Documentação do Projeto Yoann

Sumário

[**Peça 1, versão 1(21/10/2022) - Protótipo Inicial**](#_7wt51qhl33ho) **1**

[**Peça 1, versão 2(26/10/2022) - aprimoramento da versão 1**](#_yqvdrswru4ye) **4**

[**Peça 1, versão 3(29/11/2022) - atualização do código do movimento**](#_n435d7mbiv0c) **5**

[**Peça 1, versão 4(30/11/2022) - alteração: utilização do motor nema 17**](#_le1rvxnqbl9z) **8**

[**Peça 1, versão 5(01/12/2022) - aprimoramento da versão 4**](#_17ek4ibg9gua) **11**

[**Peça 1, versão 6(05/12/2022) - incorporação da Ponte H**](#_qxpr5gitxa0p) **13**

[**Peça 1, versão 7(07/12/2022) - incorporação do potênciometro**](#_lq6z5u7c6wff) **15**

[**Peça 1, versão 8(07/12/2022) - mudança de nome para estruturaUmDefinitivo.ino**](#_oij1abtg3jlr) **18**

[**Peça 2, versão 1(26/10/2022) - Protótipo Inicial**](#_n1uqvtosz8zc) **22**

[**Peça 2, versão 2(06/12/2022) - fusão dos códigos em um único arquivo**](#_eaznsgpchf16) **27**

# 

### Peça 1, versão 1(21/10/2022) - Protótipo Inicial

Componentes utilizados:

* Dois arduinos uno
* Dois cabos azuis
* 3 mg90s
* 2 conectores de bateria
* 6 jumpers
* 2 baterias

Foi o primeiro contato com o projeto, o cliente levou nenhum equipamento para utilizarmos, logo fizemos com o que tínhamos lá e criamos um código universal para que qualquer SERVOMOTOR, realizasse o movimento solicitado pelo cliente, o movimento era um pendulo de 60 graus, conseguimos fazer o que nos foi solicitado, foi aprovado pelo cliente que levou a peça para o Rio funcionando, logo finalizamos a peça.

Arquivo de software desenvolvido: sketch\_oct21a.ino

#include <VarSpeedServo.h>

VarSpeedServo Um;

void setup() {

Um.attach(3); //O pino a ser conectado à placa arduíno

int IN3 = 8;

int IN4 = 9;

}

void loop() { //instruções que serão executadas continuamente

for(float i = 60; i<120; i+= 0.1){ //este ciclo for é responsável por subir 60 graus

Um.slowmove(i,30); //move cada grau a uma velocidade 30 milissegundos

delay(1); //delay necessário para o envio de informações para a placa arduíno

if(i==120){

digitalWrite(IN3, HIGH);

digitalWrite(IN4, HIGH);

delay(1);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

}

}

for(float j = 120; j > 0; j-= 0.1){ //este ciclo for é responsável por descer 120 graus

Um.slowmove(j,30); //move cada grau a uma velocidade 30 milissegundos

delay(1); //delay necessário para o envio de informações para a placa arduíno

if(i==0){

digitalWrite(IN3, HIGH);

digitalWrite(IN4, HIGH);

delay(1);

digitalWrite(IN3, LOW);

digitalWrite(IN4, HIGH);

}

}

0

for(float k = 0; k < 60; k+= 0.1){ //este ciclo for é responsável por subir 60 graus

Um.slowmove(k,30); //move cada grau a uma velocidade 30 milissegundos

delay(1); //delay necessário para o envio de informações para a placa arduíno

}

}

### Peça 1, versão 2(26/10/2022) - aprimoramento da versão 1

O cliente reclamou da força do Servo motor utilizado, algo que nós já tínhamos avisado e solicitado peças compatíveis com o que nos foi solicitado. Nesse dia tentamos modificar um pouco o código 1, para melhorar o movimento, mas não houve melhora.

Arquivo do software desenvolvido: codigo\_atualizado.ino

#include <VarSpeedServo.h>

VarSpeedServo Um;

int IN3 = 8;

int IN4 = 9;

void setup() {

Um.attach(3);

pinMode (IN3,OUTPUT);

pinMode (IN4,OUTPUT); //O pino a ser conectado à placa arduíno

}

void loop() { //instruções que serão executadas continuamente

digitalWrite(IN3, HIGH); //esta segunda combinação faz com que o motor de passo gire para uma determinada orientação

digitalWrite(IN4, HIGH);

delay(1000);

digitalWrite(IN3, HIGH); //esta segunda combinação faz com que o motor de passo gire para uma determinada orientação

digitalWrite(IN4, LOW);

delay(1000);

digitalWrite(IN3, HIGH); //esta segunda combinação faz com que o motor de passo gire para uma determinada orientação

digitalWrite(IN4, HIGH);

delay(1000);

digitalWrite(IN3, LOW); //esta quarta combinação faz com que o motor de passo gire para a orientação contrária

digitalWrite(IN4, HIGH);

delay(1000);

}

### Peça 1, versão 3(29/11/2022) - atualização do código do movimento

O cliente solicitou que fosse utilizado um motor de para-brisa, no lugar do SERVOMOTOR, indo contra as recomendações e sendo então necessário o desenvolvimento de um novo código para movimentar e controlar o motor solicitado. Após a montagem, percebemos que o motor aquecia muito e recomendamos utilizar outro motor

Uma vez que após a compra de equipamentos solicitados, o cliente pediu uma atualização no código da peça 1, ele insatisfeito com os resultados ao colocar muito mais peso do que os SERVOMOTORES aguentavam resolveu utilizar um outro tipo de motor, como dito acima realizamos o que foi solicitado.

Também foi incrementado no código dois clicks que mudam o sentido de rotação das peças, algo que só foi necessário posteriormente e então realizamos o serviço e utilizamos em todos os códigos a partir dessa versão.

Arquivo do software desenvolvido: Yoan.ino

int b1 = 2;

int b2 = 3;

int IN1 = 8;

int IN2 = 9;

int IN3 = 10;

int IN4 = 11;

int estado = 0;

void setup() {

pinMode(b1, INPUT\_PULLUP);

pinMode(b2, INPUT\_PULLUP);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(IN3, OUTPUT);

pinMode(IN4, OUTPUT);

Serial.begin(9600);

}

void loop() {

if (digitalRead(b1) == LOW) {

estado = 0;

}

if (digitalRead(b2) == LOW) {

estado = 1;

}

if (estado == 0) {

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

Serial.println("Horario");

}

if (estado == 1) {

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

digitalWrite(IN3, LOW);

digitalWrite(IN4, HIGH);

Serial.println("Anti-horario");

}

}

### Peça 1, versão 4(30/11/2022) - alteração: utilização do motor nema 17

O cliente levou o novo motor a ser utilizado desta vez não era o motor de para-brisa nem um SERVOMOTOR, e sim um MOTORDEPASSO chamado NEMA17, novamente foi necessário desenvolver um novo código e uma nova forma de montar os componentes, ao terminarmos de ajustar o código e o movimento o cliente percebeu que o motor não tinha o torque necessário, solicitou a criação de uma caixa de redução, algo que realizamos e testamos no dia 01/12/2022, onde o cliente aprovou a peça, mas continuava insatisfeito. (A caixa de redução foi feita e ainda não foi paga e está sendo utilizada constantemente pelo cliente).

Arquivo de Software desenvolvido: Nema\_17\_Stepper\_Motor\_With\_Arduino.ino

#define EN 8

//Direction pin

#define X\_DIR 5

//Step pin

#define X\_STP 2

#define X\_LIM 9

#define Y\_LIM 10

int kauan = 0;

#include <Servo.h>

Servo servo;

const byte servoPin = A3; // coolant enable

byte pos = 0;

//A498

int delayTime = 25;

int stps = 800;

void step(byte dirPin, byte stepperPin, int steps) {

digitalWrite(dirPin, kauan);

//delay(100);

digitalWrite(stepperPin, HIGH);

delayMicroseconds(delayTime);

digitalWrite(stepperPin, LOW);

delayMicroseconds(delayTime);

}

void setup() {

pinMode(X\_DIR, OUTPUT);

pinMode(X\_STP, OUTPUT);

pinMode(Y\_LIM, INPUT\_PULLUP);

pinMode(EN, OUTPUT);

digitalWrite(EN, LOW);

Serial.begin(9600);

pinMode(X\_LIM, INPUT\_PULLUP);

servo.attach(servoPin);

}

void loop() {

if (digitalRead(X\_LIM) == LOW) {

kauan = 0;

Serial.println(kauan);

}

if (digitalRead(Y\_LIM) == LOW) {

kauan = 1;

Serial.println(kauan);

}

step(X\_DIR, X\_STP, stps);

}

### Peça 1, versão 5(01/12/2022) - aprimoramento da versão 4

Nesse dia o cliente voltou a ideia do motor de para-brisa, desta vez utilizando uma ponte H, para melhor controlar o motor de passo. Nesse dia foi solicitado à contratada que fizesse a montagem sem o Arduino o que não seria possível, fizemos toda a montagem, mas estávamos com dificuldades em ajustar a velocidade, o cliente por sua vez disse não ter problemas, aprovou a peça da forma em que estava, demos então como finalizada a peça.

Arquivo de Software desenvolvido: Nema17\_2.0.ino

#define EN 8

#define X\_DIR 5

#define X\_STP 2

#define X\_LIM 9

#define Y\_LIM 10

int DIR = 0;

void step(byte dirPin, byte stepperPin) {

digitalWrite(dirPin, DIR);

digitalWrite(stepperPin, HIGH);

delayMicroseconds(1500);

digitalWrite(stepperPin, LOW);

delayMicroseconds(1500);

}

void setup() {

pinMode(X\_DIR, OUTPUT);

pinMode(X\_STP, OUTPUT);

pinMode(Y\_LIM, INPUT\_PULLUP);

pinMode(EN, OUTPUT);

digitalWrite(EN, LOW);

Serial.begin(9600);

pinMode(X\_LIM, INPUT\_PULLUP);

}

void loop() {

if (digitalRead(X\_LIM) == LOW) {

DIR = 0;

Serial.println(DIR);

}

if (digitalRead(Y\_LIM) == LOW) {

DIR = 1;

Serial.println(DIR);

}

step(X\_DIR, X\_STP);

}

### Peça 1, versão 6(05/12/2022) - incorporação da Ponte H

Desde a criação da versão 5, avisamos sobre a velocidade o cliente identificou esse problema, pois entrega da peça e pediu para corrigirmos, o que foi feito com o incremento de um potenciômetro e a criação de mais um código.

Software desenvolvido: PonteH.ino

// botões fim de curso

int b1 = 2;

int b2 = 3;

// motor

int IN1 = 8;

int IN2 = 9;

// sentido de rotação

int estado = 0;

// função sensido de rotação

void mudarEstado() {

if (digitalRead(b1) == LOW) {

estado = 0;

Serial.println(estado);

}

if (digitalRead(b2) == LOW) {

estado = 1;

Serial.println(estado);

}

}

// função ativar motor

void girarMotor() {

if (estado == 0) {

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

}

if (estado == 1) {

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

}

}

void setup() {

pinMode(b1, INPUT\_PULLUP);

pinMode(b2, INPUT\_PULLUP);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

Serial.begin(9600);

}

void loop() {

mudarEstado();

girarMotor();

delay(100);

}

### Peça 1, versão 7(07/12/2022) - incorporação do potênciometro

Software Desenvolvido: PonteH\_com\_potenciometro(PWM).ino

// botões

int b1 = 2;

int b2 = 3;

// motor

int IN1 = 8;

int IN2 = 9;

// potenciometro

int pot = A1;

// porta PWM \*\* EDITAR \*\*

int PWM = 3;

// variável da velocidade

int vel = 0;

// sentido de rotação

int estado = 0;

// função que verifica o sentido de rotação

void mudarEstado() {

if (digitalRead(b1) == LOW) {

estado = 0;

Serial.println(estado);

}

if (digitalRead(b2) == LOW) {

estado = 1;

Serial.println(estado);

}

}

// função que faz a leitura da velocidade pelo potenciometro

void velocidade() {

vel = analogRead(pot);

}

// função para girar o motor na velocidade desejada

void girarMotor() {

if (estado == 0) {

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

analogWrite(PWM,vel/4);

}

if (estado == 1) {

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

analogWrite(PWM,vel/4);

}

}

void setup() {

pinMode(b1, INPUT\_PULLUP);

pinMode(b2, INPUT\_PULLUP);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

Serial.begin(9600);

}

void loop() {

velocidade();

mudarEstado();

girarMotor();

delay(100);

}

### Peça 1, versão 8(07/12/2022) - mudança de nome para estruturaUmDefinitivo.ino

Software Desenvolvido: estruturaUmDefinitivo.ino

// ===================================================================================== \\

// PEÇA 1 \\

// Ligações:

// Botões: GND para o primeiro pino / digitais 2 e 3 para o pino central de cada botão

// PonteH: ENA ligado ao digital 5 / IN1 ligado ao digital 7 / IN2 ligado ao digital 6 / Jumper no pino regulador

// Potenciômetro: Pino 1 ligado ao GND / Pino central ligado ao analógico A1 / Pino 3 ligado ao 5V

// ===================================================================================== \\

// botões

int b1 = 2;

int b2 = 3;

// motor

int IN1 = 6;

int IN2 = 7;

// potenciometro

int pot = A1;

// porta PWM

int PWM = 5;

// variável da velocidade

int vel = 0;

// sentido de rotação

int estado = 0;

// função que verifica o sentido de rotação

void mudarEstado() {

if (digitalRead(b1) == LOW) {

estado = 0;

}

if (digitalRead(b2) == LOW) {

estado = 1;

}

}

// função que faz a leitura da velocidade pelo potenciometro

void velocidade() {

vel = analogRead(pot);

}

// função para girar o motor na velocidade desejada

void girarMotor() {

if (estado == 0) {

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

analogWrite(PWM, vel / 4);

Serial.print("Direção: ");

Serial.println(estado);

Serial.print("Velocidade: ");

Serial.println(vel / 4);

Serial.println();

}

if (estado == 1) {

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

analogWrite(PWM, vel / 4);

Serial.print("Direção: ");

Serial.println(estado);

Serial.print("Velocidade: ");

Serial.println(vel / 4);

Serial.println();

}

}

void setup() {

pinMode(b1, INPUT\_PULLUP);

pinMode(b2, INPUT\_PULLUP);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(PWM, OUTPUT);

Serial.begin(9600);

Serial.println("Iniciado");

Serial.println();

}

void loop() {

velocidade();

mudarEstado();

girarMotor();

}

### Peça 2, versão 1(26/10/2022) - Protótipo Inicial

Nesse mesmo dia, foi solicitado a criação de uma peça que utilizaria um SERVOMOTOR e recomendamos um MOTORDEPASSSO, algo que não tínhamos em mãos, essa peça foi qualificada como complexa, pois tínhamos que fazer uma integração entre dois códigos para dois movimentos, nesse dia tivemos que desenvolver um código para nos apresentar os movimentos do BLENDER, no dia entregamos ao cliente o código do SERVOMOTOR funcionando de forma perfeita, e um código para funcionamento de um MOTORDC, apenas para que ele apresentasse para os produtores, ainda nos comprometemos a atualizar e integrar os códigos quando nos entregasse um MOTORDEPASSO, o que foi feito posteriormente.

Arquivos de Software desenvolvidos: dois.ino e AnimacaoPenduloFuncionando.ino

dois.ino:

#include <VarSpeedServo.h>

#include <AccelStepper.h>

VarSpeedServo Um;

int IN3 = 8;

int IN4 = 9;

void setup() {

Um.attach(3);

pinMode (IN3,OUTPUT);

pinMode (IN4,OUTPUT); //O pino a ser conectado à placa arduíno

}

void loop() { //instruções que serão executadas continuamente

digitalWrite(IN3, HIGH); //esta segunda combinação faz com que o motor de passo gire para uma determinada orientação

digitalWrite(IN4, HIGH);

delay(1);

for(float i = 60; i<120; i+= 0.1){ //este ciclo for é responsável por subir 60 graus

Um.slowmove(i,30); //move cada grau a uma velocidade 30 milissegundos

delay(1); //delay necessário para o envio de informações para a placa arduíno

}

digitalWrite(IN3, HIGH); //esta segunda combinação faz com que o motor de passo gire para uma determinada orientação

digitalWrite(IN4, LOW);

delay(1);

for(float j = 120; j > 0; j-= 0.1){ //este ciclo for é responsável por descer 120 graus

Um.slowmove(j,30); //move cada grau a uma velocidade 30 milissegundos

delay(1); //delay necessário para o envio de informações para a placa arduíno

}

digitalWrite(IN3, HIGH); //esta segunda combinação faz com que o motor de passo gire para uma determinada orientação

digitalWrite(IN4, HIGH);

delay(1);

digitalWrite(IN3, LOW); //esta quarta combinação faz com que o motor de passo gire para a orientação contrária

digitalWrite(IN4, HIGH);

delay(1);

for(float k = 0; k < 60; k+= 0.1){ //este ciclo for é responsável por subir 60 graus

Um.slowmove(k,30); //move cada grau a uma velocidade 30 milissegundos

delay(1); //delay necessário para o envio de informações para a placa arduíno

}

}

AnimacaoPenduloFuncionando.ino:

#include <Servo.h>

#define FPS 24

#define FRAMES 250

#define PWM\_PIN 3

/\*

Servo Position value Animation

FPS: 24

Frames: 250

Armature: Armature

\*/

const float Bone[250] PROGMEM = {90.0, 90.0, 90.010, 90.037, 90.080, 90.142, 90.223, 90.323, 90.446, 90.590, 90.759, 90.954, 91.176, 91.428, 91.711, 92.030, 92.385, 92.782, 93.223, 93.714, 94.259, 94.865, 95.539, 96.290, 97.128, 98.068, 99.125, 100.32, 101.68, 103.24, 105.04, 107.14, 109.61, 112.49, 115.71, 119.01, 121.97, 124.34, 126.13, 127.43, 128.38, 129.04, 129.50, 129.79, 129.95, 129.99, 129.92, 129.70, 129.32, 128.80, 128.12, 127.30, 126.34, 125.24, 124.01, 122.65, 121.18, 119.59, 117.90, 116.13, 114.27, 112.36, 110.39, 108.39, 106.38, 104.36, 102.35, 100.38, 98.457, 96.589, 94.797, 93.093, 91.490, 90.0, 88.591, 87.228, 85.911, 84.640, 83.416, 82.238, 81.108, 80.025, 78.988, 78.000, 77.058, 76.164, 75.316, 74.515, 73.760, 73.051, 72.386, 71.766, 71.190, 70.655, 70.162, 69.709, 69.295, 68.917, 68.575, 68.267, 67.991, 67.745, 67.527, 67.334, 67.165, 67.016, 66.886, 66.772, 66.671, 66.581, 66.500, 66.565, 66.715, 66.939, 67.228, 67.575, 67.976, 68.425, 68.920, 69.456, 70.031, 70.643, 71.291, 71.971, 72.683, 73.425, 74.197, 74.997, 75.825, 76.679, 77.559, 78.464, 79.395, 80.350, 81.329, 82.331, 83.357, 84.407, 85.480, 86.575, 87.694, 88.835, 90.0, 91.172, 92.329, 93.453, 94.528, 95.537, 96.463, 97.296, 98.023, 98.638, 99.138, 99.522, 99.790, 99.948, 99.999, 99.987, 99.949, 99.884, 99.793, 99.674, 99.527, 99.351, 99.146, 98.912, 98.649, 98.357, 98.036, 97.686, 97.310, 96.907, 96.478, 96.026, 95.551, 95.056, 94.541, 94.010, 93.465, 92.906, 92.338, 91.760, 91.177, 90.590, 90.0, 89.414, 88.840, 88.279, 87.732, 87.199, 86.683, 86.184, 85.704, 85.243, 84.805, 84.388, 83.996, 83.628, 83.285, 82.968, 82.677, 82.412, 82.173, 81.960, 81.772, 81.608, 81.467, 81.348, 81.249, 81.170, 81.109, 81.065, 81.036, 81.020, 81.017, 81.026, 81.052, 81.101, 81.172, 81.262, 81.370, 81.496, 81.637, 81.793, 81.963, 82.147, 82.343, 82.552, 82.772, 83.003, 83.245, 83.497, 83.759, 84.031, 84.313, 84.604, 84.904, 85.213, 85.531, 85.859, 86.195, 86.541, 86.897, 87.263, 87.640, 88.029, 88.433, 88.854, 89.303, 89.821, };

const float frameDurationMillis = 1000 / FPS;

const float animationDurationMillis = FRAMES \* frameDurationMillis;

Servo myservo;

long startMillis = millis();

void setup() {

myservo.attach(PWM\_PIN);

}

void loop() {

long currentMillis = millis();

long positionMillis = currentMillis - startMillis;

if (positionMillis >= animationDurationMillis) {

startMillis = currentMillis;

} else {

long frame = floor(positionMillis / frameDurationMillis);

float positionValue = pgm\_read\_float\_near(Bone + frame);

//int positionValue = pgm\_read\_word\_near(Bone + frame);

myservo.write(positionValue);

}

}

Arquivo de programa desenvolvido para capturar os ângulos gerados, usando Python: getAnglesBlenderPython.py

Arquivo gerado pelo programa getAnglesBlenderPython: angles.txt

angles.txt:

<https://drive.google.com/file/d/19NVDKmwqMsxsBhzzM1yk4ut8EK-BqjaE/view?usp=share_link>

### Peça 2, versão 2(06/12/2022) - fusão dos códigos em um único arquivo

Arquivo Software desenvolvido: fusaoPeca2.ino

#include <Servo.h>

//fusão dos códigos dos dois motores da peça 2

#define FPS 24

#define FRAMES 250

#define PWM\_PINA 3

#define PWN\_PINB 5

/\*

Servo Position value Animation

FPS: 24

Frames: 250

Armature: Armature

\*/

const float Bone[250] PROGMEM = {90.0, 90.0, 90.010, 90.037, 90.080, 90.142, 90.223, 90.323, 90.446, 90.590, 90.759, 90.954, 91.176, 91.428, 91.711, 92.030, 92.385, 92.782, 93.223, 93.714, 94.259, 94.865, 95.539, 96.290, 97.128, 98.068, 99.125, 100.32, 101.68, 103.24, 105.04, 107.14, 109.61, 112.49, 115.71, 119.01, 121.97, 124.34, 126.13, 127.43, 128.38, 129.04, 129.50, 129.79, 129.95, 129.99, 129.92, 129.70, 129.32, 128.80, 128.12, 127.30, 126.34, 125.24, 124.01, 122.65, 121.18, 119.59, 117.90, 116.13, 114.27, 112.36, 110.39, 108.39, 106.38, 104.36, 102.35, 100.38, 98.457, 96.589, 94.797, 93.093, 91.490, 90.0, 88.591, 87.228, 85.911, 84.640, 83.416, 82.238, 81.108, 80.025, 78.988, 78.000, 77.058, 76.164, 75.316, 74.515, 73.760, 73.051, 72.386, 71.766, 71.190, 70.655, 70.162, 69.709, 69.295, 68.917, 68.575, 68.267, 67.991, 67.745, 67.527, 67.334, 67.165, 67.016, 66.886, 66.772, 66.671, 66.581, 66.500, 66.565, 66.715, 66.939, 67.228, 67.575, 67.976, 68.425, 68.920, 69.456, 70.031, 70.643, 71.291, 71.971, 72.683, 73.425, 74.197, 74.997, 75.825, 76.679, 77.559, 78.464, 79.395, 80.350, 81.329, 82.331, 83.357, 84.407, 85.480, 86.575, 87.694, 88.835, 90.0, 91.172, 92.329, 93.453, 94.528, 95.537, 96.463, 97.296, 98.023, 98.638, 99.138, 99.522, 99.790, 99.948, 99.999, 99.987, 99.949, 99.884, 99.793, 99.674, 99.527, 99.351, 99.146, 98.912, 98.649, 98.357, 98.036, 97.686, 97.310, 96.907, 96.478, 96.026, 95.551, 95.056, 94.541, 94.010, 93.465, 92.906, 92.338, 91.760, 91.177, 90.590, 90.0, 89.414, 88.840, 88.279, 87.732, 87.199, 86.683, 86.184, 85.704, 85.243, 84.805, 84.388, 83.996, 83.628, 83.285, 82.968, 82.677, 82.412, 82.173, 81.960, 81.772, 81.608, 81.467, 81.348, 81.249, 81.170, 81.109, 81.065, 81.036, 81.020, 81.017, 81.026, 81.052, 81.101, 81.172, 81.262, 81.370, 81.496, 81.637, 81.793, 81.963, 82.147, 82.343, 82.552, 82.772, 83.003, 83.245, 83.497, 83.759, 84.031, 84.313, 84.604, 84.904, 85.213, 85.531, 85.859, 86.195, 86.541, 86.897, 87.263, 87.640, 88.029, 88.433, 88.854, 89.303, 89.821, };

const float frameDurationMillis = 1000 / FPS;

const float animationDurationMillis = FRAMES \* frameDurationMillis;

Servo myservo; //declaração de um objeto para o motor do pêndulo

Servo myservo2; //declaração de um objeto para o motor de movimento circular

long startMillis = millis();

void setup() {

myservo.attach(PWM\_PINA); //o motor de pendulo é conectado à porta 3

myservo2.attach(PWN\_PINB); //o motor de pendulo é conectado à porta 5

}

void loop() {

long currentMillis = millis();

long positionMillis = currentMillis - startMillis;

if (positionMillis >= animationDurationMillis) {

startMillis = currentMillis;

} else{

long frame = floor(positionMillis / frameDurationMillis);

float positionValue = pgm\_read\_float\_near(Bone + frame);

//int positionValue = pgm\_read\_word\_near(Bone + frame);

myservo.write(positionValue);

myservo2.write(positionValue);

}

}