[Programa] SHT40 (exemplo I²C)

Objetivo:

Utilizar o *ESP32-C6-DevKitM-1* para ler valores de temperatura de um sensor SHT40.

Lista de material:

- *ESP32-C6-DevKitM-1*
- Breadboard
- Sensor SHT40
- Fios de ligação
- Cabo USB C

8_material_list.png
Image not found or type unknown

Contextualização:

Grove - Temperature & Humidity Sensor (SHT40)

O sensor de temperatura e humidade da Seeed SHT40 é um sensor digital com interface I²C padrão [2].

Inter-Integrated Circuit (I²C) [3]

O *Inter-Integrated Circuit* (I²C)[3] é um barramento serial que permite a co-existência de múltiplos controladores e periféricos (historicamente designados por *master/slave*) e considera essencialmente duas ligações:

SDA	Serial Data Line
SDL	Serial Clock Line

Ambas são bidirecionais e recorrem a resistências *pull-up*. As tensões típicas utilizadas são de +5 V ou +3,3 V, embora sejam permitidos sistemas com outras tensões.

O design de referência I²C possui um espaço de endereço de 7 *bits*, com uma extensão de 10 *bits* raramente utilizada. As velocidades de barramento I²C comuns são o modo padrão de 100 *kbit/s*

existindo outras possibilidades para casos específicos.

A comunicação I²C segue uma estrutura organizada em pacotes de dados. Cada transação no barramento I²C inicia com uma condição de S*tart*, seguida pelo envio de um *byte* de endereço, que identifica o dispositivo prerifférico com o qual o controlador deseja comunicar. Após o endereço, o controlador pode enviar ou receber dados, dependendo da natureza da transação (escrita ou leitura).

Os principais elementos dos protocolos de mensagem do I²C incluem:

1. **Condição de Início (S*tart*):** O controlador gera uma transição do sinal SDA de alto para baixo enquanto SCL está alto, indicando o início de uma comunicação.
2. ***Byte* de Endereço:** Após a condição de S*tart*, o controlador envia um *byte* de endereço que contém o endereço do periférico alvo e um *bit* de leitura/escrita. O periférico com o endereço correspondente responde com um *bit* de reconhecimento (*ACK*).
3. **Dados:** Os dados são transmitidos em *bytes*, seguidos por um *bit* de reconhecimento (*ACK*) enviado pelo receptor após cada *byte*. Se o receptor não reconhecer o *byte*, ele envia um *bit* de não reconhecimento (*NACK*).
4. **Condição de Paragem (*Stop*):** A comunicação termina com uma condição de *Stop*, onde o controlador gera uma transição do sinal SDA de baixo para alto enquanto SCL está alto. Isso sinaliza o fim da transação no barramento I²C.

Essa estrutura simples e eficiente permite que o I²C suporte a comunicação entre múltiplos dispositivos de forma organizada, minimizando a necessidade de fios e facilitando a integração em sistemas eletrónicos complexos.

Desenvolvido pela Philips Semiconductor em 1982, o I²C é ideal para aplicações que exigem comunicação eficiente e de curto alcance entre componentes eletrónicos. Este protocolo suporta a comunicação entre múltiplos dispositivos, onde o dispositivo contolador comanda o barramento I²C e os peféricos respondem aos comandos recebidos. A simplicidade do I²C, combinada com a sua flexibilidade, torna-o uma escolha popular para interligar sensores, displays, memória e outros periféricos a microcontroladores em diversos projetos.

De seguida apresenta-se uma aplicação prática do I²C usando a placa de desenvolvimento *ESP32-C6-DevKitM-1*, que possui suporte integrado para I²C, em conjunto com o sensor de temperatura e humidade SHT40.

Procedimento:

Atentando nos terminais 3, 4, GND e 5V do ESP32 como referência deve considerar-se o seguinte ilustração de montagem:

8_i2c_setup.png

Image not found or type unknown

Instruções:

- Montar o circuito esquematizado anteriormente
- Ligar a placa ESP32 ao computador por via de cabo USB C
- Abrir o IDE Arduino
- Usando a ferramenta de gestão de bibliotecas do IDE Arduino procurar e instalar a biblioteca **Adafruit SHT4X** ([1])
- Selecionar o dispositivo ESP32 adequado (*ESP32C6 Dev Module*)
- Copiar o seguinte

```
#include <Wire.h>
#include <Adafruit_SHT4x.h>

// Definição de uma nova instância da classe TwoWire
TwoWire myWire = TwoWire(0x44); // Usa o barramento I2C no ID 0 (podes alterar conforme
necessário)
Adafruit_SHT4x sht40 = Adafruit_SHT4x();

void setup() {
  digitalWrite(RGB_BUILTIN, LOW);  // Turn the RGB LED off
  Serial.begin(115200);

  // Inicia a instância myWire no barramento I2C utilizando pinos SDA e SCL personalizados
  myWire.begin(4, 3); // Pinos SDA=4 e SCL=3 (substitua pelos pinos que preferires)

  // Inicializa o sensor SHT40 usando a instância personalizada de TwoWire
  if (!sht40.begin(&myWire)) {
    Serial.println("Falha ao inicializar o SHT40. Verifique a conexão I2C.");
    while (1) {
      delay(10);
    }
  }
  sht40.setPrecision(SHT4X_HIGH_PRECISION);
  sht40.setHeater(SHT4X_NO_HEATER);
  Serial.println("Sensor SHT40 inicializado com sucesso!");
}

void loop() {
  digitalWrite(RGB_BUILTIN, LOW);  // Turn the RGB LED off
```

```
  sensors_event_t humidity, temp;
  sht40.getEvent(&humidity, &temp);

  Serial.print("Temperatura: ");
  Serial.print(temp.temperature);
  Serial.println(" °C");

  Serial.print("Humidade: ");
  Serial.print(humidity.relative_humidity);
  Serial.println(" %");

  delay(2000);  // Espera 2 segundos antes de nova leitura

  digitalWrite(RGB_BUILTIN, HIGH);  // Turn the RGB LED white
}
```

- Compilar e carregar o código para a placa ESP32
- Verificar o resultado

Image not found or type unknown	8_img_out_b.png Image not found or type unknown

Referências:

[1] Adafruit. Adafruit Sensirion SHT40, SHT41 & SHT45 Temperature & Humidity Sensors. url: https://learn.adafruit.com/adafruit-sht40-temperature-humidity-sensor (acedido em 29/08/2023).

[2] Seeed Studio. Grove - Temperature & Humidity Sensor. url: https://wiki.seeedstudio.com/Grove-SHT4x/ (acedido em 29/08/2023).

[3] Wikipedia. I2C. url: https://en.wikipedia.org/wiki/I%C2%B2C (acedido em 29/08/2023).

Revision #23
Created 29 August 2024 09:51:36 by João Pedro Monteiro
Updated 6 September 2024 09:34:13 by João Pedro Monteiro