UNIVERSIDADE FEDERAL DE CAMPINA GRANDE

CENTRO DE ENGENHARIA E ELÉTRICA E INFORMÁTICA

CURSO DE ENGENHARIA ELÉTRICA

DISCIPLINA TÉCNICAS DE PROGRAMAÇÃO – PERÍODO 2019.1

PROF. DR. MARCUS SALERNO DE AQUINO

PROJETO

PROGRAMAÇÃO ORIENTADA A OBJETOS

BRAÇO MECANICO

DOUGLAS DOS SANTOS GOMES

LUCAS EDSON FARIAS DA SILVA

MAXWELL DOS SANTOS

VICTOR EMANUEL GUIMARÃES DA SILVA

CAMPINA GRANDE/PB

JULHO/2019

Sumário

INTRODUÇÃO

A linguagem de programação C++ é muito versátil. Sabendo disso, o presente projeto utiliza dela e de um controlador da série Arduino Uno para o controle de um braço mecânico. Através da Linguagem C++, voltada ao POO (Programação Orientada a objeto), descrevemos em classes e seus respectivos métodos, atributos e objetos para que fosse possível o devido funcionamento dele. Estas por sua vez são utilizadas para controlar os “Servo-Motores” que estão presentes em cada eixo do braço mecânico, que se movimentam em sentido horário e anti-horário, dependendo do parâmetro passado.

As classes recebem parâmetros vindos de outro código feito no compilador do Arduino que tem a função de instanciar todos os movimentos do braço, descritos a partir de um espaço tridimensional no qual o objeto se encontra, definido a partir de um referencial obtido com a sua posição inicial da Altura, Lonjura e Rotação. Feita a compilação, temos total controle do braço, podendo até gravar um determinado movimento e repeti-lo indefinidamente, ou seja, obedecendo os limites de rotação de cada servo, a criatividade é quem dita o jogo do programador.

OBJETIVOS

Nosso Objetivo é Implementar em C++, utilizando classes e outros conceitos relacionados a POO, o funcionamento de um Braço Mecânico. Ele deve ter sua usabilidade maximizada, possuindo apenas limitações mecânicas naturais, sem limitações causadas por um código pobre de detalhes e funcionalidades.

MATERIAL UTILIZADO

Foram utilizados para este projeto:

SOFTWARES - *Falcon C++* e *Arduino IDE*, além do uso da linguagem de programação C++ nos mesmos e através do Site: *www.lucidchart.com/* foi criado os Diagramas de classes e Diagramas de Casos e Usos.

MATERIAL FÍSICO - Além dos softwares, foram necessários para a criação do Braço, *Um Controlador Arduino Uno, Peças Fabricadas Em MDF, 4 Servo-Motores, 4 Potenciômetros, Botões, LEDS, Uma Protoboard, Cabos De Alimentação, Uma Fonte, E Outros Componentes Eletrônicos Necessários Para O Circuito Funcionar.*

DESCRIÇÃO

Primeiramente tivemos que arrumar as peças e fazer a montagem do braço mecânico, após obter toda parte mecânica, partimos para a implementação em programação POO. Pois o nosso objetivo principal do Projeto é a configuração de um Braço Mecânico, que funcionará a partir da combinação de dois códigos; um em C++ e outro na IDE de um Arduino.

Para isso foi usado uma Biblioteca Arquivo.h escrito em C++ para o Arduino, <Servo.h>, tal código traz as funções que permitem os movimentos do braço, dando o controle dos Servo-Motores presentes nos eixos. Trago logo em seguida um trecho do código utilizado, e sua classe servo:

```
class Servo
{
public:
Servo();
uint8_t attach(int pin);           // attach the given pin to the next free channel, sets pinMode, returns channel number or 0 if failure
uint8_t attach(int pin, int min, int max); // as above but also sets min and max values for writes.
void detach();
void write(int value);             // if value is < 200 its treated as an angle, otherwise as pulse width in microseconds
void writeMicroseconds(int value); // Write pulse width in microseconds
int read();                        // returns current pulse width as an angle between 0 and 180 degrees
int readMicroseconds();            // returns current pulse width in microseconds for this servo (was read_us() in first release)
bool attached();                   // return true if this servo is attached, otherwise false
private:
uint8_t servoIndex;          // index into the channel data for this servo
int8_t min;                  // minimum is this value times 4 added to MIN_PULSE_WIDTH
int8_t max;                  // maximum is this value times 4 added to MAX_PULSE_WIDTH
};
```

A partir da Classe Servo já mostrada, criamos através de Herança a Classe Braço que trará os Atributos e Funções Métodos Responsáveis pelo funcionamento de todo o Braço. Trago logo em seguida um trecho do código utilizado, e sua Classe Braco:

```
#include "Arduino.h"
```

```
#include <Servo.h>

//CLASS BRAÇA ------ SERVO
class braco : public Servo
{
private:
        //CRIANDO OS OBJETOS DA BIBLIOTECA SERVO
        Servo Servo1; //Objeto servo para controlar a base
        Servo Servo2; //Objeto servBracoo para controlar a garra
        Servo Servo3; //Objeto servo para controlar a altura do braÃ§o
        Servo Servo4; //Objeto servo para profundidade a altura do braÃ§o
        //ATRIBUTOS DA CLASSE BRAÇO
        int A, B, C, D, BUT1, BUT2, val1, val2, val3, val4;
public:
        //CONSTRUTOR
        braco();
        //FUNÇÕES METODOS
        void leitura(int, int , int , int); //RESPONSAVEL POR FAZER AS LEITURAS
ENTRE OS MOTORES E POTENCIOMETROS
        void botao(int, int);           //RESPONSAVEL POR FAZER O CODIGO DOS
BOTOES E SEUS RESPECTIVOS MOVIMENTOS
        void mostraestado();            //ELE É RESPONSAVEL POR MOSTRAR NUMA
TELA OS VALORES DAS VARIAVEIS
};
```

Todo arquivo foi implementado com o Arquivo.cpp referente a Classe Braco, é por ele que nos comunicamos com o Arquivo.ino do Arduino e com o Arquivo.h da suas respectiva Classe a qual foi criada. Segue os trechos dos códigos do Arquivo.cpp:

```
#include "Arduino.h"
#include <Servo.h>
#include "bracorobot.h"

//ZERANDO OS ATRIBUTOS
braco::braco()
{
        A = B = C = D = val1 = val2 = val3 = val4 = 0;
}
//FUNÇÃO DA LEITURA
void braco::leitura(int a, int b, int c, int d)
{
        //PASSANDO OS VALORES POR PARAMETRO PEGOS DO ARQUIVO.ino
        A = a;
        B = b;
        C = c;
        D = d;

        //ATRIBUINDO O PINO DO ARDUINO A CADA SERVO
```

```
	Servo1.attach(A);	//ALTURA
	Servo2.attach(B);	//GARRA
	Servo3.attach(C);	//PROFUNDIDADE
	Servo4.attach(D);	//LONJURA

	//PASSA OS VALORES CONVERTIDO DO POTENCIOMETRO QUE VAI DE 0 - 1023
PARA OS SERVO QUE RESISTEM DE 0 - 180.
	val1 = map(analogRead(A1), 0, 1023, 20, 130);    //LIMITES
	Servo1.write(val1);
	val2 = map(analogRead(A0), 0, 1023, 0, 30);     //LIMITES
	Servo2.write(val2);
	val3 = map(analogRead(A2), 0, 1023, 90, 165);    //LIMITES
	Servo3.write(val3);
	val4 = map(analogRead(A3), 0, 1023, 50, 170);     //LIMITES
	Servo4.write(val4);
	delay(100);
}
//FUNÇÃO BOTÃO
int braco::botao(int butt1, int butt2)
{
	//PASSANDO OS VALORES POR PARAMETRO PEGOS DO ARQUIVO.ino
	BUT1 = butt1;
	BUT2 = butt2;
	//PASSANDO QUE OS BOTÕES SERÃO VARIAVEIS DE ENTRADA - INPUT
	pinMode(BUT1, INPUT_PULLUP);
	pinMode(BUT2, INPUT_PULLUP);
	//IF RESPONSAVEL POR PASSAR UNS ESTADOS DE MOVIMENTO DO BOTOES
	if (digitalRead(butt2) == HIGH)
	{
		delay(500);
		Servo1.write(70); //POSIÇÕES DIFERENTES DE CADA SERVO
		Servo2.write(25);
		Servo3.write(120);
		Servo4.write(90);
		for(p = 80; p <= 150; p++)
			Servo4.write(p);
		delay(500);
	};
	if (digitalRead(butt1) == HIGH)
	{
		delay(500);
		Servo1.write(25); //POSIÇÕES DIFERENTES DE CADA SERVO
		Servo2.write(5);
		Servo3.write(100);
		Servo4.write(60);
		for(p = 60; p <= 150; p++)
			Servo4.write(p);
		delay(500);
	};
}
//FUNÇÃO QUE MOSTRA NA TELA OS VALORES
void braco::mostraestado()
{
	//INICIANDO O SERIAL
	Serial.begin(9600);
```

```
        //COLOCANDO LIMITES DE TEMPO
        unsigned long mostradorTimer = 1;
        const unsigned long intervaloMostrador = 1000;
        if ((millis() - mostradorTimer) >= intervaloMostrador)
        {
                Serial.print("*********GARRA***************");
                Serial.print(analogRead(potpin1)); //GARRA
                Serial.print(" Angulo Motor1:   ");
                Serial.println(val1);
                Serial.print("*******************************");

                Serial.print("**********BASE***************");
                Serial.print(analogRead(potpin2)); //BASE
                Serial.print(" Angulo Motor2:   ");
                Serial.println(val2);
                Serial.print("*******************************");

                Serial.print("*********ALTURA*************");
                Serial.print(analogRead(potpin3)); //ALTURA
                Serial.print(" Angulo Motor3:   ");
                Serial.println(val3);
                Serial.print("*******************************");

                Serial.print("*******************************");
                Serial.print(analogRead(potpin4)); //PROFUNDIDADE
                Serial.print(" Angulo Motor4:   ");
                Serial.println(val4);
                Serial.print("*******************************");

                mostradorTimer = millis();
        }
}
```

E por fim, o código do Arquivo.ino ao qual usamos para conectar todos os códigos C++, vale ressaltar que a linguagem de programação do Arduino é Baseada no C++, o que facilita na implementação, segue logo abaixo o documento referente ao .ino, nele criamos o Objeto da Classe Braço e chamamos a nossa biblioteca criada anteriormente:

```
#include <bracorobot.h>
braco teste;
void setup()
{
}
void loop()
{
  teste.leitura(5, 2, 4, 3);
```

```
  teste.mostraestado();
  teste.botao(6, 7);
}
```

Com relação aos Templates, sua utilidade se dá, principalmente, em trabalhar com tipos de dados distintos, sendo assim, podemos criar funções de um tipo genérico (Template) que nos permite operar com estes dados livremente e independendo de seu tipo. Haja visto isto, não se deveu aplicação necessária em nosso projeto, pois não trabalhamos com dados “Distintos” em seus tipos, ou seja, os métodos desenvolvidos só atribuem valores inteiros ou booleanos.

E por fim com relação a aplicação das funções virtuais se dá em virtude da ambiguidade de funções em classes distintas, ou seja, classes que estão envolvidas em uma hierarquia e tem funções com o mesmo nome. sendo assim, não foi possível aplicar este conceito em nosso projeto, pois a classe servo e a braço não possuem métodos ambíguos.

DIAGRAMA DE CASOS E USOS

Abaixo, segue o Diagrama de Casos e usos referente ao código de funcionamento do Braço Mecânico desenvolvido neste projeto:

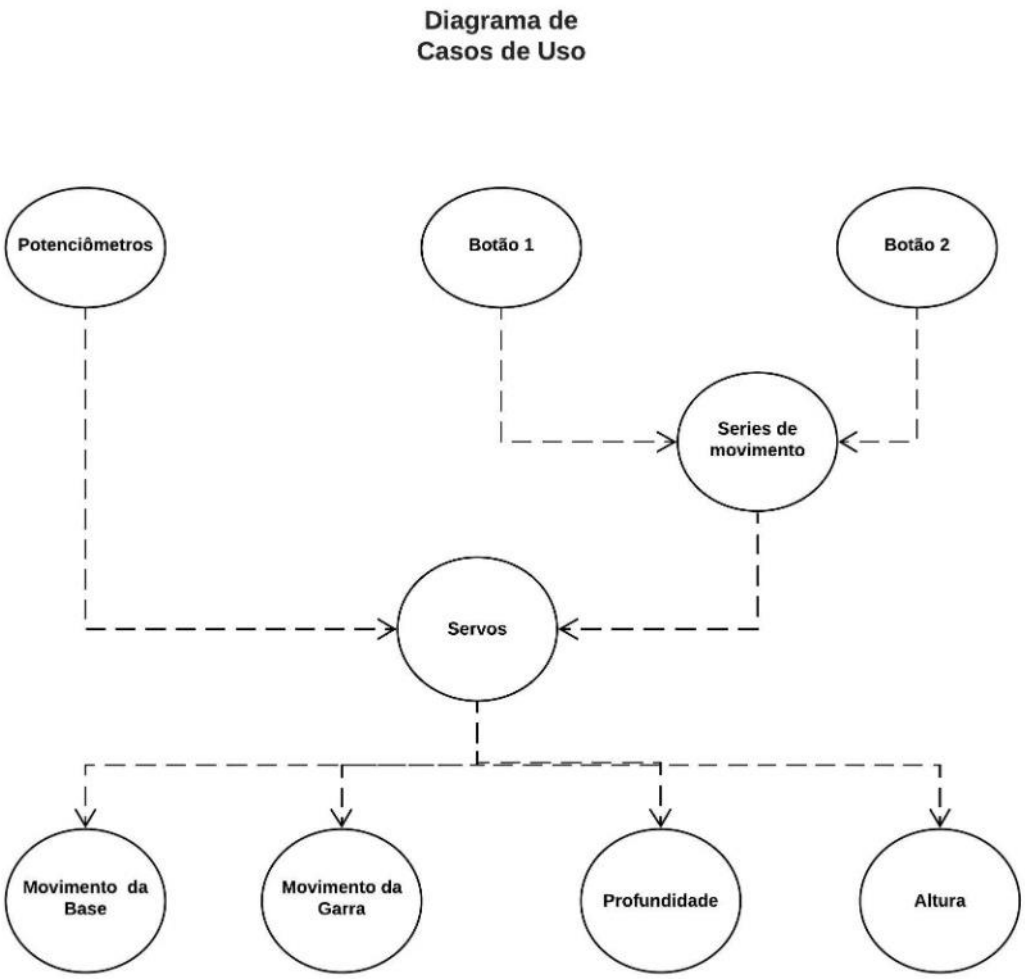

Figura 1 - Diagrama de Casos e Usos

DIAGRAMA DE CLASSES

Abaixo, segue o Diagrama de Classes UML referente ao código de funcionamento do Braço Mecânico desenvolvido neste projeto:

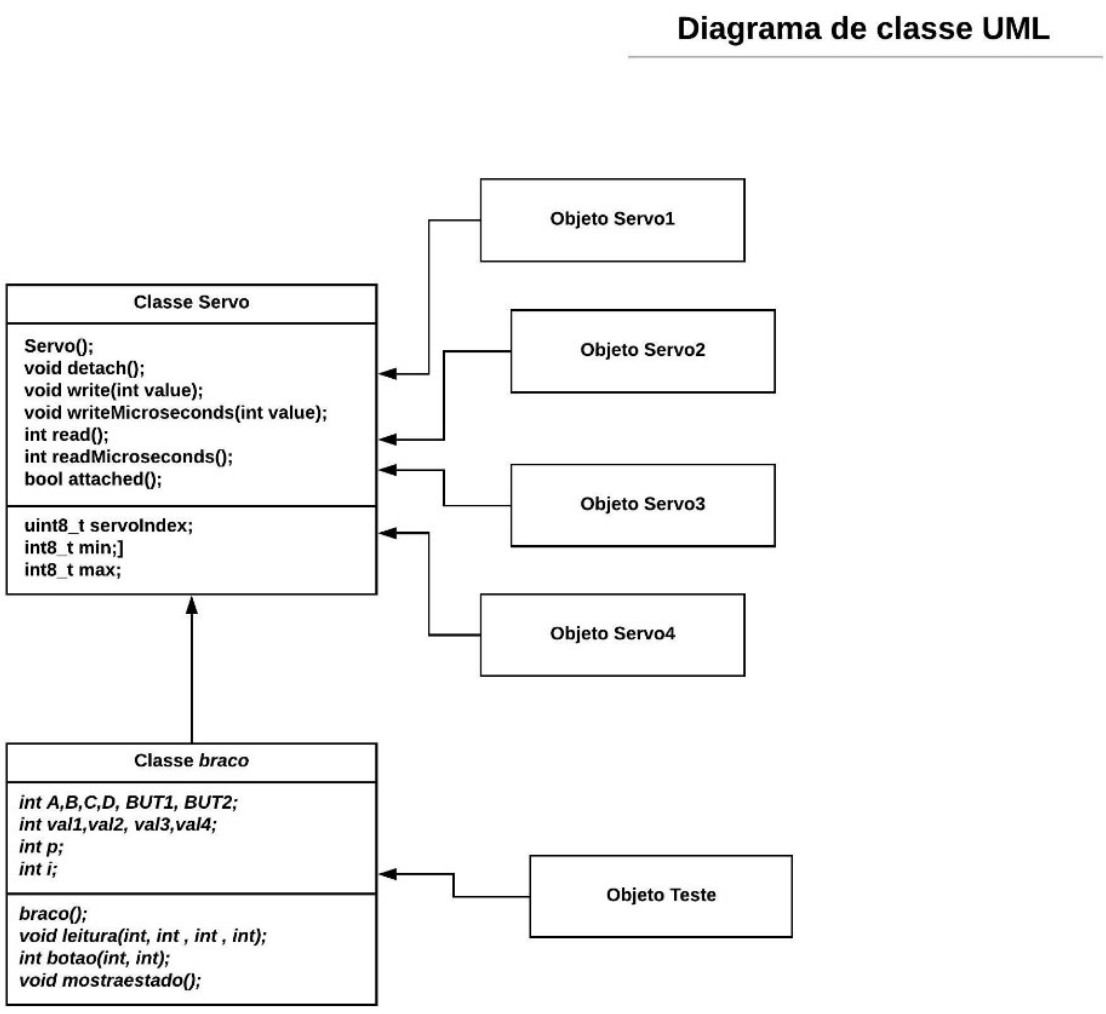

Figura 2 - Diagrama de Classes

CONCLUSÃO

Conclui-se que os objetivos estimados para este projeto foram parcialmente alcançados, visto que conseguimos aplicar quase todos os conhecimentos adquiridos durante o curso em questão para criação do “Código Programado para o Braço”, mas algumas coisas ficaram a desejar.

A partir dos testes feitos com o braço foi visto que os resultados obtidos foram bastante satisfatórios, pois alcançamos total funcionamento cabíveis nas limitações mecânicas do Braço Mecânico.

O que faz com que nossos objetivos estejam parcialmente concretizados são as aplicações de Templates e Virtuais no Código de Programação em C++. Porem com base tanto em nossos estudos teóricos concluímos que em nosso projeto elas se tornam indesejáveis, quanto na prática em que foi constatado inclusive Bugs, ao forçar uma Virtual e/ou Template. Sendo assim para a equipe a validação de que nossos objetivos estimados foram alcançados, ao ver o Braço funcionar através do POO do C++.

A maior dificuldade do grupo foi implementar as classes com as funcionalidades do braço, a partir de um código base. Porém, devido a semelhança da linguagem do Arduino com C++, essa dificuldade se tornou mais transparente na sua resolução.

A experiência de execução do trabalho aqui visto foi bastante inspiradora e divertida, pois foi possível ver a implementação direta de conceitos vistos em sala no projeto, que está bastante relacionado as aplicações do curso (robótica, neste caso), o que torna todo o desenvolvimento mais interessante. Certamente houve agruras e empecilhos no decorrer do desenvolvimento do braço, mas são mínimos comparados à satisfação de vê-lo concluído.