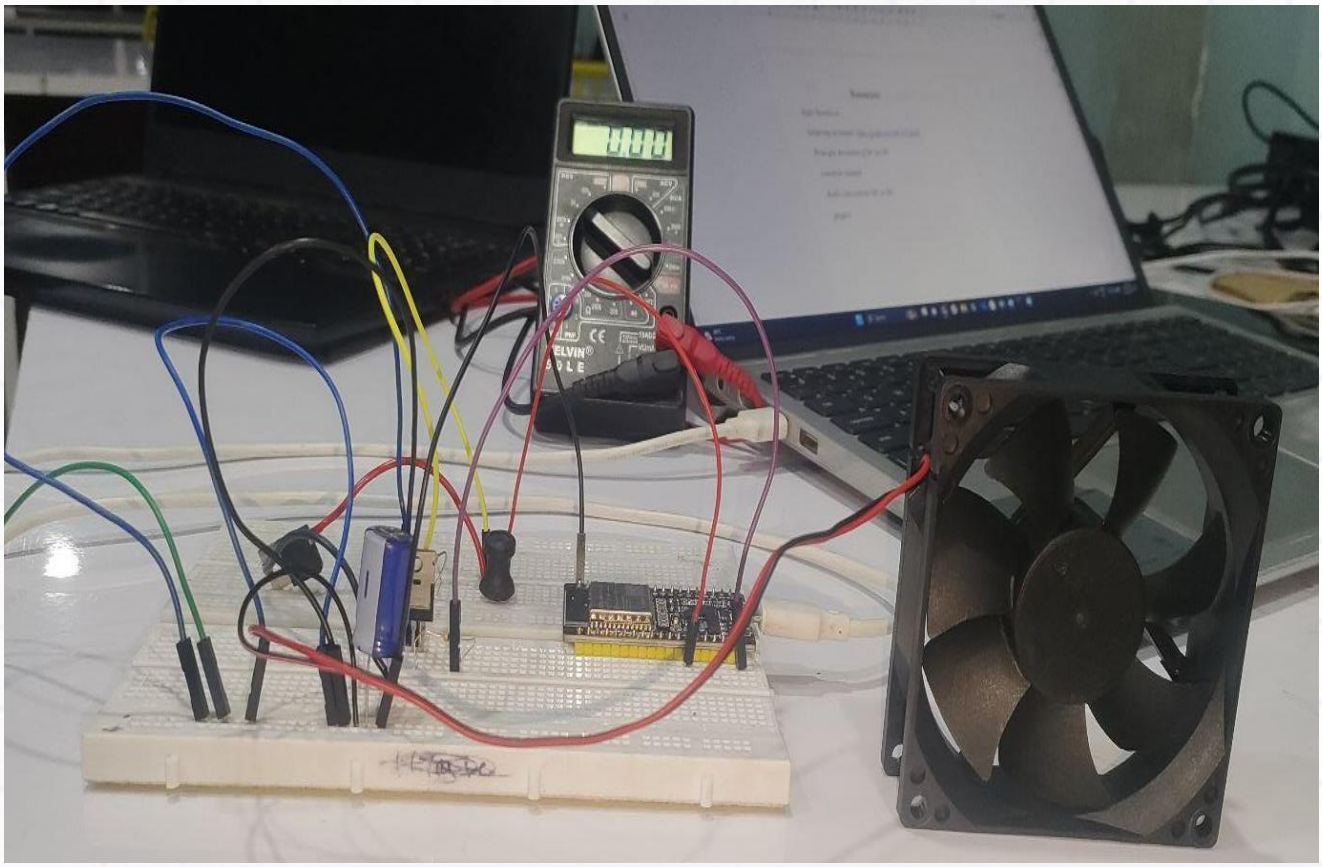
Build a simple low power DC-DC boost Converter



Introduction to the course:	Very often a low power boost converter is used to convert low voltage to a higher voltage is required in a system. This build exercise is to learn to build one such simple converter, using an open loop.
What does this course aim to achieve?	• To drive an LED • To drive a single lithium-ion cell from low volt to high volt. • To drive automotive device such as a fan from 5V to 12 V
What is being built in this course:	Boost converter (5V DC input to 12V DC output).
How is it being tested:	Based on the design requirements, the components are mounted on dot board. With the required input supply, the output results are monitored using multimeter.
Course Prerequisites	• Principle of boost converter • Basics of RLC circuits • Soldering techniques

Index

Prerequisites

Topic	Resource
Soldering technique	https://youtu.be/oqV2xU1fee8
Fundamental understanding of Dc-Dc Boost Conversion	Online Resources

Aim

Very often a low power boost converter used to convert some low voltage to a higher voltage is required in a system. This build exercise is to learn to build one such simple converter, using an open loop.

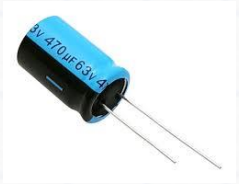
Components

Components	Specification	Cost Per Quantity	Quantity
PCB Dot board	10*15cm	40	1
Inductor	1mH	50	1
Capacitor	100-470uf (63V)	25	1
MOSFET	IRLZ24NPBF OR NVMFS6H818NL	160	1
Gate Driver IC	TC4427	50	1
Schottky	STPSC406D	150	1

Diode	OR NXPSC046506Q OR WNSC6D04650Q		
DC Fan (Load)	12V- 0.27A	95	1
Resistor	22,2.2K ohm (Through hole)	15	Each one
Soldering station (Lead, flux, IP)	Optional	250	1
Power supply	5V 2A Phone Adapter	100	1
Connecting wires	26AWG	20	1(metre)
Bread board	830 pins	50	1
Digital multimeter		175	1
Jumper wire Male to male		15	6
ESP32 C-6 Module	-	-	1
USB cable	Type B	200	1

Note:

- All the components are reusable after desoldering.
- Measure the resistor values by multimeter.



Capacitor

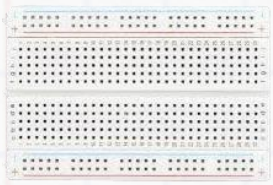
Inductor

MOSFET

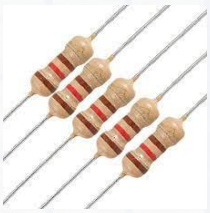
Jumper Wires

ESP32 C-6 Module

DC Fan



Breadboard

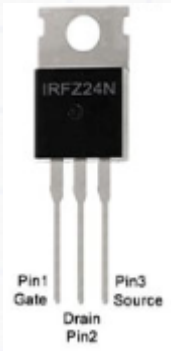


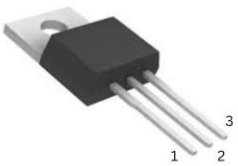
Resistor

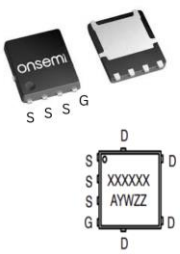
Diode

MOSFET PINOUT:

- Pin 1 = gate
- Pin 2 = drain
- Pin 3 = source







Note: While choosing the MOSFET, check from the datasheet that MOSFET Vgs should be less than 2.5V.

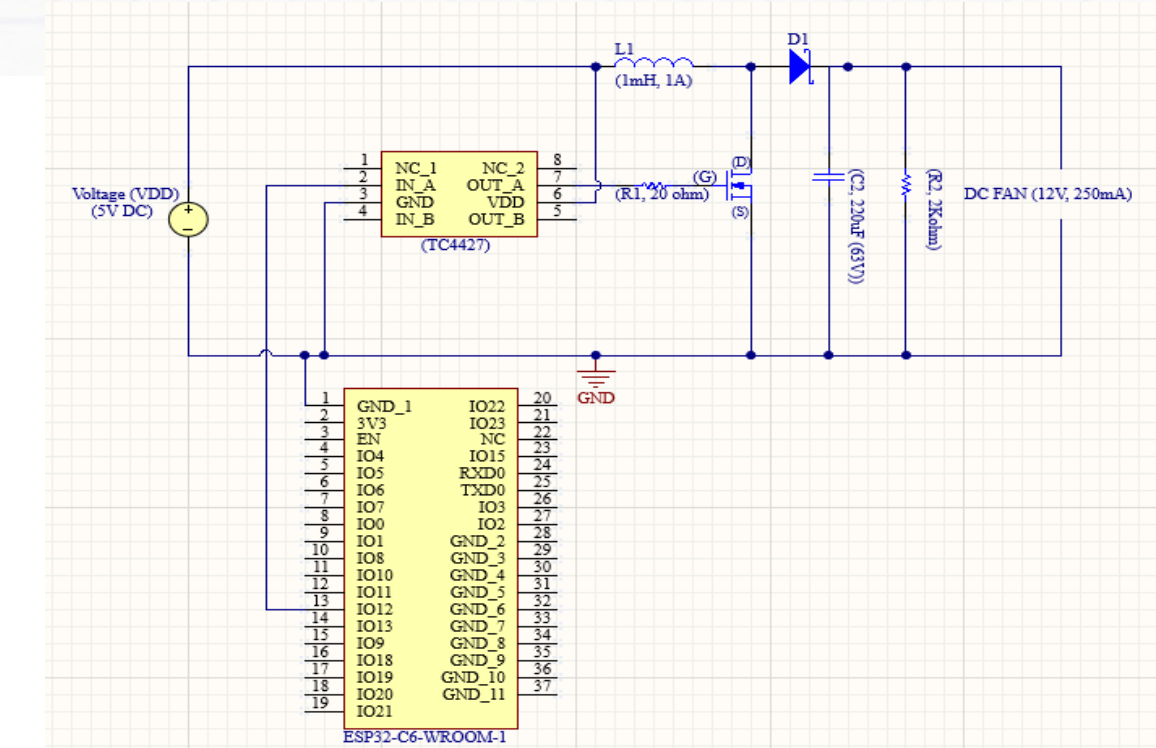
Connections

Circuit Diagram:

NOTE: First make the circuit diagram in the Bread board and verify the connection properly once you got the output then only go for soldering. Do not touch the tip of the soldering iron, its temperature can be as high as 380ºC and can cause severe burns. Keep the cleaning sponge wet when soldering.

Detailed Connection Steps

Step 1: Connect or solder inductor, diode, capacitor, MOSFET and resistor on PCB dot board as per the circuit diagram fig1.
Step 2: Set the power supply voltage to 5V or take a 5V phone adapter.
Step 3: Connect or solder positive side (A) of power supply to Inductor leg and negative side of the power supply to the ground (G).



Step 4: Connect or solder another inductor leg(B) to the drain of the MOSFET(D).

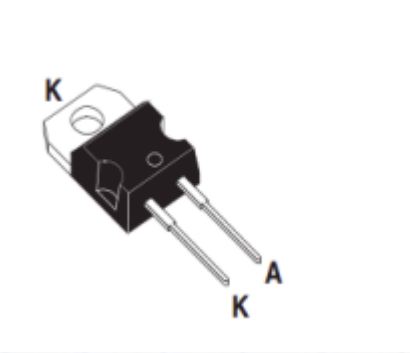
- Connect or Solder one male to male jumper wire to the MOSFET gate and connect another side of jumper wire to D33 pin of ESP 32
- Connect or Solder one male to male jumper wire to the MOSFET source and connect another side of jumper wire to GND pin of ESP32.

Step 5: Connect 1K resistor between gate of the MOSFET and D33 pin of ESP 32.

Step 6: Connect 340K resistor between gate of the MOSFET and GND of the Bread board.

Step 7: Connect 2K resistor parallel to the capacitor.

Step 8: Connect or Solder anode of the diode to point B. And cathode C of the diode to the positive of the capacitor.



K = cathode

A = anode

- Negative side of capacitor to the ground (G).
- Connect (C) with one leg of resistor and another with (G).

Step 9: Connect or solder two male to male jumper wire across the capacitor and connect to the 12V DC load fan. K = cathode A = anode DC to DC Boost Converter 11

Step 10: Switch ON the power supply and measure the output voltage using digital multimeter across the capacitor.

Step 11: To change the duty cycle, copy the below code and run it in Arduino IDE

Software

Launching the IDE for our project - Code

1. Connect Your ESP32 Module

1. Connect your ESP32 C-6 Module to your computer using a USB cable.
2. Go to Tools > Board > ESP32C6 Dev Module.
3. Go to Tools > Port and select the COM port to which your ESP32C6 is connected.

2. Prepare the Hardware

Connect your LED matrix to the ESP32C6. For the example code, ensure the CS pin of the matrix is connected to GPIO 5 of the ESP32C6.

3. Upload the Code

Copy the provided code into a new sketch in the Arduino IDE.

Click the upload button (right arrow) to compile and upload the code to your ESP32.

4. Pairing the ESP32 via Bluetooth

Once the code is uploaded, open the Serial Monitor from Tools > Serial Monitor and set the baud rate to 115200.

Pair your ESP32 with your computer or smartphone. The device name should be ESP32test.

Once paired, you can send text messages to the ESP32 via a Bluetooth terminal app on your smartphone or a serial terminal on your computer.

Once the code is uploaded to ESP32, the LED matrix will start displaying the text in various alignments.

Code

"Open the .ino file which was attached in the folder and write a code by the below instruction."

Block 1: Define Pin Connections and Constants

Explanation

Define pin connections and constants:

- **pwmPin** is set to GPIO 12, which is connected to the MOSFET gate.
- **freq** is the PWM frequency set to 50 kHz.
- **ledChannel** is the PWM channel, here channel 0 is used.
- **resolution** is the PWM resolution set to 8 bits (values range from 0 to 255).
- **period** is set to 255, which corresponds to the maximum value for 8-bit resolution (2^8 - 1).

```
// Define pin connections
const int pwmPin = 12;      // GPIO pin connected to MOSFET gate
const int freq = 50000;     // PWM frequency (50 kHz)
const int ledChannel = 0;   // PWM channel (using channel 0)
const int resolution = 8;   // PWM resolution (8-bit, values range from 0 to 255)

const int period = 255;     // For 8-bit resolution, the period is 255 (2^8 - 1)
```

Block 2: Setup Function

Explanation

Setup function to configure PWM and serial communication:

- **ledcSetup**(ledChannel, freq, resolution); configures the PWM channel with the specified frequency and resolution.
- **ledcAttachPin**(pwmPin, ledChannel); attaches the PWM channel to the specified **GPIO pin** (pwmPin).
- **Serial.begin(115200);** initializes serial communication at a baud rate of 115200 for debugging.

Copy the below code that is inside in the void setup function and paste it on your code void setup function.

```
void setup() {
  // Configure the PWM channel with the specified frequency and resolution
  ledcSetup(ledChannel, freq, resolution);

  // Attach the PWM channel to the specified GPIO pin
  ledcAttachPin(pwmPin, ledChannel);

  // Initialize serial communication for debugging purposes
  Serial.begin(115200);
}
```

Block 3: Loop Function and Duty Cycle Functions

Explanation

Loop function:

The **loop()** function is empty and does not perform any actions in this example.

Function to calculate duty value based on desired percentage:

- **calculateDuty(float dutyPercent)** takes a percentage as input.
- **return round(period * (dutyPercent / 100.0));** converts the percentage to a duty cycle value between 0 and the period (255).

Function to set the duty cycle:

- **Set_Duty(int dutyValue)** takes a duty cycle value as input.
- **if (dutyValue >= 0 && dutyValue <= period)** checks if the duty value is within the valid range (0 to 255).
- **ledcWrite(ledChannel, dutyValue)**; sets the duty cycle for the specified PWM channel to the provided duty value.

Copy the below code that is inside in the void set_Duty and paste it on your code.

```
void loop() {
  // The main loop is empty and does not perform any actions
}

// Function to calculate duty value based on desired percentage
int calculateDuty(float dutyPercent) {
  // Calculate the duty cycle value by converting the percentage to a value between
0 and period
  return round(period * (dutyPercent / 100.0));
}

// Function to set the duty cycle
void Set_Duty(int dutyValue) {
```

```
  // Check if the duty value is within the valid range (0 to period)
  if (dutyValue >= 0 && dutyValue <= period) {
    // Write the duty value to the PWM channel
    ledcWrite(ledChannel, dutyValue);
  }
}
```

After completing all the steps verify and upload a code to the esp32.

Write the following command to change the duty cycle and run the code.

```
Set_Duty(70);
```

- The Set_Duty( ) function allows you to change the duty cycle.
  To change
  Duty = period ×(%percentage of duty cycle)
  Example: Change duty cycle to 58%
  Duty = 200*(58/100) = 116
  Set_Duty(116); // for 58% duty cycle

Hurray you have learnt how to drive a boost circuit!!!!!!!

Tasks:

For open loop

1. On the same boost converter change the duty cycle 50%, 60% and 70% and measure the voltage.
2. Measure the output voltage keeping R =12ohm. Compute the expected voltage using mathematical formula for each of the duty cycle. Check measure voltage and compute voltage are matching.

Exercise:

1. Calculate the losses and the efficiency using the formula.
2. Make a graph
    - between ripple voltage and duty
    - between ripple voltage and load