

INSTITUTO FEDERAL
Sudeste de Minas Gerais

Campus
Barbacena



Disciplina: Programação para Dispositivos Móveis e Sem Fio

6º Período – 2019-2

Prof.: Rafael José Alencar

MyArmControl Do It Yourself

Tarilles Roman Sfredo
Vinícius dos Santos Tristão

Sumário

Introdução	2
Construindo seu braço	3
Juntando suas partes	3
Preparando a base	3
Construa o lado esquerdo	5
Construindo o lado direito	7
Reunindo os lados e conhecendo o porco	8
Casando com a base!	10
Antebraços esquerdo e direito.	10
A garra...	11
Dicas pós montagem	15
Conectando seu braço robô ao controlador arduino	16
Conectando seu controlador ao controle remoto Android	19
Considerações finais	26

Introdução

O MyArmControl é um trabalho realizado para a disciplina de Programação para Dispositivos Móveis e Sem Fio, durante o segundo semestre de 2019. O trabalho proposto pelo professor Rafael José Alencar ao aluno Tarlles Roman Sfredo tinha como objetivo construir um projeto Arduino funcional que se comunicasse com um aparelho Android. A solução apresentada pelo aluno foi a construção de um braço robótico que pudesse ser controlado através do aplicativo.

Este projeto é uma adaptação ao projeto MeArm¹ e tem por objetivo a tradução livre de seu manual de montagem, criação de um código Arduino que permite ao braço robótico ser controlado através de um aplicativo de celular e a criação do Aplicativo Android em si.

Como o trabalho pretendia demonstrar a comunicação entre essas duas tecnologias, Android e Arduino, o código Java Android demonstrado aqui trata apenas dessa comunicação, visto que você pode poderá personalizar a cara do seu aplicativo e realizar a comunicação utilizando o código demonstrado. Porém um aplicativo básico está disponível para download².

¹ → <https://www.thingiverse.com/thing:360108>

² → <https://github.com/TarllesRoman/MyArmControl>

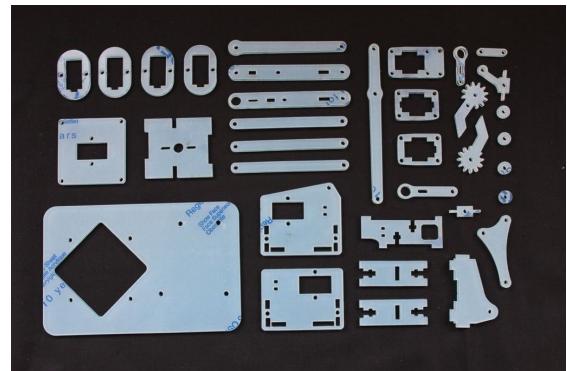
Construindo seu braço

Juntando suas partes

Posteriormente você irá precisar de parafusos e porcas. Iremos usar M3 (métrica de 3mm). A quantidade necessária será:

<input type="checkbox"/> Porcas	x10
<input type="checkbox"/> 6mm	x9
<input type="checkbox"/> 8mm	x12
<input type="checkbox"/> 10mm	x3
<input type="checkbox"/> 12mm	x7
<input type="checkbox"/> 20mm	x3

Você também irá precisar de 4 motores servo, todas as partes¹ impressas em material de sua preferência, nós utilizamos madeira MDF e as fotos demonstradas aqui foram retiradas do projeto original e por isso demonstram o robô em acrílico.



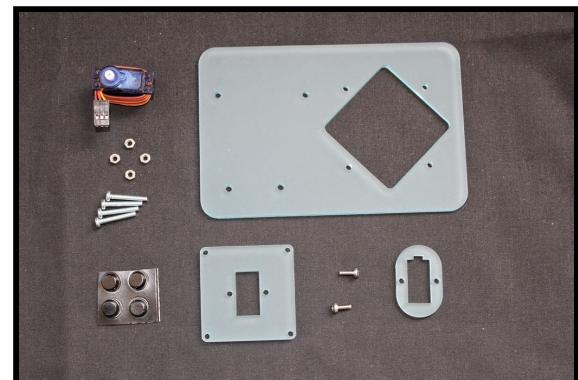
¹ →

<https://www.thingiverse.com/thing:360108/files>

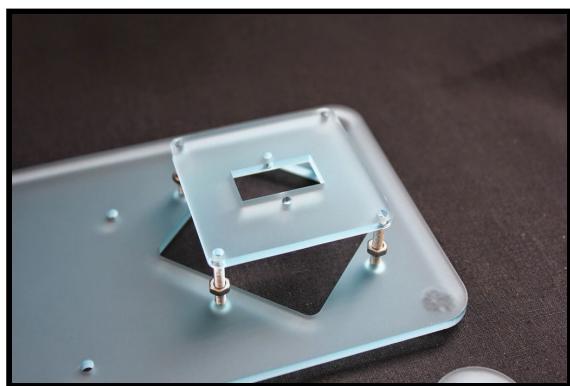
Preparando a base

As partes que você irá usar aqui são:

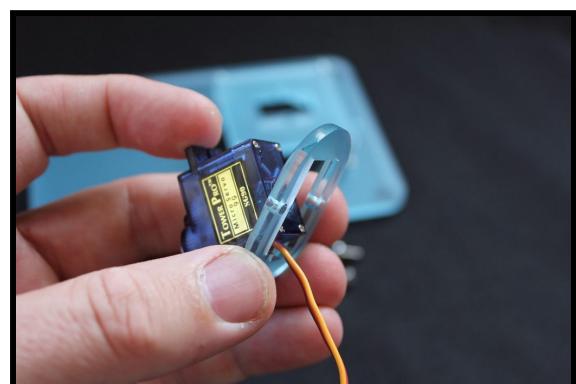
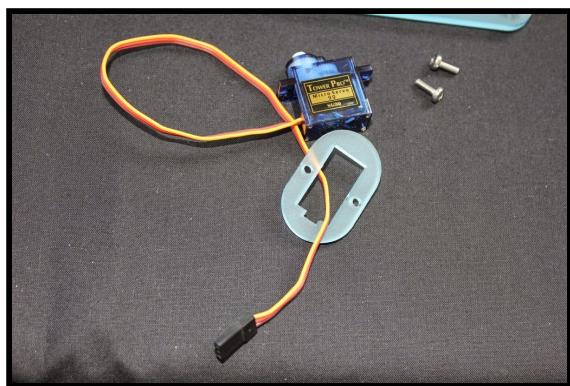
- Base - A parte mais larga que você têm.
- Colar - 1 dos 4 fornecidos
- Quadrado para montagem do servo - Cabe diretamente no buraco da base.
- 20mm x4
- Porcas x4
- 8mm x2
- Pêns x4
- Servo x1



Cole um dos quatro pés pegajosos em cada um dos cantos da base. Em seguida, comece a inserir os parafusos de 20 mm nos orifícios ao redor do orifício quadrado grande. Agora torça as porcas nos parafusos de 20 mm da parte superior até que você esteja na metade do caminho. Em seguida, pegue a parte quadrada e coloque-a em cima dos parafusos de 20 mm, com o retângulo recortado na mesma direção da base (como mostrado na figura).

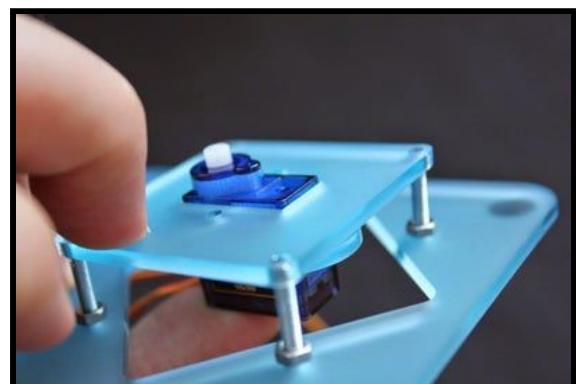
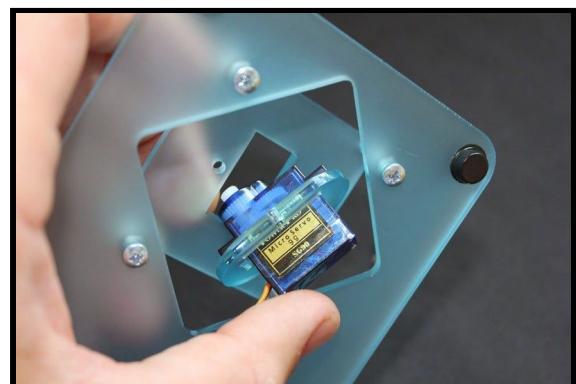


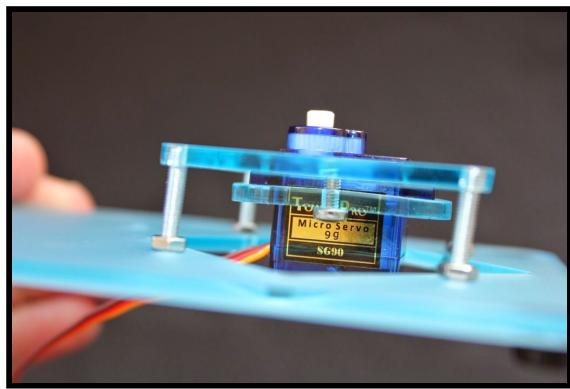
Passe o fio do servo através do colar Alinhe o recorte na gola com a extremidade do servo onde o fio prende Coloque a coleira na parte inferior do servo. Empurre para casa para que fique plana com a flange no servo.



Agora você colocou a linha do servo no gancho e empurre-a através do orifício em forma de servo na parte quadrada.

Insira os dois parafusos de 8 mm por baixo para que eles passem pelos orifícios do colar com pouca resistência e dê um tapinha na parte quadrada. Aperte até que o servo seja mantido firmemente. Não apertar demais!

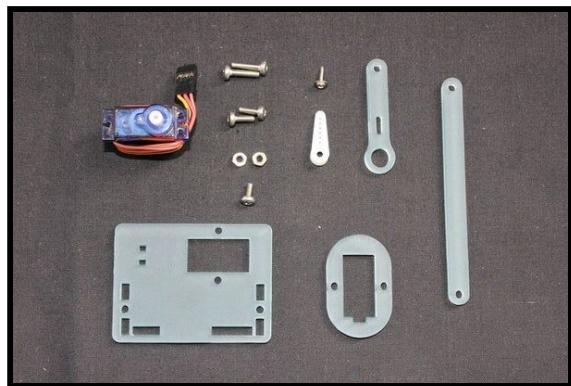




Construa o lado esquerdo

As partes que você precisa aqui são:

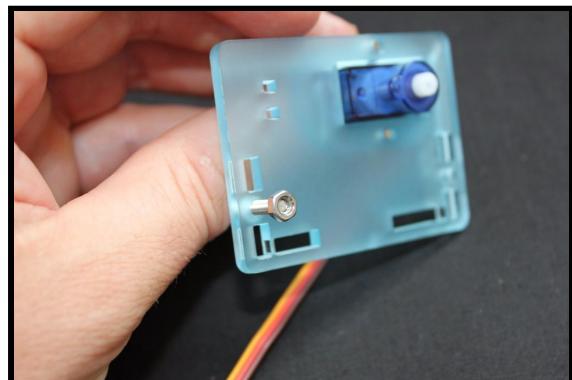
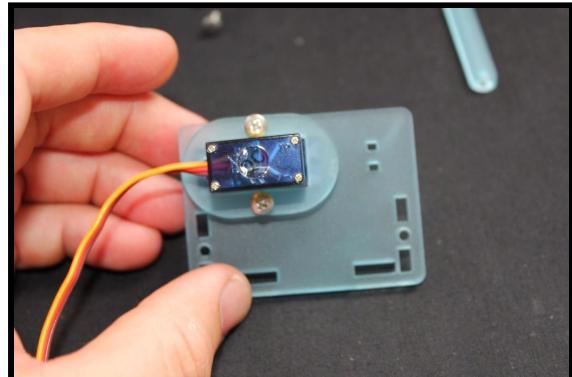
- 8mm x2
- 12mm x2
- Porca x2
- 6mm x1
- Colar x1
- Lado retangular como na figura.
- O maior braço que você tiver, como na figura.
- 1 alavanca, você possui 3 iguais.
- Servo x1
- 1 parafuso servo curto.
- 1 parafuso servo longo.



Rosqueie o servo como antes e aparafuse na peça lateral usando os parafusos de 8 mm.

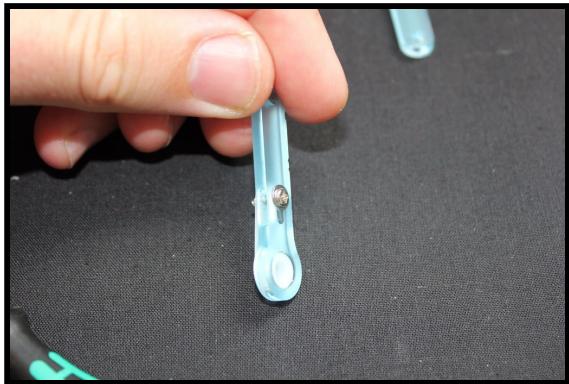
Preste muita atenção à orientação aqui. Observe a direção do fio e a maneira como o servo sai da peça lateral.

Agora passe os parafusos de 12 mm pelos orifícios redondos restantes e coloque as porcas apenas meia volta.



Prenda a buzina de servo de plástico branco na peça da alavanca de servo (como na figura) usando o parafuso servo longo. Isso vai aparecer na parte de trás da peça resultante e é um pouco pontudo. Eu as apareço quando tiver certeza de que as juntei corretamente!





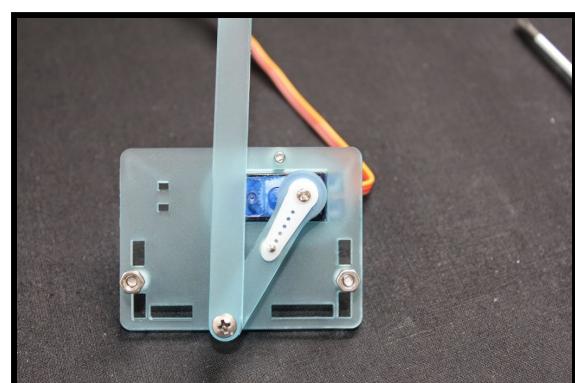
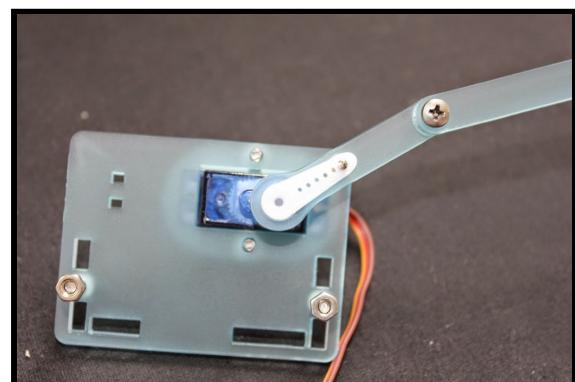
Agora conecte a alavanca longa à alavanca do servo, com o parafuso passando pelo mesmo lado da buzina do servo. Esta é sua primeira parte em movimento. Você pode ajustá-lo mais tarde, mas é importante movê-lo com muito pouca força. Você deseja encontrar o equilíbrio certo entre não se mover lateralmente e se mover livremente. Cada força que você precisa para superar o movimento dessas articulações é menos força que você tem para levantar objetos.



Prenda a alavanca do servo que você acabou de criar no servo, ele apenas pressiona. Esses servos menores girarão manualmente, girando-o com cuidado no sentido horário até parar. Quando parar, puxe o braço do servo de volta e coloque-o para que ele corresponda à primeira imagem mostrada aqui.

Coloque o pequeno parafuso do servo no meio e aperte um pouco para que ele grude - não aperte demais - por algum motivo, esse parafuso pode travar o servo e não queremos isso.

Se o seu servo clicar aqui, significa que ele está pulando os dentes e pode precisar de reconstrução; no pior caso, uma engrenagem quebrou, então substitua-o.



Construindo o lado direito

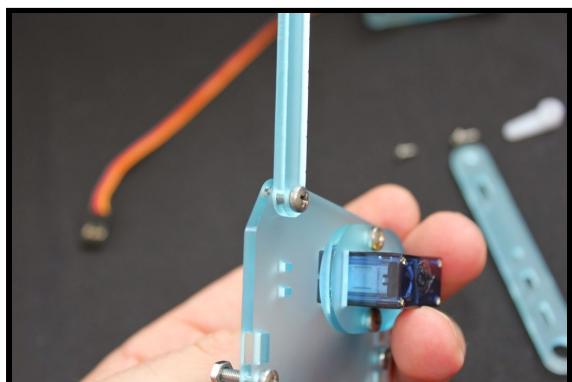
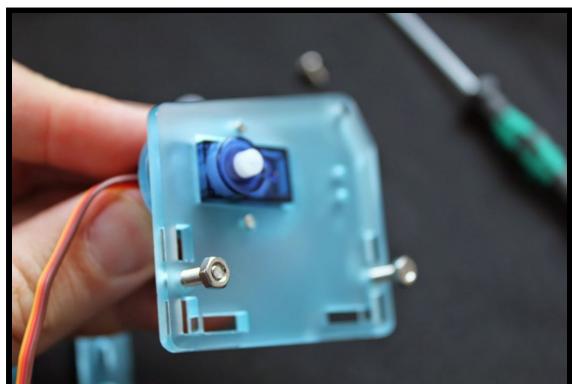
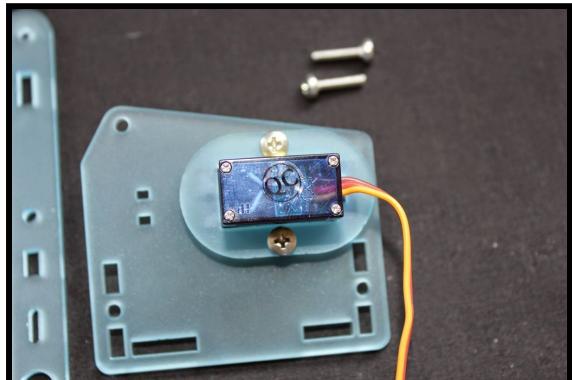
Isso surpreendentemente é igual a construção do lado esquerdo. As partes que você precisa aqui são:

- 8mm x2
- 12mm x2
- Porcas x2
- 6mm x1
- Colar x1
- A peça do lado direito como na figura.
- O maior braço que você tiver, como na figura.
- Alavanca central do lado direito. (Olhe cuidadosamente o da figura)
- Servo x1
- 1 parafuso servo curto.
- 1 parafuso servo longo.
- 1 braço do servo

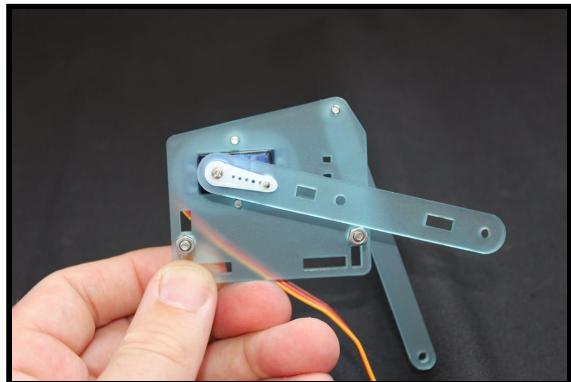
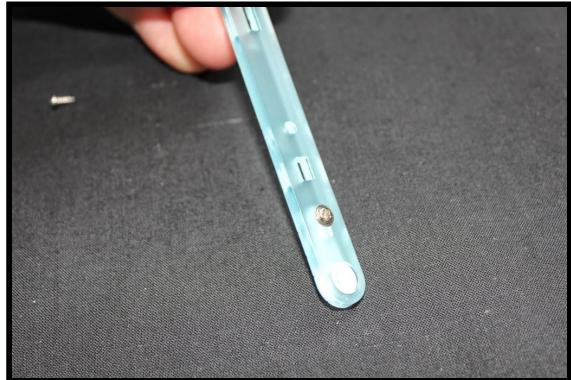


Passe a gola e prenda, observando cuidadosamente a orientação. Insira os parafusos de 12 mm e gire as porcas pela metade.

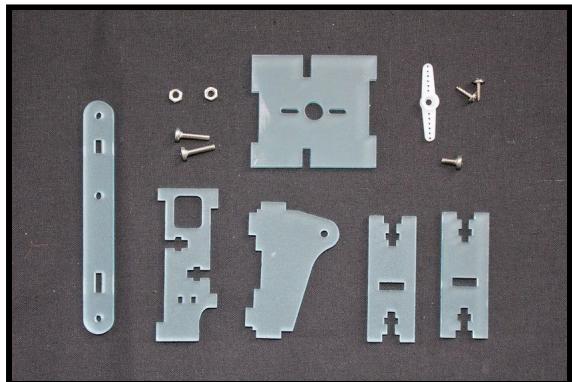
Prenda a alavanca longa na parte externa da peça RH Side com o parafuso de 6 mm. Esta é outra parte que precisa se mover. Lembre-se de que é bom e solto, eles devem se mover de avião, por assim dizer. Se estiver rígido no início, trabalhe para trás e para frente algumas vezes.



Pegue novamente a servo-buzina de plástico e prenda-a na seção central longa. Empurre-o para o servo e gire-o gentilmente no sentido anti-horário. Remova a alavanca e coloque-a novamente para coincidir com a primeira das três imagens mostradas aqui. Insira o parafuso pequeno (sem apertar demais!) E gire no sentido horário para que corresponda à última imagem mostrada aqui.



- Base do berço, quadrado.
- Alavanca central esquerda.
- 2 extremidades do berço.
- "O porco".
- Seção central da alavanca central.
- Braço longo do servo.
- 2 parafusos longos do servo.
- 1 parafuso curto do servo.

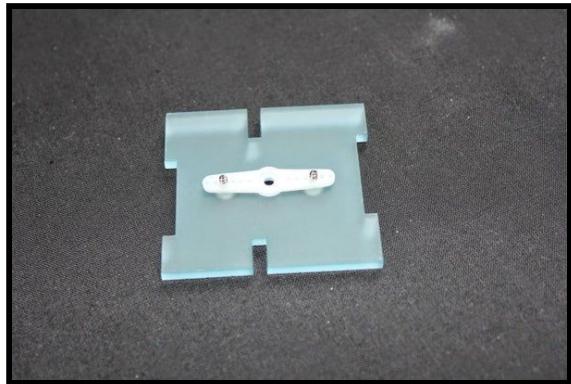


Reunindo os lados e conhecendo o porco

Agora vamos unir os lados com as partes centrais e conhecer uma das minhas peças favoritas pessoais "O Porco". O porco é mostrado na segunda imagem aqui anexada a uma alavanca longa, uma das iterações desta parte parecia muito com um porco e o nome ficou comigo. As partes necessárias são:

- 12mm x2
- Porca x2
- 6mm x1

Use os dois parafusos longos para prender esta buzina. Recortes para a esquerda, como na imagem.

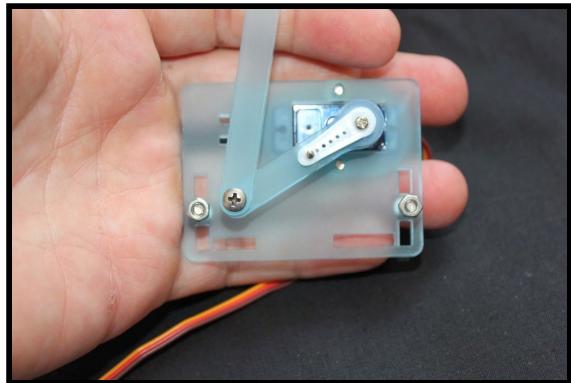
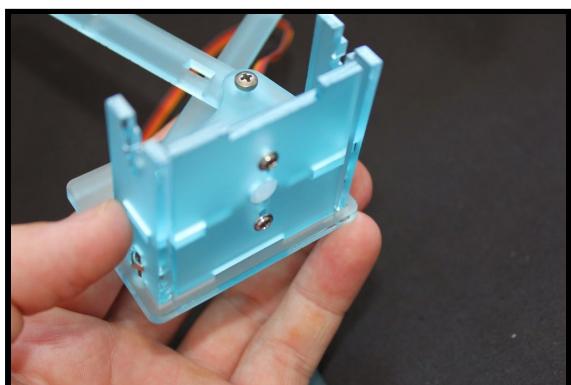
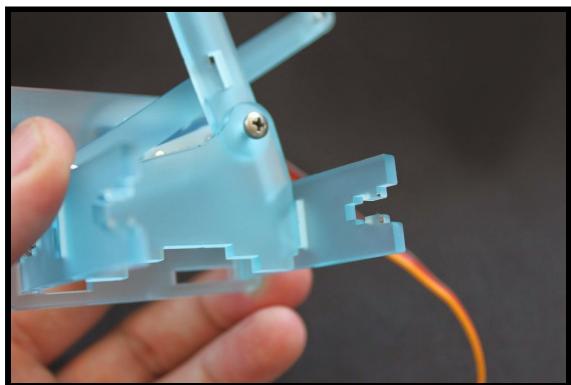
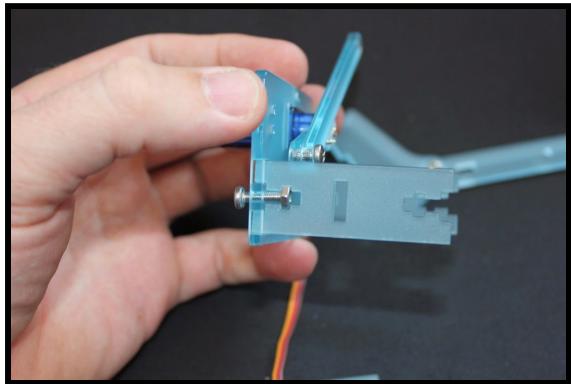


Pegue no LHS montado (lado esquerdo!), Certificando-se de que os parafusos de 12 mm que anexamos anteriormente sejam empurrados para a direita através da inserção de uma peça do berço final, para que o recorte fique mais próximo do lado esquerdo. Aperte esse parafuso uma ou duas voltas, mas não completamente.

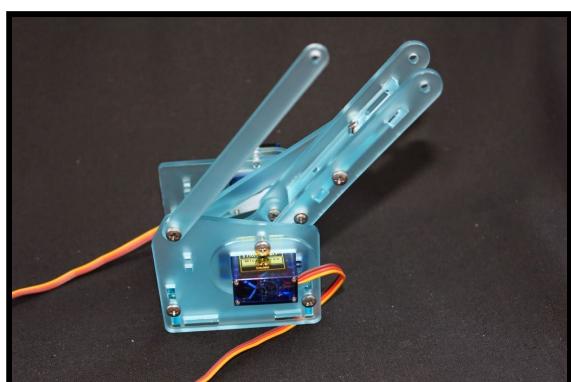
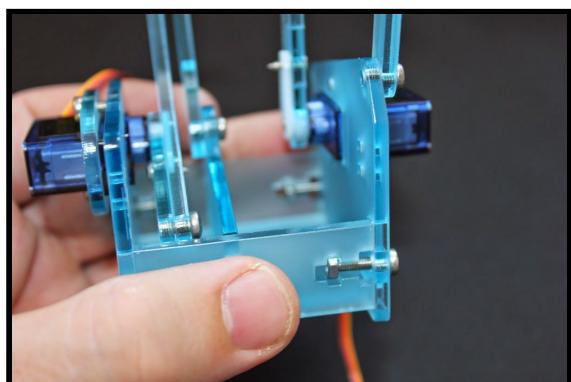
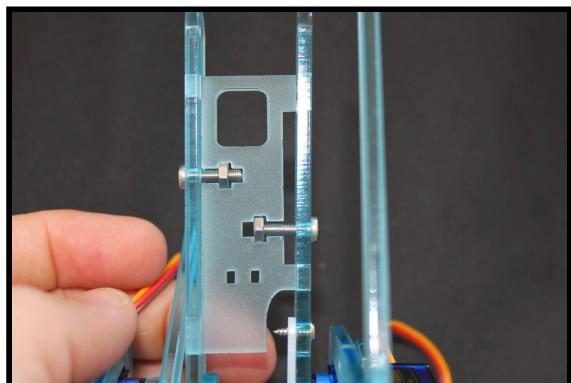
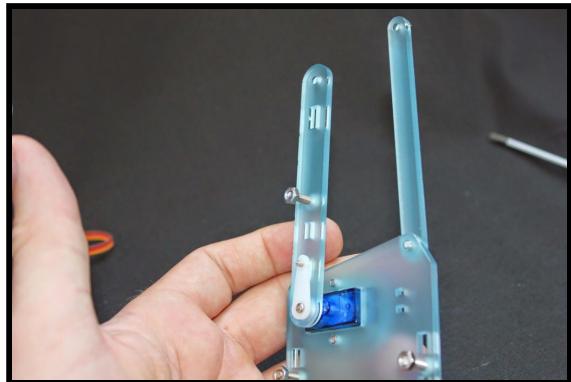
Agora faça o mesmo com a outra parte final. Com os parafusos soltos, tente inserir o porco entre os recortes, ele deve encaixar e segurar, no entanto, dependendo do corte e da sua sorte, desapertar um parafuso, se necessário.

Depois de reunir isso e ainda com alguma folga nos parafusos, você pode encaixar a base do berço. Agora aperte, mas não aperte demais.

Verifique se todas as peças se parecem com essas imagens.

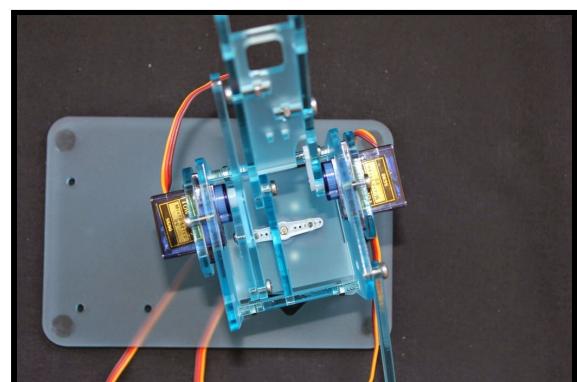
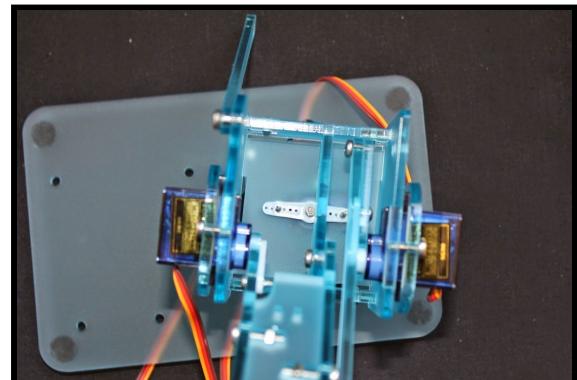


Agora vamos trazer o RHS para a festa. Usando as duas porcas restantes e os dois parafusos de 12 mm, conecte a alavanca central livremente. Guie a base e o LHS que você acabou de montar sobre os parafusos e porcas de 12 mm no RHS e aperte tudo (não aperte demais!).



Casando com a base!

Empurre o berço montado no servo base. Gire no sentido horário e remova-o, coloque-o novamente como mostrado na segunda imagem aqui e coloque o pequeno parafuso (não muito apertado!). Verifique se ele gira no sentido anti-horário e se parece com a última imagem aqui.



Antebraços esquerdo e direito.

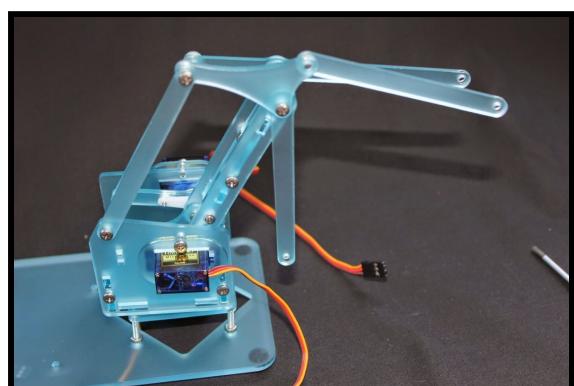
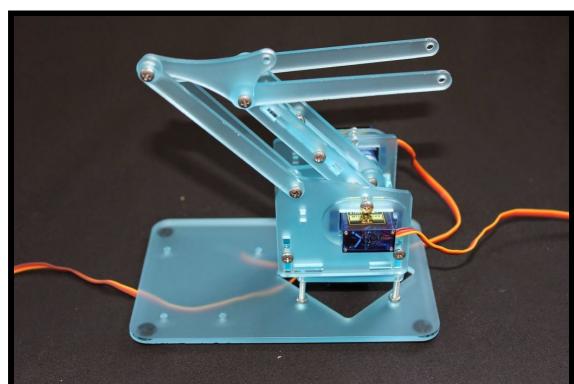
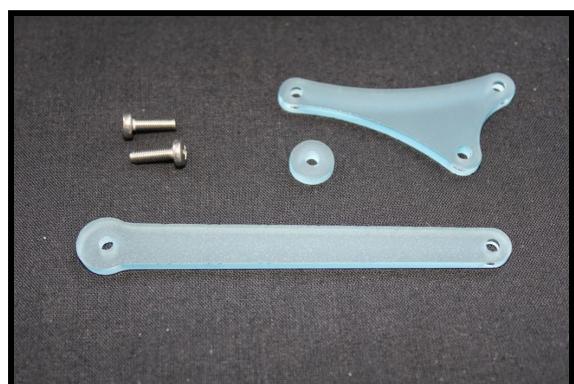
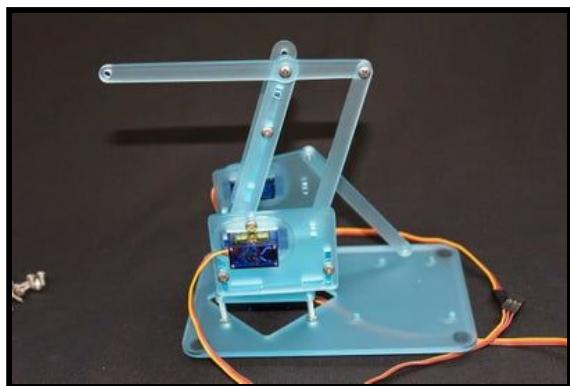
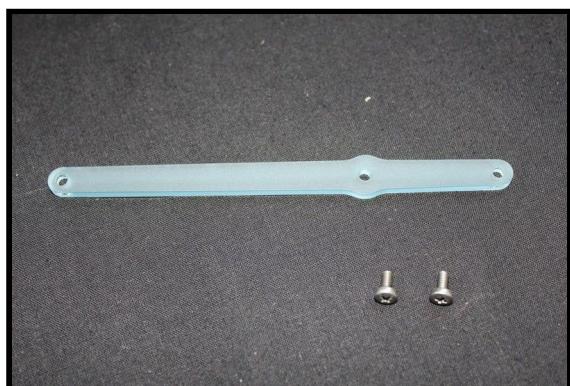
A esquerda é super simples. Uma peça com dois parafusos de 6 mm. Faça parecer a segunda imagem aqui!

O antebraço direito requer um espaço e você finalmente usa dois desses 10mm! Se você não tiver três parafusos de 10 mm neste momento, observe os de 12 mm que você acabou de usar, um ou

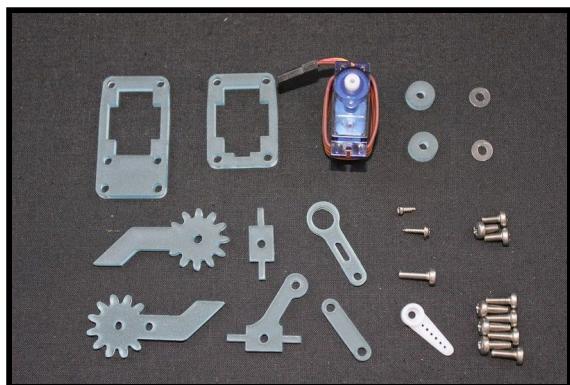
mais deles parecerão cerca de 2 mm mais curtos que os demais!

A conexão com a alavanca central à direita é: alavanca central, alavanca do antebraço, ponta triangular. Na parte de trás está um pouco triangular, espaçador, alavanca longa (a que foi anexada ao RHS anteriormente). Lembre-se de movimento fácil. Se necessário, trabalhe as alavancas algumas vezes. Agora, as coisas estão conectadas. Esse movimento vai lhe dar uma idéia de como tudo se encaixa.

Finalmente use a última alavanca longa com um parafuso de 6 mm na parte interna, como mostrado na última imagem aqui.

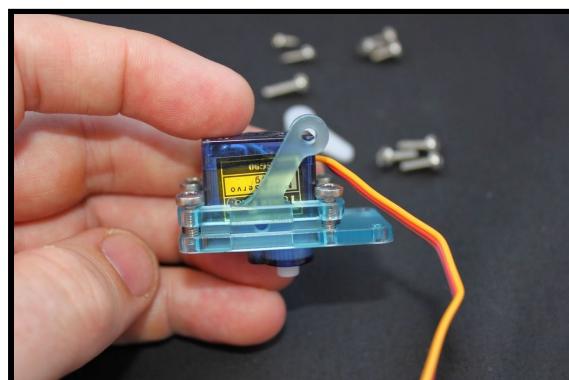
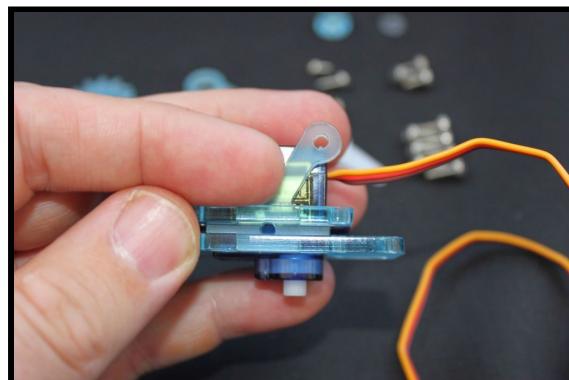
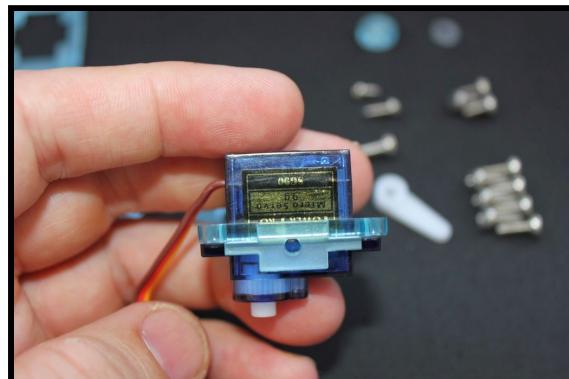
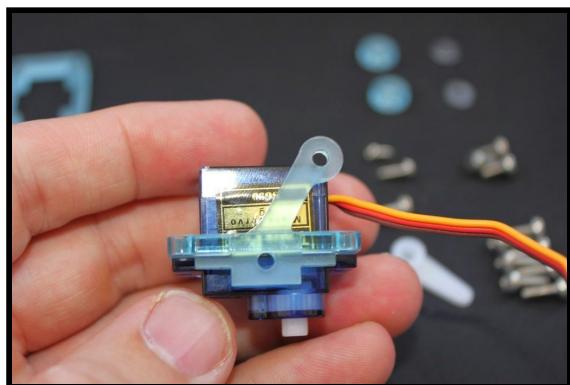


A garra...



Encontre a mais curta das duas partes retangulares. Este é um colar especializado! Enfie isso como se tivesse feito os outros três (ou quatro, se você quebrou um!). Em seguida, use as partes finas mostradas nas imagens, observe a orientação. Estes deslizam para o lado e agem como montagens. Eu acho que essa construção é muito inteligente e é o trabalho de Jack Howard, co-criador do #meArm.

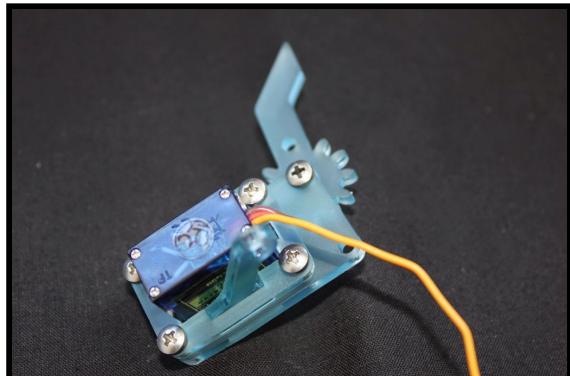
Agora, a parte retangular maior pode ser colocada sobre a parte inferior da peça composta que você acabou de criar. Verifique a orientação pela última vez, pegue os quatro parafusos restantes de 8 mm e não os aperte demais! Mas aperte-os. Certifique-se de que não sobressaia da base da peça que você acabou de fazer.



Pegue um parafuso de 6 mm e prenda a mandíbula dentada com dois orifícios no lado esquerdo da garra. Para obter melhores resultados, verifique se essas duas partes estão o mais niveladas possível, se você apertar o parafuso. Se você achar que sua pinça está saindo do plano, remova a peça, segure-a nivelada novamente e recorte a linha.

Alinhe a outra mandíbula para engrenar corretamente e prenda essa descarga com outro parafuso de 6 mm. Teste o movimento das mandíbulas agora. Se não for fácil, podem ser as juntas, como discutido anteriormente,

podem estar fora do plano (remover e recortar a rosca) ou os parafusos de 8 mm que usamos para a braçadeira estão apenas tocando a parte traseira das engrenagens , solte-os com um toque.



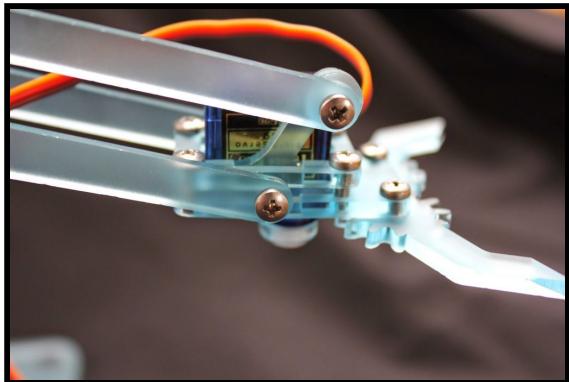
A seguir, faremos a ligação para conectar o servo e as garras. Prenda a buzina restante na alavanca de conexão do servo curto. Em seguida, conecte isso à pequena parte de ligação minúscula. Esse deveria ter sido o seu último parafuso de 6 mm.

Agora, com os últimos 12 mm, empurre-o através da pequena parte de ligação, adicione dois espaçadores e possivelmente as arruelas que incluímos e prenda ao orifício de reposição na mandíbula esquerda.

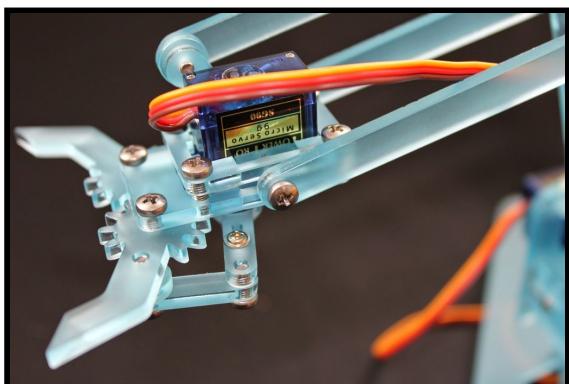


Costumo deixar isso desconectado do servo até que eu tenha controle por microcontrolador e possa decidir onde está a posição fechada.

Tudo o que resta é prender a garra ao resto do robô! Use dois parafusos de 8 mm no pivô da garra e os 10 mm finais com um espaçador para prender o pulso. Os parafusos de 8 mm normalmente entram em contato com a parte superior e inferior do grampo antes de encontrar o orifício nas peças espaçadoras. Você pode facilitar isso desapertando os parafusos no servo grampo levemente.



Agora é hora de conectá-lo ao seu controlador favorito! Uma vez conectado, você pode achar que precisa fazer ajustes finos na construção, provavelmente você apertou demais as peças e precisará dar um pouco de folga.



Dicas pós montagem

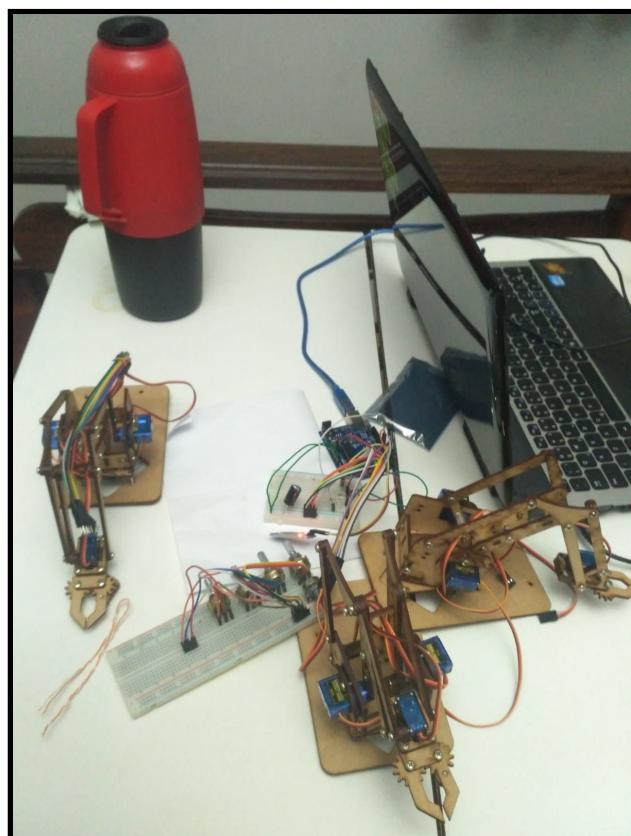
Parabéns, você acabou de montar seu braço robô! Use o código Arduino disponível em: <https://www.tinkercad.com/things/1KtkxZG6M7p> para realizar testes em seu braço.

Após essa sessão iremos detalhar um pouco de código Java e Arduino para que você possa controlar seu robô remotamente. Porém você pode reparar que alguns ajustes finos são necessários no que acabamos de construir.

Primeiro, pode ser necessário que você ajustar alguns parafusos, a maioria deles não deve ficar muito apertado, principalmente aqueles que estão em lugar onde há movimento. Cada força necessária para mover os braços será uma força a menos na hora de levantar objetos. Também é necessário colocar uns calços entre a base e o berço, dessa forma seu braço não irá tombar para frente e forçar o motor 1. Adicione também alguns a parte debaixo da base, fazendo assim com que ela fique nivelada.

Se você também cortou suas peças em MDF pode ter notado algumas rebarbas no corte, isso irá comprometer o bom funcionamento do projeto, retire-as e se necessário acrescente um pouco de talco nas juntas para que os movimentos sejam mais fluidos. Após usá-lo algumas vezes é notável que é necessário uma base para que o robô não caia para frente em movimentos mais longos. Para isso você pode anexar uma bateria e/ou placa de Arduino a base ou parafusá-la em uma base mais pesada.

Agora tome um café, descance um pouco e vamos para um pouco de programação.



Conectando seu braço robô ao controlador arduino

Recomendamos fortemente que você primeiramente utilize o circuito com potenciômetros¹, para que assim entenda o funcionamento dos motores e possa fazer os ajustes necessários. Com alguns ajustes todos os motores poderão ir de 0° - 180° sem nenhuma dificuldade. Agora vamos para a parte final do projeto. O código Arduino utilizando Bluetooth² é bem semelhante ao com potenciômetros.

Importe as bibliotecas necessárias; crie um objeto para representar seu socket bluetooth, não se esqueça de ligar as portas RX,TX invertidas; crie um buffer para armazenar as mensagens recebidas; defina os pinos dos motores servo, e crie um vetor de configuração para cada motor.

```
// inclui biblioteca do servomotor
#include <Servo.h>
#include <SoftwareSerial.h>

//Define as portas do bluetooth
SoftwareSerial BT(8, 9); // RX, TX

// Stores response of bluetooth device
char cmd;
String command = "";

//define pinos dos servos
#define pinServ1 2
#define pinServ2 3
#define pinServ3 4
#define pinServ4 5

//pos_inicial, min, max
int lim_m1[]={90,0,180};
int lim_m2[]={60,0,180};
int lim_m3[]={70,0,180};
int lim_m4[]={110,0,180};
```

Crie funções para separar as strings recebidas do celular, neste código iremos adotar o seguinte padrão: Será enviada uma string via bluetooth para o arduino com o caractere delimitador, fim de linha, '\n'. Esta string irá conter o valor a ser somado ou subtraído pelo motor em sua posição atual e um código que indicará qual é o motor e qual a operação a ser realizada, separados pelo caractere '#' e seguindo a seguinte formatação: 'codigo#valor\n'.

```

//Obtém o numero do motor na string recebida, numero antes do '#'
int get_codigo(String string){
    String n = "";
    for(int i = 0;string[i] != '#'; i++){
        n += string[i];
    }
    return n.toInt();
}

//Obtém, o valor da string recebida, numero após '#'
long get_valor(String string){
    String n = "";
    int i = 0;
    for(; string[i] != '#'; i++){
        continue;
    }
    for(i++; i < string.length(); i++){
        n += string[i];
    }
    return n.toInt();
}

```

Defina mais algumas variáveis auxiliares, em seguida no método setup inicie as duas comunicações seriais; faça as atribuições iniciais e posicione seu robô.

```

// nomeia os servos
Servo serv1,serv2,serv3,serv4;

// cria as variavies dos angulos de cada motor
int motor1=0,motor2=0,motor3=0,motor4=0;

//Inicia o timer do mostrador e define seu intervalo
unsigned long mostradorTimer = 1;
const unsigned long intervaloMostrador = 2000;

```

```

void setup() {
    // HC-06 usually default baud-rate
    BT.begin(9600);
    //inicia o monitor serial
    Serial.begin(9600);
    // atribui pinos dos servos
    serv1.attach(pinServ1);
    serv2.attach(pinServ2);
    serv3.attach(pinServ3);
    serv4.attach(pinServ4);
    //Volta o robô a uma posição inicial.
    motor1=lim_m1[0];
    motor2=lim_m2[0];
    motor3=lim_m3[0];
    motor4=lim_m4[0];
    serv1.write(motor1);
    delay(500);
    serv2.write(motor2);
    delay(500);
    serv3.write(motor3);
    delay(500);
    serv4.write(motor4)
    delay(1600);
}

```

No método loop , verifique se há uma mensagem e enquanto houver mensagem disponível armazena no buffer, logo após separe o código do valor e realize uma escolha para determinar qual robô sofrerá alteração no valor de sua posição. Notamos que ao reescrever o valor de todos os motores, mesmo que eles não tenham sido alterados tornava os movimentos do robô mais estáveis.

```

void loop(){
    if (BT.available()) {
        while(BT.available()){
            delay(10);
            char c = BT.read();
            command += c;
        }
        Serial.println("COMANDO-->" + command);

        int codigo = get_codigo(command);
        int valor = get_valor(command);

        int v = 0;
        switch(codigo){
            case 11:
                v = motor2 + valor;
                motor2 = (v > lim_m2[2])?lim_m2[2] : v;
                Serial.println("2GT"+motor2);
                break;
            case 12:
                v = motor1 - valor;
                motor1 = (v < lim_m1[1])?lim_m1[1] : v;
                Serial.println("1GT"+motor1);
                break;
        }
    }
}

```

```

        case 24:
            v = motor4 - valor;
            motor4 = ( v < lim_m4[1])?lim_m4[1] : v;
            Serial.println("4GT"+motor4);
            break;
        }

        serv1.write(motor1);
        serv2.write(motor2);
        serv3.write(motor3);
        serv4.write(motor4);
        command = "";

    } //if (BT.available())

```

Em seguida verifique o tempo do mostrador e caso o intervalo tenha sido ultrapassado exiba a posição dos motores e aguarde um pouco para evitar possíveis falhas e encerre o método loop. Pronto seu receptor bluetooth já está configurado, passemos ao controle remoto.

```

//Envio para o monitor serial do posicionamentos dos motores
if ((millis() - mostradorTimer) >= intervaloMostrador) {
    Serial.println("*****");
    Serial.print(" Angulo Motor1:");
    Serial.println(serv1.read());
    Serial.print(" Angulo Motor2:");
    Serial.println(serv2.read());
    Serial.print(" Angulo Motor3:");
    Serial.println(serv3.read());
    Serial.print(" Angulo Motor4:");
    Serial.println(serv4.read());

    mostradorTimer = millis();
}

// tempo de espera para recomeçar
delay(100);

} //void loop()

```

¹ → <https://www.tinkercad.com/things/1KtkxZG6M7p>

² → <https://www.tinkercad.com/things/q6V66FnFYDw>

Conectando seu controlador ao controle remoto Android

Utilizando a classe Java que será apresentada abaixo você será capaz de criar qualquer tipo de controlador via bluetooth para seu braço robótico. A classe em questão

consiste em uma thread para gerenciar 2 sockets de conexão bluetooth e está habilitada a criar uma conexão, enviar e receber mensagens de texto de maneira serial.

Estendemos a classe Thread e adicionamos o seguintes atributos android.bluetooth.BluetoothSocket e BluetoothServerSocket para realizarmos a comunicação com o código arduino já apresentado anteriormente. Também é importante usarmos um myUUID que será enviado como identificação ao dispositivo HC-06(Arduino). Utilizaremos de 2 construtores para identificar quando a thread será cliente ou servidora. No exemplo de uso que será demonstrado posteriormente utilizaremos o construtor cliente.

```
public class ConnectionThread extends Thread{

    BluetoothSocket btSocket = null;
    BluetoothServerSocket btServerSocket = null;
    InputStream input = null;
    OutputStream output = null;
    String btDevAddress = null;
    String myUUID = "00001101-0000-1000-8000-00805F9B34FB";
    boolean server;
    boolean running = false;
    boolean isConnected = false;

    /* Este construtor prepara o dispositivo para atuar como servidor.
     */
    public ConnectionThread() {
        this.server = true;
    }

    /* Este construtor prepara o dispositivo para atuar como cliente.
       Tem como argumento uma string contendo o endereço MAC do dispositivo
       Bluetooth para o qual deve ser solicitada uma conexão.
     */
    public ConnectionThread(String btDevAddress) {

        this.server = false;
        this.btDevAddress = btDevAddress;
    }
}
```

- Antes de te apresentar o método run() que irá conter toda a regra de comunicação irei te apresentar os métodos utilitários que serão utilizados.

```

/*
 * Utiliza um handler para enviar um byte array à Activity principal.
 * O byte array é encapsulado em um Bundle e posteriormente em uma Message
 * antes de ser enviado.
 */
private void toMainActivity(byte[] data) {

    Message message = new Message();
    Bundle bundle = new Bundle();
    bundle.putByteArray("data", data);
    message.setData(bundle);
    MainActivity.handler.sendMessage(message);
}

/*
 * Método utilizado pela Activity principal para transmitir uma mensagem ao
 * outro lado da conexão.
 * A mensagem deve ser representada por um byte array.
 */
public void write(byte[] data) throws IOException {

    if(output != null) {
        output.write(data);
    } else {

        /*
         * Envia à Activity principal um código de erro durante a conexão.
         */
        toMainActivity("----N".getBytes());
    }
}

```

```

/*
 * Método utilizado pela Activity principal para encerrar a conexão
 */
public void cancel() {

    try {

        running = false;
        this.isConnected = false;
        btServerSocket.close();
        btSocket.close();

    } catch (IOException | NullPointerException e) {
        e.printStackTrace();
    }
    running = false;
    this.isConnected = false;
}

//Informa se o bluetooth está conectado
public boolean isConnected() {
    return this.isConnected;
}

```

- Dessa maneira o método run() sobrescrito será dividido em 2 partes, por meio da estrutura de seleção *if*, se a variável *server* conter o valor *true* a thread irá se comportar como servidor da seguinte maneira:

```

/*
 * O método run() contém as instruções que serão efetivamente realizadas
 * em uma nova thread.
 */
public void run() {

    /*
     * Anuncia que a thread está sendo executada.
     * Pega uma referência para o adaptador Bluetooth padrão.
     */
    this.running = true;
    BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();

    /*
     * Determina que ações executar dependendo se a thread está configurada
     * para atuar como servidor ou cliente.
     */
    if(this.server) {

        /*
         * Servidor.
         */
        try {

            /*
             * Cria um socket de servidor Bluetooth.
             * O socket servidor será usado apenas para iniciar a conexão.
             * Permanece em estado de espera até que algum cliente
             * estabeleça uma conexão.
             */
            btServerSocket = btAdapter.listenUsingRfcommWithServiceRecord( name: "MyArmControl",
                UUID.fromString(myUUID));
            btSocket = btServerSocket.accept();

            /*
             * Se a conexão foi estabelecida corretamente, o socket
             * servidor pode ser liberado.
             */
            if(btSocket != null) {
                btServerSocket.close();
            }
        }
        catch (IOException e) {

            /*
             * Caso ocorra alguma exceção, exibe o stack trace para debug.
             * Envia um código para a Activity principal, informando que
             * a conexão falhou.
             */
            e.printStackTrace();
            toMainActivity("----N".getBytes());
        }
    } else {
}
}

```

- Senão a thread irá atuar como cliente:

```

} else { /*Cliente*/
    try {
        /* Obtem uma representação do dispositivo Bluetooth com
        endereço btDevAddress.
        Cria um socket Bluetooth.
        */
        BluetoothDevice btDevice = btAdapter.getRemoteDevice(btDevAddress);
        btSocket = btDevice.createRfcommSocketToServiceRecord(UUID.fromString(myUUID));

        /* Envia ao sistema um comando para cancelar qualquer processo
        de descoberta em execução.
        */
        btAdapter.cancelDiscovery();

        /* Solicita uma conexão ao dispositivo cujo endereço é
        btDevAddress.
        Permanece em estado de espera até que a conexão seja
        estabelecida.
        */
        if (btSocket != null) {
            btSocket.connect();
        }
    } catch (IOException e) {
        /* Caso ocorra alguma exceção, exibe o stack trace para debug.
        Envia um código para a Activity principal, informando que
        a conexão falhou.
        */
        e.printStackTrace();
        toMainActivity("----N".getBytes());
    }
}
}

```

- Agora basta gerenciar a conexão:

```

/*
  Pronto, estamos conectados! Agora, só precisamos gerenciar a conexão.*/
if(btSocket != null) {
    //Envia um código para a Activity principal informando que a conexão ocorreu com sucesso
    this.isConnected = true;
    toMainActivity("---S".getBytes());
    try {
        /* Obtem referências para os fluxos de entrada e saída do socket Bluetooth.*/
        input = btSocket.getInputStream();
        output = btSocket.getOutputStream();

        /* Permanece em estado de espera até que uma mensagem seja recebida. Armazena a
        mensagem recebida no buffer. Envia a mensagem recebida para a Activity principal,
        do primeiro ao último byte lido. Esta thread permanecerá em estado de escuta até
        que a variável running assuma o valor false. */
        while(running) {
            /* Cria um byte array para armazenar temporariamente uma mensagem recebida.
            O inteiro bytes representará o número de bytes lidos na última transmissão
            recebida. O inteiro bytesRead representa o número total de bytes lidos antes
            de uma quebra de linha. A quebra de linha representa o fim da mensagem. */
            byte[] buffer = new byte[1024];
            int bytes;
            int bytesRead = -1;

            /* Lê os bytes recebidos e os armazena no buffer até que uma quebra de linha
            seja identificada. Nesse ponto, assumimos que a mensagem foi transmitida
            por completo. */
            do {
                bytes = input.read(buffer,  off: bytesRead+1,  len: 1);
                bytesRead+=bytes;
            } while(buffer[bytesRead] != '\n');

            //A mensagem recebida é enviada para a Activity principal.
            toMainActivity(Arrays.copyOfRange(buffer,  from: 0,  bytesRead));
        }
    }
}

```

- Em caso de erro encerre a conexão e avise a activity principal:

```

        } catch (IOException e) {
            /*Caso ocorra alguma exceção, exibe o stack trace para debug. Envia um código
            para a Activity principal, informando que a conexão falhou. */
            e.printStackTrace();
            toMainActivity("---N".getBytes());
            this.isConnected = false;
        }
    }
} // Fim do método run()

```

- Iniciando uma conexão no método oncreate da activity principal:

```

private Intent enableBtIntent;

@SuppressLint("ClickableViewAccessibility")
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    context = this;
    BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();
    if(btAdapter == null){
        Toast.makeText(context, "BT não está funcionando", Toast.LENGTH_SHORT).show();
    }else{
        Toast.makeText(context, "BT Funcionando", Toast.LENGTH_SHORT).show();
    }
    //Verifica se o bluetooth está ligado, se não estiver abre um alerta pedindo a ativação.
    enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    if (!btAdapter.isEnabled()) {
        startActivityForResult(enableBtIntent, requestCode: 19);
    }else{
        onActivityResult(requestCode: 19,RESULT_OK, enableBtIntent);
    }
}

```

- Crie um Handler para tratar as mensagens enviadas pela thread de conexão:

```

@SuppressLint("HandlerLeak")
public static Handler handler = handleMessage(msg) -> {
    /* Esse método é invocado na Activity principal sempre que a thread de
       conexão Bluetooth recebe uma mensagem. */
    Bundle bundle = msg.getData();
    byte[] data = bundle.getByteArray(key: "data");
    String dataString= new String(data);

    /* Aqui ocorre a decisão de ação, baseada na string recebida. Caso a string
       corresponda à uma das mensagens de status de conexão (iniciadas com --),
       atualizamos o status da conexão conforme o código. */
    if(dataString.equals("---N")){
        Toast.makeText(context , text: "Ocorreu um erro durante a conexão",
                      Toast.LENGTH_SHORT).show();
    }else if(dataString.equals("---S")){
        Toast.makeText(context , text: "Conectado", Toast.LENGTH_SHORT).show();
    }else {
        /* Se a mensagem não for um código de status, então ela deve ser tratada
           pelo aplicativo como uma mensagem vinda diretamente do outro lado da conexão.
           Nesse caso, simplesmente atualizamos o valor contido no TextView do contador.*/
        //counterMessage.setText(dataString);
        Toast.makeText(context , dataString, Toast.LENGTH_SHORT).show();
        Log.e(tag: "MSG", dataString);
    }
}

```

- Por fim, crie um onActivityResult para tratar o resultado da intenção de ligar o bluetooth:

```

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    //Encerra o aplicativo caso o usuário não tenha ligado o Bluetooth.
    if(resultCode == RESULT_CANCELED) finish();

    BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();

    //Buscando uma lista de dispositivos pareados com o celular
    Set<BluetoothDevice> pairedDevices = btAdapter.getBondedDevices();
    if (pairedDevices.size() > 0) {
        for (BluetoothDevice device : pairedDevices) {
            String deviceName = device.getName();
            String deviceHardwareAddress = device.getAddress(); // MAC address
            //Para quando encontrar dispositivo com nome HC-06
            if(deviceName.equals("HC-06")){
                MAC_BT = deviceHardwareAddress;
                break;
            }
        }
    }else{
        Toast.makeText( context: this, text: "Nenhum dispositivo pareado",Toast.LENGTH_SHORT).show();
    }
}

```

- De posse do mac do dispositivo bluetooth, inicie uma conexão:

```

if(MAC_BT.isEmpty()){
    Toast.makeText( context: this, text: "Pareie seu MyArm e reinicie o aplicativo!",
                    Toast.LENGTH_SHORT).show();
    return;
}
connect = new ConnectionThread(MAC_BT);
connect.start();
try { Thread.sleep( millis: 100); } catch (InterruptedException e) { e.printStackTrace(); }

// onActivityResult

```

Não se esqueça que sua activity deverá conter a variável *ConnectionThread connect* nem de dar uma pausa após a conexão estabelecida para que não ocorram bugs estranhos. A partir de agora você já é capaz de chamar o método *write* de sua variável *connect* para enviar comandos ao código Arduino já criado.

Considerações finais

Desejamos novamente agradecer a Iberê Thenório do [Manual do Mundo](#) por ter nos apresentado o projeto original e ser de grande ajuda mesmo que ele não tenha conhecimento disso. E esperamos que você tenha concluído o manual com o, dúvidas e sugestões favor entrar em contato conosco através do Github ou email