UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
INGENIERÍA ELÉCTRICA – SISTEMAS ELECTRÓNICOS

SISTEMA DE CONTROL PARA MANIPULACIÓN DE OBJETOS EN UNA CELDA DE MANUFACTURA FLEXIBLE

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
ING. IVÁN GONZÁLEZ GARCÍA

TUTOR PRINCIPAL
DR. JUAN MARIO PEÑA CABRERA, IIMAS-UNAM

MÉXICO, CDMX, MARZO DE 2019

JURADO ASIGNADO

Presidente:	Dra. Navarrete Montesinos Margarita
Secretario:	Dr. Maya Ortiz Paul Rolando
Vocal:	Dr. Peña Cabrera Juan Mario
1er. Suplente:	Dra. Moumtadi Fátima
2do. Suplente:	Dr. De La Rosa Nieves Saúl

Ciudad Universitaria, Ciudad de México.

Tutor de Tesis

Dr. Peña Cabrera Juan Mario

_____

Firma

Agradecimientos

A la Universidad Nacional Autónoma de México por permitirme continuar creciendo profesional y personalmente.

A mi tutor Dr. Juan Mario Peña Cabrera quien siempre creyó en mí y mantuvo su apoyo, ayudándome a concluir esta faceta en mi vida, tanto con sus conocimientos como por los consejos que me brindo.

A la coordinación de Estudio de Posgrado (CEP) de la UNAM y al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico durante mis estudios, así como al Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS) por facilitarme sus instalaciones para el desarrollo del proyecto.

A mis compañeros y amigos de generación, ha sido un gusto conocerlos y cursar los estudios de maestría a su lado.

A mis amados padres Socorro y Javier quienes siempre me ayudan e impulsan en los planes de vida que decido forjarme, así como a mis queridos hermanos Javier y Andrés ques siempre me apoyan para alcanzar mis propósitos.

A mi corazón, muchas gracias Gie Bele por todo tu amor y cariño, quiero decirte que te admiro mucho, siempre me incitas a querer ser mejor en todos los aspectos. Te amo demasiado, gracias por llegar a mi vida, me siento muy afortunado.

Resumen

Las innovaciones en la tecnología avanzan muy rápidamente, los desarrollos de proyectos son cada vez más complejos y multidisciplinarios, los entornos donde se utilizan máquinas automatizadas e inteligentes que utilizan las nuevas tecnologías con ideas innovadoras para realizar aplicaciones de vanguardia son más necesarios. Tal es el caso de un sistema para el control de diversas tareas dentro de una celda de manufactura con características de flexibilidad y reconfiguración, en donde las tareas se puedan planificar y operar por medio de una interface hombre-máquina (HMI) que les brinde interactividad y mayor eficiencia a los procesos de manufactura.

El objetivo del proyecto consiste en diseñar, implementar e integrar un sistema de control digital para manipular objetos de manufactura desde una posición fija inicial y manipularlos sobre diferentes puntos de estaciones de trabajo dentro de una celda de manufactura, y así, darle una característica de flexible y configurable. Para conseguir el objetivo y efectuar diferentes tareas de manufactura como un proceso integral en la elaboración de algún producto manufacturado con la celda, se diseñó e integró un planificador de tareas.

El desarrollo del presente proyecto implica el uso de tecnologías para la transmisión inalámbrica de datos, procesamiento de datos por computadora y manejo de señales con microcontroladores electrónicos que conforman los elementos que controlan la manipulación de objetos entre diferentes estaciones de trabajo que realizan las distintas tareas en las aplicaciones de manufactura.

Para la transmisión y recepción de datos se hace uso de dispositivos electrónicos con el protocolo *zegbee,* microcontroladores de ocho bits para el manejo y distribución de las señales, una PC que alberga una interface HMI y un brazo robot de seis grados de libertad, para conformar los elementos de una celda de manufactura flexible.

La celda de manufactura entonces, se integra con los elementos mencionados para crear un espacio de trabajo para la fabricación de un producto final que requiera de diferentes procesos para su terminado, cada proceso se realiza en una estación de trabajo con tareas bien definidas y bajo el control de un planificador de tareas.

En el proceso, el insumo o producto inicial es tomado por un brazo robótico y transportado hacia la estación de trabajo correspondiente para realizar la tarea especificada.

El sistema de control digital desarrollado, consiste de la integración de piezas electrónicas para el proceso de las señales que comandarán las acciones a realizar en cada una de las estaciones de trabajo de la celda de manufactura, las tareas son previamente estructuradas por un planificador de tareas basado en un modelo de las posibles acciones a realizar con el conjunto de los elementos de la celda e implementado en una interfaz HMI en una computadora PC, el funcionamiento integral de los componentes proporciona la característica de flexibilidad y reconfiguración al sistema de celda de manufactura.

Contenido

Capítulo 1.

Introducción

Una de las líneas de trabajo en el Departamento de Ingeniería de Sistemas Computacionales y Automatización del IIMAS-UNAM es la investigación en el desarrollo de sistemas inteligentes de manufactura (SIM), la integración de un sistema de este tipo involucra tener estaciones de trabajo autónomas e inteligentes que realizan diferentes tareas como: reconocimiento de objetos por medio de visión artificial, maquilado de una pieza de manufactura, pintado, doblado, transporte de materiales y objetos, y empacado entre otras.

Considerando la importancia económica que representa para el país el desarrollo de SIM's, el IIMAS-UNAM, decidió realizar investigación aplicada en integración de arquitecturas inteligentes robustas y capaces de interactuar en ambientes no estructurados donde se requiere gran adaptabilidad.
En el trabajo de primera instancia se introduce la noción de un Sistema Inteligente de Manufactura (SIM), seguido por la descripción de los elementos de su arquitectura. Posteriormente, se presentan las etapas de fabricación de componentes y su ensamble en un producto prototipo de manera autónoma.

Un sistema flexible e inteligente de manufactura consiste y se caracteriza por tener una celda de trabajo, un manejo y transporte automatizado de piezas y un sistema de control normalmente implementado por medio de un procesador. Para facilitar el diseño de FMS se han propuesto metodologías de meta-análisis, que pretende utilizar los aspectos estratégicos, financieros y de operaciones de manera integral para facilitar el diseño. Estudios de prospectiva tecnológica indican que los procesos de manufactura del futuro estarán fuertemente caracterizados por la necesidad de adaptarse a las demandas de manufactura ágil, incluyendo una rápida respuesta a los cambios en los requerimientos del cliente, diseño e ingeniería concurrente, bajos costos en volúmenes pequeños de producción, manufactura distribuida, entrega justo-a tiempo, planeación y calendarización en tiempo-real, incremento en la demanda por precisión y calidad, disminución en tolerancia de errores y mediciones del proceso *in situ*.

Las demandas del mercado propiciaran que los productos se fabriquen en el punto de consumo, que la manufactura sea completamente distribuida y orientada a sistemas *holónicos*. La Manufactura Holónica es parte del Programa de Sistemas Inteligentes de Manufactura que involucra a varios países desarrollados y que intenta resolver la fragilidad de los sistemas actuales de producción reemplazando sistemas rígidos e inflexibles por sistemas más adaptables al cambio; en este aspecto los holones ocupan el rol de intermediarios jerárquicos. Por lo tanto, los holones son autónomos, unidades discretas y cooperativas capaces de lidiar con perturbaciones y aun proveer la funcionalidad de soportar el gran *todo* y por lo tanto incrementar la robustez del sistema.

1.1 Planteamiento del problema

El diseño de Sistemas Flexibles de Manufactura (FMS, por sus siglas en inglés), basado en una arquitectura holónica, es una tarea compleja debida a la amplia variedad de alternativas de control y configuraciones disponibles para el diseñador. Para llevar a cabo múltiples tareas definidas en un proceso de manufactura, se presenta como un problema a resolver la necesidad de realizarlas de manera organizada y automatizada con un sistema que así lo permita, y tomando en cuenta múltiples criterios como costos, flexibilidad, calidad, riesgos financieros y técnicos. El correcto diseño e implementación de un sistema que resuelva esta necesidad, se presenta como el problema a resolver.

La elección del hardware y software para el control de la celda son fundamentales para obtener un funcionamiento óptimo, así como el diseño y la implementación de los algoritmos para la ejecución de los dispositivos que comandaran las estaciones de trabajo utilizadas.

En general una celda de manufactura flexible no está atendida por humanos por lo que su diseño y operación debe ser preciso, utilizando un sistema computacional que le permita comunicarse con el exterior y tener el control sobre la maquinaria específica utilizada dentro de la celda de manufactura a través de una interface hombre-máquina.

1.2 Propuesta de solución

Diseñar e implementar un sistema de control para manipulación de objetos en una celda de manufactura flexible, representa la solución para el problema planteado anteriormente, y necesita del diseño e implementación de un *"planificador de tareas"* para atender la ejecución de los procesos dentro de una arquitectura de celda de manufactura inteligente y flexible.

Para cumplir con los criterios (costos, flexibilidad, calidad, riesgos financieros y técnicos) del ordenamiento del proceso de manufactura y sus tareas a ejecutar, se requiere de integrar el proceso de adquisición, el procesamiento y clasificación de datos, y la manipulación de un brazo robótico con sistemas implementados con dispositivos electrónicos y computacionales. Tener un planificador de tareas representa ventajas importantes como la capacidad de procesamiento con un sistema reducido de elementos, reducción de costos y la posibilidad de modificar el código y señales de control para darle una característica de reconfigurable.

1.3Justificación

La gran demanda que existe en la actualidad en la industria para la producción masiva de productos especializados y con características particulares, requiere de menos tiempo para la fabricación de un determinado producto, utilizar sistemas de manufactura avanzada e inteligentes en los procesos significa un factor muy importante para el desarrollo del sector industrial. La necesidad de automatizar los procesos, ha dado como resultados utilizar mejores herramientas y equipos. Los sistemas de automatización en la industria, y enfocados a los procesos de producción en donde exista alto riesgo de seguridad para trabajadores humanos, han obtenido en los robots industriales un gran aliado, estos han ido formando parte de las plantas productoras a gran escala en las tareas de manejo y suministro de material en las que el robot es una parte importante, se utilizan para transferir piezas a la entrada del proceso de producción o bien para recogerlas al final del mismo, para ello, se requiere de planificar las tareas a realizar de forma inteligente, organizada y eficiente, lo que justifica el desarrollo del proyecto de esta tesis.

1.4Objetivos

Objetivo General

El objetivo principal de este trabajo es el diseño y la implementación de un sistema controlador de tareas para una celda de manufactura flexible e inteligente, implementada con dispositivos electrónicos, que permita la manipulación de objetos entre las diferentes estaciones de trabajo y la comunicación entre ellas.

Objetivo especifico

Con el diseño electrónico realizado, Integrar una maqueta experimental de celda de manufactura flexible para simular procesos de manufactura y controlarlos con un planificador de tareas, con procesos como insumos de material a un lugar fijo y destinado controlando una banda transportadora, la manipulación de objetos con un brazo robot en la parte central de la maqueta para tomar piezas y colocarlas en diferentes estaciones de trabajo donde se realizarán procesos dependiendo de las tareas asignadas. Cada estación de trabajo tendrá un microcontrolador que estará comunicado con un protocolo de comunicación inalámbrico con la computadora central que alberga una interfaz HMI para indicar al planificador el estado del proceso en la estación de trabajo y con el que se comandarán las acciones de operación de actuadores que realizan la tarea en proceso.

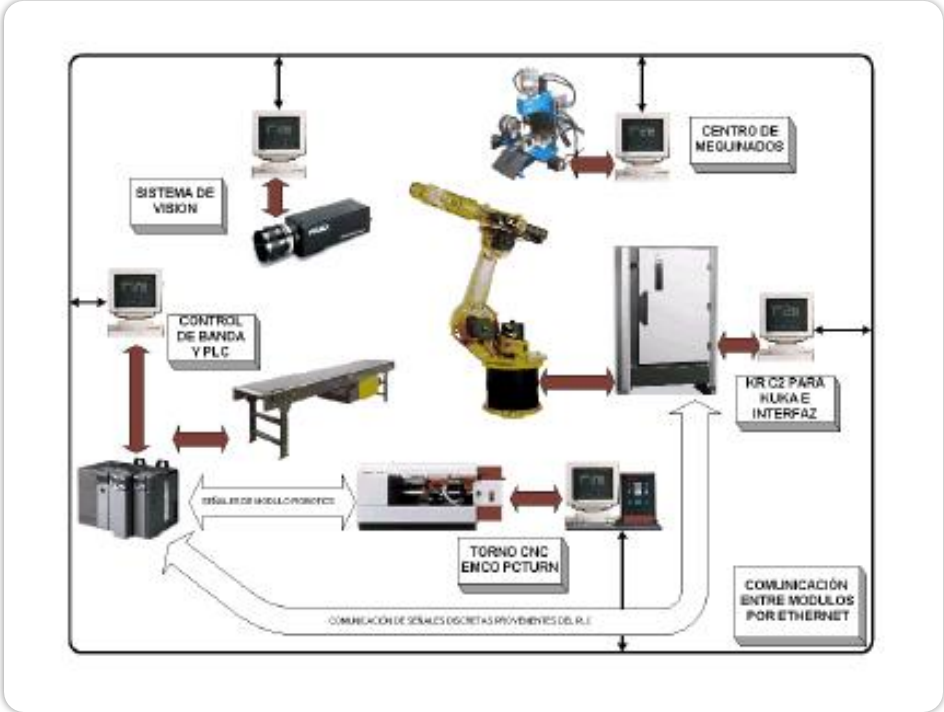


Fig. 1 Esquema general de una celda de manufactura. Ejemplo de arquitectura de un Sistema Inteligente de Manufactura- con un robot industrial tipo KUKA KR 15/2 de seis grados de libertad, con carga útil de 15 Kg, (5).

1.5 Metodología

Para el diseño de la maqueta experimental del SIM utilizada en el proyecto de esta tesis, se consideró en primera instancia el desarrollo de los elementos inteligentes de la celda (holones) de manera particular según la tarea a realizar.
Integrar un robot de seis grados de libertad LabotPro5 capaz de realizar transporte de materiales entre zonas de la celda, posteriormente incorporarlos elementos de producción, que en su conjunto, proveen al SIM de las capacidades necesarias para maquinar los componentes de un producto y realizar un proceso de manufactura autónomo.
Conocer las características de los componentes a utilizar, situarlos de tal manera que su ubicación permita la interacción entre ellos y sea posible manipular los objetos como es debido dentro de la maqueta de celda de manufactura. Posteriormente, realizar la configuración mediante software.

Para la ejecución de los movimientos que se necesitan, se diseña e implementa el programa que los realiza. En el programa, se describen las secuencias del proceso que se ejecutaran en cada estación de trabajo.
Determinar el tipo de red y protocolos para la comunicación entre los diferentes dispositivos que se utilizan.
Finalmente implementar el planificador de tareas y las interconexiones de comunicación inalámbrica para dar paso a la experimentación en la maqueta de la ejecución de tareas y mostrar el proceso de manufactura en su conjunto.
La interfaz gráfica permite la comunicación hombre-máquina con el sistema y lo que establece el monitoreo, la configuración y almacenamiento de los procesos a realizar.

1.6 Trabajo Previo

El desarrollo de un sistema planificador de tareas que se encargue de coordinar y organizar las tareas a realizarse en un sistema flexible de manufactura se muestra en(1), una arquitectura del tipo "holónica" que pueda ser reconfigurada y pueda comandar las acciones tanto de actuadores y sensores dentro de estaciones local es presentada en(2)(3).

En (4) se presentan acciones que se llevan a cabo mediante comandos que son transmitidos por una computadora maestra que alberga el planificador, los elementos de la arquitectura son identificados y mediante una serie de dispositivos electrónicos se lleva a cabo la transmisión de datos de comandos a las diferentes estaciones de trabajo para efectuar las tareas en tiempo real.

La producción de lotes manufacturados por un Sistema Inteligente de Manufactura (SIM) adquiere gran importancia en las metodologías de manufactura avanzada. En los países industrializados, alrededor del 30% del Producto Interno Bruto (PIB) proviene de las manufacturas, 40% de éstas se producen en lotes y solo el 15% en masa, y de la fabricación en lotes, el 75% son menores a 50 unidades. La fuerte dependencia tecnológica del exterior que implica costos de asesoría y de uso demarcas y patentes justifica contar con diseños propios de planificadores de tareas automatizados. Datos muestran que, de 34,895 empresas altamente representativas de la economía mexicana, 77.1% emplea tecnología obsoleta, 19.5% cuenta con equipo moderno vulnerable, 2.9% tiene fuerza tecnológica, pero carece de capacidad competitiva estratégica, y únicamente el 0.5% utiliza tecnología de punta. Así mismo, un sector atractivo es la automatización de los procesos de ensamble ya que estudios revelan que estos procesos representan un 20% de los costos unitarios de producción. Se mencionan los avances al respecto y finalmente se presentan las conclusiones considerando el desarrollo futuro del SIM, (5).

La mayor eficiencia, la reducción de costos y recursos materiales obtenida en manufactura con sistemas automatizados, se presenta en(6), donde se clasifican los modelos establecidos para el diseño de FMS categorizados como: técnicas de medición de desempeño y métodos, estimación de costos y métodos, y técnicas de análisis de decisión y métodos.

La automatización de los procesos se vuelve fundamental para el desarrollo del sector industrial(7),(8). La necesidad de automatizar ha dado como resultados la mejora de las herramientas y equipos que se utilizan; ya sean muy básicos a nivel artesanal o industrial. En lo concerniente al ramo industrial, los sistemas de automatización, enfocados a los procesos de producción de alto riesgo ambiental, han obtenido en los robots industriales un gran aliado, los cuales han ido formando parte de las plantas productoras a gran escala.

Son muy comunes las tareas de manejo y suministro de material en las que el robot es una parte importante, pues se utiliza para transferir piezas a la entrada del proceso de producción o bien para recogerlas al final del mismo. Por tanto, es necesario que el robot pueda discriminar entre diferentes

piezas: tamaños, formas o colores,(9), (10), (11).La discriminación de piezas, puede ser tan simple, que se sacrifique la confiabilidad del sistema; o tan exacta, que comprometa la velocidad de procesamiento,(12). La adaptación de sensores inalámbricos en la robótica industrial no solo tiene un impacto visual estético; la más grande aportación de un sistema escasamente cableado, es la seguridad. Una cantidad importante de tendidos de cables contribuye en gran medida, a tener una condición insegura de trabajo para el operador. También se podrán eliminar las fallas por conexiones defectuosas o envejecimiento de conductores, así como posibles capacitancias parásitas en los mismos, que arrojen falsas lecturas de los transductores y evitar los procedimientos de instalación y los costos que implican,(13).
En la actualidad, las aplicaciones son muy diversas como es el caso de empresas embotelladoras, empacadoras de embutidos, compañías de reciclaje, etc., donde el proceso de selección puede ser muy repetitivo y en muchos de ellos se requiere de precisión y automatización. De esta manera, se puede evitar la intervención del hombre en ambientes hostiles, mejorando su calidad a nivel laboral, y reduciendo el índice de accidentes y acelerando de forma segura la producción.

1.7 Antecedentes

La robótica es un tema de estudio constante y progresivo en todo el mundo. Desde la consolidación de robots industriales como alternativas de operación económica, eficiente, robusta y flexible, y el estudio de sus arquitecturas; los aspectos de interés y estudio han sido hasta nuestros días proveer a estas máquinas con toda la información y herramientas necesarias para realizar su trabajo más rápido, preciso y seguro y de manera inteligente y automatizada. La interfaz entre la persona y la máquina es conocida como HMI y significa “Human Machine Interface”, es decir es el dispositivo o sistema que permite la interacción entre el sistema y el operador humano.

Tradicionalmente estos sistemas consistían en paneles compuestos por indicadores y comandos, tales como luces pilotos, indicadores digitales y análogos, registradores, pulsadores, selectores y otros que se interconectaban con la máquina o proceso. En la actualidad, dado que las máquinas y procesos en general están implementadas con controladores y otros dispositivos electrónicos que permiten el acceso de puertos de comunicación, es posible contar con sistemas de HMI bastantes más poderosos y eficaces, además de permitir una conexión más sencilla y económica con el proceso o máquina a interactuar(14).

Un sistema HMI está constituido de las siguientes partes:

Terminal de Operador- El cual consistente en un dispositivo, generalmente construido para ser instalado en ambientes agresivos, donde pueden ser solamente de despliegues numéricos, o alfanuméricos o gráficos. Pueden ser además con pantalla sensible al tacto (touch screen)

PC + Software- Constituye una alternativa basada en un PC en donde se carga un software apropiado para la aplicación. Como PC se puede utilizar los llamados Industriales (para ambientes agresivos), los de panel (Panel PC) que se instalan en gabinetes dando una apariencia de terminal de operador, y de igual modo una PC de escritorio o laptop.

Software HMI: Estos softwares permiten entre otras cosas las siguientes funciones: Interfaz-gráfica a modo de poder ver el proceso e interactuar con él, registro en tiempo real e histórico de datos, manejo de alarmas. Se requiere de una herramienta de diseño o desarrollo, la cual se usa para configurar la aplicación deseada, y luego debe quedar corriendo en el PC un software de ejecución (Run Time). Este software puede comunicarse directamente con los dispositivos externos (proceso) o bien hacerlo a través de un software especializado en la comunicación.

Comunicación: La comunicación con los dispositivos de las máquinas o proceso se realiza mediante comunicación de datos empleando las puertas disponibles para ello, tanto en los dispositivos como en los PCs. Actualmente para la comunicación se usa un software denominado servidor de comunicaciones, el que se encarga de establecer el enlace entre los dispositivos y el software de aplicación (HMI u otros) los cuales son sus clientes.

El uso de tecnología inalámbrica permite la construcción de celdas de manufacturas que requieran de varios puntos los cuales pueden estar comunicados para el intercambio de información y llevar un registro del producto que se está realizando. Características de las tecnologías utilizadas en el proyecto para la configuración de los elementos por software, se muestran en el *Anexo 1.*

Para el proyecto y la integración de la maqueta experimental se utilizaron los siguientes componentes y cuyas características técnicas para los elementos de hardware se describen en el *Anexo 2.*

- Computadora de escritorio o laptop
- Microcontroladores
- Unidades Xbee
- Brazo robótico
- Actuadores varios
    - Motor DC
    - Relevador
    - Display LCD 16x2
    - Ventilador
    - Bomba de agua

Capítulo 2

Desarrollo

2.1 Plan de trabajo

- Establecer los componentes en el sistema de manufactura flexible que se necesitan para la realización del proyecto.
- Determinar las piezas que serán monitoreadas para su manipulación.
- Implementación de los módulos para las estaciones de trabajo con dispositivos electrónicos.
- Desarrollo del software (HMI) necesario para la comunicación de los componentes.
- Realizar la simulación del mecanismo para la verificación de los movimientos, así como el proceso o acción a realizar en el objeto y su colocación en el lugar correspondiente.
- Verificar si los métodos son los adecuados o pueden mejorarse
- Describir el procedimiento alcanzado, así como sus posibles adaptaciones dependiendo del entorno donde se requiera establecer.

Evolución de la Arquitectura Inteligente de Manufactura

La arquitectura consistió inicialmente de un sistema robótico básico, como se muestra en la Figura1. En ella se muestra una celda que estaba constituida por un robot industrial y su controlador, computadora esclava, sensor de Fuerza/Torque y computadora maestra. El controlador del robot integra los elementos de control potencia hacia los servomotores del robot. La computadora maestra se comunica con el controlador mediante dos puertos seriales: El primero en modo “supervisorío”, a través del cual se envían comandos de alto nivel hacia el controlador y el otro en modo “Alter” a través de la computadora esclava para movimientos incrementales finos.

Un ejemplo de un proceso con este tipo de metodología se muestra en la figura 2. Para realizar un ensamble inteligente.

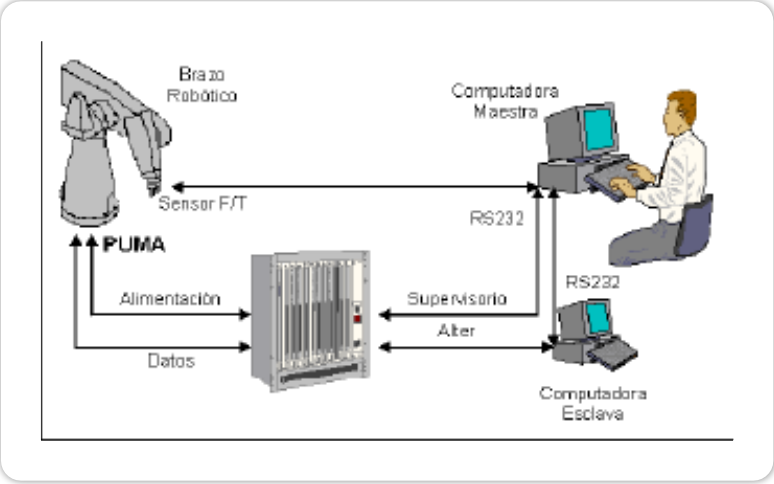


Fig. 2. Arquitectura inicial de ensamble inteligente.

Tomando como base esta arquitectura se integran los elementos de percepción sensorial y producción, La arquitectura de comunicación es similar a la de la figura 1, con la diferencia de que no existe una computadora esclava, (5).

La arquitectura que se implementa en el desarrollo del proyecto está basada en la mostrada anteriormente. La celda de manufactura se compone de un brazo robótico manipulador, las estaciones de trabajo son colocadas dentro del perímetro de alcance del brazo y la computadora es la encargada de la interacción entre ellas, la cual está situada a un costado (fig. 4).

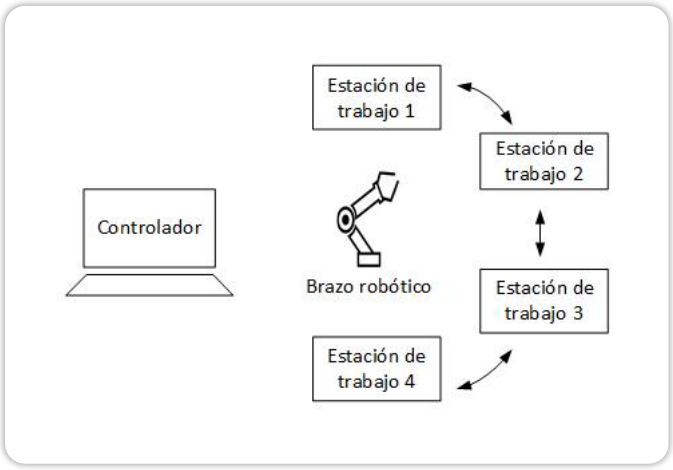


Fig. 3. Arquitectura general de Celda de manufactura implementada.

Son cuatro estaciones de trabajo las que se implementan. Las cuales tendrán un microcontrolador, encargado de manejar las entradas y salidas de los actuadores para las tareas a efectuar, con base a los datos enviados por el controlador de la interfaz. Se crean diferentes programas para cada estación y en estas se llevan a cabo las instrucciones que solicitemos. Por lo tanto, cada estación procesara diferentes acciones para ser aplicadas a los objetos.

La primera estación de trabajo, es el área insumo, será la encargada del alimentar y transportar el material a manufacturar.
La segunda estación, es el área de barrenado, simula la acción de un taladro y se encargará de realizar un barrenado a la pieza. Se dispondrá de cuatro posibles formas de trabajo. Las cuales son designadas, aparentando el acabado de acuerdo al material que se manufactura, como podría serlo duro o suave.
En la tercera estación se implementará un área de limpieza. En ella se podrá dar un tratamiento a las piezas. Dentro de las tareas que se pueden asignar se encuentran la de lavado, enjuagado y secado.
Para la cuarta estación se simulará la acción de una estación CNC. Manejara cuatro diferentes tipos de terminado. Mandará el mensaje de la forma elegida en la interfaz y la señal de tarea en proceso.

La comunicación entra las estaciones de trabajo, el brazo robótico y la computadora central con la interfaz, se ejecutará de forma inalámbrica, mediante dispositivos Xbee, los cuales permiten enviar y recibir datos, permitiendo el remplazo de los cables de unión en la comunicación serial.

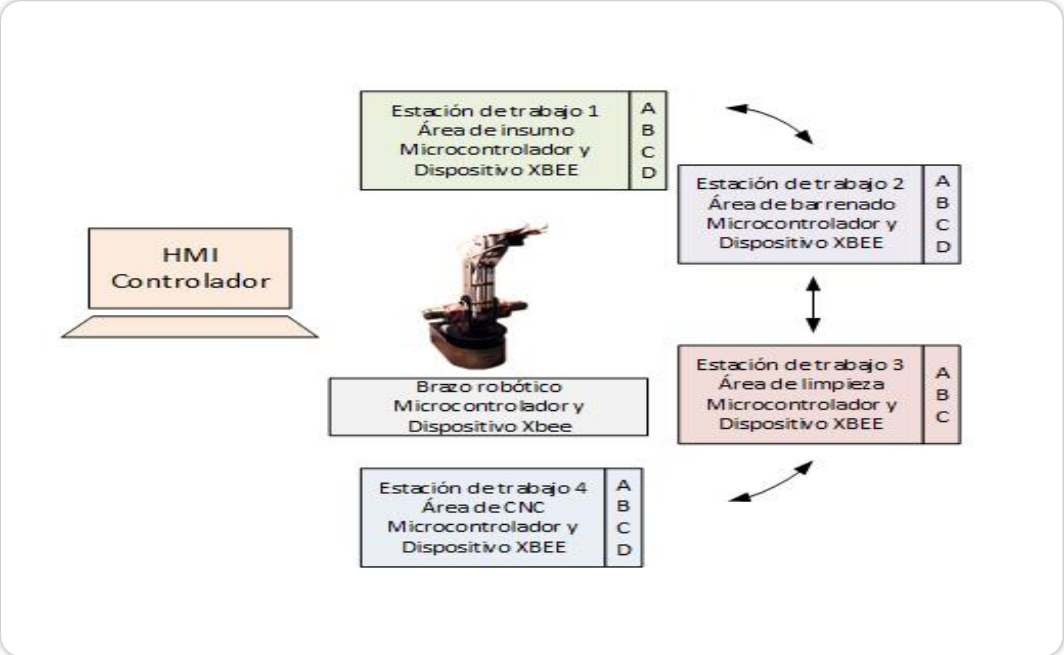


Fig. 4. Arquitectura de Celda de manufactura conforme al proyecto.

Las secuencias de las instrucciones para la operación de la celda de manufactura se producirán de acuerdo al plan asignado mediante el controlador, ubicado dentro de la interfaz HMI.

Para poder llevar a cabo el proceso dentro de la celda de manufactura se debe implementar un Sistema Controlador, que coordine las tareas asignadas y permita la comunicación con las diferentes estaciones de trabajo.

La interfaz (HMI) creada, permite asignar las diferentes tareas que se aplicaran sobre las piezas a trabajar. La interfaz será la encargada procesar las señales recibidas y enviará los datos a las estaciones para ordenar los movimientos dentro de la celda.

En el controlador se asignan las labores para comenzar el ciclo de trabajo, es decir se realiza la planificación de tareas. La computadora hace ése trabajo por medio del software, el cual es el encargado de procesar los datos recibidos y realizar la comunicación con los microcontroladores que posee cada estación.

La función del programa controlador, es recibir la lectura del objeto en turno, decidir qué lugar ocupara en la mesa de trabajo, y dar la orden al robot para ejecutar los movimientos de piezas.

Para comenzar el ciclo de trabajo primero se asignan las tareas que queremos se realicen en la celda de manufactura, desde la interfaz (controlador).
Una vez asignadas se dará la orden de arranque por medio de una señal, la cual se llevará a cabo mediante un botón de Inicio.
La primera etapa deberá conformarse con el insumo de material, aplicada en la estación 1, por medio de una banda transportadora.
Cuando la banda finalice su proceso enviara un dato al controlador, indicando que la tarea de la estación ha finalizado y la pieza se encuentra en la posición designada.
El controlador mandara la orden al brazo robótico para recoger dicha pieza y una vez tomada la llevara a la siguiente estación para continuar con la siguiente tarea que se haya elegido.

Cuando la pieza ya haya sido colocada en la siguiente fase, el brazo envía la señal y el controlador dará la orden de comenzar la siguiente acción.
Y una vez terminada la estación mandara la señal de finalizado al controlador para que el material sea recogido y continuar el ciclo.
Cada vez que se termine de realizar un proceso de manufacturado en la pieza, la estación en turno mandara un dato al controlador para indicar que ha terminado.
De igual forma cuando el brazo recoja el material de la estación o lo lleve hasta ella, enviara una señal al controlador para indicar que el material está disponible para la siguiente tarea asignada.

Como el proceso en cuestión es el de manufacturar un material, el ciclo de trabajo es infinito hasta que se procesen el número total de piezas que se desea producir.

Durante la ejecución de movimientos por parte del robot, deben administrarse correctamente los intervalos de las órdenes enviadas. Ya que, si se mandan varias órdenes en poco tiempo, el robot no las ejecutará adecuadamente.

Es necesario conceder tiempos de espera entre las órdenes dadas al robot, tomando en cuenta los tiempos de ejecución de éste, para que no se pierdan movimientos. Para ello se cuenta con los agentes inteligentes, que son los encargados de enviar la información de la tarea en ejecución y así poder determinar los movimientos del brazo.

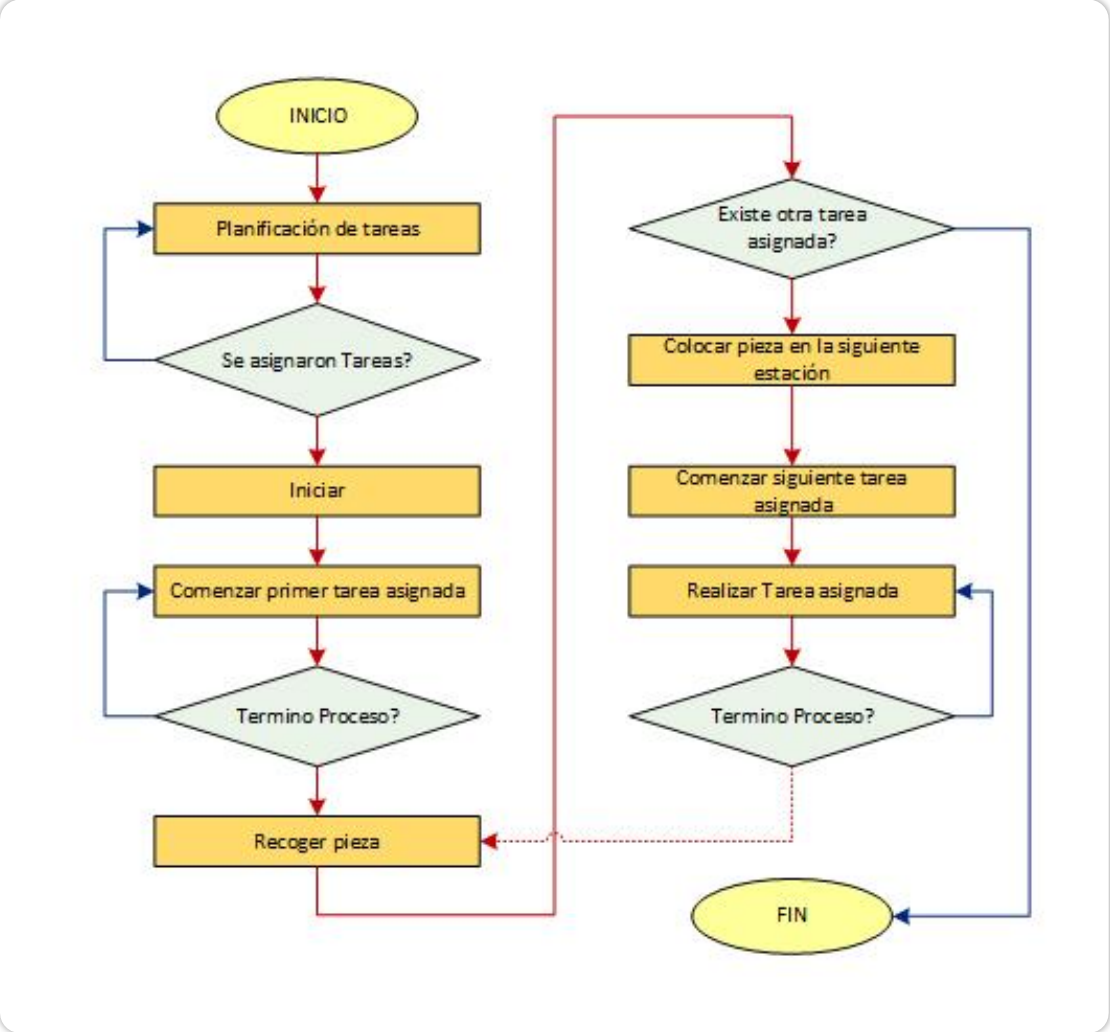


Fig. 5. Diagrama de flujo para un ciclo en Celda de Manufactura.

2.2 Desarrollo del Sistema de Control para planificador de tareas.

De acuerdo a lo establecido en el plan de trabajo los elementos que integrarán la celda estarán conformados como indica la figura 6. El sistema de control desarrollado tendrá en cada estación un microcontrolador para la ejecución tareas con sus respectivos actuadores y módulos Xbee para la comunicación serial. El brazo robótico también contara con un microcontrolador y módulo Xbee para la manipulación del mismo. El microcontrolador es el encargado de enviar los datos y señales indicando que el brazo recogió o dejo el material en la estación indicada, así como las instrucciones para que el brazo se desplace de la manera requerida para alcanzar el material.

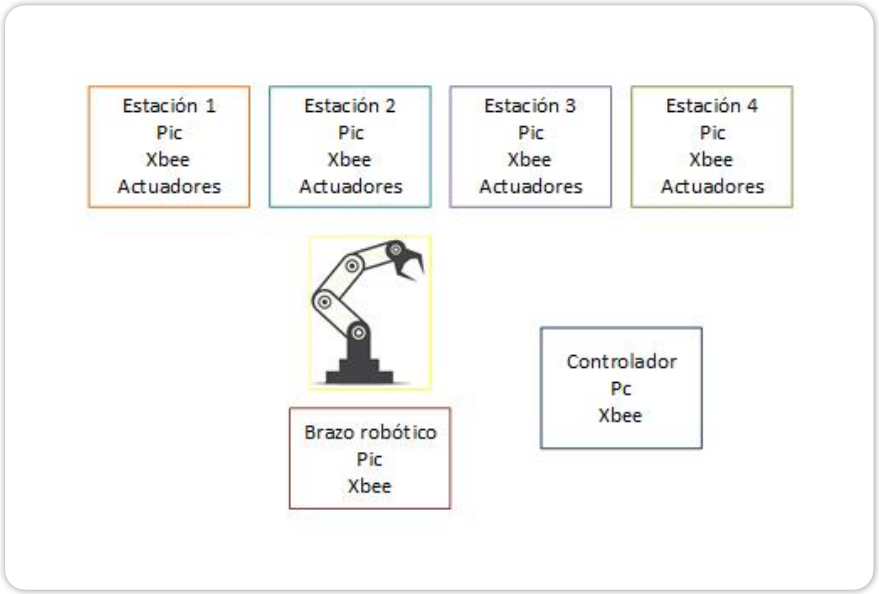


Fig. 6. Componentes para la celda de manufactura.

El proyecto tiene el propósito de integrar en un sistema, elementos para su adquisición de datos y manipulación de actuadores, de acuerdo a la sección anterior, el sistema está compuesto por:

- Computadora de escritorio o laptop
- Microcontroladores
- Unidades Xbee
- Brazo robótico
- Actuadores varios

2.3Integración de infraestructura

La computadora constituye la parte central del proyecto, se encarga de la interfaz de usuario, del procesamiento para coordinar las tareas de la mesa de trabajo, la toma de decisiones con base a los datos adquiridos, y finalmente, de dar las órdenes de movimiento al robot.

La computadora puede determinar los movimientos que el brazo robot debe hacer. La PC manda las órdenes de movimiento vía inalámbrica un dispositivo que se encarga de decodificar la información y entregar niveles lógicos en las entradas digitales del control del brazo.

Dicho dispositivo consiste en un microcontrolador de Microchip *(Anexo 2, Microcontrolador)*, que cuenta con un módulo interno de comunicación y puertos de salida.

La red de comunicación, para el envió y adquirían de datos, se establece de acuerdo a los parámetros permitidos por el brazo robótico Labot Pro Ver5, *(Anexo 2, Brazo robótico)*.

El brazo de robot utilizado para la manipulación de objetos cuenta con una unidad de control. La unidad de control contiene todos los circuitos para el robot, principalmente, es la terminal donde se interpretan los datos para su programación y movimientos.

La topología establecida para la implementación del proyecto será del tipo estrella, *(Anexo 1, topologías de red)*.
En el controlador se colocará el modulo coordinador, que será el encargado de distribuir la información a las diferentes estaciones de trabajo.
Por lo tanto, cada estación de trabajo estará conformada por un módulo end device, quien recibirá y enviará información directamente al coordinador, conectado a la interfaz HMI, que almacena al controlador.

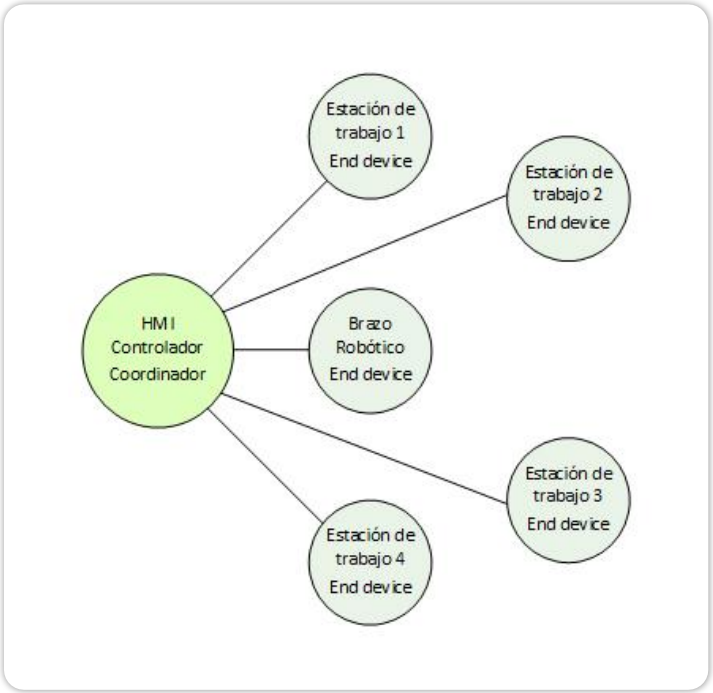


Fig. 7. Topología estrella implementada en el proyecto.

2.4Integracióndel material para estaciones de trabajo

Una vez que se tiene conocimiento de los elementos que formaran parte de la celda de manufactura, se especifican las funciones que debería realizar cada estación de trabajo.

Como ya se mencionó cada estación de trabajo contiene un microcontrolador Pic 16F876A, encargado de dar las instrucciones para llevar a cabo las acciones de la estación y un módulo XBee, en modo End Device, quien recibe y manda los datos vía inalámbrica al Coordinador, que se encuentra conectado al sistema controlador (HMI), y conforme a los datos se ejecute la planificación de la tarea designada.

Área de insumo – Estación de trabajo 1

La primera estación de trabajo es la encargada del insumo del material.
Compuesta por una banda transportadora que maneja diferentes velocidades de traslado, así como el sentido de dirección de la banda.
Un motor de corriente directa hace girar una banda transportadora. El motor es accionado a través de un módulo L298N.Con el cual se puede cambiar el sentido de giro, mediante los pines IN1 e IN2 del módulo del motor. También se modifica la velocidad, para ello se genera una señal PWM en la salida del microcontrolador que se conecta al pin ENA del módulo L298N.
Las salidas a controlar serán cuatro:

1. Regreso lento.
2. Regreso rápido.
3. Avance lento.
4. Avance rápido.

Se colocaron cuatro leds indicadores, cada led corresponde a un tipo de salida y se enciende cuando está en proceso.

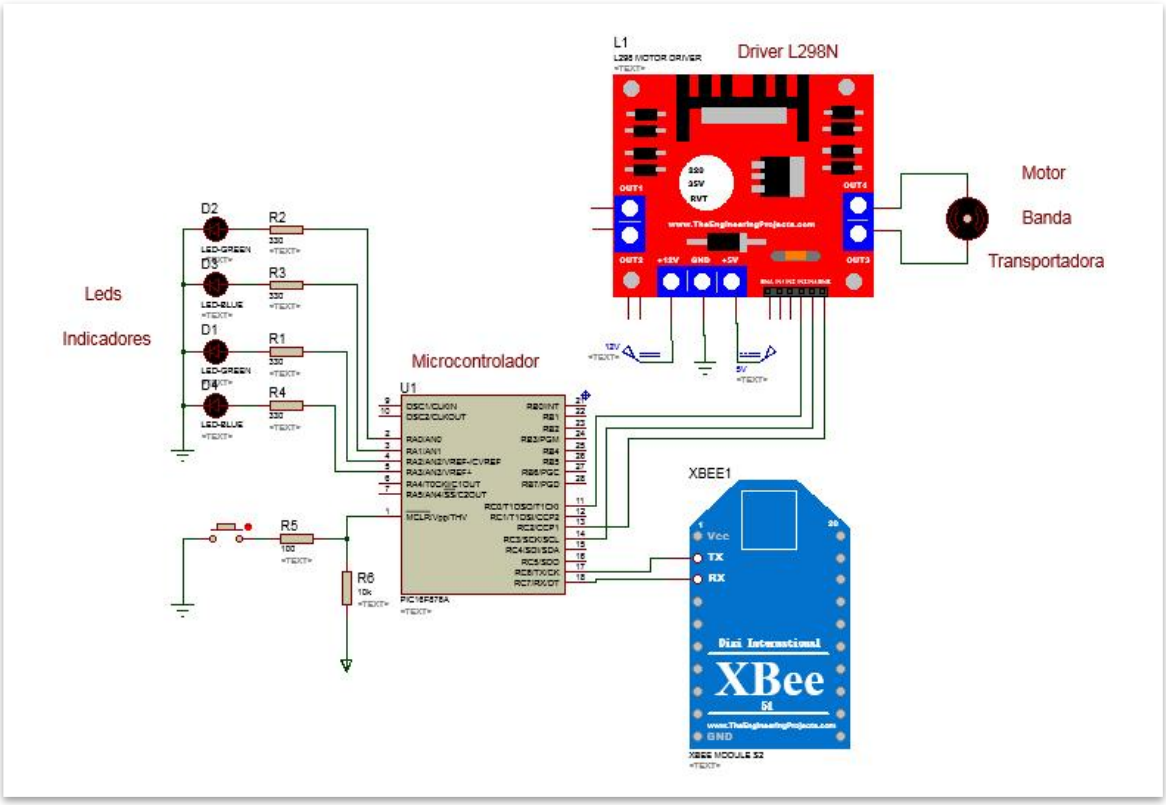


Fig. 8. Diagrama Esquemático estación de trabajo 1.

De este modo los componentes que integraran la estación de trabajo 1, área de insumo, son:

- 1 Microcontrolador Pic 16f876A.
- 1 XBee serie 2.
- 1 Módulo L298N.
- 1 Motorreductor 5V.
- 1 Fuente de alimentación 5V.
- 1 Fuente de alimentación 12V.

- 4 Leds indicadores.

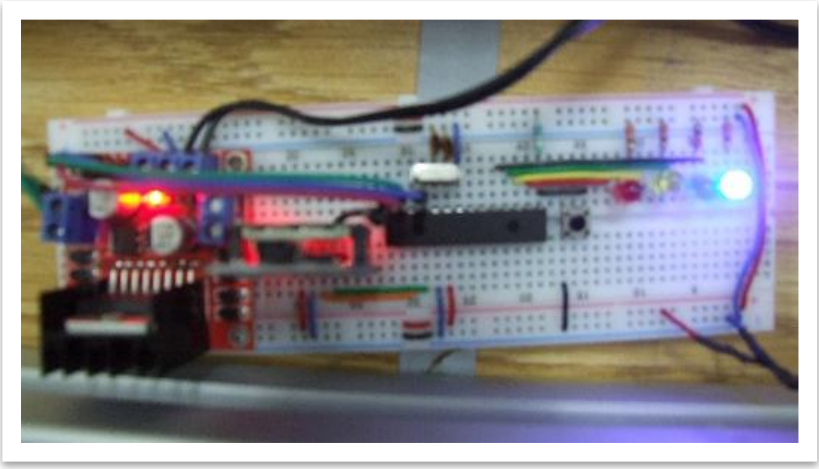


Fig. 9. Estación de trabajo 1, Área de insumo.

Área de barrenado – Estación de trabajo 2

Esta estación simula la acción de un taladro. Es un área donde se pueden fabricar cuatro diferentes tipos de barrenado.
Se manejan dos motores de corriente directa a diferentes velocidades, mediante un driver LN298N. Cada motor representa un tipo de broca, que puede ser fina o gruesa, y se puede aplicar en duro o suave. De este modo en esta estación se aplicarán cuatro diferentes barrenados, es decir tareas:

1. Material Suave – Broca Fina.
2. Material Suave – Broca Gruesa.
3. Material Duro – Broca Fina.
4. Material Duro – Broca Gruesa.

Para visualizar la tarea de barrenado en proceso se coloca un led indicador para cada acción.

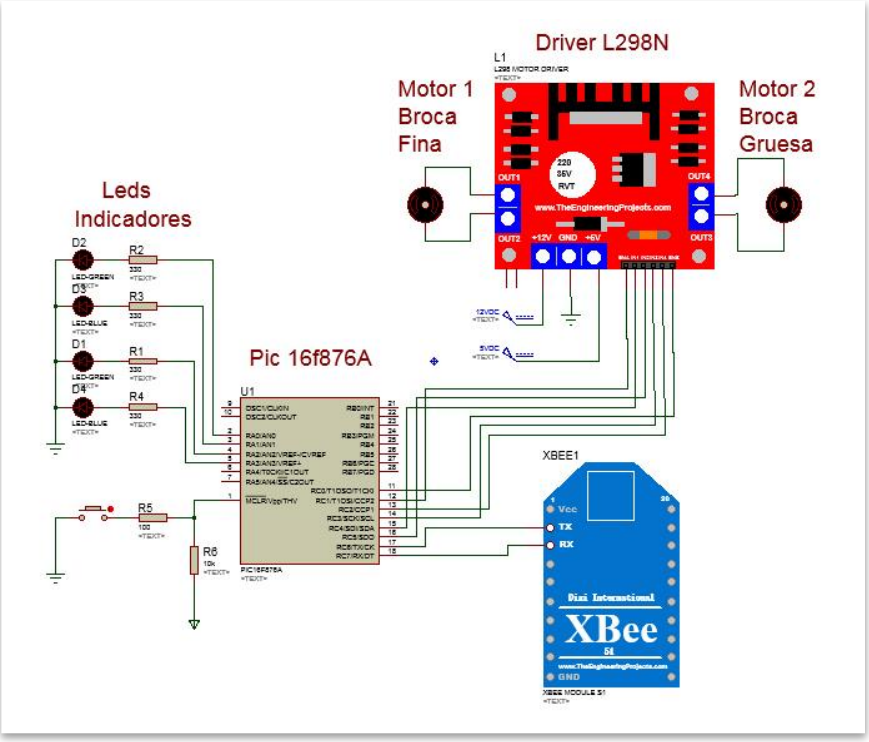


Fig. 10. Diagrama Esquemático estación de trabajo 2.

Los componentes de la estación de trabajo 2 son:

- 1 Microcontrolador Pic 16f876A.
- 1 XBee serie 2.
- 1 Módulo L298N.
- 2 motores 12V DC.
- 1 Fuente de alimentación 5V.
- 1 Fuente de alimentación 12V.
- 4 Leds indicadores.

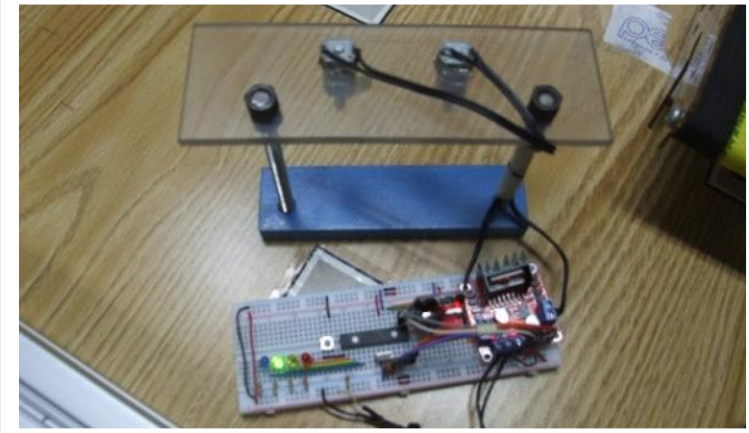


Fig. 11. Estación de trabajo 2, Área de Barrenado.

Área de Limpieza – Estación de trabajo

Esta área hace referencia a una estación donde se puede dar un terminado de limpieza a las piezas que se manufacturan.
Las tareas posibles a realizar son: aplicación de algún removedor o jabón, para facilitar la limpieza, lavado mediante agua y por ultimo un proceso de sacado con ventilación.
Las ejecuciones de estas tareas se pueden implementar utilizando bombas hidráulicas que permitan el transporte del líquido sobre la pieza y un ventilador para la aplicación de aire y secado del material.
Cada dispositivo es activado mediante una señal enviada por el microcontrolador a un relevador que está conectado a cada elemento.
Cuando el relevador es activado, la carga conectada a sus terminales arranca, realizando la tarea que le corresponde.
Las tareas designadas, por lo tanto, quedan de la siguiente manera:

1. Remover
2. Enjuagar
3. Secar

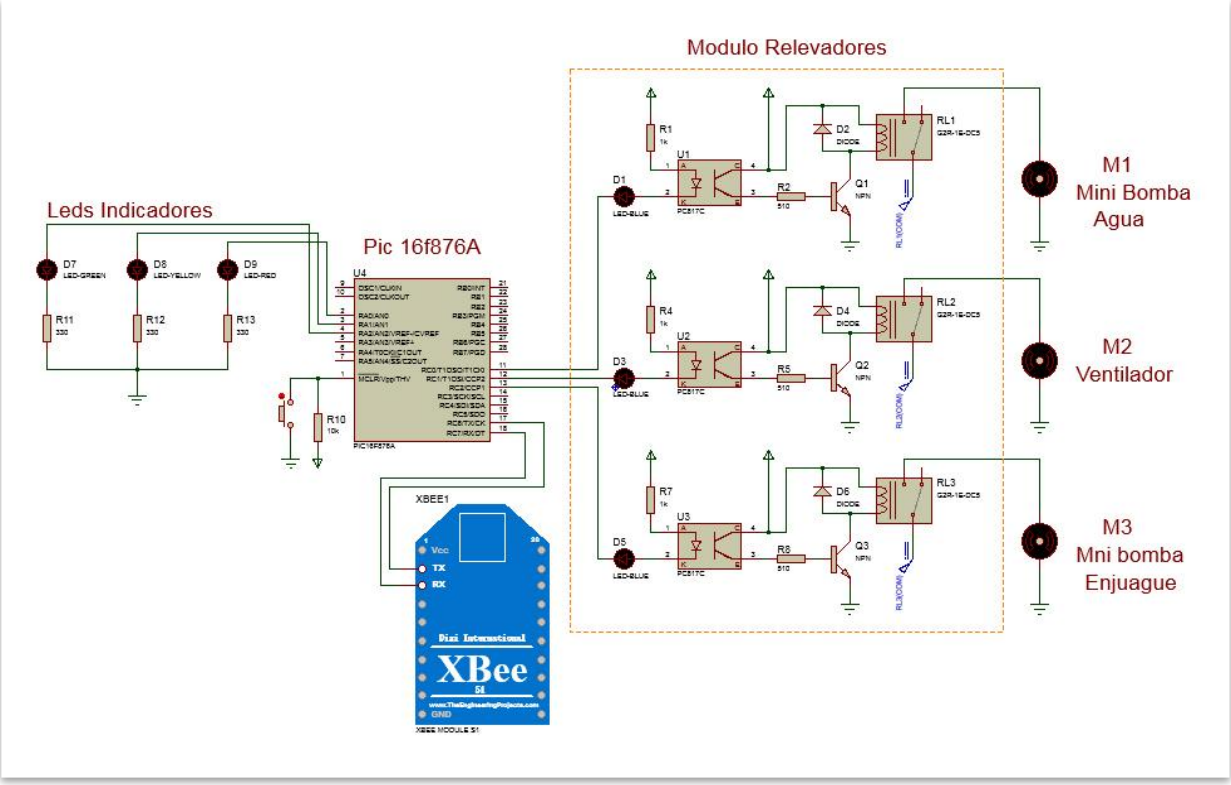


Fig. 12. Diagrama Esquemático estación de trabajo 3.

Los componentes que integran la estación son:

- 1 Microcontrolador Pic 16f876A.
- 1 XBee serie 2.
- 1 Módulo cuatro relevadores.
- 2 mini bombas hidráulicas 12V.
- 1 Ventilador 12V.
- 2 Fuentes de alimentación 5V.
- 1 Fuente de alimentación 12V.
- 4 Leds indicadores

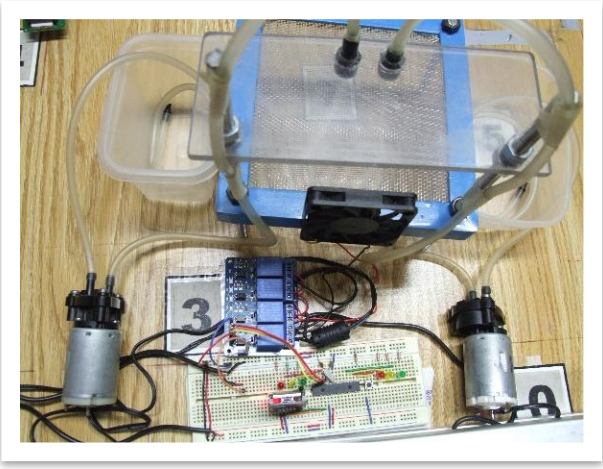


Fig. 13. Estación de trabajo 3, Área de Limpieza.

Área de CNC – Estación de trabajo 4

Esta área hace referencia a un centro de mecanizado, donde se podrían realizar diferentes acabados y formas a las piezas. Área donde se simula que se elabora un terminado al objeto, y se representa con un mensaje mostrado en un Display, así como un led intermitente para cada acción, que simboliza el tiempo de duración de la tarea.
Son cuatro las formas o figuras posibles que se pueden realizar. Al microcontrolador se le asigna un carácter especial para cada tarea y con ello diferenciar la acción que debe realizar según ordene el controlador.

1. Cuadrado.
2. Triángulo.
3. Rectángulo.
4. Pentágono.

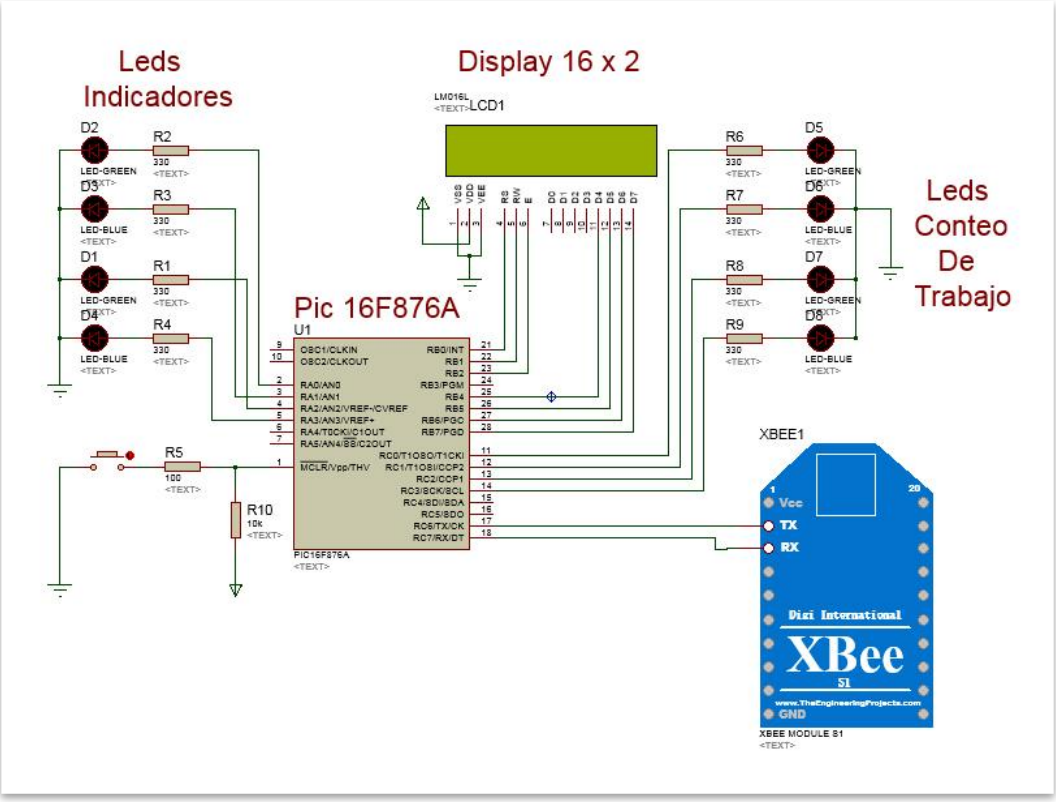


Fig. 14. Diagrama Esquemático estación de trabajo 3.

Los componentes que integran la estación son:

- 1 Microcontrolador Pic 16f876A.
- 1 XBee serie 2.
- 1 Display LCD 16 x 2.
- 1 Fuente de alimentación 5V.
- 8 Leds indicadores

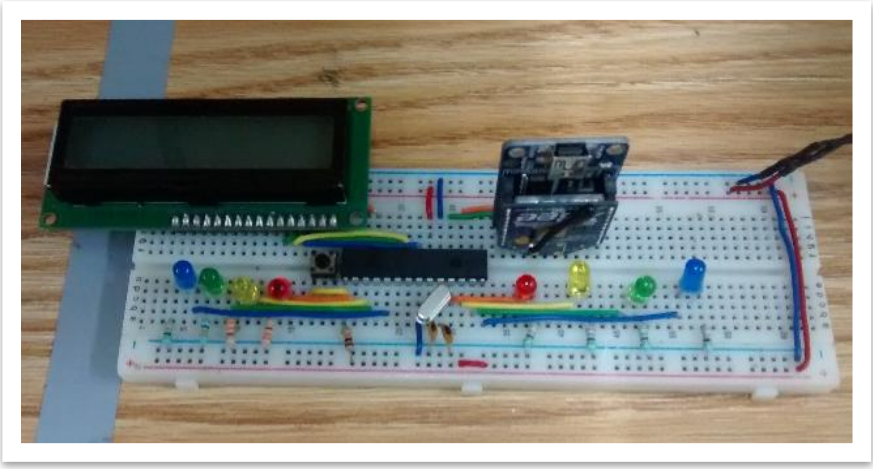


Fig. 15. Estación de trabajo 4, Área de CNC.

Área de Brazo

Esta estación es la encargada de recibir y enviar los datos para ejecutar los movimientos que el brazo robótico necesita y activar los motores correspondientes.
Los datos son recibidos por medio del Xbee que está conectado al microcontrolador de la estación y mediante el conector DB9 se envían los datos a la placa del Brazo robótico para mover los motores.

Cuando el brazo finaliza su ciclo de movimiento el Pic envía la señal al sistema controlador para ejecutar la siguiente tarea.

Las tareas que se efectúan en la estación serian diez:

1. Mover el brazo a posición inicial con pinza abierta.
2. Mover el brazo a posición inicial con pinza cerrada.
3. Mover el brazo a la estación 1 para sujetar pieza.
4. Mover el brazo a la estación 1 para dejar pieza.
5. Mover el brazo a la estación 2 para sujetar pieza.
6. Mover el brazo a la estación 2 para dejar pieza.
7. Mover el brazo a la estación 3 para sujetar pieza.
8. Mover el brazo a la estación 3 para dejar pieza.
9. Mover el brazo a la estación 4 para sujetar pieza.
10. Mover el brazo a la estación 4 para dejar pieza.

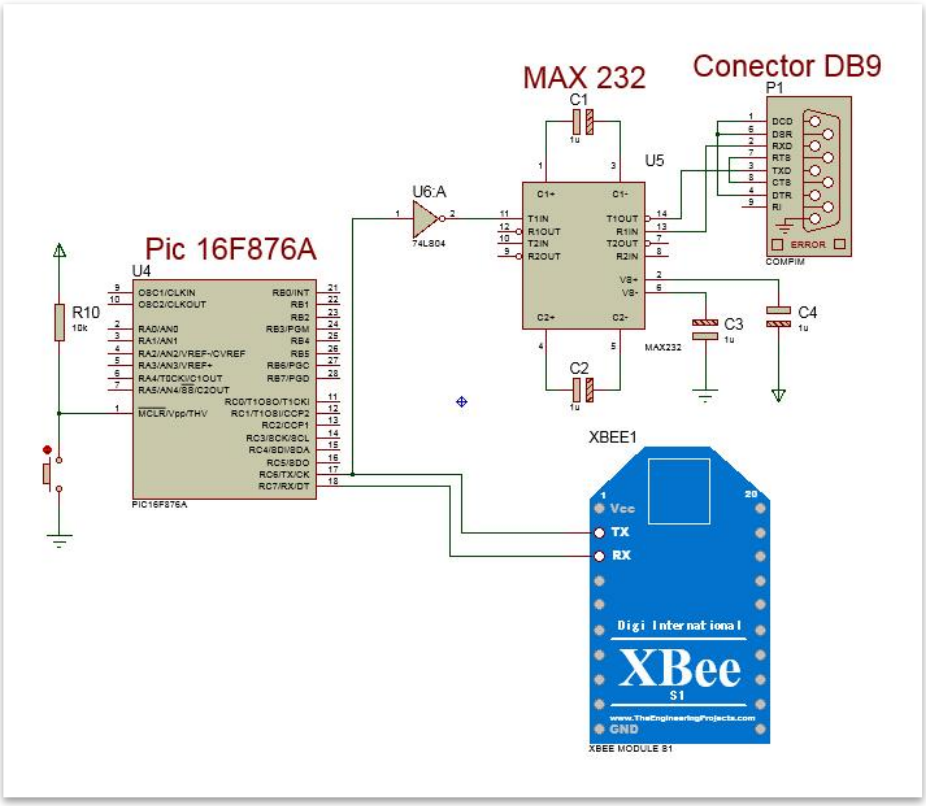


Fig. 17. Diagrama Esquemático estación de trabajo 5.

Los elementos que integran la estación son:

- Microcontrolador Pic 16f876A.
- XBee serie 2.
- Módulo Max232.
- Conector DB9.
- Fuente de alimentación 5V.

Fig. 16. Estación de Brazo robótico.

2.19 Asignación de caracteres en el microcontrolador para cada tarea

Serán 15 acciones o tareas las que se podrán seleccionar en el planificador. A cada tarea se le asignó un carácter único, esto le permite al sistema diferenciar cada una de ellas. Sin embargo, las tareas asignadas corresponden a 33, ya que para ejecutar la planificación se deben ejecutar diferentes procesos.

Para que la tarea seleccionada ejecute la acción correspondiente, el controlador envía un dato vía inalámbrica por medio del Xbee que contiene. El dato corresponde al carácter que se le designo a la acción en el microcontrolador.

El sistema controlador envía un dato a todas las estaciones de trabajo y si coincide con un valor preprogramado en el Pic, se ejecutará el proceso indicado, de no ser así la estación de trabajo se mantendrá detenida.

En la tabla 13 se muestran los caracteres asignados a cada tarea, así como la acción que ejecuta si es recibida por el microcontrolador en la estación.

Se muestra también las acciones designadas a la estación del Brazo robótico. Cada carácter corresponde a la ejecución de movimiento de los motores del brazo para desplazarlo al lugar indicado.

Estación de trabajo	Tareas	*No. Tarea*	*Carácter*	*Tareas asignadas al carácter*
Banda transportadora	***Sentido***	1	***a***	Apagada
	Avanzar	2	***b***	Banda avanza lenta
	Regresar	3	***c***	Banda avanza rápido
	Velocidad	4	***d***	Banda regresa lento
	Lento	5	***e***	Banda regresa rápido
	Rápido			
Barrenar	***Material***	6	***f***	Apagada
	Suave	7	***g***	Barrenar material suave y broca fina
	Duro	8	***h***	Barrenar material suave y broca gruesa
	Broca	9	***i***	Barrenar material duro y broca fina
	FINO	10	***j***	Barrenar material duro y broca gruesa
	Grueso			
Limpieza	Remover	11	***0***	Nada
	Enjuagar	12	***1***	Limpieza con enjuague
	Secar	13	***3***	limpieza con removedor
		14	***5***	Limpieza secando
		15	***4***	Limpieza con enjuague y removedor
		16	***6***	Limpieza con enjuague y secado
		17	***8***	Limpieza con removedor y secado
		18	***9***	Limpieza con enjuague, removedor y secado
CNC		19	***k***	Nada
	Uno	20	***l***	Figura 1 Cuadrado
	Dos	21	***m***	Figura 2 Triangulo
	Tres	22	***n***	Figura 3 Rectángulo
	Cuatro	23	***o***	Figura 4 Pentágono
Brazo	Pinza abierta	24	***p***	Posición inicial abierta
	Pinza cerrada	25	***q***	Posición inicial cerrada
	Estación 1	26	***a***	Mover a Estación 1 para agarrar
	Estación 1	27	***B,C,D,E***	Mover a Estación 1 para soltar
	Estación 2	28	***F***	Mover a Estación 2 para agarrar
	Estación 2	29	***G,H,I,J***	Mover a Estación 2 para soltar
	Estación 3	30	***0***	Mover a Estación 3 para agarrar
	Estación 3	31	***R,S,T,U,V,W,X***	Mover a Estación 3 para soltar
	Estación 4	32	***K***	Mover a Estación 4 para agarrar
	Estación 4	33	***L,M,N,O***	Mover a Estación 4 para soltar

Tabla 1. Tareas y Caracteres asignados.

Capítulo 3

Implementación software controlador

3.1 Interfaz HMI

El sistema controlador se realizó mediante el software LABVIEW, *(anexo 1)*.
En la pantalla de interacción con el usuario se muestran cada una de las estaciones de trabajo, así como el brazo robótico, los botones de control para cada tarea, un botón para iniciar el proceso, una ventana o recuadro de las tareas planificas, un recuadro para selección del puerto y por ultimo un botón de paro.

Fig. 18. Interfaz HMI implementada, Sistema Controlador.

Se debe seleccionar el puerto en el que se encuentra conectado el módulo XBee (coordinador), encargado de la adquisición y envió de datos, a los dispositivos ubicados en las estaciones.
Para seleccionar el puerto correspondiente se dispone de un cuadro de lista, con el nombre “Puerto”, donde al momento de presionar la flecha despliega el listado de los puertos que se encuentran disponibles, se elige el perteneciente al módulo coordinador.

Fig. 19. Cuadro de lista para selección de puerto.

En la parte derecha del sistema controlador se muestra el recuadro “Planificar Tarea”, donde están colocados los botones correspondientes a cada tarea que se puede asignar en la celda. Los botones hacen referencia a las acciones que se realizan en las estaciones de trabajo, que en total son quince.
Para agregar las tareas que se desean realizar en la celda, solo basta con presionar dicho botón. El botón seleccionado cambia de color, lo que ayuda a visualizar que dicho botón fue presionado.

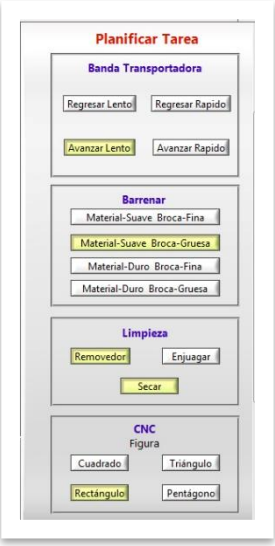


Fig. 20. Botones de tareas, en color amarillo se muestra los botones presionados.

En una parte del Sistema Controlador se tiene una pequeña ventana con el nombre “Tarea Planificada”, donde se visualizan las acciones elegidas. Esta ventana permite ver el orden de las acciones que serán aplicadas a las piezas en la celda de manufactura.

Fig. 21. Ventana de tarea planificada.

La interfaz HMI cuenta con dos botones que son para iniciar o detener el proceso en marcha.
Una vez definidas las tareas en el planificador y seleccionado el puerto donde se encuentra el coordinador, se puede dar marcha al proceso.
Par arrancar el proceso solo se presiona el botón “INICIAR”, y en ese momento comenzaran a ejecutarse las tareas, en el orden mostrado en la ventana de “Tarea Planificada”.

Fig. 22. Botón Inicio de proceso.

Si se desea detener por algún motivo el proceso se presiona el botón “PARO”. En ese momento la celda de manufactura dejara de ejecutar el ciclo en cuestión.

Fig. 23. Botón Paro de proceso.

3.2 Diagrama de bloques del Sistema controlador

Dentro del diagrama de bloques del sistema controlador, se encuentran todas las funciones e instrucciones para que el funcionamiento del sistema se ejecute correctamente.

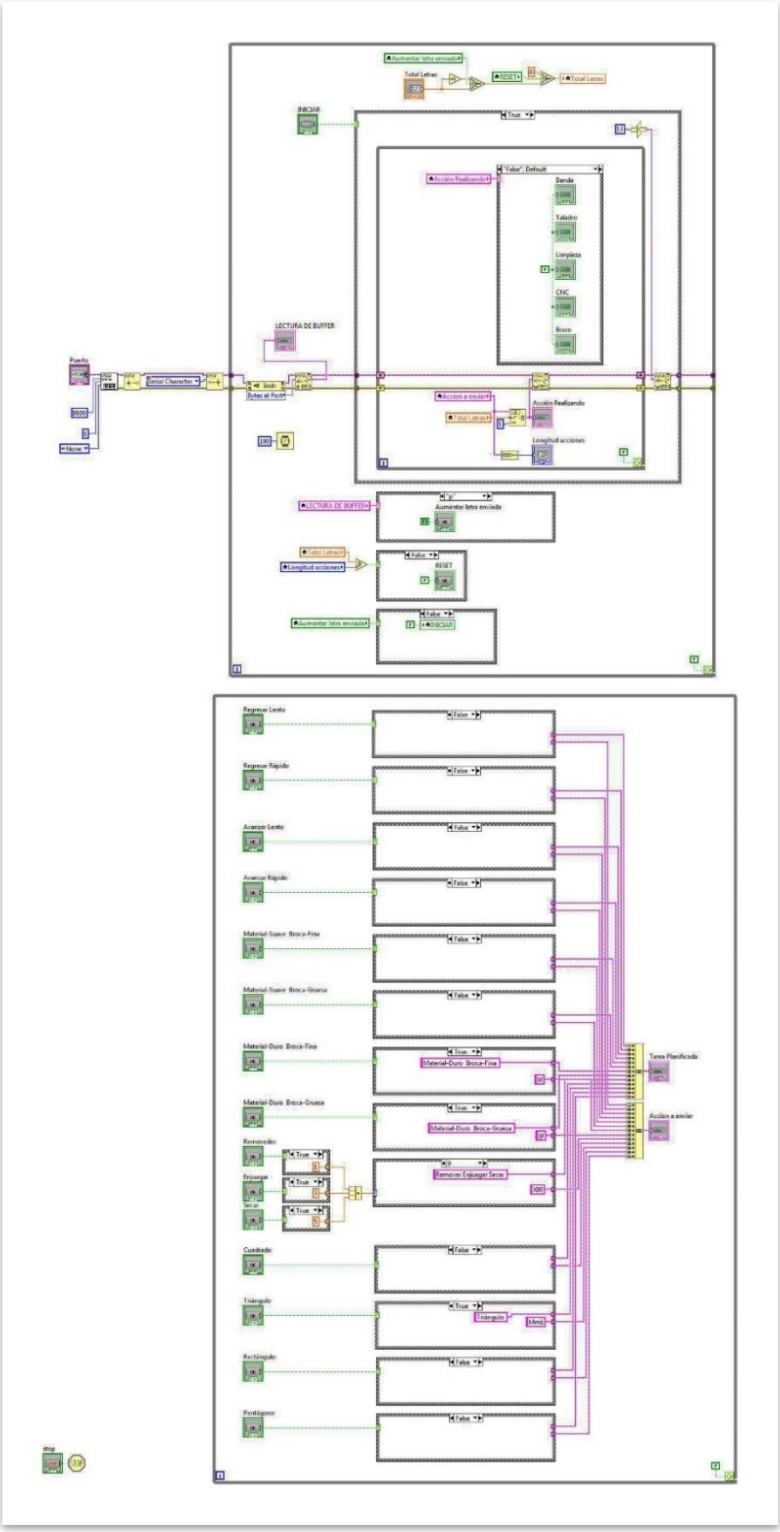


Fig. 24. Diagrama de bloques Sistema Controlador.

3.3 Funcionamiento del Sistema controlador

Para que el sistema controlador realice las funciones mencionadas se agregaron componentes que no se muestran en la interfaz HMI o panel frontal, pero que son requeridos para el funcionamiento del sistema controlador.

Fig. 25. Elementos adicionales en el panel frontal.

Se colocaron tres controles de lectura de cadenas de texto.
En el primero se visualizan los datos que son recibidos y enviados por el puerto serial, lleva por nombre “Lectura Buffer”.

Fig. 26. Control de cadena, “Lectura Buffer”.

El segundo cuadro muestra el carácter que es enviado por el puerto serial y que corresponde a una de las posibles tareas que fueron planificadas. Este cuadro tiene situado el nombre “Carácter tarea realizando”.

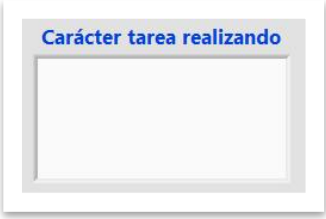


Fig. 27. Control de cadena, “Carácter tarea realizando”.

Por último en el tercer cuadro se aprecian todos los caracteres que deberán ser enviados por el puerto serial a las estaciones de trabajo y realizar la rutina de tareas planeada, se conoce como “Caracteres a enviar”.

Fig. 28. Control de cadena, “Caracteres a enviar”.

Se agregó un indicador numérico, llamado “Longitud acciones”. Este control muestra el número total de caracteres para la rutina planteada, está definido por la agrupación de los caracteres de las tareas elegidas, el cual hace referencia al número total de caracteres visualizados en el cuadro “Caracteres a enviar”.

Fig. 29. Indicador numérico, “Longitud acciones”.

El control numérico llamado “Total letras”, se utiliza para definir el carácter que debe ser enviado, por el controlador a través del puerto. De acuerdo al número que contenga el control, se envía el dato contenido en el cuadro “Caracteres a enviar”.

Fig. 30. Control numérico, “Total letras”.

El primer carácter visualizado en el control “Caracteres a enviar” corresponde al valor cero (0), el segundo carácter a la posición uno (1), el tercer carácter a la posición dos (2) y así sucesivamente.

Fig. 31. Ejemplo de posición de caracteres en cuadro “Caracteres a enviar”.

También se colocó un botón controlador llamado “Aumentar letra enviada”. Cuando este botón es activado, es decir su estado se encuentra en verdadero (True), se suma una unidad al número que se encuentra en el control numérico “Total letras”. Esto permite que el sistema controlador pueda enviar el siguiente carácter mostrado en cuadro “Caracteres a enviar”.

Fig. 32. Botón de control “Aumentar letra enviada”.

Por último, se colocó un botón controlador nombrado “Reset”. Cuando este botón está activado (T), el control numérico “Total letras” toma el valor cero (0).

Fig. 33. Botón de control “Reset”.

3.4 Análisis del diagrama de bloques del sistema controlador

El sistema controlador está compuesto por dos estructuras del tipo while. Una estructura para el sistema de comunicación, lectura y envió de datos, y otra estructura que maneja la selección de las tareas. En una estructura while se encuentran 16 estructuras del tipo case, las cuales pertenecen a las acciones que realizan los botones de control, que están destinados a la planificación de la rutina.

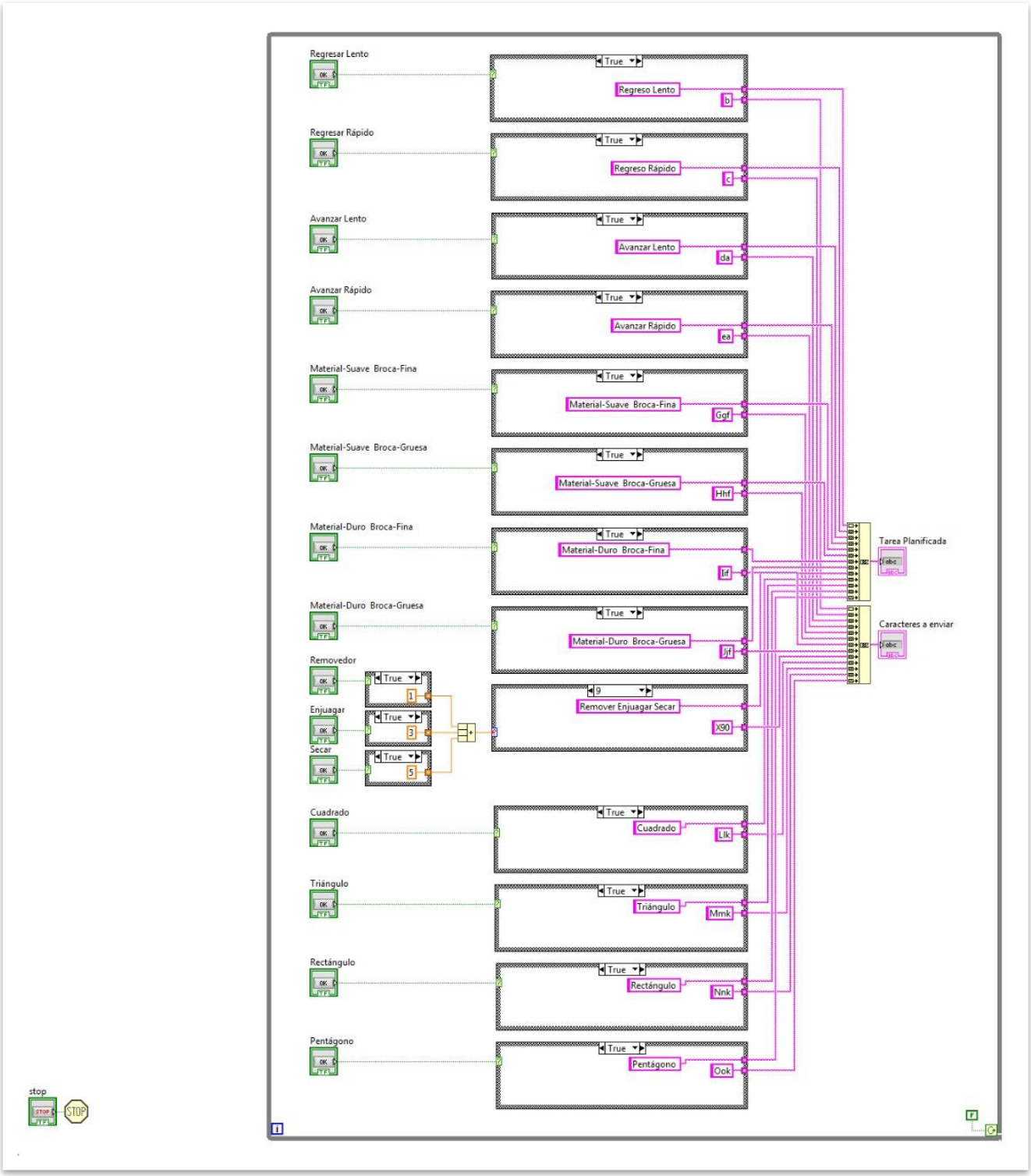


Fig. 34. Estructura while para la función de los botones de tareas.

Cada botón de control está unido a una estructura case, y dependiendo del estado en que se encuentre el botón, verdadero (T-true) o falso (F-false), realiza las acciones que están dentro de la estructura case.

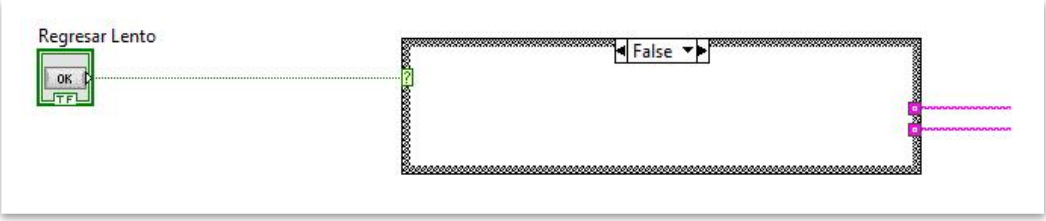


Fig. 35. Botón “Regresar lento”, desactivado, estructura case en “False”.

Si el botón de control se encuentra desactivado, no se realiza ninguna tarea en la estructura case.

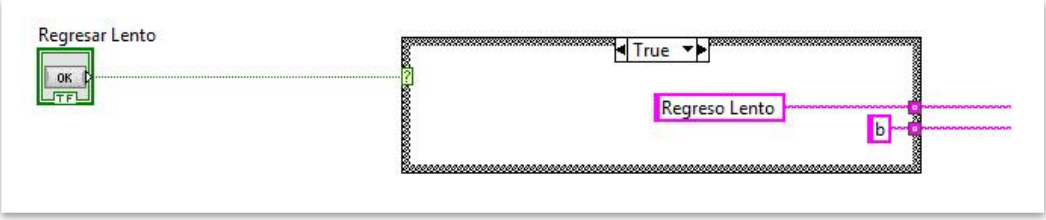


Fig. 36. Botón “Regresar lento”, activado, estructura case en “True”.

Cuando el botón de control se encuentra en estado activo, se realizan las operaciones dentro de la estructura case.
Se puede apreciar que la estructura case, cuando la etiqueta de selección es verdadera (True), contiene dos cadenas constantes de texto.
La primera cadena hace referencia a la tarea asignada al botón de control, en este caso “Regresar lento”.
Y la segunda cadena se refiere a los caracteres que serán enviados por el controlador, mediante el coordinador por el puerto serial, a las estaciones de trabajo. En este caso solo contiene el carácter “b”, utilizando la tabla 13, se puede apreciar que se refiere a la tarea “Banda avanza lenta”, de la estación 1.

A excepción de Case perteneciente a la estación de limpieza, todos los Case encargados de los mensajes pertenecientes a las tareas, contienen lo mismo, la parte “False” se encuentra vacía, mientras que para la etiqueta “True” se localizan dos constantes de cadenas.
La diferencia está en el mensaje escrito en cada constante de cadena.
Para cada botón la primera constante contiene escrita la tarea a la que hace referencia, y en la segunda constante están los caracteres que debe enviar según la acción de la que se trate.

En el diagrama los botones de control “Removedor”, “Enjuagar” y “Secar”, pasan primero por una selección.
Cada botón se encuentra anidado a una estructura case, y la salida de la estructura se conecta con la herramienta de sumatoria de valores numéricos. La salida de la herramienta de suma está unida a una estructura case.

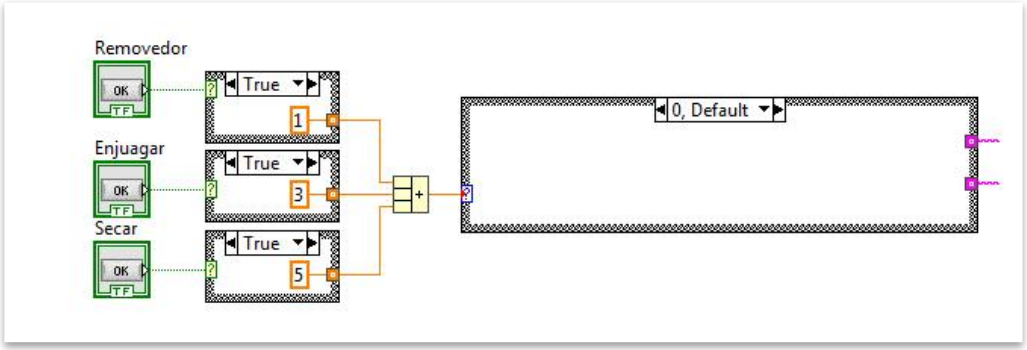


Fig. 37. Botones de control para la estación de limpieza.

En la estructura case unida al botón, si este se encuentra desactivado, no se realiza proceso alguno.

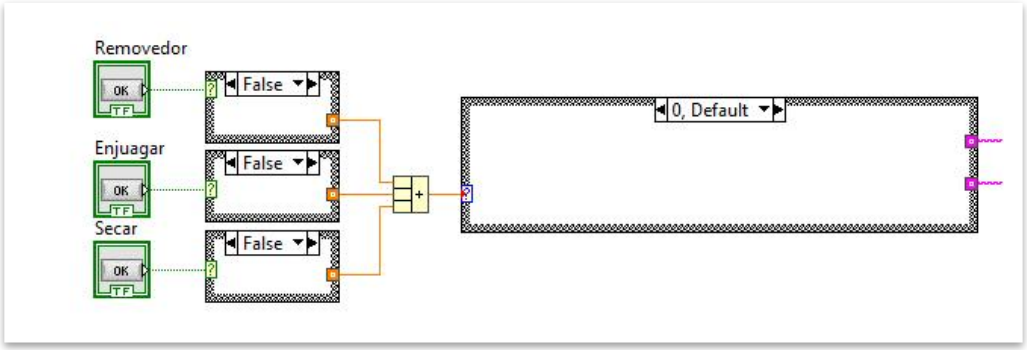


Fig. 38. Botones de control desactivados, estructuras case “False”.

Mientras que, si se encuentran activados, la estructura case toma el valor verdadero (True) y manda una constate numérica a la herramienta de sumatoria.

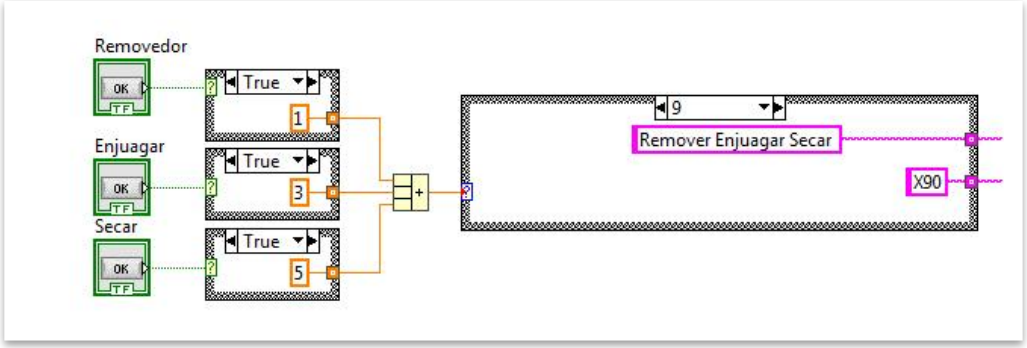


Fig. 39. Botones de control activados, estructuras case “True”.

Cada botón tiene asignado una constante numérica.
El botón “Removedor” tiene el valor uno (1), el botón “Enjuagar” el valor tres (3) y por último el botón “Secar” el valor cinco (5).
En la entrada de la herramienta de suma llegan estos valores para ser procesados. A la salida de la herramienta se encuentra una estructura case.
Se puede apreciar que en la estructura case, la etiqueta de selección equivale al valor de la sumatoria, donde los valores posibles son: 0, 1, 3, 5, 4, 6, 8, o 9.
El case ejecutara las acciones correspondientes al resultado de la suma.
Si el valor es cero, la estructura case no realiza tarea alguna.

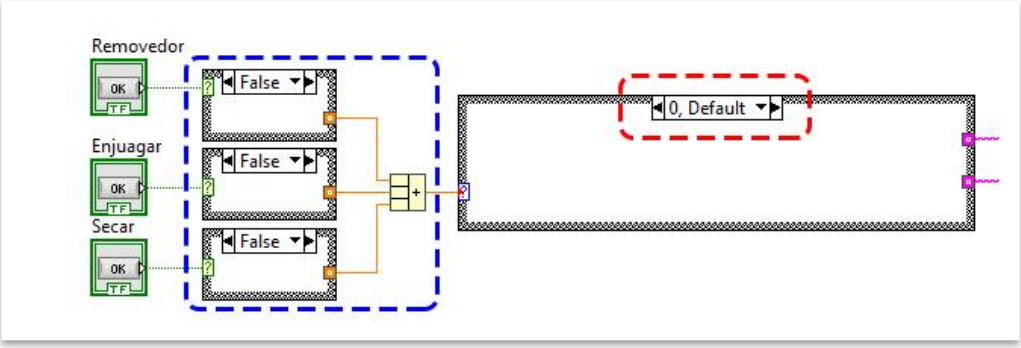


Fig. 40. Suma igual a cero, etiqueta de la estructura case “0”, no se realizan tareas.

Sin en cambio, si el valor de la suma es diferente de cero, se ejecuta el case que contenga el valor del resultado de la suma, el cual contiene dos cadenas constantes de texto.

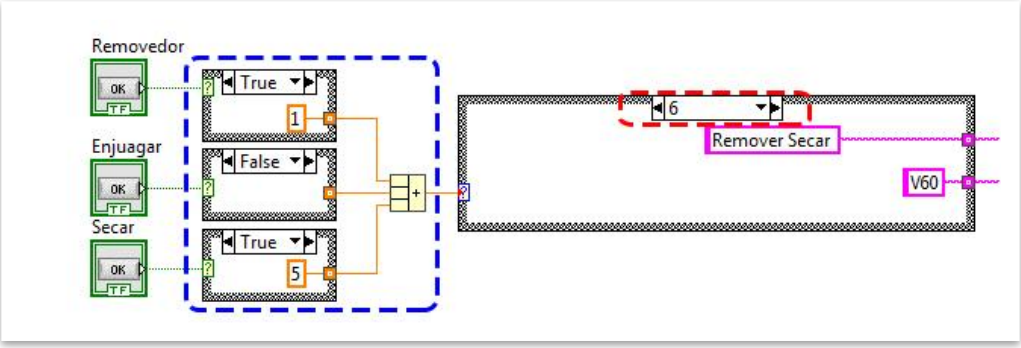


Fig. 41. Suma igual a seis, etiqueta de la estructura case “6”.

La primera cadena hace referencia a la tarea asignada por los botones de control, del área de limpieza presionados, en este caso “Removedor” y “Secar”, por lo tanto, muestra el mensaje “Remover Secar”.
Y la segunda cadena se refiere a los caracteres que serán enviados por el controlador, mediante el coordinador por el puerto serial, a las estaciones de trabajo. En este caso los caracteres “V60”, basados en la tabla 13, las tareas asignadas son Mover a Estación 3 para soltar (Estación Brazo), Limpieza con enjuague y secado (Estación Limpieza), Mover a Estación 3 para agarrar (Estación Brazo) y Nada (Estación de limpieza).

Los mensajes que salen de las estructuras cases van dirigidos hacia una herramienta de concatenación, que se encarga de unir varios textos y mostrarlos en una sola salida.

Los mensajes que contienen el texto de la tarea se conectan hacia las entradas de la primera herramienta de concatenación. Los mensajes unidos se muestran en el indicador de cadena “Tarea planificada”.

Los mensajes que contienen los caracteres para ser enviados, se agrupan con la segunda herramienta de concatenación y se visualizan en el indicador de cadena “Caracteres a enviar”.

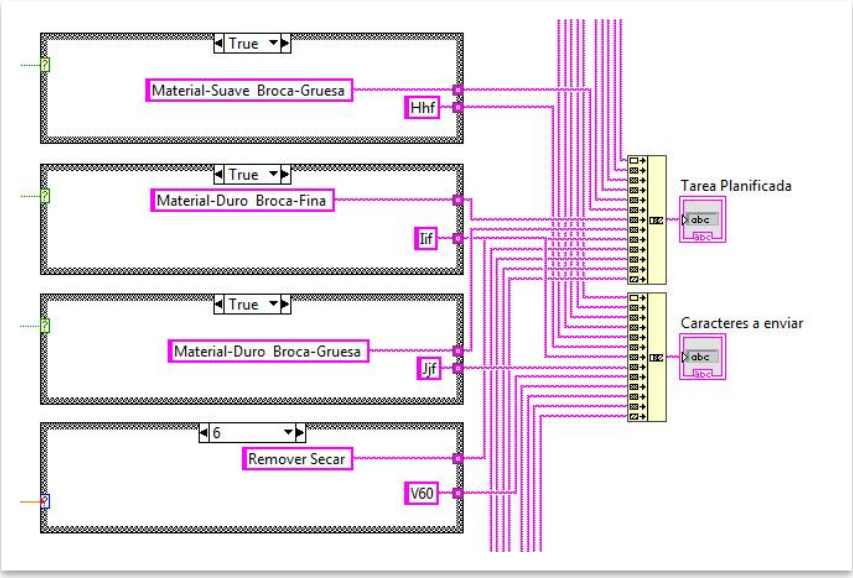


Fig. 42. Mensajes conectados a herramienta de concatenación con sus indicadores.

En la otra estructura while, se encuentran cuatro estructuras case, donde se aprecia que una de ellas a su vez anida otra estructura while y dentro de este se encuentra una estructura case.
En esta parte del diagrama de bloques, se realizan las operaciones para la comunicación, controlar el dato que debe ser enviado para realizar la tarea correspondiente, así como leer los caracteres mandados por las estaciones de trabajo al sistema controlador.

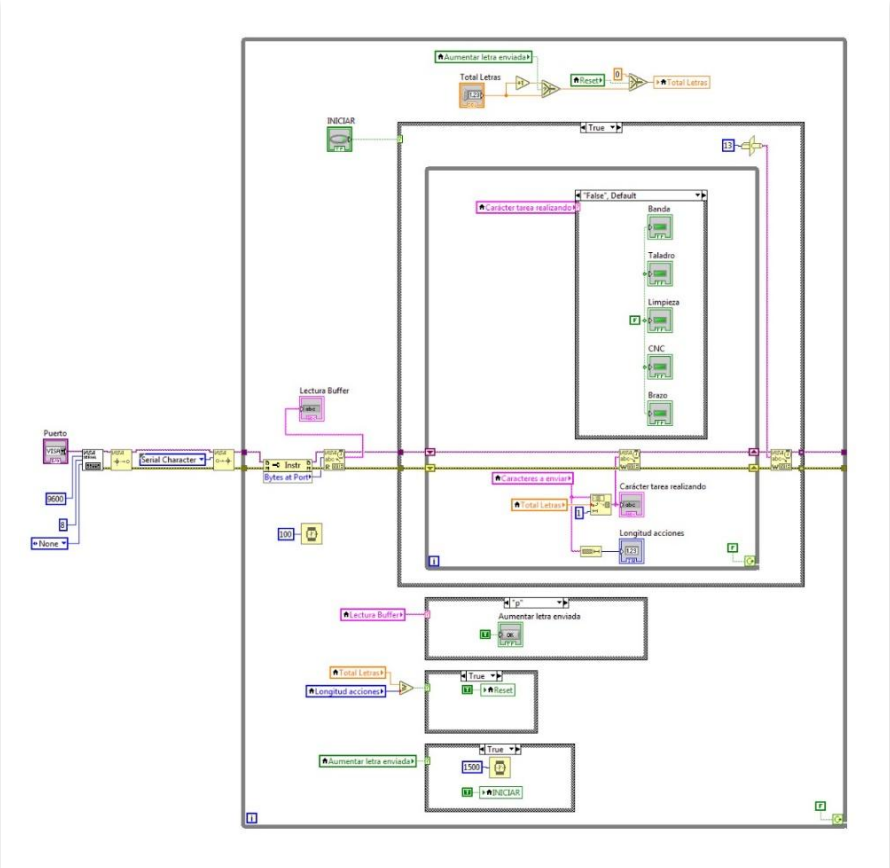


Fig. 43. Estructura while para comunicación y control de datos.

La primera parte del código implica la configuración del puerto vía serial, utilizando las herramientas de comunicación NI-VISA. Las cuales se muestran del lado derecho de la estructura.

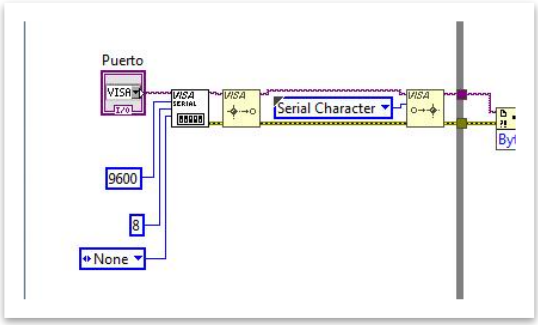


Fig. 44. Herramientas NI-VISA para comunicación serial.

Esta parte del código permite el inicio de sesión VISA, con la que se tiene acceso al puerto serial. Y se logra usando la herramienta “VISA configure port”. Aquí se configuran los parámetros de conexión, el puerto de conexión, la velocidad de baudios, el número de bits y la paridad.

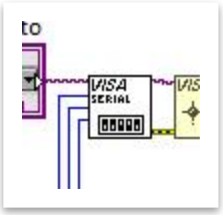


Fig. 45. Herramientas VISA configure port.

Para elegir puerto de conexión se utiliza una caja de control, donde aparecen los puertos que están siendo utilizados por la computadora, entre los cuales debe estar el puerto del módulo XBee Coordinador. Se le asignó el nombre “Puerto”, para reconocer a que hace referencia.

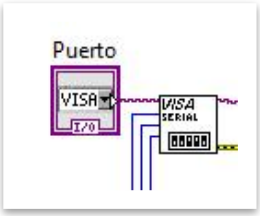


Fig. 46. Caja de control para seleccionar puerto serial.

Los demás parámetros son designados como constantes:

- La velocidad es de 9600 baudios.
- El número de bits de 8.
- No existe paridad – None.

Los datos de configuración se basan de acuerdo a los parámetros permitidos para el uso del brazo robótico, Labot Pro Ver5 *(Anexo 2, brazo robótico)*.

Estos datos deben corresponder con la configuración de trabajo tanto del brazo robótico, como de los módulos Xbee, para la red estrella implementada.

Ya establecidos los parámetros de la configuración se especifica el tipo de evento que se trabajaran en el puerto, en este caso del tipo carácter.

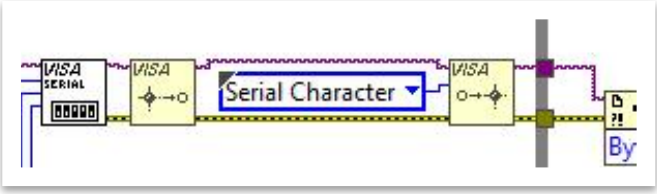


Fig. 47. Tipo de evento en la sesión VISA, serial character.

Se puede ver que una vez inicializada la sesión VISA, esta va conecta a una estructura while. Dentro de ella se tienen cuatro estructuras case.

En esta estructura while, se configurar la herramienta “VISA Read”, la cual se encarga de leer los datos que se encuentran en el buffer.

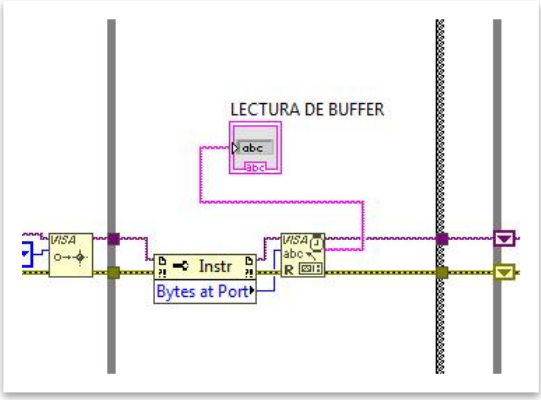


Fig. 48. Conexión VISA Read.

En la entrada de la herramienta de lectura se agrega un nodo “byte count”, el cual recibe el tamaño de buffer que se escribió en el puerto.

A la salida del VISA Read se conecta un control de cadena (String control), que hace referencia al indicador de cadena “Lectura de buffer” y en cual se van mostrando los datos que son adquiridos y enviados en el puerto serial, en este caso, los datos enviados por la interfaz y los datos recibidos, que envían las estaciones de trabajo.

El Visa Read está conectado a una estructura case que es controlado por el botón “Iniciar”. Si el botón se encuentra desactivado el caso es falso (False) y no lleva a cabo acción alguna.

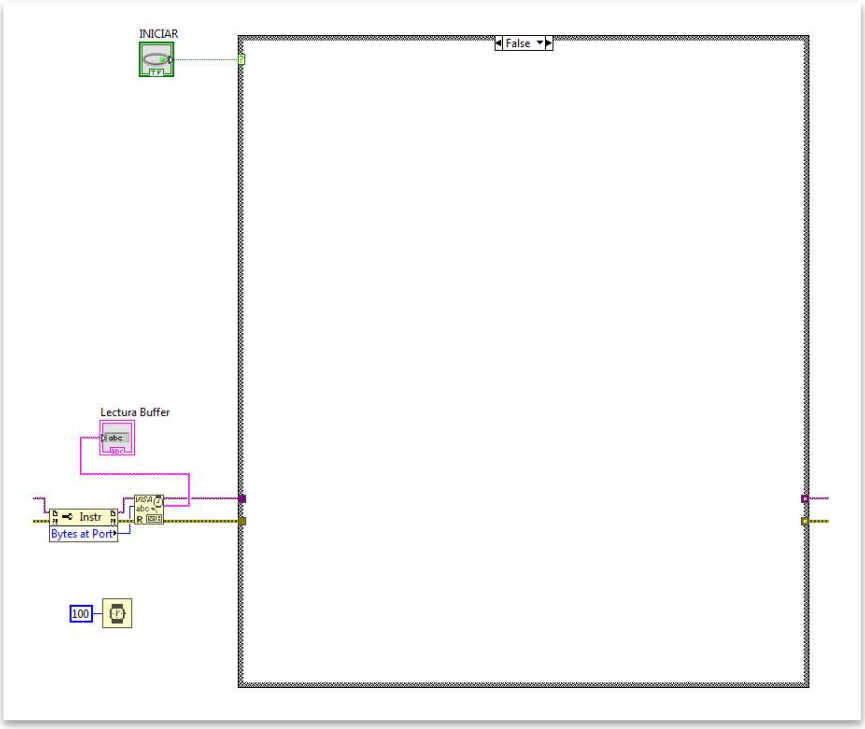


Fig. 49. Estructura case desactivada (False), controlada por el botón “Iniciar”.

De lo contrario si el botón está activado el caso es verdadero (True) y se ejecutan las acciones dentro de la estructura case.

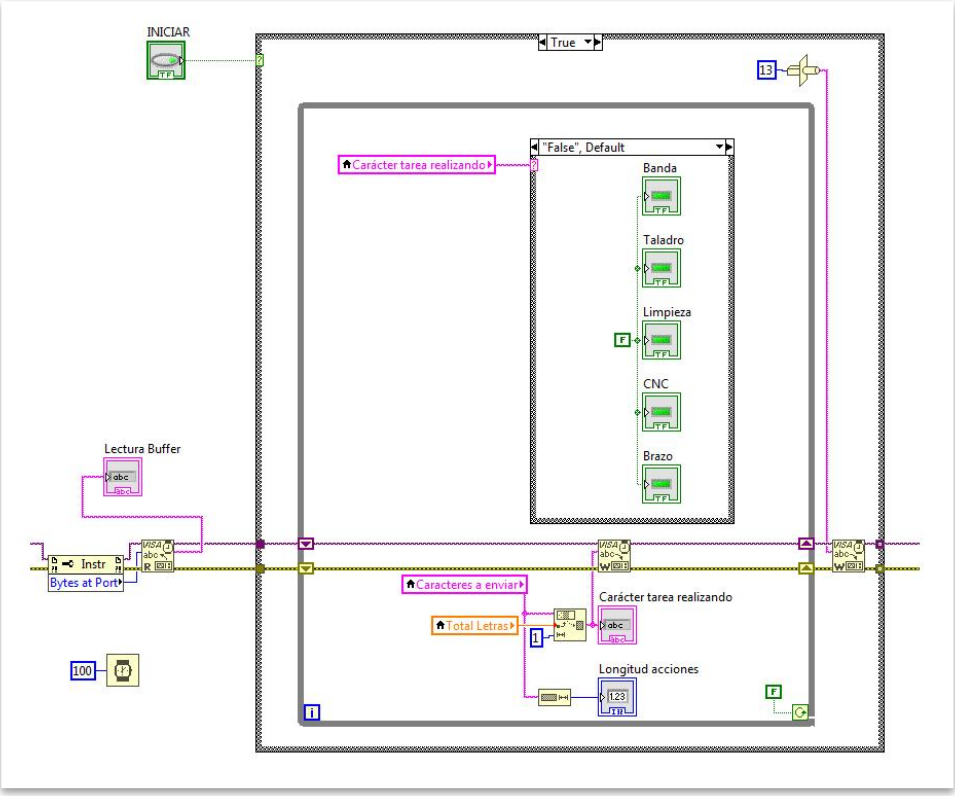


Fig. 50. Estructura case activada (True), controlada por el botón “Iniciar”.

Dentro de la acción verdadera (True) se tiene una estructura while. Las acciones que se ejecutan cuando la estructura case esta activada, comienzan con la herramienta VISA-Write y se usa para escribir en el puerto serie.
Sus entradas se conectan a la salida de la herramienta VISA-Read, para colocar el mismo nombre de puerto y también unir la línea de error. Se debe colocar un control de cadena en la estrada.
En este caso el control de cadena de la entrada se programó para que solo escriba o envié un solo carácter, el cual se refiere a la tarea que debe realizarse en la celda, y va de acuerdo a la rutina planificada.

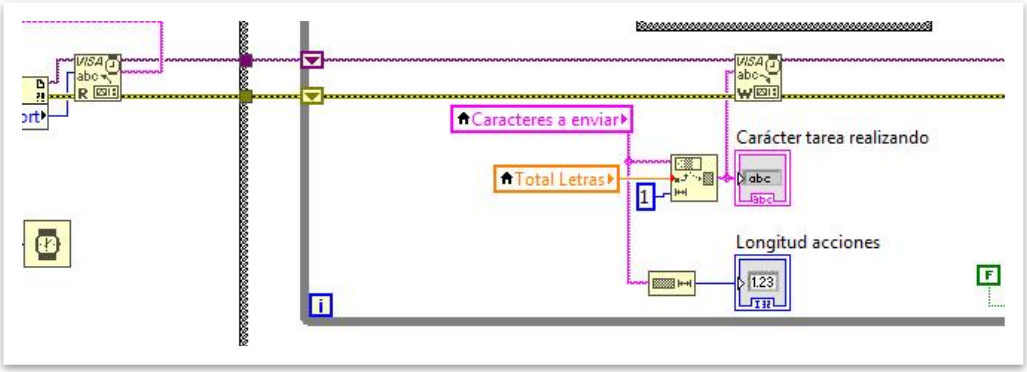


Fig. 51. Conexión herramienta VISA-Write.

El dato que se escribe en el puerto serial lo determina la herramienta “String Subset”. Esta herramienta se usa para extraer los elementos de una cadena de caracteres ya existente, y al resultado se le conoce como subcadena.
A dicha subcadena se le puede asignar el valor máximo de longitud de caracteres, así como desde que valor de la cadena original se debe tomar la muestra. Ambas son definidas mediante valores numéricos.

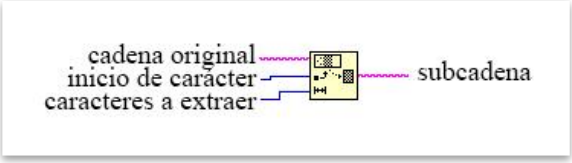


Fig. 52. Herramienta “String Subset”.

En este caso, el sistema controlador toma como cadena original, el contenido del indicador de cadena “Caracteres a enviar”, recordemos que dicho indicador contiene los caracteres a enviar de los botones de tareas activados.
La subcadena debe ser tomada a partir del valor que contenga la variable local referente al control numérico “Total letras”, el cual vale cero, al arrancar el sistema controlador, y por último el valor de caracteres a extraer de la cadena solo será uno, y se define por la constante “1”.

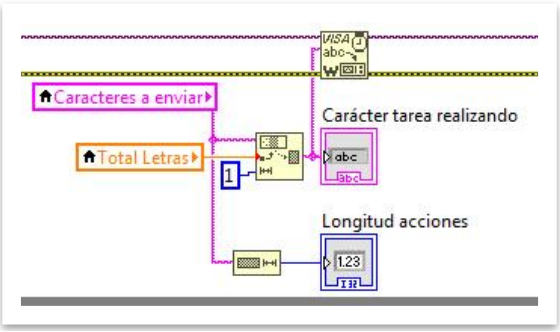


Fig. 53. Herramienta “String Subset” configurada para el sistema controlador.

También se puede ver la herramienta longitud de cadena a la salida de la variable local “Caracteres a enviar”, con su respectivo indicador numérico, “Longitud acciones”, para visualizar el número de caracteres escritos en la cadena “Caracteres a enviar”.
Y el indicador de cadena conectado a la salida de la herramienta “String Subset”, “Carácter tarea realizando”. Que indica que carácter será enviado.

Una vez que se tiene el dato para escribir en el puerto serie, se agrega a la herramienta VISA Write de la estructura while, para ser enviado.

Después de enviar el dato por el puerto serie, a la salida del ciclo while, se conectó otra herramienta VISA Write.
En la entrada del segundo VISA Write se colocó la herramienta “Type Cast Function”, la cual permite moldear un tipo de dato, en este caso un carácter ASCII.
El valor introducido en la herramienta es “13” que equivale en la tabla ASCII a un retorno de carro, lo que permite situar el cursor en la primera posición de la línea.
Mediante esta operación se asegura que el carácter a enviar siempre será el primero de la línea de texto, en proceso.

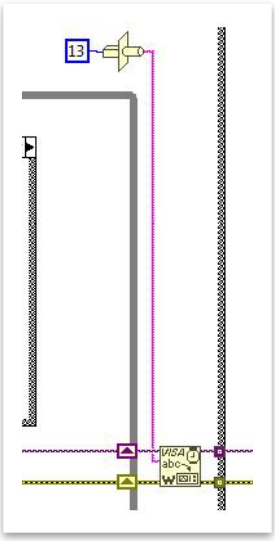


Fig. 54. VISA Write para colocación de cursor en primera posición de línea.

La estructura case que se encuentra dentro de este ciclo while, hace referencia a las imágenes de las estaciones de trabajo.
Es la encargada de iluminar de verde la estación que se encuentra en proceso.
Las imágenes de las estaciones corresponden a cinco leds indicadores.
Cada led indicador tiene el nombre referente a su estación de trabajo. Los nombres de los leds son: “Banda”, “Taladro”, “Limpieza”, “CNC” y “Brazo”.
Los cuales fueron modificados en apariencia. En lugar de mostrar un led típico, se visualiza la imagen de la estación de trabajo.

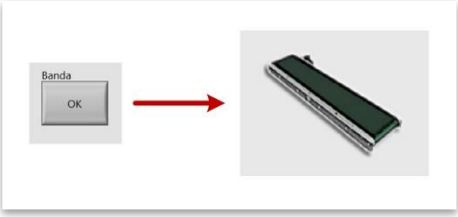


Fig. 55. Botón de control “Banda” modificado, cuando esta desactivado.

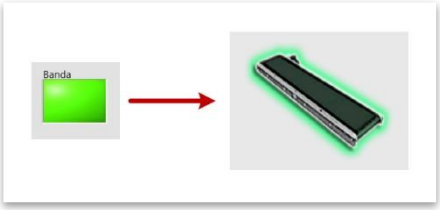


Fig. 56. Botón de control “Banda” modificado, cuando se activa.

La estructura case es controlada por la variable local “Carácter tarea realizando”, y de acuerdo al valor que tenga ejecutara el proceso correspondiente.
El led de la estación se enciende cuando el dato corresponde a una de sus tareas de manufactura y apaga los otros leds que no corresponden al dato.
En caso de no corresponder a ningún valor de la tabla, todos los leds se mantienen apagados.

Led de Estación	Carácter para encender
Banda	b, c, d, e
Taladro	g, h, i, j
Limpieza	1, 3, 4, 5, 6, 8, 9
CNC	l, m, n, o
Brazo	0, B, C, D, E, G, H, I, J, L, M, N, O, R, S, T, U, V, W, X, a, f, k

Tabla 2. Caracteres asignados para encender y apagar Leds de estación.

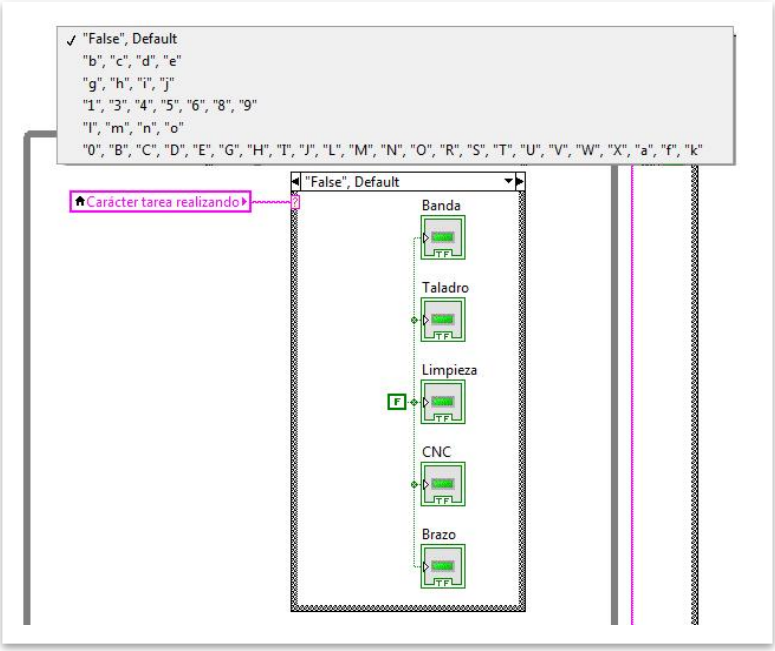


Fig. 57. Estructura case de Leds de estación de trabajo.

La segunda estructura case, que se encuentra dentro de la primera estructura while, se activa de acuerdo a la variable local referente al indicador de cadena “Lectura de Buffer”. Es la encargada de encender o apagar el botón de control “Aumentar letra enviada”.
Se establece una comparación y si en la cadena se encuentra escrita un carácter diferente a “p”, en este caso “q” o cualquier otro, el botón de control “Aumentar letra enviada” se desactiva (F-False).

Fig. 58. Botón “Aumentar letra enviada” desactivado (F=false).

Si en la cadena “Lectura de Buffer” aparece el carácter “p”, el botón de control “Aumentar letra enviada” se activa (T-True).

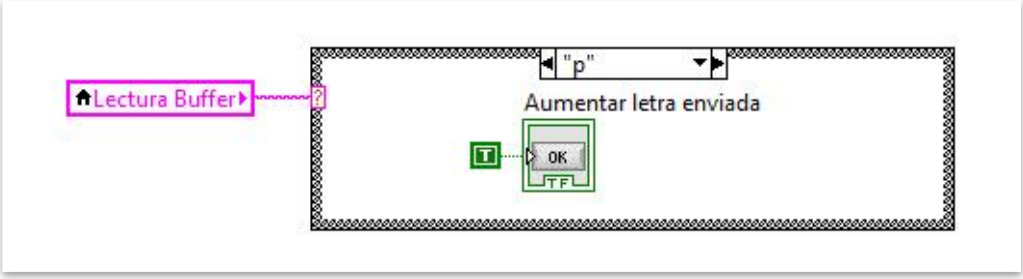


Fig. 59. Botón “Aumentar letra enviada” activado (T-true).

La tercera estructura case se encarga de encender o apagar el botón de control “Reset” y trabaja en función de una comparación.
Si el valor de la variable local “Total letras” es menor a la variable local “Longitud de acciones”, la estructura case toma el valor falso (False), apagando el botón de control “Reset” (F-False).

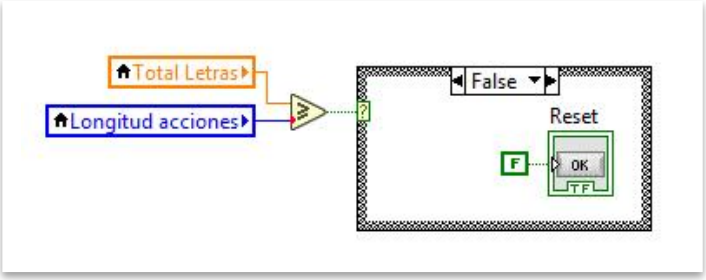


Fig. 60. Estructura case para control de botón “Reset” (F-False).

Si el valor de la variable local “Total letras” es mayor o igual a la variable local “Longitud de acciones”, la estructura case toma el valor verdadero (True), encendiendo el botón de control “Reset” (T-True).

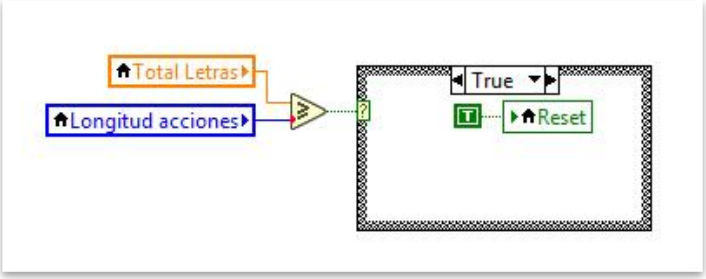


Fig. 61. Estructura case para control de botón “Reset” (T-true).

La cuarta estructura case es controlada por la variable local “Aumentar letra enviada”. Y se encarga de encender o apagar el botón de control “Iniciar”.
Si el botón de control “Aumentar letra enviada” se encuentra desactivado, la estructura case toma el valor de falso (False) y desactiva el botón “Iniciar”.

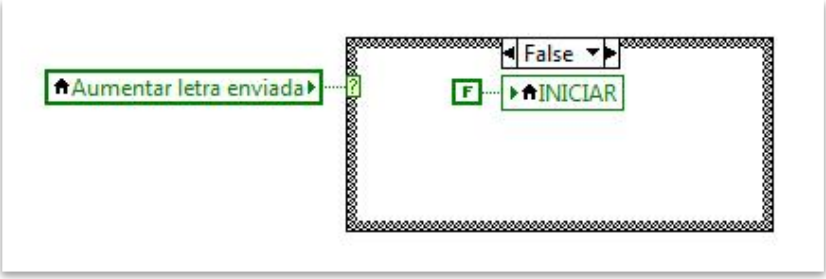


Fig. 62. Estructura case para control de botón “Iniciar” (F-False).

Si el botón de control “Aumentar letra enviada” se encuentra activado, la estructura case toma el valor de verdadero (True), activa el botón “Iniciar” durante un segundo y medio, como lo marca la herramienta de tiempo adjunta.

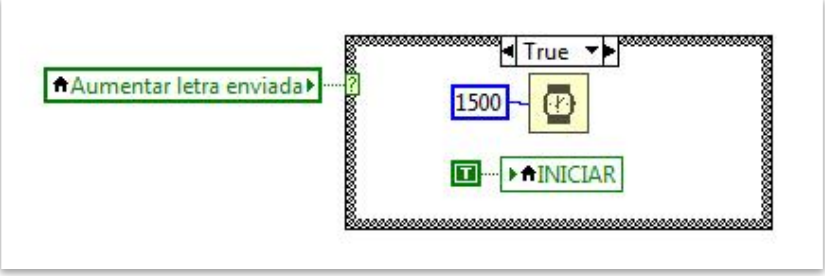


Fig. 63. Estructura case para control de botón “Iniciar” (T-true).

Dentro de la estructura while también se realiza una operación para el funcionamiento del control numérico “Total letras”.
Trabaja mediante la herramienta selector de funciones. La cual devuelve el valor conectado a las estradas “t” y “f”, dependiendo del valor “s”.
Si el valor de “s” es verdadero (True) la función regresa el valor conectado en la terminal t, pero si es falso (False) la función devuelve al valor conectado en la terminal f.

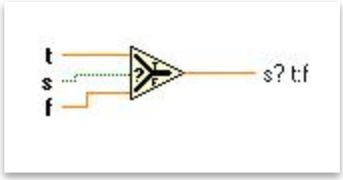


Fig. 64. Herramienta de función “Select Function”.

Esta operación permite poner a cero el valor del control, una vez que se ha sobrepasado el número que contiene el indicador “Longitud acciones”.

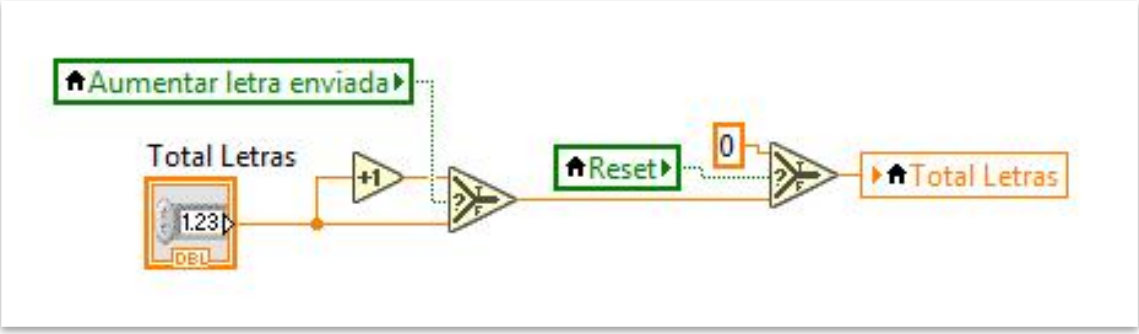


Fig. 65. Operación para poner a cero (0) el control “Total letras”.

El selector del sistema controlador se activa con el botón de control “Aumentar letra enviada”.
Si el botón “Aumentar letra enviada está activado (T-True), en la salida del selector se devuelve el valor del control numérico “Total letras” + una unidad.
Por el contrario, si el botón esta desactivado (F-False), la salida será el valor del control numérico “Total letras” sin cambios.

El resultado pasa por otro selector de funciones. El selector es controlado por el botón “Reset”.
Si el botón “Reset” está activado al valor del indicador numérico “Total letras” se vuelve cero (0).
Si el botón “Reset” no se encuentra activado, el valor del indicador numérico “Total letras” toma el valor asignado por el primer selector.
Esta función permite reiniciar el ciclo de trabajo en el sistema controlador, ya que al enviar el ultimo carácter del indicador “Caracteres a enviar”, se regresa a la posición cero, con lo que el siguiente dato a enviar será el primero de la cadena.

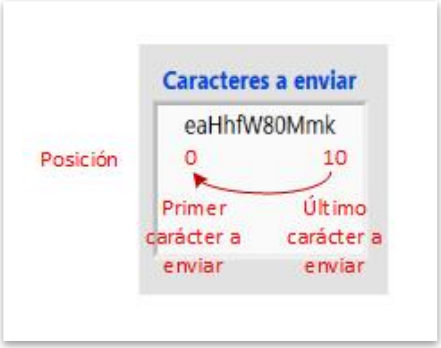


Fig. 66. Envío de último dato, regresa a la posición 0.

3.5 Operación del Sistema Controlador

Para entender la forma de trabajo del Sistema Controlador se ejemplifica una rutina de planificación de tarea.
La rutina consiste en manipular una pieza, la cual realiza las siguientes acciones:

1. Incorporar el material por medio de la banda transportadora, a una velocidad rápida.
2. Realizar un barrenado a la pieza. El barreno debe ser mediante una broca gruesa y que se pueda aplicar en un material suave.
3. En seguida se aplicará un lavado por medio de agua y una vez terminado se realizará un secado para retirar el exceso de líquido.
4. Por último, se deberá dar un acabado a la pieza en la estación de CNC, realizándole una figura del tipo triángulo.

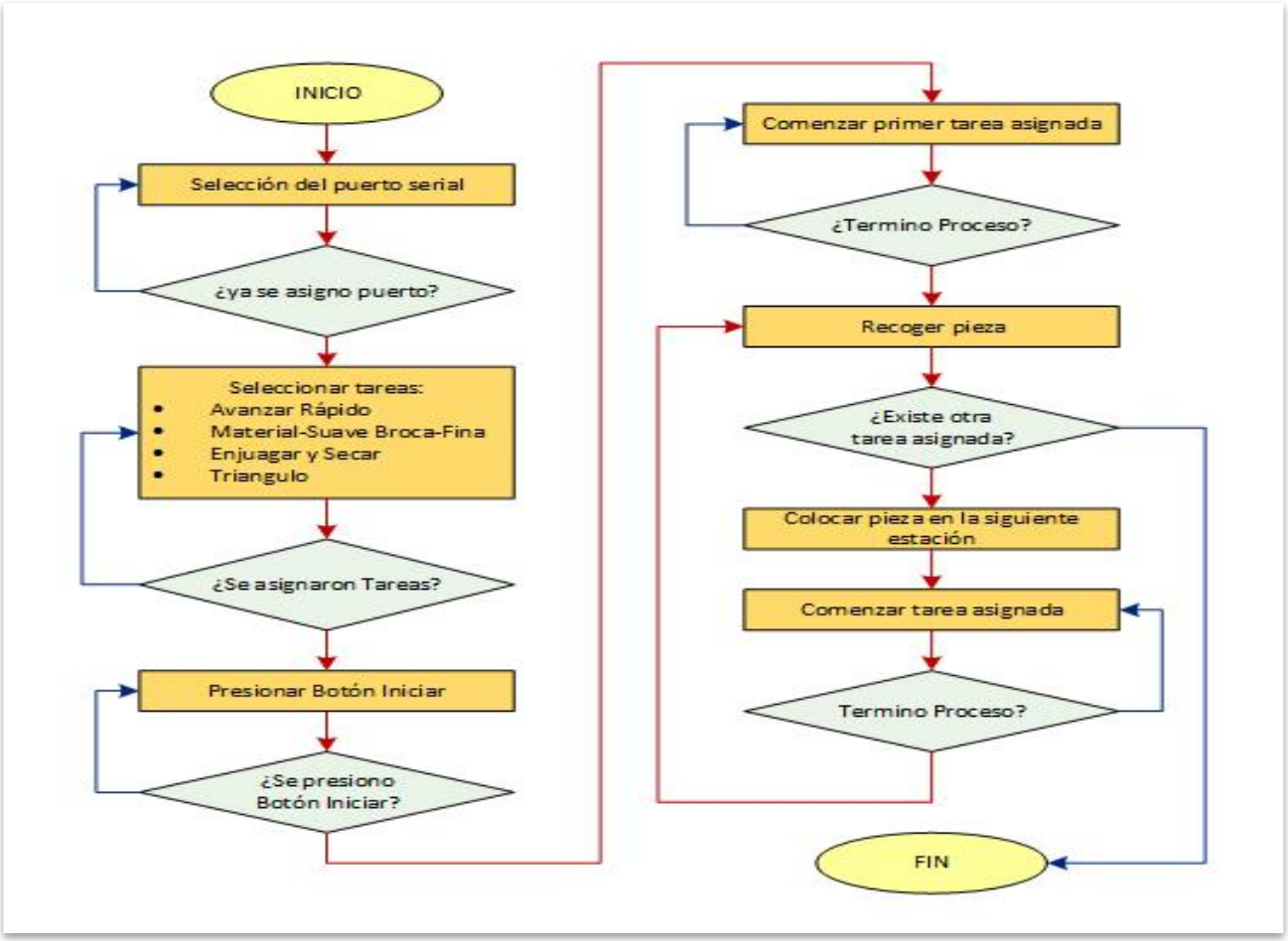


Fig. 67. Diagrama de flujo para el ejemplo.

Los pasos a seguir en el Sistema Controlador serían los siguientes:

1. Como primer paso se debe elegir el puerto donde está conectado el módulo coordinador XBee, de la red implementada.

Fig. 68. Selección del puerto para comunicación.

2. Se presionan los botones correspondientes a las tareas designadas para el proceso de manufactura.

   Conforme se van presionando los botones de las tareas, en la pantalla, se van agrupando las tareas seleccionadas dentro del cuadro “Tarea Planificada”:

Fig. 69. Selección Tareas para el proceso.

3. Una vez elegidas todas las tareas, se presiona el botón Iniciar para comenzar el ciclo de trabajo.

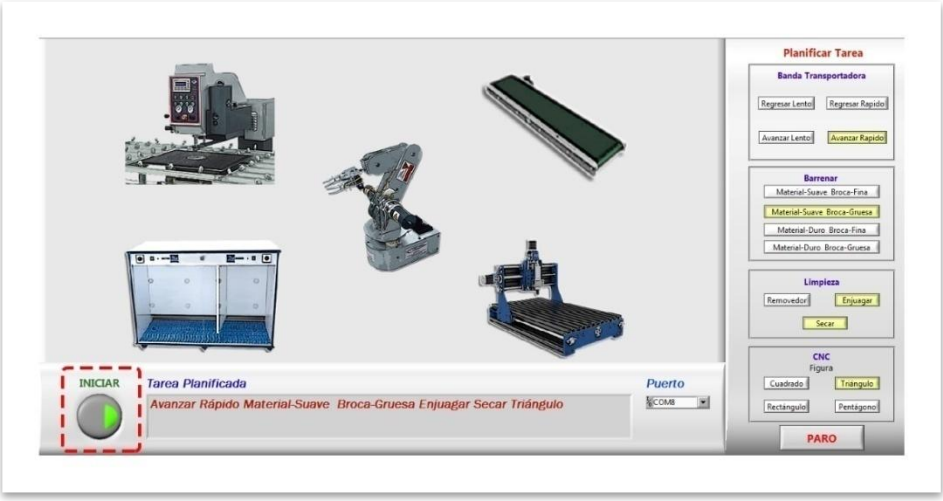


Fig. 70. Comienzo de ciclo, botón Iniciar.

Cuando unas de las estaciones de trabajo y el brazo se encuentran en operación esta se ilumina de color verde, indicando que dicha estación de trabajo está realizando un proceso y al finalizar regresara a su estado original. De este modo el operador sabe qué parte del proceso se encuentra realizando en ese momento.

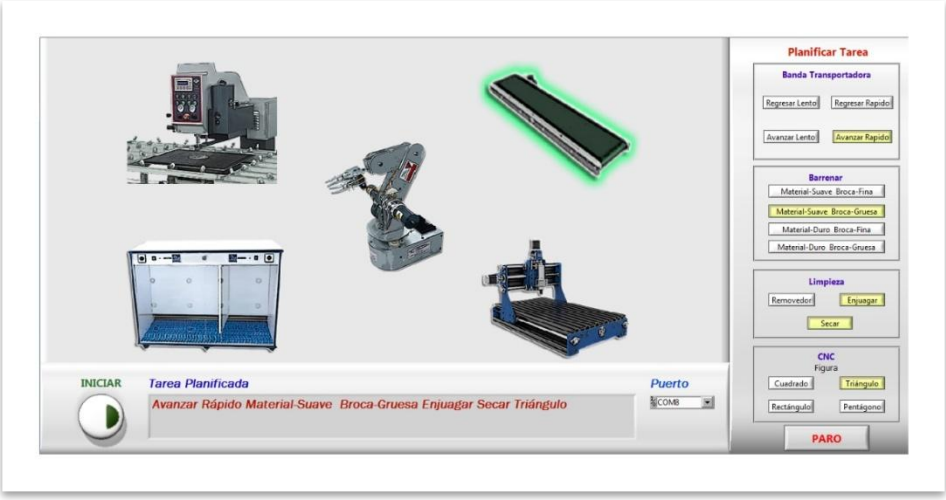


Fig. 71. Modulo iluminado en verde indicando estación trabajando.

Cada vez que se termina una tarea el brazo recoge la pieza de la estación y la lleva a la siguiente. Cuando la pieza es depositada en la siguiente estación, el sistema controlador envía la señal para que dicha estación comience su proceso.

Fig. 72. Sistema controlador en funcionamiento de acuerdo a la secuencia elegida.

4. En caso de requerir que el proceso se detenga se presiona el botón de paro.

Fig. 73. Paro de ciclo

Capítulo 4.

Resultados experimentales

Para la comprobación del sistema se realizaron una serie de pruebas, en las cuales se ejecutaron diferentes rutinas y con ello verificar que el sistema cumpliera las tareas solicitadas.
En el sistema controlador se designaron diferentes procesos de manufactura, cada uno de ellos con distintas tareas, tanto en la interfaz como en la celda de manufactura se verificaron que las tareas propuestas fueran correctas (tabla 15).

Prueba	Tareas asignadas			
	Estación 1	Estación 2	Estación 3	Estación 4
	Banda Transportadora	Barrenar	Limpieza	CNC
1	Avanzar Rápido	Material-Suave Broca-Fina	Enjuagar y Secar	Triangulo
2	Avanzar Lento	---	Removedor	Cuadrado
3	Avanzar Rápido	Material-Suave Broca-Gruesa	Removedor y Enjuagar	---
4	Avanzar Lento	Material-Duro Broca-Fina	---	Rectángulo
5	Avanzar Lento	Material-Suave Broca-Fina	Removedor, Enjuagar y Secar	Triangulo

Tabla 3. Pruebas para resultados experimentales.

En la primera rutina se simula la llegada de material mediante la banda transportadora, con una velocidad de desplazamiento rápido.
Una vez que la pieza haya llegado al final de la banda, se manda la señal de tarea terminada, y comienza a moverse el brazo manipulador para recoger la pieza de la estación 1.
Después de tomar la pieza, es llevada a la siguiente estación, en este caso, se designo la tarea de realizar un barreno al objeto. Como la pieza es de plástico se considera un material suave, y se levara a cabo mediante una broca fina.
Cuando finaliza la tarea, el brazo recolecta la pieza y la manda a la estación 3, ya que se ha seleccionado realizarle una limpieza, y consiste en un lavado por medio de agua, así como el sacado de la misma, y cuando ya termina la tarea, se envía la señal al sistema controlador para proseguir son las tareas restantes.
Por último se solicita realizar un acabado en el CNC al objeto, con una forma de triangulo. La pieza entonces es movida hacia la estación, y se comienza su proceso.
Cuando ya se terminan las tareas designadas, el ciclo comienza desde primera tarea, y se manufactura la siguiente pieza.

Para las demás pruebas, se cambian los valores de las tareas, y se observo que el sistema cumpla con las tareas seleccionadas. Si en alguna estación no se solicita alguna tarea, el sistema controlador no la toma en cuenta y continúa con la siguiente.

Las acciones a realizar en el sistema controlador se describen a continuación.

Después de seleccionar el puerto de conexión, en la lista de control de puertos, los elementos VISA del programa se conectan para poder recibir y enviar datos.
En este caso el modulo coordinador se encuentra conectado en el puerto 8 de la computadora.

Fig. 74. Puerto seleccionado.

Se presionan los cinco botones de control de tarea: “Avanzar rápido”, “Material-Suave Broca Gruesa”, “Enjuagar”, “Secar” y “Triángulo”.

Fig. 75. Botones presionados de las tareas correspondientes a la rutina.

Una vez que se eligen las tareas, los indicadores de cadenas agrupan los mensajes.
En el cuadro de “Tarea Planificada”, se ven los mensajes de las tareas a realizar y en el cuadro de “Caracteres a enviar” los caracteres que el sistema controlador enviara a través del puerto serial.

Fig. 76. Tareas a realizar correspondientes a los botones accionados.

Fig. 77. Caracteres a enviar de acuerdo a las tareas seleccionadas.

En el mensaje, los caracteres de la línea se encuentran ubicados dentro de una posición determinada, el primer carácter se ubica en la posición cero.

# Carácter	1	2	3	4	5	6	7	8	9	10	11
Carácter	e	A	H	h	f	W	8	0	M	m	k
Posición	0	1	2	3	4	5	6	7	8	9	10

Tabla 4. Posición de cada carácter a enviar.

Nota- El número de carácter no equivale a la posición, ya que la posición siempre comienza con el valor cero “0”.

El indicador numérico “Longitud acciones” muestra el valor total de los caracteres a enviar y el indicador “Total letras”, muestra el valor que equivale a la posición del carácter que se envía.

En este caso la longitud de acciones es once (11), se refiere al total de caracteres incluidos en el mensaje “Caracteres a enviar”.
El valor del control “Total letras” vale cero cuando el sistema arranca por primera vez.

Fig. 78. Valores de la longitud de acciones y el total de letras para el ejemplo.

Se presiona el botón de control “Iniciar” para comenzar la rutina de la tarea. El botón cambia su estado a verde al momento de estar presionado, indicando que se activó.

Fig. 79. Botón de control “Iniciar” activado.

Cuando se presiona el botón “Iniciar”, activa la estructura case que controla, la cual realiza la función de escribir en el buffer el dato ingresado a la herramienta VISA Write, dato que depende de la operación de los selectores de función.

Como en el buffer no se encuentra escrito el carácter “p”, la estructura case controlada por el indicador “Lectura Buffer”, es falsa, apagando el botón de control “Aumentar letra enviada”.

Fig. 80. Botón de control “Aumentar letra enviada” apagado.

Al estar apagado el botón “Aumentar letra enviada”, el primer selector de función está en posición “false”, lo que hace que el valor de salida sea igual al valor del control indicador “Total letras” que en este caso vale “0”.

Fig. 81. Control numérico “Total letras” permanece sin cambio.

El carácter que va a enviase se determina por la subcadena que resulta de la herramienta “String Subset”, la cual sabemos, trabaja con los parámetros cadena original, inicio de carácter y caracteres a extraer.

La cadena original se toma del indicador de cadena “Caracteres a enviar”.
El inicio de carácter es el valor del cuadro numérico “Total letras”.
Y una constante igual a “1” que corresponde al valor de caracteres a extraer.

Cadena de Caracteres	**e**	**a**	**H**	**h**	**f**	**W**	**8**	**0**	**M**	**m**	**k**
Posición del carácter en la cadena	0	1	2	3	4	5	6	7	8	9	10

Tabla 5. Posición de cada carácter en la cadena del ejemplo.

- En este ejemplo la cadena original se conforma por los caracteres “eaHhfW80Mmk”.
- El inicio de carácter, vale “0”, entonces la subcadena se toma desde la letra que se encuentra en la posición “0”, la cual es la letra “e”.
- A partir de esa letra solo se extrae un carácter.

Por lo tanto, el resultado del valor de la subcadena equivale a la letra “e” (e minúscula).

De este modo se escribe el valor en el VISA Write y se envía el primer carácter, este se visualiza en el indicador “Carácter tarea realizando”, en este caso es la letra “e”.

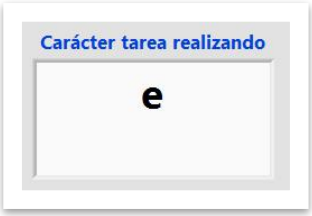


Fig. 82. Carácter “e” enviado a través del puerto serial.

A continuación, se realiza el retorno de carro en el VISA Write, para colocar el cursor al inicio de la línea.

Después de que el sistema controlador envía el primer carácter, por medio de módulo Xbee coordinador, vía inalámbrica, el dato es recibido por todas las estaciones de trabajo, a través de sus módulos Xbee End device.

Cada estación de trabajo tiene un microcontrolador, que está programado para ejecutar diferentes acciones de trabajo, dependiendo del carácter que reciba.

Cuando el microcontrolador recibe un dato por el puerto serial, lo almacena en una variable llamada "ch".
En seguida realiza una comparación de valores y si el valor que contiene la variable "ch" es igual a un carácter designado en el programa, ejecuta las instrucciones y tareas que se le programaron.

	Microcontrolador	*Caracteres que maneja*
1	**Banda transportadora**	***a, b, c, d, e***
2	**Barrenar**	***f, g , h, i, j***
3	**Limpieza**	***0, 1, 3, 4, 5, 6, 8, 9***
4	**CNC**	***k, l, m, n, o***
5	**Brazo**	***q, a, f, 0, k, B, C, D, E, G, H, I, J, L, M, N, O, P, R, S, T, U, V, W, X***

Tabla 6. Caracteres que opera cada microcontrolador.

En este caso el dato enviado es la letra "e", el dato coincide con los caracteres que maneja el microcontrolador 1, de la estación de la Banda transportadora.
Y basados en la tabla 13, equivale a mover la banda transportadora a una velocidad rápida.

En el sistema controlador se enciende el led "Banda", iluminando de verde el fondo de la imagen, indicando que esa estación se encuentra realizando un proceso.

Fig. 83. Led "Banda" activado, señal de que la estación realiza un proceso.

Cuando el microcontrolador de la estación termina su tarea, mover la banda transportadora con una velocidad rápida, la banda se detiene y el programa del microcontrolador envía el carácter “p” (letra *pe* minúscula), señal de que ha finalizado la acción, al Sistema Controlador, a través del módulo XBee End device.

El Sistema Controlador recibe el carácter “p” y lo visualiza en el indicador de cadena “Lectura Buffer”.

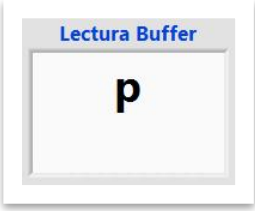


Fig. 84. Valor “p” recibido en el buffer.

Como el valor del indicador “Lectura buffer” es igual a “p”, se activa la estructura case que controla el indicador “Lectura Buffer”, accionando el botón de control “Aumentar letra enviada”.

Fig. 85. Botón de control “Aumentar letra enviada” activado.

Al estar accionado el botón “Aumentar letra enviada”, el primer selector de función está en posición “true”, lo que hace que se sume una unidad al control numérico “Total letras” y adquiera el valor “1”.

Fig. 86. Control numérico “Total letras” aumento una unidad.

Al mismo tiempo, al estar accionado el botón “Aumentar letra enviada”, se activa la estructura case operada por la variable “Letra enviada”, realizando la acción de mantener presionado el botón de control “Iniciar” durante un segundo y medio (1500 milisegundos).

Fig. 87. Botón de control “Iniciar” activado durante 1500 milisegundos.

Como el botón de control “Iniciar” está activado, la estructura case que controla realiza las acciones que están en la posición “True”.
Las cuales son la escritura en el puerto serial, para mandar un dato.

El carácter que va a enviase ahora ha cambiado debido a los valores de entrada en la herramienta “String Subset”, se realiza nuevamente la función.

La cadena original se toma del indicador de cadena “Caracteres a enviar”.
El inicio de carácter es el valor del cuadro numérico “Total letras”.
Y una constante igual a “1” que corresponde al valor de caracteres a extraer.

Cadena de Caracteres	e	a	H	h	f	W	8	0	M	m	k
Posición del carácter en la cadena	0	1	2	3	4	5	6	7	8	9	10

Tabla 7. Posición de cada carácter en la cadena del ejemplo.

- La cadena original no ha cambiado y mantiene los valores de los caracteres “eaHhfW80Mmk”.
- El inicio de carácter, ahora vale “1”, entonces la subcadena se toma desde la letra que se encuentra en la posición “1”, la cual es la letra “a”.
- A partir de esa letra solo se extrae un carácter.

Por lo tanto, el resultado del valor de la subcadena equivale a la letra “a” (a minúscula).

De este modo se escribe el valor en el VISA Write y se envía el siguiente dato, este se visualiza en el indicador “Carácter tarea realizando”.

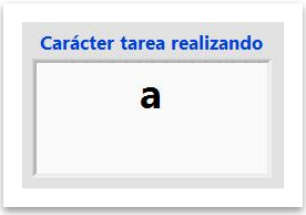


Fig. 88. Carácter “a” enviado a través del puerto serial.

A continuación, se realiza el retorno de carro en el VISA Write, para colocar el cursor al inicio de la línea.

Nuevamente el sistema controlador envía el dato, por medio de módulo Xbee coordinador, vía inalámbrica, el dato es recibido por todas las estaciones de trabajo, a través de sus módulos Xbee End device.

El microcontrolador recibe el dato por el puerto serial y lo almacena en una variable llamada “ch”. Realiza la comparación de valores y si el valor que contiene la variable “ch” es igual a un carácter designado en el programa, ejecuta las instrucciones y tareas que se le programaron.

	Microcontrolador	*Caracteres que maneja*
1	Banda transportadora	***a, b, c, d, e***
2	Barrenar	***f, g , h, i, j***
3	Limpieza	***0, 1, 3, 4, 5, 6, 8, 9***
4	CNC	***k, l, m, n, o***
5	Brazo	***q, a, f, 0, k, B, C, D, E, G, H, I, J, L, M, N, O, P, R, S, T, U, V, W, X***

Tabla 8. Caracteres que opera cada microcontrolador.

En este caso el dato enviado ahora es la letra “a”. El dato coincide con dos microcontroladores.

El microcontrolador 1, de la estación de la Banda transportadora. El cual hace referencia a detener la estación de trabajo.

Y el microcontrolador 5, perteneciente a la estación del brazo. Dato que ejecuta la acción “Mover a estación 1 para agarrar pieza”.
Entonces el brazo realiza los movimientos pertinentes para tomar la pieza de la estación de trabajo 1.

El sistema controlador enciende el led “Brazo”, iluminando de verde el fondo de la imagen, indicando que esa estación se encuentra realizando un proceso.
También apaga los demás leds indicadores que corresponden a las otras estaciones de trabajo.

Fig. 89. Led “Brazo” activado, señal de que la estación realiza un proceso.

Cuando el microcontrolador de la estación termina su tarea, recoger pieza de estación 1, el programa del microcontrolador envía el carácter “p” (letra *pe* minúscula), señal de que ha finalizado la acción, hacia el Sistema Controlador, a través del módulo XBee End device.

El Sistema Controlador recibe el carácter “p” y lo visualiza en el indicador de cadena “Lectura Buffer”.
A partir de este momento el ciclo se repite.

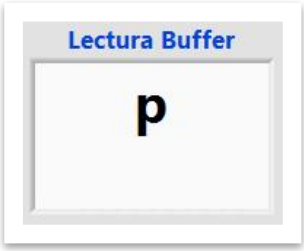


Fig. 90. Valor “p” recibido en el buffer.

El valor del indicador “Lectura buffer” es igual a “p” y se activa la estructura case controlada por la variable “Lectura Buffer”, accionando el botón de control “Aumentar letra enviada”.

Fig. 91. Botón de control “Aumentar letra enviada” activado.

Al estar accionado el botón “Aumentar letra enviada”, el primer selector de función está en posición “true”, lo que hace que se sume una unidad al control numérico “Total letras” y adquiera el valor “1”.

Fig. 92. Control numérico “Total letras” aumento una unidad.

Como el valor del control numérico ahora vale 2, el dato en la subcadena será la letra “H” (mayúscula).

Y el microcontrolador que contenga ese dato realizar el proceso indicado.

Cuando finaliza envía una letra “p”, hacia el Sistema controlador.

El sistema recibe el dato y repiten las operaciones para enviar el siguiente dato.

Existe un momento en el que el valor del control numérico hace referencia a la última posición del indicador “Caracteres a enviar”, en este caso a la posición 10.

En ese momento el valor del control numérico “Total letras” es igual a 11.

Fig. 93. Control numérico “Total letras” igual a 11.

Como el valor del control numérico “Total letras” es igual al valor del indicador numérico “Longitud acciones”, se activa la estructura case operada por ambos elementos, con el valor verdadero. La cual acciona el botón de control “Reset”.

Fig. 94. Botón de control “Reset” activado.

Al momento de accionar el botón reset, la operación que realiza el segundo selector de funciones hace una comparación.
Donde si el estado del botón “Reset” es verdadero (True), el control numérico “Total letras” toma el valor “0”.
Con lo cual el siguiente valor que el sistema controlador envíe, será el carácter que se encuentre en la posición “0” de la cadena “Caracteres a enviar”, de manera que la rutina se repetirá por completo.

Con la serie de pasos descritos se pudo realizar las diferentes pruebas al sistema controlador. En cada uno de ellos se determinaban diferentes acciones o tareas, para la planificación de distintos procesos de manufactura. Dando resultados satisfactorios, pues cada una de ellas realizaba las acciones solicitadas por el operador. En la tabla 9, se visualizan las acciones que se llevaron a cabo de manera correcta por el sistema controlador.

Prueba	Tarea				Resultado
1	Avanzar Rápido	Material-Suave Broca-Fina	Enjuagar y Secar	Triangulo	✓
2	Avanzar Lento	Material-Duro Broca-Gruesa	Removedor	Cuadrado	✓
3	Avanzar Rápido	Material-Suave Broca-Gruesa	Removedor y Enjuagar	Pentágono	✓
4	Avanzar Lento	Material-Duro Broca-Fina	Secar	Rectángulo	✓
5	Avanzar Lento	Material-Suave Broca-Fina	Removedor, Enjuagar y Secar	Triangulo	✓

Tabla 9. Pruebas para resultados experimentales.

Para el diseño de las estaciones de trabajo se trabajaron diferentes elementos electrónicos. Se estudió e investigo la forma de funcionamiento de cada dispositivo, conociendo sus características de trabajo, mediante sus hojas de datos *(Anexo 4)*. Esto permitió manejar distintos actuadores y sensores en cada una de las celdas implementadas.

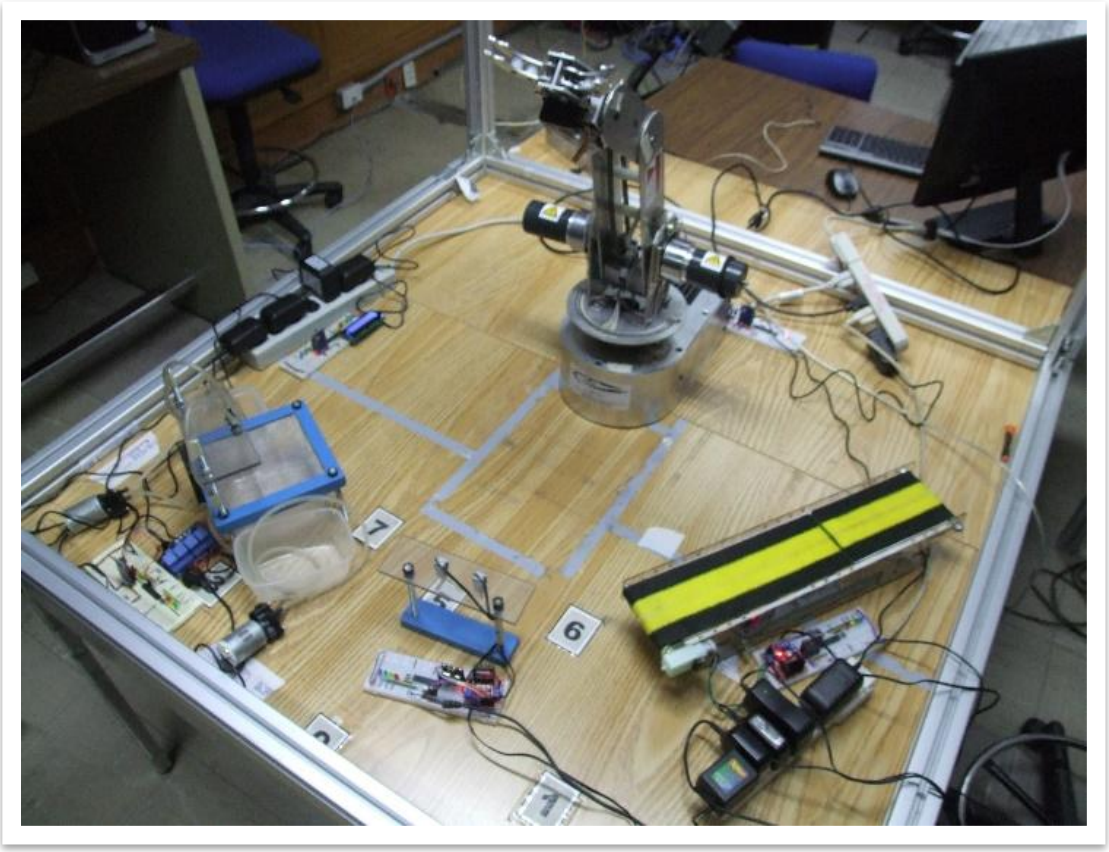


Fig. 95. Sistema controlador. Estaciones de trabajo y Brazo.

Se realizaron diferentes programas para cada uno de los microcontroladores y así coordinar los movimientos finales que se pretendían diseñar *(Anexo 3)*.

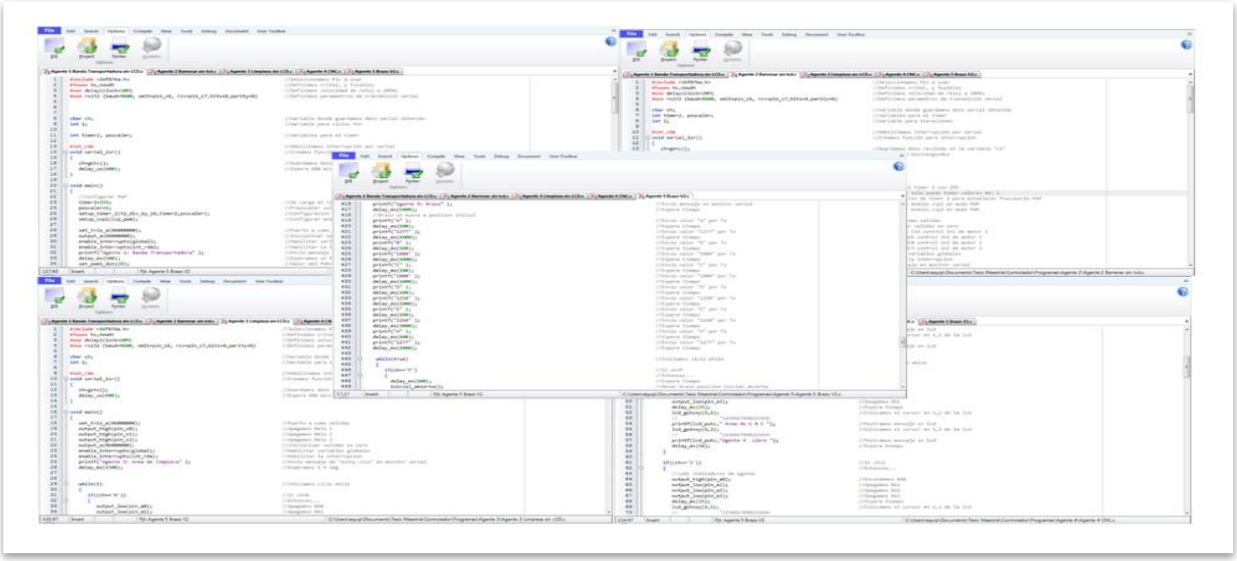


Fig. 96. Programas para cada estación de trabajo.

Respecto a la comunicación inalámbrica se estudiaron los posibles sistemas que pueden ser ocupados para la creación de una red. Los módulos XBee fueron los adecuados con base a las características que posee frente a los demás dispositivos *(Anexo 1)*.

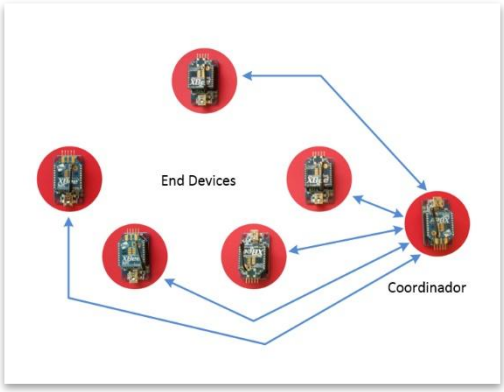


Fig. 97. Red estrella de XBee's. Coordinador y End Devices.

Los dispositivos inalámbricos XBee manejan diferentes versiones, existen desde la serie 1, serie 2, serie Pro, entre otras. Para implementar una red inalámbrica con estos módulos, estos deben ser de la misma serie. En el proyecto se ocuparon módulos serie 2.

Una de las posibles fallas con los módulos XBee, es que, la tecnología cambia rápidamente, los dispositivos evolucionan constantemente, dando como resultado que los primeros productos se dejen de fabricar.

La serie 2 de estos módulos cambio durante el estudio del proyecto, los dispositivos que se agregaron al final, corresponden a otra serie 2, una nueva versión de los anteriores, el cambio radica en el alcance y el rendimiento, pero continúan perteneciendo a esta serie.

Se debieron reconfigurar los primeros para compartir el mismo firmware y se reconocieran y comunicaran entre todos los módulos de manera correcta.

Al brazo robótico se le dio un pequeño mantenimiento debido a que algunos de sus motores ya no realizaban los movimientos que se indicaban, puesto que el laboratorio IIMAS lleva tiempo utilizando, para diferentes proyectos.

Una vez dado el mantenimiento se mejoraron los movimientos de los seis motores que contiene, desplazándose a los lugares solicitados.

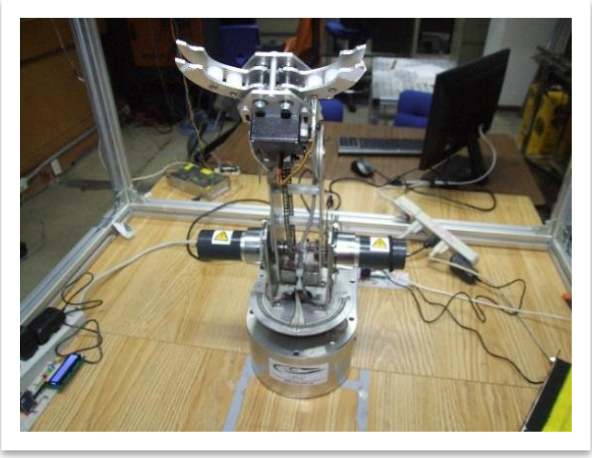


Fig. 98. Brazo del sistema controlador.

Para la interfaz HMI, la cual alberga el sistema controlador, se maneja el software LabVIEW, este software era conveniente para la creación del sistema, contiene funciones y herramientas que facilitan la manipulación de acuerdo a los datos que son adquiridos por los dispositivos conectados al sistema.

Fig. 99. Interfaz HMI ubicada en Laptop junto con el coordinador.

Dentro de las fallas que se encontraron en el desarrollo del sistema, está la adecuada instalación del software de operación.

El sistema LabVIEW, se instaló en dos ocasiones, debido a que las herramientas de comunicación no trabajaban de forma correcta. El programa diseñado contenía los elementos correctos para la manipulación de los datos, pero al no contar con una versión correcta del software, los datos no eran procesados como se esperaba.

Con la instalación correcta de LabVIEW, los datos obtenidos se interpretaban como se debían realizar, mandando y coordinando las acciones que se buscaba ejecutara.

Se realizaron diferentes programas en LabVIEW para controlar de manera individual cada estación y verificar que se llevaran a cabo las acciones diseñadas a cada una. Los cuales mandaban y recibían los datos de forma de correcta.

Una vez verificados los programas individuales se procedió a realizar el programa final, integrando todas las partes necesarias para manejar las estaciones en conjunto.

Fig. 100. Sistema controlador final, *"Planificador de Tareas"*.

CONCLUSIONES

El sistema desarrollado e implementado en una maqueta experimental para manipular objetos dentro de una celda de manufactura consiguió realizar los objetivos planteados en el inicio, logrando el diseño, implementación e integración de un sistema de control digital a nivel maqueta que permitió realizar la experimentación necesaria en la manipulación de objetos dentro de la celda de manufactura y así comprobar su funcionamiento.

El sistema permite planificar las acciones que se deben realizar para un sistema de manufactura como se planteo en la solución del problema y la interfaz HMI implementada, logró la interacción con el usuario para la planificación de tareas requeridas para un proceso y dentro de los alcances configurados. El entorno es práctico, facilitando el modo de operación para la elaboración del producto final.

Cada estación de trabajo realiza diferentes acciones, lo que proporciona una mayor ampliación de tareas a ejecutar dentro de la celda.

Se mostró también que con el uso de los microcontroladores Pic's y su programación, junto a los distintos elementos electrónicos para el proceso e intercambio de información inalámbrica digital, se consiguió procesar las señales y comandar las acciones a realizar en cada una de las estaciones de trabajo.

La comunicación inalámbrica, entre el sistema controlador y las diferentes estaciones de trabajo, se consiguió realizar de manera correcta, mediante los dispositivos XBee, enviando y recibiendo los datos que coordinan las acciones del desplazamiento de los objetos en de la celda.

El brazo robótico utilizado realiza los movimientos que le fueron asignados dentro del área de trabajo establecida, accionando sus actuadores y dirigiendo los movimientos hacia las coordenadas y lugares indicados por la interacción con el planificador de tareas.

El desarrollo de sistemas como el logrado, permite comprobar que es posible la realización de diferentes procesos de manufactura en las empresas, de una manera flexible y reconfigurable, así como la reutilización de las estaciones de trabajo con las que cuenta, dando como resultado una mayor eficiencia en los procesos de manufactura, esto es, reconfigurando el sistema por medio de la interfaz HMI, se asignan los cambios requeridos para la incorporación de nuevos procesos en la celda de manufactura, lo que le da una característica de flexibilidad para desarrollar la manufactura de distintos productos a elaborar utilizando las mismas estaciones de trabajo con tareas asignadas configurables.

Este trabajo significa un logro en las investigaciones realizadas en el desarrollo de sistemas inteligentes de manufactura (SIM), dentro del Departamento de Ingeniería de Sistemas Computacionales y Automatización del IIMAS-UNAM. El desarrollo será de utilidad para la implementación de una celda de manufactura de mayor escala y con un manipulador robótico industrial KUKA (KR 5-2 arc HW), así como estaciones de trabajo que realizan tareas propias para la elaboración de algún producto dentro de los alcances de sus capacidades técnicas

REFERENCIAS

1. P. Grasa OC. Integración de una celda de manufactura flexible. Revist de la universidad de Costa Rica. 1993 Enero/junio; 3(1).

2. P. Valckenaers HVB. Holonic Manufacturing Execution Systems. CIRP Annals, Manufacturing Technology. 2017; 54(1): p. 427-432.

3. Vicente Botti VB. Chapter 2, Holonic Manufacturing Systems. In ANEMONA, A Multi-agent Methodology.: Springer-Verlag London Limited; 2008. p. 7-38.

4. Paulo Leitão MFR. Implementation of a Holonic Control System in a Flexible Manufacturing System. IEEE Systems, Man, and Cybernetics Society. 2008 August; 38(5).

5. Luna-Vazquez Israel LJICcJORAPCM. Hacia la integración de un sistema inteligente de manufactura: consideraciones y experimentos. Queretaro: CIATEQ, A.C. Centro de Tecnología Avanzada, Grupo de Investigación en Mecatrónica y Sistemas Inteligentes de Manufactura (GIMSIM).

6. José Javier Doria García1* FLMCJJC. Implementation of a pick and place process using a robotic crane applied to a flexible manufacturing cell. Scientia et Technica. 2013 Diciembre; 18(4).

7. Kensuke Harada TTKNNYHO. Validating an object placement planner for robotic pick-and-place tasks. Robotics and Autonomous Systems, Elsevier. 2014 October.

8. Taeyong Choi HDDPCPJK. Real-time synchronisation method in multi-robot system. ELECTRONICS LETTERS. 2014 November; 50(24): p. 1824–1826.

9. Hyun Min Do TYCJHK. Automation of cell production system for cellular phones using dual-arm robots. Int J Adv Manuf Technol. 2016 August 12.

10. Supachai Vongbunyong SK(MP. pplication of cognitive robotics in disassembly of products. CIRP, Manufacturing Technology, Elsevier. 2013.

11. Fernando Torres SPCD. Automatic cooperative disassembly robotic system: Task planner to distribute tasks among robots. Control Engineering Practice. Elsevier. 2009 January.

12. Chen WLaX. Dual-Arm Cartesian Robotic System for Parallel Tasking. In International Conference on Intelligent Robots and System; 2001; Maui, Hawaii, USA.

13. Felipe Pérez Roque EVZOAdF. Sistema de Adquisición de datos con comunicación inalámbrica. RIELAC. 2013 septiembre; XXXIV(3): p. 63-73.

14. Oriol Gomis Bellmunt DMMSGAASaASA. A Distance PLC Programming Course Employing. IEEE TRANSACTIONS ON EDUCATION. 2006 May; 49(2).

15. Masterhacks. masterhacks. [Online].; 2013 [cited 2016 Diciembre. Available from: www.masterhacks.net.

.

16 Ignacio Moreno Velasco PLSO. GTE Grupo de Tecnologia Electronica. [Online].; 2007 [cited 2016 Diciembre. . Available from: http://www.gte.us.es/ASIGN/IE_4T/Programacion%20en%20labview.pdf.

17 Osuna PS. http://www.profesaulosuna.com/. [Online]. [cited 2016 Diciembre. Available from: . http://www.profesaulosuna.com/data/files/ELECTRONICA/INSTRUMENTACION/TUTORIAL%20LABVIEW/LabVIEW.pdf.

18 National Instruments. http://www.ni.com. [Online].; 2008 [cited 2017. Available from: . http://www.ni.com/white-paper/7907/es/.

19 Electronica Estudio. http://www.electronicaestudio.com/. [Online]. [cited 2017 Marzo. Available from: . http://www.electronicaestudio.com/microcontrolador.htm.

20 Breijo EG. Compilador ccs y simulador proteus p/microprocesodores pic. Primera ed. Alfaomega , editor. . Barcelona: Alfaomega; 2008.

21 Esteva CB. Basic para microcontroladores pic. Firts Edition edition ed.: Trafford Publishing; 2007. .

22 López AC. http://www.circuitoselectronicos.org. [Online].; 2009 [cited 2016. Available from: . http://www.bairesrobotics.com.ar/data/Manual_Compilador_CCS_PICC.pdf.

23 Aprendiendo Electrónica. http://aprendiendoelectronicafacil.blogspot.mx/. [Online]. [cited 2017. Available . from: http://aprendiendoelectronicafacil.blogspot.mx/2015/08/pic-c-compiler-comunicacion-serial.html.

24 RedPic. http://www.todopic.com.ar/. [Online].; 2009 [cited 2017. Available from: . http://www.todopic.com.ar/foros/index.php?topic=4620.0.

25 Emmanouil Georgakakis SANDDVaCD. An Analysis of Bluetooth, Zigbee and Bluetooth Low Energy and Their . Use in WBANs. In Wireless Mobile Communication and Healthcare; 2010 October; Ayia Napa, Cyprus. p. 168-175.

26 Jin-Shyan Lee YWSaCCS. A Comparative Study of Wireless Protocols. In IECON 2007 - 33rd Annual . Conference of the IEEE Industrial Electronics Society; 2007; Taipei, Taiwan. p. 46-51.

27 Julen Iraceburu González JGF. Desarrollo e implementación de una red inalámbrica de sensores de . temperatura y humedad. 2014..

28 Freddy A. Leal González MMHC. Implementación de protocolos inteligentes de comunicación y control . generando nuevas soluciones en automatización y sistemas de domótica e inmótica. In 5to congreso iberoamericano de estudiantes de ingeniería eléctrica; 2012; Mérida, Estado Mérida, Venezuela. p. 23-27.

29 Fortuño AG. Desarrollo e implementación de una red de sensores Zigbee mediante el dispositivo Xbee de

. Digi. Mayo 2012..

30 Ingeniería MCI Ltda. http://xbee.cl/. [Online]. Available from: http://xbee.cl/que-es-xbee/.
.

31 MACTRONICA. http://www.mactronica.com.co/. [Online]. Available from:
. http://www.mactronica.com.co/adaptador-xbee-explorer-usb-43740324xJM.

32 CRYA, CORROB S.A DE C.V. Manual del Sistema Labot Pro 5. 2008..
.

33 diymakers. http://diymakers.es. [Online].; 2013 [cited 2017. Available from: http://diymakers.es/control-
. velocidad-y-sentido-de-motor-dc/.

34 Cruz A. https://electronilab.co/. [Online].; 2014 [cited 2017. Available from:
. https://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/.

35 wikipedia. https://es.wikipedia.org. [Online]. Available from: https://es.wikipedia.org/wiki/Rel%C3%A9.
.

36 Tolocka P. http://www.profetolocka.com.ar. [Online].; 2015 [cited 2017. Available from:
. http://www.profetolocka.com.ar/2015/05/09/modulo-de-4-reles-para-arduino/.

37 diymakers. http://diymakers.es. [Online].; 2014 [cited 2017. Available from: http://diymakers.es/aprender-
. usar-un-display-lcd/.

38 Llamas L. https://www.luisllamas.es. [Online].; 2016 [cited 2017. Available from:
. https://www.luisllamas.es/bomba-de-agua-con-arduino/.

39 Llamas L. https://www.luisllamas.es. [Online].; 2016 [cited 2017. Available from:
. https://www.luisllamas.es/controlar-un-ventilador-con-arduino/.

ANEXO 1

LabVIEW

El controlador se desarrolló utilizando el software LabVIEW de la compañía National instruments. Es un programa para ingeniería de sistemas que requieren pruebas, medidas y control con acceso rápido a hardware y análisis de datos, un entorno de programación destinado al desarrollo de aplicaciones, similar a los sistemas de desarrollo comerciales que utilizan el lenguaje C o BASIC. Sin embargo, se diferencia de dichos programas en un importante aspecto: los citados lenguajes de programación se basan en líneas de texto para crear el código fuente del programa, mientras que LabVIEW emplea la programación gráfica o lenguaje G para crear programas basados en diagramas de bloques.

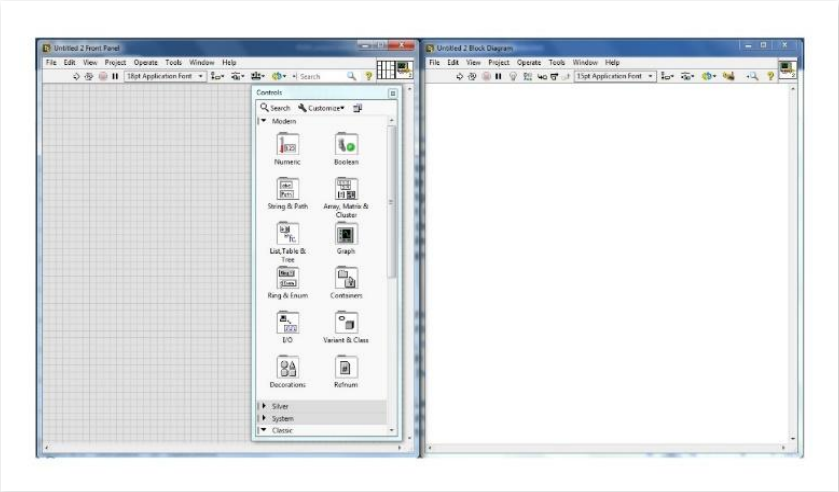


Fig. 101. Entorno de trabajo LabVIEW, National Instruments.

Posee extensas librerías de funciones y subrutinas. Además de las funciones básicas de todo lenguaje de programación, LabVIEW incluye librerías específicas para la adquisición de datos, control de instrumentación VXI, GPIB y comunicación serie, análisis presentación y guardado de datos, también proporciona potentes herramientas que facilitan la depuración de los programas.
Los programas desarrollados mediante LabVIEW se denominan Instrumentos Virtuales (VIs), porque su apariencia y funcionamiento imitan los de un instrumento real. Sin embargo, son análogos a las funciones creadas con los lenguajes de programación convencionales. Los VIs tienen una parte interactiva con el usuario y otra parte de código fuente, y aceptan parámetros procedentes de otros VIs.

Todos los VIs tienen un panel frontal y un diagrama de bloques. Las paletas contienen las opciones que se emplean para crear y modificar los VIs.

Panel Frontal

Se trata de la interfaz gráfica del VI con el usuario. Esta interfaz recoge las entradas procedentes del usuario y representa las salidas proporcionadas por el programa. Un panel frontal está formado por una serie de botones, pulsadores, potenciómetros, gráficos, etc.

Cada uno de ellos puede estar definido como un control (a) o un indicador (b). Los primeros sirven para introducir parámetros al VI, mientras que los indicadores se emplean para mostrar los resultados producidos, ya sean datos adquiridos o resultados de alguna operación.

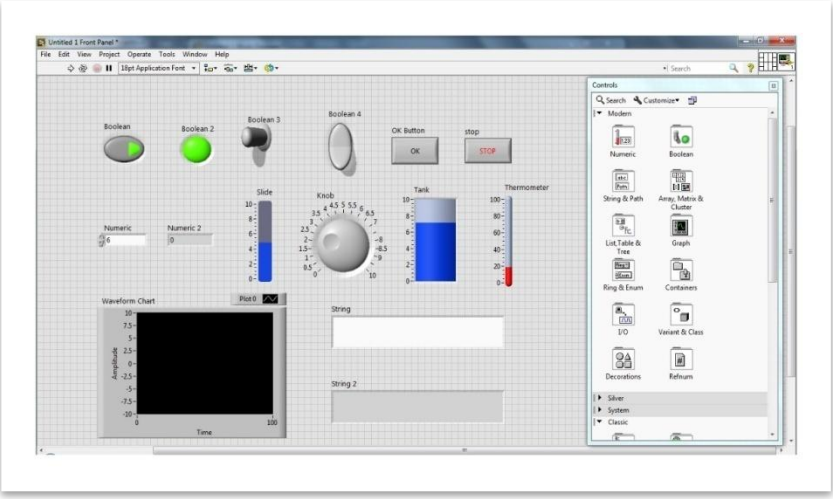


Fig. 102. Panel Frontal LabVIEW.

Diagrama de bloques

El diagrama de bloques constituye el código fuente del VI. En el diagrama de bloques es donde se realiza la implementación del programa del VI para controlar o realizar cualquier procesado de las entradas y salidas que se crearon en el panel frontal.
El diagrama de bloques incluye funciones y estructuras integradas en las librerías que incorpora LabVIEW. En el lenguaje G las funciones y las estructuras son nodos elementales. Son análogas a los operadores o librerías de funciones de los lenguajes convencionales.
Los controles e indicadores que se colocaron previamente en el Panel Frontal, se materializan en el diagrama de bloques mediante los terminales.
El diagrama de bloques se construye conectando los distintos objetos entre sí, como si de un circuito se tratara. Los cables unen terminales de entrada y salida con los objetos correspondientes, y por ellos fluyen los datos.
LabVIEW posee una extensa biblioteca de funciones, entre ellas, aritméticas, comparaciones, conversiones, funciones de entrada/salida, de análisis, etc.
Las estructuras, similares a las declaraciones causales y a los bucles en lenguajes convencionales, ejecutan el código que contienen de forma condicional o repetitiva (bucle for, while, case, etc.).
Los cables son las trayectorias que siguen los datos desde su origen hasta su destino, ya sea una función, una estructura, un terminal, etc. Cada cable tiene un color o un estilo diferente, lo que diferencia unos tipos de datos de otros.

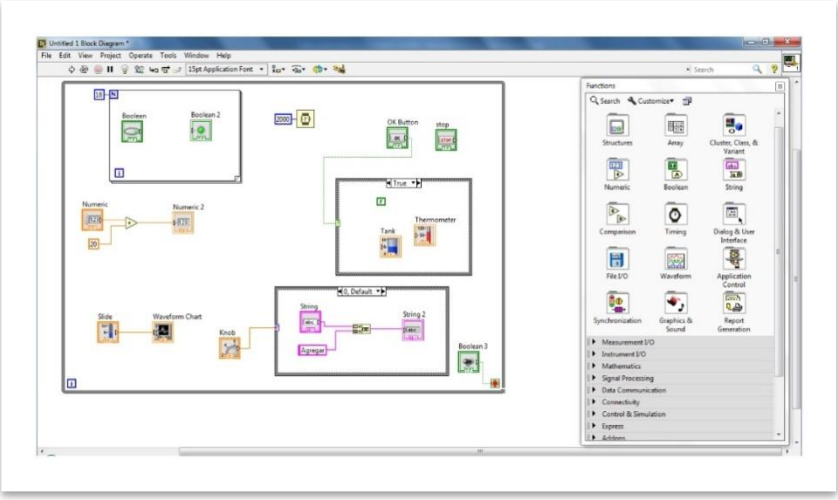


Fig. 103. Diagrama de Bloques LabVIEW.

El panel frontal es la interface del usuario con el VI. Cuando un programa está terminado, el usuario final hace uso del panel frontal, donde se encuentra todo lo necesario para controlar un sistema. Los controles son botones, botones de empuje, marcadores y otros componentes de entradas. Los indicadores son las gráficas, luces y otros dispositivos. Los controles simulan instrumentos de entradas de equipos y suministra datos al diagrama de bloques del VI. Los indicadores simulan salidas de instrumentos y suministra datos que el diagrama de bloques adquiere o genera.
Mediante la representación en pantalla de los elementos gráficos de visualización y control que servirán de interfaz con el usuario, este observará los estados de las entradas seleccionadas en la pantalla e interactuará con las salidas directamente o mediante la ejecución de las rutinas que se hayan programado.

En LabVIEW se pueden agregar Toolkit (Juegos de Herramientas o controladores de instrumentos), que son paquetes para manejar y configurar dispositivos específicos. Las herramientas son descargadas directamente de la página oficial de National Instruments, proporcionan una selección completa de los controladores de instrumentos.
La mayoría de las PCs incluyen un puerto serial (generalmente RS232). Aquellas que no tienen este puerto generalmente pueden ser aumentadas con una interfaz serial vía ranuras PCI, adaptadores de puertos USB y otros. La única configuración de hardware requerida es conectar el cable serial al puerto serial en la PC y el puerto serial en el instrumento. El instrumento serial puede incluir algunos controladores de hardware o utilidades de software para comunicación. Puede contener documentación sobre velocidad de transferencia, tamaño del paquete, bits de paro y bits de paridad que el instrumento usará. Estas especificaciones son necesarias para asegurar la comunicación adecuada en el bus serial.
Uno de los Toolkit que se incorporaron en el sistema es él NI-VISA, un software de interfaz de E/S Universal, el cual sirve para configurar, programar y depurar sistemas de instrumentación que comprenden interfaces GPIB, VXI, PXI, serial (RS232/RS485), Ethernet/LXI y/o interfaces USB, (15),(16), (17), (18).

PIC C Compiler

Para la creación de los programas que se cargaran en los microcontroladores se usa el software llamado PIC C Compiler (CCS), creado por la compañía CCS Custom Computer Services inc. Es un entorno de trabajo inteligente y optimizado para la compilación de programas para microcontroladores Pic, el cual contiene operadores estándar del lenguaje C y funciones incorporados en bibliotecas que son específicas para los registros de Pic, (20), (21), (22)., que simplifican el acceso al hardware, proporciona un eficiente y altamente optimizado código. Incluye funciones de hardware del dispositivo de características tales como:

* Temporizadores y módulos PWM
* Convertidores A / D
* Memoria de datos on-chip EEPROM
* LCD controladores
* Memoria externa buses
* Entre otras...

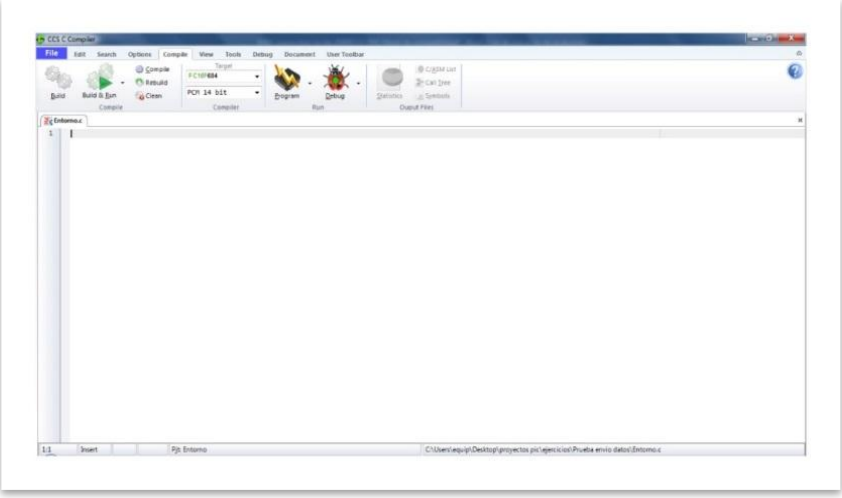


Fig. 104. Entorno de trabajo Pic C Compiler, CCS.

PICkit 2 Programmer

Se trata de un software creado por Microchip Technology Inc., para programar y depurar las memorias FLASH, EEPROM y los fusibles de configuración de los microcontroladores Microchip, utilizando una tarjeta programadora.
Permite abrir directamente el código de programación compilado, el código objeto (.hex), para poder ser cargado a los microcontroladores, mediante una tarjeta de programación, en la cual se coloca el integrado.
Soporta línea base (PIC10F, PIC12F5xx, PIC16F5xx), gama media (PIC12F6xx, PIC16F), PIC18F, PIC24, dsPIC30, dsPIC33 y familias PIC32 de microcontroladores de 8 bits, 16 bits y 32 bits, y muchos productos Microchip Serial EEPROM.

Fig. 105. Entorno de trabajo Pickit2 Programmer, Microchip Technology Inc.

Fig. 106. Tarjeta programadora de microcontroladores Pic.

Sistema de comunicación inalámbrica

El sistema está basado en microcontroladores PIC y una computadora central a los que se les ha incorporado un módulo de transmisión/recepción del tipo XBee, basados en la tecnología ZigBee, que permite la comunicación inalámbrica bidireccional entre ellos. La adquisición, digitalización, procesamiento, transmisión, así como el almacenamiento y la presentación de la información, se realiza con el empleo de programas diseñados específicamente para esta aplicación.

Comunicación serial CCS Compiler – PIC

Los microcontroladores PIC usan el módulo USART, para realizar comunicación en serie (Universal Synchronous Asynchronous Receiver Transmitter), (23).

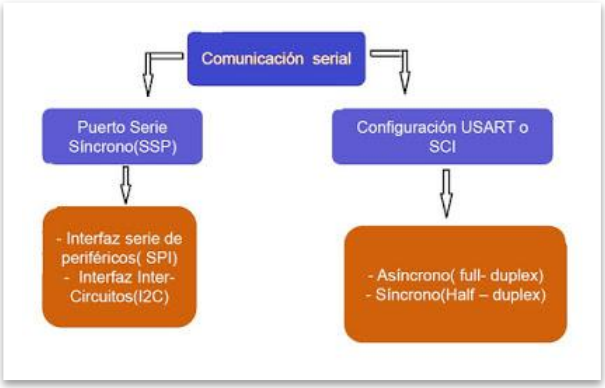


Fig. 107. Diagrama comunicación serial.

La función del USART es la de transmitir y recibir datos, es compatible con el protocolo RS232.

TRANSMISIÓN SÍNCRONA

Los datos se transfieren de forma continua, no existe límite de tamaño
Características:

- Modo semi-duplex
- La comunicación serie se transmite en una sola línea, en ambos sentidos.
- No se pueden enviar información en ambos sentidos de forma simultánea
- La transmisión puede ser maestro o esclavo

Fig. 108. Transmisión síncrona

TRANSMISIÓN ASÍNCRONA

En este modo se emplean relojes tanto en el emisor como el receptor. La frecuencia del reloj se acuerda antes de la transmisión. La sincronización se realiza durante la transmisión
La transmisión es full-dúplex (se utilizan dos líneas, una de transmisión (Tx) y otra de recepción (Rx))
Cada trama de datos tiene un tamaño fijo y posee un bit de arranque (inicio) y un bit de parada (final).

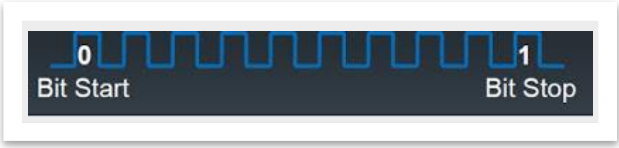


Fig. 109. Trama de datos de envió

El modo de comunicación en el proyecto será Asíncrono, ya que los dispositivos Xbee constan de dos líneas Tx y Rx, una para transmisión y otra para recepción.

Para establecer la interacción entre los componentes se debe programar el microcontrolador cargando la función para configurar el módulo USART en C:

#USE RS232 (opciones)

El cual permite configurar las características del módulo USART:

Opciones	Descripción
BAUD = X	Velocidad en Baudios
XMIT = pin	Pin de transmisión
RCV = pin	Pin de recepción
PARITY =X	Donde X es N, E, u O
BITS = X	Tamaño de trama (5.9)

Tabla 10. Opciones para configurar comunicación RS232 en Pic C Compiler.

Sintaxis de la función para conexión rs232 en el microcontrolador:

```
#use rs232(baud=9600, xmit=pin_c6, rcv=pin_c7, parity=N, bits=8)
```

Función	Descripción
putc(data)	Envía un dato. data es un carácter de 8 bits
putchar(data)	
puts(string)	Envía una cadena. string es una cadena de caracteres constante o una matriz de caracteres terminada con un cero
printf(fname,cstring,values)	Escribe en monitor serial. fname: función a utilizar para escribir la cadena indicada, por defecto se utiliza putc() cstring: cadena de caracteres o matriz de caracteres terminada en 0. values: valores a indicar en la cadena, se debe indicar %nt.
value=getc()	Recibe dato. value es un carácter de 8 bits. Espera recibir un carácter por la línea RS232 y devuelve su valor.
value=getch()	
value=getchar()	
valor=kbhit()	Compara si recibió dato. valor=0 (false) si getc() debe esperar a que llegue un carácter valor=1 (true) si ya hay un carácter para ser leído por getc()

Tabla 11. Funciones para Transmisión y recepción de datos en Pic C Compiler.

El hardware mínimo que hay que montar para establecer la conexión se muestra en el siguiente esquema:

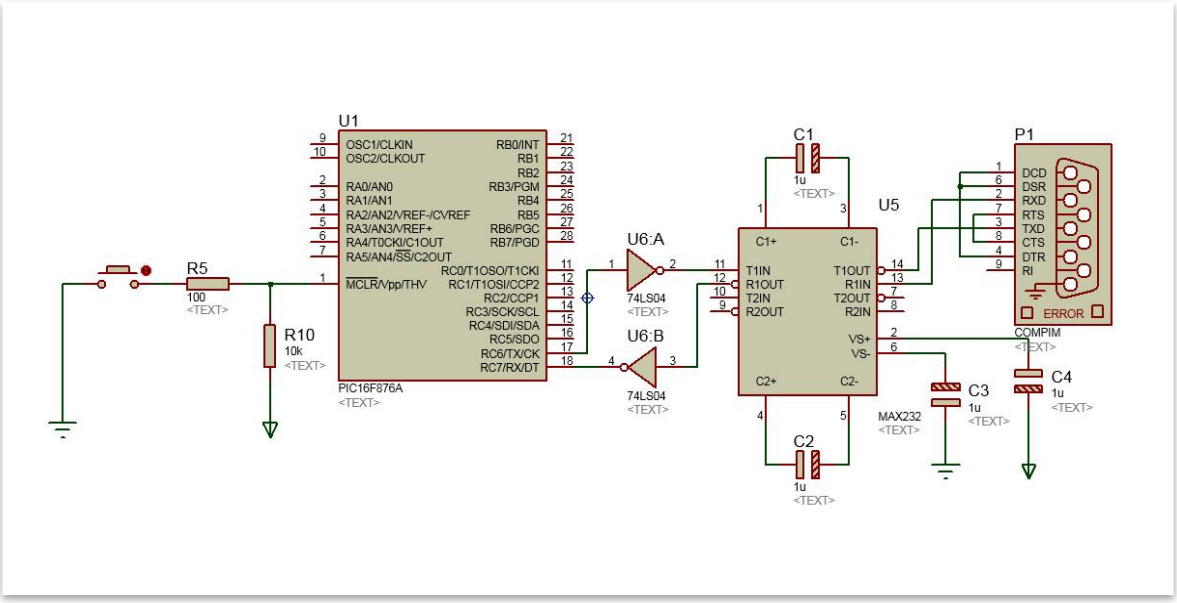


Fig. 110. Esquema PIC - RS232.

Ejemplo de Programa CCS para comunicación, (24):
Lo único que hace es mandar mensaje de espera, y posteriormente se pone a devolver, como eco, cualquier carácter que reciba por el puerto serie.

```
//************************************************************
#include <16F876a.h>
#fuses XT,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT
#use delay(clock=4000000)
#use rs232(baud=9600, xmit=pin_C6, rcv=pin_C7)

main() {
  printf("PIC16F876A a la espera" );  // Envío mensaje para mostrar comunicación
  while(true) {
    putc(getc());                     // al presionar cualquier tecla se envía y produce
                                      //eco de lo que recibo
  }
}
//************************************************************
```

Protocolos de comunicación en Tecnologías inalámbricas

Entre las tecnologías inalámbricas que existen el mercado se pueden encontrar Bluetooth, ZigBee, ANT, Wi-Fi, (25), (26).

Bluetooth

Es un estándar de tecnología inalámbrica para el intercambio de datos a cortas distancias (empleando un enlace por radiofrecuencia en la banda ISM de 2,4 GHz) desde dispositivos fijos y móviles construyendo redes de área personal (PAN). La distancia nominal del enlace está comprendida entre 10cm y 10m, pero aumentando la potencia de transmisión se puede llegar a 100m

ANT

Es una tecnología propietaria de libre acceso para redes inalámbricas de sensores multicast que ofrece una pila de protocolos de comunicación inalámbrica para dispositivos que operan en la banda de 2,4 GHz (aplicaciones industriales, científicas y médicas) permitiendo a estos comunicarse estableciendo reglas estándar de coexistencia, representación de datos, señalización, autentificación y detección de errores.
El protocolo ANT está disponible en transceptores de baja potencia de fabricantes como Nordic Semiconductor y Texas Instruments. Se caracteriza por una baja carga computacional y una baja eficiencia, lo que se traduce en un consumo de energía mínimo. Las aplicaciones más conocidas y difundidas de ANT se encuentran en el sector del deporte, más en concreto en fitness y ciclismo. En estas, los transceptores se integran en elementos como relojes o cinturones que miden parámetros tales como velocidad y distancia permitiendo al usuario una monitorización de su rendimiento.

- ZigBee

Surge de la necesidad de desarrollar una tecnología inalámbrica de no muy alta transferencia de datos. En 1998, un conjunto de empresas, conocidas como la ZigBee Alliance se juntaron para crear un estándar de comunicaciones que complementara a Wi-fi y Bluetooth. Surge así ZigBee, publicado por el IEEE en mayo de 2003.
ZigBee se establece como principal objetivo el de comunicar aplicaciones que requieren una comunicación segura, con baja tasa de envío y bajo consumo. Se basa en dispositivos inalámbricos operando en la banda ISM para usos industriales, científicos y médicos (868 MHz, 915 MHz y 2.4 GHz) con una modulación en espectro ensanchado por secuencia directa (DSSS) también conocida como acceso múltiple por división de código en secuencia directa (DS-CDMA). En el rango de

frecuencias de 2.4 GHz (banda más extendida) se definen 16 canales con un ancho de banda de 5 MHz, (27).
Tiene como ventajas: el bajo coste, bajo consumo de potencia, uso de banda de radio libre y sin necesidad de licencia, instalación económica y simple, redes flexibles y extensible. El protocolo ZigBee permite desde reemplazar un cable por una comunicación serial inalámbrica, hasta el desarrollo de configuraciones punto a punto, punto a multipunto, peer to peer (todos los nodos conectados entre sí) o redes complejas de sensores. La red ZigBee está formada básicamente por tres elementos, un único dispositivo coordinador, dispositivos router y dispositivos finales (end points). Una de las mayores aportaciones del protocolo ZigBee y el que mayor interés despierta, es el concepto de la red nodal o mesh network por el que cualquier dispositivo puede conectar con otro, utilizando varios dispositivos como repetidores, (28).

El protocolo de comunicación que se utilizará en el proyecto será ZigBee, que es con el que trabajan los módulos XBee.

	Wi-Fi	Bluetooth	ZigBee
Velocidad	<50 Mbps	1 Mbps	<250 Kbps
Numero de nodos	32	8	255/65535
Consumo transmisión	400 ma.	40 ma.	30 ma.
Consumo reposo	20 ma.	0.2 ma.	2µa.
Precio	Alto	Medio	Bajo
Configuración	Compleja	Compleja	Simple
Aplicaciones	Internet	Informática y móviles	Domótica y monitorización

Tabla 13. Cuadro comparativo entre Wi-Fi, Bluetooth y ZigBee, (29)

Tipos de dispositivos para ZigBee

Coordinador - Es el dispositivo más completo. Controla el ruteado y la administración de la red. Solo existe uno por red.
Router - Se encarga de interconectar los dispositivos mediante técnicas de encaminamientos y direccionamiento.
End Device – Es un elemento pasivo de la red, ya que no transmite información de manera autónoma; simplemente dispone de la funcionalidad mínima para ser capaz de responder a peticiones de dispositivos superiores (coordinador o router).

ZigBee está diseñado para ser una solución que permita crear redes e interconectar dispositivos remotos. De esta forma existen diversas topologías de red que se pueden formar con los dispositivos previamente mencionados.

Topologías de red

Par

La forma más sencilla con la red es de dos nodos. Uno de ellos debe ser un Coordinador, el otro puede ser bien un Router o un End device.

Estrella – en esta topología el coordinador es el centro de la red y es el que se conecta en círculo con los demás dispositivos (End Devices). Por lo tanto, todos los mensajes deben pasar por el coordinador. Dos End device no pueden comunicarse entre sí directamente.

Malla

La configuración cuenta con nodos Router y con un nodo Coordinador. Se trata de una topología no jerárquica, en el sentido de que cualquier dispositivo puede interactuar con cualquier otro. Este tipo de tecnología permite que, si en un momento un nodo o camino fallan en la comunicación, esta pueda seguir rehaciendo los caminos. La gestión de los caminos es tarea del coordinador.

Árbol

Es una variación de la topología malla, por lo que no se considera una cuarta topología. En este diseño los Routers forman una columna vertebral con los dispositivos finales que están agrupados en torno a los Routers.

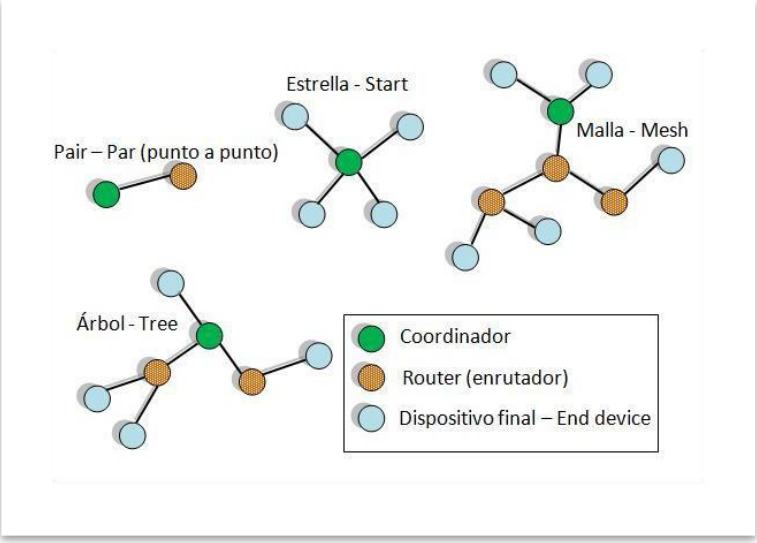


Fig. 111. Topologías de Red ZigBee, (27).

XCTU

Para los dispositivos inalámbricos utilizados en el proyecto, se requiere la configuración de los componentes y para eso se utilizará el software desarrollado por Digi International Inc. llamado XCTU.
Se utilizará para configurar los módulos XBee y crear una topología de red, del tipo estrella, compuesta por un Coordinador y cinco puntos finales (End points).

Es una aplicación multiplataforma, se puede instalar en sistemas como Windows y MacOS, además de ser gratuito. Interactúa con el firmware de los módulos Digi RF. Provee una interfaz gráfica intuitiva de fácil uso. Un visor grafico de red para una simple configuración de la nueva arquitectura XCTU. Permite testear en los módulos de RF, el alcance, la detección otros módulos en la red, leer entradas del módulo,
También ofrece una pantalla terminal para comunicarse con los módulos y configurarlo usando los comandos y paquetes de datos que se deberían enviar desde un microcontrolador, el funcionamiento de una red de varios módulos, enviar y recibir datos entre módulos en una red, así como configurar los módulos de RF.

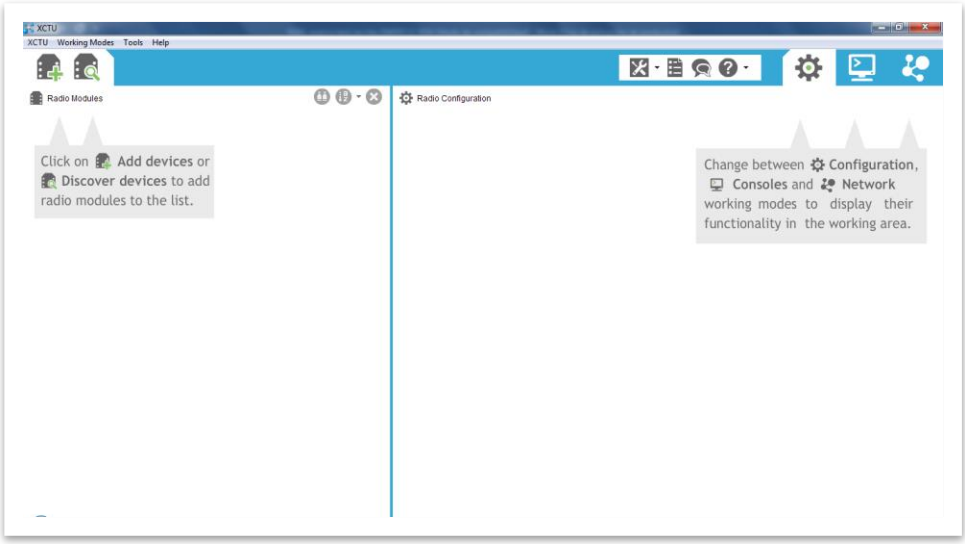


Fig. 112. Entorno de trabajo XCTU

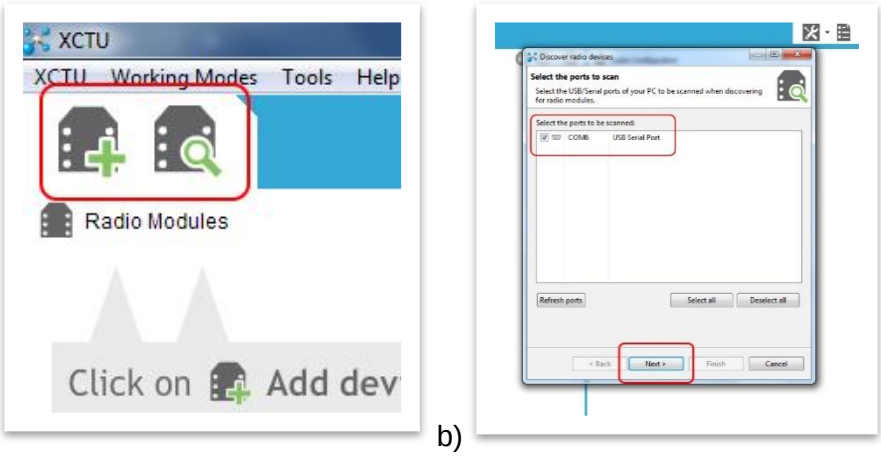
Configuración de XBee's

Para la configuración básica de los módulos es necesario contar con los siguientes materiales:

XBee's de la misma serie
Adaptador XBee Explorer USB
XCTU (software)

Pasos para la configuración.

Empleando el XBee #1.
Abrir una ventana de la terminal X-CTU.
Conectar el Xbee a la computadora por medio del adaptador.
Agregar o buscar el dispositivo en la terminal XCTU, seleccionando alguno de los dos iconos indicados en la figura 113 (a). En la ventana que aparece, seleccionar el puerto USB en donde está conectado el XBee mediante el adaptador y dar click en siguiente, fig. 113 (b).



a) b)

Fig. 113. Iconos para agregar o buscar Modulo XBee conectado y su puerto.

En la siguiente ventana se configuran los parámetros a los cuales se trabajará el modulo, los cuales deberán ser iguales para cada uno, mismos que son ingresados en el programa del microcontrolador para la comunicación serial.

Velocidad de baudios	9600
Bits de datos	8
Paridad	Ninguna
Bits de parada	1
Control de flujo	Ninguno

Tabla 14. Datos para configurar comunicación en módulos XBee.

Una vez seleccionados los parámetros dar click en finalizar, (fig. 114).

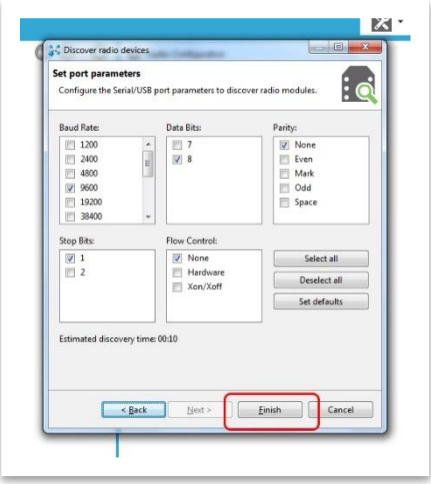


Fig. 114. Ventana para configurar los parámetros del módulo.

El programa comenzara a buscar el dispositivo y una vez localizado mostrara otra pantalla donde aparece el dispositivo localizado, enseñando el puerto de conexión, nombre del módulo y su dirección, para continuar se da click en agregar dispositivo, (fig. 115).

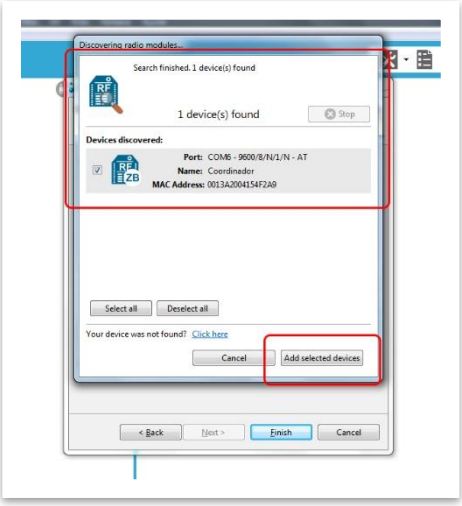


Fig. 115. Dispositivo localizado y listo para agregar.

Una vez agregado el dispositivo, aparecerá en la parte derecha del entorno, al cual se le da click para que del lado izquierdo nos muestre los parámetros que contiene el modulo, (fig. 116).

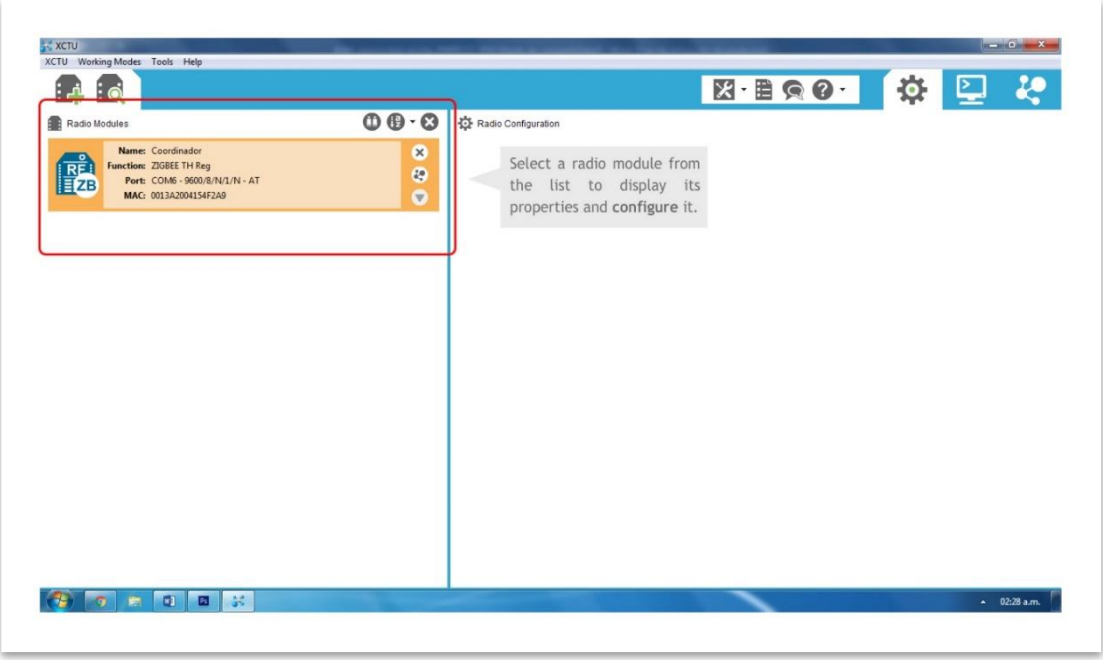


Fig. 116. Dispositivo agregado.

Al hacer click sobre el dispositivo, la pantalla muestra el mensaje de proceso de lectura de los parámetros.

Luego los coloca en la parte derecha del entorno, (fig. 117). Para visualizar todos los parámetros movemos la barra de desplazamiento ubicada en la derecha.

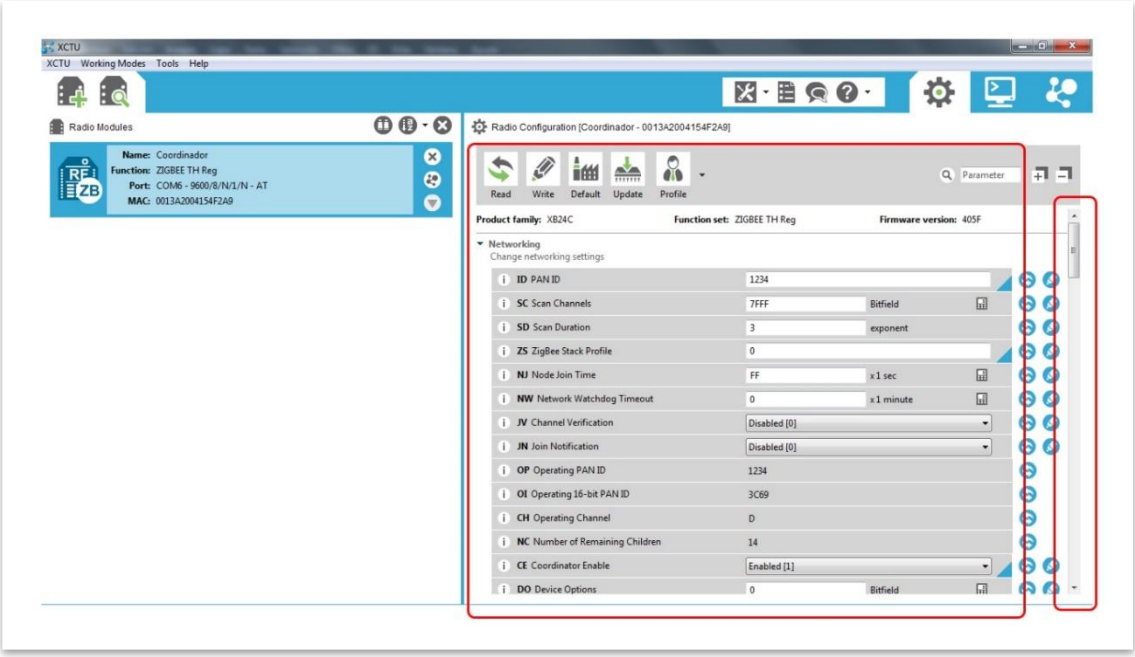


Fig. 117. Lista de parámetros existentes en el módulo conectado.

Existen muchos parámetros a configurar, pero los esenciales para crear la red tipo estrella son cinco:

ID PAN ID	Numero de red a la que se conectaran los módulos.
CE Coordinator Enable	Habilitar o deshabilitar modulo como coordinador
DH Destination Address High	Dirección de destino alta para transmitir los datos
DL Destination Address Low	Dirección de destino baja para transmitir los datos
NI Node Identifier	Define el identificador de nodo, un nombre amigable para el usuario para el módulo.

Tabla 15. Parámetros esenciales para crear red.

– Todos los XBee's deben compartir el mismo identificador. Es el caso de la red el numero asignado será 1234.
– Solo debe existir un Coordinador en la red. Por lo tanto, solo un módulo tendrá esta opción habilitada (Enable [1]), en los otros cuatro deberá quedar deshabilitada, (Disable [0]).
– La dirección alta será la misma para los módulos, tanto para el Coordinador como para los End Devices, ya que se trata de una red tipo estrella, su valor será cero "0".
– En la dirección baja se colocará el valor "65535" en hexadecimal, o sea, "FFFF" para el Coordinador y para todos los End Devices se escribirá el mismo valor, el cual es cero "0".
– En el identificador de nodo se escribirá el nombre de la estación de trabajo a la que pertenece, por ejemplo, el módulo XBee conectado a la interfaz se llamara "Coordinador", y para los otros cuatro se le asignar el nombre de "Agente X", donde "X" representa el número de estación que va desde la 1 a la 5, contando el modulo del brazo robótico.

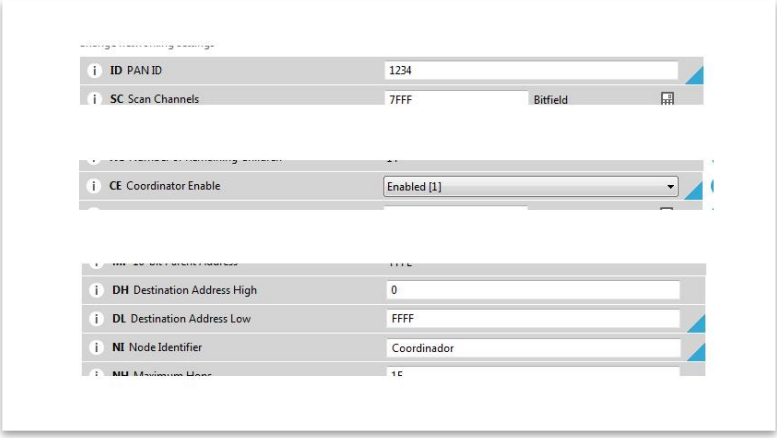


Fig. 118. Parámetros a modificar para crear tipo estrella.

Las modificaciones que se van realizando en los parámetros se pueden identificar por medio del triángulo que se ubica en el costado derecho de función. El triángulo se muestra de color azul cuando no se ha modificado, y cambia a color verde cuando se ha cambiado el valor, fig. 119.

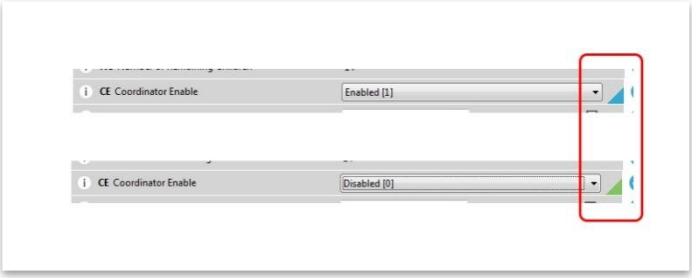


Fig. 119. Identificación de parámetros modificados antes y después.

Una vez escrito los valores en las casillas correspondientes al parámetro a cambiar, se presiona el icono de escritura (Write), fig. 120, para grabar los nuevos valores en el módulo XBee conectado.

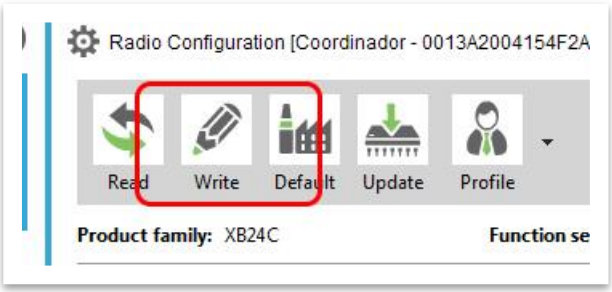


Fig. 120. Botón para grabar los datos modificados.

A continuación, se muestra el mensaje de grabación de parámetros en la pantalla. Y cuando finalice, los datos habrán sido cambiados y el módulo XBee tendrá los valores asignados para la red a utilizar.

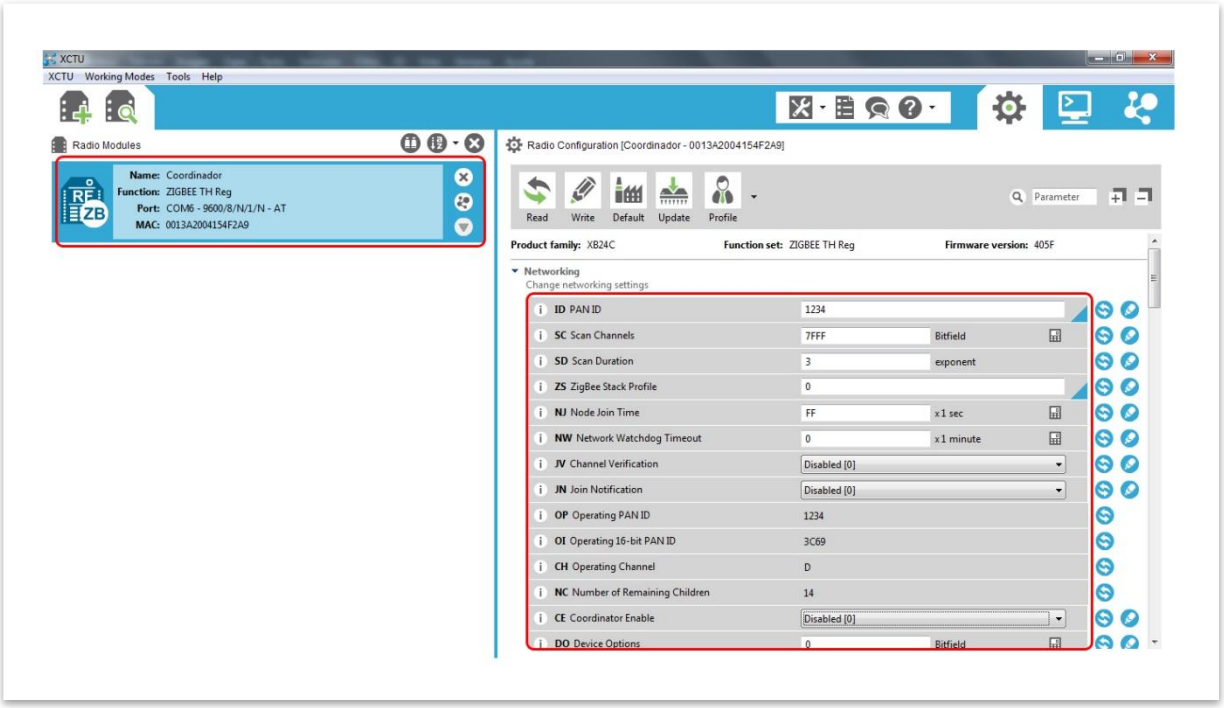


Fig. 121. Módulo XBee con valores asignados como Coordinador.

De igual modo para grabar los módulos End Devices se siguen los pasos anteriores, en este caso algunos los valores a modificar cambian.

- La dirección de red PAN ID es la misma que la del coordinador, la cual es “1234”.
- En la casilla CE Coordinator Enable cambia y ahora se marca como deshabilitado (Disable”).
- La dirección alta también es la misma que el coordinador, en este caso cero, “0”.
- La dirección baja cambia y para todos los End Devices el valor es cero, “0”.
- En el Nodo de identificación se escribe le nombre “Agente X”, donde “X” es el número de estación donde pertenece.

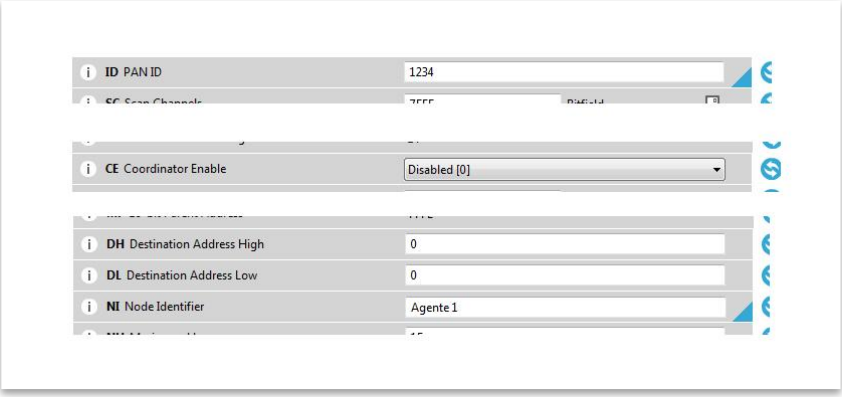


Fig. 122. Parámetros a modificar para los módulos de las estaciones de trabajo.

Verificación de módulos configurados

Una vez grabado todos los módulos se realiza una prueba con la ayuda del software XCTU.
Se conectan el modulo coordinador y algún módulo de estación de trabajo a la computadora.
Se abren entornos XCTU, en cada ambiente se busca un dispositivo, una ventana será para abrir el coordinador y la otra será para abrir el módulo de la estación.
Se agregan los dispositivos con las configuraciones mencionadas en la tabla 5.

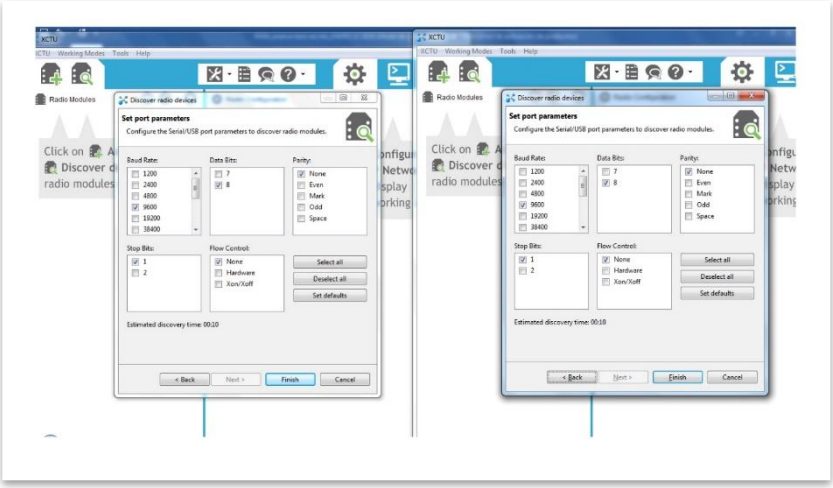


Fig. 123. Configuración de parámetros para ambos dispositivos.

Una vez agregados se presiona el botón de monitor serial, en ambas ventanas, el cual se encuentra en la parte superior derecha del entorno.

Fig. 124. Botón para abrir ventana de monitor serial.

Se despliega la pantalla del monitor serial, la cual permite el envío y recepción de datos entre los dispositivos agregados.

Enseguida se hace click en el icono de conexión “open”, el cual abrirá la conexión con el modulo, pasando de color gris a verde, indicando que ya se estableció la conexión con el dispositivo.

Fig. 125. Botón para abrir conexión.

Cuando la conexión en ambos lados se realice, se presiona el botón de “+”, para mostrar la ventana para agregar paquete de datos en la lista.
En la ventana se puede escribir caracteres en código ASCII o Hexadecimal, y una vez escritos se presiona agregar paquete para colocarlo en la lista.

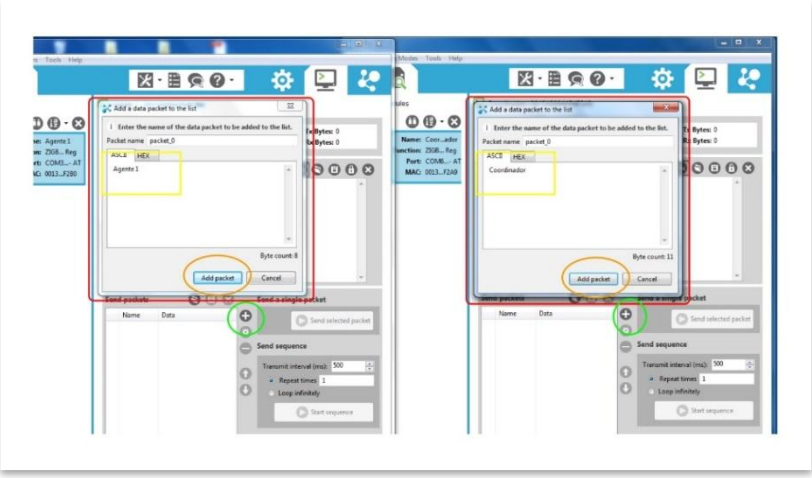


Fig. 126. Ventana para agregar paquete de datos en la lista

Cuando los paquetes son agregados, estos se muestran en la lista. Para poder enviarlos se selecciona y se presiona el botón enviar “Send selected packet”.
Cada vez que se presiona el botón, se envía el paquete seleccionado al otro módulo, visualizándose en el monitor serial del dispositivo contrario.
El paquete enviado del coordinador se notará en la pantalla del agente conectado, y el paquete enviado por el agente se verá reflejado en la pantalla del coordinador.

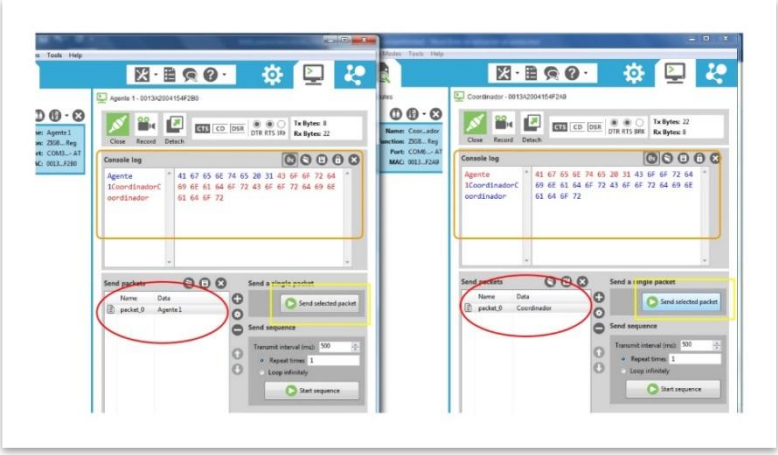


Fig. 127. Envío de datos entre dos módulos XBee.

Con esta prueba se verifica que los módulos están configurados de la manera correcta para crear la red tipo estrella en el proyecto a desarrollar.
Se pueden conectar varios módulos XBee a la computadora, dependiendo claro del número total de puertos USB disponibles con que se cuente. Y así poder interactuar entre ellos abriendo una ventana XCTU para cada uno y agregándolo al entorno correspondiente. Los mensajes se verán reflejados en todas las pantallas, debido a que el coordinador envía la señal a todos los módulos End Devices.

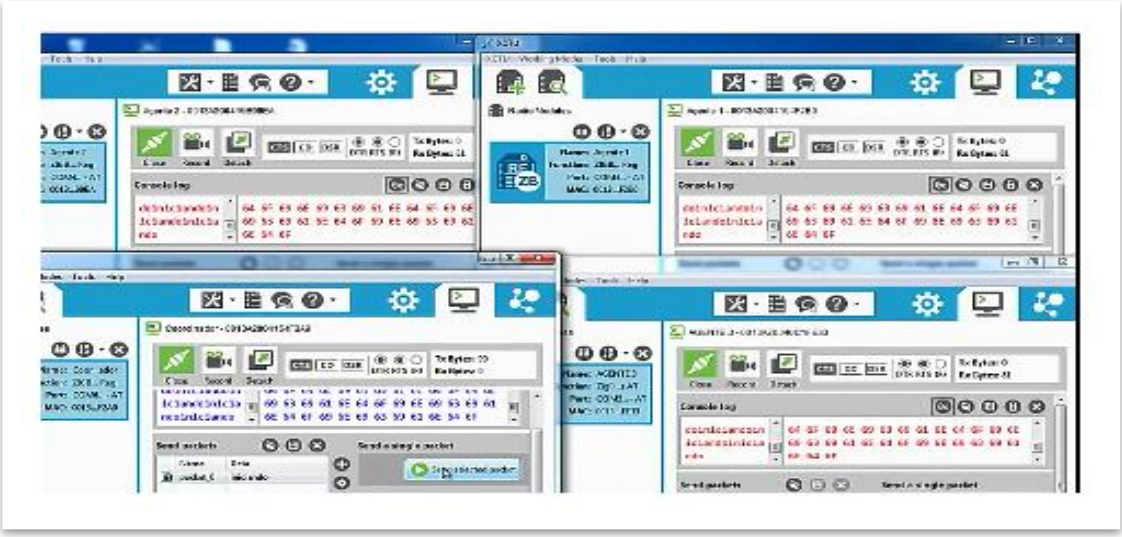


Fig. 128. Envío de datos entre cuatro módulos XBee, coordinador y agentes.

Labot Pro 5.

La forma más sencilla de operar el brazo robótico es mediante el Software Labot Pro 5, incluido junto con el robot, que debe ser instalado en una Pc.
Se conecta el Robot a la computadora por medio un cable serial DB9-USB.
Después se ejecuta el Software Labot Pro 5
Se eligen los parámetros de comunicación, el puerto al que está conectado el brazo y la velocidad, la cual es de 9600 baudios, *la tarjeta recibe datos seriales solo a esta velocidad, este dato es muy importante porque establece la configuración de la red inalámbrica*, una vez seleccionadas las opciones se presiona la opción que dice Abrir Puerto.

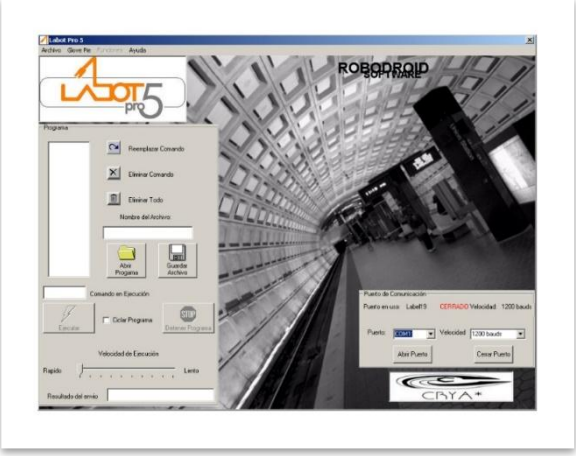


Fig. 129. Entorno software Labot Pro Ver5.

Si los datos están correctos se muestra el mensaje Abierto en letras color verde.

Fig. 130. Parámetros correctos, mensaje Abierto.

Se ejecuta la opción control Preciso del menú Funciones.
En la ventana que parece se muestran los motores del Brazo, permitiendo su manipulación con las barras de desplazamiento.
Cuando una barra se desplaza, cambia el número de la parte derecha, que es el valor enviado vía serial al Robot, moviendo el servomotor correspondiente, según el valor elegido.

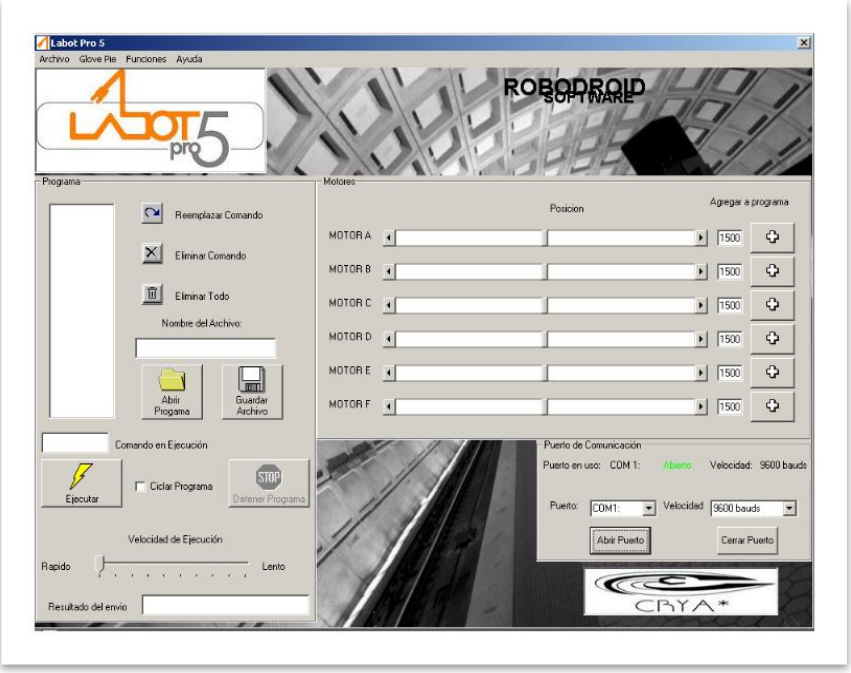


Fig. 131. Ventana para girar motores y crear secuencias.

Cada servomotor tiene asignada una letra, se debe tener en cuenta el rango de valores, máximo y mínimo, correspondientes a los límites de giro del motor.
Se pueden crear secuencias, agregando los valores del lado izquierdo, y una vez realizadas se presiona el botón de ejecutar, lo que hará que la secuencia se ejecute una vez por completo. Si se selecciona la opción ciclar ciclo, la secuencia se repetirá, cuando llegue al último dato, continuamente.

En el proyecto se utilizó este método para registrar los valores adecuados para los movimientos que el Brazo debe ejecutar y moverse al lugar preciso de la estación de trabajo.

Motor	Estación 1	Estación 2	Estación 3	Estación 4	Brazo Posición inicial
A	0500	0960	1620	1760	1277
B	1700	1520	1700	1650	1000
C	1630	1520	1750	1750	1000
D	0800	0850	0850	0851	1250
E	1235	1260	1260	1260	1260

Tabla 17. Valores registrados para colocar el brazo en estación de trabajo

Modo controlador Picaxe

Este método consiste en programar un microcontrolador tipo Picaxe. Es una familia de microcontroladores basados en los Pic, con firmware pre programado que habilita directamente el arranque de código directa, ente de un PC, simplificando el sistema de desarrollo embebido.
Consiste en realizar un programa que controle alguna de las 6 salidas para servos con las que cuenta la tarjeta se necesita utilizar el comando Serout a una velocidad de 2400 baudios, este comando envía un dato de 8 bit de tipo serial, sin paridad y con un bit de parada, es el único protocolo de comunicación aceptado por la tarjeta y se deben de enviar los datos por el pin 0 siempre, se puede apreciar un programa ejemplo, (fig. 132):

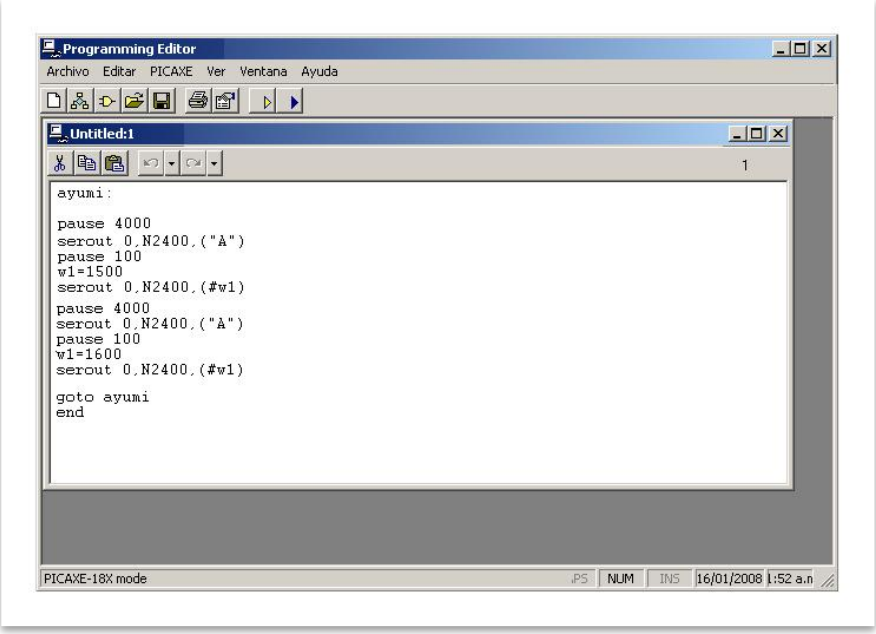


Fig. 132. Ejemplo de programa en Picaxe para mover motor del Brazo.

El funcionamiento del programa trabaja de la siguiente manera:

ayumi: Es un nombre de una rutina
pause 4000; Genera un retardo de 4 segundos
serout 0,N2400,(“A”); Envía una letra A mayúscula de manera serial por el pin 0 del PICAXE con el protocolo 8 bits de datos, sin paridad, con un bit de parada y la salida invertida.
pause 100; Genera un retardo de 100ms, antes de enviar el siguiente dato serial.
w1=1500; Se iguala la variable w1 a un valor de 1500, el cual es un valor de posición.

serout 0,N2400,(#w1); Se envía el valor numérico de la variable w1 vía serial, por el pin 0 del PICAXE
pause 4000; Retardo de 4 segundos
serout 0,N2400,("A"); Se envía una letra A mayúscula de manera serial por el pin 0 del PICAXE.
pause 100; Retardo de 100ms, antes de enviar el siguiente dato serial.
w1=1600; Igualar la variable w1 a un valor de 1500, el cual es un valor de posición.
serout 0,N2400,(#w1); Esta línea se encarga de enviar el valor numérico de la variable w1 vía serial, por el pin 0 del PICAXE
goto ayumi: Regresar al nombre de la rutina principal llamada ayumi, para ejecutar un ciclo permanente.
end; Este comando indica el final del programa y aunque el programa nunca cae en este comando es necesario ponerlo siempre en todos los programas.

El resultado sería el siguiente: el servo que está conectado en el M1 empezara a moverse solo cada 4 segundos cambia de la posición 1500 a la 1600 y mantiene el ciclo indefinidamente, para mover cualquiera de los otros motores es necesario utilizar la siguiente tabla donde se muestran los valores de posiciones que se pueden manejar junto con el motor que controlan:

Marca del motor.	Numero de motor	Dato Serial para enviar
M1	1	**A**
M2	2	**B**
M3	3	**C**
M4	4	**D**
M5	5	**E**
M6	**6**	**F**

Tabla 18. Valores asignados a cada motor del Brazo.

Esta tabla muestra la relación del motor que se va a mover con respecto a la letra que se le va a enviar serialmente, la tarjeta solo recibe para la selección del motor a mover las letras mayúsculas de la A hasta la F, no utilizar números ni letras minúsculas, ya que no funcionara correctamente.
El rango de valores en los que se puede desplazar cada motor es un rango numérico el cual comprende:

Valor Mínimo:	Valor Máximo:
0500	**2500**

Tabla 19. Rango de valores para desplazar Motor.

Donde 0500 es un extremo del movimiento y 2500 es el otro extremo.
Cuando Se envié un dato menor a 1000, como 600 es necesario enviar el dato con un cero antes es decir 0600, ya que la tarjeta espera un dato decimal de 4 dígitos, sin esto la tarjeta generara un error.

Con estos datos se observa que para mover un solo motor es necesario primero enviarle a la tarjeta la letra del motor que se va a mover y después se envía un valor numérico para determinar la posición.

ANEXO 2

COMPONENTES DE ESTACIONES DE TRABAJO

Brazo robótico

El brazo robótico que se utiliza en el proyecto fue desarrollado por la empresa CRYA (Control Robótica y Automatización), llamado Robot Labot Pro Ver5.
Consiste en un robot de alta tecnología, completamente articulado. Es un manipulador industrial, solo que a una escala menor. Utiliza servomotores especiales de alta capacidad para mover sus segmentos a la posición que se le programe, (32).

Fig. 133. Robot Labot Pro Ver5.

Ejes controlados	**5 GDL más la pinza.**
Largo del brazo	48 cm
Peso total del robot	10 Kg.
Dimensiones del robot	63cm *17cm*28cm
Medidas de objetos que puede tomar la pinza	3cm
Máxima carga de manipulación	½ Kg
Método de control	Punto por punto
Actuadores	3*18 V DC servomotores de lazo cerrado y 3*9 V DC slc
Fuente de alimentación	18 V DC 5 amp

Tabla 20. Características técnicas Robot Labot Pro Ver5

El brazo dispone seis grados de movimiento, cada uno dirigido por los servomotores que lo integran.

	Máximo rango de movimientos	Máxima Velocidad
Base	180 grados	180 grados /Seg
Hombro	180 grados	180 grados /Seg
Codo	180 grados	180 grados /Seg
Muñeca	180 grados	45 grados /Seg
Mano	90 grados	45 grados /Seg
Pinza	90 grados	Abierto/Cerrado en 1 Seg

Tabla 21. Parámetros de los motores del Robot Labot Pro Ver5

En la siguiente imagen se ilustran los conectores con su respectiva explicación detallada:

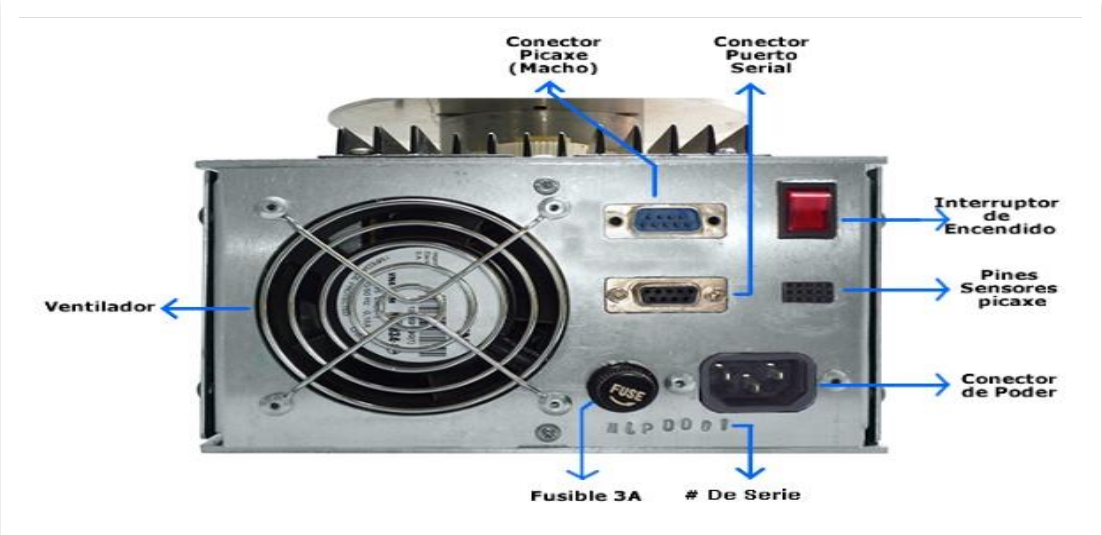


Fig. 134. Vista conectores Robot Labot Pro Ver5.

Ventilador: El ventilador esta siempre encendido de tal manera que cuando se alimenta el robot, este encenderá automáticamente, este tarda en estabilizarse alrededor de 1 minuto, tiempo en el cual no se debe accionar el Interruptor principal de encendido.

- Interruptor de Encendido: Este se encarga de arrancar el sistema Labot pro 5, cuenta con luz indicadora de encendido.
- Conector Picaxe: En este conector se introduce un cable serial 1 a 1 macho-macho, cuando se va a programar la unidad con el microcontrolador PICAXE 18X.
- Conector Puerto Serial: En este puerto se conecta un cale serial macho hembra, el cual sirve para controlar al Labot Pro vía el puerto serial de la máquina.
- Pines de Sensores Picaxe: Estos pines cuentan con la siguiente configuración:

Bits por segundo	9600
Bits de datos	**8**
Paridad	**Ninguna**
Bits de paridad	**1**
Control de flujo	**Xon/Xoff**

Tabla 22. Parámetros de comunicación serial RS-232

Modos de operación para brazo Robot Labot Pro Ver5

Esta tarjeta cuenta con 4 modos de operación, los cuales son:

a) Modo de comandos por puerto serial.
b) Modo de comandos por puerto serial con Wiimote por Aceleraciones.
c) Modo de comandos por puerto serial con Wiimote Preciso y con Teclado.
d) Modo de programación por microcontrolador Picaxe.

Microcontrolador

Un microcontrolador es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. Estas partes están interconectadas dentro del microcontrolador, y en conjunto forman lo que se le conoce como microcomputadora. Se puede decir que un microcontrolador es una microcomputadora completa encapsulada en un circuito integrado.

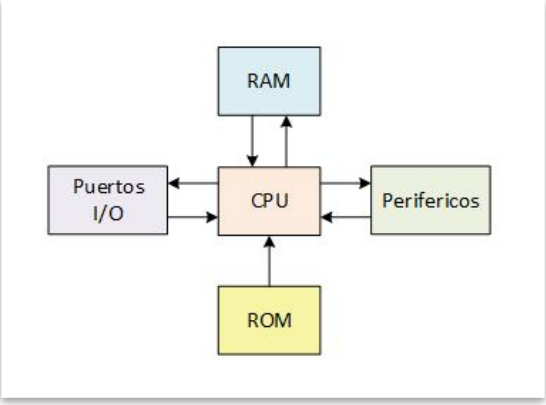


Fig. 135. Esquema de microcontrolador

Toda microcomputadora requiere de un programa para que realice una función específica. Este se almacena normalmente en la memoria ROM.
El propósito fundamental de los microcontroladores es el de leer y ejecutar los programas que el usuario le escribe, la programación es una actividad básica e indispensable cuando se diseñan circuitos y sistemas que los incluyan. Los caracteres programables de los microcontroladores simplifican el diseño de circuitos electrónicos. Permiten modularidad y flexibilidad, ya que un mismo circuito se puede utilizar para que realice diferentes funciones con solo cambiar el programa del microcontrolador.
Las aplicaciones de los microcontroladores son extensas, es común encontrar microcontroladores en campos como la robótica y el automatismo, en la industria del entretenimiento, en las telecomunicaciones, en la instrumentación, en el hogar, en la industria automotriz, etc.
Están diseñados para interpretar y procesar datos e instrucciones en forma binaria. Patrones de 1's y 0's son los que conforman el lenguaje máquina de los microcontroladores, y es lo único que son capaces de entender. Estos 1's y 0's representan la unidad mínima de información, conocida como bit, ya que solo puede adoptar uno de dos valores posibles: 0 y el 1.
La representación de datos, instrucciones y señales en forma de bits resulta dificultosa y tediosa, no resulta tan evidente la interpretación de instrucciones en forma binaria o lenguaje máquina (el lenguaje maquina se le conoce también como lenguaje de bajo nivel debido a que las instrucciones no son propias del lenguaje humano). Es por esto que la programación comúnmente se lleva a cabo en un lenguaje de alto nivel, es decir, un lenguaje que utilice frases o palabras semejantes o propias del lenguaje humano. Las sentencias de los lenguajes de alto nivel facilitan enormemente la programación ya que son familiares a nuestra manera de comunicarnos. Lenguajes como el C o BASIC son comúnmente utilizados en la programación de microcontroladores.
Otro tipo de lenguaje más especializado es el lenguaje ensamblador. El lenguaje ensamblador es una lista con un limitado número instrucciones a los cuales puede responder un microcontrolador. Estas instrucciones son palabras o abreviaciones que representan las instrucciones en lenguaje máquina del microcontrolador.
Las instrucciones en lenguaje ensamblador, son conocidas como nemotécnicos, y son fáciles de entender con ellos se permite operar directamente con los registros de memoria, así como con las instrucciones intrínsecas del microcontrolador. Cualquiera que sea el lenguaje que se utilice en la

programación de microcontroladores, es de lo más recomendable profundizar en su arquitectura interna, ya que con este conocimiento se pueden aprovechar más y mejor las capacidades de un microcontrolador dado.
Todo programa escrito en un lenguaje de alto nivel debe ser transformado en código máquina, para que puedan ser entendidos por el microcontrolador.
Un software de computadora, llamado compilador, traduce y transforma el programa en código máquina, que es lo que realmente puede leer e interpretar el microcontrolador.
Una vez compilado el programa, es momento de transferir el código maquina hacia la memoria interna del microcontrolador, usualmente hacia la ROM. Para esta tarea se utiliza un programador físico, que es una pieza de hardware que tiene el propósito de escribir el programa en la memoria interna del micro.

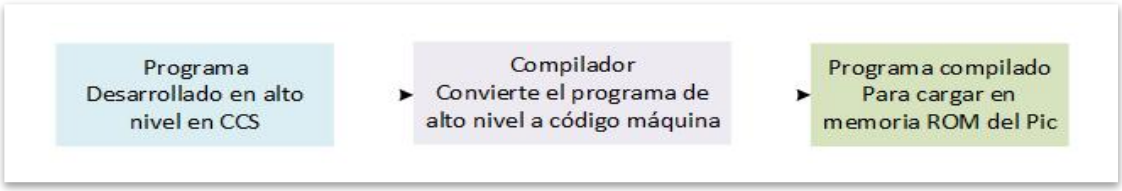


Fig. 136. Compilación para microcontrolador

Existen varios fabricantes de microcontroladores tales como Texas Instruments, Motorola, Atmel, Intel, Microchip, Toshiba, Nacional, etc. Todos ellos ofrecen microcontroladores con características más o menos similares, sin embargo, en términos generales se puede decir que todos sirven para lo mismo: leer y ejecutar los programas del usuario.
La diferencia radica en los modelos ya que pueden tener más capacidad que otros, en cuanto a memoria, velocidad, periféricos, etc.
Para la selección del microcontrolador adecuado se fijó la disponibilidad en el mercado local. Algunos microcontroladores son más comúnmente encontrados en las casas de electrónica que otros. No es conveniente emprender un proyecto basado en cierto microcontrolador que escasea en el mercado local, ya que podría no satisfacer nuestra demanda y detener el proyecto. La disponibilidad de información y herramientas de desarrollo. La mayoría de los fabricantes de microcontroladores ofrecen información suficiente para entender la operación y funcionamiento de sus dispositivos. El punto débil de algunos fabricantes es la pobre distribución de sus herramientas de desarrollo (programadores, emuladores, software, etc.) o bien su alto costo. Así como el precio, que resulta un punto a favor en la selección de un fabricante de microcontroladores.
El modelo de microcontrolador específico que se debe elegir dependiendo de la aplicación, tomando en cuenta su capacidad de memoria, la cantidad de puertos de entrada y salida, los periféricos, la velocidad a la cual ejecuta las instrucciones, etc.

MICROCONTROLADORES PIC®

Los PIC, de Microchip, son una opción y son fáciles de encontrar, existe alta disponibilidad en el mercado y a un bajo precio. El fabricante provee la información relativa a sus productos.
Se han desarrollado una serie de herramientas de bajo costo por parte de terceros (empresas, profesionales y aficionados), como son programadores, software, etc., que facilitan el uso y programación de estos dispositivos. Compiladores de C y Basic están disponibles para programar a los PIC.

El proyecto presentado utiliza el microcontrolador Pic 16F876A. Se eligió este modelo de entre varios por poseer 256 bytes de memoria EEPROM, 22 pines configurables para entrada o salida digital, 5 canales para conversión análogo-digital, 2 entradas para modulación por ancho de pulsos, además de contar con comunicación USART. Los pines de este microcontrolador que trabajan con el módulo USART PIC son el pin RX o pin receptor y el pin TX o pin transmisor.

Si la comunicación USART PIC es asíncrona, uno de los hilos será para la transmisión de los datos de un dispositivo a otro y el otro hilo será para la recepción de datos entre un dispositivo a otro, la transmisión y la recepción pueden ocurrir en forma simultánea, lo que si se tiene que cumplir es que la frecuencia de trabajo de ambos dispositivos tiene que ser la misma, a esto se le conoce como los baudios que viene a ser la cantidad de bits por segundo que se transmitirán entre ambos dispositivos, (19).

Fig. 137. Pic 16F876A

Módulo XBee

Los XBee´s son pequeñísimos chips azules capaces de comunicarse de forma inalámbrica unos con otros. Pueden hacer cosas simples, como reemplazar un par de cables en una comunicación serial.
Existen muchos tipos diferentes de módulos, una de las ventajas de estos XBee, es que todos, independiente del modelo o serie, tienen los pines similares. Alimentación, tierra y los pines de comunicación (TX/RX) se encuentran en el mismo lugar, haciendo que los chips sean totalmente intercambiables, para la mayoría de las aplicaciones más simples, (30).
La manera de configurar los dispositivos es mediante software, el cual se menciona más adelante, donde se definen las características para la comunicación que se desea, entre las cuales esta, la velocidad de transmisión, paridad, entre otras.

Fig. 138. Módulo XBee serie 2

Adaptador Xbee Explorer USB

Este adaptador permite programar módulos XBee, además de establecer comunicación entre la computadora y otro módulo a distancia. Se conecta al puerto USB de la computadora. Funciona con todos los módulos de XBee: Series 1 y 2.5 en versiones estándar y pro.

Fig. 139. Adaptador XBee Explorer USB.

Incorpora un regulador de voltaje de 1A lo que admite emplear módulos XBee Pro sin problemas de alimentación. También posee un botón de reset para reiniciar el XBee tanto en operación como en modo de configuración.
Entre sus características de operación se encuentran:

Voltaje	**5VDC del USB o 3.3VDC generados por la Board**
Comunicación	Serial pass-through to XBee module/USB to Host PC
Temperatura	-40°C a +70°C

Tabla 23. Características adaptador XBee Explorer USB, (31).

Motor DC

Un motor de corriente continua convierte la energía eléctrica en mecánica. Se compone de dos partes: el estator y el rotor.

El estator es la parte mecánica del motor donde están los polos del imán.

El rotor es la parte móvil del motor con devanado y un núcleo, al que llega la corriente a través de las escobillas.

Cuando la corriente eléctrica circula por el devanado del rotor, se crea un campo electromagnético. Este interactúa con el campo magnético del imán del estator. Esto deriva en un rechazo entre los polos del imán del estator y del rotor creando un par de fuerza donde el rotor gira en un sentido de forma permanente.

Si queremos cambiar el sentido de giro del rotor, tenemos que cambiar el sentido de la corriente que le proporcionamos al rotor; basta con invertir la polaridad de la pila o batería.

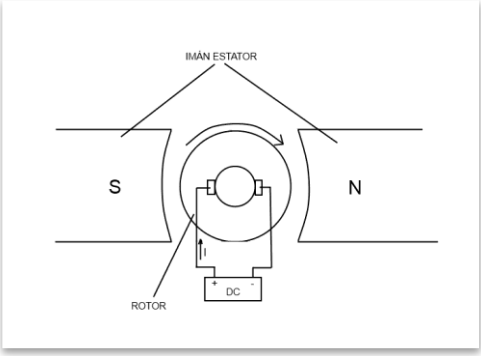


Fig. 140. Esquema general motor DC, (33).

Módulo L298N

Es un módulo para manejo de motores, un driver dual para motores (full-Bridge), es un integrado para controlar motores DC que usa el sistema puente en H. Un sistema para controlar el sentido de giro de un motor DC usando cuatro transistores. Los transistores se comportan como interruptores y dependiendo que transistores conducen y cuáles no cambia la polarización del motor, y con esto el sentido de giro, (34).

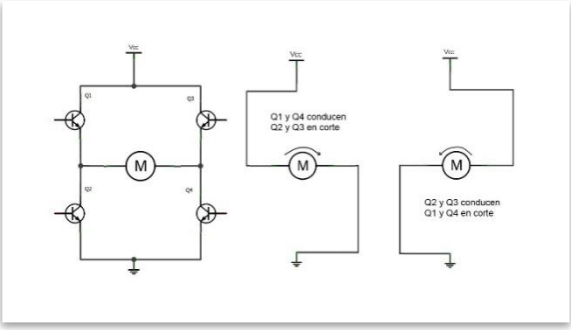


Fig. 141. Esquema general motor DC controlado con puente H, (33).

El módulo cuenta con todos los componentes necesarios para funcionar sin necesidad de elementos adicionales, entre ellos diodos de protección y un regulador LM7805 que suministra 5V a la parte lógica del integrado L298N.

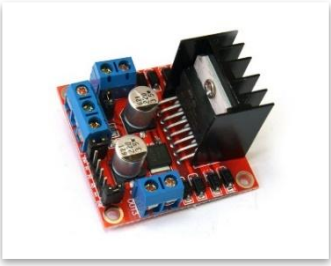


Fig. 142. Módulo L298N para motores.

Por medio de los jumpers de selección se habilitan cada una de las salidas del módulo (A y B). La salida A está conformada por OUT1 y OUT2, mientras que la salida B por OUT3 y OUT4.
Los pines de habilitación son ENA y ENB respectivamente. En la parte inferior se encuentran los pines de control del módulo, marcados como IN1, IN2, IN3 e IN4.

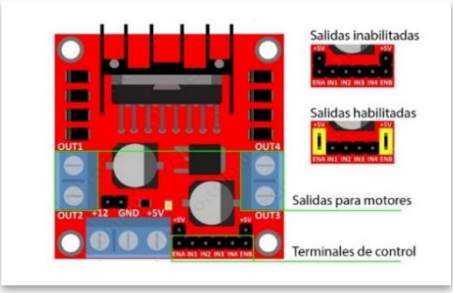


Fig. 143. Conexiones módulo L298N.

Este módulo se puede alimentar de 2 maneras gracias al regulador integrado LM7805.

Cuando el jumper de selección de 5V se encuentra activo, el módulo permite una alimentación de entre 6V a 12V DC. Como el regulador se encuentra activo, el pin marcado como +5V tendrá un voltaje de 5V DC. Este voltaje se puede usar para alimentar la parte de control del módulo por ejemplo un microcontrolador, el consumo no debe ser mayor a 500 mA.
Cuando el jumper de selección de 5V se encuentra inactivo, el módulo permite una alimentación de entre 12V a 35V DC. Como el regulador no está funcionando, se conecta el pin de +5V a una tensión de 5V para alimentar la parte lógica del L298N. Usualmente esta tensión es la misma de la parte de control (microcontrolador).

Control de un motor DC variando su velocidad

Para controlar la velocidad del motor, se hace uso de PWM (Modulación por ancho de pulsos). Este PWM será aplicado a los pines de activación de cada salida o pines ENA y ENB respectivamente, por tanto, los jumpers de selección no son usados.
En la fig. 49 se muestra la conexión de un motor en la salida B. los pines in3, in4 y ENB son conectados a los pines de control del microcontrolador.

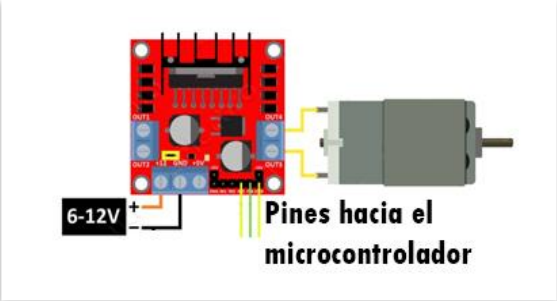


Fig. 144. Conexión de motor en salida B.

Relevador

El relé o relevador es un dispositivo electromagnético. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes, (35).

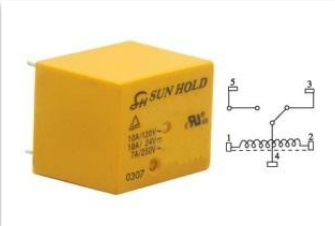


Fig. 145. Relevador bobina 5V.

Modulo cuatro relevadores

Se trata de un módulo de 4 relés que funcionan a 5 Voltios, capaces de manejar cargas de hasta 10 Amperes en 250 Voltios, aislados mediante optoacopladores de las entradas, las que cuentan con leds individuales que sirven como indicadores de estado, (36)

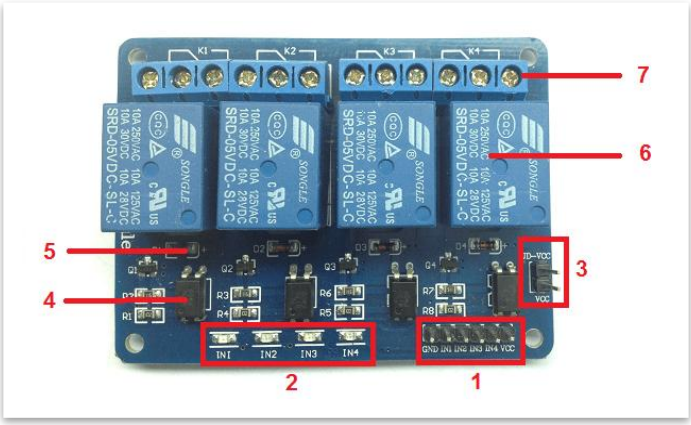


Fig. 146. Partes del módulo 4 relevadores.

La placa tiene un conector de entradas (IN1 a IN4) y alimentación (GND es masa o negativo y Vcc es el positivo).
Cuatro leds que indican el estado de las entradas.
Un jumper selector para la alimentación de los relés.
Cuatro optoacopladores del tipo FL817C.
Cuatro diodos de protección.
Cuatro relés con bobinas de 5V y contactos capaces de controlar hasta 10 Amperes en una tensión de 250V.
Cuatro borneras, con tres contactos cada una (Común, Normal abierto y Normal cerrado), para las salidas de los relés.

Esquemático módulo cuatro relevadores

En la fig. 56 se puede apreciar el circuito esquemático de un canal, el resto de los canales repite la misma configuración.

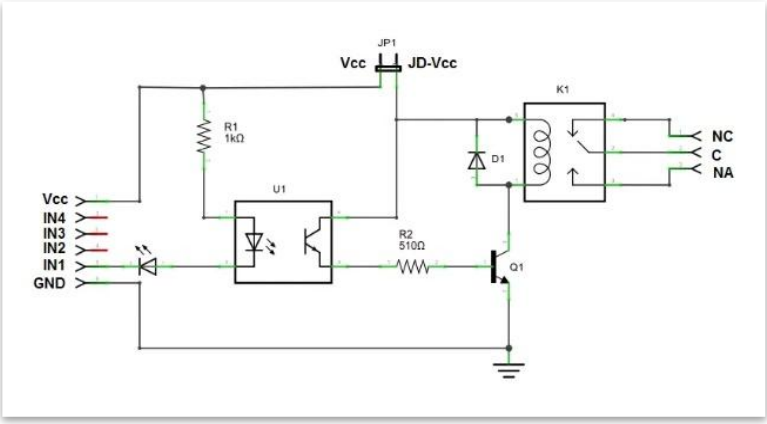


Fig. 147. Esquemático de canal relevador.

Funcionamiento módulo cuatro relevadores

La entrada IN1 está conectada al cátodo del diodo del optoacoplador a través del led indicador. El ánodo del diodo del optoacoplador se conecta a Vcc (positivo) por intermedio de R1, una resistencia de 1000 ohms. Estos tres componentes, el diodo indicador, el diodo del opto y la R1 forman un

circuito serie por el cual circula la corriente cuando la entrada está a un nivel BAJO (conectada a GND) y no circula si la entrada está a un nivel ALTO (conectada a Vcc).
El transistor del opto tiene su colector a JD-Vcc y su emisor conectado a Q1 a través de una resistencia de 510 ohms. Este es otro circuito serie por el cual circula corriente cuando el transistor del opto conduce al ser “iluminado” por su diodo, con lo que se introduce corriente en la base de Q1 a través de R2.
Finalmente, Q1 está conectado en una típica configuración emisor común, con su emisor a masa (GND) y la bobina del relé como carga en el colector. Cuando circula corriente por la base desde el opto, Q1 se satura permitiendo el paso de la corriente a través de la bobina del relé, lo que produce que se cierren los contactos del mismo (común con normal abierto). El diodo D1 protege al transistor de la tensión que aparece en la bobina del relé cuando deja de circular corriente por la misma.

Al ponerse la entrada **a nivel BAJO** se pone a la saturación el transistor Q1 a través del optoacoplador con lo que se cierra el contacto normal abierto del relé.

Las entradas a la placa pueden conectarse directamente a las salidas digitales del microcontrolador.

Alimentación y consumo módulo cuatro relevadores

La forma más segura es remover el jumper y alimentar la placa de relés con dos fuentes: una fuente para el microcontrolador conectada a Vcc y una segunda fuente, con el positivo a JD-Vcc y el negativo a GND, sin estar éste unido a la primera. Esta conexión tiene como ventajas:
Hay completa aislación entre la carga y el microcontrolador.
Todo el consumo de los relés es tomado de la segunda fuente y no de la fuente del microcontrolador.

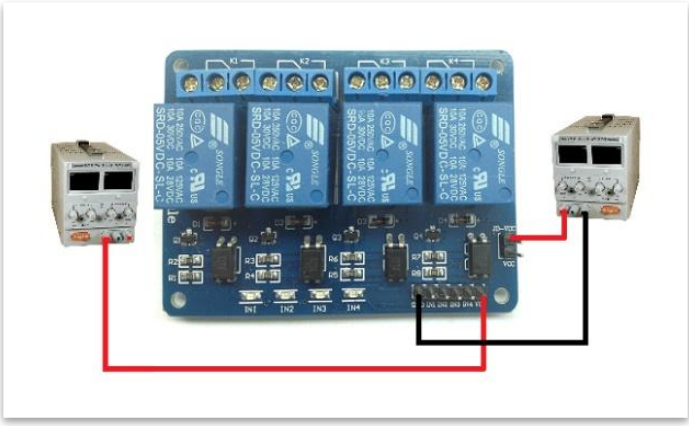


Fig. 148. Fuente externa para alimentación de módulo.

Display LCD 16x2

Un display LCD (Liquid Crystal Display) es un display alfanumérico de matriz de puntos (dot-matrix) que sirve para mostrar mensajes a través de caracteres como letras, números o símbolos. La placa del display viene equipado con un microcontrolador (normalmente el modelo HD44780) que se encarga de generar los caracteres, polarizar la pantalla, desplazar el cursor, etc. También viene equipado con una memoria ROM donde están almacenados los caracteres a través de una matriz de puntos, y una memoria RAM donde se pueden almacenar caracteres creados por el programador.

Estos displays disponen de unos pins para conectar un microcontrolador para poder dar instrucciones al display.
Existen displays de distintos tamaños de pantalla. En el proyecto se usa un display 16×2, que quiere decir que contiene 16 columnas y 2 filas.(37).

Fig. 149. Display LCD 16x2.

Conexiones display LCD

Los display tienen 16 pines (14 si la pantalla no se retroiluminada). Cada pin esta designado para realizar una función específica.
Pins 1-2: Alimentación +5V
Pin 3: Voltaje para el contraste de la pantalla +5V, con un potenciómetro se puede regular.
Pin 4: RS (Register Select) controla la memoria del display.
Pin 5: R/W (Read/Write) Selecciona modo escritura o modo lectura.
Pin 6: Enable. Habilitar o deshabilitar la escritura del display.
Pins 7-8-9-10-11-12-13-14: Bus de datos de 8 bits.
Pins 15-16: Habilitar retroiluminación aplicando +5V.

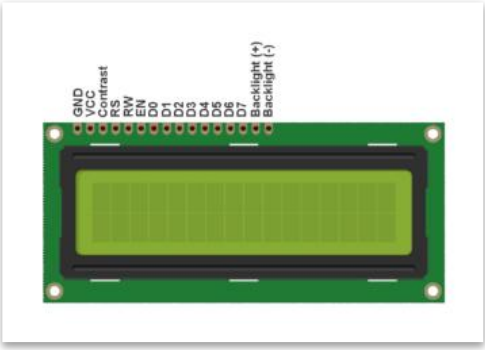


Fig. 150. Pines de conexión Display LCD 16x2.

El bus de datos se puede trabajar a 8 o 4 bits. El compilador C incluye un fichero (driver) que permite trabajar con un LCD. El archivo es LCD.C y debe llamarse como un #include. Este archivo dispone de varias funciones ya definidas.

Bomba de agua

Una bomba de agua es una máquina hidráulica que permite incrementar la energía cinética de un caudal de agua.

Las bombas hidráulicas son elementos ampliamente conocidos y empleados en la industria y constituyen toda una rama de la técnica. Existe una gran variedad de bombas, que abarcan un amplio rango de potencias y características hidráulicas. Industrialmente, forman parte de un sin fin de equipamientos e impulsan todo tipo de fluidos.

Fig. 151. Mini bomba de agua para microcontrolador.

Independientemente de sus características o potencia, siempre podemos controlar un equipo de bombeo mediante un procesador, siendo de hecho frecuente que estén controlados por un autómata. Mediante el microcontrolador Pic podemos encender cualquier tipo de bomba de agua mediante las salidas digitales y el uso de un MOSFET o una salida por relé.

Una bomba de agua consta de un accionamiento, en la mayoría de los casos un motor eléctrico, acoplado a un elemento rotativo denominado rodete.

El rodete está formado por álabes que, al girar, transmiten parte de la energía al fluido que lo atraviesa. Normalmente los álabes están curvados formando una guía para las partículas, y su forma determina la cantidad de energía que se transmite al fluido y el grado en que esta se reparte entre velocidad o presión.

Sin embargo, en algunas bombas de muy pequeña potencia los álabes son rectos, formando una simple aspa.

En las bombas axiales, el agua entra en la bomba por el centro del rodete, incrementa su energía a medida que lo atraviesa girando con el mismo, y finalmente abandona la bomba en sentido tangencial.

Fig. 152. Interior bomba de agua.

Existen bombas que emplean otro tipo de fluidos, en lugar de agua. Por ejemplo, existen bombas para mover hidrocarburos, aceites, o disoluciones. Sin embargo, aunque la teoría dice que cualquier máquina hidráulica puede funcionar al cambiar el fluido que impulsa, en el mundo real deberemos comprobar en las especificaciones que la bomba está preparada para el tipo de fluido que vamos a emplear.
También podemos clasificar las bombas entre sumergibles y bombas no sumergibles.

Las bombas sumergibles el motor se encuentra sellado en un encapsulado, por lo que toda la bomba se introduce en el fluido, lo que evita tener una tubería de aspiración.

En las bombas no sumergibles el motor no está impermeabilizado, por lo que no puede ser introducido dentro del fluido. Por tanto, necesitan una tubería de admisión, que debe cumplir unas determinadas de condiciones para que la bomba funcione correctamente, (38).

Ventilador

Un ventilador es una máquina hidráulica que incrementa la energía cinética de un gas, normalmente aire.

Un ventilador dispone de un rotor con aspas accionado, habitualmente, por un motor eléctrico. Al rotar, las aspas impulsan las moléculas de aire aumentando su energía cinética sin apenas variación de volumen, motivo por el cual se consideran máquinas hidráulicas en lugar de turbo máquinas.

En el campo de la electrónica es frecuente emplear ventiladores como los utilizados en informática para disipación de calor en ordenadores. Estos ventiladores se encuentran disponibles en una gran variedad de tamaños, siendo frecuentes 40x40mm, 80x80mm y 120x120mm. La tensión de alimentación normalmente es 12V.

Industrialmente los ventiladores se emplean en todo tipo de aplicaciones, como control de temperatura, instalaciones de climatización, homogenización de gases, evacuación de humos, refrescamiento de maquinaria, o disipación de calor en sistemas de refrigeración.(39)

Fig. 153. Ventilador 12V.

ANEXO3

PROGRAMAS PARA ESTACIONES DE TRABAJO

Programa estación 1

```
#include <16f876a.h>                                    //Seleccionamos Pic a usar
#fuses hs,nowdt                                         //Definimos crital, y fusibles
#use delay(clock=20M)                                   //Definimos velocidad de reloj a 20MHz
#use rs232 (baud=9600, xmit=pin_c6, rcv=pin_c7,bits=8,parity=N)       //Definimos parameetros de transmisión serial

char ch;                                                //Variable donde guardamos dato serial obtenido
int i;                                                  //Variable para ciclos for

int timer2, poscaler;                                   //Variables para el timer

#int_rda                                                //HAbilitamos Interrupción por serial
void serial_isr()                                       //Creamos función para interrupción
{
  ch=getc();                                            //Guardamos dato recibido en la variable "ch"
  delay_us(600);                                        //Espera 600 microsegundos
}

void main()
{
  //Configurar PWM
  timer2=255;                                           //Se carga el timer 2 con 255
  poscaler=1;                                           //Preescaler solo puede tomar valores de: 1
  setup_timer_2(t2_div_by_16,timer2,poscaler);                  //Configuracion de timer 2 para establecer frecuencia PWM
  setup_ccp1(ccp_pwm);                                  //Configurar modulo ccp1 en modo PWM

  set_tris_a(0b000000);                                 //Puerto a como salidas
  output_a(0b000000);                                   //Inicializar salidas en cero
  enable_interrupts(global);                            //Habilitar variables globales
  enable_interrupts(int_rda);                           //Habilitar la interrupcion
  printf("Agente 1: Banda Transportadora" );                    //Envío mensaje en monitor serial
  delay_ms(500);                                        //Esperamos un tiempo
  set_pwm1_duty(0);                                     //Valor del PWM=0

  while(1)                                              //Iniciamos ciclo while
  {
    if((ch=='a'))                                       //Si ch=a
    {                                                   //Entonces...
    output_low(pin_a0);                                 //Apagamos RA0
    output_low(pin_a1);                                 //Apagamos RA1
output_low(pin_a2);                                     //Apagamos RA2
    output_low(pin_a3);                                 //Apagamos RA3
    output_low(pin_c0);                                 //Apagamos RC0
    output_low(pin_c3);                                 //Apagamos RC3
set_pwm1_duty(0);                                       //Valor del PWM=0
}

    if((ch=='b'))                                       //Si ch=b
{                                                       //Entonces...
      for(i=0;i<20;i++)                                 //Iniciamos ciclo for para repeticiones
      {
        //Moviento de banda atras y velocidad lenta
output_high(pin_c0);                                    //Encendemos RC0
        output_low(pin_c3);                             //Apagamos RC3
set_pwm1_duty(125);                                     //Valor del PWM=90
output_high(pin_a0);                                    //Encendemos RA0
```

```
        output_low(pin_a1);                     //Apagamos RA1
        output_low(pin_a2);                     //Apagamos RA2
        output_low(pin_a3);                     //Apagamos RA3
delay_ms(700);                          //Espera tiempo
      }                                 //Despues de los ciclos...

output_low(pin_a0);                     //Apagamos RA0
      output_low(pin_a1);                     //Apagamos RA1
      output_low(pin_a2);                     //Apagamos RA2
      output_low(pin_a3);                     //Apagamos RA3
      output_low(pin_c0);                     //Apagamos RC0
      output_low(pin_c3);                     //Apagamos RC3
set_pwm1_duty(0);                       //Valor del PWM=0
      delay_ms(750);                        //Espera tiempo
      ch='p';                           //Asignar valor "p" a variable ch
      putc(ch);                         //Enviamos caracter ch=p
      delay_ms(300);                        //Espera tiempo
}

    if((ch=='c'))                       //Si ch=c
{                                   //Entonces...
      for(i=0;i<7;i++)                      //Iniciamos ciclo for para repeticiones
      {
        //Moviento de banda atras y velocidad rapida
output_high(pin_c0);                    //Encendemos RC0
        output_low(pin_c3);                     //Apagamos RC3
set_pwm1_duty(245);                     //Valor del PWM=245
output_low(pin_a0);                     //Apagamos RA0
        output_high(pin_a1);                    //Encendemos RA1
        output_low(pin_a2);                     //Apagamos RA2
        output_low(pin_a3);                     //Apagamos RA3
delay_ms(850);                          //Espera tiempo
      }                                 //Despues de los ciclos...

output_low(pin_a0);                     //Apagamos RA0
      output_low(pin_a1);                     //Apagamos RA1
      output_low(pin_a2);                     //Apagamos RA2
      output_low(pin_a3);                     //Apagamos RA3
      output_low(pin_c0);                     //Apagamos RC0
      output_low(pin_c3);                     //Apagamos RC3
set_pwm1_duty(0);                       //Valor del PWM=0
      delay_ms(750);                        //Espera tiempo
      ch='p';                           //Asignar valor "p" a variable ch
      putc(ch);                         //Enviamos caracter ch=p
      delay_ms(300);                        //Espera tiempo
}

    if((ch=='d'))                       //Si ch=d
{                                   //Entonces...
      for(i=0;i<20;i++)                     //Iniciamos ciclo for para 10 repeticiones
      {
        //Moviento de banda adelante y velocidad lenta
output_low(pin_c0);                     //Apagamos RA0
        output_high(pin_c3);                    //Encendemos RA1
set_pwm1_duty(125);                     //Valor del PWM=100
output_low(pin_a0);                     //Apagamos RA2
        output_low(pin_a1);                     //Apagamos RA3
        output_high(pin_a2);                    //Encendemos RA4
        output_low(pin_a3);                     //Apagamos RA5
delay_ms(700);                          //Espera tiempo
      }                                 //Despues de los ciclos...
```

```
output_low(pin_a0);                    //Apagamos RA0
      output_low(pin_a1);                    //Apagamos RA1
      output_low(pin_a2);                    //Apagamos RA2
      output_low(pin_a3);                    //Apagamos RA3
      output_low(pin_c0);                    //Apagamos RC0
      output_low(pin_c3);                    //Apagamos RC3
set_pwm1_duty(0);                      //Valor del PWM=0
      delay_ms(750);                       //Espera tiempo
      ch='p';                            //Asignar valor "p" a variable ch
      putc(ch);                          //Enviamos caracter ch=p
      delay_ms(300);                       //Espera tiempo
    }

    if((ch=='e'))                        //Si ch=e
    {                                  //Entonces...
      for(i=0;i<7;i++)                     //Iniciamos ciclo for repeticiones
      {
        //Moviento de banda adelante y velocidad rapida
output_low(pin_c0);                    //Apagamos RA0
        output_high(pin_c3);                 //Encendemos RA1
set_pwm1_duty(245);                    //Valor del PWM=100
output_low(pin_a0);                    //Apagamos RA2
        output_low(pin_a1);                  //Apagamos RA3
        output_low(pin_a2);                  //Apagamos RA4
        output_high(pin_a3);                 //Encendemos RA5
delay_ms(850);                       //Espera tiempo
      }                                  //Despues de los 10 ciclos...

output_low(pin_a0);                    //Apagamos RA0
      output_low(pin_a1);                    //Apagamos RA1
      output_low(pin_a2);                    //Apagamos RA2
      output_low(pin_a3);                    //Apagamos RA3
      output_low(pin_c0);                    //Apagamos RC0
      output_low(pin_c3);                    //Apagamos RC3
set_pwm1_duty(0);                      //Valor del PWM=0
      delay_ms(750);                       //Espera tiempo
      ch='p';                            //Asignar valor "p" a variable ch
      putc(ch);                          //Enviamos caracter ch=p
      delay_ms(300);                       //Espera tiempo
    }
  }
}
```

Programa estación 2

```
#include <16f876a.h>                        //Seleccionamos Pic a usar
#fuses hs,nowdt                            //Definimos crital, y fusibles
#use delay(clock=20M)                        //Definimos velocidad de reloj a 20MHz
#use rs232 (baud=9600, xmit=pin_c6, rcv=pin_c7,bits=8,parity=N)        //Definimos parametros de transmisión serial

char ch;                               //Variable donde guardamos dato serial obtenido
int timer2, poscaler;                        //variables para el timer
int i;                                 //Variable para iteraciones

#int_rda                                //HAbilitamos Interrupción por serial
void serial_isr()                          //Creamos función para interrupción
{
  ch=getc();                           //Guardamos dato recibido en la variable "ch"
  delay_us(600);                         //Espera 600 microsegundos
}
```

```
void main()
{
  //Configurar PWM
  timer2=255;                                    //Se carga el timer 2 con 255
  poscaler=1;                                    //Preescaler solo puede tomar valores de: 1
  setup_timer_2(t2_div_by_16,timer2,poscaler);          //Configuracion de timer 2 para establecer frecuencia PWM
  setup_ccp1(ccp_pwm);                             //Configurar modulo ccp1 en modo PWM
  setup_ccp2(ccp_pwm);                             //Configurar modulo ccp2 en modo PWM

  set_tris_a(0b000000);                            //Puerto a como salidas
  output_a(0b000000);                              //Inicializar salidas en cero
  output_low(pin_c4);                              //Encendemos CA4 control in1 de motor 1
  output_low(pin_c5);                              //Apagamos RA5 control in2 de motor 1
output_low(pin_c0);                                //Apagamos RC0 control in2 de motor 2
  output_low(pin_c3);                              //Apagamos RC3 control in2 de motor 2
enable_interrupts(global);                         //Habilitar variables globales
  enable_interrupts(int_rda);                      //Habilitar la interrupcion
  printf("Agente 2: Area de Barrenado" );                 //Envío mensaje en monitor serial
  delay_ms(500);                                   //Esperamos tiempo
  set_pwm1_duty(0);                                //Valor del PWM=0
  set_pwm2_duty(0);                                //Valor del PWM=0


  while(1)                                         //Iniciamos ciclo while
  {
    if((ch=='f'))                                  //Si ch=f
    {                                              //Entonces...
      output_low(pin_a0);                                 //Apagamos RA0
      output_low(pin_a1);                                 //Apagamos RA1
output_low(pin_a2);                                //Apagamos RA0
      output_low(pin_a3);                                 //Apagamos RA1
delay_ms(25);
      set_pwm1_duty(0);                                   //Valor del PWM=100
      set_pwm2_duty(0);                                   //Valor del PWM=100
}

    if((ch=='g'))                                  //Si ch=g
{                                                  //Entonces...
     for(i=0;i<10;i++)                                    //Iniciamos ciclo for para repeticiones
     {
       //Moviento de Taladro para material suave y broca fina
       output_high(pin_c4);                                //Encendemos CA4 control in1 de motor 1
       output_low(pin_c5);                                 //Apagamos RA5 control in2 de motor 1
output_low(pin_c0);                                //Apagamos RC0 control in2 de motor 2
       output_low(pin_c3);                                 //Apagamos RC3 control in2 de motor 2
       output_high(pin_a0);                                //Encendemos RA0
       output_low(pin_a1);                                 //Apagamos RA1
       output_low(pin_a2);                                 //Apagamos RA2
       output_low(pin_a3);                                 //Apagamos RA3
delay_ms(25);
       set_pwm1_duty(0);                                   //Valor del PWM="duty"
       set_pwm2_duty(120);                                 //Valor del PWM=0
       delay_ms(1000);                                 //Espera tiempo
     }                                             //Despues de los ciclos...

output_low(pin_a0);                                //Apagamos RA0
       output_low(pin_a1);                                 //Apagamos RA1
       output_low(pin_a2);                                 //Apagamos RA0
       output_low(pin_a3);                                 //Apagamos RA1
delay_ms(25);                                      //Espera tiempo
```

```
        set_pwm1_duty(0);                       //Valor del PWM=100
        set_pwm2_duty(0);                       //Valor del PWM=100
        delay_ms(750);                      //Espera 300 milisegundos
        ch='p';                             //Asignar valor "p" a variable ch
        putc(ch);                           //Enviamos caracter ch=p
        delay_ms(300);                          //Espera tiempo
}

    if((ch=='h'))                           //Si ch=h
{                                       //Entonces...
      for(i=0;i<10;i++)                         //Iniciamos ciclo for para repeticiones
      {
        //Moviento de Taladro para material suave y broca gruesa
        output_high(pin_c4);                    //Encendemos RA4 control in1 de motor 1
        output_low(pin_c5);                     //Apagamos RA5 control in2 de motor 1
output_low(pin_c0);                     //Apagamos RC0 control in2 de motor 2
        output_low(pin_c3);                     //Apagamos RC3 control in2 de motor 2
delay_ms(25);                           //Espera tiempo
        set_pwm1_duty(0);                       //Valor del PWM="duty"
        set_pwm2_duty(255);                       //Valor del PWM=0
output_low(pin_a0);                     //Apagamos RA0
        output_high(pin_a1);                    //Encendemos RA1
        output_low(pin_a2);                     //Apagamos RA2
        output_low(pin_a3);                     //Apagamos RA3
delay_ms(1000);                         //Espera tiempo
      }                                 //Despues de los ciclos...

output_low(pin_a0);                     //Apagamos RA0
        output_low(pin_a1);                     //Apagamos RA1
        output_low(pin_a2);                     //Apagamos RA0
        output_low(pin_a3);                     //Apagamos RA1
delay_ms(25);                           //Espera tiempo
        set_pwm1_duty(0);                       //Valor del PWM=100
        set_pwm2_duty(0);                       //Valor del PWM=100
        delay_ms(750);                      //Espera tiempo
        ch='p';                             //Asignar valor "p" a variable ch
        putc(ch);                           //Enviamos caracter ch=p
        delay_ms(300);                          //Espera tiempo
}

    if((ch=='i'))                           //Si ch=i
{                                       //Entonces...
      for(i=0;i<10;i++)                         //Iniciamos ciclo for para repeticiones
      {
        //Moviento de Taladro para material duro y broca fina
        output_low(pin_c4);                     //Apagamos RA4 control in1 de motor 1
        output_low(pin_c5);                     //Apagamos RA5 control in2 de motor 1
output_high(pin_c0);                    //Encendemos RC0 control in2 de motor 2
        output_low(pin_c3);                     //Apagamos RC5 control in2 de motor 2
delay_ms(25);                           //Espera tiempo
        set_pwm1_duty(120);                       //Valor del PWM="duty"
        set_pwm2_duty(0);                       //Valor del PWM=0
output_low(pin_a0);                     //Apagamos RA0
        output_low(pin_a1);                     //Apagamos RA1
        output_high(pin_a2);                    //Encendemos RA2
        output_low(pin_a3);                     //Apagamos RA3
delay_ms(1000);                         //Espera tiempo
      }                                 //Despues de los ciclos...

output_low(pin_a0);                     //Apagamos RA0
        output_low(pin_a1);                     //Apagamos RA1
```

```
        output_low(pin_a2);                         //Apagamos RA0
        output_low(pin_a3);                         //Apagamos RA1
delay_ms(25);                               //Espera tiempo
        set_pwm1_duty(0);                           //Valor del PWM=100
        set_pwm2_duty(0);                           //Valor del PWM=100
        delay_ms(750);                            //Espera tiempo
        ch='p';                                   //Asignar valor "p" a variable ch
        putc(ch);                                  //Enviamos caracter ch=p
        delay_ms(300);                              //Espera tiempo
}

    if((ch=='j'))                                  //Si ch=j
{                                             //Entonces...
      for(i=0;i<10;i++)                             //Iniciamos ciclo for para repeticiones
      {
        //Moviento de Taladro para material duro y broca gruesa
        output_low(pin_c4);                         //Apagamos RA4 control in1 de motor 1
        output_low(pin_c5);                         //Apagamos RA5 control in2 de motor 1
output_high(pin_c0);                          //Encendemos RC0 control in2 de motor 2
        output_low(pin_c3);                         //Apagamos RC3 control in2 de motor 2
delay_ms(25);                               //Espera tiempo
        set_pwm1_duty(255);                           //Valor del PWM="duty"
        set_pwm2_duty(0);                           //Valor del PWM=0
output_low(pin_a0);                           //Apagamos RA0
        output_low(pin_a1);                         //Apagamos RA1
        output_low(pin_a2);                         //Encendemos RCA2
        output_high(pin_a3);                        //Apagamos RA3
delay_ms(1000);                             //Espera tiempo
      }                                         //Despues de los ciclos...

output_low(pin_a0);                           //Apagamos RA0
        output_low(pin_a1);                         //Apagamos RA1
        output_low(pin_a2);                         //Apagamos RA0
        output_low(pin_a3);                         //Apagamos RA1
delay_ms(25);                               //Espera tiempo
        set_pwm1_duty(0);                           //Valor del PWM=100
        set_pwm2_duty(0);                           //Valor del PWM=100
        delay_ms(750);                            //Espera tiempo
        ch='p';                                   //Asignar valor "p" a variable ch
        putc(ch);                                  //Enviamos caracter ch=p
        delay_ms(300);                              //Espera tiempo
    }

  }
}
```

Programa estación 3

```
#include <16f876a.h>                               //Seleccionamos Pic a usar
#fuses hs,nowdt                                   //Definimos crital, y fusibles
#use delay(clock=20M)                               //Definimos velocidad de reloj a 20MHz
#use rs232 (baud=9600, xmit=pin_c6, rcv=pin_c7,bits=8,parity=N)       //Definimos parameetros de transmisión serial

char ch;                                      //Variable donde guardamos dato serial obtenido
int i;                                       //Variable para iteraciones

#int_rda                                      //HAbilitamos Interrupción por serial
void serial_isr()                               //Creamos función para interrupción
{
  ch=getc();                                  //Guardamos dato recibido en la variable "ch"
  delay_us(600);                                //Espera 600 microsegundos
```

```
}

void main()
{
  set_tris_a(0b000000);                    //Puerto a como salidas
output_high(pin_c0);                      //Apagamos Rele 1
  output_high(pin_c1);                     //Apagamos Rele 2
  output_high(pin_c2);                     //Apagamos Rele 3
output_a(0b000000);                       //Inicializar salidas en cero
  enable_interrupts(global);                 //Habilitar variables globales
  enable_interrupts(int_rda);                //Habilitar la interrupcion
  printf("Agente 3: Area de limpieza" );          //Envío mensaje de "estoy vivo" en monitor serial
  delay_ms(1500);                         //Esperamos 1.5 seg


  while(1)                                //Iniciamos ciclo while
{
    if((ch=='0'))                          //Si ch=0
    {                                     //Entonces...
      output_low(pin_a0);                      //Apagamos RA0
      output_low(pin_a1);                      //Apagamos RA1
      output_low(pin_a2);                      //Apagamos RA0
      output_low(pin_a3);                      //Apagamos RA1
delay_ms(25);                             //Espera tiempo

      output_high(pin_c0);                     //Apagamos Rele 1
output_high(pin_c1);                      //Apagamos Rele 2
      output_high(pin_c2);                     //Apagamos Rele 3
      delay_ms(25);
    }

    if((ch=='1'))                          //Si ch=1
{                                         //Entonces...
      for(i=0;i<1;i++)                       //iniciamos ciclo for para repeticiones
      {
        //Encender led de removedor
        output_low(pin_c0);                    //Encendemos RC0 relevador 1
        output_high(pin_c1);                   //Apagamos RC1 relevador 2
        output_high(pin_c2);                   //Apagamos RC2 relevador 3

        for(i=0;i<6;i++)                       //Iniciamos ciclo for para repeticiones
        {
          //Leds indicadores de Estación en función
output_high(pin_a0);                      //Encendemos RA0
          output_low(pin_a1);                  //Apagamos RA1
          output_low(pin_a2);                  //Apagamos RA2
delay_ms(500);                            //Espera tiempo
          output_low(pin_a0);                  //Apagamos RA0
output_low(pin_a1);                       //Apagamos RA1
          output_low(pin_a2);                  //Apagamos RA2
delay_ms(500);                            //Espera tiempo
        }                                     //Despues de los ciclos...

output_low(pin_a0);                       //Apagamos RA0
        output_low(pin_a1);                    //Apagamos RA1
        output_low(pin_a2);                    //Apagamos RA0
        output_low(pin_a3);                    //Apagamos RA1
delay_ms(25);                             //Espera tiempo

        output_high(pin_c0);                   //Apagamos Rele 1
output_high(pin_c1);                      //Apagamos Rele 2
```

```
        output_high(pin_c2);                    //Apagamos Rele 3
delay_ms(25);                           //Espera tiempo
        delay_ms(750);                          //Espera tiempo
        ch='p';                         //Asignar valor "p" a variable ch
        putc(ch);                       //Enviamos caracter ch=p
        delay_ms(300);                          //Espera tiempo
}
    }

    if((ch=='3'))                       //Si ch=3
{                                       //Entonces...
      for(i=0;i<1;i++)                  //Iniciamos ciclo for para repeticiones
      {
        //Encender led de enjuagado
        output_high(pin_c0);                    //Apagamos RC0 relevador 1
        output_low(pin_c1);                     //Encendemos RC1 relevador 2
        output_high(pin_c2);                    //Apagamos RC2 relevador 3

        for(i=0;i<6;i++)                //Iniciamos ciclo for para repeticiones
        {
        //Leds indicadores de Estación en función
output_low(pin_a0);                     //Apagamos RA0
        output_high(pin_a1);                    //Encendemos RA1
        output_low(pin_a2);                     //Apagamos RA2
delay_ms(500);                          //Espera tiempo
        output_low(pin_a0);                     //Apagamos RA0
output_low(pin_a1);                     //Apagamos RA1
        output_low(pin_a2);                     //Apagamos RA2
delay_ms(500);                          //Espera tiempo
        }

        output_low(pin_a0);                     //Apagamos RA0
output_low(pin_a1);                     //Apagamos RA1
        output_low(pin_a2);                     //Apagamos RA0
        output_low(pin_a3);                     //Apagamos RA1
delay_ms(25);                           //Espera tiempo

        output_high(pin_c0);                    //Apagamos Rele 1
output_high(pin_c1);                    //Apagamos Rele 2
        output_high(pin_c2);                    //Apagamos Rele 3
delay_ms(25);                           //Espera tiempo
        delay_ms(750);                          //Espera tiempo
        ch='p';                         //Asignar valor "p" a variable ch
        putc(ch);                       //Enviamos caracter ch=p
        delay_ms(300);                          //Espera tiempo
}
    }

    if((ch=='5'))                       //Si ch=5
{                                       //Entonces...
      for(i=0;i<1;i++)                  //Iniciamos ciclo for para repeticiones
      {
        //Encender led de secado
        output_high(pin_c0);                    //Apagamos RC0 relevador 1
        output_high(pin_c1);                    //Apagamos RC1 relevador 2
        output_low(pin_c2);                     //Encendemos RC2 relevador 3

        for(i=0;i<6;i++)                //Iniciamos ciclo for para repeticiones
        {
        //Leds indicadores de Estación en función
output_low(pin_a0);                     //Apagamos RA0
```

```
        output_low(pin_a1);                     //Apagamos RA1
        output_high(pin_a2);                    //Encendemos RA2
delay_ms(500);                          //Espera tiempo
        output_low(pin_a0);                     //Apagamos RA0
output_low(pin_a1);                     //Apagamos RA1
        output_low(pin_a2);                     //Apagamos RA2
delay_ms(500);                          //Espera tiempo
        }

        output_low(pin_a0);                     //Apagamos RA0
output_low(pin_a1);                     //Apagamos RA1
        output_low(pin_a2);                     //Apagamos RA0
        output_low(pin_a3);                     //Apagamos RA1
delay_ms(25);                           //Espera tiempo

        output_high(pin_c0);                    //Apagamos Rele 1
output_high(pin_c1);                    //Apagamos Rele 2
        output_high(pin_c2);                    //Apagamos Rele 3
delay_ms(25);                           //Espera tiempo
        delay_ms(750);                          //Espera tiempo
        ch='p';                                 //Asignar valor "p" a variable ch
        putc(ch);                               //Enviamos caracter ch=p
        delay_ms(300);                          //Espera tiempo
}
    }

    if((ch=='4'))                               //Si ch=4
{                                       //Entonces...
      for(i=0;i<1;i++)                          //Iniciamos ciclo for para repeticiones
      {
        for(i=0;i<1;i++)                        //Iniciamos ciclo for para repeticiones
        {
          //Encender led de removedor
          output_low(pin_c0);                   //Encendemos RC0 relevador 1
          output_high(pin_c1);                  //Apagamos RC1 relevador 2
          output_high(pin_c2);                  //Apagamos RC2 relevador 3

          for(i=0;i<6;i++)                      //Iniciamos ciclo for para repeticiones
          {
            //Leds indicadores de Estación en función
output_high(pin_a0);                    //Encendemos RA0
            output_low(pin_a1);                 //Apagamos RA1
            output_low(pin_a2);                 //Apagamos RA2
delay_ms(500);                          //Espera tiempo
            output_low(pin_a0);                 //Apagamos RA0
output_low(pin_a1);                     //Apagamos RA1
            output_low(pin_a2);                 //Apagamos RA2
delay_ms(500);                          //Espera tiempo
          }
        }

        for(i=0;i<1;i++)                        //Iniciamos ciclo for para repeticiones
        {
          //Encender led de enguagado
          output_high(pin_c0);                  //Apagamos RC0 relevador 1
          output_low(pin_c1);                   //Encendemos RC1 relevador 2
          output_high(pin_c2);                  //Apagamos RC2 relevador 3

          for(i=0;i<6;i++)                      //Iniciamos ciclo for para repeticiones
          {
            //Leds indicadores de Estación en función
```

```
output_low(pin_a0);                    //Apagamos RA0
            output_high(pin_a1);                    //Encendemos RA1
            output_low(pin_a2);                     //Apagamos RA2
delay_ms(500);                         //Espera tiempo
            output_low(pin_a0);                     //Apagamos RA0
output_low(pin_a1);                    //Apagamos RA1
            output_low(pin_a2);                     //Apagamos RA2
delay_ms(500);                         //Espera tiempo
          }
        }

        output_low(pin_a0);                     //Apagamos RA0
output_low(pin_a1);                       //Apagamos RA1
        output_low(pin_a2);                     //Apagamos RA0
        output_low(pin_a3);                     //Apagamos RA1
delay_ms(25);                          //Espera tiempo

        output_high(pin_c0);                    //Apagamos Rele 1
output_high(pin_c1);                      //Apagamos Rele 2
        output_high(pin_c2);                    //Apagamos Rele 3
delay_ms(25);                          //Espera tiempo
        delay_ms(750);                          //Espera 300 milisegundos
        ch='p';                                //Asignar valor "p" a variable ch
        putc(ch);                               //Enviamos caracter ch=p
        delay_ms(300);                          //Espera tiempo
}
    }


    if((ch=='6'))                               //Si ch=6
{                                          //Entonces...
      for(i=0;i<1;i++)                          //Iniciamos ciclo for para repeticiones
      {
        for(i=0;i<1;i++)                        //iniciamos ciclo for para repeticiones
        {
          //Encender led de removedor
          output_low(pin_c0);                   //Encendemos RC0 relevador 1
          output_high(pin_c1);                  //Apagamos RC1 relevador 2
          output_high(pin_c2);                  //Apagamos RC2 relevador 3

          for(i=0;i<6;i++)                      //Iniciamos ciclo for para repeticiones
          {
            //Leds indicadores de Estación en función
output_high(pin_a0);                   //Encendemos RA0
            output_low(pin_a1);                     //Apagamos RA1
            output_low(pin_a2);                     //Apagamos RA2
delay_ms(500);                         //Espera tiempo
            output_low(pin_a0);                     //Apagamos RA0
output_low(pin_a1);                    //Apagamos RA1
            output_low(pin_a2);                     //Apagamos RA2
delay_ms(500);                         //Espera tiempo
          }
        }

        for(i=0;i<1;i++)                        //Iniciamos ciclo for para repeticiones
        {
          //Encender led de secado
          output_high(pin_c0);                  //Apagamos RC0 relevador 1
          output_high(pin_c1);                  //Apagamos RC1 relevador 2
          output_low(pin_c2);                   //Encendemos RC2 relevador 3
```

```
          for(i=0;i<6;i++)                    //Iniciamos ciclo for para repeticiones
          {
            //Leds indicadores de Estación en función
output_low(pin_a0);                  //Apagamos RA0
            output_low(pin_a1);                    //Apagamos RA1
            output_high(pin_a2);                   //Encendemos RA2
delay_ms(500);                  //Espera tiempo
            output_low(pin_a0);                    //Apagamos RA0
output_low(pin_a1);                  //Apagamos RA1
            output_low(pin_a2);                    //Apagamos RA2
delay_ms(500);                  //Espera tiempo
          }
        }

        output_low(pin_a0);                    //Apagamos RA0
output_low(pin_a1);                  //Apagamos RA1
        output_low(pin_a2);                    //Apagamos RA0
        output_low(pin_a3);                    //Apagamos RA1
delay_ms(25);                  //Espera tiempo

        output_high(pin_c0);                   //Apagamos Rele 1
output_high(pin_c1);                 //Apagamos Rele 2
        output_high(pin_c2);                   //Apagamos Rele 3
delay_ms(25);                  //Espera tiempo
        delay_ms(750);                    //Espera tiempo
        ch='p';                          //Asignar valor "p" a variable ch
        putc(ch);                         //Enviamos caracter ch=p
        delay_ms(300);                    //Espera tiempo
}
    }


    if((ch=='8'))                          //Si ch=8
{                                     //Entonces...
      for(i=0;i<1;i++)                    //Iniciamos ciclo for para repeticiones
      {
        //Encender led de enguagado
        output_high(pin_c0);                   //Apagamos RC0 relevador 1
        output_low(pin_c1);                    //Encendemos RC1 relevador 2
        output_high(pin_c2);                   //Apagamos RC2 relevador 3

        for(i=0;i<6;i++)                    //Iniciamos ciclo for para repeticiones
        {
        //Leds indicadores de Estación en función
output_low(pin_a0);                  //Apagamos RA0
        output_high(pin_a1);                   //Encendemos RA1
        output_low(pin_a2);                    //Apagamos RA2
delay_ms(500);                  //Espera tiempo
        output_low(pin_a0);                    //Apagamos RA0
output_low(pin_a1);                  //Apagamos RA1
        output_low(pin_a2);                    //Apagamos RA2
delay_ms(500);                  //Espera tiempo
        }

      for(i=0;i<1;i++)                    //iniciamos ciclo for para repeticiones
        {
          //Encender led de secado
          output_high(pin_c0);                 //Apagamos RC0 relevador 1
          output_high(pin_c1);                 //Apagamos RC1 relevador 2
          output_low(pin_c2);                  //Encendemos RC2 relevador 3
```

```
          for(i=0;i<6;i++)                    //Iniciamos ciclo for para repeticiones
          {
            //Leds indicadores de Estación en función
output_low(pin_a0);                  //Apagamos RA0
            output_low(pin_a1);                    //Apagamos RA1
            output_high(pin_a2);                   //Encendemos RA2
delay_ms(500);                  //Espera tiempo
            output_low(pin_a0);                    //Apagamos RA0
output_low(pin_a1);                  //Apagamos RA1
            output_low(pin_a2);                    //Apagamos RA2
delay_ms(500);                  //Espera tiempo
          }
        }

        output_low(pin_a0);                    //Apagamos RA0
output_low(pin_a1);                   //Apagamos RA1
        output_low(pin_a2);                    //Apagamos RA0
        output_low(pin_a3);                    //Apagamos RA1
delay_ms(25);                   //Esperamos 25 milisegundos

        output_high(pin_c0);                   //Apagamos Rele 1
output_high(pin_c1);                  //Apagamos Rele 2
        output_high(pin_c2);                   //Apagamos Rele 3
delay_ms(25);                   //Espera tiempo
        delay_ms(750);                    //Espera tiempo
        ch='p';                          //Asignar valor "p" a variable ch
        putc(ch);                         //Enviamos caracter ch=p
        delay_ms(300);                    //Espera tiempo
}
    }


    if((ch=='9'))                          //Si ch=9
{                                   //Entonces...
      for(i=0;i<1;i++)                    //Iniciamos ciclo for para repeticiones
      {
        for(i=0;i<1;i++)                  //Iniciamos ciclo for para repeticiones
        {
          //Encender led de removedor
          output_low(pin_c0);                  //Encendemos RC0 relevador 1
          output_high(pin_c1);                 //Apagamos RC1 relevador 2
          output_high(pin_c2);                 //Apagamos RC2 relevador 3

          for(i=0;i<6;i++)                    //Iniciamos ciclo for para repeticiones
          {
            //Leds indicadores de Estación en función
output_high(pin_a0);                 //Encendemos RA0
            output_low(pin_a1);                    //Apagamos RA1
            output_low(pin_a2);                    //Apagamos RA2
delay_ms(500);                  //Espera tiempo
            output_low(pin_a0);                    //Apagamos RA0
output_low(pin_a1);                  //Apagamos RA1
            output_low(pin_a2);                    //Apagamos RA2
delay_ms(500);                  //Espera tiempo
          }
        }

        for(i=0;i<1;i++)                  //Iniciamos ciclo for para repeticiones
        {
          //Encender led de enguagado
          output_high(pin_c0);                 //Apagamos RC0 relevador 1
```

```
          output_low(pin_c1);                    //Encendemos RC1 relevador 2
          output_high(pin_c2);                   //Apagamos RC2 relevador 3

          for(i=0;i<6;i++)                       //Iniciamos ciclo for para repeticiones
          {
            //Leds indicadores de Estación en función
output_low(pin_a0);                      //Apagamos RA0
            output_high(pin_a1);                 //Encendemos RA1
            output_low(pin_a2);                  //Apagamos RA2
delay_ms(500);                           //Espera tiempo
            output_low(pin_a0);                  //Apagamos RA0
output_low(pin_a1);                      //Apagamos RA1
            output_low(pin_a2);                  //Apagamos RA2
delay_ms(500);                           //Espera tiempo
          }
        }

        for(i=0;i<1;i++)                         //Iniciamos ciclo for para repeticiones
        {
          //Encender led de secado
          output_high(pin_c0);                   //Apagamos RC0 relevador 1
          output_high(pin_c1);                   //Apagamos RC1 relevador 2
          output_low(pin_c2);                    //Encendemos RC2 relevador 3

          for(i=0;i<6;i++)                       //Iniciamos ciclo for para repeticiones
          {
            //Leds indicadores de Estación en función
output_low(pin_a0);                      //Apagamos RA0
            output_low(pin_a1);                  //Apagamos RA1
            output_high(pin_a2);                 //Encendemos RA2
delay_ms(500);                           //Espera tiempo
            output_low(pin_a0);                  //Apagamos RA0
output_low(pin_a1);                      //Apagamos RA1
            output_low(pin_a2);                  //Apagamos RA2
delay_ms(500);                           //Espera tiempo
}
        }

        output_low(pin_a0);                      //Apagamos RA0
output_low(pin_a1);                        //Apagamos RA1
        output_low(pin_a2);                      //Apagamos RA0
        output_low(pin_a3);                      //Apagamos RA1
delay_ms(25);                            //Espera tiempo

        output_high(pin_c0);                     //Apagamos Rele 1
output_high(pin_c1);                       //Apagamos Rele 2
        output_high(pin_c2);                     //Apagamos Rele 3
delay_ms(25);                            //Espera tiempo
        delay_ms(750);                           //Espera tiempo
        ch='p';                                //Asignar valor "p" a variable ch
        putc(ch);                                //Enviamos caracter ch=p
        delay_ms(300);                           //Espera tiempo
      }
    }

  }
}
```

Programa estación 4

```
#include <16f876a.h>                                    //Seleccionamos Pic a usar
#fuses hs,nowdt                                         //Definimos crital, y fusibles
#use delay(clock=20M)                                   //Definimos velocidad de reloj a 20MHz
#use rs232 (baud=9600, xmit=pin_c6, rcv=pin_c7,bits=8,parity=N)       //Definimos parameetros de transmisión serial

#define lcd_data_port getenv("sfr:portb")               //DEfinimos el puerto de datos para lcd
#define lcd_rs_pin      pin_b0                          //Definimos RB0 como pin rs de la lcd
#define lcd_rw_pin      pin_b1                          //Definimos RB1 como pin rw de la lcd
#define lcd_enable_pin  pin_b2                          //Definimos RB2 como pin enable de la lcd
#include <lcd.c>                                        //Incluimos libreria lcd

char ch;                                                //Variable donde guardamos dato serial obtenido
int timer2, poscaler;                                   //variables para el timer
int i;

#int_rda                                                //HAbilitamos Interrupción por serial
void serial_isr()                                       //Creamos función para interrupción
{
  ch=getc();                                            //Guardamos dato recibido en la variable "ch"
  delay_us(600);                                        //Espera 600 microsegundos
}

void main()
{
  set_tris_a(0b000000);                                 //Puerto a como salidas
  output_a(0b000000);                                   //Inicializar salidas en cero
  lcd_init();                                           //Inicializar lcd
  enable_interrupts(global);                            //Habilitar variables globales
  enable_interrupts(int_rda);                           //Habilitar la interrupcion
  printf("Agente 4: Area de CNC");                      //Envío mensaje en monitor serial
  lcd_gotoxy(1,1);                                      //Colocamos el cursor en 1,1 de la lcd
  //          "1234567890123456
  printf(lcd_putc,"   Agente 4:    ",);                 //Mostramos mensaje de bienvenida en lcd
  delay_ms(100);                                        //Espera tiempo
  lcd_gotoxy(1,1);                                      //Colocamos el cursor en 1,1 de la lcd
  //          "1234567890123456
  printf(lcd_putc," Area de C N C ");                   //Mostramos mensaje en lcd
  lcd_gotoxy(1,2);                                      //Colocamos el cursor en 1,2 de la lcd
  //          "1234567890123456
  printf(lcd_putc,"Agente 4  Libre ");                  //Mostramos mensaje en lcd
  delay_ms(50);                                         //Espera tiempo

  while(1)                                              //Iniciamos ciclo while
{
    if((ch=='k'))                                       //Si ch=k
    {                                                   //Entonces...
      output_low(pin_a0);                               //Apagamos RA0
      output_low(pin_a1);                               //Apagamos RA1
      output_low(pin_a2);                               //Apagamos RA0
      output_low(pin_a3);                               //Apagamos RA1
delay_ms(25);                                           //Espera tiempo
      lcd_gotoxy(1,1);                                  //Colocamos el cursor en 1,1 de la lcd
      //          "1234567890123456
      printf(lcd_putc," Area de C N C ");               //Mostramos mensaje en lcd
      lcd_gotoxy(1,2);                                  //Colocamos el cursor en 1,2 de la lcd
      //          "1234567890123456
      printf(lcd_putc,"Agente 4  Libre ");              //Mostramos mensaje en lcd
      delay_ms(50);                                     //Espera tiempo
    }
```

```
    if((ch=='l'))                                   //Si ch=l
    {                                             //Entonces...
      //Leds indicadores de Agente
      output_high(pin_a0);                              //Encendemos RA0
output_low(pin_a1);                                 //Apagamos RA1
      output_low(pin_a2);                               //Apagamos RA2
      output_low(pin_a3);                               //Apagamos RA3
delay_ms(25);                                     //Espera tiempo
      lcd_gotoxy(1,1);                                  //Colocamos el cursor en 1,1 de la lcd
      //          "1234567890123456
      printf(lcd_putc," Realizando.... ");                    //Mostramos mensaje en lcd
      lcd_gotoxy(1,2);                                  //Colocamos el cursor en 1,2 de la lcd
      //          "1234567890123456
      printf(lcd_putc,"Fig.: Cuadrado  ");                    //Mostramos mensaje en lcd
      delay_ms(50);                                     //Espera tiempo
      for(i=0;i<20;i++)                                 //Iniciamos ciclo for para repeticiones
{
        output_high(pin_c0);                            //Encendemos RA5
        output_low(pin_c1);                             //Apagamos RC0
        output_low(pin_c2);                             //Apagamos RC3
        output_low(pin_c3);                             //Apagamos RC4
delay_ms(250);                                    //Espera tiempo
        output_low(pin_c0);                             //Apagamos CA5
output_low(pin_c1);                                 //Apagamos RC0
        output_low(pin_c2);                             //Apagamos RC3
        output_low(pin_c3);                             //Apagamos RC4
delay_ms(250);                                    //Espera tiempo
      }
        output_low(pin_a0);                             //Apagamos RA0
output_low(pin_a1);                                 //Apagamos RA1
        output_low(pin_a2);                             //Apagamos RA0
        output_low(pin_a3);                             //Apagamos RA1
delay_ms(25);                                     //Espera tiempo
        lcd_gotoxy(1,1);                                //Colocamos el cursor en 1,1 de la lcd
        //          "1234567890123456
        printf(lcd_putc," Area de C N C ");                   //Mostramos mensaje en lcd
        lcd_gotoxy(1,2);                                //Colocamos el cursor en 1,2 de la lcd
        //          "1234567890123456
        printf(lcd_putc,"Agente 4  Libre ");                  //Mostramos mensaje en lcd
        delay_ms(750);                                  //Espera 300 milisegundos
        ch='p';                                       //Asignar valor "p" a variable ch
        putc(ch);                                       //Enviamos caracter ch=p
        delay_ms(300);                                  //Espera tiempo
}

    if((ch=='m'))                                   //Si ch=m
{                                               //Entonces...
      //Leds indicadores de Agente
      output_low(pin_a0);                               //Apagamos RA0
output_high(pin_a1);                                //Encendemos RA1
      output_low(pin_a2);                               //Apagamos RA2
      output_low(pin_a3);                               //Apagamos RA3
lcd_gotoxy(1,1);                                  //Colocamos el cursor en 1,1 de la lcd
      //          "1234567890123456
      printf(lcd_putc," Realizando.... ");                    //Mostramos mensaje en lcd
      lcd_gotoxy(1,2);                                  //Colocamos el cursor en 1,2 de la lcd
      //          "1234567890123456
      printf(lcd_putc,"Fig.: Triangulo ");                    //Mostramos mensaje en lcd
      delay_ms(50);                                     //Espera tiempo
      for(i=0;i<10;i++)                                 //Iniciamos ciclo for para repeticiones
{
```

```
        output_low(pin_c0);                     //Apagamos RA5
        output_high(pin_c1);                    //Encendemos RC0
        output_low(pin_c2);                     //Apagamos RC3
        output_low(pin_c3);                     //Apagamos RC4
delay_ms(500);                          //Espera tiempo
        output_low(pin_c0);                     //Apagamos RA5
output_low(pin_c1);                       //Apagamos RC0
        output_low(pin_c2);                     //Apagamos RC3
        output_low(pin_c3);                     //Apagamos RC4
delay_ms(500);                          //Espera tiempo
      }
        output_low(pin_a0);                     //Apagamos RA0
output_low(pin_a1);                       //Apagamos RA1
        output_low(pin_a2);                     //Apagamos RA0
        output_low(pin_a3);                     //Apagamos RA1
delay_ms(25);                           //Espera tiempo
        lcd_gotoxy(1,1);                        //Colocamos el cursor en 1,1 de la lcd
        //          "1234567890123456
        printf(lcd_putc," Area de C N C ");             //Mostramos mensaje en lcd
        lcd_gotoxy(1,2);                        //Colocamos el cursor en 1,2 de la lcd
        //          "1234567890123456
        printf(lcd_putc,"Agente 4  Libre ");            //Mostramos mensaje en lcd
        delay_ms(750);                          //Espera tiempo
        ch='p';                                 //Asignar valor "p" a variable ch
        putc(ch);                               //Enviamos caracter ch=p
        delay_ms(300);                          //Espera tiempo
}

    if((ch=='n'))                               //Si ch=n
{                                       //Entonces...
      //Leds indicadores de Agente
      output_low(pin_a0);                       //Apagamos RA0
output_low(pin_a1);                       //Apagamos RA1
      output_high(pin_a2);                      //Encendemos RA2
      output_low(pin_a3);                       //Apagamos RA3
lcd_gotoxy(1,1);                        //Colocamos el cursor en 1,1 de la lcd
      //          "1234567890123456
      printf(lcd_putc," Realizando.... ");              //Mostramos mensaje en lcd
      lcd_gotoxy(1,2);                          //Colocamos el cursor en 1,2 de la lcd
      //          "1234567890123456
      printf(lcd_putc,"Fig.: Rectangulo");              //Mostramos mensaje en lcd
      delay_ms(50);                             //Espera tiempo
      for(i=0;i<7;i++)                          //iniciamos ciclo for para repeticiones
{
        output_low(pin_c0);                     //Apagamos RA5
        output_low(pin_c1);                     //Apagamos RC0
        output_high(pin_c2);                    //Encendemos RC3
        output_low(pin_c3);                     //Apagamos RC4
delay_ms(750);                          //Espera tiempo
        output_low(pin_c0);                     //Apagamos RA5
output_low(pin_c1);                       //Apagamos RC0
        output_low(pin_c2);                     //Apagamos RC3
        output_low(pin_c3);                     //Apagamos RC4
delay_ms(750);                          //Espera tiempo
      }
        output_low(pin_a0);                     //Apagamos RA0
output_low(pin_a1);                       //Apagamos RA1
        output_low(pin_a2);                     //Apagamos RA0
        output_low(pin_a3);                     //Apagamos RA1
delay_ms(25);                           //Espera tiempo
        lcd_gotoxy(1,1);                        //Colocamos el cursor en 1,1 de la lcd
```

```
        //          "1234567890123456
        printf(lcd_putc," Area de C N C ");                 //Mostramos mensaje en lcd
        lcd_gotoxy(1,2);                                    //Colocamos el cursor en 1,2 de la lcd
        //          "1234567890123456
        printf(lcd_putc,"Agente 4  Libre ");                //Mostramos mensaje en lcd
        delay_ms(750);                                      //Espera tiempo
        ch='p';                                             //Asignar valor "p" a variable ch
        putc(ch);                                           //Enviamos caracter ch=p
        delay_ms(300);                                      //Espera tiempo
}

    if((ch=='o'))                                           //Si ch=w
{                                                           //Entonces...
      //Leds indicadores de Agente
       output_low(pin_a0);                                  //Apagamos RA0
output_low(pin_a1);                                         //Apagamos RA1
       output_low(pin_a2);                                  //Encendemos RCA2
       output_high(pin_a3);                                 //Apagamos RA3
lcd_gotoxy(1,1);                                            //Colocamos el cursor en 1,1 de la lcd
       //          "1234567890123456
       printf(lcd_putc," Realizando.... ");                 //Mostramos mensaje en lcd
       lcd_gotoxy(1,2);                                     //Colocamos el cursor en 1,2 de la lcd
       //          "1234567890123456
       printf(lcd_putc,"Fig.: Pentagono ");                 //Mostramos mensaje en lcd
       delay_ms(50);                                        //Espera tiempo
       for(i=0;i<5;i++)                                     //Iniciamos ciclo for para repeticiones
{
         output_low(pin_c0);                                //Apagamos CA4
         output_low(pin_c1);                                //Apagamos RA5
         output_low(pin_c2);                                //Apagamos RC0
         output_high(pin_c3);                               //Encendemos RC3
delay_ms(1000);                                             //Espera tiempo
         output_low(pin_c0);                                //Apagamos RA5
output_low(pin_c1);                                         //Apagamos RC0
         output_low(pin_c2);                                //Apagamos RC3
         output_low(pin_c3);                                //Apagamos RC4
delay_ms(1000);                                             //Espera tiempo
}
         output_low(pin_a0);                                //Apagamos RA0
output_low(pin_a1);                                         //Apagamos RA1
         output_low(pin_a2);                                //Apagamos RA0
         output_low(pin_a3);                                //Apagamos RA1
delay_ms(25);                                               //Espera tiempo
         lcd_gotoxy(1,1);                                   //Colocamos el cursor en 1,1 de la lcd
         //          "1234567890123456
         printf(lcd_putc," Area de C N C ");                //Mostramos mensaje en lcd
         lcd_gotoxy(1,2);                                   //Colocamos el cursor en 1,2 de la lcd
         //          "1234567890123456
         printf(lcd_putc,"Agente 4  Libre ");               //Mostramos mensaje en lcd
         delay_ms(750);                                     //Espera tiempo
         ch='p';                                            //Asignar valor "p" a variable ch
         putc(ch);                                          //Enviamos caracter ch=p
         delay_ms(300);                                     //Espera tiempo
    }

  }
}
```

Programa estación 5

```
#include <16F876a.h>                                        //Seleccionamos Pic a usar
#fuses HS,NOWDT,NOPROTECT,NOLVP,PUT,BROWNOUT                //Definimos crital, y fusibles
#use delay(clock=20000000)                                  //Definimos velocidad de reloj a 20MHz
#use rs232(baud=9600, xmit=pin_C6, rcv=pin_C7)              //Definimos parameetros de transmisión serial

char ch;                                                    //Variable donde guardamos dato serial obtenido

#int_rda                                                    //HAbilitamos Interrupción por serial
void serial_isr()                                           //Creamos función para interrupción
{
  ch=getc();                                                //Guardamos dato recibido en la variable "ch"
  //putc(ch);                                               //Mandamos el dato recibido al monitor srial
  delay_us(600);                                            //Espera 600 microsegundos
}

/* 10 funciones
  2 para posicion inicial abierta y cerrada
  4 para bajar y agarrar. Cuado Agente termino - aX_agarrar
  4 para bajar y dejar. Cuando agente pide empezar - aX_dejar*/

void inicial_abierta()                                      //Función mover brazo a inicio pinza abierta
{
/* POSICION INICIAL ABIERTA
  F XXXX
  E 1260
  B 1000
  C 1000
  D 1250
  A 1277*/

    //print("F");                                           //Envío valor "F" por Tx
    //delay_ms(100);                                        //Espera tiempo
    //printf("xxxx");                                       //Envio valor "xxxx" por Tx
    //delay_ms(1000);                                       //Espera tiempo
    printf("E" );                                           //Envío valor "E" por Tx
    delay_ms(100);                                          //Espera tiempo
    printf("1260" );                                        //Envio valor "1260" por Tx
    delay_ms(3000);                                         //Espera tiempo
    printf("C" );                                           //Envío valor "C" por Tx
    delay_ms(100);                                          //Espera tiempo
    printf("1000" );                                        //Envio valor "1000" por Tx
    delay_ms(1000);                                         //Espera tiempo
    printf("B" );                                           //Envío valor "B" por Tx
    delay_ms(100);                                          //Espera tiempo
    printf("1000" );                                        //Envio valor "1000" por Tx
    delay_ms(1000);                                         //Espera tiempo
    printf("D" );                                           //Envío valor "D" por Tx
    delay_ms(100);                                          //Espera tiempo
    printf("1250" );                                        //Envio valor "1250" por Tx
    delay_ms(1000);                                         //Espera tiempo
    printf("A" );                                           //Envío valor "A" por Tx
    delay_ms(100);                                          //Espera tiempo
    printf("1277" );                                        //Envio valor "1277" por Tx
    delay_ms(1000);                                         //Espera tiempo

    ch='p';                                                 //Asignar valor "p" a variable ch
    putc(ch);                                               //Enviamos caracter ch=p
    delay_ms(300);                                          //Espera tiempo
}
```

```
void inicial_cerrada()                          //Función mover brazo a inicio y pinza cerrada
{
/* POSICION INICIAL
  F XXXX
  E 1260
  B 1000
  C 1000
  D 1250
  A 1277*/

    //print("F");                               //Envío valor "F" por Tx
    //delay_ms(100);                            //Espera tiempo
    //printf("xxxx");                           //Envio valor "xxxx" por Tx
    //delay_ms(1000);                           //Espera tiempo
    printf("E" );                               //Envío valor "E" por Tx
    delay_ms(100);                              //Espera tiempo
    printf("1260" );                            //Envio valor "1260" por Tx
    delay_ms(3000);                             //Espera tiempo
    printf("C" );                               //Envío valor "C" por Tx
    delay_ms(100);                              //Espera tiempo
    printf("1000" );                            //Envio valor "1000" por Tx
    delay_ms(1000);                             //Espera tiempo
    printf("B" );                               //Envío valor "B" por Tx
    delay_ms(100);                              //Espera tiempo
    printf("1000" );                            //Envio valor "1000" por Tx
    delay_ms(1000);                             //Espera tiempo
    printf("D" );                               //Envío valor "D" por Tx
    delay_ms(100);                              //Espera tiempo
    printf("1250" );                            //Envio valor "1250" por Tx
    delay_ms(1000);                             //Espera tiempo
    printf("A" );                               //Envío valor "A" por Tx
    delay_ms(100);                              //Espera tiempo
    printf("1277" );                            //Envio valor "1277" por Tx
    delay_ms(1000);                             //Espera tiempo

    ch='p';                                     //Asignar valor "q" a variable ch
    putc(ch);                                   //Enviamos caracter ch=p
    delay_ms(300);                              //Espera tiempo
}

void a1_agarrar()                               //Función mover brazo a estación 1 para agarrar pieza
{
if((ch=='a'))                                   //Si ch=a
/* Posicion Agarrar de agente 1
  E 0500
  B 1700
  C 1630
  D 0800
  A 0500
  F XXXX*/

    delay_ms(500);                              //Espera tiempo
    printf("A" );                               //Envío valor "A" por Tx
    delay_ms(100);                              //Espera tiempo
    printf("0980" );                            //Envio valor "0980" por Tx
    delay_ms(1200);                             //Espera tiempo
    printf("B" );                               //Envío valor "B" por Tx
    delay_ms(100);                              //Espera tiempo
    printf("1640" );                            //Envio valor "1640" por Tx
    delay_ms(1000);                             //Espera tiempo
    printf("C" );                               //Envío valor "C" por Tx
```

```
    delay_ms(100);                                  //Espera tiempo
    printf("1620" );                               //Envio valor "1620" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("D" );                                  //Envío valor "D" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("0850" );                               //Envio valor "0850" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("E" );                                  //Envío valor "E" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1260" );                               //Envio valor "1260" por Tx
    delay_ms(1000);                                 //Espera tiempo
    //print("F");                                  //Envío valor "F" por Tx
    //delay_ms(100);                                //Espera tiempo
    //printf("xxxx");                              //Envio valor "xxxx" por Tx
    //delay_ms(1000)                                //Espera tiempo
}

void a1_soltar()                                    //Función mover brazo a estación 1 para soltar pieza
{
if((ch=='B') || (ch=='C') || (ch=='D') || (ch=='E'))                    //Si ch = B ó C ó D ó E...
/* Posicion soltar de agente 1
  A 0500
  B 1700
  C 1630
  D 0800
  E 1235
  F XXXX*/

    delay_ms(500);                                  //Espera tiempo
    printf("A" );                                  //Envío valor "A" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("0980" );                               //Envio valor "0980" por Tx
    delay_ms(1200);                                 //Espera tiempo
    printf("B" );                                  //Envío valor "B" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1640" );                               //Envio valor "1640" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("C" );                                  //Envío valor "C" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1620" );                               //Envio valor "1620" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("D" );                                  //Envío valor "D" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("0850" );                               //Envio valor "0850" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("E" );                                  //Envío valor "E" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1260" );                               //Envio valor "1260" por Tx
    delay_ms(1000);                                 //Espera tiempo
    //print("F");                                  //Envío valor "F" por Tx
    //delay_ms(100);                                //Espera tiempo
    //printf("xxxx");                              //Envio valor "xxxx" por Tx
    //delay_ms(1000)                                //Espera tiempo
}

void a2_agarrar()                                   //Función mover brazo a estación 2 para agarrar pieza
{
if((ch=='f'))                                      //Si ch=f
/* Posicion Agarrar de agente 2
  A 0960
  B 1520
```

```
  C 1520
  D 0850
  E 1260
  F XXXX*/

    delay_ms(500);                                  //Espera tiempo
    printf("A" );                                  //Envío valor "A" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1270" );                               //Envio valor "1270" por Tx
    delay_ms(1200);                                 //Espera tiempo
    printf("B" );                                  //Envío valor "B" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1750" );                               //Envio valor "1750" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("C" );                                  //Envío valor "C" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1720" );                               //Envio valor "1720" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("D" );                                  //Envío valor "D" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("0850" );                               //Envio valor "0850" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("E" );                                  //Envío valor "E" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1260" );                               //Envio valor "1260" por Tx
    delay_ms(1000);                                 //Espera tiempo
    //print("F");                                  //Envío valor "F" por Tx
    //delay_ms(100);                                //Espera tiempo
    //printf("xxxx");                              //Envio valor "xxxx" por Tx
    //delay_ms(1000);                               //Espera tiempo
}

void a2_soltar()                                    //Función mover brazo a estación 2 para soltar pieza
{
if((ch=='G') || (ch=='H') || (ch=='I') || (ch=='J'))                    //Si ch = G ó H ó I ó J...
/* Posicion soltar de agente 2
  A 0960
  B 1520
  C 1520
  D 0850
  E 1260
  F XXXX*/

    delay_ms(500);                                  //Espera tiempo
    printf("A" );                                  //Envío valor "A" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1270" );                               //Envio valor "1270" por Tx
    delay_ms(1200);                                 //Espera tiempo
    printf("B" );                                  //Envío valor "B" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1750" );                               //Envio valor "1750" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("C" );                                  //Envío valor "C" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("1720" );                               //Envio valor "1720" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("D" );                                  //Envío valor "D" por Tx
    delay_ms(100);                                  //Espera tiempo
    printf("0850" );                               //Envio valor "0850" por Tx
    delay_ms(1000);                                 //Espera tiempo
    printf("E" );                                  //Envío valor "E" por Tx
```

```
    delay_ms(100);                                          //Espera tiempo
    printf("1260" );                                       //Envio valor "1260" por Tx
    delay_ms(1000);                                         //Espera tiempo
    //print("F");                                          //Envío valor "F" por Tx
    //delay_ms(100);                                        //Espera tiempo
    //printf("xxxx");                                      //Envio valor "xxxx" por Tx
    //delay_ms(1000);*/                                     //Espera tiempo

}

void a3_agarrar()                                           //Función mover brazo a estación 3 para agarrar pieza
{
if((ch=='0'))                                              //Si ch=0
/* Posicion Agarrar de agente 3
  A 1620
  B 1700
  C 1750
  D 0850
  E 1260
  F XXXX*/

    delay_ms(500);                                          //Espera tiempo
    printf("A" );                                          //Envío valor "A" por Tx
    delay_ms(100);                                          //Espera tiempo
    printf("1620" );                                       //Envio valor "1620" por Tx
    delay_ms(1200);                                         //Espera tiempo
    printf("B" );                                          //Envío valor "B" por Tx
    delay_ms(100);                                          //Espera tiempo
    printf("1640" );                                       //Envio valor "1640" por Tx
    delay_ms(1000);                                         //Espera tiempo
    printf("C" );                                          //Envío valor "C" por Tx
    delay_ms(100);                                          //Espera tiempo
    printf("1650" );                                       //Envio valor "1650" por Tx
    delay_ms(1000);                                         //Espera tiempo
    printf("D" );                                          //Envío valor "D" por Tx
    delay_ms(100);                                          //Espera tiempo
    printf("0850" );                                       //Envio valor "0850" por Tx
    delay_ms(1000);                                         //Espera tiempo
    printf("E" );                                          //Envío valor "E" por Tx
    delay_ms(100);                                          //Espera tiempo
    printf("1260" );                                       //Envio valor "1260" por Tx
    delay_ms(1000);                                         //Espera tiempo
    //print("F");                                          //Envío valor "F" por Tx
    //delay_ms(100);                                        //Espera tiempo
    //printf("xxxx");                                      //Envio valor "xxxx" por Tx
    //delay_ms(1000);                                       //Espera tiempo
}

void a3_soltar()                                            //Función mover brazo a estación 3 para soltar pieza
{
if((ch=='R') || (ch=='S') || (ch=='T') || (ch=='U') || (ch=='V') || (ch=='W') || (ch=='X'))       //Si ch = R ó S ó T ó U ó V ó W ó X...
/* Posicion soltar de agente 3
  A 1620
  B 1700
  C 1750
  D 0850
  E 1260
  F XXXX*/

    delay_ms(500);                                          //Espera tiempo
    printf("A" );                                          //Envío valor "A" por Tx
```

```
    delay_ms(100);                          //Espera tiempo
    printf("1620" );                       //Envio valor "1620" por Tx
    delay_ms(1200);                         //Espera tiempo
    printf("B" );                         //Envío valor "B" por Tx
    delay_ms(100);                          //Espera tiempo
    printf("1640" );                       //Envio valor "1640" por Tx
    delay_ms(1000);                         //Espera tiempo
    printf("C" );                         //Envío valor "C" por Tx
    delay_ms(100);                          //Espera tiempo
    printf("1650" );                       //Envio valor "1650" por Tx
    delay_ms(1000);                         //Espera tiempo
    printf("D" );                         //Envío valor "D" por Tx
    delay_ms(100);                          //Espera tiempo
    printf("0850" );                       //Envio valor "0850" por Tx
    delay_ms(1000);                         //Espera tiempo
    printf("E" );                         //Envío valor "E" por Tx
    delay_ms(100);                          //Espera tiempo
    printf("1260" );                       //Envio valor "1260" por Tx
    delay_ms(1000);                         //Espera tiempo
    //print("F");                          //Envío valor "F" por Tx
    //delay_ms(100);                          //Espera tiempo
    //printf("xxxx");                       //Envio valor "xxxx" por Tx
    //delay_ms(1000);                         //Espera tiempo
}

void a4_agarrar()                           //Función mover brazo a estación 4 para agarrar pieza
{
if((ch=='k'))                               //Si ch=k
/* Posicion Agarrar de agente 4
  A 1760
  B 1650
  C 1770
  D 0851
  E 1260
  F XXXX*/

    delay_ms(500);                          //Espera tiempo
    printf("A" );                         //Envío valor "A" por Tx
    delay_ms(100);                          //Espera tiempo
    printf("1710" );                       //Envio valor "1710" por Tx
    delay_ms(1200);                         //Espera tiempo
    printf("B" );                         //Envío valor "B" por Tx
    delay_ms(100);                          //Espera tiempo
    printf("1750" );                       //Envio valor "1750" por Tx
    delay_ms(1000);                         //Espera tiempo
    printf("C" );                         //Envío valor "C" por Tx
    delay_ms(100);                          //Espera tiempo
    printf("1700" );                       //Envio valor "1700" por Tx
    delay_ms(1000);                         //Espera tiempo
    printf("D" );                         //Envío valor "D" por Tx
    delay_ms(100);                          //Espera tiempo
    printf("0850" );                       //Envio valor "0850" por Tx
    delay_ms(1000);                         //Espera tiempo
    printf("E" );                         //Envío valor "E" por Tx
    delay_ms(100);                          //Espera tiempo
    printf("1260" );                       //Envio valor "1260" por Tx
    delay_ms(1000);                         //Espera tiempo
    //print("F");                          //Envío valor "F" por Tx
    //delay_ms(100);                          //Espera tiempo
    //printf("xxxx");                       //Envio valor "xxxx" por Tx
    //delay_ms(1000);                         //Espera tiempo
```

```
}

void a4_soltar()                                        //Función mover brazo a 4 para soltar pieza
{
if((ch=='L') || (ch=='M') || (ch=='N') || (ch=='O'))                    //Si ch = g ó h ó i ó j...
/* Posicion soltar de agente 4
  A 1760
  B 1650
  C 1770
  D 0851
  E 1260
  F XXXX*/

    delay_ms(500);                                      //Espera tiempo
    printf("A" );                                   //Envío valor "A" por Tx
    delay_ms(100);                                      //Espera tiempo
    printf("1710" );                                  //Envio valor "1710" por Tx
    delay_ms(1200);                                     //Espera tiempo
    printf("B" );                                   //Envío valor "B" por Tx
    delay_ms(100);                                      //Espera tiempo
    printf("1750" );                                  //Envio valor "1750" por Tx
    delay_ms(1000);                                     //Espera tiempo
    printf("C" );                                   //Envío valor "C" por Tx
    delay_ms(100);                                      //Espera tiempo
    printf("1700" );                                  //Envio valor "1700" por Tx
    delay_ms(1000);                                     //Espera tiempo
    printf("D" );                                   //Envío valor "D" por Tx
    delay_ms(100);                                      //Espera tiempo
    printf("0850" );                                  //Envio valor "0850" por Tx
    delay_ms(1000);                                     //Espera tiempo
    printf("E" );                                   //Envío valor "E" por Tx
    delay_ms(100);                                      //Espera tiempo
    printf("1260" );                                  //Envio valor "1260" por Tx
    delay_ms(1000);                                     //Espera tiempo
    //print("F");                                   //Envío valor "F" por Tx
    //delay_ms(100);                                    //Espera tiempo
    //printf("xxxx");                                 //Envio valor "xxxx" por Tx
    //delay_ms(1000);                                   //Espera tiempo

}



void main()                                         //Función principal (inicia el programa)
{
  enable_interrupts(global);                                //Habilitar variables globales
  enable_interrupts(int_rda);                               //Habilitar la interrupcion

  printf("Agente 5: Brazo" );                               //Envío mensaje en monitor serial
  delay_ms(5000);                                   //Espera tiempo
  //Brazo se mueve a posicion inicial
  printf("A" );                                   //Envío valor "A" por Tx
  delay_ms(100);                                      //Espera tiempo
  printf("1277" );                                  //Envio valor "1277" por Tx
  delay_ms(1000);                                     //Espera tiempo
  printf("B" );                                   //Envío valor "B" por Tx
  delay_ms(100);                                      //Espera tiempo
  printf("1000" );                                  //Envio valor "1000" por Tx
  delay_ms(1000);                                     //Espera tiempo
```

```
  printf("C" );                                          //Envío valor "C" por Tx
  delay_ms(100);                                           //Espera tiempo
  printf("1000" );                                        //Envio valor "1000" por Tx
  delay_ms(1000);                                          //Espera tiempo
  printf("D" );                                          //Envío valor "D" por Tx
  delay_ms(100);                                           //Espera tiempo
  printf("1250" );                                        //Envio valor "1250" por Tx
  delay_ms(1000);                                          //Espera tiempo
  printf("E" );                                          //Envío valor "E" por Tx
  delay_ms(100);                                           //Espera tiempo
  printf("1260" );                                        //Envio valor "1260" por Tx
  delay_ms(3000);                                          //Espera tiempo
  printf("A" );                                          //Envío valor "A" por Tx
  delay_ms(100);                                           //Espera tiempo
  printf("1277" );                                        //Envio valor "1277" por Tx
  delay_ms(1000);                                          //Espera tiempo

   while(true)                                            //Iniciamos ciclo while
{
    if(ch=='P')                                          //Si ch=P
{                                                      //Entonces...
     delay_ms(100);                                          //Espera tiempo
     inicial_abierta();                                     //Mover brazo posicion inicial abierta
}

    if(ch=='q')                                          //Si ch=q
{                                                      //Entonces...
     delay_ms(100);                                          //Espera tiempo
     inicial_cerrada();                                     //Mover brazo posicion inicial cerrada
}

    if(ch=='a')                                          //Si ch=a
{                                                      //Entonces...
     delay_ms(100);                                          //Espera tiempo
     a1_agarrar();                                         //Mover brazo a agente 1 para agarrar pieza
     delay_ms(100);                                          //Espera tiempo
     inicial_cerrada();                                     //Mover brazo posicion inicial cerrada
     delay_ms(100);                                          //Espera tiempo
    }

    if((ch=='B') || (ch=='C') || (ch=='D') || (ch=='E'))                  //Si ch = B ó C ó D ó E
    {                                                  //Entonces...
     delay_ms(100);                                          //Espera tiempo
     a1_soltar();                                          //Mover brazo a agente 1 para soltar pieza
     delay_ms(100);                                          //Espera tiempo
     inicial_abierta();                                     //Mover brazo posicion inicial abierta
     delay_ms(100);                                          //Espera tiempo
    }

    if(ch=='f')                                          //Si ch=f
    {                                                  //Entonces...
     delay_ms(100);                                          //Espera tiempo
     a2_agarrar();                                         //Mover brazo a agente 2 para agarrar pieza
     delay_ms(100);                                          //Espera tiempo
     inicial_cerrada();                                     //Mover brazo posicion inicial cerrada
     delay_ms(100);                                          //Espera tiempo
    }

    if((ch=='G') || (ch=='H') || (ch=='I') || (ch=='J'))                  //Si ch = G ó H ó I ó J
    {                                                  //Entonces...
     delay_ms(100);                                          //Espera tiempo
```

```
      a2_soltar();                                    //Mover brazo a agente 1 para soltar pieza
      delay_ms(100);                                  //Espera tiempo
      inicial_abierta();                              //Mover brazo posicion inicial abierta
      delay_ms(100);                                  //Espera tiempo
    }

    if(ch=='0')                                       //Si ch=0
    {                                                 //Entonces...
      delay_ms(100);                                  //Espera tiempo
      a3_agarrar();                                   //Mover brazo a agente 3 para agarrar pieza
      delay_ms(100);                                  //Espera tiempo
      inicial_cerrada();                              //Mover brazo posicion inicial cerrada
      delay_ms(100);                                  //Espera tiempo
    }

    if((ch=='R') || (ch=='S') || (ch=='T') || (ch=='U') || (ch=='V') || (ch=='W') || (ch=='X')) //Si ch = R ó S ó T ó U ó V ó W ó X...
    {                                                 //Entonces...
      delay_ms(100);                                  //Espera tiempo
      a3_soltar();                                    //Mover brazo a agente 3 para soltar pieza
      delay_ms(100);                                  //Espera tiempo
      inicial_abierta();                              //Mover brazo posicion inicial abierta
      delay_ms(100);                                  //Espera tiempo
    }

    if(ch=='k')                                       //Si ch=k
    {                                                 //Entonces...
      delay_ms(100);                                  //Espera tiempo
      a4_agarrar();                                   //Mover brazo a agente 4 para agarrar pieza
      delay_ms(100);                                  //Espera tiempo
      inicial_cerrada();                              //Mover brazo posicion inicial cerrada
      delay_ms(100);                                  //Espera tiempo
    }

    if((ch=='L') || (ch=='M') || (ch=='N') || (ch=='O'))          //Si ch = L ó M ó N ó O
    {                                                 //Entonces...
      delay_ms(100);                                  //Espera tiempo
      a4_soltar();                                    //Mover brazo a agente 4 para soltar pieza
      delay_ms(100);                                  //Espera tiempo
      inicial_abierta();                              //Mover brazo posicion inicial abierta
      delay_ms(100);                                  //Espera tiempo
    }
  }
}
```

ANEXO 4

Hoja de datos Pic 16f876A (pág. 1,2,5)

MICROCHIP

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM),
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (V_{REF}) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

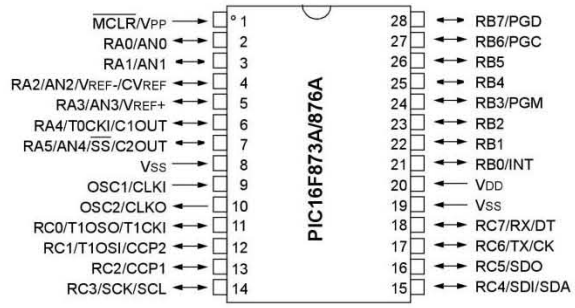
CMOS Technology:

- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

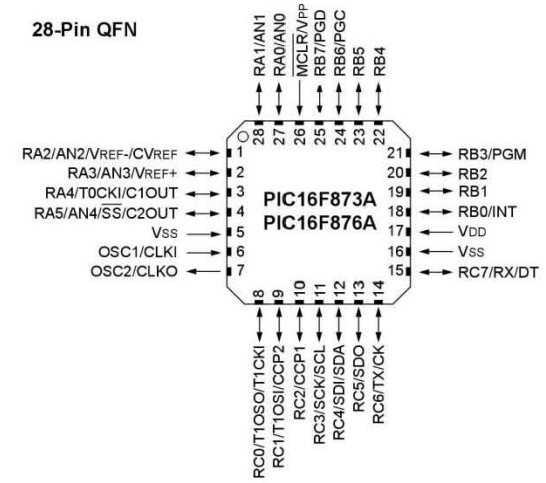
Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^2C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

Pin Diagrams

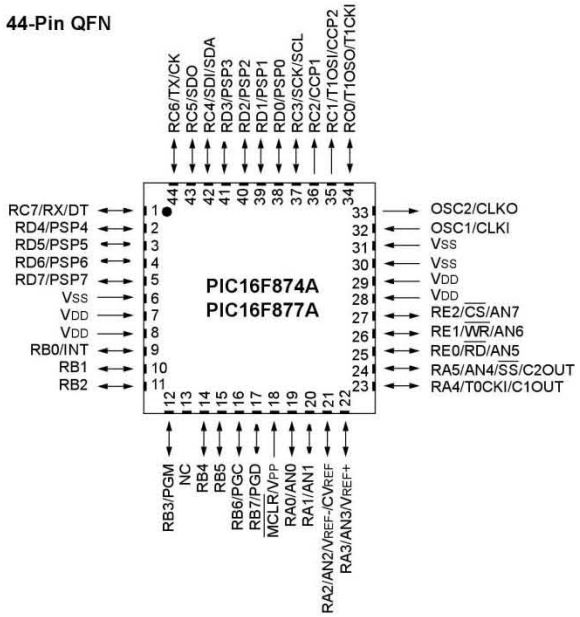
28-Pin PDIP, SOIC, SSOP



28-Pin QFN



44-Pin QFN



1.0 DEVICE OVERVIEW

This document contains device specific information about the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

PIC16F873A/876A devices are available only in 28-pin packages, while PIC16F874A/877A devices are available in 40-pin and 44-pin packages. All devices in the PIC16F87XA family share common architecture with the following differences:

- The PIC16F873A and PIC16F874A have one-half of the total on-chip memory of the PIC16F876A and PIC16F877A
- The 28-pin devices have three I/O ports, while the 40/44-pin devices have five
- The 28-pin devices have fourteen interrupts, while the 40/44-pin devices have fifteen
- The 28-pin devices have five A/D input channels, while the 40/44-pin devices have eight
- The Parallel Slave Port is implemented only on the 40/44-pin devices

The available features are summarized in Table 1-1. Block diagrams of the PIC16F873A/876A and PIC16F874A/877A devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2 and Table 1-3.

Additional information may be found in the PICmicro® Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip web site. The Reference Manual should be considered a complementary document to this data sheet and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

TABLE 1-1: PIC16F87XA DEVICE FEATURES

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

Hoja de datos XBee serie 2 (pág. 4,5,7,8)

1. XBee/XBee-PRO OEM RF Modules

The XBee and XBee-PRO OEM RF Modules were engineered to meet IEEE 802.15.4 standards and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between devices.

The modules operate within the ISM 2.4 GHz frequency band and are pin-for-pin compatible with each other.

1.1. Key Features

Long Range Data Integrity

XBee
- Indoor/Urban: up to 100' (30 m)
- Outdoor line-of-sight: up to 300' (100 m)
- Transmit Power: 1 mW (0 dBm)
- Receiver Sensitivity: -92 dBm

XBee-PRO
- Indoor/Urban: up to 300' (100 m)
- Outdoor line-of-sight: up to 1 mile (1500 m)
- Transmit Power: 100 mW (20 dBm) EIRP
- Receiver Sensitivity: -100 dBm

RF Data Rate: 250,000 bps

Advanced Networking & Security

Retries and Acknowledgements

DSSS (Direct Sequence Spread Spectrum)

Each direct sequence channels has over 65,000 unique network addresses available

Source/Destination Addressing

Unicast & Broadcast Communications

Point-to-point, point-to-multipoint and peer-to-peer topologies supported

Coordinator/End Device operations

Low Power

XBee
- TX Current: 45 mA (@3.3 V)
- RX Current: 50 mA (@3.3 V)
- Power-down Current: < 10 µA

XBee-PRO
- TX Current: 215 mA (@3.3 V)
- RX Current: 55 mA (@3.3 V)
- Power-down Current: < 10 µA

ADC and I/O line support

Analog-to-digital conversion, Digital I/O

I/O Line Passing

Easy-to-Use

No configuration necessary for out-of box RF communications

Free X-CTU Software (Testing and configuration software)

AT and API Command Modes for configuring module parameters

Extensive command set

Small form factor

Free & Unlimited RF-XPert Support

1.1.1. Worldwide Acceptance

FCC Approval (USA) Refer to Appendix A [p59] for FCC Requirements. Systems that contain XBee/XBee-PRO RF Modules inherit MaxStream Certifications.

ISM (Industrial, Scientific & Medical) **2.4 GHz frequency band**

Manufactured under **ISO 9001:2000** registered standards

XBee/XBee-PRO RF Modules are optimized for use in the **United States**, **Canada**, **Australia**, **Israel and Europe**. Contact MaxStream for complete list of government agency approvals.

1.2. Specifications

Table 1-01. Specifications of the XBee/XBee-PRO OEM RF Modules

Specification	XBee	XBee-PRO
Performance		
Indoor/Urban Range	up to 100 ft. (30 m)	Up to 300' (100 m)
Outdoor RF line-of-sight Range	up to 300 ft. (100 m)	Up to 1 mile (1500 m)
Transmit Power Output (software selectable)	1mW (0 dBm)	60 mW (18 dBm) conducted, 100 mW (20 dBm) EIRP*
RF Data Rate	250,000 bps	250,000 bps
Serial Interface Data Rate (software selectable)	1200 - 115200 bps (non-standard baud rates also supported)	1200 - 115200 bps (non-standard baud rates also supported)
Receiver Sensitivity	-92 dBm (1% packet error rate)	-100 dBm (1% packet error rate)
Power Requirements		
Supply Voltage	2.8 – 3.4 V	2.8 – 3.4 V
Transmit Current (typical)	45mA (@ 3.3 V)	If PL=0 (10dBm): 137mA(@3.3V), 139mA(@3.0V) PL=1 (12dBm): 155mA (@3.3V), 153mA(@3.0V) PL=2 (14dBm): 170mA (@3.3V), 171mA(@3.0V) PL=3 (16dBm): 188mA (@3.3V), 195mA(@3.0V) PL=4 (18dBm): 215mA (@3.3V), 227mA(@3.0V)
Idle / Receive Current (typical)	50mA (@ 3.3 V)	55mA (@ 3.3 V)
Power-down Current	< 10 µA	< 10 µA
General		
Operating Frequency	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960" x 1.297" (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)	-40 to 85° C (industrial)
Antenna Options	Integrated Whip, Chip or U.FL Connector	Integrated Whip, Chip or U.FL Connector
Networking & Security		
Supported Network Topologies	Point-to-point, Point-to-multipoint & Peer-to-peer	
Number of Channels (software selectable)	16 Direct Sequence Channels	12 Direct Sequence Channels
Addressing Options	PAN ID, Channel and Addresses	PAN ID, Channel and Addresses
Agency Approvals		
United States (FCC Part 15.247)	OUR-XBEE	OUR-XBEEPRO
Industry Canada (IC)	4214A XBEE	4214A XBEEPRO
Europe (CE)	ETSI	ETSI (Max. 10 dBm transmit power output)*
Japan	n/a	005NYCA0378 (Max. 10 dBm transmit power output)**

* When operating in Europe: XBee-PRO RF Modules must be configured to operate at a maximum transmit power output level of 10 dBm. The power output level is set using the PL command. The PL parameter must equal "0" (10 dBm).

Additionally, European regulations stipulate an EIRP power maximum of 12.86 dBm (19 mW) for the XBee-PRO and 12.11 dBm for the XBee when integrating high-gain antennas.

** When operating in Japan: Transmit power output is limited to 10 dBm. A special part number is required when ordering modules approved for use in Japan. Contact MaxStream for more information [call 1-801-765-9885 or send e-mails to sales@maxstream.net].

Antenna Options: The ranges specified are typical when using the integrated Whip (1.5 dBi) and Dipole (2.1 dBi) antennas. The Chip antenna option provides advantages in its form factor; however, it typically yields shorter range than the Whip and Dipole antenna options when transmitting outdoors. For more information, refer to the "XBee Antenna" application note located on MaxStream's web site (http://www.maxstream.net/support/knowledgebase/article.php?kb=153).

1.5. Pin Signals

Figure 1-03. XBee/XBee-PRO RF Module Pin Numbers
(top sides shown - shields on bottom)

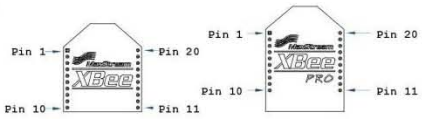


Table 1-02. Pin Assignments for the XBee and XBee-PRO Modules
(Low-asserted signals are distinguished with a horizontal line above signal name.)

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / **CONFIG**	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	**RESET**	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	**CTS** / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / **SLEEP**	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	**RTS** / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

* Function is not supported at the time of this release

Design Notes:

- Minimum connections: VCC, GND, DOUT & DIN
- Minimum connections for updating firmware: VCC, GND, DIN, DOUT, RTS & DTR
- Signal Direction is specified with respect to the module
- Module includes a 50k Ω pull-up resistor attached to **RESET**
- Several of the input pull-ups can be configured using the PR command
- Unused pins should be left disconnected

1.6. Electrical Characteristics

Table 1-03. DC Characteristics (VCC = 2.8 - 3.4 VDC)

Symbol	Characteristic	Condition	Min	Typical	Max	Unit
V_{IL}	Input Low Voltage	All Digital Inputs	-	-	0.35 * VCC	V
V_{IH}	Input High Voltage	All Digital Inputs	0.7 * VCC	-	-	V
V_{OL}	Output Low Voltage	I_{OL} = 2 mA, VCC >= 2.7 V	-	-	0.5	V
V_{OH}	Output High Voltage	I_{OH} = -2 mA, VCC >= 2.7 V	VCC - 0.5	-	-	V
II_{IN}	Input Leakage Current	V_{IN} = VCC or GND, all inputs, per pin	-	0.025	1	µA
II_{OZ}	High Impedance Leakage Current	V_{IN} = VCC or GND, all I/O High-Z, per pin	-	0.025	1	µA
TX	Transmit Current	VCC = 3.3 V	-	45 (XBee) / 215 (PRO)	-	mA
RX	Receive Current	VCC = 3.3 V	-	50 (XBee) / 55 (PRO)	-	mA
PWR-DWN	Power-down Current	SM parameter = 1	-	< 10	-	µA

Table 1-04. ADC Characteristics (Operating)

Symbol	Characteristic	Condition	Min	Typical	Max	Unit
V_{REFH}	VREF - Analog-to-Digital converter reference range		2.08	-	V_{DDAD}	V
I_{REF}	VREF - Reference Supply Current	Enabled	-	200	-	µA
		Disabled or Sleep Mode	-	< 0.01	0.02	µA
V_{INDC}	Analog Input Voltage[1]		V_{SSAD} - 0.3	-	V_{DDAD} + 0.3	V

1. Maximum electrical operating range, not valid conversion range.

Table 1-05. ADC Timing/Performance Characteristics[1]

Symbol	Characteristic	Condition	Min	Typical	Max	Unit
R_{AS}	Source Impedance at Input[2]		-	-	10	kΩ
V_{AIN}	Analog Input Voltage[3]		V_{REFL}		V_{REFH}	V
RES	Ideal Resolution (1 LSB)[4]	$2.08V \le V_{DDAD} \le 3.6V$	2.031	-	3.516	mV
DNL	Differential Non-linearity[5]		-	±0.5	±1.0	LSB
INL	Integral Non-linearity[6]		-	±0.5	±1.0	LSB
E_{ZS}	Zero-scale Error[7]		-	±0.4	±1.0	LSB
F_{FS}	Full-scale Error[8]		-	±0.4	±1.0	LSB
E_{IL}	Input Leakage Error[9]		-	±0.05	±5.0	LSB
E_{TU}	Total Unadjusted Error[10]		-	±1.1	±2.5	LSB

1. All ACCURACY numbers are based on processor and system being in WAIT state (very little activity and no IO switching) and that adequate low-pass filtering is present on analog input pins (filter with 0.01 µF to 0.1 µF capacitor between analog input and VREFL). Failure to observe these guidelines may result in system or microcontroller noise causing accuracy errors which will vary based on board layout and the type and magnitude of the activity.
Data transmission and reception during data conversion may cause some degradation of these specifications, depending on the number and timing of packets. It is advisable to test the ADCs in your installation if best accuracy is required.
2. R_{AS} is the real portion of the impedance of the network driving the analog input pin. Values greater than this amount may not fully charge the input circuitry of the ATD resulting in accuracy error.
3. Analog input must be between V_{REFL} and V_{REFH} for valid conversion. Values greater than V_{REFH} will convert to $3FF.
4. The resolution is the ideal step size or 1LSB = $(V_{REFH}-V_{REFL})/1024$
5. Differential non-linearity is the difference between the current code width and the ideal code width (1LSB). The current code width is the difference in the transition voltages to and from the current code.
6. Integral non-linearity is the difference between the transition voltage to the current code and the adjusted ideal transition voltage for the current code. The adjusted ideal transition voltage is $(Current\ Code-1/2)*(1/((V_{REFH}+E_{FS})-(V_{REFL}+E_{ZS})))$.
7. Zero-scale error is the difference between the transition to the first valid code and the ideal transition to that code. The Ideal transition voltage to a given code is $(Code-1/2)*(1/(V_{REFH}-V_{REFL}))$.
8. Full-scale error is the difference between the transition to the last valid code and the ideal transition to that code. The ideal transition voltage to a given code is $(Code-1/2)*(1/(V_{REFH}-V_{REFL}))$.
9. Input leakage error is error due to input leakage across the real portion of the impedance of the network driving the analog pin. Reducing the impedance of the network reduces this error.
10. Total unadjusted error is the difference between the transition voltage to the current code and the ideal straight-line transfer function. This measure of error includes inherent quantization error (1/2LSB) and circuit error (differential, integral, zero-scale, and full-scale) error. The specified value of E_{TU} assumes zero E_{IL} (no leakage or zero real source impedance).