05

Programando o mapeamento

Transcrição

Na IDE do Arduino, podemos implementar o programa, visando o mapeamento entre o servo motor e o joystick.

A primeira coisa que podemos fazer é importar a biblioteca do servo motor:

```
#include <Servo.h>

void setup() {
}

void loop() {
}
```

Vamos definir um ângulo inicial para o servo motor, que será de 90°. Além disso, vamos criar constantes para mapear os pinos dos eixos X e Y do joystick:

```
#include <Servo.h>

#define ANGULO_INICIAL_MOTOR 90

// --- Mapeamento dos Joysticks ---
#define joystick1X A0
#define joystick1Y A1

void setup() {
}

void loop() {
}
```

Resta agora definir o servo motor. Definiremos o primeiro como `motorBase` :

```
#include <Servo.h>

#define ANGULO_INICIAL_MOTOR 90

// --- Mapeamento dos Joysticks ---
#define joystick1X A0
#define joystick1Y A1

// --- Mapeamento dos Servos ---
Servo motorBase;

void setup() {
}
```

```
void loop() {
}
```

Dentro da função `setup()`, precisamos definir o pino está o controle do servo motor, o pino **5**:

```
void setup() {
  motorBase.attach(5);
}
```

Ainda nesse função, vamos definir as portas do joystick como portas de entrada. Por enquanto só faremos o mapeamento do pino X:

```
void setup() {
  motorBase.attach(5);
  pinMode(joystick1X, INPUT);
}
```

E colocamos o motor na posição de 90º, guardada na constante `ANGULO_INICIAL_MOTOR`:

```
void setup() {
  motorBase.attach(5);
  pinMode(joystick1X, INPUT);
  motorBase.write(ANGULO_INICIAL_MOTOR);
}
```

Até aqui nenhuma novidade, vamos então iniciar o mapeamento da entrada analógica para o servo motor.

Mapeamento da entrada analógica para o servo motor

Dentro da funçãon `loop()`, vamos criar a variável `posX`, que irá receber o valor do eixo X do joystick:

```
void loop() {
  int posX = analogRead(joystick1X);
}
```

Agora precisamos mapear o intervalo numérico da entrada analógica (0 até 1023) para o intervalo de ângulos do servo motor (0 até 180). Para isso existe a função `map()`, que permite efetuar o mapeamento de um intervalo numérico em outro intervalo numérico desejado. Isso significa que em um intervalo numérico, que vai de um valor mínimo até um valor máximo, o valor mínimo será mapeado em um novo valor mínimo, e o valor máximo será mapeado em um novo valor máximo, assim como os valores intermediários, que serão remapeados em novos valores intermediários, de forma correspondente.

No nosso caso, vamos mapear o intervalo 0-1023 para o intervalo 0-180. A função recebe por parâmetro o número a ser mapeado (no nosso caso, o número contido na variável `posX`), o limite inferior do intervalo de valores atual (**0**), o limite superior do intervalo de valores atual (como o valor de `posX` vem da entrada analógica, o limite superior é **1023**), além do limite inferior do novo intervalo de valores mapeados (**0**) e do limite superior do novo intervalo de valores mapeados (**180**, maior ângulo possível do servo motor).

Então, o código fica assim:

```
void loop() {
  int posX = analogRead(joystick1X);
  posX = map(posX, 0, 1023, 0, 180);
}
```

Agora que `posX` representa um ângulo válido, vamos passá-lo para o servo motor e adicionar um *delay* para garantir um intervalo entre as leituras:

```
void loop() {
  int posX = analogRead(joystick1X);
  posX = map(posX, 0, 1023, 0, 180);
  motorBase.write(posX);
  delay(100);
}
```

Agora que o código está pronto, vamos testá-lo no próximo vídeo!