توضیحات دوره ویدئویی آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM

در سال‌های اخیر، میکروکنترلرهای STM32 محبوبیت روزافزونی را در میان طراحان سیستم‌های نهفته (Embedded Systems)کسب کرده‌اند و در طیف وسیعی از صنایع (مانند صنعت خودرو و یا اتوماسیون صنعتی)، لوازم خانگی و تجهیزات پزشکی مورد استفاده قرار گرفته‌اند. مهم‌ترین دلایل این امر عبارتند از:

تطبیق‌پذیری و انعطاف‌پذیری: خانواده STM32 در رنج وسیعی از میکروکنترلرها ارایه می‌شوند که هر کدام دارای ویژگی‌ها و امکانات مخصوص به خود هستند که آنها را برای یک کاربرد خاص بهینه می‌کنند. برای کاربردهای مختلف مانند بردهای الکترونیکی کم‌مصرف، تغذیه‌شده با باتری و یا سیستم‌های الکترونیکی با عملکرد بالا، میکروکنترلرهای STM32هم با امکانات ویژه عرضه شده‌اند.

مجموعه پریفرال‌های غنی: در اغلب میکروکنترلرهای STM32 ادوات جانبی (پریفرال‌های) متنوعی مانندDMA ، USART، I2C، SPIو Timer وجود دارند که امکان پیاده‌سازی سیستم‌های نهفته پیچیده را راحت‌تر می‌کنند.

محیط‌ها و ابزارهای برنامه‌نویسی متنوع: برای برنامه‌نویسی این میکروکنترلرها هم محیط‌های نرم‌افزاری متعددی مانند Keilو STM32CubeIDE وجود دارند که هر کدام ابزارهای ویژه‌ای را برای توسعه پروژه و دیباگ‌کردن کدها در اختیار قرار می‌دهند.

انجمن‌های حمایتی: به دلیل استفاده گسترده از این میکروکنترلرها، انجمن‌های متنوع هم در سطح جهانی شکل گرفته‌اند که امکان همکاری بین برنامه‌نویسان مختلف را فراهم می‌کنند. تعداد مستندات، تالارهای گفت‌وگو و منابع آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM رو به روز در حال گسترش است.

قابلیت اطمینان بالا: این میکروکنترلرها می‌توانند در محیط‌های خشن و پر نویز صنعتی عملکرد مورد اطمینان را فراهم کنند.

عملکرد بالا و مقیاس‌پذیری: میکروکنترهای ۳۲-بیتی STM32 بسته به قدرت پردازشی مورد نیاز، توان مصرفی، تعداد پریفرال‌های مورد نیاز و حجم حافظه در سایزهای مختلفی عرضه می‌شوند. برخی از خانواده‌های این میکروکنترلرها می‌توانند پردازش‌های سنگین ریاضی را با کمترین توان مصرفی انجام دهند.

علاوه بر این، طبق داده‌های گزارش‌شده توسط Grand View Research، سهم بازار میکروکنترلرهای STM32 در سال 2024 حدود 31.45 میلیارد دلار پیش‌بینی شده و این مقدار با نرخ سالانه 11.7 درصد افزایش پیدا خواهد کرد. از این رو، یادگیری عمیق و اصولی برنامه‌نویسی میکروکنترلرهای STM32 بر همه مهندسین برق و الکترونیک و علاقه‌مندان ضروری است. همچنین، داشتن مهارت حرفه‌ای در برنامه‌نویسی میکروکنترلرها، علاوه بر افزایش اعتماد به نفس، منجر به افزایش فرصت‌های شغلی با درآمد بیشتر خواهد شد.

روش‌های برنامه‌نویسی میکروکنترلرهای STM32 ARM

سه روش اصلی برای برنامه‌نویسی میکروکنترلرهای STM32 عبارتند از:

- Bare Metal
- Low Layer – LL
- Hardware Abstraction Layer – HAL

روش HAL آسان‌ترین راه برای برنامه‌نویسی میکروکنترلرهای STM32 است. در این روش، مجموعه‌ای از دستورات سطح‌بالا وجود دارند که برنامه‌نویس بسته به نیاز از آنها برای راه‌اندازی میکروکنترلر استفاده می‌کند. در این روش، نیازی به اطلاعات عمیق از ساختار میکروکنترلر و رجیسترهای آن نیست. و در حقیقت، برنامه‌نویس هیچ ارتباطی با رجیسترها برقرار نخواهد کرد. این زبان بیشتر برای مبتدیان مناسب است و برای انجام پروژه‌های تجاری بهینه نیستند، زیرا حجم زیادی از حافظه میکرو توسط توابع HAL اشغال خواهد شد. در مقابل، سریع‌ترین ولی سخت‌ترین روش برای برنامه‌نویسی میکروکنترلرهای STM32 روش اول یعنی Bare Metal است. در این روش، برنامه‌نویسی تنها از طریق

آدرس رجیسترهای پریفرال‌ها (که داخل دیتاشیت و فایل راهنمای میکروکنترلر موجود هست) انجام می‌شود. این روش نیازمند دانش عمیق از الکترونیک و میکروکنترلر است ولی منجر به برنامه‌های بهینه با عملکرد بالا (حجم کمتر همراه با سرعت بیشتر) خواهد شد. در میان این دو روش، روش برنامه‌نویسی LL هم وجود دارد که باز هم همانند روش HAL مجموعه‌ای از توابع برای تنظیم رجیسترهای میکرو در اختیار برنامه‌نویس قرار می‌دهد، ولی بر خلاف HAL فاصله زیادی از سطح رجیستر ندارند. به همین دلیل، سرعت پردازش میکرو نسبت به روش Bare Metal افت محسوسی پیدا نمی‌کند. در حقیقت، در روش LL برای تنظیم رجیسترهای میکرو از توابع با اسامی بامعنا و مشخص استفاده می‌شود تا خوانایی برنامه بالاتر رود و همچنین سرعت توسعه پروژه هم زیادتر شود. برای انجام پروژه‌های صنعتی ترکیبی از برنامه‌نویسی Bare Metal و LL پیشنهاد می‌شود که در این آموزش این رویه دنبال خواهد شد. باید توجه کرد که یادگیری عمیق برنامه‌نویسی میکروکنترلرهای STM32 ممکن است در ابتدا گیج‌کننده و خسته‌کننده به نظر برسد. اما با یادگیری اصولی چگونگی استفاده از فایل‌های دیتاشیت و راهنمای میکروکنترلرها و نرم‌افزارهای مربوطه این کار خیلی لذت‌بخش می‌شود و تجربه عملی ارزشمندی را هم به ارمغان خواهد آورد.

30ساعت آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM

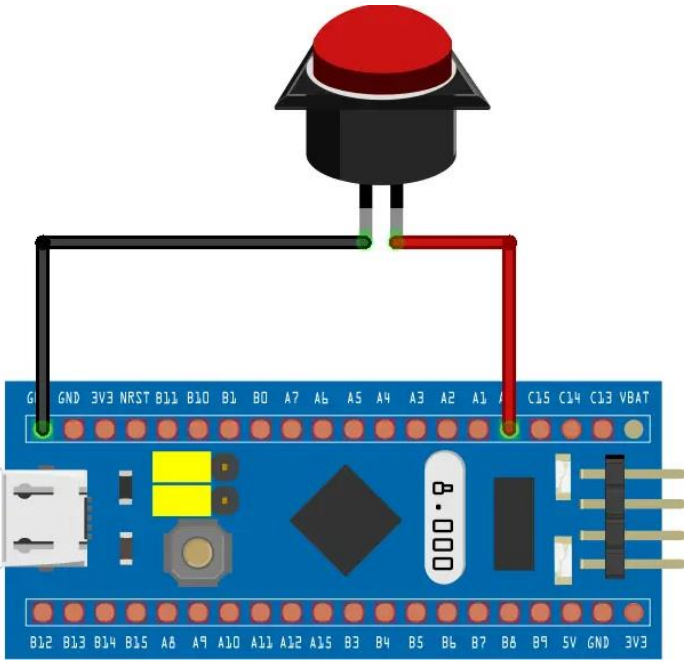
در آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM ، در ۱۰ فصل مختلف برنامه‌نویسی بخش‌های مختلف میکروکنترلرهای STM32 آموزش داده خواهد شد. توجه کنید که روش برنامه‌نویسی ترکیبی از رجیستری و LL است و به صورت هم‌زمان هر دو روش آموزش داده خواهد شد. همچنین محیط برنامه‌نویسی STM32CubeIDE است ولی به راحتی و تنها با کپی کردن کدها می‌توان آن را به نرم‌افزار Keil هم منتقل کرد. مزایای ویژه‌ی این دوره‌ی آموزشی عبارتند از:

- قبل از برنامه‌نویسی هر پریفرالی، اصول عملکرد آن پریفرال بر طبق دیتاشیت و فایل راهنمای میکروکنترلر توضیح داده خواهد شد.
- برنامه‌نویسی ترکیبی از رجیستر و LL است تا سرعت اجرای برنامه و حجم مصرفی آن بهینه باشد.
- بر اساس تجربیات چندساله‌ی مدرس در توسعه پروژه‌های صنعتی مختلف، سعی شده است که با بهترین روش انتقال مطالب انجام شود.
- سیگنال‌های پریفرال‌های مختلف (مانند پروتکل‌های UART وSPI) توسط لاجیک آنالایزر به دقت بررسی خواهند شد تا درک عمیق‌تری از آنها حاصل شود.

- انجام پروژه‌های عملی مهم برای یادگیری بهتر: برای مثال بعد از یادگیری GPIO یک کتابخانه برای پیاده‌سازی پروتکل OneWire انجام می‌شود. و یا بعد از یادگیری ADC و PWM کنترل ولتاژ یک مبدل سوییچینگ بوست انجام خواهد شد. همچنین برای پروتکل‌های I2C و SPI هم کتابخانه‌های رجیستری کاملی برای دریافت اطلاعات از سنسورهای MPU6050 و ADXL345 انجام می‌شود. در نهایت، بعد از یادگیری UART ، پروتکل صنعتی مُدباس هم برای ارتباط بین میکروکنترلر STM32 و HMI Panel Master پیاده‌سازی خواهد شد.

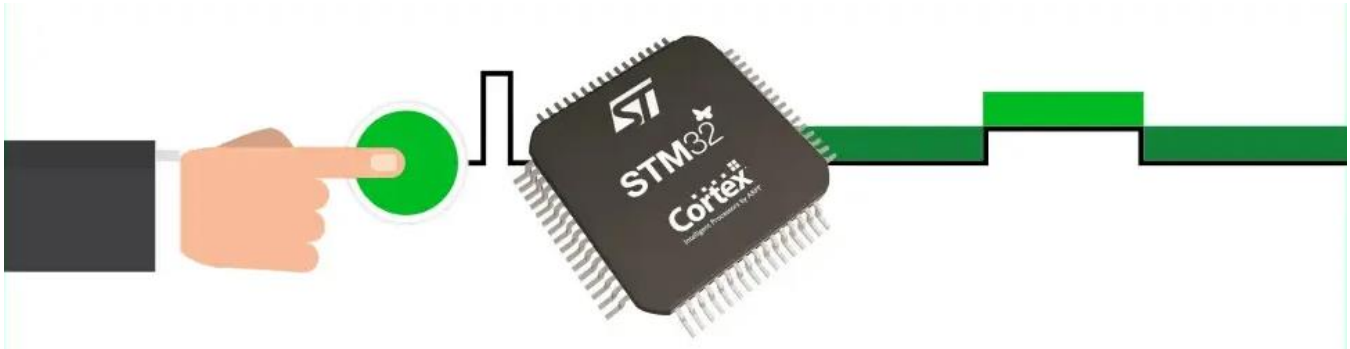
فصل اول :پین‌های ورودی-خروجی چندمنظورهGPIO

در این فصل از آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM نحوه کنترل پین‌های دیجیتال و چندمنظوره میکروکنترلر آموزش داده می‌شود. پین‌های GPIO برای کاربردهای مختلفی مانند تولید خروجی‌های دیجیتال، دریافت ورودی‌های دیجیتال، سیگنال‌های ورودی و خروجی پروتکل‌های ارتباطی مانند UART و I2C و همچنین دریافت داده‌های آنالوگ و تولید داده‌های آنالوگ به کار می‌روند. در اولین فصل از آموزش با نحوه تولید خروجی‌های دیجیتال کنترل‌شده و سپس دریافت ورودی‌های دیجیتال از دنیای خارج میکروکنترلر آشنا خواهیم شد. در این فصل و البته همه فصل‌های دیگر آموزش، در ابتدا مروری بر مشخصه‌های GPIO طبق فایل‌های دیتاشیت و راهنمای میکروکنترلر خواهیم داشت و سپس به کدنویسی و پیاده‌سازی عملی آنها خواهیم پرداخت. در بخش اول بدون هیچ گونه درایوری و تنها بر اساس اطلاعات درج‌شده در فایل‌های دیتاشیت و راهنمای میکروکنترلر (آدرس‌های خام رجیسترها) راه‌اندازی GPIO انجام خواهد شد. در طول برنامه‌نویسی همواره به صورت تعاملی از فایل‌های راهنمای میکروکنترلر هم استفاده می‌شود تا درک عمیق‌تری از رجیسترهای میکروکنترلر هم حاصل شود. در ادامه این فصل، راه‌اندازی GPIO توسط درایور CMSISانجام خواهد شد تا دیگر نیازی به تعریف آدرس‌های پایه‌ای رجیسترها وجود نداشته باشد. سپس برنامه‌نویسی GPIOتوسط درایورهای LL یا Low Layer هم به صورت کامل تشریح می‌شود تا خوانایی برنامه بالاتر برود. در انتهای این فصل هم جهت یادگیری بهتر و عمیق‌تر GPIO در میکروکنترلرهای STM32 ، کدنویسی اندازه‌گیری دما توسط سنسور DS18B20 با پروتکل تک‌سیمه یا One Wire از صفر انجام خواهد شد.



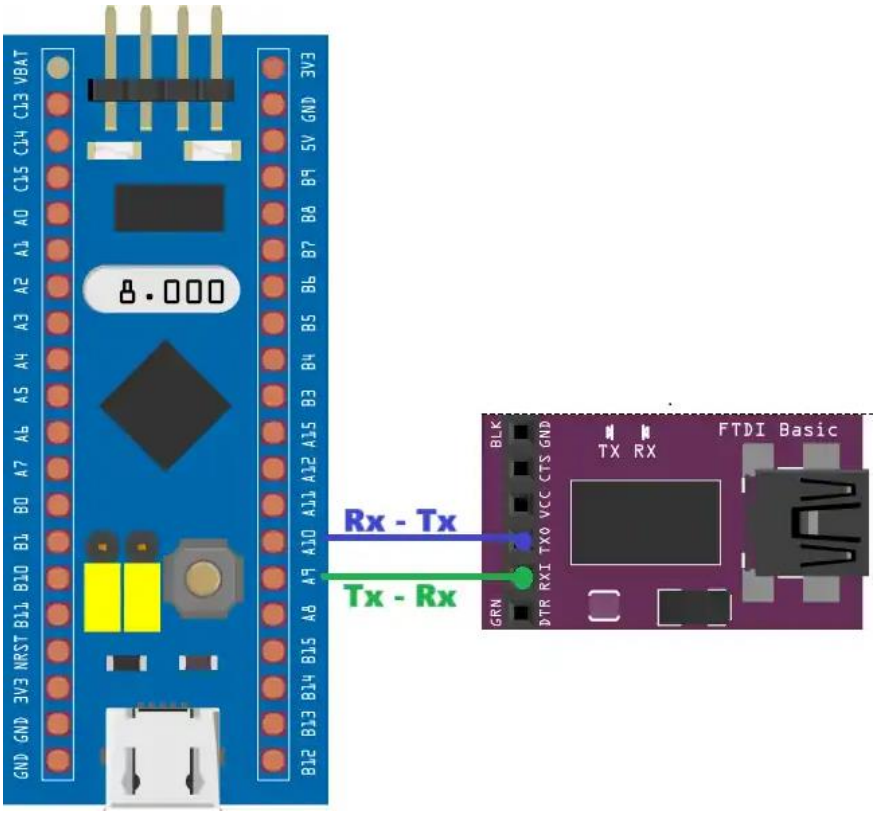
فصل دوم: وقفه خارجی EXTI

در فصل اول آموزش نحوه خواندن ورودی‌های دیجیتال توسط میکروکنترلر STM32 تشریح شد. روش پیاده‌سازی بر مبنای polling است. به این معنی که CPU باید منتظر بماند تا ورودی تغییر وضعیت دهد. اما در عمل، باید فقط در زمانی که تغییر وضعیتی در سیگنال‌های دیجیتال ورودی اتفاق می‌افتد CPU هم درگیر شود. برای این کار از وقفه استفاده می‌شود. منابع وقفه در میکروکنترلرهای SMT32 خیلی زیاد و متنوع هستند. در فصل دوم آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM، وقفه‌های خارجی یا EXTI آموزش داده می‌شود. یاد خواهیم گرفت که چگونه یک پین GPIO میکروکنترلر را طوری تنظیم کنیم که فقط زمانی که یک لبه‌ی پایین‌رونده و یا بالارونده (و یا در هر دو حالت) در سیگنال دیجیتال ورودی اتفاق افتاد، یک تابع معینی اجرا شود.



فصل سوم: ارتباط سریال USART

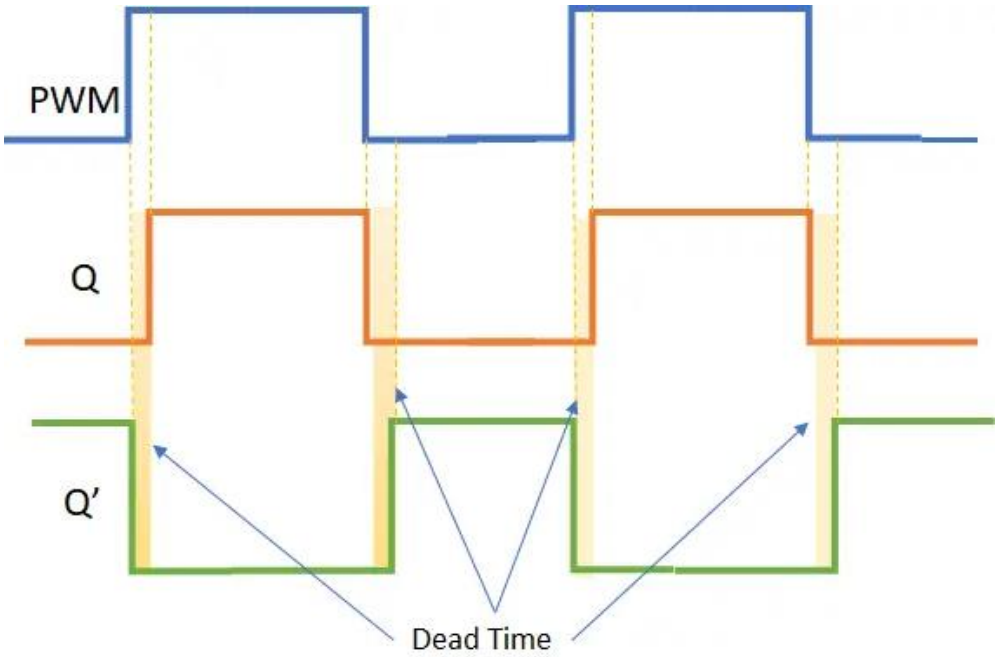
پروتکل‌های ارتباطی جز جدایی‌ناپذیر همه مدارات الکترونیکی هستند. با استفاده از پروتکل‌ها اطلاعات دیجیتال از یک میکروکنترلر به یک میکروکنترلر دیگر و یا از یکی آیسی مدار مجتمع به یک میکروکنترلر منتقل می‌شود. در میکروکنترلرهای SMT32 پروتکل‌های ارتباطی متنوعی مانند USART، I2C و SPI وجود دارند که در این فصل به پروتکل سریال آسنکرون یا UART پرداخته می‌شود و بقیه پروتکل‌ها هم در فصول مجزا پیاده‌سازی می‌شوند. در پروتکل USART اطلاعات به صورت سریال (پشت سر هم و بیت به بیت) و در یک قالب مشخصی شامل بیت استارت، بیت‌های داده، بیت پریتی و بیت‌های توقف منتقل می‌شوند. اصول کار این پروتکل و پیاده‌سازی آن توسط درایور LL در این فصل از آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM بررسی خواهد شد.



فصل چهارم: تایمرها و شمارنده‌ها

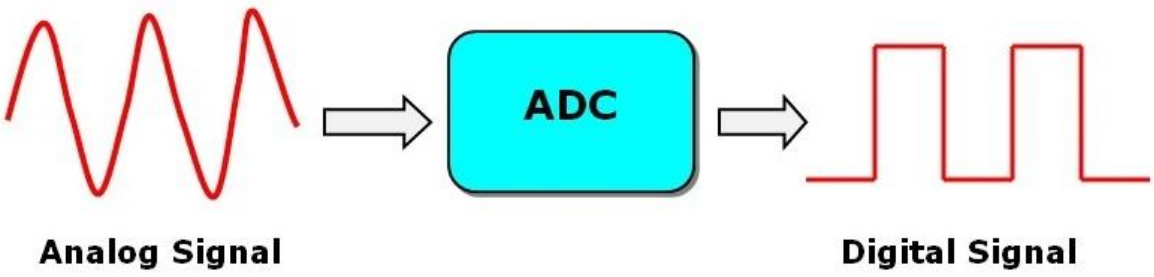
تایمرها و شمارنده‌ها (به اختصار همان تایمرها) هم جز اساسی همه میکروکنترلرها هستند. با استفاده از این ابزار می‌توان زمان‌بندی‌های دقیقی را برای انجام کارهای مختلف ایجاد نمود. و یا می‌توان شکل‌موج‌های دلخواه مانند مدولاسیون عرض پالس PWM را برای راه‌اندازی موتورها، سروموتورها و مبدل‌های سوییچینگ ساخت. همچنین می‌توان فرکانس

سیگنال‌های دیجیتال ورودی را اندازه‌گیری کرد و خیلی از کاربردهای دیگر. راه‌اندازی همه این ابزارها به صورت گام به گام و تشریحی در فصل چهارم آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM انجام می‌شود.



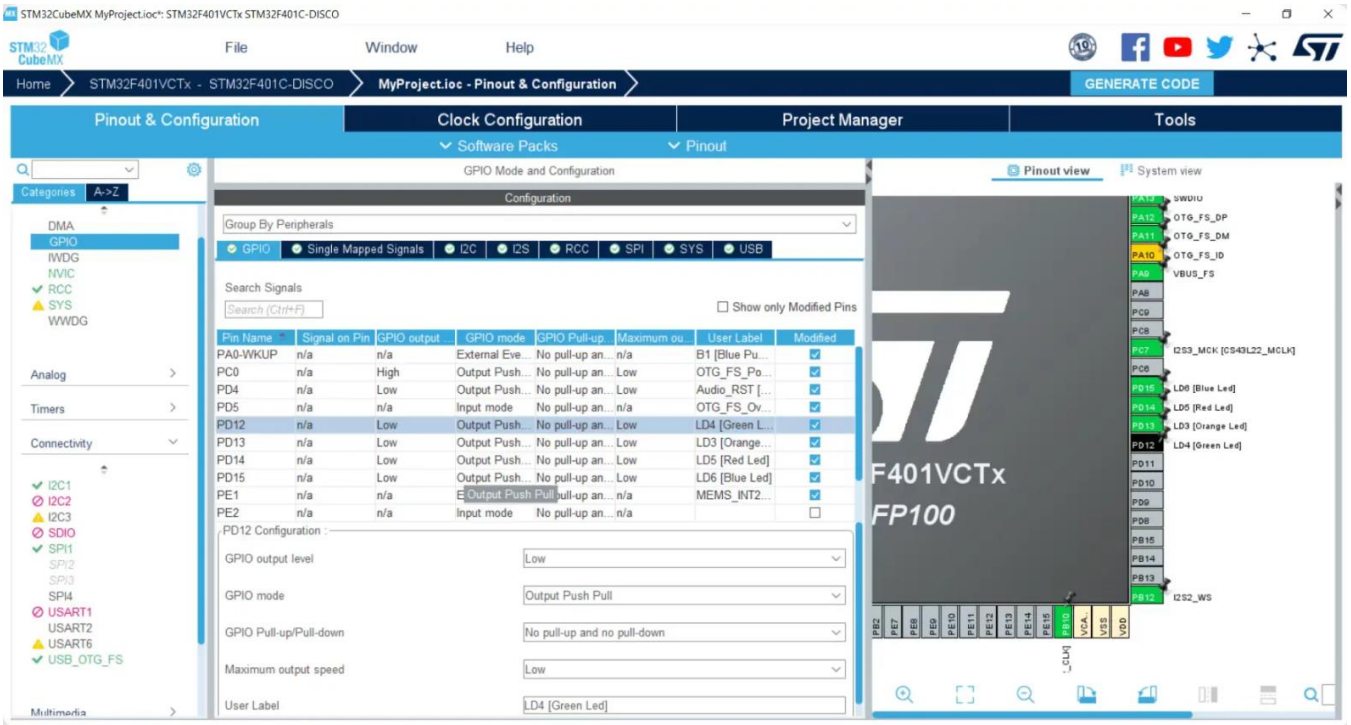
فصل پنجم: مبدل‌های آنالوگ به دیجیتال (ADC) و دیجیتال به آنالوگ(DAC)

با استفاده از مبدل‌های آنالوگ به دیجیتال ADC می‌توان داده‌های آنالوگ خارج از میکروکنترلر را داده‌برداری و برای پردازش‌های ثانویه وارد میکروکنترلر کرد. علاوه بر این، با استفاده از مبدل‌های دیجیتال به آنالوگ DAC هم می‌توان داده پردازش‌شده را به فرم آنالوگ به خارج از میکروکنترلر فرستاد. در این فصل کدنویسی پریفرال‌های ADC و DAC در میکروکنترلرهای STM32 را یاد خواهیم گرفت.



فصل ششم: راه‌اندازی پریفرال‌ها باSTM32CubeMX

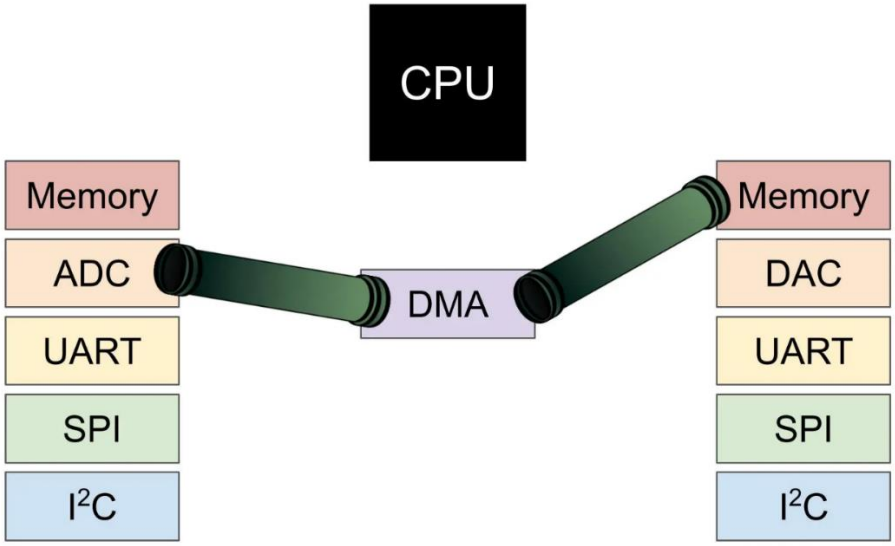
در فصل‌های قبل راه‌اندازی پریفرال‌ها بدون هیچ ابزار کمکی دیگری فقط توسط رجیسترهای CMSIS و درایور LL انجام شد. در حقیقت، باید خودمان با توجه به فایل‌های راهنمای میکروکنترلر، تمام تنظیمات را برای راه‌اندازی یک پریفرال خاص انجام دهیم. اما با استفاده از ابزار گرافیکی STM32CubeMX می‌توان این قدم را به صورت گرافیکی و خیلی سریع‌تر انجام داد. برای مثال، برای راه‌اندازی پروتکل USART می‌توان تنظیماتی که فقط یک بار باید انجام شوند (مانند تنظیم نرخ داده و یا نوع پریتی) را با STM32CubeMX انجام داد و بقیه کارها که باید پیوسته انجام شوند (مانند ارسال و دریافت داده) را با همان رجیسترهای CMSIS و یا توابع LL انجام داد. در فصل ششم آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM نحوه استفاده از STM32CubeMX برای تنظیمات اولیه پریفرال‌ها را یاد خواهیم گرفت.



فصل هفتم: دسترسی مستقیم حافظهDMA

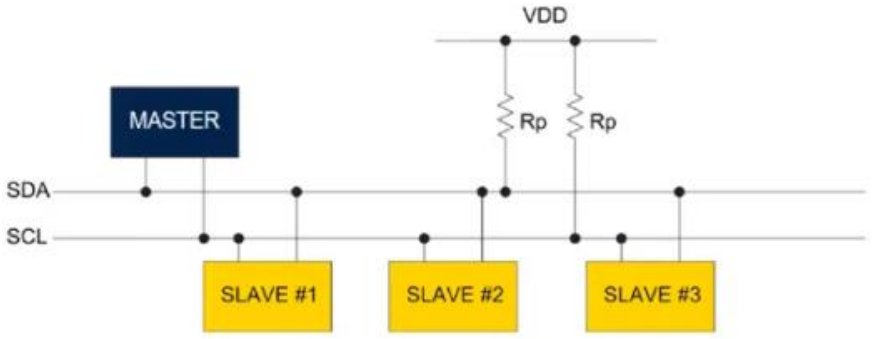
یکی از امکانات خیلی مهم در میکروکنترلرهای STM32 امکان دسترسی مستقیم به حافظه بدون دخالت میکروکنترلر است. در مواردی که لازم است اطلاعات حجیم از یک نقطه به نقطه دیگر منتقل شود، استفاده از خود CPU زمان زیادی را هدر می‌دهد. برای رفع این مشکل ابزار DMA یا Direct Memory Access در میکروکنترلرهای STM32 فراهم

شده است. با این ابزار می‌توان تا حدود 65535 بایت را بدون هیچ دخالتی از سمت CPU منتقل کرد. نحوه انجام این کار در فصل هفتم آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM با پروژه‌های متنوع تشریح خواهد شد.



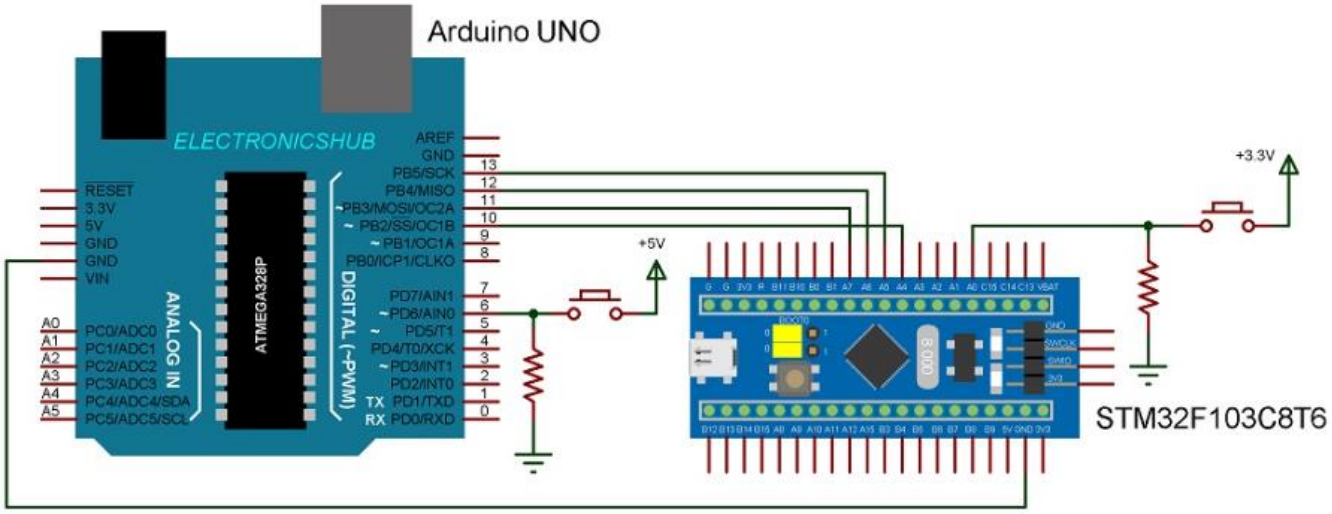
فصل هشتم: پروتکل دوسیمهI2C

یکی دیگر از پروتکل‌های ارتباطی مهم در میکروکنترلرهای STM32 پروتکل دوسیمه یا I2C است. ویژگی مهم این پروتکل این است که اطلاعات چندین گره داده (چندین سنسور) را می‌توان تنها با یک میکروکنترلر اندازه‌گیری کرد. زیرا، در فریم داده این پروتکل علاوه بر داده‌های اصلی، یک آدرس منحصر به فرد که سنسور مورد نظر را مشخص می‌کند هم ارسال می‌شود. در فصل هشتم، ابتدا اصول کار این پروتکل بیان خواهد شد. سپس جهت یادگیری عمیق‌تر، پیاده‌سازی پروتکل تنها با استفاده از رجیسترهای CMSIS انجام خواهد شد و یک کتابخانه I2C طراحی می‌شود. در انتها، کتابخانه طراحی‌شده برای خواندن اطلاعات از یک سنسور شتاب‌سنج ژیروسیکوپ شش محوره MPU6050 مورد استفاده قرار می‌گیرد.



فصل نهم: پروتکل SPI

پروتکل SPI یکی از سریع‌ترین ارتباطات ممکن را مدارات الکترونیکی فراهم می‌کند. زیرا هم فرکانس کلاک آن بالاتر است و هم انتقال اطلاعات به صورت دوطرفه و هم‌زمان انجام می‌شود. در فصل نهم آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM، ابتدا اصول کار این پروتکل آموزش داده می‌شود. سپس تنها با استفاده از رجیسترهای CMSIS به طراحی یک کتابخانه SPI خواهیم پرداخت. با استفاده از کتابخانه طراحی‌شده هم ارتباط بین یک میکروکنترلر STM32 و یک میکروکنترلر AVR انجام خواهد شد. علاوه بر این، اطلاعات شتاب‌سنج ADXL345 هم توسط پروتکل SPI قرائت خواهد شد. همچنین، نرم‌افزار STM32CubeMonitor برای نمایش گرافیکی تغییرات زمانی متغیرهای جهانی داخل برنامه معرفی خواهد شد.



فصل دهم: مطالب تکمیلی

در نهایت، در فصل آخر آموزش برنامه‌نویسی میکروکنترلرهای STM32 ARM به چند مورد تکمیلی خواهیم پرداخت. در ابتدا تایمرهای زمان حقیقی یا RTC آموزش داده می‌شود. با این تایمرها می‌توان زمان و تاریخ واقعی را با دقت بالایی همواره ثبت کرد و حتی ریست‌کردن میکروکنترلر باید ریست‌شدن تایمر نمی‌شود. در قدم بعدی تایمرهای نگهبان یا Watchdog Timersیاد داده خواهند شد. با استفاده از این تایمرها می‌توان در صورتی که اجرای بخشی از برنامه‌ی داخل میکروکنترلر بیش از حد طولانی شد و یا زودتر از حد لازم طول کشید، میکروکنترلر را ریست نمود. در نهایت، در بخش آخر این فصل هم نحوه تولید کد C توسط نرم‌افزار MATLAB برای انجام عملیات‌های ریاضی پیچیده توضیح داده خواهد شد. برای مثال، از تبدیل فوریه سریع FFT برای محاسبه طیف فرکانسی یک سیگنال دیجیتال استفاده می‌شود.

این الگوریتم به راحتی توسط دستور fft در متلب پیاده‌سازی می‌شود. حال در این فصل یاد خواهیم گرفت که چگونه دستورات متلب را به کتابخانه‌های مستقل به زبان C تبدیل کرد و از آنها در برنامه‌نویسی میکروکنترلرهای STM32 استفاده کرد.

معرفی مدرس

این دوره توسط دکتر امید زندی دکتری تخصصی رشته مهندسی برق گرایش کنترل از دانشگاه علم و صنعت تدریس شده است.

ایشان مقاطع تحصیلی کارشناسی و کارشناسی ارشد خود را نیز در دانشگاه علم و صنعت ایران گذرانده و به ترتیب با معدل‌های ۱۸.۵۲ و ۱۹.۷۶ فارغ‌التحصیل و رتبه اول را کسب نمودند. علاوه بر این، ایشان موفق به کسب رتبه اول (مدال طلا) المپیاد علمی دانشجویی مهندسی برق کشور در سال ۱۳۹۴ و همچنین رتبه سوم (مدال برنز) المپیاد علمی دانشجویی کشوری در سال ۱۳۹۳ در رشته برق شده‌اند.

دکتر امید زندی سابقه تدریس بیش از 1000 ساعت دروس مختلف مهندسی برق، الکترونیک، کنترل و ریاضیات را در وبسایت فرادرس دارند و تاکنون بیش از ۲۹۰ هزار نفر از آموزش‌های ایشان بهره برده‌اند.

دوره های ویدئویی جدید ایشان که از طریق وبسایت الکترو ولت منتشر می شود، متناسب با نیازهای روز مهندسین برق و الکترونیک می باشد که با هدف پیشرفت مخاطبان این حوزه مورد تایید می باشد.

نحوه خرید و هزینه آموزش برنامه‌نویسی میکروکنترلرهایSTM32 ARM

هزینه آموزش: ۴ میلیون تومان

(تخفیف ۵۰٪) >> فقط ۲ میلیون تومان – به مدت محدود

لینک مشاهده ویدئوهای رایگان و خرید دوره از سایت الکترو ولت