MODELADO DE UN ALGORITMO DE COMUNICACIÓN USANDO REDES DE PETRI PARA LA INTERFAZ GEN 2 RFID READER

ISRAEL GARCíA HERNÁNDEZ

Asesor

M.C. Walter Torres Robledo

Revisores

Dr. Héctor Guerra Crespo

M.C. José Alberto Morales Mancilla

INSTITUTO TECNOLÓGICO DE TUXTLA GUTIÉRREZ

RESUMEN

En la actualidad el desarrollo de dispositivos RFID ha traído ciertos problemas de compatibilidad y uso, así como la aplicación a sistemas ya existentes. El principal problema es la estandarización y manejo múltiples dispositivos, por lo que la presente investigación se enfoca al análisis de un dispositivo RFID Reader(GIO216001) y su control, mediante una interfaz de control "concentrador" basada en microcontrolador.

La interfaz tiene la posibilidad de manejar varios dispositivos mediante la aplicación de un algoritmo, aplicado y simulado a una red de dispositivos RFID.

El algoritmo es fundamental en la comunicación entre los dispositivos RFID(GIO216001) y una PC, ayuda reducir la codificación en el programa, resultando una comunicación óptima y rápida.

El protocolo de comunicación es modelado mediante la herramienta de modelado gráfico y matemático llamada Redes de Petri (Petri Nets), identificando la sincronización, secuencia, paralelismo y toma de decisiones del sistema general de comunicación entre varios dispositivos.

Índice

Índice de figuras

Índice de cuadros

1. Introducción

1.1. Antecedentes

Con el paso de los años la tecnología ha evolucionado rápidamente trayendo consigo nuevas implementaciones y dispositivos, una tecnología que ha alcanzado nuevas fronteras es la tecnología de radiofrecuencias, anteriormente usada para la transmisión de radio y televisión ahora usada en el control de personal, inventarios, metas de competencias, etc. La nueva implementación de la tecnología se le llamó RFID por sus siglas en ingles (Radio Frequency IDentification) la cual ha tomado una fuerte presencia en los mercados de Europa y Asia con el desarrollo de antenas RFID y Tag (Figura 1). Lamentablemente aún existe mucho camino por recorrer debido a que la aplicación de la tecnología RF debe ser estandarizada en todos sus aspectos.

Figura 1: Antenna y Tag RF ID

La mayoría de los dispositivos RFID tiene como vía de conectividad los cables RS232, pero como bien se sabe, las PC's sólo cuentan con un único socket, por lo que es imposible conectar más de un dispositivo, y aunque se puede implementar convertidores de serial a USB y eso convertidores conectarlos a un dispositivo multipuerto USB (Figura 2), esto sacrifica el tiempo de comunicación entre los dispositivos y la PC que los controla.

Figura 2: Adaptadores y/o convertidores

1.2. Planteamiento del Problema

Debido al creciente uso de la tecnología RFID y la falta de estándares de comunicación y dispositivos de control, trae como consecuencia una deficiencia en el manejo de los dispositivos RFID, así como de la información que se obtiene, ejemplo claro es la pérdida de información, errores de acceso a un dispositivo, pérdida de tiempo al obtener información, mal funcionamiento e incluso dificulta su implementación.

Las diferentes empresas (GAO, EPC global, etc) a nivel mundial han implementado diferentes estándares existentes como RS-232 y USB para la comunicación entre los dispositivos RFID y una PC, la mayoría de las empresas venden soluciones completas (paquetes o kit), los que únicamente constan de los dispositivos RFID y su software; si uno quiere agregar un dispositivo RFID extra o uno que sea de diferente marca o modelo, es posible que no funcione adecuadamente, las empresas implementan variaciones en la construcción de sus dispositivos y el desarrollo de sus aplicaciones, algunos dispositivos a pesar de utilizar el estándar RS-232 para comunicarse utilizan también un método de encriptación de la trama o la forma de acceso, exclusivamente de una marca; las aplicaciones que desarrollan, son software desarrollados únicamente para un tipo de dispositivo RFID lo que impide en ocasiones que se use otro modelo incluso de la misma marca. Si se desarrollara una aplicación que permita la conexión de varios

dispositivos RFID tendría las siguientes complicaciones en su funcionamiento:

- Conectividad física: la conectividad física de los dispositivos RFID son RS-232 o USB (versión 1) y la mayoría de las PC's del mercado tiene 1 conector RS-232 y 4 conectores USB, por lo que la aplicación solo puede tener 5 dispositivos conectados. Si se llegase a conectar un Hub de USB podría extenderse hasta 10 antenas, pero se obtendría de pérdida de tiempo en el momento enviar y recibir información.
- Lenguaje: cada uno de los dispositivos tienen distintos comandos, por ejemplo leer o escribir, pero no todos tienen el mismo número de comandos, ni mucho menos la misma palabra binaria, corresponde a la misma orden que en otro dispositivo, por ejemplo 0101010 puede ser *"iniciar lectura"*, pero en otro dispositivo puede ser *"apagar el dispositivo"*, incluso puede que dicha palabra no sea utilizada. En el mayor de los casos puede ser que el dispositivo encripte la información.
- Conectividad de lógica: el proceso de un dispositivo RFID puede variar entre uno y otro, por ejemplo a continuación se muestran tres procesos de dispositivos RFID :
  - Dispositivo A: encendido-> preparación_configuración-> inicia_lectura-> obtención-> envió_de_información-> terminar_la_conexión.
  - Dispositivo B: encendido-> envió_de_información-> envió_de_información-> envió_de_información-> envió_de_información->.....
  - Dispositivo C: encendido-> inicia_lectura-> obtención-> envió_de_información-> obtención-> envió_de_información->.....

Por lo anterior no se puede llegar a realizar una interfaz que satisfaga todos y cada una de las características únicas de las antenas; por lo que se decidió enfocar la investigación al desarrollar un dispositivo que sirva de intermediario entre varios dispositivos RFID y una PC, para ellos se utiliza el dispositivo Gen 2 RFID Reader (GIO216001) como base; se analizará y generará un algoritmo de comunicaciones que permita el control óptimo de los dispositivos Gen 2 RFID Reader (GIO216001) de forma simultánea.

Este documento describe sobre la parte del modelado del algoritmo que incluye reglas para la comunicación entre PC y un dispositivo central que permite administrar los dispositivos RFID con el objetivo de solucionar el problema de manipular varios dispositivos RFID de forma simultánea.

1.3. Objetivos

1.3.1. Objetivo General

Validar el algoritmo de comunicación entre una estación central y una interfaz remota RFID, mediante el modelado gráfico basado en Redes de Petri.

1.3.2. Objetivo Especifico

1. Analizar las características, configuración, entradas, salidas, etc, del dispositivo "Gen 2 RFID Reader" (GAO216002).
2. Desarrollar un conjunto de reglas para la comunicación entre el dispositivo "Gen 2 RFID Reader" (GAO216002) y el host.
3. Modelar el algoritmo usando Redes de Petri que permita relacionar la interconexión de diferentes antenas de forma simultáneas.
4. Validar el funcionamiento del algoritmo en conjunción de las reglas mediante su programación y simulación.

1.4. Hipótesis

El desarrollo del algoritmo de comunicación usando Redes de Petri como método de modelado, permitirá la administración de varios dispositivos RFID "Gen 2 RFID Reader" de forma sincronizada dando como resultado la optimización de obtención de información de manera simultánea, mejorando los tiempos de respuesta de los dispositivos que se conecten.

1.5. Justificación

El proceso de control de dispositivos RFID que realizan las aplicaciones (control de acceso, manejo de inventario, etc) del mercado actual es muy deficiente, debido a que la mayoría de las aplicaciones, sólo soportan un dispositivo RFID conectado y las aplicaciones que soportan más de un dispositivo conectado, suelen ser lentos al recibir o enviar información.

El algoritmo permite tener una comunicación entre un dispositivo concentrador y la PC, teniendo un control óptimo de los dispositivos RFID, solucionando el problema de conectividad lógica.

Para el desarrollo del algoritmo se implementa un modelo basado en Redes de Petri las cuales son, un efectivo método de modelado que permite realizar diseño de algoritmos de una forma rápida y eficiente. Las Redes de Petri consisten en un conjunto de grafos con el que se puede modelar el comportamiento y la estructura de un sistema.

El resultado de la implementación de las redes de Petri permite crear un conjunto de reglas afines a mejorar los tiempos de lectura/escritura y controlar el dispositivo central mediante un software de computadora.

2. Fundamentos Teóricos

2.1. Estado del Arte

A continuación se presentan algunos trabajos que comparten similitudes con el presente trabajo.

En marzo de 2011 Almeida P. Marco, Roldán M. Elsa y M. Soraya estudiantes de la Universidad Politécnica Nacional de Quito Ecuador realizaron el modelado e implementación del protocolo de comunicaciones para el control de un brazo robot, usando como base la arquitectura de protocolo planteado por Holzmann (véase 2.2.1), el cual define cinco elementos que constituyen a el protocolo (servicio, ambiente, vocabulario, codificación y reglas) y como complemento en el área del desarrollo, la metodológica de desarrollo de software en espiral (requerimientos, análisis de riesgos, planteamiento, diseño primer prototipo y pruebas) para reforzar el entendimiento del proceso mejora sus resultados. El medio empleado para la transferencia de información fue Bluetooth [16].

En noviembre de 2011 Granados Rojas Benito, Jiménez Saucedo Mario A., Vallejo Alarcón Manuel A., González Navarro Yesenia E., Villarreal Cervantes Miguel G. y Corona Ramírez Leonel G presenta un protocolo de comunicación anillo para el control de un robot móvil modular, usando como herramienta de modelado para el diseño del protocolo de comunicación el diagrama de flujo; donde se plantea una topología anillo unidireccional, es decir una serie de módulos conectados entre sí, uno a uno, emisor conectado a receptor formando un ciclo, con n cantidad de módulos esclavos m; para la transmisión de información entre módulos, se propuso usar un protocolo estándar de comunicaciones RS-232 el cual se usa para transferir paquetes de datos de un byte. En cuanto la transmisión cada módulo recibe dos bytes, los almacena temporalmente y revisa si el primer byte corresponde a alguna dirección existente, si no es así reenvía el byte de dirección y el byte de datos; en el caso de ser alguna dirección asignada a él. Todo módulo en el sistema debe de ser capaz de poder enviar y recibir transmisiones bajo el protocolo serial RS-232, por ello, cada módulo cuenta con un micro-controlador, el cual se encarga de la gestión de transmisiones, interpretación de instrucciones recibidas y ejecución de dichas instrucciones[14].

En julio de 2009 Mario L. Ruiz, Francisco Vázquez da a conocer en la revista actitud preventiva en conjunto con la Universidad de Córdoba, el artículo control de un lector RFID mediante microcontroladores usando como método de control. En él se define como medio de transmisión el estándar RS-232 y como interfaz de control un PIC16F28A. También muestra los elementos que conforman un sistema de RFID (véase 2.3.2), describe los diferentes modos de lecturas que tiene un dispositivo RFID. El principal aporte del artículo es el análisis que se hace a la trama de transmisión y recepción de cada uno de los modos de lectura (normal, line, executive y gate), el cual permite el diseño del conjunto de reglas, que es representado mediante un diagrama de flujo para posteriormente ser

implementado[11].

En la Universidad Autónoma de Occidente en enero de 2008 Karol Nataly Benavides Lopez realiza el proyecto de un módulo de comunicación con dispositivos RFID para la empresa Robotek. El proyecto se dirigió al desarrollo de una interfaz de prueba que manejara dispositivos RFID de diferentes compañías y tag de diferentes estándares. En el dispositivo se usó como base el integrado HTRC11001T que combina todo el hardware de un dispositivo RFID (lectura/escritura), como interfaz de control se usó el Pic PIC18F6720 y como medio de trasmisión el estándar RS-232. El modelado de la interfaz se realizó mediante el usó de diagramas de bloque y se implementó diagramas de flujo para modelar las reglas y procedimientos para el diseño del protocolo de comunicaciones entre la interfaz y la PC[15].

En el 2001 Néstor Peña y Mario E. Salazar P. publican su artículo llamado aplicación de Redes de Petri a la evaluación de desempeño de sistemas de comunicaciones. El artículo aborda brevemente los elementos que conforman las Redes de Petri, resaltando las Redes de Petri estocásticas generalizadas (GSPN) y MRSPN (MARKOV REGENERATIVE STOCHASTIC PETRI NETS).

Se realizó la evaluación a los protocolos de comunicaciones de "Para y Espera" y el "Aloha Puro", la cual se enfocó en el proceso de trasmisión de paquetes con respecto al tiempo. El segundo análisis se llevó a cabo a protocolos TSP y DSP que son para dispositivos autónomos inteligentes, con la finalidad de mejorar el tiempo de respuesta a los fallos [13].

En abril del 2000 Diego R. Llanos Ferraris desarrolla el protocolo VSR-COMA, un protocolo COMA de gestión distribuida con reemplazo para sistemas multicomputador unidos por un bus común, que es también una evolución del protocolo COMABC.

Este protocolo trabaja en sistemas multicomputadoras débilmente acoplados de memoria compartida distribuida, es utilizado en una red de estaciones de trabajo que permiten acelerar la ejecución de aplicaciones paralelas a bajo costo y en la utilización de un espacio compartido de direcciones entre los diferentes nodos que posibilita el uso de un paradigma de programación de variables compartidas.

Fue diseñado para un sistema multicomputador y tiene como principal finalidad minimizar el número de accesos a datos situados en nodos remotos disminuyendo el coste asociado a la transferencia de un bloque de datos de un nodo a otro y la reducción el tiempo de acceso a los datos del espacio compartido de memoria. El protocolo VSR-COMA se ha verificado utilizando Redes de Petri coloreadas, lo que ha permitido eliminar errores de diseño y verificar su funcionamiento en las primeras etapas de su desarrollo lo que permitió mantener la coherencia de los datos presentes en las memorias de cada uno de los nodos y reduciendo el tiempo [17].

El presente trabajo se relaciona con aspectos técnicos-científicos de fuentes confiables escritas en este apartado, resaltando el modelado del protocolo de comunicación mediante Redes de Petri.

La aplicación de las Redes de Petri al verificar diversos Protocolos y demostrar su eficacia al simularlos.

2.2. Protocolo

Se debe recordar que un resultado de la tesis es lograr la comunicación entre los dispositivos RFID y la PC, por lo que se debe abordar el tema referente al diseño de los protocolos, ya que los protocolos son parte fundamental en la comunicación de cualquier sistema.

2.2.1. ¿Qué es un protocolo?

Inicialmente se debe entender claramente el concepto de protocolo, para ello se hace referencia a dos autores los cual definen al protocolo de la siguiente manera.

Un protocolo es un conjunto de reglas que tienen que ser seguida en el curso de alguna actividad. Originalmente, el término fue utilizado únicamente por las actividades humanas, especialmente las de un tipo un tanto formal.

De forma general, es la comunicación entre ordenadores que tiene lugar al intentar el intercambio de datos o información codificada que depende del sistema en cuestión.Se puede considerar este intercambio como comunicación elemental, en cada uno de los cuales un mensaje es transferido. En función del sistema, un mensaje puede ser una señal electrónica única, o una gran cantidad de datos. De forma general, se utiliza el término mensaje para abarcar tanto el contenido general así como la codificación.

El comando de un protocolo de comunicaciones se define como el conjunto de reglas para ordenar cada tipo de mensajes que se intercambiará [2].

Otra definición usada incluso en estos días debido a lo acertada que es, es la de Holzmann, el cual describe protocolo como el conjunto de cinco elementos

- El servicio que es proporcionado por el protocolo : en esta etapa se debe definir el propósito del protocolo, que va a hacer el protocolo de una forma clara y bien descrita.
- El medio ambiente en el que se ejecuta el protocolo : el medio ambiente en el que el protocolo se va a ejecutar consta como mínimo de dos usuarios que usarán el servicio de transferencia de archivos y un canal.

- El vocabulario de los mensajes utilizados para implementar el protocolo: define tres tipos de mensajes ACK para un mensaje combinado con un reconocimiento positivo, NAK un mensaje combinado con un reconocimiento negativo, y se equivocan para un mensaje con un error de transmisión.
- La codificación de cada mensaje en el vocabulario: cada mensaje se compone de un campo de control, para la identificación del tipo de mensaje y un campo de datos con el código de carácter.
- Las reglas de procedimiento para asegurar la coherencia de el intercambio de mensajes:Las reglas de procedimiento para el protocolo se describe de manera informal. Para formalizar estas reglas, podemos utilizar los diagramas de transición de estado, diagramas de flujo, expresiones algebraicas, las descripciones de la forma de programas, etc[5].

Al conjunto de estos cinco elemento se le llama Arquitectura de protocolo, otros autores manejan un número diferente de elementos como el que se muestra a continuación:

- Aspectos electrónicos: los cables, conectores, las señales, etc.
- La manera de agrupar los bits para formar paquetes y la de controlar que no se produzcan errores en la trasmisión.
- La identificación de los ordenadores dentro de una red y la manera de conseguir que la información que genera un ordenador llegue a quien se pretende[4].

2.2.2. Problemas Comunes de Protocolo

Durante el diseño del protocolo pueden aparecer problemas genéricos los cuales pueden ser solucionados anticipadamente, esto permite crear una lista que tenga la descripción de los problemas más comunes durante el diseño de el protocolo. A continuación se mencionan los problemas más comunes:

Direccionamiento
A menos que se reutiliza la dirección de algún servicio de la capa inferior, un servicio por lo general incorpora algún tipo de nombres o direcciones de esquema para identificar los destinos de los mensajes. La comunicación habitualmente se destina a ser de doble vía, por lo que fuentes de mensajes también son típicamente los destinos de mensajes con el direccionamiento de los suyos. la mayoría de los protocolos incluyen una dirección de origen, así como una dirección de destino para que el destinatario sepa dónde enviar la respuesta estructurada.

Petición de mensaje
El servicio se dedica a la fragmentación y el montaje para la formulación del mensaje, si el sustrato no conserva el orden de los mensajes, es necesario ordenar los fragmentos para preservar el orden de los mensajes. Este proceso requiere la

adición de un esquema de numeración adecuados para mensajes y/o sus fragmentos.

Manejo de errores
Este problema reside en la creación de una lista de error-causa-solución para poder analizar y evaluar todos los posibles errores ocasionados durante la transmisión y aplicar alguna estrategia para su solución .

Control de flujo
Los datos no siempre se pueden recibir tan rápido como se envía. Si no hay memoria de almacenamiento intermedio para almacenar los mensajes entrantes, el mensaje se pierde. En lugar de que envíe a la tasa máxima y recuperar el mensaje, el destinatario puede guardar la información del emisor y de su disposición a aceptar los datos entrantes.

Seguridad
Se refiere específicamente a la autentificación, la integridad, y privacidad en la trasmisión. Por lo general se refiere al método de encriptación implementado, verificar la autentificación del servicio, y la integridad de la transmisión de la información[3].

2.2.3. Modelo OSI

El modelo básico de referencia OSI, afronta el problema de las comunicaciones de datos y las redes informáticas dividiéndolo en niveles. Cada participante de la comunicación incorpora como mínimo uno de los mismos, y los equipos terminales los incorporan todos.

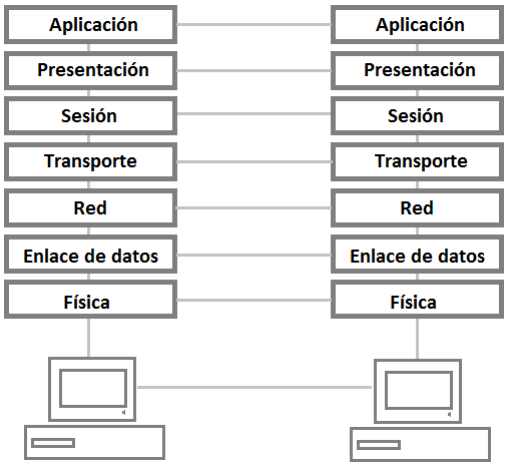


Figura 3: Capas de modelo OSI

De forma horizontal la comunicación sólo se da entre niveles homónimos mientras que de forma vertical se da entre niveles adyacentes de un mismo sistema.

2.2.4. Capas del Modelo OSI

Debido a que el algoritmo a modelar será implementado en una placa con un PIC 16F677A se tomaron las consideraciones de sólo usar las siguientes capas del modelo OSI:

Capa Física
El nivel físico se encarga de las tareas de trasmisión física de la señal eléctrica entre los diferentes sistemas. Las limitaciones del nivel físico imponen otras al resto del sistema: por un lado, la limitación de la velocidad de trasmisión y por otro, hacen aparecer una probabilidad de error, el porcentaje de bit erróneos que llegan a destino.

La primera es casi insalvable partiendo de un medio de transmisión, puesto que los parámetros físicos de este último impone un límite superior no superable por medio de una mejora tecnológica. Los medios de trasmisión poseen una capacidad de transmisión acotada y la electrónica que utilizamos para llevar a cabo las trasmisiones en los mismos pueden mejorar la velocidad de transmisión, pero no puede superar este límite.

La probabilidad de error puede corregirse por medio de algoritmos y protocolos si su valor esta contenido. Si las cotas de error son inferiores al 1 %, se puede reducir su impacto si se agrupan los bits en pequeños bloques de datos y se añade software en el receptor que vigile la corrección de los datos recibidos.

Nivel de enlace
El nivel de enlace proporciona un servicio similar al nivel físico, mejorando las características de fiabilidad de la transmisión. Añade bit adicionales a los que forman el mensaje para poder detectar errores de trasmisión en el mismo y poder pedir su retransmisión. Para ello, es preciso conferir una estructura a los bits: se agrupan en bloques denominados tramas , que contienen los bits de mensaje, los bits añadidos para detectar errores y diferentes capas de control tales como el número de trama.

Además de la tarea de control de errores, el nivel de enlace lleva a cabo otra importante, el control del flujo.

El receptor debe procesar las tramas a medida que las recibe, en algunos casos este proceso comporta un gasto de mínimo de tiempo, teniendo en cuenta la velocidad de trasmisión; sin embargo en otros, puede ser costoso. En esta situación el receptor necesita un mecanismo que notifique al transmisor que momentáneamente detenga la trasmisión y así poder llevar acabo la tarea[5].

2.2.5. La Norma RS232C

La interfaz RS-232 es de tipo serie, para sistemas de baja velocidad y asíncrona. Características:

- El estándar prevé hasta 20kbps de velocidad, aunque hay implementaciones que permiten alcanzar velocidades máximas de trasmisión que van de 100kbps a 1Mbps.
- El estándar también define un alcance de hasta 15m.
- La trasmisión suele ser asíncrona, aunque el estándar prevé la posibilidad de trabajar con circuitos de reloj y adoptar la forma síncrona. Esta forma , sin embargo, no acostumbra a implementarse en circuitos estándar.

Transmisión Asíncrona

El término transmisión asíncrona simple alude al hecho de que no se trasmite el reloj de bit con un circuito específico. Es un término que sólo tiene sentido en interfaces RS-232, por el bit de arranque. La codificación que utiliza RS-232 es del tipo NRZ, es decir, la señal toma un valor positivo concreto durante todo el tiempo del bit, si el bit es 1, y si el bit es 0, toma el valor 0. El sistema de transmisión asíncrona permite al receptor no perder el sincronismo de bits mientras el error no supere el 10 %. Para poder funcionar, los paquetes se agrupan en caracteres, que puede tener entre 5 y 8 bits de datos. Para definir este mecanismo de sincronización, tenemos que definir dos nuevos símbolos, que sumados a los dos que ya tenemos para los bits 1 y 0 hacen un total de cuatro símbolos.

- Bytes 1: bit codificado con el símbolo positivo.
- Bytes 0: bit codificado con el símbolo negativo.
- Estado de inactividad: se usa en el arranque y al final de cada carácter.
- Señal de arranque : símbolo que permite separar el estado de inactividad de un carácter y saber dónde empieza un carácter.

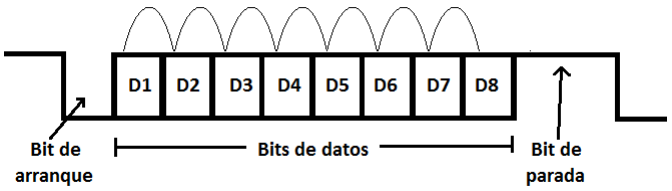


Figura 4: Trama

Una vez se ha enviado el bit de arranque, se envían los bits que forman el carácter de manera consecutiva. Cuando todos los bits han sido enviados, la línea se pone en estado de inactividad, para enviar el carácter siguiente , sólo hay que hacer preceder sus bits por el bit de arranque [5].

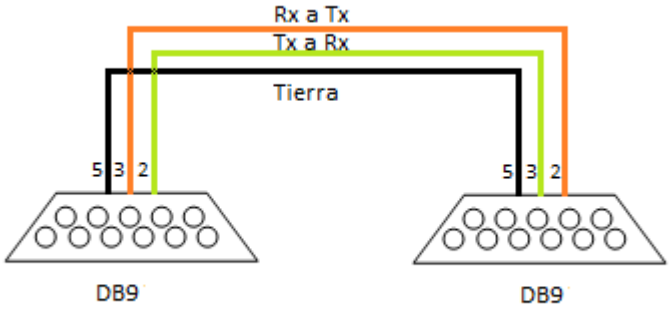


Figura 5: RS- 232

En la figura se muestra la configuración del cable para estándar RS-232, se observa el intercambio entre el pin 2 (salida) al pin 3 (entrada) y el pin 5 al pin 5 del segundo conector.

2.3. Los Dispositivos RFID

La tecnología RFID principalmente las antenas son dentro del desarrollo de la tesis, el objeto clave por qué son los dispositivos a manejar y la razón por la que se pretende desarrollar el algoritmo, por lo cual se muestra enseguida un resumen de los puntos más significativos como, qué es RFID, cómo funciona, historia, etc.

2.3.1. Historia

Es complicado establecer un punto de partida claro para la tecnología RFID. La historia de la RFID aparece entrelazada con la del desarrollo de otras tecnologías de comunicaciones a la largo del siglo XX: ordenadores, tecnologías de la información, teléfonos móviles, redes inalámbricas, comunicaciones por satélite, GPS, etc.

La existencia actual de aplicaciones viables basadas en RFID se debe al desarrollo progresivo de tres áreas tecnológicas principales:

- Electrónica de radiofrecuencia. Necesaria para el desarrollo de las antenas y los sistemas de radiofrecuencia presentes en las etiquetas e interrogadores RFID.

- Tecnologías de la información. En su vertiente de computación (en el lector, en la propia etiqueta y en el sistema de información asociado) y en su vertiente de comunicaciones para el envío de información (entre etiqueta y lector, y entre lector y sistema de información asociado).
- Tecnología de materiales. Necesaria para el abaratamiento de las etiquetas.

RFID no es una tecnología nueva, sino que lleva existiendo desde 1940. Durante la Segunda Guerra Mundial, los militares estadounidenses utilizaban un sistema de identificación por radiofrecuencia para el reconocimiento e identificación a distancia de los aviones: "Friend or Foe" (amigo o enemigo). Acabada la guerra, los científicos e ingenieros continuaron sus investigaciones sobre estos temas. Harry Stockman publicó un artículo en los Proceedings of the IRE titulado "Communications by Means of Reflected Power"(Actas del IRE, pp. 1196-1204, octubre de 1948), que se puede considerar como la investigación más cercana al nacimiento de la RFID.

A partir de ese momento, el desarrollo de la tecnología RFID ha sido lento pero constante. Durante la década de los 50 se realizaron multitud de estudios relacionados con la tecnología, principalmente orientados a crear sistemas seguros para su aplicación en minas de carbón, explotaciones petrolíferas, instalaciones nucleares, controles de acceso o sistemas antirrobo. Durante esta época se publicaron dos artículos importantes: "Applications of Microwave Homodyne", de F. L. Vernon (1952), y "Radio Transmission Systems with Modulatable Passive Responders", de D. B. Harris.

En los años 60 se profundizó en el desarrollo de la teoría electromagnética y empezaron a aparecer las primeras pruebas de campo, como por ejemplo, la activación remota de dispositivos con batería, la comunicación por radar o los sistemas de identificación interrogación-respuesta. Aparecieron las primeras invenciones convocación comercial, como "Remotel y Activated Radio Frequency Powered Devices", de Robert Richardson, "Communication by Radar Beams" de Otto Rittenback, "Passive Data Transmisión Rechniques Utilizing Radar Beams" de J. H. Vogelman, y "Interrogator-Responder Identification System", de J. P. Vinding.

Así mismo, comenzaron las primeras actividades comerciales. Se fundaron Sensormatic y Checkpoint, que junto con otras compañías, desarrollaron un equipo de vigilancia electrónica anti-intrusión denominado EAS (Electronic Article Surveillance). EAS fue el primer desarrollo de RFID y el que indiscutiblemente se ha venido utilizando más ampliamente. Fue el preludio de la explosión de esta tecnología.

Durante los años 70 desarrolladores, inventores, fabricantes, centros de investigación, empresas, instituciones académicas y administración realizaron una activo trabajo de desarrollo de la tecnología, lo que redundó en notables avances,

apareciendo las primeras aplicaciones de RFID. Las primeras patentes para dispositivos RFID fueron solicitadas en Estados Unidos, concretamente en Enero de 1973 cuando Mario W. Cardullo se presentó con una etiqueta RFID activa que portaba una memoria reescribible. El mismo año, Charles Walton recibió la patente para un sistema RFID pasivo que abría las puertas sin necesidad de llaves. A pesar de ello, la tecnología se siguió utilizando de modo restringido y controlado. Grandes empresas como Raytheon, RCA y Fairchild empezaron a desarrollar tecnología de sistemas de identificación electrónica, y en 1978 ya se había desarrollado un transpondedor pasivo de microondas. A finales de esta década ya se había completado una buena parte de la investigación necesaria en electromagnetismo y electrónica para RFID, y la investigación en otros de los componentes necesarios, las tecnologías de la información y las comunicaciones, estaba empezando a dar sus frutos, con la aparición del PC y de ARPANET.

En los años 80 aparecieron nuevas aplicaciones. Fue la década de la completa implementación de la tecnología RFID. Los principales intereses en Estados Unidos estuvieron orientados al transporte, al acceso de personal y, más débilmente, a la identificación de animales. En Europa sí cobró un especial interés el seguimiento de ganado con receptores de identificación por radiofrecuencia como alternativa al mercado. Más tarde también aparecieron los primeros peajes electrónicos. La primera aplicación para aduanas se realizó en 1987, en Noruega, y en 1989 en Dallas. Todos los sistemas eran propietarios, y no existía la interoperatividad.

Ya en la década de los 90 se tomó conciencia de las enormes posibilidades que podía brindar la explotación de RFID y comenzaron a aparecer los primeros estándares. En Estados Unidos se siguió profundizando en la mejora de los peajes automáticos y la gestión de autopistas. Mientras tanto en Europa se implementaron aplicaciones RFID para controles de acceso, peajes y otras aplicaciones comerciales. En 1999, un consorcio de empresas fundó el Auto-ID Center en el MIT.

Y a partir del año 2000, empezó a quedar claro que el objetivo de desarrollo de etiquetas a 0,05 dólares podría alcanzarse, con lo que la RFID podía convertirse en una tecnología candidata a sustituir a los códigos de barras existentes. El año 2003 marcó un hito en el desarrollo de la tecnología RFID: Walmart y el Departamento de Defensa(DoD) estadounidense decidieron adherirse a la tecnología RFID. Les siguieron otros fabricantes, como Target, Procter & Gamble y Gillette. En 2003 el centro AutoID se convirtió en EPC global, creadora de estándares adoptados por Walmart y el DoD.

La empresa Texas Instruments desarrolló diversas aplicaciones para el control del encendido del motor del vehículo, control de acceso de vehículos o pases de es aquí. Así mismo, numerosas empresas en Europa se introdujeron en el mercado, más aún tras detectar la potencial aplicación en la gestión de artículos.

En año 2002 empezó a despuntar la tecnología NFC (Near Field Communication),tecnología que mejora las prestaciones de RFID gracias a que incluye en un único dispositivo, un emisor y un receptor RFID, y que puede insertarse en un dispositivo móvil, aportando a éste nuevas funcionalidades para un gran número de aplicaciones.

En Europa, el proyecto lanzado en 2005 por Correos (España), Q-RFID, liderado por AIDA Centre SL, ha contribuido a incorporar las últimas tecnologías de control por radiofrecuencia para permitir la trazabilidad de la correspondencia a lo largo de todo el proceso postal. Q-RFID ha resultado uno de los más importantes proyectos de RFID de Europa, suponiendo una gran contribución al desarrollo e implantación de la tecnología. Aunque el proyecto ha finalizado en 2007, el éxito alcanzado garantiza la continuidad del mismo.

Todo hace pensar que en los próximos años la tecnología RFID va camino de convertirse en una tecnología ampliamente utilizada en multitud de sectores. El creciente interés en el comercio electrónico móvil traerá consigo más aplicaciones, gracias a la capacidad de RFID para transportar datos que pueden ser capturados electrónicamente[12].

2.3.2. Características de RFID

Los sistemas de identificación por radiofrecuencia o RFID (Radio Frequency Identification) son una nueva tecnología para la identificación de objetos a distancia sin necesidad de contacto, ni siquiera visual además de ser una buena opción en ambientes rudos. Los principales elementos necesarios en un sistema RFID son:

- Tag
- Antena (ARFID)
- Protocolo
- Software

La Tag o etiqueta consiste en un microchip que va adjunto a una antena de radio y que va a servir para identificar unívocamente al elemento portador de la etiqueta. Con esto podemos almacenar hasta 2 Kbytes de datos. La Antena RFID es el dispositivo de lectura de datos, éstos datos se encuentran almacenados en la etiqueta o Tag y son obtenidos mediante la excitación por ondas de radio, emitidas por la antena. El protocolo es el lenguaje o el método por el cual se comunicara la antena RFID con el dispositivo de procesamiento ya sea una PC, HamHelt, Teléfono Celular, etc. El Software el cual se comunicará con la antena mediante comandos para la obtención de información, su manipulación y/o almacenamiento. A continuación se muestra en la figura6la interacción de cada uno de los elementos.

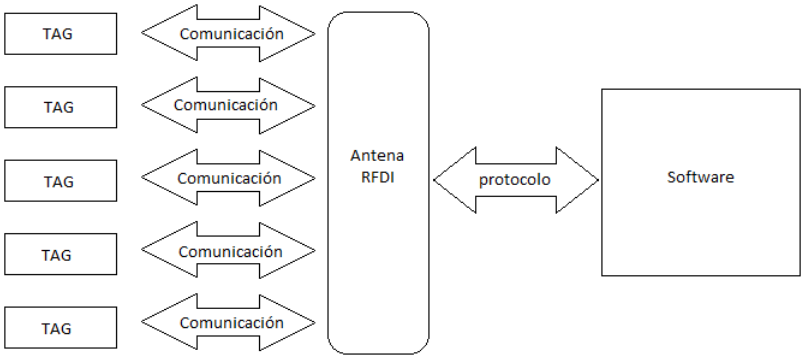


Figura 6: Procedimiento RFID

Las bandas de frecuencia en las que trabajan los sistemas RFID son 125 o 134 Khz para baja frecuencia y 1356 Khz para alta frecuencia, aunque pueden trabajar en muchos otros rangos de frecuencia. Para el uso del espectro UHF los distintos países no consiguen llegar a un estándar ya que en Europa se trabaja en 868 Mhz., en Estados Unidos y en Japón 915 Mhz, reticente al uso de esta banda, empieza a trabajar en 960 Mhz. El problema que se genera en el empleo de la banda UHF es que hay distintos dispositivos que operan sobre la misma, y generan ruidos sobre los sistemas RFID y viceversa, con lo cual los gobiernos tienen que realizar detallados estudios para determinar y minimizar los trastornos que puedan suceder como consecuencia de cambiar las bandas de trabajo de los dispositivos RFID[6].

2.4. Redes de Petri

Por último las Redes de Petri permiten obtener un mayor entendimiento del funcionamiento del protocolo, al modelar las reglas que lo componen, de igual forma se puede analizar y la verificar el algoritmo mediante su implementación.

2.4.1. Fundamentos de las Redes de Petri

Las Redes de Petri, son una herramienta de modelado matemático y gráfico, aplicables a una gran cantidad de sistemas, las cuales permiten no sólo una representación de dichos sistemas sino su análisis. Estas herramientas son útiles para la descripción y estudio de los sistemas de procesamiento de información, que son caracterizados como concurrentes, asíncronos, distribuidos, paralelos, no determinísticos y/o estocásticos.

Las Redes de Petri al ser una herramienta gráfica, pueden ser usadas como ayuda en la comunicación visual, similar a los diagramas de bloques y diagramas

de flujo. En las Redes de Petri se utilizan marcas para simular la dinámica y actividades concurrentes en el sistema.

Como una herramienta matemática, las Redes de Petri permiten obtener ecuaciones de estado, ecuaciones algebraicas y otros modelos matemáticos gobernados por el comportamiento del sistema.

Las Redes de Petri también son aplicables al modelado de sistemas discretos concurrentes y admiten una representación gráfica. Por otro lado, la simulación de sistemas modelados con Redes de Petri, se pueden llevar a cabo sin dificultad con cualquier tecnología (electrónica, fluídica). Las dos cualidades anteriores se complementan con la potencia de las herramientas de análisis sobre ellas desarrolladas.

Existen áreas de aplicación, tales como la evaluación del desempeño y protocolos de comunicación y existen otras que son áreas que requieren de un mayor análisis, por ejemplo, los sistemas de software distribuido, sistemas de bases de datos distribuidos, programas en paralelo y concurrentes, sistemas de control industrial y de manufactura flexible, sistemas a eventos discretos, sistemas de memoria multiproceso, sistemas de flujo de datos, sistemas de tolerancia a fallos, lógica programable, estructuras y circuitos asíncronos, sistemas operativos y compiladores, lenguajes formales, entre otros[7].

Una Red de Petri es un grafo orientado que permite modelar de forma adecuada los sistemas concurrentes y consta de los siguientes elementos:

- Estados: Representados por círculos.
- Transiciones: Representados por barras.
- Arcos: Que unen lugares a transiciones y viceversa.

Los arcos que unen estados a transiciones son llamados arcos de entrada y los que unen transiciones a estados se conocen como arcos de salida. Los arcos son etiquetados con valores numéricos o características de la información, así un arco con valor k, puede ser interpretado como el conjunto de k-arcos paralelos.

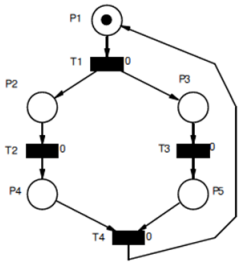


Figura 7: Ejemplo de Red de Petri

Cada estado puede contener un determinado número de pequeños puntos llamados tokens. Una distribución de tokens en los lugares constituye el marcado de una red. El marcado evoluciona según las siguientes reglas:

- Una transición está habilitada si todos sus estados de entrada están debidamente marcados.
- Una transición habilitada puede dispararse cuando se verifica el evento asociado.
- El disparo de una transición consiste en quitar marca(s) a cada lugar de entrada y añadir marca(s) a cada lugar de salida.

La representación matemática de todo lo anterior, permite desarrollar ecuaciones algebraicas y ecuaciones de estado que describen la conducta del sistema y que permiten analizarlos. A continuación se da la definición formal de las Redes de Petri plaza- transición.

Notación: $N = \{0, 1, 2, \ldots\}, R_+ = [0,\infty), N_{no}^+ = \{n_0, n_0+1, \ldots n_0+k, \ldots\}; n_0 \geq 0$
Una Red de Petri ordinaria es una 5-tupla, $PN = (P, T, F, W, M_0)$ donde:

PN =Red ordinaria.
$P = \{p_1, p_2 \ldots p_m\}$es un conjunto finito de lugares.
$T = \{t_1, t_2, \ldots t_n\}$ es un conjunto finito de transacciones.
$F \subset (P \times T) \cup (T \times P)$ es el conjunto de arcos.
$W : F \to N^+$es la función que asigna peso a los arcos.
$M_0 : P \to N$es el marcaje inicial.
$P \cap T = \Phi$y $P \cup T \neq \Phi$.

Una estructura de Red de Petri sin un marcaje inicial específico, se denota por PN. Una Red de Petri con marcaje inicial dado, se denota por (PN, M_0). Si $W(p,t) = \alpha(oW(t,p) = \beta)$, entonces esto se representa gráficamente por $\alpha, (\beta)$ arcos de $p\ a\ t\ (t, a, p).M_k(p_i)$ representa el marcaje (es decir, el número de tokens) en el lugar $p_i \in P$ en un tiempo k y $Mk = [M_k(p_i) \ldots, M_k(P_m)]^T$denota el marcaje (estado) de la Red de Petri en el tiempo k. Una transición $tj \in T$ se dice estar habilitada para un tiempo k si $M_k(p_i) \geq W(p_i, t_j)$ para toda $p_i \in P$, tal que $(p_i, t_j) \in F$. Se asume que en cada tiempo k existe al menos una transición habilitada. Si la transición está habilitada, entonces puede ser disparada. Si una transición habilitada $t_j \in T$ se dispara en un tiempo k, entonces el siguiente marcaje para $p_i \in P$ esta dado por:

$$M_{k+1}(p_i) = M_k(p_i) + W(t_j, p_i) - W(p_j, t_i)$$

Sea $A = [a_{ij}]$, una matriz de enteros $n \times m$(matriz de incidencia), donde $a_{ij} = a_{ij}^+ - a_{ij}^+$ con $a_{ij}^+ = W(t_i, p_j)$ y $a_{ij}- = W(p_i, t_j)$ y sea $u_k \in \{0,1\}^n$ el vector de disparo, donde si $t_j \in T$se dispara, entonces su correspondiente vector de disparo sera $U_k = [0, \ldots, 0, 1, 1, \ldots, 0]^T$, con un uno en la j − ésima posición

y ceros en los demás lugares. La ecuación matricial (ecuación de diferencias no lineales), que describe el comportamiento dinámico en una Red de Petri es:

$$M_{k+1} = M_k + A^T u_k$$

Donde si en el estado k, a $ij- < Mk\,(pj)$ para todo $p_j \in P$, entonces $t_j \in T$, está habilitada y si dicha transición se dispara, entonces su correspondiente vector de disparo u_k se utilizará en la ecuación (2), para generar el siguiente estado, dicho vector es utilizado en la ecuación de diferencia para generar el siguiente paso. Notar que si M' puede ser alcanzado desde algún otro marcaje M, y si se dispara alguna secuencia de d transiciones con los correspondientes vectores de disparo $u_0, u_1, \ldots, u_{d-1}$ se obtiene que[4].

$$M' = M + A^T u, u = \sum_{k=0}^{d-1} u_k$$

2.4.2. Configuraciones y Estructuras Básicas de las Redes de Petri

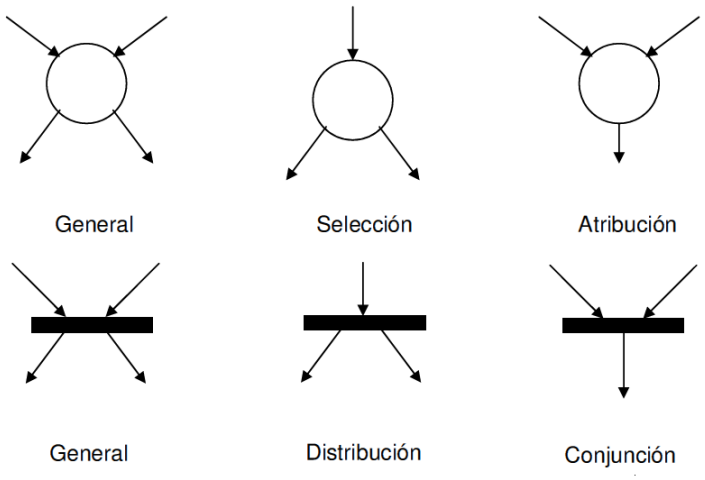


Figura 8: Configuración permitidas para Redes de Petri

Enseguida se presentan algunas definiciones y diagramas del conjunto de estructuras básicas utilizadas en las Redes de Petri (figura 9).

Secuencia
La secuencia es observable debido a que todas las cosas suceden en orden (figura 9.a).

Conflicto

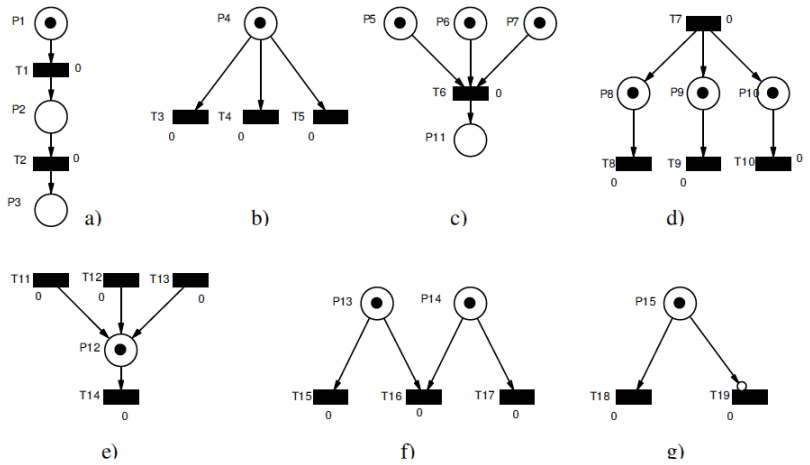


Figura 9: Estructuras de las Redes de Petri

El token en p4 activa 3 transiciones, pero cuando una de éstas se dispara el token es removido, por lo tanto los dos restantes continúan inactivos (figura 9.b).

Sincronización
La sincronización es muy bien modelada por las Redes de petri; cuando los procesos cargados en p5, p6 y p7 son finalizados, los tres son sincronizados para inicializar p11 (figura 9.c).

Concurrencia
Muchos sistemas operan con actividades concurrentes y éste es el modelo usado; después de dispararse la transición T7 aparecen los marcados en P8, P9 y P10 (figura 9.d).

Conjunción
Esto no es lo mismo que la sincronización, no requiere que las tres transiciones se disparen al mismo tiempo o que las tres se disparen antes de T14, ésta simplemente conjuga tres procesos paralelos (figura 9.e).

Confusión
Es una combinación de conflicto y concurrencia. P13 activa T15 y T16, pero si T16 se activa, T15 no puede ser activada produciéndose la confusión (figura 9.f).

Prioridad/inhibición
Un tipo especial de arco, el arco inhibidor, es usado para invertir la lógica de un lugar de entrada. Con un arco inhibidor, la ausencia de un token en el lugar de entrada activa la transición de salida. Usando este tipo de arco para controlar T19; como P16 tiene el token no puede dispararse T19 (figura 9.g)[4].

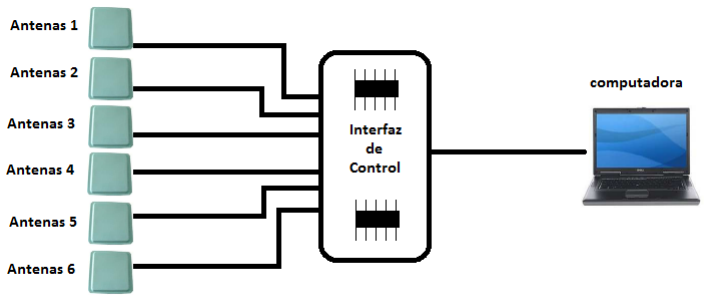
2.4.3. Propiedades Básicas de las Redes de Petri

Vivacidad
Una transición es viva para un marcado M^0 si para todo marcado M alcanzable desde M^0 existe un marcado M' sucesor de M en el que la transición está habilitada. Una red es viva para un marcado M^0 si todas sus transiciones son vivas para ese marcado.

Limitación
Un lugar es limitado para un marcado M^0 si en cualquier marcado alcanzable desde M^0 el número de marcas del lugar está acotado. Un lugar es binario para M^0 si en cualquier marcado alcanzable desde M^0 el lugar contiene como máximo una marca. Una red es limitada (binaria) para M^0 si todos sus lugares son limitados (binarios) para M^0 .

3. DESARROLLO

Inicialmente se realizó el análisis de la problemática a resolver con lo que se llego, al siguiente diagrama de bloques el cual muestra la conexión entre la Pc, la Ic y las ARFID.



Cuadro 1: Diagrama de bloques de la interfaz de control

Como se muestra en el capítulo 2 en la sección "características de RFID" las antenas se conectan a la PC directamente, la antena lee Tag y después de analizarlas envía el identificador a la PC, en este caso la interfaz sustituirá a la PC como estación de control y únicamente la PC recibirá la trama de los identificadores de todas las Tag. La primera actividad es el análisis de la antena.

3.1. Análisis de Gen 2 RFID Reader

Para esta actividad se uso snnifer(Serial Port Monitor by Eltima Software que pudiera monitorear la antena mientras se comunicaba con la PC, esto con la finalidad de obtener las tramas de información que se usan para cada uno de los procesos que realiza la antena.

Los resultados fueron los siguientes:

3.1.1. Método de Comunicación.

Mediante el uso de software "Serial Port Monitor" se realizó el monitoreo de las tramas que envía la antena a la PC durante el proceso de conexión, configuración, lectura y desconexión además de ver la estructura e identificar bytes claves en la trama.

A continuación se muestra el comportamiento de la antena.

Debido a que la antena que se usa es programable, lectura y escritura, sin olvidar que maneja 5 protocolos y multi-lectura; esto da como resultado una

gran cantidad de comandos. Por esta razón se decidió hacer una lista corta con los comandos que se emplearán.

Proceso	Descripción	Trama enviada a la ARFID	Trama recividad de la ARFID
Conectar	Abre el puerto y	a5 01 02 60 f9	e9 00 03 60 00 b4
	prepara la antena	a5 01 03 74 00 e4	e9 00 03 74 00 a0
	para leer y escribir	a5 01 02 7a df	e5 00 04 7a 00 07 96
Desconectar	La PC termina la comunicación con la antena.	a5 02 02 75 e4	e9 00 03 75 00 9f
Lectura	La PC lee la tag	a5 02 03 92 04 c2	e9 00 03 92 02 80
	mas cercano	a5 01 03 92 04 c2	e5 00 0f 92 04 XX .. .. XX TT
Lectura	La PC lee las tag	a5 01 03 c2 04 92	e9 00 03 c2 05 4d
		a5 01 03 c2 04 92	e5 00 03 c2 02 54
Múltiple	mas cercanas	a5 01 04 61 01 01 f4	e5 00 13 61 01 01 00 04 01 X..XYY
		a5 01 04 61 01 02 f3	e5 00 21 61 01 04 00 04 01 X..XYY 0401 X..XYY TT

Cuadro 2: Lista de comandos Gen 2 RFID Reader

Como se muestra en la tabla anterior la antena RFID tiene 4 procesos de funcionamiento conectar, desconectar, lectura, lectura múltiple; cada uno de estos procesos tiene una o más tramas de envío y trasmisión, claramente se aprecia que la antena trabaja bajo es un protocolo maestro-esclavo y debido a que todas las antenas se conectan a una única PC, esta conexión es una topología estrella.

3.1.2. Descripción de las Tramas

Cada uno de los bytes que contiene la trama tiene un significado pero únicamente los tres primeros bytes de la trama son fijos, por lo que los demás bytes tiene un significado diferente según en el proceso en que se encuentre la antena, a continuación se muestra la codificación de la trama figura10.

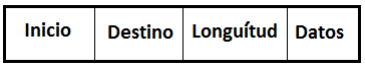


Figura 10: Trama

La descripción del vocabulario del protocolo se muestra a continuación:
Inicio:
El byte 1 pone en alerta a la antena avisándole que se realizará el envío de una trama (A5) (E9).
Destino :
El byte 2 define el nombre o número de la antena a la que va dirigida la trama (00,01 ..... FF).
Longitud :
El byte 3 define el número de bytes que tendrá datos (02, 03 ..... FF).
Datos :

Estos son un conjunto de bytes que de acuerdo al proceso en que se encuentra la antena cambiará su longitud y su secuencia.
En el proceso de conexión se envían tres tramas y "Datos" que tiene la función de configurar.

- En la primera trama se envía la petición del estado de la antena si está disponible o si tiene una avería en los bytes 4 y 5 (F9).
- En la segunda trama se envía la configuración de la velocidad de trasmisión y recepción con la que trabajará con la PC en los byte 5 y 6 (00 E4).
- En la tercera trama solicita la versión del Firmware de la antena y cambia el estado de la antena a esperando petición en los bytes 4 y 5 (7A DF).

De igual manera la antena responde con tres tramas.

- En la primera trama la antena devuelve el estado de la antena en los bytes 5 y 6 (00 B4)
- En la segunda trama la antena confirma la velocidad en los bytes 5 y 6(00 A0).
- En la tercera trama la antena devuelve el firmware en los bytes 6, 7 y 8 (00 07 96).

En el proceso de lectura se envían dos trama y "Datos" tiene la función de petición y configuración de la petición.

- El byte 4 de la trama define el tipo de lectura sencilla o múltiple (92).
- El byte 5 define el tipo de Tag que leerá EPC o ISO 18000 (04).
- El byte 6 y último define (C2).

La respuesta por parte de la antena puede ser una trama de dos, dependiendo de si hay o no hay Tag. Las tramas posibles son las siguiente:

- La primera trama únicamente es enviada por la antena si no existe ninguna Tag dentro del radio de lectura esto se refleja en el byte 5 y 6 (02 80).
- La segunda trama contiene el byte (92) el cual define el tipo de lectura y el (04) el cual confirma el tipo de Tag además de esos dos bytes le siguen 12 bytes (XX) los cuales son el código RFID y por ultimo un byte (TT) que define el fin de la transmisión.

En el proceso de lectura múltiple se envían dos tramas

- La primera para comprobar si hay Tag o no, donde el byte 4 (c2) define el tipo de lectura y el byte 5 (04) define el tipo de Tag a leer.
- La segunda para pedir los identificadores donde el byte 6 (01) define el numero de Tag que debe leer la antena.

En el proceso de lectura múltiple de igual manera que la lectura simple, la trama que envía la antena dependerá de si hay o no una Tag.

- La primera trama únicamente devuelve que no existe una Tag dentro del radio de lectura que se ven reflejado en los bytes 5 y 6 (05 4D).
- La segunda trama devuelve si hay Tag en el radio de lectura y además del numero de Tag dentro del radio, en el byte 5 (02)
- Una vez que la PC recibe la trama anterior la PC trasmite la segunda trama y como respuesta la antena envía su siguiente trama en la que el byte 6 (02) es el numero de Tag en el radio; dentro de la misma trama el byte 8 (04) define el tipo de Tag, el byte 9 (01) define el inicio de transmisión del identificador, seguido de 12 bytes que contiene el identificador (XX), seguido un byte de terminación de identificador (YY) y así sucesivamente hasta que el número de identificadores definidos por el byte (02) sean trasmitidos y por ultimo envía un byte de terminación de trama (TT).

3.2. Especificaciones del Diseño

La segunda actividad a realizar es definir las condiciones iniciales dentro de las que se llevará acabo el diseño del protocolo.

3.2.1. El Servicio Provisto por el Protocolo

- Tipo de trasmisión: half-dúplex
- Modo de comunicación; asíncrona.
- Interfaz para la comunicación con la PC y las antenas; RS-232
- Velocidad de transmisión; inicial de 9600 baudios y modificable.

3.2.2. Suposiciones Acerca del Medio Ambiente

- Existe un sólo medio de comunicación.
- La interfaz funciona como maestro con respecto a las antenas y envia tramas de indentificadores a la PC
- El protocolo está en el nivel físico y de enlace por lo que el manejo de las capas superiores dependen del software en la PC.
- Todas las antenas son del mismo modelo especificado (GEN 2 RFID READER).

3.2.3. Vocabulario del Protocolo

- Inicio: Este byte define el inicio de trasmisión de una trama además de definir el tipo de información que contiene la misma.
- Destino: Este byte define ARFID de destino.
- Longitud: Este byte define el número de bytes que se trasmitirá o que contendrá las tramas.
- Num-tag: Es la cantidad de tag que fueron identificadas por las ARFIDs.
- Datos: Este es un conjunto byte que contiene los identificadores.

3.2.4. Codificación de Cada Mensaje del Vocabulario

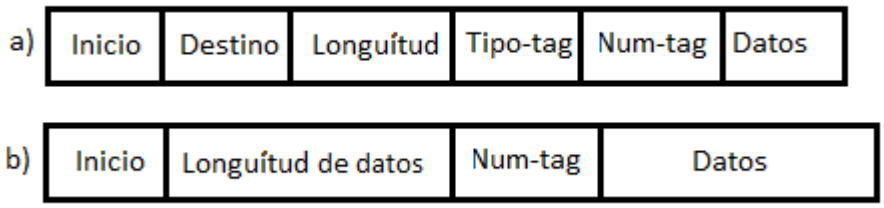


Figura 11: Formato de trama

Debido a que la IC se comunicará con la PC y las diferentes ARFID, se definieron las codificaciones que se muestran en la figura11.

- La primera (a) definición es para la trasmisión de datos entre la IC a la ARFID.
- La segunda (b) definición es para la trasmisión de datos entre de la IC a la PC.

Con lo anterior se definieron tres tipos de tramas

- trama de envio: esta es la trama enviada por la IC a la ARFID.
- trama recepción: esta es la trama que recibe la IC de la ARFID.
- trama de indentificadores: esta es la trama enviada por la IC a la PC.
- trama de inicio: esta es la trama enviada por laPC a la IC.

3.2.5. Reglas de Intercambio de Mensajes

Como se planteó anteriormente la interfaz de control sustituirá a la PC, esta interfaz será programada lenguaje de alto y bajo nivel y tendrá la responsabilidad del control de las 7 ARFID. Además la interfaz también tendrá la función de retransmitir la información obtenida por las antenas hacia la PC.

Dispositivo Interfaz de Control a PC Consideraciones del Protocolo. El dispositivo RFID inicial se define como $ARFIDn$ y el número máximo de dispositivos como $ARFIDmax$.
El tiempo de espera está definido por t.
Reglas:

1. Si "PC está lista" entonces pone trama de inicio en el medio con dirección al IC; y espera respuesta.
2. Si "IC está listo", entonces pone trama de envio en el medio con direccion al "ARFID n", y espera respuesta durante un tiempo máximo (tmax).
3. Si tiempo (tmax) de espera del IC agotado, entonces direccionar al siguiente ARFID (ARFID n+uno), de lo contrario recibe la trama de recepción y direcciona al siguiente ARFID(n+uno).
4. Si IC ha direccionado al ultimo ARFID (ARFID max), entonces pone la trama identificadores en el medio con direccion a la PC y volver al paso uno.

3.3. Modelado del Protocolo con las Redes de Petri

Enseguida se muestra el conjunto de plazas y el conjunto de transiciones que representan a la Red de Petri del protocolo diseñado a partir de las reglas de intercambio de mensajes.
Plazas

- P1: PC está lista.
- P2: Envía trama de inicio
- P3: Inicia lectura PC
- P4: Verifica puerto
- P5: Recibe trama
- P6: IC listo
- P7: IC inicia proceso
- P8: IC envía trama.

- P9: IC espera
- P10: IC recibe trama
- P11: Tiempo agotado
- P12: Verifica ARFIDn
- P13: IC envía trama a la PC
- P14: ARFID iniciado
- P15 ARFID recibe trama
- P16: ARFID envía trama a IC
- P17: ARFID detenido
- P18: ARFID envía trama

Transiciones

- T1: Enviando trama de inicio a IC
- T2: Recibiendo trama
- T3: Leyendo
- T4: Reintentando
- T5: Procesando trama
- t6: Inicando proceso IC
- T7: Enviando
- T8: Incrementando t
- T9: Verificando t=tmax
- T10: Recibiendo trama
- T11: Incrementando ARFIDn
- T12: Incrementando ARFIDn
- T13: Redireccionando ARFID
- T14: Enviando trama a PC y redireccionando ARFID
- T15: Recibiendo trama
- T16: Procesando trama
- T17 Bloqueado

- T19: Enviando trama

A continuación se muestra las diferentes plazas, transiciones en la Red de Petri

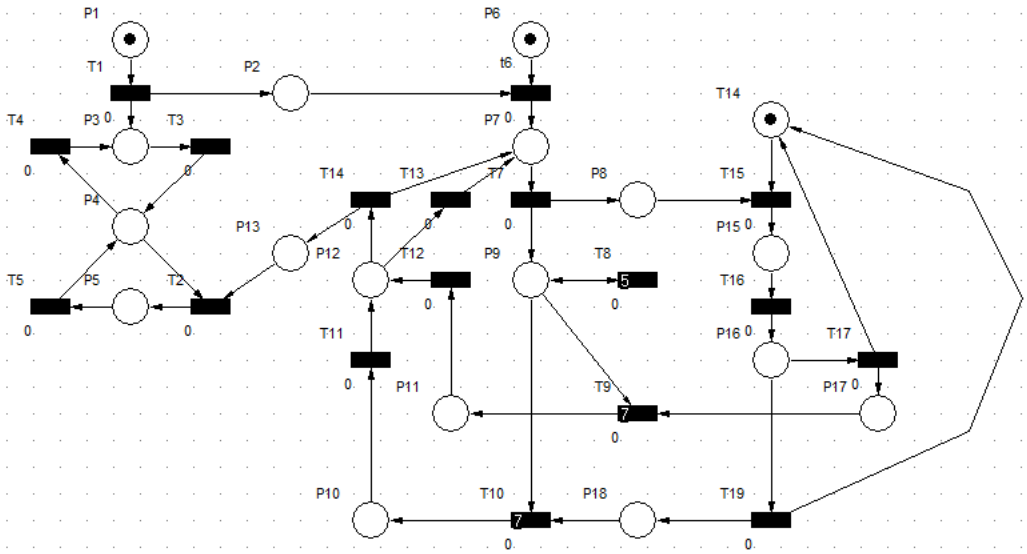


Figura 12: Modelado del protocolo con Redes de Petri

3.4. Verificación de las Propiedades del Modelo de la Red de Petri

Vivacidad La vivacidad del modelo se llevaron acabo mediante su implementación el software Visual Object Net ++ el cual permite la detección de plazas muertas.

En las siguientes gráficas se muestran las secuencias de procesos en concurrencia a través del simulador de las Redes de Petri.

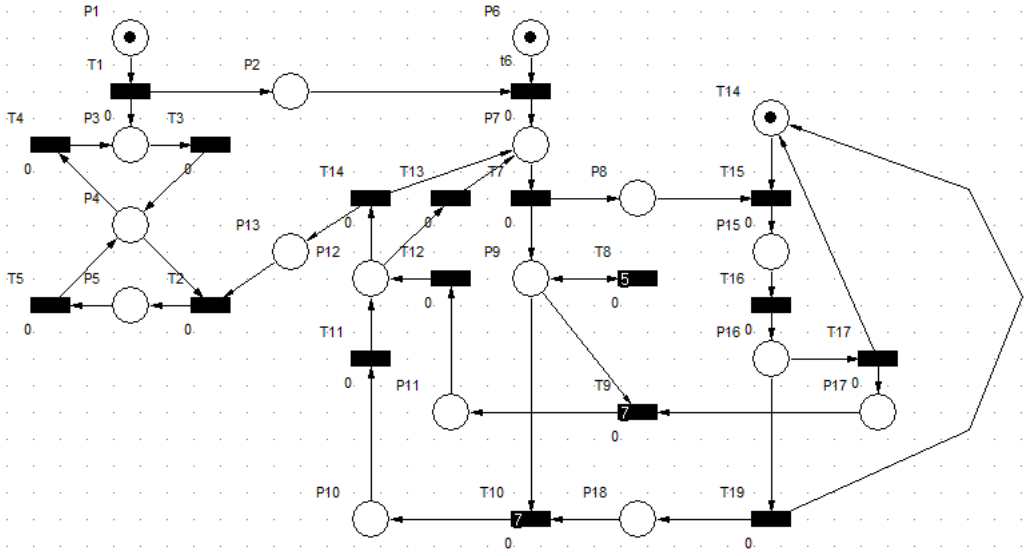


Figura 13: Modelado del protocolo con Redes de Petri (estado 1)

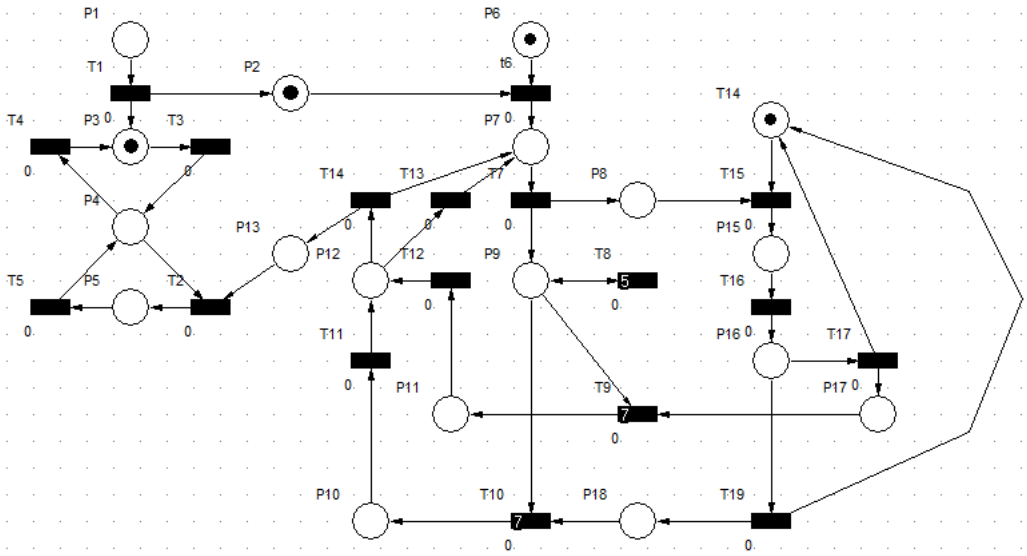


Figura 14: Modelado del protocolo con Redes de Petri (estado 2)

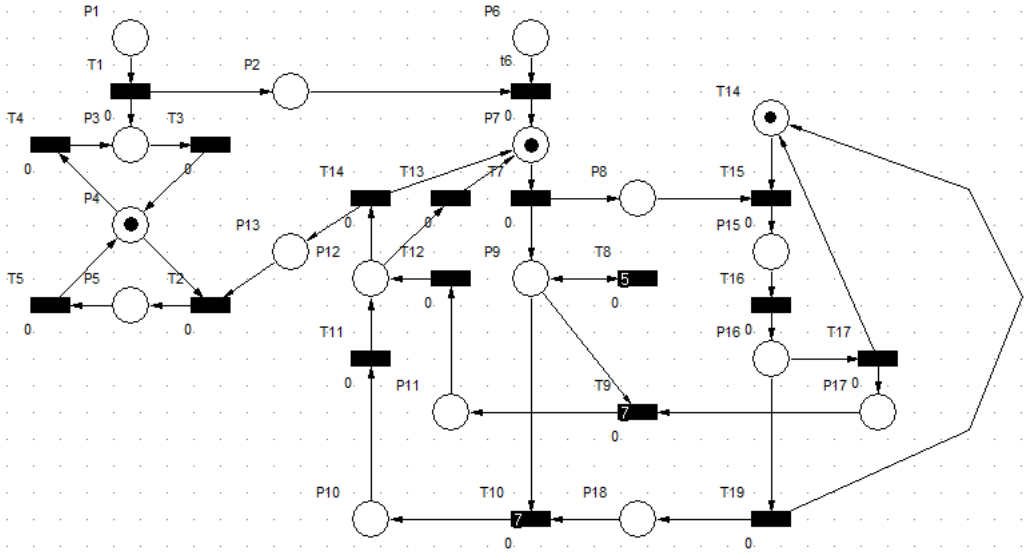


Figura 15: Modelado del protocolo con Redes de Petri (estado 3)

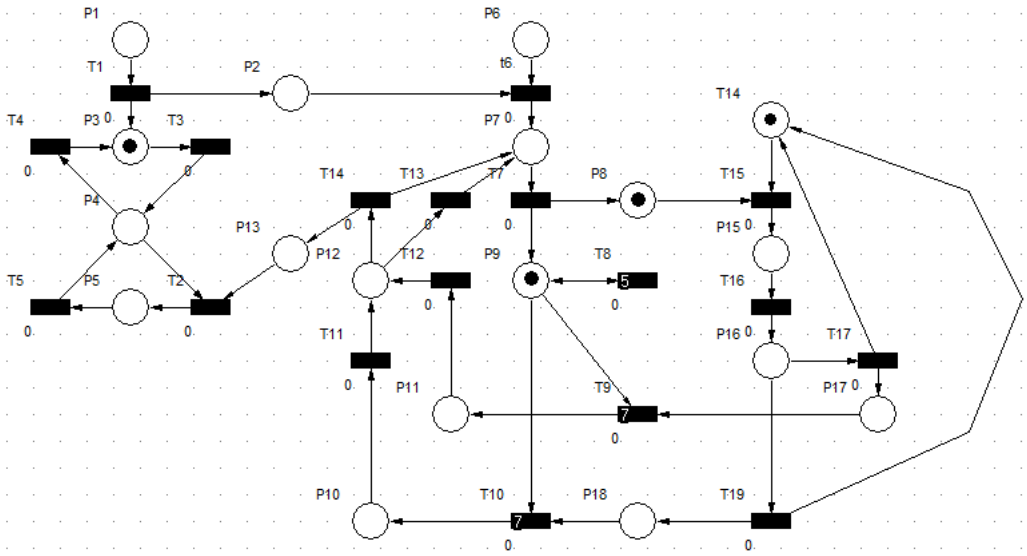


Figura 16: Modelado del protocolo con Redes de Petri (estado 4)

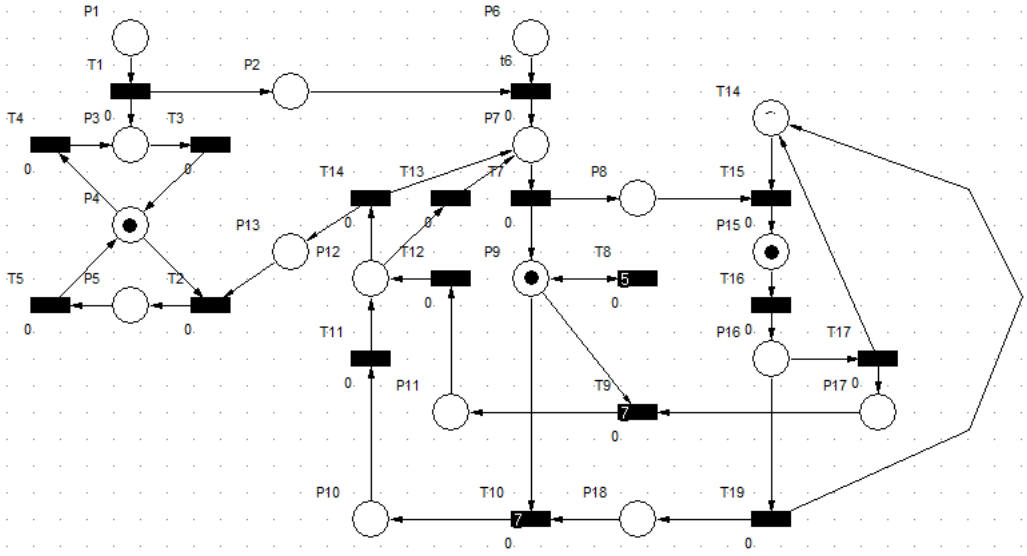


Figura 17: Modelado del protocolo con Redes de Petri (estado 5)

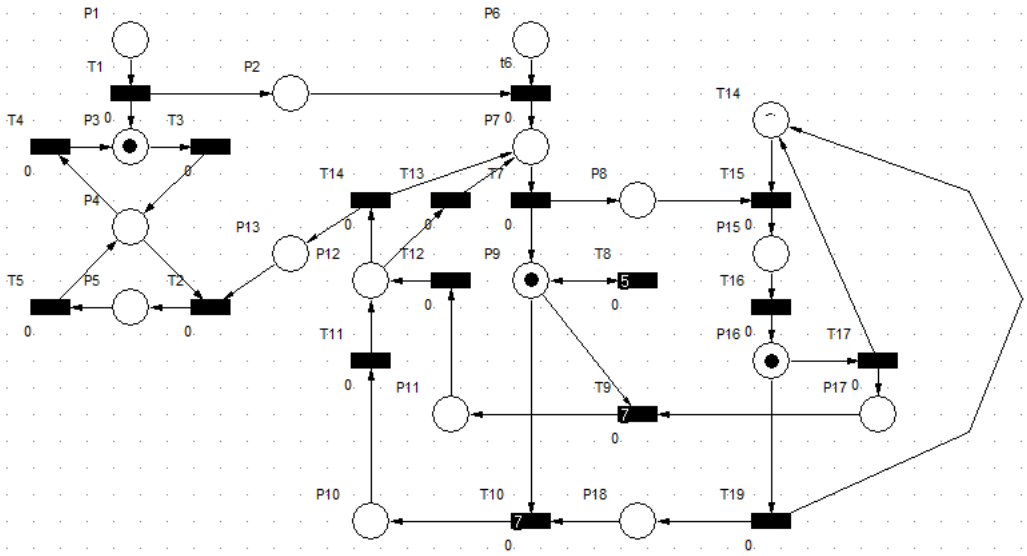


Figura 18: Modelado del protocolo con Redes de Petri (estado 6)

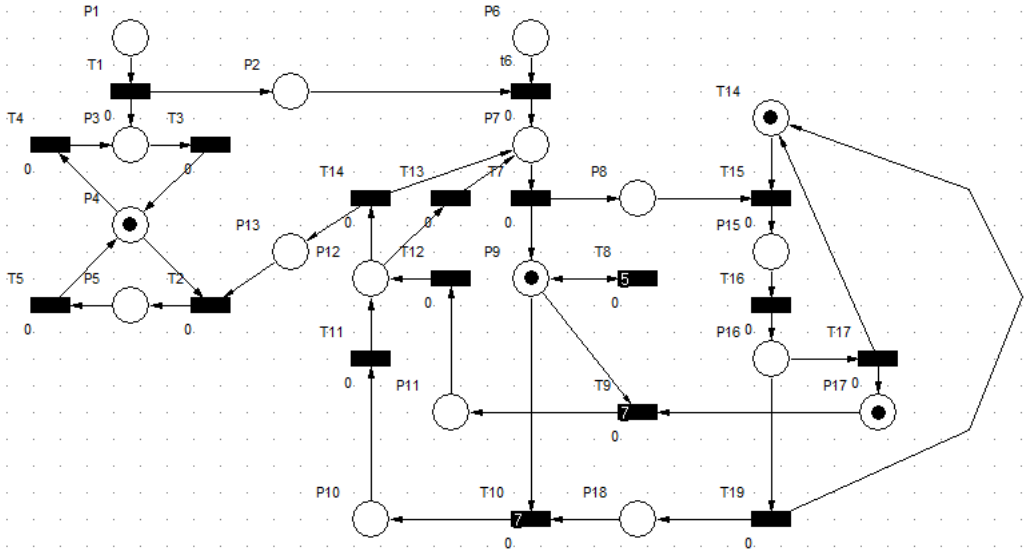


Figura 19: Modelado del protocolo con Redes de Petri (estado 7)

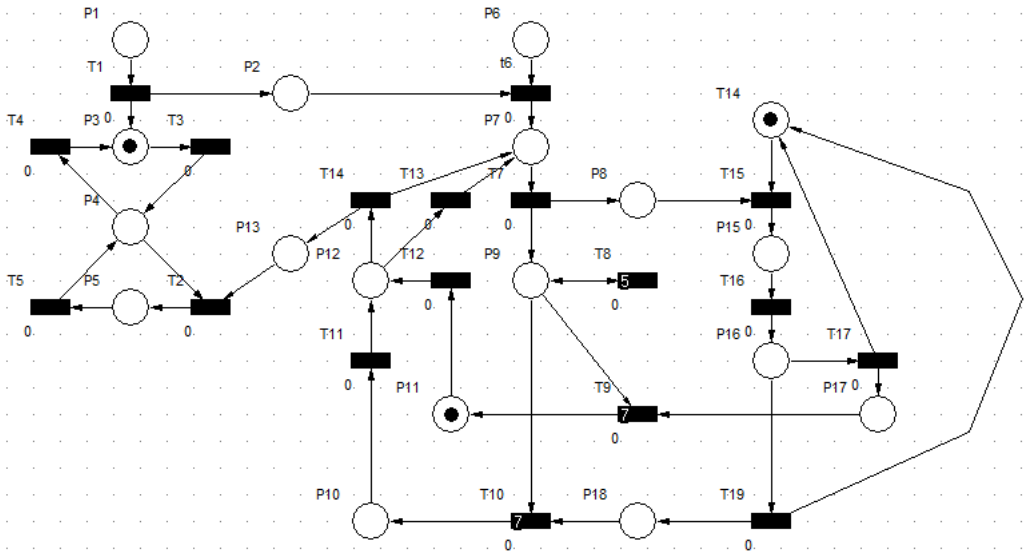


Figura 20: Modelado del protocolo con Redes de Petri (estado 8)

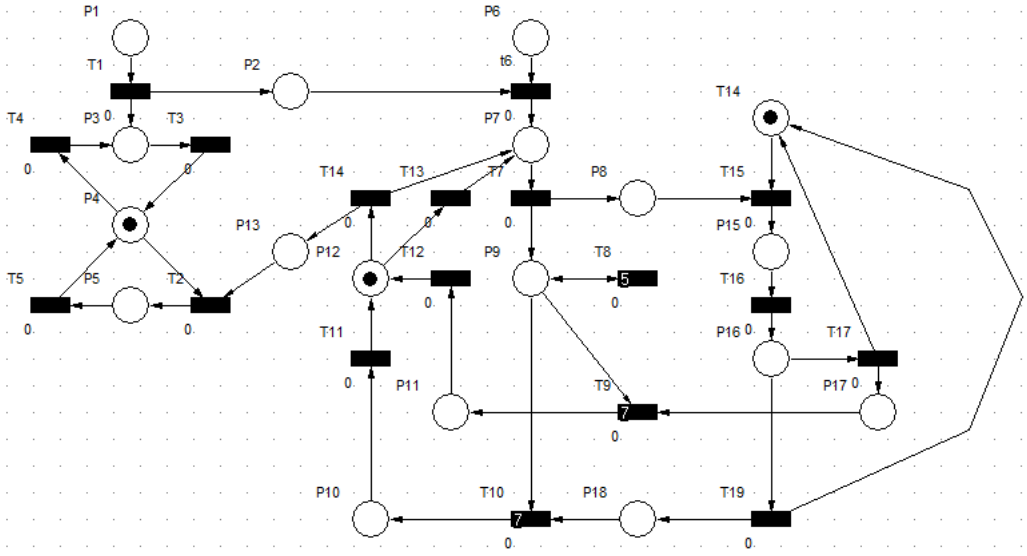


Figura 21: Modelado del protocolo con Redes de Petri (estado 9)

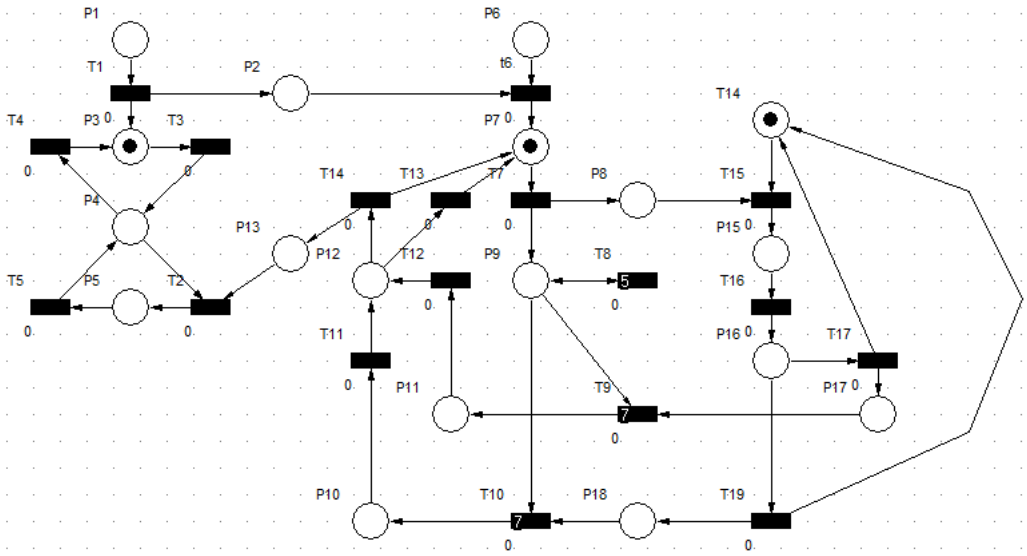


Figura 22: Modelado del protocolo con Redes de Petri (estado 10)

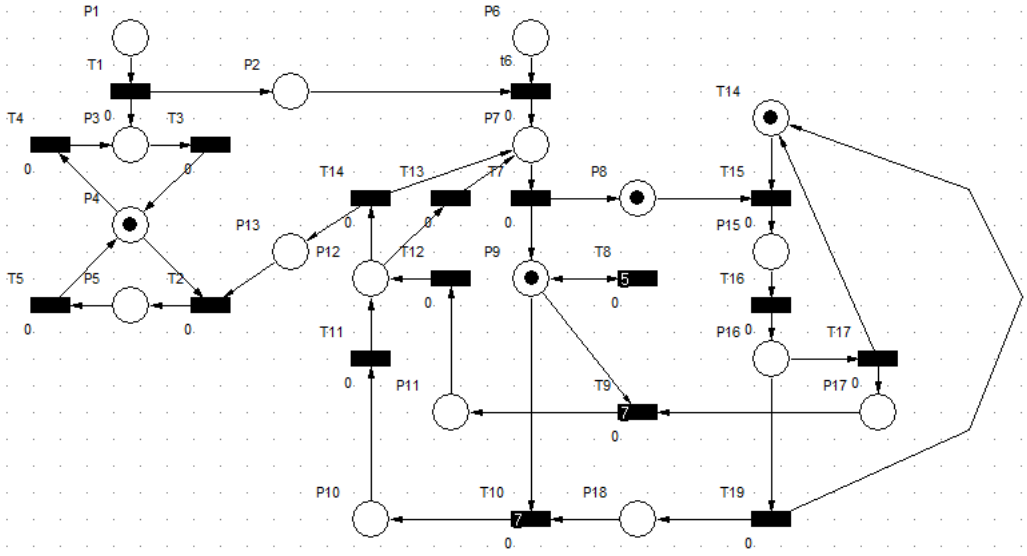


Figura 23: Modelado del protocolo con Redes de Petri (estado 11)

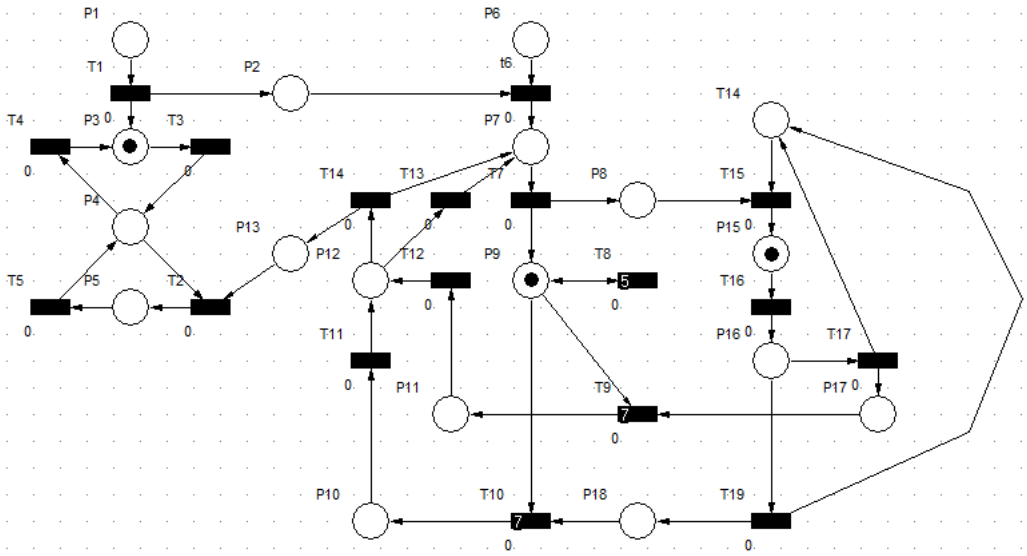


Figura 24: Modelado del protocolo con Redes de Petri (estado12)

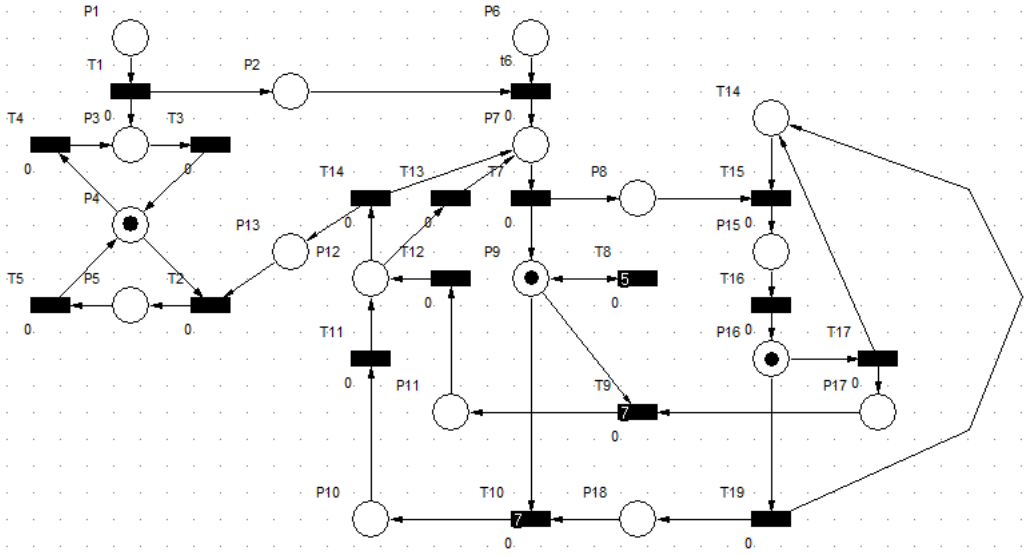


Figura 25: Modelado del protocolo con Redes de Petri (estado 13)

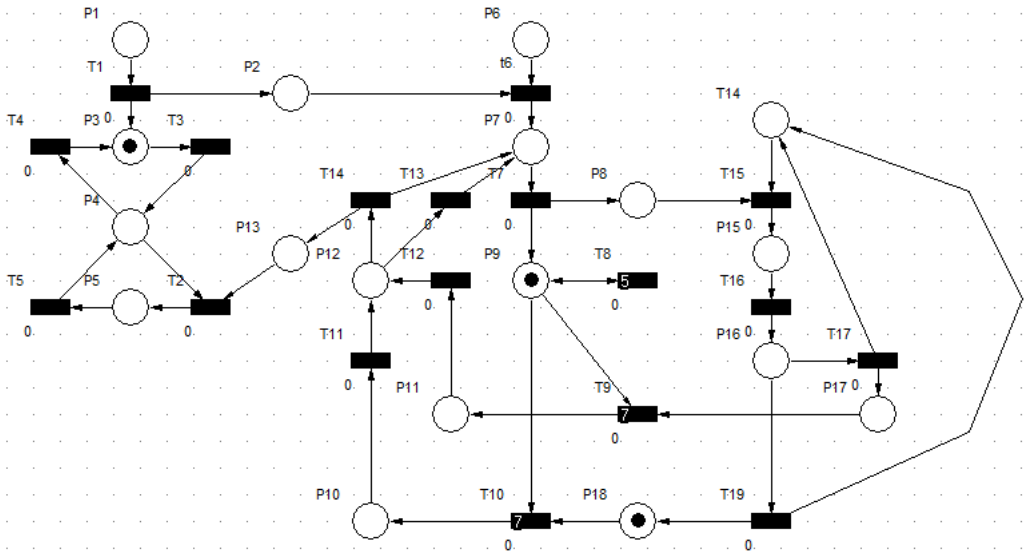


Figura 26: Modelado del protocolo con Redes de Petri (estado 14)

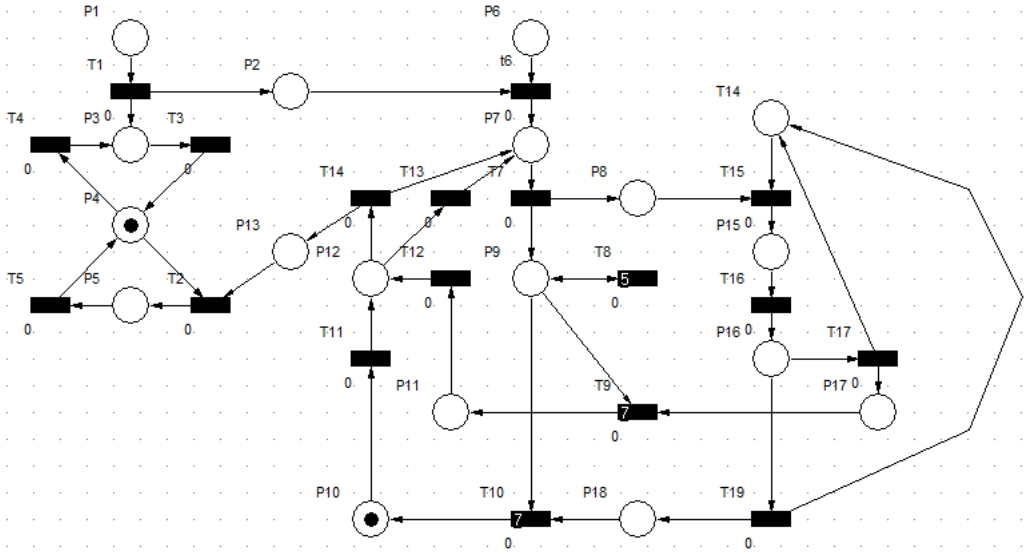


Figura 27: Modelado del protocolo con Redes de Petri (estado 15)

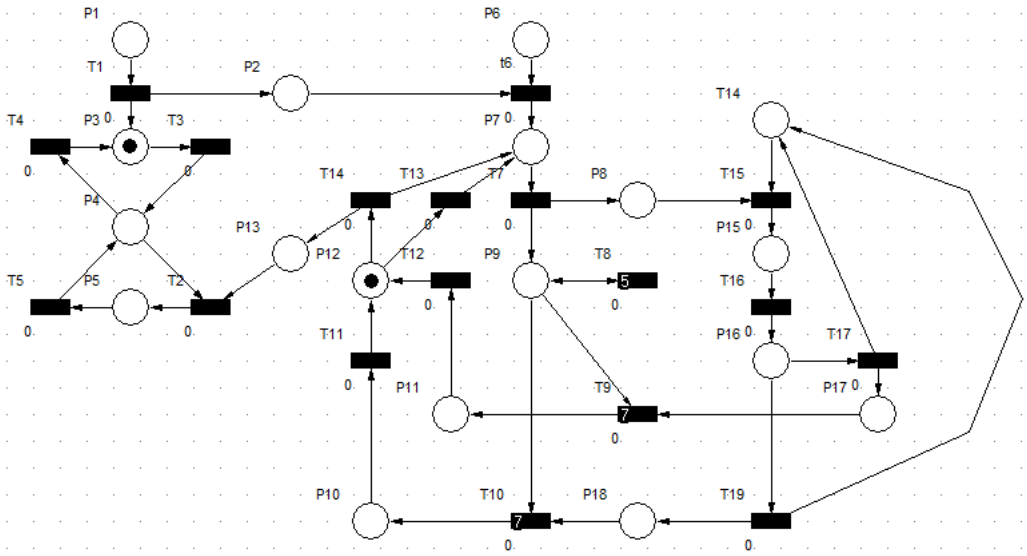


Figura 28: Modelado del protocolo con Redes de Petri (estado 16)

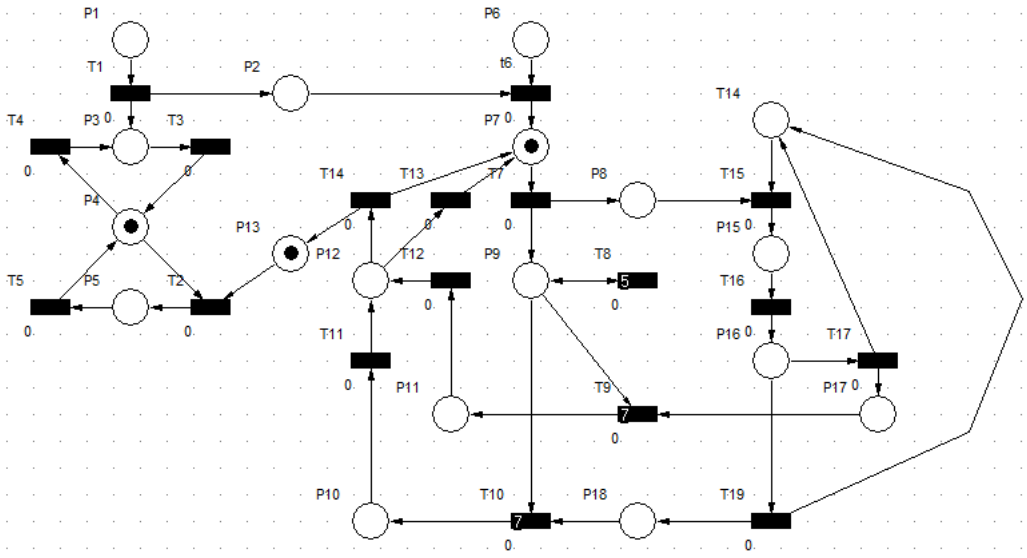


Figura 29: Modelado del protocolo con Redes de Petri (estado 17)

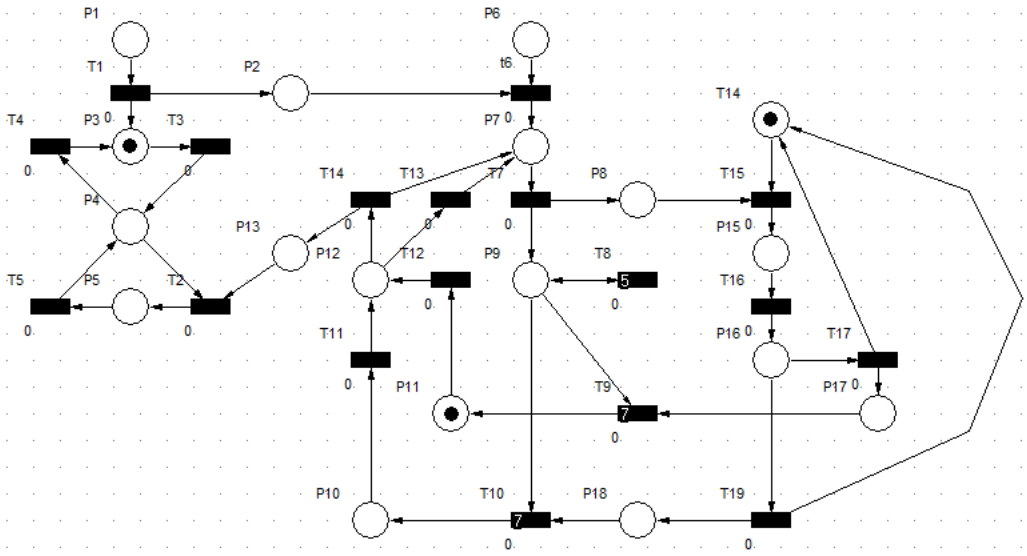


Figura 30: Modelado del protocolo con Redes de Petri (estado 18)

Con las figuras anteriores podemos observar los diferentes estados y transiciones por los que pasa la IC además de que se comprobó la vivacidad del protocolo usando el simulador Visual object net++, obteniendo buenos resultados.

3.5. Algoritmo e Implementación

3.5.1. Algoritmo de Envío IC a RFID

1. Se define el inicio de la trama.
2. Se transmite la trama byte por byte desde el inicio hasta llegar al fin de la trama
3. Se recibe la respuesta de la ARFID y se compara byte por byte para verificar si es correcta.
4. Se espera byte y aumenta la variable t.
5. Se compara $t = tmax$ si es si termina transmisión.
6. Se procede a la siguiente trama.
7. De lo contrario procesa datos.

Algoritmo 1 Algoritmo de comunicación IC RFID

```
Begin
while inicio trama < fin trama loop
trasmite trama[byte n]
n:=n+1;
endloop
while inicio trama < fin trama loop
recive byte
if recive==trama2[byte n] them
guarda dato
else
t:=t+1;
if t=tmax them
fin trasmision
endloop
cambio de antena
procesa datos
End
```

```
while (inicio_trama2<fin_trama2){

if(0xe5==recep0()){

   recep0();
   recep0();
   recep0();
   recep0();

   tag[0]=recep0();
   tag[1]=recep0();
   tag[2]=recep0();
   tag[3]=recep0();
   tag[4]=recep0();
   tag[5]=recep0();
   tag[6]=recep0();
   tag[7]=recep0();
   tag[8]=recep0();
   tag[9]=recep0();
   tag[10]=recep0();
   tag[11]=recep0();
   recep0();
}

if(t== tmax){
   t=0;
   return antena++;
}
else{
   t++;
   }

return antena;
}
```

Figura 31: Codificación IC leer

3.5.2. Algoritmo de Envío IC a PC

1. Se define el inicio de la trama.
2. Se transmite la trama byte por byte de inicio que está compuesta por 3 bytes.
3. Se transmite la trama byte por byte los identificadores.

Algoritmo 2 Codificación enviar trama a PC

Begin
trasmite trama[byte inicio]
trasmite trama[byte logitud]
trasmite trama[byte ntag]
contt<longitud loop
trasmite trama[byte n]
endloop
End

```
int t=0;

   trans7(0xA5);
   trans7(longitud);
   trans7(ntag);

for(contt=0;contt<84;contt++){

   trans7(tag[contt]);


   }
```

Figura 32: Algoritmo comunicación PC

3.5.3. Algoritmo Configuración ARFID

1. Carga tramas de configuración en memoria.
2. Envía trama uno.
3. Compara byte por byte.
4. Si un byte no es igual aumenta número de antena.
5. Envía trama de estado del RFID.

Algoritmo 3 Algoritmo configuració ARFID

```
Begin
tramas[n];
tramas2[n];
while inicio trama < fin trama loop
trasmite trama[byte n]
n:=n+1;
endloop
while inicio trama < fin trama loop
recive byte
if recive!=trama2[byte n] them
cambio de antena
endloop
while inicio trama < fin trama loop
trasmite estado[byte n]
endloop
End
```

```
int con0(){


while (inicio_trama<fin_trama){

trans0(TRAMA1[inicion_trama]);
inicio_trama++;

}

while (inicio_trama<fin_trama){

if(TRAMA2[contt]!=recep0()){

   while (inicio_trama<fin_trama){
   trans0(TRAMA1[inicion_trama]);
   }
return antena++;
}

inicio_trama++;

while (inicio_trama<fin_trama){

trans0(TRAMAESTADO[inicion_trama]);
inicio_trama++;

   }

}
}
```

Figura 33: Codificaciónconfiguració ARFID

3.5.4. Algoritmo de Transmisión de Bytes

1. Establece el puerto de transmisión.
2. Define la velocidad de transmisión.
3. Envía byte.
4. Espera un retardo.

Algoritmo 4 Algoritmo de transferencia de byte

Begin
Port_b:=0
velocidad:= 9600;
transfiere byte;
End

```
void trans0(char TRANS){

char CONT,REGX2;


#ASM
      BSF      STATUS,5
      BCF      port_b,0
      BCF      STATUS,5

      BCF      port_b,0 //<--
      MOVLW    8
      MOVWF    CONT   //;0D =CONTA1
      GOTC     AA
AA:
      NOP
      BSF      CONT,7
      GOTC     TT
ZZ:
      BCF      CONT,7
OTROBIT:
      RRF      TRANS,F      //;12H =TRANS
      BTFSC    STATUS,0
      BSF      port_b,0  //<--
      BTFSS    STATUS,0
      BCF      port_b,0 //<--
      BSF      CONT,6
      GOTC     TT
RR:
      BCF      CONT,6
      DECFSZ   CONT,F
      GOTC     OTROBIT
      GOTC     AQUI
AQUI:
      NOP
      BSF      port_b,0 //<-- 0
TT:                  //;MOVLW   0X50
      MOVF     VEL_BPS,W
      MOVWF    REGX2
WTR:
      DECFSZ   REGX2,F
      GOTC     WTR
      GOTC     AQUI2
AQUI2:
      BTFSC    CONT,7
      GOTC     ZZ
      BTFSC    CONT,6
      GOTC     RR
      //RETURN
#ENDASM

}
```

Figura 34: Codificación de trasmisión de bytes

3.5.5. Algoritmo de Recepción de Bytes

1. Establece el puerto de transmisión.
2. Define la velocidad de transmisión.
3. Recibe byte.
4. Espera un retardo.

Algoritmo 5 Algoritmo de recepción de byte

Begin
Port_d:=0
velocidad:= 9600;
recibe byte;
End

```
char recep0(){

char RECEP,CONTA_REC;

#ASM

      BSF      STATUS,5
      BSF      port_d,0
      BCF      STATUS,5

et5:
      BTFSC    port_d,0
      GOTO     et5
      MOVLW    8
      MOVWF    CONTA_REC
      CLRF     RECEP
      BSF      CONTA_REC,7
      GOTO     et6
et3:
      BCF      CONTA_REC,7
      GOTO     et6
et1:
      BCF      STATUS,0
      BTFSC    port_d,0
      BSF      STATUS,0
      RRF      RECEP,F
      BSF      CONTA_REC,6
      GOTO     et6
et2:
      BCF      CONTA_REC,6
      DECFSZ   CONTA_REC,F
      GOTO     et1
      MOVF     RECEP,W
      MOVWF    TEMP
      goto     finx

et6:
      MOVF     VEL_BPS,W //MOVLW  0X25;0XA7 9600BPS
      BTFSC    CONTA_REC,7
      MOVF     VEL_BPS2,W //MOVLW  0X0D   ;2DH 9600BPS
      MOVWF    TEMP
et4:
      DECFSZ   TEMP,F
      GOTO     et4
      NOP
      BTFSC    CONTA_REC,7
      GOTO     et3
      BTFSC    CONTA_REC,6
      GOTO     et2
      GOTO     et1
finx:
      NOP
  #ENDASM

return RECEP;
}
```

Figura 35: Codificación de recepción de bytes

4. Resultados

En este apartado se muestran los resultados obtenidos de tiempos y tramas de comunicación entre IC y la ARFID, dentro de las pruebas se considera el siguiente diagrama de bloques.

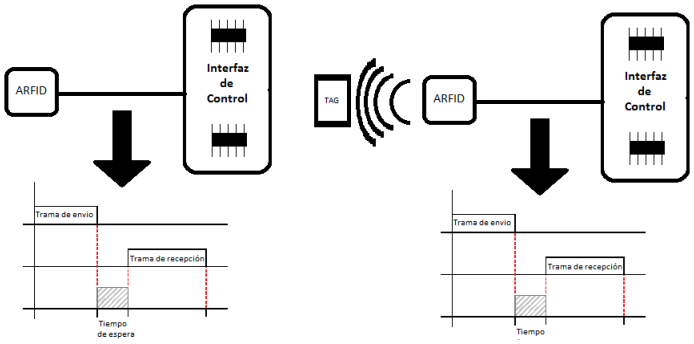


Figura 36: Gráfica de trafico

La figura 36muestra un diagrama a bloques que representa como se conectan el IC y al ARFID además hay una representación de una grafica que muestran los tiempos de envío y recepción de la trama, así como el tiempo de respuesta entre envío y recepción, el cual es uno de los resultados obtenidos.

Se conectó el circuito y se programó el microcontrolador para que enviara tramas de forma continua, el puerto RA0 del microcontrolador se habilitó para que se prendiera un led, con la finalidad de medir el tiempo de respuesta de ARFID, debido a que el led se encendía al finalizar el envío de la trama y se apagaba al recibir la trama proveniente de la ARFID.

Mediante el uso del osciloscopio se pudo observar el tiempo que tardaba en responder la ARFID el cual es 0.9ms, de igual forma se observó el tiempo que se lleva IC en enviar y recibir las tramas, cuando había y no había presencia de tag.

Como se puede apreciar existen tres bloques, el primero que muestra el envió de trama, el segundo muestra la recepción de la trama y el ultimo que muestra el periodo entre el envió y la recepción de tramas.

El primer y el segundo bloque pueden ser calculados de forma precisa mediante ecuaciones, esto se debe ha que se tiene conocimiento sobre la longitud en bytes que conforman cada una de las tramas.

El tercer bloque debe calcularse mediante el uso del osciloscopio debido a que es un periodo donde la señal se mantiene en estado low.

Otro punto muy importante es la longitud de la trama que recibe el IC, ya que cuando existe una tag presente la trama tiene una longitud de 16 bytes y difiere de la longitud cuando no existe una tag presente la cual es de 6 bytes; por lo que se deben modelar tres casos, el primero cuando no existe una tag presente, el segundo cuando existe por lo menos una tag presente y la tercera cuando existe una tag presente en cada una de las ARFID.

Caso 1

Como bien se comento anteriormente, el primer caso es referente a cuando no existe una tag presente.

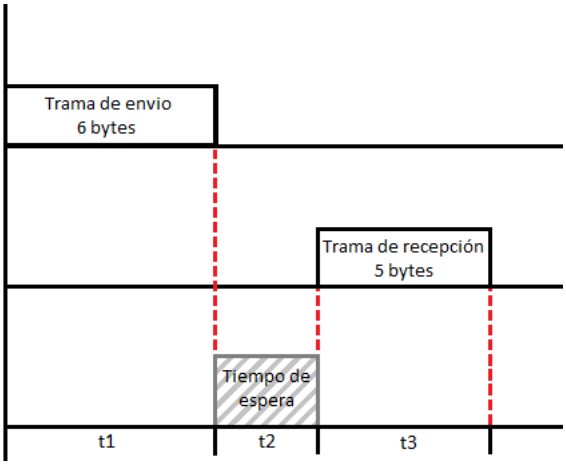


Figura 37: Grafica cuando no hay tag preseste

En la figura anterior se muestran un bloque de envió de 6 bytes, uno de recepción de 5 bytes y el tiempo de espera. El tiempo que tarda en completarse el ciclo de comunicación para este caso es el siguiente:

El tiempo en trasmitir un byte durante la comunicación se define de la siguiente forma:

$$tb = \frac{1}{vel} \times byte$$

Donde vel es la velocidad de trasferencia y byte es la suma de los ocho bit que con forman un byte, más un bit de arranque y uno de parada para el protocolo de comunicaciones serial asincrono, sustituyendo:

$$tb = \frac{1}{9600} \times 10 = 1.04ms$$

Por lo que el tiempo de tiempo de la trama de envío es:

$$t_1 = nbytes \times tb$$

donde nbytes es un conjunto de bytes y tb es el tiempo que se tarda en trasmitir un bytes entonces:

$$t_1 = 6 \times 1.04 = 6.24ms$$

y para tiempo de la trama de recepción es:

$$t_2 = nbytes \times tb$$

$$t_2 = 5 \times 1.04 = 5.20ms$$

Por lo tanto el tiempo del clico tc es:

$$tc = t_1 + t_2 + t_3$$

$$tc = 6.24 + t_2 + 5.20ms$$

Como se menciono anteriormente, mediante el uso del osciloscopio se midió el tiempo que tardaba el periodo entre enviar y recibir tramas el resultado obtenido fue de 0.9ms; por lo que el tiempo total del ciclo de comunicación cuando no existe tag presente es de:

$$tc = 6.24 + 0.9 + 5.20 = 12.34ms$$

Esto por la cantidad de ARFID conectadas al IC; entonces el barrido de las 7 ARFID conectadas al IC es de:

$$t_{barrido} = 12.34 \times 7 = 86.38ms$$

De igual manera que envía y recibe tramas el IC tiene la tarea de enviar la trama de identificadores la cual es el conjunto de todos los identificadores presentes, además la velocidad de transmición entre la IC y la PC es de 38400 bit por seg.

Para este primer caso se transmitirá 3 byte únicamente (inicio, longitud, ntag) definidos previamente en el desarrollo véase(Subseccion 3.2.4.), por lo que primero el tiempo en enviar un byte a la PC es de:

$$tb^i = \frac{1}{38400} x10 = 0.26ms$$

Con lo que se tiene que el tiempo de transmisión de la trama identificadores es

$$t_{total} = t_{barrido} + tb^i \times 3 = 86.38 + 3 \times 0.26 = 87.16ms$$

CASO 2

A continuación se muestra el clico de comunicación cuando existe una tag presente.

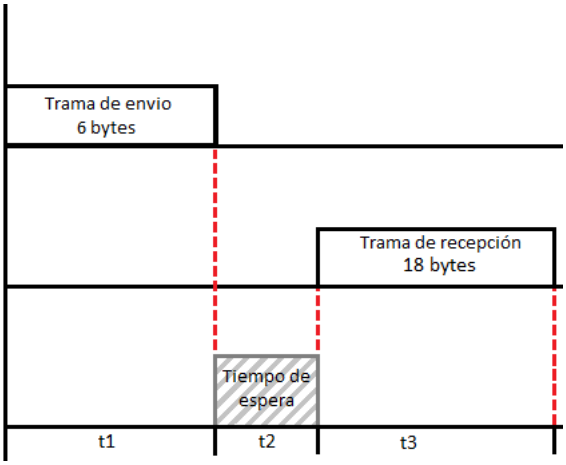


Figura 38: Grafica cuando hay una tag presente

Para este caso cambia la longitud de la trama de recepción a 18 bytes, por lo que únicamente cambia t3

$$t_3 = 18 \times 104 = 18.32ms$$

Por lo tanto el tiempo total del ciclo de comunicación es de

$$tc = t_1 + t_2 + t_3 = 6.24 + 0.9 + 18.32 = 25.46ms$$

Con esto podemos calcular el proceso de barrido cuando existe por lo menos una tag presente en una ARFID, este es igual:

$$t_{barrido} = (tc \times 6) + 25.46 = 99.5ms$$

$$t_{barrido} = (12.34 \times 6) + 25.46 = 99.5ms$$

Como únicamente necesitamos el valor de la tag se desecha los bytes innecesarios reduciendo la longitud de 18 bytes a 12 más los 3 byte previamente definidos, dando un total de 15 bytes a una velocidad 38400 bit por segundo, entonces el tiempo total es

$$t_{total} = 99.5 + (0.26 \times 15) = 103.4ms$$

CASO 3

Por último esta el caso cuando existe una tag presente en cada una de las ARFID, en este caso el tiempo de barrido de las 7 ARFID es:

$$t_{barrido} = 25.46 \times 7 = 178.22ms$$

por lo tanto la longitud de la trama a envia a la PC es de 48 byte que contiene los identificadores y los 3 bytes obligatorios, por lo que el tiempo total es de

$$t_{total} = 178.22 + (0.26 \times 51) = 191.48ms$$

Caso/tiempo	Trama envio	Espera	trama recepción	Barrido	Envio de tag	Total
1	6.24ms	0.9ms	5.20ms	86.38ms	0.78ms	87.16ms
2	6.24ms	0.9ms	18.32ms	99.5ms	3.9ms	103.04ms
3	6.24ms	0.9ms	18.32ms	178.22ms	13.26ms	191.48ms

Cuadro 3: Tabla de resultados

Como se puede observar la tabla muestra un resumen de los tiempo obtenidos de los cálculos, por lo que podemos decir que el proceso de lectura de tag de 7 y trasmisión de el conjunto de identificadores no puede ser menor a 87.16 ms usando una velocidad de 9600 Bps hacia la ARFID y 38400 Bps hacia la PC, además el tiempo máximo en realizar el proceso de lectura de tag de 7 y trasmisión de el conjunto de identificadores es igual o menor a 191.48 ms con una velocidad 9600bps hacia la ARFID y 38400 Bps hacia la PC.

5. Conclusiones

Durante el desarrollo de la Tesis se fue cumpliendo con los objetivos que se tenía previsto, el análisis de la ARFID fue éxito permitiendo obtener un panorama de su funcionamiento y como manipularlo.

Otro objetivo era el diseño de un conjunto de reglas para poder definir la forma en que se comunicaría la ARFID con la IC y con la PC con la finalidad de modelar con mayor entendimiento el algoritmo.

El modelado mediante las Redes de Petri fue de gran ayuda para la tesis ya que permitió modelar las reglas del algoritmo de forma eficiente y ordenada, dándole un buen nivel de coherencia al desarrollo.

La implementación de software de simulación permitió analizar y depurar error en el modelado brindando al modelo una mayor eficiencia y dinamismo, todo esto trayendo como resultado un modelo funcional que permitirá controlar de maneja rápida y eficiente varias antenas sin sacrificar tiempo en las lectura.

Por último se logró obtener una mejor respuesta de las ARFID con respecto al tiempo, disminuyéndolo significativamente la petición de lectura de tag y su respuesta.

Referencias

[1] Janette Cardoso, R. V, Redes de Petri,: Florianópolis, 1997.

[2] Sharp Robin, Principles of Protocol Design, Berlin: Springer, 2008.

[3] Claude Girault. Rüdiger Valk,Petri Nets for System Enginnering. Francia: Springer,2003.

[4] Holzmann Gerard, Design and Validation of Computer Protocols: New Jersey: Prentic, 1991.

[5] Barcelo Ordinas Josep M., Estructura de redes de computadores.Barcelona:UOC, 2008.

[6] Brown, Dennis. E., Implementación de RFID. McGraw-Hill Professional, 2007.

[7] Tadao Murata, Petri Nets: Properties, Analysis and applications. IEEE, 1990.

[8] eltima.(18, Ago 2012). Serial Port Monitor - Eltima Software [online]. Available : http://www.eltima.com/products/serial-port-monitor/

[11] Mario L. Ruiz, Francisco Vázquez and Juan Galán, Control de un lector RFID mediante micro-controladores. Actualidad Preventiva andaluza, Sevilla:Asapri, 2009, pp12.

[12] Javier I. Portillo García, Ana Belén Bermejo Nieto, Ana M. Bernardos Barbolla, Tecnología RFID: aplicaciones en el ámbito de la salud, Madrid 2011.

[13] Mario E. Salazar, Néstor Peña (17, Ago 2012). Aplicación De Redes De Petri A La Evaluación De Desempeño De Sistemas De Comunicaciones- Revista de Ingenieros[online]. Availble: http://revistaing.uniandes.edu.co/index.php?idr=20

[14] Granados Rojas Benito, Jiménez Saucedo Mario A., Vallejo Alarcón Manuel A., González Navarro Yesenia E., Villarreal Cervantes Miguel G. and Corona Ramírez Leonel G, Protocolo de Comunicación en Anillo para el Control de un Robot Móvil Modular in Congress Nacional de Mecatrónica, Puerto Vallarta, Jalisco, 2011.

[15] Karol Nataly Benavides Lopez .Diseño de módulos de comunicación para dispositivos RFID. Dep. Elect. Eng. Univ. Santiago de cali, Colombia, 2008.

[16] Almeida P. Marco, Roldán M. Elsa , M. Soraya, Implementación de un Protocolo de Comunicación para el Control de los Movimientos de un Brazo Robot a través del Interfaz Bluetooth de un Teléfono Celular. Univ. Escuela Politecnica Nacional de Quito, Ecuador, 2011.

[17] Diego R. Llanos Ferraris, VSR-COMA: Un Protocolo de Coherencia Cache con Reemplazo para Sistemas Multicomputadores con gestión de memoria de tipo COMA. Unv. de Valladolid Dep. de Informática, España 2000

Apéndice A: Análisis de la Antena.

Cuando se adquirió la antena " Gen 2 Reader GIO" se nos proporcionó los controladores y el programa de demostración así como las librerías para realizar una aplicación específica para la antena, como se menciona en la tesis únicamente puedes controlar un número reducido de antenas máximo 3 a 4 por lo que se pretende crear la interfaz de control, pero no se sabe el funcionamiento de la antena o tramas por lo que se utilizó el programa Reader Demo fig. 39 v1.2.1. que viene con la antena, para poder monitorear las salidas y entradas de tramas.

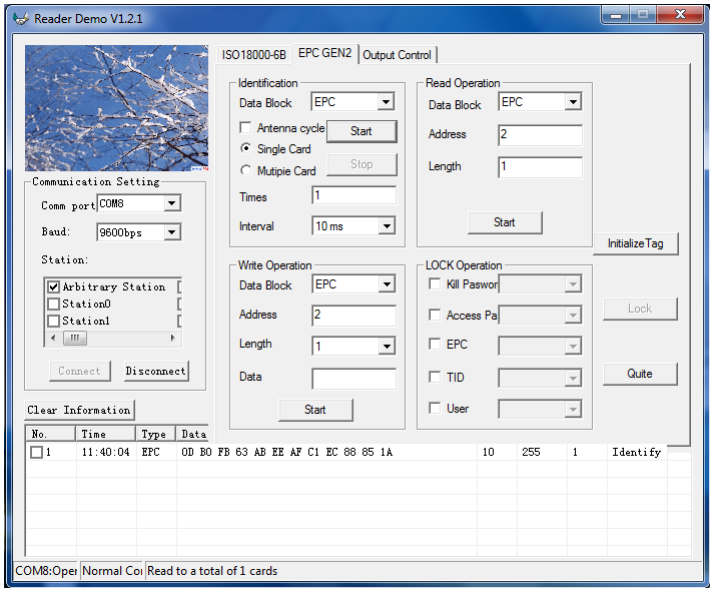


Figura 39: Reader Demo

El monitoreo de la salidas y entradas se realizó con la ayuda de snnifer de puerto serie Serial Port Monitor y una software de terminal llamado Ultra Serial Port Monitor fig40, el primero para poder extraer las diferentes tramas que enviaba el software a la antena y viceversa y el segundo se empleó para las pruebas de las tramas durante la prueba de análisis y de desarrollo del circuito prototipo como herramienta de test.

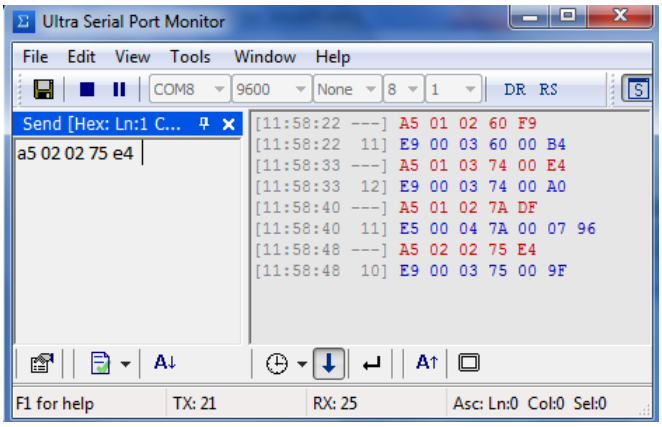


Figura 40: Ultra Serial Port Monitor

Apéndice B: Diseño de la interfaces de control

Una vez identificada las tramas y realizado pruebas con la antena se continuó con la parte de diseño de la interfaces que tendrá la función de controlar las antenas y de retransmitir la información de las Tag a la PC.

Para esto se realizó un prototipo usando pic 16f84A como cerebro central de la interfaz esto con la finalidad de diseñar un programa donde se pudiera verificar cada uno de los comandos obtenidos en el análisis de la antena.

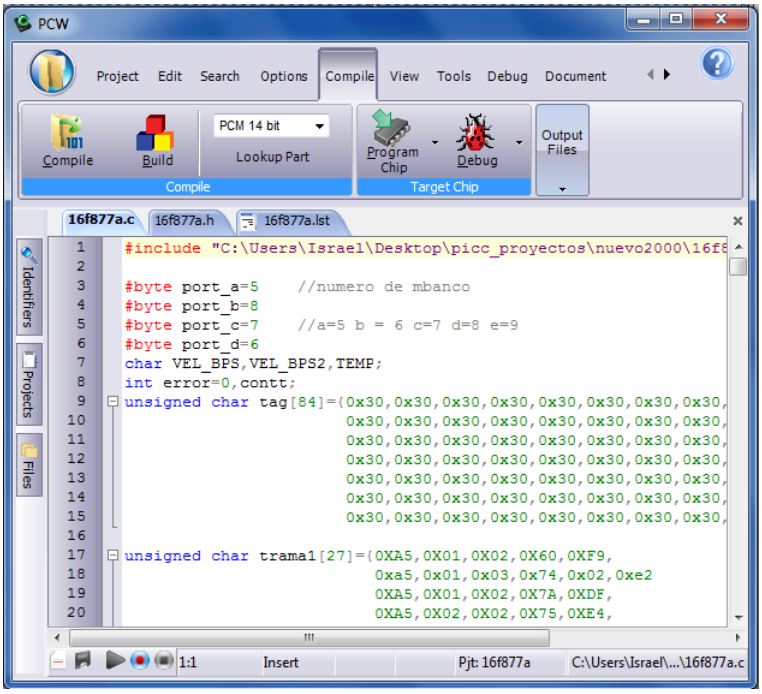


Figura 41: Pic-C

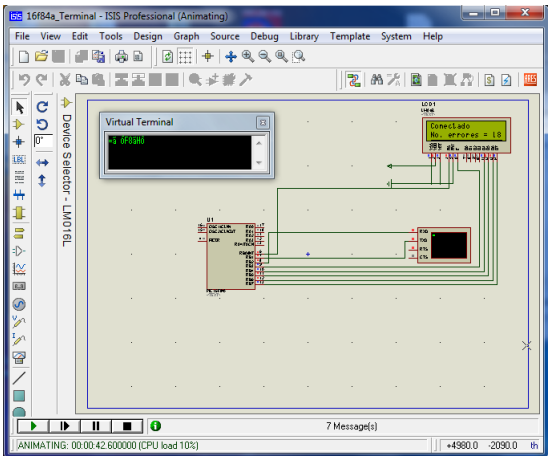


Figura 42: ISIS

El diseño se realizó inicialmente usando el Programa Proteous ISIS 7 fig42 para el diseño de placa y circuito además de que costa de un simulado de circuitos muy versátil. Otro software que se usó fue PIC-C fig. 41para la codificación del programa.

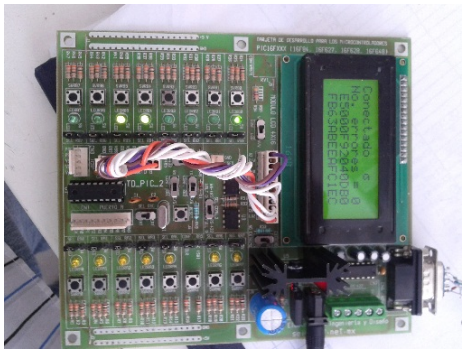


Figura 43: Placa Prototipo

Una vez completadas las pruebas fig 43 necesarias y verificar el funcionamiento correcto bajo ese microcontrolador, se decidió tomar el pic 16f877a como cerebro central final de la interfaz final por sus característica de mayor RAM yROM, ádemas de que este micro-controlador permite crear puertos TTL virtuales (UART), lo que nos será de mucha ayuda debido a que se pretende conectar 7 antenas al dispositivo y la PC.

El diseño del circuito que servirá como interfaz central en Proteus ademas que se realizó su simulación. A continuación se muestra como se conectó de forma virtual el circuito.

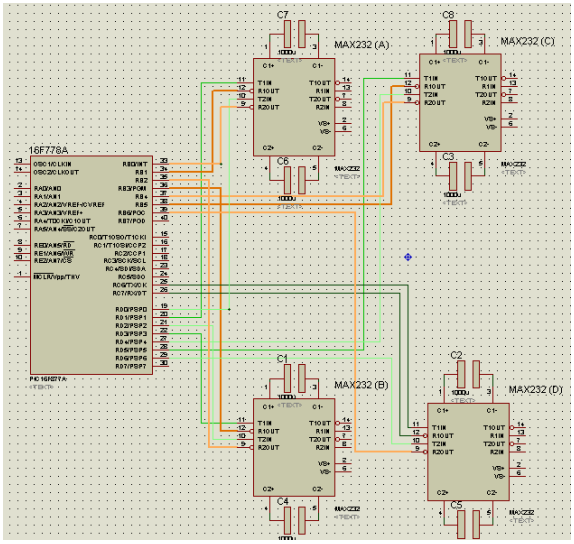


Figura 44: Diagrama IC (ISIS)

Con la finalidad de verificar correctamente la funcionalidad el protocolo se llevó el diseño anterior a un protoboard.

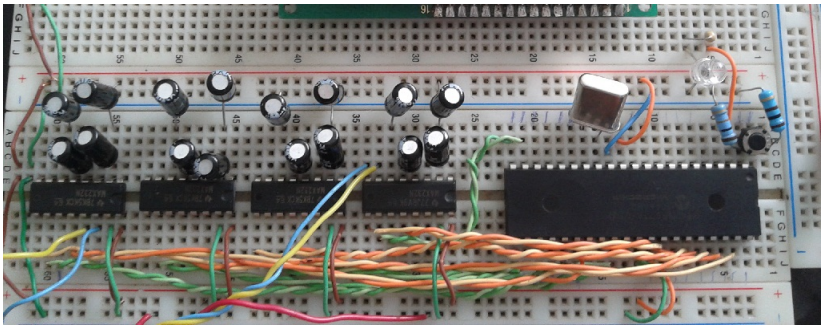


Figura 45: Prototipo IC

Apéndice C: Simulación con Software Visual Object Net ++

La Simulación de la Red de Petri se realizó mediante el software Visual Object el cual permite plasmar un conjunto de reglas y diseñar una red. El software consta con todo lo necesario plazas, transacciones y arcos. Una de sus características es poder implementar tiempos, cuneta con dos botones plazas, dos botones de transacciones y un botón de arcos.

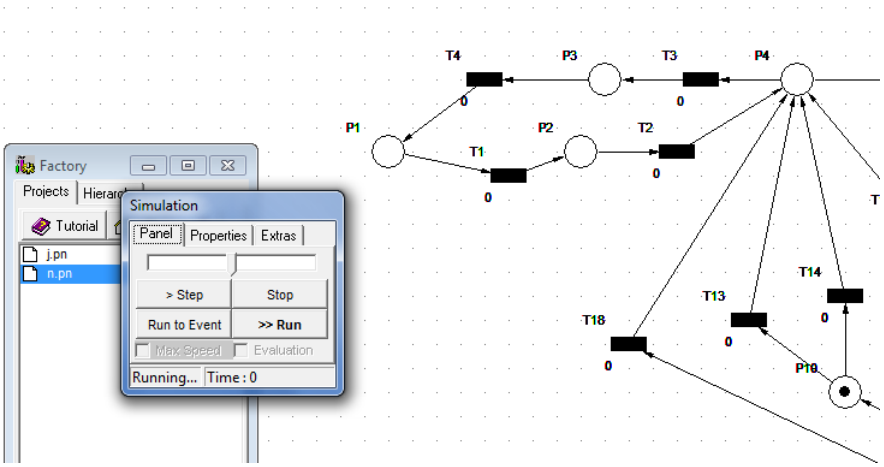


Figura 46: Visual object

Apendice D : Osiloscopio

Una ves que se codifico y se copilo el codigo se puso a prueba para porder medir los tiempos de respuesta entre envio y recepcion de tramas.

Los resultados fueron los siguientes:

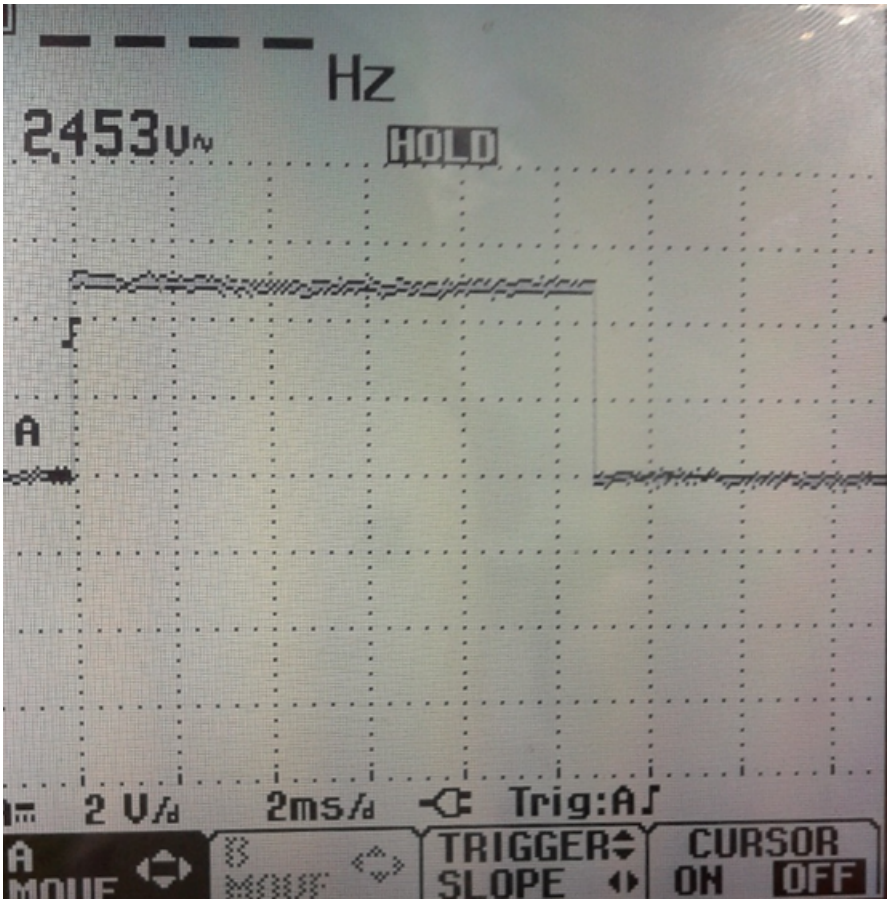


Figura 47: Envio y recepcion

enla figura 47se ve claramente las tramas de envio y recepcioin juntas cada cuadro equivale a 2ms por lo que el calculo hecho se aproxima al valor obtenido por el osiloscopio.

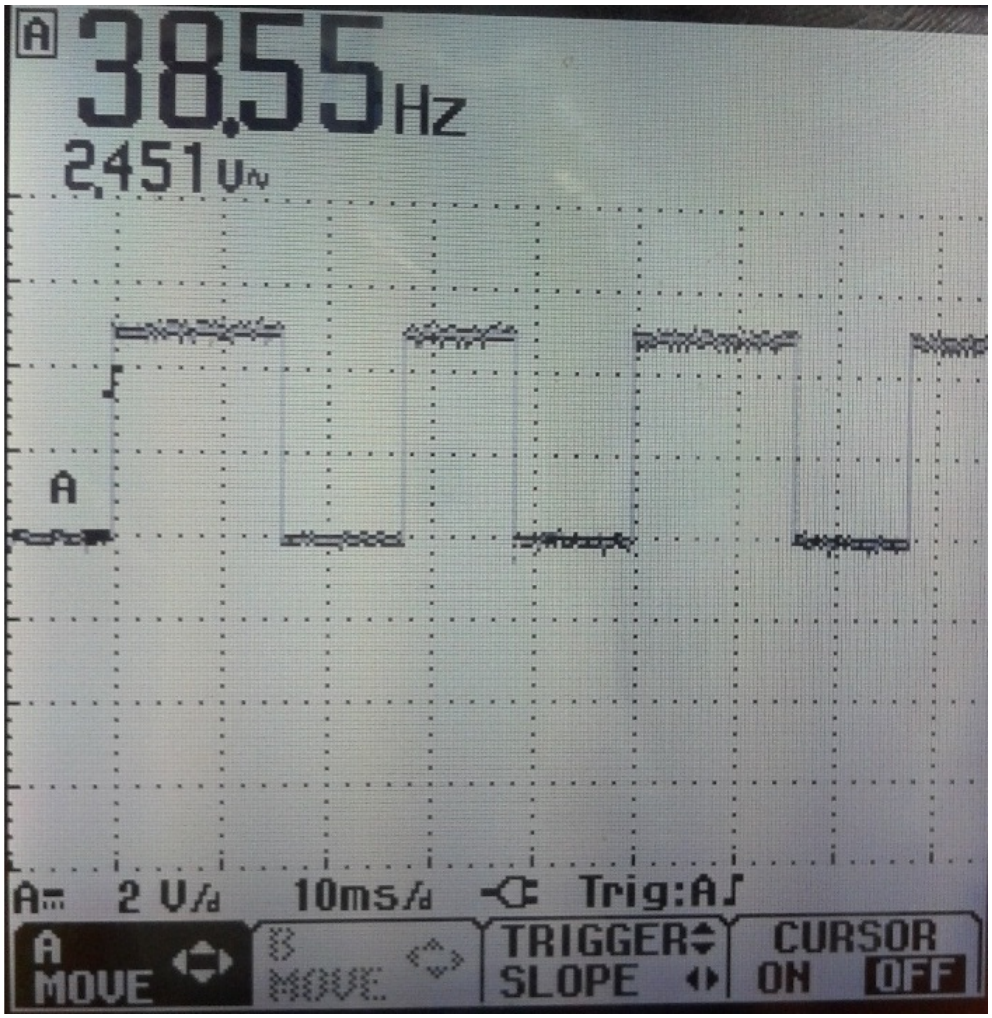


Figura 48: Lapso de tiempo entre envio y recepcion

En la figura 48 se muestra las mediciones del lapso de tiempo el cual es aproximado a 0.9ms.

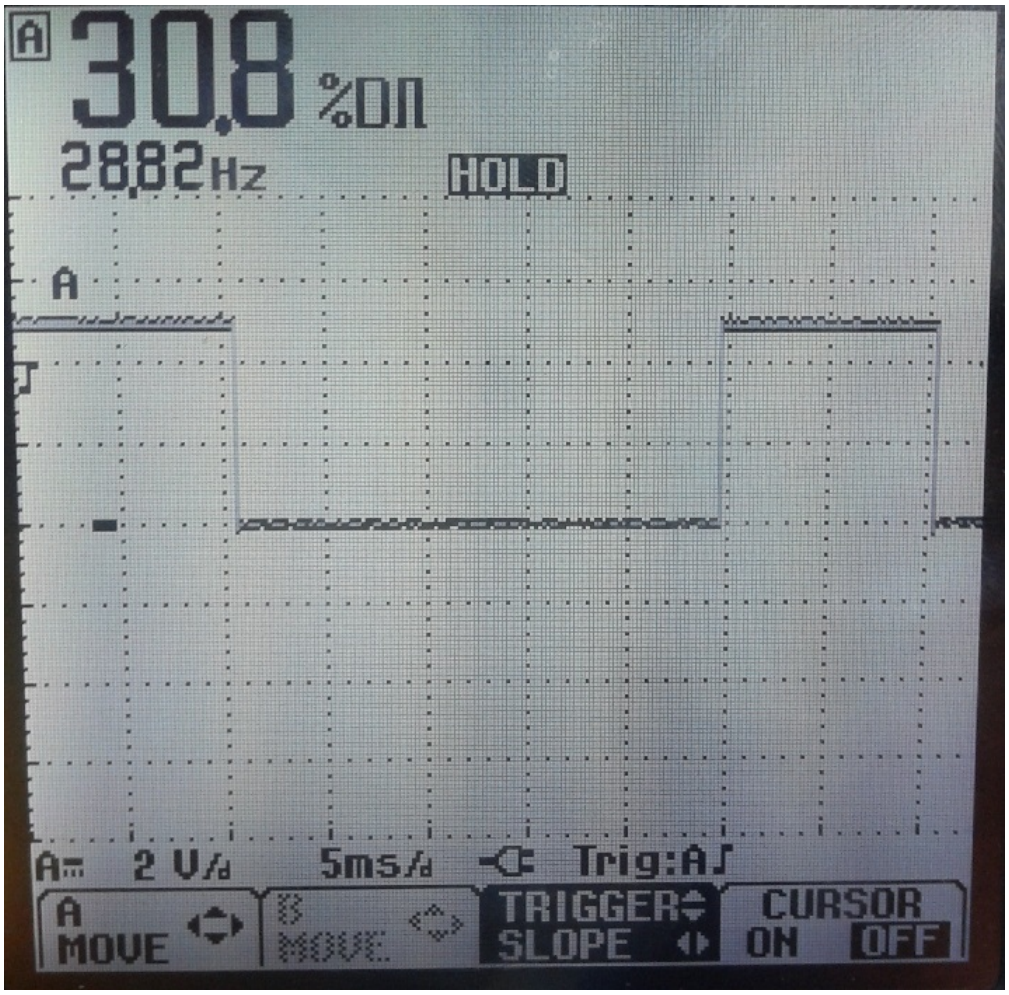


Figura 49: Trama de envio y recepcion cuando hay tag

En la figura 49 se muestra la trama de envio y recepcion juntas pero en esta ocacion exite la presencia de una tag. Arrogando un valor aproximado 25ms

Apéndice E: Código fuente

El resultado del algoritmo fue el código fuente que permite el manejo de varias ARFID a continuación se muestra dicho código.

```
#include "C:\Users\Israel\Desktop\picc_proyectos\nuevo2000\16f877a.h"

#byte port_a=5    //numero de mbanco
#byte port_b=8
#byte port_c=7    //a=5 b = 6 c=7 d=8 e=9
#byte port_d=6
char VEL_BPS,VEL_BPS2,TEMP;
int error=0,contt;
unsigned char tag[84]={0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0
                       0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0
                       0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0
                       0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0
                       0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0
                       0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0
                       0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0x30,0

unsigned char trama1[27]={0XA5,0X01,0X02,0X60,0XF9,
                          0xa5,0x01,0x03,0x74,0x02,0xe2
                          0XA5,0X01,0X02,0X7A,0XDF,
                          0XA5,0X02,0X02,0X75,0XE4,
                          0xa5,0x01,0x03,0x92,0x04,0xc2};

unsigned char trama2[25]={0XE9,0X00,0X03,0X60,0X00,0XB4,
                          0XE9,0X00,0X03,0X74,0X00,0XA0,
                          0XE5,0X00,0X04,0X7A,0X00,0X07,0X96,
                          0XE9,0X00,0X03,0X75,0X00,0X9F};

////////////////////////////////////////////////////////////////////////
//
// OTROS
//
////////////////////////////////////////////////////////////////////////
void test(int n){
byte value, cnt=0;

set_tris_a(0);
port_a =0;
value=0x01;


while (cnt<n)
 {
  port_a= value;
   DELAY_MS(250);
  port_a= 0x00;
    DELAY_MS(250);
     cnt++;
 }

}




char recep0(){

char RECEP,CONTA_REC;

#ASM

      BSF      STATUS,5
      BSF      port_d,0
      BCF      STATUS,5

et5:
      BTFSC    port_d,0
      GOTO     et5
```

```
 69:       MOVLW    8
 70:       MOVWF    CONTA_REC
 71:       CLRF     RECEP
 72:       BSF      CONTA_REC,7
 73:       GOTO     et6
 74: et3:
 75:       BCF      CONTA_REC,7
 76:       GOTO     et6
 77: et1:
 78:       BCF      STATUS,0
 79:       BTFSC    port_d,0
 80:       BSF      STATUS,0
 81:       RRF      RECEP,F
 82:       BSF      CONTA_REC,6
 83:       GOTO     et6
 84: et2:
 85:       BCF      CONTA_REC,6
 86:       DECFSZ   CONTA_REC,F
 87:       GOTO     et1
 88:       MOVF     RECEP,W
 89:       MOVWF    TEMP
 90:       goto     finx
 91:
 92: et6:
 93:       MOVF     VEL_BPS,W //MOVLW  0X25;0XA7 9600BPS
 94:       BTFSC    CONTA_REC,7
 95:       MOVF     VEL_BPS2,W //MOVLW  0X0D   ;2DH 9600BPS
 96:       MOVWF    TEMP
 97: et4:
 98:       DECFSZ   TEMP,F
 99:       GOTO     et4
100:       NOP
101:       BTFSC    CONTA_REC,7
102:       GOTO     et3
103:       BTFSC    CONTA_REC,6
104:       GOTO     et2
105:       GOTO     et1
106: finx:
107:       NOP
108:   #ENDASM
109:
110: return RECEP;
111: }
112:
113: char recep1(){
114:
115: char RECEP,CONTA_REC;
116:
117: #ASM
118:
119:       BSF      STATUS,5
120:       BSF      port_d,1
121:       BCF      STATUS,5
122:
123: et5:
124:       BTFSC    port_d,1
125:       GOTO     et5
126:       MOVLW    8
127:       MOVWF    CONTA_REC
128:       CLRF     RECEP
129:       BSF      CONTA_REC,7
130:       GOTO     et6
131: et3:
132:       BCF      CONTA_REC,7
133:       GOTO     et6
134: et1:
135:       BCF      STATUS,0
136:       BTFSC    port_d,1
```

```
137:        BSF       STATUS,0
138:        RRF       RECEP,F
139:        BSF       CONTA_REC,6
140:        GOTO      et6
141: et2:
142:        BCF       CONTA_REC,6
143:        DECFSZ    CONTA_REC,F
144:        GOTO      et1
145:        MOVF      RECEP,W
146:        MOVWF     TEMP
147:        goto      finx
148:
149: et6:
150:        MOVF      VEL_BPS,W //MOVLW  0X25;0XA7 9600BPS
151:        BTFSC     CONTA_REC,7
152:        MOVF      VEL_BPS2,W //MOVLW  0X0D   ;2DH 9600BPS
153:        MOVWF     TEMP
154: et4:
155:        DECFSZ    TEMP,F
156:        GOTO      et4
157:        NOP
158:        BTFSC     CONTA_REC,7
159:        GOTO      et3
160:        BTFSC     CONTA_REC,6
161:        GOTO      et2
162:        GOTO      et1
163: finx:
164:        NOP
165:   #ENDASM
166:
167: return RECEP;
168: }
169:
170: char recep2(){
171:
172: char RECEP,CONTA_REC;
173:
174: #ASM
175:
176:        BSF       STATUS,5
177:        BSF       port_d,2
178:        BCF       STATUS,5
179:
180: et5:
181:        BTFSC     port_d,2
182:        GOTO      et5
183:        MOVLW     8
184:        MOVWF     CONTA_REC
185:        CLRF      RECEP
186:        BSF       CONTA_REC,7
187:        GOTO      et6
188: et3:
189:        BCF       CONTA_REC,7
190:        GOTO      et6
191: et1:
192:        BCF       STATUS,0
193:        BTFSC     port_d,2
194:        BSF       STATUS,0
195:        RRF       RECEP,F
196:        BSF       CONTA_REC,6
197:        GOTO      et6
198: et2:
199:        BCF       CONTA_REC,6
200:        DECFSZ    CONTA_REC,F
201:        GOTO      et1
202:        MOVF      RECEP,W
203:        MOVWF     TEMP
204:        goto      finx
```

```
205:
206: et6:
207:       MOVF     VEL_BPS,W //MOVLW  0X25;0XA7 9600BPS
208:       BTFSC    CONTA_REC,7
209:       MOVF     VEL_BPS2,W //MOVLW  0X0D   ;2DH 9600BPS
210:       MOVWF    TEMP
211: et4:
212:       DECFSZ   TEMP,F
213:       GOTO     et4
214:       NOP
215:       BTFSC    CONTA_REC,7
216:       GOTO     et3
217:       BTFSC    CONTA_REC,6
218:       GOTO     et2
219:       GOTO     et1
220: finx:
221:       NOP
222:   #ENDASM
223:
224: return RECEP;
225: }
226:
227: char recep3(){
228:
229: char RECEP,CONTA_REC;
230:
231: #ASM
232:
233:       BSF      STATUS,5
234:       BSF      port_d,3
235:       BCF      STATUS,5
236:
237: et5:
238:       BTFSC    port_d,3
239:       GOTO     et5
240:       MOVLW    8
241:       MOVWF    CONTA_REC
242:       CLRF     RECEP
243:       BSF      CONTA_REC,7
244:       GOTO     et6
245: et3:
246:       BCF      CONTA_REC,7
247:       GOTO     et6
248: et1:
249:       BCF      STATUS,0
250:       BTFSC    port_d,3
251:       BSF      STATUS,0
252:       RRF      RECEP,F
253:       BSF      CONTA_REC,6
254:       GOTO     et6
255: et2:
256:       BCF      CONTA_REC,6
257:       DECFSZ   CONTA_REC,F
258:       GOTO     et1
259:       MOVF     RECEP,W
260:       MOVWF    TEMP
261:       goto     finx
262:
263: et6:
264:       MOVF     VEL_BPS,W //MOVLW  0X25;0XA7 9600BPS
265:       BTFSC    CONTA_REC,7
266:       MOVF     VEL_BPS2,W //MOVLW  0X0D   ;2DH 9600BPS
267:       MOVWF    TEMP
268: et4:
269:       DECFSZ   TEMP,F
270:       GOTO     et4
271:       NOP
272:       BTFSC    CONTA_REC,7
```

```
273:       GOTO     et3
274:       BTFSC    CONTA_REC,6
275:       GOTO     et2
276:       GOTO     et1
277: finx:
278:       NOP
279:   #ENDASM
280:
281: return RECEP;
282: }
283:
284: char recep4(){
285:
286: char RECEP,CONTA_REC;
287:
288: #ASM
289:
290:       BSF      STATUS,5
291:       BSF      port_d,4
292:       BCF      STATUS,5
293:
294: et5:
295:       BTFSC    port_d,4
296:       GOTO     et5
297:       MOVLW    8
298:       MOVWF    CONTA_REC
299:       CLRF     RECEP
300:       BSF      CONTA_REC,7
301:       GOTO     et6
302: et3:
303:       BCF      CONTA_REC,7
304:       GOTO     et6
305: et1:
306:       BCF      STATUS,0
307:       BTFSC    port_d,4
308:       BSF      STATUS,0
309:       RRF      RECEP,F
310:       BSF      CONTA_REC,6
311:       GOTO     et6
312: et2:
313:       BCF      CONTA_REC,6
314:       DECFSZ   CONTA_REC,F
315:       GOTO     et1
316:       MOVF     RECEP,W
317:       MOVWF    TEMP
318:       goto     finx
319:
320: et6:
321:       MOVF     VEL_BPS,W //MOVLW  0X25;0XA7 9600BPS
322:       BTFSC    CONTA_REC,7
323:       MOVF     VEL_BPS2,W //MOVLW  0X0D   ;2DH 9600BPS
324:       MOVWF    TEMP
325: et4:
326:       DECFSZ   TEMP,F
327:       GOTO     et4
328:       NOP
329:       BTFSC    CONTA_REC,7
330:       GOTO     et3
331:       BTFSC    CONTA_REC,6
332:       GOTO     et2
333:       GOTO     et1
334: finx:
335:       NOP
336:   #ENDASM
337:
338: return RECEP;
339: }
340:
```

```
341: char recep5(){
342:
343: char RECEP,CONTA_REC;
344:
345: #ASM
346:
347:       BSF      STATUS,5
348:       BSF      port_d,5
349:       BCF      STATUS,5
350:
351: et5:
352:       BTFSC    port_d,5
353:       GOTO     et5
354:       MOVLW    8
355:       MOVWF    CONTA_REC
356:       CLRF     RECEP
357:       BSF      CONTA_REC,7
358:       GOTO     et6
359: et3:
360:       BCF      CONTA_REC,7
361:       GOTO     et6
362: et1:
363:       BCF      STATUS,0
364:       BTFSC    port_d,5
365:       BSF      STATUS,0
366:       RRF      RECEP,F
367:       BSF      CONTA_REC,6
368:       GOTO     et6
369: et2:
370:       BCF      CONTA_REC,6
371:       DECFSZ   CONTA_REC,F
372:       GOTO     et1
373:       MOVF     RECEP,W
374:       MOVWF    TEMP
375:       goto     finx
376:
377: et6:
378:       MOVF     VEL_BPS,W //MOVLW  0X25;0XA7 9600BPS
379:       BTFSC    CONTA_REC,7
380:       MOVF     VEL_BPS2,W //MOVLW  0X0D   ;2DH 9600BPS
381:       MOVWF    TEMP
382: et4:
383:       DECFSZ   TEMP,F
384:       GOTO     et4
385:       NOP
386:       BTFSC    CONTA_REC,7
387:       GOTO     et3
388:       BTFSC    CONTA_REC,6
389:       GOTO     et2
390:       GOTO     et1
391: finx:
392:       NOP
393:   #ENDASM
394:
395: return RECEP;
396: }
397:
398: char recep6(){
399:
400: char RECEP,CONTA_REC;
401:
402: #ASM
403:
404:       BSF      STATUS,5
405:       BSF      port_d,6
406:       BCF      STATUS,5
407:
408: et5:
```

```
409:       BTFSC    port_d,6
410:       GOTO     et5
411:       MOVLW    8
412:       MOVWF    CONTA_REC
413:       CLRF     RECEP
414:       BSF      CONTA_REC,7
415:       GOTO     et6
416: et3:
417:       BCF      CONTA_REC,7
418:       GOTO     et6
419: et1:
420:       BCF      STATUS,0
421:       BTFSC    port_d,6
422:       BSF      STATUS,0
423:       RRF      RECEP,F
424:       BSF      CONTA_REC,6
425:       GOTO     et6
426: et2:
427:       BCF      CONTA_REC,6
428:       DECFSZ   CONTA_REC,F
429:       GOTO     et1
430:       MOVF     RECEP,W
431:       MOVWF    TEMP
432:       goto     finx
433:
434: et6:
435:       MOVF     VEL_BPS,W //MOVLW  0X25;0XA7 9600BPS
436:       BTFSC    CONTA_REC,7
437:       MOVF     VEL_BPS2,W //MOVLW  0X0D   ;2DH 9600BPS
438:       MOVWF    TEMP
439: et4:
440:       DECFSZ   TEMP,F
441:       GOTO     et4
442:       NOP
443:       BTFSC    CONTA_REC,7
444:       GOTO     et3
445:       BTFSC    CONTA_REC,6
446:       GOTO     et2
447:       GOTO     et1
448: finx:
449:       NOP
450:   #ENDASM
451:
452: return RECEP;
453: }
454:
455: ////////////////////////////////////////////////////////////////////
456: //
457: // CODIGO DE ENVIO
458: //
459: ////////////////////////////////////////////////////////////////////
460: void trans0(char TRANS){
461:
462: char CONT,REGX2;
463:
464:
465: #ASM
466:       BSF      STATUS,5
467:       BCF      port_b,0
468:       BCF      STATUS,5
469:
470:       BCF      port_b,0 //<--
471:       MOVLW    8
472:       MOVWF    CONT   //;0D =CONTA1
473:       GOTO     AA
474: AA:
475:       NOP
476:       BSF      CONT,7
```

```
477:        GOTO      TT
478: ZZ:
479:        BCF       CONT,7
480: OTROBIT:
481:        RRF       TRANS,F       //;12H =TRANS
482:        BTFSC     STATUS,0
483:        BSF       port_b,0  //<--
484:        BTFSS     STATUS,0
485:        BCF       port_b,0 //<--
486:        BSF       CONT,6
487:        GOTO      TT
488: RR:
489:        BCF       CONT,6
490:        DECFSZ    CONT,F
491:        GOTO      OTROBIT
492:        GOTO      AQUI
493: AQUI:
494:        NOP
495:        BSF       port_b,0 //<-- 0
496: TT:                    //;MOVLW   0X50
497:        MOVF      VEL_BPS,W
498:        MOVWF     REGX2
499: WTR:
500:        DECFSZ    REGX2,F
501:        GOTO      WTR
502:        GOTO      AQUI2
503: AQUI2:
504:        BTFSC     CONT,7
505:        GOTO      ZZ
506:        BTFSC     CONT,6
507:        GOTO      RR
508:        //RETURN
509: #ENDASM
510:
511: }
512:
513: void trans1(char TRANS){
514:
515: char CONT,REGX2;
516:
517:
518: #ASM
519:        BSF       STATUS,5
520:        BCF       port_b,1 ///
521:        BCF       STATUS,5
522:
523:        BCF       port_b,1 ///
524:        MOVLW     8
525:        MOVWF     CONT   //;0D =CONTA1
526:        GOTO      AA
527: AA:
528:        NOP
529:        BSF       CONT,7
530:        GOTO      TT
531: ZZ:
532:        BCF       CONT,7
533: OTROBIT:
534:        RRF       TRANS,F       //;12H =TRANS
535:        BTFSC     STATUS,0
536:        BSF       port_b,1///
537:        BTFSS     STATUS,0
538:        BCF       port_b,1///
539:        BSF       CONT,6
540:        GOTO      TT
541: RR:
542:        BCF       CONT,6
543:        DECFSZ    CONT,F
544:        GOTO      OTROBIT
```

```
545:        GOTO      AQUI
546: AQUI:
547:        NOP
548:        BSF       port_b,1///
549: TT:                      //;MOVLW   0X50
550:        MOVF      VEL_BPS,W
551:        MOVWF     REGX2
552: WTR:
553:        DECFSZ    REGX2,F
554:        GOTO      WTR
555:        GOTO      AQUI2
556: AQUI2:
557:        BTFSC     CONT,7
558:        GOTO      ZZ
559:        BTFSC     CONT,6
560:        GOTO      RR
561:        //RETURN
562: #ENDASM
563:
564: }
565:
566: void trans2(char TRANS){
567:
568: char CONT,REGX2;
569:
570:
571: #ASM
572:        BSF       STATUS,5
573:        BCF       port_b,2 ///
574:        BCF       STATUS,5
575:
576:        BCF       port_b,2 ///
577:        MOVLW     8
578:        MOVWF     CONT    //;0D =CONTA1
579:        GOTO      AA
580: AA:
581:        NOP
582:        BSF       CONT,7
583:        GOTO      TT
584: ZZ:
585:        BCF       CONT,7
586: OTROBIT:
587:        RRF       TRANS,F       //;12H =TRANS
588:        BTFSC     STATUS,0
589:        BSF       port_b,2///
590:        BTFSS     STATUS,0
591:        BCF       port_b,2///
592:        BSF       CONT,6
593:        GOTO      TT
594: RR:
595:        BCF       CONT,6
596:        DECFSZ    CONT,F
597:        GOTO      OTROBIT
598:        GOTO      AQUI
599: AQUI:
600:        NOP
601:        BSF       port_b,2///
602: TT:                      //;MOVLW   0X50
603:        MOVF      VEL_BPS,W
604:        MOVWF     REGX2
605: WTR:
606:        DECFSZ    REGX2,F
607:        GOTO      WTR
608:        GOTO      AQUI2
609: AQUI2:
610:        BTFSC     CONT,7
611:        GOTO      ZZ
612:        BTFSC     CONT,6
```

```
613:       GOTO     RR
614:       //RETURN
615: #ENDASM
616:
617: }
618:
619: void trans3(char TRANS){
620:
621: char CONT,REGX2;
622:
623:
624: #ASM
625:       BSF      STATUS,5
626:       BCF      port_b,3 ///
627:       BCF      STATUS,5
628:
629:       BCF      port_b,3 ///
630:       MOVLW    8
631:       MOVWF    CONT   //;0D =CONTA1
632:       GOTO     AA
633: AA:
634:       NOP
635:       BSF      CONT,7
636:       GOTO     TT
637: ZZ:
638:       BCF      CONT,7
639: OTROBIT:
640:       RRF      TRANS,F      //;12H =TRANS
641:       BTFSC    STATUS,0
642:       BSF      port_b,3///
643:       BTFSS    STATUS,0
644:       BCF      port_b,3///
645:       BSF      CONT,6
646:       GOTO     TT
647: RR:
648:       BCF      CONT,6
649:       DECFSZ   CONT,F
650:       GOTO     OTROBIT
651:       GOTO     AQUI
652: AQUI:
653:       NOP
654:       BSF      port_b,3///
655: TT:                  //;MOVLW   0X50
656:       MOVF     VEL_BPS,W
657:       MOVWF    REGX2
658: WTR:
659:       DECFSZ   REGX2,F
660:       GOTO     WTR
661:       GOTO     AQUI2
662: AQUI2:
663:       BTFSC    CONT,7
664:       GOTO     ZZ
665:       BTFSC    CONT,6
666:       GOTO     RR
667:       //RETURN
668: #ENDASM
669:
670: }
671:
672: void trans4(char TRANS){
673:
674: char CONT,REGX2;
675:
676:
677: #ASM
678:       BSF      STATUS,5
679:       BCF      port_b,4 ///
680:       BCF      STATUS,5
```

```
681:
682:        BCF       port_b,4 ///
683:        MOVLW     8
684:        MOVWF     CONT    //;0D =CONTA1
685:        GOTO      AA
686: AA:
687:        NOP
688:        BSF       CONT,7
689:        GOTO      TT
690: ZZ:
691:        BCF       CONT,7
692: OTROBIT:
693:        RRF       TRANS,F       //;12H =TRANS
694:        BTFSC     STATUS,0
695:        BSF       port_b,4///
696:        BTFSS     STATUS,0
697:        BCF       port_b,4///
698:        BSF       CONT,6
699:        GOTO      TT
700: RR:
701:        BCF       CONT,6
702:        DECFSZ    CONT,F
703:        GOTO      OTROBIT
704:        GOTO      AQUI
705: AQUI:
706:        NOP
707:        BSF       port_b,4///
708: TT:                    //;MOVLW    0X50
709:        MOVF      VEL_BPS,W
710:        MOVWF     REGX2
711: WTR:
712:        DECFSZ    REGX2,F
713:        GOTO      WTR
714:        GOTO      AQUI2
715: AQUI2:
716:        BTFSC     CONT,7
717:        GOTO      ZZ
718:        BTFSC     CONT,6
719:        GOTO      RR
720:        //RETURN
721: #ENDASM
722:
723: }
724:
725: void trans5(char TRANS){
726:
727: char CONT,REGX2;
728:
729:
730: #ASM
731:        BSF       STATUS,5
732:        BCF       port_b,5 ///
733:        BCF       STATUS,5
734:
735:        BCF       port_b,5 ///
736:        MOVLW     8
737:        MOVWF     CONT    //;0D =CONTA1
738:        GOTO      AA
739: AA:
740:        NOP
741:        BSF       CONT,7
742:        GOTO      TT
743: ZZ:
744:        BCF       CONT,7
745: OTROBIT:
746:        RRF       TRANS,F       //;12H =TRANS
747:        BTFSC     STATUS,0
748:        BSF       port_b,5///
```

```
749:        BTFSS     STATUS,0
750:        BCF       port_b,5///
751:        BSF       CONT,6
752:        GOTO      TT
753: RR:
754:        BCF       CONT,6
755:        DECFSZ    CONT,F
756:        GOTO      OTROBIT
757:        GOTO      AQUI
758: AQUI:
759:        NOP
760:        BSF       port_b,5///
761: TT:                    //;MOVLW   0X50
762:        MOVF      VEL_BPS,W
763:        MOVWF     REGX2
764: WTR:
765:        DECFSZ    REGX2,F
766:        GOTO      WTR
767:        GOTO      AQUI2
768: AQUI2:
769:        BTFSC     CONT,7
770:        GOTO      ZZ
771:        BTFSC     CONT,6
772:        GOTO      RR
773:        //RETURN
774: #ENDASM
775:
776: }
777:
778: void trans6(char TRANS){
779:
780: char CONT,REGX2;
781:
782:
783: #ASM
784:        BSF       STATUS,5
785:        BCF       port_b,6 ///
786:        BCF       STATUS,5
787:
788:        BCF       port_b,6 ///
789:        MOVLW     8
790:        MOVWF     CONT    //;0D =CONTA1
791:        GOTO      AA
792: AA:
793:        NOP
794:        BSF       CONT,7
795:        GOTO      TT
796: ZZ:
797:        BCF       CONT,7
798: OTROBIT:
799:        RRF       TRANS,F       //;12H =TRANS
800:        BTFSC     STATUS,0
801:        BSF       port_b,6///
802:        BTFSS     STATUS,0
803:        BCF       port_b,6///
804:        BSF       CONT,6
805:        GOTO      TT
806: RR:
807:        BCF       CONT,6
808:        DECFSZ    CONT,F
809:        GOTO      OTROBIT
810:        GOTO      AQUI
811: AQUI:
812:        NOP
813:        BSF       port_b,6///
814: TT:                    //;MOVLW   0X50
815:        MOVF      VEL_BPS,W
816:        MOVWF     REGX2
```

```
817: WTR:
818:       DECFSZ   REGX2,F
819:       GOTO     WTR
820:       GOTO     AQUI2
821: AQUI2:
822:       BTFSC    CONT,7
823:       GOTO     ZZ
824:       BTFSC    CONT,6
825:       GOTO     RR
826:       //RETURN
827: #ENDASM
828:
829: }
830: ////////////////////////////////////////////////////////////////////////
831: //
832: // TRASMISCION A LA PC
833: //
834: ////////////////////////////////////////////////////////////////////////
835: void trans7(char TRANS){
836:
837: char CONT,REGX2;
838:
839:
840: #ASM
841:       BSF      STATUS,5
842:       BCF      port_c,6 ///
843:       BCF      STATUS,5
844:
845:       BCF      port_c,6 ///
846:       MOVLW    8
847:       MOVWF    CONT   //;0D =CONTA1
848:       GOTO     AA
849: AA:
850:       NOP
851:       BSF      CONT,7
852:       GOTO     TT
853: ZZ:
854:       BCF      CONT,7
855: OTROBIT:
856:       RRF      TRANS,F      //;12H =TRANS
857:       BTFSC    STATUS,0
858:       BSF      port_c,6///
859:       BTFSS    STATUS,0
860:       BCF      port_c,6///
861:       BSF      CONT,6
862:       GOTO     TT
863: RR:
864:       BCF      CONT,6
865:       DECFSZ   CONT,F
866:       GOTO     OTROBIT
867:       GOTO     AQUI
868: AQUI:
869:       NOP
870:       BSF      port_c,6///
871: TT:                 //;MOVLW   0X50
872:       MOVF     VEL_BPS,W
873:       MOVWF    REGX2
874: WTR:
875:       DECFSZ   REGX2,F
876:       GOTO     WTR
877:       GOTO     AQUI2
878: AQUI2:
879:       BTFSC    CONT,7
880:       GOTO     ZZ
881:       BTFSC    CONT,6
882:       GOTO     RR
883:       //RETURN
884: #ENDASM
```

```
885:
886: }
887: char recep7(){
888:
889: char RECEP,CONTA_REC;
890:
891: #ASM
892:
893:       BSF      STATUS,5
894:       BSF      port_c,7
895:       BCF      STATUS,5
896:
897: et5:
898:       BTFSC    port_c,7
899:       GOTO     et5
900:       MOVLW    8
901:       MOVWF    CONTA_REC
902:       CLRF     RECEP
903:       BSF      CONTA_REC,7
904:       GOTO     et6
905: et3:
906:       BCF      CONTA_REC,7
907:       GOTO     et6
908: et1:
909:       BCF      STATUS,0
910:       BTFSC    port_c,7
911:       BSF      STATUS,0
912:       RRF      RECEP,F
913:       BSF      CONTA_REC,6
914:       GOTO     et6
915: et2:
916:       BCF      CONTA_REC,6
917:       DECFSZ   CONTA_REC,F
918:       GOTO     et1
919:       MOVF     RECEP,W
920:       MOVWF    TEMP
921:       goto     finx
922:
923: et6:
924:       MOVF     VEL_BPS,W //MOVLW  0X25;0XA7 9600BPS
925:       BTFSC    CONTA_REC,7
926:       MOVF     VEL_BPS2,W //MOVLW  0X0D   ;2DH 9600BPS
927:       MOVWF    TEMP
928: et4:
929:       DECFSZ   TEMP,F
930:       GOTO     et4
931:       NOP
932:       BTFSC    CONTA_REC,7
933:       GOTO     et3
934:       BTFSC    CONTA_REC,6
935:       GOTO     et2
936:       GOTO     et1
937: finx:
938:       NOP
939:   #ENDASM
940:
941: return RECEP;
942: }
943:
944:
945: ////////////////////////////////////////////////////////////////////////
946: //
947: //POCESOS
948: //
949: ////////////////////////////////////////////////////////////////////////
950:
951: char con0(){
952:
```

```
 953:
 954: ////////////////////////////////////////////////////////////////////
 955: //TRAMA UNO
 956: ////////////////////////////////////////////////////////////////////
 957:
 958: for(contt=0;contt<5;contt++){    trans0(TRAMA1[contt]);    }
 959: for(contt=0;contt<6;contt++){    if(TRAMA2[contt]!=recep0()){ error++
 960:
 961: trans7('v');trans7('e');trans7('r');trans7('i');trans7('f');trans7('i
 962: trans7('c');trans7('a');trans7('d');trans7('o');trans7(0x0a);
 963: test(1);
 964: ////////////////////////////////////////////////////////////////////
 965: //TRAMA DOS
 966: ////////////////////////////////////////////////////////////////////
 967:
 968: for(contt=5;contt<11;contt++){      trans0(TRAMA1[contt]);  }
 969: for(contt=6;contt<12;contt++){    if(TRAMA2[contt]!=recep0()){ error+
 970:
 971: trans7('C');trans7('o');trans7('n');trans7('f');trans7('i');trans7('g
 972: trans7('u');trans7('a');trans7('d');trans7('o');trans7(0x0a);
 973: test(1);
 974: ////////////////////////////////////////////////////////////////////
 975: //TRAMA TRES
 976: ////////////////////////////////////////////////////////////////////
 977: VEL_BPS=0X24;         //VEL_BPS=0X25;
 978: VEL_BPS2=0x0C;                    //VEL_BPS2=0x0D;
 979:
 980: for(contt=11;contt<16;contt++){      trans0(TRAMA1[contt]);  }
 981: for(contt=12;contt<19;contt++){    if(TRAMA2[contt]!=recep0()){ error
 982: trans7('C');trans7('o');trans7('n');trans7('e');trans7('c');trans7('t
 983: trans7('a');trans7('d');trans7('0');trans7(0x0a);
 984: test(1);
 985: return error;
 986:
 987: }
 988:
 989: char des0(){
 990:
 991:
 992:
 993: for(contt=16;contt<21;contt++){      trans0(TRAMA1[contt]);  }
 994: for(contt=19;contt<25;contt++){    if(TRAMA2[contt]!=recep0()){ error
 995:
 996: trans7('D');trans7('e');trans7('s');trans7('c');trans7('o');trans7('n
 997: trans7('e');trans7('c');trans7('t');trans7('a');trans7('d');trans7('o
 998:
 999: test(1);
1000: return error;
1001:
1002: }
1003:
1004:
1005: void leer(){
1006:
1007:
1008:
1009: for(contt=21;contt<27;contt++){      trans0(TRAMA1[contt]);  }
1010:
1011: if(0xe5==recep0()){
1012:    //con tag
1013:
1014:    recep0();
1015:
1016:
1017:    recep0();
1018:    recep0();
1019:    recep0();
1020:
```

```
1021:    tag[0]=recep0();
1022:    tag[1]=recep0();
1023:    tag[2]=recep0();
1024:    tag[3]=recep0();
1025:    tag[4]=recep0();
1026:    tag[5]=recep0();
1027:    tag[6]=recep0();
1028:    tag[7]=recep0();
1029:    tag[8]=recep0();
1030:    tag[9]=recep0();
1031:    tag[10]=recep0();
1032:    tag[11]=recep0();
1033:    recep0();
1034: }
1035:
1036: else{
1037:    recep0();
1038:    recep0();
1039:    recep0();
1040:    recep0();
1041:    recep0();
1042: }
1043:
1044:
1045: }
1046:
1047: void trasmitir(){
1048:
1049: int t=0;
1050:
1051: for(contt=0;contt<12;contt++){
1052:
1053:    trans7(tag[contt]);
1054:
1055:    if(t==12){trans7(0x0a); t=0;}
1056:
1057:    else{t++;}
1058:
1059:    }
1060:
1061: }
1062:
1063:
1064:
1065: void main(){
1066:
1067: int ciclos=70;
1068: ////////////////////////////Configuracionde puertos
1069: set_tris_b(0);//Salida
1070: port_b =0xff;  //1111 1111
1071: port_c =0x40; //1000 0000
1072: set_tris_d(1); //Entrada
1073: port_d =0x00;  //0000 0000
1074: VEL_BPS=0XA5;
1075: VEL_BPS2=0x2D;
1076: ////////////////////////////////////////////////////
1077:    setup_adc_ports(NO_ANALOGS);
1078:    setup_adc(ADC_OFF);
1079:    setup_psp(PSP_DISABLED);
1080:    setup_spi(FALSE);
1081: //   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
1082: //   setup_timer_1(T1_DISABLED);
1083:   // setup_timer_2(T2_DISABLED,0,1);
1084:    setup_comparator(NC_NC_NC_NC);
1085:    setup_vref(FALSE);
1086:    setup_timer_1 ( T1_INTERNAL | T1_DIV_BY_1 );
1087:    enable_interrupts(INT_TIMER1);
1088:    enable_interrupts(GLOBAL);
```

```
1089:
1090:
1091:
1092: ////////////////////CONECTAR////////////////////////////////
1093: test(4);
1094: error=con0();
1095:
1096: ////////////////////TRASMITIR////////////////////////////////
1097: trans7('L');trans7('e');trans7('y');trans7('e');trans7('n');trans7('d
1098: trans7('o');trans7(0x0a);
1099:
1100:
1101: #asm
1102:    bsF   STATUS,5
1103:    BCF   port_a,0
1104:    BCF   STATUS,5
1105: #endasm
1106:
1107: while(1){
1108:
1109: des=0;
1110: for(ciclos;0<ciclos;ciclos--){
1111: leer();
1112:
1113: //test(1);
1114: trasmitir();
1115:
1116: }
1117:
1118: #asm
1119:    movlw 0xff
1120:    xorwf port_a
1121: #endasm
1122: trans7(des);
1123:
1124: ciclos=70;
1125: }
1126: ////////////////////DESCONECTAR////////////////////////////////
1127:
1128: test(2);
1129: error=des0();
1130:
1131:
1132: }
1133:
```