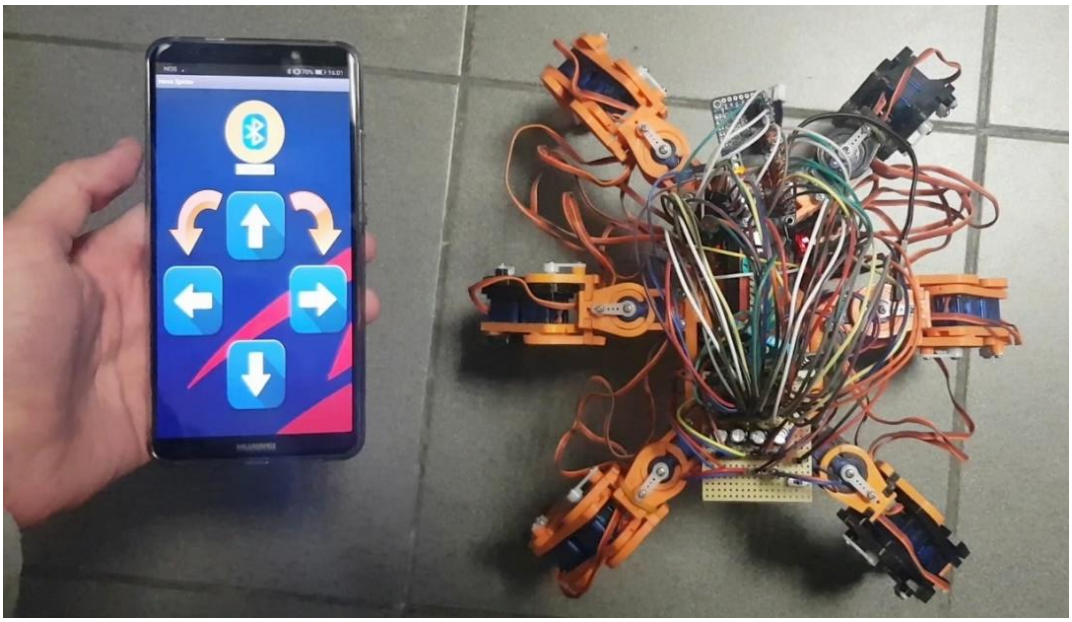


Manual de Instruções

Curso Técnico de Eletrónica, Automação e Comando

Escola Profissional Gustave Eiffel

Luís Santos Nº115693 Turma: 474

Materiais necessários

- Micro Servos SG90 (18x);

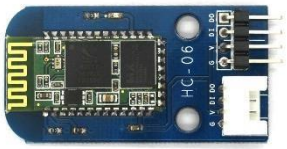
- Driver Servos I2C de 16 canais (1x);

- Bateria 7.4V – 2300mAh (1x);

- Regulador de Tensão (4x);

- Arduino Uno (1x);

- Módulo Bluetooth HC-06 (1x);



Ficheiros STL:

Board bottom; board up; dcdc mount; leg 2xservo bottom; leg 2xservo top; leg crew b; leg crew c; leg crew p; leg crew t; servo bottom, servo top; twist mount; twist ring; twist ring b.

Telecomunicações

As redes e dispositivos de comunicações mantêm-nos permanentemente ligados ao mundo. Os dispositivos eletrónicos e os computadores são parte da nossa vida, sendo as suas aplicações infindáveis.

Usa essencialmente grandezas elétricas de pequena amplitude e/ou de elevadas frequências, sendo designados por sinais elétricos ou eletrónicos. A engenharia eletrónica trata da energia elétrica sob os aspetos de controlo, automação e telecomunicação.

Frequência

É a representação gráfica da onda ao longo do tempo e o ciclo de várias ondas em um determinado tempo. A frequência de uma forma de onda é a quantidade de vezes que a onda oscila, sendo representada por Hertz (Hz). O período, normalmente indicado por *T*, é o período de tempo correspondente a um ciclo, e é o recíproco da frequência *f*:

$$f = \frac{1}{T}$$

Radiofrequência

As radiofrequências são um conjunto de radiações eletromagnéticas que oscilam simultaneamente no campo elétrico e magnético. As radiofrequências ocupam a banda dos 3kHz aos 300GHz no espectro eletromagnético. Todos os equipamentos transmissores produzem ondas eletromagnéticas de radiofrequências, como o controlo remoto.

São sinais que se propagam, normalmente, pelo cobre. Para além deste tipo de sinal, existe o sinal por uma antena, onde se converte o meio cablado por um sinal *wireless*.

Para este sinal de propagação, a forma de onda é uma perturbação ou variação que transfere uma energia progressivamente de ponto para ponto num meio, podendo haver vários tipos de formas de onda.

Comunicação via Bluetooth

Atualmente os dispositivos podem enviar e receber informações através das redes Wi-Fi e 3G. Porém, a tecnologia Bluetooth é uma alternativa a ser utilizada para transmissão de dados em curta distância, de forma gratuita, e sem a necessidade de uma infraestrutura de rede. A transmissão de dados é feita por meio de radiofrequência, que permite que um dispositivo detete o outro independentemente das suas posições, sendo necessário apenas que ambos estejam dentro do limite de proximidade. Para tal, existem 3 níveis de proximidade, podendo ter até 1 metro, até os 10 metros, ou então o nível mais alto, chegando aos 100 metros de proximidade.

Módulo Bluetooth HC-06

Um módulo HC-06 é um dispositivo com o intuito de estabelecer a comunicação, neste caso, entre um aparelho que contenha Bluetooth e um Arduíno, no qual também pode ser usado um aparelho como um tablet ou computador.

Com este módulo Bluetooth é possível enviar via porta serial, com o RX/TX, até aproximadamente 10 metros de distância. É um dispositivo que trabalha com 3,3V e com 5V, onde o mais recomendado pelos fabricantes é a utilização dos 5V para um bom funcionamento do módulo, já que as placas Arduíno funcionam com os 5V.

Antes de se comunicar com o módulo, é preciso que haja o emparelhamento com o dispositivo a utilizar. Este processo vai variando dependendo do tipo de Sistema Operativo que se utiliza no dispositivo, mas em termos gerais é necessário que exista a procura de módulos Bluetooth ao redor do telemóvel, através do acesso às definições do telemóvel, neste caso. De seguida, é possível encontrar um endereço chamado de "itead", no qual se tem que inserir um código de ativação, sendo este o código: "1234".

O módulo Bluetooth tem a porta serial totalmente modificada para 3Mbps, com *Bluetooth* V2.0 + *EDR* (*Enhanced Data Rate*), com transcetor de rádio completo de 2,4 GHz e banda de base. Usa um *CSR Bluecore* 04 com um sistema externo para um chip único com tecnologia *CMOS* e *AFH* (Função adaptativa de frequência).

Este dispositivo dispõe de 4 pins, sendo dois deles os de alimentação, onde o G é o pin de alimentação negativo e o V é o pin de alimentação positiva. Os outros dois pins são os de receção e transmissão de dados, em que o que transmite encontra-se indicado por D1 (*TX*) e o outro pin é o recetor representado por D0 (*RX*).

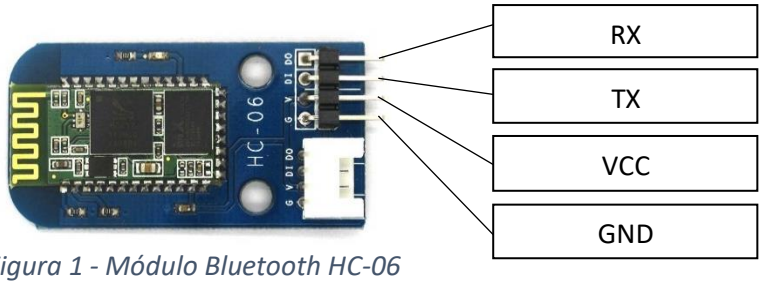


Figura 1 - Módulo Bluetooth HC-06

Componentes

Microcontroladores

Um microcontrolador é um circuito integrado, onde é possível reunir um certo circuito que se queira implementar para que exerça uma determinada instrução dada pelo programador. São pequenas placas que possuem todos os recursos necessários para que se consiga inserir uma dada programação, sendo ainda composta por um circuito de alimentação próprio, para a alimentação dos componentes que se queira utilizar no circuito.

Usa-se microcontroladores para que se possa controlar circuitos a partir de informações que são transmitidas entre um computador e o respetivo microcontrolador.

De uma forma esquemática, um microcontrolador é composto por uma entrada de alimentação, proveniente do cabo que é ligado com computador, em que depois faz funcionar cinco elementos importantes para um bom desempenho do microcontrolador.

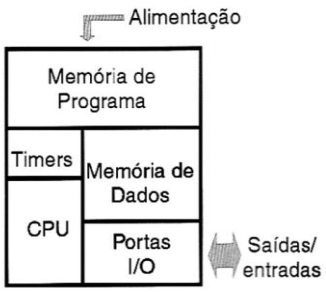


Figura 2 - Constituição de um microcontrolador

A **Memória do Programa** designa-se pelo espaço ocupado pelo programa criado em que o tamanho dessa memória varia segundo a finalidade e o tipo de microcontrolador, tem ainda a **Memória de Dados** que é onde são guardadas as informações/instruções dadas através da programação. Existe ainda a ***CPU***, sendo considerada pela parte mais importante de um microcontrolador, que é responsável pelo processamento das informações, onde as siglas significam, em inglês, *Central Processing Unit*, e em português, é Unidade Central de Processamento.

Existem também ***Timers*** como elementos fundamentais, em que consistem no envio de informações através de intervalos de tempo precisos, onde esse processamento é feito com o auxílio de um "*clock*".

Por fim, existem as **portas de *Input* e de *Output*** onde através de um chip existe a conversão das saídas pela ligação com as portas digitais e analógicas da placa, podendo elas funcionar como portas de saída ou como portas de entrada dependendo do tipo de circuito que se queira implementar.

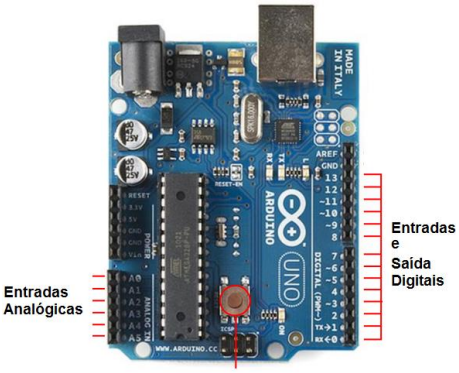


Figura 3 - Portas Analógicas e Digitais

Micro Servo

Os servomotores são dispositivos de malha fechada, ou seja, recebem um sinal de controlo, verificam a posição atual e atuam no sistema indo para a posição desejada.

Um servo é sempre de alta qualidade e excelente para as suas necessidades, seja em aeromodelismo ou em projetos de robótica com Arduíno. Os servomotores são usados em várias aplicações quando se deseja movimentar algo de forma precisa e controlada.

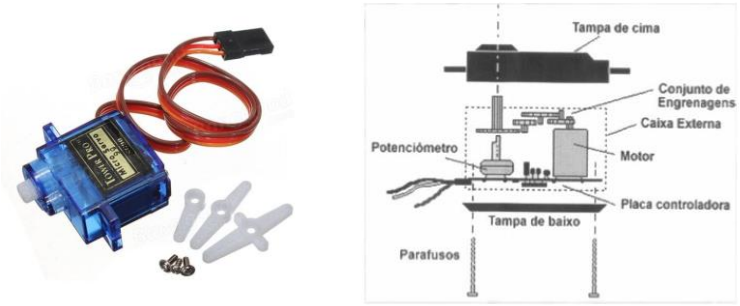


Figura 4 - Representação de um servomotor

Um servomotor encontra-se sempre com sinais de impulso de 20 ms, em que quando há uma mudança nesse sinal faz com que seja alterada a posição do servo, para que corresponda ao sinal recebido.

Um sinal com largura de pulso de 1 ms corresponde a posição do servo todo a esquerda ou 0 grau. Um sinal com largura de pulso de 1,5 ms corresponde a posição central do servo ou de 90 graus. Um sinal com largura de pulso de 2 ms corresponde a posição do servo todo a direita ou 180 graus.

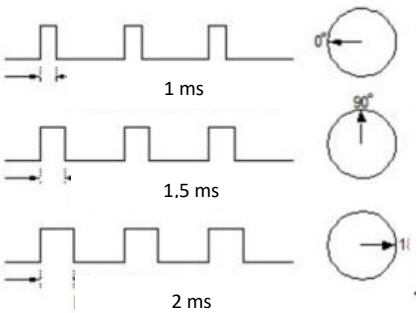


Figura 5 - Sinais de Impulso

Estes servomotores são compostos por 3 pins para que seja feita a ligação com o Arduíno. Para tal, existem dois desses pins que são para a alimentação do servo (*GND* e *VCC*), pin castanho e vermelho, respetivamente, e um de comunicação/sinal, representado pela cor amarela.

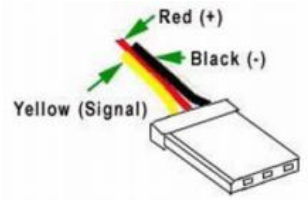


Figura 6 - Representação dos pinos

Os tópicos seguintes são representativos dos elementos que constituem um micro servo, no qual serão explicados cada um deles:

Sistema Atuador – O sistema atuador é constituído por um motor elétrico, onde na maioria das vezes se utiliza motores de corrente contínua. **O tamanho, o binário e a velocidade do motor, material das engrenagens, liberdade de rotação do eixo e consumo** são características chaves para especificação dos servomotores.

Potenciómetro – É um potenciómetro solidário ao eixo do servo. A qualidade do potenciómetro interfere na precisão, estabilidade e vida útil do servo motor.

Circuito de Controlo – É formado por circuitos integrados e é composto por um oscilador e por um controlador PID (Controlo proporcional integrativo e derivativo) que recebe um sinal do potenciómetro (posição do eixo) e o sinal de controlo e aciona o motor no sentido necessário para posicionar o eixo na posição desejada.

Engrenagens – Reduzem a rotação do motor, transferem mais binário ao eixo principal de saída e movimentam o potenciómetro junto com o eixo.

Driver Servos I2C de 16 canais

Este módulo usa o protocolo de comunicação I2C para estabelecer a interligação com um microcontrolador, mais especificamente com um Arduíno, sendo necessário a conexão de dois pinos, que são o SDA e o SCL, para além dos pinos de alimentação para acionar a placa, que são o GND e o VCC.

O controlo PWM é feito pelo próprio módulo para todos os 16 canais, simultaneamente sem sobrecargas de processamento por parte do microcontrolador em utilização. Com este tipo de drivers é possível estabelecer uma conexão em cascata permitindo que 62 drivers estejam interligados, fazendo com que seja possível controlar 992 servos ao mesmo tempo, tudo isto através dos dois pinos referidos anteriormente.

Este driver é muito utilizado para projetos que necessitem de muitos servos, como por exemplo, robots bípedes e robots com formato de aranha, como é o projeto descrito neste relatório.

Figura 7 - Shield Adafruit 16 Channels

Características:

- Usa o endereço I2C de 7 bits entre 0x60-0x80, selecionável com jumpers;
- Conectores de 3 pinos em grupos de 4 para que se possa conectar 16 servos ao mesmo tempo;
- Driver controlado por I2C com um relógio incorporado;
- Controlo com microcontroladores a partir de 3.3V;
- Frequência ajustável até cerca de 1.6KHz;

Pinos SDA e SCL

O SCL é a linha do relógio. É usado para sincronizar todas as transferências de dados pelo barramento I2C. O SDA é a linha de dados. As linhas SCL e SDA estão conectadas a todos os dispositivos no barramento I2C.

A tabela abaixo mostra onde os pinos TWI estão localizados em várias placas Arduíno.

Placa	Pinos I2C/TWI
Uno; Ethernet	A4 (SDA); A5 (SCL)
Mega2560	20 (SDA); 21 (SCL)
Leonardo	2 (SDA); 3 (SCL)
Due	20 (SDA); 21 (SCL); SDA1; SCL1

A conexão de um servo à placa da Adafruit é feita através do conector padrão de 3 pinos existente no servo, que pode ser conectado diretamente à placa.

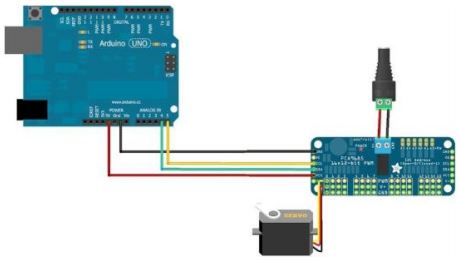


Figura 8 - Conexão de um servo

Com mais de uma placa da Adafruit é possível estabelecer uma conexão designada de cascata, no qual se consegue interligar 62 placas simultaneamente, fazendo com que se passe de 16 saídas PWM para 62 placas, resultando em 992 saídas PWM. Esta interligação é feita com a ligação dos pinos de alimentação (Vcc e GND) e os pinos de transmissão (SDA e SCL), passando isto de placa para placa como demonstra a seguinte figura.

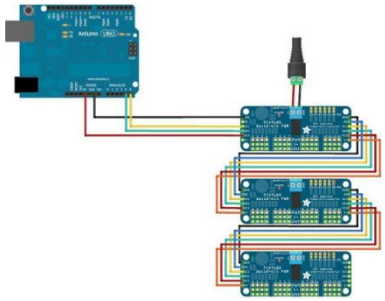


Figura 9 - Conexão em Cascata

Construção da Hexa Spider

Com este capítulo pretende-se demonstrar todos os passos relativos à construção da Hexa Spider, no qual é possível referir o encaixe entre as várias peças e o encaixe dos componentes com as peças impressas.

NOTA: Antes de se dar início à construção do projeto tem que se ter em conta que todos os servos de todas as pernas, têm de ser colocados na posição de 90 graus, para que depois de montada a Hexa Spider, esta esteja devidamente calibrada em todas as pernas.

De seguida, o primeiro passo a ser feito, em relação à construção do projeto, é a colocação dos servos na peça 3D com a capacidade de agrupar dois servos ao mesmo tempo, seja através da peça 3D que é colocada na parte da roda dentada do servo, bem como na base do servo, tal e qual como demonstram as seguintes imagens.

Figura 10 - Agrupamento de 2 servos na peça

O segundo passo consiste na colocação de um elo rotativo na zona circular que agrupa os dois servos referidos anteriormente, capazes de executar os movimentos predefinidos para o projeto. De salientar, que esses elos rotativos são, neste caso, compostos por um pequeno pilar para a colocação de um parafuso. Depois de colocados os elos, são postas as patilhas que são encaixadas à roda dentada dos servos, colocando dois parafusos para sustentar a estrutura feita.

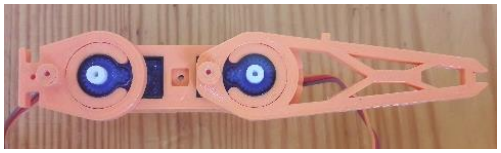


Figura 101 - Elos de rotação

Parafusos e patilhas necessárias para a rotação dos servos.

O próximo passo consiste na colocação das mesmas peças, com a diferença que estas não têm o pilar para agrupar o parafuso, na peça que cobre a base dos dois servos. De seguida, une-se a ponta da perna do projeto através de uma peça relativamente oval, que irá servir de superfície da perna.

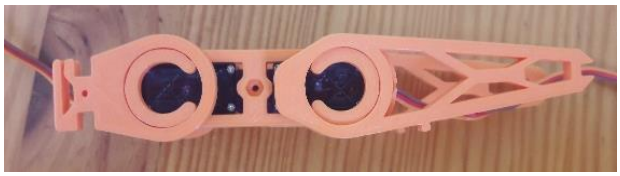


Figura 122 - Elos na base dos servos

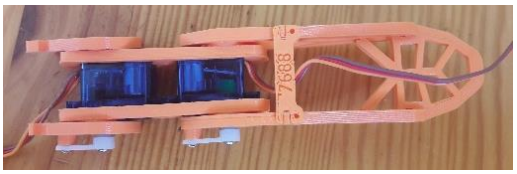
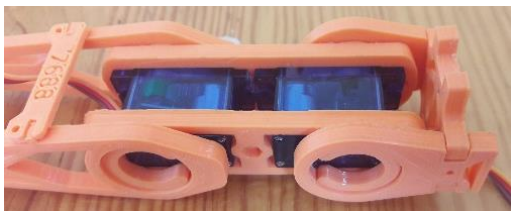


Figura 113 - Junção das pontas da perna

De seguida, através de uma peça constituída por quatro junções para outras peças, será possível agrupar as peças com um formato relativamente redondo já colocadas anteriormente, colocando-as de forma oposta uma à outra, tal e qual como demonstram as seguintes imagens.

As outras duas junções restantes serão explicadas em passos seguintes, já que é necessário realizar outros passos antes do referido anteriormente.



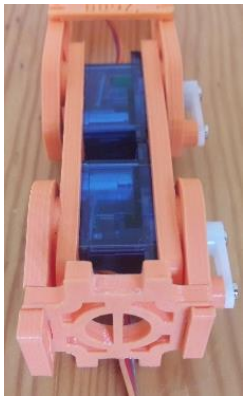


Figura 14 - Junção das peças circulares

O próximo passo é incluir o servo que faltava na estrutura. Sendo assim, com as peças que agrupam apenas um servo, é preciso colocar uma dessas peças na parte da roda dentada e outra na base do servo. Depois de colocadas essas peças, será preciso voltar a colocar os elos rotativos, no qual o elo que contem o pilar para o parafuso fica do lado da roda dentada, enquanto que o elo que não tem o pilar fica do lado da base do servo. Por fim, insere-se a patilha e os parafusos que fixam o servo à estrutura.

Figura 15 - Elos rotativos

Figura 16 - Patilha e parafusos

O passo seguinte consiste em unir a peça que contem apenas um servo com a peça que agrupa dois servos, isto tudo através da peça com quatro junções, colocando os elos rotativos nos espaços que faltam dessa peça de quatro junções.

As imagens seguintes demonstram como é feita essa mesma união, ficando assim a perna completa, sendo agora preciso de aplicar o mesmo método para as outras cinco pernas.

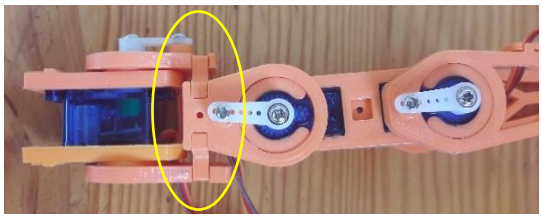


Figura 17 - Junção das partes

Depois de realizada a montagem das seis pernas constituintes do projeto, é preciso interligá-las com a estrutura oval, que é a zona de colocação dos componentes. Sendo assim, a peça que agrupa apenas um servo contem dois furos que se conectam com a estrutura oval, no qual se irá colocar dois parafusos que se possa sustentar a perna à peça oval.

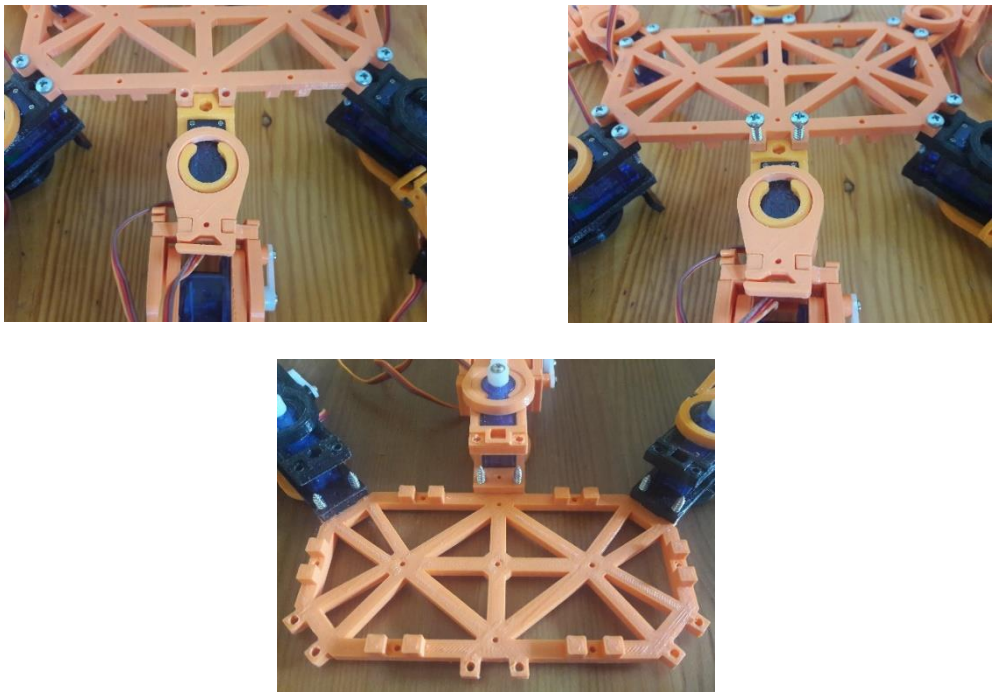


Figura 18 - Junção à estrutura principal

Circuito da Hexa Spider

Será agora demonstrada a representação do circuito num *software* que simula circuitos, onde será possível visualizar as ligações que devem ser feitas para uma montagem correta da Hexa Spider. Sendo assim, a tensão da bateria é regulada com os quatro reguladores de tensão para uma voltagem de 5V. A partir do primeiro regulador, alimenta-se o microcontrolador, o shield e o módulo bluetooth. O segundo regulador alimenta a primeira secção de 6 servos (2 pernas, onde 2 servos estão ligados aos pinos digitais do microcontrolador). O terceiro regulador alimenta a segunda secção de 6 servos (2 pernas). O quarto regulador alimenta a última secção de servos (2 pernas).

A conexão do módulo bluetooth é feita através do RX e TX ao porto de comunicação série. O shield é conectado a partir dos pinos SCL e SDA A4 (SDA); A5 (SCL) ao microcontrolador, nos pinos analógicos A5 e A4, respetivamente. Por fim, existe a conexão de dois servos ao microcontrolador, porque não existe mais espaço de conectividade no shield PCA9685, ficando assim ligados aos pinos digitais 7 e 8.

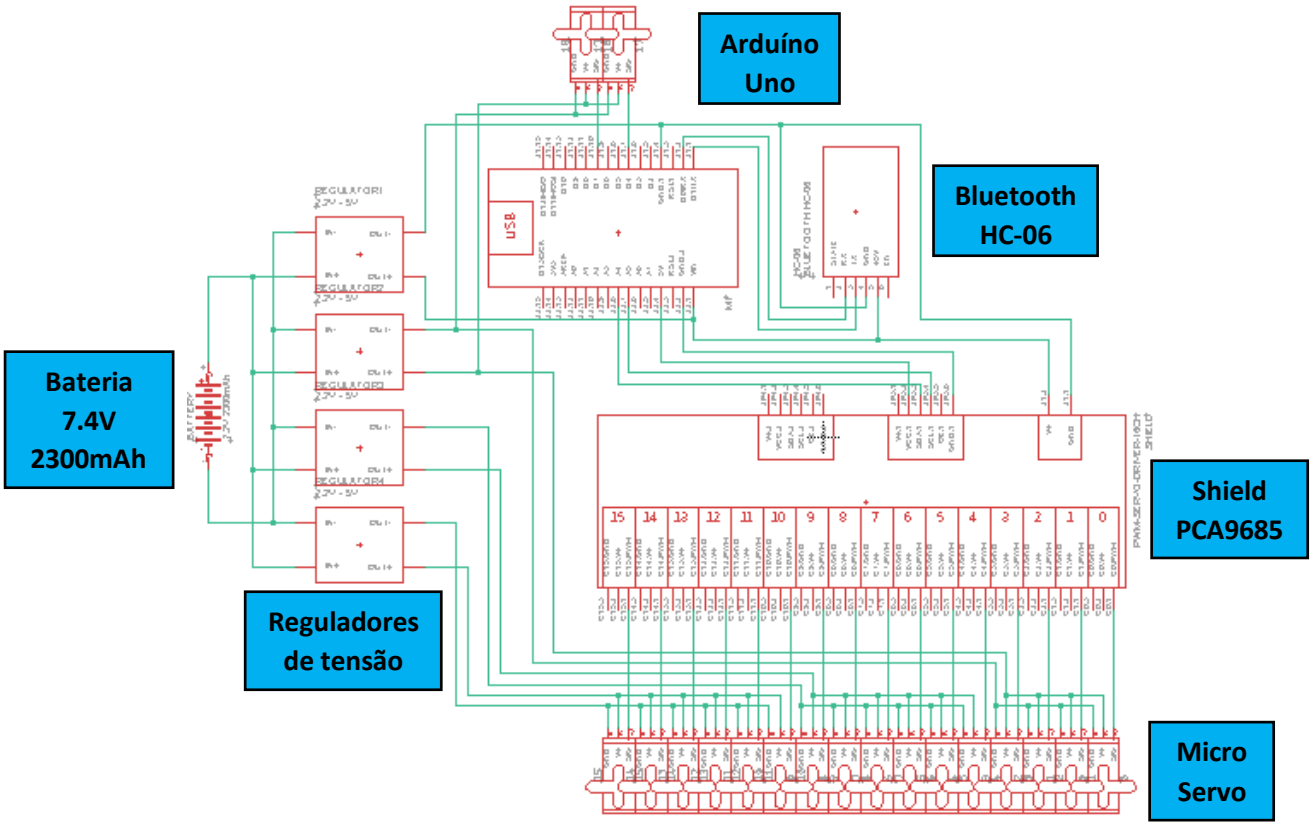


Figura 19 - Circuito em Eagle

Programação do Microcontrolador

Arduíno Uno

Neste projeto foi utilizado o Arduíno Uno que é uma placa de microcontrolador baseado no ATmega328P. Este Arduíno é composto por 6 portas analógicas, e 14 portas digitais. Dispõe de um "*clock*" com uma velocidade de 16MHz, tem 4 pinos de alimentação (5V e *GND*), e pinos de comunicação (*RX* e *TX*).

A utilização deste Arduíno Uno, ao contrário de outro tipo de microcontrolador existente no mercado, deve-se a ser um microcontrolador que apresenta características suficientes para o controlo do resto do circuito do projeto.

Figura 20 - Arduíno Uno

Objetivo da programação

Com esta programação tenciona-se que, a partir da aplicação criada no MIT App Inventor, sejam enviados caracteres, via *Bluetooth*, fazendo com que as funções predefinidas e programadas no código sejam executadas, colocando neste propósito, o projeto em andamento através das seis direções de movimento ou colocar o Hexa Spider a ser controlado a partir do reconhecimento da voz.

Void Setup – Programação

Antes das configurações em relação ao **void setup ()**, é necessária a inclusão de bibliotecas para o funcionamento do shield PCA9685, com intuito de ativar as funções do shield para respetivo funcionamento dos servos. A inclusão da biblioteca é adicionada ao se escrever a linha de código **#include <HCPCA9685.h>**.

```
#include <HCPCA9685.h>
```

De seguida, é predefinido o modelo do shield a utilizar indicando na programação que se irá usar o shield com o endereço de origem, definindo a função **#define I2CAdd 0x40** que consiste no tipo de barramento série a utilizar para que sejam cumpridas certas regras em relação a conexões, protocolos, formatos, endereços e procedimentos. Por fim, adiciona-se o endereço "slave" I2C do dispositivo através da função **HCPCA9685 HCPCA9685 (I2CAdd);**.

```
#define I2CAdd 0x40
HCPCA9685 HCPCA9685(I2CAdd);
```

O próximo passo consiste na configuração dos limites de rotação dos servos, onde se predefine um valor máximo e um mínimo, para que o servo de 180° utilizado não rode mais do que o espectável. Estas funções são definidas da seguinte maneira: **#define SERVOMIN 125** e **#define SERVOMAX 575**.

```
#define SERVOMIN  125
#define SERVOMAX  575
```

Antes do **void setup ()** foi criada uma função chamada **void Ler()** que é capaz de fazer uma nova leitura do porto de comunicação série com a função **if (Serial.available() > 0)** e assim ler um novo caracter, que se for maior que "0" será atribuído à variável "**pos**" que se continuar a receber o mesmo caracter irá retornar uma vez, com a função **return 1**, caso contrário se for recebido outro caracter deixará de retornar e passará a desempenhar outro tipo de função, utilizando a função **return** com o valor lógico de "0".

```
void Ler(){
  if (Serial.available() > 0) {
    pos = Serial.read();
    return 1;
  }
  return 0;
}
```

Dentro do **void setup ()**, serão configuradas todas as funções respetivas dos servos, shield e ainda o porto de comunicação série. Desta forma, e começando pelas funções dos servos, utilizou-se a função **myservo.attach (*pino*)** para se predefinir os dois pinos digitais que serão ocupados pelos servos que não estarão conectados ao shield. Sendo assim, com esta função é possível indicar o pino a utilizar, colocando o número do pino dentro dos parênteses. Esta função utiliza-se apenas para dispositivos que sejam constituídos por pinos de sinal e é utilizada somente para configurar qual o pino a designar, não tendo assim mais nenhum tipo de função concreta.

```
myservo17.attach(9);
myservo18.attach(8);
```

Em relação à parte de se configurar as funções do shield, o primeiro procedimento que se tem de fazer é indicar para que função se irá utilizar os shield, ou seja, que tipo de dispositivo será conectado ao shield. A função a utilizar é **HCPCA9685.Init (SERVO_MODE);** que inicializa o modo servo, fazendo com que o shield esteja preparado para funcionar com servos e não com outro tipo de dispositivos. Por fim, é necessário colocar o shield como um dispositivo ativado e para isso utiliza-se a função **HCPCA9685.Sleep (false);** para que a condição "sleep" esteja sempre com um valor de "false" (desativado), colocando o dispositivo sempre ligado.

```
HCPCA9685.Init(SERVO_MODE);
HCPCA9685.Sleep(false);
```

A última configuração é respetiva à inicialização da porta série. Desta forma, é colocada a função **Serial.begin (9600);** que faz com que se dê início à comunicação da porta série, dizendo ainda com que tipo de velocidade é que se pretende essa transmissão, que neste caso é de 9600 bits por segundo.

```
Serial.begin(9600);
```

Void Loop – Programação

Dentro do **void loop ()** serão colocadas todas as funções que a Hexa Spider terá de desempenhar para se movimentar. Sendo assim, a primeira instrução a ser dada é a criação de uma condição capaz de avaliar todos os caracteres vindos do porto de comunicação série, com a função **if (Serial.available () > 0)**, e se esses caracteres forem maiores que "0" fará com que o porto de comunicação série os leia e os atribua à variável "pos", a partir da seguinte função **pos = Serial.read();**.

```
if (Serial.available() > 0) {
pos = Serial.read();
```

A partir daqui começa a ser feita as sequências e as respetivas condições de teste para que a Hexa Spider desempenhe as funções pretendidas pelo utilizador. Desta forma, são introduzidas as tais condições que atribuem os caracteres recebidos a partir do porto de comunicação série, **if (pos == '1')**, que se corresponderem a um valor superior a 0, são direcionadas para uma certa sequência de movimento. **Exemplo:** Se o caracter recebido for "1", fará com que este caracter desempenhe a sequência de movimento de andar para a "Frente", mas se o caracter recebido for igual a "2" já terá que ser desempenhada a função de andar para a "Direita".

Movimento "Frente"

```
if (pos == '1'){
  sequencial:
HCPCA9685.Servo(4, 355 );
HCPCA9685.Servo(8, 275 );
HCPCA9685.Servo(2, 335 );
delay(350);
HCPCA9685.Servo(10, 340 );
HCPCA9685.Servo(11, 320 );
myservo18.write(150);
```

```
if (pos == '2'){
  sequencia2:
HCPCA9685.Servo(2, 235 );
HCPCA9685.Servo(5, 205 );
HCPCA9685.Servo(8, 185 );
delay(350);
HCPCA9685.Servo(9, 210 );
HCPCA9685.Servo(12, 230 );
HCPCA9685.Servo(15, 205 );
```

Movimento "Direita"

Em seguimento, é colocado dentro das condições todo o tipo de rotações que os servomotores têm de desempenhar. Para tal, utiliza-se a função **HCPCA9685.Servo (*pino*, *grau de rotação*);** para que se consiga rodar todos os servos interligados ao shield, fazendo com que esta função consiga converter a disposição de graus entre 0 e 180 para os parâmetros já referidos anteriormente, como é o caso dos 125 aos 575 (mínimo e máximo, respetivamente, que os servos podem rodar).

Em relação aos servos interligados diretamente ao microcontrolador, a função a utilizar é **myservo.write (*grau de rotação*);** que, neste caso, utiliza os parâmetros de graus normais e os mais utilizados que são entre 0 e 180 graus.

```
HCPCA9685.Servo(11, 320 );
myservo18.write(150);
```

Por fim, para que a Hexa Spider ande apenas quando o botão pretendido é pressionado, foram colocados certos testes para que quando se pressiona um botão o projeto desempenhe de imediato a função e quando se deixar de pressionar execute neste preciso momento uma outra função.

Estes testes são colocados durante a execução de um movimento, sendo eles constituídos, neste caso como exemplo, por uma condição **if (pos == '1')** e por outra que é **else if (pos == '0')**, em que na primeira função existe uma nova leitura da porta série com a função **Ler ();** e caso continue a ser enviado o mesmo caracter, direciona-se a leitura do código para o início da função de deslocamento para que se repita o mesmo deslocamento, isto através da função **goto sequencia1;** que obriga o código a ir para a linha de código desejada, colocando a linha de código **sequencia1;** no início da função do deslocamento. No caso da segunda função, existe uma nova leitura da porta série com a função **Ler ();** e caso receba um novo caracter, faz com que a leitura do código seja direcionada para outro tipo de deslocamento do projeto, através da função **goto sequencia0;**. Neste exemplo, a função **if (pos == '1')** corresponde à função de deslocar o projeto para a "Frente" já que se tem o botão a ser pressionado, enquanto que a função **else if (pos=='0')** corresponde à função de colocar o projeto em modo "Posição Inicial", ou seja, coloca o projeto parado já que se deixou de pressionar o botão.

```
if (pos == '1'){
  sequencial:
HCPCA9685.Servo(4, 355 );
HCPCA9685.Servo(8, 275 );

 if (pos == '1'){
   Ler();
   goto sequencial;
 }
```

```
if (pos == '0'){
  sequencia0:
HCPCA9685.Servo(2, 235 );
HCPCA9685.Servo(5, 205 );

 else if (pos == '0'){
   Ler();
   goto sequencia0;
 }
```

Em relação à parte do reconhecimento da voz, é inserida a instrução **while(Serial.available())** para que enquanto a porta série estiver disponível, haja a leitura da variável "**char c**" a cada 10 milisegundos. Esta variável funcona como forma de adição por parte da outra variável "**voice**".

```
while(Serial.available()) {
  delay(10);
  char c=Serial.read();

  voice += c;
}
```

A partir daqui, é introduzida a instrução **if** com a condição de analisar os dados provinientes da string criada ("**voice**"), sendo esta análise feita pela função **length()**, que determina o comprimento dos dados da string e se esses dados tiverem um valor superior a "**0**", permite que uma procura para ver se os dados recebidos se enquadram a uma função de movimento do robot.

```
if (voice.length() > 0)
```

De seguida, é feito um processo igual ao da receção de caracteres vindos das setas direcionais, ou seja, são colocadas diversas condições, **if (voice == "olá")**, que se os dados recebidos da string forem exatamente iguais às palavras predefinidas para a ativação de movimento do robot, existe a execução do movimento colocando uma função semelhante a esta, **ola();** .

```
if (voice == "olá")
{  ola() ;}
```

No final destas condições todas, é colocada a função **voice = " "**, para que a string criada corresponda à palavra inserida dentro de aspas.

Para finalizar este capítulo, será demonstrado um fluxograma que representa todo o programa que o microcontrolador desempenha. Desta forma, o fluxograma é composto pela inicialização do projeto, que de seguida começará o processo de leitura de caracteres vindos a partir aplicação Android, via Bluetooth. A partir daqui são testadas todas as condições que agrupam cada uma das funções, que caso seja recebido um certo caracter faz com que haja a ativação da função “Frente”, caso contrário serão testadas o resto das condições. De salientar, que se não for pressionado nenhum botão que desempenhe uma função de movimento, será enviado o caracter “0” que atribui ao projeto uma função de posição inicial, ficando assim até receber um novo caracter que atribua um outro tipo de movimento.

NOTA: Em relação à parte do reconhecimento da voz, o processo de ativação das diversas funções é muito semelhante já que o único aspeto que difere é o envio de palavras em vez de caracteres.

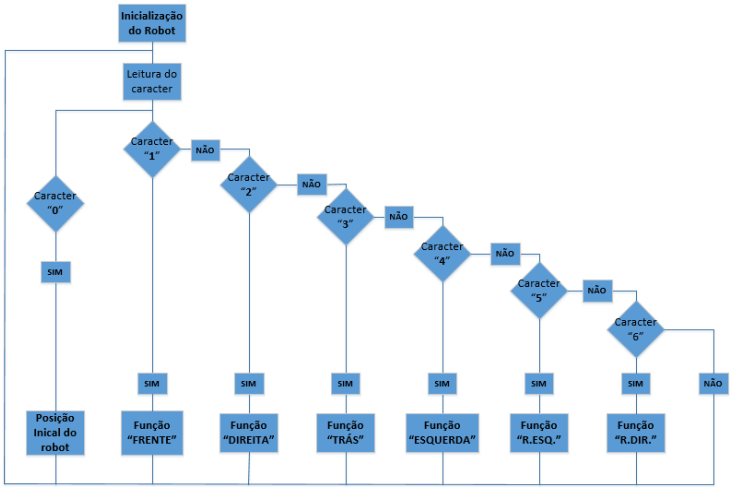


Figura 21 - Fluxograma

Aplicação *Android*

A aplicação desenvolvida é constituída por um botão de conexão com o *Bluetooth* e por seis botões de controlo, entre deslocar-se para a frente, trás ou rodar, entre outros. Dispõe de dois botões, com a função de colocar a Hexa Spider em funcionamento e outro para a desligar. Por fim, contém o botão para o controlo por voz.

Para a conexão do módulo *Bluetooth*, precisa-se de um botão "*Bluetooth*", que quando for pressionado ativará o módulo *Bluetooth* e chamar o respetivo "*BluetoothClient*". Esse bloco do "*BluetoothClient*" faz acionar o endereço do módulo *Bluetooth* através de uma caixa de texto, onde se escreve esse mesmo endereço.

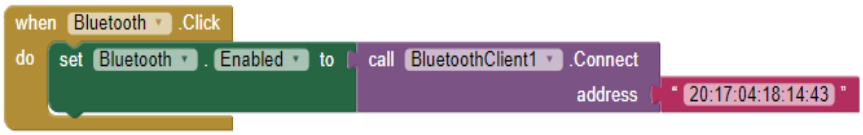


Figura 22 - Bloco de emparelhamento

Será agora demonstrado o processo que é feito para que o robot execute, por exemplo, o deslocamento para a frente. Para esse processo é preciso que, enquanto o botão for pressionado execute essa função, fazendo que se chame o "*BluetoothClient*", enviando sempre o caracter "1", quando já não houver o acionamento desse botão, volta a ser chamado o "*BluetoothClient*", enviando agora o caracter "0" para colocar o robot na sua posição inicial. Este processo, no MIT App Inventor 2, é chamado de "*TouchDown*", que é quando existe o acionamento do botão para a execução dessa função, ou de "*TouchUp*" quando já não existe nada a pressionar o botão.

Este processo é repetido para as outras três direções, apenas havendo a alteração do botão que deve ser pressionado, e o caracter que deve ser enviado para fazer o acionamento da função.

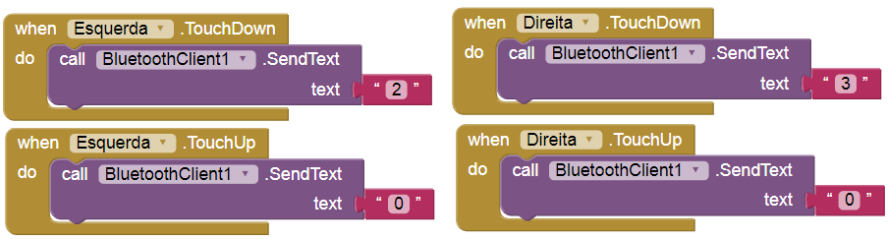


Figura 23 - Blocos de movimento do Robot

Será demonstrado agora os botões responsáveis pela colocação da Hexa Spider em posição de funcionamento e em posição para se desligar. Isto é feito ao se colocar dois botões que têm como objetivo chamar a função "BluetoothClient" e de seguida enviar certos caracteres, a partir de uma caixa de texto, que ativam a posição de estar em funcionamento e a posição de estar pronta para ser desligada, como demonstram as seguintes figuras.

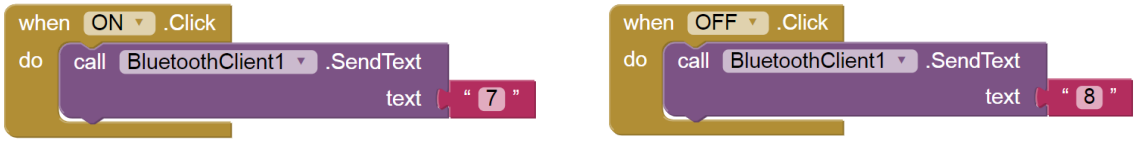


Figura 24 - Acionamento de posições

Para o desenvolvimento do controlo pelo reconhecimento da voz, é necessário que depois de se carregar no botão "*Voice*" haja o acionamento da função "*SpeechRecognizer*" responsável pela detenção da voz através de um texto ("*GetText*"), tal e qual como demonstra a seguinte figura.

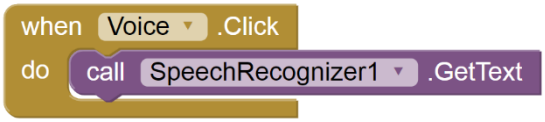


Figura 25 - Ativação do reconhecimento da voz

De seguida, é feita a interação com a "*label*" que se encontra em cima do botão "*Voice*". Esta interação faz-se através da função "*SpeechRecognizer*" que antes de ter recebido algum texto por voz, faça com que a "*label*" se encontre vazia, ou seja, sem texto até receber algo por voz.

Figura 26 - Interação com a "label"

Por fim, depois de ser reconhecido um texto por voz com a função "*SpeechRecognizer*" é atribuída uma função à "*label*" criada para que o texto que foi dito seja transcrito para a "*label*", sendo esse texto coincidente com o resultado da função "*SpeechRecognizer*". Esse mesmo resultado faz com que seja chamada a função "*BluetoothClient*" e envie esse resultado por *Bluetooth* para a programação do microcontrolador, fazendo com que na programação seja ativado um controlo da Hexa Spider.

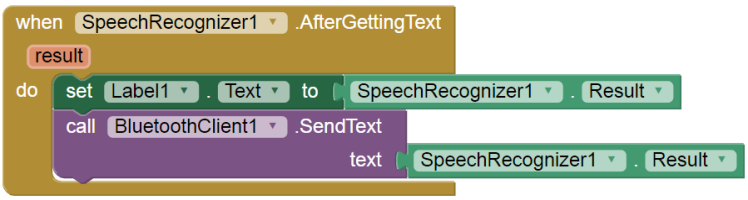


Figura 27 - Funções do reconhecimento por voz

Desta forma, foi possível concluir com a parte de *blocks* da aplicação que comanda todos os movimentos da Hexa Spider, em que para o qual a aplicação encontra-se com os seis direcionais principais (esquerda, direita, frente e trás, rodar para a esquerda e direita) para colocar o robot em andamento para essas mesmas direções, dois botões para colocar o projeto em posição de funcionamento e outro em posição para se desligar, e por fim, o botão responsável pelo reconhecimento de voz.

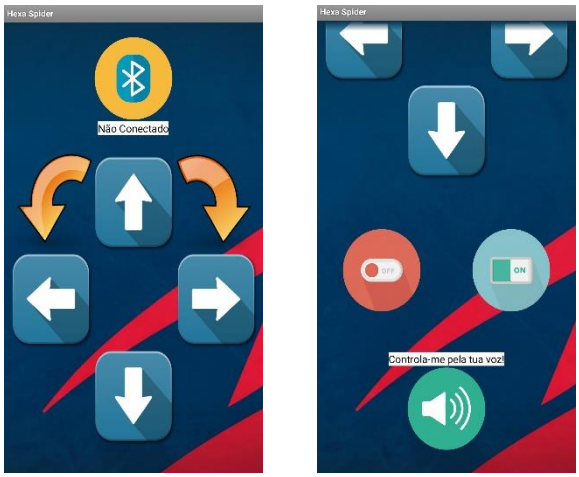


Figura 28 - Aplicação Android