UPDF
WWW.UPDF.COM

Sistemas Embebidos

Trabalho Prático 7

Carlos Abreu[1] e João Faria[2]

[1]**cabreu@estg.ipvc.pt**

[2]**joao.pedro.faria@estg.ipvc.pt**

Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Viana do Castelo

Instituto Politécnico de Viana do Castelo
Escola Superior de Tecnologia e Gestão
2024

Carlos Abreu
www.estg.ipvc.pt/~cabreu

Curso:

Licenciatura em Engenharia de Redes e Sistemas de Computadores

UPDF
WWW.UPDF.COM

Objetivo Pedagógico

Protocolos de comunicação série: USART e I²C.

Sumário: **Duração:** 3 horas

1. Entender o modo de funcionamento da USART (*Universal Synchronous and Asynchronous serial Receiver and Transmitter*) no Atmega328
2. Entender o modo de funcionamento dos protocolo I²C (*Inter-Integrated Circuit*)
3. Interligar MCUs através do barramento série I²C no modo Um Master - Múltiplos Slaves.
4. Implementar um sistema básico de troca de mensagens entre MCUs num sistema embebido: Computação por Passagem de Mensagens

1. USART (*Universal Synchronous and Asynchronous serial Receiver and Transmitter*)

A *USART* tem sido usada ao longo das aulas práticas para estabelecer um canal de comunicação entre o Arduino e a *Consola* através da ligação USB disponível no Arduino (figura 1).

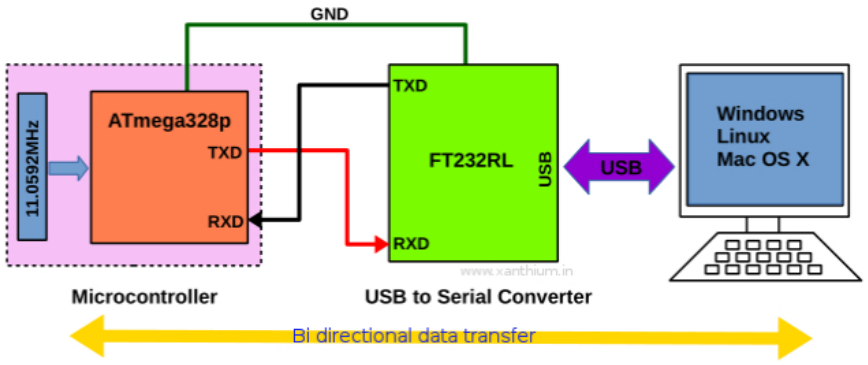

Figura 1

Diagrama de blocos da ligação do Atmega328 à porta USB de um computador.

UPDF
WWW.UPDF.COM

O modo de operação da $USART0$ no Atmega328 encontra-se descrito na secção 20 do datasheet disponível no Moodle. Abra o datasheet e identifique:

- O formato de trama usado pela USART. Quais os campos que a compõem?
- Quais os registos usados para configurar o *baudrate* que é passado como entrada à função Serial.begin().
- Qual o registo para ativar o receptor na USART?
- Qual o registo para ativar o transmissor na USART?

2. USART (*Universal Synchronous and Asynchronous serial Receiver and Transmitter*) - Exercicio

Exercício 1 *A utilização da porta série para comunicação entre uma placa Arduino e um computador ou outros dispositivos pode ser feita com recurso às funções* Serial*. Estas funções não são mais do que* device drivers *que permitem utilizar um protocolo série e controlar a USART abstraindo o utilizador dos detalhes do* hardware*. Neste exercício pretende-se implementar uma solução para o exercício 1 do trabalho prático 5 sem a utilização das funções* Serial*: Implementar uma solução que permita enviar através da consola um código de cor entre 0 e 6 para configurar a cor pretendida no LED. Sempre que o utilizador inserir um novo código a cor do LED deve ser alterada em conformidade.*

1.1 *Importe para a sua área de trabalho o circuito disponibilizado neste link e analise o código fornecido.*

1.2 *Implemente a função* initSerial() *para configurar a porta série com um* baud rate *de 9600 bps, 8 bits de dados, sem paridade e um stop bit.*

1.3 *Implemente a função* rxByteAvailable() *para aferir a existência de informação recebida na porta série.*

1.4 *Implemente a função* rxByte() *para devolver a informação recebida na porta série.*

1.5 *Teste o código implementado e não se esqueça de o comentar convenientemente.*

1.6 *Faça uma cópia do circuito/código implementado nos pontos anteriores e substitua a utilização da função* rxByteAvailable *por uma solução baseada em interrupções:*

```
ISR(USART_RX_vect) {
  ...
}
```

UPDF
WWW.UPDF.COM

3. I²C (*Inter-Integrated Circuit*) - Introdução

O protocolo I²C permite a comunicação série entre dois MCUs e circuitos integrados dedicados. É utilizado quando a distância entre os dispositivos é curta, normalmente na mesma placa de circuito impresso. A ligação entre dispositivos é estabelecida através de dois condutores:

- SDA (*Serial Data*) - dedicado à transferência de dados
- SCL (*Serial Clock*) - utilizado para sincronização (sinal de relógio).

Na figura 2 está representado um exemplo de utilização, com um dispositivo configurado como *Master* e os outros a operar com *Slaves*. O dispositivo *Master* efetua o endereçamento de um circuito ou MCU a operar em modo *Slave* antes do início da comunicação.

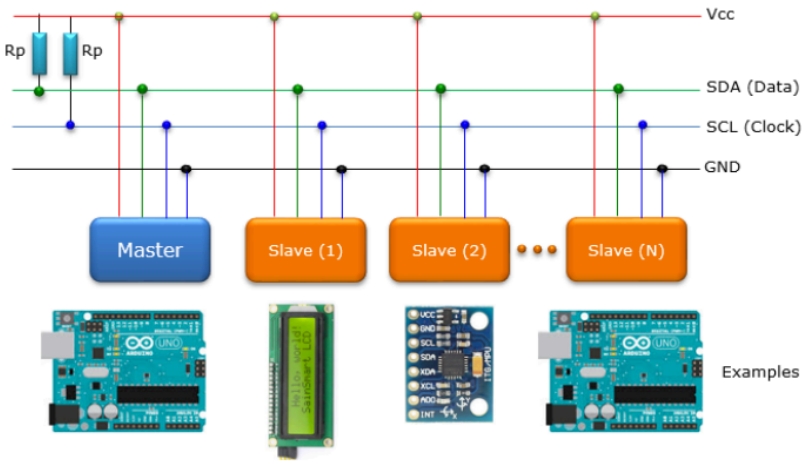

Figura 2

Exemplo de um barramento I²C

Nesse sentido, um MCU pode comunicar com múltiplos dispositivos diferentes. A taxa de transmissão pode ser de 100 Kbps (modo standard), 400 Kbps (modo rápido) ou 3,4 Mbps (modo de alta velocidade). O dispositivo *Master* gera e impõe o seu sinal de relógio na comunicação.

O protocolo pode ser implementado com recurso a software para controlar 2 pinos GPIO (técnica habitualmente designada *Bit Bang*) mas é comum existirem módulos de hardware dedicados à implementação do protocolo. Este módulo encontra-se descrito na secção 22 (*2-wire Serial Interface*) do *datasheet* do Atmega328 disponível no Moodle.

UPDF
WWW.UPDF.COM

A biblioteca *Wire* permite utilizar o protocolo I²C no Arduino e no Arduino UNO os pinos ligados ao módulo de hardware do Atmega328 são:

- SDA: pino A4
- SCL: pino A5
- GND: Não esquecer de ligar o sinal de *ground*, entre os diversos dispositivos

4. I²C (*Inter-Integrated Circuit*) - Exercícios

Exercício 2 *Pretende-se implementar um sistema composto por um* Master *e dois* Slaves *que permita a troca de mensagens utilizando o protocolo I²C.*

2.1 *Importe para a sua área de trabalho o circuito disponibilizado neste link e identifique o dispositivo* MASTER *e os dispositivos* SLAVE. *Escolha e configure os respetivos endereços nos três MCUs.*

2.2 *Considere os seguinte bloco de código C usado para a escrita do* MASTER *num* SLAVE *e edite a função loop() do dispositivo* Master *de modo a enviar para o* Slave_A *a sting* "MASTER > SLAVE_A\n"*:*

```
//////////////////////////////
// MASTER WRITES to SLAVE_A
//////////////////////////////
Wire.beginTransmission(SLAVE_A); // begin transmission to Slave A
Wire.write("MASTER > SLAVE_A\n"); // sends a string
Wire.endTransmission(); // stop transmitting
```

2.3 *Após a comunicação efetuada no ponto anterior configure o MASTER para esperar um segundo e de seguida considere os seguinte bloco de código usado para a leitura de dados de um* SLAVE *e edite a função loop() do* MASTER *de modo a que este receba uma string após ser efectuado um request:*

```
//////////////////////////////
// MASTER READS from SLAVE_A
//////////////////////////////
Wire.requestFrom(SLAVE_A,17); // request 17 bytes from slave device #8
while (Wire.available()) { // slave may send less than requested
char c = Wire.read(); // receive a byte as character
Serial.print(c); // print the character
```

2.4 *No* SLAVE_A *inicialize os dois eventos que irão ser usados no processo de comunicação na função setup():*

UPDF
WWW.UPDF.COM

```
void setup() {
    Serial.begin(9600);
    Wire.begin(SLAVE_A); // join i2c bus with address SLAVE_B
    Wire.onRequest(requestEvent); // register onRequest event
    Wire.onReceive(receiveEvent); // register onReceive event
}
```

2.5 *No* SLAVE_A *adicione os respetivas funções que são disparadas quando ocorrem os eventos definidos antes:*

```
void requestEvent() {
    Wire.write("SLAVE_A > MASTER\n"); // respond with message of 17 bytes
    // as expected by master
}
void receiveEvent(int numBytes) {
    while (Wire.available()) { // loop through all received bytes
        char c = Wire.read(); // receive a byte as a character
        Serial.print(c); // print the character
    }
}
```

2.6 *Teste o código implementado para a leitura e escrita do* MASTER *no* SLAVE_A *e não se esqueça de o comentar convenientemente.*

2.7 *Repita o procedimento de 3) a 7) de forma a configurar agora o* SLAVE_B*;*

2.8 *Altere o código do* MASTER *de forma a que este efectue uma escrita seguida de uma leitura do* SLAVE_A*, espere um segundo e de seguida efectue uma escrita seguida de uma leitura do* SLAVE_B*, espere um segundo. Deverá repetir este procedimento indefinidamente;*

2.9 *Teste o código implementado e não se esqueça de o comentar convenientemente.*

2.10 *Desenhe um diagrama de blocos que represente o código desenvolvido para o* MASTER *e para o* SLAVE_A*.*

Exercício 3 *Pretende-se implementar um sistema de controlo de temperatura usando computação por passagem de mensagens de acordo com os seguintes requisitos:*

- *O controlador (*MASTER*) deverá a cada segundo efetuar um* request *ao sensor (*SLAVE_B*) e de seguida atuar no led RGB (*SLAVE_A*).*

UPDF
WWW.UPDF.COM

Temperatura	*Código Cor*
$75 < T \leq 125$	*0 - LED_RGB(255, 0, 0)*
$50 < T \leq 75$	*1 - LED_RGB(255, 127, 0)*
$25 < T \leq 50$	*2 - LED_RGB(255, 255, 0)*
$10 < T \leq 25$	*3 - LED_RGB(0, 255, 0)*
$0 < T \leq 10$	*4 - LED_RGB(0, 0, 255)*
$-20 < T \leq 0$	*5 - LED_RGB(75, 0, 130)*
$-40 < T \leq -20$	*6 - LED_RGB(143, 0, 255)*

Tabela 1 Representação cromática da temperatura.

- *Mediante o valor da temperatura medida pelo sensor (*SLAVE_B*) o controlador (*MASTER*) do sistema deverá actuar o led RGB (*SLAVE_A*) de acordo com a tabela 1.;*
- *O sensor de temperatura (*SLAVE_B*) deverá efetuar uma medida de temperatura sempre* on request*;*
- *O controlador deverá imprimir o estado do sistema na consola sempre que for lido um novo valor do sensor (*SLAVE_B*) e sempre que o led RGB (*SLAVE_A*) for atualizado.;*

3.1 *Faça uma cópia do circuito do exercício anterior e implemente uma solução em arduino C que cumpra com os requisitos apresentados. Para isso:*

- *Configure o* SLAVE_A *como um led RGB com interface* I^2C.
- *Configure o* SLAVE_B *como um Sensor de Temperatura análogo ao implementado no trabalho prático #6 com interface* I^2C.
- *Configure o* MASTER *como controlador do sistema embebido.*

3.2 *Teste o código implementado e não se esqueça de o comentar convenientemente.*