CENTRO DE ENSEÑANZA TÉCNICA INDUSTRIAL

SISTEMA PARA AGILIZAR EL SERVICIO DE LAS ÁREAS DE COMIDA EN CENTROS COMERCIALES

Diseño y/o desarrollo de prototipos

SUSTENTANTES:

Fernando Leonel González Rizo 12300211 8G1

José Luis Torrentera Arróniz 12300505 8G1

Carlos Adrián Palacios González 12300386 8G1

Edgar Eduardo Lomelí González 12300278 8G1

CARRERA:

TECNOLOGO EN ELECTRONICA Y COMUNICACIONES

ASESOR:

Mtra. Ana Karen Margain Pérez

Guadalajara, Jal, junio de 2016

Yam-Yam

Agradecimientos.

A Dios y el universo por mantenerme firme durante este gran esfuerzo.

A mis padres por su ayuda y apoyo.

Al plantel por prestar sus instalaciones y en especial a la plantilla de profesores de la carrera de Electrónica y comunicaciones

A la Maestra Ana Karen Margain por su ayuda tanto en el apartado metodológico como técnico.

A los compañeros que de alguna u otra forma aportaron al proyecto.

Dedicatoria

Dedicamos de manera especial este proyecto a nuestros seres queridos, pues fueron los cimientos principales para nuestra construcción como personas, quienes nos sentaron bases de responsabilidad y deseos de superación.

Resumen

Desarrollamos un sistema capaz de facilitar y agilizar el procedimiento de ordenar y recoger comida rápida en los establecimientos que ofrecen este servicio en centros comerciales, áreas de múltiples gastronetas (food truck parks) y cafeterías de centros universitarios, a partir de una aplicación móvil compatible con el sistema operativo Android.

Nuestro proyecto consta de tres etapas, la primera de ellas es el proceso de prepago en donde el usuario puede abonar crédito electrónico a una tarjeta, por medio de tecnología de comunicación de campo cercano (NFC por sus siglas en ingles), introduciendo directamente su efectivo en la máquina. Esta máquina validará el efectivo y por el momento solo será capaz de aceptar monedas de 1, 2, 5 y 10 pesos, así como billetes de 100 pesos, pero no le es posible a dicha máquina regresar cambio. Posteriormente la aplicación para Smartphone o Tablet, interfaz con el menú de cada restaurante, en donde se realiza el pedido y se pueden agregar comentarios por medio de internet después de corroborar que el código de validación sea correcto.

Por último, en cada mesa, una máquina de cobro que descontará a la tarjeta del consumidor el total correspondiente y generará el código de validación. Esta última etapa tendrá la facultad de reembolsar dinero en caso de un error del usuario, gracias a un código generado por el restaurante una vez que se compruebe la veracidad del error.

Creamos a su vez una interfaz para cada restaurante, en donde aparecerán en la pantalla los pedidos pendientes de cada mesa, con un botón para notificar al cliente que su pedido está listo, y otro para borrar la orden, cuando el cliente recoja su comida. Esta interfaz en segundo plano llevará la cuenta de las ventas para que al final el gerente pueda realizar su corte de caja.

Índice

Sumario **1**

Introducción **2**

ANTECEDENTES. 3
PLANTEAMIENTO DEL PROBLEMA. 4
SOLUCIÓN. 4
OBJETIVOS 5
General. *5*
Específicos. *5*
SUPUESTOS. 5
JUSTIFICACIÓN 6
ALCANCES Y LIMITACIONES 7

Capítulo 1: Fundamentos. **8**

ESTADO DE LA TÉCNICA. 9
G-food. *9*
E-table. *9*
E-menu *10*
EuroWin bares/restaurantes. *10*
MARCO TEÓRICO. 11
Componentes de una red. *11*
RFID/NFC (Radio Frequency Identification / Near Field Communication). *14*
SPI (Serial Peripherical Interface) *15*
SPI en AVR *17*
Módulo MFRC522 ..... *17*
MIFARE Classic 1k MF1S50yyX *19*
I2C (Inter-Integrated Circuit). *20*
Sensor de efecto Hall. *23*
AVR. *25*
RTC (Real Time Clock). *30*
App Inventor *35*

Capítulo 2: Diseño **38**

MÁQUINA DE PREPAGO 39
Detección de monedas (Sensor Hall). *39*
Detección de monedas (Conversión digital). *41*
MÁQUINA DE DESCUENTO DE SALDO 43
Configuración de fuente de reloj ATmega8. ..... *43*
Configuración I2C en ATmega8. ..... *45*
Lectura y escritura del DS1307 ..... *47*
PROGRAMA PARA LA MAQUINA DE PREPAGO 49
Algoritmo *49*
Diagrama de Flujo *50*
Algoritmo para la detección del dinero: *50*
Diagrama de Flujo *51*
Configuración de Timer e Interrupciones *52*

PROGRAMA PARA LOS PERIFÉRICOS DE PAGO EN LAS MESAS 55
Algoritmo *55*
Diagrama de flujo *57*
CONFIGURACIÓN SPI 57
ESCRITURA Y LECTURA CON MFRC522 59
Algoritmo escritura/lectura en tarjeta *60*
Diagrama de Flujo *61*
APLICACIÓN PARA ORDENAR 61
Algoritmo de la Aplicación para ordenar: *61*
ESTRUCTURA DEL CÓDIGO YEP (YA ESTÁ PAGADO) 62
CONTROL DEL MOTOR. 63

Capítulo 3: Desarrollo y Resultados 65

CALIBRACIÓN DE SALIDA DIFERENCIAL. 66
MEDICIÓN DE LOS PARÁMETROS DE LAS MONEDAS. 66
RECOMENDACIONES PARA LA CALIBRACIÓN. 67
-Caída de tensión en la red *68*
-Variación del voltaje regulado por otro componente eléctrico *68*
ACTIVACIÓN Y CONTROL DEL MOTOR 68
DESARROLLO DE PCB PARA LA MÁQUINA DE PREPAGO. 69
PRUEBAS DE LA MÁQUINA DE PREPAGO 73
DESARROLLO DE PCB PARA PERIFÉRICO DE PAGO 76
PRUEBAS DE PERIFÉRICOS EN LA MESA 77
APLICACIÓN 81

Conclusiones 83

CONCLUSIONES 84
RECOMENDACIONES Y FUTURAS MEJORAS 85

Referencias 86

Anexo A i

PLANEACIÓN I

Anexo B iii

PRESUPUESTO III

Anexo C iv

CÓDIGO FUENTE MÁQUINA DE PREPAGO IV

Anexo D viii

CÓDIGO FUENTE PERIFÉRICO DE PAGO VIII

Anexo E xii

ARCHIVOS .H XII

Anexo F xxvi

MANUAL DE USUARIO XXVI
SISTEMA PARA AGILIZAR EL SERVICIO DE ÁREAS DE COMIDA EN CENTROS COMERCIALES XXVI

Perfil de usuario: ..... *xxvi*
Máquina de prepago: ..... *xxvi*
A) *INSTALACIÓN.* XXVI
B) *USUARIO / CLIENTE.* XXVII
Aplicación Yam - Yam ..... *xxviii*
Periféricos de cobro. ..... *xxx*
A) *INSTALACIÓN* XXX
B) *USUARIO / CLIENTE* XXX
Interfaz del restaurante. ..... *xxxi*

Apéndices ..... **xxxii**

Ilustraciones

Ilustración 1 Bus SPI 15
Ilustración 2 Modo 1 16
Ilustración 3 Modo 2 16
Ilustración 4 Modo 4 16
Ilustración 5 Modo 3 16
Ilustración 6 Corrimiento SPI 17
Ilustración 7 Composición módulo 18
Ilustración 8 MFRC522 18
Ilustración 9 Tarjeta MiFire 20
Ilustración 10 Trama I2C 22
Ilustración 11 Composición de la trama I2C 23
Ilustración 12 Diagrama a bloques del sensor Hall lineal 24
Ilustración 13 Direccionamiento AVR 26
Ilustración 14 Registro MCUCSR 28
Ilustración 15 Fuentes de reloj AVR 28
Ilustración 16 ATmega 8 y ATmega 16 30
Ilustración 17 DS1307 31
Ilustración 18 Mapa de memoria DS1307 32
Ilustración 19 Registros de hora y fecha 33
Ilustración 20 Modos de configuración de RS 33
Ilustración 21 Modos de lectura y escritura 34
Ilustración 22 Plataforma App Inventor 35
Ilustración 23 Amplificador diferencial 39
Ilustración 24 ADC Flash 42
Ilustración 25 Byte alto fusibles 43
Ilustración 26 Byte bajo fusibles 44
Ilustración 27 Registro CLKSEL 44
Ilustración 28 Registro TCR0 53
Ilustración 29 Prescalers de reloj 53
Ilustración 30 Registro TIMSK 54
Ilustración 31 Registro TCNT0 54
Ilustración 32 Registro MCUSR 55

Ilustración 33 Descripción del ISC 55
Ilustración 34 Registro SPCR 57
Ilustración 35 Instancias de reloj SPR 58
Ilustración 36 Registro SPSR 59
Ilustración 37 Registro SPDR 59
Ilustración 38 Interfaz de la aplicación 63
Ilustración 39 Diagrama del controlador del motor 63
Ilustración 40 Acondicionador de señal Hall 70
Ilustración 41 Diseño PCB acondicionador de señal Hall 70
Ilustración 42 Microcontrolador para prepago 71
Ilustración 43 Diseño PCB microcontrolador prepago 71
Ilustración 44 Interior máquina de prepago 72
Ilustración 45 Exterior máquina de prepago 72
Ilustración 46 Interfaz prepago 73
Ilustración 51 Menú de prepago 73
Ilustración 52 Lectura de saldo 74
Ilustración 49 Monedas de prueba 74
Ilustración 50 Depósito de saldo 75
Ilustración 51 Diseño PCB periférico de mesa 76
Ilustración 52 Exterior de periféricos 77
Ilustración 53 Menú de configuración de mesa 77
Ilustración 54 Lectura de fecha 78
Ilustración 55 Interfaz de periférico 78
Ilustración 56 Confirmación del pago realizado 79
Ilustración 57 Códigos YEP de los restaurantes por el mismo monto 80
Ilustración 58 Códigos SUP de los restaurantes 81
Ilustración 59 Código YEP en aplicación 81
Ilustración 60 Plataforma del restaurante 82

Tablas

Tabla 1.Lectura de datos MFRC522 19
Tabla 2 Escritura de datos MFRC522 19
Tabla 3 Dirección MFRC522 19
Tabla 4 Asignación de valores a monedas 42
Tabla 5 Voltaje diferencial de salida 66
Tabla 6 Calibración 67
Tabla 7 Prueba de la detección de monedas 75
Tabla 8 Generación de código YEP 79
Tabla 9 Uso del código de reembolso 80

Sumario

En el siguiente documento se mostrará todo el proceso para la realización del proyecto, en las siguientes páginas se describen las razones para realizar el proyecto a la vez que se explican cuáles son los objetivos (o los mínimos requerimientos) del proyecto.

En el capítulo uno muestra todas laste tecnologías necesarias para desarrollas el proyecto, también muestra algunos de los sistemas y software que se encuentran en el mercado y tienen funciones similares a la de nuestro proyecto.

El segundo capítulo, muestra cómo fueron diseñados todos los circuitos y los algoritmos para los micros de cada máquina, y la configuración de todos los elementos como los módulos de NFC y el RTC.

El tercer capítulo muestra cómo se llevó a cabo el proyecto, donde se muestran imágenes del proceso, todos los circuitos y las maquinas ya montadas.

El último capítulo es una lista de mejoras, que requiere para poder llevar a nivel comercial el proyecto.

Introducción.

Antecedentes.

Para la sociedad actual en las zonas metropolitanas del mundo, como lo es la ZMG, la vida es muy agitada, por eso la mayoría de las personas se ven en la necesidad de ingerir sus alimentos fuera de casa, ya sea entre clases en algún centro universitario, al salir a realizar las compras o por entretenimiento en una plaza comercial, o simplemente salir a comer a un restaurante, ya sea uno del estilo más clásico que ofrece un tipo de comida (mariscos, cortes, comida italiana, etc.) o en un “food truck” (también conocido como gastroneta) de la emergente cultura de “Food Truck Park” (que es un lugar en el que se puede comer en diferentes “establecimientos de comida” que comparten el área donde los clientes comen).

Este tipo de lugares tienden a hacer fuertes inversiones para mantenerse competitivos, sobre todo para poder ofrecer un mejor servicio a un mayor número de clientes, por ejemplo, las cafeterías de la mayoría de los centros universitarios dan servicio a cerca del 67% del total de la plantilla de alumnos de su centro universitario, por lo que busca alternativas que le permitan tener un servicio más ágil.

En lugares de mayor aforo, como podría ser una plaza comercial la demanda de servicios, ha forzado la ampliación de espacios y la construcción o modificación de las vialidades de tránsito internas y externas de los estacionamientos, ya que el comercio se prepara para la visita de cada vez más personas. A la fecha centros como Plaza Galerías Guadalajara y La Gran Plaza Fashion Mall construyeron estacionamientos elevados por el aumento de la clientela, teniendo esta última un total de más de 2700 cajones para automóviles.

El 18 de junio de 2014, se inauguró el nodo vial Santa Anita para abastecer las necesidades de tráfico en Plazas Outlet. Tan solo en Plaza Galerías, existen a la fecha 46 locales de comida distribuidos en todos los niveles.

Planteamiento del problema.

Debido al reducido espacio y la oferta que debe tener un centro comercial de gran demanda, los locales requieren trabajar bajo condiciones no óptimas e incluso adversas y esto limita la calidad del servicio, quitándole la practicidad por la que son buscados estos negocios de comida rápida, se vuelve lento y a la vez daña la imagen de dicha plaza comercial. Como ya se mencionó, muchas personas que requieren del servicio, no son visitantes de la plaza, si no trabajadores de la misma o de zonas aledañas. Esto acentúa y dramatiza las deficiencias del servicio que ofrece el área de comida.

En cambio, en una institución educativa, los estudiantes cuentan con un horario que obstaculiza comer a la hora adecuada. Los cortos periodos entre clases no permiten comer en establecimientos y obligan al consumo de comida “Chatarra”.

Y por último, en restaurantes y parques de gastronetas, el pago se puede volver tedioso, sobre todo en el último caso donde se puede ordenar comida en diferentes establecimientos a la vez, y se tiene que hacer el pago en cada furgoneta.

Solución.

Se desarrollará un sistema en el que, a partir de una aplicación móvil se pueda ordenar lo deseado del menú y en la cantidad requerida a cualquiera de los restaurantes que deseen proveer el servicio, sin necesidad de filas, así como una máquina que puede estar integrado o no, según requiera el establecimiento, a las mesas que les permita hacer pagos, esta estructura de cobro/pago tendrá base en recargas de prepago a una tarjeta que le pertenecerá al cliente similar a una tarjeta de débito.

Todo esto junto con un programa que tendrá cada restaurante que este en esta red, que le permita recibir las órdenes, de que mesa/cliente son y pedirle a dicho cliente que recoja su orden, además de darle la opción de borrar el mensaje en el teléfono del cliente para evitar que el cliente exija dos veces su producto.

Objetivos.

General.

- Construir un sistema integral que agilice el proceso de ordenar productos en el área de alimentos de los centros comerciales, instituciones educativas de nivel superior o parques de gastronetas.

Específicos.

- Desarrollar la plataforma de prepago, que por medio de una tecnología de identificación vía NFC permita al usuario recargar efectivo para hacer pagos en el área de comida cada que lo desee.

- Crear un sistema eficiente de cobro de la orden para las áreas de alimentos y snack de los establecimientos ya planteados. Con sistemas de seguridad para evitar, en la medida de lo posible, errores por parte del usuario.

- Realizar una aplicación Android que permita al usuario realizar su orden y comprobar que su pago fue realizado, a la vez que le señale en que momento puede ir por lo listado en su comanda .

- Desarrollar un programa que le permita al restaurante recibir las comandas de sus clientes, indicando de que cliente es cada orden, y que sumado a esto le permita enviarle una notificación a su cliente de “pedido listo”, y borrar dicha notificación cuando ya no sea necesaria.

Supuestos.

Para la máquina de prepago, se requerirá de un microcontrolador que permita hacer comunicación SPI para poder utilizar un módulo de NFC, este microcontrolador debe de tener los suficientes pines como para recibir todas las señales de los diferentes sensores de autentificación de monedas y billetes, el módulo anteriormente mencionado, una pantalla y un par de botones que le permitan al usuario navegar por la interfaz de dicha máquina, además, debe tener la suficiente capacidad de almacenamiento y procesamiento para

realizar los algoritmos de interfaz, autentificación y clasificación de dinero, y manipulación de información en las tarjetas de NFC.

En el aparato que permite el cobro previo al envío de la orden, solo se requiere un RTC, para su uso interno en codificación de información, una pantalla, un teclado matricial, un módulo de NFC y el microcontrolador que hará que todas las cosas funcionen.

Tanto para la app telefónica como el software para computadora no se requerirá más que un ordenador para realizar dichos programas.

Justificación.

Hoy el cliente, además de precio, exige calidad, y ello sucede en casi todas las actividades. La certificación de calidad ha dejado de ser patrimonio exclusivo de las empresas manufactureras para pasar a las empresas de servicios, entre las cuales se destacan las empresas de salud y turismo. Por ello, hoy más que nunca, implementar un sistema de calidad en el servicio que se presta es de vital importancia, sobre todo en el negocio gastronómico, la calidad se nota y es percibida por el cliente en los aspectos mínimos.

El profesional puede ayudar al empresario con algunas cuestiones a modificar, de manera tal que el servicio ofrecido mejore su calidad que traerá aparejado mayor clientela y en muchos casos reducirá costos. Al contrario de lo que muchos piensan, los costos de la no calidad son más importantes.

Los negocios han cambiado y el empresario debe adaptarse a los nuevos tiempos, o no saldrá airoso. Es por ello que las empresas gastronómicas, que hasta no hace más de una década se manejaban de manera informal, hoy requieren de herramientas de gestión de calidad.

Alcances y limitaciones.

Para el prototipo se propone implementar el sistema para 3 restaurantes o establecimientos de servicio de comida rápida diferentes, dos mesas (con la máquina de cobro antes mencionada), para comprobar que el bloque correspondiente a la comunicación funcione correctamente y sin colisiones entre la información compartida entre dispositivos, además de 5 diferentes tags de NFC para 5 usuarios, que fungirán como el dispositivo con el crédito o efectivo, para pagar la cuenta.

La máquina de cobro solo aceptará monedas de cuatro denominaciones diferentes (uno, dos, cinco y diez pesos), y será capaz de aceptar billetes exclusivamente de 100 pesos.

Así mismo, dicha maquina no puede dar al usuario su tag, este se debe tramitar en el centro comercial o en el área de comida y no es capaz de entregar cambio.

Capítulo 1: Fundamentos.

Estado de la Técnica.

G-food.

Es un software especializado en venta, administración y gestión para restaurantes, cafeterías, comida rápida, pub y delivery. Permite gestionar la atención a clientes a través de POS de garzón y tablet desde la comanda hasta el pago del documento comercial (boleta fiscal y factura electrónica).

Sus principales áreas de acción son la atención de clientes, control y gestión de inventario (recetas), compras y pago a proveedores y entrega informes de gestión, cada una de ellas con una serie de funcionalidades para optimizar la gestión de restaurantes:

- Usa pantalla touch- screen y/o tablet para toma de comanda
- Direccionamiento automático de comandas, cocina o bar
- Establece la posición del comensal en la mesa
- Utiliza plantillas para menú y promociones
- Permite anular un plato o producto que ya fue comandado y la anulación queda registrada en un informe
- Manejo de mesas, tiempo que fue ocupada, comandada y solicitada la pre-cuenta
- En la caja, efectúa descuentos y cortesías, divide cuentas
- Controla el inventario de una o varias bodegas.
- Rebaja stock a través de las recetas
- Genera los libros de compra y venta
- Ventas de un período determinado, por garzón, por sucursal

E-table.

E-table es un sistema de mesas-ordenadores que permite elegir el platillo a degustar, ver una vista previa de la comida y enviar este pedido a la cocina a través de bluetooth. También existe la posibilidad de ver cómo los cocineros preparan el menú mediante una cámara web, a fin de entretenimiento y supervisión de calidad.

El sistema E-Table usa una proyección sobre para mostrar un menú y otras imágenes digitales. Posee dos paneles táctiles en cada mesa, que permite al consumidor navegar por el menú, pedir la orden, e interactuar con el sistema.

E-menu.

Es una aplicación de software que auxilia al consumidor, a la elección del platillo que consumirá, propicia la atracción del cliente hacia los platillos. Se trata de un menú o carta que interactúa con el cliente. La empresa ofrece dos aplicaciones, una para la selección de vinos y bebidas, y la otra para platillos de comida. Es utilizado y recomendado por grandes marcas de empresas de la industria del servicio, como Crowne Plaza Resorts y W Hotels. La aplicación está diseñada para ser compatible con iOS 6.0.

EuroWin bares/restaurantes.

Eurowin 8.0 SQL bares/restaurantes es una solución diseñada para negocios del sector servicios tales como bares, cafeterías y restaurantes, que desean controlar de forma detallada e individual diversas salas, ocupación de mesas, control de camareros, configuración de tarifas y artículos de venta. Destaca por su rapidez en la generación de tickets y el enlace de las notas de los camareros con cocina.

Está orientado a un entorno táctil con la finalidad de agilizar y facilitar su manejo; cuenta con funcionalidades básicas que cubren la gestión estándar de ventas, compras, almacén contabilidad y amortizaciones de cualquier tipo de empresa.

Permite la visualización gráfica del estado de las mesas, es configurable para crear y añadir menús nuevos y platos del día, reservar mesas, cobros parciales de tickets, permite establecer múltiples combinaciones de un mismo artículo.

Marco Teórico.

Componentes de una red.

Explicábamos que la realización del proyecto, abarcará tres etapas fundamentales, una de ellas es la comunicación entre los dispositivos, así que es necesario conocer los tipos de topología de red, protocolos, elementos y conceptos indispensables para crear una red, y de esta manera, obtener un amplio panorama de lo que es posible implementar en aproximadamente seis meses, para elegir la alternativa más admisible.

Por lo tanto, en este apartado redactaremos una síntesis de la investigación teórica realizada, con referencia a las telecomunicaciones y redes.

¿Qué es una red? consiste en la interconexión de dos o más equipos, con la finalidad de compartir almacenamiento y recursos intercambiar archivos y permitir una comunicación. El objetivo de una red es hacer que todos los programas, datos y equipo estén disponibles para cualquiera en la red, que así lo solicite, sin importar la localización física del recurso o del usuario.

Cabe señalar una importante ventaja al implementar una red, y es que proporciona una alta fiabilidad, todos los archivos podrían duplicarse en más de una máquina, si llegara a darse el caso en que una de ellas no se encuentre disponible, se utilizaría una de las otras copias, además el hecho de tener múltiples ordenadores, significa que cuando uno de estos dejes de funcionar, el resto podrá encargarse de su trabajo, aunque disminuya el rendimiento global.

Servidor: computadora utilizada para gestionar el sistema de archivos de la red, da servicio a las impresoras y controla las comunicaciones. La tarea de un servidor es procesar las peticiones realizadas por la estación de trabajo. Cuanto mayor es la red, resulta más importante tener un servidor con elevadas prestaciones, gran capacidad de RAM para optimizar los accesos a disco y mantener las colas de impresión.

Estación de trabajo: se pueden conectar a través de la placa de conexión de red y el cableado correspondiente. Las estaciones son generalmente, sistemas inteligentes capaces de encargarse de sus propias tareas de procesamiento.

Tarjeta de conexión de red: permite conectar el cableado entre servidores y estaciones de trabajo. Existen numerosos tipos de placas que soportan distintos tipos de cables y topologías de red. Las placas contienen los protocolos y ordenes necesarios para soportar el tipo de red al que está destinada. La compatibilidad a nivel físico y lógico se convierte en una cuestión relevante al considerar el uso de cualquier placa de red.

Topología de bus: todas las estaciones están conectadas a un único canal de comunicaciones por medio de unidades interfaz y derivadores. Las estaciones utilizan este canal para comunicarse con el resto.

Tiene todos sus nodos directamente a un enlace y no tiene ninguna otra conexión entre nodos. Esta topología permite a todos los dispositivos de la red ver las señales de los demás dispositivos, lo que es una ventaja si desea que todos los dispositivos obtengan esta información. La desventaja es que se produce tráfico y colisiones.

Después de conectar las computadoras físicamente al alambre, el siguiente paso es instalar el software de red en cada computadora.

Ventajas:

- Permite economizar cable.
- Inmunidad al ruido.

Desventajas:

- Si falla el canal, se cae la red.
- Si falla un terminador, se cae la red.
- Si falla un conector, se cae la red.

Topología de estrella: las redes construidas con esta topología tienen una caja de conexiones llamada HUB o concentrador en el centro de la red. Todas las PC se conectan al concentrador, el cual administra las comunicaciones entre computadoras. Es decir, si una computadora no funciona, no afecta a las demás, siempre y cuando el servidor no este caído.

La ventaja más importante de esta topología sobre la de bus, es la confiabilidad, en una topología bus, desconectar una computadora es suficiente para que toda la red colapse. En cambio, en un tipo estrella, se pueden conectar computadoras a pesar de que la red este en operación.

Ventajas:

- Si se daña un equipo, no se cae la red.
- Es fácil encontrar una falla.
- Es fácil agregar otro equipo.

Desventajas:

- Se necesita comprar el concentrador.
- Si falla el HUB, se cae la red.

Topología de anillo: las estaciones se conectan formando un aro. Cada estación se conecta a la siguiente, y la última se encuentra conectada a la primera. Cada equipo tiene un receptor y un transmisor y realizan la función de repetidor. En este tipo de red la comunicación se da por el paso de un token o testigo. Si algún nodo de la red se cae, la comunicación en todo el anillo se pierde.

Ventajas:

- Fácil de instalar y reconfigurar.
- Para añadir o quitar dispositivos, solo se mueven dos conexiones.
- Arquitectura compacta.
- El rendimiento no se declina cuando hay muchos usuarios conectados a la red

Desventajas:

- Todas las señales van en una dirección, y para llegar a una computadora debe pasar por todos los equipos del medio.
- Cuando una computadora falla, altera toda la red.

RFID/NFC (Radio Frequency Identification / Near Field Communication).

La identificación por radiofrecuencia, es un sistema para la transferencia de datos a través de distancias cortas (típicamente menos de seis pulgadas). Comúnmente solo uno de los dos dispositivos debe encenderse, mientras que el otro es pasivo. Esto permite un uso fácil en aplicaciones como tarjetas de crédito llaveros y collares para mascotas, al no tener que preocuparse por la vida de la batería.

En el proyecto, la tecnología de RFID es ampliamente útil ya que les permite a los usuarios identificarse en el medio de comunicación, mediante un dispositivo personal; todas las aplicaciones de la tecnología de reconocimiento por RF están pensadas para uso cercano, cabe mencionar que por tratarse de una tecnología de comunicación inalámbrica, no representa un medio de transmisión alcance medio tal como WiFi o Bluetooth y tampoco está preparado para el envío y recepción de grandes paquetes de información.

La viabilidad de aplicar dicha tecnología, reside en la nueva sistemática de cobro que fomenta el proyecto. Un usuario posee una tarjeta en la cual se almacenarán solamente dos datos, un identificador personal, y el monto que este tiene. Cuando el usuario desee cargar su monto o conocer su monto puede acudir a un módulo, ahí se le permitirá modificar la información para acumular una cantidad de dinero, introduciéndolo en una máquina.

Posteriormente se realiza el pedido desde la mesa; para pagar se acercará el tag y se restará la cantidad correspondiente al producto; pasado cierto tiempo, el restaurante que ofrece el producto envía un mensaje de que el producto está listo, advirtiendo al cliente para que pueda pasar a recoger su encargo; en el restaurante se pedirá que se vuelva a pasar el tag para validar la identidad de la persona que encargó el pedido. (Chavarría, 2011)

Como puede verse, la tecnología RFID/NFC forma una parte muy importante del proyecto, ya que abarca los requerimientos, sin ser sobrada u obsoleta para establecer la dinámica que proponemos y la soluciona de la mejor forma, con el objetivo de hacer más eficiente el proceso en los establecimientos de comida rápida de centros comerciales. (“RFID: Tecnología, Aplicaciones, y Perspectivas,” 2010)

SPI (Serial Peripherical Interface)

Spi es un bus de tres líneas, en el cual se transmiten paquetes de información. Es considerado una comunicación serial full dúplex, debido a que cada dispositivo conectado al bus puede actuar como transmisor y receptor al mismo tiempo. Las líneas del bus transmiten la información en una única dirección.

Las líneas del bus son: SCLK es la línea de reloj, generada por el maestro para sincronizar la transferencia de datos, MOSI (Master Out Slave In) transporta los datos del maestro hacia el esclavo y la línea MISO (Master In Slave Out) transporta los datos del esclavo al maestro. En el bus, los dispositivos son definidos como maestros o esclavos; el maestro inicia la transferencia de información en el bus y se encarga de generar las señales de reloj y control; el esclavo es controlado por el maestro a través de una línea selectora llamada SS (Slave Select), por lo tanto, es esclavo solo cuando es seleccionado por esa línea.

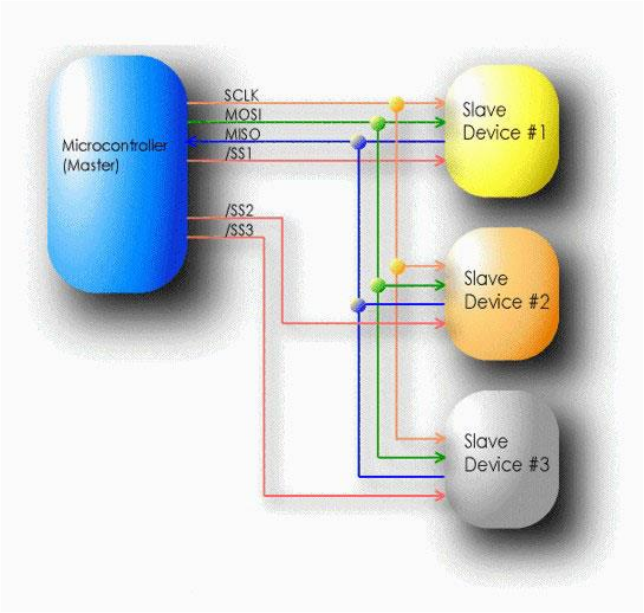


Ilustración 1 Bus SPI

Fuente: (Atmal,2010)

La transferencia de datos se realiza en bloques de 8 bits, transmitiéndose primero el bit más significativo (MBS). Los modos de reloj sincronizan la transferencia de dator a través de la línea del reloj. Para la configuración de la línea se tienen 2 bits: CPOL (Clock Polarity) el cual determina si la línea del reloj estará en bajo o en alto durante el estado Idle y CPHA (Clock Phase) donde se determina si los datos son detectados en transiciones de bajada o de subida. (López, n.d.)

Al tener 2 estados cada bit, se permite tener 4 diferentes configuraciones o modos, las cuales son incompatibles entre sí, por lo tanto, todos los dispositivos de la línea deben compartir la misma configuración. Las 4 distintas configuraciones se ilustran a continuación:

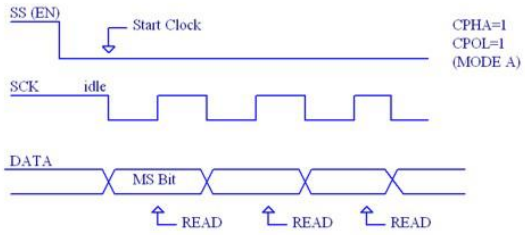


Ilustración 2 Modo 1

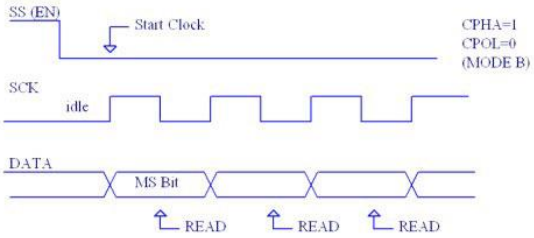


Ilustración 3 Modo 2

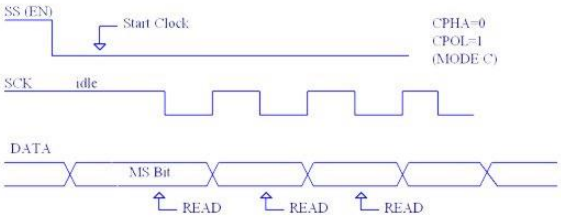


Ilustración 5 Modo 3

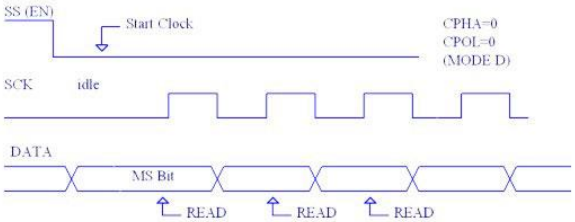


Ilustración 4 Modo 4

Fuente: (Atmel, 2013).

SPI en AVR

El SPI permite la transferencia de datos síncrona a alta velocidad entre distintos periféricos o dispositivos AVR. El maestro inicia la comunicación cuando el pin SS se envía a cero. El maestro y el esclavo preparan los datos a ser enviados en su respectivo registro, después de cada transferencia de paquetes de datos, el maestro sincronizara al esclavo, enviando un 1 a la línea SS.

Cuando se configura como maestro, la interfaz SPI no tiene un control automático de la línea SS, esto debe ser configurado por software antes de la comunicación, al realizar esto la comunicación iniciará escribiendo un byte en el SPIDR (Registro de datos), después de enviar el byte se activará SPIF (bandera de transmisión). Cuando es configurado como esclavo, la interfaz SPI se mantendrá desactivada si el pin SS se encuentra en alto, y no se realizará el proceso de transferencia hasta que SS se encuentre en bajo, una vez que termine de recibir datos, SPIF se activará.

La configuración general de la interfaz de SPI se realiza en el registro SPCR (SPI Control Register), la revisión de banderas se encuentra en el registro SPSR (SPI Status Register) y los datos enviados o recibidos se encuentran en el SPDR (SPI Data Register).

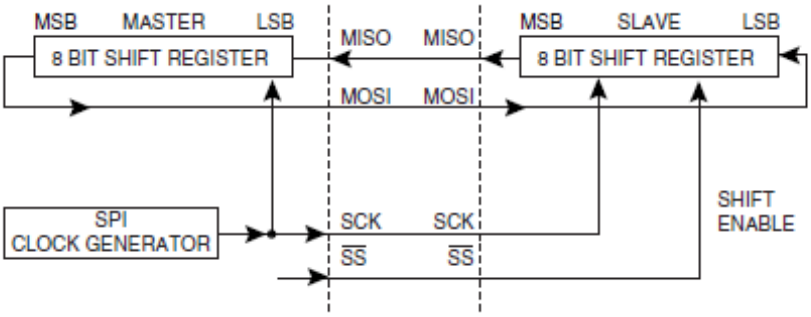


Ilustración 6 Corrimiento SPI

Fuente: (Atmel, 2013).

Módulo MFRC522

El MFRC522 es un lector/escritor para la comunicación inalámbrica a 13.56 MHz, diseñado para la comunicación con tarjetas ISO/IEC 14443 A/MIFARE sin la necesidad de circuitos activos. El módulo logra la funcionalidad completa de encuadre y detección de

error del ISO/IEC 14443 A, así como la transferencia a velocidad alta hasta 848 kBd en ambas direcciones. La distancia típica de operación del módulo puede llegar a los 50 mm. El circuito integrado en el módulo contiene interfaces para SPI, I2C y UART RS232; para el modulo en específico se encuentra habilitado solo para el SPI.(N X P Semiconductors, 2014)

Una interfaz analógica maneja la modulación y demodulación de las señales analógicas. Una interfaz sin contacto UART realiza los protocolos requeridos para el dispositivo anfitrión o maestro. Un buffer tipo FIFO asegura la transmisión de datos de la interfaz sin contacto UART y el anfitrión o maestro.

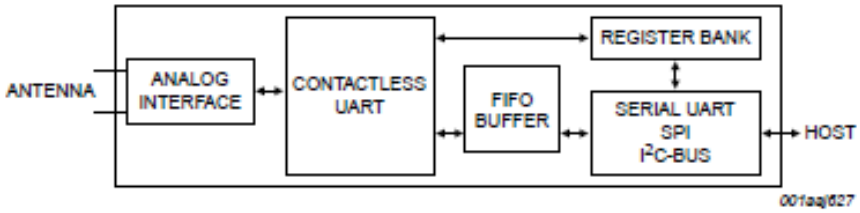


Ilustración 7 Composición módulo

Fuente: (NXP, 2014).

La interfaz SPI puede soportar hasta velocidades de 10 Mbits/s. Al comunicarse con el anfitrión, el MFRC522 actúa como el esclavo, recibiendo datos de un dispositivo maestro, y a su vez, enviando y recibiendo datos por RF. En las líneas MOSI y MISO los datos son enviados con el bit más significativo primero (MSB), deben ser estables en la transición de subida del reloj y pueden ser cambiados durante la transición de bajada.

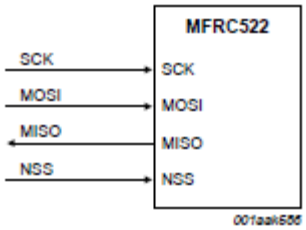


Ilustración 8 MFRC522

Fuente: (NXP, 2014).

Para la lectura de datos se sigue el siguiente orden:

Tabla 1.Lectura de datos MFRC522

Línea	**Byte 0**	**Byte 1**	**Byte 2**	**Byte n**	**Byte n +1**
MOSI	Dirección 0	Dirección 1	Dirección 2	Dirección n	00
MISO	X	Dato 0	Dato 1	Dato n-1	Dato n

Fuente: (NXP, 2014).

Para la escritura de datos se sigue el siguiente orden:

Tabla 2 Escritura de datos MFRC522

Línea	**Byte 0**	**Byte 1**	**Byte 2**	**Byte n**	**Byte n +1**
MOSI	Dirección 0	Dato 0	Dato 1	Dato n-1	Dato n
MISO	X	X	X	X	X

Fuente: (NXP, 2014).

Para la dirección se sigue el siguiente orden:

Tabla 3 Dirección MFRC522

7 (MSB)	**6**	**5**	**4**	**3**	**2**	**1**	**0 (LSB)**
1 = lectura **0 = escritura**	Dirección						0

Fuente: (NXP, 2014).

MIFARE Classic 1k MF1S50yyX

El circuito MIFARE Classic 1K se desarrolló para el uso en tarjetas inteligentes de acuerdo a la norma ISO/IEC 14443 Type A. El circuito cuenta con una función anticolisiones que permite operar más de una tarjeta en el campo al mismo tiempo. El algoritmo de

anticolisión selecciona cada tarjeta individualmente y asegura que la transacción con la tarjeta seleccionada ocurra sin interferencia de alguna otra tarjeta. La tarjeta puede realizar transmisión de datos y energía sin la necesidad de contacto, opera a una frecuencia de 13.56 MHz, cuenta con la integración de datos para verificación de 16 bits CRC y bit de paridad. El tiempo de transmisión es menor a los 100 ms, su distancia de operación puede llegar hasta 100 mm dependiendo del lector y tiene una transferencia de datos de 106 kbits/s.(N X P Semiconductors, 2011)

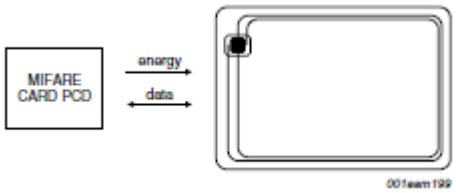


Ilustración 9 Tarjeta MiFire

Fuente: (NXP, 2011).

La memoria EEPROM es de 1kB organizado en 16 sectores de 4 bloques, y cada bloque consiste de 16 byte, a parte el usuario puede definir las condiciones de acceso de cada bloque. Puede retener los datos durante 10 años y soporta hasta 100000 ciclos de escritura.

I2C (Inter-Integrated Circuit).

El bus I2C fue originalmente desarrollado por Philips a principios de los años 80 como una forma sencilla y economía de conectar controladores con circuitos integrados periféricos en equipos de televisión. A pesar de haber sido definido por un fabricante concreto se ha consolidado como un estándar que ha sido adoptado por diferentes fabricantes como interfaz a numerosos tipos de dispositivos como conversores A/D y D/A, memorias, displays, expansores de puertos entro otros periféricos.

La interfaz I2C es un bus síncrono a dos hilos compuesto por una línea de reloj (SCL) y una línea de datos (SDA), ambas líneas son bidireccionales e implementadas eléctricamente en drenador abierto, requiriendo un pull up externo. Además de las líneas

de señal, es evidente que es también necesaria una línea común de retorno que conecte los pines de masa de los diferentes circuitos integrados conectados al bus.

Para permitir el intercambio de información, cada uno de los dispositivos conectados al bus posee una dirección única que lo identifica y puede actuar como transmisor o como receptor dependiendo de sus funciones.

El bus I2C permite configuraciones multimaestro, es decir, pueden coexistir simultáneamente en el mismo bus varios dispositivos con capacidad de controla el bus e iniciar transacciones. Estas transacciones pueden ser tanto de escritura como de lectura, por lo que los dispositivos maestros o esclavos pueden ser bidireccionales.

La máxima velocidad de transmisión de datos por el bus que puede alcanzarse depende del número de circuitos integrados interconectados, puesto que al utilizar tecnología de drenador abierto, el tiempo de subida de las señales en las líneas del bus viene determinado por la mínima resistencia de pull up aplicable y la capacitancia de los circuitos conectados al bus. Esta capacitancia está limitada a 400pF para permitir alcanzar los 100Kb/s. Además de la velocidad estándar, se han establecido en revisiones más recientes un fast mode que permite alcanzar hasta 400Kb/s y un high speed mode con un límite en 3.4Mb/s.

Transmisión de datos.

La transacción de datos por el bus consiste en un paquete de bits que se transmiten serializados por la línea SDA y son validados por la señal de reloj SDA. La transmisión de datos inicia y termina por acción del maestro, quien es además quien controla los flancos de la señal SCL. Para delimitar el comienzo y el final de la transacción de una trama de datos, el dispositivo genera unas condiciones especiales denominadas arranque y parada (start y stop). La condición de arranque se produce cuando se genera un flanco de bajada se presenta en SDA mientras SCL se encuentra en alto, e indica que un maestro toma control del bus. La condición de parada se produce cuando se genera un flanco de subida en SDA mientras SCL permanece en alto, indicando que el maestro liberó el bus.

Los datos se transmiten por la línea siempre en bloques de 8 bits. Por cada byte transmitido, el dispositivo que acepta la información genera una señal de aceptación o

reconocimiento (acknowledge), forzando un nivel bajo en la línea SDA, cuando no se reconoce se genera una señal de rechazo (not acknowledge).

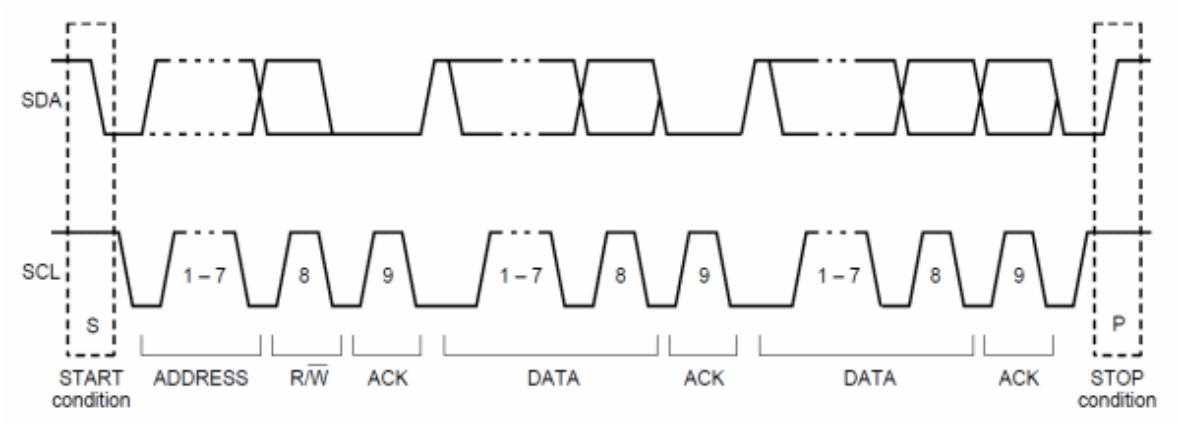


Ilustración 10 Trama I2C

Fuente: (NXP, 2014).

Sincronización y control de flujo

En el bus, el maestro puede efectuar dos tipos de operaciones sobre el esclavo, las cuales son lectura y escritura.

Escritura:

- El maestro genera una condición de arranque.
- El maestro envía el primer byte, de los cuales los 7 MSB corresponden a la dirección del esclavo, mientras que el último bit se pone en 0 para indicar que se desea escribir.
- El esclavo genera un reconocimiento (acknowledge).
- El maestro envía n cantidad de bytes de datos, que contienen la información útil, cada uno seguido de un reconocimiento producido por el esclavo
- Una vez terminada la transacción, el maestro produce una condición de parada

Lectura:

- El maestro genera una condición de arranque.

- El maestro envía el primer bye, de los cuales los 7MSB corresponden a la dirección del esclavo, mientras que el último bit se pone en 1 para indicar que se desea efectuar una lectura.
- El esclavo toma el control de SDA y el maestro genera un reconocimiento (acknowledge).
- El esclavo envía n cantidad de bytes de datos, que contiene la información, cada uno seguido de un reconocimiento producido por el maestro.
- Cuando se desea finalizar, el maestro genera un rechazo en el último byte (not acknowledge) y produce una condición de parada.

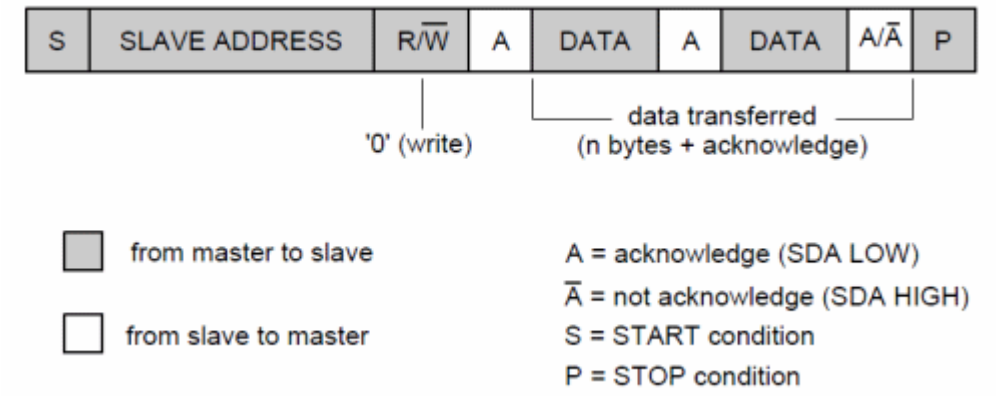


Ilustración 11 Composición de la trama I2C

Fuente: (NXP, 2014).

Sensor de efecto Hall.

El efecto Hall, descubierto por Edwin C. Hall en 1879, consiste en la producción de una caída de voltaje a través de un conductor o semiconductor con corriente, bajo la influencia de un campo magnético externo. Para esto es necesario que la dirección del campo magnético sea perpendicular a la dirección de flujo de la corriente. El campo magnético transversal ejerce una fuerza sobre el conductor o semiconductor. Esta fuerza causa la desviación de los portadores de carga que se mueven a través del material. Como resultado, aparece una diferencia de potencial llamada voltaje de Hall entre los extremos del conductor.

Existen dos tipos de sensores de efecto Hall, los sensores de salida lineal, y los de salida en latch (salida digital). Los sensores Hall digitales entregan dos estados lógicos;

cuando son sometidos a un campo magnético norte, se activan y mandan su salida a estado alto, y se mantienen en dicho estado hasta que un campo inverso actúa sobre ellos. En cambio, los sensores Hall lineales, entregan un voltaje en proporción a la magnitud y sentido del campo magnético incidente; internamente poseen un amplificador diferencial que incrementa las variaciones de tensión respecto al campo, pero el amplificador incorpora una tensión de polarización, de modo que con un campo magnético nulo tengamos en la salida una tensión positiva. Esta tensión es la que se conoce como null offset. (Martín, n.d.)

Los sensores de efecto Hall tienen un amplio campo de aplicación en la medición de corrientes y velocidad de motores. En nuestro proyecto, el sensor de efecto Hall se vuelve necesario ya que el sistema de cobro requiere detectar la denominación de monedas. Al colocar el sensor entre un campo y una moneda se detectará un cambio de tensión que será único para cada denominación de la moneda. El valor analógico de la moneda será digitalizado y posteriormente procesado.

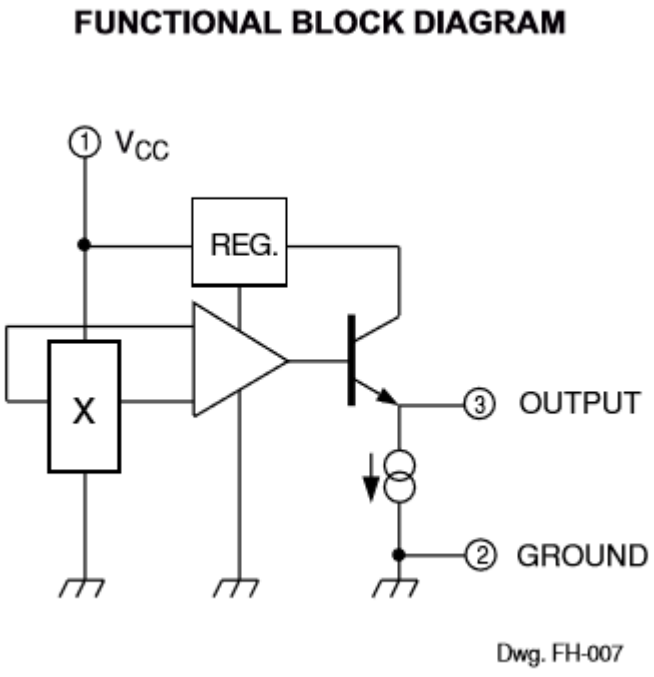


Ilustración 12 Diagrama a bloques del sensor Hall lineal

Fuente: (Allegro MicroSystems, 2013).

AVR.

Los AVR son una familia de microcontroladores RISC del fabricante estadounidense Atmel. La arquitectura de los AVR fue concebida por dos estudiantes en el Norwegian Institute of Technology, y posteriormente refinada y desarrollada en Atmel Norway, la empresa subsidiaria de Atmel, fundada por los dos arquitectos del chip. El AVR es una CPU de arquitectura Harvard. Tiene 32 registros de 8 bits.

Algunas instrucciones sólo operan en un subconjunto de estos registros. La concatenación de los 32 registros, los registros de entrada/salida y la memoria de datos conforman un espacio de direcciones unificado, al cual se accede a través de operaciones de carga/almacenamiento. A diferencia de los microcontroladores PIC, el stack se ubica en este espacio de memoria unificado, y no está limitado a un tamaño fijo.

Se pueden dividir en los siguientes grupos:

- ATxmega: procesadores muy potentes con 16 a 384 kB de memoria flash programable, encapsulados de 44, 64 y 100 pines (A4, A3, A1), capacidad de DMA, eventos, criptografía y amplio conjunto de periféricos con DACs.
- ATmega: microcontroladores AVR grandes con 4 a 256 kB de memoria flash programable, encapsulados de 28 a 100 pines, conjunto de instrucciones extendido (multiplicación y direccionamiento de programas mayores) y amplio conjunto de periféricos.
- ATtiny: pequeños microcontroladores AVR con 0,5 a 8 kB de memoria flash programable, encapsulados de 6 a 20 pines y un limitado set de periféricos.
- AT90USB: ATmega integrado con controlador USB.
- AT90CAN: ATmega con controlador de bus CAN.
- Tipos especiales: algunos modelos especiales, por ejemplo, para el control de los cargadores de baterías, pantallas LCD y los controles de los motores o la iluminación.

- AT90S: tipos obsoletos, los AVRs clásicos.

Para maximizar el rendimiento y paralelismo, el AVR usa una arquitectura Harvard con memorias y buses separados para instrucciones y datos. Las instrucciones en la memoria de programa son ejecutadas con una segmentación de dos etapas. Mientras una instrucción está siendo ejecutada, la siguiente es pre-capturada de la memoria de programa. El archivo de registros contiene 32 registros de propósito general de 8 bits, habilitados para un acceso rápido.

Seis de los 32 registros pueden ser usados como apuntadores de 16 bits para direccionamiento indirecto en el espacio de datos, habilitando cálculos de direcciones eficientes. La mayoría de instrucciones AVR tienen un formato en una palabra de 16-bits. En un ciclo de reloj se pueden leer 2 registros que funcionen como operandos para la ALU, que se realice la operación y el resultado quede disponible para escribirse en uno de esos registros. Para la ejecución, la duración del ciclo es suficiente para permitir la lectura de registros, la operación de la ALU y la escritura en el registro destino. (Espinosa, 2016)

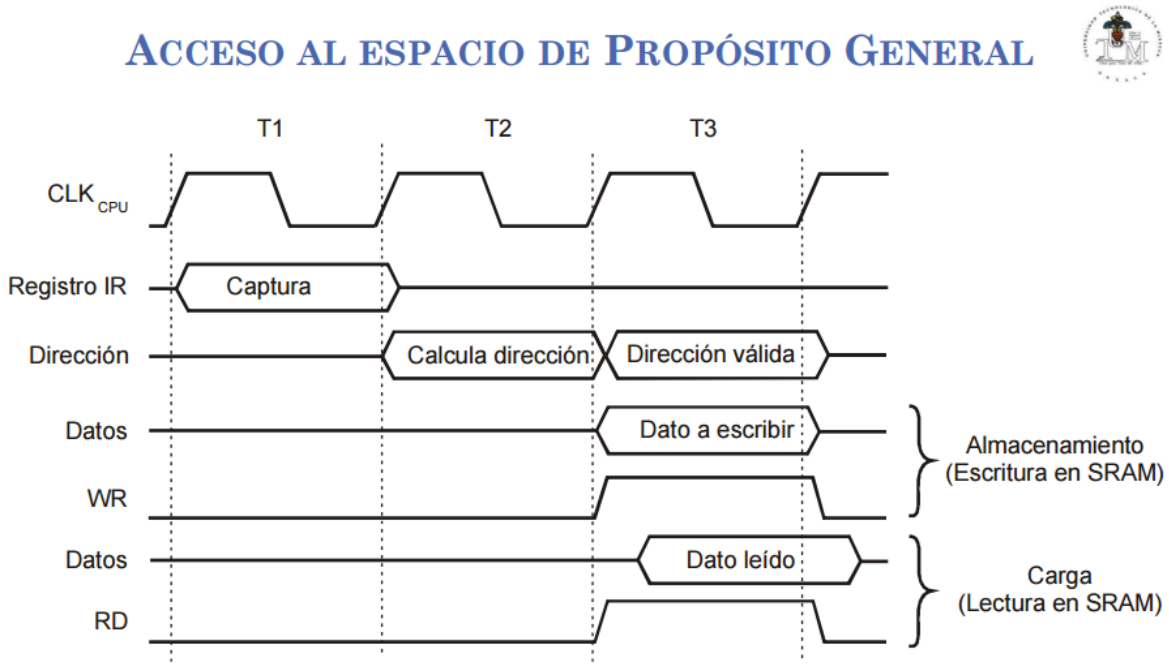


Ilustración 13 Direccionamiento AVR

Fuente: (UTM, 2013).

A continuación, se hará referencia a los dos microcontroladores principales del proyecto.

ATmega8 y ATmega16

El ATmega8 y ATmega16 tienen cuatro fuentes de inicialización o reset.

- El MCU es inicializado cuando el voltaje de la fuente está por debajo del voltaje de umbral de encendido.
- **Reset externo:** El MCU es inicializado cuando un nivel bajo está presente en la terminal RESET por un tiempo mayor que la longitud mínima del pulso.
- **Watchdog reset:** El MCU es inicializado cuando el watchdog timer está habilitado y su periodo termina.
- **Brown-out reset:** El MCU es inicializado cuando el detector de reducción de voltaje está habilitado y el voltaje VCC de la fuente va por debajo del umbral establecido.

Una vez que el MCU se recupera de una condición de reset, espera un tiempo de establecimiento de 4 ms, antes de recuperar al sistema para garantizar que los registros tienen su valor inicial.

El registro MCUCSR posee 8 bits que almacenan la información de la fuente de inicialización o reset. En dicho registro se puede configurar la alternativa deseada para este proceso. En el caso especial del microcontrolador ATmega16, en dicho registro se encuentran disponibles en los bits 7 y 6 JTD e ISC2 que se configuran para el uso de ciertas alternativas de los I/O puertos. (Atmel, 2010)

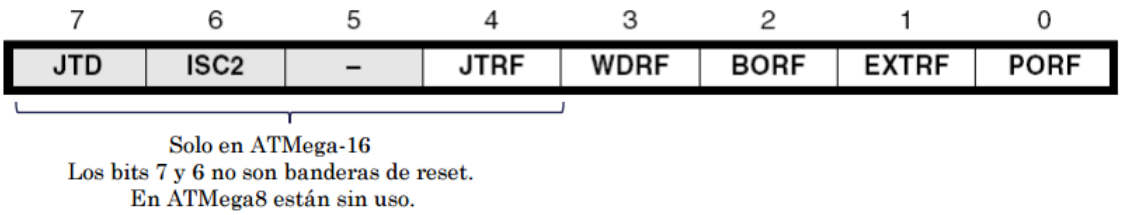


Ilustración 14 Registro MCUCSR

Fuente: (Atmel, 2010).

Se tienen diferentes fuentes para manejar al oscilador interno y a la vez, las señales de reloj se distribuyen por los módulos.

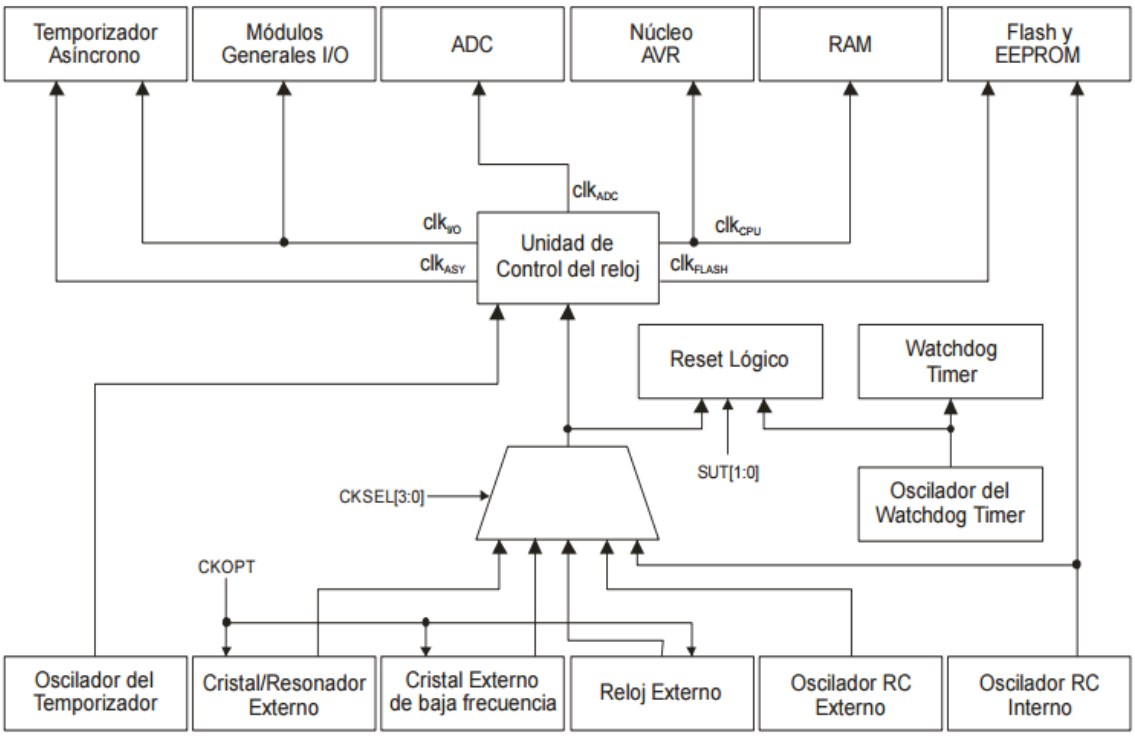


Ilustración 15 Fuentes de reloj AVR

Fuente: (Atmel, 2010).

La unidad de control del reloj se encarga de generar diferentes señales de reloj y distribuirlas a los diferentes módulos, las diferentes señales de reloj son:

- **clkCPU:** Ruteado al núcleo AVR, incluyendo al archivo de registros, registro de estado, memoria de datos, apuntador de pila, etc.

- **clkFLASH:** Señal de reloj suministrada a las memorias FLASH y EEPROM.
- **clkADC:** Reloj dedicado al ADC. Trabaja a una frecuencia menor que la CPU con el objetivo de reducir el ruido generado por interferencia digital y mejorar las conversiones.
- **clkI/O:** Es la señal de reloj utilizada por los principales módulos de recursos: Temporizadores, interfaz SPI y USART. Además de ser requerido por el módulo de interrupciones externas.
- **clkASY:** Esta señal de reloj es asíncrona con respeto al resto del sistema y es empleada para sincronizar al temporizador 2, el módulo que genera esta señal está optimizado para manejarse con un cristal externo de 32.768 kHz. Frecuencia que permite usar al temporizador como un contador de tiempo real, aun cuando el dispositivo está en algún modo de reposo.

La fuente de reloj se selecciona con 4 bits de configuración internos CKSEL y 2 bits SUT. Se tienen las opciones:

- Resonador cerámico o cristal externo
- Cristal de baja frecuencia externo
- Oscilador RC externo
- Oscilador RC calibrado interno
- Reloj externo

Es muy importante destacar que para el caso del microcontrolador ATmega8, se requiere una configuración forzosa de la fuente de reloj, ya que no existe un método de adquirir la señal de sincronía sin especificarlo mediante hardware, ya que las terminales o pines externos XTAL1 y XTAL2, comparten con pines I/O del puerto B, en cambio en el ATmega16, queda implícita la fuente de reloj mediante hardware, ya que dichas terminales se encuentran separadas de los pines I/O.

En general ambos microcontroladores están basados en la misma arquitectura, y por ende comparten mapas de memoria, tiempos de ejecución de instrucciones, modos de ahorro y distribución de energía, manejo de vectores de interrupción etc. Lo que los hace ser diferentes, es la disponibilidad y el acceso de registros.

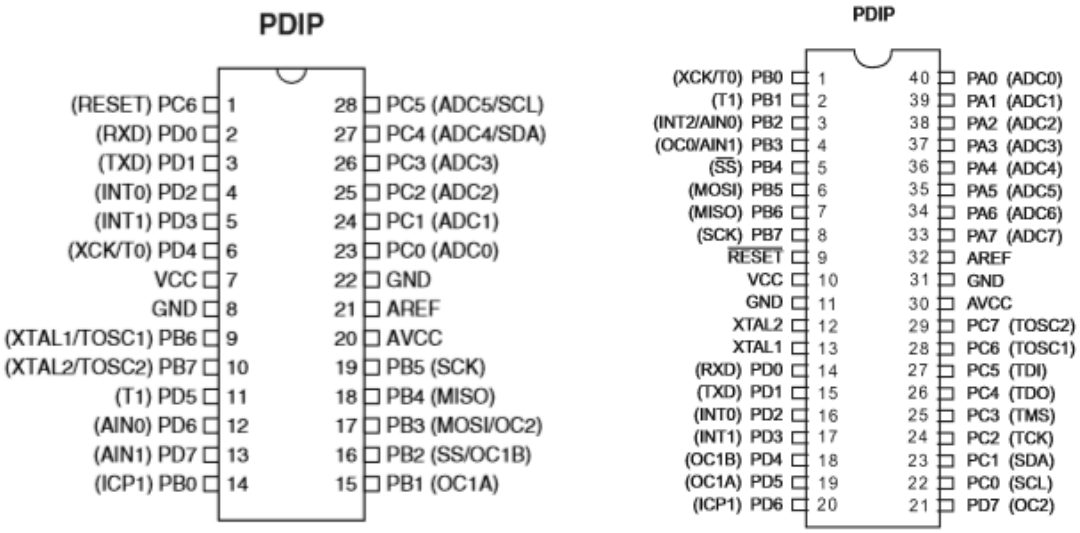


Ilustración 16 ATmega 8 y ATmega 16

Fuente: (Atmel, 2013).

RTC (Real Time Clock).

Un RTC es un dispositivo que emplea una base de tiempo implementada por un oscilador, para llevar a cabo una cuenta de tiempo. Resultan indispensables cuando en una aplicación se requiere llevar a cabo el registro de un evento en el tiempo.

El DS1307 es un RTC cuya interfaz con el microcontrolador se realiza utilizando el bus I2 C. Es un reloj calendario en BCD, cuyas características más destacadas son las siguientes:

- Reloj de tiempo real que cuenta los segundos, los minutos, las horas, la fecha, el mes, el día de la semana, y el año, con compensación de años bisiestos, válido hasta el año 2100.
- Formato de 12 Horas con indicador AM/PM o de 24 horas.
- Emplea protocolo I2C.
- 56 bytes de RAM no volátil, para almacenamiento de datos.

- Señal de onda cuadrada programable.
- Circuitos internos de respaldo para la alimentación automático.
- Bajo consumo de potencia: menor a 500nA en modo respaldo, a 25 grados centígrados.

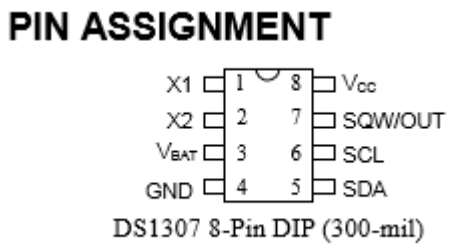


Ilustración 17 DS1307

Fuente: (Dallas Semiconductors, 2013).

A continuación, se describe la asignación de pines del DS1307:

- **VCC y GND:** Terminales por la que suministra la tensión requerida por el circuito integrado. Típicamente Vcc = +5V mientras que Gnd es la referencia.
- **X1 y X2:** Conexiones para un cristal de cuarzo estándar de 32.768 kHz.
- **VBAT:** En esta terminal se suministra la tensión de emergencia, por medio de una batería de litio de 3V. Su función es reemplazar VCC cuando esta es 0V.
- **SQW/OUT:** Es una terminal de salida, que produce una base de tiempo a 1Hz, 4kHz, 8kHz, o 32kHz cuando es configurada por medio de software. Al igual que SCL y SDA, requiere un pull up para operar por drenador abierto.
- **SCL:** Señal de reloj del protocolo I2C.
- **SDA:** Línea de datos del protocolo I2C.

Mapa de memoria.

El DS1307 como se hizo mención posee un mapa de memoria distribuido en 8 registros de lectura y escritura que contienen la información de la fecha y hora actual, además de una memoria RAM no volátil de 56 x 8 de propósito general.(Narváez, n.d.)

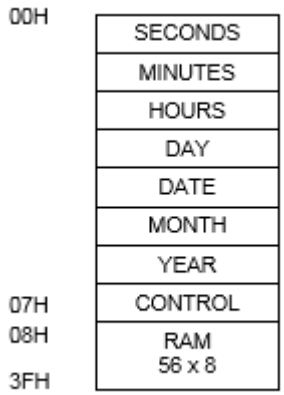


Ilustración 18 Mapa de memoria DS1307

Fuente: (Dallas Semiconductors, 2013).

La información del tiempo y el calendario está en formato BCD, y es obtenida leyendo los registros apropiados. Los bits marcados con 0 en esta figura son intrascendentes para el funcionamiento del circuito. El bit 7 del registro 0 es el bit CH (Clock Halt). Cuando este bit se establece a 1, el oscilador se deshabilita, deteniendo el funcionamiento del reloj. Cuando se establece a 0, el oscilador es habilitado. El RTC, puede trabajar en formato de 12 horas o de 24 horas siendo el bit 6 del registro 2 el encargado de determinar esta función. Si se establece a 1, funciona en modo 12 horas, y el bit 5 es el indicador AM/PM, siendo 1 cuando es PM. En formato 24 horas, el bit 5 de este registro es la parte más significativa de las decenas de horas.

	BIT7							BIT0	
00H	CH	10 SECONDS			SECONDS				00–59
	0	10 MINUTES			MINUTES				00–59
	0	12 / 24	10 HR / A/P	10 HR	HOURS				01–12 00–23
	0	0	0	0	0	DAY			1–7
	0	0	10 DATE		DATE				01–28/29 01–30 01–31
	0	0	0	10 MONTH	MONTH				01–12
	10 YEAR				YEAR				00–99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	

Ilustración 19 Registros de hora y fecha

Fuente: (Dallas Semiconductors, 2013).

El registro 7 es el registro de control. Este registro es utilizado para determinar el funcionamiento del pin de salida SQW/OUT. El bit 7 OUT determina el nivel de salida cuando la salida de onda cuadrada esta deshabilitada (SQWE = 0). Será alto si el bit OUT = 1 y bajo si OUT = 0. El bit 4 o SQWE, habilita la salida del oscilador de onda cuadrada, cuya frecuencia dependerá de los bits RS1 y RS0.(DALLAS SEMICONDUCTORS, n.d.)

RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1Hz
0	1	4.096kHz
1	0	8.192kHz
1	1	32.768kHz

Ilustración 20 Modos de configuración de RS

Fuente: (Dallas Semiconductors, 2013).

Protocolo I2C.

Considerando que la dirección del dispositivo es 0b1101000x donde x representa el bit de lectura o escritura, se describe la generación de la trama del dispositivo:

Escritura:

- Generación de condición de inicio.
- Envío de dirección de escritura 0xD0.
- Envió de dirección de memoria a escribir.
- Envió de datos, considerando que el apuntador de registro del DS1307 incrementa una unidad después de cada transacción.
- Generación de condición de parada.

Lectura:

- Generación de condición de inicio.
- Envió de dirección de lectura 0xD1.
- Recepción de datos, considerando que el primer byte corresponde al direccionado por el apuntador de registro, y cada byte posterior es direccionado por un incremento de una unidad en dicho apuntador.
- Generación de condición de parada.

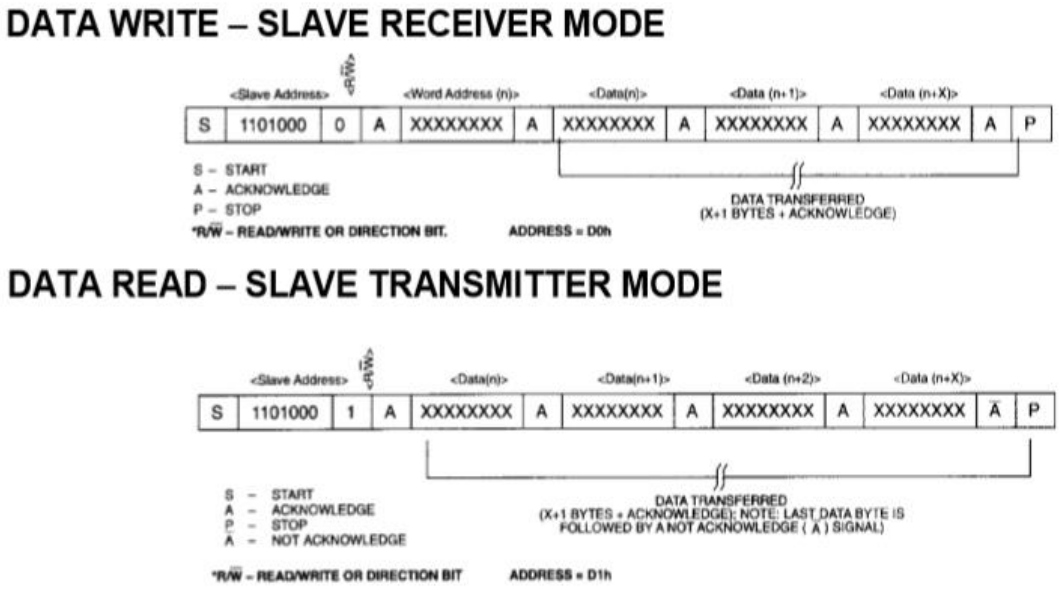


Ilustración 21 Modos de lectura y escritura

Fuente: (NXP, 2010)

App Inventor

App Inventor es una plataforma originalmente desarrollada por Google, y mantenida ahora por el MIT (Massachusetts Institute of Technology), que permite a cualquier persona, aunque no esté familiarizada con la programación y SDK (Software development kit) de Android, llevar a cabo la interfaz gráfica y la lógica de programación necesaria para crear una aplicación sin escribir código en Java.

Utilizando una interfaz gráfica amigable, permite al usuario arrastrar y soltar objetos visuales en forma de bloques para crear un ejecutable en el sistema operativo Android.

La aplicación se puso a disposición de los usuarios, desde el 15 de diciembre del 2010, para todos los usuarios registrados con una cuenta de correo electrónico de Google.

App Inventor se salva y ejecuta vía web, por lo que se puede acceder a nuestros proyectos desde cualquier equipo conectado a Internet.

Permite también guardar el proyecto en la PC, para enviarlo a otros usuarios o hacer una copia de seguridad, así como descargar la aplicación. apk necesaria para la instalación en los dispositivos con sistema operativo Android.

Este entorno de desarrollo es compatible con Mac OS X, GNU / Linux, y los sistemas operativos Windows, y algunos de los modelos más populares de Android. (Ramirez,2011)

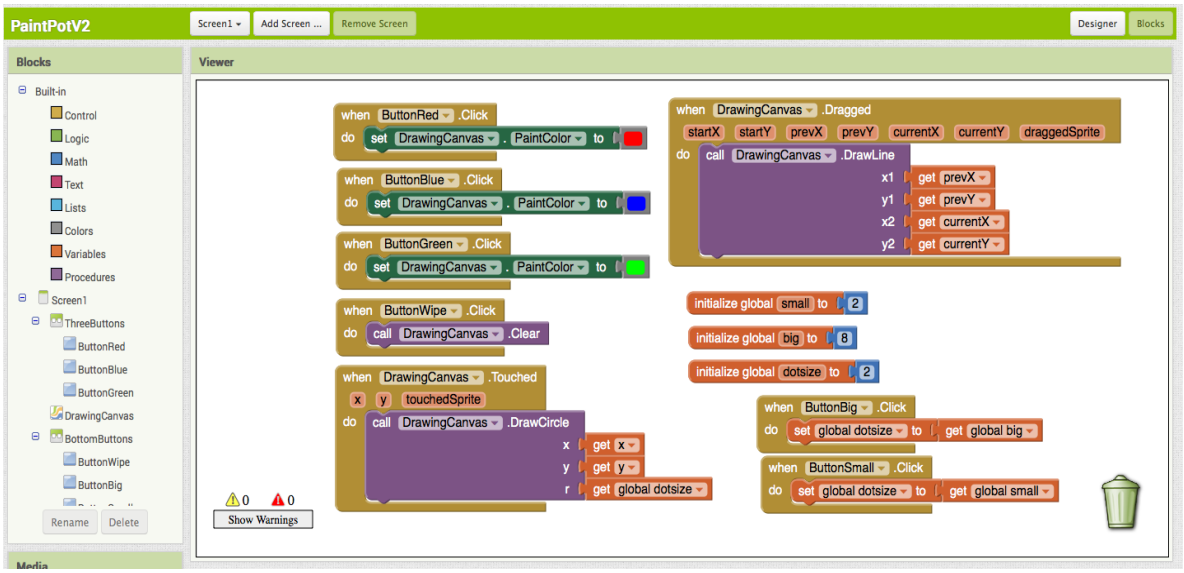


Ilustración 22 Plataforma App Inventor

Fuente: (MIT, 2016).

MIT AI2 Companion

Al momento de estar diseñando una aplicación con ayuda de App Inventor, resulta necesario visualizar en tiempo real los cambios realizados desde la plataforma en línea, y esto es posible gracias a MIT AI2 Companion, una aplicación gratuita y disponible en Play Store. Después de instalar y abrir la aplicación en el teléfono, basta con introducir el código de 6 dígitos, o capturar el QR Code generado en App Inventor en el campo de “Conectar”, y la aplicación realizada aparecerá en el Smartphone paso a paso a medida que se añaden piezas a la misma, para poder probar el trabajo a medida que se construye, cabe mencionar que ambos dispositivos deben estar conectados a internet y con la misma dirección IP.

TinyDB y TinyWebDB

App Inventor proporciona 2 componentes para facilitar al usuario el uso de base de datos: TinyDB y TinyWebDB.

TinyDB es un componente no visible para almacenar datos directamente en el dispositivo Android, que estarán disponibles cada vez que la aplicación se ejecute. Las aplicaciones creadas en App Inventor se inicializan cada vez que se abre el activity principal, es decir si en algún momento se establece el valor de una variable y el usuario sale de la aplicación, el valor de esa variable no será recordado la próxima vez que se ejecute la aplicación, sin embargo al utilizar el componente TinyDB se almacena en el dispositivo el valor asignado a la variable en cuestión, y al abrir de nuevo la aplicación será recordado, esto es muy utilizado en juegos en donde se guarda la puntuación más alta, o el último escenario en donde se perdió una partida.

Los elementos de texto, cadenas o listas se almacenan en TinyDB mediante etiquetas. Para almacenar un elemento se debe mandar a llamar la función TinyDBx.StoreValue, especificando la etiqueta, que debe ser una cadena de texto, y el valor a almacenar que puede ser una cadena de texto o una lista.

Posteriormente para recuperar la información almacenada, se manda a llamar la función TinyDBx.GetValue solamente con la etiqueta de donde se desea obtener el valor, si no hay ningún valor almacenado en la etiqueta devuelve el texto vacío.

Para limpiar la base de datos para una aplicación, vaya en el teléfono bajo Configuracion > Aplicaciones > Administrar aplicaciones, en la opción de “Borrar datos de la aplicación”

Los datos de TinyDB son persistentes solo cuando se han empaquetado se descarga la aplicación. Si está desarrollando conectado al teléfono y reinicia AppInventor, o si resetea la conexión la base de datos será inicializada de nuevo.

TinyWebDB es otro componente más de AppInventor que le permite almacenar al usuario datos de forma persistente ya no en el dispositivo móvil, si no en una base de datos en la web. Debido a esto, este elemento se puede utilizar para facilitar la comunicación entre los teléfonos y las aplicaciones.

De forma predeterminada, el TinyWeb almacena datos en un servicio de prueba proporcionados por la aplicación de Inventor http://appinvtinywebdb.appspot.com/. Sin embargo, este servicio es útil para las pruebas, pero es compartida por todos los usuarios de Inventor de la aplicación, y tiene un límite de 1000 entradas. Si lo usa, sus datos se sobren escriben con el tiempo.

Básicamente el proceso de Store & Get (Almacenar y Descargar) datos es el mismo que en TinyDB, se utilizan etiquetas que no son más que cadenas de texto fungiendo como identificadores, solo que ahora la aplicación no buscará el tag en el dispositivo, se direccionará inmediatamente al servidor web de App Inventor, para buscar la etiqueta requerida por el usuario y posteriormente obtendrá el valor almacenado.(MIT,2014)

Capítulo 2: Diseño

Máquina de prepago.

Detección de monedas (Sensor Hall).

En la máquina de prepago se requiere averiguar la denominación de cada moneda que el usuario introduce. Las monedas de 1, 2, 5 y 10 pesos poseen propiedades magnéticas diferentes, por lo que su detección se puede realizar con el sensor explicado en el capítulo 1. Este sensor tiene una referencia de voltaje de CD cuando se aplica un campo magnético constante; dicho campo se mantendrá en la misma magnitud mientras no se introduzca ninguna moneda al sistema, en caso contrario provocará una perturbación en las líneas de flujo y por ende en la tensión del sensor de efecto Hall.

Debido a esto, se intuye que la información útil que arroja el transductor se encuentra en un formato diferencial. Otra consideración que se debe tener presente es que el acondicionador de señal que se debe diseñar, debe amplificar las pequeñas variaciones generadas con cada denominación de moneda, para obtener un nivel apto para que en una etapa siguiente se le pueda dar el adecuado procesamiento a la señal. Una solución que cubre estos requerimientos es hacer uso de la entrada diferencial que poseen los amplificadores operacionales.

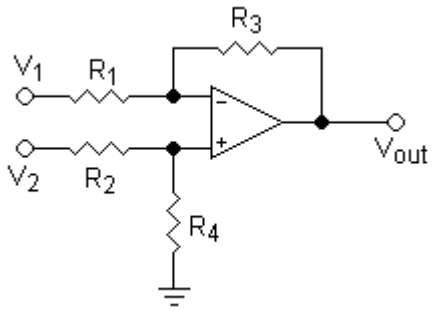


Ilustración 23 Amplificador diferencial

Fuente: (Boylestad,2013).

La configuración que se muestra en la figura anterior es un amplificador diferencial, su respuesta a una señal de entrada se describe por la ecuación:

Ecuación 1

$$Vo = V1\left(\frac{R3}{R1}\right) - V2\left(\frac{R4}{R2}\right)$$

Fuente: (Boylestad,2013).

Para efectos de simplificación, si logramos que $R1 = R2 = R1$ y que $R3 = R4 = R2$ se obtiene:

Ecuación 2

$$Vo = V1\left(\frac{R1}{R2}\right) - V2\left(\frac{R1}{R2}\right)$$

Fuente: (Boylestad,2013).

Factorizando el término común:

Ecuación 3

$$Vo = (V1 - V2)\left(\frac{R1}{R2}\right)$$

Fuente: (Boylestad,2013).

Puesto que el factor que nos incumbe es la ganancia del sistema se puede concluir que:

Ecuación 4

$$\Delta V = \frac{R1}{R2}$$

Fuente: (Boylestad,2013).

En donde ΔV representa Vi-Vi', es decir la información diferencial del sensor.

Es preciso señalar que la entrada inversora del amplificador diferencial estará conectada a la salida del sensor de efecto Hall, es decir Vi_1 y la entrada no inversora deberá colocarse a un valor de voltaje de referencia idéntico al que entrega el sensor en

condiciones regulares, es decir Vi_2. Esta referencia será calibrada mediante el uso de un divisor de voltaje variable implementado a través de un resistor variable de precisión.

Detección de monedas (Conversión digital).

La etapa siguiente consiste en digitalizar la señal que otorga el acondicionador de señal propuesto. La solución más sencilla a esto es mediante un convertidor analógico digital tipo flash. Este convertidor hace uso de una serie de comparadores en paralelo con referencias de voltaje en distintos valores. Un comparador es un circuito que entrega dos niveles de acuerdo al signo obtenido mediante una resta de dos señales de entrada. La implementación de un comparador se puede lograr nuevamente, aprovechando las características de los amplificadores operacionales, ya que estos poseen una entrada diferencial y una ganancia a lazo abierto infinita.

La operación aplicada por el circuito anterior es la siguiente:

Ecuación 5

$$Vo = V^{+} \mid Vi_1 > Vi_2$$

$$Vo = V^{-} \mid Vi_1 < Vi_2$$

Fuente: (Boylestad,2013).

En seguida se muestra un convertidor analógico digital tipo Flash:

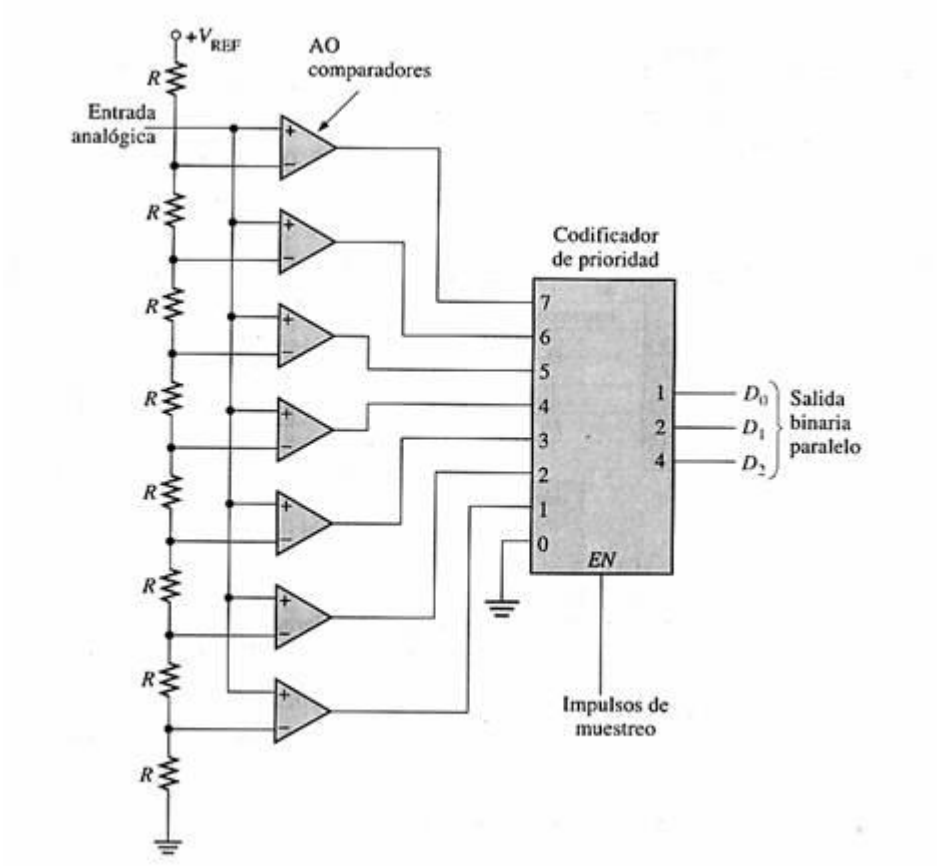


Ilustración 24 ADC Flash

Fuente: (Boylestad,2013).

Para 4 monedas se requieren como mínimo 3 comparadores paralelos, los cuales darán como posibles combinaciones de salida:

Tabla 4 Asignación de valores a monedas

Denominación de moneda	Valor digital ADC Flash
$10	000
$1	001
$2	011
$5	111

Fuente: (Elaboración propia).

La asignación de valores y la interpretación de estos se realizará por medio de software, implementado a través del ATmega16.

Máquina de descuento de saldo

Configuración de fuente de reloj ATmega8.

Antes de comenzar con el diseño, en necesario retomar un punto muy importante en el uso del ATmega8; este modelo requiere para su funcionamiento, determinar la fuente de reloj de la que tomará la base de tiempo para realizar sus operaciones.

La alternativa que se decidió para la fuente de reloj, es hacer uso del oscilador interno de 8MHz, pensando en que, en una mejora futura, se requiera hacer uso de los pines I/O del PORTB.

Para configurarlo se habilitarán los fusibles correspondientes haciendo uso de un programador. En acorde a las hojas de datos del fabricante, el mapa de fusibles se divide en dos bytes. En el byte alto encontramos los siguientes:

Fuse High Byte	Bit No.	Description	Default Value
RSTDISBL[4]	7	Select if PC6 is I/O pin or RESET pin	1 (unprogrammed, PC6 is RESET-pin)
WDTON	6	WDT always on	1 (unprogrammed, WDT enabled by WDTCR)
SPIEN[1]	5	Enable Serial Program and Data Downloading	0 (programmed, SPI prog. enabled)
CKOPT[2]	4	Oscillator options	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed, EEPROM not preserved)
BOOTSZ1	2	Select Boot Size (see Table 82 on page 213 for details)	0 (programmed)[3]
BOOTSZ0	1	Select Boot Size (see Table 82 on page 213 for details)	0 (programmed)[3]
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Ilustración 25 Byte alto fusibles

Fuente: (Atmel, 2013).

Y en el byte bajo encontramos:

Fuse Low Byte	Bit No.	Description	Default Value
BODLEVEL	7	Brown out detector trigger level	1 (unprogrammed)
BODEN	6	Brown out detector enable	1 (unprogrammed, BOD disabled)
SUT1	5	Select start-up time	1 (unprogrammed)(1)
SUT0	4	Select start-up time	0 (programmed)(1)
CKSEL3	3	Select Clock source	0 (programmed)(2)
CKSEL2	2	Select Clock source	0 (programmed)(2)
CKSEL1	1	Select Clock source	0 (programmed)(2)
CKSEL0	0	Select Clock source	1 (unprogrammed)(2)

Ilustración 26 Byte bajo fusibles

Fuente: (Atmel, 2013).

De los cuales es requerido configurar los CKSEL0..3 y el CKOPT, sin modificar los valores predefinidos de los otros fusibles.

CKSEL3..0	Nominal Frequency (MHz)
0001(1)	1.0
0010	2.0
0011	4.0
0100	8.0

Ilustración 27 Registro CLKSEL

Fuente: (Atmel, 2013).

En acorde a la tabla anterior, extraída de la hora de datos del fabricante, podemos observar que para configurar el oscilador interno a 8Mhz se activaran los fusibles CKSEL3, CKSEL1 y CKSEL0, considerando que la lógica de la tabla está negada. La condición para que esto suceda es que CKOPT se encuentre desactivado, por lo que CKOPT = 1.

Configuración I2C en ATmega8.

Por cuestiones de derechos de autor, en los microcontroladores AVR se hace referencia al protocolo I2C, como TWI (Two Wire Interface), por lo tanto todos los registros poseen dicha nomenclatura. Para generar una trama de I2C, será necesario realizar las funciones de configuración correspondientes en los registros del TWI, para generar las condiciones de inicialización I2C, inicio, parada, envió y recepción de datos.

Inicialización I2C

Para encender el módulo de I2C del microcontrolador, se requiere establecer el Baud Rate al cual, operará la comunicación. En nuestro caso la frecuencia a la que trabajaremos es de 100kHz. De acuerdo a la ecuación obtenida de las hojas de datos podemos establecer:

Ecuación 6

$$SCL\ frequency = \frac{CPU\ Clock\ frequency}{16 + 2(TWBR) * 4^{TWPS}}$$

Fuente: (Atmel, 2013).

Si hacemos que los bits TWPS1 y TWPS2 en el registro TWSR sean 0, obtendremos un preescalador de 1, es decir TWPS=1, si despejamos TWBR de la ecuación, se consigue:

Ecuación 7

$$TWBR = \left(\frac{1}{8}\right)\left(\frac{CPU\ Clock\ frequency}{SCL\ frequency} - 16\right)$$

Fuente: (Atmel, 2013).

Sustituyendo los valores deseados encontramos:

Ecuación 8

$$TWBR = \left(\frac{1}{8}\right)\left(\frac{8MHz}{100kHz} - 16\right) = 8$$

Fuente: (Atmel, 2013).

Por ende el valor cargado al registro TWBR será TWBR=0x08.

Para habilitar el reloj del módulo se escribe un 1 en el bit TWEN en el registro TWCR, y de esta forma terminamos la configuración de inicialización.

Condición de inicio.

Para dar la condición de inicio se activan en el registro TWCR los bits 7, 5 y 2 correspondientes a TWINT, TWSTA y TWEN. TWINT es encendida por hardware cuando se genera una interrupción, y es limpiada mediante software escribiendo un 1, por tal motivo es que se debe limpiar antes de cada ejecución referente al TWI. El bit TWEN habilita el módulo, mientras que al escribir 1 en TWSTA se genera la condición de arranque, por ende TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN)

Esta actividad causará que la bandera de interrupción se habilite, así que tenemos que esperar este evento para asegurarnos de que ya dio la condición de arranque. Esto puede lograr mediante un bucle while.

Condición de fin.

La condición de fin se genera habilitando el bit TWSTO en el registro TWCR, no sin antes haber limpiado la bandera de interrupción y habilitar el módulo. De forma análoga a la condición de inicio se produce TWCR = (1<<TWINT) | (1<<TWSTO) | (1<<TWEN)

Envío de datos.

Para realizar el envío de un byte de datos este debe estar precargado en el registro TWDR, el cual se encarga de serializar la información. Para transmitir dicho dato cargado, se escribe el TWCR los bits que limpian la interrupción y que activan el módulo. Al activar el módulo de manera automática el valor cargado en el TWDR se reflejará en la línea de datos SDA. Para asegurarnos de que el dato se transmitió de manera correcta, debemos esperar el reconocimiento (acknowledge), el cual hace que se active la bandera de interrupción. Para esperar el reconocimiento, tenemos que verificar que TWINT sea 0.

Recepción de datos.

Este proceso es similar al de envió, pero en orden reverso. Primero limpiamos la bandera de interrupción y habilitamos el módulo; después tenemos que verificar la bandera, la cual nos indica que hay un dato disponible en el TWDR. Cuando la bandera se active, el valor recibido en la línea SDA, estará guardado en dicho registro.

Lectura y escritura del DS1307

El proceso de lectura y escritura en el DS1307, se construye una vez que se han generado las funciones de configuración para la implementación del protocolo TWI.

Retomando la información que se presentó en el capítulo pasado, el RTC en cuestión está conformado por una serie de registros de los cuales 8 son configurables para la funcionalidad del CI y los restantes fungen como una memoria RAM no volátil.

Los datos que nos interesan escribir son estos primeros 8 registros, ya que son imprescindibles para el adecuado funcionamiento del módulo.

La trama que construiremos, por tanto, será:

- Condición de inicio
- Envío de dirección de esclavo para escritura (0xD0)
- Envío de dirección de registro de segundos (0x00)
- Envío de valor numérico de segundos
- Envío de valor numérico de minutos
- Envío del valor numérico de horas
- Envío del valor numérico de día de la semana
- Envío del valor numérico de día del mes
- Envío del valor numérico del mes
- Envío del valor numérico del año
- Envío del valor numérico para configuración de SQWE (0x90)
- Condición de paro

Como puede notarse resulta un proceso poco eficiente y repetitivo, de tal forma que la manera de reducirlo es empleando un ciclo for, que permita desplazar datos desde un buffer del microprocesador, hacia una función de envío. Las condiciones de configuración del protocolo I2C, se pueden volver funciones para reducir código. En el anexo mostramos el código fuente que permite realizar esta tarea por medio de la función Send_RTC.

Para efectuar una lectura de los registros que contienen los datos de interés, será necesario construir una trama semejante a esta:

- Condición de inicio
- Envío de dirección de esclavo para escritura (0xD0)
- Envío de dirección de registro de segundos (0x00)
- Condición de paro
- Condición de inicio
- Envío de dirección de esclavo para escritura (0xD1)
- Recepción de valor numérico de segundos
- Recepción de valor numérico de minutos
- Recepción del valor numérico de horas
- Recepción del valor numérico de día de la semana
- Recepción del valor numérico de día del mes
- Recepción del valor numérico del mes
- Recepción del valor numérico del año
- Condición de paro

A simple vista no parece tener un sentido lógico tener que escribir para poder leer una dirección de memoria del RTC, sin embargo, es menester recordar que la trama en modo receptor, recibe datos sin poder tener un control de la dirección que nos importa. Para asegurar que se leerán las localidades correctas, primero se desplaza el puntero de registro del DS1307 hacia la 0x00, de segundos.

Una vez implementada esta instrucción, se podrá comenzar a leer los datos, con la certeza de que el puntero incrementará después de haber mandado el valor actual. De igual forma, se puede reducir código si hacemos uso de un puntero. Este puntero se encargará de mandar a un buffer vacío del microcontrolador, cierta cantidad de elementos leídos. La

debida implementación, puede verse en el apéndice. Se hace a través de una función llamada Get_RTC.

Programa para la Maquina de Prepago

La máquina de prepago debe ser capaz de poder detectar el saldo ingresado y depositarlo en la tarjeta, así como poder leer el saldo de la tarjeta. Al hacer esto también se implementa la escritura y lectura de la tarjeta RFID a través del módulo MFRC522.

Algoritmo

1. Configurar pines de entrada/salida.
2. Inicializar SPI, módulo MFRC522, pantalla LCD.
3. Esperar a que el modulo encuentre una tarjeta nueva.
4. Preguntar qué operación se va a realizar.
    a) Revisión de saldo, solo se lee el saldo de la tarjeta.
    b) Deposito, se aumenta el valor de saldo de la tarjeta con la detección del dinero.
5. Esperar que retire la tarjeta al finalizar la operación.
6. Volver al paso 3.

Diagrama de Flujo

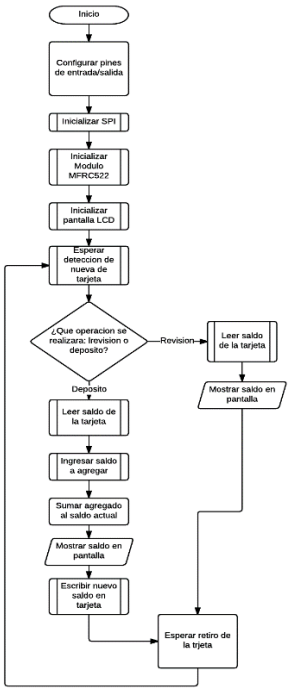


Diagrama de Flujo 1 Maquina de Prepago

Para la detección de las monedas el microcontrolador compara 3 valores para determinar el valor de estas: la salida del sensor Hall, el tamaño a través de distintos sensores para cada moneda y que esta conduzca. Para la detección del billete se compara el tamaño, comparando el tiempo que tarda en pasar por un sensor de herradura.

Algoritmo para la detección del dinero:

1. Obtener valores del sensor Hall.
2. Detectar conducción de la moneda.
3. Ordenar valores obtenidos del sensor Hall.
4. Esperar valor del sensor óptico.
5. Comparar valor obtenido del sensor óptico con el mayor valor obtenido del sensor de Hall.

Para el billete:

1. Esperar detección del billete por parte del primer sensor de herradura (flanco de subida).
2. Activar timer al pasar por el segundo sensor de herradura (flanco de subida).
3. Detener timer al detectar término del paso del sensor de herradura (flanco de bajada).
4. Comparar conteo del timer con valor establecido del billete.

Diagrama de Flujo

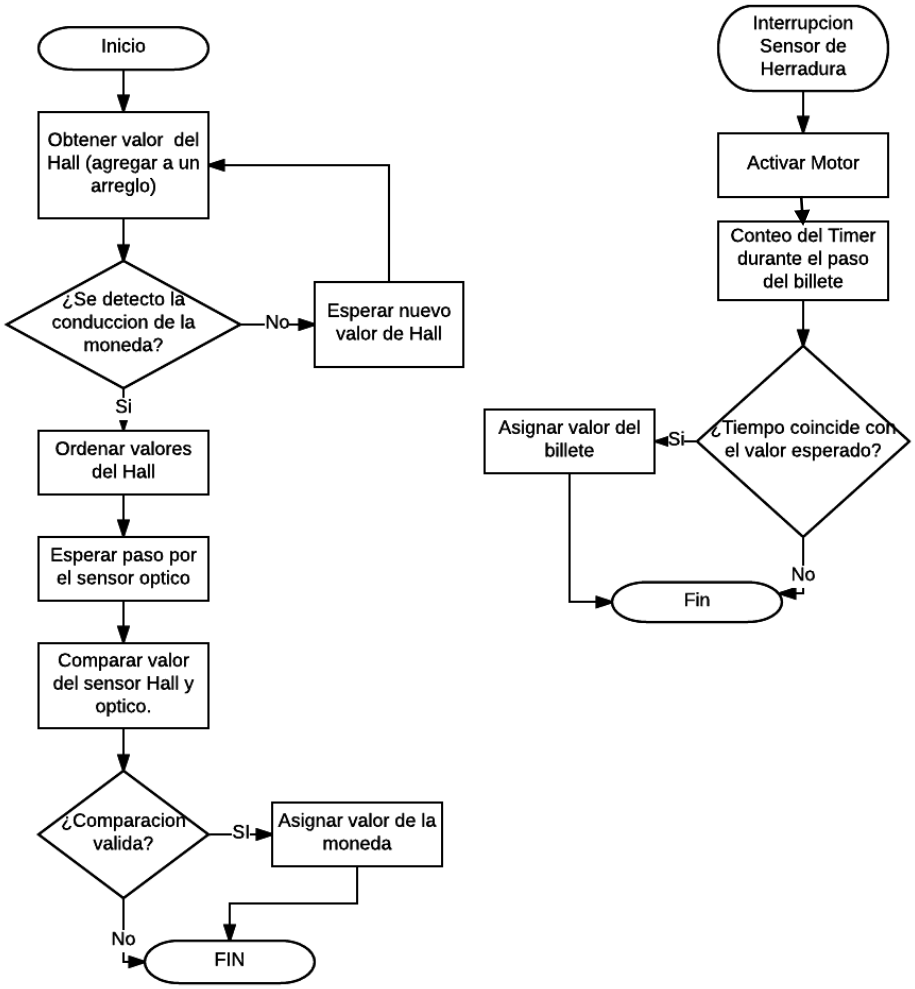


Diagrama de Flujo 2 Diagrama de flujo detección de dinero

Configuración de Timer e Interrupciones

El timer realizará una base de tiempo de 1ms, es decir que la medida del tiempo estará en ms. Para realizar esto se genera una interrupción por desbordamiento que indique cuando haya pasado 1ms. El desbordamiento ocurre cada vez que la cuenta llega a 255, por lo que es necesario la determinación de cuánto tarda en realizar un solo conteo y cuantos son necesarios para realizar la base de tiempo:

$$Tiempo\ para\ un\ conteo = \frac{1}{clk/prescaler}$$

Se determinó el prescaler de 1024, para tener el menor número de conteos:

$$Tiempo\ para\ un\ conteo = \frac{1}{12MHz/1024} = 85.3333\mu s$$

Para determinar el número de cuentas necesarias:

$$Base\ de\ Tiempo = N°\ de\ Conteos * Tiempo\ de\ un\ conteo$$

$$N°\ de\ Conteos = \frac{Base\ de\ Tiempo}{Tiempo\ de\ un\ conteo}$$

$$N°\ de\ Conteos = \frac{1ms}{85.333\mu s} = 11.718$$

Para determinar en qué número debe iniciar el conteo para tener una interrupción por desbordamiento, utilizaremos el número entero más cercano, en este caso 12, y se lo restaremos al número de desbordamiento:

255-12 = 243.

Para la configuración del timer se utilizan los siguientes registros:

TCCR0:

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Ilustración 28 Registro TCR0

Fuente: (Atmel,2010)

Con los bits CS02, CS01 y CS00 se determina el prescaler y a la activación del timer, si se tienen los bits en 0, el timer estará desactivado. Para los prescaler se establecen asi:

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

Ilustración 29 Prescalers de reloj

Fuente:(Atmel,2010)

TIMSK:

Bit	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Ilustración 30 Registro TIMSK

Fuente:(Atmel,2010)

Los bits OCIE0 y TOIE0 activan las interrupciones por valor de comparación y desbordamiento, para nuestro caso se usará la interrupción de desbordamiento:

```
TIMSK |= (1<<TOIE0);
```

TCNT0:

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Ilustración 31 Registro TCNT0

Fuente:(Atmel,2010)

En este registro se tendrá el conteo que tenga el timer, al mismo tiempo en este registro podemos escribirle para determinar el número que tendrá inicialmente la cuenta, en nuestro caso 243.

El Atmega16 cuenta con 2 interrupciones externas, estas serán usadas como para saber el valor a la salida del sensor de herradura, para la detección del billete.

MCUSR:

Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Ilustración 32 Registro MCUSR

Fuente:(Atmel,2010)

Los bits ISC11 e ISC10 indican como será activada la interrupción externa 1, mientras que ISC01 e ISC00 indican de la interrupción externa 0.

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

Ilustración 33 Descripción del ISC

Fuente:(Atmel,2010)

Usaremos para la activación del motor una interrupción de flanco de subida y para activar el timer y desactivarlo, un cambio de nivel.

```
MCUCR = (1<<ISC00)|(1<<ISC11)|(1<<ISC10);
```

Programa para los periféricos de pago en las mesas

Los periféricos ubicados en la mesa, deben tener la capacidad de leer las tarjetas y descontar de ellas el pago, así como generar el código YEP. De igual forma, para el reembolso debe poder agregar saldo a la tarjeta y decodificar el código de reembolso.

Algoritmo

1. Inicializar SPI, módulo MFRC522, I2C, RTC y pantalla LCD.

2. Obtener hora del RTC.

3. Modificar RTC si es necesario.

4. Elegir entre pago o reembolso.

4.1 Para el pago:

a) Elegir el restaurante.

b) Obtener cantidad a pagar.

c) Revisar saldo de la tarjeta.

d) Descontar el pago a la tarjeta.

e) Generar código YEP.

4.2 Para el reembolso:

a) Elegir el restaurante.

b) Obtener código de reembolso.

c) Decodificar código.

d) Agregar saldo a la tarjeta.

5. Volver al paso 4.

Diagrama de flujo

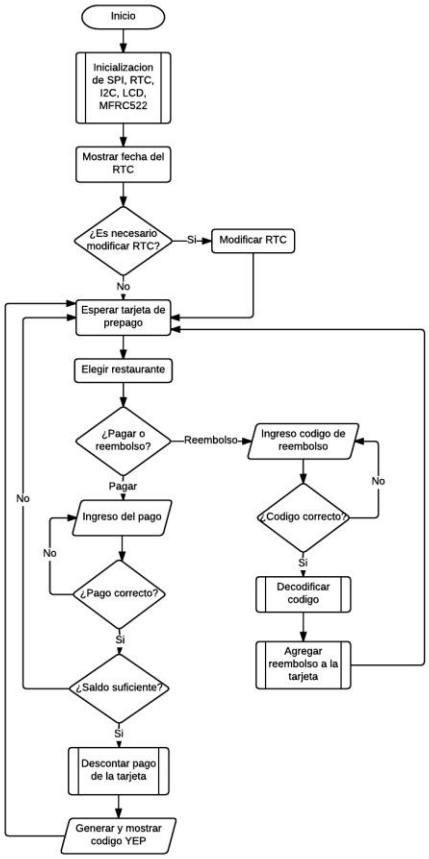


Diagrama de Flujo 3 Operación de periféricos en la mesa

Configuración SPI

Para poder utilizar el módulo de SPI en el Atmega16 y Atmega8 es necesario configurar los siguientes registros:

Bit	7	6	5	4	3	2	1	0	
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	SPCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Ilustración 34 Registro SPCR

Fuente:(Atmel,2010)

El módulo MFRC522 funciona con la fase y polaridad del reloj en configuración 00 y enviando primero el bit más significativo, aparte no será necesario el uso de la interrupción por lo los bits SPIE, DORD, CPOL, CPHA se mantendrán en 0. Lo que necesitamos configurar es la activación del SPI y su funcionamiento como maestro, por lo que se pondrán a 1 los bits de SPE y MSTR. Por último, los bits SPR1 y SPR0 configuran el prescaler, para el uso con el módulo MFRC522 utilizaremos un prescaler de 16:

SPR1	SPR0	SCK Frequency
0	0	$f_{osc}/4$
0	1	$f_{osc}/16$
1	0	$f_{osc}/64$
1	1	$f_{osc}/128$
0	0	$f_{osc}/2$
0	1	$f_{osc}/8$
1	0	$f_{osc}/32$
1	1	$f_{osc}/64$

Ilustración 35 Instancias de reloj SPR

Fuente:(Atmel,2010)

La configuración en programa quedaría de la siguiente manera:

```
SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
```

Para la transmisión de datos en SPI, solo es necesario enviar un dato al registro SPDR, el módulo de SPI enviara un dato por MOSI y le llegara otro por MISO que se almacenara de igual forma en el registro de SPDR. Cada vez que ocurra la transición se activara la bandera SPIF en el registro SPSR, que sirve para indicar que hay un dato recibido en el SPDR.

SPSR:

Bit	7	6	5	4	3	2	1	0	
	SPIF	WCOL	–	–	–	–	–	SPI2X	SPSR
Read/Write	R	R	R	R	R	R	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Ilustración 36 Registro SPSR

Fuente:(Atmel,2010)

Donde se encuentra la bandera SPIF.

SPDR:

Bit	7	6	5	4	3	2	1	0	
	MSB							LSB	SPDR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	X	X	X	X	X	X	X	X	Undefined

Ilustración 37 Registro SPDR

Fuente:(Atmel,2010)

Registro para transmisión/recepción de datos.

Por lo tanto, el manejo de la transmisión por SPI quedaría de la siguiente manera:

```
SPDR = data;
while(!(SPSR & (1<<SPIF)));

return SPDR;
```

Escritura y lectura con MFRC522

El modulo lector MFRC522 necesita comandos específicos para comunicarse con la tarjeta MiFare Classic. Tomando en cuenta también el acomodo de los bloques y sectores de la tarjeta se tomó en cuenta el siguiente algoritmo:

Algoritmo escritura/lectura en tarjeta

1. Buscar tarjeta (activar tarjeta).
2. Entrar en modo de anticolisión (obtención de número de serie de la tarjeta).
3. Seleccionar tarjeta a partir de número de serie.
4. Entrar a modo de autentificación (comprobar clave de la tarjeta).
5. Sobre escribir con los nuevos datos al bloque o leer datos del bloque de la tarjeta seleccionado:

   5.1 Realizar cálculo de CRC (Verificación por redundancia cíclica).

   5.2 Entrar en modo transceptor.

   5.3 Transmisión de datos.
6. Enviar tarjeta a modo Halt (dormido).

Diagrama de Flujo

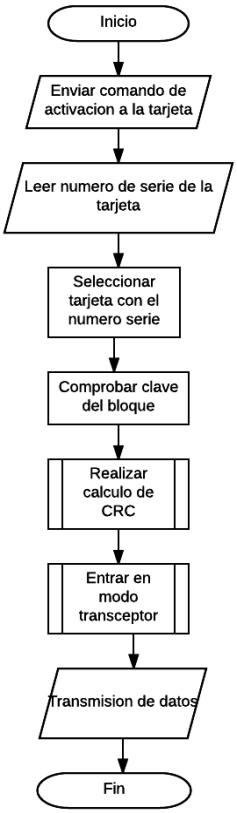


Diagrama de Flujo 4 Lectura/Escritura de la tarjeta

Aplicación para ordenar

Algoritmo de la Aplicación para ordenar:

1. Elegir restaurante.
2. Realizar pedido:
   a) Al elegir un platillo: mostrar en pantalla el precio.
   b) Al presionar el botón de agregar: sumar al total el costo del platillo, y agregar a la lista del pedido el código del platillo seleccionado.

c) Al presionar el botón de eliminar: poner variable del total en 0, y borrar todos los elementos de la lista del pedido.

3. Pagar:

a) Imprimir en pantalla cada uno de los elementos agregados al pedido y el total a pagar.

b) Indicar al usuario los pasos a seguir para pagar y enviar el pedido al restaurante.

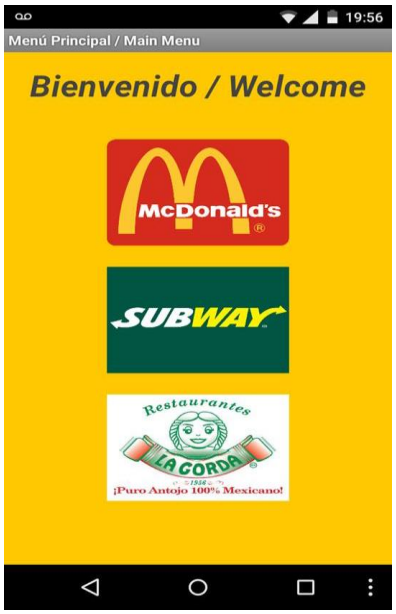
c) Verificar código YEP, y si este es correcto: enviar al restaurante la orden.

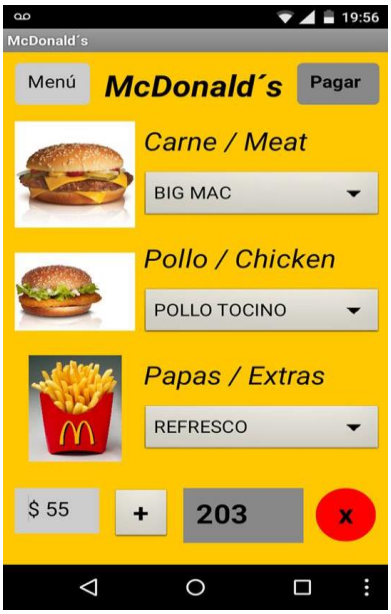
4. Pedidos:

a) Cada segundo: mandar a llamar la función de GetValue y si corresponde a tu pedido, mostrar en pantalla el mensaje de recoger pedido en caja.

Estructura del Código YEP (Ya Está Pagado)

Número de Mesa	Total codificado	Día de la Semana	Hora





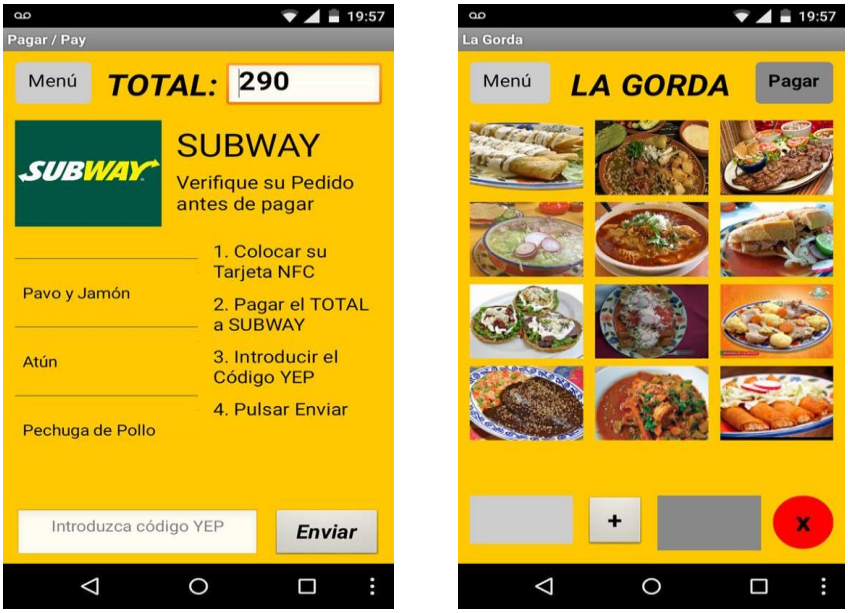


Ilustración 38 Interfaz de la aplicación

Fuente:(Elaboración propia)

Control del motor.

El circuito de activación del motor, es un driver que permite que se active únicamente cuando se detecta la presencia de un cuerpo en la ranura donde se introducen los billetes. Dicho control no permite la devolución del billete una vez dentro de la máquina ya que solo girará en una dirección. Mediante un sensor de herradura, se enciende el motor, empleando un transistor como interruptor.

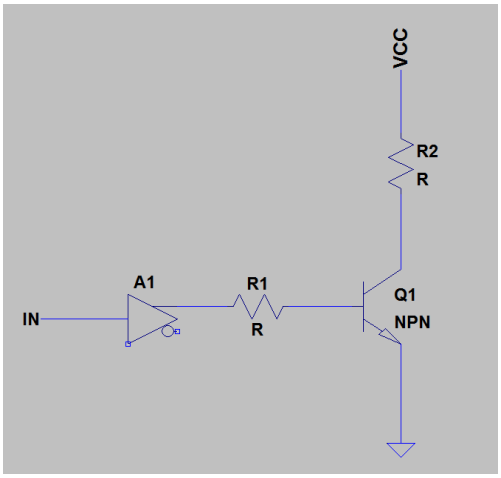


Ilustración 39 Diagrama del controlador del motor

Fuente:(Elaboración propia)

Se añadió un buffer en la base del transistor con el objetivo de reforzar la corriente que entrega el sensor de herradura, dispuesto de igual forma como interruptor. El motor (En este caso R2) al medir su resistencia, resultó ser de 18.3Ω. Aplicando un análisis de mallas encontramos:

$$Ic = \frac{Vcc}{Rm}$$

Si reemplazamos a Vcc=12V:

$$Ic = \frac{12V}{18.3\Omega} = 6505mA$$

Aplicando la regla de diseño 10Ib=Ic encontramos:

$$Ib = \frac{Ic}{10} = \frac{655mA}{10} = 65.5mA$$

Si el voltaje en la entrada del sistema es de 5V, entonces la resistencia de base que produce esos efectos es de:

$$Rb = \frac{Vbb - Vbe}{Ib} = \frac{5V - 0.7V}{65.5mA} = 65.6\Omega$$

Según las especificaciones de la hoja de datos del sensor de herradura, se emplea una resistencia típica de 220Ω para el diodo emisor y una de 3.3kΩ para el colector.

La detección del billete será dada por la longitud del mismo. Para ello se colocó otro sensor de herradura en la parte posterior del sistema que permite el accionamiento del motor. El tiempo en el que la señal permanezca en alto, corresponde de forma proporcional a la longitud del billete, si la velocidad del motor es constante. Este diseño se implementa por software por medio del timer del ATmega16.

Capítulo 3: Desarrollo y Resultados

Calibración de salida diferencial.

Recordando las razones expuestas en el diseño del circuito, se hace mención nuevamente de un dato importante relacionado con el sensor de efecto Hall.

Este sensor entrega una tensión proporcional a la magnitud de la densidad de flujo magnético. Si se aplica un campo magnético constante, como el caso que se presenta en el proyecto, este sensor mantiene una tensión de offset. Dicha tensión resulta ser un problema ya que, al momento de amplificar la señal, esta se saturaría a la salida del sistema por la alta ganancia que tiene.

Por ende, en el diseño se incluye una entrada diferencial en la cual, una de las dos entradas es la del sensor Hall, mientras que la otra es una tensión de referencia que calibrada tentativamente al mismo nivel de offset del Hall en estado estacionario. Con ello el sistema resta el nivel de offset, a la otra entrada, dejando como salida las perturbaciones producidas por una entrada que perturbe la condición de equilibrio.

De esta forma la tensión calibrada a la entrada debería producir una tensión de 0V. Por razones del diseño, esto no debe ocurrir ya que se requieren niveles máximos distintos entre moneda y moneda para poder compararlos, lo suficientemente grandes para llevar a cabo una calibración adecuada. La tensión de salida debe mantenerse a 2.54V negativos en estado de reposo, para poder seguir adelante con la calibración y otros parámetros medidos con las monedas.

Medición de los parámetros de las monedas.

Con el objetivo de llevar a cabo las mediciones de los parámetros magnéticos de las monedas, se registraron los resultados de la tensión entregada por el sensor de efecto hall para cada moneda. Deslizando lentamente una moneda de cada denominación, se analizó el valor de la salida del amplificador diferencial, para encontrar los puntos cúspides de voltaje.

Tabla 5 Voltaje diferencial de salida

Denominación	Voltaje diferencial de salida
$1	6.71V
$2	8.03V
$5	9.71V
$10	NA

Fuente:(Elaboración Propia)

Como se ha hecho referencia, la moneda de $10 no posee propiedades magnéticas, por ende, su presencia no altera el campo establecido en el sensor.

Recomendaciones para la calibración.

La tensión de comparación debe ser ligeramente menor a la de cada moneda, para dejar una holgura en el rango en el que pudiese variar entre moneda y moneda. Por ende, se han enlistado a continuación, una serie de valores a los que es recomendado calibrar las entradas de referencia. Sin embargo, es menester saber que esto puede cambiar, dependiendo de la interferencia en la red eléctrica del ambiente en el que se somete el circuito.

Tabla 6 Calibración

Denominación	Voltaje de referencia a calibrar	Resistencia a establecer
$1	6.65V	1890Ω
$2	7.97V	2417Ω
$5	9.65V	3350Ω

Fuente:(Elaboración Propia)

Los resultados expuestos son producto de una calibración maestra, tomando en cuenta que puede llegar a variar la tensión de alimentación por dos condiciones

-Caída de tensión en la red

El diseño de la fuente de tensión está pensado para regularse mediante un sistema de retroalimentación que poseen los circuitos integrados 7805, 7812 y 7912, de manera interna, sin embargo, no se pueden regular voltajes inferiores ya que no existe un convertidor de DC a DC que se conmute para compensar las tensiones mínimas de la red.

-Variación del voltaje regulado por otro componente eléctrico

Si en algún momento dado, uno de los reguladores se daña y es reemplazado por otra pieza puede llegar a cambiar la tensión que entrega este, haciendo que por una mínima fracción de voltaje, cambien los valores de referencia en incluso los valores típicos medidos para cada moneda.

Activación y control del motor

En este apartado se justifica el uso del buffer para la activación del motor. Si se analiza más a fondo el circuito que compone el control del motor, podemos percatarnos de que la resistencia de pull up puesta en el colector del transistor, que permite el estado de corte y saturación, también produce la corriente en la base del transistor, quedando en serie con Rb en el circuito analizado en el capítulo de diseño.

Sin embargo, la modificación de este resistor no puede establecerse, debido a que no se establecerían de forma adecuada los valores 1 y 0 en digital. Esto representa un conflicto para activarlo ya que se requiere forzosamente una etapa intermedia entre estas dos. El buffer actúa como un acoplador de impedancias, y a su vez un refuerzo de corriente por su salida en colector abierto.

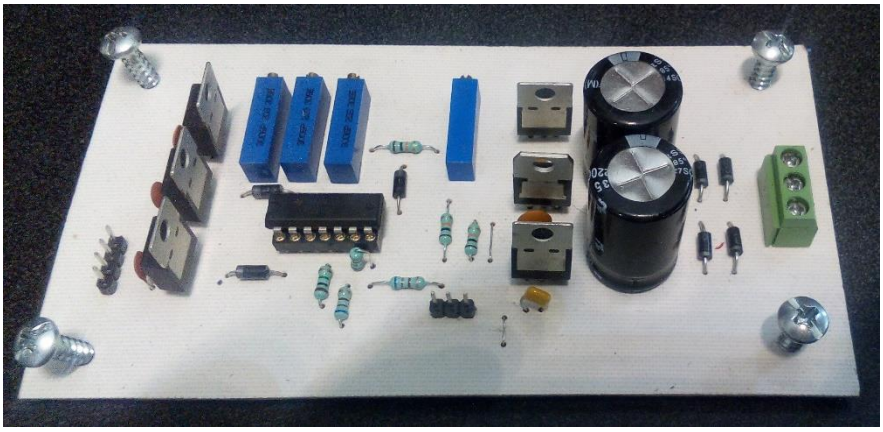
De este modo, se obtiene un aislamiento entre el transistor del sensor de herradura, y el transistor de potencia que activa la siguiente etapa. Un método alternativo para esta

aplicación, es emplear un opto acoplador, con este, podemos modular una señal de corriente alta, con una corriente relativamente pequeña con un aislamiento óptico.

Desarrollo de PCB para la máquina de prepago.

Diseñamos el siguiente PCB, el cual ofrece en su esquina superior izquierda, dos pines de entrada, en donde se conectará el voltaje de corriente directa que entrega el UGN3503 a su salida; dicha tensión será amplificada, y comparada con distintos niveles de referencia, con ayuda de los amplificadores operacionales del circuito integrado TL084.

En la esquina superior derecha se encuentran los pines para la alimentación (de arriba hacia abajo, -Vcc, Vcc y Gnd respectivamente), y por último las 3 salidas que conectaremos al microcontrolador se encuentran en la esquina inferior derecha, después de pasar por un diodo rectificador, por un capacitor (filtro), y el regulador 7805, asegurando con este último los niveles lógicos para la siguiente etapa de procesamiento.



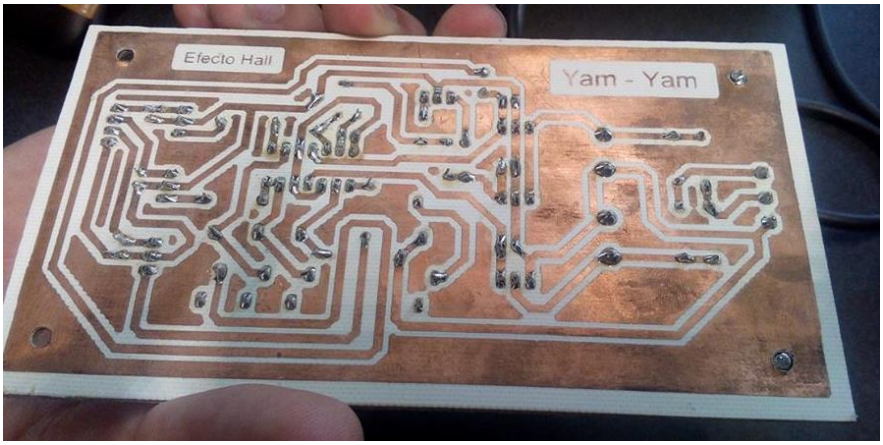


Ilustración 40 Acondicionador de señal Hall

Fuente:(Elaboración Propia)

Diseño del PCB en Eagle:

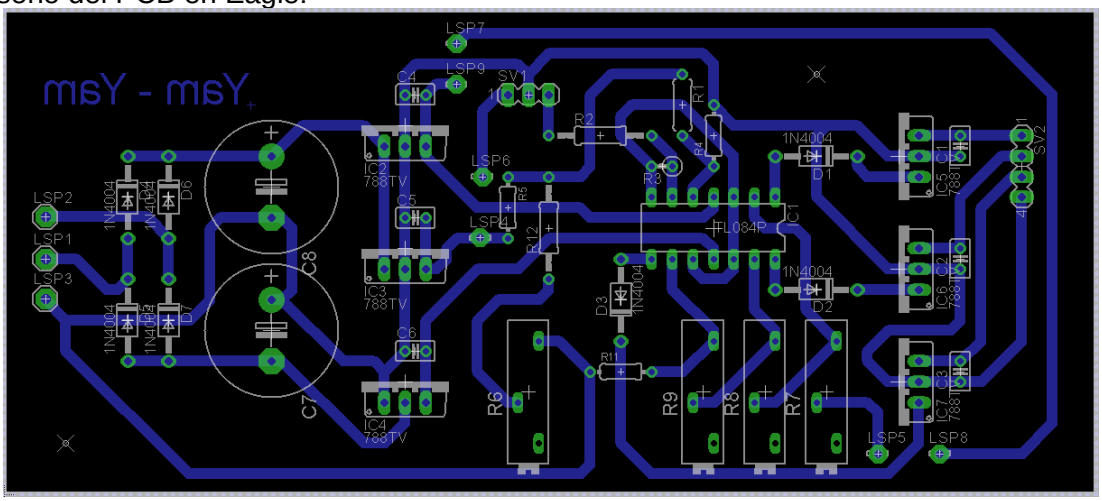


Ilustración 41 Diseño PCB acondicionador de señal Hall

Fuente:(Elaboración Propia)

En el PCB de la etapa de procesamiento se encuentra el Atmega16, donde recibe las señales lógicas de hall, las salidas a la pantalla LCD, las entradas de los botones para la navegación del menú, las conexiones del módulo MFRC522 y las salidas de los fototransistores. También están presentes el cristal de 12 MHz, las resistencias de pull-up en los botones, y un regulador de 3.3 v para la alimentación del módulo MFRC522.

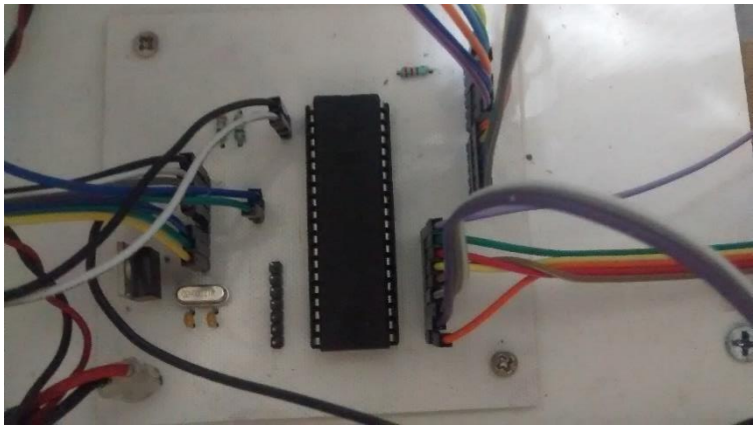


Ilustración 42 Microcontrolador para prepago

Fuente:(Elaboración Propia)

Diseño del PCB en Eagle:

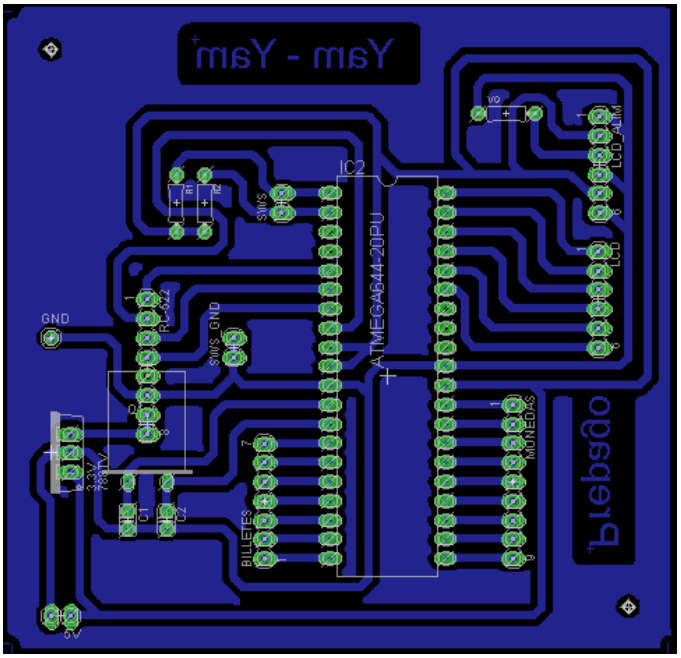
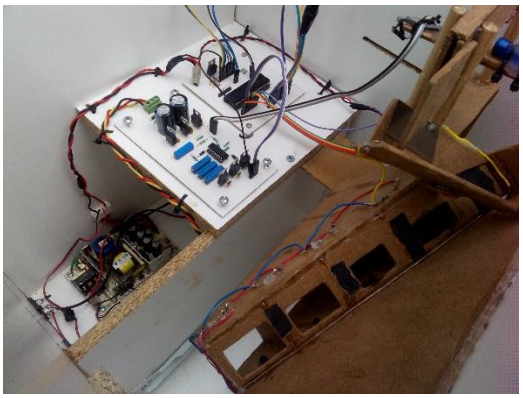


Ilustración 43 Diseño PCB microcontrolador prepago

Fuente:(Elaboración Propia)

La conexión final de los PCB al chasis se realizó de la siguiente manera:



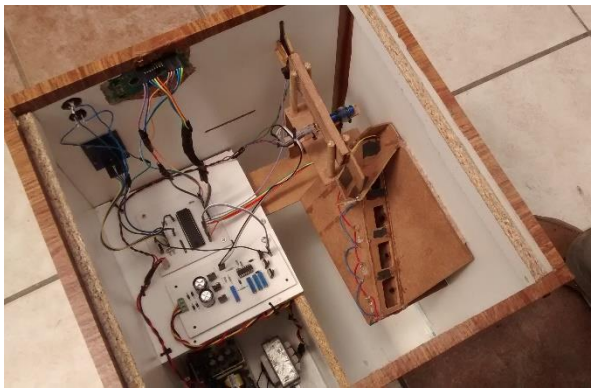


Ilustración 44 Interior máquina de prepago

Fuente:(Elaboración Propia)

Finalmente, la máquina de prepago terminó montada de la siguiente manera:

Ilustración 45 Exterior máquina de prepago

Fuente:(Elaboración Propia)

Pruebas de la máquina de prepago

La interfaz en la pantalla de LCD, para la máquina de prepago mostro todos los mensajes necesarios, así como también se logró el funcionamiento de los botones para la navegación. También fue exitosa la detección de la tarjeta por parte de la máquina. Esta es la pantalla que muestra al encender la maquina esperando a la tarjeta de prepago:

Ilustración 46 Interfaz prepago

Fuente:(Elaboración Propia)

Una vez que se pasa la tarjeta la pantalla muestra el siguiente mensaje para la selección de la acción que debe realizar: revisar saldo o depositar a la tarjeta. Para seleccionar cualquiera de los dos se utilizan los botones, el botón superior es para la selección del caso A (revisar saldo) y el inferior del caso B (depositar).

Ilustración 47 Menú de prepago

Fuente:(Elaboración Propia)

La revisión del saldo se realizó con éxito, se utilizó una tarjeta que contenía 207 pesos de saldo, el resultado mostrado por la pantalla fue el siguiente:

Ilustración 48 Lectura de saldo

Fuente:(Elaboración Propia)

El mensaje de fin mostrado en la parte inferior derecha de la pantalla, es para poder volver al menú de selección de acción de la máquina.

Para el depósito de saldo, de igual manera se muestra el saldo en la pantalla superior, y lo que se está depositando en la parte inferior, de igual manera un mensaje de fin en la parte inferior derecha de la pantalla para dejar de realizar los depósitos. Las pruebas se realizaron con monedas de $1, $2, $5, $10, que son las monedas que debe detectar la máquina, también se realizaron pruebas con la moneda de plástico, una moneda de 50 centavos y rondanas de metal.

Ilustración 49 Monedas de prueba

Fuente:(Elaboración Propia)

Resultado de las pruebas:

Tabla 7 Prueba de la detección de monedas

Objeto de Prueba	Resultado
Moneda 50 centavos	No efecto
Moneda $1	Suma de 1
Moneda $2	Suma de 2
Moneda $5	Suma de 5
Moneda $10	Suma de 10
Moneda de plástico	No efecto
Rondana (2.8 cm de diámetro)	No efecto
Rondana (2.1 cm de diámetro)	No efecto

Fuente:(Elaboración Propia)

Una vez ingresado el saldo, se muestra en la parte inferior de la maquina la cantidad que se ha ingresado. Al darle a la indicación de fin, la maquina debe realizar la suma de la cantidad depositada con el saldo actual de la tarjeta y escribirlo a esta.

Ilustración 50 Depósito de saldo

Fuente:(Elaboración Propia)

Como se puede observar la suma se realizó de manera correcta. Se tenían 207 pesos de saldo en la tarjeta de le sumo 5 pesos a esta, por lo tanto, el resultado final debe ser la suma de estos, que es 212, como se muestra en las pantallas.

Desarrollo de PCB para periférico de pago

En el PCB de los periféricos que se encuentran en las mesas está diseñado para el uso del Atmega8, donde se tienen pines para las salidas a la pantalla LCD, el teclado matricial, y las conexiones del módulo MFRC522. También están presentes el cristal de 12 MHz y un regulador de 3.3 v para la alimentación del módulo MFRC522.

Diseño de PCB en Eagle:

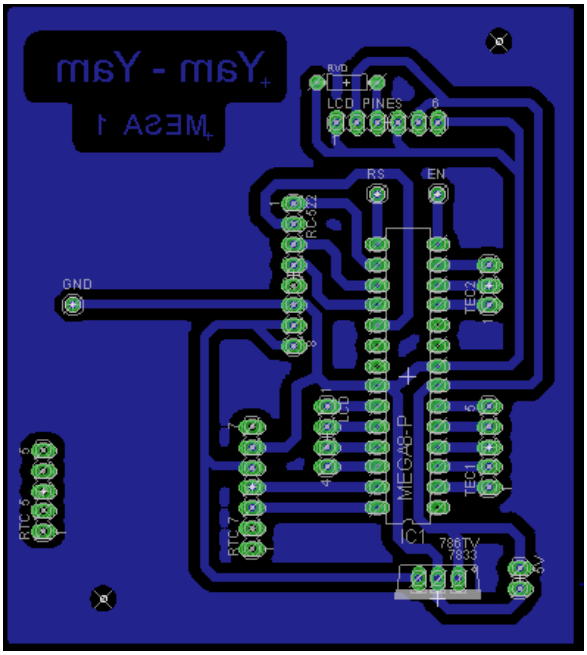


Ilustración 51 Diseño PCB periférico de mesa

Fuente:(Elaboración Propia)

Los chasises de los periféricos de la mesa se les montó por la parte superior el teclado, también se mostrará la pantalla y el área donde se debe montar la tarjeta de NFC, la cual estará colocada por dentro.

Ilustración 52 Exterior de periféricos

Fuente:(Elaboración Propia)

Pruebas de periféricos en la mesa

El primer mensaje dado por la mesa es la fecha y hora actual dada por el RTC. Después aparece el mensaje para la configuración de la hora, para su configuración se debe ir metiendo los datos de la siguiente manera: segundos, minutos, hora, día de la semana, día del mes, mes y año.

Ilustración 53 Menú de configuración de mesa

Fuente:(Elaboración Propia)

Entonces se cargará la siguiente secuencia de datos:

00, 41, 10, 07, 18, 06, 16.

La fecha obtenida debe ser 18 de junio de 2016, 10:41 sábado, con una variación de aproximadamente un minuto por lo que se pueda tardar en teclear la secuencia.

Fecha obtenida:

Ilustración 54 Lectura de fecha

Fuente:(Elaboración Propia)

Una vez configurada la fecha se tiene una pantalla de bienvenido, y después las opciones de pagar o realizar un reembolso. En la opción de pagar es ingresado la cantidad que se va a descontar a la tarjeta, y de igual manera pide una confirmación de esta. Una vez obtenida la confirmación generara el código YEP.

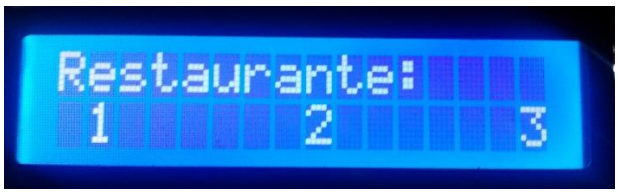


Ilustración 55 Interfaz de periférico

Fuente:(Elaboración Propia)

Para demostrar el funcionamiento de la generación del código se realiza un pago de $50 en los 3 restaurantes a las 10:00 pm en sábado, los códigos obtenidos deben ser:

Tabla 8 Generación de código YEP

Restaurante	**Código**
1	29D882764
2	2AAAAA764
3	28E364764

Fuente:(Elaboración Propia)

Ilustración 56 Confirmación del pago realizado

Fuente:(Elaboración Propia)

Resultado:

Restaurante 1.

Restaurante 2.

Restaurante 3

Ilustración 57 Códigos YEP de los restaurantes por el mismo monto

Fuente:(Elaboración Propia)

En la opción de reembolso se debe ingresar el código de reembolso para que pueda ser decodificado y añada la cantidad a reembolsar a la tarjeta. Al igual que el pago pide confirmación del código antes de realizar la transacción. Para demostrar su funcionamiento se ingresarán tres códigos para los tres restaurantes distintos, con tres cantidades diferentes, teniendo en cuenta que el saldo inicial de la tarjeta es 672.

Tabla 9 Uso del código de reembolso

Restaurante	**Código**	**Saldo a Agregar**	**Saldo de la tarjeta después del agregado**
1	456354	12	684
2	1252525	25	709
3	2161544	50	759

Fuente:(Elaboración Propia)

Resultados:

Restaurante 1.

Restaurante 2.

Restaurante 3.

Ilustración 58 Códigos SUP de los restaurantes

Fuente:(Elaboración Propia)

Aplicación

La aplicación del celular es capaz de mandar la confirmación del pago a la aplicación para la caja de cada restaurante. Esto se hace mediante la confirmación del código YEP es válido, es decir que el precio codificado sea posible con la elección del menú del restaurante, además del tiempo en que fue emitida la orden y el código sea válido para el restaurante.

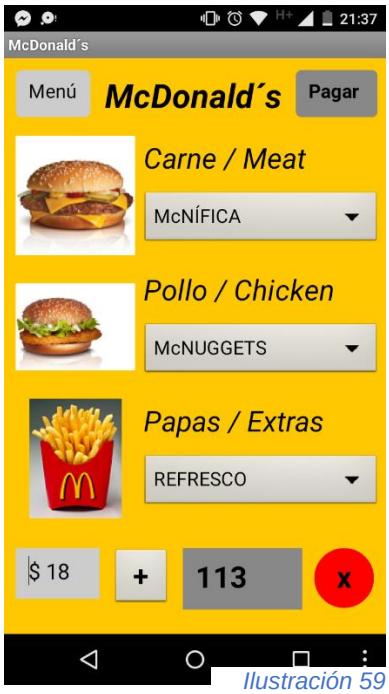


Ilustración 59 Código YEP en aplicación

Fuente:(Elaboración

Una vez ingresado el código y confirmado que es válido, el celular envía un mensaje de la orden a la aplicación en caja. En caja este envía de vuelta un mensaje cuando la orden se encuentra lista, al celular para que el cliente solo vaya por ella, después de eso en caja se puede borrar la orden de la lista.

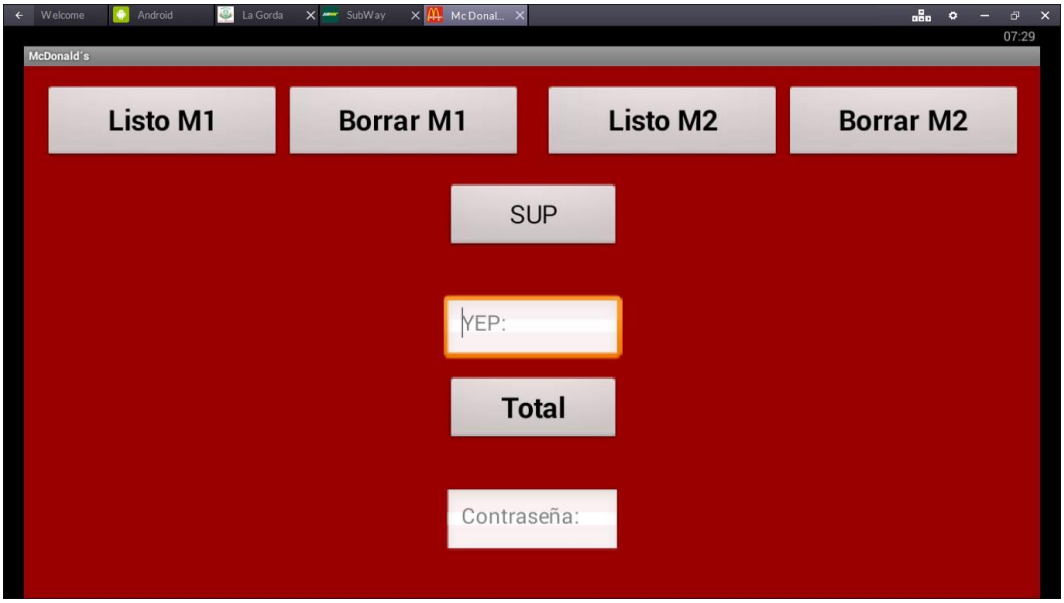


Ilustración 60 Plataforma del restaurante

Fuente:(Elaboración Propia)

Conclusiones

Conclusiones

El objetivo general que era la construcción de un sistema integral que agilizara la manera en la que se ordena comida, fue llevado a cabo con éxito, pues se crearon todos los elementos necesarios para dicho sistema y funcionan tanto de manera individual como colectiva, tiene elementos que agilizan la petición de la comanda (aplicación telefónica y el software del restaurante) y elementos que mejoran la manera en que se realiza el pago (Máquina de prepago y las maquinas generadoras de código YEP).

La manera en la que funciona la tarjeta de prepago es satisfactoria pues en la máquina de recargas/ revisión de saldo se permite ver lo que la cantidad de “saldo” a favor que tiene la tarjeta, la cantidad de dinero que se le abonará, y una vez que se bonifica la cantidad especificada se sobre escribe la cantidad en la tarjeta a la par que se muestra la misma en pantalla, y cuando se desea hacer un pago se descuenta la cantidad seleccionada antes de generar el código YEP para a su vez mostrarlo en pantalla, y con las mismas instrucciones que en la máquina de prepago se pueden hacer bonificaciones desde los elementos periféricos que van en las mesas.

Buscamos reforzar la seguridad en los cobros con el código YEP cuyo algoritmo de generación ya fue explicado en el capítulo anterior, para que no se puedan realizar pedidos sin que se haya realizado el pago, pues parte de la información que contiene es la cantidad de saldo descontada de la tarjeta, la fecha y la hora, por lo tanto, es muy difícil que ingresando valores al azar se logre enviar el pedido, pero claro que como posible mejora para el futuro se pueda utilizar un mejor algoritmo de encriptación. Cosa que en realidad no es tan viable, ya que en caso de que se realice una mejora, se buscaría por confort para el usuario se genere un código QR en lugar de una serie de dígitos y letras.

Las aplicaciones tanto de usuario como administrativas son útiles para la demostración, pero, sobre todo la aplicación para restaurantes (el software administrativo) aún le faltan mejorar, pues, solo sirve para dos mesas y quizá si se buscara agregar más mesas, se volvería engorroso de utilizar, por lo tanto, debería de cambiar su interfaz.

Una forma de resolver este problema se podría utilizar herramientas más especializadas y menos básicas como Photoshop para el diseño de los botones o java para el desarrollo de la aplicación.

Recomendaciones y Futuras Mejoras

Como mejoras y recomendaciones además de las ya mencionadas se piensa que, a futuro, si se buscara llevar este sistema a un nivel comercial es necesario mejorar la máquina de prepago, pues por el momento tanto el reconocimiento de monedas como el de billetes se realiza de manera muy arcaica y tomando en cuenta que existen empresas que se dedican enteramente a realizar este tipo de aparatos, creemos que es mejor opción y se debe utilizar, otro “fallo” que tiene la máquina de prepago es que no encripta la cantidad de dinero que se está abonando en la tarjeta por lo cual puede haber quien pueda leer las tarjetas, e incluso ponerles “dinero”.

Este procedimiento muy probablemente requiera de una mejor codificación y, por lo tanto, un mejor microcontrolador. Y dependiendo de qué equipo se utilice para estas tareas puede que requiera utilizar alguna codificación o algún protocolo de comunicación especifico.

En la mesa se requiere de un dispositivo más pequeño, y una pantalla diferente, que fuera capaz de reproducir en pantalla el código QR anteriormente mencionado, evitando ingresar el YEP al celular. Y en ese caso se necesitará, probablemente, un mejor microcontrolador o incluso una tarjeta de desarrollo que sea capaz de generar dicho código.

Referencias

Atmel. (2010). Datasheet ATmega16 (L). Retrieved from http://www.atmel.com/Images/doc2466.pdf

Chavarría, D. (2011). *Tecnología de Comunicación de Campo Cercano (NFC) y sus Aplicaciones*.

DALLAS SEMICONDUCTORS. (n.d.). *DS1307*.

Espinosa, M. C. F. S. (2016). Rganización interna de los microcontroladores.

López, E. (n.d.). Protocolo SPI(Serial Peripherical Interface). *Ingenieria En Microcontroladores*, 10.

Martín, R. (n.d.). *Sensores de efecto Hall*. San Luis.

N X P Semiconductors. (2011). MF1S50yyX MIFARE Classic 1K - Datasheet, (May), 1–39.

N X P Semiconductors. (2014). MFRC522 Standard 3V MIFARE reader solution Datasheet, (September), 95–150.

Narváez, C. (n.d.). Reloj de Tiempo Real DS1307.

Ramírez, K. (2011). App Inventor. Universidad de Costa Rica, 32. Retrieved from http://www.kramirez.net/Robotica/Material/Presentaciones/AppInventor.pdf

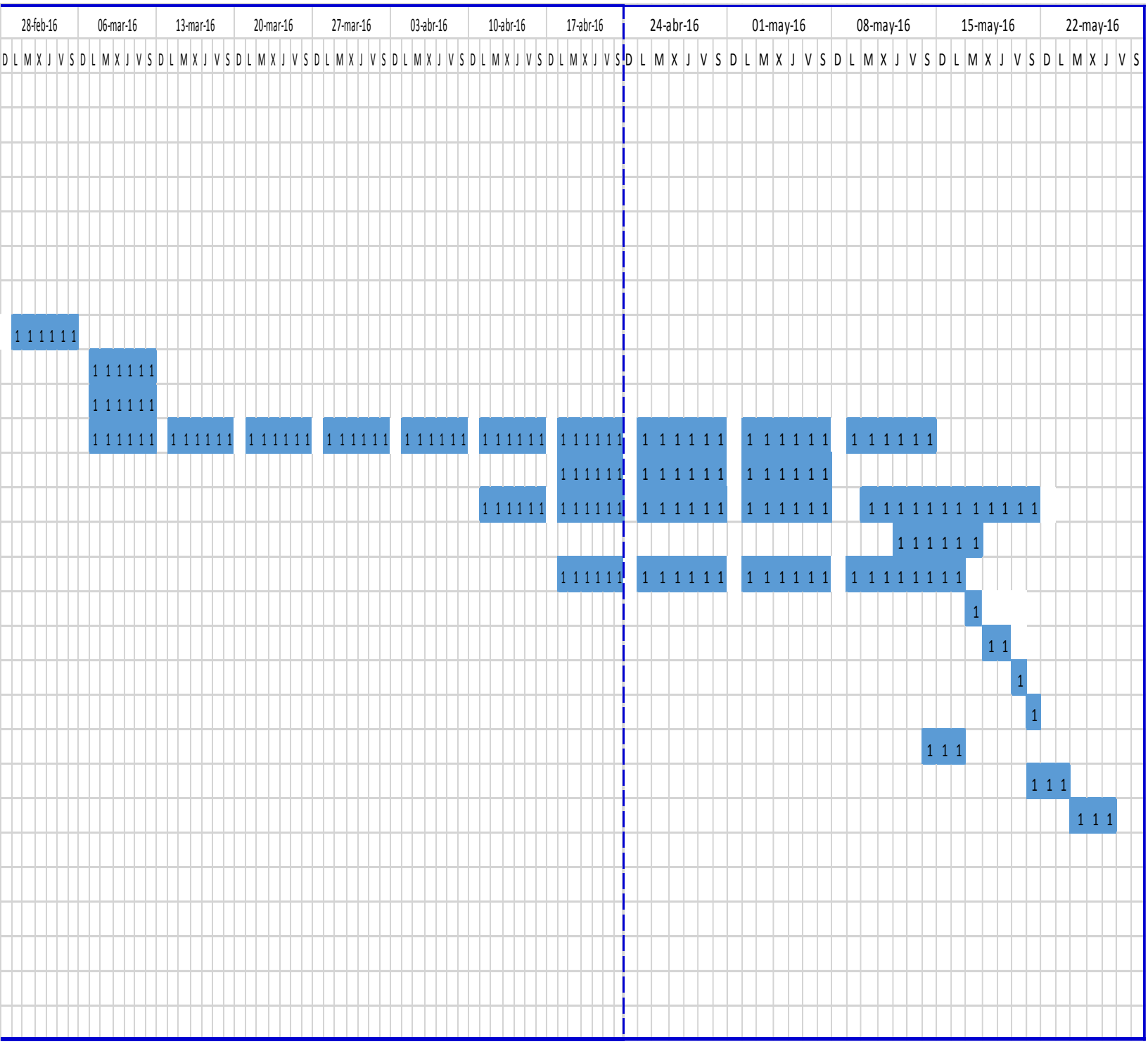
RFID: Tecnología, Aplicaciones, y Perspectivas. (2010). *Libera*, 21.

Working with Databases Storing Persistent Data in TinyDB. (2014). App Inventor. Retrieved from http://www.appinventor.org/assets/pdf/ch22Databases.pdf

Anexo A

Planeación

Nombre de tarea	Días	22-nov-15	29-nov-15	06-dic-15	13-dic-15	20	27-dic-15	03-ene-16	10-ene-16	17-ene-16	24-ene-16	31-ene-16	07-feb-16	14-feb-16
Diseño del circuito para máquina de prepago	9	1 1 1 1 1 1	1 1 1											
Conseguir material para máquina de prepago	1			1										
Conseguir materiales NFC	9			1 1	1 1 1 1 1 1	1								
Construcción y pruebas del sistema de prepago	8			1 1	1 1 1 1 1 1						1 1 1 1 1 1	1 1 1 1 1		
Simulacion del sistema NFC	3		1 1 1											
Diseño del chasís del sistema de prepago	5								1 1 1 1 1 1	1				
Conseguir material para chasís de la máquina de prepago	5									1 1 1 1 1				
Experimentación en App Inventor	18													1 1 1 1 1 1 1
Definicion de los menus de los restaurantes	6													
Creacion de tablas de datos con los menus	6													
Realizacion de la aplicación para la adquisicion de un producto	60													
Prueba del sistema NFC	18													
Desarrollo de la interfáz en las computadoras	36													
Diseño de PCBs del proyecto	6													
Programación de la máquina de prepago	26													
Pruebas de todo el sistema comunicado	1													
Rediseño y modificaciones del sistema de prepago	2													
Elaboración de PCBs	1													
Colocación y soldado de componentes	1													
Elaboración de chasís del sistema de prepago	3													
Integración del sistema	3													
Pruebas de funcionamiento con el sistema finalizado	3													
total	243													
hecho hasta ahora	202													


28-feb-16
06-mar-16
13-mar-16
20-mar-16
27-mar-16
03-abr-16
10-abr-16
17-abr-16
24-abr-16
01-may-16
08-may-16
15-may-16
22-may-16
D L M X J V S

Anexo B

Presupuesto

Componente Electrónico	Cantidad	Precio	Subtotal
Atmega 8	2	110	220
Atmega 16	1	160	160
RTC	2	70	140
LCD	3	80	240
Teclado	2	35	70
TC	3	50	150
Ventilador	1	25	25
Fuente Conmutada	3	70	210
Baquelita	5	20	100
TL084	1	10	10
Trimmpot	4	18	72
Metros de Cable	4	3	12
Tira de Pines	2	5	10
Bases para Integrados	4	10	40
Motor DC	1	20	20
TIP 41C	1	9	9
Buffer	1	15	15
Fototransistor	4	7	28
LED Infrarojo	4	2	8
Sensor de Herradura	2	12	24
Módulo RC-522	3	60	180
Cristal	1	10	10
Botones	2	8	16
Alambre (H-H)	100	2	200
Regulador de Voltaje	6	11	66
Transformador	1	30	30
sensor de Efecto Hall	1	60	60
Material Chasis (MDF, Melamia, Tornillos, Pegamento, etc)		600	600
Total			2725

Anexo C

Código fuente máquina de prepago

```
#include "ComidaProyecto.h"


/* Activar motor */
ISR(INT0_vect){
       PORTD |= (1<<PIND4);
}

ISR(INT1_vect){
    PORTD &= ~(1<<PIND4);
       u16Coin += 100;
       LCD_JUMP();
       snprintf((char*)&u8aLCD_Mensaje[0],MAX_LEN,"    %d              ",u16Coin);
       LCD_MESSG(u8aLCD_Mensaje,18);

}

int main(void){

/*Configuracion de puertos*/
    MCUCSR|= (1<<JTD);
    MCUCSR|= (1<<JTD);
    DDRA = 0xFF;
       DDRB &= ~(1<<PINB0);
       DDRB &= ~(1<<PINB1);
       DDRC = 0x00;
       DDRD &= ~(1<<PIND7);
       DDRD |= (1<<PIND4);
       PORTD |= (1<<PIND7);

       MCUCR = (1<<ISC00)|(1<<ISC01)|(1<<ISC11);

       EN_LOW;

       ptrfuncLCD=LCD_Idle;

       spi_init();
       MFRC522_Init();
       LCD_INIT();


       for (;;)
       {

              /*Esperar tarjeta*/
          Detect_Card();

              LCD_Clean();

              /*Seleccionar accion a realizar*/
              snprintf((char*)&u8aLCD_Mensaje[0],17,"Revisar Saldo  A");
              LCD_MESSG(u8aLCD_Mensaje,32);

              LCD_JUMP();
```

```
		snprintf((char*)&u8aLCD_Mensaje[0],17,"Depositar      B");
		LCD_MESSG(u8aLCD_Mensaje,32);

		u8Flag_Operacion = 0;


		for(;u8Flag_Operacion == 0;){

		u8Opcion = BUTTONS;
		for (; u8Opcion > 2 && u8Opcion <= 0;){
		u8Opcion = BUTTONS;
		}


	/* Leer Saldo de la tarjeta*/
		if(u8Opcion == 1){
			u8Flag_Operacion = 1;
			LCD_Clean();

			Detect_Card();  //Detectar y leer tarjeta
			Read_Card(4);

			LCD_MESSG(read,30); //Mostrar Saldo en pantalla


			u8aInstruccion[0] = 0xCC;
			LCD_INST(u8aInstruccion);

			snprintf((char*)&u8aLCD_Mensaje[0],16," Fin");
			LCD_MESSG(u8aLCD_Mensaje,32);

			while(BUTTONS == 3);

			//Esperar Retirado de Tarjeta
			LCD_Clean();

			snprintf((char*)&u8aLCD_Mensaje[0],17,"Retire   Tarjeta");
			LCD_MESSG(u8aLCD_Mensaje,32);

		/*Depositar Saldo a Tarjeta*/
		}else if(u8Opcion == 2){
			u8Flag_Operacion = 1;
			u16Coin = 0;
			u16Saldo = 0;
			LCD_Clean();

			Detect_Card();   //Detectar y leer tarjeta
			Read_Card(4);
			LCD_MESSG(read,30);
			sscanf((char*)read," %d",&u16Saldo);  //Obtener Saldo de tarjeta
			u8aInstruccion[0] = 0xC4;
			LCD_INST(u8aInstruccion);
			snprintf((char*)&u8aLCD_Mensaje[0],16,"0         Fin");  //Valor
inicial de deposito
			LCD_MESSG(u8aLCD_Mensaje,32);

			sei();  //Activar interrupciones

			/* Deteccion de Monedas*/
```

```
                    while(Coin_Cycle()){

                    /*Leer Hall hasta detectar conduccion de la moneda*/
                    for(u8index = 0; METAL == 0x80; u8index++){
                          u8aCoin_Hall[u8index]= HALL;
                          while((u8aCoin_Hall[u8index] == HALL) && (METAL ==
0x80));
                    }

                    /*ordenar valores del Hall*/
                    for(u8index = 0; u8index < 8; u8index++){
                          for(u8jindex = 0; u8jindex < 8; u8jindex++){
                                if (u8aCoin_Hall[u8index] <=
u8aCoin_Hall[u8jindex]){
                                      u8Temp= u8aCoin_Hall[u8index];
                                      u8aCoin_Hall[u8index] =
u8aCoin_Hall[u8jindex];
                                      u8aCoin_Hall[u8jindex] = u8Temp;
                                }
                          }
                    }                                                       //Fin
Hall


                    while(OPTICAL == 0);  //Esperar deteccion del sensor optico
                    u8Coin_Optical = (OPTICAL >> 3);
                    switch (u8aCoin_Hall[7]){              //Comparar valor del Hall
                          case 0x00:
                          if(u8Coin_Optical == 0x01){  //Comparar para moneda de 10
                                u16Coin += 10;
                          }
                          break;
                          case 0x01:                   //Comparar para moneda de 1
                          if(u8Coin_Optical == 0x08){
                                u16Coin += 1;
                          }
                          break;
                          case 0x03:                   //Comparar para moneda de 2
                          if(u8Coin_Optical == 0x04){
                                u16Coin += 2;
                          }
                          break;
                          case 0x07:                   //Comparar para moneda de 5
                          if(u8Coin_Optical == 0x02){
                                u16Coin += 5;
                          }
                          break;
                          default:
                          break;
                    }
                    for(u8index = 0; u8index<8;u8index++){ //Limpiar valores del
arreglo Hall
                          u8aCoin_Hall[u8index] = 0;
                    }

                    LCD_JUMP();
                    snprintf((char*)&u8aLCD_Mensaje[0],MAX_LEN,"    %d
",u16Coin);     //Mostrar monto depositado
                    LCD_MESSG(u8aLCD_Mensaje,18);
```

```
                    }

                    cli();   //Desactivar interrupciones

                    LCD_Clean();

                    /*Mostrar Saldo*/
                    snprintf((char*)&u8aLCD_Mensaje[0],16,"Saldo:");
                    LCD_MESSG(u8aLCD_Mensaje,12);

                    u8aInstruccion[0] = 0x86;
                    LCD_INST(u8aInstruccion);

                    Read_Card(4);
                    sscanf((char*)read," %d",&u16Saldo);
                    snprintf((char*)&write[0],16,"    %d             ",u16Coin +
u16Saldo);  //Mostrar saldo mas saldo depositado
                    LCD_MESSG(write,32);

                    while(BUTTONS == 3);

                    Detect_Card();
                    Write_Card();  //Escribir saldo a la tarjeta

                  u8aInstruccion[0] = 0x01;
                  LCD_INST(u8aInstruccion);

                  snprintf((char*)&u8aLCD_Mensaje[0],17,"Retire   Tarjeta");
                  LCD_MESSG(u8aLCD_Mensaje,32);

              }
              }

       }
              }

/* Condicion para el ciclo de deposito de saldo */
unsigned char Coin_Cycle(void){
       while(BUTTONS == 3 && !HERR);  //Esperar sensor de Herradura o Boton de
Salida
               if(HERR){    //Sensor de herradura: seguir en ciclo
              return 1;
       }
       return 0;        //Boton: Salir del ciclo
}
```

Anexo D

Código fuente periférico de pago

```
#include "Recepcion_Mesas.h"

int main(void)
{
       unsigned char i;

/* Inicializacion de modulos y perifericos*/
 spi_init();
 MFRC522_Init();
 Configuration_I2C();
 LCD_INIT();

 /* Configuracion de hora y fecha por RTC*/
 Show_RTC();
 LCD_Clean();
 snprintf((char*)&u8aLCD_Mensaje[0],17,"Modificar Hora   ");
 LCD_MESSG(u8aLCD_Mensaje,32);
 LCD_JUMP();
 snprintf((char*)&u8aLCD_Mensaje[0],17,"SI            NO");
 LCD_MESSG(u8aLCD_Mensaje,32);

 u8Opcion = Keyboard();
 while(u8Opcion != 0x0F && u8Opcion != 0x0E){
        u8Opcion = Keyboard();
 }

 if(u8Opcion == 0x0F){
 CorreccionHora: RTC_Configuration();
 }

 /* Confirmacion de Fecha y Hora */
  LCD_Clean();
  snprintf((char*)&u8aLCD_Mensaje[0],17,"¿Fecha correcta?");
  LCD_MESSG(u8aLCD_Mensaje,32);
  LCD_JUMP();
  snprintf((char*)&u8aLCD_Mensaje[0],17,"SI            NO");
  LCD_MESSG(u8aLCD_Mensaje,32);

  u8Opcion = Keyboard();
  while(u8Opcion != 0x0F && u8Opcion != 0x0E){
         u8Opcion = Keyboard();
  }

  if(u8Opcion == 0x0E){
         goto CorreccionHora;
  }

 /* Mensaje de Bienvenido */
 for(;;){
  LCD_Clean();
  snprintf((char*)&u8aLCD_Mensaje[0],17,"Bienvenido      ");
  LCD_MESSG(u8aLCD_Mensaje,32);
  LCD_OK();
```

```
u8Opcion = Keyboard();
while(u8Opcion != 0x0E){
       u8Opcion = Keyboard();
}

/* Seleccionar Accion */
 LCD_Clean();
 snprintf((char*)&u8aLCD_Mensaje[0],17,"Pagar          A");
 LCD_MESSG(u8aLCD_Mensaje,32),
 LCD_JUMP();
 snprintf((char*)&u8aLCD_Mensaje[0],17,"Reembolso      B");
 LCD_MESSG(u8aLCD_Mensaje,32);

u8Opcion = Keyboard();
while(u8Opcion != 0x0A && u8Opcion != 0x0B){
       u8Opcion = Keyboard();
}
 LCD_Clean();
 Detect_Card(); //Detectar Tarjeta
 /*Seleccionar Restaurante*/
 LCD_Clean();
 snprintf((char*)&u8aLCD_Mensaje[0],17,"Restaurante:    ");
 LCD_MESSG(u8aLCD_Mensaje,32),
 LCD_JUMP();
 snprintf((char*)&u8aLCD_Mensaje[0],17," 1      2      3");
 LCD_MESSG(u8aLCD_Mensaje,32);

 u8Restaurant = Keyboard();
 while(u8Restaurant < 0x01 || u8Restaurant > 0x03){
        u8Restaurant = Keyboard();
 }

 /* Pagar */
if(u8Opcion == 0x0A){
Regresar:     LCD_Clean();

      snprintf((char*)&u8aLCD_Mensaje[0],17,"Pago:              ");
      LCD_MESSG(u8aLCD_Mensaje,32);
      LCD_OK();
      u8aInstruccion[0] = 0x86;
      LCD_INST(u8aInstruccion);

      /*Introduccion del pago*/
      for(i=0; i<5; i++){
      u8aBuffer_Keyboard[i] = Keyboard();
        if (u8aBuffer_Keyboard[i] != 0x0F && u8aBuffer_Keyboard[i] != 0x0E){
               u8aBuffer_Keyboard[i] = ASCII(u8aBuffer_Keyboard[i]);
        }else{
                u8aBuffer_Keyboard[i] = 0x20;
               i += (5-i);
        }
      }

        u16Pago = ASCIITo_Decimal(&u8aBuffer_Keyboard[0],4);
        for(i=0; i<10; i++){
               u8aBuffer_Keyboard[i] = 0x00;
        }

        LCD_Clean();
```

```
          /*Confirmar Pago*/
          snprintf((char*)&u8aLCD_Mensaje[0],17,"Confirma: %d     ",u16Pago);
          LCD_MESSG(u8aLCD_Mensaje,32);
          LCD_JUMP();
          snprintf((char*)&u8aLCD_Mensaje[0],17,"Regresar      OK");
          LCD_MESSG(u8aLCD_Mensaje,32);

       u8Opcion = Keyboard();
         while(u8Opcion != 0x0F && u8Opcion != 0x0E){
                u8Opcion = Keyboard();
         }

         if(u8Opcion == 0x0F){
               goto Regresar;              //Regresar a ingreso de pago
         }else if(u8Opcion == 0x0E){
                LCD_Clean();
               if (Descontar_Card(u16Pago)){     //Descontar Saldo y Verificar
suficiente
                Generate_Yep(u16Pago,u8Restaurant);  //Generar YEP
                LCD_JUMP();
                LCD_OK();
               }else{
                snprintf((char*)&u8aLCD_Mensaje[0],17,"Saldo            ");
//Mensaje de Saldo Insuficiente
                LCD_MESSG(u8aLCD_Mensaje,32);
                LCD_JUMP();
                snprintf((char*)&u8aLCD_Mensaje[0],17,"Insuficiente  OK");
                LCD_MESSG(u8aLCD_Mensaje,32);
                }
              u8Opcion = Keyboard();
                while(u8Opcion != 0x0E){
                u8Opcion = Keyboard();
                }

         }
          /*Reembolso*/
  }else if(u8Opcion == 0x0B ){
          RegresarB:   LCD_Clean();
          snprintf((char*)&u8aLCD_Mensaje[0],17,"SUP:            ");
          LCD_MESSG(u8aLCD_Mensaje,32);
          LCD_OK();
          u8aInstruccion[0] = 0x85;
          LCD_INST(u8aInstruccion);
          u16Pago = 0;
          /* Ingresar codigo de reembolso */
          for(i=0; i<10; i++){
                u8aBuffer_Keyboard[i] = Keyboard();
                 if (u8aBuffer_Keyboard[i] != 0x0F && u8aBuffer_Keyboard[i] != 0x0E){
                        u8aBuffer_Keyboard[i] = ASCII(u8aBuffer_Keyboard[i]);
                        }else{
                        u8aBuffer_Keyboard[i] = 0x20;
                        i += (10-i);
                 }
          }

         u16SUP = ASCIITo_Octal(&u8aBuffer_Keyboard[0],9);
         for(i=0; i<10; i++){
                u8aBuffer_Keyboard[i] = 0x00;
         }
```

```
            LCD_Clean();
            /* Confirmar codigo*/
            snprintf((char*)&u8aLCD_Mensaje[0],17,"Confirm: %lo     ",u16SUP);
            LCD_MESSG(u8aLCD_Mensaje,32);
            LCD_JUMP();
            snprintf((char*)&u8aLCD_Mensaje[0],17,"Regresar      OK");
            LCD_MESSG(u8aLCD_Mensaje,32);

            u8Opcion = Keyboard();
            while(u8Opcion != 0x0F && u8Opcion != 0x0E){
                  u8Opcion = Keyboard();
            }

            if(u8Opcion == 0x0F){
                  goto RegresarB;  //Regresar ingreso de codigo
               }else if(u8Opcion == 0x0E){
                  LCD_Clean();
                  Decode_SUP(u16SUP,u8Restaurant); //Decodificar SUP
                        LCD_OK();
                        LCD_MESSG(u8aLCD_Mensaje,4);

                u8Opcion = Keyboard();
                  while(u8Opcion != 0x0E){
                        u8Opcion = Keyboard();
                  }

            }

      }
      LCD_Clean();
            }
}
```

Anexo E

Archivos .h

SPI.h

```
#ifndef SPI_H
#define SPI_H

#include "spi_config.h"
#include <stdint.h>

void spi_init()
{
	SPI_DDR = (1<<SPI_MOSI)|(1<<SPI_SCK)|(1<<SPI_SS);
	SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);//prescaler 16
}


uint8_t spi_transmit(uint8_t data)
{
	SPDR = data;
	while(!(SPSR & (1<<SPIF)));

	return SPDR;
}


#define ENABLE_CHIP() (SPI_PORT &= (~(1<<SPI_SS)))
#define DISABLE_CHIP() (SPI_PORT |= (1<<SPI_SS))

#endif
```

Spi_config.h

```
#ifndef SPI_CONFIG_H
#define SPI_CONFIG_H

#include <avr/io.h>


#define SPI_DDR		DDRB
#define SPI_PORT		PORTB
#define SPI_PIN		PINB
#define SPI_MOSI		PB5
#define SPI_MISO		PB6
#define SPI_SS		PB4
#define SPI_SCK		PB7

#endif
```

CardDetect.h

```
#ifndef CARDDETECT_H_
#define CARDDETECT_H_

#include <avr/io.h>


unsigned char byte;
unsigned char str[MAX_LEN];
```

```c
unsigned char write[MAX_LEN];
unsigned char u8aLCD_Mensaje[16];
unsigned char u8aRedo[1] = {0x80};

/* Resetar tarjeta */
void Reset_Card(){
	MFRC522_Init();
}

/* Detectar Tarjeta*/
void Detect_Card(void){
	byte = MFRC522_Request(PICC_REQALL,str);

	while(byte != MI_OK)
	{
		byte = MFRC522_Request(PICC_REQALL,str);
		LCD_INST(u8aRedo);
		snprintf((char*)&u8aLCD_Mensaje[0],17,"Insertar Tarjeta");
		LCD_MESSG(u8aLCD_Mensaje,32);
	}
	LCD_Clean();
}

/* Leer bloque de saldo de la tarjeta*/
void Read_Card(uchar bloque){
	unsigned char i;

	Reset_Card();
	readDataASCII(bloque);

	for(i=0; i<4; i++){
		read[i] = 0x20;
	}
}

/*Escribi al bloque de saldo de la tarjeta */
void Write_Card(void){

	Reset_Card();
	writeTagBlockMemory(&write[0]);
}

#endif /* CARDDETECT_H_ */
```

YEP.h

```c
#include "YEP.h"

/* Conversion de ASCII a Decimal */
unsigned int ASCIITo_Decimal(unsigned char *ascii,unsigned char size){
	unsigned int decimal = 1, convertion = 0;
	unsigned char u8index;

	ascii += size;

	for(u8index = size; u8index <= size; u8index--){
		if (*ascii < 0x30 || *ascii > 0x39){
			convertion += 0;
			ascii--;
```

```
		}else {
			convertion += ((*ascii-0x30)*decimal);
			decimal *= 10;
			ascii--;
		}
	}
	return convertion;
}

/*Conversion de ASCII a Octal*/
unsigned int long ASCIITo_Octal(unsigned char *ascii,unsigned char size){
	unsigned int long octal = 1;
	unsigned int long convertion = 0;
	unsigned char u8index;

	ascii += size;

	for(u8index = size; u8index <= size; u8index--){
		if (*ascii < 0x30 || *ascii > 0x37){
			convertion += 0;
			ascii--;
			}else {
			convertion += ((*ascii-0x30)*octal);
			octal *= 8;
			ascii--;
		}
	}
	return convertion;
}

/* Descontar pago de la tarjeta*/
unsigned char Descontar_Card (unsigned int u16Pago){
	unsigned int u16Saldo;
	unsigned char write[16];
Detect_Card();
Read_Card(4);
	u16Saldo = ASCIITo_Decimal(&read[0],15);
	if(u16Saldo < u16Pago){

		return 0;
	}else{
	u16Saldo = u16Saldo - u16Pago;

	snprintf((char*)&write[0],16,"    %d              ",u16Saldo);
	Reset_Card();
	Detect_Card();
	writeTagBlockMemory(&write[0]);
	return 1;
}
	}

/* Generar codigo YEP*/
void Generate_Yep(unsigned int long u16Pago,unsigned char u8Restaurant){
	unsigned int long YEP = 0;
	unsigned char RTC[7], Hour;
	Get_RTC(RTC);
	switch(u8Restaurant){
		case 1: YEP = u16Pago*12905; break;
		case 2: YEP = u16Pago*13981; break;
		case 3: YEP = u16Pago*11650; break;
```

```
        }
        Hour = ((RTC[2]>>4)*10);
        Hour = (Hour + (RTC[2]&0x0F));

        snprintf((char*)&u8aLCD_Mensaje[0],17,"YEP: 1%lX%d%.2X
",YEP,RTC[3],Hour*10);
        LCD_MESSG(u8aLCD_Mensaje,32);
}

/* Decodificar SUP y Añadir saldo a la tarjeta*/
void  Decode_SUP(unsigned int long u16SUP,unsigned char u8Restaurant){
        unsigned int long SUP = 0;
        unsigned int  u16Saldo;
        unsigned char write[16];

                switch(u8Restaurant){
                        case 1: SUP = u16SUP/12905; break;
                        case 2: SUP = u16SUP/13981; break;
                        case 3: SUP = u16SUP/11650; break;
                }

        Detect_Card();
        Read_Card(4);
        u16Saldo = ASCIITo_Decimal(&read[0],15);

        u16Saldo = u16Saldo + SUP;

        snprintf((char*)&write[0],16,"    %d              ",u16Saldo);
        Reset_Card();
        Detect_Card();
        writeTagBlockMemory(&write[0]);

        Detect_Card();
        Read_Card(4);
        u16Saldo = ASCIITo_Decimal(&read[0],15);

        snprintf((char*)&u8aLCD_Mensaje[0],17,"Saldo: %d            ",u16Saldo);
        LCD_MESSG(u8aLCD_Mensaje,32);
}
```

I2C_ATmega8.h

```
#include <avr/io.h>

void Configuration_I2C (void);
void Start_I2C (void);
void Stop_I2C (void);
void Send_I2C (char Data_I2C);
char Get_I2C (void);
void Send_RTC (char Data_String[]);
void Reset_Register_Pointer (char x);
void Get_RTC (char Data_RTC[]);
char *Get_Data_RTC;


void Configuration_I2C (void) {
        //Set the SCL to 57.6Khz
        TWBR = 0xA3;
        TWSR = 0x00;
        //Enable TWI module
```

```
        TWCR = (1<<TWEN);
}

void Start_I2C (void) {
        //Start condition
        TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
        //Waiting the start condition
        while ((TWCR & (1<<TWINT)) == 0);
}


void Stop_I2C (void) {
        //Stop condition
        TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
}

void Send_I2C (char Data_I2C) {
        //Get value for I2C bus
        TWDR = Data_I2C;
        //Clear the interrupt
        TWCR = (1<<TWINT)|(1<<TWEN);
        //Waiting to send Data_I2C
        while ((TWCR & (1<<TWINT)) == 0);
}

char Get_I2C (void) {
        //Clear the interrupt
        TWCR = (1<<TWINT)|(1<<TWEN);
        //Waiting to receive Data_I2C
        while ((TWCR & (1<<TWINT)) == 0);
        //Get out the value
        return TWDR;
}

void Send_RTC (char Data_String[]) {
        //Send initial conditions
        char x;
        Start_I2C();
        Send_I2C(0xD0);
        Send_I2C(0x00);
        //Send string
        for(x=0; x<8; x++) {
                Send_I2C(Data_String[x]);
        }
        Stop_I2C();
}

void Reset_Register_Pointer (char x){
        //Reset vector of register pointer to 0x00 address
        Start_I2C();
        Send_I2C(0xD0);
        Send_I2C(x);
        Stop_I2C();
}

void Get_RTC (char Data_RTC[]) {
        //Set initial conditions
        char x;
        Get_Data_RTC=Data_RTC;
        for (x=0; x<7; x++, Get_Data_RTC++) {
```

```
        Reset_Register_Pointer(x);
        Start_I2C();
        Send_I2C(0xD1);
        *Get_Data_RTC=Get_I2C();
        }
        Stop_I2C();
}
```

LCDSM.h

```
#ifndef LCDSM_H_
#define LCDSM_H_

#define RS_HIGH PORTD |= (1<<PIND4)
#define RS_LOW PORTD &= ~(1<<PIND4)
#define EN_HIGH PORTD |= (1<<PIND5)
#define EN_LOW PORTD &= ~(1<<PIND5)
#define LOWERNIBBLE 0x0F
#define TIME (200)

void(*ptrfuncLCD)(void);
void LCD_Idle (void);
void LCD_WriteNibble (void);
void LCD_Wait (void);
unsigned int u16Time = TIME;
unsigned char *gptrMessage;
unsigned char u8Size_Message;
unsigned char gFlag;
unsigned char u8Temp;


void LCD_Idle (void)
{
        gFlag = 1;
}

void LCD_WriteNibble (void){


        if (u8Size_Message & 0x1){
                u8Temp = (*gptrMessage++) & LOWERNIBBLE;
                }else{
                u8Temp = (*gptrMessage >> 4) & LOWERNIBBLE;
        }
        EN_HIGH;
        PORTD = (PORTD & ~(0xF)) | (u8Temp & 0xF);
        ptrfuncLCD = LCD_Wait;
}

void LCD_Wait (void){


        if (!u16Time--){
                EN_LOW;
                u16Time = TIME;
                ptrfuncLCD = LCD_WriteNibble;
                if (!(--u8Size_Message)){
                        ptrfuncLCD = LCD_Idle;
```

```
            }
        }
}

#endif /* LCDSM_H_ */
```

LCD_Functions.h

```
#ifndef LCDFUNCTIONS_H_
#define LCDFUNCTIONS_H_

#include "LCDSM.h"

char Keyboard (void);
const unsigned char u8aInitLCD[6]={0x02, 0x28, 0x0C, 0x06, 0x01, 0x80};


void LCD_HANDLER(void){
        ptrfuncLCD();
}

unsigned char Get_LCDStatus(void){
        return gFlag;
}


void LCD_Write (unsigned char *u8Char, unsigned char u8Size){
        gFlag = 0;
        gptrMessage = u8Char;
        u8Size_Message = u8Size;
        ptrfuncLCD = LCD_WriteNibble;
        for (;Get_LCDStatus()==0;)
        {
                LCD_HANDLER();
        }
}

void LCD_INIT (void){

        RS_LOW;
        DDRD=0x3F;
        LCD_Write((char*)&u8aInitLCD[0],(sizeof(u8aInitLCD) << 1));
}


void LCD_DATA (unsigned char u8Data){

        RS_HIGH;

        LCD_Write(&u8Data,(sizeof(u8Data) << 1));
}

void LCD_INST (unsigned char u8Instruction[]){

        RS_LOW;
```

```
	LCD_Write(&u8Instruction[0],0x02);
}

void LCD_MESSG (unsigned char u8aMessg[],unsigned char Size){

	RS_HIGH;

	LCD_Write(&u8aMessg[0],Size);
}


void BCD_2_ASCII (char Data){

	char x,y;
	x=(((Data&0xF0)>>4)|0x30);
	LCD_DATA(x);
	y=((Data&0x0F)|0x30);
	LCD_DATA(y);

}

void ASCII (char Data){
if (Data<0x0A) {
	Data += 0x30;
}
else {
	Data += 0x37;
}
LCD_DATA(Data);
}

char Keyboard (void){
char x,y;
PORTD=0xF0;
while(PORTD==0xF0);
x=PORTD;
PORTD=0x0F;
while(PORTD==0x0F);
y=PORTD;
x |=y;
switch(x){
	case 0x77: y=0x01; break;
	case 0x7B: y=0x02; break;
	case 0x7D: y=0x03; break;
	case 0x7E: y=0x0A; break;
	case 0xB7: y=0x04; break;
	case 0xBB: y=0x05; break;
	case 0xBD: y=0x06; break;
	case 0xBE: y=0x0B; break;
	case 0xD7: y=0x07; break;
	case 0xDB: y=0x08; break;
	case 0xDD: y=0x09; break;
	case 0xDE: y=0x0C; break;
	case 0xE7: y=0x0F; break;
	case 0xEB: y=0x00; break;
	case 0xED: y=0x0E; break;
	case 0xEE: y=0x0D; break;
}
return (y);
}
```

```
#endif /* LCDFUNCTIONS_H_ */
```

MFRC522_NFC.h

```
#ifndef MFRC522_NFC_H_
#define MFRC522_NFC_H_

#include "spi.h"

#define       uchar  unsigned char
#define       uint   unsigned int

//data array maxium length
#define MAX_LEN 16

//MF522 command bits
#define PCD_IDLE              0x00               //NO action; cancel current
commands
#define PCD_AUTHENT           0x0E               //verify password key
#define PCD_RECEIVE           0x08               //receive data
#define PCD_TRANSMIT          0x04               //send data
#define PCD_TRANSCEIVE        0x0C               //send and receive data
#define PCD_RESETPHASE        0x0F               //reset
#define PCD_CALCCRC           0x03               //CRC check and caculation

//Mifare_One card command bits
#define PICC_REQIDL           0x26               //Search the cards that not into
sleep mode in the antenna area
#define PICC_REQALL           0x52               //Search all the cards in the
antenna area
#define PICC_ANTICOLL         0x93               //prevent conflict
#define PICC_SElECTTAG        0x93               //select card
#define PICC_AUTHENT1A        0x60               //verify A password key
#define PICC_AUTHENT1B        0x61               //verify B password key
#define PICC_READ             0x30               //read
#define PICC_WRITE            0xA0               //write
#define PICC_DECREMENT        0xC0               //deduct value
#define PICC_INCREMENT        0xC1               //charge up value
#define PICC_RESTORE          0xC2               //Restore data into buffer
#define PICC_TRANSFER         0xB0               //Save data into buffer
#define PICC_HALT             0x50               //sleep mode

//THe mistake code that return when communicate with MF522
#define MI_OK                 0
#define MI_NOTAGERR           1
#define MI_ERR                2

//------------------MFRC522 register ----------------------
//Page 0:Command and Status
#define     Reserved00            0x00
#define     CommandReg            0x01
#define     CommIEnReg            0x02
#define     DivlEnReg             0x03
#define     CommIrqReg            0x04
#define     DivIrqReg             0x05
#define     ErrorReg              0x06
#define     Status1Reg            0x07
```

```
#define     Status2Reg            0x08
#define     FIFODataReg           0x09
#define     FIFOLevelReg          0x0A
#define     WaterLevelReg         0x0B
#define     ControlReg            0x0C
#define     BitFramingReg         0x0D
#define     CollReg               0x0E
#define     Reserved01            0x0F
//Page 1:Command
#define     Reserved10            0x10
#define     ModeReg               0x11
#define     TxModeReg             0x12
#define     RxModeReg             0x13
#define     TxControlReg          0x14
#define     TxAutoReg             0x15
#define     TxSelReg              0x16
#define     RxSelReg              0x17
#define     RxThresholdReg        0x18
#define     DemodReg              0x19
#define     Reserved11            0x1A
#define     Reserved12            0x1B
#define     MifareReg             0x1C
#define     Reserved13            0x1D
#define     Reserved14            0x1E
#define     SerialSpeedReg        0x1F
//Page 2:CFG
#define     Reserved20            0x20
#define     CRCResultRegM         0x21
#define     CRCResultRegL         0x22
#define     Reserved21            0x23
#define     ModWidthReg           0x24
#define     Reserved22            0x25
#define     RFCfgReg              0x26
#define     GsNReg                0x27
#define     CWGsPReg          0x28
#define     ModGsPReg             0x29
#define     TModeReg              0x2A
#define     TPrescalerReg         0x2B
#define     TReloadRegH           0x2C
#define     TReloadRegL           0x2D
#define     TCounterValueRegH     0x2E
#define     TCounterValueRegL     0x2F
//Page 3:TestRegister
#define     Reserved30            0x30
#define     TestSel1Reg           0x31
#define     TestSel2Reg           0x32
#define     TestPinEnReg          0x33
#define     TestPinValueReg       0x34
#define     TestBusReg            0x35
#define     AutoTestReg           0x36
#define     VersionReg            0x37
#define     AnalogTestReg         0x38
#define     TestDAC1Reg           0x39
#define     TestDAC2Reg           0x3A
#define     TestADCReg            0x3B
#define     Reserved31            0x3C
#define     Reserved32            0x3D
#define     Reserved33            0x3E
#define     Reserved34            0x3F
```

```
/* Escribir a bloque de la tarjeta */
void writeTagBlockMemory(uchar *data4){
       uchar serNum[5];
       //clave por default de la tarjeta
       uchar sectorX_KeyA[]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};
       uchar i,j,ij;
       uchar status;
    uchar str[MAX_LEN];
       for(ij = 0; ij < 10; ij++){
              //Buscar y activar tarjeta
              status = MFRC522_Request(PICC_REQIDL, str);
              if (status == MI_OK){  }
              //Obtener numero de serie de la tarjeta
              status = MFRC522_Anticoll(str);
              for(i=0; i<5; i++){   serNum[i]=str[i];  }
              if (status == MI_OK){  }
              status = MFRC522_SelectTag(serNum);

              status = MFRC522_Auth(0x60,4,sectorX_KeyA,serNum);
              if(status==MI_OK){
                     writeTagBlockData(4,data4);
                     }
              MFRC522_Halt();
                                                        }
              }

/* Escribir dato al bloque  */
void writeTagBlockData(int block, uchar *data) {
       uchar status;
       char string[31];
       status = MFRC522_Write(block, data);
}



/* Leer dato en ASCII de la tarjeta       */
void readDataASCII(uchar bloque){
       uchar serNum[5];
       uchar ij;
       uchar sectorX_KeyA[]={0xFF,0xFF,0xFF,0xFF,0xFF,0xFF};
       uchar i,j;
       uchar status;
for(ij = 0; ij < 10; ij++){
              status = MFRC522_Request(PICC_REQIDL, read);
              if (status == MI_OK){  }
               status = MFRC522_Anticoll(read);
              for(i=0; i<5; i++){   serNum[i]=read[i];  }
              if (status == MI_OK){
                                                                  }
              status = MFRC522_SelectTag(serNum);
              status = MFRC522_Auth(0x60,bloque,sectorX_KeyA,serNum);
                     status = MFRC522_Read(bloque, read);
              MFRC522_Halt();
                      }}



/* Inicializar MFRC522 */
void MFRC522_Init(void) {
```

```
        MFRC522_Reset();
        //Timer: TPrescaler*TreloadVal/6.78MHz = 24ms
    Write_MFRC522(TModeReg,       0x8D);              //Tauto=1; f(Timer) =
6.78MHz/TPreScaler
    Write_MFRC522(TPrescalerReg,0x3E);                //TModeReg[3..0] +
TPrescalerReg
    Write_MFRC522(TReloadRegL,    30     );
    Write_MFRC522(TReloadRegH,    0      );
        Write_MFRC522(TxAutoReg,   0x40);              //100%ASK
        Write_MFRC522(ModeReg,            0x3D);              //CRC initilizate value
0x6363
        AntennaOn();                      }

/* Escribir byte a MFRC522 */
void Write_MFRC522(uchar reg, uchar data) {
              ENABLE_CHIP();
              spi_transmit((reg<<1)&0x7E);
              spi_transmit(data);
              DISABLE_CHIP();            }

/* Leer byte a MFRC522     */
uchar Read_MFRC522(uchar reg) {
        uchar val;
        ENABLE_CHIP();
        spi_transmit(((reg<<1)&0x7E)|0x80);
        val = spi_transmit(0x00);
        DISABLE_CHIP();
        return val;                             }

/* Poner a uno bit/s de algun registro del MFRC522           */
void SetBitMask(uchar reg, uchar mask)  {
    uchar tmp;
    tmp = Read_MFRC522(reg);
    Write_MFRC522(reg, tmp | mask);      }      // set bit mask

/*Limpiar registros del MFRC522          */
void ClearBitMask(uchar reg, uchar mask) {
    uchar tmp;
    tmp = Read_MFRC522(reg);
    Write_MFRC522(reg, tmp & (~mask));    }  // clear bit mask

/* Encender antena del modulo            */
void AntennaOn(void) {
        uchar temp;
        temp = Read_MFRC522(TxControlReg);
        if (!(temp & 0x03)){  SetBitMask(TxControlReg, 0x03);  } }

/* Apagar antena del modulo*/
void AntennaOff(void){
        ClearBitMask(TxControlReg, 0x03);
          }

/* Resetear modulo                */
void MFRC522_Reset(void){  Write_MFRC522(CommandReg, PCD_RESETPHASE);  }

/* Buscar y activar tarjeta*/
uchar MFRC522_Request(uchar reqMode, uchar *TagType) {
        uchar status;
        uint backBits;
```

```
        Write_MFRC522(BitFramingReg, 0x07);
        TagType[0] = reqMode;
        status = MFRC522_ToCard(PCD_TRANSCEIVE, TagType, 1, TagType, &backBits);
        if ((status != MI_OK) || (backBits != 0x10)){  status = MI_ERR;  }
        return status;                                }

/* Comunicacion entre MFRC522 y tarjeta  */
uchar MFRC522_ToCard(uchar command, uchar *sendData, uchar sendLen, uchar *backData,
uint *backLen){
    uchar status = MI_ERR;
    uchar irqEn =   0x00;
    uchar waitIRq = 0x00;
    uchar lastBits;
    uchar n;
    uint i;
    switch (command) {
        case PCD_AUTHENT:  {      //Verificar clave de tarjeta
                    irqEn = 0x12;
                    waitIRq = 0x10;
                    break;              }
              case PCD_TRANSCEIVE:{      //Enviar datos a FIFO
                    irqEn = 0x77;
                    waitIRq = 0x30;
                    break;              }
              default:     break;       }
    Write_MFRC522(CommIEnReg, irqEn|0x80);      //Activar interrupciones
    ClearBitMask(CommIrqReg, 0x80);                   //Limpiar interruociones
    SetBitMask(FIFOLevelReg, 0x80);                   //inicializar FIFO
        Write_MFRC522(CommandReg, PCD_IDLE);     //Enviar a modo Idle
        //Escribir datos al FIFO
    for (i=0; i<sendLen; i++){     Write_MFRC522(FIFODataReg, sendData[i]);   }
        Write_MFRC522(CommandReg, command);
    if (command == PCD_TRANSCEIVE){   SetBitMask(BitFramingReg, 0x80);     }
        i = 2000;
    do {

        n = Read_MFRC522(CommIrqReg);
        i--;
    }while ((i!=0) && !(n&0x01) && !(n&waitIRq));
    ClearBitMask(BitFramingReg, 0x80);
    if (i != 0) {
        if(!(Read_MFRC522(ErrorReg) & 0x1B))
        {
            status = MI_OK;
            if (n & irqEn & 0x01){       status = MI_NOTAGERR;      }
            if (command == PCD_TRANSCEIVE){
                    n = Read_MFRC522(FIFOLevelReg);
                    lastBits = Read_MFRC522(ControlReg) & 0x07;
                if (lastBits){    *backLen = (n-1)*8 + lastBits;   }
                else{      *backLen = n*8;   }
                if (n == 0){      n = 1;   }
                if (n > MAX_LEN){   n = MAX_LEN;      }
                            //leer datos del fifo
                for (i=0; i<n; i++){     backData[i] = Read_MFRC522(FIFODataReg);
        }
            }
        }
        else{ status = MI_ERR;     }
    }
    return status;                                }
```

```
/* Leer numero de serie de la tarjeta y verificarlo*/
uchar MFRC522_Anticoll(uchar *serNum){
    uchar status;
    uchar i;
       uchar serNumCheck=0;
    uint unLen;
       Write_MFRC522(BitFramingReg, 0x00);
    serNum[0] = PICC_ANTICOLL;
    serNum[1] = 0x20;
    status = MFRC522_ToCard(PCD_TRANSCEIVE, serNum, 2, serNum, &unLen);
    if (status == MI_OK){
             //Verificar numero de serie
             for (i=0; i<4; i++){ serNumCheck ^= serNum[i];  }
             if (serNumCheck != serNum[i]){    status = MI_ERR;    }
    }
    return status;                              }

/* Calculo del CRC */
void CalulateCRC(uchar *pIndata, uchar len, uchar *pOutData){
    uchar i, n;
    ClearBitMask(DivIrqReg, 0x04);
    SetBitMask(FIFOLevelReg, 0x80);
    for (i=0; i<len; i++){ Write_MFRC522(FIFODataReg, *(pIndata+i));   }
    Write_MFRC522(CommandReg, PCD_CALCCRC);
    i = 0xFF;
    do {
        n = Read_MFRC522(DivIrqReg);
        i--;
    }while ((i!=0) && !(n&0x04));
    pOutData[0] = Read_MFRC522(CRCResultRegL);
    pOutData[1] = Read_MFRC522(CRCResultRegM);                      }

/* Seleccion de tarjeta y lectura de tamaño de tarjeta        */
uchar MFRC522_SelectTag(uchar *serNum) {
    uchar i;
       uchar status;
       uchar size;
    uint recvBits;
    uchar buffer[9];
    buffer[0] = PICC_SElECTTAG;
    buffer[1] = 0x70;
    for (i=0; i<5; i++){  buffer[i+2] = *(serNum+i);  }
       CalulateCRC(buffer, 7, &buffer[7]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, buffer, 9, buffer, &recvBits);
    if ((status == MI_OK) && (recvBits == 0x18)){     size = buffer[0];   }
    else{     size = 0;     }
    return size;                                      }

/* Verificar clave de tarjeta            */
uchar MFRC522_Auth(uchar authMode, uchar BlockAddr, uchar *Sectorkey, uchar *serNum)
{
    uchar status;
    uint recvBits;
    uchar i;
       uchar buff[12];
    buff[0] = authMode;
    buff[1] = BlockAddr;
    for (i=0; i<6; i++){   buff[i+2] = *(Sectorkey+i);   }
    for (i=0; i<4; i++){   buff[i+8] = *(serNum+i);     }
```

```
    status = MFRC522_ToCard(PCD_AUTHENT, buff, 12, buff, &recvBits);
    if ((status != MI_OK) || (!(Read_MFRC522(Status2Reg) & 0x08))){  status =
MI_ERR;  }
    return status;
                                                    }

/* Leer datos de la tarjeta                     */
uchar MFRC522_Read(uchar blockAddr, uchar *recvData) {
    uchar status;
    uint unLen;
    recvData[0] = PICC_READ;
    recvData[1] = blockAddr;
    CalulateCRC(recvData,2, &recvData[2]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, recvData, 4, recvData, &unLen);
    if ((status != MI_OK) || (unLen != 0x90)) {  status = MI_ERR;  }
    return status;                                                        }

/* Escribir datos a bloque de la tarjeta        */
uchar MFRC522_Write(uchar blockAddr, uchar *writeData) {
    uchar status;
    uint recvBits;
    uchar i;
       uchar buff[18];
    buff[0] = PICC_WRITE;
    buff[1] = blockAddr;
    CalulateCRC(buff, 2, &buff[2]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 4, buff, &recvBits);
    if ((status != MI_OK) || (recvBits != 4) || ((buff[0] & 0x0F) != 0x0A)){  status
= MI_ERR;    }
    if (status == MI_OK){
        for (i=0; i<16; i++){   buff[i] = *(writeData+i);   }
        CalulateCRC(buff, 16, &buff[16]);
        status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 18, buff, &recvBits);
              if ((status != MI_OK) || (recvBits != 4) || ((buff[0] & 0x0F) !=
0x0A)){  status = MI_ERR; }
    }
    return status;                                                            }

/* Desactivar tarjeta (modo sleep)              */
void MFRC522_Halt(void){
       uchar status;
    uint unLen;
    uchar buff[4];
    buff[0] = PICC_HALT;
    buff[1] = 0;
    CalulateCRC(buff, 2, &buff[2]);
    status = MFRC522_ToCard(PCD_TRANSCEIVE, buff, 4, buff,&unLen);            }
```

Anexo F

Manual de Usuario

Sistema para agilizar el servicio de áreas de comida en centros comerciales

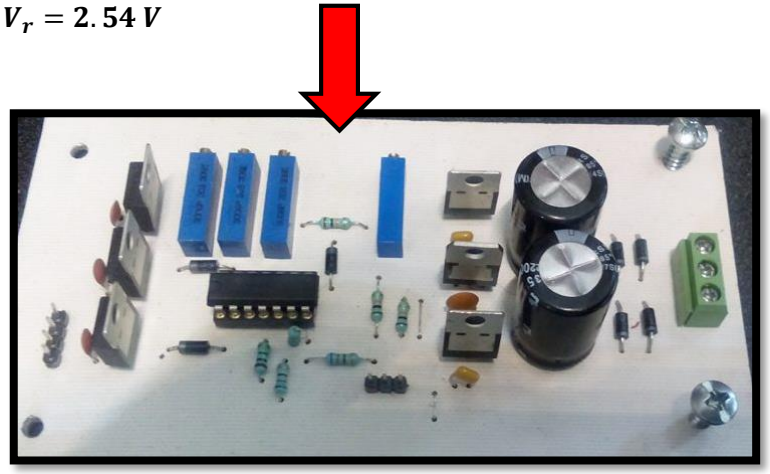
Perfil de usuario:

- Yam – Yam es un prototipo para instalar en el área de comida de un centro comercial, en un parque de gastronetas, en la cafetería de cualquier universidad privada y en cualquier lugar en donde más de un establecimiento ofrezca el servicio de comida rápida a la sociedad.
- Debe ser instalado cuidadosamente por personal autorizado con antecedentes técnicos, el mismo encargado de brindar mantenimiento y calibración al equipo.
- El cliente que desee ordenar su comida con ayuda de nuestro sistema debe tomar las siguientes precauciones y seguir los pasos que a continuación se describen para su correcto funcionamiento.

Máquina de prepago:

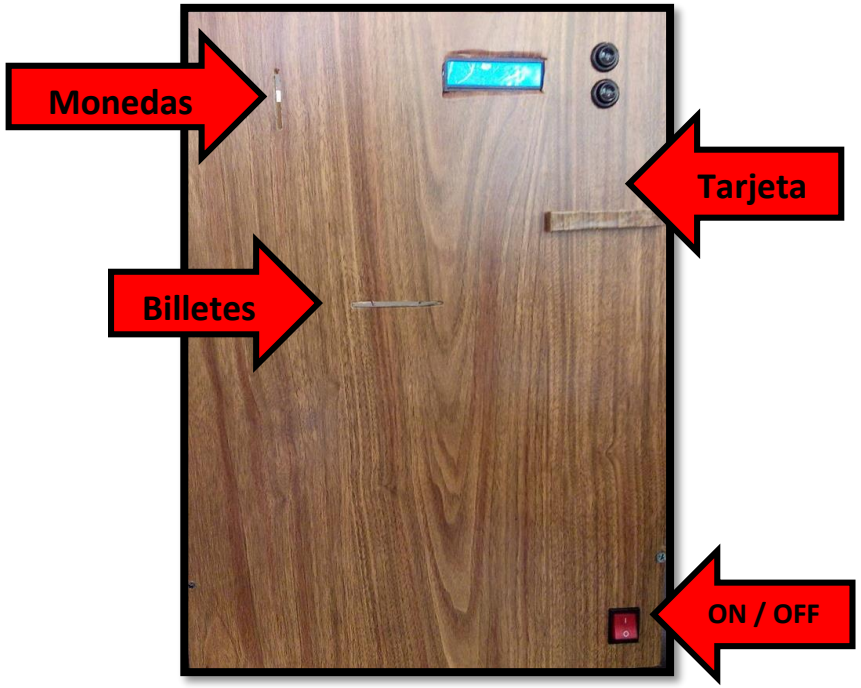
a) Instalación.

- Colocar la máquina en un lugar seguro, bajo techo, sin humedad, fuera del alcance de niños menores, con una distancia de por lo menos 10cm entre el mueble y la pared.
- Conectar a la corriente alterna: 120 – 127 V.
- Encender la máquina (interruptor en la esquina inferior derecha).
- Calibrar voltaje de referencia en el circuito de detección de monedas, resistencia variable de precisión independiente. $\boldsymbol{V_r = 2.54\ V}$



b) Usuario / Cliente.

- Insertar tarjeta en la ranura indicada en la parte superior derecha de la máquina (debajo de los botones).
- Para navegar en la interfaz se utilizan los dos botones negros a la derecha de la pantalla.
- Introducir efectivo: monedas y billetes por su respectiva ranura, verificando la suma correcta en la pantalla.

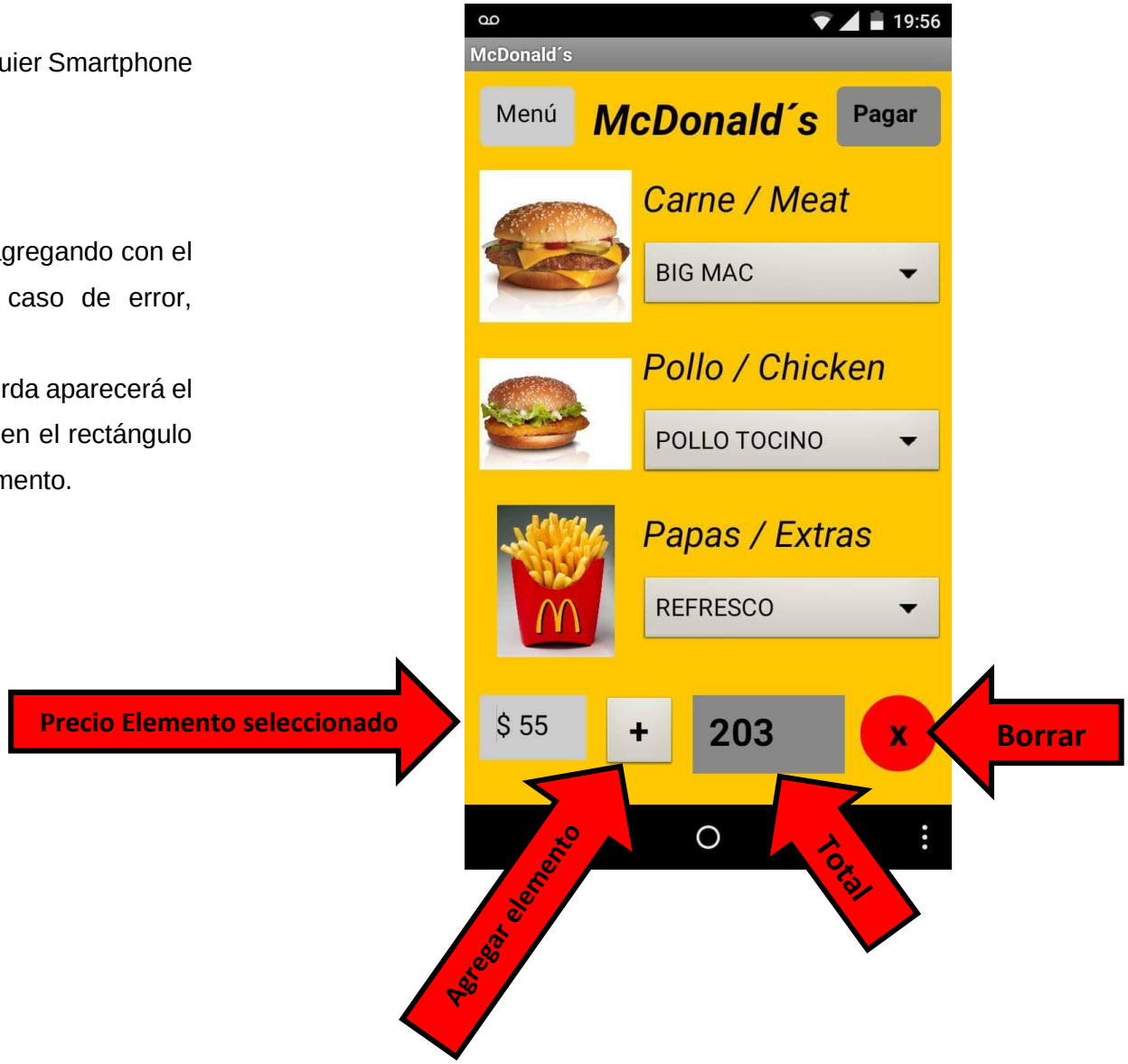


- Presionar el botón inferior para finalizar el proceso de abonar crédito, esperar hasta que aparezca su saldo actual para retirar su tarjeta.

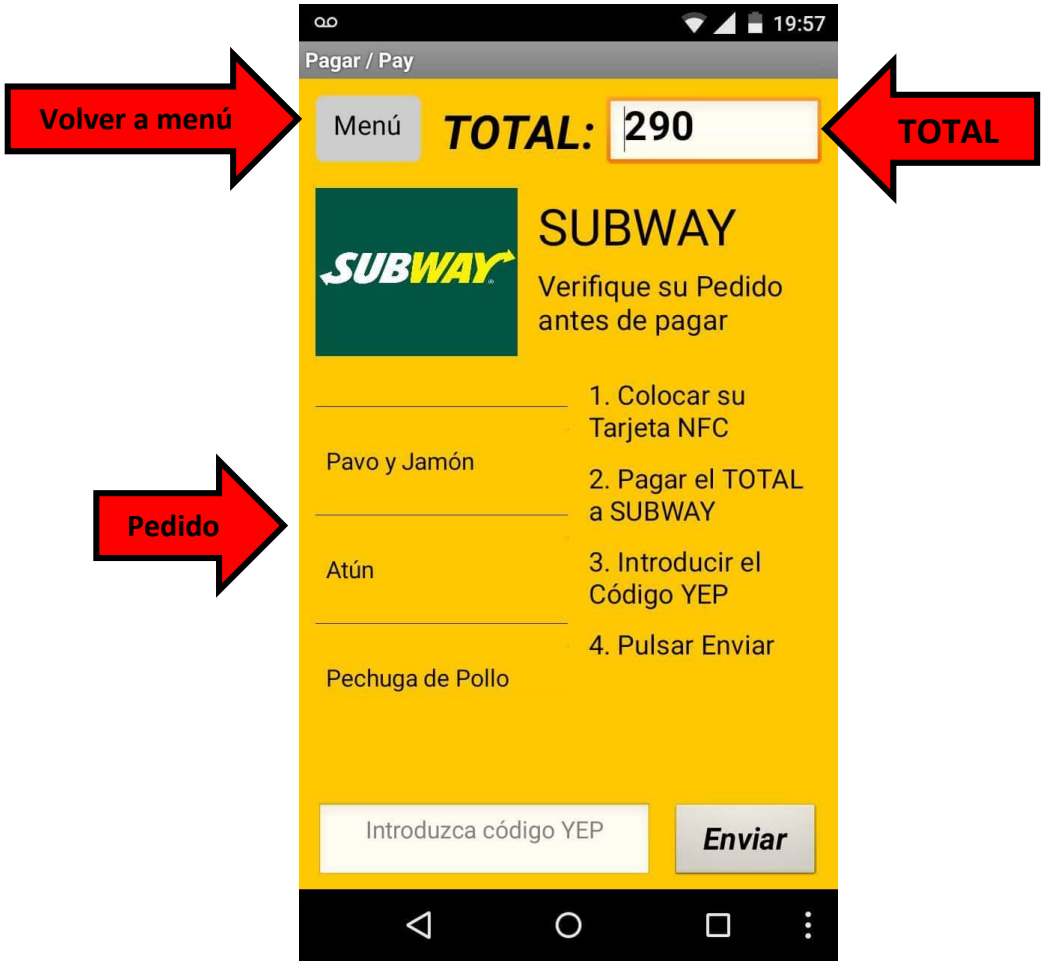
Aplicación Yam - Yam

- Instalar el archivo *"YamYam.apk"* en cualquier Smartphone con sistema operativo Android.
- Abrir Aplicación.
- Elegir restaurante.
- Realizar pedido, desplegando el menú y agregando con el botón [+], para eliminar el pedido en caso de error, presionar el botón (x).

  En el recuadro de la esquina inferior izquierda aparecerá el precio de cada elemento seleccionado, y en el rectángulo de la derecha el total a pagar hasta el momento.



- Al terminar el pedido de máximo 20 elementos presionar el botón “Pagar” en la esquina superior derecha del teléfono.



- Revisar que su pedido sea correcto en el lado izquierdo de la pantalla del celular.
- Introducir el código de seguridad “YEP” generado por el periférico que más adelante explicaremos, y pulsar enviar.
- Verificar el mensaje “Orden Enviada” en la pantalla.
- Opcional: si se tienen comentarios, se pueden escribir en el mismo recuadro, y para enviarlos al respectivo restaurante debe dejar pulsado el botón “Enviar” en la esquina inferior derecha por al menos tres segundos.

- Esperar la notificación de “Orden Lista”, cuando llegue, recoger su comida, identificándose con el mensaje del teléfono.

  - Para volver a la pantalla donde aparecen todos los pedidos, basta con tener activado el acelerómetro del celular, y agitar su teléfono.

Periféricos de cobro.

a) *Instalación*

- Colocar la máquina en la mesa o lo más próxima posible a esta, bajo techo y fuera del alcance de niños menores,
  - La idea es que este periférico sea parte del mueble (mesa).
- Conectar a la corriente alterna 120 – 127V.
- Configurar fecha y hora del RTC manualmente.
- Este dispositivo no debe ser desconectado por ningún motivo, en caso de que el RTC sea desprogramado, el encargado de mantenimiento deberá configurar la fecha y la hora manualmente y revisar sus respectivas baterías.

b) *Usuario / Cliente*

- Elegir: generar código “YEP (Código para pagar) o reembolsar (ingresando un código SUP).
- Elegir el restaurante.
- Colocar tarjeta en el lugar indicado, (debajo de la pantalla).
- Introducir total a pagar, o código SUP.
- Al generar el código YEP, o al aparecer su saldo actual, puede retirar su tarjeta.

.

Interfaz del restaurante.

- Instalar el emulador de Android gratuito “Bluestacks”
- Instalar el correspondiente archivo apk en el emulador Bluestacks:
  - McDonald´s.apk
  - Subway.apk
  - LaGorda.apk
- Conectar a internet el ordenador.
- Organizado por mesas aparecerán en pantalla cada uno de los pedidos realizados por los clientes, así como los comentarios justo arriba de los pedidos.

- Listo Mx: este botón sirve para enviar la notificación al celular de orden lista, y aparecerá un mensaje en el smartphone “Recoja su pedido”.
- El cliente recogerá su pedido identificándose con dicho mensaje que el restaurante borrará con el botón de Borrar Mx, después de entregar la comida.

- El botón “SUP” genera automáticamente un código de reembolso a partir de un código YEP erróneo.

- Al presionar el botón “TOTAL”, pedirá una contraseña de 4 caracteres, al validarla como correcta imprimirá en la pantalla el total vendido hasta ese momento del día.

Apéndices

Para mayor información de las tarjetas MF1S50YYX consulte:

Apéndices\MF1S50YYX.pdf

Para mayor información del módulo MFRC522 consulte:

Apéndices\MFRC522.pdf

Para mayor información del avr Atmega8 consulte:

Apéndices\atmel-2486-8-bit-avr-microcontroller-atmega8_l_datasheet.pdf

Para mayor información del avr Atmega16 consulte:

Apéndices\doc2466.pdf

Para mayor información de la norma ISO14443 consulte:

Apéndices\doc2056.pdf