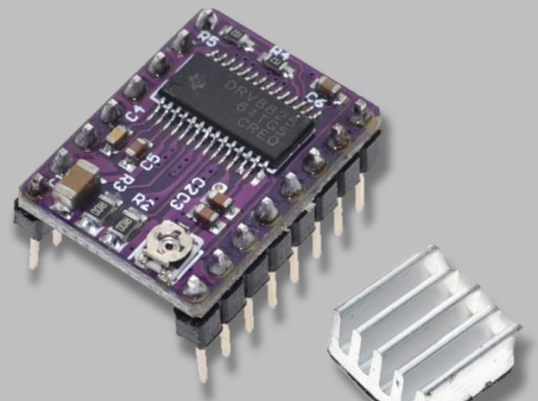
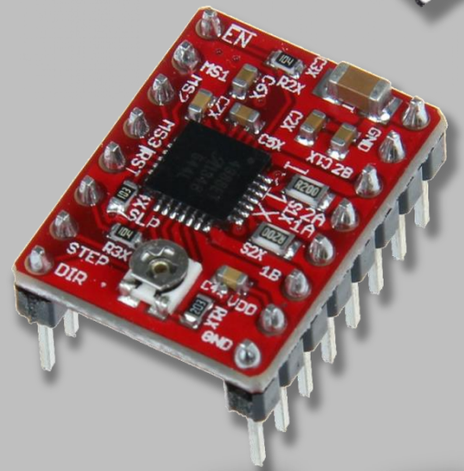
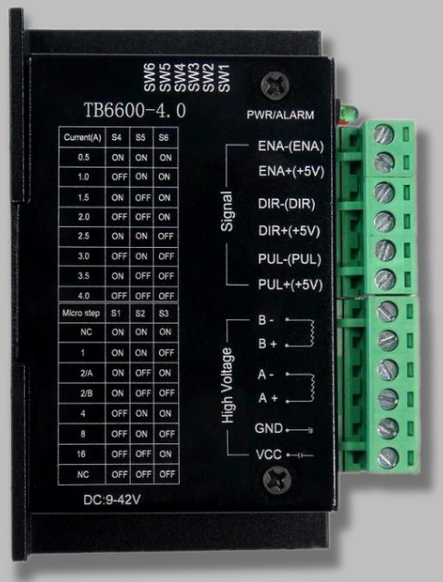
Biblioteca para Driver de Motor de Passo







Por Fernando Koyanagi

Intenção dessa aula

Mostrar uma biblioteca para driver de motor de passo completa com chaves de fim de curso, movimento dos motores com aceleração e micro passo.

Biblioteca StepDriver

Para que serve:

- A biblioteca abrange os três tipos de drivers mais comuns: A4988, DRV8825 e TB6600.
- Configura os pinos dos drivers, possibilitando realizar o reset, colocar em modo Sleep, bem como ativar e desativar as saídas do motor atuando no pino Enable.
- Configura as entradas dos pinos de micro passo do driver.
- Configura as chaves de fim de curso e o nível de ativação das mesmas (alto ou baixo)
- Também conta com código de movimento do motor com aceleração contínua em mm/s², velocidade máxima em mm/s e velocidade mínima em mm/s.

Variáveis Globais

```
int RST;        // Porta reset do DRV8825
int SLP;        // Porta dormir (Sleep) DRV8825
int ENA;        // Porta ativa (Enable) DRV8825
int M0;         // Porta M0 do DRV8825
int M1;         // Porta M1 do DRV8825
int M2;         // Porta M2 do DRV8825
int DIR;        // Porta direção (Direction) do DRV8825
int STP;        // Porta passo(Step) do DRV8825
int SCF;        // Modo do passo
int SPR;        // Passos por volta
int EMAX;       // Fim de curso máximo
int EMIN;       // Fim de curso mínimo
double SPM;     // Passos por milímetro
bool LVL;       // Usado para verificar se os fins de curso são ativados em nível alto ou baixo
int ACC;        //Define a aceleração do motor em milímetros por segundo ao quadrado
double MIN_SPEED;   //Define a velocidade mínima do motor em milímetros por segundo
double MAX_SPEED;   //Define a velocidade máxima do motor em milímetros por segundo
```

Funções – Configuração dos pinos do driver

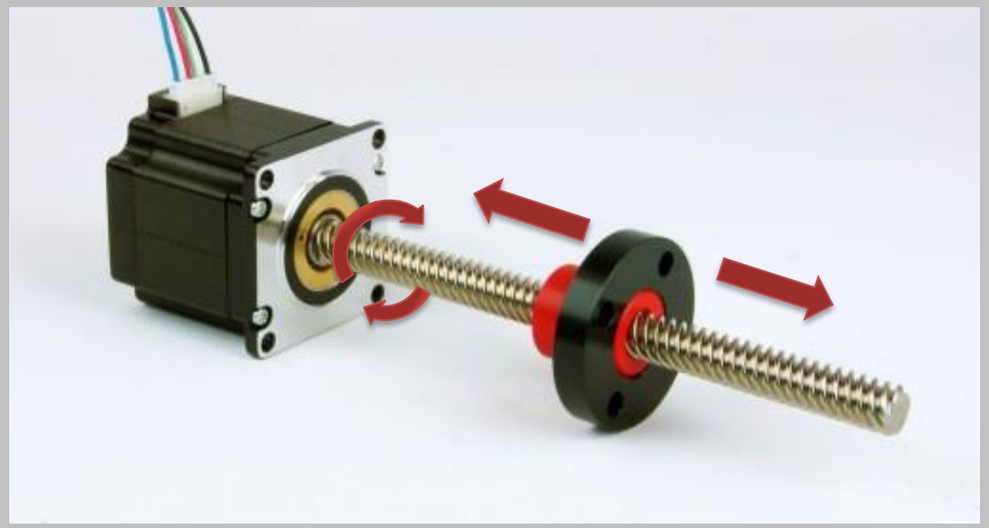
```
void DRV8825::pinConfig(int ena, int m0, int m1, int m2, int rst, int slp, int stp, int dir){
ENA = ena;          // Porta ativa (enable) DRV8825
M0 = m0;            // Porta M0 do DRV8825
M1 = m1;            // Porta M1 do DRV8825
M2 = m2;            // Porta M2 do DRV8825
RST = rst;          // Porta reset do DRV8825
SLP = slp;          // Porta dormir (sleep) DRV8825
STP = stp;          // Porta passo(step) do DRV8825
DIR = dir;          // Porta direção (direction) do DRV8825

//Define os pinos do arduino como saída
pinMode(RST, OUTPUT);
pinMode(SLP, OUTPUT);
pinMode(ENA, OUTPUT);
pinMode(M0, OUTPUT);
pinMode(M1, OUTPUT);
pinMode(M2, OUTPUT);
pinMode(DIR, OUTPUT);
pinMode(STP, OUTPUT);
}
```

Funções – Funções básicas do driver

```
void DRV8825::reset()
{
digitalWrite(RST, LOW);      // Realiza o reset do DRV8825
delay (1);                   // Atraso de 1 milissegundo
digitalWrite(RST, HIGH);     // Libera o reset do DRV8825
delay (10);                  // Atraso de 10 milissegundos
}

void DRV8825::enable(bool state)
{
    digitalWrite(ENA, !state);// Ativa e desativa o chip DRV8825
                              //(FALSE = ativa, TRUE = desativa)
    delay (10);               // Atraso de 10 milissegundos
}

void DRV8825::sleep(bool state)
{
    digitalWrite(SLP, !state);// Ativa e desativa o modo Sleep do chip DRV8825
                              //(FALSE = ativa, TRUE = desativa)
    delay (10);               // Atraso de 10 milissegundos
}
```

Funções – Configuração de passo do motor

```
//Configura a quantidade de passos por milímetro que o motor deve realizar
void DRV8825::stepPerMm(double steps)
{
    SPM = steps;
}

//Configura a quantidade de passos por volta que o motor deve realizar
void DRV8825::stepPerRound(int steps)
{
    SPR = steps;
}
```



Funções – Configuração do modo de passo do motor

```
void DRV8825::stepConfig(int s) //Configura o modo do passo
{
    SCF = s;

    switch (SCF){
        case 1: FULL();
        break;
        case 2: s1_2();
        break;
        case 4: s1_4();
        break;
        case 8: s1_8();
        break;
        case 16: s1_16();
        break;
        case 32: s1_32();
        break;
        default: FULL();
        break;
    }
}
```

```
// Configura modo Passo completo
void DRV8825::FULL() {
    digitalWrite(M0, LOW);
    digitalWrite(M1, LOW);
    digitalWrite(M2, LOW);
}
```

	M0	M1	M2
FULL	LOW	LOW	LOW
1/2	HIGH	LOW	LOW
1/4	LOW	HIGH	LOW
1/8	HIGH	HIGH	LOW
1/16	LOW	LOW	HIGH
1/32	HIGH	LOW	HIGH

Funções – Configuração do modo de passo do motor

```
void DRV8825::stepConfig(int s) //Configura o modo do passo
{
    SCF = s;

    switch (SCF){
        case 1: FULL();
        break;
        case 2: s1_2();
        break;
        case 4: s1_4();
        break;
        case 8: s1_8();
        break;
        case 16: s1_16();
        break;
        case 32: s1_32();
        break;
        default: FULL();
        break;
    }
}
```

```
// Configura modo Passo 1/2
void DRV8825::s1_2() {
    digitalWrite(M0, HIGH);
    digitalWrite(M1, LOW);
    digitalWrite(M2, LOW);
}
```

	M0	M1	M2
FULL	LOW	LOW	LOW
1/2	HIGH	LOW	LOW
1/4	LOW	HIGH	LOW
1/8	HIGH	HIGH	LOW
1/16	LOW	LOW	HIGH
1/32	HIGH	LOW	HIGH

Funções – Configuração do modo de passo do motor

```
void DRV8825::stepConfig(int s) //Configura o modo do passo
{
    SCF = s;

    switch (SCF){
        case 1: FULL();
        break;
        case 2: s1_2();
        break;
        case 4: s1_4();
        break;
        case 8: s1_8();
        break;
        case 16: s1_16();
        break;
        case 32: s1_32();
        break;
        default: FULL();
        break;
    }
}
```

```
// Configura modo Passo 1/4
void DRV8825::s1_4()
{
    digitalWrite(M0, LOW);
    digitalWrite(M1, HIGH);
    digitalWrite(M2, LOW);
}
```

	M0	M1	M2
FULL	LOW	LOW	LOW
1/2	HIGH	LOW	LOW
1/4	LOW	HIGH	LOW
1/8	HIGH	HIGH	LOW
1/16	LOW	LOW	HIGH
1/32	HIGH	LOW	HIGH

Funções – Configuração do modo de passo do motor

```
void DRV8825::stepConfig(int s) //Configura o modo do passo
{
    SCF = s;

    switch (SCF){
        case 1: FULL();
        break;
        case 2: s1_2();
        break;
        case 4: s1_4();
        break;
        case 8: s1_8();
        break;
        case 16: s1_16();
        break;
        case 32: s1_32();
        break;
        default: FULL();
        break;
    }
}
```

```
// Configura modo Passo 1/8
void DRV8825::s1_8()
{
    digitalWrite(M0, HIGH);
    digitalWrite(M1, HIGH);
    digitalWrite(M2, LOW);
}
```

	M0	M1	M2
FULL	LOW	LOW	LOW
1/2	HIGH	LOW	LOW
1/4	LOW	HIGH	LOW
1/8	HIGH	HIGH	LOW
1/16	LOW	LOW	HIGH
1/32	HIGH	LOW	HIGH

Funções – Configuração do modo de passo do motor

```
void DRV8825::stepConfig(int s) //Configura o modo do passo
{
    SCF = s;

    switch (SCF){
        case 1: FULL();
        break;
        case 2: s1_2();
        break;
        case 4: s1_4();
        break;
        case 8: s1_8();
        break;
        case 16: s1_16();
        break;
        case 32: s1_32();
        break;
        default: FULL();
        break;
    }
}
```

```
// Configura modo Passo 1/16
void DRV8825::s1_16()
{
    digitalWrite(M0, LOW);
    digitalWrite(M1, LOW);
    digitalWrite(M2, HIGH);
}
```

	M0	M1	M2
FULL	LOW	LOW	LOW
1/2	HIGH	LOW	LOW
1/4	LOW	HIGH	LOW
1/8	HIGH	HIGH	LOW
1/16	LOW	LOW	HIGH
1/32	HIGH	LOW	HIGH

Funções – Configuração do modo de passo do motor

```
void DRV8825::stepConfig(int s) //Configura o modo do passo
{
    SCF = s;

    switch (SCF){
        case 1: FULL();
        break;
        case 2: s1_2();
        break;
        case 4: s1_4();
        break;
        case 8: s1_8();
        break;
        case 16: s1_16();
        break;
        case 32: s1_32();
        break;
        default: FULL();
        break;
    }
}
```

```
// Configura modo Passo 1/32
void DRV8825::s1_32()
{
    digitalWrite(M0, HIGH);
    digitalWrite(M1, LOW);
    digitalWrite(M2, HIGH);
}
```

	M0	M1	M2
FULL	LOW	LOW	LOW
1/2	HIGH	LOW	LOW
1/4	LOW	HIGH	LOW
1/8	HIGH	HIGH	LOW
1/16	LOW	LOW	HIGH
1/32	HIGH	LOW	HIGH

Funções – Configuração das chaves de fim de curso

```
//Configura os pinos das chaves de fim de curso
void DRV8825::endstopConfig(int emin, int emax, bool level)
{
    EMAX = emax; //Fim de curso máximo
    EMIN = emin; //Fim de curso mínimo
    LVL = level; //Define se as chaves ativam
//em nível alto ou nível baixo
    pinMode(EMAX, INPUT);
    pinMode(EMIN, INPUT);
}
```

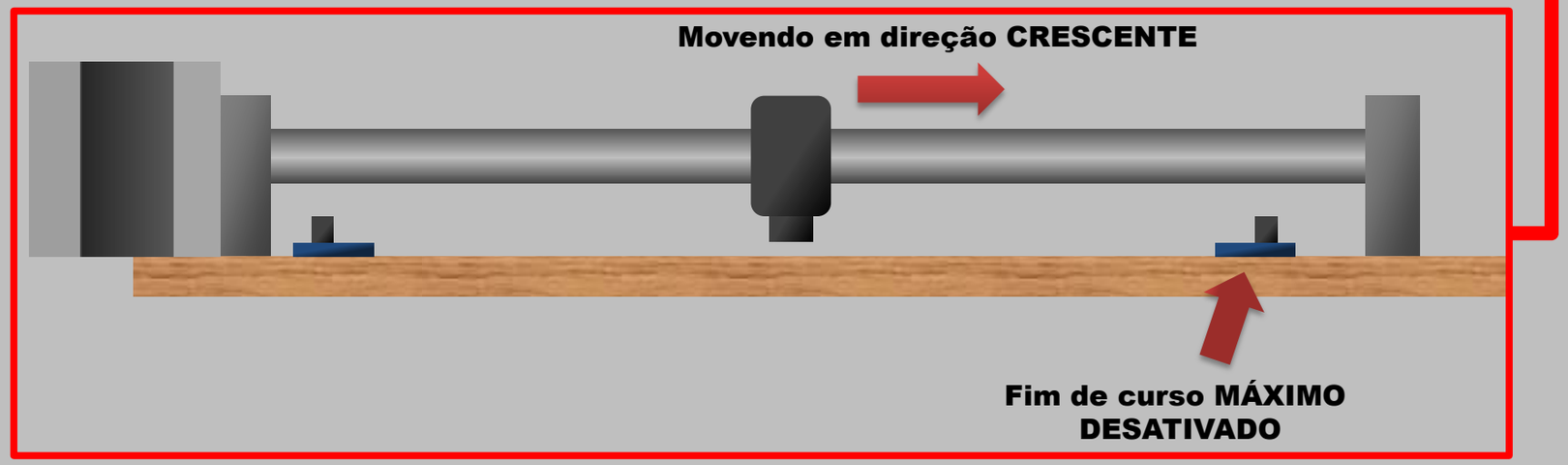
Funções – Leitura das chaves de fim de curso

```
//Lê o nível da chave de fim de curso
bool DRV8825::eRead(int pin){
    if(LVL&&digitalRead(pin))   //Se a chave ativa em nível ALTO
                                //E a leitura da chave esta ALTA

        return true;  //Retorna VERDADEIRO

    if(!LVL&&!digitalRead(pin)) //Se a chave ativa em nível BAIXO
                                //E a leitura da chave esta BAIXA

        return true;  //Retorna VERDADEIRO

    return false;  //Retorna FALSO
}
```

Nível	Leitura	Saída
0	0	True
0	1	False
1	0	False
1	1	True

Funções – Leitura das chaves de fim de curso

```
bool DRV8825::canGo(bool direction)
{
        if(direction && eRead(EMAX))//Se o motor está movendo em direção CRESCENTE
                                    //E a chave de fim de curso MÁXIMA está ativada

            return false;//Interrompe o movimento

        if(!direction && eRead(EMIN))//Se o motor está movendo em direção DECRESCENTE
                                     //E a chave de fim de curso MÍNIMA está ativada

            return false;//Interrompe o movimento

        return true;//Se não, Libera o movimento }
```

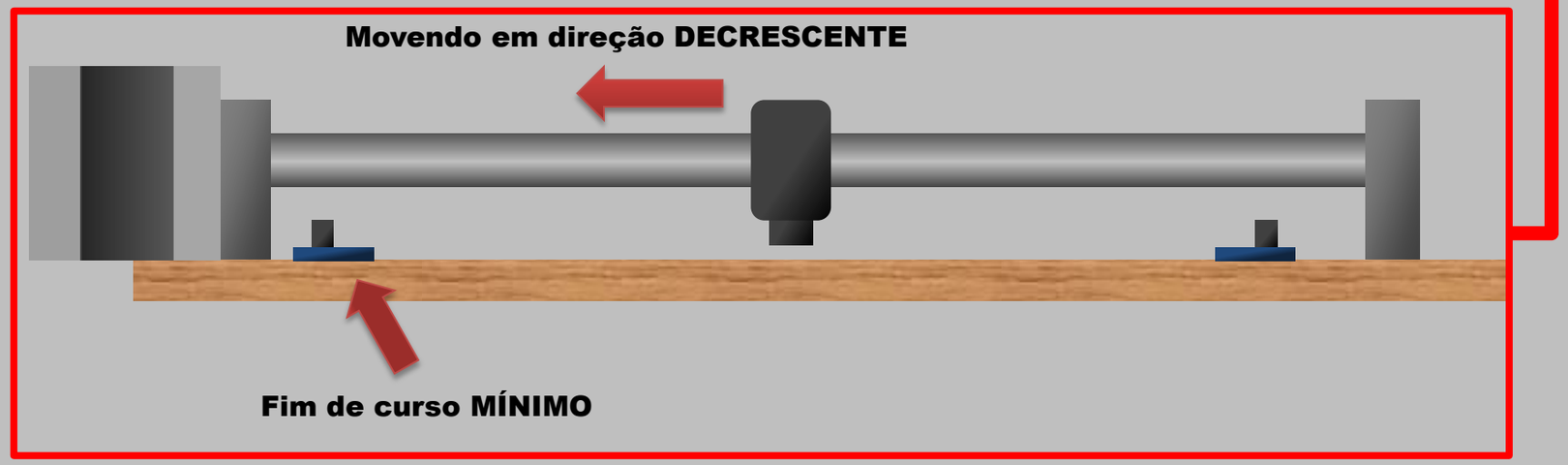


Funções – Leitura das chaves de fim de curso

```
bool DRV8825::canGo(bool direction)
{
        if(direction && eRead(EMAX))//Se o motor está movendo em direção CRESCENTE
                                    //E a chave de fim de curso MÁXIMA está ativada

            return false;//Interrompe o movimento

        if(!direction && eRead(EMIN))//Se o motor está movendo em direção DECRESCENTE
                                     //E a chave de fim de curso MÍNIMA está ativada

            return false;//Interrompe o movimento

        return true;//Se não, Libera o movimento }
```

Funções – Leitura das chaves de fim de curso

```
bool DRV8825::canGo(bool direction)
{
        if(direction && eRead(EMAX))//Se o motor está movendo em direção CRESCENTE
                                    //E a chave de fim de curso MÁXIMA está ativada

            return false;//Interrompe o movimento

        if(!direction && eRead(EMIN))//Se o motor está movendo em direção DECRESCENTE
                                     //E a chave de fim de curso MÍNIMA está ativada

            return false;//Interrompe o movimento

        return true;//Se não, Libera o movimento }
```



Funções – Leitura das chaves de fim de curso

```
bool DRV8825::canGo(bool direction)
{
        if(direction && eRead(EMAX))//Se o motor está movendo em direção CRESCENTE
                                    //E a chave de fim de curso MÁXIMA está ativada

            return false;//Interrompe o movimento

        if(!direction && eRead(EMIN))//Se o motor está movendo em direção DECRESCENTE
                                     //E a chave de fim de curso MÍNIMA está ativada

        return false;//Interrompe o movimento

        return true;//Se não, Libera o movimento }
```

Funções – Configuração de movimento

```
//Configuracao de movimento: Aceleração em mm/s^2, Velocidade máxima em mm/s, velocidade mínima em
mm/s
void DRV8825::motionConfig(double acceleration, double maxSpeed, double minSpeed)
{
    //Converte Aceleração de: mm/s^2 para passos/s^2
    ACC = acceleration * SPM;

    //Converte Velocidade máxima de: mm/s para passos/s
    MAX_SPEED = maxSpeed * SPM;

    //A velocidade mínima não pode ser maior que a velocidade máxima
    if (minSpeed > maxSpeed)

        //Converte Velocidade máxima de: mm/s para passos/s
        MIN_SPEED = maxSpeed * SPM;
    else
        //Converte Velocidade mínima de: mm/s para passos/s
        MIN_SPEED = minSpeed * SPM;
}
```

Funções – Função de Movimento

```
//Move um passo na direção solicitada em um período de microssegundos
void DRV8825::motorMove(int period, bool direction)
{
    //Configura o pino de direção do driver
    digitalWrite(DIR,direction);

    //Verifica se as chaves de fim de curso estão DESATIVADAS na direção solicitada
    if (canGo(direction))
    {
        digitalWrite(STP, HIGH);       //Altera o nível da porta STP do driver para nível alto
        delayMicroseconds(period / 2); //Aguarda meio período
        digitalWrite(STP, LOW);        //Altera o nível da porta STP do driver para nível baixo
        delayMicroseconds(period / 2); //Aguarda meio período
    }
    //Se as chaves de fim de curso estão ATIVADAS na direção solicitada
    else
        delayMicroseconds(period);     //Aguarda todo o período e não move o motor

}
```

Funções – Função de Movimento - variáveis

```
//Move o motor em uma determinada distância,
//Acelerando até a velocidade máxima e desacelerando até parar
void DRV8825::motorMoveTo(double distance, bool direction)
{
    //Período da velocidade mínima = 1/(velocidade mínima)
    double MIN_SPEED_PERIOD = 1.0 / MIN_SPEED;

    //Período da velocidade máxima = 1/(velocidade máxima)
    double MAX_SPEED_PERIOD = 1.0 / MAX_SPEED;

    //Período da velocidade de aceleração recebe inicialmente o período da velocidade mínima
    double ACC_SPEED_PERIOD = MIN_SPEED_PERIOD;

    //Converte a distância da trajetória de milímetros para passos
    unsigned long PATH_STEPS = distance * SPM;

    //Passos para parar recebe inicialmente a metade dos passos da trajetória
    long STEPS_TO_STOP = PATH_STEPS / 2;

    //Guarda a velocidade máxima alcançada pelo motor durante a aceleração
    double MAX_SPEED_REACHED;
    bool flag = true;
```

Funções – Função de Movimento - Aceleração

```
for (long i = 0; i < PATH_STEPS; i++) {
    //Enquanto estiver na primeira metade da trajetória
    if (i < (PATH_STEPS / 2)) {
        //Se o período da velocidade de aceleração for maior que a período da velocidade máxima
        if (ACC_SPEED_PERIOD > MAX_SPEED_PERIOD) {
```

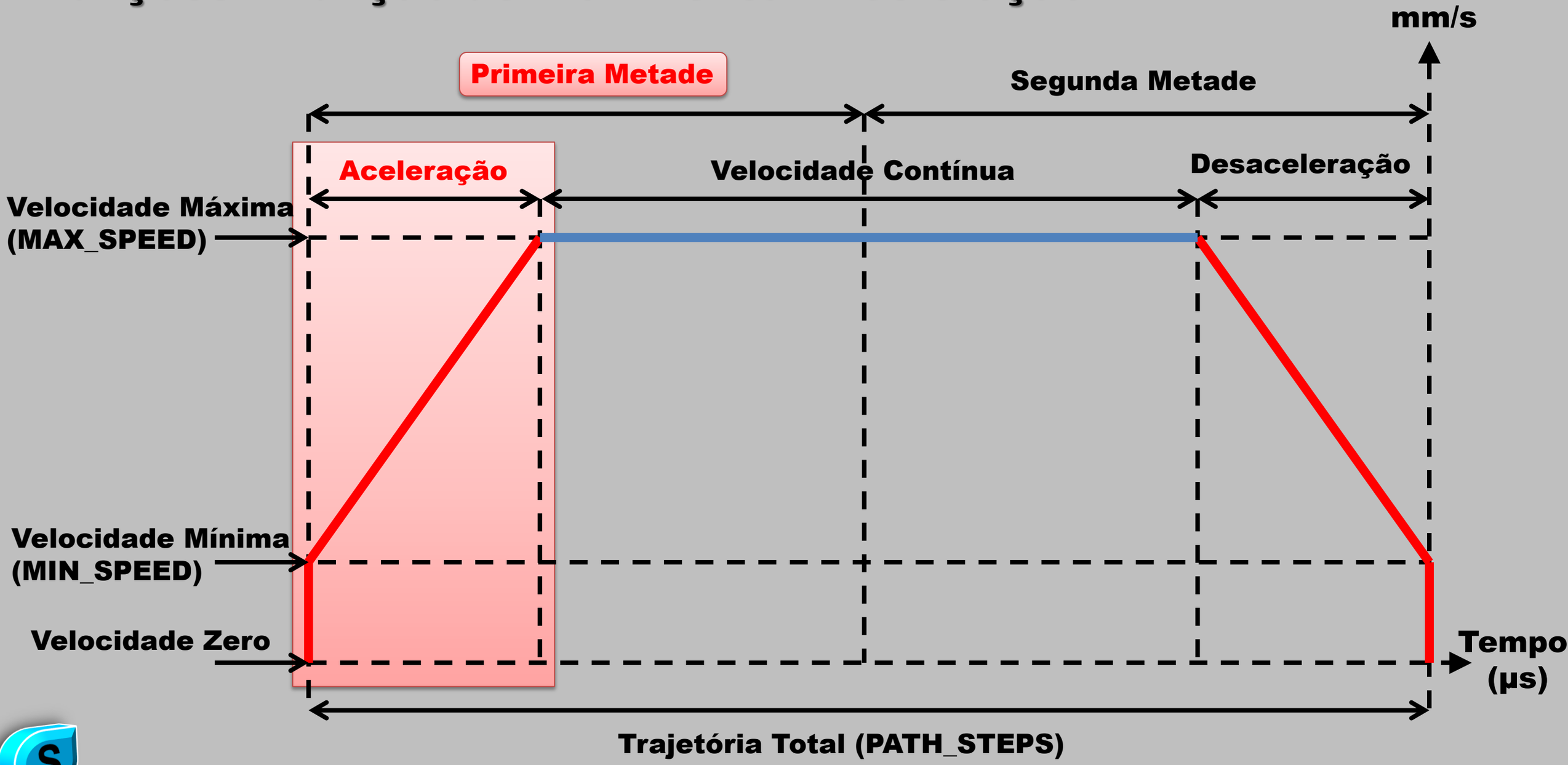
Equação de Torricelli Aceleração:

$$\frac{1}{\sqrt{V_{min}^2 + 2 * AC * i_a}}$$

V_{min} = Velocidade Mínima(mm/s)
AC = Aceleração(mm/s²)
i_a = Passo atual

```
            //Calcula o período de aceleração do motor:
            ACC_SPEED_PERIOD = 1.0 / sqrt((MIN_SPEED * MIN_SPEED) + (2.0 * ACC * i));

            //Converte o período de aceleração em segundos
            //para microssegundos e move o motor em um passo na direção solicitada
            motorMove(1000000 * ACC_SPEED_PERIOD, direction);

            //Converte e guarda o período máximo alcançado
            //para velocidade máxima alacançada: (1/periodo)/passos por mm
            MAX_SPEED_REACHED = (1 / ACC_SPEED_PERIOD) / SPM;
        }else{...
```

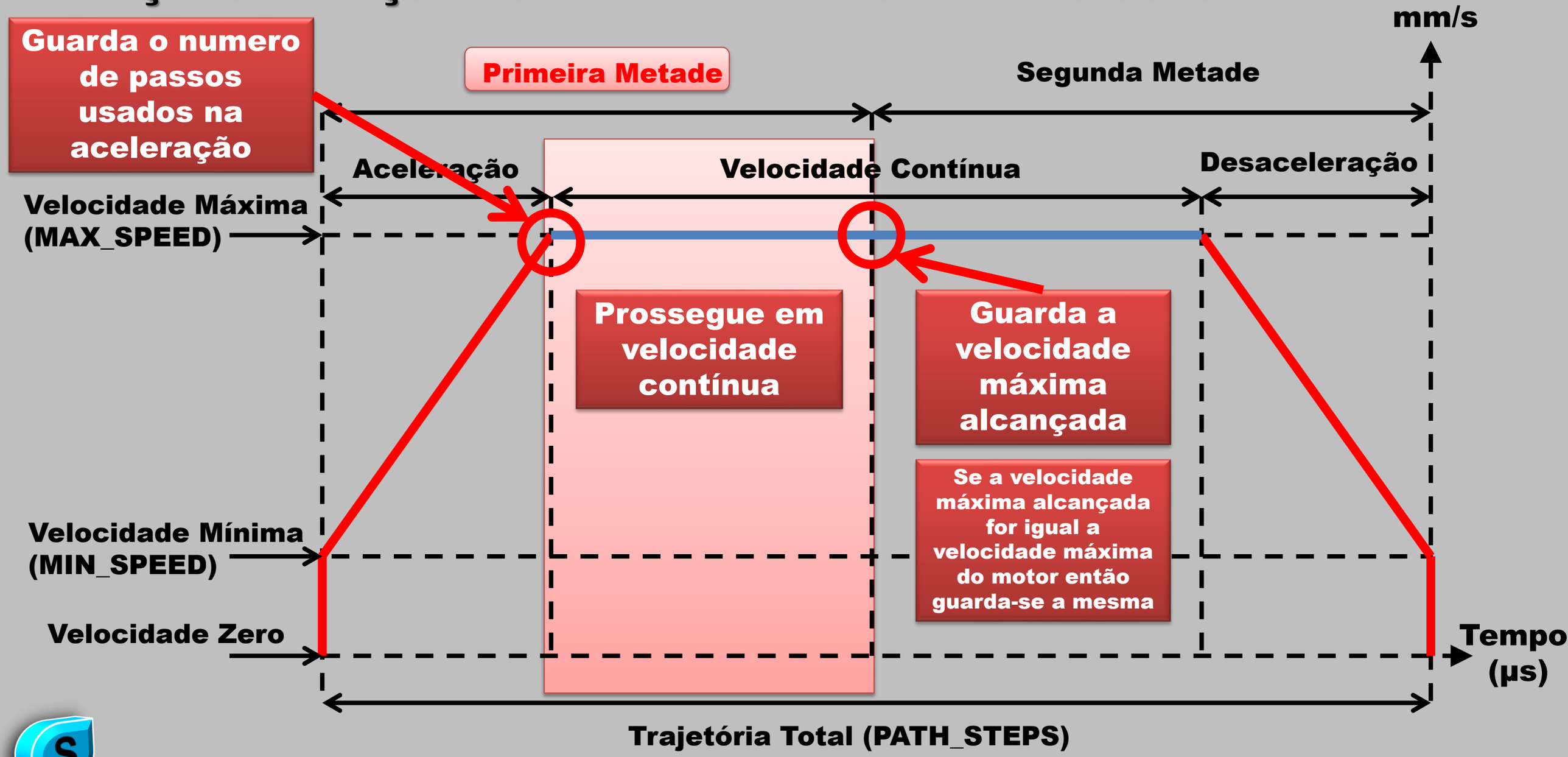
Funções – Função de Movimento - Aceleração



Funções – Função de Movimento – Velocidade Contínua

```
    ...
    else  //Se o periodo da velocidade de aceleração for
          //menor ou igual que a perído da velocidade máxima
    {
        if (flag) {  //Apenas um marcador para que execute a operação apenas uma vez

        //Guarda o numero de passos usados na aceleração
        STEPS_TO_STOP = i;

        //A velocidade máxima alcançada torna-se a velocidade máxima do motor
        MAX_SPEED_REACHED = MAX_SPEED;

        flag = false;
    }
    //Converte o período de velocidade máxima em segundos
    //para microssegundos e move o motor em um passo na direção solicitada
    motorMove(1000000 * MAX_SPEED_PERIOD, direction);
}//Fim da primeira metade
...
```

Funções – Função de Movimento – Velocidade Contínua



Funções – Função de Movimento - Desaceleração

```
        else//Enquanto estiver na segunda metade da trajetória
        {
            //Determina quando o motor deve desacelerar
            //(passos totais da trajetória menos o número de passos para parar)
            if (i > PATH_STEPS - STEPS_TO_STOP) {
```

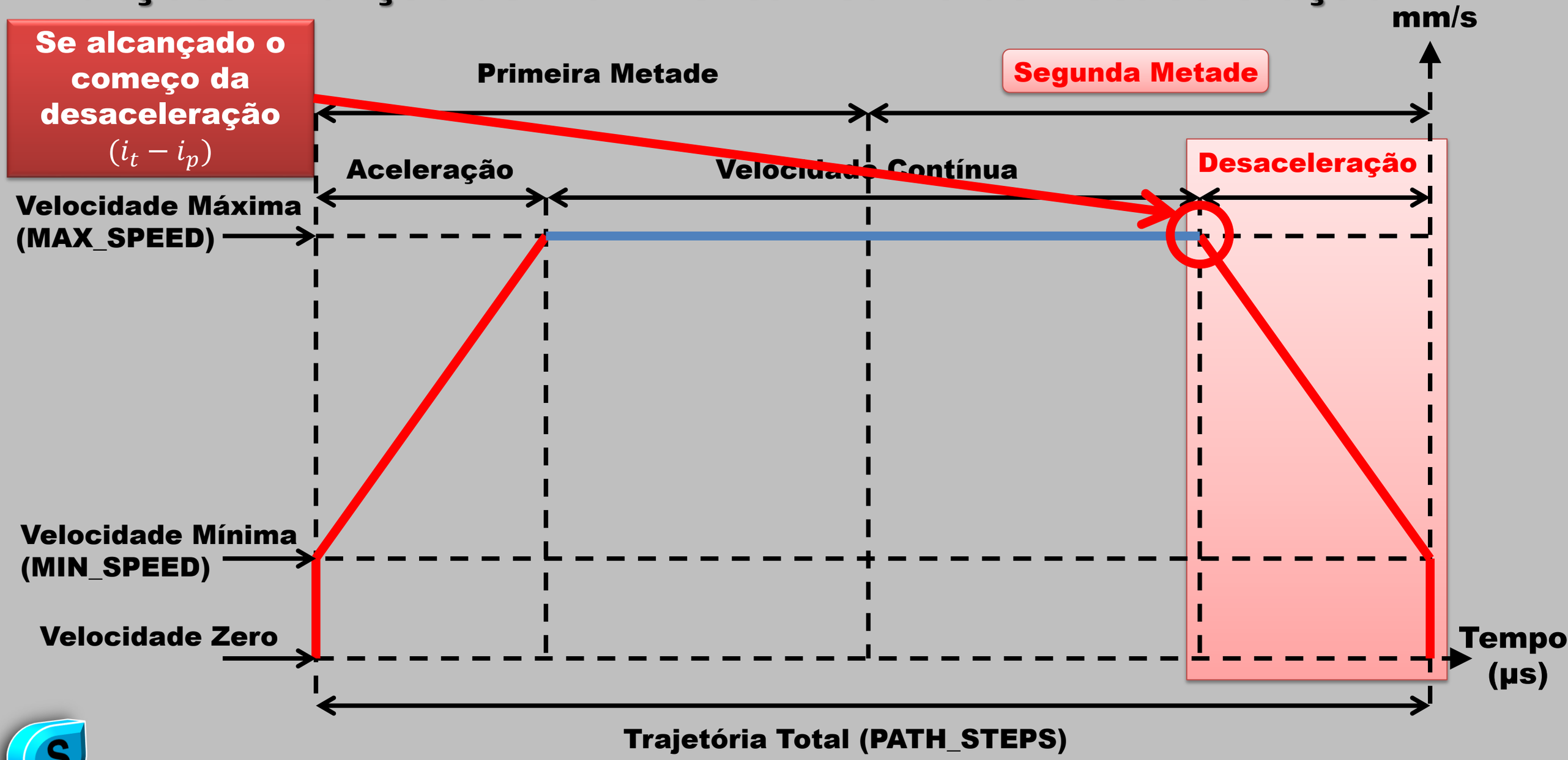
Desaceleração:

$$\frac{1}{\sqrt{V_{max}^2 + 2 * -AC * (i_a - (i_t - i_p))}}$$

V_{max} = Velocidade Máxima alcançada(mm/s)
AC = Aceleração(mm/s²)
i_a = Passo atual
i_t = Passo total da trajetória
i_p = Passo para parar

```
            //Calcula o periodo de desaceleração do motor:
            ACC_SPEED_PERIOD = 1.0 / sqrt((MAX_SPEED_REACHED * MAX_SPEED_REACHED) + 2.0 * (ACC * -
1.0) * (i - (PATH_STEPS - STEPS_TO_STOP)));

            //Converte o período de aceleração em segundos para microssegundos
            //e move o motor em um passo na direção solicitada
            motorMove(1000000 * ACC_SPEED_PERIOD, direction);
        }
```

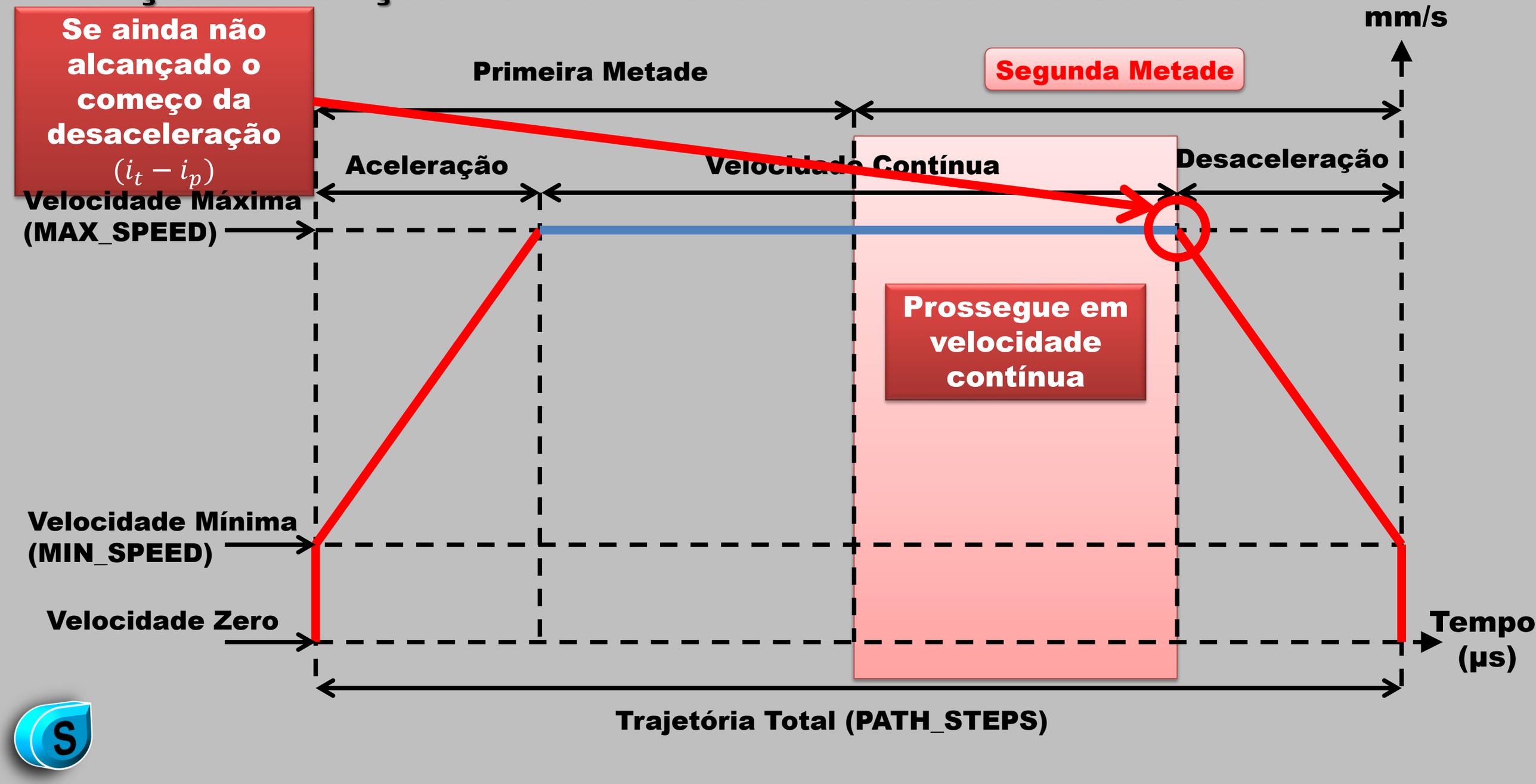
Funções – Função de Movimento - Curva de Desaceleração



Funções – Função de Movimento – Velocidade Contínua

```
        else//Se ainda não alcançada a desaceleração
        {
            //Converte o período de velocidade máxima em segundos
            //para microssegundos e move o motor em um passo na direção solicitada
            motorMove(1000000 * MAX_SPEED_PERIOD, direction);

            }
        } //Fim da segunda metade da trajetória
    } //Fim do for
} //Fim da função
```

Funções – Função de Movimento – Velocidade Contínua



Funções – Função de Movimento – Mover Voltas

```
//Move o motor um determinado número de voltas na direção solicitada
void DRV8825::motorMoveRounds(int rounds, bool direction)
{
//converte converte o número de voltas solicitadas em milímetros
    double distance = (rounds * SPR )/SPM;
//Move o motor na direção solicitada
    motorMoveTo(distance, direction);

}
```

Gráfico de movimento – Velocidade em função da Posição

Posição	Pulso	Instante	Velocidade
1	1992	1992	502,0080321
2	1984	3976	504,0322581
3	1976	5952	506,0728745
4	1968	7920	508,1300813
5	1961	9881	509,9439062
6	1953	11834	512,0327701
7	1946	13780	513,8746146
8	1938	15718	515,995872
9	1931	17649	517,8663905
10	1924	19573	519,7505198
11	1917	21490	521,648409
12	1910	23400	523,5602094
13	1903	25303	525,4860746
14	1896	27199	527,4261603
15	1889	29088	529,3806247
16	1883	30971	531,0674456
17	1876	32847	533,0490405
18	1869	34716	535,0454789
19	1863	36579	536,7686527
20	1856	38435	538,7931034
21	1850	40285	540,5405405
22	1844	42129	542,2993492
23	1838	43967	544,0696409
24	1831	45798	546,149645
25	1825	47623	547,9452055
26	1819	49442	549,7526113
27	1813	51255	551,5719801
28	1807	53062	553,4034311
29	1801	54863	555,247085
30	1796	56659	556,7928731

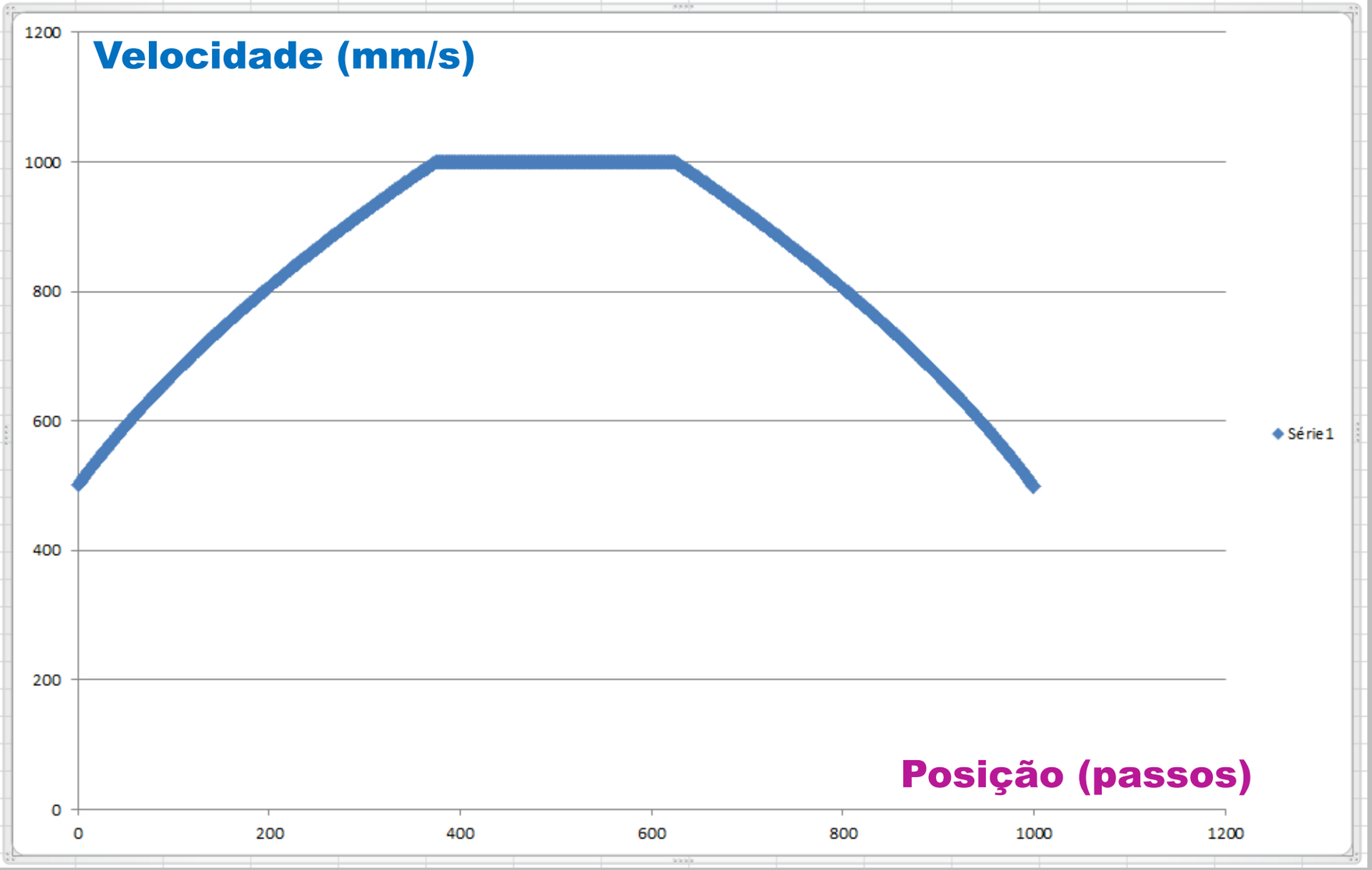


Gráfico de movimento – Período em função da Posição

Posição	Pulso	Instante	Velocidade
1	1992	1992	502,0080321
2	1984	3976	504,0322581
3	1976	5952	506,0728745
4	1968	7920	508,1300813
5	1961	9881	509,9439062
6	1953	11834	512,0327701
7	1946	13780	513,8746146
8	1938	15718	515,995872
9	1931	17649	517,8663905
10	1924	19573	519,7505198
11	1917	21490	521,648409
12	1910	23400	523,5602094
13	1903	25303	525,4860746
14	1896	27199	527,4261603
15	1889	29088	529,3806247
16	1883	30971	531,0674456
17	1876	32847	533,0490405
18	1869	34716	535,0454789
19	1863	36579	536,7686527
20	1856	38435	538,7931034
21	1850	40285	540,5405405
22	1844	42129	542,2993492
23	1838	43967	544,0696409
24	1831	45798	546,149645
25	1825	47623	547,9452055
26	1819	49442	549,7526113
27	1813	51255	551,5719801
28	1807	53062	553,4034311
29	1801	54863	555,247085
30	1796	56659	556,7928731

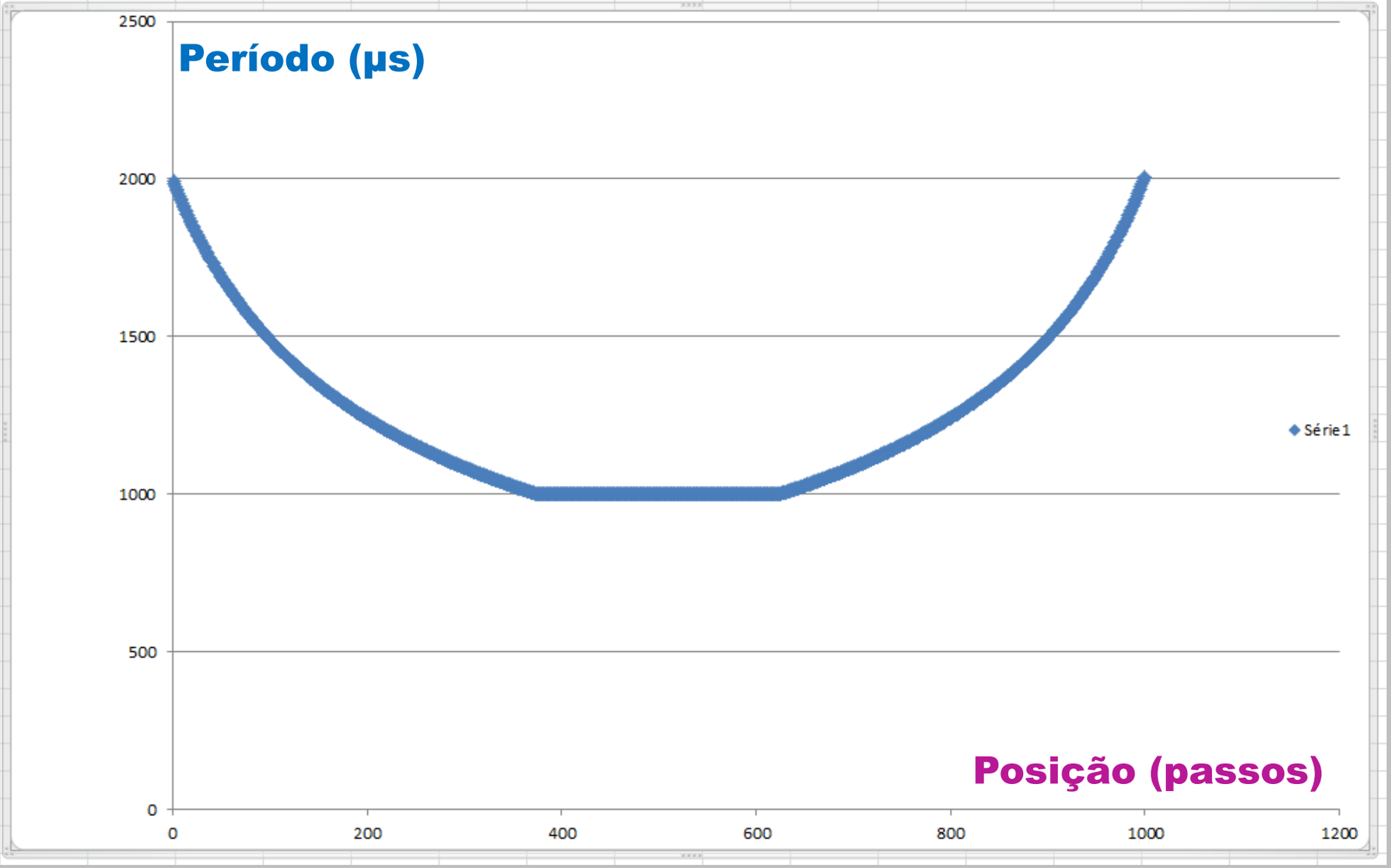
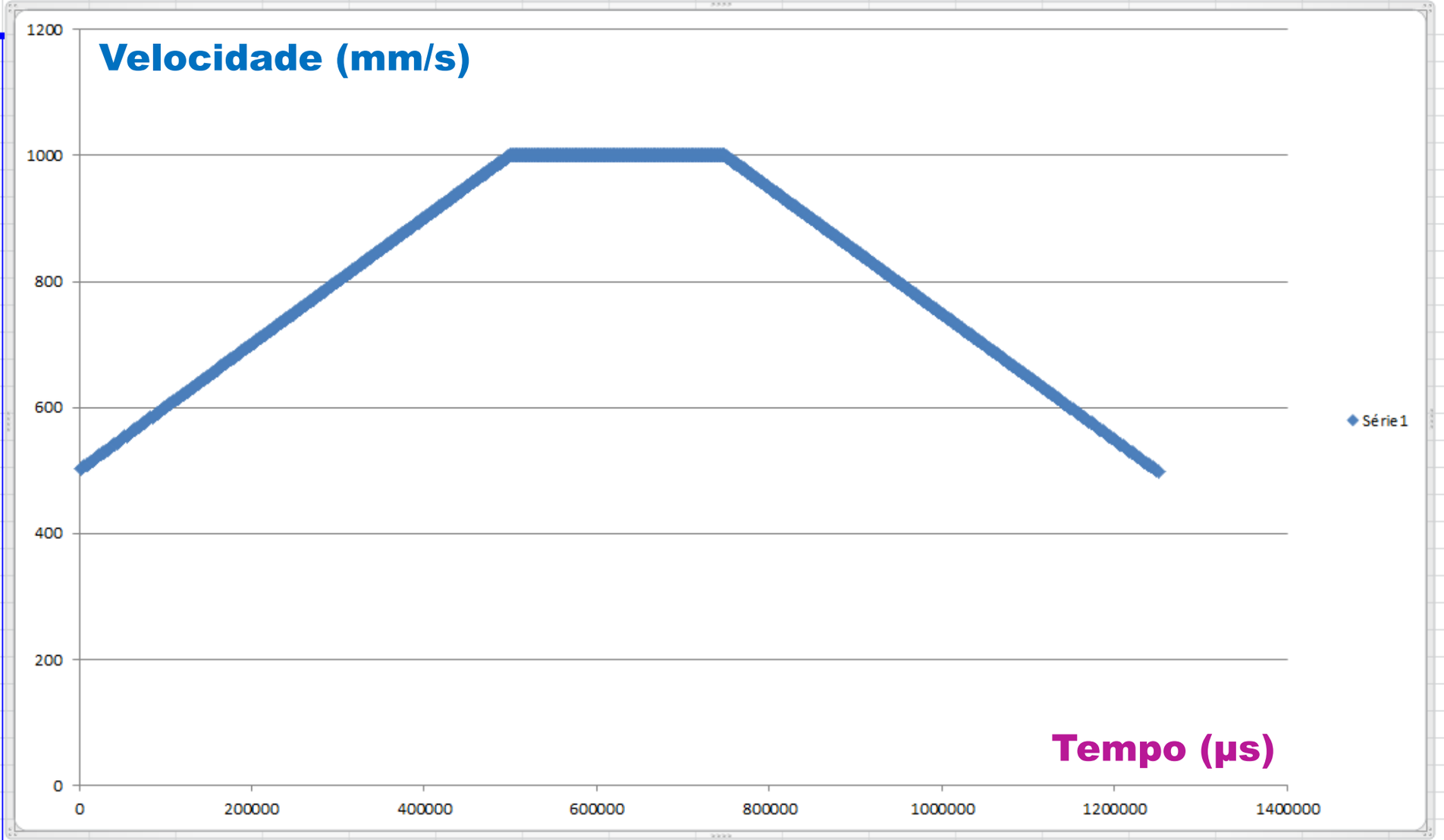
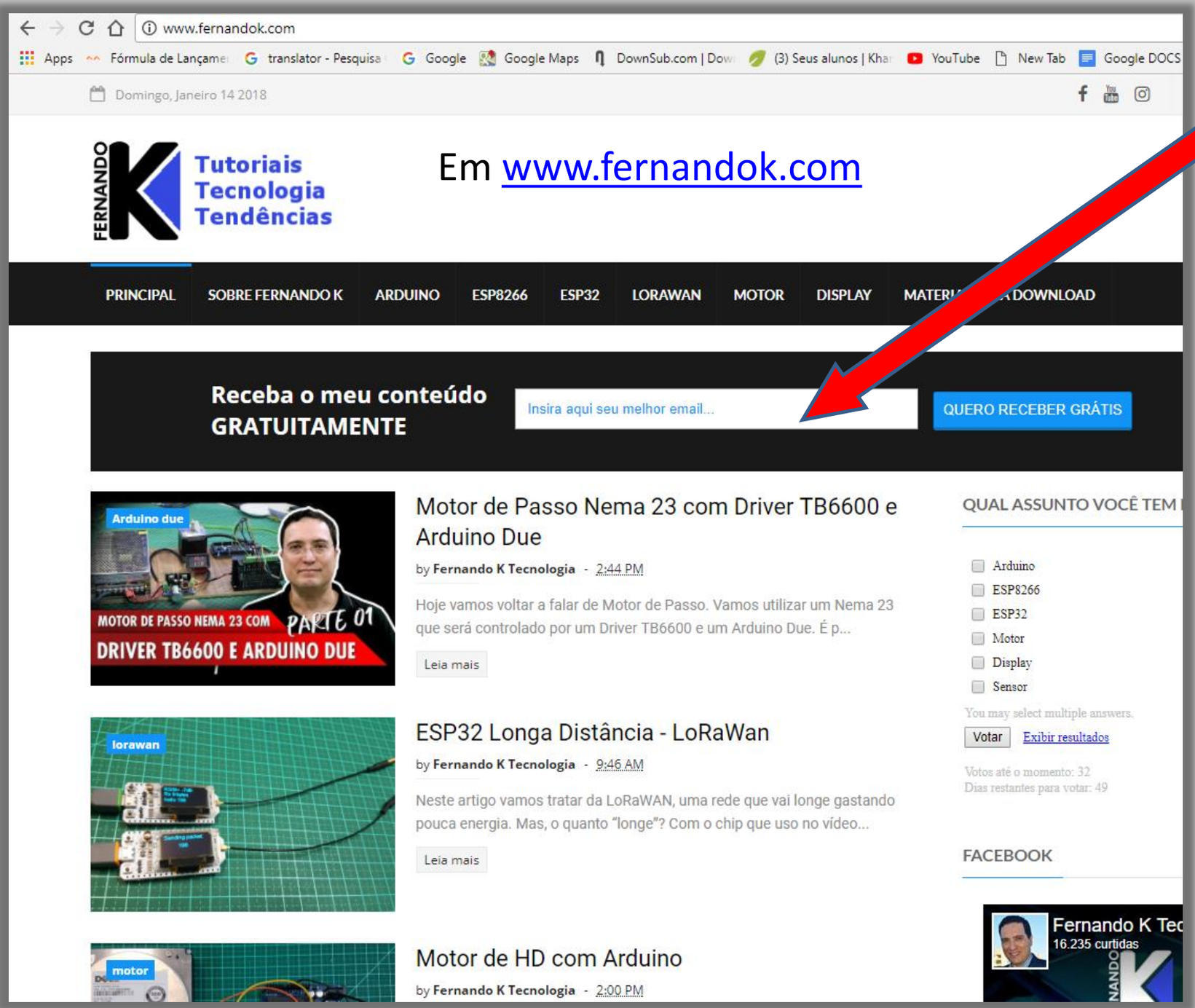


Gráfico de movimento – Velocidade em função do Instante

Posição	Pulso	Instante	Velocidade
1	1992	1992	502,0080321
2	1984	3976	504,0322581
3	1976	5952	506,0728745
4	1968	7920	508,1300813
5	1961	9881	509,9439062
6	1953	11834	512,0327701
7	1946	13780	513,8746146
8	1938	15718	515,995872
9	1931	17649	517,8663905
10	1924	19573	519,7505198
11	1917	21490	521,648409
12	1910	23400	523,5602094
13	1903	25303	525,4860746
14	1896	27199	527,4261603
15	1889	29088	529,3806247
16	1883	30971	531,0674456
17	1876	32847	533,0490405
18	1869	34716	535,0454789
19	1863	36579	536,7686527
20	1856	38435	538,7931034
21	1850	40285	540,5405405
22	1844	42129	542,2993492
23	1838	43967	544,0696409
24	1831	45798	546,149645
25	1825	47623	547,9452055
26	1819	49442	549,7526113
27	1813	51255	551,5719801
28	1807	53062	553,4034311
29	1801	54863	555,247085
30	1796	56659	556,7928731




www.fernandok.com
Apps
Fórmula de Lançame
translator - Pesquisa
Google
Google Maps
DownSub.com | Dow
(3) Seus alunos | Kha
YouTube
New Tab
Google DOCS
Domingo, Janeiro 14 2018
FERNANDO K
Tutoriais
Tecnologia
Tendências
Em www.fernandok.com
PRINCIPAL
SOBRE FERNANDO K
ARDUINO
ESP8266
ESP32
LORAWAN
MOTOR
DISPLAY
DOWNLOAD
Receba o meu conteúdo GRATUITAMENTE
Insira aqui seu melhor email...
QUERO RECEBER GRÁTIS
Arduino due
MOTOR DE PASSO NEMA 23 COM DRIVER TB6600 E ARDUINO DUE
PARTE 01
Motor de Passo Nema 23 com Driver TB6600 e Arduino Due
by Fernando K Tecnologia - 2:44 PM
Hoje vamos voltar a falar de Motor de Passo. Vamos utilizar um Nema 23 que será controlado por um Driver TB6600 e um Arduino Due. É p...
Leia mais
lorawan
ESP32 Longa Distância - LoRaWan
by Fernando K Tecnologia - 9:46 AM
Neste artigo vamos tratar da LoRaWAN, uma rede que vai longe gastando pouca energia. Mas, o quanto "longe"? Com o chip que uso no vídeo...
Leia mais
motor
Motor de HD com Arduino
by Fernando K Tecnologia - 2:00 PM
QUAL ASSUNTO VOCÊ TEM
Arduino
ESP8266
ESP32
Motor
Display
Sensor
You may select multiple answers.
Votar
Exibir resultados
Votos até o momento: 32
Dias restantes para votar: 49
FACEBOOK
Fernando K Tec
16.235 curtidas

Seu e-mail

Em www.fernandok.com

Download arquivos PDF e INO do código fonte