

THALLYTA CRISTINA PEDROSO

**Acionamento remoto do braço robótico *OWI. Inc. Robotic Arm Edge* utilizando
comunicação via Bluetooth**

Thallyta Cristina Pedroso

**Acionamento remoto do braço robótico *OWI, Inc. Robotic Arm Edge* utilizando
comunicação via Bluetooth**

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

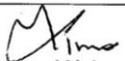
Orientador: Prof. Dr. Leonardo Mesquita

P372a Pedroso, Thallyta Cristina
Acionamento remoto do braço robótico OWI. Inc. Robotic Arm Edge
utilizando comunicação via Bluetooth / Thallyta Cristina Pedroso –
Guaratinguetá, 2018.
58 f : il.
Bibliografia: f. 57-58

Trabalho de Graduação em Engenharia Elétrica – Universidade
Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2018.
Orientador: Prof. Dr. Leonardo Mesquita

1. Robótica. 2. Arduino (Controlador programável). 3. Motores
elétricos. I. Título.

CDU 007.52


Luciana Máximo

Bibliotecária/CRB-8 3595

THALLYTA CRISTINA PEDROSO

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO
PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE
“GRADUADO EM ENGENHARIA ELÉTRICA”

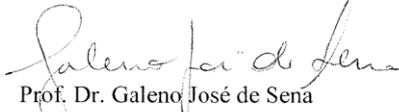
APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE
GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Prof. Dr. LEONARDO MESQUITA
Coordenador do Curso de Engenharia Elétrica

BANCA EXAMINADORA:


Prof. Dr. Leonardo Mesquita
Orientador/UNESP-FEG


Prof. Dra. Paloma Maria Silvia Rocha Rizol
UNESP-FEG


Prof. Dr. Galeno José de Sena
UNESP-FEG

Março de 2018

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus, fonte da vida e da graça. Agradeço pela minha vida, minha inteligência, minha família e meus amigos,

ao meu orientador, *Prof. Dr. Leonardo Mesquita* que jamais deixou de me incentivar. Sem a sua orientação, dedicação e auxílio, o estudo aqui apresentado seria praticamente impossível.

ao meu pai *Jairo Pedroso*, que apesar das dificuldades enfrentadas, sempre incentivou meus estudos.

às funcionárias da Biblioteca do Campus de Guaratinguetá pela dedicação, presteza e principalmente pela vontade de ajudar,

aos funcionários da Faculdade de Engenharia do Campus de Guaratinguetá pela dedicação e alegria no atendimento.

RESUMO

Acionamento remoto do braço robótico *OWI, Inc. Robotic Arm Edge* utilizando quatro modelos distintos de hardware para a mesma aplicação, partindo de um modelo simples, puramente mecânico até o modelo final, onde o braço é acionado através de uma interface *Android*, utilizando uma comunicação *Bluetooth* e tendo o Arduino como elemento central de processamento. No aplicativo para *Android* o usuário será capaz de controlar quatro graus de liberdade do braço robótico, bem como a velocidade do movimento em cada um destes pontos.

PALAVRAS-CHAVE: Acionamento remoto. motor CC. comunicação via Bluetooth.

ABSTRACT

Remote actuation of the robotic arm *OWI Robotic Arm Edge, Inc.* using four different models of hardware for the same application, assuming a simple model, purely mechanical until the final model, where the arm is engaged through an Android interface, using a Bluetooth communication and the Arduino is used as the central element of processing. In the Android app the user will be able to control four degrees of freedom of the robotic arm, as well as the speed of the movement in each one.

KEYWORDS: Remote actuation. DC motor. Bluetooth communication.

SUMÁRIO

1	INTRODUÇÃO	7
1.1	MOTOR DE CORRENTE CONTÍNUA	8
1.2	BRAÇO ROBÓTICO <i>OWI, INC. ROBOTIC ARM EDGE</i>	9
1.3	PONTE H	11
1.4	ARDUINO.....	13
2	ACIONANDO O MOTOR DC COM PONTE H.....	15
2.1	ACIONAMENTO ORIGINAL DO BRAÇO <i>OWI, INC. ROBOTIC ARM EDGE</i> ...	15
2.2	PONTE H UTILIZANDO CHAVES ELETRÔNICAS	18
2.3	DRIVER L298 PARA MOTOR DC	23
2.4	DRIVER L293D PARA MOTOR DC	32
3	CONCLUSÃO	50
	REFERÊNCIAS	51

1 INTRODUÇÃO

Com o advento da tecnologia a presença de robôs no campo industrial tem sido cada vez mais necessária, uma vez que sua atuação torna o processo mais rápido, eficiente e traz mais qualidade ao produto final. Em locais onde o processo possui níveis consideráveis de periculosidade, o acionamento de modo remoto dessas máquinas se torna não só vantajoso, mas também indispensável.

O trabalho traz a proposta de se realizar o acionamento remoto do braço robótico *OWI, Inc. Robotic Arm Edge* utilizando quatro modelos distintos de hardware para a mesma aplicação, partindo de um modelo simples, puramente mecânico até o modelo final, onde o braço é acionado através de uma interface *Android*, utilizando uma comunicação *Bluetooth* e tendo o *Arduino* como elemento central de processamento. No aplicativo para *Android* o usuário será capaz de controlar quatro graus de liberdade do braço robótico, bem como a velocidade do movimento em cada um destes pontos.

Esse conceito de acionamento remoto possibilita a integração com as novas tendências do mercado industrial, tais como sistemas *ciber-físicos*, que são sistemas físicos complexos que se comunicam com o mundo digital, no qual sensores são responsáveis pela captura dos dados dos sistemas, e uma rede inteligente processa essa amostragem de dados e define a ação a ser tomada mediante o quadro apresentado e, em seguida, esse comando é transmitido ao sistema, que pode ser o acionamento de uma válvula em uma caldeira, do movimento de uma esteira em uma linha de produção ou ainda de um braço robótico como no caso a ser apresentado nesse projeto. Essa metodologia permite que o desempenho do sistema seja otimizado e sua eficiência melhorada.

Como o projeto envolve vários conceitos, a introdução e exploração de cada conceito será feito em etapas, partindo de uma concepção simples de acionamento de motores de corrente contínua, seguindo com a explanação desse acionamento com controle de velocidade e chegando ao acionamento de motores de corrente contínua com controle de velocidade através de um controle digital em um aplicativo *Android*, o qual utiliza comunicação via *Bluetooth* para o acionamento.

A seguir são apresentadas as características principais de todos os elementos de hardware necessários para a execução deste trabalho.

1.1 MOTOR DE CORRENTE CONTÍNUA

Segundo Franca (2001) um motor de corrente contínua (do inglês *direct current motor* - motor DC) converte energia elétrica, proveniente de uma fonte de tensão contínua, em energia mecânica.

Os principais componentes de um motor de corrente contínua são:

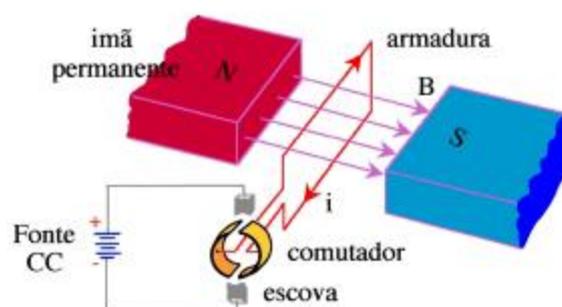
- **Estator:** enrolamento responsável por produzir o campo magnético, que é alimentado diretamente por uma fonte de tensão contínua, sendo que no caso de pequenos motores, pode ser um simples ímã permanente;

- **Rotor:** enrolamento, chamado armadura, que é alimentado por uma fonte de tensão contínua através do comutador e escovas de grafite;

- **Comutador:** dispositivo mecânico, ao qual se encontram conectados os terminais das espiras da armadura; sua finalidade é inverter o sentido da corrente contínua que circula na armadura.

A Figura 1 mostra a estrutura básica de um motor CC elementar com ímã permanente no estator.

Figura 1: Estrutura básica de um motor DC.



Fonte: Franca (2001)

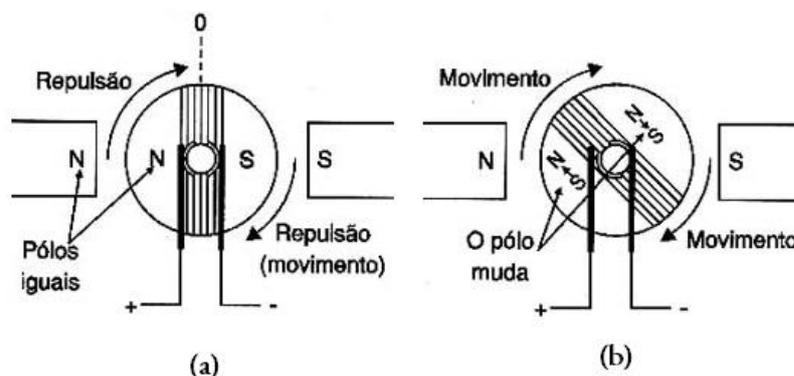
O funcionamento dos motores CC baseia-se no princípio do eletromagnetismo clássico pelo qual um condutor carregando uma corrente e mergulhado em um fluxo magnético fica submetido a uma força eletromagnética.

Se fizermos passar corrente elétrica por duas bobinas próximas, os campos magnéticos criados produzirão forças de atração ou repulsão. A ideia básica de um motor é montar uma

bobina entre os polos de um ímã permanente ou então de uma bobina fixa que funcione como tal. Braga (2009).

Partindo então da posição inicial, em que os polos da bobina móvel (rotor), percorrida por uma corrente, estão alinhados com o ímã permanente, temos a manifestação de uma força de repulsão. Esta força faz o conjunto móvel mudar de posição, conforme mostra a Figura 2(a). A tendência do rotor é dar meia volta para que seu polo Norte se aproxime do polo Sul do ímã permanente. Da mesma forma, seu polo Sul se aproximará do polo Norte pelo qual será atraído. No entanto, no eixo do rotor, por onde passa a corrente que circula pela bobina, existe um comutador. A finalidade deste comutador é inverter o sentido da circulação da corrente na bobina, fazendo com que os polos mudem conforme a Figura 2(b). Braga (2009).

Figura 2: Modelo de um motor DC



Fonte: Braga (2009).

O resultado disso será uma transformação da força de atração em repulsão, o que fará com que o rotor continue seu movimento, buscando sua posição de equilíbrio. Evidentemente isso nunca vai acontecer, e enquanto houver corrente circulando pela bobina o rotor não irá parar. Braga (2009).

1.2 BRAÇO ROBÓTICO OWI, INC. ROBOTIC ARM EDGE

O OWI, Inc. Robotic Arm Edge, Figura 3, é um braço robótico que muito se assemelha a braços robóticos usados industrialmente, como por exemplo os robôs que o Centro de Carreira Chester County usa em sala de aula no ensino de linha de montagem industrial automatizada. Hirschy (2013).

Figura 3 – Braço Robótico OWI, Inc. Robotic Arm Edge

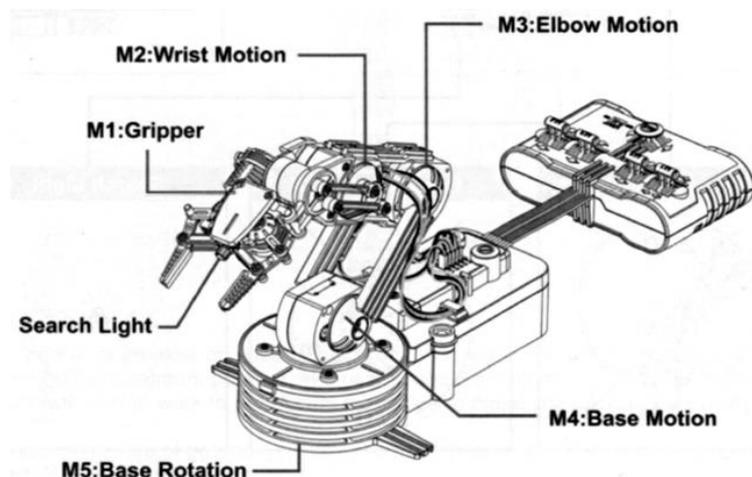


Fonte: Hirschy (2013).

O Braço Robótico *OWI, Inc. Robotic Arm Edge* possui uma garra que executa a função “abrir e fechar”, representada na Figura 4 pelo motor M1; um punho articulado que possui em deslocamento de 120 graus, motor M2; um cotovelo articulado de deslocamento de 300 graus, motor M3; uma base de deslocamento angular de 270 graus, motor M4, e uma de deslocamento angular de 180 graus, motor M5. Esses cinco motores do braço robótico *OWI, Inc. Robotic Arm Edge* oferecem ao mesmo cinco graus de liberdade.

Todos os motores do braço robótico *OWI, Inc. Robotic Arm Edge* são motores DC com alimentação de 3V. A corrente elétrica contínua máxima suportada por esses motores é de 255 mA e a máxima corrente de pico (corrente de *stall*), que é a corrente equivalente demandada pelo motor quando seu eixo é travado, é de 3170 mA.

Figura 4: Representação dos graus de liberdade do braço robótico *OWI, Inc. Robotic Arm Edge*



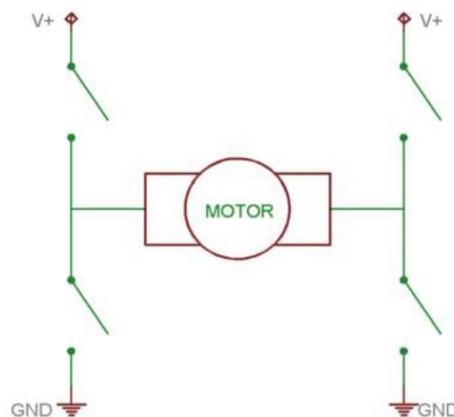
Fonte: Hirschy (2013).

O alcance vertical do *OWI, Inc. Robotic Arm Edge* é de 38,1 centímetros e o alcance horizontal de 32 centímetros, com uma capacidade de elevação de 100g.

1.3 PONTE H

A ponte H é um circuito eletrônico utilizado para controlar o sentido de rotação de um motor DC. Uma ponte H básica é composta por 4 chaves mecânicas ou eletrônicas posicionadas formando a letra “H”, do que originou seu nome, sendo que cada chave localiza-se em um extremo e o motor é posicionado no meio. A Figura 5 apresenta o esquemático elétrico de uma ponte H. Patsko (2006).

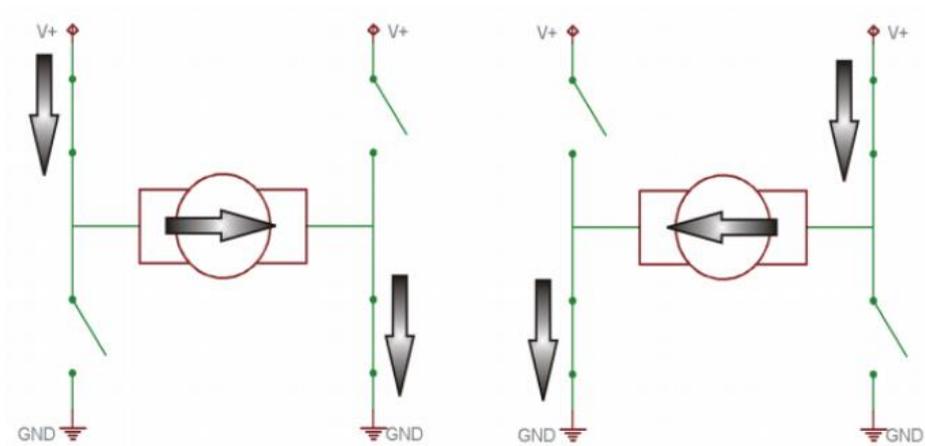
Figura 5 – Esquema elétrico de uma ponte H básica



Fonte: Patsko (2006).

Para que o motor gire é preciso que duas chaves diagonalmente opostas sejam fechadas, fazendo com que a corrente flua do polo positivo para o negativo, passando pelo motor e fazendo-o girar em um determinado sentido. Para inverter a rotação, é preciso abrir essas chaves e fechar o outro par de chaves. Ao fazer isso, a corrente segue na direção oposta e, conseqüentemente, inverte o sentido da rotação do motor. A Figura 6 apresenta a inversão do sentido da corrente em uma ponte H.

Figura 6 – Ilustração do fluxo de corrente elétrica sobre o motor em cada modo de operação da ponte H.



Fonte: Patsko (2006).

Caso se deseje parar o motor por completo, há três maneiras:

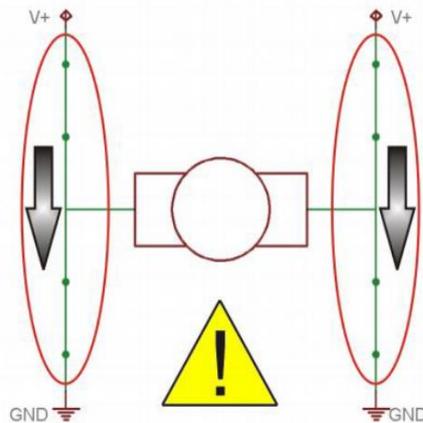
- 1) Fechar ambas as chaves superiores da ponte H
- 2) Fechar ambas as chaves inferiores da ponte H
- 3) Abrir todas as chaves da ponte H

Os métodos 1 e 2 efetivamente põem os terminais do motor em curto-circuito (em um mesmo potencial), fazendo com que a corrente gerada pelo campo magnético resultante do motor circule por ele mesmo de modo a tender girar no sentido oposto ao sentido de quando estava ligado, fazendo-o parar rapidamente. Este modo de parada é chamado de freio dinâmico.

O método 3 fará com que toda corrente circulante pelo motor anteriormente pare instantaneamente, porém o motor reduzirá sua velocidade até zero de forma lenta, pois contará apenas com a força de atrito do rotor ao invés de uma corrente reversa de descarga como no freio dinâmico. Tal modo de parada é chamado de *coasting*.

Um cuidado muito importante que se deve ter é o não acionar as chaves de um mesmo lado da ponte “H” simultaneamente, conforme a Figura 7. Esse efeito é conhecido como *Shoot-through* e faz com que o fluxo de corrente vá direto do pólo positivo para o negativo, causando um curto-circuito na fonte de alimentação, o que gera correntes muito altas e que podem danificar componentes do circuito. Patsko (2006).

Figura 7 – “Efeito *Shoot-through*”: curto-circuito da fonte de alimentação causado pelo chaveamento indevido da ponte H.



Fonte: Patsko (2006).

1.4 ARDUINO

O Arduino, Figura 8, é como um pequeno computador no qual o usuário pode programar para processar entradas e saídas entre o dispositivo e os componentes externos conectados a ele, ou seja, possibilitando a interação com o ambiente externo. MCROBERTS (2011).

Figura 8: Arduino Uno



Fonte: Mcroberts (2011).

A placa do Arduino é uma tecnologia de hardware e o software de fonte aberta e é composta de um microprocessador Atmel AVR, um cristal ou oscilador (relógio simples que envia pulsos de tempo em uma frequência especificada, para permitir sua operação na velocidade correta) e um regulador linear de 5 volts. Mcroberts (2011).

Há muitas variantes diferentes do Arduino. A versão que será utilizada é o Arduino Uno. Dependendo do modelo do Arduino ele pode possuir uma saída USB, que permite conectá-lo a um computador, possibilitando o *upload* (transferência) ou recuperação dos dados. O Arduino também possui pinos de entrada/saída, pelos quais pode-se conectá-lo a outros circuitos ou sensores. Para programar o Arduino utiliza-se o IDE do Arduino, que é um software livre no qual o usuário escreve o código em linguagem C. O IDE permite que o usuário escreva um programa de computador (*sketches*), ou seja, um conjunto de instruções passo a passo, que após o *upload* para o Arduino, fara com que o mesmo execute as instruções desse programa de forma serial, ou seja, uma por vez. Mcroberts (2011).

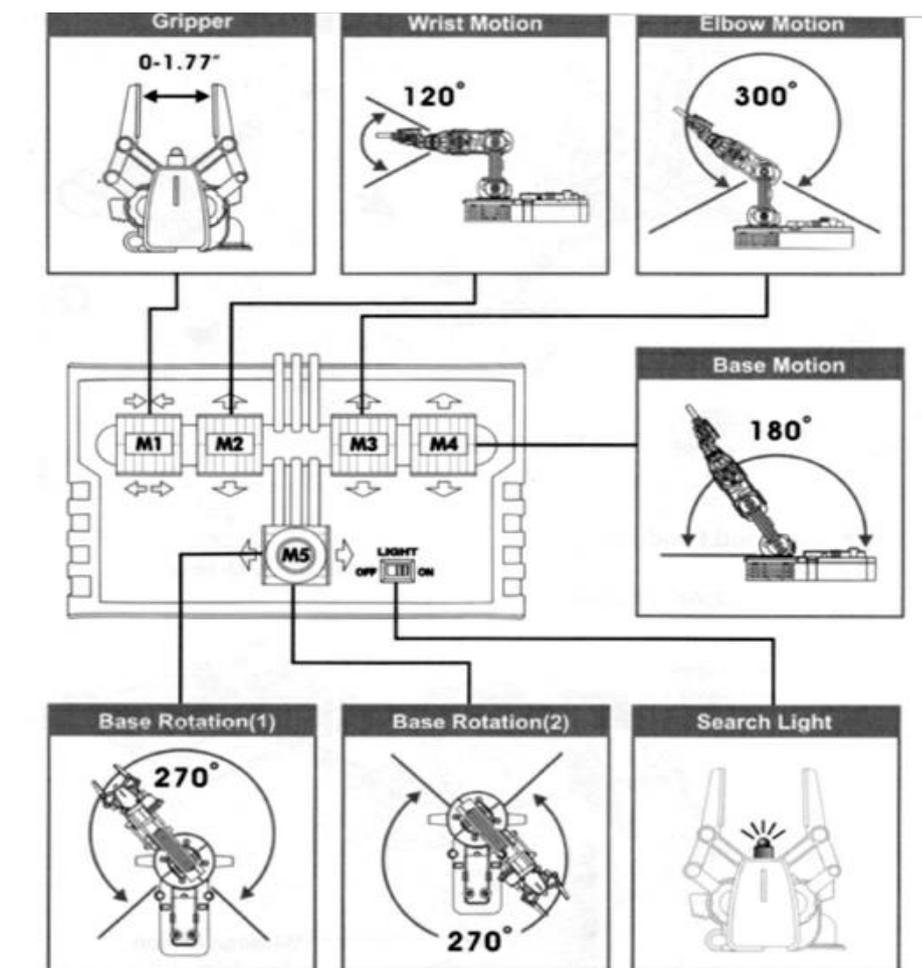
2 ACIONANDO O MOTOR DC COM PONTE H

2.1 ACIONAMENTO ORIGINAL DO BRAÇO OWI, INC. ROBOTIC ARM EDGE

O Braço Robótico *OWI, Inc. Robotic Arm Edge* possui um controle acoplado que aciona os motores e também modifica o seu sentido de rotação, conforme indicado na Figura 9. Esse controle possui cinco chaves do tipo alavanca, as quais são manuseadas pelo usuário. A Figura 10 apresenta o diagrama do circuito elétrico desse controle.

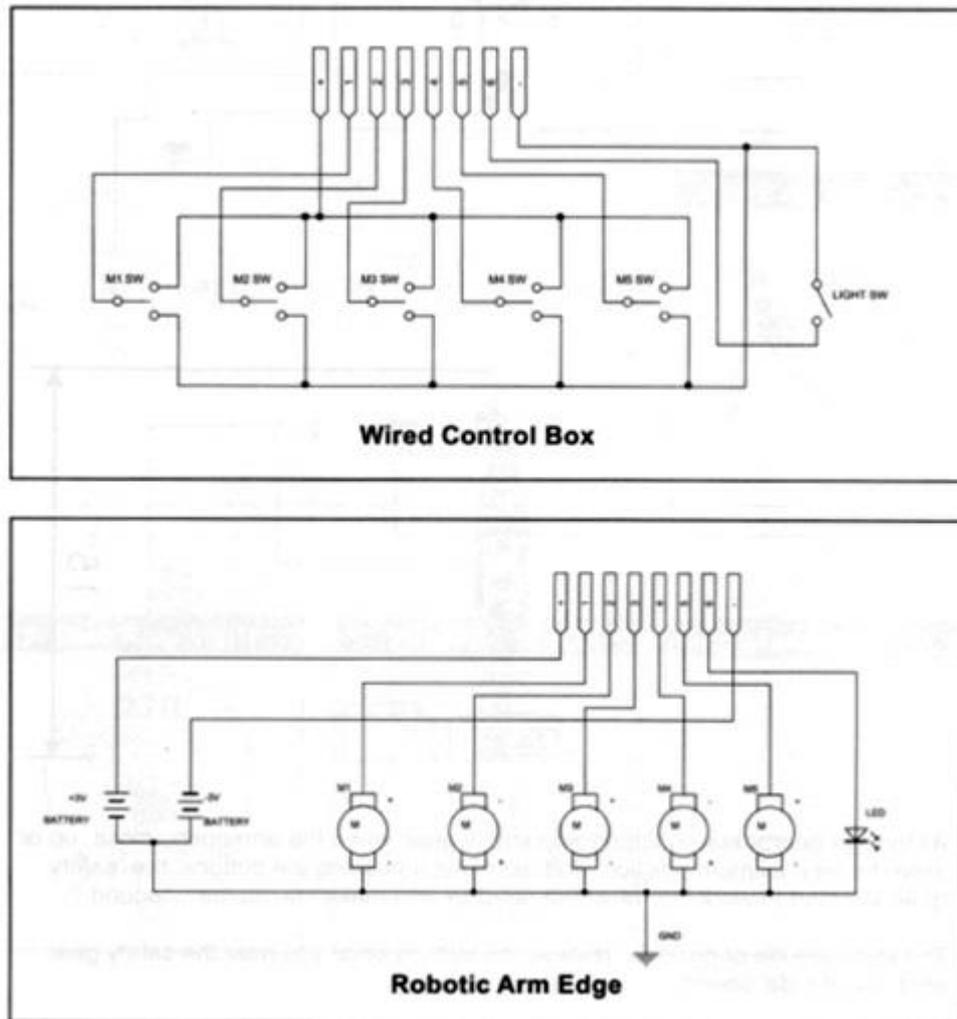
Para se acionar um motor basta mover a chave correspondente a este. Se o usuário desejar alterar o sentido de rotação desse motor, deve-se mover para o outro sentido esta mesma chave.

Figura 9: Controle acoplado ao braço robótico *OWI, Inc. Robotic Arm Edge*



Fonte: Hirschy (2013).

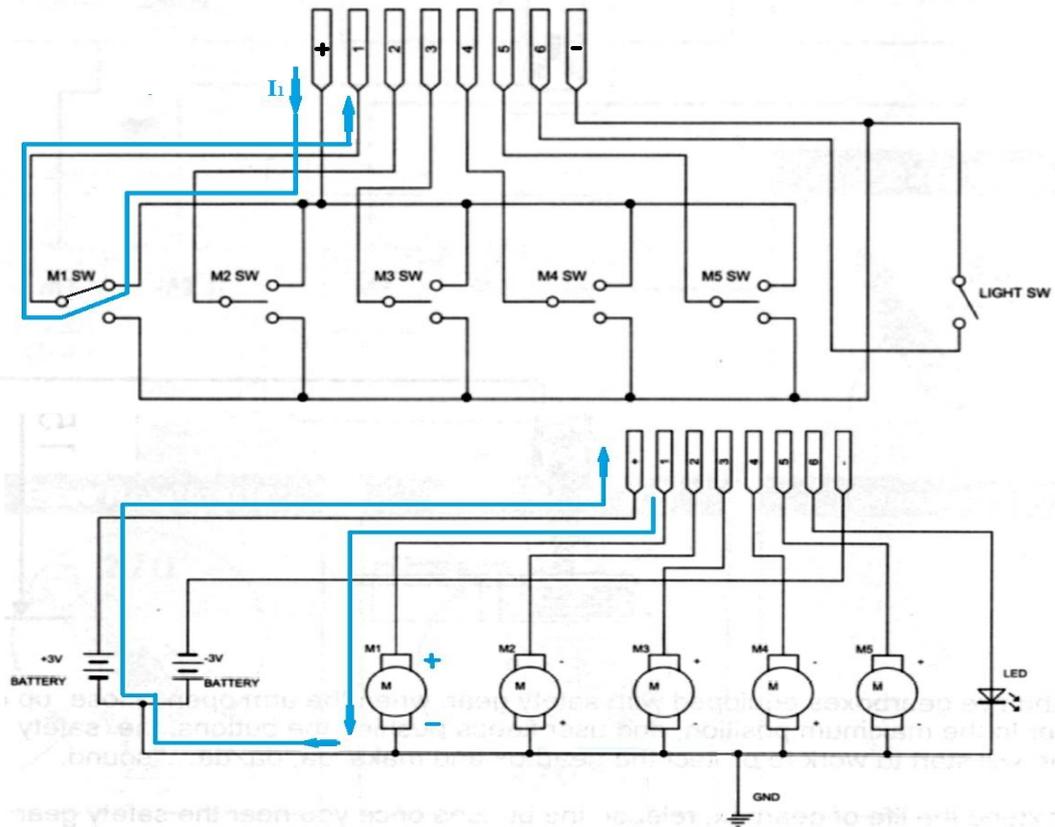
Figura 10: Diagrama do circuito elétrico do controle do braço robótico *OWI, Inc. Robotic Arm Edge*



Fonte: Hirschy (2013).

O funcionamento desse controle é na verdade muito simples. Através da Figura 10 observamos que se o usuário move a chave do motor M1, por exemplo, para cima, a trilha do motor M1 se conecta à fonte positiva de 3V, formando um circuito fechado, o que faz com que a corrente flua pelo motor no sentido apresentado na Figura 11.

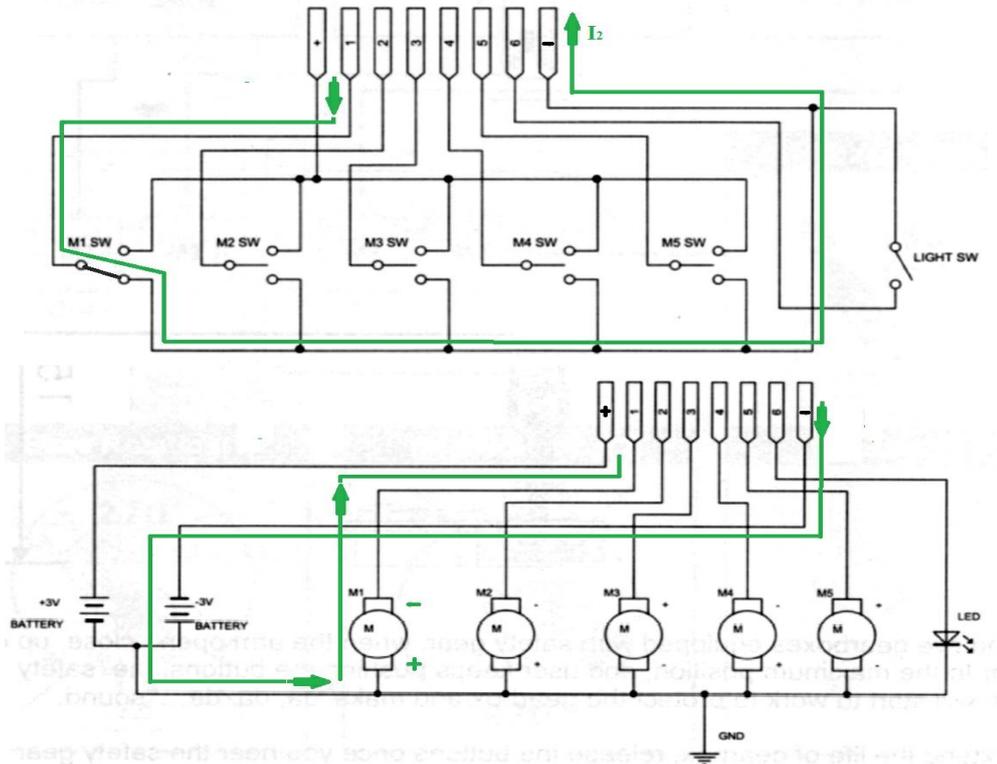
Figura 11: Sentido da corrente que flui pelo motor quando a chave é comutada para cima.



Fonte: Produção do próprio autor

Ao se inverter a posição da chave para baixo, a trilha do motor M1 se conecta à fonte negativa de 3 V, formando um circuito fechado, fazendo com que a corrente flua pelo motor no sentido contrário ao anterior, invertendo o sentido de rotação do motor, conforme ilustrado na Figura 12.

Figura 12: Sentido da corrente que flui pelo motor quando a chave é comutada para baixo.



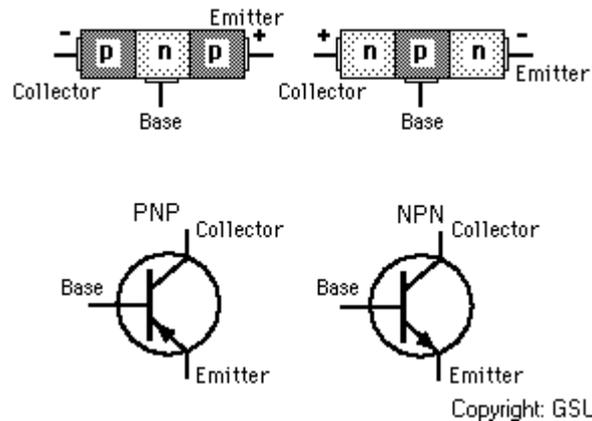
Fonte: Produção do próprio autor

2.2 PONTE H UTILIZANDO CHAVES ELETRÔNICAS

Como foi descrito anteriormente, uma ponte H básica é composta por 4 chaves mecânicas ou eletrônicas posicionadas formando a letra “H”. Para a automatização de um sistema é necessário que as chaves da ponte H sejam eletrônicas, sendo controladas por sinais eletrônicos, que podem ser oriundos de um microcontrolador, por exemplo, que teria o papel de processar os dados do sistema e controlar a ponte H conforme programado.

Um componente eletrônico que funciona como uma chave eletrônica é o transistor, que é um tipo de dispositivo semicondutor formado por três materiais semicondutores, podendo ter uma estrutura NPN ou uma estrutura PNP. A Figura 13 mostra a simbologia de ambas as variações de transistores:

Figura 13 – Símbolo de transistores de junção bipolares NPN e PNP



Fonte: Serodja (2010)

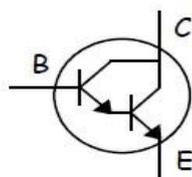
O transistor apresentado na Figura 13 é chamado de transistor bipolar de junção (TBJ), este possui três terminais: o coletor (C), o emissor (E) e a base (B). Em sua aplicação mais básica, o transistor se comporta como uma chave eletrônica, que dependendo da presença ou ausência de corrente na base do transistor, fará com que os terminais entre o coletor e o emissor se fechem ou se abram.

Via de regra, para transistores NPN, se há corrente chegando no terminal de base do transistor, haverá uma tensão (tensão entre base e emissor) positiva no transistor, fechando-se um curto-circuito entre os terminais de coletor e emissor do mesmo e permitindo a passagem de corrente elétrica entre estes terminais. Se não houver corrente na base, a tensão será nula e haverá um circuito aberto entre os terminais de coletor e emissor, fazendo-os ficarem eletricamente isolados. Transistores PNP seguem a seguinte lógica: se houver corrente sendo drenada da base do mesmo, isto implicará em uma tensão negativa, o que fechará um curto-circuito entre os terminais de emissor e coletor. No caso de ausência de corrente sendo drenada da base, a tensão do transistor será praticamente nula e os terminais de coletor e emissor estarão isolados entre si.

No projeto as chaves do esquemático da ponte H apresentado na Figura 5 serão substituídas por componentes eletrônicos que operam como chave, no caso, por transistores TBJ. Como dito anteriormente, o transistor TBJ necessita de uma corrente de base para ser acionado e para este projeto, essa corrente será oriunda do Arduino, que é o microcontrolador. Entretanto, a corrente disponibilizada pelo Arduino em cada porta digital é de 40mA, uma corrente de pequena magnitude. Assim sendo, foi escolhido o transistor TIP120, por ser um transistor do tipo *Darlington*.

O transistor *Darlington* é a junção de dois transistores em um, conforme a Figura 14. Esse transistor pode ser acionado com valores pequenos de corrente elétrica, viabilizando o uso do Arduino, como fornecedor da corrente de acionamento dos transistores.

Figura 14: Símbolo do transistor de junção bipolar *Darlington*



Fonte: Arduinolivre (2013)

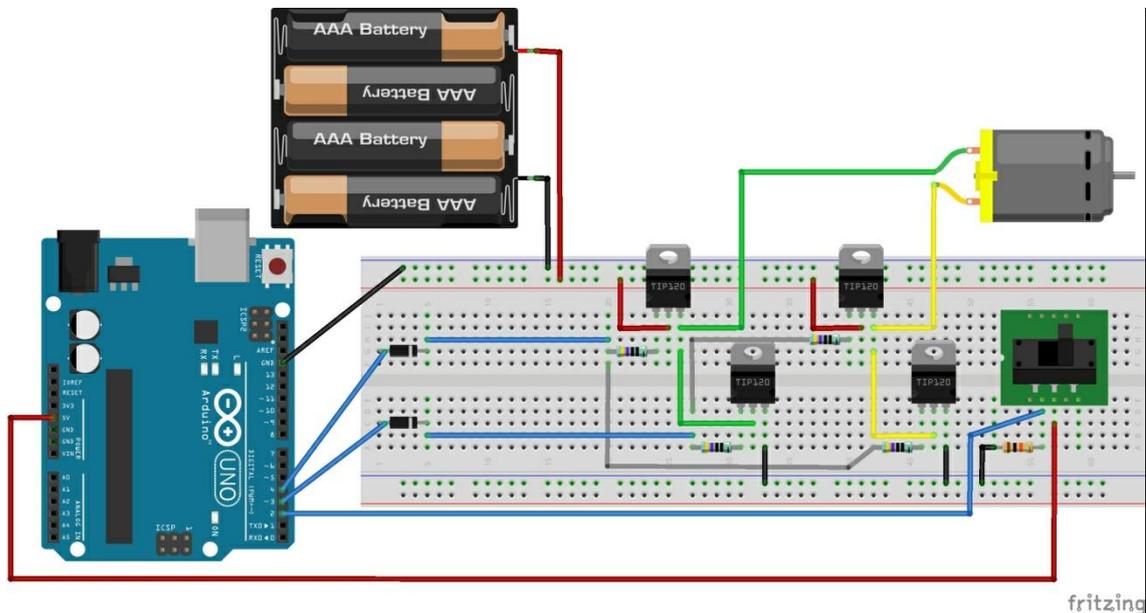
Como foi visto no tópico 3.2, o braço robótico *OWI. Inc. Robotic Arm Edge* pode demandar uma corrente elétrica contínua de até 255mA, o que notoriamente é bem superior à corrente disponibilizada pelas portas digitais do Arduino. Por isso, é importante destacar que o Arduino suprirá somente a corrente necessária para acionar os transistores da ponte H, conhecida como “corrente de sinal”, por apresentar pequenos valores de magnitude. A corrente que será disponibilizada para os motores é oriunda de uma fonte externa ao Arduino, conforme a Figura 15.

Para o modelo do projeto foram utilizados os seguintes materiais:

- 1 Arduino
- 4 Resistores de 470 ohms
- 4 Transistores TIP120
- 2 Diodos 1N4148
- 1 Protoboard
- 1 Bateria 9V
- 1 Conector de bateria
- 1 Motor CC
- 1 chave on/off
- 1 resistor de 10k Ω (pull-down)
- Alguns fios para conexão

O primeiro passo foi montar o circuito apresentado na Figura 15.

Figura 15: Esquemático do circuito da ponte H com transistores TIP120



Fonte: Roboott (2015).

No circuito da Figura 15 podemos observar os transistores do tipo TIP120 ocupando a função das chaves na ponte H. Quando é enviado um comando para mudar o sentido de rotação do motor, essa mudança não ocorre de forma instantânea, devido à inércia do motor. Assim, pode ocorrer a geração de uma corrente elétrica com magnitude suficiente para voltar ao Arduino através de suas portas, e danificá-lo. Por isso, se faz necessário a presença dos dois diodos, que protegerão o Arduino de qualquer corrente de retorno gerada pelos motores. O diodo possui apenas um único sentido de fluxo para a corrente elétrica, a faixa branca indica o lado com a barreira para fluxo e energia. Assim, deve-se usar o diodo no sentido mostrado na Figura 15, com o polo onde tem a faixa branca voltada para o resistor, de forma a impedir a passagem de correntes do circuito para o Arduino.

Após a montagem do circuito da Figura 15, carregou-se o programa no Arduino para fazer o comando do sentido de rotação do motor. A seguir apresenta-se o código utilizado.

```
//Acionamento de Motor DC com ponte H
```

```
int switchPin = 2; //pino onde será conectado a chave
```

```

int motor1Pin1 = 3; //pino onde será conectado o terminal 1 do motor
int motor1Pin2 = 4; //pino onde será conectado o terminal 2 do motor
void setup()
{
    pinMode(switchPin, INPUT); //configura o pino switchPin como entrada
    pinMode(motor1Pin1, OUTPUT); //configura o pino motor1Pin1 como saída
    pinMode(motor1Pin2, OUTPUT); //configura o pino motor1Pin2 como saída
}
void loop()
{
    If(digitalRead(switchPin) == HIGH) //verifica se o pino switchPin está em nível lógico alto
    {
        digitalWrite(motor1Pin1,LOW); //envia sinal lógico baixo para o pino
motor1Pin1
        digitalWrite(motor1Pin2,HIGH); //envia sinal lógico alto para o pino
motor1Pin2
    }
    else
    {
        digitalWrite(motor1Pin1,HIGH);
        digitalWrite(motor1Pin2,LOW);
    }
}

```

No código apresentado há três variáveis do tipo inteiro (int) que correspondem ao número das portas digitais do Arduino que serão utilizadas. A variável *switchPin* recebe o número 2, indicando que toda vez que essa variável for chamada, a mesma terá o valor 2 associado. O mesmo ocorre com as variáveis *motor1Pin1* e *motor1Pin2*, recebendo os valores 3 e 4 respectivamente.

Dentro do bloco *void setup()* é definido se os pinos serão de entrada ou saída de dados. Esse bloco é executado apenas uma vez no programa e tem a função de configurar os parâmetros a serem utilizados pelo mesmo.

No pino *switchPin* está conectada uma chave que comutará entre os valores lógicos 1 e 0, de forma a fazer com que o motor gire em determinado sentido, conforme o estado da chave. O pino *motor1Pin1* refere-se ao sinal oriundo do Arduino que ativará o transistor

necessário para criar o caminho para a corrente na ponte H que fará o motor girar no sentido solicitado pelo usuário através da chave conectada no pino *switchPin*, pino 2.

O bloco *void loop()* será executado em um loop infinito enquanto o programa estiver rodando. A instrução *if(digitalRead(switchPin) == HIGH)* é uma instrução condicional. O comando *digitalRead* fará a leitura do pino *switchPin*, se o valor lido for nível lógico alto (HIGH), ou seja, 5V, o programa executará as instruções dentro das chaves dessa instrução, caso contrário o programa executará os comandos do bloco *else*.

Dentro desse bloco *if*, que será executado caso a condição descrita anteriormente seja verdadeira, o pino *motor1Pin1* recebe sinal lógico baixo (LOW), 0V, e o pino *motor1Pin2* recebe sinal lógico alto, 5V, fazendo com que a corrente flua em determinado sentido pelo motor. No bloco *else* os valores lógicos dos pinos *motor1Pin1* e *motor1Pin2* são invertidos, fazendo com que o sentido da corrente elétrica através do motor de inverta, invertendo assim seu sentido de rotação. Dessa forma o usuário pode inverter o sentido de rotação do motor simplesmente comutando o estado da chave.

2.3 DRIVER L298 PARA MOTOR DC

O *driver* L298 para motores DC é um circuito integrado (CI) que possui duas pontes H em seu interior, podendo controlar dois motores DC. A Figura 16 apresenta a pinagem do *driver* L298 e a Figura 17 seu circuito interno em visão macro.

Figura 16: Pinagem do *driver* L298

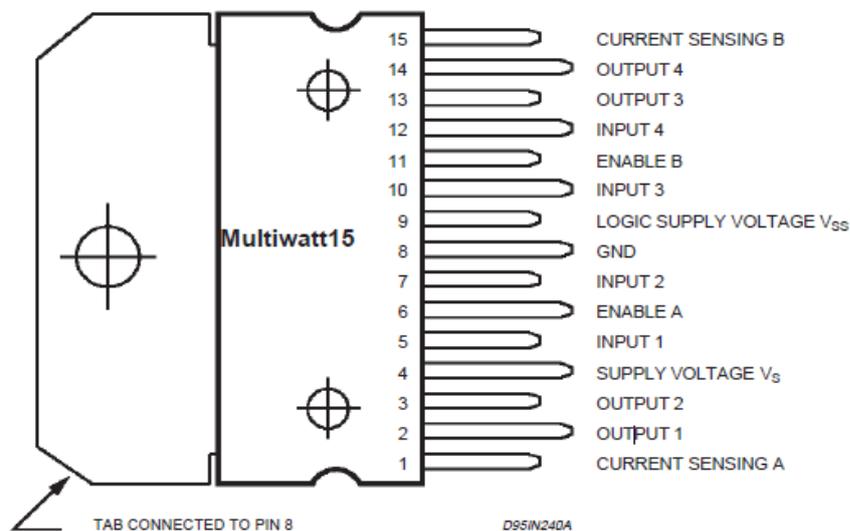
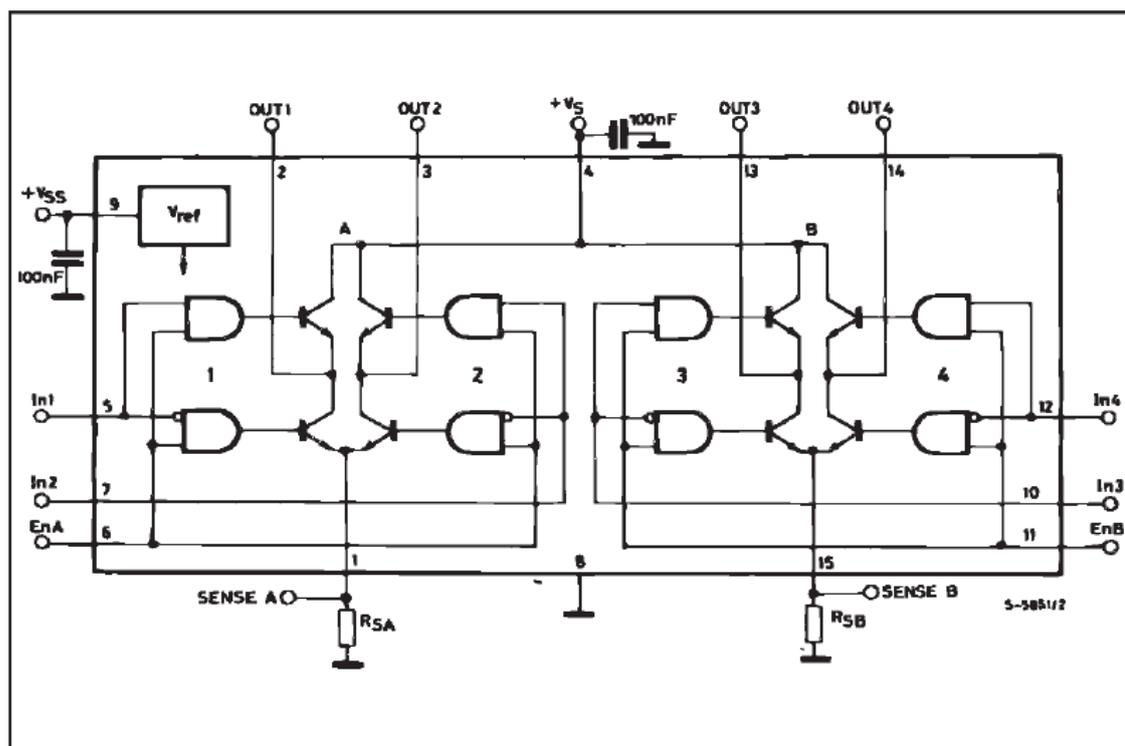


Figura 17: Circuito interno em visão macro do *driver* L298



Fonte: ST Microelectronics (2000)

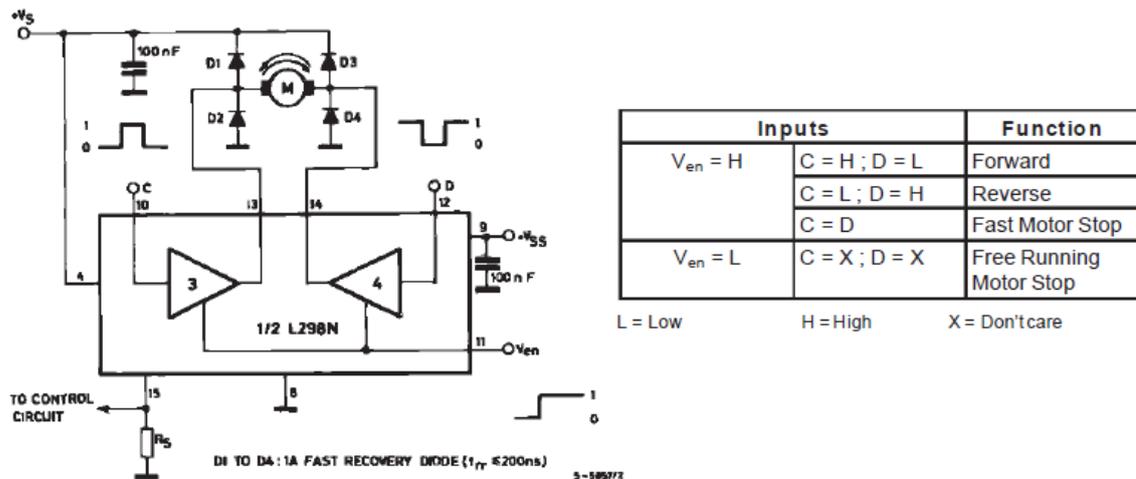
Conforme a Figura 16, verificamos que o *driver* L298 possui 15 pinos. Os pinos 1 e 15 são denominados *Current sense* (sensor de corrente), que possibilitam fazer a leitura da corrente demandada por cada motor. Para se fazer uso desse recurso é necessário conectar a esses pinos um resistor de baixo valor, de forma a não interferir na magnitude da corrente de saída. Para se determinar o valor da corrente é preciso medir a tensão sobre este resistor e calcular a corrente que está sendo fornecida (funciona como um *shunt*). É importante observar que para essa finalidade o resistor utilizado deve possuir uma dissipação de potência elevada, da ordem de 1 a 5 W.

Quanto à alimentação do CI, o pino 4 (*Supply voltage Vs*) é referente a tensão de alimentação para os motores, que pode ser de 5 a 35V, enquanto que o pino 9 (*Logical Supply Voltage Vss*) é referente à tensão do circuito de controle, que no caso do Arduino é de 5V. O pino 8 é referente ao GND e é comum aos dois circuitos.

Os pinos 6 e 11 são denominados de *ENABLE* e tem a função de habilitar ou não cada uma das pontes H. Para se habilitar a ponte H desejada, deve-se manter o pino de *ENABLE* correspondente em nível lógico alto, 5 V.

Os pinos 5, 7, 10 e 12 são referente às entradas do drive L298, denominadas INPUT 1, INPUT 2, INPUT 3 e INPUT 4, respectivamente, conforme Figura 16. Esses pinos de entrada obedecem à lógica apresentada na Figura 18, a qual apresenta apenas uma ponte H do driver L298, pois o funcionamento da outra ponte H em seu interior é análogo.

Figura 18: Controle bidirecional do driver L298 para motor DC



Fonte: ST Microelectronics (2003)

Sendo que a pinagem ilustrada na Figura 17 corresponde ao diagrama de blocos do driver L298 apresentado na Figura 16. Temos assim a relação:

Ven: referente ao pino Enable B

C: referente ao pino INPUT 3

D: referente ao pino INPUT 4

Pino 13: referente ao pino saída OUTPUT 3

Pino 14: referente ao pino saída OUTPUT 4

H: corresponde ao nível lógico alto (HIGH) 5V;

L: corresponde ao nível lógico baixo (LOW) 0V;

X: corresponde ao nível lógico irrelevante, podendo ser HIGH ou LOW.

Os pinos 2, 3, 13 e 14 são referentes, respectivamente, a OUTPUT 1, OUTPUT 2, OUTPUT 3 e OUTPUT 2. Aos pinos 2 e 3 serão conectados os terminais do motor 1 e aos pinos 13 e 14 os terminais do motor 2.

Analisando a Figura 18, podemos verificar que estando o pino Ven em nível lógico alto, e as entradas C e D em nível lógico alto e baixo, respectivamente, resultará na movimentação do motor M (motor 2) em determinado sentido de rotação. Ao invertermos os níveis lógicos dos pinos C e D, resultará na inversão do sentido de rotação do motor. Caso o nível lógico dos pinos C e D sejam iguais, os terminais do motor ficarão em um mesmo potencial, fazendo com que a corrente gerada pelo campo magnético resultante do motor circule por ele mesmo de modo a fazer o motor tender a girar no sentido oposto ao sentido de quando estava ligado, levando-o a uma frenagem rápida.

Resumindo as informações da Figura 18 e ajustando para a nomenclatura dos pinos apresentados na Figura 17, tem-se a Tabela 1, apresentando a lógica de operação das duas pontes H internas ao driver L298.

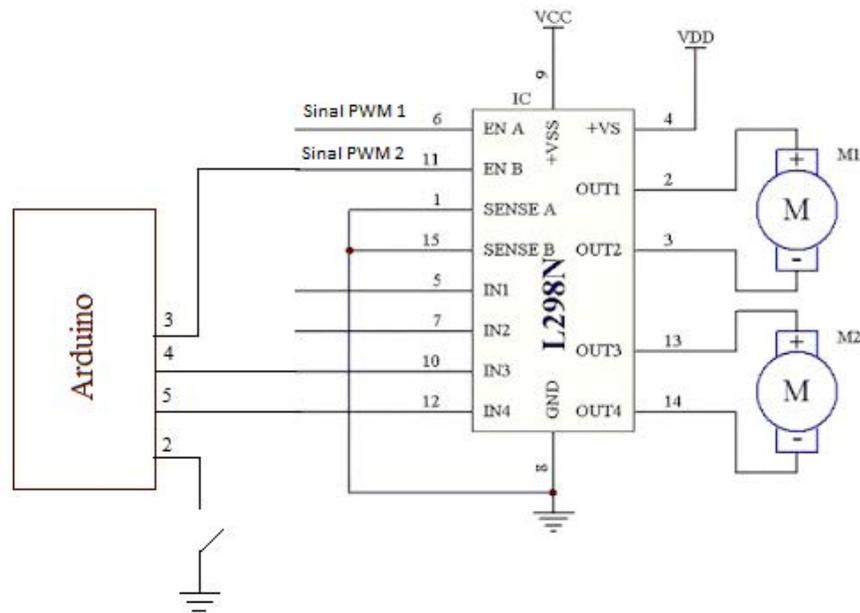
Tabela 1: Lógica aplicada às entradas do driver L298 para o acionamento do motor DC.

Entradas		Função
ENABLE A = H	INPUT 1 = H; INPUT 2 = L	Motor 1 gira em um sentido
	INPUT 1 = L; INPUT 2 = H	Motor 1 gira no sentido inverso
	INPUT 1 = INPUT 2	Frenagem rápida do motor 1
ENABLE A = L	INPUT 1 = X; INPUT 2 = X	Desabilita a ponte H, e para o motor 1
ENABLE B = H	INPUT 3 = H; INPUT 4 = L	Motor 2 gira em um sentido
	INPUT 3 = L; INPUT 3 = H	Motor 2 gira no sentido inverso
	INPUT 3 = INPUT 4	Frenagem rápida do motor 2
ENABLE B = L	INPUT 3 = X; INPUT 4 = X	Desabilita a ponte H, e para o motor 2

Fonte: Produção do próprio autor

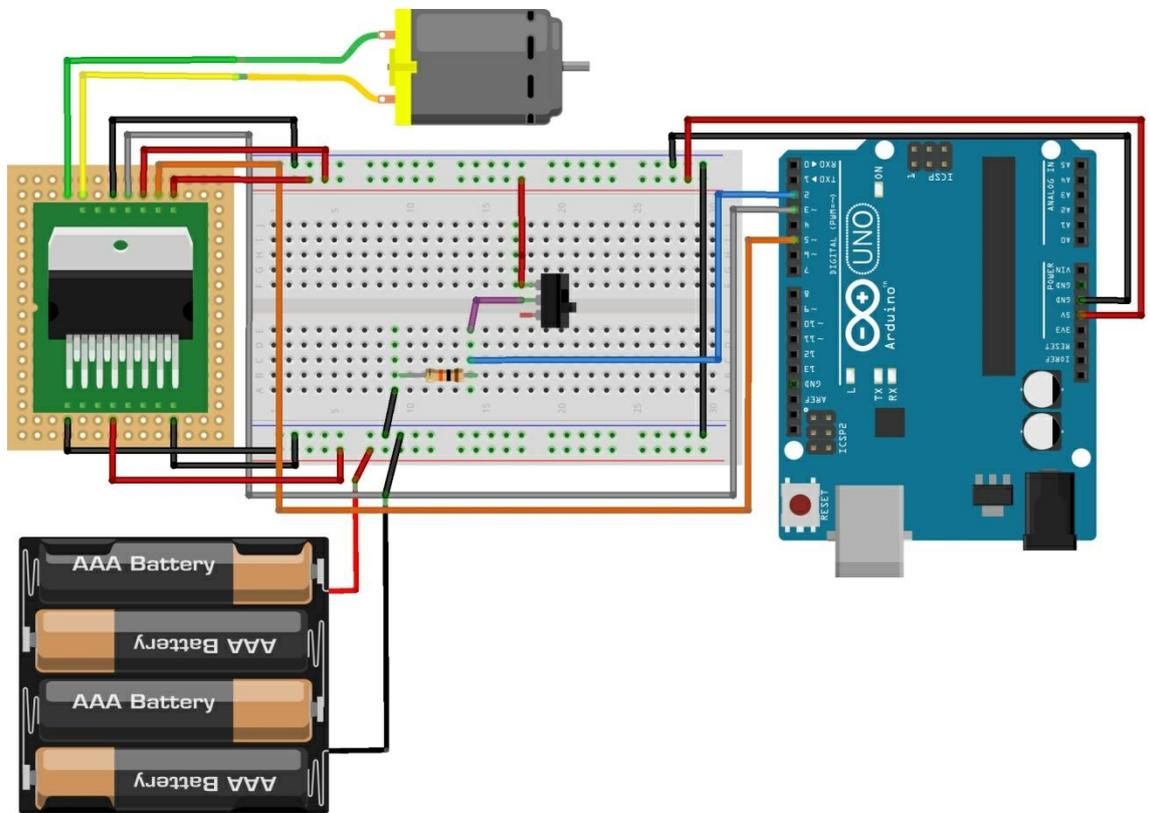
A Figura 19 apresenta o esquemático de montagem do circuito utilizando o *driver* L298 e a Figura 20 a montagem completa.

Figura 19: Esquemático de montagem utilizando o *driver* L298



Fonte: Moraes (2012).

Figura 20: circuito do *driver* L298 sendo controlado pelo Arduino.



fritzing

Fonte: Produção do próprio autor

Após a montagem do circuito da Figura 20, fez-se o *upload* do programa a seguir no Arduino.

```
// Programa: Acionamento de motor CC com o L298D com PWM
```

```
int PinoVelocidade = 3; //Ligado ao pino 11 do L298D
int Entrada1 = 4; //Ligado ao pino 10 do L298D
int Entrada2 = 5; //Ligado ao pino 12 do L298D
int switchPin = 2; // switch input

void setup()
{
  //Define os pinos como saída
  pinMode(PinoVelocidade, OUTPUT);
  pinMode(Entrada1, OUTPUT);
  pinMode(Entrada2, OUTPUT);
}

void loop()
{
  //Define a velocidade de rotação
  int velocidade = 153; // duty cycle de 60%
  analogWrite(PinoVelocidade, velocidade);

  if(digitalRead(switchPin) == HIGH)
  {
    para_motor(); //Chama a rotina de parada do motor
    while(digitalRead(switchPin) == HIGH)
    {
      digitalWrite(Entrada1, LOW);
      digitalWrite(Entrada2, HIGH);
    }
  }
}
```

```

else
  { //Aciona o motor no sentido inverso
  para_motor();
  while(digitalRead(switchPin) == LOW)
  {
    digitalWrite(Entrada1, HIGH);
    digitalWrite(Entrada2, LOW);
  }
}

void para_motor()
{
  digitalWrite(Entrada1, LOW);
  digitalWrite(Entrada2, LOW);
  delay(1000);
}

```

Nessa etapa, além de controlar o sentido de rotação do motor, foi feito também o controle de velocidade do mesmo através da Modulação por Largura de Pulso (PWM) do inglês *Pulse Width Modulation*. Esse método consiste em ligar e desligar o motor, numa frequência fixa, através de uma chave eletrônica, no caso um transistor bipolar, fazendo com que o motor gire numa velocidade proporcional à relação entre o tempo ligado (T_{on}) e período (T). Essa relação é chamada de *Duty Cycle* (D). Multiplicando essa relação pelo valor da tensão de pico, que é a tensão de alimentação do motor, teremos uma tensão média equivalente à tensão DC que deveria ser aplicada ao motor para fazê-lo girar à mesma velocidade. Maimon (2004).

Figura 21 – (a) PWM com D próximo de 100%; (b) $D = 50\%$; (c) D próximo de 0%.



Fonte: Maimon (2004).

Na Figura 21, há três formas de ondas geradas com PWM, todas com uma frequência fixa. Na Figura 21(a) a onda possui um PWM com um Duty Cycle de quase 100%, ou seja, o motor se comporta como se estivesse conectado diretamente à tensão de alimentação nominal. Já na Figura 21(b) o motor estaria girando à aproximadamente a metade de sua velocidade máxima, uma vez que a tensão média aplicada ao motor é de 50%. Na Figura 21(c) o motor estaria em uma velocidade muito baixa, sendo aplicado a ele uma tensão média menor que 10%.

A placa Arduino Uno possui 6 pinos de saída PWM, sendo eles 3, 5, 6, 9, 10 e 11. Esses pinos de PWM são indicados pelo caractere ‘~’ na frente de seu número do pino, conforme a Figura 22.

Figura 22 - Saídas PWM na placa Arduino UNO



Fonte: Souza (2014).

Para fazer uso do PWM no Arduino, basta utilizar a função *analogWrite()*, a qual escreve um valor de PWM em um pino digital que possui essa função. A *sintaxe* da função é dada por *analogWrite(pino, valor)*, onde “pino” corresponde ao pino em que se deseja obter um sinal tipo PWM de saída e “valor” corresponde ao valor de *duty cycle* desejado, o qual deve ser de 0 a 255 onde em 0 equivale a um *duty cycle* de 0% e em 255 equivale a um *duty cycle* de 100%.

Quando essa função é chamada, o pino passa a operar com uma onda quadrada de frequência fixa o *duty cycle* conforme valor indicado pelo usuário na função. É importante ressaltar que a frequência dessa onda é aproximadamente de 490 Hz na maioria dos pinos, porém, os pinos 5 e 6 operam em 980 Hz. Outro ponto importante a ser destacado é que para utilizar a função *analogWrite()*, deve-se configurar o pino correspondente como saída digital, uma vez que as saídas do Arduino não são conversores digital-analógico como o nome sugere, e estes pinos não estão relacionados às entradas analógicas. Souza (2014).

No código apresentado há quatro variáveis do tipo inteiro (int) que correspondem ao número das portas digitais do Arduino que serão utilizadas. A variável *switchPin* recebe o número 2, indicando que toda vez que essa variável for chamada, a mesma terá o valor 2 associado. O mesmo ocorre com as variáveis *PinoVelocidade*, *Entrada1*, *Entrada2*, recebendo os valores 3, 4 e 5 respectivamente.

O pino *PinoVelocidade* refere-se ao sinal PWM oriundo do Arduino, que será conectado ao pino 11 do driver L298, que é referente ao pino ENABLE B, e fará a habilitação da ponte H, conforme a Figura 18. Como o sinal ENABLE B está conectado a duas portas lógicas AND, as quais habilitam os transistores inferiores da ponte H interna do L298, quando o sinal do PWM está em nível lógico baixo, ou seja, 0V, a ponte H é desabilitada, o que faz com que o motor funcione apenas os intervalos de tempo em que o sinal PWM encontra-se em nível lógico alto, 5V. Assim, o valor médio de tensão a ser aplicada ao motor DC dependerá do valor de *duty cycle* do sinal PWM.

Assim como no código analisando no tópico anterior, o bloco *void loop()* será executado em um loop infinito. A primeira instrução desse bloco, *int velocidade = 153*, atribui à variável *velocidade* o valor inteiro 153, que representa um valor de *duty cycle*, conforme a equação 1:

$$duty\ cycle = \frac{velocidade}{255} \quad (1)$$

Portanto:

$$duty\ cycle = \frac{153}{255}$$

$$duty\ cycle = 60\%$$

Conforme demonstrado, teremos um valor de *duty cycle* de 60%, o que fará com que o motor gire à 60% de sua velocidade nominal.

Após a variável *velocidade* receber o valor inteiro de 153, a instrução *analogWrite(PinoVelocidade, velocidade)*, habilitará um sinal PWM de *duty cycle* de 60% no pino *PinoVelocidade*, conectado ao pino 6 do L298.

Em seguida encontramos a instrução condicional *if(digitalRead(switchPin) == HIGH)*, na qual o comando *digitalRead* fará a leitura do pino *switchPin* e se o valor lido estiver em nível lógico alto (HIGH), ou seja, em 5V, o programa executará as instruções dentro dos parênteses dessa instrução, caso contrário o programa executará os comandos do bloco *else*.

Dentro desse bloco *if*, que será executado caso a condição descrita anteriormente for verdadeira, é chamada a função *para_motor()*, que, como o nome sugere, fará com que o motor DC pare e garanta um intervalo de 1 segundo de atraso. Essa estratégia foi adotada com o intuito de proteger o motor de mudanças bruscas de sentido de rotação, o que geraria uma corrente reversa de alto pico de magnitude, danificando o motor.

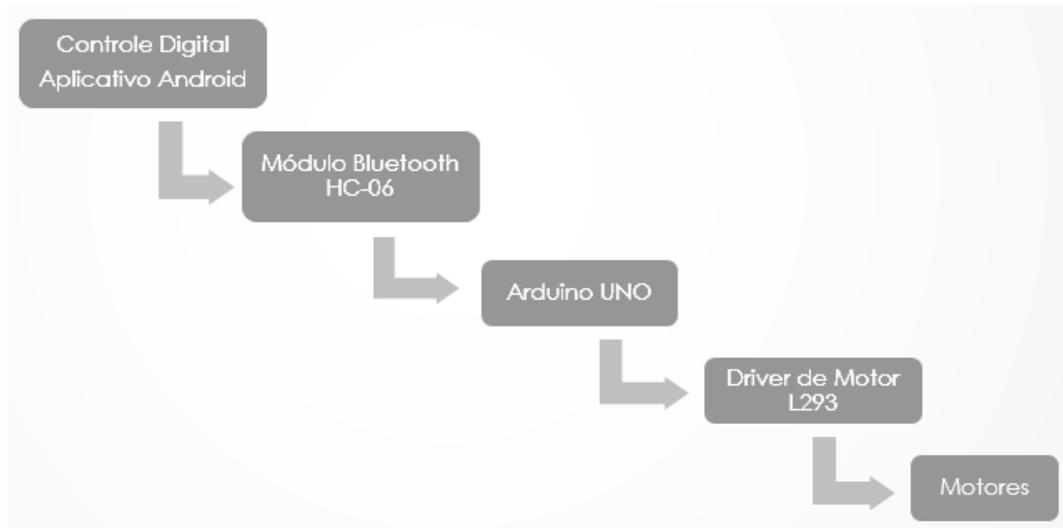
Após a chamada da função *para_motor()*, é executado a instrução *while(digitalRead(switchPin) == HIGH)*. Nessa linha de código a instrução *digitalRead(switchPin) == HIGH* verificará o estado lógico do pino *switchPin*, que está conectado a uma chave que fará a seleção de sentido de rotação do motor, conforme o estado da chave. Se o estado lógico do pino *switchPin* for nível lógico alto, HIGH, serão executadas as instruções *digitalWrite(Entrada1, LOW)* e *digitalWrite(Entrada2, HIGH)*, fazendo que o motor gire em determinado sentido de rotação, e o motor manterá esse movimento até que o estado do pino *switchPin* seja alterado.

Caso o estado lógico do pino *switchPin* for nível lógico baixo, LOW, serão executadas as instruções dentro do bloco *else*, seguindo a mesma lógica do bloco *if*, apenas invertendo o estado das entradas 1 e 2, conforme as instruções *digitalWrite(Entrada1, HIGH)* e *digitalWrite(Entrada2, LOW)*, alterando o sentido de rotação do motor.

2.4 DRIVER L293D PARA MOTOR DC

Nessa etapa além de controlarmos o sentido de rotação do motor DC e a sua velocidade, através da modulação PWM, será feito o uso do aplicativo Android *Bluetooth Electronics*. A Figura 23 apresenta uma visão macro dessa etapa.

Figura 23: Visão macro da etapa utilizando o *driver* L293D para motores DC.



Fonte: Produção do próprio autor

Conforme indicado na Figura 23, essa etapa inicia-se com a aquisição de dados para o acionamento dos motores do braço robótico através do controle no aplicativo Android *Bluetooth Electronics*, que é uma cópia digital do controle mecânico original de acionamento do braço robótico. Após essa aquisição de dados, estes serão enviados para o Arduino através de um módulo *Bluetooth*, no caso HC-06. O Arduino fará então o processamento desses dados e, se assim solicitado, fará o acionamento dos motores através do driver de motor DC L293D.

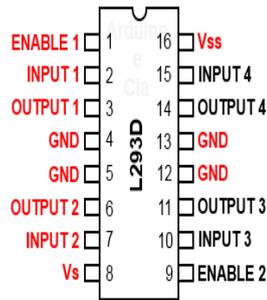
O driver L293D é um CI similar ao driver L298, possuindo duas pontes H em seu interior, porém, com um encapsulamento mais compacto.

Esse componente suporta correntes de saída de 600mA por canal, possibilitando o controle de até 2 motores de 600mA cada. A corrente máxima de pico suportada pelo driver L293D é de 1,2A, sendo recomendado utilizar um dissipador de calor quando se utilizar esse drive para motores que demandem correntes próximas à 600mA. As tensões de alimentação dos motores suportadas pelo drive L293D estão na faixa de 4.5 à 36V.

O driver L293D possui 16 pinos, o diagrama de pinagem do componente pode ser analisado através da Figura 23. O pino 1 é referente ao ENABLE 1, ao qual será conectado o sinal PWM oriundo do Arduino para fazer o controle de velocidade do motor 1 e analogamente o pino 9 refere-se ao ENABLE 2, que por sua vez recebe o sinal PWM para o controle de velocidade do motor 2. Os pinos 2 e 7 (INPUT 1 e INPUT 2), são as entradas para o controle de sentido de rotação do motor 1, conectado aos pinos 3 e 6 (OUTPUT 1 e OUTPUT 2), respeitando a mesma lógica vista na análise do driver L298, ou seja, alternando

os estados LOW (baixo) e HIGH (alto) das entradas INPUT 1 e INPUT 2 para mudança do sentido de rotação. Os pinos 10 e 15 (INPUT 3 e INPUT 4) são análogos aos pinos 2 e 7, sendo as entradas para controle de sentido de rotação do motor 2, conectado aos pinos 11 e 14 (OUTPUT 3 e OUTPUT 4).

Figura 23: Pinagem do driver L293D



Fonte: Arduino e cia (2014).

Nessa etapa além de controlarmos o sentido de rotação do motor DC e a sua velocidade, através da modulação PWM, será feito o uso do aplicativo *Android Bluetooth Electronics*. Este aplicativo pode ser encontrado para download gratuito no *Play Store* e possui uma biblioteca que contém 10 exemplos prontos que podem ser utilizados sem prévia modificação. O aplicativo *Bluetooth Electronics* se comunica com o microcontrolador através de comunicação via *Bluetooth*. Para processar os dados oriundos do aplicativo pode-se fazer uso do Arduino, como no caso desse projeto, do *Raspberry Pi*, ou de qualquer outro sistema de prototipagem rápida.

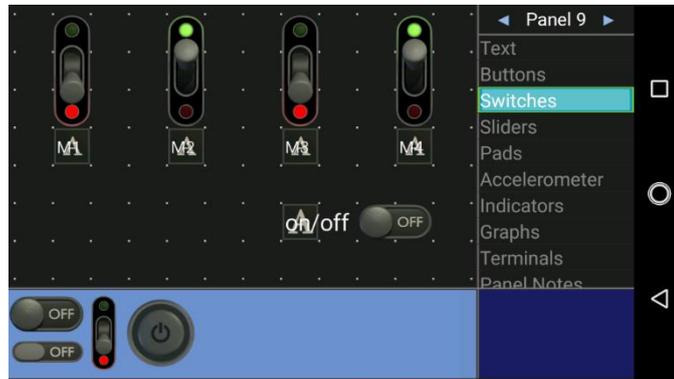
O Aplicativo conta com uma grande seleção de controles disponíveis incluindo botões, interruptores, controles on/off, leds, medidores, acelerômetros e gráficos. Para adicionar os funcionalidades desejadas basta apenas selecioná-las na biblioteca de ferramentas, arrastar e soltar na grade de uma das 20 tela editáveis e em seguida, editar suas propriedades.

O aplicativo *Bluetooth Electronics* será utilizado para elaborar uma versão digital do controle mecânico original do braço robótico *OWI. Inc. Robotic Arm Edge*, pelo qual o usuário poderá fazer o acionamento de quatro graus de liberdade dos cinco do braço robótico através de um aparelho celular, utilizando comunicação via *Bluetooth*.

O aplicativo *Android Bluetooth Electronics* possui telas de edição, nas quais o usuário pode editar livremente quais funcionalidades ele deseja realizar, de forma a possibilitar a interface do aparelho celular com o ambiente externo, seja para operação de acionamento/controle, tais como acender LED, acionar motores ou para aquisição de dados. A

Figura 24 apresenta uma tela de edição do aplicativo *Bluetooth Electronics*, desenvolvida para a aplicação, com o campo de *switches* habilitado.

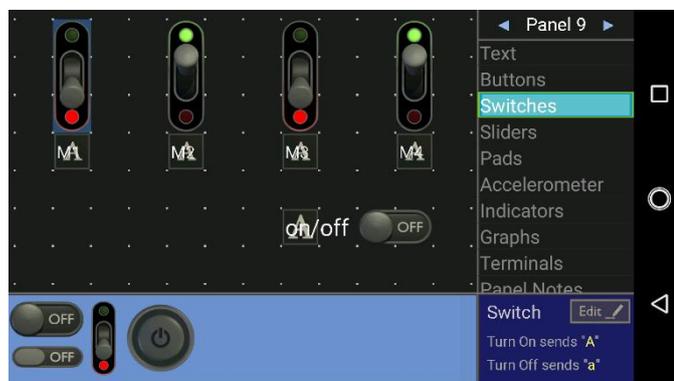
Figura 24: Tela de edição do aplicativo *Bluetooth Electronics*.



Fonte: Produção do próprio autor

Na tela de edição do aplicativo *Bluetooth Electronics* apresentada na Figura 24, observamos que há quatro possibilidades de switches, ou chaves do tipo on/off, que podem ser utilizadas para o acionamento do braço robótico. Para fazer uso dessa chave, basta somente clicar na escolhida e arrastá-la até o lugar desejado dentro da área editável da tela. Após montar o layout desejado, deve-se editar cada componente, no caso as chaves, com os caracteres que deseja-se que o aplicativo envie ao microcontrolador representando a ação designada para cada chave. Para isso, deve-se clicar no componente e em seguida em *edit*, no canto inferior direito, conforme indicado na Figura 25.

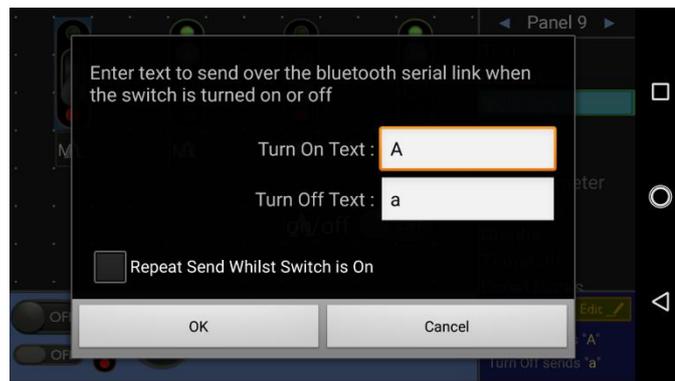
Figura 25: Configuração de componente no aplicativo *Bluetooth Electronics*.



Fonte: Produção do próprio autor

Ao clicar nesse campo, será aberta uma nova janela, Figura 26, na qual poderemos editar o caractere a ser enviado quando for acionada a chave.

Figura 26: Configuração de componente no aplicativo *Bluetooth Electronics*.



Fonte: Produção do próprio autor

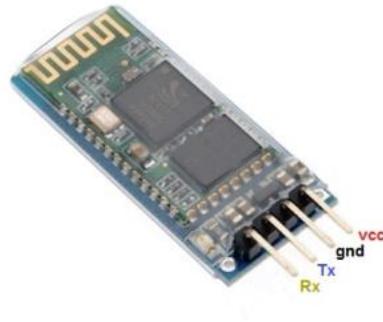
No campo de edição rotulado “*Turn On Text:*” deverá ser colocado o caractere desejado para representar o estado *on* (ligado), que no caso foi escolhido o caractere “A”. O mesmo ocorre com o campo a frente de texto “*Turn Off Text:*”, indicando o caractere que representa o estado *off* (desligado). Após essa edição deve-se clicar em “*Ok*”, e voltar à tela de edição da Figura 25.

Esse procedimento deve ser feito com todas as chaves dispostas na tela de edição. Para adicionar textos, de forma a identificar as chaves, deve-se clicar na aba *text* no canto superior direito da Figura 25 e fazer a edição de forma análoga à das chaves.

Concluído o layout do controle e a edição dos caracteres das chaves devemos conectá-lo ao Arduino, para que o mesmo possa processar os dados que serão enviados do aplicativo e fazer o acionamento do braço robótico. A interface do aplicativo *Bluetooth Electronics* com o mundo externo é feito através de comunicação *Bluetooth*. Para que o Arduino possa se comunicar com o aplicativo, faz-se necessário o uso de um módulo de comunicação Bluetooth, de forma a perceber os sinais oriundos do aplicativo e enviar para o microcontrolador e vice-versa. O módulo Bluetooth que será utilizado é o HC-06. Esse módulo opera como *slave* (escravo), ou seja, permite que outros dispositivos se comuniquem com ele, mas não permite que ele próprio se comunique com outros dispositivos *Bluetooth*. O alcance do módulo é de aproximadamente 10 metros e possui 4 pinos, conforme ilustra a Figura 27.

Esse módulo foi adotado em virtude do fato de que não haverá necessidade de leitura de dados da parte do aplicativo *Bluetooth Electronics* para o acionamento do braço robótico.

Figura 27: Módulo Bluetooth HC-06



Fonte: Engesier (2015)

Onde:

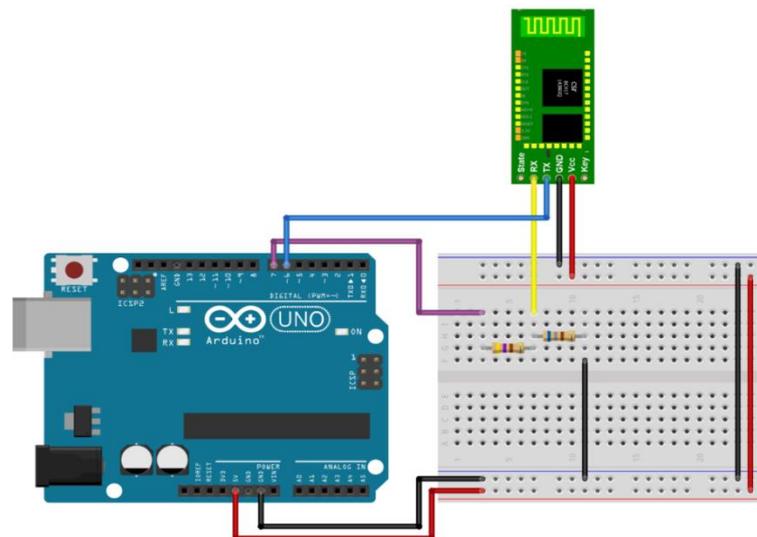
Vcc: pino de alimentação, que pode ser de 3,6 à 6V;

GND: pino que deverá ser conectado ao terra;

RX e TX: os dois últimos, utilizados para comunicação com o Arduino via serial, sendo TX o de transmissão e o RX de recepção.

O nível lógico alto dos pinos RX e TX é de 3,3V, porém o nível lógico alto de trabalho do Arduino Uno é de 5V, portanto há a necessidade do uso de um divisor de tensão no pino RX para evitar que o módulo seja danificado. A Figura 28 apresenta o divisor de tensão utilizado para adequar o uso do módulo HC-06.

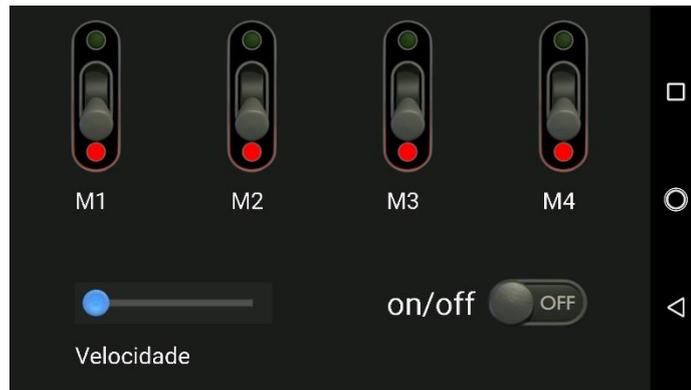
Figura 28: Divisor de tensão para adaptação do nível lógico do módulo *Bluetooth* HC-06.



Fonte: Thomsen (2015).

Para simular digitalmente o acionamento mecânico original do braço robótico *OWI, Inc. Robotic Arm Edge*, abordado no tópico 3.1. fez-se a disposição das chaves no aplicativo Android *Bluetooth Electronics* de forma a assemelhar-se ao original, conforme Figura 29.

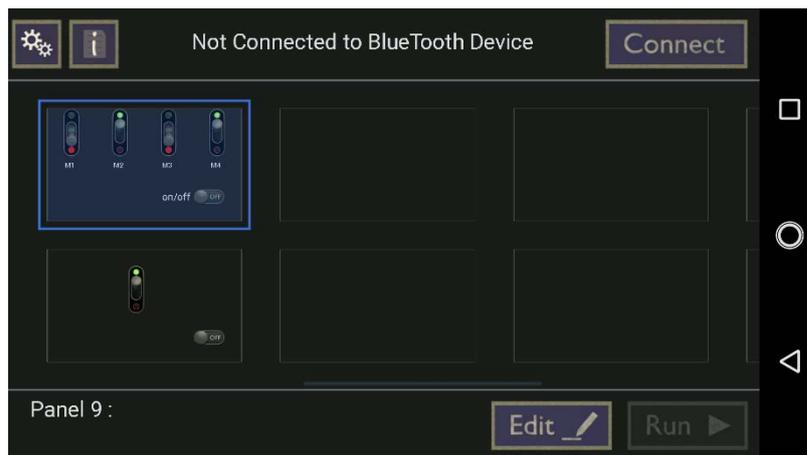
Figura 29: Modelo digital do controle original do braço robótico *OWI, Inc. Robotic Arm Edge* através do aplicativo Android *Bluetooth Electronics*.



Fonte: Produção do próprio autor

Como o controle digital do braço robótico pronto, basta apenas conectá-lo ao módulo *Bluetooth HC-06*. Para isso deve-se voltar a tela inicial do aplicativo, Figura 30, selecionar a tela desejada, no caso a tela de edição do controle digital do braço robótico e após em *connect*, no canto superior direito da Figura 30.

Figura 30: Tela inicial do aplicativo *Bluetooth Electronics*.



Fonte: Produção do próprio autor

Ao clicar em *Connect*, será aberta um nova tela, Figura 31. Nessa tela o primeiro passo é clicar em *Discover*, que mostrará dentro do bloco *Discovered Divices* os dispositivos ativos que possuem comunicação *Bluetooth* que estão próximos ao celular que contém o aplicativo.

Como no caso estamos utilizando o módulo Bluetooth HC-06, deve-se clicar no nome HC-06 dentro o bloco *Discovered Devices* e posteriormente em *Pair*, para fazer o pareamento dos dispositivos. Ao terminar o pareamento, o dispositivo aparecerá no bloco *Paired Devices*, assim, clica-se no dispositivo já pareado (com conexão estabelecida) e após em *Connect* e em seguida em *Done*, ambos no canto inferior direito da Figura 31.

Figura 31: Tela de pareamento do aplicativo *Bluetooth Electronics*.

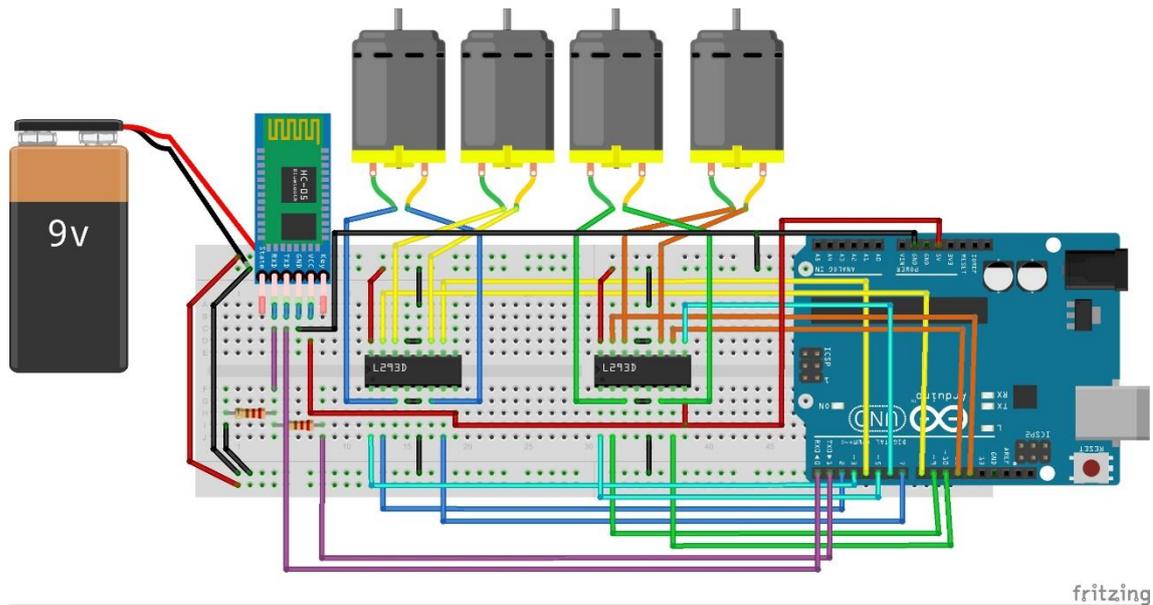


Fonte: Produção do próprio autor

Após o pareamento e conexão do módulo *Bluetooth* HC-06, basta retornar na tela inicial do aplicativo, Figura 30, onde o campo Run estará agora habilitado, e clicar nesse campo, levando-nos à tela apresentada na Figura 29.

Temos agora o controle pronto, basta apenas a montagem do driver para os motores. A Figura 32 mostra o esquemático de montagem para o acionamento de 4 motores do braço robótico *OWI. Inc. Robotic Arm Edge* sendo controlado remotamente através do aplicativo Android *Bluetooth Electronics*, através de comunicação *Bluetooth*.

Figura 32: Esquemático de montagem para o controle de quatro motores do braço robótico de forma remota.



Fonte: Produção do próprio autor

Após a montagem do circuito da Figura 32 fez-se o upload do programa a seguir no Arduino Uno.

// Programa : Controle de motor CC com o L293D com Bluetooth

char BluetoothData; // dado recebido do bluetooth via comunicação serial

// Pinos do Motor 1

int PWM1 = 3; //Ligado ao pino 1 do L293D Pwm do M1 e M2

int Entrada1M1 = 2; //Ligado ao pino 2 do L293D

int Entrada2M1 = 7; //Ligado ao pino 7 do L293D

// Pinos do Motor 2

int Entrada1M2 = 4; //Ligado ao pino 10 do L293D

int Entrada2M2 = 8; //Ligado ao pino 15 do L293D

// Pinos do Motor 3

int PWM2 = 5; //Ligado ao pino 1 do segundo L293D

int Entrada1M3 = 9; //Ligado ao pino 2 do segundo L293D

int Entrada2M3 = 10; //Ligado ao pino 7 do segundo L293D

// Pinos do Motor 4

```
int PWM3 = 6; //Ligado ao pino 9 do segundo L293D
int Entrada1M4 = 11; //Ligado ao pino 10 do segundo L293D
int Entrada2M4 = 12; //Ligado ao pino 15 do segundo L293D

int PwmValue = 0; // valor de PWM que será enviado do Aplicativo

void setup()
{
  //Define os pinos como saída
  pinMode(PWM1, OUTPUT);
  pinMode(PWM2, OUTPUT);
  pinMode(PWM3, OUTPUT);

  pinMode(Entrada1M1, OUTPUT);
  pinMode(Entrada2M1, OUTPUT);
  pinMode(Entrada1M2, OUTPUT);
  pinMode(Entrada2M2, OUTPUT);
  pinMode(Entrada1M3, OUTPUT);
  pinMode(Entrada2M3, OUTPUT);
  pinMode(Entrada1M4, OUTPUT);
  pinMode(Entrada2M4, OUTPUT);
  Serial.begin(9600); // velocidade da comunicação serial

  // Para que os motores iniciem sem movimento
  digitalWrite(Entrada1M1, LOW);
  digitalWrite(Entrada2M1, LOW);
  digitalWrite(Entrada1M2, LOW);
  digitalWrite(Entrada2M2, LOW);
  digitalWrite(Entrada1M3, LOW);
  digitalWrite(Entrada2M3, LOW);
  digitalWrite(Entrada1M4, LOW);
  digitalWrite(Entrada2M4, LOW);
}
```

```

void loop()
{
  if (Serial.available())
  {
    BluetoothData=Serial.read(); //lê o próximo character via bluetooth

    if(BluetoothData=='O') // quando a chave on/off está em modo on
    {
      while (BluetoothData!='o') // quando a chave on/off está em modo off
      {
        //Define a velocidade de rotação
        if(BluetoothData=='V')PwmValue = Serial.parseInt(); // lê valor da velocidade vinda do
aplicativo

        int velocidade1 = PwmValue; // dutycycle do PWM
        int velocidade2 = PwmValue; // dutycycle do PWM
        int velocidade3 = PwmValue; // dutycycle do PWM da garra

        analogWrite(PWM1, velocidade1);
        analogWrite(PWM2, velocidade2);
        analogWrite(PWM3, velocidade3);

        BluetoothData=Serial.read();

        if(BluetoothData=='A')
        {
          digitalWrite(Entrada1M1, LOW);
          digitalWrite(Entrada2M1, HIGH);

        }

        if(BluetoothData=='a')
        {
          //Aciona o motor no sentido inverso
          digitalWrite(Entrada1M1, HIGH);

```

```
    digitalWrite(Entrada2M1, LOW);

}
if(BluetoothData=='B')
{
    digitalWrite(Entrada1M2, LOW);
    digitalWrite(Entrada2M2, HIGH);

}

if(BluetoothData=='b')
{
    //Acciona o motor no sentido inverso
    digitalWrite(Entrada1M2, HIGH);
    digitalWrite(Entrada2M2, LOW);

}

if(BluetoothData=='C')
{
    digitalWrite(Entrada1M3, LOW);
    digitalWrite(Entrada2M3, HIGH);

}

if(BluetoothData=='c')
{
    //Acciona o motor no sentido inverso
    digitalWrite(Entrada1M3, HIGH);
    digitalWrite(Entrada2M3, LOW);

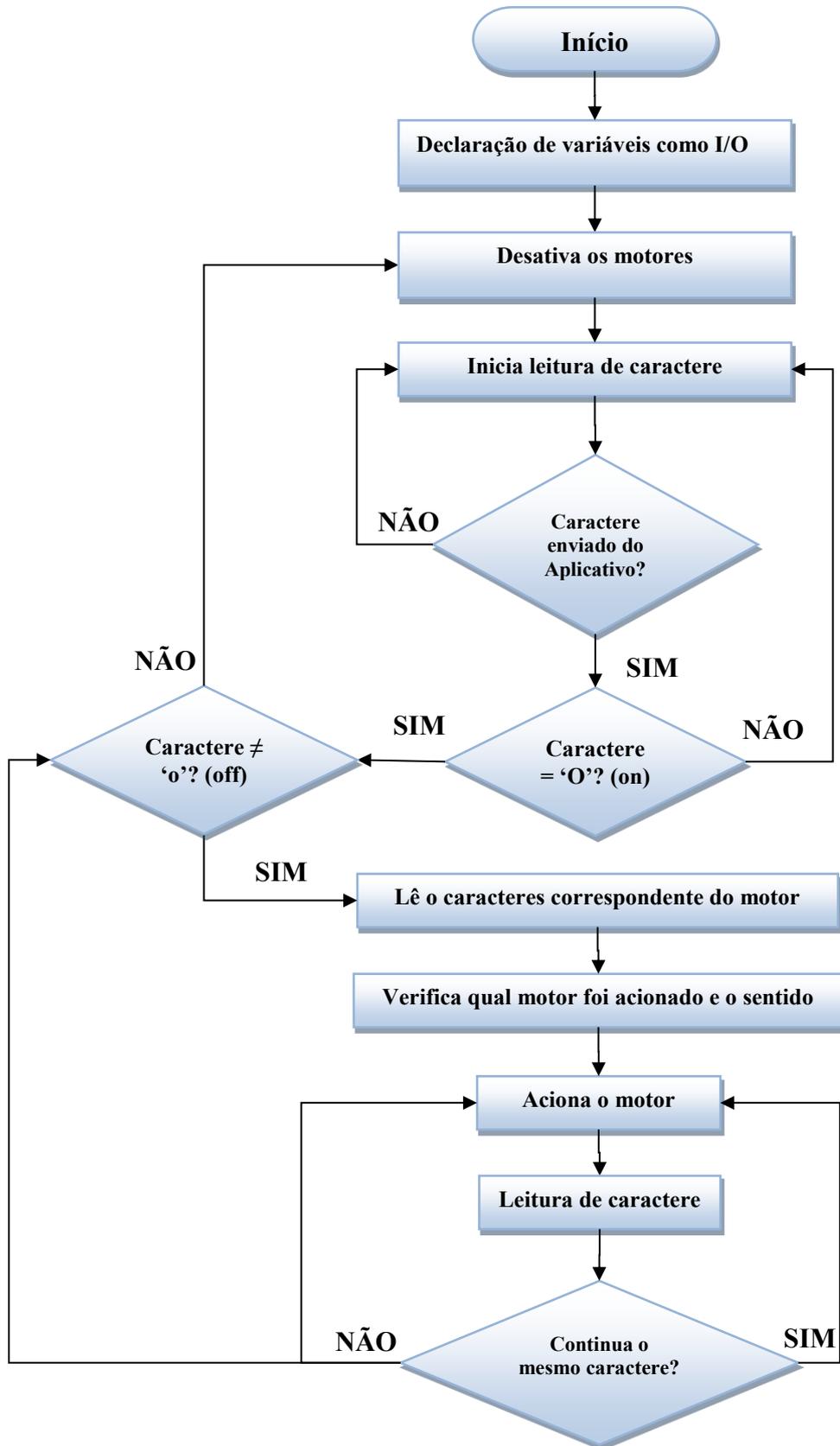
}

if(BluetoothData=='D')
{
    digitalWrite(Entrada1M4, LOW);
    digitalWrite(Entrada2M4, HIGH);
```

```
    }  
  
    if(BluetoothData=='d')  
        {  
            //Aciona o motor no sentido inverso  
            digitalWrite(Entrada1M4, HIGH);  
            digitalWrite(Entrada2M4, LOW);  
  
        }  
    }  
}  
  
// desativa os motores  
digitalWrite(Entrada1M1, LOW);  
digitalWrite(Entrada2M1, LOW);  
digitalWrite(Entrada1M2, LOW);  
digitalWrite(Entrada2M2, LOW);  
digitalWrite(Entrada1M3, LOW);  
digitalWrite(Entrada2M3, LOW);  
digitalWrite(Entrada1M4, LOW);  
digitalWrite(Entrada2M4, LOW);  
}  
}
```

A Figura 33 apresenta o fluxograma do programa apresentado.

Figura 33: Fluxograma do acionamento do braço robótico utilizando o *driver* L293D



Fonte: Produção do próprio autor

Observando o programa em uma visão macro e conforme o fluxograma da Figura 33, o programa inicia declarando as variáveis de entrada e saída (*input/output -I/O*) e a velocidade com que será feita a comunicação serial para o *Bluetooth*. Após essa declaração o programa desativa todos os motores do braço por segurança, evitando que os motores se movimentem sem que tenham sido solicitados.

Após esse etapa de inicialização o programa entra em uma rotina que será executada indefinidamente, a qual inicia com a leitura de caracteres oriundos do aplicativo *Bluetooth Electronics*. Enquanto não houver caracteres enviados, o programa fará a leitura de caracteres até que o usuário acione o controle digital no aplicativo. Ao receber um caractere, o programa verifica se esse caractere é a letra 'O', que indica que o controle digital foi acionado, em caso negativo o programa volta à rotina de leitura de caracteres e em caso positivo o programa verifica que se o caractere é diferente de 'o', que indica que o controle foi desativado. Essa verificação se faz necessária, uma vez que ao receber o caractere 'o', o programa deve desativar os motores. Caso o modo *on* continue ativado, a condição caractere \neq 'o' será verdadeira, assim, o programa faz uma nova leitura de caracteres, verificando agora para qual motor foi solicitado o acionamento e qual o sentido de rotação. Feito essa verificação, o programa fará a ativação do respectivo motor e fará mais uma leitura de caracteres, se o caractere lido continuar o mesmo, mantém-se o acionamento desse motor, caso contrário, além de manter o acionamento do motor verifica-se se a condição caractere \neq 'o' continua verdadeira e se sim, faz uma nova leitura de caracteres, de forma a verificar qual motor o usuário solicitou acionamento e qual o sentido de rotação, ou se o usuário apenas solicitou a inversão de rotação do motor que anteriormente estava em movimento. Caso a condição caractere \neq 'o' seja falsa, os motores serão desligados.

Analisando agora o código do programa, no início, encontramos a variável do tipo char chamada *BluetoothData*, essa variável receberá o código do tipo char, pelo qual o Arduino identificará a função a ser executada, tal como fazer determinado motor se movimentar ou parar.

A variável tipo int PWM1, pino 3 do Arduino, refere-se ao sinal PWM que fará o controle de velocidade dos motores M1 e M2. As variáveis tipo int Entrada1M1 e Entrada2M1, pinos 2 e 7 do Arduino respectivamente, representam os pinos aos quais o motor M1 será conectado. Cada motor possuirá atrelado a si duas variáveis do tipo int intituladas ENTRADAMx e ENTRADAMx, onde a letra x dessa variável representa o motor em questão. Como há quatro motores a serem controlados, teremos oito variáveis para o acionamento dos mesmos.

Assim como a variável PWM1, a variável PWM2 fará o controle de velocidade, porém do motor 3 e a variável PWM3 do motor 4.

No bloco `void setup()`, são definidas as variáveis descritas anteriormente como saída, utilizando a instrução `pinMode(variavel, OUTPUT)`, onde no lugar da palavra variável é colocado o nome da variável em questão.

O comando `Serial.begin(9600)` informa ao Arduino que será usada a comunicação serial para estipular a comunicação via Bluetooth e a velocidade com que será feita a troca de dados, que no caso é de 9600bps. Como o módulo Bluetooth HC-06 é conectado nos pinos TX e RX do Arduino, que é o mesmo pino que o Arduino Uno utiliza para realizar o upload do programa em seu microcontrolador, há a necessidade de se desconectar o módulo HC-06 do Arduino, no momento do upload do programa para não causar interferência, impossibilitando o upload.

Após ajustar a comunicação serial que será usada, é aplicado aos terminais dos motores o mesmo potencial, que no caso é o de 0V, para garantir que ao iniciar o ciclo de funcionamento do braço robótico os motores não iniciem em movimento. Utiliza-se as instruções `digitalWrite(ENTRADA1M1, LOW)` e `digitalWrite(ENTRADA2M1, LOW)` para parar o motor M1 e instruções análogas são utilizadas para os demais motores.

Dentro do bloco `void loop()`, que será executado em um loop infinito, encontramos a instrução condicional `if (Serial.available())`, que possui a finalidade de verificar se o aplicativo enviou alguma informação para o Arduino. Em caso afirmativo são executadas as instruções dentro desse bloco, que são inerentes ao acionamento dos cinco motores do braço robótico; caso contrário, não é executada nenhuma ação e o programa fica verificando se houve algum envio de dado por parte do aplicativo *Bluetooth Electronics*.

Ao receber algum dado do aplicativo, que no caso são caracteres, a instrução `BluetoothData=Serial.read()` fará a leitura do próximo caractere enviado via *Bluetooth*. De posse desse caractere a instrução condicional `if(BluetoothData=='O')` verificará se a chave virtual *on/off* do controle digital do aplicativo *Bluetooth Electronics* está em modo *on*, em caso afirmativo será executado a instrução `while (BluetoothData!='o')`, que fará o programa ficar em loop na rotina de controle dos motores até que a chave *on/off* do aplicativo for comutada para estado *off*, caso contrário os motores serão desativados, conforme no bloco `void setup()`, de forma a garantir que não haja movimento indesejado dos motores entre as mudanças de rotinas.

Dentro do bloco da instrução `while (BluetoothData!='o')`, é definido o valor de *duty cycle* dos sinais PWM que farão o controle de velocidade dos motores. Para isso declarou-se

as variáveis do tipo int velocidade1, velocidade2 e velocidade3 com o valor que receberão o valor de *duty cycle* do aplicativo. A instrução *if(BluetoothData=='V')PwmValue = Serial.parseInt();* lê valor da velocidade vinda do aplicativo, esse valor é repassado para as variáveis velocidade1 a 3.

A instrução *analogWrite(PWM1, velocidade1)* habilita o sinal PWM no pino PWM1 com o *duty cycle* escolhido. O mesmo ocorre com os pinos PWM2 e PWM3.

Novamente a instrução *BluetoothData=Serial.read()* é executada para fazer a leitura do caractere que selecionará o motor a ser ativado e o sentido de rotação. Caso alguma chave virtual referente aos motores do braço robótico seja acionada, a instrução condicional *if(BluetoothData=='caractere')* selecionará qual motor deverá ser acionado, sendo que a palavra caractere da instrução faz referência à letra correspondente de cada motor, como sintetizado na Tabela 2.

Tabela 2: Relação de caracteres equivalentes aos motores do braço robótico.

Caractere	Função
A	Aciona o motor M1 em determinado sentido de rotação
a	Aciona o motor M1 em sentido contrário de rotação
B	Aciona o motor M2 em determinado sentido de rotação
b	Aciona o motor M2 em sentido contrário de rotação
C	Aciona o motor M3 em determinado sentido de rotação
c	Aciona o motor M3 em sentido contrário de rotação
D	Aciona o motor M4 em determinado sentido de rotação
d	Aciona o motor M4 em sentido contrário de rotação

Fonte: Produção do próprio autor

Para descrevermos melhor a etapa de acionamento dos motores, vamos tomar como exemplo o motor M1. Suponhamos que a chave virtual referente ao motor M1 do controle virtual do aplicativo, apresentado na Figura 25, seja comutada para cima. Nesse momento, o aplicativo enviará o caractere 'A', conforme programado no aplicativo, para o Arduino, que fará a leitura desse caractere. Ao receber o caractere 'A' serão executadas as instruções *digitalWrite(ENTRADA1M1, LOW)* e *digitalWrite(ENTRADA2M1, HIGH)*, fazendo com que o motor gire em determinado sentido de rotação. Para fazer o motor girar em sentido contrário, basta comutar a chave virtual para baixo. O motor continuará nesse sentido até que

seja comutada novamente a chave virtual, ou até que a chave virtual on/off seja comutada para off.

Para o acionamento dos demais motores segue-se o mesmo raciocínio, mudando apenas o caractere responsável pelo acionamento. Como o Arduino compila suas instruções de forma serial, utilizando esse programa não será possível acionar mais de um motor simultaneamente.

3 CONCLUSÃO

Este trabalho mostrou de forma explicativa e didática como operar o controle de uma braço robótico com motores DC, utilizando a metodologia de apresentar o conteúdo em etapas, por meio das qual foi-se incrementado gradualmente o nível de conhecimento necessário e a aplicação envolvidos, gerando de forma gradual a base necessária para se controlar o braço robótico *OWI, Inc. Robotic Arm Edge* de forma remota e digital através de comunicação via *Bluetooth*.

Embora a lógica utilizada nesse trabalho tenha sido aplicada ao controle de um braço robótico didático, essa mesma lógica e conceitos aplicados podem ser utilizados em robôs em uma linha de produção industrial, possibilitando seu controle de forma remota, trazendo mais segurança e precisão ao processo.

REFERÊNCIAS

- ARDUINO E CIA (Org.). **Controle de motor CC com o L293D - Ponte H**. 2014. Disponível em: <<https://www.arduinoecia.com.br/2014/04/controle-de-motor-cc-com-o-l293d-ponte-h.html>>. Acesso em: 21 fev. 2017.
- ARDUINOLIVRE (Ed.). **Transistor + motor DC + arduino – parte 1**. 2013. Disponível em: <<https://arduinolivres.wordpress.com/author/sssillas/>>. Acesso em: 15 fev. 2018.
- BRAGA, Newton C.. **Eletrônica Básica**. 5. ed. São Paulo: Ncb, 2009. 216 p.
- ENGEASIER (Ed.). **Módulo bluetooth HC-06**. 2015. Disponível em: <<http://engeasier.com.br/modulo-bluetooth-hc-06+160569>>. Acesso em: 24 fev. 2018.
- ELECTRONICS HUB. **Arduino DC motor control using L298N**. 2018. Disponível em: <<https://www.electronicshub.org/arduino-dc-motor-control-using-l298n/>>. Acesso em: 06 fev. 2018.
- FRANCA., Morelato. **Motores elétricos de corrente contínua e universal**. 2001. Disponível em: <https://cdn.hackaday.io/files/9127390489568/motor_cc.pdf>. Acesso em: 15 fev. 2018.
- HIRSCHY, Gail R.. **The chester county career center's article about OWI's robotic arm edge**. 2013. Disponível em: <<http://owirobot.blogspot.com.br/2013/04/the-chester-county-career-centers.html>>. Acesso em: 06 jun. 2017.
- MAIMON, F. **Projeto de um sistema eletrônico para o controle de motores de alta potência por PWM**. 2004. 55 f. Trabalho de Conclusão de Curso (Engenharia de controle e Automação) - Pontifícia Universidade Católica, Rio de Janeiro, 2004.
- MCROBERTS, Michael. **Arduino básico**. 2. ed. São Paulo: Novatec, 2011. 456 p.
- MORAES, Marcelo. **Ponte H - controle de motores DC**. 2012. Disponível em: <<https://arduinobymyself.blogspot.com.br/2012/08/ponte-h-controle-de-motores-dc.html>>. Acesso em: 09 fev. 2017.
- PATSKO, Luís Fernando. **Tutorial montagem da ponte H**. 2006. Elaborada por Maxwell Bohr. Disponível em: <http://www.maxwellbohr.com.br/downloads/robotica/mec1000_kdr5000/tutorial_eletronica_-_montagem_de_uma_ponte_h.pdf>. Acesso em: 23 mai. 2017.
- ROBOOTT (Org.). **PONTE H COM TIP120**. 2015. Disponível em: <<https://roboott.wordpress.com/2015/10/25/ponte-h-com-tip120/>>. Acesso em: 08 fev. 2017.
- SERODJA, Ryan. **Transistor jenis NPN dan jenis PNP**. 2010. Disponível em: <<http://ryanserodja.blogspot.com.br/2010/06/transistor-jenis-npn-dan-jenis-pnp.html>>. Acesso em: 15 fev. 2018.
- SOUZA, Fábio. **Arduino - saídas PWM**. 2014. Disponível em: <<https://www.embarcados.com.br/arduino-saidas-pwm/>>. Acesso em: 11 out. 2017.

ST MICROELECTRONICS (Estados Unidos) (Ed.). **Dual full-bridge driver**. 2000. Disponível em: <https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf>. Acesso em: 15 fev. 2018.

ST MICROELECTRONICS (Estados Unidos) (Ed.). **Push-pull four channel driver with diodes**. 2003. Disponível em: <https://www.arduino.cc/documents/datasheets/H-bridge_motor_driver.PDF>. Acesso em: 15 fev. 2018.

THOMSEN, Adilson. **Configuração do módulo bluetooth HC-06 com arduino**. 2015. Disponível em: <<http://buildbot.com.br/blog/configuracao-do-modulo-bluetooth-hc-06-com-arduino/>>. Acesso em: 24 fev. 2018.