred to as an unmanned aerial vehicle (UAV) or drone. The aircraft can be remotely controlled or can fly autonomously through software-controlled flight plans in their embedded systems working in conjunction with onboard computers. The technology for drones is improving every day. Drones are now also used in a wide range of civilian roles ranging from search and rescue, surveillance, traffic monitoring, weather monitoring and firefighting to personal drones and business drone-based photography, as well as videography, agriculture and even delivery services. In an effort to increase the drone's utility in emergency situations and to make it more efficient, we are demonstrating a drone using robotic arm, the robotic arm is constructed in a manner that lifts a 2kg weight and with the wireless control of arm to safely drop the payload in the intended location in the case of a flood or other natural disasters. To put it to effective use in a condition of emergency, we are going to design a robotic arm for drone. One of the application includes emergency transportation. Medicines are typically lightweight but are extremely valuable in terms of their impact on the health of global populations, thus making these items a potential focus for the development of drone-based delivery solutions. The main function of the arm is to lift and drop the payload. It provides vehicle lift and can be piloted remotely, can be expandable or recoverable, and can carry a lethal or nonlethal payload. In this instance, a drone is capable of lifting 2kg, making it useful for carrying emergency medications as well as food during floods. That also safely launches the payload at the destination.

CHAPTER-2

LITERATURE REVIEW

2.1 General Introduction

A literature survey is a fundamental practice, to understand and develop an idea. A literature survey not only summarizes the knowledge of the area or field but also gives us an idea of what had to be done. In the following section, we discuss the issues and works related to greenhouse monitoring systems based on IoT.

2.2 Literature Survey

2.2.1 Aerial grasping of cylindrical object [1]

It was proposed by Hoseong Seo . This paper concentrates on design of a vison based guidance command for aerial manipulation of a cylindrical object. This paper focused only on the utilisation of cylindrical items. The positioning of the arms in this document makes flying the drone difficult.

2.2.2 Drone applications in transportation [2]

It was proposed by Drazen Cvitanic. The author emphasises the numerous applications of UAVs for improving and facilitating transportation difficulties, as well as for planning, designing, and monitoring transportation and other infrastructure. Their application, in particular, could be critical in the transportation of goods in medical supply. There is no suitable lifting mechanism.

2.3 Summary

The above-discussed papers presented us with a broad perspective on how to tackle what we have aimed for and gave us the different possible approaches to follow to build our project i.e., Drone with robotic arm.

CHAPTER-3

Problem Statement and Objectives

3.1 Problem Statement

Transporting necessities like food, water, and medication over land can be exceedingly difficult or even impossible in an emergency circumstance like a flood, landslip, or other natural disaster. Roads may be blocked or completely flooded away, bridges may be hazardous and infrastructure may be destroyed or damaged. This may result in significant shortages of requirements, putting the communities in consideration at risk for a number of health and safety hazards.

3.2 Objectives

The core objective of this project is to design robotic arm for drone and increase its application in emergency transportation. That can also lift 2kg of payload with wireless control of arm and drop the payload to its intended location.

CHAPTER-4

System Requirements

4.1 Hardware Requirements

4.1.1 Arduino Uno

The Arduino Uno is a microcontroller board that is free source. The ATmega328P microprocessor powers the Uno, which includes 32KB of flash memory, 2KB of SRAM, and 1KB of EEPROM. There are 14 digital input/output pins on the board, six of which can be utilized as PWM outputs, and six analogue inputs. It also includes a reset button, an ICSP header, and a USB port. The Uno can be charged through USB or via an external power supply. The recommended voltage range is 7-12V. The Arduino Software (IDE), which is based on the Processing programming language, is used to program the Arduino Uno. It supports the programming languages C and C++. Figure 4.1 shows picture of Arduino uno.

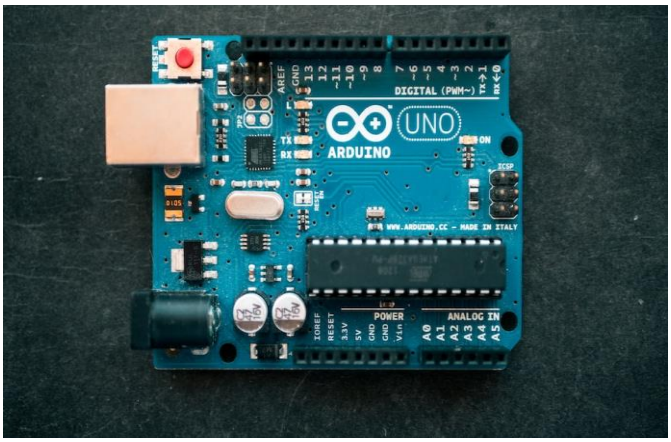

Figure 4.1: Arduino Uno

4.1.2 NRF24L01

The NRF24L01 is a well-known low-cost wireless transceiver module that runs at 2.4 GHz. It's common in wireless communication applications like remote control systems, wireless sensor networks, and industrial control systems. The module is built around the Nordic Semiconductor nRF24L01+ chip, which enables reliable wireless communication while consuming little power. The NRF24L01 module has an open-space range of up to 100 meters and can communicate with other NRF24L01 modules in either a point-to-point or multi-point mode. The module has a maximum data rate of 2 Mbps and supports multiple

channel. Figure 4.2 shows the picture of nrf24l01.

Figure 4.2: Nrf24l01

4.1.3 Joystick

A joystick is an input device that is often used in a digital environment to control the movement or direction of a cursor, object, or device. It is made out of a handheld stick or lever that can be moved in any direction, as well as extra buttons for controlling other operations. Joysticks are extensively employed in video games, flight simulators, and other applications that need precise movement control. They can also be used to control machines in industrial environments, such as cranes or robots. Depending on its intended application, joysticks can come in a variety of shapes and sizes. Some are intended to be held in one hand, while others are intended to be installed on a desk or panel. Figure 4.3 shows the picture of joystick.

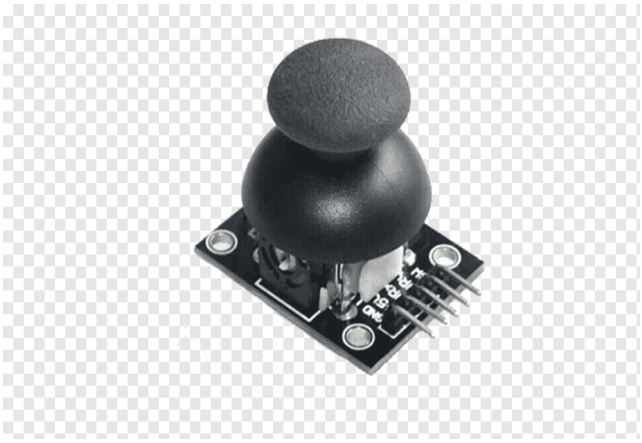

Figure 4.3: Joystick

4.1.4 Servomotor (MG995)

The MG995 servo motor is a type of rotary actuator that is commonly used in various applications, such as robotics, automation, and remote control systems. It is a high-quality servo motor that provides precise control of angular position, speed, and acceleration. The MG995 servo motor features a metal gear train, which makes it highly durable and able to withstand high torque loads. It also has a high torque output, with a maximum stall torque of around 10 kg-cm, and a voltage range of 4.8V to 7.2V. The motor can rotate up to 180 degrees, and it includes a feedback potentiometer that provides position feedback to the controller. Figure 4.4 shows the picture of MG995 servo motor.

Figure 4.4: MG995 Servo motor

4.1.5 3D printed model

A 3D model of a robotic arm using PLA material with 50% infill would result in a sturdy and strong arm that can effectively perform its intended functions. PLA (Polylactic Acid) is a widely used 3D printing material that is known for its strength and durability. It is a biodegradable and eco-friendly plastic. The 50% infill refers to the amount of material that is used to fill the interior of the arm. The infill density affects the strength and weight of the 3D model. A higher infill density results in a stronger and heavier model, while a lower density results in a lighter and less durable model. A 50% infill density is a good compromise between strength and weight. Figure 4.5 shows the picture of 3D printed arm.

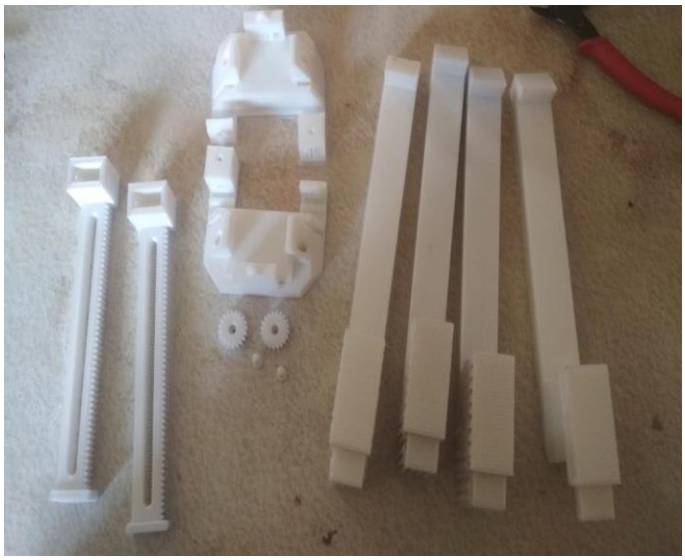

Figure 4.5: 3D printed arm

4.1.6 Power supply

A 9V DC power supply is a device that provides a constant and stable direct current (DC) voltage of 9 volts. It is commonly used to power electronic devices that require a 9V DC power source, such as musical instruments, effects pedals, routers, modems, and other small appliances. Figure 4.6 shows the picture of 9v battery.

Figure 4.6: 9v battery

4.2 Software Requirements

4.2.1 Flow chart and algorithm of Transmitter

Flow chart of transmitter:

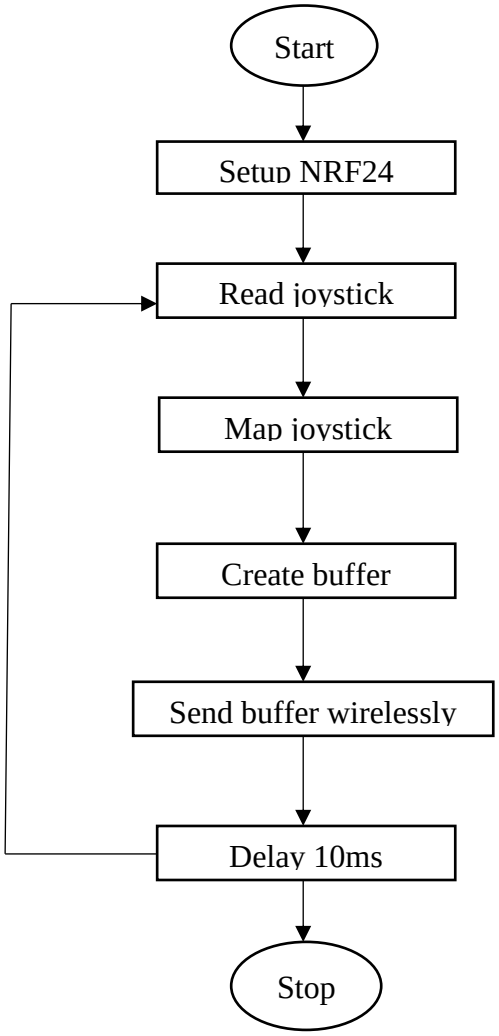

Figure 4.7: Flow chart of transmitter

Algorithm of transmitter:

Step 1: Start

Step 2: Set up the NRF24 module by calling its begin()

Step 3: Read the current position of the joystick X and Y axes using the analogRead() function.

Step 4: Map the joystick values to a range of 0 to 255 using the map() function.

Step 5: Create buffer of type char with a length of 2 bytes.

Step 6: Assign the mapped joystick values to the first and second elements of the buffer.

Step 7: Send the buffer wirelessly using the NRF24 module's write() method.

Step 8: Wait for a short period of time using the delay() function before sending more data.

Step 9: Repeat steps 2 & step 7 indefinitely.

Step 10: Stop

4.2.2 Flowchart and algorithm of receiver

Flowchart of receiver:

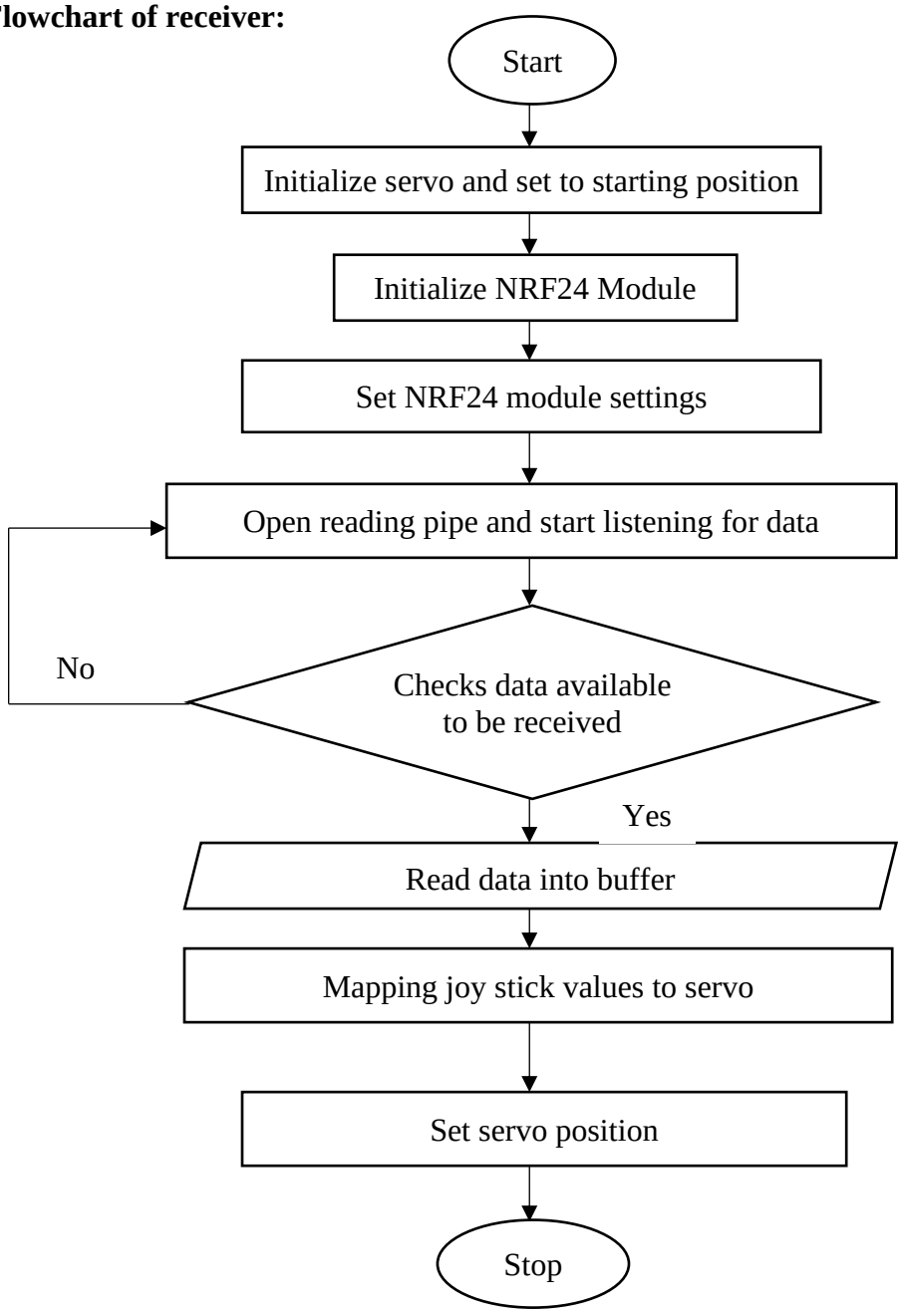

Figure 4.8: Flowchart of receiver

Algorithm of receiver:

Step 1: Start

Step 2: Initialize the NRF24 module and configure its settings.

Step 3: If data is available read the data into a buffer using read function of the RF24 object.

Step 4: Convert the buffer data to integer values.

Step 5: Map the joystick X values to a range of 0 to 180 to get the servo position.

Step 6: Set the servo position to the mapped value using the write function of the servo object.

Step 7: Repeat steps 2 to 6 to continuously check for incoming data and update the servo position accordingly.

Step 8: Stop

CHAPTER-5

Methodology and Implementation

5.1 Methodology

5.1.1 3D Design of Robotic arm

The 3D printed arm was designed in Tinker cad, a free online CAD software. The arm was designed with a focus on versatility and strength, with the ability to lift payloads up to 2kg. The arm was created using a 3D printer with PLA material, which is a widely used thermoplastic material known for its strength and durability. The arm was printed with an infill of 50%, which refers to the amount of material inside the printed object. This infill percentage provides a balance between strength and weight, making the arm robust while keeping it lightweight. The arm's design allows for easy attachment to the two MG995 servomotors, which are used to control the arm's movement. The 3D printed arm is a key component of the project, enabling lifting of payloads.

5.1.2 Interfacing of robotic arm

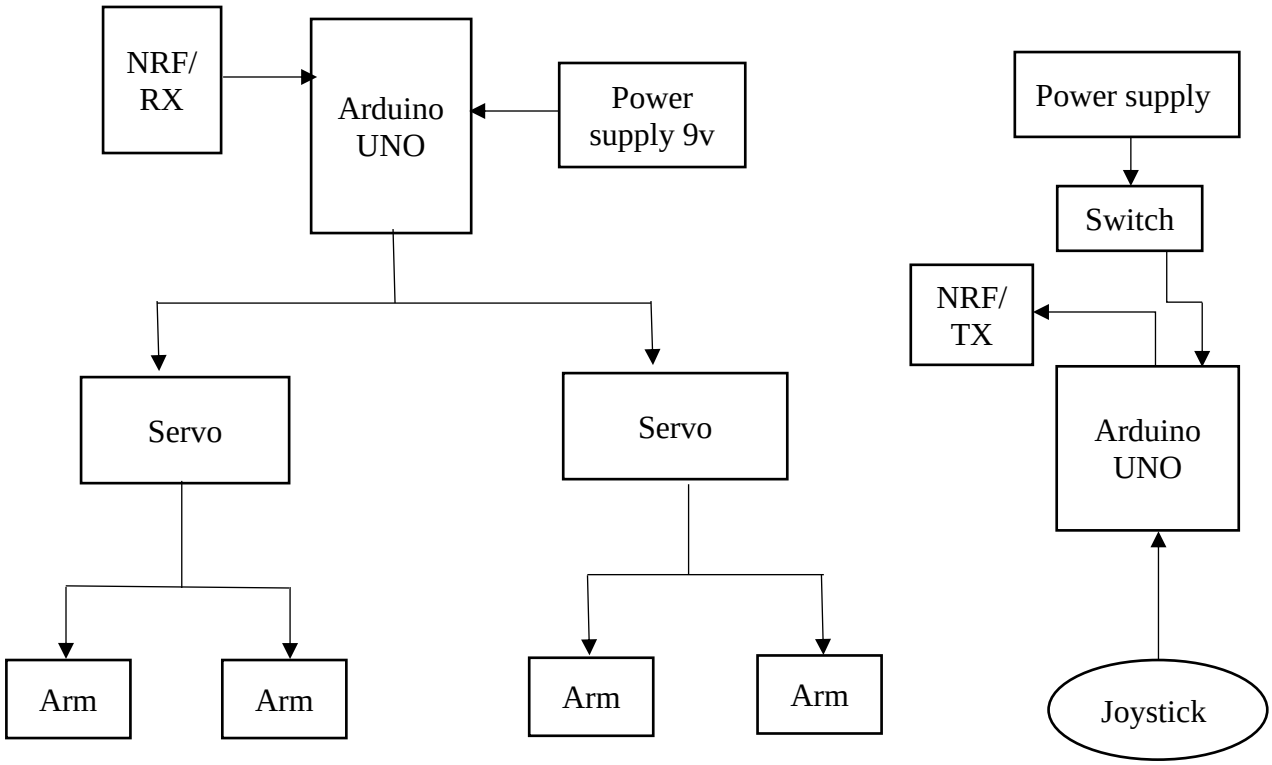

Figure 5.1: Interfacing of robotic arm

Designing the Arm: The first step in the project was to design the arm using Tinker cad software. The design focused on versatility and strength, and the arm is created with multiple interconnected parts to enable smooth and precise movement. The design was optimized for 3D printing and included cable routing for neat and efficient cable management.

3D Printing the Arm: After the design was finalized, the arm was 3D printed using PLA material with an infill of 50%. The 3D printing process is carefully monitored to ensure the accuracy and precision of the printed parts. After printing, the parts were carefully removed from the printer bed and assembled according to the design.

Attaching the Servomotors: The next step was to attach the two MG995 servomotors to the arm. The servomotors were mounted to the base of the arm and connected to the various arm segments using screws and brackets. The servomotors were also connected to the Arduino UNO using jumper wires.

Wireless Control: The arm was controlled wirelessly using a joystick and NRF wireless communication modules. The NRF modules were connected to the Arduino UNO using jumper wires, and the joystick was connected to the second Arduino UNO.

Testing: The final step in the project was to test the arm and control system. The arm was tested to ensure its ability to lift payloads up to 2kg and its precision in movement. The control system was also tested to ensure its reliability and accuracy. The project was tested and demonstrated successfully, and the arm was shown to be versatile and precise, making it an ideal candidate for use in a variety of applications.

In summary, the methodology for the project involved designing and 3D printing the arm, attaching the servomotors with their respective pin configurations, implementing a wireless control system with NRF modules and a joystick with their respective pin configurations, and testing the arm and control system for accuracy and precision. The methodology enabled the successful development of a versatile and precise arm for use in a variety of applications.

5.1.3 Circuit connection:

The pin configurations for the servomotors are as follows:

Servomotor 1:

Signal: Pin 3

VCC: 5V

GND: GND

Servomotor 2:

Signal: Pin 3

VCC: 5V

GND: GND

NRF Module 1:

CE: Pin 9

CSN: Pin 10

MOSI: Pin 11

MISO: Pin 12

SCK: Pin 13

NRF Module 2:

CE: Pin 9

CSN: Pin 10

MOSI: Pin 11

MISO: Pin 12

SCK: Pin 13

The joystick was connected to the second Arduino UNO as follows:

X-Axis: Analog Pin A0

Y-Axis: Analog Pin A1

Vin :5v.

CHAPTER-6

Results and Analysis

6.1 Calculations

The torque required to lift a 2kg payload using the Tower Pro MG995 servo motor depends on the distance between the pivot point of the payload and the point where the servo motor is attached to the 3D printed arm.

The distance between the pivot point and the point of attachment is 30cm for, the torque required to lift a 2kg payload can be calculated as follows:

Torque = Force x Distance

The force required to lift a 2kg payload is given by:

Force = Mass x Gravity

where Mass is the mass of the payload (2kg) and Gravity is the acceleration due to gravity (9.81 m/s^2). So, the force required to lift the 2kg payload is:

Force = 2kg x 9.81 m/s^2 = 19.62 N

The distance between the pivot point and the point of attachment is 30cm, or 0.3m. Therefore, the torque required to lift the payload is:

Torque = 19.62 N x 0.3m = 5.89 N-m

To calculate the power required to produce this torque, we can use the formula:

Power = Torque * Angular Velocity

The angular velocity of the MG995 servo motor at no load is approximately 0.1047 rad/s, which corresponds to a speed of 0.17 sec/60 degrees. Therefore:

Power = 5.89 * 0.1047 = 0.616 W

Assuming an operating voltage of 6V, the current required to produce this power is:

Current = Power / Voltage = 0.616 / 12 = 0.0513 A

So, the Tower Pro MG995 servo motor will require a current of approximately 0.0513A to lift a 2kg payload at a pivot point of 30cm using 3D printed arms.

6.2 Result

The Fig 6.1 shows wireless remote controller, which is used to wirelessly control the arm. The robotic arm is successfully interfaced with drone as is shown in fig 6.2. The joystick data is successfully transmitted wirelessly using the NRF24 module and received by the receiving device. The servo motor is successfully moved to the desired position based on the joystick values received, which is attached to the robotic arm.

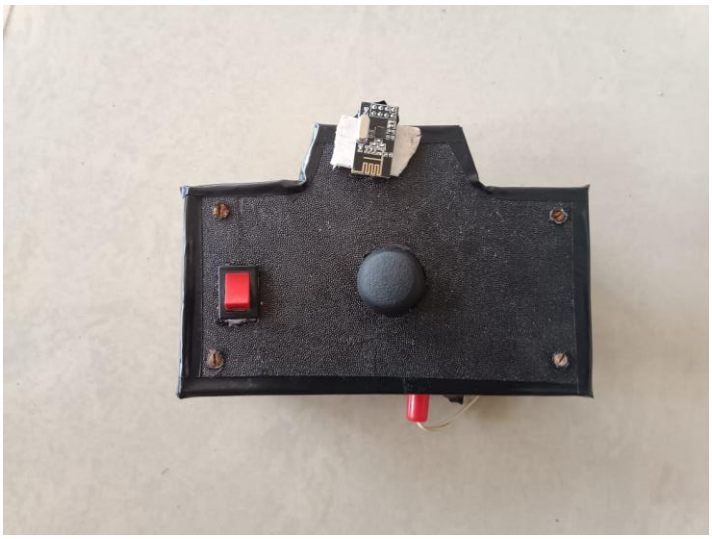

Fig 6.1 :Wireless remote control of arm.

Overall, the system was able to effectively transmit and receive data wirelessly, and successfully control a servo motor. The system was also able to accurately map the joystick values to the desired range and move the servo motor to the corresponding position. The robotic arm is successfully designed and built with the ability to lift up to 2kg of payload.

Fig 6.2 : Final model of drone with robotic arm.

CONCLUSION

In conclusion, this project successfully demonstrated drone with robotic arm that can be controlled wirelessly using a joystick and NRF wireless communication modules.

The project's objective was to develop a robotic arm that can be used for emergency medicine transportation during floods, and the arm is capable of lifting payloads up to 2kg.

REFERENCES

[1]. Seo, Hoseong, Suseong Kim, and H. Jin Kim. "Aerial grasping of cylindrical object using visual servoing based on stochastic model predictive control." 2017 IEEE international conference on robotics and automation (ICRA). IEEE, 2017.

[2]. Cvitanić, Dražen. "Drone applications in transportation." 2020 5th international conference on smart and sustainable technologies (SpliTech). IEEE, 2020..

APPENDIX

Code for transmitter:

```
 #include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
// Set up the NRF24 module
RF24 radio(9, 10);
// Define the address for the communication channel
const uint64_t address = 0xF0F0F0F0E1LL;
// Set up the joystick pins
const int joyXPin = A0;
const int joyYPin = A1;
void setup() {
  // Initialize the NRF24 module
  radio.begin();
  radio.setAutoAck(true);
  radio.setChannel(115);
  radio.setPALevel(RF24_PA_MAX);
  radio.setDataRate(RF24_2MBPS);
  radio.openWritingPipe(address);
}
void loop() {
  // Read the joystick values
  int joyXVal = analogRead(joyXPin);
  int joyYVal = analogRead(joyYPin);
```

```
 // Map the joystick values to a range of 0 to 255
  int joyXValMapped = map(joyXVal, 0, 1023, 0, 255);
  int joyYValMapped = map(joyYVal, 0, 1023, 0, 255)
  // Create a buffer with the joystick values
  char buffer[2];
  buffer[0] = (char) joyXValMapped;
  buffer[1] = (char) joyYValMapped;
  // Send the buffer wirelessly using the NRF24 module
  radio.write(&buffer, sizeof(buffer));
  // Wait a short time before sending more data
delay(10);
}
```

Code for receiver:

```
#include <Servo.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
// Declare the servo object
Servo myservo;
// Set up the NRF24 module
RF24 radio(9, 10);
// Define the address for the communication channel
const uint64_t address = 0xF0F0F0F0E1LL;
void setup() {
  // Initialize the servo and set it to the starting position
```

```
  myservo.attach(3);
  myservo.write(90);
  // Initialize the NRF24 module
  radio.begin();
  radio.setAutoAck(true);
  radio.setChannel(115);
  radio.setPALevel(RF24_PA_MAX);
  radio.setDataRate(RF24_2MBPS);
  radio.openReadingPipe(1, address);
  radio.startListening();
}
void loop() {
  // Check if there is any data available to be received
  if (radio.available()) {
    // Read the data into a buffer
    char buffer[2];
    radio.read(&buffer, sizeof(buffer));
    // Convert the buffer data to integer values
    int joyXValMapped = (int) buffer[0];
    int joyYValMapped = (int) buffer[1];
    // Map the joystick values to a range of 0 to 180
    int servoPos = map(joyXValMapped, 0, 255, 0, 180);
    myservo.write(servoPos);
  }}
```