

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, octubre 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, octubre de 2019

A quien deseas dedicar este trabajo

*A las tantas horas de esfuerzo
a las voluntades que desfallecen pero que se fortalecen
a las ínfimas muestras de orgullo y pasión
a las cualidades que emergen ante nuestros ojos
a la incertidumbre que a veces agobia e inspira*

RECONOCIMIENTOS Y AGRADECIMIENTOS

Ismael J. Bautista G.

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

Tutor Académico: Tamara Pérez. Tesis. Caracas, Universidad Central de Venezuela. Facultad de Ingeniería. Escuela de Ingeniería Eléctrica. Mención Electrónica Computación y Control. Año 2019, 110 pp.

Palabras Claves: Palabras clave.

Resumen.- Escribe acá tu resumen

ÍNDICE GENERAL

RECONOCIMIENTOS Y AGRADECIMIENTOS	III
ÍNDICE GENERAL	VIII
LISTA DE FIGURAS	XII
LISTA DE TABLAS	XV
LISTA DE SIGLAS Y ACRÓNIMOS	XVII
INTRODUCCIÓN	1
MARCO REFERENCIAL	2
1.1. El problema	2
1.1.1. Planteamiento del problema	2
1.1.2. Justificación	2
1.1.3. Objetivos	3
1.1.4. Alcance y limitaciones	3
1.1.5. Análisis de factibilidad	4
1.2. Antecedentes	4
MARCO TEÓRICO	8
2.1. Fundamentos sobre brazos manipuladores de tipo pick and place.	8
2.1.1. PUMA.	11
2.1.2. Canadarm	12

2.1.3. Brazo manipulador MA2000.	13
2.2. TÉCNICAS Y METODOLOGÍAS DE ACCESO REMOTO EN BRAZOS MANIPULADORES	14
2.3. Acceso remoto.	14
2.3.1. Radiofrecuencia (RF).	15
2.3.2. Bluetooth y Wi-Fi.	18
2.3.3. Telefonía móvil	22
DESCRIPCIÓN DEL SISTEMA BRAZO MANIPULADOR MA2000.	23
3.1. ACCIONES DEL SISTEMA DE CONTROL DEL BRAZO MANIPULADOR MA2000.	23
3.2. Software Robocom	29
3.2.1. Elementos fundamentales	30
3.2.2. Generación de la trama de comunicación	33
3.3. Unidad de control	37
3.4. PWM	37
DISEÑO DEL SISTEMA DE ACCESO REMOTO	40
4.1. Selección de dispositivos	40
4.1.1. Requerimientos	40
4.1.2. Raspberry Pi 3 (modelo B+)	44
4.1.3. Cypress CYW54907	45
4.1.4. ESP 8266	46
4.1.5. ESP 32	48
4.2. Selección	49
SISTEMA DE ACCESO REMOTO	52

5.1.	Detalles de software	52
5.1.1.	Configuración del IDE Atom PlatformIO	53
5.1.2.	Diseño de página web	55
5.2.	Detalles de hardware	57
5.3.	Detalles de firmware	59
5.3.1.	Programación en lenguaje Lua	59
5.3.2.	Programación en lenguaje Simba	60
5.3.3.	Programación en lenguaje C	60
5.4.	Uso de librerías en lenguaje C	61
5.5.	Kit de desarrollo de software: Espressif NON OS ESP-8266 SDK .	62
5.5.1.	osapi.h	62
5.5.2.	user_interface.h	63
5.5.3.	driver/uart.h	64
5.5.4.	espconn.h	64
5.5.5.	gpio.h	65
5.5.6.	eagle_soc.h	65
5.6.	Contenido del programa	66
5.6.1.	user_main.c	67
5.6.2.	user_config.h	67
5.6.3.	funciones.h y funciones.c	67
5.6.4.	pagina.h	68
5.6.5.	Compilado	68
5.7.	Funcionamiento	69
5.7.1.	Pseudocódigo	70
5.7.2.	Función server_recv	74

5.7.3. Función server_sent	75
5.7.4. Función server_discon	75
5.7.5. Función server_listen	76
5.7.6. Función server_recon	76
5.7.7. Función init_tcp	76
5.7.8. Función ap_config_func	77
5.7.9. Función gpio_init	77
5.7.10. Función mover_motor	78
5.7.11. Función mover_motor_2	80
5.7.12. Funciones cambiar_constante, parametro_pid, puenteH	80
5.7.13. Función myatof	81
RESULTADOS	82
6.1. Interfaz	82
6.1.1. Página web alojada en servidor	82
6.1.2. Solución a posibles incovenientes	90
6.1.3. Aplicación Android	92
6.2. Montaje	95
CONCLUSIONES	96
RECOMENDACIONES	97
REFERENCIAS	98

LISTA DE FIGURAS

2.1. Estructura básica de un brazo robótico.	9
2.2. Anatomías básicas de brazos Pick and place.	10
2.3. Representación de un brazo PUMA 560	11
2.4. Modelos del brazo Canadarm.	12
2.5. Robot MA2000	13
2.6. Estructura del sistema de control de brazo MA 2000.	14
2.7. KYL-500S.	15
2.8. nRF24L01.	16
3.1. Esquema de indentificación de los motores	24
3.2. Límites máximos de movimiento para el motor A.	25
3.3. Límites máximos de movimiento para el motor B.	25
3.4. Límites máximos de movimiento para el motor C.	25
3.5. Límites máximos de movimiento para el motor D.	26
3.6. Límites máximos de movimiento para el motor E.	26
3.7. Límites máximos de movimiento para el motor F.	26
3.8. Estructura del sistema de control de brazo MA 2000.	27
3.9. Tarjeta controladora con microcontrolador dsPIC30F3011	28
3.10. Tarjeta del convertidor USB/Serial FTI232	28
3.11. Puentes H L296.	29
3.12. Vista inicial al ejecutar el software Robocom.	30
3.13. Panel inferior de Robocom	31

3.14. Función 'Setpointer' del software Robocom.	32
3.15. onfiguración de constantes de calibració	33
3.16. Vista de trama transmitida por el software Robocom	36
3.17. Pines de conexión del módulo controlador original.	39
4.1. Esquema de comunicación original	41
4.2. Esquema de comunicación inalámbrica.	41
4.3. Diagrama de bloques del sistema por conexión Wi-Fi	43
4.4. Tarjeta de desarrollo Raspberry Pi 3 Model B+	44
4.5. Tarjeta de desarrollo CYW954907AEVAL1F de Cypress.	45
4.6. ESP8266 con chip ESP-01	46
4.7. ESP 32 de Espressif.	48
4.8. Comparación entre los dispositivos considerados	50
5.1. Ventana principal del IDE Atom PlatformIO	53
5.2. Creación de nuevo proyecto	54
5.3. Compilación y carga del programa en la tarjeta.	55
5.4. Vista del software Brackets para el diseño de la página web.	56
5.5. Placa del microcontrolador ESP-8266.	57
5.6. Esquemático de conexión entre ESP-8266 y dsPIC30F3011	58
5.7. Montaje del prototipo.	59
5.8. Pinout de sistema embebido con microcontrolador ESP8266.	66
5.9. Compilado del programa.	68
5.10. Ejemplo de comunicación cliente- servidor.	69
6.1. Detección de señal Wi-Fi	82
6.2. Solicitud de clave para entrada a la página.	83

6.3.	Página web completa	84
6.4.	Encabezado de página	85
6.5.	Esquema del brazo robot y probador de trayectorias	86
6.6.	Movimiento del brazo en tiempo real	88
6.7.	Configuración de parámetros generales	89
6.8.	Vista de chrome:// net-internals	91
6.9.	Aplicación vista desde el menú.	93
6.10.	Vista principal de la aplicación Android.	94
6.11.	Esquemático de conexión entre ESP-8266 y dsPIC30F3011	95

LISTA DE TABLAS

2.1.	Ancho de banda de datos según las versiones.	19
2.2.	Clases según potencia de transmisión y alcance.	19
2.3.	Versiones de Wi-Fi.	21
2.4.	Comparación entre Bluetooth y Wi-Fi.	21
3.1.	Estructura general de la trama de comunicación.	33
3.2.	Comandos emitidos desde el software Robocom	35
3.3.	Comandos enviados al generar trama por defecto.	36
4.1.	Estimación de requerimientos para el diseño.	44
5.1.	Especificaciones de software	52
5.2.	Especificaciones del software empleado en la página web.	56
5.3.	Especificaciones de hardware de la tarjeta de desarrollo.	57
5.4.	Registros de GPIO del ESP-8266. Fuente: Espressif.	65
5.5.	Información sobre la conexión WiFi	72
5.6.	Comandos aceptados en las solicitudes HTTP.	73
5.7.	Descripción de Función server_recv.	74
5.8.	Descripción de Función server_sent.	75
5.9.	Descripción de Función server_discon	75
5.10.	Descripción de Función server_listen	76
5.11.	Descripción de Función server_recon	76
5.12.	Descripción de Función init_tcp.	76

5.13. Descripción de Función ap_config_func	77
5.14. Descripción de Función gpio_init	77
5.15. Descripción de Función mover_motor	78
5.16. Descripción de Función mover_motor_2	80
5.17. Descripción de Función cambiar_constante	80
5.18. Descripción de Función parametro_pid	80
5.19. Descripción de Función puenteH	81
5.20. Descripción de Función myatof	81

LISTA DE SIGLAS Y ACRÓNIMOS

API: Application Programming Interface.

GPIO: General Purpose Input Output.

IDE: Integrated Development Environment.

IoT: Internet of Things.

ISM: Industrial, Scientific and Medical Radio Band.

JS: JavaScript

OS: Operative System.

PID: Proportional Integral Derivative controller.

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

PWM: Pulse Width Modulation.

RF: Radiofrecuencia

RX: Recepción.

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

SoC: System on Chip.

TX: Transmisión.

UART: Universal Asynchronous Receiver Transmitter.

WPA: Wi-Fi Protected Access.

WPAN: Wireless Personal Area Network.

SSH: Secure Shell.

SFTP: SSH File Transfer Protocol. Protocolo Seguro de Transferencia de Archivos.

FTP: File Transfer Protocol. Protocolo de Transferencia de Archivos.

SSHFS: Secure Shell File System.

rsync:

CIFS: Common Internet File System

SMB: Server Message Block

MAC: Media Access Control

CAN: Controller Area Network

LCID: Laboratorio de Control, Investigación y Digitales

ISS: International Space Station

SRMS: Shuttle Remote Manipulator System

NRCC: National Research Canada Council

IBM: International Business Machines

BBC: British Broadcasting Corporation

GPRS: Global System for Mobile Communications

UMTS: Universal Mobile Telecommunications System

CAE Electronics:

DSMA Atcon:

IEEE: Institute of Electrical and Electronics Engineering

SIG: Special Interest Group

HMI: Human-Machine Interface. Interfaz Humano-Máquina

CISA: Certified Information Systems Auditor.

NIST: National Institute of Standards and Technology. Instituto Nacional de Estándares y Tecnología.

ADC: Analog-to-digital converter. Convertidor analógico-digital

HDMI: High-Definition Multimedia Interface («interfaz multimedia de alta

definición »).

BT: Bluetooth

BR: Bluetooth Basic Rate.

EDR: Bluetooth Enhanced Data Rate.

BLE: Bluetooth Low Energy.

RTOS: Real Time Operative System. Sistema Operativo de Tiempo Real.

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

MARCO REFERENCIAL

En este capítulo se expondrá el problema que motiva a la investigación de este trabajo, considerando un brazo manipulador ubicado en las instalaciones del Laboratorio de Control, Instrumentación y Digitales (LCID) dentro de la Universidad Central de Venezuela. Se menciona la justificación, objetivos, alcance, factibilidad y antecedentes para este proyecto.

1.1. El problema

1.1.1. 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.

1.1.2. 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) [1], es siempre importante una propuesta para mejorar la utilidad del mismo utilizando nuevas tecnologías relacionadas con accesos remotos.

1.1.3. Objetivos

Objetivo general

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

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.

1.1.4. 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.

1.1.5. 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.

1.2. Antecedentes

En primer lugar se tiene el trabajo de grado presentado en mayo de 2013 por Juan Marcano en la Universidad Central de Venezuela, titulado "Implementación de sistema de programación de Trayectorias para el brazo manipulador MA2000".
[1]

Este proyecto se trató de la implementación de un sistema que permitió la comunicación con el brazo manipulador a través de un software instalado en un computador y de esta manera realizar las acciones de movimiento y programación de trayectorias. Para alcanzar esto, se realizó un controlador PID para el manejo

de la posición de los motores con ayuda de un microcontrolador Microchip dsPIC y las librerías DSP compatibles con el mismo.

El software, desarrollado en lenguaje C++, es capaz de comunicarse con el brazo manipulador a través de un convertidor USB a Serial conectado en uno de los puertos del microcontrolador. De esta manera, se puede realizar desde el computador el ajuste dinámico del controlador de posición y la respuesta dinámica de cada articulación del robot por medio de una interfaz gráfica.

Como producto de este trabajo se generó un manual con el código fuente de las aplicaciones y demás características sobre los motores, además de obtener la documentación necesaria a través del documento de grado.

Este sistema ayudó a comprender el funcionamiento del software, hardware y las acciones de movimiento del brazo manipulador, cuya información fue de gran importancia para la ejecución de la propuesta de mejora planteada en el presente trabajo de grado.

En noviembre de 2018, Alexis Labrador presentó un trabajo de grado titulado "Diseño de un equipo para el control y monitoreo de un motor asincrónico usando una aplicación móvil.^{en} la Universidad Central de Venezuela. [2]

En este trabajo se realizó una aplicación móvil basada en una plataforma en línea de Internet de las Cosas (Ubidots for Education). En la cuál se aloja la interfaz de usuario que permitió monitorear las variables de tensión, corriente y otros parámetros de interés de forma remota usando una arquitectura en red estrella para programar la plataforma.

Se utilizó un ordenador de placa reducida Raspberry Pi 3 modelo B en el

cuál se desarrolló la aplicación programada en lenguaje Python, algunos relés para el accionamiento de las etapas de potencia, un sensor para medir corriente y diversos componentes electrónicos. Desde el dispositivo conectado a internet fue posible adquirir toda la información captada por las unidades de monitoreo.

Aunado a esto, desde el sistema implementado se pueden visualizar algunos modos de aviso que requiere el sistema durante el accionamiento, control y monitoreo del motor con ayuda del programa contenido en la placa Raspberry Pi.

El proyecto es de utilidad para la propuesta de este trabajo en cuestión ya que muestra una posible solución de comunicación remota y manejo de variables de una entidad física a través de un portal web, tomando como iniciativa la implementación de un sistema basado en el Internet de las Cosas.

Jorge Valderrama presentó el trabajo de grado titulado "Diseño de un conjunto de prácticas para la configuración y uso básico del microcontrolador ESP8266" en la Universidad Central de Venezuela, en el mes de mayo de 2018. [3]

El trabajo consistió en la utilización y configuración de una tarjeta de desarrollo de bajo costo económico basada en el microcontrolador ESP8266 de la empresa Espressif por medio de prácticas desarrolladas con sus periféricos y otros elementos como: UART, GPIO, Interrupciones, Timer y Wifi. En él se explican los modos de configuración de cada uno de estos junto con guías de programación y ejemplos para implementación.

Por otra parte, el trabajo cuenta con soporte para la programación de este dispositivo utilizando el IDE Atom Platformio, compatible con las librerías del SDK. Toda la información utilizada para la documentación de este proyecto está

basada en el contenido obtenido de los múltiples manuales, hojas de datos y Kits de Desarrollo de Software proporcionados por Espressif.

El producto de este trabajo fue generar una guía comentada en español utilizando las distintas APIs del Non Operative System Software Development Kit (NON OS-SDK) con el propósito de documentar las funciones más relevantes del Kit de Desarrollo de Software que permitan poner en funcionamiento el dispositivo usando únicamente lenguaje C.

Es de utilidad para el presente trabajo ya que propone una solución con relación al beneficio/costo económico porque permite desarrollar funciones por medio del SDK que son implementadas de forma similar en dispositivos de mayor gama tecnológica, pudiendo utilizarse como una potente herramienta para realizar aplicaciones de IoT. Además, ofrece la posibilidad de programar este equipo usando un IDE de código abierto y un lenguaje de programación estándar como lo es el lenguaje C.

CAPÍTULO II

MARCO TEÓRICO

En este capítulo se hablará de los conceptos referentes a algunos brazos manipuladores de tipo pick and place con morfologías similares al MA2000, incluyendo información fundamental sobre este último. En las secciones siguientes se hablará sobre algunas técnicas y metodologías de acceso remotos aplicadas en brazos manipuladores.

2.1. Fundamentos sobre brazos manipuladores de tipo pick and place.

Brazo robot pick and place

Un brazo manipulador, o comúnmente llamado "brazo robot", es un tipo de brazo mecánico que posee un sistema electrónico programable para realizar acciones de control a un conjunto de motores. 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. Cada grado de libertad de un brazo manipulador se describe como una unión que permite al robot doblar, rotar o trasladar la parte móvil de este. [4]

A menudo, estos grados de libertad suelen describirse como partes del cuerpo humano. Un ejemplo de esto se observa en la figura 2.1 donde cada grado de

libertad posee un nombre con relación a términos antropomórficos como "cintura", "mano", "antebrazo", "muñeca", "brazo", "codo", "hombro", entre otros.

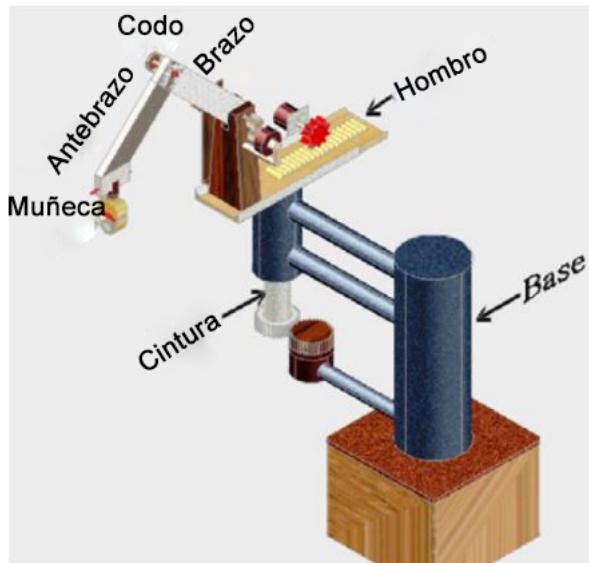


Figura 2.1. Estructura básica de un brazo robótico.

Fuente: [4]

Existen de tipo: esférico, articulado, SCARA, cilíndrico y cartesiano. Como se observa en la figura 2.2.

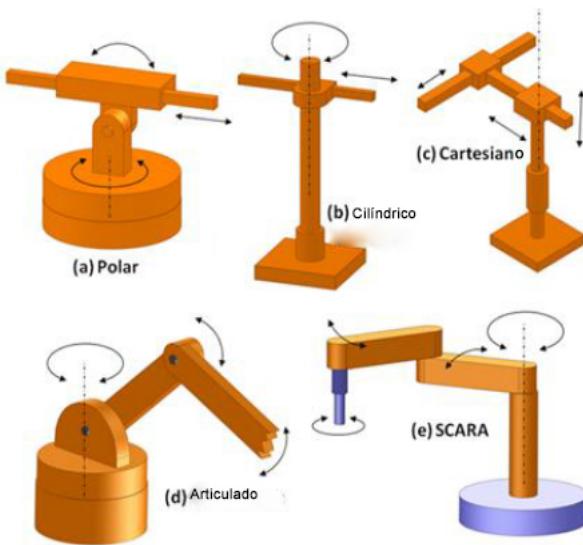


Figura 2.2. Anatomías básicas de brazos Pick and place.
 a) Polar b) Cilíndrico c) Cartesiano d) Articulado e) SCARA
 Fuente: [5]

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 2.2 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.

Brazo pick and place articulado

Los brazos manipuladores articulados constan de una base fija a una superficie y sobre ella se ejecutan las acciones de movimiento. Existen de múltiples morfologías y son comúnmente empleados los brazos de seis grados de libertad para aplicaciones industriales, científicas o médicas. Dentro de esta categoría se encuentran los brazos: PUMA, Canadarm 1, Canadarm 2 y un brazo manipulador implementado por la Open University: MA 2000; que serán explicados en las siguientes secciones.

2.1.1. PUMA.



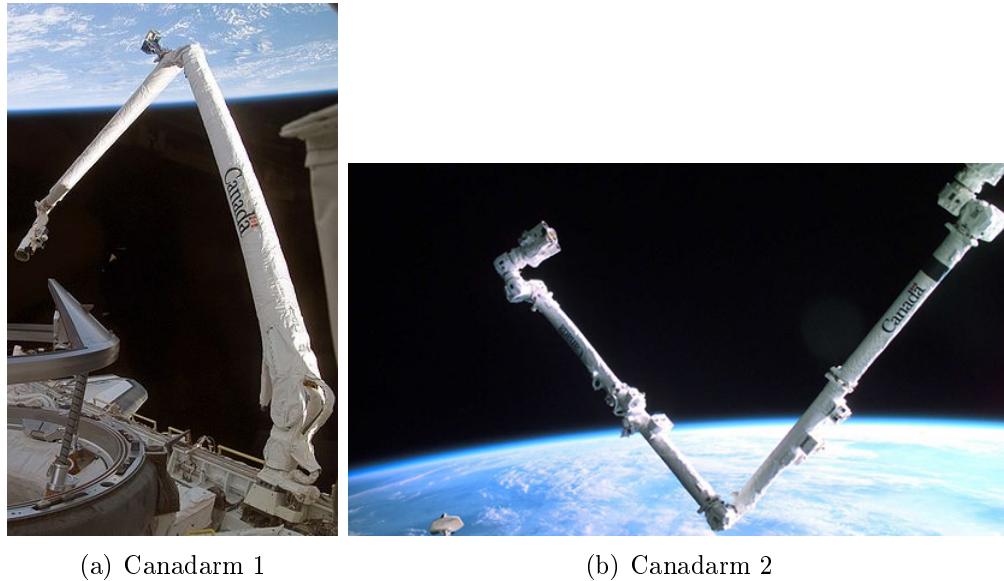
Figura 2.3. Representación de un brazo PUMA 560

Fuente: [6]

El brazo PUMA: Programmable Universal Machine for Assembly, o Programmable Universal Manipulation Arm, por sus siglas en inglés; es un robot industrial desarrollado por la empresa Unimation alrededor del año 1980 [7]. 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. En la figura 2.3 se ilustra un brazo PUMA 560, de la serie 500.

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 [8].

2.1.2. Canadarm



(a) Canadarm 1

(b) Canadarm 2

Figura 2.4. Modelos del brazo Canadarm.

Fuente: [9]

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 (ISS) y fue conocido también con el nombre Shuttle Remote Manipulator System (SRMS). [10]

2.1.3. Brazo manipulador MA2000.



Figura 2.5. Robot MA2000

Fuente: [11]

El robot de la figura 2.5 es un brazo manipulador articulado de tipo pick and place de seis grados de libertad diseñado alrededor de los años 1980s por George Carter, fundador de Labman Automation Ltd y fabricado por TecQuipment International Ltd, como una solicitud de la Open University británica. [11]

El diseño original era accionado por seis servomotores que permitían realizar control a lazo cerrado en cualquier instante. Requería adicionalmente una unidad compresora de aire para accionar la herramienta final de control (pinza neumática). Además, era controlado por computadores IBM o por microcontroladores BBC, quienes poseían el firmware para generar el movimiento de los motores. [11]

Este diseño se concibió como una herramienta de estudio para el área de la

robótica en ambientes académicos gracias a la Open University. Por otro lado, hasta la fecha de publicación de este trabajo de grado, el brazo manipulador se encuentra operando con seis motores DC y una pinza neumática. El sistema de control está implementado sobre un microcontrolador Microchip dsPIC30 que acciona un controlador PID digital en conjunto con una circuito de potencia para los motores y un software instalado en un computador para generar la trama de comunicación.

En [1], se explica detalladamente el proceso que se realizó para la optimización del sistema de control original a través de los elementos ilustrados en la figura 2.6.

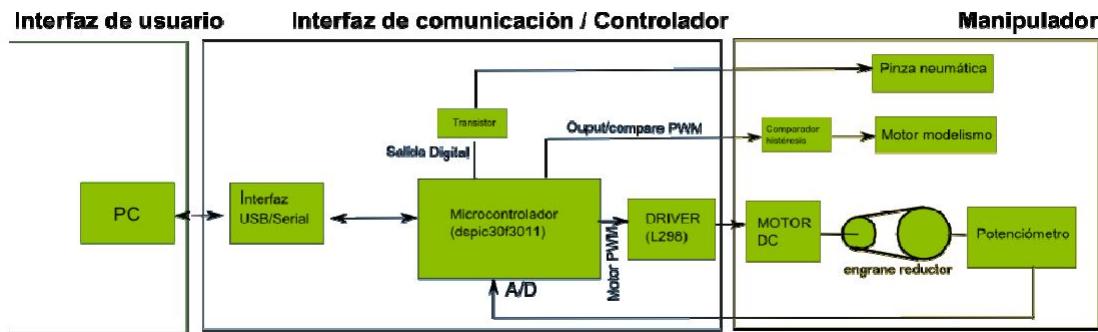


Figura 2.6. Estructura del sistema de control de brazo MA 2000.

2.2. TÉCNICAS Y METODOLOGÍAS DE ACCESO REMOTO EN BRAZOS MANIPULADORES

2.3. Acceso remoto.

El Instituto Nacional de Estándares y Tecnología de Estados Unidos (NIST) y el Departamento de Seguridad de la Nación de Estados Unidos (CISA), definen Acceso remoto como "la capacidad de los usuarios y operadores de una organización para acceder a sus recursos computacionales no públicos o sistemas y datos que residen dentro de una red protegida físicamente y/o lógicamente de locaciones

externas que puedan ser consideradas como fuera de la red de la organización"

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

2.3.1. Radiofrecuencia (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 2.8) o KYL-500S (Figura 2.7), se puede realizar el mismo proceso optimizado para el trabajo con microcontroladores.

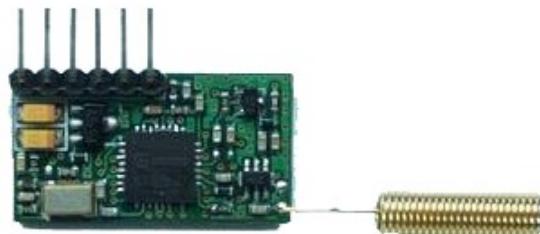


Figura 2.7. KYL-500S.
Fuente: Shenzhen KYL Communication Equipment Co., Ltd.

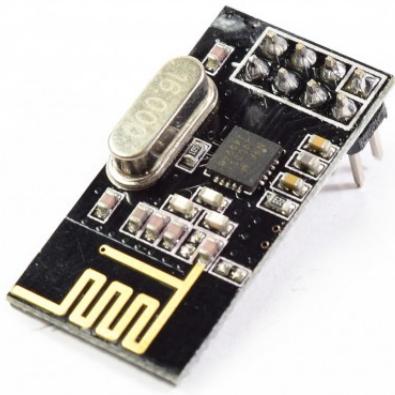


Figura 2.8. nRF24L01.

Fuente: Nordic Semiconductor.

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.

Es necesario el establecimiento de una interfaz de comunicación para la ejecución de la acción de control de forma inalámbrica, por lo tanto es común el diseño de circuitos osciladores transmisores/receptores que se colocan en el sistema a controlar y su unidad remota.

Por otro lado, una mejora de estos sistemas se obtiene al utilizar circuitos integrados como el NFR24L01 o el KYL-500S. Estos circuitos transceptores poseen la unidad de transmisión y recepción disponibles según sea la necesidad. Aunado a esto, es importante señalar que existen con la variedad de frecuencias y ancho de banda que requiera el diseñador. Por este motivo, son de gran utilidad para el diseño de accesos remotos.

Una aplicación de un modelo de acceso remoto en brazos manipuladores o de robots en general, puede verse en [4], [12] y [13].

Lo señalado en el trabajo [4] trata de un diseño e implementación de un brazo robótico controlado de forma inalámbrica con fines industriales hecho en la Universidad de Ingeniería y Tecnología en Bangladesh. En este proyecto se contempla un brazo robótico de seis grados de libertad capaz de sostener un objeto de peso específico y colocarlo en una posición deseada. Para controlarlo se estableció un sistema que recibe instrucciones vía un joystick inalámbrico.

El sistema se incluyó dentro del brazo para mayor facilidad de uso. El controlador del brazo se diseñó usando un microcontrolador ATMEGA-328P, el microcontrolador utilizado en el control remoto fue un ATMEGA-8; ambos dispositivos programados bajo el IDE de Arduino. Estos integrados se encargan de recibir la señal digital que es procesada por un módulo transceptor de comunicación RF no especificado en el documento.

La utilidad del manejo remoto de sistemas de medición puede observarse en propuestas como el trabajo [12], donde se realizó un diseño e implementación de un ambiente robótico de asistencia y monitoreo de variables físicas, controlado de forma inalámbrica usando un módulo de comunicación por RF, NFR24L01 y dos microcontroladores PIC16F77A, uno para monitoreo y manipulación de periféricos; y otro como Interfaz Humano Máquina (HMI).

De forma similar al trabajo anterior, en [13] se utilizan dos microcontroladores AT89c52 y dos módulos KYL-500S para el diseño del controlador remoto de un interruptor de pared de 13A.

La tecnología de Radiofrecuencia también involucra sistemas de comunicación vía Bluetooth o WiFi, debido a que estas ofrecen facilidades compatibles con la tecnología en vanguardia; además de permitir la conectividad a internet y ser un recurso importante en los sistemas para IoT.

2.3.2. Bluetooth y Wi-Fi.

Entre las propuestas tecnológicas existentes en las comunicaciones inalámbricas; Bluetooth y Wi-Fi son las principales opciones debido a su facilidad de acceso y la disponibilidad de software y hardware compatibles con una gran cantidad de dispositivos.

Estas tecnologías ofrecen una oportunidad para realizar sistemas de IoT, ya que las tecnologías Bluetooth y Wi-Fi están contenidas en muchos dispositivos móviles de múltiples gamas. Además, existen módulos de microcontroladores y microprocesadores que cuentan con capacidad de conectarse a una red Wi-Fi o establecer una conexión por Bluetooth. Dispositivos como: Espressif ESP-8266, Espressif ESP-32, Raspberry Pi 3, Cypress CYW954907, entre otros; son algunos ejemplos de tarjetas de desarrollo compatibles con ambas tecnologías.

Para esquematizar la importancia de ambas tecnologías, es necesario describir las características principales de estas:

Bluetooth

Es una tecnología diseñada por el grupo Bluetooth Special Interest Group (SIG), Inc, que se encarga de comercializar e implementar esta herramienta de comunicación en sus productos.

Está diseñada para redes WPAN, o Redes Inalámbricas de Área Personal y su principal funcionalidad es la transmisión de datos entre dispositivos sin requerir conectores.

Su banda de funcionamiento es la ISM de 2.4GHz en el espectro de radiofrecuencia y cuenta con un consumo de potencia reducido en sus últimas versiones.

Desde sus inicios, ha presentado mejoras en la tasa de transmisión de datos, conectividad, ancho de banda, alcance y consumo de potencia. La última versión publicada en la página web oficial es la 5.1, que cuenta con una tasa de 50Mbps. La mejora de esta característica se puede observar en la tabla 2.1. Del mismo modo, la potencia de transmisión se caracteriza por clases como se muestra en la tabla 2.2.

Tabla 2.1. Ancho de banda de datos según las versiones.

Versión	Ancho de banda
Versión 1.2	1 Mbps
Versión 2.0 + EDR	3 Mbps
Versión 3.0 + HS	24 Mbps
Versión 4.0	32 Mbps
Versión 5	50 Mbps

Tabla 2.2. Clases según potencia de transmisión y alcance.

Clase	Potencia máxima permitida	Alcance
Clase 1	100 mW	100 metros
Clase 2	2.5 mW	5-10 metros
Clase 3	1 mW	1 metro
Clase 4	0.5 mW	0.5 metro

En [14] se utilizó Bluetooth para controlar remotamente los movimientos de un brazo robot de cuatro grados de libertad. El controlador del brazo se realizó con un microcontrolador ATMEGA2560, la transmisión de las instrucciones se hizo con un control remoto de Playstation con conectividad Bluetooth.

Wi-Fi

Es una tecnología desarrollada por la Alianza Wi-Fi en los años 2000. Es un estándar diseñado para la familia del protocolo IEEE 802, para redes inalámbricas de área local y es comúnmente utilizado con el protocolo Ethernet para establecer conexiones en red a internet.

Su capacidad de alcance es elevada, esto lo hace un elemento útil en el monitoreo y control de variables en campo abierto usando sensores que cuentan con esta tecnología. Además, tiene una alta velocidad de transmisión, comparable con la comunicación cableada. Las bandas que utiliza están determinadas según los estándares de IEEE 802.11. Como aspecto adicional, posee baja latencia para comunicaciones en tiempo real. Sin embargo, dependerá del procesamiento realizado por el dispositivo en cuestión.

Una ventaja de esta tecnología ante otras de radiofrecuencia es la seguridad en el envío de datos. Cada dispositivo emisor/receptor cuenta con la posibilidad de encriptar la transmisión usando WPA o WPA2.

En la página web oficial establece la sexta generación para el año 2019 como la más actualizada. Esta cuenta con mejoras de seguridad, tasa de transmisión, alcance y fidelidad. Parte de esta información se puede observar en la tabla 2.3

En [15] se construyó un brazo robot de cuatro grados de libertad basado en IoT. El controlador del robot se realizó usando una placa de desarrollo NXP FRDM-KL25Z, conectado a un módulo ESP-8266 para conectividad Wi-Fi.

Este sistema recibe las instrucciones vía una plataforma web conectada a internet, y esta se comunica con el módulo ESP para establecer la conexión y de esta forma completar el sistema de IoT.

Tabla 2.3. Versiones de Wi-Fi.

Generación	Estándar IEEE	Tasa de enlace máxima
Wi-Fi 6	802.11ax	600-9608 Mbps
Wi-Fi 5	802.11ac	433-6933 Mbps
Wi-Fi 4	802.11n	72-600 Mbps
Wi-Fi 3	802.11g	3-54 Mbps
Wi-Fi 2	802.11a	1.5 to 54 Mbps
Wi-Fi 1	802.11b	1 to 11 Mbps

Luego de estudiar los aspectos generales de ambas tecnologías, en la tabla 2.4 puede apreciarse la comparación de estas en sus principales características.

Tabla 2.4. Comparación entre Bluetooth y Wi-Fi.

Característica	Bluetooth	Wi-Fi
Ancho de banda	Bajo	Alto
Requerimientos de hardware	Adaptador de Bluetooth en todos los dispositivos que requieran conexión.	Adaptador inalámbrico en todos los dispositivos de la red. Router inalámbrico.
Facilidad de uso	De uso simple y el intercambio de dispositivos es más sencillo.	Es más complejo y requiere configuración de hardware y software.
Rango	10 metros	100 metros
Seguridad	Menos seguro en comparación	Las características de seguridad son mejores. Posee algunos riesgos.
Consumo de potencia	Bajo	Alto
Rango de frecuencia	2.400 GHz y 2.483 GHz	2.4 GHz y 5 GHz
Flexibilidad	Soporte para un número limitado de usuarios. (Conexión por pares)	Soporte para gran cantidad de usuarios. (Conexión en red)
Técnicas de modulación	GFSK (Desplazamiento de frecuencia gausiana)	OFDM (Multiplexación por división de frecuencias ortogonales) y QAM (Modulación de amplitud en cuadratura)

2.3.3. Telefonía móvil

GPRS.

UMTS

CAPÍTULO III

DESCRIPCIÓN DEL SISTEMA BRAZO MANIPULADOR MA2000.

En este capítulo se ilustrará con mayor detalle la información sobre el brazo manipulador MA2000, tomando en cuenta la configuración original desde el software Robocom y demás detalles referentes a la trama de comunicación y el sistema de control.

3.1. ACCIONES DEL SISTEMA DE CONTROL DEL BRAZO MANIPULADOR MA2000.

Este brazo manipulador cuenta con seis grados de libertad, siendo tres correspondientes al hombro (A), codo (B) y muñeca (C), controlados con motores DC. Los grados restantes (D, E, F), son los correspondientes a la mano, y son controlados mediante servomotores. Cuenta también con una pinza neumática (pinza) como herramienta final de control. Esto se observa en las figuras 3.1.

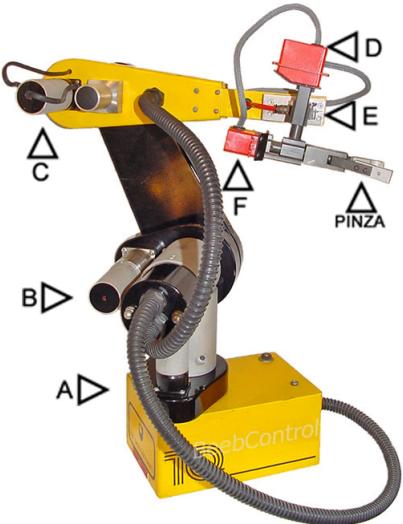


Figura 3.1. Esquema de identificación de los motores del brazo y la pinza neumática.

Limitaciones de movimiento

Los motores del brazo manipulador MA2000, identificados como A, B, C, D, E y F en la figura 3.1, tienen limitaciones físicas que impiden que tenga una total libertad de movimiento. Esto conlleva a que el sistema controlador del brazo deba evitar que los motores hagan un esfuerzo mayor al permitido limitando estos movimientos según las condiciones de cada motor. pudiendo disminuir la posibilidad de daños a la integridad del equipo para garantizar una mejor funcionalidad del sistema.

En las figuras 3.2, 3.3 y 3.4 se puede observar que para los motores A, B y C, se tienen aproximadamente 270° de movilidad en torno a sus ejes centrales.

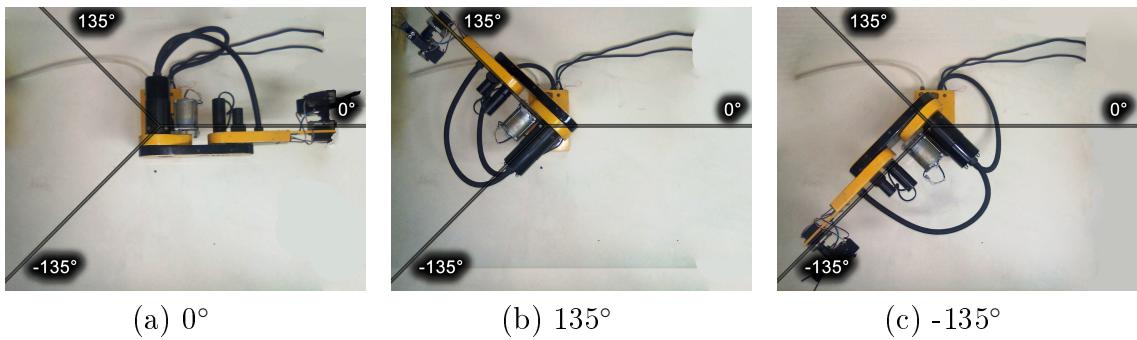


Figura 3.2. Límites máximos de movimiento para el motor A.

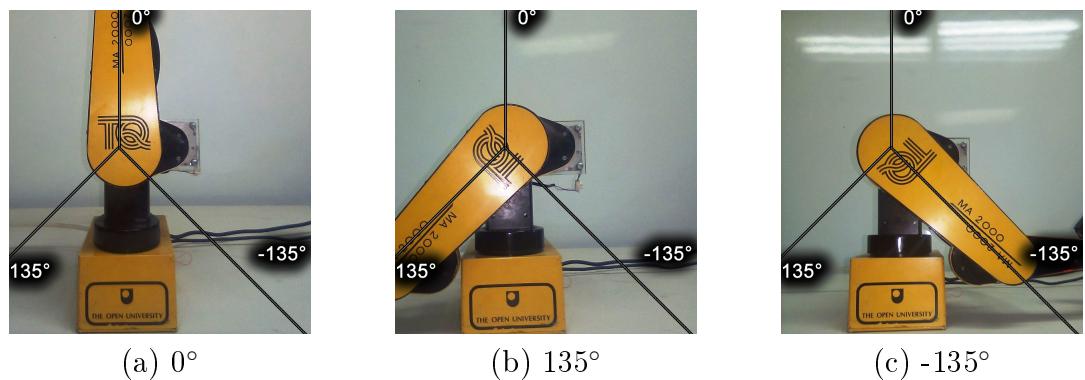


Figura 3.3. Límites máximos de movimiento para el motor B.

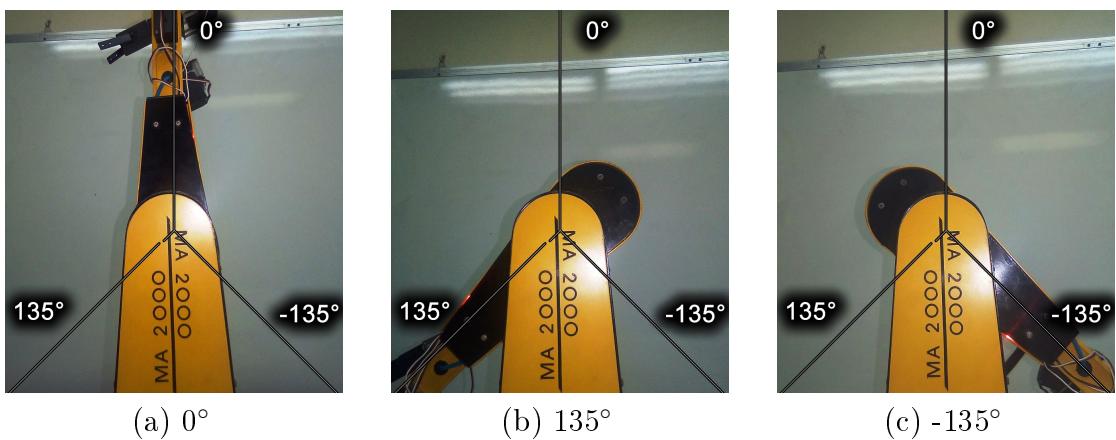


Figura 3.4. Límites máximos de movimiento para el motor C.

En las figuras 3.5, 3.6 y 3.7 se puede observar que para los motores D, E y F, se tienen aproximadamente 120° de movilidad en torno a sus ejes centrales.

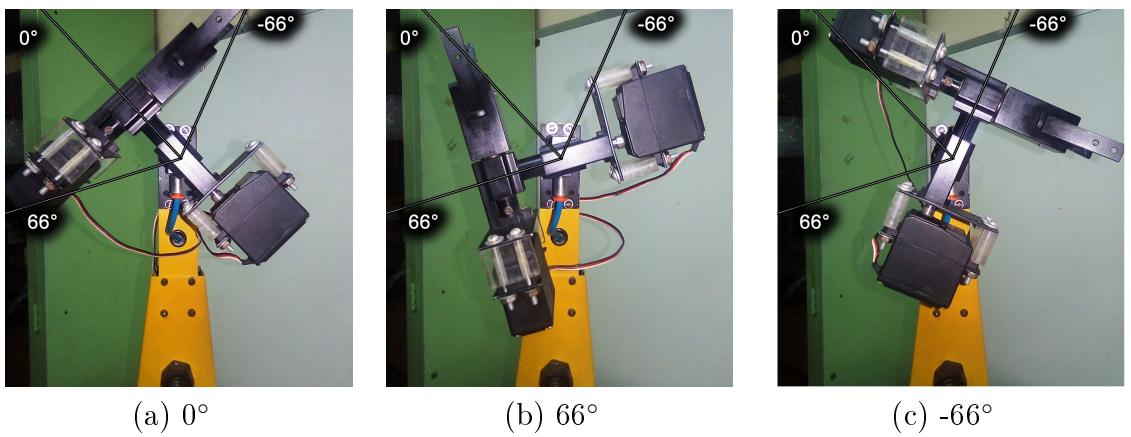


Figura 3.5. Límites máximos de movimiento para el motor D.

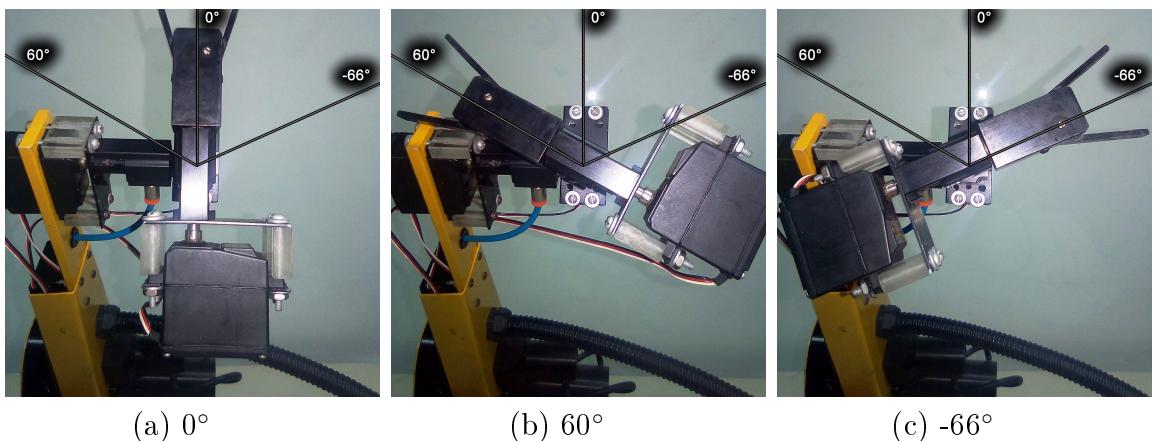


Figura 3.6. Límites máximos de movimiento para el motor E.

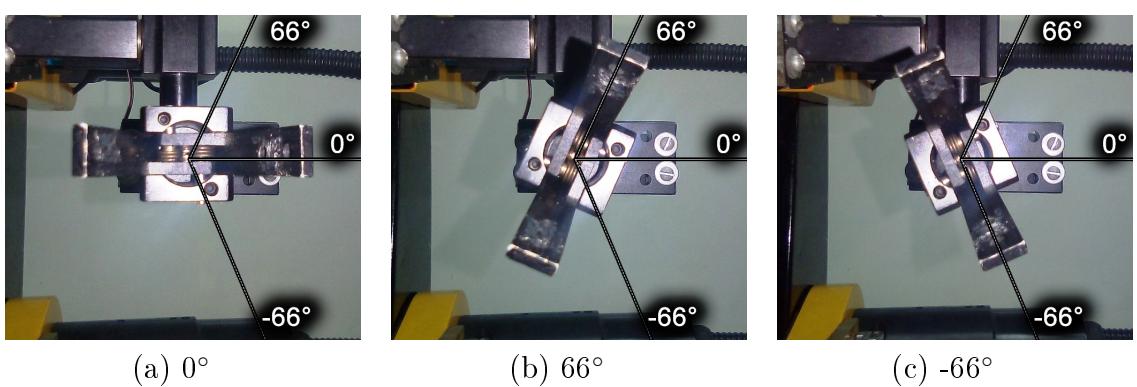


Figura 3.7. Límites máximos de movimiento para el motor F.

En la figura 3.8 se puede observar el diagrama de bloques que comprende

detalladamente cada una de las partes del sistema del brazo manipulador MA2000. En esta se puede apreciar la unidad controladora principal apoyada por el microcontrolador dsPIC30F3011, quien procesa la trama de comunicación para accionar a los actuadores de cada motor.

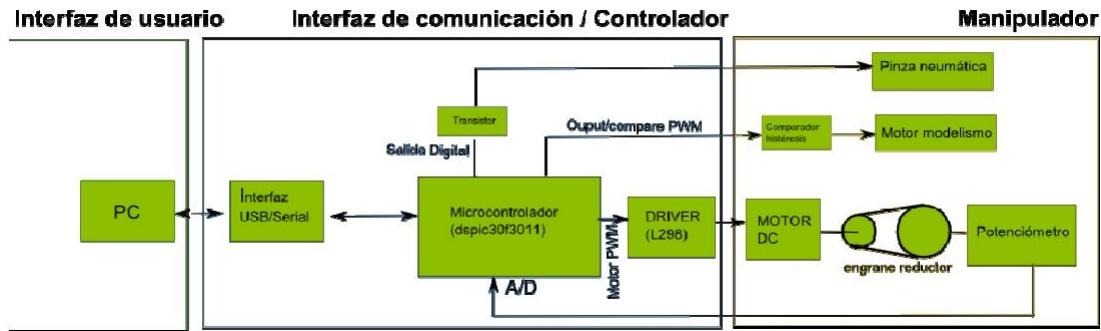


Figura 3.8. Estructura del sistema de control de brazo MA 2000.

En la figura 3.9 puede observarse con detalle la tarjeta del controlador principal del brazo robot, quien actúa como servidor recibiendo vía puerto serial las órdenes de control, para posteriormente realizar las correspondientes acciones indicadas en la trama de comunicación. Para ello, cuenta con un módulo convertidor de USB-Serial FTDI232 (figura 3.10) para una óptima comunicación con el computador a través de un software cliente. El controlador también puede responder el estado de cada articulación relacionada a los motores DC, leyendo las entradas analógicas conectadas a potenciómetros y engranajes que ayudan a conocer la posición del brazo y así ejecutar el control PID, tal como se indica en la figura 3.8.

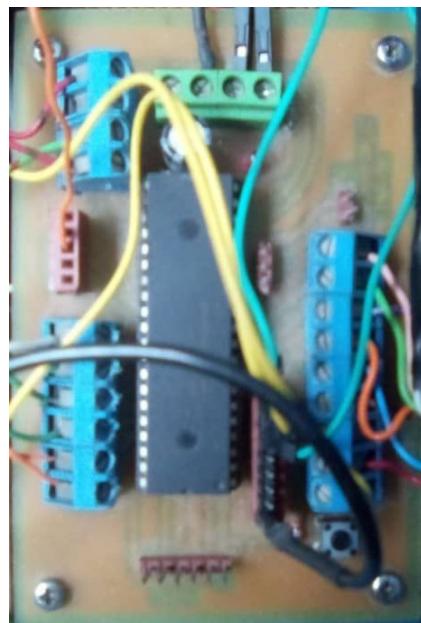


Figura 3.9. Tarjeta controladora con microcontrolador dsPIC30F3011



Figura 3.10. Tarjeta del convertidor USB/Serial FTDI232

El sistema realiza una acción de control PID en tres articulaciones principales (hombro, codo y muñeca) con ayuda de los puentes H de la figura 3.11 para permitir el movimiento de los motores en sentido horario o antihorario. Para esto se configuran los parámetros desde un software (cliente) y se envían al controlador en una trama de tres bytes vía puerto serial, la cual será descrita en secciones posteriores.



Figura 3.11. Puentes H L296.

3.2. Software Robocom

En la sección 3.1 se mencionó la utilización de un software cliente que se encarga de generar la trama de comunicación que es enviada al controlador a través del puerto serial, este programa es una interfaz gráfica de usuario desarrollada en lenguaje C++ que permite, además, configurar los parámetros de mayor importancia del robot.

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 del microcontrolador con el que se estima el tiempo de muestreo para la ejecución de las acciones de control.

3.2.1. Elementos fundamentales

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

Al ejecutar el programa, se puede observar una pantalla como la expuesta en la figura 3.12:

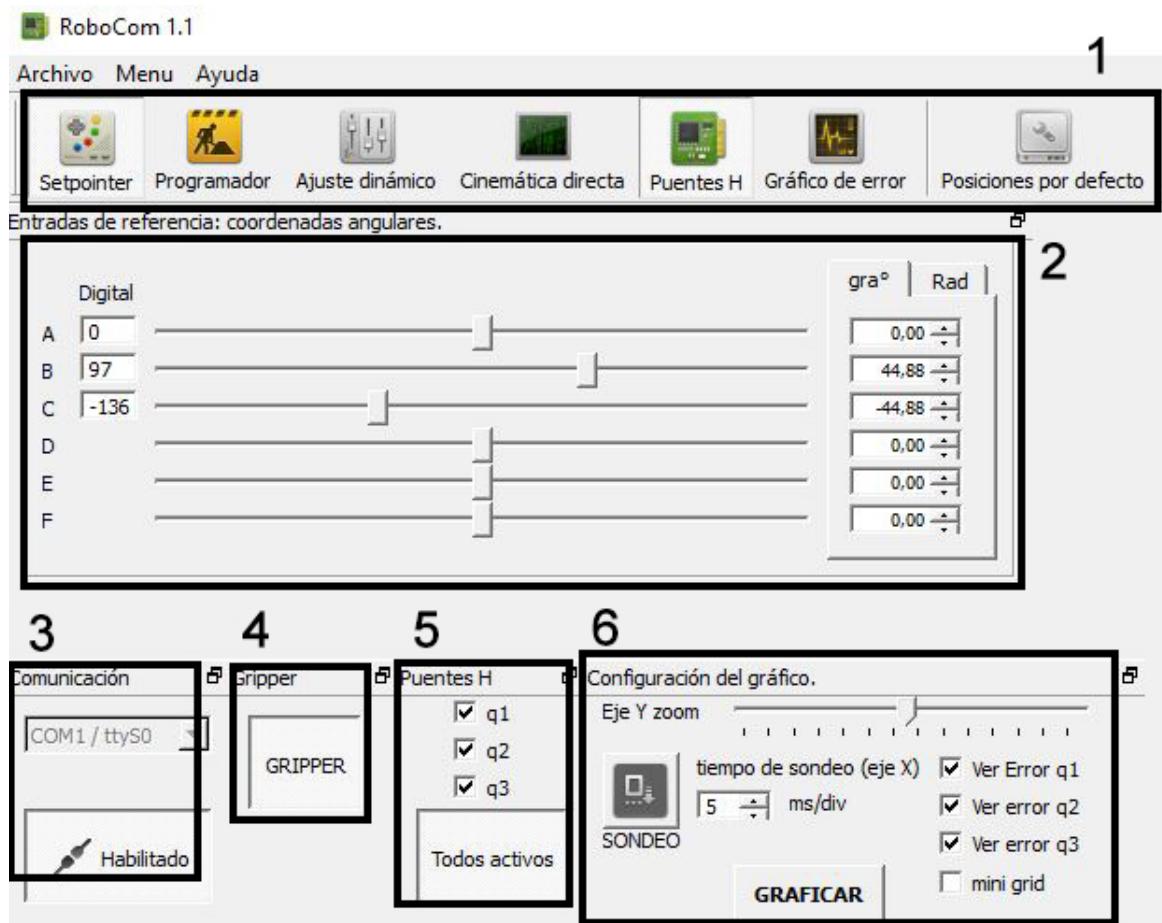


Figura 3.12. Vista inicial al ejecutar el software Robocom.

1. Panel de funciones rápidas.
2. Setpointer.
3. Opciones de comunicación.
4. Habilitar/Deshabilitar pinza neumática.
5. Habilitar/Deshabilitar puentes H.
6. Graficador de error de posición.

En el panel inferior, según se observa en la figura 3.13, 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 3.13. 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.

Un elemento importante es el 'Setpointer', indicado en la figura 3.14. Ya que esto permite ajustar la entrada de referencia al controlador y también realizar movimientos en tiempo real con el brazo.

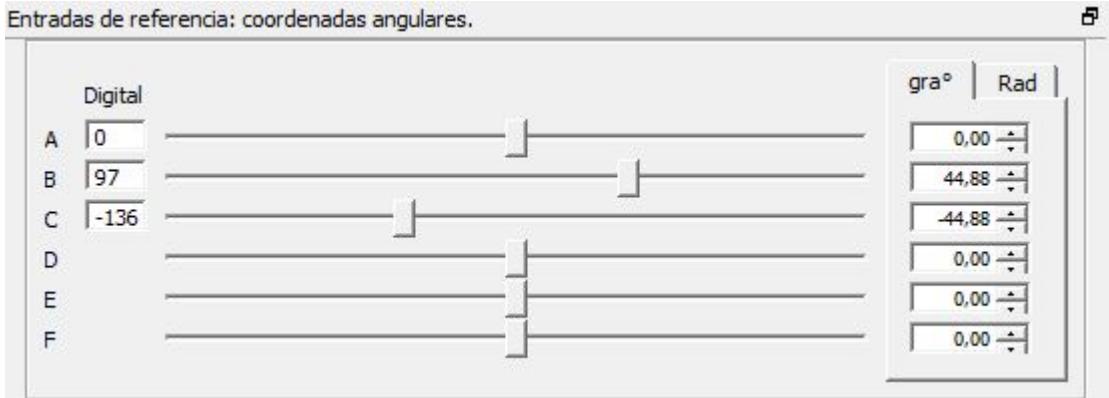


Figura 3.14. Función 'Setpointer' del software Robocom.

A la izquierda se observan las letras 'A,B,C,D,E,F' que corresponden a cada motor. 'A, B, C' son para los motores de mayor potencia que dan movimiento al armazón del brazo robot (hombro, codo y muñeca). El resto es para los motores de la mano (D, E, F). De igual modo, la parte central de este cuadro cuenta con barras deslizantes cuya información permite generar la trama de comunicación que es enviada al microcontrolador.

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 3.1:

$$Valor_{grados} = (Valor_{digital} + K_{offset})K_{pro} \quad (3.1)$$

Donde K_{pro} es una constante proporcional de calibración que es utilizada para convertir el ángulo de binario a grados, medida en grados/bits y su valor programado por defecto es 0.33 grados/bits para los tres motores principales (A, B, C). Para el caso de los motores de la mano (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 los motores A y C, y 39 para el motor B. Para los motores de la mano (D, E, F), el valor es fijo y vale 5682 bits.

En la pestaña 'Calibración', mostrada en la figura 3.15, puede apreciarse mejor la información sobre K_{offset} y K_{pro} descritas anteriormente.

Calibración		
	<u>K_{pro}</u> [grados/bit]	<u>K_{offset}</u> [bits]
q1	0,33000	0
q2	0,33000	39
q3	0,33000	0

Figura 3.15. Configuración de constantes de calibración del software Robocom

3.2.2. Generación de la trama de comunicación

Estructura general de la trama

La estructura general de la trama enviada desde el software Robocom al sistema de control consta de tres bytes cuya estructura corresponde a la tabla 3.2.2. Esta trama carece de bytes que determinen el inicio o el fin de la comunicación o de algún un método de comprobación de errores en la transmisión.

Tabla 3.1. Estructura general de la trama de comunicación.

Comando	Dato high	Dato low
8 bits	8 bits	8 bits

Sintaxis de comandos de movimiento

- La trama de comunicación será de 24 bits distribuidos: 8 bits para el comando y 16 bits para el dato a transmitir, como se indica en la tabla 3.2.2.
- La cabecera o 'comando' de la trama puede tomar cualquier valor referido en la tabla 3.2 en la columna 'Comando'. El valor será de 8 bits e indicará qué instrucción ejecutará el controlador.
- El dato será el valor equivalente en hexadecimal del ángulo que se desea mover y será guardado en una variable de 16 bits 'Vdigital'.
- Para los motores A, B, C este valor binario debe trasladarse el contenido de la variable 'Vdigital' 6 bits a la izquierda. Esto no se realiza para los motores D, E y F.
- Para todos los motores, se obtendrá el valor bajo (byte menos significativo) al igualar 'Vdigital' a una variable de 8 bits 'Vlow'.
- Para todos los motores, se obtendrá el valor alto (byte más significativo) al trasladar el contenido de la variable 'Vdigital' 8 bits a la derecha e igualarlos a una variable de 8 bits 'Vhigh'.
- La trama final será <Comando><Vhigh><Vlow>.

Tabla 3.2. 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	Habilitar articulación A
	01	Deshabilitar articulación A
	02	Habilitar articulación B
	03	Deshabilitar articulación B
	04	Habilitar pinza neumática
	05	Deshabilitar pinza neumática
	06	Habilitar articulación C
	07	Deshabilitar articulación C
	09	Solicitud de la posición del ADC de articulación A
	10	Solicitud de la posición del ADC de articulación B
	11	Solicitud de la posición del ADC de articulación C
F0	Dato	Escribir Período 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

Al ejecutar Robocom se envía una rutina por defecto para una configuración previa de la posición de cada motor. Esto se logra presionando la pestaña 'Posiciones por defecto' que se observa en la figura 3.12(sección 1).

Al presionar la función 'Posiciones por defecto' se producirá la trama de comunicación mostrada en la figura 3.16. Esto se observó con ayuda del software DockLight v2.2, que es un sniffer/monitor de puerto serial RS232 para Windows.

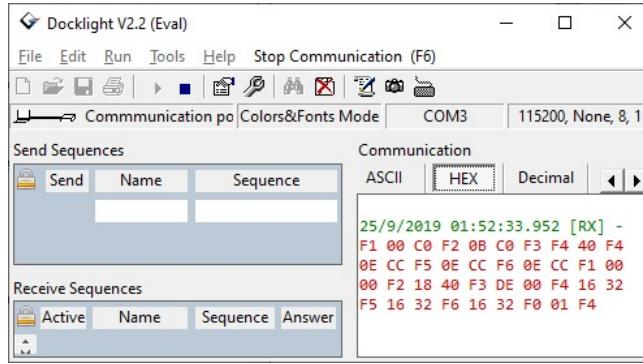


Figura 3.16. Vista de trama transmitida por el software Robocom al iniciar conexión.

Tabla 3.3. Comandos enviados al generar trama por defecto.
Información obtenida con el software DockLight.

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 3.3, se tiene una lista de comandos permitidos en la comunicación del controlador y Robocom. Esta lista se muestra en la tabla 3.2.

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.

Para los motores A, B, C, el cálculo del equivalente en grados se realizará según la ecuación 3.1, 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.

En cuanto a los motores D, E, F, el cálculo del equivalente en grados se determina mediante la ecuación 3.2.

$$V_{grados} = 0,035(V_{digital} - 5682) \quad (3.2)$$

3.3. 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 3.17.

3.4. PWM

En la figura 3.17 se observan los pines de salida seleccionados para la señal PWM que controlará a los motores. En el manual anexo al trabajo de Marcano [1] indica cuáles son las características básicas que deben poseer los registros de

configuración de PWM para el microcontrolador Microchip dsPIC30F3011:

Motores A, B, C:

Estos motores tienen configurada una señal PWM de 11 bits y frecuencia 29,328kHz. Con esta resolución de señal se puede obtener un ciclo de trabajo máximo configurando el registro de Duty Cycle (PDC) a un valor de 2048. Los pines de salida de PWM fueron configurados en modo complementario, por lo tanto para que la corriente promedio entregada al motor sea cero se configura el registro PDC=1023. Para que el motor produzca par máximo, el registro de duty cycle debe ser PDC=0 para que gire en un sentido o PDC =2048 para que gire en el otro sentido.

Motores D, E, F:

Se configuraron 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 salida secundario debe ser igual a 5682 en decimal. Es necesario generar un pulso desde 1 ms hasta 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.

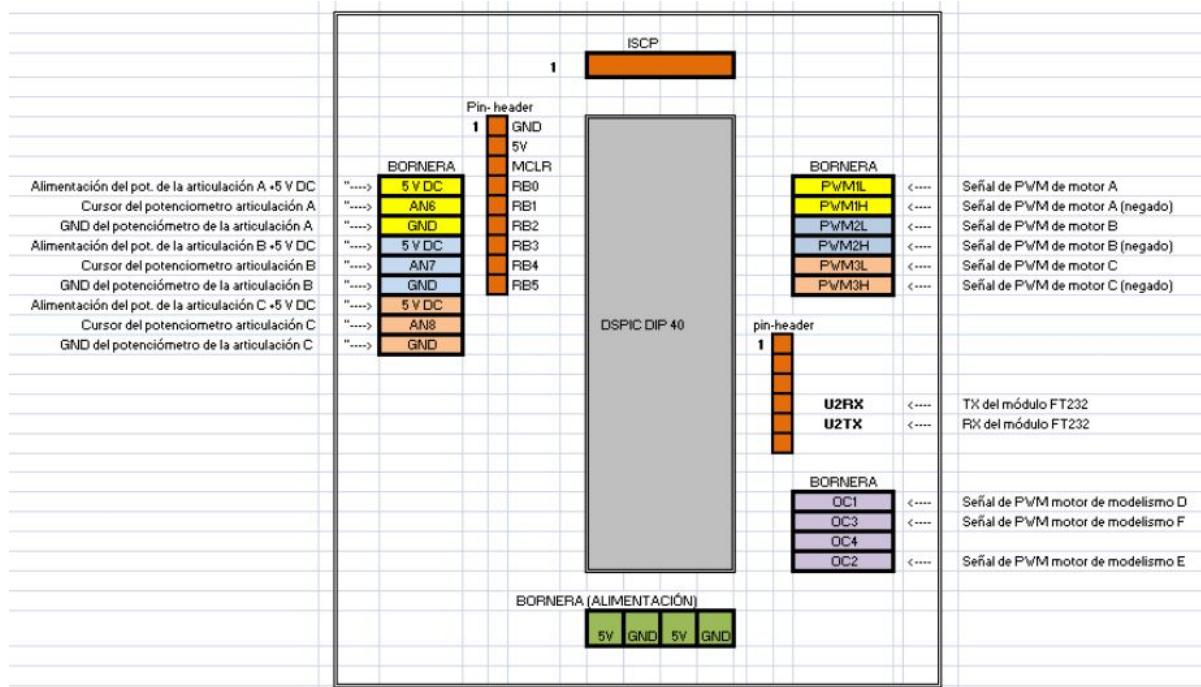


Figura 3.17. Pines de conexión del módulo controlador original.

CAPÍTULO IV

DISEÑO DEL SISTEMA DE ACCESO REMOTO

En este capítulo se mostrarán los requerimientos necesarios para el diseño propuesto en este trabajo. Además, se hace una comparación entre diversos dispositivos que cumplen con estos requerimientos. Luego de esta comparación, se hace una selección de acuerdo a las características que mejor se ajusten a los objetivos planteados.

4.1. Selección de dispositivos

4.1.1. Requerimientos

El sistema original realiza la comunicación de forma alámbrica a través de un convertidor USB-serial y un software instalado en un computador, para de esta forma generar las instrucciones de movimiento en el brazo manipulador. Tal como se indica en la figura 4.1.

El brazo robot se encuentra ubicado en un área limitada o con dificultades para la recepción de comunicación de telefonía móvil y conectividad a internet.

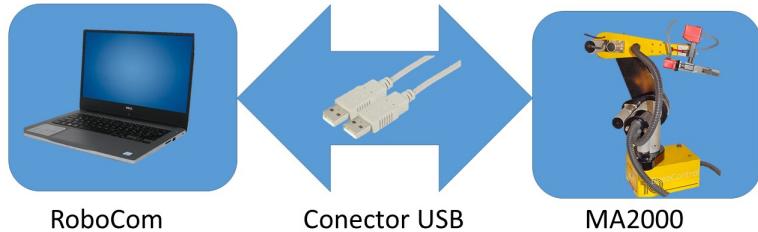


Figura 4.1. Esquema de comunicación original

Por otra parte, la mejora que propone este trabajo de grado trata sobre una optimización del método de comunicación que se realiza con el controlador del brazo robot a través de un dispositivo remoto. Es por ello que se requiere que el sistema que se propone sea el correspondiente a la figura 4.2.

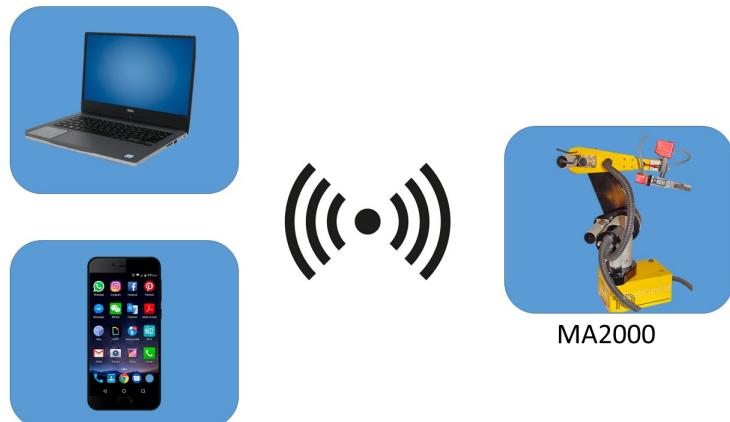


Figura 4.2. Esquema de comunicación inalámbrica.

Las especificaciones de diseño del sistema de acceso remoto requieren del establecimiento de un servicio que permita comunicarse inalámbricamente con el brazo robot sin la necesidad de modificar invasivamente el hardware de este. De utilizarse tecnología de RF es necesario implementar el hardware emisor y

receptor para procesar las señales, por lo que modificaría el sistema más de lo esperado. Lo mismo ocurre con las tecnologías de telefonía como GPRS y UMTS que dependen de la señal transmitida por las antenas repetidoras y de la cobertura local al momento de comunicarse para transmitir señales de largo alcance, lo cuál lo hace una tecnología económica y óptima para diversos sistemas, pero poco práctica para aplicaciones en áreas geográficas disminuidas como es el caso del brazo robot ubicado en el LCID de la Universidad Central de Venezuela.

La tecnología bluetooth puede considerarse como una buena opción ya que los dispositivos que cuentan con ella son en su mayoría sencillos de implementar. No obstante, su conectividad está limitada a un área disminuida en donde la señal puede perderse a través de diversas superficies generando problemas de comunicación. Esto podría solucionarse estableciendo un servidor de los datos transmitidos desde un computador y algún otro dispositivo. No obstante, la tecnología bluetooth no soporta la posibilidad de creación de servidores en red ya que su conectividad es de punto a punto, omitiendo la opción de otros equipos de unirse a esta red e imposibilitando el esquema propuesto en la figura 4.2.

La tecnología Wi-Fi implica implícitamente un mayor consumo de potencia en comparación con las tecnologías mencionadas anteriormente. Sin embargo, para algunas aplicaciones este factor no es delimitante para determinar su utilidad. En contraste, esta tecnología permite conectar una mayor cantidad de dispositivos a través de una red y a una distancia superior a la lograda por Bluetooth, lo cuál permite que computadores, teléfonos celulares y dispositivos de IoT, puedan establecer comunicación a través de un servidor web de forma local (sin conexión a internet) o en línea, sin necesidad de una conexión física. Para compensar la ausencia del computador, este dispositivo debe conectarse al controlador del brazo manipulador vía puerto serial para así establecer la comunicación entre el servidor y el cliente hacia el robot. Logrando una instalación sin afectar el hardware original

del sistema.

Considerando una distribución relacionada al diagrama de bloques de la figura 4.3, puede observarse que al añadir una etapa de comunicación inalámbrica implementada mediante un servidor en red local a través de aplicaciones, logra una adaptación no invasiva al sistema del brazo robot. Permitiendo que se manipule el firmware del microcontrolador Wi-Fi sin mayores alteraciones.

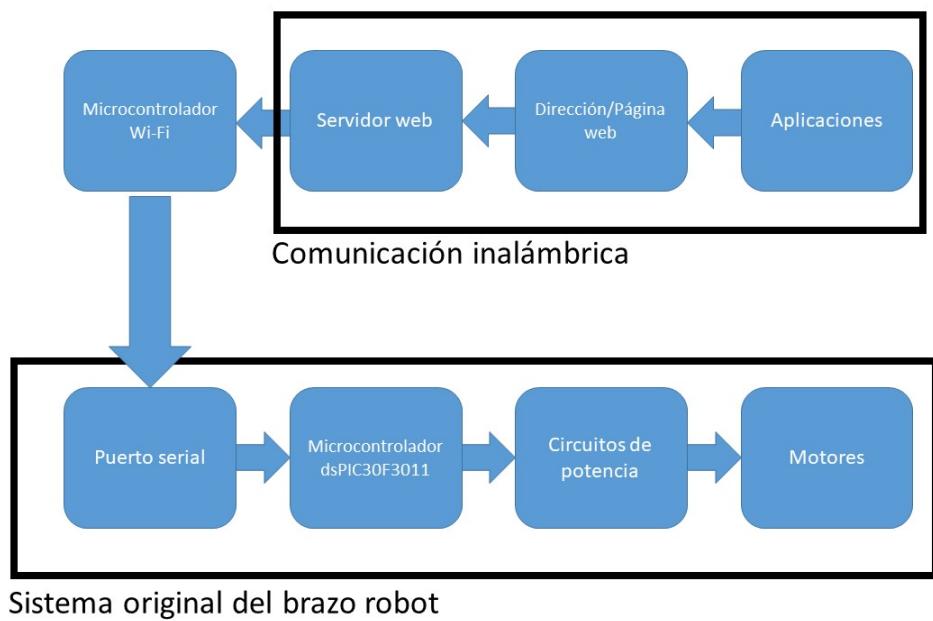


Figura 4.3. Diagrama de bloques del sistema por conexión Wi-Fi

Motivado a esta estructura, se considera que las necesidades del proyecto requieren de un dispositivo que cuente con periféricos UART, medios de conexión por Wi-Fi, memoria con capacidad de las necesidades del código (soportar el código fuente de una página web, gráficos y código fuente del firmware), y puertos de propósito general suficientes para implementar las señales indicadoras del prototipo. Dichas de otro modo, se resumen los requerimientos en la tabla 4.1.

Tabla 4.1. Estimación de requerimientos para el diseño.

Comunicación por radiofrecuencia	Sí
Memoria flash	Por el orden de los MB
GPIO	Al menos 10
Costo	5 a 20 \$
UART	Sí
Capacidad de programación en lenguaje C	Sí
Capacidad de programación en otros lenguajes	Sí

4.1.2. Raspberry Pi 3 (modelo B+)



Figura 4.4. Tarjeta de desarrollo Raspberry Pi 3 Model B+

Fuente: <https://www.raspberrypi.org/products/>

Esta tarjeta de desarrollo cuenta con microprocesador Broadcom BCM2837B0, propiamente un Cortex-A53 (ARMv8) de 64-bit. La velocidad de procesamiento del SoC es de 1.4GHz.

Posee cuarenta pines de GPIO, dos UART, además de cuatro puertos USB, un puerto HDMI, un conector auxiliar de 3.5mm y posibilidad de conectar una memoria SD y display táctil. Cuenta con puerto RJ45 para conexión Ethernet,

entre otros elementos.

Su gran capacidad de procesamiento lo hace ser una de las primeras opciones a considerar en la realización de servidores web debido a su bajo costo (alrededor de 35\$ según distribuidor oficial. Consultado el 24 de septiembre de 2019.) con relación a todos los elementos que ofrece, ya que permite la ejecución de sistemas operativos basados en Linux, pudiendo programarse en lenguajes como Python, C# o C.

Cuenta, además, con Bluetooth 4.2/BLE y compatibilidad de módulos de conexión Wi-Fi que no están incluidos en el producto principal.

Las opciones de acceso remoto que ofrece este dispositivo son:

Acceder a internet a través de un servidor web con conectividad a internet estableciendo un sitio web, o a través de su IP personal. También permite la copia y transferencia de archivos desde/a otro computador usando VNC, SSH, SFTP, SCP, SSHFS, rsync, FTP y SAMBA/CIFS.

4.1.3. Cypress CYW54907

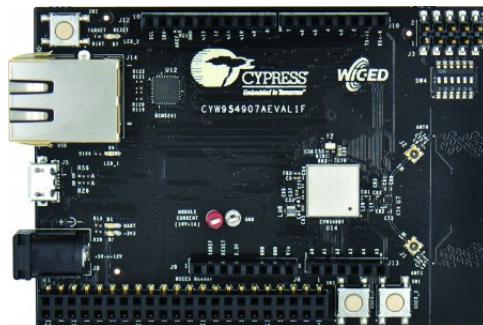


Figura 4.5. Tarjeta de desarrollo CYW54907AEVAL1F de Cypress.

Fuente: <https://www.cypress.com/documentation/development-kitsboards/cyw54907aeval1f-evaluation-kit>

Este equipo es una tarjeta de desarrollo basada en el microcontrolador CYW54907 de Cypress. Contiene un microprocesador ARM Cortex R4, con velocidad de procesamiento máxima de 1.4GHz.

El dispositivo cuenta con 30 pines de GPIO, tres UART y posibilidad de conectividad por radiofrecuencia con Wi-Fi o Bluetooth. Cuenta con un conector RJ45 para conexión Ethernet, además de ejecutar sistemas operativos y ser compatible con diversos módulos.

Al poderse programar en diversos lenguajes, este dispositivo es una opción importante a considerar, ya que ofrece algunas ventajas convenientes en comparación a la Raspberry, como lo es poseer Wi-Fi incluido en el SoC. Su costo oficial es de 99\$. (Consultado el 24 de septiembre de 2019.)

4.1.4. ESP 8266

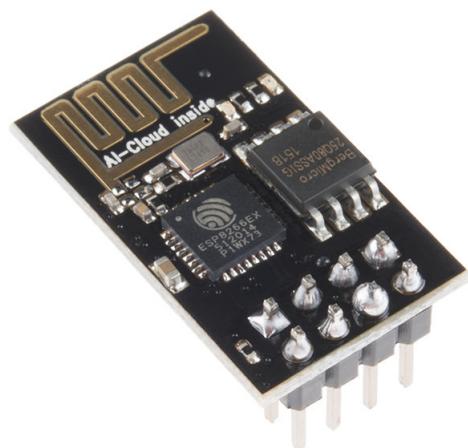


Figura 4.6. ESP8266 con chip ESP-01
Fuente: <https://www.espressif.com/>

El ESP8266 es uno de los dispositivos más conocidos en el mercado debido a su alta versatilidad para el diseño de sistemas para IoT y su bajo costo.

Este microcontrolador fabricado por Espressif Systems, con núcleo Xtensa Tensilica L106 de 32 bit con velocidad de 80MHz. Posee 16 GPIO, en los cuales pueden configurarse las entradas, salidas, UART, o algún otro puerto que requiera. Su convertidor analógico/digital es de 10bit, y a su vez posee un único pin de entrada analógica.

Cuenta con una memoria flash de 4MB en algunos modelos, y 32kB de memoria RAM. Posee compatibilidad con SPI, I2C, I2S

Este dispositivo posee incluye en el SoC, la posibilidad de conectividad Wi-Fi bajo seguridad WPA y WPA2. Pudiendo establecer servidores TCP/IP, UDP; así como clientes de los mismos. No posee puerto Ethernet, pero al tener Wi-Fi puede conectarse fácilmente a internet en la banda de 2.4GHz según el estándar IEEE 802.11 b/g/n (Hasta las versiones ESP 12-E).

Puede programarse en lenguaje C, Simba o Lua. Además, cuenta con un SDK para implementar RTOS o programar sin sistema operativo.

El costo varía entre los 3\$ hasta alrededor de los 5\$ (consultado el 24 de septiembre de 2019). Es por ello que se ha posicionado como una de las mejores opciones en sistemas embebidos en el mercado del IoT.

4.1.5. ESP 32



Figura 4.7. ESP 32 de Espressif.
Fuente: <https://www.espressif.com/>

El ESP 32 es también fabricado por la empresa Espressif Systems y surgió como una mejora del ESP-8266.

Una de las nuevas implementaciones que este dispositivo trajo fue la posibilidad de establecer todos los pines de GPIO como entradas analógicas, con un convertidor analógico/digital de 12bits de resolución y un convertidor digital/analógico de 8bits. Es importante mencionar que esta versión cuenta con 36 GPIO y el doble de memoria flash, 16MB.

Cuenta con Wi-Fi y Bluetooth 4.2 BR/EDR/BLE, ademas de compatibilidad de comunicación SPI, I2C, I2S.

Entre otros elementos resaltantes, posee 3 UART, compatibilidad Ethernet MAC, bus CAN, controlador remoto infrarrojo, sensor de efecto Hall, 10 sensores capacitivos táctiles, y 16 canales para señales PWM.

Tiene un costo entre 5\$ y 10\$ (consultado el 24 de septiembre de 2019) y desde su aparición ha ganado popularidad entre los seguidores del ESP-8266 debido a la gran cantidad de periféricos que posee por un costo económico en comparación a competidores de mayor gama.

4.2. Selección

Basado en la información mostrada en la tabla 2.4, se estima que las características más importantes a considerar son: ancho de banda, rango y flexibilidad. Esto debido a que este sistema debe tener la posibilidad de comunicarse desde la página web con el servidor sin desconexión a pesar de recibir instrucciones desde una aplicación externa, ya que es importante que puedan realizarse adaptaciones al mismo sin modificar el código contenido en el dispositivo.

En cuanto al ancho de banda, esto permite que la comunicación sea de baja latencia y pueda transmitir datos en tiempo real con un retardo despreciable. Además, el rango es un factor determinante ya que el dispositivo puede estar considerablemente alejado y aún así poder comunicarse sin realizar reconexiones.

Motivado a la necesidad de no conectividad a internet, se considera que la comunicación vía Wi-Fi cumple las expectativas de los requerimientos si se establece una conexión a través de un servidor local. No obstante, cualquiera de los dispositivos anteriormente mencionados (con excepción de la Raspberry Pi 3) puede ajustarse a esta necesidad del diseño.

Para discriminar y seleccionar el dispositivo que mejor se ajuste a los requerimientos mencionados, puede recurrirse al gráfico de la figura 4.8.

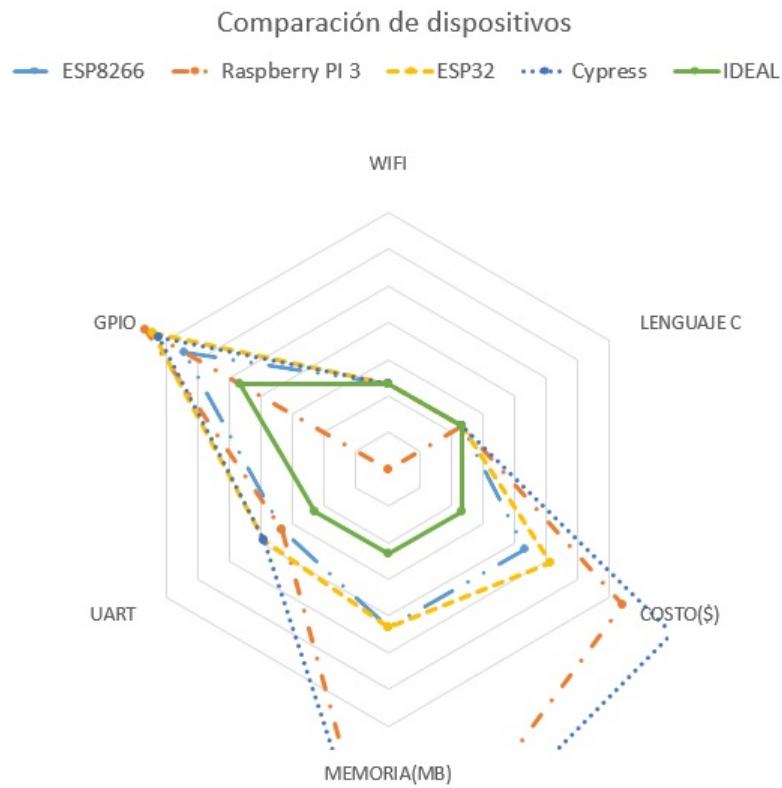


Figura 4.8. Comparación entre los dispositivos considerados

En esta figura puede observarse la gráfica verde como guía, ya que es la que representa a las necesidades mínimas del proyecto, según como se indica en la tabla 4.1. Alrededor se muestran las representaciones de las características de las tarjetas de desarrollo estudiadas en comparación con los parámetros ideales. En este gráfico se puede observar que a pesar de las considerables ventajas que ofrecen todos los dispositivos en cuestión, la gráfica que más se acerca a las condiciones ideales es la azul. Es decir, el ESP-8266.

Esto se evidencia al comprobar que es el dispositivo de menor costo entre los estudiados. Además, cubre las condiciones mínimas del diseño a pesar de no tener una extrema holgura de recursos como se demuestra en los dispositivos de mayor gama.

Por las características observadas en la figura 4.8, además de las dificultades de conectividad a internet, la posibilidad de conexión a diversos dispositivos, y las ventajas de la utilización de la tecnología Wi-Fi antes que Bluetooth como se muestra en la tabla 2.4, se seleccionará el dispositivo ESP-8266 con conectividad Wi-Fi sin acceso a internet para establecer un servidor de comunicación con una página web incluida en el SoC el cuál podrá accederse de forma remota para lograr el movimiento del brazo robot sin presencia de computadores intermediarios.

CAPÍTULO V

SISTEMA DE ACCESO REMOTO

En este capítulo se describirán el hardware y software empleados para la programación del firmware de la tarjeta de desarrollo seleccionada para el prototipo. Además, se mostrarán los resultados alcanzados durante este proceso.

5.1. Detalles de software

El sistema de acceso remoto se desarrolló 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. Esta información está resumida en la tabla 5.1.

Tabla 5.1. 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

5.1.1. Configuración del IDE Atom PlatformIO

Al ejecutar el programa, se observará una pantalla como la indicada en la figura 5.1.

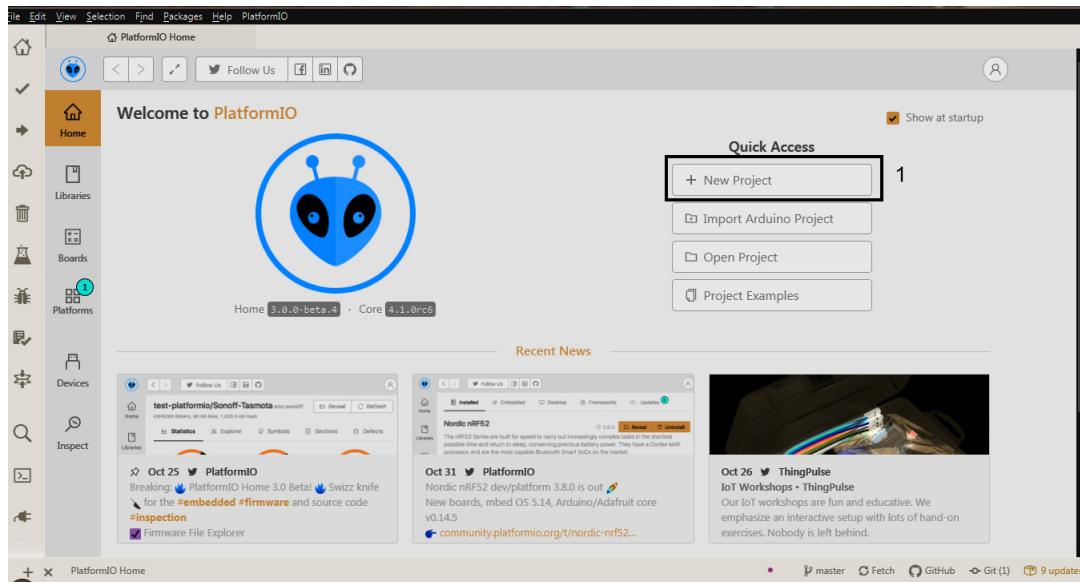


Figura 5.1. Ventana principal del IDE Atom PlatformIO

1. Opción de creación de nuevo proyecto.

En la ventana principal se selecciona la opción de nuevo proyecto, identificada en la sección 1 de la figura 5.1. De esta manera el IDE pondrá a disposición del usuario todas las librerías contenidas en el SDK. Para lograr esto, se configuran los parámetros según la figura 5.2.

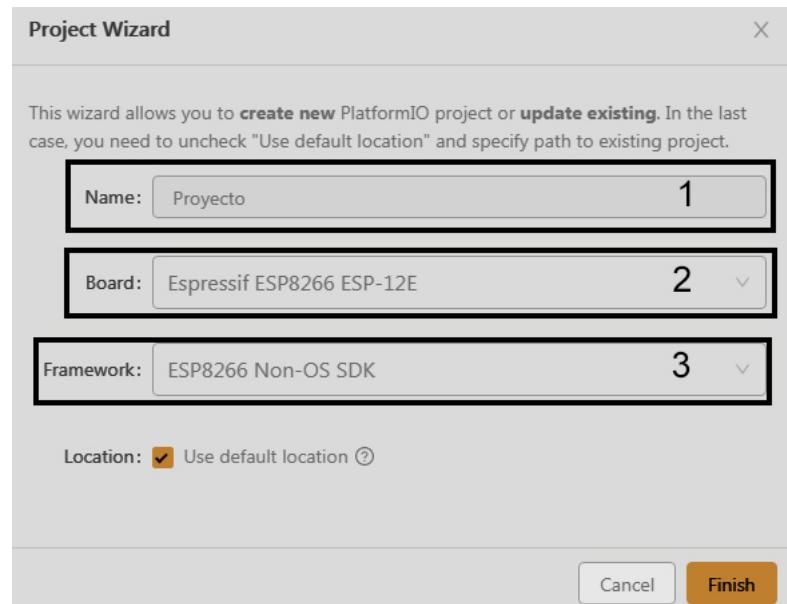
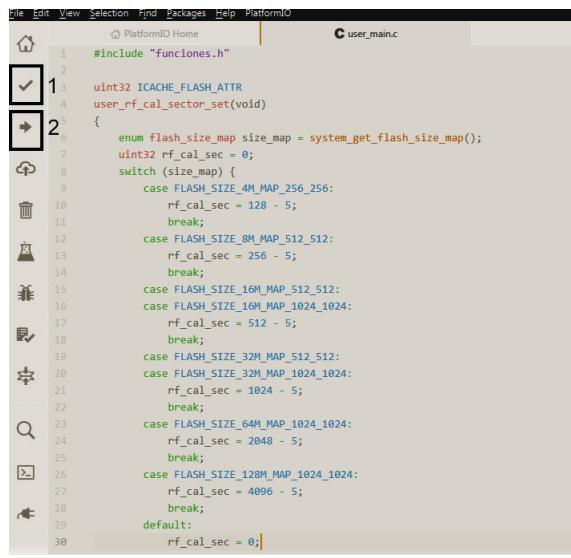


Figura 5.2. Creación de nuevo proyecto

1. Nombre del proyecto.
2. Selección de microcontrolador. (Para este proyecto se usará Espressif ESP8266 12-E).
3. Selección de framework de desarrollo. (En este caso Non-OS SDK).

Hecho esto, se podrá programar el microcontrolador en lenguaje C a través del IDE Atom PlatformIO. Compilando y cargando el programa como se muestra en la figura 5.3



```
#include "funciones.h"
uint32 ICACHE_FLASH_ATTR
user_rf_cal_sector_set(void)
{
    enum flash_size_map size_map = system_get_flash_size_map();
    uint32 rf_cal_sec = 0;
    switch (size_map) {
        case FLASH_SIZE_4M_MAP_256_256:
            rf_cal_sec = 128 - 5;
            break;
        case FLASH_SIZE_8M_MAP_512_512:
            rf_cal_sec = 256 - 5;
            break;
        case FLASH_SIZE_16M_MAP_512_512:
        case FLASH_SIZE_16M_MAP_1024_1024:
            rf_cal_sec = 512 - 5;
            break;
        case FLASH_SIZE_32M_MAP_512_512:
        case FLASH_SIZE_32M_MAP_1024_1024:
            rf_cal_sec = 1024 - 5;
            break;
        case FLASH_SIZE_64M_MAP_1024_1024:
            rf_cal_sec = 2048 - 5;
            break;
        case FLASH_SIZE_128M_MAP_1024_1024:
            rf_cal_sec = 4096 - 5;
            break;
        default:
            rf_cal_sec = 0;
    }
}
```

Figura 5.3. Compilación y carga del programa en la tarjeta.

1. Compilar programa.
2. Cargar el programa a la tarjeta.

5.1.2. Diseño de página web

En cuanto al diseño de la página web; se desarrolló este código a través del editor de texto Brackets versión 1.14, ya que este trae soporte en tiempo real para modificaciones del archivo HTML y los diversos scripts. Las versiones de software utilizadas para esta sección se encuentras indicadas en la tabla 5.2.

Posteriormente, el código fuente generado para la página web, observado en la figura 5.4, fue agregado al programa del sistema embebido como una cadena de caracteres, generando una página inicial para el servidor que se establecerá.

Tabla 5.2. Especificaciones del software empleado en la página web.

PÁGINA WEB	
Editor	Brackets versión 1.14
HTML	versión 5
CSS	versión 3
Javascript	versión ECMAScript 2016 (1)
jQuery	versión 3.4.1 (1)

(1). Estas versiones de Javascript y jQuery dependerán del navegador utilizado, ya que automáticamente se tomará la más actualizada que posea el sistema.

index.html (CodigoPaginaTesis) - Brackets

Archivo Edición Buscar Ver Navegación Desarrollo Ayuda Emmet

CodigoPaginaTesis .git .gitattributes banner.css eienegro.jpg eienegro.png estilos.css index.html javascript.js README.md pagina.zip

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <title>TEG</title>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1">
7   <link rel="stylesheet" href="estilos.css">
8   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
9
10
11   <link rel="icon" href="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABAAQAAHSCAAAoDoTzSAZAAAGXCRFWHRTzJZed2FyZQ8BZUGz5BzJmFnZV1YWR5c1LPAAAQVNFUeNrs3d1fDneB+AaDvFLBLdCMEQptu7wViaptWPa5jApa2BnZHUoEcKgN4J1InjEqEoGvBHFSSzNxznOpj2BWM12lo4kv+RtAVOrYz83B3s5/_v/3NUb/nWc3pFCyzoqBUDnLbd3LcFj2553/_5zb/_fSz3bePdbs/f3Pztpv8f3Pzpp8uu7/65pf/eFqrASAH1zQ0BZGw+St+7gFC/7hgTyv1lc0Af5+XbWD9Wmm0lo+3c7B7-5dLOPx0TBHF95Hw_MgAIQDJAsgkf2GcnkW4+e1shRuntMk2/_vnnqj0nBvDxw+DxeA9RFkv/TR
```

Figura 5.4. Vista del software Brackets para el diseño de la página web.

5.2. Detalles de hardware



Figura 5.5. Placa del microcontrolador ESP-8266.

Las características de hardware del sistema embebido utilizado para el diseño del prototipo pueden verse en la tabla 5.3. Es importante mencionar que estos datos fueron obtenidos desde la información grabada en la placa (figura 5.5) y desde la información técnica proporcionada por Espressif en [16].

Tabla 5.3. Especificaciones de hardware de la tarjeta de desarrollo.

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

Para establecer la comunicación entre el dispositivo ESP8266 y el microcontrolador del brazo robot, es necesario enlazar el puerto serial de ambos integrados.

A continuación se muestra el diagrama esquemático de conexión de la tarjeta de desarrollo ESP-8266 con el dsPIC30F3011 del controlador del brazo robot en la figura 5.6.

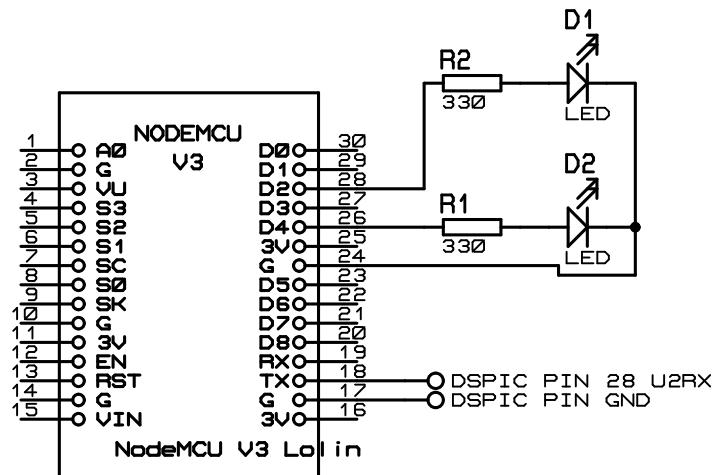


Figura 5.6. Esquemático de conexión entre ESP-8266 y dsPIC30F3011

Una vez establecidos y configurados todos los parámetros del prototipo, se instaló el hardware sobre una tarjeta de prototipos tipo stripboard a modo de producto final de este trabajo. Este circuito puede observarse en la figura 5.7.

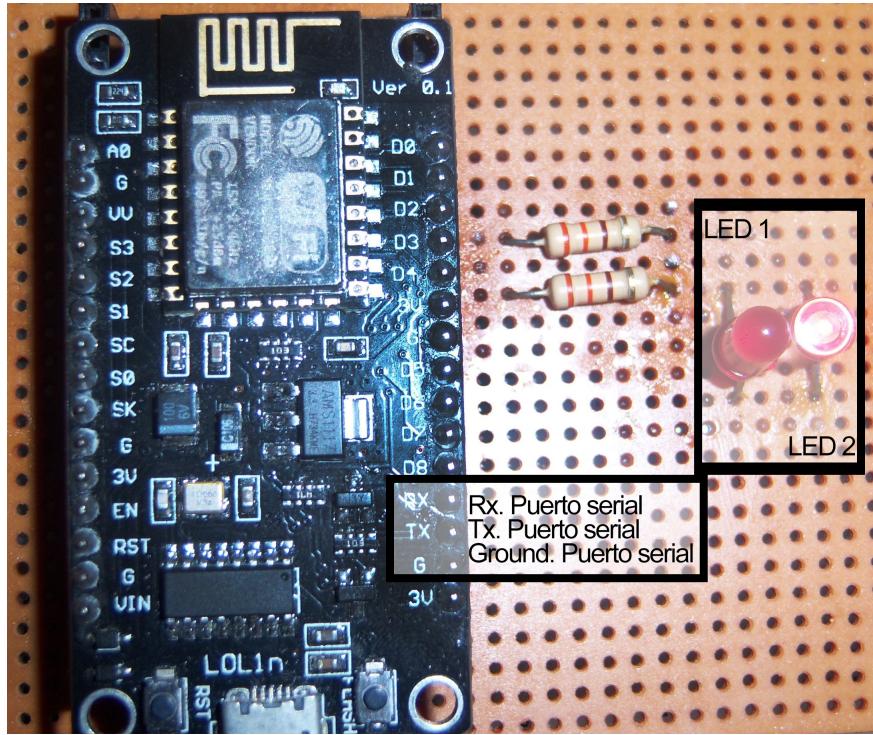


Figura 5.7. Montaje del prototipo.
LED 1. LED indicador de comunicación.
LED 2. LED indicador de encendido.

5.3. Detalles de firmware

5.3.1. Programación en lenguaje Lua

Lua se trata de un lenguaje interpretado con código base escrito en C, con semejanzas al lenguaje Python pero menos eficiente. Es de fácil aprendizaje y posee una forma de programación intuitiva para el usuario no experto. Es decir, es un lenguaje diseñado para aplicaciones poco complejas ya que se encuentra en desarrollo. Su puesta en marcha requiere una modificación del bootloader del microcontrolador, debiendo cargar al dispositivo un archivo binario previo con los módulos que desean ser utilizados. Esto a su vez ofrece pocas opciones para el manejo de memoria y de los periféricos a un modo mejor optimizado. [17]. Por

estos motivos expuestos, no será utilizado para la programación de este prototipo.

5.3.2. Programación en lenguaje Simba

Simba es un sistema operativo de tiempo real (RTOS) escrito en lenguaje C que surgió como una opción para programar el microcontrolador ESP8266. Es soportado por el IDE Atom PlatformIO y ofrece ventajas en cuanto a su fácil aprendizaje y a un mayor potencial de las aplicaciones que pueden generarse. Sin embargo, este entorno aún se encuentra en desarrollo y por lo tanto presenta muchos fallos en torno a las librerías y drivers no implementados. Además, las librerías inherentes al firmware del dispositivo no están dispuestas para modificación del programador y su compilación es lenta en comparación al la realizada en lenguaje C. [18]. Por este motivo no será utilizado en este prototipo.

5.3.3. Programación en lenguaje C

Espressif RTOS ESP-8266 SDK

SDK, en inglés Software Development Kit (Kit de Desarrollo de Software). Es un compilado de herramientas necesarias para desarrollar aplicaciones específicas para el entorno deseado.

El ESP8266 ofrece la capacidad de ser programado basado en un sistema operativo de tiempo real (RTOS). Este SDK proporcionado por Espressif, se trata de una adaptación del entorno de código libre freeRTOS (página oficial: <https://www.freertos.org/>).

Un sistema operativo de tiempo real es una herramienta eficiente para dispositivos con poca capacidad de procesamiento. Es decir, aquellos dispositivos que no cuentan con una variedad de núcleos o kernels. RTOS trata de implementar

en estos microsistemas una similitud de comportamiento de múltiples unidades de procesamiento aún estando contenidas en un solo núcleo. Lo cual es una ventaja importante para la implementación de sistemas de mayor complejidad de código.

La lógica funcional del método de programación RTOS es mediante establecimiento de tareas e interrupciones controladas por orden de prioridad y por el timer principal del sistema. Esto genera la apariencia de estar procesándose múltiples tareas en un mismo instante. No obstante, estas tareas o 'tasks' están limitadas por el factor de tiempo que dicta el procesador. [19].

Este sistema operativo hace uso de todas las virtudes del procesador a una mayor escala que las otras opciones de programación (Simba, Lua, C), debido a que debe aprovechar todos los recursos para permitir la libre ejecución de la mayor cantidad de tareas; esto ocasiona que su compilación sea relativamente más lenta que el caso de Non-OS.

Debido a que en este prototipo no es de interés la implementación de tareas para el manejo de la comunicación del brazo robot, ya que puede hacerse con un mismo núcleo; no se considerará para la programación de este proyecto.

Espressif Non-OS ESP-8266 SDK

Se trata de un kit diseñado por la empresa Espressif para desarrollar software en dispositivos sin sistema operativo. En ella tiene las librerías de mayor importancia para la configuración y uso del ESP-8266. [16] [20] [21] [22]

5.4. Uso de librerías en lenguaje C

Para el desarrollo de aplicaciones con manipulación de caracteres y cadenas es necesario utilizar la librería *string.h*. Esta librería está incluída en el estándar

de lenguaje C, ya que es de uso común.

Incluye funciones para manipulación de arreglos de caracteres. Propiamente, define un tipo de variable, un macro y varias funciones para manipulación de arreglos y cadenas. Algunas de las funciones utilizadas comúnmente son: *strtok*, *strcmp*, *strncpy*, *memset*, *strlen*, *strcpy*, *strncpy*, entre otros.

Otra librería importante que sirve soporte a *string.h* es *stdlib.h*, debido a que posee las funciones necesarias para la conversión de caracteres en números, utilización de memoria dinámica para la generación de listas, colas y demás algoritmos de organización y posicionamiento de memorias. También está incluída en el lenguaje C estándar y es considerada de uso común. A grandes rasgos, esta librería define cuatro tipos de variables, múltiples macros, y varias funciones de uso general; funciones como: *atoi*, *atof*, *atol*, *malloc*, *free*, *abort*, *abs*, *rand*, entre otras; son las más utilizadas.

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

El SDK NON-OS provee un conjunto de interfaces de programación de aplicaciones (API) para las funcionalidades del núcleo ESP-8266, como recepción/transmisión de datos a través de Wi-Fi, funciones para el manejo de TCP/IP, funciones para la interfaz de hardware y el manejo básico del sistema.

Las librerías más importantes incluídas dentro del SDK son: osapi.h, user_interface.h, driver/uart.h, espconn.h, gpio.h, eagle_soc.h.

5.5.1. osapi.h

Esta librería realiza una re-definición de algunas de las funciones más importantes de la librería string.h y algunas funciones para manejo del timer. Por lo

tanto, es una librería importante para el manejo de cadena de caracteres.

Algunas de las funciones incluidas en esta librería incluyen el prefijo 'os', que simboliza que estas dan una respuesta de debug vía puerto serial al momento de utilizarlas.

Para deshabilitar estos comandos, es necesario escribir previo a todas las declaraciones del programa principal: 'system_set_os_print(0)'. Y con esto pueden utilizarse sin inconvenientes las funciones con comandos 'os'.

Visto en lenguaje C:

```
void ICACHE_FLASH_ATTR
user_rf_pre_init(void)
{
    system_set_os_print(0);
}
```

Nótese que se hace el llamado de esta función dentro de la definición de 'user_rf_pre_init' debido a que esta se ejecuta primero que la función main, haciendo que se evite cualquier error inesperado con los comandos 'os'.

5.5.2. user_interface.h

Contiene las definiciones de las estructuras para configurar el Wi-Fi en modo estación o punto de acceso.

Esto es, la definición de los macros: 'NULL_MODE', 'STATION_MODE', 'SOFTAP_MODE' y 'STATIONAP_MODE'; que permiten configurar los modos de funcionamiento del Wi-Fi.

También define la estructura asociada a la función 'wifi_softap_get_config', quien es la función encargada de configurar todos los parámetros para establecer la conexión Wi-Fi.

5.5.3. driver/uart.h

Posee los macros y variables necesarias para la configuración del UART. Entre ellos, define la función 'uart0_tx_buffer', la cuál es de vital importancia para la transmisión de cadenas de caracteres vía puerto serial en este prototipo.

5.5.4. espconn.h

Define las estructuras para la creación del socket TCP o UDP, así como los prototipos y declaraciones de las funciones de la API correspondiente. Con estas funciones es posible establecer un servidor o cliente mediante solicitudes y respuestas.

Las funciones de interés en esta librería son: 'espconn_connect', quien permite conectar al canal de comunicación una vez este sea creado con 'espconn_create' y aceptado con la función 'espconn_accept'.

Las funciones 'espconn_disconnect' y 'espconn_delete' son de utilidad para desconectar el canal previo a eliminarlo, y así de esta manera evitar la saturación de la comunicación servidor-cliente, ya que esta solo permite la creación de un máximo de 12 canales, pero manteniendo únicamente 4 en paralelo en constante comunicación. Esto genera un problema que se resuelve aplicando estas funciones de desconexión y borrado.

5.5.5. gpio.h

Contiene los prototipos y definiciones de las funciones que configuran los puertos de propósito general (GPIO). En concreto, las funciones que seleccionan un pin como entrada o salida, en modalidad baja o alta; entre otros parámetros.

5.5.6. eagle_soc.h

Esta librería es, quizá, de las más importantes. En ella se definen los macros que permiten acceder a las múltiples funciones de los GPIO.

Como puede observarse en la tabla 5.4, el ESP-8266 posee múltiples puertos de propósito general, por ello es necesario indicar el propósito de uso de cada GPIO deseado.

Tabla 5.4. Registros de GPIO del ESP-8266. Fuente: Espressif.

FUNCTION 3	FUNCTION 4	AT RESET	AFTER RESET	SLEEP
GPIO0	CLK_OUT	oe=0, wpu	wpu	oe=0
GPIO1	CLK_RTC	oe=0,wpu	wpu	oe=0
GPIO2	U0TXD	oe=0,wpu	wpu	oe=0
GPIO3	CLK_XTAL	oe=0,wpu	wpu	oe=0
GPIO4		oe=0		oe=0
GPIO5		oe=0		oe=0
GPIO6	U1CTS	oe=0		oe=0
GPIO7	U1DXD	oe=0		oe=0
GPIO8	U1RXD	oe=0		oe=0
GPIO9	HSPIHD	oe=0		oe=0
GPIO10	HSPIWP	oe=0		oe=0
GPIO11	U1RTS	oe=0		oe=0
GPIO12	U0DTR	oe=0, wpu	wpu	oe=0
GPIO13	U0CTS	oe=0, wpu	wpu	oe=0
GPIO14	U0DSR	oe=0, wpu	wpu	oe=0
GPIO15	U0RTS	oe=0, wpu	wpu	oe=0
DEEPSLEEP	BT_XTAL_EN	oe=1, wpd	oe=1, wpd	oe=1

Los registros descritos en la tabla 5.4 pueden observarse mejor en la figura 5.8 , donde se muestra el pinout de este sistema embebido. La descripción de estos parámetros está descrita con detalle en la hoja de datos [22].

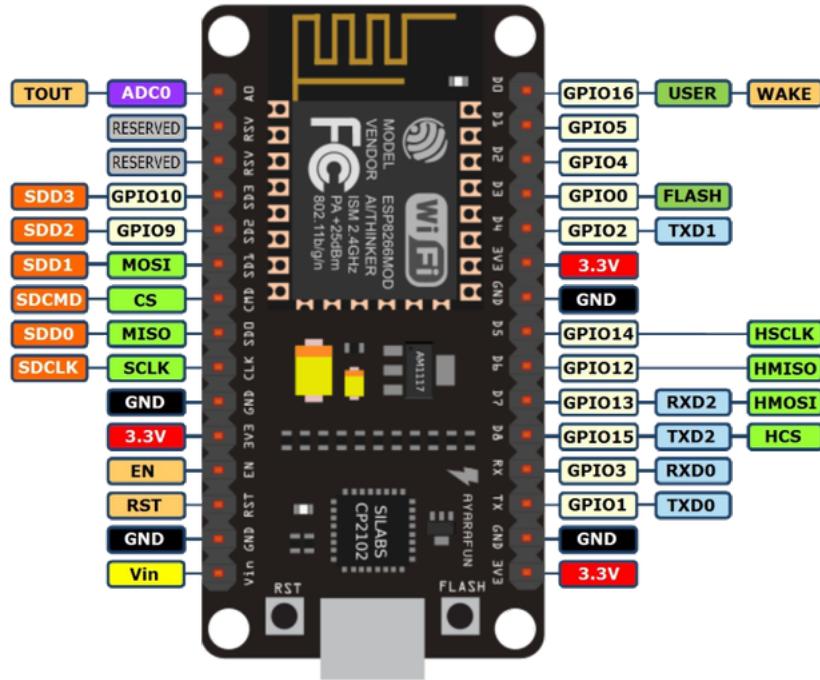


Figura 5.8. Pinout de sistema embebido con microcontrolador ESP8266.

La importancia de la correcta definición de los puertos de propósito general en este dispositivo dependerá de la aplicación que desea hacerse. En el caso del prototipo que se propone, es necesario definir el periférico de UART para operar con la comunicación serial hacia el brazo manipulador. Además, es necesario definir algunos pines como salida para visualización del comportamiento del programa y sin que estos interfieran con otros periféricos.

5.6. Contenido del programa

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

5.6.1. 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.

5.6.2. 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.

5.6.3. 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.

5.6.4. 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.

5.6.5. Compilado

Al compilar el código y cargarlo en el microcontrolador, el IDE muestra los valores totales del consumo de memoria de esta aplicación, como se observa en la figura 5.9.

```
platformio run
Processing esp12e (platform: espressif8266; board: esp12e; framework: esp8266-nonos-sdk)

Verbose mode can be enabled via `--verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/espressif8266/esp12e.html
PLATFORM: Espressif 8266 2.0.4 > Espressif ESP8266 ESP-12E
HARDWARE: ESP8266 80MHz, 80KB RAM, 4MB Flash
PACKAGES: toolchain-xtensa 1.40802.0 (4.8.2), tool-esptool 1.413.0 (4.13), framework-esp8266-nonos-sdk 2.1.0
LDF: Library Dependency Finder -> http://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 2 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <func>
Retrieving maximum program size .pio\build\esp12e\firmware.elf
Checking size .pio\build\esp12e\firmware.elf
Memory Usage -> http://bit.ly/pio-memory-usage
DATA: [=====] 75.0% (used 61413 bytes from 81920 bytes)
PROGRAM: [=====] 69.5% (used 262073 bytes from 376832 bytes)
[SUCCESS] Took 10.13 seconds
```

Figura 5.9. Compilado del programa.

5.7. 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.

El funcionamiento específico se resume en la figura, que muestra la comunicación cliente-servidor al momento de solicitar un comando.

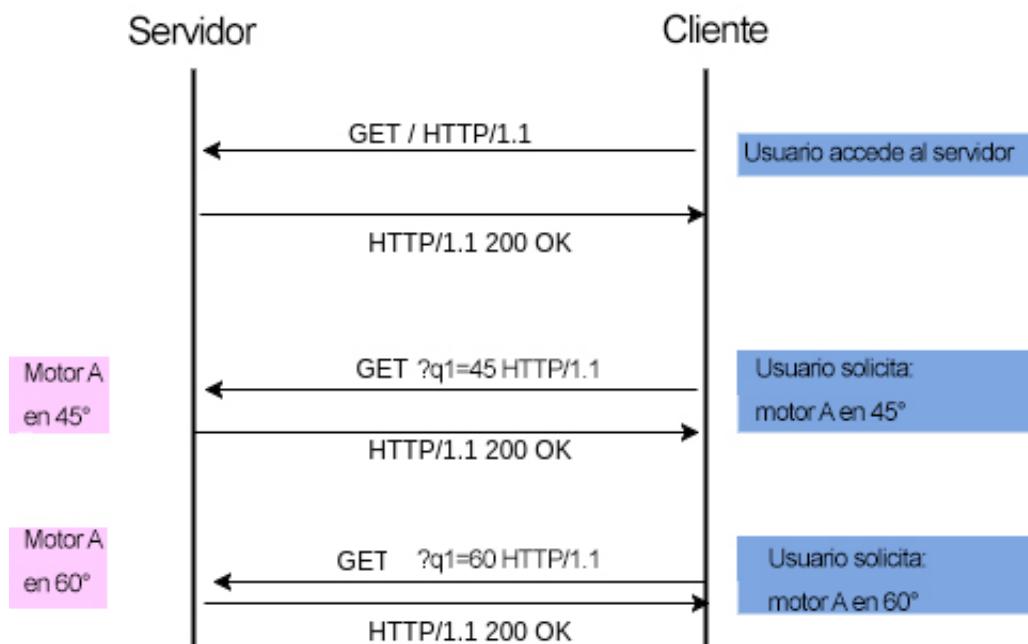


Figura 5.10. Ejemplo de comunicación cliente- servidor.

5.7.1. 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
```

Función de desconexión;

```
begin
    Borrar canal TCP;
    Crear nuevo canal TCP. //Esperar;
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

```

En la tabla 5.7.1, 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 5.5. 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)...
```

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, 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 5.6.

Tabla 5.6. 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 carácter 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.

5.7.2. Función server_recv

Tabla 5.7. Descripción de Función server_recv.

Nombre de función:	server_recv(void *arg, char *pdata, unsigned short len)
Descripción:	Función de respuesta ante recepción de información
Parámetros:	arg – estructura espconn
	pdata – informacion recibida
	len – tamaño de información recibida
Resultado:	none

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 posterior 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.

5.7.3. Función server_sent

Tabla 5.8. Descripción de Función server_sent.

Nombre de función:	server_sent(void *arg);
Descripción:	Función de respuesta ante envío de información
Parámetros:	arg – estructura espconn
Resultado:	none

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 desarrollo. Además, con cada comunicación emitida parpadeará el LED D4.

5.7.4. Función server_discon

Tabla 5.9. Descripción de Función server_discon

Nombre de función:	server_discon(void *arg);
Descripción:	Función de respuesta ante desconexión
Parámetros:	arg – estructura espconn
Resultado:	none

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.

5.7.5. Función server_listen

Tabla 5.10. Descripción de Función server_listen

Nombre de función:	server_listen(void *arg);
Descripción:	Función de respuesta ante un servidor creado exitosamente
Parámetros:	arg – estructura espconn
Resultado:	none

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.

5.7.6. Función server_recon

Tabla 5.11. Descripción de Función server_recon

Nombre de función:	server_recon(void *arg, sint8 err);
Descripción:	Función de respuesta ante un error en la creación del servidor
Parámetros:	arg – estructura espconn
Resultado:	none

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.

5.7.7. Función init_tcp

Tabla 5.12. Descripción de Función init_tcp.

Nombre de función:	init_tcp(uint32_t Local_port);
Descripción:	Función para configurar el servidor
Parámetros:	Local_port – puerto donde se creara el socket
Resultado:	none

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_register_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.

5.7.8. Función *ap_config_func*

Tabla 5.13. Descripción de Función *ap_config_func*

Nombre de función:	<i>ap_config_func();</i>
Descripción:	Funcion para configurar el WiFi
Parámetros:	none
Resultado:	none

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 5.7.1.

5.7.9. Función *gpio_init*

Tabla 5.14. Descripción de Función *gpio_init*

Nombre de función:	<i>gpio_init();</i>
Descripción:	Configura los pines de propósito general
Parámetros:	none
Resultado:	none

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.

5.7.10. Función mover_motor

Tabla 5.15. Descripción de Función mover_motor

Nombre de función:	void mover_motor(int comando, char* recibido,float constante_grados,float constante_offset);
Descripción:	Mueve el motor indicado con el parametro comando
Parámetros:	comando – nombre del motor a mover recibido – string recibido en el request de la pagina
	constante_grados – constante de calibracion para convertir la informacion a un valor binario
	constante_offset – constante de calibracion para corregir el error de bits
Resultado:	none

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 librería *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 5.1 obtenida de la ecuación 3.1.

$$Valor_{digital} = \left(\frac{Valor_{grados}}{K_{pro}} \right) - K_{offset} \quad (5.1)$$

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 3.2.2, 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.

5.7.11. Función mover_motor_2

Tabla 5.16. Descripción de Función mover_motor_2

Nombre de función:	mover_motor_2(int comando, char* recibido);
Descripción:	Mueve el motor indicado con el parametro comando
Parámetros:	comando – nombre del motor a mover
Resultado:	none

Esta función realiza la misma operación que *mover_motor* (5.7.10). 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).

5.7.12. Funciones cambiar_constante, parametro_pid, puenteH

Tabla 5.17. Descripción de Función cambiar_constante

Nombre de función:	cambiar_constante(char* recibido);
Descripción:	Cambia la constante del motor indicado con el parametro comando
Parámetros:	recibido – string recibido en el request de la pagina
	constante_cambiada – constante de calibracion cambiada
Resultado:	none

Tabla 5.18. Descripción de Función parametro_pid

Nombre de función:	parametro_pid(int comando,char* recibido);
Descripción:	Cambia la constante del motor indicado con el parametro comando
Parámetros:	comando – parametro del PID a modificar
	recibido – string recibido en el request de la pagina
Resultado:	none

Tabla 5.19. Descripción de Función puenteH

Nombre de función:	puenteH(int comando,char* recibido,int instrucion);
Descripción:	Modifica el parametro del Puente H
Parámetros:	comando – parametro del PID a modificar
	recibido – string recibido en el request de la pagina
	instrucion – Indica el encendido o apagado
Resultado:	none

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*

5.7.13. Función myatof

Tabla 5.20. Descripción de Función myatof

Nombre de función:	myatof(char *p);
Descripción:	Convierte de ascii a float
Parámetros:	char – string a convertir
Resultado:	valor convertido a float

Esta función se usa para convertir un carácter 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.

CAPÍTULO VI

RESULTADOS

6.1. Interfaz

Para acceder al servidor, primero debe comprobarse que el dispositivo está conectado correctamente. Usando otro dispositivo que cuente con Wi-Fi puede hacerse esto rápidamente, mostrando lo indicado en la figura.



Figura 6.1. Detección de señal Wi-Fi emitida por el microcontrolador ESP-8266.

6.1.1. Página web alojada en servidor

La aplicación principal consta de una página web alojada en el servidor ESP-8266 a la que se puede acceder vía <http://192.168.4.1:8266> que luce como se observa en la figura 6.3.

Al abrir la web, esta solicitará al usuario que ingrese una clave. De ser exitosa, permitirá el ingreso; en caso contrario, enviará al usuario a Google. Esto se observa

en la figura 6.2.

192.168.4.1:8266 dice

Introduzca la clave para entrar

Aceptar Cancelar

(a) Solicitud de clave

192.168.4.1:8266 dice

Clave exitosa! Haga clic para entrar

Aceptar

(b) Clave exitosa

192.168.4.1:8266 dice

Clave incorrecta! Haga clic para ir a Google

Aceptar

(c) Clave incorrecta

Figura 6.2. Solicitud de clave para entrada a la página.

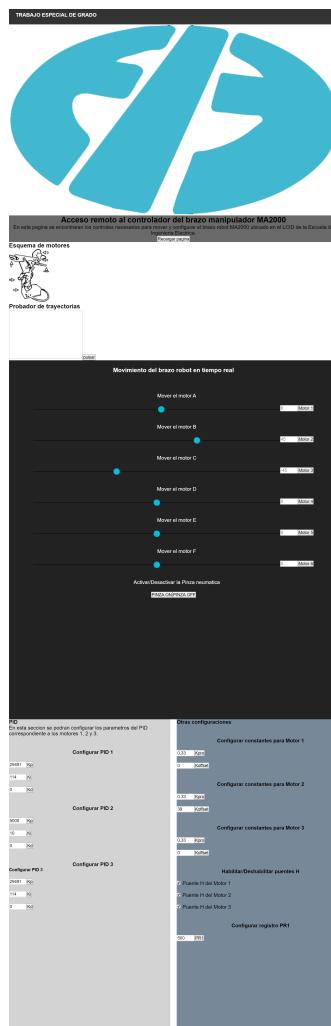


Figura 6.3. Página web completa.

Vista en detalle, las partes de la página son:

Encabezado



Figura 6.4. Encabezado de página

En la figura 6.4 puede verse la presentación de la web, con una imagen alojada en el microcontrolador codificada en base64.

Por otra parte, en la parte inferior puede observarse el botón 'Recargar página' que reconfigura todos los parámetros a sus valores originales, enviando esta trama al brazo robot. Es importante mencionar que cada vez que la página se recargue, esto ocurrirá.

Sección de pruebas

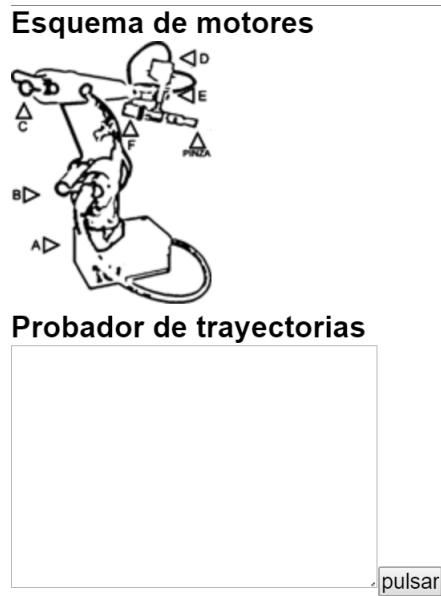


Figura 6.5. Esquema del brazo robot y probador de trayectorias

En la figura 6.5 se observan dos elementos: Una imagen del esquema de motores, codificada en base64. Y un cuadro de texto que indica 'Probador de trayectorias'.

Este probador tiene la finalidad de probar rutinas de movimiento sencillas (sin retardos), con la idea de realizar un bosquejo rápido del movimiento que desea el usuario.

Al presionar el botón 'pulsar', toda la rutina contenida en el cuadro de texto se enviará al brazo robot.

Las rutinas deben tener la estructura siguiente:

q1 -65.00

q2 35.00

q3 -65.00

q4 34.83

q5 -65.01

q6 -34.65

grip on

q1 34.48

q2 -65.02

q3 34.30

q4 -65.04

q5 34.13

q6 65.05

grip off

En estas rutinas, 'qi' es el indicador de movimiento para el motor deseado. Siendo 1, 2, 3, 4, 5 y 6 correspondientes a los motores A, B, C, D, E y F. El número siguiente es el aproximado en grados del movimiento deseado. Puede ser un número entero o un número real de dos cifras significativas. El signo indica si el movimiento es en sentido negativo o no.

Movimiento en tiempo real

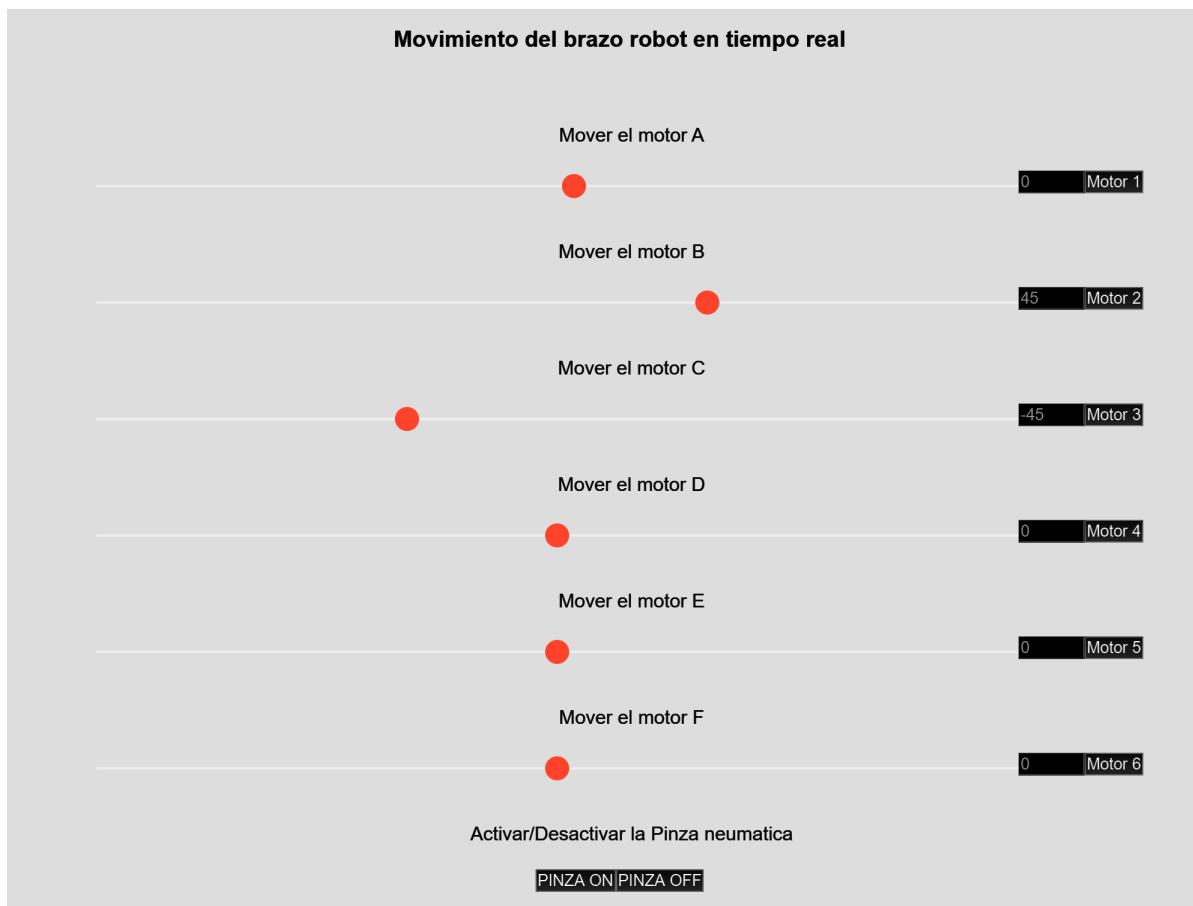


Figura 6.6. Movimiento del brazo en tiempo real

En la figura 6.6, se observan los cursores de movimiento en tiempo real. Al accionar cualquiera de estos, el valor indicado en la casilla de la derecha cambiará y a su vez será enviado al brazo robot. Además, el usuario puede colocar manualmente un valor específico en estas casillas y seguidamente presionar el botón del motor de su selección.

Es importante mencionar que los valores presentados al cargar la página son los enviados por defecto al inicio de la carga de esta.

En la parte inferior se encuentran los botones para la habilitación o deshabilitación de la pinza neumática.

Configuración de parámetros

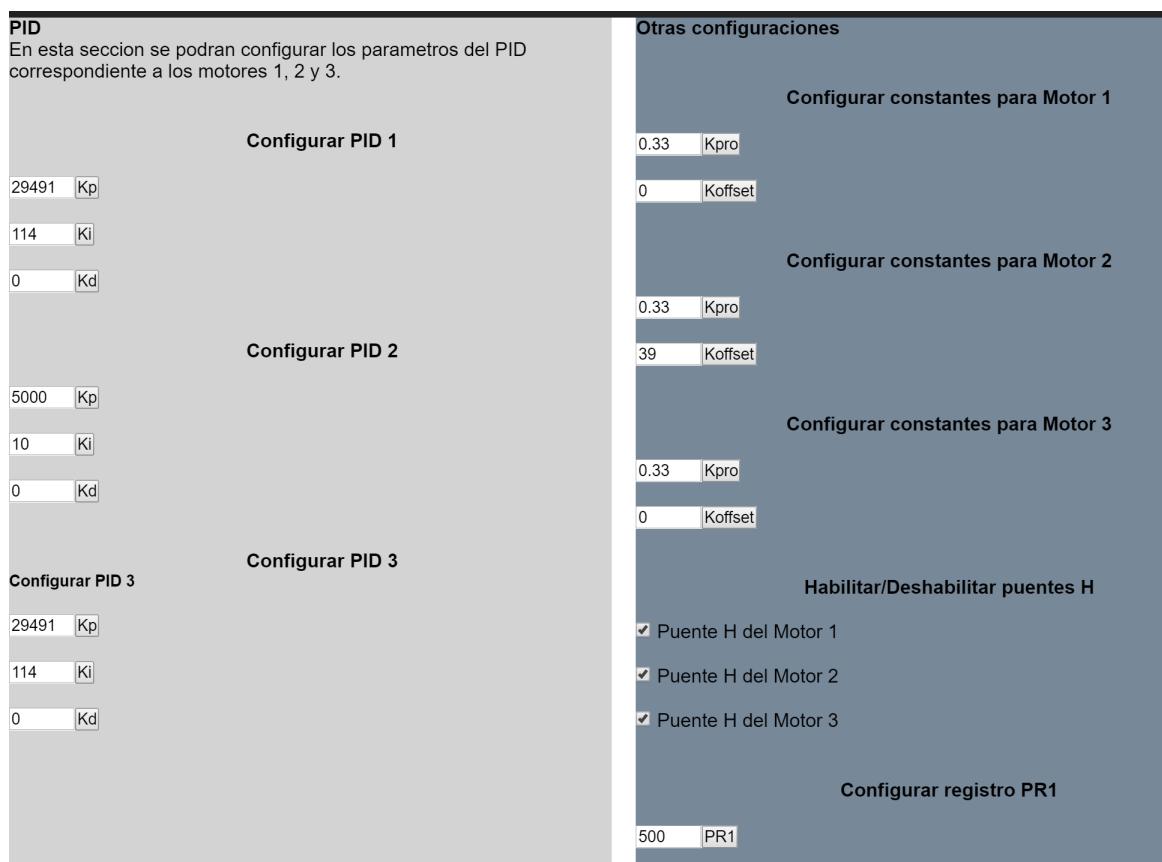


Figura 6.7. Configuración de parámetros generales

En la figura 6.7 pueden observarse los parámetros configurables del brazo robot.

A la izquierda se encuentran los parámetros correspondientes a los tres controladores PID presentes en el robot. Los valores mostrados son los enviados por defecto. Sin embargo, se recomienda al usuario cambiarlos a valores óptimos para evitar oscilaciones indeseadas.

A la derecha, se encuentran los parámetros de calibración para el cálculo del valor de movimiento. Estos son los considerados en la ecuación ??, por lo tanto es importante realizar variaciones solo cuando el usuario esté completamente seguro.

Seguidamente, se observan las opciones para habilitar o deshabilitar los puentes H de los motores principales. No obstante, esto no es recomendable modificar debido a que los motores constantemente realizan movimientos en sentidos positivos y negativos.

Por último, está la opción de configuración del registro PR1, quien se encarga de seleccionar la tasa de muestreo del controlador manejado por el dsPIC. El valor mostrado es el seleccionado por defecto. Es importante mencionar que este valor solo debe ser modificado cuando se esté completamente seguro de sus consecuencias.

6.1.2. Solución a posibles inconvenientes

La aplicación web contenida en la página está desarrollada en JavaScript. Concretamente, con herramientas de JS y de JQuery.

Cada navegador o browser actualizado cuenta con JavaScript. Además, cualquiera que ejecute una página web que utilice JQuery automáticamente descarga esta librería (jquery.min.js), permitiendo que pueda cargarse con mayor rapidez en la siguiente ocasión. No obstante, algunas veces esto no se guarda en la memoria por diversos motivos.

Dentro del código HTML, se hace la solicitud de

<https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js>

Para que el browser integre esta librería debe estar previamente descargada,

de lo contrario ocasionará problemas en la página y no se ejecutará correctamente.

Un método para solucionar esta eventualidad es introducir el enlace anterior en el navegador a utilizar con conectividad a internet. El parámetro '/1/' de la cadena solicita la versión más actual disponible.

En algunos casos, esto no soluciona el problema de conexión; por lo que es necesario ajustar parámetros internos del navegador.

Si se está utilizando Google Chrome (probado con versión 76.0.3809.132) u Opera (probado con versión 63.0.3368.94) se puede acceder a la memoria caché de DNS, y así ajustar el valor por defecto. Esto se hace accediendo a la dirección:

```
chrome://net-internals/#dns
```

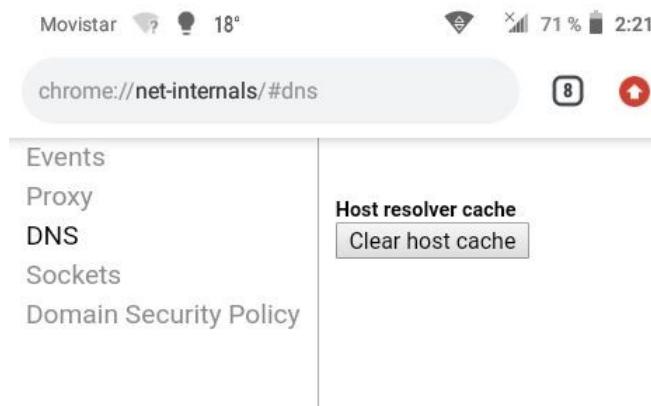


Figura 6.8. Vista de chrome:// net-internals

En la ventana mostrada en la figura 6.8 se debe presionar el botón 'Clear host cache'.

Si el problema de conectividad persiste, es necesario eliminar los valores de cookies, caché y demás datos de navegación y repetir lo explicado al inicio de esta sección (descargar nuevamente la librería jquery.min.js).

6.1.3. Aplicación Android

Esta aplicación fue desarrollada en Android Studio versión 3.4.1 para API 27 (Android 8.1 Oreo) en lenguaje Java.

La funcionalidad de la aplicación es comunicarse a través de un canal TCP con el servidor establecido en el ESP-8266. Allí enviará los comandos generados por el archivo de texto en el que será leído línea por línea. Esto se repetirá tantas veces como quiera el usuario.

En esta aplicación sí se permite el uso de comandos de retardo (wait). Un ejemplo es la siguiente rutina:

```
q1 -74
q3 -110
wait 200
q1 -73
q3 -109
wait 200
q1 -72
q3 -108
wait 200
q1 -71
q3 -107
wait 200
```

Los comandos 'wait' deben ir acompañados de un valor numérico entero que simbolizará un retardo en milisegundos.

En la figura 6.9 se observa la vista en miniatura desde el menú de aplicaciones.

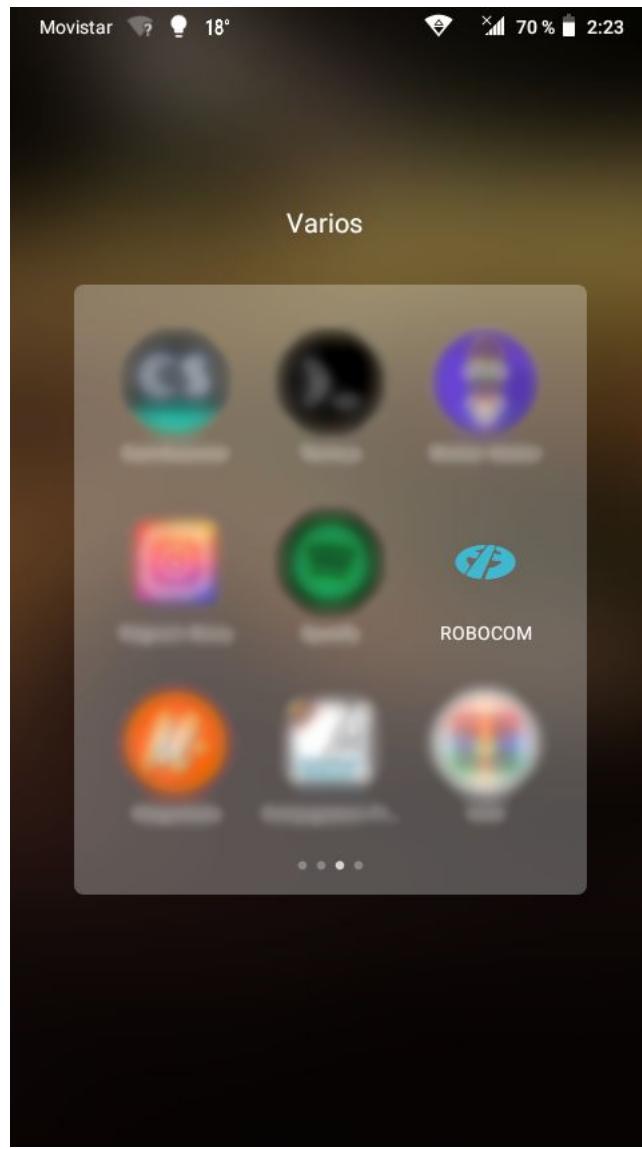


Figura 6.9. Aplicación vista desde el menú.

En la figura 6.10 se observa la interfaz de la aplicación.

El banner muestra el nombre de esta, 'ROBOCOM'. Seguidamente, el cuerpo de esta establece un campo de texto editable con etiqueta 'Número de repeticiones'. En esta opción se coloca el número de veces que desea repetirse la rutina. Al dejar el valor en '0', el programa se ejecutará una sola vez.

A un lado se encuentra la etiqueta 'Status', que indica si la comunicación fue exitosa o no. De serlo, marcará el número '200'; en caso contrario marcará '404'.



Figura 6.10. Vista principal de la aplicación Android.

Al centro de la aplicación está el campo de texto en el que se posará el texto de la rutina. Allí pueden escribirse los comandos de movimiento y configuración directamente o desde un archivo de texto que puede ser cargado al presionar el botón 'Cargar Archivo'.

Las rutinas generadas o modificadas pueden ser guardadas en un nuevo archivo indicando el nombre y presionando la opción 'Guardar a archivo'.

6.2. Montaje

A continuación se muestran los diagramas esquemáticos de conexión de la tarjeta de desarrollo ESP-8266 con el dsPIC del controlador del brazo robot.

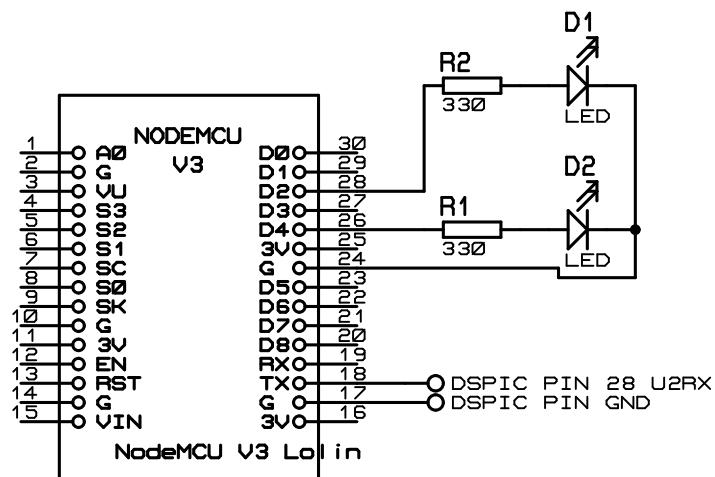


Figura 6.11. Esquemático de conexión entre ESP-8266 y dsPIC30F3011

CAPÍTULO VII

CONCLUSIONES

CAPÍTULO VIII

RECOMENDACIONES

Conectividad a internet

Otros métodos de control

Aprendizaje de máquinas

Realidad aumentada

Otros dispositivos con Wi-Fi

REFERENCIAS

- [1] J. Marcano, *Implementación de sistema de programación de Trayectorias para el brazo manipulador MA2000*. Caracas: Escuela de Ingeniería Eléctrica, Universidad Central de Venezuela, 2013.
- [2] A. Labrador, *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, 2018.
- [3] J. Valderrama, *Diseño de un conjunto de prácticas para la configuración y uso básico del microcontrolador ESP8266*. Caracas: Escuela de Ingeniería Eléctrica, Universidad Central de Venezuela, 2018.
- [4] A. Md. Bony, “Design and implementation of a remote controlled robotic arm based on industrial application perspective,” *International Conference on Mechanical, Industrial and Materials Engineering 2013 (ICMIME2013)*, 2013.
- [5] M. VS, “Automatic pick and place robot manipulation using a microcontroller,” *Journal of Applied and Computational Mathematics*, vol. 7, no. 3, pp. 1–8, 2018. DOI: 10.4172/2168-9679.1000408.
- [6] G. L. Neri, “Robot puma 560,” 16 de marzo de 2013. [Internet; accedido el 13-09-2019] Disponible en: <https://grabcad.com/library/robot-puma-560>.
- [7] B. García, “George devol, el creador de la robótica industrial,” 20 de agosto de 2018. [Internet; accedido el 20-09-2019] Disponible en: <https://blogthinkbig.com/george-devol-el-creador-de-la-robotica-industrial>.

- [8] R. Miller, L. Eriksson, L. Fleisher, J. Wiener-Kronish, and W. Young, *Anesthesia E-Book*. Elsevier Health Sciences, 2009.
- [9] NA, “Csa’s canadarm 1 and 2,” 10 de agosto de 2018. [Internet; accedido el 12-08-2019] Disponible en: <https://sites.google.com/site/issscienceproject/csa-s-candarm>.
- [10] G. L. Karl H. Doetsch, “Canadarm,” 4 de marzo de 2015. [Internet; accedido el 12-08-2019] Disponible en: <https://www.thecanadianencyclopedia.ca/en/article/canadarm>.
- [11] G. L. Karl H. Doetsch, “Canadarm,” 4 de marzo de 2015. [Internet; accedido el 12-08-2019] Disponible en: <http://www.riscy.uk/beebcontrol/arms/ma2000/index.html>.
- [12] M. Alam and Jamil, “Design and implementation of a rf controlled robotic environmental survey assistant system,” pp. 438–442, 03 2014.
- [13] A. Aru, “Design exploration of a microcontroller based rf remote control 13amps wall socket,” *IOSR Journal of Computer Engineering*, vol. 11, pp. 56–60, 01 2013.
- [14] M. Yusoff and Samin, “Wireless mobile robotic arm,” *Procedia Engineering*, vol. 41, pp. 1072–1078, 12 2012.
- [15] G. Kaustubh, “Wireless mobile robotic arm,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 04, pp. 757–759, 3 2017.
- [16] Espressif Systems, *ESP 8266 Technical Reference version 1.3*, 2017.
- [17] NA, “Esp8266 (parte 1) : programación en lua y arduino,” 25 de mayo de 2017. [Internet; accedido el 27-09-2019] Disponible en: <http://elb105.com/esp8266-parte-1-programacion-en-lua-y-arduino/>.

- [18] NA, “Esp8266 (parte 2) : programación en c, simba y freertos,” 26 de mayo de 2017. [Internet; accedido el 27-09-2019] Disponible en: <http://elb105.com/esp8266-parte-2-programacion-en-c-simba-y-freertos/>.
- [19] C. Svec, “Freertos,” 26 de abril de 2012. [Internet; accedido el 27-09-2019] Disponible en: <https://www.aosabook.org/en/freertos.html>.
- [20] Espressif Systems, *ESP 8266 Non-OS SDK API Reference version 3.0.1*, 2019.
- [21] Espressif Systems, *ESP 8266 SDK Getting Started Guide version 2.0*, 2016.
- [22] Espressif Systems, *ESP8266EX Datasheet version 6.0*, 2018.