راه‌اندازی میکروکنترلر STM32f407VGT6 و کار با GPIO

هدف از این تمرین راه‌اندازی بخش‌های اولیه و اصلی میکروکنترلر و آشنایی با نحوه‌ی کار با GPIO می‌باشد. برای این منظور یک پروژه‌ی سیستم آبیاری خودکار به عنوان مثال تعریف شده است.

این سیستم برای سنجش میزان رطوبت خاک و آبیاری خودکار به کار خواهد رفت.

سیستم مذکور متشکل از چندین بخش خواهد بود. این بخش‌ها شامل ماژول رطوبت‌سنج، میکروکنترلر، keypad، LCD و شیر آب می‌شود.

بخش ۱) ماژول رطوبت‌سنج خاک:

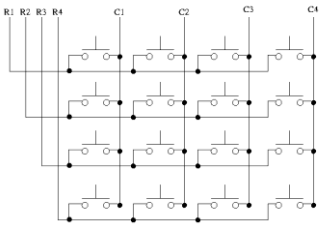
برای سنجش میزان رطوبت خاک انواع مختلفی از ماژول‌ها وجود دارند. ماژول مورد استفاده در این تمرین، YL-69 خواهد بود (ماژول TE215 هم مشابه YL-69 بوده و تفاوت چندانی با آن ندارد). دلیل انتخاب این ماژول، قیمت کم آن و مناسب بودن برای پروژه‌های ساده می‌باشد. شکل زیر این ماژول را نشان می‌دهد:

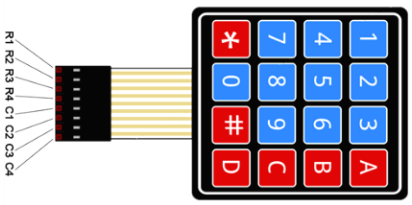
این ماژول دارای چهار پین VCC، GND، AO و DO است. پین AO مربوط به خروجی آنالوگ و پین DO مربوط به خروجی دیجیتال می‌باشد. خروجی آنالوگ ولتاژی متناسب با میزان رطوبت خاک تولید می‌کند و به منظور اندازه‌گیری این ولتاژ نیاز به راه‌اندازی واحد ADC میکروکنترلر است که در این تمرین ما با این پین کاری نخواهیم داشت. خروجی دیجیتال بدین شکل است که اگر رطوبت از حد معینی کمتر باشد، مقدار 1، و اگر از آن حد معین بیشتر باشد مقدار 0 به خود می‌گیرد. این حد هم توسط پتانسیومتر آبی‌رنگ که روی ماژول قرار دارد، قابل تنظیم است.

پین DO را به پین PB7 میکروکنترلر وصل کنید.

بخش ۲) کیپد (keypad):

کیپدها در واقع ماتریسی از push buttonها هستند که در یک پد قرار گرفته‌اند. شکل سمت راست زیر یک کیپد 4x4 و شکل سمت چپ نحوه‌ی اتصالات درون آن را نشان می‌دهد.





مزیت کیپد این است که تعداد پین کمتری برای اتصال نیاز دارد (مثلا برای کیپد فوق ۸ پین به جای ۱۶ پین). اما خواندن وضعیت کلیدها به سادگی یک push button نیست. الگوریتم‌های مختلفی برای این کار ارائه شده است. یکی از این الگوریتم‌ها شیفت دادن یک 0 بین سطرهای کیپد و خواندن وضعیت ستون‌های آن است.

برای این کار چهار پین PD8:PD11 از میکروکنترلر را در مد خروجی قرار داده و این چهار پین را به پین‌های R1:R4 کیپد وصل کنید. و پین‌های PD12:PD15 را در مد ورودی با مقاومت pull-up داخلی قرار داده و به پین‌های C1:C4 کیپد وصل کنید. مقدار [R1:R4] بین چهار حالت 0111، 1011، 1101 و 1110 دائما در حال چرخش خواهد بود و مقدار خوانده شده‌ی [C1:C4] مشخص‌کننده‌ی این خواهد بود که کدام کلید فشرده شده است. مثلا فرض کنید مقادیر پین‌های R1:R4 در ابتدا به صورت [R1:R4]=0111 باشد، در حالتی که هیچ کلیدی فشرده نشده باشد، [C1:C4]=1111 خواهد بود. اگر مثلا [C1:C4]=1101 باشد، می‌توان فهمید که کلید بین C3 و R1 (کلید 3 در کیپد شکل فوق) فشرده شده است. سپس [R1:R4]=1011 شده و دوباره وضعیت پین‌های [C1:C4] خوانده خواهد شد تا معلوم شود آیا کلیدی از سطر دوم کیپد فشرده شده است یا نه، و به همین ترتیب برای سطر سوم و چهارم.

فرمت کلی کدی که نوشته می‌شود به صورت زیر خواهد بود:

```
unsigned char keys[16]={'1','2','3','A','4','5','6','B','7','8','9','C','*','0','#','D'};

unsigned char keypad_scan()
{
  int position;

  for (int i=0; i<4; i++){
    /* circular shift of "0111" by one bit on PD8:PD11*/

    /* read columns on PD12:PD15 to see which key is pressed */

    /* calculate the positin of pressed key */

    return(keys[position]);
  }
}
```

با توجه به این که تابع keypad_scan فوق در حلقه‌ی while اصلی مورد استفاده قرار خواهد گرفت و با توجه به بالا بودن فرکانس کاری میکروکنترلر نسبت به فرکانس فشرده شدن کلید توسط دست، و همچنین لرزش دست هنگام فشردن یا رهاسازی کلید (مشکل Switch Bounce)، اگر ملاحظات لازم انجام نشود، اسکن کلید فشرده شده با مشکل مواجه می‌شود. راهکارهایی برای حل این دو مشکل نیز ارائه و سپس در کد لحاظ کنید.

بخش ۳) شیر آب الکتریکی:

باز و بسته شدن شیر آب با توجه به میزان رطوبت، و میزان باز ماندن شیر آب با توجه به زمانی که توسط کاربر (با استفاده از کیپد) وارد می‌شود، تعیین خواهد شد.

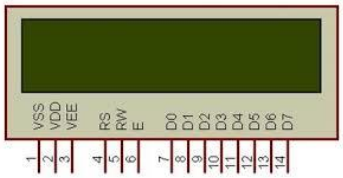
نکته‌ی قابل توجه این است که شیر آب را نمی‌توان مستقیما به یکی از پین‌های میکروکنترلر وصل کرد. چون پین‌های میکروکنترلر جریان‌دهی محدودی دارند و نمی‌توان جریان زیادی از آن‌ها کشید. از طرفی شیر آب جریان نسبتا زیادی هنگام باز بودن نیاز دارد. لذا باید مداری طراحی شود تا این جریان را تامین کند. حتی در بیشتر موارد نیز ولتاژ کاری شیر از 3.3V بیشتر است. به عنوان نمونه یک شیر برقی سلونوئیدی 24V,1.2W,DC در زیر مشاهده می‌شود. این شیر در حالت عادی بسته بوده و با اعمال ولتاژ لازم به دو پایه‌اش مسیر آب را باز می‌کند.

بلوک دیاگرام مدار طراحی شده در این قسمت به صورت زیر خواهد بود:

بدین صورت که با 1 شدن پین PB6 میکروکنترلر، پایه‌ی SW-In به پایه‌ی SW-Out وصل شود. با این شرط که حداکثر جریان کشیده شده از پین PB6، از I_{IO}، که نشان دهنده‌ی حداکثر جریان قابل تامین توسط یک پین میکروکنترلر است، کمتر باشد. همچنین پین PB6 از لحاظ الکتریکی از پایه‌های SW-In و SW-Out ایزوله باشد.

بخش ۴) LCD:

برای نمایش اطلاعات از LCD می‌توان کمک گرفت. LCDها در دو نوع کاراکتری و گرافیکی وجود دارند. LCDهای گرافیکی قابلیت نمایش اطلاعات با دقت بیشتر را دارند. ولی راه‌اندازی پیچیده‌تر و قیمت بیشتری نیز دارند. همچنین هر دو LCD کاراکتری و گرافیکی در اندازه‌های مختلفی تولید می‌شوند. ما از LCD کاراکتری 2x16 به نام LM016L استفاده خواهیم کرد. تراشه کنترلر این LCD، HD44780 است. این LCD در شکل زیر نشان داده شده است:



شکل سمت راست LCD و شکل سمت چپ پایه‌های آن را نشان می‌دهد. پایه‌های VSS و VDD که مربوط به تغذیه هستند. پایه‌ی VEE هم برای تنظیم روشنایی کاراکترهای نمایش داده شده به کار می‌رود. این تنظیم روشنایی معمولا توسط یک پتانسیومتر انجام می‌شود. پایه‌های RS و RW و E پایه‌های کنترلی و پایه‌های D0 تا D7 مربوط به دیتا هستند. همچنین تراشه‌ی HD44780 امکان استفاده از دیتا به دو صورت 8 بیتی و 4 بیتی را فراهم کرده است. مزیت روش 4 بیتی استفاده از چهار بیت کمتر نسبت به حالت دیتای 8 بیتی است. ما نیز از مد چهار بیتی برای اتصال به میکروکنترلر استفاده خواهیم کرد. در این حالت فقط چهار پایه از پایه‌های دیتا مورد استفاده قرار می‌گیرند و به چهار پایه‌ی دیگر کاری نداریم. پس بدین ترتیب پایه‌های D4 تا D7 را به پین‌های PC4 تا PC7 میکروکنترلر وصل کنید. پایه‌های RS و E را به پین‌های PC0 و PC1 وصل کنید. پایه‌ی VEE را هم به پتانسیومتر وصل کنید. پایه‌ی $R\bar{W}$ هم مربوط به انتخاب حالت read و write است و چون در این تمرین چیزی از LCD نمی‌خوانیم و فقط در آن می‌نویسیم، این پایه را به GND وصل کنید.

کدی که برای استفاده از LCD می‌نویسید را در یک کتابخانه به اسم lm016.h ذخیره کنید تا بتوان از توابع آن در کد تابع اصلی استفاده کرد. قسمتی از کتابخانه به صورت زیر خواهد بود و بقیه‌ی کد را شما کامل خواهید کرد:

```
                /* Pin Configuration */
//RS - Register select pin
#define LCD_RS_PORT     GPIOC
#define LCD_RS_PIN      GPIO_PIN_0
//E - Enable pin
#define LCD_E_PORT      GPIOC
#define LCD_E_PIN       GPIO_PIN_1
//D4 - Data 4 pin
#define LCD_D4_PORT     GPIOC
#define LCD_D4_PIN      GPIO_PIN_4
//D5 - Data 5 pin
#define LCD_D5_PORT     GPIOC
#define LCD_D5_PIN      GPIO_PIN_5
//D6 - Data 6 pin
#define LCD_D6_PORT     GPIOC
#define LCD_D6_PIN      GPIO_PIN_6
//D7 - Data 7 pin
#define LCD_D7_PORT     GPIOC
#define LCD_D7_PIN      GPIO_PIN_7

              /* LCD Initialization */
void lcd_init(){

}

              /* Print a character on the current Cursor position*/
void lcd_putchar(char character_p){

}

              /* Move Cursor to the position defined by row and col */
void lcd_set_cursor(uint16_t row, uint16_t col){

}

              /* Print a string on the current Cursor positin */
//This function can use the "lcd_putchar" and "lcd_set_cursor functions
void lcd_puts(char *string_p){

}

              /* Clear all contents of LCD */
void lcd_clear(){

}
```

دقت داشته باشید چون فقط از ۴ پین قسمت دیتا استفاده کردیم، یک داده‌ی 8 بیتی که قرار است به کنترلر LCD فرستاده شود را، به دو قسمت high nibble و low nibble تقسیم کنید و سپس به ترتیب آن‌ها را روی پایه‌های [D4:D7] بفرستید.

بخش ۵) میکروکنترلر:

در این بخش از همان برد طراحی شده در تمرین اول استفاده می‌کنیم. یعنی فرض کنید آن برد را داریم و در این تمرین برد دومی را طراحی می‌کنیم که شامل بخش‌های ۱ تا ۴ این تمرین خواهد بود. اتصال این دو برد نیز از طریق سیم و با استفاده از پین‌هدرهای موجود روی دو برد انجام خواهد شد.

عملکرد کدی که برای میکروکنترلر نوشته می‌شود به صورت زیر است:

میکروکنترلر دائما وضعیت پین PB7 (که به خروجی دیجیتال رطوبت‌سنج وصل شده است) را چک می‌کند و در صورتی که رطوبت خاک کم شود، پین PB6 (که به شیر آب وصل است) را به وضعیت high برده (باز شدن شیر آب) و به مدت زمان t_open آن را در وضعیت high نگه داشته و سپس دوباره آن را در وضعیت low قرار دهد. LCD نیز در حالت عادی، در سطر اول عبارت "Moisture: Wet" یا "Moisture: Dry" را با توجه به میزان رطوبت خاک نشان دهد و در سطر دوم عبارت "Open Time: 1 Min" یا "Open Time: 2 Min" یا "Open Time: 3 Min" را با توجه به مقدار t_open نشان دهد. t_open یک متغیر است که مقدار باز ماندن شیر آب را در هر بار باز شدنش بر حسب دقیقه نشان می‌دهد. مقدار اولیه‌ی t_open برابر 1 می‌باشد. با فشردن کلید A در کیپد، LCD عبارت "Choose Time from" را در سطر اول و عبارت "Min: {1,2,3}" را در سطر دوم نشان داده و سپس منتظر فشرده شدن کلید باشد. اگر کلید فشرده شده یکی از سه کلید 1 یا 2 یا 3 باشد، آن را در متغیر t_open ذخیره کرده و حالت نمایش LCD را به حالت قبل برگرداند. اگر کلید دیگری فشرده شود هیچ اتفاقی نیفتد و همچنان منتظر فشرده شدن کلید بماند.

در دو قسمت از کد، نیاز به دانستن زمان سپری شده داریم. این دو قسمت شامل زمان مربوط به باز ماندن شیر آب و زمان مربوط به وقفه‌های زمانی لازم بین بعضی دستورات LCD است. برای پیاده‌سازی قسمت‌هایی که زمان‌دار هستند در عمل از واحدهای Timer موجود در خود میکروکنترلر استفاده می‌شود. برای ایجاد وقفه‌های زمانی بدون تایمر، از توابع delay استفاده می‌شود. این توابع در ساده‌ترین حالت یک رشته از دستورات مشخص را به تعداد دفعات معینی تکرار می‌کنند. با ضرب T_{clk} در تعداد سیکل لازم برای اجرای آن دستورات، مدت زمان لازم برای اجرای آن دستورات به دست می‌آید. سپس مدت زمان کل اجرا را می‌توان توسط تعداد دفعات اجرای این دستورات تعیین کرد. این روش دقت و قابلیت اطمینان خیلی خوبی ندارد. در این تمرین برای ایجاد وقفه‌های زمانی مورد نیاز، از تابع delay استفاده خواهیم کرد. تابع مورد استفاده تابع HAL_Delay است. البته عملکرد این تابع بر مبنای یک تایمر است پس دقت آن از توابع ساده که گفته شد بیشتر است. آرگومان ورودی تابع HAL_Dealy نیز مدت زمان ایجاد وقفه بر حسب میلی‌ثانیه را مشخص می‌کند. برای استفاده از این تابع درون کدها، بایستی این تابع را (با مقدار تأخیر مورد نیاز در آرگومان ورودی) بین دو دستور که می‌خواهید بین آن‌ها تأخیر ایجاد کنید، بنویسید.

از تابع delay برای حل ایرادات مطرح شده در بخش ۲ نیز می‌توانید کمک بگیرید.

نکات قابل توجه:

- در قسمت Clock Configuration، با استفاده از یک کریستال خارجی 8MHz کلاک هسته‌ی میکروکنترلر را روی 168MHz تنظیم کنید.

- نیازی به کشیدن شماتیک برای ماژول رطوبت‌سنج، کیپد، LCD و شیر آب نیست. بلکه اتصال این قطعات به برد، از طریق پین‌هدرهایی روی برد صورت خواهد گرفت.

- کدهای نوشته شده به صورت مرتب و ساختاریافته بوده و همچنین از گذاشتن کامنت نیز دریغ نکنید.

فایل‌های تحویلی شامل:

۱. پروژه‌ی Altium Designer حاوی فایل شماتیک برد طراحی شده؛

۲. پروژه‌ی نرم‌افزار Keil uVision؛

۳. گزارش‌کار در فرمت PDF؛ باشد.

موارد مورد نیاز در گزارش‌کار:

- ارائه‌ی راه‌حل برای مشکل مطرح شده در بخش ۲

- با توجه به کدی که نوشتید به این سوال پاسخ دهید که آیا هنگامی که شیر آب باز است، میکروکنترلر می‌تواند فشرده شدن کلیدی را در کیپد متوجه شود؟ توضیح دهید.

- به نظرتان ایراد عمده‌ی استفاده از توابع delay به غیر از دقیق نبودن چه چیزی می‌تواند باشد؟ برای پاسخ دادن به این سوال از عملکرد تابع HAL_Delay که در زیر نشان داده شده است، بهره بگیرید:

```
__weak void HAL_Delay(uint32_t Delay)
{
  uint32_t tickstart = HAL_GetTick();
  uint32_t wait = Delay;

  /* Add a freq to guarantee minimum wait */
  if (wait < HAL_MAX_DELAY)
  {
    wait += (uint32_t)(uwTickFreq);
  }

  while((HAL_GetTick() - tickstart) < wait)
  {
  }
}
```

در کد بالا تابع HAL_GetTick زمان نسبی بر حسب میلی‌ثانیه را به ما می‌دهد.

- بقیه‌ی گزارش‌کار شامل مواردی خواهد بود که نیاز به توضیح دارند. مثلا مقدار عددی I_{IO} ای که در بخش ۳ مطرح شد را بنویسید، یا مواردی مثل نحوه‌ی طراحی مدارات و انتخاب مقادیر المان‌ها یا ...

«موفق باشید»