FLIP-FLOP LÓGICO EM FUNÇÃO

Considere a necessidade de complementar o estado de uma variável a cada vez que uma função do usuário é executada. Por exemplo, criar uma função que complementa o estado de saída de uma porta (pino) a cada vez que é executada. Se existir um LED ligado nessa saída, a cada vez que a função é executada o LED muda de estado (de ligado para desligado e vice-versa).

O LED mais conhecido do Arduino é o **`LED_BUILTIN`**, que fica permanentemente ligado ao pino 13. Nas versões mais novas da IDE Arduino, a cada vez que um novo sketch é criado temos o código de exemplo:

```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Para os utilizadores do Tinkercad, a cada novo circuito com Arduino o código textual apresentado é:

```
// C++ code
//

void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000); // Wait for 1000 millisecond(s)
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000); // Wait for 1000 millisecond(s)
}
```

Esse código é válido para o Arduino padrão, e fará o LED da placa do Arduino piscar (acender e apagar) a cada dois segundos. Pode-se entretanto criar uma variável booleana (ex: **`bool w = 1;`**) e utilizar essa variável como valor a ser escrito no pino 13 (**`LED_BUILTIN`**). Com isso, o código do **`loop()`** seria alterado como se segue:

```
// C++ code
//
bool w = 1;
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
```

```
{
  digitalWrite(LED_BUILTIN, w);
  delay(1000); // Wait for 1000 millisecond(s)
  w = !w;
}
```

A cada vez que o código de **`loop()`** é executado, a linha **`w = !w;`** complementa o próprio valor de **`w`** que muda de 0 para 1 ou vice-versa (como um flip-flop). Esse código pode ser ainda modificado para:

```
// C++ code
//
bool w = 1;
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  blink();
}

void blink()
{
  digitalWrite(LED_BUILTIN, w);
  delay(1000); // Wait for 1000 millisecond(s)
  w = !w;
}
```

Esse código certamente não é eficiente para uso em projetos, mas é adequado para ilustrar o princípio do flip-plop lógico.

Luiz C. Vieira - 2024/01/20