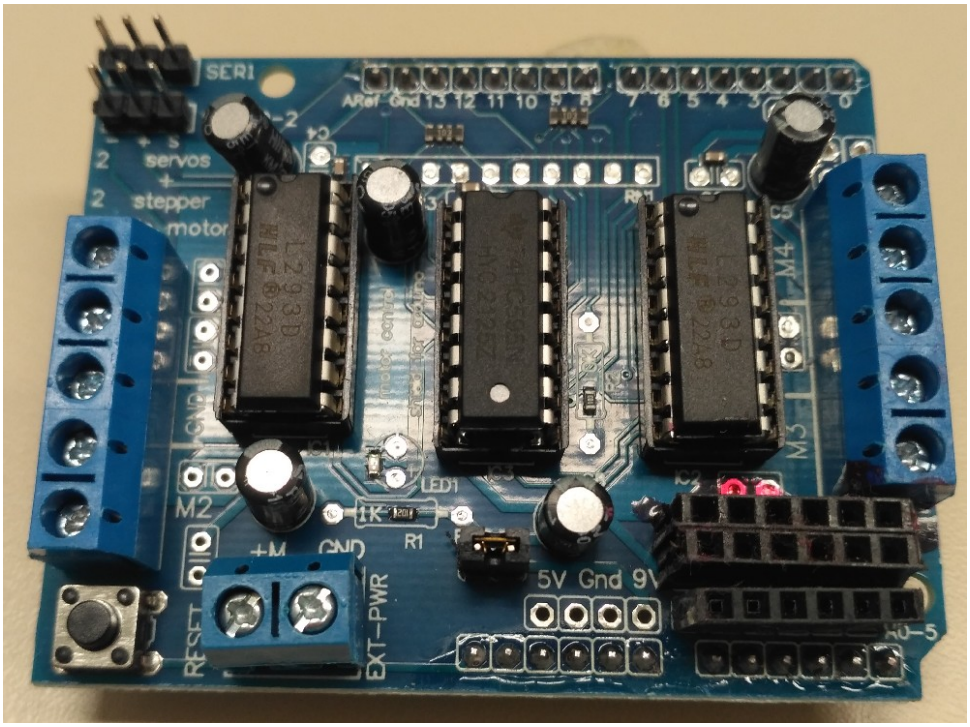
Controle de servo motores usando o *shield* LD293

Do Inglês, *shield* quer dizer escudo; no ambiente do Arduino chamamos de *shield* queles módulos que podem ser encaixados sobre a placa do Arduino. Os mais comuns são aqueles voltados ao Arduino UNO. O que utilizaremos é composto de dois controladores LD293 e um registrador de deslocamento 74HC795. Ele é capaz de controlar 4 motores DC ou 2 motores de passo. E também 2 servo motores. Os circuitos integrados são instalados em soquetes, de forma que podem ser retirados para experiências e circuitos individuais.



O *Motor Shield* utiliza o circuito integrado ponte H L293D e o circuito integrado 74HC595N para controle de comunicação. As 6 entradas analógicas (A0 a A5) estão disponíveis ara uso na placa (embora nem sempre com conectores – na foto do meu *shield* soldei conectores para facilitar, incluindo as ligações GND e +5V que são ao lado). Ele utiliza os seguintes pinos do Arduino:

- Motor 1 (ou motor de passo 1 conexão 1): pino digital 11 do Arduino
- Motor 2 (ou motor de passo 1 conexão 2): pino digital 3 do Arduino
- Motor 3 (ou motor de passo 2 conexão 1): pino digital 5 do Arduino
- Motor 4 (ou motor de passo 2 conexão 2): pino digital 6 do Arduino
- Servo Motor 1: pino digital 10 do Arduino
- Servo Motor 2: pino digital 9 do Arduino
- Os pinos digitais 2 e 13 não são utilizados pelo *shield*, então você pode usá-los em seus protótipos se precisar.
- Os pinos digitais 4, 7, 8 e 12 são utilizados para controlar os motores por meio do 74HC795.

Veja a referência do fabricante (fornecedores chineses podem variar a placa, mas seguem as mesmas ligações físicas): https://cdn-learn.adafruit.com/downloads/pdf/adafruit-motor-shield.pdf

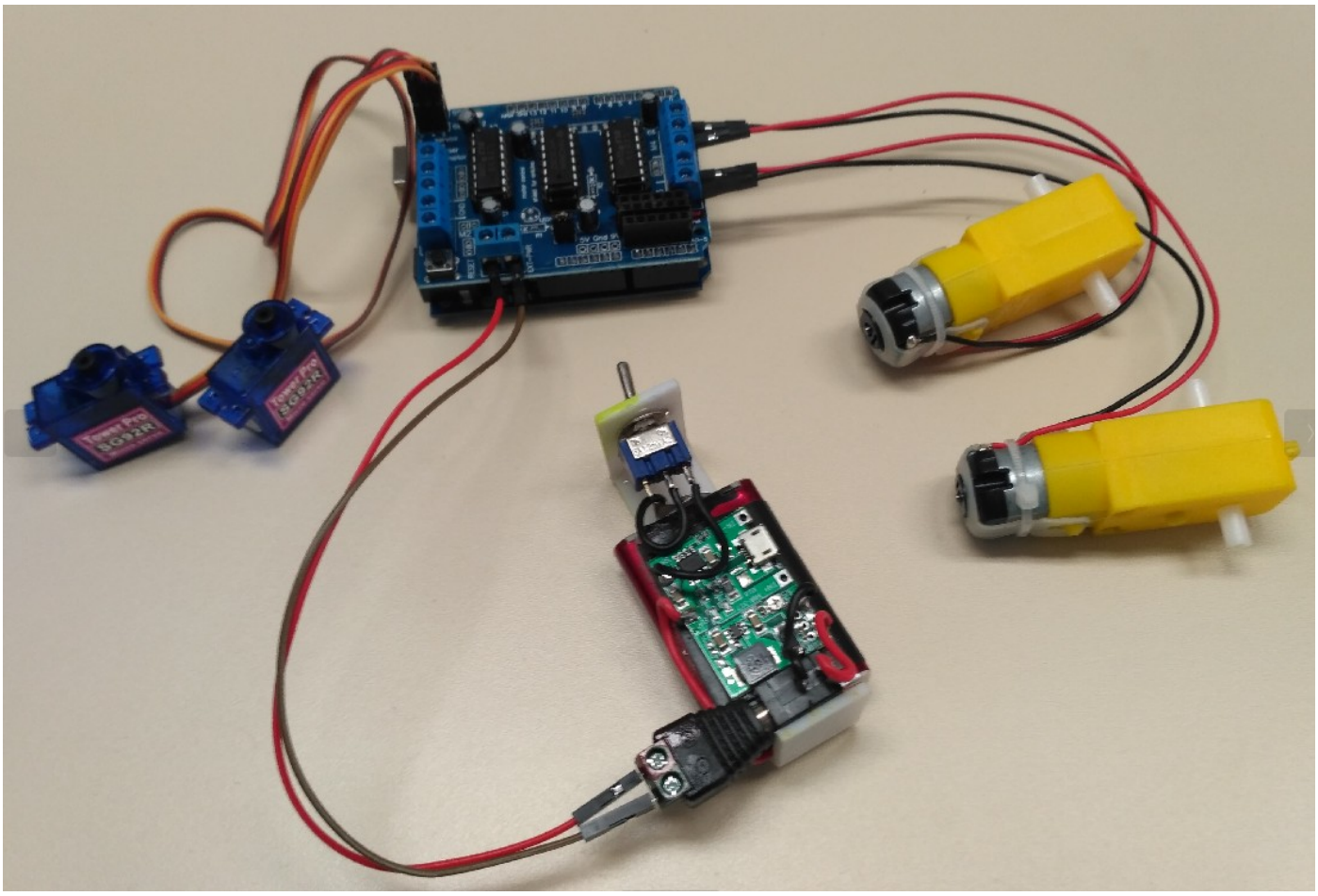
servosComShieldLD293 §

```
1 #include <Servo.h> // Biblioteca padrão para controlar servo motores
2 //note que para o controle dos servos nesta versão da placa não usamos a biblioteca Adafruit
3
4 Servo servo1; //O servo 1 é controlado pela porta digital 9 do Arduino
5 Servo servo2; //O servo 2 é controlado pela porta digital 10 do Arduino
6
7 void setup() {
8
9 }
10
11 void loop() {
12   servo1.attach(9); // Não mude, o shield vem ligado assim
13   servo1.write(0); //posição 0 graus
14   delay(1000);
15   servo1.write(90); //posição 90 graus
16   delay(1000);
17   servo1.write(180); //posicao 180 graus
18   delay(1000);
19   servo1.detach(); //desconecta o servo
20   servo2.attach(10); // Não mude, o shield vem ligado assim
21   servo2.write(0); //posição 0 graus
22   delay(1000);
23   servo2.write(90); //posição 90 graus
24   delay(1000);
25   servo2.write(180); //posicao 180 graus
26   delay(1000);
27   servo2.detach(); //desconecta o servo
28 }
```

Explicação:

- na linha 1 foi incluída a biblioteca de controle;
- nas linhas 3 e 4 foram declaradas variáveis (escolhi os nomes Mesq e Mdir para referência a um motor esquerdo e um direito em um carrinho), e foi informado em qual porta cada um foi conectado (usei a M3 e a M4, mas poderia ser M1 ou M2 ou qualquer ‘mistura’ de duas portas);
- entre as linhas 6 e 17 foi declarada a função **setup** e dentro dela há duas informações para cada um dos dois motores: a velocidade inicial (usei 150, por meio do método **setSpeed** - pode ser qualquer número entre 0 e 255); e, por meio do método ‘**run**’, foi informado o argumento ‘RELEASE’, a qual faz os motores desligarem;
- entre as linhas 19 e 41 temos a função **loop**; dentro dela foram utilizados os métodos **run** com os argumentos ‘FORWARD’ e ‘BACKWARD’, além do ‘RELEASE’, intercalados com comandos **delay**. Isto fará com que os motores girem em um sentido durante 2 segundos (2000 ms), desliguem e permaneçam assim por 0,1s (100ms) e depois liguem girando no sentido oposto por mais 2 segundos (se você testar e os motores não girarem ambos no mesmo sentido, inverta as ligações de um dos motores).

Ligação dos servos:



Preste atenção na polaridade do servo – o pino de controle (laranja) fica ‘para dentro’ da placa; o pino do GND (preto, ou, no meu exemplo, marrom), fica ‘para fora’ da placa. A alimentação foi fornecida por uma bateria de 3,7V ligada em um módulo carregador / *step up* integrado identificado pelo código J5019 - – veja como em https://tiaplicada.ufpr.br/wp-content/uploads/2024/08/usodomoduloj5019stepupcarregador.pdf.

ATENÇÃO - os servos consomem uma corrente elevada, se puder alimente-os com energia separa daquela da placa do Arduino (também é aconselhável colocar um capacitor eletrolítico de 100uF em paralelo com a alimentação do servo motor).

O código a seguir permite que você digite no monitor serial qual o ângulo desejado para o servo.

anguloFixoDigitado

```
1  #include <Servo.h>// biblioteca de controle do servo motor
2
3  Servo servo1;     //cria um 'objeto' servo motor
4  int angulo = 90;   //variável para ajuste do ângulo
5
6  void setup() {
7    servo1.attach(9); //servo ligado na porta digital 9, que é pwm - servo no módulo
8    Serial.begin(9600); //comunicação serial
9  }
10
11 void loop() {
12   if(Serial.available() > 0 ){ //se houver dados na Serial
13     angulo = Serial.parseInt(); //lê um número inteiro
14     servo1.write(angulo); //escreve no servo o ângulo lido
15     Serial.end(); //desliga a serial
16                   //comente a linha acima e veja o efeito
17                   //vai ler o ângulo e depois gravar um '0' na variável
18   }
19   delay(200);
20   Serial.begin(9600);
21 }
```

O servo começa com um ângulo de 90º. Ligue o monitor serial (Ferramentas/ monitor serial). Digite um valor para o ângulo. Na linha 12 o Arduino vai detectar que há algo na interface serial (por meio do método available do objeto **Serial**). Na linha seguinte (12) a variável inteira angulo receberá o número lido no monitor serial (parseInt() converte para inteiro o que você digitar – se for possível, se não for, despreza). Na sequência o ângulo escolhido é enviado ao servo (por meio do método write, que tem como parâmetro o valor do ângulo). A linha 15 está lá para resolver uma situação comum: quando você digita algo no monitor serial em geral será enviado também um comando de nova linha; este comando é interpretado como '0', e moverá o servo para o ângulo '0'. Para evitarmos isso foi acrescentada a linha 15, que termina ('end') a comunicação serial evitando o envio de um caractere extra (se você quiser testar – teste – remova a linha 15 ou comente-a e veja o resultado: o servo irá se movimentar para a posição escolhida e depois retornar).