تاکنون در این دوره، با 2 نوع حافظه در آردوینو آشنا شدیم. حافظه Flash که وظیفه‌ی ذخیره‌‌ی برنامه را به عهده دارد و حافظه‌ی SRAM که حافظه‌ی موقت است و متغیرها را هنگام اجرای برنامه در خود نگه می‌دارد. اما حافظه‌ی سومی هم در بردهای آردوینو هست که حافظه‌ای دائمی است به نام EEPROM.

منظور از حافظه‌ی دائمی حافظه‌ای است که با قطع جریان برق یا ریست کردن میکروکنترلر پاک نشود. گاهی دسترسی به برخی داده‌ها بعد از فاصله‌های زمانی مشخص لازم است. مثلا فرض کنید بخواهیم چگونگی تاثیرگذاری شرایط محیطی را بر عملکرد یک صفحه خورشیدی بررسی کنیم. دما، رطوبت و شدت نور از پارامترهای تاثیرگذار بر این عملکردند. می‌توانیم مقدار هر پارامتر را با حسگر مخصوص خود به دست بیاوریم. اما با یک بار خواندن این مقادیر نمی‌توانیم به نتیجه‌ای برسیم زیرا با دانستن تغییرات آنها در بازه‌های زمانی مختلف و میزان اثرگذاری هر کدام است که می‌توانیم به جواب درست برسیم. پس باید این داده‌ها را در بازه‌های زمانی مختلف جمع‌آوری کنیم و با ذخیره‌سازی دائمی آنها مثلا بعد از یک ماه به خواسته‌ی خود برسیم.

دو نوع حافظه دائمی رایج در اختیار داریم: حافظه‌ی EEPROM که حافظه‌ی داخلی روی خود برد آردوینو است و حافظه‌های SD Card که حافظه‌های خارجی‌اند که برای ذخیره اطلاعات روی آنها احتیاج به ماژول جانبی است. ما در این بخش که بخش پایانی دوره‌ی آردوینو است، به EEPROM می‌پردازیم و آموزش حافظه‌های SD را به بخش آموزش‌ ماژول‌های آردوینو می‌سپاریم.

کتابخانه EEPROM

کار با EEPROM بسیار راحت است. اول آنکه چون سخت‌افزاری بر روی خود برد آردوینو است، احتیاج به هیچ مدار جانبی‌ای نیست. دوم آنکه چون از ویژگی‌های خود میکروکنترلر است، کتابخانه‌ی استانداری دارد که بر روی خود IDE شرکت آردوینو نصب است و کار با آن نیز بسیار ساده است.

پس از فراخوانی کتابخانه #include می‌توان از آن استفاده کرد و نباید به آن اسمی اختصاص داد؛ دستورات با خود اسم EEPROM استفاده می‌شوند.

به سادگی می‌توان دریافت که کار ما با هر حافظه‌ی دائمی‌ای به دو بخش کلی محدود می‌شود: نوشتن روی حافظه و خواندن از آن. دستوراتی که در ادامه می‌آید چگونگی انجام این دو کار را شرح می‌دهد.

**.write()**

**.read()**

**.update()**

**.put()**

**.get()**

چند نکته درباره‌ی EEPROM

* روند نوشتن متغیرها و ذخیره آنها در این حافظه حدود 3.3 میلی‌ثانیه زمان می‌برد.
* این حافظه حداکثر قابلیت 100000 بار نوشتن و پاک‌کردن را دارد و در برنامه باید تعداد این دفعات را درنظر گرفت تا در آینده خطا ایجاد نکند.
* از آن جا که با هر بار دستور write از ظرفیت نوشتن بر روی حافظه کم می‌شود، بهتر است از این دستور یک بار و آن هم در تابع setup استفاده شود. (برای مقدار دهی اولیه حافظه در آدرس معین)
* دستور update برخلاف دستور write دفعات نوشتن و پاک کردن حافظه را کم نمی‌کند چون که در صورت مشابه بودن مقدار متغیر دیگر آن را تغییر نمی‌دهد.
* دستور put از روش دستور update برای ذخیره داده‌ها استفاده می‌کند و در صورت مشابه‌بودن مقدار متغیر از دوباره نوشتن آن خوداری می‌کند.

مثال