فهرست مطالب

فصل ۱

مقدمه

اولین سؤالی که باید پاسخ داده شود، این است که یک مایکروکنترلر دقیقاً چی هست؟ به صورت مشخص این تنها یک قطعه سیلیکونی است اما چه چیزی درون آن قرار دارد؟

مایکروکنترلر چیست؟ تصویر بزرگ

ارزش دارد قبل از رفتن به سراغ جزئیاتی مانند بیت‌ها، فلش کردن حافظه برنامه و موارد دیگر ابتدا نگاهی به تصویر بزرگ بیندازیم.

یک کامپیوتر روی یک چیپ

مایکروکنترلرها اغلب به عنوان یک ماهیت کامپیوتری کامل روی یک تک چیپ تعریف می‌شوند و این کاملاً درست است.

در هسته، مایکروکنترلرها دارای یک پردازشگر مشابه با CPU کامپیوتر شما هستند. این پردازشگر دستورالعمل‌ها را از یک فضای حافظه (به جای هارددیسک درون یک حافظه فلش) دریافت کرده، ریاضیات به یک واحد منطق ریاضی ارسال شده (به جای یک پردازشگر ریاضی) و در زمان اجرا متغیرها درون RAM ذخیره می‌شوند.

بسیاری از چیپ‌ها دارای سخت‌افزار سریالی خاصی هستند که به آن‌ها اجازه برقراری ارتباط با دنیای بیرون را می‌دهد. برای مثال، در فصل ۵ می‌توانید داده‌ها را از کامپیوتر رومیزی خود ارسال و دریافت کنید. درست است که این یک اترنت (Ethernet) گیگابایتی نیست اما مایکروکنترلر شما دیگر مجبور نیست به تنهایی زندگی کند.

همانند هر کامپیوتری، می‌توانید از انواع مختلفی از زبان‌های برنامه‌نویسی برای مایکروکنترلر خود استفاده کنید. ما از C استفاده می‌کنیم و اگر به نرم‌افزار علاقه داشته باشید، مثال‌های کدی که در این کتاب می‌بینید به آسانی قابل درک هستند. این حاوی چیزهایی مانند حلقه‌های for و نسبت دادن به متغیرها است. اگر به چرخه طراحی-کد-کامپایل-اجرا-دیباگ عادت داشته باشید یا IDE خود را دارید، با قسمت نرم‌افزاری مایکروکنترلرها کاملاً راحت خواهید بود.

پس مایکروکنترلرها در حقیقت کامپیوترهای بسیار کوچکی روی یک چیپ هستند.

... اما یک کامپیوتر بسیار کوچک

اما مایکروکنترلرهای AVR کامپیوترهای بسیار کوچکی روی یک چیپ هستند و اندازه کوچک آن‌ها باعث شده توسعه و برنامه‌نویسی برای مایکروکنترلرها از کامپیوترهای معمولی متفاوت باشد.

یک مورد مهم که باید به آن دقت کرد این است که در خط تولید AVR، از ATtiny15 تا ATmega328، مقدار فضای حافظه فلش درون نام مشخص شده است. بله، درست خواندید ما درباره ۱ کیلوبایت تا ۳۲ کیلوبایت فضا برای کد شما صحبت می‌کنیم. به دلیل این فضای حافظه برنامه محدود شده، قلمرو برنامه

شما برای اجرا روی یک چیپ به مراتب کمتر از برنامه جاوایی است که برای یک سیستم بانکداری می‌نویسید.

مایکروکنترلرها دارای حافظه RAM محدود نیز هستند. چیپ‌های ATmega168 دارای یک حافظه ۱ کیلوبایتی هستند. با این که می‌توان حافظه‌های RAM خارجی را با رابط‌هایی وصل کرد تا این محدودیت از بین برود، اما اغلب موارد این حافظه کاری محدود تنها چیزی است که باید از آن استفاده کنید. اما ۱۰۲۴ بایت در اکثر مواقع محدود کننده نیست. کاربرد معمول مایکروکنترلرها یک استریم داده ورودی دریافت کرده، آن را نسبتاً سریع پردازش کرده و به محض ممکن آن را با یک بافر نسبتاً کوچک به خروجی می‌دهد.

و اکنون که در حال صحبت درباره مشخصات فنی هستیم، سرعت ساعت هسته CPU مایکروپروسسورهای AVR حدود ۱ تا ۲۰ مگاهرتز است (وقتی با یک کریستالی اکسترنال استفاده می‌شود) که خیلی با گیگاهرتزهایی که عادت دارید متفاوت هستند. حتی با طراحی RISC برای AVR که به یک دستورالعمل در هر چرخه نزدیک می‌شود، سرعت پردازش خالص یک مایکروکنترلر نمی‌تواند به یک PC مدرن نزدیک شود. (اما وقتی بدانید با چند *میلیون* عملیات در ثانیه چه کارهایی می‌توان انجام داد متعجب خواهید شد.)

در آخر خانواده AVR مایکروکنترلرها دارای پردازنده‌های ۸ بیتی بدون پردازشگر ریاضی نقطه اعشار هستند. این بدان معنی است که اغلب محاسبات و ریاضی که انجام می‌دهید شامل اعداد ۸ یا ۱۶ بیت هستند. می‌توانید از اعداد صحیح ۳۲ بیت استفاده کنید اما دقت بیشتر باعث کاهش سرعت می‌شود. یک کتابخانه ریاضی نقطه اعشار برای AVRها وجود دارد اما مقدار زیادی از حافظه برنامه را مصرف می‌کند و بنابراین اغلب باید نرم‌افزار خود را برای استفاده هوشمندانه از اعداد صحیح بازنویسی کنید. (البته اگر حافظه آزادی داشتید می‌توانید از این کتابخانه استفاده کرده تا زندگی شما آسان‌تر شود.)

به دلیل این که کامپیوتر درون مایکروکنترلر واقعاً در ابعاد مایکرو است، بعضی از امکاناتی که روی PC خود به آن‌ها عادت داشتید در این جا وجود ندارند. برای مثال هیچ ویدئو، صدا، صفحه کلید، ماوس یا هارددیسکی پیدا نمی‌کنید. هیچ سیستم‌عامل وجود ندارد یعنی هیچ قابلیت داخلی برای چندوظیفه‌ای وجود ندارد. در بخش ۲ به شما نشان می‌دهم چگونه از وقفه‌های سخت‌افزاری و تایمر برای کمک به حل این مشکل استفاده کنید.

اما مایکروکنترلرها دارای بازه‌ای از تجهیزات سخت‌افزاری داخلی هستند که بسیاری از کارهای معمول را آسان‌تر می‌کنند. برای مثال اینترفیس سریال سخت‌افزاری داخلی به این معنی است که مجبور نیستید درایورهای سریال بنویسید بلکه باید بایت‌های خود را در مکان صحیح نوشته و منتظر انتقال آن‌ها شوید. سخت‌افزار ماژوله کردن پهنای پالس داخلی به شما اجازه داده تا یک بایت در حافظه را نوشته و سپس AVR یک خروجی ولتاژ را با توجه به دقت مایکروثانیه کسری فعال می‌کند.

مایکروکنترلرها چه کاری می‌توانند انجام دهند؟

مثال‌های کاربردی مایکروکنترلرها به مراتب بیش از مغز پشت مایکروفر شما هستند که فشردن دکمه‌ها را لمس کرده، ورودی را به یک سری مقادیر تبدیل کرده و همه آن‌ها را روی صفحه نشان داده تا بتوانید بخوانید. مایکروکنترلر درون ریموت کنترل عمومی شما کلیدهای فشرده شده را به یک سری دقیق از پالس برای یک LED مادون قرمز تبدیل کرده که به مایکروکنترلر درون تلویزیون شما می‌گوید کانال را عوض کرده یا صدا را افزایش دهد.

اما این تنها کاربردهای ساده نیست که از مایکروکنترلر در آن‌ها استفاده می‌شود. از آن‌ها برای کد ترمز و شتاب دادن در خودروهای نروژ استفاده شده و از آن‌ها به عنوان بخشی از مغز ماهواره‌ها نیز استفاده می‌شود.

پروژه‌های هکر که از مایکروکنترلرها استفاده می‌کنند شامل هر چیز جالبی می‌شوند از کنترل‌های موتور RepRap و الکترونیک‌های برنامه‌ریزی و واحدهای مدیریت اینرسی کوادکوپترها تا ثبت‌کننده‌های داده بالون‌های ارتفاع بالا؛ روبات‌های اندازه کوچک؛ کنترل‌های کابینت‌های MAME و شبیه‌سازهای درایو دیسک برای C64. اگر این کتاب را می‌خوانید احتمالاً هم اکنون تعدادی کاربرد در ذهن خود دارید.

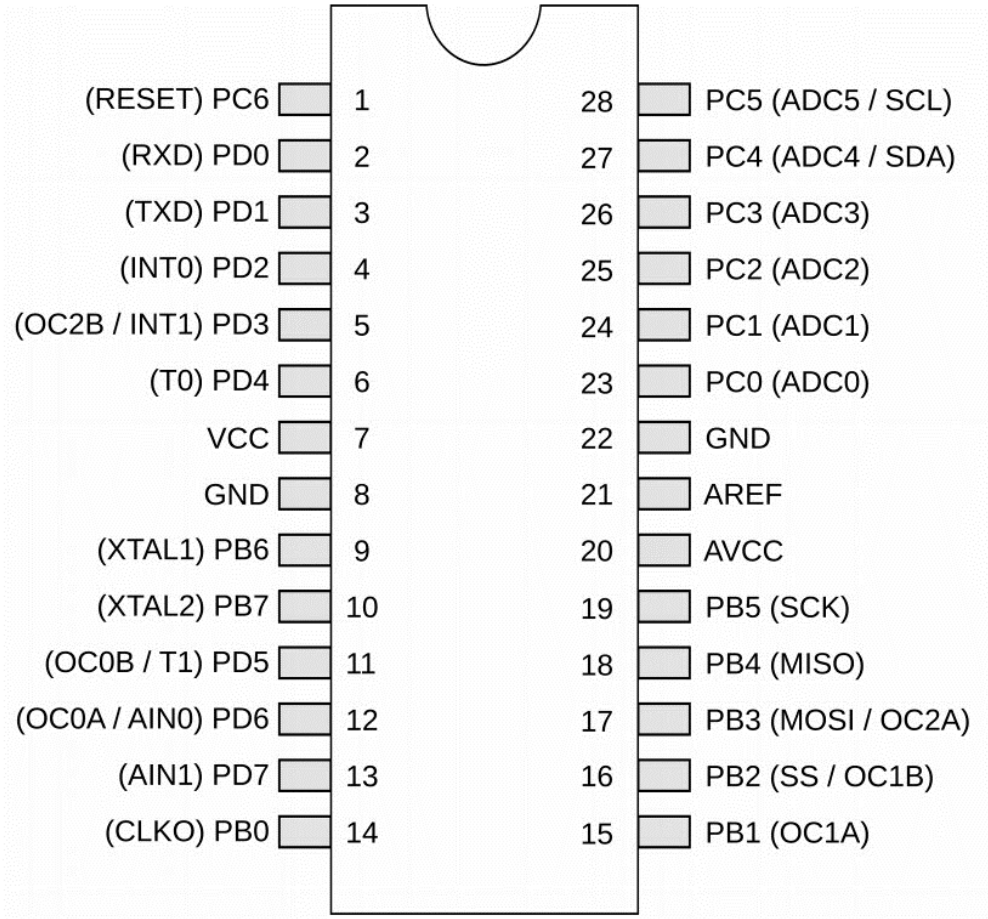
سخت‌افزار: تصویر بزرگ

بنابراین یک مایکروکنترلر یک کامپیوتر بسیار محدود است- در حقیقت چیزی بین یک کامپیوتر و یک کامپوننت (Componenet). درباره قسمت کامپیوتر صحبت زیادی می‌کنم. اما درباره چیپ‌های AVR به عنوان کامپوننت چطور؟ چگونه آن‌ها را وصل می‌کنید؟ و چگونه از آن‌ها استفاده می‌نمایید؟ شکل ۱-۱ تمام پین‌های چیپ را نشان می‌دهد.

اگر هیچ پس‌زمینه‌ای ندارید، احتمالاً تعجب می‌کنید که یک مایکروکنترلر همه این کارها را انجام می‌دهد. پاسخ کوتاه خواندن ولتاژهای اعمال شده بین پین‌های مختلف آن یا با تعیین ولتاژ خروجی به پین‌های مشابه است. یک LED چشمک‌زن مثال خوبی است- وقتی ولتاژ خروجی بالا باشد، LED روشن می‌شود و وقتی ولتاژ پایین باشد LED خاموش می‌شود. مثال‌های پیچیده‌تر شامل پورت‌های سریالی هستند که با انکد کردن اعداد به مقادیر مبنای دو (باینری- Binary) با آن‌ها ارتباط برقرار می‌کنند. در این حالت ولتاژ بالا نشان دهنده ۱ و ولتاژ پایین نشان‌دهنده صفر است و تغییر ولتاژ در یک مدت زمان خاص نشان دهنده پیغام‌های دلخواه است.

هر پین روی AVR دارای یک نام است و بعداً خواهید دید که چگونه از آن‌ها استفاده می‌کنیم. بنابراین اگر یک LED باید به پین شماره ۱۴ وصل شود، می‌توانم بگویم که یک ولتاژ بالا و پایین به پینی داده می‌شود که به صورت PB0 مشخص شده است. اغلب پین‌های روی AVR دارای عملکردهای جانبی نیز هستند و این عملکردها درون پرانتز مشخص شده‌اند. برای مثال RXD و TXD توابع دریافت و ارسال برای

عملکرد پورت سریال بوده و روی پین‌های PD0 و PD1 هستند. در انتهای کتاب، معنی همه عبارت‌های درون پرانتز را متوجه شده و تقریباً از اکثر این عملکردها استفاده کرده‌اید.



شکل ۱-۱: پین‌های AVR و عملکرد آن‌ها.

به صورت داخلی پین‌ها درون بانک‌هایی از هشت پین چیده شده‌اند. به دلیل این که هر بانک، به عنوان مثال بانک B، دارای حداکثر ۸ پین درون خود است، می‌توانید با یک عدد باینری ۸ بیتی به آن‌ها مراجعه کرده تا منبع ولتاژ آن‌ها را فعال یا غیرفعال کنید. جزئیات بیشتری در این زمینه را در فصل ۳ و بعد از آن خواهید دید.

به صورت خلاصه، می‌توانید از درون کد خود با خواندن از و نوشتن به *رجیسترهای I/O* (I/O Register) ویژه درون چیپ به این پین‌ها و عملکردهای مختلف آن‌ها دسترسی داشته باشید. بعد از آن کد شما می‌توانند همانند هر متغیر دیگری به آن‌ها دسترسی پیدا کند. برای مثال، یک رجیستری PINB حاوی حالت پین ورودی برای همه پین‌های درون Bank B است. با استفاده از کد، آن را به این صورت می‌خوانید:

```
thePins = PINB;
```

و سپس می‌توانید هر کاری که می‌خواهید با متغیر thePins انجام دهید که حاوی وضعیت‌های کنونی pin است. به صورت مشابه، یک متغیر رجیستری PORTB را می‌توان همانند یک متغیر عادی استفاده کرد مانند این:

```
PORTB = 42;
```

که با این کار پین‌های خاص درون بانک B برابر با سطح ولتاژ بالای منطقی شده و بقیه برابر با 0 ولت می‌شوند. و تغییر سطح ولتاژ روی این پین‌ها باعث روشن یا خاموش شدن موتورها یا LEDها، پخش صدا روی اسپیکر، یا انکد و انتقال داده‌های عددی می‌شود.

تمام این مباحث ارزش چند فصل دارند و بنابراین اگر موضوع برای شما مشخص نیست نگران نباشید. نکته این است که کدی که نوشته‌اید دارای اثر جانبی فیزیکی روی چیپ است: به صورت مستقیم سطح‌های ولتاژ روی پین‌ها را خوانده یا کنترل می‌کنند. و با این می‌توانید هر کاری انجام دهید. بقیه تنها جزئیات هستند.

برگه داده

تازه‌کارها برای مایکروکنترلرهای AVR اغلب وقتی متوجه می‌شوند که برگه‌های داده (Datasheet) چقدر مفید هستند متعجب می‌شوند. آن‌ها تقریباً به هر سؤالی که درباره عملکرد چیپ دارید پاسخ می‌دهند. برای مثال من شکل ۱-۱ را از صفحه ۲ یک برگه شیوه ایجاد کردم.

اما آن‌ها می‌توانند ترسناک نیز باشند- برگه‌داده‌های سری 48/88/168 برای ATmega حدود ۴۵۰ صفحه است! و برگه‌های داده در نگاه اول به نظر مفید نمی‌رسند- آن‌ها تقریباً هر چیزی درباره عملکرد چیپ را می‌گویند اما تقریباً چیزی درباره چگونگی کار کردن با چیپ را نمی‌گویند. بعضی بخش‌های برگه‌های شیوه تقریباً غیرقابل نفوذ هستند مگر این که اصول درون چیپ را بدانید.

ترفند رفتار کردن با برگه داده همانند یک کتاب مرجع است (به جای این که همانند یک رمان به آن نگاه کنید). اگر یک زبان خارجی جدید یاد می‌گیرید، کار را با باز کردن یک دیکشنری و خواندن آن شروع نمی‌کنید. یک دیکشنری وقتی مفید است که هم اکنون یک ایده ساده درباره چگونگی کارکرد آن زبان دارید اما فراموش کرده‌اید تا به عنوان مثال چگونه به پرتغالی کلمه lemur را بگویید.

من در این کتاب و هنگام یادگیری چگونگی استفاده از توابع و کارکردهای جدید به برگ‌های داده مراجعه می‌کنم. یکی از مهم‌ترین توانایی‌هایی که هنگام کار کردن با این کتاب یاد می‌گیرید، چگونگی استفاده از برگه‌های داده است.

بنابراین اکنون اندکی زمان گذاشته و برگه داده کامل برای چیپی را پیدا کنید که قرار است از آن استفاده کنیم یعنی ATmega168. اولین صفحه را خوانده و فعلاً متوقف شوید. اگر خواننده PDF شما

از امکان فهرست مطالب پشتیبانی می‌کند، می‌توانید یک نگاه به کل محتویات بیندازید. در این کتاب زیاد به برگه شیوه مراجعه خواهیم کرد.

بنابراین بدون هیاهوی بیشتر، به سراغ یک تور کوتاه از سخت‌افزاری می‌رویم که درون یک مایکروکنترلر AVR قرار داده شده و می‌بینیم که برای چه چیزی خوب است.

هسته: پردازشگر، حافظه و I/O

CPU

واحد پردازش مرکزی (CPU) یک AVR در اصلی خیلی به پردازنده کامپیوتر رومیزی یا لپ‌تاپ شما شباهت دارد. این نیز یک مدار الکترونیکی است که دارای یک دسته عملیات منطقی و ریاضی ساخته شده درون آن است و می‌داند لیست این عملیات برای انجام (برنامه شما) و داده‌های مورد نیاز برای اجرای آن‌ها را پیدا کند.

حافظه

مایکروکنترلرهای AVR حداقل سه نوع حافظه مختلف با کاربردهای متفاوت دارند:

۱. حافظه فلش (Flash Memory): کد کامپایل شده شما درون حافظه فلش غیرفرار قرار می‌گیرد. وقتی چیپ خاموش می‌شود (الکتریسیته در آن نباشد) این حافظه پاک نمی‌شود. در حقیقت، تضمین شده است که این نوع حافظه در دمای اتاق هر ۱۰۰ سال یک بیت در میلیون از دست می‌دهد. چگونگی آپلود کد به حافظه فلش را در فصل ۲ می‌بینیم.

۲. حافظه RAM: به صورت طبیعی این حافظه‌ای برای نگه‌داری متغیرهای موقتی هنگام انجام محاسبات است.

۳. حافظه EEPROM: این نوع حافظه سرعت کمی برای نوشتن داشته و مقدار آن کم است اما مشابه با حافظه فلش پاک نمی‌شود.

ساعت‌ها

تمام کامپیوترها به حسی از زمان نیاز دارند. در چیپ‌های AVR، چند ساعت وجود دارد که همگی از مبنای زمانی یکسانی استفاده کرده اما اغلب مقیاس‌های زمانی جداگانه‌ای دارند. ما از نوسانگر RC داخلی به عنوان منبع ساعت اصلی استفاده می‌کنیم. این با حدود ۸ مگاهرتز کار می‌کند. ساعت CPU نیز از ساعت اصلی جدا شده و به صورت پیش‌فرض با سرعت ۱ مگاهرتز کار می‌کند اما در صورت لزوم می‌توان سرعت آن را تا ۸ مگاهرتز کامل رساند.

بعد از ساعت CPU، دیگر ساعت‌های جانبی هستند که اغلب آن‌ها دارای مقیاس‌های خود نسبت به CPU می‌باشند. ساعت‌هایی برای زیرسیستم I/O، مبدل‌های آنالوگ به دیجیتال، RAM، و فلش و EEPROM وجود دارند. وقتی هرکدام از این زیرسیستم‌ها را استفاده می‌کنید باید مقیاس دهنده ساعت را به خاطر

داشته باشید- اغلب باید مقدار آن‌ها را تعیین کنید. این ساعت‌ها از منبع یکسانی آماده که باعث شده همه چیز به صورت منظم با یکدیگر باشند.

خروجی‌ها

تقریباً تمام پین‌های روی چیپ‌های AVR را می‌توان پیکربندی کرده تا بتوان از آن‌ها به عنوان خروجی‌های دیجیتال استفاده کرد، یعنی پین می‌تواند از نرم‌افزار دستور گرفته تا خروجی به صورت سطح ولتاژ یا سطح ولتاژ زمین ایجاد کند. (این سطح‌های ولتاژ را VCC و GND یا زمین می‌نامند.) در هر صورت خروجی دیجیتال قلب و روح برنامه‌نویسی مایکروکنترلر است. این چگونگی صحبت شما با دنیای بیرون است. با جزئیات بیشتری در فصل ۳ به خروجی دیجیتال می‌پردازیم.

خروجی آنالوگ

اگر از پلتفرم آردینو استفاده می‌کنید، می‌توانید بعضی از خروجی‌ها را به عنوان خروجی آنالوگ در نظر بگیرید اما واقعاً هیچ خروجی آنالوگی روی سری‌های AVR نیست. کد آردینو در پشت صحنه آن‌قدر سریع وضعیت پین را بین حالت بالا و پایین تغییر می‌دهد که ولتاژ میانگین در وسط خواهد بود. به این ماژوله کردن پهنای پالس (Pulse Width Modulation) می‌گویند که در فصل ۱۰ توضیح داده می‌شود.

ورودی‌ها

همان طور که می‌توان تقریباً همه پین‌ها را برای خروجی آماده کرد، می‌توان آن‌ها را برای ورودی‌های دیجیتال نیز پیکربندی کرد که در این حالت تشخیص می‌دهند آیا ولتاژ اعمال شده روی پین از بیرون بالا یا پایین است. به خصوص اگر ولتاژ روی پین بزرگ‌تر از نصف ولتاژ اعمالی باشد، چیپ یک بیت درون یک متغیر درونی را برابر با ۱ قرار می‌دهد. اگر ولتاژ کمتر از حد آستانه باشد، همان بیت به عنوان صفر خوانده می‌شود.

یک دکمه را از یک طرف به ولتاژ فراهم آمده وصل کرده و در طرف دیگر از طریق یک مقاومت به زمین وصل می‌کنید. وقتی ولتاژ سمت بالا را می‌خوانید می‌دانید که وقتی ولتاژ بالا (یا یک) است یعنی دکمه آزاد بوده و وقتی ولتاژ پایین (صفر) خوانده شود یعنی دکمه فشرده شده است. اگر اکنون این را به یک پین AVR وصل کنید که برای ورودی آماده شده است، کد شما می‌تواند ببیند آیا دکمه فشرده شده یا نه که برای این کار ولتاژ اعمال شده روی پین برای نزدیک بودن به صفر یا VCC ارزیابی می‌شود.

در فصل ۶ جزئیات زیادی درباره ورودی‌های دیجیتال و دکمه‌ها خواهید دید.

تا این جا یک کامپیوتر کوچک را توصیف کردم که می‌تواند برنامه‌هایی را اجرا کند که مقادیر روی پین را به صورت ولتاژهای منطقی دیجیتال خوانده و می‌نویسند. در اصل این همه چیز است و با استفاده از این فریم‌ورک می‌توانید تقریباً هر چیزی را پیاده کنید. و تعداد بسیار زیادی کاربرد مفید مایکروکنترلر تنها با این عملکرد انجام می‌شوند.

بقیه سخت‌افزار مایکروکنترلر برای آسان‌تر کردن زندگی شما به عنوان یک برنامه‌نویس و مطمئن‌تر کردن وظایف معمول‌تر است.

تجهیزات جانبی: آسان‌تر شدن کار

ارتباطات سریال

یکی از محبوب‌ترین کاربردهای مایکروکنترلرها، استفاده از آن‌ها به عنوان کانکتور بین یک کامپیوتر واقعی و یک سخت‌افزار خاص است. فرض کنید می‌خواهید یک شتاب‌سنج را به بدن خود وصل کرده، ورزش کرده و داده‌ها را به لپ‌تاپ انتقال دهید که یک تصویر سه بعدی از شما را به صورت همزمان نشان می‌دهد. وظیفه مایکروکنترلر ساده است: صحبت کردن با شتاب‌سنج، انجام مقداری ریاضیات و روشن کردن تعدادی LED و ارسال همه داده‌ها به لپ‌تاپ شما. اما شتاب‌سنج به چه زبانی صحبت می‌کند؟ کامپیوتر رومیزی شما چطور؟

AVR دارای سه سخت‌افزار ارتباطی داخلی است. سریال USART در فصل ۵ برای ارتباطات با کامپیوتر رومیزی، مودم‌های رادیویی، و واحدهای GPS مناسب است. SPI برای ارتباطات بسیار سریع روی فواصل بسیار کوتاه با تجهیزاتی مانند حافظه‌ها، ADCها و DACها مناسب است. I2C مشابه با یک شبکه کوچک است که اجازه وصل کردن حداکثر ۱۷ سنسور به یک جفت سیم را می‌دهد. تجهیزاتی که به اطراف حرکت می‌کنند بهتر است از I2C استفاده کنند. این برای شبکه شتاب‌سنج‌ها مناسب است.

به دلیل این که هر کدام از این سخت‌افزارهای سریال درون AVR جدا هستند، می‌توانید از هر سه در یک زمان استفاده کنید. AVR شما می‌تواند تقریباً با هر چیزی ارتباط برقرار کند.

مبدل آنالوگ به دیجیتال

تعدادی سنسور مفید که باید به پروژه‌های خود وصل کنید، نمی‌توانند با زبان بومی مایکروکنترلر شما صحبت کنند؛ بلکه آن‌ها به صورت ولتاژهای آنالوگ دائم صحبت می‌کنند. برای خواندن این مقادیر و اداره آن‌ها همانند اداره داده‌های دیجیتال، باید آن‌ها را روی یک مبدل دیجیتال به آنالوگ (Analog to Digital Converter: ADC). در فصل ۷، از ADCها برای ساختن یک سنسور نوری و نور شب با حساسیت متغیر استفاده می‌کنید. در فصل ۱۲، به سراغ کاربردهای پیشرفته‌تر ADC می‌روم شامل ساختن یک سری سنسور که می‌توانند حرکت درون خانه را شناسایی کرده و با فوق‌نمونه‌برداری (Oversampling) و هموارسازی نمایی (Exponential Smoothing) آشنا می‌شوید که دو تکنیک برای افزایش دقت یا حذف نویز از خواندن‌های ADC هستند.

وقفه‌ها

مسلماً می‌خواهید برنامه شما بتواند به دنیای بیرون واکنش نشان دهد: دکمه‌ای را فشار داده و می‌خواهید چیزی اتفاق بیفتد. اما این نصف لذت نوشتن کد برای یک مایکروکنترلر است. یا شاید بخواهید برنامه

شما کاری را هر چند مدت انجام دهد که برای این کار از تایمرها/شمارنده‌ها استفاده می‌شود. وقفه‌های سطح سخت‌افزار بلیت این کار هستند.

یک *روتین سرویس وقفه (Interrupt Service Routine)* یک تابع نرم‌افزاری است که می‌توانید بنویسید تا به صورت خودکار هر وقت یک شرط وقفه برقرار بود اجرا شود. دلیل نام‌گذاری این است که پردازنده کاری که انجام می‌دهد را متوقف کرده و تابع صحیح را انجام می‌دهد. بعد از این که کار شما با روتین وقفه خاتمه پیدا کرد، پردازنده عملیات عادی خود را از جای قبلی ادامه می‌دهد.

روش‌های زیادی برای فعال کردن وقفه‌ها در مایکروکنترلر AVR وجود دارد. این شرایط وقفه می‌توانند شامل فشردن یک دکمه Reset، تغییر یک مقدار ورودی، یک تیک ساعت درونی، رسیدن به یک مقدار برای شمارنده، و بسیاری موارد دیگر باشند. نکته این است که شرایط وقفه بسیار زیادی وجود داشته و آن‌ها می‌توانند توابع خود را فرابخوانند.

وقفه‌ها و روتین‌های سرویس وقفه برای برنامه‌نویسی پیشرفته AVR بسیار مهم هستند و به همراه تایمرها قسمت بزرگی از نیمه دوم کتاب می‌باشند.

تایمرها/شمارنده‌ها

مایکروپروسسورهای AVR دارای شمارنده‌های داخلی هستند. این شمارنده‌ها در اصل همان چیزی هستند که به نظر می‌رسد- تعداد تغییر سطح ولتاژ یک پین یا یک منبع داخلی را شمارش می‌کنند.

ساده‌ترین کاربرد شمارنده وصل کردن شمارنده داخلی به یک دکمه است. اکنون هر بار برنامه بخواهد، می‌توانید بگویید چند مرتبه دکمه فشرده شده است که برای این کار رجیستری شمارنده را می‌خوانید. (می‌توانید روی شمارنده نیز بنویسید تا بتوانید به عنوان مثال در ابتدای هر روز مقدار آن را برابر با صفر قرار دهید.)

شمارنده‌ها وقتی با ساعت‌ها جفت شوند بسیار کاربردی می‌شوند و به همین خاطر است که اغلب به آن‌ها تایمر/شمارنده گفته می‌شود. با یک ساعت و یک شمارنده، می‌توانید مقدار زمان انجام رویدادها را اندازه گرفته یا این که فرکانس آن رویداد را مشخص کنید. حتی می‌توانید خروجی به پین‌های AVR خاصی بدهید (در دیاگرام‌ها پین 0CRxn) و ساب‌روتین‌هایی را با فاصله زمانی مشخصی انجام دهید. تجهیزات جانبی تایمر/شمارنده قابلیت پیکربندی زیادی داشته و عملکردهایی به مراتب بیشتر از چیزی که فکر می‌کنید فراهم می‌آورند.

کاربردهای زیادی از تایمرها/شمارنده‌ها در نیمه دوم کتاب خواهید دید.

خانواده AVR مایکروکنترلرها

با این که ما در این کتاب روی AVR مدل ATmega168 متمرکز می‌شویم، اما بعداً می‌توانید از چیپ‌های دیگری نیز در پروژه‌های خود استفاده کنید. چیپ‌های مایکروکنترلر AVR بسیار زیادی برای انتخاب

وجود داشته که هرکدام دارای قابلیت‌های متفاوتی با هزینه‌های مختلف هستند. پیدا کردن مورد مناسب برای پروژه می‌تواند ترسناک باشد. ممکن است ساعت‌ها وقت صرف انتخاب چیپی کنید که دارای امکانات مورد نظر شما و حافظه مناسب برای کد است و سپس ارزان‌ترین بخش را پیدا کنید.

اگر روی یک نمونه کار کرده و می‌خواهید تا حد امکان سریع کار کنید، احتمالاً نباید نگران صرف ۵۰ سنت بیشتر برای هر چیپ باشید. بهتر است همیشه چند نوع چیپ در دسترس داشته باشید و هرکدام که مناسب‌تر کار شما بود را انتخاب کنید. در این جا سری را می‌بینید که من همیشه استفاده می‌کنم:

کوچک: ATtiny45

چیپ‌های سری x5 کوچک و ارزان هستند که برای زمانی که تنها به ۵ پین I/O نیاز داشته باشید مناسب هستند. آن‌ها دارای ساعت جانبی سرعت بالایی هستند که می‌تواند حداکثر تا ۶۶ مگاهرتز سرعت داشته باشد که باعث شده برای کاربردهای PWM و استفاده به همراه کتابخانه فریم‌ور V-USB (Firmware) برای ایجاد تجهیزات سخت‌افزاری USB مناسب باشند.

تنها تفاوت بین مدل‌های ۲۵، ۴۵ و ۸۵ مقدار حافظه برنامه (۲، ۴ و ۸ کیلوبایت) و قیمت است. برای من ۴ کیلوبایت حافظه کافی است.

یک ATtiney45s حدود ۱ دلار تک فروشی و ۰/۶۵ دلار عمده قیمت دارد.

متوسط: ATtiny44

وقتی به PWM سرعت بالای Tiny45 نیاز ندارید این یک مورد جذاب است. برای تنها چند سنت بیشتر، ۱۱ پین I/O و یک تایمر ۱۶ بیت به دست می‌آورید. حتی با این که آن‌ها تقریباً مورد جدیدی در بازار هستند، اما این چیپ‌ها به یکی از ملزومات من برای پروژه‌های کوچک تبدیل شده‌اند.

ATtiny44s در زمان نوشتن این کتاب حدود ۱/۱۵ دلار برای تک و ۰/۷۵ دلار به صورت عمده قیمت دارد.

بزرگ: خانواده ATmega xx8

چیپ‌های Mega xx8 مواردی لوکس محسوب می‌شوند. اگر می‌خواهید روی یک سری چیپ متمرکز شوید، این همان است. در این سطح، ۲۰ پین I/O، USART سخت‌افزاری، سه تایمر و یک حافظه برنامه ۱۶ کیلوبایتی خواهید داشت. به همین خاطر است که پلتفرم آردینو برمبنای Mega 168s و 328s پرطرفدار شده است.

و به دلیل این که Mega 88,88,168,328 همگی دارای امکانات یکسان به جز حافظه برنامه هستند، اغلب می‌توان یک چیپ را عوض کرده تا چند دلاری صرفه‌جویی کرد. Mega48s عملکرد یکسان با قیمت نصف Mega168s به شما می‌دهد به شرطی که به حافظه اضافی نیاز نداشته باشید.

ATmega168s حدود ۲/۲۵ دلار به صورت تک یا ۱/۵۰ دلار به صورت عمده قیمت دارد.

امروزه انتخاب‌های بسیار زیادی به غیر از این خانواده‌های چیپ وجود دارند. با جستجو به دنبال mega168 در یک فروشگاه الکترونیکی بیش از ۵۰ نتیجه خواهید داشت مانند ATmega168PA-10PU، ATmega168-AU و مانند آن‌ها.

حروف بعد از نام بخش (168P یا 168PA یا 168A) نشان دهنده نسخه‌های مختلف چیپ هستند. مدل‌های V در ولتاژ پایین‌تری کار می‌کنند اما سرعت آن‌ها نیز کاهش یافته است. سری‌های P و V طراحی‌های قدیمی هستند. سری‌های A و PA نشان‌دهنده طراحی‌های جدیدتری هستند که از توان کمتری استفاده کرده (P) یا روی بازه کامل سرعت و ولتاژ عمل می‌کنند (A) یا هردو.

پسوند نشان دهنده اندازه پکیج بوده و اغلب با حروف توصیف می‌شود. PU(PDIP) بزرگ‌ترین بوده و قطعه استاندارد است. AU (TQFP) دارای نصب سطحی با فاصله ۱ میلی‌متری بوده و اگر با SMT راحت باشید کاملاً مناسب است. لحیم پکیج‌های MU با دست بسیار سخت است.

عدد درون پسوند یک درجه سرعت است و به عنوان مثال چیپ‌هایی که با 10 خاتمه می‌یابند حداکثر تا ۱۰ مگاهرتز سرعت دارند. وقتی هیچ پسوند عددی نباشد (مانند چیپ‌های مدرن)، چیپ با سرعت کامل کار می‌کند که معمولاً ۲۰ مگاهرتز در ۵ ولت است.

بنابراین کدام نوع و کدام پکیج؟ P، A یا PA همگی مناسب هستند و در اصل تفاوت چندانی بین آن‌ها نیست. اگر گیر کردید به سراغ A یا PA بروید. برای پکیج‌ها، اگر تجربه لحیم‌کاری ندارید PU (PDIP) را بردارید و اگر از کار نصب سطحی لذت می‌برید به سراغ AU(TQFP) بروید.