Básicos de C e placa de montagem

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-Onde montar?:

Placa de montagem (breadboard)

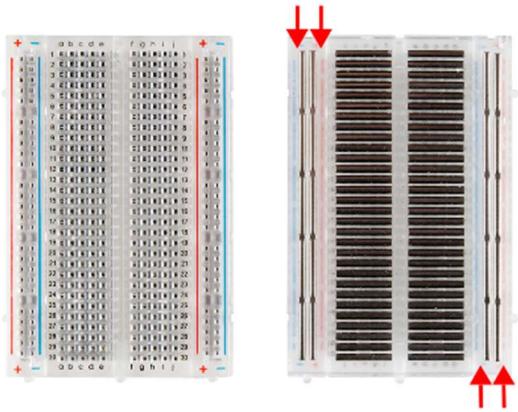
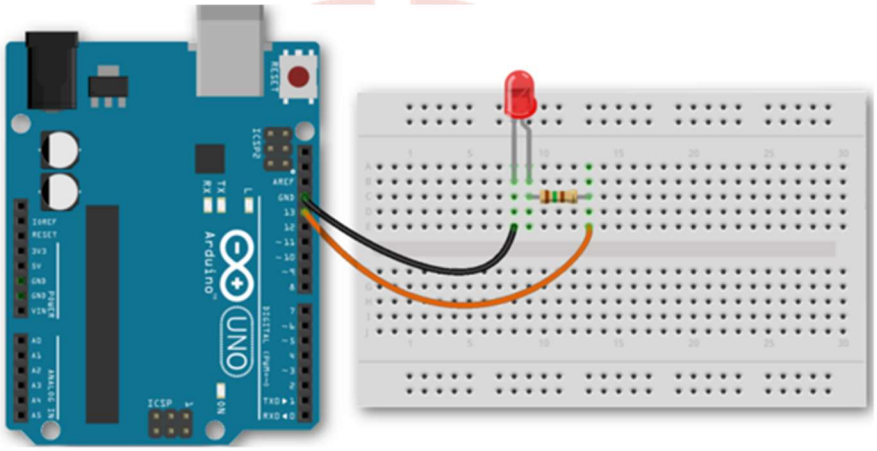


Figura 1- Esquema de uma breadboard

Ligações para um LED externo:



Modelo de montagem de um LED num arduino

Ligações para um botão: