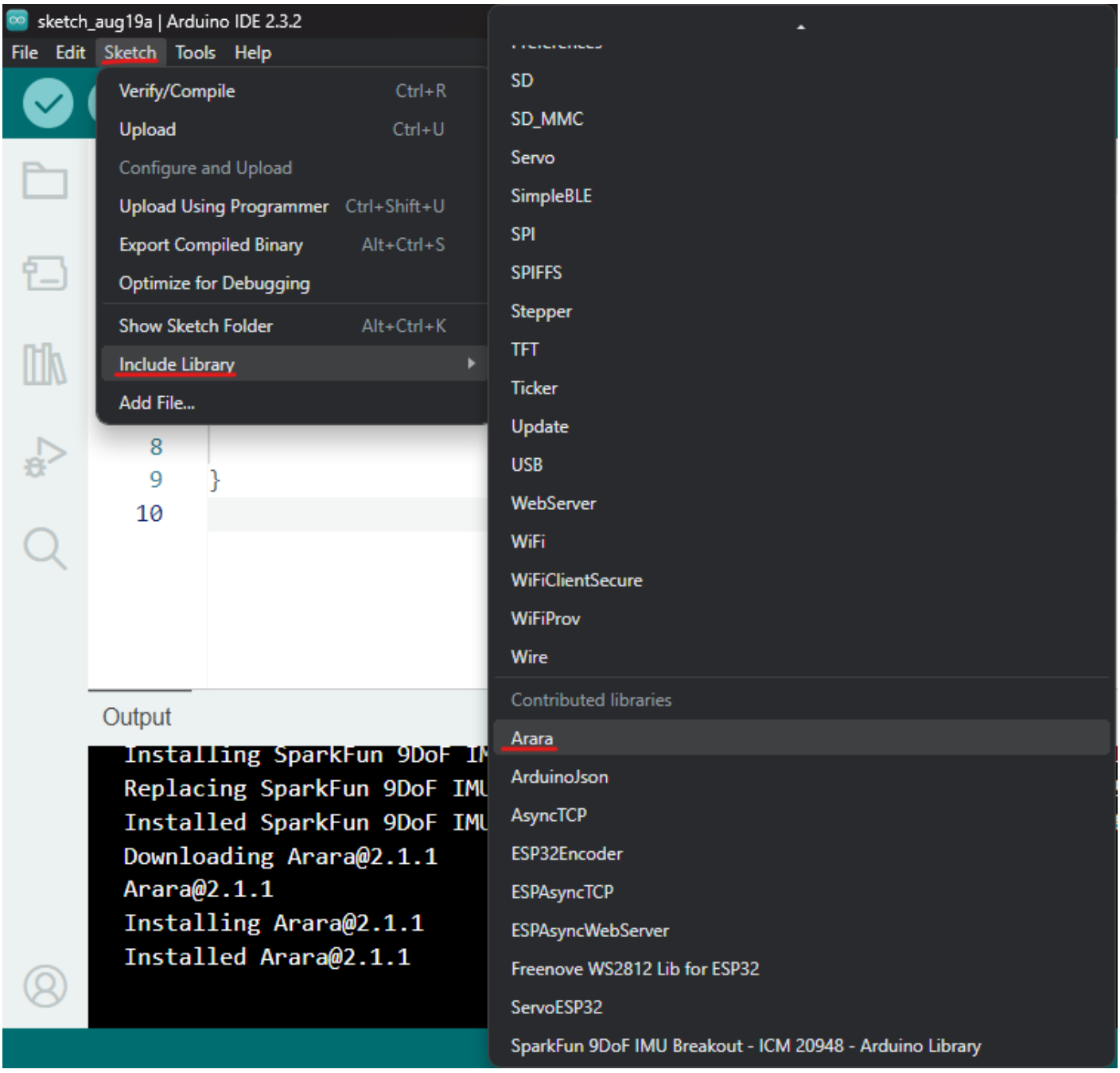
Programação

Neste capítulo veremos como podemos utilizar o hardware disponível pela Arara

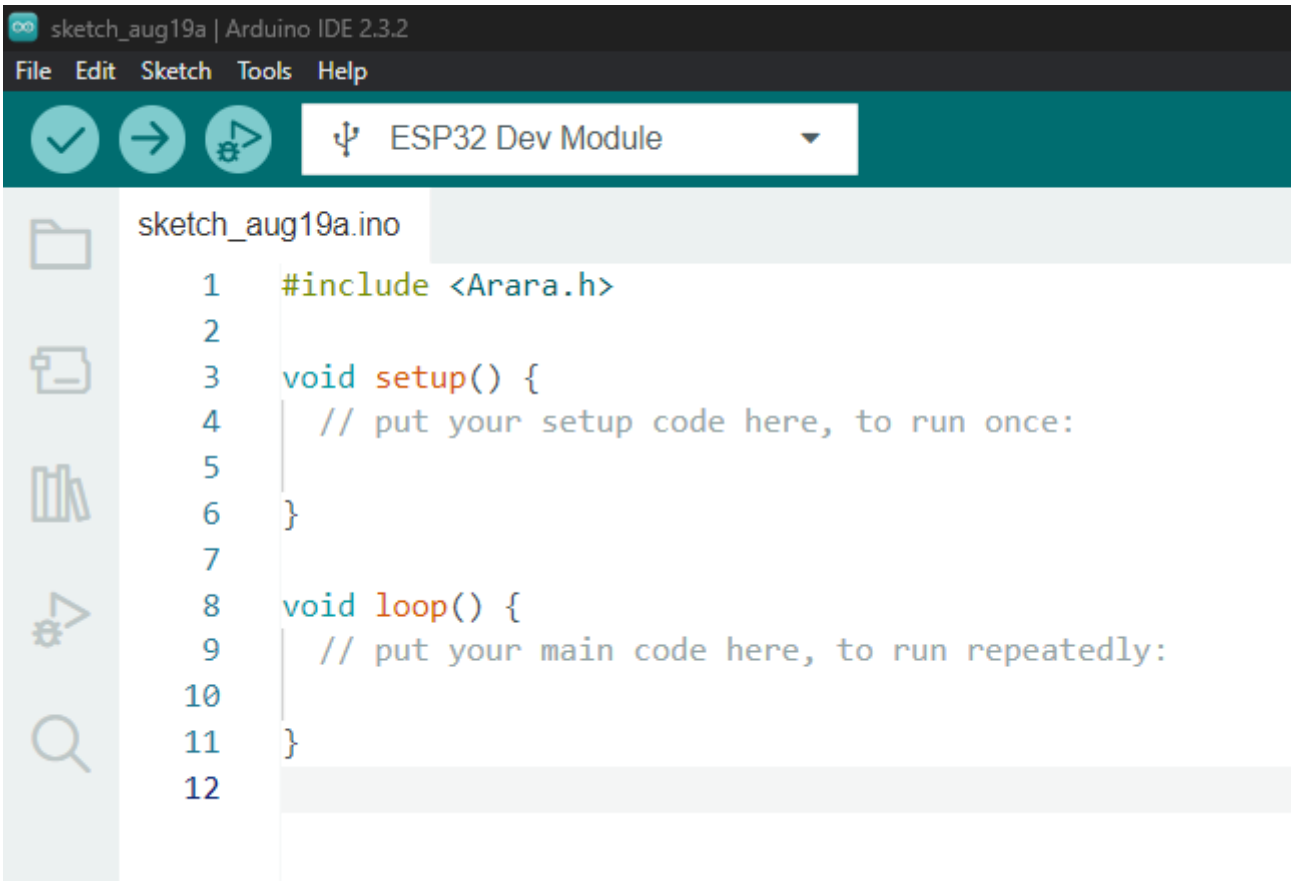
Importação da biblioteca

Esta página pode ser considerada como introdutória a todas as outras, pois devemos realizar esse processo antes de começar a programar qualquer esboço (nome de um projeto na Arduino IDE).

O processo dito anteriormente, é a importação da biblioteca, ou seja, utiliza-lá em nosso código. Para fazer isso é siga os processos abaixo.



Seu esboço deve ficar da seguinte maneira.



> Você também pode escrever diretamente essa linha #include, é ela quem faz a importação da biblioteca ao projeto

Motores

Para programar os motores é bem simples, afinal, eles já estão declarados na biblioteca com suas portas correspondentes.

Portanto, podemos apenas fazer o seguinte para acionar um motor.

```
#include <Arara.h>

void setup() {
  // put your setup code here, to run once:
  motor1.setPower(0.5);
}

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

}
```

Esse código fará com que o motor opere a, aproximadamente, 50% de sua velocidade máxima. E é claro que como dito anteriormente, temos um objeto correspondente para cada porta, como indicado a seguir.

> Com esse parágrafo anterior é importante entender o seguinte, o parâmetro setPower só aceita valores entre -1.0 até 1.0, então 1.0 para 100%, vale dizer que é o mesmo para valores negativos, mas agora o motor irá girar em outra direção.

```
#include <Arara.h>

void setup() {
  // put your setup code here, to run once:
  motor1.setPower(0.5);
  motor2.setPower(1.0);
  motor3.setPower(0.0);
  motor4.setPower(-0.5);
}
```

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


}
```

Tente nesse momento entender o que cada motor fará.

> Resposta:
>
> Motor da porta 1: operando a 50% de sua velocidade para "frente";
>
> Motor da porta 2: operando a 100% de sua velocidade para "frente";
>
> Motor da porta 3: parado;
>
> Motor da porta 4: operando a 50% de sua velocidade para "trás".

Encoder

De forma breve, um encoder é um sensor digital que tem como função medir a posição/velocidade de um motor.

Em nossa biblioteca ele já é declarado como objeto inerente do motor, portanto, para utiliza-lo podemos fazer o seguinte.

```
#include <Arara.h>

void setup() {
  // put your setup code here, to run once:
  motor1.setPower(0.5);
}

void loop() {
  // put your main code here, to run  repeatedly:
  double position = motor1.encoder.getPosition();
}
```

> Agora essa variável position ficará armazenando o valor do encoder repetidamente

Podemos verificar seu valor imprimindo-a no monitor serial. Da seguinte forma.

```
#include <Arara.h>

void setup() {
  // put your setup code here, to run once:
  motor1.setPower(0.5);
}

void loop() {
  // put your main code here, to run  repeatedly:
  double position = motor1.encoder.getPosition();
  Serial.print("Encoder: ");
  Serial.println(position);
}
```

Servos

Um servo também é um acionador, como um motor, a diferença é que seu movimento tem como intenção ser mais preciso, de forma que podemos dizer para qual ângulo de seu escopo ele pode se mover.

Então, para movimentar ele para uma posição específica, podemos fazer da seguinte forma.

```
#include <Arara.h>

void setup() {
  // put your setup code here, to run once:
  servo1.setPosition(270);
}

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

}
```

Nesse código movimentamos o servo da porta para a posição de 270º de seu escopo (no caso do servo da REV Robotics esse é considerado o limite).

> Importante adicionar que é a mesma ideia de programar um motor, já existem os 3 servos declarados na biblioteca, podemos apenas chama-los no código.

```
#include <Arara.h>

void setup() {
  // put your setup code here, to run once:
  servo1.setPosition(270);
  servo2.setPosition(0);
  servo3.setPosition(180);
}

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

```
}
```

Sensores digitais

Um sensor digital é um dispositivo que mede apenas dois valores, verdadeiro ou falso.

Para utiliza-lo no código o seguinte pode ser feito.

```
#include <Arara.h>

Digital di1(PortasDigitais::PORTA_1);

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

}

void loop() {
  // put your main code here, to run  repeatedly:
  bool valorDigital = di1.getInput();
}
```

> Observe que diferente dos outros dispositivos vistos anteriormente, a porta digital deve ser declarada

Também podemos imprimir seu valor no monitor serial.

```
#include <Arara.h>

Digital di1(PortasDigitais::PORTA_1);

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

}

void loop() {
```

```
  // put your main code here, to run  repeatedly:
  bool valorDigital = di1.getInput();
  Serial.println(valorDigital);
}
```

IMU

IMU é um dispositivo, portanto, um conjunto de componentes que tem como intenção final indicar a posição/velocidade angular de um mecanismo. Seu nome pode ser traduzido como dispositivo de medição inercial.

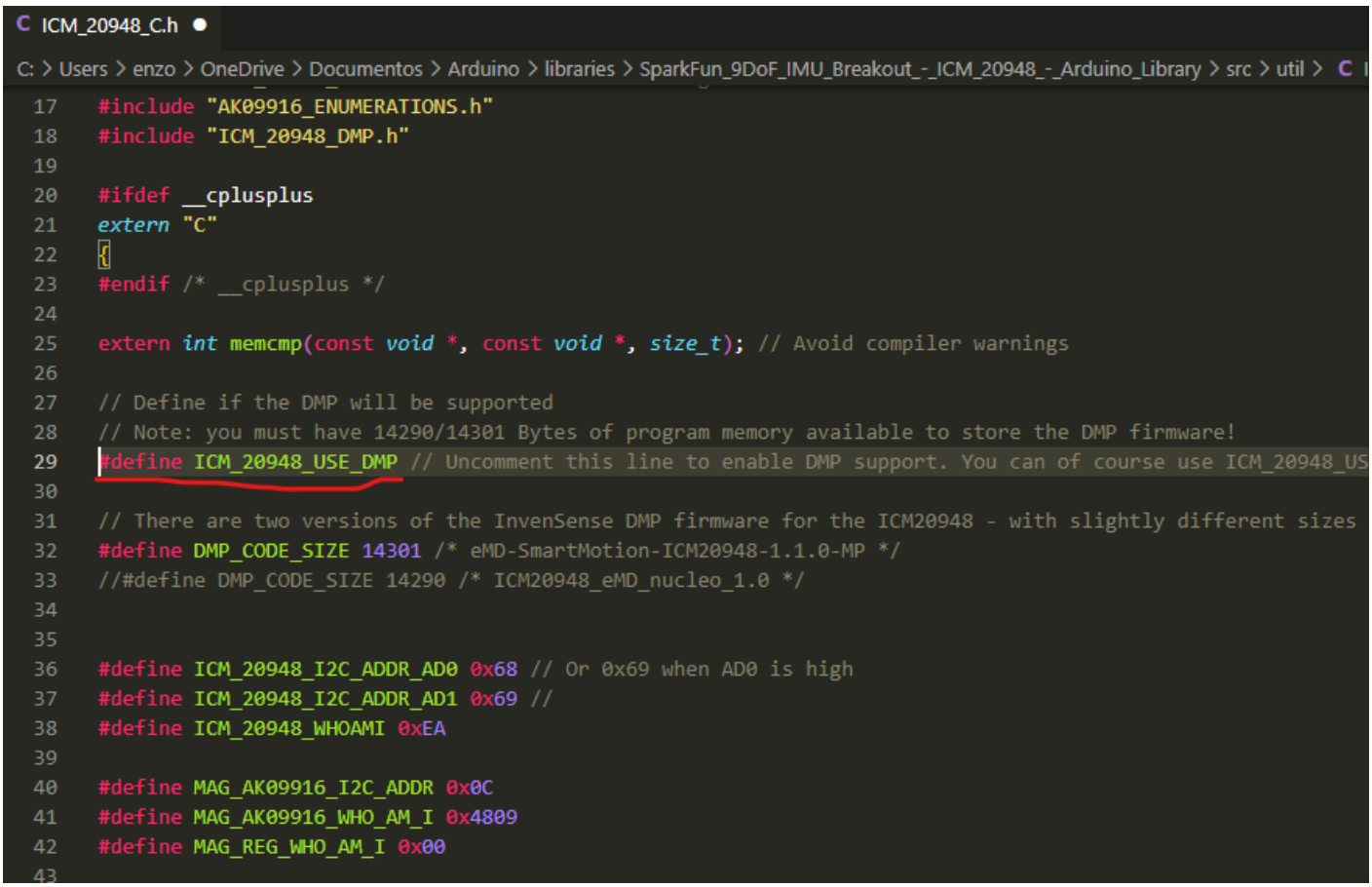
Para utiliza-lo é necessário editar um arquivo da biblioteca do Arduino IDE.

Faça o seguinte caminho em seu computador:

C:\Users\"User"\Documentos\Arduino\libraries\SparkFun_9DoF_IMU_Breakout_-*ICM_20948*-_Arduino_Library\src\util

Abra o arquivo ICM_20948_C.h, caso ele peça algum aplicativo para abrir, use o editor de texto do windows.

Por fim, descomente (remova os caracteres //) da seguinte linha.



Agora em nosso código podemos fazer o seguinte.

```
#include <Arara.h>

IMU imu;

void setup() {
  // put your setup code here, to run once:
  imu.init();
}

void loop() {
  // put your main code here, to run  repeatedly:
  Serial.print("Pitch: ");
  Serial.println(imu.getPitch());
  Serial.print("Roll: ");
  Serial.println(imu.getRoll());
  Serial.print("Yaw: ");
  Serial.println(imu.getYaw());
}
```

Gamepad

Na primeira parte desse livro foi ensinado como instalar a Arara Driver Station, software utilizado para fazer a comunicação wireless entre a Arara e o computador. Essa comunicação tem como finalidade entregar os valores do gamepad a Arara. Portanto, vamos aprender como se da inicio a essa comunicação pela parte de programação da Arara.

Para começar é preciso entender que a comunicação começa chamando a seguinte linha de código.

```
#include <Arara.h>

void setup() {
  Arara.start();
}

void loop() {

}
```

Essa função **start()** faz com que o Wi-Fi da placa seja iniciado, assim como os processos de comunicação.

> Será ensinado como fazer a conexão a placa pela Driver Station em um capítulo posterior desse livro

Agora, nosso objeto de gamepad poderá ter seu valor atualizado quando a driver station for utilizada.

Então, como ele já está declarado, podemos chama-lo da seguinte forma.

```
#include <Arara.h>

void setup() {
  Arara.start();
}

void loop() {
```

```
  double y = gamepad.getLeftAxisY();
  double x = gamepad.getLeftAxisX();
}
```

Esses são os eixos analógicos do controle. Podemos também chamar os botões, como indicado.

```
#include <Arara.h>

void setup() {
  Arara.start();
}

void loop() {
  bool a = gamepad.getButtonA();
  bool b = gamepad.getButtonB();
}
```

> Verifique por si mesmo quais são as funções disponíveis

Código exemplo

Com essas etapas que seguimos anteriormente já é possível programarmos um motor para ser acionado conforme o gamepad se movimenta. Portanto, nessa página está disponibilizado um código de exemplo para mostrar como isso pode ser feito.

```
#include <Arara.h>

void setup() {
  Arara.start();
}

void loop() {
  motor1.setPower(gamepad.getLeftAxisY());
  motor2.setPower(gamepad.getRightAxisY());
}
```

> Isso pode ser traduzido como um acionamento do estilo Tank