UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE: GUAYAQUIL
CARRERA DE: MECATRÓNICA

TEMA: IMPLEMENTACIÓN DE UN SISTEMA EMBEBIDO DE BAJO COSTO PARA LA ASIGNACIÓN DE TURNOS CON ALMACENAMIENTO MASIVO DE DATOS

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Mecatrónica

AUTORES: Milton Genaro Benavides Duran
Alberth Marcel Hurtado Celi
TUTOR: Franklin Illich Kuonquí Gaínza

Guayaquil-Ecuador
2022

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

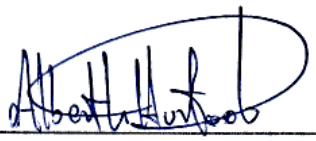
Nosotros, **Milton Genaro Benavides Duran** con documento de identificación N° **0703583716** y **Alberth Marcel Hurtado Celi** con documento de identificación N° **0704434265**; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo.

Guayaquil, 15 de septiembre del año 2022

Atentamente,

Milton Genaro Benavides Duran
0703583716



Alberth Marcel Hurtado Celi
0704434265

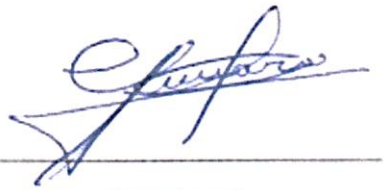
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, **Milton Genaro Benavides Duran** con documento de identificación N° **0703583716** y **Alberth Marcel Hurtado** con documento de identificación N° **0704434265**, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del **Dispositivo Tecnológico: IMPLEMENTACIÓN DE UN SISTEMA EMBEBIDO DE BAJO COSTO PARA LA ASIGNACIÓN DE TURNOS CON ALMACENAMIENTO MASIVO DE DATOS**, el cual ha sido desarrollado para optar por el título de: Ingeniero en Mecatrónica, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo a final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana

Guayaquil, 15 de septiembre del año 2022

Atentamente,



Milton Genaro Benavides Duran
0703583716

Alberth Marcel Hurtado Celi
0704434265

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, **Franklin Illich Kuonquí Gaínza**, docente de la Universidad Politécnica Salesiana , declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **IMPLEMENTACIÓN DE UN SISTEMA EMBEBIDO DE BAJO COSTO PARA LA ASIGNACIÓN DE TURNOS CON ALMACENAMIENTO MASIVO DE DATOS**, realizado por **Milton Genaro Benavides Duran** con documento de identificación N° **0703583716** y por **Alberth Marcel Hurtado Celi** con documento de identificación N° **0704434265**, obteniendo como resultado final el trabajo de titulación bajo la opción **Dispositivo Tecnológico** que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 15 de septiembre del año 2022

Atentamente,

Ing. Franklin Illich Kuonquí Gaínza, Mg.
0909627432

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

DEDICATORIA

La presente tesis está dedicada a:
A mis padres Tania Celi y Alver Hurtado los cuales con su amor, esfuerzo y dedicación me han permitido forjarme como profesional, gracias por inculcar en mí el ejemplo del trabajo y honestidad, de no temer a las adversidades porque ellos siempre estarán a mi lado.
A mi hermana Jhosvet por su cariño y apoyo incondicional, durante toda la etapa de mi instrucción profesional, por estar conmigo en todo momento gracias.
Finalmente, quiero dedicar esta tesis a mis profesores que con dedicación y esmero fueron capaces de difundir sus conocimientos en mí.

Alberth Marcel Hurtado Celi

A mis queridos padres, a mis adorados hijos que constituyen la fuerza y razón que me impulsan a seguir adelante para hacer realizar mis objetivos.

Milton Genaro Benavides Duran

AGRADECIMIENTO

Al término este trabajo quiero utilizar este espacio para agradecer, a mis Padres que durante todo este tiempo me apoyaron incondicionalmente para estudiar la carrera de Ing. Mecatrónica.
También quiero agradecer a la Universidad Politécnica Salesiana, directivos y profesores por ser parte de este logro que desde sus conocimientos supieron instruirme adecuadamente.

Alberth Marcel Hurtado Celi

A mis padres Milton Benavides y Cecilia Duran, por su ayuda incondicional, para que cumpla una meta más en mi vida y logro profesional.
Gracias a mi docente Ing. Franklin Kuonquí por todos sus conocimientos y experiencia que me han aportado grandemente en mi proyecto.
Gracias a Dios por darme todos los días la bendición de estar con salud para cumplir mis objetivos y darme el regalo de tener a todos ustedes.
Mi agradecimiento especial a la Universidad Politécnica Salesiana, por abrirme sus puertas para formarme profesionalmente.

Milton Genaro Benavides Duran

I. Resumen

En la actualidad para las empresas, la competencia no solo se basa en un menor precio y gran variedad de artículos, el mercado también responde al servicio con que es atendido el consumidor, desde el marketing hasta las opiniones vertidas en la web que sirven como la tarjeta de presentación hacia los clientes quienes buscan compañías con excelente imagen publica a la hora de considerar donde y cuando comprar los artículos. Por esta razón cuando las empresas empiezan a buscar sistemas que automaticen la atención del cliente, encuentra proformas que en muchos casos representan altas inversiones de dinero y que no se pueden permitir, por esta razón el objetivo general fue el de Desarrollar un sistema de asignación de turnos, con manejo de prioridades en la atención y Big Data, para una distribución eficiente del tiempo de espera, de los clientes en cola, usando microcontroladores y comunicación inalámbrica y entre los objetivos específicos fueron Diseñar la etapa de hardware y software para un sistema embebido de bajo costo, para la asignación de turnos, con almacenamiento masivo de datos que se encarga del manejo de los datos recabados durante la jornada de trabajo. La propuesta de solución busca brindar tanto los componentes que se tomaron en cuanta como el tipo de comunicación que se usara en el sistema y los resultados obtenidos luego de ser implementado en una Cooperativa de taxis.

II. ABSTRAC

Currently, the competition for businesses is not only based on lower prices and great variety of products. The market also responds to the service with which the consumer is tend to. From marketing to opinions poured on the web that serve as a greeting card for clients looking for an excellent public image when it comes to when and where to buy their products. For this reason, when businesses search for systems to automatize costumer service, they find price quote that in most cases represent high investments they cannot allow themselves to make. For this reason, the general objective was to develop an assigning system of turns, with management of priority of attention and Big Data, for an efficient distribution of waiting time of costumers in line, by using micro controllers and wireless communication. Among the specific objectives, it came down to a phase of designing a hardware and software for an immersed system of low cost, for the assignment of turns, with massive storage, tasked with the management of collected data during the working day. The proposed solution looks to bring both components, which were taken into consideration as the type of communication used for the system, and the results obtained after being implemented in a Taxi union.

III. INTRODUCCIÓN

Las empresas se han visto afectadas por la pandemia reciente del COVID-19, por lo cual se exigió el cierre de locales comerciales durante 2 años con la finalidad de reducir la propagación de la enfermedad, y por lo cual la economía ecuatoriana decayó drásticamente por la falta de clientes, quienes se encargaban del crecimiento propio de la empresa y por ello las empresas tuvieron que mejorar sus servicios de atención virtual con la finalidad de mantener a los clientes que ya tenían en la nómina.

En la actualidad para las empresas, la competencia no solo se basa en un menor precio y gran variedad de artículos, el mercado también responde al servicio con que es atendido el consumidor, desde el marketing hasta las opiniones vertidas en la web que sirven como la tarjeta de presentación hacia los clientes quienes buscan compañías con excelente imagen publica a la hora de considerar donde y cuando comprar los artículos.

Por esta razón, cuando las empresas empiezan a buscar sistemas que automaticen la atención del cliente, encuentra proformas que en muchos casos representan altas inversiones de dinero y que no se pueden permitir, recordemos que la atención al cliente es un servicio no remunerado y por lo cual no se puede considerar hasta que punto se pueda recuperar la inversión realizada.

Con el objetivo de proporcionar un servicio de calidad y sin incurrir en grandes gastos administrativos, desarrollar un sistema de asignación de turnos de bajo costo, con manejo de prioridades en la atención y con el uso apropiado del Big Data con la finalidad de mejorar el servicio.

Por esta razón la investigación para el diseño del sistema se efectuó una vez conocido los requerimientos más comunes de las empresas, las diferencias y limitaciones de los sistemas que ya existen en el mercado, con finalidad de fabricar un prototipo que satisfaga la mayoría de la necesidad y sin incurrir en una inversión muy alta.

En este sentido, la investigación propone el uso de display de 7 segmentos para la visualización del turno, una impresora térmica con conexión de puerto paralelo para recibir comunicación por parte de la tarjeta que la controladora, por ello, la investigación está estructurada de la siguiente manera: El problema, sus antecedentes, importancia, alcance, justificación, objetivo general y específicos. Un marco teórico referencial, propuesta de solución, marco metodológico, diseño, implementación, resultados, cronograma, presupuesto, conclusiones y recomendaciones.

IV. PROBLEMATICA

En la actualidad las empresas desestiman el servicio al cliente, ignorando los beneficios que estos aportarían. A diferencia de la ingeniería, los departamentos de ventas o marketing son los encargados de mantener la nómina de clientes fieles a la empresa, procurando, entre otros aspectos, que los servicios de atención actúen de manera ágil y apropiada [16].

Un claro ejemplo es Jeff Bezos, CEO (Chief Executive Officer) de Amazon, quien, tuvo la visión de usar los datos en "bruto" de las visitas y compras realizadas en su página, para obtener información de los hábitos de compras de sus usuarios. Adicionalmente, con esta información la compañía es capaz de predecir los requerimientos de compra futura de sus clientes, ofreciéndoles una experiencia personalizada que se traduce en un mayor nivel de satisfacción [3].

No importa en qué tipo de sector productivo se dedique una empresa, probablemente se enfrenta a una gran cantidad de competidores que ofrecen productos o servicios muy similares y ante tal situación, es mejor aprovechar cualquier opción para destacar y crecer sanamente.

Una de las opciones es ofrecer un servicio excepcional al cliente. Para ello, es esencial conocer las necesidades de los usuarios, con el fin de darles una atención más agilizada, dado que las personas o consumidores están dispuestas a mantener la fidelidad a la empresa que les brinda un servicio rápido y eficiente [3].

El servicio al cliente es un factor muy importante para que un consumidor decida mantenerse con el mismo proveedor. De acuerdo con las investigaciones, se ha encontrado que hasta el 89 % de los clientes podrían cambiar de proveedor de servicios después de una mala experiencia [15] .

En la mayoría de las veces, cuando un cliente busca acercarse a la ventanilla, tiene que realizar largas horas de colas para poder ser atendido por un ejecutivo. Sin embargo, hay ocasiones en que los asesores requieren transferir al consumidor hacia algún personal especializado, que pueda manejar mejor la consulta. Para evitar la pérdida de clientes debido a un servicio de atención lento o deficiente, es posible utilizar la Big Data para acelerar dicho proceso [7].

Desde el punto de vista financiero, un sistema de turnos que permita al público, escoger entre los diferentes tipos de atención que ofrece la institución, con impresión del ticket que incluye el turno, tiempo de espera aproximado y una pantalla que permita confirmar los turnos que son llamados por los operadores, representa un costo alto dentro del mercado ecuatoriano. Con sistemas que oscilan entre los $5000 y $8000 dólares estadounidenses.

No es de extrañar, que las pequeñas sucursales que pertenecen a la institución, no se puedan permitir adquirir este tipo de sistemas, además, que aquellas que puedan permitirse la compra de estas herramientas, mantendrán altos costos de gastos y mantenimiento.

Considerando los argumentos presentados, cabe recalcar que las empresas gastan miles, si no es que millones de dólares, en publicidad con la finalidad de conseguir nuevos consumidores, siendo 5 veces más costoso conseguir clientes nuevos que mantener a los que se encuentran en la nómina, mientras que se generan veinticinco clientes nuevos por cada cien satisfechos [13].

V. JUSTIFICACIÓN

Un estudio del Banco Mundial indica que el 49 % de los puestos de trabajo de Ecuador serán susceptibles a ser reemplazados por la automatización en las fábricas; sin embargo, no se detalla el periodo de tiempo cuando ocurrirá [17].

Un cliente satisfecho transmite su experiencia a otros tres como mínimo, siendo el marketing boca a boca la publicidad más confiable, al obtener la información de una persona que el receptor conoce, la cual observa el mensaje de manera no publicitaria [1].

En busca de la mejora en la atención al cliente, la interfaz debe ser amigable al usuario para su fácil manejo, por ello, el sistema cuenta con una interfaz, que mostrara los datos recogidos durante la jornada de trabajo, y presentado en una pantalla de siete segmentos de color rojo para su fácil manejo.

Con los datos obtenidos, el personal encargado de organizar las horas de trabajo podrá asignar de manera eficiente, el personal requerido para las horas de mayor multitud, además que el sistema podrá redirigir al público para cada operación especifica que desea realizar dentro de la institución.

Para ello, mediante el uso de un despachador de ticket para atención al cliente, que guarde los registros de tráfico de clientes, por hora, agencia, día y mes, ayudara a conocer, donde se necesita aumentar el número de asesores y las horas en las que se debe de trabajar, con la finalidad de reducir los tiempos espera y aglomeraciones.

Mediante el uso de la Big Data, recogida durante los días de trabajo, proporcionan datos fidedignos para los organizadores de los turnos de trabajo, además, de distribuir de manera eficiente el personal a las diferencias agencias sin la necesidad de contratar más trabajadores [3].

Por esta razón, nuestro sistema ofrece las mismas prestaciones que ofrecen los competidores existentes, por el valor más bajo del mercado de $1800 dólares estadounidenses, lo cual permitirá que no solo las grandes agencias sino también las pequeñas, puedan adquirir este servicio.

Como estudiantes de la carrera de Ingeniería Mecatrónica de la universidad Politécnica Salesiana, con sede en Guayaquil, es nuestro deber ofrecer sistemas acordes a las necesidades del mercado, que para este proyecto estaremos haciendo uso de los conocimientos aprendidos en el aula, como son las materias de, Programación, Electrónica, Circuitos eléctricos, Proyectos, Probabilidad y Estadística, entre otros.

VI. Objetivos

VI-A. Objetivo general

- Desarrollar un sistema de asignación de turnos, con manejo de prioridades en la atención y big data, para una distribución eficiente del tiempo de espera, de los clientes en cola, usando microcontroladores y comunicación inalámbrica.

VI-B. Objetivos específicos

- Explorar los sistemas semejantes existentes en el mercado, para la identificación de las características ofrecidas actualmente por otros proveedores
- Realizar un estudio, de tiempo de espera que realizan los clientes, antes de ser atendido por un asesor.
- Diseñar la etapa de hardware para un sistema embebido de bajo costo, para la asignación de turnos, con almacenamiento masivo de datos.
- Diseñar la etapa de software, que se encarga del manejo de los datos recabados durante la jornada de trabajo.

VII. MARCO TEÓRICO.

La automatización es un concepto que suele utilizarse en el ámbito empresarial, con referencia al sistema que permite que una máquina desarrolle ciertos procesos o realice tareas sin intervención del ser humano. Es empleada con frecuencia por su capacidad de ahorrar tiempo y dinero, al tener resultados más homogéneos al no depender del criterio del operario.

VII-A. Altium

Altium Designer es un software usado para el diseño de placas electrónicas, ya sea en sus fases de diseño de circuitos impresos y simulaciones, además, de contar con la implementación de FPGA, o desarrollo de código para microprocesadores [2].

Adicionalmente, permite diseñar la tarjeta electrónica para el sistema de turnos, ya que, al contar con todos los componentes electrónicos existentes en el mercado, se pueden realizar simulaciones y comprobar su funcionalidad, sin presentar costo alguno.

VII-B. Capacitor

La principal característica de los capacitores es la de almacenar energía, ya que al estar fabricados por un dieléctrico que puede estar fabricado de cerámica, aire, vidrio, nylar o papel, que se encuentra entre dos placas metálicas, puede almacenar grandes o pequeñas cargas, siendo su unidad de trabajo el farad (uf) [9].

Debido a su alto nivel de estabilidad y baja temperatura de trabajo, son usados en los circuitos electrónicos en diferentes aplicaciones, ya sea, de acoplo, paso, de filtro, etc., además al trabajar junto con inductores y resistores, obtienen características especiales como circuitos sintonizados o de filtro.

VII-C. Diodo Schottky

A diferencias de los diodos comunes con características semiconductoras, ya que la corriente circula a través de él, pero en un solo sentido, el diodo schottky puede trabajar de varias maneras, ya sea, No polarizado, polarizado de manera directa o inversa, por lo cual, es usado en aplicaciones de radiofrecuencia (RF) [12].

Al tener una caída de tensión directa más baja que el diodo PN, y puede usarse en aplicaciones de conmutación a alta velocidad, debido a que, el diodo PN mantiene una tensión directa de 0.6 a 0.75 voltios (V), mientras que el Schottky es de 0.15 a 0.45V, dando como resultado una conmutación más rápida [12].

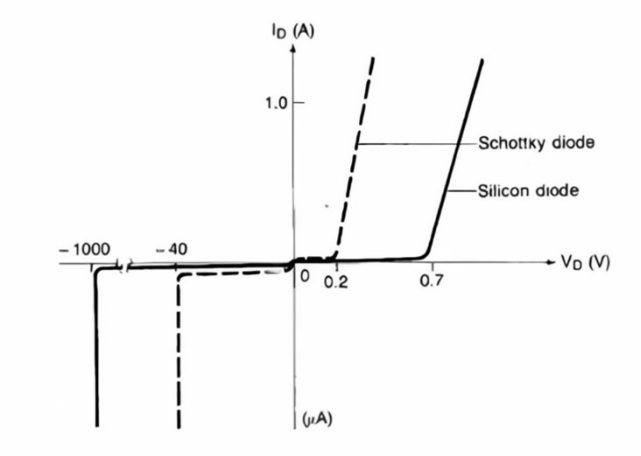


Figura 1. Curva de operación de diodo schottky

VII-D. Diodo TVS

Los diodos TVS son fabricados como la finalidad de proteger los componentes electrónicos, ante los posibles picos de voltaje que se presentan, ya que, al contar con gran sensibilidad y rápida respuesta en comparación a otros dispositivos electrónicos de protección, además de ofrecer diferentes tipos de montaje para las placas PCB [20].

El funcionamiento del diodo tvs es de limitar en cierta medida el voltaje al cual llamaremos sujeción, dicho voltaje pasa por el área del diodo desde sus conexiones p-n de manera transversal, y al ser está más amplia que un diodo normal le permite conducir grandes cantidades de corrientes sin sufrir daño alguno [20].

Debido a las características antes mencionadas, los diodos tvs fueron diseñados para proteger las interfaces de E/S, por lo cual son usados en equipos industriales, computadora, productos electrónicos y telecomunicaciones, por aquello, serán usados en la tarjeta electrónica del presente proyecto tutorial, con la finalidad de proteger el dispositivo.

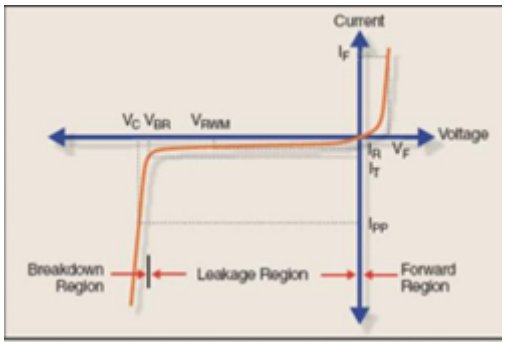


Figura 2. Curva de operación de diodo TVS

VII-E. Indicadores Fijos

Es un componente eléctrico pasivo, que almacena energía en su campo magnético debido a la corriente eléctrica que fluye a través de él, el inductor suele consistir, en cable de cobre aislado que se enrolla por el núcleo formando

la bobina, el núcleo puede ser de hierro o ferrita [8].

Al ser un inductor que se caracteriza por la relación entre la tensión y la tasa de variación de la corriente, gracias al material del núcleo, le permite aumentar el campo magnético que se traduce en mayor inductancia. Siendo uno de los tres elementos pasivos lineales que se usan para los circuitos electrónicos de corriente alterna (CA)[8].

Figura 3. Indicador Fijo

VII-F. PCB

Es una superficie formada por pistas de material conductor (Cobre, Plata, Oro) laminado sobre una placa no conductora. Los circuitos impresos se utilizan para conectar eléctricamente a través de pistas conductoras los componentes electrónicos, las cuales se encargan de transmitir las señales eléctricas. Adicionalmente, la placa suele estar hecha de plástico reforzado con fibra de vidrio, cerámica, plástico, teflón o un polímero como la baquelita [14].

VII-G. Microcontroladores

El Microcontrolador es un circuito integrado, siendo el componente principal de una aplicación embebida. Es como una pequeña computadora (figura 4) que incluye sistemas para controlar elementos de entrada/salida. También incluye a un procesador y por supuesto memoria que guarda el programa y sus variables (flash y RAM). Funciona como una mini PC. Su función es la de automatizar procesos y manipular información [21].

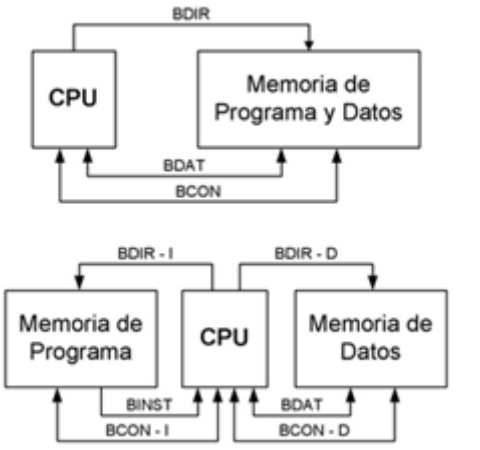


Figura 4. Arquitecturas del microcontrolador

VII-H. PIC18F67J50-I

El PIC18F67J50-I / PT es un microcontrolador USB flash de alto rendimiento de 64 pines con tecnología nano-Watt. Ideal para aplicaciones de baja potencia (nanoWatt) y conectividad que se benefician de la disponibilidad de cuatro puertos seriales FS-USB (12Mbps), MI²C ™ y SPI ™ (hasta 10Mbps) y un puerto asíncrono (con capacidad LIN) de dos puertos seriales (EUSART). Presenta una memoria RAM de alto contenido para almacenamiento en búfer, la memoria flash mejorada del programa, haciéndolo ideal para aplicaciones de control y monitoreo integradas, que requieren una conexión periódica con una computadora personal (gratuita) a través de USB para cargar / descargar datos y / o actualizaciones de firmware[21].

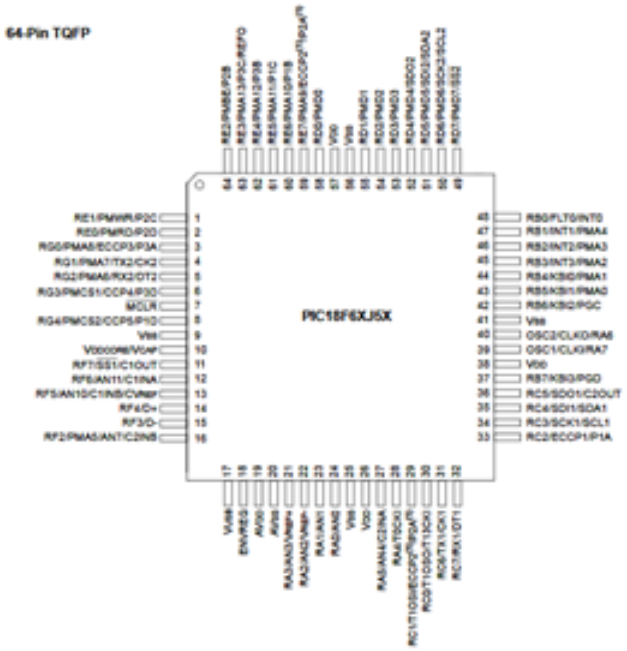


Figura 5. Diagrama del PIC18F67J50

Características generales:

- Tipo de encapsulado: 64-TQFP.
- Voltaje de operación de 2V a 3.6 V.
- Velocidad de operación: 0 a 20 MHz.
- Cuatro puertos I/O de 8 bits, dos puertos I/O de 6 bits y un puerto adicional I/O de 5 bits.
- Oscilador interno.
- Rango de temperatura de -40°C a +85°C.

VII-I. Resistores de película gruesa

Los resistores de película gruesa consisten en una película resistiva, generalmente en el orden de las 10 micras de espesor, que se ha depositado sobre una base de cerámica. Son económicos de producir y de uso más común en la actualidad. La composición de la película es un área activa de desarrollo para los fabricantes [8]
.

Las resistencias de película delgada también son construidas sobre una base cerámica, pero hasta ahí llegan las similitudes. El elemento resistivo es fabricado de níquel-cromo depositada en vacío sobre la base. La película resultante es unas 1000 veces más delgada que la de película gruesa. Las resistencias de película delgada son más costosas de fabricar que las resistencias de película gruesa, pero tienen una inductancia y capacitancia parásitas bajas y tienen excelentes coeficientes de temperatura. Las resistencias de película gruesa son físicamente más robustas y pueden diseñarse para transportar más corriente [8]
.

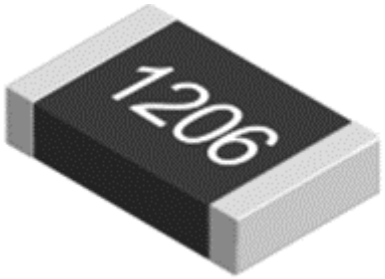


Figura 6. Resistores de pelíula gruesa

VII-J. Impresora térmica

Una impresora térmica es un equipo que utiliza calor como medio para producir la imagen o el texto en papel, por lo que no se necesita tinta o tóner. Sin embargo, sí requiere de un papel térmico que permita la impresión. Debido a la calidad de su impresión y su velocidad, se ha vuelto cada vez más popular y se emplea, principalmente, en supermercados, aerolíneas, entretenimiento, servicios de inspección técnica o atención médica.

Figura 7. Impresora térmica

LETRA	NUMERO	FECHA	HORA	ATENCION
A-Z	0-99	Actual	Actual	1-4

Figura 8. Parámetros impresos en el ticket

VII-K. Base de Datos

La adquisición de datos o adquisición de señales involucra el muestreo del mundo real (sistema analógico) para producir datos que pueden ser manipulados por computadoras u otros dispositivos digitales. El cual implica en tomar un conjunto de señales físicas, convertirlas en tensiones eléctricas y digitalizarlas para que la computadora pueda procesarlas [19].

Se requiere una etapa de acondicionamiento, para que la señal se adecue de manera compatible con los elementos que realizan la transformas en una señal digital. El elemento que realiza esta conversión es un módulo digitalizador o una tarjeta de captura de datos [19].

El sistema tendrá la información del turno de atención, también en que modulo fue atendido, la hora en que se solicitó el ticket, en que fue llamado por el asistente, el tiempo que le tomo estar con el usuario y el tipo de atención que solicito el usuario.

VII-L. Microcode Studio

.

MicroCode Studio es una herramienta de diseño integrado (IDE) que incluye un circuito para depurar errores (In circuit Debugging – ICD) que está especialmente diseñada para laboratorios de micro ingeniería, en otras palabras, es una herramienta diseñada para elaborar los textos de programación [22].

Al contar con un editor que es amigable al programador, ya que, dispone de un sistema de realce de sintaxis, ayuda contextual y esconder el código con el que no se está trabajando, para focalizarse en las pequeñas partes,

además, tener la posibilidad mediante el buscador encontrar archivos definiciones, variables y etiquetas [22].

Por lo tanto, este software nos permitirá, escribir el texto encargado de transmitir las funciones de operación, que deberá realizar cada componente electrónico conectado al sistema, además el programa nos permite ingresar la programación al PIC18F67J50, desde un quemador de microchip.

VII-M. OrCAD

OrCAD es un software usado para la automatización de diseño electrónico (EDA). El software, el cual es usado por técnicos e ingenieros de diseño, especialmente para simulación electrónica, creación de esquemas electrónicos y elaboración de circuito impresos para manufacturar placas PCB [14].

VII-N. Proteus

El programa Proteus está conformado por dos aplicaciones llamadas Ares e Isis. Isis está diseñado para realizar esquemas de circuitos con casi todos los componentes electrónicos que se encuentran actualmente disponibles en el mercado de los circuitos integrados y los componentes pasivos y activos utilizados en las aplicaciones electrónicas [18].

Pose una aplicación de simulación que permite comprobar la efectividad de un circuito determinado ante una alimentación de voltaje, este voltaje en la aplicación es virtual; también permite cargar a los microcontroladores presentes en sus librerías con los programas previamente desarrollados en los programas ensambladores y en los compiladores de Basic según sea el tipo de lenguaje elegido por el programador [18].

El programa proteos no solo nos permite diseñar la placa electrónica, también con la simulación de funcionamiento tanto de la tarjeta como de la programación, ya que, la biblioteca que contiene los diferentes componentes electrónicos, esto permite al grupo encontrar los posibles errores de fabricación y programación, antes de ser enviada a su diseño final.

VII-Ñ. Bluetooth

Bluetooth posibilita la transmisión de datos y voz entre diferentes dispositivos por radiofrecuencia, debido a que tiene incorporado un microchip llamado transceptor, que permite transmitir y recibir en la frecuencia de 2.4 GHz, permitiendo la comunicación entre varios dispositivos mientras estos se encuentren al alcance, debido a una interfaz abierta y de bajo coste [6].

La tecnología inalámbrica Bluetooth nos permite comunicar varios dispositivos digitales, para este proyecto la comunicación entre la pantalla, la impresora y la botonera de turnos, sin la necesidad de cables, Lo cual nos ahorra tiempo de trabajo en la instalación y la posibilidad de ubicar los dispositivos de la manera que mayor nos convenga, siempre y cuando no se exceda los 10 m de su rango de funcionalidad.

Figura 9. Transceptor Bluetooth

VII-O. BIG DATA

El Big Data es el conjunto de tecnologías de adquisición de datos, que se dedican a la recolección, análisis y gestión, de los datos generador por los usuarios al dar clic sobre cualquier página o aplicación. La finalidad es que estos datos en "bruto" sirvan, de manera eficiente, para el crecimiento de la empresa [3].

La ventaja de tener estos datos en "bruto", es que permiten a la empresa realizar sus propios estudios de mercado y sacar sus propias conclusiones, luego de haberlos organizado apropiadamente, debido a que contienen las preferencias de cada usuario y nos puede indicar la tendencia donde se dirige el mercado.

Todo deja huella, y aunque sean datos que son anonimizados y no te identifican como persona, sí representan lo que hace un usuario de Internet. Siendo una cantidad tan grande, incluso difícil de imaginar, y lo que hace el Big Data es utilizar estas inmensas cantidades de datos para ser analizarlos [3].

El Big Data no solo recoge los datos que se generan al navegar por Internet, también son los datos generados por las diversas aplicaciones. Por lo tanto, cualquier empresa que brinde servicios puede recopilar los datos de sus usuarios, para luego analizarlos y procesarlos [3].

VIII. Propuesta de Solución

El desarrollo del proyecto se efectuará de la siguiente manera:

VIII-A. Presentación de los dígitos en pantalla

- Identificar los dígitos del turno y módulo
- Intensidad de brillo adecuada de la pantalla para cada lugar de trabajo.
- Tiempo de reposo luego de haber realizado actividad hasta que ingrese una nueva, con el fin de ampliar el tiempo de vida de la pantalla

VIII-B. Impresora térmica

- La comunicación entre la impresora y la pantalla será de manera serial.
- Al tratarse de una impresora térmica no se necesita mantenimiento por tinta.
- Cada ticket se imprimirá con el módulo de atención, el turno, la hora, la fecha y el tiempo de espera.

VIII-C. Tiempo de respuesta

- El tiempo de espera entre escoger el turno y la impresión será de 200 milisegundos.
- El tiempo de espera se basará en la resta del tiempo de impresión y el tiempo de atención.
- El tiempo desde que se inicia la atención hasta el tiempo que vuelve a solicitar otro cliente.

VIII-D. Registro de datos

- La comunicación de los registros de datos se ejecuta de manera inalámbrica.
- Con una memoria de 512 kb se podrá registrar un máximo de 13200 usuarios por día.
- Al final de cada día, los datos (entre los cuales tenemos, tiempo de espera, tiempo de atención por cada cliente) recolectados serán enviados a una computadora específica para su posterior análisis.

VIII-E. Fabricación de la tarjeta electrónica

- El proceso de construcción digital de la PCB se la realizará en IDE gratuitos ORCAD o PROTEUS con las pruebas previas obtenidas de todos los componentes para la solución.
- El archivo de la PCB se la enviará a fabricar en www.jlcpcb.com donde ya tenemos una experiencia previa en otros proyectos de manufactura.

VIII-F. Código fuente

- Implementar el código fuente analizado en Microcode, para luego ser transformado en el lenguaje del micro-controlador.
- El código del controlador PIC 18F6722 de la familia Microchip y se ejecutará el código en Microcode studio.
- El código fuente luego para ser incorporado en el PIC usamos un programador pickit3 que soporta toda la familia 18Fxxxx.

VIII-G. Pruebas de funcionamiento y corrección de errores

- Con el prototipo funcionando, se procederá a presionar repetidas veces las diferentes opciones en la pantalla y observar si todas ellas son impresas de manera adecuada.
- Verificar si existe algún tipo conflicto entre el pulsador que tiene cada asistente de atención y el pulsador de la pantalla.
- Revisar si el orden de atención se ejecuta de manera adecuada.
- Verificar si la numeración es de manera ascendente y continua.
- Elaboración del informe final.

IX. METODOLOGÍA

El objetivo de este capítulo es identificar y especificar los procedimientos, técnicas y nivel de conocimiento, lo que permitirá investigar y desarrolla el presente proyecto en este sentido, [10] citado por [5], expone:

El marco metodológico es el conjunto de acciones destinadas a describir y analizar el fondo del problema planteado, a través de procedimientos específicos que incluye las técnicas de observación y recolección de datos, determinando el "cómo" se realizará el estudio, esta tarea consiste en hacer operativa los conceptos y elementos del problema que estudiamos. (p. 118).

Para la realización de este equipo, se llevó a cabo una investigación documental, en esta investigación de campo se tomó la versión de un distribuidor ecuatoriano de venta de equipos electrónicos para el control de atención al público, con las necesidades y requerimientos solicitadas en el mercado ecuatoriano.

En colaboración con el distribuidor, se pudo establecer las características del funcionamiento de cada modelo que él importaba y cuáles eran los más requeridos por sus clientes, dependiendo de las necesidades, estos modelos se categorizan como:

- Turno simple
- Turno Doble
- Turno simple RF
- Turno doble RF
- Turno simple RF e impresora de turno
- Turno Doble RF e impresora de turno

Para todos estos modelos, el distribuidor tenía que importar todos los accesorios debido a que el funcionamiento era único, lo cual le llevaba esperar la entrega y realizar los tediosos procedimientos de importación, que eran solicitados por la aduana ecuatoriana.

En esta investigación se elaboró un equipo que cuenta con la funcionalidad de todos los modelos convirtiéndolos en modular, ya que estará conformado por un módulo base y otros complementarios para atender los diferentes requerimientos. Esto ayudará a que el inventario en bodega sea de un solo producto, reduciendo el espacio ocupado.

Para las empresas que atienden diariamente numerosos usuarios, los sistemas de control de turnos automáticos son un elemento esencial para la satisfacción con los trámites y servicios prestados. Existen muchas soluciones en el mercado actual, sin embargo, aquellas soluciones, en términos generales, implican altos costos a las empresas y al ser sistemas genéricos, representan problemas de compatibilidad en los procesos de cada organización.

IX-A. Análisis

En esta fase se realizó una lista de las funcionalidades de cada uno de los equipos que importaba, empezando por una identificación de los diferentes requerimientos se consideró: voltaje de funcionamiento, intensidad de brillo en la pantalla, tiempo de descanso de la pantalla, distancia de enlace RF, información que presenta el ticket impreso, cantidad de turnos almacenados, manejo de turnos y módulos.

IX-A1. Requerimientos Funcionales: Los requerimientos funcionales son los atributos que debe cumplir el sistema para cada perfil que interactuara con él, así como son, los clientes que llegaran a la espera del servicio, el asesor de atención de cliente que estará a cargo de atender al usuario dependiendo del tipo de actividad que busca realizar dentro de la oficina y el administrador del personal de la compañía, el cual evaluara los datos obtenidos en los días de trabajo.

Cuadro I
REQUERIMIENTOS FUNCIONALES - PERFIL DEL CLIENTE

#	REQUERIMIENTO	DESCRIPCIÓN	USUARIO
1	El sistema debe tener diferente tipo de atención	El sistema debe permitir al usuario escoger entre diferentes tipos de atención, entre ellos (Cajero, Atención al cliente, Apertura de cuenta, Reclamos)	Usuario
2	Impresora Térmica	El sistema debe contar con una impresora térmica, la cual se encargará de la impresión del turno de atención, que será recogida por el cliente a la espera de ser llamado	Usuario
3	El sistema debe de imprimir el ticket de atención con la información	El sistema debe de imprimir el ticket, con el numero por el cual será llamado, la letra que especificará el tipo de atención que se dará, el tiempo de espera estimado para la atención, la hora y la fecha	Usuario
4	Pantalla de visualización	El sistema debe tener una pantalla que permita al usuario conocer, el turno de atención actual y el casillero por el cual será atendido	Usuario
5	Alerta auditiva	El sistema deberá tener una campana que permita conocer al usuario en el momento que el personal de servicio, llame a un nuevo turno para su atención.	Usuario

Cuadro II
REQUERIMIENTOS FUNCIONALES - PERFIL DEL PERSONAL DE ATENCIÓN DEL CLIENTE

#	REQUERIMIENTO	DESCRIPCIÓN	USUARIO
1	Control	Cada módulo de atención tiene un control RF 433 que permite llamar al cliente, además el personal podrá acceder a los cuatro diferentes tipos de atención, en el caso de que uno se encuentre colapsado y necesite de una ventanilla adicional para la atención	Personal
2	Registro de atención	El sistema podrá llevar un registro del número de clientes que atendió cada módulo, además del tiempo que fue utilizado.	Personal
3	Reportes estadísticos	El sistema recopilará los datos obtenidos durante el día de trabajo, para lo cual se mostrará en tablas Excel para una compresión más sencilla	Personal
4	Tipos de atención	El sistema permitirá escoger entre dos tipos de atención, ya sea, turno simple, donde cada personal atienden un único requerimiento o el turno dinámico donde el personal pueda recibir hasta cuatro tipos diferentes de atención.	Personal

Cuadro III
REQUERIMIENTOS FUNCIONALES – PERFIL DE USUARIO ADMINISTRADOR

#	REQUERIMIENTO	DESCRIPCIÓN	USUARIO
1	Configuración del tipo de atención	El usuario Administrador puede configurar el tipo de atención que ofrecerá cada módulo, ya sea simple o dinámico, además esta configuración la puede realizar en cualquier momento sin que se pierdan los datos, aun si esto es realizado durante la jornada de trabajo.	Administrador
2	Registro de trabajo	El sistema registra las actividades que se realizaron durante el día; los cuales deben ser extraídos desde el software que esta instalado en la computadora que se encuentra conectada a la tarjeta controladora.	Administrador
3	Reportes	Con los datos obtenidos de productividad, el usuario Administrador podrá generar reportes de trabajo para cada personal de atención, con el fin de mejorar el rendimiento del personal.	Administrador

IX-A2. Requerimientos no funcionales: .

Los requerimientos no funcionales representan las características y restricciones que tendrá el sistema de atención de cliente, con el fin de que el usuario conozca con detalle la funcionalidad de trabajo.

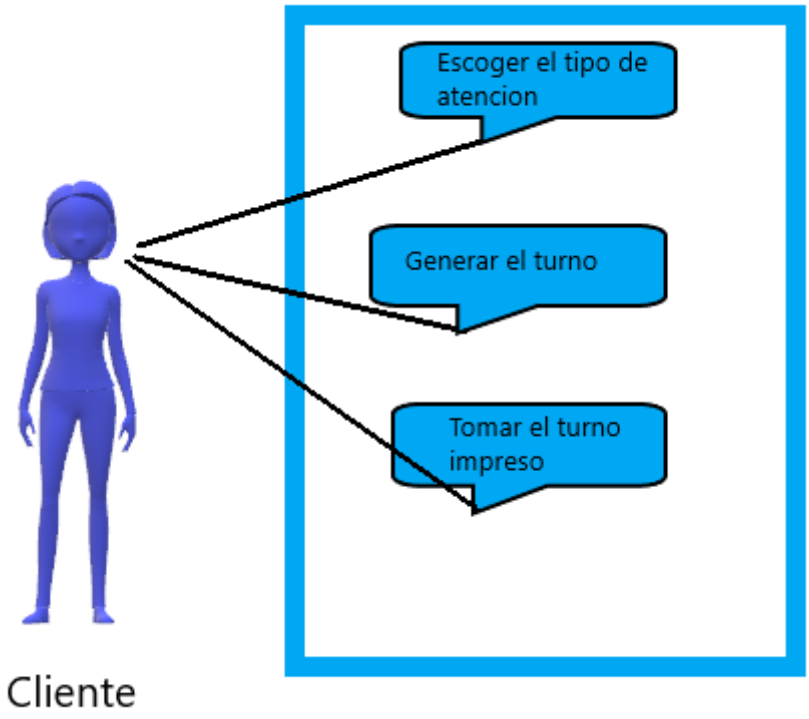
- Elaboración de un manual de usuario, que permitirá conocer las funciones, características y el modo de uso del sistema, además de una capacitación que se ofrecerá al personal que hará uso del sistema.
- El sistema permitirá que el usuario Administrador pueda elaborar reportes con los datos de la atención realizada en los módulos.
- El sistema tendrá una capacidad máxima de almacenamiento de información de los últimos 13200 turnos atendidos.
- La tarjeta controladora del sistema se comunica vía RS485 con la computadora donde esté instalada la aplicación cliente.
- La comunicación entre la impresora y la tarjeta controladora de la pantalla es con protocolo SPI a 2.4 GHZ.
- En caso de un corte energía, el sistema mantendrá los datos guardados de tal manera que no se pierda ningún registro de atención.

IX-A3. Diagrama de casos de uso : Los diagramas permiten conocer el tipo de interacción que surgirá entre el usuario y los elementos del sistema de impresión de ticket para la atención de clientes, a continuación, se listaran los diferentes elementos de este diagrama para este proyecto.

- Solicitar Turno

El cliente debe acercarse al módulo de ticket como se muetra en la ”Figura 10”, en el cual se podrá escoger entre 4 diferentes tipo de atención, los cuales fueron configurados previamente por el administrador.

Figura 10. Diagrama de caso de uso del cliente



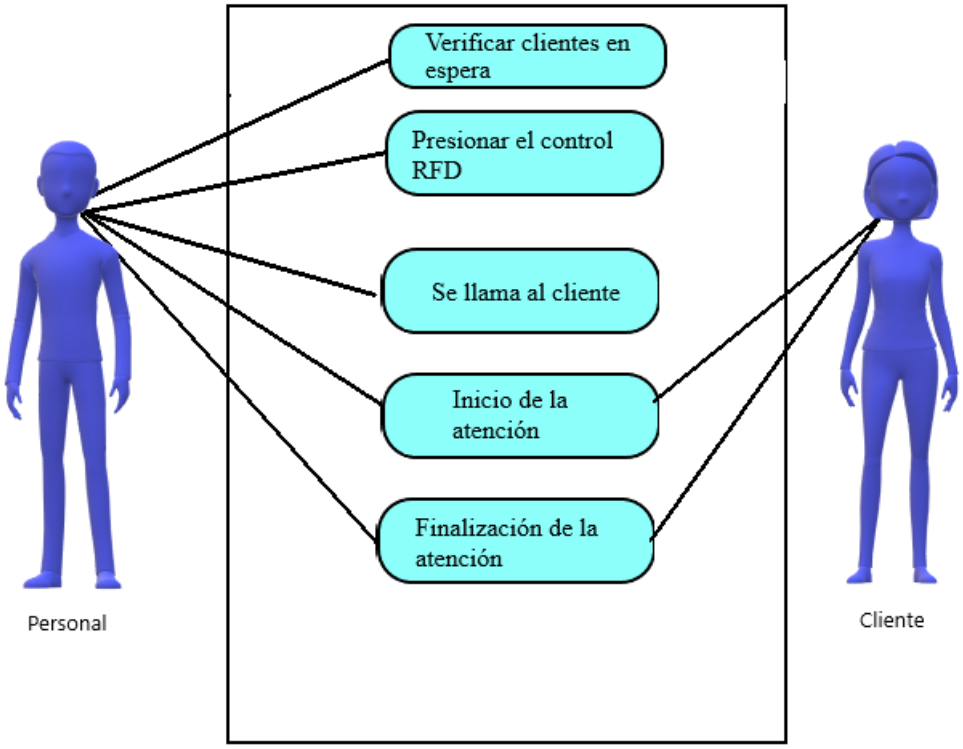
Cuadro IV
CASOS DE USO DE ATENCIÓN AL CLIENTE

Caso de Uso	**Generar un turno físico**
Actores	Cliente
Propósito	Imprimir un turno físico.
Resumen	El cliente tomara el ticket del turno para ser atendido, según su requerimiento y tiempo de llegada
Acción del Módulo de atención	**Respuesta del Sistema**
Presiona el botón de acuerdo con el tipo de actividad que necesita ser atendida	Imprime el ticket con el número de atención, la letra que identifica el modo de atención, hora y fecha

- Atención al cliente

El personal de atención al cliente usa el control RF 433 el cual tiene 4 botones (A B C D) como se muestra en la ”Figura XVII-B3”, el cual hace referencia al tipo de atención previamente configurado, para llamar a la cliente que se encuentra a la espera de atención.

Figura 11. Diagrama de caso de uso de la interacción Personal - Cliente



Cuadro V
CASO DE USO PARA VERIFICAR PERSONAS EN ESPERA

Caso de Uso	**Verificar clientes en espera**
Actores	Personal
Propósito	Verificar clientes en espera.
Resumen	El personal puede verificar de manera visual, si existen clientes en espera de atención.
Acción del Actor	**Respuesta del Personal**
Observar la existencia de personas en espera	El personal procede hacer uso del control RF para llamar al cliente a su modulo de atención.
Acción Alternativa	**Respuesta del Sistema**
Presionar el control	La pantalla realiza una alerta y muestra el número de atención.

Cuadro VI
CASOS DE USO DEL CONTROL

Caso de Uso	**Presionar el control RF 433**
Actores	Personal
Propósito	Llamar al cliente
Resumen	El módulo cuenta con cuatro botones (A B C D) cada uno de ellos representa los diferentes tipos de atención
Acción del Actor	**Respuesta del Sistema**
El personal presiona el botón correspondiente a su tipo de atención	El Sistema alerta mediante una alarma que el personal solicita la presencia de un cliente y al mismo tiempo, de manera visual la pantalla muestra el número del cliente que será atendido
	Respuesta Alternativa del Sistema
	En caso de no haber personas en espera para ese tipo de atención, la pantalla no cambia, lo cual permite al personal llamar otro tipo de usuario si está configurado previamente.

Cuadro VII
CASOS DE USO PARA LLAMAR AL CLIENTE

Caso de Uso	**Se llama al cliente**
Actores	Personal, Cliente
Propósito	Llamar al cliente
Resumen	El personal llama al cliente para su atención
Acción del Actor	**Respuesta del Sistema**
El cliente es llamado	El sistema captura el tiempo en que se llama al cliente.
Acción Alternativa	**Respuesta del Sistema**
Si no existe cliente en cola, el personal puede atender otro tipo de requerimiento	El Sistema alerta al personal de que no hay cliente en espera, lo cual le permite presionar otra opción de los cuatro tipos de atención, para llamar a un cliente diferente tanto en requerimiento como de modulo

Cuadro VIII
CASOS DE USO PARA EL INICIO DE LA ATENCIÓN

Caso de Uso	**Inicio de la atención**
Actores	Personal, Cliente
Propósito	Se registra el tiempo de atención
Resumen	Mediante un reloj interno, el sistema captura el tiempo que se está utilizando para atender al cliente
Acción del Actor	**Respuesta del Sistema**
El personal recibe al cliente y escucha su requerimiento	El sistema contabiliza el tiempo que se está usando para atender al cliente, mientras que evita que otra persona sea llamada para el módulo que se encuentra ocupado.

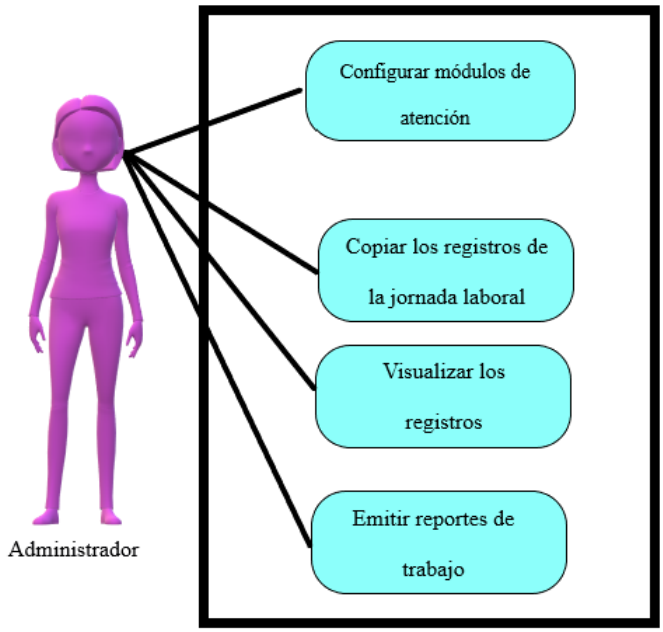
Cuadro IX
CASOS DE USO PARA FINALIZAR LA ATENCIÓN

Caso de Uso	**Finalizar la atención**
Actores	Personal, Cliente
Propósito	El sistema captura el tiempo en que se finalizó la atención
Resumen	Luego de atender el requerimiento del cliente, el sistema captura el tiempo que se utilizó.
Acción del Actor	**Respuesta del Sistema**
El Personal presiona el botón	El sistema realiza dos acciones: • Registra la hora, la cual sirve para contabilizar el tiempo que se empleó para atender el cliente. • Llamar al siguiente cliente en espera

- Usuario Administrador

El usuario Administrador es la persona encargada de realizar la captura de los registros de la jornada laboral, los mismos que sirven para realizar los reportes de productividad que tienen como objetivo detectar el personal que necesita una capacitación para mejorar su rendimiento.

Figura 12. Diagrama de caso de uso del usuario Administrador



Cuadro X
CASOS DE USO PARA LOS MÓDULOS DE ATENCIÓN

Caso de Uso	**Módulos de atención**
Actores	Administrador
Propósito	Configurar los modos de atención.
Resumen	Permite configurar cada módulo con un modo de atención ya sea simple o dinámica
Acción del Actor	**Respuesta del Sistema**
Atención Simple	El personal solo puede llamar a un cliente específico de atención.
Atención Dinámica	El personal puede llamar a diferentes tipos de cliente para su atención.

Cuadro XI
CASOS DE USO PARA EL REGISTRO DE LA JORNADA LABORAL

Caso de Uso	**Registro de la jornada laboral**
Actores	Administrador
Propósito	Recopilar la información de trabajo de cada personal de atención al cliente
Resumen	Copiar los archivos registrados por el sistema
Acción del Actor	**Respuesta del Sistema**
Ingresar al programa de configuración	El software permite copiar los registros desde un periodo en específico hasta por módulo individual.
Acción Alternativa	**Respuesta del Sistema**
Abrir el archivo descargado	Los datos se muestran en formato Excel

Cuadro XII
CASOS DE USO PARA VISUALIZAR LOS REGISTROS

Caso de Uso	**Visualizar los registros**
Actores	Administrador
Propósito	Emitir reportes
Resumen	Los datos presentados en formato Excel permite conocer la productividad de cada personal, basado en el tiempo que tarda en dar una solución y el número de personas que atendió
Acción del Actor	**Respuesta del Sistema**
Formato Excel	Los registros son presentados en formato Excel, donde se registra, el tiempo que tardo en atender un cliente, el número de clientes atendidos y se los compara con el resto del personal
Acción Alternativa	
Formato gráfico	El usuario Administrador puede usar los datos capturados para presentarlos en cuadros estadísticos para un mejor entendimiento.

Cuadro XIII
CASOS DE USO PARA EL REPORTE DE TRABAJO

Caso de Uso	**Reportes de trabajo**
Actores	Administrador
Propósito	Mejorar la productividad
Resumen	El administrador podrá emitir reportes de productividad basados en los datos recopilados por el sistema con el objetivo de mejorar la atención,
Acción del Actor	**Programa Excel**
Reporte de trabajo	El administrador podrá emitir un reporte de trabajo, utilizando los registros del sistema como base para calificar la productividad de cada personal.

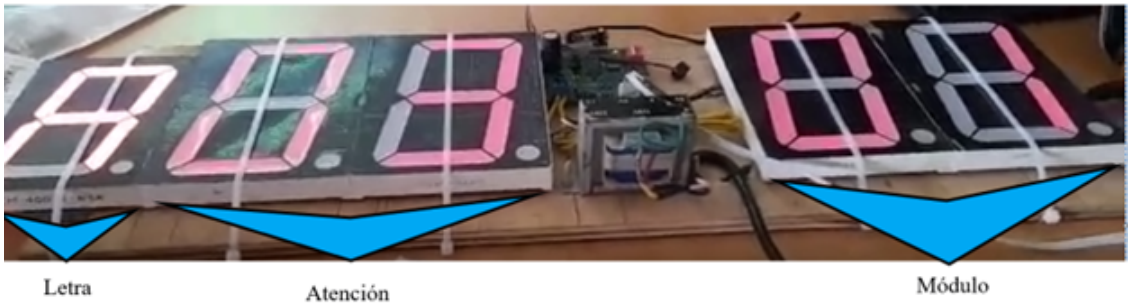
IX-B. Diseño

Para el diseño del sistema se tomará en cuenta los requerimientos que comento el distribuidor de Promaquinas Richard Medina con el cual se charló respecto al tema, además se limitaran los componentes que integraran el diseño del sistema a los que se encuentran actualmente en las electrónicas ecuatorianas, con la finalidad de tener a disposición los repuestos en caso de mantenimiento o reparaciones.

IX-B1. Diseño del la Pantalla: .

Para el diseño del display se usarán 5 módulos de 7 segmentos como se muestra ”Figura 13”, los cuales mostraran el módulo de atención al cual se dirigirá el cliente que va del 0-99, el número de atención del 0-99 y el tipo de atención que se definirá por una letra del abecedario.

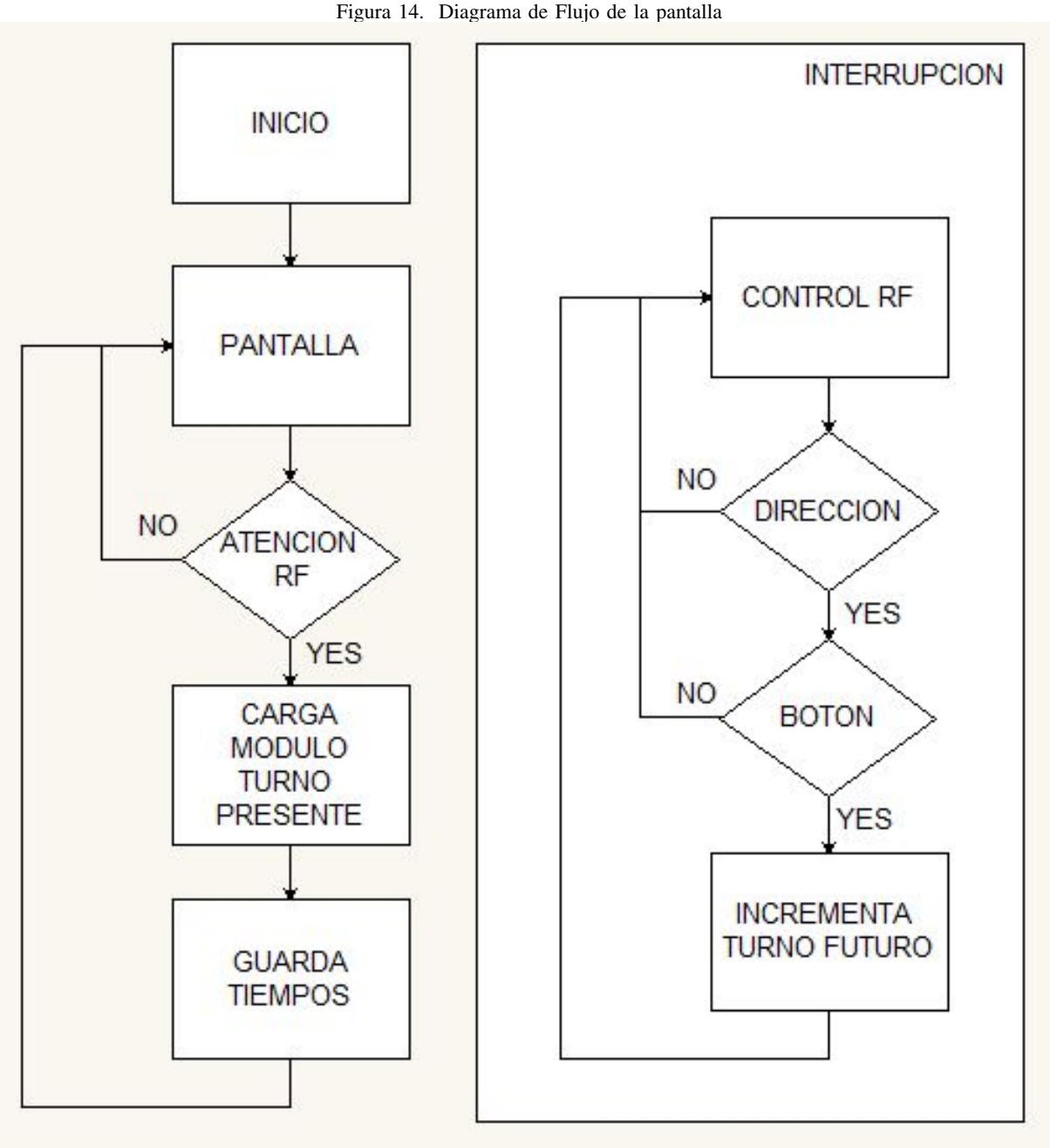
Figura 13. Diseño preliminar del display



Cuadro XIV
TABLA DE MODO DE LECTURA DE LA PANTALLA

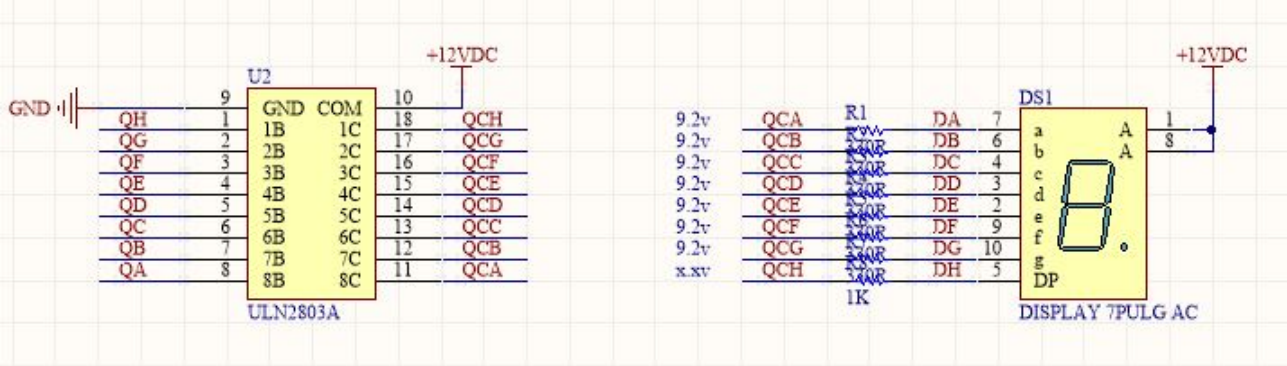
Display	Descripción
Letra	Se utiliza para identificar el tipo de atención que necesita el usuario, el cual estará definido por una letra del abecedario en letra minúscula, según configure el administrador.
Atención	Sirve para visualizar el número atención que previamente estará impreso en el ticket, consta de 2 displays de 7 segmentos en el cual se observará desde 0 hasta el 99, luego de llegar al 99, se vuelve a visualizar desde 0 y así sucesivamente
Módulo	Conformado por 2 displays de 7 segmentos, se utilizará para reconocer cuál es el módulo de atención al cual se debe dirigir el cliente para solucionar su inquietud.

Figura 14. Diagrama de Flujo de la pantalla



Como se muestra en la figura ”Figura 15”, la configuración para el control de una pantalla display de 7 segmentos, lo cual es controlado por un bloque de transistores que dispone el controlador Uln2803 hacia la pantalla, la misma que se comunica con la tarjeta controladora con la finalidad de recibir la información que activara los segmentos para su iluminación.

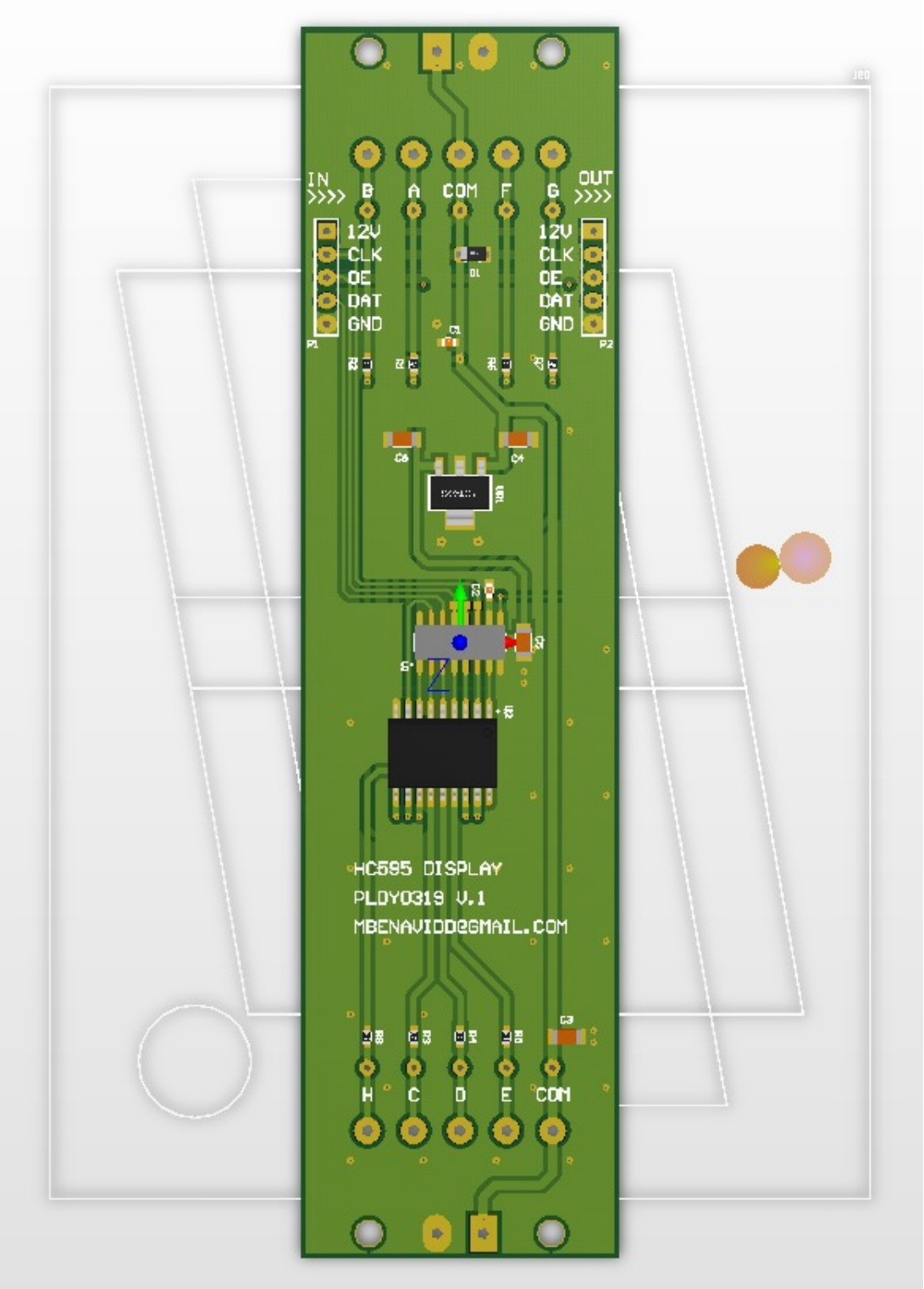
Figura 15. Diagrama de configuración del display



Cada pantalla de la tarjeta controladora del display de 7 segmentos, tiene un chip register como se muestra en la ”Figura 16”, al cual se le carga bits por bits para el encendido de los segmentos del display, es decir de 4 bits altos y 4 bits bajos dando un total de 8 bits que representan al byte completo, además cada pantalla recibe un Byte con la información necesaria para encender los segmentos dando lugar el número o letra, que se visualiza en la pantalla.

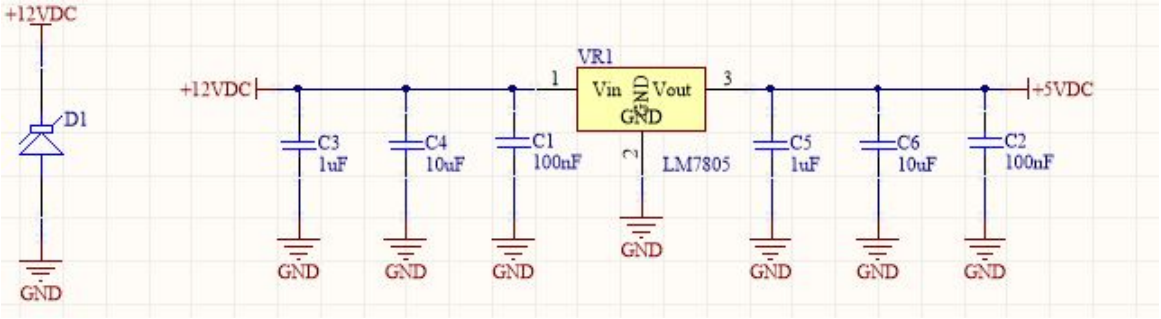
El encargado de enviar esta información será la tarjeta controladora de la pantalla, cabe señalar que esta información ocurre secuencialmente, es decir, el primer de los 5 bytes enviados representa la letra, los dos segundos bytes representan la atención y los dos últimos el módulo de atención.

Figura 16. Tarjeta controladora del display de 7 segmentos



Para la alimentación de la tarjeta se implementó un regulador de 12V voltios hacia 5V, ya que tenemos un chip de registro de desplazamiento débil para el encendido de cada segmento del display.

Figura 17. Diagrama de alimentación del display



Para la comunicación entre la tarjeta de control y chip de registro de desplazamiento de los bytes, como se muestra ”Figura 19”. , encargado del encendido de cada segmento de la pantalla, se reciben los datos por medio de los pines LATCH, shift y el habilitador de encendido AQ-QH.

Figura 18. Diagrama de interfaz de conexión del display

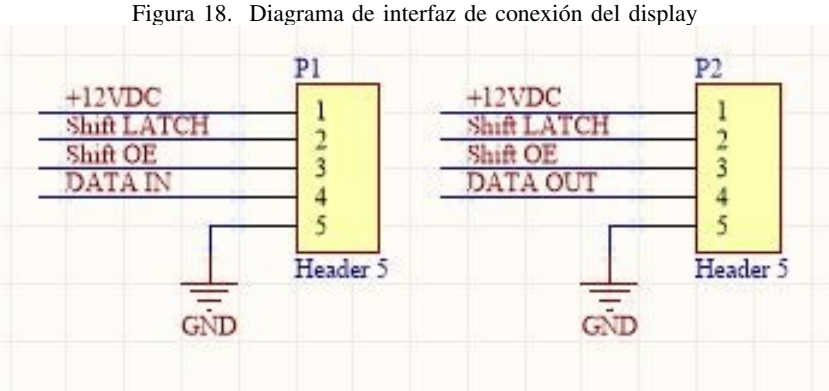
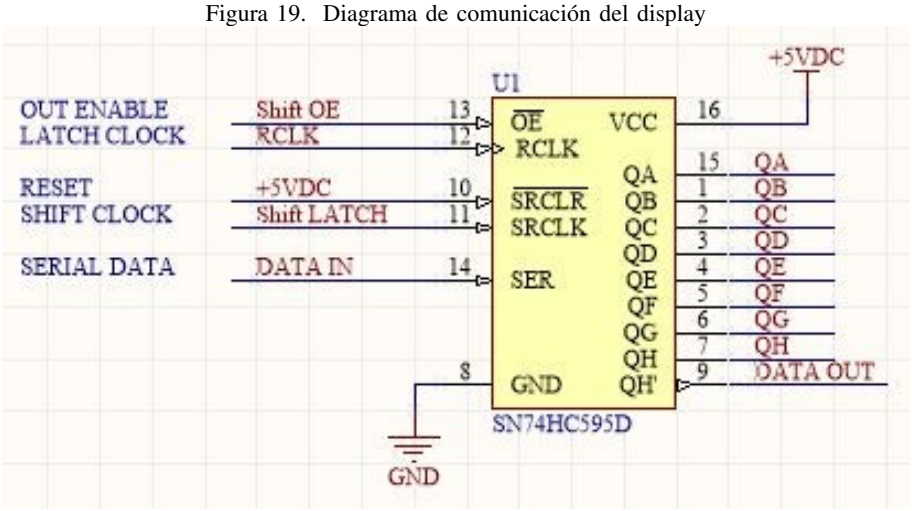


Figura 19. Diagrama de comunicación del display



IX-B2. Diseño de la tarjeta Master: .

Con el fin de controlar la comunicación entre, la impresora, la botonera, y el display, usaremos un microcontrolador 18LF4620 de la familia Microsoft, que servirá como procesador de la tarjeta controladora, como se muestra en la ”Figura 20”, que registra todo lo que sucede entre los componentes, además de analizar y valorar la información que se obtiene de los periféricos para ser presentada mediante transmisión RICH haca la pantalla.

Figura 20. Diseño de la tarjeta controladora Master

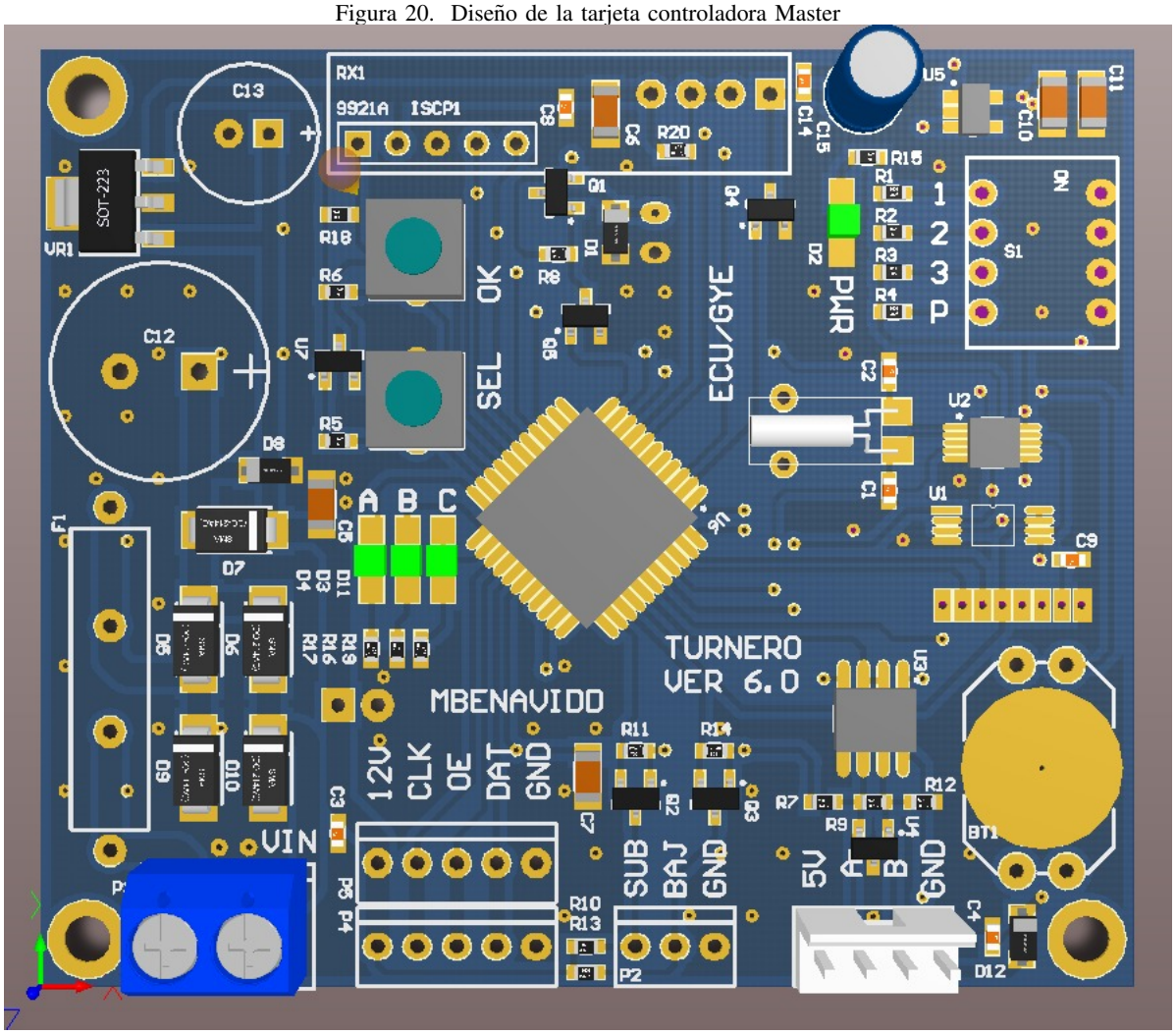
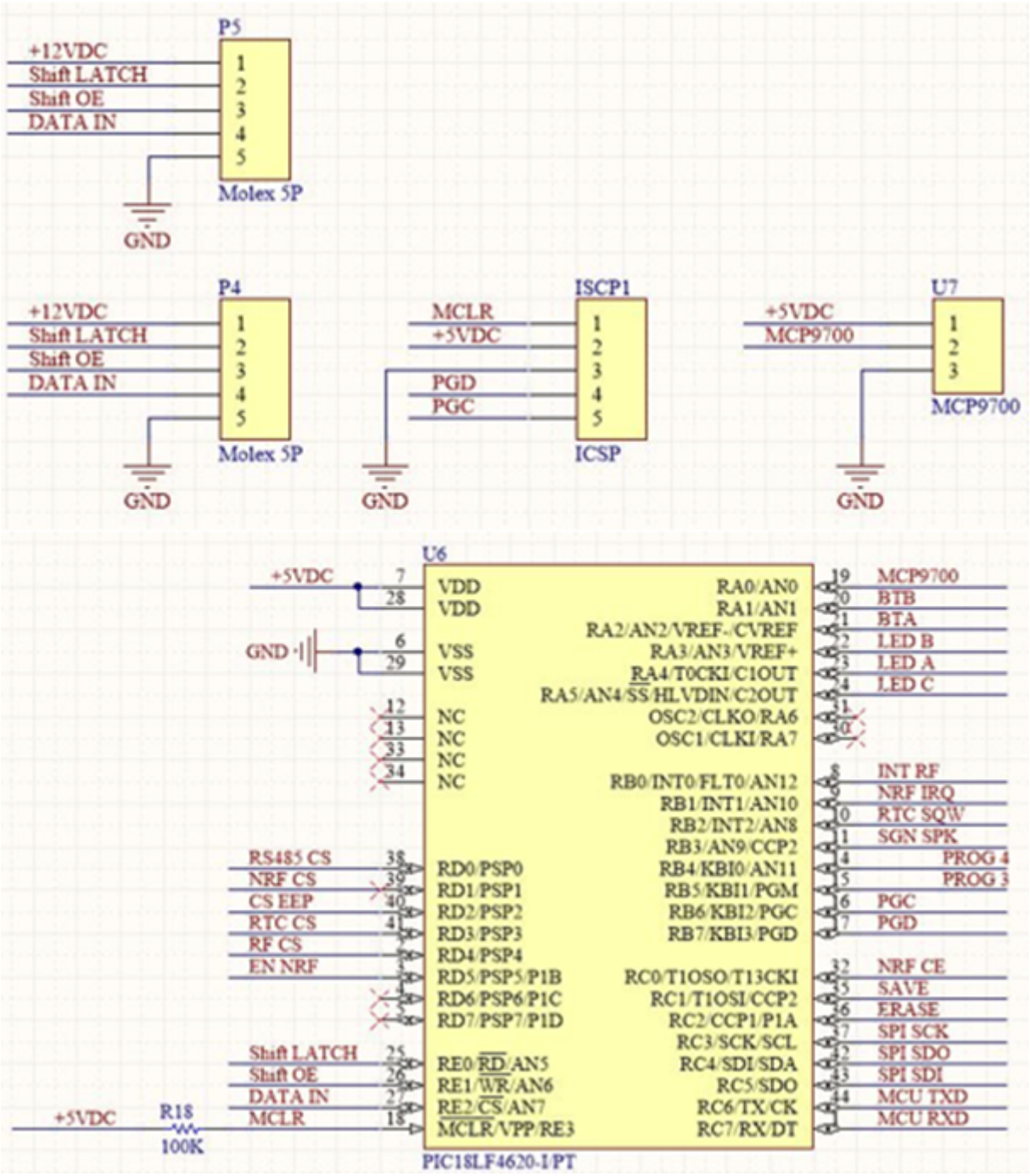
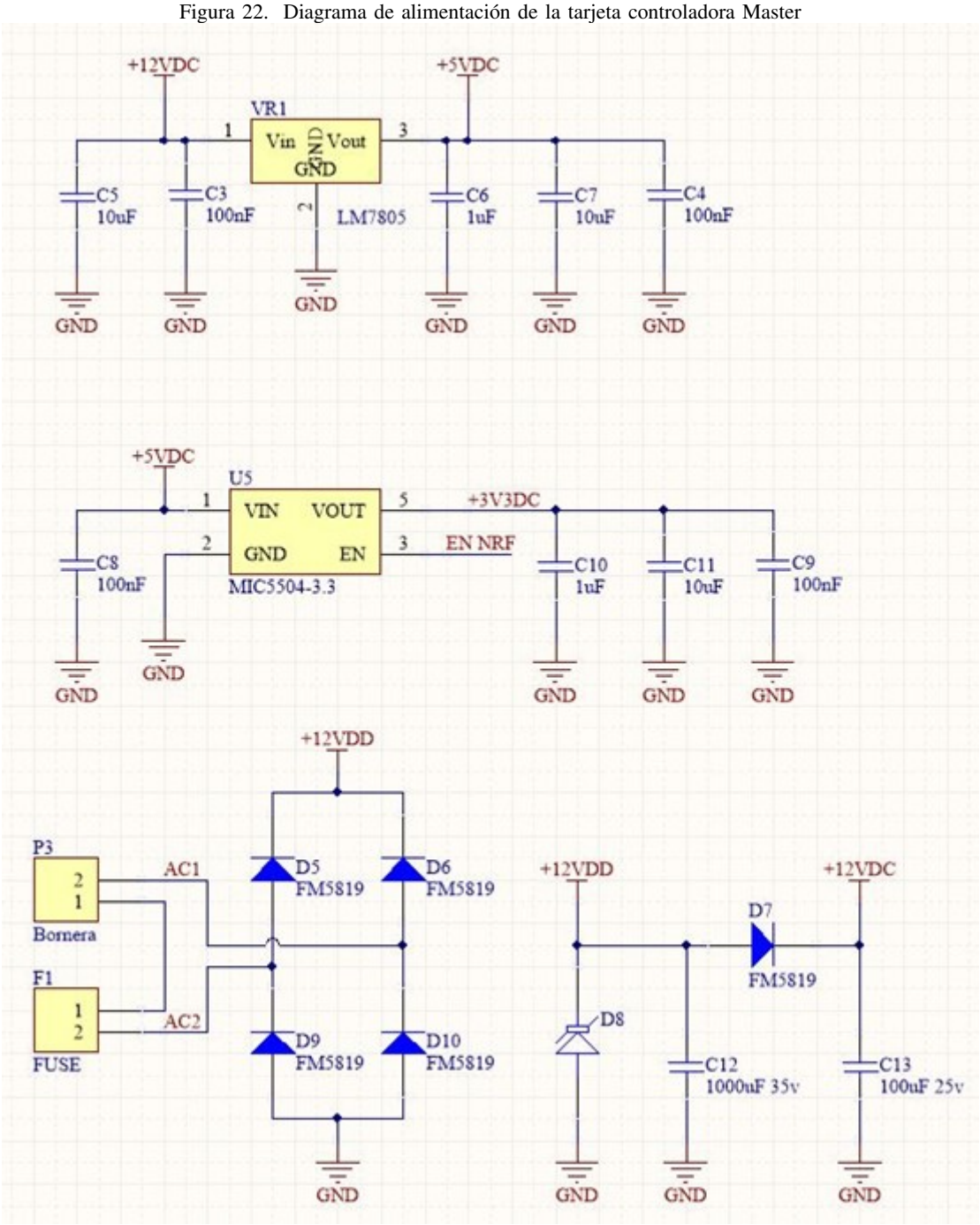


Figura 21. Diagrama del PIC18LF4620 con sus interfaces



En la ”Figura 21”se presenta la configuración para la fuente, para la cual se utiliza un puente de diodos para la posibilidad de usar un transformador de 110V o simplemente una fuente regulable, además se usa un regulador de 5V para la tapa de alimentación del controlador Fig.16, necesarios para alimentar el PIC18LF4620 y luego tenemos una fuente más pequeña de 3.3V para el módulo de comunicación de radiofrecuencia.

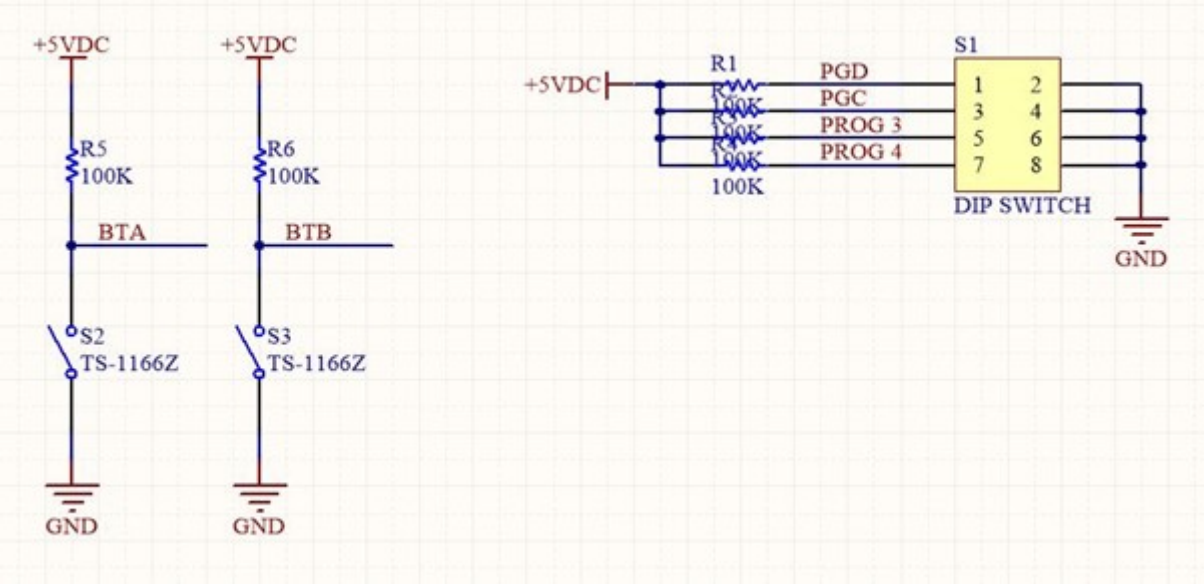
Figura 22. Diagrama de alimentación de la tarjeta controladora Master



Con la intención de configurar el modo de trabajo se incorporó botones AYB que sirven para grabar o borrar según lo que sé desea, además tiene un bloque de switch que sirve para seleccionar el modo de trabajo, con ello se busca realizar, por ejemplo, modo de grabado de aprendizaje del control, modo de trabajo en la pantalla y encerar

los números de la pantalla.

Figura 23. Diagrama de alimentación de la tarjeta controladora Master

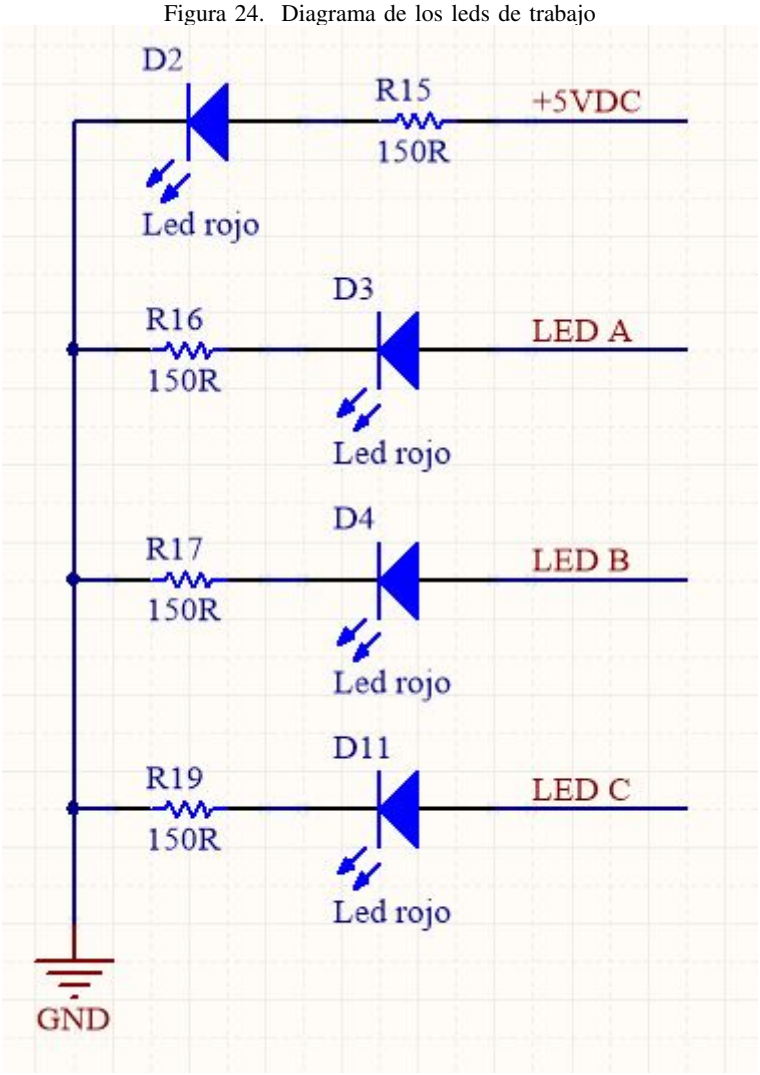


Para conocer de manera visual los diferentes procesos de trabajo de la tarjeta, se utilizó un bloque de diodos que servirán como pilotos para saber el modo de trabajo que está corriendo dentro de la tarjeta controladora, como se muestra en la ”Tabla XV”.

Cuadro XV
INDICADORES LEDS

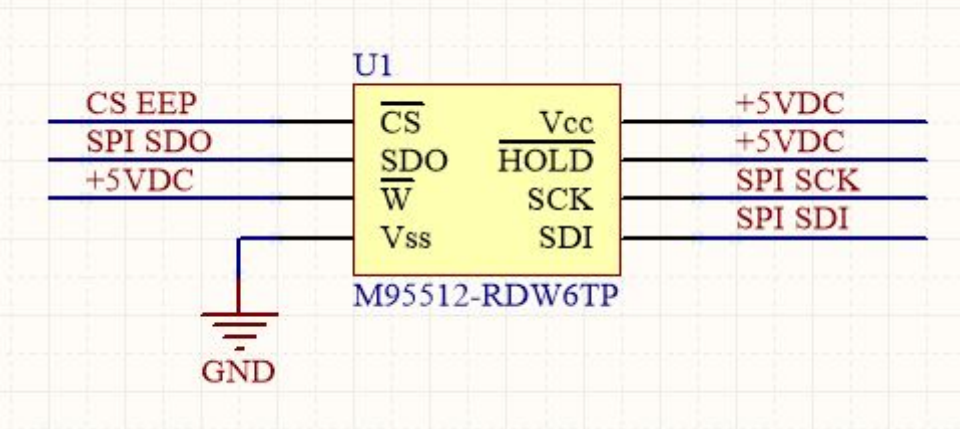
Modo de trabajo	Descripción
Luz A encendida y Luz B apagada (On)	Selección del modo de trabajo
Luz A apagada y Luz B encendida	Aprendizaje de la frecuencia de oscilación del control
A y B encendida	Borrar todo y regresar al modo de fábrica
Luz C	Se encuentra deshabilitada hasta una necesidad futura

Figura 24. Diagrama de los leds de trabajo



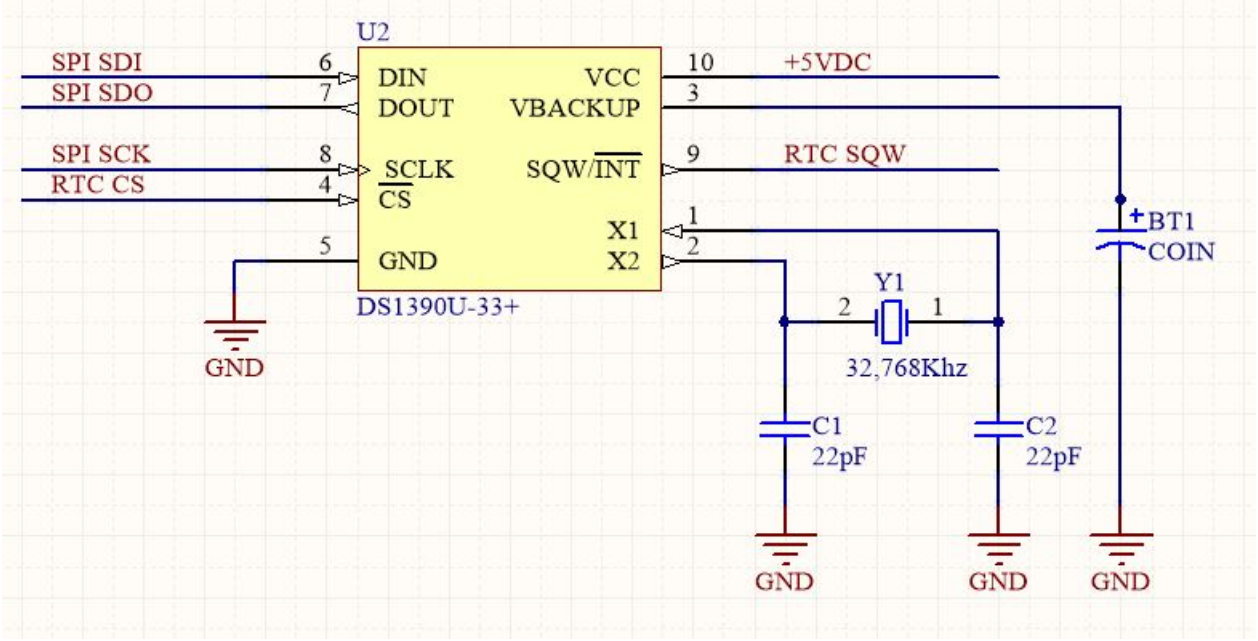
Con el fin de almacenar todos los procesos que se realizan en el equipo, el cual dispone de comunicación Serial Peripheral Interface (SPI) la cual fue desarrollada por Motorola que permite una comunicación de interfaz serial sincrónica para dispositivos a corta distancia, se usa un chip de capacidad 512 bytes como se muestra en la ”Figura 25”, el cual representa una capacidad suficiente para el trabajo que se necesita realizar.

Figura 25. Diagrama del chip M95512



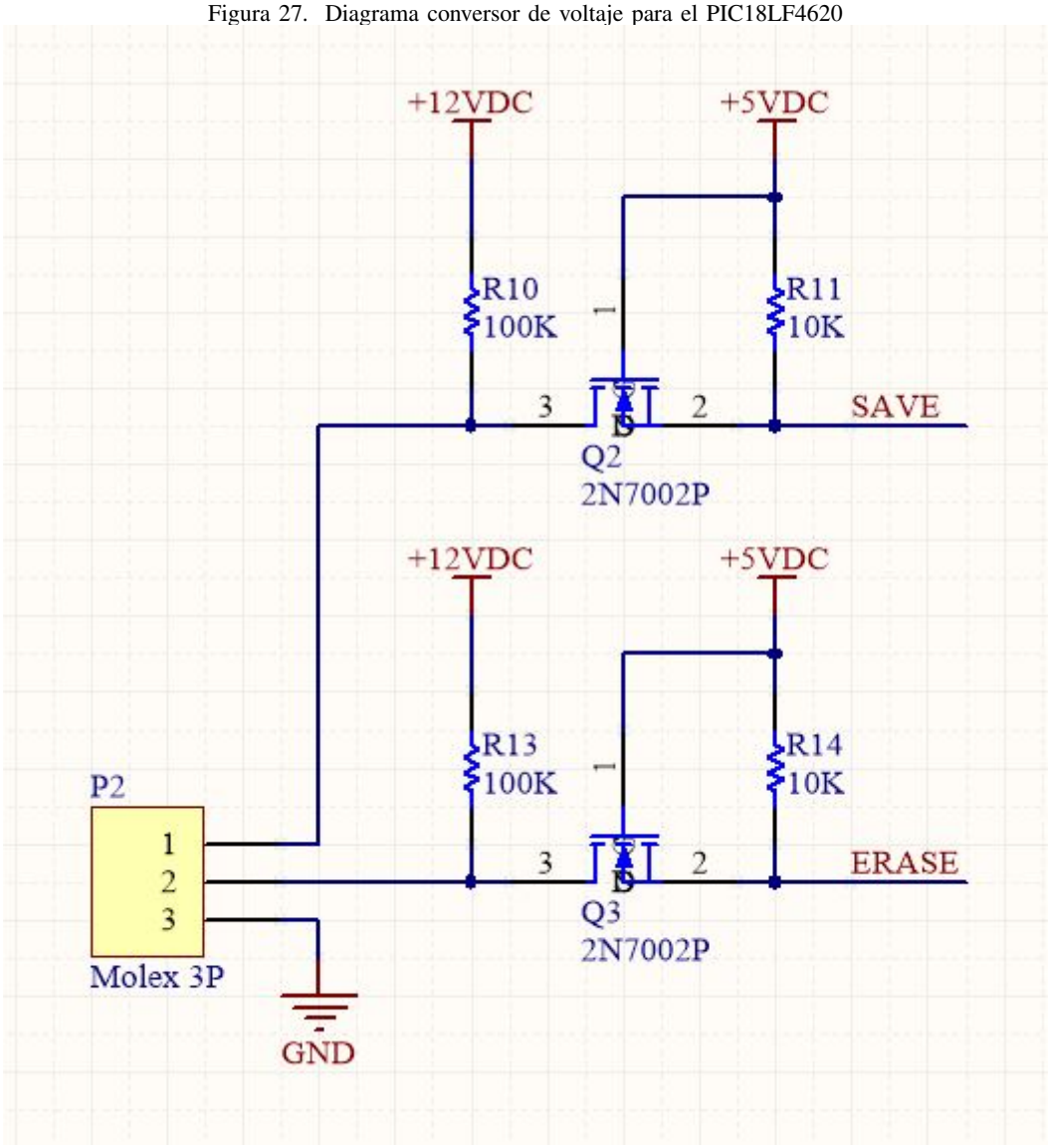
Para la captura y almacenamiento del tiempo se usará una memoria eeprom ds1390u, al tratarse de un RTC en real timé clock, la cual se alimentará con 5V, además que nos permite una comunicación SPI con frecuencia programable.

Figura 26. Diagrama del circuito del reloj RTChttps://es.overleaf.com/project/627ec9fc1a49e3d22bff09ca con su circuito oscilante



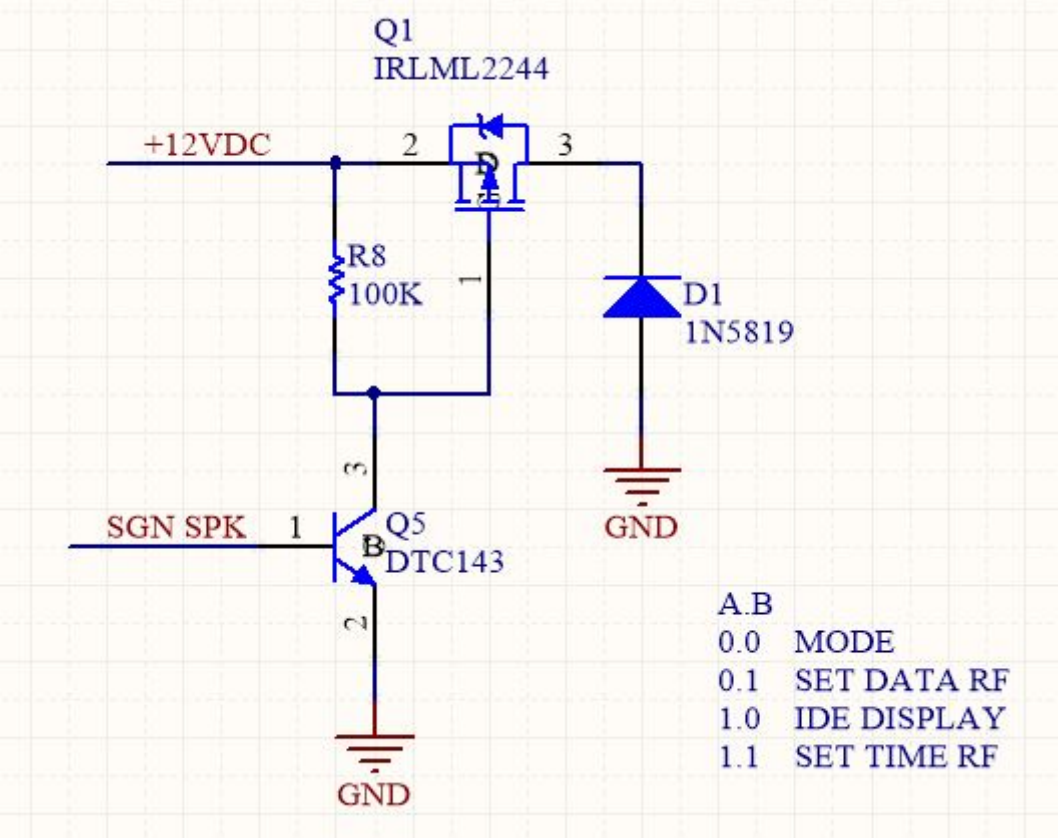
Para el condicionamiento de la señal, debido a que el microcontrolador PIC18LF4620 trabaja con niveles de voltaje pequeños de 3.3V, es preferible que haya una interfaz de conversión de voltaje para que no vaya a suceder que se ingrese un voltaje no adecuado y el chip termine quemado por sobre voltaje.

Figura 27. Diagrama conversor de voltaje para el PIC18LF4620



Con el fin de activar la alerta al usuario para el cambio de turno de la atención, se incorporó un circuito de potencia, como se muestra en la ”Figura 28”, para el accionamiento del booster o parlante, que trabaja con 12V.

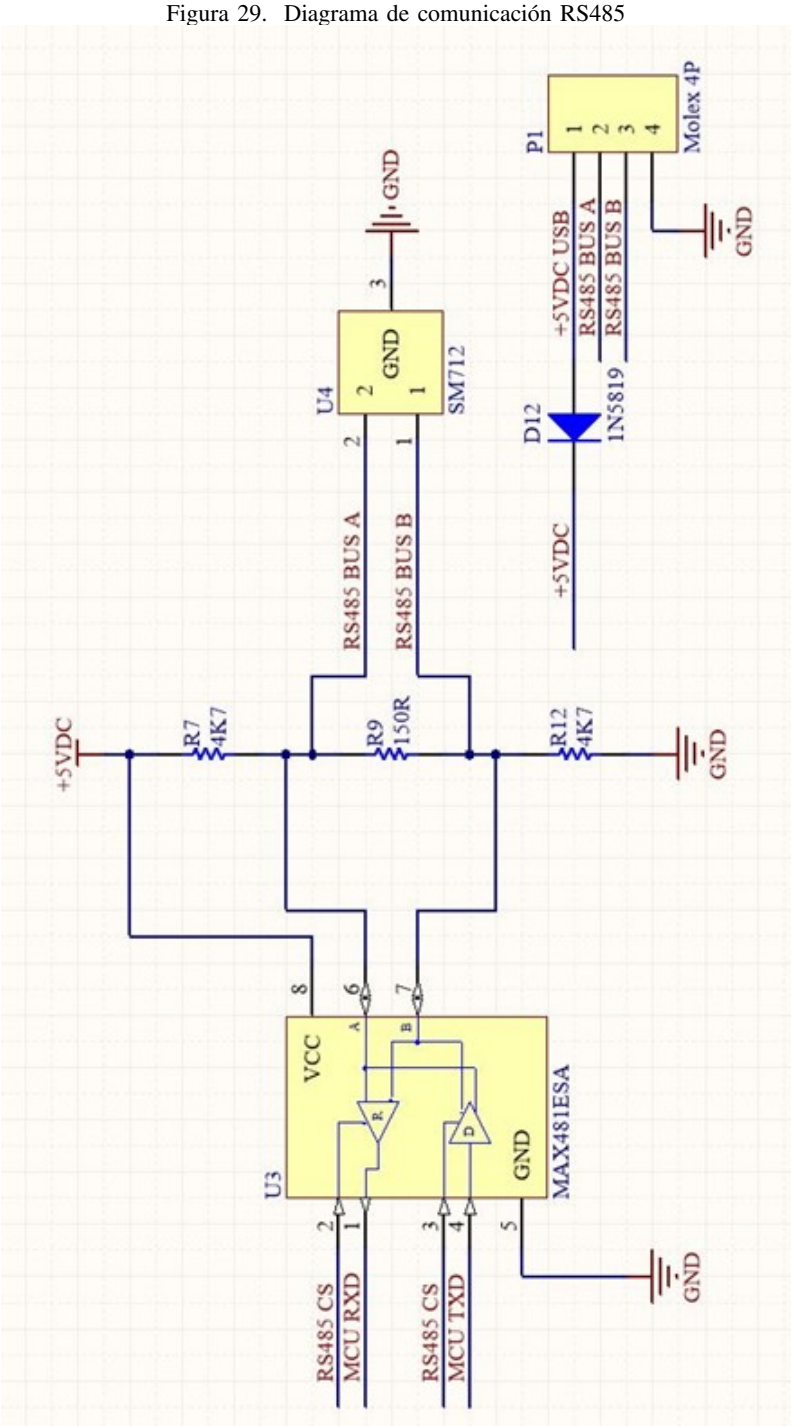
Figura 28. Diagrama del circuito de potencia para el parlante



- Comunicación remota

Para la configuración remota ente del sistema desde un computador, se incorporó la interfaz de comunicación RS 485 que sirve para seleccionar el modo de trabajo que se desea realizar en un sitio en particular, se usó este tipo de comunicación porque permite conectar varios dispositivos a través del mismo bus.

Figura 29. Diagrama de comunicación RS485



Para la recepción de datos usamos una interfaz de comunicación de radiofrecuencia 433mhz que permite la admisión de datos, por medio de los controles RF 433 de 4 canales, como se muestra en la ”Figura 30”, que disponen los asistentes al momento de llamar a un nuevo usuario, cabe mencionar que la frecuencia que se recibe el módulo es controlada para encender o apagar según sea su modo de trabajo.

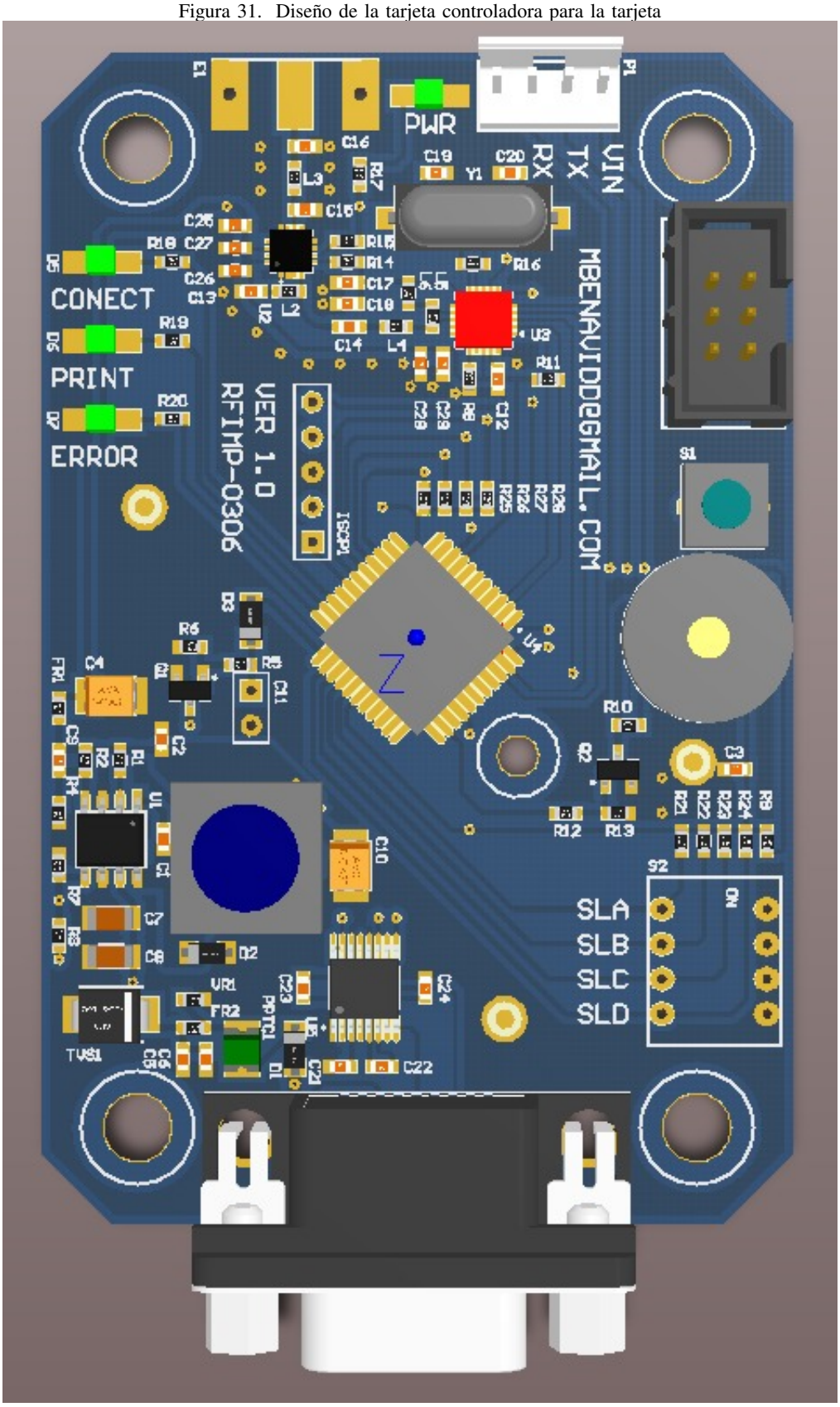
Figura 30. Control RF433 para el Personal de atención al cliente

- Diagrama de flujo.

IX-B3. Diseño de la tarjeta controladora para la impresora térmica: .

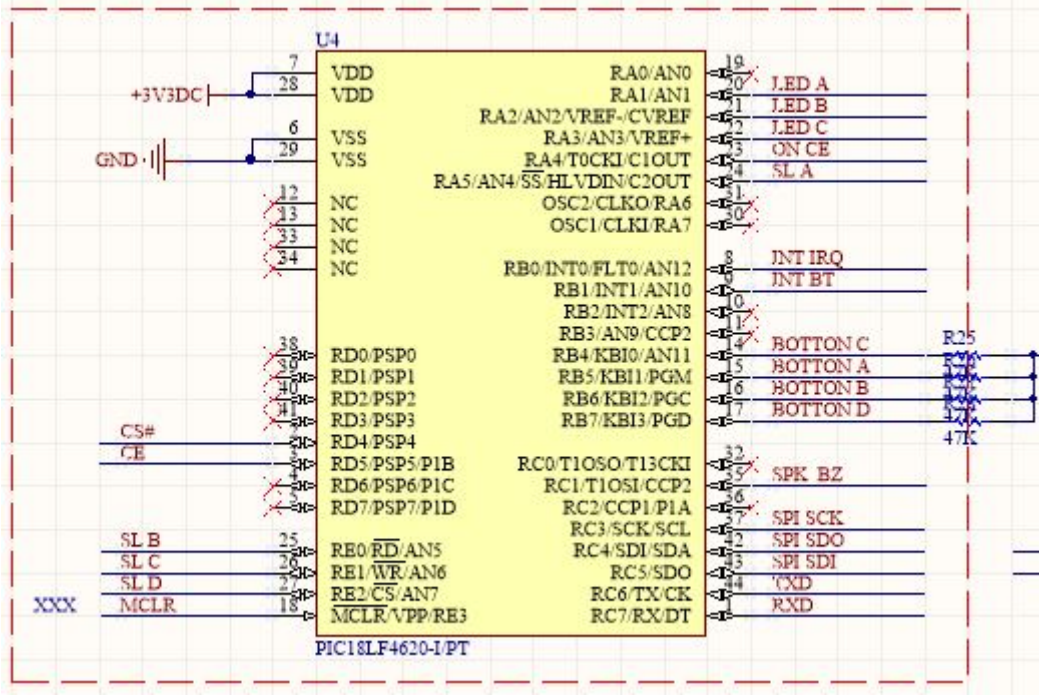
Para la impresión de ticket se usó una impresora EPSON TM-T20, ya que tiene el terminal RJ45 que permite la comunicación RS232, la cual nos da la ventaja de conectar depósitos de manera inalámbrica a una mayor distancia que I2C o TTL, además, de ser una las comunicaciones estándar en varios dispositivos, facilitando en el trabajo de configuración entre componentes.

Figura 31. Diseño de la tarjeta controladora para la tarjeta



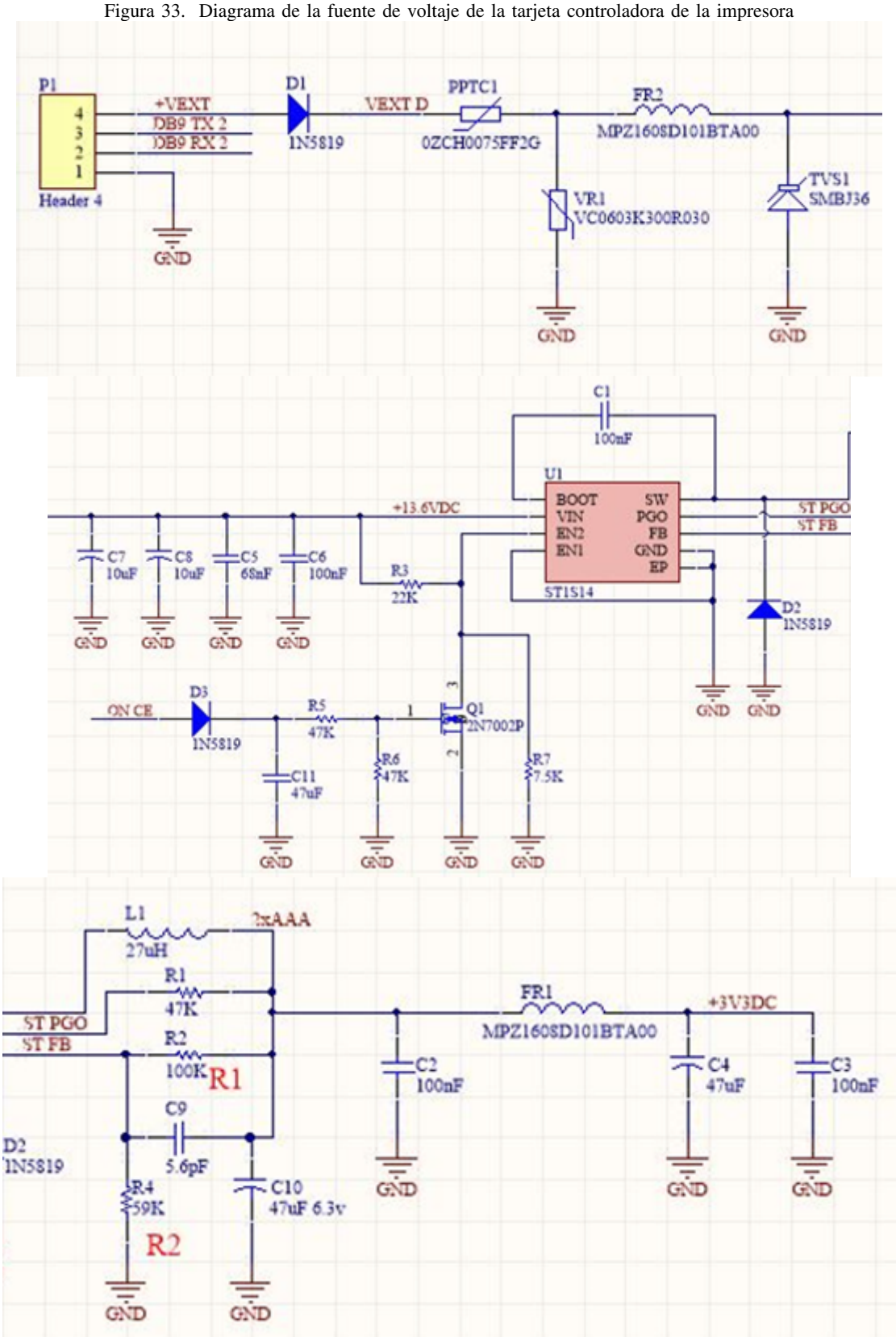
Para la tarjeta controladora de la impresora se volvió a usar PIC18LF4620 de la familia microchip, que se encargara del proceso de analizar la información que interactúa entre los equipos conectados por la comunicación RS232, recordemos que la tarjeta realizara dos trabajos al mismo tiempo, el primero es enviar a la tarjeta controladora los datos del turno que serán archivados y como segundo enviar la información que será impresa en el ticket.

Figura 32. Diagrama de comunicación del PIC18LF4620 de la tarjeta controladora de la impresora inferior



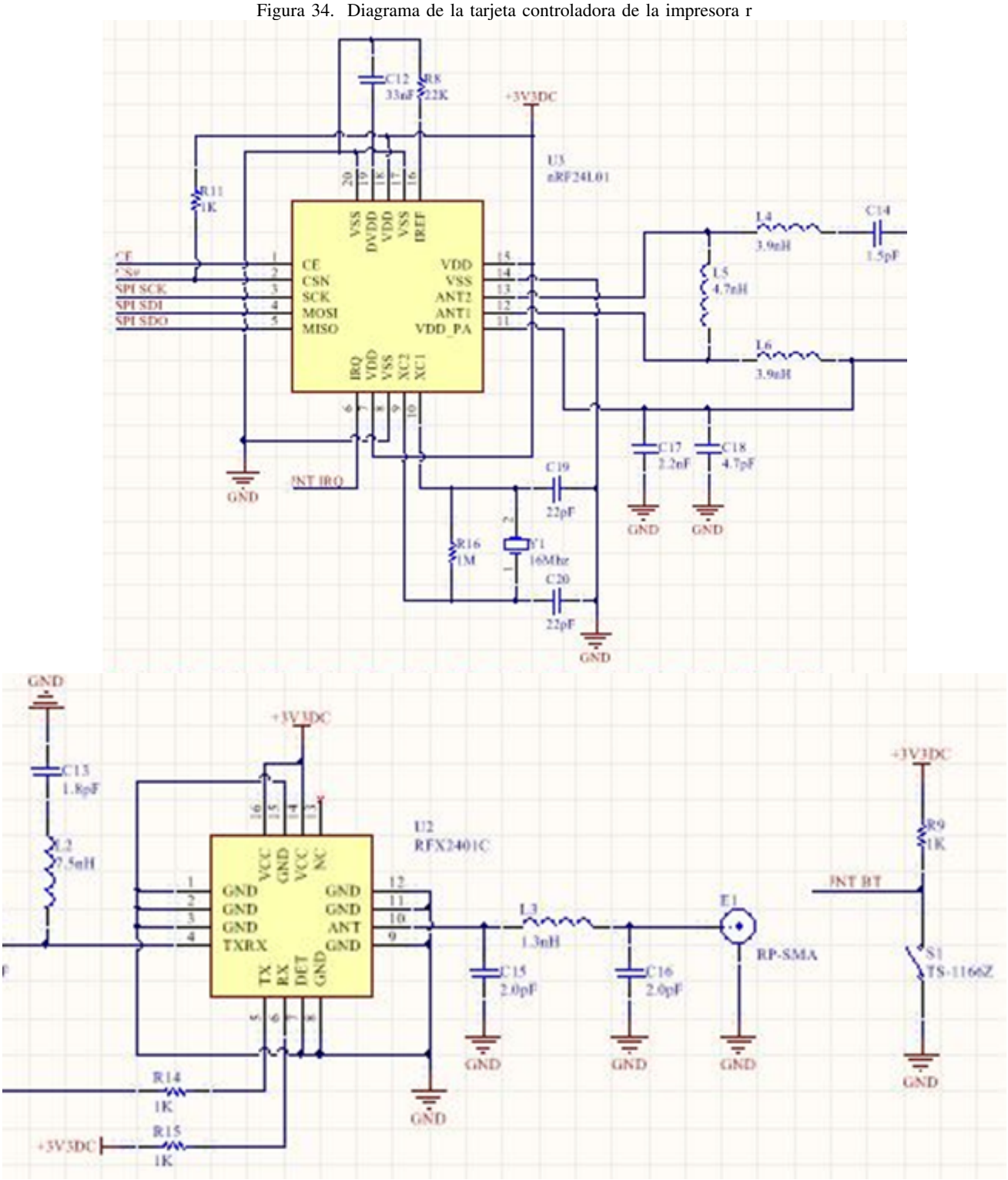
Se diseña una fuente de voltaje que es más sofisticada debido, a que se está capturando el propio voltaje de la impresora, dado que el voltaje que entrega esta por los 25V y necesitamos reducirla a 3.3V.

Figura 33. Diagrama de la fuente de voltaje de la tarjeta controladora de la impresora



Para la interconexión entre la impresora y la pantalla que transmite de manera bidireccional la información, se realiza mediante un circuito de radiofrecuencia, ya que la pantalla recibe de la impresora la solicitud, de un nuevo turno, luego la pantalla le transmite la información del turno para que imprima de manera seguida la información de la hora y fecha.

Figura 34. Diagrama de la tarjeta controladora de la impresora r



El módulo de comunicación trabaja en una frecuencia de 2.4 GHZ, puesto que, al ser una frecuencia no comercial libre, nos permite trabajar sin la necesitada de solicitar licencias o patentes para realizar el trabajo.

Mientras tanto la interfaz RS232 sirve para controlar la impresora, por el hecho de que está tiene la particularidad de contar con dos tipos de control, una USB y la otra una interfaz de puerto paralelo, siendo la segunda la del interés para la comunicación, debido a esto, se puede transmitir los comandos de impresión del usuario que se solicita un turno.

Figura 35. Diagrama de la comunicación de la tarjeta para la impresora

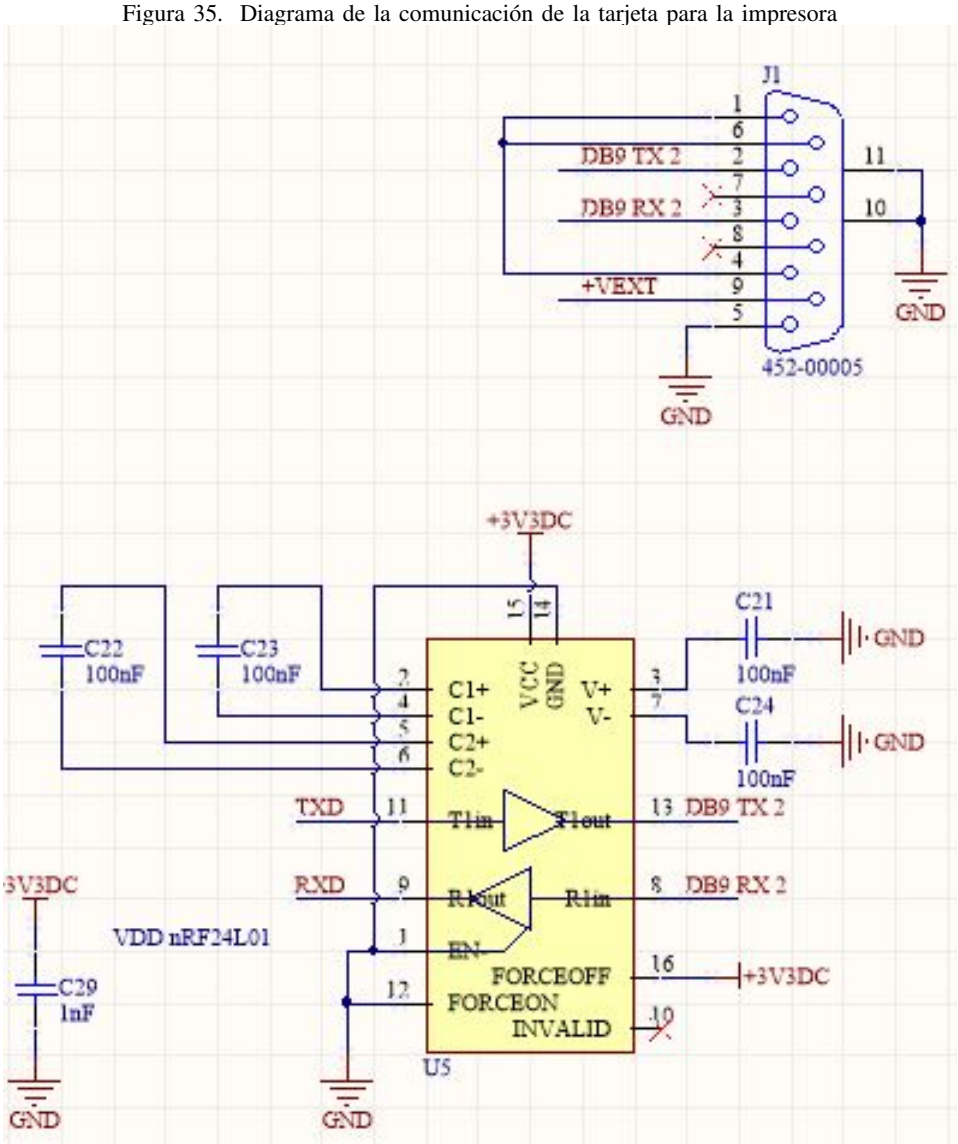
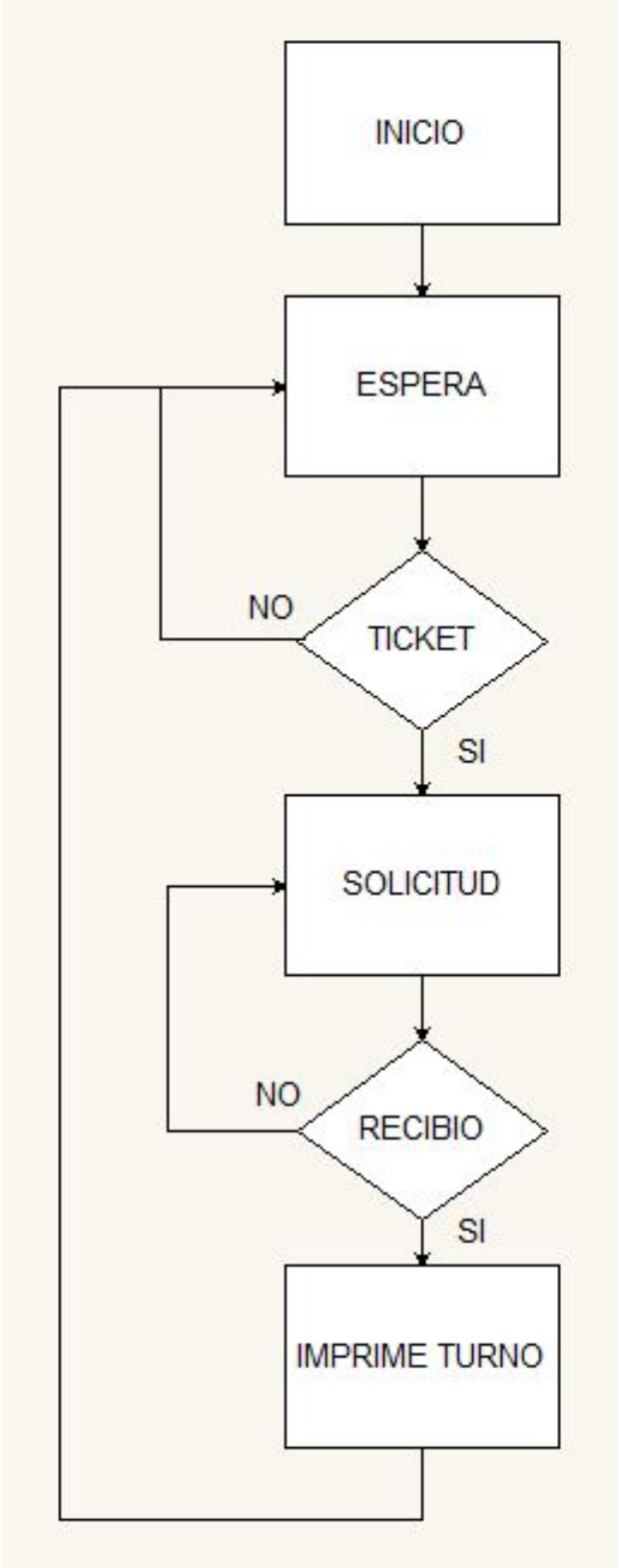


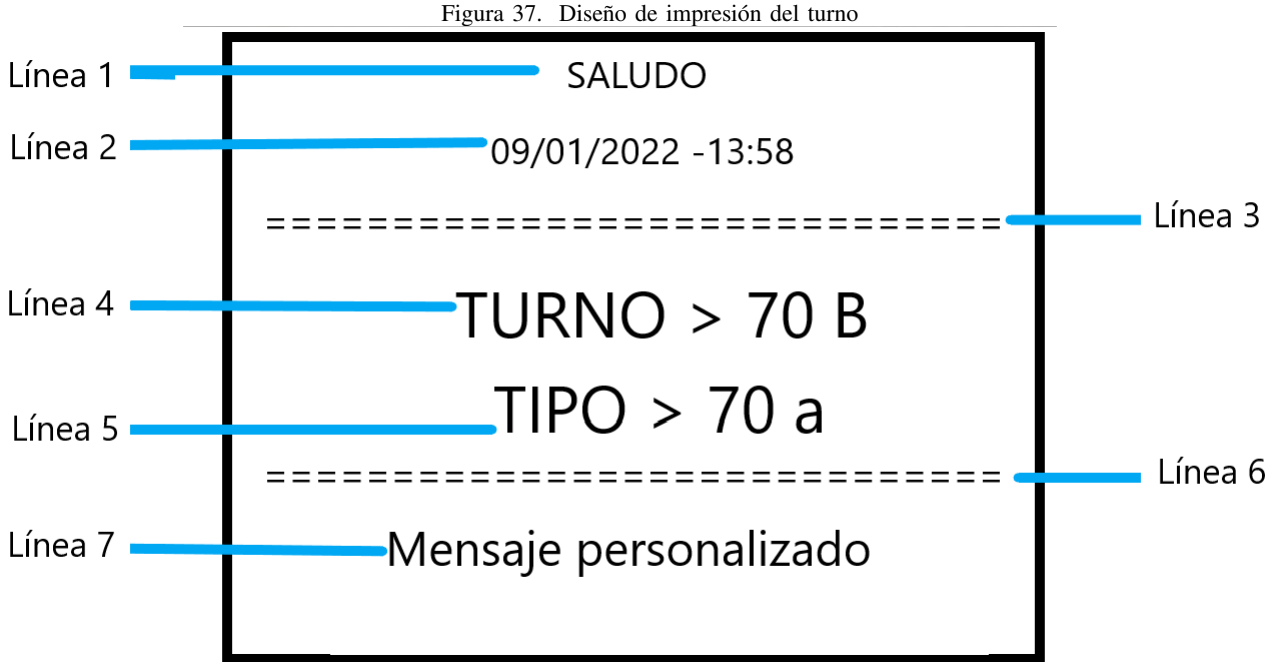
Figura 36. Diagrama de flujo Impresora.



IX-B4. Diseño del ticket de impresión: Para el diseño del ticket, se usará un diseño estándar que suele ser usar dentro del mercado ecuatoriano, donde se ofrecerá la información necesaria como es:

- Línea 1 Mensaje de saludo.
- Línea 2 La fecha y la hora en que se imprimió el turno.
- Línea 3 Separador.
- Línea 4 Número de turno con su Letra mayúscula que identifica la serie del ticket
- Línea 5 El tipo de atención con su letra minúscula que identifica el módulo de servicio.
- Línea 6 Separador
- Línea 7 Mensaje personalizado

Figura 37. Diseño de impresión del turno

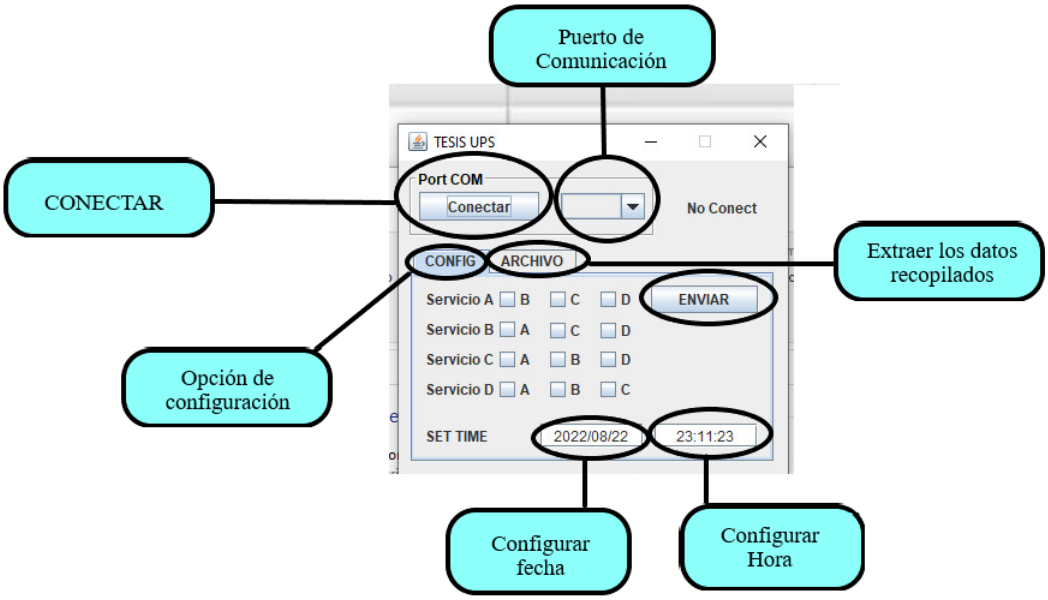


Esta configuración es propia de la impresora Epson, mientras que la tarjeta se comunica por el puerto paralelo con el objetivo de enviar los datos de turno, fecha y hora para su posterior impresión.

IX-B5. Diseño del programa de Configuración y recolección de datos:

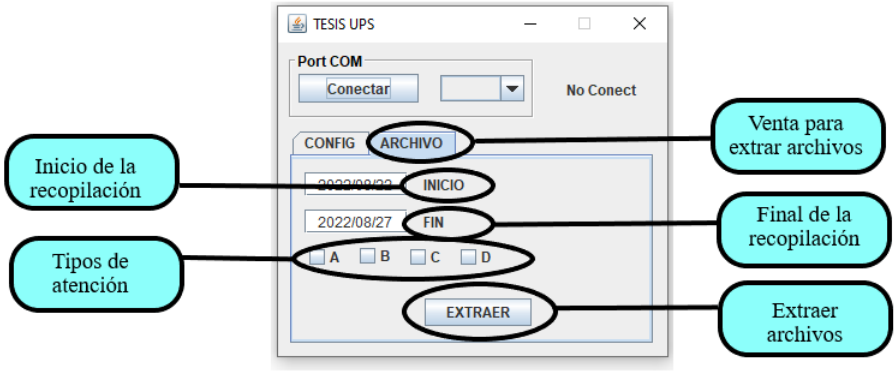
- Ventana de configuración
  Para la ventana de configuración se diseñó se toma los siguientes datos:
  - Conectar: Configura los datos de manera alámbrica desde una computadora que tenga instalado Java.
  - Puerto de comunicación: Permite configurar el puerto de salida de comunicación por la cual se enviará los datos hacia la placa controladora.
  - Opción de configurar: Escoger la venta de configuración por la cual se puede permitir el tipo de atención entre simple o dinámico, para los cual se puede setear si el operador A podrá atender también los requisitos de B, C o D y así sucesivamente.
  - Set timé: Permite configurar la hora y la fecha, la cual es necesaria que esté actualizada para que el ticket se imprima con los datos correctos.
  - Enviar: Luego de revisar que los datos de configuración sean los deseados, se podrá subir a la placa controladora al presionar el botón de envío.

Figura 38. Ventana del programa para configuración



- Ventana para extracción de los datos recopilados
  Con la finalidad de hacer un programa amigable para el usuario, se diseñó la ventana de Archivo que se encargara de extraer los datos recopilados durante la jornada de trabajo, este programa funciona de la siguiente manera:
  - Archivo Seleccionar esta opción proporciona el ingreso a la venta para extracción de datos recopilados.
  - Inicio: Concede al usuario Administrador seleccionar la fecha desde el cual deseamos obtener los datos.
  - Fin: Concede al usuario Administrador seleccionar la fecha desde el cual deseamos obtener los datos.
  - Tipos de atención: El programa permite al Administrador obtener ya sea los datos por modo de atención o extraerlos todos al mismo tiempo, al seleccionarlos dándoles clic.
  - Extraer: Una vez configurado los parámetros antes mencionados, se podrá extraer los datos de la manera que usuario Administrador mejor le parezca, cabe mencionar que estos datos se presentaran en formato Excel.

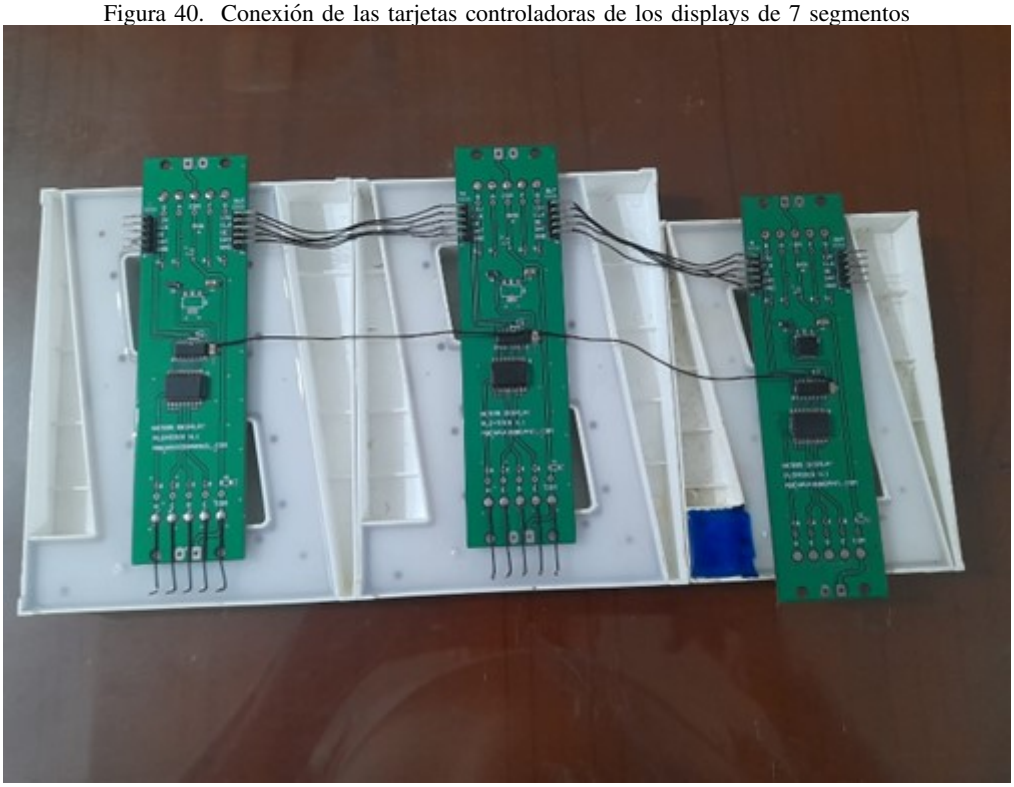
Figura 39. Ventana de recopilación de datos

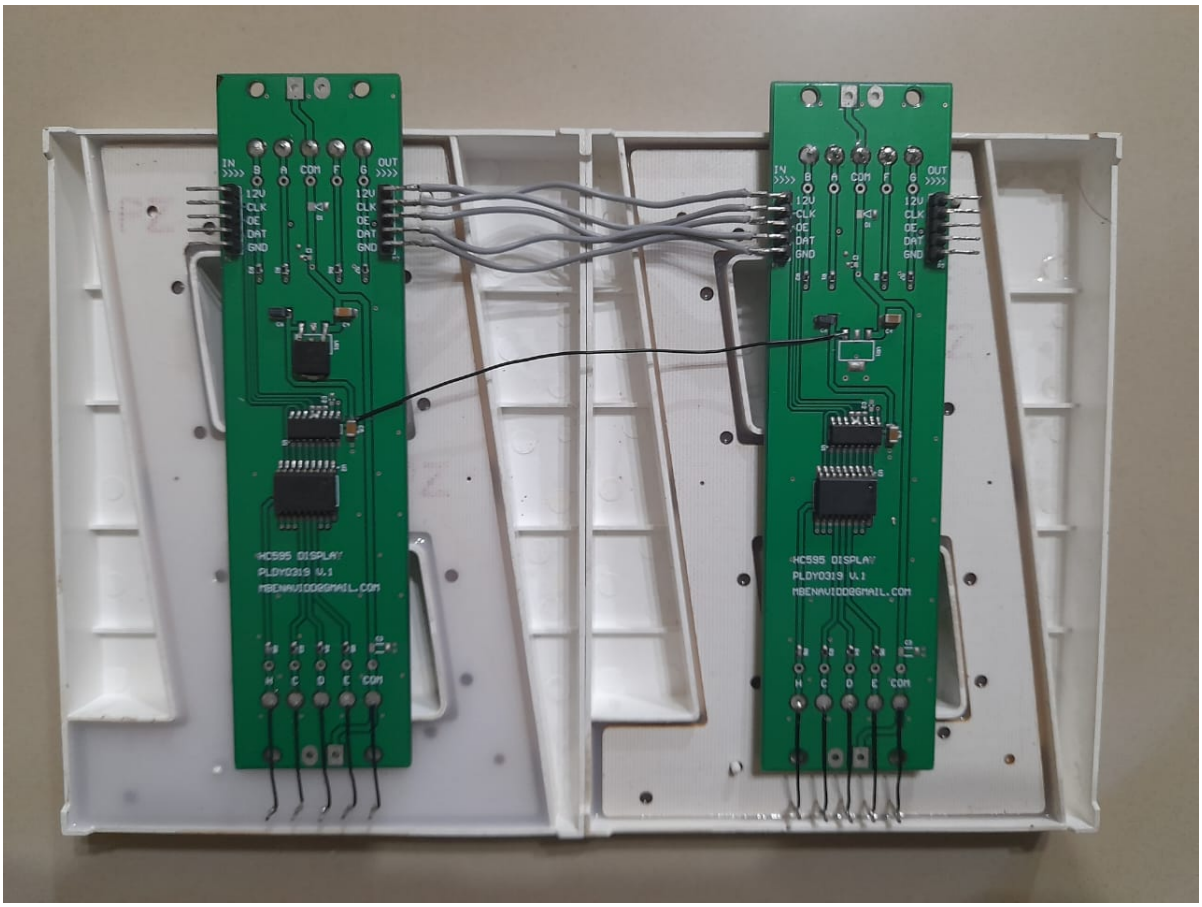


X. Implementación

Para iniciar la implementación, se empieza soldando los cables eléctricos que sirven como ruta para el envío de la información entre los displays de 7 segmentos y la tarjeta controladora, como se muestra en la ”Figura 40”, se debe recordar que para el proyecto se usan 5 módulos para visualizar los datos hacia el cliente.

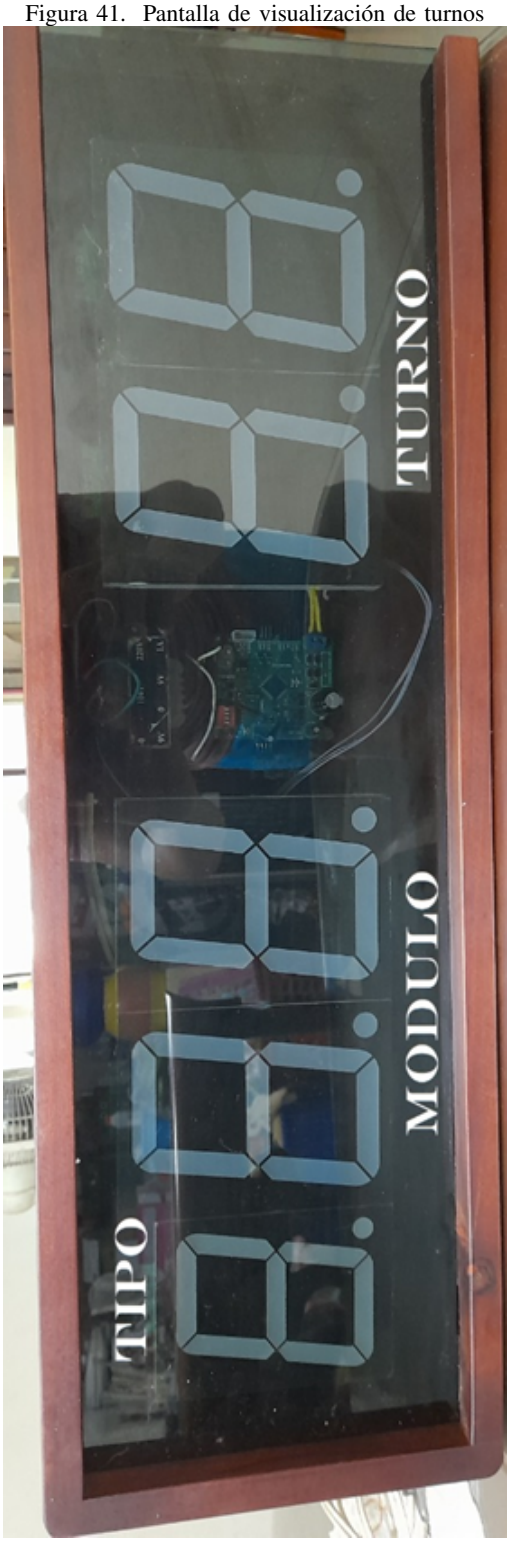
Figura 40. Conexión de las tarjetas controladoras de los displays de 7 segmentos





En el caso de la pantalla se envió a fabricar un case de madera de dimensiones 70 cm de largo x 15 de profundidad x 23 de ancho que incluye vidrio oscuro, el mismo que tiene como finalidad intensificar los segmentos iluminados por el display, de tal manera que se pueda observar el turno a una distancia más pronunciada de manera clara.

Figura 41. Pantalla de visualización de turnos



Con la finalidad de proteger a la impresora de posibles manipulaciones, se envió a construir el stand con altura de 105 cm para estar al mismo nivel de la mano de personas de estatura promedio con el objetivo de facilitar el manejo, con dimensiones de 20 cm x 21 cm para el case de la impresora, además, se consideró la madera por tratarse de un material elegante a la vista y de bajo costo.

Figura 42. Ubicación de la impresora en el módulo de atención

Luego de discutir los requerimientos que desea mostrar el Administrador de la Cooperativa de Taxis 9 de Mayo para los mensajes que mostrara el ticket de atención, se procedió a diseñar los textos, los cuales se deben ser ingresados en la impresora.

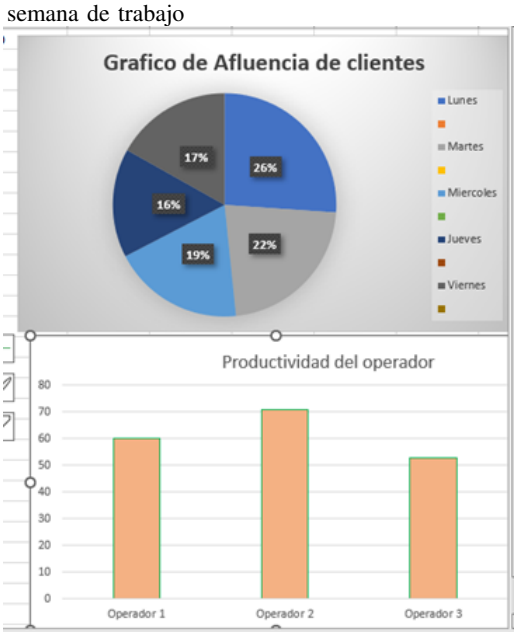
XI. RESULTADOS

En este capítulo se presentan los resultados producto de las experiencias que se tomó luego de implementar el proyecto en un local de atención al cliente de la cooperativa de taxis 9 de octubre de la ciudad de Machala [4]señala: "Consiste en exponer el significado de los hallazgos obtenidos (...)". (p. 139), dado que, para este primer paso, el proceso es informativo, se tomó los datos recolectados durante la semana de trabajo.

Al utilizar la aplicación Excel se procesaron los datos en brutos obtenidos del sistema, para presentarlos en formatos de porcentaje y cuadros estadístico, con el fin de darle un sentido a la Big Data recolectada, la cual es referenciada por [11] citando a [4] como "(...) un conjunto de técnicas y medidas que permiten caracterizar y condensar los datos obtenidos en forma de tablas y gráficos (...)". (p. 81).
A continuación, se presentan la información recolectada durante la semana de trabajo en la Cooperativa de Taxis 9 de octubre, a través de gráficos y cuadros estadísticos, que permitirán al administrador tener un mejor entendimiento del trabajo realizado por su personal de atención al cliente, que cuenta con un total 3 personas y 2 tipos de atención como lo son; atención para Socio y Consultas generales.

Figura 43. Datos de obtenidos durante la semana de trabajo

Operador	Lunes		Martes		Miercoles		Jueves		Viernes		TOTAL DE ATENCION
	S	G	S	G	S	G	S	G	S	G	
Operador 1	10	10	8	5	6	2	5	5	3	6	60
Operador 2	6	5	13	8	9	5	4	6	7	8	71
Operador 3	5	12	5	2	8	5	6	3	3	4	53
Total por atencion	21	27	26	15	23	12	15	14	13	18	184
Total por dia	48		41		35		29		31		
Total atenscion Socios	98										
Total atencion General	86										
Porcentaje Socios	53.2609										
Porcentaje General	46.7391										



Análisis: Según se observa en el cuadro 1 del gráfico 1, el 53.2 % de los clientes se dirigieron para la atención a socios, representando de esta manera solo 46.8 % la atención general, además se puede analizar en el segundo recuadro que el operador número 2 tuvo una mayor productividad que los demás, al haber atendido una mayor cantidad de clientes.

Por otro lado, se observa en el segundo recuadro en el cual se observa los días en que más fue visitado las oficinas de atención al cliente y el tiempo de espera promedio que se registró durante la semana laboral, en el cual se puede observar que es el día lunes, es el de mayor afluencia de la semana.

XI-A. Visualización del turno

Al ejecutar el inicio del programa la visualización de los dígitos se muestran claramente, luego a medida que llegan los usuarios, el personal de atención al cliente manipula el control, con el objetivo de llamar a la persona

que se encontraba en espera, se les solicitó que trataran de presionar varios botones al mismo tiempo del control de mando, el cual previamente estaba configurado para turno dinámico, con el propósito de conocer los posibles errores.

Al presionar varios botones en el mismo tiempo, el programa no cometía errores, ya que, al estar configurado correctamente, condiciona la prioridad al llamar al cliente en espera, teniendo como restricción el tipo de atención que efectúa cada operador, por ello, aunque el mismo operador tratara de llamar a un cliente de diferente atención a la suya, esta no lo hacía.

Figura 44. Módulo de atención

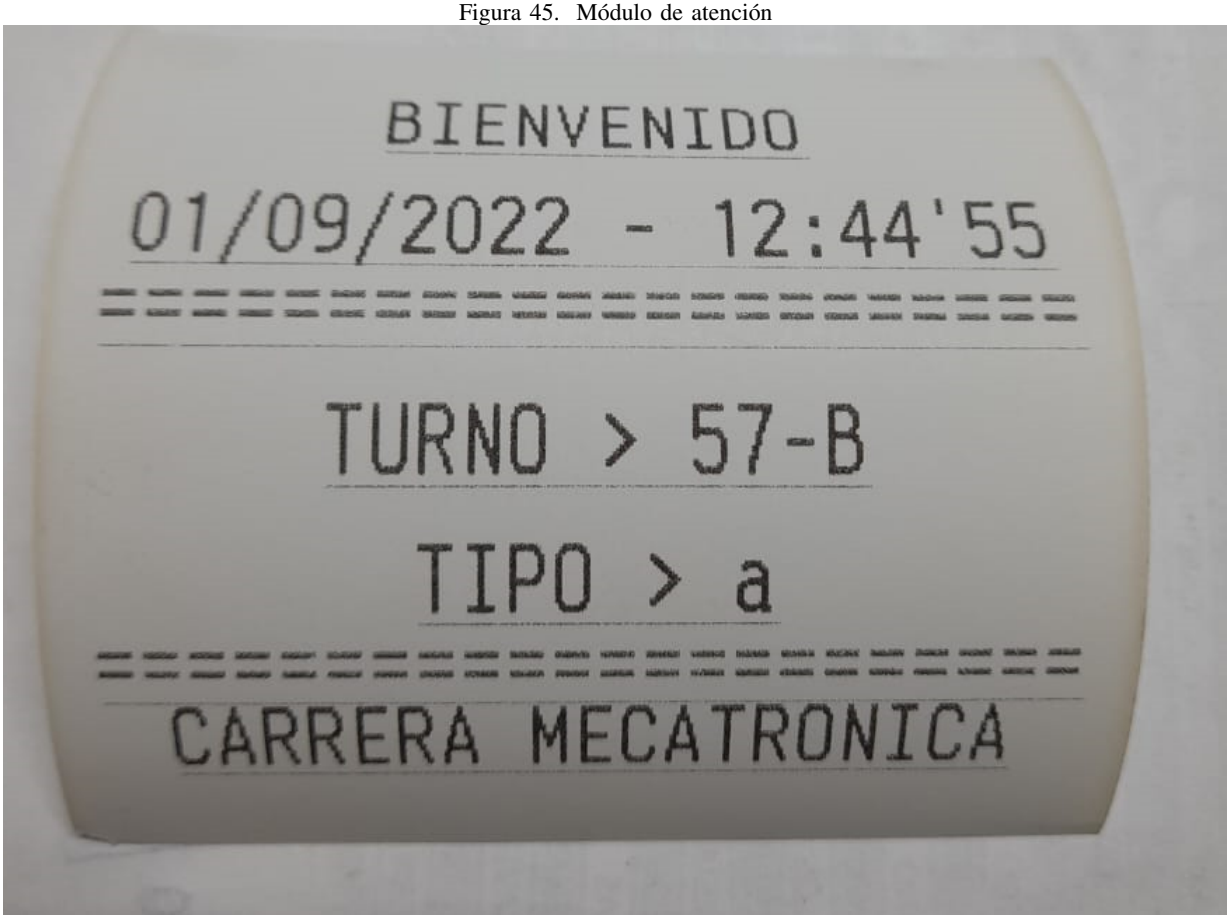
Análisis: Al iniciar el día el personal debía reiniciar el tablero de manera manual para que los turnos empezaran desde el 0, con el fin de realizar pruebas no se configuró el RESET de manera automática al finalizar el tiempo de trabajo, por ello al día siguiente se mantenía los dígitos del último cliente atendido.

Durante la semana de trabajo no hubo fallas en la visualización de los turnos, el cliente al llegar elegía entre los dos tipos de atención, el haber cambiado los displays de 7 segmentos por un más grande, fue la decisión correcta, ya que permiten diferenciar los dígitos a larga distancia, con el mismo objetivo, el fondo oscuro cumplió su cometido.

XI-B. Turno Impreso

El cliente al llegar al módulo de impresión de ticket verifica en primer lugar el turno que se visualiza por la pantalla, para conocer cuántos turnos estaban antes que él, luego de presionar el tipo de atención, se imprime el ticket de manera inmediata cabe mencionar que se agregó una alerta auditiva para señalar la finalización de esta etapa.

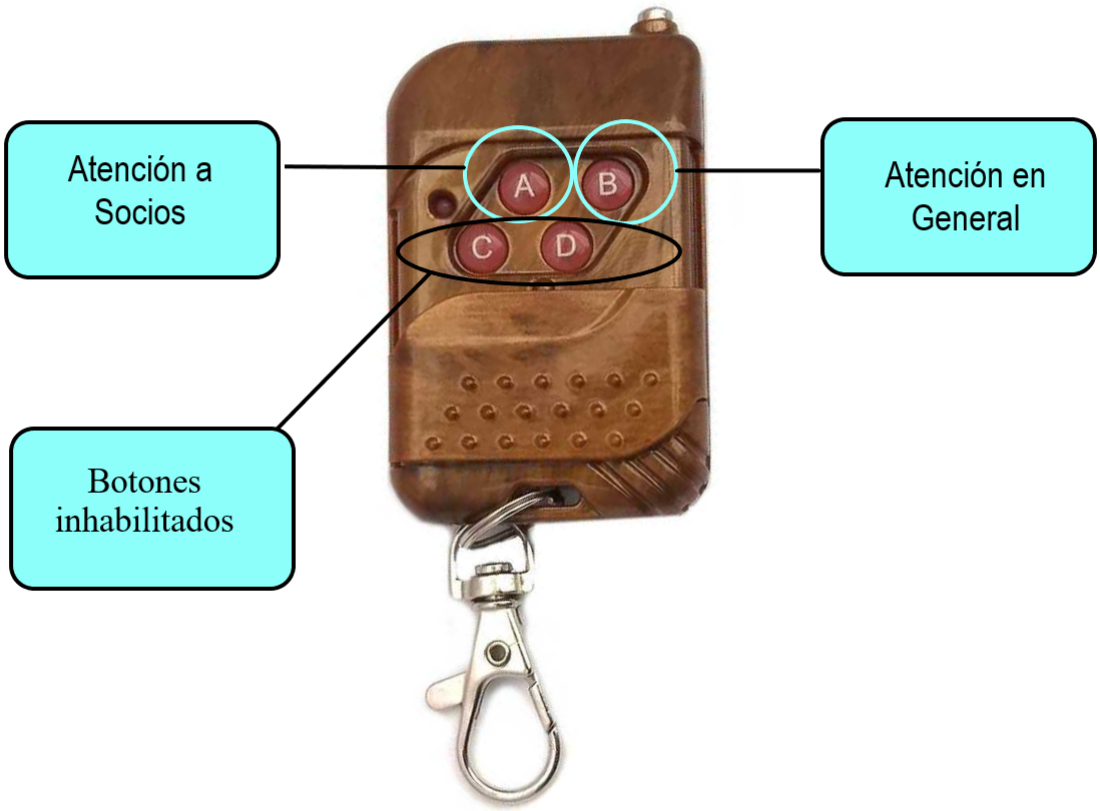
Figura 45. Módulo de atención



Análisis: Al no contar con el tiempo de espera impreso en el ticket por solicitud del gerente, se pudo observar un poco de desconformidad por parte de los clientes, ya que no sabían qué tiempo tardaría hasta ser atendidos, muchos de ellos, salieron para realizar otra actividad y luego regresar, de conocer el tiempo de espera el cual era corto, seguramente podemos intuir que habría esperado su atención antes de realizar otra diligencia.

Por ello este dato debe ser impreso en cada ticket y deberíamos dejar una línea extra para que se pueda agregar algún mensaje de agradecimiento o eslogan por parte de la empresa, esto es interesante, dado que hasta se podría promocionar ciertos servicios o paquetes dentro del mismo turno impreso.

Figura 46. Configuración de los botones para el uso del Personal



Se configura el botón A para la atención a socios y el botón B para la atención en general, de estas dos opciones el personal atiende a los socios que integran la nómina de la cooperativa de Taxis, para ellos la atención debe ser preferencial, por este motivo los 3 operadores pueden llamar a los socios como prioridad.

Para evitar errores por presión de botón se inhabilitó los pulsadores correspondientes al C y D, el personal de atención comento que sentían los pulsadores eran muy pequeños, por otro lado, los controles funcionaron sin errores al momento de realizar la interacción con la pantalla para llamar al cliente.

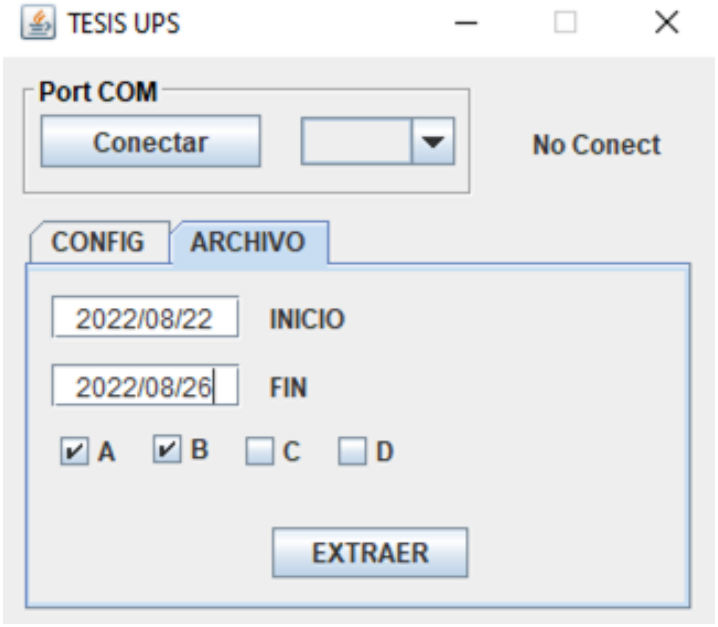
Analisis: Cabe mencionar que los controles son para el prototipo y que al tratarse de una comunicación 2.4 GHz se podría usar otro tipo de pulsadores más grandes para solucionar la queja del personal de atención, dejando de lado este inconveniente, se tomó varias observaciones, entre una de ellas es que el control de llamado del cliente

debe estar fijo a la mesa del operador, de esta manera no se pierde de vista y evita posibles caídas.

Con el objetivo de analizar el tiempo de vida de la batería de los controles, es necesario realizar un tiempo de prueba más extenso, así como el tiempo de vida útil de los pulsadores, dejando de lado esto, el sistema, al tratarse de bajo costo, cumple con los requerimientos que se presentó al inicio del proyecto, por ello estamos satisfechos.

XI-D. Programa de recolección de datos – Big Data

Figura 47. Ventana de recolección de datos



Luego de que se extraen los datos en bruto desde la ventana de recolección, como se muestra en la ”Figura 47”, se debe realizar la visualización mediante el uso de gráficos, aplicando fórmulas de Excel, como se muestra en la ”Figura 43”. Esto permite su mejor entendimiento, esta operación no es realizada por el programa, debe ser realizada por el administrador.

Análisis: Los datos recopilados en bruto y presentados por Excel ofrecen una manera fácil de entender, el modo de trabajo del Personal, aun así, ser presentada mediante gráficos como se mencionó anteriormente, ayudara a una comprensión más profunda, el grupo tratará de que el sistema realice este diseño de manera automática en una futura actualización.

En los datos recopilados se puede observar una clara ventaja del Operador número 2, ante sus compañeros, ya que ejecuto la atención de manera rápida y eficaz los requerimientos de los usuarios, por consiguiente la gráfica también permite intuir que el Operador número 3, necesita ser capacitado con el fin de mejorar su tiempo de atención.

XII. DIAGRAMAS

Figura 48. Diseño y diagrama de tarjeta del Display

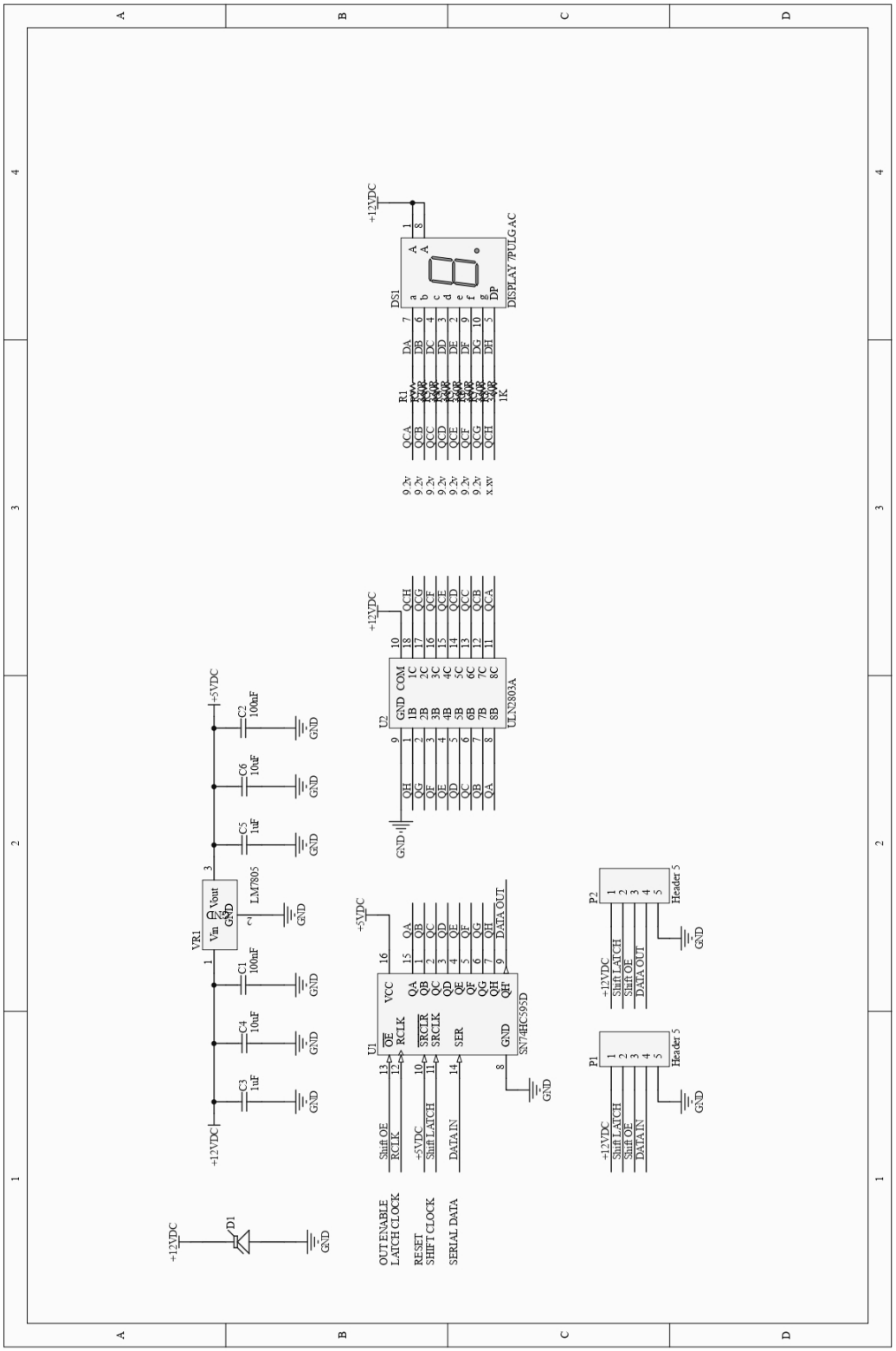


Figura 49. Diseño y diagrama de tarjeta de la pantalla

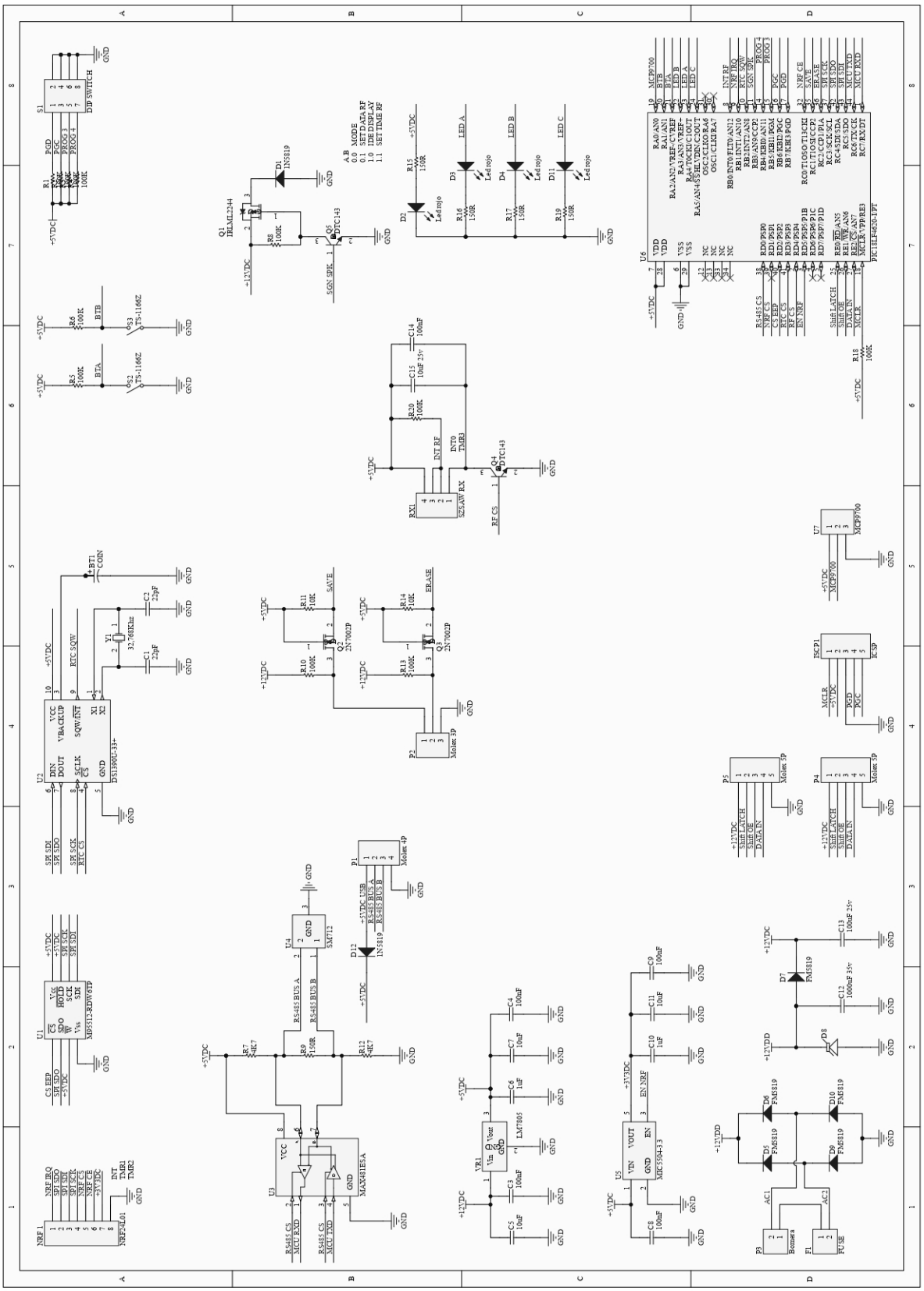


Figura 50. Diseño y diagrama de la tarjeta de la Impresora

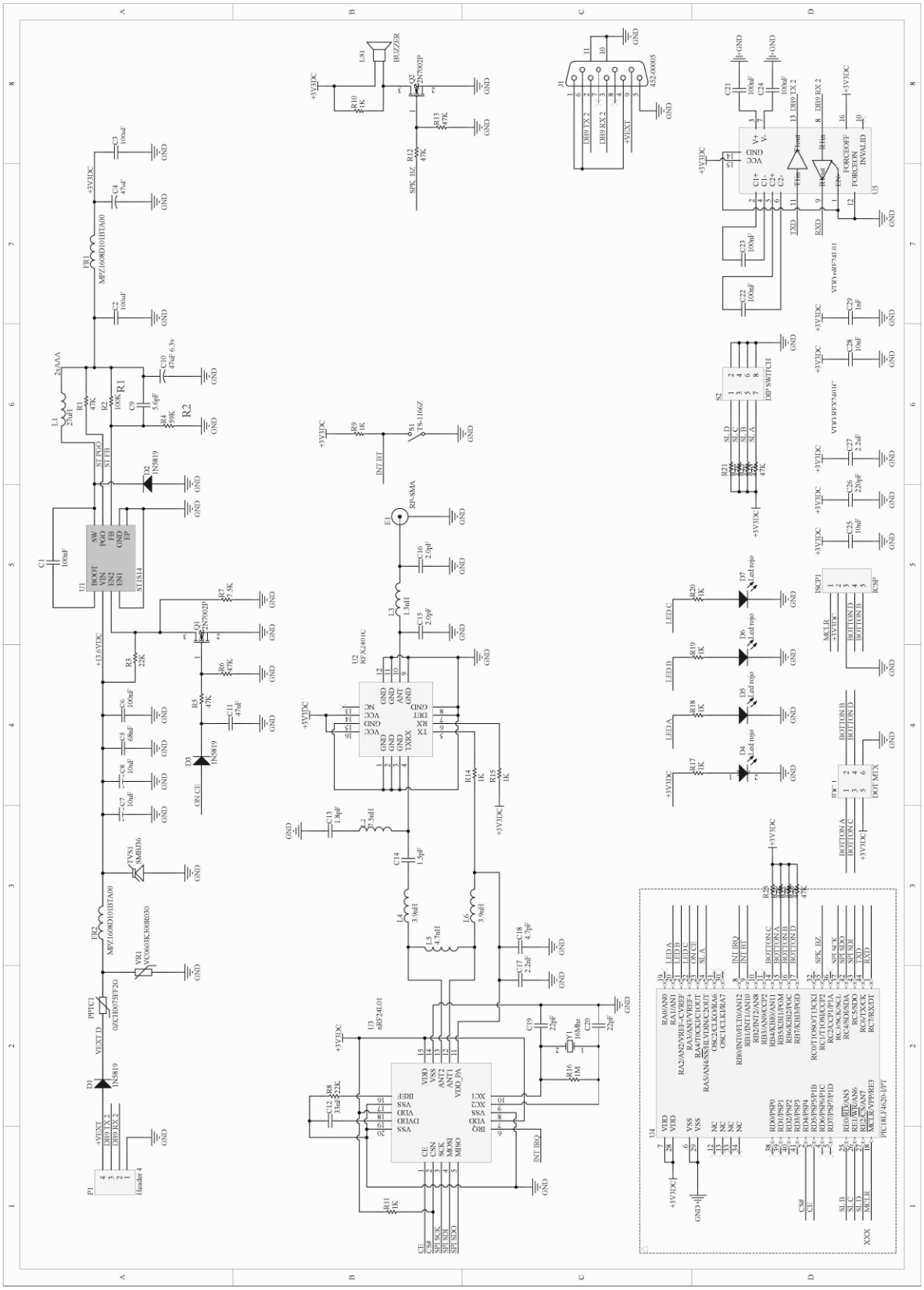
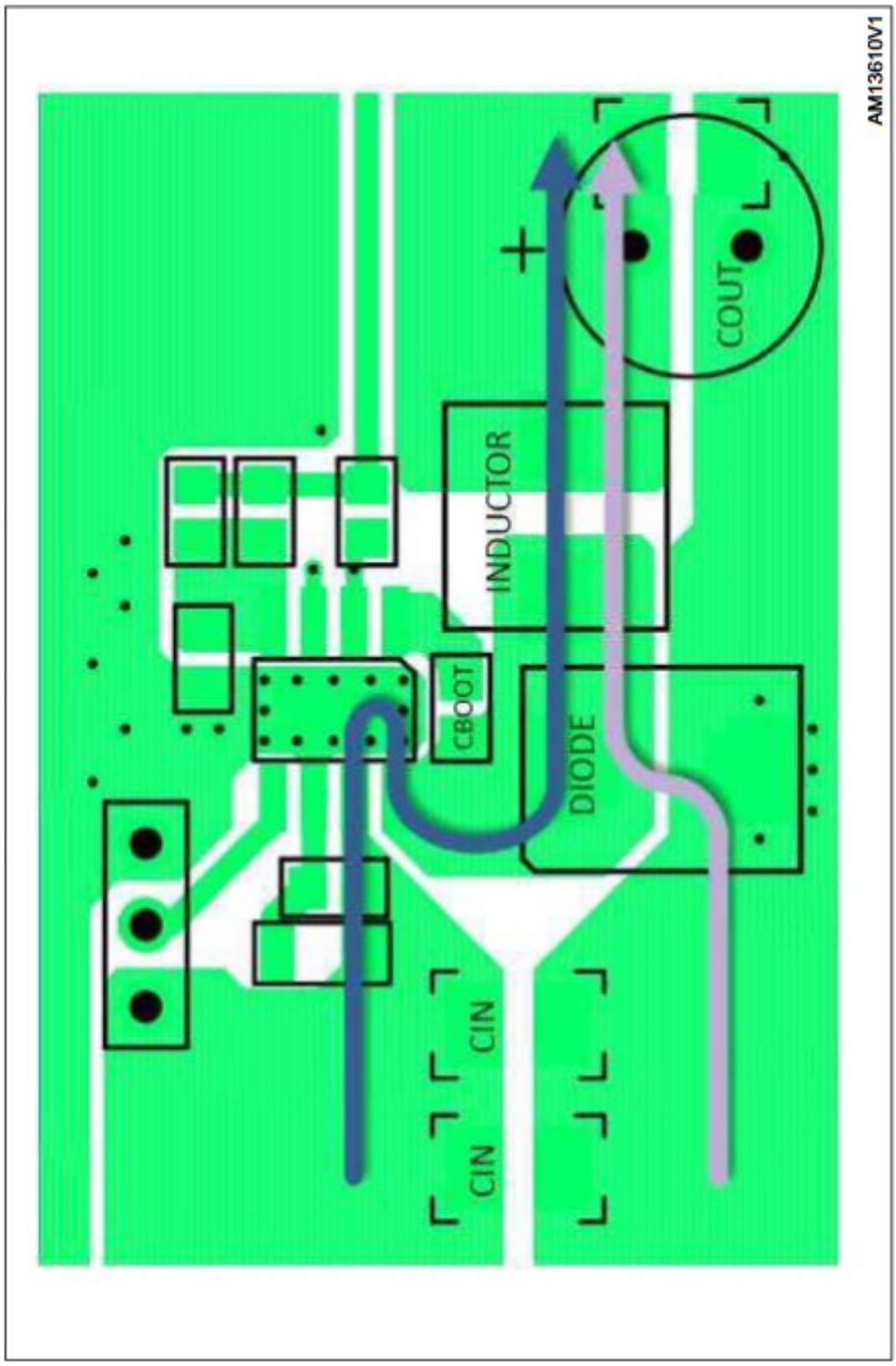


Figura 51. Diseño y Orientación de la fuente sugerida por el Fabricante Chino



XIII. Cronograma de actividades

(Sánchez, 2015) “Un cronograma de actividades es simplemente un calendario en el que estableces los tiempos en los que realizaras el proyecto, una tarea, o un conjunto de actividades a trabajar o desarrollar”, las actividades detallas en la ”Tabla XVI”, son las actividades que realizo el grupo con respecto al proyecto actual.

Cuadro XVI
Cronograma de actividades

Nombre del proyecto	Sistema de Ticket				
Encargado del proyecto	Alberth Hurtado				
Encargad del proyecto	Milton Benavides				
Fecha de inicio	5-mar				
Fecha final	3-jul				
Avance general	90%				
Tareas	Responsable	echa de inic	Fecha final	Días	Estado
Eleccion del tema					
Revision de temas	Alberh H	5/2	5/3	1	Completado
Eleccion del tema	Milton B.	2-may	4-may	4	Completado
INICIO					
Presentacion del tema de proyecto	Alberth H.	10-may	11-may	1	Completado
Compra de componentes en aliexpress	Milton B.	15-may	16-may	1	Completado
DESARROLLO DE LOS TEMAS:	Alberth H.	15-may	30-ago	107	Completado
Elaboracion Objetivos	Alberth H.	17-may	17-may	0	Completado
Elaboracion del Planeamiento del Problema	Alberth H.	18-may	19-may	1	Completado
Elaboracion Justificacion del proyecto	Alberth H; Milton B.	19-may	20-may	1	Completado
Elaboracion Marco Teorico	Alberth H; Milton B.	21-may	25-may	4	Completado
Busqueda de Referencias	Milton B.	21-may	25-may	4	Completado
Elaboración de base de dato, Articulos	Alberth H.	26-may	27-may	1	Completado
DESARROLLO DEL PROYECTO	Alberth H.	28-may	25-jul	58	Completado
Analisis de presentancion en patanlla	Milton B.	29-may	31-may	2	Completado
Analizar la programacion de la pantalla	Milton B.	5-jun	7-jun	2	Completado
Desarrollar el pulsador para los operadores	Milton B.	10-jun	12-jun	2	Completado
Importacion de equipos a emsamblar	Milton B.	11-jun	13-jun	2	Completado
Implementar la comunicación entre el controlador y la impresora	Alberth H.	14-jun	15-jun	1	Completado
Probar los turnos impresos	Milton B.	17-jun	18-jun	1	Completado
Implementar el tiempo de espera	Milton B.	20-jun	22-jun	2	Completado
DESARROLLO TT1	Alberth H.	8-jun	22-jun	14	Completado
Trabajo consolidado	Alberth T.	25-jun	27-jun	2	Completado
Analisis del proyecto	Milton B.	28-jun	30-jun	2	Completado
Pruebas de funcionamiento y corrección de errores	Milton B.	3-jul	7-jul	4	Completado
Importacion de equipos a emsamblar	Milton B.	15-jul	18-jul	3	Completado
Viaje a Guayaquil para reunion	Milton B; Alberth H.	21-jul	21-jul	0	Completado
Elaboracion Cronograma de actividades	Milton B.	24-jul	25-jul	1	Completado
Creacion del Presupuesto	Milton B.	25-jul	26-jul	1	Completado
Reunion de la Primera revision	Alberth H; Milton B.	26-jul	27-jul	1	Completado
Correcciones	Alberth H.	27-jul	28-jul	1	Completado
Reunion de la Segunda Revision	Alberth H; Milton B.	28-jul	29-jul	1	Completado
Correccion final del proyecto	Milton B.	29-jul	30-jul	1	Completado
Entrega del proyecto	Alberth H.	3-sep	15-sep	12	Completado

XIV. Presupuesto

En cualquier investigación es muy importante tener en cuenta los recursos económicos que necesita el proyecto para materializarse, por ello el presupuesto incluye todos los valores que permitieron la fabricación del prototipo, así como, el costo de fabricación para su posterior venta.

"Dirección"
"Ciudad"
"Provincia" "Codigo Postal" 59396
"Telefóno" "Pagina Web"

Fecha de Cotizacion	19-07-21
Grupo	7
Tema	Presupuesto
Metodo de Pago	Efectivo
Terminos de Pedido	

olicitado po

Producto	Descripción	Unidades	Precio / unidad	Precio
1	Placa Control principal	4	$ 34.26	$ 137.04
2	Placa Digitos de 8"	12	$ 34.35	$ 412.20
3	Placa Impresora	2	$ 68.97	$ 137.94
4	Impresión PCB Control principal	12	$ 3.50	$ 42.00
5	Impresión PCB Placa digitos	12	$ 4.10	$ 49.20
6	Impresión PCB Impresora	12	$ 5.20	$ 62.40
7	Control RF 433 Mhz	20	$ 20.00	$ 400.00
8	Pasta para soldar	2	$ 5.00	$ 10.00
10	Rollo de estaño	5	$ 5.00	$ 25.00
11	Periféricos USB a Serial	1	$ 30.00	$ 30.00
12	Adaptador de voltaje 12v 2 Amp	3	$ 5.00	$ 15.00
13	Antena RF 2,4 Ghz	3	$ 8.00	$ 24.00
14	Cautin Proski	3	$ 16.00	$ 48.00
17	Compilador MicroBase	1	$ 250.00	$ 250.00
18	Transporte	1	$ 200.00	$ 200.00
19	Salarios	2	$ 400.00	$ 800.00
21	Impresora Epson TM-20II	1	$ 200.00	$ 200.00
22	Diseño de placa electronica	1	$ 150.00	$ 150.00
23	Gastos basicos	1	$ 200.00	$ 200.00
24	Materiales de oficina	1	$ 100.00	$ 100.00
25	Viaticos	1	$ 80.00	$ 80.00
26	Gastos Varios	1	$ 400.00	$ 400.00
			Subtotal	$ 3,772.78
		12.00%	IVA	$ 452.73
			Costes de Envio	
			Seguro	
			Total	**$ 4,225.51**

Figura 52. Presupuesto del prototipo.

Con el fin de la recuperación de la inversión inicial, es necesario la venta de al menos 4 kits de generador de turnos por el valor $1800 dólares estadounidenses cdu, sabiendo que a partir del 5to kit el equipo ganara un valor aproximado de $1000 dólares estadounidenses.

"Dirección"
"Ciudad"
"Provincia" "Codigo Postal" 59396
"Telefóno" "Pagina Web"

Fecha de Cotizacion	19-07-21
Grupo	7
Tema	Presupuesto
Metodo de Pago	Efectivo
Terminos de Pedido	

olicitado po

Producto	Descripción	Unidades	Precio / unidad	Precio
1	Placa Control principal	4	$ 34.26	$ 137.04
2	Placa Digitos de 8"	12	$ 34.35	$ 412.20
3	Placa Impresora	4	$ 68.97	$ 275.88
4	Impresión PCB Control principal	12	$ 3.50	$ 42.00
5	Impresión PCB Placa digitos	12	$ 4.10	$ 49.20
6	Impresión PCB Impresora	12	$ 5.20	$ 62.40
7	Control RF 433 Mhz	20	$ 20.00	$ 400.00
8	Pasta para soldar	2	$ 5.00	$ 10.00
9	Rollo de estaño	5	$ 5.00	$ 25.00
10	Periféricos USB a Serial	4	$ 30.00	$ 120.00
11	Adaptador de voltaje 12v 2 Amp	3	$ 5.00	$ 15.00
12	Antena RF 2,4 Ghz	3	$ 8.00	$ 24.00
13	Cautin Proski	3	$ 16.00	$ 48.00
14	Impresora Epson TM-20II	4	$ 200.00	$ 800.00
			Subtotal	$ 2,420.72
		12.00%	IVA	$ 290.49
			Costes de Envio	
			Seguro	
			Total	**$ 2,711.21**

Figura 53. Presupuesto de fabricación de 4 kits

XV. Conclusiones

Se logra cumplir con el objetivo de crear un sistema para la asignación de turnos de bajo costo, al realizar una inversión de $2711.21 dólares para la fabricación de 4 kits y un valor unitario de $677.75 dólares, mientras que en el resto del mercado los más económicos oscilan entre los $4000 dólares, lo cual permite a la empresa acceder a un dispositivo de calidad sin tener que realizar una gran inversión.

La programación que se realizó logra que los componentes interactúen de manera ordenada, rápida y eficiente, usando componentes que ya se encontraban en las tiendas de electrónica. Además, la programación concede que el sistema sea adaptable a los diferentes requisitos o lugares de trabajo en los que se pretender usar, siendo esto significativo porque permite vender el producto en varios sectores comerciales.

El formato de presentación de los datos recopilados de las jornadas de trabajo otorga al usuario Administrador, evaluar de manera concisa la eficiencia de trabajo de sus empleados, con la finalidad de emitir reportes, teniendo en mente el objetivo de mejorar el servicio de atención al cliente, lo cual ofrece a la empresa la posibilidad de crecer sin dejar al consumidor desatendido.

Cabe resaltar que al tener la posibilidad de archivar un gran número de clientes (13200 usuarios) concede al usuario Administrador realizar reportes de manera esporádica, antes de que se empiece a borrar el primero de la lista registrado.

El uso adecuado de la Big Data es necesario para entender cómo se mueve el mercado, y de qué manera podemos aprovechar esa corriente para crecer, ya que las empresas que se adaptan son las que sobreviven, ya lo dijo Charles Darwin “No es la especia más fuerte la que sobrevive, ni la más inteligente, sino la que responde mejor al cambio”.

XVI. RECOMENDACIONES

Con el propósito de reducir los costos de producción, se recomienda importar los componentes necesarios para la construcción de al menos 5 kits, ya que, la empresa JLCPCB donde se envía a fabricar las tarjetas controladoras de procedencia China, la cual permite la compra mínima de 5 tarjetas electrónicas por diseño.

Aunque el sistema permita un alto número de registro de clientes, se recomienda que el primer mes de funcionamiento, el usuario Administrador realice respaldos de los archivos desde el programa de configuración de al menos una vez por semana, con la finalidad de conocer el promedio de visitantes semanales con el objeto de saber la regularidad con la que debe realizar los respaldos a futuro.

Se recomienda que los reportes emitidos por el usuario Administrador sean tomados en cuenta con la finalidad de mejorar el rendimiento de los operadores, pero esto debe caminar junto con capacitaciones adecuadas por parte de la empresa, con el objetivo de mejorar el servicio que se entrega a los clientes.

Para el regulador de la fuente, la empresa JLCPCB recomienda y enfatiza que; la construcción de la fuente tenga el sentido de orientación como lo muestra el .Anexo 51", ya que al tratarse de una fuente conmutada que incorpora switch, debe de mantenerse los parámetros y orientación de los componentes.

Con el fin que el sistema funcione apropiadamente, se recomienda realizar mantenimiento de los componentes al menos 2 veces al año con el objetivo de extender su vida útil, debido a que, al ser pulsadores estos se llenaran de grasa proveniente de la interacción común con los clientes, lo cual podría traer problemas al momento de usar el módulo de atención o los controles.

La luz del display debe ser regulable con la finalidad de que se pueda adaptar a la iluminación propia de cada oficina, ya que, de no ser así, no se podrá distinguir de manera correcta los turnos que este marque.

REFERENCIAS

[1] V. E. Aguilar Arcos, S. San Martín Gutiérrez, R. J. Payo Hernanz y col., «La aplicación empresarial del marketing viral y el efecto boca-oreja electrónico. Opiniones de las empresas,» *Cuadernos de Gestión, 2014, vol. 14, n. 1, pp. 15-31*, 2014.

[2] *Altium designer 6: An introduction*. Altium, 2005.

[3] L. Álvarez Torre, *El Big Data y el cambio en el Modelo de Negocio de las empresas de e-commerce : El Caso de Amazon y alibaba*, ene. de 2018.

[4] Arias y Fidias, *El proyecto de investigación*. 2012.

[5] Azuero. y Angel., «Significatividad del marco metodológico en el desarrollo de proyectos de investigación,» 2019.

[6] A. R. Castellano, «Bluetooth. Introducción a su Funcionamiento,» *Univ. Pontif. Comillas, Madrid*, vol. 1, n.º 1, págs. 1-16, 2012.

[7] A. C. Castro Triviño, *Perspectiva para la mejora del servicio al cliente en las empresas públicas de la ciudad de Guayaquil.* 2019.

[8] K. A. Charles y M. Sadiku, *Fundamentals of Electric Circuits*. McGraw-Hill, 2019.

[9] J. R. Cogdell, J. R. R. Francisco y T. C. Pérez, *Fundamentos de Electrónica*. Prentice Hall, 2000.

[10] Y. Franco, «La metodología de la investigación,» Universidad Autónoma de Occidente, 2011.

[11] E. Gallardo, *Metodologia de la investigacion. Manual autoformativo e interactivo*. 2017.

[12] D. W. Hart y A. B. Bautista, *Electrónica de potencia*. Prentice Hall Madrid, España, 2001, vol. 32.

[13] J. V. G. López, *COMT004PO-Fundamentos de atención al cliente*. Editorial Elearning, SL, 2020.

[14] K. Mitzner, *Complete PCB design using OrCAD Capture and PCB editor*. Newnes, 2009.

[15] C. V. Neira, R. V. Casielles y V. I. Argüelles, «Comportamiento de abandono de la relación de un cliente con la empresa en un contexto de fallo y recuperación del servicio,» *Cuadernos de Economía y Dirección de la Empresa*, vol. 12, n.º 40, págs. 143-169, 2009.

[16] O. Perez, «5 Razones de la importancia de la Gestión y Seguimiento de Clientes para el crecimiento de Tu Negocio,» *Test Nextup*, abr. de 2022. dirección: https://blog.nextup.com.mx/5-razones-de-la-importancia-del-servicio-al-cliente-para-el-crecimiento-de-tu-negocio/.

[17] X. Ramos, «La era del robot se instala a paso lento en Ecuador,» *El Universo*, 2018.

[18] P. S. Rojas, «Manual de uso del programa de diseño de circuitos y simulación Proteus Layout Editor,» *Centro de Automatización Industrial, SENA Regional Caldas, Colombia*, pág. 1, 2008.

[19] C. G. SELMA, «Adquisición de datos, supervisión y control de equipos de laboratorio por ordenador,» *Técnicas de análisis y caracterización de materiales*,

[20] J. R. Urresti Ibáñez, *Modelización y fabricación de dispositivos supresores TVS para protección en aplicaciones de baja tensión*. Universitat Autònoma de Barcelona, 2009.

[21] F. Valdés y R. P. Areny, *Microcontroladores fundamentos y aplicaciones con PIC*. Marcombo, 2007, vol. 1149.

[22] Waxoo, feb. de 2013. dirección: www.microcode-studio.waxoo.com.

Figura 54. Mapa conceptual de la interacción de los componentes del sistema

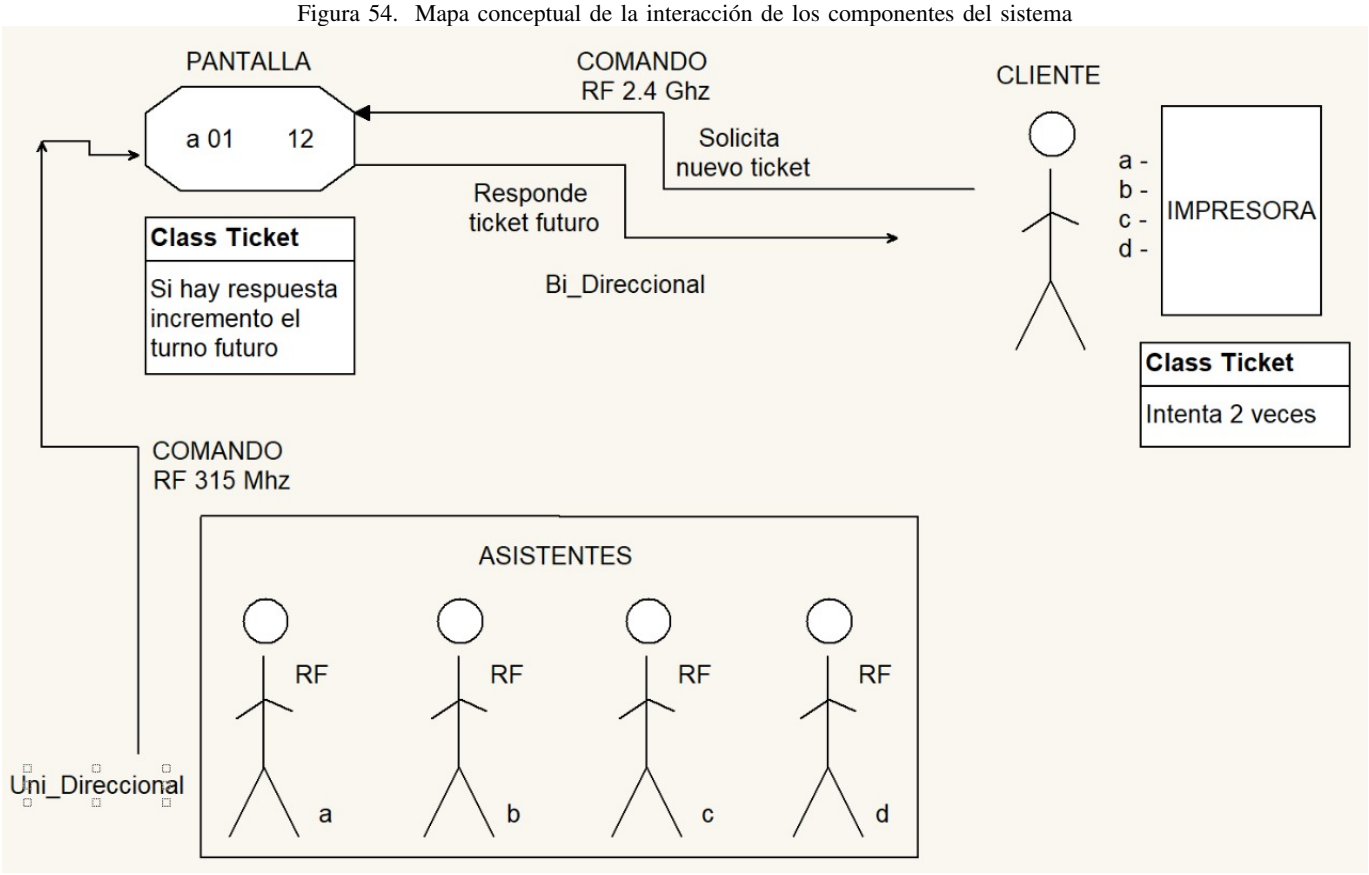
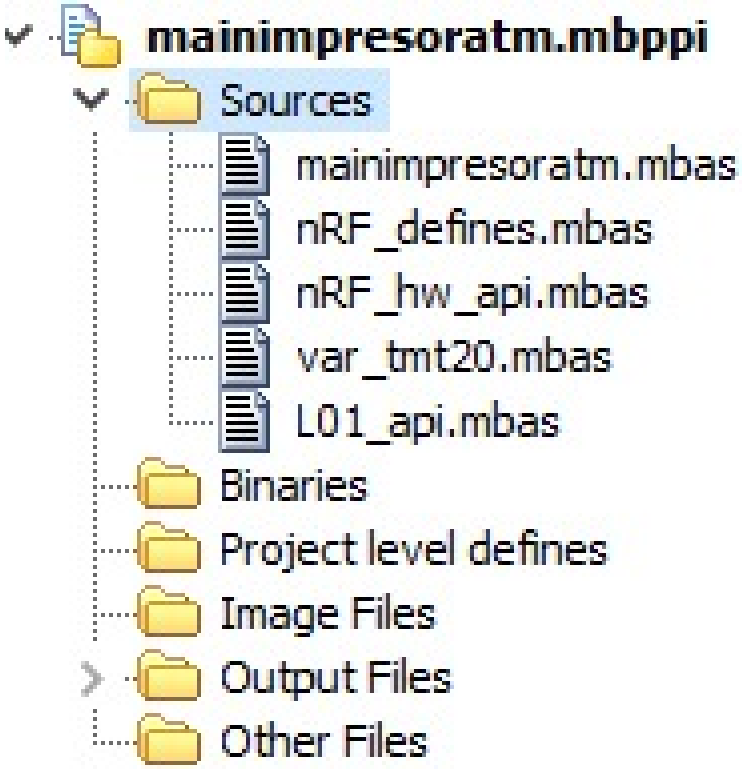


Figura 55. Programas para el funcionamiento del Display



XVII-A1. Código de programación de variables para el uso de la memoria EEPROM:

```
module M95512_EE

sub function M95512_Read(dim vaddr as word) as byte
sub procedure M95512_Write(dim vaddr as word, dim vdata as byte)
sub procedure M95512_INIT()
sub function M95512_READY() as byte

implements

dim CS_EEP as sbit sfr external

const ADR_M95_WREN  = %00000110
const ADR_M95_WRDI  = %00000100
const ADR_M95_RDSR  = %00000101
const ADR_M95_WRSR  = %00000001
const ADR_M95_READ  = %00000011
const ADR_M95_WRITE = %00000010
const ADR_M95_RDID  = %10000011
const ADR_M95_WRID  = %10000010
const ADR_M95_RDLS  = %10000011
const ADR_M95_LID   = %10000010

sub function M95512_Read(dim vaddr as word) as byte
   dim vdata as byte
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   'SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV64,_SPI_DATA_SAMPLE_END,_SPI_CLK_IDLE_HIGH,_SPI_LOW_2_HIGH)
'Delay_ms(50) CLRWDT
   while (M95512_READY()=0x00) wend
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   CS_EEP = 0
   SPI1_Write( ADR_M95_READ ) 'write
   SPI1_Write( hi(vaddr) ) 'write
   SPI1_Write( lo(vaddr) ) 'write
   vdata = SPI1_Read(0) '0 LETRA
   CS_EEP = 1
   Delay_ms(5)
   result = vdata
end sub

sub procedure M95512_Write(dim vaddr as word, dim vdata as byte)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   'SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV64,_SPI_DATA_SAMPLE_END,_SPI_CLK_IDLE_HIGH,_SPI_LOW_2_HIGH)
'Delay_ms(50) CLRWDT
   while (M95512_READY()=0x00) wend
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
   CS_EEP = 0
   SPI1_Write( ADR_M95_WREN )
   CS_EEP = 1
   Delay_ms(5)
   CS_EEP = 0
   SPI1_Write( ADR_M95_WRITE )
   SPI1_Write( hi(vaddr) )
   SPI1_Write( lo(vaddr) )
   SPI1_Write( vdata )
   CS_EEP = 1
   Delay_ms(5)
end sub

sub function M95512_READY() as byte
```

```
    dim vdato as byte
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV64,_SPI_DATA_SAMPLE_END,_SPI_CLK_IDLE_HIGH,_SPI_LOW_2_HIGH)
Delay_ms(50) CLRWDT
    Delay_ms(50)
    CS_EEP = 0
    SPI1_write( ADR_M95_RDSR )
    vdato = SPI1_read( 0X00 )
    CS_EEP = 1
    Delay_ms(5)
    if vdato.0 = 0 then
        result = 0xFF
    else
        result = 0x00
    end if
end sub

sub procedure M95512_INIT()
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    while (M95512_READY()=0x00) wend
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    CS_EEP = 0
    SPI1_Write( ADR_M95_WREN )
    CS_EEP = 1
    Delay_ms(5)
    CS_EEP = 0
    SPI1_Write( ADR_M95_WRSR )
    SPI1_Write( 0X00 )
    CS_EEP = 1
    Delay_ms(5)
    CS_EEP = 0
    SPI1_Write( ADR_M95_WREN ) 'write
    CS_EEP = 1
    Delay_ms(5)
end sub

end.
```

XVII-A2. Código de programación de variables para el uso del reloj:

```
module mod_ds1390

dim ds1390_time, ds1390_date as string[4]
dim ds1390_SQWOLD as byte

sub procedure ds1309_setinit(dim hourset, minuteset, secondset, milisecondset, dateset, dayset, monthset, yearset as
byte)
sub procedure ds1309_setreg(dim addr, vdata as byte)
sub procedure ds1309_readall(dim byref hourset as byte, dim byref minuteset as byte, dim byref secondset as byte, dim
byref milisecondset as byte, dim byref dateset as byte, dim byref dayset as byte, dim byref monthset as byte, dim byref
yearset as byte)
sub procedure ds1390_init()

sub function add_hora(dim byref ds1307_hour2 as short, dim ds1307_hour1 as short) as short
sub function add_minut(dim byref ds1307_minute2 as short, dim ds1307_minute1 as short) as short
sub function sync_time_utc(dim hourutc as short, dim byref hourset as byte, dim byref dayset as byte) as byte

implements

dim CS_RTC_DS1390 as sbit sfr external
dim INT_SWQ as sbit sfr external

sub function sync_time_utc(dim hourutc as short, dim byref hourset as byte, dim byref dayset as byte) as byte
   dim vhour, vdate as short
   '********
   vhour = Bcd2Dec(hourset) + hourutc ' 17 - 5
   '********
   if vhour < 0 then
      vdate = Bcd2Dec(dayset) - 1
      vhour = 24 + vhour
      if vdate <> 0 then
         dayset = Dec2Bcd(vdate)
      end if
   end if
   '*********************************
   hourset = Dec2Bcd(vhour)
   '*********************************
   result = 0
end sub

sub function add_hora(dim byref ds1307_hour2 as short, dim ds1307_hour1 as short) as short
   dim resultado, addhora as short

   addhora = Bcd2Dec(ds1307_hour2) + Bcd2Dec(ds1307_hour1)
   resultado = (Bcd2Dec(ds1307_hour1) + Bcd2Dec(ds1307_hour2)) div 24

   if addhora >= 24 then
      ds1307_hour2 = Dec2Bcd( (Bcd2Dec(ds1307_hour1) + Bcd2Dec(ds1307_hour2)) mod 24 )
   else
      ds1307_hour2 = Dec2Bcd( addhora )
   end if
   result = Dec2Bcd( resultado )
end sub

sub function add_minut(dim byref ds1307_minute2 as short, dim ds1307_minute1 as short) as short
   dim resultado, addminut as short

   addminut = Bcd2Dec(ds1307_minute2) + Bcd2Dec(ds1307_minute1)
   resultado = (Bcd2Dec(ds1307_minute1) + Bcd2Dec(ds1307_minute2)) div 60
```

```
    if addminut >= 60 then
       ds1307_minute2 = Dec2Bcd( (Bcd2Dec(ds1307_minute1) + Bcd2Dec(ds1307_minute2)) mod 60 )
    else
       ds1307_minute2 = Dec2Bcd( addminut )
    end if
    result = Dec2Bcd( resultado )
end sub

sub procedure ds1390_init()
    '************
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV4,_SPI_DATA_SAMPLE_END,_SPI_CLK_IDLE_LOW,_SPI_LOW_2_HIGH)
Delay_ms(30)
    '************
    'GIE_bit = 0
    CS_RTC_DS1390 = 0
    SPI1_write(0x8D)
    SPI1_write(0x20)
    CS_RTC_DS1390 = 1
    Delay_ms(5)
    CS_RTC_DS1390 = 0
    SPI1_write(0x8F)
    SPI1_write(0x00)
    CS_RTC_DS1390 = 1
    Delay_ms(5)
    'GIE_bit = 1
end sub

sub procedure ds1309_readall(dim byref hourset as byte, dim byref minuteset as byte, dim byref secondset as byte, dim
byref milisecondset as byte, dim byref dateset as byte, dim byref dayset as byte, dim byref monthset as byte, dim byref
yearset as byte)
    '************
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV4,_SPI_DATA_SAMPLE_END,_SPI_CLK_IDLE_HIGH,_SPI_LOW_2_HIGH)
Delay_ms(30)
    '************
    'GIE_bit = 0
    CS_RTC_DS1390 = 0
    SPI1_write(0X00)
    milisecondset = SPI1_read(0X00)
    secondset = SPI1_read(0X00)
    minuteset = SPI1_read(0X00)
    hourset = SPI1_read(0X00)
    dateset = SPI1_read(0X00)
    dayset = SPI1_read(0X00)
    monthset = SPI1_read(0X00)
    yearset = SPI1_read(0X00)
    CS_RTC_DS1390 = 1
    Delay_ms(5)
    'GIE_bit = 1
end sub

sub procedure ds1309_setreg(dim addr, vdata as byte)
    '************
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV4,_SPI_DATA_SAMPLE_END,_SPI_CLK_IDLE_HIGH,_SPI_LOW_2_HIGH)
Delay_ms(30)
    '**********
    'GIE_bit = 0
    '************
    CS_RTC_DS1390 = 0
```

```
    SPI1_write(addr)
    SPI1_write(vdata)
    '**********
    CS_RTC_DS1390 = 1
    Delay_ms(5)
    'GIE_bit = 1
end sub


sub procedure ds1309_setinit(dim hourset, minuteset, secondset, milisecondset, dateset, dayset, monthset, yearset as
byte)
    '************
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV4,_SPI_DATA_SAMPLE_END,_SPI_CLK_IDLE_LOW,_SPI_LOW_2_HIGH)
Delay_ms(30)
    '************
    'GIE_bit = 0
    CS_RTC_DS1390 = 0
    SPI1_write(0X80)
    SPI1_write(milisecondset) 'mseconds
    SPI1_write(secondset) 'seconds
    SPI1_write(minuteset) 'minutes
    SPI1_write(hourset) 'hour
    SPI1_write(dateset) 'date
    SPI1_write(dayset) 'day
    SPI1_write(monthset) 'month
    SPI1_write(yearset) 'year
    CS_RTC_DS1390 = 1
    Delay_ms(5)
    'GIE_bit = 1
    '**********
end sub

end.
```

XVII-A3. Código de control del funcionamiento del Chip de radiofrecuencia:

```
'{*****************************************************************************************************
*
'* Project Name: nRF definitions
'* Company: (c) mikroElektronika, 2012
'* Revision History:
'*    20120724 (DA):
'*     - initial release;
'* Description:
'*    This project contains definitions used by API functions for NORDIC nRF24L01+.
'*****************************************************************************************************
}
module nRF_defines

const ADR_LENGTH         = 5    ' 5 bytes TX(RX) address width
const TX_PLOAD_WIDTH     = 32   ' 16 bytes TX payload
const RX_PLOAD_WIDTH     = 32   ' 16 bytes RX payload

' Define nRF24L01 interrupt flag's
const IDLE_              = 0x00  ' Idle, no interrupt pending
const MAX_RT             = 0x10  ' Max #of TX retrans interrupt
const TX_DS              = 0x20  ' TX data sent interrupt
const RX_DR              = 0x40  ' RX data received

' SPI(nRF24L01) commands
const READ_REG           = 0x00  ' Define read command to register
const WRITE_REG          = 0x20  ' Define write command to register
const RD_RX_PLOAD        = 0x61  ' Define RX payload register address
const WR_TX_PLOAD        = 0xA0  ' Define TX payload register address
const FLUSH_TX           = 0xE1  ' Define flush TX register command
const FLUSH_RX           = 0xE2  ' Define flush RX register command
const REUSE_TX_PL        = 0xE3  ' Define reuse TX payload register command
const NOP                = 0xFF  ' Define No Operation, might be used to read status register

' SPI(nRF24L01) registers(addresses)
const CONFIG             = 0x00  ' 'Config' register address
const EN_AA              = 0x01  ' 'Enable Auto Acknowledgment' register address
const EN_RXADDR          = 0x02  ' 'Enabled RX addresses' register address
const SETUP_AW           = 0x03  ' 'Setup address width' register address
const SETUP_RETR         = 0x04  ' 'Setup Auto. Retrans' register address
const RF_CH              = 0x05  ' 'RF channel' register address
const RF_SETUP           = 0x06  ' 'RF setup' register address
const STATUS_            = 0x07  ' 'Status' register address
const OBSERVE_TX         = 0x08  ' 'Observe TX' register address
const CD                 = 0x09  ' 'Carrier Detect' register address
const RX_ADDR_P0         = 0x0A  ' 'RX address pipe0' register address
const RX_ADDR_P1         = 0x0B  ' 'RX address pipe1' register address
const RX_ADDR_P2         = 0x0C  ' 'RX address pipe2' register address
const RX_ADDR_P3         = 0x0D  ' 'RX address pipe3' register address
const RX_ADDR_P4         = 0x0E  ' 'RX address pipe4' register address
const RX_ADDR_P5         = 0x0F  ' 'RX address pipe5' register address
const TX_ADDR_           = 0x10  ' 'TX address' register address
const RX_PW_P0           = 0x11  ' 'RX payload width, pipe0' register address
const RX_PW_P1           = 0x12  ' 'RX payload width, pipe1' register address
const RX_PW_P2           = 0x13  ' 'RX payload width, pipe2' register address
const RX_PW_P3           = 0x14  ' 'RX payload width, pipe3' register address
const RX_PW_P4           = 0x15  ' 'RX payload width, pipe4' register address
const RX_PW_P5           = 0x16  ' 'RX payload width, pipe5' register address
const FIFO_STATUS        = 0x17  ' 'FIFO Status Register' register address
```

```
' Definitions used in TEST application

const CLEAR_              = 0x00

const FREQ_TABLE_SIZE     = 0x10
const TX_PLOAD_MAX_WIDTH  = 0x20
const RX_PLOAD_MAX_WIDTH  = 0x20  ' RX payload length = 32 bytes
const NUM_OF_PIPES        = 0x06
const TX_ADDR_WIDTH       = 0x05
const TRANS_PARAMS        = 0x02
const TRANS_SOURCE        = 0x00
const TRANS_VALUE         = 0x01
const TIMER_              = 0x00
const BUTTON              = 0x01
const TX_MODE_            = 0x00
const RX_MODE_            = 0x01
const IDLE_MODE           = 0x02
const MASK_IRQ_FLAGS      = 0x70
const MASK_RX_DR_FLAG     = 0x40
const MASK_TX_DS_FLAG     = 0x20
const MASK_MAX_RT_FLAG    = 0x10
const RX_P_NO             = 0x0e

const RX_FIFO_EMPTY       = 0x07
const RX_EMPTY            = 0x01

const PIPE0               = 0x00
const PIPE1               = 0x01
const PIPE2               = 0x02
const PIPE3               = 0x03
const PIPE4               = 0x04
const PIPE5               = 0x05

const LINK_STATUS         = 0x00
const LINK_CHANNEL        = 0x01
const LINK_NO_MESS        = 0x00
const LINK_LOSS           = 0x01
const LINK_ESTABLISH      = 0x02
const LINK_RELOST         = 0x03
const STOP_               = 0x04

const CLEARED             = 0x00
const TIMEOUT             = 0x01

' CE_Action definitions
const CE_LOW              = 0x00
const CE_HIGH             = 0x01
const CE_PULSE            = 0x02

' Implemented HOST commands
const CSN_STATE           = 0x01  ' Set state of CSN(SSN) signal
const SPI_COMMAND         = 0x02  ' Perform a SPI read/write operation
const CE_BIT              = 0x03  ' Set state of CE signal
const FW_VER              = 0x04  ' Returns FirmWare version of this system
const READ_USB_ID         = 0x05  ' Returns the rotary switch value
const WRITE_LEDS          = 0x06  ' Turn LEDs on/off
const WR_FLASH_BLOCK      = 0x07  ' Writes 32 bytes of flash data to block 'n'
const RD_FLASH_BLOCK      = 0x08  ' Reads and return 32 bytes of flash block 'n'
const WR_FLASH_PAGE       = 0x09  ' Writes the flash page n*256 buffer to flash
```

```
const RD_FLASH_PAGE       = 0x0a  ' Read the n*256 flash page to flash buffer
const WD_RESET            = 0x0b  ' Perform a WD reset
const READ_LOCK_BYTE      = 0x0c  ' Returns the F32x device lock byte...
const SPI_SELECT          = 0x0d  ' Select between HW and SW SPI mode
const UNUSED              = 0x0e  ' Not used command
const ST_FR_SWEEP_TX      = 0x0f  ' Start freq.sweep, TX, from channel 'cl' to 'ch' in 20ms intervals
const ST_FR_SWEEP_RX      = 0x10  ' Start freq.sweep, RX, from channel 'cl' to 'ch' in 20ms intervals
const STOP_SWEEP          = 0x11  ' Stop either the TX or the RX freq.sweep execution....
const WR_TRANS_CTRL       = 0x12  ' Select between auto. or manual transmission(TX device)
const RD_TRANS_CTRL       = 0x13  ' Read Transmit Control state
const WR_FREQ_AGIL        = 0x14  ' Select Freq. Agility for pipe.n
const RD_FREQ_AGIL        = 0x15  ' Read freq agility state for pipe.n
const WR_CH_TABLE         = 0x16  ' Writes the 16 ch's used for freq. agility
const RD_CH_TABLE         = 0x17  ' Read the 16 ch's used for agility
const WR_TX_PAYLOAD       = 0x18  ' Writes n bytes of payload data
const START_COM_MODE      = 0x19  ' Starts the communication mode, (application)
const READ_TEMP           = 0x1a  ' Returns F320 temperature
const READ_VOLT           = 0x1b  ' Returns RF_VDD voltage
const ENTER_TEST_MODE     = 0x1c  ' Ev.board enters production test
const READ_RX_DATA        = 0x1d  ' Host read RX data on pipe.n
const READ_TX_PLOAD       = 0x1e  ' Host read current RX payload
const WR_RX_PLOAD_LENGTH  = 0x1f  ' Host write current RX payload length for pipe.n
const RD_RX_PLOAD_LENGTH  = 0x20  ' Host read current RX payload length for pipe.n
const WR_FREQ_AGILITY     = 0x21  ' Command to enable/disable frequency agility
const RD_FREQ_AGILITY     = 0x22  ' Read previous command's parameter
const UPDATE_DEVICE       = 0x23  ' This command is sent before the nRF24L01 are beeing updated
const STOP_COMM_MODE      = 0x24  ' This sommand is sent when "Stop Communication Mode" button is pressed
const RD_COMM_MODE_STAT   = 0x25  ' Host read current communication mode status (0:TX, 1:RX, 2: IDLE)
const RD_LINK_STATUS      = 0x26  ' Host read current link status, i.e. message to host, RX device side

'{*********************************************************************************************************
*
'* End of File
'*********************************************************************************************************
}

implements

end
```

```
'{*******************************************************************************************************
*
'* Project Name: API functions for nRF24L01+
'* Company: (c) mikroElektronika, 2012
'* Revision History:
'*    20120724 (DA):
'*      - initial release;
'* Description:
'*    This project contains API functions for NORDIC nRF24L01+.
'*******************************************************************************************************
}
module nRF_hw_api

sub function nRF_SPI1_RW(dim byte_ as byte) as byte
sub function nRF_SPI1_RW_Reg(dim reg as byte, dim value as byte) as byte
sub function nRF_SPI1_RW_Reg_IRQ(dim reg as byte, dim value as byte) as byte
sub function nRF_SPI1_Read(dim reg as byte) as byte
sub function nRF_SPI1_Write_Buf(dim reg as byte, dim pBuf as ^byte, dim bytes as byte) as byte
sub function nRF_SPI1_Read_Buf(dim reg as byte, dim pBuf as ^byte, dim bytes as byte) as byte

implements

'{*******************************************************************************************************
*
'* nRF24L01+ connections
'*******************************************************************************************************
}
dim nRF_CS           as sbit sfr external
   nRF_CS_Direction  as sbit sfr external

   nRF_CE            as sbit sfr external
   nRF_CE_Direction  as sbit sfr external

   nRF_IRQ_Direction as sbit sfr external

'{*******************************************************************************************************
*
'*  Function: nRF_SPI1_RW
'*
'*  Description:
'*  Writes one byte to nRF24L01, and return the byte read
'*  from nRF24L01 during write, according to SPI protocol
'*
'*  In/Out parameters:
'*  In: 'byte', current byte to be written
'*  Out: 'SPI2'
'*******************************************************************************************************
}
sub function nRF_SPI1_RW(dim byte_ as byte) as byte
  result = (SPI1_Read(byte_))                ' Perform HW SPI operation
end sub

'{*******************************************************************************************************
*
'  Function: nRF_SPI1_Read
'
'  Description:
'  Read one byte from nRF24L01 register, 'reg'
'
```

```
' In/Out parameters:
' In: reg, register to read
' Out: return reg_val, register value.
'*************************************************************************************************
}
sub function nRF_SPI1_Read(dim reg as byte) as byte
dim reg_val as byte

 nRF_CS = 0                    ' CSN low, initialize SPI communication...
 nRF_SPI1_RW(reg)                    ' Select register to read from..
 reg_val = nRF_SPI1_RW(0)              ' ..then read registervalue
 nRF_CS = 1                    ' CSN high, terminate SPI communication

 result = reg_val               ' return register value
end sub

'{*************************************************************************************************
*
' Function: nRF_SPI1_RW_Reg
'
' Description:
' Writes value 'value' to register 'reg'
'
' In/Out parameters:
' In: 'reg' register to write value 'value' to.
' Return status byte.
'*************************************************************************************************
}
sub function nRF_SPI1_RW_Reg(dim reg as byte, dim value as byte) as byte
dim status_ as byte

 nRF_CS = 0                    ' CSN low, init SPI transaction
 status_ = nRF_SPI1_RW(reg)              ' select register
 nRF_SPI1_RW(value)                ' ..and write value to it..
 nRF_CS = 1                    ' CSN high again

 result = status_               ' return nRF24L01 status byte
end sub

'{*************************************************************************************************
*
' Function: nRF_SPI1_RW_Reg
'
' Description:
' Writes value 'value' to register 'reg'
'
' In/Out parameters:
' In: 'reg' register to write value 'value' to.
' Return status byte.
'*************************************************************************************************
}
sub function nRF_SPI1_RW_Reg_IRQ(dim reg as byte, dim value as byte) as byte
dim status_ as byte

 nRF_CS = 0                    ' CSN low, init SPI transaction

 SSPBUF = reg
 while (SSPSTAT.BF = 0)
 wend
```

```
 status_ = SSPBUF                    ' select register

 SSPBUF = value                      ' ..and write value to it..
 while (SSPSTAT.BF = 0)
 wend
 reg = SSPBUF

 nRF_CS = 1                          ' CSN high again

 result = status_                    ' return nRF24L01 status byte
end sub

'{*****************************************************************************************************
*
' Function: nRF_SPI1_Write_Buf
'
' Description:
' Writes contents of buffer '*pBuf' to nRF24L01
' Typically used to write TX payload, Rx/Tx address
'
' In/Out parameters:
' In: register 'reg' to write, buffer '*pBuf*' contains
' data to be written and buffer size 'buf_size' is #of
' bytes to be written
' Out: return nRF24L01 status byte.
'*****************************************************************************************************
}
sub function nRF_SPI1_Write_Buf(dim reg as byte, dim pBuf as ^byte, dim bytes as byte) as byte
dim status_, byte_ctr as byte

 nRF_CS = 0                          ' Set CSN low, init SPI tranaction
 status_ = nRF_SPI1_RW(reg)                ' Select register to write to and read status byte

 for byte_ctr = 0 to bytes - 1             ' then write all byte in buffer(*pBuf)
  nRF_SPI1_RW(pBuf^)
  pBuf = pBuf + 1
 next byte_ctr

 nRF_CS = 1                          ' Set CSN high again

 result = status_                    ' return nRF24L01 status byte
end sub

'{*****************************************************************************************************
*
' Function: nRF_SPI1_Read_Buf
'
' Description:
' Reads 'bytes' #of bytes from register 'reg'
' Typically used to read RX payload, Rx/Tx address
'
' In/Out parameters:
' In: 'reg', register to read from, '*pBuf' are buffer
' the read bytes are stored to and 'bytes' are #of bytes
' to read.
' Out: return nRF24L01 status byte.
'*****************************************************************************************************
}
```

```
sub function nRF_SPI1_Read_Buf(dim reg as byte, dim pBuf as ^byte, dim bytes as byte) as byte
dim status_, byte_ctr as byte

 nRF_CS = 0                        ' Set CSN low, init SPI tranaction
 status_ = nRF_SPI1_RW(reg)                ' Select register to write to and read status byte

 for byte_ctr = 0 to bytes - 1             ' Perform SPI1_RW to read byte from nRF24L01
  pBuf^ = nRF_SPI1_RW(0)
  pBuf = pBuf + 1
 next byte_ctr

 nRF_CS = 1                        ' Set CSN high again

 result = status_                   ' return nRF24L01 status byte
end sub

'{*****************************************************************************************************
*
'* End of File
'*****************************************************************************************************
}
end.
```

```
'{*****************************************************************************************************
*
'* Project Name: L01 protocol API
'* Company: (c) mikroElektronika, 2012
'* Revision History:
'*     20120724 (DA):
'*       - initial release;
'* Description:
'*     This project contains L01 protocol API functions for NORDIC nRF24L01+.
'*****************************************************************************************************
}
module L01_api

include nRF_hw_api
include nRF_defines

dim nRF_CE as sbit sfr external
sub procedure CE_Pin(dim action as byte)
sub procedure Update_Link_Status()
sub procedure Check_Send_TX(dim byref vdato as string)
sub procedure nRF24L01_IRQ()
sub procedure TIMER1_IRQ()
sub procedure TIMER2_IRQ()
sub procedure do_nRF_IRQ()
sub procedure RX_Mode()
sub procedure TX_Mode()

dim  value as word
    i_loop, ADC_Flag as byte
dim  ucCom_Mode as byte                 ' Communication mode on/off
dim  ucTrans_Tmr_Ctr as byte            ' variables for trans timer comm mode

dim i_nfr, nrf_listo as byte
dim STR_SEND as char[32]
dim STR_RECV, NRF_SEND as string[32]
dim NRF_checksum as byte

implements

'{*****************************************************************************************************
*
'* VARIABLES AND CONSTANTS
'*****************************************************************************************************
}

' Predefine a static pipe address
const ADDRESS_P0 as byte[ADR_LENGTH] = (0x34,0x43,0x10,0x10,0x01)
     ADDRESS_P1 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x02)
     ADDRESS_P2 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x03)
     ADDRESS_P3 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x04)
     ADDRESS_P4 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x05)
     ADDRESS_P5 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x06)

' Predefine TX payload packet..
dim  TX_PAYLOAD as byte[TX_PLOAD_WIDTH]

    TX_pload as byte[TX_PLOAD_MAX_WIDTH]          ' TX payload buffer
    RX_pload as char[RX_PLOAD_MAX_WIDTH]          ' RX payload buffer
```

```
    TX_Addr as byte[TX_ADDR_WIDTH]                ' TX address buffer
    RX_pload_length as byte[NUM_OF_PIPES]         ' holds #of bytes for pipe 0..5
    ucTX_pload_width as byte                      ' TX payload width variable

' Status bytes
    LinkStatus as byte[2]                         ' Link status array, status & channel
    ucLinkStat, ucLastStat as byte
    uiLink_Loss_Delay as byte                     ' Link Loss dalay variable
    ucLink_Loss_Status as byte

    ucRX_pipe as byte                             ' store RX_FIFO pipe number
    uiRX_info as word                             ' store RX_FIFO pipe number and pload width
    ucRX_info as word

    ucTrans_Tmr as byte                           ' Variables for trans timer comm mode
    ucTimer_Mode as byte                          ' Variable for timer mode
    ucTry_Ctr as byte                             ' Try Counter variable

' Variable that indicates nRF24L01 interrupt source
    ucIRQ_Source as byte

' TEST VARIABLES
    ucCount as byte
    str_ as char[16]

'{*****************************************************************************************************
*
'* Functionas L01_Set_Channel
'*
'* Description:
'* Set RF channel.
'*
'* In/Out parameters:
'* In: RF channel
'* Out: none
'*****************************************************************************************************
}
sub procedure L01_Set_Channel(dim rf_ch as byte )
    nRF_SPI1_RW_Reg(WRITE_REG + RF_CH, rf_ch)
end sub

'{*****************************************************************************************************
*
'* Function: L01_Get_Channel
'*
'* Description:
'* Get current RF channel.
'*
'* In/Out parameters:
'* In: none
'* Out: RF channel
'*****************************************************************************************************
}
sub function L01_Get_Channel() as byte
    result = nRF_SPI1_Read(RF_CH)
end sub

'{*****************************************************************************************************
*
```

```
'*  Function: L01_Clear_IRQ
'*
'*  Description:
'*  Clear nRF24L01 IRQ flag(s).
'*
'*  In/Out parameters:
'*  In: IRQ Flag
'*  Out: Reg state
'*************************************************************************************************
}
sub function L01_Clear_IRQ(dim irq_flag as byte ) as byte
    result = nRF_SPI1_RW_Reg(WRITE_REG + STATUS_, irq_flag)
end sub

'{*************************************************************************************************
'*
'*  Function: L01_Write_TX_Pload
'*
'*  Description:
'*  Write TX payload.
'*
'*  In/Out parameters:
'*  In: payload in *pBuf & #of bytes = plWidth
'*  Out: none
'*************************************************************************************************
}
sub procedure L01_Write_TX_Pload(dim pBuf as ^byte, dim plWidth as byte)
    nRF_SPI1_Write_Buf(WR_TX_PLOAD, pBuf, plWidth)
end sub

'{*************************************************************************************************
'*
'*  Function: L01_Get_Status
'*
'*  Description:
'*  Read status byte
'*
'*  In/Out parameters:
'*  In: none
'*  Out: Status result
'*************************************************************************************************
}
sub function L01_Get_Status() as byte
    result = nRF_SPI1_Read(STATUS_)
end sub

'{*************************************************************************************************
'*
'*  Function: L01_RD_RX_PW_n
'*
'*  Description:
'*  Get current RX payload width for pipe.n
'*
'*  In/Out parameters:
'*  In: pipe.n
'*  Out: RX payload
'*************************************************************************************************
}
sub function L01_RD_RX_PW_n(dim pipe as byte ) as byte
```

```
    result = nRF_SPI1_Read(RX_PW_P0 + pipe)
end sub

'{*********************************************************************************************************
*
'*  Function: L01_WR_RX_PW_n
'*
'*  Description:
'*  Set RX payload width for pipe.n
'*
'*  In/Out parameters:
'*  In: pipe.n & #of bytes = plWidth
'*  Out: none
'*********************************************************************************************************
}
sub procedure L01_WR_RX_PW_n(dim pipe as byte, dim plWidth as byte)
    nRF_SPI1_RW_Reg(WRITE_REG + RX_PW_P0 + pipe, plWidth)
end sub

'{*********************************************************************************************************
*
'*  Function: Init_RF
'*
'*  Description:
'*  Get current pipe.n
'*
'*  In/Out parameters:
'*  In: none
'*  Out: pipe.n
'*********************************************************************************************************
}
sub function L01_Get_Current_Pipenum() as byte
    result = ((L01_Get_Status()) and (RX_P_NO >> 1))
end sub

'{*********************************************************************************************************
*
'*  Function: L01_Read_RX_Pload
'*
'*  Description:
'*  Read current pipe#'s RX payload
'*
'*  In/Out parameters:
'*  In: buffer for pipe's payload
'*  Out: pipe# & pipe#.plWidth
'*********************************************************************************************************
}
sub function L01_Read_RX_Pload(dim pBuf as ^byte) as word
    dim plWidth, pipe as byte

    pipe = L01_Get_Current_Pipenum()
    plWidth = L01_RD_RX_PW_n(pipe)   ' Read current pipe's payload width
    nRF_SPI1_Read_Buf(RD_RX_PLOAD, pBuf, plWidth)            ' Then get RX data

    result = ((pipe << 8) + plWidth)                    ' return pipe# & pipe#.plWidth
end sub

'{*********************************************************************************************************
*
```

```
'* Function: L01_Flush_TX
'*
'* Description:
'* Flush TX FIFO
'*
'* In/Out parameters:
'* In: none
'* Out: none
'*****************************************************************************************************
}
sub procedure L01_Flush_TX()
   nRF_SPI1_RW_Reg(FLUSH_TX,0)
end sub

'{*****************************************************************************************************
'*
'* Function: L01_Flush_RX
'*
'* Description:
'* Flush RX FIFO
'*
'* In/Out parameters:
'* In: none
'* Out: none
'*****************************************************************************************************
}
sub procedure L01_Flush_RX()
   nRF_SPI1_RW_Reg(FLUSH_RX,0)
end sub

'{*****************************************************************************************************
'*
'* Function: Init_RF
'*
'* Description:
'* Read FIFO_STATUS register
'*
'* In/Out parameters:
'* In: none
'* Out: Status of FIFO register
'*****************************************************************************************************
}
sub function L01_Get_FIFO() as byte
   result = nRF_SPI1_Read(FIFO_STATUS)
end sub

'{*****************************************************************************************************
'*
'* Function: TX_Mode
'*
'* Description:
'* This function initializes one nRF24L01 device to TX mode, set TX address,
'* set RX address for auto.ack, fill TX payload, select RF channel, datarate & TX pwr.
'* PWR_UP is set, CRC(2 bytes) is enabled, & PRIM:TX.
'*
'* ToDo: One high pulse(>10µs) on CE will now send this
'* packet and expext an acknowledgment from the RX device.
'*
'* In/Out parameters:
```

```
'*  In: none
'*  Out: none
'*****************************************************************************************************
}
sub procedure TX_Mode()
    dim temp_array as byte[5]
    temp_loop as byte

    'GIE_bit = 0                          ' Disable global interrupt

    TMR1ON_bit = 0                        ' Stop Timer 1

    CE_Pin(CE_LOW)                        ' Set CE pin low to enable stanby mode

    ucCom_Mode = TX_MODE_

    L01_Flush_TX()
    L01_Flush_RX()

    L01_Clear_IRQ(MASK_IRQ_FLAGS)               ' Clear interrupts
    ucIRQ_Source = CLEAR_
    ucLinkStat = LINK_ESTABLISH
    ucLastStat = LINK_ESTABLISH

    nRF_SPI1_RW_Reg(WRITE_REG + EN_AA, 0x3f)         ' Enable Auto.Ack:Pipes 0-5
    nRF_SPI1_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f)     ' Enable Pipes 0-5
    nRF_SPI1_RW_Reg(WRITE_REG + SETUP_RETR, 0x1a)    ' 500µs + 86µs, 10 retrans...
    nRF_SPI1_RW_Reg(WRITE_REG + RF_CH, 40)           ' Select RF channel 40

    nRF_SPI1_RW_Reg(WRITE_REG + RF_SETUP, 0x0F)      ' 0X21 250KHZ 'TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
    nRF_SPI1_RW_Reg(WRITE_REG + CONFIG, 0x0E)        ' Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT &
TX_DS enabled..

    for temp_loop = 0 to 4                    ' Copy ADDRESS_P0 from ROM to RAM
       temp_array[temp_loop] = ADDRESS_P0[temp_loop]
    next temp_loop

    nRF_SPI1_Write_Buf(WRITE_REG + TX_ADDR_, @temp_array, sizeof(ADDRESS_P0))   ' Writes TX_Address to
nRF24L01
    nRF_SPI1_Write_Buf(WRITE_REG + RX_ADDR_P0, @temp_array, sizeof(ADDRESS_P0)) ' RX_Addr0 same as TX_Adr for
Auto.Ack

    Delay_ms(5)

    'GIE_bit = 1                          ' Enable global interrupt

    '  This device is now ready to transmit one packet of 16 bytes payload to a RX device at address
    '  '3443101001', with auto acknowledgment, retransmit count of 10(retransmit delay of 500µs+86µs)
    '  RF channel 40, datarate = 2Mbps with TX power = 0dBm.
end sub

'{*****************************************************************************************************
*
'*  Function: RX_Mode
'*
'*  Description:
'*  This function initializes one nRF24L01 device to RX Mode, set RX address,
'*  writes RX payload width, select RF channel, datarate & LNA HCURR.
'*  After init, CE is toggled high, which means that this device is now
```

```
'*  ready to receive a datapacket.
'*
'*  In/Out parameters:
'*  In: none
'*  Out: none
'*******************************************************************************************************
}
sub procedure RX_Mode()
    dim temp_array as byte[5]
    temp_loop as byte

    'GIE_bit = 0                            ' Disable global interrupt

    TMR1ON_bit = 0                          ' Stop Timer 1

    ucCom_Mode = RX_MODE_

    CE_Pin(CE_LOW)                          ' Set CE pin low to enable stanby mode

    L01_Flush_TX()
    L01_Flush_RX()

    L01_Clear_IRQ(MASK_IRQ_FLAGS)                   ' Clear interrupts
    ucIRQ_Source = CLEAR_

    nRF_SPI1_RW_Reg(WRITE_REG + EN_AA, 0x3f)          ' Enable Auto.Ack:Pipes 0-5
    nRF_SPI1_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f)        ' Enable Pipes 0-5
    nRF_SPI1_RW_Reg(WRITE_REG + RF_CH, 40)            ' Select RF channel 40

    nRF_SPI1_RW_Reg(WRITE_REG + RF_SETUP, 0x09)        ' 0X21 250KHZ 'TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
    nRF_SPI1_RW_Reg(WRITE_REG + CONFIG, 0x0F)         ' Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR
enabled..

    for temp_loop = 0 to 4                      ' Copy ADDRESS_P0 from ROM to RAM
        temp_array[temp_loop] = ADDRESS_P0[temp_loop]
    next temp_loop

    nRF_SPI1_Write_Buf(WRITE_REG + RX_ADDR_P0, @temp_array, sizeof(ADDRESS_P0)) ' Use the same address on the
RX device as the TX device

    for temp_loop = 0 to 4                      ' Copy ADDRESS_P1 from ROM to RAM
        temp_array[temp_loop] = ADDRESS_P1[temp_loop]
    next temp_loop

    nRF_SPI1_Write_Buf(WRITE_REG + RX_ADDR_P1, @temp_array, sizeof(ADDRESS_P1))

    nRF_SPI1_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03)
    nRF_SPI1_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04)
    nRF_SPI1_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05)
    nRF_SPI1_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06)

    nRF_SPI1_RW_Reg(WRITE_REG + RX_PW_P0, RX_PLOAD_WIDTH) ' Select same RX payload width as TX Payload width
    nRF_SPI1_RW_Reg(WRITE_REG + RX_PW_P1, RX_PLOAD_WIDTH)
    nRF_SPI1_RW_Reg(WRITE_REG + RX_PW_P2, RX_PLOAD_WIDTH)
    nRF_SPI1_RW_Reg(WRITE_REG + RX_PW_P3, RX_PLOAD_WIDTH)
    nRF_SPI1_RW_Reg(WRITE_REG + RX_PW_P4, RX_PLOAD_WIDTH)
    nRF_SPI1_RW_Reg(WRITE_REG + RX_PW_P5, RX_PLOAD_WIDTH)

    Delay_ms(5)
```

```
    TMR1H       = 0x63                      ' Reset Timer 1
    TMR1L       = 0xBF
    TMR2IF_bit  = 0
    TMR2ON_bit  = 0                         ' required for RX mode

    'GIE_bit = 1                            ' Enable global interrupt

    CE_Pin(CE_HIGH)                         ' Set CE pin high to enable RX device

    ' This device is now ready to receive one packet of 16 bytes payload from a TX device sending to address
    ' '3443101001', with auto acknowledgment, retransmit count of 10, RF channel 40 and datarate = 2Mbps.
end sub

'{*****************************************************************************************************
*
'* CE pin select state
'*****************************************************************************************************
}
sub procedure CE_Pin(dim action as byte)    ' CE pin high, low or pulse..
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV16, _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW,
_SPI_LOW_2_HIGH)
    Delay_ms(25)
    select case (action)
        case CE_LOW             ' action == 0, CE low
            nRF_CE = 0
        case CE_HIGH            ' action == 1, CE high
            nRF_CE = 1
        case CE_PULSE           ' action == 2, CE pulse (10µs)
            TMR2   = 0
            nRF_CE = 1          ' Set CE pin high
            TMR2ON_bit = 1      ' Start Timer2, CE pulse timer
    end select
end sub

'{*****************************************************************************************************
*
'* Interrupt from nRF24L01+
'*****************************************************************************************************
}
sub procedure nRF24L01_IRQ()
    dim temp as byte

    temp = nRF_SPI1_RW_Reg_IRQ(MASK_IRQ_FLAGS, WRITE_REG + STATUS_)   ' Read status & clear IRQ flag's

    select case (temp and MASK_IRQ_FLAGS)
        case MAX_RT
            ucIRQ_Source = MAX_RT       ' Max Retries
        case TX_DS                      ' TX data sent
            ucIRQ_Source = TX_DS
        case RX_DR                      ' RX data received
            ucIRQ_Source = RX_DR
    end select
end sub

'{*****************************************************************************************************
*
'* TIMER 1 Interrupt handler - Sys tick timer
```

```
'*****************************************************************************************************
}
sub procedure TIMER1_IRQ()
    Update_Link_Status()                ' Link_Status used for "Events:" RX mode
    if(ucLinkStat = LINK_LOSS) then
        uiLink_Loss_Delay =  uiLink_Loss_Delay + 1
    else
        uiLink_Loss_Delay = 0
    end if
end sub

'{*****************************************************************************************************
*
'* TIMER 2 Interrupt handler
'*****************************************************************************************************
}
sub procedure TIMER2_IRQ()
    TMR2ON_bit = 0      ' Disable Timer2
    nRF_CE = 0
end sub

'-- HELPER FUNCTIONS ----------------------------------------------------------'/
sub procedure Update_Link_Status()
    ucLastStat = ucLinkStat
end sub

'{*-------------------------------------------------------------------------------------------------*}
' User code
'{*-------------------------------------------------------------------------------------------------*}

'{*****************************************************************************************************
*
'*  Function: do_RX_IRQ
'*
'*  Description: Handler function. Checks status of the RF module and acts accordingly
'*  Status of the RF module is updated in nRF24L01_IRQ (generated by external interrupt from nRF)
'*
'*  In/Out parameters:
'*  In: IRQ source - Max Retries, TX_DS & RX_DR
'*  Out: Reset IRQ source
'*****************************************************************************************************
}

sub procedure do_nRF_IRQ()
    dim rfid_checksum as byte
    if(ucIRQ_Source = CLEAR_) then
        exit
    end if

    'GIE_bit = 0                    ' Disable global interrupt
    TMR1ON_bit = 0                  ' Stop Timer 1

    select case (ucIRQ_Source)
        case RX_DR                      ' RX Data received interrupt
                                ' we have received something
            while ( not ((L01_Get_FIFO() and 0x01) = RX_EMPTY))      ' ..until FIFO empty
                uiRX_info = L01_Read_RX_Pload(@RX_pload)      ' Read current payload
            wend
```

```
    if (RX_pload[0] = 0X33) then
    'if (RX_pload[0] = 0X23) then
        NRF_checksum = 0
        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        STR_RECV[0] = RX_pload[0]
        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        for i_nfr=2 to RX_pload[1]
            'STR_RECV[i_nfr] = RX_pload[i_nfr]
            NRF_checksum = NRF_checksum + RX_pload[i_nfr] '02,03,04,05,06,07,08,09
        next i_nfr
        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        if ( (RX_pload[ RX_pload[1] + 1 ]) = (0XFF - NRF_checksum)) then
            for i_nfr=2 to RX_pload[1]
                STR_RECV[i_nfr] = RX_pload[i_nfr]
            next i_nfr
            nrf_listo = 0XFF
        else
            if (RX_pload[2] = 0X01) then
                for i_nfr=0 to 31
                    STR_RECV[i_nfr] = RX_pload[i_nfr]
                next i_nfr
                nrf_listo = 0XFF
            end if
        end if
        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    end if
    RX_pload[0] = 0X00
    RX_pload[1] = 0X00
    RX_pload[2] = 0X00

    ucLinkStat = LINK_ESTABLISH              ' set LinkStat = LINK_ESTABLISH
    L01_Clear_IRQ(MASK_RX_DR_FLAG)           ' clear RX_DR flag

case TX_DS                          ' TX Data sent interrupt
                                    ' we have sent message
    'PORTA.1 = PORTA.1 xor PORTA.1              ' _DEBUG_LED_ Led1 BLINK
    ucLinkStat = LINK_ESTABLISH              ' set LinkStat = LINK_ESTABLISH
    L01_Clear_IRQ(MASK_TX_DS_FLAG)           ' Clear TX_DS flag

    ucRX_info = nRF_SPI1_Read(OBSERVE_TX)

case MAX_RT                              ' Check if we had received ACK from other module
    'PORTA.1 = PORTA.1 xor PORTA.1              ' _DEBUG_LED_ Led3 BLINK

    inc(ucTry_Ctr)
    if(ucTry_Ctr = 3) then
        ucLinkStat = STOP_
    end if
    ucRX_info = nRF_SPI1_Read(OBSERVE_TX)

    select case (ucLinkStat)
        case LINK_ESTABLISH                 ' max retransmitt, but had comm on last channel
            L01_Clear_IRQ(MASK_MAX_RT_FLAG)         ' clear MAX_RT flag (nRF24L01)
            ucLinkStat = LINK_LOSS              ' change LinkStat state to LINK_LOSS, wait for new timeout

        case LINK_LOSS                      ' still not connected, run channel scan..
            L01_Clear_IRQ(MASK_MAX_RT_FLAG)         ' clear MAX_RT flag (nRF24L01)
            CE_Pin(CE_PULSE)                    ' retransmitt packet
            ucLinkStat = LINK_RELOST            ' change LinkStat state to LINK_RELOST
```

```
            case LINK_RELOST
               L01_Clear_IRQ(MASK_MAX_RT_FLAG)         ' clear MAX_RT flag (nRF24L01)
               CE_Pin(CE_PULSE)                        ' retransmitt packet
               ucLinkStat = LINK_RELOST

            case STOP_
               L01_Clear_IRQ(MASK_MAX_RT_FLAG)         ' clear MAX_RT flag
               ucLinkStat = LINK_ESTABLISH             ' stop retransmission
         end select
   end select
   ucIRQ_Source = CLEAR_

   TMR1H       = 0x63                        ' Reset Timer 1
   TMR1L       = 0xBF

   'GIE_bit = 1                              ' Enable global interrupt
end sub

'{*****************************************************************************************************
*
'* Send message
'*****************************************************************************************************
}
sub function Send_Packet() as byte              ' Send one data packet, controlled by Button1(SW1)
   L01_Write_TX_Pload(@TX_pload, TX_PLOAD_WIDTH)    ' Write new TX payload
   CE_Pin(CE_PULSE)                             ' Enable CE pulse, 12µs
   ucTry_Ctr = 0                             ' Reset Try_Ctr before nex transmitt..
   result = 1
end sub

'{*****************************************************************************************************
*
'*
'*****************************************************************************************************
}
sub procedure Check_Send_TX(dim byref vdato as string)
   if(ucCom_Mode = TX_MODE_) then               ' Scan button1 ONLY if "TX_MODE" AND Button_Mode
      select case (ucLinkStat)
         case LINK_ESTABLISH
            'if sizeof(vdato) <= TX_PLOAD_WIDTH then
               for i_loop = 0 to (TX_PLOAD_WIDTH-1)
                  TX_pload[i_loop] = vdato[i_loop]
               next i_loop
               if(Send_Packet()) then               ' Blink LED2 and set new LinkStat if packet sent
                  ucTry_Ctr = 0
                  ucLinkStat = LINK_LOSS               ' default LINK_LOSS
               end if
            'end if
      end select
   end if
end sub

'{*****************************************************************************************************
*
'* End of File
'*****************************************************************************************************
}
end.
```

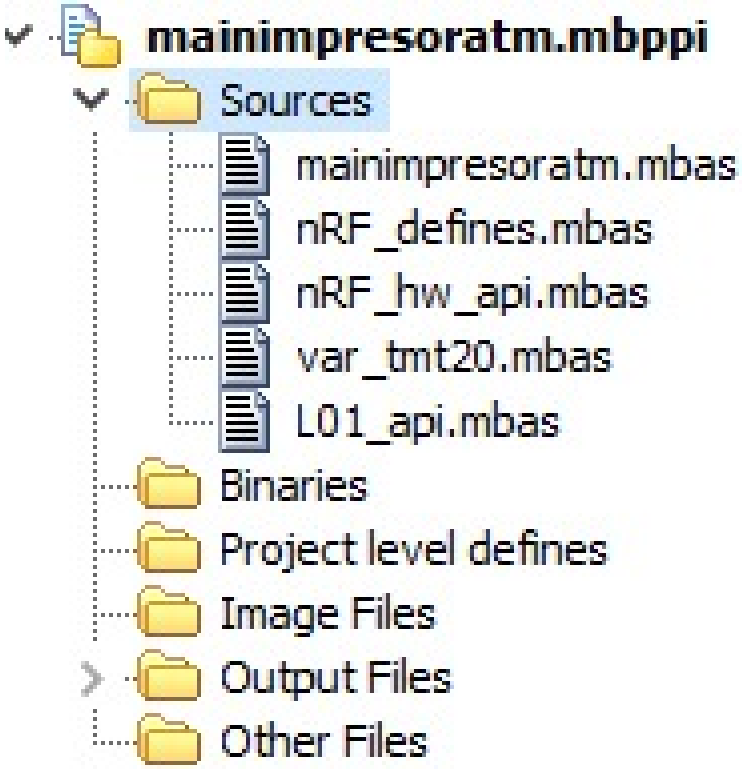
Cuadro XVII
TABLA DE COMPONENTES DEL DISPLAY.

Comment	Description	Designator	Footprint	Quantity	Part Number
100nF	Capacitores cerámicos d	C1, C2	CAP0603-1608	2	C0603C104M5RACTU
1uF	Capacitores cerámicos d	C3, C5	CAP1206-3216	2	885012208093
10uF	Capacitores cerámicos d	C4, C6	CAP1206-3216	2	885012108021
SMF4L15	Supresores de descarga	D1	SOD-123	1	SMF4L15A
DISPLAY 7PULG AC	Pantallas y accesorios de	DS1	Display 7Seg 4"	1	SA40-18SRWA
Header 5	Header, 5-Pin	P1, P2	HDR1X5	2	
330R	Resistor	R1, R2, R3, R4, R5, R6, R	RES0603-1608	7	CRCW0603330RJNEAC
1K	Thick Film Resistors - SM	R8	RES0603-1608	1	RCS0603560RFKEA
SN74HC595D	8-Bit Shift Register with	U1	D016_N	1	
ULN2803A	Transistores Darlington	U2	SOIC-SO18_M	1	ULN2803ADWR
LM7805	Voltage Regulator	VR1	SOT-223	1	UA78M05IDCYR

Cuadro XVIII
TABLA DE COMPONENTES DE LA PANTALLA.

Comment	Description	Designator	Footprint	Part Number	Quantity
COIN	Soporte para baterías d	BT1	BAT-HLD-012-THM	BAT-HLD-012-THM	1
22pF	Multilayer Ceramic Capa	C1, C2	CAP0603-1608	06035A220JAT2A	2
100nF	Multilayer Ceramic Capa	C3, C4, C8, C9, C14	CAP0603-1608	C0603C104M5RACTU	5
10uF	Multilayer Ceramic Capa	C5, C7, C11	CAP1206-3216	CL31F106ZPHNNNE	3
1uF	Multilayer Ceramic Capa	C6, C10	CAP1206-3216	CL31B105KBHNFNE	2
1N5819	Schottky Diodes & Recti	D1, D12	SOD-123	PMEG6010ELRX	2
Led rojo	LED estándar - SMD (mo	D2, D3, D4, D11	LED1206-3216	APT3216SRCPRV	4
FM5819	Rectificadores y diodos	D5, D6, D7, D9, D10	DO-214AC	FM5819-W	5
SMF4L15A	ESD Suppressors / TVS D	D8	SOD-123	SMF4L15A	1
FUSE	Header, 2-Pin	F1	FC-101	FC-101	1
IRLML2244	MOSFET 50V 300mW	Q1	sot-23	BSS84-7-F	1
2N7002P	N-Channel MOSFET	Q2, Q3	SOT-23	2N7002P,235	2
DTC143	NPN Bipolar Transistor	Q4, Q5	sot-23	DTC143ZCAT116	2
100K	Thick Film Resistors - SM	R1, R2, R3, R4, R5, R6, R	RES0603-1608	CRCW0603100KFKEAC	11
4K7	Resistores de película gr	R7, R12	RES0603-1608	CRCW06034K70FKEAHP	2
150R	Thick Film Resistors - SM	R9, R15, R16, R17, R19	RES0603-1608	CRCW0603150RFKEA	5
10K	Resistores de película gr	R11, R14	RES0603-1608	CRCW060310K0FKEAHP	2
M95512-RDW6TP	EEPROM 1.8V to 5.5V 51	U1	TSSOP-8	M95512-RDW6TP	1
DS1390U-33+	Low-Voltage SPI/3-Wire	U2	10uSOP_M	700-DS1390U-33T&R	1
MAX481ESA	Circuito integrado para	U3	NSO8_M	MAX481ESA+	1
SM712	Secuencias de diodos d	U4	sot-23	SM712-02HTG	1
MIC5504-3.3	Reguladores de tensión	U5	SOT-23-5	MIC5504-3.3YM5-TR	1
PIC18LF4620-I/PT	Microcontroladores de 8	U6	TQFP-PT44_M	PIC18LF4620-I/PT	1
MCP9700	Sensor Temperature	U7	SOT-23	MCP9700AT-E/TT	1
LM7805	Voltage Regulator	VR1	SOT-223	UA78M05IDCYR	1
32,768Khz	Crystal Oscillator	Y1	32.768kHz	520-TFC3X8-X	1
SZSAW RX	Header, 4-Pin	RX1	RX 433.92Hz		1
DIP SWITCH	Header, 4-Pin, Dual row	S1	SWITCH DIP 4P		1
TS-1166Z	Single-Pole, Single-Thro	S2, S3	TS-1166Z		2
ICSP	No soldar	ISCP1	HDR1X5		1
NRF24L01	Header, 8-Pin	NRF	module nrf24l01 v3		1
Molex 4P	Header, 4-Pin	P1	molex 4p		1
Molex 3P	Header, 3-Pin	P2	MOLEX 3P		1
Bornera	Header, 2-Pin	P3	AZUL5-2		1
Molex 5P	Header, 5-Pin	P4, P5	MOLEX 5P		2
1000uF 35v	Capacitor electrolitico 1	C12	1000uF 50v		1
100uF 25v	Capacitor electrolitico 1	C13	100uF 35v		1
10uF 25v	Capacitor electrolitico 1	C15	1uF 50v		1

Figura 56. Programas para el funcionamiento de la impresora



XVII-B1. Código de programación principal para la impresora:

```
program mainimpresoratm

include L01_api
'*******************************************************************************
symbol LED_CONEC = PORTA.1
symbol LED_PRINT = PORTA.2
symbol LED_ERROR = PORTA.3
'*******************************************************************************
symbol ST_ON = PORTA.4
'*******************************************************************************
symbol BT_PCB = PORTB.1
'*******************************************************************************
symbol SW_SLA = PORTE.2 ' ACT RF
symbol SW_SLB = PORTE.1
symbol SW_SLC = PORTE.0 ' ADDRS 0
symbol SW_SLD = PORTA.5 ' ADDRS 1
'*******************************************************************************
symbol BT_SLA = PORTB.5
symbol BT_SLB = PORTB.6
symbol BT_SLC = PORTB.4
symbol BT_SLD = PORTB.7
'*******************************************************************************
symbol BUZZER = PORTC.1

dim tmr3count as word

'dim STR_SEND as string[32]
dim STR_TEMP as string[5]

'{*****************************************************************************************************
*
'* nRF24L01+ connections
'*****************************************************************************************************
}
dim nRF_CS          as sbit at LATD4_bit
   nRF_CS_Direction  as sbit at TRISD4_bit

   nRF_CE          as sbit at LATD5_bit
   nRF_CE_Direction  as sbit at TRISD5_bit

   nRF_IRQ_Direction as sbit at TRISB0_bit

structure Pass_Counter
    dim Letra as char
    dim Contador as word
end structure

dim Count_SendNRF as word
dim i_pass as byte
dim Pasajero as Pass_Counter[5]
dim BPUERTA, Comb_Change as byte
dim tipo_mode as byte
dim INC_NRFSEND, INCT_NRFSEND as word
dim T_PAUSE, TP_PAUSE as word

'{*****************************************************************************************************
*
'* nRF24L01+ RX complete interrupt vector
```

```
'*****************************************************************************************************
}
sub procedure interrupt() 'iv 0x0008 ics ICS_AUTO
    if (RBIF_bit = 1) and (RBIE_bit = 1) then
        RBIF_bit = 0
        RBIE_bit = 0
        '*******************************************************************
        TICKET = 0
        TICKET.0 = BT_SLA
        TICKET.1 = BT_SLB
        TICKET.2 = BT_SLC
        TICKET.3 = BT_SLD
        TICKET.7 = 1
        '*******************************************************************
    end if

    ' External interrupt
    if (INT0IF_bit=1) then
        INT0IF_bit = 0      ' Clear the INT0 interrupt flag or else
        nRF24L01_IRQ()
    end if

    ' timer1 interrupt @ 5ms
    if (TMR1IF_bit=1) then
        TMR1IF_bit = 0
        TMR1ON_bit = 0      ' Timer 1 OFF
        TIMER1_IRQ()
        TMR1H       = 0x63
        TMR1L       = 0xBF
        TMR1ON_bit   = 1
        TMR1IE_bit   = 1
    end if

    ' timer2 interrupt @ 12us
    if (TMR2IF_bit=1) then
        TMR2IF_bit = 0
        TMR2ON_bit = 0
        TIMER2_IRQ()
        TMR2ON_bit = 1
    end if

    if (TMR0IF_bit=1) and (TMR0IE_bit=1) then ' 0.1 seg
        TMR0IF_bit=0
        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        if postmr0 >= 25 then
            LED_ERROR = 1
            postmr0 = 0
            TMR0ON_bit = 0
        else
            inc(postmr0)
        end if
        ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        '0X9E57 ' 0.20 Seg
        '0XCF2B ' 0.10 Seg
        TMR0H = 0XCF
        TMR0L = 0X2B
    end if

end sub
```

```
'{*****************************************************************************************************
*
'* Function: Init_RF
'*
'* Description:
'* Initailize oscillator, SPI comunication with nRF24L01+, RX interrupt and etc.
'*
'* In/Out parameters:
'* In: none
'* Out: none
'*****************************************************************************************************
}
sub procedure Init_RF()
    ' Initialize I/O ports
    nRF_CS_Direction  = 0
    nRF_CE_Direction  = 0
    nRF_IRQ_Direction = 1

    ' Set inital state for mRF connections
    nRF_CE = 0
    nRF_CS = 1

    ' Initialize SPI1 module
    SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV16, _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_LOW,
_SPI_LOW_2_HIGH)

    ' Initialize TIMER0 module as period timer with 5mS time base
    T0CON = %00000101 '64
    TMR0IE_bit = 1
    TMR0IF_bit = 1
    TMR0IP_bit = 0

    ' Initialize TIMER1 module as period timer with 5mS time base
    T1CON       = 0x11
    TMR1IF_bit   = 0     ' Reset Timer1 interrupt flag
    TMR1H       = 0x63   ' Timer1 period register
    TMR1L       = 0xBF
    TMR1IE_bit   = 1     ' Enable Timer1 interrupt

    ' Initialize TIMER2 module for CE pulse
    T2CON       = 0x04
    TMR2IE_bit   = 1     ' Enable Timer2 interrupt
    PR2         = 0X9F    ' Timer2 period register

    ' Initialize external interrupt @ INT0
    INTEDG0_bit  = 0     ' Falling edge
    INT0IE_bit   = 1

    RBPU_bit     = 1 ' ALL PULL UP

    ucCom_Mode = IDLE_MODE
    ucTrans_Tmr_Ctr = CLEAR_
end sub

' tomamos el valor alto del byte y lo pasamos a valor ascci
 sub function BCD2_H(dim bcd as byte) as byte
    dim temp as byte      ' bcd = 0x17
    temp = bcd >> 4       ' temp = 0x01
```

```
    result = temp or 0x30  ' result = 0x31
' valor que va a retornar es 0x31 equivalente 1 en ascci
end sub

sub function BCD2_L(dim bcd as byte) as byte
    dim temp as byte      ' bcd = 0x17
    temp = bcd and 0x0F   ' temp = 0x07
    result = temp or 0x30  ' result = 0x37
' valor que va a retornar es 0x37 equivalente 7 en ascci
end sub

sub procedure PRINTQR(dim byref dato_r as string, dim vsize as byte)
    dim sqrp, lsb, msb, qrsize as byte

    qrsize = vsize
    sqrp = strlen(dato_r) + 3
    lsb = (sqrp mod 256)
    msb = (sqrp / 256)

    UART1_Write(0X1D) UART1_Write(0X28) UART1_Write(0X6B) UART1_Write(0X03) UART1_Write(0X00)
UART1_Write(0X31) UART1_Write(0X43) UART1_Write(QRSIZE)
    UART1_Write(0X1D) UART1_Write(0X28) UART1_Write(0X6B) UART1_Write(0X03) UART1_Write(0X00)
UART1_Write(0X31) UART1_Write(0X45) UART1_Write(0X33)
    UART1_Write(0X1D) UART1_Write(0X28) UART1_Write(0X6B) UART1_Write( LSB) UART1_Write( MSB)
UART1_Write(0X31) UART1_Write(0X50) UART1_Write(0X30)
    UART1_Write_Text(dato_r)
    UART1_Write(0X1D) UART1_Write(0X28) UART1_Write(0X6B) UART1_Write(0X03) UART1_Write(0X00)
UART1_Write(0X31) UART1_Write(0X51) UART1_Write(0X30)
end sub

'{*****************************************************************************************************
*
'* Main program
'*****************************************************************************************************
}
main:
'*****************************************************************************
    OSCCON.6 = 1
    OSCCON.5 = 1
    OSCCON.4 = 1
    OSCTUNE.PLLEN = 1
'*****************************************************************************
    TRISA = %11100001
    PORTA = %11100001
    TRISB = %11111111
    PORTB = %00000000
    TRISC = %10010101
    PORTC = %11010101
    TRISD = %11001111
    PORTD = %11111111
    TRISE = %00001111
    PORTE = %00000000
'*****************************************************************************
    INTCON  = 0
    INTCON2 = 0
    INTCON3 = 0
'*****************************************************************************
    RBPU_bit = 0
    RBIP_bit = 1
```

```
RBIE_bit = 1
RBIF_bit = 0
'*******************************************************************************
ADCON1 = 0X0F
CMCON = 0X07

UART1_Init(38400)
for ix=0 to 3
   Delay_ms(250) CLRWDT Delay_ms(250) CLRWDT Delay_ms(250) CLRWDT Delay_ms(250) CLRWDT
next ix

if (EEPROM_Read(255) = 0XFF) or (PORTB.1 = 0) then
   Delay_ms(25)
   'A,000,A000,A000,A000
   for ix=0 to 3 '012 345 678 901
      EEPROM_Write((ix*3)+0, 0X41) Delay_ms(25) 'LETRA
      EEPROM_Write((ix*3)+1, 0X00) Delay_ms(25) 'CONTADOR H
      EEPROM_Write((ix*3)+2, 0X00) Delay_ms(25) 'CONTADOR L
   next ix

   EEPROM_Write(255, 0X00) Delay_ms(25)
   while 1
      LED_CONEC = LED_CONEC XOR 1
      LED_PRINT = LED_PRINT XOR 1
      LED_ERROR = LED_ERROR XOR 1
      Delay_ms(125)
   wend
end if

for ix=0 to 3 '012 345 678 901
   pressBT[ix].Letra = EEPROM_Read((ix*3)+0) Delay_ms(25)
   hi(pressBT[ix].Contador) = EEPROM_Read((ix*3)+1) Delay_ms(25)
   lo(pressBT[ix].Contador) = EEPROM_Read((ix*3)+2) Delay_ms(25)
next ix

Init_RF()

tiempoRTC = "01/12/2014 - 15:30'00"
ticketIMP = "A000"
QrCodeIMP = "WA000070419124500"
MODE_RXTX = 3
TICKET = 0
impAddrs = 0
nrf_listo = 0
postmr0 = 0
'******************
impAddrs.1 = SW_SLD 'H
impAddrs.0 = SW_SLC 'L
'******************
LED_CONEC = 0
LED_PRINT = 0
LED_ERROR = 0

GIE_bit = 1
PEIE_bit = 1

'*******************************************************************************
UART1_Write(0x1B) UART1_Write(0x20) UART1_Write(0x01)
UART1_Write(0x1B) UART1_Write(0x21) UART1_Write(0xB9)
```

```
UART1_Write_Text("SISTEMA TURNO ECU") UART1_Write(0x0D) UART1_Write(0x0A)
UART1_Write_Text("CONTACTO: 0991285646") UART1_Write(0x0D) UART1_Write(0x0A)
UART1_Write_Text("mail: mbenavidd@gmail.com") UART1_Write(0x0D) UART1_Write(0x0A)
UART1_Write(0x1B) UART1_Write(0x21) UART1_Write(0x00)
UART1_Write(0x1B) UART1_Write(0x20) UART1_Write(0x00)
PRINTQR("mbenavidd@gmail.com",7)
UART1_Write(0x0D) UART1_Write(0x0A)
UART1_Write(0x0D) UART1_Write(0x0A) UART1_Write(0x0D) UART1_Write(0x0A)
UART1_Write(0x0D) UART1_Write(0x0A) UART1_Write(0x0D) UART1_Write(0x0A)
UART1_Write(0x1B) UART1_Write(0x69)
'*****************************************************************************

while (TRUE)
   clrwdt
   do_nRF_IRQ()          ' call handler function frequently

   select case MODE_RXTX    ' scanea si hay dispositivos cerca
      case 1 ' TX MODE
         TX_Mode()
         MODE_RXTX = 2
      case 2 ' send data the tag
         rfid_checksum = 0
         inc_StrSend = 0XFF
'*****************************************************************************
         STR_SEND[inc(inc_StrSend)] = 0X33 ' IDE
         STR_SEND[inc(inc_StrSend)] = 0X00 ' LNGT
         '*************************************************************
         STR_SEND[inc(inc_StrSend)] = 0X20 ' FRAME
         rfid_checksum = rfid_checksum + STR_SEND[inc_StrSend]
         '*************************************************************
         STR_SEND[inc(inc_StrSend)] = impAddrs ' ADDRS
         rfid_checksum = rfid_checksum + STR_SEND[inc_StrSend]
         '*************************************************************
         STR_SEND[inc(inc_StrSend)] = "T"
         rfid_checksum = rfid_checksum + STR_SEND[inc_StrSend]
         '*************************************************************
         STR_SEND[inc(inc_StrSend)] = ((ticketsend and 0X03) + 0X41)
         rfid_checksum = rfid_checksum + STR_SEND[inc_StrSend]
         '*************************************************************
         STR_SEND[inc(inc_StrSend)] = 0XFF - rfid_checksum
         STR_SEND[ 1] = inc_StrSend - 1
'*****************************************************************************
         Check_Send_TX(STR_SEND)
         MODE_RXTX = 3
         '**************
         TMR0ON_bit = 1
         '***********
         Delay_ms(5)
      case 3 ' RX MODE 1 SEG
         MODE_RXTX = 4
         RX_Mode()
      case else
         NOP
   end select
   '*******************************************************************************

   if (ticket.7 = 1) then
      '*****************************************************************
      select case ticket
         case 0X8E 'BT 01
```

```
            ticket = 0
        case 0X8D 'BT 02
            ticket = 1
        case 0X8B 'BT 03
            ticket = 2
        case 0X87 'BT 04
            ticket = 3
        case else
            goto exitbtpress
    end select
    '*********************************************************************
    if ticket <> ticketold then
        ticketsend = ticket
        BUZZER = 1 Delay_ms(25)
        if SW_SLA = 0 then
            MODE_RXTX = 1
            ticketsend.7 = 1 '1
        else
            nrf_listo = 0XFF
            ticketsend.7 = 0 '0
        end if
        LED_ERROR = 0
    end if
    '*********************************************************************
    exitbtpress:
    ticketold = ticket
    RBIE_bit = 1
end if
'*************************************************************************
if nrf_listo = 0XFF then
    nrf_listo = 0X00
    LED_PRINT = 1
    '***********************
    if (ticketsend.7 = 1) then 'CON RF
        LED_CONEC = 1
        LED_ERROR = 0
        TMR0ON_bit = 0
        '00.01.02.03.04.05.06.07.08.09.0A.0B.0C.0D.0E
        '44.00.20.DD.TT.NN.AA,BB,DD,MM,YY,HH,MM,SS,CS
        if (0x20 = STR_RECV[2]) and (impAddrs = STR_RECV[3]) then
            'tiempoRTC = "01/12/2014 - 15:30'00"
            tiempoRTC[ 0] = BCD2_H(STR_RECV[ 8])
            QrCodeIMP[ 5] = tiempoRTC[ 0]
            tiempoRTC[ 1] = BCD2_L(STR_RECV[ 8])
            QrCodeIMP[ 6] = tiempoRTC[ 1]
            tiempoRTC[ 3] = BCD2_H(STR_RECV[ 9])
            QrCodeIMP[ 7] = tiempoRTC[ 3]
            tiempoRTC[ 4] = BCD2_L(STR_RECV[ 9])
            QrCodeIMP[ 8] = tiempoRTC[ 4]
            tiempoRTC[ 8] = BCD2_H(STR_RECV[10])
            QrCodeIMP[ 9] = tiempoRTC[ 8]
            tiempoRTC[ 9] = BCD2_L(STR_RECV[10])
            QrCodeIMP[10] = tiempoRTC[ 9]
            '*******************************
            tiempoRTC[13] = BCD2_H(STR_RECV[11])
            QrCodeIMP[11] = tiempoRTC[13]
            tiempoRTC[14] = BCD2_L(STR_RECV[11])
            QrCodeIMP[12] = tiempoRTC[14]
            tiempoRTC[16] = BCD2_H(STR_RECV[12])
```

```
            QrCodeIMP[13] = tiempoRTC[16]
            tiempoRTC[17] = BCD2_L(STR_RECV[12])
            QrCodeIMP[14] = tiempoRTC[17]
            tiempoRTC[19] = BCD2_H(STR_RECV[13])
            QrCodeIMP[15] = tiempoRTC[19]
            tiempoRTC[20] = BCD2_L(STR_RECV[13])
            QrCodeIMP[16] = tiempoRTC[20]
            '*******************************
            ticketsend = (STR_RECV[ 4] AND 0X03)
            dec(ticketsend)
            ticketsend.7 = 1
            '*******************************
            NumImp.letra = STR_RECV[ 5]
            HI(NumImp.contador) = STR_RECV[ 6]
            LO(NumImp.contador) = STR_RECV[ 7]
        else
            goto exitprint
        end if
    else 'SIN RF
        NumImp.letra = pressBT[ ticketsend ].letra
        NumImp.contador = pressBT[ ticketsend ].contador
'*****************************************************************************
        pressBT[ ticketsend ].contador = pressBT[ ticketsend ].contador + 1
        if pressBT[ ticketsend ].contador >= 200 then
            '*********************************************************
            pressBT[ ticketsend ].letra = pressBT[ ticketsend ].letra + 1
            if pressBT[ ticketsend ].letra >= 0X46 then ' 0x41,0x42,0x43,0x44,0x45
                pressBT[ ticketsend ].letra = 0X41
            end if
            '*********************************************************
            pressBT[ ticketsend ].contador = 0
        end if
        EEPROM_Write((ticketsend*3)+0,    pressBT[ ticketsend ].letra)
        EEPROM_Write((ticketsend*3)+1, HI(pressBT[ ticketsend ].contador))
        EEPROM_Write((ticketsend*3)+2, LO(pressBT[ ticketsend ].contador))
'*****************************************************************************
    end if
    '****************************************
    WordToStrWithZeros( NumImp.contador, txtemp) '01234
    '**************************
    ticketIMP[0] = NumImp.Letra
    ticketIMP[1] = txtemp[2]
    ticketIMP[2] = txtemp[3]
    ticketIMP[3] = txtemp[4]
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    UART1_Write(0x1B) UART1_Write(0x21) UART1_Write(0x08)
    UART1_Write(0x1B) UART1_Write(0x61) UART1_Write(0x30)
    UART1_Write(0x1B) UART1_Write(0x61) UART1_Write(0x31)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
    UART1_Write_Text("ATENCION ")
    select case (ticketsend AND 0X03)
        case 0
            UART1_Write_Text("TIPO A")
            QrCodeIMP[0] = "W"
        case 1
            UART1_Write_Text("TIPO B")
            QrCodeIMP[0] = "X"
        case 2
            UART1_Write_Text("TIPO C")
```

```
                QrCodeIMP[0] = "Y"
            case 3
                UART1_Write_Text("TIPO D")
                QrCodeIMP[0] = "Z"
        end select
        UART1_Write(0x0D) UART1_Write(0x0A)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
'******************************************************************************
        if ticketsend.7 = 1 then 'CON RF
            UART1_Write_Text(tiempoRTC)
            UART1_Write(0x0D) UART1_Write(0x0A)
        end if
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        UART1_Write_Text("================================")
        UART1_Write(0x0D) UART1_Write(0x0A)
'******************************************************************************
        UART1_Write(0x1B) UART1_Write(0x20) UART1_Write(0x01)
        UART1_Write(0x1B) UART1_Write(0x21) UART1_Write(0xB9)
        UART1_Write_Text("TURNO > ")
        UART1_Write_Text(ticketIMP)
        UART1_Write(0x1B) UART1_Write(0x21) UART1_Write(0x00)
        UART1_Write(0x1B) UART1_Write(0x20) UART1_Write(0x00)
        UART1_Write(0x0D) UART1_Write(0x0A)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        'QrCodeIMP[0] = 'WXYZ
        QrCodeIMP[1] = ticketIMP[0]
        QrCodeIMP[2] = ticketIMP[1]
        QrCodeIMP[3] = ticketIMP[2]
        QrCodeIMP[4] = ticketIMP[3]
        'QrCodeIMP[5] = ticketIMP[4]
        PRINTQR(QrCodeIMP,6)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        UART1_Write_Text("================================")
        UART1_Write(0x0D) UART1_Write(0x0A)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        UART1_Write_Text("AGRADECE SU PREFERENCIA")
        UART1_Write(0x0D) UART1_Write(0x0A)
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
        UART1_Write(0x0D) UART1_Write(0x0A) UART1_Write(0x0D) UART1_Write(0x0A)
        UART1_Write(0x0D) UART1_Write(0x0A) UART1_Write(0x0D) UART1_Write(0x0A)
        UART1_Write(0x1B) UART1_Write(0x69)
        '*********
        exitprint:
      end if
      ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
      LED_CONEC = 0
      LED_PRINT = 0
      'LED_ERROR = 0
      BUZZER = 0 Delay_ms(25)
   wend
end.

'{******************************************************************************************
*
'* End of File
'*********************+********************************************************************
*}
```

XVII-B2. Código de programación de comandos y variables para el control de la pantalla de la impresora: .

```
module RFID_ACTIVE

const ADR_TAG_ACTIVE   = 0X00 '0,1,2,3
const ADR_MODE_WORK    = 0X04 '4
const ADR_ABCDE_COUNT  = 0X05 '05,06,07- 08,09,10- 11,12,13- 14,15,16- 17,18,19
const ADR_WAIT_TAG     = 0X14 '20,21
const ADR_TIME_RESET   = 0X16 '22,23
const ADR_BAND_RESET   = 0X18 '24
const ADR_HARDWARE_RST = 0X19 '25,26
const ADR_DEFAULT      = 0XFF

structure IMP_Counter
   dim Letra as byte
   dim Contador as word
end structure

dim pressBT as IMP_Counter[4]
dim NumImp as IMP_Counter
dim ticketIMP as string[5] 'A000
dim QrCodeIMP as string[18]

dim ticket, ticketold, ticketsend as byte
dim impAddrs as byte

dim fecha, hora as byte[3]
dim tiempoRTC as string[21] '01/12/2014 - 15:30'00

dim ix as byte
dim inc_StrSend as byte
dim txtemp as string[5]
dim postmr0 as byte

dim HARDWARE_RST, HARDWARE_RST_CNT as word
dim ADDRS_COUNT as word
dim ADDRS_TAG as byte[5]
dim rfid_checksum as byte
dim TMR_SYNC as word
dim RFID_TX as byte
dim MODE_RXTX as byte
dim break_tag as word
dim veces_send as word

implements

end.
```

.

XVII-B3. Conjunto de código de registro para acceder al chip de comunicación: .

```
'[*****************************************************************************************************
*
'* Project Name: API functions for nRF24L01+
'* Company: (c) mikroElektronika, 2012
'* Revision History:
'*    20120724 (DA):
'*      - initial release;
'* Description:
'*    This project contains API functions for NORDIC nRF24L01+.
'*****************************************************************************************************
}
module nRF_hw_api

sub function nRF_SPI_RW(dim byte_ as byte) as byte
sub function nRF_SPI_RW(dim byte_ as byte) as byte
sub function nRF_SPI_RW_Reg(dim reg as byte, dim value as byte) as byte
sub function nRF_SPI_RW_Reg_IRQ(dim reg as byte, dim value as byte) as byte
sub function nRF_SPI_Read(dim reg as byte) as byte
sub function nRF_SPI_Write_Buf(dim reg as byte, dim pBuf as ^byte, dim bytes as byte) as byte
sub function nRF_SPI_Read_Buf(dim reg as byte, dim pBuf as ^byte, dim bytes as byte) as byte

implements

'[*****************************************************************************************************
*
'* nRF24L01+ connections
'*****************************************************************************************************
}
dim nRF_CS           as sbit sfr external
  nRF_CS_Direction   as sbit sfr external

  nRF_CE           as sbit sfr external
  nRF_CE_Direction   as sbit sfr external

  nRF_IRQ_Direction  as sbit sfr external

'[*****************************************************************************************************
*
'* Function: nRF_SPI_RW
'*
'* Description:
'* Writes one byte to nRF24L01, and return the byte read
'* from nRF24L01 during write, according to SPI protocol
'*
'* In/Out parameters:
'* In: 'byte', current byte to be written
'* Out: 'SPI2'
'*****************************************************************************************************
}
sub function nRF_SPI_RW(dim byte_ as byte) as byte
 result = (SPI_Read(byte_))                ' Perform HW SPI operation
end sub

'[*****************************************************************************************************
*
' Function: nRF_SPI_Read
'
' Description:
' Read one byte from nRF24L01 register, 'reg'
```

```
' In/Out parameters:
' In: reg, register to read
' Out: return reg_val, register value.
'*************************************************************************************************
}
sub function nRF_SPI_Read(dim reg as byte) as byte
dim reg_val as byte

 nRF_CS = 0                        ' CSN low, initialize SPI communication...
 nRF_SPI_RW(reg)                       ' Select register to read from..
 reg_val = nRF_SPI_RW(0)                 ' ..then read registervalue
 nRF_CS = 1                        ' CSN high, terminate SPI communication

 result = reg_val                   ' return register value
end sub

'{*************************************************************************************************
*
' Function: nRF_SPI_RW_Reg
'
' Description:
' Writes value 'value' to register 'reg'
'
' In/Out parameters:
' In: 'reg' register to write value 'value' to.
' Return status byte.
'*************************************************************************************************
}
sub function nRF_SPI_RW_Reg(dim reg as byte, dim value as byte) as byte
dim status_ as byte

 nRF_CS = 0                        ' CSN low, init SPI transaction
 status_ = nRF_SPI_RW(reg)               ' select register
 nRF_SPI_RW(value)                   ' ..and write value to it..
 nRF_CS = 1                        ' CSN high again

 result = status_                   ' return nRF24L01 status byte
end sub

'{*************************************************************************************************
*
' Function: nRF_SPI_RW_Reg
'
' Description:
' Writes value 'value' to register 'reg'
'
' In/Out parameters:
' In: 'reg' register to write value 'value' to.
' Return status byte.
'*************************************************************************************************
}
sub function nRF_SPI_RW_Reg_IRQ(dim reg as byte, dim value as byte) as byte
dim status_ as byte

 nRF_CS = 0                        ' CSN low, init SPI transaction

 SSPBUF = reg
 while (SSPSTAT.BF = 0)
```

```
 wend

 status_ = SSPBUF                    ' select register

 SSPBUF = value                      ' ..and write value to it..
 while (SSPSTAT.BF = 0)
 wend
 reg = SSPBUF

 nRF_CS = 1                          ' CSN high again

 result = status_                    ' return nRF24L01 status byte
end sub

'{*********************************************************************************************************
*
'  Function: nRF_SPI_Write_Buf
'
'  Description:
'  Writes contents of buffer '*pBuf' to nRF24L01
'  Typically used to write TX payload, Rx/Tx address
'
'  In/Out parameters:
'  In: register 'reg' to write, buffer '*pBuf*' contains
'  data to be written and buffer size 'buf_size' is #of
'  bytes to be written
'  Out: return nRF24L01 status byte.
'*********************************************************************************************************
}
sub function nRF_SPI_Write_Buf(dim reg as byte, dim pBuf as ^byte, dim bytes as byte) as byte
dim status_, byte_ctr as byte

 nRF_CS = 0                          ' Set CSN low, init SPI tranaction
 status_ = nRF_SPI_RW(reg)               ' Select register to write to and read status byte

 for byte_ctr = 0 to bytes - 1           ' then write all byte in buffer(*pBuf)
  nRF_SPI_RW(pBuf^)
  pBuf = pBuf + 1
 next byte_ctr

 nRF_CS = 1                          ' Set CSN high again

 result = status_                    ' return nRF24L01 status byte
end sub

'{*********************************************************************************************************
*
'  Function: nRF_SPI_Read_Buf
'
'  Description:
'  Reads 'bytes' #of bytes from register 'reg'
'  Typically used to read RX payload, Rx/Tx address
'
'  In/Out parameters:
'  In: 'reg', register to read from, '*pBuf' are buffer
'  the read bytes are stored to and 'bytes' are #of bytes
'  to read.
'  Out: return nRF24L01 status byte.
```

```
'*************************************************************************************************
}
sub function nRF_SPI_Read_Buf(dim reg as byte, dim pBuf as ^byte, dim bytes as byte) as byte
dim status_, byte_ctr as byte

  nRF_CS = 0                        ' Set CSN low, init SPI tranaction
  status_ = nRF_SPI_RW(reg)              ' Select register to write to and read status byte

  for byte_ctr = 0 to bytes - 1          ' Perform SPI_RW to read byte from nRF24L01
    pBuf^ = nRF_SPI_RW(0)
    pBuf = pBuf + 1
  next byte_ctr

  nRF_CS = 1                        ' Set CSN high again

  result = status_                   ' return nRF24L01 status byte
end sub

'{*************************************************************************************************
*
'* End of File
'*************************************************************************************************
}
end.
```

```
'{*****************************************************************************************************
*
'* Project Name: L01 protocol API
'* Company: (c) mikroElektronika, 2012
'* Revision History:
'*    20120724 (DA):
'*      - initial release;
'* Description:
'*    This project contains L01 protocol API functions for NORDIC nRF24L01+.
'*****************************************************************************************************
}
module L01_api

include nRF_hw_api
include nRF_defines
include var_tmt20

dim nRF_CE as sbit sfr external
sub procedure CE_Pin(dim action as byte)
sub procedure Update_Link_Status()
sub procedure Check_Send_TX(dim byref vdato as string)
sub procedure nRF24L01_IRQ()
sub procedure TIMER1_IRQ()
sub procedure TIMER2_IRQ()
sub procedure do_nRF_IRQ()
sub procedure RX_Mode()
sub procedure TX_Mode()

dim  value as word
    i_loop, ADC_Flag as byte
dim  ucCom_Mode as byte                ' Communication mode on/off
dim  ucTrans_Tmr_Ctr as byte           ' variables for trans timer comm mode

dim i_nfr, nrf_listo as byte
dim STR_RECV as char[32]
dim STR_SEND as char[32]

implements

'{*****************************************************************************************************
*
'* VARIABLES AND CONSTANTS
'*****************************************************************************************************
}

' Predefine a static pipe address
const ADDRESS_P0 as byte[ADR_LENGTH] = (0x34,0x43,0x10,0x10,0x01)
    ADDRESS_P1 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x02)
    ADDRESS_P2 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x03)
    ADDRESS_P3 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x04)
    ADDRESS_P4 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x05)
    ADDRESS_P5 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x06)

' Predefine TX payload packet..
dim  TX_PAYLOAD as byte[TX_PLOAD_WIDTH]

   TX_pload as byte[TX_PLOAD_MAX_WIDTH]          ' TX payload buffer
   RX_pload as char[RX_PLOAD_MAX_WIDTH]          ' RX payload buffer
```

```
'*************************************************************************************************
}
sub function nRF_SPI_Read_Buf(dim reg as byte, dim pBuf as ^byte, dim bytes as byte) as byte
dim status_, byte_ctr as byte

 nRF_CS = 0                        ' Set CSN low, init SPI tranaction
 status_ = nRF_SPI_RW(reg)                ' Select register to write to and read status byte

 for byte_ctr = 0 to bytes - 1             ' Perform SPI_RW to read byte from nRF24L01
  pBuf^ = nRF_SPI_RW(0)
  pBuf = pBuf + 1
 next byte_ctr

 nRF_CS = 1                        ' Set CSN high again

 result = status_                   ' return nRF24L01 status byte
end sub

'{*************************************************************************************************
*
'* End of File
'*************************************************************************************************
}
end.
```

```
'{*****************************************************************************************************
*
'* Project Name: L01 protocol API
'* Company: (c) mikroElektronika, 2012
'* Revision History:
'*    20120724 (DA):
'*      - initial release;
'* Description:
'*    This project contains L01 protocol API functions for NORDIC nRF24L01+.
'*****************************************************************************************************
}
module L01_api

include nRF_hw_api
include nRF_defines
include var_tmt20

dim nRF_CE as sbit sfr external
sub procedure CE_Pin(dim action as byte)
sub procedure Update_Link_Status()
sub procedure Check_Send_TX(dim byref vdato as string)
sub procedure nRF24L01_IRQ()
sub procedure TIMER1_IRQ()
sub procedure TIMER2_IRQ()
sub procedure do_nRF_IRQ()
sub procedure RX_Mode()
sub procedure TX_Mode()

dim  value as word
   i_loop, ADC_Flag as byte
dim  ucCom_Mode as byte                ' Communication mode on/off
dim  ucTrans_Tmr_Ctr as byte           ' variables for trans timer comm mode

dim i_nfr, nrf_listo as byte
dim STR_RECV as char[32]
dim STR_SEND as char[32]

implements

'{*****************************************************************************************************
*
'* VARIABLES AND CONSTANTS
'*****************************************************************************************************
}

' Predefine a static pipe address
const ADDRESS_P0 as byte[ADR_LENGTH] = (0x34,0x43,0x10,0x10,0x01)
    ADDRESS_P1 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x02)
    ADDRESS_P2 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x03)
    ADDRESS_P3 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x04)
    ADDRESS_P4 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x05)
    ADDRESS_P5 as byte[ADR_LENGTH] = (0xB3,0xB4,0xB5,0xB6,0x06)

' Predefine TX payload packet..
dim  TX_PAYLOAD as byte[TX_PLOAD_WIDTH]

   TX_pload as byte[TX_PLOAD_MAX_WIDTH]          ' TX payload buffer
   RX_pload as char[RX_PLOAD_MAX_WIDTH]          ' RX payload buffer
```

```
    TX_Addr as byte[TX_ADDR_WIDTH]              ' TX address buffer
    RX_pload_length as byte[NUM_OF_PIPES]       ' holds #of bytes for pipe 0..5
    ucTX_pload_width as byte                    ' TX payload width variable

' Status bytes
    LinkStatus as byte[2]                   ' Link status array, status & channel
    ucLinkStat, ucLastStat as byte
    uiLink_Loss_Delay as byte                   ' Link Loss dalay variable
    ucLink_Loss_Status as byte

    ucRX_pipe as byte                       ' store RX_FIFO pipe number
    uiRX_info as word                       ' store RX_FIFO pipe number and pload width
    ucRX_info as word

    ucTrans_Tmr as byte                     ' Variables for trans timer comm mode
    ucTimer_Mode as byte                    ' Variable for timer mode
    ucTry_Ctr as byte                       ' Try Counter variable

' Variable that indicates nRF24L01 interrupt source
    ucIRQ_Source as byte

' TEST VARIABLES
    ucCount as byte
    str as char[16]

'{*****************************************************************************************************
*
'* Functionas L01_Set_Channel
'*
'* Description:
'* Set RF channel.
'*
'* In/Out parameters:
'* In: RF channel
'* Out: none
'*****************************************************************************************************
}
sub procedure L01_Set_Channel(dim rf_ch as byte )
    nRF_SPI_RW_Reg(WRITE_REG + RF_CH, rf_ch)
end sub

'{*****************************************************************************************************
*
'* Function: L01_Get_Channel
'*
'* Description:
'* Get current RF channel.
'*
'* In/Out parameters:
'* In: none
'* Out: RF channel
'*****************************************************************************************************
}
sub function L01_Get_Channel() as byte
    result = nRF_SPI_Read(RF_CH)
end sub

'{*****************************************************************************************************
*
```

```
'*  Function: L01_Clear_IRQ
'*
'*  Description:
'*  Clear nRF24L01 IRQ flag(s).
'*
'*  In/Out parameters:
'*  In: IRQ Flag
'*  Out: Reg state
'*********************************************************************************************************
}
sub function L01_Clear_IRQ(dim irq_flag as byte ) as byte
    result = nRF_SPI_RW_Reg(WRITE_REG + STATUS_, irq_flag)
end sub

'{*********************************************************************************************************
*
'*  Function: L01_Write_TX_Pload
'*
'*  Description:
'*  Write TX payload.
'*
'*  In/Out parameters:
'*  In: payload in *pBuf & #of bytes = plWidth
'*  Out: none
'*********************************************************************************************************
}
sub procedure L01_Write_TX_Pload(dim pBuf as ^byte, dim plWidth as byte)
    nRF_SPI_Write_Buf(WR_TX_PLOAD, pBuf, plWidth)
end sub

'{*********************************************************************************************************
*
'*  Function: L01_Get_Status
'*
'*  Description:
'*  Read status byte
'*
'*  In/Out parameters:
'*  In: none
'*  Out: Status result
'*********************************************************************************************************
}
sub function L01_Get_Status() as byte
    result = nRF_SPI_Read(STATUS_)
end sub

'{*********************************************************************************************************
*
'*  Function: L01_RD_RX_PW_n
'*
'*  Description:
'*  Get current RX payload width for pipe.n
'*
'*  In/Out parameters:
'*  In: pipe.n
'*  Out: RX payload
'*********************************************************************************************************
}
sub function L01_RD_RX_PW_n(dim pipe as byte ) as byte
```

```
    result = nRF_SPI_Read(RX_PW_P0 + pipe)
end sub

'{*********************************************************************************************************
'*
'*  Function: L01_WR_RX_PW_n
'*
'*  Description:
'*  Set RX payload width for pipe.n
'*
'*  In/Out parameters:
'*  In: pipe.n & #of bytes = plWidth
'*  Out: none
'*********************************************************************************************************
'}
sub procedure L01_WR_RX_PW_n(dim pipe as byte, dim plWidth as byte)
    nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P0 + pipe, plWidth)
end sub

'{*********************************************************************************************************
'*
'*  Function: Init_RF
'*
'*  Description:
'*  Get current pipe.n
'*
'*  In/Out parameters:
'*  In: none
'*  Out: pipe.n
'*********************************************************************************************************
'}
sub function L01_Get_Current_Pipenum() as byte
    result = ((L01_Get_Status()) and (RX_P_NO >> 1))
end sub

'{*********************************************************************************************************
'*
'*  Function: L01_Read_RX_Pload
'*
'*  Description:
'*  Read current pipe#'s RX payload
'*
'*  In/Out parameters:
'*  In: buffer for pipe's payload
'*  Out: pipe# & pipe#.plWidth
'*********************************************************************************************************
'}
sub function L01_Read_RX_Pload(dim pBuf as ^byte) as word
    dim plWidth, pipe as byte

    pipe = L01_Get_Current_Pipenum()
    plWidth = L01_RD_RX_PW_n(pipe)   ' Read current pipe's payload width
    nRF_SPI_Read_Buf(RD_RX_PLOAD, pBuf, plWidth)             ' Then get RX data

    result = ((pipe << 8) + plWidth)                      ' return pipe# & pipe#.plWidth
end sub

'{*********************************************************************************************************
'*
```

```
'* Function: L01_Flush_TX
'*
'* Description:
'* Flush TX FIFO
'*
'* In/Out parameters:
'* In: none
'* Out: none
'*************************************************************************************************
}
sub procedure L01_Flush_TX()
   nRF_SPI_RW_Reg(FLUSH_TX,0)
end sub

'{*************************************************************************************************
*
'* Function: L01_Flush_RX
'*
'* Description:
'* Flush RX FIFO
'*
'* In/Out parameters:
'* In: none
'* Out: none
'*************************************************************************************************
}
sub procedure L01_Flush_RX()
   nRF_SPI_RW_Reg(FLUSH_RX,0)
end sub

'{*************************************************************************************************
*
'* Function: Init_RF
'*
'* Description:
'* Read FIFO_STATUS register
'*
'* In/Out parameters:
'* In: none
'* Out: Status of FIFO register
'*************************************************************************************************
}
sub function L01_Get_FIFO() as byte
   result = nRF_SPI_Read(FIFO_STATUS)
end sub

'{*************************************************************************************************
*
'* Function: TX_Mode
'*
'* Description:
'* This function initializes one nRF24L01 device to TX mode, set TX address,
'* set RX address for auto.ack, fill TX payload, select RF channel, datarate & TX pwr.
'* PWR_UP is set, CRC(2 bytes) is enabled, & PRIM:TX.
'*
'* ToDo: One high pulse(>10µs) on CE will now send this
'* packet and expext an acknowledgment from the RX device.
'*
'* In/Out parameters:
```

```
'*  In: none
'*  Out: none
'*****************************************************************************************************
}
sub procedure TX_Mode()
    dim temp_array as byte[5]
    temp_loop as byte

    GIE_bit = 0                         ' Disable global interrupt
    TMR1ON_bit = 0                      ' Stop Timer 1

    CE_Pin(CE_LOW)                      ' Set CE pin low to enable stanby mode

    ucCom_Mode = TX_MODE_

    L01_Flush_TX()
    L01_Flush_RX()

    L01_Clear_IRQ(MASK_IRQ_FLAGS)               ' Clear interrupts
    ucIRQ_Source = CLEAR_
    ucLinkStat = LINK_ESTABLISH
    ucLastStat = LINK_ESTABLISH

    nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f)         ' Enable Auto.Ack:Pipes 0-5
    nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f)     ' Enable Pipes 0-5
    nRF_SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x1a)    ' 500µs + 86µs, 10 retrans...
    nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40)           ' Select RF channel 40

    nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x0F)      ' 0X21 250KHZ 'TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
    nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0E)        ' Set PWR_UP bit, enable CRC(2 bytes) & Prim:TX. MAX_RT &
TX_DS enabled..

    for temp_loop = 0 to 4                      ' Copy ADDRESS_P0 from ROM to RAM
       temp_array[temp_loop] = ADDRESS_P0[temp_loop]
    next temp_loop

    nRF_SPI_Write_Buf(WRITE_REG + TX_ADDR_, @temp_array, sizeof(ADDRESS_P0))   ' Writes TX_Address to nRF24L01
    nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, @temp_array, sizeof(ADDRESS_P0)) ' RX_Addr0 same as TX_Adr for
Auto.Ack

    Delay_ms(5)

    GIE_bit = 1                         ' Enable global interrupt

    '  This device is now ready to transmit one packet of 16 bytes payload to a RX device at address
    '  '3443101001', with auto acknowledgment, retransmit count of 10(retransmit delay of 500µs+86µs)
    '  RF channel 40, datarate = 2Mbps with TX power = 0dBm.
end sub

'{*****************************************************************************************************
*
'*  Function: RX_Mode
'*
'*  Description:
'*  This function initializes one nRF24L01 device to RX Mode, set RX address,
'*  writes RX payload width, select RF channel, datarate & LNA HCURR.
'*  After init, CE is toggled high, which means that this device is now
'*  ready to receive a datapacket.
'*
```

```
'*  In/Out parameters:
'*  In: none
'*  Out: none
'*******************************************************************************************************
}
sub procedure RX_Mode()
    dim temp_array as byte[5]
    temp_loop as byte

    GIE_bit = 0                             ' Disable global interrupt
    TMR1ON_bit = 0                          ' Stop Timer 1

    ucCom_Mode = RX_MODE_

    CE_Pin(CE_LOW)                          ' Set CE pin low to enable stanby mode

    L01_Flush_TX()
    L01_Flush_RX()

    L01_Clear_IRQ(MASK_IRQ_FLAGS)                   ' Clear interrupts
    ucIRQ_Source = CLEAR_

    nRF_SPI_RW_Reg(WRITE_REG + EN_AA, 0x3f)         ' Enable Auto.Ack:Pipes 0-5
    nRF_SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x3f)       ' Enable Pipes 0-5
    nRF_SPI_RW_Reg(WRITE_REG + RF_CH, 40)           ' Select RF channel 40

    nRF_SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x09)        ' 0X21 250KHZ 'TX_PWR:0dBm, Datarate:2Mbps, LNA:HCURR
    nRF_SPI_RW_Reg(WRITE_REG + CONFIG, 0x0F)         ' Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR
enabled..

    for temp_loop = 0 to 4                      ' Copy ADDRESS_P0 from ROM to RAM
       temp_array[temp_loop] = ADDRESS_P0[temp_loop]
    next temp_loop

    nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P0, @temp_array, sizeof(ADDRESS_P0)) ' Use the same address on the
RX device as the TX device

    for temp_loop = 0 to 4                      ' Copy ADDRESS_P1 from ROM to RAM
       temp_array[temp_loop] = ADDRESS_P1[temp_loop]
    next temp_loop

    nRF_SPI_Write_Buf(WRITE_REG + RX_ADDR_P1, @temp_array, sizeof(ADDRESS_P1))

    nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P2, 0x03)
    nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P3, 0x04)
    nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P4, 0x05)
    nRF_SPI_RW_Reg(WRITE_REG + RX_ADDR_P5, 0x06)

    nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P0, RX_PLOAD_WIDTH) ' Select same RX payload width as TX Payload width
    nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P1, RX_PLOAD_WIDTH)
    nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P2, RX_PLOAD_WIDTH)
    nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P3, RX_PLOAD_WIDTH)
    nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P4, RX_PLOAD_WIDTH)
    nRF_SPI_RW_Reg(WRITE_REG + RX_PW_P5, RX_PLOAD_WIDTH)

    Delay_ms(5)

    TMR1H       = 0x63                      ' Reset Timer 1
    TMR1L       = 0xBF
```

```
    TMR2IF_bit   = 0
    TMR2ON_bit   = 0                        ' required for RX mode

    GIE_bit = 1                             ' Enable global interrupt

    CE_Pin(CE_HIGH)                         ' Set CE pin high to enable RX device

    ' This device is now ready to receive one packet of 16 bytes payload from a TX device sending to address
    ' '3443101001', with auto acknowledgment, retransmit count of 10, RF channel 40 and datarate = 2Mbps.
end sub

'{*****************************************************************************************************
*
'* CE pin select state
'*****************************************************************************************************
}
sub procedure CE_Pin(dim action as byte)    ' CE pin high, low or pulse..
    select case (action)
        case CE_LOW             ' action == 0, CE low
            nRF_CE = 0
        case CE_HIGH            ' action == 1, CE high
            nRF_CE = 1
        case CE_PULSE           ' action == 2, CE pulse (10µs)
            TMR2   = 0
            nRF_CE = 1          ' Set CE pin high
            TMR2ON_bit = 1      ' Start Timer2, CE pulse timer
    end select
end sub

'{*****************************************************************************************************
*
'* Interrupt from nRF24L01+
'*****************************************************************************************************
}
sub procedure nRF24L01_IRQ()
    dim temp as byte

    temp = nRF_SPI_RW_Reg_IRQ(MASK_IRQ_FLAGS, WRITE_REG + STATUS_)   ' Read status & clear IRQ flag's

    select case (temp and MASK_IRQ_FLAGS)
        case MAX_RT
            ucIRQ_Source = MAX_RT       ' Max Retries
        case TX_DS                      ' TX data sent
            ucIRQ_Source = TX_DS
        case RX_DR                      ' RX data received
            ucIRQ_Source = RX_DR
    end select
end sub

'{*****************************************************************************************************
*
'* TIMER 1 Interrupt handler - Sys tick timer
'*****************************************************************************************************
}
sub procedure TIMER1_IRQ()
    Update_Link_Status()                ' Link_Status used for "Events:" RX mode
    if(ucLinkStat = LINK_LOSS) then
        uiLink_Loss_Delay =  uiLink_Loss_Delay + 1
    else
```

```
        uiLink_Loss_Delay = 0
    end if
end sub

'{*****************************************************************************************************
*
'* TIMER 2 Interrupt handler
'******************************************************************************************************
}
sub procedure TIMER2_IRQ()
    TMR2ON_bit = 0      ' Disable Timer2
    nRF_CE = 0
end sub

'-- HELPER FUNCTIONS ------------------------------------------------------------'/
sub procedure Update_Link_Status()
    ucLastStat = ucLinkStat
end sub

'{*--------------------------------------------------------------------------------------------------*}
' User code
'{*--------------------------------------------------------------------------------------------------*}

'{*****************************************************************************************************
*
'*  Function: do_RX_IRQ
'*
'*  Description: Handler function. Checks status of the RF module and acts accordingly
'*  Status of the RF module is updated in nRF24L01_IRQ (generated by external interrupt from nRF)
'*
'*  In/Out parameters:
'*  In: IRQ source - Max Retries, TX_DS & RX_DR
'*  Out: Reset IRQ source
'******************************************************************************************************
}
sub procedure do_nRF_IRQ()

    if(ucIRQ_Source = CLEAR_) then
        exit
    end if

    GIE_bit = 0                     ' Disable global interrupt
    TMR1ON_bit = 0                  ' Stop Timer 1

    select case (ucIRQ_Source)
        case RX_DR                  ' RX Data received interrupt
                                    ' we have received something
          while ( not ((L01_Get_FIFO() and 0x01) = RX_EMPTY))      ' ..until FIFO empty
             uiRX_info = L01_Read_RX_Pload(@RX_pload)       ' Read current payload
          wend

          if (RX_pload[0] = 0X44) then
             rfid_checksum = 0
             ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
             STR_RECV[0] = RX_pload[0]
             ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
             for i_nfr=2 to RX_pload[1]
                STR_RECV[i_nfr] = RX_pload[i_nfr]
                rfid_checksum = rfid_checksum + RX_pload[i_nfr] '02,03,04,05,06,07,08,09
```

```
            next i_nfr
            ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
            if ( (RX_pload[ RX_pload[1] + 1 ]) = (0XFF - rfid_checksum)) then
                nrf_listo = 0XFF
            end if
            ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
            RX_pload[0] = 0X00
        end if

        ucLinkStat = LINK_ESTABLISH                ' set LinkStat = LINK_ESTABLISH
        L01_Clear_IRQ(MASK_RX_DR_FLAG)              ' clear RX_DR flag

    case TX_DS                          ' TX Data sent interrupt
                                        ' we have sent message
        'PORTA.1 = PORTA.1 xor PORTA.1                  ' _DEBUG_LED_ Led1 BLINK
        ucLinkStat = LINK_ESTABLISH                ' set LinkStat = LINK_ESTABLISH
        L01_Clear_IRQ(MASK_TX_DS_FLAG)              ' Clear TX_DS flag

        ucRX_info = nRF_SPI_Read(OBSERVE_TX)

    case MAX_RT                              ' Check if we had received ACK from other module
        'PORTA.1 = PORTA.1 xor PORTA.1                  ' _DEBUG_LED_ Led3 BLINK

        inc(ucTry_Ctr)
        if(ucTry_Ctr = 3) then
            ucLinkStat = STOP_
        end if

        ucRX_info = nRF_SPI_Read(OBSERVE_TX)

        select case (ucLinkStat)
            case LINK_ESTABLISH                    ' max retransmitt, but had comm on last channel
                L01_Clear_IRQ(MASK_MAX_RT_FLAG)          ' clear MAX_RT flag (nRF24L01)
                ucLinkStat = LINK_LOSS                 ' change LinkStat state to LINK_LOSS, wait for new timeout

            case LINK_LOSS                         ' still not connected, run channel scan..
                L01_Clear_IRQ(MASK_MAX_RT_FLAG)          ' clear MAX_RT flag (nRF24L01)
                CE_Pin(CE_PULSE)                     ' retransmitt packet
                ucLinkStat = LINK_RELOST              ' change LinkStat state to LINK_RELOST

            case LINK_RELOST
                L01_Clear_IRQ(MASK_MAX_RT_FLAG)          ' clear MAX_RT flag (nRF24L01)
                CE_Pin(CE_PULSE)                     ' retransmitt packet
                ucLinkStat = LINK_RELOST

            case STOP_
                L01_Clear_IRQ(MASK_MAX_RT_FLAG)          ' clear MAX_RT flag
                ucLinkStat = LINK_ESTABLISH            ' stop retransmission
        end select
    end select
    ucIRQ_Source = CLEAR_

    TMR1H       = 0x63                         ' Reset Timer 1
    TMR1L       = 0xBF

    GIE_bit = 1                               ' Enable global interrupt
end sub
```

```
'{*****************************************************************************************************
*
'* Send message
'*****************************************************************************************************
}
sub function Send_Packet() as byte              ' Send one data packet, controlled by Button1(SW1)
   L01_Write_TX_Pload(@TX_pload, TX_PLOAD_WIDTH)    ' Write new TX payload
   CE_Pin(CE_PULSE)                             ' Enable CE pulse, 12µs
   ucTry_Ctr = 0                              ' Reset Try_Ctr before nex transmitt..
   result = 1
end sub

'{*****************************************************************************************************
*
'*
'*****************************************************************************************************
}
sub procedure Check_Send_TX(dim byref vdato as string)
   if(ucCom_Mode = TX_MODE_) then                ' Scan button1 ONLY if "TX_MODE" AND Button_Mode
      select case (ucLinkStat)
         case LINK_ESTABLISH
            'if sizeof(vdato) <= TX_PLOAD_WIDTH then
               for i_loop = 0 to (TX_PLOAD_WIDTH-1)
                  TX_pload[i_loop] = vdato[i_loop]
               next i_loop
               if(Send_Packet()) then               ' Blink LED2 and set new LinkStat if packet sent
                  ucTry_Ctr = 0
                  ucLinkStat = LINK_LOSS              ' default LINK_LOSS
               end if
            'end if
      end select
   end if
end sub

'{*****************************************************************************************************
*
'* End of File
'*****************************************************************************************************
}
end.
```

Cuadro XIX
TABLA DE COMPONENTES DE LA IMPRESORA.

Comment	Designator	Footprint	Part Number	Quantity
100nF	C1, C2, C3, C6, C21, C	CAP0603-1608	C0603C104M5RACTU	8
47uF	C4	CAP1210-3528 21	TAJB476M010TNJ	1
68nF	C5	CAP0603-1608	CGA3E2X7R1H683K0	1
10uF	C7, C8	CAP1206-3216	UMK316BBJ106ML-T	2
5.6pF	C9	CAP0603-1608	C0603C569C5GACTU	1
47uF 6.3v	C10	CAP1210-3528 21	TPSB476K006R0350	1
47uF	C11	HDR1X2	JMK107ABJ106MA-T	1
33nF	C12	CAP0603-1608	885012206017	1
1.8pF	C13	CAP0603-1608	CC0603BRNPO9BN1R	1
1.5pF	C14	CAP0603-1608	81-GRM1885C1H1R5	1
2.0pF	C15, C16	CAP0603-1608	GRM1885C1H2R0BA	2
2.2nF	C17	CAP0603-1608	885012206061	1
4.7pF	C18	CAP0603-1608	CC0603CRNPO9BN4R	1
22pF	C19, C20	CAP0603-1608	06035A220JAT2A	2
10nF	C25, C28	CAP0603-1608	C0603C103J5RACTU	2
220pF	C26	CAP0603-1608	VJ0603Y221KXJPW1B	1
2.2uF	C27	CAP0603-1608	EMK107BJ225MA-T	1
1nF	C29	CAP0603-1608	C0603C102K1RACTU	1
1N5819	D1, D2, D3	SOD-123	1N5819HW1-7-F	3
Led rojo	D4, D5, D6, D7	LED1206-3216	LTST-C150GKT	4
MPZ1608D101BTA00	FR1, FR2	RES0603-1608	MPZ1608D101BTA00	2
452-00005	J1	DB9	452-00005	1
27uH	L1	ELL-ATV	ELL-ATV270M	1
7.5nH	L2	res0603-1608	LQW18AN7N5D00D	1
1.3nH	L3	res0603-1608	LQP18MN1N3C02D	1
3.9nH	L4, L6	res0603-1608	L-14C3N9SV4T	2
4.7nH	L5	res0603-1608	609-L-14C4N7SV4T	1
0ZCH0075FF2G	PPTC1	PPTC 1210	0ZCH0075FF2G	1
2N7002P	Q1, Q2	SOT-23	2N7002P,235	2
47K	R1, R5, R6, R12, R13,	RES0603-1608	GWCR0603-47KFT5	13
100K	R2	RES0603-1608	CR0603-FX-1003ELF	1
22K	R3, R8	RES0603-1608	RC0603FR-0722KL	2
59K	R4	RES0603-1608	RC0603FR-0759KL	1
7.5K	R7	RES0603-1608	CR0603-FX-7501ELF	1
1K	R9, R10, R11, R14, R1	res0603-1608	RC0603FR-071KL	9
1M	R16	RES0603-1608	RC0603FR-071ML	1
TS-1166Z	S1	TS-1166Z	TS-1166Z	1
SMBJ36	TVS1	DO-214AA	SMBJ36CA-13-F	1
ST1S14	U1	HsoP-8	511-ST1S14PHR	1
RFX2401C	U2	VQFN-16	RFX2401C	1
nRF24L01	U3	QFN-20 EP	nRF24L01P-R7	1
PIC18LF4620-I/PT	U4	TQFP-PT44_M	PIC18LF4620-I/PT	1
ICL3221EIVZ	U5	TSSOP16	ICL3221EIVZ	1
VC0603K300R030	VR1	RES0603-1608	VC0603K300R030	1
16Mhz	Y1	HCM49	FOXSDLF/160-20	1
RP-SMA	E1	RPSMA PCB		1
DIP SWITCH	S2	SWITCH DIP 4P		1
Header 4	P1	MOLEX 4P		1
DOT MTX	IDC 1	IDC 3X2		1
ICSP	ISCP1	HDR1X5		1
BUZZER	LS1	PS1240P02CT3		1


A
B
C
D