Taller de Proyecto II

2019

Proyecto N° 5

Consolidación del proyecto del robot humanoide

Grupo de Desarrollo

- Abba, Pedro Nicolás - 1257/5
- Ares, Charo - 1224/5
- Galán, Martín Andrés - 1300/0

Informe Final 5/12/19

Índice

Índice de gráficos

1. Proyecto

Como introducción a este proyecto se presentó un robot humanoide con quince servos funcionales y una interfaz web muy simple que permitía que el mismo realice movimientos servo a servo por indicación del usuario.

A partir de esta base, se plantearon los siguientes objetivos:

- Configurar movimientos humanoides en el robot
- Limitar el rango de movilidad de los servos para asegurar su integridad
- Modificar la interfaz web existente para controlar las acciones en tiempo real
- Añadir un conector entre el sistema de alimentación del robot y la fuente de energía
- Diseñar una placa tipo shield para ordenar y proteger los componentes ubicados en la parte posterior del robot.
- Implementar la interfaz web utilizando Bootstrap
- Ordenar y agrupar los cables de alimentación y control de los servos.

En las siguientes secciones se realizará una descripción del modo de resolución de los objetivos planteados, así como de los problemas que surgieron y las soluciones implementadas.

2. Materiales y Presupuesto

En la siguiente tabla se detallan los materiales utilizados hasta el momento en el desarrollo de este proyecto.

Tabla 1 - Materiales y presupuesto del proyecto

Cantidad	Material	Precio x material	Referencia de precio
1	Robot Humanoide Lobot Robo-soul H3.0	U$D 305	Ref Robot Humanoide
1	Arduino UNO	$656	Ref Arduino UNO
1	Modúlo de WiFi ESP8266-01	$280	Ref Módulo WiFi
1	Placa controladora de servos PCA9685	$364	Ref Placa Controladora Servos
1	Fuente de PC adaptada a 7.5V	$639	Ref Fuente de PC
1	Conector Jack 3.5mm	$24	Ref Conector Jack
1	Regulador de Tensión AMS1117	$50	Ref Regulador de Tensión
1	Cable Impresora USB-A a USB-B	$98	Ref Cable USB
1	Cables para alimentación 10 cm	$6	Ref Cable Alimentación
1	Cables UTP 10 cm	$2	Ref Cable UTP
1	Placa Perforada 10x5	$127	Ref Placa Perforada
1	Bornera 2 Bornes	$10	Ref Bornera
1	Tira de pines hembra 2x20	$37	Ref Tira de pines hembra
1	Tira de pines macho 1x40	$36	Ref Tira de pines macho
1	Tira de pines hembra “L”	$87	Ref Tira de pines "L"
20	Precintos pequeños	$10	Ref Precintos
	Total	**305 U$D + 2426 ARS**	

3. Descripción del Proyecto

3.1. Esquema general del proyecto

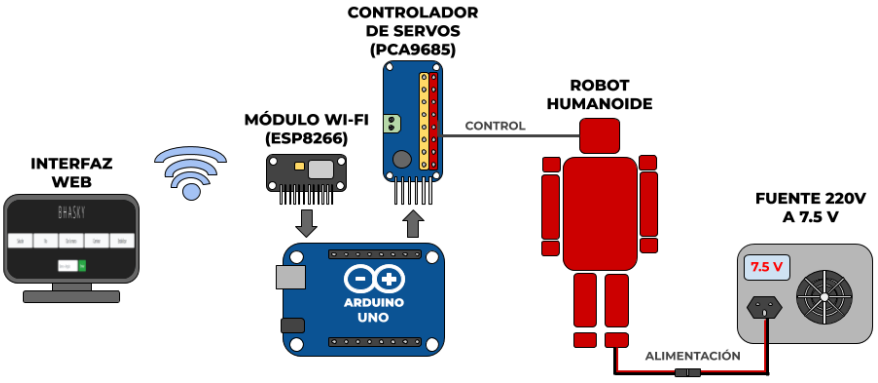

Figura 1 - Esquema general del proyecto

Se dispone de un robot humanoide con diecisiete servos alimentado con una fuente de PC de 220V a 7.5V. Dieciséis servos se encuentran conectados con una placa controladora de servos PCA9685 que permite enviar las señales de control al robot a través del microprocesador Arduino UNO. El robot recibirá las instrucciones de movimiento a través del módulo de WiFi ESP8266, donde se levanta un servidor web con una interfaz amigable para el usuario. En la Figura 1 se puede apreciar un diagrama que muestra los diferentes componentes anteriormente explicados y sus conexiones.

3.2. Conexiones

3.2.1. Esquema general de conexiones

Las conexiones entre los componentes mencionados previamente se encuentran representadas en el esquema de la Figura 2.

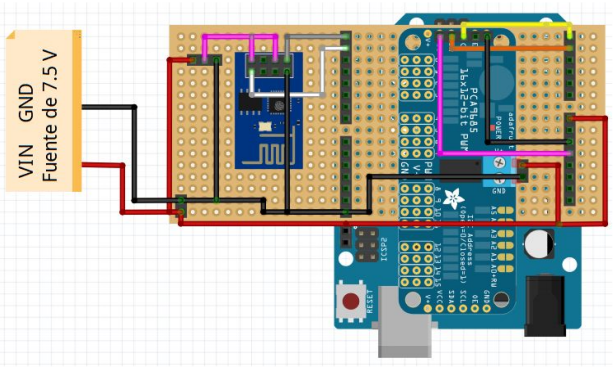

Figura 2 - Esquema general de conexión

3.2.2. Alimentación del dispositivo/placa de desarrollo

Para alimentar el robot y a la placa de desarrollo se tiene una fuente de 220V a 7.5V.

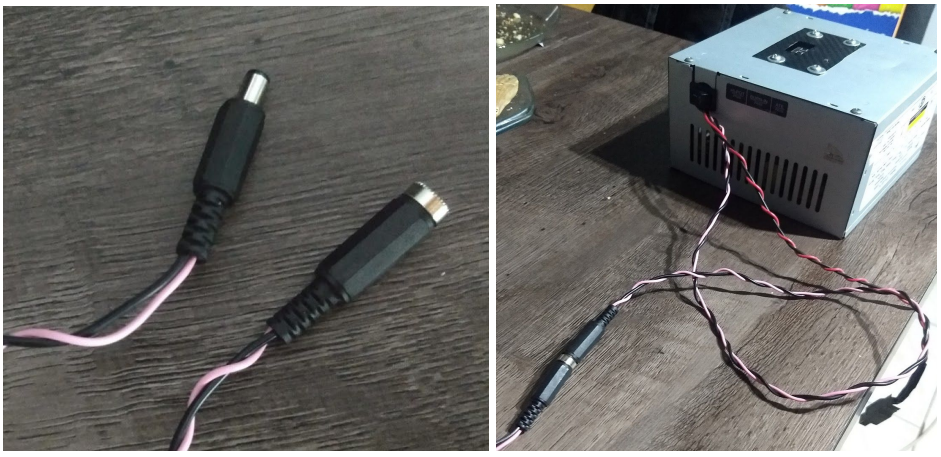

Figura 3 - Conexión entre el Robot y la fuente

3.2.3. Conexiones de E/S de la placa de desarrollo con el exterior excepto PC

Específicamente, la conexión de la placa controladora de servos PCA9685 con el Arduino UNO y los servos puede esquematizarse tal y como se muestra en la Figura 4.

Es importante mencionar que la PCA9685 se encuentra alimentada con 7.5V, es decir, directamente a la tensión de la fuente.

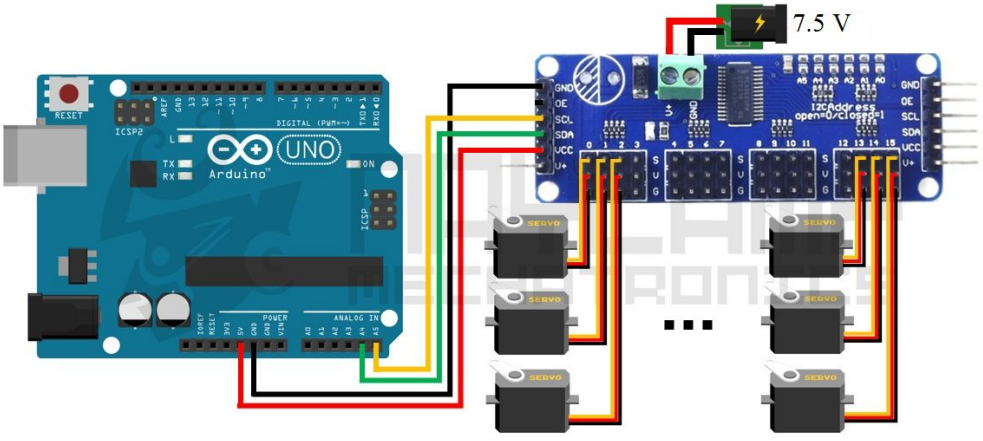

Figura 4 - Conexión entre la PCA9685 y el Arduino UNO

3.2.4. Comunicaciones de la placa de desarrollo con la PC

En la Figura 5 se puede apreciar la conexión realizada entre el ESP8266 y la placa de desarrollo, para realizar la comunicación entre la misma y la PC, además de su alimentación. En este caso, no se alimenta directamente con los 7.5V de la fuente sino que se utiliza un regulador de tensión AMS1117 para disminuir el potencial entregado a 3.3V.

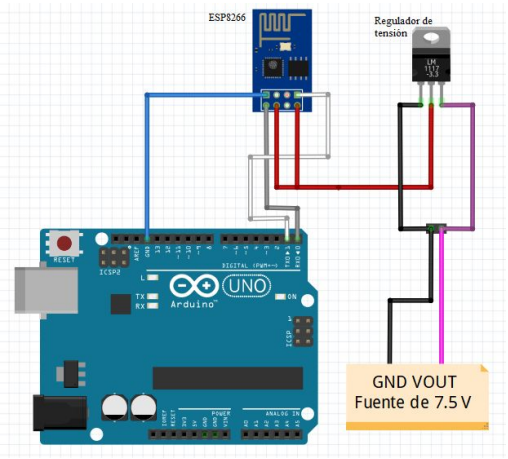

Figura 5 - Conexión entre el ESP8266 y el Arduino UNO

3.3. Sistema/interfaz web

En la Figura 6 se puede apreciar la disposición de los diferentes componentes visuales de la interfaz web.

Figura 6 - Esquema de la Interfaz web

La web se encuentra diseñada a través de su lenguaje correspondiente HTML mientras que para los estilos de la misma se utilizó CSS y una biblioteca de código abierto para componentes frontend conocida como Bootstrap. Para la carga de Bootstrap fue necesario utilizar un plugin para el manejo del FileSystem del ESP8226. En la sección 4 se adjunta un instructivo en caso de querer replicarse este procedimiento.

3.4. Software

3.4.1. Módulo de Wi-Fi ESP8266

3.4.1.1. Comunicación inalámbrica a través del ESP8266

El ESP8266 es el encargado de crear una red WiFi para poder conectarse al mismo a través de cualquier dispositivo. Para llevar a cabo esta acción se puede configurar el WiFi mediante la asignación de un nombre y una contraseña si se desea, tal como se muestra en la Figura 7.

```
//AP SSID and Password
const char *ssid = "Bhasky";
const char *password = "12345678";

ESP8266WebServer server(80); //Web server
```

Figura 7 - Credenciales de la red WiFi

También utilizando la librería ESP8266WebServer se levanta un servidor web capaz de manejar HTTP request tales como GET y POST. El dispositivo estará escuchando las requests por el puerto por defecto: el puerto 80. Asimismo a través de la IP 10.10.10.10 se puede acceder a la interfaz web. Dicha configuración es llevada a cabo mediante diferentes funciones de la misma librería, las cuales podemos ver en la Figura 8.

```
void setup(void)
{
  delay(2000);
  Serial.begin(9600);

  WiFi.mode(WIFI_AP);            //Set AP
  WiFi.softAP(ssid,password);

  IPAddress Ip(10, 10, 10, 10);
  IPAddress NMask(255, 255, 255, 0);
  WiFi.softAPConfig(Ip, Ip, NMask);

  server.on("/", handleRoot);      //GET / ==> run handleRoot()
  server.on("/bootstrap.min.css", bootstrap);

  SPIFFS.begin();
  server.begin();                  //Start web server
}
```

Figura 8 - Setup del ESP8266

La función handleRoot es aquella que se encarga de manejar las request que llegan desde la interfaz web. La misma redirecciona los parámetros recibidos al Arduino a través de la comunicación serie. Mediante la acción llevada a cabo en esta etapa se limitan las acciones que pueda llegar a realizar el robot evitando así que lleguen parámetros indeseados o desconocidos. Por lo tanto los únicos parámetros que permiten realizar una acción son “angles”, “función” o “caminar”,

produciéndose así el envío por comunicación serial de los parámetros que acompañan a la función, como se puede ver en la Figura 9.

```
void handleRoot() {
  String s = MAIN_page;
  server.send(200, "text/html", s);   //Send index(html)

  if ( server.hasArg("angles") || server.hasArg("funcion") || server.hasArg("caminar")){
    if (server.argName(0) == "angles"){
      Serial.println("1");
    } else if (server.argName(0) == "funcion"){
      Serial.println("2");
    } else if (server.argName(0) == "caminar"){
      Serial.println("3");
    }
    Serial.println(server.arg(0));
  }
}
```

Figura 9 - Función handleRoot() del ESP8266

3.4.1.2. Interfaz web

Como se pudo ver en la Figura 6, el frontend del servidor se basa en una interfaz simple a través de la cual se podrá interactuar de tres maneras diferentes; por un lado tenemos un conjuntos de botones que mandan a ejecutar un movimiento predeterminado del robot (Saludar, Dab, etc); por otro lado, se da la posibilidad al usuario de definir la cantidad de pasos a realizar por el robot para ejecutar la función caminar; y como última opción, contiene un input de entrada que le permite al usuario ingresar el número servo que desea mover más el ángulo en el que se desea setear dicho servo. Estas acciones se diferencian a través de un atributo específico de html de las etiquetas input denominado **name**, tal y como se puede ver en la Figura 10, 11 y 12. Este atributo permite especificar el tipo de dato que se está enviando: si es una acción predeterminada tendrá el valor "función", para reconocer que se trata del ingreso de un servo y un ángulo se envía el parámetro "angles" y para la función de caminar, el tipo de datos es "caminar".

```
<label class="sr-only">Angulos</label>
<input type="text" class="form-control" name="angles" placeholder="Servo + Angulo">
```

Figura 10 - Código HTML referente al botón de enviar el servo + angulo

```
<button class="btn btn-lg btn-light mr-2 mt-2 col-md-2" type="submit" name="funcion" value="1" method="get">
  Estabilizar
</button>
<button class="btn btn-lg btn-light mr-2 mt-2 col-md-2" type="submit" name="funcion" value="2" method="get">
  Saludar
</button>
```

Figura 11 - Código HTML referente a los botones de enviar funciones

```html
<h2 class="title-text">Caminar</h2>
<div class="row">
  <h4 class="title-text pasos"> Pasos </h4>
  <div class="form-check form-check-inline">
    <input class="form-check-input" type="radio" id="2pasos"  value='2' name="caminar" checked>
    <label class="form-check-label text-radio" for="2pasos"> 4 </label>
  </div>
  <div class="form-check form-check-inline">
    <input class="form-check-input" type="radio" id="3pasos"  value='3' name="caminar">
    <label class="form-check-label text-radio" for="3pasos"> 6 </label>
  </div>
  <div class="form-check form-check-inline">
    <input class="form-check-input" type="radio" id="4pasos"  value='4' name="caminar">
    <label class="form-check-label text-radio" for="4pasos"> 8 </label>
  </div>
  <div class="form-check form-check-inline">
    <input class="form-check-input" type="radio" id="5pasos"  value='5' name="caminar">
    <label class="form-check-label text-radio" for="5pasos"> 10 </label>
  </div>
</div>
```

Figura 12 - Código HTML referente a los botones de caminar

En el caso específico de las funciones, las request irán acompañadas de un segundo campo denominado value el cual indicará mediante valores numéricos la acción a realizar. Para el caso del movimiento de un servo específico, el valor enviado será el servo más el ángulo hacia el que se tiene que rotar dicho servo. Por último, la función caminar es acompañada por valor correspondiente a los pasos.

Estas request serán recibidas por el ESP8266 específicamente por el módulo handleRoot explicado anteriormente, donde dependiendo del parámetro name se enviará por comunicación serial al Arduino qué acción realizar junto con los valores recibidos en la request.

3.4.2. Arduino UNO R3

3.4.2.1. Definición de constantes y librerías

Para realizar los movimientos de servos, ya sea para las funciones o para los movimientos predeterminados, se utilizaron las librerías Wire.h y Adafruit_PWMServoDriver.h

Se definieron 16 constantes para referenciar cada servo dependiendo de la conexión realizada en la PCA9685, ya que las mismas mejoran la legibilidad del código. Podemos ver estas referencias en la Figura 13.

```
#define LELBOW  0   // Left elbow articulation
#define LSHLDRA 1   // Left shoulder articulation with arm
#define LSHLDRT 2   // Left shoulder articulation with torso
#define LHIP    3   // Left hip articulation
#define LTHIGH  4   // Left thigh articulation
#define LKNEE   5   // Left knee articulation
#define LANKLE  6   // Left ankle articulation
#define LFOOT   7   // Left foot articulation
#define RFOOT   8   // Right foot articulation
#define RANKLE  9   // Right ankle articulation
#define RKNEE   10  // Right knee articulation
#define RTHIGH  11  // Right thigh articulation
#define RHIP    12  // Right hip articulation
#define RSHLDRT 13  // Right shoulder articulation with torso
#define RSHLDRA 14  // Right shoulder articulation with arm
#define RELBOW  15  // Right elbow articulation
```

Figura 13 - Constantes referenciando cada servo

En la Figura 14 podemos ver la definición de arreglos de máximos, mínimos y la posición inicial (home) de cada servo. Los arreglos anguloMin y anguloMax se utilizan posteriormente para validar si es correcto llevar al servo hasta la posición requerida por el usuario. Estos valores fueron elegidos para evitar que el robot se dañe. Los arreglos servoMin y servoMax se utilizan para setear el valor del PWM.

```
int servoMin[] =  {145, 145,  120,  145,  145,  145,  145,  145,  145,  145,  145,  145,  160,  145, 145 };
int servoMax[] =  {640, 640,  640,  640,  640,  640,  640,  635,  635,  640,  640,  640,  640,  630,  640, 640 };
int anguloMin[] = { 80,  15,    5,   80,   75,   85,   35,  60 ,  60 ,   70,   50,    5,   60,   65,    5,  70 };
int anguloMax[] = { 90, 160,  110,  100,  145,  110,   85,  100,  100,  120,   75,   75,   75,  175,  150,  80 };
int posHome[] =   { 90, 160,   85,   80,  110,   85,   60,  80 ,  80 ,  100,   75,   45,   75,   75,    5,  70 };
```

Figura 14 - Definición de los arreglos Min, Max y Home

3.4.2.2. Setup

En la inicialización del código se establece la comunicación serial a 9600 baudios por segundo y se inicia el PWM a una frecuencia de 60 Hz. También se ejecuta la función de estabilización del humanoide para llevar los servos a su posición inicial. En la Figura 15 podemos ver las distintas instrucciones utilizadas para la inicialización del programa.

```
/************************************ SETUP ************************************/
void setup()
{
  Serial.begin(9600);
  Serial.setTimeout(1000); // for parseInt()...
                           //  Serial.println("Go");
  Serial.println("Ingrese :");
  Serial.println("1 SERVO ANGULO");
  Serial.println("2 FUNCION");
  Serial.println("  1 - Estabilizar");
  Serial.println("  2 - Saludar");
  Serial.println("  3 - Decir que no");
  Serial.println("  4 - Dar la mano");
  Serial.println("  5 - Dab");
  Serial.println("  6 - Onda");
  Serial.println("  7 - Arigato");
  Serial.println("  8 - Chuchuwa");
  Serial.println("  9 - Sentadillas");
  Serial.println("3 CAMINAR");
  Serial.println("  2 - 4 pasos");
  Serial.println("  3 - 6 pasos");
  Serial.println("  4 - 8 pasos");
  Serial.println("  5 - 10 pasos");

  pwm.begin();
  pwm.setPWMFreq(60); // Analog servos run at ~60 Hz updates

  posArray = 0;
  pasos = 0;

  estabilizar();
}
```

Figura 15 - Inicialización del programa

3.4.2.3. Main Loop

Una vez realizada la lectura en la función readMonitorSerie() de los parámetros enviados a través del puerto serial desde el ESP8266, se procederá a ejecutar las acciones correspondientes dependiendo del modo elegido por el usuario.

Para el modo "1" se realizará el movimiento del servo y ángulo previamente cargado en sus correspondientes variables, para proseguir a ejecutar la función encargada de mover los ángulos (setAngleParallel, cuya utilidad se explicará más adelante).

En el caso de que el modo elegido sea el "2", se procederá a ejecutar uno de los 9 movimientos predeterminados del robot, el cual se seleccionará a través de una estructura de control switch en base al segundo parámetro recibido.

Si recibimos el modo "3", se deberá ejecutar la función caminar que realizará dicho movimiento de acuerdo a los pasos enviados como parámetro.

3.4.2.4. Lectura del monitor serie

La función **readMonitorSerie()** realiza la recepción del request a través de la comunicación serial y dependiendo de dicho modo, el valor correspondiente a la función a ejecutar, de los valores de servo y ángulo a mover o de la cantidad de pasos a caminar.

3.4.2.5. Carga del arreglo de servos a mover

Para la ejecución de las diferentes acciones a realizar primero debe realizarse la carga de los servos y ángulos que se desean mover en sus correspondientes arreglos, para esto se cuenta con dos funciones: cleanInputs y addInputs.

cleanInputs: Dicha función es la encargada de realizar la limpieza de los arreglos correspondientes a los servos y ángulos y establecer el índice nuevamente en cero.

```
void cleanInputs(){
  int i;
  for(i=0;i<MAX_SERVOS;i++){
    servos[i]=-1;
    angles[i]=0;
  }
  posArray=0;
}
```

Figura 16 - Función cleanInputs

addInput: Esta función se encarga de setear en los arreglos el servo y ángulo que le viene por parámetro. A través de esta función se podría agregar más de un servo a mover, permitiendo efectuar un movimiento en paralelo.

```
void addInput(int servo, int angle){
  servos[posArray]=servo;
  angles[posArray]=angle;
  posArray++;
}
```

Figura 17 - Función addInput

3.2.4.6. Movimiento de servos en paralelo

La función **setAngleParallel** es la encargada de efectuar los movimientos de los servos, pudiendo, en el caso de que se haya cargado más de un servo y ángulo a los arreglos, ejecutarlos paralelamente.

En esta función primero se realiza una comprobación de todos los servos y ángulos cargados, para verificar que los ángulos se encuentren entre el mínimo y máximo permitido para el servo correspondiente. Dicha funcionalidad se puede apreciar en la siguiente figura.

```
boolean isAngleValid(){
  for(int i=0;i<posArray;i++){
    if ((angles[i] < anguloMin[servos[i]]) || (angles[i] > anguloMax[servos[i]])){
      return false;
    }
  }
  return true;
}
```

Figura 18 - Función de validación de ángulos

En el caso de que dicha verificación sea correcta se procede a comprobar si el movimiento será aumentando o disminuyendo los grados en comparación con el ángulo actual del servo. Esto se realiza por cada servo guardando dicha información en un arreglo y actualizando su ángulo actual en un grado como se puede ver en la Figura 19.

```
for(i=0;i<posArray;i++){
  if (lastVal[servos[i]] > angles[i])
    incV[i] = -1;
  else
    incV[i] = 1;
  lastVal[servos[i]]+=incV[i];
}
```

Figura 19- For donde se toma la decisión de si el movimiento será ascendente o descendente.

A continuación se ejecuta una porción de código que se irá repitiendo hasta que todos los servos alcancen el ángulo especificado (ver Figura 20).

```
while(notFinish){
  if (servos[j] != -1 && j<posArray){
    i = lastVal[servos[j]];
    if ((incV[j] == 1 && i <= angles[j] ) || (incV[j] == -1 && i >= angles[j])){
      pulselen = map(i, 0, 180, servoMin[servos[j]], servoMax[servos[j]]);
      pwm.setPWM(servos[j], 0, pulselen);
      delay(INCRDEL);
      i += incV[j]*speed;
      lastVal[servos[j]] = i;
    } else {
      lastVal[servos[j]] = angles[j];
      if (servos[j] < 3 || servos[j] > 12)
        limitarBrazos();
      servos[j] = -1;
      cantComplet++;
      if (cantComplet == posArray){
        notFinish = false;
      }
    }
  }else{
    if (j == posArray){
      j=-1;
    }
  }
  j++;
}
```

Figura 20 - Función que se encarga de llevar un servo determinado a un ángulo previamente especificado.

Para que cada servo alcance su ángulo especificado, dicha función va incrementando o disminuyendo cada servo en un grado multiplicado por un factor que corresponde a la velocidad a la cual se quiere que se realice el movimiento. Esto permite que, a los ojos de una persona, los movimientos de los servos parezcan simultáneos.

Para mover el servo a través del controlador PCA9685, se hace uso de una función map, como se puede ver en la Figura 21, que establece el ancho del pulso. La misma recibe el nuevo ángulo en grados, en este caso i, el mínimo en grados, 0, el máximo en grados, 180, el mínimo en número de cuentas, y el máximo en número de cuentas, cuyos valores ya se encuentran preestablecidos. Dicha función mapea entre sus mínimos y máximos para así obtener el número de cuentas a utilizar.

Luego de obtener el ancho de pulso, se setea el PWM con el número de servo a mover, el UP del ciclo de trabajo, en este caso 0, y el DOWN del mismo, en este caso el ancho del pulso.

Por último se realiza un delay de 20ms para que el servo ejecute el movimiento y se incremente o disminuya dependiendo el ángulo para la siguiente ejecución.

```
pulselen = map(i, 0, 180, servoMin[servos[j]], servoMax[servos[j]]);
pwm.setPWM(servos[j], 0, pulselen);
delay(INCRDEL);
```

Figura 21- Función que calcula el ancho de pulso en base a un ángulo y se lo setea al servo a mover.

Cabe mencionar que se implementó una función **limitarBrazos()** para validar si se está intentando realizar una combinación de movimientos que podría dañar la integridad del robot, por ejemplo, realizar una flexión del codo cuando el brazo del robot se encuentra en la posición de home. Para una ilustración gráfica se puede observar la Figura 22.

La misma se evalúa cada vez que se realiza un movimiento de los servos del tren superior del robot, ya que el resto de las combinaciones peligrosas detectadas ya se encuentran limitadas por los rangos mínimos y máximos. Su funcionamiento se basa en setear los distintos ángulos Min y Max de cada servo del tren superior dependiendo de la posición del resto de los servos.

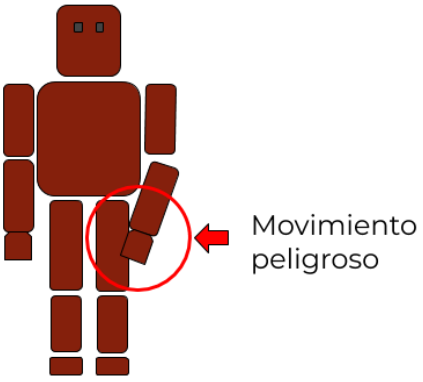

Figura 22 - Esquema de un movimiento peligroso para el robot

3.2.4.7. Movimientos predeterminados

Para la creación de los movimientos predeterminados se hizo uso de una funcionalidad implementada para ejecutar movimientos en diferentes servos en paralelo (setAngleParallel).

Se generaron 10 movimientos en total:

- Estabilizar
- Saludar
- Enojado
- Dar la mano
- Dab
- Onda
- Arigato
- Chuchuwa
- Sentadillas
- Caminar (Figura 23)

Como ejemplo de la creación y funcionamiento de un movimiento predeterminado, el cual es una serie de movimientos individuales, se usará la función caminar. La estructura para armar un solo movimiento, ya sea mover uno o varios servos, es la siguiente:

```
cleanInput();
addInput(SERVO, ANGULO);
.
.
addInput(SERVO, ANGULO);
setAngleParallel()
```

Por ejemplo, en la función caminar, para dar el paso izquierdo se mueven los servos correspondientes a los muslos, los tobillos y los pies de ambas piernas. Luego para finalizar la caminata vuelve cada servo a su posición inicial.

La mecánica para caminar es básicamente levantar el muslo correspondiente al lado que vas a dar el paso, estirar el tobillo del mismo lado y acomodar el pie a la posición inicial. Mientras tanto en la otra pierna se vuelve el muslo a su posición inicial, se inclina el pie hacia adentro para poder separar del suelo la pierna contraria y como último paso se flexiona el tobillo para romper la inercia y adelantar al robot. La misma mecánica aplica para el paso siguiente.

Si bien sería natural que el movimiento de la caminata agregara también movilidad en los brazos, se decidió eliminarlos para no exigir tanta potencia al controlador de servos.

```
/************************************** Caminar ***************************************/
void caminar(){
  for(int i=0;i<pasos;i++){

    //Paso Izquierdo
    cleanInputs();
    addInput(RTHIGH,posHome[RTHIGH]);
    addInput(LTHIGH, 135);
    addInput(LANKLE, 85);
    addInput(RANKLE, 120);
    addInput(RFOOT, 90);
    addInput(LFOOT,posHome[LFOOT]);
    setAngleParallel();

    //Paso Derecho
    cleanInputs();
    addInput(LTHIGH,posHome[LTHIGH]);
    addInput(RTHIGH, 15);
    addInput(LANKLE, 35);
    addInput(RANKLE, 80);
    addInput(LFOOT, 75);
    addInput(RFOOT,posHome[RFOOT]);
    setAngleParallel();
  }

  cleanInputs();
  addInput(LFOOT,posHome[LFOOT]);
  addInput(RTHIGH,posHome[RTHIGH]);
  addInput(RFOOT,posHome[RFOOT]);
  addInput(LTHIGH,posHome[LTHIGH]);
  addInput(RANKLE,posHome[RANKLE]);
  addInput(LANKLE,posHome[LANKLE]);
  setAngleParallel();
}
```

Figura 23 - Código de la función Caminar

4. Guía de Instalación: Proyecto y Ambiente de Desarrollo

4.1. Ambiente de desarrollo

4.1.1. Instalación del IDE de Arduino

En este proyecto se usó hizo del IDE de Arduino para la programación de la placa y del ESP8266 en su correspondiente lenguaje basado en C. Este IDE se puede descargar desde su página oficial:

https://www.arduino.cc/en/Main/Software

4.1.2. Descarga del código fuente

Todo el código correspondiente a dicho proyecto se encuentra alojado en un repositorio en Github que podemos descargar mediante el siguiente link:

https://github.com/proyectoHumanoideTDP2/ProyectoHumanoide

También puede realizarse la clonación del repositorio mediante la siguiente URL:

https://github.com/proyectoHumanoideTDP2/ProyectoHumanoide.git

4.1.3. Carga del Arduino

Para cargar el programa en el Arduino, se debe conectar el mismo a la PC con el cable USB A-B. Luego, teniendo los programas abiertos a subir en el IDE, presione el botón “Subir” (Figura 24).

Figura 24 - Botón para la carga del código desde el IDE de Arduino

Para realizar la carga, los pines RX y TX del Arduino deben estar desconectados y para esto se debe desconectar al ESP8266 de la placa. También se recomienda desconectar el PCA9685 para evitar que se muevan los servos.

Cuando finalice la carga del programa en el Arduino aparecerá el mensaje que se puede apreciar en la Figura 25.

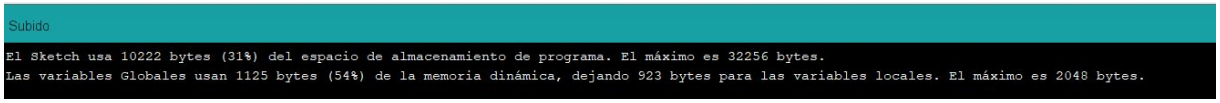

Figura 25 - Mensaje de finalización de subida del programa al Arduino

4.1.4. Carga de la Placa ESP8266 al IDE de Arduino

Para realizar la carga de la placa ESP8266 al IDE de Arduino, primero debemos dirigirnos a Archivo > Preferencias y en la sección Gestor de URLs adicionales de tarjetas agregamos la siguiente línea, tal como podemos ver también en la Figura 26:

https://arduino.esp8266.com/stable/package_esp8266com_index.json

Figura 26 - Gestor de URLS con la librería ESP8226 cargada

El siguiente paso es dirigirnos a Herramientas > Administrador de bibliotecas y buscar la librería correspondiente al ESP8266 para su instalación. En la Figura 27 podemos ver la librería que debemos seleccionar.

Figura 27 - Librería ESP8266

IMPORTANTE: Es necesario que para la instalación de la librería se elija la versión 2.5.0 y no una superior.

4.1.5. Carga del ESP8266

Para iniciar la carga de un programa al ESP8266 primero debemos verificar que la versión de la librería del ESP8266 sea la 2.5.0. Una vez verificado dicho paso debemos continuar con la correcta conexión de los cables para que no se produzca un error. Se recomienda seguir los siguientes pasos:

1. Conectar el pin Reset del Arduino a GND
2. Conectar a GND los pines GND y GPIO0 del ESP8266
3. Conectar a 3,3V los pines VCC y CH_PD del ESP8266
4. Conectar el Rx del Arduino al Rx del ESP8266
5. Conectar Tx del Arduino al Tx del ESP8266

Una vez realizada dicha conexión procedemos a dejar la configuración del ESP8266 por defecto como se puede ver en la Figura 28. Luego, podremos ya subir nuestro código.

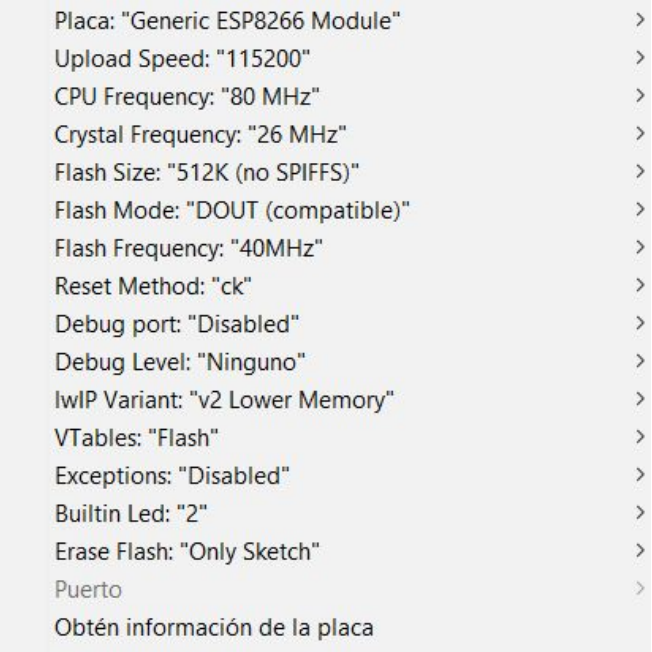

Figura 28- Configuración por defecto del ESP8266

Una vez que se haya realizado la carga del programa (se verá el mensaje de la Figura 29) procedemos a desenchufar el cable USB, intercambiar entre ellos los cables TX y RX del Arduino y por último retirar el GND de los pines GPIO0 y Reset del Arduino. La próxima vez que conectemos el ESP8266 al robot debemos esperar unos minutos para que la red se levante nuevamente.

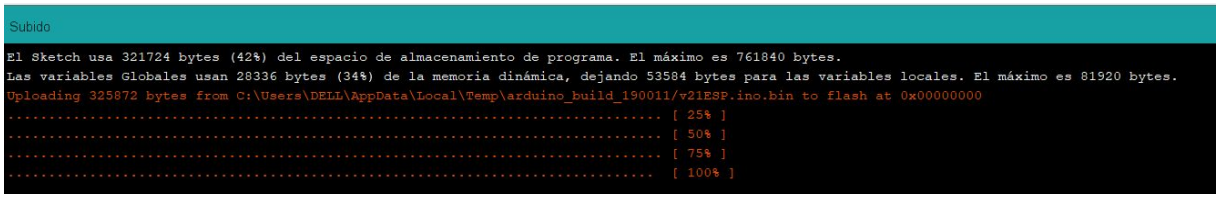

Figura 29 - Mensaje de carga exitosa del ESP8266

4.1.6. Instalación y uso del plugin FS.h para la carga de Bootstrap

La instalación de dicho plugin nos permitirá realizar la carga de diferentes tipos de archivos al ESP8266, desde texto plano hasta un código realizado en JavaScript para poder ser ejecutado en la Web.

Para comenzar debemos al siguiente link y descargar el plugin:

- https://github.com/esp8266/arduino-esp8266fs-plugin/releases/tag/02.0.

Esto nos descargara un zip que deben descomprimir en la carpeta tools que generalmente se encuentra en la siguiente dirección:

- C:\Users\usuario\Documents\Arduino

En el supuesto caso que no exista dicha carpeta se deberá proceder a crearla.

Para el uso de dicha librería se deberá crear una carpeta llamada "**data**" en la correspondiente carpeta del proyecto donde se quieran cargar los archivos. Los archivos deberán ser colocados dentro de esta carpeta para realizar su carga. En este ejemplo haremos uso de los archivos compilados correspondientes a la librería de Bootstrap para su uso.

Para la descarga de Bootstrap podremos ir al siguiente link: https://getbootstrap.com/. esto nos descargara un archivo .zip que al descomprimirse generará dos carpetas:

- CSS: Contiene los archivos relacionados a los estilos de bootstrap y sus componentes.
- JS: Contiene diferentes archivos escritos en JavaScript que permiten darle diferentes funciones a los elementos de BootStrap.

Dado que únicamente se queremos utilizar los componentes y estilos de Bootstrap cargaremos el archivo que está dentro de la carpeta "**CSS**" llamado "**bootstrap.min.css**".

Una vez preparada la carpeta "**data**" con sus correspondientes archivos, procedemos a conectar el Arduino a nuestra computadora y abrimos el IDE Arduino con el correspondiente proyecto.

En la sección de herramientas se tendrá de configurar la capacidad de la carga del ESP8266, en este caso seleccionaremos la opción "1M (256 SPIFFS)", tal y como podemos ver en la Figura 30.

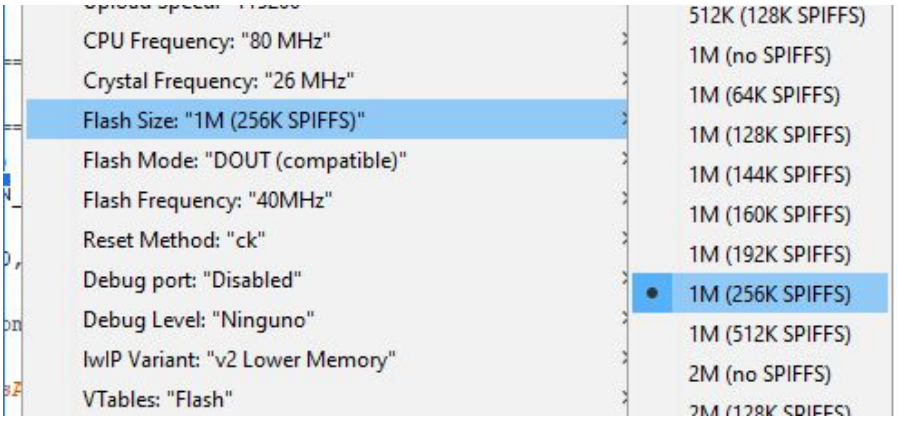

Figura 30 - Configuración para la carga de archivos en el ESP8266

Como siguiente paso debemos configurar la carga de los archivos en el código de nuestro programa para poder utilizarlos. Debemos utilizar los comandos que se muestran en la Figura 31 donde en la función bootstrap carga el archivo en una variable para luego indicarle al server que tipo de archivo será (en este caso una hoja de estilos css). Luego, con las siguientes funciones, se monta el sistema de ficheros.

```
#include <FS.h>

void bootstrap()
{
  File file = SPIFFS.open("/bootstrap.min.css", "r");
  size_t sent = server.streamFile(file, "text/css");
}

server.on("/bootstrap.min.css", bootstrap);

SPIFFS.begin();
```

Figura 31 - Serie de comandos para la configuración y carga de archivos

En la Figura 32 podemos el último donde se le informa a nuestra página HTML donde se encuentran los archivos cargados para su utilización.

Figura 32 - Línea que linkea los estilos con la hoja HTML

A partir de este momento podremos hacer uso de todos los elementos correspondientes a la librería de Bootstrap, como por ejemplo, sus botones para facilitar el maquetado y armado de la Web.

4.2. Proyecto

4.2.1. Instructivo para el uso del robot humanoide

1) Conectar la fuente de alimentación a la red eléctrica, verificando que la misma se encuentre apagada.
2) Conectar el cable de alimentación del robot a la fuente.
3) Acomodar al robot en una posición estable. Sería conveniente sostenerlo en el aire para llevar a cabo el siguiente pasos
4) Encender la fuente, cambiando la posición del botón ON/OFF. Automáticamente el robot se debería mover, estabilizando cada uno de sus servos.
5) Parar al robot en una superficie plana y libre de obstáculos.
6) Tomar algún dispositivo con conexión a Internet (preferentemente una PC o celular), abrir la configuración de Wi-Fi y esperar que aparezca la red con el nombre “Bhasky”
7) Conectarse a la red introduciendo la password 12345678.
8) Una vez conectado a la red, abrir un navegador web e introducir la siguiente dirección IP: 10.10.10.10
9) Cuando aparezca el sitio con el nombre “Bhasky”, ya puede interactuar con el robot a través de su interfaz web.
    - Si desea ver al robot realizar sus movimientos predeterminados, tocar el botón que corresponde al movimiento que se quiera realizar.
    - Si desea ejecutar sus propios movimientos, ingresar el número de la articulación que desea mover + el ángulo correspondiente a la posición deseada, procurando que el mismo se encuentre dentro del rango de movimiento predeterminado.
    - Si desea que el robot camine, seleccione el número de pasos correspondiente y presione el botón de Caminar. Por defecto, dará 4 pasos.
10) Para apagar el robot, sosténgalo para evitar una caída. Luego, apagar la fuente de alimentación, cambiando de posición el botón ON/OFF.

5. Problemas y Soluciones

En la siguiente sección se describirán los incidentes que se tuvieron durante el desarrollo del proyecto y sus soluciones:

5.1. Cargar programa al ESP8266

En un principio, se presentaron inconvenientes al intentar flashear el ESP8266, con lo cual no se podía cargar el nuevo programa. Al intentar cargar el nuevo sketch mostraba un mensaje de error. Luego, cuando finalmente se pudo cargar el sketch, no iniciaba la red WiFi creada por el ESP8266, lo cual no permitía controlar al robot inalámbricamente. Como solución temporal probamos las funcionalidades del robot conectando el mismo a la PC y enviado las peticiones a través del monitor serie del Arduino. Para solucionar el problema definitivamente se usó una versión anterior de las librerías del ESP8266, la 2.5.0 con las configuraciones por defecto, para la carga del mismo.

5.2. Cargar Bootstrap al ESP8266

También se presentaron inconvenientes para cargar correctamente Bootstrap junto todos sus componentes en el ESP8266. El problema era causado por el escaso espacio de almacenamiento que este presentaba. Para poder utilizar Bootstrap la solución que se encontró fue cargar solamente el archivo css, dejando de lado los elementos relacionado a JavaScript.

5.3. Corriente durante la función caminar

Antes de la implementación del SHIELD, para ejecutar la caminata era necesario desconectar los servos de uno de los brazos ya que la placa controladora de servos no llegaba a entregar la potencia necesaria para activar los servos indicados y mantener aquellos que no se encontraban en movimiento. La placa aumentaba su temperatura y fallaba. Luego de la conexión del SHIELD, este problema se solucionó, ya que conectando la placa a través del SHIELD con cables adecuados para la alimentación, no se observan las fallas mencionadas. Es decir ya no existe necesidad de conectar y desconectar ningún servo para ningún movimiento.

5.4. Brazos en la función caminar

Sin embargo, para la función caminar, se eliminó el movimiento de los brazos debido a que la potencia entregada por la fuente no es suficiente para movilizar todos los servos involucrados en la caminata simultáneamente y, al añadir estos movimientos, la placa controladora deja de funcionar después de un par de pasos.

5.5. Placa controladora de servos

Por la limitación de 16 servos dada por la placa controladora, se decidió desconectar el servo que controla la cabeza del robot. Para solucionar este problema, a futuro se podría:

- Investigar de qué forma generar una señal PWM como la que entrega el controlador de servos con el Arduino e implementar el control del servo por fuera de la placa.
- Adquirir una nueva placa controladora con espacio para conectar más servos, lo cual también podría solucionar los problemas de falta de potencia descriptos anteriormente.

5.6. Trabajo futuro: Controlador de 32 servos Lobot

Dado los problemas de energía generados al intentar controlar los servos mediante el PCA9685 actual, que no posee el espacio ni la potencia suficiente alimentar a todos los servos, se encontró la posibilidad de reemplazarlo por el controlador de 32 servos de la Figura 33.

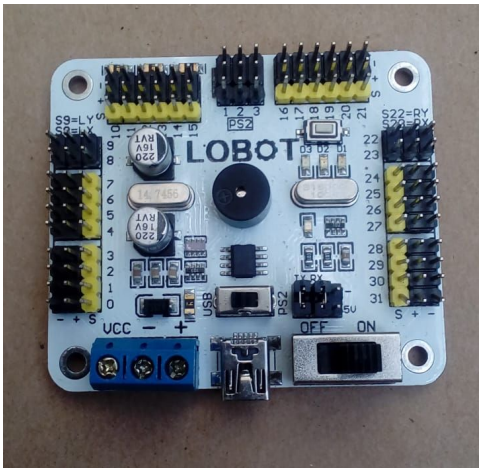

Figura 33 - Controlador de 32 servos Lobot

Sin embargo, no fue posible encontrar la documentación de este controlador de servos específico, pero sí se encontró la documentación de otros controladores con características similares.

5.6.1. Opción 1

El siguiente controlador a especificar podemos verlo en la Figura 34. Posee entre sus características:

- Protección contra sobretensión
- Memoria de alta capacidad de 16M
- Soporte para depuración gráfica en línea
- Comunicación serie
- Módulo bluetooth
- Alarma de baja tensión

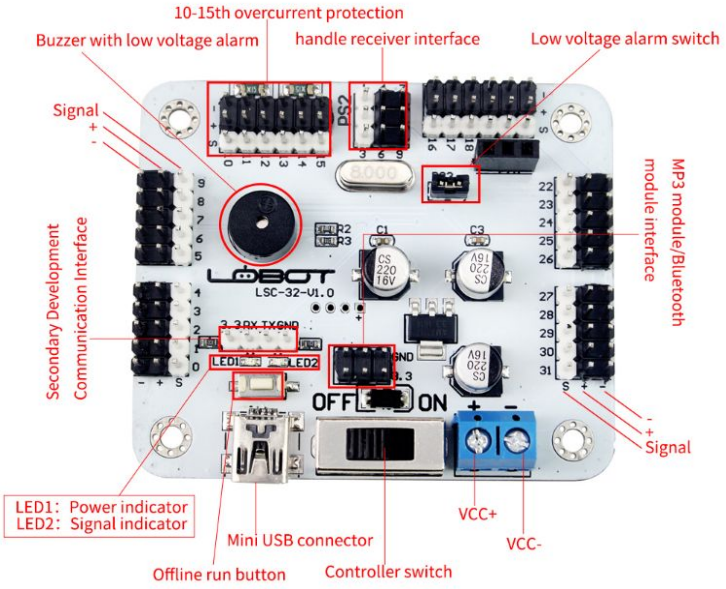

Figura 34 - Componentes del controlador de servos

Estos controladores poseen una interfaz gráfica como programa que nos permite controlar los 32 servos a la vez, tal y como se puede ver en la Figura 35.

Figura 35 - Software para control de servos

En la figura 36 se puede apreciar el diagrama de conexión correspondiente a dicho controlador de servos, que requiere un voltaje de 3.3 V.

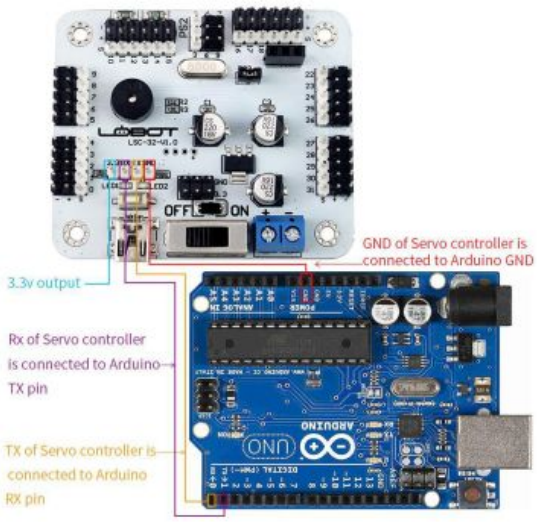

Figura 36 - Diagrama de conexión entre el controlador de Servos y el Arduino

5.6.2. Opción 2

El otro controlador de servos del cual está disponible su documentación es el mostrado en la Figura 37, el cual tiene una distribución de los componentes similar al primero presentado.

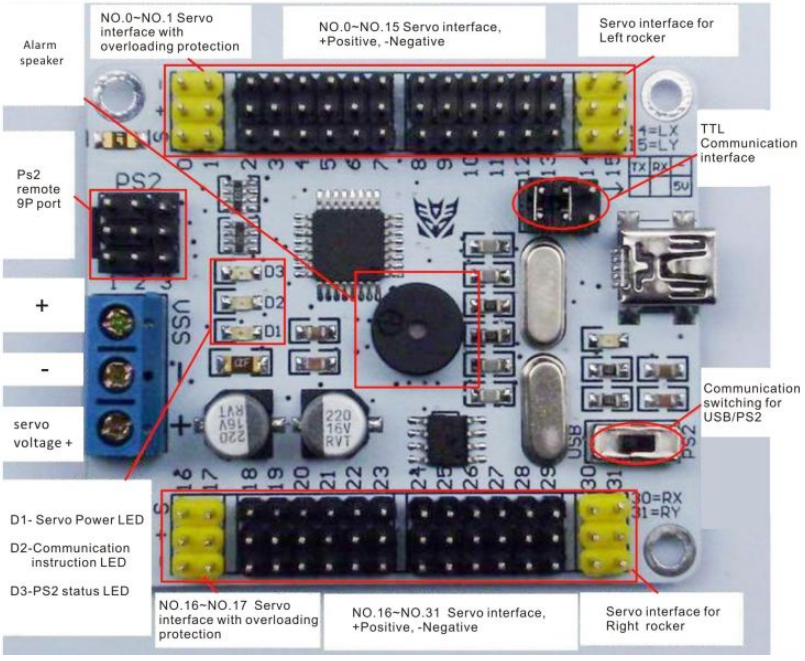

Figura 37 - Controlador de 32 servos

A diferencia del anterior controlador de servos el siguiente requiere una fuente de energía de 5V. Para el control de dichos servos con Arduino se debe establecer una comunicación serial a 115200.

Una vez realizado esto se debe enviar a través de dicha comunicación el siguiente formato de texto para controlar los servos en particular:

#<ch>P<pw>S<spd>..#<ch>P<pw>S<spd>T<time>

Donde:

- ch: Número de servos (0..31)
- pw: Ancho de pulso en microsegundos [500 - 2500]
- spd: Frecuencia de movimiento en us/s , solo tiene sentido para controlar un servo solo
- Time: Los milisegundos que se necesitarán para moverse a la posición designada. (Opcional)

Como se puede apreciar se puede controlar no solo un servo, sino que varios servos a la vez mediante esa sentencia.

Ejemplo: Se mueve el servo 5 con un ancho de pulso de 1600 y 500 ms para realizar dicho movimiento.

Serial.println("#5 P1600 T500");//Servo 5 runs to P1600 with 500MS.

6. Documentación en Formato Gráfico y Video

6.1. Documentación en Formato Gráfico

6.1.1. Fotos del robot

En las siguientes imágenes se muestran las caras anterior y posterior del robot luego de la implementación del SHIELD y el reordenamiento de los cables.

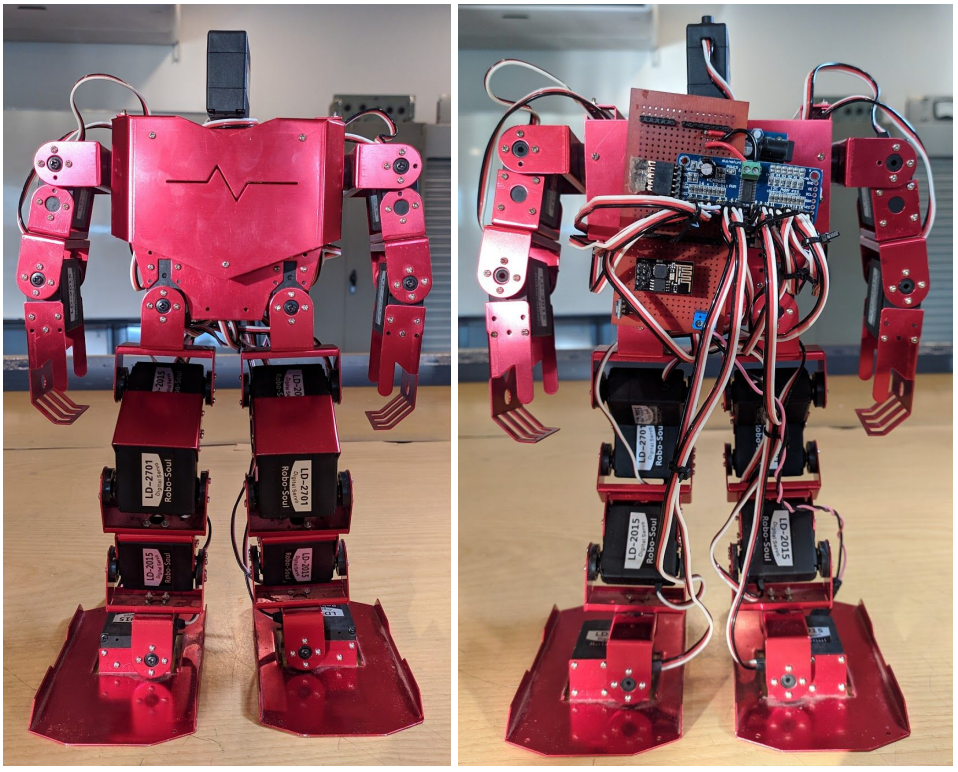

Figura 38 - Foto delantera y trasera del Robot

6.1.2. Rangos de movilidad de servos

En las siguientes imágenes se documentan los rangos de movilidad de cada servo y su posición de estabilidad.

0- Codo izquierdo

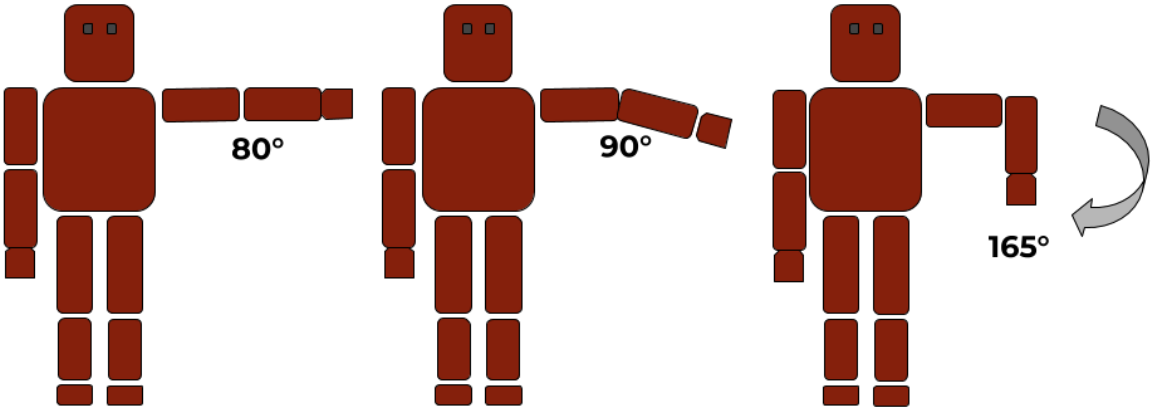

Figura 39 - Rango codo izquierdo

1- Hombro izquierdo Aducción y Abducción

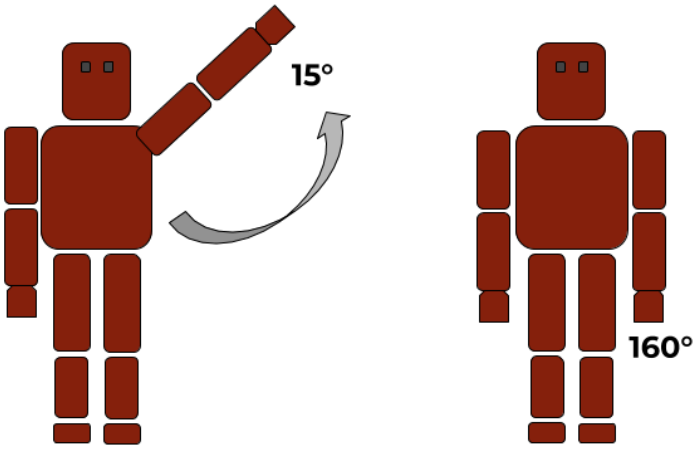

Figura 40 - Rango hombro izquierdo aducción y abducción

2- Hombro izquierdo Flexión y Extensión

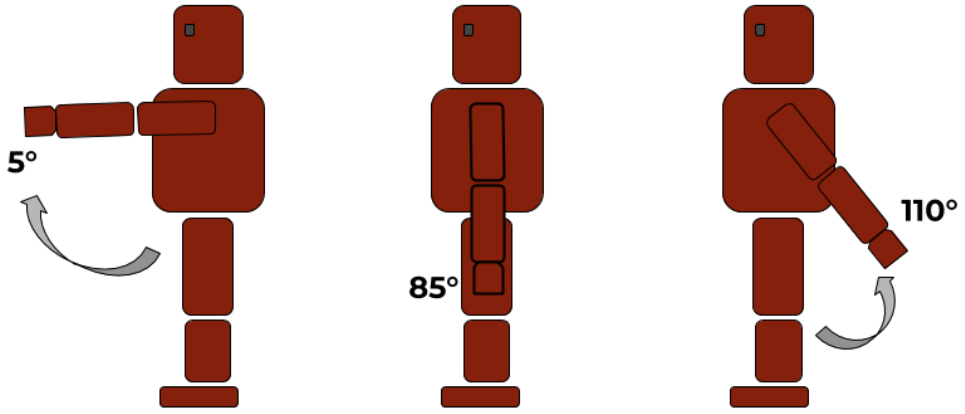

Figura 41 - Rango hombro izquierdo flexión y extensión

3- Cadera izquierda Aducción y Abducción

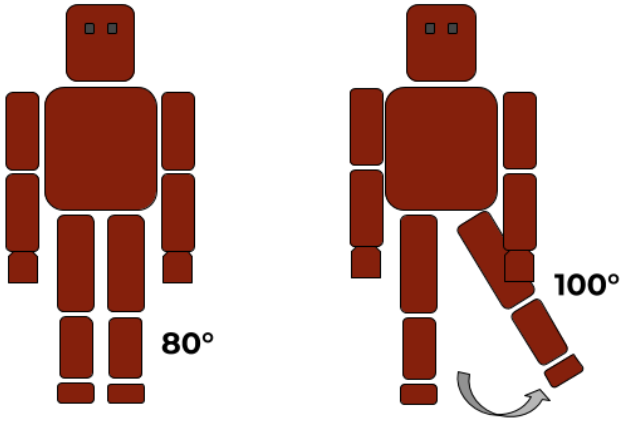

Figura 42 - Rango cadera izquierda aducción y abducción

4- Cadera izquierda Flexión y Extensión

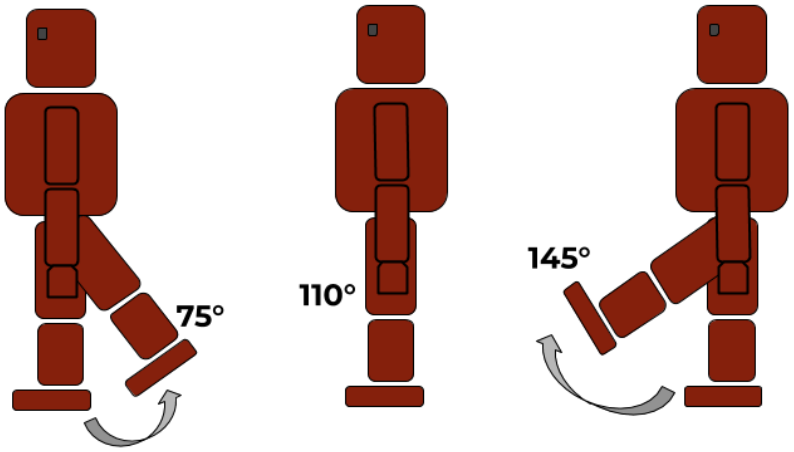

Figura 43 - Rango cadera izquierda flexión y extensión

5- Rodilla izquierda

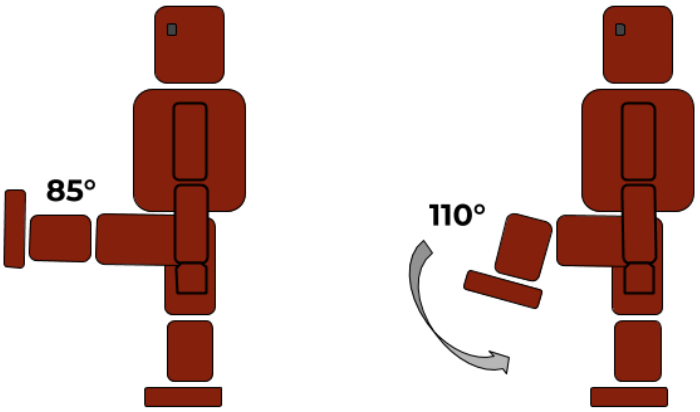

Figura 44 - Rango rodilla izquierda

6- Tobillo izquierdo

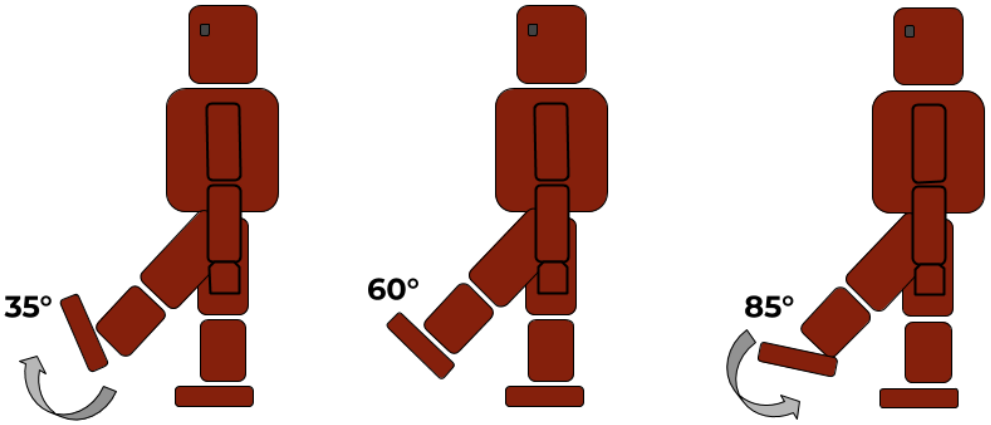

Figura 45 - Rango tobillo izquierdo

7- Pie izquierdo

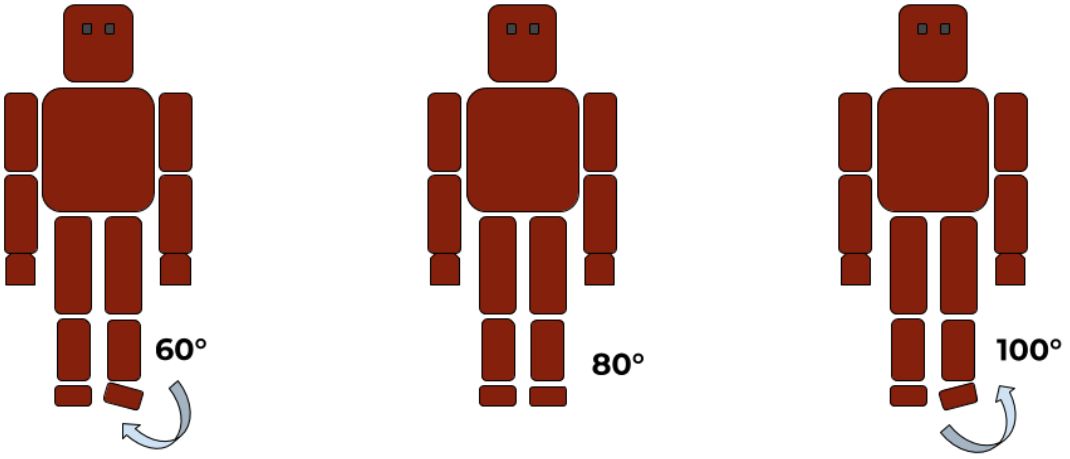

Figura 46 - Rango pie izquierdo

8- Pie derecho

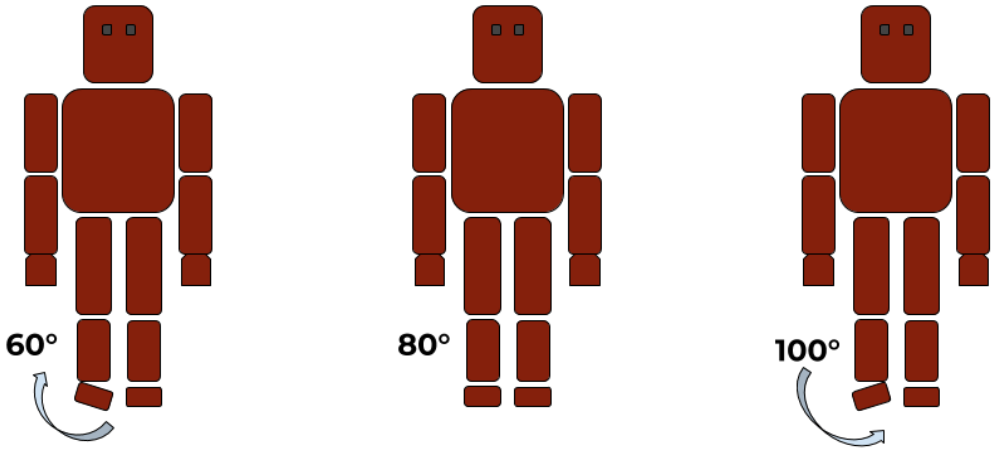

Figura 47 - Rango pie derecho

9- Tobillo derecho

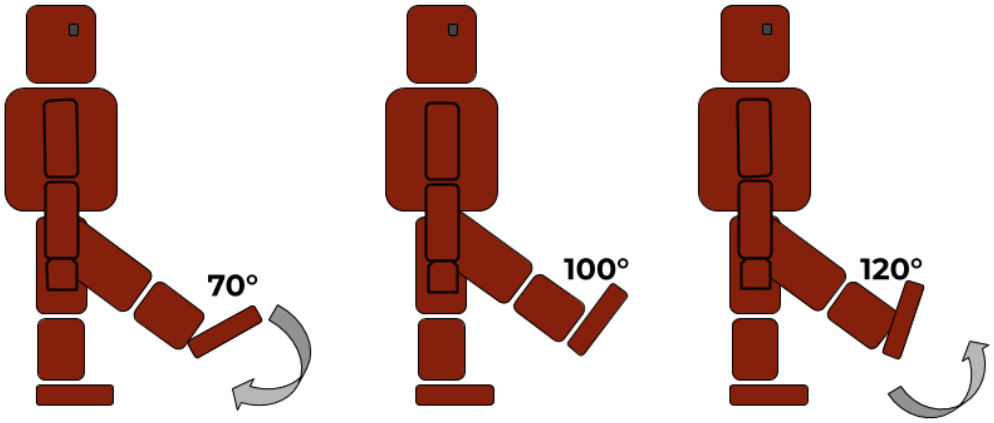

Figura 48 - Rango tobillo derecho

10- Rodilla derecha

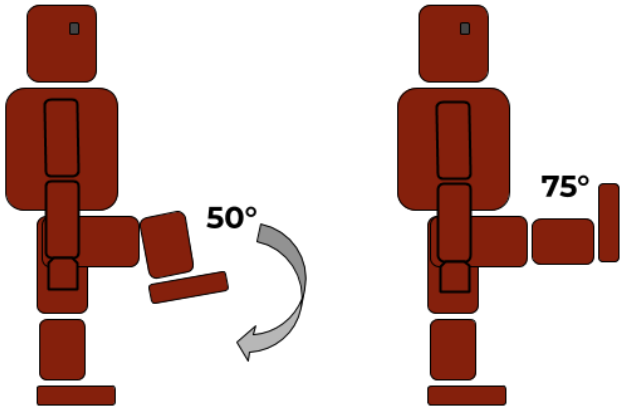

Figura 49 - Rango rodilla derecha

11 - Cadera derecha Flexión y Extensión

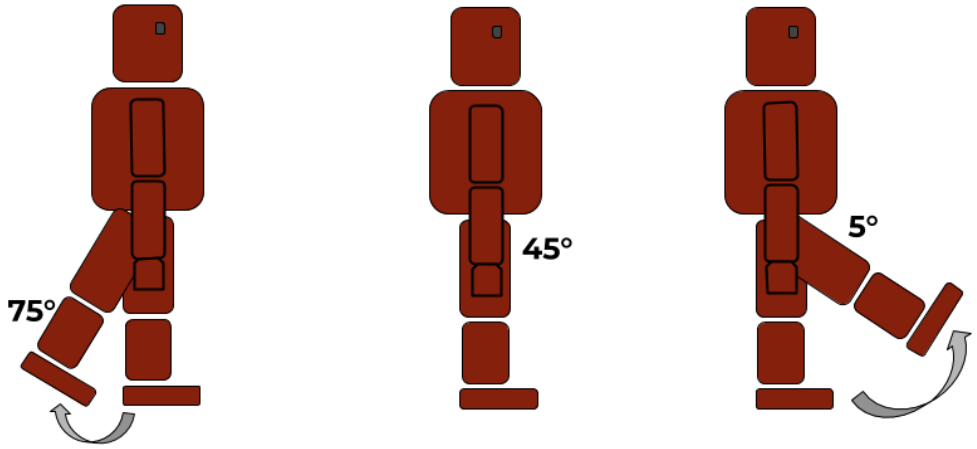

Figura 50 - Rango cadera derecha flexión y extensión

12- Cadera derecha Aducción y Abducción

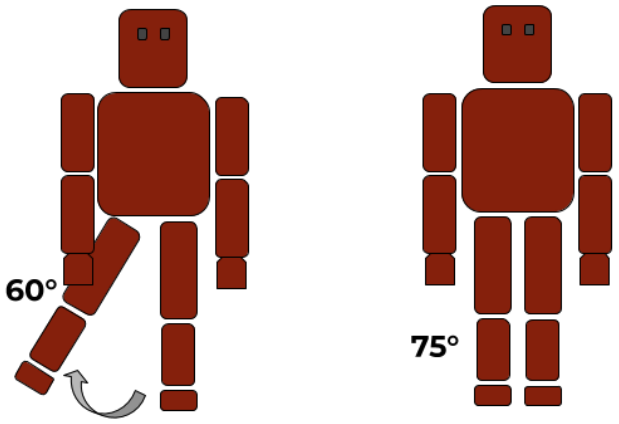

Figura 51 - Rango cadera derecha aducción y abducción

13- Hombro derecho Flexión y Extensión

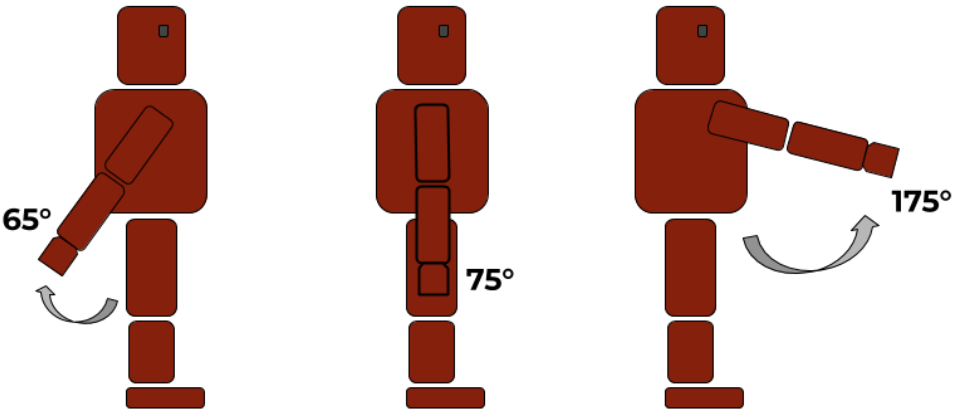

Figura 52 - Rango hombro derecho flexión y extensión

14- Hombro derecho Aducción y Abducción

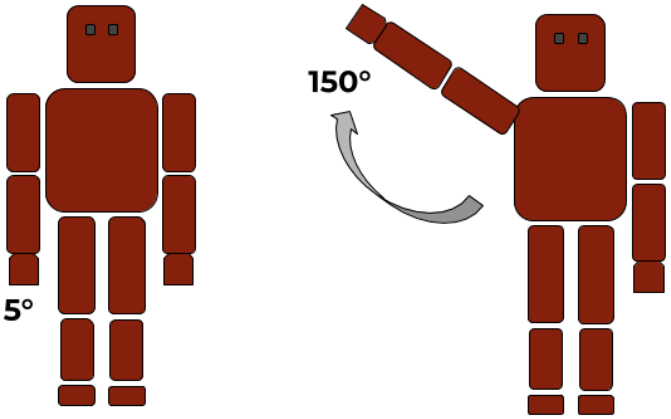

Figura 53 - Rango hombro derecho aducción y abducción

15- Codo derecho

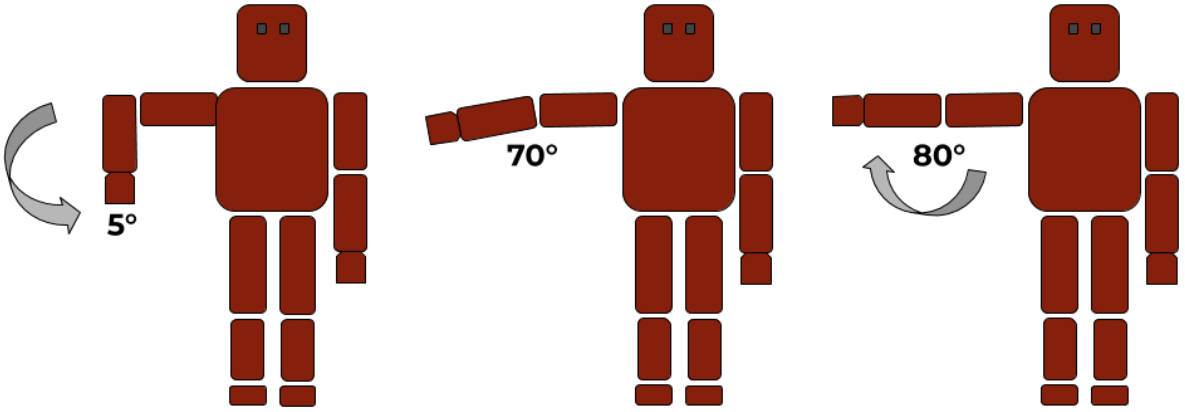

Figura 54 - Rango codo derecho

6.2. Documentación en formato video

6.2.1. Versión 1

Movimientos sin implementación de la función setAngleParallel.

Saludar - https://www.youtube.com/watch?v=B-196Km1Myo

Decir no - https://www.youtube.com/watch?v=KODBUrgBDZY

Dar la mano - https://www.youtube.com/watch?v=Rarzfblq8hQ

Estabilizarse - https://www.youtube.com/watch?v=5CK8cbEDXwc

6.2.2. Versión 2

Movimientos con implementación de la función setAngleParallel.

Saludar - https://youtu.be/eOE6iJLv7x4

Dar la mano - https://youtu.be/S31TXwz8M84

Decir no - https://youtu.be/vaPtBCv6G20

Caminar sin ayuda - https://youtu.be/63J8za4ZaAM

Caminar con ayuda - https://youtu.be/ACoBl4F0-uA

6.2.3. Movimientos Finales

Estabilizar - https://youtu.be/cQyFzmq4ChQ

Saludar - https://youtu.be/XWZDFYwq7A0

Enojado - https://youtu.be/PF08eTQR5JM

Dar la mano - https://youtu.be/x0uOcyyJor0

Dab - https://youtu.be/Nu--3cq8qPQ

Onda - https://youtu.be/96d8P8v2l28

Arigato - https://youtu.be/mD4NBXHdouw

Chuchuwa - https://youtu.be/sRophefr4C4

Sentadillas - https://youtu.be/mAUdlgjsIGg

Caminar mirado de frente - https://youtu.be/BwzAMgDGRUY

Caminar mirado de costado - https://youtu.be/Ja7Aef8Ef1E

6.3. Links útiles

6.3.1 Repositorio de código

https://github.com/proyectoHumanoideTDP2/ProyectoHumanoide (en este repositorio se encuentra todo el código fuente desarrollado hasta el momento).

6.3.2. Bitácora

https://docs.google.com/document/d/17II8vmugr2nzEfr_pMrFTH37RNfLXJ6u-RfyEOKXS0Y/edit?usp=sharing