

CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UNIEVANGÉLICA

CIRCUITOS DIGITAIS

Acadêmica: Cíntia da Silva Galvão

Relatório Técnico Sobre o Funcionamento do Código Em Arduíno: Portas Lógicas

Anápolis – GO

2019

```

1  int verde = 11;
2  int azul = 12;
3  int vermelho = 13;
4
5  void setup () {
6      pinMode (verde, OUTPUT);
7      pinMode (azul, OUTPUT);
8      pinMode (vermelho, OUTPUT);
9  }
10
11 void loop() {
12     bool valor1[4]={false, false, true, true};
13     bool valor2[4]={false, true, false, true};
14
15     for (int i=0; i<4; i++) {
16         digitalWrite(azul, valor1[i]);
17         digitalWrite(vermelho, valor2[i]);
18         digitalWrite(verde, andPorta(valor1[i], valor2[i]));
19         delay(1500);
20     }
21
22     for (int i=0; i<4; i++) {
23         digitalWrite(azul, valor1[i]);
24         digitalWrite(vermelho, valor2[i]);
25         digitalWrite(verde, or_p(valor1[i], valor2[i]));
26         delay(1500);
27     }
28
29     for (int i=0; i<4; i++) {
30         digitalWrite(azul, valor1[i]);
31         digitalWrite(vermelho, valor2[i]);
32         digitalWrite(verde, nor_p(valor1[i], valor2[i]));
33         delay(1500);
34     }
35
36     for (int i=0; i<4; i++) {
37         digitalWrite(azul, valor1[i]);
38         digitalWrite(vermelho, valor2[i]);
39         digitalWrite(verde, xor_p(valor1[i], valor2[i]));
40         delay(1500);
41     }
42
43     for (int i=0; i<4; i++) {
44         digitalWrite(azul, valor1[i]);
45         digitalWrite(vermelho, valor2[i]);
46         digitalWrite(verde, xnor_p(valor1[i], valor2[i]));
47         delay(1500);
48     }
49 }
50
51 bool andPorta(bool x, bool y) {
52     if (x && y){
53         return HIGH;
54     }
55     else{
56         return LOW;
57     }
58 }
59
60 bool or_p(bool x, bool y) {
61     if (x || y){
62         return HIGH;
63     }
64     else{
65         return LOW;
66     }
67 }
68
69 bool nand_p(bool x, bool y) {
70     if (!(x && y)){
71         return HIGH;
72     }
73     else{
74         return LOW;
75     }
76 }
77
78 bool nor_p(bool x, bool y) {
79     if (!(x || y)){
80         return HIGH;
81     }
82     else{
83         return LOW;
84     }
85 }
86
87 bool xor_p(bool x, bool y) {
88     if (!x && y || x && !y){
89         return HIGH;
90     }
91     else{
92         return LOW;
93     }
94 }
95
96 bool xnor_p(bool x, bool y) {
97     if (!(!x && y || x && !y)){
98         return HIGH;
99     }
100    else{
101        return LOW;
102    }
103 }

```

```
int verde = 11;
```

Está atribuindo o valor e o tipo da variável.

```
int azul = 12;
```

```
int vermelho = 13;
```

```
void setup () {
```

Essa parte do código está atribuindo a “função” de cada variável

```
pinMode (verde, OUTPUT);
```

ex: a variável “verde” será saída

```
pinMode (azul, OUTPUT);
```

```
pinMode (vermelho, OUTPUT);
```

```
}
```

```
void loop() {
```

```
bool valor1[4]={false, false, true, true};
```

Aqui mostra que terá um loop baseado nestes valores. Os valores

```
bool valor2[4]={false, true, false, true};
```

representam a tabela verdade.

digitalWrite(azul, valor1[i]) – Está escrevendo na porta azul, ou seja, número 12 o valor contido na posição i do vetor valor1. digitalWrite(vermelho, valor2[i]) - Está escrevendo na porta vermelho, ou seja, número 13 o valor contido na posição i do vetor valor2.

```
for (int i=0; i<4; i++) {
```

```
digitalWrite(azul, valor1[i]);
```

```
digitalWrite(vermelho, valor2[i]);
```

```
digitalWrite(verde, andPorta(valor1[i], valor2[i]));
```

```
delay(1500);
```

```
}
```

bool and_porta(bool x, bool y) Recebe o valor contido no vetor1[i] na variável X e recebe o valor contido no vetor2[i] na variável Y. **return x && y** Retorna o resultado da combinação entre dois valores booleanos correspondentes a AND

Recebe o valor contido no vetor1[i] na variável X e recebe o valor contido no vetor2[i] Retorna o resultado da combinação entre dois valores booleanos correspondente a OR.

```
bool andPorta(bool x, bool y) {
```

```
if (x && y){
```

```
return HIGH;
```

```
}
```

```
else{  
    return LOW;  
}  
}
```

```
bool nand_p(bool x, bool y) {  
    if (!(x && y)){  
        return HIGH;  
    }  
    else{  
        return LOW;  
    }  
}
```

bool nand_port(bool x, bool y) – Recebe o valor contido no vetor1[i] na variável Apresenta o resultado da combinação entre dois valores booleanos de acordo com a porta lógica NAND.

Recebe o valor contido no vetor1[i] na variável X e Apresenta o resultado da combinação entre dois valores booleanos de acordo com a porta lógica NOR.

```
bool nor_p(bool x, bool y) {  
    if (!(x || y)){  
        return HIGH;  
    }  
    else{  
        return LOW;  
    }  
}
```

Recebe o valor contido no vetor1[i] na variável X e recebe o valor contido no vetor2[i] na variável Y. apresenta !x && Mostra o resultado da combinação entre dois valores booleanos de acordo com a porta lógica XOR.

```
bool xor_p(bool x, bool y) {  
    if (!x && y || x && !y){
```

```
        return HIGH;
    }
    else{
        return LOW;
    }
}
```

Recebe o valor contido no vetor1[i] na variável X e recebe o valor contido no vetor2[i] na variável Y. !(!x && apresenta o resultado da combinação entre dois valores booleanos de acordo com a porta lógica XNOR.

```
bool xnor_p(bool x, bool y) {
    if (!(!x && y || x && !y)){
        return HIGH;
    }
    else{
        return LOW;
    }
}
```