Centro de Educación Técnica N.º 2 "Jorge Newbery"

Proyecto "El Invernadero Inteligente" | Resumen del código

Cursos: 3º 2º C.S. y 3º 4º C.S.

Funcionamiento de la versión 2.1.0 de software para la maqueta del invernadero inteligente

Quintana Santiago Esteban

Autor

Martínez Ricardo

Coordinador

S.C. de Bariloche. 21 de Noviembre, 2022

Lugar y fecha

Índice de contenidos

Palabras clave

- Función, subrutina (programación).
- EEPROM (Electrically Erasable Programmable Read-Only Memory).
- IDE (Integrated Development Environment).
- Bot (Telegram).
- LED (Light Emitting Diode).

1. Resumen

La programación del controlador del invernadero inteligente se llevó a cabo entre los días 25 de Septiembre y 19 de Noviembre del 2022. Para ella se utilizó el lenguaje de programación C++, en la IDE “Visual Studio Code” con la extensión “PlatformIO”; utilizando funciones de Arduino.

El código se divide en ocho archivos de origen C Header (.h) y un archivo de origen C++ (.cpp). En este último se encuentra la función setup() que es ejecutada al principio del código, y la función loop(), que se ejecuta en bucle durante el resto del tiempo de encendido del controlador.

Es importante destacar que, si bien la estructura del software es definitiva, el código contiene detalles que lo hacen apto solamente para la maqueta para la que se diseñó. Sin embargo, la estructura principal está creada, y su división en archivos le confiere un fácil mantenimiento y modificación.

Todo el código y algunas de sus versiones anteriores se encuentran públicos en el sitio web GitHub:

[https://github.com/Quintana-S-E/Invernadero-Inteligente-C.E.T.-N.-2-maqueta]

2. Versiones

El progreso de todo el código desde sus primeras 200 líneas hasta sus actuales 1305 fue guardado en 26 versiones representadas por tres números (x.x.x), siguiendo un criterio para cada uno. El último número (x.x.**y**) cambió con pequeños arreglos o implementaciones menores. El número del medio (x.**y**.x) cambió con implementaciones importantes o pruebas exitosas en el controlador de características anteriores; y finalmente el primer número (**y**.x.x) cambió con la finalización de la programación relacionada al comportamiento del invernadero, y la implementación de una funcionalidad clave del controlador (modificaciones trascendentales).

3. Librerías

El código se vale de 12 librerías públicas para el manejo de los gráficos en la nube, la EEPROM, Telegram, sensores, el servomotor y el display. Estas son:

- Arduino (Arduino)
- ArduinoJson (Arduino)
- EEPROM (Arduino)
- WiFi (Arduino)
- Wire (Arduino)
- Adafruit GFX (Adafruit)
- Adafruit Unified Sensor Driver (Adafruit)
- Adafruit SSD1306 (Adafruit)
- DHT (Adafruit)
- ESP32Servo (Kevin Harrington)
- ThingSpeak (MathWorks)
- CTBot (Stefano Ledda)

4. Diagrama de flujo

En la siguiente página se encuentra el diagrama de flujo, que describe en un alto nivel el funcionamiento del software del invernadero inteligente.

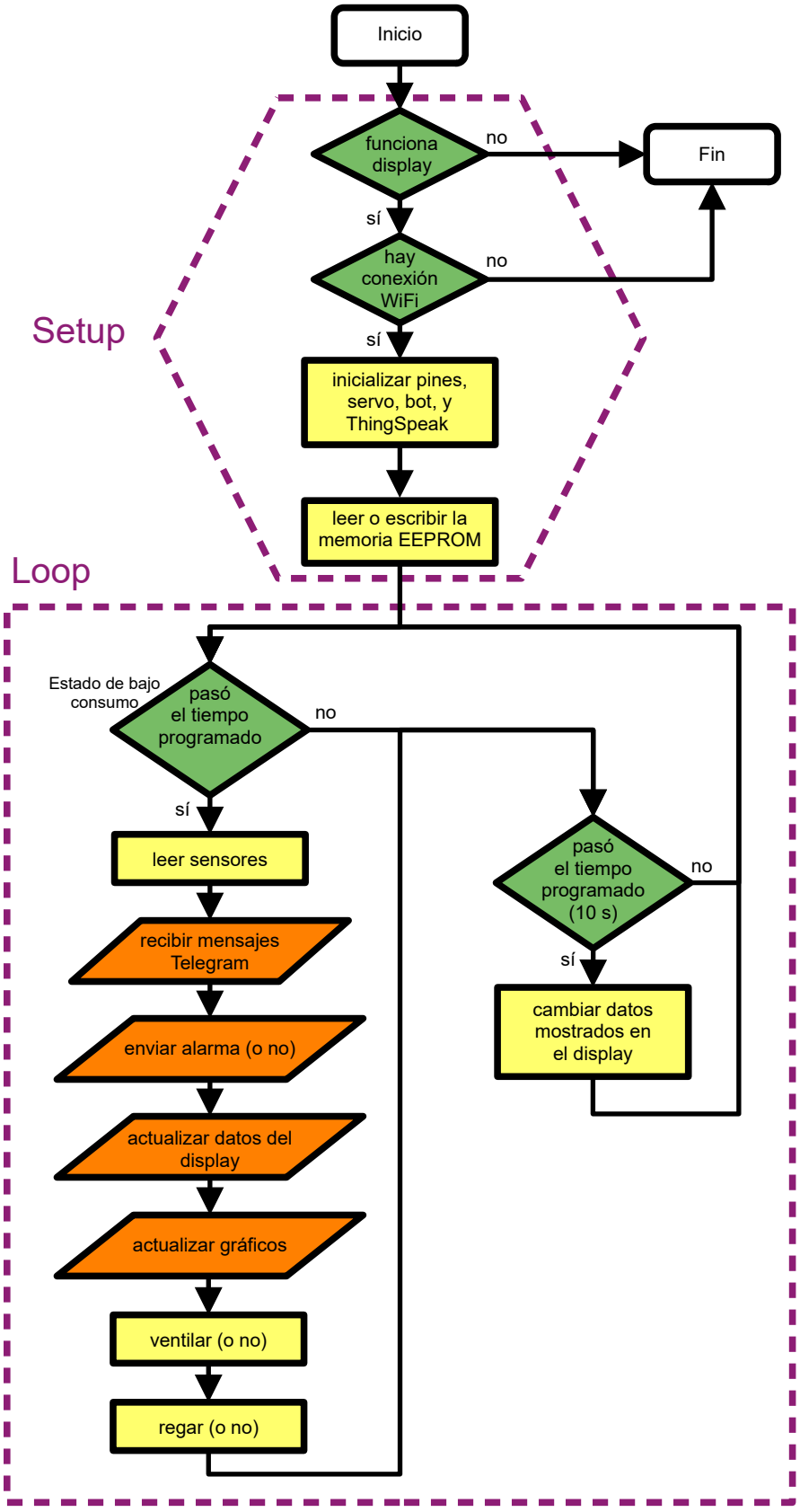

Figura 1: Diagrama de flujo del código.

4.1. Estructura y archivo “main.cpp”

La estructura del Diagrama de flujo se ve plasmada en el archivo “main.cpp” del código. En él, las dos funciones principales [setup() y loop()] de Arduino son llamadas, para ejecutar todos los procesos que tienen lugar en el invernadero, a través de llamadas a otras funciones (representadas por los distintos bloques del diagrama), definidas en los otros archivos “.h”. Setup() se ejecuta al encender el controlador, mientras que loop() se ejecuta durante el resto de su operación.

La estructura de la función loop() prevé la futura implementación de un modo de bajo consumo de energía por tiempo configurable, que es una cualidad brindada por el controlador utilizado (ESP-WROOM-32). Si bien el consumo del controlador no es alto, puede mejorarse mediante esta práctica.

5. EEPROM y datos configurables

El invernadero cuenta con 8 datos configurables por Telegram, los cuales son guardados en la memoria EEPROM del controlador. Todos ellos se almacenan entre ciclos de potencia, lo que quiere decir que incluso al apagar el controlador, se conservan hasta volver a ser cambiados por el usuario. Estos datos son (con sus valores predeterminados):

- Temperatura de ventilación: $35\ °C$.
- Humedad mínima del suelo: $60\ \%$.
- Tiempo de bombeo del riego: $10\ s$.
- Tiempo de espera luego del riego: $15\ min$.
- Activación de la alarma: activada.
- Lapso de envío de la alarma: $1\ h$.
- Temperatura mínima de alarma: $-5\ °C$.
- Temperatura máxima de alarma: $45\ °C$.

6. Telegram

Las tareas relacionadas con el bot de Telegram (comandos, alarma y conexión a WiFi) se encuentran programadas en el archivo “Telegram.h”.

En el proceso “recibir mensajes Telegram” (Figura 1), se llama a la función chequearMensajesRecibidosTelegram(), siendo esta la subrutina más extensa y compleja del programa. En ella, por cada mensaje nuevo recibido de tipo texto (no ubicación, contacto, u otro tipo), se hace lo siguiente:

1. Se marca el mensaje como leído y se obtiene el texto (Figura 2).

```
void chequearMensajesRecibidosTelegram() // en "loop()"
{
    TBMessage msj;
    // mientras hayan nuevos mensajes
    while (CTBotMessageText == Bot.getNewMessage(msj))
    {
        String texto = msj.text;
        String nombre_solicitante = msj.sender.firstName;
        respuesta = ""; // borramos lo que hayamos escrito en llamadas anteriores
        // imprimir el mensaje recibido
        imprimirln("Mensaje obtenido:");
        imprimirln(texto);

        bool respondiendo_grupo;
        if (msj.sender.id != msj.group.id) // si no se le habla por grupo, msj.group.id = msj.sender.id
        {
            chat_id = msj.group.id;
            respondiendo_grupo = true;
        }
        else
        {
            chat_id = msj.sender.id;
            respondiendo_grupo = false;
        }
```

Figura 2: Primeras líneas de la función chequearMensajesRecibidosTelegram()

2. Se obtiene la ID del emisor, y se comprueba si proviene de un grupo (Figura 2).
3. Opcionalmente, se rechaza al usuario por no estar autorizado.
4. De ser el primer mensaje, se envía una bienvenida.
5. Se evalúa el texto del mensaje para determinar si es un comando. De serlo, se ejecutan las acciones programadas para ese comando y se crea una respuesta, y de no serlo, se crea como respuesta un mensaje de error (ver título 7. “Comandos”).
6. Por último, se envía la respuesta.

6.1. Alarma

Actualmente, el bot es capaz de enviar una alarma por Telegram cada determinado tiempo (configurable; véase título 7.8. “/tiempoAl”), cuando detecta que la temperatura del invernadero se encuentra por fuera de los límites establecidos (también configurables. Véase títulos 7.11. “/tmax” y 7.12. “/tmin”). Esta alarma puede desactivarse (véase título 7.5. “/alarma”), y en futuras versiones fácilmente podrían programarse envíos por otros motivos, como la falta de luz.

```
void chequearAlarma() // en "loop()"
{
    // no hay problema con que lapso_alarma_mins sea uint16_t, se multiplica por un UL
    if (millis() - ultima_vez_alarma >= (lapso_alarma_mins * 60000UL)  &&  alarma_activada)
    {
        ultima_vez_alarma = millis();
        if (chat_id == 0) return; // el usuario debe hablar al Bot primeramente
        String mensaje;
        mensaje = "ALARMA: ";

        // evaluamos la temperatura
        if (temp_interior_promedio >= temp_max_alarma)
        {
            mensaje += "La temperatura del invernadero es excesivamente alta";
            enviarMensaje(chat_id, mensaje);
        }
        else if (temp_interior_promedio <= temp_min_alarma)
        {
            mensaje += "La temperatura del invernadero es excesivamente baja";
            enviarMensaje(chat_id, mensaje);
        }

    }
}
```

Figura 3: Función encargada de enviar la alarma por temperatura (Telegram.h).

7. Comandos

El bot cuenta con 14 comandos públicos y dos ocultos para el desarrollo. Si el mensaje recibido por el bot comienza por “/”, pero no pertenece a los comandos válidos, se entiende que fue a causa de un error, y por ello el bot responde “*El comando enviado no es válido. Envíe /start para ver las opciones*”. Cuando el mensaje recibido por Telegram no empieza por “/”, el bot no responde si el mensaje fue enviado por un grupo, con el fin de poder sostener conversaciones con los demás participantes sin la intervención del mismo. Si el texto proviene del chat individual, la respuesta será “*No ha enviado un comando. Envíe /start para ver las opciones*”. A continuación la lista de comandos:

7.1. /start

Este comando se envía automáticamente al iniciar la conversación con el bot, y al igual que el resto, pueden enviarse en cualquier momento. La respuesta del bot ante él es:

“*Puede enviar los siguientes comandos:*

/info : Para revisar el estado del invernadero y los parámetros guardados

/lecturas : Para obtener datos actuales e históricos de los sensores

/prog : Para efectuar órdenes y modificaciones”

7.2. /info

La respuesta ante este comando es el estado de: La alarma (si está activada), la ventilación, la humedad de la tierra, y otras salidas binarias (en esta versión del hardware, un LED rojo). Además, el bot envía las configuraciones que contenga en cuanto a: Humedad mínima del suelo, lapso de envío de alarmas, lapso de bombeo del riego, lapso de espera luego del riego, temperaturas de envío de la alarma, y temperatura de apertura de la ventilación (véase título 5. “EEPROM y datos configurables”).

7.3. /lecturas

Ante este comando, el bot responde con las lecturas de los sensores en cuanto a temperaturas y humedades del suelo y aire. Además, envía el siguiente link a una página web en la cual pueden verse los datos históricos en gráficos:

[https://thingspeak.com/channels/1937932]

7.4. /prog

Como respuesta a este comando, el bot envía la lista de todos comandos disponibles (excepto los ya descritos).

7.5. /alarma

Con este comando, puede activarse y desactivarse la alarma de temperatura del bot. La respuesta del bot es el nuevo estado de la alarma.

7.6. /hum

Este es uno de los comandos modificatorios de los parámetros del invernadero (véase título 5. “EEPROM y datos configurables”). Al enviarlo, el bot responde con el parámetro guardado actualmente, y espera 10 segundos para que el usuario envíe un nuevo valor. De no recibirlo, no se efectúan modificaciones, y de hacerlo, se almacena el nuevo dato. En caso de que el número ingresado no sea válido, el valor no se modificará, y se enviarán nuevas instrucciones.

7.7. /led

Actualmente este es un comando con fines demostrativos. Con él puede activarse y desactivarse un LED rojo en el controlador, con la intención de evidenciar el alcance de los comandos. La respuesta del bot es el nuevo estado del LED.

7.8. /tiempoAl

Este es otro de los comandos modificatorios, que se comporta igual al del título 7.6. “/hum”. Con él, puede cambiarse el lapso de tiempo que pasa entre cada envío de alarmas.

7.9. /tiempoRiego

Con este comando modificatorio, puede cambiarse el lapso de tiempo durante el cual se activa el riego, al detectarse una humedad del suelo muy baja.

7.10. /tiempoEspera

Con este comando modificatorio, puede cambiarse el lapso de tiempo que transcurre luego de cada riego, durante el cual se ignoran los sensores de humedad del suelo. La finalidad de este lapso de tiempo es dejar una pausa para que el agua se disperse en el suelo, para no efectuar el riego hasta que la humedad llegue a los sensores, momento en el cual podría haberse bombeado demasiada agua. Su duración dependerá de la disposición de los sensores, el tipo de riego y las características del suelo, entre otros factores.

7.11. /tmax

Con este comando modificatorio puede cambiarse el valor de temperatura por sobre el cual se envía una alarma al productor.

7.12. /tmin

Con este comando modificatorio puede cambiarse el valor de temperatura por debajo del cual se envía una alarma al productor.

7.13. /tvent

Con este comando modificatorio puede cambiarse el valor de temperatura por sobre el cual se abre la compuerta de ventilación y se activa el ventilador eléctrico del invernadero. Dichas salidas permanecen es ese estado hasta que la temperatura baje del valor establecido, o se las fuerce en el estado contrario (véase título de abajo).

7.14. /ventilar

Finalmente, el comando /ventilar cambia el estado en el que se encuentre la compuerta de ventilación y el ventilador. Al enviarlo, la ventilación queda “forzada” (cualquiera sea su estado), y se ignoran las lecturas de temperatura de los sensores. Esto se hace con el fin de obedecer al productor sin importar la lógica del invernadero, y puede ser útil, por ejemplo, cuando el controlador intente ventilar en condiciones en las cuales no sea óptimo hacerlo (vientos altos, lluvia u otras situaciones).

8. Mantenimiento y modificabilidad

Al estar dividido estratégicamente en ocho archivos de origen C Header (.h) y un archivo de origen C++ (.cpp), el código es fácilmente modificable, mantenible y expandible. Uno de los criterios para su división es la utilización de librerías. Así, de querer modificar un componente o mecánica (por ejemplo, el display), el único archivo a modificar sería el que contiene su librería y programación (en el ejemplo anterior, Display.h). O de querer añadir más y distintos sensores al controlador, el único archivo a modificar sería Sensores.h (y Control.h para las decisiones en base a él).

Por ello, pese a que el software actual está diseñado para una maqueta, es fácilmente modificable para trabajar en su ambiente final: el domo geodésico o cualquier invernadero diseñado para los productores regionales.