ENG1450

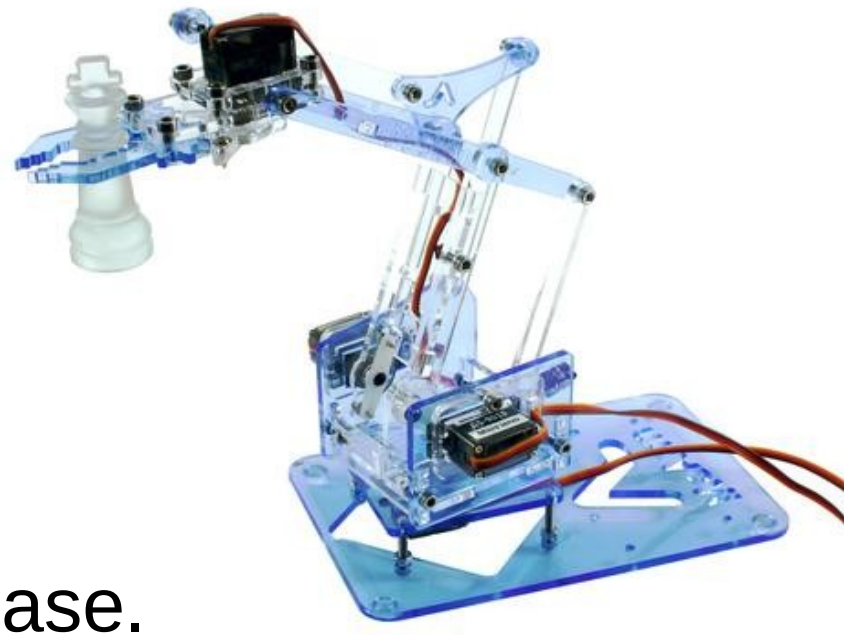
Microcontroladores e Sistemas Embarcados

Introdução para a tarefa do

Braço Robótico

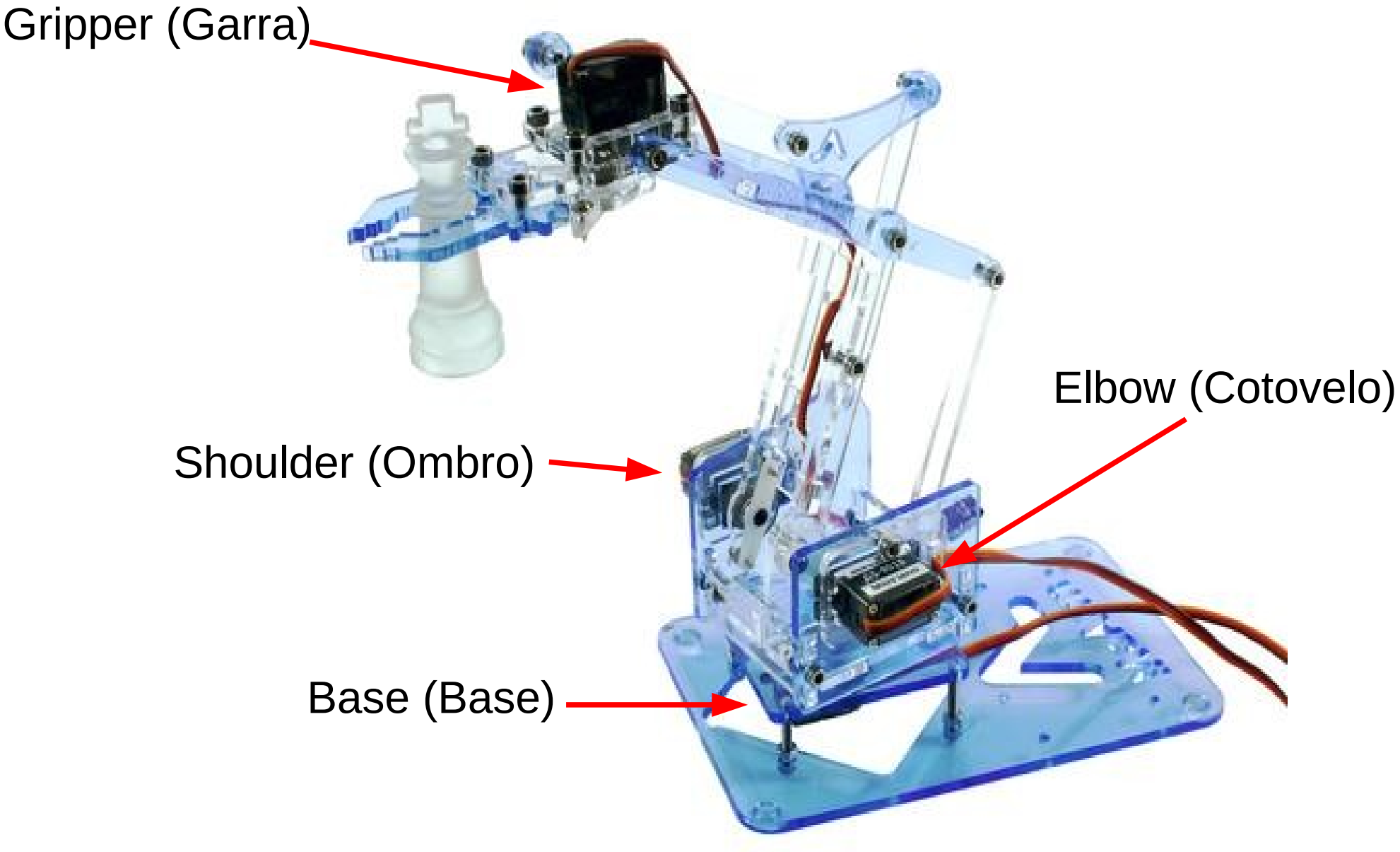
Autor: Adriano Branco
Prof.: Moisés H. Szwarcman

Cuidado!!!!!



- **Muito frágil.**
- Não puxar pelo braço.
- Manusear com muita atenção.
- Sempre pegar e carregar pela base.
- Alimentar somente com a fonte externa de 5V.
- Observar os limites de movimento de cada servo.
- Cuidado com o movimento vertical para não forçar o braço abaixo do nível da base.
- Cuidado com os movimentos repentinos para não acertar algum obstaculo, incluindo você.

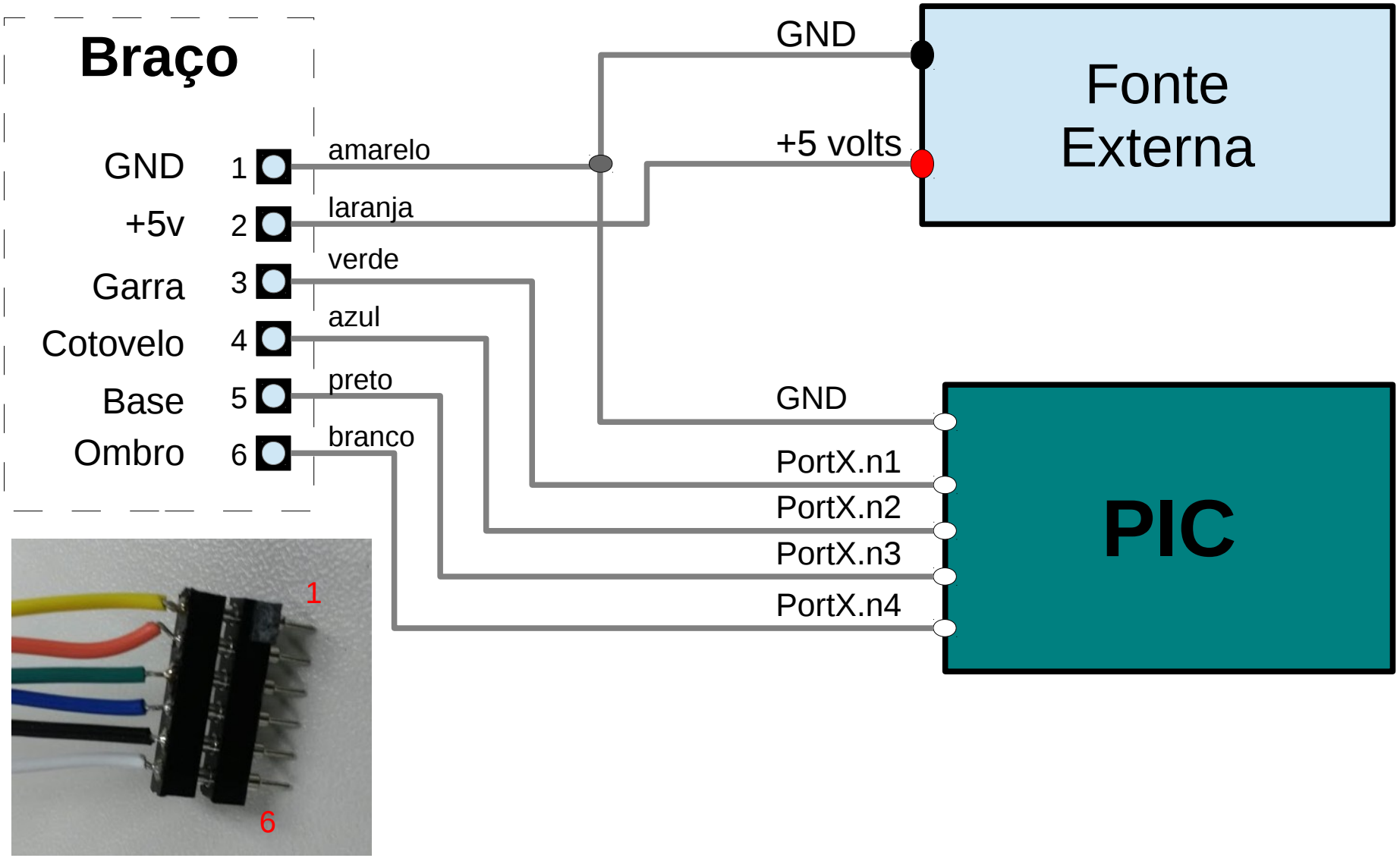
MeArm© – Servos



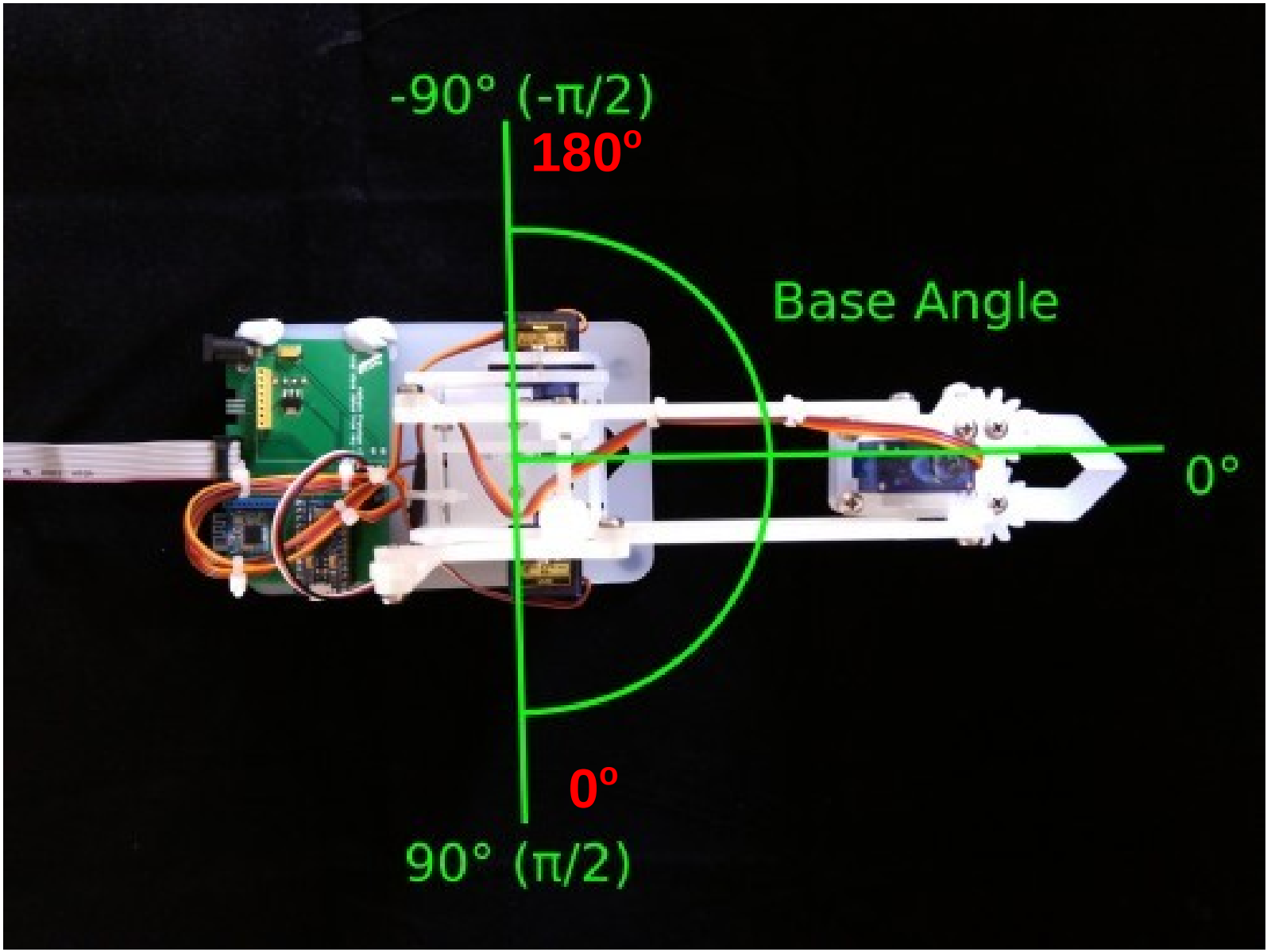
Servo SG-90

- Movimento de 0º a 180º
- Controlado por PWM de 50Hz.
- Pulso mínimo de 544µs e máximo de 2400µs
- Utilizar biblioteca adaptada meArm
  - Permite até 4 servos.
  - Usa o Timer1 do PIC. Cuidado com o seu código.
  - Baixar da página:
    - http://www.inf.puc-rio.br/~abranco/eng1450/meArm.zip
    - Descompactar no diretório do seu projeto.
    - Insira o caminho da nova pasta na configuração de “Search Paths” do seu projeto no MikroC.
  - Instruções de uso mais a frente.

Conexões do Braço

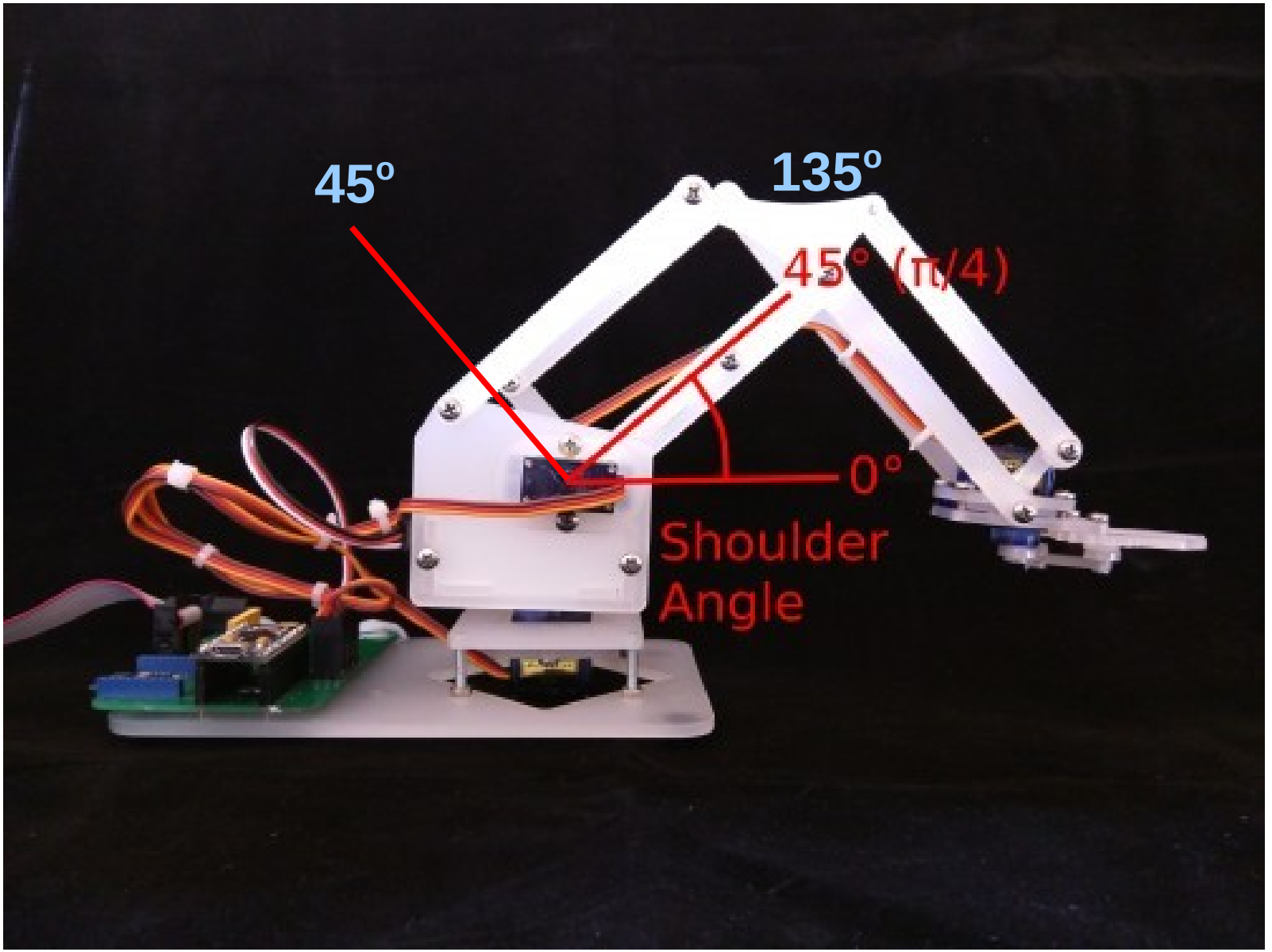


Movimento da Base – 0º..180º



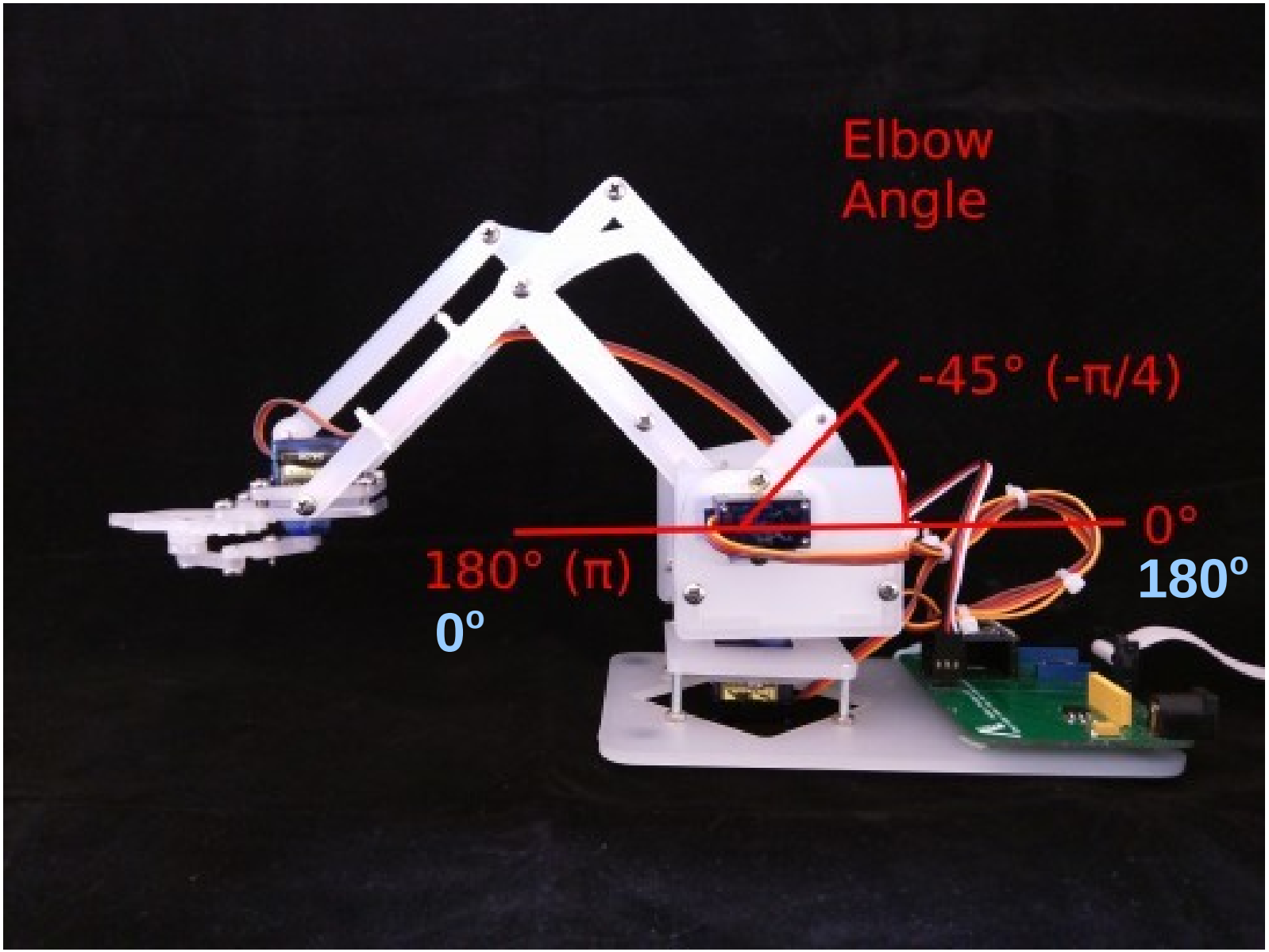
Crédito da imagem: Bit Of A Hack (http://bitofahack.com/post/1433701488)

“Ombro” - 45º .. 135º



Crédito da imagem: Bit Of A Hack (http://bitofahack.com/post/1433701488)

“Cotovelo” - 45º .. 180º



Crédito da imagem: Bit Of A Hack (http://bitofahack.com/post/1433701488)

Tarefas

1. Controle dos ângulos dos servomotores
   - Posiciona o braço “estimando” o ângulo de cada servo.

2. Controle da posição da garra no espaço cartesiano
   - Utiliza uma biblioteca de “Cinemática Inversa” para calcular o ângulo de cada servo a partir de um ponto no espaço (x,y,z).

Biblioteca Servo – ângulos

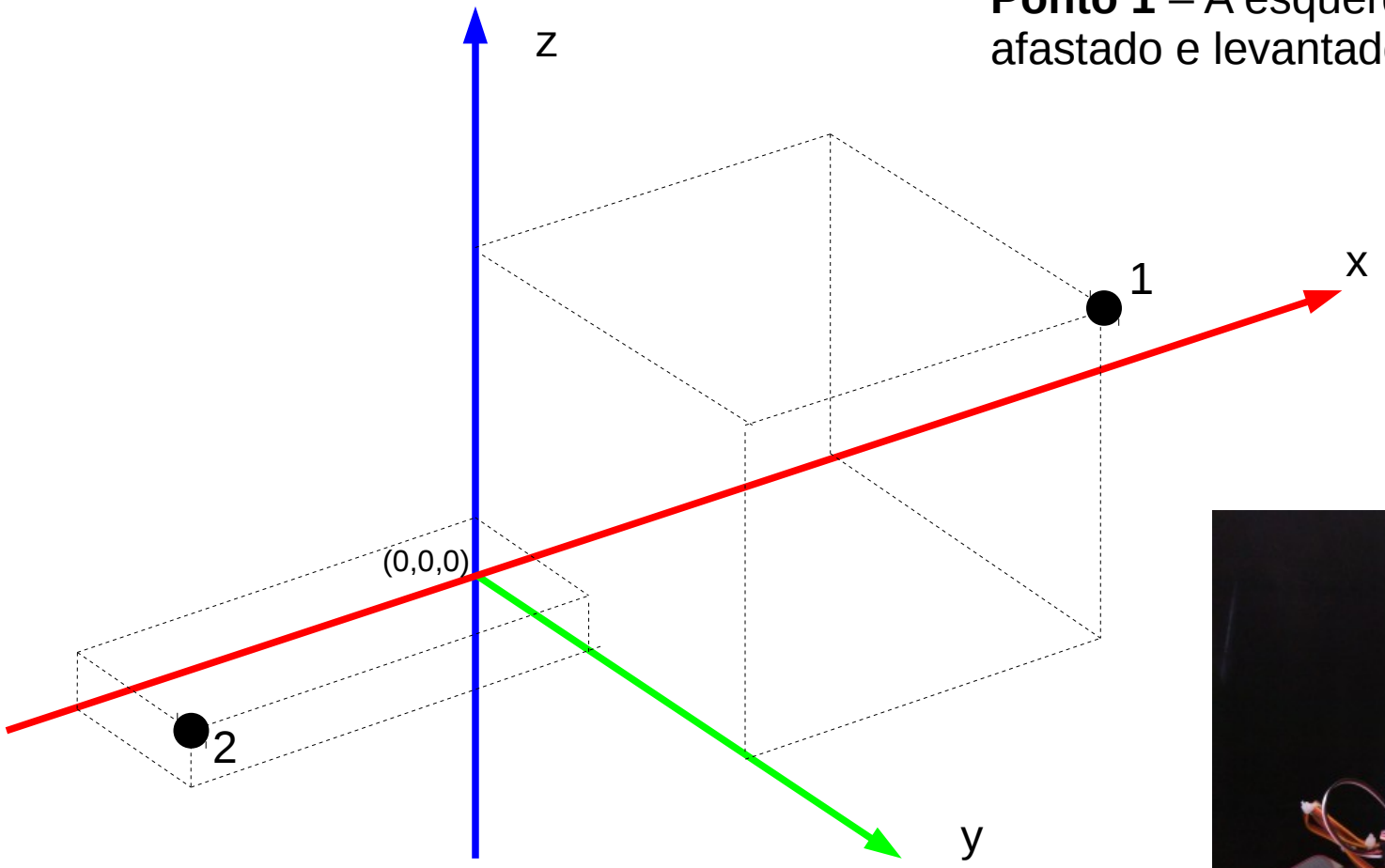
(Ainda em versão beta)

- Incluir a biblioteca
  - #include "servo.h"
  - Adicionar os arquivos servo.h e servo.c e a lib C_Math
- Inicializar a biblioteca
  - ServoInit();
- Inicializa cada servo
  - ServoAttach(char id, char port, char pin);
    - Id=0..3, port = &PORTB, pin=0..7
- Posicionar os servos
  - ServoWrite(char id, float Angle);
    - Angle: 0 .. 180

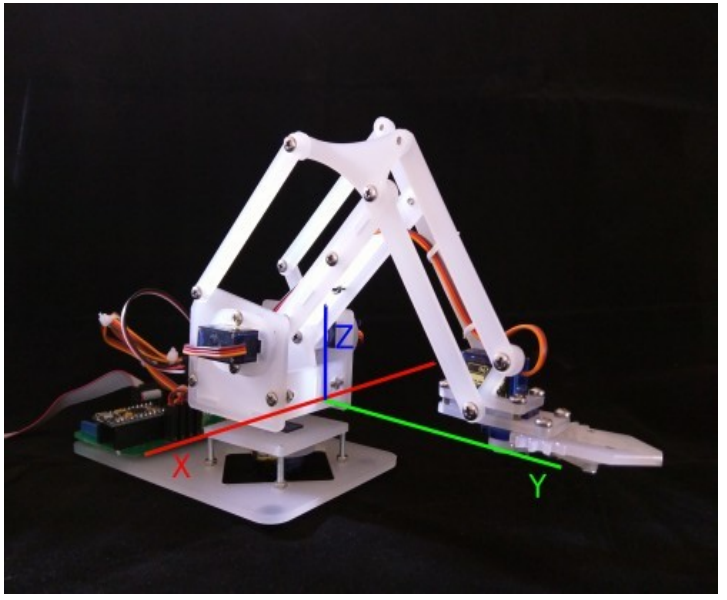
Tarefa 4 - Parte 1

Posicionamento do braço em dois pontos

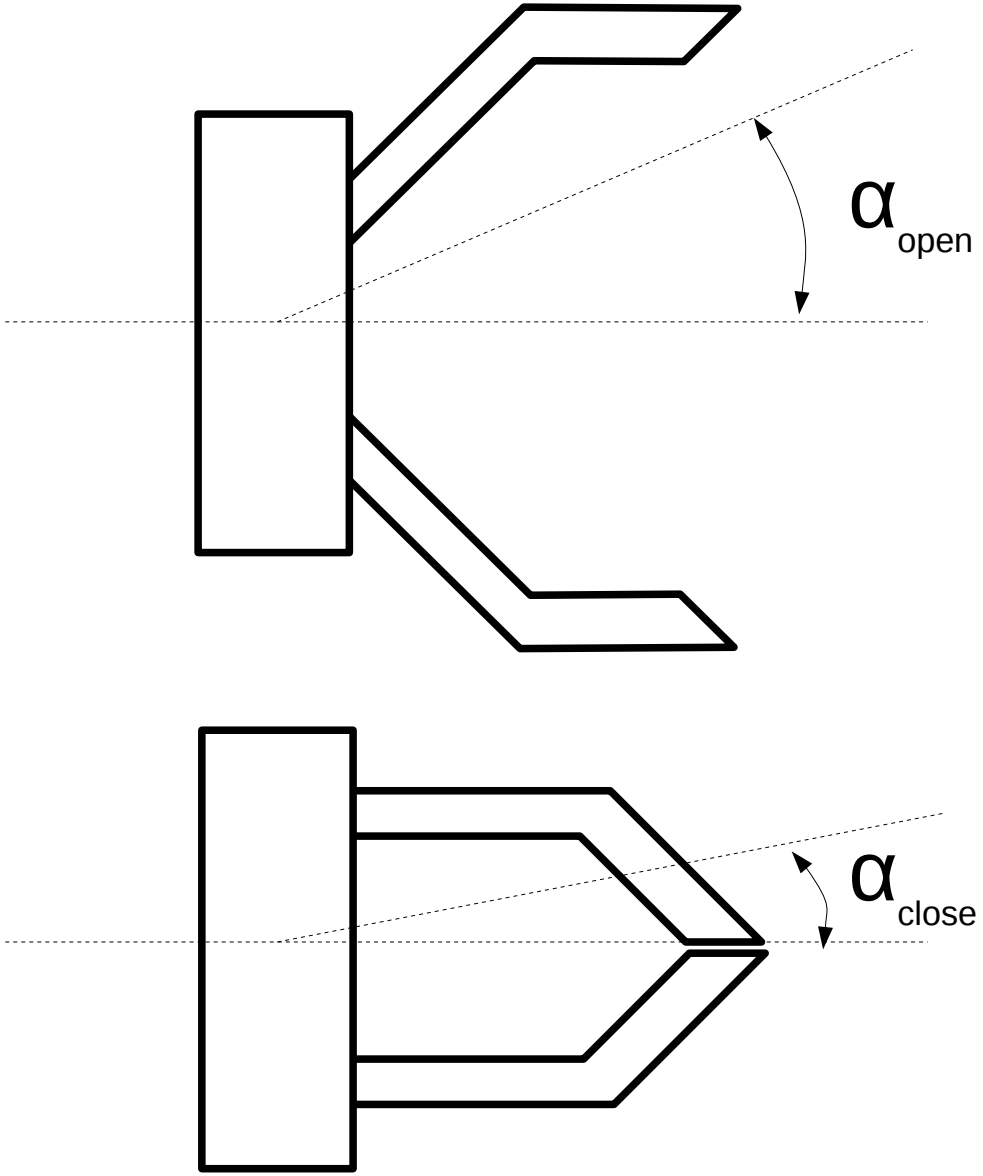
Ponto 1 – A esquerda do braço, a frente afastado e levantado.



Ponto 2 – A direita do braço, a frente próximo e abaixado.



Posicionamento da Garra



Braço	α_{open}	α_{close}
1	80º	45º
2	80º	62º
3	110º	70º
4	90º`	65º
5	75º	50º
6	80º	56º
7	80º	55º
8	70º	35º
9	130º	103º
10	20º	5º

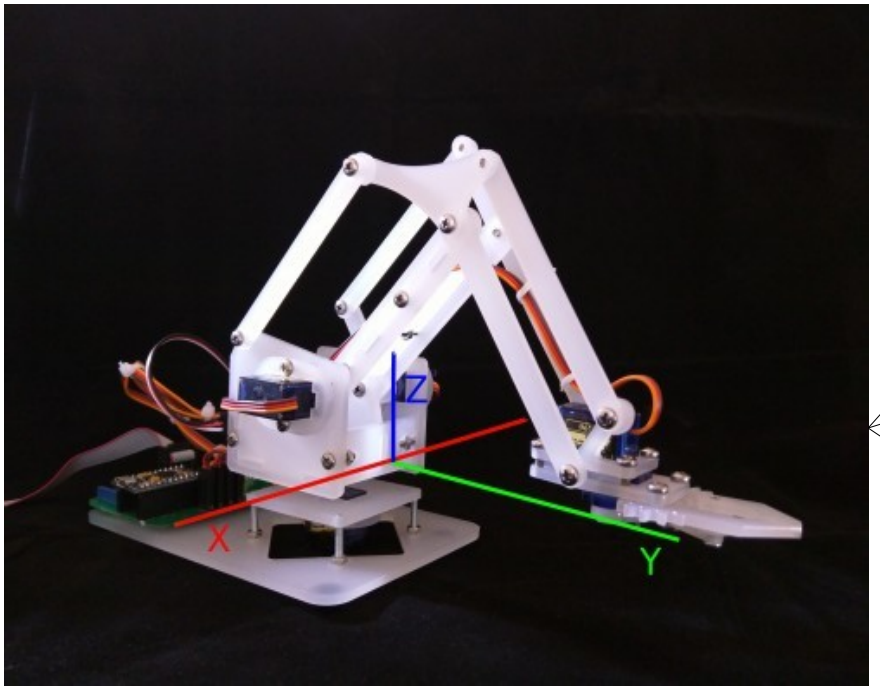
Cinemática Inversa

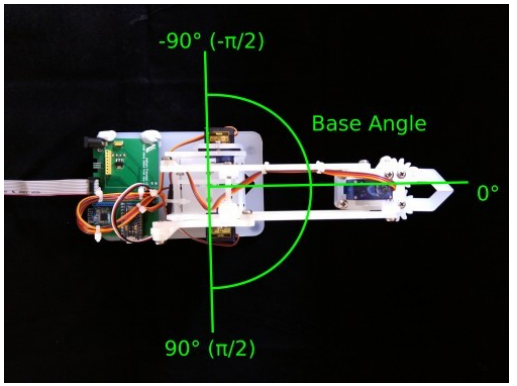
IK - Inverse Kinematics

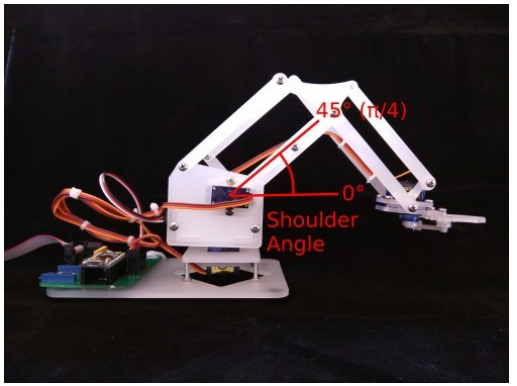
Cinemática Inversa

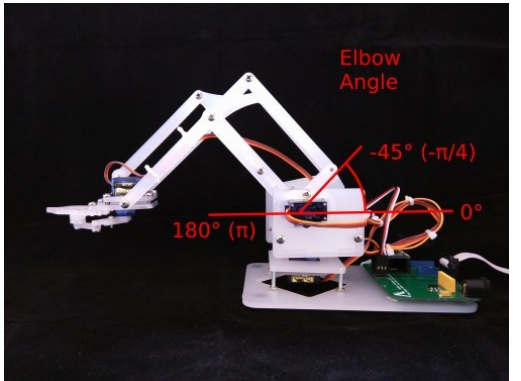
IK - Inverse Kinematics

(X,Y,Z) => { Angulo Base
Angulo Ombro
Angulo Cotovelo









As coordenadas são medidas em mm a partir do centro de rotação da base. A posição inicial é (0, 130, 50), isto é, 130 mm à frente da base e 50 mm do chão.

Biblioteca meArm/IK – x, y e z
(Ainda em versão beta)

- Incluir as bibliotecas e definir o ID do braço utilizado
  - #include "meArm.h"
  - #define ARM_ID 6
  - #include "armData.h"
  - Adicionar ao projeto os outros arquivos baixados.
- Inicializar a biblioteca
  - meArm_calib(armData);
    - A variável “armData” já é definida dentro armData.h
  - meArm_begin(char port, int pinBase, int pinOmbro, int pinCotovelo, int pinGarra);
    - port= &PORTB, pin*=0..7
- Ações
  - meArm_openGripper() e meArm_closeGripper()
  - meArm_gotoPoint(x,y,z); // Suavemente
  - meArm_goDirectlyTo(x,y,z);
  - meArm_servo(id,angle);

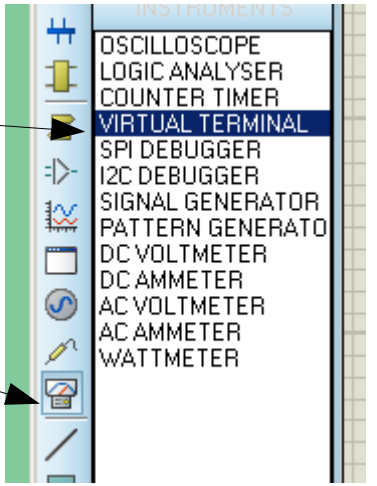
Curiosidades sobre calibração

- As características do projeto do braço, dos servos utilizados e da montagem não permitem uma precisão muito grande para determinar as coordenadas x, y e z.
- Precisamos da informação da posição real (ângulos de referência) de cada servo para poder estimar com mais precisão a movimentação correta do braço. Esses valores variam de um braço para outro.
- A biblioteca disponibilizada já inclui os dados de calibração para os braços montados para o nosso experimento. Bastando que o usuário identifique qual é o braço utilizado.
- Se necessário, os dados de calibração devem ser corrigidos no arquivo armData.h.

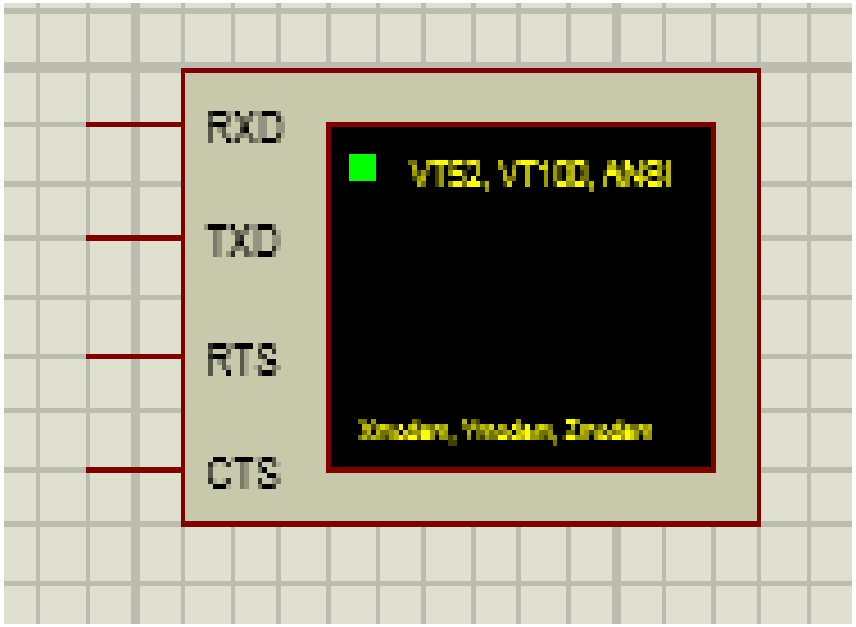
Simulação no Proteus
Uart/Serial e Servo-motor

Serial no Proteus

- Utilizar o “Virtual Terminal”
  - ícone de instrumentos
  - Baudrate 57600
  - Durante a simulação ativar “Echo Typed Characters”



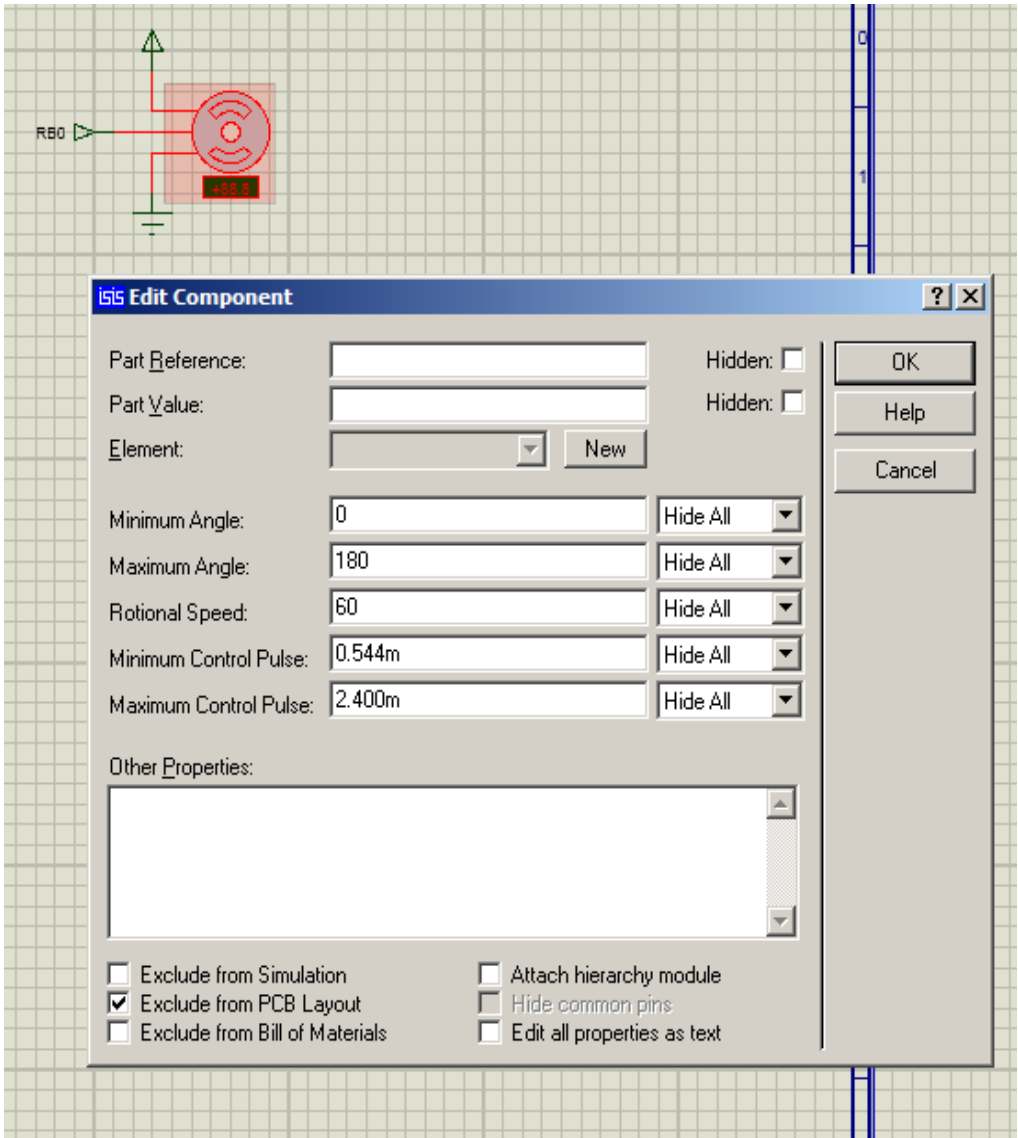
- Conectar
  - TXD no RX/RC7
  - RXD no TX/RC6



- Usar a UART1

Servo no Proteus

- Componente:
  - MOTOR-PWMSERVO
- Configurar componente:
  - Minimum Angle: 0
  - Maximum Angle: 180
  - Minimum Control Pulse: 0.544m
  - Maximum Control Pulse: 2.400m



Protocolo sugerido

Ann.nn**,**nn.nn**,**nn.nn**,**nn.nn**;**

- **A** - Uma letra indicando o código da Ação
- nn.nn - Um número ponto flutuante (float)
- **,** - Separadores dos números
- **;** - Indicador de final de comando;
- OBS:
  - O comando pode ter de 1 até 4 números.
  - A casa decimal não é obrigatória.

Parser simples em C

http://www.inf.puc-rio.br/~abranco/eng1450/meArm_PIC/TestParser.c

```
// Estrutura de dados do comando
typedef struct command{
    char action;
    char len;
    float num[4];
} command_t;

// Função de decodificação
void decodeAction(char* text, command_t* command){
    int i=0;
    char *token;
    command->action = text[0];              // Atualiza o caractér da ação
    for (i=0;i<4;i++) command->num[i]=0; // Zera valores
    i=0;
    token = strtok(&text[1],",");           // Busca 1º valor
    while(token && i < 4) {
        command->num[i++]=atof(token);  // Converte para float
        token = strtok(0, ",");             // Busca próximo valor
    }
    command->len = i;                       // Atualiza quantidade convertida
 }
```

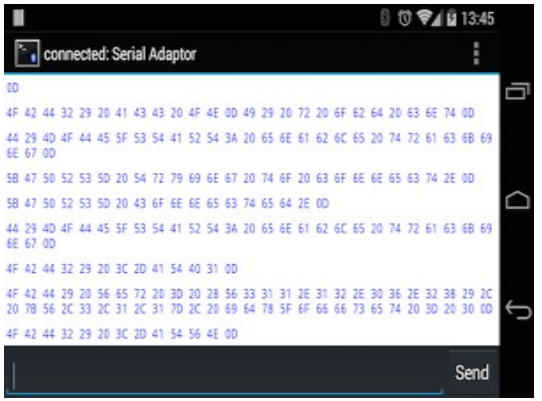
Exemplo de uso

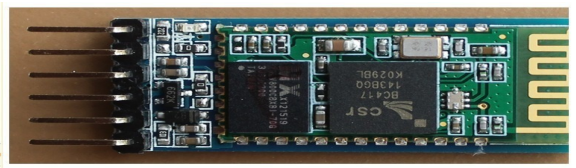
```
// Função definida pelo usuário
void executeAction(command_t* command){
    switch (command->action){
        case 'A': ……; break;
    }
}

void main() {
    // initialize UART1 module
    UART1_Init(56700);
    Delay_ms(100);

    while (1) {
      command_t command;
      char textCommand[50];
      if (UART1_Data_Ready() == 1) {                    // if data is received
          UART1_Read_Text(textCommand, ";", 30);        // reads text until ';' is found
          decodeAction(textCommand,&command);           // decode text
          executeAction(&command);                      // user function
     }
  }
}
```

Conexão
Bluetooth + PIC

Bluetooth

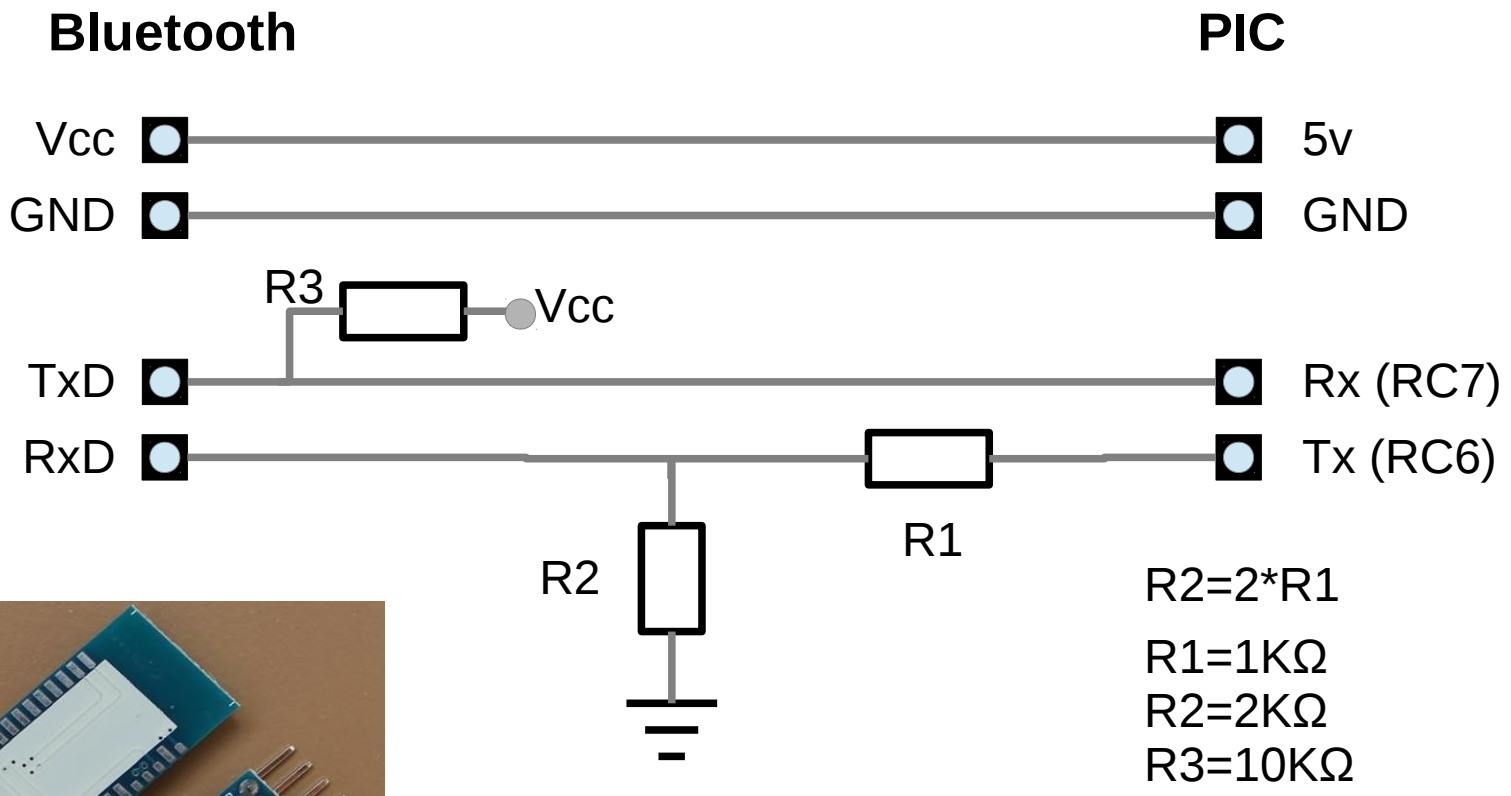
- Interface serial do PIC
  - Biblioteca UART do MikroC
  - Baudrate 9600
- Smartphone
  - Qualquer Terminal Bluetooth
- Módulo Bluetooth
  - Serial TTL
  - Nível do sinal 3,3 volts
  - Alimentação: 3,6v – 6v
- Teste
  - Fazer um programa que redireciona a entrada serial na saída serial.

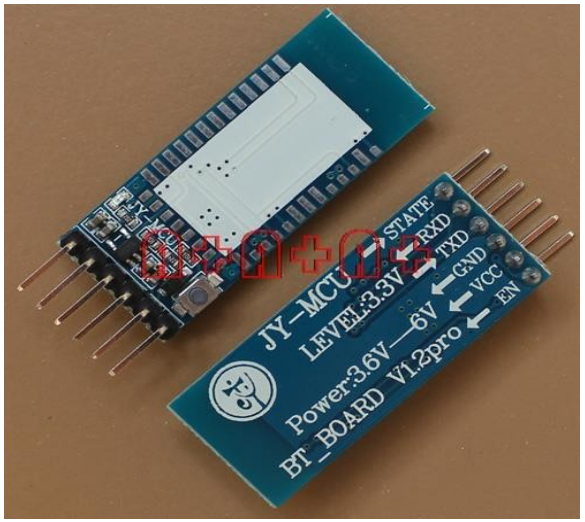




Bluetooth

Módulo Bluetooth – Serial TTL 3,3 volts





Obs:

- Cada módulo do laboratório tem um ID diferente na hora de “parear” com o smartphone.
- Usar a lib UART do MikroC - UART1.