Universidad Nacional de La Plata

Facultad de Informática

Sistemas de Tiempo Real

2017

Desarrollo de brazo de grúa robótico

Integrantes

- Arcuri, Antonio - 538/8
- Blanco Regojo, Mauricio - 492/0

Índice

1. Propuesta Original del Proyecto

En la propuesta original se planteó la idea de la creación de un brazo robótico, conformado por tres servomotores controlados desde un Arduino Mega a partir de los movimientos realizados en un joystick que se comunicaría mediante bluetooth, o una aplicación de celular para controlar los movimientos, también se utilizará una fuente alimentación externa para alimentar de ser necesarios los servos, y se configuraría el Arduino Mega para comunicarse con un servidor web y enviar la data asociada a la posición de los servos y cuál es el que está siendo utilizado actualmente.

En el Apéndice A se encuentra la propuesta original del proyecto, tal como fue presentada a la cátedra en la elección del proyecto a desarrollar.

2. Correcciones/Cambios de la Propuesta

2.1 Indicadas por la Cátedra

- 3 servos iguales a corta distancia
- Joystick se conecta mediante cables al arduino
- El ESP tiene que ser Access Point
- Partes plásticas => Los servos tienen brazos, si tienen problemas, pueden consultar

2.2 Definidas por el Avance/Disponibilidad

Los cambios que se realizaron a la propuesta original involucran la ausencia de un módulo bluetooth dado que el joystick se conecta mediante cables.

Las partes plásticas fueron reemplazadas por una estructura de madera balsa debido a las complicaciones correspondientes al tiempo para la impresión de dichas partes.

3. Descripción de Hardware y Conexiones

Los siguientes dispositivos son los que fueron utilizados en el proyecto:

- Arduino Mega 2560
- Módulo WIFI ESP8266
- 3 Servomotores CS-51 Hobbico
- Joystick

A continuación, se muestra en la Fig. 1 las conexiones de los diversos componentes que componen el proyecto, conectados mediante una protoboard. Luego, en la Fig. 2, se muestra el diagrama de conexión del Joystick con el Arduino, el joystick se alimenta con 5V externos y utiliza el pin A1 para el mapeo del movimiento desde el eje X, aunque también tiene conectado al pin A2 para el movimiento del eje Y, el cual no fue utilizado, y el pin 3 para la lectura del botón para el cambio de servo a controlar.

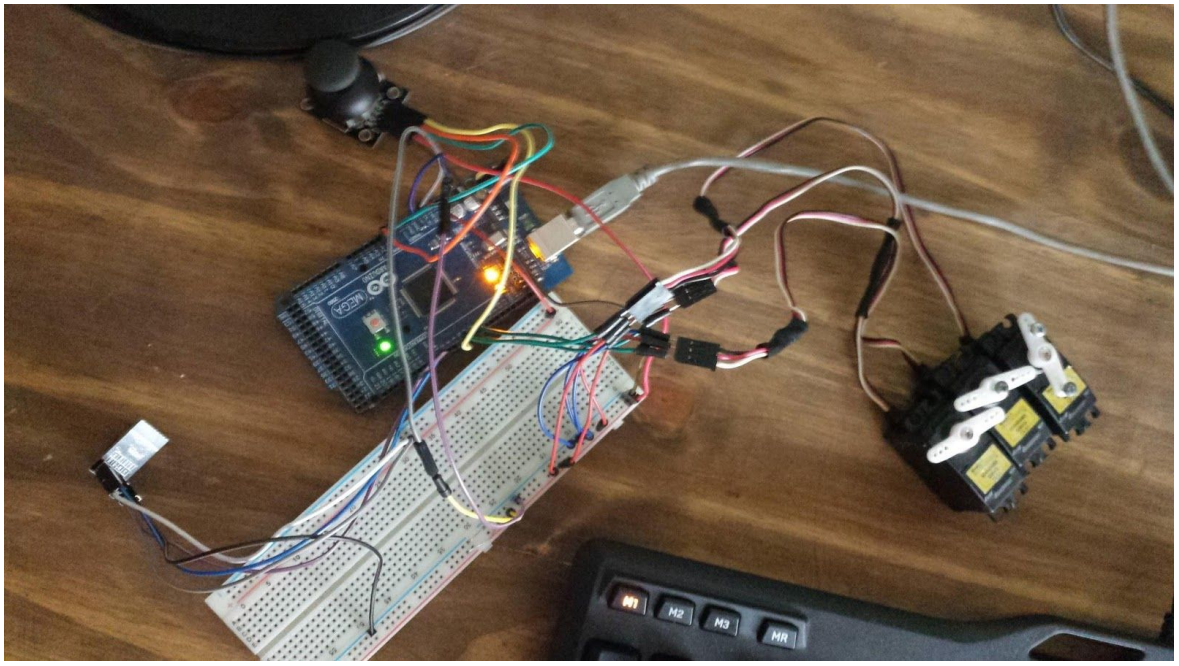

Fig. 1. Conexión de los servos, el joystick y el ESP8266 al Arduino Mega.

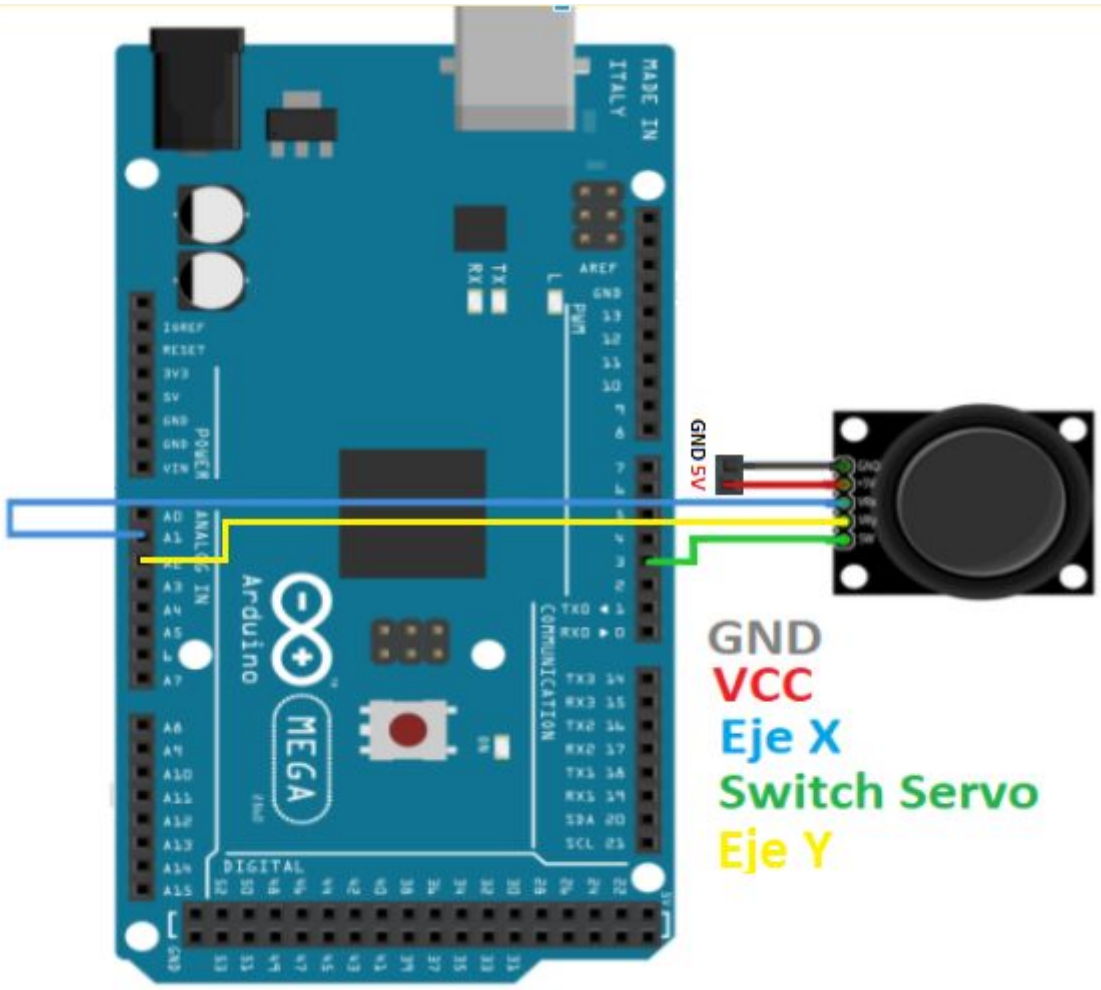

Fig. 2. Conexionado del Arduino Mega con el Joystick

En la Fig. 3, se muestra el diagrama de conexión de los servos con el Arduino, los servos son alimentados con una fuente externa de 5V y cada uno se conecta a uno de los pines 5, 6 y 7 para permitir realizar el cambio de movimiento solicitado a través del joystick.

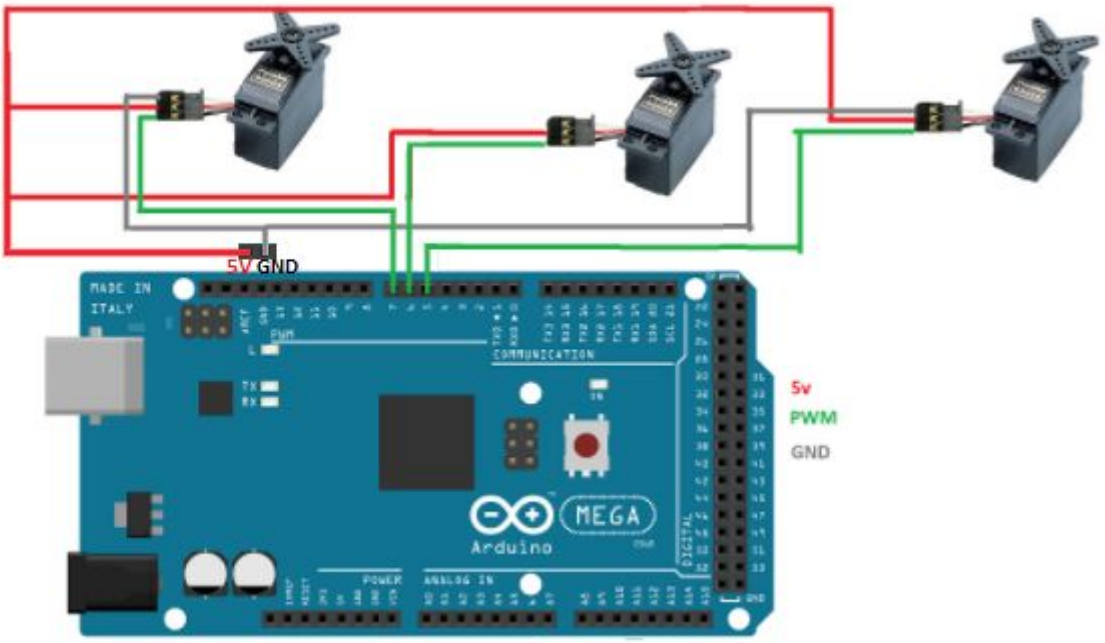

Fig. 3. Conexionado del Arduino Mega con los Servos

En la Fig. 4 se muestra el diagrama de conexión del módulo ESP8266 con el Arduino, el ESP8266 se alimenta con 3.3V provenientes del Arduino Mega y se conectan los pines Tx y Rx del ESP a los pines Rx1 y Tx1 correspondientes del Arduino Mega para permitir la comunicación serie que se hará con un baud rate de 115200 baudios.

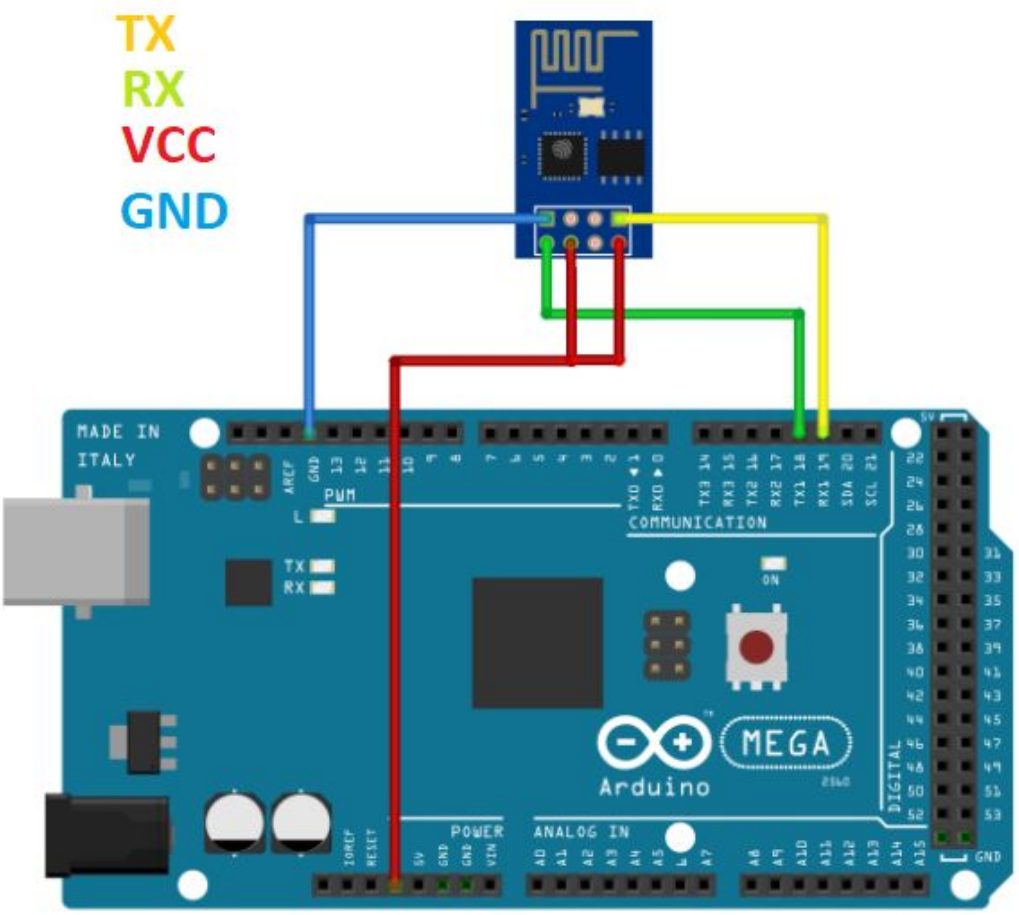

Fig. 4. Conexionado del Arduino Mega con el Módulo ESP8266

Las siguientes figuras representan la placa armada, la pinera A es conectada desde el pin de 3.3V del Arduino hasta el pin de entrada analógica A2, y la pinera B es conectada desde el pin 7 de PWM hasta el pin RX1 de comunicación. El ESP8266 es conectado de modo que el cuerpo del mismo apunta hacia el lugar donde se encuentra posicionado el joystick, el joystick va atornillado a la placa y los servos se conectan de modo que la línea de comunicación de estos quede lo más próxima posible a los pines de PWM.

La alimentación externa se conecta en los lugares detallados en la placa.

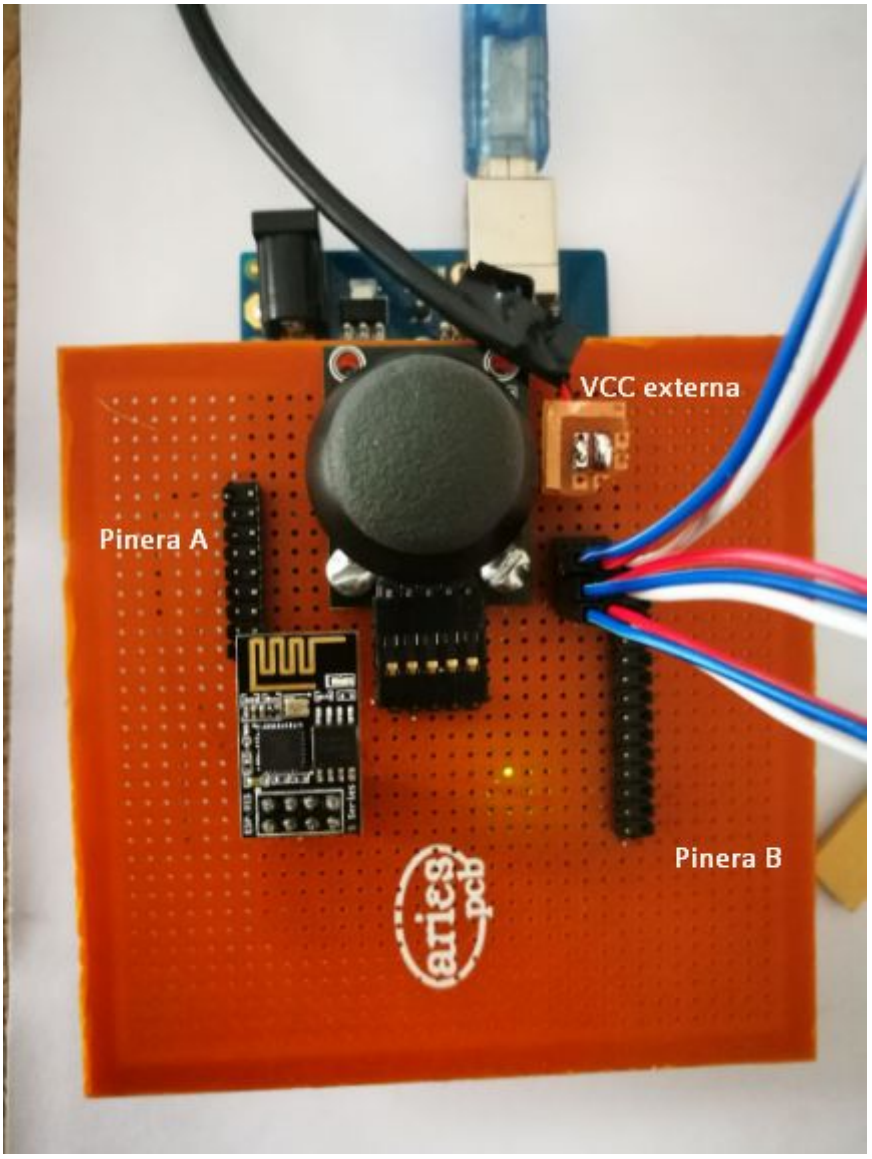

Fig. 5. Fotografía indicando las pineras y la alimentación externa

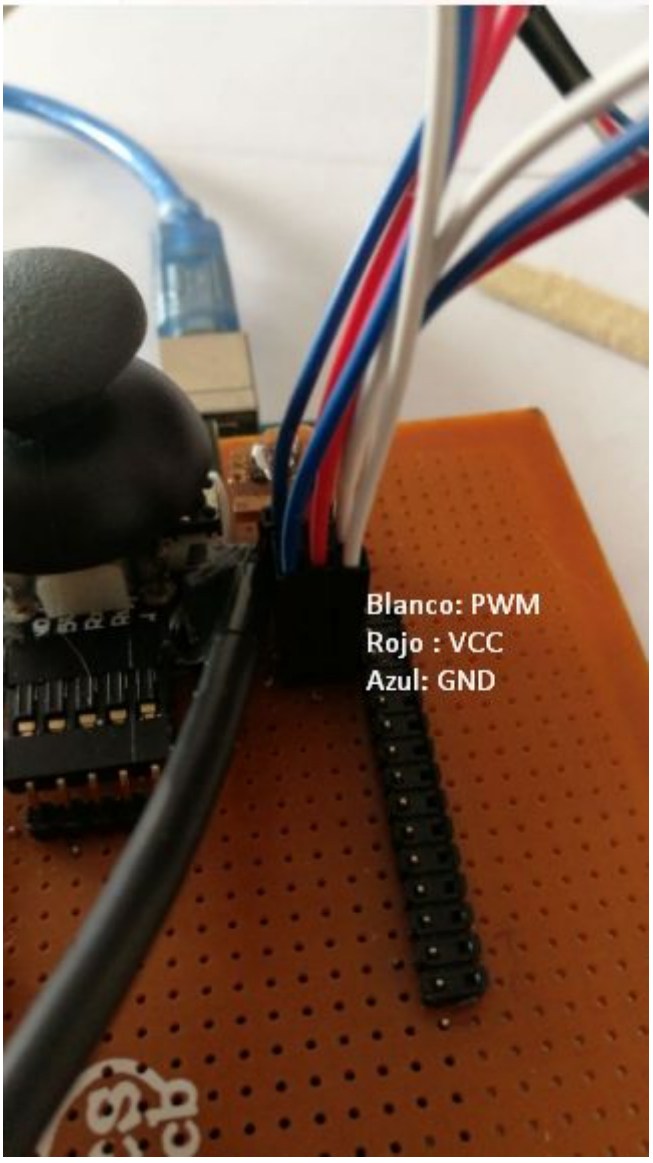
Blanco: PWM
Rojo : VCC
Azul: GND

Fig. 6. Fotografía indicando los colores de los cables usados para los servos

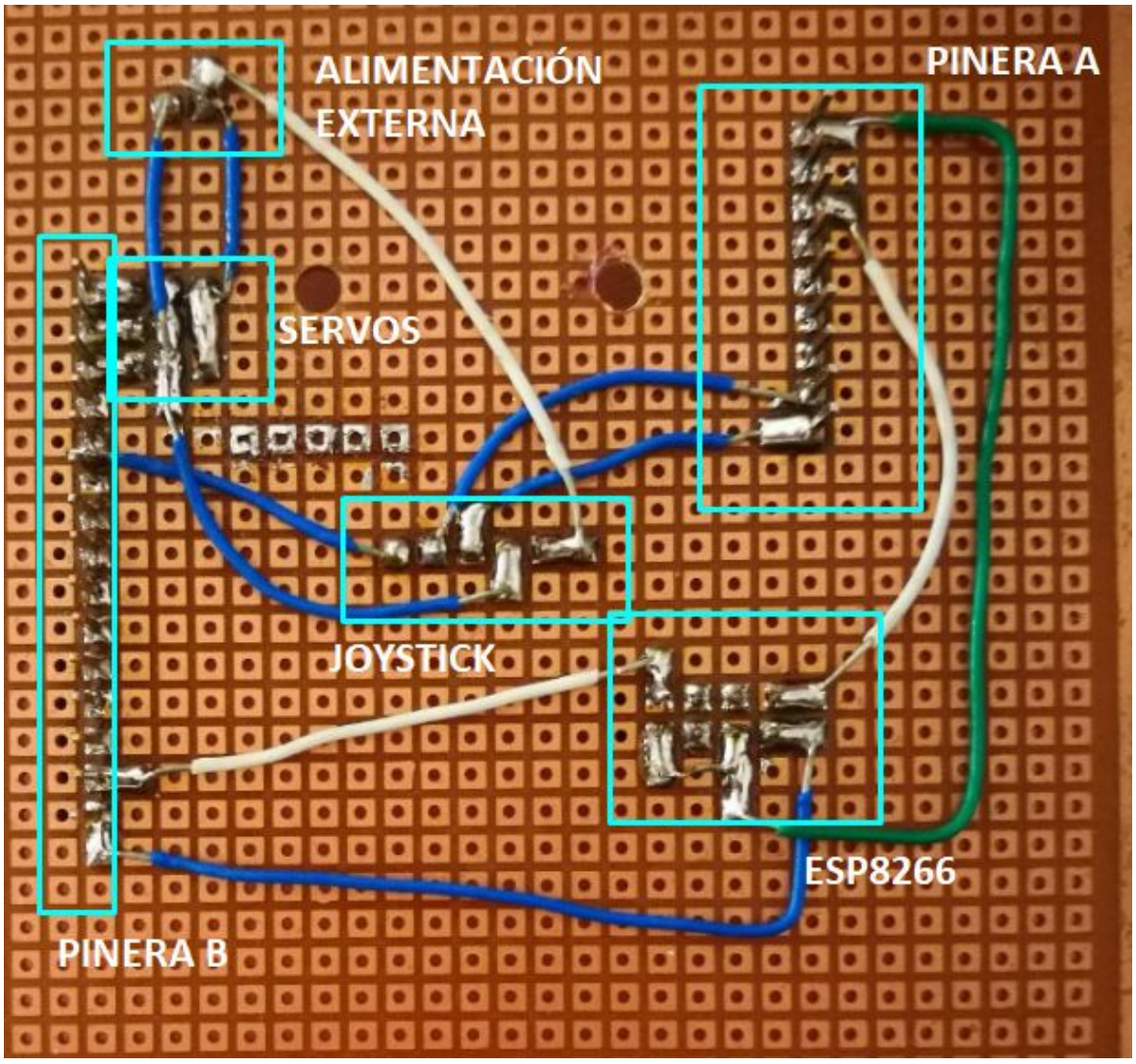

Fig. 7. Fotografía de la PCB indicando las zonas que se explicarán en las siguientes figuras

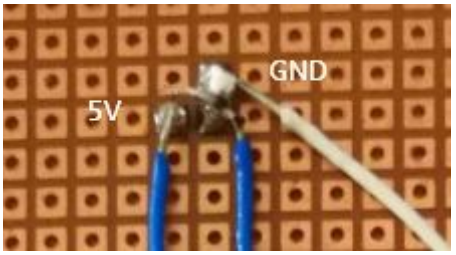

Fig. 8. Fotografía del subsistema de alimentación externa

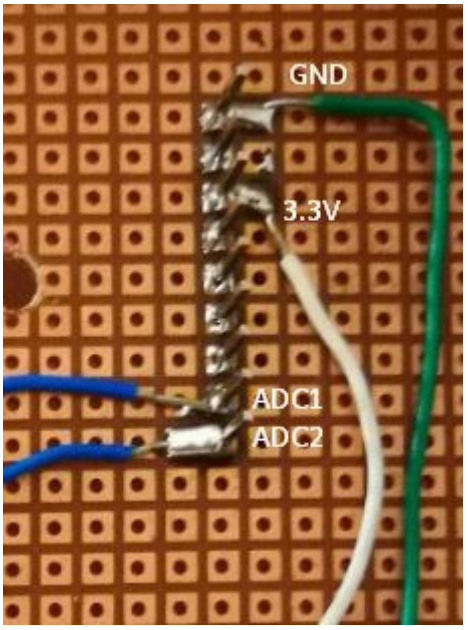

Fig. 9. Fotografía detallando los pines de la pinera A

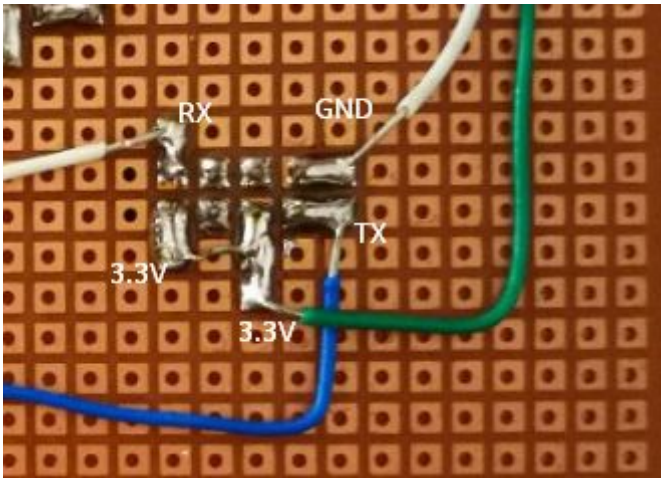

Fig. 10. Fotografía explicativa del subsistema ESP8266

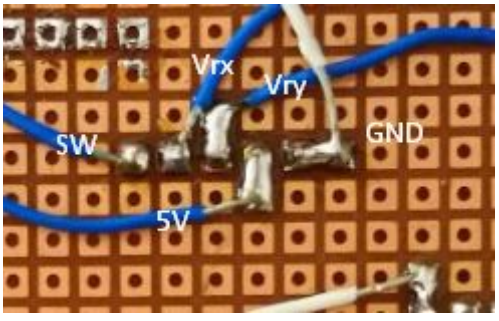

Fig. 11. Subsistema del Joystick

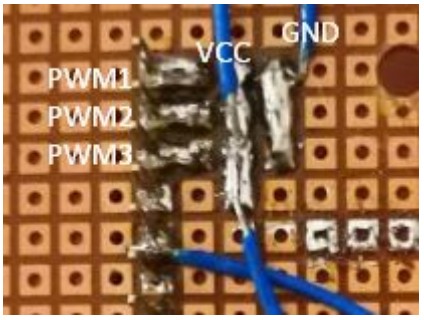

Fig.12. Subsistema de los servos

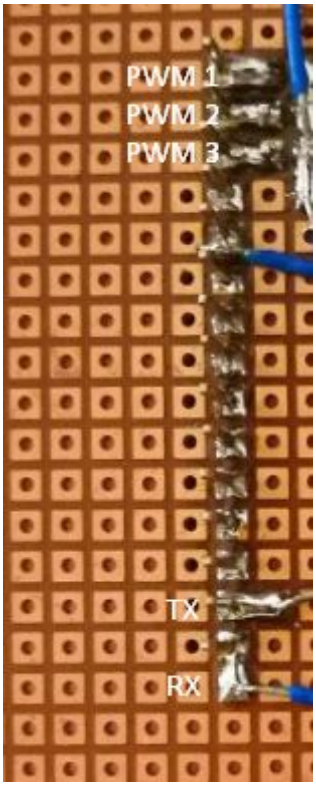

Fig. 13. Fotografía indicando los pines de la pinera B

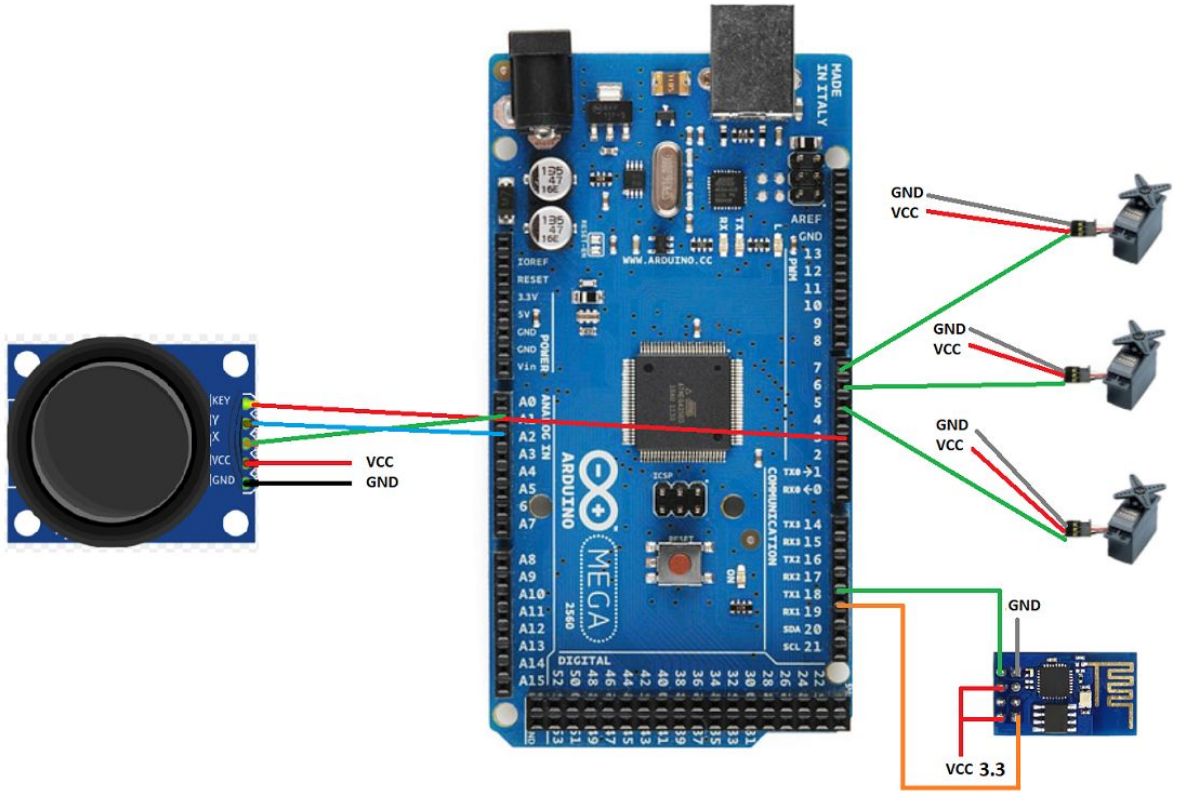

Fig. 14. Esquemático general del proyecto

El link al video de movimiento de los servos a partir de un joystick:

https://www.youtube.com/watch?v=toE5Ox8oEX0

4. Descripción Funcional

El joystick se comunica con el Arduino Mega enviando de forma analógica la información que representara el movimiento de los servos, el Arduino Mega a partir de esa información mapea el voltaje en un ángulo para ver el movimiento de salto que se enviarán a los servos. El módulo wifi consta de un ESP8266 el cuál se conecta en forma dual, tanto Access Point (AP) como Station, como AP crea una red a la cual se va a conectar el servidor web montado en una computadora y será accedido desde el navegador, y cómo Station el Arduino postea la información al servidor para actualizar los datos.

La Fig. 15 representa el esquema de la comunicación descrita anteriormente.

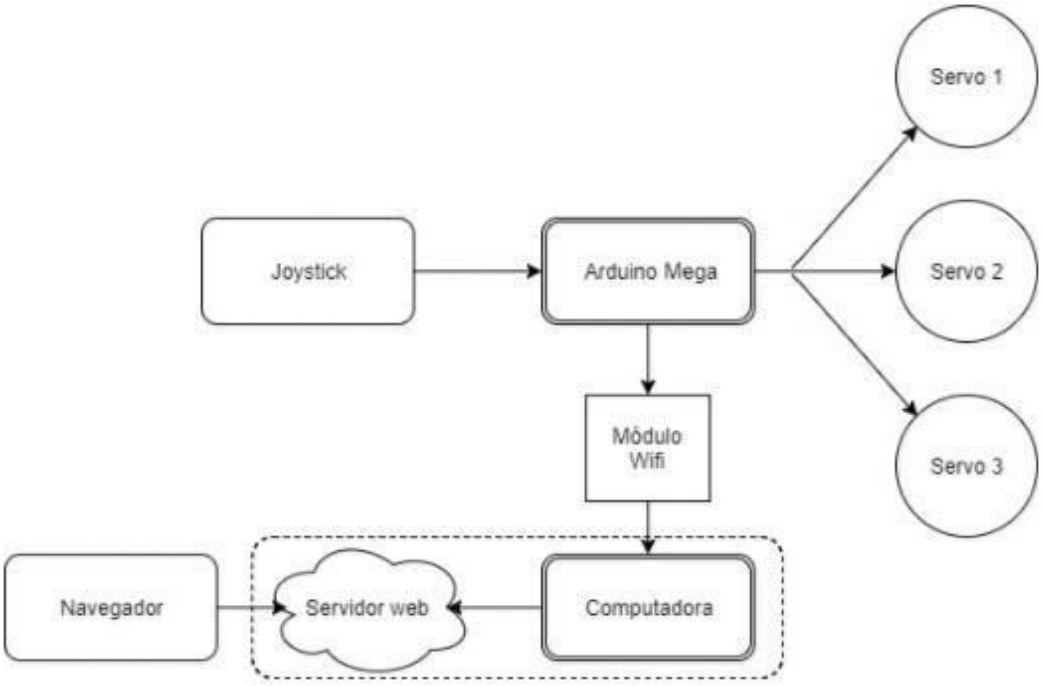

Fig. 15.

Esquema gráfico de comunicación entre módulos modificado.

Identificación de las partes del proyecto

a.- Alimentación de la placa Arduino Mega 2560

b.- E/S de la placa Arduino/NodeMCU/Wemos con el exterior excepto PC

c.- Comunicaciones de la placa Arduino/NodeMCU/Wemos con la PC

d.- Sistema/interfaz web

e.- Infraestructura de software propuesta para la PC

f.- Alimentación externa de 5V para los servomotores y el joystick

Eventos asociados al servidor web:

Cuando al servidor web llega una petición desde el navegador esta carga la pantalla de inicio de la página que contiene la información asociada a los servos por default (que todos estén centrados en 90 grados), una vez que se envía desde el Arduino un POST a través del ESP8266 hacia el servidor, este lo recibe, actualiza los datos de la posición de los servos y actualiza la página para que el usuario pueda observar el cambio.

En las siguientes capturas de pantalla puede verse la respuesta obtenida en el navegador con información sobre la posición de los servos a lo largo de la ejecución del programa.

En la Fig. 16 puede verse el estado de la página web al principio del programa, la posición de los servos es la central, y el servo seleccionado, el número uno.

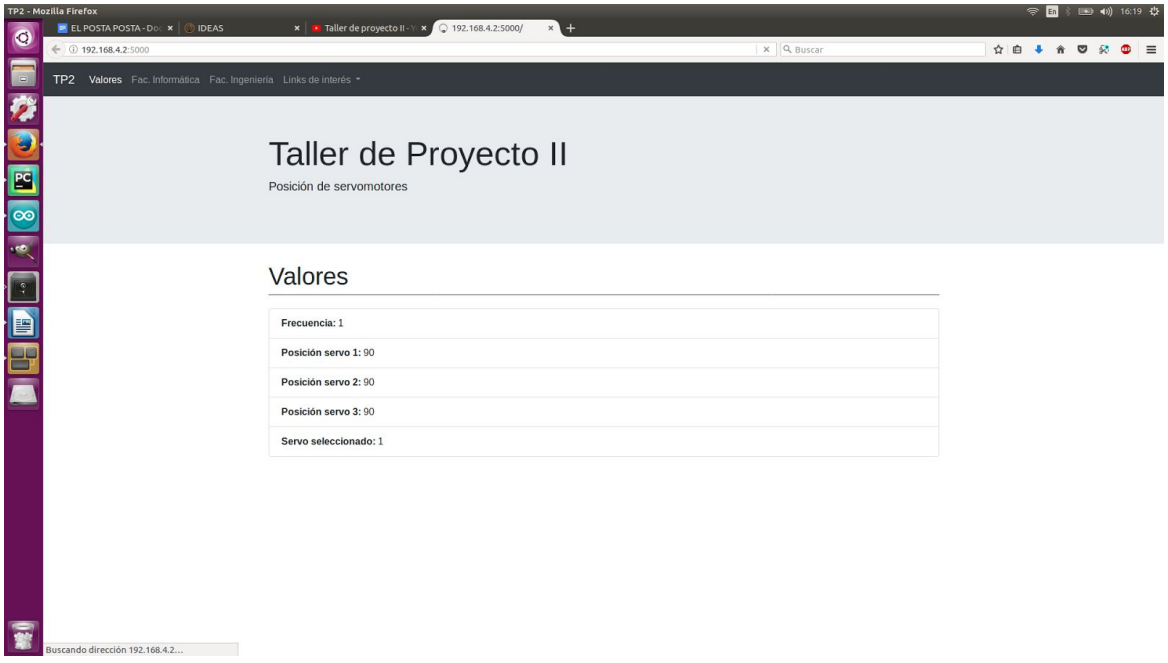

Fig. 16. Pantalla inicial

En la Fig. 17 se ve la posición de los servos en valores distintos a los iniciales al haber sido movidos a lo largo de la ejecución del programa.

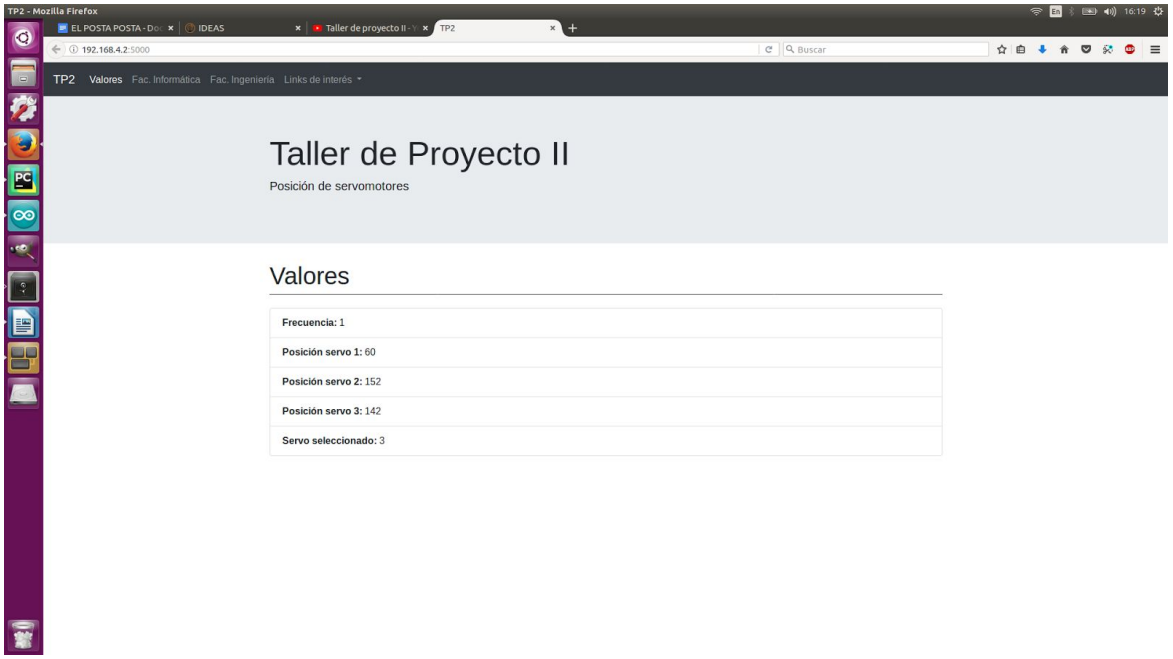

Fig. 17. Pantalla de un momento de ejecución al azar.

La Fig. 18 muestra por monitor serie, configurado en 9600 baudios, la información que será enviada y el chequeo de que funcione correctamente.

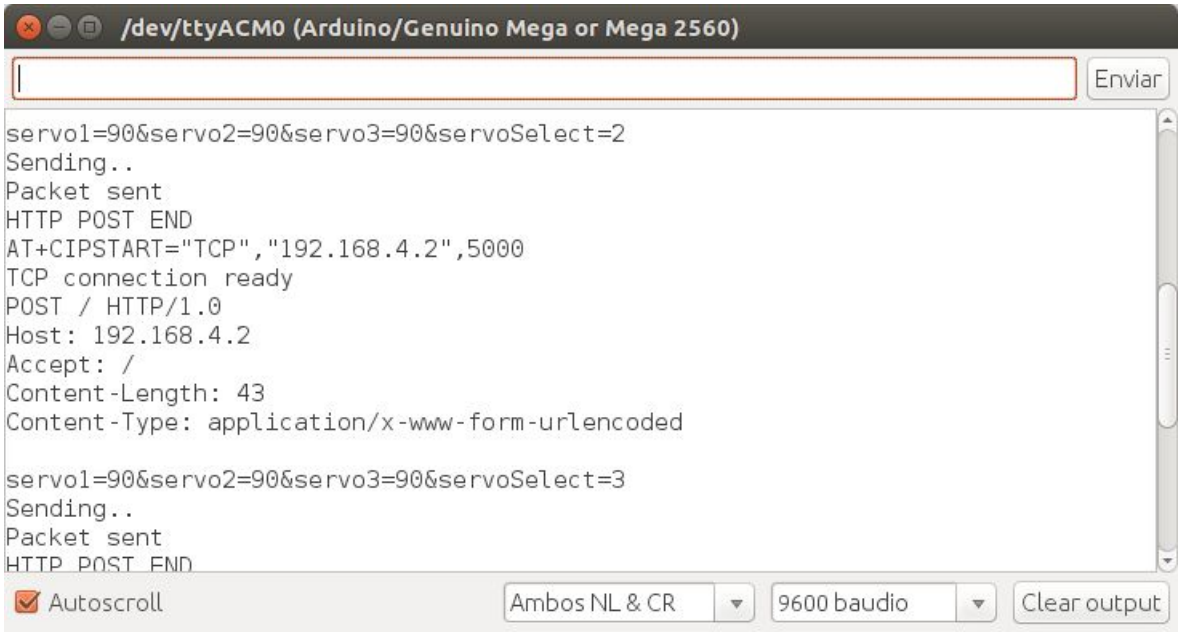

Fig. 18. Monitor serie del IDE Arduino

Siguiendo el link se accede a un video de la secuencia completa de funcionamiento: https://www.youtube.com/watch?v=QJAy68wZxKs

En el mismo se puede ver lo siguiente:

-0:00 se realiza la conexión entre la computadora y el ESP8266.

-0:19 arranca el programa, llevando todos los servos a la posición inicial (90 grados).

-0:30 se accede desde un navegador a la dirección IP 192.168.4.2:5000, en la cual se encuentra alojado el servidor, y se muestra la posición de los servos en el comienzo.

-0:41 se mueve el joystick, provocando un cambio en la posición del primer servo. Momentos después, se muestra este cambio reflejado en la web accedida por el navegador.

-0:48 se cambia el servo seleccionado mediante el presionado del botón del joystick y la posición del mismo es cambiada en cierto ángulo. Esto es testeado tres veces más.

-1:07 se enfoca al brazo para ver los cambios de ángulo ocurridos desde el comienzo y se muestran otros cambios de ángulo.

Eventos asociados a la placa de desarrollo:

El programa cuando inicia, realiza el setup() que inicializa los dispositivos y posiciona los servos en la posición de 90 grados. Una vez finaliza esto, ingresa a la función loop() en el cual:

-Cuando se detecta un movimiento del joystick, se mapea el voltaje para medir si el movimiento es ascendente o descendente.

-Cuando se detecta una pulsación del botón del joystick, se realiza el cambio de servo a controlar.

-Cuando se produce alguno de los eventos anteriores (en el caso del movimiento de los servos cuando la palanca vuelve al centro del joystick), se llama al método httppost() el cual establece una comunicación con el servidor y envía a través de un POST los datos asociados al estado del sistema.

5. Descripción del Software

El servidor web consta de 4 secciones de código:

1. Se importan desde flask las librerías Flask, render_template y request.
2. Se inicializan las variables app que será la variable del programa, y por defecto las variables de la frecuencia de actualización de la página, de los tres servos y del servo seleccionado actualmente.
3. Se crea la ruta "/" que permite métodos de POST y GET y se define una función index() que cuando recibe un método POST actualiza los valores de las variables de los servos y del servo seleccionado, y renderiza la página "response.html" con los valores nuevos de las variables.
4. Un if que permite seleccionar cual será la dirección del servidor (host), y el puerto (port) a utilizar.

El software que se ejecuta en la placa de desarrollo está formado por 6 secciones diferenciables:

1. Segmento de inclusión de librerías, en este caso la utilizada fue la librería VarSpeedServo.
2. Variables globales definidas utilizadas en el código.
3. La función setup() la cual contiene el código correspondiente a la inicialización de la comunicación serie de la terminal y la utilizada para comunicarse con el ESP8266, el reset del ESP8266, la llamada a la función createWifi(), la asignación de los pines correspondientes a cada servo, la selección de modo del botón del joystick, y escribir en los servos la posición inicial.
4. La función loop() que se ejecuta repetidamente conteniendo dos secciones: un "if" que cuando se presiona el botón del joystick hace el cambio de servo a controlar, y un "switch" que mapea la posición del joystick, determina el ángulo de movimiento y escribe en el servos determinado la nueva posición.
5. La función createWifi() que se encarga de la creación de la red Wifi, al comunicarle los comandos correspondientes al ESP8266 por comunicación serie para generar la red llamada "ESPNET" con contraseña "password".

6. Y por último la función httppost() que le asigna a la variable “data” la información de los ángulos correspondientes a cada servo y cual es el servo seleccionado actualmente y le envía esa data al servidor web en formato HTTP 1.0, a través de comandos AT comunicándose con el ESP8266.

5.1 Pseudocódigo Arduino:

```
setup( )
{
        Configurar ESP8266 con 115200 baudios
        Configurar puerto serie
        Resetear ESP8266
        Esperar a que el ESP8266 responda “OK”
        Crear Wifi
        Conectar Servos 1, 2 y 3
        Inicializar servos
        Configurar Botón
}

Loop ( )
{
        Si el flag de envío está activo
                Se hace un HTTPPOST
                Desactiva flag de envío

        Si se presionó el botón
                Se selecciona el siguiente servo
                Activa el flag de envío


        Según qué servo esté activo
                Leer entrada analógica
                Si se movió de la posición central
```

Cambiar el ángulo

Activar flag de cambio de ángulo

Si esta en la posición central y el flag de cambio de ángulo está activado

Activar flag de envío

Desactivar flag de cambio de ángulo

Enviar nuevo ángulo al servo seleccionado

}

CreateWifi( )

{

Configurar el ESP8266 en modo dual

Si el ESP8266 responde OK

Se configura un nombre y un password para la red WIFI

Sino

Retornar error

Volver a intentar CrearWifi

}

HTTPPOST( )

{

Iniciar una conexión TCP enviando un comando AT al ESP8266

Concatenar en un string la información de los servos

Concatenar en un string información para enviarle al ESP8266 en formato de comando AT

Determinar la longitud del comando AT

Notificar al ESP8266 la longitud del comando AT con la información

Si el ESP8266 responde “>”

Enviar la información al ESP8266

Cerrar la conexión TCP

}

6. Guía de Instalación Completa

a) Ambiente de desarrollo:

i)Instalación del IDE Arduino en Windows:

Descargar la última versión del IDE de Arduino desde: http://arduino.cc/en/Main/Software.

Elegir el instalador a descargar correspondiente al sistema operativo Windows.

Ejecutar el instalador previamente descargado.

Durante la instalación aceptamos el acuerdo de licencia.

Marcar todas las opciones y elegir directorio de instalación, generalmente C:\Program Files (x86)\Arduino\

Permitir instalar los drivers (si lo solicita).

ii) Instalación del IDE Arduino en Linux:

Descargar la última versión del IDE de Arduino desde: http://arduino.cc/en/Main/Software.

Elegir el instalador a descargar correspondiente al sistema operativo Linux.

Extraer los archivos del paquete.

Abrir la terminal e ingresar a la carpeta correspondiente al archivo previamente extraído.

Ejecutar el comando ./install.sh y esperar a que el proceso de instalación termine (en el caso de que falle otorgar permisos de ejecución al archivo install.sh).

iii) Instalación de Python 2.7 en Windows:

Para obtener la última versión de Python, ir a https://www.python.org/download/

Elegir el instalador a descargar correspondiente al sistema operativo Windows.

Después de la descarga, damos doble click al archivo python-2.7.7.msi y procedemos a elegir la opción siguiente hasta llegar al botón de Finalizar.

Agregar python en las variables de entorno del sistema, para esto, dar click derecho sobre el icono de Equipo->Propiedades y nos abrirá una ventana Sistema, buscamos la opción Configuración avanzada del sistema. Una vez situado ahí da click en el botón “Variables de Entorno” y nos dirigimos a “Variables del sistema” y damos click en Path y en el campo Valor de la variable agregamos la ruta donde se instaló python. C:\Python27\

iv) Instalación de Python 2.7 en Linux:

Ejecutar los siguientes comandos en la terminal para instalar los prerrequisitos:

```
sudo apt-get update
sudo apt-get install build-essential checkinstall
sudo apt-get install libreadline-gplv2-dev libncursesw5-dev libssl-dev
libsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev
```

Descargar python 2.7 ejecutando los siguientes comandos:

```
cd /usr/src
sudo wget https://www.python.org/ftp/python/2.7.14/Python-2.7.14.tgz
```

Extraer el paquete descargado:

```
sudo tar xzf Python-2.7.14.tgz
```

Compilar el código python:

```
cd Python-2.7.14
sudo ./configure
sudo make altinstall
```

v) Instalación de Pip en Windows:

Ingresar en la dirección: https://bootstrap.pypa.io/get-pip.py, descargar el archivo get-pip.py.

Abrir la consola de windows (CMD), acceder al directorio donde fue descargado el archivo .py, por último ejecutar:

```
python get-pip.py
```

Para utilizar pip en windows, agregar la dirección C:\Python2x\Scripts a el Path de Windows para poder ejecutarlo desde cualquier ruta.

vi) Instalación de Pip en Linux:

Ejecutar en la terminal el siguiente comando:

```
sudo apt-get install python-pip
```

vii) Instalación de Flask:

Ejecutar en la terminal el siguiente comando (en el caso de ser necesario otorgar privilegios de administrador):

```
pip install Flask
```

viii) Instalación de la librería VarSpeedServo al IDE Arduino:

Descargar el archivo .zip desde la sección "releases" del GitHub: https://github.com/netlabtoolkit/VarSpeedServo.

En el IDE Arduino, seleccionar Sketch->Importar librería->Añadir librería y ubicar el .zip descargado previamente.

b) Copia/Instalación de Código:

En el GitHub: https://github.com/m474dor/Tiempo-Real se encuentra todo el código correspondiente al proyecto, tanto la parte web como el sketch de Arduino.

Una vez descargado el proyecto, la carpeta Server contiene los archivos correspondientes al servidor web Python, y la carpeta Codigo Arduino contiene una subcarpeta llamada wifi que contiene el archivo wifi.ino, el cual es el que posee el sketch.

c) Guía de Compilación/Instalación/Upload de Ejecutable/s:

Indicar configuraciones y ubicaciones de los ejecutables y código del sistema en el caso del subsistema web y si hay alguna consideración o configuración especial para el código que se ejecute en la placa de desarrollo.

Para ejecutar el servidor Python ingresar a la consola de comandos, ingresa a la carpeta Server y ejecutar:

```
python app.py
```

Para cargar el sketch Arduino a la placa se deberá conectar la placa a la computadora.

Ingresar a Codigo Arduino->wifi y hacer doble click al archivo wifi.ino.

Una vez abierto el IDE Arduino, seleccionar Tools y en Board seleccionar Arduino/Genuino Mega or Mega 2560 y en Proccessor seleccionar ATmega2560 (Mega 2560).

Apéndice A

El objetivo del proyecto consiste en la construcción de un brazo robótico que permite cubrir un área 3D esférica. Para lograrlo, se deberá:

- Configurar el joystick o aplicación de celular en conexión con el microcontrolador.
- Programar el microcontrolador para recibir comandos a través de un dispositivo de entrada bluetooth, enviar a una interfaz web la información sobre la posición de los servos y control de los servos.
- Configurar los servos y el controlador de servos.
- Agregar una fuente de alimentación
- Obtención o creación de las partes plásticas que forman la estructura.
- Página web para mostrar la posición de los servos.

Como objetivos secundarios en caso de que el proyecto avance correctamente puede agregarse:

- Si se utilizó un joystick para controlar el producto, crear la aplicación de celular o viceversa.
- Patrones predefinidos de movimiento.

Apéndice B

Código correspondiente a la aplicación python:

```
# Aqui se listan los imports
from flask import Flask
from flask import render_template
from flask import request

app = Flask(__name__)

frec = 1
servo1 = 90
servo2 = 90
servo3 = 90
servoSelect = 1

@app.route("/",methods = ['POST','GET'])
def index():
    global frec
    global servo1
    global servo2
    global servo3
    global servoSelect
    if request.method == 'POST':
        servo1 = request.values.get('servo1')
        servo2 = request.values.get('servo2')
        servo3 = request.values.get('servo3')
        servoSelect = request.values.get('servoSelect')
    return render_template('response.html', frec=frec,servo1=servo1,
servo2=servo2, servo3=servo3, servoSelect=servoSelect)

if __name__ == "__main__":
    # Define HOST y PUERTO para accerder
    app.run(host='192.168.4.2', port=5000, debug=True)
```

Código correspondiente al html de la pagina response.html:

```
<html>
    <head>
        <meta http-equiv="refresh" content="{{ frec }}" />
        <title>Tiempo Real</title>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <!-- JQuery -->
        <script src="{{url_for('static',
filename='jquery/jquery-3.2.1.min.js')}}"></script>
        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="{{url_for('static',
filename='bootstrap/css/bootstrap.min.css')}}" />
        <!-- Bootstrap JS -->
        <script src="{{url_for('static',
filename='bootstrap/js/popper.min.js')}}"></script>
        <script src="{{url_for('static',
filename='bootstrap/js/bootstrap.min.js')}}"></script>

        <style type="text/css">
```

```
            .page-header{
                  border-bottom: 1px solid;
                  margin-bottom: 2%;
            }

        </style>

    </head>

    <body>

       <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
            <a class="navbar-brand" href="#">Tiempo Real</a>
            <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNavDropdown" aria-controls="navbarNavDropdown"
aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNavDropdown">
                <ul class="navbar-nav">
                    <li class="nav-item active">
                        <a class="nav-link" href="#"> Valores <span
class="sr-only">(current)</span></a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link"
href="http://info.unlp.edu.ar/">Fac. InformÃ¡tica</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link"
href="http://www.ing.unlp.edu.ar/">Fac. IngenierÃa</a>
                    </li>
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle"
href="http://example.com" id="navbarDropdownMenuLink" data-toggle="dropdown"
aria-haspopup="true" aria-expanded="false">
                            Links de interÃ©s
                        </a>
                        <div class="dropdown-menu"
aria-labelledby="navbarDropdownMenuLink">
                <a class="dropdown-item"
href="https://ideas.info.unlp.edu.ar/">IdeasUNLP</a>
                            <a class="dropdown-item" href="#">Otro..</a>
                        </div>
                    </li>
                </ul>
            </div>
        </nav>

        <div class="jumbotron jumbotron-fluid">
            <div class="container">
                <h3 class="display-4">Sistemas de Tiempo Real</h3>
                <p class="lead">PosiciÃ³n de servomotores</p>
            </div>
        </div>

        <div class="container">
```

```
            <div class="page-header">
                      <h1>Valores</h1>
                  </div>

                  <ul class="list-group">
                      <li class="list-group-item"><b>Frecuencia:</b> {{ frec
}} </li>
                      <li class="list-group-item"><b>PosiciÃ³n servo 1:</b> {{
servo1 }} </li>
                      <li class="list-group-item"><b>PosiciÃ³n servo 2:</b> {{
servo2 }} </li>
                      <li class="list-group-item"><b>PosiciÃ³n servo 3:</b>
{{ servo3 }} </li>
                      <li class="list-group-item"><b>Servo seleccionado:</b>
{{ servoSelect }} </li>
                  </ul>

        </div>

    </body>
</html>
```

Apéndice C

Código correspondiente al sketch de arduino:

```
#include <VarSpeedServo.h>       // Incluir la librería Servo
#define esp Serial1

VarSpeedServo servo1;            // Crear un objeto tipo Servo llamado servo1
VarSpeedServo servo2;            // Crear un objeto tipo Servo llamado servo2
VarSpeedServo servo3;            // Crear un objeto tipo Servo llamado servo3
int p;
int angulo1 = 90;
int angulo2 = 90;
int angulo3 = 90;
int salto = 1;
int servoSelect = 0;
int Eje_X = A1 ;
int Eje_Y = A2 ;
int boton = 3  , LED = 12 ;
int cant = 100;
int flagAngleChanged=0;
int flagSendData=0;
String data;
String server = "192.168.4.2";
String uri = "/";

void setup()
{
  esp.begin(115200);
  Serial.begin(9600);
  //reset al esp
  esp.println("AT+RST");
  delay(1000);
  if(esp.find("OK"))
    Serial.println("Module Reset");

  //creo la red wifi
  createWifi();

  servo1.attach(5);       // Conectar servo1 al pin 5
  servo2.attach(6);       // Conectar servo2 al pin 6
  servo3.attach(7);       // Conectar servo3 al pin 7
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode( boton, INPUT_PULLUP);
  servo1.write(angulo1, 0);
  delay(100);
  servo2.write(angulo2, 0);
  delay(100);
  servo3.write(angulo3, 0);
  delay(100);
}

void loop()
{
  if(flagSendData==1){
    httppost();
    flagSendData=0;
```

```
  }
  if ( ! digitalRead(boton)){
    while ( ! digitalRead(boton)) ;
    servoSelect=(servoSelect+1)%3;
    flagSendData=1;
  }

  switch (servoSelect) {

    case 0  :
      p  = map( analogRead(A1), 0, 1024, 0, 180);
      if (( p < 30 ) && (angulo1 > 5)){// Si la lectura es menor de 400
        angulo1 = angulo1 - salto ;    // disminuimos el angulo
        flagAngleChanged=1;
      }
      else if (( p > 100 ) && (angulo1 < 175)){     // Si mayor de 600
        angulo1 = angulo1 + salto ;    // Aumentamos el angulo
        flagAngleChanged=1;
      }
      else if ((p>30)&&(p<100)&&(flagAngleChanged==1)){    //cuando esta en el
medio y hubo un cambio, mandar los datos
        flagSendData=1;
        flagAngleChanged=0;
      }
      servo1.write(angulo1, 255);      // Y este es el que mueve el servo
      break;

    case 1  :
      p  = map( analogRead(A1), 0, 1024, 0, 180);
      if (( p < 30 ) && (angulo2 > 5)){// Si la lectura es menor de 400
        angulo2 = angulo2 - salto ;    // disminuimos el angulo
        flagAngleChanged=1;
      }

      else if (( p > 100 ) && (angulo2 < 175)){     // Si mayor de 600
        angulo2 = angulo2 + salto ;    // Aumentamos el angulo
        flagAngleChanged=1;
      }
      else if ((p>30)&&(p<100)&&(flagAngleChanged==1)){ //cuando esta en el
medio y hubo un cambio, mandar los datos
        flagSendData=1;
        flagAngleChanged=0;
      }
      servo2.write(angulo2, 255);      // Y este es el que mueve el servo
      break;

    case 2  :
      p  = map( analogRead(A1), 0, 1024, 0, 180);
      if (( p < 30 ) && (angulo3 > 5)){// Si la lectura es menor de 400
        angulo3 = angulo3 - salto ;    // disminuimos el angulo
        flagAngleChanged=1;
      }
      else if (( p > 100 ) && (angulo3 < 175)){     // Si mayor de 600
        angulo3 = angulo3 + salto ;    // Aumentamos el angulo
        flagAngleChanged=1;
      }
      else if ((p>30)&&(p<100)&&(flagAngleChanged==1)){    //cuando esta en el
medio y hubo un cambio, mandar los datos
```

```
        flagSendData=1;
        flagAngleChanged=0;
      }
      servo3.write(angulo3, 255);      // Y este es el que mueve el servo
      break;
  }
  delay(20);
}

void createWifi() {
  label:
  esp.println("AT+CWMODE=3");

  delay(1000);

  if (esp.find("OK") ) {
    label2:
    esp.println("AT+CWSAP=\"ESPNET\",\"password\",1,4");
    delay(1000);
    if (esp.find("OK") )
      Serial.println("OK CWJAP");
    else {
      Serial.println("FAIL CWJAP");
      goto label2;
    }
  }
  else {
    Serial.println("FAIL CWMODE 3");
    goto label;//createWifi();
  }
}

void httppost () {
  String wf="AT+CIPSTART=\"TCP\",\"192.168.4.2\",5000";
  Serial.println(wf);
  esp.println(wf);//start a TCP connection.
  if ( esp.find("OK")) {
    Serial.println("TCP connection ready");
  }
  else {
    Serial.println("TCP no");
  }

  data = "servo1=" + (String)angulo1 + "&servo2=" + (String)angulo2 +
"&servo3=" + (String)angulo3 + "&servoSelect=" + (String)(servoSelect+1);

  String postRequest =

    "POST " + uri + " HTTP/1.0\r\n" +

    "Host: " + server + "\r\n" +

    "Accept: " + "/" + "\r\n" +

    "Content-Length: " + data.length() + "\r\n" +

    "Content-Type: application/x-www-form-urlencoded\r\n" +
```

```
    "\r\n" + data;

  Serial.println(postRequest);

  String sendCmd = "AT+CIPSEND=";//determine the number of caracters to be
sent.

  esp.print(sendCmd);

  esp.println(postRequest.length() );

  if (esp.find(">")) {
    Serial.println("Sending.."); esp.print(postRequest);

    if ( esp.find("SEND OK")) {
      Serial.println("Packet sent");
      // close the connection
      esp.println("AT+CIPCLOSE");
    }
  }
  Serial.println("HTTP POST END");
}
```