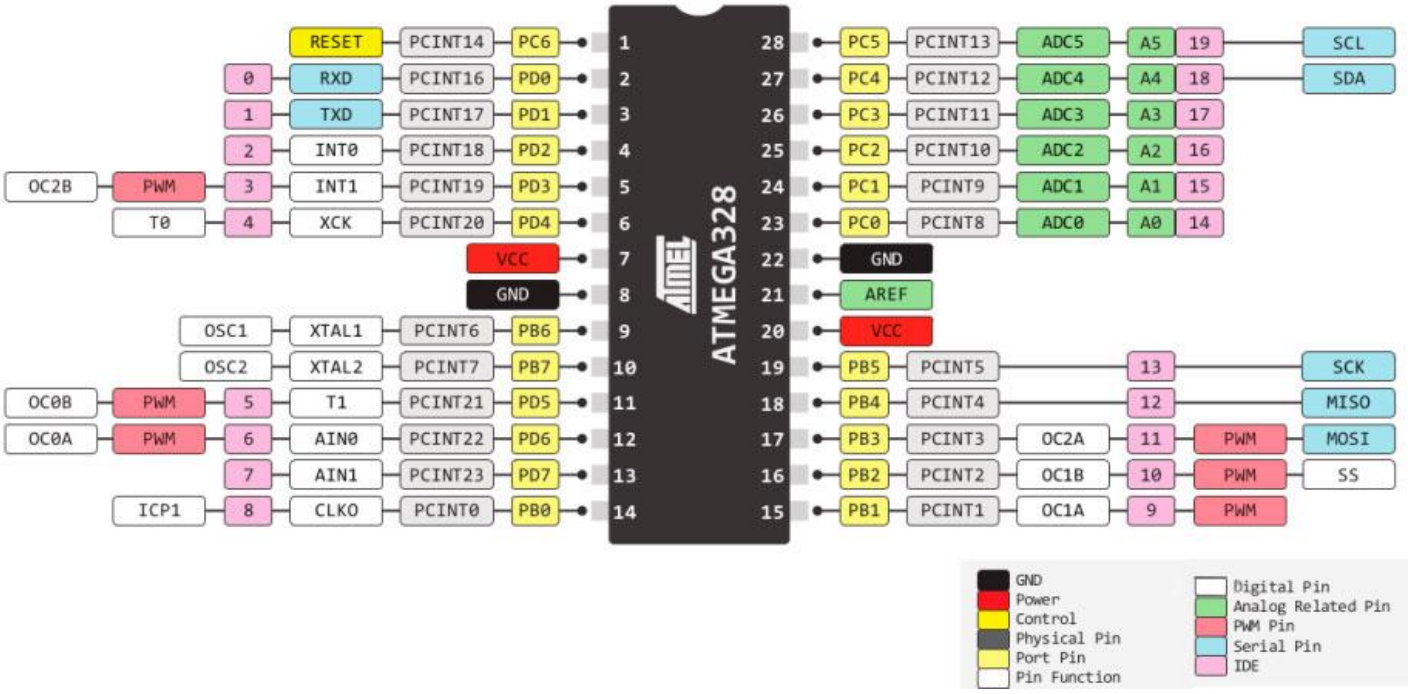
Inatel Instituto Nacional de Telecomunicações	**RELATÓRIO 4**	**Data: / /**
	Disciplina: E209	
	Prof: Yvo Marcelo Chiaradia Masselli **Monitores: João Lucas/Luan Siqueira/Maria Luiza/ Lucas Lares/Rafaela Papale**	
Conteúdo: ATmega328P		
Tema: Dispositivos de Entrada e Saída (GPIO) – Utilização com máscara		
Nome:	**Matrícula:**	**Curso:**

OBJETIVOS:

- Conceituar basicamente a arquitetura do microcontrolador Atmega328P.
- Utilizar ferramentas de simulação para desenvolver programas para o Atmega328P.
- Interpretar as funcionalidades dos registros de GPIO do Atmega328P.
- Testar o programa que faz uso da GPIO.

Parte Teórica

O microcontrolador Atmega328P



O ATmega328P apresenta seus portais identificados por **PB, PC** e **PD.** Cada pino do microcontrolador possui a função básica de GPIO **(entrada e saída)**, mas pode apresentar outras funções que serão abordadas nos próximos relatórios.

Exercícios Teóricos(com orientação do professor/monitor):

1) Procure o datasheet do microcontrolador ATmega328P e responda os itens a seguir:

a) Qual a tensão de saída de uma porta GPIO em nível lógico 0(**VOL**)? E em 1 (**VOH**)? Cite a página e o tópico do datasheet onde encontrou esta informação.

b) Qual a tensão de alimentação do ATmega?

c) Qual a corrente máxima de saída das portas GPIO do ATmega?

d) Quais os valores das tensões de **VIH3** e **VIL3**?

Uso de linguagem C para microcontroladores - Introdução

A linguagem C, inicialmente criada para desenvolvimento de programas de computador, foi aos poucos sendo substituída por outras linguagens que facilitavam o uso de componentes visuais. Porém, ganhou uma sobrevida devido ao seu uso nos sistemas embarcados, ou seja, nos microcontroladores.
A sintaxe da linguagem é a mesma, seja para PC ou para MCU, mas cabe salientar algumas observações quanto ao seu uso em embarcados:

a) Deve-se prestar atenção nos tipos de variáveis utilizadas (char, short, int, long) devido a limitação de espaço de memória de dados (RAM);

b) Normalmente, os programas de microcontroladores não possuem fim, ou seja, nunca acabam. Dessa forma, utiliza-se estruturas de repetição infinita **(loop-infinito)** no programa: **for(;;)** ou **while(1);**

c) Quando os programas são de baixa complexidade e apresentam lógica de controle simples, podem ser desenvolvidos de forma que sejam executados sequencialmente: realiza a leitura das entradas e armazena em variáveis, interpreta os valores das variáveis e executa a lógica desejada, atualiza as saídas (método conhecido como **super-loop**). Também é possível o uso de máquina de estados;

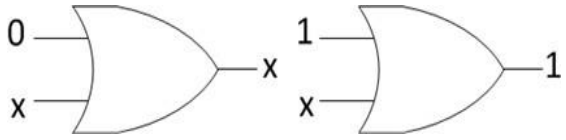
d) Deve-se utilizar recursos que facilitam a alteração do uso dos pinos de **GPIO/portais**. Normalmente isso é feito utilizando a diretiva **"#define".** Dessa forma, caso um periférico tenha que ser trocado de pino, fica simples adaptar o programa. Exemplo:

```
#define P7 0b10000000
#define P4 0b00010000
```

Técnica de mascaramento

É muito comum ser necessário trabalhar com bits. Porém, as variáveis mínimas são de 8 bits. Para manipular bits, utiliza-se a aritmética binária com a lógica **"OU"** e **"E"** da seguinte forma:

- **Lógica OU**: possível fazer com que uma informação **X** seja **"1"**. Se fizermos a lógica **OU** entre **"bit qualquer"** e **"1"**, o resultado sempre será **"1"**. Se fizermos a lógica **OU** entre **"bit qualquer"** e **"0"**, o resultado será o valor do **"bit qualquer"**.



Exemplos:

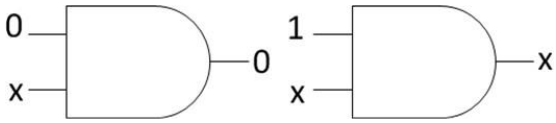
a) Escrever **"1"** no bit 0: **PORTx = PORTx | 0b00000001;**

PORTx - bits	7	6	5	4	3	2	1	0
PORTx antes	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
Máscara a ser aplicada	0	0	0	0	0	0	0	**1**
PORTx depois	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**1**

b) Escrever **"1"** no bit 6: **PORTx = PORTx | 0b01000000;**

PORTx - bits	7	6	5	4	3	2	1	0
PORTx antes	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
Máscara a ser aplicada	0	**1**	0	0	0	0	0	0
PORTx depois	**X**	**1**	**X**	**X**	**X**	**X**	**X**	**X**

- **Lógica E**: possível fazer com que uma informação **X** seja **"0"** ou mascarar(filtrar) uma informação **X** desejada para ser lida. Se fizermos a lógica **E** entre **"bit qualquer"** e **"0"**, o resultado sempre será **"0"**. Se fizermos a lógica **E** entre **"bit qualquer"** e o valor **"1"**, o resultado será o valor do **"bit qualquer"**.



Exemplos:

a) Escrever **"0"** no bit 0: **PORTx = PORTx & ~(0b00000001);**

PORTx - bits	7	6	5	4	3	2	1	0
PORTx antes	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
Máscara a ser aplicada	1	1	1	1	1	1	1	**0**
PORTx depois	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**0**

b) Escrever "0" no bit 6: **PORTx = PORTx & ~(0b01000000);**

PORTx - bits	7	6	5	4	3	2	1	0
PORTx antes	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
Máscara a ser aplicada	1	**0**	1	1	1	1	1	1
PORTx depois	**X**	**0**	**X**	**X**	**X**	**X**	**X**	**X**

c) Ler a informação contida no bit 3: **var = PINx & 0b00001000;**

PINx - bits	7	6	5	4	3	2	1	0
PINx antes	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
Máscara a ser aplicada	0	0	0	0	**1**	0	0	0
VAR	**0**	**0**	**0**	**0**	**X**	**0**	**0**	**0**

Parte Prática

2) Crie um projeto na ferramenta de simulação, digite o programa apresentado em anexo e compile-o para verificar se houve erros.

3) Execute o programa e verifique seu funcionamento: pressione o botão e veja o resultado do funcionamento do Led.

4) Faça um desenho do **diagrama em blocos** que represente o circuito que foi utilizado. Evidencie a **conexão do Led** e **do botão (Push-Button)** ao **MCU**.

5) Comente (coloque **//** antes da linha) **"DDRD = DDRD | 0b10000000;"**. Compile o programa, execute e verifique o resultado. O que ocorreu? Explique.

6) Modifique o programa de forma que o **LED2** seja acionado e não o **LED1**, quando o **botão S2 for pressionado**. Anote as modificações que foram realizadas.

7) Modifique o programa de forma que **LED1** e **LED2** sejam acionados quando o **botão S2 for pressionado**. Anote as modificações que foram realizadas. Lembre-se de configurar o registro **DDRD** para que ambos os pinos dos **LEDs** sejam configurados como saída.

8) Retire o resistor de Pull-up da montagem. Compile o programa e verifique o resultado. O que ocorreu? Explique.

9) Modifique o programa, de forma que **LED1** e **LED2** sejam acionados de **forma alternada** a cada 2 segundos, ou seja, um aceso e o outro apagado e vice-versa. Anote as modificações que foram realizadas.

ANEXO) PROGRAMA GPIO COM MÁSCARA

```c
int main(void)
{
    DDRD = DDRD | 0b10000000;       // Pino PD7 definido como saída
    PORTD = PORTD | 0b00010000;     // Habilitar resistor de PULL-UP no PD4
    PORTD = PORTD & ~(0b10000000); // Desliga a saída PD7;
    for (;;)
    {
        if ((PIND & 0b00010000) == 0) // Botão pressionado ?
        {
            PORTD = PORTD | 0b10000000;    // Aciona a saída
            _delay_ms(5000);
            PORTD = PORTD & ~(0b10000000); // Desliga a saída
        }
    }
}
```