FACULDADE ENGENHARIA

Departamento de
Engenharia Eletromecânica

Microprocessadores

Relatório nº2

Resumo: Este relatório descreve o desenvolvimento de um projeto de controlo de temperatura numa estufa usando o microcontrolador ATmega2560 e a linguagem assembly. O objetivo foi criar um sistema eficiente e confiável para manter a temperatura, os níveis de humidade e CO2 dentro de uma faixa desejada.

Docente:

- **António Pinheiro**

Alunos:

- **José Rita, nº42343**

Índice

Introdução 5
Objetivos 7
Funcionamento 7
Procedimento 9
Código – Controlo de Temperatura 14
Esquema de montagem 17
Conclusão 18

Índice de Figuras

Figura 1 – Arduíno ATMEGA2560 6
Figura 2 - Diagrama do Sistema a conceber 8
Figura 3 - Fluxograma 9
Figura 4 - ADLAR 11
Figura 5 - MUX 11
Figura 6 - ADCSRA 11
Figura 7 - ADEN 11
Figura 8 - ADSC 12
Figura 9 - ADIE 12
Figura 10 - SMCR 13
Figura 11 - Esquema de montagem 17

Índice de Tabelas

Tabela 1 – ADMUX e seleção de bits de referência 10
Tabela 2 - Bits de seleção ADPS 12
Tabela 3 - Sleep Mode 13

Introdução

Este relatório descreve o desenvolvimento de um projeto para o controlo da temperatura, dos níveis de humidade e CO2 numa estufa, utilizando o microcontrolador ATmega2560 e a linguagem de programação assembly.

A temperatura é um fator crucial em ambientes que exigem que certas condições, como as estufas, é daí que a monitorização precisa e o controlo adequado podem ser determinantes.

O microcontrolador ATmega2560, pertencente à família AVR, com o uso deste em conjunto com a programação em assembly leva a um controlo mais refinado e direto sobre o hardware, permitindo um ajuste fino do sistema de controlo de temperatura e os outros parâmetros.

Este relatório apresentará em detalhe o processo de desenvolvimento do projeto, incluindo o projeto do circuito, a programação em assembly para o controlo da temperatura, níveis de humidade e CO2 bem como os resultados obtidos.

ATMEGA 2560

Para a sintetização de conceitos que foram dados ao longo do semestre sobre microprocessadores foi nos proposto um último trabalho em que iríamos desenvolver um código de assembly através de um microcontrolador.

O microcontrolador que foi usado para a realização do trabalho, é o da figura (1):

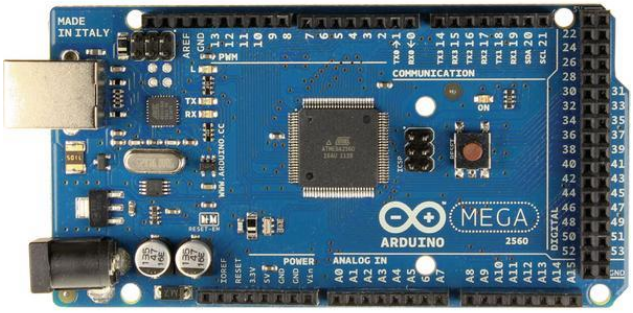

Figura 1 – Arduíno ATMEGA2560

O ATmega2560 é um microcontrolador da família AVR desenvolvido pela Microchip Technology, possui uma arquitetura RISC (Reduced Instruction Set Computing) de 8 bits que executa instruções num único ciclo de clock, e permite um bom balanço entre consumo de energia e de velocidade de processamento o que torna uma boa escolha para o nosso trabalho.

Objetivos

Para o último trabalho foi me proposto o trabalho nº 7, neste trabalho tinha de desenvolver um programa em assembly no Arduíno ATmega 2560 que permitia o controlo da temperatura, o controlo dos níveis de CO2 e o controlo dos níveis de humidade presentes numa estufa e executar certas funções em certos casos.

O meu trabalho envolvia uma estufa equipada com um ar condicionado e uma janela controlada:

- O objetivo do trabalho é de manter a temperatura, a humidade e o nível de CO2 dentro de limites predefinidos.
- O sistema possui três portos que definem os valores máximos para temperatura, humidade e CO2.
- E quatro sensores que medem a temperatura interna, temperatura externa, humidade da estufa e nível de CO2. Os sensores são lidos por conversores analógico-digitais (ADC's).

Funcionamento

Casos de controlo:

- Se a temperatura atingir o valor máximo -> a janela deve ser aberta, caso a temperatura externa seja mais baixa que a interna
- Caso contrário, o ar condicionado deve ser ligado para arrefecer o ambiente
- Há que ter em conta que em condições normais o ar condicionado opera no modo de aquecimento
- Se a humidade ou o nível de CO2 excederem os valores estabelecidos, a janela deve ser aberta

Operação da janela:

- A operação da janela e do ar condicionado é controlada por um porto de saída chamado PCTRL
- O bit mais significativo do PCTRL controla a abertura (nível baixo) ou o fechamento (nível alto) da janela da estufa

Operação do ar condicionado:

- O segundo bit menos significativo liga o ar condicionado em modo de aquecimento (nível baixo)
- Enquanto que o bit menos significativo liga o ar condicionado em modo de arrefecimento (nível baixo)

Na figura (2) é possível observar o sistema a ser concebido para o trabalho:

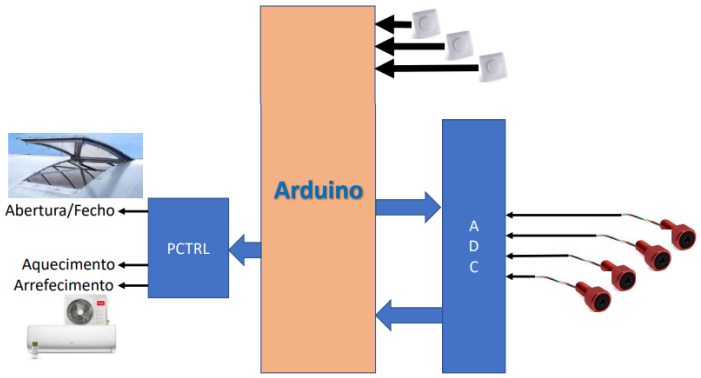

Figura 2 - Diagrama do Sistema a conceber

Mas este trabalho não foi totalmente concebido e apenas foi implementado o controlo de temperatura e as operações de controlo consequentes do controlo da temperatura. Que seriam a abertura de janela e a ativação do ar condicionado no modo de arrefecimento.

Procedimento

Para o desenvolvimento do código foi importante ter em mente um esquema de como iria funcionar o código, então foi desenvolvido um fluxograma para seguir e implementar na integra para a resolução do problema que tinha em frente.

Na figura (3) podemos ver o fluxograma que foi usado para ter uma lógica para a realização do trabalho:

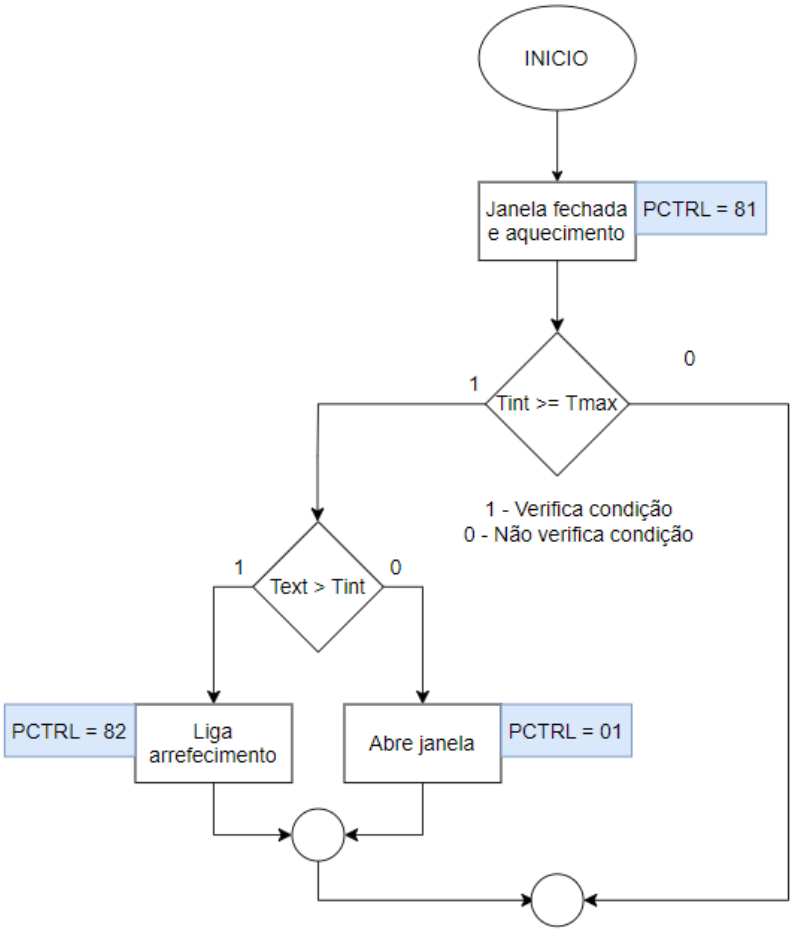

Figura 3 - Fluxograma

Explicação:

- O estado inicial será de janela fechada e o ar condicionado em aquecimento visto que a estufa está sempre em aquecimento

- Após isso a primeira verificação será verificar se a temperatura máxima (Tmax) já foi atingida, se for vamos verificar a exterior e agir conforme os seus valores

- Caso contrário apenas seguiríamos para o controlo dos níveis de CO2 e humidade

- Na comparação da temperatura exterior com a temperatura interior, iremos abrir a janela caso temperatura exterior seja menor que a interior. Se não vamos ligar o arrefecimento.

Para este trabalho é necessário ter em conta vários pormenores como a utilização e ativação dos ADC's bem como a utilização, e inicialização do modo SLEEP para apenas fazermos as leituras quando é feita a conversão do ADC.

Para a conversão analógico-digital e digital-analógico foi preciso ativar os seguintes registos:

ADMUX

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 26-3. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 26-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection(1)
0	0	AREF, Internal V_{REF} turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Internal 1.1V Voltage Reference with external capacitor at AREF pin
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

Tabela 1 – ADMUX e seleção de bits de referência

Na tabela (1) é possível observar os bits de seleção para a seleção da tensão de referência do ADC. Os dois bits mais significativos no ADMUX são desligados visto que vamos usar o AREF para termos uma tensão de referência visto que o ADC vai converter os valores lidos (analógicos) de tensão num valor digital.

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the ADLAR bit will affect the ADC Data Register immediately, regardless of any ongoing conversions. For a complete description of this bit, see "ADCL and ADCH – The ADC Data Register" on page 286.

Figura 4 - ADLAR

ADLAR: Na figura (4) é possível observar o bit intitulado de ADLAR. A leitura do ADC é feita pelos registos ADCL e ADCH como a conversão é feita em 8 bits, não precisaremos do ADCL (resultando apenas na ativação do ADCH) e efetuaremos uma conversão do ADC com Left Adjust Result em que resulta na representação de números em 8 bits (em que o bit mais significativo é o sinal). Resultando assim na ativação do ADLAR.

- **Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. See Table 26-4 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

Figura 5 - MUX

MUX: Na figura (5) é possível observar o bit intitulado por MUX. O MUX é responsável pela seleção do ADC que iremos utilizar, esta seleção é feita através de uma combinação binária. Se por exemplo quisermos uma leitura com o ADC0 usamos uma combinação de binário.

ADCSRA

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 6 - ADCSRA

Na figura (6) temos o Analog-to-Digital Converter Control and Status Register A (ADCSRA) é o registo usado para o controlo dos ADC's, neste registo iremos ativar bits para o controlo e inicialização do ADC.

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

Figura 7 - ADEN

ADEN: O ADEN observado na Fig. (7) é responsável pela ativação/desativação do ADC, quando é colocado em nível alto (1) o ADC está ativo. Quando é colocado em nível baixo (0) o ADC está desativado.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

Figura 8 - ADSC

ADSC: O ADSC que tem a sua descrição na figura (8) é responsável pelo começo da conversão de analógico para digital, então tem de estar em nível alto. Quando a conversão termina este fica em nível baixo, logo terá de ser colocado em nível alto novamente.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

Figura 9 - ADIE

ADIE: Com a ativação deste bit as interrupções estarão prontas a serem geradas. (Figura 9)

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

Table 26-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Tabela 2 - Bits de seleção ADPS

ADPS: Quanto maior for o fator de divisão mais impulsos de clock serão gerados, assim para obtermos mais samples a ativação dos ADPS2, ADPS1, ADPS0 é crucial. (Tabela 2)

SMCR

SMCR – Sleep Mode Control Register

The Sleep Mode Control Register contains control bits for power management.

Bit	7	6	5	4	3	2	1	0	
0x33 (0x53)	–	–	–	–	SM2	SM1	SM0	SE	SMCR
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Figura 10 - SMCR

O Sleep Mode Control Register (SMCR) é o registo que é responsável pelo controlo do modo que a função sleep irá atuar, esta função permite poupanças de consumo ao utilizá-la. Embora tenha outros modos, o modo utilizado (Idle) quando ativado deixa o nosso sistema em modo dormente. (Figura 10)

- **Bits 3, 2, 1 – SM2:0: Sleep Mode Select Bits 2, 1, and 0**

These bits select between the five available sleep modes as shown in Table 11-2.

Table 11-2. Sleep Mode Select

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

Tabela 3 - Sleep Mode

Para a ativação do sleep, foi ativado o bit menos significativo (SE) para dar enable ao sleep e o como queríamos que o programa ficasse à espera da conversão dos ADC's o sleep foi usado no modo Idle ou seja o SM0, SM1 e SM2 foram colocados a nível baixo. (Tabela 3)

Código – Controlo de Temperatura

Embora o trabalho fosse um controlo um sistema de uma estufa, em que os níveis de CO2 e humidade eram regulados por um sistema em conjunto com a regulação da temperatura.

Apenas foi implementado o código em placa o código para o controlo da temperatura, nas imagens seguintes é possível observar o código utilizado para o controlo da temperatura da estufa.

```
.include "m2560def.inc"


jmp MAIN
.org 0x3A
jmp ADC0

MAIN:

ldi r16,LOW(RAMEND)
out SPL,r16
ldi r16,HIGH(RAMEND)
out SPH,r16

ldi r18,0x00
out DDRA,r18
ldi r18,0xFF
out DDRD,r18 ; PCTRL
ldi r24,0x81 ; Estado inicial
out PORTD,r24


STARTADCTint:
    ldi r18,0x20
    sts ADMUX,r18
```

```
sleep0:
    cli
    ldi r16,1
    out SMCR,r16
    ldi r18,0xCF
    sts ADCSRA,r18
    sei
    sleep

    cbi PORTD,2
    cbi PORTD,3
    rcall Delay500
lerADC0:
    lds r19,ADCH
    ldi r18,0
    out SMCR,r18

    sbi PORTD,2
    cbi PORTD,3
    rcall Delay500


    in r0,PINA
    cp r19,r0
    brsh tempex
    rjmp sleep0


tempex:

    sbi PORTD,2
    sbi PORTD,3
    rcall Delay500


    cli
    ldi r16,1
    out SMCR,r16
    ldi r18,0x21
    sts ADMUX,r18
    ldi r18,0xCF
    sts ADCSRA,r18
    sei

LERADC1:
    sleep


    lds r17,ADCH
    ldi r16,0
    out SMCR,r16
    rcall Delay500
    cbi PORTD,2
    sbi PORTD,3
    rcall temp
    rjmp STARTADCTint
```

```
temp:
    cp r17,r19
    brlo janela
    brsh ar

janela:
cbi PORTD,7
ret

ar:
sbi PORTD,7
cbi PORTD,0
sbi PORTD,1

cbi PORTD,2
sbi PORTD,3
rcall Delay500

ret

ADC0:
    reti


 Delay500:
     push R16 ; T=2 - – Consider clock 16MHz
     ldi R16,40 ; T=1
 a01dj500:
     rcall a01dr500 ; T=40* (4+T01dr500 )- >
     dec R16 ; T=1*40
     brne a01dj500 ; T=2*40-1
     pop R16 ; T=2
     ret ; T=5
 a01dr500:
     push R0 ; T=2 –> total= 200713
     clr R0 ; T=1
 a02dj500:
     rcall a02dr500 ; T=256* (4+T02dr500)
     dec R0 ; T=1*256
     brne a02dj500 ; T=2*256-1
     pop R0 ; T=2
     ret ; T=5
 a02dr500:
     push R0 ; T=2 –> total= 777
     clr R0 ; T=1
 a03dj500:
     dec R0 ; T=1*256
     brne a03dj500 ; T=2*256-1
     pop R0 ; T=2
     ret ; T=5
```

Esquema de montagem

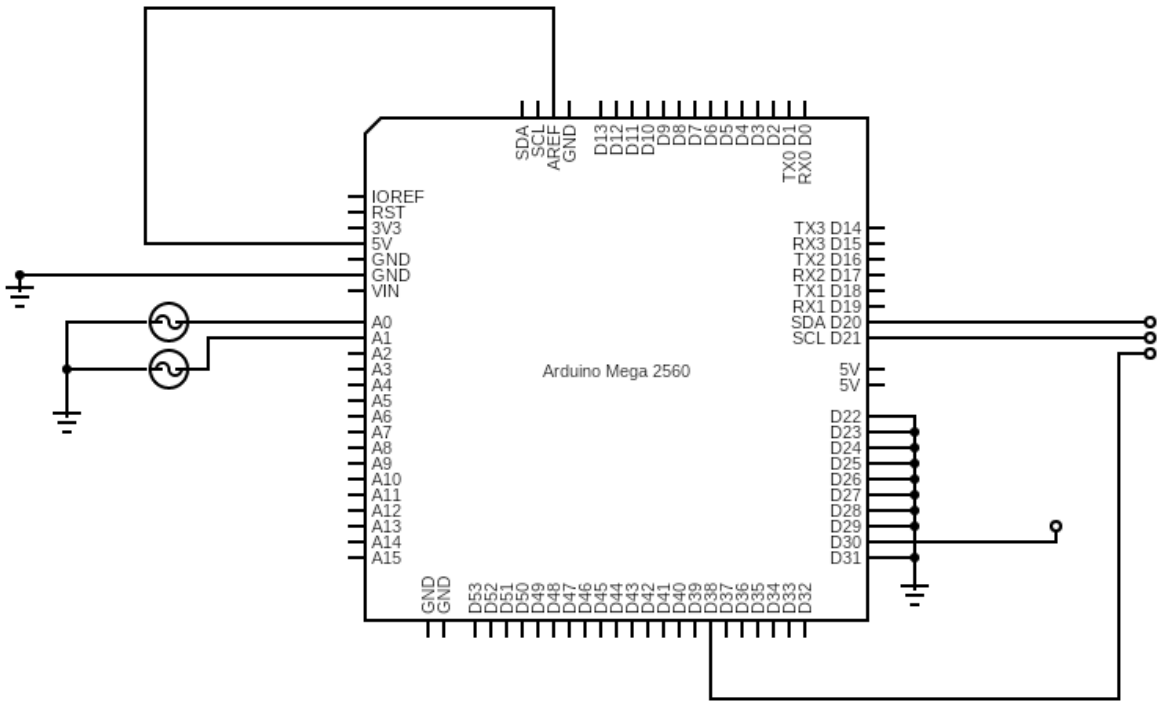

Figura 11 - Esquema de montagem

Na figura (11) é possível observar o esquema de montagem, para obter esta simulação do circuito foi usado a ferramenta do Circuit Diagram. Infelizmente este simulador não incluía nos seus componentes a breadboard então o resultado obtido foi o da imagem anterior.

Do pin 22 ao 31, excluindo o pin 30, podemos observar as ligações feitas para obtermos uma referência. Para esta referência foi usado o porto A resultando na utilização dos pins deste porto. Os pins 22-31 (exceto o 30) foram ligados a ground (nível baixo) e o pin 30 foi ligado a nível alto. Com isto obtemos uma tensão de referência na ordem dos 1.25 V.

Para o porto de controlo (PCTRL) foi utilizado o porto D, para o seu funcionamento correto os devidos bits do porto foram ligados às sondas da placa. Para que fosse possível observar o output.

Para a leitura dos ADC'S foram ligados dois, para que seja possível lermos a temperatura interior e a exterior. Foram então ligados o ADC0 e o ADC1 à fonte de tensão.

Para termos uma tensão de referência, que é a nossa referência analógica, foram ligados os 5V do arduíno ao pin AREF.

Conclusão

Com este trabalho foi possível aprofundar o meu conhecimento na programação em assembly através do AVR, aprofundar o meu conhecimento sobre o ATmega2560 e ver as dificuldades que encontramos ao tentar fazer a ponte de ligação entre programa e hardware.

Relativamente aos resultados obtidos, pode-se dizer que foram resultados bons visto que o controlo do sistema era bastante preciso. E efetuava aquilo que era esperado.