UA - MISA

Afonso Rodrigues (107715), Gonçalo Lima (108254), João Rodrigues (108214), Luís Leal (103511), Paulo Macedo (102620), Tomás Fonseca (107245)
TP1 3 - Grupo 6

Microcontroladores e Interação com Sensores e Atuadores

23-06-2023

Relatório

Conteúdos

1 Introdução

O projeto proposto consiste no desenvolvimento de um sistema de monitorização de temperatura, utilizando o microcontrolador *Arduino UNO*, baseado no *ATmega328*. Esta placa versátil e de fácil utilização, em conjunto com um sensor de temperatura, um LCD e um módulo de comunicação *Bluetooth*, permite a criação de um sistema capaz de determinar a temperatura ambiente e transmitir essas informações para exibição, em tempo real. Tanto no *display* como em dispositivos externos que se encontrem conectados ao módulo.

Neste projeto, serão explorados os conceitos de programação do *Arduino*, integração de sensores e atuadores, bem como a comunicação sem fios através do módulo *Bluetooth*. Com isso, espera-se proporcionar uma experiência prática e

didática para os envolvidos, incentivando a aprendizagem e a aplicação de conhecimentos em eletrónica e automação.

2 Material utilizado

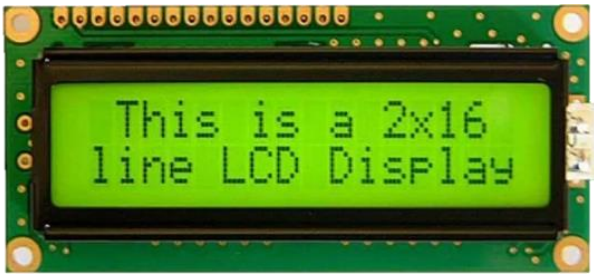

Fig. 1 – LCD

O LCD *(Liquid Crystal Display)* proporciona uma interface visual clara e legível, permitindo a exibição do valor da temperatura em tempo real. Dessa forma, os utilizadores podem facilmente visualizar a temperatura atual, sem a necessidade de se conectarem ao módulo *Bluetooth*.

Fig. 2 – Sensor de temperatura

O sensor de temperatura *TMP36GT9Z*, por sua vez, desempenha a função de capturar a temperatura ambiente com uma resolução de 10mV/°C e com uma precisão de ±3°C.

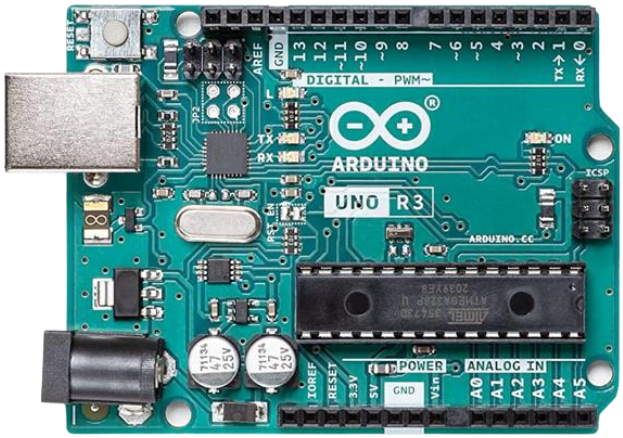

Fig. 3 – Microcontrolador Arduino

A placa de microcontrolador Arduino UNO é amplamente conhecida pela sua facilidade de uso e versatilidade. A placa utiliza o microprocessador *ATmega328* e esta possui recursos que permitem a leitura de sinais analógicos, como o valor de temperatura fornecido pelo sensor.

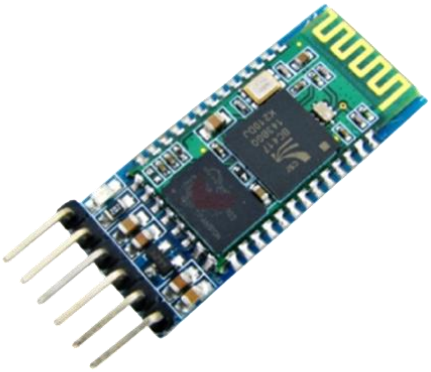

Fig. 4 - Módulo Bluetooth

O módulo *HC-05* permite a comunicação sem fio dos dados da temperatura para dispositivos externos. Com a comunicação Bluetooth, os valores de temperatura podem ser enviados para um dispositivo externo, como um PC ou telemóvel, possibilitando o monitoramento remoto da temperatura ambiente. Essa funcionalidade oferece flexibilidade e conveniência, permitindo que os utilizadores acompanhem as variações de temperatura de forma remota.

3 Configuração e Implementação

3.1 Configuração

3.1.1 LCD

A configuração do *display* foi feita da seguinte forma:

```
LiquidCrystal lcd(13, 12, 5, 4, 3, 2);
```

"`LiquidCrystal`" é uma classe fornecida pela biblioteca *LiquidCrystal*, importada no início do código, que representa o *display* e fornece funções para controlar o mesmo. Neste caso, "`lcd`" foi o nome dado ao objeto.

A biblioteca foi importada da seguinte forma:

```
#include <LiquidCrystal.h>
```

Os parâmetros passados no construtor da classe "`LiquidCrystal`" são '`(13, 12, 5, 4, 3, 2)`' e estes representam as seguintes ligações:

- Pino 13: LCD RS (Register Select)
- Pino 12: LCD Enable
- Pino 5: LCD D4
- Pino 4: LCD D5
- Pino 3: LCD D6
- Pino 2: LCD D7

3.1.2 Timer

O *timer* utilizado para este projeto foi o *Timer 1*, que foi configurado da seguinte forma:

```
void setup_timer_1(void) {
            // Timer1 Configuration
            TCCR1A = 0; // Clear Timer1 control register A
            TCCR1B = 0; // Clear Timer1 control register B

            // Set Timer1 to CTC mode with prescaler 64
            TCCR1B |= (1 << WGM12) | (1 << CS12);

            // Set compare value for desired frequency on
            // Timer1
            OCR1A = 31249; // (16MHz / (256 * 2)) - 1
```

```
        // Enable Timer1 compare match interrupt
        TIMSK1 |= (1 << OCIE1A);
    }
```

Em primeiro lugar, os registos de controlo, A e B, do *timer* foram definidos com o valor '0', limpando qualquer configuração anterior. De seguida, o *timer* é configurado com o modo de comparação 'CTC' e é definido um *prescaler* com o valor 64. O valor de comparação utilizado para o *timer* é calculado dividindo a frequência do *clock* da placa (16MHz) pelo valor máximo que o contador pode atingir antes de ser reiniciado (2^8 * 2, '8' corresponde a resolução do *Timer 1*, em bits). Uma vez que o contador inicia a contagem em '0', é subtraído '1' ao resultado da operação anterior.

Desta forma, são configurados o modo de operação, o *prescaler* e o valor de comparação para controlar a frequência das interrupções.

3.1.3 Módulo Bluetooth

De forma a configurar o módulo *HC-05*, foi importada a biblioteca *SoftwareSerial,*

```
#include <SoftwareSerial.h>
```

e, de seguida, foi instanciada a classe "`SoftwareSerial`" com os parâmetros "rxPin" e "txPin", definidos com os valores '10' e '11', respetivamente.

```
#define rxPin 10
#define txPin 11

SoftwareSerial bltSerial(rxPin, txPin);
```

Desta forma, os pinos 10 e 11 são configurados para a comunicação, permitindo a receção e envio de informação por parte do módulo, utilizando o objeto "bltSerial".

3.1.4 Setup

Na função "setup", são feitas as configurações iniciais, que antecedem a execução do programa principal. Por ordem, as configurações feitas foram as seguintes:

- *Pins* "rxPin" e "txPin" configurados como entrada e saída, respetivamente, permitindo que estes possam receber (rxPin) e enviar (txPin) dados para um dispositivo externo que se encontre conectado ao módulo;

- A comunicação série é iniciada com uma *baudrate* cujo valor é '9600', o que permite uma velocidade de transmissão de dados de 9600 *bits*/s;
- O objeto "lcd" é inicializado, ativando as 16 colunas e as 2 linhas disponíveis no mesmo;
- É chamada a função "setup_timer_1", descrita anteriormente, cuja finalidade é configurar o *Timer 1*;
- As interrupções globais são ativadas, permitindo à placa responder a interrupções de *hardware*.

Código correspondente à função "setup":

```
void setup(void)
{
        pinMode(rxPin, INPUT);
        pinMode(txPin, OUTPUT);

        bltSerial.begin(9600);
        lcd.begin(16, 2);

        setup_timer_1();

        interrupts();
}
```

3.2 Implementação

3.2.1 Constantes e Variáveis

De forma a ler os dados obtidos pelo sensor de temperatura, é inicializada uma constante de um valor inteiro chamada "sensorPin", à qual será atribuído o valor 'A0', que corresponde ao primeiro *pin* analógico da placa, ao qual o sensor se encontra ligado.

São também inicializadas, com o valor '0.0', duas variáveis do tipo *float,* cujo propósito será armazenar o valor da temperatura lido do sensor, variável "temperature", e armazenar o valor de temperatura mais alto, registado até ao momento, variável "highestTemp". Para armazenar o valor mais baixo de temperatura, foi inicializada uma variável do tipo *float*, com o valor '100.0'.

```
const int sensorPin = A0;
Volatile float temperature = 0.0;
float highestTemp = 0.0;
float lowestTemp = 100.0;
```

A designação “Volatile” é usada para indicar que uma variável pode mudar o seu valor, inesperadamente, devido a fatores externos que estão fora do controlo do programa. Dá a conhecer ao compilador que a variável não deve ser otimizada ou armazenada em cache nos registos, devendo ser sempre lida e gravada em memória.

3.2.2 Rotina de Serviço à Interrupção

A frequência de leitura dos valores do sensor de temperatura é dada pelo *Timer* 1. Quando este gera uma interrupção o programa entra na Rotina de Serviço à Interrupção, específica para este *timer.*

Dentro desta rotina, o valor do sensor (naquele instante) é lido para a variável “SensorVal”, este valor corresponde à conversão analógica para digital pelo ADC do *Arduino*, cuja resolução é de 10 bits, estando então este valor no intervalo 0 a 1023.

De seguida, este valor é convertido numa tensão real, de 0 a 5 volts, que é guardada na variável “voltage”. Dado que o sensor utilizado tem uma resolução de 10mV/ºC, esta tensão é facilmente convertida para graus Celsius, multiplicando-a por 100. Para diminuir o erro da medição do sensor, subtraímos 0,5 ao valor da tensão real (“voltage”) antes desta multiplicação. Este valor, em graus Celsius, é guardado na variável “temperature”.

No final da rotina, é feito o *reset* da *flag* de interrupção do *Timer* 1.

Código correspondente à Rotina de Serviço à Interrupção:

```
ISR(TIMER1_COMPA_vect) {
  int SensorVal = analogRead(sensorPin);
  float voltage = (SensorVal / 1024.0) * 5.0;
  temperature = (voltage - 0.5) * 100;
  TIFR1 |= (1 << OCF1A);
}
```

3.2.3 Loop

A função “loop” é a função principal do programa, na qual é programada toda a lógica correspondente à receção, envio e tratamento de dados.

Em primeiro lugar, é feito um *reset* ao *display*, de forma a limpar qualquer tipo de informação que tenha sido previamente armazenada neste. O cursor é movido para a posição inicial, coluna 0 e linha 0, e as restantes linhas de código permitem dar *print* à mensagem “Temperature: {temperature}ºC”, sendo que a variável “temperature” é substituída pelo valor armazenado aquando da execução da “ISR”.

```
lcd.clear();
lcd.setCursor(0, 0);
```

```
        lcd.print("Temperature:");
        lcd.setCursor(0, 1);
        lcd.print(temperature);
        lcd.createChar(6, customShape);
        lcd.setCursor(6, 1);
        lcd.write((byte)6);
        lcd.print("C");
```

Importa referir que, para ser possível dar *print* ao caracter 'º', foi necessário criar o *array* do tipo *byte*, cujo nome é "customShape".

```
byte customShape[] = {
          B00000,
          B00111,
          B00101,
          B00111,
          B00000,
          B00000,
          B00000,
          B00000
      }
```

Quanto à lógica associada à atualização dos valores correspondentes à temperatura máxima e mínima registadas, esta processa-se da seguinte forma:

- Quando a temperatura lida pelo sensor e armazenada na variável "temperature" é superior ao valor armazenado na variável "highestTemp", a variável que armazena a temperatura mais alta é atualizada para a o valor armazenado em "temperature":

```
if (temperature > highestTemp)
          {
               highestTemp = temperature;
          }
```

- No caso da lógica associada à temperatura mais baixa, esta processa-se da mesma forma que no caso anterior, com a diferença de que o valor da variável "lowestTemp" é atualizado quando o valor armazenado em "temperature" é inferior ao de "lowestTemp".

```
if (temperature < lowestTemp)
          {
               lowestTemp = temperature;
          }
```

De forma a tornar possível a transmissão, para os dispositivos conectados ao módulo *Bluetooth*, dos valores registados, é efetuado um ciclo *polling* que verifica a disponibilidade do módulo.
Neste ciclo, é criada a variável "receiveData", do tipo *char*, que irá armazenar a informação transmitida pelo terminal do dispositivo conectado e, dependendo do caracter recebido, a informação enviada será diferente. Caso a placa receba o caracter 'c' *(current)*, irá enviar para o dispositivo conectado o último valor registado pelo sensor, caso receba um 'h' *(highest)*, irá enviar a temperatura mais alta e, caso receba um 'l' *(lowest)*, irá comunicar a temperatura mais baixa, registada até ao momento.

```
while (bltSerial.available() > 0)
  {
    char receiveData = (char)bltSerial.read();

    switch (receiveData)
    {
    case 'c':
      bltSerial.print(temperature);
      bltSerial.println(" °C");
      break;
    case 'h':
      bltSerial.print(highestTemp);
      bltSerial.println(" °C");
      break;
    case 'l':
      bltSerial.print(lowestTemp);
      bltSerial.println(" °C");
    default:
      break;
    }
  }
```

Por último, é introduzido um *delay* de 100 milissegundos entre cada iteração da função "loop".

```
delay(100);
```

3.2.4 Esquema

De modo a elaborar um esquema do circuito para representar todas as ligações feitas no sistema, entre a placa *Arduino*, o LCD, o sensor de temperatura e o módulo de *bluetooth*, usou-se uma plataforma de modulação e simulação de

circuitos. No entanto, esta ferramenta não disponibiliza o módulo de *bluetooth,* graficamente.

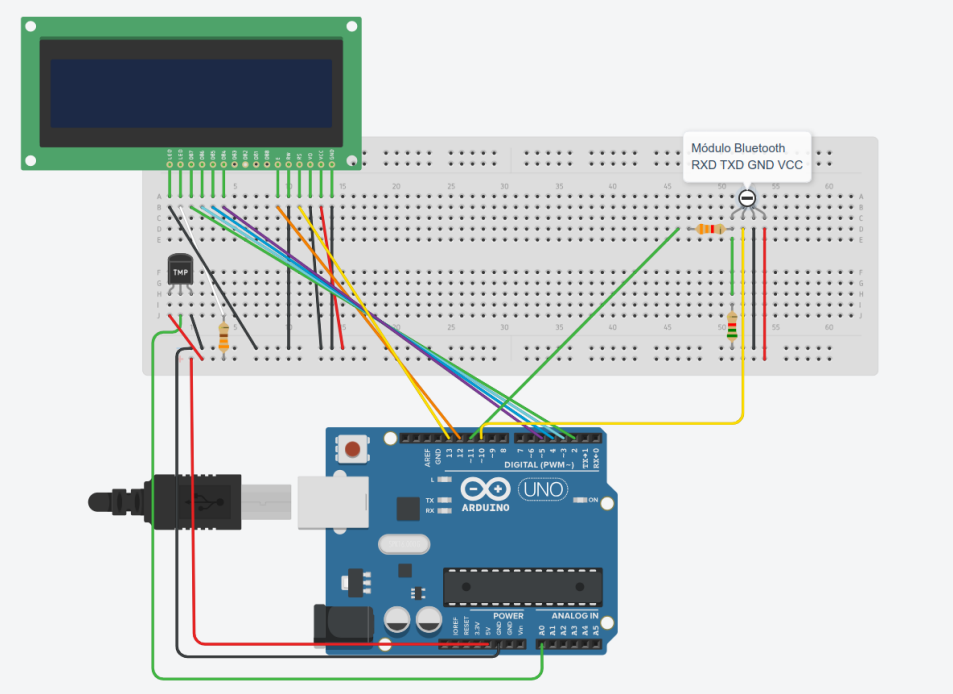

Fig. 5 – Esquema de montagem

4 Aspetos finais e observações

Com este projeto apresentamos o funcionamento de um sensor de temperatura em conjunto com o uso de um atuador (LCD) e um módulo Bluetooth, de forma a determinar a temperatura ambiente e transmitir essa informação para o *display* e dispositivos externos.

Durante a realização do projeto houve falhas na transmissão do módulo Bluetooth HC-05, onde ocasionalmente o módulo de Bluetooth envia incorretamente os dados da temperatura, e consequentemente, não apresenta os valores da temperatura nos dispositivos externos. Foi realizada a depuração do protocolo de comunicação UART (*baudrate* e formato dos dados) e configuração do módulo de *bluetooth*, no entanto, não foi possível identificar a fonte de erro.

Apesar de o problema apresentado, podemos concluir que atingimos os objetivos propostos, visto que concretizamos o projeto inicialmente idealizado.

Importante referir, que a realização deste trabalho permitiu-nos explorar o mundo dos microcontroladores e aumentar o nosso conhecimento acerca da interação do microcontrolador com os sensores e atuadores.

Demonstração : https://youtu.be/6VU_5wQKzcs