CONTROL DE RIEGO

TRABAJO EN GRUPO MICROS
SISTEMAS ELECTRÓNICOS DIGITALES

Grupo 24:

Laura Álvarez Pinedo, 53811

Ignacio Dopazo López, 55218

Adrián Gómez García, 55269

Curso 2022-2023

Tutor: Rubén Núñez

Índice

Introducción

El proyecto elegido para implementar con el microcontrolador STM32F411 es un sistema de riego.

Funciones básicas del proyecto

Estas son las funciones básicas del proyecto que se prevé implementar, sin perjuicio de que se pudieran añadir algunas de las funcionalidades adicionales contempladas más abajo.

- Medir la humedad de una maceta para decidir si la planta necesita riego
- Programar un riego a una hora del día determinada
- Permitir un riego manual
- Medir el nivel de agua del depósito
- Uso de una bomba de agua para el riego
- Mostrar la información de interés en una pantalla LCD comunicada con la placa mediante protocolo I2C

Requisitos mínimos del proyecto

- **Utilización de entradas y salidas**: se emplea el pulsador de la placa, un display, un relé para el control de la bomba de agua y los sensores de humedad y nivel del agua.
- Se utilizan **interrupciones** asociadas a dos temporizadores y al pulsador.
- **Temporizadores**: se utilizan dos temporizadores que controlan el ritmo al que se refresca la pantalla, al que se guardan los datos de humedad para el cálculo de la media y también para el control del tiempo que permanece regando el sistema. En concreto son un temporizador básico para las dos primeras tareas indicadas y un temporizador de tipo "output compare" para el control del relé.
- Uso de **convertidor analógico digital** para la lectura de los sensores de humedad del suelo y de nivel del depósito de agua.

Otras características

- Uso de **DMA** para pasar los datos leídos de los sensores a la memoria y poder trabajar con ellos en el resto del programa.
- Uso del protocolo de **comunicación serie I2C** para la conexión de una pantalla LCD de 16 columnas y 2 filas.
- Uso de **sensores de humedad** del suelo / nivel de agua.
- Uso de un **relé** para el control de una regleta de enchufes de 220V AC a la que se conecta la bomba de agua.
- Uso de una **bomba de agua** alimentada con corriente alterna a 220V.
- Se ha construido la **maqueta** completa y se han grabado **videos** documentando el funcionamiento

Información complementaria

El desarrollo del proyecto se ha seguido en GitHub, por lo que toda la información de la memoria se encuentra de forma más extendida en la Wiki del proyecto.

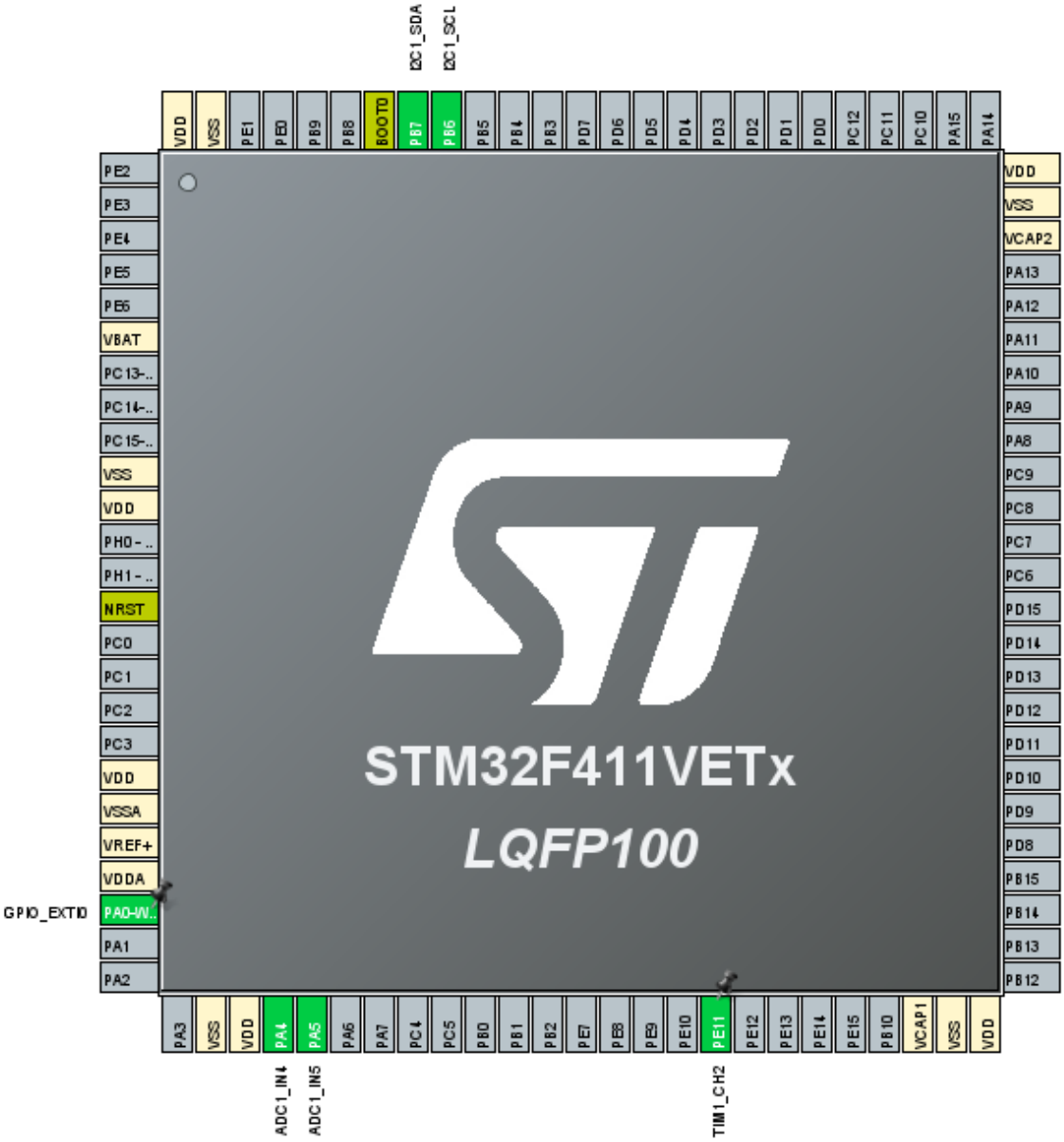
Además, se han grabado unos videos demostrativos del funcionamiento del proyecto con la placa STM32F411:

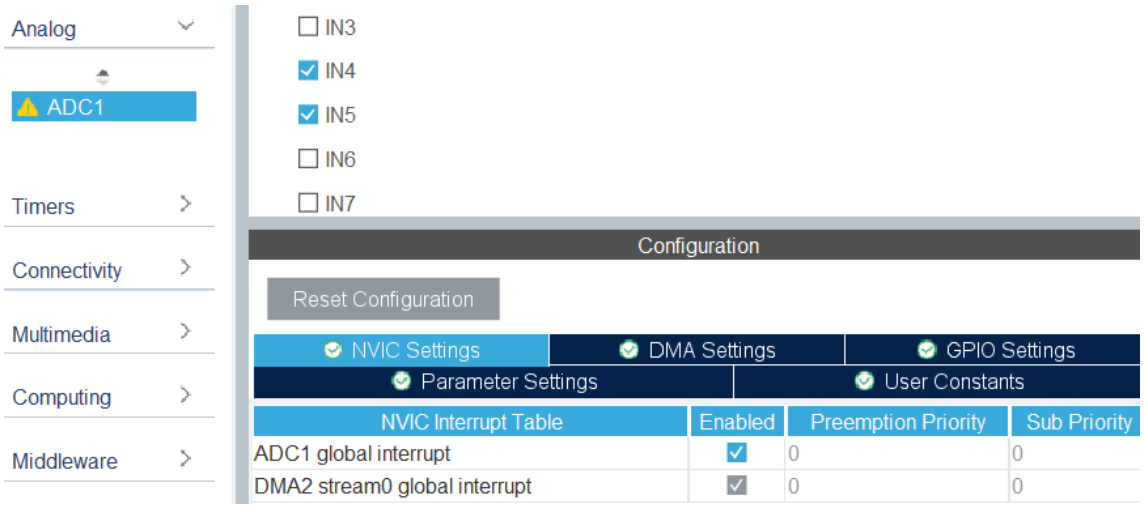
Parte 1:
https://upm365.sharepoint.com/:v:/s/SED_TRABAJO/EVpGtDIy2SpGo7Je-izxlO8BG9L7oC8CO0LWMe3WyJK6pw?e=x1DQGZ
Parte 2:
https://upm365.sharepoint.com/:v:/s/SED_TRABAJO/EdbJ4s1STZJNiIenHVVWurwBw67K3j3oW3H0REuOX2FHgQ?e=dU8LPI

Configuración del STM32CubeIDE

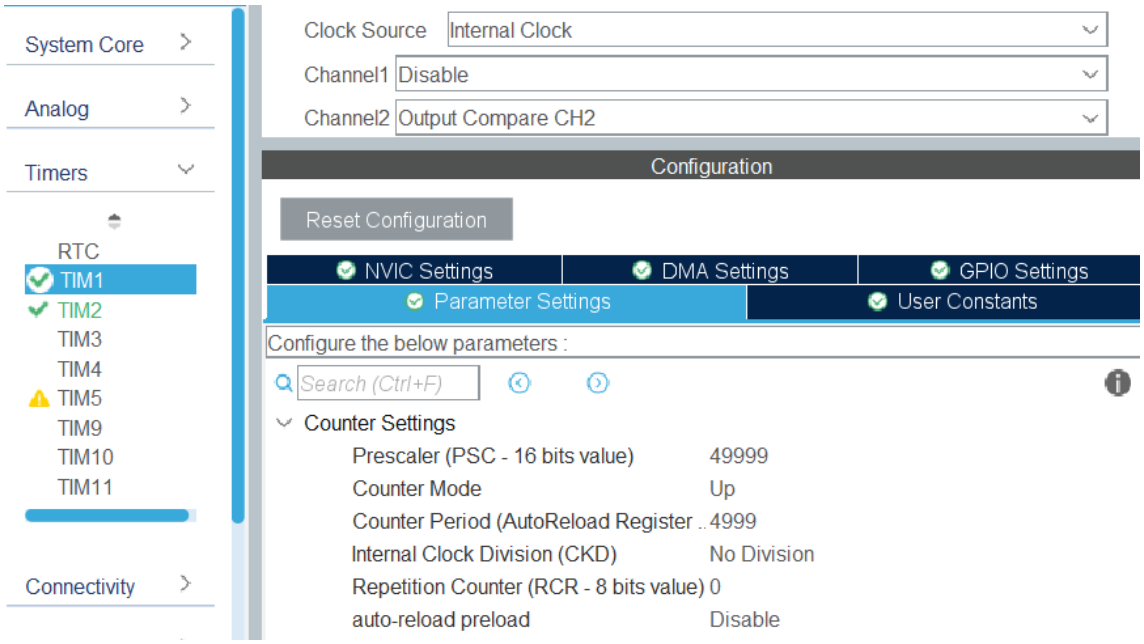
Lo primero a realizar es la configuración del STM32CubeIDE. Se necesitan:

- PA0 como GPIO_EXTI0: para el botón (con EXTI line0 interrupt enabled)
- PE11 como TIM1_CH2: para la bomba
- PA4 como ADC1_IN4: para el sensor encargado de medir el nivel del agua
- PA5 como ADC1_IN5: para el sensor encargado de medir la humedad de la tierra
- PB6 como I2C1_SCL: para la comunicación con la pantalla
- PB7 como I2C1_SDA: para la comunicación con la pantalla



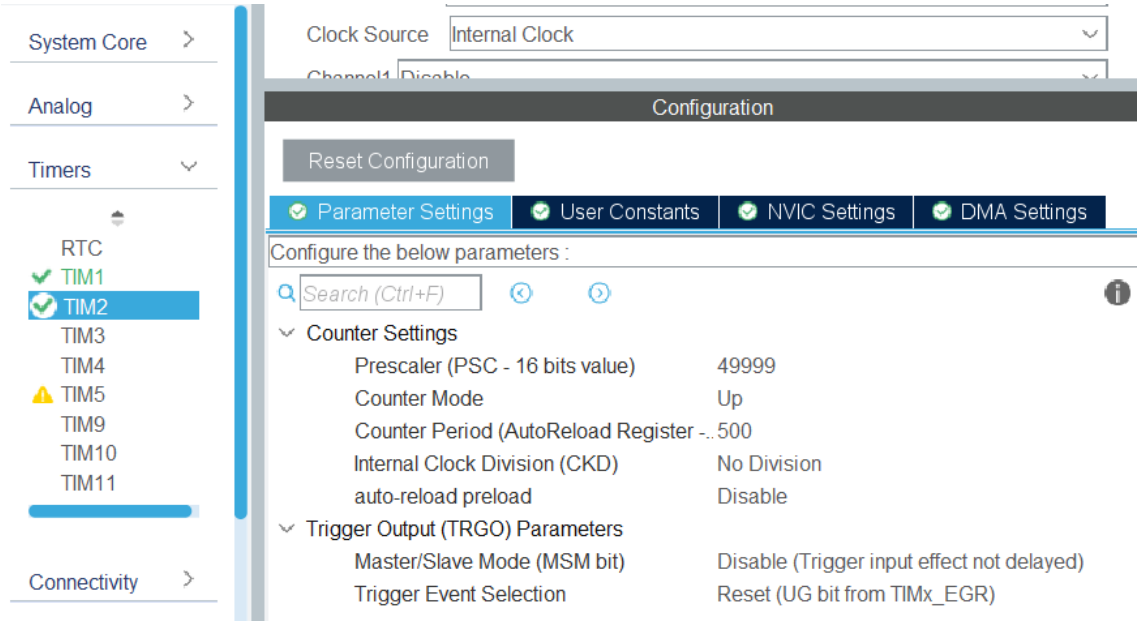


- ADC1 Global Interrupt enabled



Para el TIM 1 del tipo Output Compare se ha configurado, para un reloj de 50MHz, de forma que se obtiene un periodo de 5 segundos:

- Prescaler = 49999
- Counter period = 4999
- Counter mode = up
- Mode = Active Level on match
- Pulse = 0
- TIM1 capture compare interrupt enabled



Para el TIM 1 del tipo Básico se ha configurado:

- Prescaler = 49999
- Counter period = 499
- Counter mode = up
- TIM2 Global Interrupt enabled

Sensores

Se va a trabajar con dos sensores de humedad analógicos, uno situado en el depósito de agua para medir el nivel y el otro en la tierra para medir la humedad de esta.

Estos sensores miden la diferencia de resistencia entre los electrodos. Si están en tierra mojada o sumergidos dan el máximo valor posible porque está ocurriendo un cortocircuito. Así, cuanto más seca esté la tierra o cuanto menos sumergido esté el sensor, más resistencia habrá y por lo tanto el valor medido por el sensor será menor. En primer lugar, hay que realizar la conversión Analógico-Digital con la función:

```
void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef * hadc)
```

Al estar usando dos canales de conversión analógica digital se ha optado por utilizar DMA. De esta manera se consigue que el tratamiento de la conversión no ocupe tiempo de procesamiento del microprocesador. El DMA se configura de manera que se comuniquen los periféricos con la memoria, en este caso siendo los periféricos los dos sensores de humedad. En el código del programa se inicia la conversión DMA, indicando que dicha conversión se vuelque en un buffer, el cual ya será asignado una vez acabada la conversión en las variables correspondientes.

Así, posteriormente se puede detectar el nivel agua en el depósito: interesándonos si el valor medido por el sensor es 0 (no queda agua suficiente para el correcto funcionamiento de la bomba) o superior. Función: `detec_lvl`

Por otra parte, se calcula el porcentaje de humedad que detecta el sensor de la tierra, convirtiendo los valores de 0 a 4095 a porcentaje. Aunque para un valor menor de 1000 se ha establecido que el porcentaje de humedad que se mostrará será 0% por ajuste empírico. Función: `humedad`

Por último, los valores de humedad que se obtienen se van guardando y con ellos se calcula la media, de forma que ante algo de ruido que pueda modificar los valores de forma repentina, no salte el riego. Esto se ha realizado en la función del temporizador básico, para espaciar las medidas lo suficiente, puesto que el sistema trabaja a una velocidad muy alta. Función:

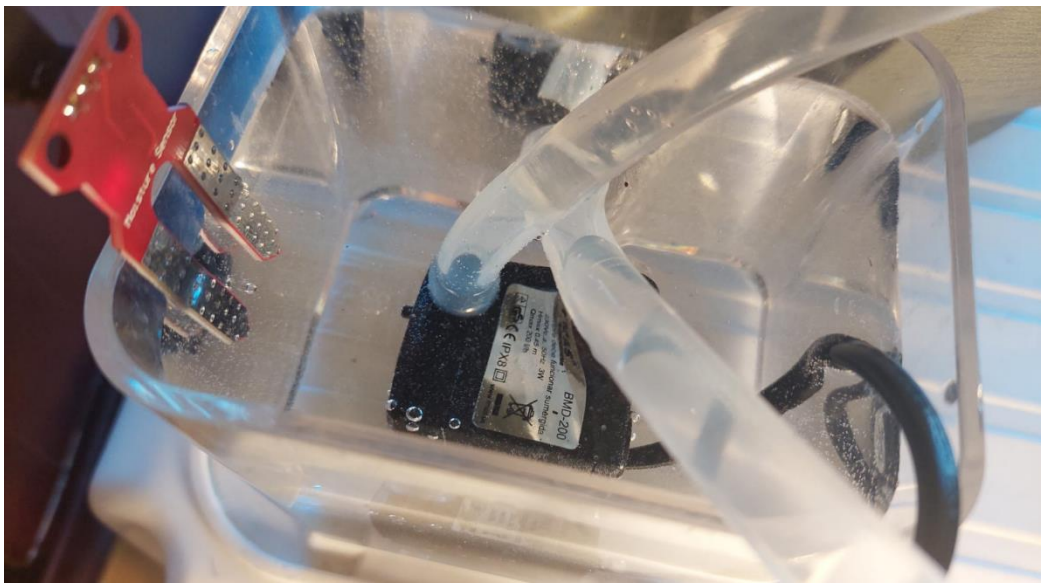
```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim
```

Pulsador

Además, también se tiene como entrada el botón de la placa, con el cual se puede activar y detener el riego, además de configurar el reloj y la hora de regado. Por lo que se trata en una función callback con antirrebotes, la cual activa o desactiva un flag y puede detener el temporizador de riego. Función:

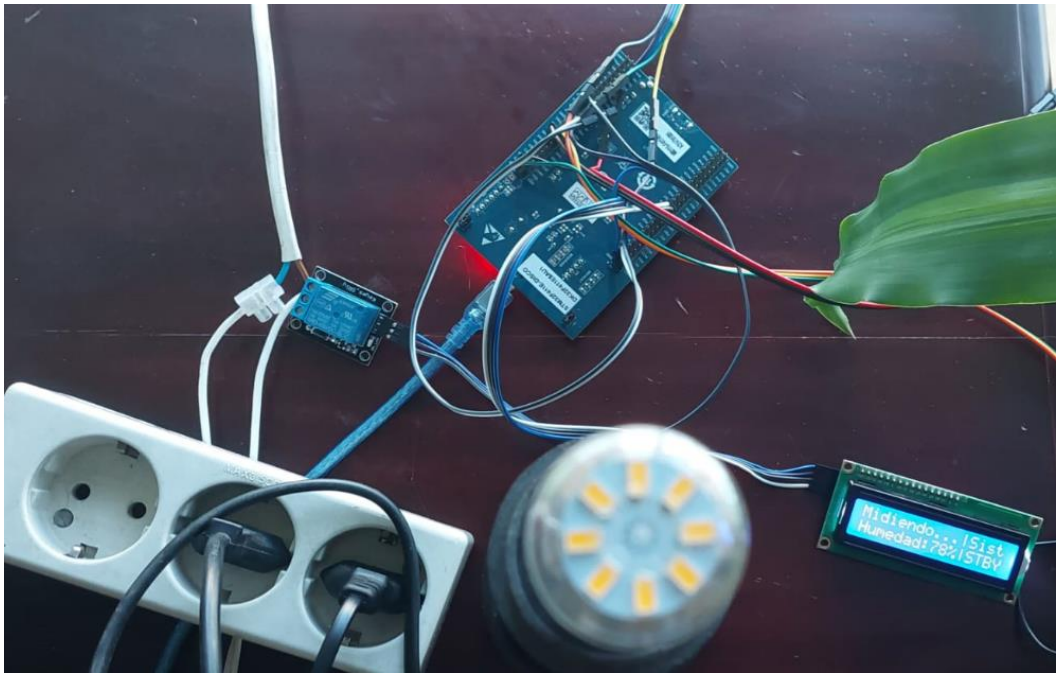
```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
```

Actuadores

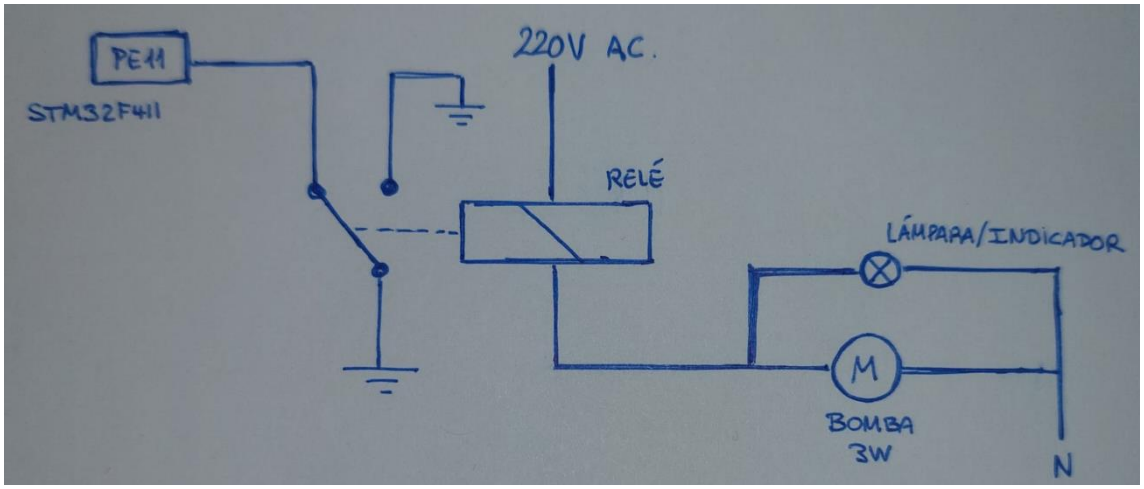


El objetivo es hacer funcionar una bomba de agua. En cualquier caso, las salidas de la placa nunca serían suficiente para hacer funcionar una bomba por pequeña que fuese. De hecho, inicialmente se estaba contemplando el uso de una bomba de 3V DC, pero finalmente se ha utilizado una bomba de 220V AC de 3W.

Para controlar la bomba se ha empleado un relé (en la parte izquierda de la imagen de abajo) conectado a la salida del microcontrolador que conecta y desconecta una regleta de la red. A dicha regleta se conecta la bomba y, como se puede apreciar en las imágenes, una lámpara que permite ver con mayor claridad cuando la bomba está funcionando (esto no sería parte del montaje final, se trata de un indicador para las demostraciones de funcionamiento).



Esquema eléctrico



Funcionamiento

Para activar el riego se ha establecido la siguiente condición:

- Que la humedad registrada sea menor del 50%, que se haya pulsado el botón o que sea la hora de regar
- Y además, que flag_riego = 0 y que hayan pasado más de 5 segundos desde la última vez que se regó (haciendo uso del HAL_GetTick()) Si se cumplen estas condiciones se desactiva el flag del botón y si hay suficiente agua en el depósito: comienza a regar e inicia el temporizador.

Para la detención del riego, las condiciones establecidas son:

- Que se quede sin agua (en el `while` de `main`), que se haya pulsado el botón mientras se regaba (`void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin`)) o que se acabe el tiempo (`void HAL_TIM_OC_DelayElapsedCallback (TIM_HandleTypeDef* htim)`) En ese momento se desactiva el flag de riego y se detiene el temporizador

Pantalla

En esta página se describe el funcionamiento de la pantalla que muestra la información al usuario.

Sobre la pantalla

La pantalla utilizada es una pantalla LCD de 16 columnas y 2 filas controlada por I2C. Estas pantallas disponen de un módulo adosado a la parte trasera que simplifica las comunicaciones con la pantalla al poder manejarla por I2C. Dicho módulo dispone de un potenciómetro para regular el contraste de las letras y de dos pines que permiten encender la retroiluminación de la pantalla.

La conexión de la pantalla a la placa por lo tanto es como la de la mayoría de los dispositivos I2C, 2 cables de alimentación (5V y GND) y dos cables para la comunicación (SCL y SDA). Esta pantalla cuenta con la posibilidad de configurar 8 direcciones distintas como se explica más adelante.

Configuración en el STM32CubeIDE

Se habilitan las comunicaciones por el puerto I2C1. No es preciso realizar ningún otro ajuste. La velocidad por defecto es adecuada y no se emplean interrupciones puesto que solo se requiere transmitir información y en ningún momento es necesario leer información del dispositivo. Los tiempos de transmisión son lo suficientemente bajos como para que no sea necesario el uso de interrupciones.

Los pines SCL y SDA se indican en el IDE. Los correspondientes al puerto I2C1 son el pin PB7 para SDA y PB6 para SCL.

Programando la pantalla por I2C

Introducción

En primer lugar, es necesario definir la dirección de la pantalla. El módulo para la comunicación I2C dispone de tres pines que se pueden conectar o no para modificar la dirección de la pantalla y así poder conectar hasta 8 pantallas a una misma línea I2C. En este caso se dejarán los pines sin conectar, por lo que la dirección será la `0x4E`, que es la dirección por defecto.

Para la comunicación se envían comandos y datos de forma diferenciada.

Los comandos son instrucciones para la pantalla del tipo borrar, mover el cursor... mientras que los datos son los caracteres que se quieren escribir en la pantalla.

A partir de estas funciones se pueden programar otras funciones más complejas que, por ejemplo, además de enviar el texto necesario, envíen comandos para posicionarlo en la pantalla correctamente.

Actualizando la pantalla

Como actualizar la pantalla requiere un cierto tiempo, es una tarea que no se hace a lo largo del programa: se hace en un momento puntual cada cierto tiempo. Un temporizador genera interrupciones que hacen que se refresque la pantalla con el contenido que se ha ido dejando en la cadena de texto correspondiente.

El ajuste de la frecuencia a la que se refresca la pantalla se ha hecho de forma empírica, comprobando qué valor se adaptaba mejor a las necesidades del proyecto. Se ha determinado que el tiempo óptimo está entre medio segundo y un segundo. Refrescar la pantalla demasiado rápido dificulta la lectura ya que prácticamente no da tiempo a mostrar el mensaje. Un tiempo excesivo haría que no se actualicen los datos con la suficiente rapidez.

La configuración del temporizador por lo tanto es la siguiente:

- Frecuencia de reloj: `50MHz`
- Prescaler: `49999`
- Period: `500`

El tiempo por lo tanto es de 0,5 segundos (2Hz): frecuencia = reloj/ (prescaler + 1) * (period + 1) = 50.000.000/(50.000 * 501) = 2 (Aproximadamente, para exactitud total el periodo debería ser 499).

El temporizador que se ha empleado es el TIM2, de tipo básico con interrupciones. Es en el callback donde se llama a la función lcd_update. La cadena de texto enviada ya ha sido preparada previamente en la función main del programa. Este temporizador se ha reutilizado para el cálculo de la media de humedad de los últimos 5 segundos.

Código en main

El código en main solo se tiene que encargar de introducir la información necesaria en la cadena de texto `texto_pantalla` declarada como variable global para que el callback del temporizador que actualiza la pantalla tenga acceso posteriormente.

Para guardar información en dicha cadena de texto se emplean las funciones de la string library `string.h`. Las funciones utilizadas son en particular `strcpy` para la copia de una cadena de texto en la cadena de texto `texto_pantalla` y la función `strcat` para concatenar posteriormente la primera función con otros fragmentos.

Para la representación en la pantalla se requiere que toda la información esté almacenada como cadenas de texto, por lo que el valor de la humedad (almacenada en la variable global media) se convierte a un string mediante la función `sprintf` de la biblioteca `stdio`. Dicho string se concatena mediante la mencionada función `strcat`. Adicionalmente, para que siempre ocupe el mismo espacio en la pantalla se añade un cero a la izquierda a los valores de un dígito. En el caso del 100, por cuestiones de espacio, se omite el "%".

Las operaciones anteriores son iguales para cualquier estado en el que se encuentre el sistema. El resto de la información mostrada en la pantalla dependerá del estado en el que se encuentre el sistema. A continuación, se muestra la parte de código común a todos los casos y lo que se muestra cuando el sistema está midiendo y listo para regar cuando se requiera (standby):

```
 if (media >= 10)
          sprintf(buffer, "%i", media);
  else if (media==100)
          sprintf(buffer, "%i", media);
  else
          sprintf(buffer, "0%i", media);

  if (!flag_riego && flag_agua != 0){
// si no esta regando y hay agua en el deposito = midiendo
          strcpy(texto_pantalla, "Midiendo...|SistHumedad:");
      strcat(texto_pantalla, buffer);
      if(media!=100)
      strcat(texto_pantalla, "%");
      strcat(texto_pantalla, "|STBY");
     }
```

Funciones

Inicialización

```
lcd_init();
```

Realiza la configuración inicial de la pantalla siguiendo las órdenes que se especifican en el datasheet del módulo de comunicación de la pantalla (un enlace a una web con dicho datasheet se puede encontrar más abajo en la bibliografía).

Primero se envía la orden de encendido del display de mostrar el cursor repetidas veces. Posteriormente se configura la pantalla para recibir la información en 4 bits. Más tarde se configuran aspectos como dónde debe escribirse, si el cursor debe parpadear o se borra el contenido de la pantalla entre otras cosas.

Cabe destacar que la función utiliza numerosas veces HAL_Delay. Esto se debe a que se especifica en el datasheet que debe esperarse tras enviar algunos comandos. Los tiempos que se han empleado en este proyecto están más ajustados que los que se encuentran en algunas de las referencias bibliográficas indicadas, pero son necesarios y no se pueden eliminar o reducir mucho más. Se emplea HAL_Delay y no otras alternativas como el uso de HAL_GetTick porqeu es más sencillo y en esta parte del programa no supone ningún inconveniente, puesto que es el arranque y todavía no deberían producirse interrupciones de ninguna clase. Si se produjesen, tampoco sería especialmente problemático ni que la función init quede a medias ni que se tarde unas milésimas de segundo en atender la interrupción. En su conjunto, la función init se ejecuta en bastante menos de un segundo.

Enviar un comando

```
lcd_send_cmd(char cmd);
```

Se envía un comando a la pantalla. Esta función es ligeramente diferente de lcd_send_data porque los datos se enmascaran de forma diferente, habilitando distintos bits. Sin embargo, en líneas generales, el procedimiento es el mismo en ambos casos: preparar la información que se va a enviar en un buffer y posteriormente transmitirla con `HAL_I2C_Master_Transmit`.

Enviar datos

```
lcd_send_data(char data);
```

Esta instrucción envía una letra a la pantalla. La pantalla se encarga de generar el carácter que se verá en la pantalla.

Representar una cadena de texto

```
lcd_send_string(char *str);
```

Esta instrucción llama repetidas veces a send_data, una por cada carácter que se desea enviar. El módulo de la pantalla automáticamente va desplazando el cursor una vez a la derecha por cada carácter que representa. El problema vendrá con el salto de línea. Cuando se quiere comenzar a escribir en cualquier a de las dos líneas hay que enviar un comando para que el cursor se ubique en la línea correspondiente. Para ello hay que llevar la cuenta del número de caracteres que se han representado para saber cuándo enviar ese comando.

El segundo problema aparece con esa cuenta: cada vez que se llame a la función el valor de la variable no se conservaría. Por ello se emplea una variable de tipo `static` que mantiene la cuenta en todo momento. Así, cuando se enviase el mensaje "Bienvenido al" y posteriormente "sistema de riego", en la pantalla se escribiría en la primera línea "Bienvenido al si" y después "tema de riego" en la segunda línea, sin importar que las cadenas de texto se hayan pasado en momentos diferentes.

```
      while (*str){
        lcd_send_data (*str++);
        contador_linea = (contador_linea) % (ROWS*LINES);
//Contar hasta el número de caracteres máximo
        contador_linea++;
        if (contador_linea == ROWS)
                lcd_send_cmd(0xc0); //Pasa a la segunda línea
        if (contador_linea > ROWS*LINES)
                return; //No hace nada más porque no hay espacio
```

Actualización de la información en pantalla

```
lcd_update(char *str);
```

Esta función llama a lcd_send_string y le pasa la cadena de texto que ha recibido como parámetro. La diferencia radica en que se ejecuta previamente la función lcd_clear, que borra la pantalla. Aunque ambas funciones realicen tareas muy similares, lcd_send_string permitiría ir enviando pequeñas cadenas de texto y que se vayan representando unas detrás de otras en la pantalla mientras que en el caso de lcd_update, solo se puede escribir toda la pantalla de golpe.

La función empleada en este proyecto es esta, lcd_update, pero implementarlo de esta forma, el código sería más reutilizable ya que en otros proyectos puede interesar tener la posibilidad de escribir varios mensajes sin borrar la pantalla.

Borrado de la pantalla

```
lcd_clear();
```

Es una función que mediante un bucle escribe espacios en cada una de las posiciones de la pantalla. (Para cambiar de línea debe emplear el comando de salto de línea).

Sobre la reutilización de código

El código en líneas generales está pensado para poderse reutilizar, pero existe un gran problema para ello: las direcciones de las posiciones de cada carácter de la pantalla varían si se cambia el hardware. Esto obliga a que, si se utiliza una pantalla de distinto tipo, por ejemplo, una LCD de 16x4 en lugar de una de 16x2 como se ha utilizado en este proyecto, habría que añadir las instrucciones que permiten el salto a las siguientes líneas.

Mientras la pantalla que se utilice sea una LCD 16x2 no debería haber ningún problema a la hora de reutilizar el código.

Muchas funciones están basadas en información de las siguientes páginas, aunque han sido modificadas, adaptadas y en algunos casos escritas "de cero" para satisfacer los requisitos de este proyecto (Tech, s.f.) (AllDataSheet, s.f.) (Husamuldeen, s.f.).

Configuración de la hora

Inicialización

Al iniciar el programa aparece en la pantalla el mensaje de Bienvenido al sistema de riego durante 1 segundo.

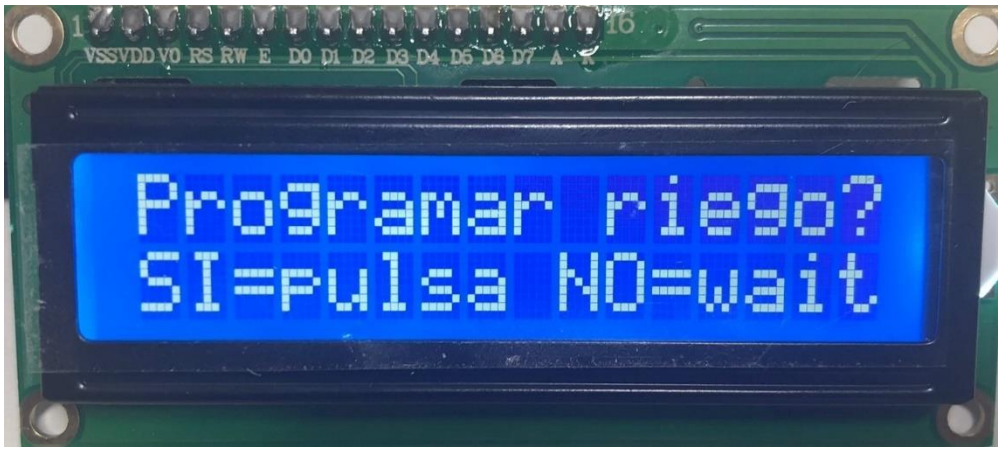
A continuación, te da la opción de poner la hora actual (si no la pones tomará como que son las 00:00):

*NOTA: trabajar despacio puesto que la velocidad de actualización de la pantalla puede ser lenta en comparación con la velocidad a la que se puede pulsar el botón.

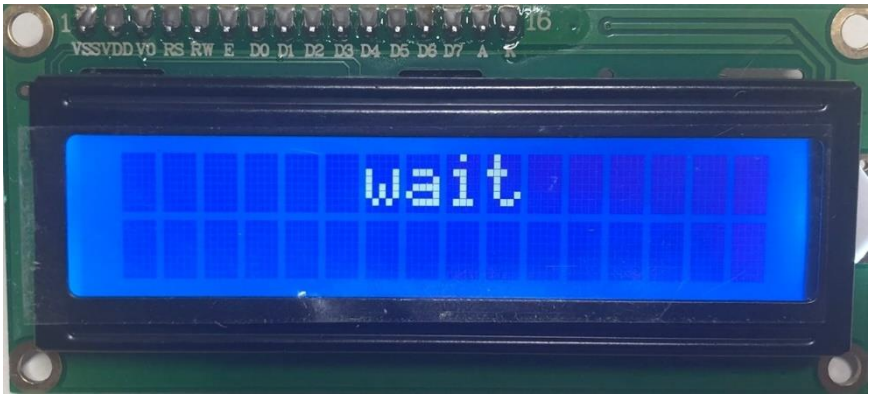
- Primero te permite ajustar la hora (va aumentando en uno cada vez que se pulsa el botón y si se llega a 23 vuelve a 00). Si se está más de 5 segundos sin tocar el botón se pasa automáticamente al siguiente paso.

- Tras los 5 segundos de espera te permite ajustar los minutos. Igual que antes, va aumentando en uno cada vez que se pulsa el botón y cuando se está más de 5 segundos sin tocar el botón se pasa automáticamente al siguiente paso.

En este momento aparece la opción de programar una hora de regado:



- Si no se quiere establecer ninguna hay que esperar y luego saldrá este mensaje y comenzará el funcionamiento normal del programa:



Esto es posible ya que la variable correspondiente a la hora de regado se ha inicializado a 100, de forma que si no se toca nunca se va a alcanzar esa hora y por tanto no va a saltar el riego.

- En cambio, si se pulsa el botón quiere decir que se quiere configurar una hora para que se riegue cada día, por lo que aparece una pantalla similar a la de configuración de la hora actual y con un funcionamiento similar: pulsando el botón se ajusta la hora, se deja de tocar el botón durante 5 segundos, ahora al pulsar el botón se ajustan los minutos y de nuevo se esperan 5 segundos sin pulsar. Al terminar comenzará el funcionamiento normal del programa.

Código

Esto se ha conseguido primero situando dentro del main y antes del while unos contadores y unas configuraciones de pantalla como las mostradas a continuación:

```
counter = HAL_GetTick();

while (!OK_hour){ // Set hora
  if (flag_boton){ //Se incrementa 1 hora cada vez que se pulsa el boton
                if (intro_hour < 23){
                        intro_hour++;
                } else {
                        intro_hour = 0; //Vuelta a 0
                }
                flag_boton = 0;
                counter = HAL_GetTick();
        }
        if (HAL_GetTick() - counter > 5000){
                OK_hour = 1;
//Si no se pulsa en 5segundos, se da por terminado el cambio de la hora
        }

  //Mensaje en la pantalla
  if (intro_hour >= 10)
          sprintf(time_h, "%i", intro_hour);
  else
          sprintf(time_h, "0%i", intro_hour);

        strcpy (texto_pantalla, "Actual (hora):  ");
        strcat(texto_pantalla, time_h);
        strcat(texto_pantalla, ":00           ");
```

Por otra parte, ya dentro del while se ha programado el conteo del tiempo para crear un reloj a partir de la hora actual indicada, a partir de HAL_GetTick():

```
//Conteo de tiempo

  count_second = ((HAL_GetTick()-counter) * 0.001);
  count_minute = ((HAL_GetTick()-counter) / 60000);
  count_hour = ((HAL_GetTick()-counter) / 3600000);

//Hora real

  second = count_second % 60;
  minute = (intro_minute + count_minute) % 60;
  hour = (intro_hour + count_hour + ((intro_minute + count_minute) /60)) % 24;
```

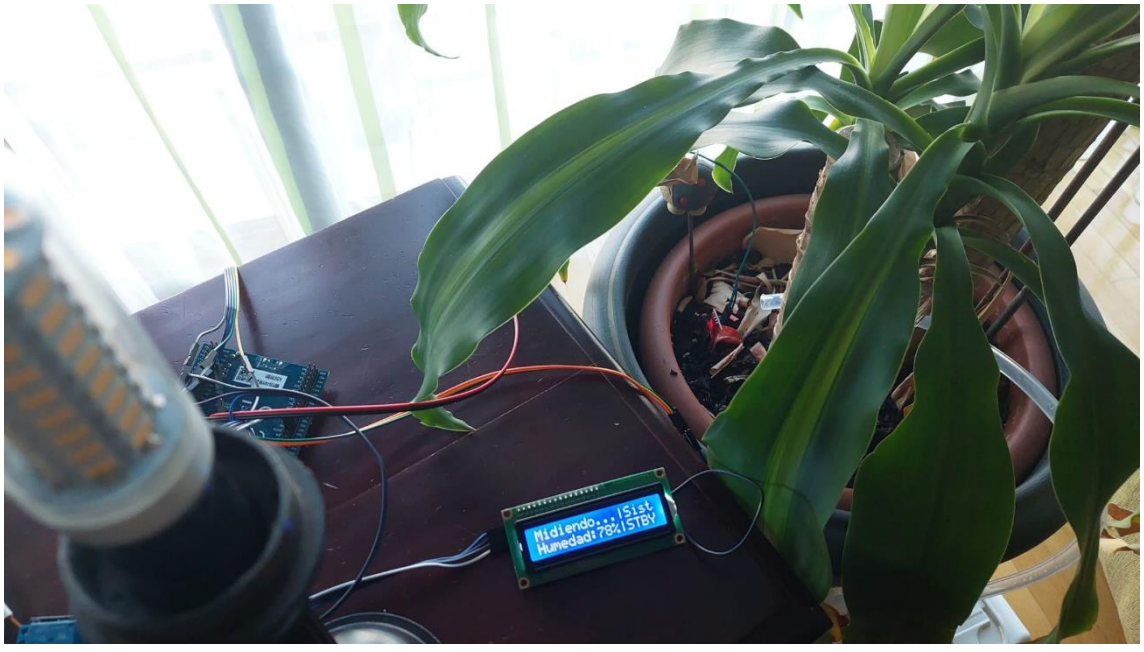
Pruebas de funcionamiento del trabajo de microcontroladores

Cuando se inicia el sistema se muestra un mensaje en la pantalla dando la bienvenida y un segundo más tarde pide que si quieres introduzcas la hora actual y la hora de regado. Luego comienza el bucle de ejecución del programa. Existen cuatro estados en los que puede estar el sistema que se repasan a continuación. Hay que tener en cuenta en línea generales que debe tenerse la precaución de que la bomba solo funcione con agua. Para ello, como ya se ha comentado, hay un sensor que detecta cuando el nivel de agua queda por debajo de la bomba de agua. El sensor está programado de forma que se indican 4 niveles de agua (0-25%, 25-50%, 50-75% y 75-100%). Debe tenerse en cuenta que ni la precisión ni la sensibilidad de los sensores empleados es muy elevada cuando se trata de medir el nivel de un líquido.

*NOTA: aunque se puede llegar a apreciar como sale agua del tubo cuando la bomba está activa, se ha conectado en la misma regleta de enchufes que la bomba de agua una lámpara, que se ve en algunas imágenes si está encendida o apagada, de modo que se aprecia con mayor claridad cuando el sistema está funcionando y cuando no. (La placa controla un relé que está conectado en el cable de alimentación de la regleta, encendiendo o apagando los tres enchufes de la misma).

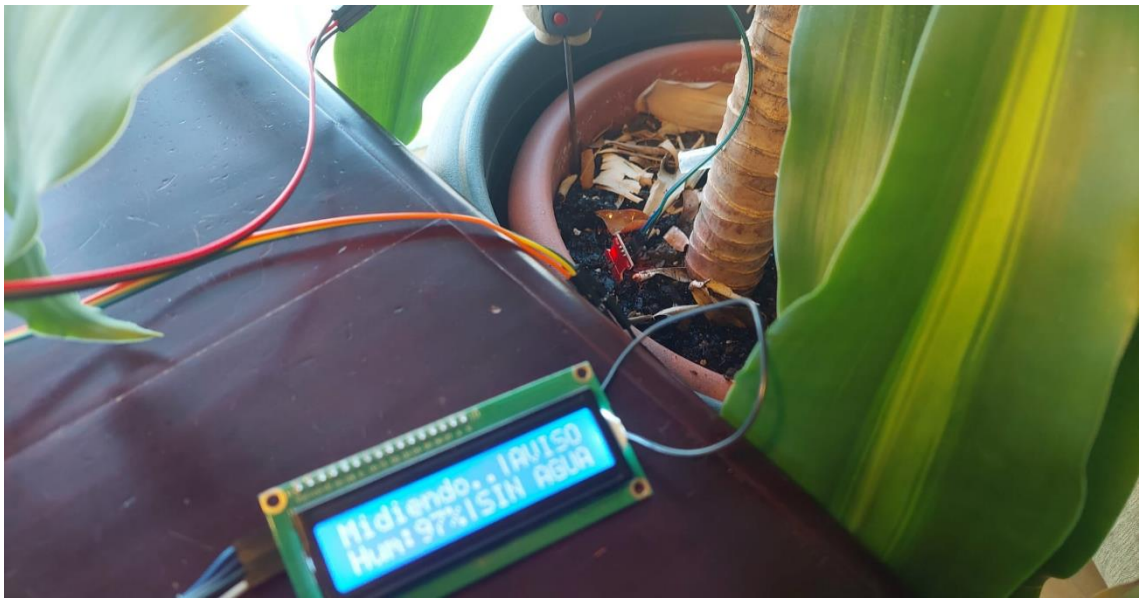
Humedad superior al umbral y hay agua

Cuando el sensor detecta que no es necesario regar y que además hay agua por si hiciese falta regar, en la pantalla se muestra el nivel de humedad detectado y un mensaje que informa de que se está midiendo. Además, aparecerá un mensaje en la parte derecha de la pantalla en el que pone que el sistema se encuentra en standby, listo para funcionar en cuanto se requiera ("SIST. STBY").



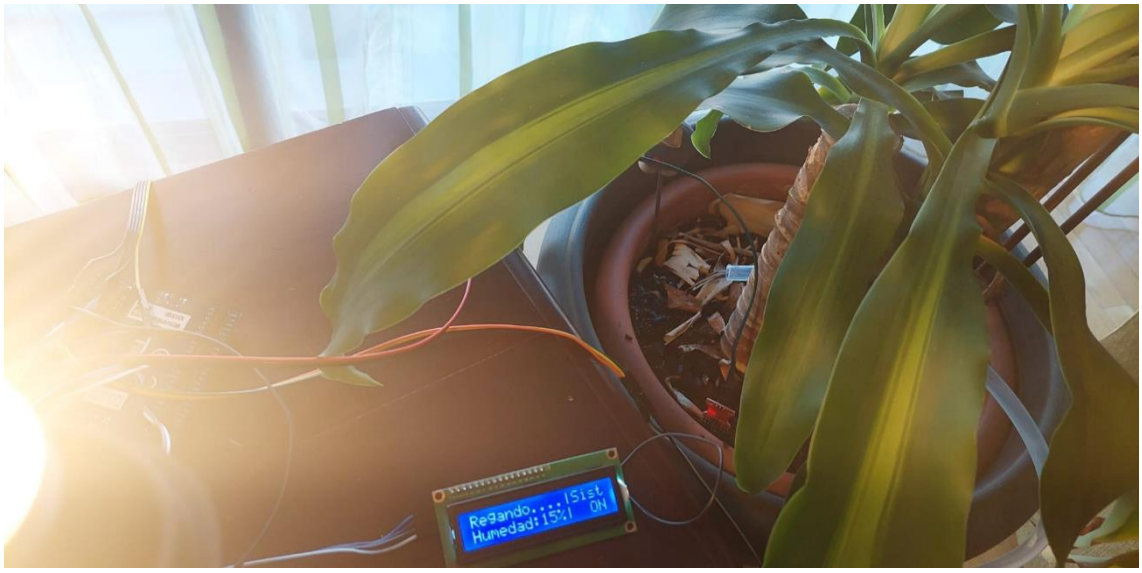
Humedad superior al umbral y no hay agua

Se indica que, aunque no es necesario regar, habría que rellenar el depósito porque no queda suficiente agua para regar la próxima vez. Se indicará que es solo un aviso y no una alerta, que se reserva para cuando realmente es necesario regar y no hay agua para ello, como se verá a continuación. Se continúa indicando el nivel de humedad de la maceta.



Humedad por debajo del umbral y hay suficiente agua para regar

Se indica en la pantalla que se está regando y en lugar de SIST STBY (sistema en standby) aparecerá SIST ON (sistema encendido). Se continúa indicando en la pantalla el nivel de humedad de la maceta mientras tanto.



Humedad por debajo del umbral y no hay agua en el depósito

Se muestra en la pantalla una alerta y la humedad de la maceta. Al indicar que hay una alerta hace referencia a que se debería regar la maceta pero que no hay agua para ello. En cuanto se rellene el depósito se comenzará a regar inmediatamente.

Otros aspectos

- Con el temporizador TIM1 se ha configurado que el tiempo de regado sea de 5 segundos para la prueba del trabajo, aunque en la vida real se debería dejar más tiempo. De esta forma, una vez se activa la salida, si no se pulsa el botón antes, esta se desactiva a los 5 segundos, dejando de regar. Función: `void HAL_TIM_OC_DelayElapsedCallback (TIM_HandleTypeDef* htim)`
- En cualquier momento, cuando se esté regando, se puede cancelar la operación manualmente.
- Desde que se termine de regar por cualquier motivo (petición manual, fin del temporizador, depósito sin agua) hay que esperar un tiempo hasta que se puede volver a regar (establecido en 5 segundos actualmente). De esta forma da tiempo a que el sensor detecte el agua que ha llegado a la maceta (puede tardar unos segundos en subir la lectura, especialmente si se tiene en cuenta que se calcula con la media de los últimos 5 segundos). También podría darse la circunstancia de que el usuario haya querido detener el sistema porque se haya soltado un tubo y se esté saliendo el agua, por lo que hay que darle tiempo para corregir la situación. En las pruebas, por comodidad y cuestiones de tiempo, se ha establecido el tiempo en 5 segundos, pero en la práctica debería ser un tiempo superior. Para realizar esto se hace uso de `HAL_GetTick()`.
- Cuando se detecte que no hay suficiente agua, el riego se interrumpe automáticamente en el momento, aunque no haya transcurrido el tiempo establecido, para garantizar que la bomba funciona siempre sumergida.

Bibliografía

AllDataSheet. (s.f.). *HD44780 Datasheet (PDF) - Hitachi Semiconductor*. Obtenido de https://pdf1.alldatasheet.com/datasheet-pdf/view/63673/HITACHI/HD44780.html

Husamuldeen. (s.f.). *Working with STM32 and Liquid Crystal display: I2C mode*. Obtenido de https://embeddedexpert.io/?p=655

Tech, C. (s.f.). *Interface LCD 16×2 via I2C with STM32*. Obtenido de https://controllerstech.com/i2c-lcd-in-stm32/