SISTEMAS ELECTRÓNICOS PARA AUTOMATIZACIÓN

MEMORIA PROYECTO MICROCONTROLADORES: Ihuerting

Rafael Jiménez Bravo

Alberto González Isorna

Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 15 febrero 2018

ÍNDICE

1. Introducción: Ihuerting

¿Se imagina un mundo en el que, al irse de vacaciones, no tuviera que incordiar a su cuñado, a su vecina o al hijo de su prima para que anden entrando a tu casa a que te rieguen las plantas? ¿Y si tuvieras la posibilidad de controlar el agua que reciben desde tu propio ordenador?

En líneas generales, sabemos que un huerto es una plantación destinada al cultivo de vegetales, hierbas y hortalizas de diferente tipo; y que, según el tamaño, el tipo de cultivos y el sistema de riego, un huerto puede ser muy diferente a otro. Pero si algo tienen todos en común, es que necesitan ser regados, de una forma u otra.

Por otro lado, un sistema domótico es un sistema que también puede ser llamado "inteligente", capaz de automatizar algo con todos los servicios de los que dispone para su máximo aprovechamiento.

Si juntamos ambas partes obtenemos un huerto domótico, un sistema para el control automático de un huerto que gestiona los recursos de la manera más eficiente posible.

¿Qué beneficios implica el desarrollo de un huerto domótico?

Hoy en día y en los tiempos que corren cada vez se va haciendo más necesario el ser autosuficientes en ciertos aspectos, y con esta idea pretendo llegar a conseguir parte de ese autoabastecimiento, puesto que podremos tener a nuestro alcance la disposición de un huerto con determinados alimentos destinados al consumo diario, y sin apenas requerir cuidados. Esto también nos aportará mejoras para nuestra salud, ya que conseguiremos alimentos 100% naturales, libres de ciertos productos químicos que habitualmente se encuentran en los productos del mercado, y que en la mayoría de los casos no tenemos constancia de su uso.

Además de un beneficio personal, obtenemos a su vez un ahorro de agua, que tan necesario es en la actualidad y que nos beneficia a toda la sociedad con la preservación de nuestro medioambiente.

2. Riego automático

El riego automático consiste en que una parcela de terreno en la que hay una plantación se riega de forma autónoma, teniendo en cuenta varios factores como la humedad, la presión, temperatura, cantidad de agua necesaria, etc. El fin último sería la automatización completa durante todo el proceso de crecimiento y maduración de la planta, de forma que sólo nos tenemos que preocupar de plantarlo en el lugar adecuado y con el abono adecuado y esperar a que la planta alcance su estado de recogida.

Para llevar a cabo el riego automático, nos hace faltan tres ingredientes fundamentales: una correcta temporización (frecuencia de riego), una bomba que nos facilite el agua necesaria (caudal de riego) para la planta elegida y un sistema automático que gestione los dos anteriores (control de riego).

a. Frecuencia de riego: depende fundamentalmente de la capacidad de almacenamiento del suelo (capacidad del estanque) y de la tasa de evapotranspiración[1] del cultivo. En concreto la frecuencia o el número de días entre riegos se calcula como:

$$\boldsymbol{Nd = \frac{HA(50\%)}{ET_{prom}}}$$

HA = Humedad aprovechable (cm)
ET = evapotranspiración promedio (cm/día)

Por otra parte, la humedad aprovechable y la ET podemos medirla y predecirla con los sensores del terreno y los datos tabulados según el cultivo, la superficie, la estación y el tipo de terreno. En nuestro proyecto, por simplicidad ya que no es objeto de la asignatura, hemos decidido escoger un riego de periodicidad diaria.

b. Caudal de riego: El cuánto aplicar se relaciona con el tiempo de riego o el número de horas en que el agua debe escurrir sobre el suelo para que infiltre y moje la zona de las raíces de las plantas. En nuestro caso, hemos empleado la información de sensores y del usuario para ajustar el riego diario.

c. Control de riego: tras el cálculo del caudal y la frecuencia de riego, mediante un reloj de tiempo real (RTC) comparamos la hora actual con la hora a la que debemos regar. Cuando coincide, regamos el tiempo necesario. Se aplican además factores de corrección según la luminosidad y la humedad relativa.

[1] La evapotranspiración se define como la pérdida de humedad de una superficie por evaporación directa junto con la pérdida de agua por transpiración de la vegetación

¿Cómo se ha implementado este riego automático en nuestro sistema?

Para implementar este modo automático es necesario llevar a cabo una integración de los distintos sensores. En primer lugar, se debe de conocer la medida proporcionada por el RTC. Esta medida se compara con una constante de tiempo vinculada a cada planta, de forma que cuando el tiempo de RTC coincida con este, se producirá el proceso de riego.

No obstante, esta constante de tiempo de riego no es un valor fijo, sino que a un determinado valor teórico se le aplican una serie de correcciones, en función de la luminosidad, de la temperatura, y de la humedad.

¿Cómo se automatiza el toldo en el modo automático?

Cada planta requiere un numero determinado de horas de sol, de forma que un exceso de horas de incidencia de este podría generar un empeoramiento del fruto cultivado. Para solucionar este problema, el toldo, accionado mediante un paso a paso, se extiende, impidiendo que los rayos de sol incidan sobre el cultivo, permitiendo de esta forma obtener un producto con unas características óptimas. La implementación del modo automático en este toldo es sencilla: Se almacena en una variable la intensidad luminosa captada por los sensores en un determinado periodo de tiempo, de forma que en esa variable se va acumulando las medidas obtenidas, extendiendo el toldo cuando se llega al valor límite de la planta plantada.

3. Estructura y diseño

Para comprobar el comportamiento de nuestro sistema se ha llevado a cabo la realización de una simulación a pequeña escala, donde podemos distinguir los siguientes elementos:

En primer lugar, una maqueta de un huerto, basado recipiente con tierra y plantas plantadas.

Otro elemento que destacar es el toldo, el cual permite controlar las horas del sol que recibe el cultivo. Su apertura y cierre se controla mediante un motor paso a paso.

El depósito es otro de los componentes, donde la extracción de agua se lleva a cabo mediante una bomba activada por un relé.

Por último, la pantalla táctil desde donde se controla la automatización del huerto.

Desde el punto de vista del software, se ha desarrollado un esquema basado en una integración de diversos componentes. En primer lugar, se ha empleado el sensors Boosterpack, desde el cual obtenemos las medidas de los sensores TMP007(temperatura), OPT3001(luminosidad), y el BME280(sensor ambiental). La información de estos sensores se combina con la proporcionada por el RTC y el sensor de nivel, y es enviada al microcontrolador TM4C1294NCPDT, donde se lleva a cabo el control del sistema.

Por último, hay que añadir al esquema descrito anteriormente la participación de la pantalla VM800, que servirá tanto para mostrar información como para tomar decisiones.

El esquema descrito se puede visualizar en la siguiente imagen:

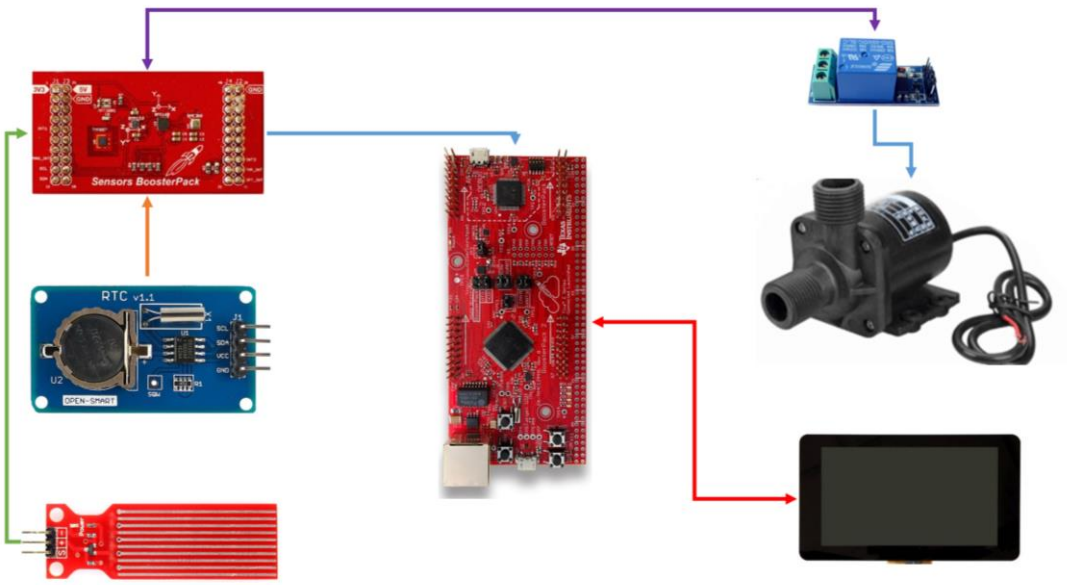

Ilustración 1: Esquema del proyecto

4. Desarrollo del proyecto

El procedimiento de este presente proyecto ha sido comprender, manejar e implementar cada uno de los componentes necesarios para el huerto automático para realizar posteriormente una implementación conjunta de todos ellos. Es por ello que explicaremos a continuación como hemos resuelto cada apartado por separado y posteriormente el resultado final. A modo de aclaración, se incluirá piezas clave de código para entender mejor el funcionamiento.

4.1. Parte I: RTC

En primer lugar, dado que necesitamos una fuente fiable y que permanezca en hora independientemente de la alimentación del microcontrolador, decidimos implementar un reloj de tiempo real (RTC). Este se compone principalmente de un oscilador de cristal y un registro interno donde se almacenan las horas, minutos, segundos, etc. El protocolo de comunicación que utiliza es I2C.

Dado que los registros son de 8 bits, el primer paso fue realizar las funciones de lectura y escritura en 8 bits (ya que no había), basándonos en las incluidas en las librerías HAL_I2C.h. Estas funciones siguen la siguiente estructura y se pueden encontrar en el código anexo.

```
int I2C_read8(unsigned int slaveAdr, unsigned char writeByte);
void I2C_write8 (unsigned char address, unsigned char writeByte);
```

Por otro lado, para habilitar la temporización es necesario hacer la siguiente configuración inicial: habilitar el bit 8 del primer registro. Es por ello por lo que aparecen las siguientes líneas cuando se inicia el RTC:

```
I2CMasterSlaveAddrSet(I2C0_BASE, RTC_SLAVE_ADDRESS, 0);  // escritura
I2C_write8(MCONFIG_REG,ENABLE_CONFIG);          // habilita Oscilador
```

Siendo:

```
#define RTC_SLAVE_ADDRESS   0x68
#define ENABLE_CONFIG       (1<<8)
#define MCONFIG_REG          0x0
```

Una vez que se han habilitado la temporización por primera vez, se puede empezar a leer los registros, para ello nos tenemos que basar en el datasheet y las conversiones necesarias para obtener la fecha en los formatos habituales. Los registros son los siguientes:

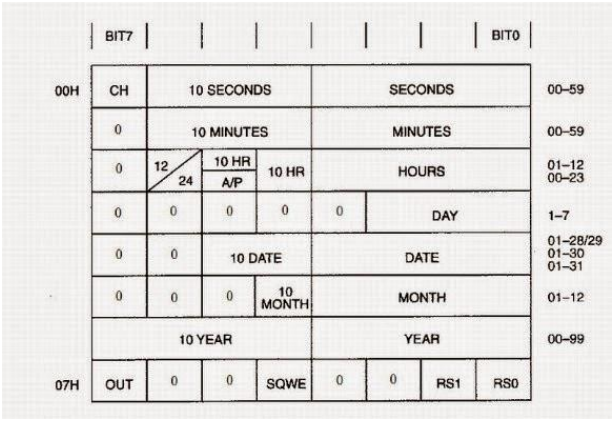

Ilustración 2: Registros internos del RTC

Los registros que nos interesan son los siguientes:

```
#define SEG_REG             0x0
#define MIN_REG             0x1
#define HOUR_REG            0x2
```

Por tanto, lo único que nos falta es leer dichos registros y realizar las conversiones especificadas. La función que realiza este procedimiento la hemos llamado **`rtc_readtime()`.** Esta función devuelve una estructura propia `mtime` que contiene los segundos, horas y minutos. Los cambios son los siguientes:

```
// segundos
raw = I2C_read8(RTC_SLAVE_ADDRESS,SEG_REG);
ss=(raw/16)*10+raw%16;
// minutos
raw = I2C_read8(RTC_SLAVE_ADDRESS,MIN_REG);
mm=(raw/16)*10+raw%16;
// horas
raw = I2C_read8(RTC_SLAVE_ADDRESS,HOUR_REG);
AMPM=bitn(raw,6); // si es am o pm
hh=raw%16;
hh=bitn(raw,4)*10+hh;
if(AMPM==0)
```

```
hh=hh+bitn(raw,5)*2;
// lo pasamos a struct
st.seg=ss;
st.min=mm;
st.hour=hh;
```

Además, para las horas nos ha sido necesario implementar una función que nos extraiga un bit concreto de un byte:

```
char bitn(char num, char numbit).
```

Esta función la podemos ver en el anexo.

4.2. Parte II: SENSORES

Para la lectura de sensores nos hemos servido de las prácticas realizadas. En primer lugar, hemos creado una función de inicialización, la cual recibe el número de boosterpack, el reloj, la frecuencia de lectura y si queremos texto por la UART.

```
minicia_sensores(int numbooster, uint32_t RELOJ, uint32_t fs_Hz, int bool_UART); //inicia los sensores
```

Por otra parte hemos realizado el mismo procedimiento para la lectura de los sensores, para tener el código más encapsulado, hemos realizado la función **`leesensores()`**. La lectura se realiza por variables globales ya que no hay problema de acceso simultáneo. Cabe mencionar que las medidas del acelerómetro no nos hacen falta, sólo nos vamos a interesar en la presión, temperatura, humedad y luminosidad.

Todos estos sensores, se irán almacenando a lo largo del día de modo que cuando llega la hora de riego corregimos el agua necesaria y podemos monitorizar o predecir cómo está la planta respecto de sus niveles óptimos de luminosidad y humedad. El funcionamiento de esta parte será explicado con posterioridad en el apartado 5 (funcionamiento completo).

4.3. Parte III: BOMBA

La bomba será la encargada de suministrar el agua necesaria a la plantación. Hemos elegido la bomba SongLong SL-381 principalmente por su precio. Tiene un caudal regulable cuyo máximo es de 250 L/h, de sobra para nuestro proyecto y perfectamente aplicable para un cultivo mediano. En ordenes de magnitud, el riego diario medio es de unos 5l/m^2, por lo que en una hora podríamos regar un huerto de unos 50m^2.

Cabe mencionar que esta bomba es sumergible, por lo que es bastante fácil la conducción del agua al huerto. Basta con sumergirla en un tanque o depósito y llevar el agua a través de un tubo.

Ilustración 3: Bomba utilizada

Uno de los inconvenientes de esta bomba es que no se puede controlar el riego electrónicamente. Al enchufarla, directamente comienza a bombear. Es por ello que hemos tenido que introducir un relé en el circuito que abra o corte el paso de agua.

Hemos escogido un relé songle srd-05vdc-sl-c, el cual debido a su popularización con arduino tenía un precio muy bajo. Funciona a 5v. El funcionamiento del relé es bastante simple, tiene una entrada digital, alimentación, tierra y dos pines donde se conecta el circuito que queremos cerrar o abrir. Para el manejo del relé se han definido las siguientes funciones:

```
#define RELE_ON GPIOPinWrite(puerto_RELAY,pin_RELAY,pin_RELAY);
#define RELE_OFF GPIOPinWrite(puerto_RELAY,pin_RELAY,0);
```

Por último, para completar el riego solo quedaría implementar una función que convierta del riego necesario en L de nuestra planta a tiempo de riego, dividiendo por el caudal de la Bomba. Este tiempo de riego lo calculamos en segundos y lo pasamos a minutos y segundos para sumarlo a la hora actual. Es decir, si son las 13:20:02 y necesitamos 123 segundos de riego la hora de riego sería a las 13:22:05. El procedimiento es leer el rtc cada segundo (a través de un temporizador interno) y comparar la hora actual con la hora objetivo. La función que realiza el riego es la siguiente y se puede encontrar en el anexo.

```
void riego(char ag_nec)
```

Adicionalmente hemos incluido una función que resta la fecha del calendario ya que sirve de utilidad en distintas ocasiones como indicar cuantos segundos y minutos faltan de riego aunque cambiemos el periodo de muestreo del rtc.

```
struct mtime restafecha(struct mtime obj,struct mtime act)
```

4.4. Parte IV: INTERFAZ GRAFICA

Para el desarrollo de la interfaz gráfica de la pantalla desde la que se controla el sistema, se han llevado a cabo diversas funciones para conseguir un diseño que facilite la interacción con el usuario y la comprensión del código.

Las funciones desarrolladas han sido las siguientes:

-*Dibuja_pantalla_presentación*: En esta función se desarrolla el código correspondiente a la pantalla de presentación en la cual se muestra la información correspondiente a los miembros del grupo, tal y como se muestra en la siguiente imagen:

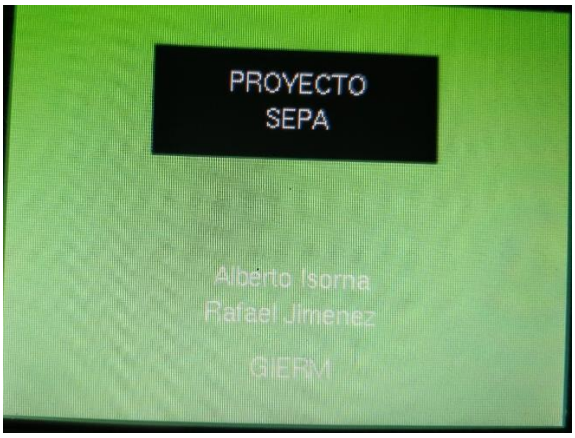

Ilustración 4 : Pantalla inicial

-*Dibuja_pantalla_inicial*: Al pulsar la pantalla táctil en la pantalla anterior se accede a la pantalla inicial, en la cual se lleva a cabo la elección del modo en el que queremos interactuar con el sistema (modo manual o automático). El código encargado de conseguir dicha interfaz se encuentra en esta función.

Ilustración 5 : Pantalla selección

-*Dibuja_pantalla_automatica*: En esta función se encapsula el código correspondiente al modo automático. En ella aparecen las imágenes correspondientes a las distintas plantas que nos permite escoger el sistema. Tras pulsar en alguna de las imágenes, accedemos a la pantalla correspondiente de dicha planta, donde se hace referencia a las características y necesidades de la elección. El aspecto de la pantalla correspondiente a esta función es la siguiente:

Ilustración 6: Elección de planta

-*Dibuja_plantaX*: Esta función incluye el código correspondiente a la interfaz de la planta seleccionada. En función de la escogida el valor del término 'X' será uno u otro, pudiendo ser Dibuja_planta1, Dibuja_planta2, Dibuja_planta3, Dibuja_planta4 . En ella se pueden observar las distintas necesidades de la planta como son la cantidad de agua necesaria, el fertilizante, las horas de sol y temperatura.

Para obtener una mejora visual, se han desarrollado otras funciones externas que son invocadas desde esta. Las funciones creadas han sido las siguientes:

- *Gota*: Permite dibujar un icono que represente una gota.

- *Sol*: Permite dibujar un icono que represente un sol.

- *RangoTemp*: Permite dibujar en una barra el rango de temperaturas admisible por la planta.

El resultado obtenido ha sido el siguiente:

Ilustración 7: modo automático

-*Dibuja_M_automatico*: Tras haber escogido una planta en el modo automático y haber aceptado las condiciones climatológicas capaces de soportar esta, se accede a una nueva pantalla en la que se pueden observar distintos indicadores del sistema, como son el nivel de agua del depósito, tiempo necesario para el próximo riego, el estado actual del toldo, y un indicador de las horas de sol que ha recibido el cultivo. Esto se encuentra encapsulado en la función *Dibuja_M_automatico*. Desde esta función se invocan a otras como son *MedidorLuminosidad* y *DibujaDepósito*. La primera se encarga de construir el gráfico que hace referencia a la cantidad de horas de sol recibidas. La segunda se encarga de dibujar un icono que representa el nivel de agua del depósito.

El resultado obtenido ha sido el siguiente:

Ilustración 8:Indicadores modo automático

-*Dibuja_pantalla_manual*: En esta función se encuentra el código correspondiente a la interfaz del modo manual. Como se ha mencionado previamente, en esta pantalla podemos distinguir 3

pulsadores encargados de controlar los actuadores, el primero se encarga de activar la bomba mientras que los otros dos se permiten extender y recoger el toldo.

Además, se ha incluido otro botón que permite acceder a una pantalla de medidas, en la cual se muestran por pantalla los valores recogidos por el Sensors Boosterpack.

Ilustración 9: Modo manual

El resultado obtenido ha sido el siguiente:

-*Dibuja_pantalla_medidas*: En esta función se incluye el código correspondiente a la pantalla de medidas a la que se accede a través de la pantalla del modo manual. Como se ha mencionado en el punto anterior, en esta se lleva a cabo una representación de las medidas recogidas por los distintos sensores. Para representar dicha información se ha hecho uso de la función ComTemometro que permite representar la temperatura a través de un icono que simula un termómetro. El resultado obtenido ha sido el siguiente:

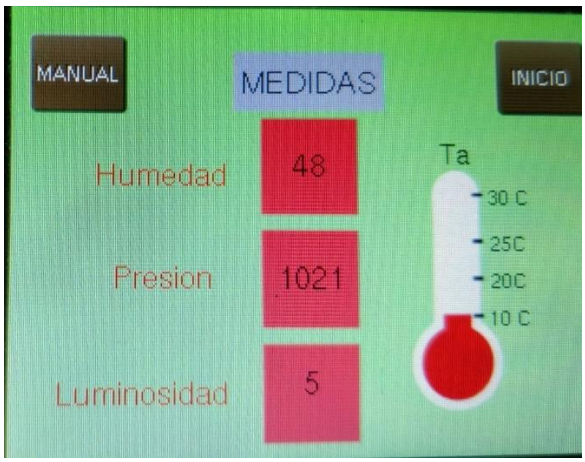

Ilustración 10: Medidas sensores

4.5. Parte V: REPRESENTACION DE IMÁGENES

Nos pareció interesante que cuando el usuario seleccionase una planta en cuestión viera una foto de esta. Vimos que en la guía de programación de la FT800 había varias funciones dedicadas al manejo de bits y nos adentramos a trabajar en ello.

En primer lugar, dado que no disponíamos de ninguna función para el manejo de imágenes, simplemente el pseudocódigo, intentamos realizar la representación de imágenes punto a punto. Esto es, para cada pixel calcular cuál es su color en cada canal, pintar dicho punto y realizar la misma operación con todos los puntos de la imagen. Este procedimiento, aparte de ser costoso y lento, no nos dio muy buenos resultados.

Por lo tantos decidimos empezar a investigar en varios documentos proporcionados por FTDI para poder representar las imágenes según el procedimiento habitual que recomienda el fabricante, pese a no tener ninguna función ya hecha. El diagrama a seguir sería el siguiente:

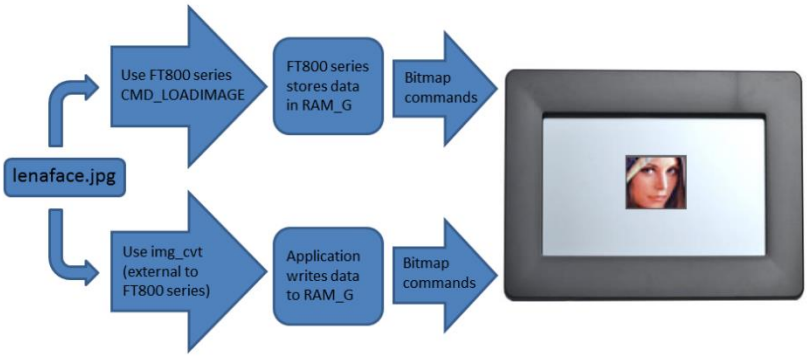

Ilustración 11: Esquema inclusión de imágenes

Elegimos la segunda opción, la de la aplicación externa, ya que desconocíamos como podíamos hacer una carga de un archivo desde el código.

Los pasos que seguimos por tanto para representar las imágenes fueron los siguientes:

1. Obtención de la imagen en raw, rawh: el primer paso es obtener el valor de todos los píxeles, con la codificación de colores propia de FTDI. Para ello la propia página proporciona un programa de conversión de imágenes a crudo, llamado *img_cvt.exe.* Este programa recibe como entrada la imagen en cuestión y como salida tiene varios archivos: raw, rawh, bin y binh. Los dos primeros son la imagen convertida en crudo al formato deseado (la que acaba en h es para incluirla directamente en el código y la otra para cargarla). Los dos últimos son con compresión, los cuales hay que descomprimir posteriormente con el comando CMD_INFLATE. En nuestro caso hemos elegido el formato rawh por simplicidad. En el código hemos incluido el fichero “raw_data.h” donde se encuentra las imágenes en cuestión.

2. Carga de imágenes en RAM gráfica: como ya tenemos la imagen en el formato deseado, el siguiente paso es cargar dicha imagen en la RAM gráfica, de forma que luego se pueda representar fácilmente. Para ello hemos realizado una función que escribe un byte en esta posición de memoria e incrementa el offset de escritura a cada paso.

```
void EscribeRam8_alb(char dato)
{
  HAL_SPI_CSLow();
  FT800_SPI_SendAddressWR(RAM_G+RAMG_Offset);
  FT800_SPI_Write8(dato);
  HAL_SPI_CSHigh();
  RAMG_Offset = FT800_IncCMDOffset(RAMG_Offset, 1);
}
```

Utilizando esta función cargamos la imagen ubicada en el fichero raw_data.h por completo con un bucle for.

Las propiedades de la imagen vienen guardadas en una estructura sugerida por la documentación de FTDI que facilita el posterior tratamiento de la imagen.

```
typedef struct SAMAPP_Bitmap_header
{
    uint8_t Format;
    int16_t Width;
    int16_t Height;
    int16_t Stride;
    int32_t Arrayoffset;
    uint8_t totalsize; // añadido
} SAMAPP_Bitmap_header_t;
```

3. Representación de imágenes: para la representación de imágenes nos hacen falta tres comandos fundamentales.

    a. BITMAP_SOURCE: especifica donde está ubicada la imagen, típicamente en RAM.
    b. BITMAP_LAYOUT: gracias a esta función concretamos el formato de color de la imagen y su tamaño.
    c. BITMAP_SIZE: ayuda a definir el tamaño donde se va a representar la imagen.

    En primer lugar, intentamos crear las funciones manualmente con la información que teníamos en la guía de la FT800 pero investigando en la página de FTDI encontramos un ejemplo donde venían definidas dichas funciones de una manera más simple.

    Por otro lado, teníamos que añadir los comandos propios de las imágenes que faltaban como los formatos de color o las opciones de representación.

    Las definiciones de dichas funciones y los parámetros se encuentran en el anexo en la sección imagen.

    Por último, solo falta crear una función que represente las imágenes. Dicha función recibe como parámetro el número de imagen y la posición a representar:

```
// ---------------------------- Dibuja Imagen guardada en rawdata.h  --------
------
void cmd_bitmap(char num, unsigned int X, unsigned int Y)
```

```
{
    // ------------ INICIALIZACION ---------------
    Comando(BITMAP_HANDLE(num));
    Comando(BITMAP_SOURCE(RAM_G+RawData_Header[num].Arrayoffset));
Comando(BITMAP_LAYOUT(RawData_Header[num].Format,RawData_Header[num].Stride,R
awData_Header[num].Height));
Comando(BITMAP_SIZE(NEAREST,BORDER,BORDER,RawData_Header[num].Width,RawData_H
eader[num].Height));

    // ------------ BIBUJAMOS BITMAPS ------------
    Comando(COLOR_RGB(255,255,255));
    Comando(BEGIN(BITMAPS));                // comenzamos el bitmap
    Comando(VERTEX2II(X,Y,num,0));          // handle específico
    Comando(CMD_END);
}
```

Cabe mencionar que para la representación de varias imágenes a la vez es necesario asignarle a cada imagen su manejador (handle) y representar el bitmap en concreto con la función `VERTEX2II(x,y,handle,cell);`

Tras numerosos intentos en los que no se representaban imágenes, notamos que la conversión en el formato RGB565 no daba buenos resultados ya que cada color ocupaba dos bytes cuando la imagen la habíamos definido con un byte. Al realizar la conversión con RGB332 que si cabe en un byte ya funcionó correctamente. A continuación podemos ver la imagen de la portada de este proyecto en la FT800, imagen usada como imagen de bienvenida:

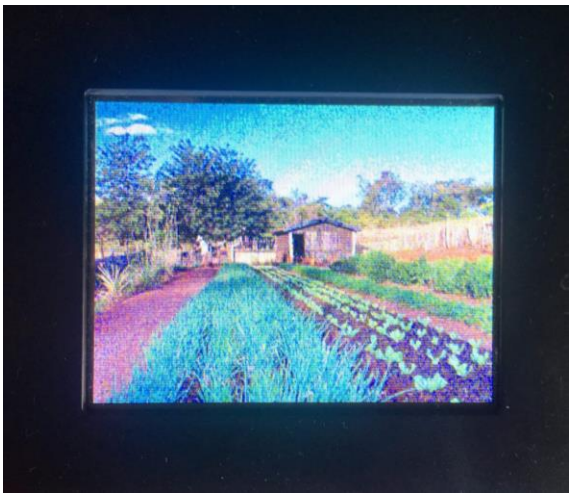

Ilustración 12: Pantalla de Bienvenida

5. Funcionamiento completo

En este apartado se va a llevar a cabo una explicación más a fondo del funcionamiento del sistema que se ha desarrollado.

En primer lugar, el usuario debe de interacciona con el sistema a través de la pantalla táctil. En ella se escogerá entre los distintos modos de funcionamiento que se desee a partir de la lectura

de la pantalla táctil. Desde el punto de vista del software, las distintas pantallas y estados de funcionamiento del sistema se ha modelado mediante una máquina de estados.

Partiendo desde la pantalla principal, en el caso de que se haya seleccionado la opción del modo manual, se almacenará en la variable pantalla el valor 3. En este estado, el usuario solo tendrá acceso al manejo de la bomba y el toldo. El primero de ellos se activa a partir de un relé, donde la activación de este provoca que se active la bomba. Por otro lado, el actuador encargado de la apertura y cierre del toldo es un paso a paso. Se ha decidido escoger este motor debido a que la maqueta realizada no requiere un par demasiado grande y, además, no consume mucha energía, en una situación real, el motor que se utilizaría podría ser también un paso a paso, pero con un mayor torque.

Por otro lado, si la opción escogida es el modo automático, la variable estado pasará a tener el valor 2. En este modo la primera decisión que se debe de tomar es la planta que se desea cultivar. Para ello aparecerá una serie de imágenes sobre las que se debe pulsar. Una vez escogida está, la variable estado cambiará de valor en función de la planta seleccionada. Tras ser seleccionada una planta, aparecerá una serie de gráficos representativos de dicha planta. Tras presionar sobre el botón de continuar comenzará el modo automático. Este modo, como ya se ha explicado previamente, permite que el usuario solo se deba de preocupar de plantar la planta, de forma que el riego se haga de forma autónoma dependiendo de la información recogida por los sensores, siempre y cuando exista agua en el depósito. Para conocer el estado del sistema, en la pantalla se mostrarán distintos indicadores.

6. Líneas futuras

A pesar de que IHuerting pueda satisfacer las necesidades de un importante sector de la población, es evidente que puede ser mejorado.

La primera mejora que se podría llevar a cabo es un incremento de la variedad de plantas que se puedan escoger, de forma que para poder observar las distintas plantas se desarrollase una función que permitiese el desplazamiento de las imágenes deslizando el dedo por la pantalla, simulando el comportamiento de los teléfonos móviles.

Desde el punto de vista de la toma de medidas, podría ser añadido un sensor de humedad del terreno, el cual supondría una mejora en la decisión sobre la duración del riego.

Otra posible mejora de nuestro sistema sería la creación de un servidor web, al cual se pudiese acceder desde cualquier ordenador, pudiendo conocer las medidas de los distintos sensores en cualquier sitio, sin tener que consultarlo a través de la pantalla VM800.

Por último, como fuente de alimentación podría utilizarse una placa solar, de forma que la corriente necesaria por el sistema fuese proporcionada por esta.

7. Costes

En este apartado se analizarán los costes del proyecto. En la siguiente tabla se expone una tabla del precio de cada componente por separado y el precio total.

Componente	Precio
Microprocesador TIVA	20.00 €
Pantalla FT800	40.00 €
Sensors boosterpack	40.00 €
RTC	1.00 €
Batería	5.00 €
Bomba	10.00 €
Relé	1.00 €
Toldo	3.00 €
Motor paso a paso	1.50 €
Conexionado	4.00 €
Otros	2.00 €
TOTAL	127.50 €

Vemos como el grueso de los costes los componen la pantalla, el microprocesador y los sensores. La solución más simple para abaratar el proyecto sería prescindir de la pantalla y comunicarnos por bluetooth por ejemplo a través de una App móvil. En este caso no haría falta un procesador de estas características, bastaría con un MSP430 o un ARDUINO, atarantando muchísimo el proyecto.

En términos de beneficios, no es un producto que soluciona todo los problemas que conlleva el crecimiento de la planta por lo que habría que mantener cierto personal de abono y de plagas, pero si es un paso más para una futura automatización completa del campo.

Por otro lado, pese a tener unas 2000 líneas de código y otras 2000 que contienen las imágenes hemos ocupado el 20% de capacidad de flash del micro, por lo que para sacarle rendimiento, podríamos añadir muchos más módulos y controlar no sólo un huerto como es el caso de nuestra maqueta, sino una plantación completa muy compleja.

8. Conclusiones del proyecto

En este proyecto es posible apreciar como con poco presupuesto es posible lograr automatizar procesos simples de nuestro día a día, permitiéndonos lograr una mayor eficiencia, comodidad y ahorro energético.

Por otro lado, hay que destacar que se ha decidido utilizar el microcontrolador TM4C1294 por diversos motivos:

• Dispone de 2 conectores Boosterpack, lo que nos permite utilizar de manera más sencilla tanto la pantalla como el Sensors boosterpack.

• Dispone de 40 pines accesibles en el borde, permitiéndonos una gran flexibilidad a la hora de utilizar periféricos.

• Se dispone de librería de funciones para configuración y manejo de los periféricos que simplifican el trabajo.

• Desde el punto de vista de la memoria, dispone de 4GB de memoria, permitiendo ser extendida mediante una memoria externa, lo que supone una gran ventaja.

• Por último, hay que destacar su alta relación calidad-precio, permitiendo llevar a cabo este proyecto y otros más complejos por un precio alrededor de los 20€.

En cuanto a conclusiones generales cabe destacar que hemos aprendido bastante acerca del funcionamiento de la pantalla y el resto de actuadores empleados, llegando a incluir nuestras propias funciones, librerías, definiciones, etc. Observamos el enorme potencial de este micro para su precio, comparado con el del año anterior. Por otro lado el manejo del RTC nos ha dado una visión más clara de cómo funciona la comunicación I2C ya que la hemos tenido que construir de cero prácticamente. En cuanto al huerto, vemos como la posibilidades de automatización y de ampliación son infinitas y que es un producto bastante viable y que se puede explotar para controlar una plantación o doméstica o a gran escala.

9. Anexo: Código Completo

```
/////////// LIBRERIAS PANTALLA
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "inc/hw_gpio.h"
#include "driverlib/ssi.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom_map.h"
#include "driverlib/gpio.h"
#include "FT800_TIVA.h"

/////////// LIBRERÍAS SENSORES
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <math.h>



#include "inc/hw_memmap.h"
#include "inc/hw_ints.h"
#include "inc/hw_types.h"
#include "inc/hw_i2c.h"

#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "driverlib/timer.h"
#include "driverlib/i2c.h"
#include "driverlib/systick.h"
```

```
#include "utils/uartstdio.h"

#include "HAL_I2C.h"
#include "bme280.h"
#include "bmi160.h"
#include "OPT3001.h"
#include "tmp007.h"

// para strtok
#include <string.h>
#include <stdlib.h>
////////////////////////////////////////




// ---------------- RTC -----------------------
#define RTC_SLAVE_ADDRESS   0x68

#define SEG_REG             0x0
#define MIN_REG             0x1
#define HOUR_REG            0x2

#define ENABLE_CONFIG       (1<<8)
#define MCONFIG_REG          0x0

//funciones
int I2C_read8(unsigned int slaveAdr, unsigned char writeByte);
void I2C_write8 (unsigned char address, unsigned char writeByte);
char bitn(char num,char numbit);
struct mtime rtc_readtime();

char buffer[100];

struct mtime
{
    char seg;
```

```
    char min;
    char hour;
};

// ------------------ RELAY ------------------------
#define perif_RELAY                 SYSCTL_PERIPH_GPIOE
#define puerto_RELAY                GPIO_PORTE_BASE
#define pin_RELAY                   GPIO_PIN_0

#define RELE_OFF GPIOPinWrite(puerto_RELAY,pin_RELAY,pin_RELAY);
#define RELE_ON GPIOPinWrite(puerto_RELAY,pin_RELAY,0);

// ------------------ BOMBA ------------------------
struct mtime horariego,diff,temp;         // hora de riego
#define HOUR_RIEGO  0           // horas
#define MIN_RIEGO   28          // minutos
#define SEG_RIEGO   10          // segundos
#define L_RIEGO     2           // cant de agua L
float   caudal      =0.1;       // caudal en L/S

void riego(char ag_nec);
struct mtime restafecha(struct mtime obj,struct mtime act);

// ------------------ SENSORES ------------------------
void leesensores();             // lee de los sensores
void imprimemedidasUART();      // imprime las medidas por la uart
void minicia_sensores(int numbooster, uint32_t RELOJ, uint32_t fs_Hz, int bool_UART);
//inicia los sensores

int fc=0;

// VARIABLES SENSORES
void Timer0IntHandler(void);

char cambia=0;
char string[80];
char string2[80];
int DevID=0;
```

```
float Tfinal;

// OPT3001 - Luminosidad
float lux;
int lux_i;

// TMP007 -  Temperatura
int16_t T_amb, T_obj;
float Tf_obj, Tf_amb;
int  T_amb_i, T_obj_i;

// BME280
int returnRslt;
int g_s32ActualTemp   = 0;
unsigned int g_u32ActualPress  = 0;
unsigned int g_u32ActualHumity = 0;
struct bme280_t bme280;

// BMI160/BMM150
int8_t returnValue;
struct bmi160_gyro_t        s_gyroXYZ;
struct bmi160_accel_t       s_accelXYZ;
struct bmi160_mag_xyz_s32_t s_magcompXYZ;


// Calibration off-sets
int8_t accel_off_x;
int8_t accel_off_y;
int8_t accel_off_z;
int16_t gyro_off_x;
int16_t gyro_off_y;
int16_t gyro_off_z;
float T_act,P_act,H_act;
bool BME_on = true;

int T_uncomp,T_comp;
char mode;
```

```
long int inicio, tiempo;
int P_disp;

// --------------------- PANTALLA -----------------------------------

// Funciones
void minicia_pantalla(int numbooster,uint32_t RELOJ); // inicia la pantalla
void drawcircle(int r,int Xp,int Yp);
void pintafondo(char R, char G, char B);
//void prueba();
void cmd_progress( int16_t x,
                   int16_t y,
                   int16_t w,
                   int16_t h,
                   uint16_t options,
                   uint16_t val,
                   uint16_t range );
//

#define dword long
#define byte char
#define XPANT 320
#define YPANT 240

char chipid = 0;                        // Holds value of Chip ID read from the FT800

unsigned long cmdBufferRd = 0x00000000;           // Store the value read from the
REG_CMD_READ register
unsigned long cmdBufferWr = 0x00000000;           // Store the value read from the
REG_CMD_WRITE register
unsigned int t=0;

int Fin_Rx=0;
char Buffer_Rx;
unsigned long POSX, POSY, BufferXY;
unsigned long POSYANT=0;
unsigned int CMD_Offset = 0;
unsigned long REG_TT[6];
const unsigned long REG_CAL[6]={21959,177,4294145463,14,4294950369,16094853};
```

```
#define xtexto 160
#define ytexto 120
#define tam 22

// para los textos
char bufplaca[50];
char buffer2[40];
float media, medida1,medida2;

#define NUM_SSI_DATA            3

// variables de dibujo
int xc,yc,r; // circulo
#define xprog 40



// ============================================================
// ----------------------- IMAGEN -----------------------------
// ============================================================

void cmd_bitmap(char num, unsigned int X, unsigned int Y);
void loadIm_toRAM();
enum imagenes {tomate,girasol,lirio,batata,huerto};

typedef struct SAMAPP_Bitmap_header
{
    uint32_t Format;
    int16_t Width;
    int16_t Height;
    int16_t Stride;
    int32_t Arrayoffset;
    uint32_t totalsize; // añadido
}SAMAPP_Bitmap_header_t;

#define BITMAPS              1
#define RGB332               4
#define RGB565               7
```

```
#define NEAREST              0

#define BORDER               0

#define ARGB1555             0

#define ARGB2                5

#define ARGB4                6

#define PALETTED             8



#define BEGIN(prim) ((31UL<<24)|(((prim)&15UL)<<0))

#define BITMAP_SOURCE(addr) ((1UL<<24)|(((addr)&0xfffff)<<0))

#define                                          BITMAP_LAYOUT(format,linestride,height)
((7UL<<24)|(((format)&31UL)<<19)|(((linestride)&1023UL)<<9)|(((height)&511UL)<<0))

#define                                     BITMAP_SIZE(filter,wrapx,wrapy,width,height)
((8UL<<24)|(((filter)&1UL)<<20)|(((wrapx)&1UL)<<19)|(((wrapy)&1UL)<<18)|(((width)&511
UL)<<9)|(((height)&511UL)<<0))

#define CLEAR(c,s,t) ((38UL<<24)|(((c)&1UL)<<2)|(((s)&1UL)<<1)|(((t)&1UL)<<0))

#define                                                    COLOR_RGB(red,green,blue)
((4UL<<24)|(((red)&255UL)<<16)|(((green)&255UL)<<8)|(((blue)&255UL)<<0))

#define                                                  VERTEX2II(x,y,handle,cell)
((2UL<<30)|(((x)&511UL)<<21)|(((y)&511UL)<<12)|(((handle)&31UL)<<7)|(((cell)&127UL)<<
0))

#define BITMAP_HANDLE(handle) ((5UL<<24)|(((handle)&31UL)<<0))

#define                                              CLEAR_COLOR_RGB(red,green,blue)
((2UL<<24)|(((red)&255UL)<<16)|(((green)&255UL)<<8)|(((blue)&255UL)<<0))

#define VERTEX2F(x,y) ((1UL<<30)|(((x)&32767UL)<<15)|(((y)&32767UL)<<0))



unsigned int RAMG_Offset=0;



#include "raw_data.h"



void EscribeRam8_alb(char dato);



// ------------------ OTROS -----------------

#define MSEC 40000



void inicia_UART();



int RELOJ;



void inicia_i2c();



#define ANSI_COLOR_RED     "\x1b[31m"
```

```
#define ANSI_COLOR_GREEN   "\x1b[32m"

#define ANSI_COLOR_YELLOW  "\x1b[33m"

#define ANSI_COLOR_BLUE    "\x1b[34m"

//#define ANSI_COLOR_MAGENTA "\x1b[35m"

//#define ANSI_COLOR_CYAN    "\x1b[36m"

#define ANSI_COLOR_RESET   "\x1b[0m"


enum estado {PLANTACION, RIEGO, FIN};

enum estado miestado=PLANTACION;


// ------------------- Interfaz

void  ComTermometro(  int16_t  x,int16_t  y,int16_t  longitud,int16_t  ancho,uint16_t
major,uint16_t minor,float val);

void ComGauge( int16_t x,int16_t y,int16_t r,uint16_t major,uint16_t minor,float val);

void Gota( int16_t x,int16_t y,int16_t r);

void Sol( int16_t x,int16_t y,int16_t r,int16_t r2);

void   RangoTemp(   int16_t   x,int16_t   y,int16_t   longitud,int16_t   ancho,uint16_t
minor,uint16_t major);

void MedidorLuminosidad( int16_t x,int16_t y,int16_t R,int16_t dif,float val,uint16_t
Red,uint16_t G,uint16_t B );

void DibujaDeposito(int16_t x,int16_t y,int16_t ancho,int16_t alto,float val);


void Dibuja_pantalla_presentación();

void Dibuja_pantalla_inicial();

void Dibuja_planta1();

void Dibuja_planta2();

void Dibuja_planta3();

void Dibuja_planta4();

void Dibuja_pantalla_automatica();

void Dibuja_pantalla_manual();

void Dibuja_pantalla_medidas();

void Dibuja_M_automatico();




int pulsado_manual;

int pulsado_automatico;

int pulsado_riego;

int pulsado_toldo;

int pulsado_info;
```

```
int pulsado_atras;




// ---------------------- Paso a paso

#define NUMPASOS 4

unsigned int steps_left=4095;
char Direction = 1;
int Steps = 0;

// paso mas rapido
unsigned int Paso [ 4 ][ 4 ] =
{   {1, 0, 0, 0},
    {0, 1, 0, 0},
    {0, 0, 1, 0},
    {0, 0, 0, 1}
};

// paso mas lento
unsigned int PasoL [ 8 ][ 4 ] =
{   {1, 0, 0, 0},
    {1, 1, 0, 0},
    {0, 1, 0, 0},
    {0, 1, 1, 0},
    {0, 0, 1, 0},
    {0, 0, 1, 1},
    {0, 0, 0, 1},
    {1, 0, 0, 1}
};

uint32_t PinM[]={
                GPIO_PIN_0,
                GPIO_PIN_1,
                GPIO_PIN_2,
                GPIO_PIN_3
};
```

```
int var_paso=0;

void stepper(unsigned int Direction)            //Avanza un paso
{
    char i;

    if (Direction==2)
        for(i=0;i<4;i++)
            GPIOPinWrite(GPIO_PORTL_BASE,PinM[i],0); // paramos
    else
    {
        // escritura
        for(i=0;i<4;i++)
            GPIOPinWrite(GPIO_PORTL_BASE,PinM[i],PinM[i]*Paso[Steps][i]);
        //Esperamos 1 ms para que se polaricen las bobinas.
        SysCtlDelay(1*MSEC);
        // ajuste de pasos
        if(Direction)
            Steps++;
        else
            Steps--;

        if (Steps>NUMPASOS-1)
            Steps=0 ;
        if (Steps<0)
            Steps=NUMPASOS-1 ;
    }
}

// ----------------------- MAIN ---------------------------------
int main(void)
{
    int pantalla=0; /* Variable que servirá para identificar la pantalla */
    int POS_X_anterior;
    int POS_Y_anterior;
```

```
    // variables rtc
    horariego.hour=HOUR_RIEGO;
    horariego.min=MIN_RIEGO;
    horariego.seg=SEG_RIEGO;

    uint32_t fs_Hz = 1; // frecuencia de muestreo de los sensores

    //relay
    SysCtlPeripheralEnable(perif_RELAY);
    GPIOPinTypeGPIOOutput(puerto_RELAY, pin_RELAY);
    RELE_OFF

    // paso a paso
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOL);
    GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE,
GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4);

    // reloj
    RELOJ=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN | SYSCTL_USE_PLL |
SYSCTL_CFG_VCO_480), 120000000);

    // inicia i2c y uart
    inicia_i2c();
    inicia_UART();

    // inicia RTC
    I2CMasterSlaveAddrSet(I2C0_BASE, RTC_SLAVE_ADDRESS, 0); // escritura
    I2C_write8(MCONFIG_REG,ENABLE_CONFIG); // habilita Oscilador

    // Inicia sensores
    minicia_sensores(1,RELOJ,fs_Hz,1); // boosterpack 1, con 1 medidas por segundo,
mostrando por uart

    // Inicia pantalla
    minicia_pantalla(2,RELOJ);  // inicia pantalla en boosterpack 2

    sprintf(buffer,ANSI_COLOR_YELLOW"Hora de riego ==>
%d:%d:%d\n\n\r"ANSI_COLOR_RESET,horariego.hour,horariego.min,horariego.seg);
    UARTprintf(buffer);
```

```
int flag=0;

// escribe en la RAM_G
loadIm_toRAM();

// motor paso a paso
var_paso=0;

// Pantalla de inicio
Nueva_pantalla(136,245,132);
ComColor(50,50,50);
cmd_bitmap(huerto,0,0);
Dibuja();
Espera_pant();

while(1)
{
    if (cambia)
    {
        cambia=0;
        // leemos del rtc
        temp=rtc_readtime();
    }

    // Maquina de Estados
    switch (pantalla)
    {
    case 0:
        /* Pantalla presentacion */
        Dibuja_pantalla_presentación();
        Lee_pantalla();
        if(POSX>0 && POSX<320 && POSY>0 && POSY<240)
            pantalla=1;
        else
            pantalla=0;
        break;
```

```
        case 1:
            /* Pantalla Inicial */
            Dibuja_pantalla_inicial();
            Lee_pantalla();
            if(pulsado_manual==1)
            {
                if(!(POSX>=20 && POSX<=130 && POSY>=180 && POSY<=220)) /*Si ya no se
esta pulsando */
                {
                    pantalla=3; /*Pantalla modo manual */
                    pulsado_manual=0;
                    break;
                }
            }
            if(pulsado_automatico==1)
            {
                if(!(POSX>=180 && POSX<=290 && POSY>=180 && POSY<=220)) /* Si ya no se
esta pulsando */
                {
                    pantalla=2; /*Pantalla modo automático */
                    pulsado_automatico=0;
                    break;
                }
            }
            if(POSX>=20 && POSX<=130 && POSY>=180 && POSY<=220)
            {
                pulsado_manual=1;
            }
            if(POSX>=180 && POSX<=290 && POSY>=180 && POSY<=220)
            {
                pulsado_automatico=1;
            }
            break;



        case 2:
            /* Modo automático */


            Dibuja_pantalla_automatica();
```

```
            Lee_pantalla();

            POS_X_anterior=POSX;
            POS_Y_anterior=POSY;

            if(POS_X_anterior==32768 && POS_Y_anterior==32768)
            {
                flag=1;
            }/* Saldra cuando la posición del pulsador sea diferente */

            if(flag==1)
            {
                if(pulsado_atras==1)
                {
                    if(!(POSX>=270 && POSX<=305 && POSY>=15 && POSY<=45))
                    {
                        pulsado_atras=0;
                        pantalla=1; /* Pantalla Inicial */
                        flag=0;
                    }

                }
                if(POSX>=270 && POSX<=305 && POSY>=15 && POSY<=45)
                {
                    pulsado_atras=1;
                }

                /* Planta 1 */
                if(POSX>=50 && POSX<=130 && POSY>=70 && POSY<=130)
                {
                    pantalla=21; /* Pantalla Planta 1 */
                    flag=0;
                }
                /* Planta 2*/
                if(POSX>=190 && POSX<=270 && POSY>=70 && POSY<=130)
                {
                    pantalla=22; /* Pantalla Planta 2 */
                    flag=0;
```

```
            }
            /* Planta 3*/
            if(POSX>=50 && POSX<=130 && POSY>=150 && POSY<=210)
            {
                pantalla=23; /* Pantalla Planta 3 */
                flag=0;
            }
            /* Planta 4 */
            if(POSX>=190 && POSX<=270 && POSY>=150 && POSY<=210)
            {
                pantalla=24; /* Pantalla Planta 4 */
                flag=0;
            }
        }
        break;

    case 21:
        /* Planta 1 */

        Dibuja_planta1();
        Lee_pantalla();
        if(POSX>=15 && POSX<=55 && POSY>=15 && POSY<=45)
        {
            pantalla=2; /* Pantalla Inicial */
            flag=0;

        }
        if(POSX>=265 && POSX<=305 && POSY>=10 && POSY<=40)
        {
            pantalla=40; /* Pantalla Inicial */
            flag=0;

        }
        break;

    case 22:
        /* Planta 2 */
        Dibuja_planta2();
```

```
            Lee_pantalla();

            if(POSX>=15 && POSX<=55 && POSY>=15 && POSY<=45)
            {
                pantalla=2; /* Pantalla Inicial */
                flag=0;

            }
            if(POSX>=265 && POSX<=305 && POSY>=10 && POSY<=40)
            {
                pantalla=40; /* Pantalla Inicial */
                flag=0;
            }
            break;

        case 23:
            /* Planta 3 */
            Dibuja_planta3();
            Lee_pantalla();
            if(POSX>=15 && POSX<=55 && POSY>=15 && POSY<=45)
            {
                pantalla=2; /* Pantalla Inicial */
                flag=0;
            }

            if(POSX>=265 && POSX<=305 && POSY>=10 && POSY<=40)
            {
                pantalla=40; /* Pantalla Inicial */
                flag=0;
            }

            break;

        case 24:
            /* Planta 4 */

            Dibuja_planta4();
            Lee_pantalla();
```

```
    if(POSX>=15 && POSX<=55 && POSY>=15 && POSY<=45)
    {
        pantalla=2; /* Pantalla Inicial */
        flag=0;

    }
    if(POSX>=265 && POSX<=305 && POSY>=10 && POSY<=40)
    {
        pantalla=40; /* MODO AUTOMATICO */
        flag=0;

    }
    break;

case 3:

    /* Pantalla modo manual */
    Dibuja_pantalla_manual();
    Lee_pantalla();
    /* Atrás */
    if(pulsado_atras==1)
    {
        if(!(POSX>=270 && POSX<=305 && POSY>=15 && POSY<=45))
        {
            pulsado_atras=0;
            pantalla=1; /* Pantalla Inicial */
        }

    }
    if(POSX>=270 && POSX<=305 && POSY>=15 && POSY<=45)
    {
        pulsado_atras=1;
    }
    /* informacion */
    if(pulsado_info==1)
    {
        if(!(POSX>=135 && POSX<=185 && POSY>=175 && POSY<=225))
```

```
                {
                    pulsado_info=0;
                    pantalla=10; /* Pantalla que muestra medida de los sensores. */

                }
            }
            if(POSX>=135 && POSX<=185 && POSY>=175 && POSY<=225)
            {
                pulsado_info=1;
            }

            /* Activar bomba */
            if(pulsado_riego==1)
            {
                if(!(POSX>=50 && POSX<=100 && POSY>=150 && POSY<=190))// Si se ha dejado
de pulsar
                {
                    pulsado_riego=0;
                    RELE_OFF
                }
            }

            if(POSX>=50 && POSX<=100 && POSY>=150 && POSY<=190)
            {
                pulsado_riego=1;
                /* Se debe de encender el relé para que echeagua la bomba*/
                RELE_ON
            }
            /* Toldo */
            if(pulsado_toldo==1)
            {
                if(!(POSX>=210 && POSX<=250 && POSY>=150 && POSY<=190))// Si se ha
dejado de pulsar
                {
                    pulsado_toldo=0;
                    var_paso=0;
                }
            }
            if(POSX>=210 && POSX<=250 && POSY>=150 && POSY<=190)
```

```
            {
                /* Se debe de encender el paso a paso para que el toldo se expanda.*/
                pulsado_toldo=1;
                var_paso=1;
            }
            if(pulsado_toldo==-1)
            {
                if(!(POSX>=260 && POSX<=300 && POSY>=150 && POSY<=190))// Si se ha
dejado de pulsar
                {
                    pulsado_toldo=0;
                    var_paso=0;
                }
            }
            if(POSX>=260 && POSX<=300 && POSY>=150 && POSY<=190)
            {
                /* Se debe de encender el paso a paso para que el toldo se contraiga.*/
                var_paso=-1;
                pulsado_toldo=-1;
            }

            break;

        case 10:
            /* Pantalla utilizada para mostrar las medidas de los sensores. */
            /* Se debe demostrar la temperatura, la humedad y la luminosidad. */
            Dibuja_pantalla_medidas();
            leesensores();
            ComTermometro( 240,190,90,25,30,15,Tfinal);

            ComNum(160,135, 22, OPT_CENTERX,(int)P_act );
            ComNum(160,75, 22, OPT_CENTERX,(int)H_act );
            ComNum(160,190, 22, OPT_CENTERX,(int)lux_i );

            Dibuja();

            Lee_pantalla();
            /* Pantalla inicio */
```

```
            if(pulsado_atras==1)
            {
                if(!(POSX>=270 && POSX<=305 && POSY>=15 && POSY<=45))
                {
                    pulsado_atras=0;
                    pantalla=1; /* Pantalla Inicial */
                }

            }
            if(POSX>=270 && POSX<=305 && POSY>=15 && POSY<=45)
            {
                pulsado_atras=1;
            }
            if(pulsado_manual==1)
            {
                if(!(POSX>=15 && POSX<=50 && POSY>=15 && POSY<=45))
                {
                    pulsado_manual=0;
                    pantalla=3;/* Pantalla manual */
                }
            }
            if(POSX>=15 && POSX<=50 && POSY>=15 && POSY<=45)
            {
                pulsado_manual=1;
            }
            break;

        case 40:
            /* En esta pantalla se va a mostrar:
             * El tiempo necesario para que se vuelvan a regar las plantas.
             * La planta que se ha plantado (VARIABLE planta)
             * Si el toldo se encuentra cerrado o abierto
             * Tanto por ciento de luminosidad recibida
             * Tanto por ciento de agua en el depósito.
             * */
            Dibuja_M_automatico();
            MedidorLuminosidad(230,190,40,20,0.75,0,0,0);
            Dibuja();
```

```
            Lee_pantalla();


            // Si coincide regamos
            if(temp.seg==horariego.seg)
            {
                fc=(float)(-1*(g_u32ActualHumity/50000)+L_RIEGO);  //  0%  -------  L
|||||| 100% --------- 0L
                riego((char)fc);
            }
            // espera 1 seg
            cambia=0;
            while(cambia==0);


            if(pulsado_atras==1)
            {
                if(!(POSX>=15 && POSX<=55 && POSY>=10 && POSY<=40))
                {
                    pulsado_atras=0;
                    pantalla=1; /* Pantalla Inicial */
                    flag=0;
                }


            }
            if(POSX>=15 && POSX<=55 && POSY>=10 && POSY<=40)
            {
                pulsado_atras=1;
                flag=0;
            }
            break;
        }


        // MOTOR PASO A PASO
        switch(var_paso)
        {
        case 0://El paso a paso no se mueve.
            //Poner las salidas a cero.
            stepper(2);
            break;
```

```
        case 1://gira derecha (El toldo se expande)
            stepper(1);
            break;

        case -1://Gira izquierda (El toldo se contrae)
            stepper(0);
            break;

        default:
            SysCtlDelay(5*MSEC); //Esperamos 1 segundo para que de tiempo a que el servo
gire.
            break;
        }
    }
    return 0;
}

// -------------------------
void Dibuja_pantalla_presentación()
{
    Nueva_pantalla(136,245,132);
    ComColor(0,0,0);
    Comando(CMD_BEGIN_RECTS);//Se escribe cada vez que vayamos a escribir rectangulos.
    ComVertex2ff(80,30);
    ComVertex2ff(240,90);
    Comando(CMD_END);//Se deja de escribir lo anterior

    ComColor(255,255,255);
    ComTXT(160,40, 21, OPT_CENTERX,"PROYECTO");
    ComTXT(160,60, 21, OPT_CENTERX,"SEPA");
    ComTXT(160,150, 21, OPT_CENTERX,"Alberto Isorna");
    ComTXT(160,170, 21, OPT_CENTERX,"Rafael Jimenez");
    ComTXT(160,200, 21, OPT_CENTERX,"GIERM");
    Dibuja();
}

void Dibuja_pantalla_inicial()
```

```
{

    Nueva_pantalla(136,245,132);
    ComColor(0,0,0);
    ComPointSize(55);
    Comando(CMD_BEGIN_POINTS);//Se escribe cada vez que vayamos a escribir rectangulos.
    ComVertex2ff(160,75);
    Comando(CMD_END);//Se deja de escribir lo anterior

    //Negro
    ComColor(255,255,255);//Lo que se pinte despues de esto va de este color
    ComTXT(160,50, 21, OPT_CENTERX,"BIENVENIDO");
    ComTXT(160,70, 21, OPT_CENTERX,"A");
    ComTXT(160,90, 21, OPT_CENTERX,"I HUERTING");

    ComColor(255,100,100);//Lo que se pinte despues de esto va de este color
    ComTXT(160,140, 21, OPT_CENTERX,"Seleccione el modo");
    ComLineWidth(2);

    if(pulsado_manual==0)
    {
        ComColor(255,255,255);
        ComFgcolor (255,146,20);
        ComButton(20,180,110,40,20,0,"Modo Manual");
    }
    else
    {
        ComColor(0,0,0);
        ComFgcolor (255,146,20);
        ComButton(20,180,110,40,20,0,"Modo Manual");
    }
    if(pulsado_automatico==0)
    {
        ComColor(255,255,255);
        ComFgcolor (255,70,20);
        ComButton(180,180,110,40,20,0,"Modo automatico");
    }
    else
```

```
    {
        ComColor(0,0,0);
        ComFgcolor (255,70,20);
        ComButton(180,180,110,40,20,0,"Modo automatico");
    }
    Dibuja();

}
/* Pantalla modo automatico */
void Dibuja_pantalla_automatica()
{
    Nueva_pantalla(136,245,132);
    /* Botón de atrás */
    if(pulsado_atras==0)
    {
        ComFgcolor(100,70,20);
        ComColor(0,0,0);
        ComButton(265,15,40,30,20,0,"ATRAS");
    }
    else
    {
        ComFgcolor(100,70,20);
        ComColor(255,255,255);
        ComButton(265,15,40,30,20,0,"ATRAS");
    }
    ComColor(0,0,0);
    ComTXT(160,25, 21, OPT_CENTERX,"SELECCIONE UNA PLANTA");
    ComLineWidth(2);

    /*Primer rectángulo*/
    cmd_bitmap(tomate,50,70);
    ComColor(255,255,255);
    ComTXT(50,70-10, 21, OPT_CENTERY,"Tomate");

    /*Segundo rectángulo*/
    cmd_bitmap(girasol,190,70);
    ComColor(255,255,255);
    ComTXT(190,70-10, 21, OPT_CENTERY,"Girasol");
```

```
    /*Tercer rectángulo*/
    cmd_bitmap(lirio,50,160);
    ComColor(255,255,255);
    ComTXT(50,160-10, 21, OPT_CENTERY,"Lirio");

    /*Cuarto rectángulo*/
    cmd_bitmap(batata,190,160);
    ComColor(255,255,255);
    ComTXT(190,160-10, 21, OPT_CENTERY,"Batata");

    Dibuja();
}


void Dibuja_planta1()
{
    Nueva_pantalla(136,245,132);

    /* Rectánguo superior */
    ComColor(245,222,132);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(0,0);
    ComVertex2ff(320,50);
    Comando(CMD_END);
    ComColor(255,255,255);
    ComTXT(160,17, 23, OPT_CENTERX,"Tomates");

    ComColor(0,0,0);
    ComLineWidth(1);            //Grosor de la línea
    Comando(CMD_BEGIN_LINES);
    ComVertex2ff(0,50);
    ComVertex2ff(320,50);
    Comando(CMD_END);

    /* Botón de atrás */
    ComFgcolor(100,70,20);
    ComButton(15,10,40,30,20,0,"ATRAS");
```

```
/* Botón de Continuar */
ComFgcolor(100,70,20);
ComButton(265,10,40,30,20,0,"OK");

/* Riego */
ComColor(255,255,255);
ComTXT(80,65, 22, OPT_CENTERX,"Riego");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(20,90);
ComVertex2ff(140,135);
Comando(CMD_END);
Gota( 35,113,9);

/* Luz de sol */
ComColor(255,255,255);
ComTXT(240,65, 22, OPT_CENTERX,"Luz de sol");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(180,90);
ComVertex2ff(300,135);
Comando(CMD_END);
Sol(205,112,7,14);


/* Temperaturas */
ComColor(255,255,255);
ComTXT(80,150, 22, OPT_CENTERX,"Temperatura");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(20,175);
ComVertex2ff(140,220);
Comando(CMD_END);
RangoTemp( 80,202,86,30,20,30);


/* Fertilizante */
```

```
    ComColor(255,255,255);
    ComTXT(240,150, 22, OPT_CENTERX,"Fertilizante");
    ComColor(242,242,242);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(180,175);
    ComVertex2ff(300,220);
    Comando(CMD_END);
    ComColor(255,0,0);
    ComLineWidth(3);            //Grosor de la línea
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(204,185);
    ComVertex2ff(204,210);
    Comando(CMD_END);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(228,185);
    ComVertex2ff(228,210);
    Comando(CMD_END);

    Dibuja();

}



void Sol( int16_t x,int16_t y,int16_t r,int16_t r2)
{
    /* x: Indica la posición en el eje x del centro del sol.
     * y: Indica la posición en el eje Y del centro del sol.
     * r: Radio del sol.
     */
    int xfin,yfin;      //Punto en el sol.
    int k=0;            //variable auxiliar;
    float rad;          //Ángulo en radianes.

    // Dibujamos circunferencia sol.
    ComColor(250,219,96); //Definimos color.
    ComPointSize(r);      //Tamaño del punto
    Comando(CMD_BEGIN_POINTS);
    ComVertex2ff(x,y);  //Centro de la circunferencia.
```

```
    Comando(CMD_END);

    // Dibujamos las rayas del sol.
    for(k=0;k<360;k=k+60)
    {
        rad=k*3.14/180;
        xfin=x+(int)((r2)*cos(rad));
        yfin=y-(int)((r2)*sin(rad));

        ComColor(250,219,96);//Definimos otro color.

        ComLineWidth(1);            //Grosor de la línea
        Comando(CMD_BEGIN_LINES);
        ComVertex2ff(x,y);
        ComVertex2ff(xfin,yfin);
        Comando(CMD_END);
    }

}


void Gota( int16_t x,int16_t y,int16_t r)
{
    /* x: Indica la posición en el eje x del centro de la gota.
     * y: Indica la posición en el eje Y del centro dela gota.
     * r: Radio del sol.
     */
    int xfin1,yfin1,xfin2,yfin2;      //Punto en la gota.


    // Dibujamos circunferencia dela gota.
    ComColor(96,230,250);    //Definimos color.
    ComPointSize(r);    //Tamaño del punto
    Comando(CMD_BEGIN_POINTS);
    ComVertex2ff(x,y);  //Centro de la circunferencia.
    Comando(CMD_END);

    // Dibujamos parte superior de la gota
```

```
    xfin1=x-(int)round(r*cos(3.1416/6));
    xfin2=x-(int)round(r*cos(5*3.1416/6));
    yfin1=y-(int)round(r*sin(3.1416/6))+1;
    yfin2=y-(int)round(r*sin(5*3.1416/6))+1;

    ComColor(96,230,250);      //Lo que se pinte despues de esto va de este color
    ComLineWidth(1);           //Grosor de la línea
    Comando(CMD_BEGIN_LINES);
    ComVertex2ff(xfin1,yfin1);
    ComVertex2ff(x,(int)(y-1.5*r));
    Comando(CMD_END);
    Comando(CMD_BEGIN_LINES);
    ComVertex2ff(xfin2,yfin2);
    ComVertex2ff(x,(int)(y-1.5*r));
    Comando(CMD_END);

}

void Dibuja_planta2()
{
    Nueva_pantalla(136,245,132);

    /* Rectánguo superior */
    ComColor(245,222,132);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(0,0);
    ComVertex2ff(320,50);
    Comando(CMD_END);
    ComColor(255,255,255);
    ComTXT(160,17, 23, OPT_CENTERX,"Girasol");

    ComColor(0,0,0);
    ComLineWidth(1);            //Grosor de la línea
    Comando(CMD_BEGIN_LINES);
    ComVertex2ff(0,50);
    ComVertex2ff(320,50);
    Comando(CMD_END);
```

```
/* Botón de atrás */
ComFgcolor(100,70,20);
ComButton(15,10,40,30,20,0,"ATRAS");

/* Botón de Continuar */
ComFgcolor(100,70,20);
ComButton(265,10,40,30,20,0,"OK");

/* Riego */
ComColor(255,255,255);
ComTXT(80,65, 22, OPT_CENTERX,"Riego");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(20,90);
ComVertex2ff(140,135);
Comando(CMD_END);
Gota( 35,114,9);
Gota( 65,114,9);

/* Luz de sol */
ComColor(255,255,255);
ComTXT(240,65, 22, OPT_CENTERX,"Luz de sol");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(180,90);
ComVertex2ff(300,135);
Comando(CMD_END);
Sol(205,112,7,14);
Sol(240,112,7,14);
Sol(275,112,7,14);


/* Temperaturas */
ComColor(255,255,255);
ComTXT(80,150, 22, OPT_CENTERX,"Temperatura");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(20,175);
```

```
    ComVertex2ff(140,220);
    Comando(CMD_END);
    RangoTemp( 80,202,86,30,10,20);

    /* Fertilizante */
    ComColor(255,255,255);
    ComTXT(240,150, 22, OPT_CENTERX,"Fertilizante");
    ComColor(242,242,242);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(180,175);
    ComVertex2ff(300,220);
    Comando(CMD_END);
    ComColor(255,0,0);
    ComLineWidth(3);            //Grosor de la línea
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(204,185);
    ComVertex2ff(204,210);
    Comando(CMD_END);



    Dibuja();
}
void Dibuja_planta3()
{
    Nueva_pantalla(136,245,132);

    /* Rectánguo superior */
    ComColor(245,222,132);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(0,0);
    ComVertex2ff(320,50);
    Comando(CMD_END);
    ComColor(255,255,255);
    ComTXT(160,17, 23, OPT_CENTERX,"Lirios");

    ComColor(0,0,0);
    ComLineWidth(1);            //Grosor de la línea
    Comando(CMD_BEGIN_LINES);
```

```
ComVertex2ff(0,50);
ComVertex2ff(320,50);
Comando(CMD_END);

/* Botón de atrás */
ComFgcolor(100,70,20);
ComButton(15,10,40,30,20,0,"ATRAS");

/* Botón de Continuar */
ComFgcolor(100,70,20);
ComButton(265,10,40,30,20,0,"OK");

/* Riego */
ComColor(255,255,255);
ComTXT(80,65, 22, OPT_CENTERX,"Riego");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(20,90);
ComVertex2ff(140,135);
Comando(CMD_END);
Gota( 35,114,9);
Gota( 65,114,9);
Gota( 95,114,9);

/* Luz de sol */
ComColor(255,255,255);
ComTXT(240,65, 22, OPT_CENTERX,"Luz de sol");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(180,90);
ComVertex2ff(300,135);
Comando(CMD_END);
Sol(205,112,7,14);
Sol(240,112,7,14);



/* Temperaturas */
ComColor(255,255,255);
```

```
ComTXT(80,150, 22, OPT_CENTERX,"Temperatura");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(20,175);
ComVertex2ff(140,220);
Comando(CMD_END);
RangoTemp( 80,202,86,30,15,25);



/* Fertilizante */
ComColor(255,255,255);
ComTXT(240,150, 22, OPT_CENTERX,"Fertilizante");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(180,175);
ComVertex2ff(300,220);
Comando(CMD_END);

ComColor(255,0,0);
ComLineWidth(3);            //Grosor de la línea
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(204,185);
ComVertex2ff(204,210);
Comando(CMD_END);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(228,185);
ComVertex2ff(228,210);
Comando(CMD_END);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(252,185);
ComVertex2ff(252,210);
Comando(CMD_END);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(276,185);
ComVertex2ff(276,210);
Comando(CMD_END);

Dibuja();
```

```
}
void Dibuja_planta4()
{
    Nueva_pantalla(136,245,132);

    /* Rectánguo superior */
    ComColor(245,222,132);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(0,0);
    ComVertex2ff(320,50);
    Comando(CMD_END);
    ComColor(255,255,255);
    ComTXT(160,17, 23, OPT_CENTERX,"Batata");
    /* Botón de atrás */
    ComFgcolor(100,70,20);
    ComButton(15,10,40,30,20,0,"ATRAS");

    /* Botón de Continuar */
    ComFgcolor(100,70,20);
    ComButton(265,10,40,30,20,0,"OK");

    /* Riego */
    ComColor(255,255,255);
    ComTXT(80,65, 22, OPT_CENTERX,"Riego");
    ComColor(242,242,242);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(20,90);
    ComVertex2ff(140,135);
    Comando(CMD_END);
    Gota( 35,114,9);
    Gota( 65,114,9);
    Gota( 95,114,9);
    Gota( 125,114,9);



    /* Luz de sol */
    ComColor(255,255,255);
    ComTXT(240,65, 22, OPT_CENTERX,"Luz de sol");
```

```
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(180,90);
ComVertex2ff(300,135);
Comando(CMD_END);
Sol(205,112,7,14);
Sol(240,112,7,14);
Sol(275,112,7,14);



/* Temperaturas */
ComColor(255,255,255);
ComTXT(80,150, 22, OPT_CENTERX,"Temperatura");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(20,175);
ComVertex2ff(140,220);
Comando(CMD_END);
RangoTemp( 80,202,86,30,10,30);



/* Fertilizante */
ComColor(255,255,255);
ComTXT(240,150, 22, OPT_CENTERX,"Fertilizante");
ComColor(242,242,242);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(180,175);
ComVertex2ff(300,220);
Comando(CMD_END);
ComColor(255,0,0);
ComLineWidth(3);            //Grosor de la línea
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(204,185);
ComVertex2ff(204,210);
Comando(CMD_END);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(228,185);
ComVertex2ff(228,210);
```

```
    Comando(CMD_END);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(252,185);
    ComVertex2ff(252,210);
    Comando(CMD_END);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(276,185);
    ComVertex2ff(276,210);
    Comando(CMD_END);

    ComColor(0,0,0);
    ComLineWidth(1);            //Grosor de la línea
    Comando(CMD_BEGIN_LINES);
    ComVertex2ff(0,50);
    ComVertex2ff(320,50);
    Comando(CMD_END);

    Dibuja();
}
void Dibuja_M_automatico()
{
    Nueva_pantalla(136,245,132);
    /* Botón de atrás */
    if(pulsado_atras==0)
    {
        ComFgcolor(100,70,20);
        ComColor(0,0,0);
        ComButton(15,10,40,30,20,0,"ATRAS");
    }
    else
    {
        ComFgcolor(100,70,20);
        ComColor(255,255,255);
        ComButton(15,10,40,30,20,0,"ATRAS");
    }
    ComFgcolor(100,70,20);
    ComButton(15,10,40,30,20,0,"ATRAS");
```

```
ComColor(0,0,0);
ComTXT(160,30, 21, OPT_CENTERX,"MODO AUTOMATICO ON");
ComColor(100,100,100);
ComLineWidth(2);

/*Primer rectángulo*/
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(30,70);
ComVertex2ff(150,130);
Comando(CMD_END);//Se deja de escribir lo anterior
ComColor(0,255,0);
DibujaDeposito(125,100,35,50,0.25);
ComTXT(90,90, 24, OPT_CENTERX,"%");

/*Segundo rectángulo*/
ComColor(100,20,90);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(170,70);
ComVertex2ff(290,130);
Comando(CMD_END);
ComColor(0,255,0);
// ponemos la hora
diff=restafecha(horariego,temp);
sprintf(buffer,"%d min : %d seg",diff.min,diff.seg);
ComTXT(230,90, 22, OPT_CENTERX,buffer);

/*Rectángulo inferior*/
ComColor(0,0,0);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(30,150);
ComVertex2ff(290,230);
Comando(CMD_END);
ComColor(0,255,0);
ComTXT(80,160, 23, OPT_CENTERX,"Toldo");

if(var_paso==0)
{
    ComTXT(80,190, 23, OPT_CENTERX,"OFF");
```

```
    }
    else
    {
        ComTXT(80,190, 23, OPT_CENTERX,"ON");
    }

}
void Dibuja_pantalla_manual()
{
    Nueva_pantalla(136,245,132);
    //Negro
    ComColor(0,0,0);//Lo que se pinte despues de esto va de este color
    ComTXT(160,60, 24, OPT_CENTERX,"MODO MANUAL");

    /* Botón de atrás */
    if(pulsado_atras==0)
    {
        ComFgcolor(100,70,20);
        ComButton(265,15,40,30,20,0,"ATRAS");
    }
    else
    {
        ComFgcolor(100,70,20);
        ComColor(255,255,255);
        ComButton(265,15,40,30,20,0,"ATRAS");
    }
    /* Botón de información */
    if(pulsado_info==0)
    {
        ComFgcolor(130,20,50);
        ComColor(0,0,0);
        ComButton(135,175,50,40,21,0,"INFO");
    }
    else
    {
        ComFgcolor(130,20,50);
        ComColor(255,255,255);
        ComButton(135,175,50,40,21,0,"INFO");
```

```
}
/* Agua deseada */
ComColor(100,150,80);
Comando(CMD_BEGIN_RECTS);//Se escribe cada vez que vayamos a escribir rectangulos.
ComVertex2ff(30,125);
ComVertex2ff(120,200);
Comando(CMD_END);//Se deja de escribir lo anterior
if(pulsado_riego==0){
    ComColor(0,0,0);
    ComTXT(76,130, 21, OPT_CENTERX,"BOMBA");
    ComButton(50,150,50,40,22,0,"OFF");
}
else
{
    ComColor(0,0,0);
    ComTXT(76,130, 21, OPT_CENTERX,"BOMBA");
    ComColor(255,255,255);
    ComButton(50,150,50,40,22,0,"ON");
}
/* TOLDO */
ComColor(100,150,80);
Comando(CMD_BEGIN_RECTS);//Se escribe cada vez que vayamos a escribir rectangulos.
ComVertex2ff(200,125);
ComVertex2ff(310,200);
Comando(CMD_END);//Se deja de escribir lo anterior
ComColor(0,0,0);
ComTXT(255,130, 21, OPT_CENTERX,"TOLDO");
if(pulsado_toldo==0)
{
    ComButton(210,150,40,40,24,0,"-");
    ComButton(260,150,40,40,24,0,"+");
}
if(pulsado_toldo==1)
{
    ComColor(255,255,255);
    ComButton(210,150,40,40,24,0,"-");
    ComColor(0,0,0);
    ComButton(260,150,40,40,24,0,"+");
```

```
    }
    if(pulsado_toldo==-1)
    {
        ComColor(0,0,0);
        ComButton(210,150,40,40,24,0,"-");
        ComColor(255,255,255);
        ComButton(260,150,40,40,24,0,"+");
    }

    Dibuja();
}
void Dibuja_pantalla_medidas()
{
    Nueva_pantalla(136,245,132);
    ComColor(200,200,200);
    Comando(CMD_BEGIN_RECTS);//Se escribe cada vez que vayamos a escribir rectangulos.
    ComVertex2ff(120,24);
    ComVertex2ff(200,54);
    Comando(CMD_END);//Se deja de escribir lo anterior
    ComColor(0,0,0);//Lo que se pinte despues de esto va de este color
    ComTXT(160,30, 22, OPT_CENTERX,"MEDIDAS");

    ComColor(255,50,50);
    Comando(CMD_BEGIN_RECTS);//Se escribe cada vez que vayamos a escribir rectangulos.
    ComVertex2ff(135,60);
    ComVertex2ff(185,110);
    Comando(CMD_END);//Se deja de escribir lo anterior
    ComColor(255,0,0);
    ComTXT(80,80, 22, OPT_CENTERX,"Humedad");

    ComColor(255,50,50);
    Comando(CMD_BEGIN_RECTS);//Se escribe cada vez que vayamos a escribir rectangulos.
    ComVertex2ff(135,120);
    ComVertex2ff(185,170);
    Comando(CMD_END);//Se deja de escribir lo anterior
    ComColor(255,0,0);
    ComTXT(80,135, 22, OPT_CENTERX,"Presion");
```

```
ComColor(255,50,50);
Comando(CMD_BEGIN_RECTS);//Se escribe cada vez que vayamos a escribir rectangulos.
ComVertex2ff(135,180);
ComVertex2ff(185,230);
Comando(CMD_END);//Se deja de escribir lo anterior
ComColor(255,0,0);
ComTXT(70,195, 22, OPT_CENTERX,"Luminosidad");

/* Botón de inicio */
if(pulsado_atras==0)
{
    ComColor(255,255,255);
    ComFgcolor(100,70,20);
    ComButton(265,15,50,40,20,0,"INICIO");
}
else
{
    ComColor(255,255,255);
    ComFgcolor(100,70,20);
    ComColor(0,0,0);

    ComButton(265,15,50,40,20,0,"INICIO");
}
/* Botón de modo manual */
if(pulsado_manual==0)
{
    ComColor(255,255,255);
    ComFgcolor(100,70,20);
    ComButton(10,15,50,40,20,0,"MANUAL");
}
else
{
    ComColor(255,255,255);
    ComFgcolor(100,70,20);
    ComColor(0,0,0);

    ComButton(10,15,50,40,20,0,"MANUAL");
}
```

```
}

void ComTermometro( int16_t x,int16_t y,int16_t longitud,int16_t ancho,uint16_t major,uint16_t minor,float val)

{

    /*

     * x: indica la posición en el eje x en el que colocaremos la posición inferior del termómetro.

     * y: indica la posición en el eje y en el que colocaremos la posición inferior del termómetro.

     * longitud: indica la longitud del termómetro.

     * ancho: indica la anchura del termómetro.

     * major: Valor máximo de temperatura.

     * minor: Valor mínimo de temperatura.

     * val: Valor actual de temperatura.

     */

    int aux,aux1,aux2; //Variable auxiliar.

    int L,i;           //Variables utilizadas para la longitud de la temperatura en el termometro y variable auxiliar respectivamente.

    int m,n;           //Variables auxiliares.

    if(val<minor)

    {

        val=minor;

    }

    if(val>major)

    {

        val=major;

    }

    //Valor de la coordenada y para la temperatura máxima.

    int Y_Tmax=y-longitud;

    //Valor de la coordenada y para la temperatura mínima.

    int Y_Tmin=y-ancho;

    //Pendiente de la recta que nos permitirá conocer el valor de la componente y en la pantalla en función de la temperatura.

    m=(int)round((Y_Tmax-Y_Tmin)/(major-minor));

    //Ordenada en el origen.

    n=Y_Tmin-m*minor;

    //Valor hasta donde debe de llegar el rectángulo que representa la temperatura.

    L=(int)round(m*val+n);

    //variables auxiliares.

    aux=(int)round(x+ancho/2);
```

```
aux2=(int)round(x-ancho/2);
aux1=(int)round(y-longitud);
ComColor(255,255,255);//Pintamos un rectángulo relleno blanco
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(aux,y);
ComVertex2ff(aux2,aux1);
Comando(CMD_END);

//Pintamos la circunferencia inferior del termómetro
ComColor(255,255,255);
ComPointSize(ancho);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(x,y);
Comando(CMD_END);

//Pintamos la circunferencia superior del termómetro
ComColor(255,255,255);
aux=(int)round(ancho/2)+1;
aux1=y-longitud;
ComPointSize(aux);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(x,aux1);
Comando(CMD_END);

//Pintamos la circunferencia inferior de la temperatura de rojo
ComColor(255,0,0);
aux=ancho-5;
ComPointSize(aux);
Comando(CMD_BEGIN_POINTS);
ComVertex2ff(x,y);
Comando(CMD_END);

//Pintamos el rectángulo rojo en función de la temperatura
aux=(int)round(x+ancho/2)-5;
aux1=y-ancho/2;
ComColor(255,0,0);
Comando(CMD_BEGIN_RECTS);
ComVertex2ff(aux,aux1);
```

```
aux=(int)round(x-ancho/2)+5;
ComVertex2ff(aux,L);
Comando(CMD_END);

aux=(int)round(x+ancho);
ComColor(0,0,0);
ComTXT(x,y-longitud-30, 21, OPT_CENTERX,"Ta");
ComTXT(aux+5,Y_Tmax-5, 20, OPT_CENTERX,"30 C");
ComTXT(aux+5,Y_Tmin-5, 20, OPT_CENTERX,"10 C");
ComColor(0,0,0);      //Lo que se pinte despues de esto va de este color
ComLineWidth(1);            //Grosor de la línea
Comando(CMD_BEGIN_LINES);
ComVertex2ff(x+10,Y_Tmax);
ComVertex2ff(x+14,Y_Tmax);
Comando(CMD_END);
ComColor(0,0,0);      //Lo que se pinte despues de esto va de este color
ComLineWidth(1);            //Grosor de la línea
Comando(CMD_BEGIN_LINES);
ComVertex2ff(x+10,Y_Tmin);
ComVertex2ff(x+14,Y_Tmin);
Comando(CMD_END);

//Bucle for para representar las rayas en el termómetro distanciadas en 5 ºC.
for(i=20;i<30;i=i+5)
{
    aux=(int)round(x+ancho);
    L=(int)round(m*i+n);
    ComColor(0,0,0);
    ComNum(aux,L-5, 20, OPT_CENTERX,i );
    ComTXT(aux+10,L-5, 20, OPT_CENTERX,"C");
    ComColor(0,0,0);      //Lo que se pinte despues de esto va de este color
    ComLineWidth(1);            //Grosor de la línea
    Comando(CMD_BEGIN_LINES);
    ComVertex2ff(x+10,L);
    ComVertex2ff(x+14,L);
    Comando(CMD_END);
}
```

```
}
void RangoTemp( int16_t x,int16_t y,int16_t longitud,int16_t ancho,uint16_t minor,uint16_t major)
{
    /*
     * x: Coordenada x central.
     * y: Coordenada y central.
     * longitud: longitud del medidor.
     * ancho.
     * minor: Temperatura menor capaz de aguantar dicha planta.
     * major: Temperatura máxima capaz de aguantar la planta.
     */
    int V_minimo,V_maximo;
    /*En primer lugar pintamos los dos puntos con centros en (x,y)+-(L/2,0)*/
    ComColor(249,242,46);
    ComPointSize((int)round(ancho/2));
    Comando(CMD_BEGIN_POINTS);
    ComVertex2ff((int)round(x-longitud/2),y);
    Comando(CMD_END);
    Comando(CMD_BEGIN_POINTS);
    ComVertex2ff((int)round(x+longitud/2),y);
    Comando(CMD_END);

    /* Pintamos el rectangulo exterior */
    ComColor(249,242,46);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff((int)round(x-longitud/2),y-(int)round(ancho/2));
    ComVertex2ff((int)round(x+longitud/2),y+(int)round(ancho/2));
    Comando(CMD_END);

    /* Pintamos el rectangulo en función de la temperatura */
    V_minimo=(int)round(x-longitud/2)+(int)round(minor*longitud/40);
    V_maximo=(int)round(x-longitud/2)+(int)round(major*longitud/40);
    ComColor(200,0,0);

    ComNum(V_minimo,y-(int)round(ancho/2)-13, 20, OPT_CENTERX,minor );
    ComTXT(V_minimo+10,y-(int)round(ancho/2)-13, 20, OPT_CENTERX,"C");
```

```
    ComNum(V_maximo,y-(int)round(ancho/2)-13, 20, OPT_CENTERX,major );
    ComTXT(V_maximo+10,y-(int)round(ancho/2)-13, 20, OPT_CENTERX,"C");

    ComTXT((int)round(x-longitud/2)-5,y-(int)round(ancho/2)-13,   20,   OPT_CENTERX,"0
C");
    ComTXT((int)round(x+longitud/2)+5,y-(int)round(ancho/2)-13,   20,   OPT_CENTERX,"40
C");



    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff(V_minimo,y-(int)round(ancho/2));
    ComVertex2ff(V_maximo,y+(int)round(ancho/2));
    Comando(CMD_END);


}


void MedidorLuminosidad( int16_t x,int16_t y,int16_t R,int16_t dif,float val,uint16_t
Red,uint16_t G,uint16_t B )
{
    /*
     * X: Coordenada eje x
     * Y: Coordenada eje y
     * R: Radio exterior
     * dif: Diferencia entre Radio interior y exterior
     * val: tanto por uno
     * RGB del color donde se va a dibujar el diagrama.
     */


    float angulo;
    float i;
    struct punto
    {
        int x;
        int y;
    };
    struct punto Pint;
    struct punto Pext;


    //Pintamos la exterior
```

```
    ComColor(255,255,255);
    ComPointSize(R);
    Comando(CMD_BEGIN_POINTS);
    ComVertex2ff(x,y);
    Comando(CMD_END);

    /* A contnuación calculamos el angulo en funcion del tanto por 1. */
    angulo=3.14/2-val*2*3.14;
    for(i=3.14/2;i>angulo;i=i-0.1)
    {
        Pint.x=x+(int)((R-dif)*cos(i));
        Pext.x=x+(int)(R*cos(i));
        Pint.y=y-(int)((R-dif)*sin(i));
        Pext.y=y-(int)(R*sin(i));
        ComColor(200,100,0);
        ComLineWidth(2);           //Grosor de la línea
        Comando(CMD_BEGIN_LINES);
        ComVertex2ff(Pint.x,Pint.y);
        ComVertex2ff(Pext.x,Pext.y);
        Comando(CMD_END);
    }
    //Pintamos la circunferencia inferior de la temperatura de rojo
    ComColor(Red,G,B);
    ComPointSize(R-dif);
    Comando(CMD_BEGIN_POINTS);
    ComVertex2ff(x,y);
    Comando(CMD_END);
}

void DibujaDeposito(int16_t x,int16_t y,int16_t ancho,int16_t alto,float val)
{
    /*
     * x:Posición eje x del centro del depósito
     * y:Posición eje y del centro del depósito.
     * ancho: Ancho del depósito
     * alto: Altura del depósito
     * val: Tanto por uno del volumen del depósito disponible.
     * */
```

```
    /* Pintamos el rectangulo exterior */
    int altura;
    ComColor(0,0,0);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff((int)round(x-ancho/2),y-(int)round(alto/2));
    ComVertex2ff((int)round(x+ancho/2),y+(int)round(alto/2));
    Comando(CMD_END);
    /* Pintamos el rectángulo interior */
    ComColor(255,255,255);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff((int)round(2+x-ancho/2),(int)round(2+y-alto/2));
    ComVertex2ff((int)round(x-2+ancho/2),(int)round(y-2+alto/2));
    Comando(CMD_END);

    /* Pintamos el rectangulo en función del volumen */
    altura=(int)round(y+alto/2-val*alto);
    ComColor(0,0,200);
    Comando(CMD_BEGIN_RECTS);
    ComVertex2ff((int)round(2+x-ancho/2),altura);
    ComVertex2ff((int)round(x-2+ancho/2),(int)round(y-2+alto/2));
    Comando(CMD_END);

    ComColor(200,0,0);
    ComNum(x-(int)round(ancho/2)-53,y-15, 25, OPT_CENTERX,(int)round(val*100) );


}


//  ---------------------------- Devuelve la hora del RTC ------
struct mtime rtc_readtime()
{
    struct mtime st;
    char raw,ss,mm,AMPM,hh;
    // segundos
    raw = I2C_read8(RTC_SLAVE_ADDRESS,SEG_REG);
    ss=(raw/16)*10+raw%16;
    // minutos
```

```
    raw = I2C_read8(RTC_SLAVE_ADDRESS,MIN_REG);
    mm=(raw/16)*10+raw%16;
    // horas
    raw = I2C_read8(RTC_SLAVE_ADDRESS,HOUR_REG);
    AMPM=bitn(raw,6); // si es am o pm
    hh=raw%16;
    hh=bitn(raw,4)*10+hh;
    if(AMPM==0)
        hh=hh+bitn(raw,5)*2;
    // lo pasamos a struct
    st.seg=ss;
    st.min=mm;
    st.hour=hh;

    return st;
}

//  ----------------------------- Devuelve el bit concreto ------
char bitn(char num, char numbit)
{
    char bitn;
    bitn=(num>>(numbit-1))&0x01;
    return bitn;
}



//  ----------------------------- lectura 1byte I2C -------------
int I2C_read8(unsigned int slaveAdr, unsigned char writeByte)
{
    volatile int val = 0;
    volatile int valScratch = 0;

    I2CMasterSlaveAddrSet(I2C0_BASE, slaveAdr, 0);  //Modo ESCRITURA
    I2CMasterDataPut(I2C0_BASE, writeByte);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START); //esc. Multiple
    Espera_I2C(I2C0_BASE);
    I2CMasterSlaveAddrSet(I2C0_BASE, slaveAdr, 1);  //MODO LECTURA
    //--- Hemos cambiado a single
```

```
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE); //Lect. Multiple
    Espera_I2C(I2C0_BASE);
    val=I2CMasterDataGet(I2C0_BASE);
    return val;
}

//  ----------------------------- escritura 1byte I2C -----------
void I2C_write8 (unsigned char address, unsigned char writeByte)
{
    I2CMasterDataPut(I2C0_BASE, address);
    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);
    Espera_I2C(I2C0_BASE);
    I2CMasterDataPut(I2C0_BASE,writeByte);
    I2CMasterControl(I2C0_BASE,I2C_MASTER_CMD_BURST_SEND_FINISH);
    Espera_I2C(I2C0_BASE);
}

//  ----------------------------- Inicia el I2C -----------------
void inicia_i2c()
{
    // Iniciamos el periférico
    SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
    //
    // Enable pin PB2 for I2C0 I2C0SCL
    //
    GPIOPinConfigure(GPIO_PB2_I2C0SCL);
    GPIOPinTypeI2CSCL(GPIO_PORTB_BASE, GPIO_PIN_2);
    //
    // Enable pin PB3 for I2C0 I2C0SDA
    //
    GPIOPinConfigure(GPIO_PB3_I2C0SDA);
    GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_3);
    I2CMasterInitExpClk(I2C0_BASE, RELOJ, false );
    I2CMasterEnable(I2C0_BASE);
}

// ---------------------------- inicializa la UART --------------
```

```
void inicia_UART()
{
    // Initialize the UART.
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    UARTStdioConfig(0, 115200, RELOJ);
}

// ------------ Riega la planta con el agua necesaria en (L) --------------
void riego(char ag_nec)
{
    unsigned int ts_riego;
    struct mtime obj,diff,temp;
    int max,val=0;
    float porcent;
    char cond;
    char seg_obj,min_obj,min,seg,seg_n,min_n;
    // calculamos el tiempo de riego
    ts_riego=(unsigned int)(ag_nec/caudal);     // caudal en l/s
    max=ts_riego;
    if (ts_riego>=60)
    {
        min = ts_riego/60;
        seg = ts_riego%60;
    }
    else
    {
        min = 0;
        seg = ts_riego;
    }

    // regamos el tiempo necesario
    temp=rtc_readtime();
    seg_obj=temp.seg+seg;
```

```
    min_obj=temp.min+min;

    if(seg_obj>=60)
    {
        seg_obj = seg_obj - 60;
        min_obj++;
    }

    // iniciamos el riego
    RELE_ON
    sprintf(buffer,ANSI_COLOR_GREEN "RELE_ON" ANSI_COLOR_RESET "\n\n\r");
    UARTprintf(buffer);
    obj.seg=seg_obj;
    obj.min=min_obj;

    // mientras estemos regando...
    do
    {
        temp=rtc_readtime();
        seg_n=temp.seg;
        min_n=temp.min;

        if(seg_n%2)
        {
            diff=restafecha(obj,temp);
            sprintf(buffer,"Quedan      %d      min      :      %d      seg      de
riego.......\n\r",diff.min,diff.seg);
            UARTprintf(buffer);
        }

        // Pintamos la barra de progreso
        Nueva_pantalla(100,100,100);
        ComColor(0x00,200,0x00);
        cmd_progress(xprog,YPANT/2,280-xprog,12,0,val,max);
        porcent=(100*val/max);
        sprintf(bufplaca,"Regando...(%d %c)",(int)porcent,'%');
        ComColor(220,220,220);
        ComTXT(XPANT/2,YPANT/2-30, 22, OPT_CENTERX,bufplaca);
```

```
        Dibuja();

        // esperamos 1 segundo
        while(cambia==0);
        cambia=0;
        val++; // ha pasado un segundo

        // verificamos si se cumple la condicion
        if (seg_n>=seg_obj && min_n==min_obj)
            cond=1;
        else
            cond=0;

    }while(!cond);

    // aqui ya ha pasado el tiempo necesario
    sprintf(buffer,ANSI_COLOR_RED "RELE_OFF" ANSI_COLOR_RESET "\n\n\r");
    UARTprintf(buffer);
    RELE_OFF
}



// ---------------------------- Resta fechas de calendario --------------
struct mtime restafecha(struct mtime obj,struct mtime act)
{
    struct mtime diff;
    char seg_obj,min_obj,diff_s,diff_m,seg_n,min_n;

    seg_obj = obj.seg;
    seg_n   = act.seg;
    min_obj = obj.min;
    min_n   = act.min;
    sprintf(buffer,"objetivo||actual             -->             %d:%d             ||
%d:%d\n\r",min_obj,seg_obj,min_n,seg_n);
    UARTprintf(buffer);

    if(seg_obj<seg_n)
    {
```

```
        diff_s=60+seg_obj-seg_n;
        diff_m=min_obj-(min_n+1);
    }
    else
    {
        diff_s=seg_obj-seg_n;
        diff_m=min_obj-min_n;
    }

    diff.seg=diff_s;
    diff.min=diff_m;

    return diff;
}



// ---------------------------- Lee los sensores --------------
void leesensores()
{
    // obtenemos las medidas
    lux=OPT3001_getLux();
    lux_i=(int)round(lux);
    sensorTmp007Read(&T_amb, &T_obj);
    sensorTmp007Convert(T_amb, T_obj, &Tf_obj, &Tf_amb);
    T_amb_i=(short)round(Tf_amb);
    T_obj_i=(short)round(Tf_obj);
    returnRslt      =      bme280_read_pressure_temperature_humidity(&g_u32ActualPress,
&g_s32ActualTemp, &g_u32ActualHumity);
    T_act=(float)g_s32ActualTemp/100.0;
    P_act=(float)g_u32ActualPress/100.0;
    H_act=(float)g_u32ActualHumity/1000.0;

    /* Omitimos la lectura del resto
            bmi160_bmm150_mag_compensate_xyz(&s_magcompXYZ);
            bmi160_read_accel_xyz(&s_accelXYZ);
            bmi160_read_gyro_xyz(&s_gyroXYZ);
     */
```

```
    // imprimimos las medidas por la UART
    // imprimemedidasUART();
    sprintf(string, "BME: T:%.2f C  P:%.2fmbar  H:%.3f % \n\r",T_act,P_act,H_act);
    UARTprintf(string);
    UARTprintf("\n\r");
}



// ---------------------------- Inicia los sensores  --------------
void minicia_sensores(int numbooster, uint32_t RELOJ, uint32_t fs_Hz, int bool_UART)
{
    // esta función inicia los sensores
    //con la variable bool_UART decimos si queremos comunicacion con la uart  o no
    uint32_t Period; // periodo de medicion

    Conf_Boosterpack(numbooster, RELOJ);

    // TIMER 1
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerClockSourceSet(TIMER0_BASE, TIMER_CLOCK_SYSTEM); // 120 MHz
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
    Period = (uint32_t) RELOJ/fs_Hz ;    //Calculamos el periodo
    TimerLoadSet(TIMER0_BASE, TIMER_A, Period -1);
    TimerIntRegister(TIMER0_BASE,   TIMER_A,Timer0IntHandler);   //   habilitamos   una
interupcion
    IntEnable(INT_TIMER0A);
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();
    TimerEnable(TIMER0_BASE, TIMER_A); // habilitamos el timerA

    if (bool_UART==1)
    {
        UARTprintf("\033[2J \033[1;1H Inicializando OPT3001... ");
        OPT3001_init();
        UARTprintf("Hecho!\n");
        UARTprintf("Leyendo DevID... ");
        DevID=OPT3001_readDeviceId();
```

```
        UARTprintf("DevID= 0X%x \n", DevID);
        UARTprintf("Inicializando ahora el TMP007...");
        sensorTmp007Init();
        UARTprintf("Hecho! \nLeyendo DevID... ");
        DevID=sensorTmp007DevID();
        UARTprintf("DevID= 0X%x \n", DevID);
        sensorTmp007Enable(true);
        UARTprintf("Inicializando BME280... ");
        bme280_data_readout_template();
        bme280_set_power_mode(BME280_NORMAL_MODE);
        UARTprintf("Hecho! \nLeyendo DevID... ");
        readI2C(BME280_I2C_ADDRESS2,BME280_CHIP_ID_REG, &DevID, 1);
        UARTprintf("DevID= 0X%x \n", DevID);
        UARTprintf("Inicializando BMI160, modo NAVIGATION... ");
        bmi160_initialize_sensor();
        bmi160_config_running_mode(APPLICATION_NAVIGATION);
        //bmi160_accel_foc_trigger_xyz(0x03, 0x03, 0x01, &accel_off_x, &accel_off_y,
&accel_off_z);
        //bmi160_set_foc_gyro_enable(0x01, &gyro_off_x, &gyro_off_y, &gyro_off_z);
        UARTprintf("Hecho! \nLeyendo DevID... ");
        readI2C(BMI160_I2C_ADDR2,BMI160_USER_CHIP_ID_ADDR, &DevID, 1);
        UARTprintf("DevID= 0X%x \n\n\n\n", DevID);
    }
    else
    {
        OPT3001_init();
        DevID=OPT3001_readDeviceId();
        sensorTmp007Init();
        DevID=sensorTmp007DevID();
        sensorTmp007Enable(true);
        bme280_data_readout_template();
        bme280_set_power_mode(BME280_NORMAL_MODE);
        bmi160_initialize_sensor();
        bmi160_config_running_mode(APPLICATION_NAVIGATION);
        readI2C(BME280_I2C_ADDRESS2,BME280_CHIP_ID_REG, &DevID, 1);
    }
}
```

```
// --------------------------- Imprime medidas UART  --------------

void imprimemedidasUART()

{

    UARTprintf("\033[10;1H--------------------------------------\n");

    sprintf(string,"  OPT3001: %.3f Lux\n",lux);

    UARTprintf(string);

    UARTprintf("--------------------------------------\n");

    sprintf(string,"  TMP007:  T_a:%.3f, T_o:%.3f \n", Tf_amb, Tf_obj);

    UARTprintf(string);

    UARTprintf("--------------------------------------\n");

    sprintf(string, "\nBME: T:%.2f C  P:%.2fmbar  H:%.3f % \n\r",T_act,P_act,H_act);

    UARTprintf(string);

    UARTprintf("--------------------------------------\n\r");

    sprintf(string,     "          BMM:          X:%6d\033[17;22HY:%6d\033[17;35HZ:%6d
\n",s_magcompXYZ.x,s_magcompXYZ.y,s_magcompXYZ.z);

    UARTprintf(string);

    UARTprintf("--------------------------------------\n");

    sprintf(string,      "            ACCL:      X:%6d\033[19;22HY:%6d\033[19;35HZ:%6d
\n",s_accelXYZ.x,s_accelXYZ.y,s_accelXYZ.z);

    UARTprintf(string);

    UARTprintf("--------------------------------------\n");

    sprintf(string,      "            GYRO:      X:%6d\033[21;22HY:%6d\033[21;35HZ:%6d
\n",s_gyroXYZ.x,s_gyroXYZ.y,s_gyroXYZ.z);

    UARTprintf(string);

    UARTprintf("--------------------------------------\n");

}


// --------------------------- Interrupcion sensores  --------------

void Timer0IntHandler(void)

{

    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT); //Borra flag

    cambia=1;

}


// --------------------------- Inicialización Pantalla  --------------

void minicia_pantalla(int numbooster,uint32_t RELOJ) // recibe el numero de boosterpack
y la frecuencia de reloj

{
```

```
    // esta función a parte de iniciar la pantalla, nos muestra una pantalla de inicio
    int i;
    HAL_Init_SPI(numbooster, RELOJ);  //Boosterpack a usar, Velocidad del MC
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0 |GPIO_PIN_1);
    Inicia_pantalla();
    SysCtlDelay(RELOJ/3); // espera

    for(i=0;i<6;i++)    Esc_Reg(REG_TOUCH_TRANSFORM_A+4*i, REG_CAL[i]);
}



// ---------------------------- Dibuja Circulos  --------------
void drawcircle(int r,int Xp,int Yp)
{
    ComPointSize(r);//TAMAÑO DEL PUNTO/CIRCULO
    Comando(CMD_BEGIN_POINTS);
    ComVertex2ff(Xp,Yp);
    Comando(CMD_END);
}


// ---------------------------- Cambia color del fondo  --------------
void pintafondo(char R, char G, char B)
{
    Comando(CMD_BEGIN_RECTS);
    ComColor(R,G,B);
    ComVertex2ff(12,12);
    ComVertex2ff(308,228);
    Comando(CMD_END);
}


// ---------------------------- Dibuja barra de progreso  --------------
void  cmd_progress(  int16_t  x,int16_t  y,int16_t  w,int16_t  h,uint16_t  ops,uint16_t
val,uint16_t range)
{
    EscribeRam32(CMD_PROGRESS);
    EscribeRam16(x);
    EscribeRam16(y);
```

```
    EscribeRam16(w);
    EscribeRam16(h);
    EscribeRam16(ops);
    EscribeRam16(val);
    EscribeRam16(range);
    EscribeRam16(0);
}

// ---------------------------- Dibuja Imagen guardada en rawdata.h  --------------
void cmd_bitmap(char num, unsigned int X, unsigned int Y)
{
    // ------------ INICIALIZACION ---------------
    Comando(BITMAP_HANDLE(num));
    Comando(BITMAP_SOURCE(RAM_G+RawData_Header[num].Arrayoffset));    //specify    the
starting address of the bitmap in graphics RAM

Comando(BITMAP_LAYOUT(RawData_Header[num].Format,RawData_Header[num].Stride,RawData_H
eader[num].Height));

Comando(BITMAP_SIZE(NEAREST,BORDER,BORDER,RawData_Header[num].Width,RawData_Header[nu
m].Height));

    // ------------ BIBUJAMOS BITMAPS ------------
    Comando(COLOR_RGB(255,255,255));
    Comando(BEGIN(BITMAPS));                // comenzamos el bitmap
    Comando(VERTEX2II(X,Y,num,0));          // handle específico
    Comando(CMD_END);
}
// ---------------------------- Carga las imágenes en la RAM gráfica  --------------
void loadIm_toRAM()
{

    uint32_t bytes;
    unsigned int i;

    // TOMATE
    bytes=RawData_Header[0].Stride*RawData_Header[0].Height;
    for(i=0;i<bytes;i++)
        EscribeRam8_alb(tomate_RawData[i]);
```

```
    // GIRASOL
    bytes=RawData_Header[1].Stride*RawData_Header[1].Height;
    for(i=0;i<bytes;i++)
        EscribeRam8_alb(girasol_RawData[i]);

    // LIRIO
    bytes=RawData_Header[2].Stride*RawData_Header[2].Height;
    for(i=0;i<bytes;i++)
        EscribeRam8_alb(lirio_RawData[i]);

    // BATATA
    bytes=RawData_Header[3].Stride*RawData_Header[3].Height;
    for(i=0;i<bytes;i++)
        EscribeRam8_alb(batata_RawData[i]);

    // INICIO
    bytes=RawData_Header[4].Stride*RawData_Header[4].Height;
    for(i=0;i<bytes;i++)
        EscribeRam8_alb(huerto_RawData[i]);

}
// ---------------------------- Escribe en la RAM gráfica  --------------
void EscribeRam8_alb(char dato)
{
    HAL_SPI_CSLow();
    FT800_SPI_SendAddressWR(RAM_G+RAMG_Offset);
    FT800_SPI_Write8(dato);
    HAL_SPI_CSHigh();

    RAMG_Offset = RAMG_Offset+1;

    // desbordamiento
    if(RAMG_Offset > 262143)
        RAMG_Offset = (RAMG_Offset - 262144);

}
```