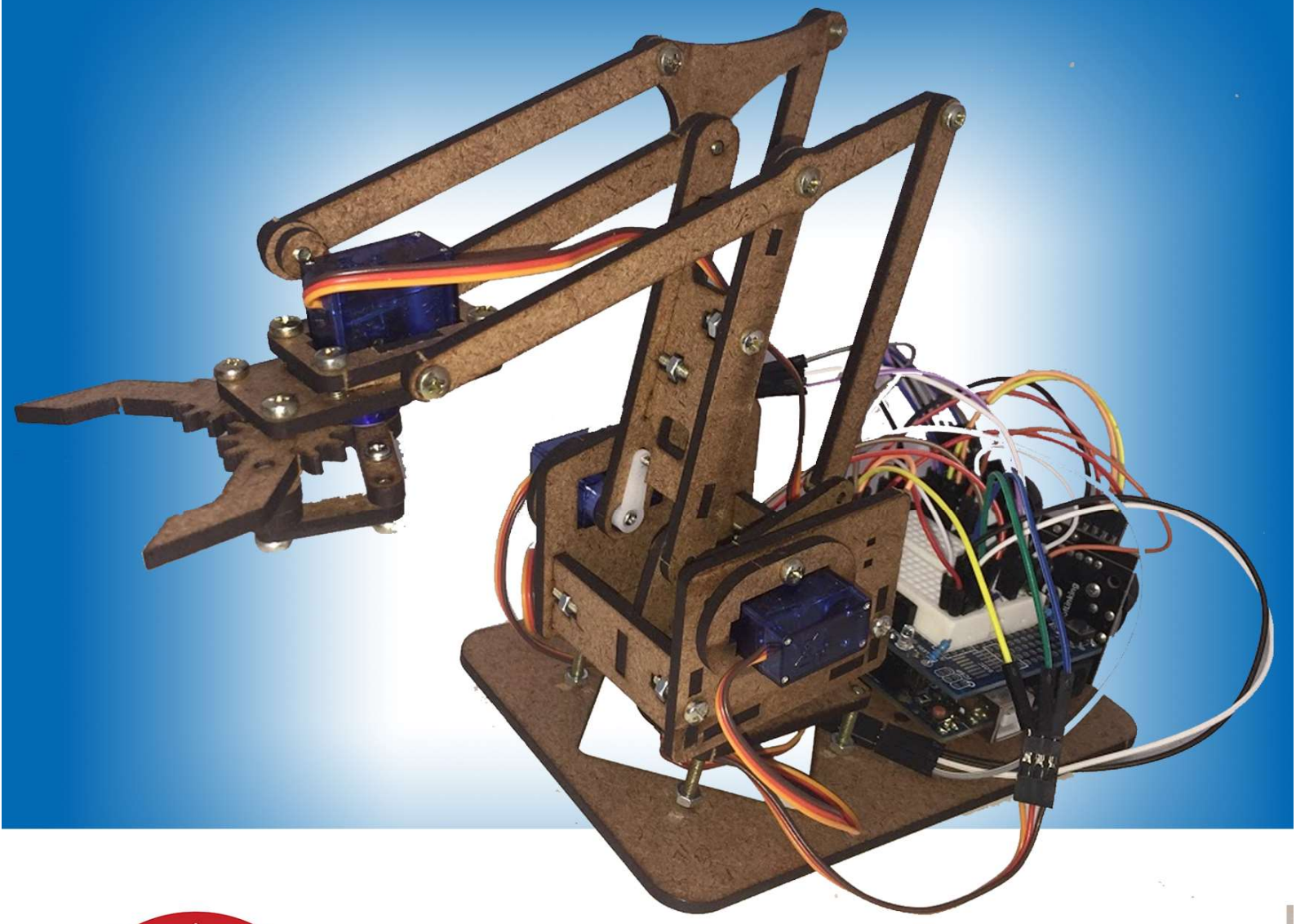
Oficina de Robótica com Arduino



FAB LAB
JOINVILLE

Sumário

Softwares Utilizados

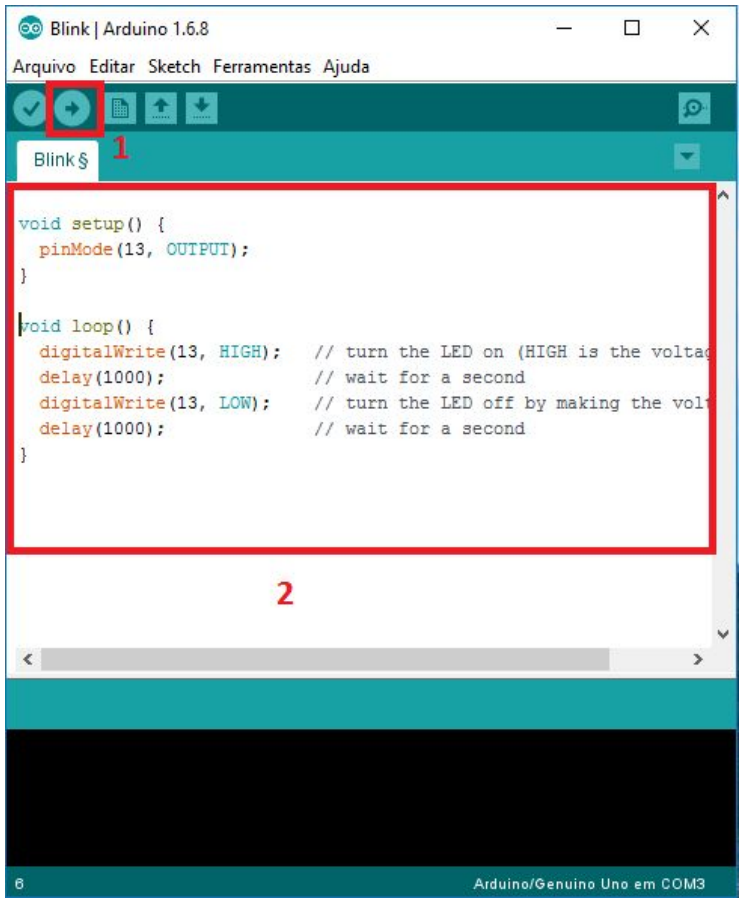
Para desenvolver os laboratórios deste guia utilizaremos dois sofwares, o Arduino IDE e o 123D Circuits.

Arduino IDE

O Arduino IDE permite que a placa Arduino seja programada de forma fácil, sem se preocupar com muitos detalhes que envolvem a compilação e a transferência do programa para a placa do Arduino. Para utilizar basta baixar o programa utilizando o link:

Download do Arduino IDE: https://www.arduino.cc/en/Main/Software

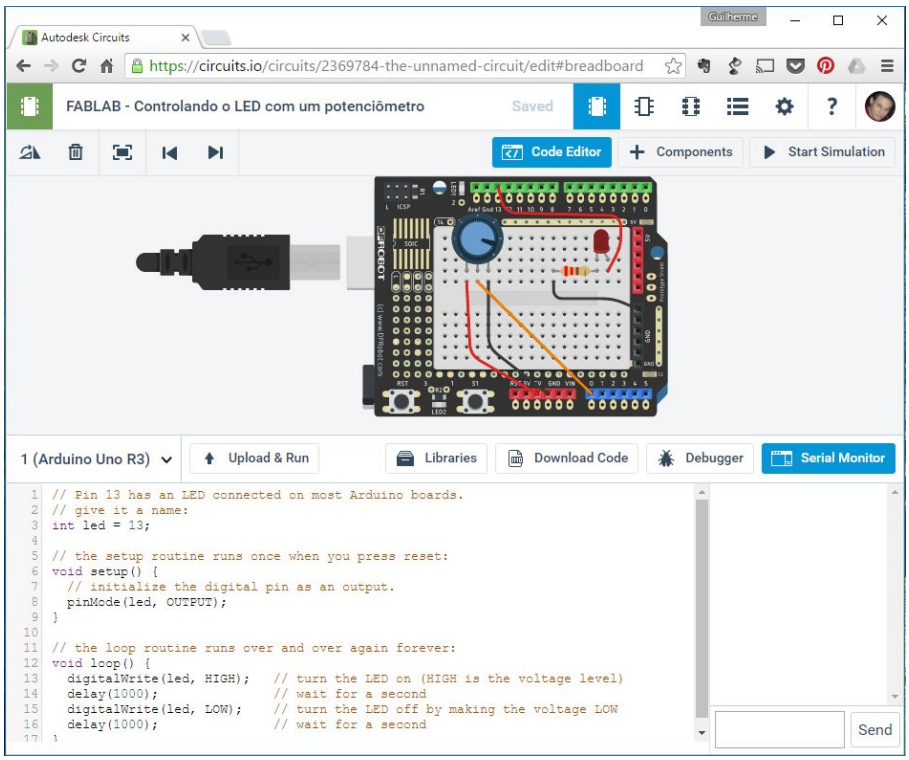
Ao abrir o programa pela primeira vez você verá uma tela como a imagem abaixo:



O botão 1 serve para compilarmos executarmos o código na placa Arduino. Na área 2 está o código que será executado.

Circuits.io

Outra ferramenta que utilizaremos é o site Circuits.io (http://circuits.io). É um site e você não precisa instalar nada no seu computador, basta entrar e se cadastrar. Neste site é possível montar os componentes eletrônicos, escrever o código e simular sua execução antes de montar todos esses passo na placa física. A imagem abaixo mostra um exemplo.



Todos os laboratórios serão disponibilizados no 123D Design para que você possa ver com calma as ligações, o código e possa executar a simulação para ver o seu funcionamento.

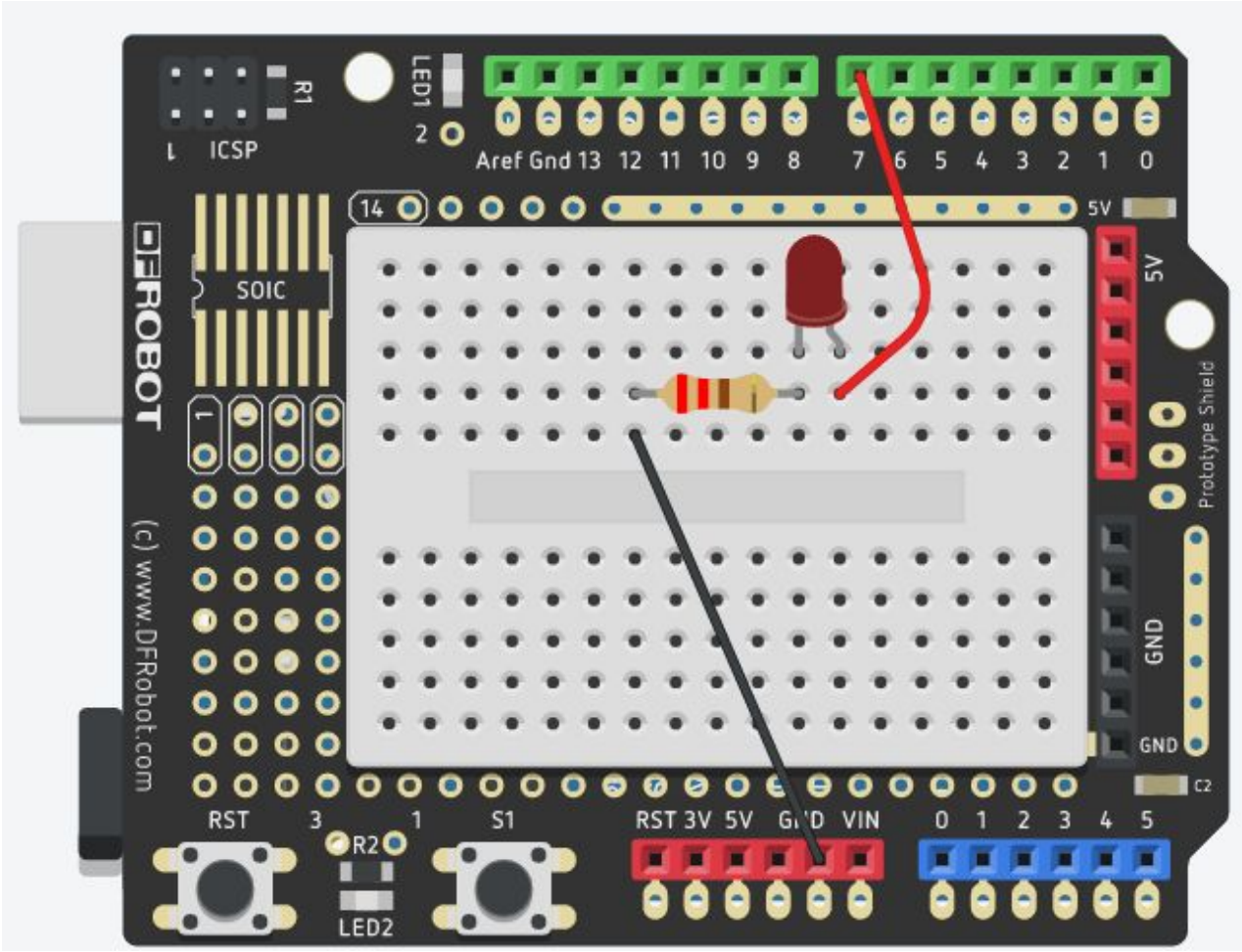
Laboratório 1 - Piscando um LED

Nesse laboratório controlaremos o acionamento de um LED utilizando o Arduino. Para ligar os componentes eletrônicos utilizaremos uma protoboard. A protoboard permite ligar os componentes sem a necessidade de soldá-los definitivamente.

Partes

1x LED (qualquer cor)
1x Resistor 220Ω
2x Jumpers (cabos)
1x Arduino
1x Protoboard

Eletrônica



Link para o labortatório:
http://bit.ly/fablab_lab1

Código

```
1. //Identifica em qual pino o LED foi
2. //ligado
3. int led = 7;
4.
5. void setup() {
6.  //Configura o pino 7 como saída
7.  pinMode(led, OUTPUT);
8. }
9.
10.
11. void loop() {
12.  //Liga o LED
13.  digitalWrite(led, HIGH);
14.
15.  //Espera 1 segundo
16.  delay(1000);
17.
18.  //Desliga o LED
19.  digitalWrite(led, LOW);
20.
21.  //Espera 1 segundo
22.  delay(1000);
23. }
```

O que aconteceu?

Para que o Arduino saiba o que fazer é necessário programá-lo. O programa escrito acima possui duas partes, a função **setup** e a **loop**. A função setup é executada uma única vez quando o Arduino é ligado. Já a função **loop** é executada repetidamente. O Arduino fica executando a função **loop** continuamente, quando ela termina, ela começa do início novamente.

Portas de saída
O Arduino possui diversas portas para se comunicar com o mundo externo, acender luzes, verificar o valor de sensores (ex: um sensor de temperatura). No programa anterior utilizamos a porta 7 para acionar um LED. No Arduino uma mesma porta pode funcionar como entrada ou saída, para isso precisamos informar o Arduino como a porta 7 deve ser tratada (entrada ou saída). Nesse caso a porta funciona como saída, então na função **setup** é executado o comando **pinMode(led, OUTPUT)**. Nesse comando foi informado o número da porta (led = 7) e o tipo (OUTPUT).

A função **loop** é o coração do programa, ela é executada repetidamente até que o Arduino seja desligado. No programa acima utilizamos o comando **digitalWrite(led, HIGH)** para ligar o LED e digitalWrite(led, LOW) para desligá-lo. Repare que a única diferença é o HIGH e LOW. O comando delay(2000) permite fazer o Arduino esperar um período de tempo. Esse período deve ser informado em milisegundos.

Comandos aprendidos

Comando	**Descrição**
pinMode(porta, OUTPUT);	Configura uma porta como saída.
digitalWrite(porta, HIGH);	Liga uma porta
digitalWrite(porta, LOW);	Desliga uma porta
delay(tempo);	Faz o Arduino esperar um período de tempo em milisegundos

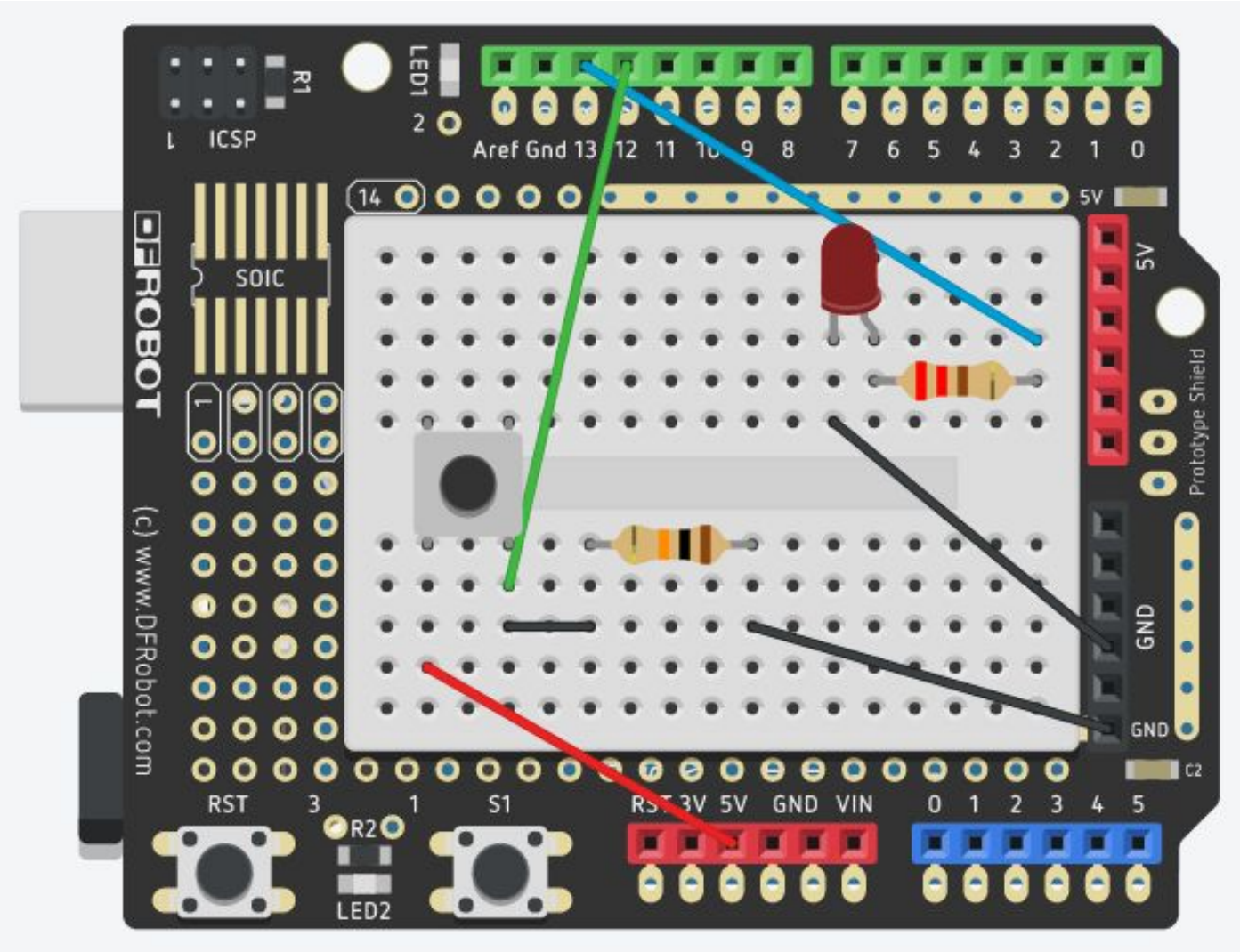
Laboratório 2 - Acionando o LED com um botão

Além das saídas digitais, o Arduíno também permite que sejam utilizados componentes de entrada, como sensores e botões, para que seja possível tomar decisões. Neste exercício vamos utilizar um botão para instruir o Arduino a acender um LED quando pressionado.

Partes

1x Arduino
1x Protoboard
1x Led
1x Resistor 220Ω
1x Botao
1x Resistor 10kΩ

Eletrônica



Link para o laboratório:
http://bit.ly/fablab_lab2

Código

```
1. //Cria uma variável led porta 13 para o LED
2. int led = 13;
3.
4. //Cria uma variável botao para a porta 12
5. int botao = 12;
6.
7. void setup() {
8.   //Configura a porta do LED como o
9.   //tipo de saída
10.  //(OUTPUT)
11.  pinMode(led, OUTPUT);
12.
13.  //Configura a porta do botao como o tipo de
14.  //entrada (INPUT)
15.  pinMode(botao, INPUT);
16. }
17.
18.
19. void loop() {
20.  //Captura se o botao está pressionado
21.  int acionado = digitalRead(botao);
22.
23.  //Verifica se o botão está pressionado (HIGH)
24.  if(acionado == HIGH){
25.    //Se estiver pressionado, liga o LED
26.    digitalWrite(led, HIGH);
27.  }else{
28.    //Se não estiver, desliga o LED
29.    digitalWrite(led, LOW);
30.  }
31. }
```

O que aconteceu?

Neste exercício, além, de configurarmos o LED como no exercício 1, utilizamos também um botão. Repare no comando **pinMode(botao, INPUT)** (linha 15), ele configura a porta **botao** (12) para que funcione como uma entrada digital.

Na função **loop** temos a lógica principal do nosso programa. Ele verifica se um botão está pressionado e, caso esteja, acende o LED. Para fazer essa verificação é utilizado o comando **digitalRead(botao)** (linha 21). Esse comando captura se o botão está pressionado (HIGH) ou não (LOW).

Para verificar se o botão está pressionado é necessário comparar o resultado obtido do comando digitalRead com HIGH ou LOW, essa verificação é feita utilizando o comando **if** (do português SE), na linha 24. Se a variável **acionado** for igual a HIGH (repare que usamos duas vezes o símbolo de igual), então ligamos o LED com o comando **digitalWrite(led, HIGH)** (linha 26). Se a variável **acionado** não for igual a HIGH o programa executará tudo que está dentro do **else**, linha 27, nesse caso, desligará o LED com o comando **digitalWrite(led, LOW)** (linha 29).

Comandos aprendidos

Comando	**Descrição**
pinMode(porta, INPUT);	Configura uma porta como entrada.
int resultado = digitalRead(porta);	Captura o valor de uma entrada digital
if(resultado == HIGH){ }	Verifica se o resultado da entrada digital é HIGH (Ex: botão pressionado)

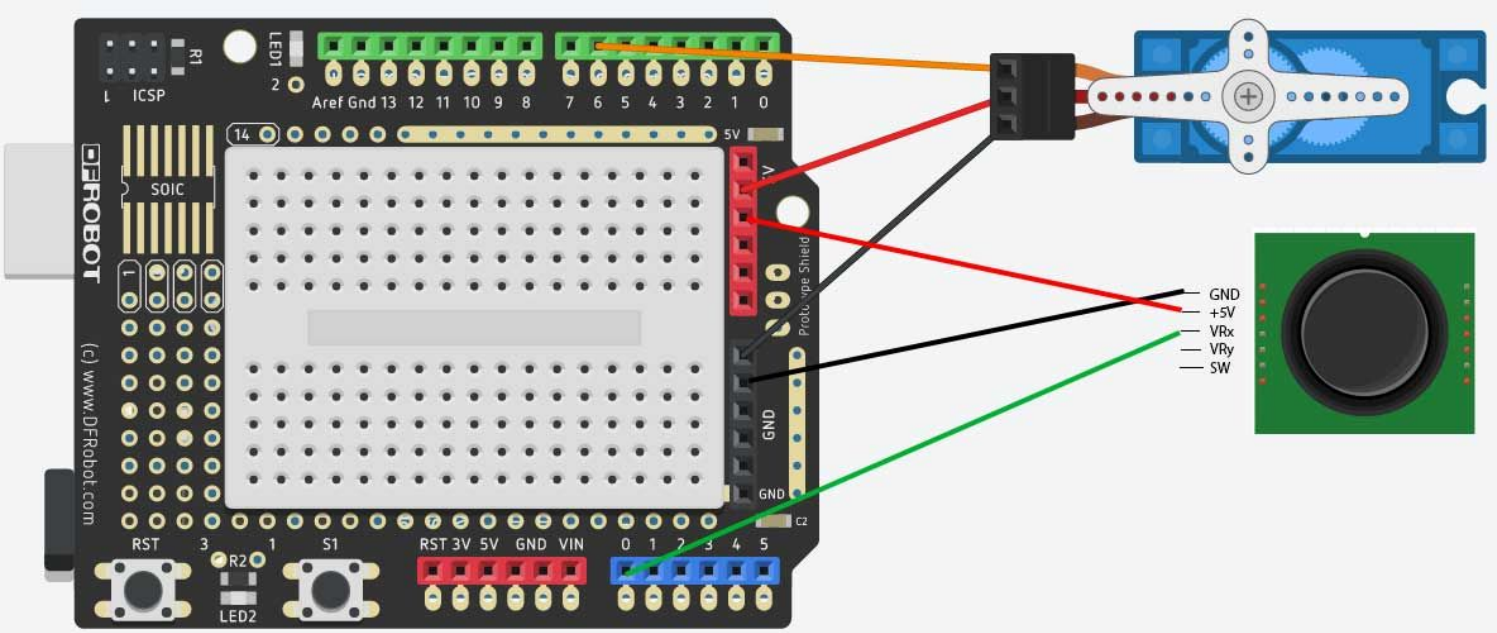
Laboratório 3 - Controlando um Servo motor com um potenciômetro

Neste exercício controlaremos a posição de um Servo motor utilizando um potenciômetro. O servo motor é um motor capaz de manter a sua posição em um determinado ângulo. Quando passamos para um servo motor o ângulo ele tentará manter-se nessa mesma posição mesmo que forças externas tentem movê-lo. O servo que utilizaremos neste projeto possui 180 graus (meia volta completa).

Partes

1x Arduino
1x Protoboard Shield
1x Joystick
1x Servo Motor

Eletrônica



Link para o laboratório:
http://bit.ly/fablab_garra_lab3

Código

```
#include "Servo.h"

int portaServo = 6;
int potenciometro = A0;
Servo servo;

void setup() {
  Serial.begin(9600);
  pinMode(potenciometro, INPUT);
  servo.attach(portaServo);
}

 void loop() {
   //Le a posicao do potenciômetro/joystick
   int valorPot = analogRead(potenciometro);

   //O valor que vem do potenciômetro está entre os valores 0
   //1023, porém, o servo necessita de um valor entre 0 e 180
   //O comando map faz a conversão entre esses dois
   //intervalos de valores
  int anguloServo = map(valorPot, 0, 1023, 0, 180);

  //Mostra no Monitor de serial o valor do servo
  Serial.print("Servo: ");
  Serial.println(anguloServo);

  //Manda o servo se posicionar no angulo convertido
  servo.write(anguloServo);
}
```

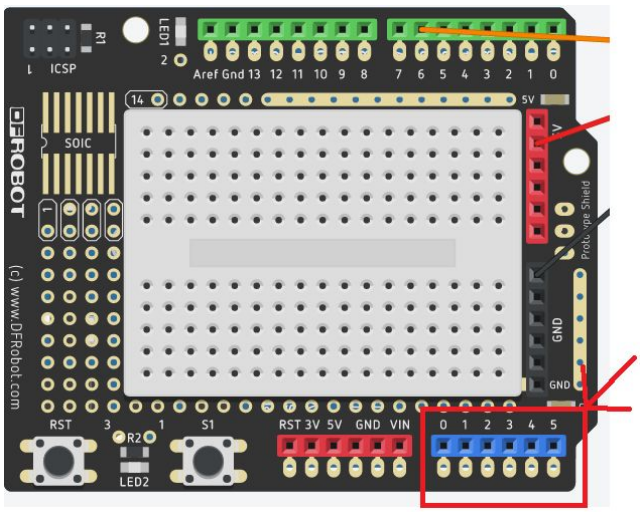
O que aconteceu?

Neste exercício temos alguns elementos novos, mas não se preocupe, vamos ver com calma cada um deles. O programa começa com a declaração da biblioteca que vamos utilizar para controlar o servo motor.

Uma biblioteca é uma porção de código escrita por outra pessoa para fazer uma funcionalidade específica, reutilizando essas bibliotecas não precisamos escrever tudo novamente.

Na linha 1 é declarada a biblioteca para controle de servos com o comando **#include "Servo.h"**.
Na linha 3 é declarada uma variável apontando para a porta que se comunicará com o servo motor. Na linha 4 é declarada uma variável apontando para a porta ligada ao potenciômetro. A linha 5 declara uma variável do tipo Servo. Utilizaremos essa variável quando enviarmos comandos para o servo motor.

Além das portas de entrada e saída digitais, que funcionam apenas como ligada (HIGH) e desligada (LOW), o Arduino possui também a capacidade de ler dados analógicos. Os dados analógicos são valores que podem variar de 0v até 5v. Mas para ler dados analógicos precisamos utilizar algumas portas especiais do Arduino, as portas azuis destacadas na imagem abaixo:



Essas portas começam com a letra A (ex: A0, A1, A2, A3, A4 e A5). Por isso o potenciômetro está ligado na porta A0.

Na função de **setup** são configuradas as portas. Na linha 8 temos um elemento novo, o **Serial.begin(9600)**. Como o Arduino não possui uma tela, fica difícil saber quais valores estão sendo lidos dos componentes, como o potênciômetro. Esse comando permite que o Arduino se comunique com o computador e mostre informações do Arduino no programa Serial Monitor no computador. Isso é muito útil para identificar problemas e saber o que o Arduino está fazendo.

Na linha 9 é configurada a porta do potenciômetro como sendo uma porta de entrada (INPUT) utilizando o comando **pinMode(potenciometro, INPUT)**;
Por último, na linha 10 o servo é configurado para responder na porta 6 (portaServo).

Com as configurações feitas no **setup** vamos ver a lógica principal dentro do **loop**.

A primeira coisa que precisamos fazer é saber em qual posição está o potenciômetro. Isso é feito na linha 15. O comando `int valorPot = analogRead(potenciometro);` obtém o valor da posição do potenciômetro e guarda na variável valorPot, essa posição pode variar entre 0 e 1023. O valor do potenciômetro varia entre 0 e 1023, mas o valor necessário para controlar o servo deve ser entre 0 e 180 (ângulo no qual o servo deve virar). Para fazer essa conversão na linha 15 é utilizado o comando map. Ele faz a conversão automaticamente. As linhas 24 e 25 mostram o ângulo calculado no Serial Monitor. Por fim, na linha 28 o programa envia para o servo o ângulo que ele deve ser posicionado.

Comandos Aprendidos

Comando	**Descrição**
Serial.begin(9600)	Configura a comunicação entre o Arduino e o computador
Serial.println(“mensagem”)	Mostra uma mensagem no Monitor Serial do computador
analogRead(porta)	Captura um valor analógico de uma porta (valor entre 0 e 1023)
#include “Servo.h”	Importa a biblioteca para controle de servos
servo.write(angulo)	Posiciona o servo motor em um ângulo específico

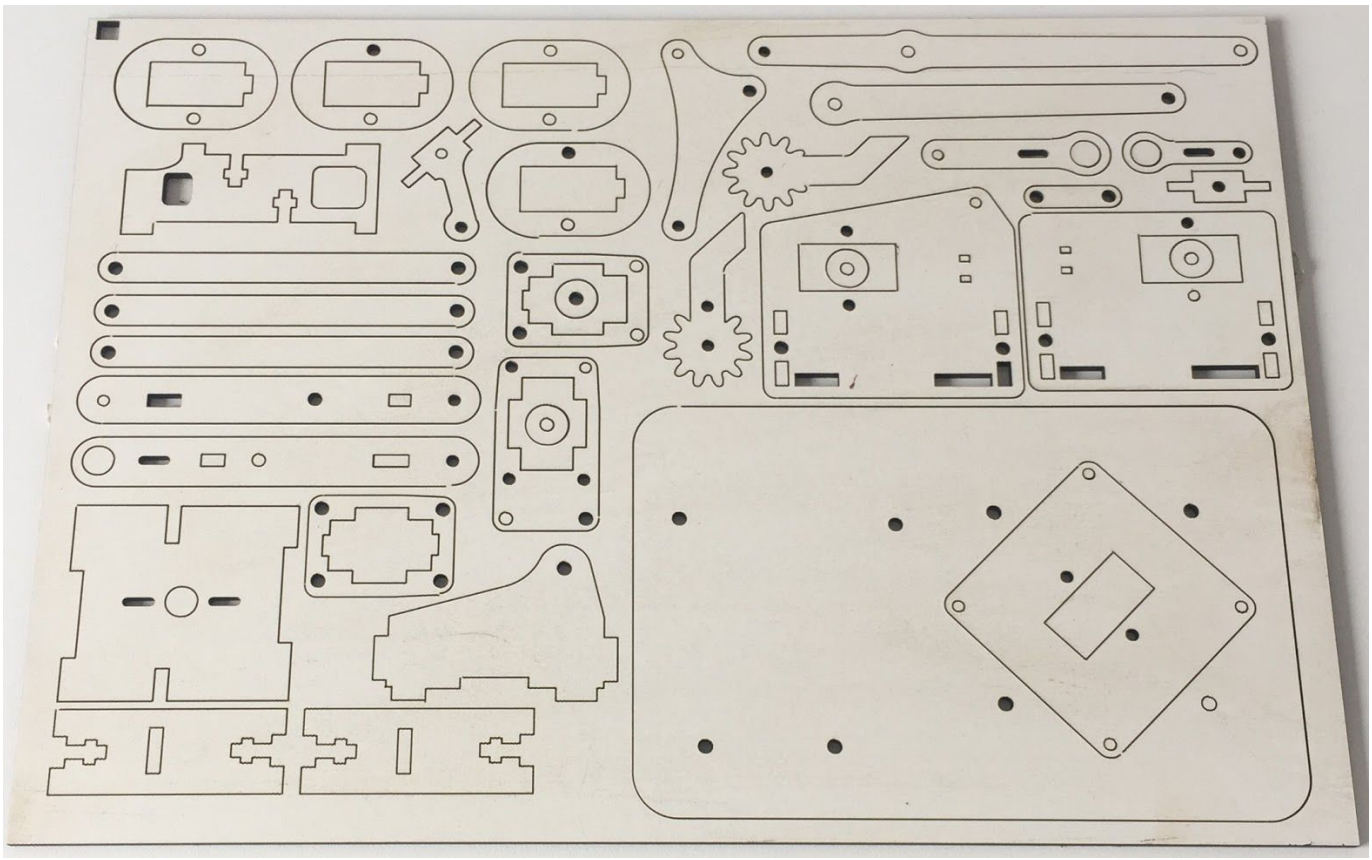
Garra Robótica

Agora que já conhecemos os fundamentos de como trabalhar com o Arduino vamos montar a nossa garra robótica!

A garra consiste de 3 partes fundamentais:

- Mecânica: partes físicas e parafusos
- Eletrônica: Arduino, protoboard e servos
- Programa/Código: Fará a nossa garra funcionar!

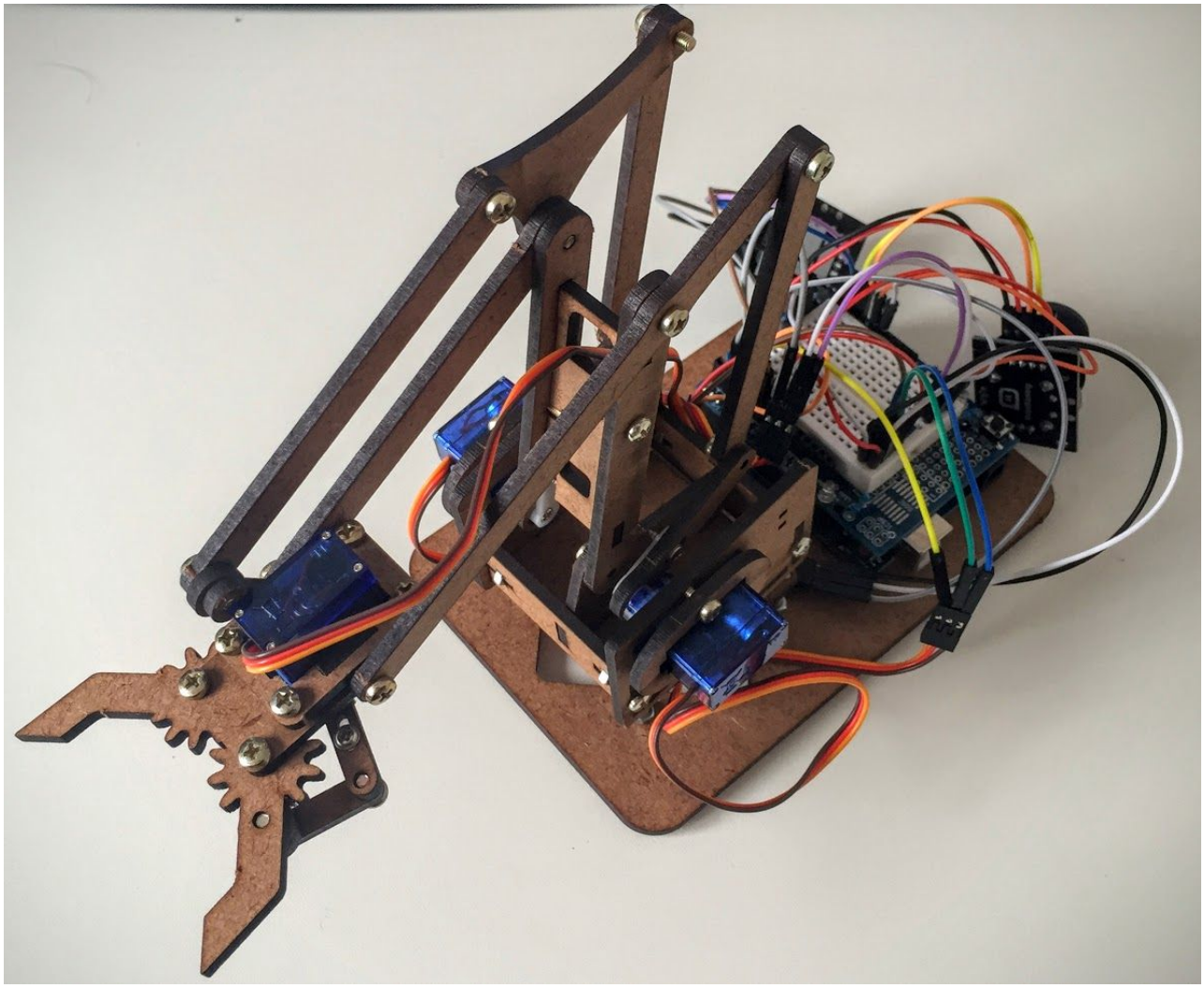
A imagem abaixo mostra a as partes mecânicas da garra.



Modelo 2D

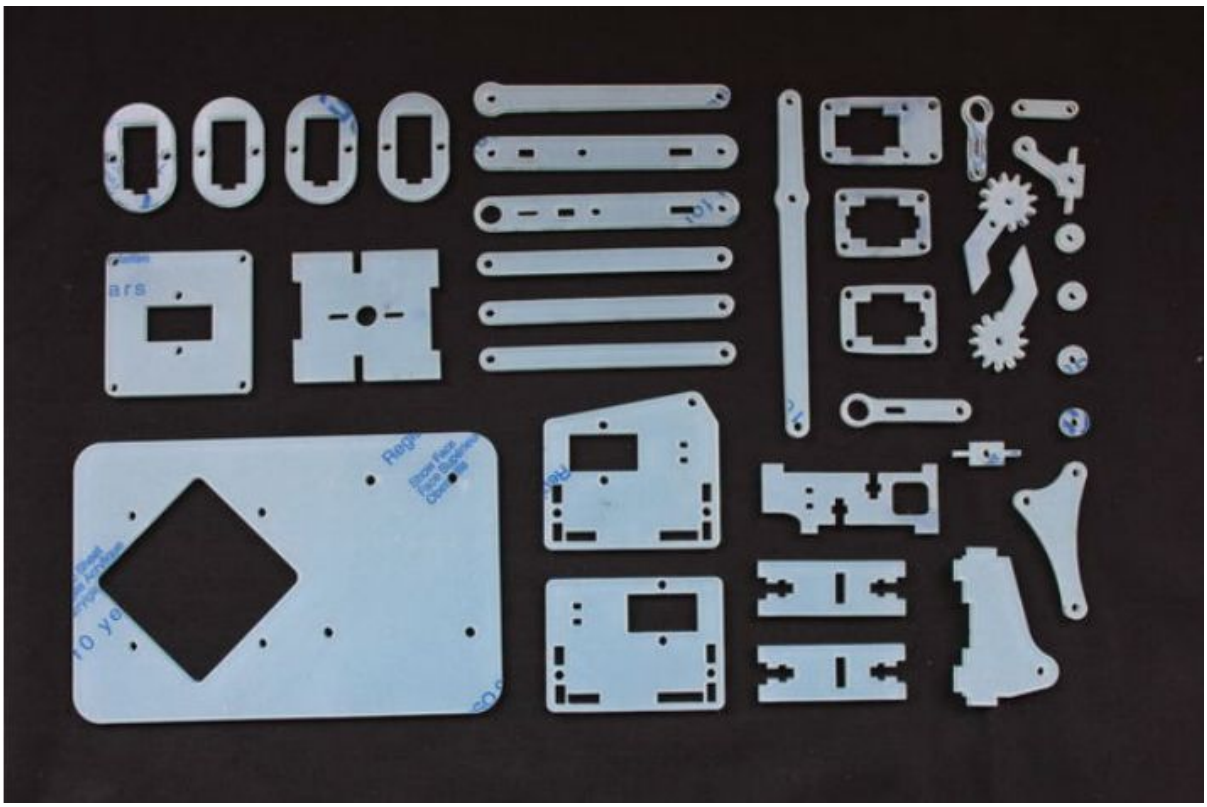
Os arquivos para corte a laser podem ser encontrados em:
http://www.thingiverse.com/thing:360108

Imagem do braço montado



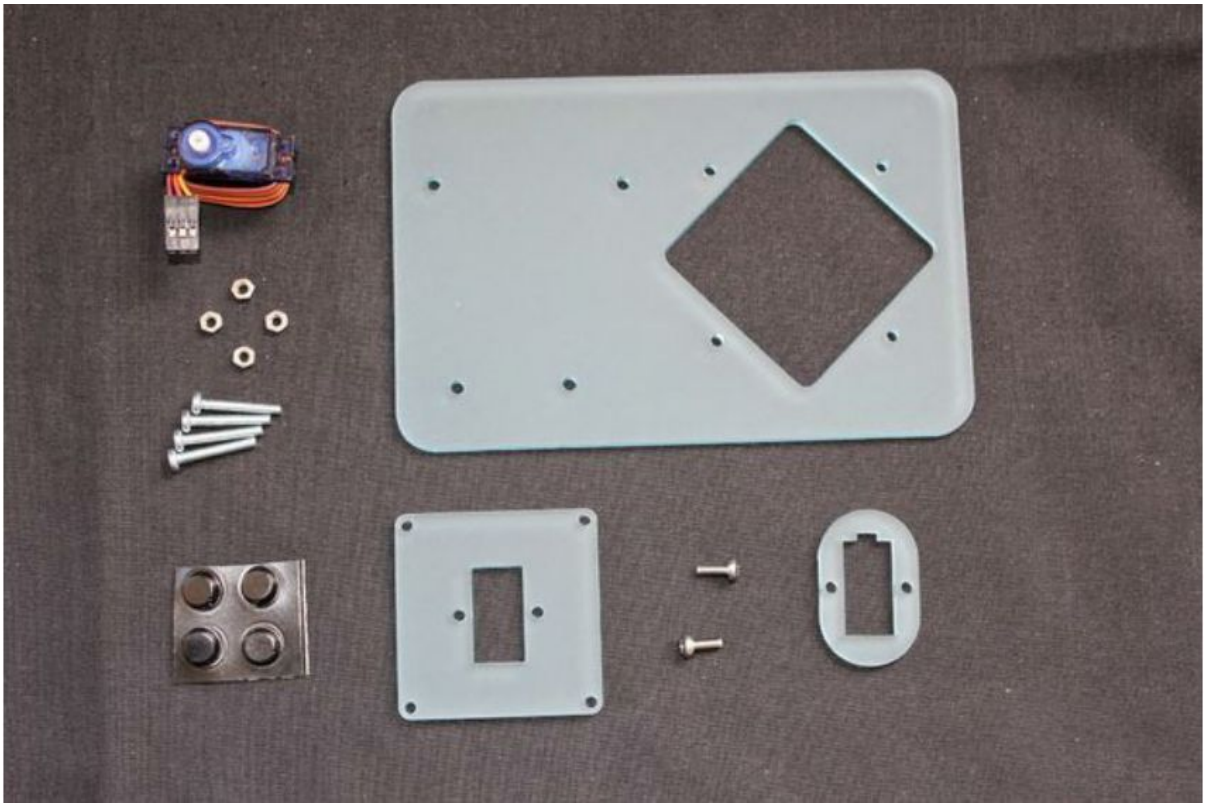
Montagem

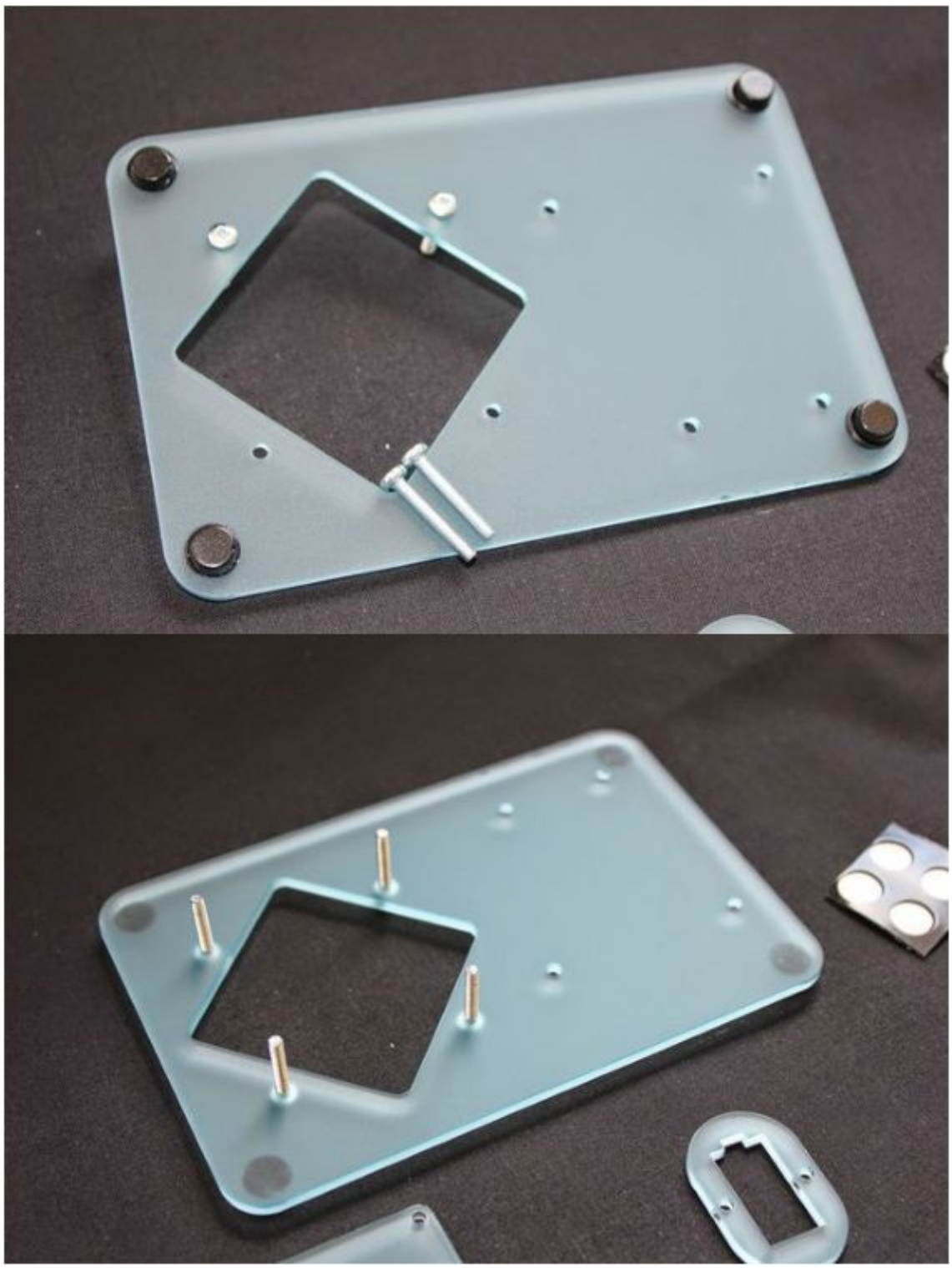
Destacando todas as peças

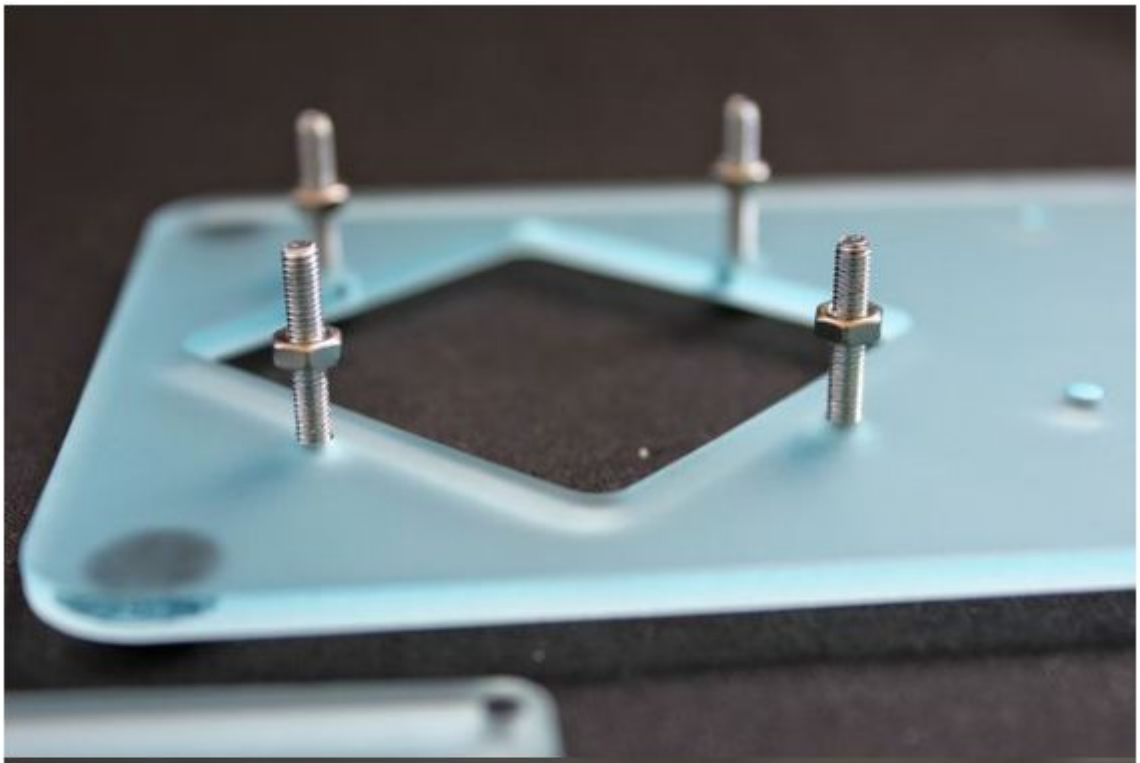


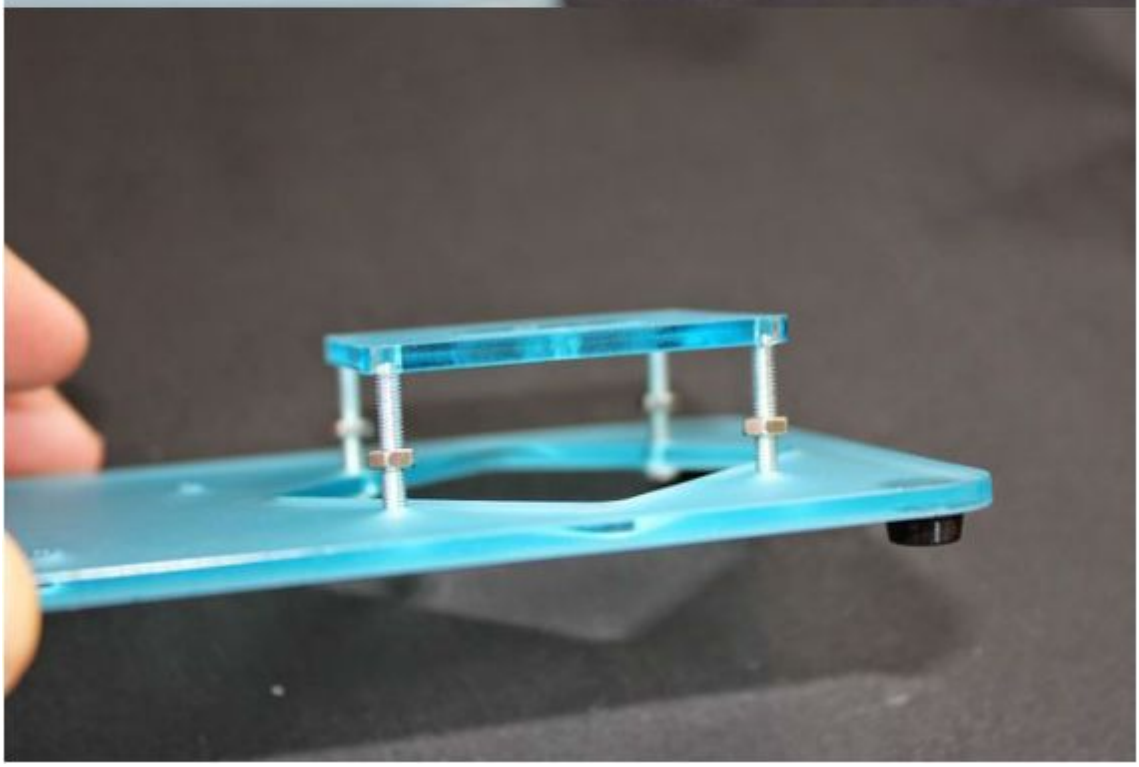
Destaque as peças da madeira de suporte. Tenha cuidado ao destacar pois várias peças são finas e podem se quebrar facilmente se muita força for aplicada.

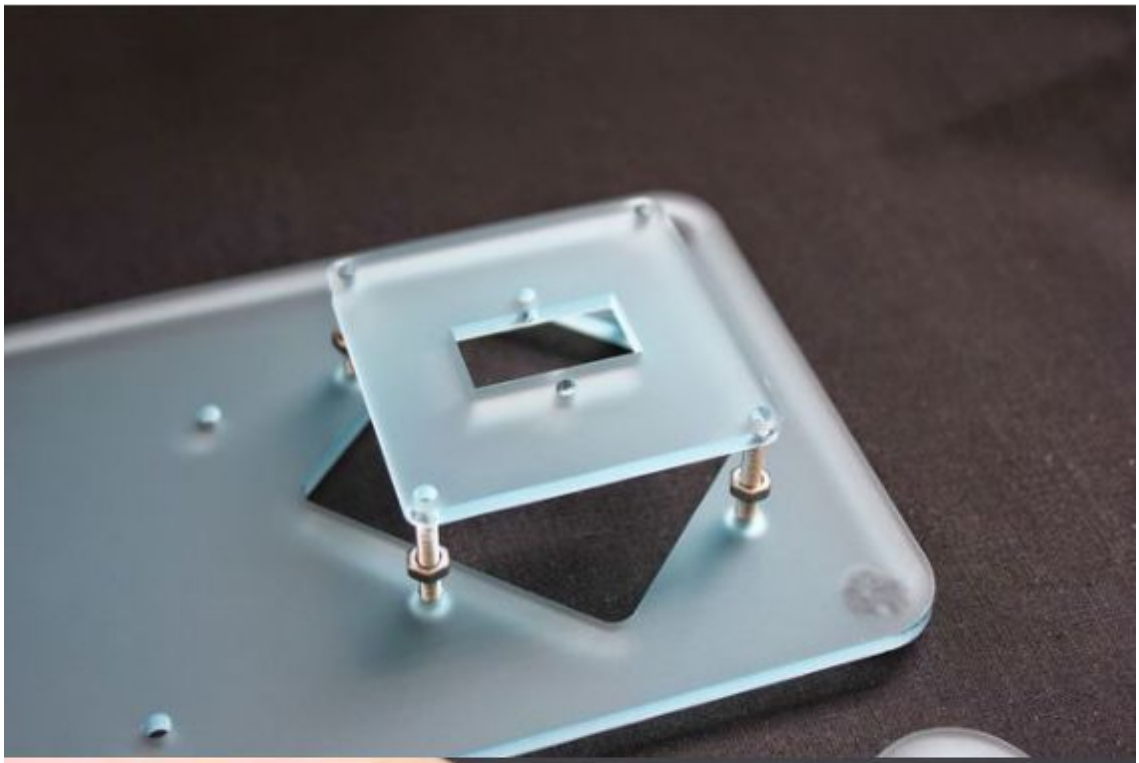
Montando a Base

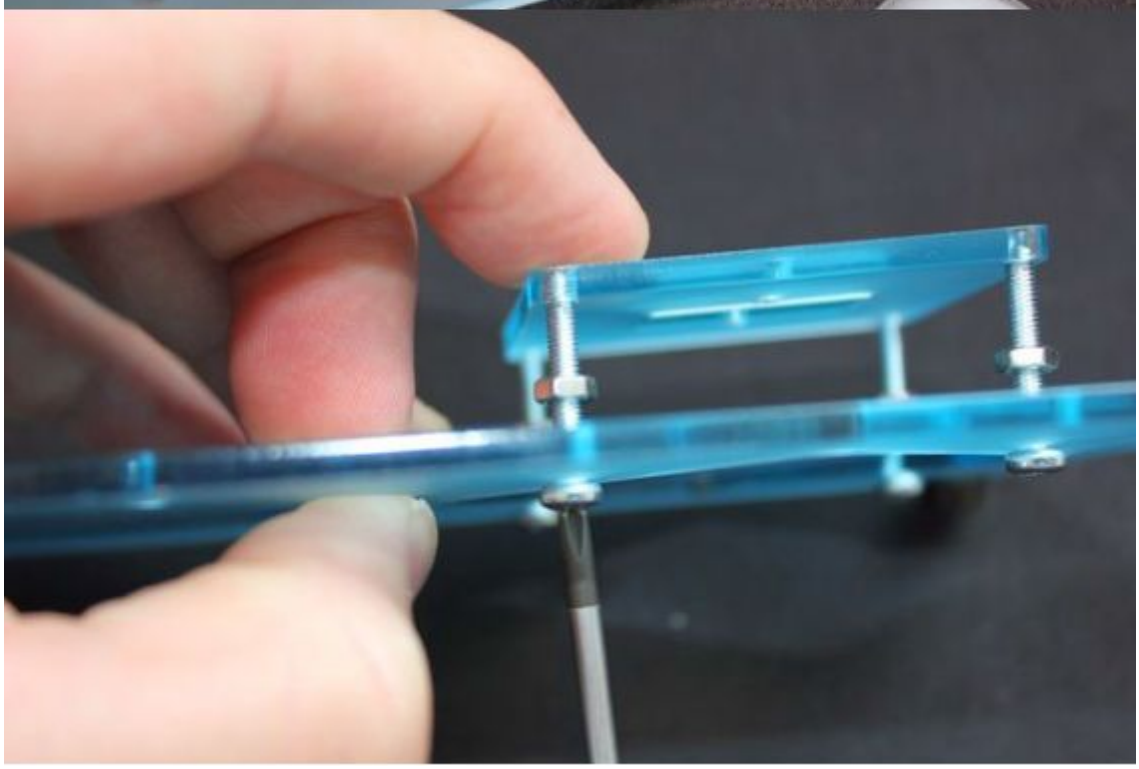


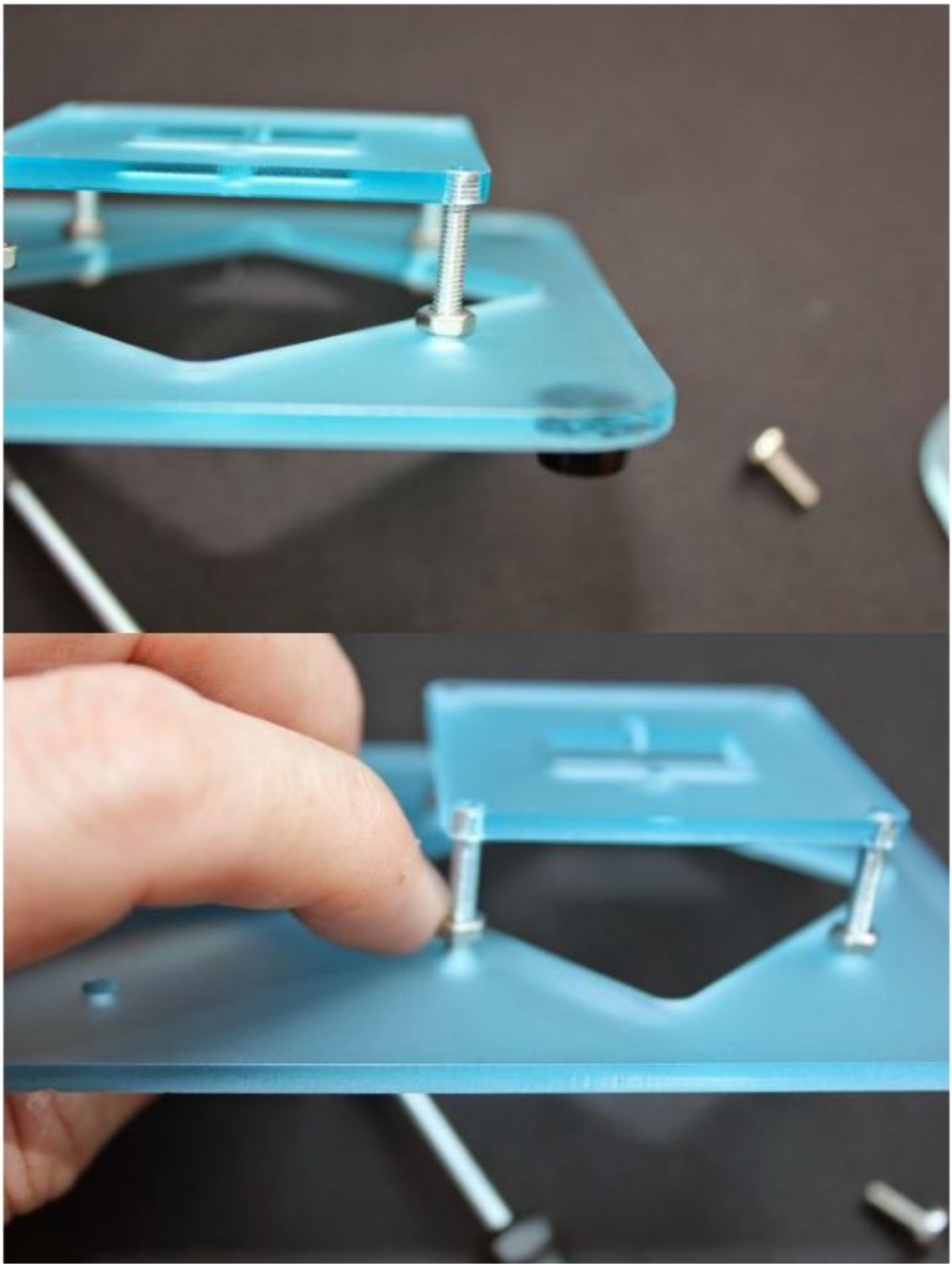


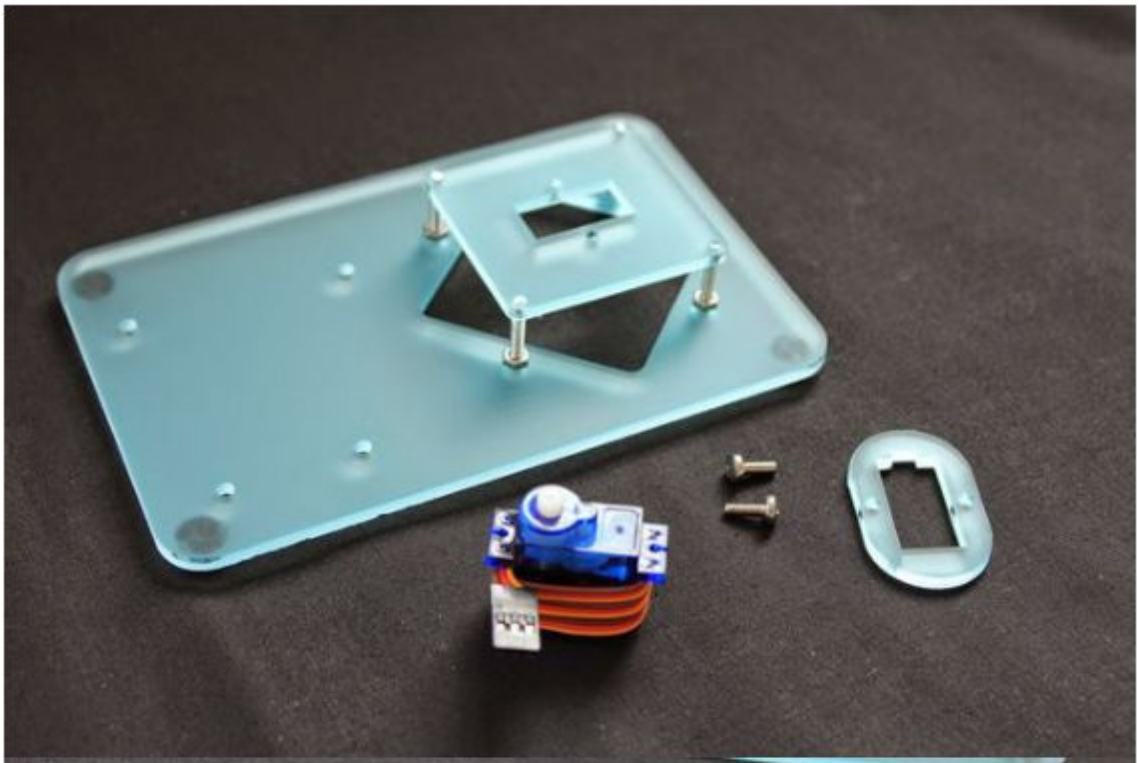


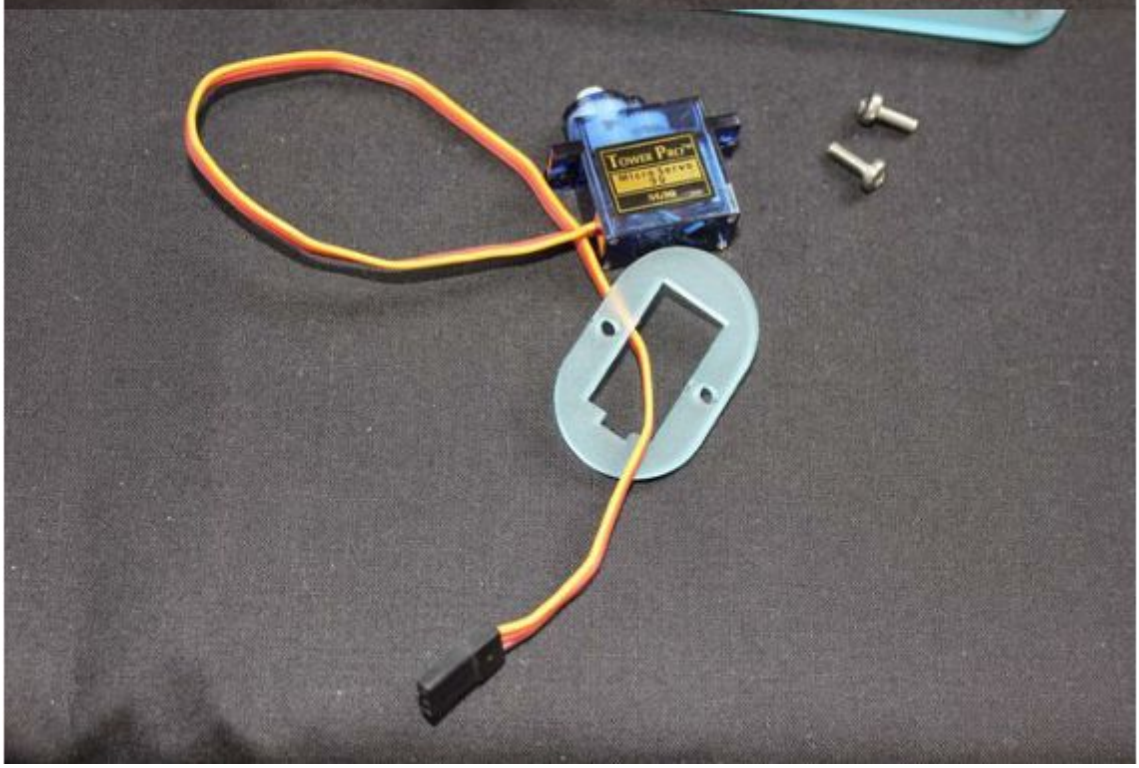


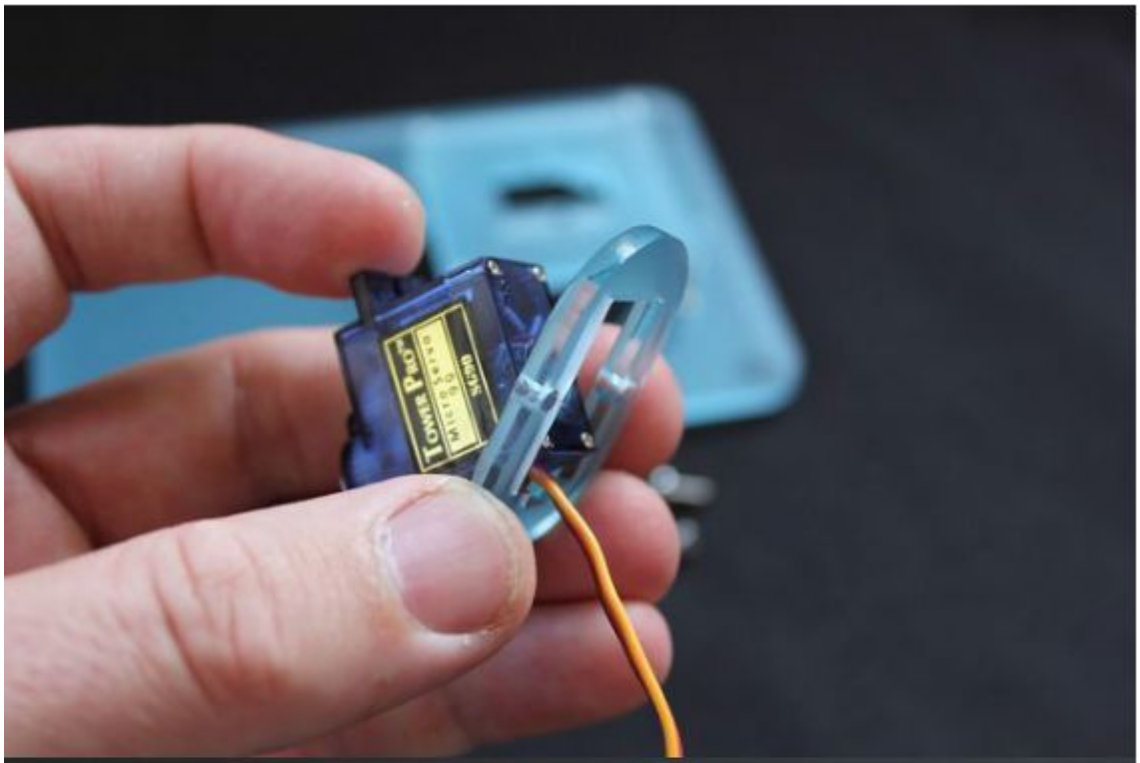




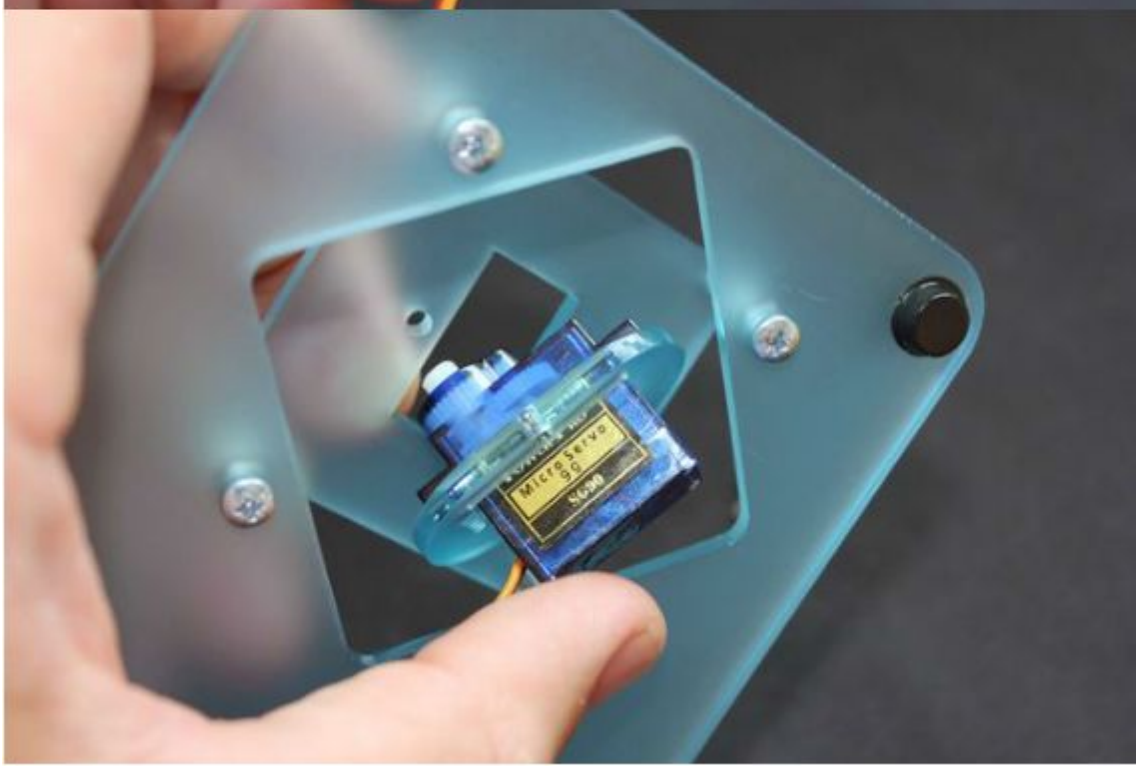


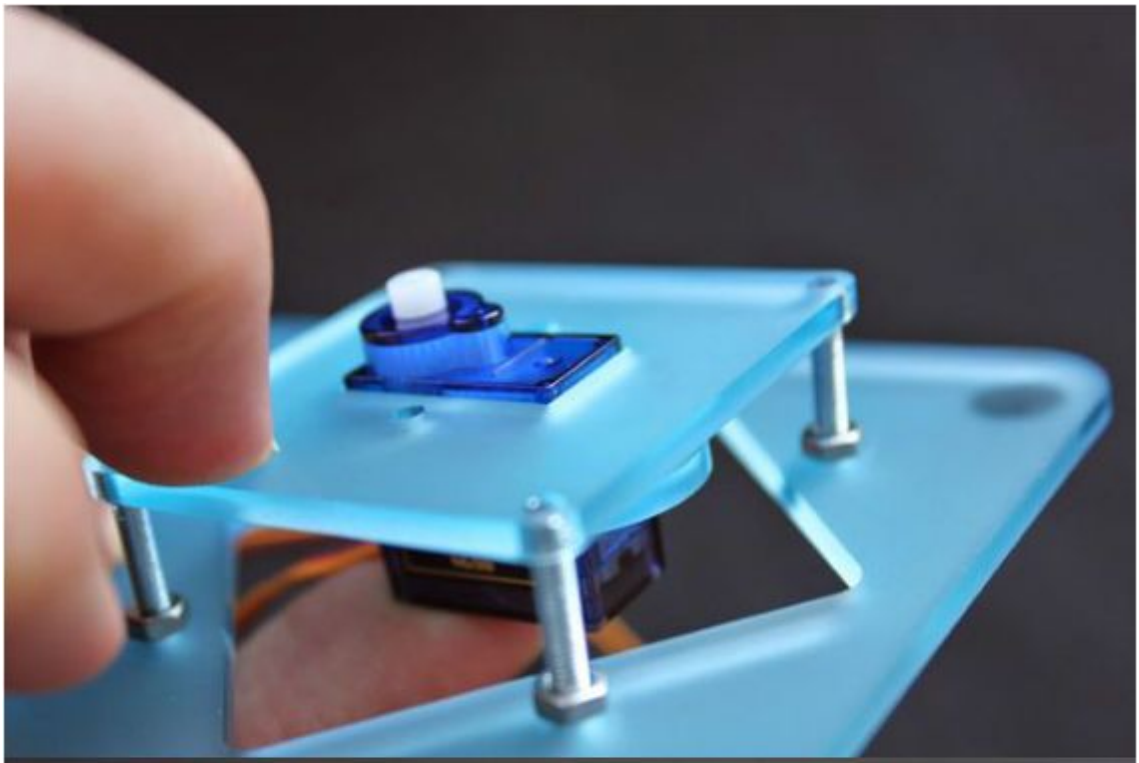


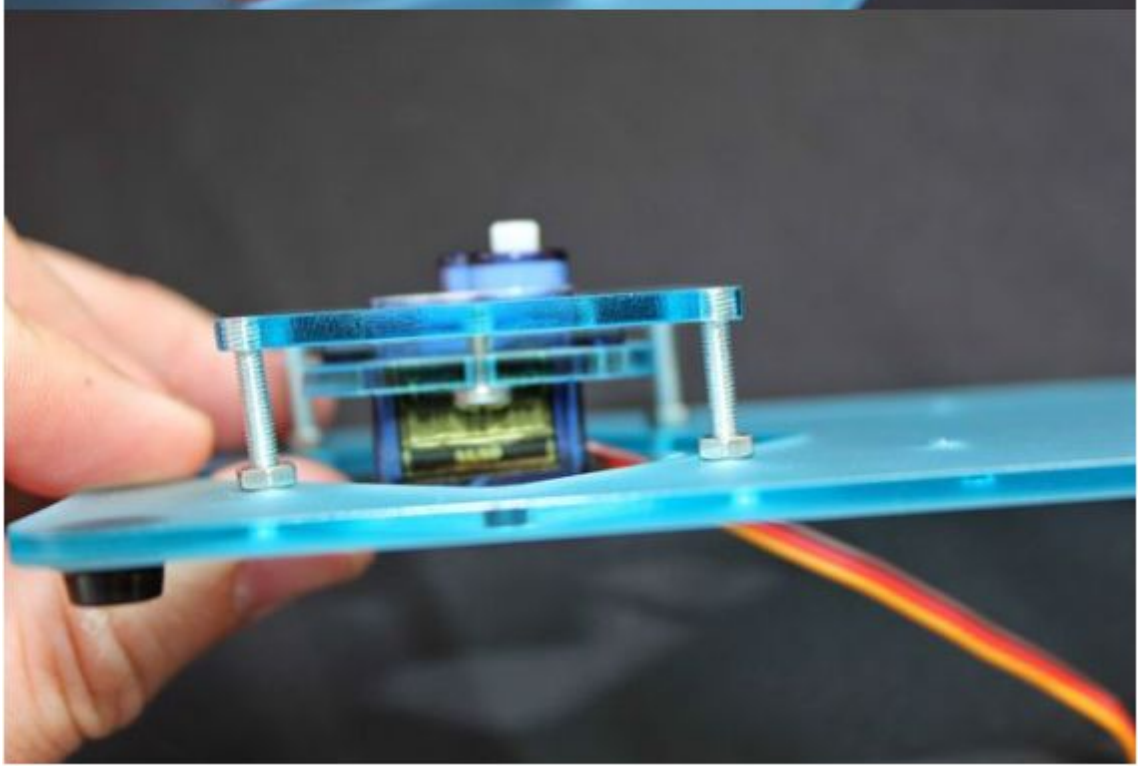

TOWER PRO

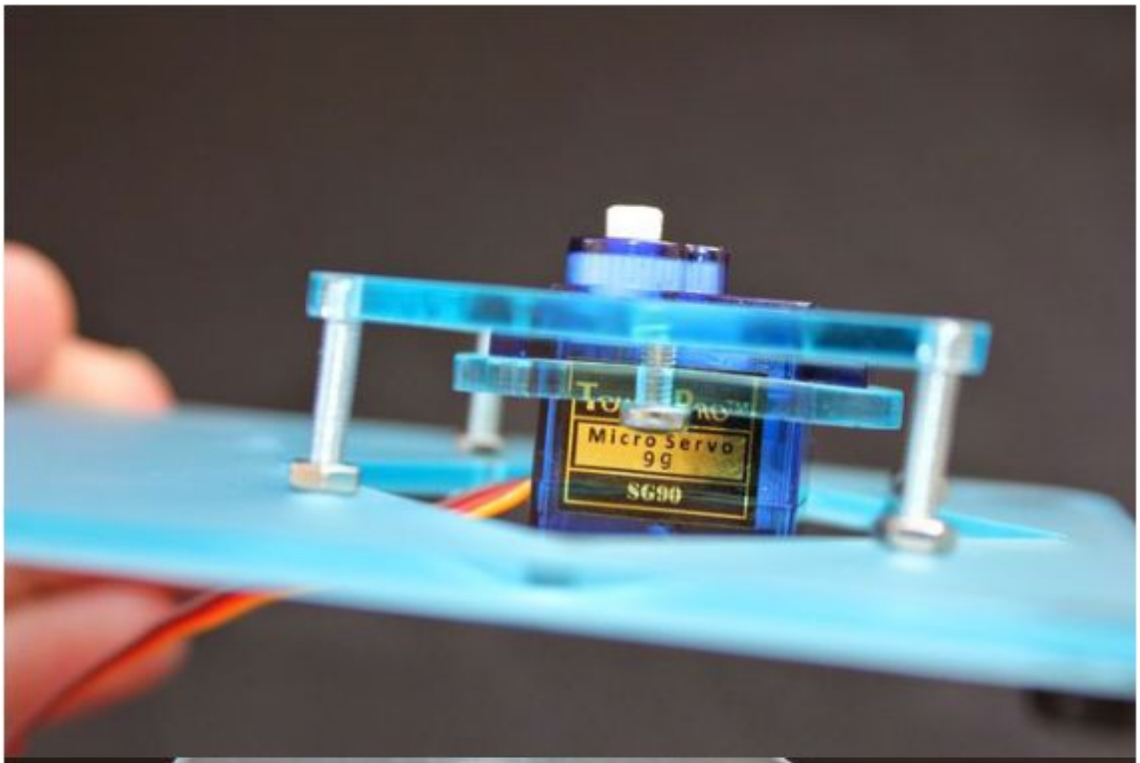

TOWER PRO
MICRO SERVO
SG90

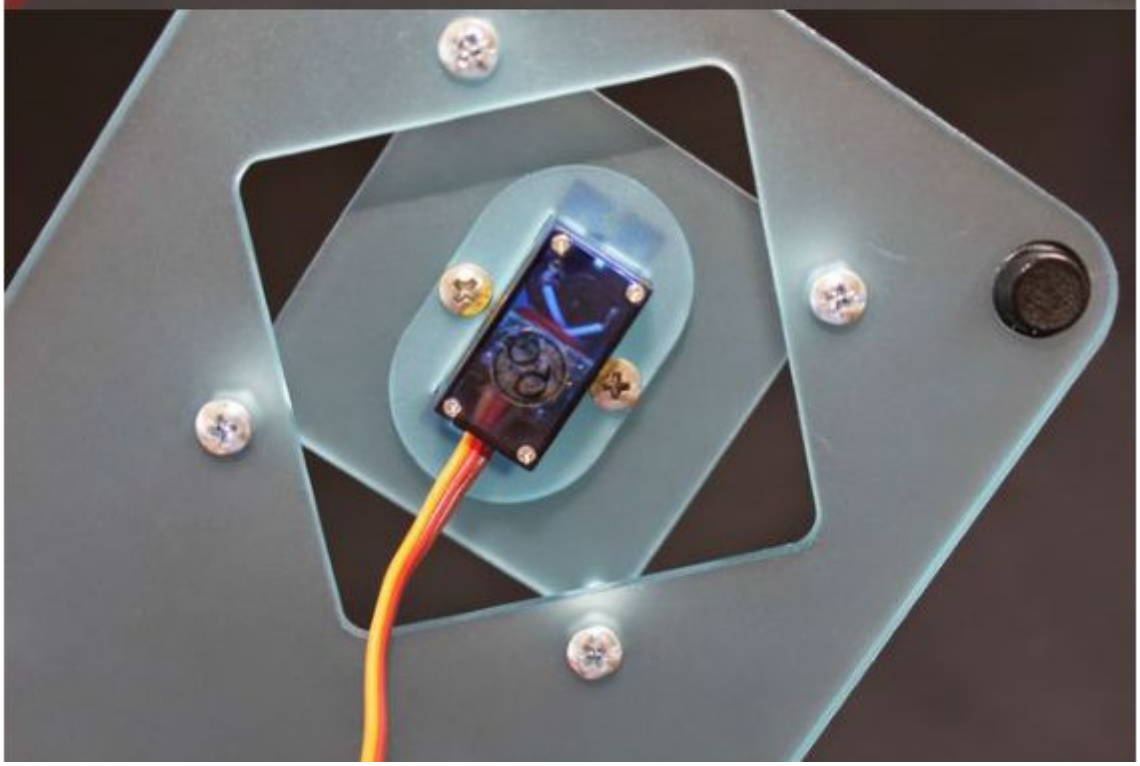

Micro Servo
9g
SG90

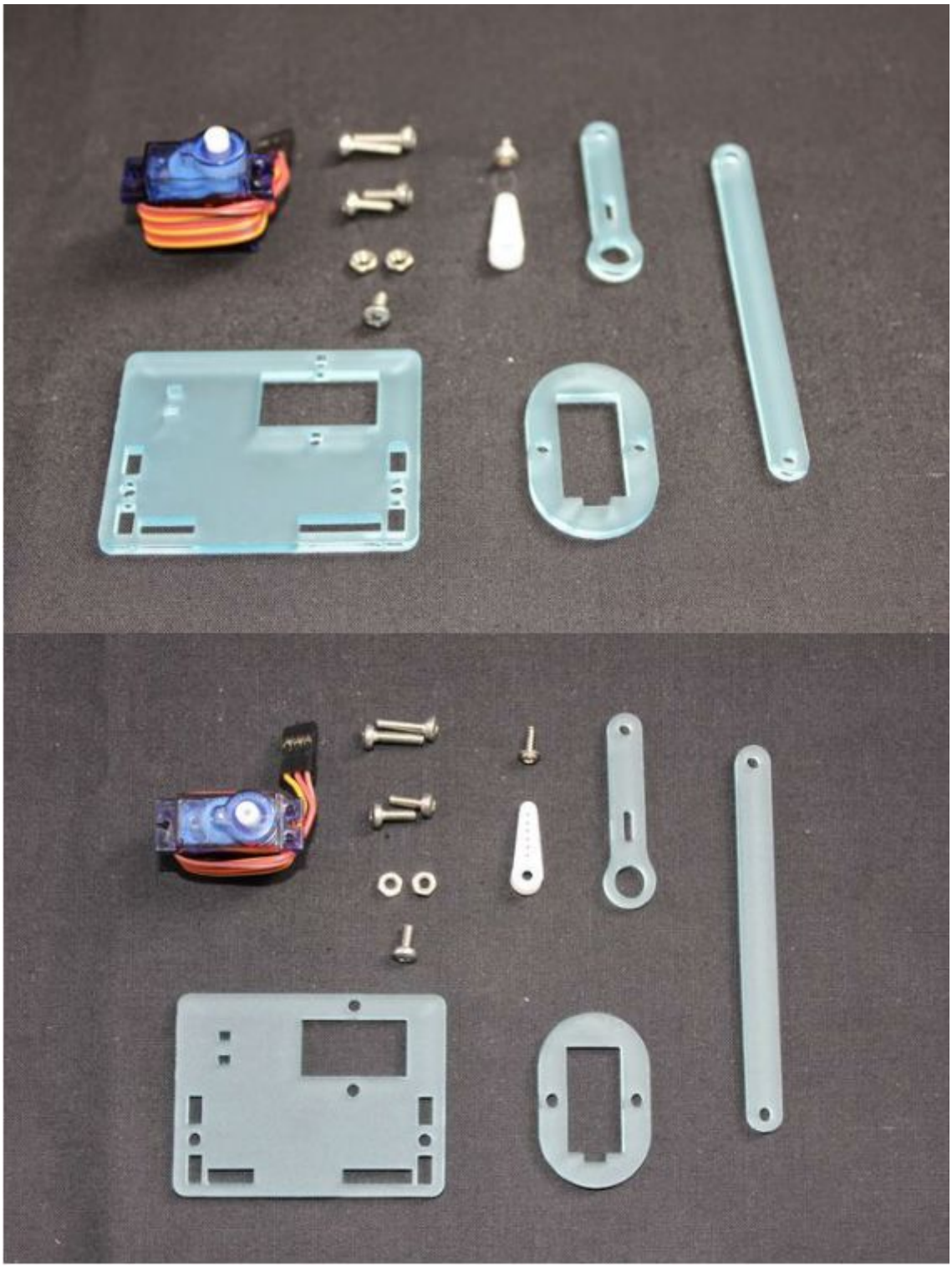

Micro Servo
9g
SG90

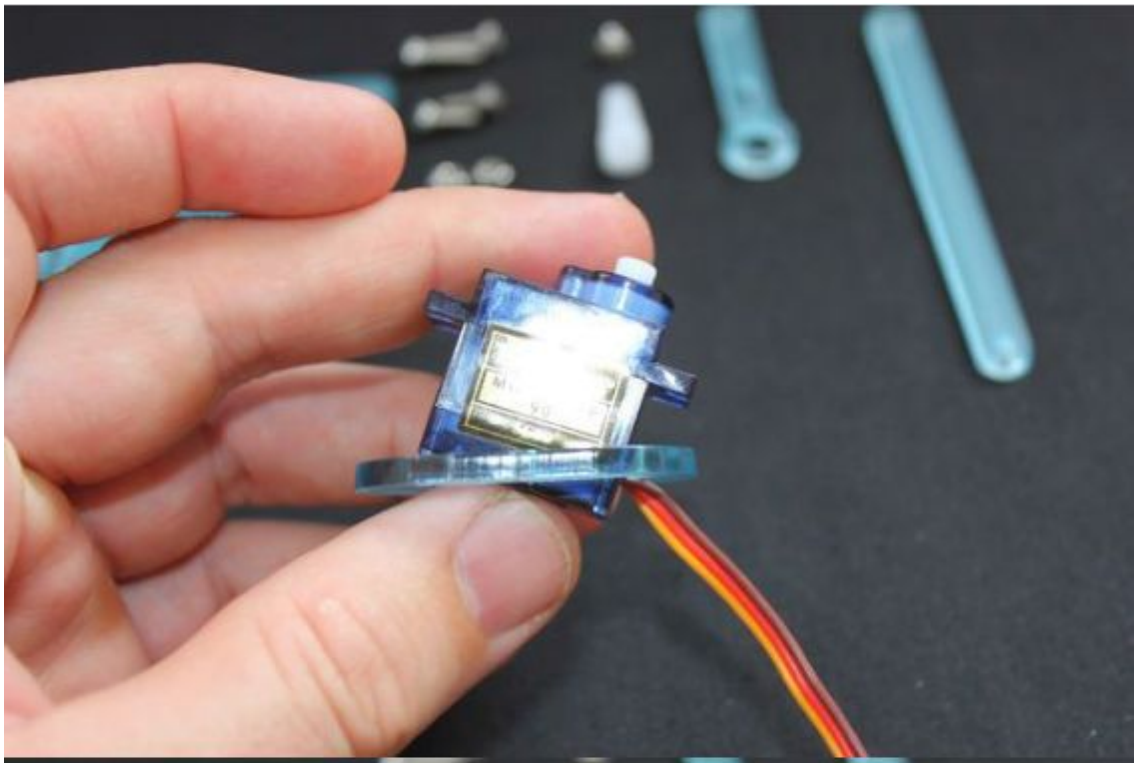


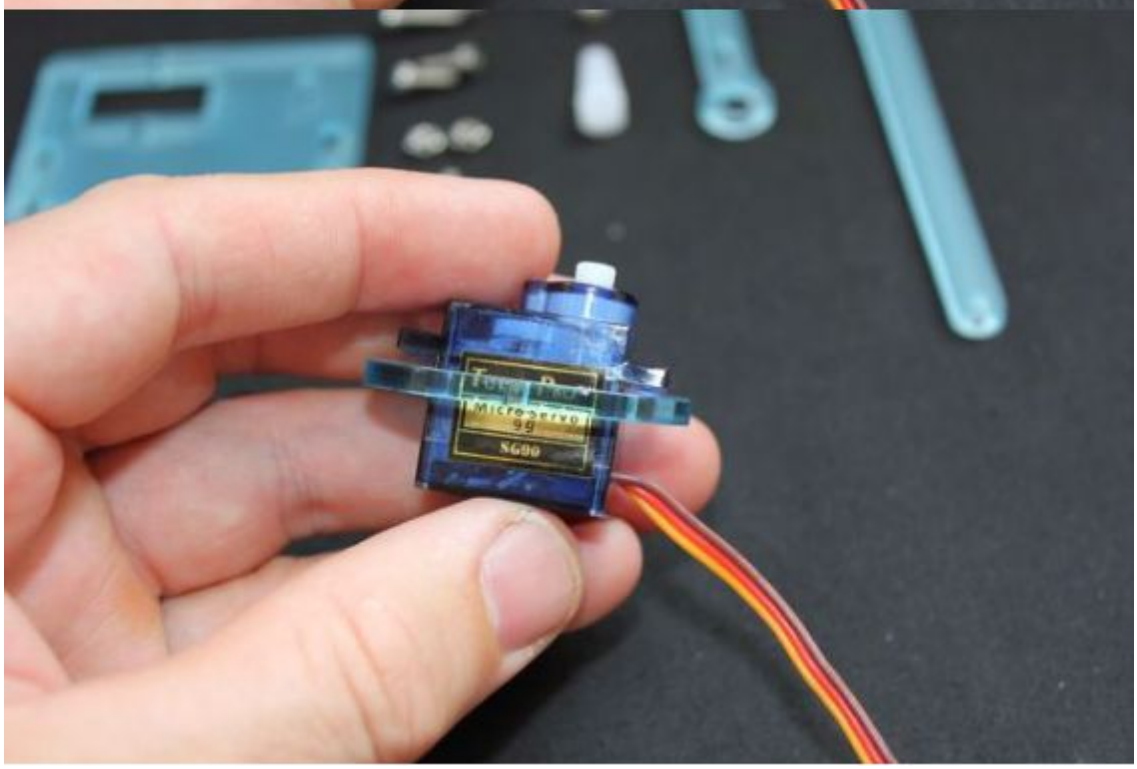


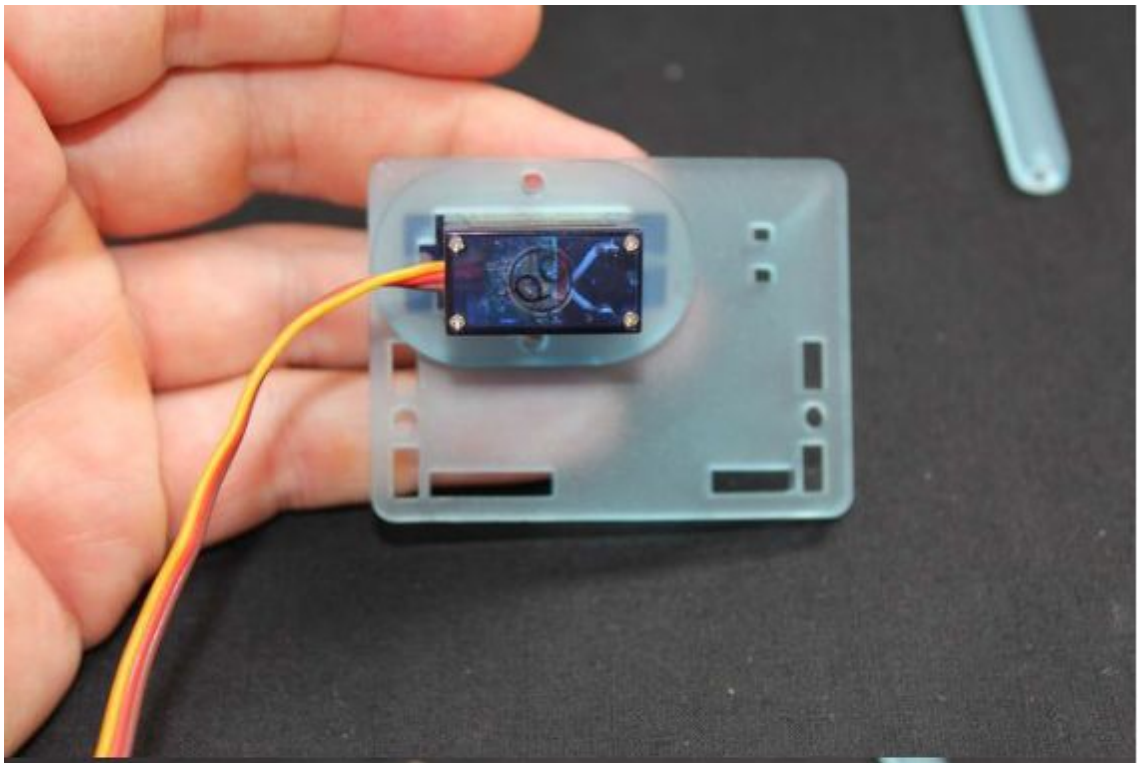

Micro Servo
9g
SG90

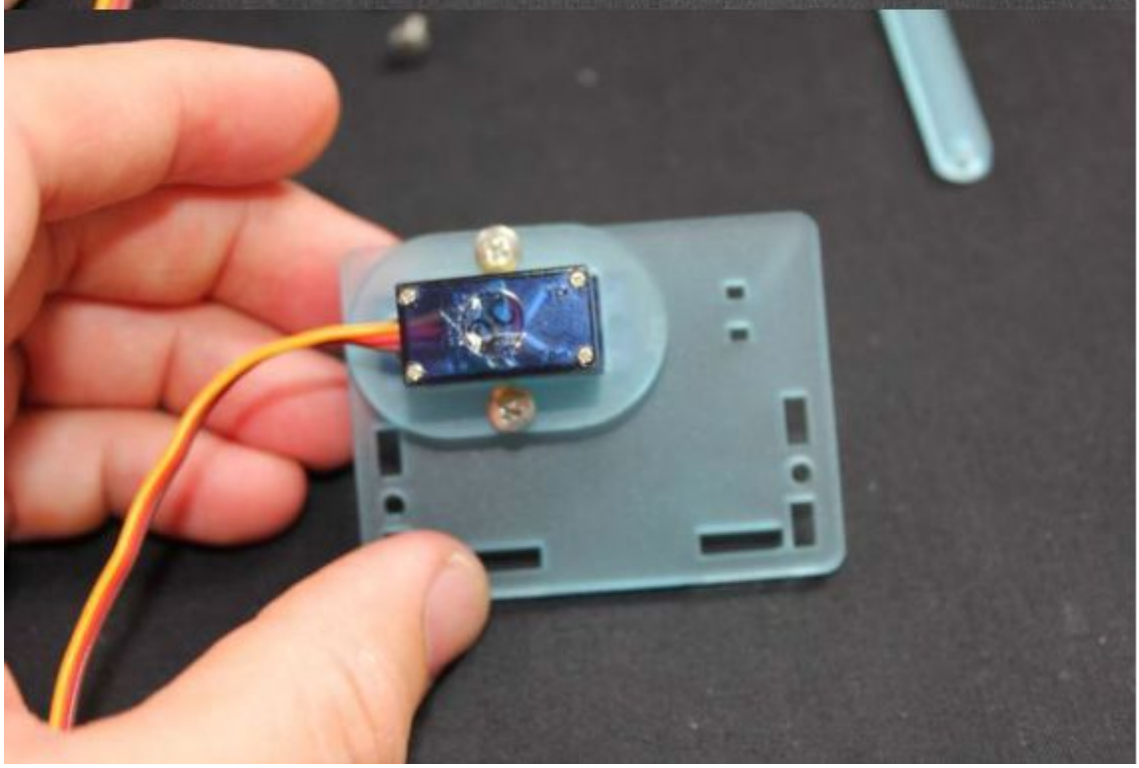


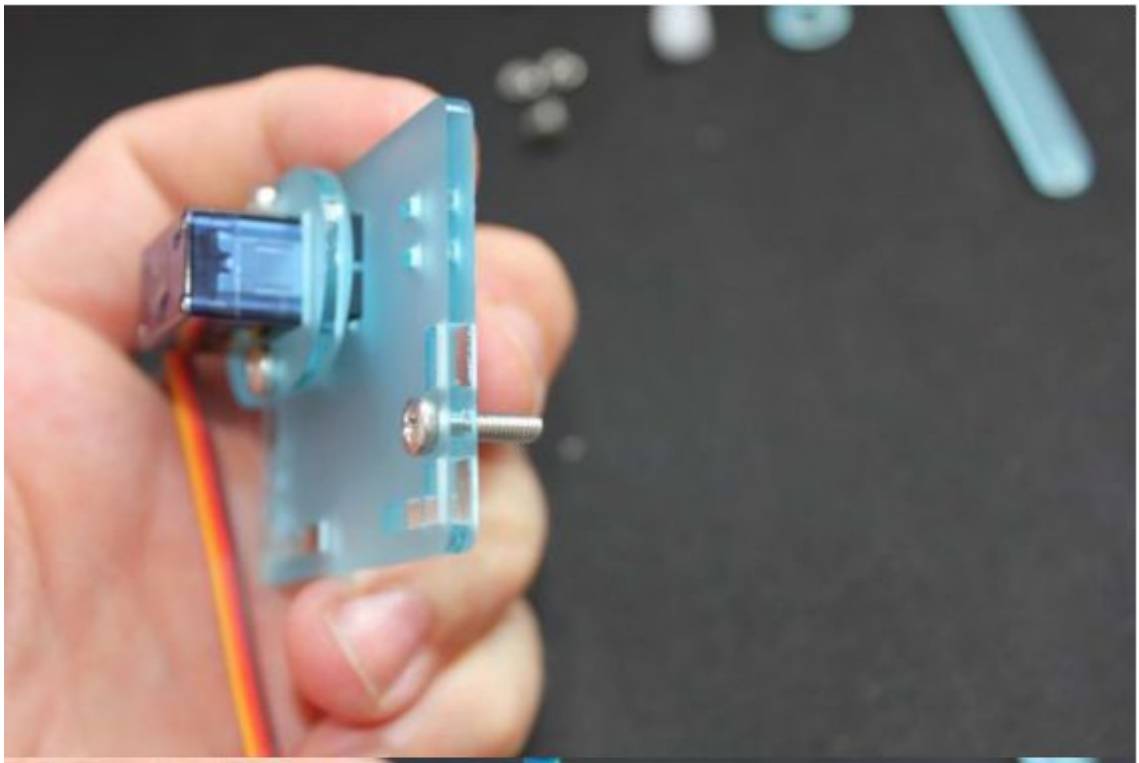


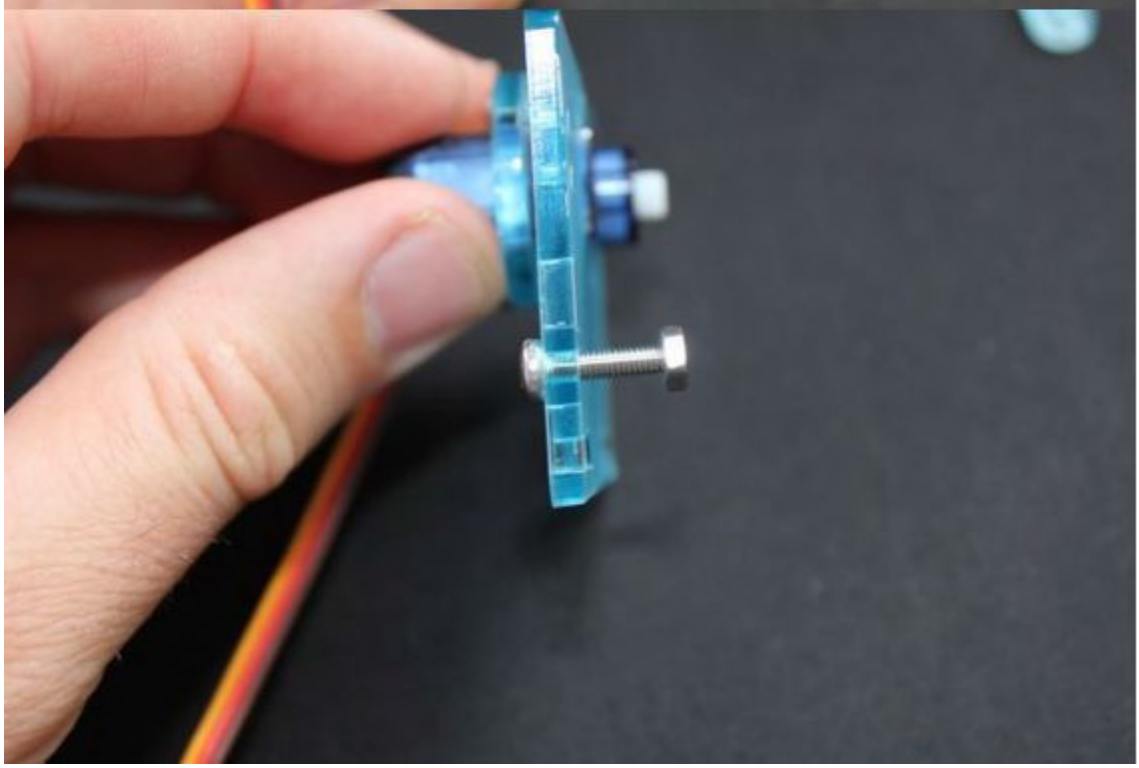


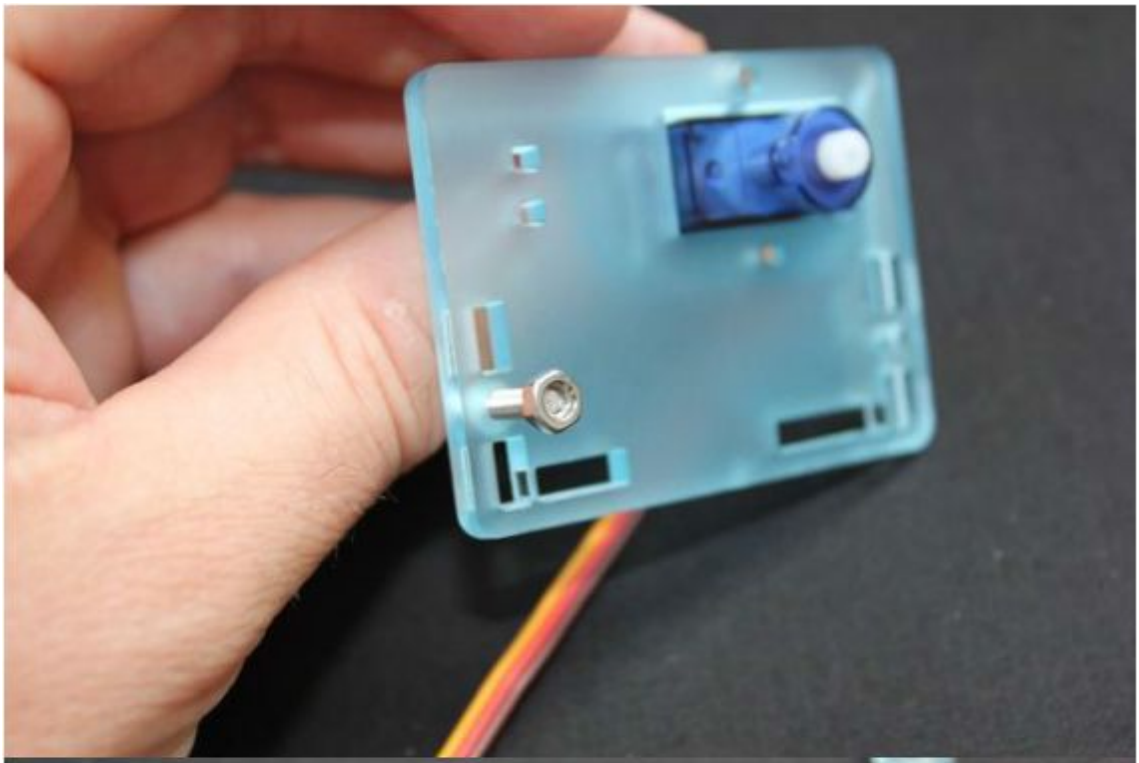

9g
SG90

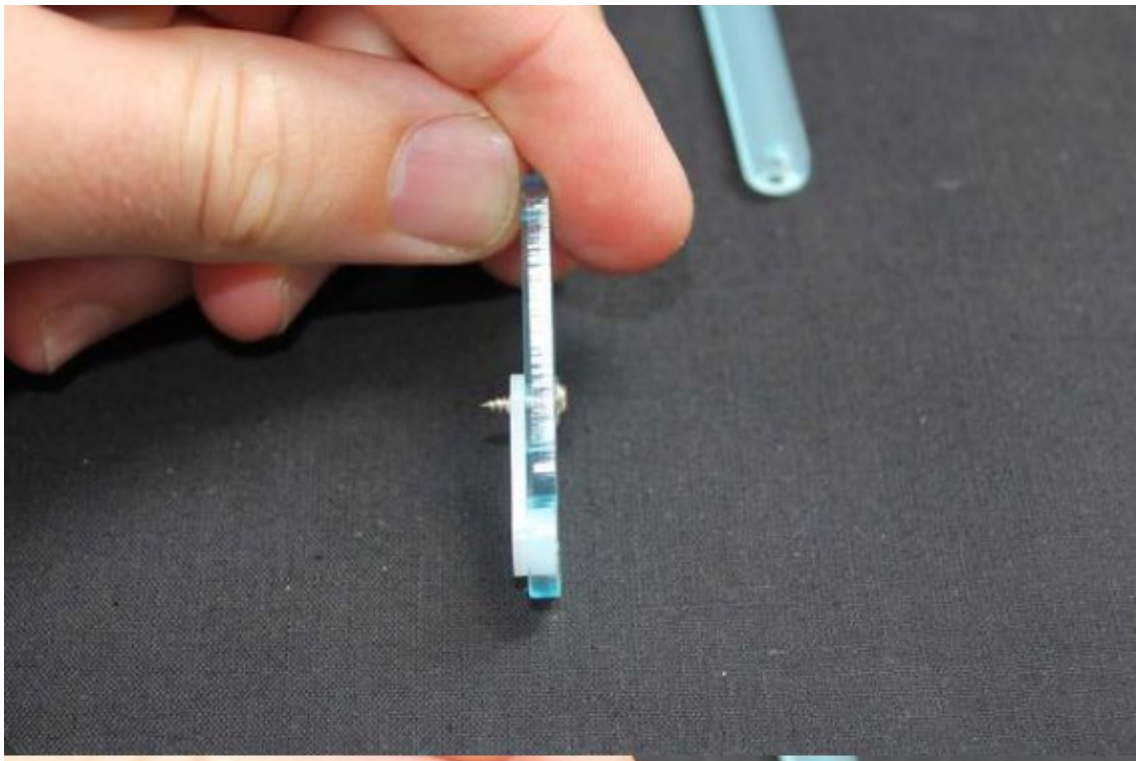


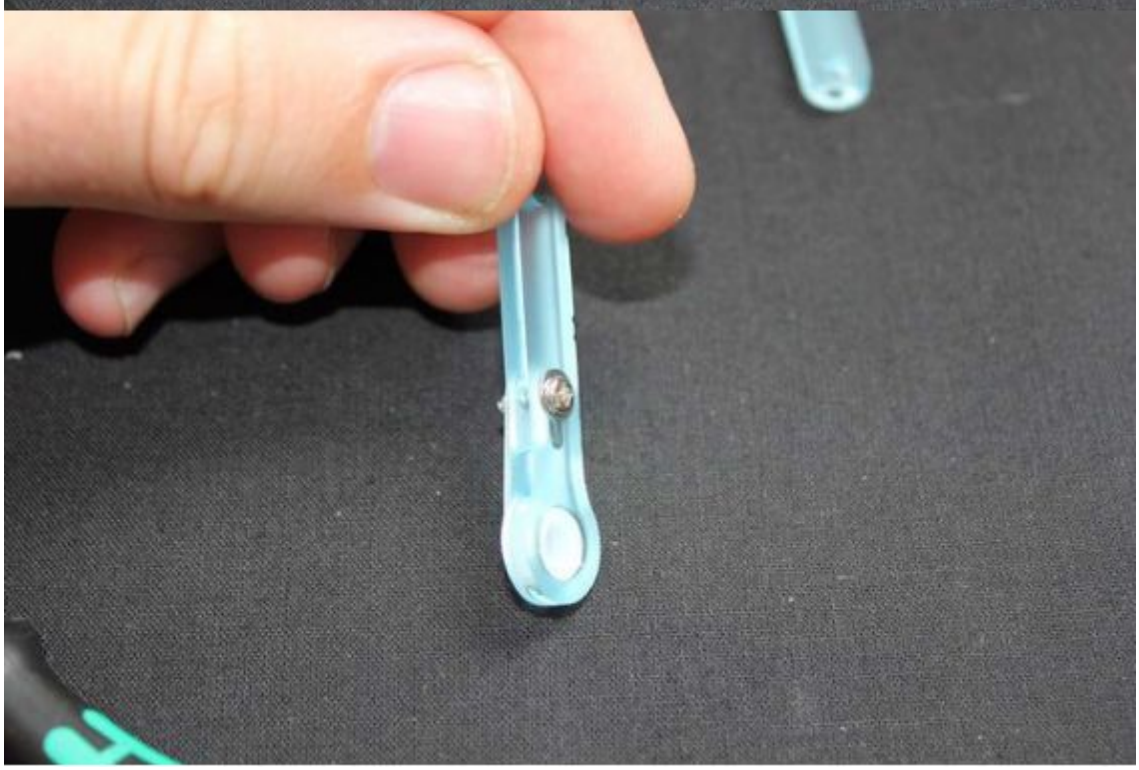


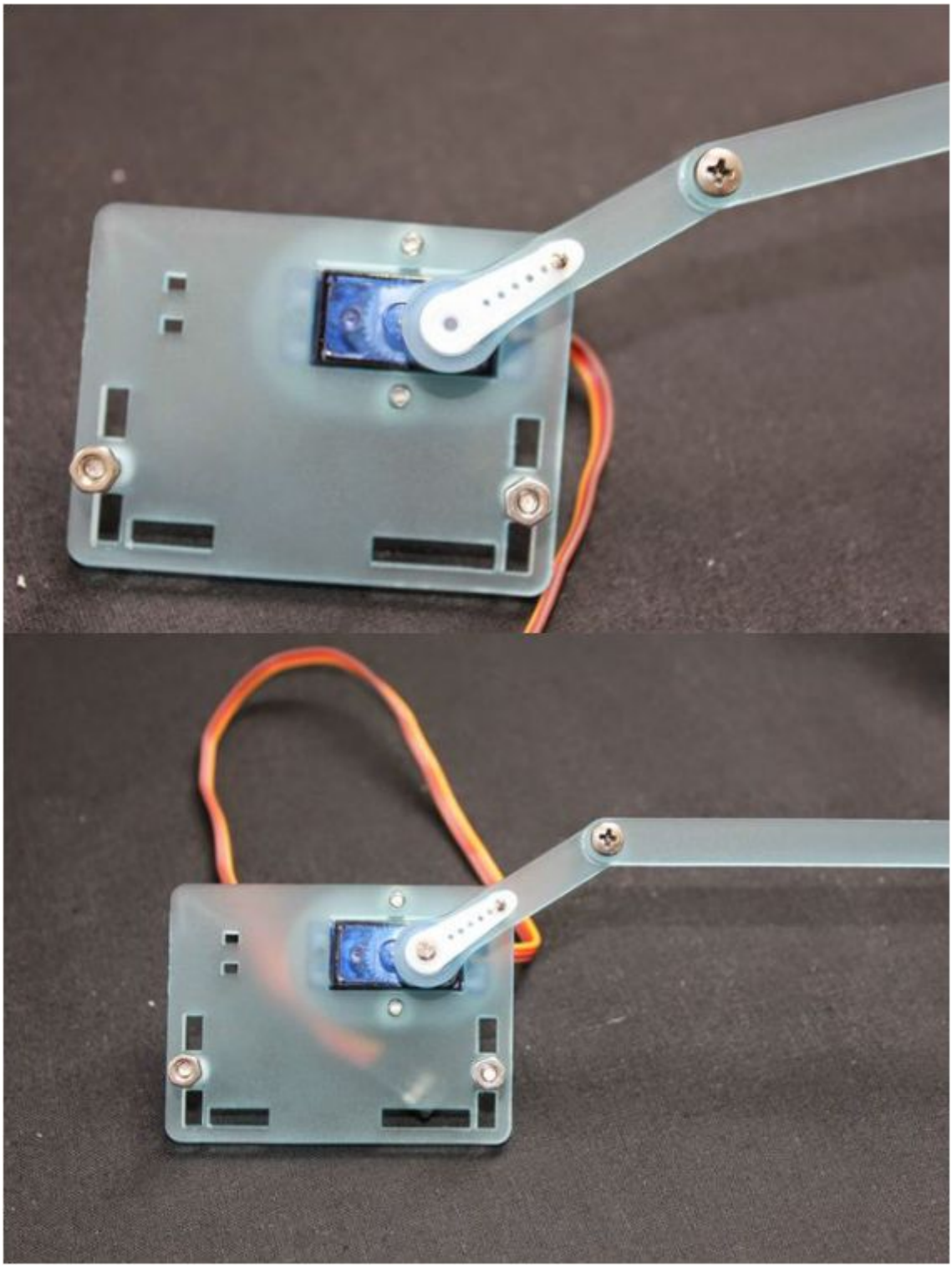




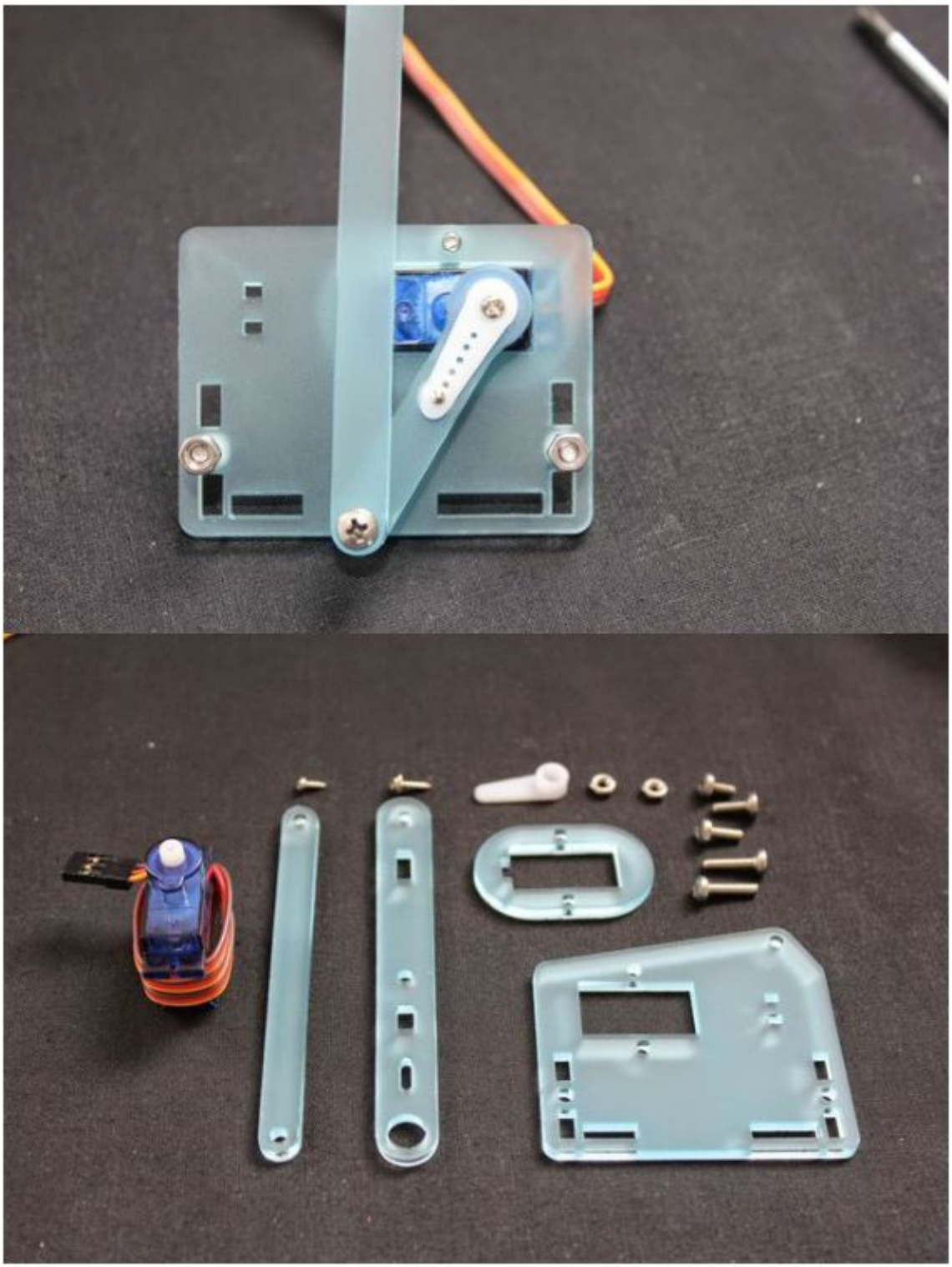




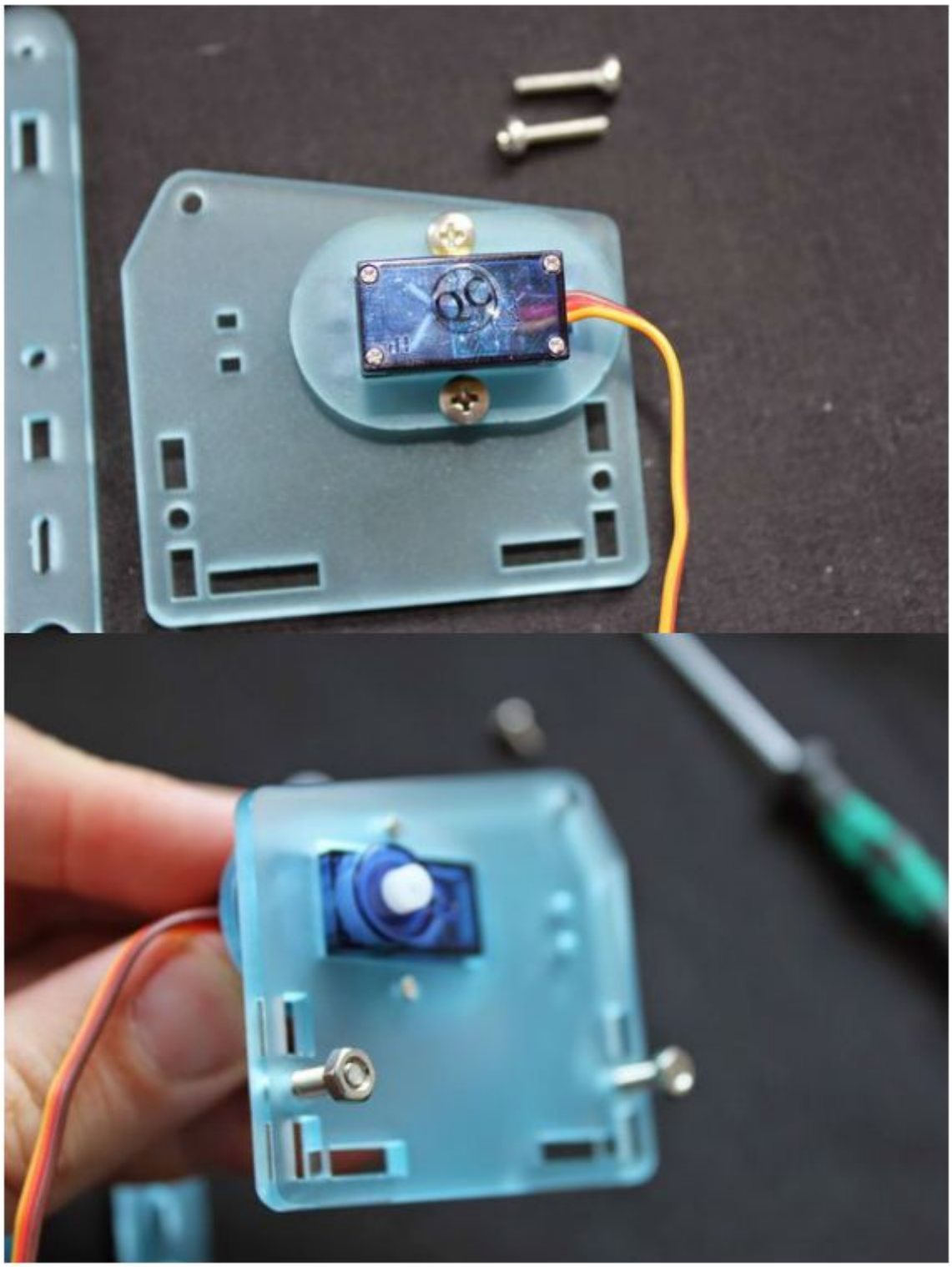


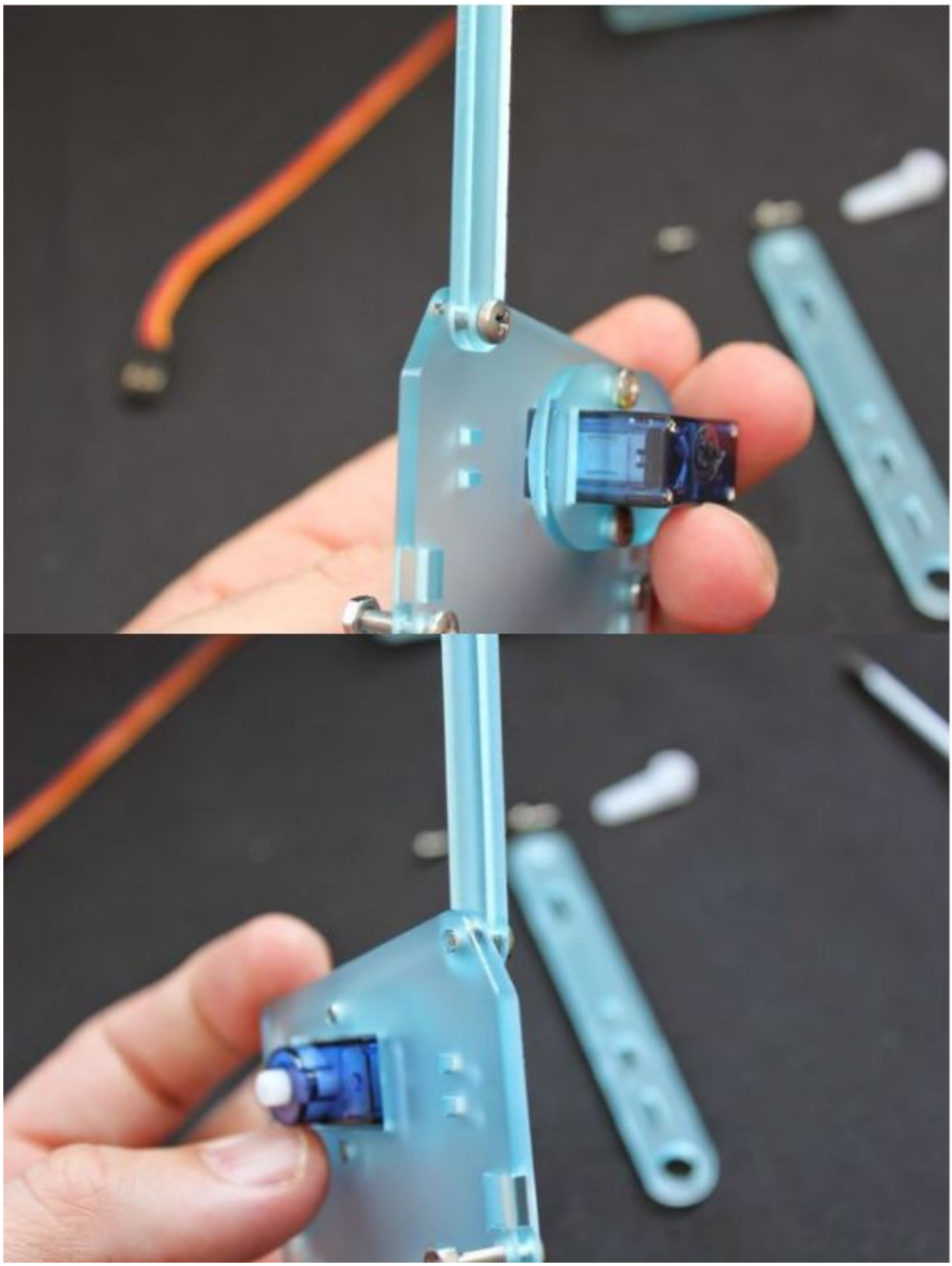


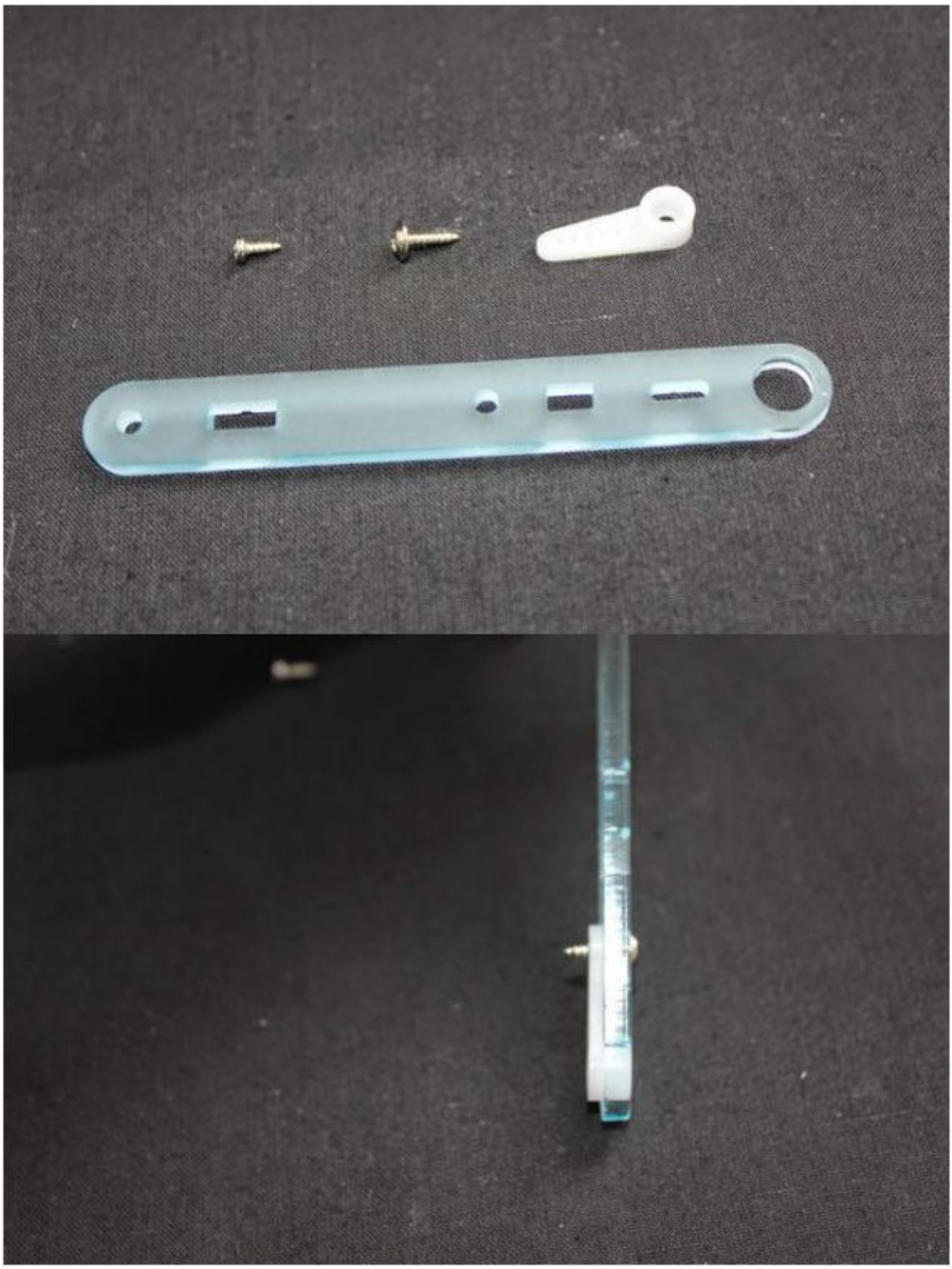
ATENÇÃO: Regule o servo para que o braço na posição mostrada na imagem corresponda a metade da rotação do servo!!!

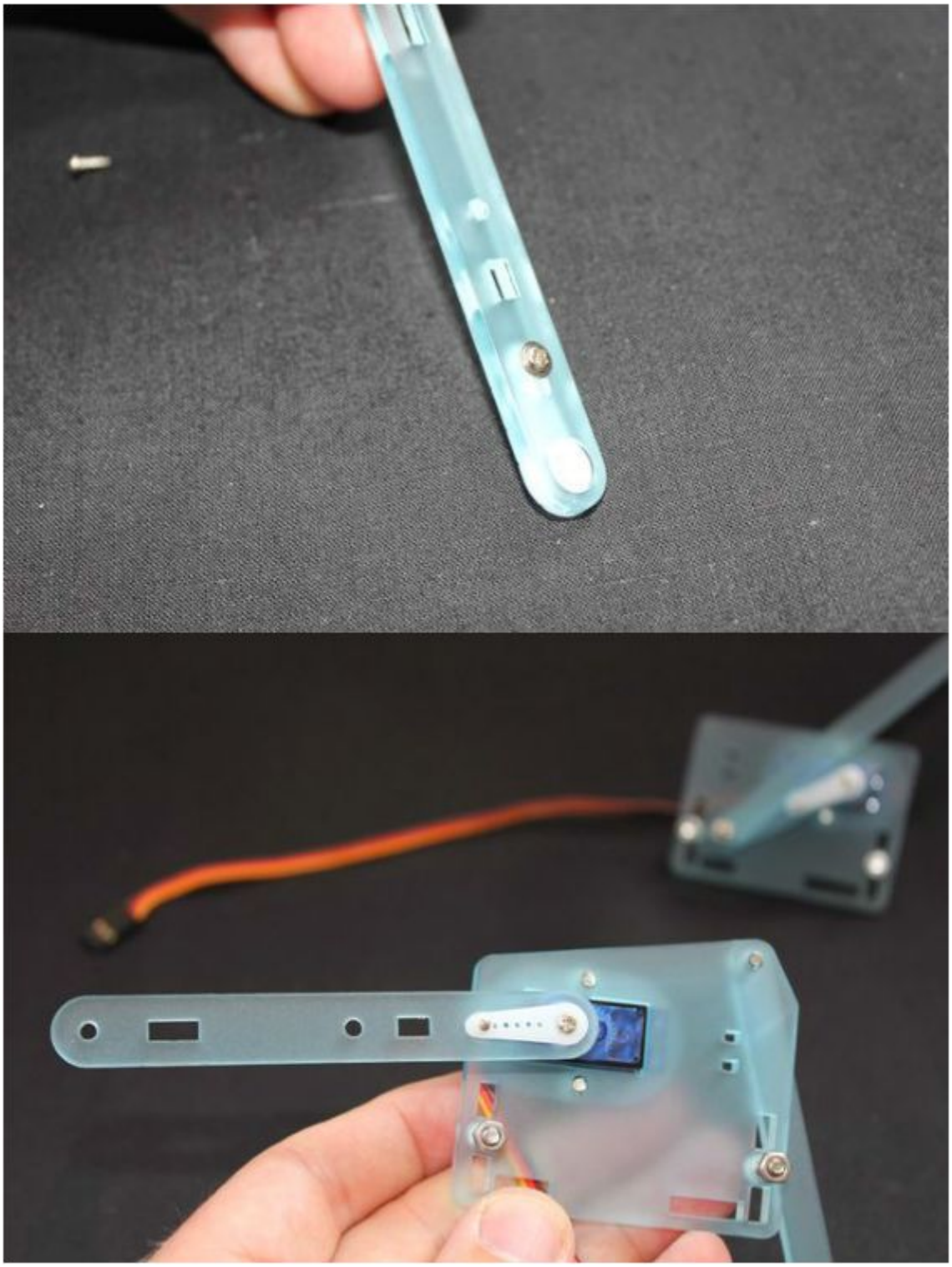


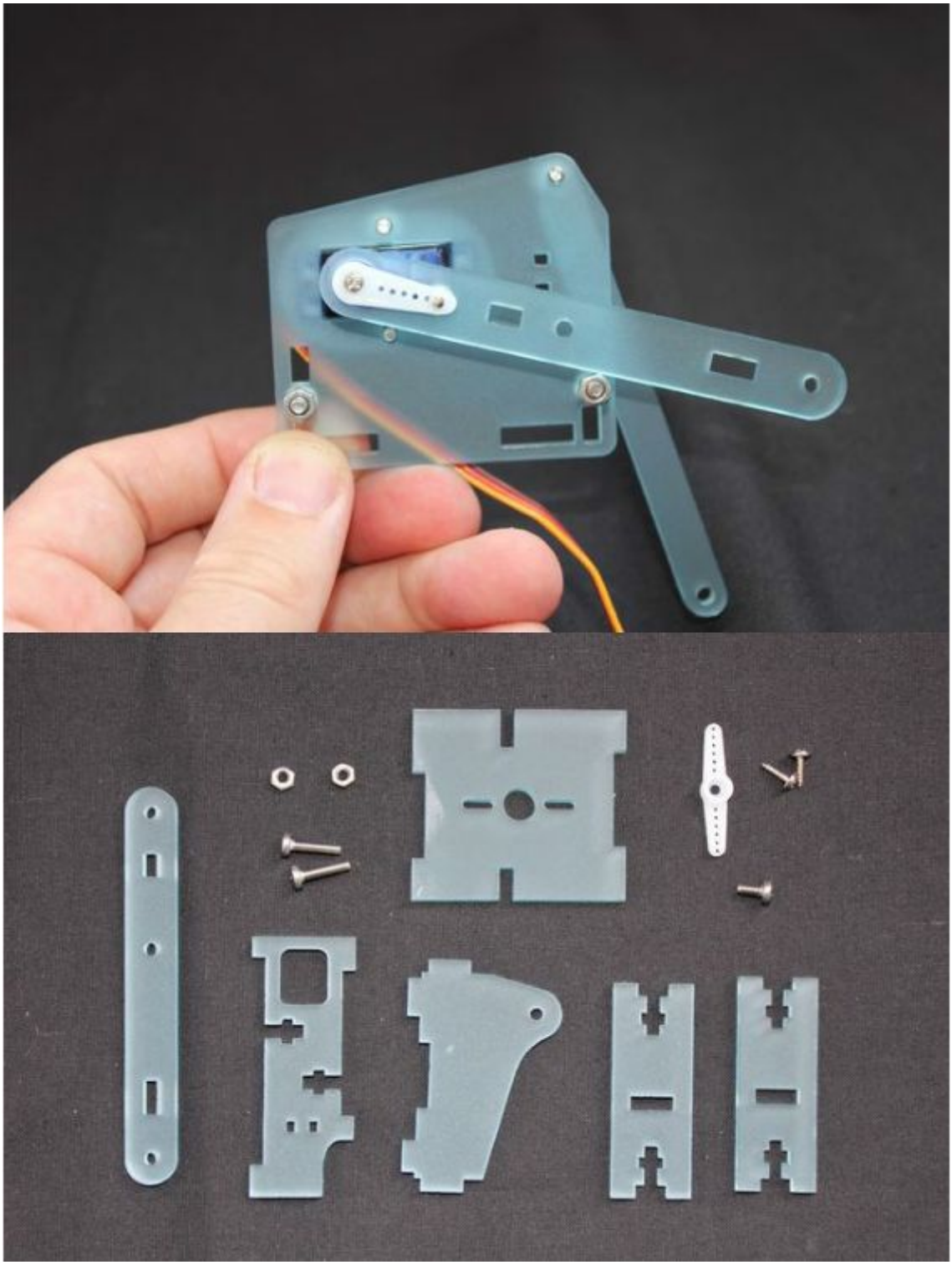
ATENÇÃO: Antes de parafusar o braço no servo, verifique se o servo está no limite quando o braço é colocado na posição mostrada na imagem superior.



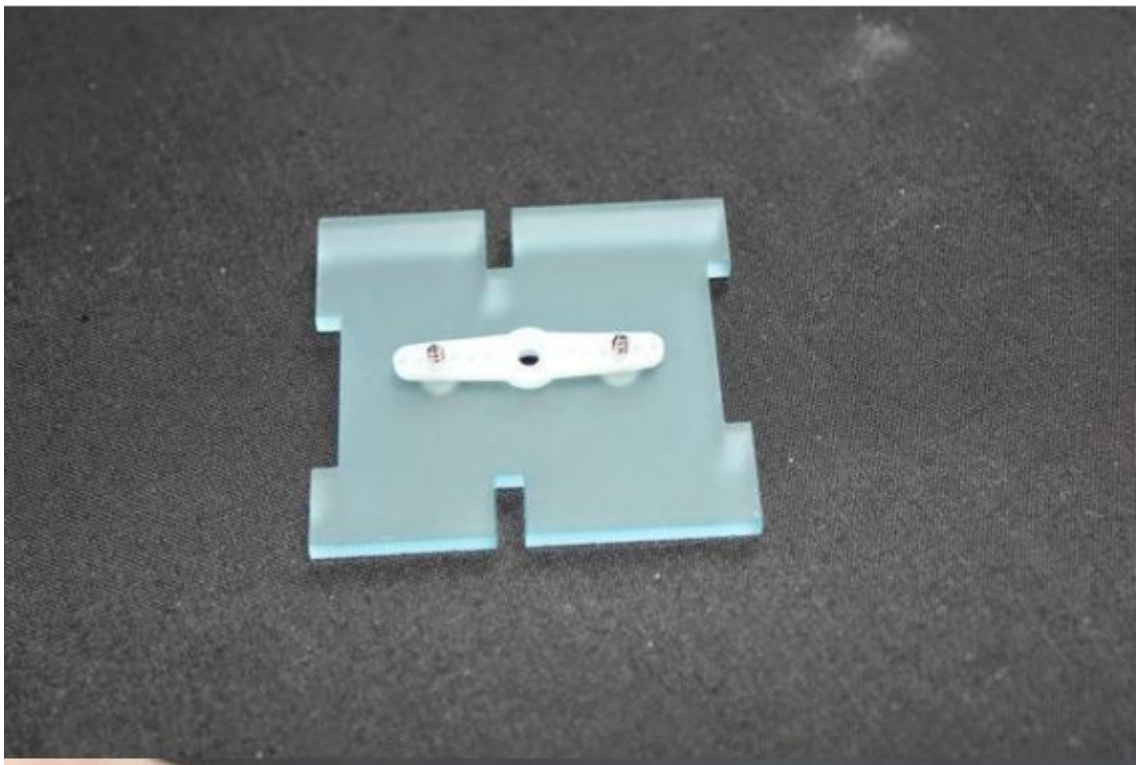


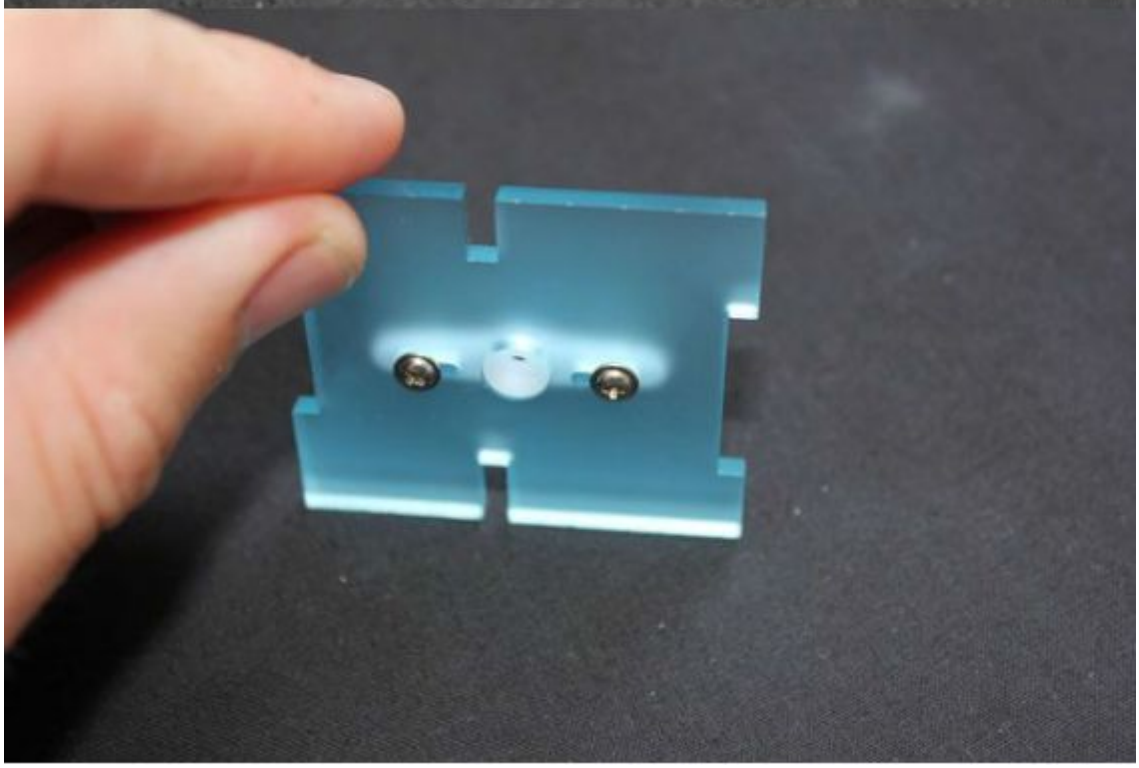


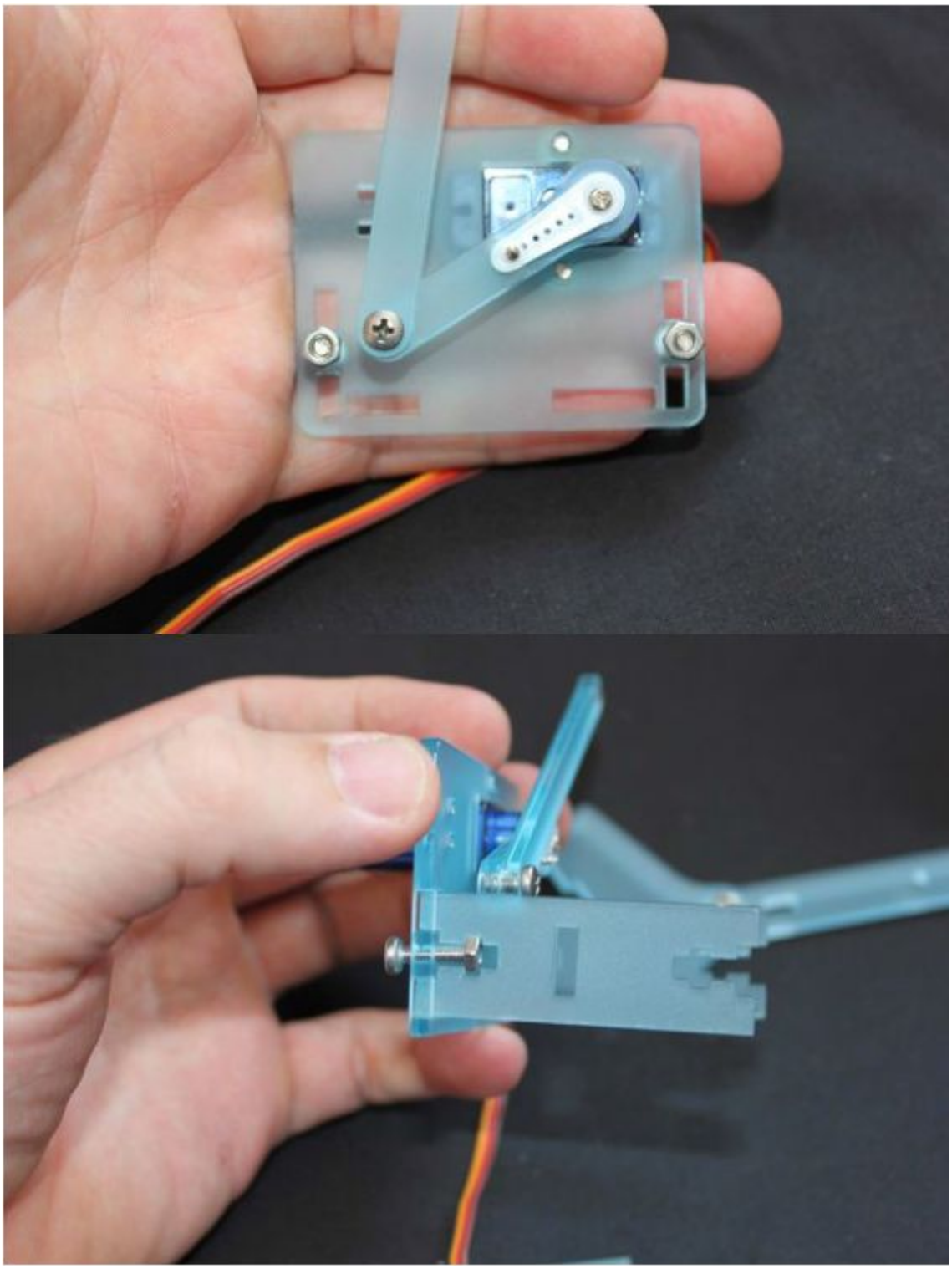


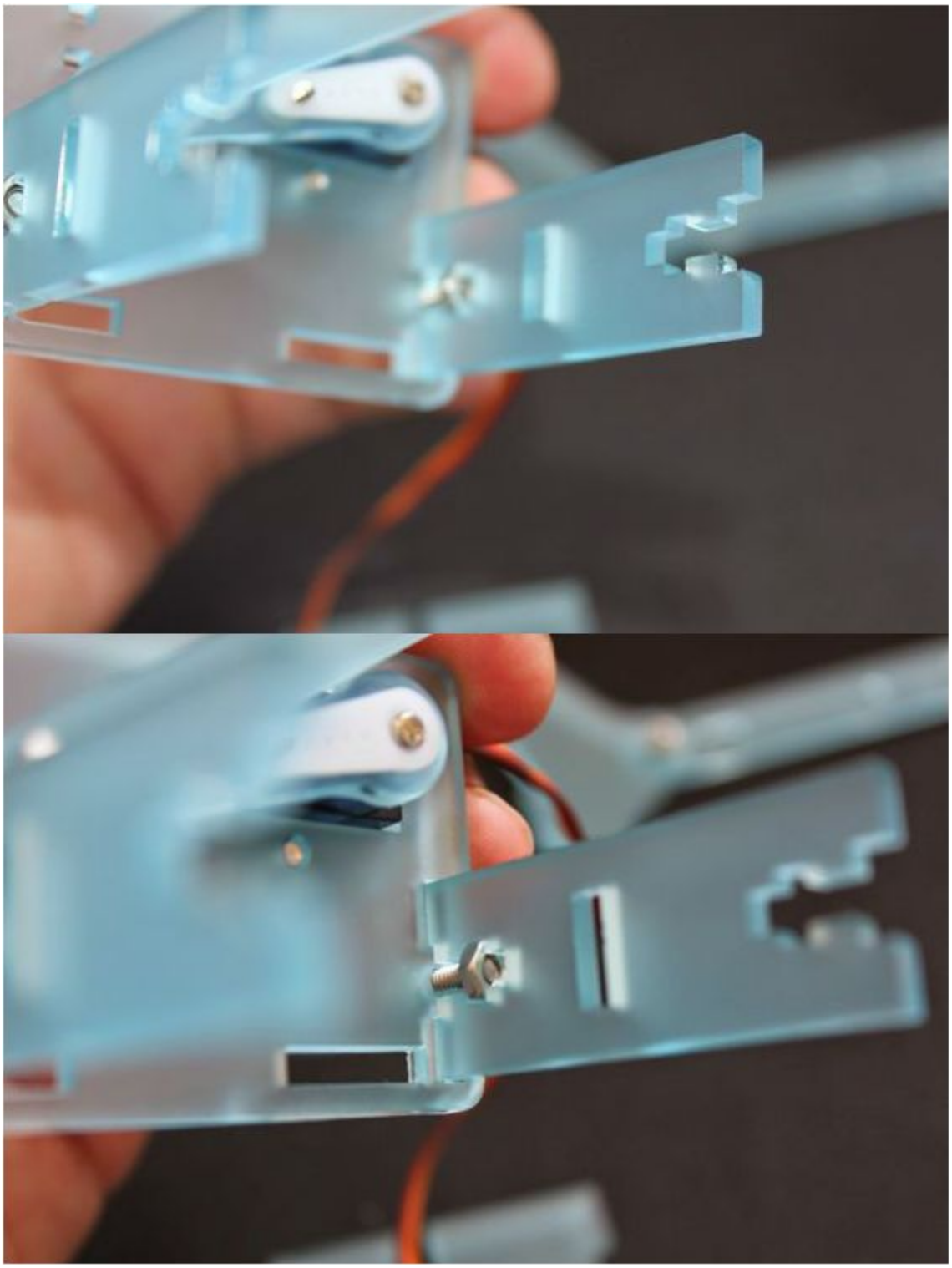


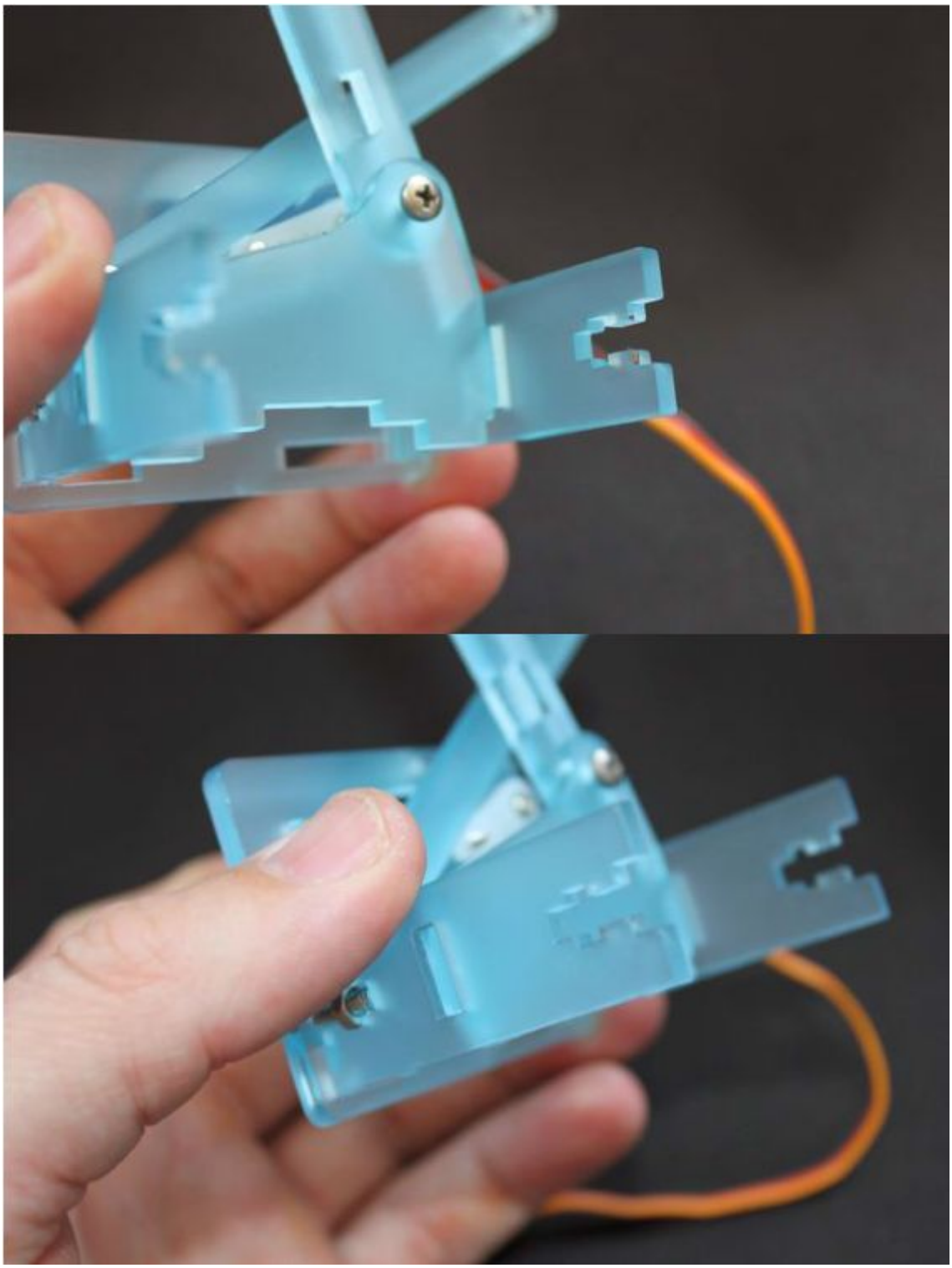
ATENÇÃO: Antes de parafusar o braço no servo verifique se o servo está no seu limite quando o braço estiver na posição da imagem superior.

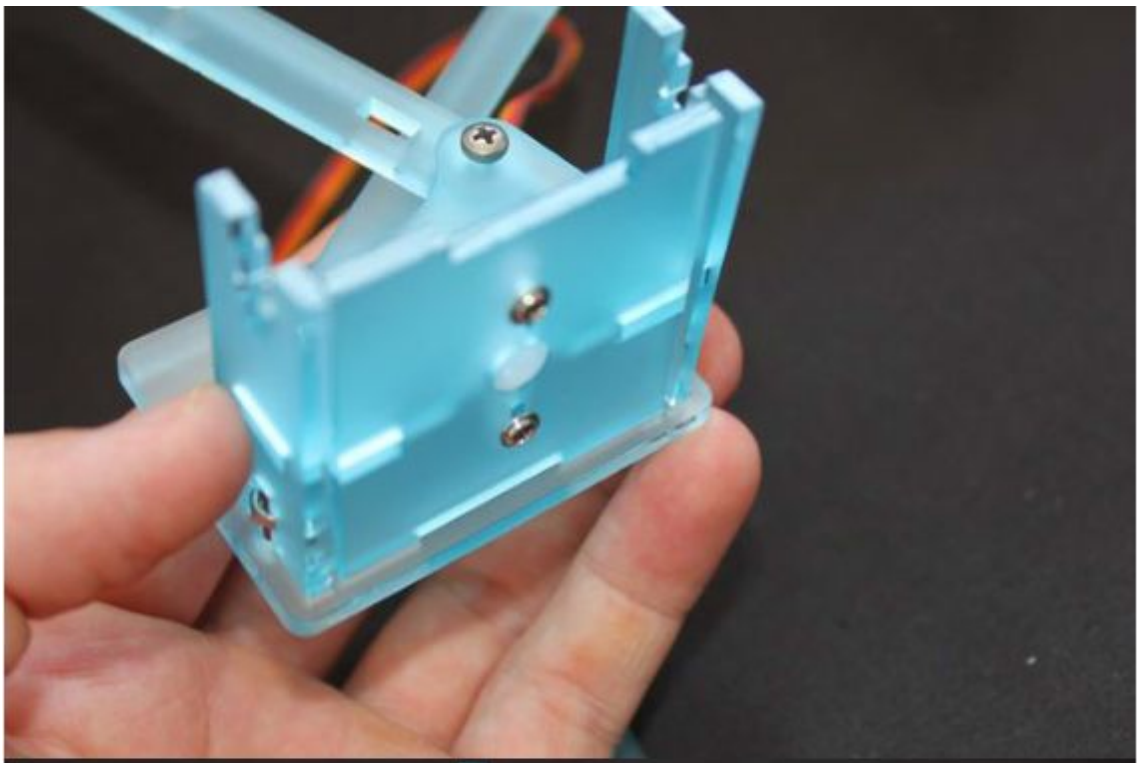


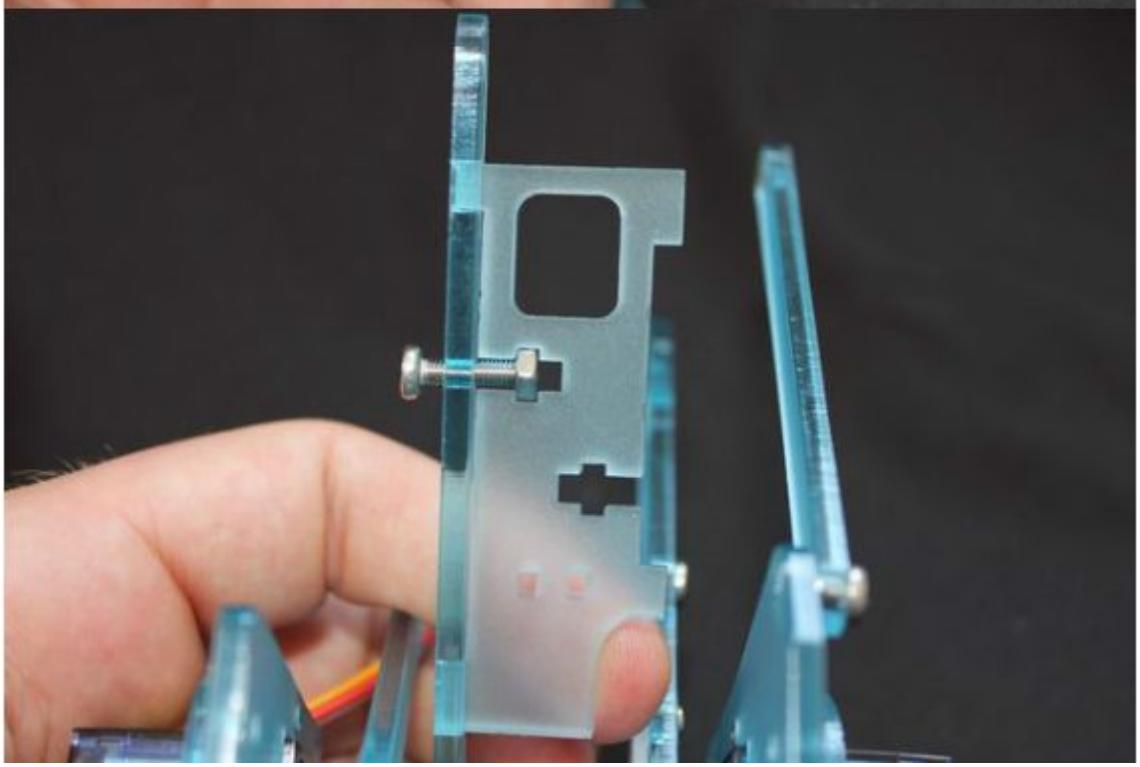


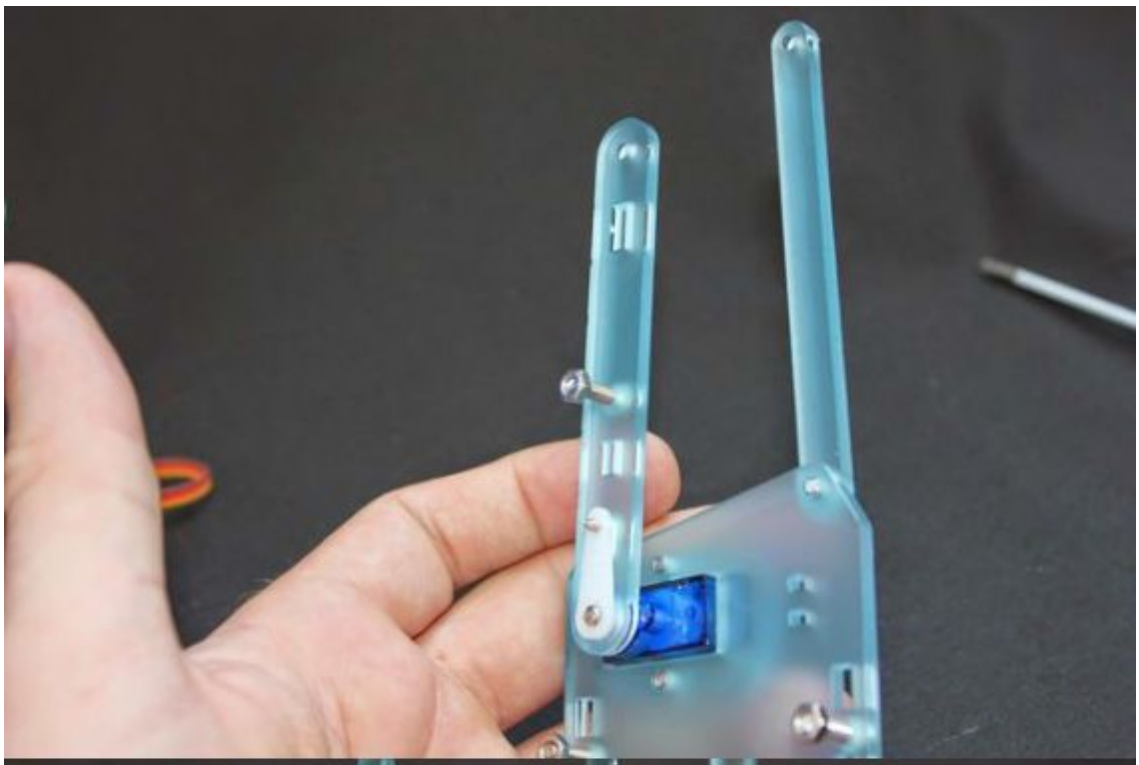


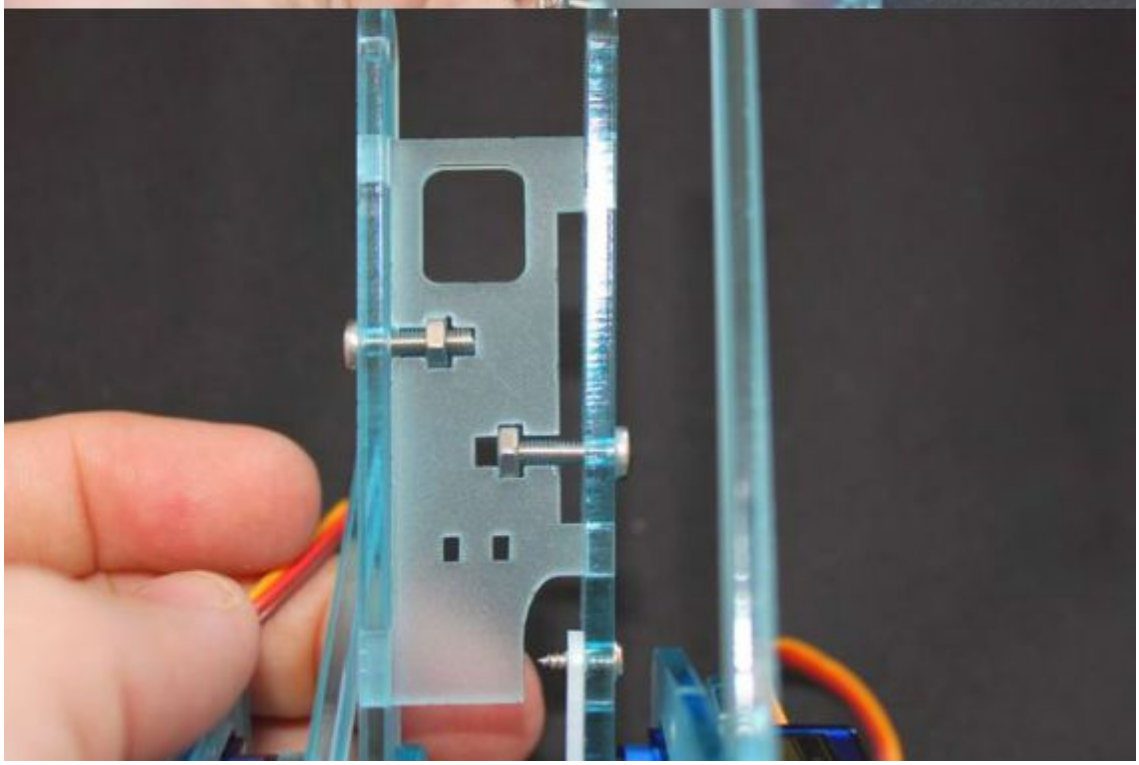


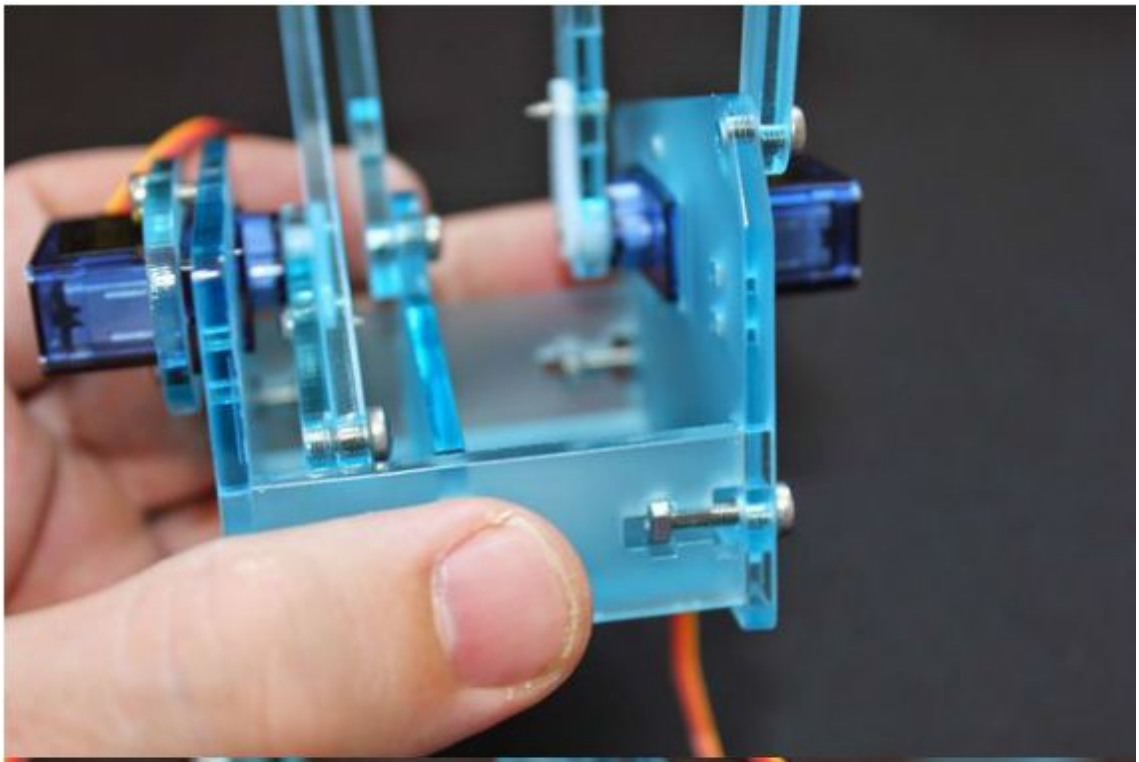


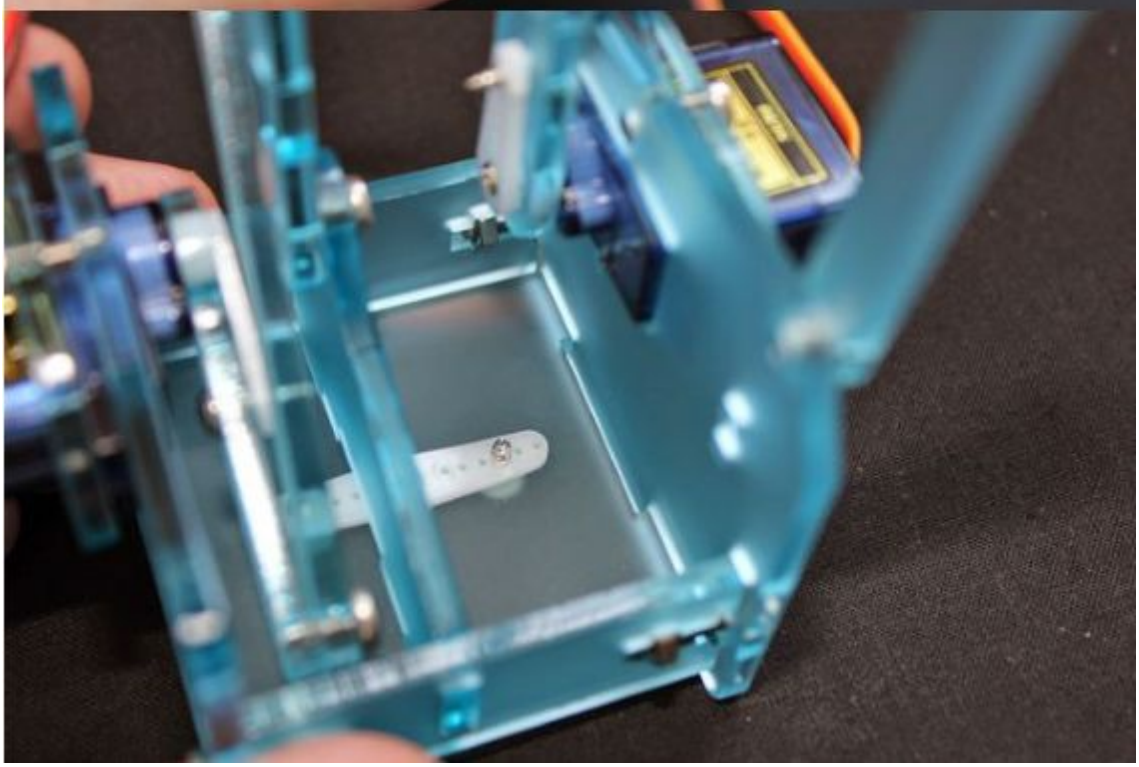


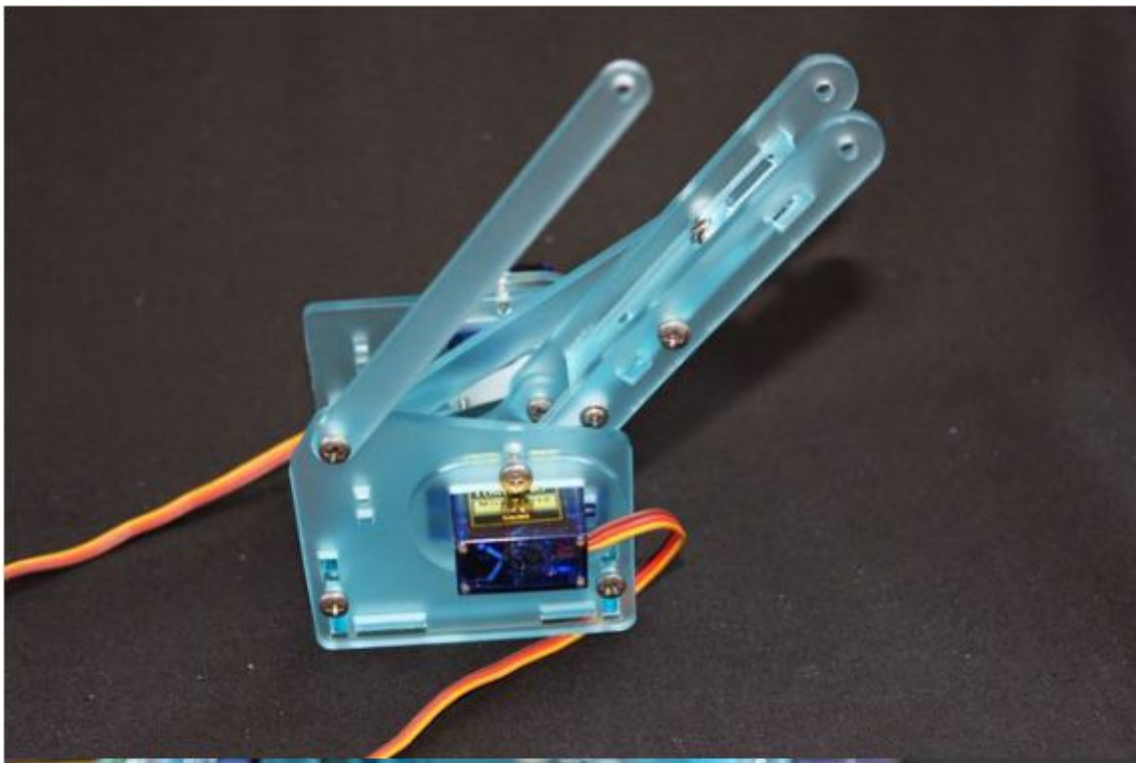


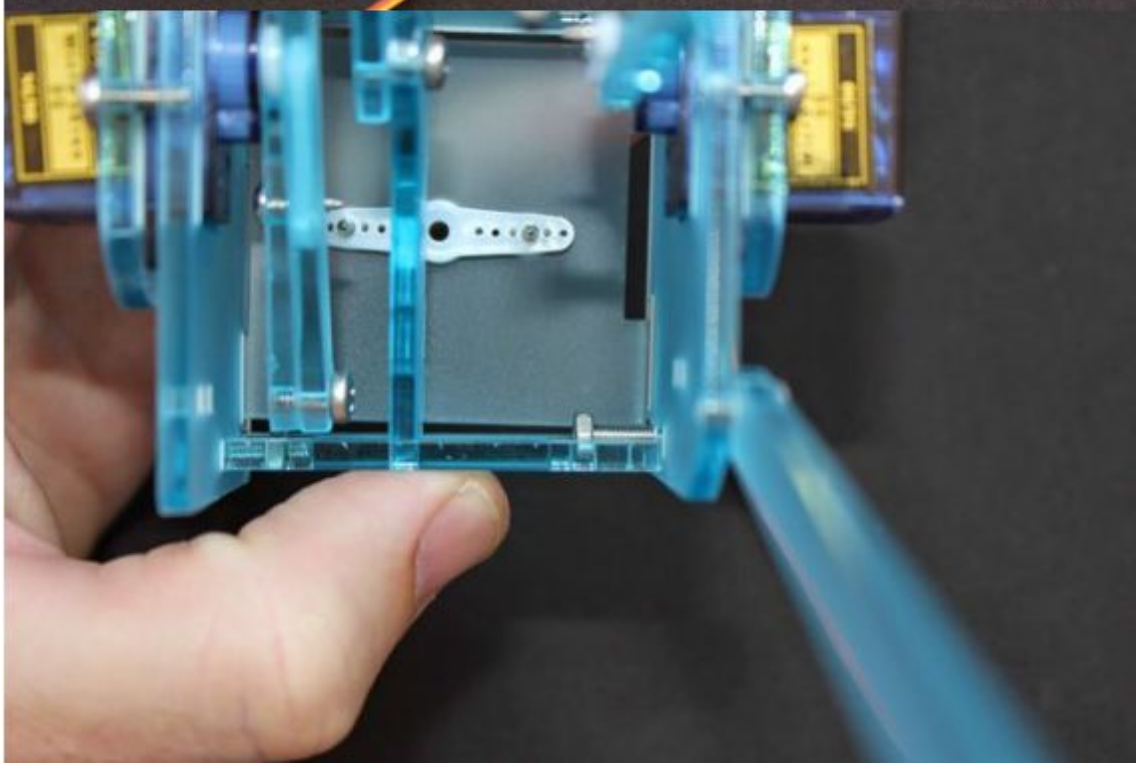


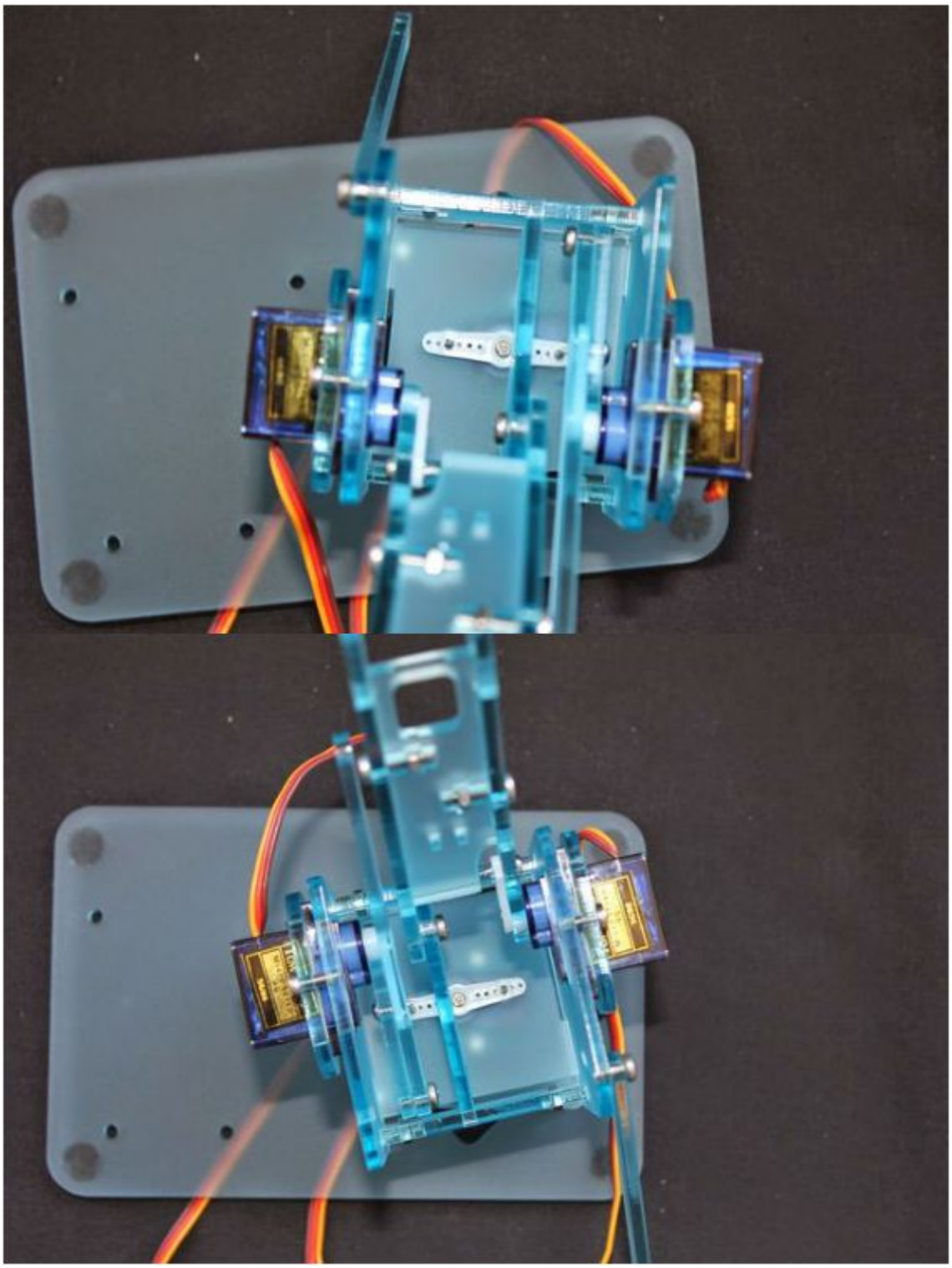






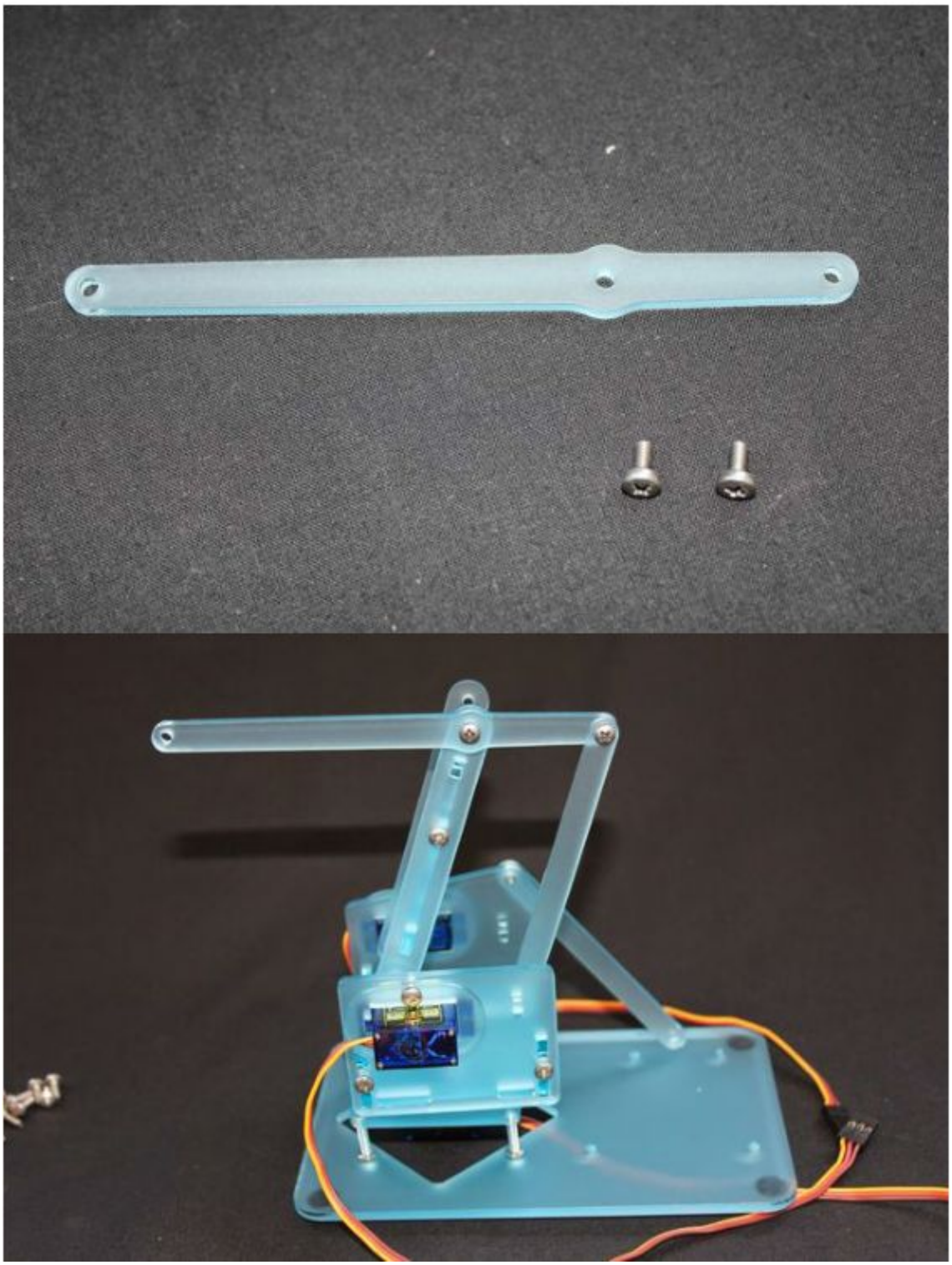


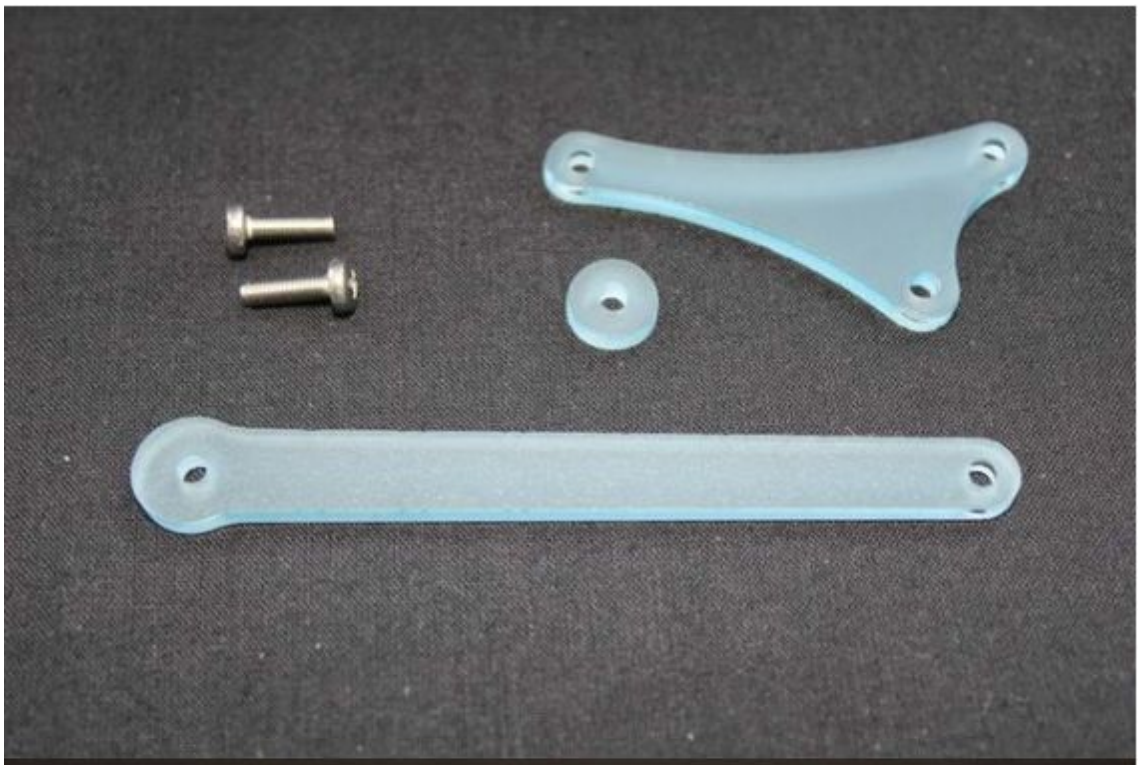


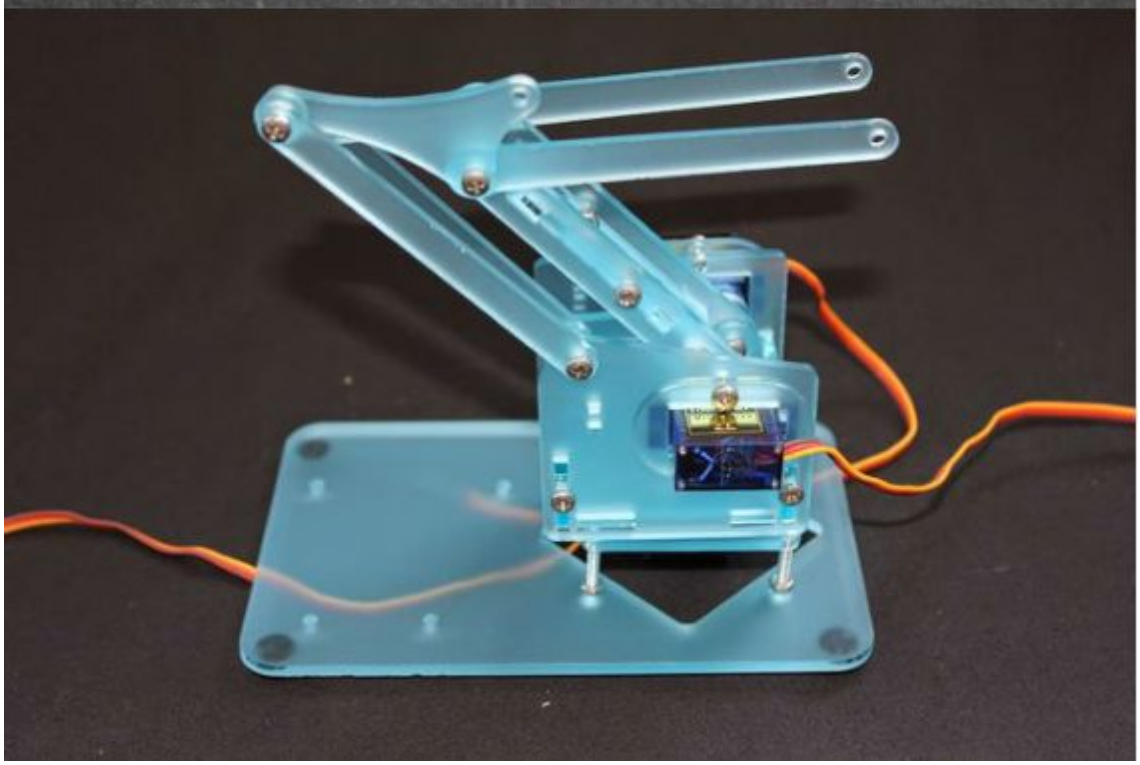


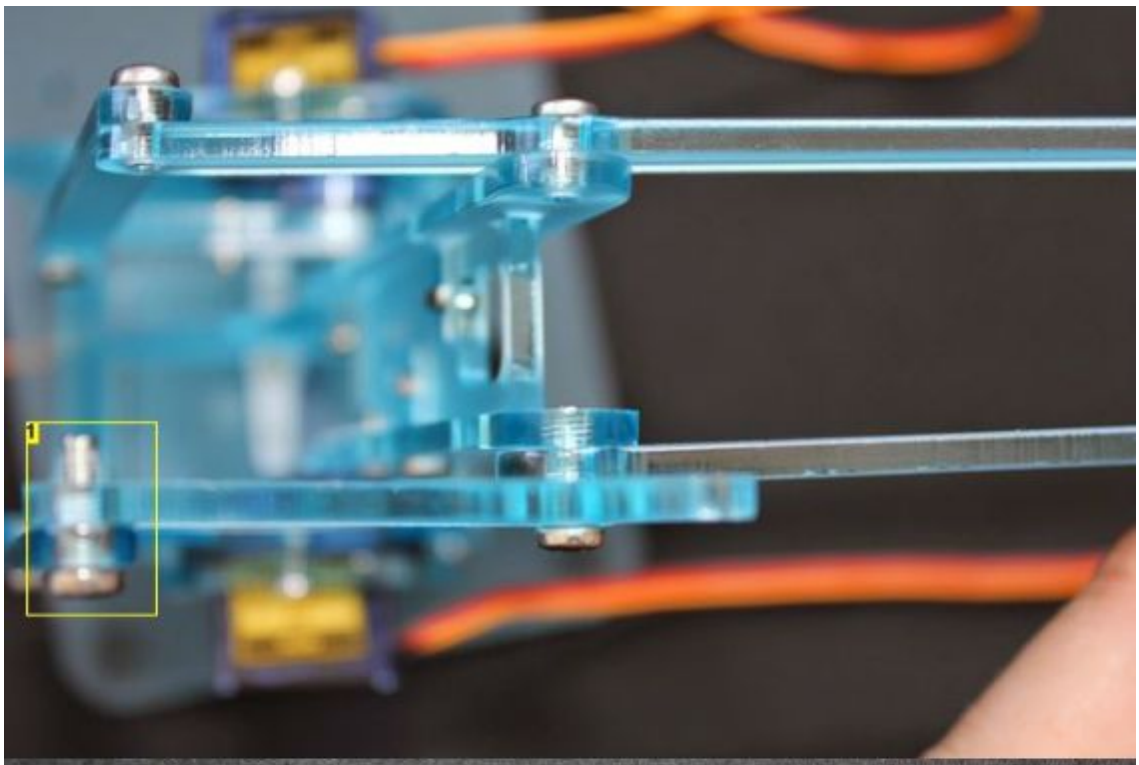
ATENÇÃO: Antes de parafusar a base no servo, posicione a base de forma que ela consiga rotacionar totalmente para a direita e para a esquerda, correspondente ao limite dos servos. Quando a base estiver reta, ela deve estar na posição central do servo.

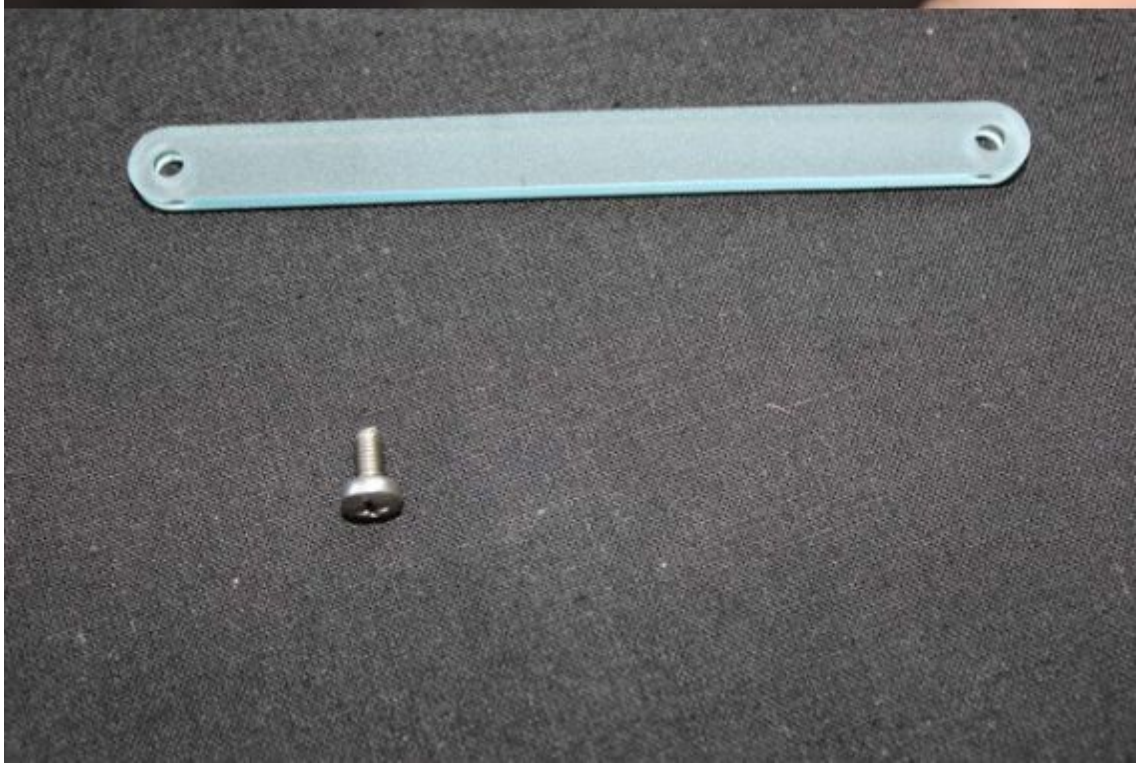
Montando o Braço

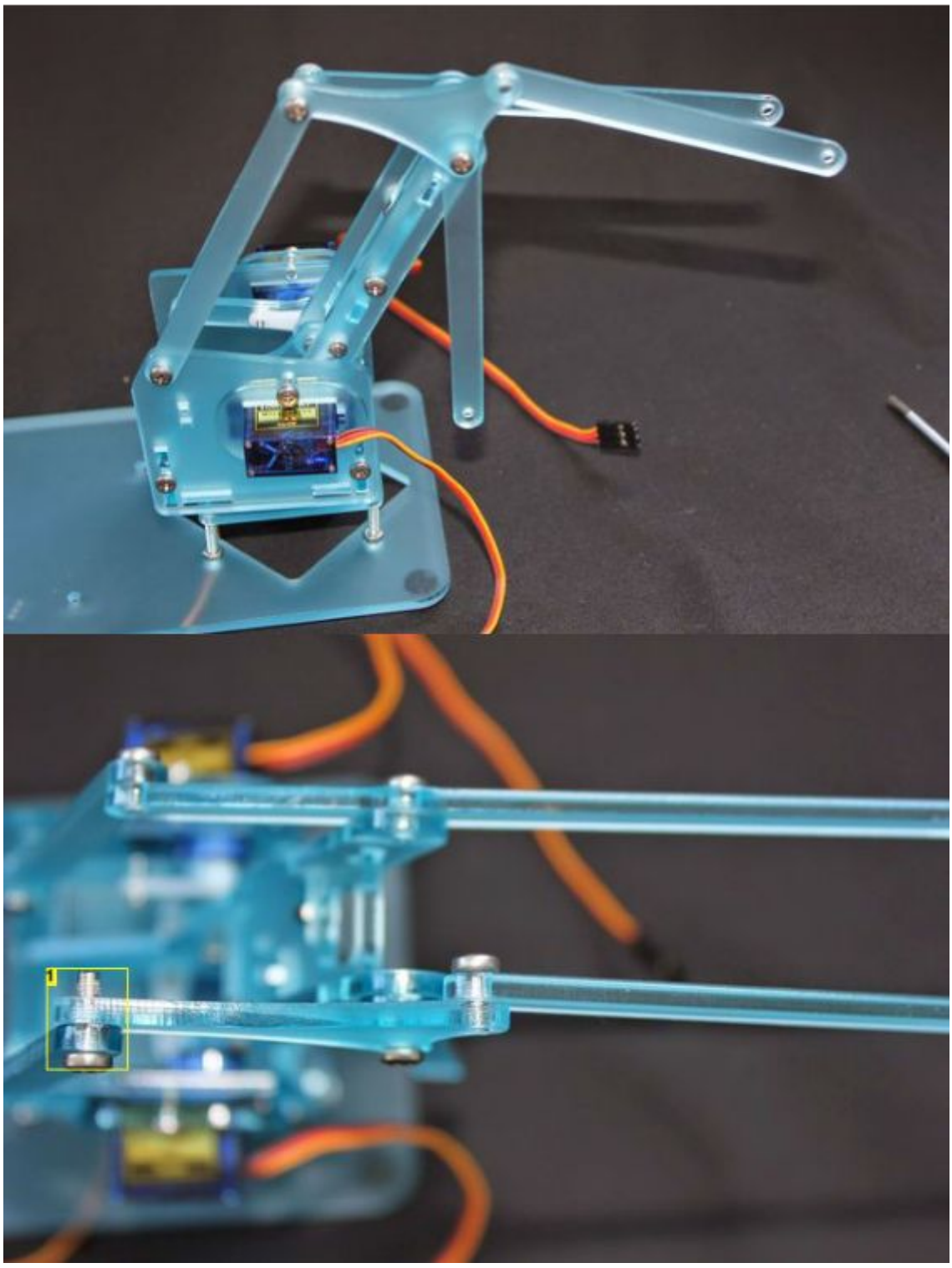




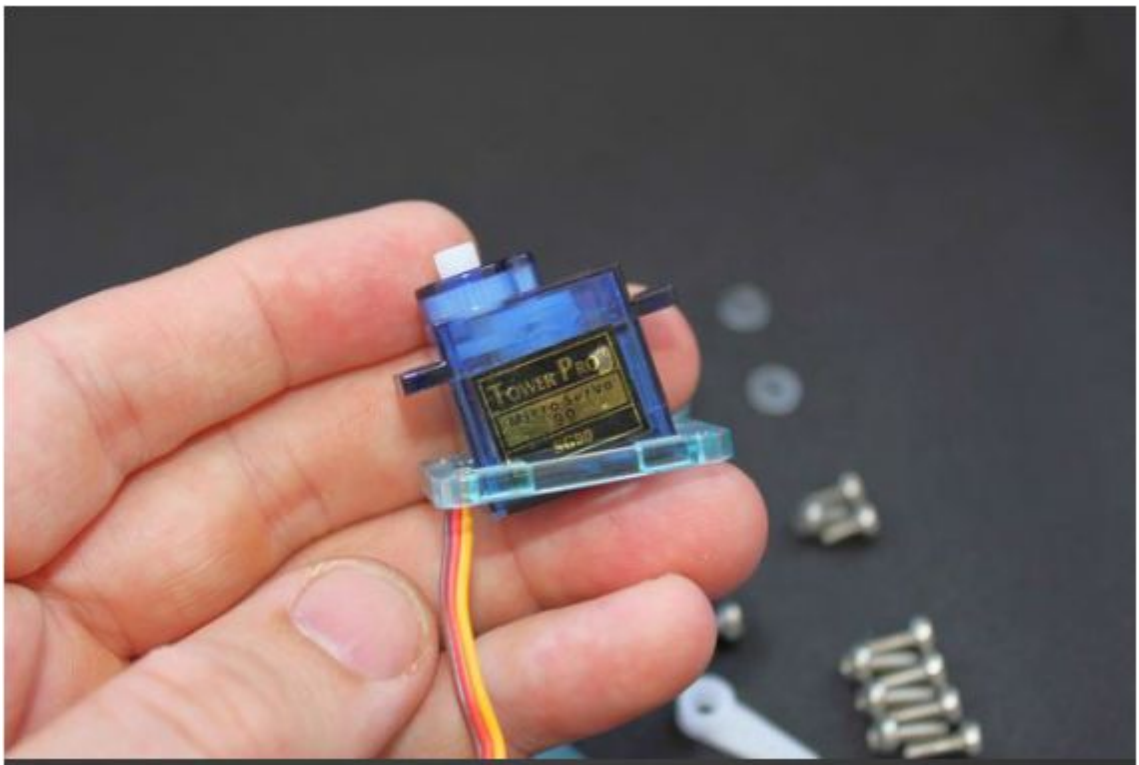


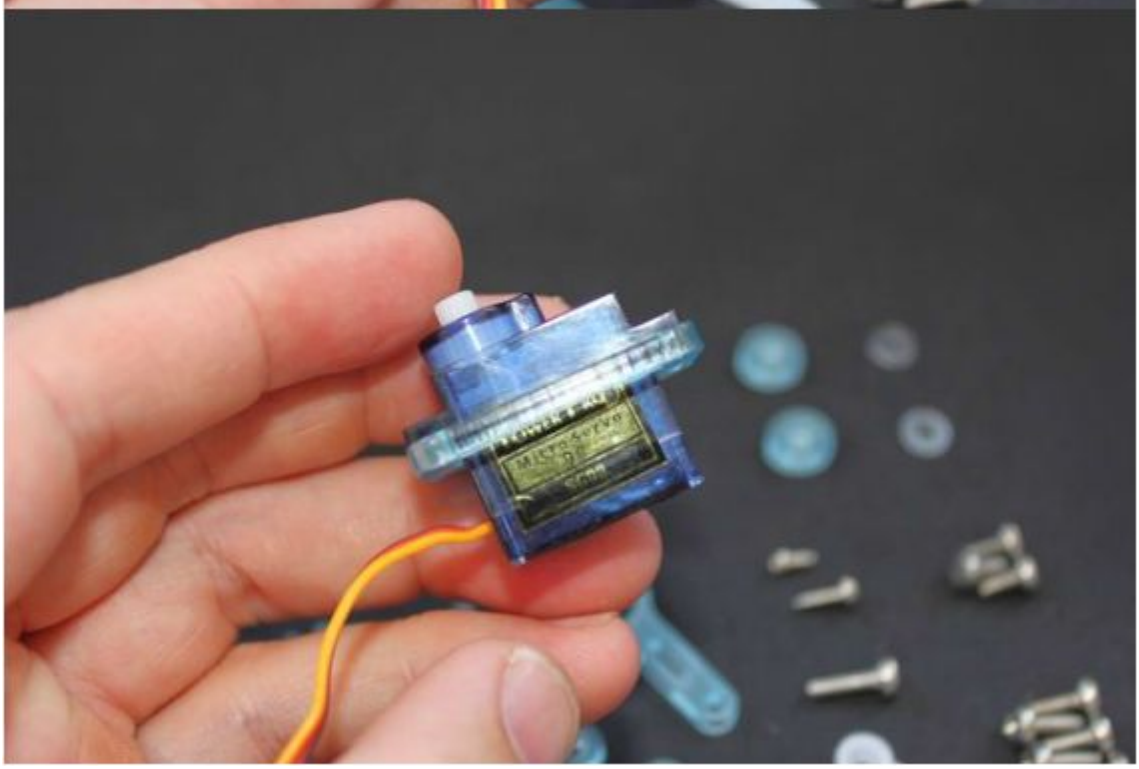


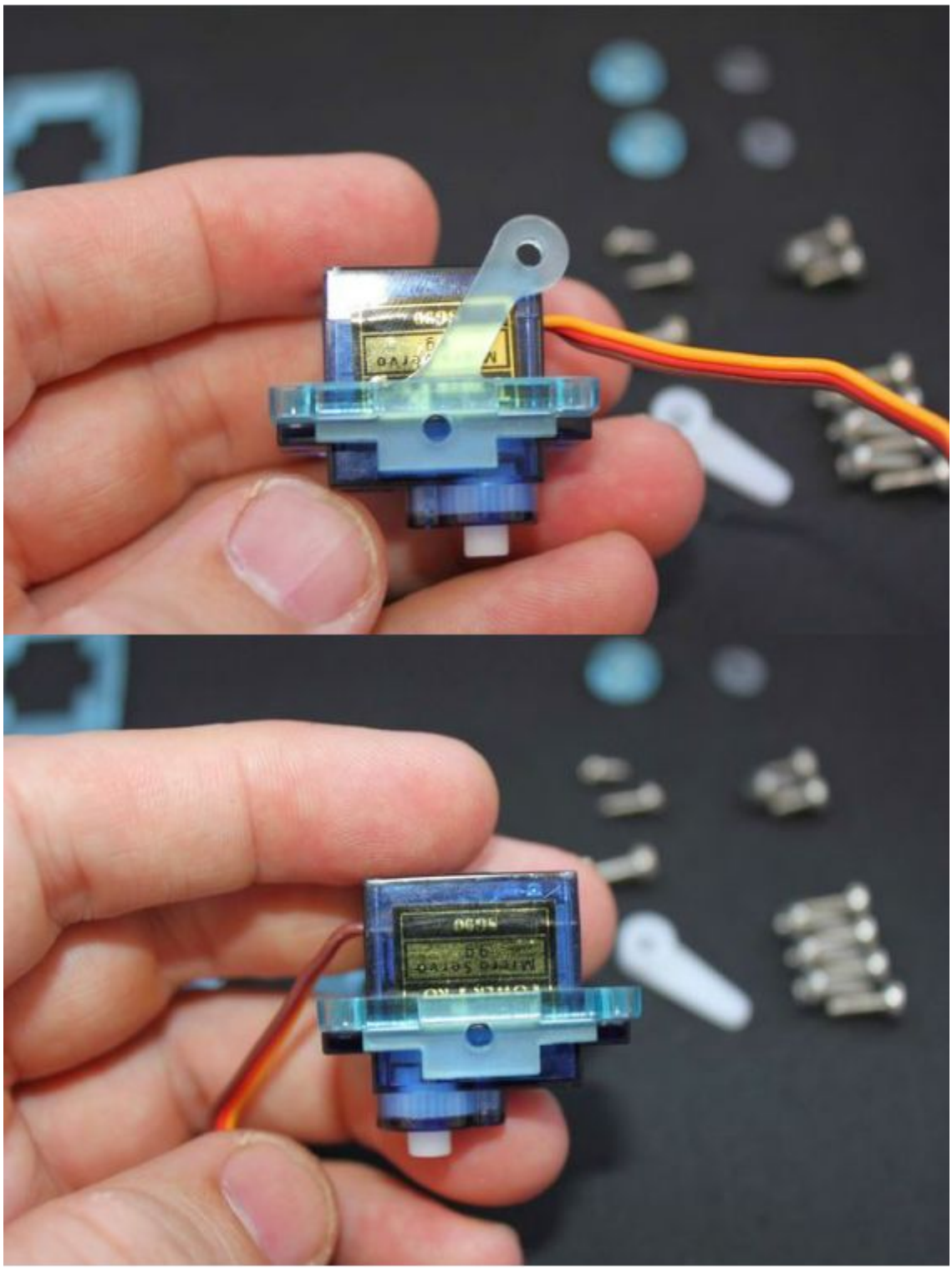


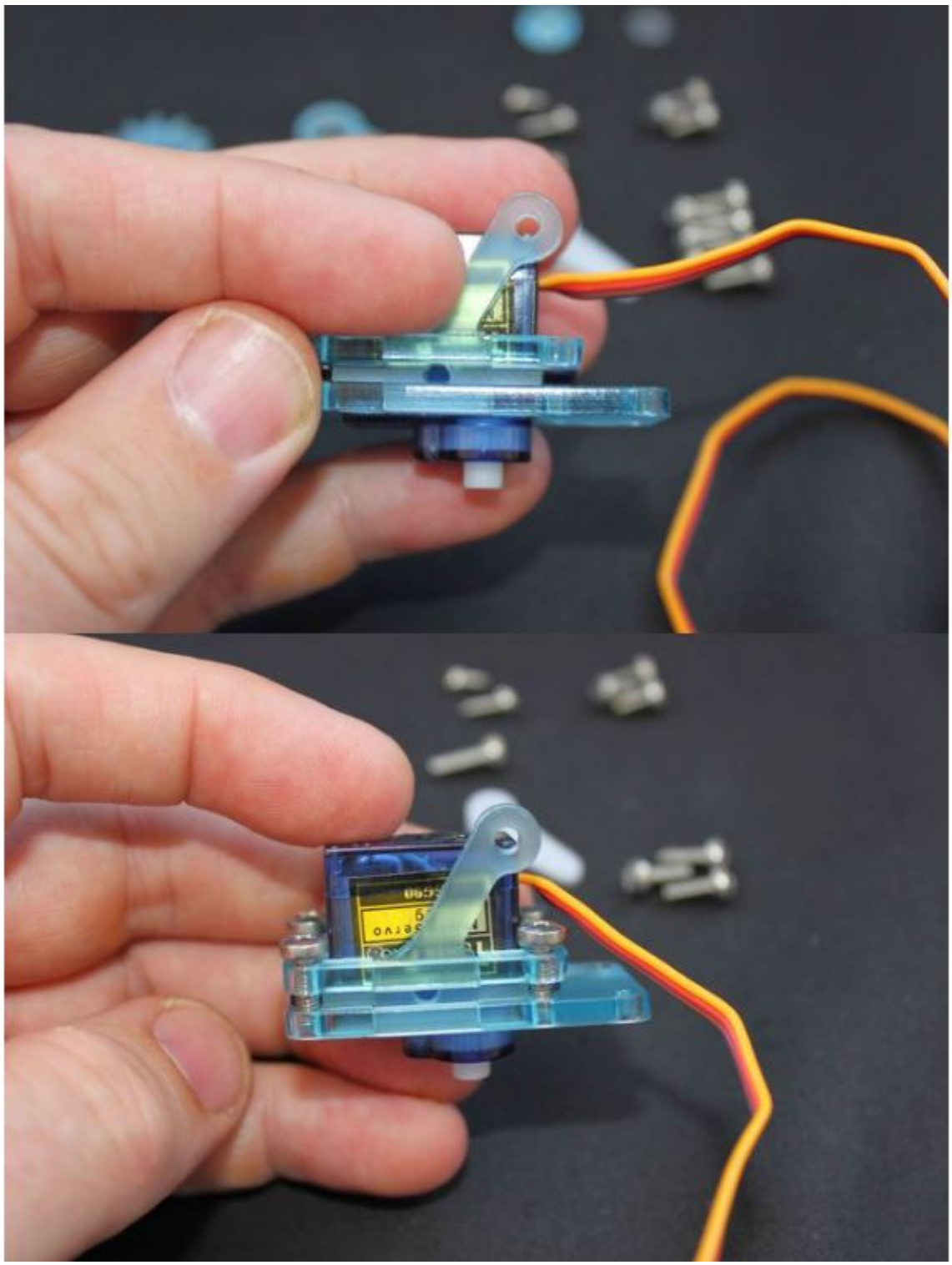

1

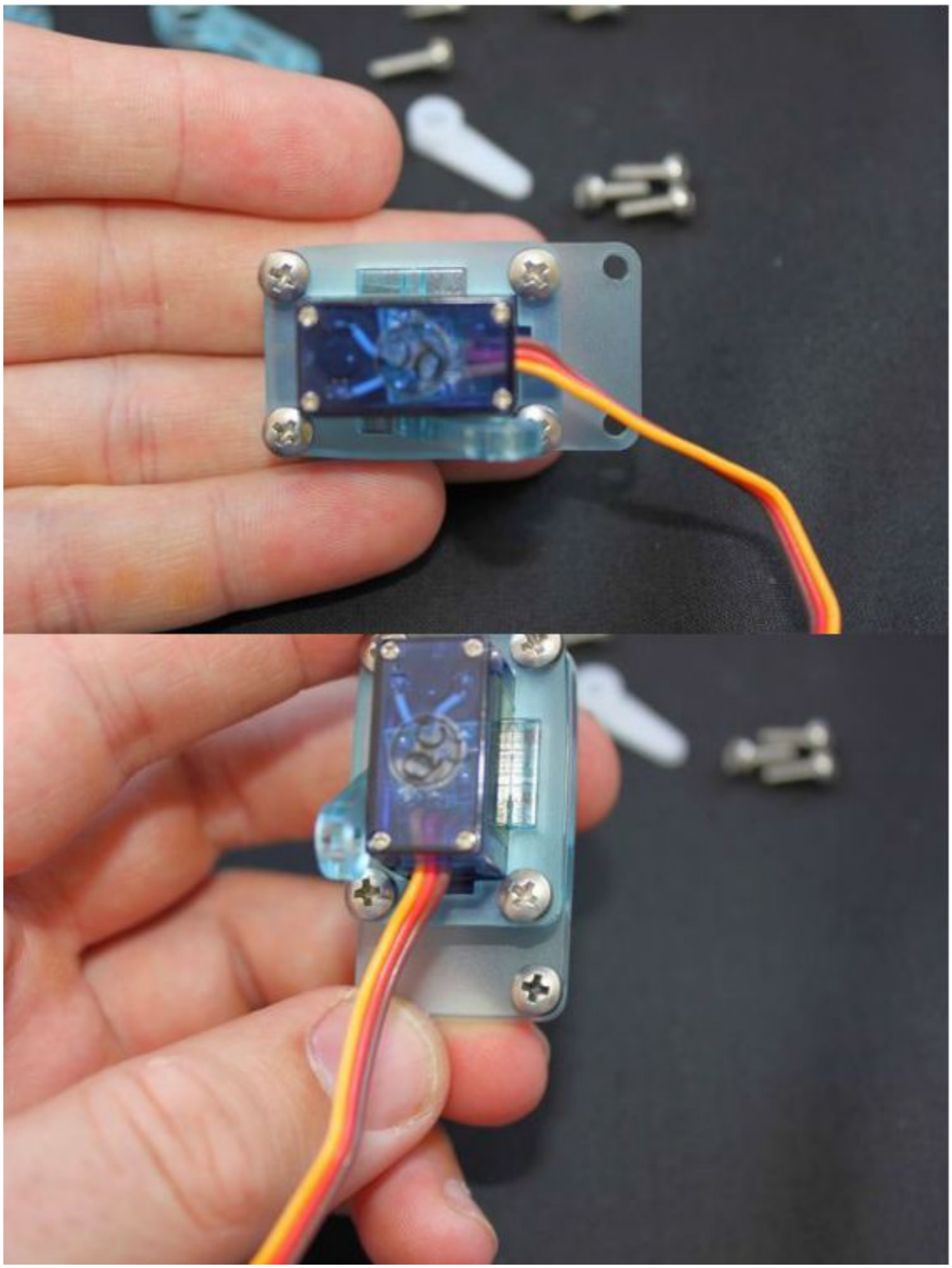
Montando a Garra

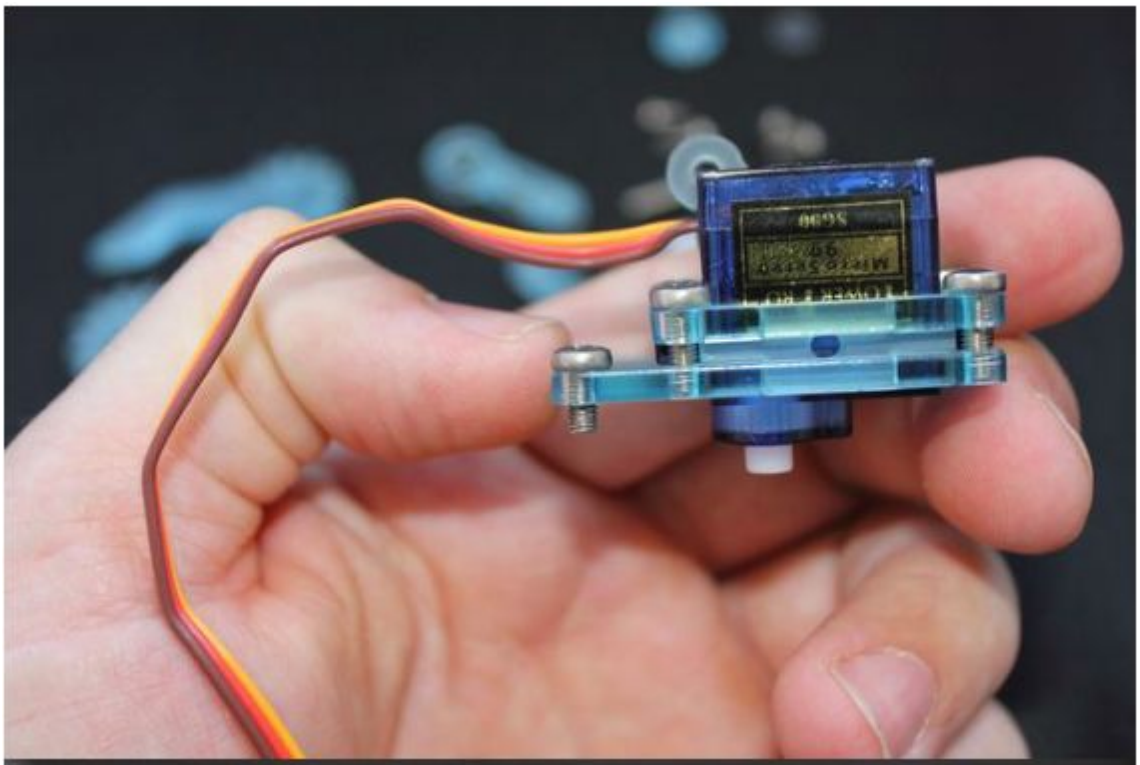

TOWER PRO
MICRO SERVO
9G

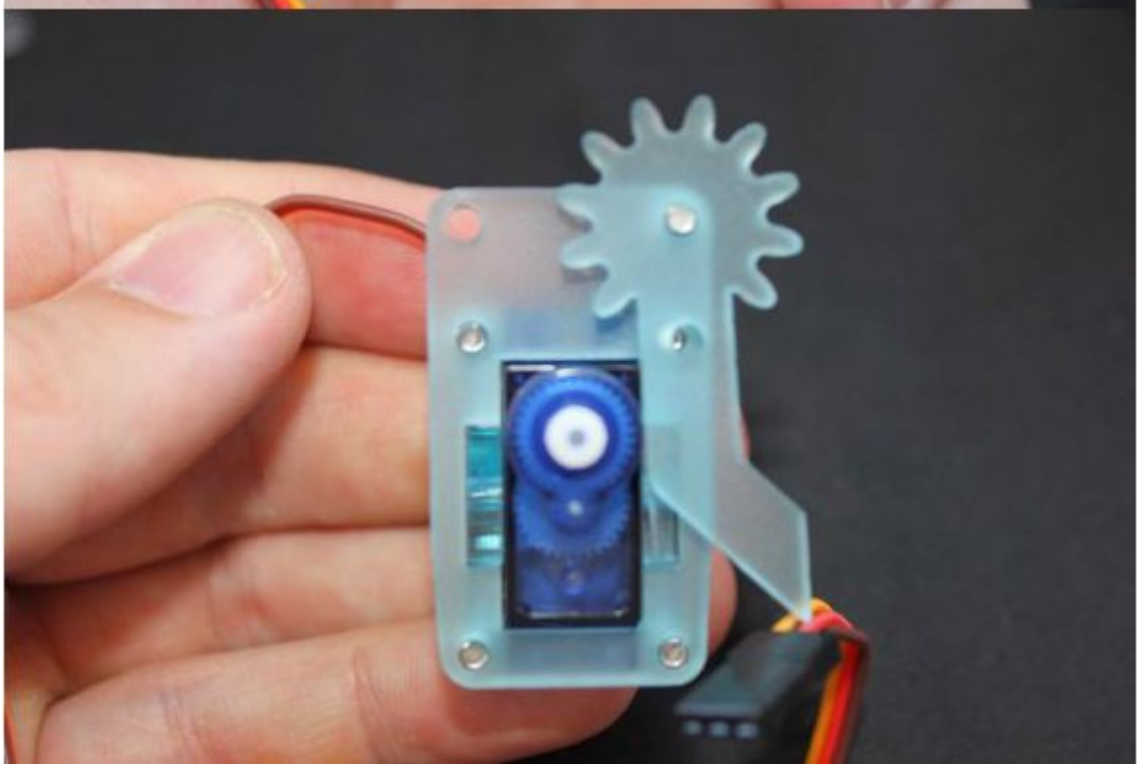


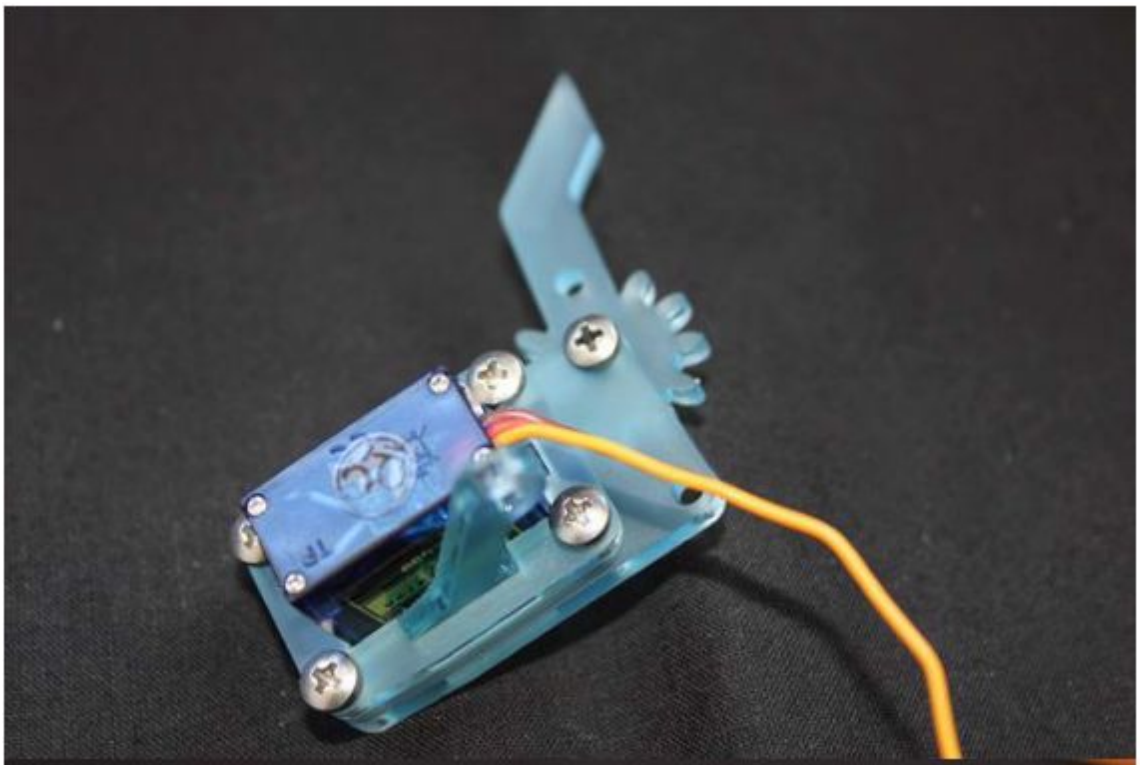


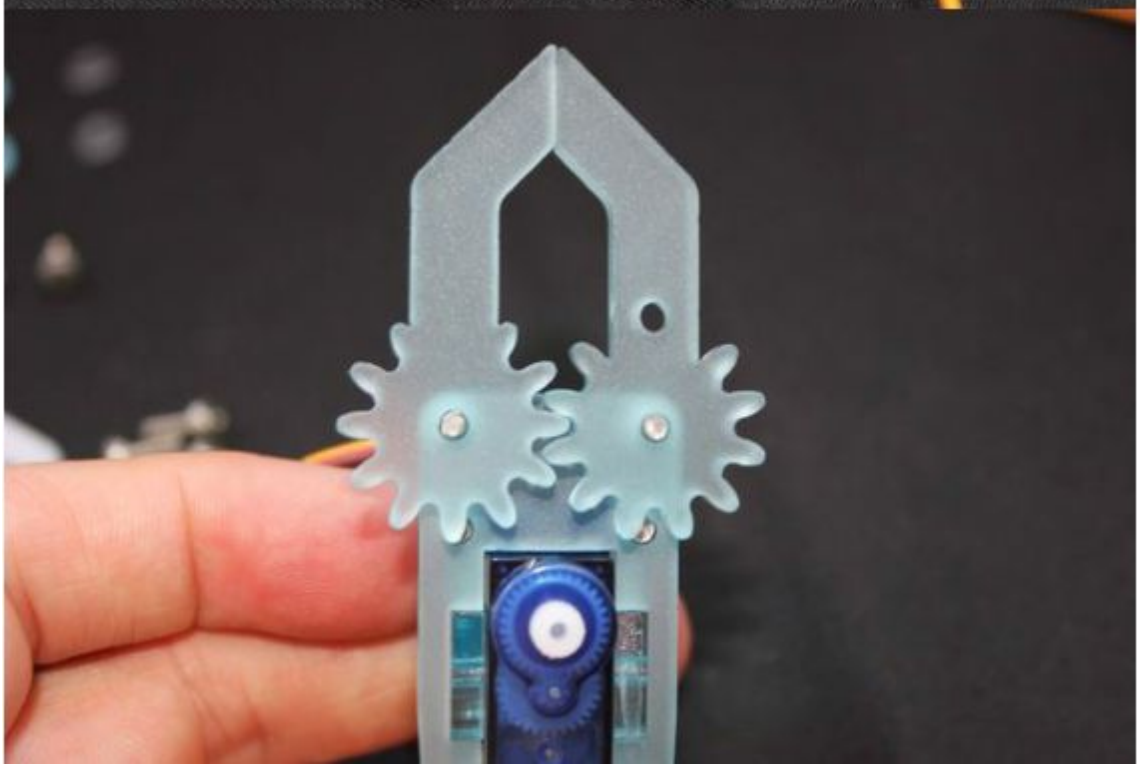


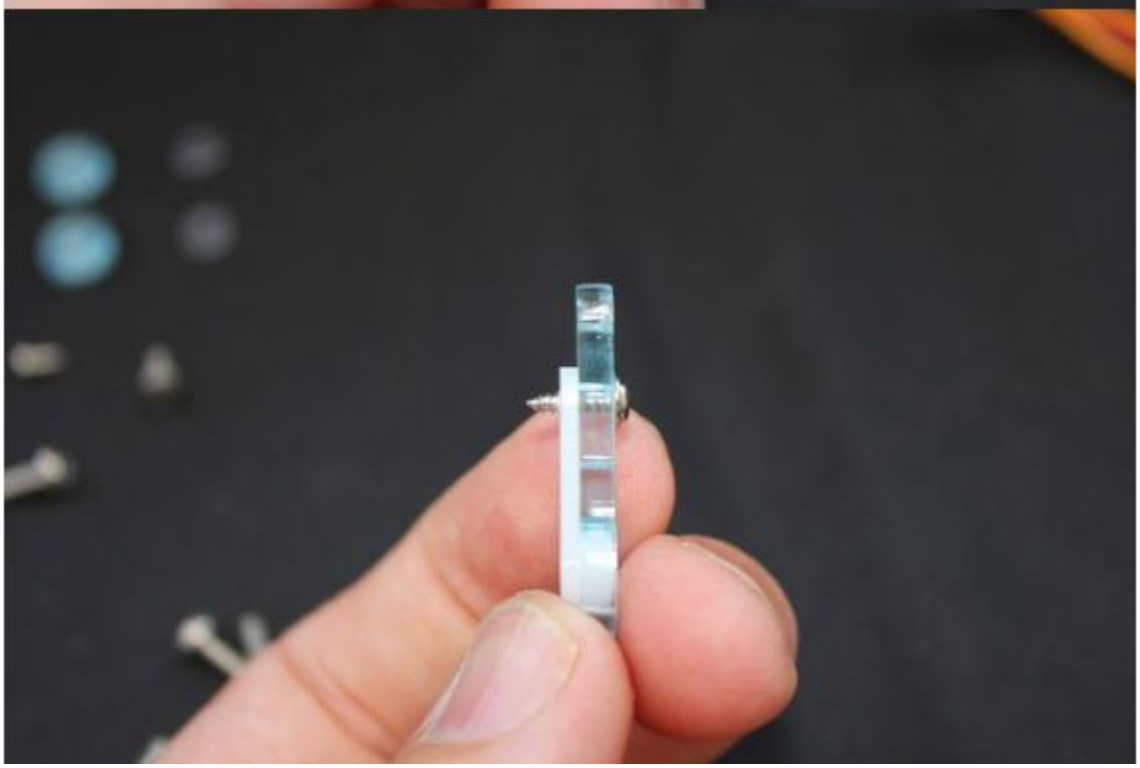


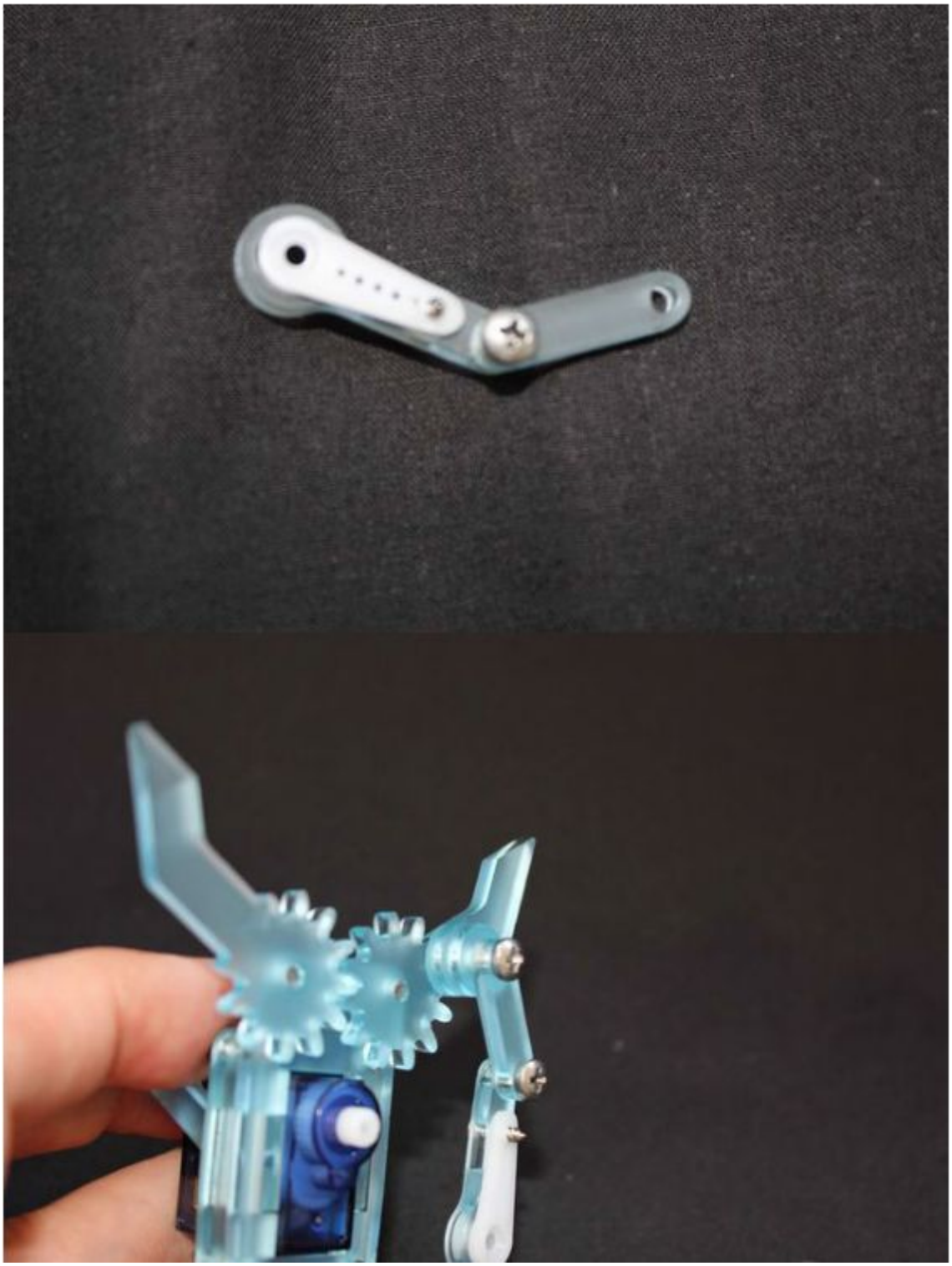




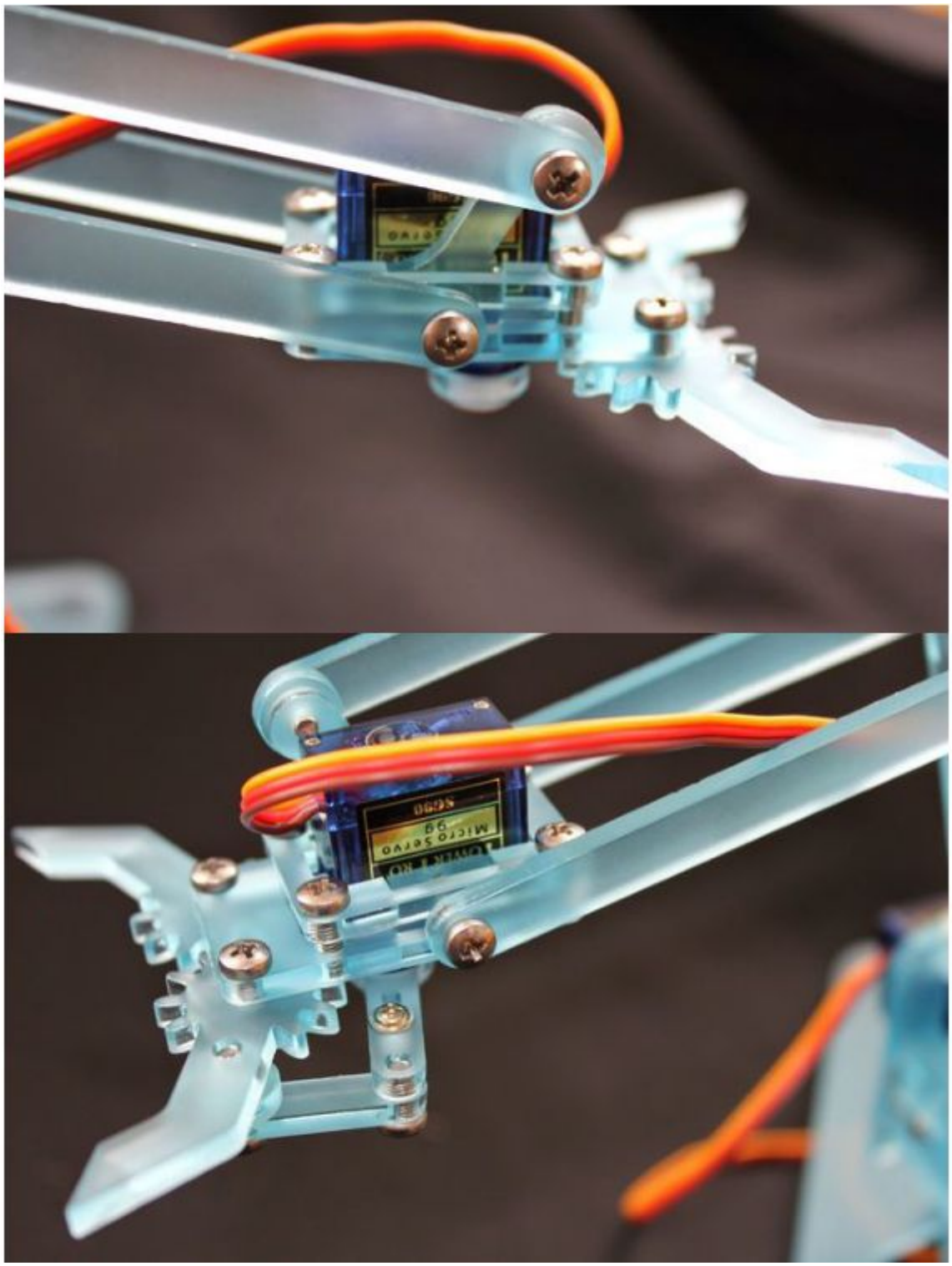


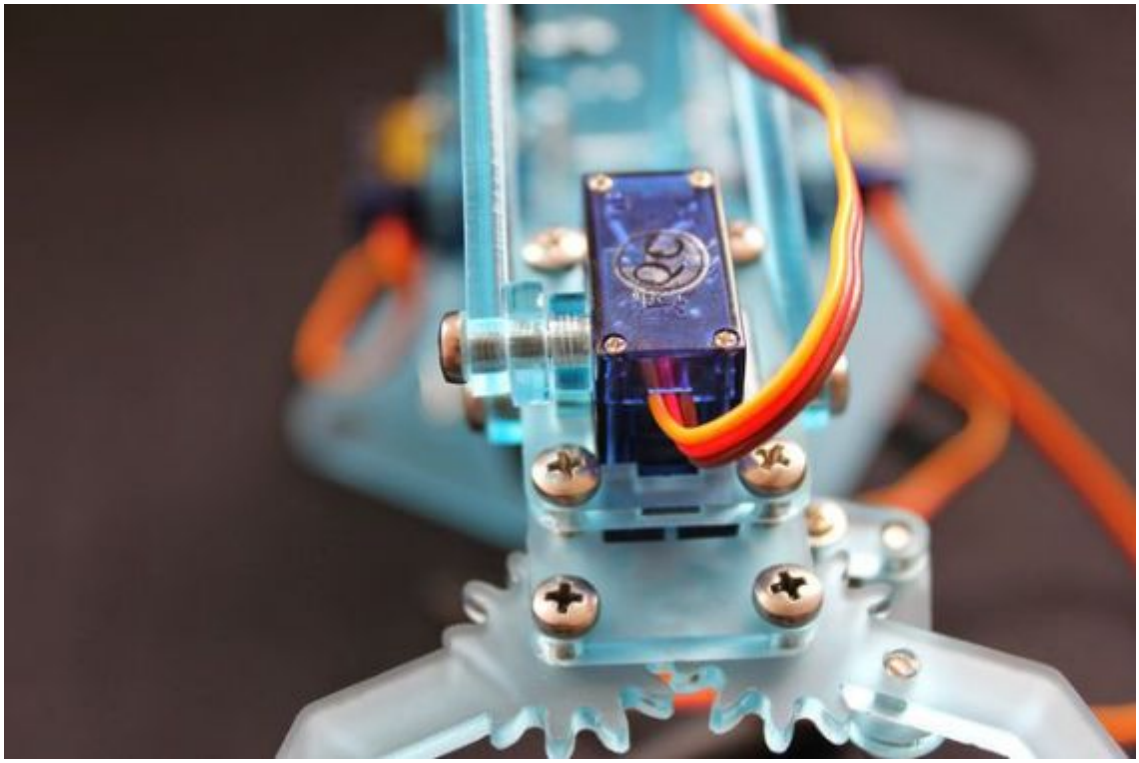






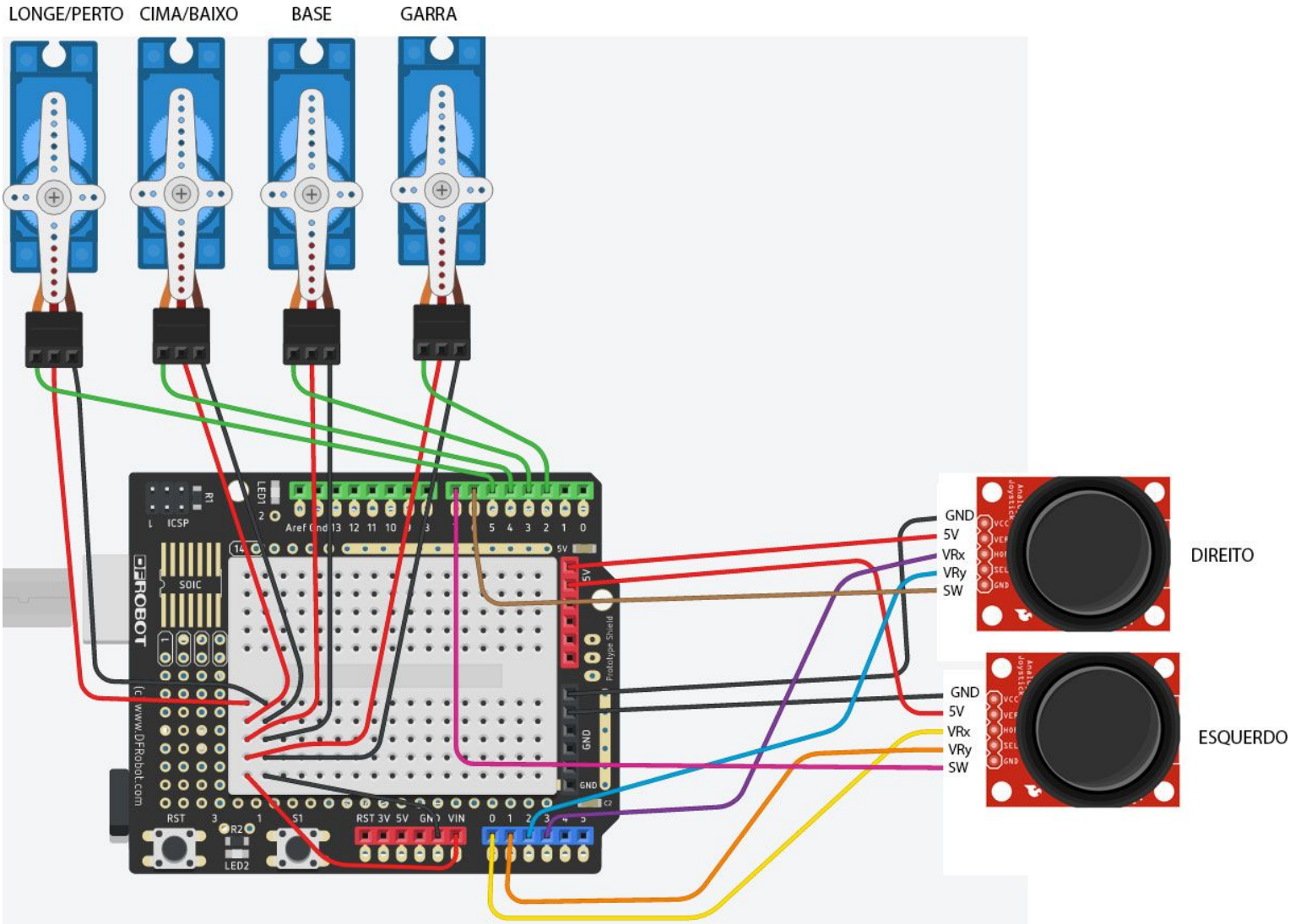
ATENÇÃO: Antes de parafusar o braço no servo verifique se o limite do servo corresponde aos movimentos de abertura e fechamento da garra.


MICRO SERVO
9G
SG90



Eletrônica

Utilizaremos os seguintes componentes:
1x Arduino Uno
1x Protoboard Shield
2x Joysticks
4x Servo motores



Código

Os códigos necessários para o funcionamento do robô podem ser obtidos em: https://github.com/fablabjoinville/bracorobotico/archive/master.zip

```
#include <Servo.h>

//HorizontalDireito
#define CONTROLE_HOR_DIR A1

//VerticalDireito
#define CONTROLE_VER_DIR A0

//HorizontalEsquerdo
#define CONTROLE_HOR_ESQ A2

//VerticalEsquerdo
#define CONTROLE_VER_ESQ A3

//Garra
#define PORT_SERVO_GARRA 2

//Base
#define PORT_SERVO_BASE 3

//CimaBaixo
#define PORT_SERVO_CIMABAIXO 4

//LongePerto
#define PORT_SERVO_PERTOLONGE 5

int valCtrHorDir = 0;
int valCtrVerDir = 0;
int valCtrHorEsq = 0;
int valCtrVerEsq = 0;

int angServoGarra = 90;
int angServoBase = 90;
int angServoCimaBaixo = 90;
int angServoPertoLonge = 90;

Servo servoGarra;
Servo servoBase;
Servo servoCimaBaixo;
Servo servoPertoLonge;

void setup() {
  Serial.begin(9600);
  pinMode(CONTROLE_HOR_DIR, INPUT);
  pinMode(CONTROLE_VER_DIR, INPUT);
  pinMode(CONTROLE_HOR_ESQ, INPUT);
```

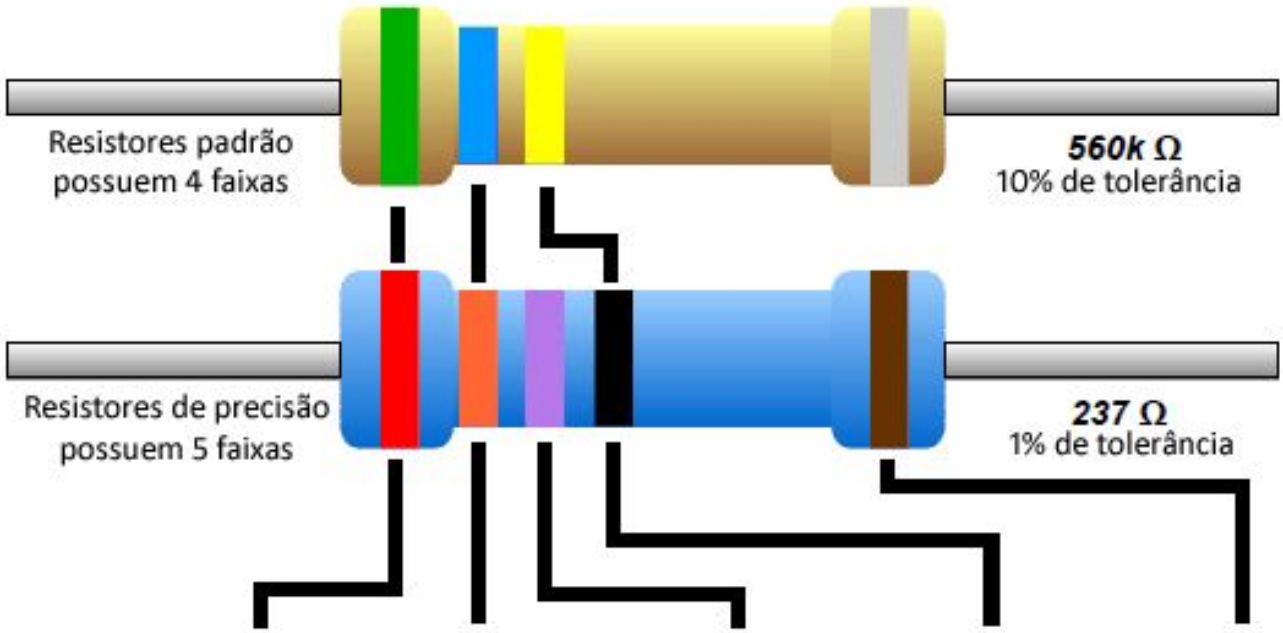
```
47.   pinMode(CONTROLE_VER_ESQ, INPUT);
48.   servoGarra.attach(PORT_SERVO_GARRA);
49.   servoBase.attach(PORT_SERVO_BASE);
50.   servoCimaBaixo.attach(PORT_SERVO_CIMABAIXO);
51.   servoPertoLonge.attach(PORT_SERVO_PERTOLONGE);
52.
53.   //Coloca todos os servos em posição de 90 graus
54.   servoGarra.write(110);
55.   servoBase.write(90);
56.   servoCimaBaixo.write(90);
57.   servoPertoLonge.write(90);
58. }

59. void loop() {
60.   //Le os valores de cada eixo dos joysticks
61.   valCtrHorDir = analogRead(CONTROLE_HOR_DIR);
62.   valCtrVerDir = analogRead(CONTROLE_VER_DIR);
63.   valCtrHorEsq = analogRead(CONTROLE_HOR_ESQ);
64.   valCtrVerEsq = analogRead(CONTROLE_VER_ESQ);
65.
66.   //Inverte a direção
67.   valCtrVerDir = (valCtrVerDir - 1023) * -1;
68.   valCtrVerEsq = (valCtrVerEsq - 1023) * -1;
69.
70.   //Converte os valores dos joystics para os valores
71.   //aceitos pelos servos
72.   angServoGarra = map(valCtrHorEsq, 0, 1023, 0, 180);
73.   angServoBase = map(valCtrHorDir, 0, 1023, 0, 180);
74.   angServoPertoLonge = map(valCtrVerDir, 0, 1023, 0, 180);
75.   angServoCimaBaixo = map(valCtrVerEsq, 0, 1023, 0, 180);
76.
77.   //Limita o ângulo máximo
78.   //de abertura da garra
79.   if(angServoGarra < 110){
80.     angServoGarra = 110;
81.   }
82.
83.   //Posiciona os servos de arcordo
84.   //com os ângulos convertidos
85.   servoGarra.write(angServoGarra);
86.   servoBase.write(angServoBase);
87.   servoCimaBaixo.write(angServoCimaBaixo);
88.   servoPertoLonge.write(angServoPertoLonge);
89. }
```

Desafio

Utilizando os comandos aprendidos, tente criar um programa que faça o braço robótico pegar um objeto em um ponto A, movê-lo para um ponto B e soltá-lo.

Código de Cores

A extremidade com mais faixas deve apontar para a esquerda



Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	x 1 Ω	
Marrom	1	1	1	x 10 Ω	+/- 1%
Vermelho	2	2	2	x 100 Ω	+/- 2%
Laranja	3	3	3	x 1K Ω	
Amarelo	4	4	4	x 10K Ω	
Verde	5	5	5	x 100K Ω	+/- .5%
Azul	6	6	6	x 1M Ω	+/- .25%
Violeta	7	7	7	x 10M Ω	+/- .1%
Cinza	8	8	8		+/- .05%
Branco	9	9	9		
Dourado				x .1 Ω	+/- 5%
Prateado				x .01 Ω	+/- 10%