Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Automação e Sistemas
Engenharia de Controle e Automação
Introdução à Informática para Controle e Automação - DAS 5334

Aula 3 — Execução Condicional[1]

1 Fluxo de execução

O fluxo de execução refere-se à ordem com que os comandos são executados em um programa. Vimos na primeira aula que um programa em Arduino primeiro executa todos os comandos, de cima para baixo, dentro da função `setup()`. Depois o programa executa todos os comandos dentro da função `loop()` e os repete até que o Arduino seja desligado ou o botão *reset* seja pressionado. Na seção seguinte, e ao longo do curso, veremos que o fluxo de execução pode ser alterado e isso será muito útil para a resolução de problemas.

2 Execução condicional

A execução condicional de comandos nos dá a habilidade de verificar condições e direcionar o comportamento do programa de acordo com estas condições. Assim, com base nas condições verificadas é possível seguir caminhos diferentes no programa, isto é, direcionar o fluxo de execução. A verificação de condições tipicamente dependerá do uso de operadores de comparação e booleanos, ou seja, de expressões booleanas. A base da execução condicional é o comando condicional `if`. Por exemplo:

```
1  if (digitalRead(A3) == HIGH)
2    lcd.print("A3 HIGH!");
```

O comando condicional é formado pela palavra chave `if`, seguida de um expressão booleana entre parênteses e de um comando que será executado caso a condição seja verdadeira. No caso do exemplo, se a expressão `digitalRead(A3) == HIGH` resultar `true` (pino `A3` estiver em nível alto), mostra-se um texto no *display* LCD. Caso contrário, o comando é ignorado.

O exemplo mostrou um caso com apenas um comando a ser executado. Porém, na maioria das vezes deseja-se executar mais de um comando. Neste caso os comandos são agrupados entre chaves (`{}`) formando um bloco de código ou um comando composto. Assim, no exemplo anterior, se além de mostrar o valor do pino `A3` no *display* LCD, quiséssemos alterar o valor de um pino de saída, teríamos:

```
1  if (digitalRead(A3) == HIGH) {
2    lcd.print("A3 HIGH!");
3    digitalWrite(7, HIGH);
4  }
```

Nesse exemplo, apenas dois comandos são condicionalmente executados, mas qualquer número de comandos poderia ser incluído. Veja que o uso do comando condicional, nos deu a possibilidade de alterar o fluxo de execução.

2.1 Execução alternativa

No exemplo anterior, o programa continuaria normalmente o fluxo de execução caso a condição fosse falsa. O mais comum, entretanto, é que se deseje executar um outro conjunto de comandos. Neste caso o comando `if` é acompanhado de `else` da seguinte maneira:

```
1  if (digitalRead(A3) == HIGH) {
2    lcd.print("A3 HIGH!");
3  }
4  else {
5    lcd.print("A3 LOW!");
6  }
```

Se o valor no pino `A3` for `LOW`, será executado o bloco de código após o `else`, antes de seguir o fluxo normal de execução. Cada alternativa é chamada de ramo.

[1] Baseado em conteúdo do livro Think Python 2nd Edition by Allen B. Downey e conteúdo da página oficial do Arduino.

2.2 Condicionais encadeados

Pode ser que seja necessário termos mais do que dois ramos. Nesse caso, `if` e `else` podem ser usados de forma encadeada. Por exemplo:

```
1  if ((digitalRead(A3) == HIGH) && (digitalRead(A2) == HIGH)) {
2    lcd.print("A3 HIGH A2 HIGH!");
3  }
4  else if ((digitalRead(A3) == LOW) && (digitalRead(A2) == HIGH)) {
5    lcd.print("A3 LOW A2 HIGH!");
6  }
7  else if ((digitalRead(A3) == HIGH) && (digitalRead(A2) == LOW)) {
8    lcd.print("A3 HIGH A2 LOW!");
9  }
10 else {
11   lcd.print("A3 LOW A2 LOW!");
12 }
```

Nesse exemplo, cada `else` é seguido por um novo comando `if`, exceto o último. Há quatro ramos. Para o último ramo, nas linhas 10 a 12, a verificação da condição é desnecessária, pois é a última condição possível. Ainda assim, a condição poderia ter sido colocada para ficar explícita no código. Veja nas linhas 1, 4 e 7 que usamos operadores booleanos para formar condições mais elaboradas.

As condições alternativas são verificadas em ordem e uma vez que a condição seja verdadeira, o respectivo ramo será executado e a seguir o programa volta a seguir o fluxo normal de execução.

2.3 Condicionais aninhados

Condicionais aninhados ocorrem quando temos execuções alternativas dentro de execuções alternativas. O código do exemplo anterior poderia ser programado por meio de condicionais aninhados:

```
1  if (digitalRead(A3) == HIGH) {
2    if (digitalRead(A2) == HIGH) {
3      lcd.print("A3 HIGH A2 HIGH!");
4    }
5    else {
6      lcd.print("A3 HIGH A2 LOW!");
7    }
8  }
9  else if (digitalRead(A3) == LOW) {
10   if (digitalRead(A2) == HIGH) {
11     lcd.print("A3 LOW A2 HIGH!");
12   }
13   else {
14     lcd.print("A3 LOW A2 LOW!");
15   }
16 }
```

No exemplo, temos dois ramos cada um com outros dois ramos. Condicionais aninhados devem ser evitados sempre que possível por meio do uso de operadores lógicos, uma vez que são mais sujeitos a erros e mais difíceis de debugar.

3 *Debugging*

O erros de programação, ou *bugs*, podem ser classificados em três tipos:

- **Erros de sintaxe** são aqueles que envolvem a estrutura do programa e as regras a respeito dessa estrutura. Em outras palavras, acontece quando a sintaxe não é respeitada. Um exemplo de erro de sintaxe é tentar usar um operador aritmético binário (que requer dois operandos), como o de adição, com apenas um operando: `5 +`. Outro exemplo, é esquecer de encerrar um comando com ponto-e-vírgula, como visto no Problema 3 da Aula 1. No Problema 4 da Aula 1 a falta da aspas dupla também caracterizou um erro de sintaxe. Os erros de sintaxe são acusados em tempo de compilação.

- **Erros de execução** acontecem durante a execução do programa e, em computadores, geralmente interrompem inadvertidamente sua execução, exceto se for dado tratamento específico no código. No Arduino, erros de execução provavelmente levarão a comportamentos erráticos do programa. No estágio que nos encontramos na disciplina, ainda é prematuro fornecer exemplos desse tipo de erro. Erros de execução também são chamados de exceções e medidas para evitá-los são chamadas de tratamento de exceções.

- **Erros de semântica** acontecem quando o programa é compilado e executa normalmente, mas não faz a tarefa desejada, ou não a faz corretamente. Isso significa que o código que escrevemos não reflete aquilo que queríamos que o programa fizesse. Um exemplo deste tipo de erro seria o caso em que queremos acender um LED conectado ao pino 12, mas enviamos o sinal para o pino 11 que acaba por ter o LED acendido.

4 Problemas

A seguir resolveremos os problemas desta aula. Acesse o circuito Estacionamento 1 e faça uma cópia em sua própria conta com um clique no botão `Copiar e Tinker` se ainda não o tiver feito. Já usamos esse circuito na aula passada, mas convém revisar os recursos disponíveis. São quatro LEDs vermelhos, três LEDs verdes, quatro chaves (caixinhas vermelhas com o centro branco). Os LEDs estão conectados aos pinos `D0`, `D1`, `D6`, `D7`, `D8`, `D9`, e `D10`, que devem ser configurados como `OUTPUT`. As chaves estão conectadas aos pinos `A0`, `A1`, `A2`, e `A3`, que devem ser configurados como `INPUT`. Há também um LED verde integrado à placa e conectado internamente ao pino 13, o qual deve ser configurado como `OUTPUT`. Os componentes eletrônicos do circuito estão conectados de tal maneira que um valor `LOW` num pino de saída apaga o respectivo LED e um valor `HIGH` acende o LED. As chaves como mostradas na Figura 1, mantêm nível `LOW` no respectivo pino de entrada, enquanto se estiverem na posição 1 (para cima) mantêm nível `HIGH`. Ao final do último problema teremos usado todos esses recursos por meio de nossos programas.

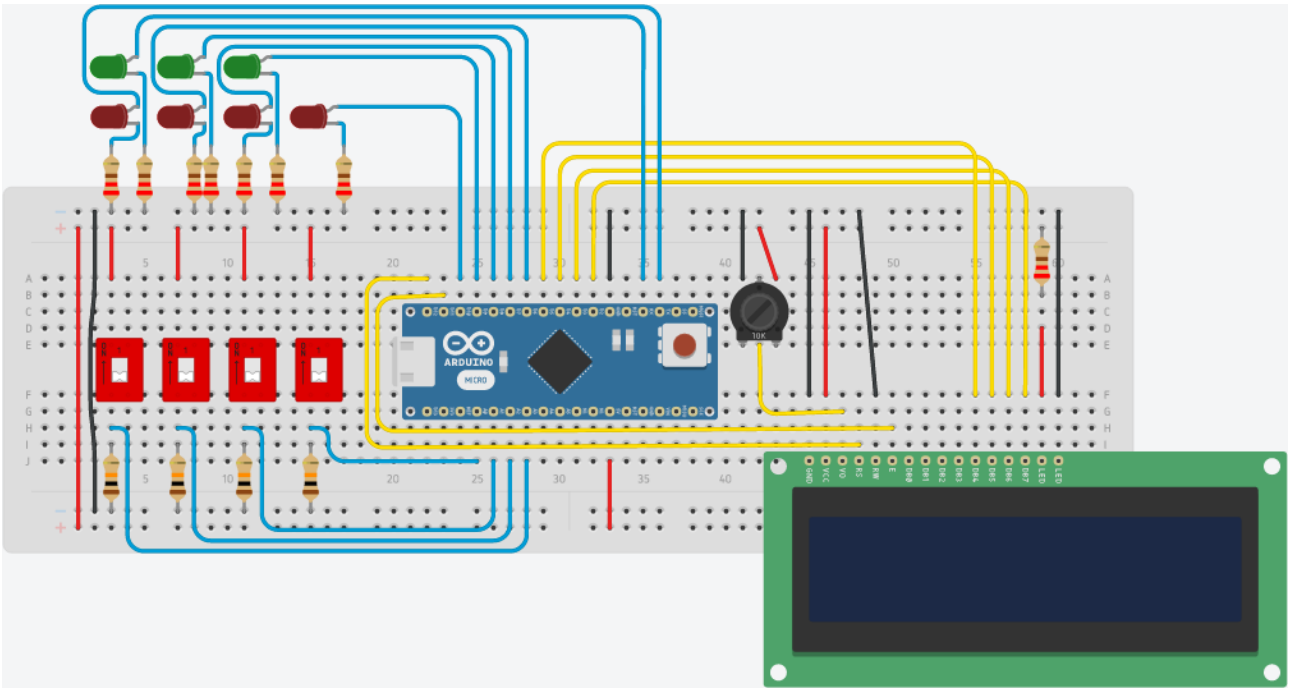

Figura 1: Circuito Estacionamento 1.

Problema 1

Reveja os problemas da aula anterior e analise em quais a execução condicional poderia ser utilizada no lugar de operadores.

Problema 2

Hoje em dia é bastante comum que estabelecimentos com grandes estacionamentos, como supermercados e shopping centers, tenham indicadores de vagas livres instalados. Mais especificamente, algum tipo de sensor de presença é colocado em cada uma das vagas. Se o sensor não detecta a presença de um carro na vaga, uma luz verde fica acesa e indica para motoristas, à distância, que a vaga está livre. Se o sensor detecta a presença de um carro na vaga, uma luz vermelha fica acesa e indica que a vaga está ocupada. Além disso, painéis na entrada ou dentro do estacionamento indicam o número de vagas livres.
Escreva um programa para o circuito Estacionamento 1 que funcione como o sistema descrito. Há quatro vagas, numeradas da esquerda para a direita de 1 a 4, cada uma equipada com um sensor de presença (chave), um LED vermelho e um LED verde. A Tabela 1 apresenta os pinos correspondentes a cada vaga. Lembre-se que o LED verde da quarta vaga está integrado à placa do Arduino Micro e conectado ao pino 13. A Figura 2 mostra o caso com 3 vagas disponíveis.

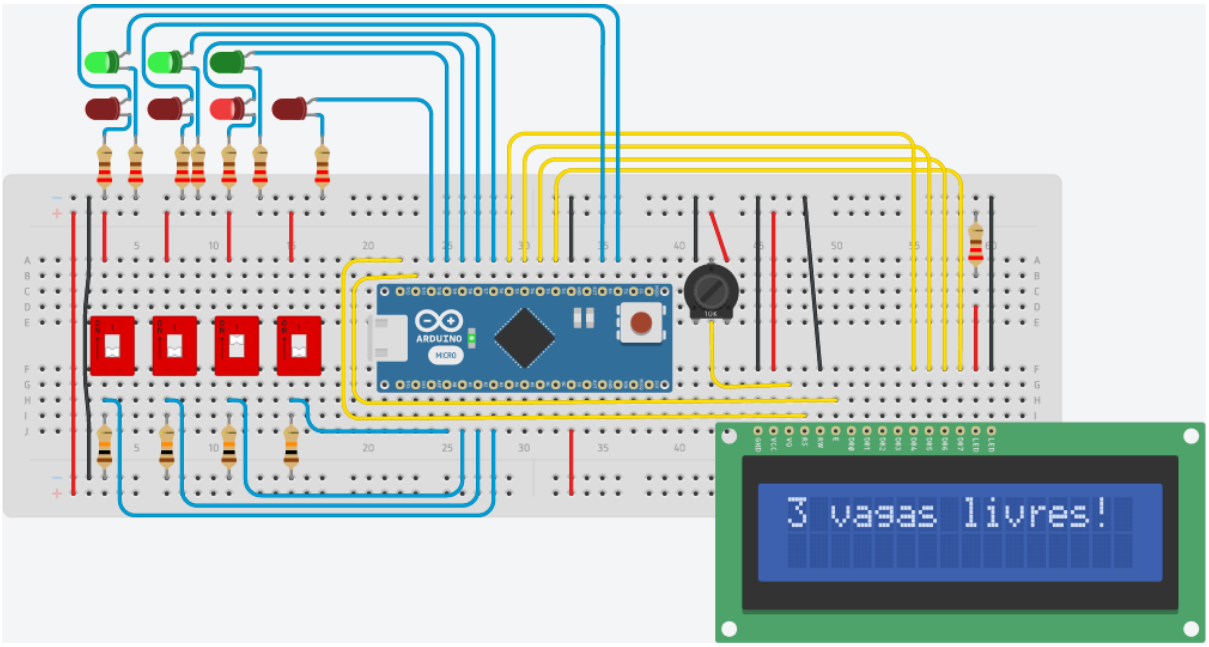

Figura 2: Exemplo do resultado do Problema 3 com três vagas disponíveis.

Tabela 1: Identificação dos pinos e dispositivos.

	Vaga 1	Vaga 2	Vaga 3	Vaga 4
Chave	`A3`	`A2`	`A1`	`A0`
LED Vermelho	`D1`	`D6`	`D8`	`D10`
LED Verde	`D0`	`D7`	`D9`	`D13`

Caso todas as vagas estejam ocupadas, a mensagem dever ser “LOTADO”.
Lembre de usar a função `lcd.home()`. Para evitar que mensagens anteriores atrapalhem as novas, faça textos com 16 caracteres, por exemplo, usando espaços em branco para completar as mensagens menores.
Sugestão: primeiro implemente o programa para que a chave e os LEDs funcionem. Depois que esta parte do programa estiver funcionando, modifique o código para mostrar o número de vagas do *display* LCD.

Problema 3

Sistemas de estacionamento como o descrito acima são ainda mais úteis se os painéis dispostos dentro do estacionamento indicarem o número de vagas disponíveis em cada direção. Modifique o programa para que mostre o número de vagas disponíveis à esquerda e à direita. As vagas 1 e 2 ficam à esquerda e as vagas 3 e 4 ficam à direita. A Figura 3 mostra o caso com 1 vaga disponível à esquerda e 2 vagas

disponíveis à direita.

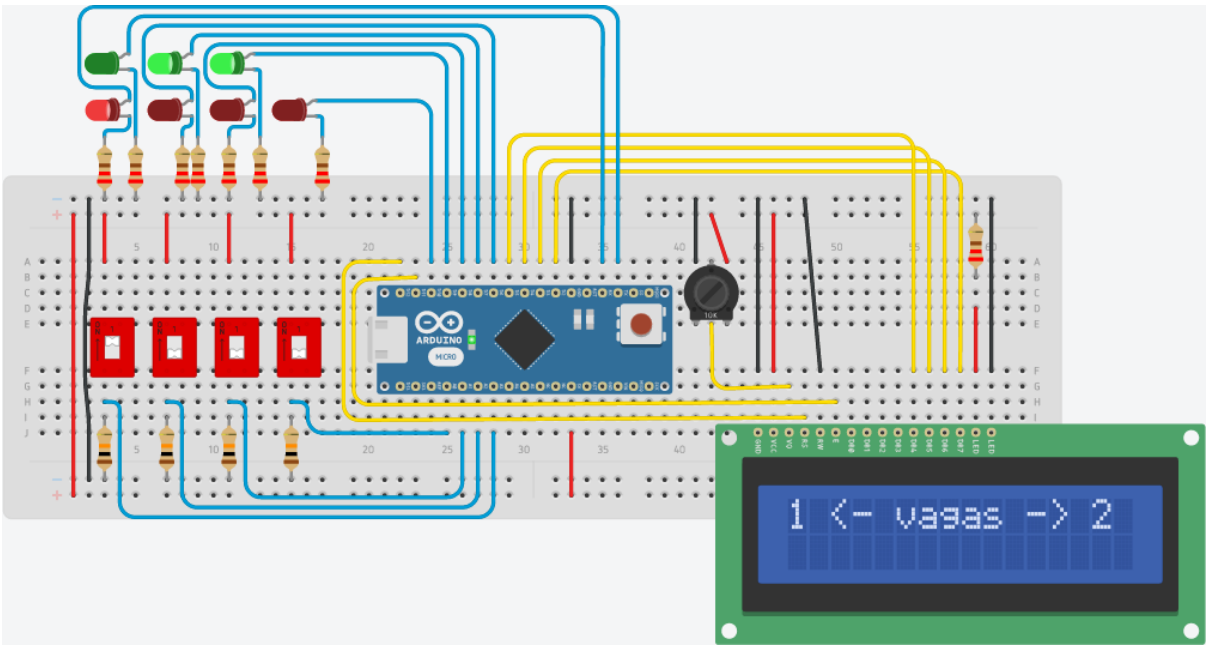

Figura 3: Exemplo do resultado com 1 vaga disponível à esquerda e 2 vagas disponíveis à direita.

Problema 4

Considere um sistema em que a velocidade com que os LEDs se alternam varia conforme a ultrapassagem de limites. São usados dois LEDs do circuito Estacionamento 1, um vermelho e um verde, e as quatro chaves. Cada chave corresponde a um limite. Se nenhuma chave estiver ativada, os LEDs se alternam com intervalo de 1 s. Se a chave 1 estiver ativada, os LEDs se alternam com intervalo de 0,750 s. Se as chaves 1 e 2 estiverem ativadas, os LEDs se alternam com intervalo de 0,5 s. Se as chaves 1, 2 e 3 estiverem ativadas, os LEDs se alternam com intervalo de 0,250 s. Se todas as chaves estiverem ativadas, os LEDs se alternam com intervalo de 0,125 s. Qualquer outra combinação de ativação das chaves faz com que os LEDs se alternem com intervalo de 0,05 segundos e exibe uma mensagem de erro no *display* LCD. A mensagem de erro desaparece quando o sistema volta a operar corretamente. O programa na listagem a seguir deveria implementar um sistema como o descrito, mas não funciona corretamente. Identifique o(s) erro(s) do programa, o(s) seu(s) tipo(s) e corrija-o(s).

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  pinMode(A3, INPUT);
  pinMode(1, OUTPUT);
  pinMode(0, OUTPUT);

  pinMode(A2, INPUT);
  pinMode(A1, INPUT);
  pinMode(A0, INPUT);

  lcd.begin(16, 2);
  lcd.print("Inicio operacao!");
}

void loop() {
```

```
digitalWrite(1, HIGH);
digitalWrite(0, LOW);
if ((digitalRead(A3) == LOW) || (digitalRead(A2) == LOW) ||
    (digitalRead(A1) == LOW) || (digitalRead(A0) == LOW)) {
  lcd.home();
  lcd.print("Em operacao!    ");
  delay(1000);
}
else if ((digitalRead(A3) == HIGH) || (digitalRead(A2) == LOW) ||
         (digitalRead(A1) == LOW)  || (digitalRead(A0) == LOW)) {
  lcd.home();
  lcd.print("Em operacao!    ");
  delay(750);
}
else if ((digitalRead(A3) == HIGH) || (digitalRead(A2) == HIGH) ||
         (digitalRead(A1) == LOW)  || (digitalRead(A0) == LOW)) {
  lcd.home();
  lcd.print("Em operacao!    ");
  delay(500);
}
else if ((digitalRead(A3) == HIGH) || (digitalRead(A2) == HIGH) ||
         (digitalRead(A1) == HIGH) || (digitalRead(A0) == LOW)) {
  lcd.home();
  lcd.print("Em operacao!    ");
  delay(250);
}
else if ((digitalRead(A3) == HIGH) || (digitalRead(A2) == HIGH) ||
         (digitalRead(A1) == HIGH) || (digitalRead(A0) == HIGH)) {
  lcd.home();
  lcd.print("Em operacao!    ");
  delay(125);
}
else {
  lcd.home();
  lcd.print("ERRO!           ");
  delay(50);
}

// muda os LEDs

digitalWrite(1, LOW);
digitalWrite(0, HIGH);
if ((digitalRead(A3) == LOW) || (digitalRead(A2) == LOW) ||
    (digitalRead(A1) == LOW) || (digitalRead(A0) == LOW)) {
  lcd.home();
  lcd.print("Em operacao!    ");
  delay(1000);
}
else if ((digitalRead(A3) == HIGH) || (digitalRead(A2) == LOW) ||
         (digitalRead(A1) == LOW)  || (digitalRead(A0) == LOW)) {
  lcd.home();
  lcd.print("Em operacao!    ");
  delay(750);
}
else if ((digitalRead(A3) == HIGH) || (digitalRead(A2) == HIGH) ||
         (digitalRead(A1) == LOW)  || (digitalRead(A0) == LOW)) {
  lcd.home();
```

```
    lcd.print("Em operacao!    ");
    delay(500);
  }
  else if ((digitalRead(A3) == HIGH) || (digitalRead(A2) == HIGH) ||
           (digitalRead(A1) == HIGH) || (digitalRead(A0) == LOW)) {
    lcd.home();
    lcd.print("Em operacao!    ");
    delay(250);
  }
  else if ((digitalRead(A3) == HIGH) || (digitalRead(A2) == HIGH) ||
           (digitalRead(A1) == HIGH) || (digitalRead(A0) == HIGH)) {
    lcd.home();
    lcd.print("Em operacao!    ");
    delay(125);
  }
  else {
    lcd.home();
    lcd.print("ERRO!           ");
    delay(50);
  }
}
```