#include <Fuzzy.h>

// Create a fuzzy instance

Fuzzy\* fuzzy = new Fuzzy();

// Input variables (measured voltages)

float solarInput;

float hydroInput;

float gridInput;

// Fuzzy input and output objects

FuzzyInput\* solar = new FuzzyInput(1);

FuzzyInput\* hydro = new FuzzyInput(2);

FuzzyInput\* grid = new FuzzyInput(3);

FuzzyOutput\* sourceControl = new FuzzyOutput(1); // Output to control which source to use

void setup() {

Serial.begin(9600);

// Define fuzzy sets for Solar input (low, medium, high)

FuzzySet\* solarLow = new FuzzySet(0, 0, 5, 7); // Example voltage ranges

FuzzySet\* solarMedium = new FuzzySet(6, 8, 12, 14);

FuzzySet\* solarHigh = new FuzzySet(13, 15, 18, 20);

solar->addFuzzySet(solarLow);

solar->addFuzzySet(solarMedium);

solar->addFuzzySet(solarHigh);

fuzzy->addFuzzyInput(solar);

// Define fuzzy sets for Hydro input (low, medium, high)

FuzzySet\* hydroLow = new FuzzySet(0, 0, 5, 7);

FuzzySet\* hydroMedium = new FuzzySet(6, 8, 12, 14);

FuzzySet\* hydroHigh = new FuzzySet(13, 15, 18, 20);

hydro->addFuzzySet(hydroLow);

hydro->addFuzzySet(hydroMedium);

hydro->addFuzzySet(hydroHigh);

fuzzy->addFuzzyInput(hydro);

// Define fuzzy sets for Grid input (low, medium, high)

FuzzySet\* gridLow = new FuzzySet(0, 0, 5, 7);

FuzzySet\* gridMedium = new FuzzySet(6, 8, 12, 14);

FuzzySet\* gridHigh = new FuzzySet(13, 15, 18, 20);

grid->addFuzzySet(gridLow);

grid->addFuzzySet(gridMedium);

grid->addFuzzySet(gridHigh);

fuzzy->addFuzzyInput(grid);

// Define fuzzy sets for Source control (1 = Solar, 2 = Hydro, 3 = Grid)

FuzzySet\* useSolar = new FuzzySet(0, 0, 1, 1);

FuzzySet\* useHydro = new FuzzySet(1, 1, 2, 2);

FuzzySet\* useGrid = new FuzzySet(2, 2, 3, 3);

sourceControl->addFuzzySet(useSolar);

sourceControl->addFuzzySet(useHydro);

sourceControl->addFuzzySet(useGrid);

fuzzy->addFuzzyOutput(sourceControl);

// Fuzzy rules: If solar is high, use solar

FuzzyRuleAntecedent\* ifSolarHigh = new FuzzyRuleAntecedent();

ifSolarHigh->joinSingle(solarHigh);

FuzzyRuleConsequent\* thenUseSolar = new FuzzyRuleConsequent();

thenUseSolar->addOutput(useSolar);

FuzzyRule\* fuzzyRule1 = new FuzzyRule(1, ifSolarHigh, thenUseSolar);

fuzzy->addFuzzyRule(fuzzyRule1);

// Rule: If hydro is high and solar is medium or low, use hydro

FuzzyRuleAntecedent\* ifHydroHighAndSolarNotHigh = new FuzzyRuleAntecedent();

ifHydroHighAndSolarNotHigh->joinWithAND(hydroHigh, solarLow); // or solarMedium

FuzzyRuleConsequent\* thenUseHydro = new FuzzyRuleConsequent();

thenUseHydro->addOutput(useHydro);

FuzzyRule\* fuzzyRule2 = new FuzzyRule(2, ifHydroHighAndSolarNotHigh, thenUseHydro);

fuzzy->addFuzzyRule(fuzzyRule2);

// Rule: If both solar and hydro are low, use grid

FuzzyRuleAntecedent\* ifSolarLowAndHydroLow = new FuzzyRuleAntecedent();

ifSolarLowAndHydroLow->joinWithAND(solarLow, hydroLow);

FuzzyRuleConsequent\* thenUseGrid = new FuzzyRuleConsequent();

thenUseGrid->addOutput(useGrid);

FuzzyRule\* fuzzyRule3 = new FuzzyRule(3, ifSolarLowAndHydroLow, thenUseGrid);

fuzzy->addFuzzyRule(fuzzyRule3);

}

void loop() {

// Read the inputs (voltage measurements)

solarInput = analogRead(A0); // Example pin for solar input

hydroInput = analogRead(A1); // Example pin for hydro input

gridInput = analogRead(A2); // Example pin for grid input

// Map the inputs to the corresponding fuzzy sets

fuzzy->setInput(1, solarInput);

fuzzy->setInput(2, hydroInput);

fuzzy->setInput(3, gridInput);

// Execute fuzzy logic

fuzzy->fuzzify();

// Get the output (which power source to use)

float source = fuzzy->defuzzify(1);

if (source < 1.5) {

// Use Solar

digitalWrite(SOLAR\_MOSFET\_PIN, HIGH);

digitalWrite(HYDRO\_MOSFET\_PIN, LOW);

digitalWrite(GRID\_MOSFET\_PIN, LOW);

} else if (source < 2.5) {

// Use Hydro

digitalWrite(SOLAR\_MOSFET\_PIN, LOW);

digitalWrite(HYDRO\_MOSFET\_PIN, HIGH);

digitalWrite(GRID\_MOSFET\_PIN, LOW);

} else {

// Use Grid

digitalWrite(SOLAR\_MOSFET\_PIN, LOW);

digitalWrite(HYDRO\_MOSFET\_PIN, LOW);

digitalWrite(GRID\_MOSFET\_PIN, HIGH);

}

delay(1000); // Update every second

}

TERST

// Fuzzy Input for SOC (based on battery Vbat)

FuzzyInput\* soc = new FuzzyInput(4);

// Define SOC fuzzy sets

FuzzySet\* socLow = new FuzzySet(0, 0, 30, 40); // 0% to 30% SOC

FuzzySet\* socMedium = new FuzzySet(30, 40, 70, 80); // 30% to 70% SOC

FuzzySet\* socHigh = new FuzzySet(70, 80, 100, 100); // 70% to 100% SOC

soc->addFuzzySet(socLow);

soc->addFuzzySet(socMedium);

soc->addFuzzySet(socHigh);

fuzzy->addFuzzyInput(soc);

// Rule: If solar is high and SOC is low, prioritize solar

FuzzyRuleAntecedent\* ifSolarHighAndSocLow = new FuzzyRuleAntecedent();

ifSolarHighAndSocLow->joinWithAND(solarHigh, socLow);

FuzzyRuleConsequent\* thenUseSolarForLowSoc = new FuzzyRuleConsequent();

thenUseSolarForLowSoc->addOutput(useSolar);

FuzzyRule\* fuzzyRule4 = new FuzzyRule(4, ifSolarHighAndSocLow, thenUseSolarForLowSoc);

fuzzy->addFuzzyRule(fuzzyRule4);

// Rule: If SOC is medium and hydro is high, use hydro

FuzzyRuleAntecedent\* ifHydroHighAndSocMedium = new FuzzyRuleAntecedent();

ifHydroHighAndSocMedium->joinWithAND(hydroHigh, socMedium);

FuzzyRuleConsequent\* thenUseHydroForMediumSoc = new FuzzyRuleConsequent();

thenUseHydroForMediumSoc->addOutput(useHydro);

FuzzyRule\* fuzzyRule5 = new FuzzyRule(5, ifHydroHighAndSocMedium, thenUseHydroForMediumSoc);

fuzzy->addFuzzyRule(fuzzyRule5);

// Rule: If SOC is high, prioritize hydro or limit charging

FuzzyRuleAntecedent\* ifSocHigh = new FuzzyRuleAntecedent();

ifSocHigh->joinSingle(socHigh);

FuzzyRuleConsequent\* thenUseHydroForHighSoc = new FuzzyRuleConsequent();

thenUseHydroForHighSoc->addOutput(useHydro); // Use hydro or stop charging

FuzzyRule\* fuzzyRule6 = new FuzzyRule(6, ifSocHigh, thenUseHydroForHighSoc);

fuzzy->addFuzzyRule(fuzzyRule6);

float calculateSOC(float vbat) {

if (vbat >= 13.7) return 100;

else if (vbat >= 13.48) return 90;

else if (vbat >= 13.26) return 80;

else if (vbat >= 13.04) return 70;

else if (vbat >= 12.82) return 60;

else if (vbat >= 12.6) return 50;

else if (vbat >= 12.38) return 40;

else if (vbat >= 12.16) return 30;

else if (vbat >= 11.94) return 20;

else if (vbat >= 11.72) return 10;

else return 0; // SOC is 0% for Vbat <= 11.5

}

void loop() {

// Read battery voltage (Vbat) and calculate SOC

float vbat = analogRead(A3); // Example pin for Vbat

float socValue = calculateSOC(vbat);

// Set SOC input in fuzzy logic

fuzzy->setInput(4, socValue);

// Other input and fuzzy logic processing

// ...

}

**1. State of Charge (SOC) Membership Function Graph**

For SOC, we'll use three fuzzy sets: **Low SOC**, **Medium SOC**, and **High SOC**. Based on the table you provided, SOC is categorized from 0% to 100%, with specific transition ranges.

**Fuzzy Set Ranges:**

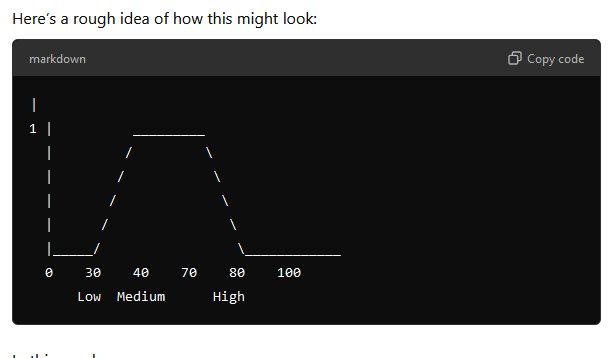
* **Low SOC**: 0% - 30%
* **Medium SOC**: 30% - 70%
* **High SOC**: 70% - 100%

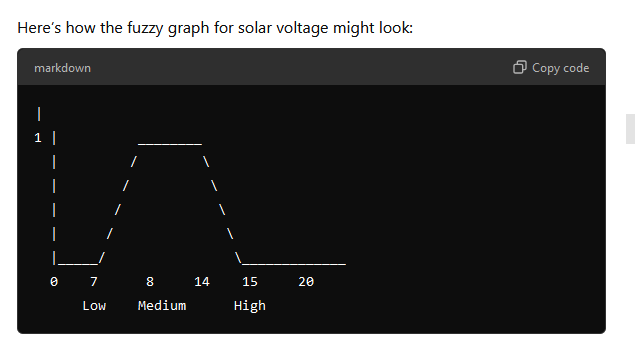
**Membership Functions:**

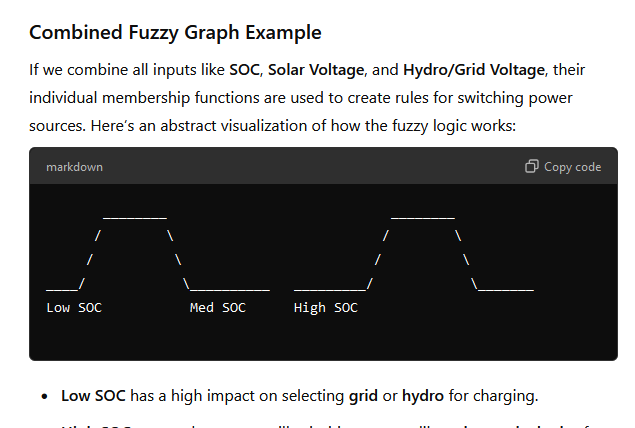
* **Low SOC** starts at 0% and smoothly transitions to zero membership around 40%.
* **Medium SOC** starts around 30%, peaks at full membership from 40% to 70%, and decreases at 80%.
* **High SOC** starts around 70%, with full membership from 80% to 100%.

Here’s a rough idea of how this might look:

markdown







MODIFIED CODE  
  
// Define Fuzzy Sets for Grid, Solar, Hydro, and SoC

FuzzyInput\* grid = new FuzzyInput(1);

FuzzyInput\* solar = new FuzzyInput(2);

FuzzyInput\* hydro = new FuzzyInput(3);

FuzzyInput\* soc = new FuzzyInput(4);

// Define SoC fuzzy sets (Low, Medium, High)

FuzzySet\* socLow = new FuzzySet(0, 0, 30, 40); // 0% to 30% SOC

FuzzySet\* socMedium = new FuzzySet(30, 40, 70, 80); // 30% to 70% SOC

FuzzySet\* socHigh = new FuzzySet(70, 80, 100, 100); // 70% to 100% SOC

soc->addFuzzySet(socLow);

soc->addFuzzySet(socMedium);

soc->addFuzzySet(socHigh);

fuzzy->addFuzzyInput(soc);

// Define fuzzy sets for Solar, Hydro, and Grid (Low, Medium, High, NI = Not In Use)

FuzzySet\* solarLow = new FuzzySet(0, 0, 5, 7);

FuzzySet\* solarMedium = new FuzzySet(6, 8, 12, 14);

FuzzySet\* solarHigh = new FuzzySet(13, 15, 18, 20);

solar->addFuzzySet(solarLow);

solar->addFuzzySet(solarMedium);

solar->addFuzzySet(solarHigh);

fuzzy->addFuzzyInput(solar);

FuzzySet\* hydroLow = new FuzzySet(0, 0, 5, 7);

FuzzySet\* hydroMedium = new FuzzySet(6, 8, 12, 14);

FuzzySet\* hydroHigh = new FuzzySet(13, 15, 18, 20);

hydro->addFuzzySet(hydroLow);

hydro->addFuzzySet(hydroMedium);

hydro->addFuzzySet(hydroHigh);

fuzzy->addFuzzyInput(hydro);

FuzzySet\* gridLow = new FuzzySet(0, 0, 5, 7);

FuzzySet\* gridMedium = new FuzzySet(6, 8, 12, 14);

FuzzySet\* gridHigh = new FuzzySet(13, 15, 18, 20);

grid->addFuzzySet(gridLow);

grid->addFuzzySet(gridMedium);

grid->addFuzzySet(gridHigh);

fuzzy->addFuzzyInput(grid);

// Define Fuzzy Outputs for actions (e.g., Grid ON Low, Grid ON Medium, etc.)

FuzzyOutput\* action = new FuzzyOutput(1);

FuzzySet\* gridOnLow = new FuzzySet(0, 0, 5, 7);

FuzzySet\* gridOnMedium = new FuzzySet(6, 8, 12, 14);

action->addFuzzySet(gridOnLow);

action->addFuzzySet(gridOnMedium);

fuzzy->addFuzzyOutput(action);

// Rule 1: If Grid is NI, Solar is NI, Hydro is NI, SoC is Low

FuzzyRuleAntecedent\* ifGridNI\_SolarNI\_HydroNI\_SocLow = new FuzzyRuleAntecedent();

ifGridNI\_SolarNI\_HydroNI\_SocLow->joinWithAND(gridLow, socLow); // Example: Grid NI (Low) and SoC Low

FuzzyRuleConsequent\* thenTurnGridLow = new FuzzyRuleConsequent();

thenTurnGridLow->addOutput(gridOnLow); // Turn Grid ON Low

FuzzyRule\* fuzzyRule1 = new FuzzyRule(1, ifGridNI\_SolarNI\_HydroNI\_SocLow, thenTurnGridLow);

fuzzy->addFuzzyRule(fuzzyRule1);

// Rule 2: If Hydro is Low, SoC is Low, turn Grid ON Medium

FuzzyRuleAntecedent\* ifHydroLow\_SocLow = new FuzzyRuleAntecedent();

ifHydroLow\_SocLow->joinWithAND(hydroLow, socLow);

FuzzyRuleConsequent\* thenTurnGridMedium = new FuzzyRuleConsequent();

thenTurnGridMedium->addOutput(gridOnMedium); // Turn Grid ON Medium

FuzzyRule\* fuzzyRule2 = new FuzzyRule(2, ifHydroLow\_SocLow, thenTurnGridMedium);

fuzzy->addFuzzyRule(fuzzyRule2);

// More rules based on the provided rule set

// For example: If Hydro is High and SoC is Medium, no action is needed

FuzzyRuleAntecedent\* ifHydroHigh\_SocMedium = new FuzzyRuleAntecedent();

ifHydroHigh\_SocMedium->joinWithAND(hydroHigh, socMedium);

FuzzyRuleConsequent\* thenNoAction = new FuzzyRuleConsequent();

// No corrective action for this case

FuzzyRule\* fuzzyRule3 = new FuzzyRule(3, ifHydroHigh\_SocMedium, thenNoAction);

fuzzy->addFuzzyRule(fuzzyRule3);

CHANGING VOLTAGE OF A PIN  
// Assuming pin 9 is the output pin (PWM pin on Arduino)

int outputPin = 9;

void setup() {

pinMode(outputPin, OUTPUT);

}

void loop() {

// Calculate fuzzy logic output

fuzzy->setInput(1, grid\_value); // Input from Grid fuzzy set

fuzzy->setInput(2, solar\_value); // Input from Solar fuzzy set

fuzzy->setInput(3, hydro\_value); // Input from Hydro fuzzy set

fuzzy->setInput(4, soc\_value); // Input from SoC fuzzy set

fuzzy->fuzzify(); // Run the fuzzy logic system

// Get the fuzzy output, which is between 0 to 100

float fuzzyOutput = fuzzy->defuzzify(1);

// Map fuzzy output to an analog value (0 to 255 for PWM)

int pwmValue = map(fuzzyOutput, 0, 100, 0, 255);

// Write the PWM value to the output pin (changes voltage)

analogWrite(outputPin, pwmValue);

delay(100); // Delay for stability

}

o define the voltage ranges for **Low (L)**, **Medium (M)**, and **High (H)** power inputs for **grid**, **solar**, and **hydro** sources, we can establish voltage boundaries based on typical operating conditions. Here's a general approach you can follow, but you may adjust the values depending on the actual system setup and requirements.

**Voltage Ranges for the Sources:**

1. **Grid (Voltage typically stable and higher)**
   * **Low (L)**: 11.5V to 12.5V
   * **Medium (M)**: 12.5V to 13.2V
   * **High (H)**: 13.2V to 14V
2. **Solar (Fluctuates based on sunlight)**
   * **Low (L)**: 0V to 12V (cloudy or low sunlight)
   * **Medium (M)**: 12V to 18V (moderate sunlight)
   * **High (H)**: 18V to 24V (bright sunlight, peak generation)
3. **Hydro (Depends on water flow)**
   * **Low (L)**: 0V to 12V (minimal water flow)
   * **Medium (M)**: 12V to 18V (moderate water flow)
   * **High (H)**: 18V to 24V (strong water flow)

**Explanation:**

* **Grid Power**: Since the grid generally provides stable power, the variations are not as wide, but fluctuations in voltage may occur, so you can use the ranges provided for L, M, and H to determine corrective actions.
* **Solar Power**: Solar panels typically output voltages between 0V (night or heavy cloud cover) and 18V-24V in full sunlight. You can set thresholds to determine if the input is low, medium, or high based on these ranges.
* **Hydro Power**: Similar to solar, hydro generators produce varying voltages based on the strength of the water flow. The low, medium, and high levels are based on the generator's capacity.

**How to Use These Voltage Ranges:**

In the fuzzy logic system, when you measure the voltage coming from each source, you can categorize the input as **Low**, **Medium**, or **High** based on the ranges mentioned above. This categorization would then feed into the fuzzy logic rules to determine the corrective actions.

For example:

* If the **solar input** voltage is 19V, this would fall under the **High** category (H), and the corresponding fuzzy rule would be triggered.
* If the **grid input** voltage is 12V, it would be categorized as **Low** (L), prompting a different corrective action.

**Example in Code:**

You can use simple conditional logic to assign fuzzy sets based on the measured voltages:

// Assuming you read the voltages from the sources using analogRead

float gridVoltage = readGridVoltage(); // Function to read grid voltage

float solarVoltage = readSolarVoltage(); // Function to read solar voltage

float hydroVoltage = readHydroVoltage(); // Function to read hydro voltage

// Fuzzy input for Grid based on the voltage range

if (gridVoltage <= 12.5) {

fuzzy->setInput(1, gridLow); // Grid Low

} else if (gridVoltage <= 13.2) {

fuzzy->setInput(1, gridMedium); // Grid Medium

} else {

fuzzy->setInput(1, gridHigh); // Grid High

}

// Fuzzy input for Solar

if (solarVoltage <= 12) {

fuzzy->setInput(2, solarLow); // Solar Low

} else if (solarVoltage <= 18) {

fuzzy->setInput(2, solarMedium); // Solar Medium

} else {

fuzzy->setInput(2, solarHigh); // Solar High

}

// Fuzzy input for Hydro

if (hydroVoltage <= 12) {

fuzzy->setInput(3, hydroLow); // Hydro Low

} else if (hydroVoltage <= 18) {

fuzzy->setInput(3, hydroMedium); // Hydro Medium

} else {

fuzzy->setInput(3, hydroHigh); // Hydro High

}

Aaa

Given that the **maximum voltage for all sources (grid, solar, hydro)** is 12V, we can redefine the voltage ranges for **Low (L)**, **Medium (M)**, and **High (H)** based on this new maximum. Here's how the voltage ranges could be adjusted:

**New Voltage Ranges (with 12V max):**

1. **Grid** (Maximum: 12V)
   * **Low (L)**: 0V to 8V
   * **Medium (M)**: 8V to 10.5V
   * **High (H)**: 10.5V to 12V
2. **Solar** (Maximum: 12V)
   * **Low (L)**: 0V to 8V (low sunlight)
   * **Medium (M)**: 8V to 10.5V (moderate sunlight)
   * **High (H)**: 10.5V to 12V (bright sunlight, peak generation)
3. **Hydro** (Maximum: 12V)
   * **Low (L)**: 0V to 8V (minimal water flow)
   * **Medium (M)**: 8V to 10.5V (moderate water flow)
   * **High (H)**: 10.5V to 12V (strong water flow)

// Assuming you read the voltages from the sources using analogRead

float gridVoltage = readGridVoltage(); // Function to read grid voltage

float solarVoltage = readSolarVoltage(); // Function to read solar voltage

float hydroVoltage = readHydroVoltage(); // Function to read hydro voltage

// Fuzzy input for Grid based on the new voltage range

if (gridVoltage <= 8) {

fuzzy->setInput(1, gridLow); // Grid Low

} else if (gridVoltage <= 10.5) {

fuzzy->setInput(1, gridMedium); // Grid Medium

} else {

fuzzy->setInput(1, gridHigh); // Grid High

}

// Fuzzy input for Solar

if (solarVoltage <= 8) {

fuzzy->setInput(2, solarLow); // Solar Low

} else if (solarVoltage <= 10.5) {

fuzzy->setInput(2, solarMedium); // Solar Medium

} else {

fuzzy->setInput(2, solarHigh); // Solar High

}

// Fuzzy input for Hydro

if (hydroVoltage <= 8) {

fuzzy->setInput(3, hydroLow); // Hydro Low

} else if (hydroVoltage <= 10.5) {

fuzzy->setInput(3, hydroMedium); // Hydro Medium

} else {

fuzzy->setInput(3, hydroHigh); // Hydro High

}

Yes, it is definitely possible for the \*\*XL4016\*\* to take a \*\*30V input\*\* and adjust it down to \*\*12V output\*\*. The XL4016 is designed to handle input voltages up to \*\*40V\*\*, so a 30V input is within its operating range. Here’s how it works and what you need to consider:

### Key Points for Using XL4016 with a 30V Input:

1. \*\*Input Voltage Range\*\*:

- The XL4016 can accept input voltages from \*\*4.5V to 40V\*\*. Therefore, a 30V input is perfectly acceptable.

2. \*\*Output Voltage Setting\*\*:

- You can set the output voltage to \*\*12V\*\* using the onboard potentiometer. Adjusting the potentiometer changes the feedback resistor values, which allows you to set the desired output voltage.

3. \*\*Efficiency\*\*:

- The efficiency of the XL4016 is generally high (around 85% to 95%) when the input voltage is higher than the output voltage, but this efficiency can vary based on the load current and the difference between input and output voltages. The higher the difference, the more heat may be generated, which can impact efficiency and requires adequate heat dissipation.

4. \*\*Load Current\*\*:

- The XL4016 is rated to provide up to \*\*5A\*\* of output current. Ensure that the load connected to the output does not exceed this limit. If you are drawing significant current, consider adding a heatsink to the XL4016 to help manage heat.

5. \*\*Capacitors\*\*:

- Make sure to use proper input and output capacitors as recommended in the datasheet. These capacitors help stabilize the voltage and improve transient response. Common values for input capacitors might be in the range of \*\*100µF to 470µF\*\*, and output capacitors might be around \*\*220µF\*\* or more.

6. \*\*Thermal Considerations\*\*:

- When stepping down from a high input voltage like 30V to a lower output voltage like 12V, the XL4016 will generate heat due to power loss. The power loss can be estimated using the formula:

\[

P\_{loss} = (V\_{in} - V\_{out}) \times I\_{out}

\]

For instance, if you are drawing 2A at 12V:

\[

P\_{loss} = (30V - 12V) \times 2A = 36W

\]

This is significant, and you will need adequate cooling to prevent overheating.

### Example Setup:

1. \*\*Input\*\*: 30V (from a suitable power supply)

2. \*\*Desired Output\*\*: 12V

3. \*\*Load\*\*: Ensure the load does not exceed 5A.

4. \*\*Cooling\*\*: Use a heatsink or a fan to keep the regulator cool.

### Conclusion:

You can effectively use the XL4016 to convert a 30V input to a 12V output by properly configuring the output voltage setting and ensuring adequate thermal management. Just be mindful of the power loss and potential heat generation when operating with higher input voltages.

