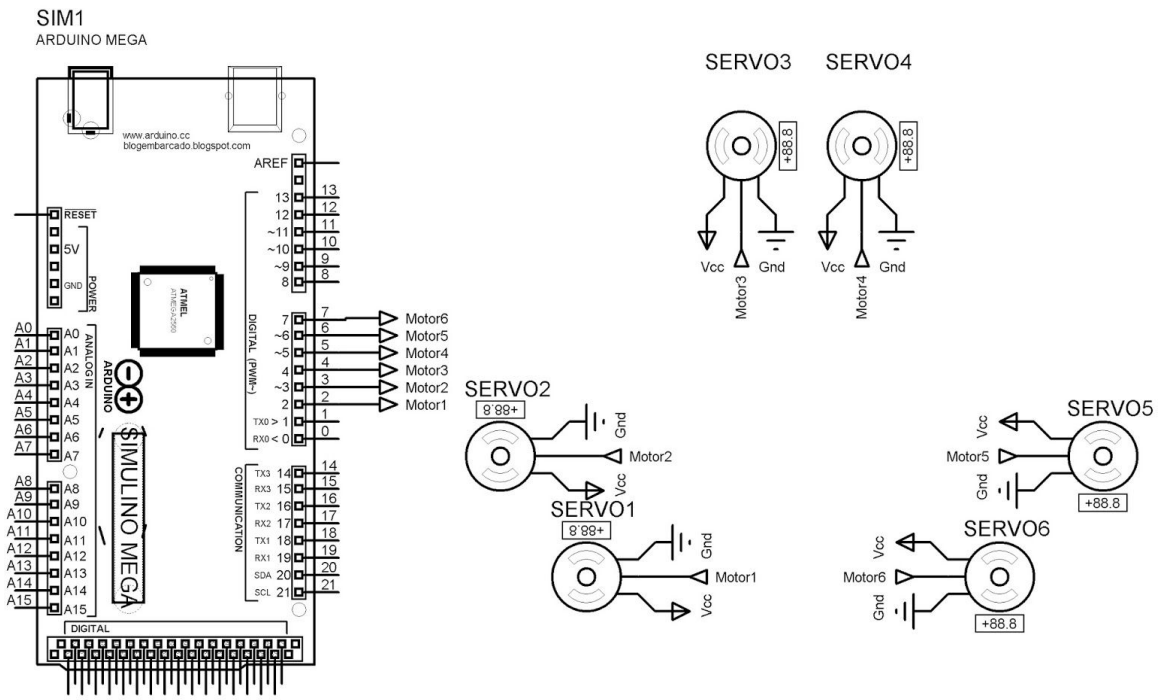
Sumário

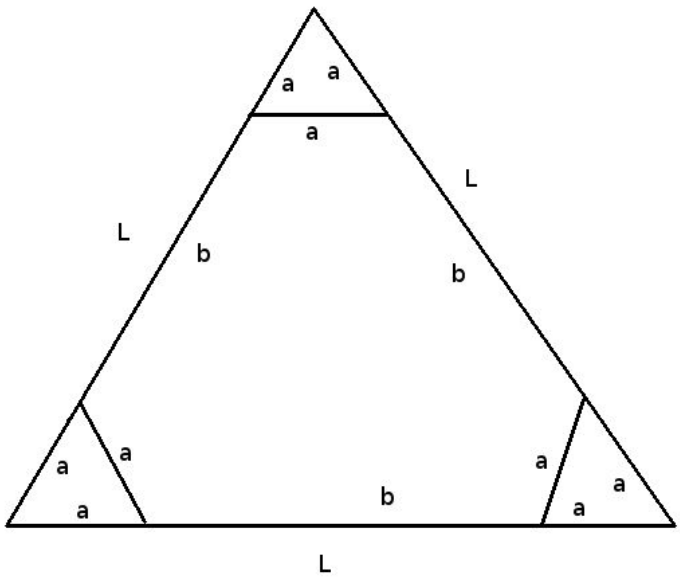
1. Esquema


SIM1
ARDUINO MEGA
www.arduino.cc
blogembarcado.blogspot.com
AREF
RESET
5V
GND
POWER
ANALOG IN
DIGITAL (PWM~)
COMMUNICATION
DIGITAL
ARDUINO
SIMULINO MEGA
Motor6
Motor5
Motor4
Motor3
Motor2
Motor1
SERVO3
SERVO4
SERVO2
SERVO1
SERVO5
SERVO6
Vcc
Gnd
+88.8
Motor3
Motor4
Motor2
Motor1
Motor5
Motor6

2. Plataforma

2.1 Base superior

A base superior (onde ficarão as juntas) é feita a partir de um triângulo equilátero com lados iguais a 20 cm. Nas pontas desse triângulo maior serão feitos triângulos equiláteros com lados iguais a 5 cm.



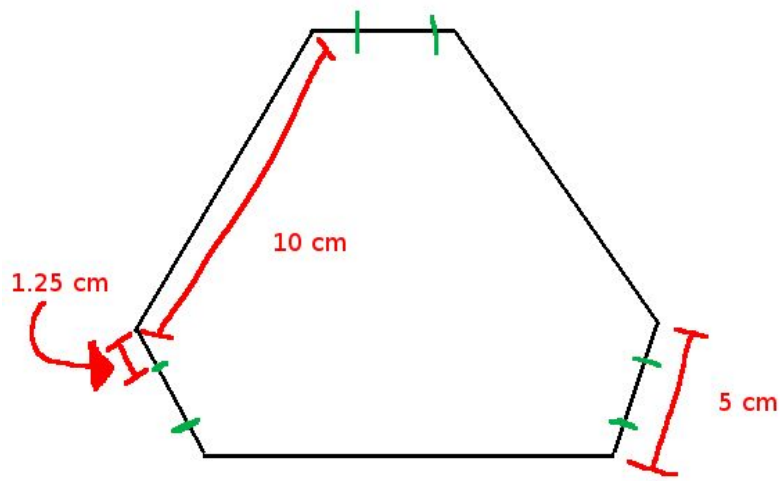
L: 20 cm
a: 5 cm
b: 10 cm

Onde ‘L’ é o lado do triângulo maior;

‘a’ é o lado do triângulo menor;

‘b’ = ‘L’ - ‘a’

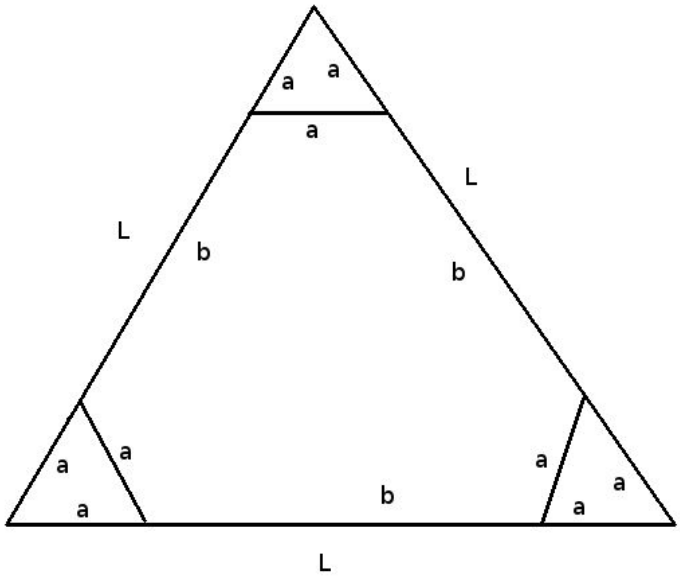
Após feita as marcações dos triângulos menores, basta retirá-los. Com isso teremos a seguinte base hexagonal:



As marcações verdes dessa base representam onde as juntas serão colocadas.
Todas as juntas deverão estar apontando para o baricentro do hexagono.

2.2 Base inferior

A base inferior (onde ficarão os motores) é feita a partir de um triângulo equilátero com lados iguais a 18 cm. Nas pontas desse triângulo maior serão feitos triângulos equiláteros com lados iguais a 5 cm.



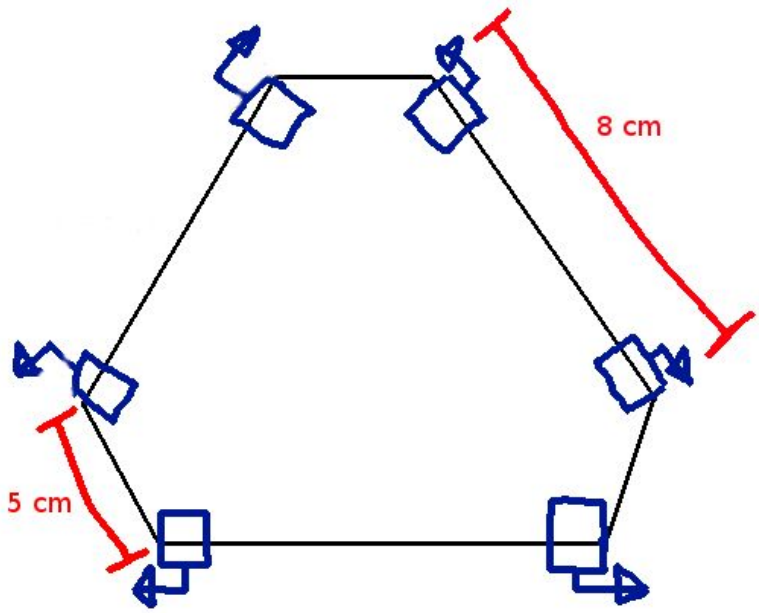
L: 18 cm
a: 5 cm
b: 8 cm

Onde ‘L’ é o lado do triângulo maior;

‘a’ é o lado do triângulo menor;

‘b’ = ‘L’ - ‘a’ - ‘a’

Após feita as marcações dos triângulos menores, basta retirá-los. Com isso teremos a seguinte base hexagonal:



As caixas azuis representam os servos motores e as setas deles apontam para a direção em que a junta está ‘apontada’.

2.3 Juntas

As juntas foram feitas com um modelo 3D baixado da internet e adaptado para melhorar a mobilidade. O modelo adaptado está disponível no repositório do github. Modelo original: https://grabcad.com/library/wgrm-06-lc-ball-and-socket-joint

2.4 Hastes

As hastes foram feitas utilizando palitos de churrasco. As hastes ficaram com um comprimento de 15 cm. Foi calculado que 1.5 cm de cada ponta da haste ficaria para dentro das juntas.

2.5 Imagens e vídeos

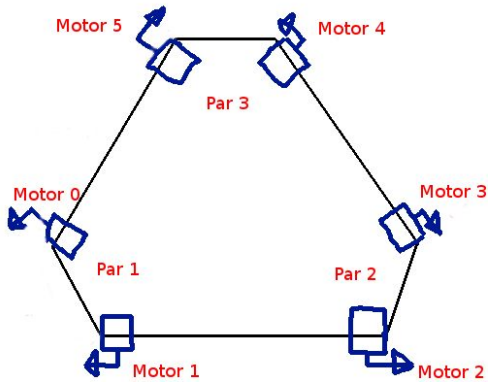
As imagens de vídeos da plataforma completa estão disponíveis no repositório do github.

3. Motores

Na implementação, os motores foram colocados de uma forma em que conseguimos movimentá-los individualmente ou em pares. Temos o vetor servo[6] que armazena todos os 6 motores.

3.1 Pares e índices

Os motores [0] e [1] formam o par 1;
Os motores [2] e [3] formam o par 2;
Os motores [4] e [5] formam o par 3;



3.2 Movimentação

Para movimentar os motores passamos o ângulo desejado, de 0º a 180º. Sendo 0º o mais baixo dos motores e 180º sendo o mais alto dos motores.

No nosso caso, os motores de cada par ficaram invertidos entre si. Por exemplo, para subir os 2 motores do par 1 devemos mandar um grau a cima de 90º para o Motor 1 e o inverso desse com relação à 90º para o Motor 0.

Sendo mais específico, para subir o par 1 para 130º por exemplo, devemos mandar 130º para o Motor 1 e (90 - (130 - 90)) = 50º para o Motor 0.

Todo motor de índice par (0, 2 e 4) recebe o inverso do grau desejado, enquanto todo motor de índice ímpar (1, 3 e 5) recebe o grau desejado.

Códigos:

```
17 ▼ int inverso(int grau){
18       int aux = grau - 90;
19       int inverso = 90 - aux;
20
21       return inverso;
22   }
```

```
54 ▼ void mover_par(int par, int grau){
55       int motor_1, motor_2;
56
57       par *= 2;
58       motor_1 = par-2;
59       motor_2 = par-1;
60
61       servo[motor_1].write(inverso(grau));
62       servo[motor_2].write(grau);
63   }
64
65 ▼ void mover(int motor, int grau){
66       if(motor % 2 == 0){
67           servo[motor].write(inverso(grau));
68       }
69       else{
70           servo[motor].write(grau);
71       }
72   }
```

Para alguns movimentos, percebemos que movimentamos um motor ou um par de motor X graus e movemos outro par ou outro motor uma quantidade Y de graus que corresponde a uma fração do grau X.

E percebemos também que essa quantidade Y pode ser incremento no posição neutra 90º ou um decremento da posição neutra.

Com base nisso implementamos funções para calcular o grau final do movimento da seguinte forma:

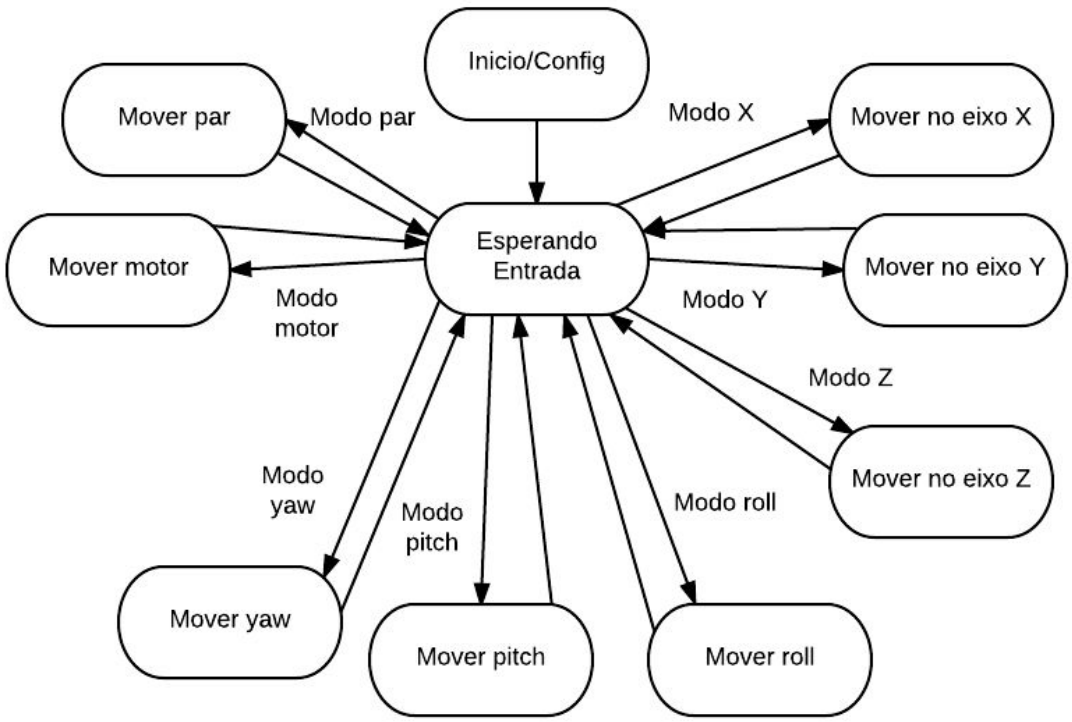
```
24   int inverso_coef(int grau, float coef){
25       int aux;
26
27       if(grau >= 90){
28           aux = grau - 90;
29       }
30       else{
31           aux = 90 - grau;
32       }
33
34       int inverso = 90 - (aux*coef);
35
36       return inverso;
37   }
38
39   int inverso_coef_compl(int grau, float coef){
40       int aux;
41
42       if(grau >= 90){
43           aux = grau - 90;
44       }
45       else{
46           aux = 90 - grau;
47       }
48
49       int inverso = 90 + (aux*coef);
50
51       return inverso;
52   }
```

3.3 Modos

Foi implementado 8 modos de controlar os motores. Sendo esses:

1. Mover no eixo X
2. Mover no eixo Y
3. Mover no eixo Z
4. Mover roll
5. Mover pitch
6. Mover yaw
7. Mover motor
8. Mover par

3.3.0 Máquina de estados



3.3.1 Mover no eixo X

Mover no eixo X foi interpretado como mover a plataforma para o lado direito e esquerdo. Implementado com o seguinte código:

```
void mover_x(int grau){
    mover(4, grau);
    mover(5, inverso_coef_compl(grau, 0.88) );

    int grau_x = inverso_coef(grau, 0.56);
    mover(0, grau_x);
    mover(1, inverso_coef(grau_x, 0.8) );

    grau_x = inverso_coef_compl(grau, 0.33 );
    mover(2, grau_x);
    mover(3, inverso_coef_compl(grau_x, 0.33 ) );
}
```

3.3.2 Mover no eixo Y

Mover no eixo Y foi interpretado como mover a plataforma para frente e para trás. Não foi implementado.

3.3.3 Mover no eixo Z

Mover no eixo Z foi interpretado como mover a plataforma para cima e para baixo. Implementado com o seguinte código:

```
void mover_z(int grau){
    for(int i = 0; i < 6; i++){
        mover(i, grau);
    }
}
```

3.3.4 Mover roll

Mover roll foi interpretado como inclinar a plataforma para o lado direito e esquerdo. Implementado com o seguinte código:

```
void mover_roll(int grau){
    if(grau <= 40){
        grau = 40;
    }

    if(grau > 90){
        mover_par(2, grau);

        mover(0, inverso_coef(grau, 0.214));
        mover(1, inverso_coef_compl(grau, 0.214));

        mover_par(3, inverso(grau));
    }

    else{
        mover_par(2, grau);

        mover(0, inverso_coef_compl(grau, 0.214));
        mover(1, inverso_coef(grau, 0.214));

        mover_par(3, inverso(grau));
    }
}
```

3.3.5 Mover pitch

Mover pitch foi interpretado como inclinar a plataforma para frente e para trás. Implementado com o seguinte código:

```
void mover_pitch(int grau){
    if(grau >= 90){
        if(grau > 160){
            grau = 160;
        }

        int grau_x = inverso(grau);
        mover_par(1, grau);

        mover(3, grau_x);
        mover(4, grau_x);
    }
    else{
        int grau_x = inverso_coef_compl(grau, 0.625);
        mover_par(1, grau);

        mover(3, grau_x);
        mover(4, grau_x);
    }
}
```

3.3.6 Mover yaw

Mover yaw foi interpretado como girar a plataforma no próprio eixo. Implementado com o seguinte código:

```
void mover_yaw(int grau){
    mover(0, grau);
    mover(1, inverso_coef(grau, 0.153) );
    mover(2, grau);
    mover(3, inverso_coef(grau, 0.153) );
    mover(4, grau);
    mover(5, inverso_coef(grau, 0.153) );
}
```

3.3.7 Mover motor

Mover motor foi interpretado como mover um motor para um determinado grau. Implementado com o seguinte código:

```
void mover(int motor, int grau){
    if(motor % 2 == 0){
        servo[motor].write(inverso(grau));
    }
    else{
        servo[motor].write(grau);
    }
}
```

3.3.8 Mover par

Mover par foi interpretado como um par de motor para um determinado grau. Implementado com o seguinte código:

```
void mover_par(int par, int grau){
    int motor_1, motor_2;

    par *= 2;
    motor_1 = par-2;
    motor_2 = par-1;

    servo[motor_1].write(inverso(grau));
    servo[motor_2].write(grau);
}
```

4. Código Fonte

O código fonte da Plataforma de Stewart juntamente com imagens, vídeos e projeto do Proteus se encontram no github https://github.com/rodrigondec/stewart_est.