به نام خدا

سیستم‌های نهفته و بی‌درنگ

تمرین سوم

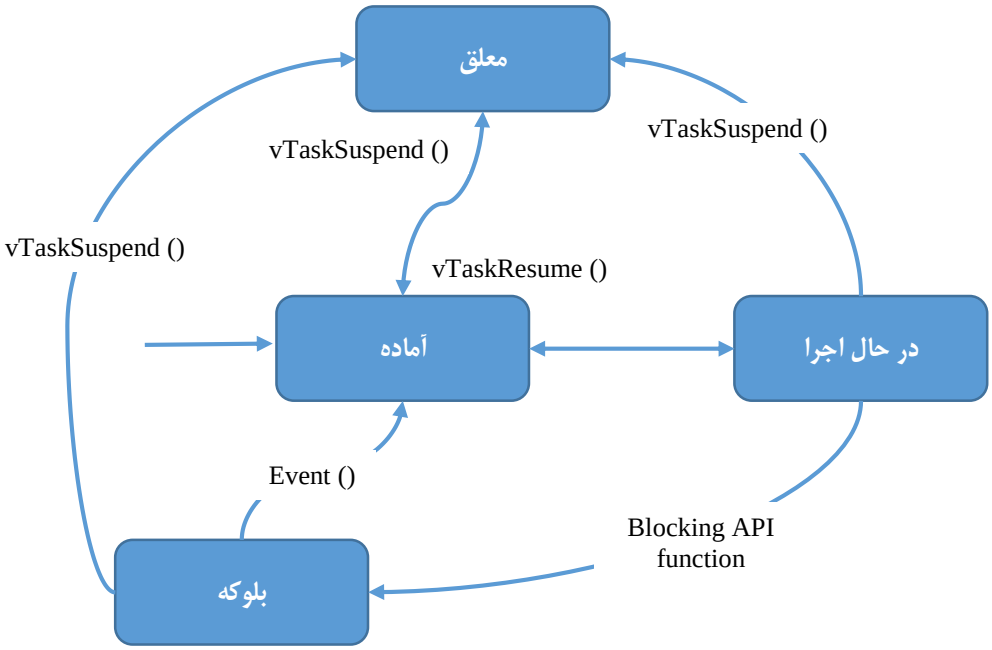
امروزه سیستم‌های نهفته‌ی توان پایین در زمینه‌های مختلف استفاده می‌شود. برای مثال می‌توان به پهپادها، خودروهای خودران، ربات‌های زنده یاب و ... اشاره کرد. برای رفع شرط توان پایین بودن سامانه، اغلب ما با محدودیت استفاده از پردازنده‌هایی با هسته‌های زیاد مواجه هستیم. از جهت دیگر، کارهای حیاتی سامانه، مانند دریافت مقادیر حسگرها و تصمیم‌گیری بر اساس آن‌ها باید بی‌درنگ انجام شود و کارهای حیاتی باید اولویت بالاتری داشته باشند. از دیگر شرایط سامانه‌های بی‌درنگ می‌توان به شرط‌های زمانی ثابت اشاره‌کرده و در صورت عدم رعایت این شروط، کار سیستم با مشکل مواجه می‌شود. به همین جهت برای زمان‌بندی و تعیین اولویت به ساختاری مانند یک سیستم‌عامل بسیار سبک با مصرف حداقل منابع نیاز داریم. برای رفع این نیازها سیستم‌عامل‌های بی‌درنگ برای بسترهای مختلف سخت‌افزاری به وجود آمدند. از این سیستم‌عامل‌ها می‌توان به موارد زیر اشاره کرد.

- ROS برای بستر Jetson
- FreeRTOS برای بسترهای مبتنی بر ARM، AVR، MicroBlaze، IA-32 و ...
- Contiki برای بسترهای MSP430، AVR و ARM

برای موارد بیشتر می‌توانید به [۱] مراجعه کنید.

در این تمرین و در ادامه ما با سیستم‌عامل متن‌باز FreeRTOS روی بستر Arduino بیشتر آشنا خواهیم شد. به‌طورکلی، FreeRTOS مجموعه‌ای از کتاب‌خانه‌های C و یک زمان‌بند است. در هر تیک (که در Arduino ۱۵ میلی‌ثانیه است) زمان‌بند وقفه‌ای تولید می‌کند و کارهای آماده برای اجرا را در نظر می‌گیرد. ابتدا کارهای با اولویت بالا را انجام می‌دهد و اگر دو کار اولویت برابر داشتند از زمان‌بندی نوبت گردشی برای انتخاب کار استفاده می‌کند. در FreeRTOS امکان بلوکه کردن کارها وجود دارد که کار از لیست کارهای آماده خارج می‌شود. همچنین امکان معلق کردن کار و غیرقابل زمان‌بندی کردن آن تا زمان مشخص وجود دارد.

به‌طور خلاصه می‌توان روند اجرا و چگونگی تغییر وضعیت کارها را به‌صورت زیر نشان داد:



شکل (۱): نمودار وضعیت کارها در FreeRTOS

حالت‌های گفته‌شده را می‌توان به‌صورت زیر شرح داد:

۱. **در حال اجرا:**
   - کار در حال اجرا است.

۲. **آماده:**
   - کار آماده اجرا است ولی کاری با اولویت برابر یا بالاتر در حال اجرا است.

۳. **بلوکه:**
   - کار منتظر یک رخداد است.
   - **زمان:** اگر کار تابع ()vTaskDelay را فراخوانی کند، آنگاه تا پایان مدت‌زمان تأخیر بلوکه می‌شود
   - **منابع:** کاری که منتظر صف و وقایع سمافور است بلوکه می‌شود

۴. **معلق**
   - مانند بلوکه است ولی کار منتظر رخداد نیست.
   - کار فقط با استفاده از فراخوانی واسطهای برنامه‌نویسی کاربردی () xTaskResume و () vTaskSuspend خارج و وارد این حالت می‌شود.

توجه داشته باشید که بدنه تابع loop در برنامه‌ی Arduino باید خالی باشد و زمان‌بند سیستم‌عامل در پایان تابع setup کار خود را شروع می‌کند.

برای اضافه کردن کتابخانه‌ی FreeRTOS در Arduino، فایل همراه این سند را از حالت فشرده درآورده و در مسیر زیر کپی کنید.

<Your Drive>:\Users\<Username>\Documents\Arduino\libraries

سپس خط زیر را به برنامه‌ی خود اضافه کنید.

```
#include <Arduino_FreeRTOS.h>
```

برخی از توابع مورداستفاده در این تمرین از کتابخانه FreeRTOS عبارت‌اند از:

- ```
  BaseType_t xTaskCreate(TaskFunction_t pvTaskCode, const char * const pcName,
  configSTACK_DEPTH_TYPE usStackDepth, void *pvParameters, UBaseType_t uxPriority,
  TaskHandle_t *pxCreatedTask)
  ```

  از این تابع برای تولید کار استفاده می‌شود. آرگومان‌های آن به ترتیب به شرح زیر است:

  - اشاره‌گر به تابع ورودی (نام تابع موردنظر را بنویسید). در حالت کلی کارها به‌صورت حلقه‌های بی‌نهایت پیاده‌سازی می‌شوند.
  - نام کار. عمدتاً برای عیب‌یابی استفاده می‌شود؛ اما از آن برای به دست آوردن Task handle نیز می‌توان استفاده کرد.
  - تعداد کلمات (بایت نه!) پشته‌ی کار را تعیین می‌کند. برای مثال اگر عرض پشته ۱۶ بیت باشد و usStackDepth برابر ۱۰۰ باشد، ۲۰۰ بایت تخصیص داده می‌شود و اگر عرض پشته ۳۲ بیت باشد و usStackDepth برابر ۴۰۰ باشد، ۱۶۰۰ بایت برای پشته‌ی کار تخصیص داده می‌شود. توجه داشته باشید مقدار تابع به عوامل بسیاری بستگی دارد. برخی از این عوامل عبارت‌اند از:
    - عمق فراخوانی تودرتو توابع
    - تعداد و اندازه‌ی متغیرهای تعریف‌شده در محدوده‌ی تابع
    - تعداد آرگومان‌های تابع
    - معماری پردازنده
    - کامپایلر
    - سطح بهینه‌سازی کامپایلر
  - پارامترهای موردنیاز که به کار پاس داده می‌شود. این آرگومان اجباری نیست و در صورت عدم استفاده از NULL استفاده شود.
  ```

- اولویت کار برای اجرا
- برای پاس دادن یک handle به کار تولیدشده استفاده می‌شود. این آرگومان اجباری نیست و در صورت عدم استفاده از NULL استفاده شود.
    - `void vTaskDelay(const TickType_t xTicksToDelay)`

از این تابع برای بلوکه کردن کار برای مدت‌زمان مشخص است. ارگمان ورودی این تابع، مدت‌زمان بلوکه کردن کار به تیک است. در Arduino هر تیک برابر ۱۵ میلی‌ثانیه است.

در این تمرین شما با استفاده از FreeRTOS یک پورت مبدل آنالوگ به دیجیتال را بین دو حس‌گر به اشتراک می‌گذارید و با تعیین تأخیرهای مناسب، مقادیر خوانده‌شده از حس‌گرها را روی یک صفحه‌ی LCD 16x2 به کاربر نشان دهد.

قطعات موردنیاز عبارت‌اند از:

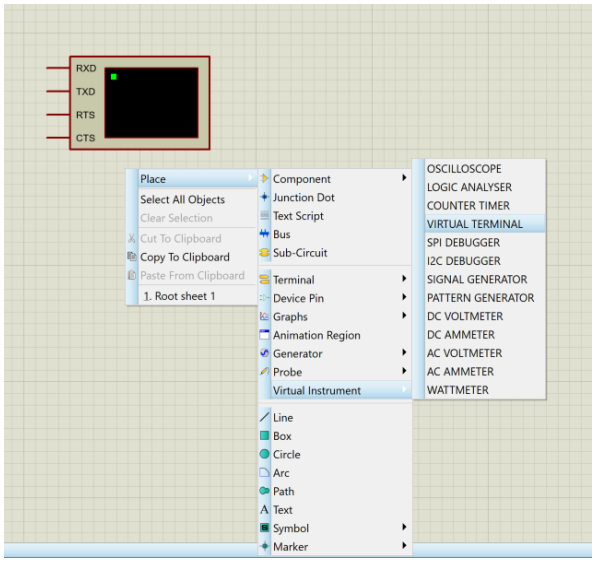
- یک Arduino Mega2560
- یک حس‌گر دما (LM35)
- یک حس‌گر نور (LDR)
- یک LCD 16x2
- دو ترانزیستور PN2222A
- دو مقاومت ۱۲۰۰ اهمی
- ۱ مقاومت ۱۵۰ اهمی

برای کارکرد LCD 16x2 کتابخانه LiquidCrystal.h را به کد خود اضافه کنید و برای ساخت شیء از این کتابخانه و معرفی پین‌های مورداستفاده از خط زیر استفاده کنید.

```
LiquidCrystal <InstanceName>(RS,RW,E,D0,D1,D2,D3,D4,D5,D6,D7,D8)
```

برای مشخص کردن محل نوشتن روی صفحه از متد setCursor(x,y) و برای چاپ متن موردنظر از متد print() استفاده کنید.

از VIRTUAL TERMINAL می‌توانید برای چاپ مقادیر نوشته‌شده روی پورت سریال و برای عیب‌یابی استفاده کنید. محل این قطعه در شکل زیر نشان داده‌شده است.



شکل (۲): محل VIRTUAL DISPLAY

برای استفاده، پورت RXD قطعه را به TXD و TXD قطعه را به RXD موجود روی Arduino متصل کنید.

ترانزیستورها نقش کلیدهای کنترل شونده با ولتاژ را دارند. پایه‌ی بیس را به یکی از پورت‌های دیجیتال Arduino وصل کنید و برای محدودسازی جریان عبوری از مقاومت‌های ۱۲۰۰ اهمی استفاده کنید. برای وصل شدن کلید، پایه‌ی بیس را برابر ۱ منطقی قرار دهید. برای تعیین مقدار خروجی دیجیتال، از تابع digitalWrite(pin,value) استفاده کنید.

از مقاومت ۱۵۰ اهمی برای اندازه‌گیری مقدار حسگر نور استفاده کنید.

برای خواندن مقدار مبدل آنالوگ به دیجیتال از تابع analogRead(pin) استفاده کنید.

توجه داشته باشید جهت کارکرد درست مبدل آنالوگ به دیجیتال پایه‌ی VCC را به Power و GND را به GROUND متصل کنید.

لینک‌های زیر نیز می‌تواند شما را در انجام کارهای گفته‌شده کمک کند:

https://www.freertos.org/a00106.html

https://www.arduino.cc/en/Tutorial/HelloWorld

https://www.arduino.cc/en/Reference/LiquidCrystalConstructor

https://www.arduino.cc/en/Reference/LiquidCrystal

https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/

https://teachmetomake.wordpress.com/how-to-use-a-transistor-as-a-switch/

پ.ن: لطفاً فایل پروژهProteus ، پروژه Arduino و فایل hex به‌دست‌آمده را در قالب یک فایل zip با ساختار نام‌گذاری StudentID.zip در سامانه بارگذاری کنید.

با آرزوی موفقیت

[۱] https://en.wikipedia.org/wiki/Comparison_of_real-time_operating_systems