


<div> <i>Instituto Nacional de Telecomunicações</i></div>	RELATÓRIO 3	Data:    /    /	
	Disciplina: E209		
	Prof: João Pedro Magalhães de Paula Paiva Monitores: Thalita Domingos, João Henrique Delfino, Pedro Fraga		
Conteúdo: Microcontroladores AVR			
Tema: GPIO ATmega 328P			
Nome:		Matrícula:	Curso:

### **OBJETIVOS:**

- Interpretar as funcionalidades dos registros de GPIO do ATmega328P;
- Utilizar o arduino para desenvolver programas para o ATmega328p;
- Testar o programa que faz uso da GPIO.

### *Parte Teórica*

#### **Registros:**

Os Registros são blocos internos do MCU configuráveis à aplicação desejada. Um exemplo básico é o conjunto de registros que configuram os Portais (**DDRx, PORTx, PINx, ...**). No Portal podemos configurar as direções das portas (**entrada ou saída**), **escrever nas saídas, ler as entradas, habilitar resistores internos** e **acessar funções especiais** (abordado posteriormente).

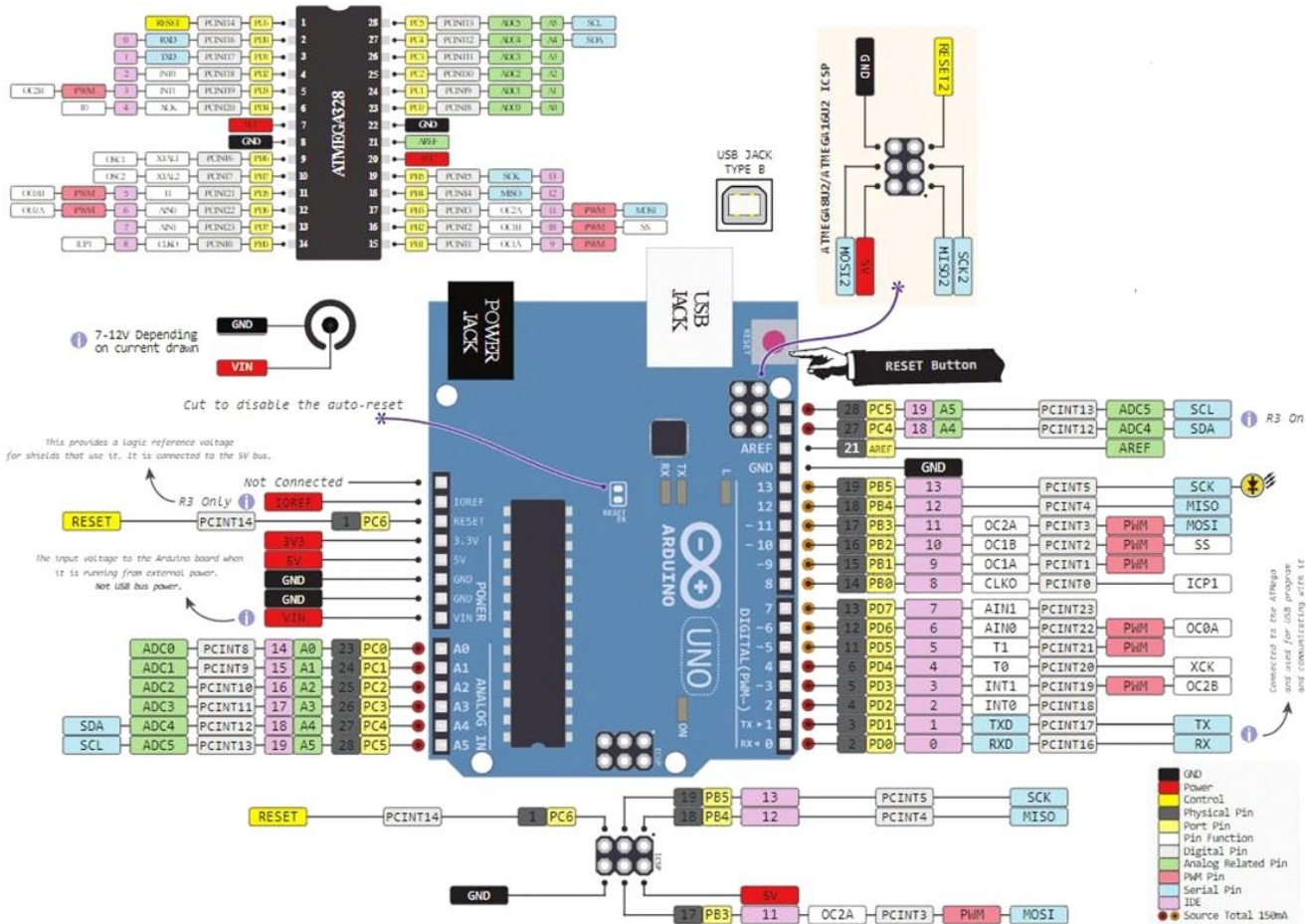
Para fazer o uso da GPIO (escrita e leitura), é necessário configurar a **direção da porta** e habilitar **resistores internos** para entradas que geram um nível lógico (alto ou baixo) quando ativas e circuito aberto quando desativadas.

Os registros têm seus nomes definidos pelo manual de utilização do ATmega328P e são incluídos com as bibliotecas do microcontrolador que estamos trabalhando.

#### **Direção da porta - DDRx:**

Para configurar uma porta como entrada ou saída, devemos manipular o registro **DDRx**. No ATmega328P temos **DDRB, DDRC** e **DDRD**, sendo o portal B e o portal D contendo 8 portas e o portal C contendo 7 portas (**PB0 a PB7, PC0 a PC6 e PD0 a PD7**). Para definir uma porta como **entrada**, é necessário **escrever 0 no bit respectivo da porta no DDRx** e para definir uma porta como **saída**, é necessário **escrever 1 no bit respectivo da porta no DDRx**.

Após o reset, todas as portas dos três portais são definidas como entradas.



### Exemplo de configuração da direção da porta:

Devemos configurar as portas PD7 e PD5 como saídas e a porta PD4 como entrada. Para realizar essa configuração, devemos escrever 1 nos bits 7 e 5 e escrever 0 no bit 4 no DDRD. Podemos acompanhar essa configuração na tabela abaixo.

BITS	7	6	5	4	3	2	1	0
DDRD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
Peso hexadecimal	8	4	2	1	8	4	2	1
Valor binário	1	0	1	0	0	0	0	0
Valor hexadecimal a ser escrito	A				0			

O valor a ser escrito no registro DDRD é **0xA0**(hexadecimal) ou **0b10100000** (binário). Essa escrita pode ser feita das seguintes maneiras:

**DDRD = 0xA0;**  
**DDRD = 0b10100000.**

### Leitura da entrada – PINx:

No registro **PINx** são armazenados os valores das portas configuradas como entradas no **DDRx**. Para lermos a informação contida em cada porta do portal de maneira mais fácil podemos utilizar uma máscara para “filtrar” apenas a porta que queremos ler. Essa técnica de mascaramento será apresentada posteriormente.

### Escrita na saída – PORTx:

No registro PORTx podemos escrever os valores que desejamos (0 ou 1) nas portas configuradas como saída no DDRx.

### Exemplo de escrita em uma porta definida como saída:

**Presumindo que as portas PD5 e PD7 já estejam definidas como saídas no DDRD,** podemos escrever 0 ou 1 nos bits 5 e 7 do registro **PORTD** e o valor escrito (nível lógico baixo ou alto) será representado por uma tensão no pino físico das portas PD5 e PD7. Suponha que em uma **primeira operação** de escrita desejamos escrever **1 na porta PD5 e 0 na porta PD7** e em uma **segunda operação** de escrita desejamos **escrever 0 na porta PD5 e 1 na porta PD7**. Podemos acompanhar essas operações nas tabelas abaixo.

### Primeira operação:

BITS	7	6	5	4	3	2	1	0
DDRD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
Peso hexadecimal	8	4	2	1	8	4	2	1
Valor binário	0	0	1	0	0	0	0	0
Valor hexadecimal a ser escrito	2				0			

O valor a ser escrito no registro **PORTD** é **0x20**(hexadecimal) ou **0b00100000** (binário). Essa escrita pode ser feita das seguintes maneiras:

**PORTD = 0x20;**  
**PORTD = 0b00100000.**

### Segunda operação:

BITS	7	6	5	4	3	2	1	0
DDRD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
Peso hexadecimal	8	4	2	1	8	4	2	1
Valor binário	1	0	0	0	0	0	0	0
Valor hexadecimal a ser escrito	8				0			

O valor a ser escrito no registro **PORTD** é **0x80**(hexadecimal). Essa escrita pode ser feita das seguintes maneiras:

**PORTD = 0x80;**

**PORTD = 0b10000000;**

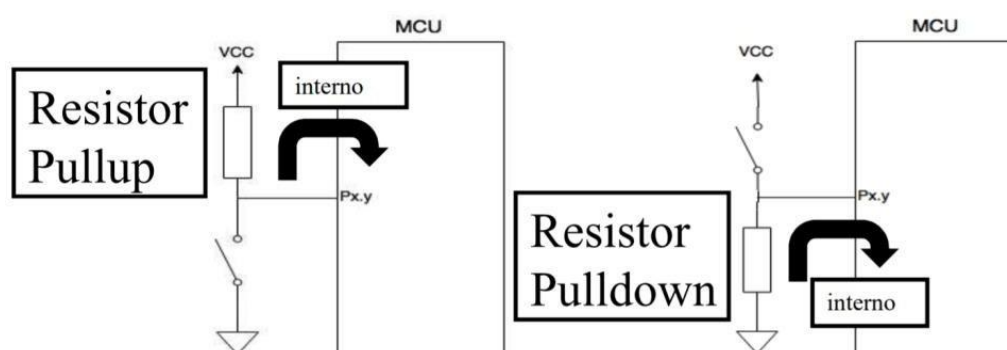
Ao fazer a escrita dessa maneira, não somente os bits que desejamos escrever são alterados, mas sim todos os bits do registro **PORTD**. Isso pode gerar grande dificuldade na manipulação das saídas. Uma possível solução é fazer o uso de uma técnica que altera somente os bits desejados. Essa técnica se chama mascaramento e será abordada posteriormente.

## Habilitar resistores internos para pull-up – PORTx:

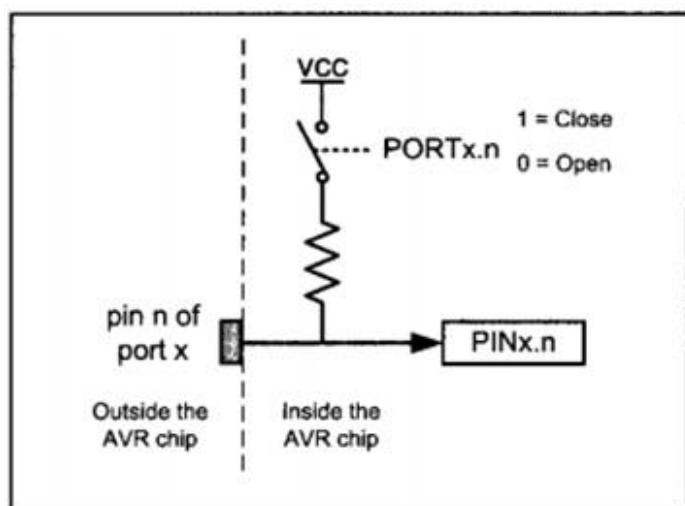
Para circuitos que geram apenas um nível lógico (alto ou baixo) quando está ativado e circuito aberto quando está desativado, faz-se necessário o uso de resistores de pull-up ou pull-down para suprir o nível lógico que falta.

Por exemplo: Um push-button quando pressionado fecha o contato entre seus dois terminais e quando não pressionado abre o contato entre seus dois terminais. Para utilizarmos esse botão como uma entrada, é necessário conectar um terminal para algum nível lógico (alto ou baixo) e o outro terminal em alguma porta configurada como entrada do MCU. Dessa maneira, com o botão pressionado, o nível lógico conectado ao outro terminal será entregue para a porta de entrada, porém, com o botão não pressionado a porta de entrada não irá receber nada, pois o circuito está aberto. Para suprir esse circuito aberto, uma possível solução é conectar um resistor entre a porta de entrada e o nível lógico oposto ao conectado no outro terminal do botão.

A imagem abaixo ilustra esse raciocínio.



O ATmega328p já possui resistores de pull-up internamente, basta habilitá-los (**PORTx**).



## Exemplo de configuração de um resistor interno como pull-up:

Presumindo que a porta PD4 já esteja definida como entrada no DDRD, podemos habilitar o resistor interno, escrevendo 1 no bit 4 do registro PORTD para ser um resistor de pull-up.

BITS	7	6	5	4	3	2	1	0
DDRD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
Peso hexadecimal	8	4	2	1	8	4	2	1
Valor binário	0	0	0	1	0	0	0	0
Valor hexadecimal a ser escrito	1				0			

O valor a ser escrito no registro **PORTD** é **0x10**(hexadecimal) ou **0b00010000** (binário). Essa escrita pode ser feita das seguintes maneiras:

```
PORTD = 0x10;  
PORTD = 0b00010000;
```

### Código exemplo:

```
void setup()  
{  
    /* configurando o pino 5(PD5) e o pino 7(PD7) do arduino como  
    pinos de saída*/  
    DDRD = 0b10100000;  
}  
  
void loop()  
{  
    //Ligando os pinos 5(PD5)e 7(PD7)  
    PORTD = 0b10100000;  
  
    //Uma pausa de 100ms no programa  
    _delay_ms(100);  
  
    //desligando os pinos 5(PD5)e 7(PD7)  
    PORTD = 0b00000000;  
  
    //Uma pausa de 100ms no programa  
    _delay_ms(100);  
}
```

**Obs:** Os registros DDRD e PORTD poderiam ter sido configurados, também, utilizando o formato decimal ou hexadecimal.

## Formato hexadecimal:

```
void setup()
{
  /* configurando o pino 5 (PD5) e o pino 7 (PD7) do arduino como
  pinos de saída*/
  DDRD = 0xA0;
}
|
void loop()
{
  //Ligando os pinos 5 (PD5) e 7 (PD7)
  PORTD = 0xA0;

  //Uma pausa de 100ms no programa
  _delay_ms(100);

  //desligando os pinos 5 (PD5) e 7 (PD7)
  PORTD = 0x00;

  //Uma pausa de 100ms no programa
  _delay_ms(100);
}
```

## Formato decimal:

```
void setup()
{
  /* configurando o pino 5 (PD5) e o pino 7 (PD7) do arduino como
  pinos de saída*/
  DDRD = 160;
}

void loop()
{
  //Ligando os pinos 5 (PD5) e 7 (PD7)
  PORTD = 160;

  //Uma pausa de 100ms no programa
  _delay_ms(100);

  //desligando os pinos 5 (PD5) e 7 (PD7)
  PORTD = 0;

  //Uma pausa de 100ms no programa
  _delay_ms(100);
}
```

**Obs:** Os três códigos acima são equivalentes.

## Parte Prática

1. Escreva as linhas de código que:
  - Configure **PD1, PD2 e PD3 como saída** e **PD4 como entrada**:
  - Escreva nível lógico alto nas portas PB1, PB2 e PB3:
2. Desenhe uma **máquina de estados** e faça seu respectivo **código em C** capaz de representar a sequência binária **00 -> 01 -> 10 -> 11 -> 00 -> ...**.  
Obs: O intervalo entre cada valor da sequência deve ser de 500 ms.
3. Desenhe uma **máquina de estados** e faça seu respectivo **código em C** capaz de representar a seguinte sequência binária com transição a partir do acionamento de um botão.  
Sequência - **01 -> 10 -> 00 -> 01 ...**.