آزمایشگاه کنترل دیجیتال

دستور کار آزمایش سوم

کنترل دیجیتال دور و موقعیت موتور جریان_مستقیم

۱ مقدّمه

در آزمایش پایانی آزمایشگاه به کنترل دور و موقعیّت سامانه‌ی سروُ_موتور جریان_مستقیم خواهیم پرداخت. در جلسه‌های قبلی با تمام ابزارهای موردنیاز برای پیاده‌سازی یک کنترل‌کننده‌ی دیجیتال آشنا شده‌اید. برای شروع پروژه‌ای با مشخصات زیر ایجاد کنید:

۱. تنظیم کلاک اصلی میکروکنترلر روی ۱۶۸ مگاهرتز

۲. یک تایمر برای تولید سیگنال PWM با فرکانس ۱۵ کیلوهرتز و دقت دیوتی‌سایکل ۰/۱ درصد

۳. دو زمان‌سنج ابتدایی TIM6 و TIM7 در حالت تولید وقفه. تنظیم پارامترها با توجّه به شرایط مساله تعیین می‌شود.

۴. دو LED متّصل به پایه‌های PG4 و PG5

۵. دو وقفه‌ی خارجی متّصل به کلیدهای PF2 و PF3

۶. اتصال انکودر برای خواندن هر سه سیگنال A، B و Z

۷. دو پایه‌ی PE6 و PE8 به عنوان خروجی دیجیتال

ابتدا بررسی کنید که چگونه می‌توان با انکودر موقعیت را هم دریافت کرد. بازه‌ی زمانی خواندن پالس سرعت را هم ۵ میلی‌ثانیه درنظر بگیرید. سپس سه تابع کمکی زیر را تعریف کنید تا سرعت، زاویه و جهت حرکت را از انکودر دریافت کند:

```
float getVelocity(void); // Returns velocity in RPM
float getAngle(void); // Returns position in degress
uint8_t getDir(void); //Returns direction
```

۲ راه‌اندازی سامانه‌ی کنترل دور و موقعیت

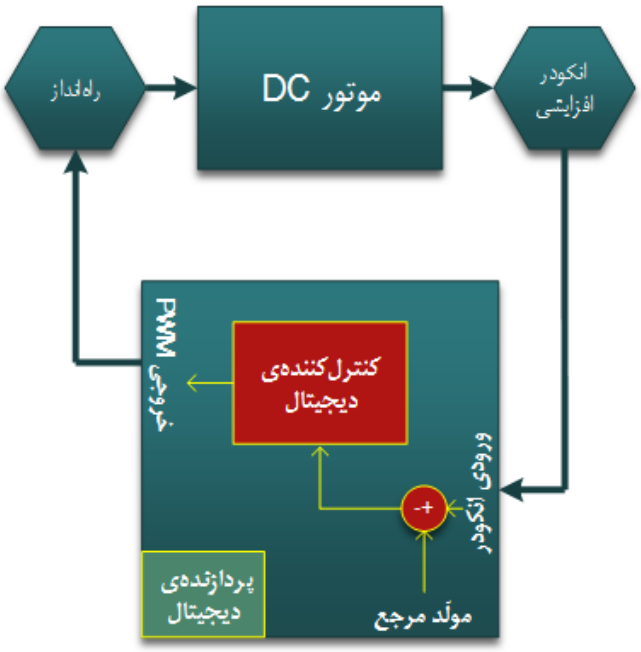
شکل (۱) ساختار نمادین سامانه‌ی کنترل موتور را نمایش می‌دهد. در این سیستم تمام اجزای کنترلی شامل حسگر، فرمان عملگر، تولید مرجع و کنترل‌کننده دیجیتال هستند و هیچ سیگنال آنالوگی در فرآیند کنترل نقش ندارد. برای شروع گام‌های زیر را پی بگیرید:

۱. خروجی‌های انکودر متّصل به موتور را به ورودی‌های مربوطه در برد آزمایشگاه وصل کنید.

۲. خروجی سیگنال PWM و یک خروجی دیجیتال دلخواه را به ترتیب به ورودی‌های PWM و DIR موتور متّصل کنید.

۳. زمین سامانه‌ی کنترل دور و برد آزمایشگاه را یکی کنید.

۴. برای فرمان دادن به موتور و روشن/خاموش کردن LEDها دستورهای زیر را تعریف کنید:



شکل ۱: ساختار کلی سامانه‌ی کنترل دور و موقعیت

```
// Sets the direction or rotation
// dir=0: CW and dir=1 CCW
void setDir(uint8_t dir);
// Sets the duty cycle of PWM signal
// duty = 0 -> 0% duty=1000 -> 100%
void setDuty(uint16_t duty);

// Turns LEDs on and off
// led = 0 -> LED0 (PG4) || led = 1 -> LED1 (PG5)
// status = 0: OFF || status = 1: ON
void turnLED(uint8_t led,uint8_t status);
```

۵. برای پیاده‌سازی هر کنترل‌کننده‌ی دیجیتال به یک زمان‌سنج با فرکانس کاری مشخّص نیاز دارید تا زمان نمونه‌برداری را مشخّص کند. فرض کنید مقدار رجیستر PSC در زمان‌سنج TIM6 برابر با ۸۳۹ تنظیم شده باشد. فرکانس APB متّصل‌شده به زمان‌سنج هم ۸۴ مگاهرتز است. رابطه‌ی بین زمان نمونه‌برداری Ts و مقدار رجیستر ARR را پیدا کنید و تابعی مانند زیر تعریف کنید تا زمان نمونه‌برداری را تنظیم کند:

```
// Sets the Sampling time
// Ts is in milliseconds and whole number
void setTs(uint8_t Ts);
```

۶. با تنظیم گام قبل زمان‌سنجی دارید که هر Ts میلی‌ثانیه یک بار یک وقفه تولید می‌کند. می‌توانید دستورهای مربوط به کنترل‌کننده را در روتین وقفه‌ی این زمان‌سنج پیاده کنید.

علاوه بر این شما به یک ورودی پله‌مانند برای تعیین مرجع چرخش موتور نیاز دارید. مثلاً یک تابع مربّعی که هر یک ۱ ثانیه یک بار مقدار آن از از ۸۰۰ (= ۸۰۰ دور بر دقیقه) به ۱۴۰۰ (=۱۴۰۰ دور بر دقیقه) برود. راهکاری بندیشید

تا به کمک همین زمان‌سنج چنین مرجعی تولید شود.

۷. فرمان کنترلی قابل اعمال به موتور ولتاژی بین ۱۲ و ۱۲- ولت است که باید به صورت PWM به آن اعمال شود. مقدار ولتاژ بر اساس Duty Cycle این سیگنال و علامت هم به کمک سیگنال جهت مشخّص می‌شود. تابعی مانند زیر بنویسید تا چنین کاری را انجام دهد:

```
// Contorl Signal Generation
// Converts u (in Volts) to equivalent PWM duty cycle
// and direction signal. -12<u<12.
// The valus outside of this range must be ignored.
void Vout2PWM(float u);
```

۳ پیاده‌سازی کنترل‌کننده

اکنون همه چیز برای پیاده‌سازی یک کنترل‌کننده‌ی دیجیتال آماده است: پردازنده دور موتور را از انکودر دریافت می‌کند، به کمک زمان‌سنج و در فاصله‌های زمانی مشخّص (Ts=) یک فرمان کنترلی را محاسبه و تولید می‌کند و آن را از طریق سیگنال‌های PWM و جهت به موتور اعمال می‌کند.

برنامه‌ای که می‌نویسید باید ویژگی‌های زیر را داشته باشد:

۱. کد بر اساس یک ماشین حالت کار کند؛

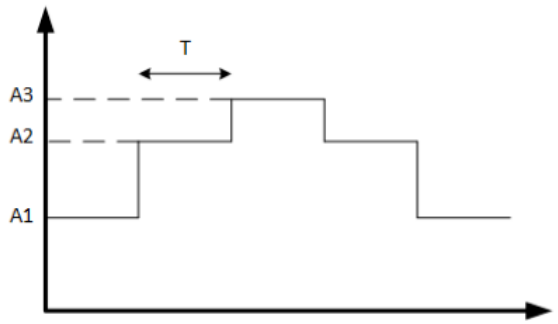
۲. تمام ضرایب به صورت پارامتری تعیین شده باشد؛

۳. زمان نمونه‌برداری (T_s) باید به صورت پارامتری تعیین شده و قابل تغییر باشد؛

۴. در صورت نیاز یک حالت راه‌اندازی تعریف شود تا موتور در یک سرعت اولیه بچرخد؛

۵. برای هر قسمت توضیحات کامل نوشته شود؛

۶. ورودی مرجع دور و موقعیت، به صورت یک پله‌ی بالا ـ پایین‌رونده‌ی سه‌سطحی طراحی شود که سطوح و دوره‌ی آن قابل تنظیم باشد؛ (مانند شکل (۲))



شکل ۲: شکل کلی سیگنال ورودی

۷. فرآیند کنترل باید با فشرده شدن دکمه‌ی PF2 آغاز شود؛

۸. در آغاز اجرای برنامه باید موتور برای مدت کوتاهی چرخانده شود. این کار برای تنظیم زاویه‌ی خروجی انکودر الزامی است.

۴ کنترل دور

کنترل‌کننده‌های P و PIای را که در تمرین «تکمیل مدلسازی و شبیه‌سازی» طراحی کرده‌اید پیاده‌سازی کنید. ورودی مرجع را مطابق شکل ۲ و با پله‌های ۵۰۰ دوربردقیقه‌ای طراحی کنید. زمان نمونه‌برداری را ۱۰ میلی‌ثانیه قرار دهید. نتایج را گزارش کنید. در ادامه در کنترل‌کننده‌ی PI زمان نمونه‌برداری را به ترتیب به ۲۵، ۱۰۰ و ۲۵۰ میلی‌ثانیه افزایش دهید و سیگنال خروجی را ثبت کنید. سپس پاسخ‌ها را از لحاظ پایداری و کارایی مقایسه کنید.

۵ کنترل موقعیت

اکنون بر اساس کد کنترل دور، کنترل‌کننده‌ی موقعیت را هم پیاده‌سازی کنید. سیگنال مرجع را مانند شکل ۲ و برای پله‌های ۳۰ درجه‌ای طراحی کنید. کنترل‌کننده‌های P، PI، PID و Deadbeat را که در تمرین طراحی کرده‌اید، پیاده‌سازی کنید.

۱.۵ رفع اثر غیرخطی ناحیه‌ی مرده

یک ایده‌ی بسیار ساده برای غلبه بر ناحیه‌ی مرده، استفاده از روش Sliding Mode است. این روش نوعی کنترل غیرخطی به شمار می‌آید و در حالت کلی پیچیده و نیازمند ریاضیات پیشرفته است، اما اینجا از ایده‌ی آن استفاده می‌کنیم. می‌دانیم که با ورود سیگنال ورودی موتور به ناحیه‌ی مرده، موتور حرکت نمی‌کند. اکنون سیگنال کنترلی را به نحوی تغییر می‌دهیم که با ورودی موتور به ناحیه‌ی مرده، ولتاژ اعمالی در سطح مشخصی باقی بماند، به عبارت دیگر:

$$u(t) = \begin{cases} U_{MAX} & u(t) \leq V_{db} \\ -U_{MAX} & u(t) \geq -V_{db} \\ u(t) & |u(t)| \geq V_{db} \end{cases}$$

این ویژگی را به کنترل‌کننده‌های موقعیت اضافه و نتیجه را گزارش کنید.

۶ پیوست - کمی بیشتر درباره‌ی کنترل موقعیت بدانیم

هدف کنترل موقعیت این است که موتور بدون خطا و با کمترین فراجهش به یک اندازه‌ی مشخص (مثلاً ۳۰ درجه)بچرخد. در اینجا اندازه‌ی چرخش نسبی است: به این معنی‌که هدف چرخیدن به اندازه‌ی ۳۰ درجه است، نه از یک زاویه‌ی مشخص (مثلاً ۲۰ درجه) تا یک زاویه‌ی مطلق دیگر (مثلاً ۵۰ درجه). کنترل موقعیت به دلایل زیر از کنترل دور پیچیده‌تر است:

ناحیه‌ی مرده‌ی موتور: برای تغییرات کوچک در موقعیت به ویژه هنگام صفر کردن خطا باید ولتاژهای کوچک به موتور اعمال کرد اما معمولاً این ولتاژ در ناحیه ی مرده ی موتور قرار می‌گیرد و عملاً امکان اعمال آن وجود ندارد. مثلاً کنترل کننده بر اساس بازخورد ورودی نتیجه می‌گیرد که باید با اعمال یک ولتاژ ۱ ولتی موتور را به موقعیت نهایی برساند، امّا موتور به ولتاژهای زیر ۲ ولت واکنشی نشان نمی‌دهد. این مساله کنترل‌کننده را در صورتی‌که به درستی طراحی نشده باشد، گیج می‌کند. غلبه بر این عامل غیرخطی در حالت کلی مساله‌ای پیچیده است و در مباحث پیشرفته‌ی کنترلی همچون کنترل سیستم‌های غیرخطی مطرح می‌شود اما در این پروژه سعی می‌کنیم به نحوی بر آن فائق آییم.

اشباع فرمان کنترلی: اشباع یکی از شایع‌ترین عوامل غیرخطی در سیستم های کنترلی با منشائی بسیار ساده و منطقی است: در سیستم سرو موتوری که تغذیه ی ۱۲ ولتی دارد نمی‌توان اختلاف پتانسیلی بزرگتر از ۱۲ ولت ایجاد کرد؛ اما ممکن است کنترل کننده در ابتدای فرآیند کنترلی بخواهد با فرمان های بزرگ (مثلاً ۳۰ ولت) موتور را به حرکت درآورد. این فرمان صادر می‌شود امّا عملاً فرمانی بزرگتر از ۱۲ ولت به موتور اعمال نمی‌شود زیرا به لحاظ فیزیکی چنین چیزی ممکن نیست.

وجود موانع فیزیکی و مکانیکی: لقی و هم تراز نبودن محور موتور باعث حرکت نامتناسب و غیرخطی آن می‌شود. به همین جهت نمی‌توان موتور را با هر دقتی چرخاند. این مساله را می‌توانید با چرخاندن محور موتور با دست به خوبی حس کنید.