TRABAJO ESPECIAL DE GRADO

DISEÑO DE UN PROTOTIPO DE SISTEMA DE ACCESO REMOTO AL CONTROLADOR DE UN BRAZO MANIPULADOR

Presentado ante la ilustre
Universidad Central de Venezuela
por el Br. Ismael Bautista
para optar al título de
Ingeniero Electricista.

Caracas, junio de 2019

TRABAJO ESPECIAL DE GRADO

DISEÑO DE UN PROTOTIPO DE SISTEMA DE ACCESO REMOTO AL CONTROLADOR DE UN BRAZO MANIPULADOR

TUTOR ACADÉMICO: Profesora Tamara Pérez

Presentado ante la ilustre
Universidad Central de Venezuela
por el Br. Ismael Jesús Bautista García
para optar al título de
Ingeniero Electricista.

Caracas, junio de 2019

A quien desees dedicar este trabajo

RECONOCIMIENTOS Y AGRADECIMIENTOS

Ismael J. Bautista G.

DISEÑO DE UN PROTOTIPO DE SISTEMA DE ACCESO REMOTO AL CONTROLADOR DE UN BRAZO MANIPULADOR

Tutor Académico: nombre del profesor. Tesis. Caracas, Universidad Central de Venezuela. Facultad de Ingeniería. Escuela de Ingeniería Eléctrica. Mención Electrónica Computación y Control. Año 2019, 144 pp.

Palabras Claves: Palabras clave.

Resumen.- Escribe acá tu resumen

ÍNDICE GENERAL

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ACRÓNIMOS

PUMA: Programmable Universal Machine for Assembly, o Programmable Universal Manipulation Arm.

SDK: Software Development Kit. Kit de Desarrollo de Software.

INTRODUCCIÓN

En este archivo debe escribir su introducción.

De acuerdo a Brea la transformada de Laplace debe estudiarse como una función definida en el campo de los números complejos

De acuerdo a la ecuación

CAPÍTULO I

EL PROBLEMA

1.1. Planteamiento del problema

El brazo manipulador que se encuentra en el Laboratorio de Control (LCID) actualmente cuenta con un controlador y una interfaz instalados en un computador que sirve de administrador de las acciones de control del sistema. No obstante, el brazo manipulador es de uso reducido porque necesita ajustes en cuanto al controlador y porque las plataformas computacionales se están haciendo obsoletas; por lo que se quiere añadir a esta implementación otro sistema para realizar acciones de control de forma remota.

1.2. Justificación

Este equipo permitirá reforzar conocimientos en el área de sistemas automáticos de control. Siendo este manipulador un diseño enfocado como equipo de laboratorio en la Escuela de Ingeniería Eléctrica (EIE)(Marcano, 2013), es siempre importante una propuesta para mejorar la utilidad del mismo utilizando nuevas tecnologías relacionadas con accesos remotos.

1.3. Objetivos

1.3.1. Objetivo general

Diseñar un prototipo de sistema de acceso remoto al controlador del brazo manipulador MA2000.

1.3.2. Objetivos específicos

- Documentar las técnicas y metodologías de acceso remoto empleadas comúnmente en brazos manipuladores.
- Determinar las acciones del sistema de control del brazo manipulador.
- Seleccionar los dispositivos necesarios para el diseño.
- Seleccionar el modo de acceso remoto.
- Diseñar el sistema de acceso remoto.
- Validar la propuesta.
- Realizar un manual del prototipo.

1.4. Alcance y limitaciones

El presente trabajo estará solo enmarcado en el diseño del prototipo. Por lo tanto, no contempla una aplicación práctica a un sistema industrial real. Cualquier implementación física o ajuste mecánico para la validación no está contemplada en este trabajo.

1.5. Análisis de factibilidad

Para la realización de este trabajo se cuenta con una documentación ampliada del sistema de control del brazo manipulador en cuestión. Además, existe suficiente información con respecto al internet de las cosas y manipulación remota de sistemas.

Por otra parte, al tratarse del diseño de un prototipo, no requiere necesariamente una implementación, por lo cual este trabajo se considera factible ya que disminuye los riesgos económicos y el autor considera que cuenta con la información necesaria.

De ser necesario, cualquier costo adicional será asumido por el autor. Por último, el tiempo que se propone se considera suficiente para la realización de este trabajo.

CAPÍTULO II

MARCO TEÓRICO

2.1. Antecedentes

Marcano, Juan. "Implementación de sistema de programación de Trayectorias para el brazo manipulador MA2000". Universidad Central de Venezuela, 2013. Tutor: Prof. Alejandro González. Para establecer el control de posición del MA2000, se implementó un controlador de posición PID independiente para tres articulaciones utilizando el motor DSP de un dsPIC. Se desarrolló un software en el PC que se comunica con el microcontrolador y permite el ajuste dinámico del controlador PID y la visualización de la respuesta dinámica de cada articulación así como la programación de trayectorias. Se ha recuperado el funcionamiento de la pinza neumática del equipo y se ha elaborado documentación en línea del proyecto. (Marcano, 2013)

Labrador, Alexis. "Diseño de un equipo para el control y monitoreo de un motor asincrónico usando una aplicación móvil". Universidad Central de Venezuela, 2018. Tutor: Prof. Servando Álvarez. El presente trabajo reporta el diseño de un equipo para controlar el encendido y apagado, junto al monitoreo de tensión, corriente y tiempo de funcionamiento de un motor asincrónico usando un ordenador de placa reducida y un dispositivo móvil por medio de la red de comunicación Internet de las cosas. Para lograrlo se selecciona el ordenador de placa reducida Raspberry Pi 3 modelo B, el sensor ACS712, módulos de relés para el acciona-

miento y componentes electrónicos, se elige y se programa la plataforma de IoT, Ubidots for Education, para el monitoreo y control de forma remota, se selecciona la arquitectura de red en estrella, se eligen los tipos y modo de avisos que tendrá el sistema según las que permite Ubidots for Education, para finalmente implementar dicho diseño con el fin de validar el correcto funcionamiento.(Labrador, 2018)

Valderrama, Jorge. "Diseño de un conjunto de prácticas para la configuración y uso básico del microcontrolador ESP8266". Universidad Central de Venezuela, 2018. Tutor: Prof. Ebert Brea. Este trabajo tiene como resultado una guía para la programación básica del microcontrolador ESP8266 a través del NON OS SDK de Espressif. En él se estudian los elementos básicos como: UART, GPIO, Interrupciones, Timer y Wifi. Cada uno de estos elementos estudiados viene acompañado por una guía de programación y ejemplos para implementación. (Valderrama, 2018)

2.2. Fundamentos sobre brazos manipuladores.

2.2.1. Brazo manipulador.

Un brazo manipulador, o comúnmente llamado "brazo robot", es un sistema electrónico programado para realizar acciones de control a una parte mecánica. Estos existen de diversos tipos y morfologías y tienen múltiples aplicaciones industriales.
El término “brazo” se le atribuye por su similitud a la extremidad humana y a su funcionalidad. Contando en algunos casos con partes como “codo”, "muñeca", "hombro", "mano", entre otros. Existen de tipo: esférico, articulado, SCARA, cilíndrico y cartesiano. Como se observa en la figura 2.1.

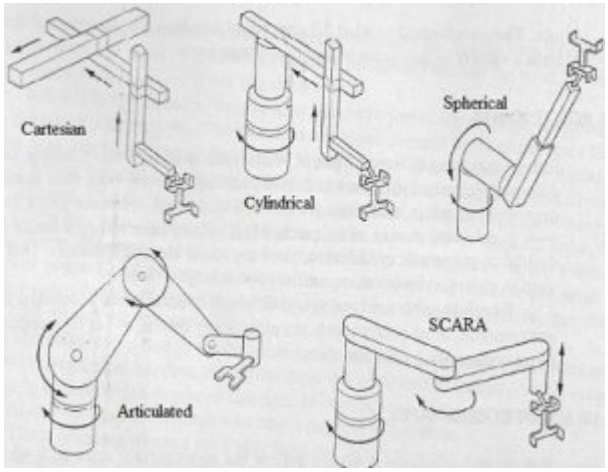


Figura 2.1. https://engineering.eckovation.com/pick-place-robotic-arm/

Entre algunos brazos manipuladores conocidos se tienen: PUMA, Stanford y Canadarm.

2.2.2. Brazo pick and place.

El término "pick and place" está asociado a la utilidad del brazo robótico. Ya que posee un elemento final que permite sostener y soltar un elemento deseado. Según la figura 2.1 cualquier tipo de brazo robot puede ser utilizado como pick and place, ya que no depende de sus grados de libertad ni de uniones, sino de poseer un elemento como una pinza o garra para cumplir una función específica.

Los siguientes brazos manipuladores a describir, se consideran del tipo "pick and place" debido a sus funciones.

2.2.3. PUMA.

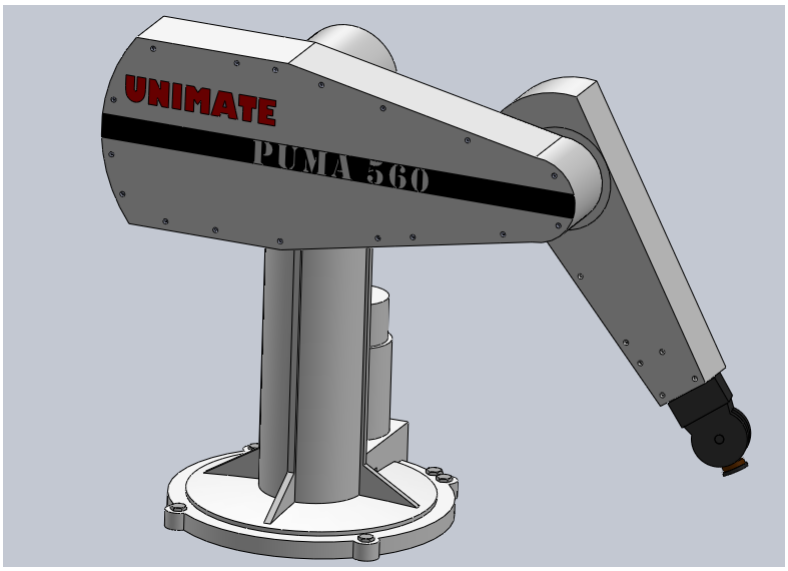
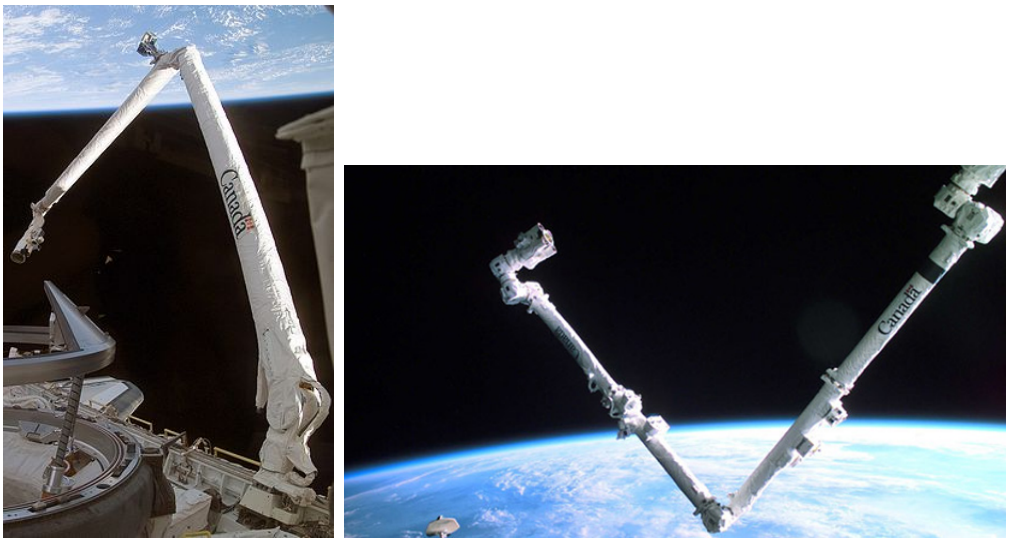


Figura 2.2. https://grabcad.com/library/robot-puma-560

El brazo PUMA: Programmable Universal Machine for Assembly, o Programmable Universal Manipulation Arm, por sus siglas en inglés; es un robot industrial desarollado por la empresa Unimation alrededor del año 1980. Entre los modelos más conocidos destacan los de la serie 500, que corresponden a brazos de seis grados de libertad con una herramienta final de control.

Ha sido muy utilizado en el campo de la medicina, ya que una característica importante de este sistema es que cuenta con un marco que le permite una mayor precisión y rapidez en los movimientos. Según se menciona en (Miller, Eriksson, Fleisher, Wiener-Kronish, y Young, 2009), este cuenta con la aceptación de su uso para fines médicos.

2.2.4. Canadarm



(a) Canadarm 1 (b) Canadarm 2

Figura 2.3. Modelos de brazo Canadarm.

Es un robot diseñado por un equipo de industrias canadienses (Spar Aerospace Ltd., CAE Electronics Ltd. y DSMA Atcon Ltd) bajo la dirección del Concilio Nacional de Investigación de Canadá (NRCC, por sus siglas en inglés). En la actualidad, el brazo Canadarm 2 se encuentra en la Estación Espacial Internacional (ISS).

Se trata de un robot de seis grados de libertad y un elemento final de control controlado por control remoto desde la ISS. El primer modelo, el Canadarm, operó durante treinta años en la Estación Espacial Internacional y fue conocido también con el nombre Shuttle Remote Manipulator System (SRMS).

2.2.5. Brazo manipulador MA2000.

Figura 2.4. Robot MA2000

Es un brazo robot diseñado alrededor de los años 1980s por George Carter, fundador de Labman Automation Ltd y fabricado por TecQuipment International Ltd, como petición de la Open University británica.

El diseño original era accionado por servomotores que permitían realizar control a lazo cerrado en cualquier instante. Requería adicionalmente una unidad compresora de aire para accionar la pinza neumática.

En un principio, este brazo era controlado por computadores IBM o por microcontroladores BBC.

En el trabajo de Marcano, se realizó un controlador manejado por computador al cuál se le realizarán mejoras para alcanzar un acceso remoto a este sistema.

2.2.6. Uso de librerías en lenguaje C

String.h

stdlib.h

2.2.7. Kit de desarrollo de software: Espressif NON OS ESP-8266 SDK

CAPÍTULO III

TÉCNICAS Y METODOLOGÍAS DE ACCESO REMOTO EN BRAZOS MANIPULADORES

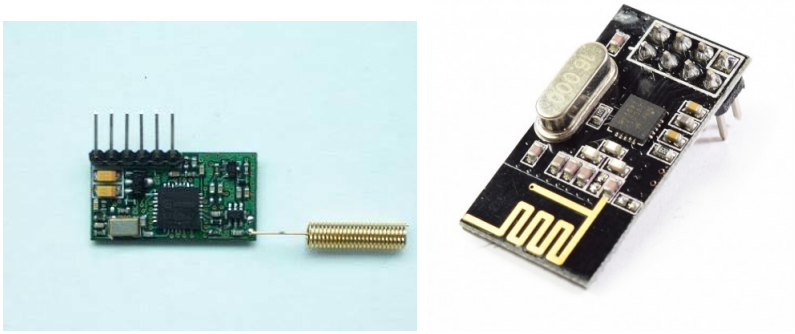
3.1. Acceso remoto.

En este trabajo se entenderá como Acceso Remoto a todo sistema o recurso ubicado físicamente en una estancia que realice una acción sobre otro en distinto lugar de forma inalámbrica vía una red local o externa a través de cualquier medio de comunicación como WiFi, Bluetooth, entre otros.

3.1.1. Acceso remoto por RF.

Cuando se habla de RF, se refiere al espectro de radiofrecuencia contemplado entre los 3Hz hasta 300GHz dentro del espectro electromagnético.

Al realizar este tipo de diseño, es necesaria la implementación de circuitos osciladores que logren emitir o recibir la onda de radio que contiene la información de interés. No obstante, con la ayuda de transceptores de radiofrecuencia integrados como el nRF24L01 (Figura 3.1(b)) o KYL-500S (Figura 3.1(a)), se puede realizar el mismo proceso optimizado para el trabajo con microcontroladores.



(a) KYL-500S. Fuente: Shenzen KYL Communication Equipment Co., Ltd.

(b) nRF24L01. Fuente: Nordic Semiconductor.

Figura 3.1. Algunos transceptores de radiofrecuencia.

El diseño de un Acceso remoto por RF debe llevar dos etapas: Transmisión y recepción. La etapa de recepción debe estar conectada al brazo robot, con el interés de realizar las mínimas modificaciones posibles.

En cuanto a la etapa transmisora, debe

3.1.2. Acceso remoto por Bluetooth.

3.1.3. Acceso remoto por Wi-Fi.

3.1.4. LEGO MindStorm NXT.

3.1.5. QNX Photon

3.1.6. Aplicación de IoT.

CAPÍTULO IV

ACCIONES DEL SISTEMA DE CONTROL DEL BRAZO MANIPULADOR MA2000.

Este brazo manipulador cuenta con seis grados de libertad, siendo tres correspondientes al hombro, codo y muñeca, controlados con motores DC. Los grados restantes, correspondientes a la mano, poseen motores de aeromodelismo. Cuenta también con una pinza neumática como herramienta final de control. Esto se observa en la figura 4.1.

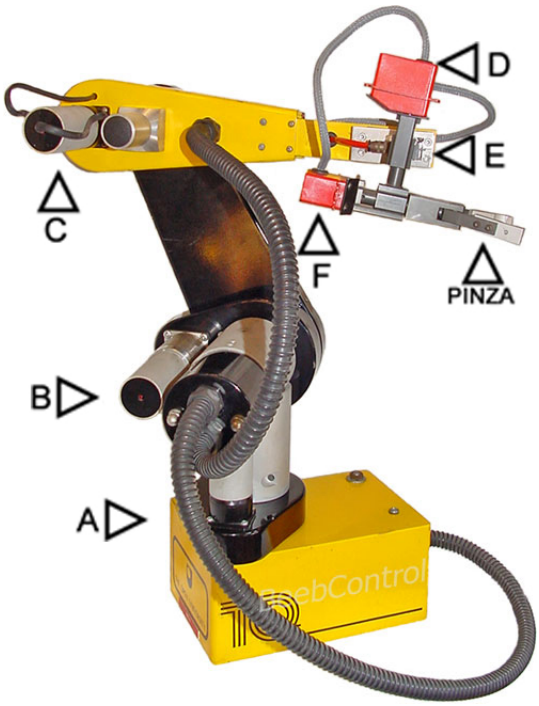


Figura 4.1. Esquema de indentificación de los motores del brazo y la pinza neumática.

Con respecto a cada grado de libertad, se tienen fijados los límites de movilidad de 270° para los motores DC y 120° para los motores de aeromodelismo. Esto para evitar daños a la integridad del equipo.

El controlador principal del sistema está desarrollado con un microcontrolador dsPIC30F3011, quien actúa como servidor recibiendo vía puerto serial las órdenes de control, para posteriormente realizar las correspondientes acciones indicadas en cada trama. Para ello, cuenta con un módulo convertidor de USB-Serial (FT-DI232) para una óptima comunicación con el computador y el software cliente. El controlador también puede responder el estado de cada articulación relacionada a los motores DC, leyendo las entradas analógicas conectadas a potenciómetros que ayudan a conocer la posición del brazo y así ejecutar el control PID.

El sistema realiza una acción de control PID en tres articulaciones principales (hombro, codo y muñeca). Para esto se configuran los parámetros desde el software cliente y se envían al controlador en una trama de tres bytes vía puerto serial.

4.1. Software Robocom

Anteriormente mencionado como 'software cliente', este programa es una interfaz gráfica de usuario desarrollada en lenguaje C++ que permite generar la trama de comunicación que se transmite del computador a la unidad de control mediante el puerto serial.

Esta herramienta permite la posibilidad de editar las variables relacionadas al movimiento del brazo y al controlador PID (factor proporcional, factor integral y

factor derivativo) para cada motor DC, correspondientes al hombro, codo y muñeca. Configura la activación o desactivación de los puentes H, el accionamiento de la pinza neumática. También puede configurar el timer PR1 del microcontrolador con el que se estima el tiempo de muestreo para la ejecución de las acciones de control.

4.1.1. Elementos fundamentales

A continuación, se explicarán los detalles de mayor uso contenidos en el software Robocom.

Al abrir el programa, se puede observar lo indicado en la figura 4.2:

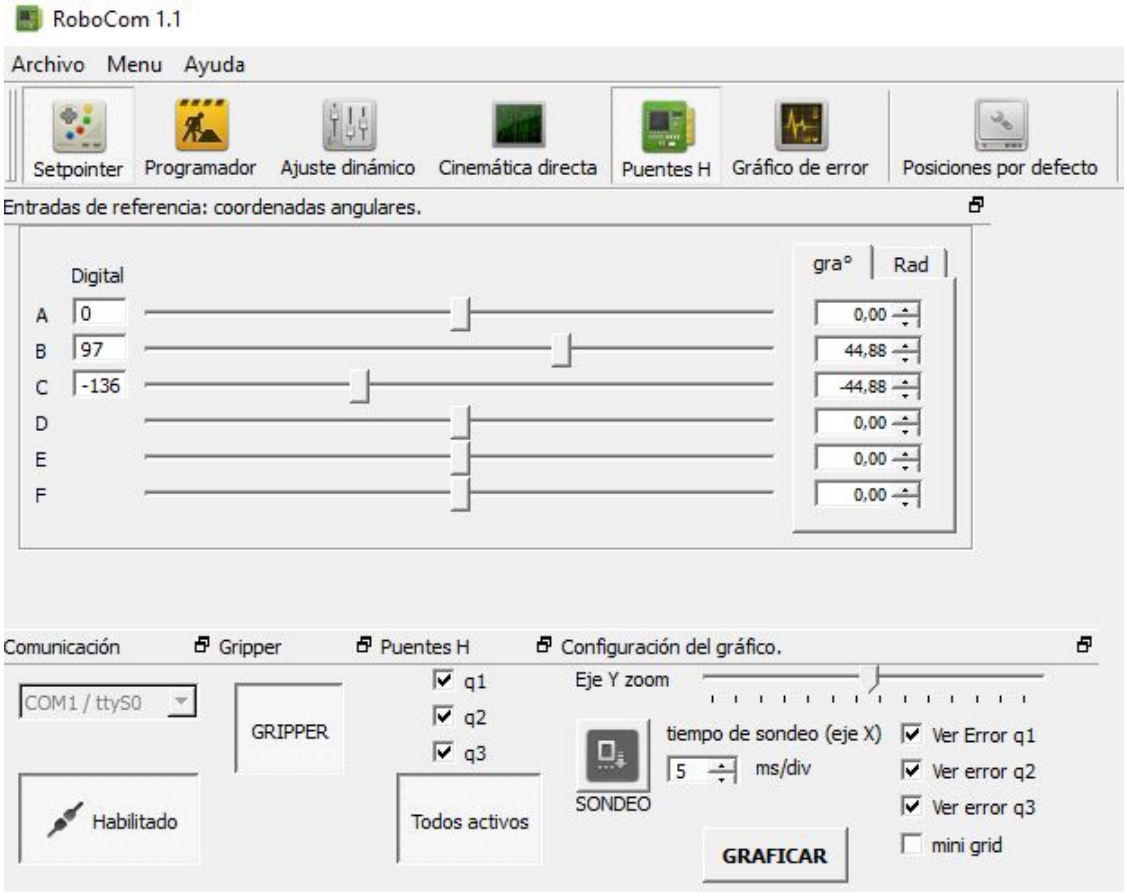


Figura 4.2. Vista inicial al abrir el software Robocom.

En el panel inferior, según se observa en la figura 4.3, se encuentran las configuraciones generales de comunicación, el accionamiento de la pinza neumática, la habilitación de los puentes H y un menú que corresponde a la configuración del graficador del error.

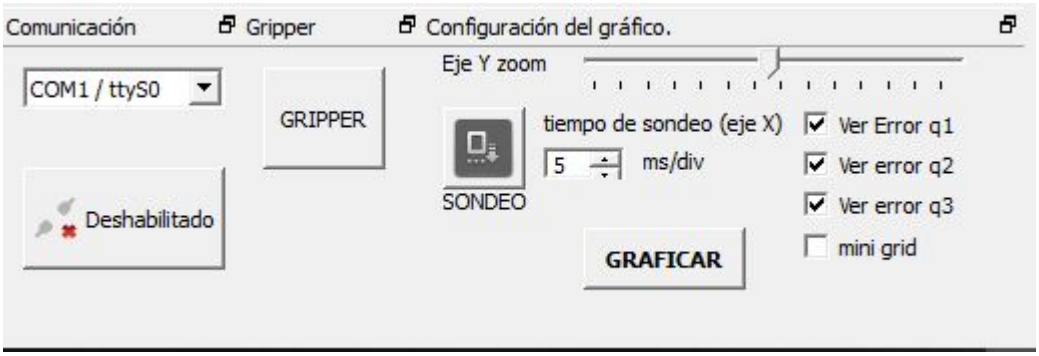


Figura 4.3. Panel inferior de Robocom

En la pestaña 'Comunicación', se selecciona el puerto COM al que se conecta originalmente el controlador del brazo.

La pestaña 'Gripper' posee un botón que activa o desactiva la pinza neumática.

La pestaña 'Puentes H' permite habilitar o deshabilitar los puentes H de los motores principales.

En la pestaña 'Configuración del gráfico' se encuentran las opciones de visualización del error. No obstante, esta funcionalidad no es estudiada en este trabajo.

Un elemento importante es el 'Setpointer', indicado en la figura 4.4. Ya que esto permite ajustar la entrada de referencia al controlador y también realizar movimientos en tiempo real con el brazo.

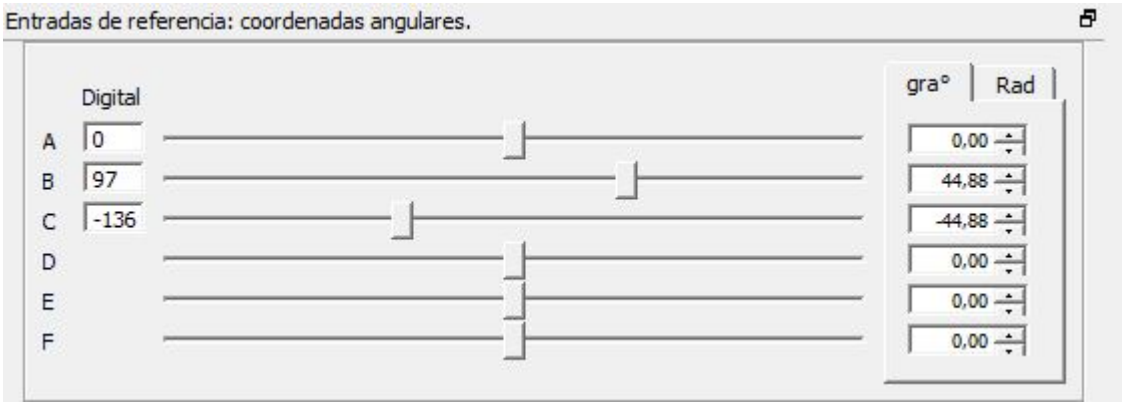


Figura 4.4. Función 'Setpointer' del software Robocom.

A la izquierda se observan las letras 'A,B,C,D,E,F' que corresponden a cada motor a mover. 'A,B,C' son para los motores del hombro, codo y muñeca. El resto es para los motores de aeromodelismo de la mano. De igual modo, la parte central cuenta con barras deslizantes que permiten generar la trama en tiempo real.

Los números que se observan en el panel 'Digital' son los utilizados para generar el segundo y tercer byte (dato) de la trama de comunicación.

El programa realiza una conversión del número observado en digital a grados o radianes según la expresión 4.1:

$$Valor_{grados} = (Valor_{digital} + K_{offset})K_{pro} \tag{4.1}$$

Donde K_{pro} es una constante de calibración para convertir el ángulo de binario a grados, medida en [grados/bits] y su valor de inicio es 0.33 grados/bits para los tres casos (A, B, C). Para los casos (D, E, F), el valor es fijo y vale 0.035[grados/bits].

K_{offset} es el error de corrección de la conversión a bits y es medida en [bits].

Su valor inicial es 0 para el motor A y C, y 39 para el motor B. Para los motores (D, E, F), el valor es fijo y vale 5682[bits].

En la pestaña 'Calibración', mostrada en la figura 4.5, puede apreciarse mejor esta información.

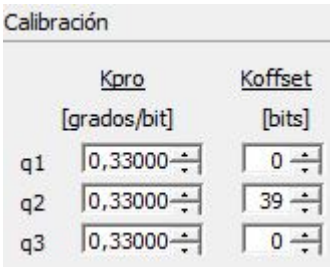


Figura 4.5. Configuración de constantes de calibración del software Robocom

4.1.2. Generación de la trama

La estructura general de la trama enviada desde el software Robocom al sistema de control consta de:

Tabla 4.1. Estructura general de la trama de comunicación.

comando	dato high	dato low

La cabecera o 'comando' de la trama puede tomar cualquier valor referido en la tabla 4.3. El valor indicará qué instrucción ejecutará el controlador y qué valor tomará el dato del comando.

Un ejemplo de esto, puede verse con ayuda del software SerialMon v1.5.0.0, que es un sniffer/monitor de puerto serial RS232 freeware para Windows.

Desde Robocom se puede enviar una rutina default para configuración previa

de la posición de cada motor. Esto se logra presionando la pestaña 'Posiciones por defecto' que se observa en la figura 4.2.

Esto produce la siguiente trama:

Tabla 4.2. Comandos enviados al generar trama default. Información obtenida con el software SerialMon.

TRAMA	INSTRUCCIÓN
F1 00 C0	Motor 1 en 0,99
F2 0B C0	Motor 2 en 28,38
F3 F4 40	Motor 3 en -15,51
F4 0E CC	Motor 4 en -66,29
F5 0E CC	Motor 5 en -66,29
F6 0E CC	Motor 6 en -66,29
F1 00 00	Motor 1 en 0
F2 18 40	Motor 2 en 44,88
F3 DE 00	Motor 3 en -44,88
F4 16 40	Motor 4 en 0,49
F5 16 32	Motor 5 en 0
F6 16 32	Motor 6 en 0
F0 01 F4	PR1 en 500

Para comprender los datos enviados que se muestran en la tabla 4.2, se tiene una lista de comandos permitidos en la comunicación del controlador y Robocom. Esta lista se muestra en la tabla 4.3.

Cada dato incluido en la trama es procesado para ser enviado de forma correcta a la unidad de control. El procesado se realiza partiendo de un valor digital que luego es calculado en su equivalente en grados o radianes.

La construcción del dato para los motores A,B y C es la siguiente:

$V_{digital} = V_{digital} << 6$

$V_{high} = V_{digital} >> 8$

$V_{low} = V_{digital}$

El calculo del equivalente en grados es:

$$V_{grados} = K_{pro}(V_{digital} + K_{offset}) \tag{4.2}$$

Considerando como valores por defecto K_{pro} = 0,33 [grados/bits] para cada motor y $K_{offset} = 0$ para los motores A y C; y $K_{offset} = 39$[bits] para el motor B. Estos valores pueden ser modificados dentro de la interfaz del programa según como lo requiera el usuario.

La construcción del dato para los motores D, E y F es la siguiente:

$V_{high} = V_{digital} >> 8$
$V_{low} = V_{digital}$

El calculo del equivalente en grados es:

$$V_{grados} = 0,035(V_{digital} - 5682) \tag{4.3}$$

La información contenida en la cabecera y cuerpo de la trama transmitida, se construye de acuerdo a las tablas 4.3 y 4.4.

Tabla 4.3. Comandos emitidos desde el software Robocom (cliente) al brazo.

CLIENTE		
1er byte Comando (8 bits)	2do y 3er Byte Dato (16 bits)	Descripción
E0	00	Enable articulacion A
	01	Disable articulacion A
	02	Enable articulacion B
	03	Disable articulacion B
	04	Enable pinza neumática
	05	Disable pinza neumática
	06	Enable articulacion C
	07	Disable articulacion C
	09	Solicitud de la posición del A/D de articulación A
	10	Solicitud de la posición del A/D de articulación B
	11	Solicitud de la posición del A/D de articulación C
F0	Dato	Escribir Periodo de muestreo
F1	Dato	Escribir Referencia de control de PID A (PID_A.controlReference)
F2	Dato	Escribir Referencia de control de PID B (PID_B.controlReference)
F3	Dato	Escribir Referencia de control de PID C (PID_C.controlReference)
F4	Dato	Escribir Referencia de motor de modelismo D (OC1RS)
F5	Dato	Escribir Referencia de motor de modelismo E (OC2RS)
F6	Dato	Escribir Referencia de motor de modelismo F (OC3RS)
F7	Dato	Escribir término proporcional del PID A
F8	Dato	Escribir término proporcional del PID B
F9	Dato	Escribir término proporcional del PID C
FA	Dato	Escribir término integral del PID A
FB	Dato	Escribir término integral del PID B
FC	Dato	Escribir término integral del PID C
FD	Dato	Escribir término derivativo del PID A
FE	Dato	Escribir término derivativo del PID B
FF	Dato	Escribir término derivativo del PID C

El controlador también puede dar respuesta a las solicitudes de posición del A/D. Los comandos emitidos por el controlador se observan en la tabla 4.4. No obstante, no son objeto de estudio de este trabajo.

Tabla 4.4. Comandos emitidos por el controlador (servidor) como respuesta al cliente.

SERVIDOR		
1er byte Comando (8bits)	2do y 3er byte Dato (16bits)	Descripción
A9	Dato	Valor del A/D de articulación A
B9	Dato	Valor del A/D de articulación A*
C9	Dato	Valor del A/D de articulación B
D9	Dato	Valor del A/D de articulación B*
E9	Dato	Valor del A/D de articulación C
F9	Dato	Valor del A/D de articulación C*

4.2. Unidad de control

El microcontrolador recibe todas las instrucciones vía puerto serial configurado con 8 bits, 1 bit de parada, 115200 baudios; y utiliza solo los pines de Rx (U2RX) y Tx(U2TX) para establecer la comunicación. Ante cada instrucción de movimiento, el microcontrolador puede emitir un total de nueve señales PWM. Para los motores A, B, C, les corresponden dos señales, una para movimiento positivo y otra para movimiento negativo, respectivamente. El resto de los motores poseen solo una señal PWM para cada uno. El diagrama de conexiones e identificación de las funciones de cada pin se puede observar en la figura 4.6.

4.3. PWM

Motores A, B, C:

11bits. 29,328kHz

"La resolución de la señal de PWM obtenida es de 11 bits, por lo tanto para obtener un duty cycle máximo es necesario que el registro PDCx tenga un valor de

2048. Los pines de salida de PWM se han configurado en modo complementario, por lo tanto para que la corriente promedio entregada al motor sea cero es necesario colocar PDCx=1023, para que el motor produzca par máximo el registro de duty cycle debe ser PDCx=0 para que gire en un sentido ó PDCx =2048 para que gire en el otro sentido."

Motores D, E, F:

El manual anexo al trabajo de Marcano indica que:

"se ha configurado el mismo para un periodo de 17,3 ms, para obtener una posición del servomotor centrada, que corresponde con un pulso 1,5 ms de duración el registro de duty cycle debe ser OCxRS=0x1632 (5682 en decimal). Es necesario generar un pulso desde los 1 ms hasta los 2ms por lo tanto los valores válidos del registro en esta aplicación van desde el 3788 hasta el 7576 (en sistema decimal), lo que permite obtener la posición deseada en dentro del rango de posiciones del servomotor."

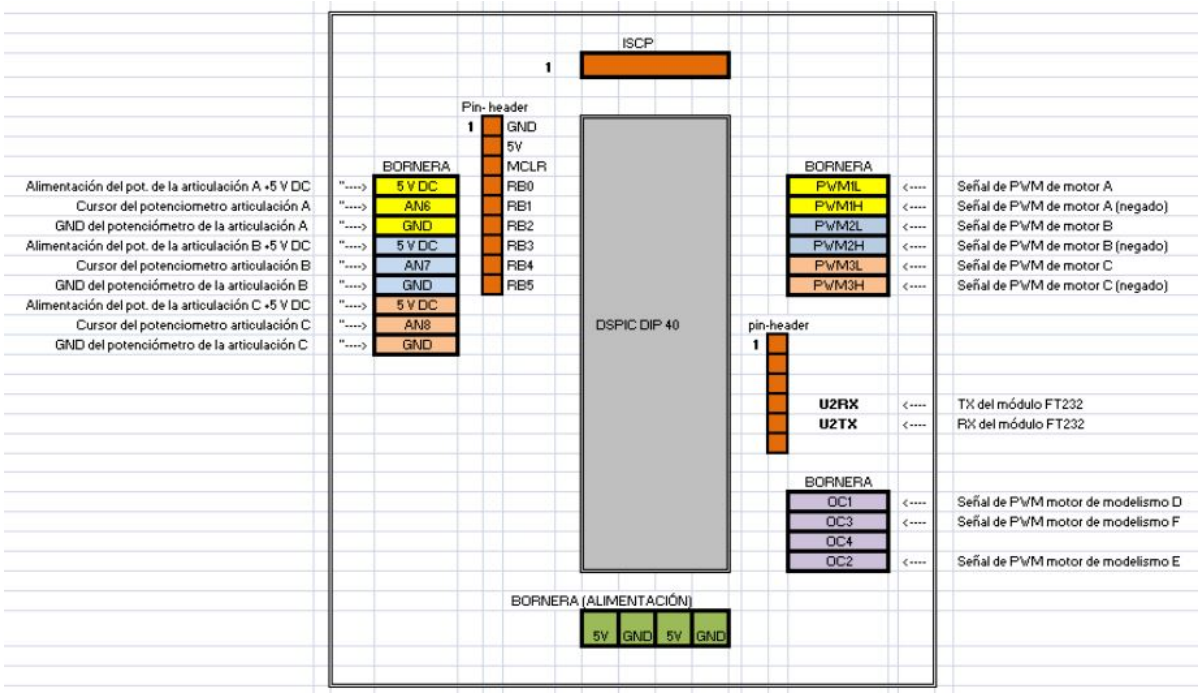


Figura 4.6. Pines de conexión del módulo controlador original.

CAPÍTULO V

SELECCIÓN DE DISPOSITIVOS

CAPÍTULO VI

SISTEMA DE ACCESO REMOTO

El sistema de acceso remoto está desarrollado sobre un microcontrolador ESP-8266 12-E, utilizando la tarjeta de desarrollo Wemos para el prototipo de este trabajo.

Para el desarrollo del sistema se utilizó como herramienta de programación el Kit de Desarrollo de Software (SDK) ESP-8266 NON-OS SDK de Espressif en lenguaje C, usando el PlatformIO IDE de Atom versión 1.38.1, compilado con GCC (GNU) 4.8.2.

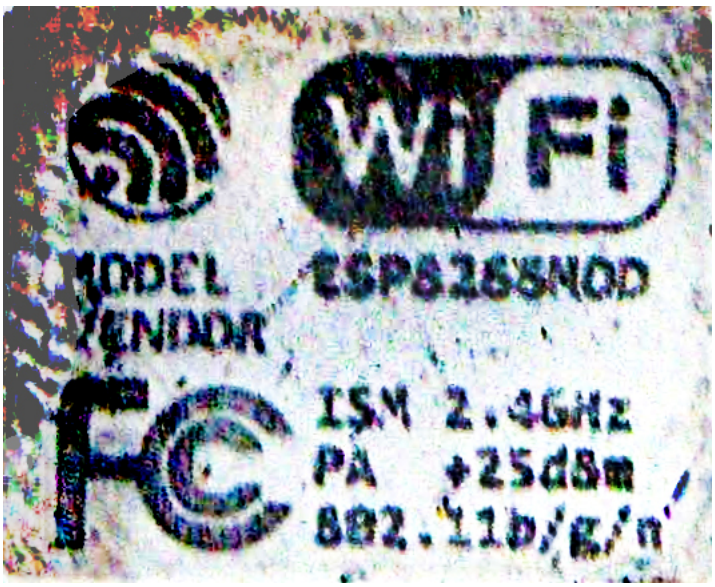


Figura 6.1. Placa del microcontrolador ESP-8266.

Tabla 6.1. Especificaciones de hardware

HARDWARE	
Tarjeta de desarrollo	LOL1n
Microcontrolador	ESP-8266 12-E
RAM	80KB
Flash	4MB
Oscilador	80MHz
Procesador	Xtensa Tensilica L106
WiFi	892.11b/g/n
ISM	2,4GHz
PA	+25dBm

Tabla 6.2. Especificaciones de software

SOFTWARE	
SDK	Espressif ESP-8266 NON-OS SDK v2.1.0
Lenguaje	C
Compilador	GCC (GNU) 4.8.2
IDE	Atom PlatformIO IDE v1.38.1

La información de las especificaciones de hardware y software se observan en la tabla 6.1 y 6.2.

6.1. Contenido del programa

El programa consta de cinco archivos: user_main.c, user_config.h, funciones.h, funciones.c, pagina.h.

6.1.1. user_main.c

En el archivo user_main.c se encuentra el segmento de código principal y las rutinas de configuración del módulo para su operatividad.

6.1.2. user_config.h

En el archivo user_config.h se define el mapeo de pines de la tarjeta de desarrollo de acuerdo a la identificación de las GPIO según el SDK. También se definen algunas variables utilizadas para generar las tramas de comunicación y las variables para SSID y contraseña de la red WiFi.

6.1.3. funciones.h y funciones.c

En el archivo funciones.h se encuentran los prototipos de todas las funciones y el llamado de las librerías de mayor importancia desde el SDK, indicadas a continuación.

```
#include "stdlib.h"
#include "stdio.h"
#include "osapi.h"
#include "user_interface.h"
#include "driver/uart.h"
#include "ets_sys.h"
#include "c_types.h"
#include "espconn.h"
#include "mem.h"
#include "gpio.h"
#include "eagle_soc.h"
```

En el archivo funciones.c están todas las definiciones de las funciones utilizadas en el programa. Además, están declaradas las variables y estructuras globales necesarias.

6.1.4. pagina.h

Este archivo contiene dos variables tipo char que contienen en una cadena el código HTML de la página web y la cadena de respuesta ante una solicitud exitosa.

6.2. Funcionamiento

El programa principal consta del establecimiento de un servidor TCP que espera una solicitud HTTP proveniente del browser al que se tiene acceso como cliente.

6.2.1. Pseudocódigo:

```
Programa principal;
begin
    Configurar UART;
    Configurar GPIO;
    Encender LED D4;
    Configurar WiFi;
    Crear servidor TCP;
    while 1==1 do
        //Esperar solicitud TCP;
    end
end
```

```
Interrupción Wifi;
begin
    if Conexión a página then
        Abrir canal TCP;
        Responder 200 OK;
        Encender LED D2 al iniciar carga;
        Apagar LED D2 al culminar carga;
        // Este canal no requiere ser desconectado.
    end
    while Canal abierto do
        switch Solicitud do
            case Mover motor do
                Recibir cadena HTTP;
                Procesar información;
                Calcular el valor del movimiento;
                Construir la trama;
                Enviar al puerto serial;
                Desconectar canal TCP;
            end
            case Configurar parámetro do
                Recibir cadena HTTP;
                Procesar información;
                Calcular el valor del parámetro;
                Construir la trama;
                Enviar al puerto serial;
                Desconectar canal TCP;
            end
            case Configurar constantes de calibración do
                Recibir cadena HTTP;
                Procesar información;
                Calcular el valor de la constante;
                Cambiar el valor de la constante;
                Desconectar canal TCP;
            end
        end
    end
end
```

Función de desconexión;

begin

    Borrar canal TCP;

    Crear nuevo canal TCP. //Esperar;

end

En la tabla 6.2.1, se especifica la información de la señal WiFi sobre la que se crea el servidor. Este es creado en el puerto 8266 sobre la IP 192.168.4.1 vía la señal WiFi emitida desde el microcontrolador. Esta comunicación se realiza a 2.4GHz y 54Mbps y solo puede crear 4 canales de comunicación TCP en paralelo, por ese motivo debe desconectarse después de cada comunicación para evitar la saturación del servidor.

Tabla 6.3. Información sobre la conexión WiFi

SSID	ESP8266-WiFi
Contraseña	esp123456
Dirección MAC	e0:e6:2e:7b:d4:7b
Dirección IP	192.168.4.2
Puerta de enlace	192.168.4.1
Máscara de subred	255.255.255.0
DNS	192.168.4.1
Frecuencia	2.4 GHz
Velocidad de conexión	54 Mbps
Seguridad	WPA/ WPA2 PSK
Dirección IPv6	fe80::525c:d653:7ba:b7cb

Para procesar cada string se utiliza la librería *string.h* (declarada dentro de *osapi.h* en el SDK) que contiene múltiples funciones para el manejo de cadenas.

La cadena recibida en cada comunicación se trata de una petición HTTP con método GET que posee la siguiente estructura:

```
GET /?comando=valor HTTP/1.1
Host: 192.168.4.1:8266
Connection: keep-alive
Accept: */*
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
Referer: http://192.168.4.1:8266/
Accept-Encoding: gzip, deflate
Accept-Language: es-ES,es;q=0.9,en;q=0.8
```

Una vez recibida la cadena tras una solicitud, esta es comparada con la función *strncmp*. Esta es usada para encontrar una coincidencia dentro de la lista de comandos permitidos que se encuentran en la tabla 6.4.

Tabla 6.4. Comandos aceptados en las solicitudes HTTP.

COMANDO	STRING	FUNCIÓN
cmd00	GET / HTTP/1.1	Solicitud de la página desde el browser
cmd1	GET /?num1	Mover motor 1
cmd2	GET /?num2	Mover motor 2
cmd3	GET /?num3	Mover motor 3
cmd4	GET /?num4	Mover motor 4
cmd5	GET /?num5	Mover motor 5
cmd6	GET /?num6	Mover motor 6
cmd7	GET /?gripon	Habilitar pinza neumática
cmd8	GET /?gripoff	Deshabilitar pinza neumática
cmd9	GET /?PR1	Configurar timer PR1
cmd11	GET /?Kp1	Factor proporcional PID 1
cmd12	GET /?Ki1	Factor integral PID 1
cmd13	GET /?Kd1	Factor derivativo PID 1
cmd14	GET /?Kp2	Factor proporcional PID 2
cmd15	GET /?Ki2	Factor integral PID 2
cmd16	GET /?Kd2	Factor derivativo PID 2
cmd17	GET /?Kp3	Factor proporcional PID 3
cmd18	GET /?Ki3	Factor integral PID 3
cmd19	GET /?Kd3	Factor derivativo PID 3
cmd20	GET /?Kpro1	Constante de calibración proporcional 1
cmd21	GET /?Koffset1	Constante de calibración de offset 1
cmd22	GET /?Kpro2	Constante de calibración proporcional 2
cmd23	GET /?Koffset2	Constante de calibración de offset 2
cmd24	GET /?Kpro3	Constante de calibración proporcional 3
cmd25	GET /?Koffset3	Constante de calibración de offset 3
cmd26	GET /?puente1=ON	Habilitar Puente H 1
cmd27	GET /?puente1=OFF	Deshabilitar Puente H 1
cmd28	GET /?puente2=ON	Habilitar Puente H 2
cmd29	GET /?puente2=OFF	Deshabilitar Puente H 2
cmd30	GET /?puente3=ON	Habilitar Puente H 3
cmd31	GET /?puente3=OFF	Deshabilitar Puente H 3

De haber una coincidencia, en esta solicitud es de principal interés obtener los parámetros *comando=valor* para algunas peticiones. Como es el caso de los movimientos de los motores y la configuración de parámetros.

Para ello se utiliza la función *strncpy* para copiar al menos 30 caracteres en una variable auxiliar y de esta manera manipular una cadena menor.

Una vez recortada la cadena, se utiliza la función *strtok* para separar cada parte del mensaje en subvariables o tokens. Colocando como selector el caracter de espacio ' ' e igual '='.

El código del microcontrolador está, además de por comandos, estructurado por un total de catorce funciones que se ejecutan según sea el caso. Estas serán explicadas en los siguientes apartados.

6.2.2. Función server_recv

Esta es principalmente la función de respuesta ante recepción de información en el servidor.

En ella se obtiene la cadena recibida en el canal TCP para la posteriór identificación de los comandos. Para esto, se realiza una evaluación de cadenas mediante la función *strncmp* de la librería *string.h* para buscar el identificador del comando.

Posteriormente, de acuerdo al resultado del comando, esta ejecutará una instrucción contenida en un bloque condicional único para cada caso.

6.2.3. Función server_sent

Función de callback cuando se envía algo desde el servidor. Esta función enciende un LED de notificación en el pin D2 de la tarjeta de desarollo.

6.2.4. Función server_discon

Función de callback cuando se desconecta un canal TCP. Esta borra por completo el canal y vuelve a habilitar el servidor TCP en el puerto 8266, seguidamente apaga un led en el pin D4 de la tarjeta de desarrollo.

6.2.5. Función server_listen

Esta función se utiliza para declarar cuáles serán las funciones de callback para 'escuchar' las conexiones en el canal TCP. Para hacer esto, se utiliza las funciones *espconn_regist_sentcb* y *espconn_regist_disconcb* del NON-OS SDK.

6.2.6. Función server_recon

Esta función se encarga de reconectar el servidor en caso de una falla en el establecimiento del canal TCP. Esto ocurre raramente. No obstante, se incluye como medida cautelar ante posibles errores en la comunicación.

6.2.7. Función init_tcp

Dentro de la estructura de configuración de la conexión, se define el apuntador, el tipo (TCP) y el puerto para establecer la conexión.

Luego, con las función *espconn_regist_connectcb* se define el callback para 'escuchar' el servidor.

Finalmente, para crear el servidor, se utiliza *espconn_accept* y se enciende un led en el pin D4 de la tarjeta de desarrollo.

6.2.8. Función ap_config_func

Esta función configura la conexión que realiza el microcontrolador por medio de la señal WiFi. Se selecciona el modo de punto de acceso (SOFTAP MODE) en donde se da valor a la SSID y contraseña de la conexión a establecer. Como SSID se colocó 'ESP8266-WiFi' y contraseña 'esp123456' con seguridad WPA/WPA2 PSK, como se indica en la tabla 6.2.1.

6.2.9. Función gpio_init

Esta función configura los pines D2 y D4 como salidas digitales. También configura el UART2 de 8 bits con bit de parada y 115200 baudios.

6.2.10. Función mover_motor

Esta función recibe la cadena y la separa por secciones para tomar su valor.

De la cadena recibida, toma los primeros 30 caracteres y se procesa como se explica a continuación:

INFORMACIÓN DE LA TRAMA HTTP ENVIADA POR EL CLIENTE (BROWSER):

El cliente realiza un request GET cuya cabecera contiene lo siguiente:

```
GET /?num1=XXX HTTP/1.1
```

En este mensaje el valor XXX debe ser aislado para ser procesado. Esto se hace mediante la función *strtok* de la libreria *string.h*. Se colocaran como selectores de token, los caracteres ' ' y '='.

```
1ra vez:
GET /?num1=XXX HTTP/1.1
   ^     token=GET
2da vez:
/?num1=XXX HTTP/1.1
      ^    token=/?num1

3ra vez:
XXX HTTP/1.1
   ^   token=XXX     y con esto ya se obtiene el valor de XXX.
```

Una vez obtenido ese valor, este es convertido en entero con la función *atoi* de la librería *stdlib.h* o convertido en float con la función *atof* (o en su defecto, la función personalizada *myatof*). Seguidamente, se aplica la conversión según la expresión 6.1 obtenida de la ecuación 4.1.

$$Valor_{digital} = (\frac{Valor_{grados}}{K_{pro}}) - K_{offset} \tag{6.1}$$

Una vez convertido el valor, este se desplaza 6 bits a la izquierda para ordenarlos.

Como se desea transmitir una trama como la indicada en la tabla 4.1.2, es necesario que el valor obtenido anteriormente se desplace 8 bits a la derecha para así dar valor al dato high. El dato low no tiene mayores modificaciones, así que esta función termina su trabajo enviando el arreglo con la instrucción *uart0_tx_buffer* del NON-OS SDK.

6.2.11. Función mover_motor_2

Esta función realiza la misma operación que *mover_motor* (6.2.10). Sin embargo, la calibración de offset y proporcionalidad ya están predefinidas y no pueden cambiarse. Esto se hace para dar movimiento a los motores de la mano del robot (D, E, F).

6.2.12. Funciones cambiar_constante, parametro_pid, puenteH

Estas funciones se utilizan para cambiar el valor de las constantes de calibración presentes en los motores A, B y C (cambiar_constante). También configura los parámetros del PID como Kp, Ki, Kd de los motores A, B y C (parametro_pid). El registro PR1 del timer del controlador y por último habilita o deshabilita los puentes H de los motores A, B y C (puenteH).

Estas funciones realizan un procesado similar a *mover_motor* y dan como resultado el número float de interés extraído de la cadena recibida para el caso de la función (cambiar_constante), o el envío directo de la trama generada con ayuda de la función *uart0_tx_buffer*

6.2.13. Función myatof

Esta función se usa para convertir un caracter ASCII a un valor float. No se utiliza la función atof, de la librería *stdlib.h*, ya que genera un error de compilación debido a que las funciones *malloc* y *free* están declaradas dentro del SDK como *os_malloc* y *os_free* y escritas de otra manera el compilador no las reconoce. Por lo tanto, se requirió un algoritmo que realizara el mismo proceso.

6.3. Interfaz

CAPÍTULO VII

VALIDACIÓN RESULTADOS

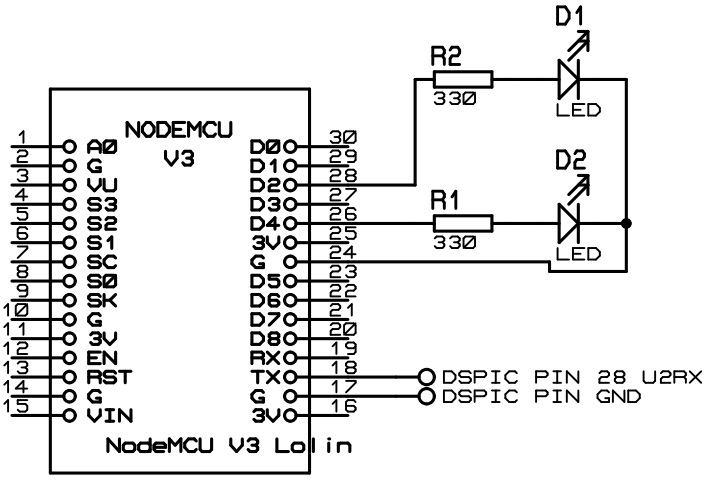


Figura 7.1. Esquemático de conexión entre ESP-8266 y dsPIC30F3011

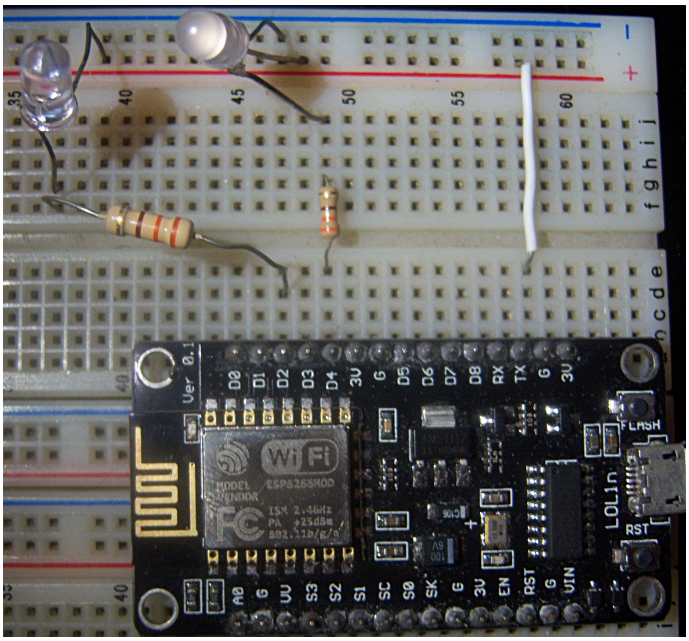


Figura 7.2. Fotografía del montaje de la tarjeta de desarrollo.

CAPÍTULO VIII

RESULTADOS

CAPÍTULO IX

CONCLUSIONES

CAPÍTULO X

RECOMENDACIONES

Apéndice I

TÍTULO DEL ANEXO

Apéndice II

TÍTULO DEL ANEXO

Apéndice III

TÍTULO DEL ANEXO

REFERENCIAS

Brea, E. (2006). *C?lculo Operacional* (1ra ed.). Caracas: Escuela de Ingenier?a El?ctrica, Universidad Central de Venezuela.

Brigham, E. O. (1974). *The fast Fourier transform.* Englewood Cliffs, N.J.: Prentice Hall.

Emin, G. (2003). External control of puma 700 series robot based on the communication protocols lun and ddcmp. *Orta Dogu Teknik ?niversitesi (Ankara, Turkey).* Descargado de `https://www.researchgate.net/publication/35715745_External_control_of_PUMA_700_series_robot_based_on_the_communication_protocols_LUN_and_DDCMP`

Labrador, A. (2018). *Diseño de un equipo para el control y monitoreo de un motor asincrónico usando una aplicación móvil.* Caracas: Escuela de Ingeniería Eléctrica, Universidad Central de Venezuela.

Marcano, J. (2013). *Implementación de sistema de programación de trayectorias para el brazo manipulador ma2000.* Caracas: Escuela de Ingeniería Eléctrica, Universidad Central de Venezuela.

Miller, R., Eriksson, L., Fleisher, L., Wiener-Kronish, J., y Young, W. (2009). *Anesthesia e-book.* Elsevier Health Sciences. Descargado de `https://books.google.co.ve/books?id=NdtScV4iZGYC`

Valderrama, J. (2018). *Diseño de un conjunto de prácticas para la configuración y uso básico del microcontrolador esp8266.* Caracas: Escuela de Ingeniería Eléctrica, Universidad Central de Venezuela.