UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO

MANUAL DE UTILIZAÇÃO DO BRAÇO ROBÓTICO LYNX AL5D V2.0

Desenvolvedores:
Engº M.Sc. Danilo Chaves Sousa Ichihara
Engº M.Sc. Desnes Augusto Nunes do Rosário
Engº M.Sc. Jean Mário Moreira de Lima

Julho
2018

Sumário

1 Introdução

Os braços robóticos da *Lynxmotion* têm como objetivo criar aplicações robóticas de forma didática e robusta. Essa empresa é uma das mais antigas fabricantes de kits robóticos didáticos, incluindo braços robóticos, robôs bípedes, quadrúpedes, hexápodes, veículos rastreados com rodas, e muito mais.

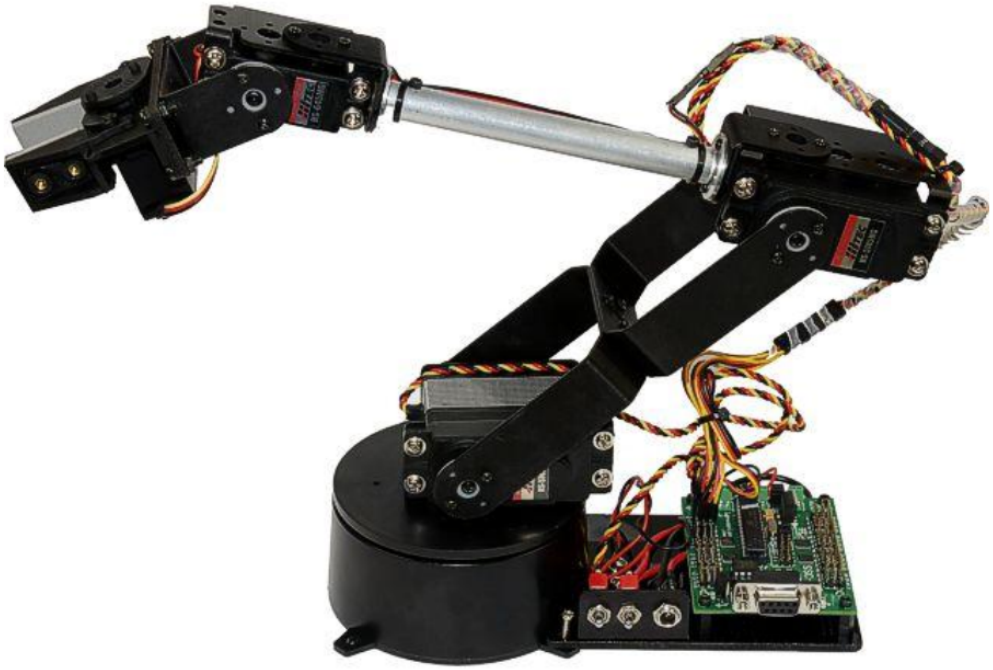

Figura 1: Braço AL5D da *Lynxmotion*

Nos capítulos a seguir, serão abordados os conceitos necessários para a realização do controle dos servo motores do braço robótico AL5D da Lynxmotion, exibido na Figura 1.

2 Braço AL5D

O braço AL5D utiliza uma bateria de 9v para alimentar a placa SSC-32 e uma fonte de 6V/2A para alimentar os servos. Existem alguns softwares da empresa que proporcionam a utilização do braço em sistema Windows, entretanto, sugere-se a utilização da biblioteca ufrn_al5d.h (disponível na seção Produção Acadêmica de http://www.dca.ufrn.br/∼engdesnes/) para programação em C ou a utilização da biblioteca ufrn_al5d.py para programação em PYTHON (disponível na seção Produção Acadêmica de http://www.dca.ufrn.br/∼jean/), ambos para sistemas Linux. É importante informar que pode ser utilizada qualquer linguagem de programação, em qualquer sistema operacional, desde que os protocolos de comunicação para a placa SSC-32 sejam obedecidos corretamente.

Os servos do AL5D são servos de pulso proporcional, projetados para sistemas radiocontrolados (R/C), carros, barcos, aviões e etc. Eles fornecem um controle com precisão para a direção, aceleradores, lemes e etc., fazendo uso de um sinal de fácil de transmissão e recepção. O sinal é constituído por impulsos positivos que vão desde $0,9ms$ a $2,1ms$ (milissegundos) de comprimento, e são repetidos 50 vezes por segundo (a cada $20ms$). Em suma, o servo posiciona o eixo de saída em relação à largura do pulso, tal como ilustrado na Figura 2 abaixo.

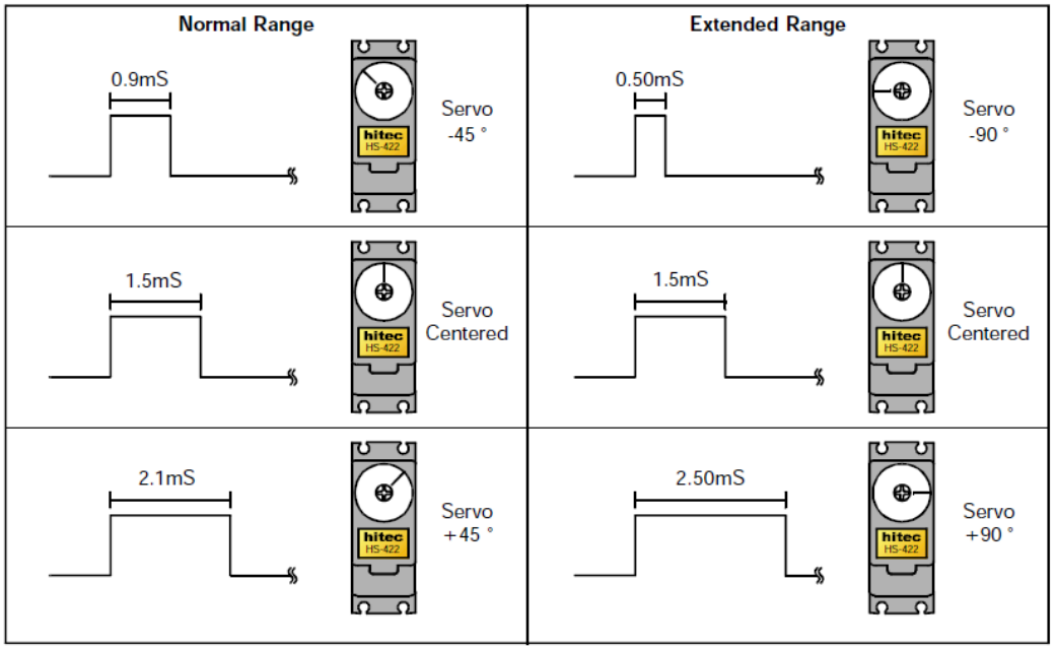

Figura 2: Controle normal e estendido da posição dos servos

Em aplicações de controle de rádio, um servo motor não precisa de mais do que 90° de movimento, já que limitam manivelas que não podem mover-se mais do que 90°. Deste modo, quando pulsos forem enviados no intervalo especificado pelo fabricante ($0,9ms$ a $2,1ms$) será percebido uma rotação por volta de 90° em amplitude de movimento.

A maioria dos servos proporcionam rotações maiores que 90° podendo atingir 180°. No entanto, é indispensável atentar para as limitações mecânicas de alguns servos, afim de evitar danificar estes.

ESTEJA ATENTO DURANTE A PROGRAMAÇÃO PARA NÃO FORÇAR AS LIMITAÇÕES MECÂNICAS DOS SERVOS NOS TESTES

O AL5D permite o uso desta faixa extra por meio de comandos de valores absolutos enviados ao SSC-32.

- O VALOR 500 CORRESPONDE A UM PULSO DE $0,5ms$
- O VALOR 2500 CORRESPONDE A UM PULSO DE $2,5ms$.
- UMA UNIDADE DE VARIAÇÃO PROVOCA UM PULSO DE $1\mu S$
- A RESOLUÇÃO DE POSICIONAMENTO É DADA EM GRAUS POR UNIDADE. PORTANTO:

$$RESOLUCAO = \frac{180^\circ}{2500-500} = \frac{180^\circ}{2000} = 0,09^\circ/unidade \quad (1)$$

- DESTE MODO:

$$POS = \frac{\theta}{RESOLUCAO} + 500 \rightarrow POS = \frac{\theta}{0,09} + 500 \quad (2)$$

Lembre-se que alguns servos não podem e nem devem ser capazes de mover toda a gama de 180°. Tenha muito cuidado ao utilizar os servos, pois caso ultrapasse os limites mecânicos nos seus cálculos, certamente queimará o servo. Teste incrementalmente a extrema esquerda ou direita, buscando o ponto em que incrementos de posição já não resultem em movimentação do eixo de saída. Quando este valor for encontrado, use-o como limite na programação para evitar danos.

REALIZE TRAVAS NO SEU PROGRAMA PARA QUE NUNCA SEJA ENVIADO UM VALOR DE POSICIONAMENTO QUE POSSA DANIFICAR UM SERVO

A comunicação com os servos é realizada via comandos em forma de strings,

que são enviados pela porta serial de um computador para a SSC-32 do AL5D. Essas strings devem seguir o padrão de comandos descrito no próprio manual do SSC-32, que está na pasta *doc* no diretório dos programas demo.

LEIA O MANUAL DO SSC-32 PARA ENTENDER COMO FUNCIONAM OS PROTOCOLOS DE COMUNICAÇÃO NECESSÁRIOS PARA REALIZAR A PROGRAMAÇÃO DOS SERVOS

O Braço AL5D é formado por cinco servo motores, sendo estes definidos como:

- Servo da base: HS-485HB
- Servo do ombro: HS-805BB
- Servo do cotovelo: HS-755HB
- Servo do punho: HS-645MG
- Servo da garra: HS-322HD

No capítulo 3, será discutido a utilização da ufrn_al5d.h para a comunicação com o SSC-32 e programação em C. Já no capítulo 4, será discutido a utilização da ufrn_al5d.py para comunicação com o SSC-32 e programação em PYTHON.

3 Programação em C

3.1 Primeiros Passos

Em http://www.dca.ufrn.br/~engdesnes/ baixe e descompacte a versão 2.0 do *demo_lyxn.tar.gz*. Para a facilidade e dinamização da programação, o programa demo já vem com um Makefile que limpa temporários, realiza compilação e a linkagem da libufrn_lynx.a ao programa executável. Para compilar faça:

```
usuario@maquina:~/Desktop/demo_lynx$ make
```

Como somente o usuário *root* tem acesso a abertura de portas no Linux, o usuário que for executar o programas que utilizem esta biblioteca deve ter tal acesso ou isto pode ser configurado por meio dos comandos de Linux *chown* e *chmod*. A execução pode ser dada da seguinte maneira:

```
usuario@maquina:~/Desktop/demo_lynx$ sudo ./demo
```

3.2 Biblioteca ufrn_al5d.h

Na biblioteca *ufrn_al5d.h* foi definido um padrão de nomes para os servos, representando seu pino de conexão ao SSC-32, assim como o limite de posição de cada servo:

```
// Servo da base HS-485HB //
#define BAS_SERVO 0
#define BAS_MIN 500
#define BAS_MAX 2380

// Servo do ombro HS-805BB //
#define SHL_SERVO 1
#define SHL_MIN 1200
#define SHL_MAX 2000

// Servo do cotovelo HS-755HB//
#define ELB_SERVO 2
#define ELB_MIN 1100
#define ELB_MAX 2100

// Servo do punho HS-645MG //
#define WRI_SERVO 3
```

```
#define WRI_MIN 500
#define WRI_MAX 2500

// Servo da garra HS-322HD //
#define GRI_SERVO 4
#define GRI_MIN 1500
#define GRI_MAX 2400
```

Também na biblioteca *ufrn_al5d.h*, foram definidos as seguintes funções importantes para a comunicação com os servos:

```
/* Abre a porta serial e retorna um file descriptor */
int abrir_porta(void);

/* Configura comunicacao, banda e outros */
int configurar_porta(int);

/* Envia um char* por meio do file descriptor a porta */
/* Retorna -1 em caso de falha */
int enviar_comando(char*,int);

/* Trava o limite maximo para que nao se estoure o limite
    fisico de cada servo */
unsigned int trava(unsigned int canal, unsigned int pos);

/* Fecha a porta serial por meio do file descriptor */
void fechar_porta(int);
```

Por meio desses comandos a comunicação com os servos do AL5D pode ser estabelecida. Agora basta ler os comandos de strings do manual SSC-32. O posicionamento básico é realizado pelo envio de uma string tal como: "#<canal>P<valor_posicao>".

```
/*### EXEMPLO 1: ###*/

  /* Copia para a string comando um pedido de
     posicionamento do servo 2 com valor 500 */
  sprintf(comando,"#2P500");

  /* Envia a string para a porta serial */
  enviar_comando(comando,serial_fd);
```

```
/*### EXEMPLO 2: ###*/

  /* LEMBRE-SE: sempre "zere" o buffer na memoria */
  memset(comando, 0, BUFSIZE);

  /* Copia para a string comando um pedido de
     posicionamento de todos servos com valor 1500 */
  sprintf(comando,"#0P1500#1P1500#2P1500#3P1500#4P1500");

  /* Envia a string para a porta serial */
  enviar_comando(comando,serial_fd);

/*### EXEMPLO 3: ###*/

  /* LEMBRE-SE: sempre "zere" o buffer na memoria */
  memset(comando, 0, BUFSIZE);

  /*
  * Copia para a string comando um pedido
  * de posicionamento do servo do punho
  * com um valor muito alto; que sera travado
  * no limite maximo do servo, realizada
  * pela funcao trava(canal,pos) da ufrn_al5d.h.
  */

  sprintf(comando,"#%dP%d",GRI_SERVO,trava(GRI_SERVO,
     429496729));

  /* Envia a string para a porta serial */
  enviar_comando(comando,serial_fd);
```

EMBORA O MANUAL EXPLICITE A INCLUSÃO DE UM COMANDO <cr> NO FINAL DE CADA STRING, VOCÊ NÃO DEVE FAZÊ-LO. A BIBLIOTECA *ufrn_al5d.h* JÁ IMPLEMENTANO ISTO NO COMANDO DE ENVIO DE STRING.

LIMPE A STRING ANTES DE ENVIAR UM NOVO COMANDO

UTILIZE OS LIMITES MÁXIMOS DE POSIÇÃO DE CADA SERVO E A FUNÇÃO trava(canal,posição), DISPONÍVEIS NA BIBLIOTECA *ufrn_al5d.h*.

Pronto! Muita atenção e agora é só programar.

4 Programação em PYTHON

4.1 Primeiros Passos

Em http://www.dca.ufrn.br/∼jean/ baixe e descompacte o arquivo *demo_lyxn.tar.gz* que contêm o programa demonstração de utilização do braço, o *demo.py*, assim como a biblioteca de comunicação com o braço, o *ufrn_al5d.pyc* que é o *bytecode* do código fonte *ufrn_al5d.py*.

4.2 Programa Demonstração

Para execução do programa demonstração, acesse o diretório do arquivo *demo.py* via terminal e digite:

```
usuario@maquina:~/Desktop/demo_lynx$ sudo python demo.py
```

No programa *demo.py* foi definido um padrão de nomes para os servos, representando seu pino de conexão ao SSC-32, assim como o limite de posição de cada servo:

```
#0. SERVO DA BASE
BAS_SERVO = 0
#LIMITES
BAS_MIN = 500
BAS_MAX = 2400

#1. SERVO DO OMBRO
SHL_SERVO = 1
#LIMITES
SHL_MIN = 1200
SHL_MAX = 2000

#2. SERVO DO COTOVELO
ELB_SERVO = 2
#LIMITES
ELB_MIN = 1100
ELB_MAX = 2000

#3. SERVO DO PUNHO
WRI_SERVO = 3
#LIMITES
WRI_MIN = 500
```

```
WRI_MAX = 2500

#4. SERVO DA GARRA
GRI_SERVO = 4
#LIMITES
GRI_MIN = 1300
GRI_MAX = 2400
```

No código fonte *ufrn_al5d.py* implmentou-se uma classe chamada **RoboticArmAL5D** que contém funções básicas para utilização do braço:

```
  #__init__: Funcao chamada ao criar um instancia
  # da classe RoboticArmAL5D. Recebe como parametros
  # as propriedades do braco: Numeros correspondentes
  # aos servos e respectivos limites
  def __init__(self,properties):

  #setup: Configura a porta serial para comunicacao
  def setup(self):

  #abre_porta: Abre a porta serial
  def abre_porta(self):

  #fecha_porta: Fecha porta e encerra comunicacao serial
  def fecha_porta(self):

  #envia_comando: Responsavel para enviar comandos
  #a placa SSC-32.
  def envia_comando(self,cmd):

  #trava: Travas de seguranca. Recebe como parametro
  #o canal, isto e, o servo que se deseja comunicar
  #e a posicao desejada. Testa a posicao de acordo
  #com os limites pre-estabelecidos para o servo
  def trava(self,channel,pos):
```

Por meio desses comandos, a comunicação com os servos do AL5D pode ser estabelecida. Agora basta ler os comandos de strings do manual SSC-32. O posicionamento básico é realizado pelo envio de uma string tal como: "#<canal>P<valor_posicao>". Além disso, também é possível determinar o tempo de duração da execução da ação por parte do servo. Ao final da string que determina a

posição, adiciona-se "T<valor_tempo>. Esse valor é dado em *ms* (milissegundos) e afeta o movimento de TODOS os servos. Ex: "#0P1500#1P1500T1000". Ambos os servos (0 e 1) levarão 1000*ms* para executarem seus respectivos movimentos. Para mais detalhes, verificar o manual do braço Al5D que está no diretório *doc* do arquivo baixado.

No programa *demo.py* há alguns exemplos da utilização das funções descritas para envio de comandos ao braço.

```
##############################
########  EXEMPLO 1 #########
##############################

print('\nSEGUNDO COMANDO - MOVER O PUNHO\n')
print('Espere 2 segundos...\n')
time.sleep(2)
print('Envio de comando SEM teste de envio: %s \n' % ('#3
    P1900T1500'))
braco.envia_comando('#3P1900T1500')

##############################
########  EXEMPLO 2 #########
##############################

print('\nTERCEIRO COMANDO - MOVER A GARRA\n')
print('Espere 2 segundos...\n')
time.sleep(2)
try:
  braco.envia_comando('#%dP%dT%d' % (4,2400,1500))
  print('Envio de comando com teste de envio: %s \n' % ('
      #4P2500T1500'))
except:
  print('Problema no envio do comando\nAbortando o
      programa...')

##############################
########  EXEMPLO 3 #########
##############################

print('\nQUARTO COMANDO - MOVER A BASE TESTANDO TRAVAS\n'
    )
print('Espere 2 segundos...\n')
time.sleep(2)
try:
```

```
    pos = braco.trava(BAS_SERVO,99999)
    braco.envia_comando('#%dP%dT%d' % (BAS_SERVO,pos,1500))
    print('Envio de comando com teste de envio e de travas:
        %s \n' % ('#0P2400T1500'))
  except:
    print('Problema no envio do comando\nAbortando o
       programa...')
```

EMBORA O MANUAL EXPLICITE A INCLUSÃO DE UM COMANDO <cr> NO FINAL DE CADA STRING, VOCÊ NÃO DEVE FAZÊ-LO. A BIBLIOTECA *ufrn_al5d.py* JÁ IMPLEMENTANO ISTO NO COMANDO DE ENVIO DE STRING.

UTILIZE OS LIMITES MÁXIMOS DE POSIÇÃO DE CADA SERVO E A FUNÇÃO trava(canal,posição), DISPONÍVEIS NA BIBLIOTECA *ufrn_al5d.py*.

Pronto! Muita atenção e agora é só programar.

Bibliografia

[1] LYNXMOTION INC., Manual written for firmware version SSC32-1.03XE Range is 0.50 ms to 2.50 ms, Ver 2.0, Pekin IL 61555-0818.

[2] LYNXMOTION, Lynxmotion Inc. Disponível em: <http://www.lynxmotion.com/> . Acesso em 16 de Maio de 2013.