CONTROLLING DRONES

A COURSE PROJECT REPORT

By

SRINATH M (RA2011003010176)
GUNNU JAI RAJ (RA2011003010171)
AKHIL RAJESH (RA2011003010174)
SHRIDHARSHAN V.A.K (RA2011003010180)

Under the guidance of

Dr. VINOD

In partial fulfilment for the Course

of

18CSC302J - COMPUTER NETWORKS

in CTECH

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chengalpattu District

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project report " **CONTROLLING DRONES** " is the bonafide work of **SRINATH M** (RA2011003010176), **GUNNU JAI RAJ** (RA2011003010171), **AKHIL RAJESH** (RA2011003010174) and **SHRIDHARSHAN V.A.K** (RA2011003010180) who carried out the project work under my supervision.

SIGNATURE

Dr. VINOD
CTECH
SRM Institute of Science and Technology

ABSTRACT

Nowadays, most hear “drone,” they think expensive military aircrafts or small consumer toys. However, the future of drones will actually be shaped by practical commercial applications, due to their ability to drive efficiency and data analytics. Drones and their total addressable market and highlights how their capabilities can be leveraged in commercial business operations.

When aspect of drone comes the main thing is to know how to operate it.

This is the base of our idea by using our network programming skills, we will try to make operating easy and simple.

Here, taking client as an remote controller and sever as an drone.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr.Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our Course project Faculty **Dr. VINOD , CTECH,** for his/her assistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD Dr. Pushpalatha, Professor, Computing technologies** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

INTRODUCTION

Nowadays, most hear “drone,” they think expensive military aircrafts orsmall consumer toys. However, the future of drones will actually be shaped bypractical commercial applications, due to their ability to drive efficiency and data analytics. Drones and their total addressable market and highlights how their capabilities can be leveraged in commercial business operations.
When aspect of drone comes the main thing is to know how to operate it.
This is the base of our idea by using our network programming skills, we will try to make operating easy and simple.
Here, taking client as an remote controller and sever as an drone.

Drones require a controller, which lets the operator use remote controls to launch, navigate and the aircraft. Controllers communicate with the drone using radio waves, such as Wi-Fi. Data link uses radio-frequency (RF) transmission to transmit and receive information to and from the UAV. Drones Internet Protocols (IP) to communicate and it involves many various protocols. Using TCP is very convenient as its API is straight forward and its connectivity with hardware components is well supported. In TCP for every request sent by sender needs an acknowledgment from the receiver to continue further and send another request. If a data packet lost in between it becomes difficult to the sender to send the request again and again because we have a continuously operating drone on the other hand. So, losing packets and trying to send the again will affect the functioning and efficiency of the drone.

So, we are using UDP to communicate because UDP has a very simple model: UDP allows you to send so-called data grams (packets) from one machine to another. These data grams or packets are received at the other end as the same packets (unless they've been lost on the way). And UDP don’t need any return acknowledgment to proceed further to send another request. So, we use UDP to communicate with server which is in the drone.

2 LITERATIRE SURVEY

TCP, or Transmission Control Protocol. Just about anything communicating through the internet uses TCP at the transport layer, and for a good reason, as using TCP is extremely convenient. The API for using TCP is rather straightforward, and TCP is well supported by all the hardware that the internet traffic has to travel through to get from one device on the internet to another. Using TCP is simple. Once you've opened a connection, you can write data into a so-called socket, and the other end can read that data from its socket. TCP makes sure that the exact data that is written in one end arrives at the other end. It hides a lot of complexity. TCP is based on top of IP, and lower-level IP data may not arrive in the order it is sent. It may, in fact, never arrive. But TCP hides this complexity. It is modeled after normal Unix pipes. TCP also manages the throughput; it constantly adapts the rate at which data is transmitted to best utilize the available bandwidth. TCP does so much magic to pull off this trick that the de-facto standard books on TCP are three volumes with a total of more than 2,556 pages of detailed explanations: TCP/IP Illustrated: The Protocols, The Implementation, TCP for Transactions.

As it is in the starting stage of the drone usage they used TCP to communicate via Wi-Fi. After facing the complications while using TCP people and researchers understood that usage of UDP might work well. So they started using UDP and which have shown a way lot of good results. So, keeping that in mind we took User Data-gram Protocol (UDP) as base and developed a socket program which is far effective than using TCP.

REQUIREMENTS

From the given scenario, we draw the following requirements:

1. AWS console.
2. Required knowledge about TCP and UDP (internet protocols).

DESIGN

The IP address of the drone is 192.168.1.1 and there are three ports we can use to connect over UDP:

Navigation Data Port = 5554

On-Board Video Port = 5555

AT Command Port = 5556

We need to use the AT Command Port to send commands to the drone. We can use the Navigation Data Port to retrieve data back from the drone. On-Board Video Port is Used to get video data. In this project we mainly concentrated on operating the drone which comes under sending commands that is AT Command Port.

Move_UP
Move_RIGHT
Move_DOWN
Move_LEFT

We use the above commands for designing the drone controlling.

IMPLEMENTATION

SERVER:

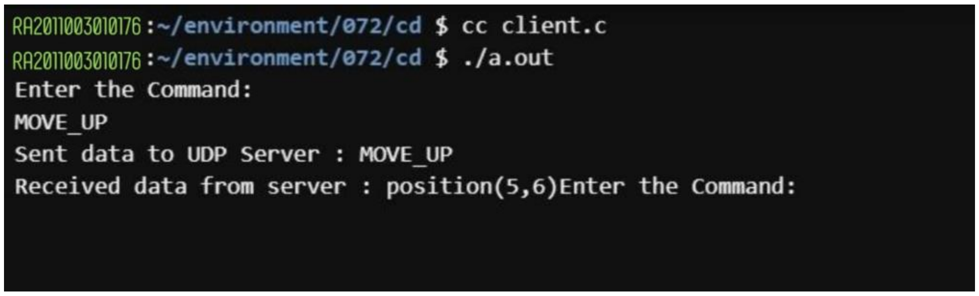
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>

#define CLIENT_CONN_WAIT_QUEUE_SIZE 3

int listenAndAcceptClient(int port) {
    struct sockaddr_in local_addr;
    struct sockaddr_in client_addr;
    socklen_t client_addr_len = sizeof(client_addr);
    int listen_sockfd;
    int sockfd;

    memset(&local_addr, 0, sizeof(local_addr));
    local_addr.sin_family = AF_INET;
    local_addr.sin_port = htons(port);
    local_addr.sin_addr.s_addr = htonl(INADDR_ANY);

    if((listen_sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket");
        return -1;
    }

    if(bind(listen_sockfd, (struct sockaddr*)&local_addr, sizeof(local_addr)) < 0) {
        perror("bind");
        return -1;
    }

    if(listen(listen_sockfd, CLIENT_CONN_WAIT_QUEUE_SIZE) < 0) {
        perror("listen");
        return -1;
    }

    if((sockfd = accept(listen_sockfd, (struct sockaddr*)&client_addr, &client_addr_len)) < 0) {
        perror("accept");
        return -1;
    }

    close(listen_sockfd);
    return sockfd;
}

int sendToClient(int sockfd, char *data, int data_len) {
    int total_send_size = 0;
    int send_size;

    do {
        send_size = send(sockfd, &data[total_send_size], data_len - total_send_size, 0);
        total_send_size = send_size;
        if(send_size < 0) {
            perror("send");
            close(sockfd);
            return -1;
        }
    } while(total_send_size < data_len);

    return 0;
}

int receiveFromClient(int sockfd, char *data, int data_len) {
    int total_receive_size = 0;
    int receive_size;

    do {
        receive_size = recv(sockfd, &data[total_receive_size], data_len - total_receive_size, 0);
        total_receive_size = receive_size;
        if(receive_size < 0) {
            perror("recv");
            close(sockfd);
            return -1;
        }
    } while(total_receive_size < data_len);

    return 0;
}

void moveDrone(char *move, char *position) {
    static int x = 5;
    static int y = 5;

    if(strcasecmp(move, "Move_UP") == 0) {
        if(y > 0) y--;
    } else if(strcasecmp(move, "Move_DOWN") == 0) {
        if(y < 10) y++;
    } else if(strcasecmp(move, "Move_LEFT") == 0) {
        if(x > 0) x--;
    } else if(strcasecmp(move, "Move_RIGHT") == 0) {
        if(x < 10) x++;
    }

    sprintf(position, "(%d,%d)", x, y);
}

int main(int argc, char *argv[]) {
    int sockfd;
    int port;

    char move[16] = {};
    char position[16] = {};

    if(argc != 2) {
        printf("Usage: ./server [port]\n");
        exit(1);
    }

    printf("Server starts...\n");

    port = atoi(argv[1]);

    if((sockfd = listenAndAcceptClient(port)) < 0) {
        perror("listenAndAcceptClient");
        exit(1);
    }

    while(1) {
        if(receiveFromClient(sockfd, move, sizeof(move)) < 0) {
            perror("move");
            exit(1);
        }

        if(strcmp(move, "exit") == 0)
            break;

        printf("Move: %s\n", move);
        moveDrone(move, position);
        printf("Position: %s\n", position);

        if(sendToClient(sockfd, position, sizeof(position)) < 0) {
            perror("position");
            exit(1);
        }
    }

    printf("Server terminates.\n");

    close(sockfd);
    return 0;
}
```

CLIENT:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>

int connectToServer(char *addr, int port) {
    struct sockaddr_in serv_addr;
    int sockfd;

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(port);
    serv_addr.sin_addr.s_addr = inet_addr(addr);

    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket");
        return -1;
    }

    if(connect(sockfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) < 0) {
        perror("connect");
        return -1;
    }

    return sockfd;
}

int sendToServer(int sockfd, char *data, int data_len) {
    int total_send_size = 0;
    int send_size;

    do {
        send_size = send(sockfd, &data[total_send_size], data_len - total_send_size, 0);
        total_send_size = send_size;
        if(send_size < 0) {
            perror("send");
            close(sockfd);
            return -1;
        }
    } while(total_send_size < data_len);

    return 0;
}

int receiveFromServer(int sockfd, char *data, int data_len) {
    int total_receive_size = 0;
    int receive_size;

    do {
        receive_size = recv(sockfd, &data[total_receive_size], data_len - total_receive_size, 0);
        total_receive_size = receive_size;
        if(receive_size < 0) {
            perror("recv");
            close(sockfd);
            return -1;
        }
    } while(total_receive_size < data_len);

    return 0;
}

int readMove(char *filename, char *move) {
    static FILE *file = NULL;

    if(file == NULL)
        file = fopen(filename, "r");

    if(feof(file)) {
        fclose(file);
        return -1;
    } else {
        if(fscanf(file, " %s", move) < 0)
            return -1;
    }

    return 0;
}

int writePosition(char *filename, char *position) {
    FILE *file = fopen(filename, "w");

    fprintf(file, "%s\n", position);
    fclose(file);

    return 0;
}

int main(int argc, char *argv[]) {
    int sockfd;
    char *address;
    int port;

    char move[16] = "";
    char position[16] = "";

    if(argc != 3) {
        printf("Usage: ./client [server_ip] [port]\n");
        exit(1);
    }

    printf("Client starts...\n");

    address = argv[1];
    port = atoi(argv[2]);

    if((sockfd = connectToServer(address, port)) < 0) {
        perror("connectToServer");
        exit(1);
    }

    while(readMove("moves.txt", move) != -1) {
        printf("Move: %s\n", move);

        if(sendToServer(sockfd, move, sizeof(move)) < 0) {
            perror("move");
            exit(1);
        }

        if(receiveFromServer(sockfd, position, sizeof(position)) < 0) {
            perror("position");
            exit(1);
        }

        writePosition("position.txt", position);
        printf("Position: %s\n", position);
    }

    if(sendToServer(sockfd, "exit", sizeof(move)) < 0) {
        perror("exit");
        exit(1);
    }

    printf("Client terminates.\n");

    close(sockfd);
    return 0;
}
```

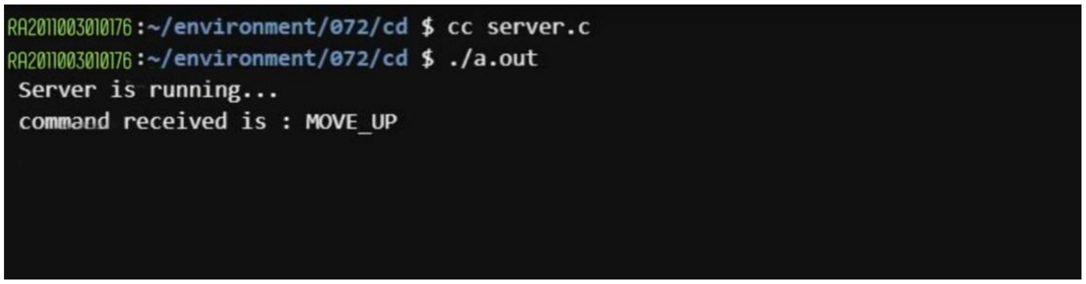
RESULTS

CLIENT SIDE:



```
RA2011003010176:~/environment/072/cd $ cc client.c
RA2011003010176:~/environment/072/cd $ ./a.out
Enter the Command:
MOVE_UP
Sent data to UDP Server : MOVE_UP
Received data from server : position(5,6)Enter the Command:
```

SERVER SIDE:



```
RA2011003010176:~/environment/072/cd $ cc server.c
RA2011003010176:~/environment/072/cd $ ./a.out
 Server is running...
 command received is : MOVE_UP
```

CONCLUSION

By using internet protocols the connection establish between the client and sever which brings the efficiency of drone.

REFERENCES

1. https://www.objc.io/issues/8-quadcopter/communicating-with-the-quadcopter/
2. https://www.geeksforgeeks.org/user-datagram-protocol-udp/
3. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8433880/
4. https://thesai.org/Downloads/Volume12No6/Paper_16-A_New_Communication_Protocol_for_Drones.pdf