2018

SAI

Informe

Materia: Técnicas Digitales II - Ing. Electrónica

Profesor: Ing. Adrian Laiuppa

Ayudante: Ing. Germán Feres

Integrantes: Alvarez Diego - Loidi Marcos

UTN FRBB

Índice

1. Descripción General

El SAI (Sistema automatizado de control de invernadero), es un sistema que controla humedad y temperatura de un invernadero. Los valores objetivo de temperatura y humedad son diferentes dependiendo del cultivo que se plante y la etapa de crecimiento en el que se encuentre. Para ello se controla una bomba de agua, un ventilador, un calentador y una ventana corrediza. Para realizar este control se toman datos de temperatura externa al invernadero, temperatura interna, humedad de suelo y disponibilidad de agua en el tanque.

Posee interfaz de usuario a través de 5 botones y una pantalla LCD, además de contar con la posibilidad de controlarlo mediante puerto serie con una PC.

1.1 Alimentación

El SAI consta de dos placas tipo *shield* para adosar a la placa discovery STM32F407. En una de estas placas se encuentra un *jack* de alimentación donde se debe conectar una fuente de 9V. Para alimentar la segunda placa se debe conectar un cable plano (incluido) de dos pines a la primera placa. Por último, la placa Discovery debe estar conectada a una fuente de alimentación de 5V.

1.2 Conexiones

Como se dijo anteriormente las placas poseen pines para conectar los distintos sensores. En la figura 1.1 se muestra la placa 1 donde se puede ver la pantalla LCD, los botones, pila para mantener la hora y la ranura de la tarjeta SD. En la figura 1.2 se muestra la placa 2, la cual posee la conexión para el sensor de temperatura exterior e interior, el sensor de luz, el sensor de humedad del suelo, los relés de bomba y calentador, el ventilador y el RS232 para comunicación serial.

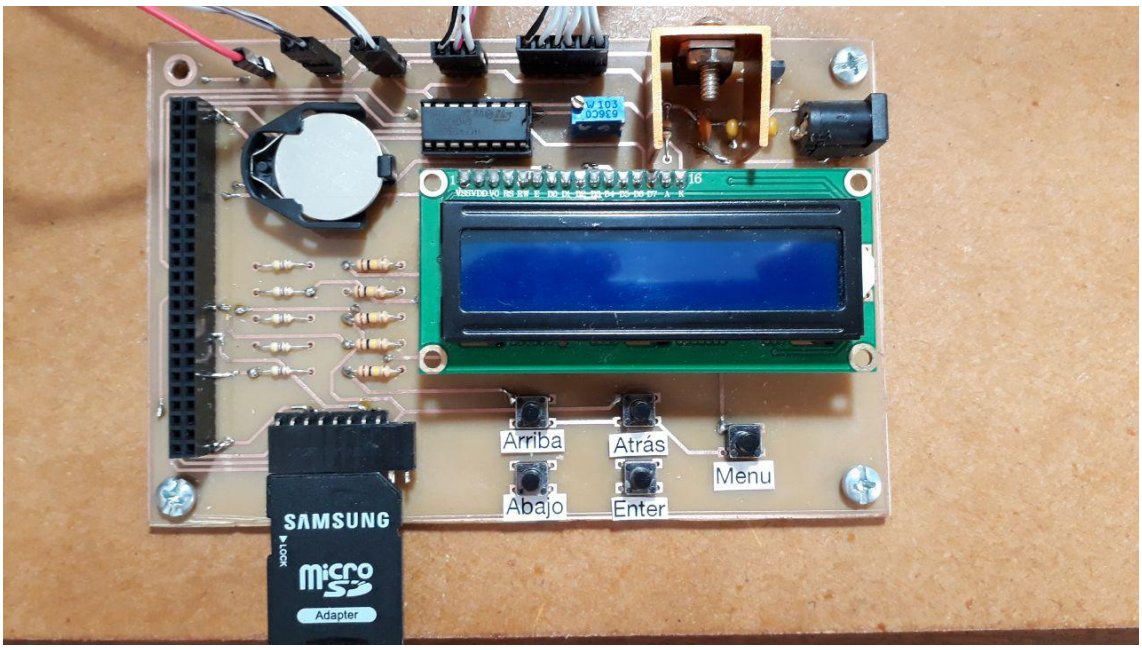


Figura 1.1 Placa 1

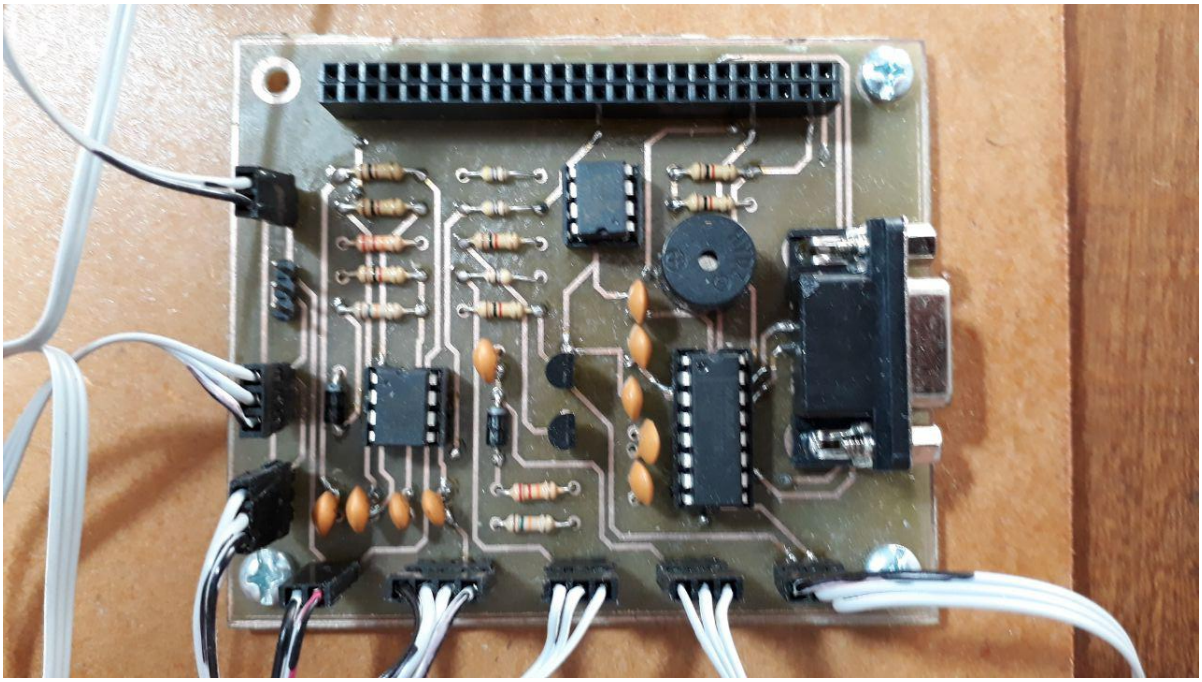


Figura 1.2 Placa 2

1.3 Encendido

Para el encendido solo hay que conectar la fuente de alimentación, si algún sensor no se encuentra el programa continuará.

1.4 Interfaz de usuario

El SAI cuenta con cinco botones físicos, MENU, ARRIBA, ABAJO, BACK y ENTER. Por defecto se muestran carteles con la hora, estado y tipo de cultivo activo. Para poder acceder a información extra se debe acceder al menú. Para ello se debe presionar la tecla MENU y luego recorrer la interfaz con las teclas ARRIBA, ABAJO, BACK y ENTER. Para volver a la pantalla por defecto solo hay que presionar la tecla BACK (repetidas veces) o MENU (una sola vez).

1.5 Tipos de cultivos

Para este prototipo existen dos tipos de seguimiento de cultivos, el tomate y la zanahoria. En cada tipo de cultivo primero se selecciona el día de la plantación y luego se lo controla durante el crecimiento. Los datos para cada etapa están precargados en el sistema.

1.6 Operación

El objetivo del proyecto es controlar la temperatura y humedad de un cultivo. Para ello se debe configurar, en primer lugar, el tipo de cultivo que se quiere seguir. Esto se puede hacer desde la interfaz de usuario con la siguiente combinación de teclas **menu->nuevo seguimiento -> tipo de cultivo**. Una vez seleccionado el tipo de cultivo, el programa calcula los tiempos de cada etapa de acuerdo con los valores precargados. El usuario no tiene que realizar ninguna tarea extra, solo le queda controlar cómo se desarrolla el crecimiento del cultivo. Para esto se generan, en la tarjeta SD, archivos que detallan los valores de los sensores cada un minuto. Cada día se crea un nuevo archivo para lograr un control más ordenado de los datos. Otra manera de visualizar los datos es a través del puerto serie. En este caso se puede monitorear, en tiempo real, los valores de cada

sensor y estado del cultivo. Lo mismo se puede hacer desde menu->variables y la información será reflejada en el display LCD.

En caso de falta de alimentación, el sistema puede comenzar otra vez a funcionar y no perder el seguimiento del cultivo. Esto se debe gracias a que en la tarjeta SD se crea un registro indicando el tipo de cultivo, la etapa actual y las fechas para completar el seguimiento. Además, la placa cuenta con una pila para retener la hora, aunque la fuente de alimentación no esté conectada.

2. *Hardware*

2.1 Diagrama en bloques

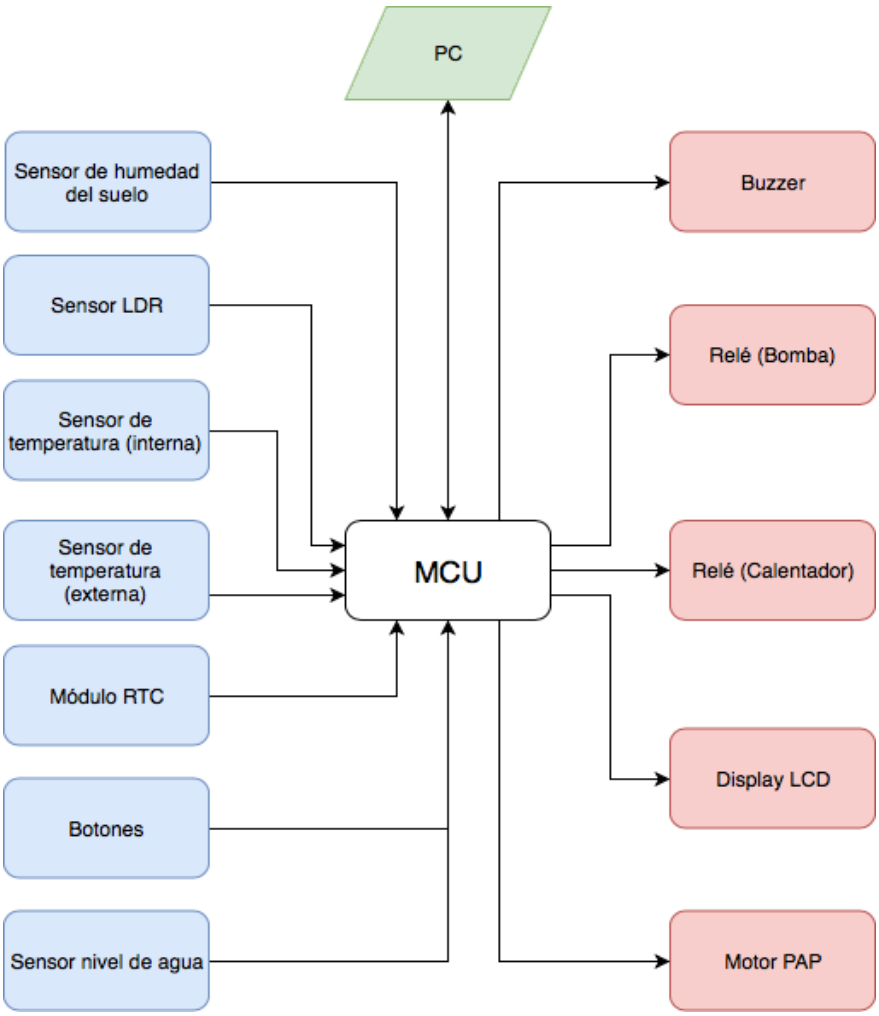


Figura 2.0 Diagrama en bloques

2.2 Circuito esquemático

El sistema cuenta con dos placas para añadir a la placa Discovery STM32F407. La primera placa incluye pines para conectar la pantalla LCD, una pila que funciona como fuente de alimentación para el reloj RTC cuando la placa se queda sin energía y una ranura de la tarjeta SD. Además, contiene 5 botones que permiten al usuario interaccionar con el control del sistema, la

fuente de alimentación y un conector para el motor paso a paso y otro para el sensor de nivel de agua. En la figura 2.1 se muestra el esquemático de esta placa:

La segunda placa incluye conectores para los sensores de temperatura interior y exterior, sensor de luz, sensor de humedad de suelo, relés para accionar la bomba de agua y el calentador. Además, contiene un conector para el driver puente H del ventilador, un Buzzer y un conector RS232 hembra para conexión por puerto serie. La figura 2.2 muestra el circuito esquemático de la segunda placa:

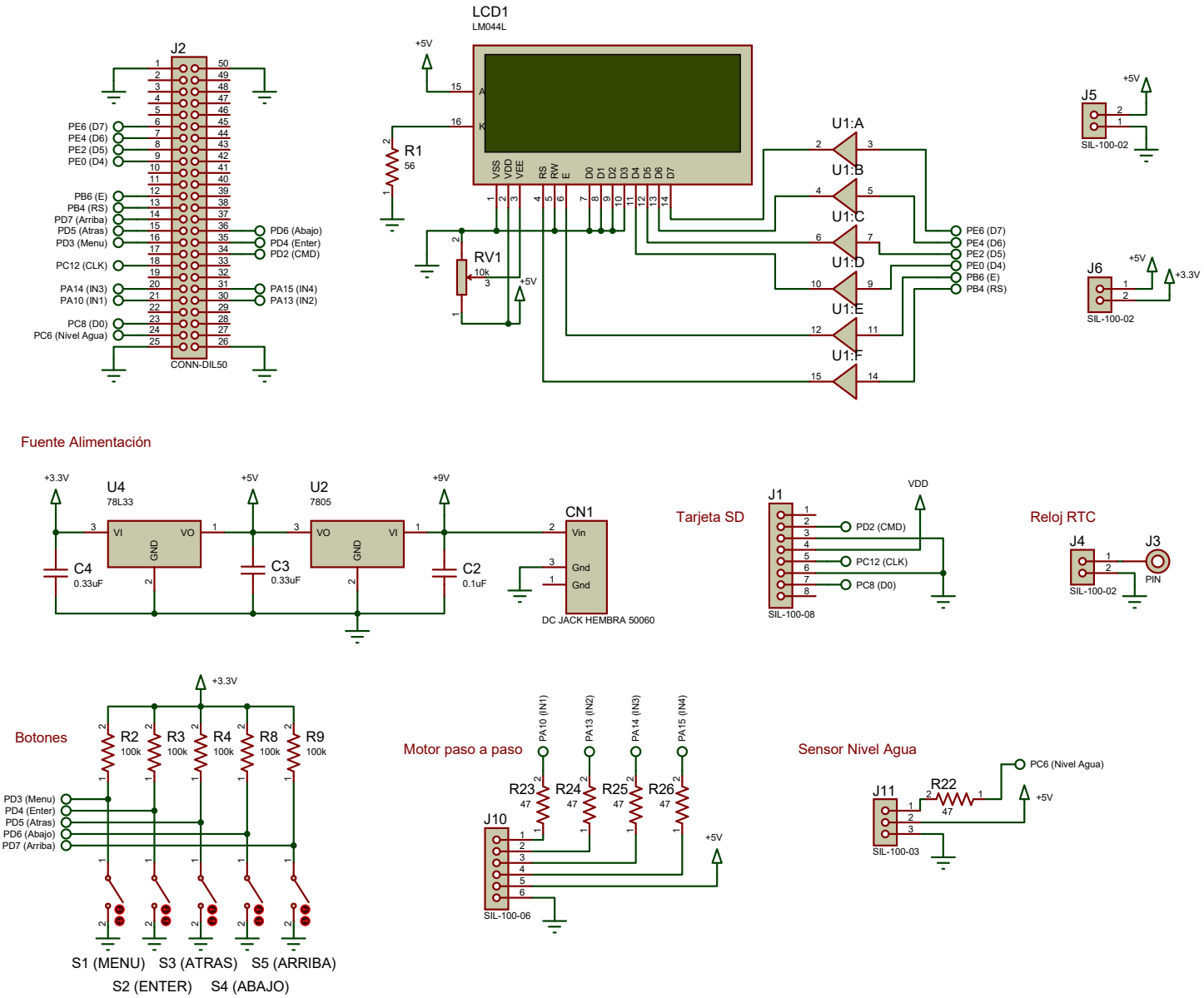


Figura 2.1 Circuito esquemático placa 1

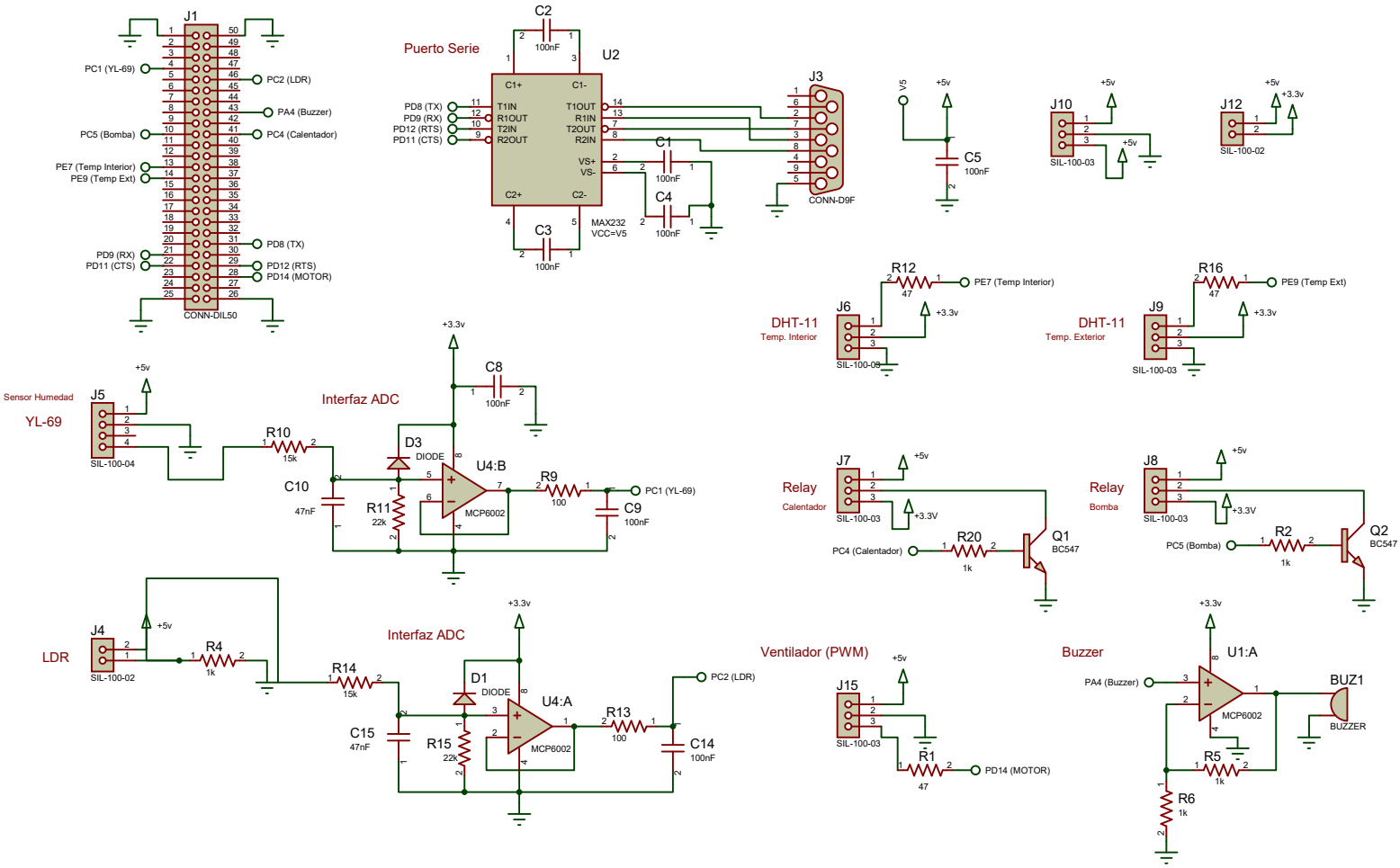


Figura 2.2 Circuito esquemático placa 2

2.3 Descripción del circuito

Los circuitos de las dos placas contienen conectores para la adosar la placa Discovery. En primer lugar, se seleccionaron los pines adecuados para cada periférico. Dependiendo si eran entradas/salidas analógicas o digitales, si necesitaban utilizar un timer, una salida de DAC o un ADC. Para los sensores analógicos (sensor de nivel de luz y de humedad) se utilizó una interfaz para evitar que caigan más de 3V en la entrada de los ADC. Estos circuitos se muestran en la figura 2.2. Se utilizó un amplificador operacional MCP6002 alimentado por 3.3V para cumplir esta tarea. A los sensores digitales se le colocó una resistencia en serie de bajo valor.

La fuente de alimentación regula 9V a 5 y 3.3V. Para ello se colocaron reguladores de tensión 78L05 y 78L33 con sus respectivos capacitores.

Al buzzer se le agregó un amplificador operacional en configuración no inversor para darle una ganancia igual a 2.

Los relés no se accionan con energía que tienen los pines de salida de la Discovery. Por lo tanto, se agregó un transistor en configuración emisor común a cada relé. Funcionan como una "llave a tierra" permitiendo accionarlos con el kit.

Para utilizar el reloj RTC interno fue necesario realizar algunos cambios en la placa del kit. Se agregó un cristal de 32.768 kHz y dos capacitores de 6.8 pF en los pines de C16 y C27. Además, fue necesario sacar la resistencia R26 para poder agregar la batería. Esta se colocó entre el pin VBAT (C28) y tierra.

2.4 Circuito impreso

- Placa 1

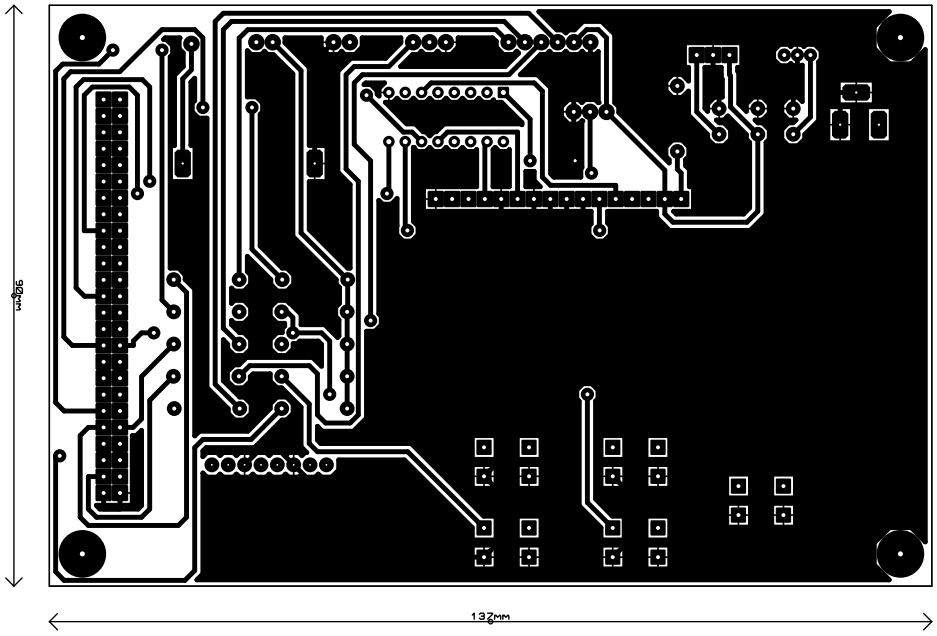


Figura 2.3 Circuito impreso placa 1 (Vista de abajo)

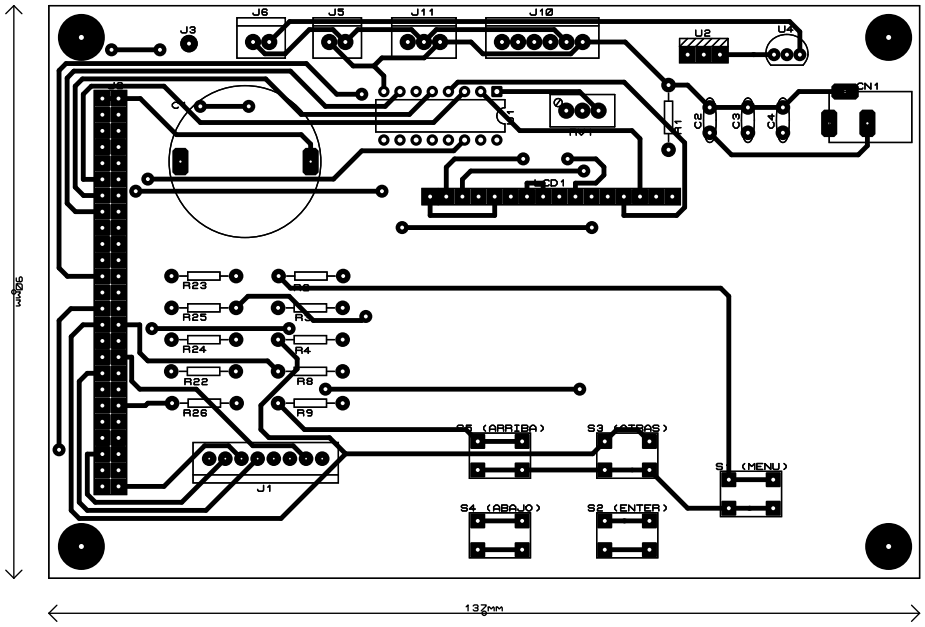


Figura 2.4 Circuito impreso de la placa 1 (Vista desde arriba)

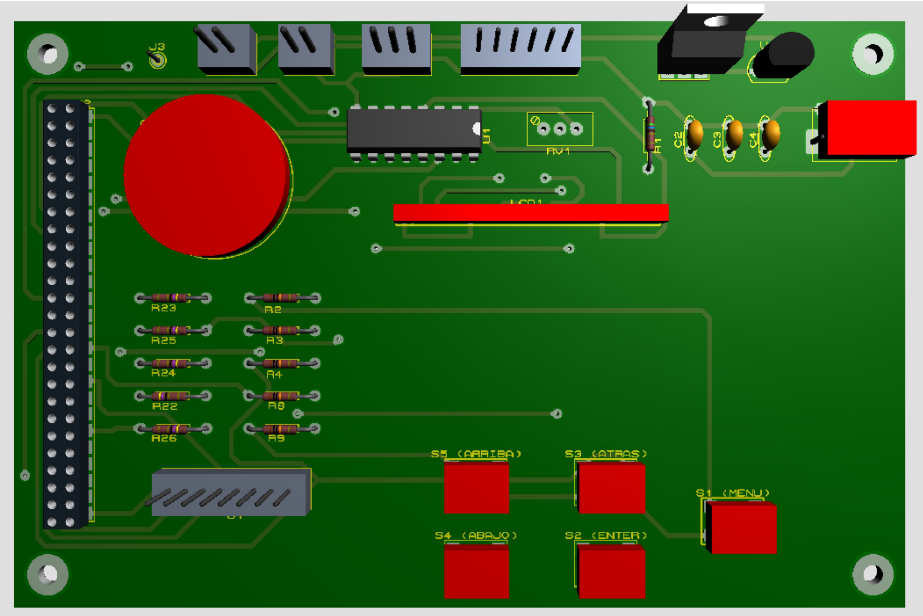


Figura 2.5 Distribución de componentes en formato gráfico (Placa 1)

-Placa 2:

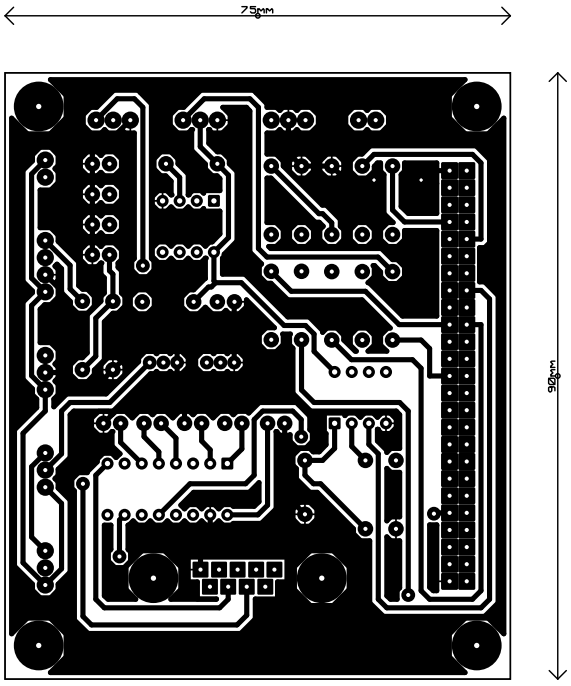


Figura 2.6 Circuito impreso de la placa 2 (vista desde abajo)

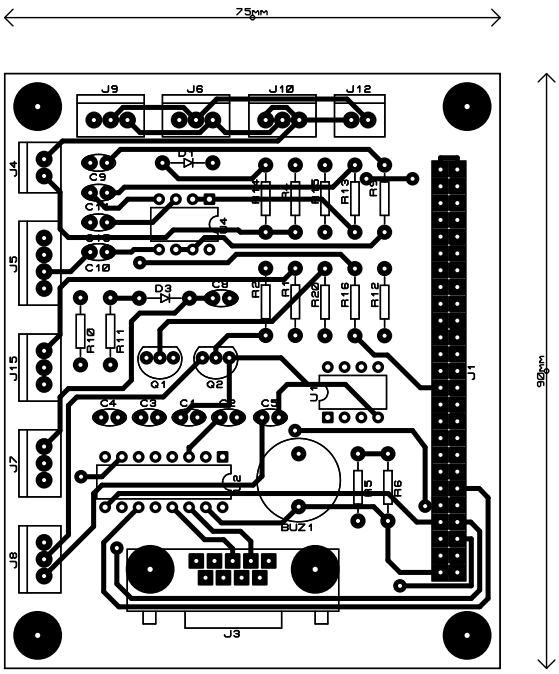


Figura 2.7 Circuito impreso de la placa 2 (Vista desde arriba)

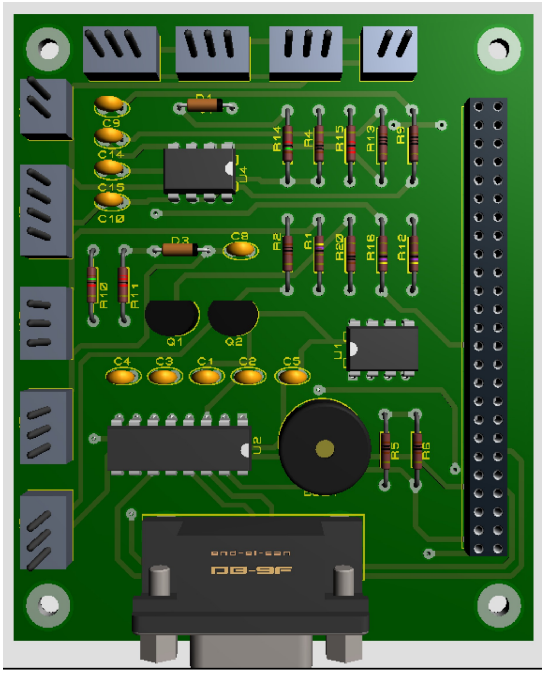


Figura 2.8 Distribución de componentes en formato gráfico (Placa 2)

2.5 Sensores

2.5.1 Sensor de humedad y temperatura DHT11

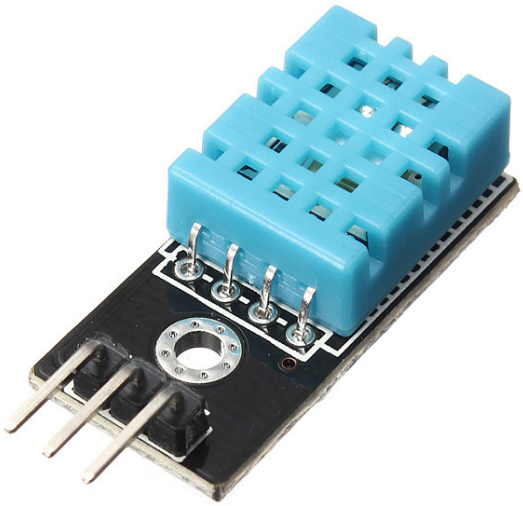


Figura 2.9 Sensor DHT11

Este fue el sensor elegido para obtener la temperatura interna y externa del invernadero. Posee un pin de datos que funciona tanto como entrada y salida. Se comunica en modo dúplex con el MCU. Entrega los datos en binario de la humedad y la temperatura.

2.5.2 Sensor de humedad de suelo YL-69

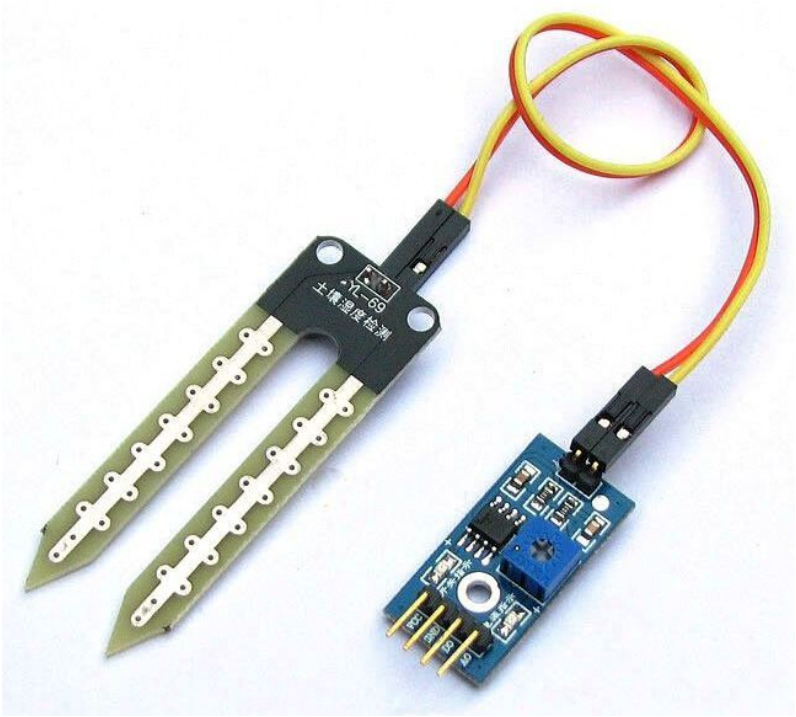


Figura 2.10 Sensor YL-69

El sensor de humedad de suelo YL-69 posee una interfaz la cual entrega un valor de tensión entre VCC y GND. Además, posee una salida digital que se puede regular con un trimpot, para que la salida sea un uno o cero lógicos a partir de cierto nivel de tensión.

2.5.3 Sensor de luminosidad LDR

Figura 2.11 Sensor LDR

LDR es una resistencia sensible a la luz la cual cambia en su valor dependiendo de la cantidad de fotones iluminen los electrodos. Con esta variación de resistencia es posible realizar un circuito que cambie de tensión en función de la luz recibida.

2.5.4 Sensor de agua en el tanque YL-83

Figura 2.12 Sensor YL-83

Este sensor es similar al YL-69 solo que en este caso el nivel analógico no interesa, sino que interesa el nivel digital. La idea es que cuando el agua toque el sensor y las pistas se unan se active el nivel lógico de la interfaz y este valor sea interpretado por el MCU.

3. Software

3.1 Entorno de Desarrollo

3.1.1 Lenguaje de programación

El lenguaje de programación utilizado fue C. Un lenguaje tipado de medio nivel ideal para el desarrollo de sistemas embebidos. Gracias al compilador gcc-arm se utilizaron las variables definidas en C99 uint las cuales son más cómodas a la hora de contar la cantidad de datos cargados en memoria.

3.1.1 IDE para ARM

El IDE utilizado para subir los archivos binarios, debuggear y desarrollar el software para la placa Discovery-STM32F4 fue el Atollic. Este software está basado en el IDE Eclipse y es distribuido gratuitamente por la compañía ST.

3.1.3 Texas instruments FSM generator.

Este generador de máquinas de estado en un archivo .xlsx (hoja de cálculo) el cual se utilizó para generar el menú del sistema. En este archivo se pueden ingresar los eventos y estados deseados de una máquina de estados, crear la tabla de transición de estados y generar dos archivos .h y uno .c donde se encuentra la máquina de estados propiamente dicha. Luego solo basta con añadirla a cualquier proyecto y utilizarla.

3.2 Software microcontrolador

A la hora de estructurar el programa se eligió un sistema con un reloj principal, eventos, tareas, contadores y un despachador de tareas. Así, cada tarea se ejecuta cada cierto tiempo (asignado a cada tarea) y una rutina de interrupción controla los contadores, que, al vencerse, levantan una bandera de evento la cual permite la ejecución de la tarea.

Cada *x* tiempo una interrupción de sistema (Systick) se dispara y llama a una función encargada de controlar los contadores asignados a cada tarea (función task_handler). Esta función es la encargada de levantar banderas de eventos las cuales habilitan la ejecución de una tarea en el plano principal. En la figura 3.1 se muestra un diagrama de bloques con el concepto del software.

En primer plano, se ejecuta un despachador de tareas el cual es interrumpido por el *systick*. Este despachador ejecuta las tareas sólo si el evento fue disparado por el *task_handler()*.

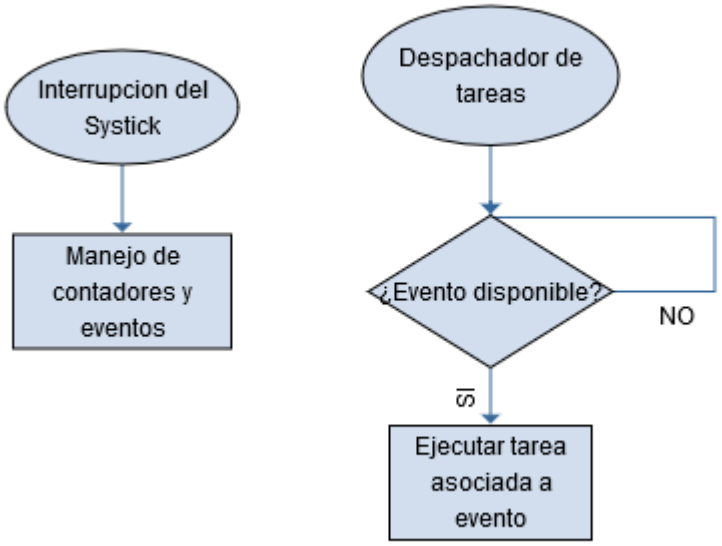


Figura 3.1 Sistema de tareas

Además, se utilizan otros recursos del hardware. En la siguiente tabla se muestran los recursos utilizados para cada sensor o interfaz y si posee o no interrupciones en el sistema.

Sensor/Interfaz	Recurso STM32F4	Interrupciones
RS232	USART	SI
DHT11	GPIO y Timer 5	SI
YL-69	ADC	NO
LDR	ADC	NO
Relés	GPIO	NO
Botones	GPIO	NO
Tarjeta SD	SDIO	NO
Motor paso a paso	GPIO y Timer 3	SI
Ventilador	GPIO y Timer 4 (PWM)	SI
Buzzer	DAC	NO
Sensor de nivel de agua	GPIO	NO

Como se ve, en algunos casos se utilizan interrupciones. En el USART se utiliza para manejar la llegada de datos al sistema y saber si la estructura del paquete recibido es correcta. En el caso del DHT11 solo se utiliza el timer 5 como un pequeño delay del orden de los microsegundos. Para el Motor paso a paso se utiliza el timer 3 para generar un pequeño delay

entre pasos y para el ventilador se utiliza el timer 4 para generar una modulación PWM. Las interrupciones de los timers poseen un orden de prioridad inferior a la del sistema. Gracias a esto no interfieren con el funcionamiento del sistema en general.

3.2.1 Descripción de las funciones principales.

Como se mencionó anteriormente, existen dos funciones madres en esta estructura de software: el *task_scheduler()* y el *task_manager()*.

task_scheduler()

El tasck_schduler es el encargado de manejar los contadores y disparar los eventos que permiten ejecutar una tarea. La estructura principal del task_scheduler y cada rutina posee esencialmente una misma estructura (Figura 3.2):

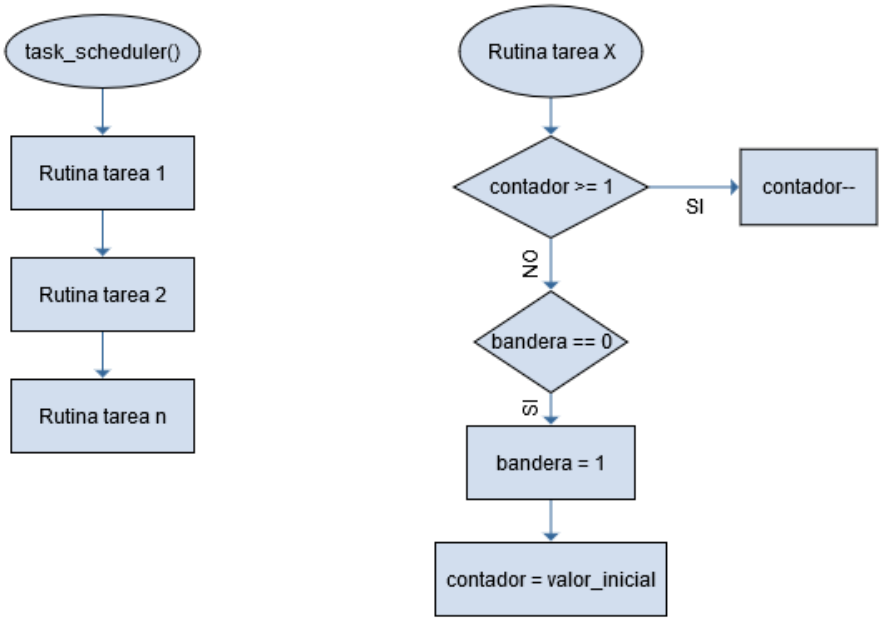


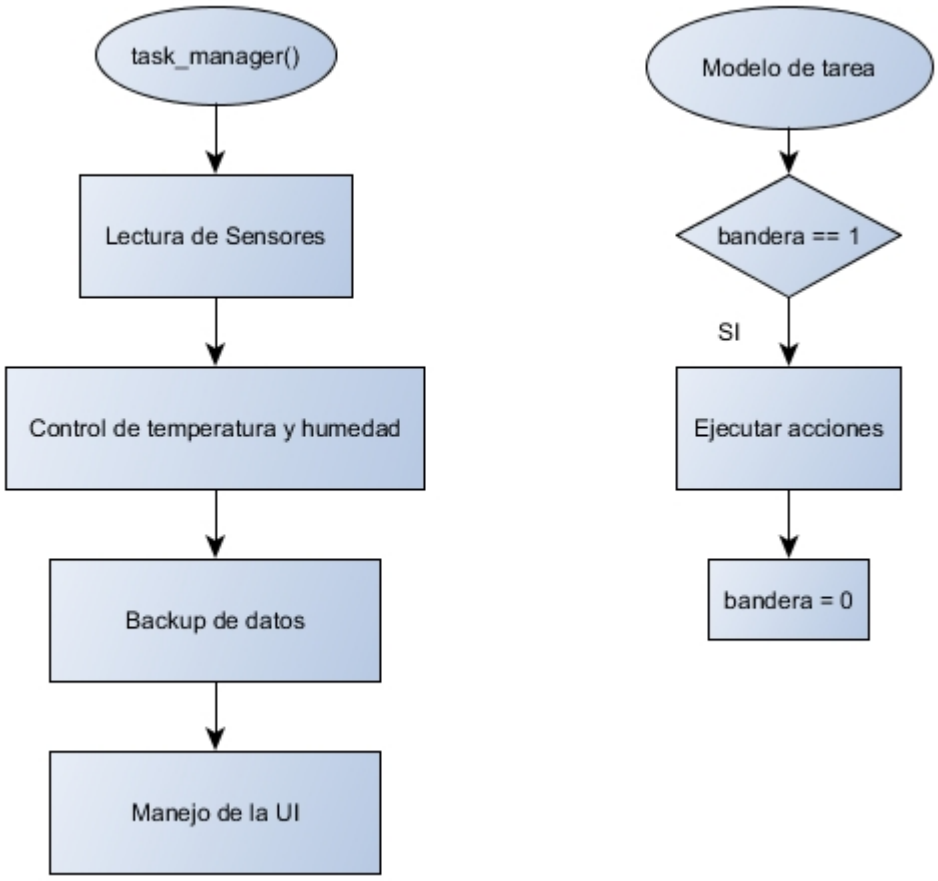
Figura 3.2 Función task_scheduler

Como se puede apreciar, la estructura es bastante simple. Se comienza desde un valor prefijado y se decrementa el contador cada vez que ocurre una interrupción. Si la interrupción se ejecuta cada un milisegundo y el valor inicial es 1001 la tarea se ejecutará cada un segundo. Con la estructura mostrada, en la rutina x se ejecuta la tarea en un bucle infinito. Además de tareas, también existen rutinas para timeouts. En este caso no se debe volver a cargar el valor inicial si el contador se vence.

task_manager()

El *tasck_manager* es la función que se ejecuta continuamente en un bucle infinito. Aquí se encuentran todas las tareas del sistema las cuales van ejecutando a medida que la función *task_scheduler* las designa. La idea fue que cada tarea controle un periférico a la vez (a excepción de las tareas de interfaz de usuario se utiliza para enviar mensajes) y que se comunique con las otras tareas a través de variables globales. A su vez, estas tareas se dividen como lectura de sensores, tareas de control (control de humedad y temperatura), tareas de

backup (escritura en la SD) y tareas de interfaz de usuario (LCD y USART). En la figura 3.3 se muestra un diagrama general de la función y de una tarea genérica.



3.2.2 Estructura y algoritmos del protocolo de comunicación USART.

La estructura para la recepción y emisión de los paquetes se diagramó de la siguiente forma:

Encabezado	Comando	Cant_bytes_datos	Datos	Finalizador

El formato de paquete quedó de la siguiente forma

:CMD,N_BYTES,DATA!

En esta versión del software no hay checksum y tampoco se comprueba la cantidad de datos del mensaje DATA. Sin embargo, el software encargado de separar el paquete que ingresa si separa los distintos campos. Por lo tanto, en un futuro update del sistema será posible agregar la comprobación de la cantidad de datos y la cantidad de bytes que se especificaron en la comunicación sin modificar el algoritmo de recepción del paquete.

En la siguiente tabla se muestran los comandos aceptados por el sistema:

Comando	Paquete	Función
Iniciar	:INI,-,-!	Inicia la conversación.
Stop	:STP,-,-	Termina la conversación.
Monitor	:MON,-,ON	Activa el modo monitor.
Monitor	:MON,-,OFF	Desactiva el modo monitor.
Tarjeta presente	:SDC,-,-	Verifica si la tarjeta SD está o no puesta.
Activar cultivo	:TOM,-,-	Comienza el seguimiento de tomate
Activar cultivo	:ZAN,-,-	Comienza el seguimiento de la zanahoria
Modificar temperaturas	:MOD_TEMP,-,ETAPA XX TIPO XX MAX_FAN XX MIN_FAN XX MIN_CAL XX MAX_CAL XX!	Modifica las temperaturas objetivo de una etapa.
Check temperaturas	:CK_TEMP,-,TOM!	Muestra las temperaturas objetivo de las etapas.
Parar seguimiento	:PAUSE,-,-!	Para el control de temperatura y humedad del seguimiento
Continuar seguimiento	:CONTINUE,-,-!	Habilita el control de temperatura y humedad del seguimiento.
Fecha	:DAY,2,YEAR xx MON xx DATE xx DAY xx HOUR xx MIN xx SEC xx!	Cambia la hora del sistema.
Ventana	:VENT0,-,-	Abrir ventana
Ventana	:VENTC,-,-	Cerrar ventana

Ventana	:STOPV,-,-	Parar ventana
Ventilador	:FAN,-,ON	Prender ventilador
Ventilador	:FAN,-,OFF	Apagar ventilador
Calentador	:CAL,-,ON	Prender calentador
Calentador	:CAL,-,OFF	Apagar calentador
Bomba	:BOMBA,-,ON	Prender Bomba
Bomba	:BOMBA,-,OFF	Apagar Bomba

En el siguiente diagrama de flujo (Figura 3.4) se muestra el algoritmo para separar los bytes que ingresan al sistema en las distintas partes del paquete.

Al ingresar un nuevo byte se comprueba si es un inicializador. Si lo es y antes no había ingresado ningún paquete nuevo se habilita la bandera de comienzo de paquete, en caso contrario se cancela la recepción del paquete y, para comenzar de nuevo la recepción, se tendría que enviar el paquete de nuevo. Si ocurriese esto último un mensaje de error es enviado a través del USART.

Si no es un inicializador y es un finalizador, se comprueba si el mensaje estaba siendo recibido (chequeando la bandera de comienzo de paquete). De ser así y todos los campos fueron recibidos se finaliza el paquete y se habilita una bandera para que la rutina del programa principal pueda procesar el paquete. En caso contrario se muestra un mensaje de error a través del USART y se reinician las banderas de recepción de paquete.

Si el byte ingresado es un separador de campo y el mensaje está siendo recibido (bandera de comienzo de paquete en alto) se cambia el campo de recepción. En caso de que se supere la cantidad de campos (por muchos mensajes de separador de campo) se envía un mensaje de error y se cancela la recepción del paquete (reiniciando las banderas correspondientes).

Si el byte ingresado no es un finalizador, separador de campo o inicializador se lo agrega al campo actual, siempre y cuando la recepción del paquete esté habilitada (bandera de comienzo de paquete en alto).

Además, en la llegada de cada byte se reinicia un contador de timeout. Si el nuevo byte ingresa despues de la finalizacion de este contador se descarta y se comienza de nuevo la recepcion del paquete (un mensaje de error es mostrado).

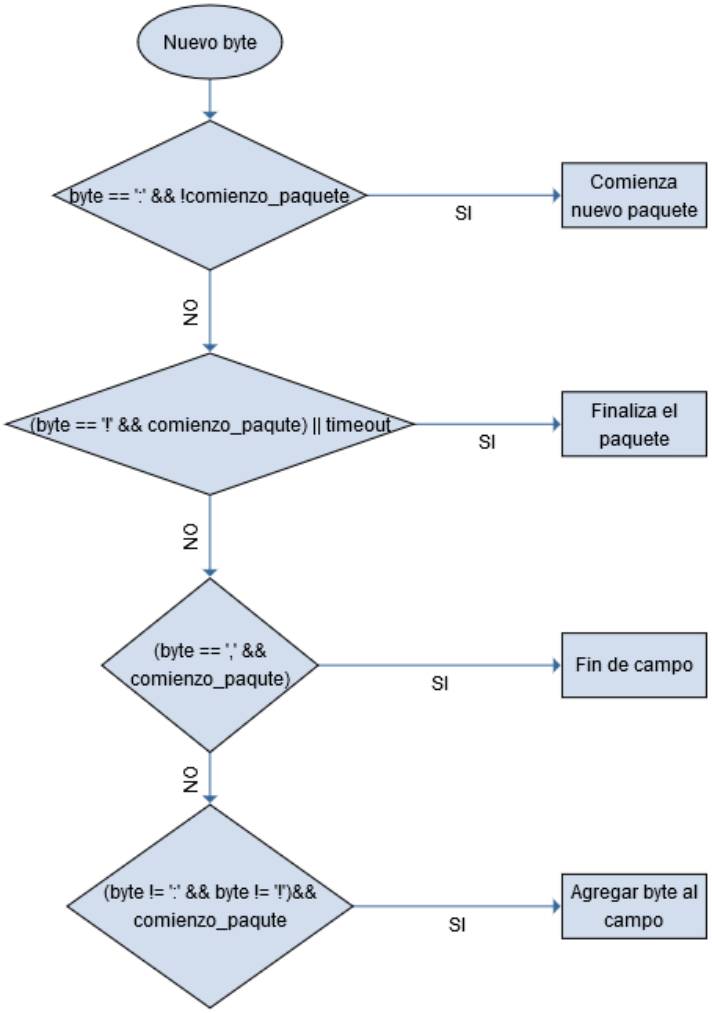


Figura 3.4 Rutina de recepción de paquete

3.2.3 Estructura y algoritmos del protocolo de la tarjeta SD.

En la tarjeta SD se crean dos archivos de backup *ETAPAS.TXT* y *LOG.TXT*, y cada una hora un nuevo archivo datalogger *TI-xxxx.csv*. Estos archivos tienen sus campos separados por los caracteres ‘ ’, ‘-’, ‘.’ y ‘,’

‘ ‘	‘-’	‘.’	-,-

De esta manera y con ayuda de la librería *string.h* se facilita la tarea de tomar y separar los datos.

El archivo *ETAPAS.TXT* posee las distintas fechas para las alarmas de la etapa configurada. En este caso se utilizaron 3 etapas por lo que cada archivo tiene tres veces la siguiente cadena:

,ETAPA xx, HOURS xx,

El archivo *LOG.TXT* posee la información del tipo de cultivo, si este está activo y la etapa en la que se encuentra:

,TIPO_CULTIVO XX, ETAPA_ACTUAL XX, ESTADO XX,

Los posibles valores de estos campos son los siguientes:

Campo	Valores posibles
ETAPA	0,1,2
TIPO_CULTIVO	0 (tomate),1(zanahoria)
ETAPA_ACTUAL	0,1,2
ESTADO	0(seguimiento activo),1(seguimiento desactivado)

Al comienzo del programa se leen los archivos LOG.TXT y *ETAPAS.TXT,* y dependiendo de su contenido se configura el seguimiento anterior o se deja el programa sin seguimiento.

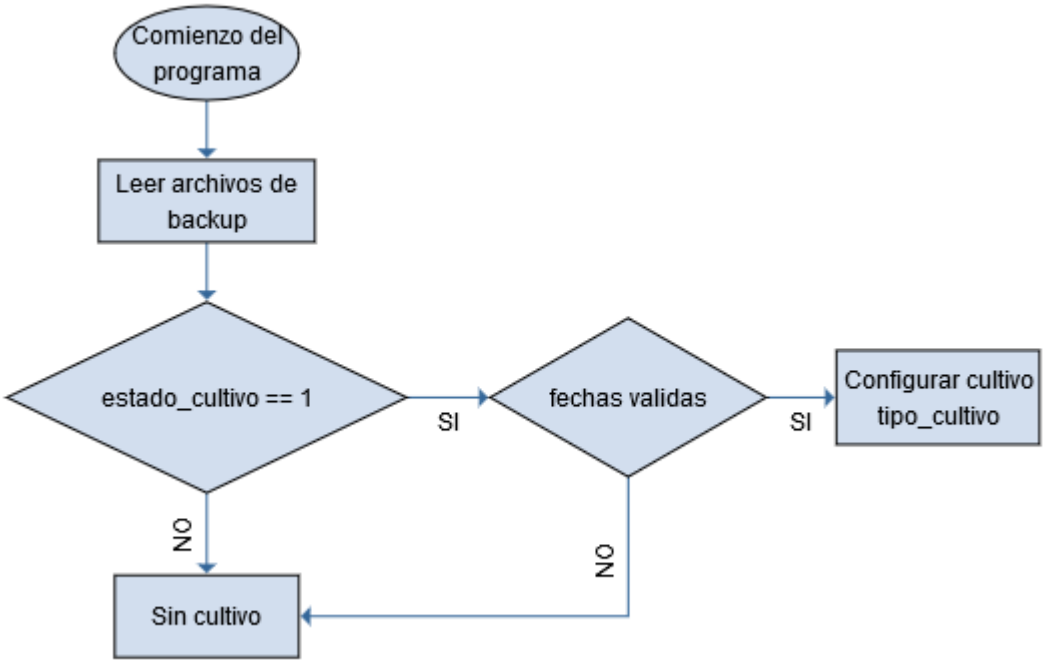
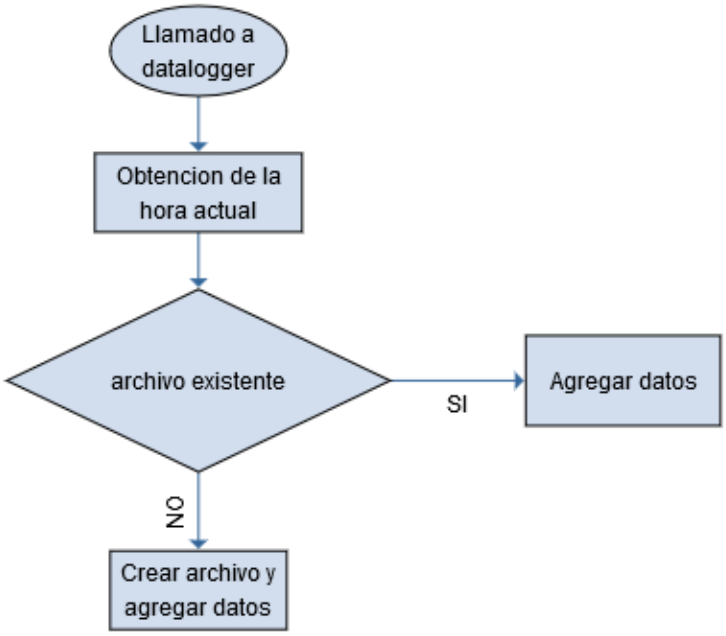


Figura 3.5 Lectura de los archivos de bakup

El archivo TI-DH.csv tiene en su nombre el día actual y la hora. Cuando un cultivo está activo se cargan los datos actuales de temperatura, humedad y luminosidad cada 10 segundos. Si el archivo no existe con la fecha y hora actual se lo crea, de lo contrario se añaden los datos al final del archivo.



3.2.4 Algoritmos de control de temperatura y humedad

Los controles de temperatura y humedad son desarrollados con un sistema de control por histéresis. Se crean límites superiores e inferiores de temperatura o humedad y se mantiene la misma entre estos valores. En la figura 3.6 se muestra un diagrama de este sistema en el caso de temperatura/control y en la figura 3.7 el mismo pero invertido.

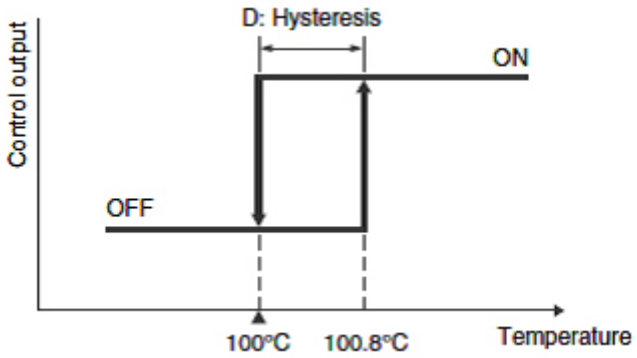


Figura 3.6 Control con histéresis

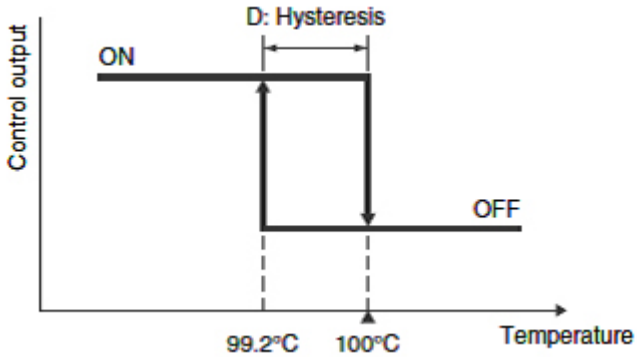


Figura 3.7 Control con histéresis

En este proyecto se utilizaron ambos esquemas. Para el control de temperatura se utilizaron dos controles de histéresis: uno para el calentador y otro para el ventilador. En este caso el diagrama de la figura 3.6 corresponde al control del relé del calentador, y el de la figura 3.7 al relé del ventilador.

Para el control de humedad se utiliza el diagrama de la figura 3.6 que controla el rele de la bomba de agua.

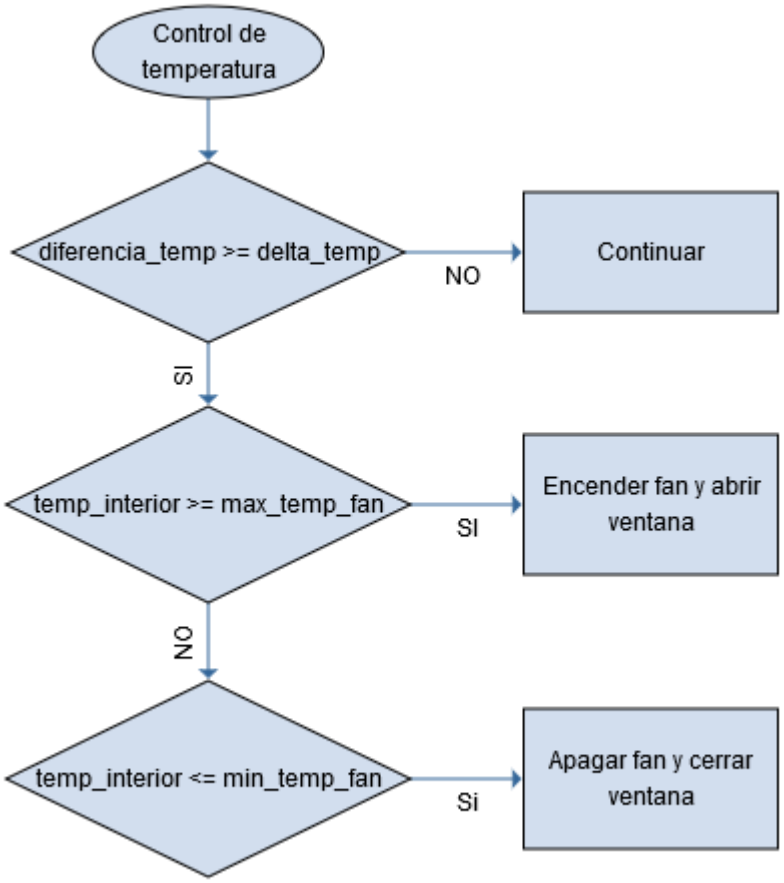


Figura 3.8 Control de temperatura (ventilador)

En la figura 3.8 se muestra el diagrama de flujo del algoritmo de control de temperatura asociado al ventilador.

Al principio se comprueba que la diferencia de temperatura entre el interior y el exterior no sea muy grande, esto se realiza para evitar ingresar aire muy caliente o aire muy frío. Luego si la temperatura en el interior es mayor a la máxima permitida en la etapa del cultivo se enciende el fan y se abre la ventana para enfriar el interior. Por otro lado, si la temperatura es inferior o igual a la mínima establecida en la etapa se apaga el ventilador y se cierra la ventana.

En la figura 3.9 se encuentra el diagrama del control de temperatura asociado al calentador. En este caso cuando la temperatura del interior es menor que la mínima establecida se enciende el calentador hasta que la temperatura sea mayor que la máxima de la etapa.

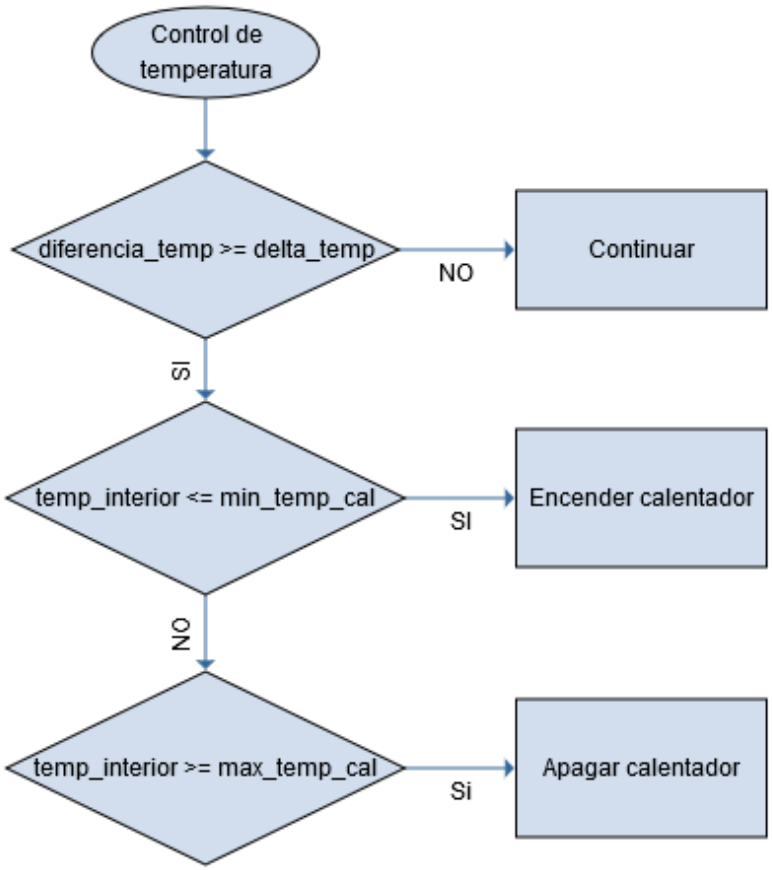


Figura 3.9 Control de temperatura (calentador)

La figura 3.10 detalla el control de la bomba, encargada de modificar la humedad del suelo. En primer lugar, se chequea si la humedad es inferior a la mínima establecida. De ser así y si hay agua en el tanque, se prende la bomba. Si la humedad es superior a la máxima, se detiene la bomba

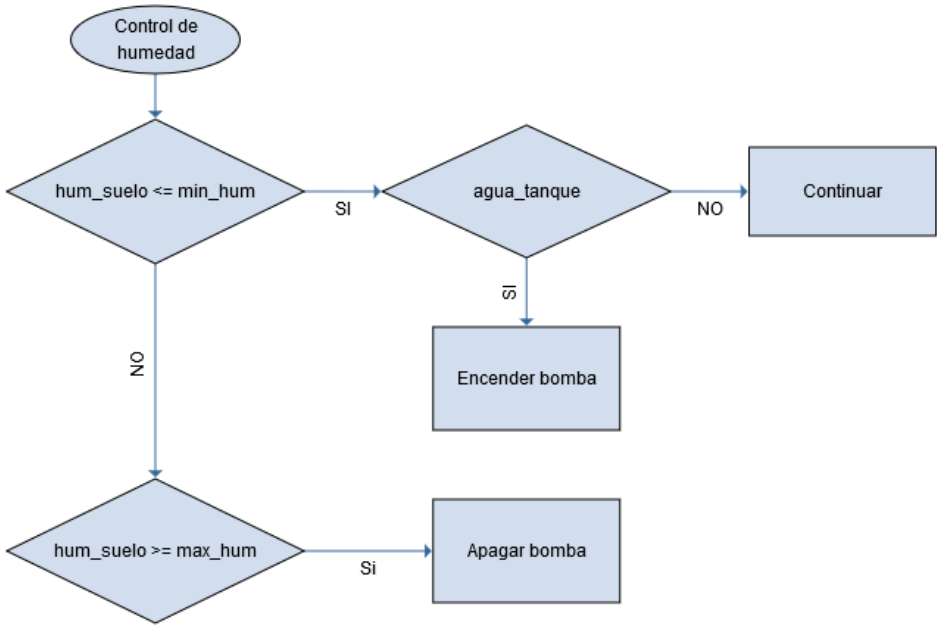


Figura 3.10 Control de humedad (bomba)

3.2.5 Algoritmos de seguimiento del cultivo.

Como se mencionó anteriormente, cada cultivo posee distintas etapas. Estas poseen, a su vez, distintos parámetros los cuales deben ser cargados en las variables que son utilizadas para el control. Esto se realizó con una alarma del reloj RTC interno. Para ello se configura un arreglo con las distintas fechas en donde comienza una nueva etapa. Este arreglo es creado cuando el seguimiento del cultivo es configurado, de acuerdo al tipo de cultivo. Además, se crea un archivo de backup en la tarjeta SD para recuperar las fechas en caso de un reinicio del sistema o corte del suministro energético.

La figura 3.11 muestra el diagrama de flujo de la configuración de un cultivo.

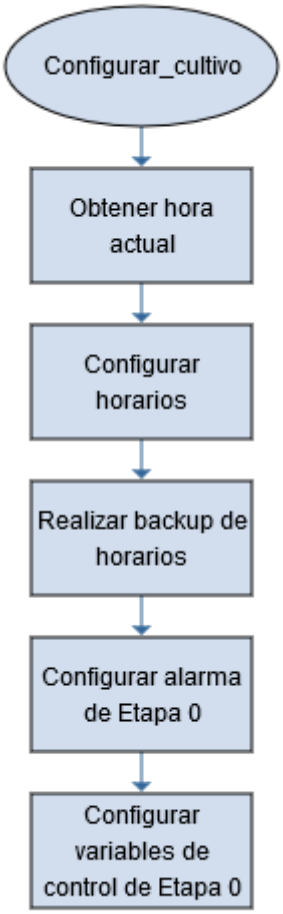


Figura 3.11 Función configurar cultivo

Cuando la alarma es disparada se llama a un *handler* de la misma. En este *handler* se realiza el cambio de etapa y se configura la alarma para la próxima etapa. La figura 3.12 muestra el diagrama de flujo de este proceso.

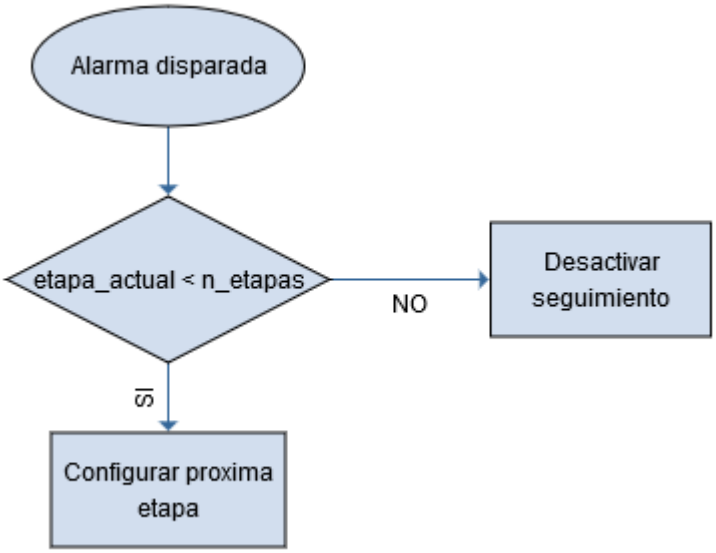


Figura 3.12 Handler de la alarma RTC

Cuando el seguimiento del cultivo está activo el control de temperatura y humedad se encuentra activo, salvo que el usuario lo pause manualmente. De esta manera el sistema automáticamente se va configurando a medida que transcurre el crecimiento del cultivo.

3.2.6 Lectura de periféricos

Los distintos periféricos que suministran los datos para el control son:

- 2 sensores de humedad y temperatura (DHT11)
- 1 sensor de luz (LDR)
- 1 sensor de humedad del suelo (YL-69)
- 1 sensor de nivel de agua ().

3.2.6.1 Lectura de temperatura ambiente

Esta tarea se efectúa cada vez que la función *task_scheduler* lo dispone. Cada vez que se vence el contador se habilita le lectura de uno de los sensores. Cabe recordar que hay dos sensores porque uno se utiliza para la temperatura interior y otro para la temperatura exterior. El siguiente diagrama de flujo muestra como es el funcionamiento de la lectura de un sensor:

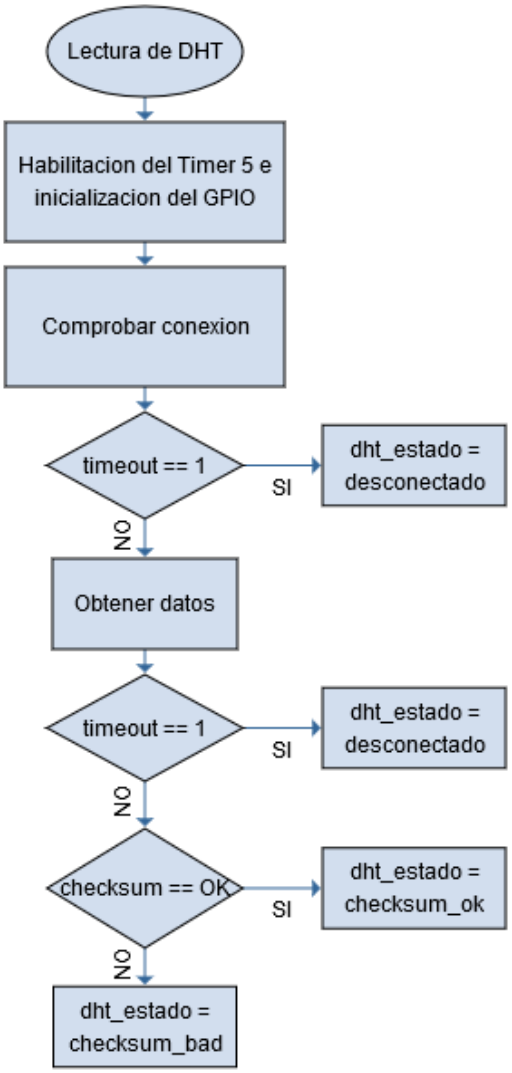


Figura 3.13 Rutina de lectura de un sensor DHT11

3.2.6.2 Lectura de humedad del suelo

En el caso de la lectura de la humedad del suelo el proceso es más simple que en el sensor DHT. Esto se debe a que el sensor posee una salida analógica la cual es capturada por el ADC. La figura 3.14 muestra el diagrama de flujo de la tarea de lectura de la humedad.

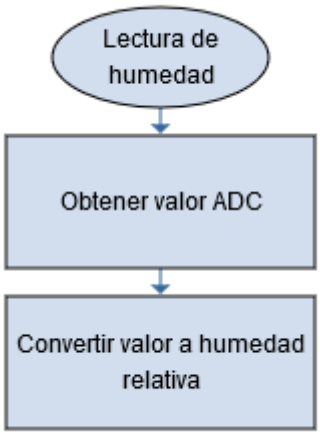


Figura 3.14 Rutina de lectura del sensor de humedad

Luego de obtener el valor en bruto del ADC se lo transforma a valores de humedad relativo (en porcentaje).

3.2.6.3 Lectura de luminosidad

Para la lectura de la luminosidad se utilizó un sensor LDR el cual a través de un circuito de adaptación se lo conecta a uno de los ADC del MCU. En este caso, para mayor estabilidad de los valores leídos, se agregó un promedio a la lectura. Debido a esto, se agregó un contador extra que lleve el número de cuentas medidas y cuando llegue al tiempo de lectura definido se computa el promedio. En la figura 3.15 se puede ver como es este proceso.

Por el momento no se realizó la conversión de los datos en bruto a un valor relativo o a una clasificación, pero esto puede ser incluido en una actualización de software.

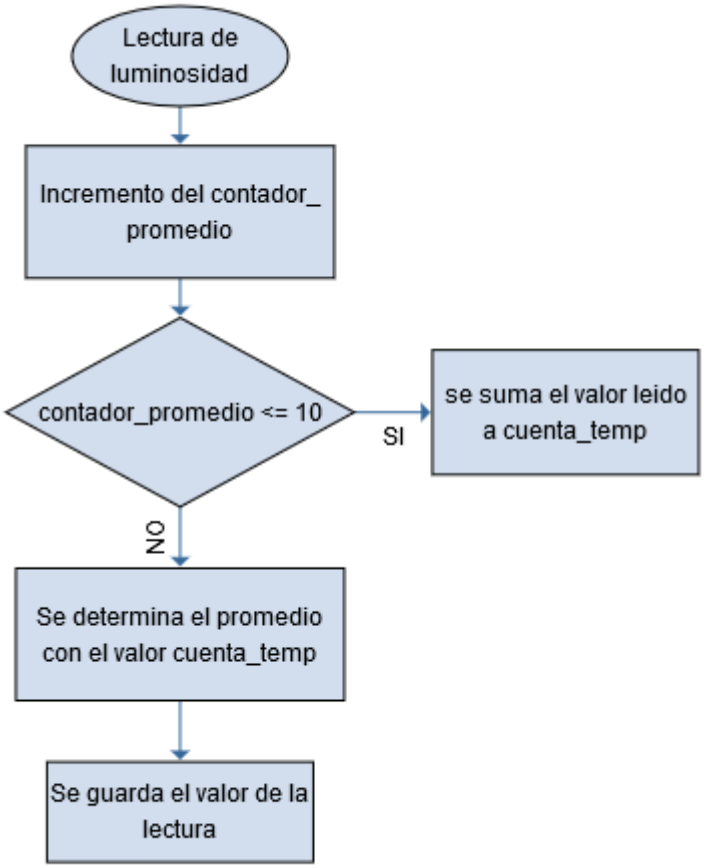


Figura 3.15 Rutina de lectura del sensor de luminosidad

3.2.6.4 Lectura de nivel de agua

El sensor de nivel de agua está configurado para que entregue un valor digital si el agua toca o no al sensor. La lectura de este sensor es muy sencilla, solo se debe obtener el valor digital en la entrada del pin al que está conectado. El diagrama se muestra en la figura 3.16.

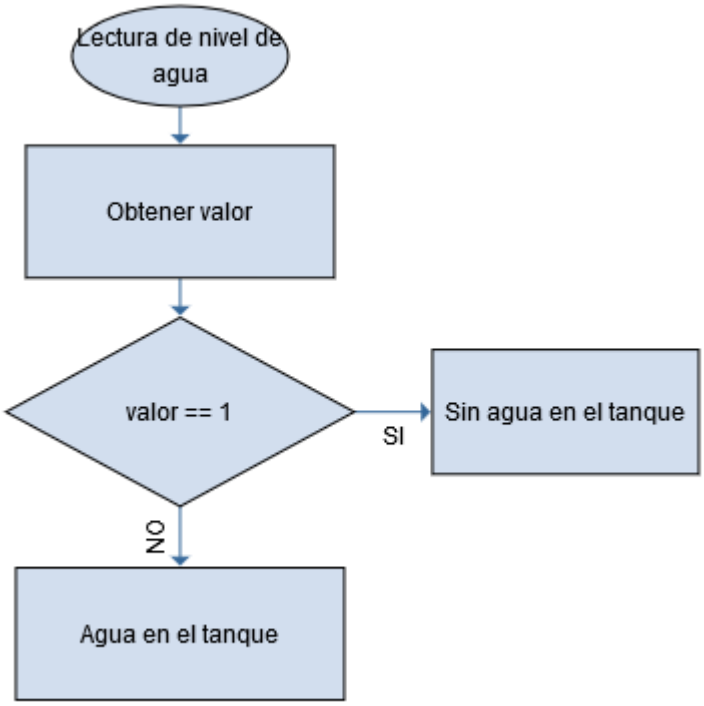


Figura 3.16 Rutina de lectura del sensor de nivel de agua

3.2.7 Interfaz de usuario

Como se mencionó anteriormente, el menú fue realizado con el software FSMGenerator. La tabla de transición quedo de la siguiente manera

	menu		enter		arriba		abajo		back	
	Function	Next State	Function	Next State	Function	Next State	Function	Next State	Function	Next State
menu_desactivado		cultivo		menu_desactivado		menu_desactivado		menu_desactivado		menu_desactivado
cultivo		menu_desactivado		nuevo_cultivo		cultivo		variables		menu_desactivado
variables		menu_desactivado		humedad		cultivo		control		menu_desactivado
control		menu_desactivado		bomba		variables		exportar		menu_desactivado
exportar		menu_desactivado		log		control		exportar		menu_desactivado
nuevo_cultivo		menu_desactivado		tomate		nuevo_cultivo		pausar		cultivo
pausar		menu_desactivado	f_pausar	pausar		nuevo_cultivo		cancelar		cultivo
cancelar		menu_desactivado	f_cancelar	cancelar		pausar		cancelar		cultivo
tomate		menu_desactivado	f_tomate	tomate		tomate		zanahoria		nuevo_cultivo
zanahoria		menu_desactivado	f_zanahoria	zanahoria		tomate		zanahoria		nuevo_cultivo
humedad		menu_desactivado	f_humedad	humedad		humedad		temp_ext		variables
temp_ext		menu_desactivado	f_temp_ext	temp_ext		humedad		temp_int		variables
temp_int		menu_desactivado	f_temp_int	temp_int		temp_ext		temp_int		variables
bomba		menu_desactivado	f_bomba	bomba		bomba		calentador		control
calentador		menu_desactivado	f_calentador	calentador		bomba		ventana		control
ventana		menu_desactivado	f_ventana	ventana		calentador		ventilador		control
ventilador		menu_desactivado	f_ventilador	ventilador		ventana		ventilador		control
log		menu_desactivado	f_log	log		log		etapas		exportar
etapas		menu_desactivado	f_etapas	etapas		log		etapas		exportar

Luego se agregaron los archivos al proyecto y se realizaron las adaptaciones necesarias para utilizar la máquina de estados.

Para mostrar la información en la pantalla existen dos modos principales, uno con el menú desactivado y otro con el menú activado. Cuando el menú esta desactivado se refresca la pantalla cada dos segundos cambiando el cartel cíclicamente.

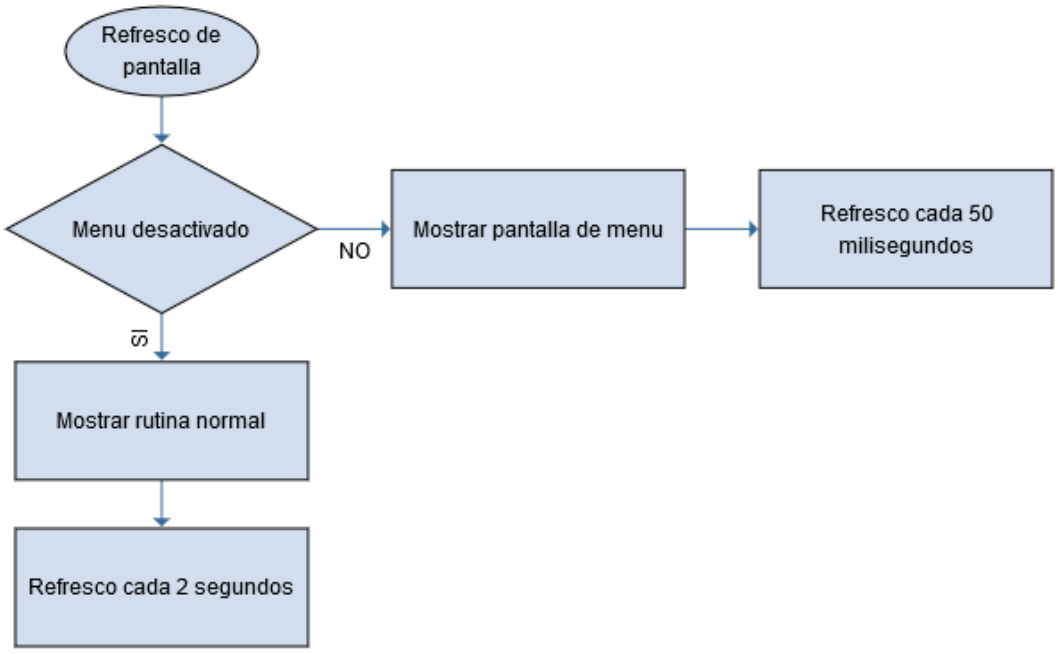


Figura 3.17 Rutina del display

Si se encuentra el menú activado, se muestra el cartel correspondiente al estado actual del menú. En este caso la pantalla se refresca cada cincuenta (50) milisegundos para obtener una mejor respuesta de los botones ante el cambio de posición del cursor. Esta rutina se muestra en la figura 3.17.

La máquina de estados generada por el software genera funciones para activar cada evento. Estos eventos son las entradas para la máquina de estados, por lo que para "mover" la máquina de estados debe llamarse a la función que corresponde a cada evento cuando un botón es presionado. La rutina se muestra en la figura 3.18.

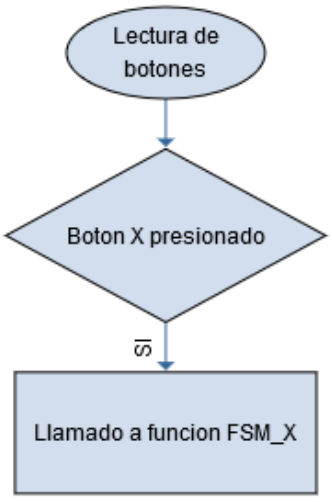


Figura 3.18 Rutina de botones

3.3 Librerías

Para el desarrollo del software del proyecto fueron utilizadas varias librerías de terceros. Estas librerías facilitaron la creación del código ya que se encargan de controlar distintos periféricos a bajo nivel y brindan funciones a más alto nivel que son fáciles de utilizar. Las librerías utilizadas se muestran en la siguiente tabla:

Nombre de la librería	Periférico que controla	Propiedad intelectual
adc	ADC1	Propia
dht	Sensor DHT11	Propia
ventana	Motor paso a paso	Propia
stm32f4xx	Todos de la STM32F4	ST
stdio	Ninguno	C
stdlib	Ninguno	C
string	Ninguno	C
FATFS_SDIO	Tarjeta SD	Uwe Becker
LCD_2x16	Display LCD	Uwe Becker
TM-STM32F4 RTC	Reloj RTC STM32F4	Tilen Majerle

A su vez estas librerías poseen ciertas dependencias de otras librerías. Para ver la documentación o contenido de las mismas, en la sección *Referencias* se encuentran los links para cada librería.

4. Futuras mejoras

A futuro se pensaron distintas modificaciones de software para mejorar la experiencia del usuario final. De momento, los tiempos que dura cada etapa están predeterminados en el sistema, y si bien se pueden cambiar, habría que agregar una función que determine sin errores las fechas para cada etapa con solo especificar la cantidad de días que dura la etapa. Esta función tendría que tener en cuenta la cantidad de días en el mes, si es año bisiesto, etc.

Otro agregado sería el de un cultivo *custom*, el cual a través de un archivo en la SD se puedan leer los distintos parámetros para cada etapa y seleccionar este tipo de cultivo desde el menu.

Además, una aplicación de PC que controle al dispositivo a través del puerto serie y con interfaz gráfica de usuario. Que cuente con la posibilidad de subir los datos recolectados por los sensores a una plataforma online (como Google Docs) para obtener información en tiempo real desde internet.

5. Referencias

[1] Repositorio Github del proyecto.

[2] Pagina web de la placa STM32F4 Discovery.

[3] Pagina web del IDE Atollic.