Sistemas Baseados em Microprocessadores

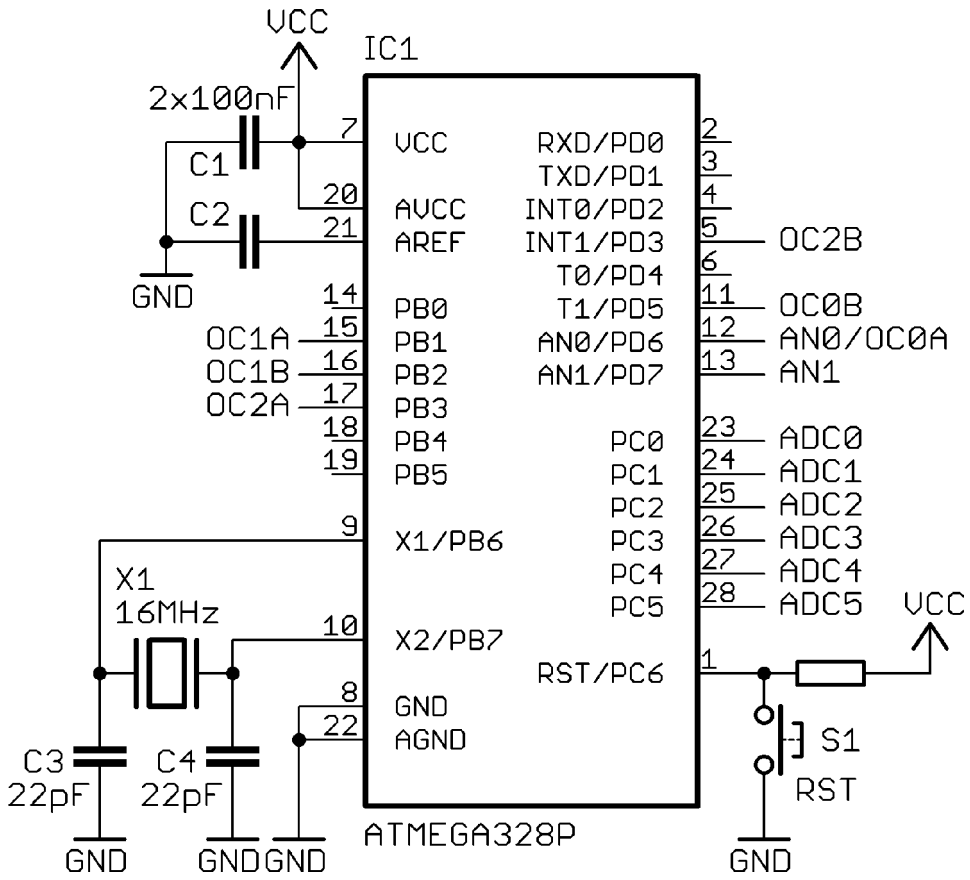
Mestrado Integrado em Engenharia Eletrotécnica e de Computadores
Faculdade de Engenharia

ATmega328P – Conversor A/D

João Paulo de Sousa

Processamento de sinais analógicos

- Entradas analógicas:
  - Conversor AD (ADCi)
  - Comparador (Ani)
- Saídas “analógicas”:
  - Saídas digitais PWM com filtro passa-baixo externo (OCiA, OCiB)



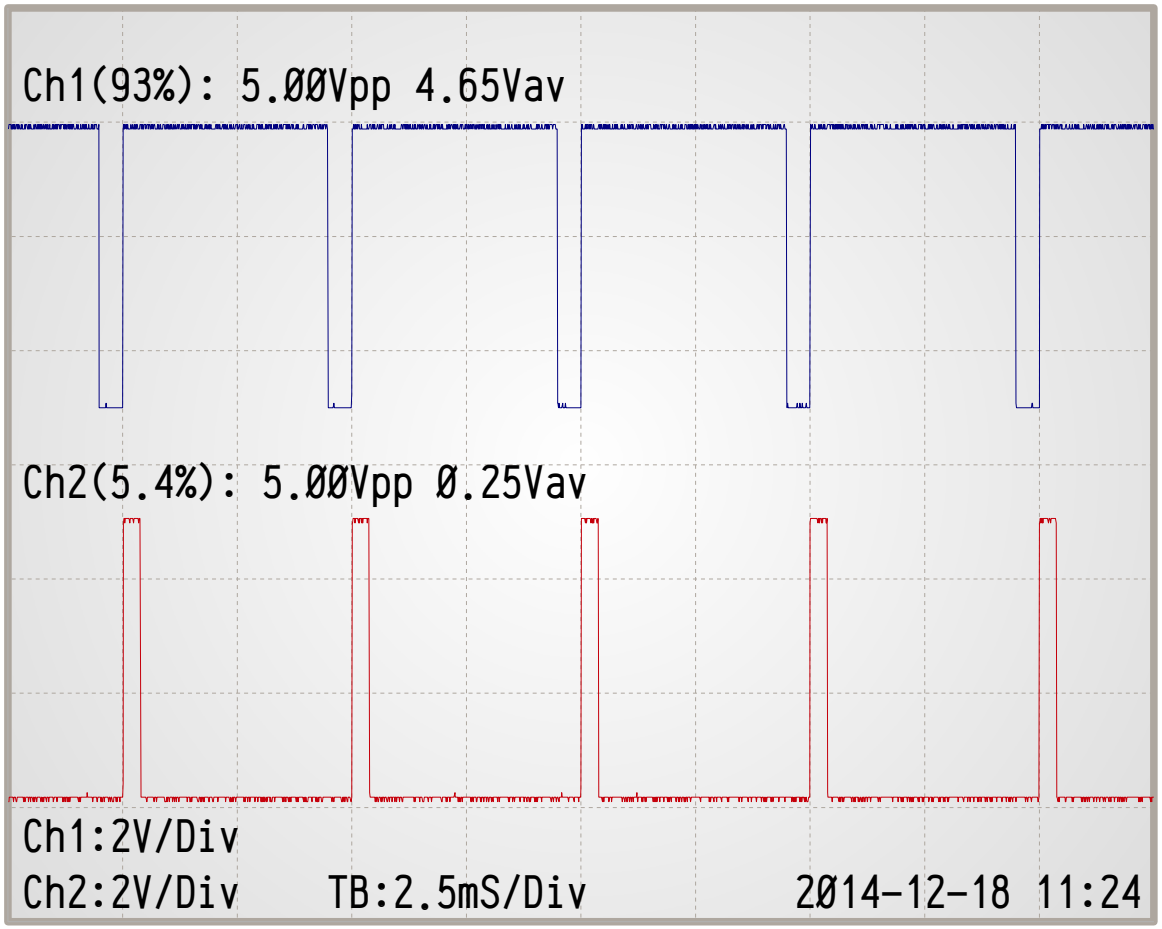
Saídas analógicas e sinais PWM

- Duty Cycle:

$$\frac{T_{ON}}{T_{ON}+T_{OFF}}$$

- Valor médio:

$$V_{AV}=\frac{T_{ON}}{T_{ON}+T_{OFF}}V_{pp}$$

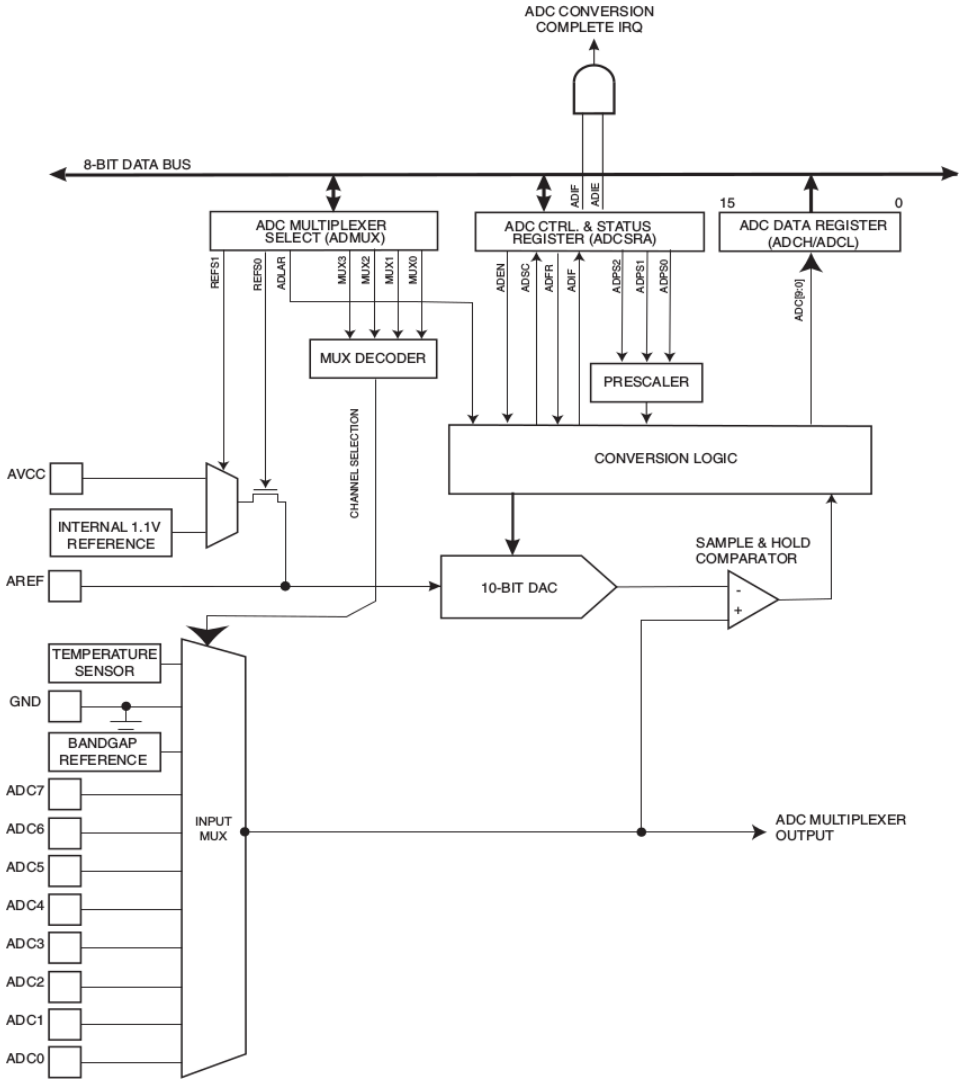


Conversor AD

- Características:
  - 6 ou 8 canais externos, 10 bits
  - Conversão: 13..260 us
  - Resultado: 8 ou 10 bits
  - Gama de entrada: 0 – Vcc
  - Conversão única ou em contínuo
  - Interrupção no fim da conversão

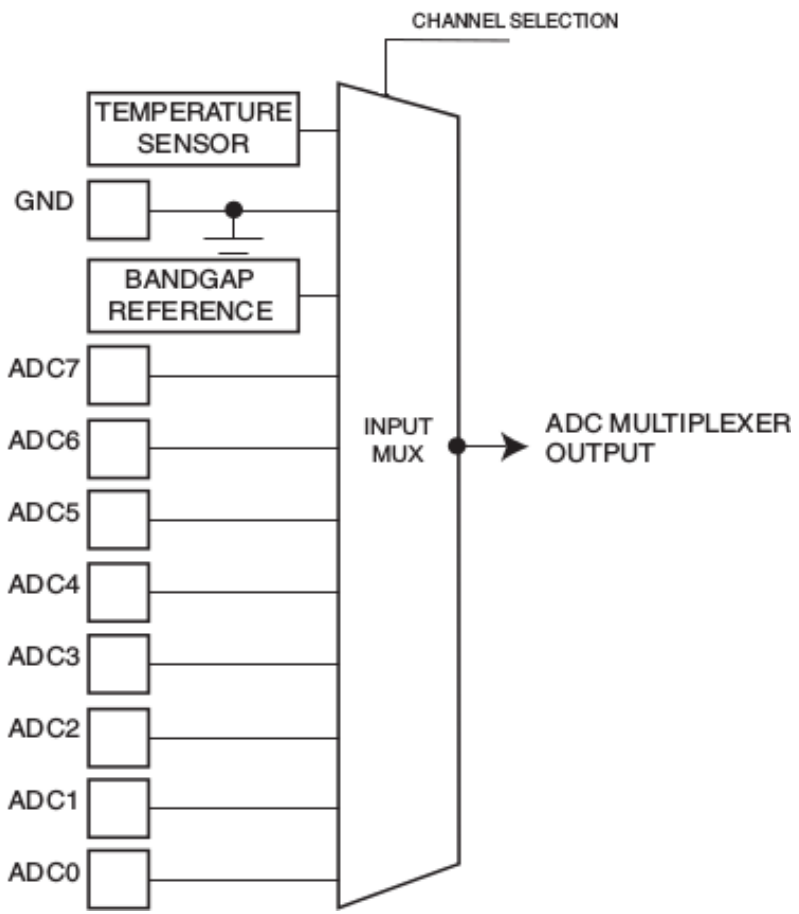
- Resultado (0..1023):

$$N = \frac{V_i \times 1024}{V_{REF}}$$

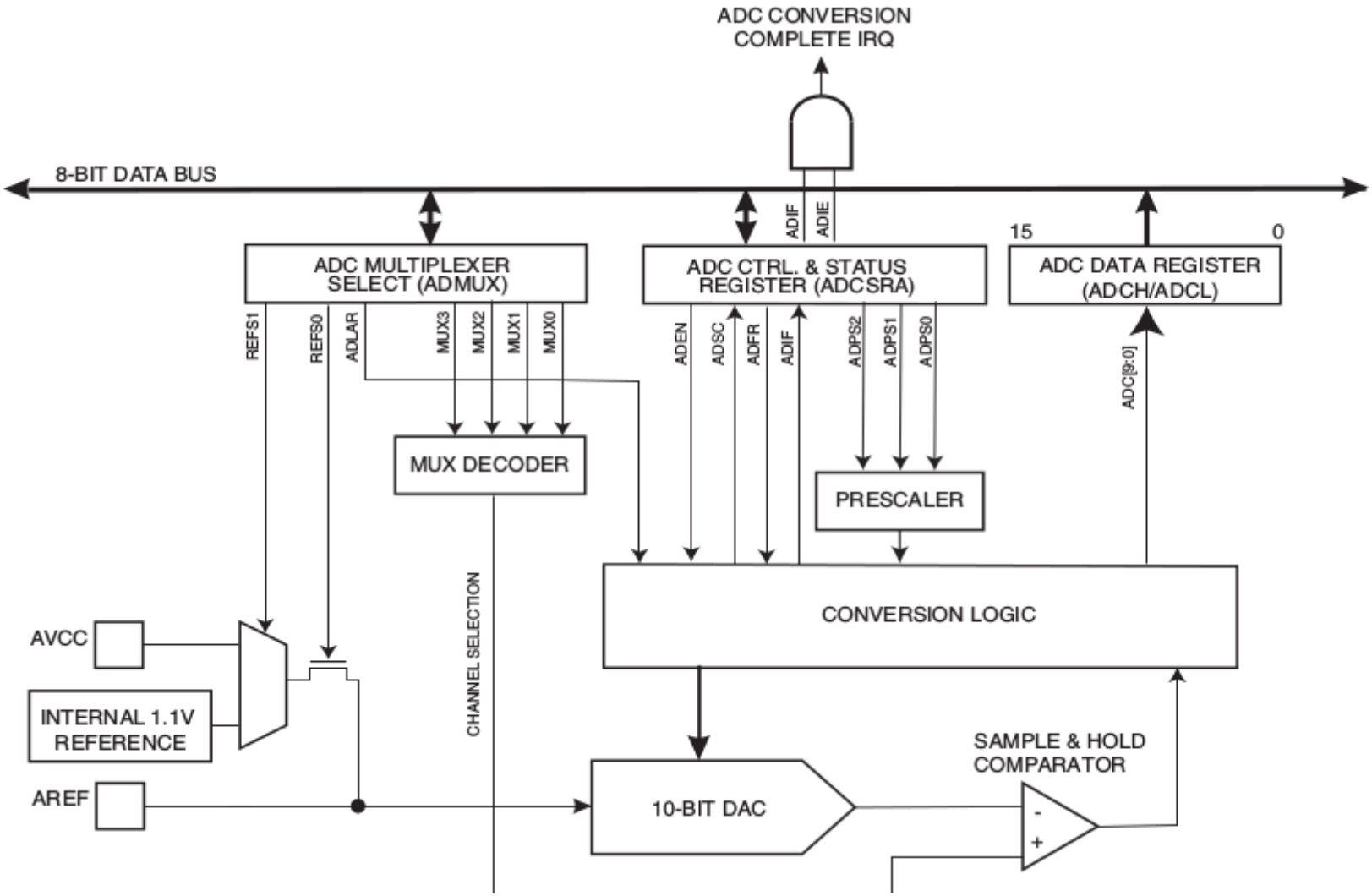


Multiplexador de entrada

- Comandado por 4 bits
- Permite escolher 1 de até 11 sinais:
  - 6 ou 8 canais
  - Sensor de temperatura
  - Tensão de referência
  - Massa



Bloco de controlo



Conversor AD – Registos associados

- ADMUX:

REFS1	REFSØ	ADLAR	–	MUX3	MUX2	MUX1	MUXØ

  - REFS1, REFS0: escolha da tensão de referência
    - (0,0) ARef; (0,1) AVcc; (1,0) Reservado; (1,1) Interna 1.1V
  - MUX3..MUX0: escolha do canal
    - 0..7 = ler canal 0..7; 8 = ler sensor de temperatura
    - 9..13 = reservados
    - 14 = ler referência de 1.1V; 15 = ler referência de 0V
  - ADLAR=1: Resultado ajustado à esquerda em ADC

Conversor AD – Registos associados

- ADCSRA:

ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPSØ

  - ADEN: ativa o conversor AD
  - ADSC: inicia uma conversão, volta a zero no fim
  - ADATE: conversões automáticas
    - 0: conversões manuais, desencadeadas por ADSC
    - 1: conversões automáticas, dependendo de ADTS2..0
  - ADIF, ADIE: Interrupt flag e interrupt enable
  - ADPS2..ADPS0: pré-divisor do conversor AD

    (0..7) = 2, 2, 4, 8, 16, 32, 64, 128

Conversor AD – Registos associados

- ADCSRB:

–	ACME	–	–	–	ADTS2	ADTS1	ADTSØ

  - ACME: Liga canal escolhido ao comparador analógico
  - ADTS2..ADTS0: configuração das conversões automáticas
    - 0 = free running; 1 = comparador; 2 = Ext int 0;
    - 3, 4 = TC0 compare A, TC0 overflow
    - 5, 6, 7 = TC1 compare B, TC1 overflow, TC1 capture event

- DIDR0:

–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADCØD

  - ADCiD desactiva a função digital do pino correspondente

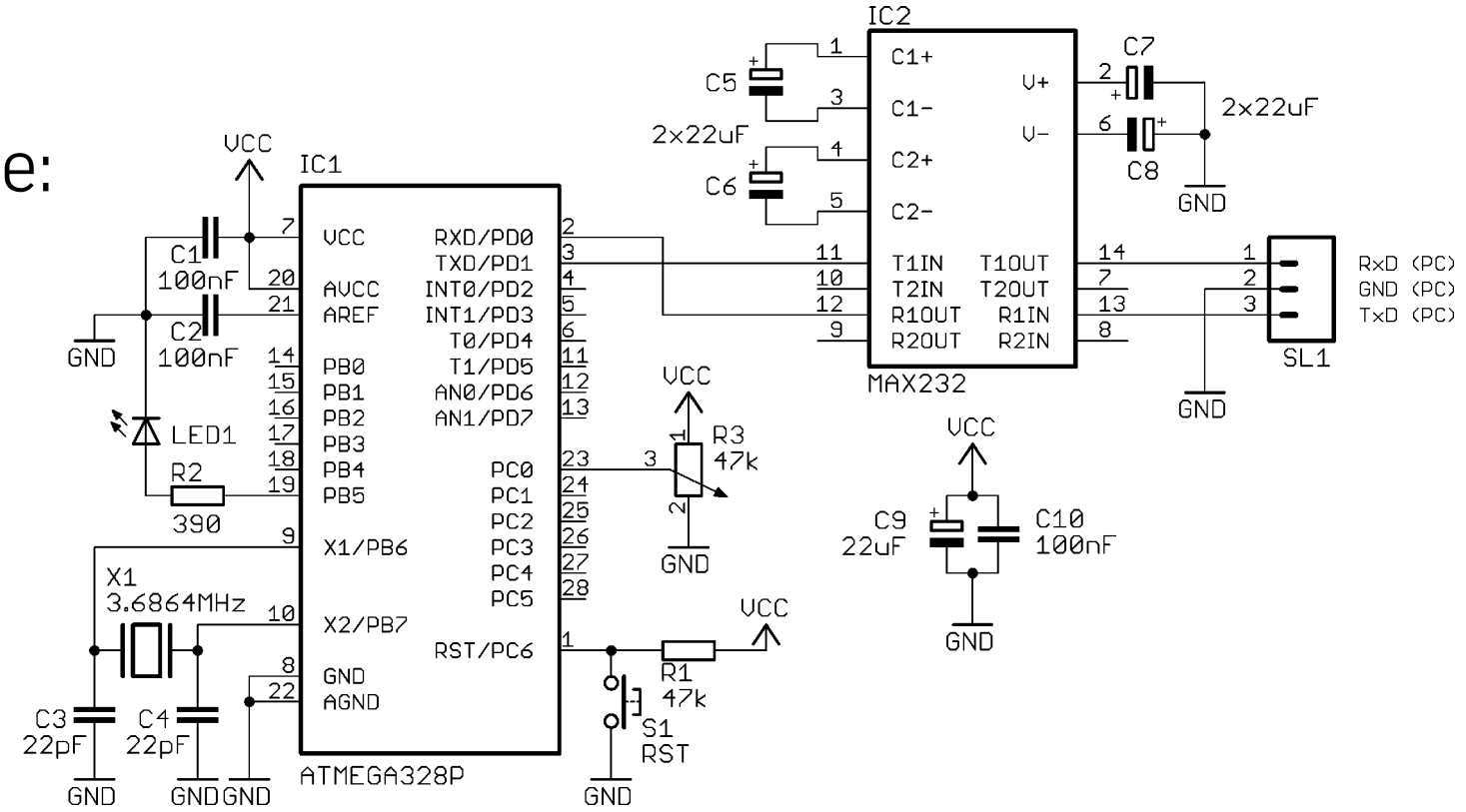
Conversor A/D – Exemplo

Enviar pela porta série, em mV, o valor da tensão no canal 0 do ADC (PC0), sempre que ele varia.

A cada envio comuta o estado de um LED.

Conversor A/D – Hardware

- Entrada: ADC0 (PC0)
- LED: PB5
- Porta série:
  - RxD
  - TxD



Conversor A/D – Software

```
#include <avr/io.h>
#include <stdio.h>

static int put_char(char c, FILE *stream);
static FILE mystdout = FDEV_SETUP_STREAM(
       put_char, NULL, _FDEV_SETUP_WRITE);

#ifndef F_CPU
#define F_CPU 16000000ul
#endif
#define BAUD 57600ul
#define UBBR_VAL ((F_CPU/(BAUD<<3))-1)

#define VREF 5
#define LED PB5

int put_char(char c, FILE *stream) {
    while((UCSR0A & (1<<UDRE0)) == 0);
    UDR0 = c;
    return 0;
}
```

```
void init_adc(void) {
    // Definir Vref=AVcc
    ADMUX = ADMUX | (1<<REFS0);
    // Desativar buffer digital em PC0
    DIDR0 = DIDR0 | (1<<PC0);
    // Pré-divisor em 128 e ativar ADC
    ADCSRA = ADCSRA | (7<<ADPS0)|(1<<ADEN);
}

void init_usart(void) {
    // Definir baudrate
    UBRR0H = (uint8_t)(UBBR_VAL>>8);
    UBRR0L = (uint8_t) UBBR_VAL;
    UCSR0A = (1<<U2X0);  // Double speed
    // Definir formato da trama
    UCSR0C = (3<<UCSZ00) // 8 data bits
           | (0<<UPM00)  // no parity
           | (0<<USBS0); // 1 stop bit
    // Ativar recetor e emissor
    UCSR0B = (1<<RXEN0) | (1<<TXEN0);
}
```

Conversor A/D – Software

```
int main (void) {
    unsigned int new,old,v = 0;

    init_usart();
    stdout = &mystdout;    //output stream

    init_adc();

    DDRB = DDRB | (1<<LED);

    while(1) {
        new=read_adc(0);
        if (new!=old) {
            old=new;
            v=(double)VREF*new*1000/1024;
            printf("%dmV\n",v);
            PORTB = PORTB ^ (1<<LED);
        }
    }
    return (0);
}
```

```
unsigned int read_adc(unsigned char chan) {
    // escolher o canal...
    ADMUX = (ADMUX & 0xF0) | (chan & 0x0F);

    // iniciar a conversão
    //   em modo manual (ADATE=0)
    ADCSRA |= (1<<ADSC);

    // esperar pelo fim da conversão
    while(ADCSRA & (1<<ADSC));

    return ADC;
}
```

Dúvidas

Para aprofundar:

- Datasheet (Jun 2016):
  - Conversor AD: cap 28, Comparador: Cap. 27
- Application Notes:
  - AVR127: Understanding ADC parameters
  - AVR120: Characterization & calibration of the ADC
  - AVR125/126: ADC in single ended mode
  - AVR121: Enhancing ADC resolution by oversampling
  - AVR128: Using the Analog comparator
  - AVR400: Low Cost ADC using the analog comparator