Nodos de ejemplo: Termostato

Este es uno de los nodos existentes junto con su código debidamente comentado.

Se presuponen conocimientos básicos sobre programación de Arduino.

Este nodo está diseñado para reemplazar a un termostato convencional.

El código está disponible en el siguiente enlace: https://github.com/turbolargoo/Free-Connect/tree/master/Ejemplos%20proporcionados/termostato

Figura 1: Nodo Termostato

Características

- Lectura de temperatura ambiente con precisión de 1 decimal.
- Lectura de humedad ambiente.
- Las mediciones pueden verse en tiempo real desde la aplicación o en la pantalla integrada.
- Programable desde la aplicación o desde los botones táctiles integrados.

Hardware empleado

- Placa esp32 con pantalla oled integrada y batería.
- Optoacoplador para manejar la señal de arranque de la caldera.
- Sensor DHT11 para la temperatura y humedad.

Librerías adicionales

- SSD1306Wire. Pantalla oled. Descargamos de `https://github.com/ThingPulse/esp8266-oled`
- DHTesp. Sensor DHT11. Disponible desde el gestor de librerías.

  `Versión 1.17.0 by beegee_tokyo`

Circuito eléctrico

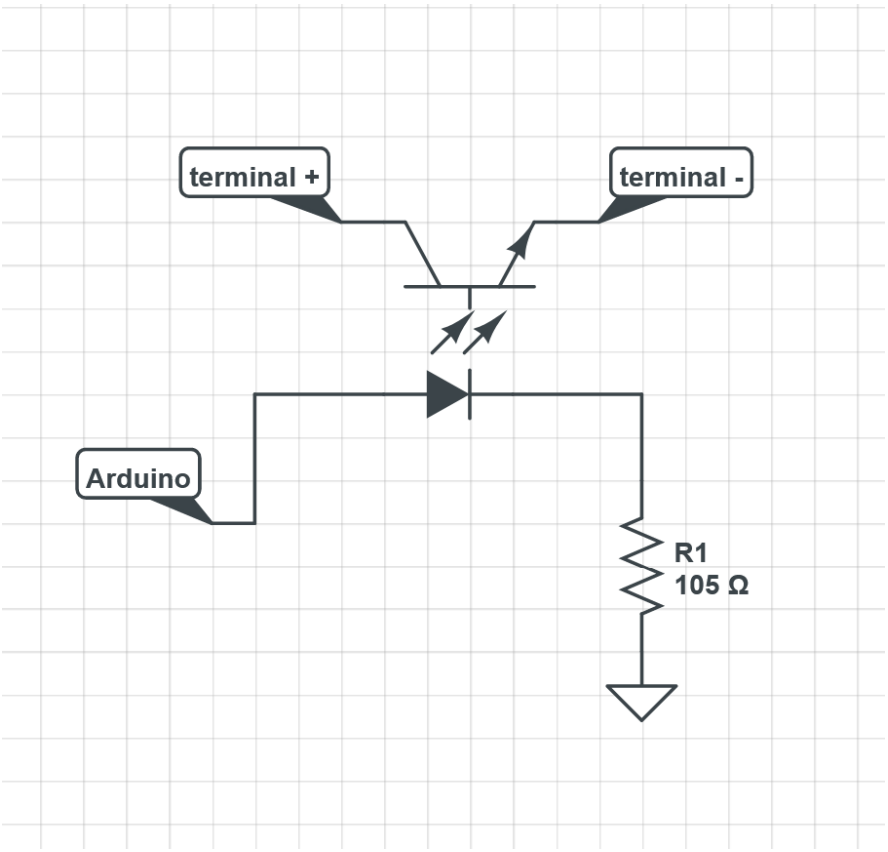

Figura 2: Esquema eléctrico del termostato

En la figura podemos ver el circuito eléctrico simplificado compuesto por un optoacoplador sharp PC817 una resistencia y nuestra placa esp32. El pin de activación definido en nuestro código se conecta a través de la resistencia correspondiente al led del optoacoplador. En el transistor del optoacoplador conectaremos el cable del termostato reemplazado. Es muy importante respetar la polaridad que deberemos haber comprobado antes con un polímetro.

Figura 3: Comprobación de polaridad

Programación

Partimos del código base para este desarrollo. A continuación se muestra el código añadido.

Archivo de configuración

Añadimos una serie de líneas:

1. Pines utilizados:

   const int heater =17;. Salida del optoacoplador y led integrado

   const int led =16;

   const int plusButton=15;. Botones

   const int minusButton=13;

   const int dhtPin = 18;. Ubicación del sensor

2. Variables utilizadas en el código.

3. Valores para la pantalla integrada y sensor.

4. Parámetros de configuración webthings:

   const char deviceName[] =”termostato”;. Nombre a mostrar

   *WebThingAdapter *adapter;*

   *const char *capabilities[] = ”Thermostat”,nullptr;*. Funcionalidades del nodo

   ThingDevice Thermostat(”termostato”, ”termostato”, capabilities);. Dispositivo

   ThingProperty Temperature(”Temperature”, ”Lectura del sensor”, NUMBER, ”TemperatureProperty”);. Propiedad 1, temperatura medida

   ThingProperty TemperatureObj(”TemperatureObj”, ”Temepratura objetivo”, NUMBER, ”TargetTemperatureProperty”);. Propiedad 2, temperatura objetivo

Setup

Añadimos *TemperatureObj.unit = ”degree celsius”;* en la sección de wethings para que se muestre correctamente la temperatura objetivo

Añadimos también la inicialización de el sensor y la pantalla junto con la declaración de las interrupciones para los botones.

Loop

El control de tiempos se ha realizado guardando el resultado de la función *millis()* y comparándolo con los valores predefinidos en la configuración en lugar de utilizar delays.

Figura 4: Código en bucle del termostato

Lo primero que encontramos es la llamada a la función de actualización inalámbrica. La función de refresco de datos se ha movido para controlar mejor su ejecución.

La sección marcada como 1 llama a la función encargada de leer los datos y a la encargada de actualizarlos.

La sección marcada como 2 comprueba si la temperatura objetivo es mayor que la leída por el sensor para dar la orden de calentar. Tiene en cuenta el tiempo que ha estado encendida o apagada para evitar múltiples arranques o paros seguidos.

La sección marcada como 3 corresponde con el código activado por las banderas que manejan los botones táctiles.

Funciones

En este archivo se encuentran las funciones llamadas desde *loop*.

El código relativo a la lectura de sensores y manejo de datos se considera demasiado básico como para ser comentado.

Actualización de datos para la interfaz:

En este caso la comunicación es bidireccional. La interfaz nos proporciona nuevos valores de temperatura objetivo y nosotros enviamos las lecturas de temperatura.

Acceso al valor de temperatura objetivo:

TemperatureObj.getValue().number;.

Envío de nuevo valor de temperatura ambiente leído:

ThingPropertyValue tempProp;. Creamos un objeto propiedad nueva para guardar el valor.

tempProp.number = tRead;. Guardamos el valor.

Temperature.setValue(tempProp);. Lo asignamos a nuestra propiedad.

adapter-¿update();. Actualizamos el adapatador.

Coste de este nodo

Descripción	Unidades	Coste unitario €	Coste total €
Placa esp32	1	15	15
Placa perfboard	1	2	2
Optoacoplador	1	0,25	0,25
Resistencias	1	0,1	0,1
Bornes	1	0,5	0,5
Sensor DHT11	1	5,5	5,5
IC carga	1	0,5	0,5
Impresión 3D	1	2	2

Tabla 1: Coste del nodo Termostato

Gastos totales de material: 25,85€