Controlando um braço robótico com micropython

Juliana Karoline - @julianaklulo

Juliana Karoline

@julianaklulo

pyladies São Carlos

grupy-sanca

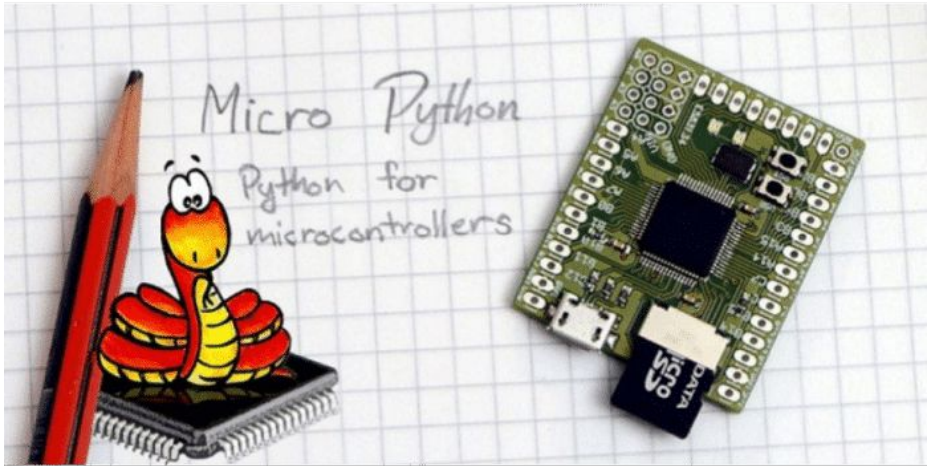
sancaLUG

Sumário

Micropython

MicroPython - Projeto

- ➔ Criado por Damien George - físico e programador australiano
- ➔ Fundado no Kickstarter em 2013
- ➔ Incluía a placa PyBoard + periféricos
  - **Sucesso:** £97k em apoios



MicroPython - Especificações

- Implementa a gramática do Python 3.4
- Escrito em C, possui diferenças do CPython
  - http://docs.micropython.org/en/latest/genrst/index.html#cpython-diffs

- Inclui compilador, parser e máquina virtual e executador runtime
- RAM mínima para compilar e executar print("Hello World!"): 4KiB

Fluxo de execução do MicroPython

MicroPython - Suporte

O suporte oficial inclui, atualmente:

- ESP8266
- ESP32
- WiPy

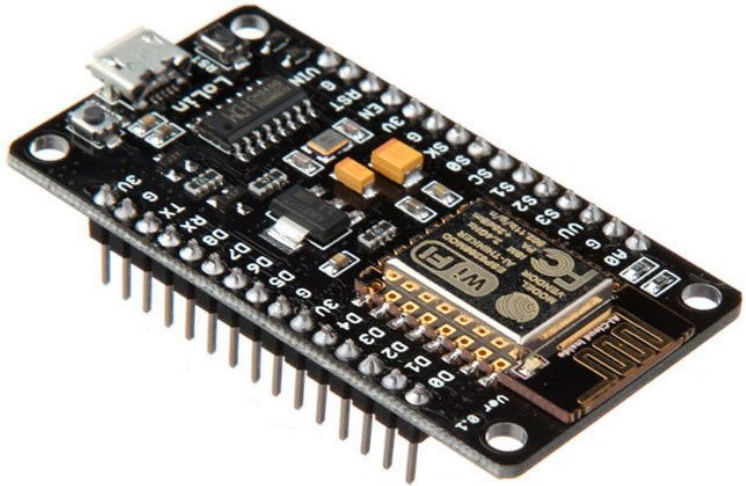
Placas da Adafruit somente suportam o CircuitPython

NodeMCU (ESP8266)

NodeMCU (ESP8266)

Plataforma para desenvolvimento em IoT
Usa o chip ESP8266 (microcontrolador + WiFi)
Possui regulador de tensão e conversor USB-serial

Compatível com Lua e Arduino, além do MicroPython



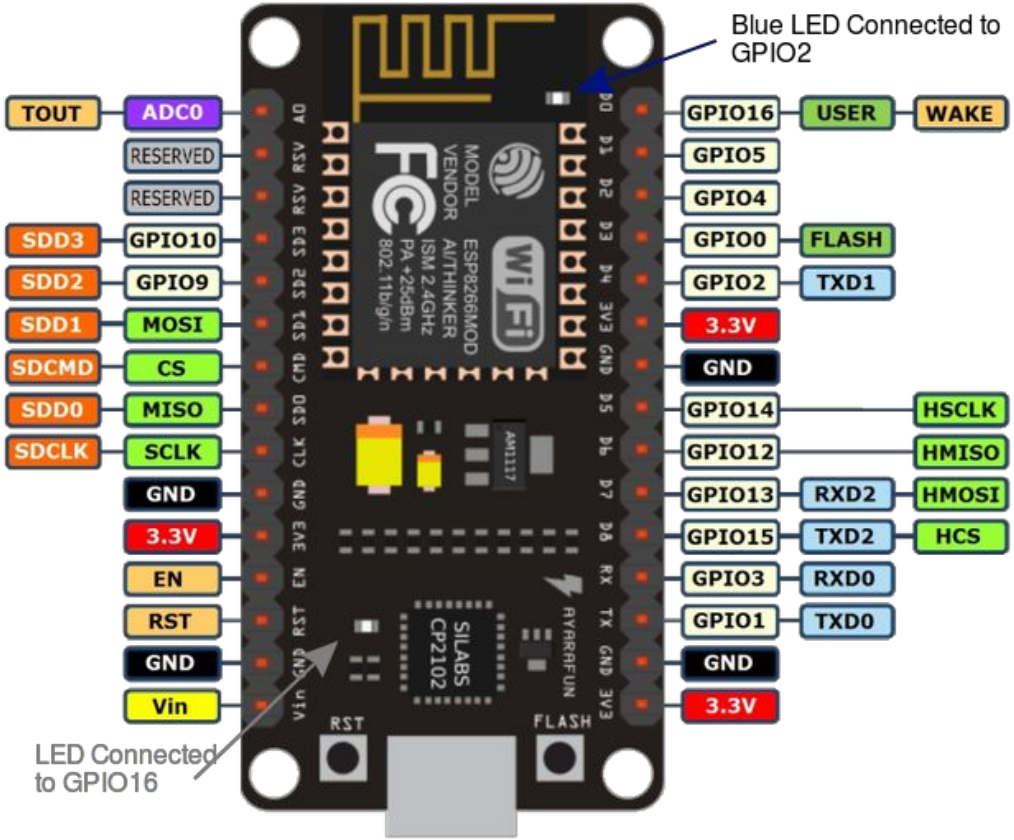
Vantagens:

- baixo custo (+- R$20,00)
- baixo consumo de energia
- pronta para uso

NodeMCU (ESP8266)

Especificações:

- ➔ Microprocessador 32-bits
- ➔ 4MB de flash
- ➔ 64KB de RAM
- ➔ 16 GPIOs
  - ◆ PWM
  - ◆ I2C
  - ◆ SPI
- ➔ ADC
- ➔ WiFi
- ➔ Tensão: 3.3V (!)



Primeiros Passos

Primeiros passos

Passos necessários para iniciar o desenvolvimento:

1. Flasheando o firmware
2. Conectando no REPL
3. Gerenciando arquivos com o ampy
4. Hello World: blink

1: Flasheando o Firmware

Recomendações:

- ➔ Usar o esptool
- ➔ Apagar a flash antes de iniciar
- ➔ Usar 460800 de baudrate

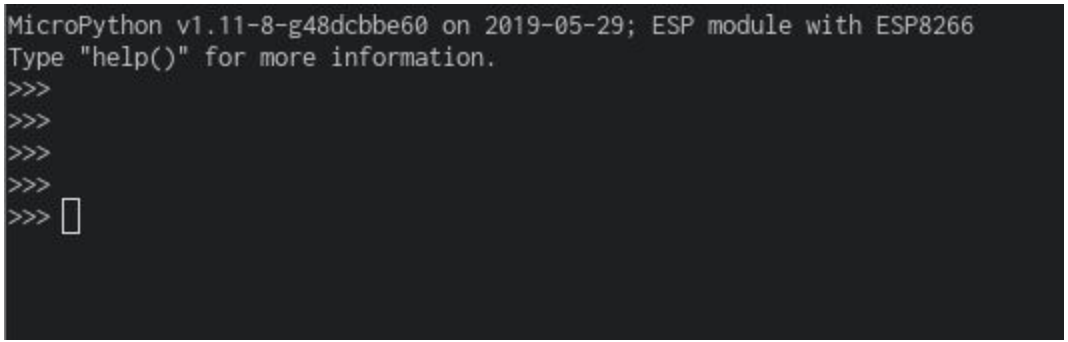
```
pip install esptool
esptool.py --port /dev/ttyUSB0 erase_flash
esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash --flash_size=detect 0 fw.bin
```

2: Conectando no REPL

REPL: Read Evaluate Print Loop

Sempre disponível na UART0
Usar baudrate de 115200
Precisa de um terminal multiplexer (ex: screen)

```
# screen /dev/ttyUSB0 115200
```



3: Gerenciando arquivos com o ampy

ampy: Adafruit MicroPython Tool

Permite carregar e executar arquivos na placa
Precisa de acesso ao REPL para funcionar

```
pip install adafruit-ampy
```

Comandos disponíveis:

- ➔ run
- ➔ put
- ➔ get
- ➔ rm

3: Gerenciando arquivos com o ampy

Para transferir um script para a placa, usa-se o **put**

```
# ampy --port /dev/ttyUSB0 put arquivo.py
```

Arquivos na raiz da placa com esses nomes executam automaticamente:

- **boot.py**
  - é executado uma vez durante o boot da placa
- **main.py**
  - é executado logo após o boot

Normalmente coloca-se um laço infinito dentro do main.py

3: Gerenciando arquivos com o ampy

Para executar um script sem persistir na placa, usa-se o **run**

```
# ampy --port /dev/ttyUSB0 run --no-output blink.py
```

Caso o arquivo imprima informações, a exibição ocorre após o fim da execução do script
Para scripts com laço infinito, passa-se --no-output para não bloquear o terminal

Para interromper a execução é preciso acessar o REPL

4: Hello World: Blink

blink.py

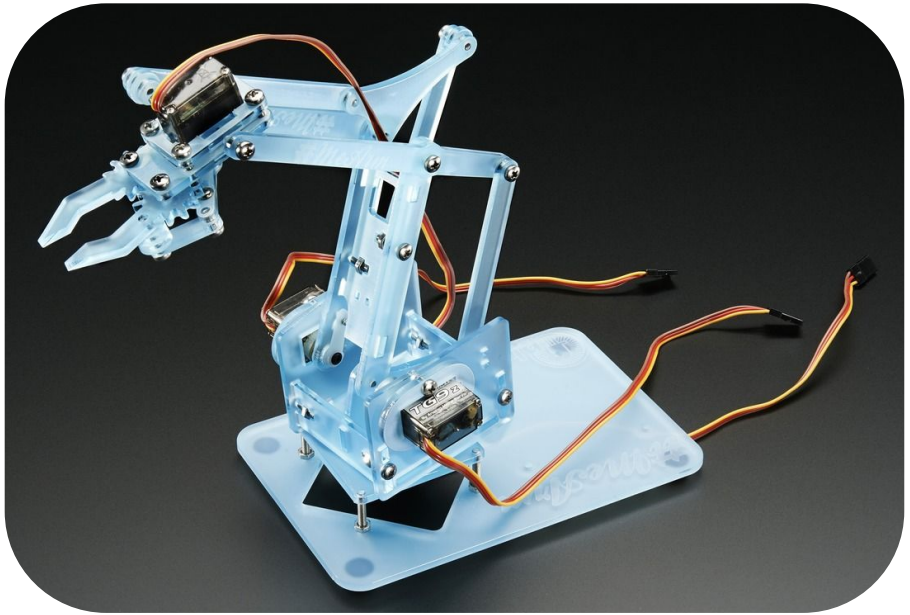
```
1  from machine import Pin
2  from time import sleep
3  led = Pin(2, Pin.OUT)
4
5  while True:
6    led.on()
7    print("LED ON")
8    sleep(0.1)
9    led.off()
10   print("LED OFF")
11   sleep(0.1)
```

MeArm

MeArm - Sobre o projeto

- ➔ Criado em março de 2014
- ➔ Inspirado no uArm
- ➔ Feito para ser DIY
- ➔ Ideal para iniciantes e crianças

Open Source!



Exemplo de um MeArm montado

MeArm - Componentes

- ★ Folha A4 de acrílico ou MDF
- ★ Parafusos genéricos
- ★ 4 Micro **servos** de hobby
- ★ Placa de prototipagem
- ★ Fonte de alimentação

Ilustração dos componentes

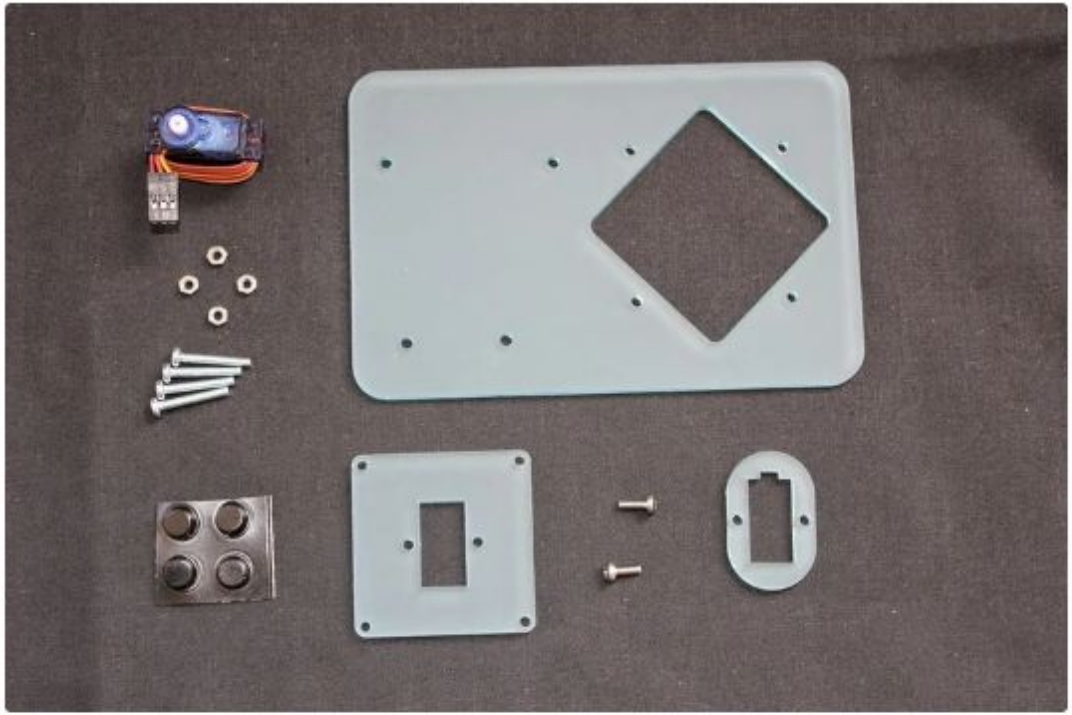
MeArm - Valores

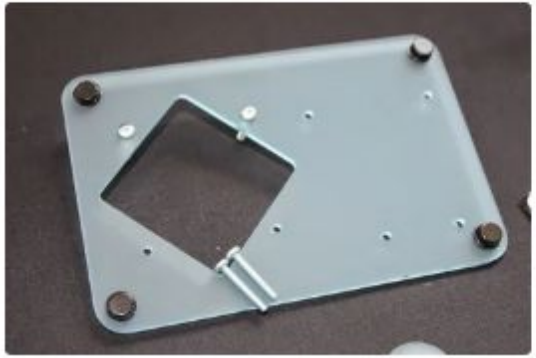
Kit cortado + parafusos: +-R$45,00
Micro servo: +-R$15,00 (R$3,00 na China)
NodeMCU: +-R$20,00 (R$8,00 na China)
Fonte de alimentação: +-R$15,00

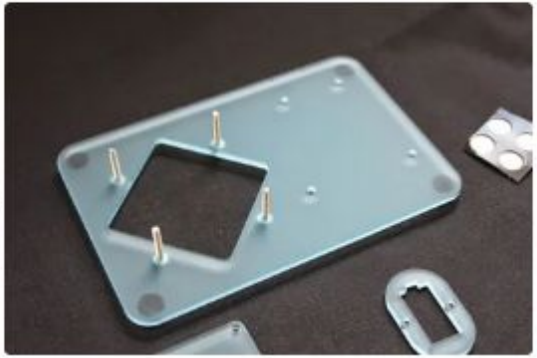
Total: aproximadamente R$100

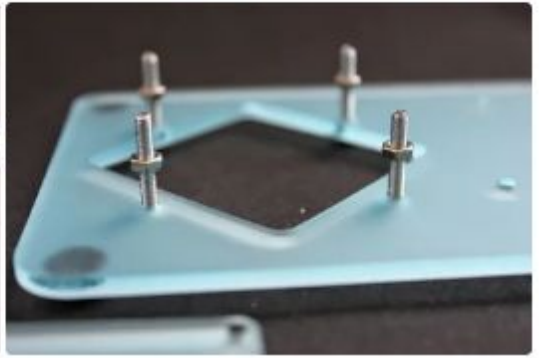
Montagem

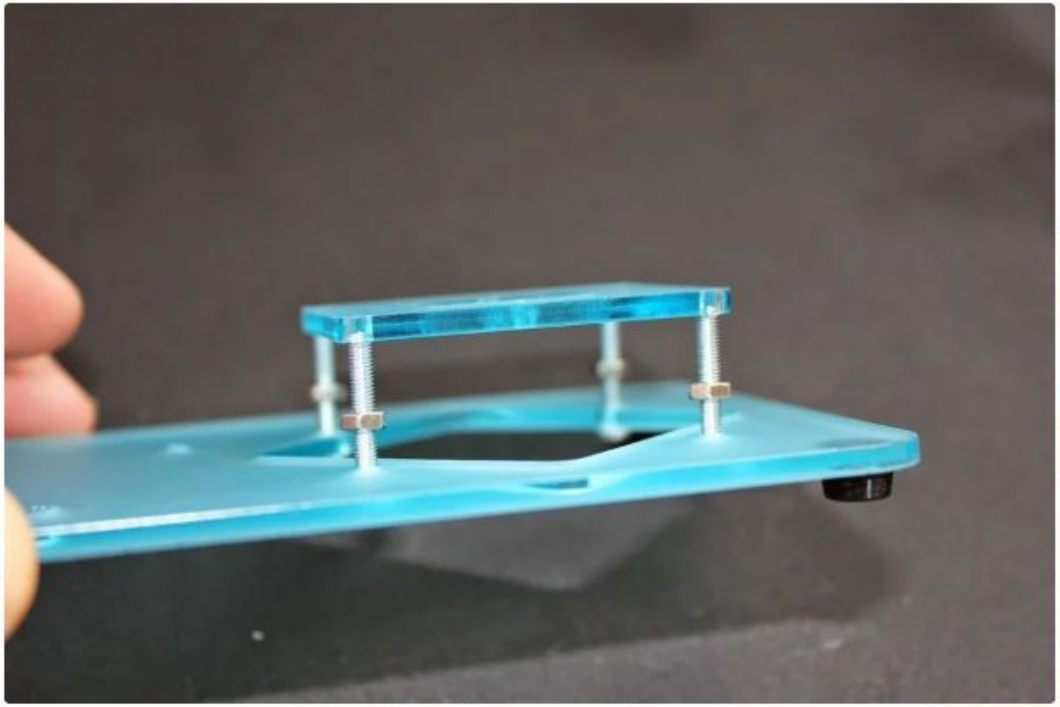
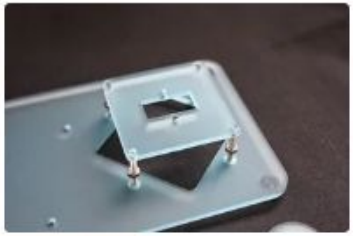
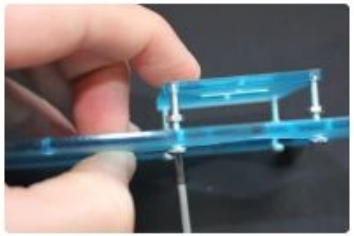
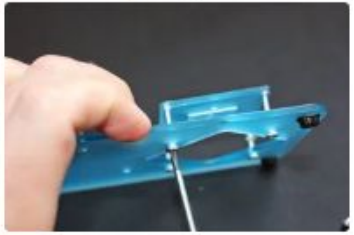
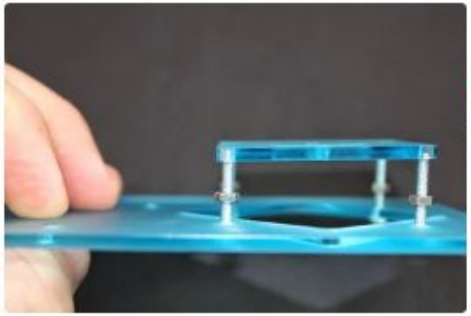
Imagens:
https://www.instructables.com/id/Pocket-Sized-Robot-Arm-meArm-V04/

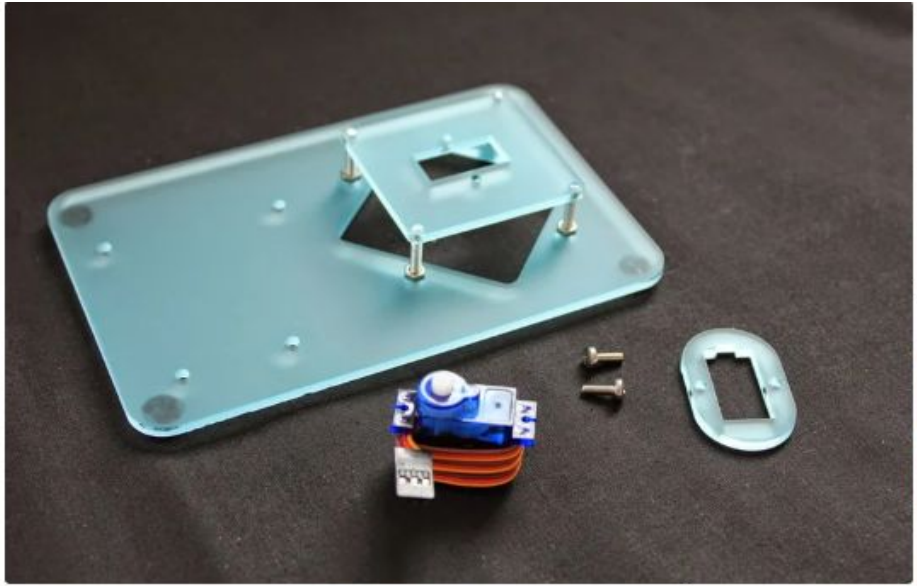
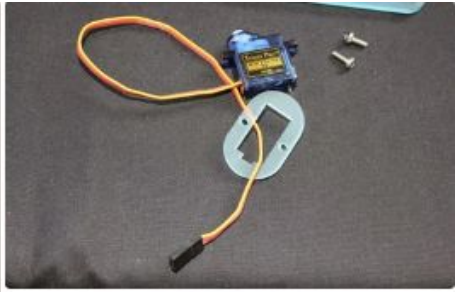
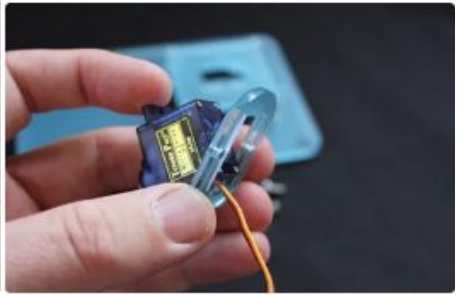


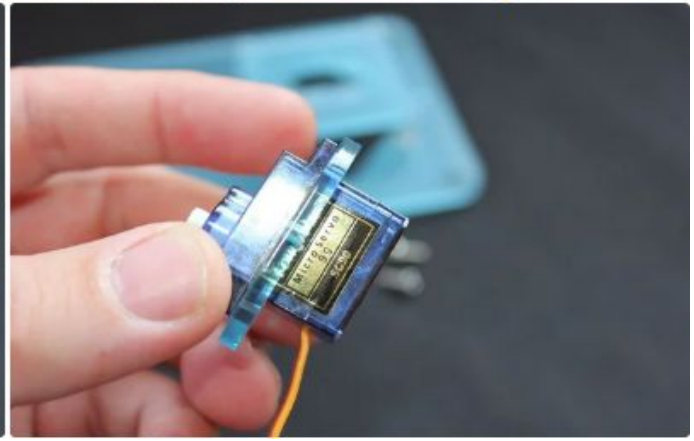


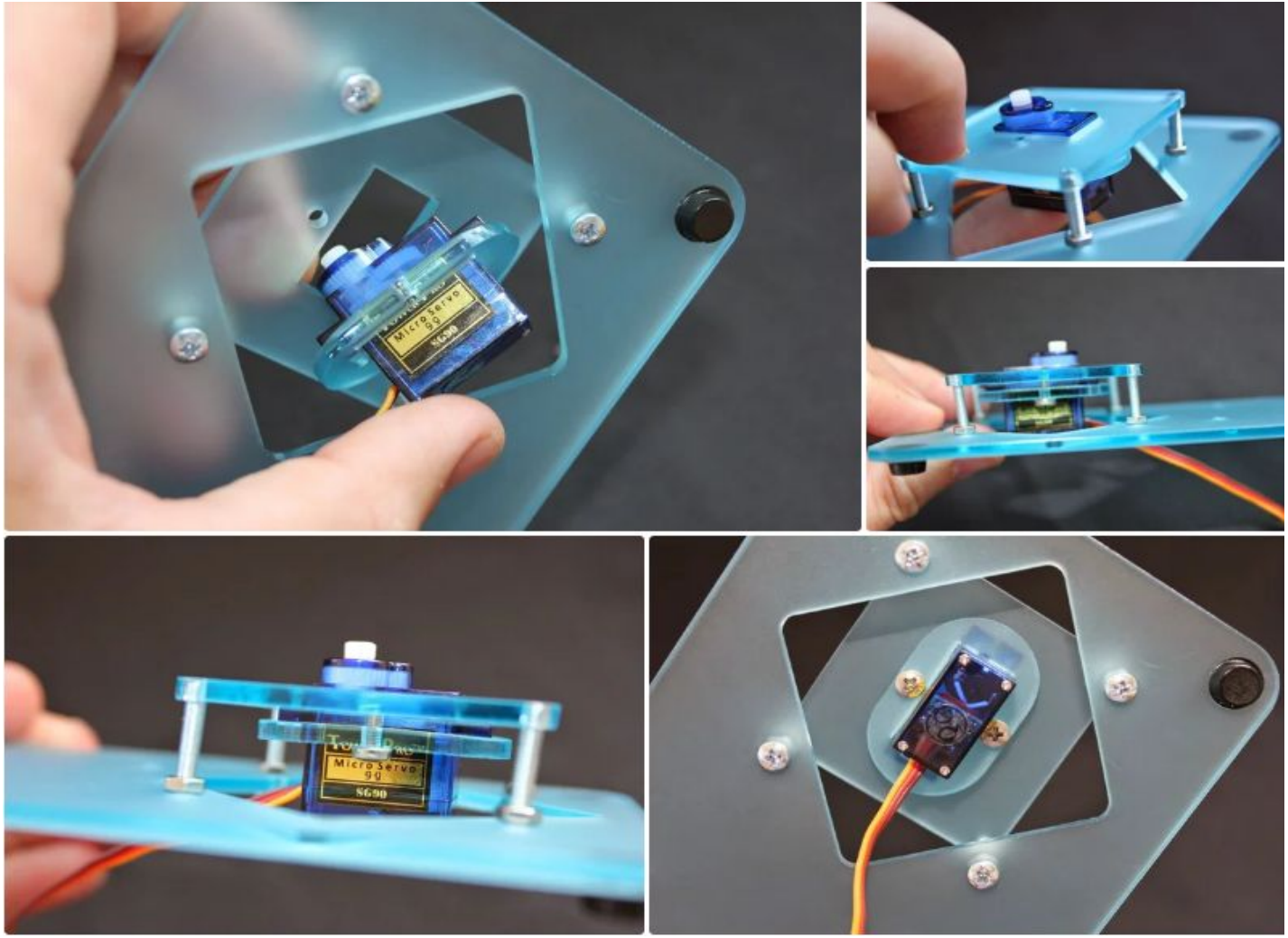


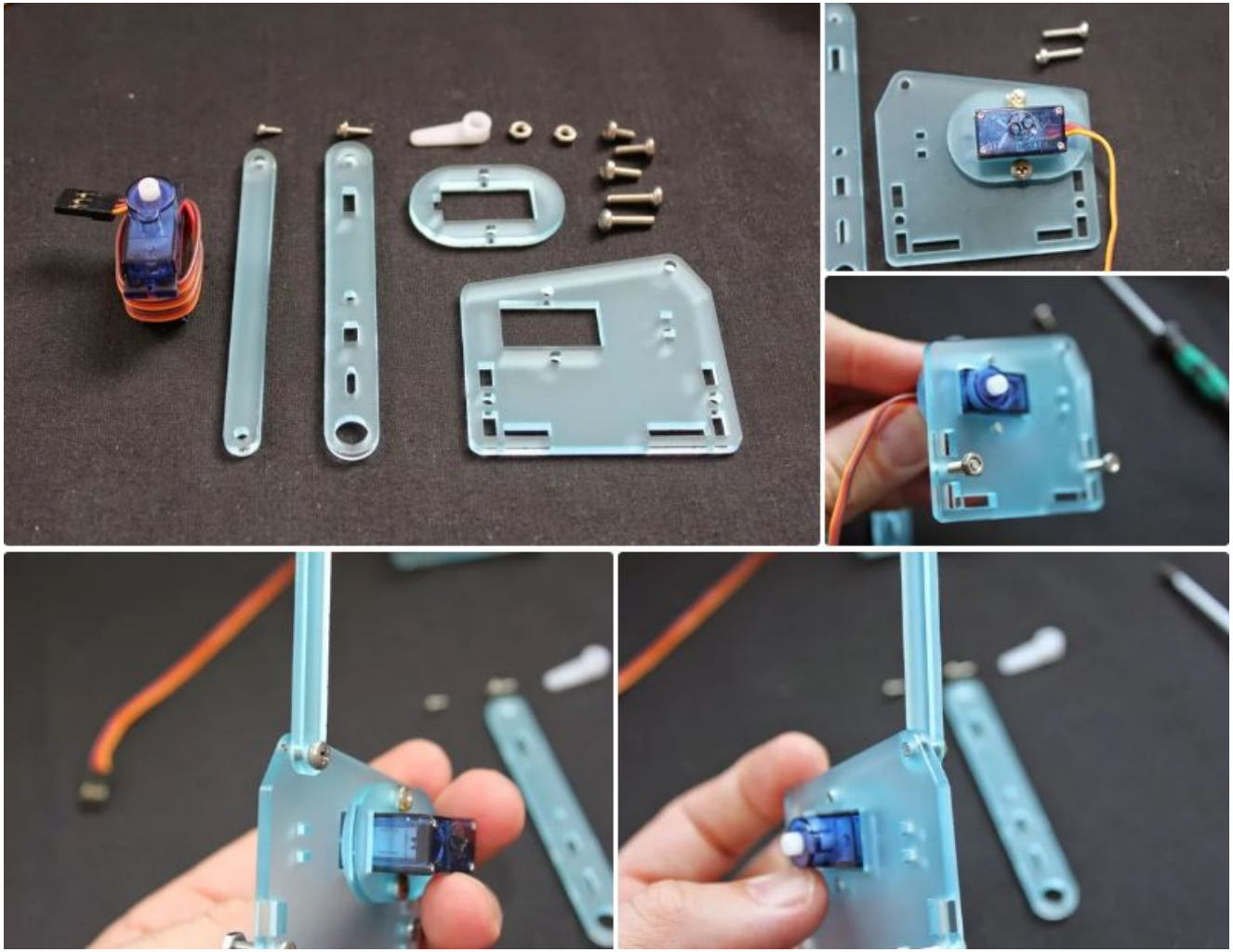


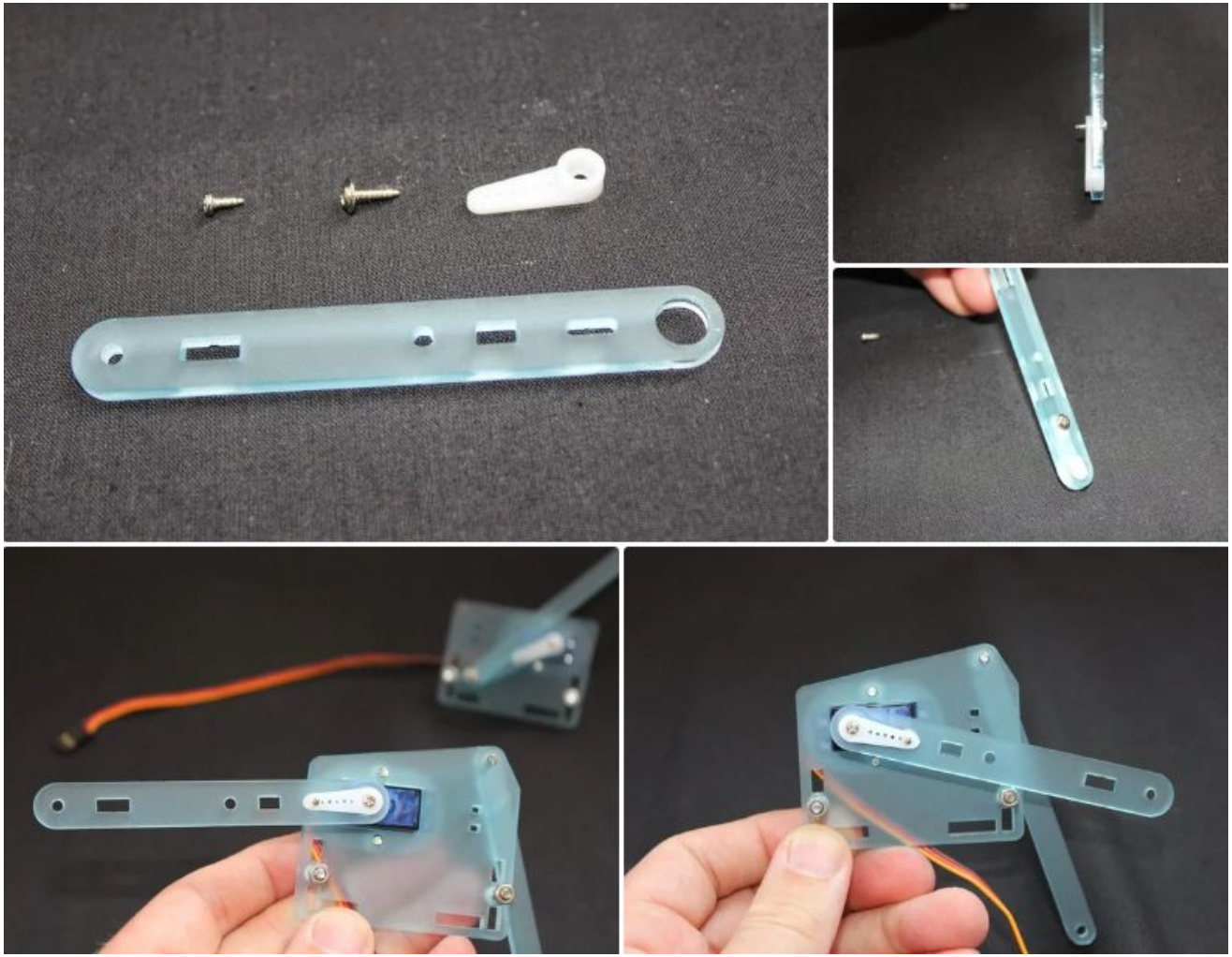






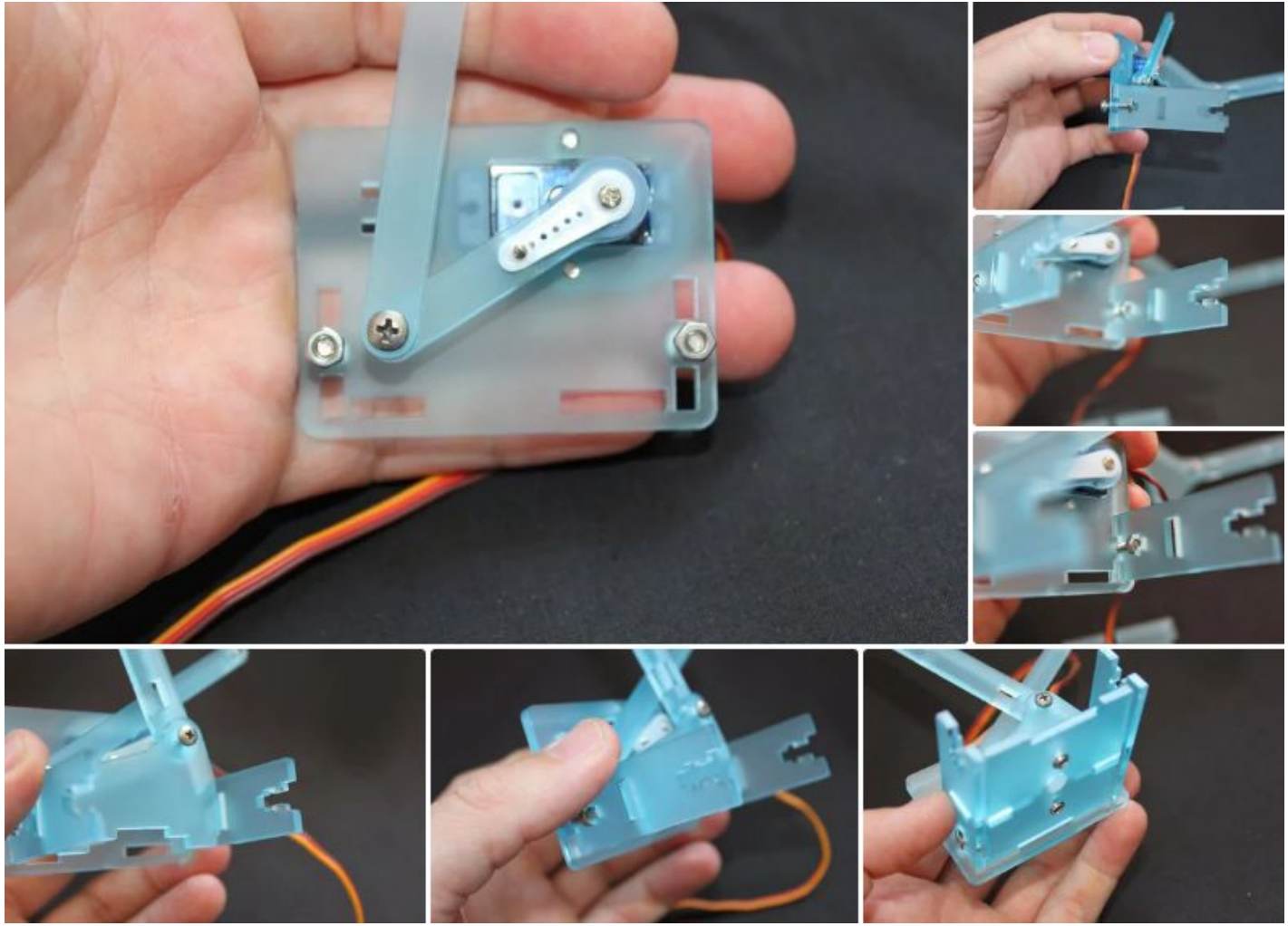




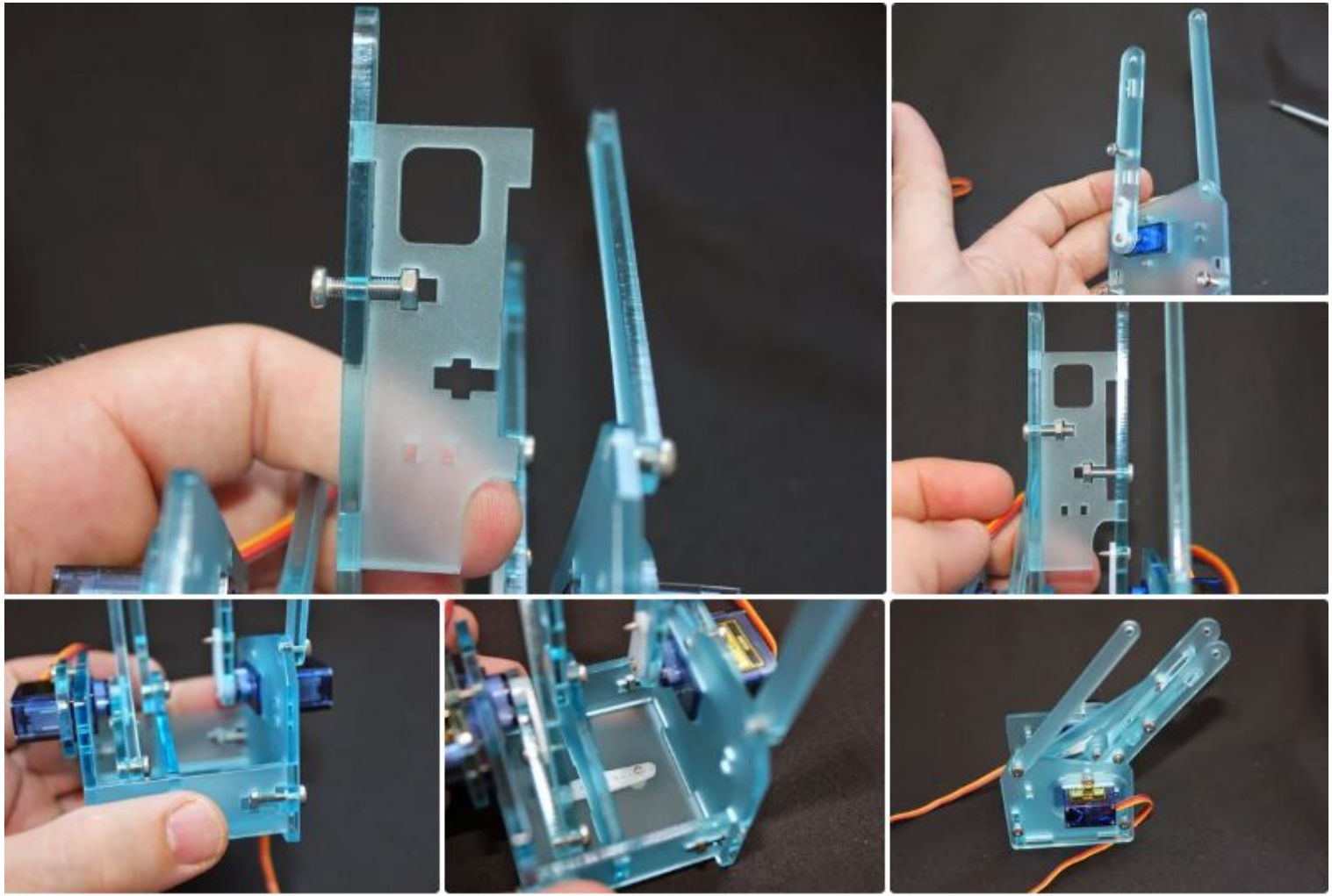


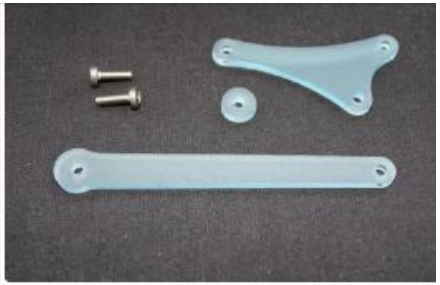
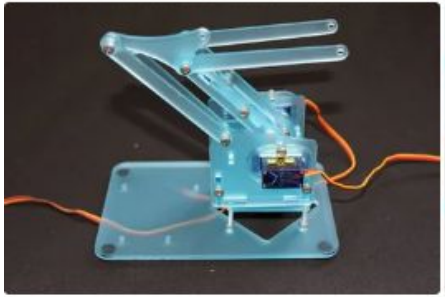
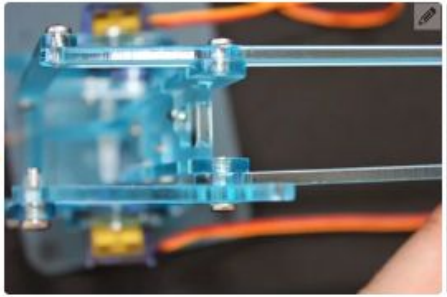
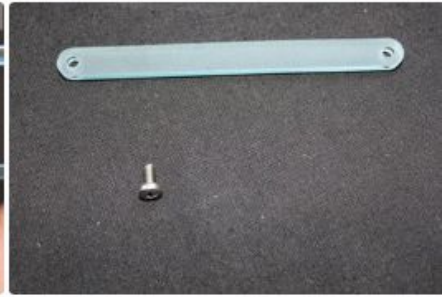
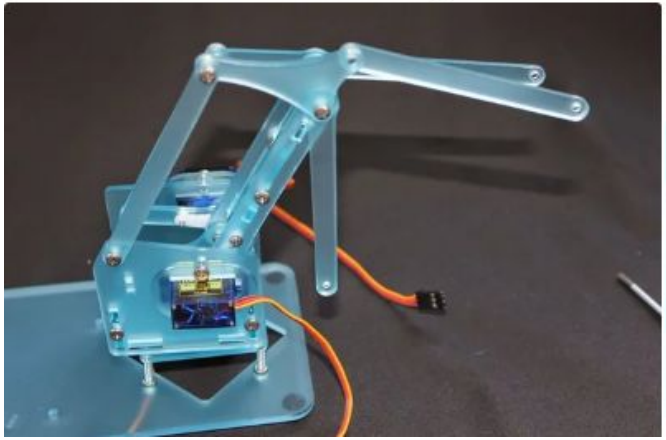
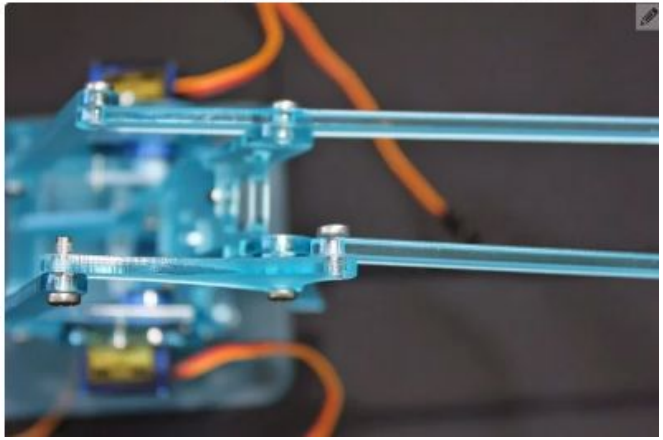

Micro Servo
9g
SG90

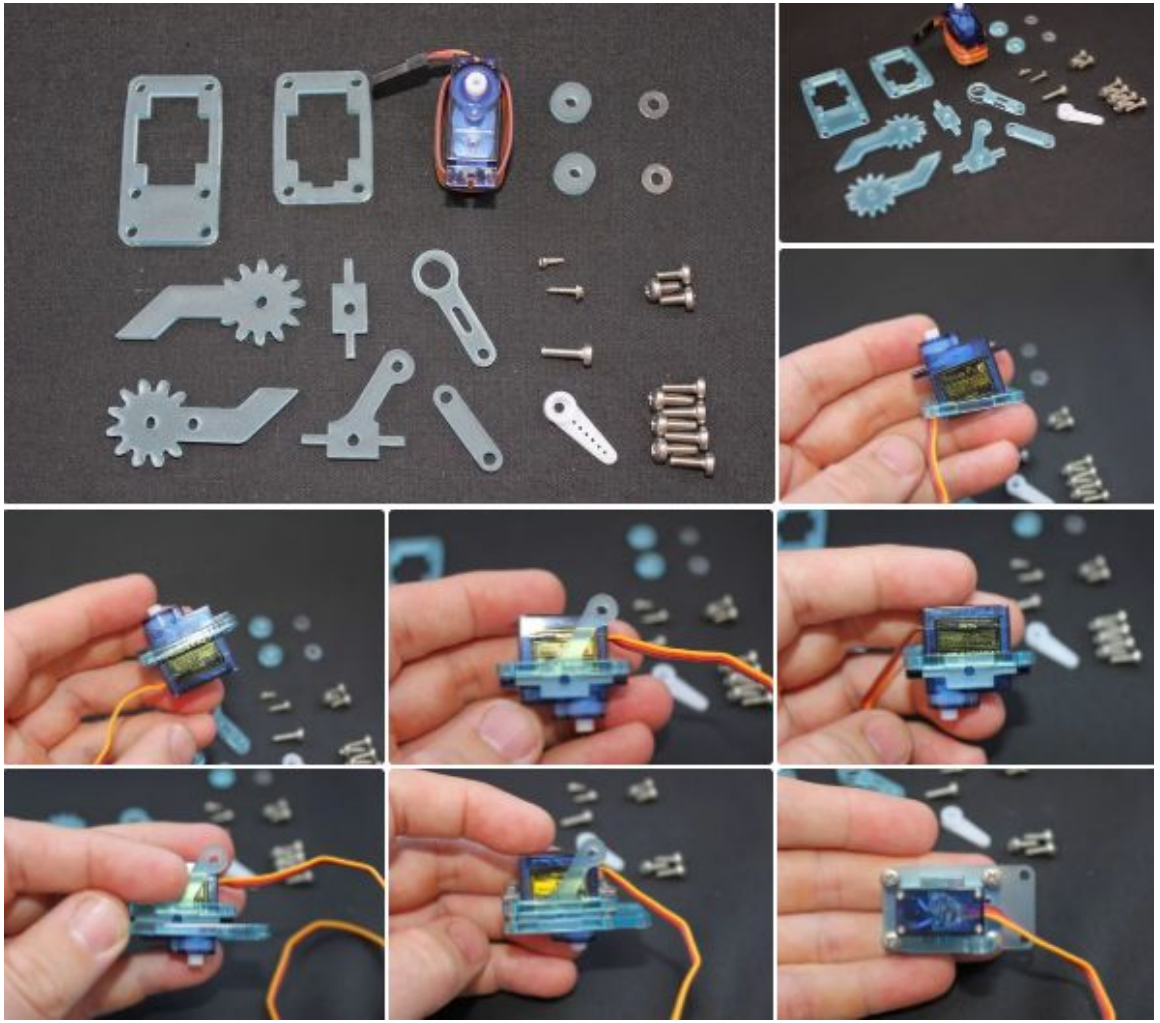


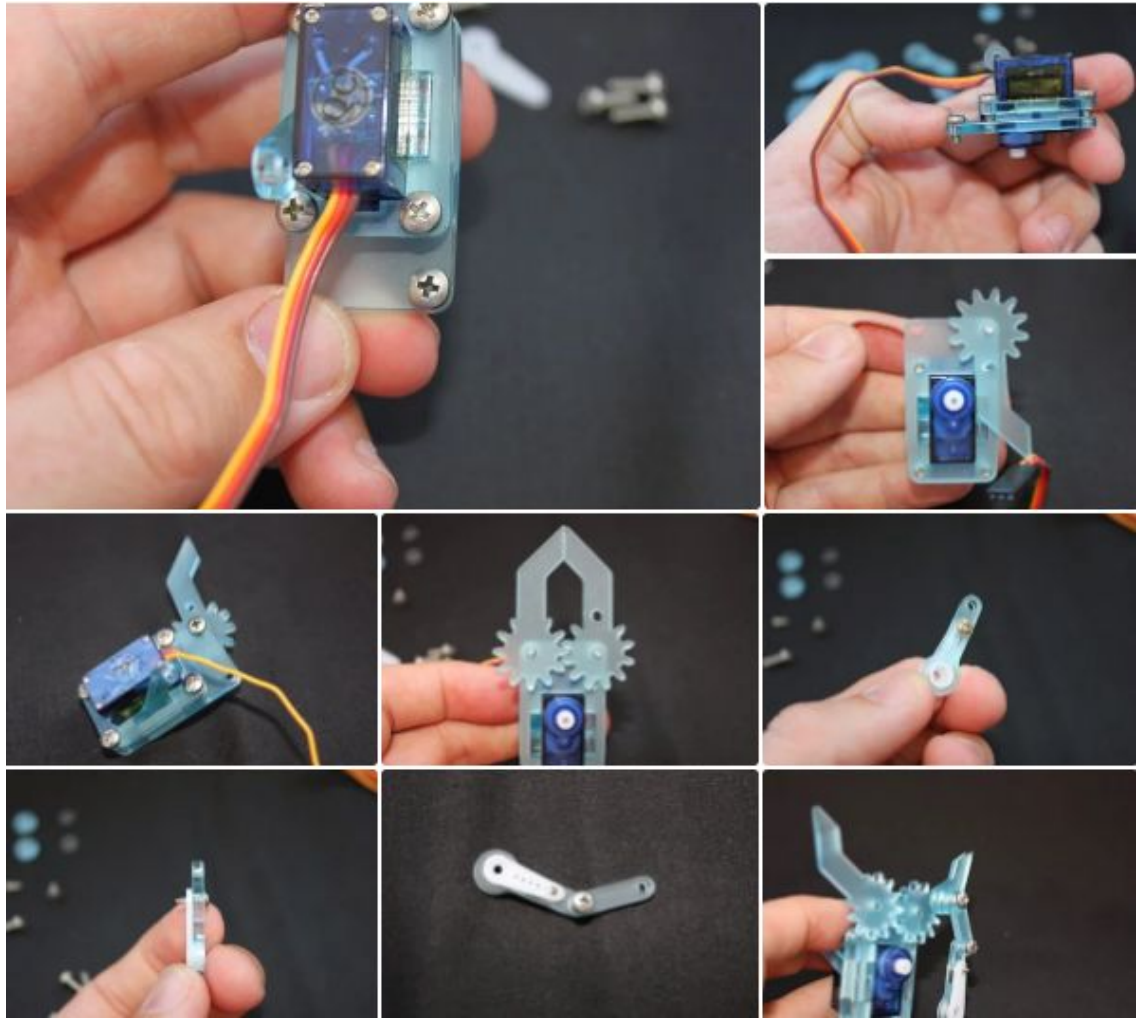


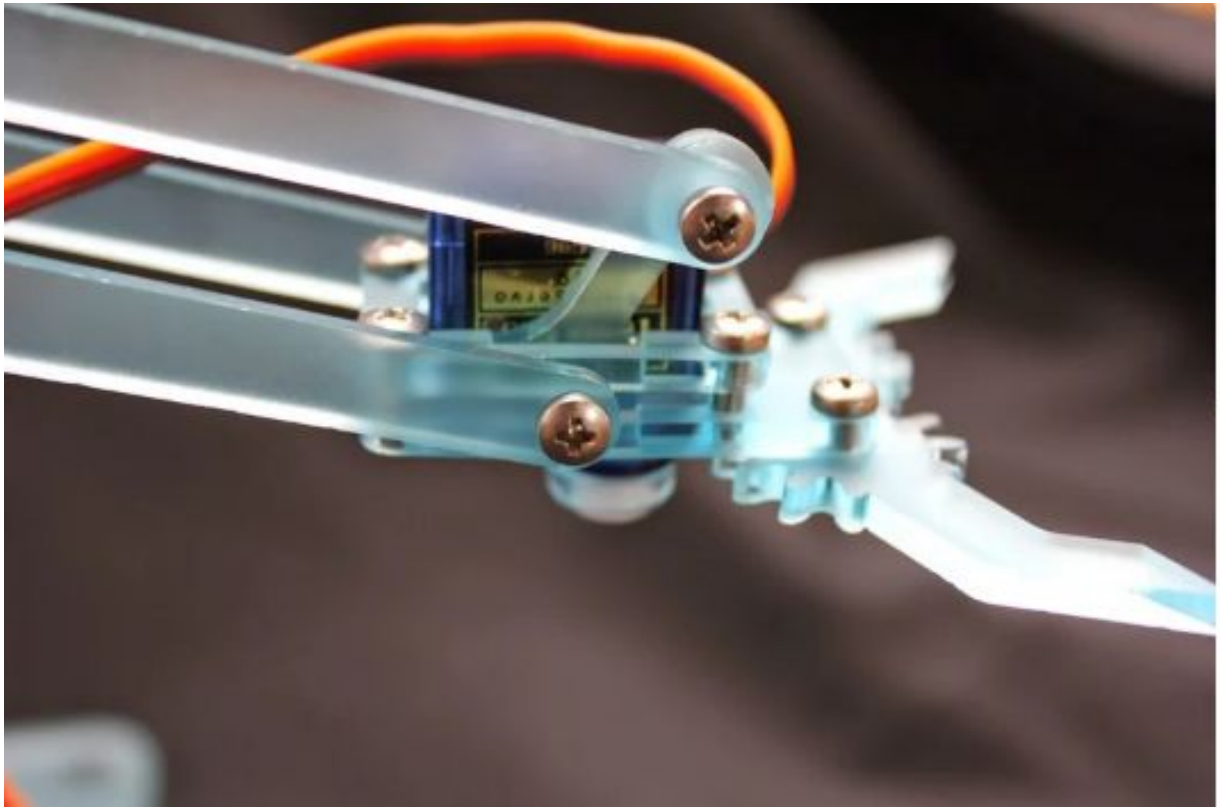


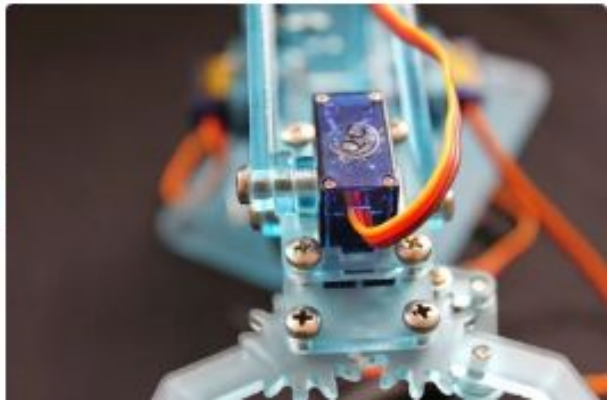










Calibragem

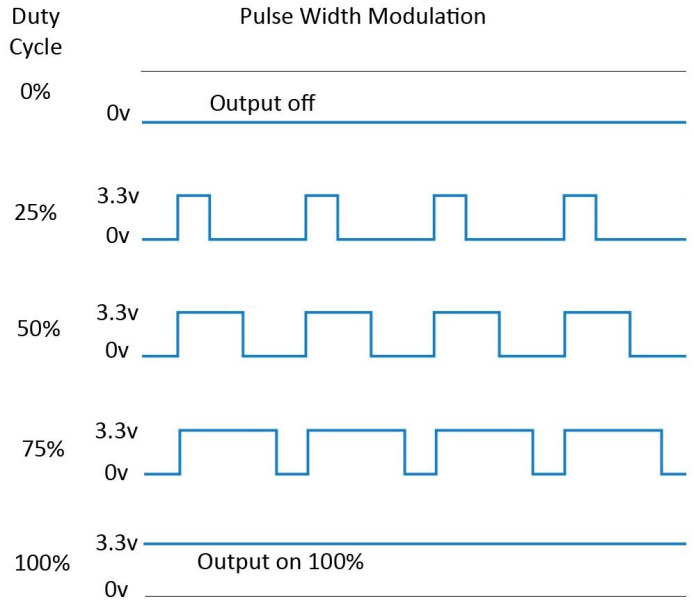
MeArm - Calibragem

PWM (modulação de largura de pulso)

- ➔ forma digital de gerar sinal analógico
- ➔ usado para definir o ângulo de cada servo
- ➔ possui frequência e ciclo de trabalho

Valores recomendados na documentação do MicroPython:

- ➔ **50Hz de frequência**
- ➔ **40 a 115 de duty cicle (77 no centro)**



MeArm - Calibragem

Definir os valores máximos e mínimos:

➔ **tentativa e erro**

```
Base: 30 a 115
Eixo X: 60 a 115
Eixo Y: 60 a 105
Garra: 30 a 55
```

calibragem_servo.py

```python
from machine import Pin, PWM
from time import sleep

base = PWM(Pin(14), freq=50)

# base: 30 a 115

while True:
  for i in range(30, 115, 5):
    base.duty(i)
    sleep(0.1)

  sleep(0.3)

  for i in range(115, 30, -5):
    base.duty(i)
    sleep(0.1)

  sleep(0.3)
```

Calibragem

Prova de Conceito

Obrigada!

Contato: @julianaklulo