Workshop "Introdução ao Arduino"

NEEEC-FEUP

Índice:

1-Básicos de C:

Variáveis:

```
Int Var1;

Var1 =  3;

float Var2 = 2.5;

bool Var3 = 0;
```

As variáveis são utilizadas em C para guardar valores.

Int- números interiodos

Float- números decimais

Bool- 0 ou 1, False ou True, LOW ou HIGH

Também há variáveis que permitem o armazenamento de carateres, mas não serão necessárias neste workshop.

Operadores aritméticos:

```
Int Var1=2;

Int Var2=4;

Int Var3=Var1 + Var3;

float Var4=(float)Var1/(float)Var2;


Var2+=2;

Var1-=2;

Var1++;

Var1--;
```

Os operadores aritméticos disponíveis para realizar operações entre as variáveis são +, -, *, / e % (resto da divisão inteira)

//O valor de Var 3 vai ser 6 pela soma das outras duas variáveis

//Apenas é possível realizar operações com variáveis do mesmo tipo sendo preciso alterar o tipo de variável com (type) var antes escrever a operação.

//Var2+=2 é igual a ter Var2=Var2+2;

//Var1-=2 é igual a ter Var1=Var1-2;

//Var1++ é igual a ter Var1=Var1+1;

//Var1-- é igual a ter Var1=Var1-1;

Operações logicas e condições:

(operações logicas básicas &&(and), ||(OR) , !(negação), ==(igualdade), > (maior) e <(menor).

```
Bool V1 = true;

Bool V2 !=V1;

If(V1==1){
        V2!=V2
}

If((V1==1) || (V2==1)){
        V1=false;
} else {
        V2!=V2;
}
```

O operador != coloca na variável V2 o valor contrário de V1, ou seja false.

O código escrito dentro de {} apenas é executado se a operação lógica dentro dos () for verdadeira.

Caso esta não seja verdadeira e haja um else ou else if() o programa vai passar para o código em else{} ou do else if(){} caso a condição deste se verifique verdadeira.

Loops:

```
int Var = 1;
while(Var ‹ 4){
    Var++;
}
for(int i = 0; i ‹ 10; i++){
Var++;
    if(Var › 5){
        break;
    }
}
```

Um loop while repete os comandos dentro de {} enquanto a condição entre () for verdadeira.

Os loops for são como os loops while mas com a inicialização da variável, a incrementação e a comparação é feita pelo próprio loop. A variável i inicializada aqui não pode ser utilizada fora do loop.

A instrução break, faz cancelar o loop, e que o programa saia do loop sem ter concluído as iterações.

Como na programação para arduino queremos que as instruções se repitam constantemente usamos a instrução while(1){} para ter um loop infinito.

Criar loops dentro do while de arduino pode resultar em situações de loops infinitos dentro de loop, que não é muito desejável.

Funções:

```
int SomeFunction(int number){
return number;
}
int Sum(int number1, int number2); ){
return number1+number2;
}
int main(){
int Var1 = SomeFunction(1);
int Var2 = 4;
int Var3 = Sum(Var1, Var2);
}
```

Uma função deve ser inicializada antes de ser chamada. O tipo especificado antes da inicialização da função é o tipo de variável que esta vai retornar. Os argumentos que a função pode usar são declarados dentro dos (). Após a função ser declarada e escrita esta pode ser chamada em qualquer parte do programa.

2- Como expandir o número de ligações?

Placa de montagem (breadboard)

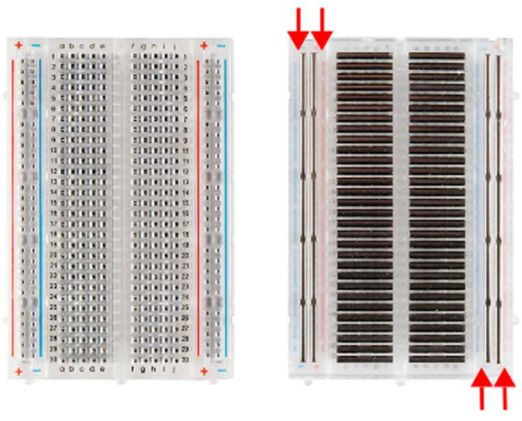

Figura 1- Esquema de uma breadboard

3-O que é um arduino?

Arduino é um projeto de hardware e software open source de uma empresa, que dimensiona e cria controladores e microcontroladores embutidos em placas para construir dispositivos digitais e interagir com objetos que controlam e recolhem informação do mundo físico.

Existem vários tipos de arduinos de entre eles o mega, nano, leonardo, uno entre outos modelos e variações. A principal diferença entre eles vem do tamanho e numero de pinos, módulos adjacentes como wi-fi ou Bluetooth, tensões que podem fornecer e que podem fornecer aos componentes, velocidade de processamento e diferentes modelos do microprocessador típico desta marca, o ATmega.

4-Funções básicas de arduino:

1. void setup(){} //Função de arduino usada para inicializações
2. void loop(){} //Função continua em loop
3. pinMode(PIN, OUTPUT/INPUT); //Define os pinos como entradas ou saídas
4. digitalWrite(PIN, HIGH/LOW); //Coloca a saida como ligada (5V) ou desligada(0V)
5. analogWrite(PIN, Valor); //Coloca um valor no intervalo [0,255] (PWM)
6. digitalRead(PIN); //Lê valores HIGH ou LOW. Tudo diferente de 0 é HIGH
7. analogRead(PIN); //Lê valores entre [0, 1023]
8. delay(ms); //introduz um atraso em milissegundos
9. millis(); //devolve à quanto tempo o programa esta a correr em ms

As funções 5 e 7 só podem ser usadas de maneira correta e eficaz caso o pino onde atuam possua um conversor analógico digital. Estes pinos permitem variar a sua saída ou reconhecer valores na sua entrada entre valores diferentes de 0 e 1 podendo então variar entre [0,255] para saídas e [0, 1023] para entradas.

Código de exemplo:

```
#define 13 LED                    //atribui ao pino 13* o nome LED
void setup() {
pinMode(LED , OUTPUT);            //configura o pino 13 como uma saída
}
void loop() {
digitalWrite(LED , HIGH);         //activa o pino 13
delay(250);                       //atraso de 250 milissegundos
digitalWrite(LED , LOW);          //desliga o pino 13
delay(250);
}
```

*O pino 13 tem associado a ele um led que esta embutido na placa arduino.

Carregar para a placa

Compilar e verificar o código

5-LEDS:

Os LEDs (light emitting diode) são dispositivos que permitem a passagem de corrente elétrica num sentido com pequena resistência bloqueando a passagem de corrente no sentido contrário com grande resistência. Os leds distinguem se dos díodos comuns pela sua capacidade de emitir luz quando atravessados pela corrente elétrica. O máximo de corrente exigido por um led é cerca de 20mA. Desta maneira é preciso colocar uma resistência em série com o led de maneira a que não passe demasiada corrente neste, danificando-o. A mínima resistência que permite que o led funcione é de 150 Ohm. Para valores abaixo o led poderá funcionar até um ponto com o risco de estragar os componentes. Neste exemplo vamos usar resistências de 1k Ohm.

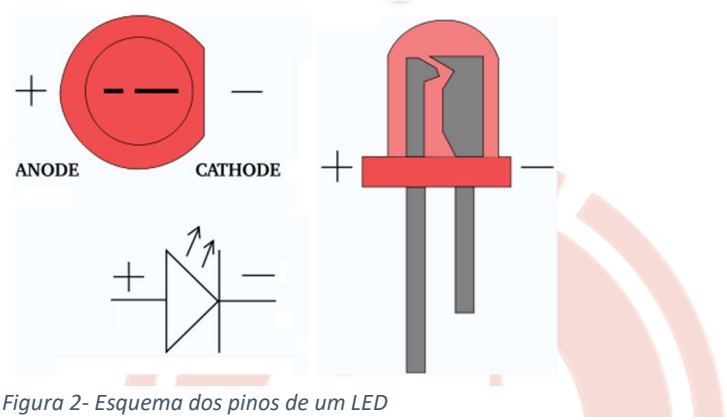

Figura 2- Esquema dos pinos de um LED

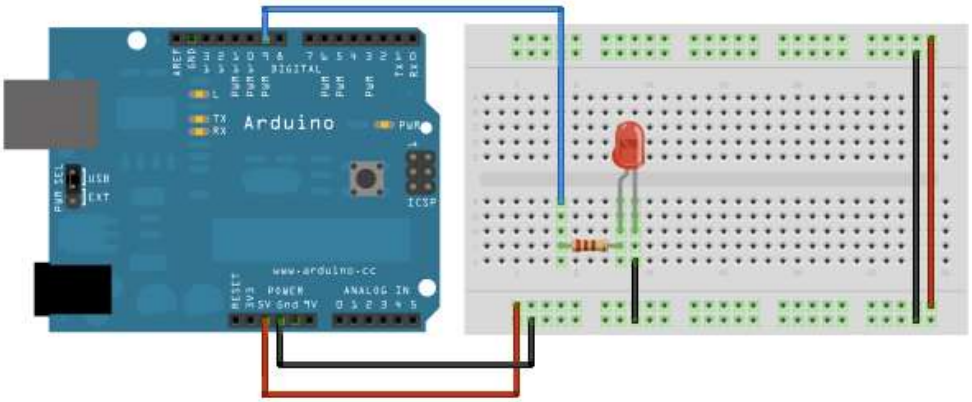

Figura 3- Modelo de montagem de um LED num arduino

Exercício 1: Montar o Led e fazer um programa que o faça estar 2 segundos ligado e 1 segundo desligado (usar o pino 11).

NEEEC-FEUP

6-Botões:

Os botões são a maneira mais útil de fornecer inputs ao circuito.

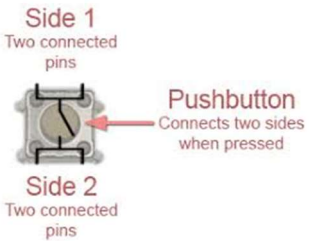

Figura 4- Esquema de um botão

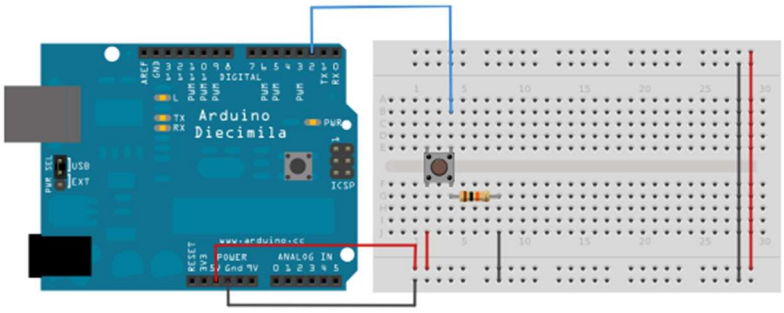

Figura 5- Modelo de montagem de um botão

Quando o botão esta premido: digitalRead(PIN)=HIGH

Quando não esta premido: digitalRead(PIN)=LOW

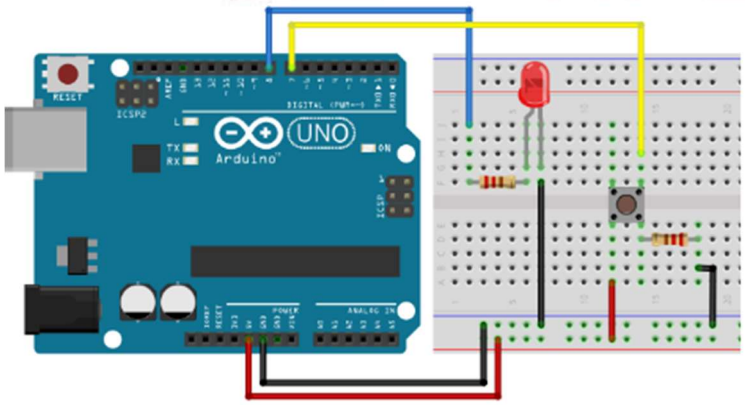

Figura 6- Modelo de montagem de um botão + Led

<u>Exercício 2</u>: Controlar o Led com usando o botão. Botão premido Led aceso, botão solto Led apagado (ligar o botão ao pino 6).

Extra: Juntar o controlo do Led com o botão com o piscar do pino incluído no arduino.

7-Temporizações:

A função delay(ms) permite separar dois comandos por um determinado tempo, mas durante esse tempo o código fica bloqueado e não avança para as instruções seguintes.

No entanto há uma maneira simples de fazer temporizações usando a função millis().

Esta função retorna o tempo em milissegundos a que o programa está a correr.

```
#define 13 LED
unsigned long int t1;          //esta variavel é definida como long para poder guardar números maiores
bool state;                    //variavel para guardar o estado do led
void setup(){
state=LOW;                     //estado inicial do led, desligado
pinMode(LED, OUTPUT);
t1=millis();                   //guardar o tempo em que se inicia o timer
}

Void loop(){

If(millis()-t1>=1000){
        state=!state;          //trocar o esta estado do led
        t1=millis();           //reinicializar o timer
}
digitalWrite(LED, state);      //ativar a saida com o valor guardado em state
}
```

Se a subtração do tempo decorrido do programa com o tempo guardado na inicialização do timer for maior que 1 segundo (1000 ms), ou seja, passou um segundo, a condição é verdadeira

8-Comunicação Série (UART):

A comunicação serie do arduino é utilizada para na comunicação entre o arduino e outros dispositivos, por exemplo modulo *bluethooth,* outros arduinos, computadores etc…

Os pinos 0(Rx) e 1(Tx) estão configurados próprias para realizar esse tipo de comunicação com outros dispositivos. Na comunicação com o PC é utilizado o cabo de alimentação e de da transferência do programa não sendo necessário a utilização dos pinos específicos para essa função. No entanto, não é possível o carregamento de um programa novo enquanto está aberta uma comunicação série devendo primeiro terminar a comunicação e até mesmo desligar dispositivos que estejam ligados aos pinos Rx e Tx e só depois carregar o programa.

Funções da porta série:

Serial.begin(9600);	Inicia a comunicação em série com uma velocidade de 9600 bits por segundo. A velocidade pode variar, tendo esta de ser coerente com a velocidade que será definida no terminal série.

Serial.print(*); //Escreve no terminal série o argumento

Ex:

Serial.print (123); // Envia "123"

Serial.print (1.234567); // Envia "1.23"

Serial.print ('N'); // Envia "N".

Serial.print ("Hello world"); // Envia "Hello world".

Serial.print("state") ; //Envia o valor da variável "state"

A função Serial.println(*) é igual a função anterior, no entanto após escrever o argumento esta irá mudar de linha, em vez de imprimir tudo seguido.

Serial.write(*); //Escreve um byte na porta serie

Utilizado normalmente para imprimir o valor de variáveis, strings entre outros.

Serial.available();

Retorna a quantidade de bytes disponíveis para leitura. Apenas retorna o número não os lê nem guarda.

Serial.read();

Lê o primeiro byte disponível no buffer da porta série.

Figura 7- Ícone do terminal série

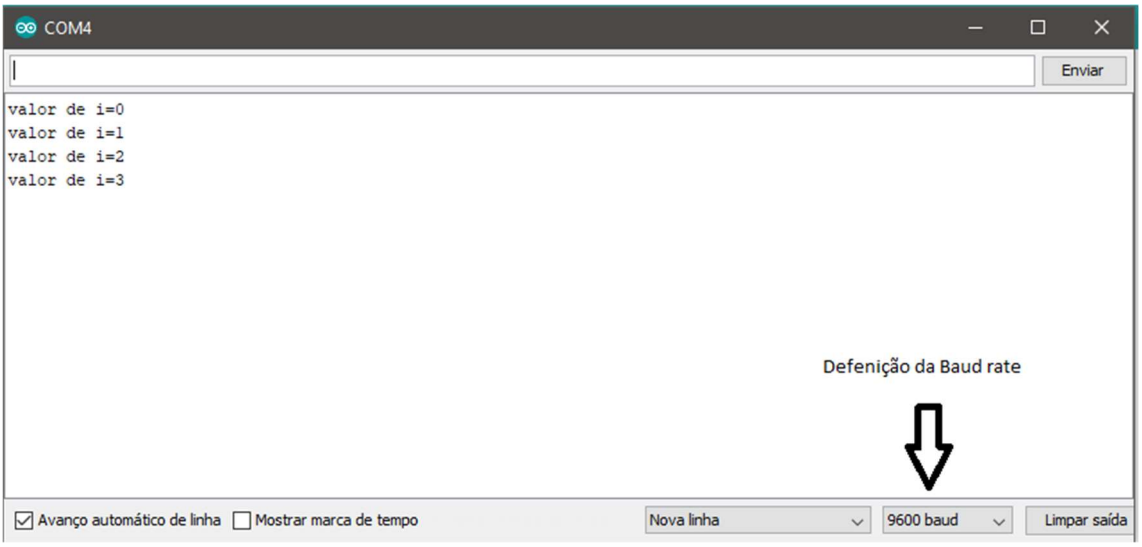

Figura 8- Exemplo terminal série

9-Sensor de distância (HC-SR04):

Este sensor funciona com o princípio de um sonar. Gera um som de alta frequência e calcula o intervalo entre o instante em que mandou o sinal e quando recebe o seu eco. Com este tempo e a velocidade do som (340m/s) é possível calcular a distância a que um objeto se encontra.

Figura 9- Sensor ultrassónico

```
void setup() {
pinMode(2,OUTPUT);                    //Pino 2 como output (trigger)
pinMode(3,INPUT);                     //Pino 3 como input (echo)
}
Void loop(){
digitalWrite(2, LOW);
delayMicroseconds(2);
digitalWrite(2, HIGH);
delayMicroseconds(10);
digitalWrite(2, LOW);
Distance = 0.017*pulseIn(3, HIGH);
delay(200);
}
```

//clear trigger
//esperar uns microssegundos para assentar
//ativar o trigger
//esperar 10 microsegundos
//clear no trigger
//calcular a distancia quando for detetado um pulso no echo
//esperar algum tempo ate voltar a medir a distância

Exercício 3: Sensor de estacionamento.

Fazer um programa que funciona como um sensor de estacionamento simplificado. Quando um objeto estiver próximo do sensor, menos de 5 cm por exemplo, o led deve acender. Adicionar também uma funcionalidade de imprimir a distância a que o objeto esta na porta série.

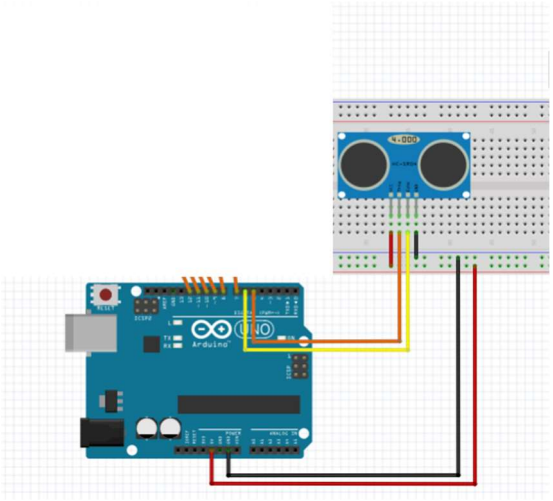

Figura 10- Montagem do sensor