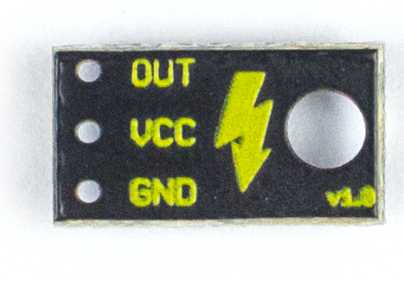
# RELATÓRIO DE TESTES REALIZADOS

Foram realizados testes no laboratório, como para verificar os funcionamentos dos motores, dos sensores QRE113GR da *Robocore*.

Com relação aos sensores, representados na Figura 1, eles possuem três terminais OUT (sinal da saída do sensor) GND (aterramento) e VCC (alimentação), foi testada sua faixa de sinal e seu funcionamento. Todos os sensores funcionaram, possuindo faixa de sinal de 0 até 1023. Cores brancas tiveram a leitura inferior que 100, enquanto cores pretas tiveram a leitura superior a 900. Os sensores possuem corrente de funcionamento de 50 mA e tensão de 5 V.

Figura 1 - Sensor QRE113 Robocore



Fonte: Disponível em: <<https://www.robocore.net/sensor-robo/sensor-de-linha-qre-analogico>>, 2024.

Já para os motores com caixa de redução, representados na Figura *2*, tendo tensão de operação de 3 a 9 V, peso de 15 g, rotação de 300 rpm, corrente de 1,6 A e Torque de 1,7 kgf\*cm (aproximadamente 0,167 N\*m) caso a tensão de operação seja de 6V, foi realizado o teste de giro deles com o uso da ponte H “L298N”

Figura 2 - Motores 6V



Fonte: Disponível em: <<https://www.robocore.net/motor-motoredutor/micro-motor-com-caixa-de-reducao-6v-300rpm>>, 2024.

Foi utilizado a seguinte programação como teste para realização do giro do motor, com uma velocidade controlável, através da ponte H citada.

/\*

\* Programa: Exemplo de uso do driver Ponte H L298N

\* Autor: Gustavo Nery, Eletrogate.

\* Arduino 1.8.12

\*/

// Iremos fazer uma classe para facilitar o uso da ponte H L298N na manipulação dos motores na função Setup e Loop.

class DCMotor {

//Classe criada para conter todos os dados dos motores

//Definição de variáveis

int spd = 255, pin1, pin2; //tipo inteiras, de 0 até 255

//parte aberta, todos no programa podem acessar

public:

void Pinout(int in1, int in2){ // Pinout é o método para a declaração dos pinos que vão controlar o objeto motor

pin1 = in1;

pin2 = in2;

pinMode(pin1, OUTPUT);

pinMode(pin2, OUTPUT);

}

void Speed(int in1){ // Speed é o método que irá ser responsável por salvar a velocidade de atuação do motor

spd = in1;

}

void Forward(){ // Forward é o método para fazer o motor girar para frente

analogWrite(pin1, spd);

digitalWrite(pin2, LOW);

// IN1 = 1 e IN2 = 0, significa que o Motor girará para frente, nesse //caso como é spd no lugar de IN1 = 1, significa que ele girará com uma //certa velocidade para frente

}

void Backward(){ // Backward é o método para fazer o motor girar para trás

digitalWrite(pin1, LOW);

analogWrite(pin2, spd);

// IN1 = 0 e IN2 = 1, significa que o Motor girará para trás, nesse caso //como é spd no lugar de IN2 = 1, significa que ele girará com uma certa //velocidade para trás

}

void Stop(){ // Stop é o metodo para fazer o motor ficar parado.

digitalWrite(pin1, LOW);

digitalWrite(pin2, LOW);

//IN1 = 0 e IN2 = 0, significa que o motor fica parado

}

};

DCMotor Motor1, Motor2; // Criação de dois objetos motores, já que usaremos dois motores, e eles já estão prontos para receber os comandos já configurados acima.

//rotina padrão do sistema, só rodará 1 vez

void setup() {

// Seleciona os pinos que cada motor usará, como descrito na classe.

Motor1.Pinout(5,6);

Motor2.Pinout(9,10);

}

//rotina padrão do sistema, rodara infinitas vezes

void loop() {

// A velocidade do motor pode variar de 0 a 255, onde 255 é a velocidade //máxima, já que a variável para velocidade é uma variável inteira.

Motor1.Speed(200);

Motor2.Speed(200);

Motor1.Forward(); // Comando para o motor ir para frente

Motor2.Forward();

delay(1000);

Motor1.Stop(); // Comando para o motor parar

Motor2.Stop();

delay(500);

Motor1.Backward(); // Comando para o motor ir para trás

Motor2.Backward();

delay(1000);

Motor1.Stop(); // Comando para o motor parar

Motor2.Stop();

delay(500);

}