Servo Taura BL35 e BL25

Conhecendo o Servo Taura BL35

Poderoso e versátil. Conheça o Servomotor Taura da stemOS.

- Motor Brushless
- Alto torque e eficiência
- Projetado para substituir aplicações que tradicionalmente usariam motores DC
- Pino de feedback analógico
- Engrenagens internas de aço de alta resistência
- Corpo usinado em alumínio sólido
- Compatível com todo ecosistema REV, goBILDA e Andymark
- Modo Continuo ou Angular, com limites e velocidades configuráveis
- Legal para FTC e FRC

O servo Taura BL35 da stemOS é um servo com motor Brushless que oferece alto torque e performance de qualidade por um preço acessível. Contando com engrenagens internas de aço de alta resistência, corpo usinado de alumínio sólido, e saída de feedback analógico.

Esse servo conta com tudo que sua equipe precisa para mover mecanismos simples ou complexos. Com o uso do Programador de Servo Taura, você pode configurar os parâmetros e o modo de rotação do servo, preparando o servo perfeitamente para o que sua equipe precisa.

O servomotor Taura BL35 tem seu potencial totalmente desbloqueado com o Servo Hub REV, e pode ser programado através do Programador de Servo Taura.

Especificações técnicas:

Velocidade sem carga:

- 4,8V – 0,140s/60° (Conectado ao Control/Expansion Hub)
- 6,0V – 0,115s/60° (Conectado ao Servo Hub)

Torque de parada:

- 4,8V – 28kg-cm (Conectado ao Control/Expansion Hub)
- 6,0V – 34kg-cm (Conectado ao Servo Hub)

Quer saber mais? Confira a Ficha Técnica Completa.

Downloads:

- Software de programação
- Firmware para modo Angular
- Firmware para modo Contínuo

Programando o Servo Taura BL35 e BL25

Instalação do Software de Programação

Requisitos de Sistema

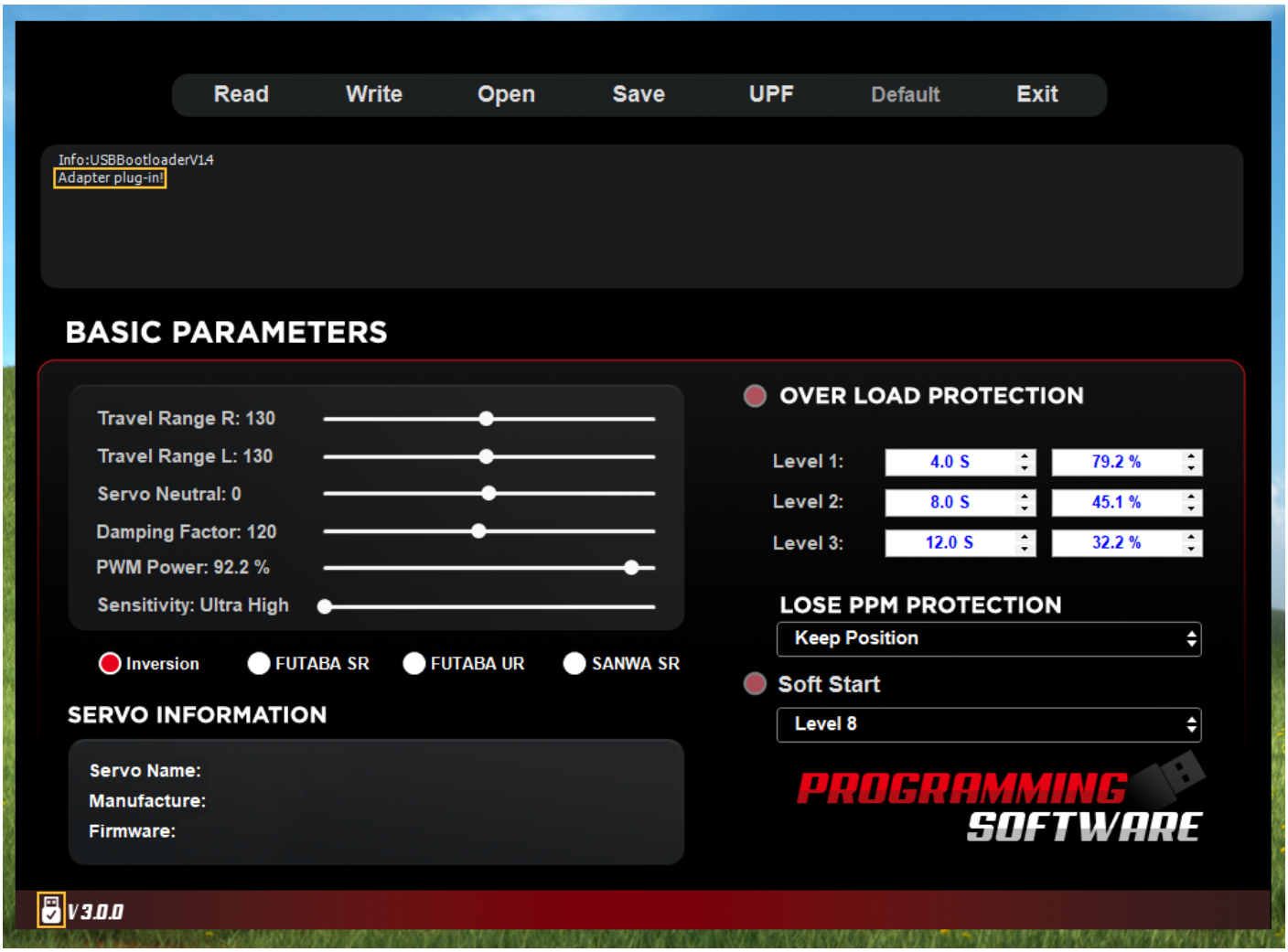
1. Tensão de Operação: USB (5V/500mA)
2. Sistema Operacional Windows XP/Vista/7/8/10 (32/64bit)

Instalação do Software

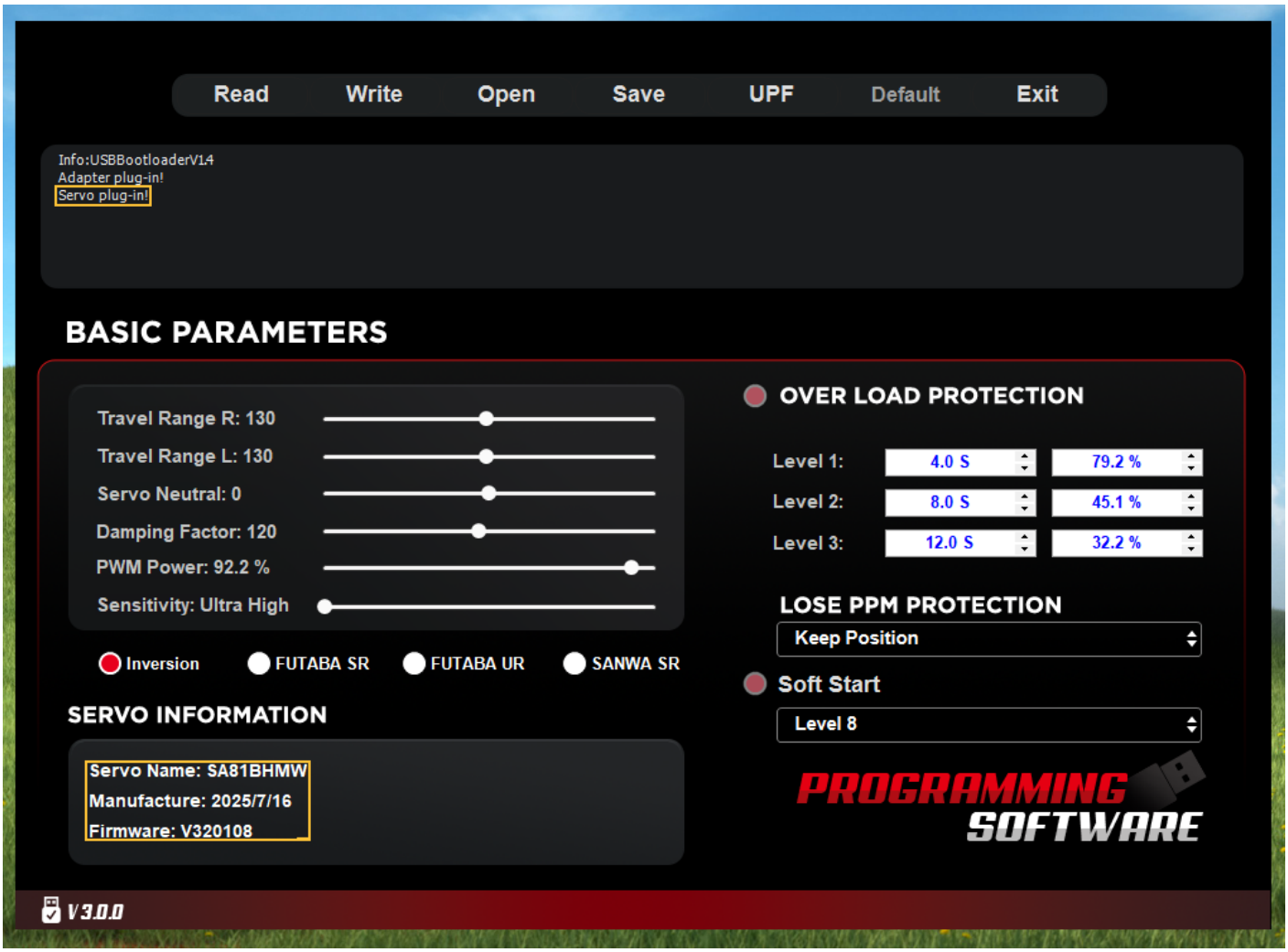
1. Instale o software de programação
2. Extraia o arquivo *.zip* recém instalado
3. Conecte o programador de servo em uma porta USB de seu computador, e aguarde a instalação dos drivers.

> Esse processo ocorre automaticamente e leva de 10 a 20 segundos. Você receberá uma notificação no fim do processo.

4. Abra o *software* e leia a janela superior para confirmar que o programador está devidamente conectado. Caso contrário, reconecte o dispositivo.



5. Conecte o servo no programador.
    - O software deve automaticamente reconhecer o servo e exibir seus parâmetros atuais, junto de suas informações.

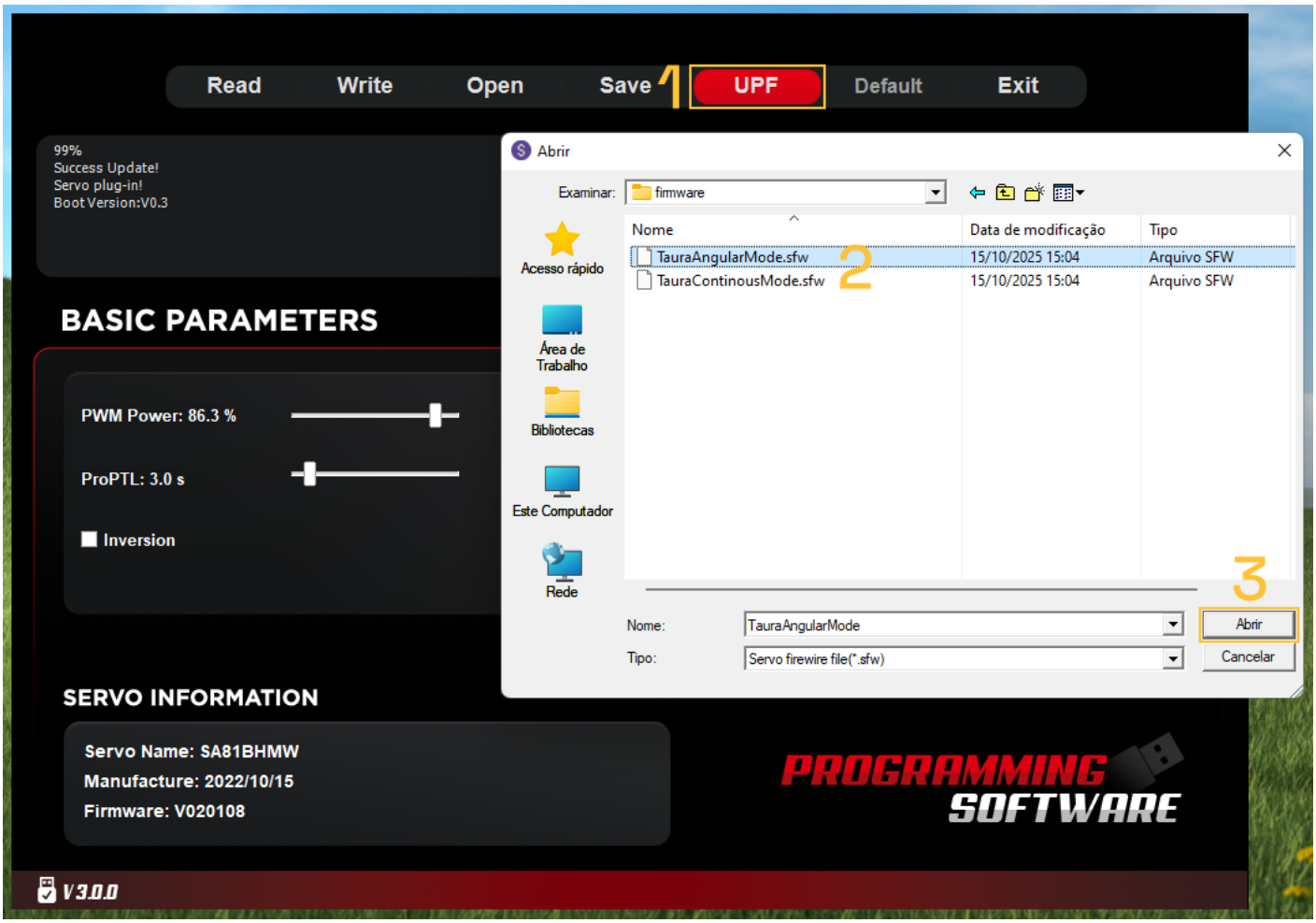

Read
Write
Open
Save
UPF
Default
Exit
Info:USBBootloaderV1.4
Adapter plug-in!
Servo plug-in!
BASIC PARAMETERS
Travel Range R: 130
Travel Range L: 130
Servo Neutral: 0
Damping Factor: 120
PWM Power: 92.2 %
Sensitivity: Ultra High
Inversion
FUTABA SR
FUTABA UR
SANWA SR
SERVO INFORMATION
Servo Name: SA81BHMW
Manufacture: 2025/7/16
Firmware: V320108
OVER LOAD PROTECTION
Level 1:
4.0 S
79.2 %
Level 2:
8.0 S
45.1 %
Level 3:
12.0 S
32.2 %
LOSE PPM PROTECTION
Keep Position
Soft Start
Level 8
PROGRAMMING
SOFTWARE
V3.0.0

Alternando entre Modo Angular e Rotação Contínua

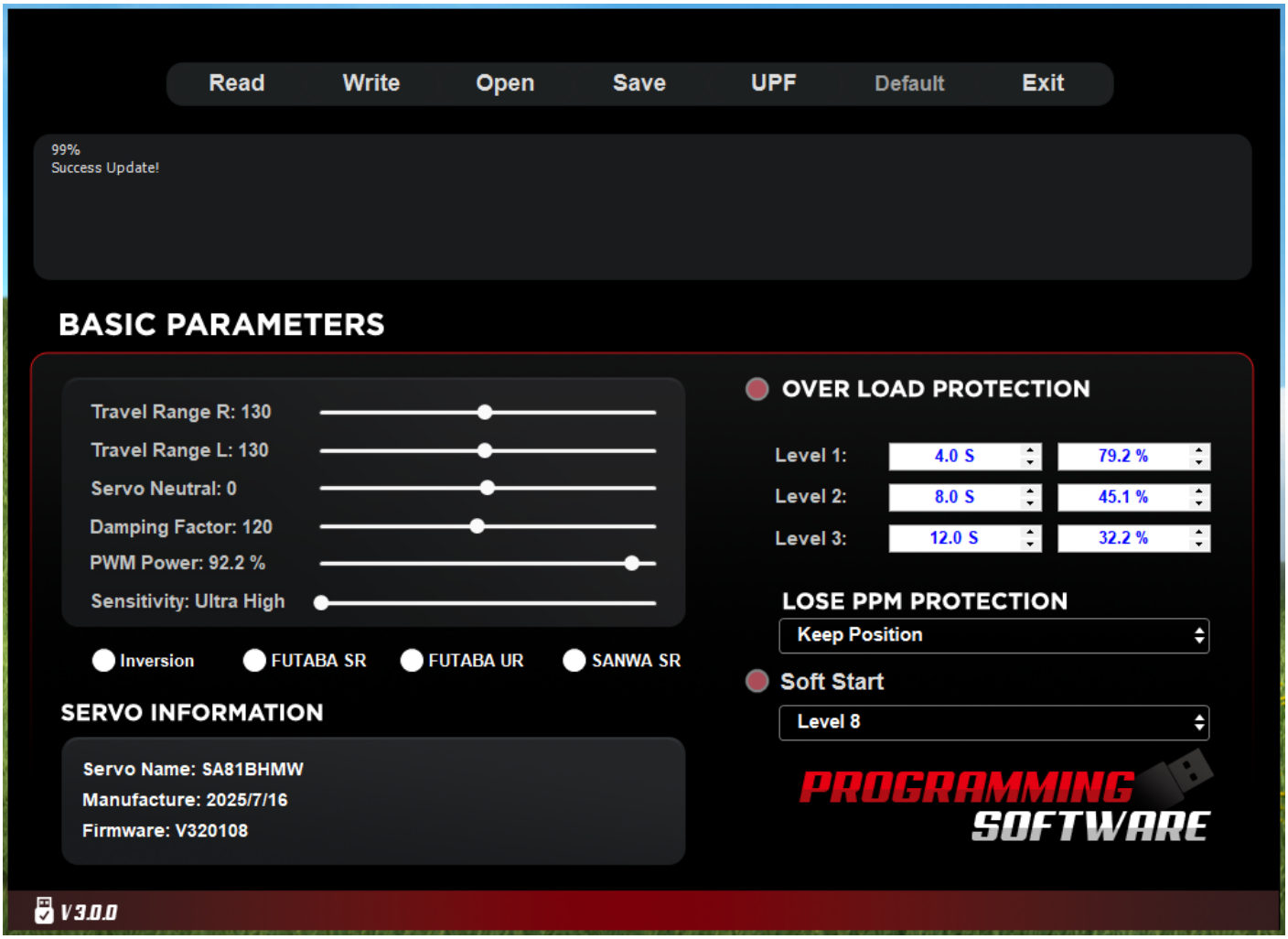
Instalação e Atualização do Firmware do Servo

> Veja a página anterior para entender como conectar o servo Taura.

1. Conecte o Taura e o programador ao computador;
2. Selecione "UPF" na barra superior, selecione o firmware que deseja (modo angular ou rotação contínua)
   - Confirme sua seleção.



3. Durante a instalação, a janela superior lhe informará do progresso da instalação.



4. Após a finalização da instalação, os novos parâmetros do servo serão automaticamente exibidos na interface do software.

Configurando o Taura

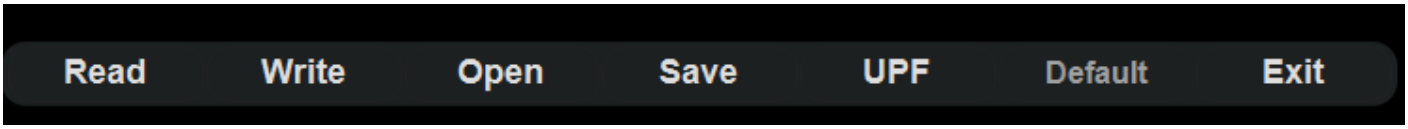
> Este guia resume os parâmetros ajustáveis no software do programador do servo Taura e como cada ajuste afeta o acionador.

Antes de começar

- Conecte o programador ao PC e, em seguida, ligue o servo ao cabo do programador.
- Ao plugar o servo, o software lê automaticamente os parâmetros atuais.

> O servo deve ser conectado com o cabo de sinal no mesmo sentido do LED do programador.

Botões e menus do software



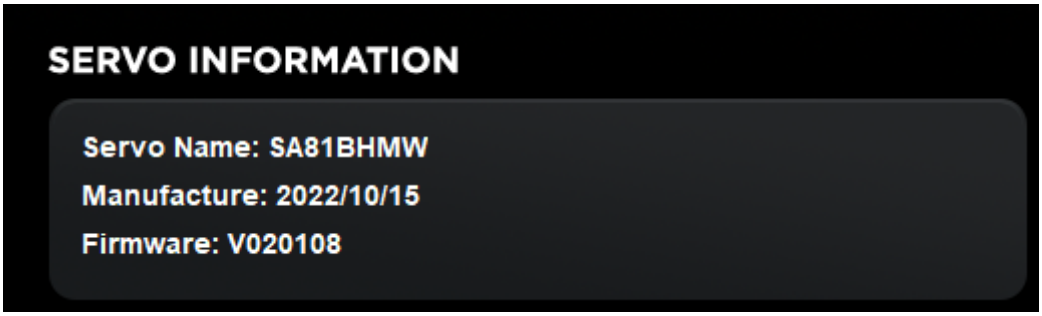
- **Read**
  - Lê os parâmetros atuais do servo e **preenche** a tela com os valores efetivamente gravados no dispositivo.
- **Write**
  - **Grava** no servo todas as alterações feitas na tela. Use após ajustar parâmetros.
- **Save**
  - **Salva** um arquivo com os parâmetros atualmente exibidos no software, para reutilizar em outras unidades ou versionar configurações.

> Coloque a extensão .svo no nome do arquivo para que funcione corretamente.

- **Open**
  - **Carrega** um perfil salvo previamente para a tela. Após abrir, clique em **Write** para aplicar esses valores ao servo conectado.
- **Default** / **Factory Default**
  - Não disponível

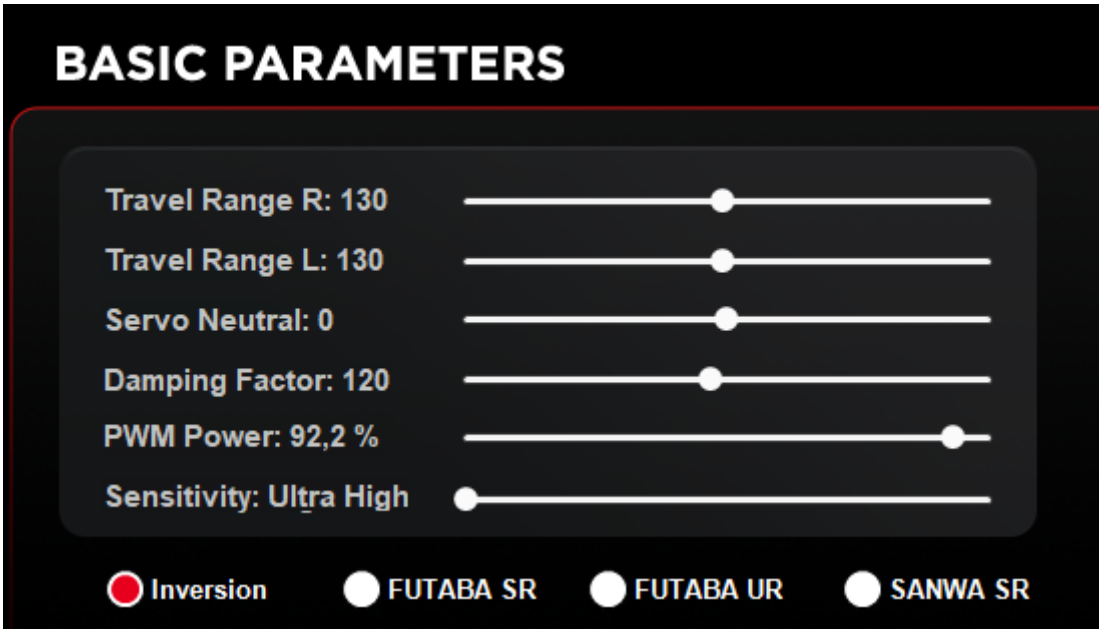
- **UPF / Firmware Update**
  - Abre o utilitário de **atualização de firmware**. Escolha o **arquivo correto** para o seu modelo e para o **modo** desejado (Angular/Continuous).

> Não desconecte o servo durante o processo.



- **Model/Info** (varia conforme versão)
  Mostra **informações do servo**: modelo, fabricante e versão de firmware.

Parâmetros básicos



Travel Range (Ângulo de Percurso)

Define o **ângulos máximo e mínimo** do servo.

O valor mostrado é um **indicador** para ajuste relativo (não é um ângulo absoluto garantido para todos os modelos).

- Aumentar o valor amplia o ângulo; reduzir limita a rotação.

>

> 255 significa o limite máximo do servo, NÃO é o valor em graus do limite.

Servo Neutral (Ponto Neutro)

Ajusta o **centro** do servo (posição quando o sinal é neutro).

PWM Power (Potência PWM)

Controla a **o torque e velocidade** do servo.

> Lembre-se que quanto maior esse valor maior é o consumo de corrente elétrica do servo.

Damping Factor (Fator de Amortecimento)

Define o **amortecimento** do controle interno. Amortecimento mais alto ajuda a reduzir oscilações após chegar ao alvo; muito alto pode deixar respostas mais lentas.

Sensitivity / Dead Band (Sensibilidade / Faixa Morta)

Ajusta a **faixa morta** do erro de posição. É dividido nos seguintes níveis:

- Low
- Medium
- High
- Ultra High

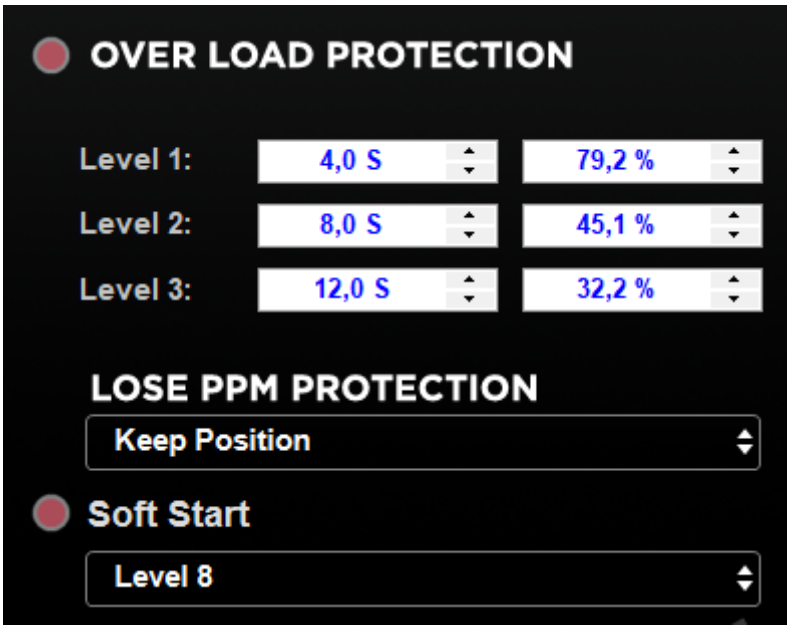
Modos de protocolo/alta velocidade

- FUTABA SR
- SANWA SR
- FUTABA UR

> Os modos acima não são suportados pelo servo Taura.

- *Inversion*
  - Define o **sentido de rotação** (normal/invertido). Use para adequar o sentido ao seu mecanismo sem precisar reverter no Control Hub.

Proteções



Signal Loss Protect (Perda de Sinal)

Configura o comportamento em **perda de sinal**:

- Manter posição;
- Ir a posição neutra (1500us);
- Desabilitar força.

Over Load Protect (Proteção contra Sobrecarga)

Ao habilitar, você pode definir **três níveis** (Level 1/2/3), cada um com:

- **Tempo de disparo** (quando a proteção entra após detectar carga excessiva).
- **Potência de saída** permitida **após** o disparo (quanto o servo “alivia”).

> Em geral, o controle à esquerda ajusta o tempo até acionar; à direita ajusta a potência/força restante após a proteção. Isso reduz aquecimento e dano quando o servo fica travado.

Soft Start (Partida Suave)

Faz o servo **iniciar suavemente** após energizar: ao ligar, ele vai **devagar** para a posição correspondente ao sinal atual, reduzindo trancos no sistema.

- É dividido em 11 níveis.

> Quanto maior o nível, mais rapidamente ele chega a posição inicial após a energização.

> Esse controle é independente do PWM Power.

Modo contínuo

PWM Power

Define a velocidade e torque do servo no modo contínuo.

> Use valores acima de 10,2%.

ProPTL

Configura o tempo até a proteção de bloqueio ser ativada e o servo parar.

Inversion

Define o sentido de giro do servo.

Programação de exemplo - Java

O servo taura pode ser programado normalmente como qualquer servo do ecosistema FTC. Nesta página você encontra um código de exemplo que:

- Movimenta o servo em diferentes posições;
- Retorna a posição (0.0 até 1.0) pelo sensor analógico presente.

Instale os códigos no seguinte endereço: Github

> É preciso baixar o arquivo TauraServo.java também.

```
package org.firstinspires.ftc.teamcode;

import com.qualcomm.robotcore.eventloop.opmode.LinearOpMode;
import com.qualcomm.robotcore.eventloop.opmode.TeleOp;
import com.qualcomm.robotcore.hardware.AnalogInput;
import com.qualcomm.robotcore.hardware.Servo;

@TeleOp(name="Taura Move", group="Linear OpMode")
public class ExampleTaura extends LinearOpMode {

    @Override
    public void runOpMode() throws InterruptedException {
        Servo servo = hardwareMap.get(Servo.class, "servo");
        AnalogInput potentiometer = hardwareMap.get(AnalogInput.class, "potentiometer");

        TauraServo tauraServo = new TauraServo(servo);

        tauraServo.setAnalogFeedbackSensor(potentiometer);

        tauraServo.setPosition(0.0);
```

```
        waitForStart();
        if (isStopRequested()) return;

        while(opModeIsActive()) {

            if(gamepad1.a)
                tauraServo.setPosition(1.0);
            else if(gamepad1.b)
                tauraServo.setPosition(0.5);
            else
                tauraServo.setPosition(0.0);

            telemetry.addData("Pos comandada", "%.3f", tauraServo.getPosition());
            telemetry.addData("Pos lida", "%.3f", tauraServo.getUniversalPosition());
            telemetry.addData("Pos graus absoluto", "%.3f", tauraServo.getRawPositionInDegrees());
            telemetry.addData("Pos graus incremental", "%.3f", tauraServo.getIncrementalPositionInDegrees());

            telemetry.update();
        }
    }

}
```

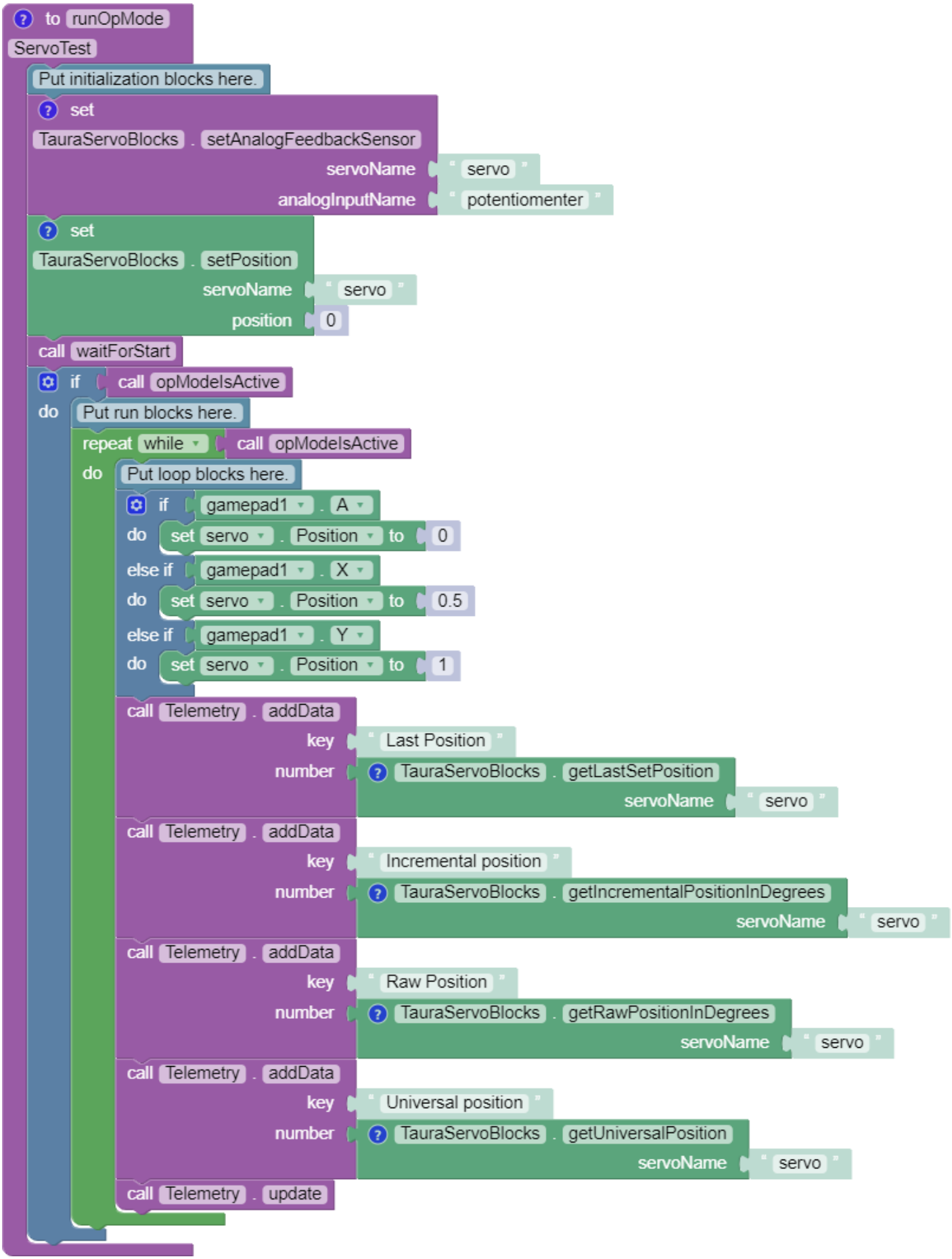
Programação de exemplo - Blocos

> Essa seção está em desenvolvimento.

Instalação

1. Baixe os seguintes arquivos:
   - Taura Blocks
   - Driver Taura
2. Adicione os arquivos no OnBot Java do seu Control Hub.
3. Construa o programa.
4. Os blocos agora devem aparecer no ambiente de programação em blocos.

Exemplo


to runOpMode
ServoTest
Put initialization blocks here.
set
TauraServoBlocks . setAnalogFeedbackSensor
servoName " servo "
analogInputName " potentiomenter "
set
TauraServoBlocks . setPosition
servoName " servo "
position 0
call waitForStart
if call opModeIsActive
do Put run blocks here.
repeat while call opModeIsActive
do Put loop blocks here.
if gamepad1 . A
do set servo . Position to 0
else if gamepad1 . X
do set servo . Position to 0.5
else if gamepad1 . Y
do set servo . Position to 1
call Telemetry . addData
key " Last Position "
number TauraServoBlocks . getLastSetPosition
servoName " servo "
call Telemetry . addData
key " Incremental position "
number TauraServoBlocks . getIncrementalPositionInDegrees
servoName " servo "
call Telemetry . addData
key " Raw Position "
number TauraServoBlocks . getRawPositionInDegrees
servoName " servo "
call Telemetry . addData
key " Universal position "
number TauraServoBlocks . getUniversalPosition
servoName " servo "
call Telemetry . update