

MAJOR PROJECT REPORT (DRONE WITH REMOTE CONTROLLER)

submitted in partial fulfillment of the requirements
for the award of the degree of

BACHELOR OF COMPUTER APPLICATIONS(BCA) *to*

Guru Gobind Singh Indraprastha University, Delhi

Under the Guidance of
Dr. Deepak Sonker
Associate Prof.



Submitted by
Harshit Goyal
02017002020
BCA-VI Sem

Batch: 2020-23

Department of Information, Communication & Technology



TECNIA INSTITUTE OF ADVANCED STUDIES
GRADE "A" INSTITUTE
Approved by AICTE, Ministry of HRD, Govt. of India, Affiliated to GGSIP University
Recognized Under Sec. 2(f) of UGC Act 1956
INSTITUTIONAL AREA MADHUBAN CHOWK, ROHINI, DELHI 110085
Tel: 91-11-27555121-24, E-Mail : directortias@tecnia.in, Website: www.tiaspg.tecnia.in



EGAC
Accredited
QMS & EMS Certification
ISO 9001:2015
ISO 14001:2015
CAB #118005



ISO
21001:2018
Management Systems for
Education Organizations



ISO
51001:2018
Energy Management Systems

Certificate of Completion

This is to certify that the Project work entitled “DRONE WITH REMOTE CONTROLLER” submitted by Harshit Goyal in fulfillment for the requirements of the award of Bachelor of Computer Applications Degree at Tecnia Institute of Advanced Studies, Institutional Area, Madhuban Chowk, Rohini, New Delhi is an authentic work Carried out by his/her under my super vision and guidance. To the best of my knowledge, the matter embodied in the project has not been submitted to any other University/Institute for the award of any Degree.

Dr. Deepak Sonker
Associate Prof.

A handwritten signature in cursive script, appearing to read 'Deepak', with a horizontal line underneath.

DECLARATION

I hereby declare that the work presented in this report entitled “**DRONE WITH REMOTE CONTROLLER**”, in fulfillment of the requirement for the award of the degree Bachelor of Computer Applications, Tecnica Institute of Advanced Studies, Rohini, New Delhi, is an authentic record of my own work Carried out during my degree under the guidance of Dr. **Deepak Sonker**.

The work reported in this has not been submitted by me for award of any other degree or diploma.



Harshit Goyal
02017002020
BCA 6th SEM.

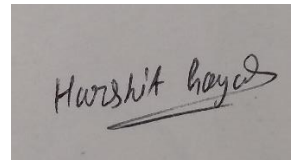
To Whom It May Concern

To Whom It May Concern

I am Harshit Goyal Enrolment No. 02017002020 from BCA-IV Semester of the Tecnia Institute of Advanced Studies, Delhi here by declares that the minor project (BCA2022-23) entitled mine. Experts (Drone with Remote) is an original work and the same has not to be submitted to any other institute for the award of any degree. The presentation of the minor project was made in 2022 and the suggestions as approved by the faculty were duly incorporated.

Date:

Signature of the student

A rectangular box containing a handwritten signature in black ink that reads "Harshit Goyal".

Certified that the major project submitted in partial fulfillment of **Information, Communication & Technology** (ICT) to be awarded by G.G.S.I.P. University, Delhi by Harshit Goyal, Enrollment No. 02017002020 has been completed under my guidance and is satisfactory

Date:

Dr. Deepak Sonker
Associate Prof.

A handwritten signature in black ink that appears to read "Deepak" with a stylized flourish.

ACKNOWLEDGEMENT

I express my sincere gratitude to Mr. Deepak Sonker and for their valuable guidance and timely suggestions during the entire duration of my minor project work, without which this work would not have been possible. I would also like to convey my deep regards to all other faculty members who have bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish this work. Finally, I would also like to thank my friends for their advice and for pointing out my mistakes.

Harshit Goyal
02017002020
BCA 6th SEM.

ABSTRACT

Remote-controlled drones have gained significant popularity in recent years, finding applications in various fields such as aerial photography, surveillance, and entertainment. However, the majority of consumer drones on the market today are equipped with cameras and often utilize voice controllers for navigation. This abstract introduces a novel concept of a remote-controlled drone that operates without a camera and voice controller.

The proposed drone design focuses on simplicity, cost-effectiveness, and enhanced user control. By omitting the camera module, the drone's overall weight and cost are significantly reduced, making it an ideal choice for beginners or individuals who prefer a streamlined drone experience. Moreover, removing the camera eliminates the need for complex image processing and storage systems, simplifying the overall drone architecture.

In lieu of a camera, the drone is equipped with advanced sensors and stabilization mechanisms. These sensors provide crucial data on altitude, orientation, and environmental conditions, ensuring stable flight performance and preventing collisions. User control is achieved through a traditional remote controller that offers intuitive joysticks and buttons for commanding the drone's movements.

To enhance user experience and expand the capabilities of the drone, additional features such as programmable flight paths, waypoint navigation, and obstacle avoidance algorithms can be incorporated. These features can be accessed through a user-friendly smartphone application that communicates wirelessly with the drone's controller.

The absence of a voice controller in this drone design allows for increased versatility, as users are not restricted by vocal commands or speech recognition limitations. Instead, the drone responds directly to user inputs, offering precise control over its movements and maneuvers.

In conclusion, this abstract presents a concept for a remote-controlled drone without a camera and voice controller, focusing on simplicity, cost-effectiveness, and enhanced user control. By prioritizing these aspects, the proposed drone design aims to provide an accessible and user-friendly experience, making drone technology more approachable to a wider audience with varying interests and requirements.

CONTENT

S No	Topic	Page No
1.	Certificate of Completion	2
2.	Declaration by the candidate	3
3.	To whom it may concern	4
4.	Acknowledgement	5
5.	Abstract	6
6.	Chapter-1: Introduction To IOT 1.1 Introduction 1.2 Objectives of IoT 1.3 Advantages of IoT 1.4 Disadvantages of IoT 1.5 Features of IoT 1.6 Future Scope of IoT 1.7 Main Components of IoT 1.8 Characteristics of IoT 1.9 Learning Outcomes of IoT 1.10 Problem Definition 1.11 About the remote controller drone 1.12 Project aims and objective 1.13 Utilization Model 1.14 Existing System	9
7.	Chapter-2: systems analysis of existing system 2.1 Hardware Requirement <ul style="list-style-type: none"> • Chassis • Motor • battery • propeller • Arduino Uno • Mpu 6050 • Jumper Cables • Receiver and Transmitter • ESC 2.2 Block Diagram 2.3 Software Requirement <ul style="list-style-type: none"> • Arduino IDE • Windows 7 and Above • Internet • Transmitter app 	19

8.	Chapter-3: business model 3.1 ER Diagram 3.2 DFD Diagram 3.2 SDLC Model	37
9.	Chapter-4: Coding	41
10.	Chapter-5: Future of The Project 5.1 Applications <ul style="list-style-type: none"> • Enhanced Flight Performance • Intelligent Autonomy • Extended Range and Connectivity • Advanced Sensor Integration • Integration with Augmented Reality (AR) • Swarm Capabilities • Customizability and Modularity • Environmental Sustainability Proposed System	69
11.	Chapter-6: Conclusion 1. Reference	71

CHAPTER-1 INTRODUCTION TO IOT

1.1 Introduction

- The Internet of Things, or IoT, is a system of interconnected computing devices, mechanical and digital machines or objects that have unique identifiers (UIDs) and the ability to transmit data over a network without requiring human connections or human-computer interaction.
- In a nutshell, the Internet of Things is the concept of connecting any device (so long as it has an on/off switch) to the Internet and to other connected devices.
- Devices and objects with built-in sensors are connected to an Internet of Things platform, which integrates data from the different devices and applies analytics to share the most valuable information with applications built to address specific needs.
- **Internet of Things (IoT)** is the networking of physical objects that contain electronics embedded within their architecture in order to communicate and sense interactions amongst each other or with respect to the external environment.
- In the upcoming years, IoT-based technology will offer advanced levels of services and practically change the way people lead their daily lives. Advancements in medicine, power, gene therapies, agriculture, smart cities, and smart homes are just a very few of the categorical examples where IoT is strongly established.
- **IoT is network of interconnected computing devices which are embedded in everyday objects, enabling them to send and receive data.**
- In the near future, IoT will become broader and more complex in terms of scope. It will change the world in terms of “Anytime, anyplace, anything in connectivity.”



Figure 1

1.2 What are the objectives of Internet of Things (IoT)

- The goal of the Internet of Things (IoT) is to make it possible to connect and integrate the physical and digital worlds. It leads the third wave of the IT industry revolution and embodies the trend of future networking.
- IoT has a wide range of uses, such as the precise real-time sensing of our surroundings and the integration of connected intelligence into commonplace products.
- Eventually, practically anything will be able to have its own verifiable identity, be aware of its surroundings, be able to interact and exchange information, directly purchase and provide services, and maybe operate entirely on its own.
- Its main goal is to collect and analyze data from objects (devices) that were previously cut off from the majority of data processing tools. Physical objects (devices) installed at the very edge of the network, such as motors, light bulbs, generators, pumps, and relays that Dorney out specified duties in order to support abusiness process are what produce this data.
- Connecting these unconnected objects (things) and transferring their data to the cloud or Internet for analysis is the goal of the Internet of Things (IoT).

1.3 Advantage of IOT

- **Minimize human effort:** As IoT devices interact and communicate with each other, they can automate the tasks helping to improve the quality of a business's services and reducing the need for human intervention.
- **Save time:** By reducing the human effort, it saves a lot of our time. Saving time is one of the primary advantages of using the IoT platform.
- **Enhanced data collection:** Information is easily accessible, even if we are far away from our actual location, and it is updated frequently in real-time. Hence these devices can access information from anywhere at any time on any device.
- **Improved security:** If we have an interconnected system, it can assist in the smarter control of homes and cities through mobile phones. It enhances security and offers personal protection.
- **Efficient resource utilization:** We can increase resource utilization and monitor natural resources by knowing the functionality and how each device works.
- **Reduced use of other electronic equipment:** Electric devices are directly connected and can communicate with a controller computer, such as a mobile phone, resulting in efficient electricity use. Hence, there will be no unnecessary use of electrical equipment.
- **Use in traffic systems:** Asset tracking, delivery, surveillance, traffic or transportation tracking, inventory control, individual order tracking, and customer management can be more cost-effective with the right tracking system using IoT technology.
- **Useful for safety concerns:** It is helpful for safety because it senses any potential danger and warns users. For example, GM OnStar is an integrated device that identifies a Drone crash or accident on the road. It immediately makes a call if an accident or crash is found.



Figure 2

1.4 Disadvantage of IOT

1. **Security issues:** IoT systems are interconnected and communicate over networks. So, the system offers little control despite any security measures, and it can lead to various kinds of network attacks.
2. **Privacy concern:** The IoT system provides critical personal data in full detail without the user's active participation.
3. **Increased unemployment:** Unskilled workers or even the skilled ones are at a high risk of losing their jobs, leading to high unemployment rates. Smart surveillance cameras, robots, smart ironing systems, smart washing machines, and other facilities are replacing the humans who would earlier do these works.
4. **The complexity of the system:** The designing, developing, maintaining, and enabling the extensive technology to IoT system is quite complicated.
5. **High chances of the entire system getting corrupted:** If there is a bug in the system, it is possible that every connected device will become corrupted.
6. **Lack of international standardizations:** As there is no international standard of compatibility for IoT, it is problematic for devices from different manufacturers to communicate with each other.
7. **High dependency on the internet:** They rely heavily on the internet and cannot function effectively without it.
8. **Reduced mental and physical activity:** Overuse of the internet and technology makes people ignorant because they rely on smart devices instead of doing physical work, causing them to become lethargic and inactive.

1.5 Features of IOT

The most important features of IoT are:

1. **Connectivity:** - Connectivity refers to creating a proper connection between all IoT things and the IoT platform, it can be a server or cloud. Once connected, IoT devices need high-speed messaging between the devices and the cloud to enable reliable, secure, two-way communication.
2. **Analyzing:** - After connecting all the relevant things, it comes to real-time analyzing the data collected and use them to build effective business intelligence.
3. **Integrating:** - IoT integrating the various models to improve the user experience as well.
4. **Artificial Intelligence:** - IoT makes things smart and enhances life through the use of data.
5. **Sensing:** -The sensor devices used in IoT technologies detect and measure any change in the environment and report on their status. IoT technology brings passive networks to active networks. Without sensors, there could not hold an effective or true IoT environment.
6. **Active Engagement:** IoT makes the connected technology, product, or services to active engagement between each other.

1.6 Future scope of IOT

- We are living in a digitalized world that is full of technological advancements. One of these technological advancements is Internet of Things (IoT). The future scope of IoT is paving its way to make the world a smarter place to live in.
- It has gained a lot of popularity in lesser time. Also, the advancements in Artificial Intelligence and Machine Learning have made the automation of IoT devices easy. Basically, AI and ML programs are combined with IoT devices to give them proper automation.
- Along with wider adoption, new technologies will help make RFID more reliable and cost-effective for a larger number of applications.
- An electronic, sensor, and software-enabled network of connected physical things is known as the Internet of Things (IoT).
- These nodes, which are connected items, communicate with one another online. By 2025, there will be 22 billion linked devices worldwide. Industries like manufacturing, agriculture, healthDronee, transportation, media/advertising, retail, water and waste management, power distribution, etc. can all benefit from the deployment of IoT technology.

- IoT is employed in many different applications, including smart solar panels, smart homes, smart parking, smart healthDronee, smart traffic lighting, and smart water levelmonitoring.
- It is necessary for the industry and other contexts to achieve high production, efficiency, a safe working environment, and minimal Dronebon emissions. Severalstartups are focusing their efforts on smart manufacturing and healthDronee.



Figure 3

1.7 Main components used in IoT

- **Low-power embedded systems:** Less battery consumption, high performance are the inverse factors that play a significant role during the design of electronic systems.
- **Sensors:** Sensors are the major part of any IoT applications. It is a physical device that measures and detect certain physical quantity and convert it into signal which can be provide as an input to processing or control unit for analysis purpose.
- **Control Units:** It is a unit of small computer on a single integrated circuit containing microprocessor or processing core, memory and programmable input/output devices/peripherals. It is responsible for major processing work of IoT devices and all logical operations are Carried out here.
- **Cloud computing:** Data collected through IoT devices is massive and this data has to be stored on a reliable storage server. This is where cloud computing comes into play. The data is processed and learned, giving more room for us to discover where things like electrical faults/errors are within the system.

- **Availability of big data:** We know that IoT relies heavily on sensors, especially in real-time. As these electronic devices spread throughout every field, their usage is going to trigger a massive flux of big data
- **Networking connection:** In order to communicate, internet connectivity is a must where each physical object is represented by an IP address. However, there are only a limited number of addresses available according to the IP naming. Due to the growing number of devices, this naming system will not be feasible anymore. Therefore, researchers are looking for another alternative naming system to represent each physical object.

1.8 Characteristics of IoT

Some characteristics of Iot are:

- Massively scalable and efficient
- IP-based addressing will no longer be suitable in the upcoming future.
- An abundance of physical objects is present that do not use IP, so IoT is made possible.
- Devices typically consume less power. When not in use, they should be automatically programmed to sleep.
- A device that is connected to another device right now may not be connected in another instant of time.
- Intermittent connectivity – IoT devices aren't always connected. In order to save bandwidth and battery consumption, devices will be powered off periodically when not in use. Otherwise, connections might turn unreliable and thus prove to be inefficient.

1.9 Learning Outcomes of IoT

- ☐ **HOW DO I REMOTELY TRACK, MONITOR, AND MANAGE MY ASSETS?**
Collect and analyze telemetry from connected sensors, devices and equipment for real-time monitoring, surveillance, management and remote control.
- ☐ **HOW DO I BUILD A LIVING LINK TO BOTH MY PRODUCT AND USERS?**
Expand your versatile business beyond traditional product boundaries with IoT-connected smart products that recognize and respond to user needs, enable new innovative experiences, and create new revenue models.
- ☐ **HOW DO I IMPROVE OPERATIONS?**
Increase operational visibility, gain insights, reduce inefficiency or waste, and control and automate processes to achieve operational benefits. Get enriched with real-time data, intelligent edge and analytics for smarter, automated and autonomous processes.

1.10 Problem Definition

We have seen the number of cities where the light is the one of the huge energy expenses for a city.

- ☐ **Disadvantages of Existing System**
 - Manual Switching off/on of Lights
 - More Energy Consumption.
 - High expense.
 - More manpower.
- ☐ **Advantages of the Proposed System**
 - Automatic Switching of lights using rfid technology
 - Maintenance Cost Reduction.
 - Wireless Communication.
 - Energy Saving.
 - Reduction of manpower.

1.11 About The Remote controlled drone

The remote-controlled drone without a camera is a cutting-edge device that offers a simplified and cost-effective approach to drone technology. Unlike traditional drones equipped with cameras, this drone design eliminates the camera module, resulting in a lighter and more affordable product. This makes it an ideal choice for beginners, recreational users, or individuals who prioritize a streamlined drone experience over aerial photography or video recording capabilities.

Instead of relying on a camera for navigation and orientation, this drone incorporates advanced sensors and stabilization mechanisms. These sensors provide crucial data on altitude, orientation, and environmental conditions, ensuring stable flight performance and preventing collisions. By leveraging these sensors, the drone can accurately respond to user commands and maintain a smooth flight path.

User control is achieved through a traditional remote controller that features intuitive joysticks and buttons.

This familiar interface allows users to command the drone's movements with precision and ease. The absence of a voice controller eliminates the need for voice recognition technology, reducing complexity and potential limitations associated with voice commands.

Furthermore, the drone can be enhanced with additional features such as programmable flight paths, waypoint navigation, and obstacle avoidance algorithms. These features can be accessed through a user-friendly smartphone application that communicates wirelessly with the drone's controller. This integration of smartphone technology expands the capabilities of the drone and offers a more interactive and customizable flight experience.

Overall, the remote-controlled drone without a camera provides a simplified and accessible approach to drone technology. By focusing on simplicity, affordability, and enhanced user control, this innovative design caters to a broader audience, making drone technology more approachable and enjoyable for various recreational and non-photography-related applications.

1.12 Project Aim and Objective

Project Aim:

The aim of the project is to design, develop, and test a remote-controlled drone without a camera, focusing on simplicity, cost-effectiveness, and enhanced user control. The project aims to provide a streamlined and accessible drone experience for users who prioritize flight performance and maneuverability over aerial photography or video recording capabilities.

Objectives:

1. Design the drone's architecture: Determine the hardware and software components required for the remote-controlled drone without a camera. Consider factors such as weight, cost, stability, and user control.
2. Develop advanced sensors and stabilization mechanisms: Design and integrate sensors that provide crucial data on altitude, orientation, and environmental conditions. Implement stabilization mechanisms to ensure stable flight performance and prevent collisions.
3. Create a user-friendly remote controller: Design a traditional remote controller with intuitive joysticks and buttons for commanding the drone's movements. Ensure seamless communication between the remote controller and the drone.
4. Implement additional features: Incorporate optional features such as programmable flight paths, waypoint navigation, and obstacle avoidance algorithms to enhance the drone's capabilities and user experience.
5. Develop a smartphone application: Create a user-friendly smartphone application that wirelessly communicates with the drone's controller. The application should provide access to additional features and offer a customizable and interactive flight experience.
6. Conduct testing and optimization: Test the drone's performance, stability, and responsiveness. Gather user feedback to identify areas for improvement and optimize the drone's design and functionality.
7. Evaluate cost-effectiveness: Assess the cost-effectiveness of the remote-controlled drone without a camera.

compared to traditional camera-equipped drones. Consider the reduction in weight and cost resulting from the absence of a camera module.

8. Document and report findings: Document the design process, development, and testing results. Prepare a comprehensive report outlining the project's objectives, methodology, and outcomes.

By achieving these objectives, the project aims to deliver a remote-controlled drone without a camera that provides a simplified, cost-effective, and user-controlled flying experience, expanding the accessibility of drone technology beyond the realm of aerial photography.

1.13 Existing system

At that time, some existing remote-controlled drones without cameras focused on providing a simplified and affordable flying experience. These drones typically had a compact and lightweight design, making them suitable for indoor use or areas where camera functionality was not necessary.

These drones often featured intuitive remote controllers with responsive joysticks and buttons, allowing users to control the drone's movements with precision. The controllers usually had a range of up to a few hundred meters, providing flexibility and freedom for flying in open spaces.

In terms of flight capabilities, these drones generally offered basic maneuverability, including ascending, descending, forward and backward movement, and lateral movement. Some models may have incorporated stabilization systems or gyroscopes to enhance flight stability and prevent crashes.

However, it is essential to research and explore the current market offerings as technology advances rapidly, and new products may have been introduced since the cutoff date. Manufacturers often release updated models with improved features and functionalities, so it's advisable to consult reliable sources and check the latest offerings to get the most up-to-date information on existing remote-controlled drones without cameras.

Chapter 2: System/Project Requirement

2.1 Hardware Requirement

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, a hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of Operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application.

2.1.1 Chassis

The chassis of a drone serves as the primary support structure, comparable to a "frame." It withstands all the stresses experienced by the drone during both stationary and moving conditions. Similar to the skeleton of a living organism, the chassis plays a crucial role in maintaining the overall structural integrity of the drone. The term "chassis" originates from the French language. Whether it's a two-wheeled or a multi-rotor drone, every drone possesses a chassis that serves as its foundational framework.



Figure 2: Chassis

2.1.2 Motor

A motor is any of a class of electrical machines that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor. Most types produce rotary motion; a linear motor directly produces force and motion in a straight line. motors were the first type widely used, since they could be powered from existing direct current

getting power distribution systems. A motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small motors are used in tools, toys, and appliances. The universal motor can operate on direct current but is a lightweight motor used for portable power tools and appliances. Larger motors are used in propulsion of electric vehicles, elevators and hoists, or in drives for steel.



Figure 3: Motor

2.1.3 Propellers

The propeller on a drone is a vital component responsible for generating the necessary thrust to keep the drone airborne. It consists of multiple wings, also known as blades, that are fixed together in a specific arrangement. These blades are designed to spin rapidly when the drone is in operation.

As the propeller blades rotate, they create a pressure difference in the surrounding air. The spinning motion generates a low-pressure pocket directly above the blades. This low-pressure area is created due to the rapid movement of the wings, which pushes the air away from the propeller.

In response to the low-pressure pocket, the air underneath the propeller blades tries to fill the void by rushing towards it. This movement of air from beneath the blades to the low-pressure area creates a flow of air, known as thrust, which propels the drone forward or upward, depending on the specific orientation of the propellers.

The number of propellers on a drone can vary, but for the purpose of this discussion, let's assume we are referring to a drone with multiple propellers. Each propeller operates independently, generating its own low-pressure area and producing thrust. The number of propellers directly influences the overall thrust produced by the drone.



Figure 4: Propellers

2.1.4 Arduino UNO

- ☐ Uno is a microcontroller board based on 8-bit ATmega328P microcontroller. Along with ATmega328P, it consists of other components such as crystal oscillator, serial communication, voltage regulator, etc. to support the microcontroller. The Arduino Uno comes with USB interface, 6 analog input pins, 14 I/O digital ports that are used to connect with external electronic circuits. Out of 14 I/O ports, 6 pins can be used for PWM output.
- ☐ It allows the designers to control and sense the external electronic devices in the real world. Since it was first debuted, the Arduino Uno has been a huge hit with electronics enthusiasts from beginner hobbyists to professional programmers. It is an open-source platform, meaning the boards and software are readily available and anyone can modify and optimize the boards for better functionality.
- ☐ The software used for Arduino devices is called IDE (Integrated Development Environment) which is free to use and requires some basic skills to learn it. It can be programmed using C and C++ language.



Figure 5: Arduino Uno

2.1.5 ESC

An electronic speed controller (ESC) is a crucial component in remote-controlled drones, including those without cameras. The ESC is responsible for controlling the speed and direction of the drone's motors based on the pilot's input through the remote controller.

The primary function of an ESC is to convert the direct current (DC) power from the drone's battery into alternating current (AC) power that drives the brushless motors. It regulates the voltage and current supplied to the motors, allowing precise control over the drone's speed.

Here are some key points about electronic speed controllers for remote-controlled drones without cameras:

1. **Brushless Motors:** Most modern drones, including those without cameras, utilize brushless motors due to their efficiency and reliability. The ESC is specifically designed to work with brushless motors, providing smooth and responsive motor control.
2. **Pulse Width Modulation (PWM):** ESCs utilize PWM signals to regulate the motor speed. The remote controller sends PWM signals to the ESC, which interprets them to adjust the motor's rotational speed. By varying the width and timing of the PWM signals, the ESC controls the motor's speed and direction.
3. **Calibration:** ESCs often require calibration before initial use or after firmware updates. This process involves establishing the neutral throttle position and endpoints to ensure proper motor response to the remote controller's input. Calibration procedures may vary depending on the ESC model, and manufacturers provide specific instructions.
4. **Communication Protocols:** Some advanced ESCs support communication protocols such as Oneshot, Multishot, or DShot, which provide faster and more accurate signal transmission between the flight controller and ESC. These protocols enhance the overall performance and

responsiveness of the drone.

5. **Overcurrent and Thermal Protection:** ESCs often include built-in safety features to protect against overcurrent and overheating. These protection mechanisms help prevent damage to the ESC and motors in case of excessive current draw or high operating temperatures.
6. **ESC Firmware Updates:** Manufacturers occasionally release firmware updates for ESCs to improve performance, add new features, or address potential issues. Firmware updates can be applied by connecting the ESC to a computer or using specialized tools provided by the manufacturer.

It's important to consider the compatibility between the ESC, motors, and flight controller when selecting or upgrading components for a remote-controlled drone without a camera. Manufacturers usually provide guidelines and recommendations for compatible components to ensure optimal performance and compatibility.



2.1.6 MPU 6050

The MPU-6050 is a commonly used Inertial Measurement Unit (IMU) module in remote-controlled drones, including those without cameras. It combines a 3-axis accelerometer and a 3-axis gyroscope, providing essential motion sensing capabilities for flight control and stability.

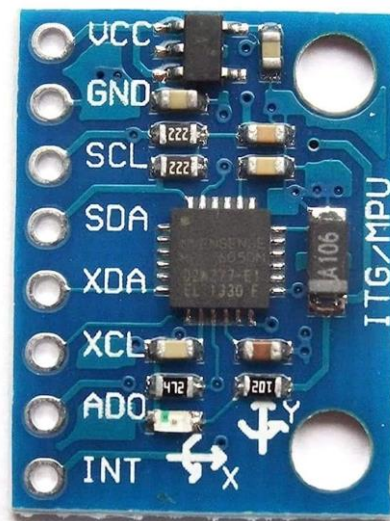
Here are some key points about the MPU-6050 IMU module for remote-controlled drones without cameras:

1. **Motion Sensing:** The MPU-6050 integrates a 3-axis accelerometer and a 3-axis gyroscope, allowing the drone's flight controller to measure both linear acceleration and angular velocity. This information is vital for maintaining stability, calculating orientation, and adjusting motor speeds.
2. **Orientation Estimation:** By fusing the accelerometer and gyroscope data, the MPU-6050 helps estimate the drone's orientation or attitude in real-time. This data is utilized by the flight controller

to stabilize the drone and make appropriate adjustments to maintain a desired flight path.

3. **Digital Motion Processor (DMP):** The MPU-6050 includes a Digital Motion Processor (DMP) that offloads complex sensor fusion calculations from the main flight controller. The DMP can provide quaternion data, which represents the drone's orientation in a 3D space, simplifying the integration of motion sensing into the flight control system.
4. **I2C Interface:** The MPU-6050 communicates with the flight controller using the I2C (Inter-Integrated Circuit) interface. This serial communication protocol enables efficient and reliable data transmission between the MPU-6050 and the flight controller.
5. **Calibration:** Similar to other sensors, the MPU-6050 may require calibration to ensure accurate readings. Calibration procedures typically involve placing the drone in specific orientations and allowing the sensor to measure and compensate for any biases or offsets in its readings.
6. **Data Filtering:** To improve the accuracy and stability of the motion data, the MPU-6050 often incorporates various filtering algorithms. These algorithms help reduce noise, filter out high-frequency vibrations, and improve the overall performance of the motion sensing system.

The MPU-6050 is widely used in the drone industry due to its cost-effectiveness, compact size, and reliable performance. It plays a critical role in providing motion sensing capabilities, enabling precise control and stabilization of remote-controlled drones without ca



2.1.7 Receiver and transmitter

In a remote-controlled drone without a camera, the receiver and transmitter are essential components for communication between the drone and the remote controller. They enable the pilot to send control signals to the drone, allowing for precise control over its movements. Here are some key points about the receiver and transmitter for a remote-controlled drone without a camera:

Transmitter (Remote Controller):

1. **Function:** The transmitter, also known as the remote controller, is the handheld device operated by the pilot. It consists of joysticks, switches, buttons, and other input mechanisms that allow the pilot to send commands to the drone.
2. **Radio Frequency (RF):** The transmitter uses RF technology to wirelessly transmit control signals to the receiver on the drone. Common RF frequencies used in remote-controlled drones are 2.4 GHz and 5.8 GHz.
3. **Control Channels:** The transmitter provides multiple control channels, which correspond to different drone functions such as throttle, yaw, pitch, and roll. The number of control channels determines the degree of control and the complexity of maneuvers possible with the drone.
4. **Mode Switches:** Transmitters often feature mode switches that allow the pilot to switch between different flight modes or control profiles. These modes can include manual control, stability-assisted modes, or even advanced autonomous features depending on the capabilities of the drone and the transmitter.

Receiver:

1. **Function:** The receiver is the component installed on the drone that receives the control signals from the transmitter. It decodes the signals and relays them to the drone's flight controller, which then processes the commands and adjusts the motor speeds accordingly.
2. **Binding:** Before operation, the receiver and transmitter need to be bound together. Binding is a process that establishes a secure connection between the two devices, ensuring that the receiver only responds to signals from the specific transmitter it is bound to.
3. **PWM or Digital Protocols:** Receivers can utilize different communication protocols, such as PWM (Pulse Width Modulation) or digital protocols like PPM (Pulse Position Modulation) or SBUS. These protocols determine how control signals are transmitted from the receiver to the flight controller.
4. **Range:** The range of the receiver determines the maximum distance between the transmitter and the drone for reliable communication. Higher-end transmitters often provide longer range capabilities, but it's important to consider the local regulations and fly within the legal range limits.
5. **Fail-Safe Functionality:** Receivers may incorporate fail-safe mechanisms that automatically trigger predefined actions in case of signal loss or interference. These actions can include returning the drone to a safe altitude or activating a pre-programmed landing sequence.

It is important to ensure compatibility between the transmitter and receiver to ensure proper communication and control. Manufacturers often provide specific receiver recommendations for their transmitters, and it is advisable to use compatible components for optimal performance and reliability.



2.1.8 Jumper Cables

- ☐ Jumper cables, booster cables or jumper leads (all three terms describe the same product), let you get a jump start of your dead Drone battery. The cables connect the battery of a running Drone to the battery of your dead (won't-start) Drone. Even if you have auto club or roadside service, you can be on your way in five minutes, quicker than waiting for a service vehicle
- ☐ Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with bread boards and other prototyping tools in order to make it easy to change a circuit as needed. Fairly simple. In fact, it doesn't get much more basic than jumper wires.
- ☐ **Types of Jumper Wires**
Jumper wires come in three versions:

- Male-to-male jumper
 - Male-to-female jumper
 - Female-to-female jumper
 - And two types of head shapes: square head and round head.
- The difference between each is in the endpoint of the wire. Male ends have a pin protruding and can plug into things, while female ends do not but are also used for plugging.
- Moreover, a male connector is referred to as a plug and has a solid pin for center conduction. Meanwhile, a female connector is referred to as a jack and has a center conductor with a hole in it to accept the male pin.

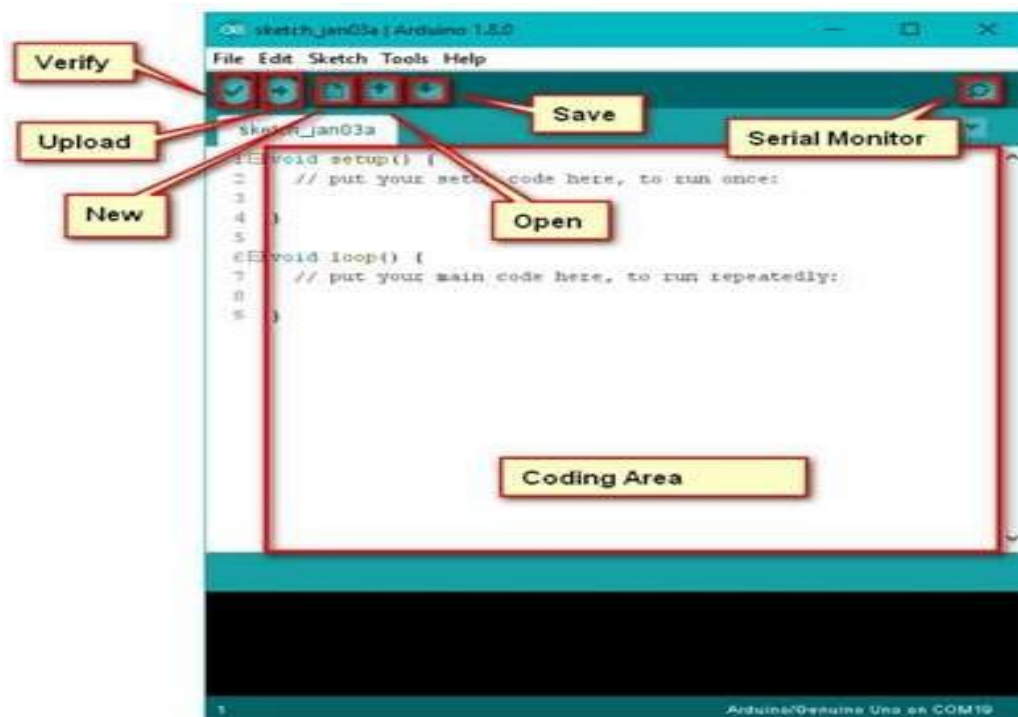


Figure 8

2.2 Software Requirement

2.2.1.1 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino and Genuino hardware to upload programs and communicate with them. Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom righthand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create,



open, and save sketches, and open the serial monitor.

Figure 11: Arduino IDE main window

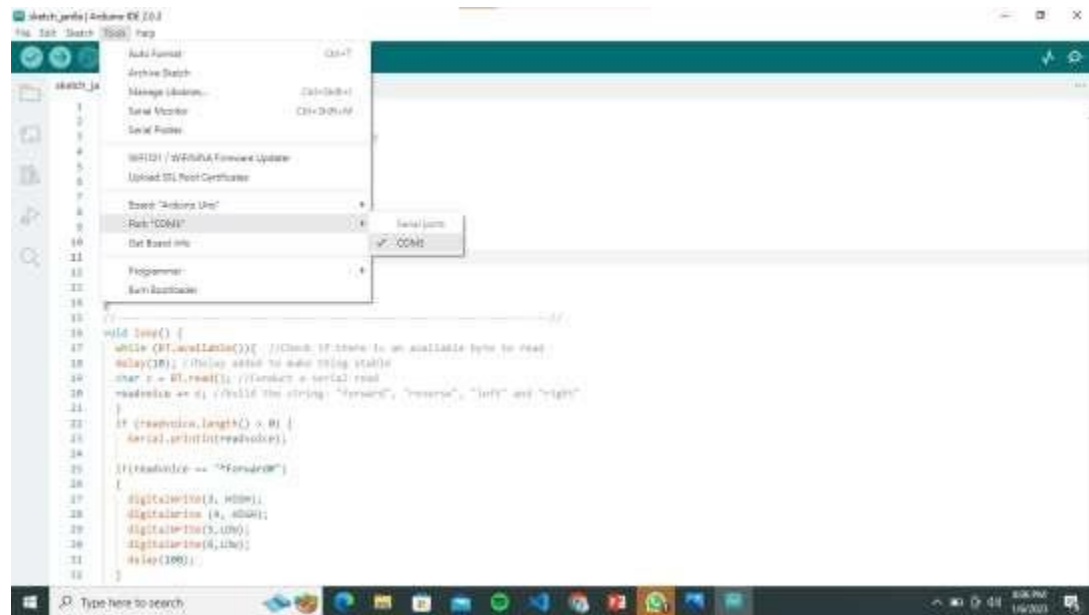


Figure 12: Port Selection

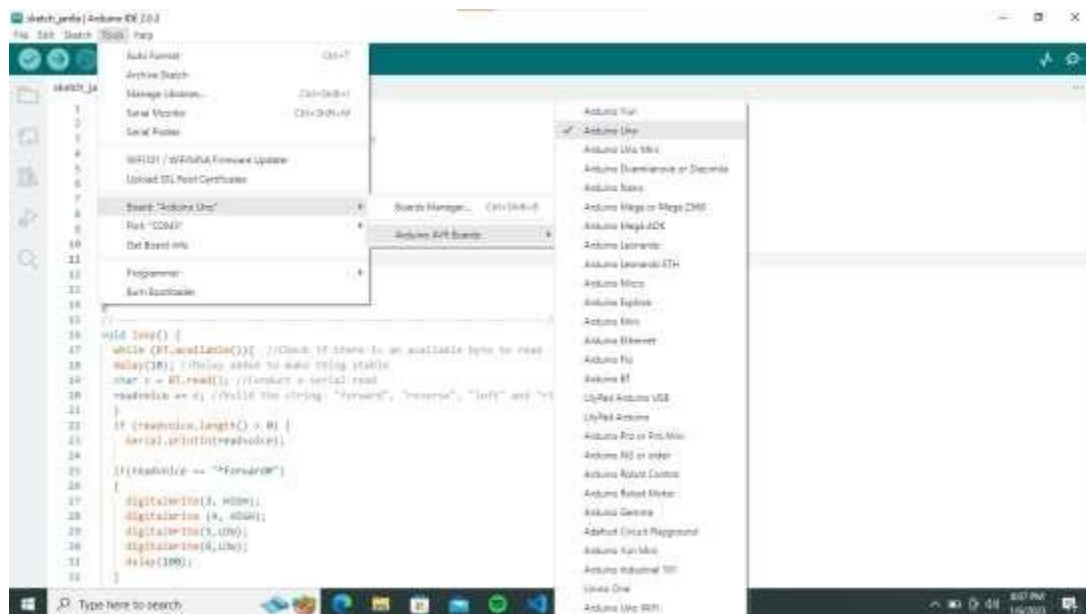


Figure 13: Board Selection

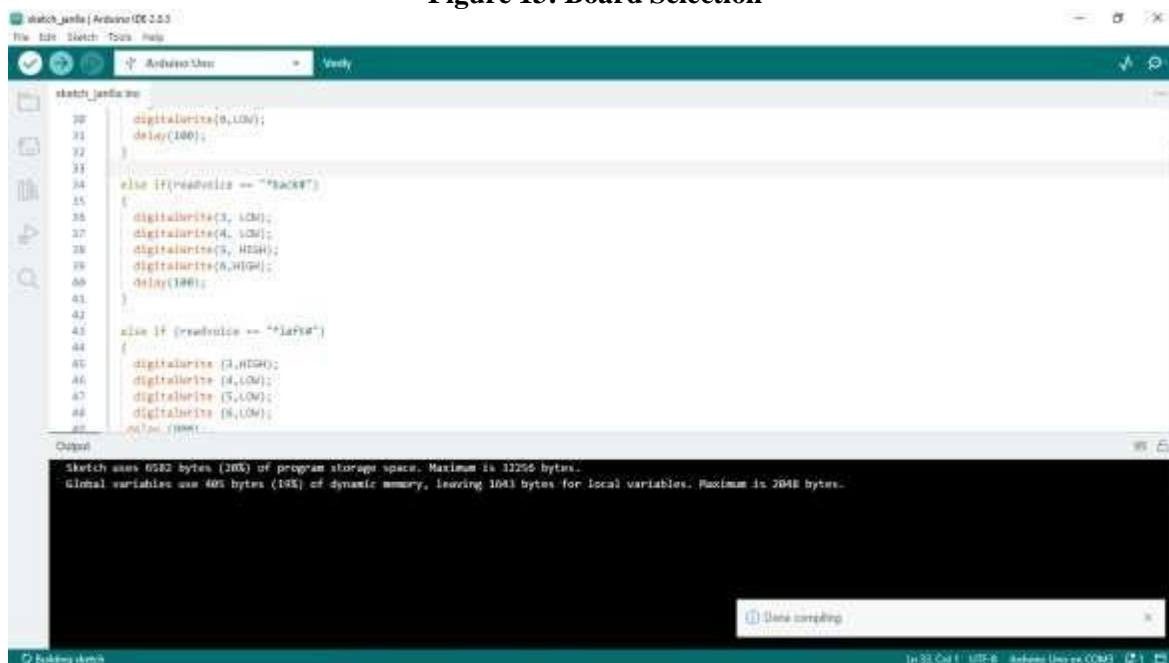


Figure 14: Compiling Project

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and linux. The Arduino project provides the Arduino integrated development environment (IDE), which is a cross-platform application written in Java. It originated from the IDE for the Processing programming language project and the Wiring project. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features

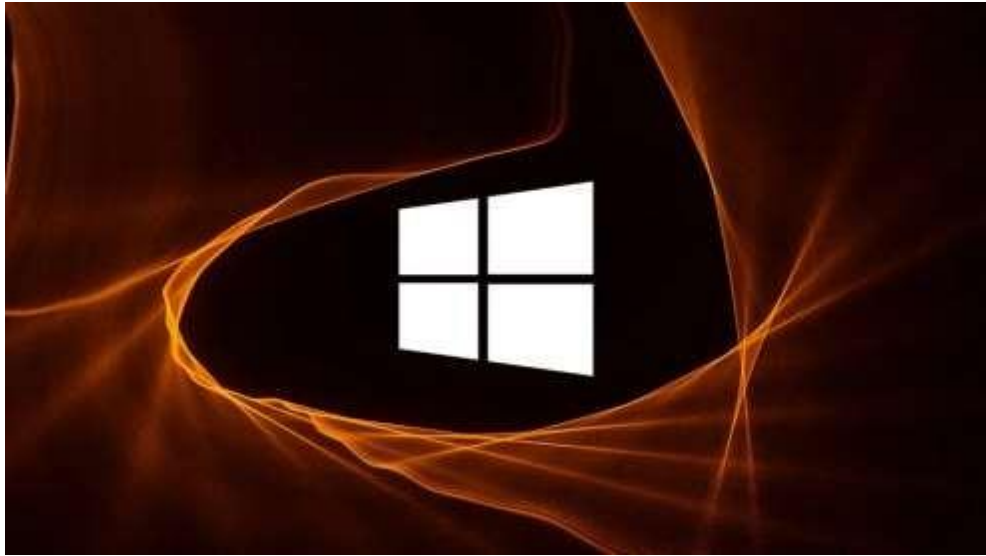
such as syntax highlighting, brace matching, and automatic indentation, and provides simple one-click mechanism for compiling and loading programs to an Arduino board. A

program written with the IDE for Arduino is called a “sketch” . The main window of Arduino IDE is shown in Fig.. The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store your programs (or sketches). The sketches in your sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time you run the Arduino software, it will automatically create a directory for your sketchbook.

You can view or change the location of the sketchbook location from with the Preferences dialog. Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from the Sketch > Import Library menu. This will insert one or more #include statements at the top of the sketch and compile the library with your sketch. Because libraries are uploaded to the board with your sketch, they increase the amount of space it takes up. There is a list of libraries in the reference. Some libraries are included with the Arduino software. Others can be downloaded from a variety of sources or through the Library Manager. Starting with version 1.0.5 of the IDE, you do can import a library from a zip file and use it in an open sketch.

2.2.2 WINDOWS 7 and Above

- Windows 7 is a major release of the Windows NT operating system developed by Microsoft. It was released to manufacturing on July 22, 2009, and became generally available on October 22, 2009.
- It is the successor to Windows Vista, released nearly three years earlier. It remained an operating system for use on personal computers, including home and business desktops, laptops, tablet PCs and media center PCs, and itself was replaced in November 2012 by Windows 8, the name spanning more than three years of the product.
- Windows 7 was intended to be an incremental upgrade to Microsoft Windows, addressing Windows Vista's poor critical reception while maintaining hardware and software compatibility. Windows 7 continued improvements on the Windows Aero user interface with the addition of a redesigned taskbar that allows pinned applications, and new window management features.
- Windows 7 is the final version of Windows that supports processors without SSE2 or NX (although an update released in 2018 dropped support for non-SSE2 processors). Its successor, Windows 8, requires a processor with SSE2 and NX in any supported architect



ure.

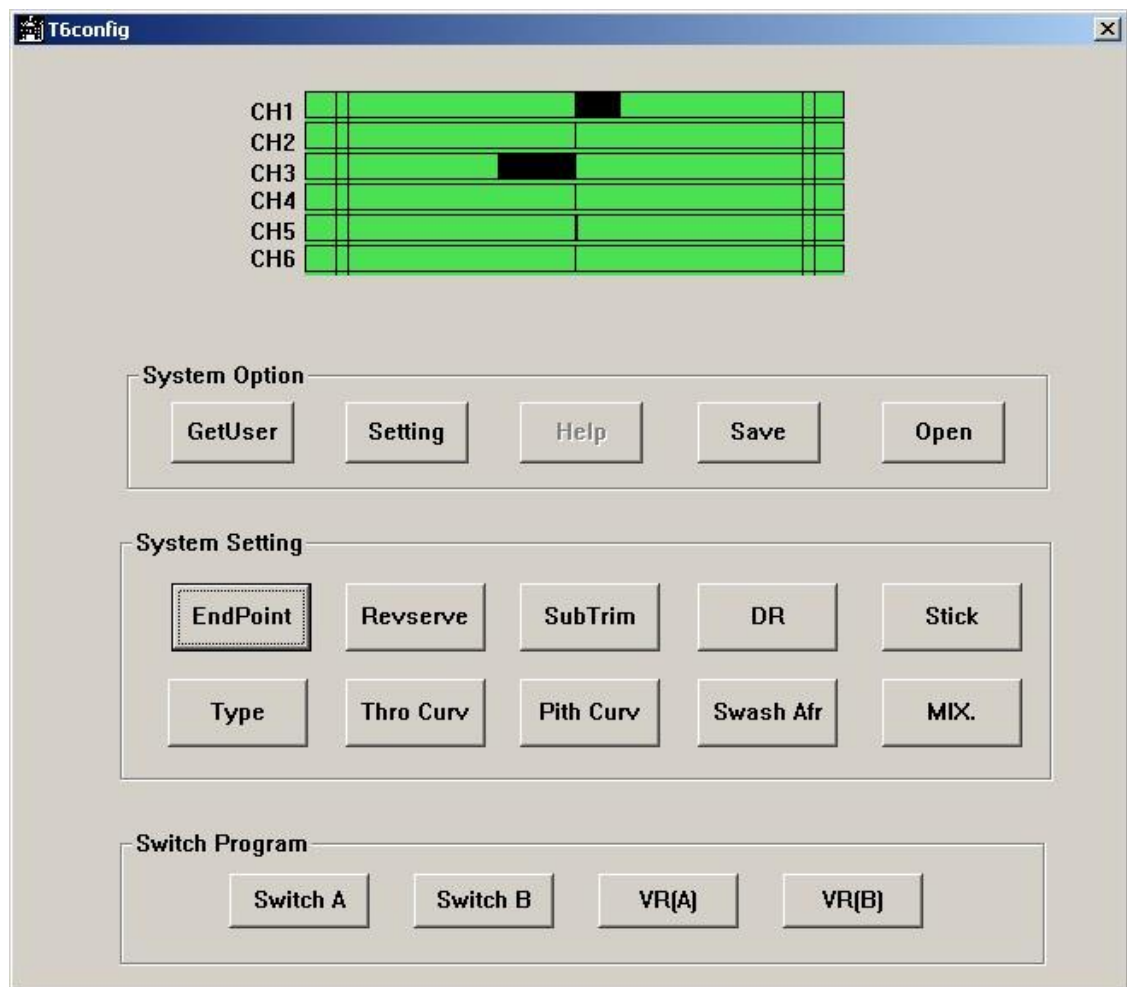
2.2.3 Transmitter App

The "T6" you mentioned could refer to the Flysky FS-T6 transmitter, a popular model among remote control enthusiasts. Here's a general overview of the configuration process for the Flysky FS-T6:

1. **Power On:** Ensure that the transmitter and receiver are both powered off. Insert batteries into the transmitter, and connect the receiver to the drone or device you want to control.
2. **Binding:** Binding establishes a secure connection between the transmitter and receiver. Follow these steps:
 - a. Turn on the transmitter while holding down the Bind button (located on the transmitter's front side).
 - b. Power on the receiver while pressing and holding its Bind button (usually found on the receiver).
 - c. Release the Bind buttons once the receiver's LED light (usually located on the receiver) stops flashing and remains solid.
3. **Mode Selection:** The Flysky FS-T6 supports multiple flight modes. To select a mode:
 - a. Use the Mode switch on the transmitter to toggle between available modes (e.g., Mode 1, Mode 2, etc.).
 - b. Verify that the mode selection is appropriate for your flight preferences and the drone's configuration.
4. **Channel Assignments:** The Flysky FS-T6 offers several control channels for different functions. Configure the channel assignments according to your drone's setup:
 - a. Enter the System menu on the transmitter by pressing and holding the Menu button.
 - b. Use the navigation buttons (usually located below the LCD screen) to navigate the menus.
 - c. Locate the Channel Assign option and assign the desired functions to the appropriate channels.
5. **Failsafe Setup (Optional):** Configuring failsafe settings is crucial to ensure a safe response when the transmitter signal is lost. Steps for setting up failsafe vary depending on the specific receiver and drone setup. Refer to the transmitter and receiver manuals for detailed instructions on configuring failsafe.
6. **Trim Adjustments:** Trim adjustments fine-tune the control surfaces' neutral positions to ensure the drone hovers or moves without drifting. Trim buttons are typically located around the joysticks. Use these buttons to adjust the trim settings for each control channel until the drone remains stable and level when the sticks are centered.

7. Dual Rate and Expo (Optional): The Flysky FS-T6 provides features like Dual Rate and Expo, which can affect control sensitivity and response. Refer to the transmitter manual for instructions on adjusting these settings, if desired.

It's important to note that the specific steps and menus may vary slightly based on firmware versions and individual configurations. Always consult the Flysky FS-T6 manual for detailed instructions and refer to the documentation for your specific drone or device for further configuration guidance.



2.2.4 INTERNET

- The Internet (or internet) is the global system of interconnected computer networks that uses the Internet protocol suite (TCP/IP) to communicate between networks and devices.
- It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies. The Internet Droneries a vast range of information resources and services, such as the inter-linked hypertext documents and applications of the World Wide Web (WWW), electronic mail, telephony, and file sharing.
- The origins of the Internet date back to the development of packet switching and research commissioned by the United States Department of Defense in the 1960s to enable time-sharing of computers. The primary precursor network, the ARPANET, initially served as a backbone for interconnection of regional academic and military networks in the 1970s to enable resource sharing.



Figure 16

2.2.5 Circuit Diagram

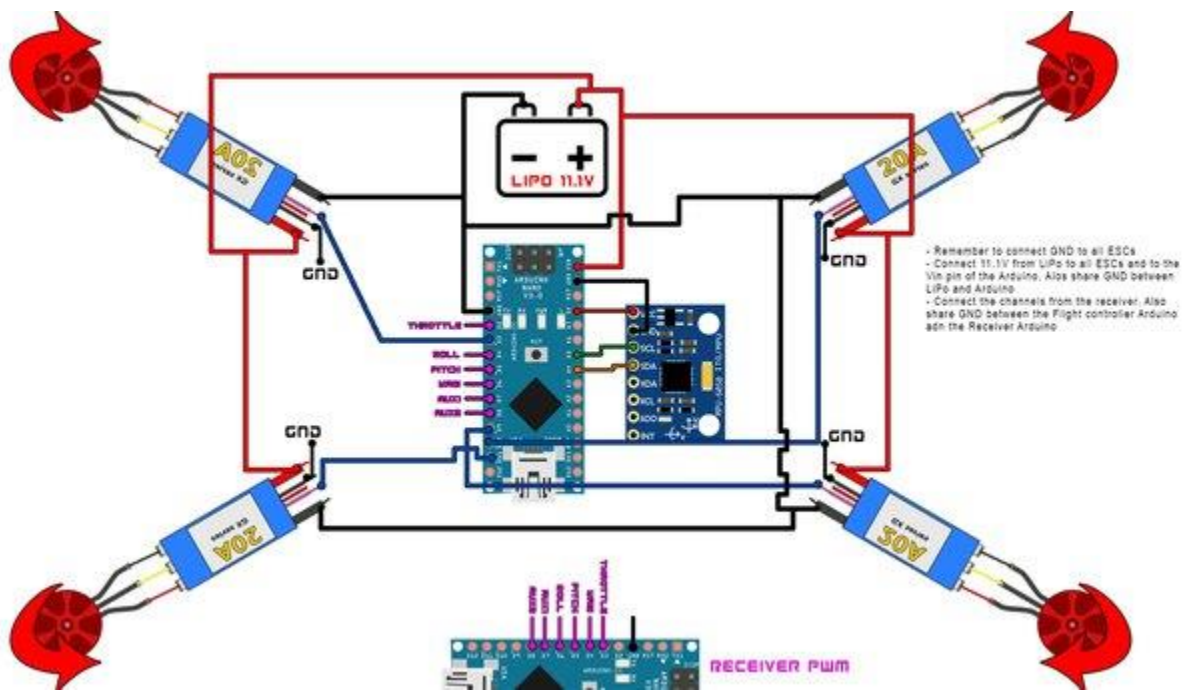


Figure 17: Circuit Diagram

CHAPTER 3 BUSINESS MODEL

A business model describes how an organization creates, delivers, and captures value, in economic, social, cultural, or other contexts. The process of business model construction and modification is also called business model innovation and forms a part of business strategy.

3.1 ER Diagram

Upon successfully pairing the device, open the app on the smart phone and press on the Bluetooth textual and emblematic pushbutton. The number of associated gadgets will now be shown. Select HC-05 from the listing to join the smart phone with HC-05 Bluetooth module on the receiver side. After successful connection, 'connected' will be displayed on the primary screen of Voice control app. Press the pushbutton with microphone image and a prompt will show up asking for voice commands.

- When it appears, voice instructions are detected via the app, which converts them into textual content and sends it to the receiver aspect wirelessly by using Bluetooth. On the receiving side, Arduino tests the text. If it is a matching string, it controls the moves of the robot in accordance to the description

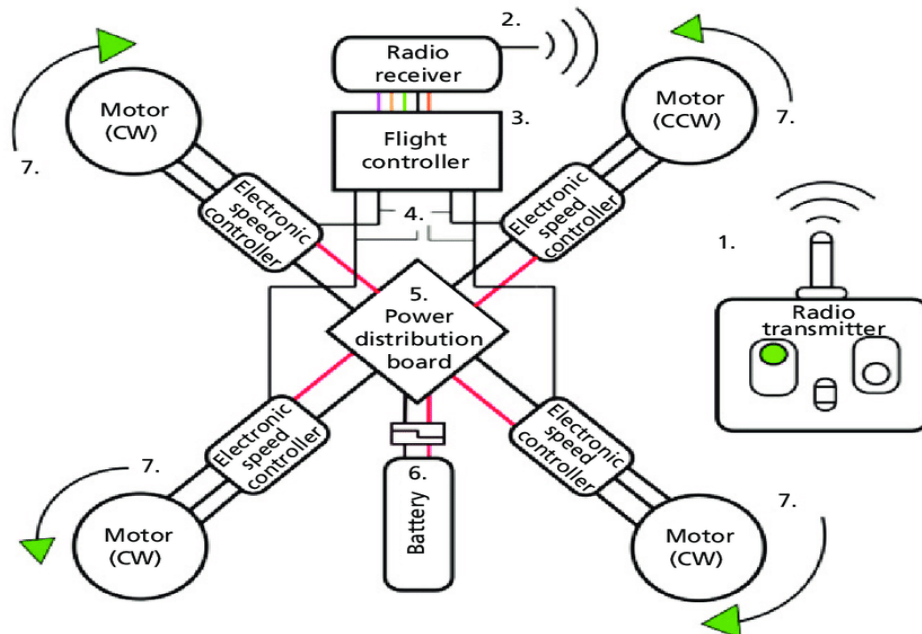


Figure 18: ER Diagram

3.2 DFD Diagram

□ What is Data Flow Diagram?

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method (SSADM).

1. What are the different DFD levels and layers?

Levels or layers are used in DFDs to represent progressive degrees of detail about the system or process. These levels include:

- **Level 0:** Also known as a "context diagram," this is the highest level and represents a very simple, top-level view of the system being represented.
- **Level 1:** Still a relatively broad view of the system, but incorporates subprocesses and more detail.
- **Level 2:** Provides even more detail and continues to break down sub processes as needed.
- **Level 3:** While this amount of detail is uncommon, complex systems can benefit from representation at this level.

3.3 SDLC Model

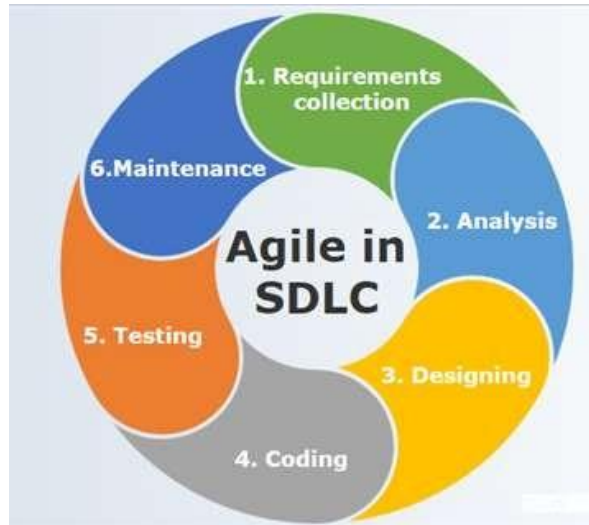
Agile Model - Agile methodology is a practice which promotes continuous interaction of development and testing during the SDLC process of any project. In the Agile method, the entire project is divided into small incremental builds. All of these builds are provided in iterations, and each iteration lasts from one to three weeks.

Any agile software phase is characterized in a manner that addresses several key assumptions about the bulk of software projects:

1. It is difficult to think in advance which software requirements will persist and which will change. It is equally difficult to predict how user priorities will change as the project proceeds.
2. For many types of software, design and development are interleaved. That is, both activities should be performed in tandem so that design models are proven

as they are created. It is difficult to think about how much design is necessary before construction is used to test the configuration.

3. Analysis, design, development, and testing are not as predictable (from a planning point of view) as we might like.



Different Types of SDLC Model:

- ☐ Waterfall Model
- ☐ Spiral Model
- ☐ V-Model
- ☐ Incremental Model
- ☐ Iterative Model

1. **Waterfall Model** - The waterfall is a universally accepted SDLC model. In this method, the whole process of software development is divided into various phases.

The waterfall model is a continuous software development model in which development is seen as flowing steadily downwards (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance.

2. Linear ordering of activities has some significant consequences. First, to identify the end of a phase and the beginning of the next, some certification techniques have to be employed at the end of each step. Some verification and validation usually do this mean that will ensure that the output of the stage is consistent with its input (which is the output of the previous step), and that the output of the stage is consistent with the overall requirements of the system.

3. **Spiral Model** - The spiral model is a **risk-driven process model**. This SDLC model helps the group to adopt elements of one or more process models like a waterfall, incremental, waterfall, etc. The spiral technique is a combination of rapid prototyping and concurrency in design and development activities.

Each cycle in the spiral begins with the identification of objectives for that cycle, the different alternatives that are possible for achieving the goals, and the constraints that exist. This is the first quadrant of the cycle (upper-left quadrant).

The next step in the cycle is to evaluate these different alternatives based on the objectives and constraints. The focus of evaluation in this step is based on the risk perception for the project.

The next step is to develop strategies that solve uncertainties and risks. This step may involve activities such as benchmarking, simulation, and prototyping.

4. **V- Model** - In this type of SDLC model testing and the development, the step is planned in parallel. So, there are verification phases on the side and the validation phase on the other side. V-Model joins by Coding phase.
5. **Incremental Model** - The incremental model is not a separate model. It is necessarily a series of waterfall cycles. The requirements are divided into groups at the start of the project. For each group, the SDLC model is followed to develop software. The SDLC process is repeated, with each release adding more functionality until all requirements are met. In this method, each cycle act as the maintenance phase for the previous software release. Modification to the incremental model allows development cycles to overlap. After that subsequent cycle may begin before the previous cycle is complete.
6. **Iterative Model** – It is a particular implementation of a software development life cycle that focuses on an initial, simplified implementation, which then progressively gains more complexity and a broader feature set until the final system is complete. In short, iterative development is a way of breaking down the software development of a large application into smaller pieces.

CHAPTER 4: CODING & SCREENSHOT

4.1 Drone Setup

```
#include
<Wire.h>
#include
<EEPROM.h>
byte last_channel_1,
last_channel_2,
last_channel_3,
last_channel_4;
byte lowByte, highByte,
type, gyro_address,
error, clockspeed_ok;
byte channel_1_assign,
channel_2_assign,
channel_3_assign,
channel_4_assign;
byte roll_axis,
pitch_axis, yaw_axis;
byte receiver_check_byte,
gyro_check_byte;
volatile int
receiver_input_channel_1,
receiver_input_channel_2,
receiver_input_channel_3,
receiver_input_channel_4;
int center_channel_1,
center_channel_2,
center_channel_3,
center_channel_4;
int high_channel_1,
high_channel_2,
high_channel_3,
high_channel_4;
int low_channel_1,
low_channel_2,
low_channel_3,
low_channel_4;
int address, cal_int;
unsigned long timer,
timer_1, timer_2,
timer_3, timer_4,
current_time;

float gyro_pitch,
gyro_roll, gyro_yaw;
float gyro_roll_cal,
gyro_pitch_cal,
gyro_yaw_cal;

void setup(){
  pinMode(12, OUTPUT);
  PCICR |= (1 <<
PCIE0);
  PCMSK0 |= (1 <<
PCINT0);
  PCMSK0 |= (1 <<
PCINT1);
  PCMSK0 |= (1 <<
PCINT2);
  PCMSK0 |= (1 <<
PCINT3);
  Wire.begin();
  Serial.begin(57600);

  delay(250);
}
void loop(){
  intro();

  Serial.println(F(""));
  Serial.println(F("====
=====
====="));
  Serial.println(F("System
check"));
  Serial.println(F("====
=====
====="));
  delay(1000);
  Serial.println(F("Check
ing I2C clock speed."));
  delay(1000);

  TWBR =
12;
  #if F_CPU ==
16000000L
    clockspeed_ok =
1;
  #endif

  if(TWBR == 12 &&
clockspeed_ok){
    Serial.println(F("I2C
clock speed is correctly
set to 400kHz."));
  }
  else{
    Serial.println(F("I2C
clock speed is not set to
400kHz. (ERROR 8)"));
    error = 1;
  }

  if(error == 0){
    Serial.println(F(""))
;
    Serial.println(F("====
=====
====="));
    Serial.println(F("Tra
nsmitter setup"));
    Serial.println(F("====
=====
====="));
    delay(1000);
    Serial.print(F("Check
ing for valid receiver
signals."));
    wait_for_receiver();
```

```

        Serial.println(F(""))
    ;
    }
    if(error == 0){
        delay(2000);
        Serial.println(F("Place all sticks and subtrims in the center position within 10 seconds."));
        for(int i = 9;i > 0;i--){
            delay(1000);
            Serial.print(i);
            Serial.print(" ");
        }
        Serial.println(" ");
        center_channel_1 = receiver_input_channel_1;
        center_channel_2 = receiver_input_channel_2;
        center_channel_3 = receiver_input_channel_3;
        center_channel_4 = receiver_input_channel_4;
        Serial.println(F(""))
    ;

    Serial.println(F("Center positions stored."));
    Serial.print(F("Digital input 08 = "));
    Serial.println(receiver_input_channel_1);
    Serial.print(F("Digital input 09 = "));
    Serial.println(receiver_input_channel_2);
    Serial.print(F("Digital input 10 = "));
    Serial.println(receiver_input_channel_3);
    Serial.print(F("Digital input 11 = "));
    Serial.println(receiver_input_channel_4);

```

```

        Serial.println(F(""))
    ;
    Serial.println(F(""))
    ;
    }
    if(error == 0){
        Serial.println(F("Move the throttle stick to full throttle and back to center"));

        check_receiver_inputs
(1);
        Serial.print(F("Throttle is connected to digital input "));
        Serial.println((channel_3_assign & 0b00000111) + 7);
        if(channel_3_assign & 0b10000000)Serial.println(F("Channel inverted = yes"));
        else
        Serial.println(F("Channel inverted = no"));
        wait_sticks_zero();

        Serial.println(F(""))
    ;
        Serial.println(F(""))
    ;
        Serial.println(F("Move the roll stick to simulate left wing up and back to center"));
        check_receiver_inputs
(2);
        Serial.print(F("Roll is connected to digital input "));
        Serial.println((channel_1_assign & 0b00000111) + 7);
        if(channel_1_assign & 0b10000000)Serial.println

```

```

(F("Channel inverted = yes"));
        else
        Serial.println(F("Channel inverted = no"));
        wait_sticks_zero();
    }
    if(error == 0){
        Serial.println(F(""))
    ;
        Serial.println(F(""))
    ;
        Serial.println(F("Move the pitch stick to simulate nose up and back to center"));

        check_receiver_inputs
(3);
        Serial.print(F("Pitch is connected to digital input "));
        Serial.println((channel_2_assign & 0b00000111) + 7);
        if(channel_2_assign & 0b10000000)Serial.println(F("Channel inverted = yes"));
        else
        Serial.println(F("Channel inverted = no"));
        wait_sticks_zero();
    }
    if(error == 0){
        Serial.println(F(""))
    ;
        Serial.println(F(""))
    ;
        Serial.println(F("Move the yaw stick to simulate nose right and back to center"));
        check_receiver_inputs
(4);

```

```

    Serial.print(F("Yaw
is connected to digital
input "));
    Serial.println((chann
el_4_assign & 0b00000111)
+ 7);
    if(channel_4_assign &
0b10000000)Serial.println
(F("Channel inverted =
yes"));
    else
Serial.println(F("Channel
inverted = no"));
    wait_sticks_zero();
}
if(error == 0){
    Serial.println(F(""))
;
    Serial.println(F(""))
;
    Serial.println(F("Gen
tly move all the sticks
simultaneously to their
extends"));
    Serial.println(F("Whe
n ready put the sticks
back in their center
positions"));
    register_min_max();
    Serial.println(F(""))
;
    Serial.println(F(""))
;
    Serial.println(F("Hig
h, low and center values
found during setup"));
    Serial.print(F("Digit
al input 08 values:"));
    Serial.print(low_chan
nel_1);
    Serial.print(F(" -
"));
    Serial.print(center_c
hannel_1);
    Serial.print(F(" -
"));

```

```

    Serial.println(high_c
hannel_1);
    Serial.print(F("Digit
al input 09 values:"));
    Serial.print(low_chan
nel_2);
    Serial.print(F(" -
"));
    Serial.print(center_c
hannel_2);
    Serial.print(F(" -
"));
    Serial.println(high_c
hannel_2);
    Serial.print(F("Digit
al input 10 values:"));
    Serial.print(low_chan
nel_3);
    Serial.print(F(" -
"));
    Serial.print(center_c
hannel_3);
    Serial.print(F(" -
"));
    Serial.println(high_c
hannel_3);
    Serial.print(F("Digit
al input 11 values:"));
    Serial.print(low_chan
nel_4);
    Serial.print(F(" -
"));
    Serial.print(center_c
hannel_4);
    Serial.print(F(" -
"));
    Serial.println(high_c
hannel_4);
    Serial.println(F("Mov
e stick 'nose up' and
back to center to
continue"));
    check_to_continue();
}
if(error == 0){

```

```

//What gyro is
connected
    Serial.println(F(""))
;
    Serial.println(F("===
=====
=====
=====
"));
    Serial.println(F("Gyr
o search"));
    Serial.println(F("===
=====
=====
=====
"));
    delay(2000);

    Serial.println(F("Sea
rching for MPU-6050 on
address 0x68/104"));
    delay(1000);
    if(search_gyro(0x68,
0x75) == 0x68){
        Serial.println(F("M
PU-6050 found on address
0x68"));
        type = 1;
        gyro_address =
0x68;
    }

    if(type == 0){
        Serial.println(F("S
earching for MPU-6050 on
address 0x69/105"));
        delay(1000);
        if(search_gyro(0x69
, 0x75) == 0x68){
            Serial.println(F(
"MPU-6050 found on
address 0x69"));
            type = 1;
            gyro_address =
0x69;
        }
    }
}

```

```

        if(type == 0){
            Serial.println(F("S
earching for L3G4200D on
address 0x68/104"));
            delay(1000);
            if(search_gyro(0x68
, 0x0F) == 0xD3){
                Serial.println(F(
"L3G4200D found on
address 0x68"));
                type = 2;
                gyro_address =
0x68;
            }
        }

        if(type == 0){
            Serial.println(F("S
earching for L3G4200D on
address 0x69/105"));
            delay(1000);
            if(search_gyro(0x69
, 0x0F) == 0xD3){
                Serial.println(F(
"L3G4200D found on
address 0x69"));
                type = 2;
                gyro_address =
0x69;
            }
        }

        if(type == 0){
            Serial.println(F("S
earching for L3GD20H on
address 0x6A/106"));
            delay(1000);
            if(search_gyro(0x6A
, 0x0F) == 0xD7){
                Serial.println(F(
"L3GD20H found on address
0x6A"));
                type = 3;
                gyro_address =
0x6A;
            }
        }

        if(type == 0){
            Serial.println(F("Se
arching for L3GD20H on
address 0x6B/107"));
            delay(1000);
            if(search_gyro(0x6B
, 0x0F) == 0xD7){
                Serial.println(F(
"L3GD20H found on address
0x6B"));
                type = 3;
                gyro_address =
0x6B;
            }
        }

        if(type == 0){
            Serial.println(F("N
o gyro device found!!!
(ERROR 3)"));
            error = 1;
        }

        else{
            delay(3000);
            Serial.println(F("
));
            Serial.println(F("=
=====
=====
"));
            Serial.println(F("G
yro register settings"));
            Serial.println(F("=
=====
=====
"));
            start_gyro();
        }

        if(error == 0){
            delay(3000);
        }
    }

    Serial.println(F(""))
;
    Serial.println(F("==
=====
=====
"));
    Serial.println(F("Gyr
o calibration"));
    Serial.println(F("==
=====
=====
"));
    Serial.println(F("Don
't move the quadcopter!!
Calibration starts in 3
seconds"));
    delay(3000);
    Serial.println(F("Cal
ibrating the gyro, this
will take +/- 8
seconds"));
    Serial.print(F("Pleas
e wait"));
    for (cal_int = 0;
cal_int < 2000 ; cal_int
++){
        if(cal_int % 100 ==
0)Serial.print(F("."));

        gyro_signalen();

        gyro_roll_cal +=
gyro_roll;
        gyro_pitch_cal +=
gyro_pitch;
        gyro_yaw_cal +=
gyro_yaw;
        delay(4);
    }
    gyro_roll_cal /=
2000;
    gyro_pitch_cal /=
2000;
    gyro_yaw_cal /= 2000
    Serial.println(F(""))
;

```



```

    Serial.println(F("LED
test"));
    Serial.println(F("===
=====
====="));
);
    digitalWrite(12,
HIGH);
    Serial.println(F("The
LED should now be lit"));
    Serial.println(F("Mov
e stick 'nose up' and
back to center to
continue"));
    check_to_continue();
    digitalWrite(12,
LOW);
}

    Serial.println(F(""));

    if(error == 0){
        Serial.println(F("===
=====
====="));
);
        Serial.println(F("Fin
al setup check"));
        Serial.println(F("===
=====
====="));
);
        delay(1000);
        if(receiver_check_byt
e == 0b00001111){
            Serial.println(F("R
eceiver channels ok"));
        }
        else{
            Serial.println(F("R
eceiver channel
verification failed!!!
(ERROR 6)"));
            error = 1;
        }
        delay(1000);

```

```

        if(gyro_check_byte ==
0b00000111){
            Serial.println(F("G
yro axes ok"));
        }
        else{
            Serial.println(F("G
yro axes verification
failed!!! (ERROR 7)"));
            error = 1;
        }
    }

    if(error == 0){
        //If all is good,
store the information in
the EEPROM
        Serial.println(F(""))
;
        Serial.println(F("===
=====
====="));
);
        Serial.println(F("Sto
ring EEPROM
information"));
        Serial.println(F("===
=====
====="));
);
        Serial.println(F("Wri
ting EEPROM"));
        delay(1000);
        Serial.println(F("Don
e!"));
        EEPROM.write(0,
center_channel_1 &
0b11111111);
        EEPROM.write(1,
center_channel_1 >> 8);
        EEPROM.write(2,
center_channel_2 &
0b11111111);
        EEPROM.write(3,
center_channel_2 >> 8);

```

```

        EEPROM.write(4,
center_channel_3 &
0b11111111);
        EEPROM.write(5,
center_channel_3 >> 8);
        EEPROM.write(6,
center_channel_4 &
0b11111111);
        EEPROM.write(7,
center_channel_4 >> 8);
        EEPROM.write(8,
high_channel_1 &
0b11111111);
        EEPROM.write(9,
high_channel_1 >> 8);
        EEPROM.write(10,
high_channel_2 &
0b11111111);
        EEPROM.write(11,
high_channel_2 >> 8);
        EEPROM.write(12,
high_channel_3 &
0b11111111);
        EEPROM.write(13,
high_channel_3 >> 8);
        EEPROM.write(14,
high_channel_4 &
0b11111111);
        EEPROM.write(15,
high_channel_4 >> 8);
        EEPROM.write(16,
low_channel_1 &
0b11111111);
        EEPROM.write(17,
low_channel_1 >> 8);
        EEPROM.write(18,
low_channel_2 &
0b11111111);
        EEPROM.write(19,
low_channel_2 >> 8);
        EEPROM.write(20,
low_channel_3 &
0b11111111);
        EEPROM.write(21,
low_channel_3 >> 8);

```

```

EEPROM.write(22,
low_channel_4 &
0b11111111);
EEPROM.write(23,
low_channel_4 >> 8);
EEPROM.write(24,
channel_1_assign);
EEPROM.write(25,
channel_2_assign);
EEPROM.write(26,
channel_3_assign);
EEPROM.write(27,
channel_4_assign);
EEPROM.write(28,
roll_axis);
EEPROM.write(29,
pitch_axis);
EEPROM.write(30,
yaw_axis);
EEPROM.write(31,
type);
EEPROM.write(32,
gyro_address);
//Write the EEPROM
signature
EEPROM.write(33,
'J');
EEPROM.write(34,
'M');
EEPROM.write(35,
'B');

Serial.println(F("Ver
ify EEPROM data"));
delay(1000);
if(center_channel_1
!= ((EEPROM.read(1) << 8)
| EEPROM.read(0)))error =
1;

if(center_channel_2
!= ((EEPROM.read(3) << 8)
| EEPROM.read(2)))error =
1;

if(center_channel_3
!= ((EEPROM.read(5) << 8)

```

```

| EEPROM.read(4)))error =
1;

if(center_channel_4
!= ((EEPROM.read(7) << 8)
| EEPROM.read(6)))error =
1;

if(high_channel_1 !=
((EEPROM.read(9) << 8) |
EEPROM.read(8)))error =
1;

if(high_channel_2 !=
((EEPROM.read(11) << 8) |
EEPROM.read(10)))error =
1;

if(high_channel_3 !=
((EEPROM.read(13) << 8) |
EEPROM.read(12)))error =
1;

if(high_channel_4 !=
((EEPROM.read(15) << 8) |
EEPROM.read(14)))error =
1;

if(low_channel_1 !=
((EEPROM.read(17) << 8) |
EEPROM.read(16)))error =
1;

if(low_channel_2 !=
((EEPROM.read(19) << 8) |
EEPROM.read(18)))error =
1;

if(low_channel_3 !=
((EEPROM.read(21) << 8) |
EEPROM.read(20)))error =
1;

if(low_channel_4 !=
((EEPROM.read(23) << 8) |
EEPROM.read(22)))error =
1;

if(channel_1_assign
!= EEPROM.read(24))error
= 1;

```

```

if(channel_2_assign
!= EEPROM.read(25))error
= 1;

if(channel_3_assign
!= EEPROM.read(26))error
= 1;

if(channel_4_assign
!= EEPROM.read(27))error
= 1;

if(roll_axis !=
EEPROM.read(28))error =
1;

if(pitch_axis !=
EEPROM.read(29))error =
1;

if(yaw_axis !=
EEPROM.read(30))error =
1;

if(type !=
EEPROM.read(31))error =
1;

if(gyro_address !=
EEPROM.read(32))error =
1;

if('J' !=
EEPROM.read(33))error =
1;

if('M' !=
EEPROM.read(34))error =
1;

if('B' !=
EEPROM.read(35))error =
1;

if(error ==
1)Serial.println(F("EEPRO
M verification failed!!!
(ERROR 5)"));
else
Serial.println(F("Verific
ation done"));
}

```



```

    if(error == 0){
        Serial.println(F("Set
up is finished."));
        Serial.println(F("You
can now calibrate the
esc's and upload the
YMFC-AL code."));
    }
    else{
        Serial.println(F("The
setup is aborted due to
an error."));
        Serial.println(F("Chec
k the Q and A page of the
YMFC-AL project on:"));
        Serial.println(F("www.
brokking.net for more
information about this
error."));
    }
    while(1);
}
byte search_gyro(int
gyro_address, int
who_am_i){
    Wire.beginTransmissio
n(gyro_address);
    Wire.write(who_am_i);
    Wire.endTransmission();
    Wire.requestFrom(gyro_a
ddress, 1);
    timer = millis() + 100;
    while(Wire.available()
< 1 && timer > millis());
    lowByte = Wire.read();
    address = gyro_address;
    return lowByte;
}

void start_gyro(){
    if(type == 2 || type ==
3){
        Wire.beginTransmissio
n(address);
        Wire.write(0x20);
        Wire.write(0x0F);

```

```

        Wire.endTransmission(
);
        Wire.beginTransmissio
n(address);
        Wire.write(0x20);
        Wire.endTransmission(
);
        Wire.requestFrom(addr
ess,
1);
        while(Wire.available(
) <
1);
        Serial.print(F("Regis
ter 0x20 is set to:"));
        Serial.println(Wire.r
ead(),BIN);

        Wire.beginTransmissio
n(address);
        Wire.write(0x23);
        Wire.write(0x90);

        Wire.endTransmission(
);

        Wire.beginTransmissio
n(address);
        Wire.write(0x23);

        Wire.endTransmission(
);
        Wire.requestFrom(addr
ess, 1);
        while(Wire.available(
) < 1);
        Serial.print(F("Regis
ter 0x23 is set to:"));
        Serial.println(Wire.r
ead(),BIN);
    }
    if(type == 1){

```

```

        Wire.beginTransmissio
n(address);
        Wire.write(0x6B);

        Wire.write(0x00);

        Wire.endTransmission(
);

        Wire.beginTransmissio
n(address);
        Wire.write(0x6B);

        Wire.endTransmission(
);
        Wire.requestFrom(addr
ess, 1);
        while(Wire.available(
) < 1);
        Serial.print(F("Regis
ter 0x6B is set to:"));
        Serial.println(Wire.r
ead(),BIN);

        Wire.beginTransmissio
n(address);
        Wire.write(0x1B);

        Wire.write(0x08);

        Wire.endTransmission(
);

        Wire.beginTransmissio
n(address);
        Wire.write(0x1B);
        Wire.endTransmission(
);
        Wire.requestFrom(addr
ess, 1);
        while(Wire.available(
) < 1);
        Serial.print(F("Regis
ter 0x1B is set to:"));
        Serial.println(Wire.r
ead(),BIN);    }

```



```

}

void gyro_signalen(){
    if(type == 2 || type
    == 3){
        Wire.beginTransaction(
n(address);
        Wire.write(168);

        Wire.endTransmission(
);
        Wire.requestFrom(addr
ess, 6);
        while(Wire.available(
) < 6);
        lowByte =
Wire.read();
        highByte =
Wire.read();
        gyro_roll =
((highByte<<8)|lowByte);

        if(cal_int ==
2000)gyro_roll -=
gyro_roll_cal;
        lowByte =
Wire.read();
        highByte =
Wire.read();
        gyro_pitch =
((highByte<<8)|lowByte);
        if(cal_int ==
2000)gyro_pitch -=
gyro_pitch_cal;
        lowByte =
Wire.read();
        highByte =
Wire.read();
        gyro_yaw =
((highByte<<8)|lowByte);
        if(cal_int ==
2000)gyro_yaw -=
gyro_yaw_cal;
    }
    if(type == 1){
        Wire.beginTransaction(
n(address);
        Wire.write(0x43);
        Wire.endTransmission(
);
        Wire.requestFrom(addr
ess,6);
        while(Wire.available(
) <
6);
        gyro_roll=Wire.read()
<<8|Wire.read();
        if(cal_int ==
2000)gyro_roll -=
gyro_roll_cal;
        gyro_pitch=Wire.read(
)<<8|Wire.read();
        if(cal_int ==
2000)gyro_pitch -=
gyro_pitch_cal;
        gyro_yaw=Wire.read()<
<8|Wire.read();
        if(cal_int ==
2000)gyro_yaw -=
gyro_yaw_cal;
    }
}

void
check_receiver_inputs(byte
movement){
    byte trigger = 0;
    int pulse_length;
    timer = millis() +
30000;
    while(timer > millis()
&& trigger == 0){
        delay(250);
        if(receiver_input_cha
nnel_1 > 1750 ||
receiver_input_channel_1
< 1250){
            trigger = 1;
            receiver_check_byte
|= 0b00000001;
            pulse_length =
receiver_input_channel_1;
        }
    }
    if(receiver_input_cha
nnel_2 > 1750 ||
receiver_input_channel_2
< 1250){
        trigger = 2;
        receiver_check_byte
|= 0b00000010;
        pulse_length =
receiver_input_channel_2;
    }
    if(receiver_input_cha
nnel_3 > 1750 ||
receiver_input_channel_3
< 1250){
        trigger = 3;
        receiver_check_byte
|= 0b00000100;
        pulse_length =
receiver_input_channel_3;
    }
    if(receiver_input_cha
nnel_4 > 1750 ||
receiver_input_channel_4
< 1250){
        trigger = 4;
        receiver_check_byte
|= 0b00001000;
        pulse_length =
receiver_input_channel_4;
    }
    if(trigger == 0){
        error = 1;
        Serial.println(F("No
stick movement detected
in the last 30 seconds!!!
(ERROR 2)"));
    }
    else{
        if(movement == 1){
            channel_3_assign =
trigger;
            if(pulse_length <
1250)channel_3_assign +=
0b10000000;
        }
    }
}

```

```

    }
    if(movement == 2){
        channel_1_assign =
trigger;
        if(pulse_length <
1250)channel_1_assign +=
0b10000000;
    }
    if(movement == 3){
        channel_2_assign =
trigger;
        if(pulse_length <
1250)channel_2_assign +=
0b10000000;
    }
    if(movement == 4){
        channel_4_assign =
trigger;
        if(pulse_length <
1250)channel_4_assign +=
0b10000000;
    }
}

void check_to_continue(){
    byte continue_byte = 0;
    while(continue_byte ==
0){
        if(channel_2_assign
== 0b00000001 &&
receiver_input_channel_1
> center_channel_1 +
150)continue_byte = 1;
        if(channel_2_assign
== 0b10000001 &&
receiver_input_channel_1
< center_channel_1 -
150)continue_byte = 1;
        if(channel_2_assign
== 0b00000010 &&
receiver_input_channel_2
> center_channel_2 +
150)continue_byte = 1;
        if(channel_2_assign
== 0b10000010 &&

```

```

receiver_input_channel_2
< center_channel_2 -
150)continue_byte = 1;
        if(channel_2_assign
== 0b00000011 &&
receiver_input_channel_3
> center_channel_3 +
150)continue_byte = 1;
        if(channel_2_assign
== 0b10000011 &&
receiver_input_channel_3
< center_channel_3 -
150)continue_byte = 1;
        if(channel_2_assign
== 0b00000100 &&
receiver_input_channel_4
> center_channel_4 +
150)continue_byte = 1;
        if(channel_2_assign
== 0b10000100 &&
receiver_input_channel_4
< center_channel_4 -
150)continue_byte = 1;
        delay(100);
    }
    wait_sticks_zero();
}

void wait_sticks_zero(){
    byte zero = 0;
    while(zero < 15){
        if(receiver_input_cha
nnel_1 < center_channel_1
+ 20 &&
receiver_input_channel_1
> center_channel_1 -
20)zero |= 0b00000001;
        if(receiver_input_cha
nnel_2 < center_channel_2
+ 20 &&
receiver_input_channel_2
> center_channel_2 -
20)zero |= 0b00000010;
        if(receiver_input_cha
nnel_3 < center_channel_3
+ 20 &&
receiver_input_channel_3

```

```

> center_channel_3 -
20)zero |= 0b00000100;
        if(receiver_input_cha
nnel_4 < center_channel_4
+ 20 &&
receiver_input_channel_4
> center_channel_4 -
20)zero |= 0b00001000;
        delay(100);
    }
}

void wait_for_receiver(){
    byte zero = 0;
    timer = millis() +
10000;
    while(timer > millis()
&& zero < 15){
        if(receiver_input_cha
nnel_1 < 2100 &&
receiver_input_channel_1
> 900)zero |= 0b00000001;
        if(receiver_input_cha
nnel_2 < 2100 &&
receiver_input_channel_2
> 900)zero |= 0b00000010;
        if(receiver_input_cha
nnel_3 < 2100 &&
receiver_input_channel_3
> 900)zero |= 0b00000100;
        if(receiver_input_cha
nnel_4 < 2100 &&
receiver_input_channel_4
> 900)zero |= 0b00001000;
        delay(500);
        Serial.print(F("."));
    }
    if(zero == 0){
        error = 1;
        Serial.println(F("."))
);
        Serial.println(F("No
valid receiver signals
found!!! (ERROR 1)"));
    }
    else Serial.println(F("
OK"));
}

```

```

}
void register_min_max(){
    byte zero = 0;
    low_channel_1 =
    receiver_input_channel_1;
    low_channel_2 =
    receiver_input_channel_2;
    low_channel_3 =
    receiver_input_channel_3;
    low_channel_4 =
    receiver_input_channel_4;
    while(receiver_input_ch
    annel_1 <
    center_channel_1 + 20 &&
    receiver_input_channel_1
    > center_channel_1 -
    20)delay(250);
    Serial.println(F("Measu
    ring endpoints...."));
    while(zero < 15){
        if(receiver_input_cha
        nnel_1 < center_channel_1
        + 20 &&
        receiver_input_channel_1
        > center_channel_1 -
        20)zero |= 0b00000001;
        if(receiver_input_cha
        nnel_2 < center_channel_2
        + 20 &&
        receiver_input_channel_2
        > center_channel_2 -
        20)zero |= 0b00000010;
        if(receiver_input_cha
        nnel_3 < center_channel_3
        + 20 &&
        receiver_input_channel_3
        > center_channel_3 -
        20)zero |= 0b00000100;
        if(receiver_input_cha
        nnel_4 < center_channel_4
        + 20 &&
        receiver_input_channel_4
        > center_channel_4 -
        20)zero |= 0b00001000;
        if(receiver_input_cha
        nnel_1 <

```

```

low_channel_1)low_channel
_1 =
    receiver_input_channel_1;
    if(receiver_input_cha
    nnel_2 <
    low_channel_2)low_channel
    _2 =
    receiver_input_channel_2;
    if(receiver_input_cha
    nnel_3 <
    low_channel_3)low_channel
    _3 =
    receiver_input_channel_3;
    if(receiver_input_cha
    nnel_4 <
    low_channel_4)low_channel
    _4 =
    receiver_input_channel_4;
    if(receiver_input_cha
    nnel_1 >
    high_channel_1)high_chann
    el_1 =
    receiver_input_channel_1;
    if(receiver_input_cha
    nnel_2 >
    high_channel_2)high_chann
    el_2 =
    receiver_input_channel_2;
    if(receiver_input_cha
    nnel_3 >
    high_channel_3)high_chann
    el_3 =
    receiver_input_channel_3;
    if(receiver_input_cha
    nnel_4 >
    high_channel_4)high_chann
    el_4 =
    receiver_input_channel_4;
    delay(100);
    }
}
void check_gyro_axes(byte
movement){
    byte trigger_axis = 0;

```

```

    float gyro_angle_roll,
    gyro_angle_pitch,
    gyro_angle_yaw;
    //Reset all axes
    gyro_angle_roll = 0;
    gyro_angle_pitch = 0;
    gyro_angle_yaw = 0;
    gyro_signalen();
    timer = millis() +
    10000;
    while(timer > millis()
    && gyro_angle_roll > -30
    && gyro_angle_roll < 30
    && gyro_angle_pitch > -30
    && gyro_angle_pitch < 30
    && gyro_angle_yaw > -30
    && gyro_angle_yaw < 30){
        gyro_signalen();
        if(type == 2 || type
        == 3){
            gyro_angle_roll +=
            gyro_roll * 0.00007;
            gyro_angle_pitch +=
            gyro_pitch * 0.00007;
            gyro_angle_yaw +=
            gyro_yaw * 0.00007;
        }
        if(type == 1){
            gyro_angle_roll +=
            gyro_roll *
            0.0000611;
            gyro_angle_pitch +=
            gyro_pitch * 0.0000611;
            gyro_angle_yaw +=
            gyro_yaw * 0.0000611;
        }

        delayMicroseconds(370
        0);
    }
    if((gyro_angle_roll < -
    30 || gyro_angle_roll >
    30) && gyro_angle_pitch >
    -30 && gyro_angle_pitch <
    30 && gyro_angle_yaw > -

```

```

30 && gyro_angle_yaw <
30){
    gyro_check_byte |=
0b00000001;
    if(gyro_angle_roll <
0)trigger_axis =
0b10000001;
    else trigger_axis =
0b00000001;
}
if((gyro_angle_pitch <
-30 || gyro_angle_pitch >
30) && gyro_angle_roll >
-30 && gyro_angle_roll <
30 && gyro_angle_yaw > -
30 && gyro_angle_yaw <
30){
    gyro_check_byte |=
0b00000010;
    if(gyro_angle_pitch <
0)trigger_axis =
0b10000010;
    else trigger_axis =
0b00000010;
}
if((gyro_angle_yaw < -
30 || gyro_angle_yaw >
30) && gyro_angle_roll >
-30 && gyro_angle_roll <
30 && gyro_angle_pitch >
-30 && gyro_angle_pitch <
30){
    gyro_check_byte |=
0b00000100;
    if(gyro_angle_yaw <
0)trigger_axis =
0b10000011;
    else trigger_axis =
0b00000011;
}

if(trigger_axis == 0){
    error = 1;
    Serial.println(F("No
angular motion is

```

```

detected in the last 10
seconds!!! (ERROR 4)"));
}
else
    if(movement ==
1)roll_axis =
trigger_axis;
    if(movement ==
2)pitch_axis =
trigger_axis;
    if(movement ==
3)yaw_axis =
trigger_axis;
}
ISR(PCINT0_vect){
    current_time =
micros();
    if(PINB & B00000001){
        if(last_channel_1 ==
0){
            last_channel_1 =
1;
            timer_1 =
current_time;
        }
        else if(last_channel_1
== 1){
            last_channel_1 =
0;
            receiver_input_channe
l_1 = current_time -
timer_1;
        }
        if(PINB & B00000010
){
            if(last_channel_2 ==
0){
                last_channel_2 =
1;
                timer_2 =
current_time;
            }
        }
    }
}

```

```

    else if(last_channel_2
== 1){
        last_channel_2 =
0;
        receiver_input_channe
l_2 = current_time -
timer_2;
    }
    if(PINB & B00000100 ){
        if(last_channel_3 ==
0){
            last_channel_3 =
1;
            timer_3 =
current_time;
        }
        else if(last_channel_3
==
1){
            last_channel_3 =
0;
            receiver_input_channe
l_3 = current_time -
timer_3;
        }
        if(PINB & B00001000 ){
            if(last_channel_4 ==
0){
                last_channel_4 =
1;
                timer_4 =
current_time;
            }
        }
        else if(last_channel_4
==
1){
            last_channel_4 =
0;
            receiver_input_channe
l_4 = current_time -
timer_4;
        }
    }
}

```

```
void intro(){
    Serial.println(F("====
=====
====="));
    delay(1500);
    Serial.println(F(""));
    Serial.println(F("Your"
));
    delay(500);
    Serial.println(F("  Mul
ticopter"));
}
```

```

    delay(500);
    Serial.println(F("      F
light"));
    delay(500);
    Serial.println(F("
Controller"));
    delay(1000);
    Serial.println(F(""));
    Serial.println(F("YMFC-
AL Setup Program"));
    Serial.println(F(""));

```

```
Serial.println(F("=====  
=====  
====="));  
delay(1500);  
Serial.println(F("For  
support and questions:  
www.brokking.net"));  
Serial.println(F(""));  
Serial.println(F("Have  
fun!"));  
}
```

4.2 Esc Calibrate

```
#include <Wire.h>
#include <EEPROM.h>

byte last_channel_1,
last_channel_2,
last_channel_3,
last_channel_4;

byte eeprom_data[36],
start, data;

boolean

new_function_request, first
_angle;

volatile int

receiver_input_channel_1,
receiver_input_channel_2,
receiver_input_channel_3,
receiver_input_channel_4;

int esc_1, esc_2, esc_3,
esc_4;

int counter_channel_1,
counter_channel_2,
counter_channel_3,
counter_channel_4;

int receiver_input[5];

int loop_counter,
gyro_address,
vibration_counter;

int temperature;

long acc_x, acc_y, acc_z,
acc_total_vector[20],
acc_av_vector,
vibration total result;
```

```
unsigned long
timer_channel_1,
timer_channel_2,
timer_channel_3,
timer_channel_4,
esc_timer, esc_loop_timer;
unsigned long zero_timer,
timer_1, timer_2, timer_3,
timer_4, current_time;
```

```
int acc_axis[4],
gyro_axis[4];
double gyro_pitch,
gyro_roll, gyro_yaw;
float angle_roll_acc,
angle_pitch_acc,
angle_pitch, angle_roll;
int cal_int;
double gyro_axis_cal[4];
void setup(){
    Serial.begin(57600);
    Wire.begin();
    TWBR =
12;
    DDRD |=
B11110000;
    DDRB |=
B00010000;
```

```

PCICR |= (1 << PCIE0);
PCMSK0 |= (1 <<
PCINT0);
PCMSK0 |= (1 <<
PCINT1);
PCMSK0 |= (1 <<
PCINT2);
PCMSK0 |= (1 <<
PCINT3);

```

```

    for(data = 0; data <=
35;
data++)eeprom_data[data] =
EEPROM.read(data);

gyro_address =
eeprom_data[32];

    set_gyro_registers();
    while(eeprom_data[33] !=
'J' || eeprom_data[34] !=
'M' || eeprom_data[35] !=
'B'){
        delay(500);
        digitalWrite(12,
!digitalRead(12));
    }
    wait for receiver();

```

```

    zero_timer =
micros();

    while(Serial.available()
)data = Serial.read();
    data = 0;
}
void loop(){
    while(zero_timer + 4000
> micros());
    zero_timer = micros();

    if(Serial.available() >
0){
        data =
Serial.read();
        delay(100);
        while(Serial.available
() > 0)loop_counter =
Serial.read();
        new_function_request =
true;
        loop_counter =
0;
        cal_int =
0;
        start =
0;
        first_angle =
false;
        if(data ==
'r')Serial.println("Readin
g receiver signals.");
        if(data ==
'a')Serial.println("Print
the quadcopter angles.");
        if(data ==
'a')Serial.println("Gyro
calibration starts in 2
seconds (don't move the
quadcopter).");
        if(data ==
'1')Serial.println("Test
motor 1 (right front
CCW.)");

```

```

        if(data ==
'2')Serial.println("Test
motor 2 (right rear
CW.)");
        if(data ==
'3')Serial.println("Test
motor 3 (left rear
CCW.)");
        if(data ==
'4')Serial.println("Test
motor 4 (left front
CW.)");
        if(data ==
'5')Serial.println("Test
all motors together");
        for(vibration_counter
= 0; vibration_counter <
625; vibration_counter++){
            delay(3);
            esc_1=1000;
            esc_2=1000;
            esc_3=1000;
            esc_4=1000;
            esc_pulse_output();
        }
        vibration_counter =
0;
    }

    receiver_input_channel_3
=
convert_receiver_channel(3
);
    if(receiver_input_channe
l_3 <
1025)new_function_request
= false;
    if(data == 0 &&
new_function_request ==
false){
        //Only
start the calibration mode
at first start.
        receiver_input_channel
_3 =

```

```

convert_receiver_channel(3
);
        esc_1 =
receiver_input_channel_3;
        esc_2 =
receiver_input_channel_3;
        esc_3 =
receiver_input_channel_3;
        esc_4 =
receiver_input_channel_3;
        esc_pulse_output();
    }
    if(data == 'r'){
        loop_counter ++;
        receiver_input_channel
_1 =
convert_receiver_channel(1
);
        receiver_input_channel
_2 =
convert_receiver_channel(2
);
        receiver_input_channel
_3 =
convert_receiver_channel(3
);
        receiver_input_channel
_4 =
convert_receiver_channel(4
);

        if(loop_counter ==
125){
            print_signals();
            loop_counter =
0;
        }
        if(receiver_input_chan
nel_3 < 1050 &&
receiver_input_channel_4 <
1050)start = 1;
        if(start == 1 &&
receiver_input_channel_3 <
1050 &&
receiver_input_channel_4 >
1450)start = 2;

```

```

    if(start == 2 &&
receiver_input_channel_3 <
1050 &&
receiver_input_channel_4 >
1950)start = 0;
    esc_1=1000;
    esc_2=1000;
    esc_3=1000;
    esc_4=1000;
    esc_pulse_output();
}
if(data == '1' || data
== '2' || data == '3' ||
data == '4' || data ==
'5'){
    loop_counter ++;
    if(new_function_reques
t == true && loop_counter
== 250){
        Serial.print("Set
throttle to 1000 (low).
It's now set to:
");
        Serial.println(recei
ver_input_channel_3);
        loop_counter =
0;
    }
    if(new_function_reques
t == false){
        receiver_input_chann
el_3 =
convert_receiver_channel(3
);
        if(data == '1' ||
data == '5')esc_1 =
receiver_input_channel_3;
        else esc_1 =
1000;

        if(data == '2' ||
data == '5')esc_2 =
receiver_input_channel_3;

```

```

        else esc_2 =
1000;
        if(data == '3' ||
data == '5')esc_3 =
receiver_input_channel_3;
        else esc_3 =
1000;
        if(data == '4' ||
data == '5')esc_4 =
receiver_input_channel_3;

        else esc_4 =
1000;

        esc_pulse_output();
        if(eeprom_data[31]
==
1){
            Wire.beginTransmis
sion(gyro_address);
            Wire.write(0x3B);
            Wire.endTransmissi
on();
            Wire.requestFrom(g
yro_address,6);
            while(Wire.availab
le() <
6);
            acc_x =
Wire.read()<<8|Wire.read()
;
            acc_y =
Wire.read()<<8|Wire.read()
;
            acc_z =
Wire.read()<<8|Wire.read()
;
            acc_total_vector[0
] =
sqrt((acc_x*acc_x)+(acc_y*
acc_y)+(acc_z*acc_z));

            acc_av_vector =
acc_total_vector[0];

```

```

        for(start = 16;
start > 0; start--
){
            accelrometer
vectors.
            acc_total_vector
[start] =
acc_total_vector[start -
1];
            acc_av_vector +=
acc_total_vector[start];
        }

        acc_av_vector /=
17;

        if(vibration_count
er < 20){
            vibration_counter
++;
            vibration_total_
result +=
abs(acc_total_vector[0] -
acc_av_vector);
        }
        else{
            vibration_counte
r = 0;
            Serial.println(v
ibration_total_result/50);

            vibration_total_
result =
0;
        }
    }
    if(data == 'a'){
        if(cal_int != 2000){
            Serial.print("Calibr
ating the gyro");

```



```

        for (cal_int = 0;
cal_int < 2000 ; cal_int
++){
    if(cal_int % 125 == 0){
        digitalWrite(12,
!digitalRead(12));
        Serial.print("."
);
    }
    gyro_signalen();

    gyro_axis_cal[1]
+=
gyro_axis[1];

    gyro_axis_cal[2]
+=
gyro_axis[2];

    gyro_axis_cal[3]
+= gyro_axis[3];
    PORTD |=
B11110000;
    delayMicroseconds(
1000);
    PORTD &=
B00001111;
    delay(3);

}
Serial.println(".");
gyro_axis_cal[1] /=
2000;
gyro_axis_cal[2] /= 2000;
gyro_axis_cal[3] /=
2000;
}
else{
    PORTD |= B11110000;
    delayMicroseconds(10
00);
    PORTD &=
B00001111;
    gyro_signalen();

```

```

        angle_pitch +=
gyro_pitch *
0.0000611;
        angle_roll +=
gyro_roll *
0.0000611;
        angle_pitch -=
angle_roll * sin(gyro_yaw
*
0.000001066);
        angle_roll +=
angle_pitch * sin(gyro_yaw
*
0.000001066);
        acc_total_vector[0]
=
sqrt((acc_x*acc_x)+(acc_y*
acc_y)+(acc_z*acc_z));
        angle_pitch_acc =
asin((float)acc_y/acc_tota
l_vector[0])*
        angle_roll_acc =
asin((float)acc_x/acc_tota
l_vector[0])* -
57.296;

        if(!first_angle){
            angle_pitch =
angle_pitch_acc;

            angle_roll =
angle_roll_acc;

            first_angle =
true;
        }
        else{
            angle_pitch =
angle_pitch * 0.9996 +
angle_pitch_acc *
0.0004;
            angle_roll =
angle_roll * 0.9996 +
angle_roll_acc *
0.0004;
        }

```

```

        if(loop_counter ==
0)Serial.print("Pitch: ");
        if(loop_counter ==
1)Serial.print(angle_pitch
,0);
        if(loop_counter ==
2)Serial.print(" Roll: ");
        if(loop_counter ==
3)Serial.print(angle_roll
,0);
        if(loop_counter ==
4)Serial.print(" Yaw: ");
        if(loop_counter ==
5)Serial.println(gyro_yaw
/ 65.5 ,0);

        loop_counter ++;
        if(loop_counter ==
60)loop_counter = 0;
    }
}

1=====
=====
    if(PINB &
B00000001){

        if(last_channel_1 ==
0){
            last_channel_1 =
1;
            timer_1 =
current_time;
        }
        else if(last_channel_1
== 1){
            last_channel_1 =
0;
            receiver_input[1] =
current_time - timer_1;
        }

```



```

    if(PINB & B00000010
){
    //Is input 9
    high?
    if(last_channel_2 ==
0){
        last_channel_2 =
1;
        timer_2 =
current_time;
    }
    else if(last_channel_2
== 1){
        last_channel_2 =
0;
        receiver_input[2] =
current_time -
timer_2;          //
Channel 2 is current_time
- timer_2.
    }
    if(PINB & B00000100 ){
        if(last_channel_3 ==
0){
            last_channel_3 =
1;
            timer_3 =
current_time;
            //Set
timer_3 to current_time.
        }
    }
    else if(last_channel_3
== 1){
        last_channel_3 =
0;
        receiver_input[3] =
current_time -
timer_3;          //
Channel 3 is current_time
- timer_3.
    }
    if(last_channel_4 ==
0){
        last_channel_4 =
1;
        timer_4 =
current_time;
    }
    else if(last_channel_4
==
1){
        last_channel_4 =
0;
        receiver_input[4] =
current_time -
timer_4;
    }
}

void wait_for_receiver(){
    byte zero = 0;
    while(zero <
15){
        if(receiver_input[1] <
2100 && receiver_input[1]
> 900)zero |=
0b00000001;
        if(receiver_input[2] <
2100 && receiver_input[2]
> 900)zero |= 0b00000010;
        if(receiver_input[3] <
2100 && receiver_input[3]
> 900)zero |= 0b00000100;
        if(receiver_input[4] <
2100 && receiver_input[4]
> 900)zero |= 0b00001000;
        delay(500);
    }
}

int
convert_receiver_channel(b
yte function){
    byte channel, reverse;
    int difference;

    channel =
eeprom_data[function + 23]
&
0b00000111;

    if(eeprom_data[function
+ 23] & 0b10000000)reverse
= 1;
    else reverse =
0;

    actual =
receiver_input[channel];

    low =
(eeprom_data[channel * 2 +
15] << 8) |
eeprom_data[channel * 2 +
14];
    center =
(eeprom_data[channel * 2 -
1] << 8) |
eeprom_data[channel * 2 -
2];
    high =
(eeprom_data[channel * 2 +
7] << 8) |
eeprom_data[channel * 2 +
6];

    if(actual < center){
        if(actual < low)actual
= low;
        difference =
((long)(center - actual) *
(long)500)
        if(reverse == 1)return
1500 + difference;
        else return 1500 -
difference;
    }
    else if(actual >
center){
        if(actual >
high)actual = high;
        difference =
((long)(actual - center) *
(long)500) / (high -
center);
    }
}

```

```

        if(reverse == 1)return
1500 - difference;
        else return 1500 +
difference;
    }
    else return 1500;
}

```

```

void print_signals(){
    Serial.print("Start:");
    Serial.print(start);

```

```

    Serial.print(" Roll:");
    if(receiver_input_channe
l_1 - 1480 <
0)Serial.print("<<<");
    else
if(receiver_input_channel_
1 - 1520 >
0)Serial.print(">>>");
    else Serial.print("-+-
");
    Serial.print(receiver_in
put_channel_1);

```

```

    Serial.print(" Pitch:")
;
    if(receiver_input_channe
l_2 - 1480 <
0)Serial.print("^^^");
    else
if(receiver_input_channel_
2 - 1520 >
0)Serial.print("vvv");
    else Serial.print("-+-
");
    Serial.print(receiver_in
put_channel_2);

```

```

    Serial.print(" Throttle
:");
    if(receiver_input_channe
l_3 - 1480 <
0)Serial.print("vvv");
    else
if(receiver_input_channel_

```

```

3 - 1520 >
0)Serial.print("^^^");
    else Serial.print("-+-
");
    Serial.print(receiver_in
put_channel_3);

```

```

    Serial.print(" Yaw:");
    if(receiver_input_channe
l_4 - 1480 <
0)Serial.print("<<<");
    else
if(receiver_input_channel_
4 - 1520 >
0)Serial.print(">>>");
    else Serial.print("-+-
");
    Serial.println(receiver_
input_channel_4);
}

```

```

void esc_pulse_output(){
    zero_timer = micros();
    PORTD |= B11110000;
    timer_channel_1 = esc_1
+ zero_timer;
    timer_channel_2 = esc_2
+ zero_timer;
    timer_channel_3 = esc_3
+
zero_timer;
    timer_channel_4 = esc_4
+
zero_timer;

```

```

    while(PORTD >=
16){
        esc_loop_timer =
micros();
        if(timer_channel_1 <=
esc_loop_timer)PORTD &=
B11101111;
        if(timer_channel_2 <=
esc_loop_timer)PORTD &=
B11011111;

```

```

        if(timer_channel_3 <=
esc_loop_timer)PORTD &=
B10111111;
        if(timer_channel_4 <=
esc_loop_timer)PORTD &=
B01111111;
    }
}

```

```

void set_gyro_registers(){
    //Setup the MPU-6050
    if(eeprom_data[31] ==
1){
        Wire.beginTransmission
(gyro_address);
        Wire.write(0x6B);
        Wire.write(0x00);
        Wire.endTransmission()
;

```

```

        Wire.beginTransmission
(gyro_address);
        Wire.write(0x1B);
        Wire.write(0x08);
        Wire.endTransmission()
;

```

```

        Wire.beginTransmission
(gyro_address);
        Wire.write(0x1C);
        Wire.write(0x10);
        Wire.endTransmission()
;

```

```

        Wire.beginTransmission
(gyro_address);
        Wire.write(0x1B);
        Wire.endTransmission()
;
        Wire.requestFrom(gyro_
address, 1);
        while(Wire.available()
< 1);
        if(Wire.read() !=
0x08){

```

```

        digitalWrite(12,HIGH
    );

    while(1)delay(10);
    }

    Wire.beginTransmission
(gyro_address);
    Wire.write(0x1A);
    Wire.write(0x03);
    Wire.endTransmission()
;

    }
}

void gyro_signalen(){
    if(eeprom_data[31] ==
1){
        Wire.beginTransmission
(gyro_address);
        Wire.write(0x3B);
        Wire.endTransmission()
;
        Wire.requestFrom(gyro_
address,14);
        while(Wire.available()
< 14);
        acc_axis[1] =
Wire.read()<<8|Wire.read()
;
        acc_axis[2] =
Wire.read()<<8|Wire.read()
;

        acc_axis[3] =
Wire.read()<<8|Wire.read()
;
        temperature =
Wire.read()<<8|Wire.read()
;
        gyro_axis[1] =
Wire.read()<<8|Wire.read()
;
        gyro_axis[2] =
Wire.read()<<8|Wire.read()
;
        gyro_axis[3] =
Wire.read()<<8|Wire.read()
;
    }

    if(cal_int == 2000){
        gyro_axis[1] -=
gyro_axis_cal[1];
        gyro_axis[2] -=
gyro_axis_cal[2];
        gyro_axis[3] -=
gyro_axis_cal[3];
    }
    gyro_roll =
gyro_axis[eeprom_data[28]
& 0b00000011];
    if(eeprom_data[28] &
0b10000000)gyro_roll *= -
1;
    gyro_pitch =
gyro_axis[eeprom_data[29]
& 0b00000011];
    if(eeprom_data[29] &
0b10000000)gyro_pitch *= -
1;
    gyro_yaw =
gyro_axis[eeprom_data[30]
& 0b00000011];
    if(eeprom_data[30] &
0b10000000)gyro_yaw *= -
1;

    acc_x =
acc_axis[eeprom_data[29] &
0b00000011];
    if(eeprom_data[29] &
0b10000000)acc_x *= -
1;
    acc_y =
acc_axis[eeprom_data[28] &
0b00000011];
    if(eeprom_data[28] &
0b10000000)acc_y *= -
1;
    acc_z =
acc_axis[eeprom_data[30] &
0b00000011];
    if(eeprom_data[30] &
0b10000000)acc_z *= -
1;
}

```

4.3 Flight Controller

```

#include
<Wire.h>

#include
<EEPROM.h>

float pid_p_gain_roll =
1.3;

float pid_i_gain_roll =
0.04;
float pid_d_gain_roll =
18.0;
int pid_max_roll =
400;

float pid_p_gain_pitch =
pid_p_gain_roll;
float pid_i_gain_pitch =
pid_i_gain_roll;
float pid_d_gain_pitch =
pid_d_gain_roll;
int pid_max_pitch =
pid_max_roll;

```

```

float pid_p_gain_yaw =
4.0;
float pid_i_gain_yaw =
0.02;
float pid_d_gain_yaw =
0.0;
int pid_max_yaw =
400;

boolean auto_level =
true;
byte last_channel_1,
last_channel_2,
last_channel_3,
last_channel_4;
byte eeprom_data[36];
byte highByte, lowByte;
volatile int
receiver_input_channel_1
,
receiver_input_channel_2
,
receiver_input_channel_3
,
receiver_input_channel_4
;
int counter_channel_1,
counter_channel_2,
counter_channel_3,
counter_channel_4,
loop_counter;
int esc_1, esc_2, esc_3,
esc_4;
int throttle,
battery_voltage;
int cal_int, start,
gyro_address;
int receiver_input[5];
int temperature;
int acc_axis[4],
gyro_axis[4];
float roll_level_adjust,
pitch_level_adjust;

```

```

long acc_x, acc_y,
acc_z, acc_total_vector;
unsigned long
timer_channel_1,
timer_channel_2,
timer_channel_3,
timer_channel_4,
esc_timer,
esc_loop_timer;
unsigned long timer_1,
timer_2, timer_3,
timer_4, current_time;
unsigned long
loop_timer;
double gyro_pitch,
gyro_roll, gyro_yaw;
double gyro_axis_cal[4];
float pid_error_temp;
float pid_i_mem_roll,
pid_roll_setpoint,
gyro_roll_input,
pid_output_roll,
pid_last_roll_d_error;
float pid_i_mem_pitch,
pid_pitch_setpoint,
gyro_pitch_input,
pid_output_pitch,
pid_last_pitch_d_error;
float pid_i_mem_yaw,
pid_yaw_setpoint,
gyro_yaw_input,
pid_output_yaw,
pid_last_yaw_d_error;
float angle_roll_acc,
angle_pitch_acc,
angle_pitch, angle_roll;
boolean gyro_angles_set;
void setup(){
    for(start = 0; start
<= 35;
start++)eeprom_data[start] = EEPROM.read(start);
    start =
0;

```

```

    gyro_address =
eeprom_data[32];
    Wire.begin();
    TWBR =
12;
    DDRD |= B11110000;
    DDRB |=
B00110000;
    digitalWrite(12,HIGH);
    while(eeprom_data[33]
!= 'J' ||
eeprom_data[34] != 'M'
|| eeprom_data[35] !=
'B')delay(10);
    if(eeprom_data[31] ==
2 || eeprom_data[31] ==
3)delay(10);
    set_gyro_registers();
    for (cal_int = 0;
cal_int < 1250 ; cal_int
++){
        PORTD |=
B11110000;
        delayMicroseconds(10
00);
        PORTD &=
B00001111;
        delayMicroseconds(30
00);
    }
    for (cal_int = 0;
cal_int < 2000 ; cal_int
++){
        if(cal_int % 15 ==
0)digitalWrite(12,
!digitalRead(12));
        gyro_signalen();
        gyro_axis_cal[1] +=
gyro_axis[1];
        gyro_axis_cal[2] +=
gyro_axis[2];
        gyro_axis_cal[3] +=
gyro_axis[3];
        PORTD |=
B11110000;

```

```

    delayMicroseconds(10
00);

    PORTD &=
B00001111;

    delay(3);

}
gyro_axis_cal[1] /=
2000;
gyro_axis_cal[2] /=
2000;
gyro_axis_cal[3] /=
2000;

PCICR |= (1 <<
PCIE0);
PCMSK0 |= (1 <<
PCINT0);
PCMSK0 |= (1 <<
PCINT1);
PCMSK0 |= (1 <<
PCINT2);
PCMSK0 |= (1 <<
PCINT3);
while(receiver_input_c
hannel_3 < 990 ||
receiver_input_channel_3
> 1020 ||
receiver_input_channel_4
< 1400){
    receiver_input_chann
el_3 =
convert_receiver_channel
(3);
    receiver_input_chann
el_4 =
convert_receiver_channel
(4);
    start ++;
    PORTD |=
B11110000;
    delayMicroseconds(10
00);

```

```

    PORTD &=
B00001111;
    delay(3);
    if(start ==
125){
        digitalWrite(12,
!digitalRead(12));
        start =
0;
    }
    start = 0;
    battery_voltage =
(analogRead(0) + 65) *
1.2317;
    loop_timer =
micros();
    digitalWrite(12,LOW);
}
void loop(){
    gyro_roll_input =
(gyro_roll_input * 0.7)
+ ((gyro_roll / 65.5) *
0.3);
    gyro_pitch_input =
(gyro_pitch_input * 0.7)
+ ((gyro_pitch / 65.5) *
0.3);
    gyro_yaw_input =
(gyro_yaw_input * 0.7) +
((gyro_yaw / 65.5) *
0.3);
    angle_pitch +=
gyro_pitch * 0.0000611;
    angle_roll +=
gyro_roll *
0.0000611;
    angle_pitch -=
angle_roll *
sin(gyro_yaw *
0.000001066);
    angle_roll +=
angle_pitch *
sin(gyro_yaw *
0.000001066);

```

```

    acc_total_vector =
sqrt((acc_x*acc_x)+(acc_
y*acc_y)+(acc_z*acc_z));

    if(abs(acc_y) <
acc_total_vector){
        angle_pitch_acc =
asin((float)acc_y/acc_to
tal_vector)* 57.296;
    }
    if(abs(acc_x) <
acc_total_vector){
        angle_roll_acc =
asin((float)acc_x/acc_to
tal_vector)* -57.296;
    }
    angle_pitch_acc -=
0.0;
    angle_roll_acc -= 0.0;
    angle_pitch =
angle_pitch * 0.9996 +
angle_pitch_acc *
0.0004;
    angle_roll =
angle_roll * 0.9996 +
angle_roll_acc * 0.0004;
    pitch_level_adjust =
angle_pitch * 15;
    roll_level_adjust =
angle_roll *
15;

    if(!auto_level){
        pitch_level_adjust =
0;
        roll_level_adjust =
0;
    }
    if(receiver_input_chan
nel_3 < 1050 &&
receiver_input_channel_4
< 1050)start = 1;
    if(start == 1 &&
receiver_input_channel_3
< 1050 &&

```

```

receiver_input_channel_4
> 1450){
    start = 2;

    angle_pitch =
angle_pitch_acc;
    angle_roll =
angle_roll_acc;
    gyro_angles_set =
true;
    pid_i_mem_roll = 0;
    pid_last_roll_d_error = 0;
    pid_i_mem_pitch = 0;
    pid_last_pitch_d_error = 0;
    pid_i_mem_yaw = 0;
    pid_last_yaw_d_error = 0;
}
if(start == 2 &&
receiver_input_channel_3
< 1050 &&
receiver_input_channel_4
> 1950)start = 0;
    pid_roll_setpoint = 0;
    if(receiver_input_channel_1
> 1508)pid_roll_setpoint =
receiver_input_channel_1
- 1508;
    else
if(receiver_input_channel_1
< 1492)pid_roll_setpoint =
receiver_input_channel_1
- 1492;

    pid_roll_setpoint -=
roll_level_adjust;
    pid_roll_setpoint /=
3.0;
    pid_pitch_setpoint =
0;

```

```

    if(receiver_input_channel_2
> 1508)pid_pitch_setpoint =
receiver_input_channel_2
- 1508;
    else
if(receiver_input_channel_2
< 1492)pid_pitch_setpoint =
receiver_input_channel_2
- 1492;

    pid_pitch_setpoint -=
pitch_level_adjust;
    pid_pitch_setpoint /=
3.0;
    pid_yaw_setpoint = 0;

    if(receiver_input_channel_3
> 1050){
        if(receiver_input_channel_4
> 1508)pid_yaw_setpoint =
(receiver_input_channel_4
- 1508)/3.0;
        else
if(receiver_input_channel_4
< 1492)pid_yaw_setpoint =
(receiver_input_channel_4
- 1492)/3.0;
    }

    calculate_pid();
    battery_voltage =
battery_voltage * 0.92 +
(analogRead(0) + 65) *
0.09853;
    if(battery_voltage <
1000 && battery_voltage
> 600)digitalWrite(12,
HIGH);

```

```

    throttle =
receiver_input_channel_3
;

    //We need
the throttle signal as a
base signal.

    if (start ==
2){
        if (throttle > 1800)
throttle = 1800;
        esc_1 = throttle -
pid_output_pitch +
pid_output_roll -
pid_output_yaw;
        esc_2 = throttle +
pid_output_pitch +
pid_output_roll +
pid_output_yaw;
        esc_3 = throttle +
pid_output_pitch -
pid_output_roll -
pid_output_yaw;
        esc_4 = throttle -
pid_output_pitch -
pid_output_roll +
pid_output_yaw;

        if (battery_voltage
< 1240 &&
battery_voltage >
800){
            esc_1 += esc_1 *
((1240 -
battery_voltage)/(float)
3500);
            esc_2 += esc_2 *
((1240 -
battery_voltage)/(float)
3500);
            esc_3 += esc_3 *
((1240 -
battery_voltage)/(float)
3500);
            esc_4 += esc_4 *
((1240 -

```

```

battery_voltage)/(float)
3500);
}

```

```

    if (esc_1 < 1100)
esc_1 = 1100;
    if (esc_2 < 1100)
esc_2 = 1100;
    if (esc_3 < 1100)
esc_3 =
1100;
    if (esc_4 < 1100)
esc_4 =
1100;

```

```

    if(esc_1 >
2000)esc_1 = 2000;
    if(esc_2 >
2000)esc_2 =
2000;
    if(esc_3 >
2000)esc_3 =
2000;
    if(esc_4 >
2000)esc_4 =
2000;
}

```

```

    else{
        esc_1 =
1000;
        esc_2 =
1000;
        esc_3 =
1000;
        esc_4 =
1000;
    }
    if(micros() -
loop_timer >
4050)digitalWrite(12,
HIGH);

```

```

    while(micros() -
loop_timer <

```

```

4000);

```

```

    loop_timer =
micros();

```

```

    PORTD |=
B11110000;
    timer_channel_1 =
esc_1 +
loop_timer;
    timer_channel_2 =
esc_2 +
loop_timer;
    timer_channel_3 =
esc_3 +
loop_timer;
    timer_channel_4 =
esc_4 +
loop_timer;

```

```

    gyro_signalen();

```

```

    while(PORTD >= 16){
        esc_loop_timer =
micros();
        if(timer_channel_1
<= esc_loop_timer)PORTD
&= B11101111;
        if(timer_channel_2
<= esc_loop_timer)PORTD
&=
B11011111;
        if(timer_channel_3
<= esc_loop_timer)PORTD
&=
B10111111;
        if(timer_channel_4
<= esc_loop_timer)PORTD
&=
B01111111;
    }
}
ISR(PCINT0_vect){
    current_time =
micros();

```

```

    if(PINB &
B00000001){
        if(last_channel_1 ==
0){
            last_channel_1 =
1;
            timer_1 =
current_time;
        }
        else if(last_channel_1
== 1){
            last_channel_1 =
0;
            receiver_input[1] =
current_time -
timer_1;
        }
        if(PINB & B00000010 ){
            if(last_channel_2 ==
0){
                last_channel_2 =
1;
                timer_2 =
current_time;
            }
            else if(last_channel_2
== 1){
                last_channel_2 =
0;
                receiver_input[2] =
current_time - timer_2;
            }
            if(PINB & B00000100 ){
                if(last_channel_3 ==
0){
                    last_channel_3 =
1;
                    timer_3 =
current_time;
                }
                else if(last_channel_3
==
1){

```

```

        last_channel_3 = 0;
        receiver_input[3] =
current_time -
timer_3;

    }
    if(PINB & B00001000 ){
        if(last_channel_4 ==
0){
            last_channel_4 =
1;
            timer_4 =
current_time;
        }
        else if(last_channel_4
==
1){
            last_channel_4 = 0;
            receiver_input[4] =
current_time - timer_4;
        }
    }
    void gyro_signalen(){

        if(eeprom_data[31] ==
1){
            Wire.beginTransmissi
on(gyro_address);

            Wire.write(0x3B);

            Wire.endTransmission
();
            Wire.requestFrom(gyro_
o_address,14);

            receiver_input_chann
el_1 =
convert_receiver_channel
(1);
            receiver_input_chann
el_2 =
convert_receiver_channel
(2);

            receiver_input_chann
el_3 =
convert_receiver_channel
(3);
            receiver_input_chann
el_4 =
convert_receiver_channel
(4);

            while(Wire.available
() < 14);
            acc_axis[1] =
Wire.read()<<8|Wire.read
();
            acc_axis[2] =
Wire.read()<<8|Wire.read
();
            acc_axis[3] =
Wire.read()<<8|Wire.read
();
            temperature =
Wire.read()<<8|Wire.read
();
            gyro_axis[1] =
Wire.read()<<8|Wire.read
();
            gyro_axis[2] =
Wire.read()<<8|Wire.read
();
            gyro_axis[3] =
Wire.read()<<8|Wire.read
();
        }

        if(cal_int == 2000){
            gyro_axis[1] -=
gyro_axis_cal[1];
            gyro_axis[2] -=
gyro_axis_cal[2];
            gyro_axis[3] -=
gyro_axis_cal[3];
        }
        gyro_roll =
gyro_axis[eeprom_data[28
] &
0b00000011];

        if(eeprom_data[28] &
0b10000000)gyro_roll *= -
1;
        gyro_pitch =
gyro_axis[eeprom_data[29
] &
0b00000011];
        if(eeprom_data[29] &
0b10000000)gyro_pitch *= -
1;
        gyro_yaw =
gyro_axis[eeprom_data[30
] &
0b00000011];
        if(eeprom_data[30] &
0b10000000)gyro_yaw *= -
1;

        acc_x =
acc_axis[eeprom_data[29]
& 0b00000011];
        if(eeprom_data[29] &
0b10000000)acc_x *= -
1;
        acc_y =
acc_axis[eeprom_data[28]
& 0b00000011];
        if(eeprom_data[28] &
0b10000000)acc_y *= -
1;
        acc_z =
acc_axis[eeprom_data[30]
& 0b00000011];
        if(eeprom_data[30] &
0b10000000)acc_z *= -
1;
    }
    void calculate_pid(){
        //Roll calculations
        pid_error_temp =
gyro_roll_input -
pid_roll_setpoint;

```



```

    pid_i_mem_roll +=
    pid_i_gain_roll *
    pid_error_temp;
    if(pid_i_mem_roll >
    pid_max_roll)pid_i_mem_r
    oll = pid_max_roll;
    else if(pid_i_mem_roll
    < pid_max_roll * -
    1)pid_i_mem_roll =
    pid_max_roll * -1;

    pid_output_roll =
    pid_p_gain_roll *
    pid_error_temp +
    pid_i_mem_roll +
    pid_d_gain_roll *
    (pid_error_temp -
    pid_last_roll_d_error);
    if(pid_output_roll >
    pid_max_roll)pid_output_
    roll = pid_max_roll;
    else
    if(pid_output_roll <
    pid_max_roll * -
    1)pid_output_roll =
    pid_max_roll * -1;

    pid_last_roll_d_error
    = pid_error_temp;
    pid_error_temp =
    gyro_pitch_input -
    pid_pitch_setpoint;
    pid_i_mem_pitch +=
    pid_i_gain_pitch *
    pid_error_temp;
    if(pid_i_mem_pitch >
    pid_max_pitch)pid_i_mem_
    pitch = pid_max_pitch;
    else
    if(pid_i_mem_pitch <
    pid_max_pitch * -
    1)pid_i_mem_pitch =
    pid_max_pitch * -1;

    pid_output_pitch =
    pid_p_gain_pitch *

```

```

    pid_error_temp +
    pid_i_mem_pitch +
    pid_d_gain_pitch *
    (pid_error_temp -
    pid_last_pitch_d_error);
    if(pid_output_pitch >
    pid_max_pitch)pid_output
    _pitch = pid_max_pitch;
    else
    if(pid_output_pitch <
    pid_max_pitch * -
    1)pid_output_pitch =
    pid_max_pitch * -1;

    pid_last_pitch_d_error
    = pid_error_temp;
    pid_error_temp =
    gyro_yaw_input -
    pid_yaw_setpoint;
    pid_i_mem_yaw +=
    pid_i_gain_yaw *
    pid_error_temp;
    if(pid_i_mem_yaw >
    pid_max_yaw)pid_i_mem_ya
    w = pid_max_yaw;
    else if(pid_i_mem_yaw
    < pid_max_yaw * -
    1)pid_i_mem_yaw =
    pid_max_yaw * -1;

    pid_output_yaw =
    pid_p_gain_yaw *
    pid_error_temp +
    pid_i_mem_yaw +
    pid_d_gain_yaw *
    (pid_error_temp -
    pid_last_yaw_d_error);
    if(pid_output_yaw >
    pid_max_yaw)pid_output_y
    aw = pid_max_yaw;
    else if(pid_output_yaw
    < pid_max_yaw * -
    1)pid_output_yaw =
    pid_max_yaw * -1;

```

```

    pid_last_yaw_d_error =
    pid_error_temp;
}
int
convert_receiver_channel
(byte function){
    byte channel,
    reverse;
    int low, center, high,
    actual;
    int difference;
    channel =
    eeprom_data[function +
    23] & 0b00000111;
    if(eeprom_data[functio
    n + 23] &
    0b10000000)reverse =
    1;
    else reverse =
    0;
    actual =
    receiver_input[channel];
    low =
    (eeprom_data[channel * 2
    + 15] << 8) |
    eeprom_data[channel * 2
    + 14];
    center =
    (eeprom_data[channel * 2
    - 1] << 8) |
    eeprom_data[channel * 2
    - 2];
    high =
    (eeprom_data[channel * 2
    + 7] << 8) |
    eeprom_data[channel * 2
    + 6];
    if(actual <
    center){
        if(actual <
        low)actual =
        low;
        difference =
        ((long)(center - actual)
        * (long)500) / (center -
        low);

```

```

        if(reverse ==
1)return 1500 +
difference;
        else return 1500 -
difference;
    }
    else if(actual >
center){
        if(actual >
high)actual =
high;
        difference =
((long)(actual - center)
* (long)500) / (high -
center);
        if(reverse ==
1)return 1500 -
difference;
        else return 1500 +
difference;
    }
    else return 1500;
}

void
set_gyro_registers(){
    if(eeprom_data[31] ==
1){
        Wire.beginTransmissi
on(gyro_address);
        Wire.write(0x6B);
        Wire.write(0x00);
        Wire.endTransmission
();

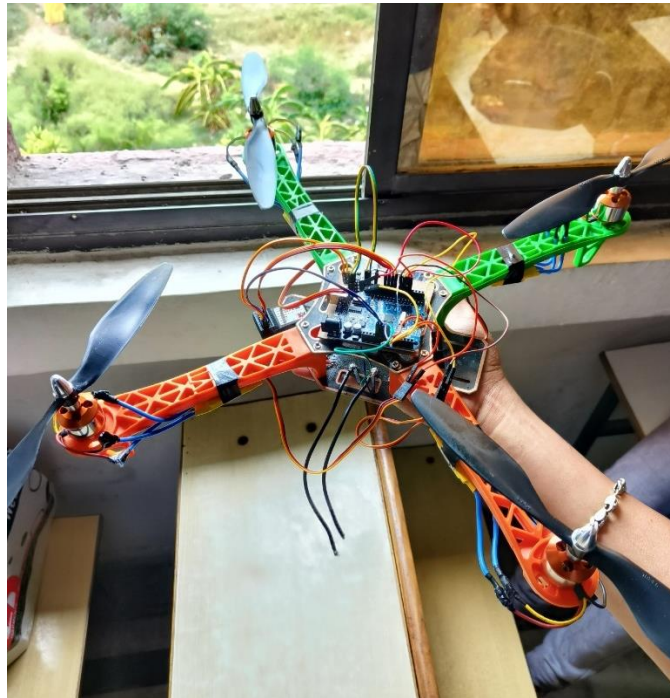
        Wire.beginTransmissi
on(gyro_address);
        Wire.write(0x1B);
        Wire.write(0x08);
        Wire.endTransmission
();

        Wire.beginTransmissi
on(gyro_address);
        Wire.write(0x1C);
        Wire.write(0x10);
        Wire.endTransmission
();
        Wire.beginTransmissi
on(gyro_address);
        Wire.write(0x1B);
        Wire.endTransmission
();
        Wire.requestFrom(gyr
o_address,
1);
        while(Wire.available
()) <
1);
        if(Wire.read() !=
0x08){
            digitalWrite(12,HI
GH);
            while(1)delay(10
)
        }

        Wire.beginTransmissi
on(gyro_address);
        Wire.write(0x1A);
        Wire.write(0x03);
        Wire.endTransmission
();
    }
}

```

4.4 Screenshot





CHAPTER 5: FUTURE OF THE PROJECT

The future of a project for remote-controlled drones without cameras holds several potential advancements and opportunities. Here are some possibilities:

1. **Enhanced Flight Performance:** Future iterations of remote-controlled drones without cameras can focus on improving flight performance. This may include advancements in motor efficiency, battery technology, and aerodynamic design to increase speed, agility, and endurance.
2. **Intelligent Autonomy:** Integration of advanced artificial intelligence (AI) and machine learning algorithms can enable drones to have intelligent autonomy. They can navigate complex environments, avoid obstacles, and perform tasks autonomously without direct user control. This opens up possibilities for applications in surveillance, search and rescue, and industrial inspections.
3. **Extended Range and Connectivity:** Improvements in wireless communication technologies can provide extended control range between the transmitter and the drone. This enables pilots to explore larger areas and fly in more challenging environments. Additionally, enhanced connectivity options can enable real-time data transmission and remote monitoring of the drone's status and telemetry.
4. **Advanced Sensor Integration:** Future projects can incorporate advanced sensor technologies into drones without cameras. This includes LiDAR (Light Detection and Ranging) sensors, infrared sensors, or ultrasonic sensors to enhance navigation, obstacle detection, and environmental awareness.
5. **Integration with Augmented Reality (AR):** AR can be leveraged to enhance the piloting experience for remote-controlled drones without cameras. Pilots can wear AR goggles or use smartphone applications to overlay flight data, navigation information, or virtual obstacles onto the real-world view, providing an immersive and informative flight experience.
6. **Swarm Capabilities:** Future projects may explore the development of swarm capabilities for drones without cameras. By coordinating and communicating with other drones, they can collaborate on tasks, synchronize movements, and work together to achieve complex objectives. This can be useful in various applications, including mapping, monitoring, or coordinated aerial displays.

5.1 PROPOSED SYSTEM

A proposed system for a remote controller drone without camera would consist of the following components:

- A drone: This is the flying vehicle that will be controlled by the remote controller. It would typically be a small, lightweight quadcopter with four rotors.
- A remote controller: This is the device that will be used to control the drone. It would typically have a joystick for controlling the drone's movement, as well as buttons for controlling its speed and other features.
- A battery: The drone and the remote controller would both need to be powered by batteries.
- A transmitter: The remote controller would need to have a transmitter that sends signals to the drone.
- A receiver: The drone would need to have a receiver that receives signals from the remote controller.

CHAPTER 6: CONCLUSION

In conclusion, a remote-controlled drone without a camera offers a simplified and cost-effective approach to drone technology. By eliminating the camera module, these drones prioritize flight performance, maneuverability, and user control. They are suitable for beginners, recreational users, or individuals who do not require aerial photography or video recording capabilities.

The drone typically includes advanced sensors and stabilization mechanisms to ensure stable flight performance and prevent collisions. User control is achieved through a traditional remote controller with intuitive joysticks and buttons, providing precise command over the drone's movements. Voice control is not necessary in this setup, reducing complexity and potential limitations.

Additional features can be incorporated into the drone, such as programmable flight paths, waypoint navigation, and obstacle avoidance algorithms. These features can be accessed through a user-friendly smartphone application that wirelessly communicates with the drone's controller, enhancing the drone's capabilities and offering a customizable flight experience.

Overall, a remote-controlled drone without a camera provides a streamlined and accessible drone experience. It caters to individuals who prioritize simplicity, affordability, and enhanced user control, making drone technology more approachable and enjoyable for various recreational and non-photography-related applications.

References

1. https://www.google.com/search?q=JUMPER+CABLES&source=lmns&bih=625&biw=1366&hl=en&sa=X&ved=2ahUKEwjUs8285ND8AhX5F7cAHWn5DQEQ_AUoAHoECAEQAA
2. <https://www.wiltronics.com.au/wiltronics-knowledge-base/what-are-jumper-wires/>
3. <https://www.flyrobo.in/combo-3-type-jumper-cables-f-f-f-m-m-m>
4. https://www.google.com/search?q=types+of+jumper+cables&source=lmns&bih=625&biw=1366&hl=en&sa=X&ved=2ahUKEwji6caG5dD8AhXYj9gFHfPYAYQQ_AUoAHoECAEQAA
5. https://www.google.com/search?q=wires&tbm=isch&ved=2ahUKEwjJ_6rN5tD8AhW80HMBHQRtCx4Q2-cCegQIABAA&oq=wires&gs_lcp=CgNpbWcQAzIECAAQQzIHCAAQsQMQQzIECAAQQzIECAAQQzIICAAQgAAQsQMyBAGAEEMyBAGAEEMyBQgAEIAEMgUIABCABDIFCAAQgARQ9A1Y1xhgCJoAHAAeACAAaACiAHmBpIBBTauNS4xmAEAoAEBqgELZ3dzLXdpei1pbWfAAQE&sclient=img&ei=S7vHY4muNryhz7sPhNqt8AE&bih=625&biw=1366&hl=en
6. https://en.wikipedia.org/wiki/Windows_7
7. <https://steemit.com/utopian-io/@drencolha/mfrc522-rfid-reader-arduino-library-setup-and-functions-usage-shown-with-an-example-project-tutorial>
8. <https://www.ucl.ac.uk/short-courses/search-courses/internet-things-iot-introduction-understanding-and-designing-iot->
9. <https://www.google.com/search?q=learning+outcomes+of+iot&oq=learning+outcomes+of+iot&aqs=chrome..69i57j0i22i30l2j0i390l5.12248j0j7&sourceid=chrome&ie=UTF-8>
10. <https://www.google.com/search?q=technology+used+in+iot+projects&oq=technology+used+in+iot+pro&aqs=chrome..69i57j33i160l2j33i22i29i30.9103j0j7&sourceid=chrome&ie=UTF-8>
11. <https://www.geeksforgeeks.org/architecture-of-internet-of-things-iot/>
12. <https://www.unmannedsystemstechnology.com/expo/drone-motors/>

- 13.**<https://www.advancecomsolutions.com/blog/drone-applications-in-the-future/>
- 14.**<https://yourtechdiet.com/blogs/drone-technology-and-its-future-uses-and-applications/>
- 15.**<https://www.google.com/search?q=single+rotor+drone&oq=single+rotor&aqs=chrome..69j69i57j0i512l8.8085j0j7&sourceid=chrome&ie=UTF-8>
- 16.** <https://www.analyticssteps.com/blogs/drone-technology-working-and-uses>