

Manual Robô de batalha

*Grupo Skynet*

Este manual tem por objetivo instruir e auxiliar na montagem e uso do sistema integrado a estrutura do robô de batalha desenvolvido pelo grupo Skynet, assim como advertir o usuário para determinados cuidados que podem resultar em mal funcionamento ou danos ao sistema.

Sendo este robô pertencente a categoria bettleweight, onde o seu peso limite é de 1,36Kg e busca atender as regras estabelecidas pela RoboCore.

[Esboço Mecânico 2](#_Toc19008351)

[Circuitos elétrico e eletrônico 4](#_Toc19008352)

[Bateria 4](#_Toc19008353)

[Chave Liga-Desliga 5](#_Toc19008354)

[Arduino MEGA2560 5](#_Toc19008355)

[Ponte H 7](#_Toc19008356)

[Fusíveis 9](#_Toc19008357)

[Motores 9](#_Toc19008358)

[Motor Brushless 11](#_Toc19008359)

[Controlador ESC 12](#_Toc19008360)

[Receptor 13](#_Toc19008361)

[Canais: 13](#_Toc19008362)

[Programa 15](#_Toc19008363)

# Esboço Mecânico

A estrutura mecânica é composta em sua maior parte por chapas de alumínio devido a este ser um metal mais leve. As rodas são compostas por Silicone e polipropileno de alta qualidade.

Uma imagem contendo música, kazoo, ocarina

Descrição gerada automaticamente

Seu formato foi projetado para integração de uma arma de disco giratório, a qual é acoplada em um eixo presente na estrutura triangular e será rotacionado com o uso de um motor Brushless de 1500kv.

Uma imagem contendo engrenagem, interior, foto, metalúrgico

Descrição gerada automaticamente

Uma imagem contendo texto, mapa

Descrição gerada automaticamente

# Circuitos elétrico e eletrônico

Os circuitos elétrico e eletrônico são os responsáveis por fazer com que o funcionamento dos movimentos mecânicos ocorra de acordo com o desejado e instruído via controle.

Os circuitos integrados a estrutura são compostos pelos seguintes itens:

* 1 chave liga-desliga
* 1 Bateria de 11.1V
* 1 Arduíno Mega2560
* 1 Driver Ponte H
* 2 Fusíveis de 2 A
* 2 Motores com Caixa de Redução 6V 320RPM
* 1 Motor Brusheless
* 1 Controlador de velocidade ESC 100 A
* 1 Módulo receptor de sinais do controle

## Bateria

Uma imagem contendo chão

Descrição gerada automaticamente

Para alimentação de energia do circuito é utilizada uma bateria de 11.1V de 3000 mAH, a qual realiza a alimentação do controlador de velocidade ESC, do Arduíno e do ponte H.

A alimentação é realizada através dos terminais vermelho e preto, sendo respectivamente VCC e VDD.

Uma imagem contendo chão

Descrição gerada automaticamente

*Obs: Antes de fazer o uso do robô verifique se a bateria se encontra carregada.*

## Chave Liga-Desliga

Uma imagem contendo sentado, preto, interior

Descrição gerada automaticamente

Por medidas de segurança e para facilitar alterações manutenções no circuito há conectada na alimentação deste uma chave para acionamento, a qual é ligada ao terminal positivo da bateria e ao terminal de alimentação do ESC.

Uma imagem contendo objeto

Descrição gerada automaticamente

## Arduino MEGA2560

Uma imagem contendo equipamentos eletrônicos, circuito

Descrição gerada automaticamente

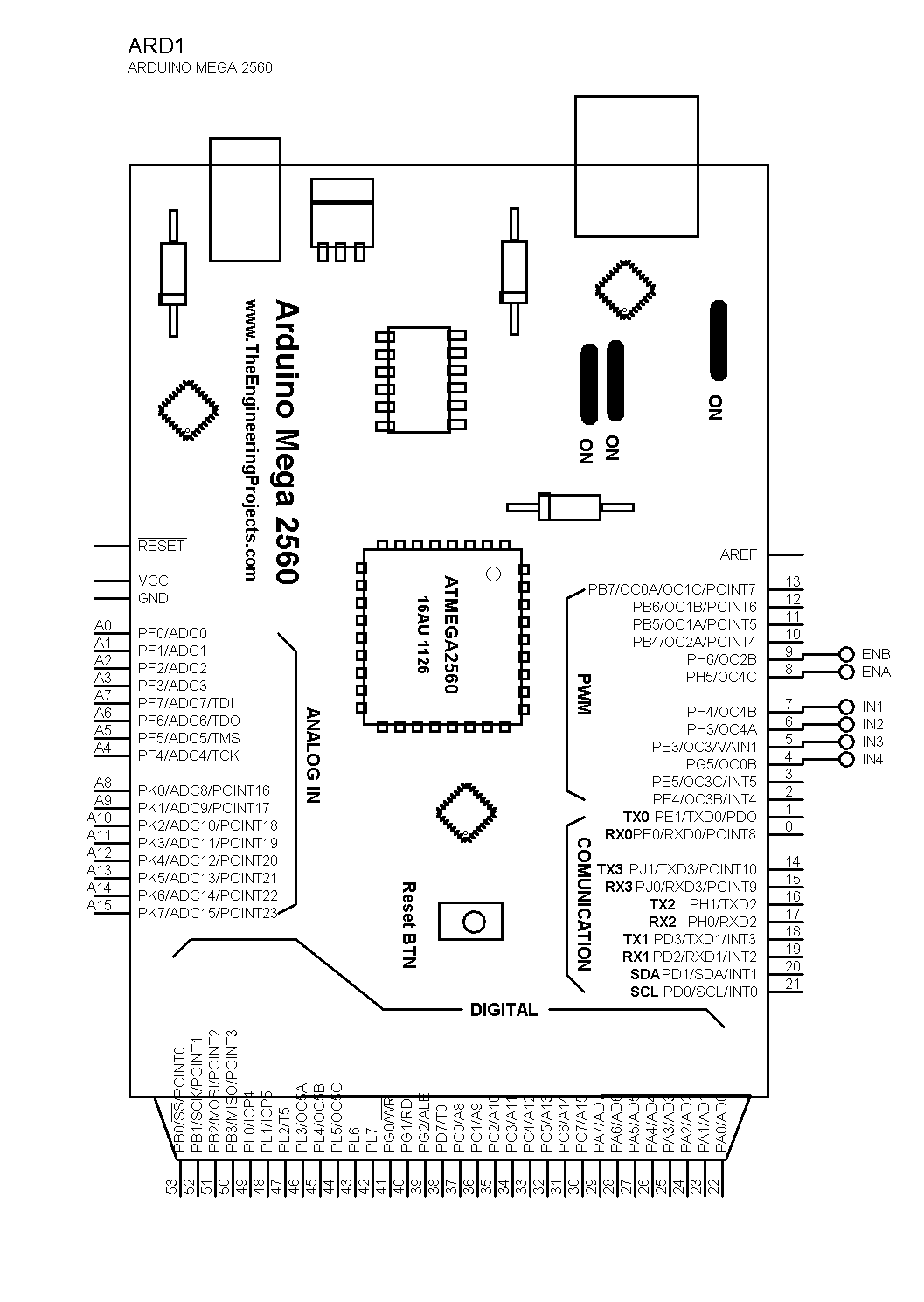
Os circuitos foram desenvolvidos tendo como base de processamento das informações um Arduíno MEGA2560, o qual é responsável por receber os sinais provindos do receptor do controle remoto e interpreta-los, assim vindo a realizar as ações estabelecidas para determinados eventos, tal como exemplo ordenar a rotação dos motores em determinada direção conforme é realizado o movimento do joystick do controle.

Para este circuito foram utilizadas as portas digitais do Arduino, utilizadas para enviar comandos ao ponte H para interagir com os motores, sendo ainda especificamente utilizadas as portas 8 e 9 , as quais possuem a função de PWM (Modulação por largura de pulso), o que torna possível o controle da velocidade dos motores.

*Para um melhor entendimento do funcionamento de um Arduíno consulte o datasheet deste ou documentações especificas.*

Assim a relação dos terminais segue da seguinte forma:

|  |  |  |
| --- | --- | --- |
| Terminal | Correspondente | Motor |
| 4 | IN4 | Direito |
| 5 | IN3 | Direito |
| 6 | IN2 | Esquerdo |
| 7 | IN1 | Esquerdo |
| 8 | EA | - |
| 9 | EB | - |



Uma imagem contendo texto

Descrição gerada automaticamente

## Ponte H

Para realizar a interação com os motores é utilizado um driver ponte H, o qual consiste em um circuito responsável por controlar o sentido de corrente elétrica, assim como controlar sua potência para o acionamento de uma determinada carga, dessa forma este circuito torna possível o controle da direção de rotação dos motores e o controle de sua velocidade.

Para estes circuitos o driver utilizado foi o modelo L298N, o qual possuí as seguintes especificações:

* Tensão de Operação: 4~35v
* Chip: ST L298N
* Controle de 2 motores DC ou 1 motor de passo
* Corrente de Operação máxima: 2A por canal ou 4A max
* Tensão lógica: 5v
* Corrente lógica: 0~36mA
* Limites de Temperatura: -20 a +135°C
* Potência Máxima: 25W
* Dimensões: 43 x 43 x 27mm
* Peso: 30g

Uma imagem contendo céu

Descrição gerada automaticamente

MOTOR A e B:

São os conectores para ligação de 2 motores DC ou 1 motor de passo.

ATIVA MA(EA) e MB(EB):

São os terminais onde é realizado o controle PWM dos motores, para sua utilização é necessário retirar o jumper presente nestes e realizar a conexão aos terminais de PWM do Arduíno, assim sendo possível realizar o controle da potência do motor.

Ativa 5V:

Este terminal disponibiliza no terminal 5V uma tensão deste valor para uso externo quando o jumper estiver conectado, sendo este recurso acessível quando estiver trabalhando com tensões entre 6-35 V, sendo necessário que não se não alimente o terminal 5V nesta situação, pois poderá haver danos a placa. O terminal 5V apenas recebe alimentação provinda do Arduino caso se esteja operando um motor de 4-5v e o jumper estiver desconectado.

6-35V e GND:

Estes terminais recebem a alimentação externa quando estiver operando motores entre 6 e 35V, sendo neste circuito conectada a bateria a este terminal.

Entrada: Consiste em um barramento onde serão conectados os pares de terminais responsáveis pela rotação dos motores.

Motor A - IN1 e IN2

Motor B – IN3 e IN4

Para realizar o controle da direção de rotação dos motores é necessária realizar na programação do Arduino o envio de pares de informações para cada motor, sendo o comportamento destes pares da seguinte forma, a exemplo o controle do motor A:

|  |  |  |
| --- | --- | --- |
| Sentido | IN1 | IN2 |
| Horário | 5V | GND |
| Anti-Horário | GND | 5V |
| Ponto Morto | GND | GND |
| Freio | 5V | 5V |

## Fusíveis



Para a proteção do circuito e dos motores foram conectados dois fusíveis na alimentação dos motores no ponte H, sendo um para cada motor, sendo a principal função do fusível a de realizar a proteção do circuito em caso de elevado nível de corrente, onde este deverá abrir o circuito assim que a corrente atingir um valor acima do definido.

Neste caso os fusíveis utilizados permitiram uma corrente de no máximo 2A na saída do ponte H, sendo está a maior corrente permitida pelo driver em cada saída ao trabalhar com dois motores DC.

Ambos os fusíveis se encontram alojados em porta-fusíveis para fácil substituição se necessário.

Uma imagem contendo ao ar livre

Descrição gerada automaticamente

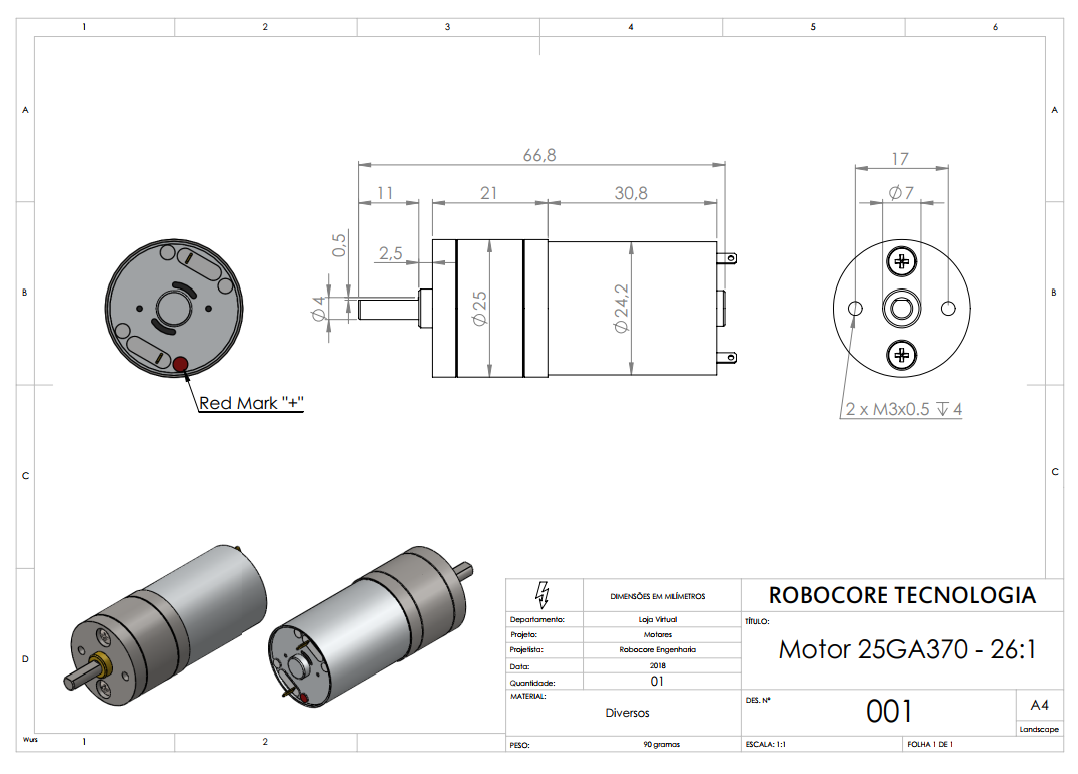
*Em caso de os motores não acionarem e o circuito se encontrar em perfeito estado verifique se os fusíveis se encontram alojados e em perfeito estado nos porta-fusíveis e os mesmos conectados ao circuito.*

## Motores

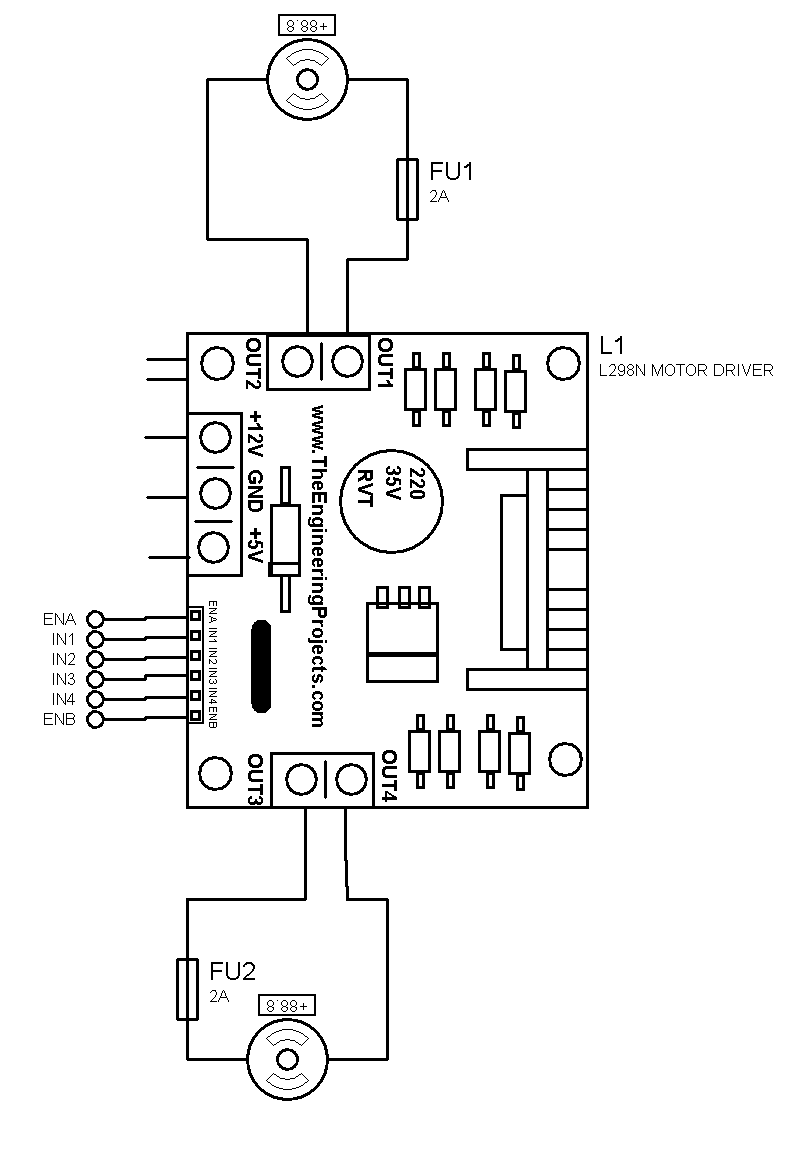


Para realizar o movimento da estrutura as rodas são acopladas a motores DC, sendo ambos os motores do modelo 25GA370, possuindo engrenagens usinadas em aço e uma caixa sólida e robusta, vindo a ter as seguintes especificações:

* Tensão nominal: 6 Vdc
* Rotação nominal: 320 rpm
* Corrente (sem carga): 170 mA
* Corrente (stall): 2,8 A
* Torque (stall): 3,83 kgf.cm
* Relação de Redução: 26:1
* Diâmetro do eixo: 4mm
* Diâmetro da caixa de redução: 25mm
* Comprimento total com eixo: 67mm
* Comprimento da caixa e motor: 54mm
* Peso: 90g



*Desenho de um motor do modelo utilizado*



*Ligação entre Arduino, driver e motores.*

## Motor Brushless



O motor Brushless consiste em um motor DC onde há a ausência de escovas, um componente responsável por transmitir contato elétrico entre as partes fixas e moveis do motor, desta forma este tipo de motor faz o uso da atração e repulsão de imãs no rotor através do acionamento das bobinas no estator de forma sequencial.

*Para maiores detalhes sobre tais processos consulte manuais e documentações a respeito do funcionamento de motores DC.*

Em vista disto para o acionamento e controle deste motor é necessário um controlador de velocidade ESC.

Uma imagem contendo texto

Descrição gerada automaticamente

*Ligação de um motor Brushless*

Desta forma os terminais das bobinas do motor devem ser conectados ao controlador ESC.

## Controlador ESC



O ESC (Electronic Speed Control) é um controlador de velocidade para os motores Brushless, o qual desempenha esta função através da variação da transição entre as fases deste motor, além desta aplicação o ESC pode servir para freio, variação lenta da velocidade do motor, entre outras.

Além destas funções o ESC possui internamente um componente BEC (Battery Eliminator Circuit), um dispositivo que converte a tensão de entrada do ESC (5,6 a 16,8V) em 5 ou 6V para alimentar o receptor e outros componentes.

O ESC possuí 8 terminais:

* 3 terminais para ligação nos terminais do motor;
* 2 terminais, vermelho e preto, para alimentação;
* Os 3 terminais restantes são 1 vermelho (5V) e 1 preto (GND) que servem como fonte de alimentação, e por último um branco (sinal), o qual é utilizado para controle do ESC.

## Receptor

Uma imagem contendo céu, objeto

Descrição gerada automaticamente

Para o controle a distância do sistema é utilizado um controle de radiofrequência do modelo FlySky FS-i6, normalmente utilizado para controle de aeromodelos e outras aplicações semelhantes, este disponibiliza 6 canais para uso, os quais cada um corresponde a um elemento específico do controle.

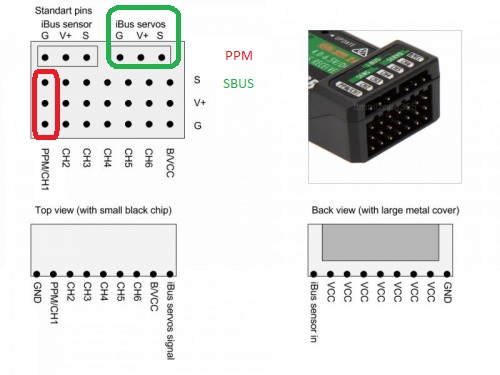
*Para maiores detalhes consulte o manual ou documentos relacionados ao controle.*

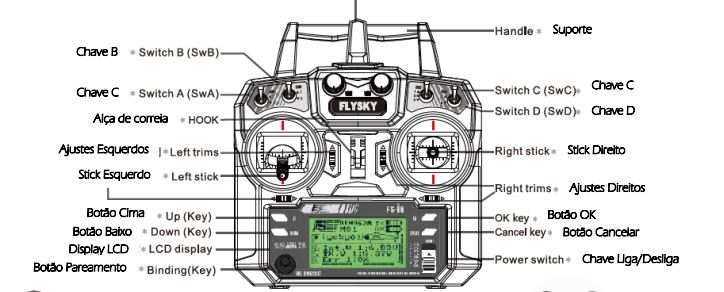
O receptor do controle por sua vez é o elemento responsável por captar os sinais enviados para cada canal e direcionar estas informações para o processamento, no caso realizado pelo Arduino, com exceção do ESC, o qual se encontra recebendo um sinal direto do receptor.

O receptor funciona com uma alimentação de 4 – 6,5V.

### Canais:

A identificação de cada um dos canais se dá na seguinte forma:





*CH1 Stick Direito Horizontal*

*CH2 Stick Direito Vertical*

*CH3 Stick Esquerdo Vertical*

*CH4 Stick Esquerdo Horizontal*

*CH5 Programável*

*CH6 Programável*

O CH1 foi utilizado para controle do direcionamento horizontal dos motores, sendo assim responsável por fazer com que o robô rotacione para esquerda ou direita.

O CH2 por sua vez foi utilizado para o direcionamento vertical dos motores, sendo responsável por fazer com que o robô avance ou recue.

Além disso o controle apresenta ajustes para calibrar os sticks e chaves programáveis para o CH5 ou CH6.

*Procure por documentações, vídeos e manuais para informações sobre como realizar a programação das chaves.*

Para realizar o bind (conectar o controle ao receptor) é necessário seguir os seguintes passos:

* Conectar o canal para bind no receptor.
* Conectar o ESC no CH1 ou CH3.
* Ligar o circuito do ESC e receptor, assim o receptor deverá estar sinalizando em uma determinada frequência com a luz presente.
* Manter pressionado o botão para bind no controle e ligar o controle, com isso deverá aparecer no controle uma mensagem informando que o bind foi realizado.
* Desligar o circuito e o controle.
* Desconectar o canal para bind e realizar as ligações desejadas.
* Ligar o controle e o circuito, assim a comunicação deverá estar funcionando normalmente.

# Programa

#define ch01 3 // Horizontal

#define ch02 2 // Vertical

#define ch05 10 // Inverter programacao

unsigned long x01=0, x02=0, x05=0;

int vel;

void setup() {

Serial.begin(9600);

pinMode(ch05, INPUT);

pinMode(ch01, INPUT);

pinMode(ch02, INPUT);

pinMode(4, OUTPUT); //Motor Direita

pinMode(5, OUTPUT); //Motor Direita

pinMode(6, OUTPUT); //Motor Esquerda

pinMode(7, OUTPUT); //Motor Esquerda

pinMode(8, OUTPUT); //Controle de velocidade

pinMode(9, OUTPUT); //Controle de velocidade

}

void loop() {

x05 = pulseIn(ch05, HIGH);

x02 = pulseIn(ch02, HIGH);

x01 = pulseIn(ch01, HIGH);

Serial.print("Horizontal \t");

Serial.print(x01);

Serial.print("\t");

Serial.print("Vertical \t");

Serial.print(x02);

Serial.print("\t");

Serial.print("Velocidade \t");

Serial.println(vel);

if((x01>1320 && x01<1390)&&(x02>1320 && x02<1390)){ //Mantem parado

vel = 0;

analogWrite(8, vel);

analogWrite(9, vel);

freio();

}

//======================================Config Channel 01================================

if(x01<=1320){ //Esqueda - Diminui

vel = map(x01, 1320, 986, 0, 127);

analogWrite(8, vel);

analogWrite(9, vel);

if(x05<1470){ esquerda();}

else{ direita();}

}

if(x01>=1390){ //Direita - Aumenta

vel = map(x01, 1390 , 1851, 0, 127);

analogWrite(8, vel);

analogWrite(9, vel);

if(x05<1470){ direita();}

else{ esquerda();}

}

//======================================Config Channel 02================================

if(x02<=1320){ //Recuo - Pulso diminui

vel = map(x02, 1320, 986, 0, 127);

analogWrite(8, vel);

analogWrite(9, vel);

if(x05<1470){ recuo();}

else{ avanco();}

}

if(x02 >1390){ //Avanco - Pulso aumenta

vel = map(x02, 1390, 1851, 0, 127);

analogWrite(8, vel);

analogWrite(9, vel);

if(x05<1470){ avanco();}

else{ recuo();}

}

}// end void

//==================================== Controle do Motor ====================================

void avanco() {

digitalWrite(4, HIGH);

digitalWrite(5, LOW);

digitalWrite(6, HIGH);

digitalWrite(7, LOW);

}

void recuo() {

digitalWrite(4, LOW);

digitalWrite(5, HIGH);

digitalWrite(6, LOW);

digitalWrite(7, HIGH);

}

void esquerda() {

digitalWrite(4, LOW);

digitalWrite(5, HIGH);

digitalWrite(6, HIGH);

digitalWrite(7, LOW);

}

void direita(){

digitalWrite(4, HIGH);

digitalWrite(5, LOW);

digitalWrite(6, LOW);

digitalWrite(7, HIGH);

}

void freio() {

digitalWrite(4, HIGH);

digitalWrite(5, HIGH);

digitalWrite(6, HIGH);

digitalWrite(7, HIGH);

}

Funcionamento da programação:

O principal objetivo dessa programação e ler os dois pulsos obtidos nos pinos 02 e 03 do Arduino e armazená-los nas variáveis x01 e x02, e assim informar qual a função que o Arduino deve executar de acordo com o valor obtido.

Primeiramente o Arduino lê o pulso PWM provindo dos canais CH1 e CH2 do receptor através da função “pulseIn()”, e salva os valores obtidos nas variáveis x01 e x02. O valor obtido na variável x01 será o sinal do Stick direito horizontal e na x02 será o sinal do Stick esquerdo vertical.

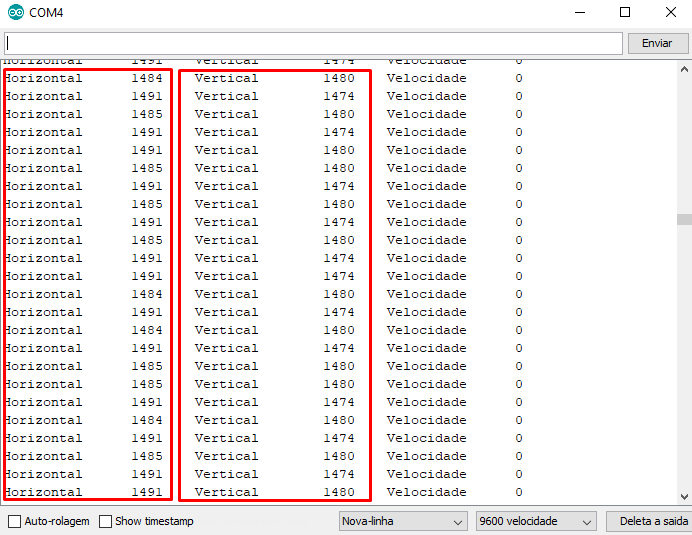
Com o valor dos dois pulsos adquiridos, iremos informar na programação qual a movimentação que o motor irá realizar, através das condições.

Dentro de cada condição possui uma função de mapeamento onde transformara a variação do pulso do ponto mínimo entre ponto máximo, no valor de 0 a 127, e armazenara o valor obtido na variável “vel”, onde será utilizada para controlar a velocidade dos motores pelos pinos 8 e 9 gerando um pulso PWM.

Normalmente usamos o mapeamento de 0 a 255, pois o pino do Arduino e de 8 bits, como iremos utilizar uma fonte de 12v e a voltagem do motor e de 6v, limitaremos a voltagem pela metade utilizado a função map de 0 a 127.

Configurando o controle com a programação:

Com o console do Arduino aberto e o circuito montado, verificar os seguintes pulsos.



Obter o valor PWM do canal 01

Pulso com o Stick para esquerda = pulse\_esq

Pulso com o Stick para direita = pulse\_dir

Menor pulso com Stick parado - 10 Hz = pulse\_neutro\_min01

Maior pulso com o Stick parado + 10 Hz = pulse\_neutro\_max01

Obter o valor PWM do canal 02

Pulso com o Stick para cima = pulse\_avanco

Pulso com o Stick para baixo = pulse\_recuo

Menor pulso com o Stick parado - 10 Hz = pulse\_neutro\_min01

Maior pulso com o Stick parado + 10 Hz = pulse\_neutro\_max01

Após isso, inserir o valor obtido na programação, e seu controle estará configurado.