به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

آزمایشگاه سیستم های کنترل دیجیتال

گزارش کار کنترل دور و موقعیت موتور

نویسندگان گزارش:

محمد مشرقی (۸۱۰۱۹۹۴۹۲)

مهدی معراجی (۸۱۰۱۹۸۴۷۴)

بهنام رنجبر (۸۱۰۱۹۹۴۳۰)

نیمسال دوم ۱۴۰۲-۱۴۰۳

فهرست

سیستم شناسایی موتور DC:

این قسمت که آزمایش دوم این آزمایشگاه می‌باشد، به بررسی شناسایی موتور DC پرداخته شده است. در این آزمایش، با استفاده از ماشین حالت، یک الگوریتمی را جهت شناسایی پارامترهای سیستم پیاده سازی کرده ایم.

در قدم اول، نیاز هست که مقدار حداقل ولتاژ مورد نیاز برای راه‌اندازی موتور را بدست آوریم. دلیل این کار این هست که این موتور دارای یک ناحیه مرده هست که اگر ولتاژ در این ناحیه باشد، موتور راه‌اندازی نمی‌شود. بنابراین، با اضافه کردن ولتاژ به صورت پله پله از مقدار صفر به سمت بالا، جایی که باعث راه‌اندازی موتور می‌شود، همان حداثل ولتاژ مورد نیاز برای راه‌اندازی موتور می‌باشد.

در قدم بعد، با توجه به اینکه موتور را به صورت سیسیتم مرتبه اول مدل می‌کنیم باید مقدار ثابت زمانی را برای این موتور محاسبه کنیم. جهت اینکار، پس از اینکه موتور در مقدار معینی سرعت ثابت شد، به آن، یک پله وارد می‌کنیم و زمانی را که طول می‌کشد که موتور ۰.۶۳ مقدار پله داده شده را طی کند، به عنوان ثابت زمانی موتور در نظر می‌گیریم.

لازم به ذکر است که جهت راه‌اندازی و فرمان دادن به موتور از خروجی PWM میکروکنترلر استفاده شده است و همچنین، از یک ورودی وقفه خارجی و ورودی دیجیتال برای خواندن انکودر و یک خروجی دیجیتال برای تغییر جهت استفاده کرده ایم. با استفاده از پایه های A و B در انکودر، می‌توان جهت چرخش را تشخیص داد و سرعت را تعیین کرد و با استفاده از پایه Z نیز می‌توان مرجع زاویه و شروع چرخش را تعیین کرد.

لازم به ذکر است که در قسمت بعد، به توضیحات کامل در ارتباط با پایه‌های استفاده شده از میکروکنترلر می‌پردازیم و نکاتی که باید رعایت شود، ذکر شده است.

کنترل دور و موقعیت موتور

میکروکنترلر

در این آزمایش، همانطور که در ابتدای دستورکار آزمایش سوم نوشته شده است، از برخی قابلیت‌های میکروکنترلر استفاده شده است که در ادامه به مختصر توضیح داده می‌شود.

یکی از این قابلیت‌ها، تایمرها هستند. از یک تایمر جهت تولید سیگنال PWM و اعمال آن به موتور استفاده شده است. از یک تایمر دیگر جهت ایجاد وقفه ۱۰ میلی ثانیه‌ای استفاده شده است که جهت محاسبه سرعت به کار برده می‌شود. تایمر دیگری نیز جهت محاسبات مربوط به پارامترهای کنترل کننده مورد استفاده قرار می‌گیرد.

یکی دیگر از قابلیت‌ها، وقفه خارجی می‌باشد که به یک کلید متصل شده است و با فشردن این کلید، عملیات کنترل آغاز می‌شود.

دو وقفه خارجی دیگر برای سیگنال‌های A و Z انکودر و یک ورودی دیجیتال برای سیگنال B انکودر مورد استفاده قرار گرفته است که به کمک این سیگنال‌ها، سرعت موتور و جهت آن محاسبه می‌شوند.

توابع تعریف شده

در ابتدای این آزمایش خواسته شده است که سه تابع تعریف شود که به کمک این توابع، بتوان مقادیر سرعت، زاویه و جهت را بدست آورد. جهت محاسبه سرعت از یکی از تایمرها و تعداد وقفه‌های ایجاد شده انکودر استفاده می‌کنیم. در وقفه تایمر ششم که در آن سرعت را محاسبه می‌کنیم، جهت محاسبه سرعت از رابطه(۱) استفاده می‌کنیم. همچنین، جهت محاسبه زاویه در وقفه خارجی مطابق رابطه(۲) عمل می‌کنیم.

$$speed = \left(\frac{counter \times 12000}{1024}\right) \quad (1)$$

$$angle = \left(\frac{counter \times 360}{1024}\right) \quad (2)$$

لازم به ذکر است که هر بار سیگنال Z در میکروکنترلر مشاهده می‌شود، مقدار counter برابر صفر می‌شود. همچنین، از سیگنال B انکودر جهت بدست آوردن جهت حرکت استفاده می‌کنیم.

در ادامه، نیاز داریم که توابعی را تعریف کنیم که مقدار ولتاژ و جهت آن را به گونه ای که می‌خواهیم، تنظیم کنیم. بنابراین، دو تابع به نام‌های setDir و setDuty تعریف می‌کنیم که جهت حرکت و مقدار دیوتی سایکل PWM که متناسب با مقدار CCR1 می‌شود را تنظیم می‌کند. جهت تعیین مقدار ولتاژی که بتوان مقدار دیوتی سایکل را تنظیم کند، باید یک تابع دیگر تعریف کنیم که اینجا نام آن را Vout2PWM گذاشته ایم که اندازه ولتاژ بین ۰ تا ۱۲ را به عددی بین ۰ ۱۰۰۰ اسکیل می‌کند و به تابع setDuty ارسال می‌کند. لازم به ذکر است که در اینجا، جهت تنظیم کردن حرکت نیز از علامت ولتاژ استفاده می‌کنیم.

طراحی کنترل کننده PID

در مرحله بعد از پیاده‌سازی توابع اولیه، کنترل‌کننده‌های موقعیت و سرعت را طراحی و پیاده‌سازی کردیم. لازم به ذکر است که به دلیل نویز و تغییرات بالایی که می‌تواند وجود داشته باشد، از قسمت D در PID استفاده نمی‌کنیم. بنابراین، ابتدا یک کنترل‌کننده PI طراحی شد و هر زمان نیاز به کنترل‌کننده P داشتیم، ضریب انتگرال‌گیر را صفر قرار دادیم.

برای طراحی کنترل‌کننده، ابتدا نیاز به تعریف سیگنال خطا داریم. مقدار خطا در هر لحظه به عنوان اختلاف بین مقدار خروجی فعلی و مقدار مطلوب در نظر گرفته می‌شود. برای طراحی کنترل‌کننده PID، علاوه بر خطای فعلی، نیاز به اطلاعات خطای قبلی نیز داریم. این موضوع در مورد سیگنال کنترلی نیز صدق می‌کند؛ بنابراین در هر مرحله، مقدار قبلی خطا و سیگنال ورودی را ذخیره می‌کنیم. سیگنال کنترلی با استفاده از ضرایب مناسب Kp و Ki، رابطه زیر که در درون کد در وقفه تایمر ۷ نوشته شده است، محاسبه می‌شود:

```
if (htim->Instance == TIM7)
{
    if(state == INIT2 || state == INIT3){
        ek = Setpoint - speed;
        uk = uk_1 + ek * (1000*Kp + Ki * Ts)/1000 - ek_1 * Kp;
        ek_1 = ek;
        uk_1 = uk;

        Vout2PWM(uk);

        setDir(dir);
        setDuty(duty);
    }

}
```

طراحی کنترل کننده Dead Beat

با توجه به آنچه که در تمرین انجام شده بود، به ازای زمان نمونه برداری ۱۰ میلی‌ثانیه، تابع تبدیل کنترل کننده بدین صورت می‌شود:

$$G(z) = \frac{0.5074 - 0.4644\, z^{-1}}{0.6807 + 0.3353\, z^{-1}} \tag{3}$$

بنابراین، جهت پیاده سازی سیستم فوق در حوزه زمان، بدین ترتیب عمل می‌کنیم:

$$\frac{U(z)}{E(z)} = \frac{A_1 + A_2 z^{-1}}{B_1 + B_2 z^{-1}} \rightarrow B_1 U(z) + B_2 z^{-1} U(z) = A_1 E(z) + A_2 z^{-1} E(z) \tag{4}$$

$$\rightarrow B_1 u[k] + B_2 u[k-1] = A_1 e[k] + A_2 e[k-1] \tag{5}$$

$$\rightarrow u[k] = \frac{A_1 e[k] + A_2 e[k-1] - B_2 u[k-1]}{B_1} \tag{6}$$

بنابراین، با توجه رابطه (۶)، کد مربوط به این بخش نوشته شده است که بدین ترتیب می‌باشد:

```
if (htim->Instance == TIM7)
{
    if(state == INIT2 || state == INIT3){
        ek = Setpoint - pos;
        A1 = 0.5074;
        A2 = -0.4644;
        B1 = 0.6807;
        B2 = 0.3353;

        uk = (A1*ek + A2*ek_1 - B2*uk_1)/B1;

        ek_1 = ek;
        uk_1 = uk;

        Vout2PWM(uk);

        setDir(dir);
        setDuty(duty);
    }

}
```

چالش‌ها و پیشنهادات

در روند طراحی و پیاده‌سازی کنترل‌کننده PID و Dead Beat برای کنترل سرعت و موقعیت موتور، با چندین چالش اساسی روبرو شدیم که هر کدام نیازمند راه‌حل‌های خاص خود بودند. در ادامه، این چالش‌ها و راه‌حل‌های مرتبط با آن‌ها تشریح می‌شوند.

چالش اول: ناحیه مرده موتور

ناحیه مرده موتور اولین چالشی بود که با آن روبرو شدیم. این مشکل در تمامی بخش‌ها به چشم می‌خورد و نیاز بود تا آن را برطرف کنیم. با استفاده از اطلاعات به‌دست‌آمده از ناحیه مرده در مرحله قبلی، توانستیم موتور را در همان ابتدا از این ناحیه خارج کنیم. با این حال، در طراحی کنترل‌کننده deadbeat، مجدداً با مشکل ناحیه مرده مواجه شدیم و ناچار به استفاده از روش Sliding Mode شدیم. این روش نیز گاهی اوقات خطایی در حدود ۱۰ تا ۱۵ درجه ایجاد می‌کرد. بدون استفاده از این روش، در صورت کم بودن تغییرات مکانی، ممکن بود موتور اصلاً حرکت نکند و کنترل‌کننده بی‌فایده باشد.

چالش دوم: تعیین پارامترهای Kp و Ki

تعیین پارامترهای Kp و Ki برای مکان‌یابی موتور در ابتدا موجب ناپایداری سیستم می‌شد. اگر این پارامترها خیلی کوچک بودند، موتور نمی‌توانست از ناحیه اشباع عبور کند. با تنظیم دقیق و بهینه‌سازی این پارامترها، توانستیم به یک کنترل پایدار دست یابیم.

چالش سوم: دقت حسگرها و تعریف بازه خطا

باید توجه داشت که ممکن است به دلیل دقت حسگرها، خطا دقیقاً صفر نشود و مقدار کوچکی باقی بماند که قابل صرف‌نظر کردن است. در این حالت، سیستم کنترلی همواره در حال اعمال سیگنال برای جبران این خطای کوچک است که می‌تواند منجر به فرسودگی سیستم شود. بنابراین، تعریف یک بازه خطا برای سیستم ضروری است تا در صورت قرار گرفتن خطا در این بازه، سیگنال کنترلی صفر شود.

چالش چهارم: دوره نمونه‌برداری

یکی از نکات مهم در طراحی کنترل‌کننده، تاثیر دوره نمونه‌برداری بر مقدار سیگنال کنترلی است. اگر دوره نمونه‌برداری بزرگ باشد، ممکن است منجر به افزایش سیگنال کنترلی شود و حتی باعث ناپایداری سیستم گردد.

چالش پنجم: تغییر موقعیت از ۳۶۰ درجه به صفر و برعکس

مشکل خوردن سیستم هنگام تغییر موقعیت موتور از ۳۶۰ درجه به صفر و برعکس نیاز به تصحیح داشت. با ایجاد تعدیلات مناسب در الگوریتم کنترل‌کننده، این مشکل رفع شد.

چالش ششم: نویز در خروجی

وجود نویز بیش از حد در خروجی Velocity موتور، چالش دیگری بود که با آن مواجه شدیم. برای رفع این مشکل، خروجی را به Tacho متصل کرده و از این طریق نویز را کاهش دادیم. همچنین، شفت موتور نیز دچار مشکل بود و خوب نمی‌چرخید که این مشکل نیز باید مرتفع می‌شد.

چالش هفتم: مشکلات کدنویسی و آپلود

مجبور شدن به ران کردن مجدد کد و آپلود آن به دلیل اشتباهات کدنویسی یا ذخیره‌سازی نادرست متغیرها، زمان زیادی از دست می‌داد. وجود یک فایل آماده با توابع نوشته‌نشده و کامنت‌های راهنما می‌توانست کمک بزرگی باشد.

چالش هشتم: محدودیت زمانی

محدودیت زمانی نیز چالش بزرگی بود. بسیاری از اعضای گروه در درس ابزار دقیق و کار با برد STM32 تازه‌کار بودند. با وجود فیلم‌های آموزشی موجود در یوتیوب، همچنان کار با برد STM32 دشواری‌های خاص خود را داشت.

چالش نهم: محدودیت در تعداد بردهای آزمایشگاه

تعداد کم بردهای آزمایشگاه چالش دیگری بود. اگر گروهی پیشرفت کافی نداشت، باید روز دیگری به آزمایشگاه مراجعه می‌کردند. افزایش تعداد بردها می‌توانست به گروه‌های عقب‌مانده کمک کند تا در سانس‌های دیگری با حضور استاد به کار خود ادامه دهند.

نتیجه‌گیری

برطرف کردن هر یک از این چالش‌ها نیازمند توجه دقیق و استفاده از روش‌های مناسب بود. با گذر از این مراحل، توانستیم یک کنترل‌کننده PID پایدار و کارآمد و همچنین، یک کنترل کننده Dead Beat برای موتور طراحی و پیاده‌سازی کنیم که نیازهای ما را تا حدود خوبی برآورده سازد.