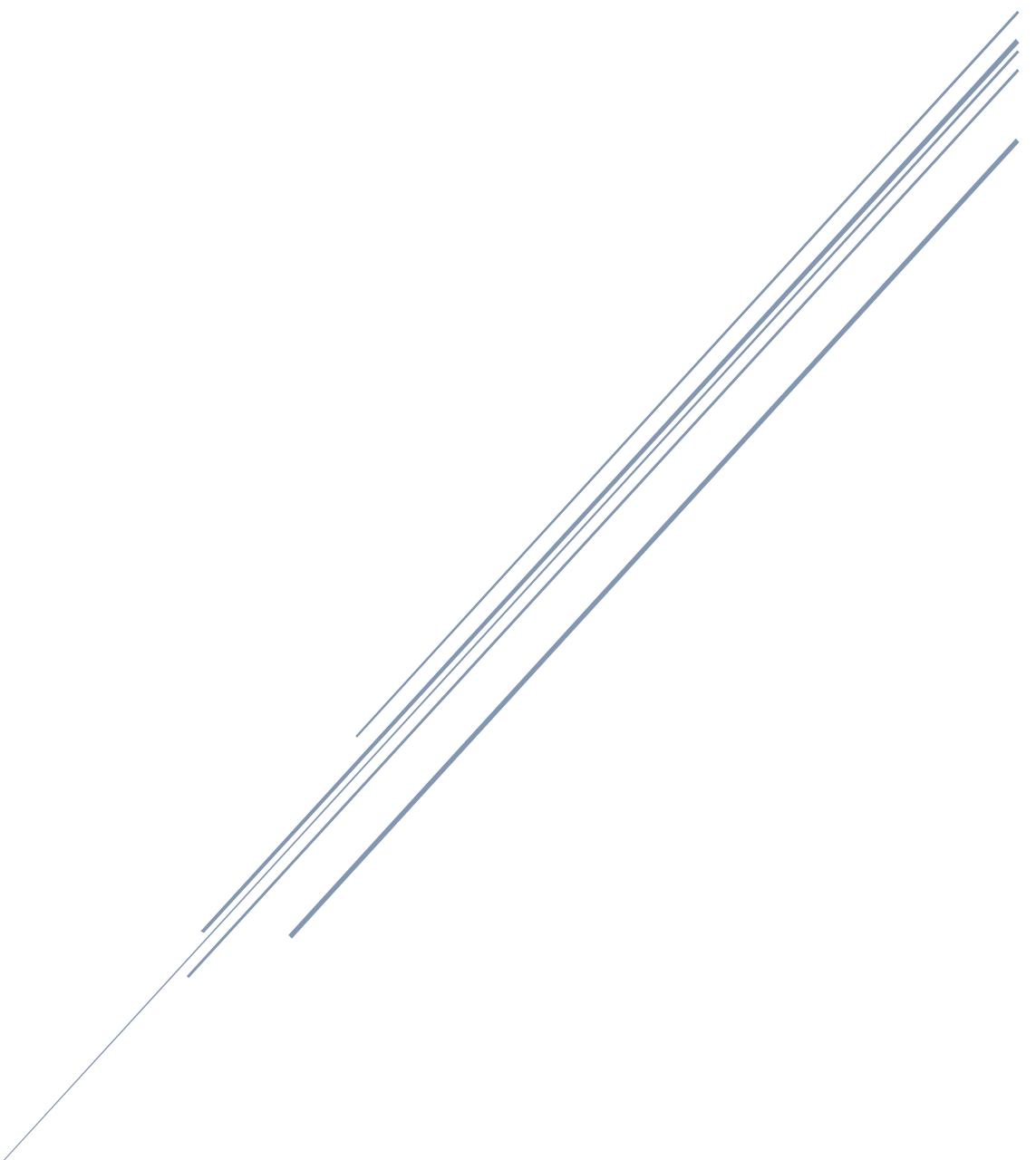


TRIAC FIRING ANGLE CONTROL

A precise TRIAC-based AC voltage control system using zero-cross detection and real-time monitoring



Nariman Ziae
November 2024

فهرست مطالب

<u>صفحه</u>	<u>عنوان</u>
	[معرفی کلی سیستم]
۳	۲-۱ هدف از انجام پروژه
۴	۲-۲ مزایا تکنیک کنترل زاویه آتش
۵	۲-۳ چالش‌ها و نکاتی پیرامون طراحی
	[تحلیل عملکرد مدار]
۶	۲-۱ تشخیص نقطه عبور از صفر (Zero-Cross Detection)
۷	۲-۲ تولید پالس گیت ترایاک و تنظیم زاویه آتش
۸	۲-۳ محاسبه زمان تأخیر و ایجاد پالس گیت
۹	۲-۴ ایزولاسیون ولتاژ بالا با استفاده از اپتوکوپلر
۱۰	۲-۵ اندازه‌گیری و پایش ولتاژ خروجی
	[شبیه‌سازی مدار در نرم‌افزار]
۱۳	۳-۱ شماتیک مدار
۱۴	۳-۲ شبیه‌سازی و نمونه برداری
۱۸	۳-۳ تحلیل نتایج شبیه‌سازی و نکات کلیدی

[تست عملی و نتایج آزمایشگاهی]

۱۹	۴-۱ عیب‌یابی و اقدامات اصلاحی
۲۲	۴-۲ شماتیک مدار اصلاح شده
۲۲	۴-۳ لیست قطعات اصلاح شده
۲۳	۴-۴ خروجی نهایی مدار در عمل

[تحلیل و تجزیه کدهای پروژه]

۲۴	۵-۱ اندازه گیری فرکانس ورودی
۲۶	۵-۲ کنترل زاویه آتش و تولید پالس تریگر
۲۸	۵-۳ اندازه گیری ولتاژ AC خروجی
۳۸	۵-۴ جمع‌بندی و کد نهایی
۴۲	۵-۵ پیشنهادات برای بهبود سیستم و توسعه‌های آتی

[جمع‌بندی]

معرفی کلی سیستم

۱-۱ هدف از انجام پروژه

هدف این پروژه، طراحی سیستمی دیجیتال مبتنی بر میکروکنترلر برای کنترل ولتاژ AC است. این پروژه از چندین المان مهم مانند فتوترانزیستور برای شناسایی نقطه عبور از صفر در سیگنال ورودی و محاسبه دقیق فرکانس، اپتوكوپلر برای ایزولاسیون مدار، و ماژول اندازه‌گیری ولتاژ برای پایش ولتاژ ساخته شده خروجی بهره می‌برد.

کنترل توان در این سیستم بر پایه‌ی تکنیک فاز کنترل شده یا **Phase Angle Control** است. در این روش، با استفاده از یک ترایاک، توان ورودی به بار در هر سیکل سینوسی کنترل می‌شود. سیگنال‌های AC دارای نوسانات سینوسی‌اند که در هر سیکل، دوبار از مقدار صفر عبور می‌کنند. با شناسایی این نقاط و اعمال یک تاخیر زمانی مشخص، می‌توان ترایاک را در لحظه دلخواهی از هر نیم‌سیکل روشن کرد و در نتیجه تنها بخشی از توان را به بار اعمال نمود. این فرآیند در عمل، زاویه آتش (فاز) سیگنال ورودی را تنظیم می‌کند و مقدار توان تحويلی به بار را به صورت خطی تغییر می‌دهد.

برای درک بهتر این موضوع، می‌توان یک نیم‌سیکل از موج سینوسی را تصور کرد. در شروع هر نیم‌سیکل، ولتاژ از صفر عبور کرده و به تدریج افزایش می‌یابد. با تغییر زاویه آتش، می‌توان میزان زمانی که ولتاژ به بار اعمال می‌شود را تنظیم کرد. اگر زاویه آتش کوچک باشد (مثلاً ۱۰ درجه)، تقریباً تمام نیم‌موج به بار اعمال می‌شود، که این به معنای توان بالاست. اما اگر زاویه آتش بزرگ‌تر شود (مثلاً ۱۵۰ درجه)، تنها بخش کوچکی از نیم‌موج به بار منتقل می‌شود و در نتیجه توان کاهش می‌یابد.

۱-۲ مزایا تکنیک کنترل زاویه آتش

استفاده از کنترل زاویه آتش برای تنظیم توان دارای مزایای متعددی است:

۱. افزایش بازدهی سیستم: با تنظیم توان تحويلی، میزان مصرف انرژی بهینه‌سازی شده و

از مصرف بیش از حد انرژی جلوگیری می‌شود. این ویژگی به ویژه در پروژه‌های صنعتی

و تجاری که هزینه‌های انرژی اهمیت بالایی دارند، بسیار مفید است.

۲. کاهش هزینه‌های عملیاتی و نگهداری: بارهای متصل به شبکه AC، تحت تأثیر مستقیم

ولتاژ ورودی هستند. با جلوگیری از افزایش ناگهانی ولتاژ اعمالی و حفظ پایداری سطح

ولتاژ خروجی، از دمای بار کاسته شده و در نتیجه طول عمر تجهیزات افزایش می‌یابد.

۳. کاهش تداخلات الکترومغناطیسی (EMI): در سیستم‌های AC با بارهای القایی یا حتی

مقاومتی، تغییرات ناگهانی ولتاژ می‌تواند به ایجاد نویزهای الکترومغناطیسی منجر شود

که با استفاده از تکنیک کنترل فاز، از تداخلات ناگهانی ولتاژ جلوگیری شده و میزان

EMI کاهش می‌یابد.

۴. کنترل آسان و دقیق توان تحويلی: با استفاده از یک میکروکنترلر و تنظیم زاویه آتش،

می‌توان توان خروجی را با دقت بالایی کنترل کرد و به شرایط دلخواه دست یافت.

به این ترتیب، امکان تنظیم دقیق شدت نور، سرعت و یا دمای بار فراهم می‌شود.

۵. ایمنی و جداسازی مدار کنترل از شبکه AC: با استفاده از المان‌های اپتیک در طراحی

مدار، قسمت کنترلی از جریان اصلی AC ایزوله شده و این مسئله ایمنی بیشتری را

برای میکروکنترلر و دیگر اجزای شبکه فراهم می‌کند.

۱-۳ چالش‌ها و نکاتی پیرامون سیستم

طراحی چنین سیستمی با چالش‌هایی همراه است که باید در روند انجام پروژه به آن‌ها توجه داشت:

۱. تشخیص دقیق نقطه عبور از صفر: از آنجایی که دقت در تشخیص عبور از صفر و تنظیم

زاویه آتش تأثیر مستقیم بر کنترل توان خروجی دارد، انتخاب و استفاده از یک اپتوکوپلر

مناسب مثل **H11AA1** یا **4N25** از اهمیت بالایی برخوردار است.

۲. محاسبه دقیق زمان تاخیر زاویه آتش: برای هر فرکانس خاص (۵۰ یا ۶۰ هرتز)، مدت

زمان کامل یک نیمسیکل محاسبه شده و بر اساس زاویه آتش مورد نیاز، مدت زمان

تأخیر به دقت تنظیم می‌شود. هر خطای در محاسبه زمان تاخیر می‌تواند به عدم دقت در

تنظیم توان خروجی منجر شود.

۳. حفظ ایزولاسیون و ایمنی مدار کنترل: اپتوکوپلر به عنوان یک واسطه برای ایزولاسیون

مدار کنترل از مدار قدرت عمل می‌کند. این امر برای حفظ ایمنی کاربران و جلوگیری

از آسیب دیدن میکروکنترلر بسیار مهم است.

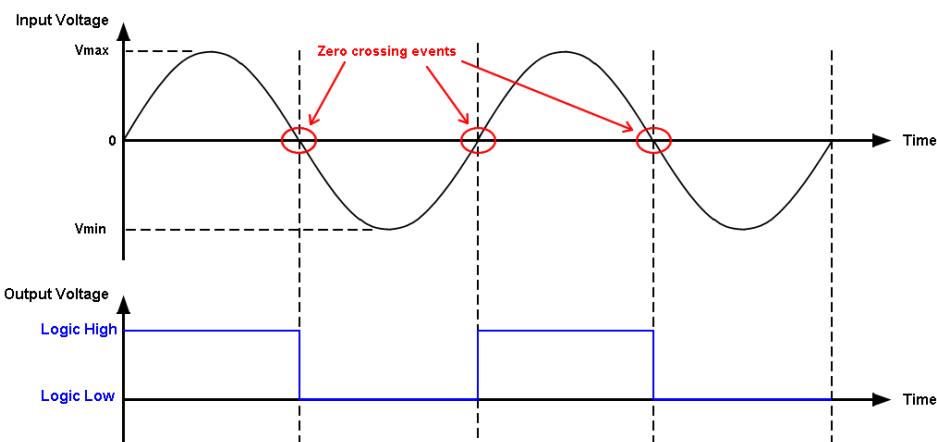
۴. پایش و اندازه‌گیری توان تحويلی به بار: در این پروژه، هدف نهایی پایش دقیق توان

تحویلی به بار است. این امر از طریق نمونه‌برداری از ولتاژ خروجی در شرایط مختلف

زاویه آتش و بررسی تأثیر آن بر بارهای مختلف انجام می‌شود.

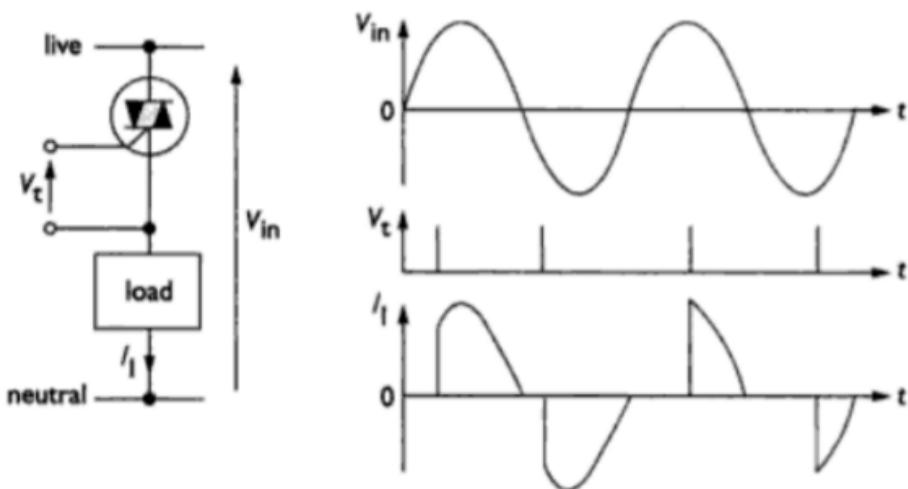
تحلیل عملکرد مدار

۲-۱ تشخیص نقطه عبور از صفر (Zero-Cross Detection)



برای کنترل زاویه آتش تراپاک، شناسایی لحظه عبور از صفر سیگنال ورودی AC اهمیت دارد. پس از تحقیق و شناسایی المان های مختلف، دو فتورانزیستور H11AA1 و 4N25 برای تشخیص نقطه عبور از صفر استفاده شد که به دلیل توانایی شناسایی نقطه عبور از صفر در هر دو لبه بالا رونده و پایین رونده، به اندازه گیری هرچه دقیق تر فرکانس کمک میکنند. فتورانزیستور سیگنال لحظه عبور از صفر را پس از ایزولاسیون بخش ولتاژ بالا از ولتاژ پایین به میکروکنترلر ارسال می کند. سپس، میکروکنترلر با استفاده از این اطلاعات، لحظه دقیق تولید پالس گیت تراپاک را تنظیم می کند. این روش تضمین می کند که تراپاک فقط در لحظات تعیین شده فعال شده و ولتاژ خروجی به طور دقیق کنترل شود.

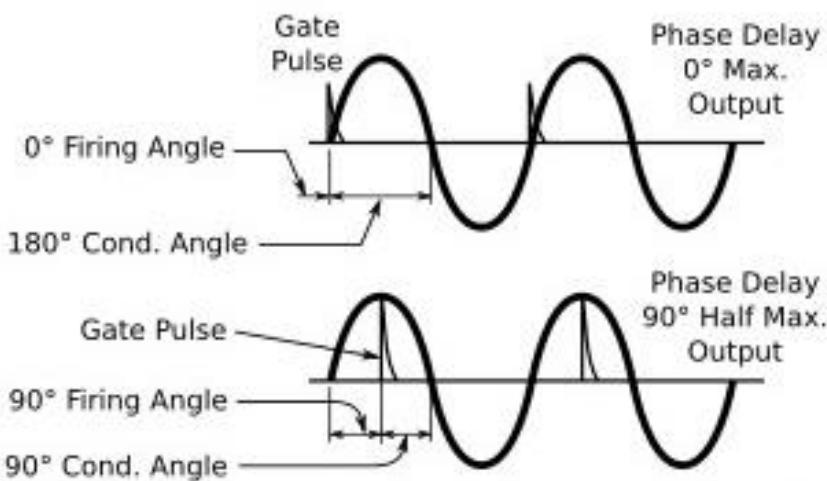
۴-۲ تولید پالس گیت ترایاک و تنظیم زاویه آتش



در سیستم‌های کنترل توان AC، یکی از مهم‌ترین مسائل تولید دقیق پالس گیت برای ترایاک است. ترایاک، به عنوان یک کلید الکترونیکی دوبل که می‌تواند جریان را در هر دو نیمسیکل موج سینوسی عبور دهد، برای روشن و خاموش کردن بارهای متصل به AC به کار می‌رود. یکی از راه‌های متداول برای کنترل ترایاک، کنترل زاویه آتش یا Phase Angle Control است. در این روش، با تنظیم زاویه آتش با تاخیر در اعمال پالس گیت به ترایاک، می‌توان مقدار توان اعمالی به بار را به دقت کنترل کرد.

در روش کنترل زاویه آتش، جریان تنها زمانی به بار متصل به ترایاک اعمال می‌شود که یک پالس به گیت ترایاک پس از عبور ولتاژ از صفر در هر نیمسیکل ارسال شود. زاویه آتش، به زاویه‌ای گفته می‌شود که در آن ولتاژ ورودی پس از عبور از صفر به نقطه‌ای می‌رسد که پالس گیت اعمال شده و ترایاک روشن می‌شود. با تنظیم این زاویه، می‌توان مقدار ولتاژ موثر (RMS) اعمالی به بار و در نتیجه توان خروجی را کنترل کرد.

۲-۳ محاسبه زمان تأخیر و ایجاد پالس گیت



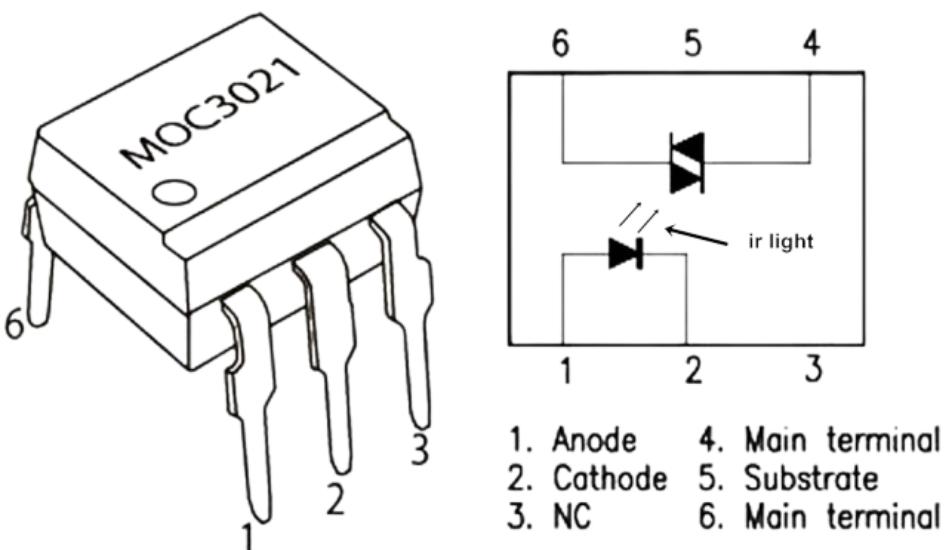
برای تنظیم زاویه آتش به یک مقدار خاص، لازم است تا زمان تأخیر پس از عبور از صفر (Zero Crossing) محاسبه و پالس گیت با تأخیر معین تولید شود. مراحل کلی تولید پالس گیت به این صورت است:

۱. تشخیص عبور از صفر: با استفاده از مدار تشخیص عبور از صفر، لحظه‌ای که ولتاژ AC از مقدار صفر عبور می‌کند شناسایی شده و تایмер میکروکنترلر شروع به شمارش می‌کند.

۲. محاسبه زمان تأخیر: بر اساس فرکانس AC (مثلاً ۵۰ یا ۶۰ هرتز) و زاویه آتش مطلوب، زمان تأخیر محاسبه می‌شود. برای مثال، در فرکانس ۵۰ هرتز، مدت زمان یک نیم‌سیکل برابر با ۱۰ میلی‌ثانیه است. اگر زاویه آتش ۹۰ درجه باشد، زمان تأخیر باید ۵ میلی‌ثانیه (نصف نیم‌سیکل) باشد. به این ترتیب، اگر زاویه آتش ۴۵ درجه باشد، زمان تأخیر تقریباً ۲.۵ میلی‌ثانیه خواهد بود.

۳. تولید پالس گیت: پس از گذشت زمان تأخیر مشخص، یک پالس کوتاه به گیت ترایاک اعمال می‌شود. این پالس باعث روشن شدن ترایاک و اعمال جریان به بار می‌شود.

۴-۴ ایزولاسیون ولتاژ بالا با استفاده از اپتوکوپلر



ایزولاسیون ولتاژ به معنای جدا کردن مدار قدرت از مدار کنترل به گونه‌ای است که از لحاظ الکتریکی به هم متصل نباشند. این جداسازی مانع از عبور مستقیم جریان بین دو مدار شده و از مدارهای حساس در برابر خطرات ناشی از ولتاژهای بالا محافظت می‌کند. این امر به خصوص در مدارهایی که از یک میکروکنترلر برای کنترل بارهایی مانند موتورهای الکتریکی، لامپ‌های رشته‌ای، تجهیزات گرمایشی و یا هر بار AC دیگری استفاده می‌کنند، اهمیت دارد.

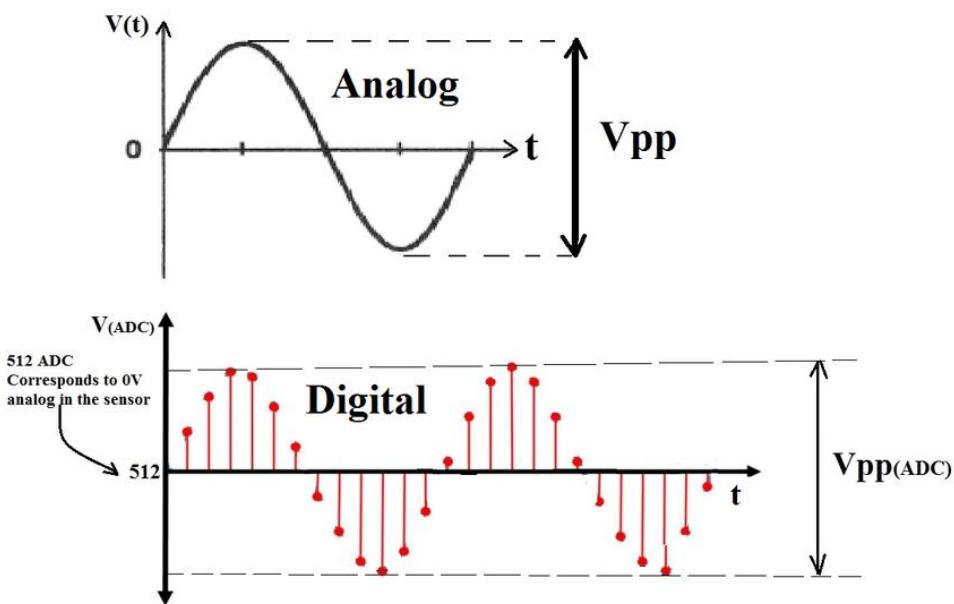
اپتوکوپلر از دو بخش اصلی تشکیل شده است:

۱. دیود نوری: در بخش ورودی اپتوکوپلر، یک LED قرار دارد که با عبور جریان از آن روشن می‌شود و نور تولید می‌کند.

۲. فتوترانزیستور یا فتودیود: در بخش خروجی، یک فتوترانزیستور یا فتودیود قرار دارد که نور تولید شده توسط LED را دریافت کرده و جریان خروجی متناظر را تولید می‌کند.

عملکرد اپتوکوپلر به این صورت است که جریان الکتریکی در سمت ورودی باعث روشن شدن LED می‌شود و نور تولید شده توسط LED به فتوترانزیستور در سمت خروجی برخورد می‌کند و جریان یا ولتاژی در سمت خروجی تولید می‌کند که با سیگنال ورودی مطابقت دارد.

۴-۵ اندازه‌گیری و پایش ولتاژ خروجی



اندازه‌گیری ولتاژ AC مستقیم از خروجی بدون کاهش سطح ولتاژ، برای مدارهای دیجیتال مناسب نیست. به همین دلیل، لازم است از تکنیک‌هایی استفاده شود که ولتاژ خروجی را به سطح قابل اندازه‌گیری برای میکروکنترلر تبدیل کنند.

برای اندازه‌گیری و پایش ولتاژ خروجی در این پروژه، مراحل زیر پیشنهاد می‌شود:

- کاهش ولتاژ خروجی: با استفاده از یک مدار تقسیم مقاومتی، ولتاژ خروجی به سطحی مناسب برای میکروکنترلر کاهش می‌یابد. برای مثال، اگر ولتاژ خروجی ۲۲۰ ولت AC باشد، با تنظیم مقاومت‌ها، ولتاژ در حد چند ولت کاهش می‌یابد تا در محدوده کاری ADC قرار گیرد.
- تبدیل ولتاژ AC به DC: پس از کاهش ولتاژ، از یک مدار یکسوکننده ولتاژ و تبدیل آن به سیگنال استفاده می‌شود. این کار، امکان اندازه‌گیری ساده‌تر و پایدارتر ولتاژ را فراهم می‌کند.

- نمونه‌برداری با ADC میکروکنترلر: ولتاژی که به‌طور پیوسته نمونه‌برداری شده و به ADC میکروکنترلر اعمال شده در میکروکنترلرهایی که سطح ولتاژ منطقی آنها برابر $5V$ میباشد چیزی بین ۰ تا ۱۰۲۳ متغیر است. این نمونه‌برداری به میکروکنترلر امکان می‌دهد تا به‌طور پیوسته ولتاژ خروجی را اندازه‌گیری کرده و داده‌های لازم برای تنظیم زاویه آتش را دریافت کند.

در رابطه زیر نحوه مدل کردن مقدار خوانده شده ADC به ولتاژ موثر نوشته شده است
و V_{ref} همان ولتاژ ورودی ما می‌باشد که در واقع برابر $220V$ هست.

$$V_{RMS} = (ADC \ reading \div 1023) \times V_{ref}$$

- محاسبه ولتاژ RMS یا پیک ولتاژ: میکروکنترلر می‌تواند با پردازش داده‌های نمونه‌برداری شده، ولتاژ RMS یا پیک را محاسبه کند. این محاسبات بر اساس فرمول‌های زیر انجام می‌شوند:
ولتاژ پیک:

$$V_{peak} = Scaling\ Factor \times ADC\ reading$$

:RMS ولتاژ

$$V_{RMS} = \frac{V_{peak}}{\sqrt{2}}$$

- نمایش ولتاژ خروجی و تنظیم زاویه آتش: اطلاعات ولتاژ خروجی روی نمایشگر LCD نشان داده شده و بر اساس آن، زاویه آتش تنظیم می‌شود تا ولتاژ خروجی به مقدار مطلوب برسد.

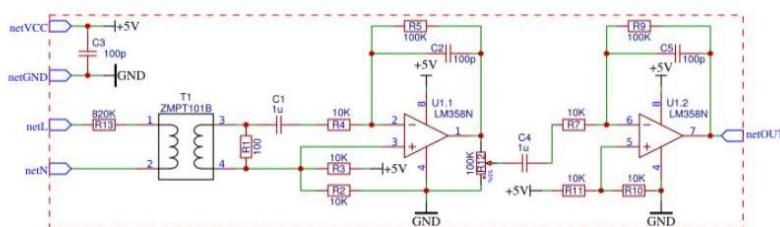
ZMPT101B به منظور حفظ ولتاژ خروجی در محدوده تعیین شده، از مژول اندازه‌گیری ولتاژ استفاده شده است. این مژول پس از کاهش سطح ولتاژ به وسیله ترانسفورماتور طراحی شده در خود، سیگنال تضعیف شده را به یک مقایسه کننده تحویل میدهد و به طور مداوم ولتاژ خروجی را اندازه‌گیری کرده و داده‌ها را به میکروکنترلر ارسال می‌کند. با دریافت این داده‌ها، میکروکنترلر می‌تواند زاویه آتش تراپاک را بر اساس مقدار ولتاژ خروجی تنظیم کند تا تغییرات ولتاژ خروجی در یک محدوده مجاز حفظ شود.

✖ مشخصات و ویژگی‌های مژول ZMPT101B

۱. دقت بالا در اندازه‌گیری ولتاژ AC: این مژول برای اندازه‌گیری ولتاژ AC با دقت بالا طراحی شده و می‌تواند تغییرات کوچک در ولتاژ را نیز تشخیص دهد.
۲. ایزولاسیون الکتریکی: با استفاده از ترانسفورماتور داخلی، مژول ZMPT101B به خوبی ولتاژ AC ورودی را از مدار میکروکنترلر ایزوله می‌کند. این ویژگی باعث ایمنی بیشتر و حفاظت مدار در برابر ولتاژهای بالا می‌شود.
۳. خروجی آنالوگ: این مژول دارای خروجی آنالوگ است که متناسب با ولتاژ ورودی AC تغییر می‌کند و می‌تواند توسط ADC میکروکنترلر نمونه‌برداری شود.
۴. امکان تنظیم گین (Gain): بر روی برد این مژول یک پتانسیومتر وجود دارد که می‌توان از آن برای تنظیم بهره (Gain) سیگنال خروجی استفاده کرد و دامنه ولتاژ خروجی را به محدوده قابل قرائت توسط ADC میکروکنترلر تنظیم کرد.

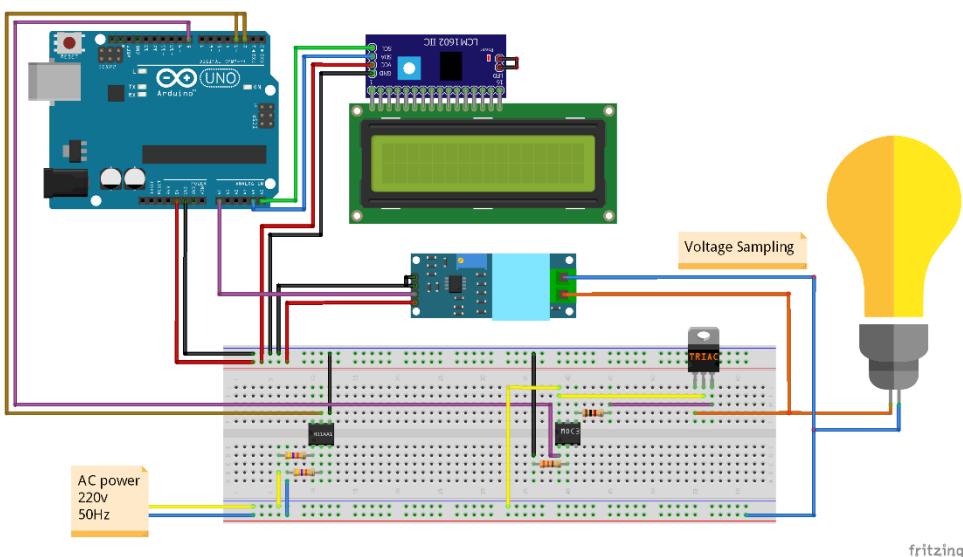


The ZMPT101B module is built with an **LM358N IC** chip, a some **resistors**, and few **capacitor** that helps reduce noise, or unwanted electrical signals.



شبیه‌سازی مدار در نرم‌افزار

۳-۱ شماتیک مدار



طراحی شماتیک مدار شامل تمام اجزای اصلی است:

برد توسعه **Arduino UNO**, فتوترانزیستور **H11AA1**, اپتوكوپلر **MOC3021**, ترایاک و ماژول اندازه‌گیری ولتاژ **ZMPT101B**. در این شماتیک، ارتباط بین قطعات به نحوی تنظیم شده است که سیگنال عبور از صفر توسط **H11AA1** به میکروکنترلر ارسال شود و میکروکنترلر پس از پردازش، پالس گیت ترایاک را از طریق **MOC3021** ارسال کند. ماژول نیز ولتاژ خروجی را اندازه‌گیری کرده و به **Arduino** منتقل می‌کند.

۳-۲ شبیه‌سازی و نمونه‌برداری



Proteus

در این بخش از پروژه، شبیه‌سازی دقیق مدار کنترل ولتاژ با استفاده از نرم‌افزار پروتئوس انجام شد تا عملکرد کلی سیستم و تعامل صحیح اجزا بررسی شود. این شبیه‌سازی، کارکرد سه بخش اصلی مدار شامل شناسایی نقطه عبور از صفر، تولید پالس گیت ترایاک، و اندازه‌گیری ولتاژ خروجی را به صورت جداگانه و همچنین در عملکرد هم‌زمان مورد ارزیابی قرار داد.

• شناسایی نقطه عبور از صفر (Zero-Cross Detection)

در مرحله اول، شبیه‌سازی بر روی شناسایی لحظه عبور از صفر سیگنال AC ورودی با استفاده از فتوترانزیستور H11AA1 متمرکز شد. این قطعه با دقت بالایی لحظات عبور از صفر را شناسایی کرده و سیگنال مربوطه را به میکروکنترلر ارسال می‌کند که این سیگنال به عنوان مرجع زمانی برای تولید پالس گیت ترایاک به کار می‌رود. همچنین نتایج شبیه‌سازی نشان دادند که H11AA1 به صورت دقیق و با کمترین تأخیر، سیگنال عبور از صفر را در زمان‌های مناسب به میکروکنترلر ارسال می‌کند، که این ویژگی برای تنظیم زاویه آتش بسیار مهم است.

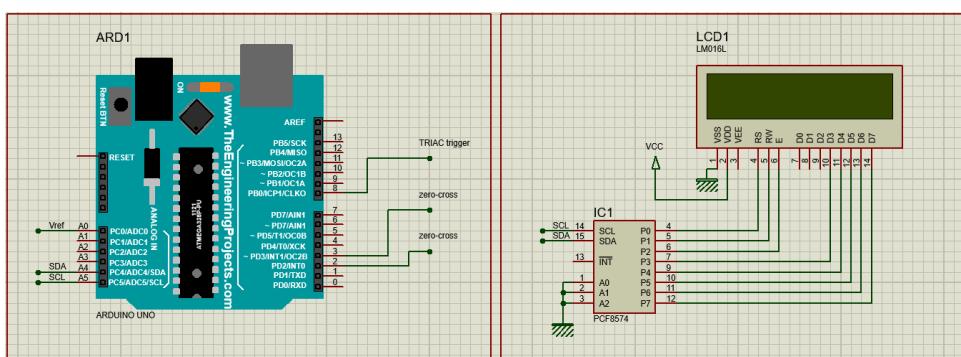
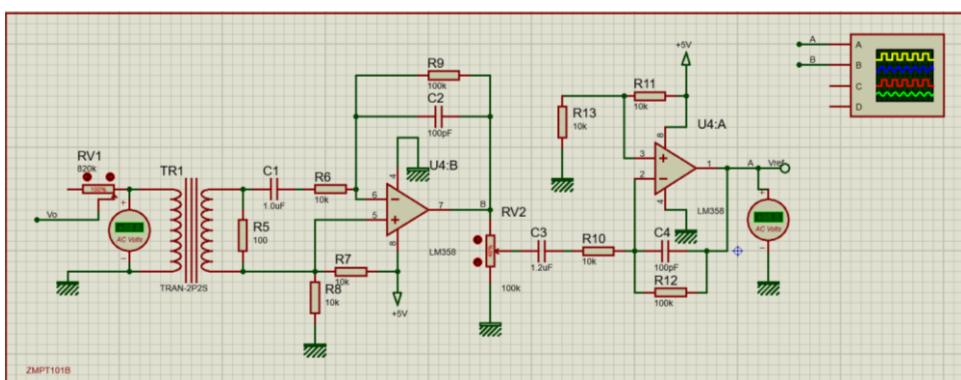
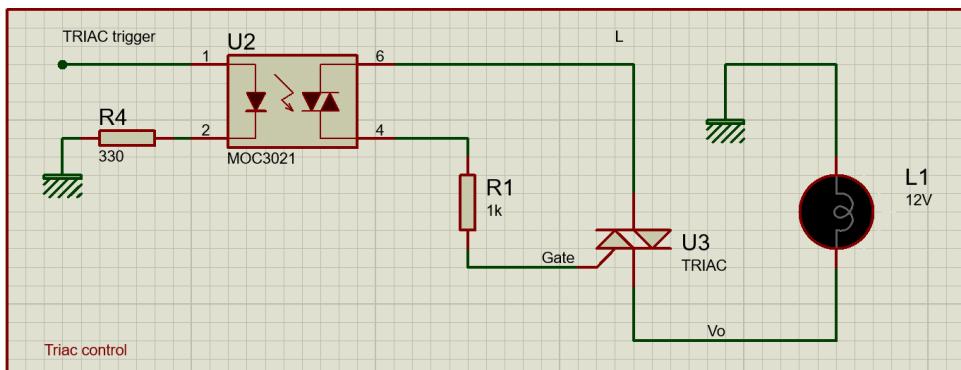
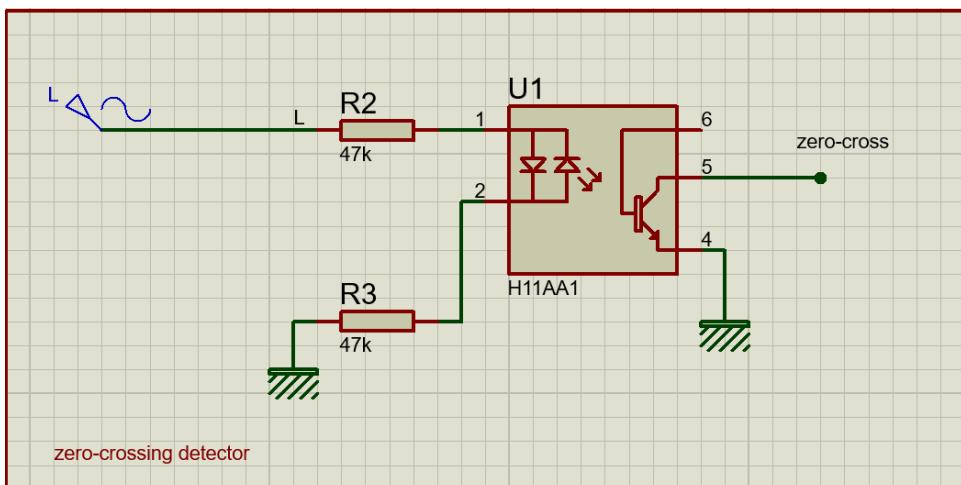
- تولید پالس گیت ترایاک و تنظیم زاویه آتش

پس از دریافت سیگنال عبور از صفر، میکروکنترلر وارد فاز تنظیم زاویه آتش و تولید پالس گیت برای ترایاک می‌شود. در شبیه‌سازی، تأخیر زمانی پس از شناسایی نقطه عبور از صفر به صورت پارامتری تنظیم شد تا زاویه آتش به دلخواه کنترل شود. به کمک این تأخیر، میکروکنترلر قادر است که پالس گیت ترایاک را در زمان دقیق و زاویه آتش مشخصی ایجاد کند. نتایج شبیه‌سازی نشان دادند که سیستم به درستی پالس‌های گیت را با توجه به زاویه تنظیم شده تولید می‌کند و ترایاک در زاویه مورد نظر فعال می‌شود. این امر امکان کنترل دقیق ولتاژ خروجی را فراهم کرده و نشان‌دهنده عملکرد صحیح سیستم در تنظیم زاویه آتش است.

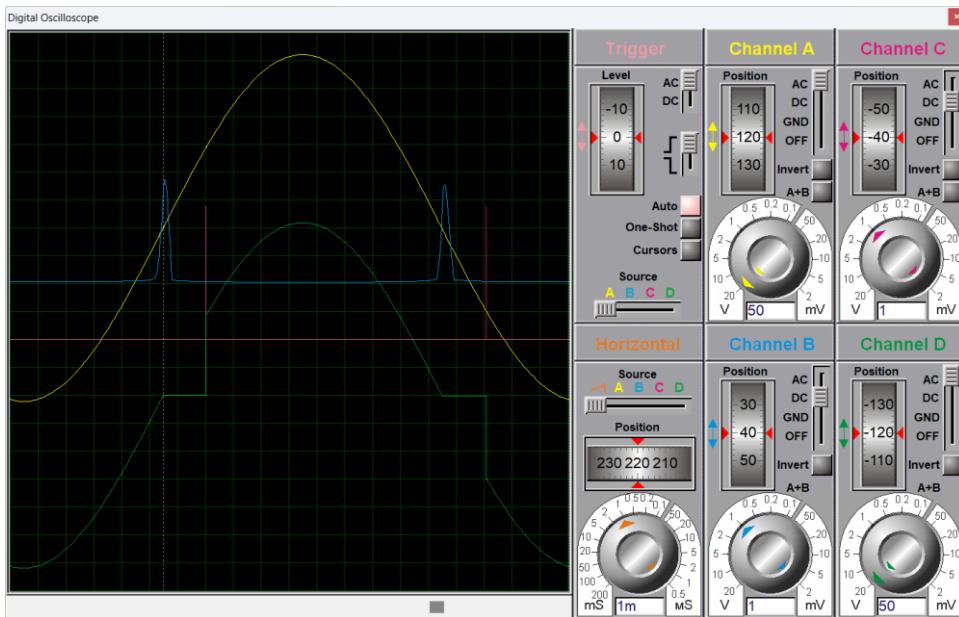
- اندازه‌گیری ولتاژ خروجی با استفاده از ماژول **ZMPT101B**

در ادامه، از ماژول ایزوله **ZMPT101B** برای اندازه‌گیری ولتاژ خروجی استفاده شد. این ماژول به طور مداوم ولتاژ خروجی را اندازه‌گیری کرده و نتایج را به میکروکنترلر ارسال می‌کند. میکروکنترلر، با استفاده از این داده‌ها، ولتاژ خروجی را پایش و در صورت نیاز زاویه آتش ترایاک را تنظیم می‌کند تا ولتاژ خروجی در محدوده مشخص شده باقی بماند. در شبیه‌سازی، تغییرات ولتاژ ورودی و همچنین نوسانات احتمالی مورد آزمایش قرار گرفت و نتایج نشان داد که ماژول با دقت بالا ولتاژ را اندازه‌گیری کرده و به میکروکنترلر گزارش می‌دهد. این داده‌ها امکان ایجاد یک مدار کنترل حلقه‌بسته را فراهم می‌کنند که منجر به ثبات بیشتر در ولتاژ خروجی مدار می‌شود.

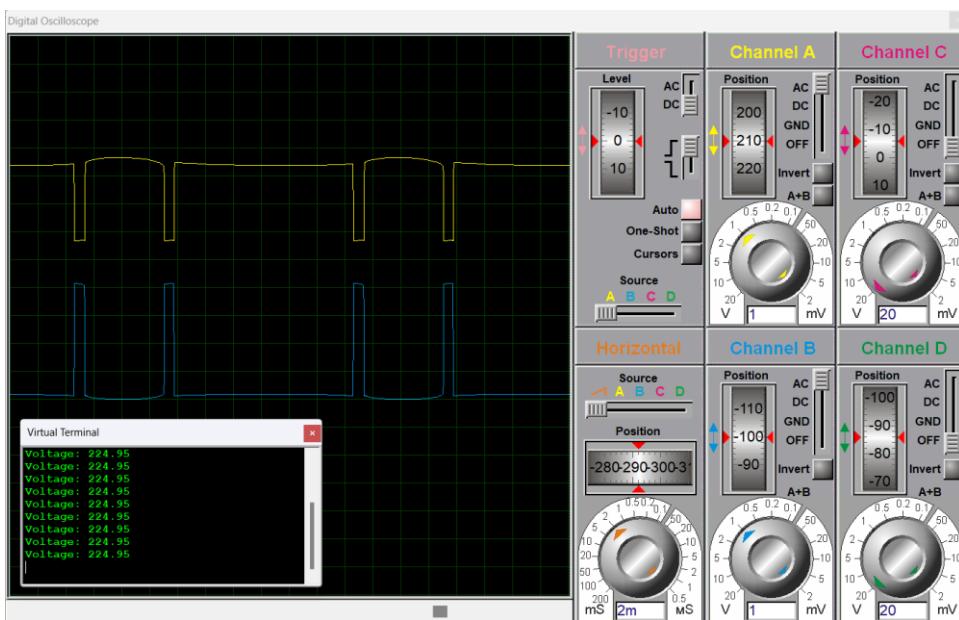
شماتیک مدار طراحی شده در نرم افزار پروتئوس:



خروجی بخش های مختلف مدار:



سیگنال زرد رنگ "کانال A" ورودی AC مدار - سیگنال آبی رنگ "کانال B" سیگنال قرمز رنگ "کانال C" پاس تریگر تولید شده توسط میکروکنترلر - سیگنال سبز رنگ "کانال D" خروجی AC مدار با زاویه آتش 30°



شکل موج خروجی مازول اندازه گیری ولتاژ zmpt101b به همراه مقادیر خوانده شده توسط میکروکنترلر

۳-۳ تحلیل نتایج شبیه‌سازی و نکات کلیدی

شبیه‌سازی‌های انجام شده در پروتئوس نشان داد که هر یک از بخش‌های مدار به درستی کار می‌کنند و خروجی دقیق و مورد انتظار را فراهم می‌کنند.

نکات کلیدی که از شبیه‌سازی به دست آمد عبارت‌اند از:

- پایداری و دقت در شناسایی نقطه عبور از صفر: سیستم به صورت دقیق لحظات عبور از صفر را شناسایی می‌کند و این دقت به افزایش کارایی تنظیم زاویه آتش و در نتیجه کنترل دقیق‌تر و لتاژ خروجی کمک می‌کند.
- کنترل دقیق زاویه آتش و تولید پالس گیت: با توجه به زمان‌بندی دقیق که در شبیه‌سازی مشاهده شد، سیستم به خوبی زاویه آتش را کنترل می‌کند و پالس‌های گیت با دقت بالا تولید می‌شوند.
- پایش و تنظیم خودکار و لتاژ خروجی: مازول ZMPT101B با دقت کافی و لتاژ خروجی را اندازه‌گیری می‌کند، و میکروکنترلر قادر است در واکنش به تغییرات و لتاژ ورودی یا بار خروجی، زاویه آتش تراپیاک را به روز کند و لتاژ را در محدوده تنظیم شده نگه دارد.
- پاسخ سیستم به تغییرات ناگهانی: نتایج شبیه‌سازی نشان داد که سیستم توانایی پاسخ سریع به تغییرات و لتاژ ورودی و بارهای مختلف را دارد. این ویژگی نشان‌دهنده قابلیت انعطاف‌پذیری سیستم برای کاربردهای مختلف است.

تست عملی و نتایج آزمایشگاهی

۴-۱ عیب‌یابی و اقدامات اصلاحی

در مراحل اولیه تست عملی مدار، دو مشکل اصلی شناسایی شد که نیاز به اصلاح داشتند:

۱. بررسی و عیب‌یابی مدار تشخیص عبور از صفر (Zero-Cross Detection)
 - * مشکل شناسایی شده: خروجی دارای نویز و غیرقابل اعتماد
 - * مشکل مشاهده شده: خروجی مدار دارای نویز زیاد بود و لحظات عبور از صفر سیگنال AC به درستی تشخیص داده نمی‌شد.
- دلایل احتمالی:
 - استفاده از سیگنال AC ولتاژ بالا (۲۲۰ ولت) بدون کاهش مناسب، که می‌تواند باعث ناپایداری و ایجاد نویز در مدار شود.
 - طراحی ضعیف یا نامناسب مدار تشخیص عبور از صفر.
 - حساسیت ناکافی قطعات استفاده شده در مدار مانند فتوترانزیستور یا مدار ورودی.

✓ اقدامات اصلاحی

- برای رفع مشکل تشخیص عبور از صفر، مراحل زیر انجام شد:
- کاهش ولتاژ سیگنال AC ورودی به سطح ایمن‌تر:
 - . استفاده از یک ترانس کاهنده برای تبدیل ولتاژ ۲۲۰ ولت AC به ۱۲ ولت AC.
 - این کاهش ولتاژ نه تنها مدار را ایمن‌تر کرد، بلکه سیگنال ورودی را به سطحی رساند که برای پردازش توسط مدارهای الکترونیکی مناسب‌تر باشد.

- تبدیل سیگنال AC به سیگنال DC پالسی با پل دیودی:

ولناژ ۱۲ ولت AC خروجی از ترانس، وارد یک مدار یک پل دیودی شد. این مدار سیگنال AC را به سیگنال DC تبدیل کرد، که برای شناسایی لحظات عبور از صفر بسیار مناسب‌تر است. مزیت این مرحله، حذف بخشی از نویز موجود در سیگنال AC و آماده سازی آن برای پردازش توسط فتوترانزیستور بود.

- استفاده از فتوترانزیستور 4N25 برای تشخیص عبور از صفر:

خروجی مدار پل دیودی به ورودی LED داخلی فتوترانزیستور 4N25 متصل شد. این فتوترانزیستور با ایزولاسیون نوری، سیگنال عبور از صفر را به یک سیگنال دیجیتال دقیق تبدیل کرد و همچنین باعث افزایش ایمنی مدار شد و از انتقال نویزهای ناخواسته به میکروکنترلر جلوگیری کرد.

- اتصال خروجی مدار زیرو کراس به پایه ایترپت میکروکنترلر:

خروجی فتوترانزیستور به پایه ایترپت صفر میکروکنترلر متصل شد. این پایه به‌گونه‌ای برنامه‌ریزی شد که لحظه عبور از صفر را شناسایی کرده و زمان عبور از صفر را برای تولید پالس گیت ذخیره کند.

❖ نتیجه اقدامات اصلاحی

- خروجی مدار زیرو کراس به یک پالس دیجیتال دقیق و پایدار تبدیل شد که لحظات عبور از صفر سیگنال AC را به خوبی شناسایی می‌کرد.
- نویزهای موجود در خروجی مدار به طور قابل توجهی کاهش یافت و سیگنال قابل اعتماد برای ایترپت میکروکنترلر تولید شد.

۲. بررسی و عیب‌یابی پایه خروجی PWM

- ✖ مشکل شناسایی شده: عدم توانایی تولید پالس تریگر PWM
- ✖ مشکل مشاهده شده: در طراحی اولیه، پایه دیجیتال ۸ میکروکنترلر برای تولید پالس تریگر انتخاب شده بود که در حین تست مشخص شد که این پایه قادر قابلیت PWM است و نمی‌تواند سیگنال مناسب برای کنترل ترایاک را تولید کند.

- تأثیر این مشکل:

- عدم تولید پالس دقیق باعث شد که ترایاک در زاویه آتش تنظیم شده فعال نشود.
- ولتاژ خروجی قابل کنترل نبود و عملکرد مدار عملاً مختلط شد.

- ✓ اقدامات اصلاحی

- انتخاب پایه مناسب با قابلیت PWM :

پایه دیجیتال ۱۰ میکروکنترلر، که از قابلیت PWM پشتیبانی می‌کند، به عنوان پایه خروجی جدید برای تولید پالس تریگر انتخاب شد که می‌تواند سیگنال PWM با فرکانس و دیوی‌سایکل دقیق تولید کند.

- بهینه‌سازی برنامه‌نویسی برای تولید پالس تریگر:

برنامه میکروکنترلر به گونه‌ای اصلاح شد که زمان عبور از صفر را به عنوان مرجع ذخیره کرده و پس از محاسبه تأخیر موردنظر برای زاویه آتش دلخواه، پالس تریگر را تولید کند.

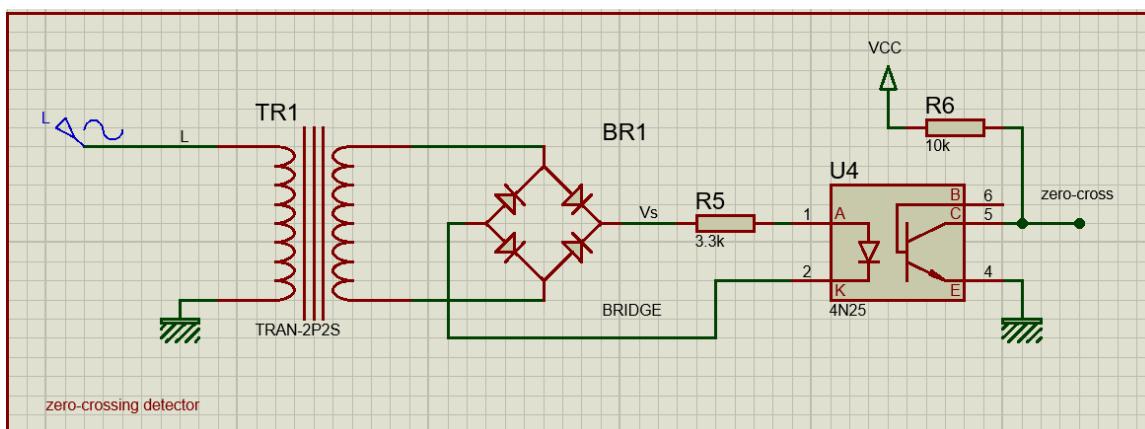
- ❖ نتیجه اقدامات اصلاحی

- پالس تریگر PWM به صورت دقیق و قابل کنترل تولید شد.
- ترایاک توانست با توجه به زاویه آتش تنظیم شده به درستی فعال شود و ولتاژ خروجی مدار با دقت کنترل شد.

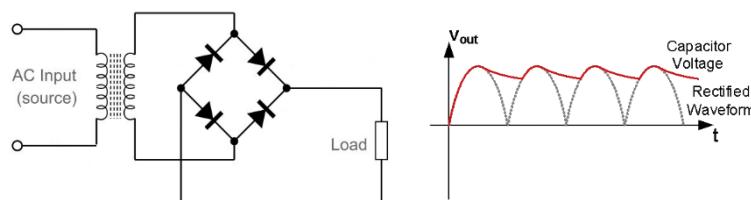
٤-٢ شماتیک مدار اصلاح شده

پس از شناسایی مشکلات اولیه، مدار به صورت کامل باز طراحی شد تا عملکرد آن بهبود یابد.

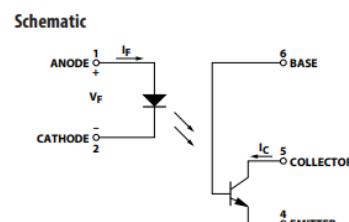
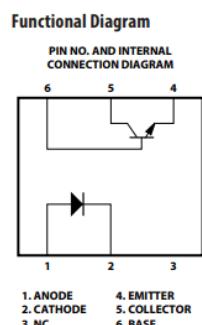
طراحی اصلاح شده شامل تغییرات زیر بود:



٤-٣ لیست قطعات اصلاح شده



ترانس کاهنده و پل دیودی



فتوترانزیستور 4N25

۴-۴ خروجی نهایی مدار در عمل

این مدار یک سیستم کاملاً یکپارچه برای کنترل زاویه آتش ترایاک با قابلیت تطبیق خودکار بر اساس ولتاژ خروجی و فرکانس ورودی فراهم می‌کند. با این طراحی:

- مدار توانایی تنظیم دقیق ولتاژ خروجی را دارد.
- قابلیت انطباق با فرکانس‌های مختلف 50Hz یا 60Hz وجود دارد.
- داده‌های لحظه‌ای برای نظارت و ارزیابی عملکرد ارائه می‌شود.

۱) نمونه‌برداری اولیه:

مدار در مدت ۳۰ ثانیه فرکانس ورودی را محاسبه می‌کند و زمان معادل هر درجه را تعیین می‌کند.

۲) تشخیص عبور از صفر:

در هر لحظه عبور از صفر، ولتاژ خروجی اندازه‌گیری شده و زاویه آتش بر اساس آن تنظیم می‌شود.

۳) کنترل تریگر ترایاک:

پس از محاسبه زمان تأخیر، پالس تریگر تولید می‌شود تا ترایاک در زاویه آتش مشخصی فعال شود.

۴) گزارش اطلاعات:

داده‌های مهم مانند ولتاژ، زاویه آتش، و زمان تأخیر به صورت زنده روی LCD و سریال مانیتور نمایش داده می‌شوند.

تحلیل و تجزیه کدهای پروژه

۱-۵ اندازه گیری فرکانس ورودی

در این بخش از کد، میکروکنترلر با تشخیص نقطه عبور از صفر (Zero-Cross Detection) توسط فتوترانزیستور H11AA1، لحظات عبور از صفر سیگنال AC را ثبت می‌کند و سپس با محاسبه زمان بین دو نقطه عبور از صفر متوالی، فرکانس سیگنال AC ورودی محاسبه می‌شود. این فرکانس برای تنظیم دقیق زاویه آتش در فازهای مختلف سیگنال AC اهمیت دارد.

```
// Define the input/output pins
#define zeroCrossingPin 2

// period of pulse accumulation and serial output, milliseconds
const int MainPeriod = 100;
long previousMillis = 0; // will store last time of the cycle end

volatile unsigned long previousMicros = 0;
volatile unsigned long duration = 0; // accumulates pulse width
volatile unsigned int pulseCount = 0; // accumulates pulse count

// interrupt handler
void freqCounterCallback() {
    unsigned long currentMicros = micros();
    duration += currentMicros - previousMicros;
    previousMicros = currentMicros;
    pulseCount++;
}

void reportFrequency() {
    float freq = 1e6 / float(duration) * float(pulseCount/2);
    Serial.print("Freq:");
    Serial.print(freq, 2);
    Serial.println(" Hz");

    // clear counters
    duration = 0;
    pulseCount = 0;
}

void setup() {
    Serial.begin(115200);
    pinMode(zeroCrossingPin, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(zeroCrossingPin), freqCounterCallback, RISING);
}

void loop() {
    unsigned long currentMillis = millis();
    if (currentMillis - previousMillis >= MainPeriod)
    {
        previousMillis = currentMillis;
        reportFrequency();
    }
}
```

* مثال محاسباتی

برای درک بهتر منطق ریاضی استفاده شده، به یک مثال عددی با فرض فرکانس ۵۰ هرتز می‌پردازیم.

اطلاعات پایه:

- پریود یک سیگنال ۵۰ هرتز برابر است با ۲۰ میلی ثانیه.
- یک سیگنال ۵۰ هرتز در هر ثانیه ۵۰ سیکل کامل دارد.
- هر سیکل دارای دو نقطه عبور از صفر است (از مثبت به منفی و بالعکس).

حال فرض کنید که کد اجرا شده و نتایج زیر را مشاهده می‌کنیم:

تابع وقفه freqCounterCallback() برای هر نقطه عبور از صفر فعال می‌شود و زمان بین آنها با استفاده از تابع micros() اندازه‌گیری می‌شود.

بنابراین پس از ۱ ثانیه، تابع وقفه ۱۰۰ بار فعال شده ($50 \text{ سیکل} \times 2 \text{ عبور از صفر در هر سیکل}$).

پس فرض می‌کنیم:

$$pulsecount = 100$$

$$duration = 1,000,000\mu\text{s}$$

در نتیجه با جایگذاری مقادیر بدست آمده در فرمول داریم:

$$freq = \frac{1e6}{duration} \times \frac{pulsecount}{2} = \frac{1 \times 10^6}{1,000,000} \times \frac{100}{2} = 50\text{Hz}$$

به دلیل آنکه مقدار pulsecount شامل دو عبور از صفر در هر سیکل است، باید برای محاسبه فرکانس تقسیم بر ۲ شود.

۵-۲ کنترل زاویه آتش و تولید پالس تریگر

بخش مهمی از کد، تنظیم زاویه آتش و تولید پالس تریگر برای تراپاک است. در این قسمت، میکروکنترلر پس از تشخیص لحظه عبور از صفر، مقدار تأخیر برای هر یک درجه را بدست آورده و سپس یک تأخیر زمانی بر اساس زاویه آتش دلخواه محاسبه می‌کند.

در ادامه خواهیم دید که این تأخیر به دلیل آنکه با توجه به فرکانس ورودی تنظیم می‌شود، زمان دقیق فعال‌سازی گیت تراپاک را محاسبه می‌کند.

```
// Define the input/output pins
#define zeroCrossingPin 2
#define TriacTrigger 10

// Firing angle in degrees
const int firingAngle = 60;
const float timePerDegree = 55.56; // Time per degree in microseconds (for 50 Hz)

// Delay corresponding to 60 degrees
const unsigned long delayTime = firingAngle * timePerDegree; // in microseconds

// Flag to indicate when to fire TRIAC
volatile bool fireTriac = false;

// interrupt handler
void fireTriacCallback() {
    // Set the flag to fire the TRIAC after the calculated delay
    fireTriac = true;
}

void setup() {
    Serial.begin(115200);
    pinMode(TriacTrigger, OUTPUT);
    pinMode(zeroCrossingPin, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(zeroCrossingPin), fireTriacCallback,
RISING);
}

void loop() {
    if (fireTriac) {
        // After detecting the zero-crossing, delay for 60 degrees (3.33 ms)
        delayMicroseconds(delayTime);

        // Trigger the TRIAC (fire it by setting the pin HIGH briefly)
        digitalWrite(TriacTrigger, HIGH);
        delayMicroseconds(10); // Fire pulse width to trigger TRIAC (adjust as needed)
        digitalWrite(TriacTrigger, LOW);

        // Reset the flag
        fireTriac = false;
    }
}
```

✖ مثال محاسباتی

فرض کنیم که فرکانس برق شهر ۵۰ هرتز است و دوره تناوب برابر با ۲۰ میلی ثانیه است.

برای محاسبه مدت زمان تأخیر به ازای هر درجه داریم:

$$timePerDegree = \frac{T}{360} = \frac{20ms}{360} = 55.56\mu s$$

حال اگر زاویه آتش را ۶۰ درجه در نظر بگیریم، زمان تأخیر متناظر با این زاویه از طریق رابطه زیر محاسبه می شود:

$$delayTime = firingAngle \times timePerDegree$$

در نتیجه:

$$55.56\mu s \times 60^\circ = 3.33ms$$

بنابراین، پس از تشخیص عبور از صفر، باید ۳.۳۳ میلی ثانیه تأخیر اعمال شود و سپس پالس گیت به ترایاک اعمال شود.

۵-۳ اندازه گیری ولتاژ AC خروجی

در این قسمت از کد، مژول **ZMPT101B** به میکروکنترلر متصل شده و مقدار **RMS** ولتاژ خروجی مدار را به طور مداوم اندازه گیری می کند. این داده ها به میکروکنترلر ارسال شده و نمایش داده می شود. در کد نهایی امکان تنظیم دقیق زاویه آتش بر اساس ولتاژ خروجی فراهم می گردد.

○ کالیبراسیون

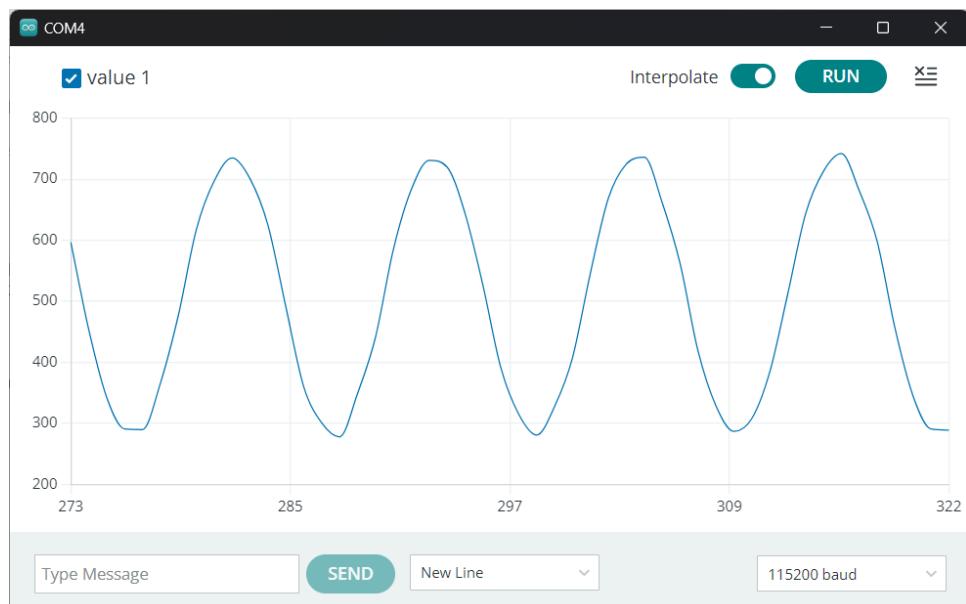
ابتدا کد زیر را آپلود کرده و سپس پنجره **Serial plotter** را باز کنید.

اگر ولتاژ **220v AC, 50Hz** را به ورودی اعمال کنید، باید نمودار سینوسی ولتاژ را مشاهده کنید.

* نکته: اگر شکل موج مشاهده شده در قسمت پیک ولتاژ برش خورده بود باید به کمک پتانسیومتر قرار گرفته روی مژول خروجی را کالیبره کرد تا هر دو پیک مثبت و منفی را داشته باشیم.

```
void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
}

void loop() {
    Serial.println(analogRead(A0));
    delay(20);
}
```



* برای کالیبراسیون سنسور:

۱. سنسور را به یک منبع ولتاژ AC مشخص وصل کنید.
۲. مقادیر حداکثر و حداقل خروجی سنسور (مقادیر ADC) را ثبت کنید.
۳. ولتاژ واقعی AC را با استفاده از یک مولتی‌متر اندازه بگیرید و ثبت کنید.
۴. داده‌های ثبت شده را برای بدست آوردن معادله کالیبراسیون استفاده کنید.

* برای پیدا کردن معادله کالیبراسیون، نیاز داریم تا یک خط مستقیم را به داده‌های نمونه برازش دهیم. این کار به روش رگرسیون خطی انجام می‌شود. برای داده‌های نمونه شامل مقادیر آنالوگ خوانده شده از سنسور و ولتاژهای واقعی، معادله خطی به شکل زیر است:

$$V_{actual} = m \times Reading + b$$

که در آن: m شیب خط و b عرض از مبدأ
حال فرض کنید که داده‌های شما به صورت دو آرایه به نام‌های x و y به ترتیب برای مقادیر خوانده شده از سنسور و ولتاژهای واقعی باشند.

$$x = [x_1, x_2, \dots, x_n]$$

$$y = [y_1, y_2, \dots, y_n]$$

$$m = \frac{N \sum (x_i y_i) - \sum x_i \sum y_i}{N \sum x_i^2 - (\sum x_i)^2}$$
$$b = \frac{\sum y_i - m \sum x_i}{N}$$

که در آن:

N نشان‌دهنده تعداد نمونه‌ها و \sum نشان‌دهنده جمع مقادیر است.

* مثال محاسباتی

فرض کنید داده‌های زیر را دارید:

$$x = [350, 420, 512, 680, 760]$$

$$y = [110, 150, 220, 300, 350]$$

محاسبات را به صورت زیر انجام می‌دهیم:

$$\sum x = 350 + 420 + 512 + 680 + 760 = 2722$$

$$\sum y = 110 + 150 + 220 + 300 + 350 = 1130$$

$$\sum (x_i^2) = 350^2 + 420^2 + 512^2 + 680^2 + 760^2 = 1601044$$

$$\begin{aligned} \sum (x_i y_i) &= (350 \times 110) + (420 \times 150) + (512 \times 220) + (680 \times 300) \\ &\quad + (760 \times 350) = 684140 \end{aligned}$$

در نتیجه:

$$m = \frac{5 \times 684140 - 2722 \times 1130}{5 \times 1601044 - 2722^2} \approx 0.578653$$

$$b = \frac{1130 - m \times 2722}{5} \approx -89.752$$

$$V_{actual} = (0.58) \times Reading + (-89.752)$$

* شبیه‌سازی متلب

```
clc
close all
clear all

% Sample data
analogReadings = [350, 420, 512, 680, 760]; % Raw sensor readings
actualVoltages = [110, 150, 220, 300, 350]; % Actual AC voltages

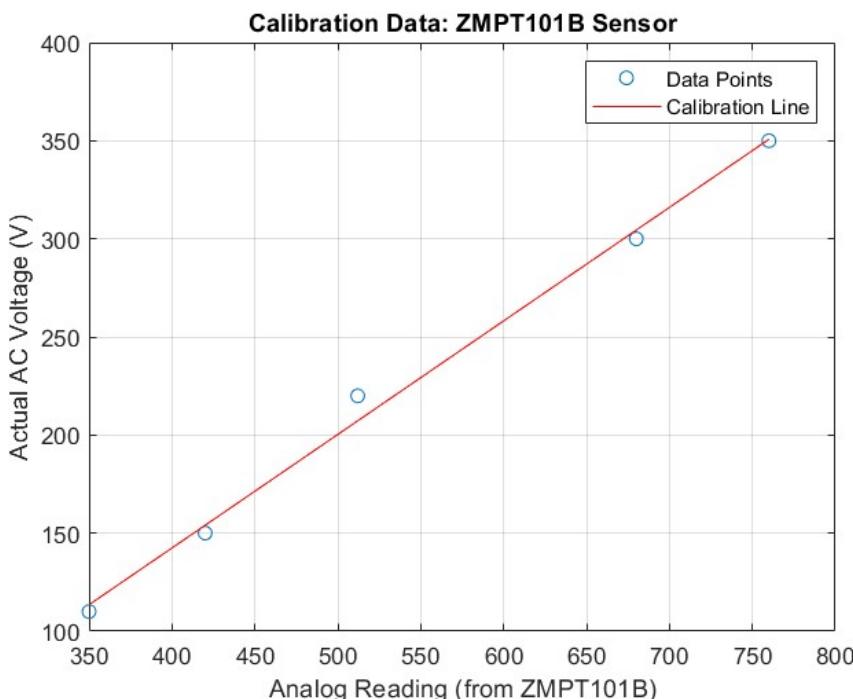
% Plot the raw data
figure;
plot(analogReadings, actualVoltages, 'o');
xlabel('Analog Reading (from ZMPT101B)');
ylabel('Actual AC Voltage (V)');
title('Calibration Data: ZMPT101B Sensor');
grid on;
hold on;

% Linear regression using polyfit
p = polyfit(analogReadings, actualVoltages, 1); % Linear fit (y = p(1)*x + p(2))
fitLine = polyval(p, analogReadings);
plot(analogReadings, fitLine, '-r');
legend('Data Points', 'Calibration Line');

% Display equation
fprintf('Calibration Equation: Voltage = %.2f * Reading + %.2f\n', p(1), p(2));
```

خروجی کد:

>> **Calibration Equation: Voltage = 0.58 * Reading + -89.02**



○ نمونه گیری و کالیبراسیون با میکروکنترلر

✓ کد اول (کالیبراسیون):

```
int adc_max = 0;      // Variable to store the maximum sensor value
int adc_min = 1023;    // Variable to store the minimum sensor value
long tiempo_init;     // Variable to store the initial time

void setup() {
  Serial.begin(115200); // Initialize the serial communication
  tiempo_init = millis(); // Get the current time in milliseconds
}

void loop() {
  // Check if 500 milliseconds have passed
  if ((millis() - tiempo_init) > 500) {
    adc_max = 0;          // Reset the maximum sensor value
    adc_min = 1023;        // Reset the minimum sensor value
    tiempo_init = millis(); // Update the initial time
  }

  int sensorValue = analogRead(A0); // Read the analog input from pin A0

  if (sensorValue > adc_max) {
    adc_max = sensorValue; // Update the maximum value if a new maximum is found
  } else if (sensorValue < adc_min) {
    adc_min = sensorValue; // Update the minimum value if a new minimum is found
  }

  // Print the maximum and minimum values to the serial monitor
  Serial.print("adc_max: ");
  Serial.print(adc_max);
  Serial.print(" adc_min: ");
  Serial.println(adc_min);

  delay(1); // Small delay for stability between each iteration
}
```

این کد پس از یک بازه ۵۰۰ میلیثانیه‌ای، با مشاهده مقادیر حداکثر و حداقل ADC می‌توان پیک مثبت و منفی سیگنال ورودی AC را شناسایی کند و به کمک آن خروجی سنسور را به محدوده ولتاژ واقعی نگاشت.

همچنین در طی این مرحله کالیبراسیون، با استفاده از مولتی‌متر ولتاژ AC واقعی را اندازه‌گیری کرده و این مقدار را ثبت کنید. این مقدار به عنوان مرجع برای محاسبه فاکتور تبدیل استفاده می‌شود.

نکته: سعی کنید که پتانسیومتر را طوری تنظیم کنید که `adc_max` به حدود ۷۰۰ یا بیشتر برسد تا مطمئن شوید که کل محدوده سیگنال ورودی را دریافت می‌کنید.

همانطور که میدانیم، آردوینو ولتاژها را در محدوده ۰ تا ۵ ولت با مقادیر ADC از ۰ تا ۱۰۲۳ می‌خواند و `adc_min` و `adc_max` به ترتیب نشان‌دهنده نقاط اوج سیگنال AC هستند.

✓ کد دوم (اندازه‌گیری ولتاژ :**RMS**

```
// Calibration values obtained from the sketch: volt_ac_cal
int adc_max = 760; // Maximum sensor value during calibration
int adc_min = 261; // Minimum sensor value during calibration

float volt_multi = 231; // RMS voltage obtained from a multimeter
float volt_multi_p; // Peak voltage
float volt_multi_n; // Negative peak voltage

void setup() {
  Serial.begin(115200);

  volt_multi_p = volt_multi * 1.4142; // Peak voltage = RMS voltage * 1.4142
  volt_multi_n = -volt_multi_p; // Negative peak voltage
}

void loop() {
  float volt_rms = get_voltage(); // Root Mean Square voltage (V-RMS)

  Serial.print("Vrms: ");
  Serial.print(volt_rms, 3);
  Serial.println(" VAC");

  // Delay for a certain interval if needed
  //delay(100);
}

float get_voltage() {
  float adc_sample;
  float volt_inst = 0;
  float sum = 0;
  float volt;
  long init_time = millis();
  int N = 0;

  // Duration of 0.5 seconds
  while ((millis() - init_time) < 500) {
    adc_sample = analogRead(A0); // Sensor voltage
    volt_inst = map(adc_sample, adc_min, adc_max, volt_multi_n, volt_multi_p);
    sum += sq(volt_inst); // Sum of Squares
    N++;
    delay(1);
  }

  //Serial.print("N: ");
  //Serial.println(N);

  volt = sqrt(sum / N); // RMS equation
  return volt;
}
```

این کد نیز از مقادیر کالیبراسیون به دست آمده از کد اول برای اندازه‌گیری دقیق ولتاژ **RMS** استفاده می‌کند.
در ادامه، نحوه عملکرد کد به صورت مرحله‌به‌مرحله شرح داده شده است:

$$V_{RMS} = \sqrt{\frac{\text{sum of squares of samples}}{\text{number of samples}}}$$

* مثال محاسباتی

فرض کنید:

$$adc_{max} = 760$$

$$adc_{min} = 261$$

$$volt_{multi} = 231 V_{RMS}$$

با توجه به روابط تک فاز داریم:

$$V_{peak} = V_{RMS} \times \sqrt{2} = 231 \times 1.4142 \approx 326.47V$$

در تابع `get_voltage()` به نمونه برداری از `adc` در بازه زمانی ۵۰۰ میلی ثانیه می پردازیم.
حال باید حداقل و حداقل مقدار آنالوگ اندازه گیری شده توسط `adc` را به نحوی در مقیاس $\pm 326.47V$ انجام میشود.

```
volt_inst = map(adc_sample, adc_min, adc_max, volt_multi_n, volt_multi_p);
```

اگر فرض کنیم که در این بازه زمانی $N = 5$ نمونه برداشته شده است،

$$ADC\ samples = [300, 500, 760, 400, 600]$$

در این صورت برای هر نمونه داریم:

$$V_{inst} = \frac{(sample - adc_{min})}{(adc_{max} - adc_{min})} \times (V_{peak} - (-V_{peak})) - V_{peak}$$

که مثلا با جایگذاری مقادیر برای نمونه اول داریم:

$$V_{inst} = \frac{(300 - 261)}{(760 - 261)} \times (326.42 - (-326.42)) \approx -275.54V$$

اکنون با محاسبه مجذور ولتاژهای بدست آمده و جمع آنها داریم:

$$sum = \sum_{i=1}^N V_i^2$$

$$sum = (-256.88)^2 + (0.28)^2 + (326.47)^2 + (-124.26)^2 + (162.53)^2$$

که با جایگذاری در رابطه محاسبه ولتاژ موثر خواهیم داشت:

$$V_{RMS} = \sqrt{\frac{214424.12}{5}} \approx 207.10V$$

○ معرفی کتابخانه zmpt101b.h

✓ کد اول (کالیبراسیون):

```
/*
 * This program shows how we can get (estimate) the right sensitivity value for
 * the sensor.
 *
 * This program will swipe from the lowest sensitivity estimate value to the
 * highest. The program will stop if the voltage reading result from the sensor
 * is within the specified tolerance limit or the sensitivity value has exceeded
 * the highest predetermined value (which in this case is considered failed to
 * be determined).
*/
#include <ZMPT101B.h>

#define ACTUAL_VOLTAGE 220.0f // Change this based on actual voltage

#define START_VALUE 0.0f
#define STOP_VALUE 1000.0f
#define STEP_VALUE 0.25f
#define TOLERANCE 1.0f

#define MAX_TOLERANCE_VOLTAGE (ACTUAL_VOLTAGE + TOLERANCE)
#define MIN_TOLERANCE_VOLTAGE (ACTUAL_VOLTAGE - TOLERANCE)

// ZMPT101B sensor output connected to analog pin A0
// and the voltage source frequency is 50 Hz.
ZMPT101B voltageSensor(A0, 50.0);

void setup() {
    Serial.begin(115200);
    Serial.print("The Actual Voltage: ");
    Serial.println(ACTUAL_VOLTAGE);

    float sensitivityValue = START_VALUE;
    voltageSensor.setSensitivity(sensitivityValue);
    float voltageNow = voltageSensor.getRmsVoltage();

    Serial.println("Start calculate");

    while (voltageNow > MAX_TOLERANCE_VOLTAGE || voltageNow <
MIN_TOLERANCE_VOLTAGE) {
        if (sensitivityValue < STOP_VALUE) {
            sensitivityValue += STEP_VALUE;
            voltageSensor.setSensitivity(sensitivityValue);
            voltageNow = voltageSensor.getRmsVoltage();
            Serial.print(sensitivityValue);
            Serial.print(" => ");
            Serial.println(voltageNow);
        } else {
            Serial.println("Unfortunately the sensitivity value cannot be determined");
            return;
        }
    }

    Serial.print("Closest voltage within tolerance: ");
    Serial.println(voltageNow);
    Serial.print("Sensitivity Value: ");
    Serial.println(sensitivityValue, 10);
}

void loop() {}
```

✓ کد دوم (اندازه‌گیری ولتاژ :RMS)

```
/**  
 * This program shows you how to use the basics of this library.  
 */  
#include <ZMPT101B.h>  
#define SENSITIVITY 500.0F  
  
// ZMPT101B sensor output connected to analog pin A0  
// and the voltage source frequency is 50 Hz.  
ZMPT101B voltageSensor(A0, 50.0);  
  
void setup() {  
    Serial.begin(115200);  
    // Change the sensitivity value based on value you got from the calibrate  
    // example.  
    voltageSensor.setSensitivity(SENSITIVITY);  
}  
  
void loop() {  
    // read the voltage and then print via Serial.  
    float voltage = voltageSensor.getRmsVoltage();  
    Serial.println(voltage);  
  
    delay(1000);  
}
```

* توجه

دیدیم که برای استفاده دقیق و اصولی از این ماثول با دردرس‌های زیادی همانند روابط و فرمول‌های سخت و پیچیده و یا حتی خطای اندازه‌گیری و محاسباتی مواجه هستیم.

برای سهولت استفاده از این ماثول و همینطور افزایش دقت اندازه‌گیری، توصیه می‌شود از کتابخانه مخصوص این قطعه استفاده شود که در دو مثال بالا به نحوه کار با این کتابخانه اشاره شده است.

با آپلود کد اول روی میکروکنترلر و گرفتن خروجی سریال پس از طی شدن چند دقیقه و انجام محاسبات، در نهایت ضریب حساسیت ماثول به شما نمایش داده می‌شود، اما قبل از آپلود توجه کنید که مقدار پارامتر ACTUAL_VOLTAGE در واقع همان ولتاژی است که به کمک مولتی‌متر اندازه‌گیری می‌شود و باید با توجه به مقدار اندازه‌گیری شده توسط خودتان آن را تنظیم نمایید.

در کد دوم نیز با تغییر پارامتر SENSITIVITY با توجه به مقداری که در کد قبل به دست آمد به اندازه‌گیری دقیق ولتاژ می‌پردازیم.

۵-۴ جمع بندی و کد نهایی

در این بخش، کدهای پروژه برای ارائه عملکرد نهایی سیستم ترکیب شده‌اند. تمامی بخش‌ها از جمله اندازه‌گیری فرکانس، تنظیم زاویه آتش، و اندازه‌گیری ولتاژ خروجی در کنار هم عمل کرده و سیستم کنترل ولتاژ را تشکیل می‌دهند.

کد نهایی باید به گونه‌ای نوشته شود که به صورت پیوسته و بلاذرنگ عمل کند، درنتیجه بهینه‌سازی‌های انجام شده برای اطمینان از اجرای روان و کارآمد پروژه عبارتند از:

- ✓ مدیریت زمان‌بندی مناسب برای اندازه‌گیری‌ها و تأخیرهای مربوط به زاویه آتش
- ✓ استفاده از روش‌های بهینه برای پردازش سیگنال و کاهش زمان اجرای کد
- ✓ پیاده‌سازی مکانیسم‌های ایمنی برای جلوگیری از بروز خطأ در شرایط غیرعادی

```

///////////////////////////////////////////////////////////////////
// CLOSING LOOP CONTROL
// TRIAC FIRING ANGLE SYSTEM
///////////////////////////////////////////////////////////////////

#include <Arduino.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//===== LCD =====//
// LCD connect to Arduino I2C (A4 = SDA, A5 = SCL)
LiquidCrystal_I2C lcd(0x27, 16, 2);

//===== Zero-Crossing =====//
#define Interrupt0 2
#define Interrupt1 3

//===== Measure Frequency =====//
// period of pulse accumulation and serial output, milliseconds
const int MainPeriod = 100;
long previousMillis = 0; // will store last time of the cycle end

volatile unsigned long previousMicros = 0;
volatile unsigned long duration = 0; // accumulates pulse width
volatile unsigned int pulsecount = 0; // accumulates pulse count

//===== TRIAC =====//
#define TriacTrigger 10

// Firing angle in degrees
volatile int firingAngle = 90;

// Time per degree calculation (~55.56us for 50Hz & 46.29us for 60Hz)
volatile float freq = 50;
volatile float timePerDegree = 55.56;

// Delay corresponding to chosen degrees
volatile unsigned long delayTime = firingAngle * timePerDegree; // in microseconds

//===== ZMPT10B =====//
#define zmptPin A0
int zmptSensor = 0;
int voltageRef = 220;
int measuredVoltage = 0;

//-----
// External Interrupt 1 (Zero-Crossing detection) service routine
void freqCounterCallback() {
    unsigned long currentMicros = micros();
    duration += currentMicros - previousMicros;
    previousMicros = currentMicros;
    pulsecount++;
}

//-----
void reportFrequency() {
    freq = 1e6 / float(duration) * float(pulsecount/2);

    Serial.print("duration: ");
    Serial.println(float(duration) , 2);
    Serial.print("pulsecount: ");
    Serial.println(float(pulsecount/2) , 2);
    Serial.print("Freq: ");
    Serial.print(freq , 2);
    Serial.println(" Hz");

    lcd.setCursor ( 0, 1 );
    lcd.print("Freq: ");
    lcd.print(freq , 2);
    lcd.print(" Hz");
}

```

```

// clear counters
duration = 0;
pulsecount = 0;

// Time per degree calculation (based on the full AC cycle)
timePerDegree = (1e6 / freq) / 360; // in microseconds
}

//-----
// External Interrupt 0 (Zero-Crossing detection) service routine
void fireTriacCallback() {
    // Read the voltage from the ZMPT10B sensor
    zmptSensor = analogRead(zmptPin);

    // Map the sensor reading to a voltage range
    measuredVoltage = (zmptSensor / 1023.0) * voltageRef; // Assuming 0-220V mapped

    // Adjust firing angle based on the voltage range
    switch (measuredVoltage) {
        case 0 ... 100:
            firingAngle -= 5;
            if (firingAngle < 30) firingAngle = 30; // Prevent negative firing angles
            break;

        case 120 ... 220:
            firingAngle += 5;
            if (firingAngle > 150) firingAngle = 150; // Prevent angle beyond 180 degrees
            break;

        default:
            firingAngle = 90;
            break;
    }

    // Delay corresponding to reference voltage
    delayTime = firingAngle * timePerDegree;

    // Generate TRIAC trigger pulse
    delayMicroseconds(delayTime); // Delay before firing TRIAC
    digitalWrite(TriacTrigger, HIGH);
    delayMicroseconds(50); // 50us pulse to trigger TRIAC
    digitalWrite(TriacTrigger, LOW);
}

//=====
void setup() {
    // Initialize serial communication
    Serial.begin(115200);

    //Initialize lcd screen
    lcd.init();
    lcd.backlight();
    lcd.clear();

    // Set zero-crossing pin as input with internal pull-up
    pinMode(Interrupt0, INPUT_PULLUP);
    pinMode(Interrupt1, INPUT_PULLUP);

    // Set TriacTrigger pin as output (TRIAC control)
    pinMode(TriacTrigger, OUTPUT);

    attachInterrupt(digitalPinToInterrupt(Interrupt1), freqCounterCallback, FALLING);

    // Allow time to collect pulses before reporting frequency
    lcd.setCursor(0,0);
    lcd.print("Processing..."); 
    delay(30000); // Adjust delay as needed to accumulate a meaningful number of
    pulses
    reportFrequency();
    delay(5000);
    lcd.clear();
}

```

```

//=====
void loop() {
    // Print debug information
    Serial.print("Voltage: ");
    Serial.print(measuredVoltage);
    Serial.print("V, Firing Angle: ");
    Serial.print(firingAngle);
    Serial.print(" degrees, Delay: ");
    Serial.print(delayTime);
    Serial.println("us");

    lcd.setCursor(0,0);
    lcd.print("Voltage: ");
    lcd.print(measuredVoltage);
    lcd.print(" V");
    lcd.setCursor(0,1);
    lcd.print((char)0xF2);
    lcd.print("=");
    lcd.print(firingAngle);
    lcd.print((char)0xDF);
    lcd.print(" , ");
    lcd.print(delayTime);
    lcd.print("us");

    // Attach interrupt handler for zero-crossing detection (FALLING edge)
    attachInterrupt(digitalPinToInterrupt(Interrupt0), fireTriacCallback, FALLING);
}

///////////////////////////////
//          © 2019 - 2024 Nariman Ziaie, All rights reserved
//          see: https://github.com/Nariman-Z for more information
/////////////////////////////

```

۵-۵ پیشنهادات برای بهبود سیستم و توسعه‌های آتی

در این بخش، پیشنهادات برای بهبود عملکرد سیستم و توسعه‌های آتی مطرح می‌شوند. این پیشنهادات بر اساس تجربه تست عملی، تحلیل عملکرد سیستم و امکان سنجی توسعه‌های بیشتر ارائه شده‌اند:

۱. بهبود دقت اندازه‌گیری و پردازش

الف) استفاده از ماظول‌های ADC دقیق‌تر

استفاده از یک ماظول ADC خارجی با دقت بالاتر (مانند ADC با دقت ۱۶ یا ۲۴ بیت) می‌تواند دقت اندازه‌گیری ولتاژ ورودی را بهبود بخشد. این تغییر، تأثیر خطاهای ناشی از محدوده ۱۰ بیتی ADC داخلی آردوینو را کاهش می‌دهد.

ب) ارتقاء روش نمونه‌برداری

افزایش نرخ نمونه‌برداری از سیگنال ورودی با استفاده از الگوریتم‌های پیشرفته‌تر یا استفاده از DMA (Direct Memory Access) در پلتفرم‌هایی مانند ESP32 می‌تواند تحلیل سیگنال را بهبود بخشد.

استفاده از فیلترهای دیجیتال مانند فیلتر میانگین متحرک (Moving Average) یا فیلتر کالمن برای حذف نویز.

ج) کالیبراسیون خودکار

افزودن مکانیزم کالیبراسیون خودکار با استفاده از یک رله برای اتصال ولتاژ مرجع به مدار و تنظیم ضرایب تبدیل ADC به ولتاژ واقعی.

۲. بهینه‌سازی عملکرد کنترل ترایاک

الف) کنترل دقیق‌تر زاویه آتش

استفاده از میکروکنترلر با تایمیرهای پیشرفته‌تر برای ایجاد پالس‌های تریگر با دقت نانوثانیه. این تغییر برای کاربردهای حساس مانند کنترل دما یا توان دقیق ضروری است.

ب) کنترل تطبیقی (Adaptive Control)

طراحی الگوریتم‌های کنترل تطبیقی برای تنظیم زاویه آتش بر اساس تغییرات لحظه‌ای در بار یا ولتاژ ورودی. این امر می‌تواند باعث بیهود پایداری سیستم در مواجهه با تغییرات ناگهانی شود.

ج) استفاده از الگوریتم PID

پیاده‌سازی یک کنترل‌کننده PID (Proportional-Integral-Derivative) برای مدیریت بهتر ولتاژ خروجی و کاهش زمان پاسخ به تغییرات بار.

۳. ارتقاء ایمنی و ایزولاسیون مدار

الف) افزایش ایمنی در مدار زیرو کراس

استفاده از اپتوکوپلرهایی با سرعت بالاتر یا طراحی مدارهای مقاوم‌تر در برابر نویز، برای افزایش دقت و ایمنی در تشخیص عبور از صفر.

ب) محافظت در برابر اضافه ولتاژ

افزودن یک مدار محافظتی مانند TVS (Transient Voltage Suppressor) یا MOV (Metal Oxide Varistor) برای محافظت از مدار در برابر اضافه ولتاژ‌های ناگهانی.

ج) ایزولاسیون گالوانیکی در اندازه‌گیری ولتاژ

برای افزایش ایمنی، استفاده از ماظولهای ایزولاسیون گالوانیکی برای اندازه‌گیری ولتاژ ورودی پیشنهاد می‌شود.

۴. بهینه‌سازی رابط کاربری

الف) افزودن نمایشگر بهتر

استفاده از نمایشگرهای گرافیکی (مانند **TFT** یا **OLED**) برای نمایش اطلاعات بیشتر و با وضوح بالاتر برای نمایش نمودارهای لحظه‌ای برای ولتاژ، زاویه آتش و توان خروجی.

ب) کنترل از طریق رابط بی‌سیم

افزودن قابلیت ارتباط بی‌سیم (مانند **Wi-Fi** یا **Bluetooth**) برای کنترل و نظارت بر عملکرد سیستم از طریق اپلیکیشن یا مرورگر.

ج) رابط کاربری پیشرفته

طراحی یک واسط کاربری ساده‌تر برای تنظیم پارامترهایی مانند ولتاژ مرجع، زاویه آتش، و محدوده عملکرد.

۵. قابلیت توسعه برای کاربردهای پیشرفته‌تر

الف) کنترل چند کاناله

طراحی سیستم برای کنترل همزمان چندین ترایاک یا کانال. این ویژگی می‌تواند در کاربردهایی مانند کنترل روشنایی چندگانه یا بارهای پیچیده مورد استفاده قرار گیرد.

ب) قابلیت ذخیره داده‌ها

افزودن حافظه داخلی یا کارت **SD** برای ذخیره مقادیر ولتاژ، زاویه آتش، و سایر پارامترها برای تحلیل‌های آتی.

ج) پشتیبانی از بارهای غیرخطی

طراحی مدارهای محافظتی و کنترلی برای مدیریت بارهای غیرخطی یا بارهایی با ضربیت توان پایین.

۶. ارتقاء سخت‌افزار و نرم‌افزار

الف) استفاده از میکروکنترلر قدرتمندتر

مهاجرت به پلتفرم‌های پیشرفته‌تر مانند **ESP32** یا **STM32** برای بهره‌گیری از قابلیت‌های بیشتر در کنترل دقیق و مدیریت داده.

ب) بهینه‌سازی نرم‌افزار

کاهش تأخیر در پردازش با بهینه‌سازی کد و استفاده از برنامه‌نویسی مبتنی بر وقفه (Interrupt-Based Programming).

۷. توسعه برای کاربردهای صنعتی

الف) کنترل توان در مقیاس صنعتی

طراحی سیستم برای کنترل بارهای صنعتی بزرگ‌تر مانند موتورهای AC یا سیستم‌های گرمایشی.

استفاده از ترایاک‌های با جریان بالا و طراحی مدارهای خنک‌کننده مناسب.

ب) توسعه قابلیت‌های اندازه‌گیری

افزودن مازول‌های اندازه‌گیری جریان برای محاسبه توان مصرفی و بهره‌وری سیستم.

ج) طراحی نسخه صنعتی مقاوم

ساخت مدار با قطعات مقاوم در برابر شرایط سخت مانند دما یا رطوبت بالا.

نتیجه‌گیری

این پیشنهادات می‌توانند به بهبود عملکرد سیستم، افزایش دقت و ایمنی، و گسترش کاربردهای آن کمک کنند. اجرای برخی از این تغییرات باعث افزایش انعطاف‌پذیری سیستم و قابلیت استفاده در پژوهش‌های پیشرفته‌تر و صنعتی خواهد شد.

جمع‌بندی

✓ توضیحات تکمیلی

این پروژه یک سیستم کنترل زاویه آتش تراپاک را طراحی و پیاده‌سازی می‌کند که با شناسایی لحظه عبور از صفر و تنظیم دقیق زمان تأخیر، ولتاژ خروجی را کنترل می‌کند.

مدار با استفاده از میکروکنترلر، سنسور ولتاژ، و اپتوكوپلر، دقت و ایمنی بالایی در عملکرد دارد. قابلیت‌های نظارتی و انعطاف‌پذیری آن، این سیستم را برای کاربردهای مختلف از جمله کنترل روشنایی و تنظیم توان مناسب می‌سازد و همچنین پیشنهادات ارائه شده، امکان ارتقاء سیستم برای مصارف صنعتی و پیشرفته را فراهم می‌کند.



جهت مشاهده کدهای برنامه و فایل شبیه‌سازی لینک بالا را اسکن نمایید