CP-301: DEVELOPMENT OF ENGINEERING PROJECT REPORT

Submitted By:

1. Nikhil Kumar (2021EEB1190)
2. Priyansh Kashyap (2021EEB1196)
3. Pulkit Singh (2021EEB1198)
4. Ritul Anand (2021EEB1207)

Submitted To- **Dr. Ashwani Sharma**

Date - 10/05/2024

ACKNOWLEDGEMENT

We would like to express our deepest gratitude for the invaluable guidance and
support provided by the course instructor Dr. Ashwani Sharma and Project Instructor Mr. Vivek Kumar Srivastava throughout this endeavor. Your expertise and mentorship have been instrumental in helping us navigate the complexities of this project and in achieving our goals. Your dedication to fostering innovation and excellence in our academic pursuits has not only inspired us but also provided us with the necessary tools and knowledge to excel in our field. We are immensely grateful for the opportunity to learn under
your leadership and are eager to apply the skills and insights gained from this project in future endeavors.
Thank you once again for your unwavering support and for believing in our potential. We look forward to continuing our journey of learning and innovation
under your guidance.

With profound appreciation,
1. Nikhil Kumar (2021eeb1190)
2. Priyansh Kashyap (2021eeb1196)
3. Pulkit Singh (2021eeb1198)
4. Ritul Anand (2021eeb1207)

Development Of Autonomous Drone Wireless Charging Station

Aim:

Design and development of hardware and software apparatus for autonomous drone wireless charging station.

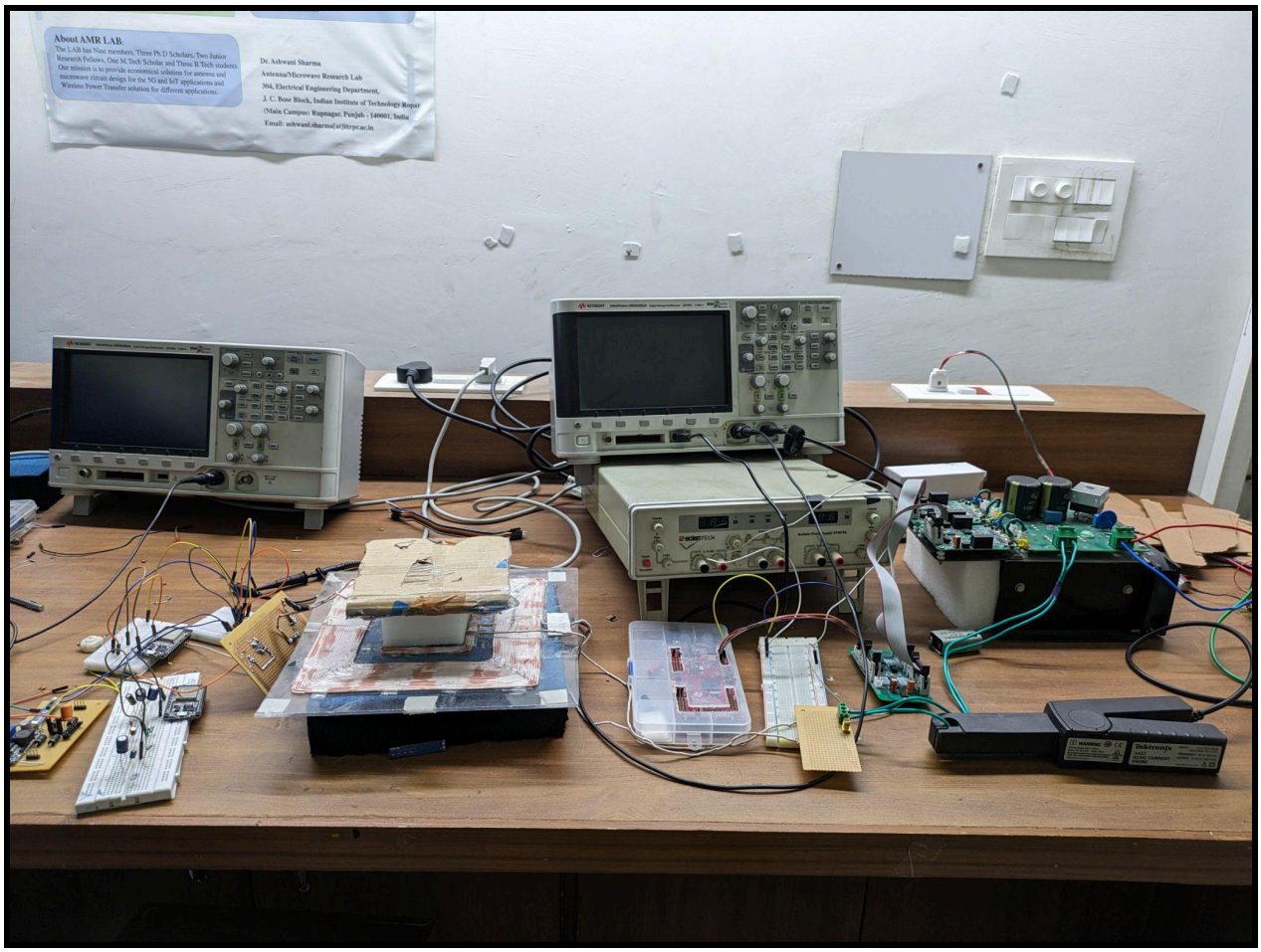


Fig 0: Practical Apparatus of WPT System

Components of Hardware Unit:

1. Input: (12-0-12) Volts step down transformer,
2. Square Wave Inverter (100 kHz),
3. Transmitter matching network,
4. Transmitter Coil,
5. Receiver Coil,
6. Receiver matching network,
7. Rectifying circuit
   - o Single-phase Schottky diode bridge rectifier,
   - o Buck converter (DC-DC Stepdown),
   - o Charging indication circuit,
   - o Program controller (ESP-32),
   - o Protection circuit
8. Drone Battery (7.4 Volts).
9. Connecting Wires (Jumpers, J-Link, Litz wire)

METHODOLOGY:

Hardware:

1. From the 230V AC mains, we stepdown it into 12 volts or 24 volts as per the requirement by (12-0-12) Volts stepdown transformer delivering the AC sinusoidal voltage waveform.

2. This AC output is fed into the inverter unit which uses SPWM (Single Pulse Width Modulation) technique to convert it into a square pulse of 100kHz frequency.

   a. This inverter uses high power high-frequency switches such as IGBTs, Power MOSFETs for efficient switching.

3. This square wave output is given to an impedance-matching network which is designed by measuring the impedance between the transmitter ends considering the resonance condition (X L =X C) using an LCR meter and then designing the matching network by series and parallel combination of capacitors.

a. To ensure effective power transmission and reduce signal reflections, an impedance-matching network is a circuit intended to change the impedance

of an antenna or load to match the impedance of the transmitter or receiver.

4. Matching network is connected to the transmitter coil made of Litz wire, which generates a non-uniform magnetic field for the purpose of optimal landing of the drone at the desired position. Optimal size used in the apparatus of transmitter coil is 25x25 cm.

5. The non-uniform magnetic field generated by transmitter coil is fed into rightly placed receiver coil, whose output is again fed to an impedance matching network made by following the same procedure as stated in '3'. The size of the receiver coil is taken to be 15x15 cm.

6. The output of matching network is passed into the receiver PCB unit which consists of a single-phase bridge rectifier, Buck converter (DC-DC Stepdown), Charging indication circuit, Program controller (ESP-32), Protection circuit.

a. Bridge rectifier converts the received AC sinusoidal output to DC signaloutput. It contains a filter RC circuit which a crucial role in reducing theripples in the DC output obtained. R and C values of this filter circuit are chosen carefully to minimize the output ripples to a higher extent.

b. DC output is fed into the buck converter which further steps down the inputDC voltage to a desired value needed for charging the battery.
c. This stepped down DC voltage is fed to a protection circuit containing various protection devices such a varistors and snubbers. The purpose of this protection circuit is to avoid the backflow of the charges from battery to the source.

7. The output of this PCB unit is fed into the drone battery directly and battery starts charging to an optimal value.

TESTS CONDUCTED AND RESULTS OBTAINED

Arrangements made:

When replacing an autotransformer unit, the output receiver voltage's operating range is taken into account, which is between 7 to 10 volts. Selecting a step-down transformer configuration of 12-0-12 Volts, 2 amperes, which can supply up to 24 Volts of sinusoidal input voltage; by adjusting the variable resistor (potentiometer), we can obtain a range of 7-10 Volts at the receiver end.

TEST-01: Observation of current and voltage characteristics by placement of aluminum, steel sheets, and absorbing foam over the Transmitter and Receiver coils.

Procedure:

1. Aluminium sheets, steel sheets, and absorbing foam are taken of the size of the transmitter coil and receiver coil respectively.
2. These sheets are then positioned beneath the transmitter coil and above the receiver coils so as to obtain the changes in the receiver end voltage and transmitter end current.

Observations:

1. The transmitter and Receiver are kept away from the inverter unit.

   Link: Away from inverter.mp4

2. By placing the Aluminium plates:

   Link: Aluminium Plates.mp4

3. By placing the Absorbing foam:

   Link: Absorbing Foam.mp4

PCB DESIGNING AND COMMUNICATION FEEDBACK SYSTEM

1. Reciever PCB Design:

Circuit diagram:

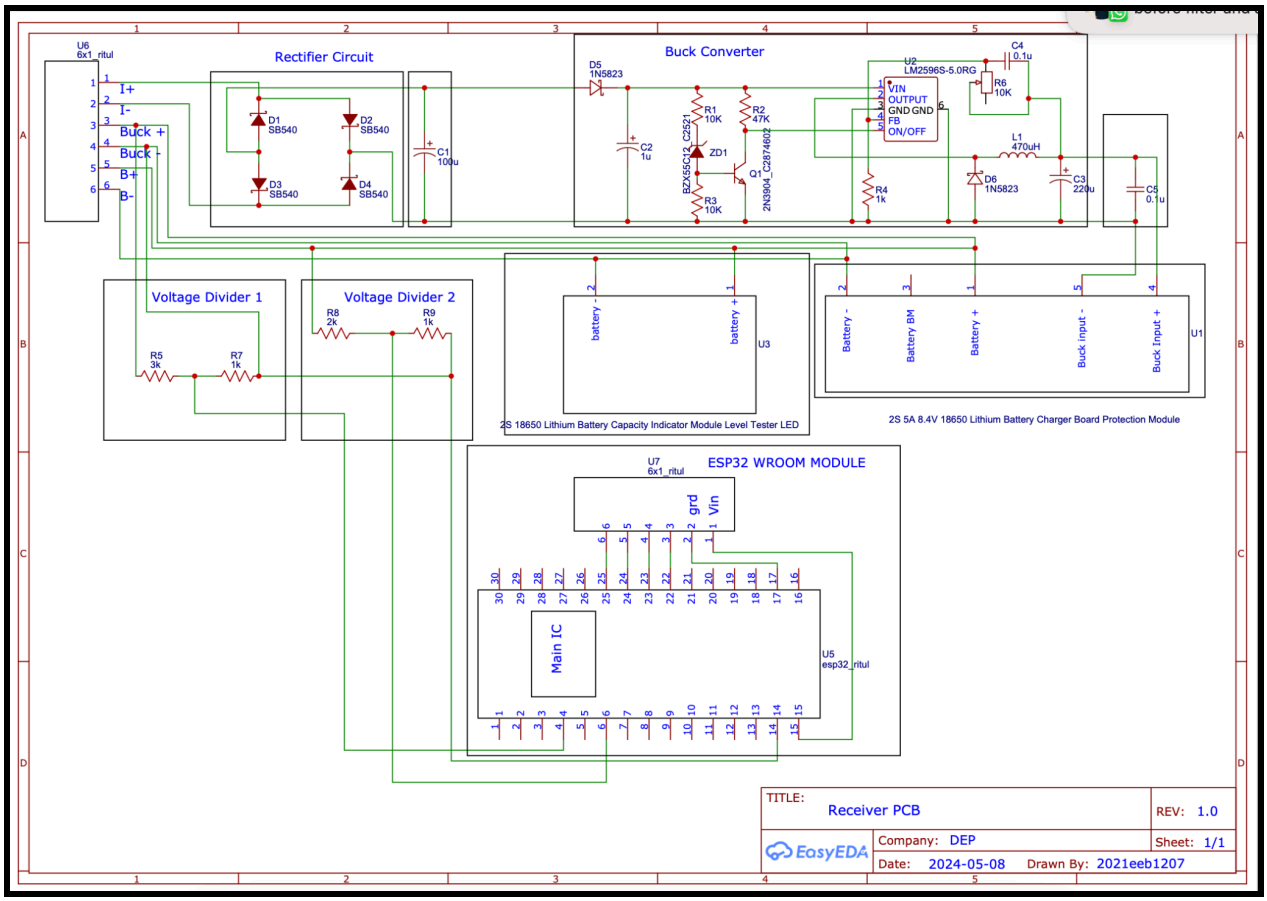


Fig 1. Schematic layout of PCB

Components used in designing the PCB Layout:

Link to the components list:

ID	Name	Designator	Footprint	Quantity	Price	Pins
1	100u	C1	CASE-A_3216	1		2
2	1u,100u,680u	C2	CASE-A_3216	3		2
3	220u	C3	CASE-A_3216	1		2
4	0.1u	C4,C5	C0603	2		2
5	SB540	D1,D2,D3,D4	DO-201AD_BD5	4		2
6	1N5823	D5,D6	DIO-TH_L7.6-W	2	57.059	2
7	470uH,47uH,33(	L1	L0603	1		2
8	2N3904_C2874(	Q1-BJT	TO-92-3_L4.6-W	1	0.01	3
9	10K	R1,R3	R0603	2		2
10	47K	R2	R0603	1		2
11	1k	R4,R7,R9	R0603	3		2
12	3k	R5	R0603	1		2
13	2k	R8	R0603	1		2
14	10K,100K,1K	R6-potentiomete	RES-ADJ-TH_3	3		3
15	2S 5A 8.4V 186	U1	2S 5A 8.4V 186	1		5
16	LM2596S-5.0RC	U2	TO-263-5_L10.2	1	0.413	6
17	2S 18650 Lithiur	U3	2S 18650 LITHI(	1		2
18	ESP32 MODUEI	U5	ESP32 WROOM	1		30
19	6X1 Connector p	U6,U7	6X1_CONN_RIT	2		6
20	BZX55C12_C25	ZD1-Zener diode	DO-35_BD2.0-L	1	0.017	2

https://docs.google.com/spreadsheets/d/1CbN28CKcEzPpRhlWWVobzNYjNCIt8PUi8pNkJKSryN0/edit#gid=1745137848

Functions of different parts of PCB layout:

- **Rectifier**: A full-wave rectifier is an electronic circuit that converts alternating current (AC) into direct current (DC) by allowing current flow in one direction during both the positive and negative halves of the AC cycle. This results in a waveform that is entirely positive or "rectified" to DC.

Why we are using Schottky diodes?

Schottky diodes are often favored over conventional PN junction diodes for several reasons:

1. Reduced Forward Voltage Drop: Schottky diodes exhibit a lower forward voltage drop compared to PN junction diodes. This characteristic results in faster switching speeds and lower power dissipation, making them advantageous for high-frequency applications.
2. Enhanced Switching Speed: Schottky diodes feature significantly faster switching speeds due to their construction. This attribute makes them well-suited for applications that demand rapid switching, such as rectifiers and high-frequency circuits.
3. Shorter Reverse Recovery Time: Schottky diodes boast a much shorter reverse recovery time than PN junction diodes. This capability allows them to transition more swiftly from a conducting to a non-conducting state, thus reducing switching losses and enhancing efficiency in specific applications.
4. Improved Temperature Stability: Schottky diodes generally exhibit superior temperature stability compared to PN junction diodes. This characteristic makes them suitable for deployment in environments characterized by fluctuating temperatures.
5. Versatile Applications: Schottky diodes find widespread use in voltage clamping, rectification, and power supply applications, leveraging their unique characteristics to offer benefits over PN junction diodes.

However, it's essential to acknowledge that Schottky diodes have limitations, such as higher leakage current and lower breakdown voltage when compared to PN junction diodes. Consequently, the

selection between Schottky diodes and PN junction diodes hinges on the specific performance requirements of the intended application.

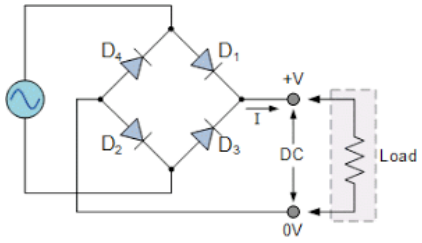


Fig 2. Full bridge Rectifier

Why RC filter?

Ans: An RC filter circuit, also known as a resistor-capacitor filter circuit, is a type of electronic filter circuit used to remove certain frequencies from a signal while allowing others to pass through. Specifically, in the context of voltage regulation, an RC filter is often used to smooth out fluctuations or "ripple" in a DC voltage signal.

Here's how it works:

1. Components: An RC filter consists of a resistor (R) and a capacitor (C) connected in series or parallel configuration.
2. Filtering Action: When a varying voltage signal passes through the RC filter, the capacitor charges and discharges in response to changes in the input voltage. The resistor limits the rate of charge and discharge of the capacitor. As a result, high-frequency components of the input signal are attenuated, while lower-frequency components, including the DC component, are allowed to pass through relatively unaffected.
3. Voltage Ripple Removal: In the context of voltage regulation, an RC filter is commonly used after a rectifier circuit to smooth out the pulsating DC voltage (often referred to as "ripple") produced by the rectification process. The capacitor in the RC filter charges during the peaks of the rectified

waveform and discharges during the troughs, effectively reducing the ripple voltage and producing a smoother, more stable DC output voltage.

4. Advantages: Using an RC filter to remove voltage ripple is advantageous because it provides a cost-effective and relatively simple solution for improving the quality of a DC voltage signal. It can help ensure that electronic devices powered by the DC voltage receive a more stable and consistent power supply, reducing the likelihood of malfunctions or performance issues.

5. Limitations: While RC filters are effective at smoothing out voltage ripple to some extent, they have limitations in terms of their ability to completely eliminate ripple or handle large fluctuations in input voltage. In applications where very low ripple voltage or high precision voltage regulation is required, more complex filtering techniques or additional circuitry may be necessary.

In summary, an RC filter circuit is used to remove voltage ripple from a DC voltage signal by attenuating high-frequency components, thereby producing a smoother and more stable output voltage.

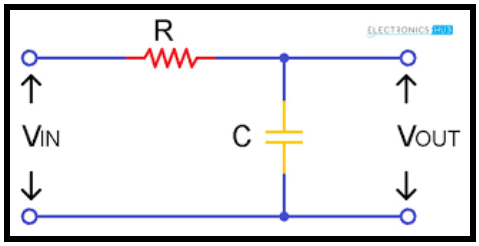


Fig 3. RC filter

Fig 4. Output before RC filter

After adding RC filter ripples got reduced –

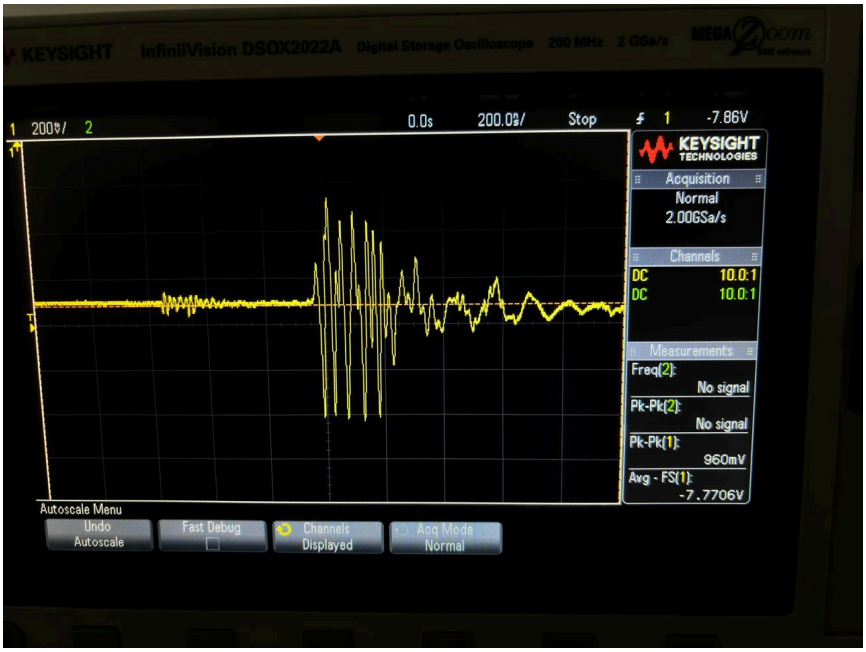


Fig 5. Output after RC filter

- **Buck converter** - The output coming from the rectifier acts as input for the buck converter because it is 10VDC and after bucking the voltage it is converted to 7.4V DC because our charging battery maximum limit is 7.4 volt

$$V_{out} = D * Vin$$

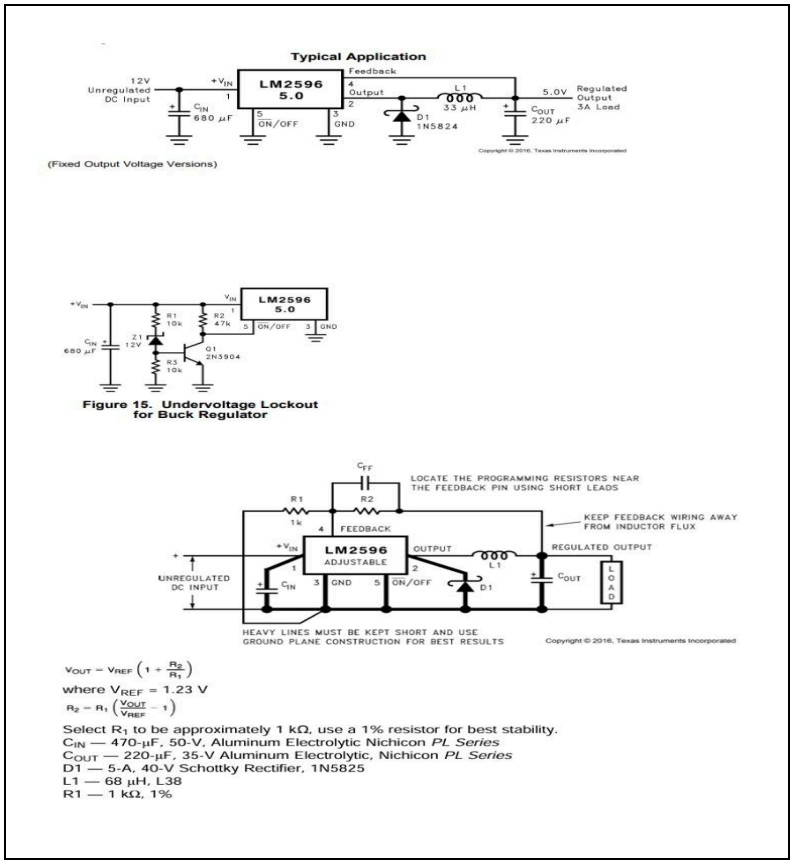


Fig 6.Buck converter IC

Why potentiometer are used in Buck convertors?

Ans. Potentiometers are sometimes used in DC-DC buck converters for adjusting or fine-tuning certain parameters of the converter circuit. Here are some reasons why potentiometers might be used in this context.

1. Output Voltage Adjustment: A potentiometer can be used to adjust the output voltage of the buck converter. By varying the resistance of the potentiometer, the feedback voltage in the converter's control loop can be adjusted, thereby regulating the output voltage to the desired level. This provides flexibility in setting the output voltage to meet the specific requirements of the application.

2. Current Limit Adjustment: In some buck converters, particularly those designed for applications where current limiting is important (such as battery charging circuits), a potentiometer may be used to adjust the current limit threshold. By adjusting the feedback voltage using a potentiometer, the current limit level can be set according to the desired maximum current output.

3. Feedback Network Calibration: Potentiometers can also be used for calibrating the feedback network of the buck converter circuit. This ensures that the converter operates within specified tolerances and accurately regulates the output voltage or current. Fine adjustments to the feedback network can be made using the potentiometer to compensate for component variations or environmental factors.

4. Prototype or Experimental Use: In prototype or experimental setups, potentiometers provide a convenient means of adjusting various parameters of the buck converter circuit during development or testing stages. This allows engineers to quickly iterate and optimize the

performance of the converter without the need to make permanent changes to the circuit layout.

5. User Adjustment: In some applications, such as adjustable power supplies or voltage regulators, potentiometers may be included as user-adjustable controls to allow end-users to customize the output voltage according to their specific requirements or preferences.

Overall, the use of potentiometers in DC-DC buck converters adds flexibility, adjustability, and convenience in setting and fine-tuning important parameters of the converter circuit to meet the needs of different applications and operating conditions.

- **Voltage divider**: A voltage divider is a simple circuit commonly used to reduce voltage levels in electronic circuits. It consists of two resistors connected in series across a voltage source, with the output voltage taken from the connection point between the two resistors. Here's how it works:

  1. Basic Principle: In a voltage divider circuit, the input voltage (V_in) is divided proportionally across the two resistors based on their values. The output voltage (V_out) is then taken from the connection point between the resistors.

  2. Voltage Division Rule: According to the voltage division rule, the output voltage (V_out) of a voltage divider circuit is determined by the ratio of the resistance of the resistor connected to the output terminal (R2) to the total resistance of the voltage divider circuit (R1 + R2), multiplied by the input voltage (V_in).

  3. Reduction of Voltage: By selecting appropriate resistor values for R1 and R2, the output voltage can be reduced to a desired level relative to the input voltage. For example, if R2 is much larger than R1, a smaller

fraction of the input voltage will appear across R2, resulting in a lower output voltage.

4. Applications: Voltage dividers are commonly used in various electronic circuits for purposes such as level shifting, signal scaling, biasing circuits, and sensor interfacing. They provide a simple and cost-effective way to obtain a desired voltage level from a higher input voltage.

5. Considerations: While voltage dividers are simple and versatile, there are some considerations to keep in mind. One important consideration is the loading effect, where the presence of the load connected to the output of the voltage divider affects the output voltage. Additionally, changes in temperature or variations in resistor values can affect the accuracy and stability of the output voltage.

In summary, a voltage divider circuit reduces voltage levels by dividing the input voltage across two resistors in proportion to their values. By selecting appropriate resistor values, the output voltage can be adjusted to a desired level relative to the input voltage, providing flexibility in various electronic circuit applications.

- **Power source**- A constant 5V DC voltage battery for providing the power to the esp32

- **Protection circuit**- 2S 5A 8.4V 18650 Lithium Battery Charger Board Protection Module has the main IC using original “precision” imported components. With overcharge, over-discharge, over current, short circuit protection function, for a variety of different shapes of 3.7V capacity lithium batteries. Using the Japanese precision protection IC, VISHAY, AOS, IR, and other high-quality MOSFET, an FR-4

low-temperature coefficient of the plate, well-designed, comprehensive test. Compact size, suitable for many requirements of high integration, low-cost occasions, can meet the performance requirements of many aspects, to ensure the absolute safety of the battery group.

Protection equipments used:

1. Short circuit protection.
2. Overcharge protection.
3. Over-discharge protection.
4. Over-current protection.

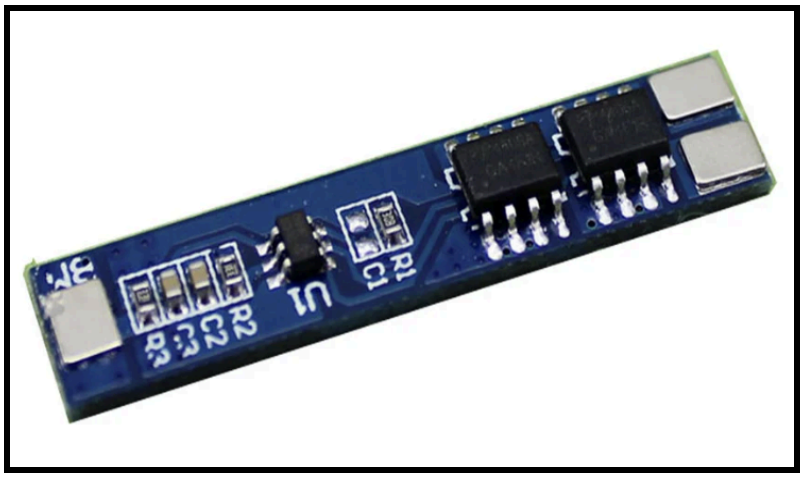


Fig 7.Protection Circuit

Specifications:

Battery	2 Cell
Input Voltage (V)	8.4 ~ 9
Overcharge detection voltage (V)	4.25 Â± 0.025
Over-discharge detection voltage (V)	4.25 Â± 0.025
Over current Protection (A)	7
Working current (A)	5
Operating Temperature (Â°C)	-40 ~ +50
Length (mm):	38.5
Width (mm):	8
Height (mm):	2.3
Weight (g):	1
Shipping Weight	0.01 kg
Shipping Dimensions	15 × 10 × 6 cm

- **Indicator Circuit:** 2S 18650 Lithium Battery Capacity Indicator Module Level Tester LED
  Intelligent electricity quantity display board; slightly push trigger switch, it will auto power off after 3 seconds display.

  LED Indication:

- 25% –>First Light is on (Voltage: 2.1 V)
- 50% –>Second light is on (Voltage: 4.2 V)
- 75% –>Third light is on (Voltage: 6.3 V)
- 100% –>Fourth light is on (Voltage:8.4 V)

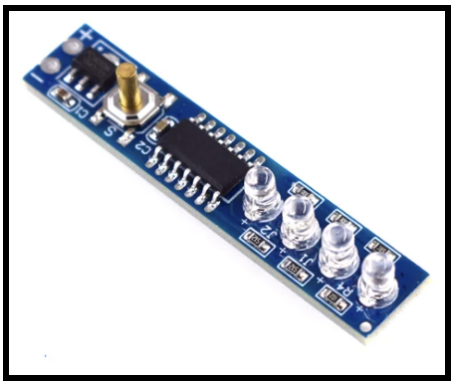


Fig 8. Indicator circuit

Specifications:

Product Type	2S 18650 Lithium Battery
Power Consumption (mA)	2~3
Shipping Weight	0.02 kg
Shipping Dimensions	5 × 1 × 1 cm

Final result:

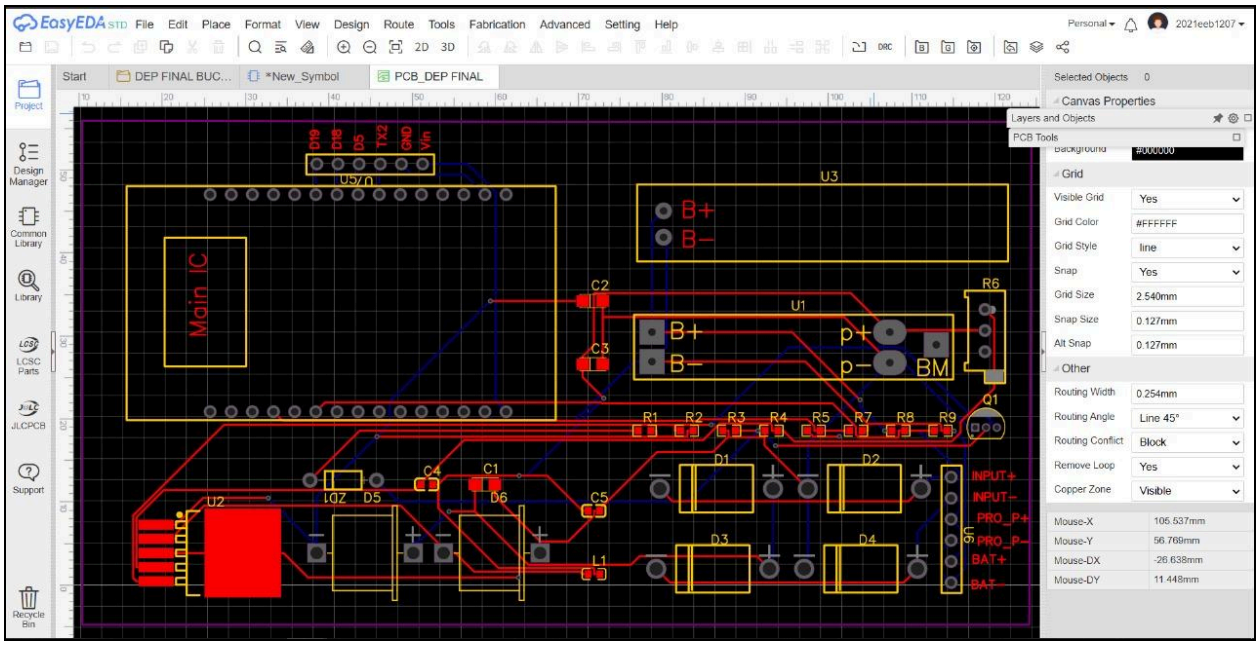


Fig 9. PCB Layout

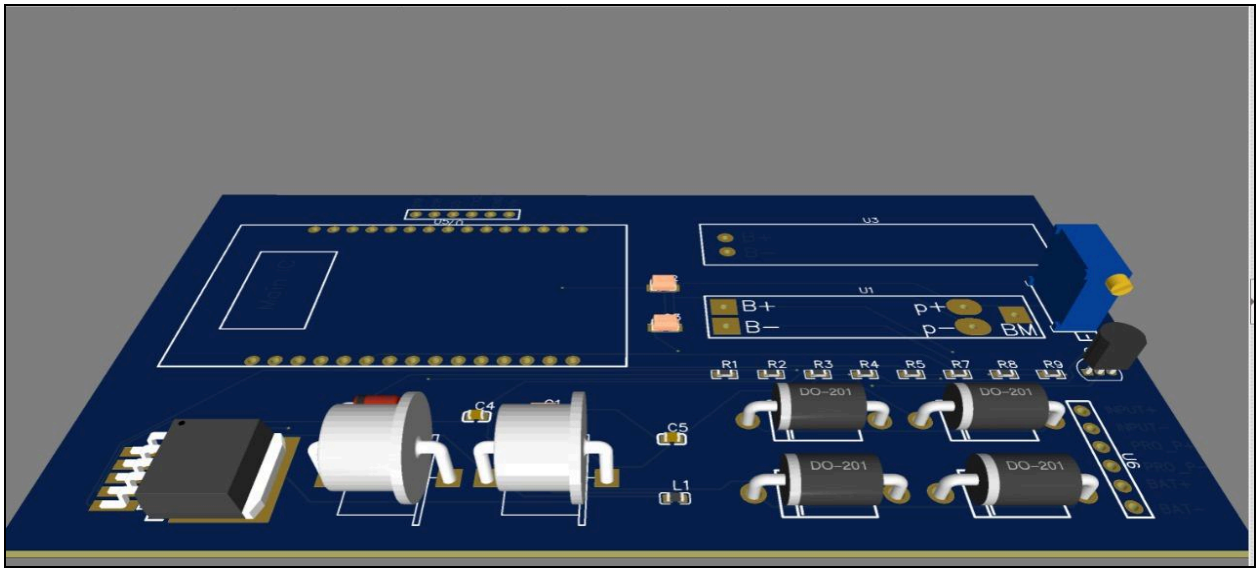


Fig 10. 3D view of the PCB

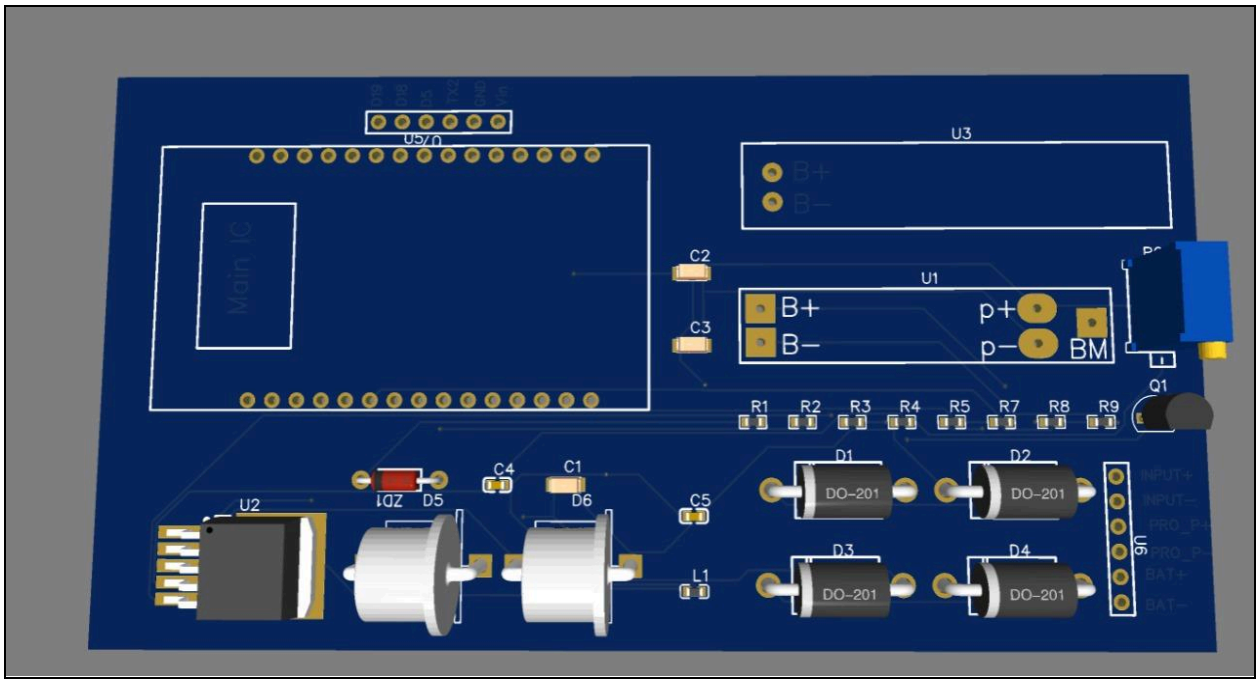


Fig 11. Top view of PCB

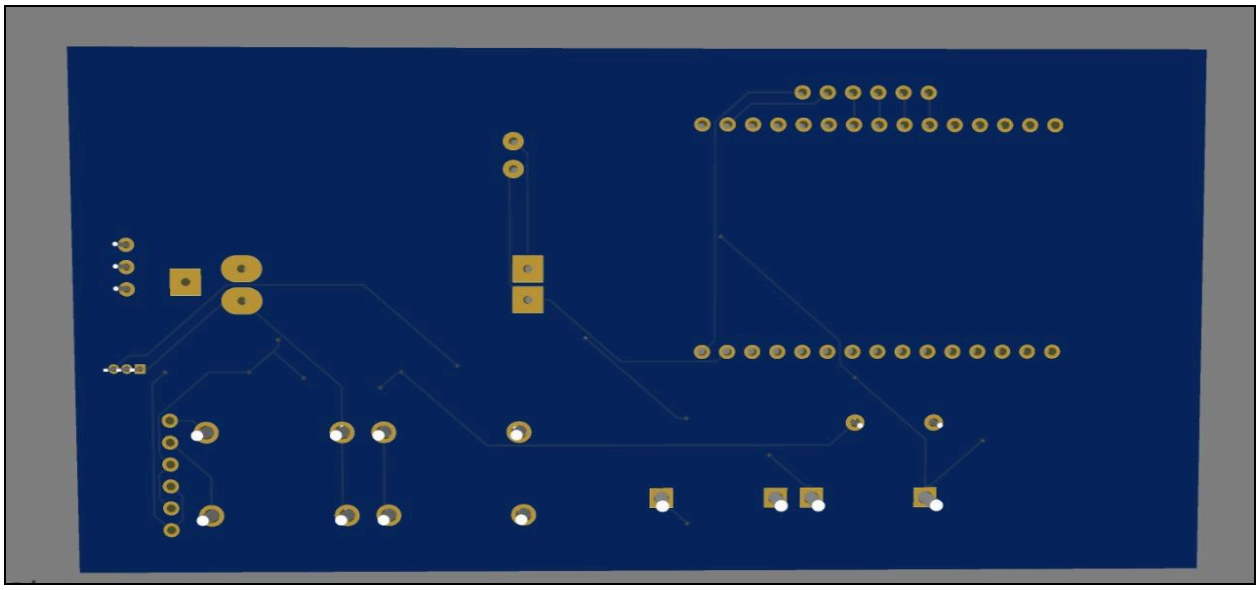


Fig 12. Back side of PCB

CONCLUSION:PCB DESIGN OF RECEIVER

The task involved designing the PCB layout for the receiver side of a charging pad. The design incorporates several key components: a rectifier circuit, filter circuit, buck converter circuit, protection module, and indicator module. The design process was facilitated using the online software EasyEDA due to its extensive component library, easy footprint assignment, and routing options.

Design Platform: The PCB schematics and layout were developed using EasyEDA, chosen for its extensive component library and user-friendly features. These features include easy footprint assignment and seamless integration of additional elements like the protection module and indicator circuit. The software's routing options proved helpful in optimizing the PCB layout. Overall, the PCB design for the charging pad receiver leverages a combination of essential components and user-friendly design tools to achieve efficient and reliable operation.

COMMUNICATION FEEDBACK SYSTEM

This system integrates communication of battery percentage and drone alignment using an ESP32 microcontroller board. The communication aspect utilizes Bluetooth to relay battery information to a connected device(mobile phone), while the alignment function utilizes sensor readings to adjust the drone's orientation.

1. ESP32 Microcontroller Board: The ESP32 is a powerful microcontroller with built-in Wi-Fi and Bluetooth capabilities. It's commonly used in IoT projects due to its versatility and connectivity options.
2. Communication via Bluetooth: The integration of Bluetooth allows your drone to communicate wirelessly with a connected device like a mobile phone. In your case, the drone sends battery percentage information to the mobile phone. This communication is typically achieved using Bluetooth Low Energy (BLE) for efficient data transmission and reduced power consumption.
3. Battery Information Relay: The drone continuously monitors its battery level using an onboard sensor. The ESP32 then sends this battery percentage data over Bluetooth to a mobile phone. This real-time feedback is crucial for drone operators to monitor battery health and make informed decisions about flight duration and landing times.
4. Alignment Function using Sensors: The ESP32 also interfaces with sensors on the drone to gather orientation and alignment data. This data is used to adjust the drone's orientation or alignment in real-time. For example, if the drone detects a deviation from its desired orientation, the microcontroller can send commands to the drone's motors or servos to correct its position.

5. Mobile Phone Interface: The mobile phone acts as a user interface, displaying battery information received from the drone via Bluetooth. This allows the drone operator to monitor the battery status remotely and plan flights accordingly. Additionally, the mobile app could potentially display real-time telemetry data and control options for the drone based on sensor readings.

Hardware used:

ESP32 microcontroller board, jumpers.

we also need an external 5V battery source to power the esp32 4 pins-namely D19,D18,D5,TX2 are available to use in esp32.

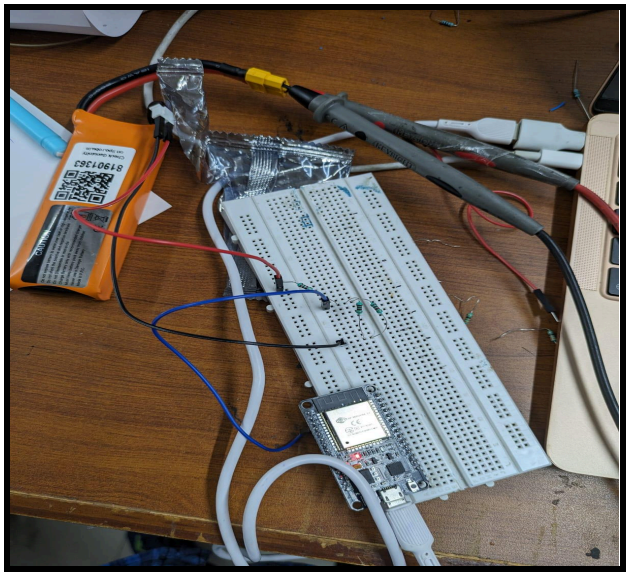


Fig 11. Hardware setupfor esp32

Working principle and algorithm used:

(A). For battery percentage Pins(32)are used of ESP32

Voltage Divider Setup:

- Purpose: The voltage divider circuit is designed to scale down the voltage of the 7.4V battery to a safe range (0V to 3.3V) that can be measured by the ESP32's analog-to-digital converter (ADC) inputs.
- Circuit Design: Use two resistors configured as a voltage divider. The values of these resistors are chosen to scale down the 7.4V battery voltage to a level suitable for the ESP32 ADC input.

2. Calculating Voltage at ESP32 Input:

Voltage Scaling: Calculate the output voltage of the voltage divider using the formula:

$Vout = Vin \times R2 / (R1 + R2)$

$R1$ and $R2$ are the resistances of the two resistors in the divider.

3. ADC Measurement:

- Read ADC Value: Connect the output of the voltage divider to one of the ADC pins on the ESP32.
- Analog Read: Use the ESP32's ADC function to read the voltage level at the ADC pin.

4. Battery Percentage Calculation:

- Mapping Voltage to Percentage: Map the ADC readings to the battery percentage based on the measured voltage.
- Determine Maximum Voltage: Define the maximum voltage (after voltage division) that corresponds to 100% battery charge.

5. Transmitting Data via Bluetooth:

- BluetoothSerial Library: Utilize the BluetoothSerial library in the Arduino IDE to establish a Bluetooth connection between the ESP32 and a mobile device.
- Sending Battery Percentage: Continuously send the calculated battery percentage over the Bluetooth connection to enable real-time monitoring of the battery status on a mobile phone or other Bluetooth-enabled device.

CODE:

```
#include <BluetoothSerial.h>
const float VOLTAGE_DIVIDER_RATIO = 3.31; // Adjust this according to your voltage divider setup
const float MAX_BATTERY_VOLTAGE = 7.4; // Maximum battery voltage
const int batteryPin = 34;
const int numReadings = 20; // Adjust based on readings per second for 2 seconds

int batteryData[numReadings]; // Array to store battery percentages
int dataIndex = 0; // Index for the array

// Define the Bluetooth device name
String deviceName = "ESP32-BT-Master-final";

// Create a BluetoothSerial object
BluetoothSerial SerialBT;

void setup() {
  Serial.begin(115200);
  SerialBT.begin(deviceName); // Initialize Bluetooth with the device name

  Serial.println("ESP32 Master-final is ready to pair.");
}

void loop() {
 // Read battery voltage and calculate percentage
 for (int i = 0; i < numReadings; i++) {
    int sensorValue = analogRead(batteryPin); // Assuming you've connected the voltage divider output to pin A0
    float voltage = (sensorValue * 3.3)/ 4096 *VOLTAGE_DIVIDER_RATIO;
    int batteryPercentage = (voltage / MAX_BATTERY_VOLTAGE) * 100;
    batteryData[dataIndex] = batteryPercentage;
    dataIndex = (dataIndex + 1) % numReadings; // Correctly wrap around the buffer
    delay(100); // Wait for 100ms before the next reading
 }

 // Find the maximum value in the buffer
 int maxVal = batteryData[0];
 for (int i = 0; i < numReadings; i++) {
    if (batteryData[i] > maxVal) {
      maxVal = batteryData[i];
    }
 }

 // Print the maximum battery percentage
 if (maxVal!=0){
    Serial.print("Battery Percentage : ");
    Serial.print(maxVal);
    Serial.println("%");
    String maxV=String(maxVal);
    SerialBT.print("Battery Percentage: "); // Added text before transmitting data
    SerialBT.print(maxV);
    SerialBT.println(); // Add a newline for Bluetooth transmission

 }


 // Reset dataIndex for the next cycle
 dataIndex = 0;
}
```

To measure and transmit average battery voltage wirelessly using an ESP32 microcontroller:

```
#include <BluetoothSerial.h>

const float VOLTAGE_DIVIDER_RATIO = 4.5; // Adjust this according to your voltage divider setup
const int batteryPin = 34;
const int numReadings = 100; // Adjust based on readings per second for 2 seconds

// Define the Bluetooth device name
String deviceName = "ESP32-BT-Master-final2";

// Create a BluetoothSerial object
BluetoothSerial SerialBT;

float batteryData[numReadings]; // Array to store battery percentages
int dataIndex = 0; // Index for the array

void setup() {
    Serial.begin(115200);
    SerialBT.begin(deviceName); // Initialize Bluetooth with the device name

    Serial.println("ESP32 Master-final2 is ready to pair.");
}

// Function to remove ripples and calculate average voltage
float calculateAverageVoltage(float voltageSamples[], int numSamples) {
    int i, j, count;
    float sum = 0.0, avg = 0.0;

    // Remove ripples
    for (i = 0; i < numSamples; i++) {
        float currentVoltage = voltageSamples[i];
        count = 0;
        for (j = 0; j < numSamples; j++) {
            if (voltageSamples[j] >= (currentVoltage - 0.5) && voltageSamples[j] <= (currentVoltage + 0.5)) {
                sum += voltageSamples[j];
                count++;
            }
        }
        if (count > 0) {
            avg += sum / count;
            sum = 0.0;
        }
    }

    // Calculate average voltage
    avg /= numSamples;

    return avg;
}

void loop() {
    for (int i = 0; i < numReadings; i++) {
        int sensorValue = analogRead(batteryPin); // Assuming you've connected the voltage divider output to pin A0
        float voltage = (sensorValue * 3.3) / 4096 * VOLTAGE_DIVIDER_RATIO;
        batteryData[dataIndex] = voltage;
        dataIndex = (dataIndex + 1) % numReadings; // Correctly wrap around the buffer
        delay(100); // Wait for 100ms before the next reading
    }
    float average = calculateAverageVoltage(batteryData, numReadings);
    Serial.println(average);
    SerialBT.print(average);
    SerialBT.println();}
```

(B)For alignment of drone:

Pins(34)are used of ESP32

1. Directional Adjustment:

- Leftward Movement: The drone begins by moving leftward. As it moves, it continuously measures the voltage (or another sensor parameter) that indicates its position or orientation.
- Voltage Feedback: If the voltage increases in the leftward direction, it means the drone is moving closer to the desired alignment point. Therefore, it continues moving left.
- Voltage Decrease: If the voltage decreases, the drone interprets this as moving away from the desired alignment and adjusts its direction accordingly, potentially moving rightward to correct its course.

2. Coordinate Shift:

- Maximizing Left-Right Alignment: The drone continues adjusting leftward until it reaches the maximum voltage value in the left-right axis, indicating it has reached the desired leftward position.
- Focus Shift to Y-Coordinate: Once the left-right alignment is optimized, the drone shifts its focus to the y-coordinate (forward-backward axis) for further alignment adjustments.

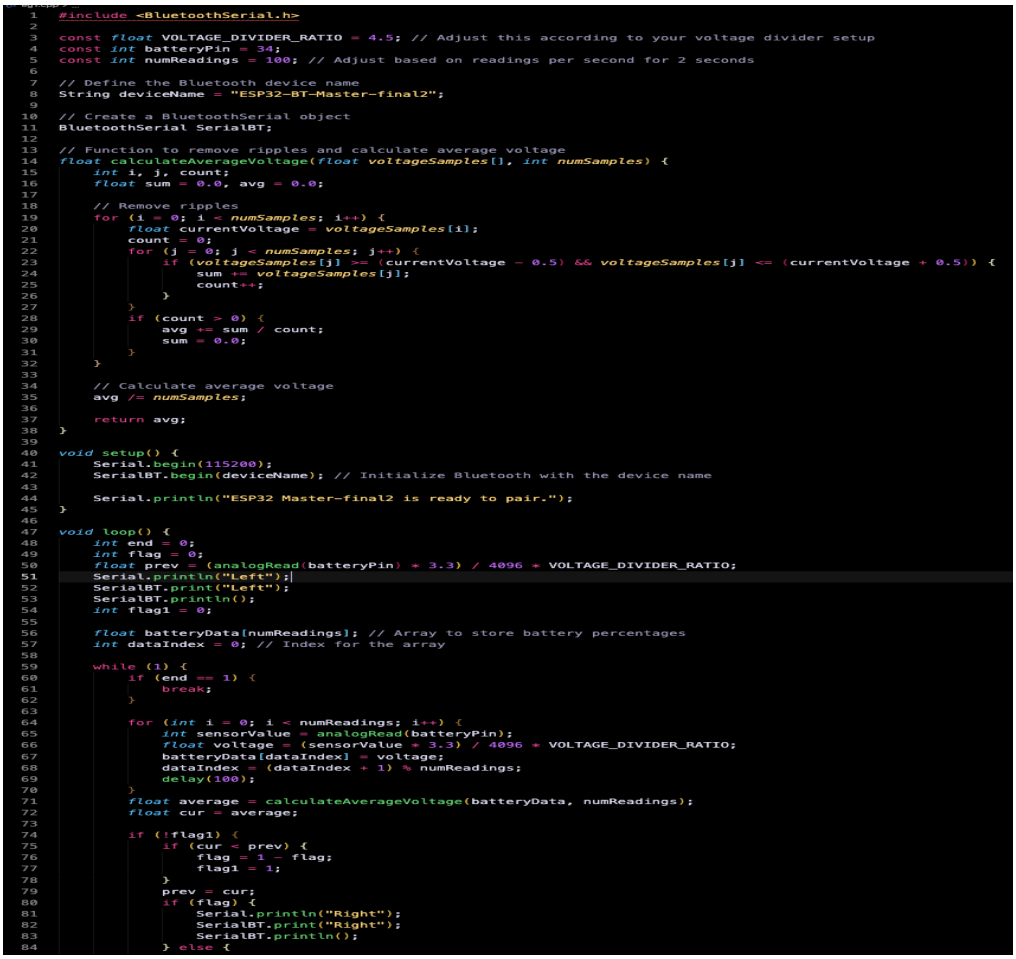
3. Voltage-Based Adjustment:

- Forward and Backward Movement: Similar to the leftward movement, the drone now adjusts its forward or backward movement based on the feedback from voltage measurements.

- Optimizing Alignment: The drone iteratively adjusts its movement direction based on whether the voltage increases or decreases, aiming to reach the maximum voltage value along the y-coordinate.

4. Optimization and Efficiency:

- Iterative Process: This process of directional adjustment and coordinate shift continues iteratively until the drone achieves maximum voltage values in both axes.
- Real-time Feedback: Utilizing real-time voltage feedback allows the drone to make swift and accurate adjustments to its position, ensuring precise alignment for optimal performance.



```
#include <BluetoothSerial.h>

const float VOLTAGE_DIVIDER_RATIO = 4.5; // Adjust this according to your voltage divider setup
const int batteryPin = 34;
const int numReadings = 100; // Adjust based on readings per second for 2 seconds

// Define the Bluetooth device name
String deviceName = "ESP32-BT-Master-final2";

// Create a BluetoothSerial object
BluetoothSerial SerialBT;

// Function to remove ripples and calculate average voltage
float calculateAverageVoltage(float voltageSamples[], int numSamples) {
    int i, j, count;
    float sum = 0.0, avg = 0.0;

    // Remove ripples
    for (i = 0; i < numSamples; i++) {
        float currentVoltage = voltageSamples[i];
        count = 0;
        for (j = 0; j < numSamples; j++) {
            if (voltageSamples[j] >= (currentVoltage - 0.5) && voltageSamples[j] <= (currentVoltage + 0.5)) {
                sum += voltageSamples[j];
                count++;
            }
        }
        if (count > 0) {
            avg += sum / count;
            sum = 0.0;
        }
    }

    // Calculate average voltage
    avg /= numSamples;

    return avg;
}

void setup() {
    Serial.begin(115200);
    SerialBT.begin(deviceName); // Initialize Bluetooth with the device name

    Serial.println("ESP32 Master-final2 is ready to pair.");
}

void loop() {
    int end = 0;
    int flag = 0;
    float prev = (analogRead(batteryPin) * 3.3) / 4096 * VOLTAGE_DIVIDER_RATIO;
    Serial.println("Left");
    SerialBT.print("Left");
    SerialBT.println();
    int flag1 = 0;

    float batteryData[numReadings]; // Array to store battery percentages
    int dataIndex = 0; // Index for the array

    while (1) {
        if (end == 1) {
            break;
        }

        for (int i = 0; i < numReadings; i++) {
            int sensorValue = analogRead(batteryPin);
            float voltage = (sensorValue * 3.3) / 4096 * VOLTAGE_DIVIDER_RATIO;
            batteryData[dataIndex] = voltage;
            dataIndex = (dataIndex + 1) % numReadings;
            delay(100);
        }
        float average = calculateAverageVoltage(batteryData, numReadings);
        float cur = average;

        if (!flag1) {
            if (cur < prev) {
                flag = 1 - flag;
                flag1 = 1;
            }
            prev = cur;
            if (flag) {
                Serial.println("Right");
                SerialBT.print("Right");
                SerialBT.println();
            } else {
```

```
            } else {
                Serial.println("Left");
                SerialBT.print("Left");
                SerialBT.println();
            }
        } else {
            if (flag) {
                Serial.println("Right");
                SerialBT.print("Right");
                SerialBT.println();
            } else {
                Serial.println("Left");
                SerialBT.print("Left");
                SerialBT.println();
            }
            prev = cur;

            for (int i = 0; i < numReadings; i++) {
                int sensorValue = analogRead(batteryPin);
                float voltage = (sensorValue * 3.3) / 4096 * VOLTAGE_DIVIDER_RATIO;
                batteryData[dataIndex] = voltage;
                dataIndex = (dataIndex + 1) % numReadings;
                delay(100);
            }
            average = calculateAverageVoltage(batteryData, numReadings);
            cur = average;

            if (cur > prev) {
                flag1 = 0;
            } else {
                flag = 1 - flag;
                if (flag) {
                    Serial.println("Right");
                    SerialBT.print("Right");
                    SerialBT.println();
                } else {
                    Serial.println("Left");
                    SerialBT.print("Left");
                    SerialBT.println();
                }

                Serial.println("Now We will Do for forward and backward");
                SerialBT.print("Now We will Do for forward and backward");
                SerialBT.println();

                break;
            }
        }
    }

    flag = 0;
    prev = (analogRead(batteryPin) * 3.3) / 4096 * VOLTAGE_DIVIDER_RATIO;
    Serial.println("Backward");
    SerialBT.print("Backward");
    SerialBT.println();
    flag1 = 0;

    while (1) {
        for (int i = 0; i < numReadings; i++) {
            int sensorValue = analogRead(batteryPin);
            float voltage = (sensorValue * 3.3) / 4096 * VOLTAGE_DIVIDER_RATIO;
            batteryData[dataIndex] = voltage;
            dataIndex = (dataIndex + 1) % numReadings;
            delay(100);
        }

        float average = calculateAverageVoltage(batteryData, numReadings);
        float cur = average;

        if (!flag1) {
            if (cur < prev) {
                flag = 1 - flag;
                flag1 = 1;
            }
            prev = cur;
            if (flag) {
                Serial.println("Forward");
                SerialBT.print("Forward");
                SerialBT.println();
            } else {
                Serial.println("Backward");
                SerialBT.print("Backward");
                SerialBT.println();
            }
```

```
166                 SerialBT.println();
167             }
168         } else {
169             if (flag) {
170                 Serial.println("Forward");
171                 SerialBT.print("Forward");
172                 SerialBT.println();
173             } else {
174                 Serial.println("Backward");
175                 SerialBT.print("Backward");
176                 SerialBT.println();
177             }
178             prev = cur;
179 
180             for (int i = 0; i < numReadings; i++) {
181                 int sensorValue = analogRead(batteryPin);
182                 float voltage = (sensorValue * 3.3) / 4096 * VOLTAGE_DIVIDER_RATIO;
183                 batteryData[dataIndex] = voltage;
184                 dataIndex = (dataIndex + 1) % numReadings;
185                 delay(100);
186             }
187             average = calculateAverageVoltage(batteryData, numReadings);
188             cur = average;
189 
190             if (cur > prev) {
191                 flag1 = 0;
192             } else {
193                 flag = 1 - flag;
194                 if (flag) {
195                     Serial.println("Forward");
196                     SerialBT.print("Forward");
197                     SerialBT.println();
198                 } else {
199                     Serial.println("Backward");
200                     SerialBT.print("Backward");
201                     SerialBT.println();
202                 }
203 
204                 Serial.println("Now Land the Drone");
205                 SerialBT.print("Now Land the Drone");
206                 SerialBT.println();
207                 end = 1;
208 
209                 break;
210                 return;
211             }
212         }
213     }
214 
```

Testing our system:

1. Battery Percentage Testing:

- Battery Type: Using a 2S LiPo battery for testing provides a practical scenario as it's a common choice for drones.
- Comparative Measurement: Comparing the battery percentage readings obtained from your ESP32-based system against measurements from a multimeter serves as a validation step.
- Accuracy Verification: Ensuring that the calculated battery percentage aligns closely with the multimeter readings confirms the accuracy of your voltage measurement and scaling setup.

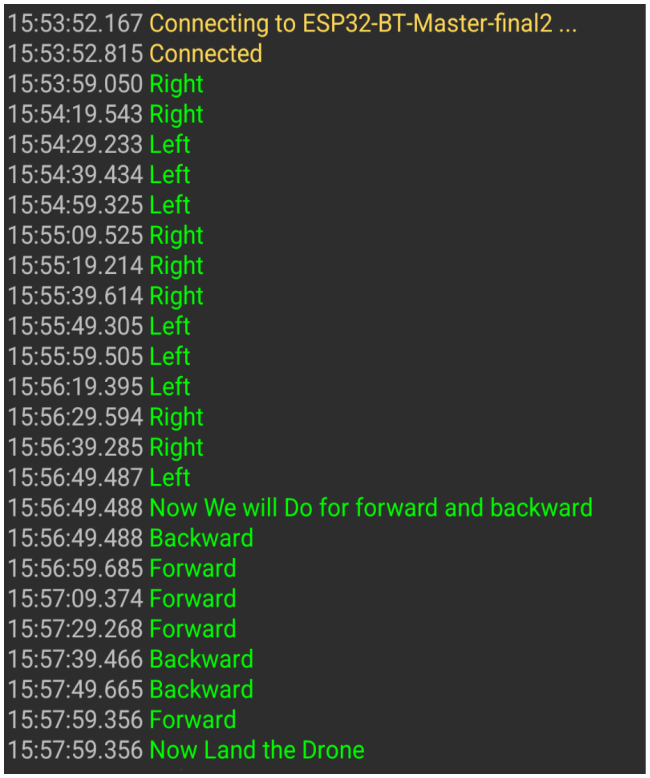
2. Alignment Validation:

- Real-time Observation: Monitoring the voltage feedback in real-time during alignment testing allows for immediate verification of the drone's positioning accuracy.
- Bluetooth Data Transmission: Displaying the voltage feedback data on a connected mobile phone via Bluetooth enables convenient and continuous observation of alignment adjustments.
- Iterative Testing: Iteratively adjusting the drone's position based on voltage feedback and observing the results ensures that the alignment process functions reliably and effectively.

3. Data Monitoring and Validation:

- Mobile Phone Interface: The use of a mobile phone as a monitoring device allows for easy visualization of critical data (such as battery percentage and alignment feedback) during testing.
- Bluetooth Connectivity: Testing the Bluetooth data transmission ensures that real-time data updates are reliably received and displayed on the mobile phone interface.
- Validation Criteria: Establishing validation criteria for both battery percentage accuracy and alignment precision helps in assessing the overall performance of your system.

Output of alignment algorithm on Serial Bluetooth Terminal:



Conclusion :

In conclusion, our project has achieved significant milestones in designing and integrating the receiver layout for our drone's charging system. Through meticulous circuit design, we have effectively managed to minimize ripples using capacitors and resistors, thereby safeguarding the battery's health during charging. This integration of various essential components onto a single PCB marks a key achievement in streamlining our system's efficiency and compactness.

Furthermore, the implementation of algorithms on the ESP32 microcontroller has been a pivotal success. These algorithms serve dual purposes: accurately calculating the drone battery's percentage and ensuring precise drone alignment. Leveraging the Arduino IDE and employing C++ programming, we have developed robust algorithms that enhance the functionality and reliability of our drone charging and alignment system.

Overall, our project not only demonstrates technical prowess in hardware design and microcontroller programming but also underscores our commitment to developing innovative solutions for drone technology applications.