**République Algérienne Démocratique et Populaire**

**Ministère de l’Enseignement Supérieur et de la Recherche Scientifique**

**Université des Sciences et de la Technologie Houari Boumediene**



**Faculté de Génie Électrique**

**Domaine Sciences et Technologie**

**Filière Électronique**

**Mémoire de Licence**

**Spécialité Electronique**

**Type de projet :Drone**

**Thème**

**Convertible Drone-Car System**

|  |  |
| --- | --- |
| Présenté par: | Boumetloua Mohamed Rayane |

|  |  |
| --- | --- |
| Encadreurs: | Riad Grioune |

**Année Universitaire 2024-2025**

## Remerciements :

First and foremost, we are grateful to alah, the Almighty, who has given us the strength, wisdom, and patience to complete this task.

We are grateful to Mr. Professor Riad Grioune. In his capacity as our pfc promoter, who has supported and guided us in our work and helped us come up with ways to advance more effectively.   
  
We would like to express our sincere gratitude to the Houari Boumediene University of Sciences and Technology, especially to the academic staff of the electrical engineering faculty, for granting us permission and providing us with a pleasant work environment.   
  
The completion of this memory would not have been possible without the patience, skills, and interventions of everyone along the way who helped and encouraged us to complete this work. It would be difficult for us to list them all. What they found here is a manifestation of our observation.

The following is the structure of this document:   
The first chapter is devoted to the drone generalities.   
The software and hardware we used are described in the second chapter.   
The modeling and conception of our system, together with its programming and experimental tests, are presented in the third chapter.   
  
Finally, a general conclusion marks the completion of our work.

# Index

**Introduction Générale**

* 1.1 Project Goal
* 1.2 General Explanation

**Chapter 1: System Design and Theoretical Background**

* 1.3 System Overview
  + General Drone Introduction
    - Figure 1.2: Drone functioning diagram with flight controller
  + How the Transforming Drone Works
  + How the Drone Operates
* 1.4 Electronic Components
  + Brushless DC Motors
  + Electronic Speed Controllers (ESCs)
  + Flight Controller
  + Power Distribution System
  + Table 1.2: Component specs, including MOSFET V\_DS, ESP32 clock speed, motor kV, PDB voltages
* 1.5 Mechanical Transformation
  + Servo Motors
    - Figure 1.4: Servo arrangement diagram, showing MG955 and MG90 servos with linkages
    - Figure 1.5: Photo of transformation mechanism in flight and road configurations [Request if available]
  + Frame
* 1.6 Control System
  + Web Interface and Communication
  + Control Logic
* 1.8 Conclusion

**Chapter 2: Materials and Components**

* 2.1 Power System Components
  + Brushless DC Motors
  + Electronic Speed Controller (ESC)
    - Functioning
    - Figure 2.0: ESC circuit schematic with ATmega328PU, IR2104S drivers, IRLZ44N MOSFETs, and PDB connections [Request if needed]
    - Figure 2.1: PDB schematic
    - Figure 2.2: ATmega328PU pinout diagram
    - Figure 2.3: IR2104S driver circuit
    - Figure 2.4: IRLZ44N MOSFET
    - Figure 2.11: LM2576-12 buck converter
    - Figure 2.12: LM7805 regulator circuit
  + Power Distribution Board (PDB)
    - LM2576-12 Buck Converter
    - LM7805 Linear Regulator
  + 3S LiPo Battery
* 2.2 Control and Navigation Components
  + ESP32 Microcontroller
    - Functionality and Role
    - Figure 2.5: ESP32 pin diagram
  + MPU6050 Gyroscope/Accelerometer
    - Functionality and Role
    - Figure 2.6: MPU6050 diagram
  + BMP280 Barometer
    - Functionality and Role
    - Figure 2.7: BMP280 diagram
  + Web-Based Control Interface
    - Figure 2.7: Web-based control interface
  + ESP32-CAM
    - Figure 2.10: esp32-cam pinout
* 2.3 Mechanical and Structural Components
  + MG955 Servo Motor
    - Figure 2.8: MG955 servo, linear gear showcase
  + MG90 Servo Motors
    - Figure 2.9: MG90 servo, axial gear showcase
* 2.4 Programmes used

**Chapter 3: Modélisation et la Conception**

* 3.1 ESP32 Flight Controller
  + Figure 3.3: ESP32 flight controller PCB schematic
  + Figure 3.4: ESP32 flight controller PCB layout
* 3.2 Flight Controller Operation
  + Receiving Instructions from the Web Interface
  + Communication with MPU-6050 (Gyroscope and Accelerometer)
  + Communication with BMP280 (Barometer)
  + PID Control
  + Generating PWM Signals for ESCs
  + Transformation Servos
  + Safety Features
  + Data Flow
  + Mode-Specific Behavior
  + Figure 3.2: Flight controller stabilization closed loop
* 3.3 DIY ESC
  + Figure 3.5: DIY ESC PCB schematic
  + Figure 3.6: DIY ESC PCB layout
* 3.4 Power Distribution Board
  + Figure 3.7: Power distribution board PCB schematic
* 3.5 Brushless Motor
  + Figure 3.8: Brushless motor rotor and stator design
  + Figure 3.9: Brushless motor stator pole wiring
* 3.6 Drone Frame and Gears
  + Figure 3.10: Drone frame 3D design
  + Figure 3.11 (1): Drone frame assembly, part 1
  + Figure 3.11 (2): Drone frame assembly, part 2
* 3.7 General Conclusion

**4.1 Bibliography**

# general Introduction:

### 1.1 Project Goal

Through the introduction of a hybrid drone-car that can maneuver through tight places, this project seeks to help with search and rescue missions in buildings with weak structural integrity or in deep woodlands and thick forests where flying is difficult. while also offering a thorough guide for enthusiasts to build an accessible, at-home drone.

### 1.2 General Explanation

The convertible drone-car is a cutting-edge, DIY platform that combines aerial and terrestrial capabilities for specialized jobs like search and rescue and cave mapping, as well as for hobbyist experimentation. It provides strong propulsion for both ground and flight movement thanks to four specially designed brushless DC motors that are managed by an Arduino-based electronic speed controller (ESC) equipped with IRLZ44N MOSFETs and IR2104 drivers. Through the processing of real-time orientation and altitude data, an ESP32 microcontroller coupled with MPU6050 (gyroscope/accelerometer) and BMP280 (barometer) sensors guarantees stable navigation. Reliable operation is supported by a homemade power distribution board (PDB) that uses LM2576-12 and LM7805 regulators to supply steady 12V and 5V from a 3S LiPo battery. Propellers are reoriented for smooth mode changeover by the transformation mechanism, which is powered by MG955 and MG90 servo motors. A web-based interface that is hosted on the ESP32's wireless network,offers intuitive control for flight and ground modes. This modular, cost-effective design empowers Algerian hobbyists and helps them over come the component scarcity and high price in the local market, it also enables precise navigation in confined spaces, detailed across three chapters of this report [2].

# Chapter 1: System Design and Theoretical Background

## 1.3 System Overview:

## General Drone Introduction:

Unmanned aerial vehicles (UAVs), sometimes known as drones, are multipurpose tools used for search and rescue, mapping, aerial photography, and surveillance. To provide accurate operation and stable flight, they combine propulsion, navigation, and control systems. Electronic speed controllers for motor regulation, flight controllers with sensors for stability, brushless motors for thrust, and power systems for energy distribution are important parts. These technologies facilitate both professional and hobbyist applications by allowing drones to operate in a variety of situations, including enclosed places and open skies. The transforming drone, a hybrid system that brings these features to terrestrial modes, is introduced in this section. (Figure 1.2)

### How the Transforming Drone Works:

In flying mode, the changing drone functions as a quadcopter, and in road mode, it functions as a wheeled vehicle. Its smooth integration of mechanical, software, and electronic components enables it to switch modes and operate dependably in a variety of settings. A geared axial and linear transformation mechanism, an advanced ESP32 flight control system, and a unique power distribution board are essential components of its design. This overview describes how these parts cooperate to provide the drone's user-friendly control, reliable operation, and effective mode switching, which makes it adaptable for both aerial and ground-based operations.

### How the Drone Operates:

Controlled via a web interface on a smartphone or computer or via a radio transmeter by slightly adjusting the code, the drone responds to user commands for movement, mode switching, and performance monitoring. In flight mode, it flies like a quadcopter, using four motors to lift, tilt, and rotate. In road mode, it drives like a tank, with the brushless motors powering wheels wheel like propelers for forward, backward, and turning motions. Sensors track orientation, altitude, and battery status, while the flight control system ensures stability and precision. A transformation mechanism reconfigures the drone’s structure for each mode, enabling smooth transitions.

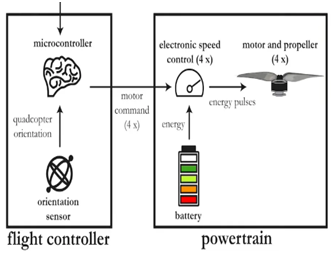
## 1.4 Electronic Components

Drone propulsion, navigation, and control are made possible by the electrical subsystem, which is the foundation of the device's operation. The main parts of most drones are described in this section. These parts include brushless DC motors, electronic speed controllers (ESCs), flight controllers with built-in sensors, and power distribution systems:

**Explanation**:

* **Brushless DC Motors**: These motors, typically rated 800–2000 kV and operating at 7.4–14.8V, power propellers to provide lift and maneuverability. Their high efficiency supports extended flight times, critical for aerial tasks (Table 1.2).
* **Electronic Speed Controllers (ESCs)**: ESCs regulate motor speed using pulse-width modulation (PWM) to ensure smooth and precise control. They manage high currents, enabling responsive throttle and directional adjustments for stable flight.
* **Flight Controller**: A central processing unit, often operating at 100–300 MHz, integrates sensor data and user inputs. Gyroscopes and accelerometers maintain orientation, while barometers support altitude control, ensuring stability in diverse conditions ( Table 1.2).
* **Power Distribution System**: This system distributes battery power (typically 7.4–14.8V LiPo) to all components, using voltage regulators and filters to maintain stable operation and protect electronics (Table 1.2).

**Figures**:

 Figure 1.2:drone functioning diagram

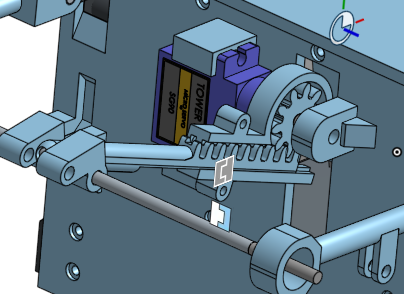
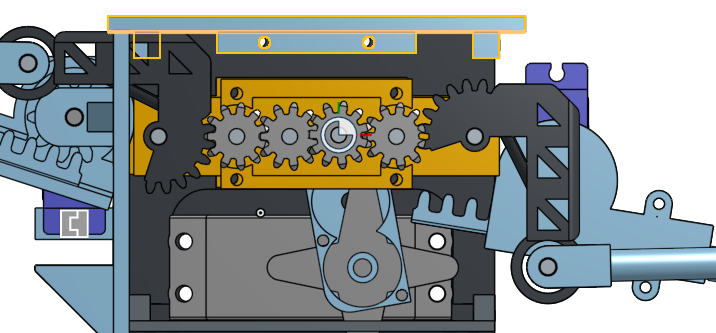
## 1.5 Mechanical Transformation:

The transformation mechanism enables mode switching by reorienting propeller arms, adapting the drone-car for aerial or terrestrial environments. This section covers the servo-driven system, designed for simplicity and reliability [4].

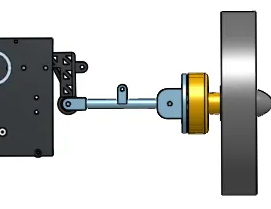
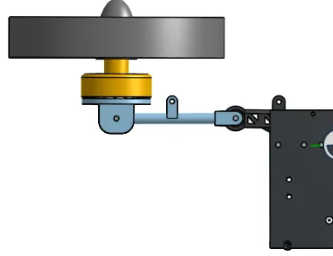
### Explanation :

* **Servo Motors**: An MG955 servo lowers propeller shoulder that are connected to the arms for road mode stability, while two MG90 servos rotate motors connected to the arms from horizontal (road) to vertical (flight) for wheel engagement, responding to flight controller signals (Figure 1.4). The lightweight mechanism, with L shaped linkages to motor axial rod mounts, completes transitions ina few steps, enduring vibrations and impacts (Figure 1.5).
* **Frame**: entirely 3d printabale with parts in PLA and PETG depending on the stress applicabale to them.

### Figures :

* **Figure 1.4**: Servo arrangement diagram, showing MG955 and MG90 servos with linkages.



* **Figure 1.5**: Photo of transformation mechanism in flight and road configurations.

## 1.6 Control System :

The control system enables intuitive operation via a web-based interface, ensuring stable flight and efficient mode switching. This section covers the ESP32-based architecture, leveraging control theory and wireless communication [5].

### Explanation :

* **Web Interface and Communication**: The ESP32’s WiFi (TCP/IP) hosts a web interface with flight mode joysticks (throttle, roll, pitch, yaw) and road mode buttons (forward, backward, steering), built with HTML/CSS/JavaScript. HTTP commands ensure low-latency control, with telemetry (altitude, velocity, battery) sent every 500 ms.
* **Control Logic**: In flight mode, PID algorithms use MPU6050 and BMP280 data for orientation, altitude hold, and yaw control. In road mode, direct motor and servo (MG955, MG90) control enables steering and transformation. Safety features include emergency stop and sensor fallbacks. **Tableau 1.2 :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Component | Brushless  Motor | Brushless  Motor | Brushless  Motor | ESC | ESC | ESC | ESP32  **FV** | ESP32  **FV** | ESP32  **FV** |
| Parameter | Rating | MAX  Current | Op  Voltage | MOSfet  VDS | MAX  Current | Op  Voltage | Clock Speed | Op  Voltage | WiFi Frequency |
| Value | 1000kV | 25A | 11.1V | 55V | 30A | 11.1V | 240 MHz | 5V | 2.4 GHz |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Component | MPU6050 | MPU6050 | MPU6050 | BMP280 | BMP280 | PDB | PDB | PDB |
| Parameter | Gyro  Range | Accel  Range | Op  Voltage | Pressure  Range | Op  Voltage | Input  Voltage | Output  Voltages | MAX  Current |
| Value | ±500 °/s | ±8 g | 3.3V | 300–1100 hPa | 3.3 V | 11.1 V | 12 V, 5 V |  |

# Chapter 2: Materials and Components

## 2.1 Power System Components:

The foundation of the convertible drone-car is electronic flight controlers and power system, which supplies and regulates energy to run motors, regulate electronics, and operate transformation processes. This section describes the functions and material characteristics of the brushless DC motors, custom power distribution board (PDB), Arduino-based electronic speed controller (ESC), and 3S LiPo battery. These parts, which provide strong power delivery for both aerial and terrestrial modes, were picked for their effectiveness, dependability, and suitability for the project's do-it-yourself philosophy. The system's performance and operational stability are greatly impacted by their integration and design [2].

**Explanation**:  
**Brushless DC Motors**: Four high-efficiency motors provide thrust for flight and traction for ground movement. Constructed with lightweight alloys and copper windings, they offer a high thrust-to-weight ratio, critical for lifting ~1–2 kg frame. Their high current draw (up to 20A each) necessitates precise control and power management.

**Electronic Speed Controller (ESC)** (figure2, Figure 2.2, Figure 2.3, Figure 2.4): The ESC, centered on the Arduino ATMega328PU microcontroller, manages motor speed through( IR2101 to IR2104 or TC4427 depending which is disponible)MOSFET drivers and IRLZ44N MOSFETs OR any N type MOSFETS, delivering precise pulse-width modulation (PWM) signals to 3 mosfet drives in a precise cycle . Built on a 1.6 mm FR4 PCB with copper traces for high-current paths (up to 20A per motor), the ESC converts flight controller commands into motor power, impacting efficiency and thermal performance. It integrates with the 11.1V 3S LiPo battery and custom power distribution board (PDB), ensuring reliable operation in both aerial and terrestrial modes. The ESC’s design emphasizes cost-effectiveness and accessibility for hobbyists, making it a pivotal component in the drone-car’s power system [2].  
**Functioning**:  
The ESC employs a sensorless control strategy, using the ATMega328PU’s analog comparator to detect back electromotive force (BEMF) for motor phase timing. Resistor-based voltage dividers (e.g., 33kΩ, 10kΩ) scale 12V phase voltages to safe levels, while a virtual neutral point, formed by resistors, connects to pin 6 (AIN0). The comparator monitors this point against phase BEMF (pins 7, A2, A3), triggering interrupts on zero-crossing events to synchronize PWM signals (~31 kHz, 8-bit resolution) on pins 9, 10, and 11. The IR2104 drivers control high and low-side IRLZ44N MOSFETs, enabling efficient motor commutation. Speed adjustments via PWM signals received from the FC modify PWM duty cycles, enhancing user control. The PDB’s stable 12V (for IR2104) and 5V (for Arduino) outputs ensure reliability [3].

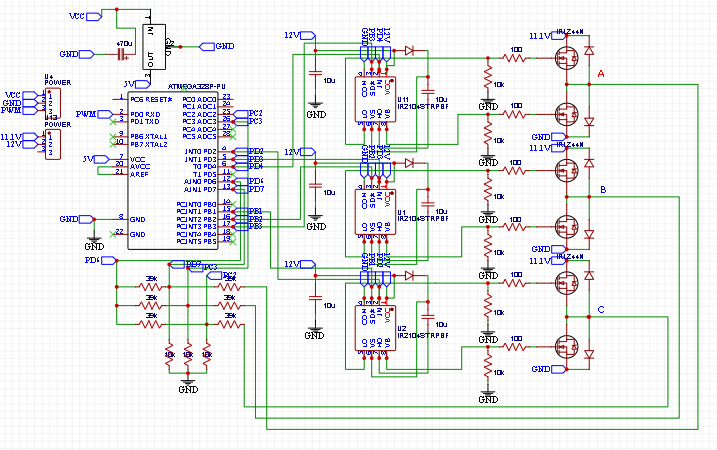
**Power Distribution Board (PDB)** (Figure 2.1): A custom PCB distributes 11.1V from the battery via an XT60 connector to four ESCs through screw terminals (20A rating). It incorporates an LM2576-12 buck converter (Figure 2.11) (with 100 µH inductor, SR5100 Schottky diode, 100 µF/470 µF capacitors) for 12V output to IR2104 drivers and an LM7805 regulator (Figure 2.12) (with 10 µF capacitors) for 5V to the ESP32 and Arduino. A 470 µF capacitor filters input voltage. The PDB ensures stable power delivery, preventing voltage drops that could destabilize the ESC or flight controller.

**LM2576-12 Buck Converter**: This switching regulator, made by Texas Instruments, reduces the battery's 11.1V to a steady 12V output (0.7-3A depending on the feed back) for IR2104 drivers. In comparison to linear regulators, it minimizes heat and achieves approximately 88% efficiency with a 100 µH inductor, SR5100 Schottky diode, and capacitors (100 µF input, 470 µF discharge). [8].

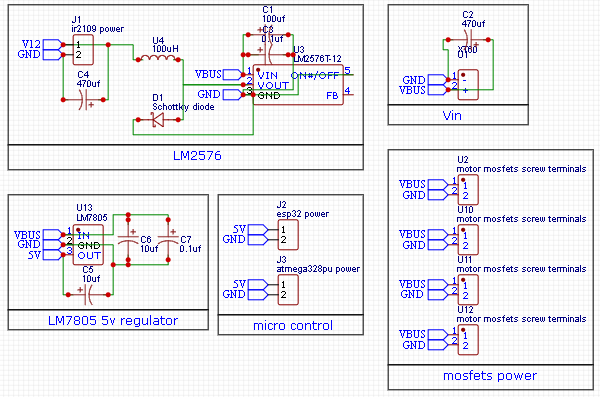
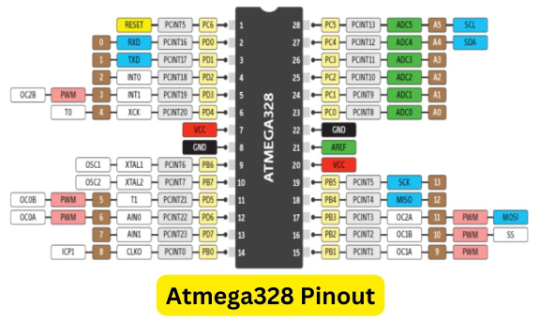
**LM7805 Linear Regulator:** This ON Semiconductor device transforms the battery's 11.1V into a constant 5V (1A) for the Arduino and ESP32. It offers dependable low-noise output when paired with 10 µF capacitors for stability, but it releases surplus voltage as heat, thus a heatsink is required for extended operation. [9].

**3S LiPo Battery**: A 3S LiPo battery, chosen for its high discharge rate and lightweight polymer housing, powers the system with an 11.1V, about 2200–3300 mAh lithium-polymer battery It affects both the duration of the ground operation and the flying time (around 10 to 15 minutes).   
These parts work very closely: the ESC's PWM influences motor performance, the battery's capacity restricts motor runtime, and the PDB's regulation guarantees ESC dependability. The filtering capacitors in the PDB help to mitigate issues like voltage spikes.

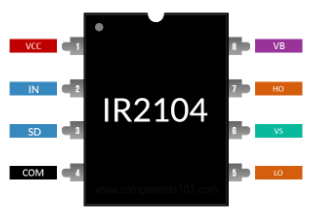
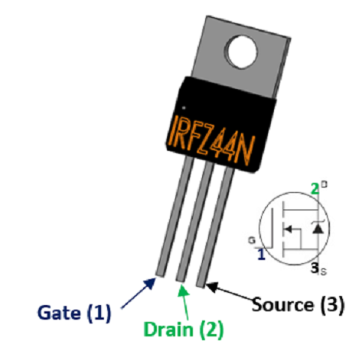
**Figures :**

****

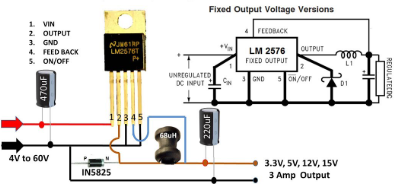
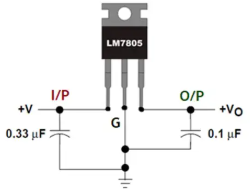
* **Figure 2**: ESC circuit schematic with ATMega328PU, IR204STRPBF drivers, IRLZ44N MOSFETs, and PDB connections. [Request if needed]

* Figure 2.1: PDB schematic. Figure 2.2: ATmega328PU pinout diagram.

* Figure 2.3: IR2104 driver circuit. Figure 2.4: IRLZ44N MOSFET

* Figure 2.11: LM2576-12 buck converter. Figure 2.12: LM7805 regulator circuit.

## 2.2 Control and Navigation Components:

Precision operation and stability in both aerial and road mode are made possible by the control and navigation components. This section describes the components, uses, and contributions to system performance of the ESP32 microcontroller, MPU6050 gyroscope/accelerometer, BMP280 barometer, and web-based control interface. These parts were chosen for the project's hobbyist emphasis because of their affordability, excellent performance, and open-source compatibility. Their combination guarantees precise processing of sensor data and intuitive operation, which affects the drone-car's responsiveness and dependability [3].

**Explanation**:  
**ESP32 Microcontroller** (Figure 2.5): The ESP32 microcontroller, a dual-core, 240 MHz processor with integrated Wi-Fi, is constructed on a 4-layer PCB with surface-mount components. It serves as the flight controller, processing sensor data and hosting the web interface. Its 3.3V logic level and GPIO pins connect to sensors and servos, influencing system latency and control precision [4].  
**Functionality and Role**:  
the ESP32 (Espressif Systems) features two Xtensa LX6 cores, 520 KB SRAM, and 4 MB flash memory, enabling robust multitasking for sensor fusion, PID control, and Wi-Fi communication. Programmed in C++ using Arduino IDE, it integrates MPU6050 and BMP280 data via I2C (figure 2.6), calculating orientation and altitude for stable flight and ground navigation. The ESP32’s Wi-Fi module (802.11 b/g/n) hosts a web server, delivering the control interface to user devices, with TCP/IP ensuring low-latency command transmission (~50–100 ms). Its 36 GPIO pins connect to the MPU6050 (SCL/SDA), BMP280 (I2C), and servos (PWM), coordinating transformation and motor control via ESC commands. The ESP32’s 3.3V operation, powered by the PDB’s 5V-to-3.3V regulator, minimizes power consumption (~150 mA), It interacts with the PDB for stable power, the ESC for motor control, and servos for transformation, making it the system’s central hub. Challenges like Wi-Fi interference are mitigated by channel optimization, or using a ESP32 now base radio transmeter ensuring reliable operation [5].

**MPU6050 Gyroscope/Accelerometer** (Figure 2.6): The MPU6050 gyroscope/accelerometer, a 6-axis MEMS sensor with silicon-based gyroscopes and accelerometers, provides pitch, roll, and yaw data. Mounted on a small breakout board, it communicates via I2C with the ESP32. Its high sensitivity (up to 16-bit resolution) ensures stable flight but requires noise filtering, affecting the ESP32’s processing load [3].

**Functionality and Role**:  
The MPU6050 (InvenSense), integrates three-axis gyroscopes and accelerometers on a silicon die, leveraging MEMS technology for compact, low-power operation (~3.6 mA at 3.3V). It measures angular velocity (±250 to ±2000 °/s) and acceleration (±2 to ±16 g), delivering 16-bit ADC outputs via I2C (400 kHz) to the ESP32’s SCL/SDA pins. Programmed using the Arduino Wire library, it provides real-time orientation data, critical for PID control in aerial mode and navigation in terrestrial mode. The MPU6050’s digital motion processor (DMP) offloads quaternion calculations, reducing ESP32 workload, but its sensitivity to vibrations (from motors or propellers) necessitates software filtering (complementary or Kalman filters). It interacts with the ESP32 for data processing, the BMP280 for complementary altitude data, and the PDB for stable 3.3V power. Noise from motor EMI can degrade accuracy, mitigated by shielding and capacitor decoupling (10 µF). The MPU6050’s data drives servo and motor adjustments, ensuring stable transitions between modes, making it vital for hybrid functionality [6].

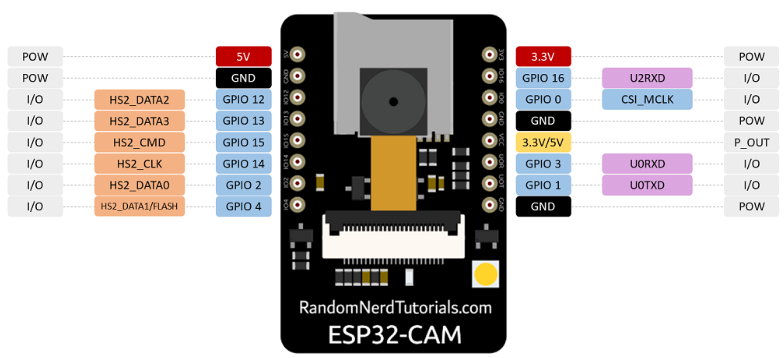
**BMP280 Barometer** (Figure 2.5): The BMP280 barometer, a compact sensor with a piezoresistive element, measures atmospheric pressure for altitude estimation. Its I2C interface integrates with the ESP32, providing 0.2 m accuracy. Its data refines flight stability, complementing the MPU6050 [3].  
**Functionality and Role**:  
The BMP280 (Bosch Sensortec), uses a piezoresistive MEMS element to measure pressure (300–1100 hPa) and temperature, with a low power draw (~2.7 µA at 1 Hz). Connected via I2C (400 kHz) to the ESP32’s SCL/SDA pins, it delivers 24-bit pressure data, achieving ±0.2 m altitude resolution after software calibration. Programmed using the Arduino Adafruit BMP280 library, it provides altitude estimates that enhance PID control for aerial stability, particularly during hover and mode transitions. Its data complements the MPU6050’s orientation inputs, enabling precise navigation in confined spaces. The BMP280’s 3.3V operation, powered by the PDB’s regulator, ensures efficiency, but its sensitivity to EMI from motors requires shielding and decoupling capacitors (10 µF). It interacts with the ESP32 for data processing, the MPU6050 for integrated sensor fusion, and indirectly with servos and motors via stability adjustments, supporting the drone-car’s hybrid functionality. Challenges like pressure noise are mitigated by oversampling and digital filtering [7].

**Web-Based Control Interface**: A software component hosted on the ESP32, built with HTML/CSS/JavaScript, running on a virtual server. It features two pages (drone mode: thrust/yaw/pitch; car mode: servo/ground control), accessible via Wi-Fi-enabled devices. Its responsiveness depends on the ESP32’s Wi-Fi performance and influences user experience (Figure 2.7).  
These components interact synergistically: the ESP32 processes MPU6050/BMP280 data for stability, while the web interface sends commands to the ESP32, affecting motor and servo actions. Noise from the MPU6050 can challenge ESP32 processing, mitigated by software filters.

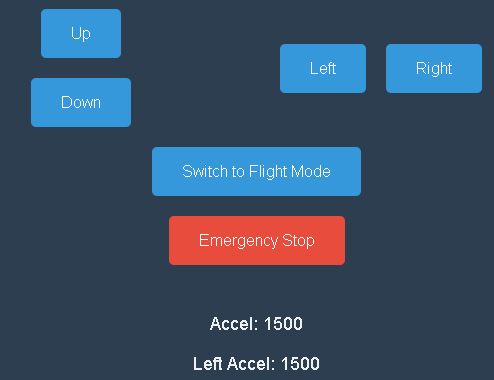
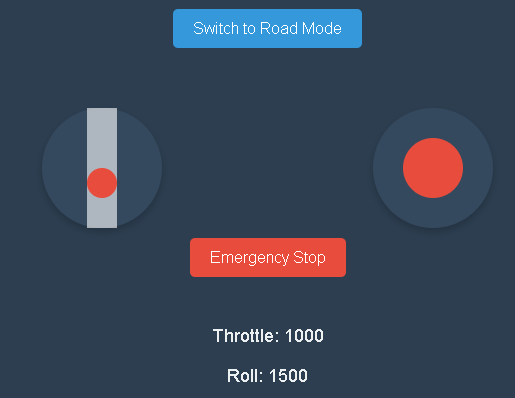
**ESP32-CAM:** The ESP32-CAM, a compact module combining an ESP32-S microcontroller and OV2640 camera, enables real-time imaging for navigation and monitoring. Constructed on a 4-layer PCB with surface-mount components, it integrates a dual-core 240 MHz processor, 4 MB PSRAM, and a 2-megapixel camera. Selected for its low cost and open-source compatibility, it supports the project search function, enhancing the drone-car’s ability to assist in search-and-rescue operations [10]. Figure 2.10

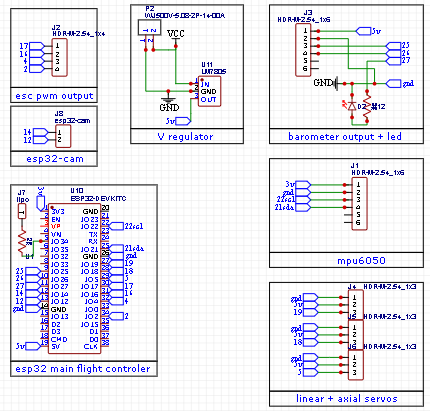
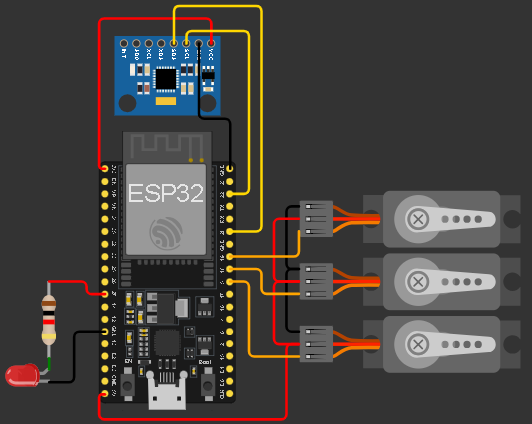
**Figures :**

**Figure 2.10:** esp32-cam pinout diagram

****

**figure2.7 : road mode and flight mode web interface**

  
**Figure 2.6**: Sensor integration diagram, showing MPU6050 and BMP280 with ESP32

****

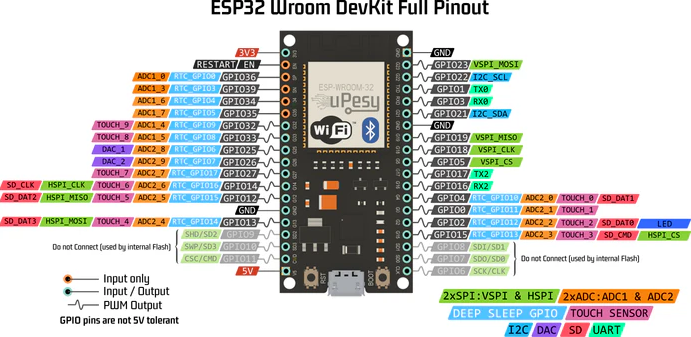
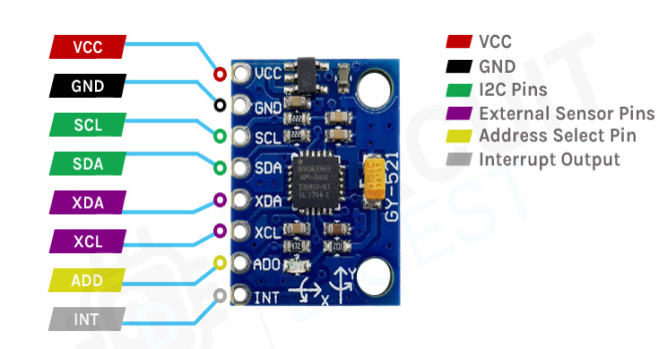
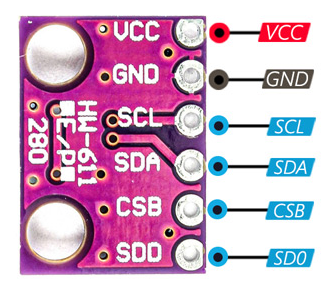


Figure 2.3: ESP32 pin diagram

* Figure 2.4: mpu6050 diagram Figure 2.5:bmp280 diagram

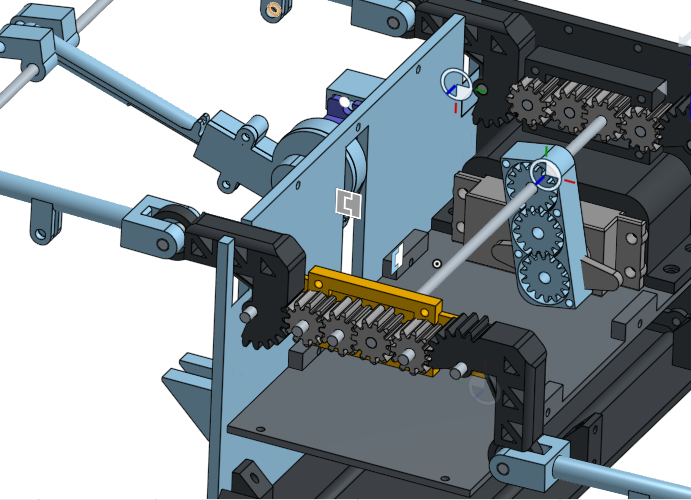
## 2.3 Mechanical and Structural Components

The drone-car's unique transition between airborne and terrestrial modes is made possible by its mechanical and structural components, which also constitute its physical framework. This section describes the materials, functions, and responsibilities of the propellers, frame, and servo motors of the MG955 and MG90 in maintaining mode switching and structural integrity. These parts were chosen for their cost and longevity, and they are made to be hobbyist-accessible while yet satisfying the requirements of hybrid operation. For uses such as cave mapping, its design affects the system's weight, stability, and transformation efficiency [4].

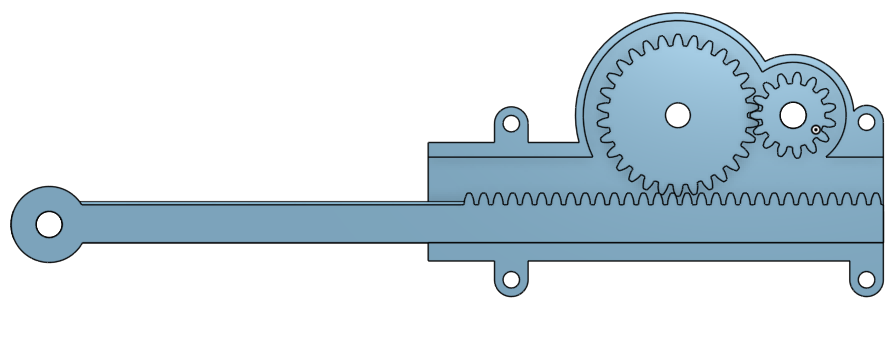
**Explanation**:

* **MG955 Servo Motor** (Figure 2.8): A high-torque servo (10 kg·cm at 6V) with metal gears and a plastic casing, used to lower propeller arms and shoulders via a4 peice gear box for ground mode. Mounted centrally, Its torque ensures reliable arm movement under load, insuring the 1kg drone to stay upright in drive mode.
* **MG90 Servo Motors** (Figure 2.9): Two compact servos (1.8 kg·cm at 5V) with metal gears, one per side, rotate propellers from horizontal (flight) to vertical (wheel-like) positions. Their small size minimizes weight, but their torque limits propeller size, affecting ground traction.

**Figures :**



* Figure 2.8: **mg955** servo, linear gear showcase.



* Figure 2.9: **mg90** servo, axial gear showcase.

## Programmes used:

### Circuit realization:

For the circuit schematics I used easyeda for all the components.for writing the code I used arduino IDE.

## Coding:

the code is written in 4 programming languages C (for the main flight controle and sensore data analysis), HTML CSS and Javasript ( for the webcontroler interface ).

## 3d design:

I used the webbase CAD design software ONSHAPE.

For the full drone schematic and flight controller code and esc code vist my personal github :

<https://github.com/boumetrayane/drone-ground-air-/tree/main/drone>.

# Chapter 3: modélisation et la conception

## 3.1 ESP32 flight controler :

I used the esp32-wroom for the main microcontroler and the mpu6050 for the gyroscope. The pcb unsures the sp32 is connected to all the sensor (mpu6050, bmp280, mg955, mg90, the ESC’s, and power distribution borad) it has a lineare 5V regulator to unsure the esp doesn’t get destroyed in case of the PDB failing. I made the code first and tested it on a bread board and used the web controller and serial output in arduino ide to make sure that the throttle yaw role and pitch values are being correctly communicate to the esp32 for the PID calebration here is how the code works:

## 3.2Flight Controller Operation:

This section explains how the ESP32-based flight controller manages a transforming drone operating in **flight mode** (quadcopter) and **road mode** (ground vehicle). The system integrates a web-based user interface, sensor data from the MPU-6050 (gyroscope and accelerometer) and BME280 (barometer), PID controllers for stabilization, and PWM signal generation to drive electronic speed controllers (ESCs) for motor control. The explanation covers instruction reception, sensor communication, PID processing, and motor actuation in both modes.

## 1. Receiving Instructions from the Web Interface:

The flight controller hosts an HTTP web server on the ESP32, accessible via WiFi, allowing users to control the drone through a browser-based interface on devices connected to the same network (SSID: "P40", password: "rayboum2005", or AP mode: "ESP32\_Drone", "rayane2005").

### Web Server Setup :

* **Initialization**: In setup(), the ESP32 connects to WiFi or creates an access point. The WebServer library runs an HTTP server on port 80, handling routes like / (flight interface), /road (road interface), /control (commands), /status (telemetry), /pid (PID tuning), and /calibrate (sensor recalibration).
* **Operation**: The server.handleClient() function in loop() processes HTTP requests every 4 ms, ensuring real-time responsiveness.

### Flight Mode Interface :

* **Controls**:
  + **Left Joystick**: Adjusts throttle (1000–2000 µs, vertical motion) and yaw (1000–2000 µs, rotation). JavaScript (handleLeftMove) maps movements to HTTP requests (e.g., /control?channel=throttle&value=1750).
  + **Right Joystick**: Controls roll and pitch (1000–2000 µs each), sending requests like /control?channel=roll&value=1600.
  + **Emergency Stop**: Sends /control?channel=emergency&value=1 to halt motors.
  + **Altitude Hold**: A checkbox toggles /config?altitudeHold=1 or 0.
* **Telemetry**: The fetchValues function polls /status every 500 ms, displaying throttle, roll, pitch, yaw, altitude, and vertical velocity.

### Road Mode Interface :

* **Controls**:
  + **Directional Buttons**:
    - "Up"/"Down" send /control?channel=accel&value=X, ramping accel from 1500 to 2000 (forward) or 1000 (backward).
    - "Left"/"Right" send /control?channel=leftaccel&value=X and /control?channel=rightaccel&value=Y for differential steering (e.g., leftaccel=2000, rightaccel=1000 for left turn).
  + **Emergency Stop**: Similar to flight mode.
* **Telemetry**: Polls /control?get\_values=1 every 500 ms, showing accel, leftaccel, and rightaccel.

### Server-Side Processing :

* **Command Handling**: The handleControl function updates variables like throttleValue, rollValue, emergencyStop, and roadMode based on /control requests.
* **Telemetry**: The handleStatus function returns JSON with current values (e.g., {"throttleValue":1750, "altitudeValue":100.0}).
* **Configuration**: The handleConfig and handleCalibrate functions manage altitude hold and barometer recalibration.

## 2. Communication with MPU-6050 (Gyroscope and Accelerometer):

The MPU-6050 provides 6-axis motion data for orientation and stabilization in flight mode.

### Initialization :

* **Setup**: Initialized via I2C (pins 21 SDA, 22 SCL, address 0x68, 400 kHz) with:
  + Power on (register 0x6B = 0x00).
  + Low-pass filter (0x1A = 0x05, 10 Hz).
  + Gyroscope range (0x1B = ±500°/s).
  + Accelerometer range (0x1C = ±8g).
* **Calibration**: Averages 2000 readings to compute biases (RateCalibrationRoll, AccXCalibration, etc.).

### Data Acquisition :

* **Function**: gyro\_signals reads every 4 ms:
  + Accelerometer: AccX, AccY, AccZ (g, ±8g range).
  + Gyroscope: RateRoll, RatePitch, RateYaw (°/s, ±500°/s).
* **Angle Calculation**:
  + Roll: AngleRoll = atan(AccY / sqrt(AccX^2 + AccZ^2)) \* 57.29 (°).
  + Pitch: AnglePitch = -atan(AccX / sqrt(AccY^2 + AccZ^2)) \* 57.29 (°).
* **Calibration**: Subtracts biases to zero rates and adjust AccZ to -1g when stationary.

### Kalman Filtering :

* **1D Kalman Filter** :(kalman\_1d): Fuses gyroscope rates and accelerometer angles to estimate KalmanAngleRoll and KalmanAnglePitch.
* **Parameters**: Process noise (4°/s), measurement noise (3°), timestep (4 ms).
* **Output**: Smooth angles for PID control.

## 3. Communication with BME280 (Barometer):

The BME280 measures altitude for altitude hold in flight mode.

### Initialization :

* **Setup**: Initialized on a second I2C bus (pins 25 SDA, 26 SCL, 400 kHz, addresses 0x76 or 0x77, with retries).
* **Calibration**: Averages 2000 readings to set AltitudeBarometerStartUp (baseline altitude, cm).

### Data Acquisition :

* **Function**: barometer\_signals reads every 24 ms (every 6th loop):
  + Altitude: AltitudeBarometer = bme.readAltitude(1013.25) \* 100 (cm).
  + Fallback: Uses last AltitudeKalman if NaN.
* **Adjustment**: Subtracts AltitudeBarometerStartUp for relative altitude (0 cm at calibration).

### Kalman Filtering :

* **2D Kalman Filter** (kalman\_2d): Estimates AltitudeKalman (cm) and VelocityVerticalKalman (cm/s) using:
  + Barometer altitude (AltitudeBarometer).
  + Inertial acceleration (AccZInertial), computed in the inertial frame (cm/s²).
* **Parameters**: Timestep (4 ms), measurement noise (±30 cm), acceleration noise (scaled by 100).
* **Output**: Smooth altitude and velocity for altitude PID.

## 4. PID Control:

PID controllers stabilize orientation and altitude in flight mode.

### Angle PID :

* **Purpose**: Matches roll/pitch to user inputs.
* **Inputs**:
  + Desired: DesiredAngleRoll = 0.10 \* (rollValue - 1500) (°), clamped to ±20°.
  + Actual: KalmanAngleRoll, KalmanAnglePitch.
* **Parameters**: PAngleRoll = 2, IAngleRoll = 0.5, DAngleRoll = 0.007.
* **Output**: InputRoll, InputPitch (desired rates, °/s).

### Rate PID :

* **Purpose**: Matches angular rates to angle PID outputs or yaw input.
* **Inputs**:
  + Desired: InputRoll, InputPitch, DesiredRateYaw = 0.15 \* (yawValue - 1500).
  + Actual: RateRoll, RatePitch, RateYaw.
* **Parameters**:
  + Roll/Pitch: PRateRoll = 0.625, IRateRoll = 2.1, DRateRoll = 0.0088.
  + Yaw: PRateYaw = 4, IRateYaw = 3, DRateYaw = 0.
* **Output**: Motor adjustments.

### Altitude PID :

* **Purpose**: Maintains desired vertical velocity (altitude hold).
* **Inputs**:
  + Desired: DesiredVelocityVertical = 0.3 \* (throttleValue - 1500) / 500.0 (m/s).
  + Actual: VelocityVerticalKalman / 100.0 (m/s).
* **Parameters**: PVelocityVertical = 3.5, IVelocityVertical = 0.0015, DVelocityVertical = 0.01.
* **Output**: InputThrottle (1000–2000 µs).

## 5. Generating PWM Signals for ESCs :

Four ESCs drive brushless motors (pins 17, 16, 4, 2) using PWM signals (500 Hz, 1000–2000 µs).

### Flight Mode :

* **Motor Mixing**:
  + Combines InputThrottle and PID outputs for roll, pitch, and yaw corrections.
  + Example: MotorInput1 = InputThrottle - PIDReturn[0] - PIDReturn[1] + PIDReturn[2] (front-left).
* **Output**: motX.writeMicroseconds(MotorInputX) sets motor speeds.

### Road Mode :

* **Control**:
  + Uniform speed: MotorInputX = accel (if accel != 1500).
  + Differential steering: MotorInput1 = MotorInput2 = rightaccel, MotorInput3 = MotorInput4 = leftaccel.
* **Output**: Same PWM mechanism as flight mode.

## 6. Transformation Servos :

* **Purpose**: Reconfigure the drone between modes using three servos (pins 18, 19, 5, 50 Hz, 1000–2000 µs).
* **Operation**: Set to flightModeServoPos (1000 µs) or roadModeServoPos (2000 µs) via /control?channel=roadMode. (figure3.2).

## 7. Safety Features :

* **Emergency Stop**: Halts motors (MotorInputX = 1000) on button press or low battery (Voltage < 10.5V for 5 readings).
* **Battery Monitoring**: Uses a 5-sample moving average on analog pin 34.
* **Sensor Fallback**: Uses last Kalman estimates if sensor readings fail.

## 8. Data Flow :

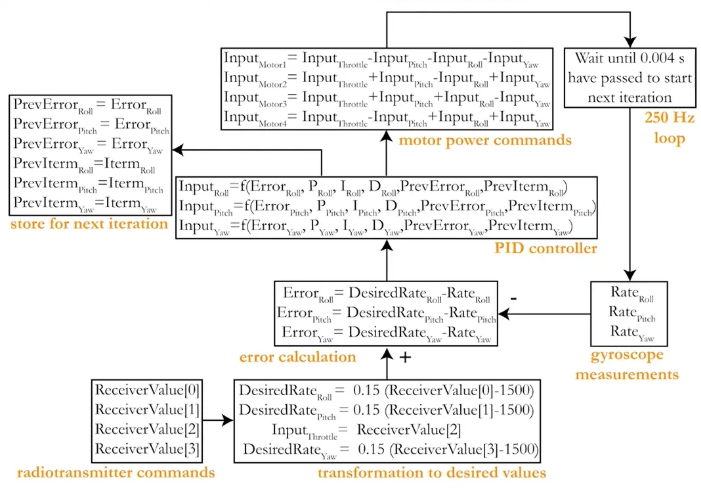
1. **Input**: Web interface sends HTTP requests to update control variables.
2. **Sensors**: MPU-6050 (4 ms) and BME280 (24 ms) provide motion and altitude data, filtered by Kalman filters.
3. **PID**: Computes control outputs for stabilization and altitude hold.
4. **Motors**: Generates PWM signals for ESCs based on mode-specific logic.
5. **Servos**: Adjusts configuration for flight or road mode.

## 9. Mode-Specific Behavior :

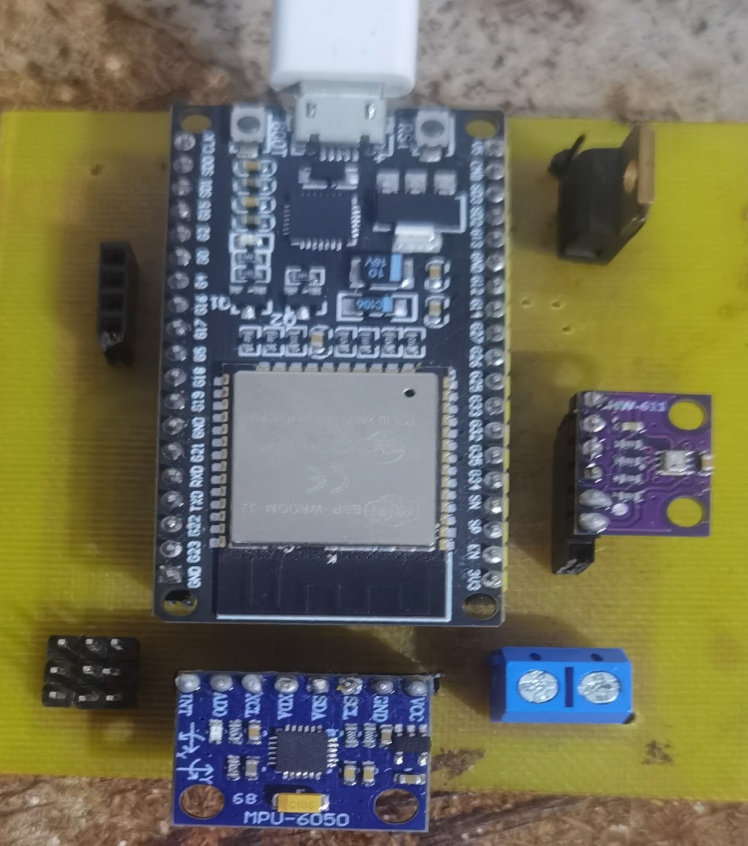
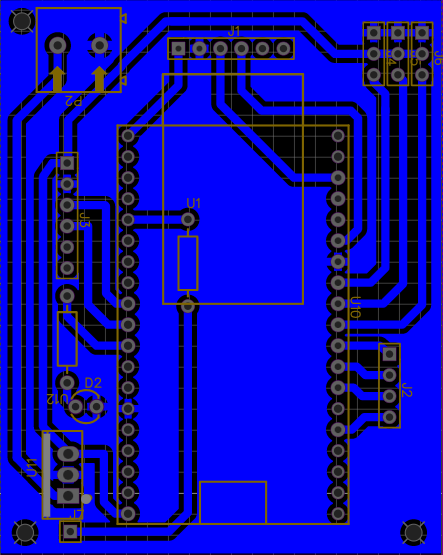
* **Flight Mode**: Stabilizes using cascaded PID controllers; optional altitude hold.
* **Road Mode**: Direct motor control for forward/backward motion and differential steering.

**Figures :**

* **Figure 3.2: flight controller stabilization closed loop**

****

for the pcb conception I made the schematic (figure 3.3) on easyeda then made the pcbs(figure 3.4) using the toner methode due to the absent of a perochlorure supplier I had to make my ownusing HCL some iron nails Fe and oxiginated water H2O2.



* figure 3.3: FC pcb design figure 3.4: FC conception

## 3.3 DIY ESC :

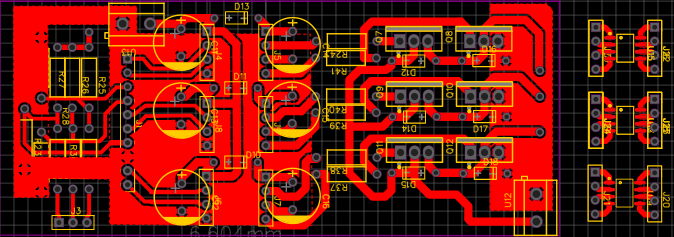
I used the arduino atmega328p for the main microcontroler and ir2104s for the mosfet drivers due to the non existence of the through-hole variant in the Algerian market and the irlz44n mosfet. I tested the code first on a bread board and then I designed the pcb for it(figure3.5), after that I made the conception to test the final version(figure3.6)

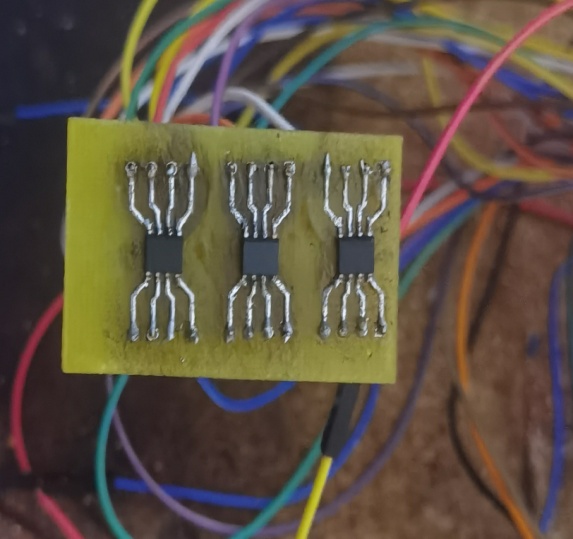
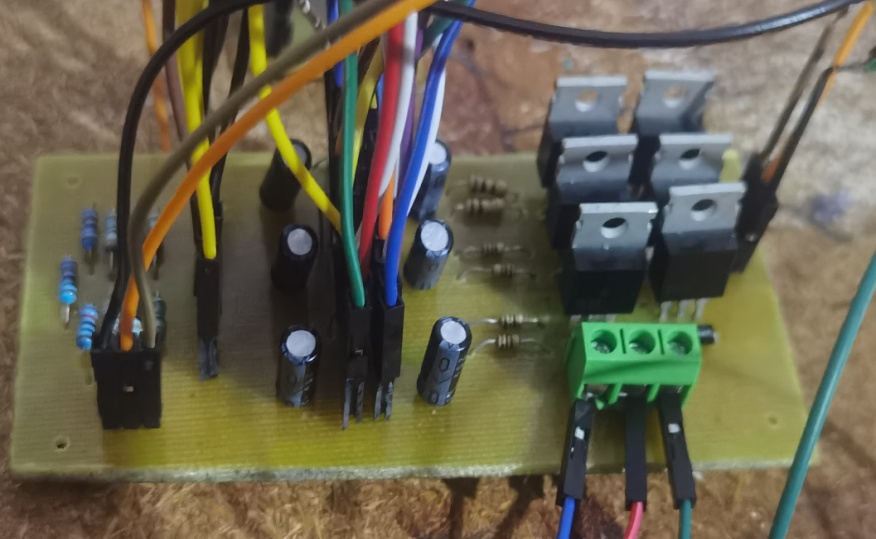
### 3.1 DIY ESC Operation :

This section explains how the Arduino-based DIY Electronic Speed Controller (ESC) manages a brushless DC (BLDC) motor in the convertible drone-car, using sensorless control for aerial and terrestrial modes. The system integrates an ATmega328PU microcontroller, IR2104S gate drivers, IRLZ44N MOSFETs, and supporting components (resistors, capacitors, diodes, pushbuttons), powered by a power distribution board (PDB). The Arduino code orchestrates BEMF detection, PWM signal generation, and commutation, ensuring precise motor control. The explanation covers initialization, BEMF processing, PWM output, speed adjustments, and component interactions, illustrated in Figure 3.1.

1. **Initialization**:
   * **Code Setup**: The setup() function configures the ATmega328PU for ESC operation. Pins 9, 10, and 11 (PWM outputs, Timer1/Timer2) are set for ~31 kHz, 8-bit PWM signals, controlling IR2104S HIN lines for phases A, B, and C. Pins 3, 4, and 5 manage LIN lines. The analog comparator is enabled (ACSR = 0x10), with pin 6 (AIN0) as the positive input (virtual neutral point). Pushbuttons (A0, A1) are initialized with pull-up resistors for speed control.
   * **Hardware**: The PDB’s LM7805 supplies 5V to the Arduino, and LM2576-12 provides 12V to IR2104S drivers, ensuring stable power [8, 9].
2. **BEMF Detection**:
   * **Operation**: The analog comparator compares the virtual neutral point (33kΩ resistors, pin 6) against phase BEMF (pins 7, A2, A3 for phases A, B, C). Voltage dividers (33kΩ/10kΩ) scale 12V phase voltages. The ISR(ANALOG\_COMP\_vect) triggers on rising/falling zero-crossing events, debouncing for 10 cycles to ensure accuracy.
   * **Code Functions**: Functions like BEMF\_A\_RISING() select comparator inputs (e.g., AIN1 for phase A) and set interrupt edges (ACSR), updating the commutation step (bldc\_step) [3].
3. **PWM Generation and Commutation**:
   * **Code Logic**: The bldc\_move() function executes six-step commutation, callin*g* functions (e.g., AH\_BL()) to set IR2104S HIN/LIN lines, energizing motor phases via IRLZ44N MOSFETs. PWM duty cycles (OCR1A, OCR1B, OCR2A) control high-side MOSFETs, modulating motor power.
   * **Hardware**: IR2104S drivers switch IRLZ44N MOSFETs, handling 20A per phase. Capacitors (10µF, 2.2µF) and IN4148 diodes stabilize gate signals [3].
4. **Speed Control**:
   * **Operation**: The loop() function starts the motor with a ramping sequence (5000–100 µs delays) and monitors pushbuttons. Pressing A0 (SPEED\_UP) increases the PWM duty cycle (motor\_speed, 50–255), while A1 (SPEED\_DOWN) decreases it, updated via SET\_PWM\_DUTY.
   * **Code**: Duty cycle limits (PWM\_MIN\_DUTY=50, PWM\_MAX\_DUTY=255) ensure safe operation, with 100 ms delays for smooth adjustments.
5. **Component Interactions**:
   * **Arduino ATmega328PU**: Executes the code, processing BEMF interrupts and generating PWM signals.
   * **IR2104S Drivers**: Control high/low-side IRLZ44N MOSFETs, powered by PDB’s 12V, with 10Ω resistors limiting gate current.
   * **IRLZ44N MOSFETs**: Switch motor phases, supported by a 1.6 mm FR4 PCB with copper traces for high-current paths.
   * **Resistors/Capacitors/Diodes**: 33kΩ/10kΩ dividers and 10Ω resistors ensure safe signal levels; 10µF/2.2µF capacitors and IN4148 diodes filter noise.
   * **Pushbuttons**: Enable user speed control.
   * **PDB**: Supplies 12V (LM2576-12) and 5V (LM7805); a 470µF capacitor mitigates voltage spikes, ensuring circuit reliability [8, 9].

**Figures**: **figure 3.4:** ESC pcb design

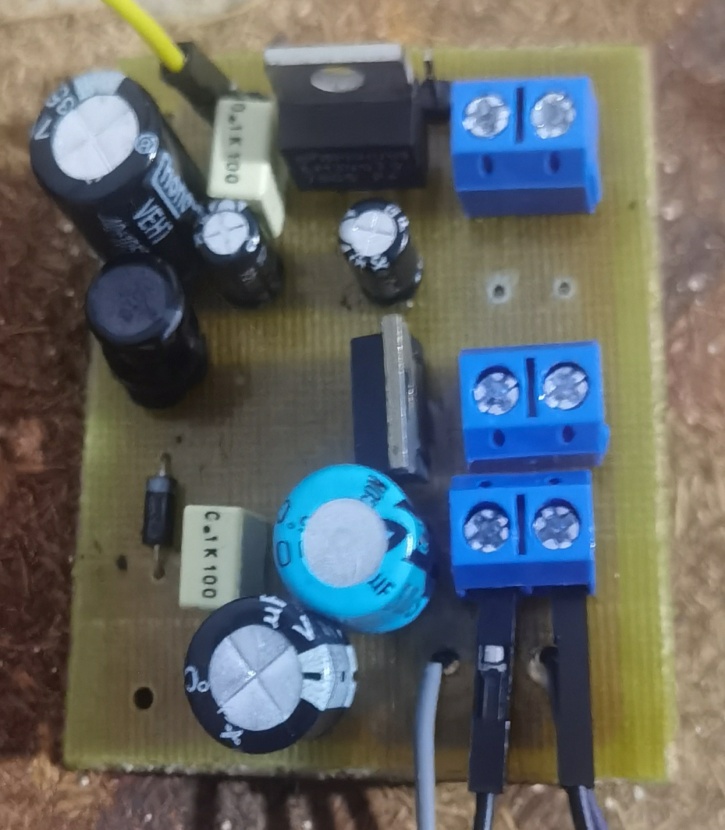
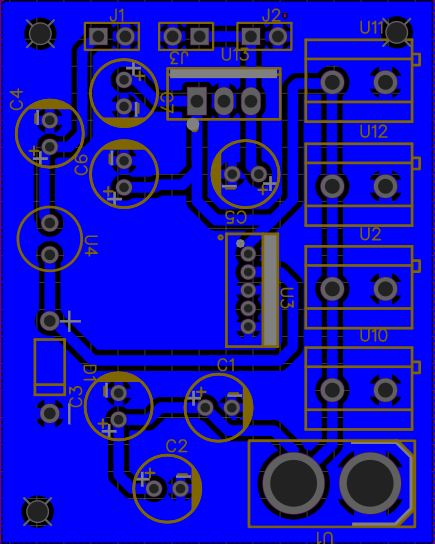
****

****

* **figure3.6:** ESC throught-hole and SMD components conception

## 3.4 power distribution board:

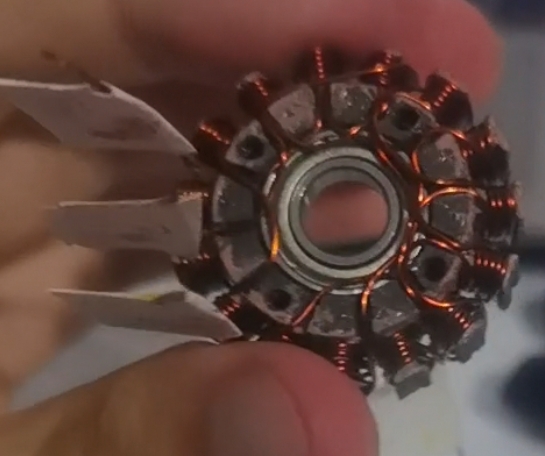
this was the easiest board I just followed the data sheet and tested it on a bread board where I found I had made a mistake in one of the capacitor upon correction it worked as intended and I made the pcb using easyeda like the last parts.no code is required for this circuit(figure3.7)



* **figure3.7**: PDB pcb design and conception

## 3.5 brushless motor:

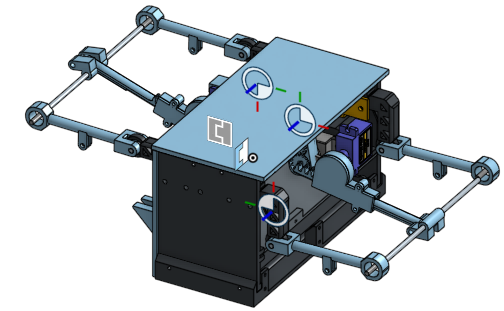
this was first designed on ONSHAPE in 2 part the rotor and stator using 2 688z bearing and a a screw for the axe we get a full motor (figure3.8). I made the rotor out of ¨PLA filament and brought neodymium magnets and used 20 for each rotor. As for the stator I used a iron filed filament to give the core iron properties for high electromagnetique fieled generation and better conductivity it hase 15 poles 5 pairs of 3 wired by hand (A ,B, C) (figure3.9).



* **figure(3.8):** finishe motor **figure(3.9):** wired stator

## 3.6 drone frame an gears:

made entirely in ONSHAPE it’s designed to be 3d printed in 44 seprate parts(figure 3.10) , the main frame can be made with carbon fiber or wood depending on the preference. All the gears are PETG and can be CNC’ed in metal. I printed each individual piece then assemble them using 2M4 screws and hot glue (figure 3.11 (1) (2)). I used 5 stainless steel rodes 3mm wide 200mm tall for all the axes that I cut depending on the part.the propellers are made to function as wheels (figure3.12)



* figure 3.10: main frame figure 3.11 (1): 3d printed parts

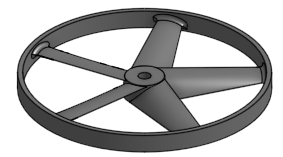


figure 3.11 (2) figure 3.12: propeller wheel design

## 3.7 genrale Conclusion :

This project helps you created a DIY convertible drone vehicle that combines airborne and ground-based capabilities for search and rescue and cave mapping. Stable navigation was made possible via an ESP32 flight controller equipped with MPU6050 and BMP280 sensors, custom brushless motors, and an Arduino-based ESC. A web interface was used to control the smooth mode switching made possible by the servo-driven transformation system. Despite obstacles like PCB manufacturing, amateurs will feel empowered by the modularand and affordable architecture. By adding a front-mounted camera utilizing the esp32-cam and esp-now.

## 4.1 Bibliography:

[1] Rechtin, M., “Transforming-Quadcopter: A Hybrid Quadcopter Design,” [Online]. Available: <https://github.com/MichaelRechtin/Transforming-Quadcopter>.   
[2] Boumet, R., “Drone-Ground-Air: Convertible Drone-Car Design and Power System Integration,” [Online]. Available: <https://github.com/boumetrayane/drone-ground-air-/tree/main/drone>. + Carbon Aeronautics, “DIY Quadcopter Drone Development Resources,” [Online]. Available: <https://github.com/CarbonAeronautics>.  
[3] Simple Circuit, “Arduino Sensorless BLDC Motor Controller,” [Online]. Available: <https://simple-circuit.com/arduino-brushless-dc-motor-controller/>.  
[4] Onshape Community, “Convertible Drone-Car Mechanical Design,” [Online]. Available: <https://cad.onshape.com/documents/35bc670a0eb66b336e0ebfa8/w/e56c4f4991d3954377268b1d/e/e93c529e365ec0daa0cdfe21>.  
[5] Espressif Systems, “ESP32 Series Datasheet,” [Online]. Available: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>.   
[6] InvenSense, “MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking Devices,” [Online]. Available: <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>.   
[7] Bosch Sensortec, “BMP280 Digital Pressure Sensor,” [Online]. Available: <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp280-ds001.pdf>.   
[8] Texas Instruments, “LM2576 SIMPLE SWITCHER® 3A Step-Down Voltage Regulator,” [Online]. Available: <https://www.ti.com/lit/ds/symlink/lm2576.pdf>.   
[9] ON Semiconductor, “LM7805 1A Positive Voltage Regulator,” [Online]. Available: <https://www.onsemi.com/pub/Collateral/LM7805-D.PDF>.  
[10] Espressif Systems, “ESP32-CAM Product Brief,” [Online]. Available: <https://www.espressif.com/sites/default/files/documentation/esp32-cam_product_brief_en.pdf>.