MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF KAZAKHSTAN

Kazakh National Research Technical University named after K.I.Satbayev

A.Burkitbaev Institute of Industrial Automation and Digitalization

Department of Industrial Engineering

Ibray.S.I

# Design of a unmanned aerial vehicle control system

**DIPLOMA WORK**

5B071200-Mechanical Engineering

Almaty 2020

MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF KAZAKHSTAN

Kazakh National Research Technical University named after K.I.Satbayev

A.Burkitbaev Institute of Industrial Automation and Digitalization

Department of Industrial Engineering

**APPROVED FOR DEFENSE**

Head of the Industrial

Engineering Department, PhD

Arymbekov B.S.

_________________________

“_____”_______________2020

**DIPLOMA WORK**

Topic: " Design of a unmanned aerial vehicle control system "

5B071200-Mechanical Engineering

Performed by Ibray.S.I

Reviewer

_______________________

_______________________

"_____" __________2020

Scientific adviser

Candidate of Technical Sciences,

Associate Professor

______Isametova M.E

"_____" __________2020

Almaty 2020

MINISTRY OF EDUCATION AND SCIENCE OF THE REPUBLIC OF KAZAKHSTAN

Kazakh National Research Technical University named after K.I.Satbayev

A.Burkitbaev Institute of Industrial Automation and Digitalization

Department of Industrial Engineering

**APPROVED FOR DEFENSE**

Head of the Industrial

Engineering Department, PhD

Arymbekov B.S.

________________________

“_____”_______________2020

**TASK**

**for completing the diploma work**

For student:*Ibray Sergazy Isabekuli*
Topic: " *Design of a unmanned aerial vehicle control system*"
Approved *by the order of university rector №762-b from "27" January 2020*
Deadline for completion the work *"24" May 2020*
Initial data for the diploma project: Control System JANSER
Summary of the diploma work:

*a) JANSER control system;*
*b) Simulation for quadcopter;*
*c) AI and ANN .*

List of graphic material: tables of mechanical and physical properties
Recommended main literature:

1. Design Fundamentals: p. 5
2. SMART AUTONOMUS AIRCRAFT: Flight Control and Planning for UAV Yasmina Bestaoui Sebbane p.20
3. State Estimation and Control for Low-cost Unmanned Aerial Vehicles

**THE SCHEDULE**

For the diploma work preparation

| Name of sections, list of issues being developed | Submission deadlines to the scientific adviser | Notes |
|---|---|---|
| General information about control system of a UAV | 10.03.2020 | |
| Simulation | 5.04.2020 | |
| AI and ANN | 25.04.2020 | |
| Research | 27.04.2020 | |

**Signatures**

Of consultants and standard controller for the completed diploma work, indicating the relevant sections of the work (project).

| The section titles | Consultant name (academic degree, title) | Date | Signature |
|---|---|---|---|
| Main part | Candidate of Technical Sciences,Isametova M.E | 10.05.2020 | |
| Normcontrol | Candidate of Technical Sciences,Isametova M.E | 20.05.2020 | |

Scientific adviser ________________ Candidate of Technical Sciences,Isametova M.E

Signature

The task was completed by student: ___________ Ibray. S. I

Signature

Date: "20" May

## ANNOTATION

The basic part of a  title is a about  understanding about a control system. Moreover there are bunch of examples of systems that are applicable for UAV. Instead of understanding a control system, there is a programming part which gives us some notes about flight control of a drone.It is visualized by a simulation which shows the moves of a drone.Consequently, AI and ANN will be talked in this title. Opportunities and Capabilities of a AI and CNN are enormous power on a knowledgeable's hand. In the end there will be future researches that could be covered step by step.

## АННОТАЦИЯ

Основная часть названия - это понимание системы управления. Кроме того, существует множество примеров систем, которые применимы для БЛА. Кроме  понимания системы управления существует программная часть, которая дает нам некоторые заметки о управлении полетом беспилотника. Визуализируется симулятором, который показывает ходы беспилотника. Следовательно, в этой работе будут говориться об ИИ и АНН. Возможности и умение ИИ и CNN - это огромная власть на руку знающим. В конечном итоге будут проводиться будущие исследования, которые можно было бы покрывать шаг за шагом.

## АҢДАТПА

Бұл дипломдық жұмыста квадрокоптердің бақылау жүйесі туралы айтылады.

Бастапқы бөлігі «JANSER» атты бақылау жүйесі туралы айтылады.Негізгі бөлігі дронның симуляциясы жайында қарастырылады.Негізгі түйін ол программалау болып келеді және сол жайлы мәлімденеді.Жасанды Сана және Жасанды Нейро Жүйелері жаңа бағыт ретінде қарастырылады.Әрбір зерттелген ақпарат болашақ проекттің бөлігі болып табылады.Сондықтан бұл дипломдық жұмыста есептік емес ақпараттық жадылар көп қолданылған.Дронның бақылау жүйесі қиын және жабық білім көзі болып табылады.

CONTENT

# INTRODUCTION

Unmanned aerial vehicle control system is a set of equipments that are maintained by CPU(Central Processing Unit). In 21$^{st}$ century development trend of a UAV control system is growing up. The implementation of such system occurs in large varities of industrial and cultural fields. For instance, there is a drones that are used for services of delivery. It is called civilian UAV which is designed to carry out person's operation needs.

. UAV consist of several options: 1. Design;  2.Control system; 3 Material part. 4. Assembling 5. Documentation. Of these all, the control system is an important part of the big project. In a control system, the main problem is flight control and interconnection of details.By looking forward, we have to be sure that drone will fluctuate on  air.

To solve this, simulation will be need as always.Because computer model is a effective to stabilize flight control of a control. For example, in a Russian news, there was staff about one research who showed his drone in a real life. However, drone has crashed by it's own because they didn't mentioned all problems of a drone.

## 1. Control System of a UAV

UAV are managed by control systems which are consist of a several main parts.There are planning and diagram part that leds to programming. In this title JANSER is used as a experimental type of system.However, It will be as example for research because it doesn't have practical exploitation.

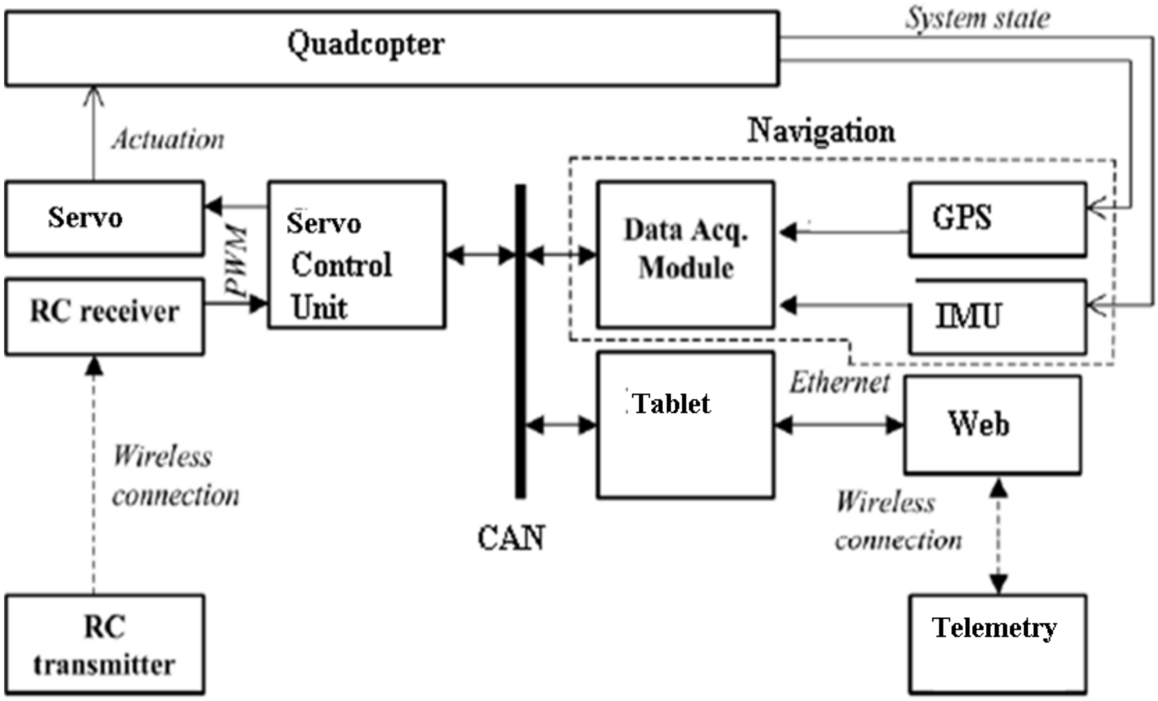


**Figure 1.B** Block diagram of a control system

JANSER is planned as a networked hierarchical conveyed framework, as is beau tiful much common nowadays. It comprises of a few essentially free functional blo cks, rctransmitte or receiver and server related to a hardware, other details in our usage. CAN is connector of hubs and manage the data transaction

JANSER ought to be a completely self-drived control framework for any kind of UAV. It is light and very compact. In spite of the fact that we utilize the JANSER framework to steer a pastime helicopter, it can be embraced for fixed-wing aircraft on the off chance that required.Most important parts of the frame are in the air. This includes an information security system (sensors and control), a monitoring system and a communication system. Tablet as it was for assembly and rendering interactive telemetry.

The CAN is the anxious framework, empowering communication. In turn, 'nodes' or 'electronic control units are like parts of the body, interconnected through the CAN transport. Data detected by one portion can be shared with another. In an car CAN transport framework,

- Variables of a control system
- Simple mode(automatic/manual)
- Variables of a Controller
- Errors

Operational data is checked synchronously with 28 Hz recurrence. The synchronization message C is sent intermittently to all centers to guarantee that all data is checked at the same time. All vehicle coefficients are sent to the Control Computer itself. When working in modified control mode, the activity assess is calculated and associated to the actuators as before long as the calculations are total. Within the manual control mode, the status variables of the controller are set so that the activity esteem is break even with to the current manual action check esteem to guarantee the move from the manual control mode to the programmed control mode. A telemetry message containing basic information is at that point promptly created and sent to the tablet.

RC Transmitter and Reciever-RC usually has a small portable device that includes certain types of control and radio transmitters. The transmitter sends frequency signals to the drone receiver. The transmitter has a power supply, usually a 9-volt battery, that provides power for control and transmission of signals. A key difference between radio control and remote control toys is that remote control toys have wires connecting the controller and the drone, and radio control is always wireless.

- Servo-There is a special sensor in the device for monitoring specific values, control units and engines. The task of the equipment is to control and maintain parameters during operation based on the signals transmitted at different points in time.
- Actuator- This component of the machine is responsible for the movement and control of mechanisms and systems, such as the opening of valves. In short, it 's a "character."
- GPS-Navigation satellites measuring time and distance; Global Positioning System)-a satellite navigation system, commonly referred to as a global positioning system
- IMU- A gyro device used to stabilize a single object or instrument and determine the angular deviation of an object. Action G is divided into direct action, power action and directive action.
- Data Acquisition mode- Data collection of various touch personnel processing.They send results to this tablet
- Telemetry- Telemetry is the collection of data or other data obtained at remote points or access points, and their automatic transmission monitoring equipment., It consists of a sensor, a transmission path, and a display, recording, or control device. Telemetry used in physical range finder equipment
- Tablet-type of a smartphone that used for controling the drone(in our case).

| | |
|---|---|
| OS | Android 8.0 |
| CPU | MT 8167 |
| RAM | 4 gb |
| Core | Cortex-A35 |
| Memory | 16 gb |
| Add.Memory | 128gb |
| Screen | 1024x600 |
| Material | Industrial |

| | |
|---|---|
| | Plastic |

**Table 1.T** - Characteristics of a tablet

| Name of a System | Weight | Geometrical parametrs | Ethernet |
|---|---|---|---|
| JANSER | 1.2kg | 335x170x55 | IEEE 802.11b |

**Table 2.C** - Characteristics of a system

Internet

- 11 mbps
- Bus utilization 0.55%
- IEEE 802.11b Ethernet
- Data size 200 bytes
- Time variation 35 times per second

## 2. Quadcopter simulation

A drone simulation with two options. It is visualized in two modes for single and multi

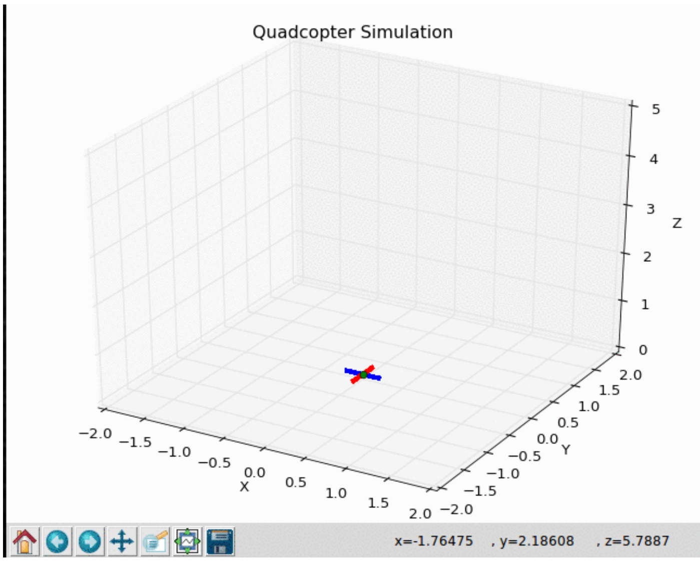


**Figure 2.D** Visualization of a simulation

### 2.1 Settings

- Libraries:
  - Numpy
  - Math
  - SciPy
  - Matplotlib
  - Matplotlib Mapping Toolkits

import

quadcopter,gui,controller

```
import signal
import sys
import argparse
# Constants
TIME_SCALING = 1.0 # Any positive number(Smaller is faster). 1.0->Real Time, 0.0->Run as fast as possible
QUAD_DYNAMICS_UPDATE = 0.002 # seconds
CONTROLLER_DYNAMICS_UPDATE = 0.005 # seconds
run = True
def Single_Point2Point():
    # Set goals to go to
    GOALS = [(1,1,2),(1,-1,4),(-1,-1,2),(-1,1,4)]
    YAWS = [0,3.14,-1.54,1.54]
    # Define the quadcopters
    QUADCOPTER={'q1':{'position':[1,0,4],'orientation':[0,0,0],'L':0.3,'r':0.1,'prop_size':[10,4.5],'weight':1.2}}
    # Controller parameters
    CONTROLLER_PARAMETERS = {'Motor_limits':[4000,9000],
                        'Tilt_limits':[-10,10],
                        'Yaw_Control_Limits':[-900,900],
                        'Z_XY_offset':500,
                        'Linear_PID':{'P':[300,300,7000],'I':[0.04,0.04,4.5],'D':[450,450,5000]},
                        'Linear_To_Angular_Scaler':[1,1,0],
                        'Yaw_Rate_Scaler':0.18,
                        'Angular_PID':{'P':[22000,22000,1500],'I':[0,0,1.2],'D':[12000,12000,0]},
                        }
    # Catch Ctrl+C to stop threads
    signal.signal(signal.SIGINT, signal_handler)
    # Make objects for quadcopter, gui and controller
    quad = quadcopter.Quadcopter(QUADCOPTER)
    gui_object = gui.GUI(quads=QUADCOPTER)
    ctrl = controller.Controller_PID_Point2Point(quad.get_state,quad.get_time,quad.set_motor_speeds,params=CONTROLLER_PARAMETERS,quad_identifier='q1')
```

```
        # Start the threads

quad.start_thread(dt=QUAD_DYNAMICS_UPDATE,time_scaling=
TIME_SCALING)

ctrl.start_thread(update_rate=CONTROLLER_DYNAMICS_UPDAT
E,time_scaling=TIME_SCALING)
        # Update the GUI while switching between destination poitions
        while(run==True):
            for goal,y in zip(GOALS,YAWS):
                ctrl.update_target(goal)
                ctrl.update_yaw_target(y)
                for i in range(300):
                    gui_object.quads['q1']['position'] =
quad.get_position('q1')
                    gui_object.quads['q1']['orientation'] =
quad.get_orientation('q1')
                    gui_object.update()
        quad.stop_thread()
        ctrl.stop_thread()
    def Multi_Point2Point():
        # Set goals to go to
        GOALS_1 = [(-1,-1,4),(1,1,2)]
        GOALS_2 = [(1,-1,2),(-1,1,4)]
        # Define the quadcopters

QUADCOPTERS={'q1':{'position':[1,0,4],'orientation':[0,0,0],'L':0.3,'
r':0.1,'prop_size':[10,4.5],'weight':1.2},
            'q2':{'position':[-
1,0,4],'orientation':[0,0,0],'L':0.15,'r':0.05,'prop_size':[6,4.5],'weight':0
.7}}
        # Controller parameters
        CONTROLLER_1_PARAMETERS =
{'Motor_limits':[4000,9000],
                    'Tilt_limits':[-10,10],
                    'Yaw_Control_Limits':[-900,900],
                    'Z_XY_offset':500,

'Linear_PID':{'P':[300,300,7000],'I':[0.04,0.04,4.5],'D':[450,450,5000]
},
                    'Linear_To_Angular_Scaler':[1,1,0],
                    'Yaw_Rate_Scaler':0.18,

'Angular_PID':{'P':[22000,22000,1500],'I':[0,0,1.2],'D':[12000,12000,
```

```
0]},
                         }
        CONTROLLER_2_PARAMETERS =
{'Motor_limits':[4000,9000],
                         'Tilt_limits':[-10,10],
                         'Yaw_Control_Limits':[-900,900],
                         'Z_XY_offset':500,

'Linear_PID':{'P':[300,300,7000],'I':[0.04,0.04,4.5],'D':[450,450,5000]
},
                         'Linear_To_Angular_Scaler':[1,1,0],
                         'Yaw_Rate_Scaler':0.18,

'Angular_PID':{'P':[22000,22000,1500],'I':[0,0,1.2],'D':[12000,12000,
0]},
                         }
        # Catch Ctrl+C to stop threads
        signal.signal(signal.SIGINT, signal_handler)
        # Make objects for quadcopter, gui and controllers
        gui_object = gui.GUI(quads=QUADCOPTERS)
        quad = quadcopter.Quadcopter(quads=QUADCOPTERS)
        ctrl1 =
controller.Controller_PID_Point2Point(quad.get_state,quad.get_time,
quad.set_motor_speeds,params=CONTROLLER_1_PARAMETERS,
quad_identifier='q1')
        ctrl2 =
controller.Controller_PID_Point2Point(quad.get_state,quad.get_time,
quad.set_motor_speeds,params=CONTROLLER_2_PARAMETERS,
quad_identifier='q2')
        # Start the threads

quad.start_thread(dt=QUAD_DYNAMICS_UPDATE,time_scaling=
TIME_SCALING)

ctrl1.start_thread(update_rate=CONTROLLER_DYNAMICS_UPDA
TE,time_scaling=TIME_SCALING)

ctrl2.start_thread(update_rate=CONTROLLER_DYNAMICS_UPDA
TE,time_scaling=TIME_SCALING)
        # Update the GUI while switching between destination poitions
        while(run==True):
            for goal1,goal2 in zip(GOALS_1,GOALS_2):
                ctrl1.update_target(goal1)
                ctrl2.update_target(goal2)
```

```
            for i in range(150):
                for key in QUADCOPTERS:
                    gui_object.quads[key]['position'] =
quad.get_position(key)
                    gui_object.quads[key]['orientation'] =
quad.get_orientation(key)
                gui_object.update()
        quad.stop_thread()
        ctrl1.stop_thread()
        ctrl2.stop_thread()
    def Single_Velocity():
        # Set goals to go to
        GOALS = [(0.5,0,2),(0,0.5,2),(-0.5,0,2),(0,-0.5,2)]
        # Define the quadcopters

QUADCOPTER={'q1':{'position':[0,0,0],'orientation':[0,0,0],'L':0.3,'r'
:0.1,'prop_size':[10,4.5],'weight':1.2}}
        # Controller parameters
        CONTROLLER_PARAMETERS =
{'Motor_limits':[4000,9000],
                    'Tilt_limits':[-10,10],
                    'Yaw_Control_Limits':[-900,900],
                    'Z_XY_offset':500,

'Linear_PID':{'P':[2000,2000,7000],'I':[0.25,0.25,4.5],'D':[50,50,5000]
},
                    'Linear_To_Angular_Scaler':[1,1,0],
                    'Yaw_Rate_Scaler':0.18,

'Angular_PID':{'P':[22000,22000,1500],'I':[0,0,1.2],'D':[12000,12000,
0]},
                    }
        # Catch Ctrl+C to stop threads
        signal.signal(signal.SIGINT, signal_handler)
        # Make objects for quadcopter, gui and controller
        quad = quadcopter.Quadcopter(QUADCOPTER)
        gui_object = gui.GUI(quads=QUADCOPTER)
        ctrl =
controller.Controller_PID_Velocity(quad.get_state,quad.get_time,qua
d.set_motor_speeds,params=CONTROLLER_PARAMETERS,quad_
identifier='q1')
        # Start the threads

quad.start_thread(dt=QUAD_DYNAMICS_UPDATE,time_scaling=
```

```
TIME_SCALING)

ctrl.start_thread(update_rate=CONTROLLER_DYNAMICS_UPDAT
E,time_scaling=TIME_SCALING)
        # Update the GUI while switching between destination poitions
        while(run==True):
            for goal in GOALS:
                ctrl.update_target(goal)
                for i in range(150):
                    gui_object.quads['q1']['position'] =
quad.get_position('q1')
                    gui_object.quads['q1']['orientation'] =
quad.get_orientation('q1')
                    gui_object.update()
        quad.stop_thread()
        ctrl.stop_thread()
    def parse_args():
        parser = argparse.ArgumentParser(description="Quadcopter
Simulator")
        parser.add_argument("--sim", help='single_p2p, multi_p2p or
single_velocity', default='single_p2p')
        parser.add_argument("--time_scale", type=float, default=-1.0,
help='Time scaling factor. 0.0:fastest,1.0:realtime,>1:slow, ex: --
time_scale 0.1')
        parser.add_argument("--quad_update_time", type=float,
default=0.0, help='delta time for quadcopter dynamics
update(seconds), ex: --quad_update_time 0.002')
        parser.add_argument("--controller_update_time", type=float,
default=0.0, help='delta time for controller update(seconds), ex: --
controller_update_time 0.005')
        return parser.parse_args()
    def signal_handler(signal, frame):
        global run
        run = False
        print('Stopping')
        sys.exit(0)
    if __name__ == "__main__":
        args = parse_args()
        if args.time_scale>=0: TIME_SCALING = args.time_scale
        if args.quad_update_time>0: QUAD_DYNAMICS_UPDATE
= args.quad_update_time
        if args.controller_update_time>0:
CONTROLLER_DYNAMICS_UPDATE =
args.controller_update_time
```

```
if args.sim == 'single_p2p':
    Single_Point2Point()
elif args.sim == 'multi_p2p':
    Multi_Point2Point()
elif args.sim == 'single_velocity':
    Single_Velocity()
```

Time: Used to determine the speed of the test system. Compare the score of 1.0 with the real-time vacation. Smaller numbers will make reconstruction faster and more harmful. The evaluation will start rebuilding on the current device as soon as possible. update: update the incremental time of the quadcopter flow.
Controller update: The incremental time for the controller to overhaul the engine .UAV: the parameters that characterize the drone: initial position and direction, arm length, center span, propeller rating and weight.Parameters Yaw_Control_Limits, gas balance, direct PID, direct and precise scaler, Yaw_Rate_Scaler and precise PID.The PID controller uses the correction flag u (t) based on the input flag y (t) so that the rectified flag is as close to the set value as possible.

**2.2 PID understandings**

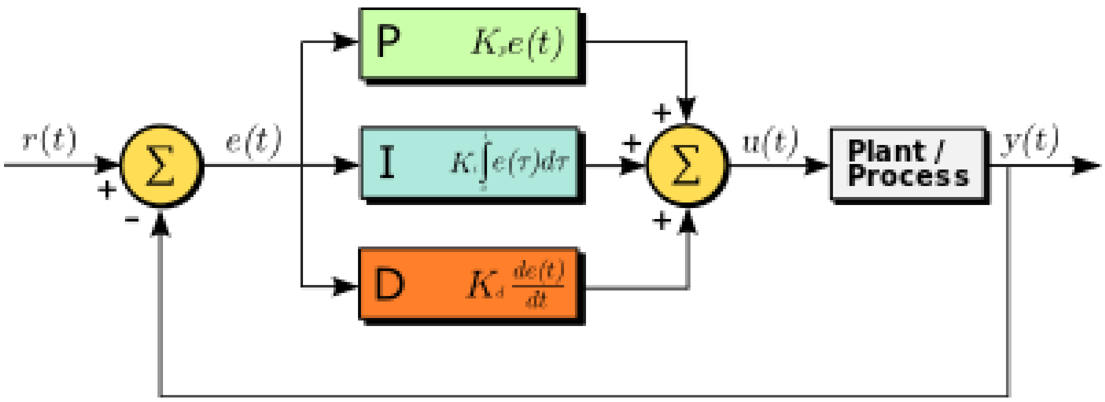


**Figure 3.P** PID scheme

PID control may be a awesome device to have in your toolbelt since it’s the establishment of a bunch of cool applications where negligible variety of the framework is basic. For illustration, flight controllers, hatcheries, suspending ping-pong balls, journey control, patching irons and much more!

To deliver you an thought of the fundamental standards of PID, let’s compare On-Off control and PID. The foremost fundamental and direct strategy for controlling a framework is the On-Off strategy. Most HVAC frameworks, fridges utilize this strategy. For example in a cooler, it cools the inside until the specified temperature is come to, and after that turn off the cooler, until it comes to a set sum over the required temperature.

PID control employments a diverse approach and accomplishes distant better;a much better;a higher;a stronger;an improved">a higher result. Let’s go back to our ice chest illustration, rather than turning the cooling unit completely on and completely off, a PID controller will alter how difficult the cooling unit is working to that the temperature remains as near as conceivable to the specified esteem

PID control is valuable in any application where it’s basic that there’s exceptionally small variety within the variable that’s being PID controlled. For illustration: a flight controller for quadcopters and planes, an hatchery, a aging tank, suspending ping-pong ball, car voyage control and so on and so forward!

A PID controller is parametrized by three constants: the proportional factor Kp, the integral factor Ki, and the derivative factor Kd. These factors encode three behaviors:

- The magnitude of the response of the controller to an error is set by the proportional factor Kp;
- The importance attributed to recent past error values is set by the integral factor Ki;
- The important of the estimation of future trend is set by the derivative factor Kd.

Mathematical Understanding:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

$$e(t) = y(t) - r(t)$$

Figure 4.F  Formula for finding u(t) signal

**3.AI Implementation**

AI is a wide ranged simulation of a human intelligence processed by a machine. Technically, it uses computers components(CPU,GPU,RAM) to train a sample. After training  data engineer can get a data which he/she will use to create a architecture for  self-acting machine. For instance, camera for recognition a human among  objects.

There are two type learning. They are Supervised and Unsupervised learning

In Supervised learning, you prepare the machine utilizing information which is well "labeled." It implies a few information is as of now labeled with the right reply. It can be compared to learning which takes put within the nearness of a boss or a teacher. A Supervised learning calculation learns from labeled preparing information, makes a difference you to anticipate results for unexpected information. Effectively building, scaling, and conveying precise administered machine learning Information science demonstrate takes time and specialized skill from a group of profoundly skilled data researchers. Additionally, Information researcher must revamp models to create beyond any doubt the bits of knowledge given remains genuine until its information changes.

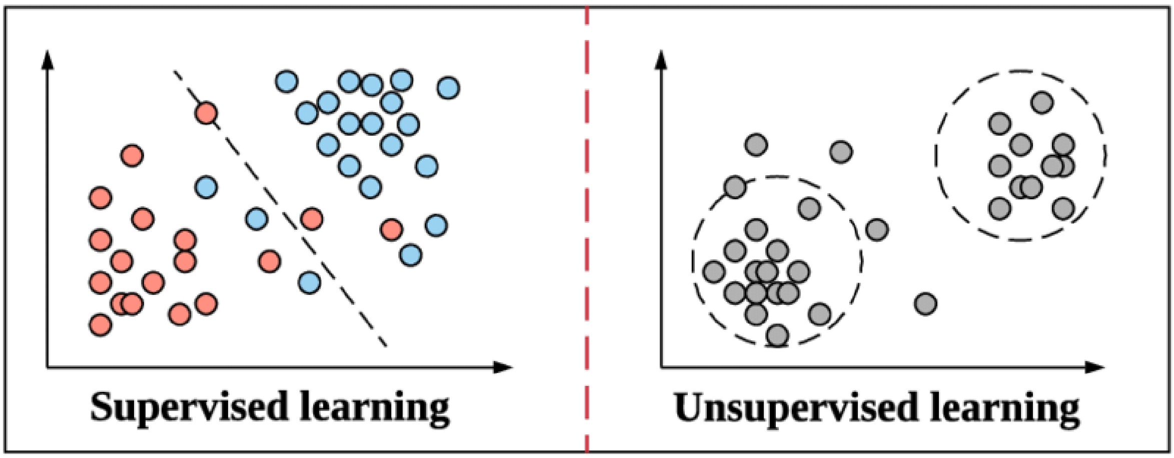


**Figure 5.E** Example of a Supervised and Unsupervised learning

Unsupervised learning could be a machine learning technique, where you are doing not ought to administer the show. Instep, you wish to permit the demonstrate to work on its possess to find data. It primarily bargains with the unlabelled data. Unsupervised learning calculations permit you to perform more complex preparing errands compared to directed learning. In spite of the fact that, unsupervised learning can be more erratic compared with other characteristic learning profound learning and support learning strategies.

For illustration, you need to prepare a machine to assist you predict how long it'll take you to drive domestic from your work environment. Here, you begin by making a set of labeled information. This information incorporates

- Weather conditions
- Time of the day
- Holidays

All these subtle elements are your inputs. The yield is the sum of time it took to drive back domestic on that particular day.

### 3.1 ANN basics

AI is based on a neural networks that are built for creating a architecture of any system.Artificial neural network architecture is motivated by the characteristic of organic neurons. Like synaptic connection in a ordinary neuron, each neuron in neural network is a function of weight. The processor has two parts, to begin with part adds the weighted inputs and moment portion could be a channel which is called dynamic work of that neuron. The input in each neuron is increased by weight and is passed on the adjacent neuron through a enactment work. Proposed artificial neural network has 3 inputs and 3 yields as appeared in Fig. 2. The inputs are reference flag, mistake and plant yield. Where outputs are Kp, Ki and Kd. As the alter in input parameters

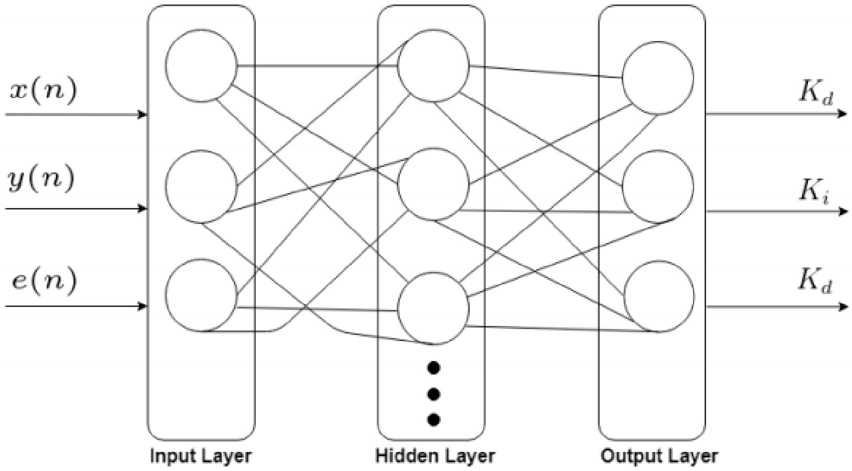


**Figure 6.N** ANN

Input Layer: In this layer, calculations are not performed, they basically transmit data to the following layer (covered up layer in most cases). The hub square is additionally called a layer.

Hidden Layer: Among these layers, for intermediate processing or calculation, these layers are calculated, and then the weight ratio (signal or information) is transferred from the input layer to the next layer (another hidden layer or output layer)Output Layer: Here we finally use the activation function to match the desired output format (e.g. Rox classification).There is also Convolutional Neural Network which is also applicable for usage. It is very similar to conventional neural networks, they are studied and shifted by the weight of neurons. The mode of connection of the device in svertochny neural network (CNN, either Confernet, or constant, or constant space) is inspired by visual bark of the organization, and the device reacts to irritants in the limited spatial area called by the receptive field. The acceptance field overlaps in terms of coverage and in the entire field of view. The reaction of the device may approach mathematical convolution operations. They change the use of multi-layered perseptrons with minimal pretreatment. It is widely used for image and video recognition, recommendation systems and natural language processing. Cells require a lot of data to train.

**4.Modeling and control algorithm**

In our case we are going to see math model of a quadcopter. By Matlab program we process numerical integration by Runge-Kutte Method.As a sample there is quadcopter with parameters:

M = 2 кг

mi = 0,2 кг

l = 0,5 м

p = 5

r = 0,07 м

c = 0,012 м

ρ = 1,4 кг/м3

Vi = 2.6 м/с
cd = 60.

As one of the basic trajectories, consider vertical take-off, hovering and landing, in this case the angular speeds of all four screws coincide and equal to ω. We choose a control so that z (t) and ω (t) were smooth continuous functions. For their implementation, we will divide management in 3 stages:

1) for 0 ≤ t <6, ω (t) changes according to a quadratic law, climb;

2) for 6 ≤ t <10 ω=gm/4k = 37,73 rad/s hovering at a height of h = 50 m

3) for 10 ≤ t <15, ω (t) changes in a similar quadratic law, landing occurs.Note that on the interval 0 ≤ t <5, the coordinate z (t) was a function fourth degree of t

$$
\begin{aligned}
y_{n+1} &= y_n + \frac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right),\\
t_{n+1} &= t_n + h\\
k_1 &= f(t_n, y_n),\\
k_2 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),\\
k_3 &= f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right),\\
k_4 &= f\left(t_n + h, y_n + hk_3\right).
\end{aligned}
$$

**Figure R.7** RK4

Forces and Moment acting on a quadcopter

$$
\left.
\begin{aligned}
F_x &= \frac{1}{2}\rho V^2 S C_D\\
F_y &= \frac{1}{2}\rho V^2 S C_L\\
F_z &= \frac{1}{2}\rho V^2 S C_C
\end{aligned}
\right\}
$$

$$
\left.
\begin{aligned}
M_x &= \frac{1}{2}\rho V^2 S l C_l\\
M_y &= \frac{1}{2}\rho V^2 S l C_n\\
M_z &= \frac{1}{2}\rho V^2 S b_A C_m
\end{aligned}
\right\}
$$

**Figure F.8** Force and Moment

Where , CL CD CC indicate the lift,drag and lateral force coefficients

Cm,Ci,Cn relates to pitching,roll,yaw torque coefficient

S-area of a wingspan

l-length of a wingspan

ba- average length of the aerodynamic chord

p-environment module acting on a density of air
V-feedback speed of the flight

## 5.Evaluation of a quadrocopter characteristics

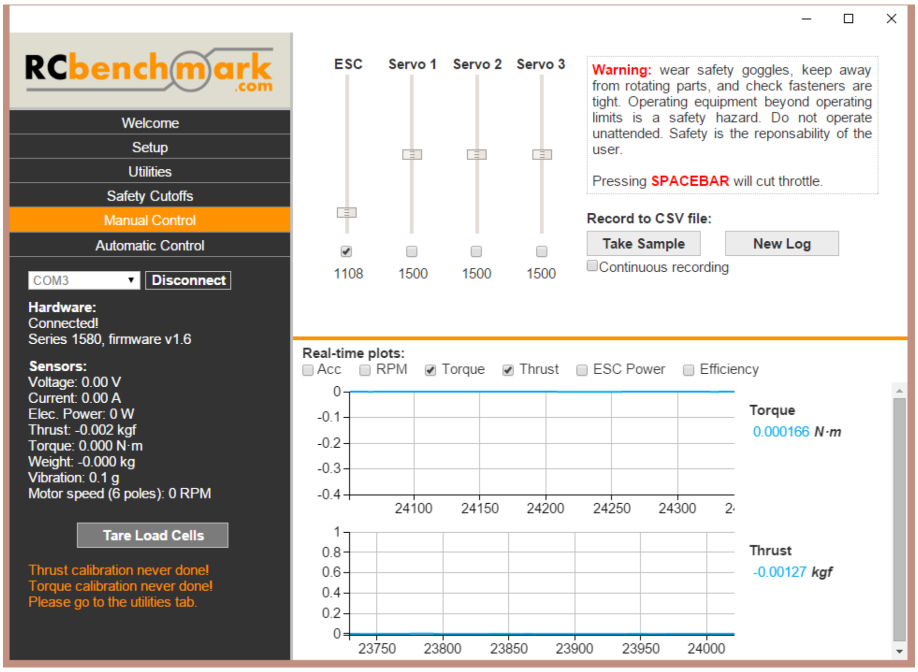


**Figure R.9** RCBenchmark

By this program we can get operational data of a quadcopter . It consist of several options by which it is possible to analyze quadcopter's current state. By choosing a Manual Control, you can gather information like:

- Torque
- Thrust
- Motor rotation speed
- Motor efficiency
- Electricity consumption

To get a data from quadcopter:

- Plug USB cabel between the quadcopter and computer or laptop
- Open the program "RCBenchmarк"
- Program will recognize the quadcopter
- Choose the Manual Control
- And here we are

## 6.Future work
### 6.1 Create a simulation that can be controlled by joystick

Simulation is a way of testing any kind of a architecture in a programming. And mostly, it is used as a testing for users(military,environmental). Thus, there could be a simulation for testing a quadrocopters.And one of the example is a AirSim.

AirSim could be a test system for rambles, cars and more, built on Unreal Engine (we presently moreover have an test Unity discharge). It is open-source, cross stage, and underpins hardware-in-loop with prevalent flight controllers such as PX4 for physically and outwardly practical reenactments. It is created as an Stunning plugin that can essentially be dropped into any Stunning environment. Additionally, we have an test discharge for a Unity plugin.

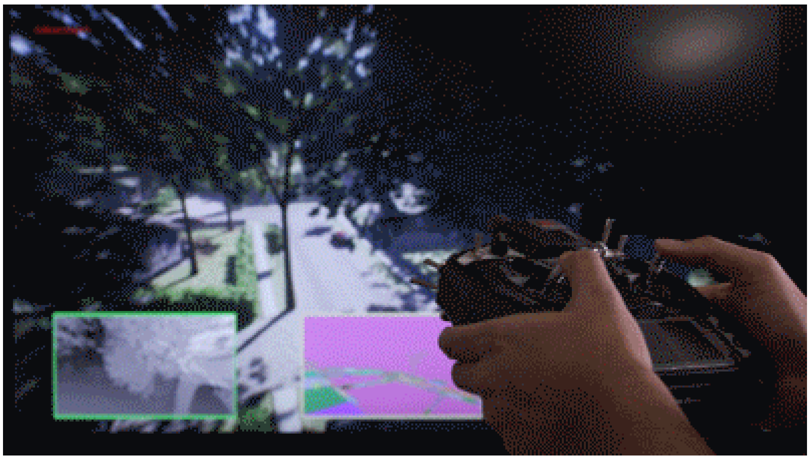

**Figure B.10** Butler

Butler discovered the API, so you 'll be contacting the vehicle as part of your rest in a programmatic way. You 'll be able to use these APIs to recover images, get state, drive a vehicle, and so on. APIs are disclosed through RPC and are available through a set of dialects, counting programming languages

**6.2 Gathering a data from simulation**

There are two ways to get ready data from Butler for profound learning. The least demanding way is basically to tap the recording button within the lower right corner. In this case, the format of the posture and pictures will start for each outline. The information registration code may be a wonderful standard and you adjust it to the substance of your heart.


**Figure S.11** Simulation

### 6.3 Computer Vision Mode

Another way to utilize Butler is the so-called "Computer Vision" mode. In this mode, you do not have vehicles or material science. You'll be able utilize the console to move around the scene, or utilize APIs to position accessible cameras in any self-assertive posture, and collect pictures such as profundity, dissimilarity, surface normals or question division.

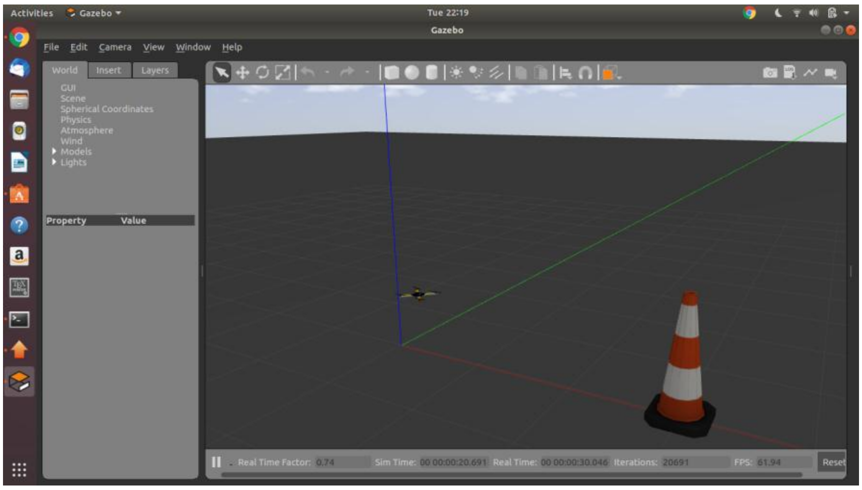


**Figure S.12** Plotted three dimension simulation.

### 6.4 Weather effects

In that simulation you can change the weather depending your requirements.By changing a weather we can get different data which will be more effective for upgrading your quadrocopters characteristics


**Figure W.12** Weather Menu

# CONCLUSION

To summarize, control system of a quadrocopter is researched to get a full image of a big project.In this title, there were provided several ideas for upgrading a future project. However, investigated data is not enough for further researches, because theoretical part refers to a practical.Therefore, title is focused on a research and works on a coding which helps to gain a data that will be needed in a further studies about a drone.By looking at a simulation we can presumable posses some opportunities on making a perfect drone simulation model.Simulation is a most effective way of developing in a our direction.So, by received information, we can decide that creating a system for drone is more implicite and complex problem for a current state. That’s why we have to expand our capabilities and opportunities by which success in a project is a inevitable.

## LIST OF A REFERENCES


1. https://dronebelow.com/2019/03/14/a-drone-controlled-with-the-help-of-a-neural-network/
2. https://rcbenchmark.gitlab.io/docs/en/
3. http://www.sunstreamsci.com/resources/CAN2.0B_Manual_v01.pdf
4. http://www.physics.mcgill.ca/~decotret/posts/pid_control.html
5. https://www.sciencedirect.com/science/article/pii/S1877050917300479
6. Design Fundamentals: p. 5
7. SMART AUTONOMUS AIRCRAFT: Flight Control and Planning for UAVYasmina Bestaoui Sebbane p.20
8. State Estimation and Control for Low-cost Unmanned Aerial Vehicles
9. https://becominghuman.ai/neural-networks-and-deep-learning-cnn-vs-rnn-7710d69feebf
10. https://pathmind.com/wiki/convolutional-network
11. https://www.datacamp.com/courses/introduction-to-r-for-data-science-edx?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631908&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=278443377101&utm_targetid=dsa-473406581475&utm_loc_interest_ms=&utm_loc_physical_ms=9063094&gclid=CjwKCAjw_LL2BRAkEiwAv2Y3SSFuA8NK9U7nOqW7pufyAU_RcbI2Zklcf6GsncA6Ja4tO81NuuJ4LRoCltcQAvD_BwE
12. https://www.datacamp.com/learn-python-with-anaconda/?utm_source=adwords_ppc&utm_campaignid=1242944157&utm_adgroupid=53325797735&utm_device=c&utm_keyword=%2Bdatacamp%20%2Bpython&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=258012599644&utm_targetid=aud-334851567295:kwd-414126611260&utm_loc_interest_ms=&utm_loc_physical_ms=9063094&gclid=CjwKCAjw_LL2BRAkEiwAv2Y3SV57SDCUtypu4X40-EJUCKE-xZngoYYMC28WeiD5Xhbe3j7XsDgYRBoCQacQAvD_BwE